UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

Control climático con lógica fuzzy de un prototipo de invernadero doméstico para el cultivo de setas

AUTOR: JOSÉ CARLOS SERRANO DOMINGO

TUTOR: EDUARDO QUILES CUCARELLA

Valencia, Septiembre 2017

AGRADECIMIENTOS

Especialmente a mi cuñado, por su dedicación y apoyo logístico y personal.

Al resto de mi familia por su colaboración y entusiasmo, tan importante en ciertos momentos para que las cosas sigan su curso.

Y como no a mi tutor del TFG, Don Eduardo Quiles Cucarella, por su asesoramiento continuo en el desarrollo del trabajo.

RESUMEN

En el presente trabajo planteamos, construimos y desarrollamos, un prototipo de invernadero doméstico al cual le aplicaremos un control climático normal en bucle cerrado, pero empleando una pequeña función de control fuzzy que regulará el caudal de aire de la bomba de nuestro sistema (buscando el rango óptimo de funcionamiento), para el cultivo de setas de ostra (Pleurotus ostreatus).

El alcance del proyecto abarca un marco a nivel personal y doméstico. Aunque no se descarta ampliarlo y mejorarlo para llegar a comercializar el prototipo en un futuro. Por ello analizamos al detalle los datos obtenidos de los cultivos experimentales y realizamos un presupuesto orientativo de los costes necesarios para su elaboración.

El proceso de desarrollo seguido en el trabajo se podría resumir de la siente manera:

1. Indagamos en diversas fuentes sobre el cultivo de setas y en especial sobre el cultivo de la seta de ostra (*Pleurotus ostreatus*) que es el tipo de seta que vamos a cultivar.
2. Indagamos sobre el uso de la lógica borrosa (Fuzzy logic) para saber de que manera podemos implementarla en nuestro invernadero doméstico.
3. Realizamos el pertinente modelado teórico previo del invernadero que vamos a fabricar.
4. Realizamos el montaje físico del invernadero.
5. Realizamos el modelado de nuestro sistema de control Fuzzy para el invernadero
6. Desarrollamos el programa en Arduino necesario para implementar el control climático en el invernadero fabricado.
7. Realizamos el cultivo experimental de setas de ostra (*Pleurotus ostreatus*) en el invernadero.
8. Registramos con Raspberry o PC externo los datos experimentales de cultivos de seta realizados, y efectuamos el posterior tratamiento de éstos con el programa matemático Octave para su estudio teórico.
9. Hacemos una recopilación de los problemas surgidos durante el desarrollo del proyecto y la solución empleada en cada caso.
10. Realizamos una breve crónica de las conclusiones a las que llegamos tras la realización del proyecto.

ÍNDICE GENERAL

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

Control climático con lógica fuzzy de un prototipo de invernadero doméstico para el cultivo de setas

Documento Nº 1: MEMORIA

AUTOR: JOSÉ CARLOS SERRANO DOMINGO

TUTOR: EDUARDO QUILES CUCARELLA

Valencia, Septiembre 2017

ÍNDICE MEMORIA

CAPÍTULO I: INTRODUCCIÓN

1.1 Antecedentes

Observando a nuestro alrededor nos damos cuenta de que estamos rodeados de multitud de cosas que llevan algún tipo de control instalado, y que nos mejoran el transcurso del día.

De este modo un buen día de otoño cuando estaba hablando con mi padre de las setas y de como le encantaría poder cultivarlas, se me enciende la bombilla y me pregunto porque no hacer un pequeño invernadero doméstico y así poder ayudarle.

Así que empiezo a pensar en utilizarlo como trabajo fin de grado.

Después de indagar un poco en el tema y consultando con varias personas, surge la idea final de desarrollar un control climático normal en bucle cerrado, pero aplicando una pequeña función de control fuzzy que regulará el caudal de aire de la bomba de nuestro sistema (buscando el rango óptimo de funcionamiento), en un prototipo de invernadero doméstico para el cultivo de setas de ostra. Elegimos esta clase de seta porque nos informamos de que es de las más sencillas de cultivar.

1.2 Objetivos del proyecto

Pues bien, son varios:

- Ser capaces de cultivar setas para consumo propio a nivel doméstico.
- Aprender a usar el Arduino, un artefacto muy empleado y necesario dado el entorno actual en el que se mueve el sector de la electrónica y la automática industrial en estos momentos.
- Emplear un entorno de control nuevo para mi, ya que la tecnología fuzzy no se estudia en el grado de electrónica y automática industrial. Y por contra resulta muy útil para controlar determinados procesos industriales.

  Es un tipo de control bastante empleado desde hace años en la industria y el cual es importante dominar.
- Otro desafío de este proyecto consiste en emplear una célula peltier como refrigerador para enfriar/calentar el prototipo de invernadero desarrollado.
- Cabe destacar también que un objetivo importante a llevar a cabo en el transcurso del proyecto es intentar emplear en la medida de lo posible únicamente software y hardware libre. Con ello se quiere reivindicar que es posible realizar tareas importantes sin la necesidad de pagar licencias para el uso de ciertos programas que no son accesibles para la mayoría de las personas.

Dado los objetivos introducimos una breve explicación de los siguientes conceptos:

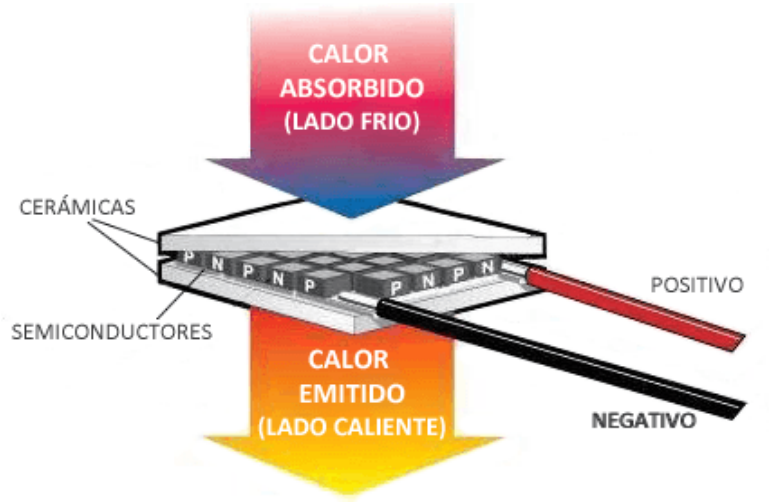
Efecto Peltier

Las células Peltier se pueden usar en el control de temperatura de diferentes aplicaciones y dispositivos de pequeñas dimensiones.

El desarrollo de circuitos de refrigeración basados en elementos Peltier, como alternativa a los

refrigerantes convencionales de origen orgánico (agua, amoniaco) y de origen inorgánico (CFC`s, HCFC`s) que resultan muy contaminantes, ha ido en aumento en los últimos años. Estos presentan grandes ventajas respecto a los antiguos sistemas de climatización: por un lado, no es necesario el uso de combustibles dado que la única fuente de alimentación que se necesita es eléctrica y, además, el volumen ocupado es muy reducido en comparación con esos métodos.

El efecto Peltier consiste en el enfriamiento o calentamiento de una unión entre dos conductores distintos al pasar una corriente eléctrica por ella y depende exclusivamente de la composición y temperatura de la unión. Las células Peltier hacen de bombas de calor donde toman calor de un foco frío y lo ceden a un foco caliente.



Estos módulos termoeléctricos tienen ciertas ventajas, como son:

- La producción de frío y calor indistintamente invirtiendo la polaridad de la tensión aplicada.
- Son totalmente silenciosos y no producen vibraciones.
- Poseen una fácil variación de la potencia refrigerante actuando sobre la tensión de alimentación.
- No tienen elementos móviles como son compresores, condensadores, válvulas, etc.

También presentan inconvenientes. En el caso del uso de los módulos termoeléctricos para refrigeración es necesario disipar el calor del lado caliente de la celda. Por tanto los elementos críticos en este tipo de tecnología (como será nuestro caso) son los ventiladores y disipadores empleados para que haya un flujo de calor que robe toda esa energia generada en el foco caliente de la peltier.

En las aplicaciones con celdas Peltier, el factor principal a tener en cuenta para la elección de la celda es el rango de temperatura al cual deben trabajar los semiconductores. Las aplicaciones más extendidas están en un rango de hasta 50 W. En general, el rendimiento o coeficiente de operación (COP) es un factor que se debe mejorar en la actualidad ya que se manejan rendimientos en torno al 5-10% frente al 40-50% conseguido por los sistemas convencionales de ciclo de compresión.

Dicho rendimiento se mide como el cociente entre el calor neto transportado y la potencia necesaria para producir el flujo de calor:

$$COP = \frac{Q_{neto}}{P}$$

Para conseguir un aumento en el rendimiento es importante la geometría y las propiedades del semiconductor entre otros factores. Una de las desventajas de la termoelectricidad en comparación con otras tecnologías de refrigeración es el bajo COP, en cambio el coste de los equipos es bastante reducido.

En nuestro caso usaremos el efecto peltier para generar frio o calor dentro del habitáculo del invernadero según convenga.

Arduino

Arduino es una compañía de hardware libre y una comunidad tecnológica que diseña y manufactura placas de desarrollo de hardware, compuestas por Microcontroladores, elementos pasivos y activos . Por otro lado las placas son programadas a través de un entorno de desarrollo (IDE), el cuál compila el código al modelo seleccionado de placa.

Toda la plataforma, incluyendo sus componentes de hardware (esquemáticos) y software, son liberados con licencia de código abierto que permite libertad de acceso a ellos.

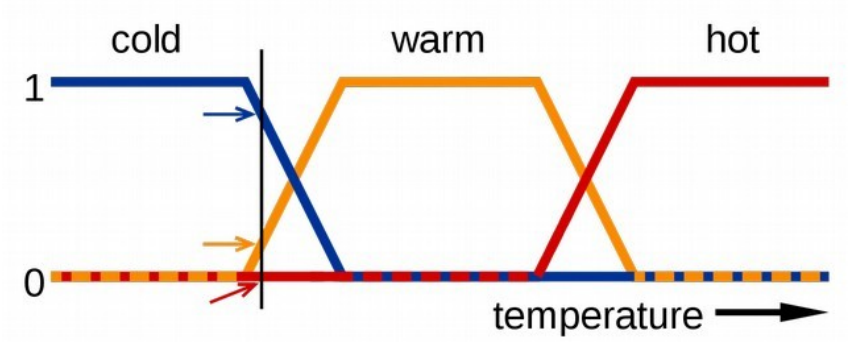
El *hardware* consiste en una placa de circuito impreso con un microcontrolador, usualmente Atmel AVR, puertos digitales y analógicos de entrada/salida, los cuales pueden conectarse a placas de expansión (shields), que amplían los funcionamientos de la placa Arduino. Asimismo, posee un puerto de conexión USB desde donde se puede alimentar la placa y establecer comunicación con el computador.

Por otro lado, el *software* consiste en un entorno de desarrollo (IDE) basado en el entorno de *processing* y lenguaje de programación basado en Wiring, así como en el cargador de arranque (*bootloader*) que es ejecutado en la placa. El microcontrolador de la placa se programa mediante un computador, usando una comunicación serial mediante un convertidor de niveles RS-232 a TTL serial.

El entorno de desarrollo integrado libre se puede descargar gratuitamente.

Lógica Fuzzy

Este tipo de lógica utiliza expresiones que no son totalmente ciertas ni totalmente falsas, es decir, es una lógica aplicada a conceptos subjetivos que pueden tomar un valor indeterminado de veracidad dentro de un conjunto de valores cuyos extremos son la verdad absoluta o la falsedad absoluta. Por así decirlo es una lógica que expresa la falta de definición del objeto al que se aplica.



Algunos ejemplos prácticos de aplicación en sistemas de la lógica fuzzy en la vida real:

- Sistemas de control de acondicionadores de aire
- Sistemas de foco automático en cámaras fotográficas
- Electrodomésticos familiares (frigoríficos, lavadoras...)
- Optimización de sistemas de control industriales
- Sistemas de escritura
- Mejora en la eficiencia del uso de combustible en motores
- Sistemas expertos del conocimiento (simular el comportamiento de un experto humano)
- Tecnología informática
- Bases de datos difusas: Almacenar y consultar información imprecisa. Para este punto, por ejemplo, existe el lenguaje FSQL.

1.3 Alcance

El alcance del proyecto abarca un marco a nivel únicamente personal y doméstico.

Es decir, no se pretende en ningún momento comercializar el producto ni sacar un beneficio económico del mismo.

Tras un análisis de los datos obtenidos y realizando el pertinente estudio económico de mercado se dejaría abierta la posibilidad de hacer un desarrollo pensado para la producción y venta del prototipo de invernadero objeto del proyecto.

1.4 Metodología empleada

El proceso cronológico que empleamos para llevar a cabo el desarrollo del proyecto fue el siguiente:

1. Indagamos en diversas fuentes sobre el cultivo de setas y en especial sobre el cultivo de la seta de ostra (*Pleurotus ostreatus*) que es el tipo de seta que vamos a cultivar.
2. Indagamos sobre el uso de la lógica borrosa (Fuzzy logic) para saber de que manera podemos implementarla en nuestro invernadero doméstico.
3. Realizamos el pertinente modelado teórico previo del invernadero que vamos a fabricar.
4. Realizamos el montaje físico del invernadero.
5. Realizamos el modelado de nuestro sistema de control Fuzzy para el invernadero
6. Desarrollamos el programa en Arduino necesario para implementar el control climático en el invernadero fabricado.
7. Realizamos el cultivo experimental de setas de ostra (*Pleurotus ostreatus*) en el invernadero.
8. Registramos con Raspberry o PC externo los datos experimentales de cultivos de seta

realizados, y efectuamos el posterior tratamiento de éstos con el programa matemático Octave para su estudio teórico.

9. Hacemos una recopilación de los problemas surgidos durante el desarrollo del proyecto y la solución empleada en cada caso.
10. Realizamos una breve crónica de las conclusiones a las que llegamos tras la realización del proyecto.

CAPÍTULO II: DESARROLLO

2.1 Indagación sobre el cultivo de setas

2.1.1 Introducción

Antes de entrar en materia de producción del *Pleurotus ostreatus* (seta de ostra), indagamos un poco sobre el mundo de las setas.

Las setas son hongos que se desarrollan principalmente sobre troncos en descomposición u otros substratos vegetales. Cada hongo está formado por una serie de finos filamentos llamados hifas, que en conjunto forman lo que se denomina micelio. En la naturaleza y bajo condiciones favorables de humedad y temperatura, este micelio extendido sobre un substrato adecuado, se transforma en pequeños grumos que van aumentando de tamaño hasta formar la típica seta. El hongo formado con su sombrero y su pie, tiene la función de producir las estructuras de reproducción llamadas esporas cuya misión es perpetuar la especie. Estas esporas se forman en la cara inferior del sombrero, en unas laminillas verticales que se extienden desde la parte superior del pie hasta el borde del sombrero. Un hongo o cuerpo fructífero representa para el micelio lo que un fruto para un árbol.

Los hongos en general son conocidos por su forma de paraguas, con un sombrero más o menos circular y un eje o pie que lo sostiene, pero para el caso de las setas este pie es más lateral que céntrico, por lo que su desarrollo se da en forma de una ostra u oreja, de hecho a este hongo técnicamente se le llama Pleurotus, término que deriva del griego pleurá o pleurón, costado o lado y del latín otus, oreja (Fig. 1).

Las setas se alimentan de la materia orgánica en la que están creciendo, degradando las substancias con enzimas que liberan al medio húmedo que les rodea, por ello es importante el suministrar un substrato adecuado al hongo cuando se le intente cultivar para que los nutrientes puedan ser aprovechados por las hifas del micelio. Para que la seta se desarrolle adecuadamente se requiere de una temperatura y humedad adecuadas, así como aire que aporte oxígeno y cierta cantidad de luz. Con estos factores se deducen las necesidades que tiene que satisfacer el cultivo del hongo seta. El conocer el desarrollo de un hongo en la naturaleza y entender su ciclo de vida (Fig. 2), nos dará el conocimiento para poder manipularlo y producirlo en condiciones artificiales de cultivo.

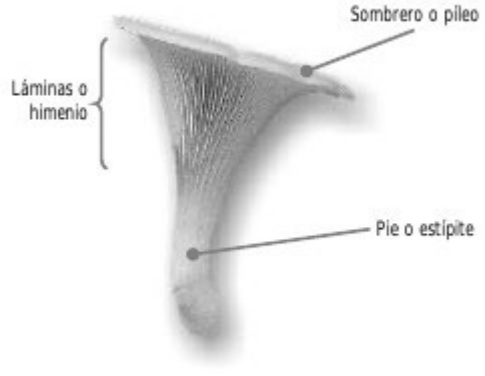


Fig.1 Hongo seta con sus partes principales

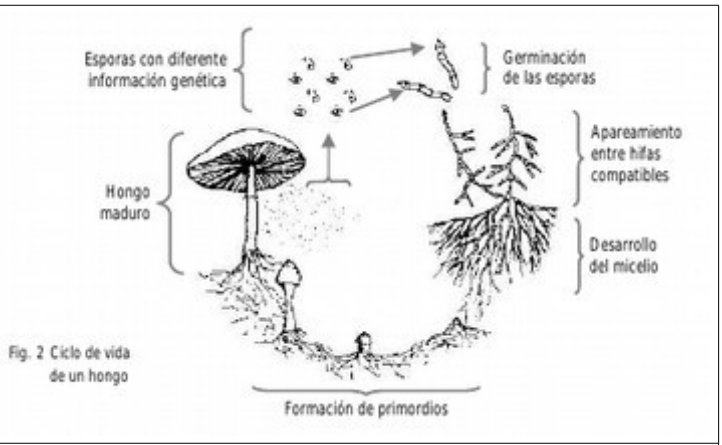


Fig.2 Ciclo de vida de un hongo

En la actualidad la biotecnología se ha convertido en una verdadera alternativa para la obtención de alimentos para el consumo humano, por la posibilidad de obtener grandes cantidades en pequeñas áreas mediante técnicas sencillas, a bajo costo, en cortos periodos de tiempo y empleando residuos agroindustriales como substrato para su cultivo, la producción de hongos comestibles, es un claro ejemplo de cómo la biotecnología es una alternativa real para la obtención de alimentos.

El valor nutricional de los hongos comestibles es notable, ya que constituyen una magnífica fuente de proteínas por contener hasta 35% en base seca, Este dato es significativo si se compara con el 13.2% del trigo y 25.2% de la leche. Además, contienen vitaminas como la B1, B2, B12, C, D, Niacina y Ácido pantoténico, así como ácidos grasos insaturados y un bajo contenido calórico.

Clasificación de los hongos por su tipo de alimentación

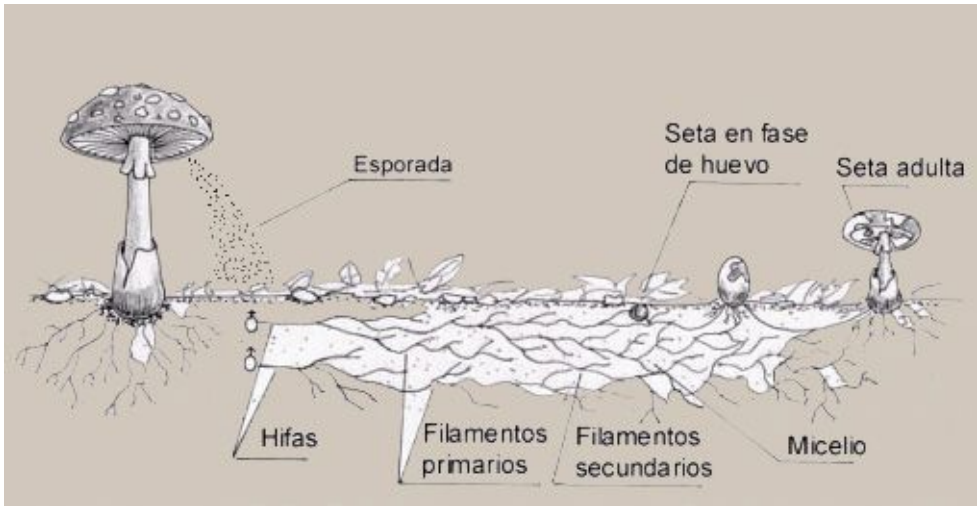
Los hongos se nutren de materia vegetal viva o muerta. Según sea el substrato en el que se encuentren se dividen en: Micorrízicos, Parásitos y Saprófitos.

Hongos Micorrízicos: Define una relación simbiótica entre las hifas de ciertos hongos y las raíces de las plantas.

Hongos Saprófitos: Viven sobre materia orgánica en descomposición, es decir sobre materia muerta, (restos orgánicos de la de plantas y animales que contiene el suelo, partes muertas de la madera de un árbol o excrementos de animales.)

Hongos Parásitos: Se alojan sobre algún ser vivo que los hospede, viviendo a expensas de éste sin ofrecerle ningún beneficio a cambio.

La seta de ostra es un tipo de hongo safrófito. El micelio de estos hongos se encuentra bajo tierra ocupando en ocasiones grandes extensiones de terreno. Se pueden reproducir tanto de forma asexual como sexual. Es esta última en la que se genera el esporocarpo (la seta en este caso). Solo cuando existen las condiciones necesarias (generalmente humedad, temperatura y luz) es cuando el carpóforo o seta “sale a la luz”.



En algunos casos hay hongos que necesitan de asociaciones simbióticas para poder desarrollarse. En el mundo de la micología, la más común de estas asociaciones es la micorriza *mycos* (hongo) *rhyzos* (raíz), es decir, el hongo necesita de las raíces de un árbol en concreto para poder desarrollarse lo que dificulta enormemente el cultivo controlado de este tipo de hongos. Uno de estos hongos es el níscalo (*Lactarius deliciosus*). Otro ejemplo de este tipo de asociación es el *Boletus edulis.* Evidentemente las variedades de setas que se cultivan como el *Pleurotus ostreatus,* champiñón, shitake, etc. no necesitan de estas asociaciones micorrizas para desarrollarse.

2.1.2 Condiciones de cultivo de *Pleurotus ostreatus*

Vamos a detallar a continuación pasos a seguir y condiciones necesarias para obtener una buena producción de *Pleurotus ostreatus.* Hay que tener en cuenta que todo el proceso debe ser realizado en condiciones lo más asépticas posible. Material esterilizado (hervido) y operando siempre cerca de un llama o fuente de calor para evitar posibles contaminaciones de otros organismos.

El cultivo de esta seta es posible realizarlo con diferentes métodos, pero en todos ellos tenemos 2 etapas principales:

- Crecimiento del micelio
- Fase de fructificación de la seta

Crecimiento del micelio

- *Obtener el micelio* en un medio nutritivo, partiendo de esporas, o de un fragmento de una seta, o incluso de un fragmento de micelio.

- *Propagar el micelio* conseguido en el punto anterior sobre granos de cereales (centeno, trigo, mijo...) o sobre serrín de madera enriquecido. El substrato hidratado se pone en tarros de vidrio y se esteriliza. Una vez esterilizado implantaremos allí una pequeña cantidad de micelio.
- *Incubar* a 20-25° C, mientras se mantiene en la oscuridad.

Fase de fructificación

- *Sembrar el micelio* (también llamado inoculo, o semilla o blanco de seta) sobre un sustrato celulósico húmedo, pasteurizado (normalmente *pajas de trigo*)

- *Cambiar las condiciones ambientales*, manteniéndolo esta vez en un sitio muy húmedo, fresco (unos 15°C), e iluminado hasta que salgan las setas.

- *Cosecharlas y comerlas.*.

1) Crecimiento del micelio

El crecimiento del micelio consta de las siguientes etapas:

- Reproducción sexual (mediante esporas) o asexual (clonando un trozo de micelio o de una seta) en una placa de petri u otro recipiente similar
- Incubar las placas de petri
- Con el micelio resultante al punto anterior inoculamos los granos de cereales para preparar el inoculo o el blanco de seta
- Incubar el micelio en grano
- El blanco de seta resultante se utiliza para colonizar el substrato "en masa"
  Incubar
- Y finalmente poner en condiciones de fructificar

Obtención del micelio

Para obtener el micelio de *Pleurotus ostreatus* necesitamos que la espora se desarrolle, pero primero tenemos que tener espora. La podemos conseguir del sombrero de una de estas setas, cortando pequeños trozos de láminas de seta. O bien por otros medios mas complejos como la técnica de la esporografía.

Propagar el micelio

Reproducción sexual(mediante esporas):

Para ello necesitamos una superficie (papel secante para absorber el exceso de agua del sombrero) sobre la que depositamos el sombrero, sin el pie y cubrimos con un recipiente o tarro para no crear corriente de aire que se lleven la espora. Es lo que se llama esporografía.

Reproducción asexual(clonar una seta):

1. Seleccionar una seta sana y con buen aspecto.
2. Limpiarla con cuidado. Utilizar alcohol o agua oxigenada no sería mala idea.
3. Coger la seta por el pie y partirla por la mitad a lo largo
4. Esterilizar el bisturí con el mechero calentándolo al rojo vivo, y deja que se enfríe unos segundos.
5. Cortar un trozo de seta del interior del sombrero de unos $2mm^3$.
6. Transferirlo a una placa con agar. Es importante clonar un mínimo de 5-6 placas para asegurarse que al menos unas cuantas resultarán libres de contaminantes.
7. Finalmente, cuando se haya acabado apilar las placas, envolverlas en una bolsa de plástico limpia, y ponerlas en un lugar conveniente para incubar a temperatura ambiente. El crecimiento micelial, si ocurre, debe hacerse visible en unos pocos días, extendiéndose hacia afuera del trozo de tejido del hongo. *Vigilar las placas porque pueden crecer mohos o bacterias en cuyo caso hay que recortar un trozo de micelio limpio y transferirlo a una placa nueva.* Si se decide subclonar el micelio para separarlo de los mohos y demás contaminantes de esta manera, hay que asegurarse de hacerlo antes de que el moho haya madurado lo suficiente como para oscurecer en color y formar esporas.

Incubar

Una vez extraída la espora debemos incubarla para que germine y se multiplique el micelio. Necesitamos un sustrato y unas determinadas condiciones para poder hacerlo.

Material necesario:

- Tarros de cristal
- Grano de cereal (trigo, cebada, centeno, arroz…)
- Algodón

Como se ha mencionado anteriormente necesitamos esterilizar el material. Hervimos los tarros de cristal durante 1 hora aproximadamente. Hacemos lo mismo con el cereal, lo cocemos durante 45-60 minutos sin que llegue a deshacerse. Si esto ocurriera estaremos destruyendo la fuente de carbono que utilizará el hongo para desarrollarse. Una vez hecho esto dejamos enfriar por lo menos hasta los 25ºC. Llenamos los botes con el grano y distribuimos la espora. Esto se puede hacer disolviendo la misma en un poco de agua (insistiendo en que tiene que ser agua hervida y enfriada) y añadiéndolo al tarro con el cereal. Tapamos el tarro con algodón. Nunca con tapa, no podemos dejar el medio de cultivo sin aporte de oxígeno. Lo incubaremos a aproximadamente 25ºC hasta que el micelio se multiplique y cubra el sustrato.

Si por la razón que sea no se puede hacer este paso por falta de tiempo, medios etc., hay empresas que venden micelio granulado listo para ser “sembrado en sustrato directamente”.

2) Fase de fructificación

Una vez obtenido el micelio necesario tendremos que sembrarlo o mezclarlo con el sustrato final. El sustrato pueden ser desechos vegetales, serrines, tallos, heno pero lo más recomendable y habitual es usar paja de cereal picada. Mezclamos el contenido de los tarros incubados con paja con una proporción del 3% en peso. La paja debe ser esterilizada, enfriada y oreada antes de hacer la mezcla. Así que la hervimos durante 1 hora.

Después de hacer la mezcla se prensa y se empaca con plástico opaco (bolsa de basura). Esto mantendrá la humedad del sustrato y evitará la exposición a la luz. Una vez hecha la mezcla necesitamos que el micelio se desarrolle e invada el sustrato antes de entrar en fase de producción. Se mantiene a 24-27ºC con un 80-90% de humedad.

En este momento tenemos listo todo para entrar en fase de fructificación. Las condiciones que vamos a necesitar para la fase de fructificación son:

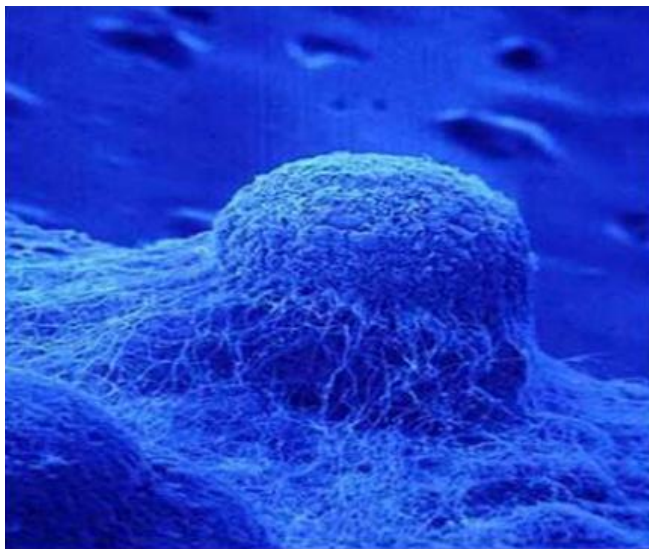
- Temperatura: 15-18 ºC. Si no se respetan los rangos puede haber consecuencias en el aspecto

del producto final.

- Humedad: Hasta un 95% para la formación de primordios. Entre 70-75% en plena producción.
- Ventilación: Se necesita ventilación moderada. Hay datos concretos pero difíciles de aplicar a niveles domésticos. 150-250 m3/h ·Tn
- Luz: 8-12h de luz natural o artificial. Si es artificial son preferibles los fluorescentes.

2.1.3 Tabla resumen de las condiciones necesarias para el cultivo de la P*leurotus ostreatus*

	Etapa del cultivo	
Factores	**Crecimiento del micelio(Grow)**	**Fructificación**
Temperatura	24º a 30°C	15º a 18ºC
Luminosidad	Oscuridad	Luz indirecta (longitudes de onda menores a 600 nm) y un fotoperiodo de 12 horas
Humedad R.	60 a 80%	85 a 90%
Aireación	28% de CO2, 20 % de oxígeno en el ambiente	20% de Oxígeno y menos de 700 ppm de CO2 en el ambiente.
pH	5-6 (bajo 4 existe inhibición)	5-6 (bajo 4 existe inhibición)

	Formación del primordio de una seta visto bajo el microscopio

2.1.4 Sustratos utilizados para el cultivo de Pleurotus ostreatus

Se emplean una gran cantidad de sustratos para el cultivo de la seta de ostra.

Los materiales más comúnmente utilizados como fuente de carbono incluyen paja de trigo, de avena, de centeno, de sorgo y de algodón, virutas de madera y cortezas, subproductos del algodón, heno, tallos de plantas de maíz, plantas y desperdicios de café, tusa de mazorca, hojas de té, cáscaras de maní, harina de soya, cáscaras de semillas de girasol, desperdicios de alcaucil, desperdicios de yuca, ágave, residuos de la industria papelera (diarios, cartones), hojas de plátano, cactus, yuca, pulpa de cardamomo, fibra de coco, hojas de limón, tallos de menta, paja de arroz, bagazo de caña, entre otros. (Stamets, 2003; Oie, 2000; Miles y Chang, 1997).
También se pueden realizar cultivos sobre bloques o troncos sintéticos. Las ventajas principales de utilizar estos sustratos en lugar de la producción en troncos naturales, es que los tiempos se acortan y la eficiencia aumenta. Las eficiencias biológicas varían de 75 a 125 % en troncos sintéticos (Miles y Chang, 1997).

2.2 Indagación sobre el uso de la lógica borrosa (Fuzzy logic)

2.2.1 Introducción a la lógica borrosa

En este documento trataremos de explicar una de las disciplinas matemáticas con mayor número de seguidores, la lógica difusa o borrosa (Fuzzy logic, en inglés).

Este tipo de lógica utiliza expresiones que no son totalmente ciertas ni totalmente falsas, es decir, es una lógica aplicada a conceptos que pueden tomar un valor indeterminado de veracidad dentro de un conjunto de valores cuyos extremos son la verdad absoluta o la falsedad absoluta. Por así decirlo es una lógica que expresa la falta de definición del objeto al que se aplica.

Si queremos dar una definición mucho más especifica podemos definir a este tipo de lógica como una técnica de la inteligencia computacional que ayuda o permite trabajar con información que es imprecisa y no está bien definida. Pertenece a la lógica multivaluada pero la lógica borrosa se diferencia de ésta en que nos permite introducir valores intermedios entre la afirmación completa o la negación absoluta.

Un ejemplo para que se entienda completamente el concepto de lógica borrosa puede ser que una persona no es simplemente alta o baja, sino que depende de si está por encima o por debajo de ciertas alturas la graduaremos de una forma u otra.

Una vez explicado este concepto, el lector puede darse cuenta de que la lógica borrosa está enraizada en la mayor parte de nuestros modos de pensar y de hablar, otra cosa es la valoración que cada persona haga a esa borrosidad existente.

Aunque ya se ha podido entender el concepto, antes de su admisión total deberán estudiarse:

- Los antecedentes históricos del concepto.
- La posibilidad de construcción de un lenguaje formal infinito valuado.
- Las consecuencias filosóficas y prácticas

2.2.2. Historia

La lógica difusa fue investigada por primera vez alrededor de mediados de los años sesenta por el ingeniero Lotfy A. Zadeh en la Universidad de Berkeley (California). En un principio este ingeniero no denominó a esta lógica como lógica borrosa sino que la llamó principio de incompatibilidad. A continuación mostraremos como describió él este principio:

”Conforme la complejidad de un sistema aumenta, nuestra capacidad para ser precisos y construir instrucciones sobre su comportamiento disminuye hasta el umbral más allá del cual, la precisión y el significado son características excluyentes”.

En este momento fue cuando introdujo el concepto de conjunto difuso (en inglés Fuzzy Set). Este nuevo concepto no es más que la idea de que los elementos sobre los que se basa el pensamiento humano no son números sino etiquetas lingüísticas. Esta idea es la que permite que se pueda representar el conocimiento, que es principalmente lingüístico de tipo cualitativo y no tanto cuantitativo, en un lenguaje matemático mediante los conjuntos difusos y funciones características asociadas a ello. Esto no quiere decir que exclusivamente se trabaje con números, este lenguaje nos permite trabajar con datos numéricos pero también con términos lingüísticos que aunque son más imprecisos que los números, muchas veces son más fáciles de entender para el razonamiento humano.

Aunque como ya hemos explicado, la lógica borrosa es hoy en día más conocida gracias a Lotfy Zadeh, la idea que se esconde detrás de este término tiene sus orígenes hace unos 2500 años atrás puesto que los filósofos griegos ya trabajaban con la idea de que existían distintos grados de veracidad y de falsedad.

Volviendo a la idea originada por Zadeh, aunque se considera que él fue quien primero habló de la

lógica borrosa, su tesis se basa también en estudios y obras de otros pensadores de otras disciplinas que tenían una visión alejada de la lógica tradicional y muy similar a la de Zadeh. Entre las obras y personas que influyeron a Zadeh, podemos destacar: la paradoja del conjunto de Russell, el principio de incertidumbre de Heisenberg y a Jack Lukasiewicz creador de la lógica multivaluada.

En un principio, la comunidad científica no vio con buenos ojos la lógica difusa, sin embargo algunos de estos investigadores que en un principio habían mostrado su resistencia ante este concepto, terminaron siendo seguidores de Zadeh e incluso mientras él seguía asentando los conocimientos de la lógica borrosa, estos científicos se dedicaron a explorar nuevas teorías referidas a este tipo de lógica. Entre estos nuevos seguidores de la lógica borrosa podemos destacar a Bellman, Lakoff, Goguen, Smith…

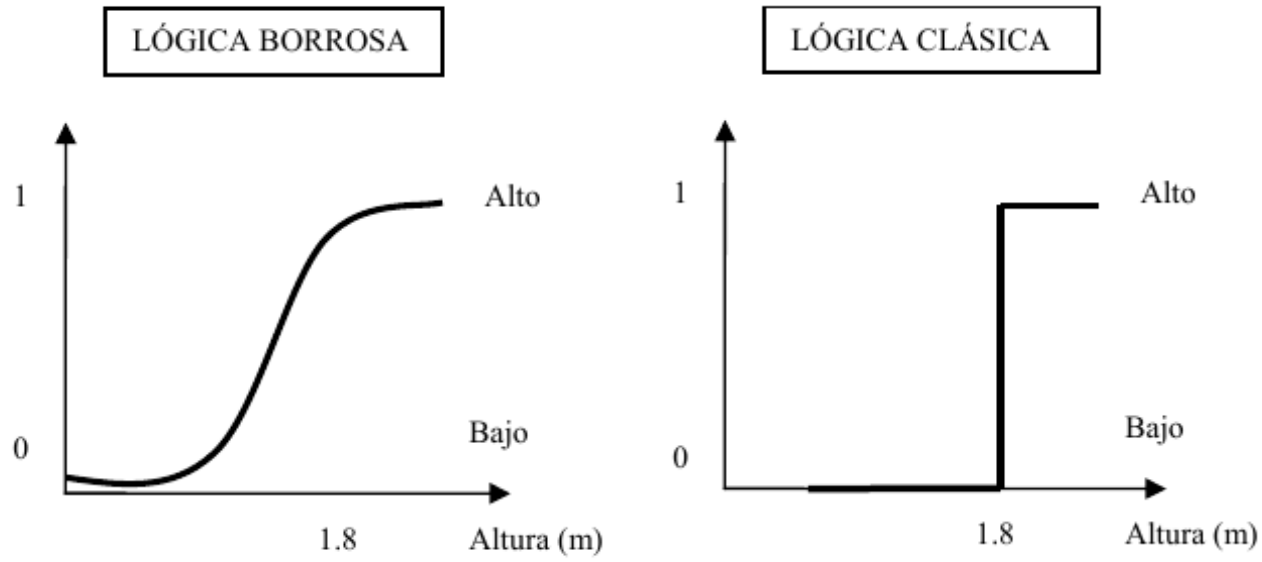
Otro paso importante para el desarrollo de la lógica difusa fue que a principios de la década de los setenta se crearon varios grupos de investigación en diferentes universidades japonesas hicieron grandes contribuciones sobre las aplicaciones que podía tener este tipo de lógica. De esta forma se consiguió crear el primer controlador difuso para una máquina de vapor o crear un controlador de inyección de química en depuradoras de agua.

En décadas posteriores esta teoría cada vez fue teniendo más éxito y se le iban encontrando nuevas aplicaciones. En la década de los ochenta, la investigación se orientó hacia las redes neuronales y su similitud con los sistemas fuzzy. Estos sistemas fuzzy lo que hacen es utilizar métodos de aprendizaje basados en redes neuronales para identificar y optimizar sus parámetros. En cuanto a la década de los noventa, a parte de la investigación de las redes neuronales y los sistemas fuzzy, surgen los algoritmos genéticos. Si combinamos estas tres técnicas computaciones, se puede conseguir una herramienta de trabajo muy potente de los sistemas de control.

Según lo expuesto hasta ahora, se puede ver que la lógica borrosa ha provocado innumerables investigaciones y aplicaciones, la mayoría orientadas a sistemas de control pero actualmente se está yendo más allá y se empieza a investigar en áreas como el reconocimiento de patrones visuales o la identificación de segmentos de ADN. Por último, mencionar que muchos de los investigadores que actualmente investigan en los temas de la lógica borrosa, comentan que el futuro de Internet (en cuanto a controlar la red, gestionarla o recuperar información), está en aplicar las tecnologías borrosas en estas áreas.

2.2.3. Conjuntos difusos

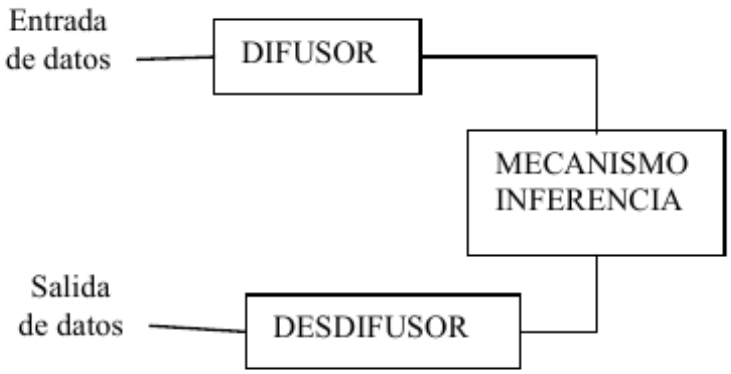
Para ilustrar el concepto de la lógica difusa y los conjuntos difusos vamos a explicar el primer ejemplo que puso Zadeh. Para ello puso el ejemplo del conjunto de “los hombres altos”. Según
la teoría de lógica clásica al conjunto de hombres altos solo pertenecen los que miden más de una determinada altura y esa altura límite es 1.80 metros, así un hombre es considerado alto cuando mide por ejemplo 1.81 metros y uno bajo cuando mide 1.79 metros. Esto no parece una razón muy lógica para catalogar a un hombre de alto o bajo ya que por ejemplo en el caso expuesto la altura de uno a otro solo se diferencia en 2 centímetros. Ahí, en casos como este donde no es fácil catalogar algo, se introduce la lógica borrosa. Según la lógica borrosa, el conjunto de “hombres altos” es un conjunto que no tiene una frontera clara que indique que perteneces a ese grupo o no. El evaluar si un hombre es alto o bajo, se hace mediante una función que define la transición entre alto a bajo y para ello asigna a las distintas alturas un valor entre 0 y 1. Según sea este valor se considera que se pertenece al conjunto o no. Aplicando esto al caso anterior, un hombre que mida 1.79 metros se puede decir que pertenece al conjunto de hombres altos con un grado de 0.75 y el hombre que medía 1.81 metros pertenece al conjunto de hombres altos con un grado de 0.8. Si representamos esto en una gráfica se obtendrá que la transición entre alto o bajo con la lógica borrosa es una curva con cambios no abruptos mientras que con la lógica clásica, el paso de alto a bajo o viceversa es brusco:



En resumen, según la lógica clásica un elemento pertenece o no pertenece al conjunto, sin embargo la lógica borrosa lo que hace es poner un grado de pertenencia al conjunto. Este grado de pertenencia se define mediante la función característica asociada al conjunto difuso: para cada valor que puede tomar la variable x, la función característica μA(x) proporciona el grado de pertenencia de ese valor x al conjunto difuso A.
Una vez aclarados mínimamente estos conceptos, a continuación explicaremos las operaciones sobre conjuntos difusos y sus propiedades.

2.2.4. Sistemas basados en lógica borrosa

Un sistema basado en lógica borrosa siempre estará compuesto por los siguientes bloques:



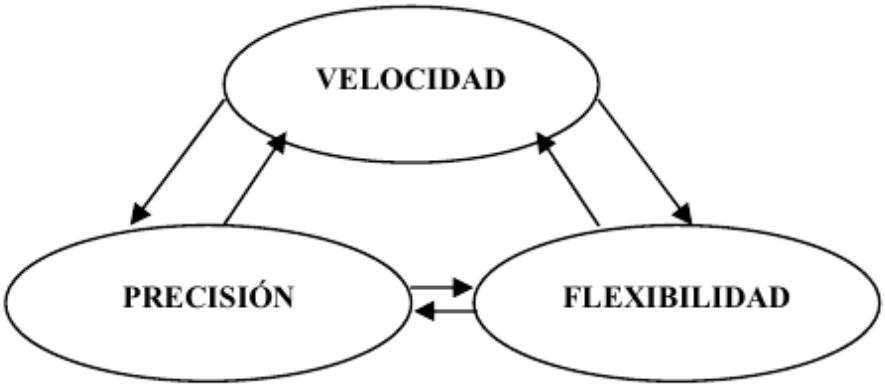
BLOQUE DIFUSOR: en este bloque a cada dato de entrada se le asigna un grado de pertenencia a cada uno de los conjuntos difusos considerados mediante la función característica ya comentada en apartados anteriores. Las entradas a este bloque son valores concretos de la variable a analizar y los datos de salida son los grados de pertenencia a los conjuntos estudiados.
BLOQUE DE INFERENCIA: este bloque relaciona conjuntos difusos de entrada y de salida y representa a las reglas que definen el sistema.
DESDIFUSOR: en este bloque a partir de los conjuntos difusos procedentes de la inferencia se obtiene un resultado concreto mediante la aplicación de métodos matemáticos de desdifusión.

2.2.5. Implementación de sistemas borrosos

En este apartado expondremos las diversas maneras disponibles para realizar en la práctica un sistema basado en lógica borrosa. Como en el caso de redes neuronales, un sistema borroso podrá implementarse como programa ejecutable por un microprocesador convencional (o microcontrolador), o podrá realizarse en hardware específico. La gran diferencia con las redes
neuronales es que un sistema borroso precisa en general de recursos de cálculo relativamente reducidos, por lo que muchas veces podrá implementarse como programa ejecutado en un sencillo microcontrolador de 8 bits.
Comenzaremos por considerar entornos de desarrollo de sistemas borrosos, mostrando el por qué son necesarios, así como algunos de los actualmente disponibles, que serán descritos con cierto detenimiento. Después el lector podrá ver los distintos métodos de realización hardware de sistemas borrosos, incluyendo los denominados aceleradores de procesamiento.

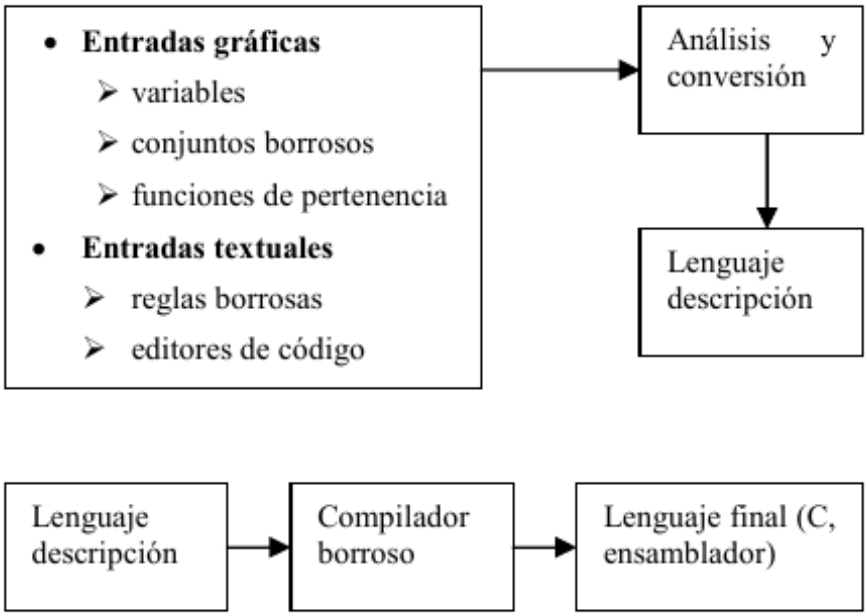
INTRODUCCIÓN

El diseño de un controlador basado en lógica borrosa supone establecer un compromiso entre diversos criterios de diseño: velocidad, precisión y flexibilidad, principalmente.



Compromiso de diseño. Falta considerar otro fundamental, que supondremos implícito: el coste

Para conseguir los resultados deseados debe plantearse la velocidad de respuesta del sistema de control, la cual vendrá limitada por otros factores, como el grado de precisión requerido o la flexibilidad del diseño. Así, si deseamos una alta precisión en el control necesitaremos una gran cantidad de conjuntos para cada variable y un alto número de reglas, lo que exigirá una elevada cantidad de cálculos, causando un aumento del tiempo de respuesta.
Si además deseamos que el sistema de control tenga flexibilidad de adaptación a los caminos del sistema y aprender de los errores cometidos, serán necesarios muchos más cálculos adicionales, que también aumentarán el tiempo de respuesta. Evidentemente, estas operaciones influyen en el coste del sistema final, y en muchos casos será ésta la mayor restricción del diseño.
Una vez decididas las prestaciones del diseño requeridas, se ha de utilizar una herramienta de desarrollo para el diseño del sistema y seleccionar la plataforma de implementación adecuada. Las herramientas de desarrollo suelen utilizar un lenguaje de descripción para independizar el diseño de la plataforma, como puede observarse en el esquema del proceso de diseño de un FLC mostrado a continuación.



En este sentido, los sistemas de desarrollo disponibles comercialmente más utilizados son FuzzyTECH, MATLAB, TILShell y FIDE.

ENTORNOS DE DESARROLLO

El continuo aumento de la complejidad de los sistemas de desarrollo ha forzado a introducir el concepto de entorno de desarrollo. Los sistemas iniciales de base puramente textual se limitaban a conjuntos de programas aislados que procesaban ficheros, produciendo otros ficheros. Estos sistemas resultaban muy complejos de manejar, y requerían de un largo periodo de aprendizaje y de alta especialización. Todo ello limitaba la productividad de los equipos de desarrollo y, en último término, la dimensión máxima de los sistemas que pueden realizarse aun coste razonable. Los primeros entornos de desarrollo se orientaban a entornos textuales, e incluían editores básicos y ayudas para la automatización de las tareas de depuración. Paulatinamente se han ido mejorando tanto las herramientas de edición, como las de depuración, y se han desarrollado sistemas automáticos de optimización y depuración tanto de los ficheros fuente como del código final generado. Este continuo aumento del número de herramientas y utilidades implicadas en el desarrollo de sistemas ha hecho cada vez más necesario el uso de entornos que integren y faciliten la tarea del programador individual, así como la integración del trabajo de los equipos de desarrollo.

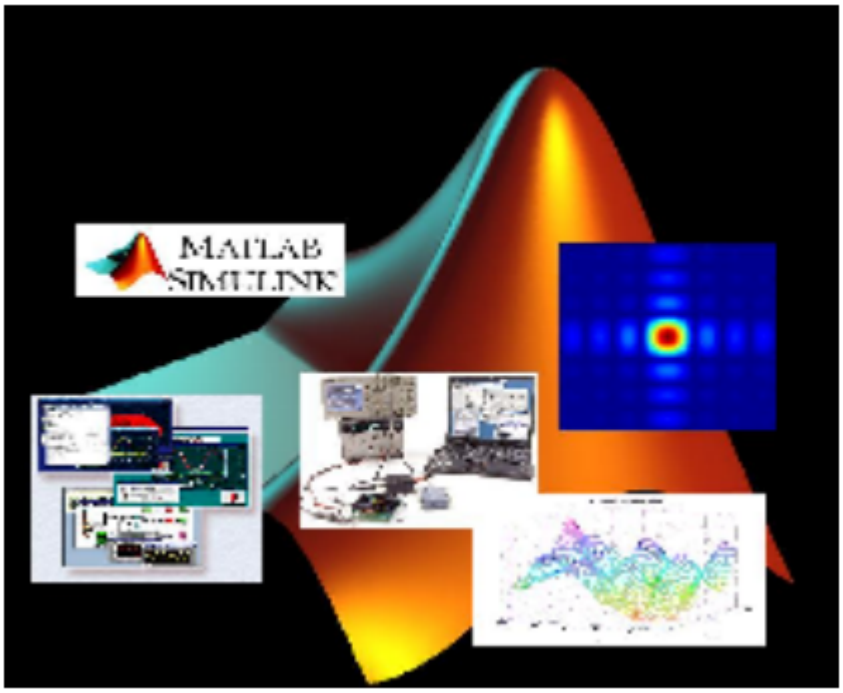
Las nuevas tecnologías de desarrollo de aplicaciones se han beneficiado de la experiencia de los sistemas clásicos y, por orientarse a aplicaciones complejas y requerir la integración con técnicas clásicas, se han diseñado mayoritariamente con el apoyo de entornos de desarrollo más o menos elaborados. Sin embargo, por ser tecnologías recientes y de origen diverso, apenas se dispone aún de sistemas que faciliten la integración en un mismo sistema de las diversas metodologías de trabajo disponibles (redes neuronales, sistemas borrosos, algoritmos genéticos, etc.) En este sentido, quizás sea MATLAB el entorno más completo actualmente, pues permite el trabajo desde un mismo entorno con técnicas clásicas y novedosas (wavelets, redes neuronales, etc.). Su principal limitación es el limitado soporte para el desarrollo de sistemas en tiempo real.

Entornos de tipo matemático

Como herramientas para la investigación en el desarrollo de sistemas de control y procesamiento de señal, se han utilizado frecuentemente entornos de base matemática que facilitan la evaluación de diversas técnicas, sin obligar inicialmente al desarrollo de programas específicos, haciendo uso de paquetes matemáticos diversos incluidos en estos sistemas matemáticos de propósito general. Así,

dado un sistema de desarrollo matemático de propósito general, se suelen suministrar por separado paquetes o conjuntos de utilidades específicos para tratamiento de señal, control de sistemas, trabajo con redes neuronales, lógica borrosa, etc.
La mayoría de estos sistemas matemáticos tienen su origen en entornos textuales, aunque en sus versiones actuales se han adaptado a los entornos gráficos disponibles hoy en día, tanto en su interacción con el usuario, como en la presentación de los resultados de las aplicaciones desarrolladas.

MATLAB



MATLAB es probablemente el entorno de desarrollo matemático más extendido para las aplicaciones de control y procesamiento de señal, especialmente en ambientes universitarios, donde se utiliza para la simulación de control de sistemas.
Este entorno puede considerarse una especie de sistema interactivo cuyos elementos básicos son matrices, y su dimensionamiento se realiza dinámicamente. La solución de los problemas se expresa en MATLAB con expresiones matemáticas similares a las habituales, de modo que en principio no requiere programación.
MATLAB se utiliza habitualmente en entornos de ingeniería para analizar y desarrollar prototipos de algoritmos o computaciones numéricas, utilizando una formulación matricial que se adapta bien tanto a la teoría de control automático clásico, como al procesamiento digital de señales.
Se pueden definir nuevas funciones como secuencias de comandos ya definidos, almacenándolos en un fichero de tipo M-file. Existen amplias colecciones de estos M-files o toolboxes que han sido escritas para aplicaciones especiales, y que se venden por separado. En los últimos años se han incorporado el Neural Networks Toolbox y el Fuzzy Logic Toolbox, que permiten el trabajo con redes neuronales y lógica borrosa, respectivamente, con relativa sencillez y buenas salidas gráficas.

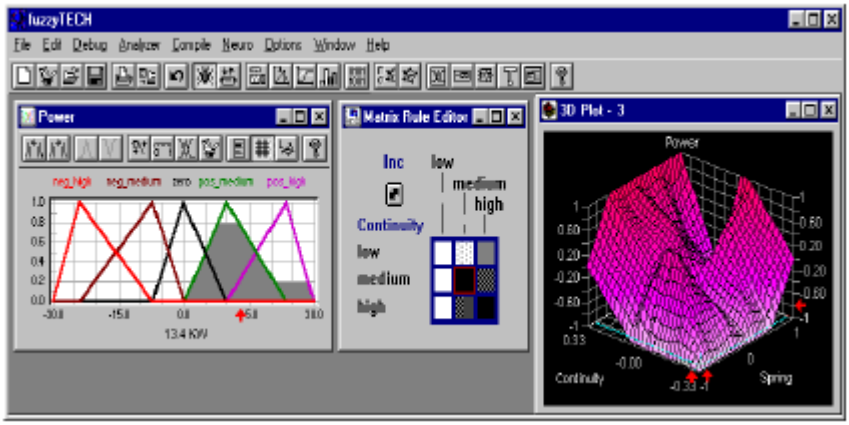
ENTORNO DE LÓGICA BORROSA

Anteriormente hemos mencionado algunas herramientas de tipo matemático, susceptibles de ser empleadas en el desarrollo de sistemas borrosos y/o neuronales. A continuación, se describirá uno de los entornos más difundidos orientados específicamente a lógica borrosa.

FUZZYTECH (INFORM GmbH)

El entorno de FuzzyTech fue desarrollado por la compañía INFORM Software GMBH, el cual surgió del

trabajo de un grupo de investigadores dirigido por el profesor Hans Zimmermann, de la Universidad de Aachen (Alemania). Zimmermann, uno de los pioneros de la lógica borrosa en Europa, es presidente y fundador de la International Fuzzy Systems Association (IFSA), la principal organización internacional para la investigación y aplicación de los sistemas basados en lógica borrosa.
El entorno fuzzyTech está formado por un GUI común basado en MS-Windows que permite la edición gráfica de las variables lingüísticas para cada una de las variables del sistema, con una precisión seleccionable según el tipo de implementación final seleccionado (8 a 16 bits).
FuzzyTech dispone de varios asistentes para facilitar las tareas más frecuentes del diseño, como asistente para la estructura, para las variables y para las reglas. Incluye además una ventana de gestión del proyecto con estructura de árbol, un sistema personalizable de generación automática de documentación del proyecto según el estándar IEC1131-7, y un gestor de versiones integrado.
El entorno admite la simulación fuera de línea y en tiempo real. Dentro del primer tipo, permite la depuración interactiva con visualización de flujos de inferencia, y optimización interactiva de los parámetros del sistema.
Como cualidad importante de este entorno destacaremos que dispone de un gran número de versiones con una interfaz de usuario común, adaptadas a un gran número de implementaciones.
Además, este entorno ha sido seleccionado por diversos fabricantes de sistemas industriales como herramienta para el desarrollo de aplicaciones de control basadas en la lógica borrosa sobre sus sistemas.



2.2.6. Aplicaciones de los sistemas borrosos

A continuación mostraremos algunas de las aplicaciones de los sistemas borrosos, tanto académicas como reales, relacionadas con procesamiento de datos y control. Como se puede deducir de lo expuesto anteriormente sobre sistemas borrosos, el campo que claramente cuenta con mayor número de aplicaciones basadas en lógica borrosa es el control, existiendo numerosas aplicaciones en el funcionamiento de la industria.

INTRODUCCIÓN

Los nuevos modelos de procesamiento y control: redes neuronales, lógica borrosa y algoritmos genéticos, junto con algunos otros de relativa novedad, se engloban dentro de la denominada “soft computing” [Zadeh, Jang] o inteligencia computacional, que tienen en común el constituir paradigmas de procesamiento muy diferentes a la convencional “hard computing”, basadas en computadores Von Neumann (serie) y la separación de hardware y software. Estas nuevas técnicas se inspiran en las soluciones que la naturaleza ha encontrado durante millones de años de evolución a numerosos problemas tecnológicos que involucran el tratamiento de cantidades masivas de información, redundante, imprecisa y ruidosa, problemas a los que en la actualidad se enfrenta el ser humano (visión, habla, control en ambiente natural). No obstante, debe quedar claro que estas nuevas técnicas no vienen a suplantar a las más tradicionales, sino más bien a completarlas en aquellos problemas donde menos eficacia proporcionan.

Es fácil deducir que tanto las redes neuronales artificiales como los sistemas borrosos, constituyen en la actualidad áreas de investigación y desarrollo (I+D) muy activas, no sólo desde un punto de vista académico, sino comercial e industrial. Recordemos que grandes compañías del sector electrónico e informático, como Siemens, Motorola, SGS-Thomson, Toshiba, National Semiconductors, OMROM, etc., trabajan en redes neuronales y sistemas borrosos.
En la línea anterior, cabe destacar el mundo empresarial japonés en la aplicación de lógica borrosa en la industria y en aparatos de consumo. Recordemos que en EEUU y Europa solamente se empezó a dar importancia a la lógica borrosa cuando desde Japón empezó a llegar abundante información sobre numerosas aplicaciones prácticas de esta técnica, desarrolladas y comercializadas por compañías japonesas.

INTERÉS DEL EMPLEO DE LA LÓGICA BORROSA

Los sistemas basados en lógica borrosa pueden ser aplicados prácticamente a los mismos problemas que las redes neuronales. De esta manera, resultarán especialmente interesantes para los problemas no lineales o no bien definidos.
El otro gran denominador común de ambas técnicas es su orientación hacia el tratamiento de tareas que involucran al procesamiento de cantidades masivas de información, de tipo redundante, imprecisa y con ruido, que aparecen en problemas tecnológicos cruciales a los que en la actualidad se enfrenta el ser humano.
No obstante, frente a las características comunes citadas, existen también importantes diferencias. Por ejemplo, los sistemas basados en lógica borrosa permiten utilizar el conocimiento que los expertos disponen sobre un tema. Por consiguiente, puede decirse que la lógica borrosa permite formalizar tanto el conocimiento ambiguo de un experto como el sentido común.
Una importante ventaja de los sistemas borrosos, es que gracias a la simplicidad de los cálculos requeridos (sumas y comparaciones fundamentalmente), normalmente pueden realizarse en sistemas baratos y rápidos, con lo que pueden implementarse en sistemas específicos (por ejemplo, para el control inteligente de un horno microondas o de un sistema de frenado ABS). Este es uno de los motivos fundamentales del hecho constatado de la existencia en la actualidad de muchas más aplicaciones prácticas funcionando basadas en lógica borrosa que en redes neuronales.
No obstante, las ventajas e inconvenientes que cada enfoque puede presentar, el futuro apunta en la dirección de combinar distintas técnicas para resolver problemas complejos. Los problemas tecnológicos de mundo real resultan en general de gran complejidad, por lo que para su resolución conviene que sean divididos en partes más simples, de manera que cada una pueda ser resuelta mediante la técnica más indicada, procedente del campo de la estadística, procesamiento de señal, reconocimiento de patrones, redes neuronales, sistemas borrosos, algoritmos genéticos o cualquier otra.
Debemos recordar en esta línea de razonamiento que no existen soluciones simples a problemas complicados. Estas nuevas técnicas emergentes aportan características sumamente interesantes, pero por si solas no resolverán todos nuestros problemas tecnológicos, sino que contribuirán en determinados aspectos, pero otros seguirán siendo mejor abordados mediante técnicas tradicionales. En este sentido, no conviene forzar la aplicación de cierta nueva técnica a determinado problema simplemente por su novedad, sino que ello debe realizarse solamente en aras de conseguir un mayor rendimiento o sencillez de implementación.
Este último es otro de los aspectos destacables de los sistemas borrosos, su relativa sencillez de aplicación. A veces mediante un sistema borroso no se logra un rendimiento superior (ni inferior) con el que se alcanzaría con un enfoque clásico, pero el tiempo de desarrollo es con frecuencia inferior, y el sistema final resultará más barato.
Para concluir, y retomando la línea de la fusión de tecnologías, merece a pena recordar el intenso trabajo que se desarrolla en sistemas neuro-borrosos. Los sistemas borrosos pueden aprovechar la capacidad de aprendizaje de una red neuronal para optimizar su funcionamiento. Por otro lado, la equivalencia que se establece entre ciertos modelos neuronales y borrosos puede ser empleada para extraer las reglas que una red neuronal ha encontrado en el entrenamiento, eliminando uno de los grandes problemas clásicamente achacado a los sistemas neuronales artificiales, su operación en forma de caja negra. Por todo ello, la combinación de redes neuronales y sistemas borrosos es un campo intenso de trabajo en la actualidad.

ALGUNAS APLICACIONES DE LOS SISTEMAS BORROSOS

En esta apartado se intentará hacer una enumeración de algunos ejemplos de aplicaciones prácticas de los sistemas borrosos. La relación no pretende ser exhaustiva, sino simplemente ilustrativa de los diferentes tipos de problemas que en la actualidad sed abordan mediante el uso de las técnicas procedentes de la lógica borrosa.
Haciendo un poco de historia, la primera aplicación práctica operativa de la lógica borrosa la desarrolló E. Mandami en Europa, realizando el control borroso de un sistema de vapor de una planta industrial. Otras de las más clásicas quizás sea la de Smith y otros, que en 1980 aplican técnicas de lógica borrosa al control de hornos rotativos en una cementera. El control en planta industrial sigue siendo hoy en día uno de los campos de aplicación más destacables.
Los sistemas basados en lógica borrosa se vienen utilizando en aplicaciones de diversas índoles. Así, en el área médica se emplea para diagnósticos, acupuntura, análisis de ritmos cardíacos, o de la arterioestenosis coronaria. Dentro del apoyo a la toma de decisiones, otras de las grandes áreas de aplicación de estos sistemas, se han utilizado, por ejemplo, en la búsqueda de caminos críticos en la ejecución de proyectos, y asesoramiento a la inversión.
Sin duda, el principal campo de aplicación de la lógica borrosa es el de control, a partir del empleo de las expresiones de la lógica borrosa para formular reglas orientadas al control de sistemas. Dichos sistemas de control borroso pueden considerarse una extensión de los sistemas expertos, pero superando los problemas que éstos presentan para el razonamiento en tiempo real, ocasionados por la explosión exponencial de las necesidades de cálculo requerido para un análisis lógico completo de las amplias bases de reglas que éstos manejan.
En el campo de control de sistemas en tiempo real destaca el control de un helicóptero por órdenes pronunciadas de viva voz (Sugeno), y el control con derrapaje controlado de un modelo de coche de carreteras de Altrok. Dentro del sector del automóvil existen gran número de patentes sobre sistemas de frenado y cambios de marcha automáticos.
El área de los aparatos de consumo es otra de las mas destacables, hasta el punto de que no es raro encontrarse propaganda del tipo “Incluye inteligencia artificial fuzzy”, en escaparates o en catálogos (por ejemplo, en los de las cámaras de video Hitachi). En el sector de los electrodomésticos se han diseñado un buen número de aplicaciones neuro-borrosas como lavadoras (Matsushita, Hitachi, Siemens, AEG), tostadoras de pan, controles de calefacción y aire acondicionado.
Para el control de maquinaria destaca el ya clásico control de frenado del metro de Sendai (Japón), realizado por Hitachi, y que opera desde julio de 1987. Se han aplicado sistemas borrosos en el control de maquinaria de perforación de túneles y en el control de ascensores (Mitsubishi-Elec., Hitachi, Fuji Tech) y guías para contenedores. Se han aplicado también al procesado de imágenes y reconocimiento de caracteres; por ejemplo, se muestra un sistema que reconoce los números de los cheques bancarios, para lo cual hace uso de un sensor de CCD, 64X1 píxeles, y un microcontrolador M68HC11, en el que el sistema borroso se materializa por programa.

A continuación como resumen del apartado, citamos algunos ejemplos prácticos de aplicación de sistemas fuzzy en la vida real:

- Sistemas de control de acondicionadores de aire
- Sistemas de foco automático en cámaras fotográficas
- Electrodomésticos familiares (frigoríficos, lavadoras...)
- Optimización de sistemas de control industriales
- Sistemas de escritura
- Mejora en la eficiencia del uso de combustible en motores
- Sistemas expertos del conocimiento (simular el comportamiento de un experto humano)
- Tecnología informática
- Bases de datos difusas: Almacenar y consultar información imprecisa. Para este punto, por ejemplo, existe el lenguaje FSQL.

2.2.7. Conclusión

La lógica borrosa ha experimentado un rápido crecimiento debido a su capacidad de resolver problemas relacionados con la incertidumbre de la información o del conocimiento de los expertos. Además, proporciona un método formal para la expresión del conocimiento en forma entendible por el ser humano. Estas cualidades le aseguran un amplio campo de aplicabilidad y un alto interés para las aplicaciones industriales, presentes y futuras.

Tanto las redes neuronales como los sistemas borrosos se aplicarán especialmente allá donde los comportamientos no lineales sean importantes. Cuando no se posea un modelo suficientemente bueno, pero si se disponga de un amplio conjunto de ejemplos (casos experimentales), el empleo de una red neuronal puede resultar útil, y podemos dejar que mediante un proceso de entrenamiento ella misma encuentre el modelo o características más relevantes. Sin embargo, cuando se disponga de un conjunto de reglas proporcionadas por los expertos en un determinado tema, el empleo de sistemas basados en lógica borrosa puede ser tremendamente útil.

Además, gracias a la simplicidad de los cálculos necesarios, normalmente pueden realizarse en sistemas baratos y rápidos de fabricar.

No obstante, la combinación de todas las técnicas conocidas incluidas las más clásicas (estadísticas, tratamiento de señal, etc.), sería la mejor opción para diseñar un control eficiente.

2.3 Modelado teórico previo del invernadero necesario

2.3.1. Cálculo de la energía necesaria para bajar la temperatura de 30ºC a 10ºC mediante células péltier.

Se supone que el invernadero no tiene pérdidas (caso ideal)

Eficiencia de una célula peltier: 0'25
Potencia de la célula peltier: 60W
Potencia de trabajo para la célula peltier: 60W al 70% → 60·0'7=42W
Potencia aprovechable = 42W · 0'25 = 10'5W

Energía $E=P \cdot t$ (en julios)
Calor $Q=c \cdot m \cdot \Delta T$

Tamaño del invernadero: 40cm x 40cm x 40cm
Volumen en litros del invernadero: 40cm x 40cm x 40cm = 64000 cm^3 = 64 litros

Densidad del aire: 1'185 g/l
Masa de aire en el invernadero: $d=M/V \rightarrow M=d \cdot V = 1'185 \cdot 64 = 75'84g$
Masa a enfriar (solo aire): 75'84g

Masa a enfriar (aire + cereal): 1000g
Calor específico del aire: 1'012 $J \cdot g^{-1} \cdot K^{-1}$
Calor específico del cereal con agua y vidrio (*consideramos como si todo fuera agua, peor caso*): 4'1813 $J \cdot g^{-1} \cdot K^{-1}$

Calor necesario para bajar la temperatura $\Delta T=20ºC$ (solo aire):

$Q=c \cdot m \cdot \Delta T = 1'012 \cdot 75'84 \cdot 20 = 1535J$
$E = P \cdot t_1 = 1535 = 10'5 \cdot t_1$
$t_1 = 1535/10'5 = 146'2s$

Calor necesario para bajar la temperatura $\Delta T=20ºC$ (aire + cereal):

$Q=c \cdot m \cdot \Delta T = 4'1813 \cdot 1000 \cdot 20 = 83626J$
$E = P \cdot t_2 = 83626 = 42 \cdot t_2$
$t_2 = 83626/10'5 = 7964'4s$

Tiempo total para conseguir enfriar la muestra:

$t_{total} = t_1 + t_2 = 146'2 + 7964'4 = 8110'6s = 135'2$ minutos = 2 horas y 15 minutos

2.3.2. Estimación de la condensación dentro de la incubadora de setas y cálculo del tiempo necesario para ajustar la humedad.

Especificaciones generales del humidificador:	
Material	ABS + PP
Método	Transductor Piezoeléctrico 107kHz
Potencia	2W
Humidification	10-35ml / h
Dimensión	13.5 x 7 cm

Dimensiones del invernadero y volumen:

H x W x L -> 50 cm x 40 cm x 40 cm = 80000 cm^3 = 80 dm^3 = 80 litros = 0’08 m^3

Condiciones iniciales:

- 60% de HR
- temperatura interior 26ºC

1º Calcularemos cuanta agua hay en el aire con estas condiciones:

```
>> humedad_absoluta(26,60)
ans =  14.640
```

Respuesta: 14’64 gramos/m^3

2º Calcularemos cuanta agua hay en el volumen de aire del invernadero con estas condiciones (26ºC y 60%HR):

```
>> 14.64*0.08
ans =  1.1712
```

Respuesta: 1’17 gramos = 1’17 ml

3º Calcularemos cuanta agua cabe en el volumen de aire del invernadero al llegar a las condiciones de frio y humedad requeridas (10ºC y 80%HR):

```
>> humedad_absoluta(10,80)
ans =  7.5200
>> 7.52*0.08
ans =  0.6016
```

Respuesta: 7’52 gramos/m^3
Dentro del invernadero = 0’6 gramos = 0’6 ml
Como 1’17 ml es mayor que 0’6 ml habrá condensación al enfriar el aire.
En este caso acabarían condensándose (1’17 ml - 0’6 ml) = 0’57 ml de agua

4º Calcularemos ahora el peor caso (cuando el aire de dentro del invernadero está caliente y saturado de humedad, 30ºC y 90%HR y lo enfriamos como antes a 10º y un 80%HR):

```
>> humedad_absoluta(30,90)
ans =  27.360
>> 27.360*0.08
ans =  2.1888
```

Respuesta: 27’36 gramos/m^3
Dentro del invernadero = 2’19 gramos = 2’19 ml
En este caso acabarían condensándose (2’19 ml - 0’6 ml) = 1’59 ml de agua

Como vemos, es muy poca agua la que nos va a condensar dentro del invernadero, teniendo además en cuenta que tardará unas 3 horas en alcanzar la temperatura mínima.
Teniendo en cuenta que en una cucharada sopera caben 15 ml, estaríamos hablando de una condensación de la décima parte de esta cuchara que se condensaría por las paredes del interior del invernadero.

NOTA: Nuestro humidificador genera entre 10ml/hr y 35ml/hr con lo que tenemos que ir con cuidado de no tener demasiado tiempo encendido el humidificador para no saturar el aire enseguida.
Por ejemplo, para saber cuanta agua cabe en el aire para pasar de 10ºC y un 60%HR a 10ºC y un 80%:

```
>> humedad_absoluta(10,80) - humedad_absoluta(10,60)
ans =  1.8800
```

Respuesta: 1'88 gr/m^3

Que para el volumen del invernadero serían: $0'08m^3 \cdot 1'88\ gr/m^3 = 0'15gr = 0'15\ ml$
Es decir que en (0'15/10) horas tendremos la humedad ajustada: 0'015 horas = 54 segundos.

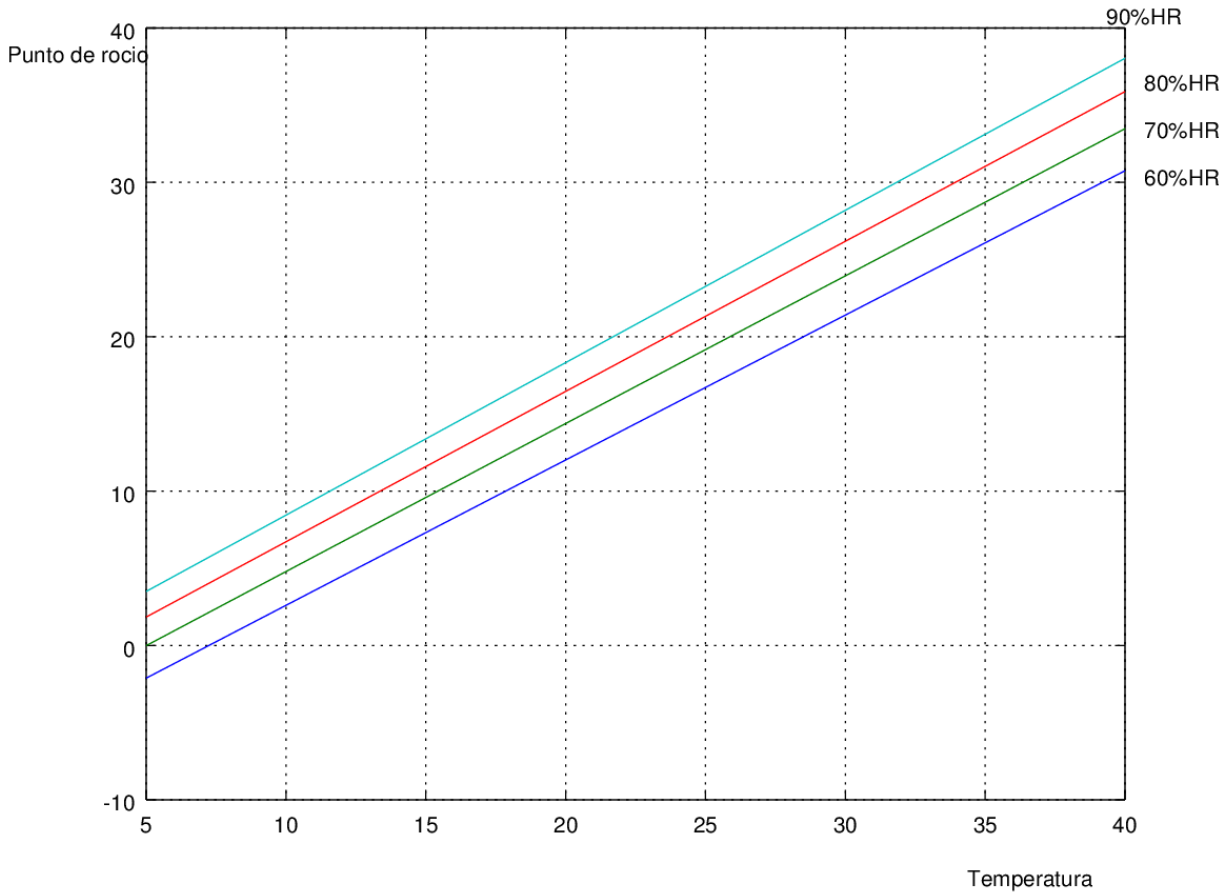
Funciones implementadas en MATLAB:

Ver codigo empleado en Anexos

2.3.3. Gráficas en Matlab de las variables de estudio

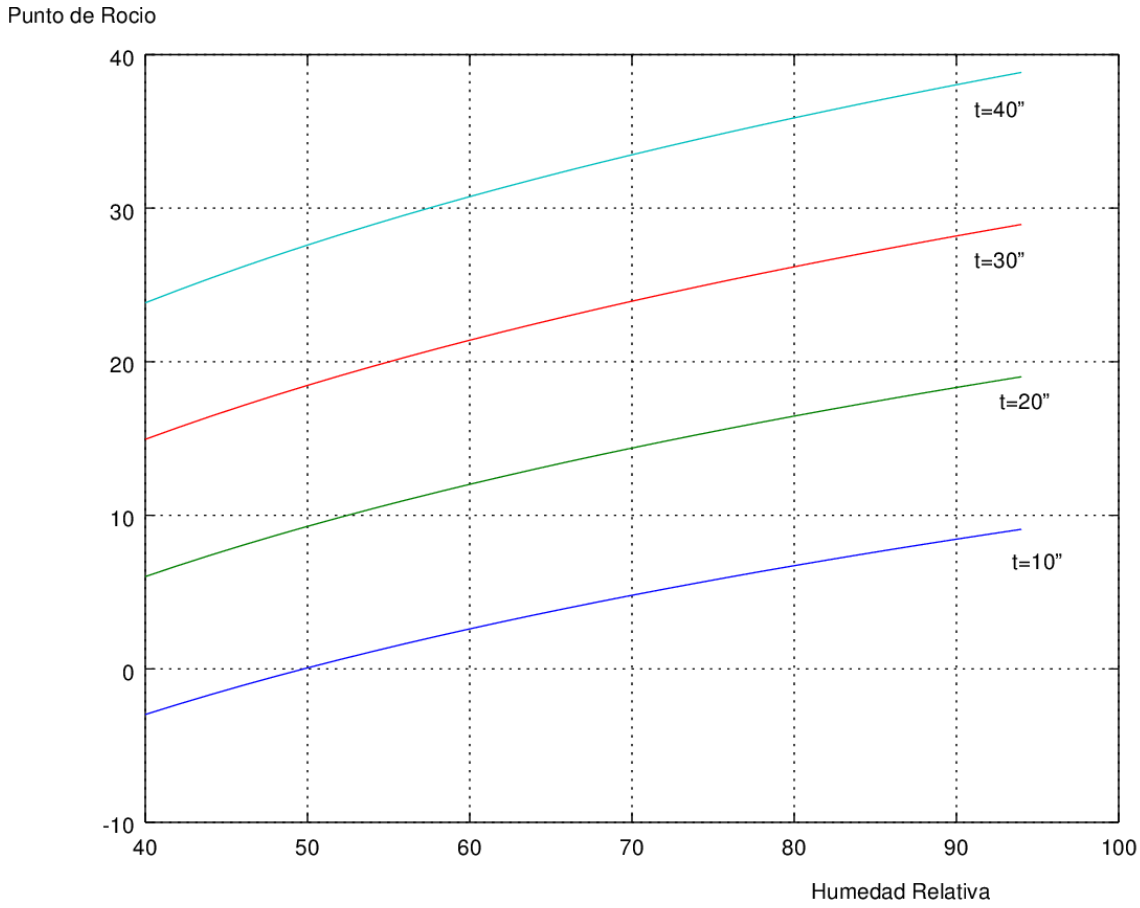
Punto de rocio en función de la temperatura:

```
>> t=[5:1:40];
>> plot(t,dew_point(t,60),t,dew_point(t,70),t,dew_point(t,80),t,dew_point(t,90))
```



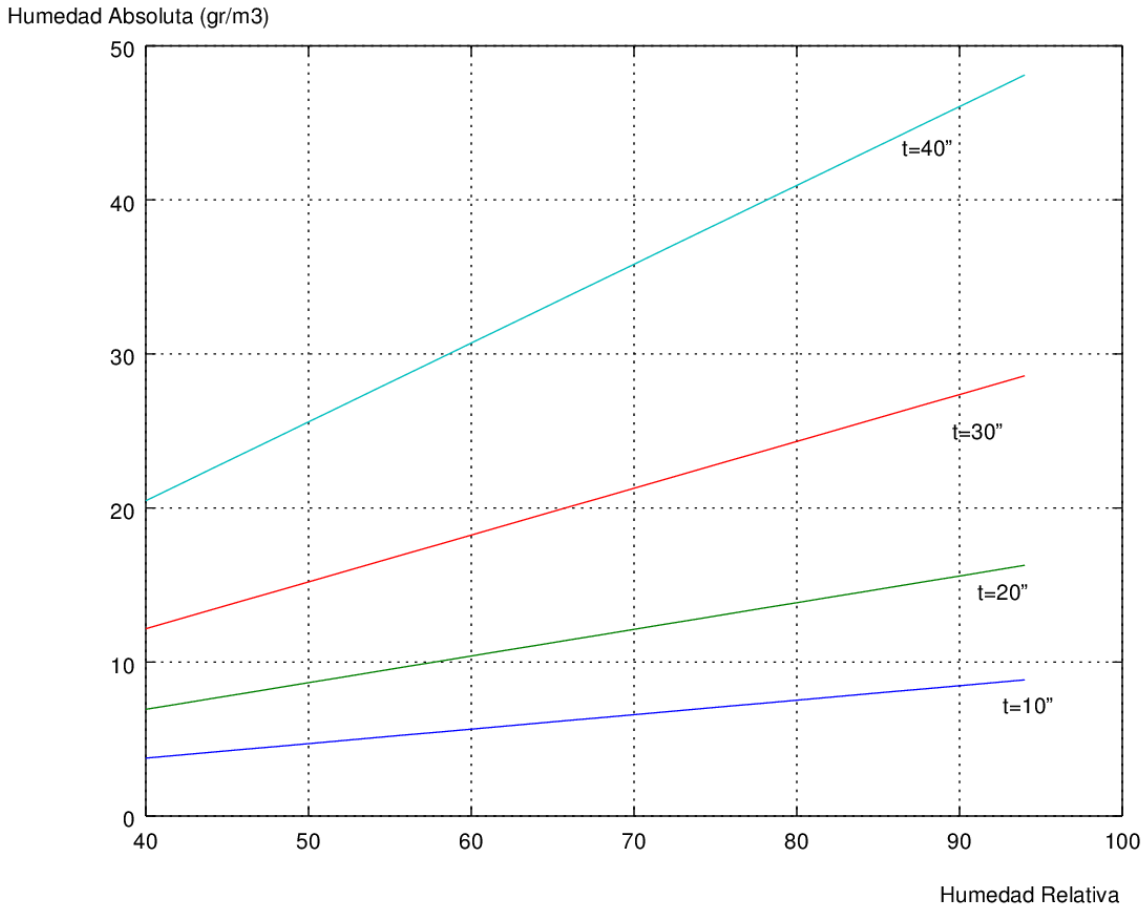
Punto de rocío en función de la Humedad:

```
>> humedad=[40:2:95];
>>plot(humedad,dew_point(10,humedad),humedad,dew_point(20,humedad),humedad,dew_p
oint(30,humedad),humedad,dew_point(40,humedad))
```



Humedad absoluta en función de la Humedad:

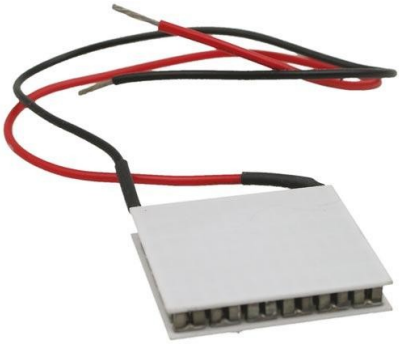
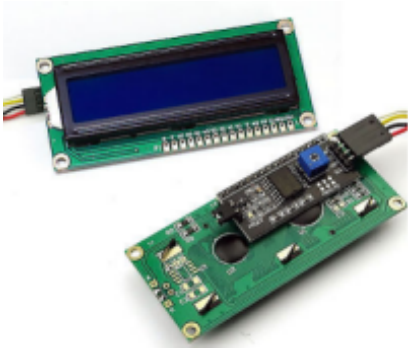
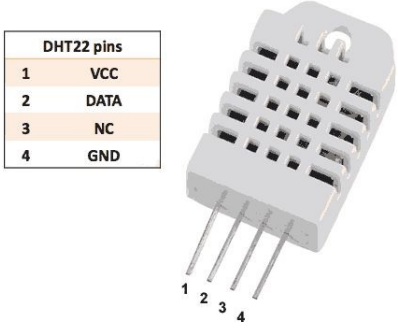
```
>> humedad=[40:2:95];
>> plot(humedad,humedad_absoluta(10,humedad),humedad, humedad_absoluta
(20,humedad),humedad, humedad_absoluta (30,humedad),humedad, humedad_absoluta
(40,humedad))
```



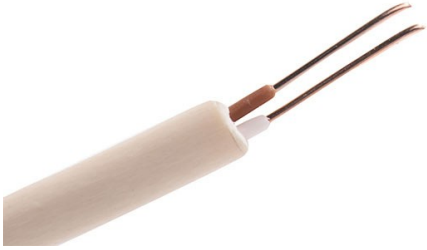
2.4 Montaje físico del invernadero

2.4.1. Componentes

A continuación damos una lista de los componentes básicos empleados para realizar el control del invernadero

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>Arduino Uno</td><td>Bomba inyectora de aire 12V</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Célula Peltier 50W</td><td>Display LCD 16x2</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Humidificador piezoeléctrico 107kHz</td><td>Vista la parte empleada del humidificador</td></tr>
<tr><td>  </td><td> 
</td></tr>
<tr><td>LED 6W 12V AC/DC</td><td>Sensor DHT 22 (AM2302)</td></tr>
</table>

	
Módulo descender DC/DC Buck 30V-3V	Placa 4 Relés Optoacoplados Arduino
	
Trafo de audio de 8:1300Ω	NPN Darlington (ESM6045AV)
	
MOSFET IRF510	Fuente de alimentacion 12V
	
Ventilador con disipador interior	Ventilador con disipador exterior

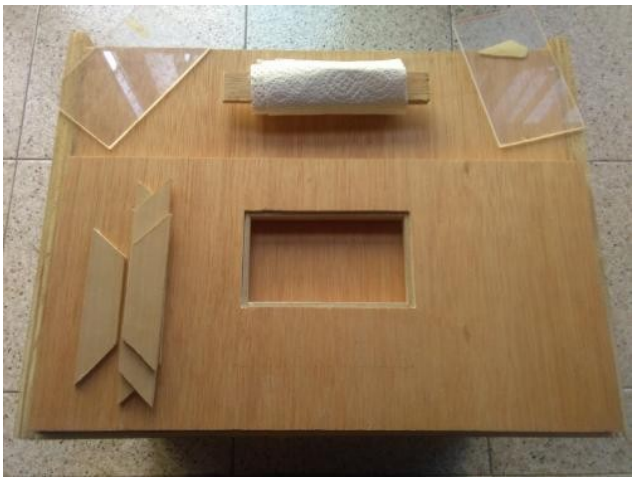
	
Conector empalme tubo de bomba	Fichas de empalme 12x2
	
Rollo cable de pares de 0’25x12, 3m	Rollo de cable para audio estéreo apantallado, 2m
	
Rollo cable telefónico, 2m	Caja de conectores DuPont para Arduino
	
Caja de puntas de electricidad para crimpar	Tubo de plástico para la bomba de aire

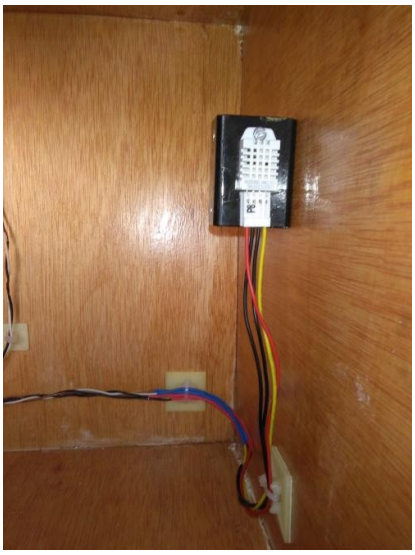
2.4.2. Montaje habitáculo del invernadero

Realizamos todo el montaje físico del habitáculo de manera artesanal.
Os dejamos un documento gráfico de todo el proceso de cortado, pegado, sellado, fabricación e instalación de los componentes de la caja de cultivo.

	
Fabricación del agujero de la ventana	Lijado de la cavidad
	
Encolado de los tableros de contrachapado	Vista parte delantera
	
Vista parte posterior	Resultado final de la caja montada

	
Y pasamos a rematar la puerta	Resultado final
	
Pasamos a los componentes: • Corte de la chapa de hierro galvanizado de 0’8mm que ira adherida a la peltier para mejorar la disipació de calor de los focos de temperatura. • Aislamiento del led	Aislamos el led para protegerlo de la humedad con un tapón de plástico y un pequeño tapón de madera fabricado y moldeado artesanalmente.
	
Corte de la chapa visto de cerca	Agujero realizado en la parte posterior de la caja a medida de la peltier, donde irá encajonada la misma

	
Agujero visto por detrás	Corte perfecto ajustado a medida
	
Pasamos a preparar el encajonado donde irá alojado el metacrilato de la ventana.	Vista plano inclinado
	
Confeccionado tipo climalit con doble capa y cámara de aire entre medias para mejorar el aislamiento térmico	Marco exterior para embellecer la ventana

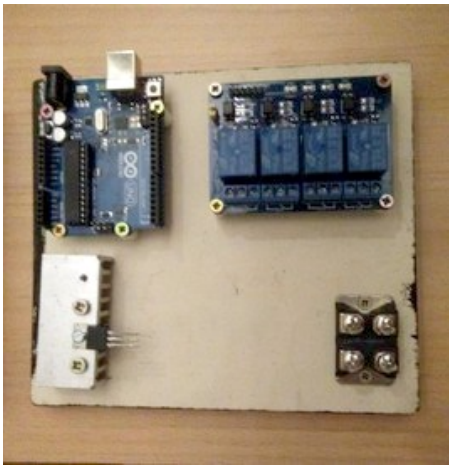
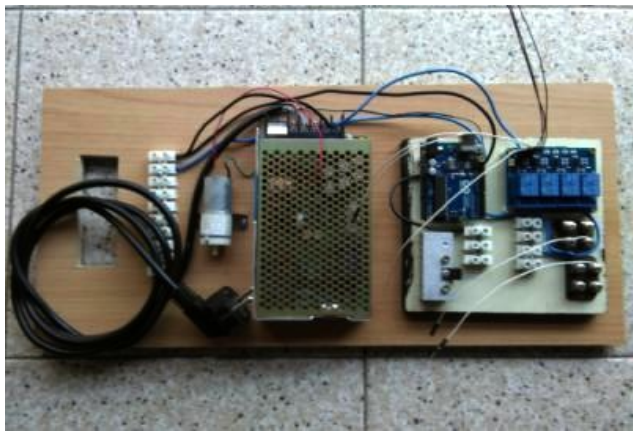
	
Resultado final de frente	Resultado final vista de canto
	
Instalación bornero de empalme	Instalación sensor DHT22 sobre una chapa metálica
	
Instalación ventilador interior caja	Instalación del led

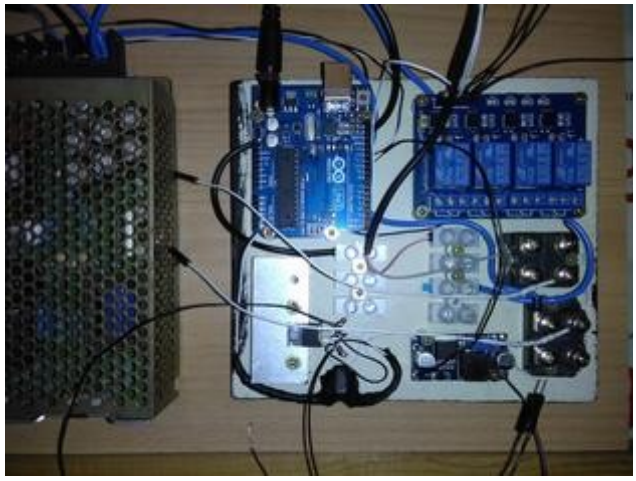
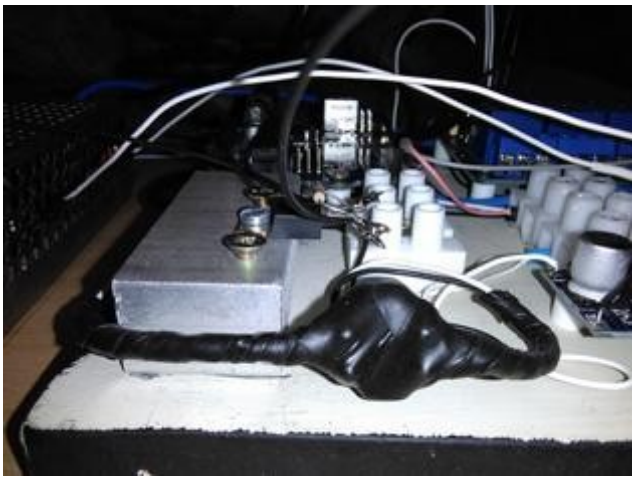
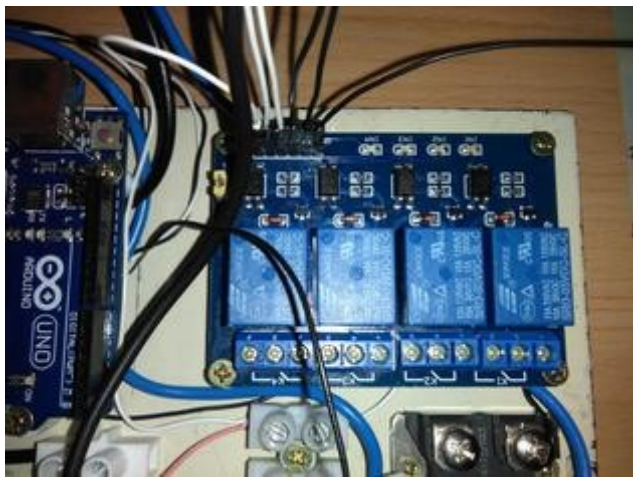
	
Vista de cerca del led	Ventilador y led
	
Vista completa con todo instalado	Instalamos la bisagra de la puerta
	
Cableamos e instalamos el ventilador exterior	Vista de frente del ventilador

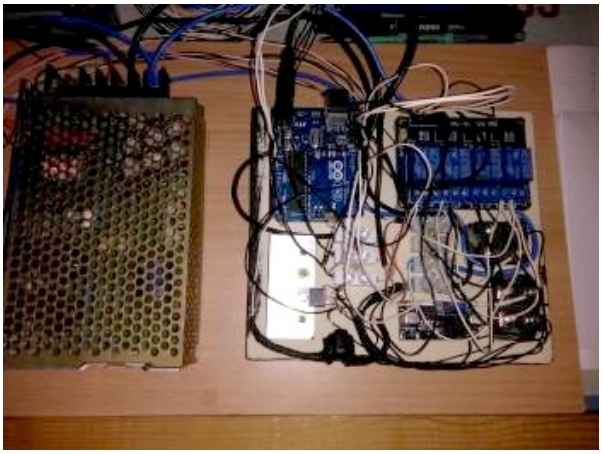
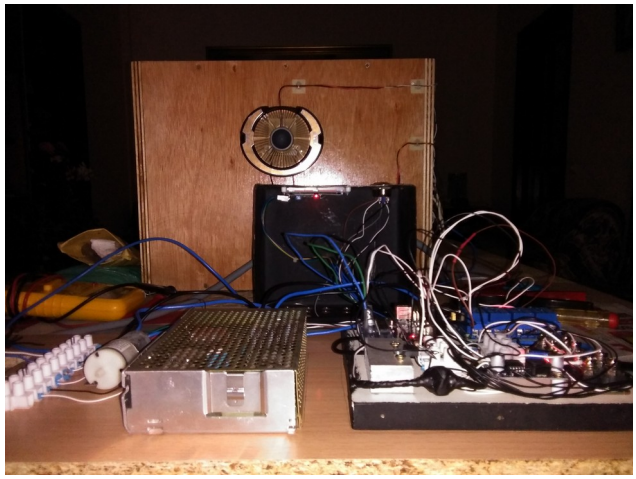
	
Resultado final	Resultado final puesto en marcha

2.4.3. Montaje tablero de control

En cuanto al desarrollo del tablero de control evitamos usar una PCB para ahorrarnos tiempos en montaje y diseño. Tratamos por tanto de tirar de otros recursos con más ingenio optimizando el espacio que tenemos en el tablero y solucionando los problemas de ubicación de componentes según nos surgen sobre la marcha.

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>Colocamos todos los componentes sobre el tablero para plantear la distribución mas óptima</td><td>Tenemos en cuenta que el espacio es reducido y tiene que coger todo...</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Lo tenemos claro</td><td>Comenzamos con el cableado</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Colocando cables con puntas de electricidad para crimpar o conectores DuPont para Arduino según convenga</td><td>Seguimos con el proceso...</td></tr>
</table>

	
Y cableamos la caja de control principal	Ya tenemos todos los componentes del circuito
	
Pasamos al trafo del humidficador	Lo aislamos con cinta de caucho
	
Vista del Módulo descender DC/DC Buck 30V-3V	Vista de la Placa de 4 Relés Optoacoplados para Arduino

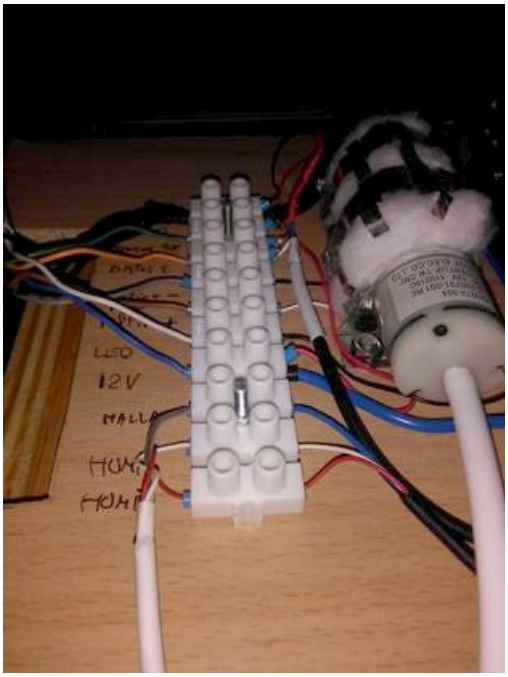
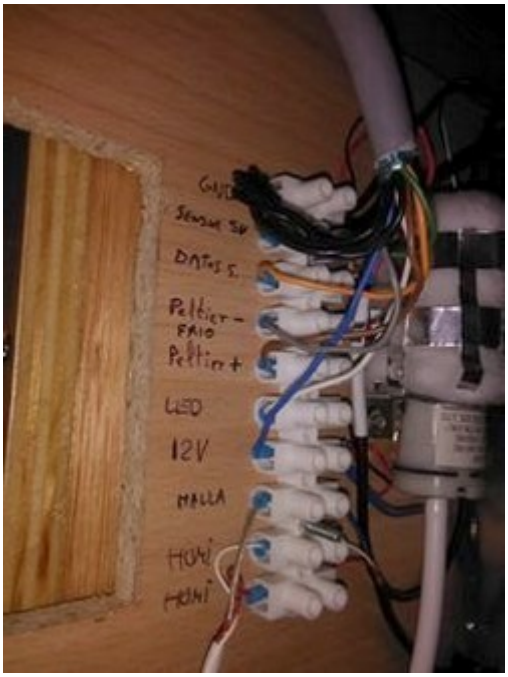
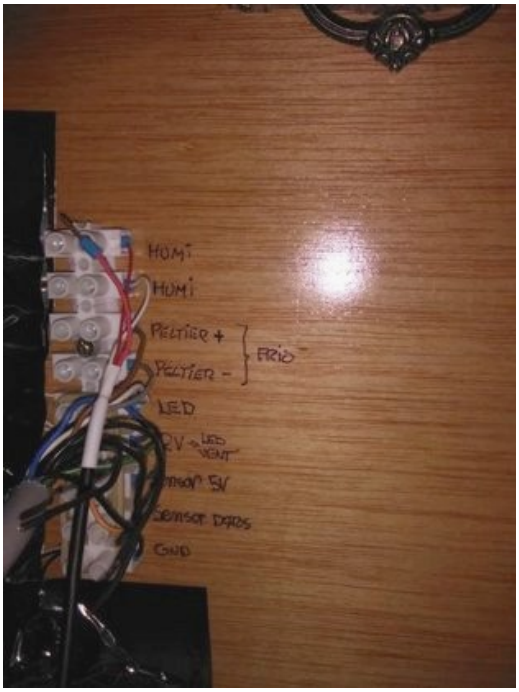
	
Cableado finalizado	Vista frontal del cableado
	
Pasamos a colocar el Display LCD en la caja de hierro de 0,5mm y esmaltada con negro mate que fabricamos. Separamos el LCD de la caja con unas arandelas de plastico hechas a medida para evitar contactos indeseados del display con la chapa de la caja de control.	Y colocamos los pulsadores de montaje en chasis con su cableado correspondiente
	
Vista final: tablero puesto en marcha	Vista lateral

	
Vista final: parte posterior	Vista de la bomba de aire con su filtro casero de algodón y cinta de aluminio
	
Vista final: conexión bornero bomba y fuente	Vista final: con el trafo del humidificador y su cable pegados al chasis de la fuente (proceso de apantallamiento para solucionar interferencias)
	
Vista final: parte trasera de la fuente de 12V	Vista final: parte trasera de la caja de control donde se aprecian los agujeros practicados para mejorar la ventilación; la ranura de salida de los cables; el conector USB; y el jack de alimentación del Arduino

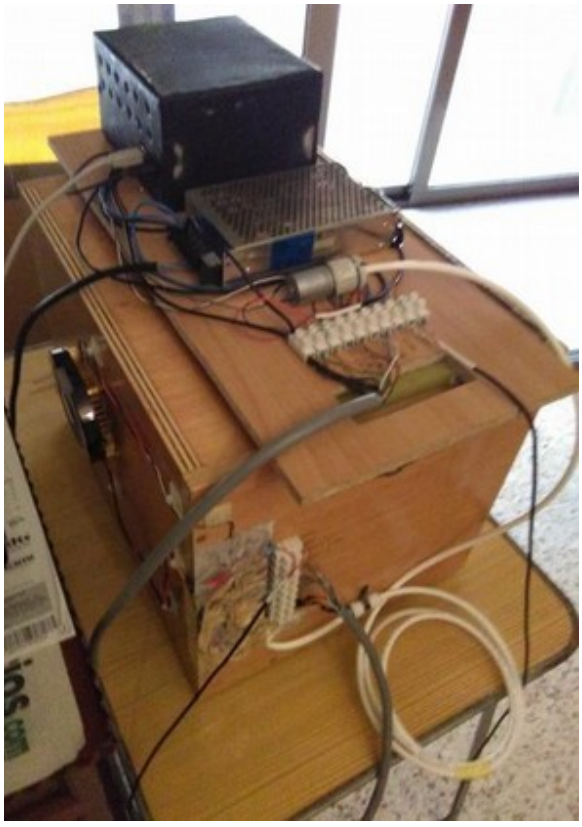
	Vista final del tablero de control una vez finalizado todas la mejoras en el sistema
	Vista final frontal del tablero puesto en marcha

2.4.4. Conexionado entre el habitáculo y el tablero de control

Ya solo nos falta efectuar la conexión entre ambas partes.
Utilizamos borneros de conexión (también llamadas fichas de empalme) de 12 para realizarla

<table>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Utilizamos el cable de audio para la conexión del humidificador la manguera de pares de 0’25x12 para el conexionado del resto del circuito</td><td>Bornero del tablero de control.
Escribimos en el tablero donde va dirigido cada cable</td></tr>
<tr><td>  </td><td> 
</td></tr>
<tr><td>Vista final del apantallado de cables de la caja</td><td>Bornero del exterior de la caja</td></tr>
</table>

Vista sin el tubo de la bomba



Vista con el tubo instalado

Vista final lateral

Vista final frontal del Invernadero puesto en marcha

2.5 Modelado de nuestro sistema de control Fuzzy para el invernadero

2.5.1. Introducción y tutorial de la herramienta empleada para el modelado del control (Toolbox Fuzzy Logic de Matlab)

Introducción

La simulación correspondiente al control que emplearemos en nuestro tabajo se desarrolló con el software de MATLAB. Dicho software presenta un apartado muy amplio en donde ofrece herramientas de simulación previamente desarrolladas.

El Fuzzy Logic Toolbox (FLT) es una herramienta para desarrollar programas difusos de manera amena en un entorno de MATLAB. El desarrollo se hace empleando la interfaz de usuario gráfica, por sus siglas en ingles GUI (graphical user interface).

Existen cinco herramientas gráficas para la construcción, edición y observación de un sistema de inferencia difuso dentro de un FLT.

Estas son:

- Sistema de inferencia difuso, por sus siglas en ingles FIS (Fuzzy Inference System).
- Función de membrecía, por sus siglas en ingles MF (Membership Function).
- Editor de reglas.
- Visualizador de reglas.
- Visualizador de superficie.

Tutorial Toolbox Fuzzy Logic de Matlab.

1. Acceso:

Para acceder al Toolbox Fuzzy se debe escribir la palabra fuzzy en la línea de comandos y luego pulsar Enter. En el caso de encontrar un error, por no hallarse cargado el toolbox se debe introducir el CD de instalación de Matlab. El menú al cual se debería acceder es el siguiente:

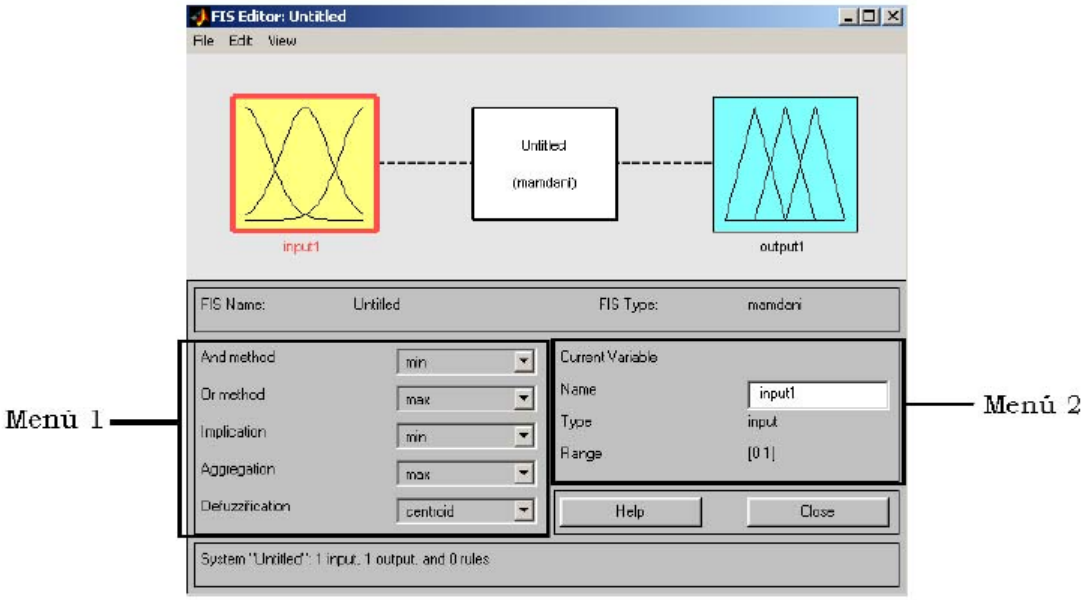


Figura B1. Menú principal del Fuzzy Toolbox, FIS Editor.

En el Menú 1, se podrá modificar los métodos de los operadores lógicos and y or, los métodos de implicación, de agregación y de defuzificación.

En el Menú 2, se podrá cambiar el nombre de la variable que se encuentre seleccionada, por ejemplo, modificar el nombre “input1” por “flujo de agua”.

2. Elección de Modelo:

Para elegir el tipo de modelo a usar, Sugeno o Mamdani, se debe acceder al menú
File -> New FIS... -> Mamdani (Sugeno).

3. Variables y Funciones de Pertenencia:

Para agregar alguna variable, ya sea de entrada o de salida, se debe seleccionar el menú *Edit -> Add Variable -> Input (Output).*

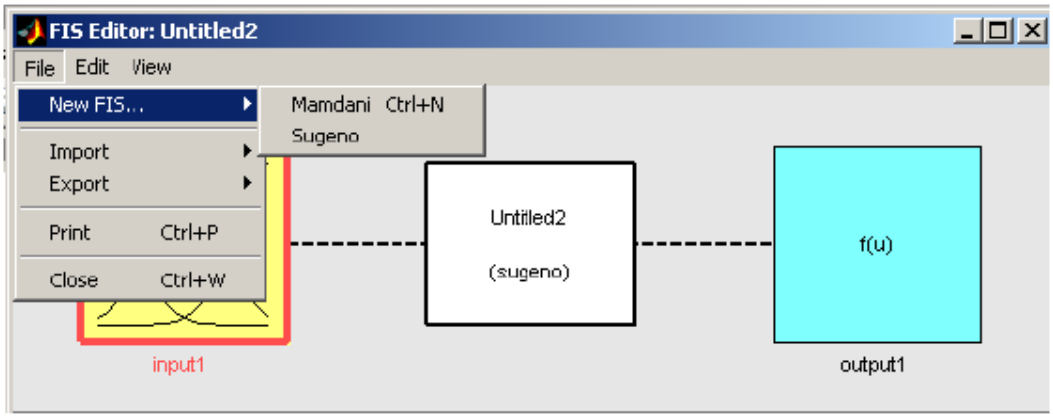


Figura B2. Elección de Modelo

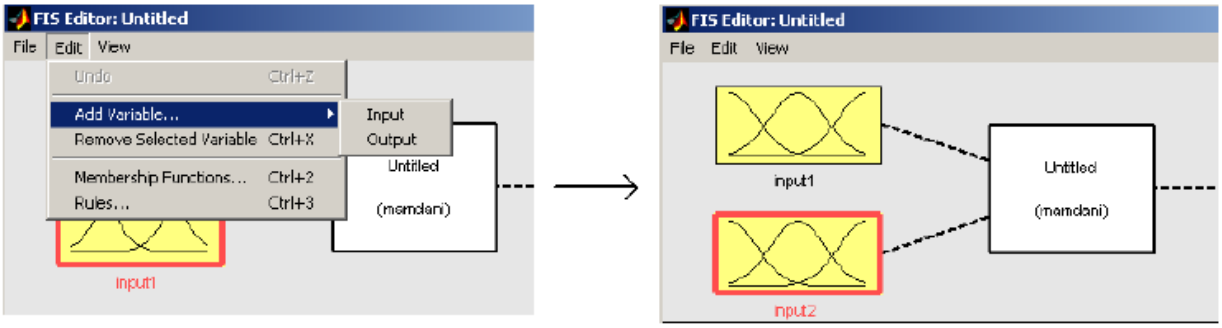


Figura B3. Al agregar una variable, es posible visualizarla en el menú gráfico. La variable actualmente seleccionada aparece enmarcada en rojo.

Las funciones de pertenencia, tanto para las variables de entrada como para las de salida, se modifican en un menú especial Membership Function Editor que aparece al hacer doble click en la variable de interés.

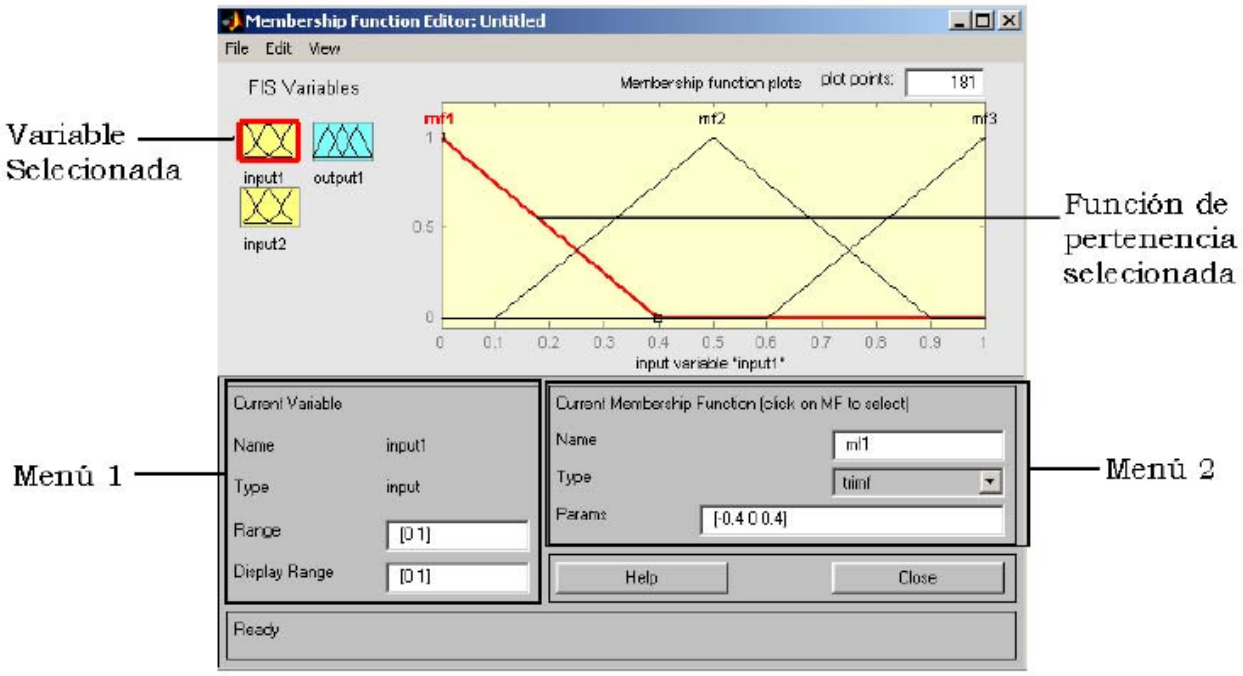


Figura B4. Editor de Funciones de Pertenencia, Membership Editor

En el Menú 1, se puede modificar el rango de la función de pertenencia, en el cual la función estará

definida.

En el Menú 2, es posible modificar el nombre de la función de pertenencia, los parámetros de la función de pertenencia y también su forma, la cual está seleccionada triangular en este caso, siendo ésta la más común.

4. Reglas del Modelo:

Para poder modificar las reglas del modelo se debe acceder al Rule Editor, haciendo doble click sobre el modelo.

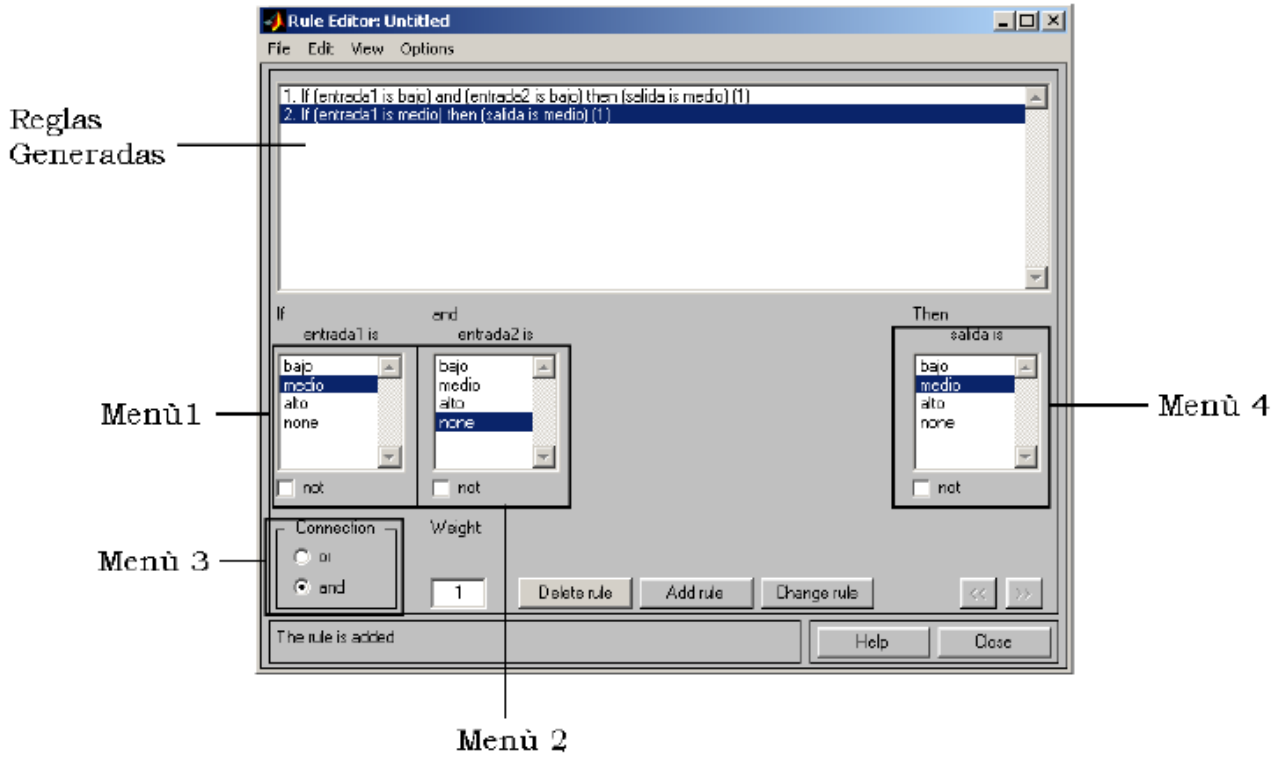


Figura B5. Editor de Reglas, Rule Editor

Según el número de variables de entrada y salida que existan y sus funciones de pertenencia será el número de reglas que es posible generar. En el Menú 1 se selecciona el valor que toma la primera variable de entrada, en el Menú 2, el valor que toma la segunda variable de entrada (si es necesario es posible negarla marcando *not*). En el Menú 3, se selecciona el tipo de conexión lógica entre ambos valores seleccionados (*and, or*), finalmente, en el Menú 4, se selecciona la salida que deberá entregar el controlador para los valores de entrada ya indicados. Luego, se presiona el botón *Add rule*, y la regla es agregada.

Para Eliminar una regla basta seleccionarla y apretar el botón *Delete rule*. Para modificarla se debe hacer click en el botón *Change rule*.

5. Implementación:

Para poder implementar el controlador es necesario guardar el modelo realizado, con el menú File -> export to... -> Disk (del FIS Editor), guardando así el trabajo.

2.5.2 Diseño del modelo a nivel teórico

Se definen 3 intervalos de funcionamiento en nuestro sistema de control en función de los datos recogidos en la bibliografía consultada. Adecuándonos a los rangos de temperatura (T) necesarios para el óptimo crecimiento de la seta de ostra (P*leurotus ostreatus*).

Consideramos dos etapas de crecimiento de la seta

1. Etapa de crecimiento del micelio (Grow)
2. Etapa de Fructificación (Fructif)

Nuestras variables de entrada del sistema serán:

ΔT *(Delta T)* → incremento de temperatura
ΔH *(Delta H)* → incremento de humedad

Y la variable de salida de nuestro sistema fuzzy:

Caudal de aire de la bomba

Nuestro sistema de intervalos es independiente de la etapa en la que nos encontremos.
Pero si que varían los rangos de T en función de la etapa:

- Etapa GROW
  - Intervalo1: $T \leq 23\,ºC$
  - Intervalo 2: $23\,ºC < T < 27\,ºC$
  - Intervalo 3: $T \geq 27\,ºC$

- Etapa FRUCTIF
  - Intervalo1: $T \leq 11\,ºC$
  - Intervalo 2: $11\,ºC < T < 14\,ºC$
  - Intervalo 3: $T \geq 14\,ºC$

Definimos las reglas de nuestro sistema fuzzy a nivel teorico

- **Intervalo 1:** El sistema esta calentando
  Nuestro sistema debe ajustarse al rango de T adecuado.

  Por tanto:
  → $\Delta T > 0\ y\ \Delta H > 0$ *Voy bien;* Mucho caudal de aire
  → $\Delta T < 0\ y\ \Delta H < 0$ *Voy mal;* Poco caudal de aire
  → $\Delta T = 0\ y\ \Delta H = 0$ *Debo reducir caudal;* Caudal medio de aire

- **Intervalo 2:** Sistema en equilibrio térmico
  Nuestro sistema debe mantener una T en equilibrio.
  Por tanto consideramos unicamente la temperatura como variable a estudio.

  Condiciones:
  → $\Delta T > 0$ Voy mal; Aplico poco caudal de aire
  → $\Delta T < 0$ Voy mal; Aplico poco caudal de aire

→ $\Delta T = 0$ Voy bien; Aplico caudal medio

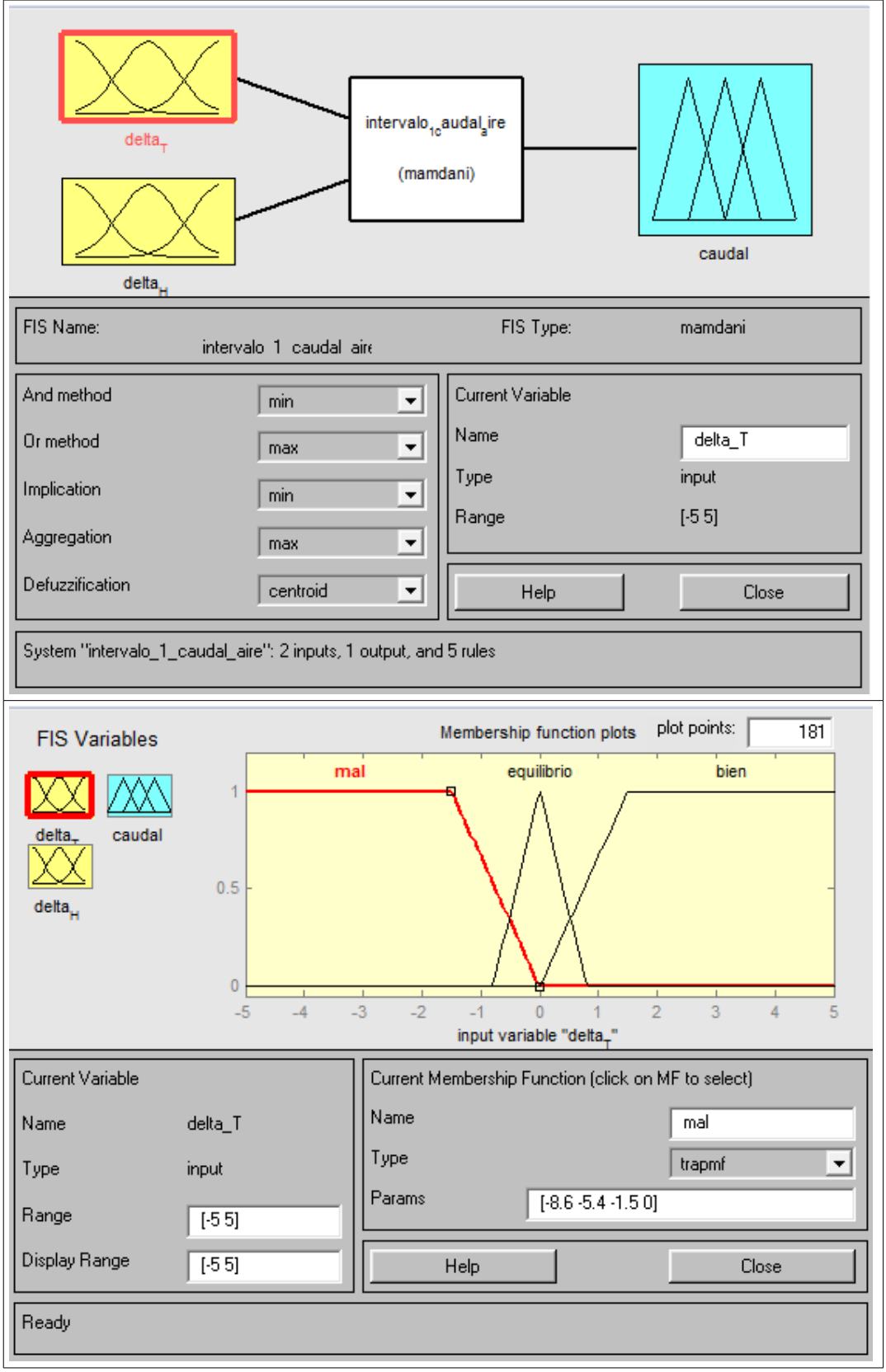
- **Intervalo 3:** El sistema está enfriando
  Nuestro sistema debe ajustarse al rango de T adecuado.

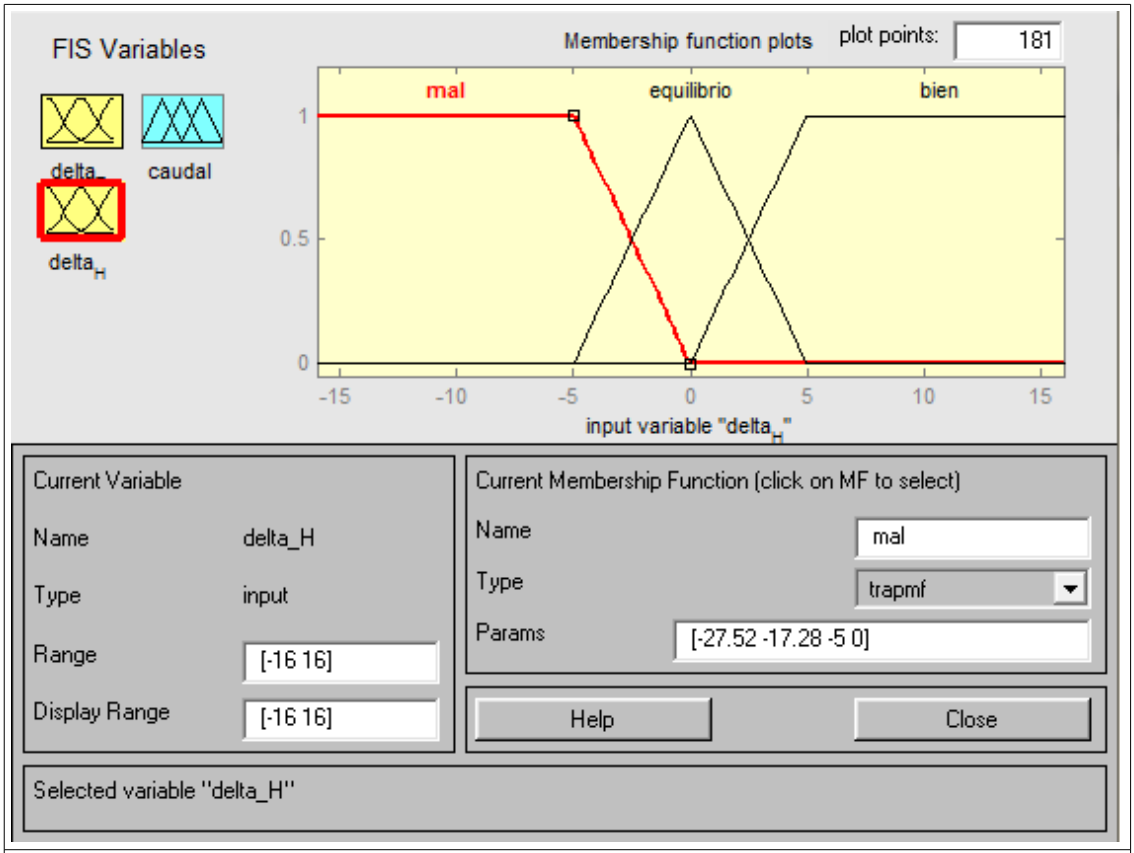
  Por tanto:
  → $\Delta T < 0$ *y* $\Delta H > 0$ *Voy bien;* Mucho caudal de aire
  → $\Delta T \geq 0$ *y* $\Delta H < 0$ *Voy mal;* Poco caudal de aire
  → $\Delta H = 0$ *Debo reducir caudal;* Caudal medio de aire

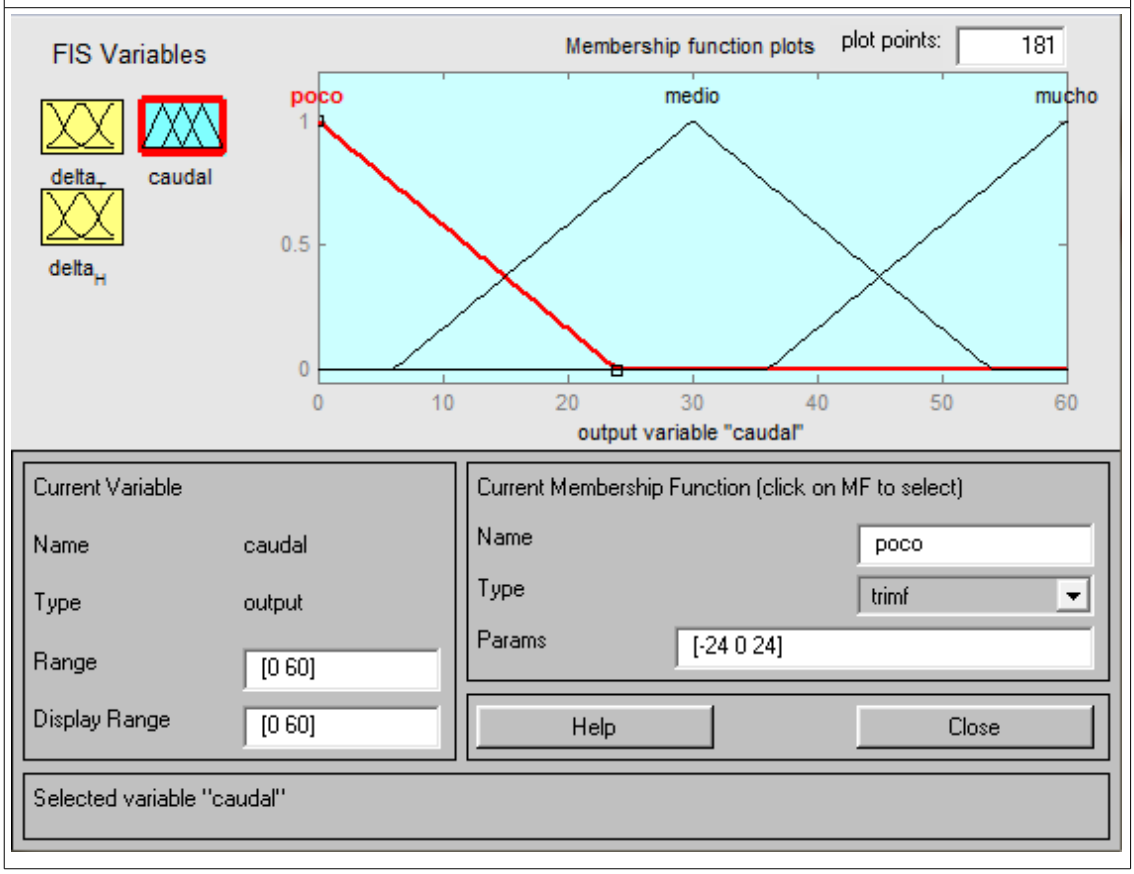
Consideraciones: Nos damos cuenta a la hora de modelar el sistema que la variable más importante de entrada será la temperatura ya que al ser un habitáculo cerrado nuestro invernadero, la humedad será más fácil de controlar.

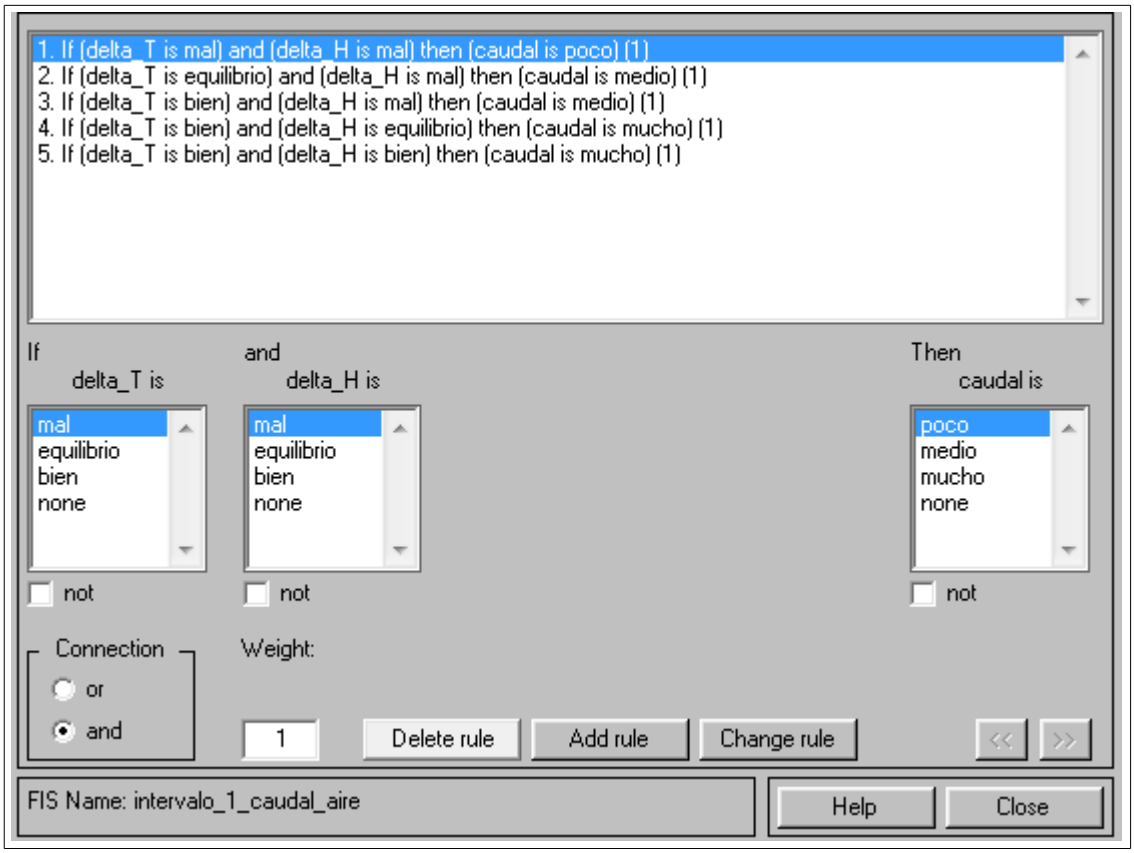
2.5.3 Introducción de datos y reglas del modelo diseñado, en la aplicación Toolbox Fuzzy Logic de Matlab

Intervalo 1

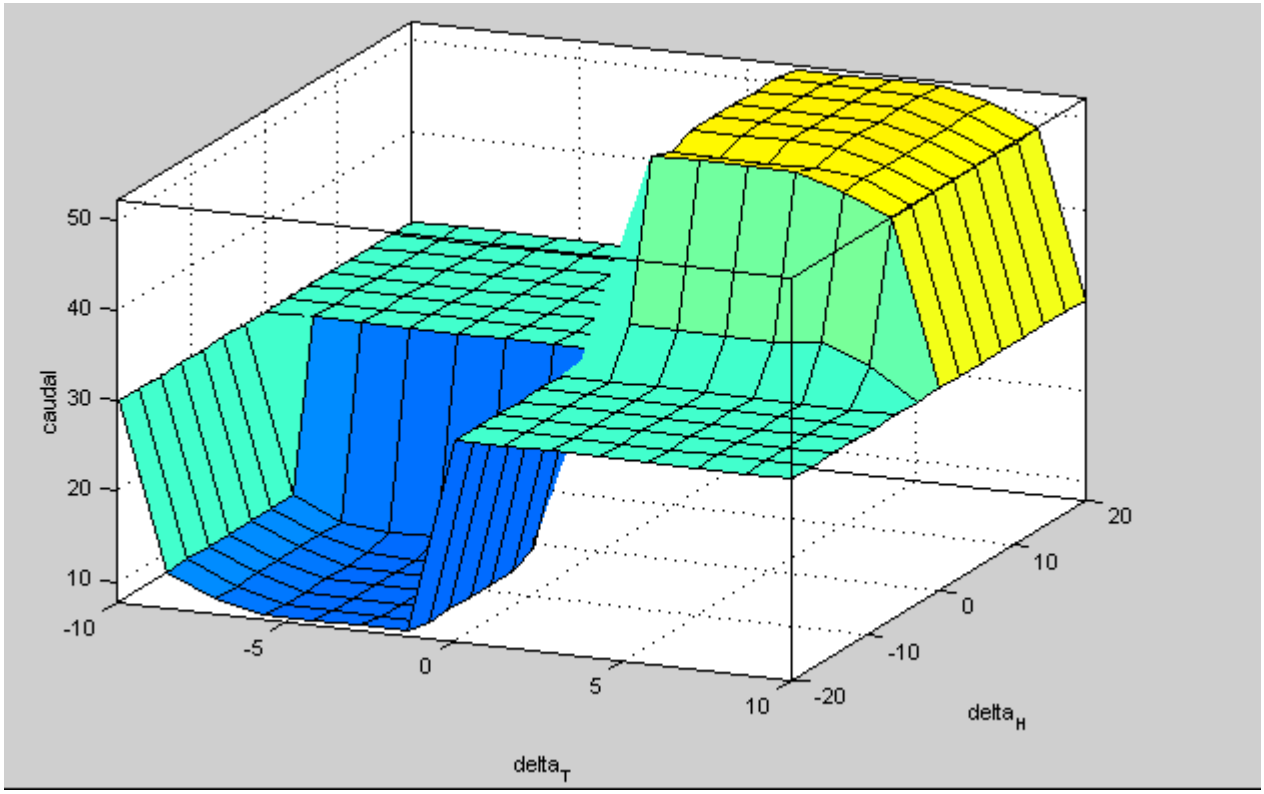



FIS Variables
Membership function plots
plot points:
181
delta_
caudal
delta_H
mal
equilibrio
bien
1
0.5
0
-15
-10
-5
0
5
10
15
input variable "delta_H"
Current Variable
Name
delta_H
Type
input
Range
[-16 16]
Display Range
[-16 16]
Current Membership Function (click on MF to select)
Name
mal
Type
trapmf
Params
[-27.52 -17.28 -5 0]
Help
Close
Selected variable "delta_H"


FIS Variables
Membership function plots
plot points:
181
delta_
caudal
delta_H
poco
medio
mucho
1
0.5
0
0
10
20
30
40
50
60
output variable "caudal"
Current Variable
Name
caudal
Type
output
Range
[0 60]
Display Range
[0 60]
Current Membership Function (click on MF to select)
Name
poco
Type
trimf
Params
[-24 0 24]
Help
Close
Selected variable "caudal"



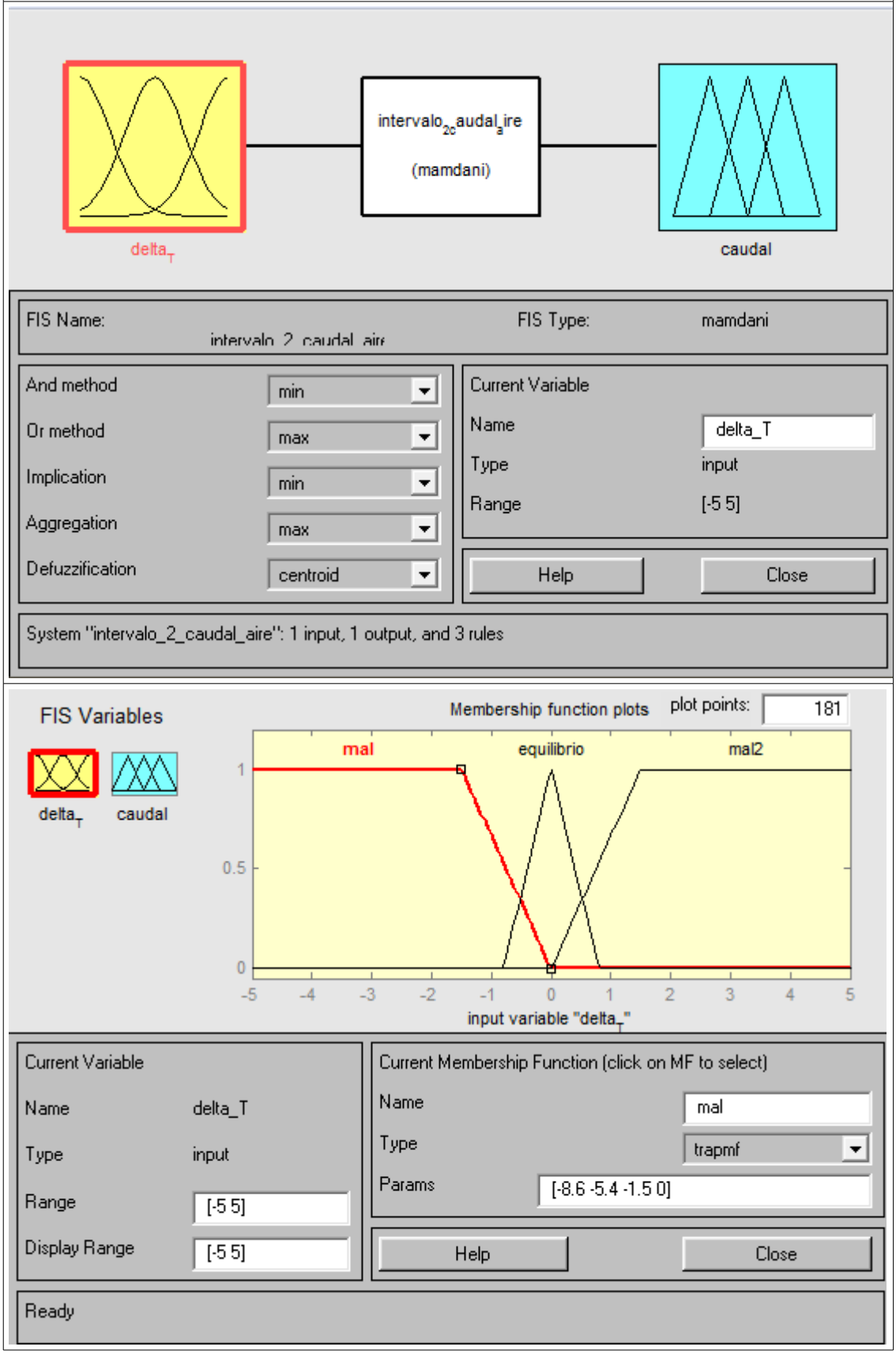
Gráfica generada por la aplicación del Intervalo 1:

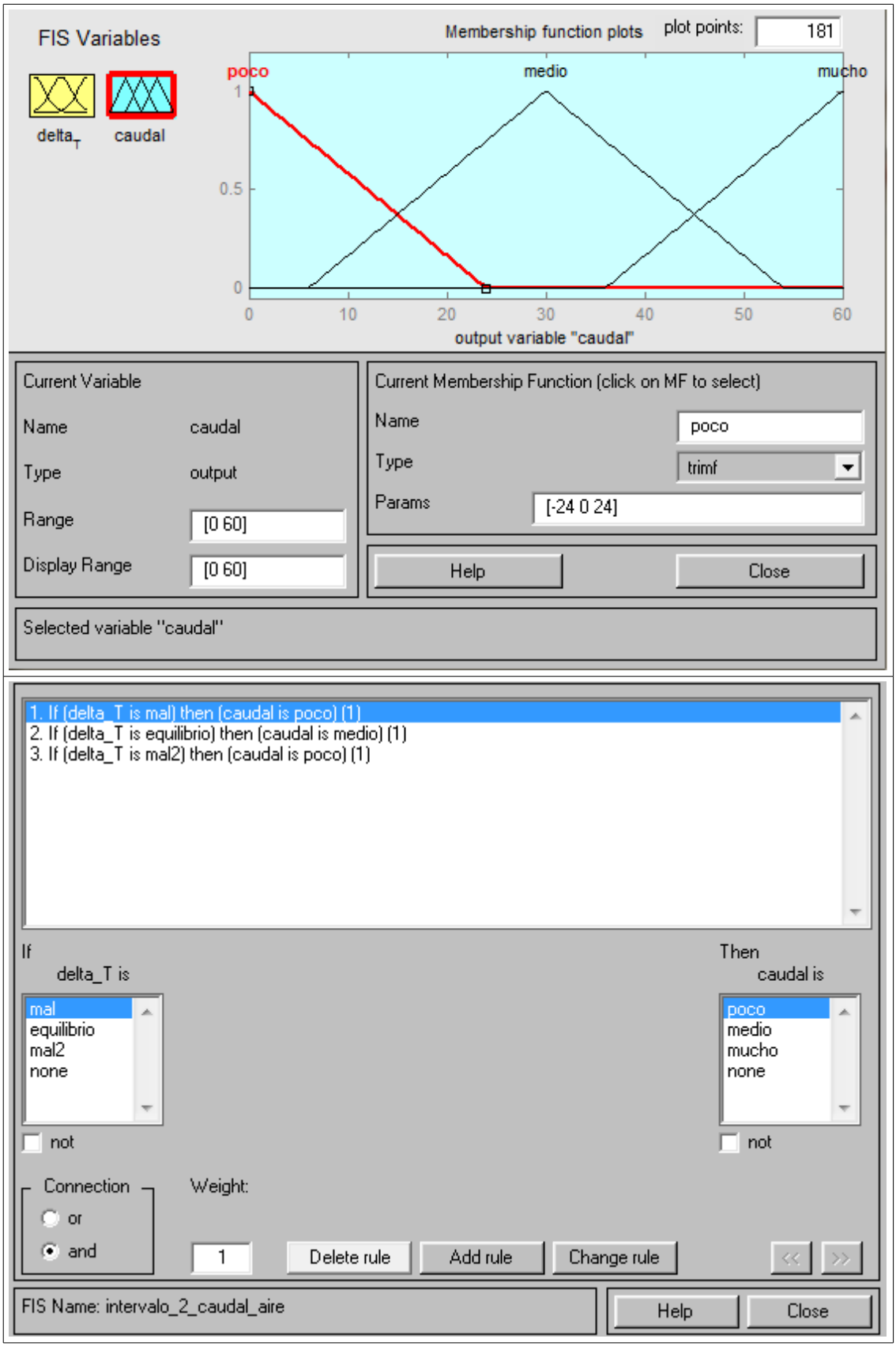


Archivo .fis generado por la aplicación:

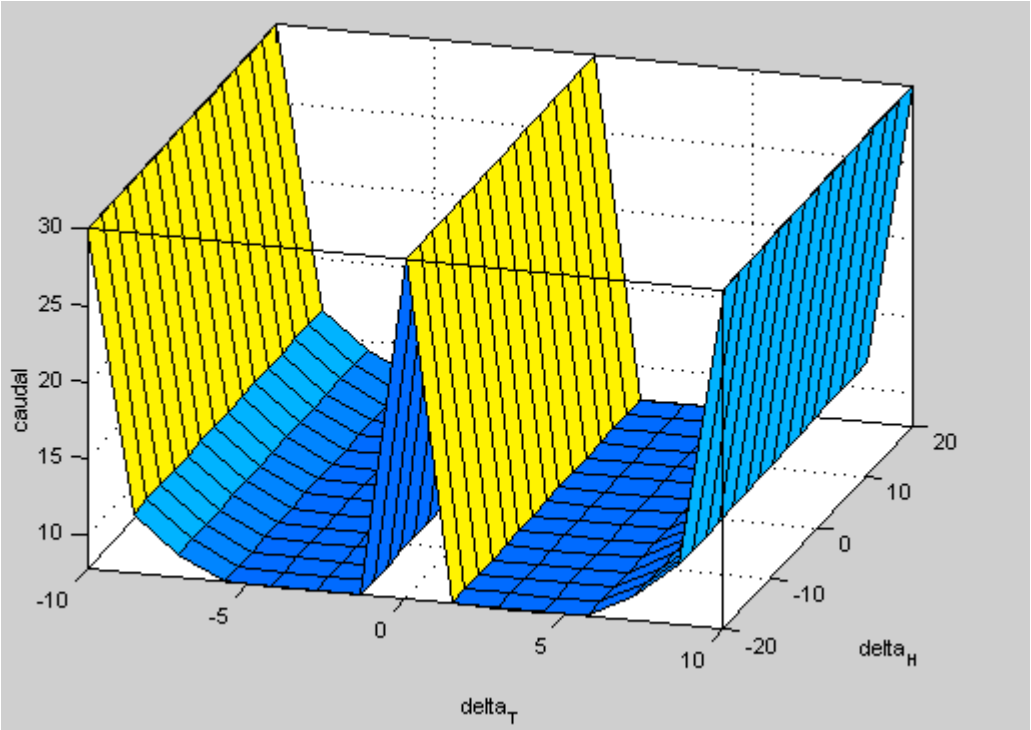
Ver código generado en anexos

Intervalo 2


intervalo_2c audal_aire
(mamdani)
delta_T
caudal
FIS Name:
intervalo_2_caudal_aire
FIS Type:
mamdani
And method
min
Or method
max
Implication
min
Aggregation
max
Defuzzification
centroid
Current Variable
Name
delta_T
Type
input
Range
[-5 5]
Help
Close
System "intervalo_2_caudal_aire": 1 input, 1 output, and 3 rules
FIS Variables
delta_T
caudal
Membership function plots
plot points:
181
mal
equilibrio
mal2
1
0.5
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
input variable "delta_T"
Current Variable
Name
delta_T
Type
input
Range
[-5 5]
Display Range
[-5 5]
Current Membership Function (click on MF to select)
Name
mal
Type
trapmf
Params
[-8.6 -5.4 -1.5 0]
Help
Close
Ready


FIS Variables
delta_T
caudal
Membership function plots
plot points:
181
poco
medio
mucho
1
0.5
0
0
10
20
30
40
50
60
output variable "caudal"
Current Variable
Name
caudal
Type
output
Range
[0 60]
Display Range
[0 60]
Current Membership Function (click on MF to select)
Name
poco
Type
trimf
Params
[-24 0 24]
Help
Close
Selected variable "caudal"
1. If (delta_T is mal) then (caudal is poco) (1)
2. If (delta_T is equilibrio) then (caudal is medio) (1)
3. If (delta_T is mal2) then (caudal is poco) (1)
If
delta_T is
mal
equilibrio
mal2
none
not
Then
caudal is
poco
medio
mucho
none
not
Connection
or
and
Weight:
1
Delete rule
Add rule
Change rule
FIS Name: intervalo_2_caudal_aire
Help
Close

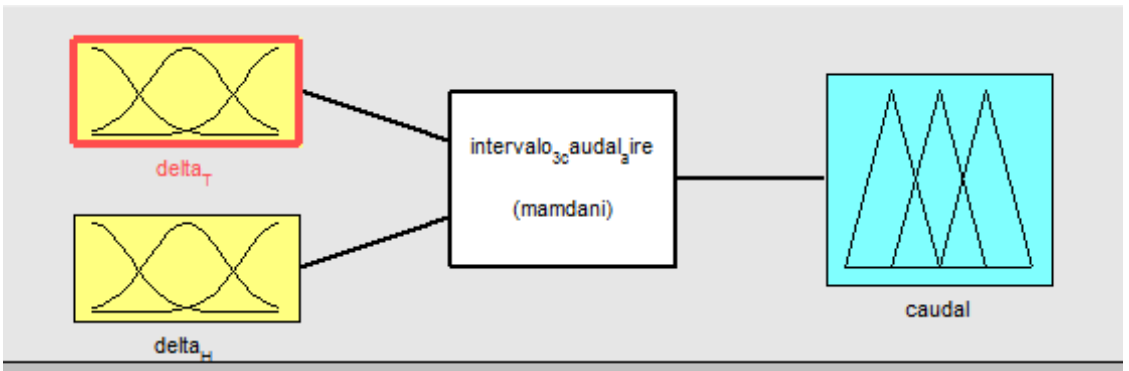
Gráfica generada por la aplicación para el Intervalo 2:



Archivo .fis generado por la aplicación:

Ver código generado en anexos

Intervalo 3



FIS Name: intervalo_3_caudal_aire

FIS Type: mamdani

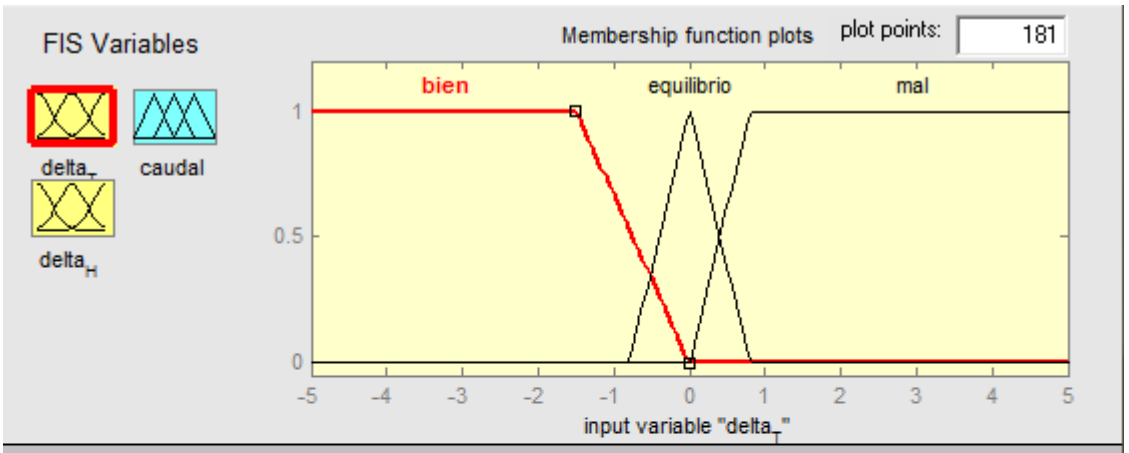
And method	min
Or method	max
Implication	min
Aggregation	max
Defuzzification	centroid

Current Variable

Name	delta_T
Type	input
Range	[-5 5]

Help | Close

System "intervalo_3_caudal_aire": 2 inputs, 1 output, and 3 rules



Current Variable

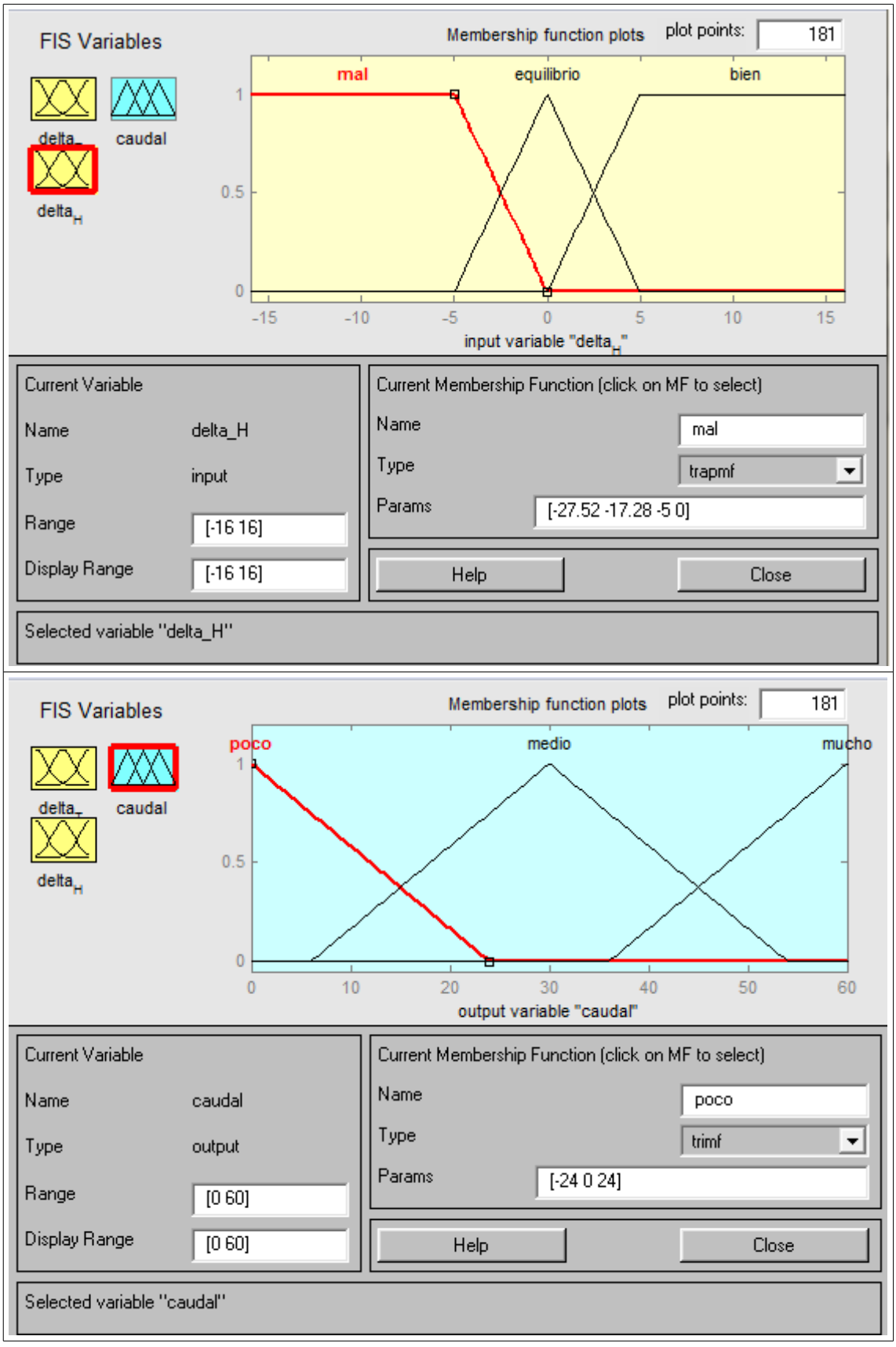
Name	delta_T
Type	input
Range	[-5 5]
Display Range	[-5 5]

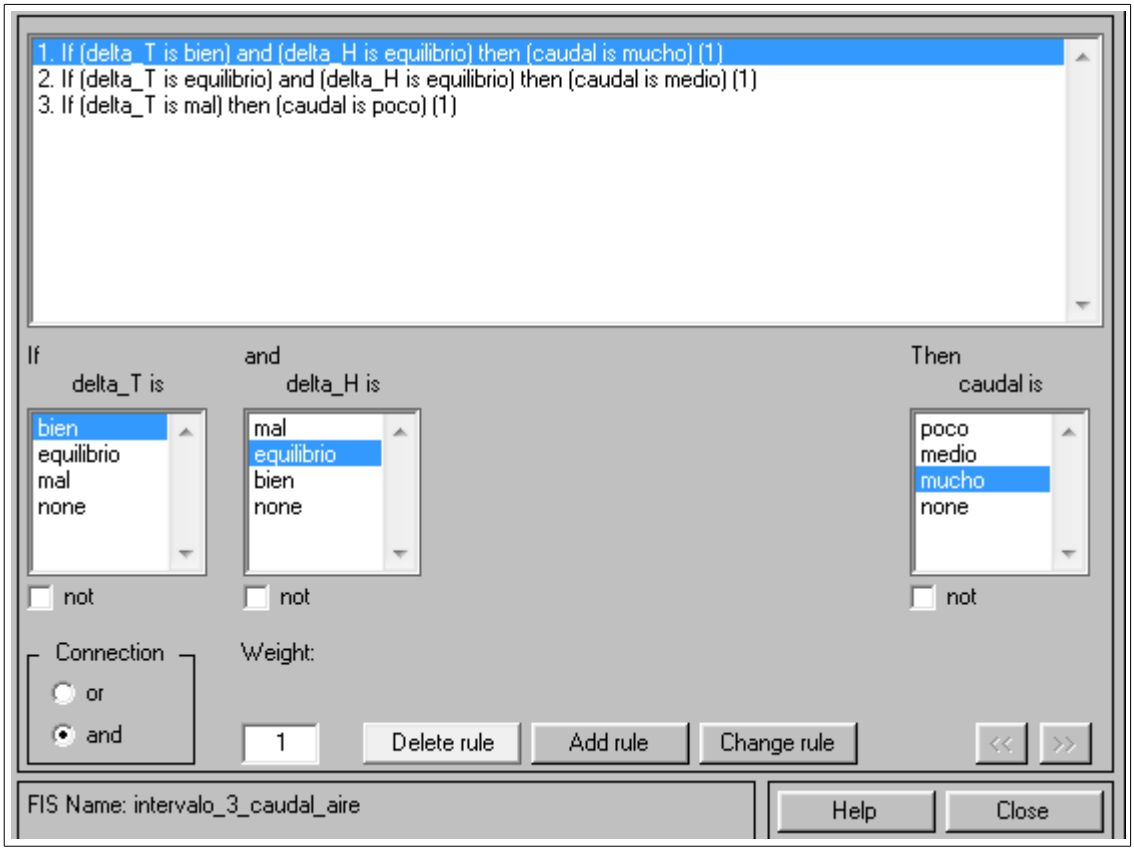
Current Membership Function (click on MF to select)

Name	bien
Type	trapmf
Params	[-8.6 -5.4 -1.5 0]

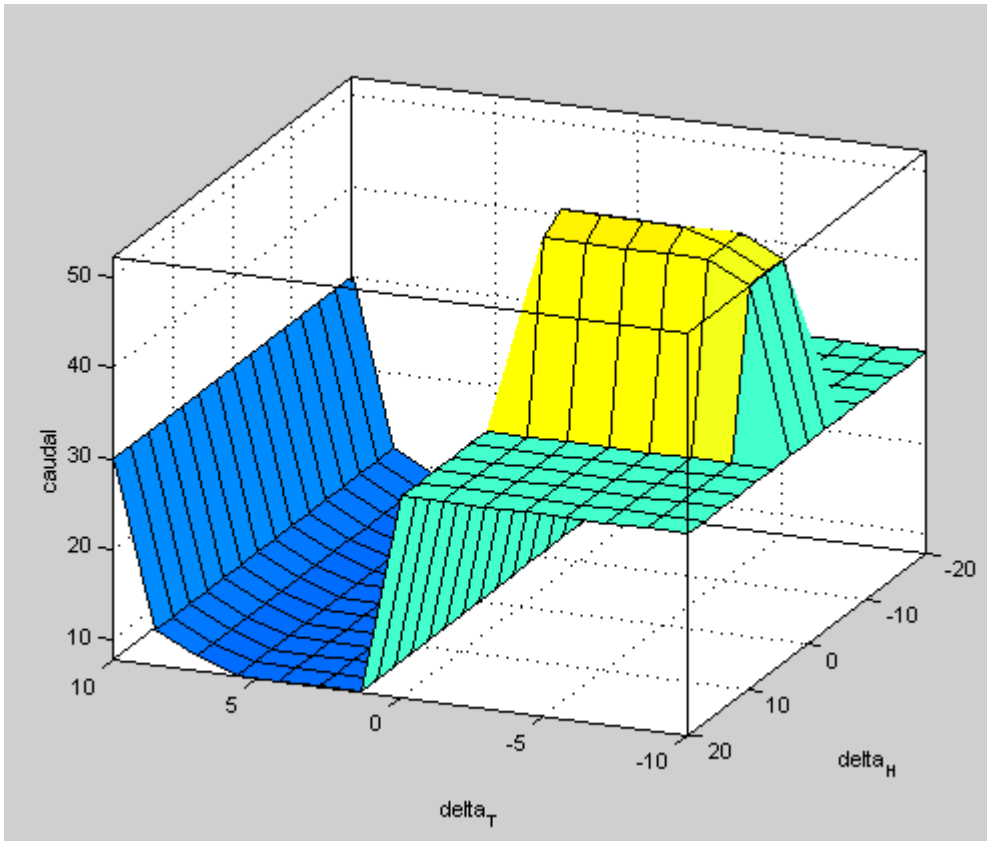
Help | Close

Ready


FIS Variables
Membership function plots
plot points:
181
mal
equilibrio
bien
delta_
caudal
delta_H
1
0.5
0
-15
-10
-5
0
5
10
15
input variable "delta_H"
Current Variable
Name
delta_H
Type
input
Range
[-16 16]
Display Range
[-16 16]
Current Membership Function (click on MF to select)
Name
mal
Type
trapmf
Params
[-27.52 -17.28 -5 0]
Help
Close
Selected variable "delta_H"
FIS Variables
Membership function plots
plot points:
181
poco
medio
mucho
delta_
caudal
delta_H
1
0.5
0
0
10
20
30
40
50
60
output variable "caudal"
Current Variable
Name
caudal
Type
output
Range
[0 60]
Display Range
[0 60]
Current Membership Function (click on MF to select)
Name
poco
Type
trimf
Params
[-24 0 24]
Help
Close
Selected variable "caudal"



Gráfica generada por la aplicación para el Intervalo 3:



Archivo .fis generado por la aplicación:

Ver código generado en anexos

2.5.4. Adaptación modelo de control .fis de Matlab a formato .ino para Arduino

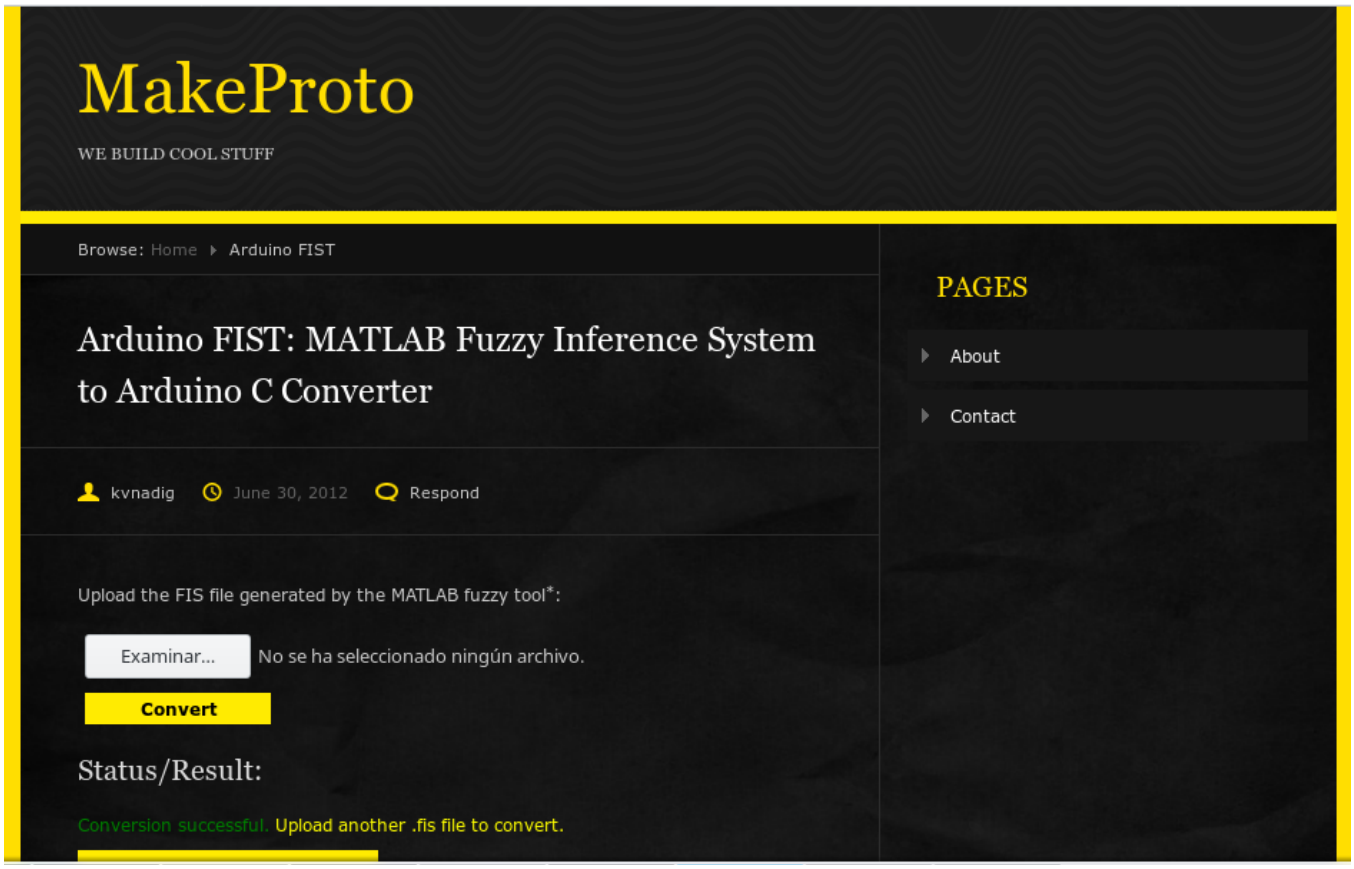
Una vez hemos obtenido nuestro modelo de sistema de control en formato .fis de Matlab, debemos adaptarlo a un formato que pueda interpretar el Arduino.

Indagando en la web encontramos una pagina en la que se puede adaptar un archivo .fis de Matlab a codigo C para Arduino en formato .ino

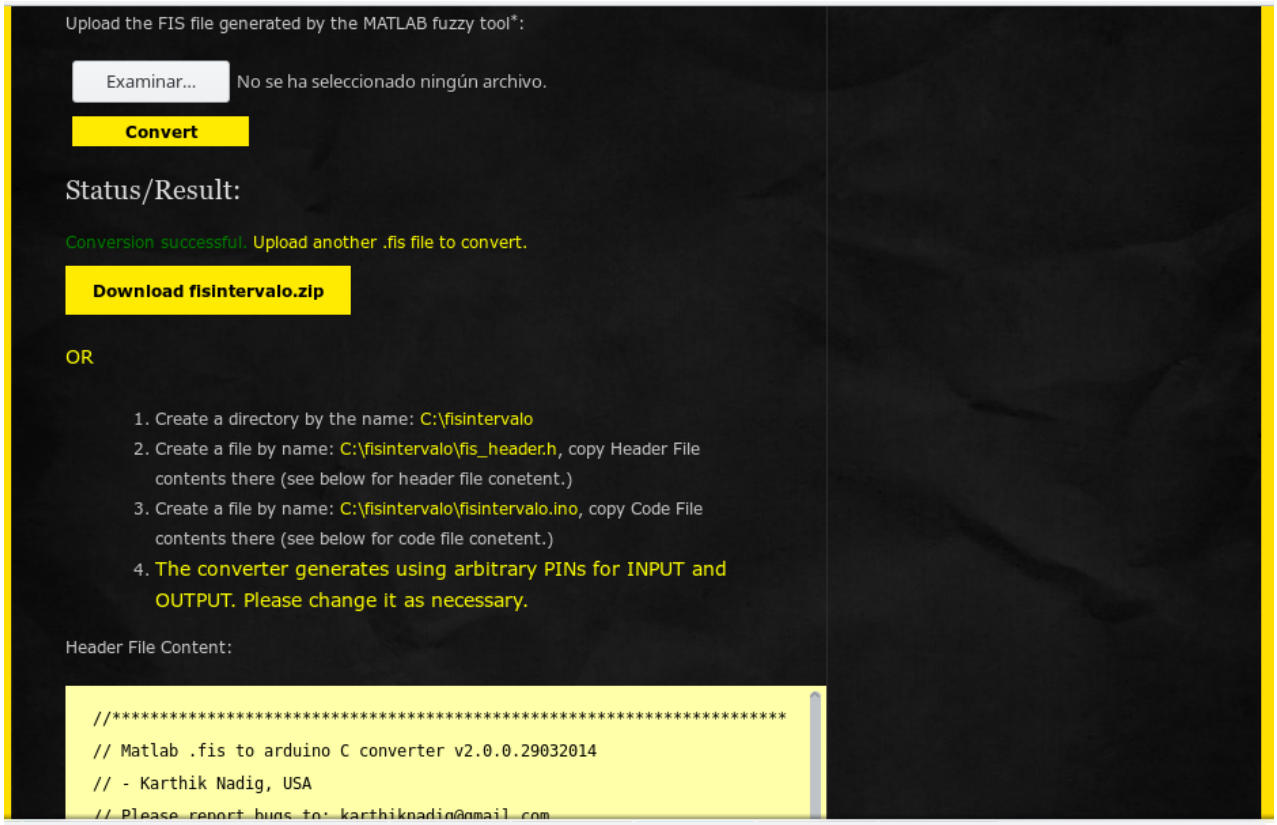
Esta es la web mencionada:

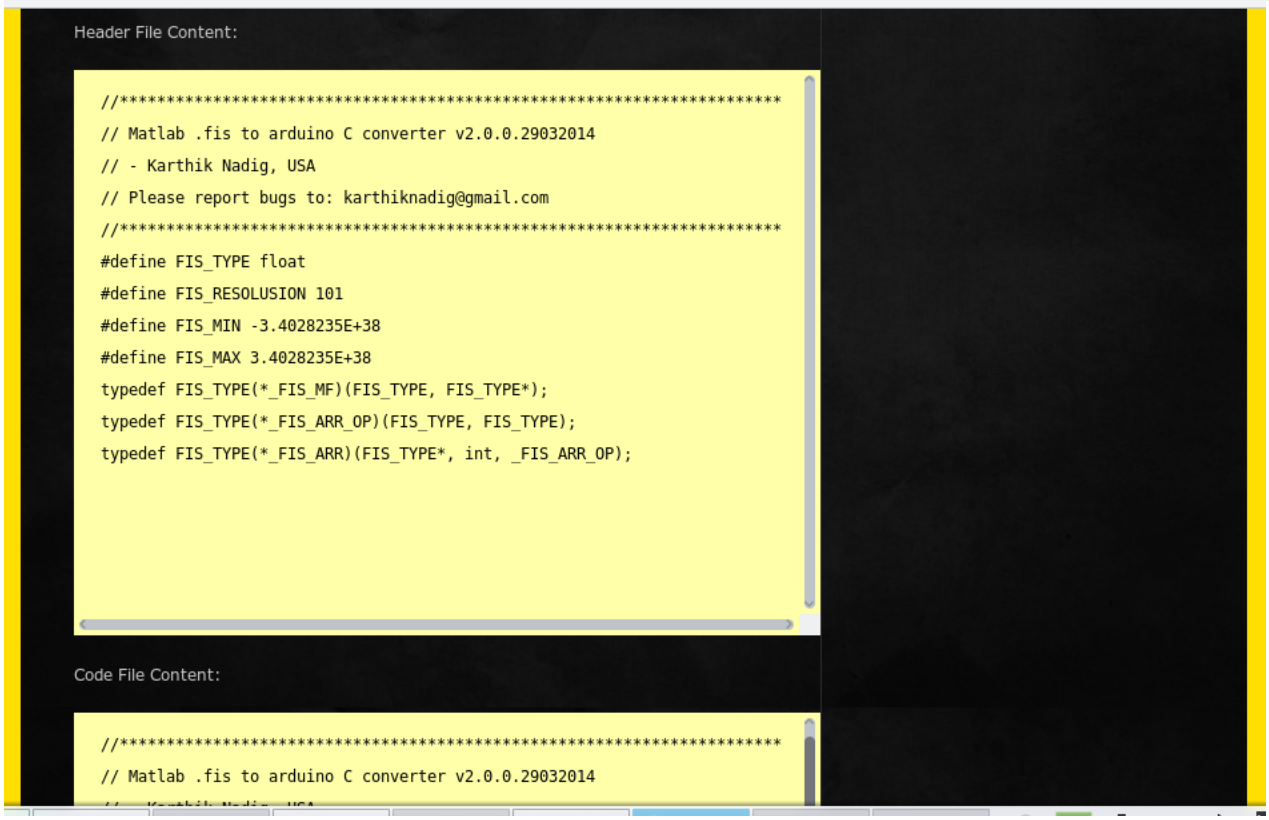
http://www.makeproto.com/projects/fuzzy/matlab_arduino_FIST/index.php

Pagina inicial donde te solicita la carga del archivo .fis:

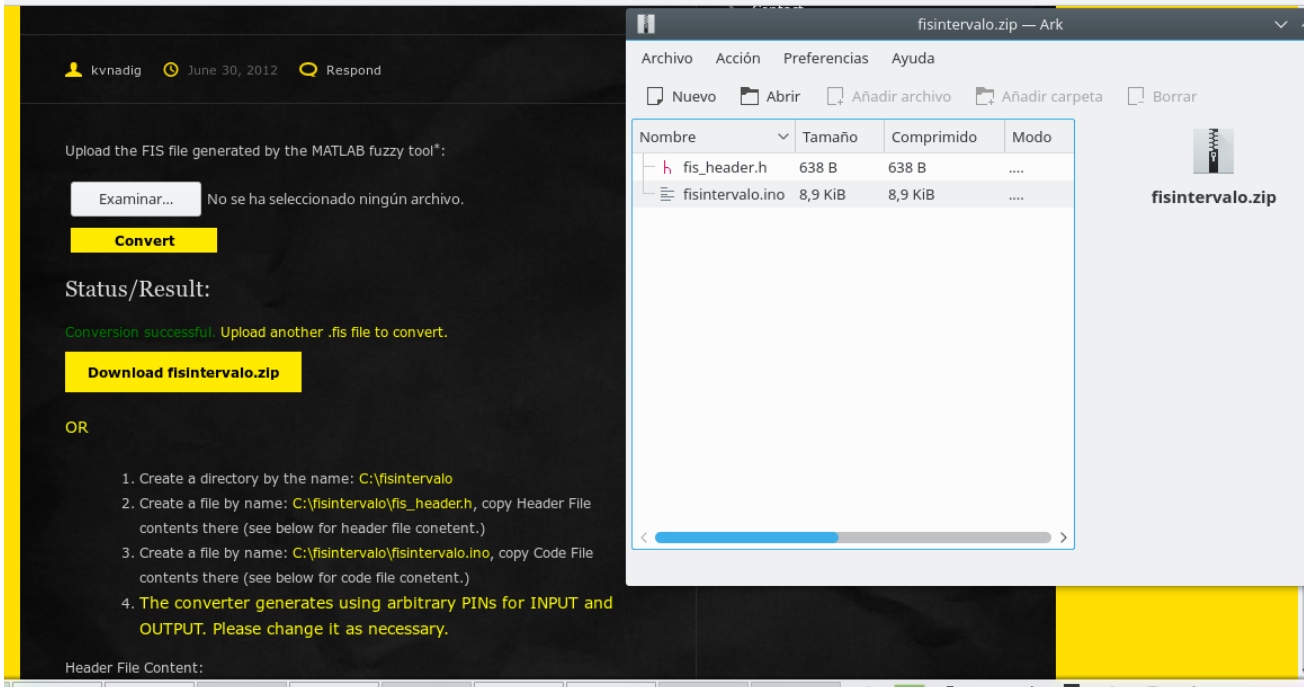


Una vez realizada la conversión:





Fichero zip generado por la aplicación con los dos archivos necesarios para trabajar en arduino con el modelo fuzzy *fis_header.h* y *fisintervalo.ino*:



2.5.5. Análisis de nuestro modelo fuzzy generado para Arduino en un entorno de programación en C

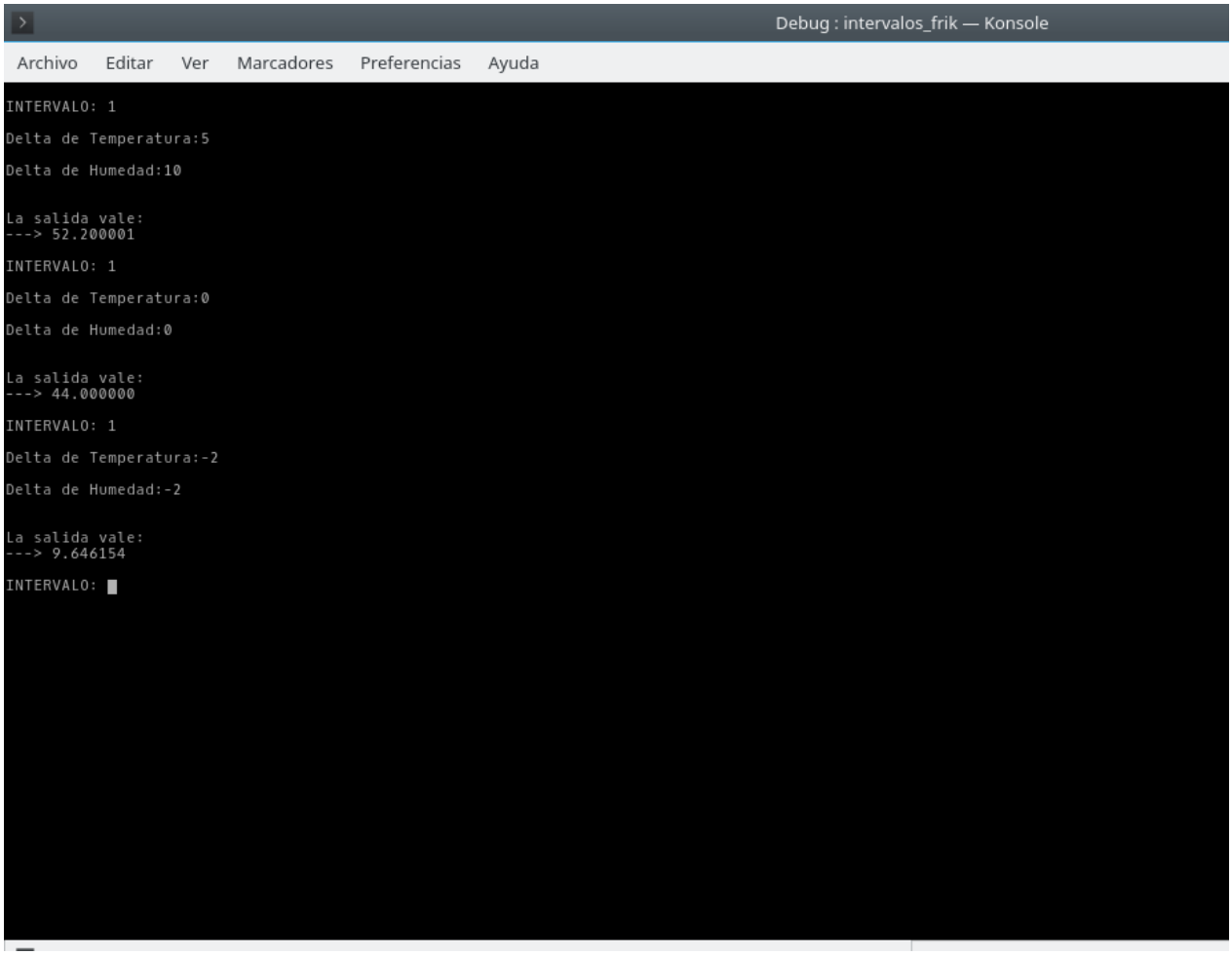
Para poder estar seguros de que nuestro modelo fuzzy generado por la web FIST se ajusta al modelo generado previamente en Matlab, ejecutamos dicho código en un entorno de programación en C tipo Code::Blocks o similar para generar nuestro sistema fuzzy.

Desarrollando un pequeño programa para ello que ejecutaremos en consola preguntaremos al usuario Intervalo, Delta T y Delta H. Y automáticamente nos generará un valor de caudal de aire calculado por fuzzy que mostrará en pantalla.

Código en C del programa empleado:

Ver código generado en anexos

Aplicación ejecutada en consola:

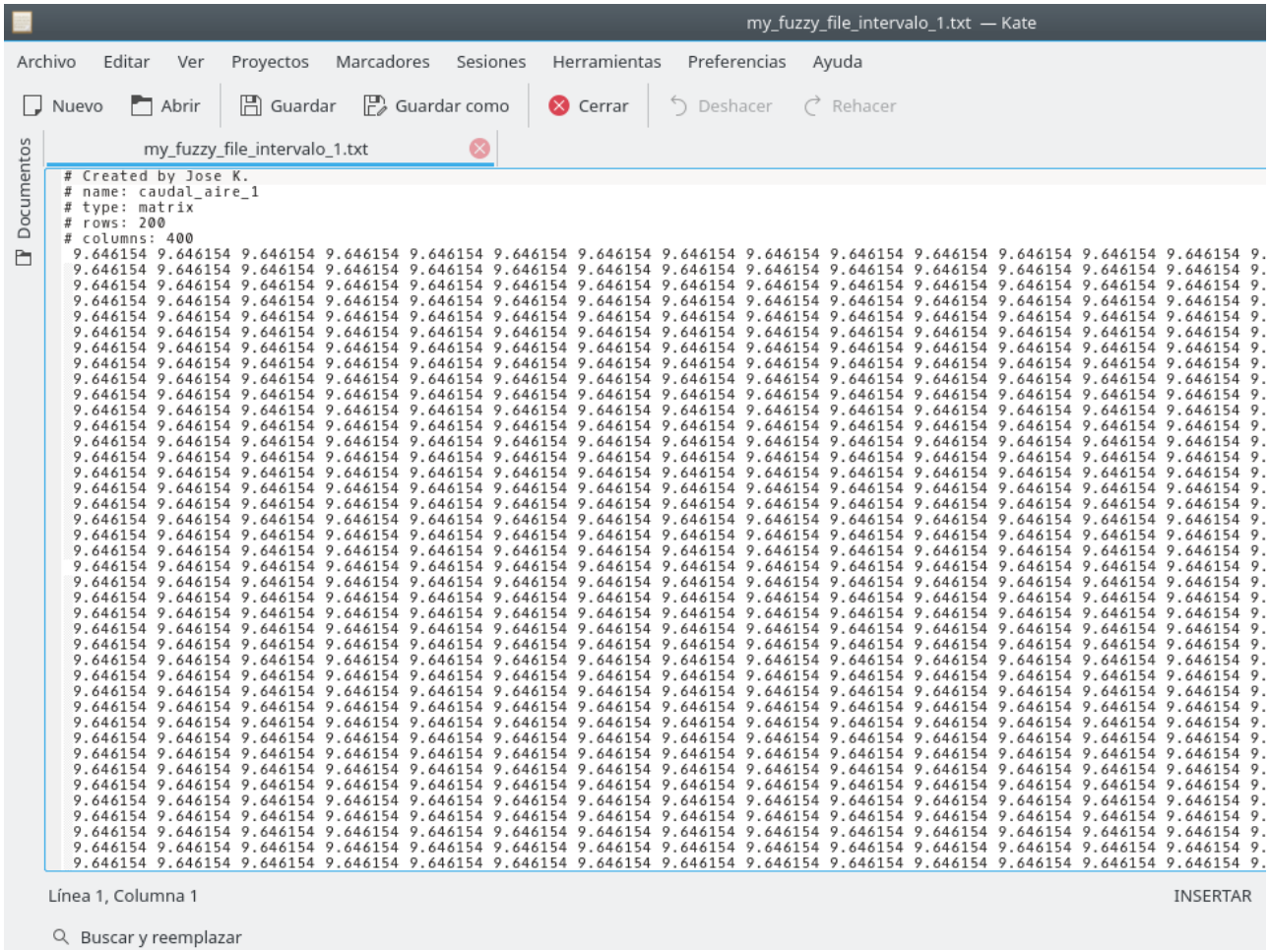


Además generamos un fichero con formato *.txt* con los valores calculados por nuestro modelo fuzzy para poder evaluarlos posteriormente.

El fichero generado se llamará my_fuzzy_file_intervalo_x.txt

- Donde *"x"* será el *número de intervalo* [1,3]

Ejemplo de archivo my_fuzzy_file_intervalo_x.txt generado:



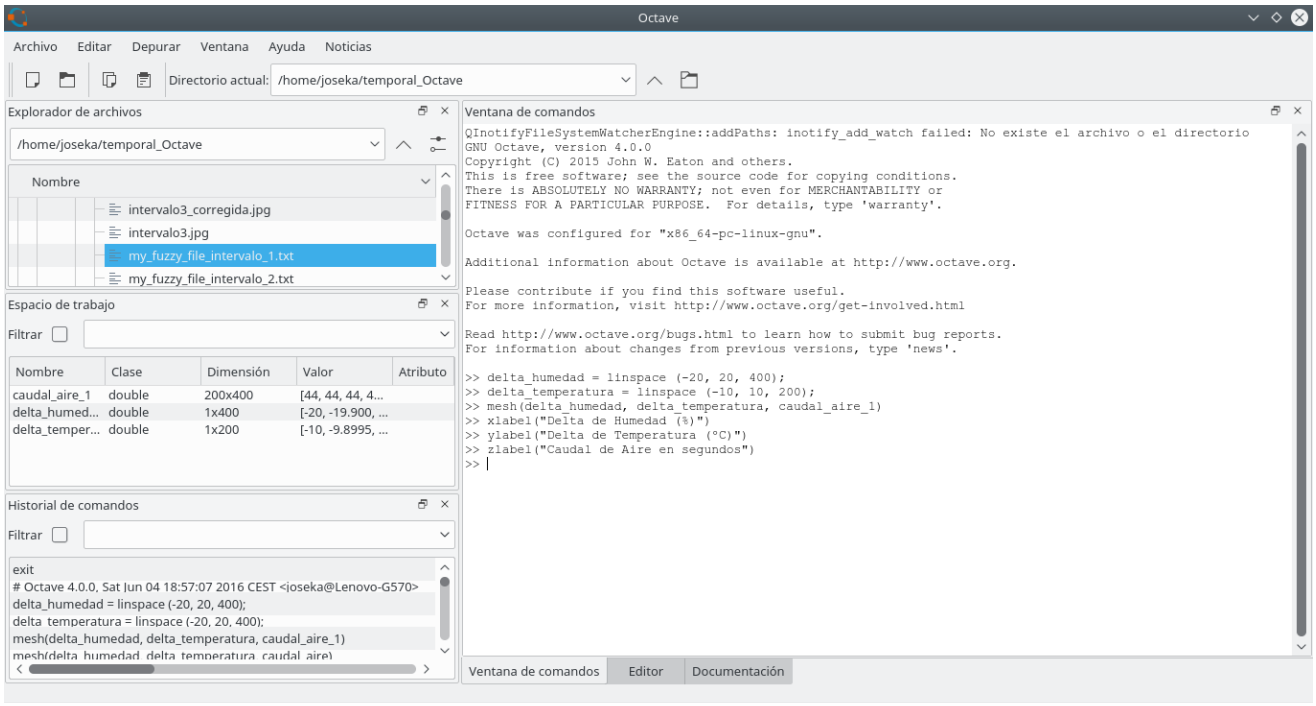
2.5.6. Tratamiento de datos generados con Octave

Para tratar los datos que genera nuestro fuzzy emplearemos el Octave.
Pasos a seguir:

- Cargamos nuestro fichero my_fuzzy_file_intervalo_x.txt
- Generamos la gráfica de nuestro sistema.

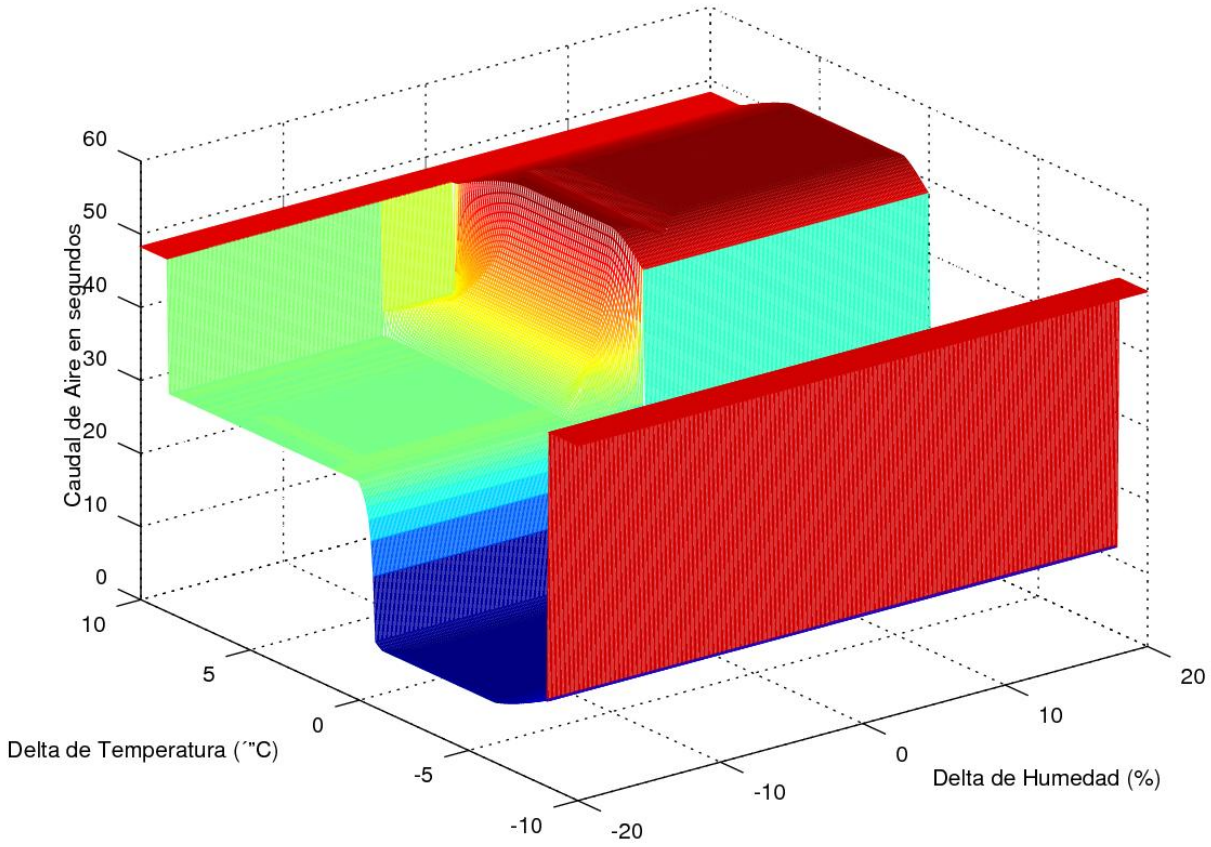
Comandos empleados con Octave para generar nuestras gráficas de los intervalos fuzzy en 3D:

Ver código generado en anexos

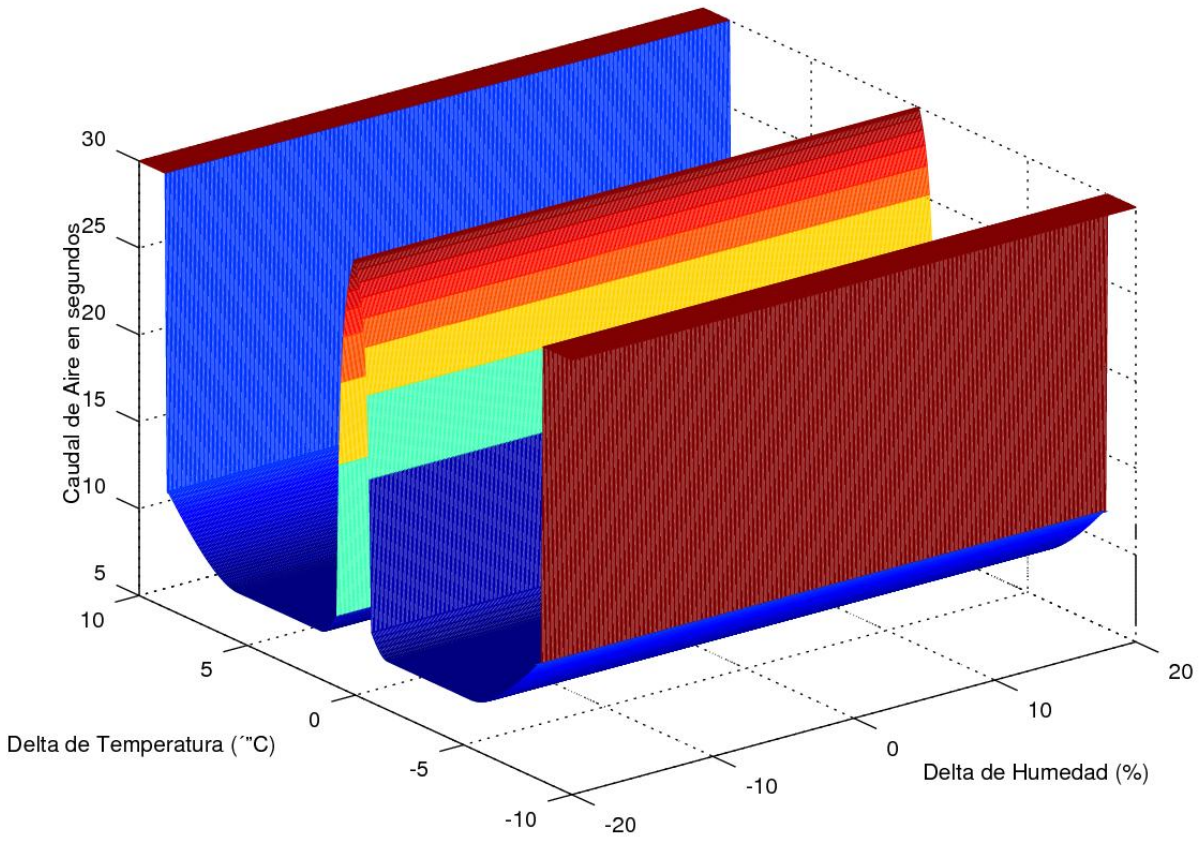


2.5.7. Gráficas generadas en Octave de los intervalos fuzzy

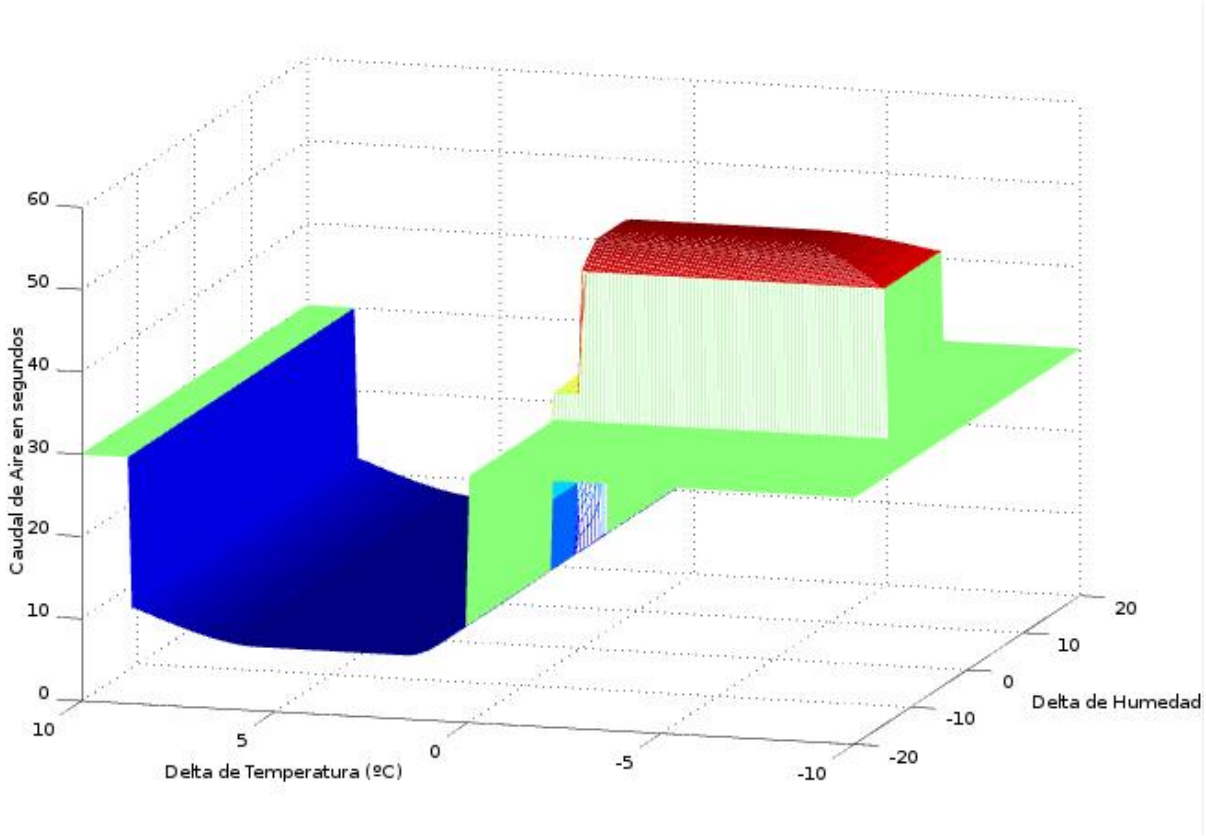
Intervalo 1



Intervalo 2

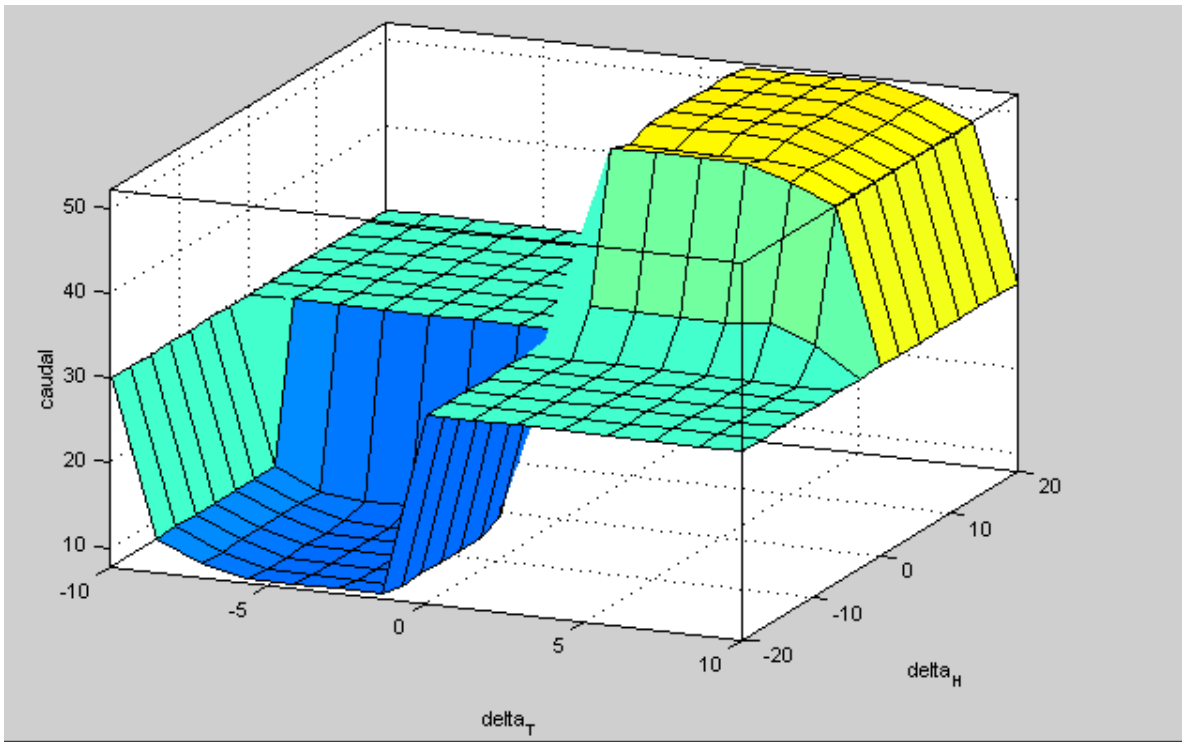

Caudal de Aire en segundos
30
25
20
15
10
5
10
5
0
-5
-10
Delta de Temperatura (´°C)
-20
-10
0
10
20
Delta de Humedad (%)

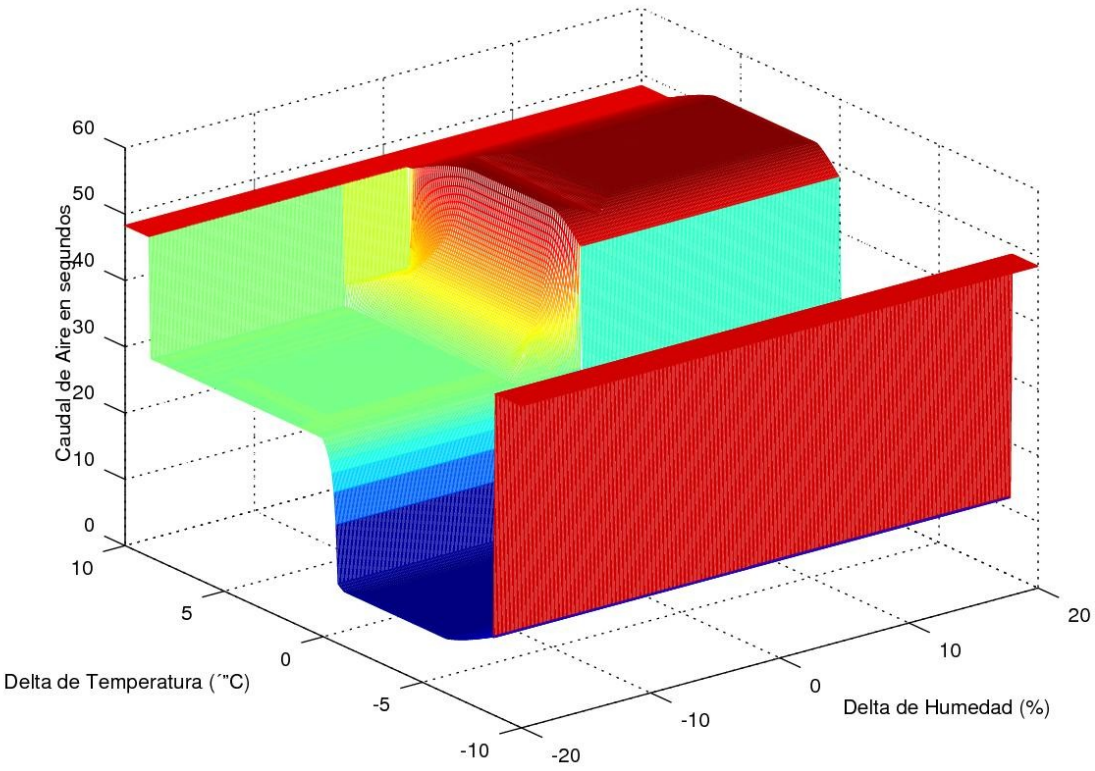
Intervalo 3


60
50
40
30
20
10
0
Caudal de Aire en segundos
10
5
0
-5
-10
Delta de Temperatura (ºC)
20
10
0
-10
-20
Delta de Humedad

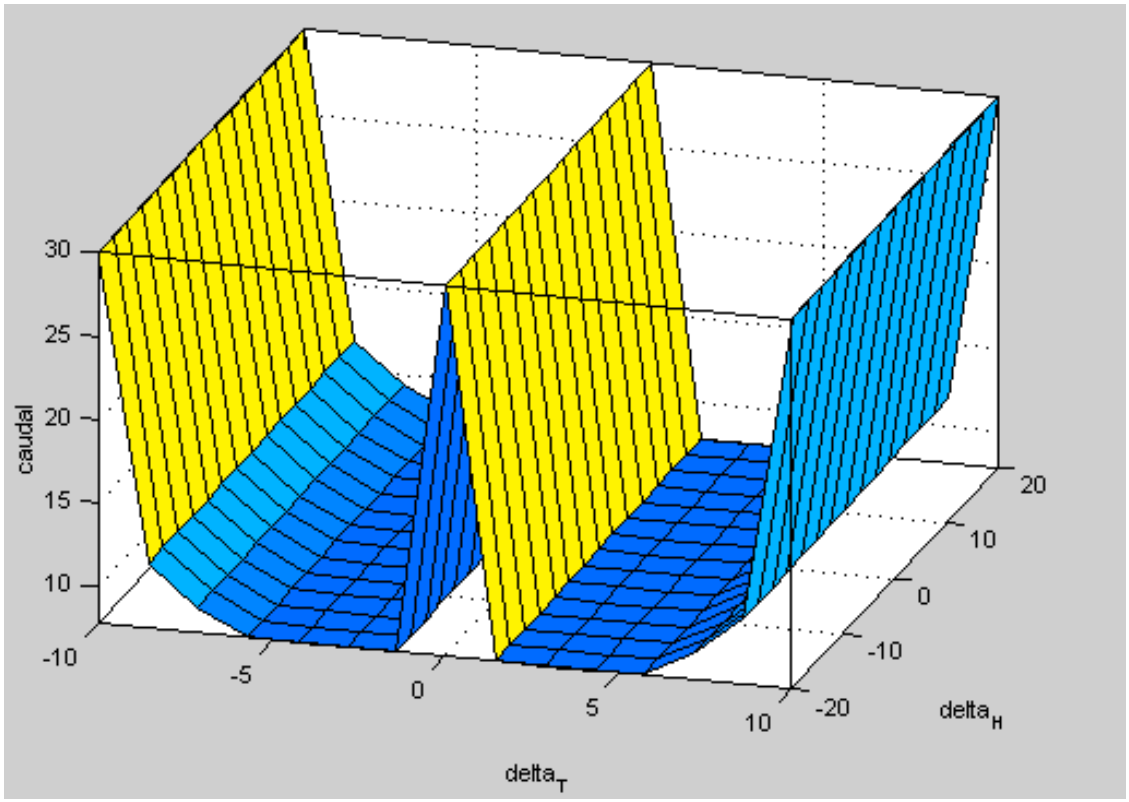
2.5.8. Comparación gráficas en 3D generadas por el modelo: En Matlab y en C (con Octave)

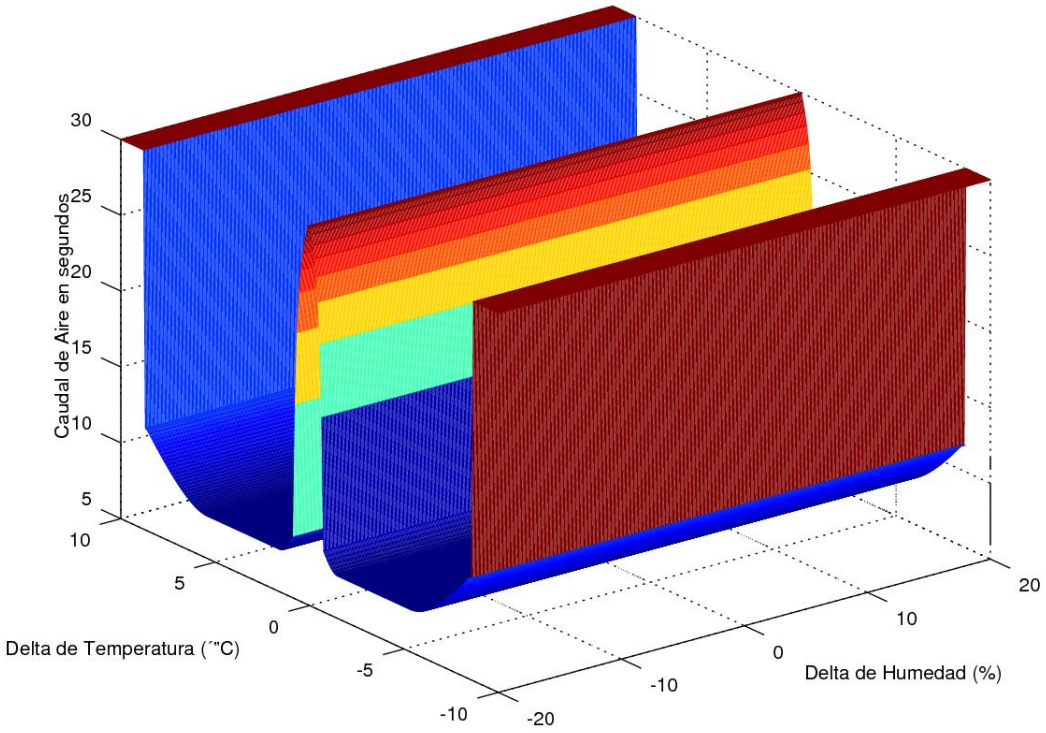
Intervalo 1



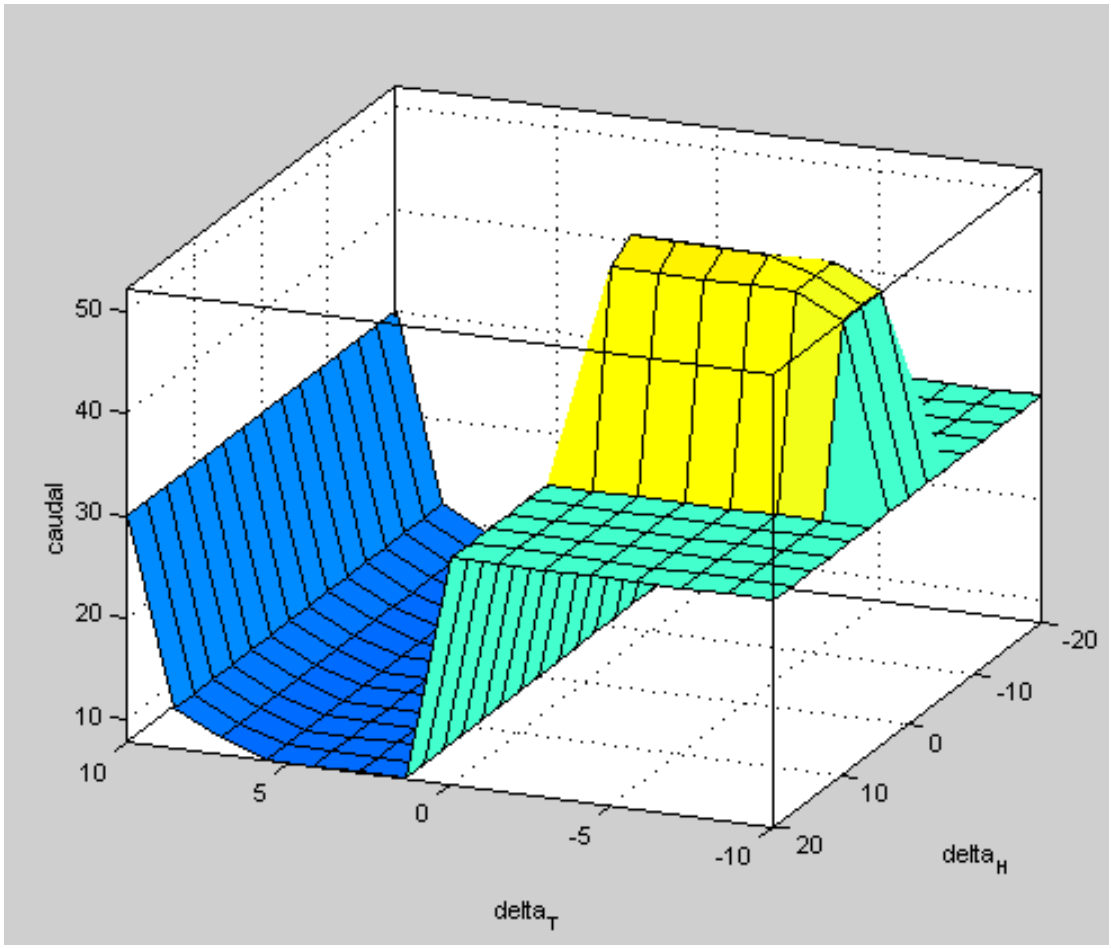


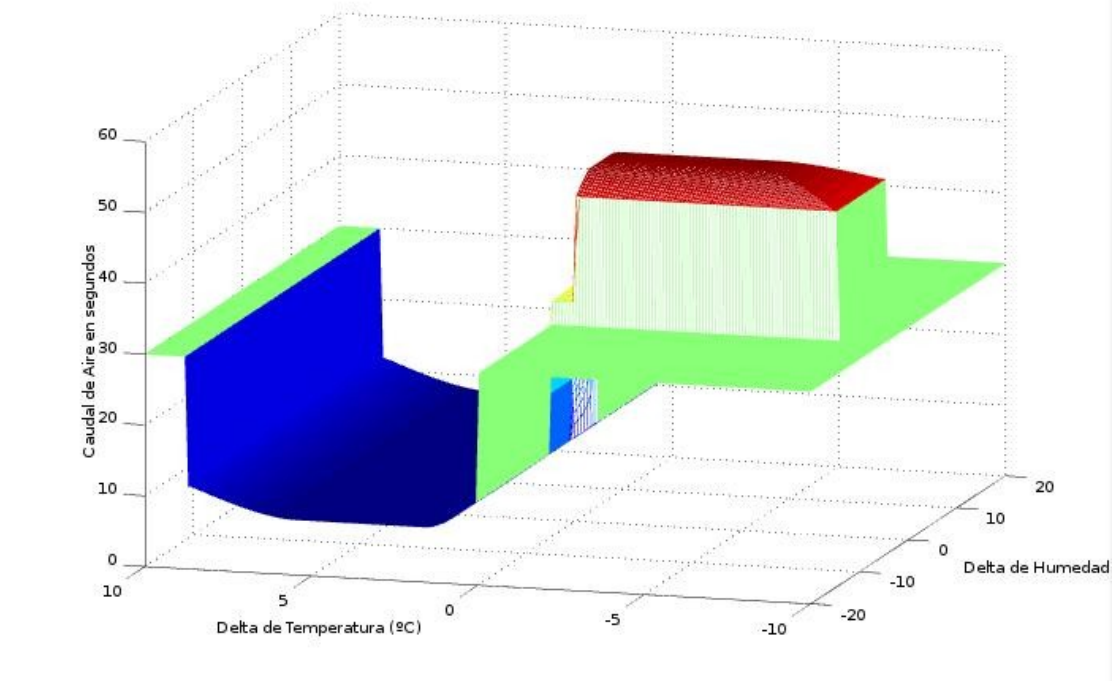
Intervalo 2


caudal
30
25
20
15
10
-10
-5
0
5
10
delta_T
20
10
0
-10
-20
delta_H


Caudal de Aire en segundos
30
25
20
15
10
5
10
5
0
-5
-10
Delta de Temperatura (°C)
-20
-10
0
10
20
Delta de Humedad (%)

Intervalo 3


caudal
50
40
30
20
10
10
5
0
-5
-10
delta_T
20
10
0
-10
-20
delta_H


Caudal de Aire en segundos
60
50
40
30
20
10
0
10
5
0
-5
-10
Delta de Temperatura (ºC)
20
10
0
-10
-20
Delta de Humedad

2.5.9. Conclusiones comparativa sistema de control fuzzy modelado

Tras realizar varias pruebas analíticas obteniendo valores aleatorios de nuestra variable de salida del sistema fuzzy(caudal de aire de la bomba) podemos decir que obtenemos valores muy parejos.

Pasos a seguir:

1. Obtenemos valores con Matlab
2. Obtenemos valores en C
3. Comparamos a nivel analítico ambos valores del modelo fuzzy
4. Comparamos a nivel gráfico ambos resultados del modelo fuzzy

Por tanto, tras esta fase de desarrollo a nivel teórico (analítico y gráfico) de nuestro modelo fuzzy, podemos afirmar que está listo para ser implementado en nuestro programa de Arduino.

2.6 Desarrollo del programa en Arduino para implementar el control climático en el invernadero fabricado

2.6.1. Antecedentes y especificaciones para el desarrollo del programa

Desarrollamos el programa teniendo el cuenta el código en C para implementar el Fuzzy generado en el apartado anterior.
Además intentamos utilizar en la medida de lo posible variables globales para que el código quede más limpio y comprensible para cualquier usuario que domine el entorno de programación en C.

Dicho esto citamos las siguientes especificaciones para la ejecución del programa:

El Timer 1 para interrupciones de 1 segundo y controlar el flujo del programa
El Timer 0 no se toca para no alterar las funciones de tiempo basicas como delay()
El Timer 2 lo usamos para el humidificador, generando una señal cuadrada de 107kHz mediante PWM

Los relés que gobiernan la celula Peltier conforman un Puente en H. Son de lógica invertida
El sensor de temperatura y humedad es un DHT22.
El LCD es uno 16x2 I2C en configuración serie.
La bomba de aire se ejecutará a intervalos calculados mediante algoritmo de control Fuzzy Logic
Regulamos el fotoperiodo mediante un LED de 6W.

1.1 Declaración de condiciones iniciales, flags, librerias y variables globales en general

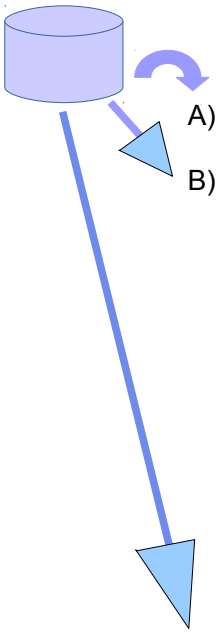
Ver código generado en Anexos.

2.6.2. Estructura y flujo general del programa

Flujo del programa

El flujo del programa desarrollado se ejecuta en paralelo mediante semáforos (también llamados flags o banderas) que bloquean o no la ejecución de cada hilo.

Básicamente tenemos los siguientes hilos:



A) El Menu del LCD

B) Modos de funcionamiento

1. Modo Grow
   a) Humidificador
   b) Peltier
   c) Fuzzy Bomba Aire
2. Modo Fructif
   a) Humidificador
   b) Peltier
   c) Fuzzy Bomba Aire
   d) Fotoperiodo
3. Modo Auto
   a) Modo Grow
   b) Modo Fructif

C) ISR (Rutina de Interrupción de Servicio)
Se realiza la gestión de los contadores para cada función de forma periódica a razón de 1 segundo

Estructura básica del programa

Mostramos las distribución general declarada dentro del programa:

```
/*-----( Cargamos las librerias necesarias )-----*/

//////////////////////////////////////////////////////////////////
// DECLARACION DE INCLUDES, DEFINES Y VARIABLES GLOBALES

//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
// CONTADORES de RETARDOS
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////
// CONDICIONES INICIALES
//////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCIONES
////////////////////////////////////////////////////////////////////////////////////////////////
/*************************************************************************************************/
/*************************************************************************************************/
// INICIO DE FUNCIONES Y DATOS FUZZY EMPLEADOS EN NUESTRO SISTEMA DE CONTROL
//      VARIABLE SOBRE LA QUE ACTUA: Caudal de aire de la Bomba Aireadora
/*************************************************************************************************/
/*************************************************************************************************/

////////////////////////////////////////////////////////////////////////////////////////////////
// FUNCION INTERVALOS
// Funcion que nos devuelve el valor calculado FUZZY del sistema de control.
// PARAMETROS DE ENTRADA: El intervalo en el que haremos el calculo
// FUZZY (Intervalos 1, 2 o 3) y el DELTA_T y el DELTA_H para ver la
// tendencia hacia arriba o hacia abajo de la temperatura y de la humedad
////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////
// FIN DE FUNCIONES Y DATOS FUZZY
////////////////////////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////////////////////////
// CONFIGURACION INICIAL DEL ARDUINO
////////////////////////////////////////////////////////////////////////////////////////////////
void setup()
{


}//fin setup


//////////////////////////////////////////////////////////////////////
//INTERRUPCIONES CADA SEGUNDO Y CONTADORES RETARDOS
//////////////////////////////////////////////////////////////////////
ISR(TIMER1)
{

////////////////////////////////////////////////////////////////////////////////////////////////
```

```
// INICIO  LOOP - MAIN
//////////////////////////////////////////////////////////////////////////////////////////////////
void loop()
{

//////////////////////////////////////////////////////////////////////////////////////////////////
// MENU DEL LCD
//////////////////////////////////////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////////////////////////////////////
// FIN DEL MENU DEL LCD
//////////////////////////////////////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////////////////////////////////////
// INICIO CALCULO DE MODOS DE FUNCIONAMIENTO
//////////////////////////////////////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////////////////////////////////////
// FIN de CALCULO DE MODOS DE FUNCIONAMIENTO
//////////////////////////////////////////////////////////////////////////////////////////////////


}//fin loop
```

2.6.3. Código completo del programa

A continuación se puede ver el código final desarrollado en lenguaje de programación C, para efectuar el control climático con lógica Fuzzy de nuestro invernadero.
Dicho control está desarrollado y optimizado para el correcto crecimiento de la seta de ostra, también llamada Pleurotus ostreatus.

Código del programa:

Ver código completo en Anexos

2.6.4. Descripción detallada de cada parte del programa

Principalmente tenemos 3 hilos que se ejecutan simultáneamente:

- El menú LCD
- Los modos de funcionamiento
    - Grow
    - Fructif
    - Auto
- ISR (control de tiempos sobre el humidificador, fuzzy logic, fotoperiodo y bomba aireadora)

1) Código dedicado a la ejecución del menú LCD

Esta parte del código se encuentra alojada en el loop (parte principal del programa) , se ejecuta cada 1s y se encarga básicamente de leer en que estado se encuentran las variables **valor_boton_CHG** (botón de “Cambio” en el display de la caja de control) y **valor_boton_MENU** (botón de “Menú” en el display de la caja de control). El “Menú” se emplea para desplazarse sobre las diferentes opciones del menú y el botón “Cambio” para activar o desactivar parametros dentro de cada estado del menú. De este modo tendremos acceso y control sobre todas y cada una de las acciones diseñadas para ejecutar el control climático deseado dentro del invernadero.

Ir al ***Documento nº3: Manual de uso del invernadero*** para ver ampliada toda la información de uso del Menu LCD.

Ver código generado para ello en Anexos

2) Código dedicado a la ejecución de los modos de funcionamiento

Esta parte del código también se ejecuta dentro del loop. Los modos se leen y comprueban cada 2s.

Recordemos que son:

- Modo Grow
- Modo Fructif
- Modo Auto

Dependiendo de que modo se encuentre activo se configurarán los flags de manera que actúen sobre:

- Modo Grow
    - Humidificador
    - Peltier
    - Fuzzy Bomba Aire
- Modo Fructif
    - Humidificador
    - Peltier
    - Fuzzy Bomba Aire
    - Fotoperiodo
- Modo Auto
    - Modo Grow
    - Modo Fructif

Ver código detallado generado para ello en Anexos

3) Código dedicado a la ejecución de interrupciones ISR

Las ISR se emplean para la gestión de los contadores para cada función de forma periódica a razón de 1 segundo.
Se ejecuta fuera del *loop* y del *setup* ya que es la encargada de gestionar cualquier tarea que interrumpa la rutina principal del programa.

Actúa sobre las siguientes variables:

- *RETARDO RELES*
- *TIEMPO EN ON BOMBA DE AIRE*
- *TIEMPO EN OFF BOMBA DE AIRE*
- *TIEMPO DEL HUMIDIFICADOR EN ON*
- *TIEMPO DEL HUMIDIFICADOR EN OFF*
- *TIEMPO MODO FUNCIONAMIENTO*
- *TIEMPO FOTOPERIODO ON*
- *TIEMPO FOTOPERIODO OFF*

Ver código detallado generado para las ISR en Anexos

2.6.5. Funciones y parámetros para la implementación del Fuzzy en el control del caudal de aire del invernadero

Nuestro Fuzzy calculado se ejecuta fuera del loop y del setup. Y se encarga de iniciar todos los datos y funciones Fuzzy empleados en nuestro sistema de control climático.
Básicamente es el código generado en el apartado anterior (*2.5 Modelado de nuestro sistema de control Fuzzy para el invernadero*) con algunos cambios para adaptarlo a nuestro codigo.

La variable sobre la que actúa es: el *caudal de aire* de la bomba aireadora.

Y en ella se encuentra la función *Intervalos*
Función que nos devuelve el valor calculado Fuzzy del sistema de control.

Parámetros de entrada:

- El intervalo en el que haremos el calculo *FUZZY* (1, 2 o 3)
- El *DELTA_T* y el *DELTA_H* (para evaluar la tendencia hacia arriba o hacia abajo de la temperatura y de la humedad)

Ver código detallado generado para el fuzzy en Anexos

2.6.6. Funciones para el control del resto de componentes

A continuación detallamos una serie de funciones que empleamos para controlar:

- La bomba de aire
- El fotoperiodo
- El humidificador
- Los reles
- La peltier
- Y otras funciones adicionales

Funciones para actuar sobre la bomba aireadora:

FUNCIÓN START_FUZZY_AIRE_BOMBA

Función que la usaremos para configurar los flags adecuadamente para que comience a funcionar la bomba de aire en el loop()
También activa el retardo correspondiente al ON

FUNCIÓN STOP_FUZZY_AIRE_BOMBA

Función que la usaremos para configurar los flags adecuadamente para que finalice la bomba de aire en el loop()
También activa el retardo correspondiente al OFF

FUNCIÓN BOMBA_AIREADORA

Función que se corresponde al hilo de ejecución de la bomba de aire dentro del loop()
Si el hilo esta habilitado la bomba funciona, sino la bomba esta desconectada

Funciones para actuar sobre el fotoperiodo:

FUNCIÓN START_FOTOPERIODO

Función que la usaremos para configurar los flags adecuadamente para que comience a funcionar el FOTOPERIODO en el loop()
También activa el retardo correspondiente al ON

FUNCIÓN STOP_FOTOPERIODO

Función que la usaremos para configurar los flags adecuadamente para que finalice el FOTOPERIODO en el loop()
También activa el retardo correspondiente al OFF

FUNCIÓN FOTOPERIODO

Función que se corresponde al hilo de ejecución del fotoperiodo dentro del loop()

Funciones para actuar sobre el humidificador:

FUNCIÓN START_HUMIDIFICADOR

Función que la usaremos para configurar los flags adecuadamente para que comience a funcionar el HUMIDIFICADOR en el loop()
También activa el retardo correspondiente al ON

FUNCIÓN STOP_HUMIDIFICADOR

Función que la usaremos para configurar los flags adecuadamente para que finalice el HUMIDIFICADOR en el loop()
También activa el retardo correspondiente al OFF

FUNCIÓN HUMIDIFICADOR

Funcion que se corresponde al hilo de ejecución del humidificador dentro del loop()
Activará el humidificador en el estado que indiquen los flags

Funciones para actuar sobre los relés:

FUNCIÓN DESCONECTO_RELES

Función que sirve para apagar los relés directamente en el puente en H.

Funciones para actuar sobre la peltier:

FUNCIÓN ACTIVO_FLAG_PELTIER_FRIO

Función para indicar a la función peltier que podemos enfriar

FUNCIÓN ACTIVO_FLAG_PELTIER_CALOR

Función para indicar a la función peltier que podemos calentar

FUNCIÓN DESACTIVO_FLAG_PELTIER

Función que sirve para indicar a la función peltier que desconecte la peltier.
Además activa un pequeño retardo para que la peltier se recupere antes de volver a conectarla

FUNCIÓN PELTIER

Función que empleamos para poner a la peltier en frio, en calor o desconectada, siempre que el retardo de la función desconexión haya finalizado.
MUY IMPORTANTE: No hay que activar nunca el RELE_A y el RELE_B a la vez, porque hacemos un corto en la fuente de 12V.
Siempre asegurarse del orden de conexión o desconectar todo primero. Primero apagar un relé y luego activar el otro.

Activará la peltier en el estado que indiquen los flags

Funciones adicionales:

FUNCIÓN COMIENZA RETARDO

Función que la usaremos para pasar de forma genérica un valor de retardo a una variable contador

FUNCIÓN PINTAR CONTADOR

Función para mostrar en el LCD el valor restante de horas del modo que este iniciado

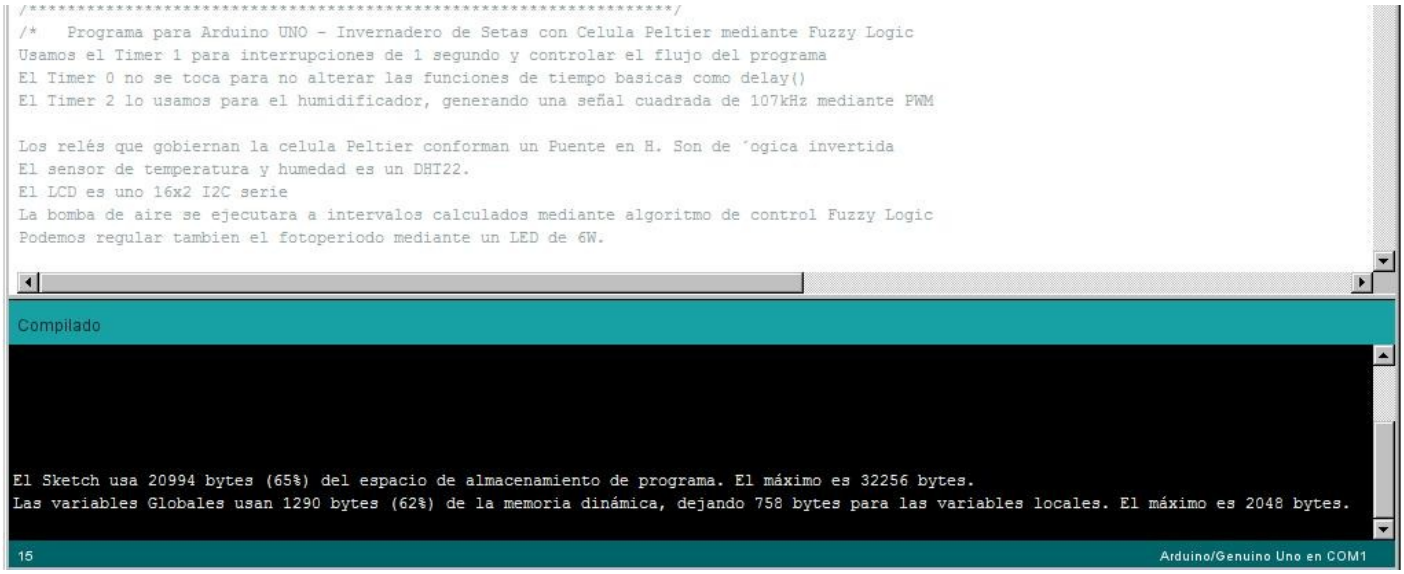
FUNCIÓN PINTAR ESPACIO VACÍO CONTADOR

Función que la usaremos para borrar del LCD el valor del contador de horas del modo que estuviese activo

Ver código empleado para las funciones en Anexos

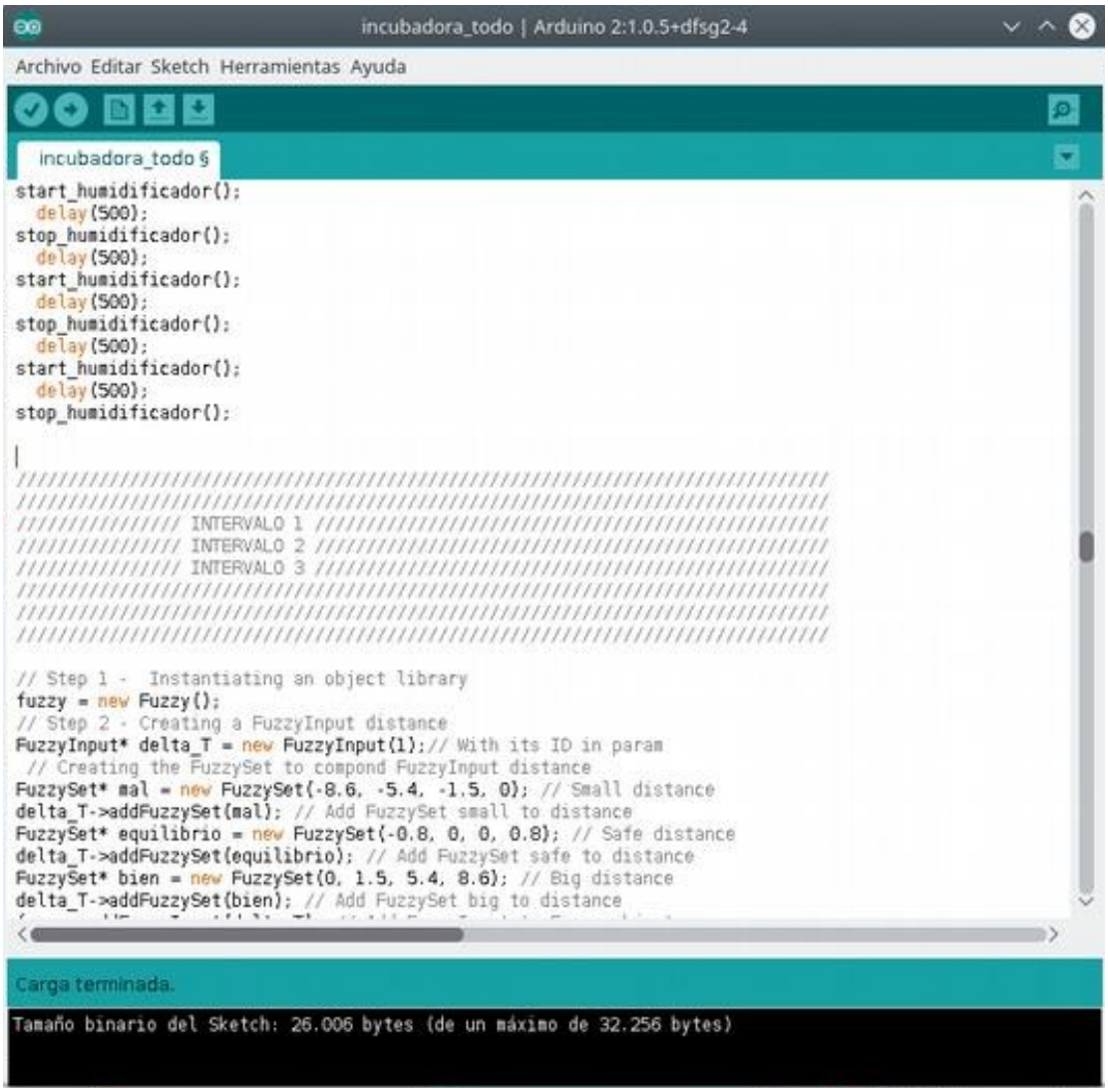
2.6.7. Problemas y resultados experimentados en el desarrollo del programa

Como se puede apreciar en la fotografía que mostramos a continuación el sketch del programa compilado usa un espacio en memoria de 20994 bytes.
Lo que supone un 65% del espacio de almacenamiento máximo del programa (32256 bytes)
Y las variables globales usan 1290 bytes, el 62% de la memoria dinámica. Dejando 758 bytes libres para las variables locales
El espacio máximo asignado para variables es de 2048 bytes.



No obstante al principio el programa ocupaba un espacio mayor del actual, y hubo que depurarlo en sucesivas ocasiones ya que comprobamos experimentalmente que el Arduino se quedaba colgado al estar próximos al máximo de espacio en memoria permitido para asegurarnos una correcta ejecución del código.

Aquí mostramos la ocupación en memoria de versiones iniciales del programa:



2.7 Realización del cultivo experimental de setas de ostra (*Pleurotus ostreatus*) en el invernadero

2.7.1. Antecedentes al cultivo

Siendo conscientes de que el cultivo se setas difiere mucho de la electrónica a la que estamos acostumbrados. Nos informamos en diversas fuentes y medios, averiguando los medios adecuados para que todo salga bien.

- Temperatura y humedad optimas
- Ventilación del cultivo
- Grado de exposición a la luz
- Periodos de incubación
- Enfermedades que puede afectar a su crecimiento
- Sustratos adecuados para su crecimiento

Y enseguida nos damos cuenta de que sera una tarea ardua y compleja.

Pero no por ello vamos a desistir, así que nos ponemos manos a la obra.

2.7.2. Material necesario para el cultivo

Aquí dejamos una lista de los materiales que se recomiendan tener para realizar el cultivo.

Aunque no todos ellos llegamos a emplearlos ni a necesitarlos, ya que algunos de ellos tan solo se emplean en cultivos a nivel industrial.

Para desinfectar la sala y los utensilios con los que se hará el cultivo:

- alcohol sanitario
- agua oxigenada (H^2O^2)
- bisturí y hojas
- mechero o soplete
- jeringuilla de plástico
- termómetro de cocina
- caja de guantes

Para propagar el micelio

- placas de petri
- agar-agar
- dextrosa
- extracto de malta
- levadura de cerveza en copos
- harina de avena
- una caja para incubar el micelio

Para preparar el inoculo:

- tarros de vidrio
- grano de centeno
- o grano de mijo
- o grano de trigo
- termómetro
- un lugar donde dejar los recipientes del cultivo

Para el substrato en masa:

- bolsas de plástico
- contenedor o
- malla para escurrir la paja
- alfalfa
- termómetro
- un rincón oscuro para incubar la paja

Para fructificar:

- un lugar con oscuridad
- una luminaria
- ventilador
- humidificador
- termometro
- higrometro

2.7.3. Cultivos experimentales

En este apartado optamos por dejar un documento gráfico de todo el proceso. Su evolución, problemas, y observaciones tomadas al respecto.

1. Cultivos fuera del invernadero

Realizamos 2 ensayos previos de cultivo de seta de ostra, para familiarizarnos con el proceso.

El primero sobre sustrato principalmente formado por cereal de cebada, centeno y triticale.

El segundo sobre setas ya maduras de la misma especie que pretendemos cultivar.

1.1 Sobre cereal

	
Micelio extendido por los tarros de cristal	Se aprecia ese tono blanco característico

1.2 Sobre seta madura

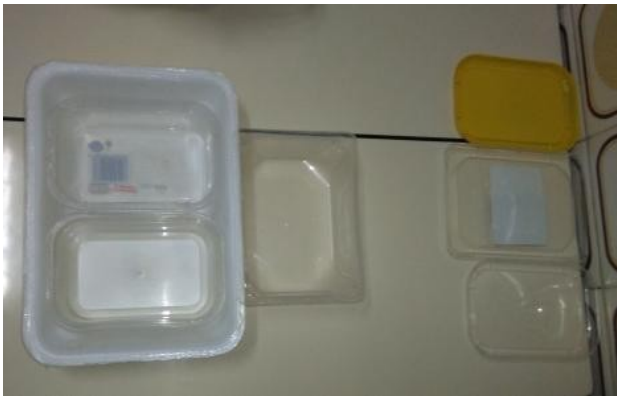
	
1ª Fase: inoculación del micelio	
	
Micelio extendido por la seta	
	
Fase de fructificación	

2. Cultivos dentro del invernadero

Una vez realizado el montaje del invernadero y su puesta en marcha en vacío para comprobar que todos los componentes hagan correctamente la función para la que han sido programados, solo queda realizar un cultivo experimental para estudiar los resultados del trabajo.

Por tanto realizamos varios cultivos sobre diferentes medios de sustrato (en este caso cereal y agar) y aquí os mostramos la prueba gráfica.

2.1 Sobre cereal

<table>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Compramos setas de ostra</td><td>Material para el sustrato: De izq. a derecha, cereal, cemento blanco y arena</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Preparamos los recipientes</td><td>Mezclamos arena y cemento blanco para controlar el PH del cultivo</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Metemos arena y agua en el fondo para mantener la humedad</td><td>Cocemos el grano</td></tr>
</table>

	
Mezclamos bien...	Y cocemos hasta que el grano está blando
	
Una vez enfriadas las muestras..	Comenzamos con la inoculación
	
Cogemos las esporas con pinzas	
	
Cubrimos con algodón todos los recipientes	Cultivo listo

	
Resultado final	Cereal con humedad en su interior
	
Y lo introducimos en el invernadero	Poniendo en marcha el modo Grow
	
Observamos acumulación de humedad en el metacrilato de la ventana	Vista del interior en el modo de funcionamiento “Fructif”
	
Comienza a extenderse el micelio...	Vista de cerca

	
Muestras necrosadas contaminadas por otros hongos ajenos a nuestro cultivo	Se observa el cambio de color
	
Examinamos los recipientes...	Estudiamos la muestra contaminada
	
Y la desechamos	Muestra reseca por exceso de caudal de aire
	
Aparición de moho verde	El moho verde (Penicillium digitatum) tiene un desarrollo fácil a 20 ºC y humedad relativa alta

	
Segunda prueba de cultivo	
	
Empezamos a mejorar el aislamiento del habitáculo con cinta de aluminio	
	
Se puede apreciar como hierve el agua...	Modificamos los parámetros de T y H
	
Comienzo de propagación del micelio	Vista de cerca

	
3er Cultivo	
	
Tarro contaminado de micelio	Pequeña muestra de setas fructificadas

2.2 Sobre agar

	
Preparamos 4 muestras en agar	Contaminamos con trozos de seta
	
	Comienza a extenderse el micelio
	
El cultivo va bien en todas las muestras	
	
Cultivo dentro de la caja	Se extiende rápidamente

	
Y pasadas unas semanas...	Empieza a fructificar
	
Blanco total	Muestra contaminada
	
Aunque debido a la poca cantidad de agar	
	
Apenas salen...	Unos pequeños primordios

2.7.4. Conclusiones

Tras realizar al menos 6 cultivos diferentes, con diferentes medios y sustratos nos damos cuenta que resulta más complejo de lo que parece en la teoría hacer crecer en condiciones las setas de ostra. Ya que tenemos que ser muy meticulosos con las condiciones de higiene y desinfección de los instrumentos que empleamos para llevar a cabo el cultivo. Además debemos mantener esas condiciones durante todo el proceso.

En nuestro caso tenemos éxito haciendo crecer el micelio sobre grano, sobre agar e incluso sobre setas en proceso de descomposición; pero no así cuando entramos en la etapa de fructificación dentro del invernadero.

Esto es debido a diversos motivos:

- La falta de potencia de la peltier hace que no sea posible bajar la temperatura del habitáculo al rango de crecimiento necesario.
- Hay que realizar una buena planificación estacional para poder operar dentro de esos márgenes teniendo en cuenta la temperatura externa en la que nos movemos.
- Analizando la respuesta de la peltier (y por tanto la eficiencia térmica de nuestro habitáculo) nos damos cuenta como tan solo podemos operar con un rango de ΔT con respecto al interior de apenas 6ºC y un ΔT_{MAX} con respecto a la temperatura externa de apenas 3-4ºC, como veremos en el análisis de los datos de cultivo del apartado siguiente.

Observamos indicios de *contaminación* de hongos y bacterias en diversas ocasiones lo que nos obliga a desechar muestras, y mejorar las condiciones de higiene. Los focos principales de contaminación proceden de:

- Las juntas de puerta y ventana de metacrilato
- Bisagra de la puerta
- Tubo inyector de aire
- Junta de encajonamiento de la peltier

Necesitamos ajustar en diversas ocasiones los rangos de temperatura y humedad para mejorar la respuesta de nuestro sistema, en función de como vemos los resultados del cultivo.

En cuanto a la humedad, observamos que se acumula un exceso de agua en el interior de la caja.
Al final concluimos que en el modo *grow* deberíamos humidificar menos (apenas es necesario hacerlo) si la humedad externa es favorable. Y en el modo *fructif* debido a los ciclos de frío (los cuales hacen operar cerca del punto de rocío del agua) se precipita más agua de la deseada.

Para el ajuste de *temperatura* y *humedad* nos habría venido bien instalar un sensor externo para regular sus rango en función de la temperatura externa.

En cuanto al *caudal de aire* también hemos visto que deberíamos reducirlo para que el sustrato no se seque demasiado. Y ademas es importante direccionar dicho caudal de forma que no incida de lleno sobre el cultivo (esto le perjudica). Sería interesante buscar otro método, quizás como hacen las neveras.

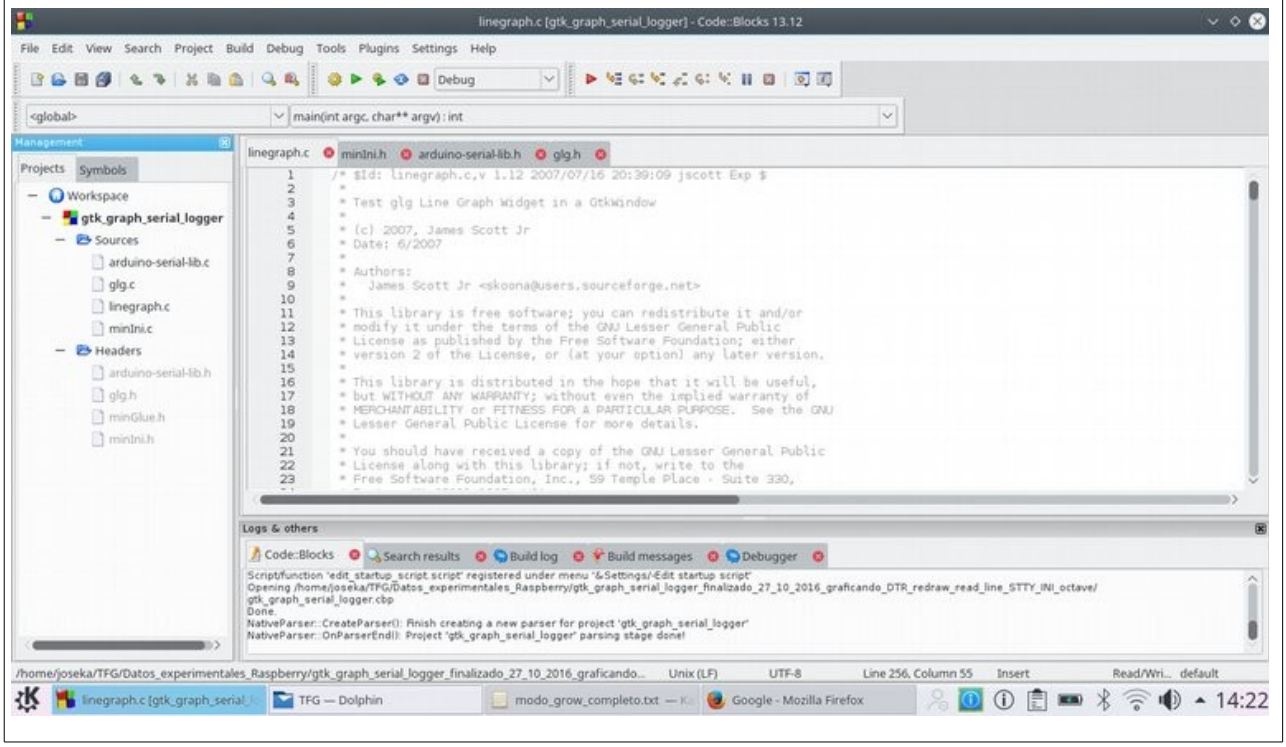
En próximos proyectos la idea será completar la fase final del cultivo con el sustrato de paja y aserrín (como se recomienda en diversos estudios) para que salgan con cuerpo las setas y sacar una cosecha en condiciones.
En el cereal crece el micelio y fructifica, pero las setas salen pequeñas y sin cuerpo. Nos informaremos de cuales son los mejores sustratos para cosechar las setas con el mayor tamaño posible.

Tenemos muchos factores a mejorar en el proceso, pero gracias al registro de parámetros que hicimos posteriormente (como veremos en el apartado siguiente) seremos capaces de mejorar el comportamiento del control climático.

2.8 Registro con Raspberry o PC externo de los datos experimentales de cultivos de seta realizados y el posterior tratamiento con Octave para su estudio teórico

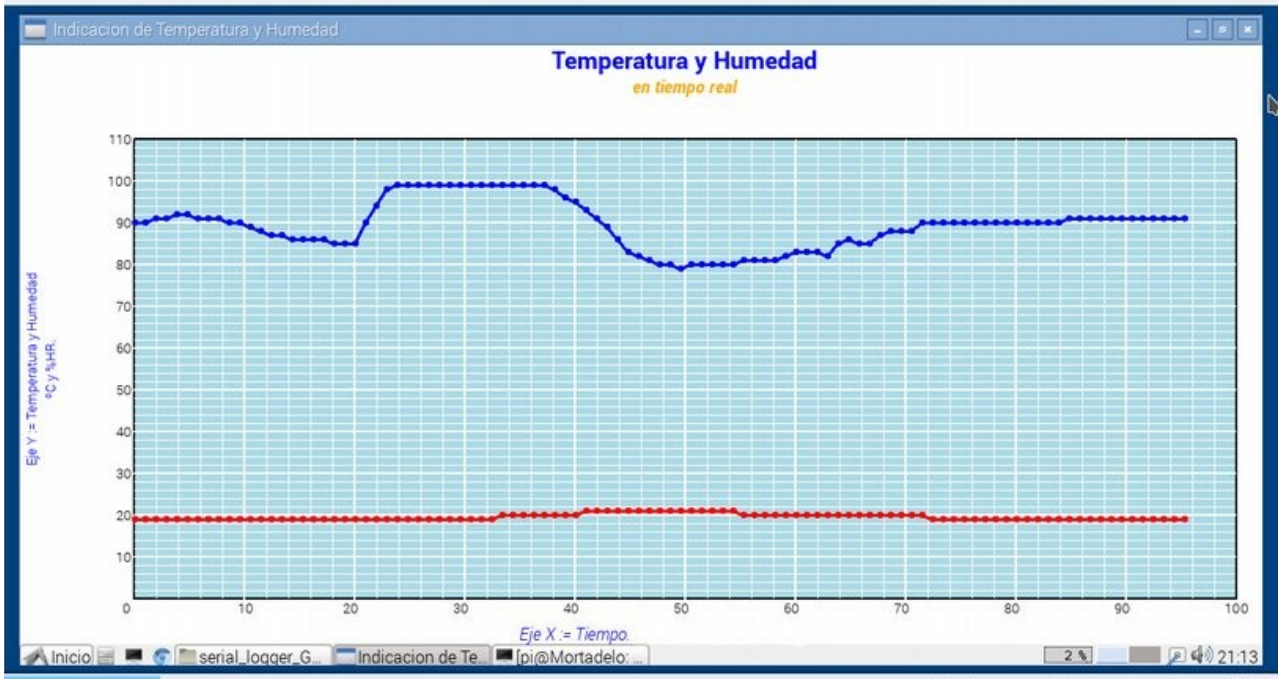
2.8.1. Captura de datos experimentales con dispositivo Raspberry y visualización de estos en tiempo real

Realizamos la captura y visualización en tiempo real de los datos de Temperatura y Humedad del interior del invernadero gracias a un programa que conseguimos diseñado por un amigo en Code:Blocks

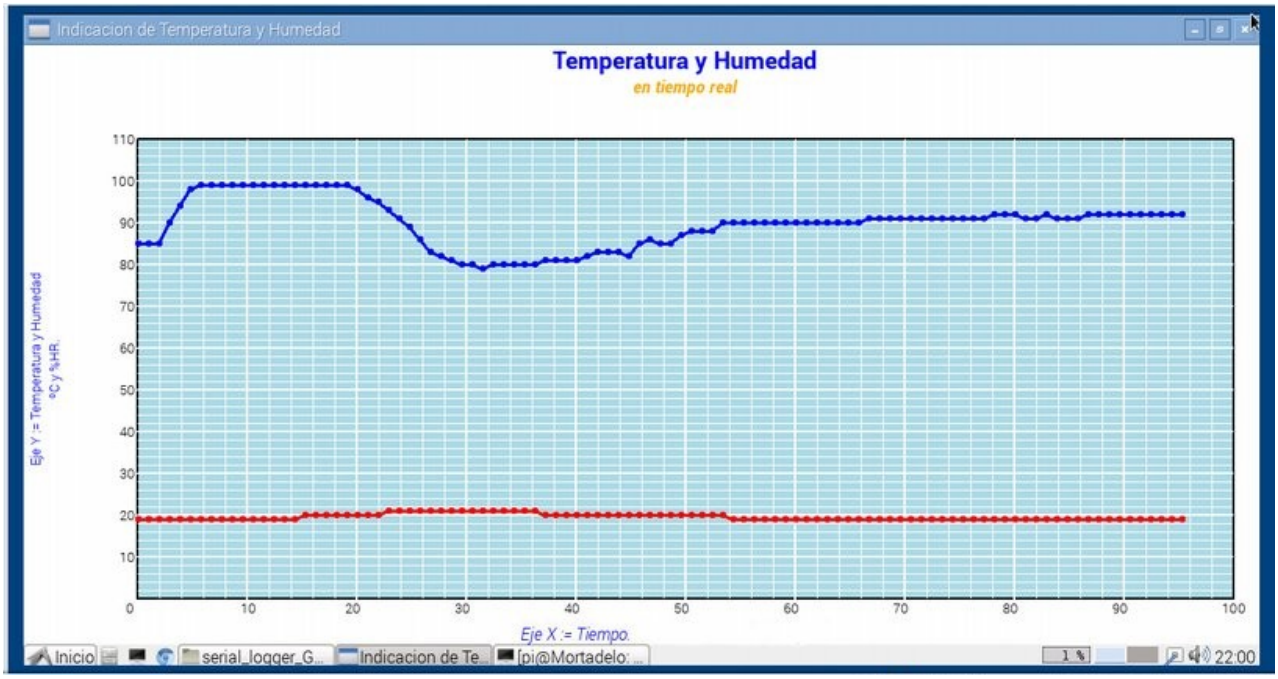


Realizamos varias pruebas de cultivo a lo largo del desarrollo del proyecto. Y por tanto hay una gran cantidad de datos registrados para analizar a posteriori. Por ello solo nos centraremos en puntos claves como el inicio y el fin del régimen transitorio; así como del comienzo y estabilización del régimen permanente de los modos de funcionamiento Grow y Fructif.

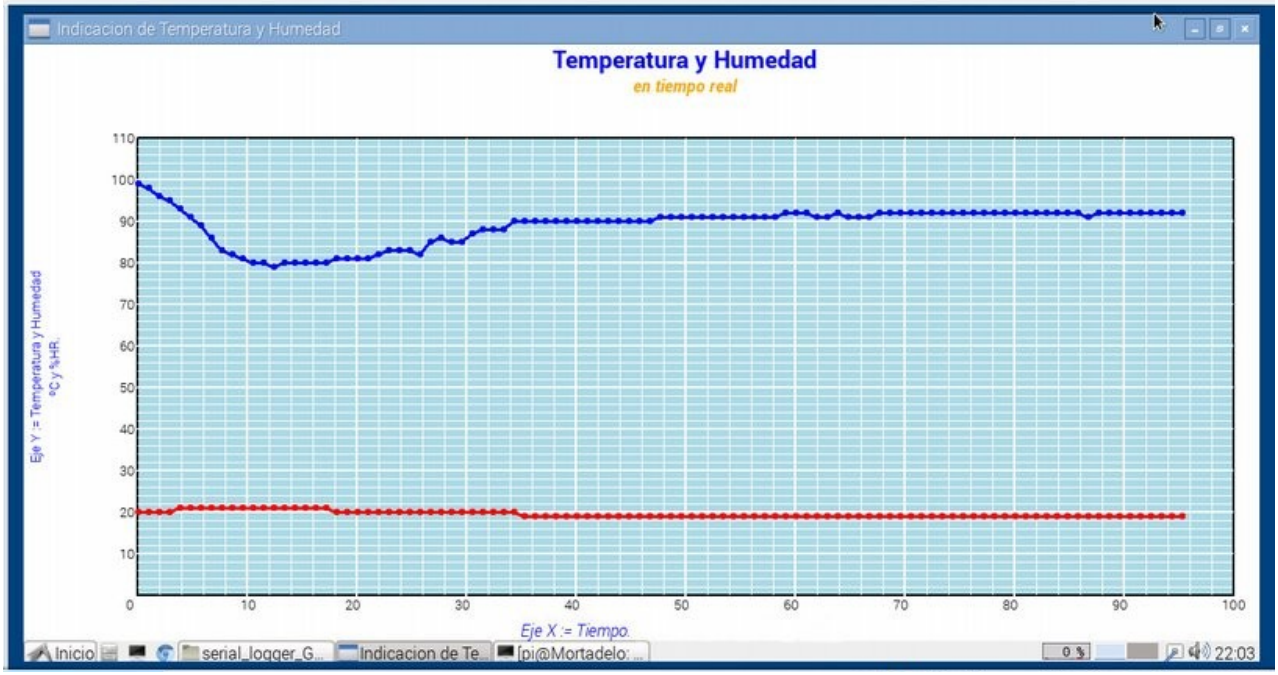
Capturas de pantalla en tiempo real del comportamiento de la T y la H



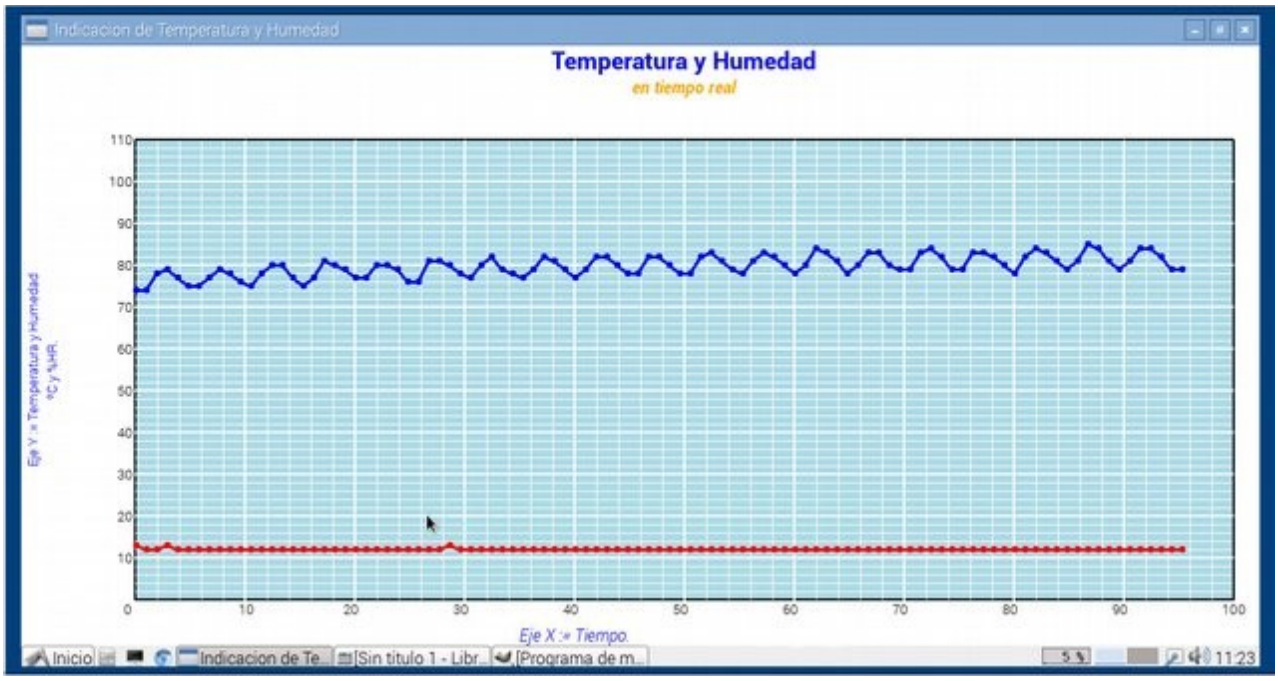
Inicio del régimen transitorio del modo Grow, en el que se puede apreciar el efecto de tener que calentar desde la temperatura exterior al rango deseado dentro del invernadero.



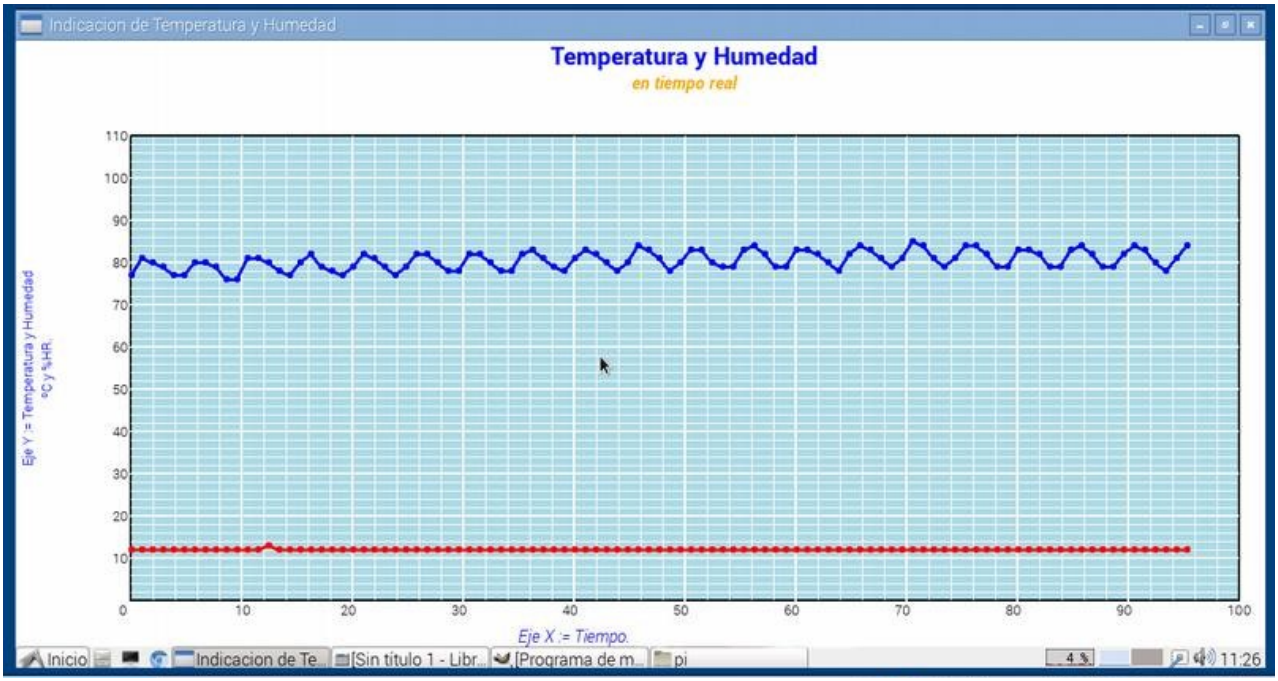
Fin del transitorio del modo Grow



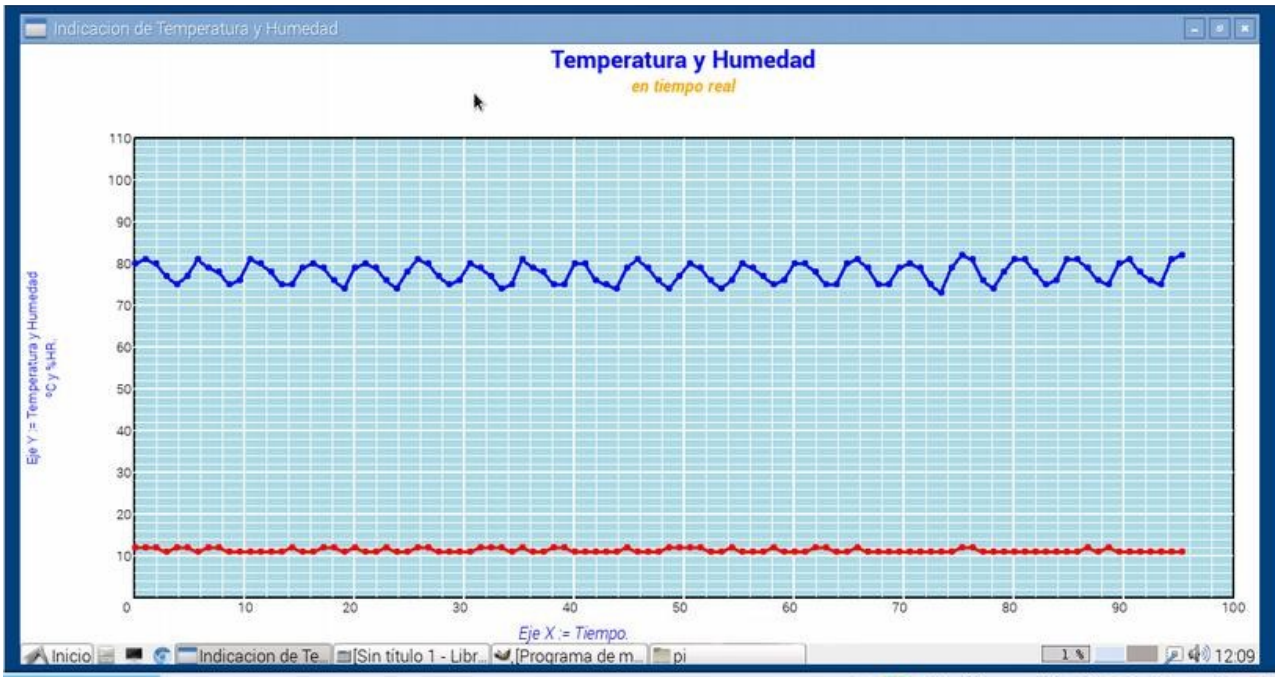
Inicio del régimen permanente del modo Grow



Inicio del régimen transitorio del modo Fructif



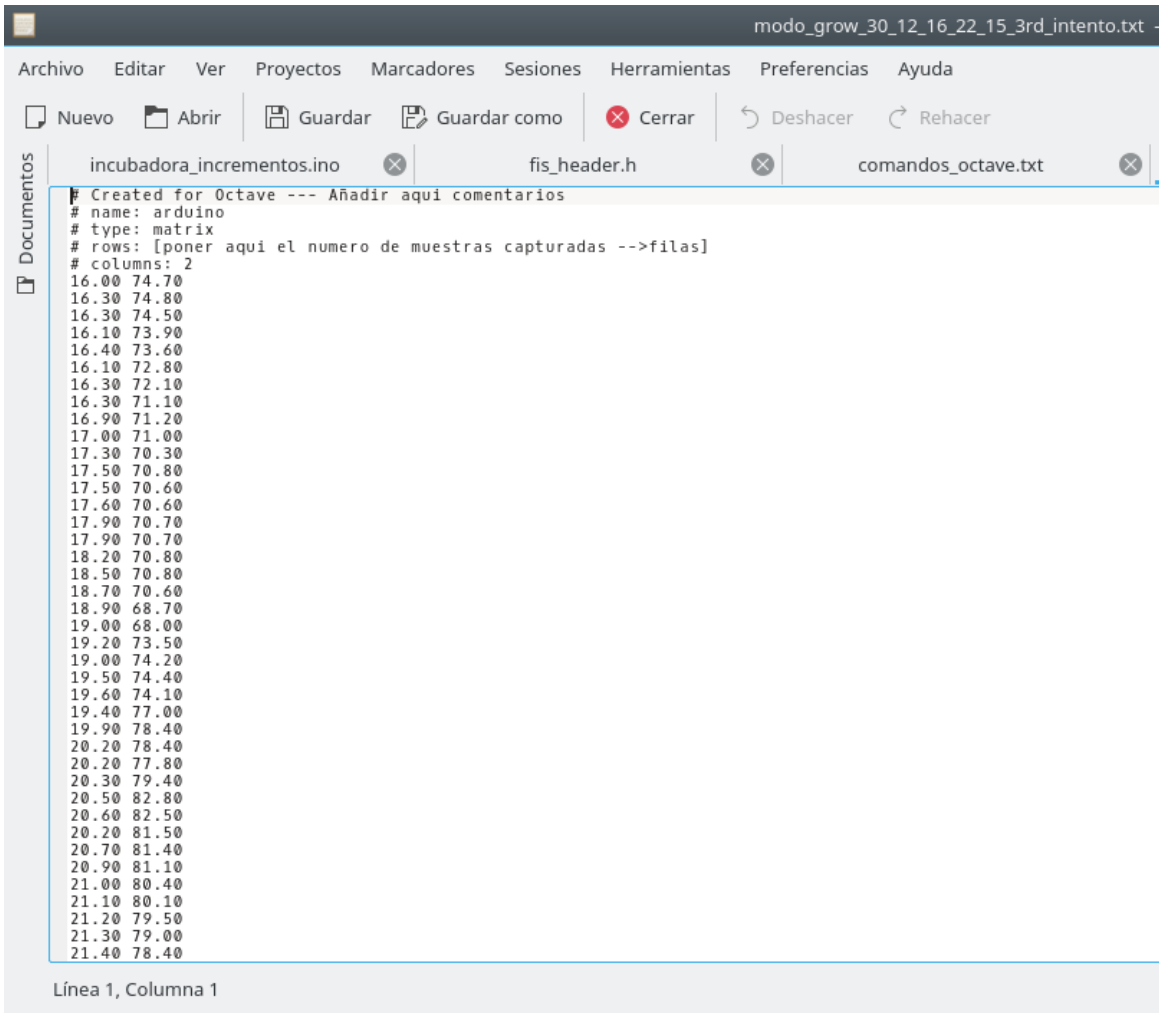
Fin del régimen transitorio del modo Fructif



Régimen permanente del modo Fructif

2.8.2. Almacenamiento de datos modo Grow en formato .txt

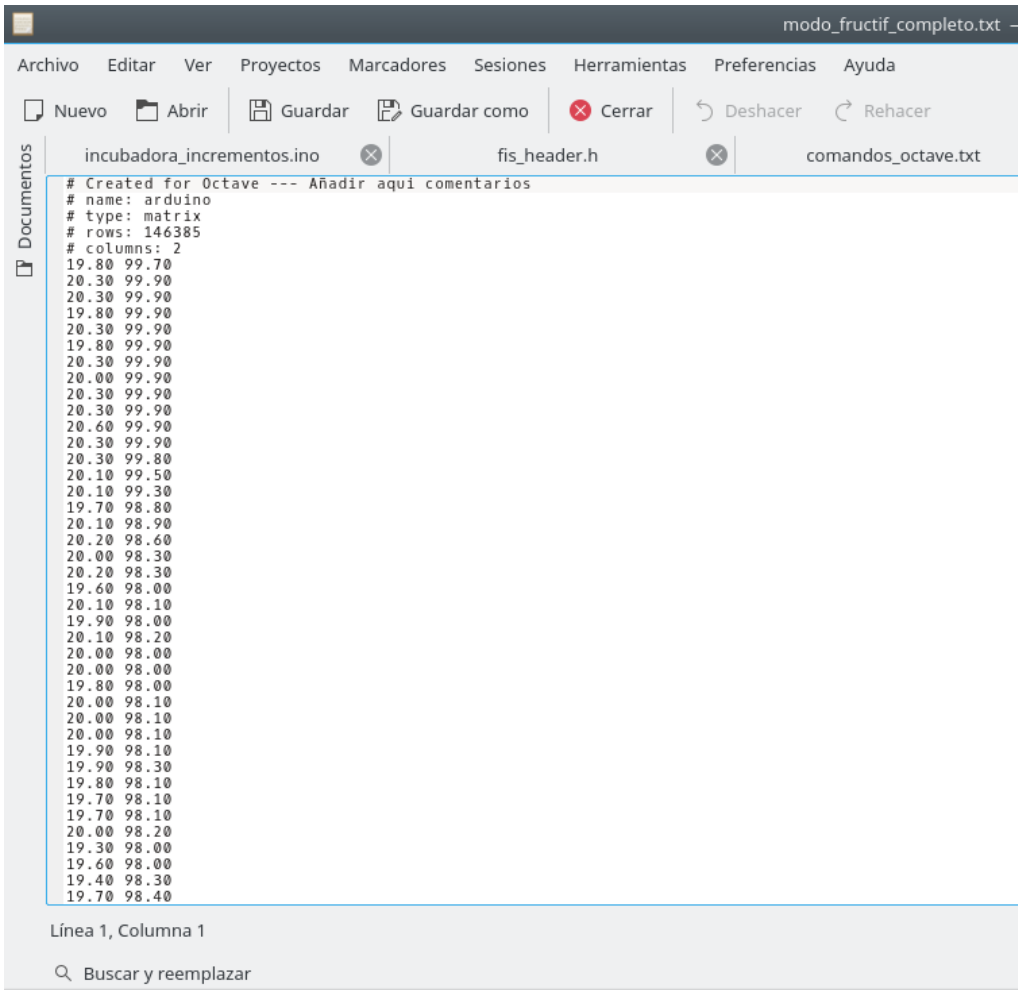
Almacenamos los datos del comportamiento en archivos .txt para su posterior análisis



Datos evolución T y H

2.8.3. Almacenamiento de datos modo Fructif en formato .txt

Almacenamos los datos del comportamiento en archivos .txt para su posterior análisis

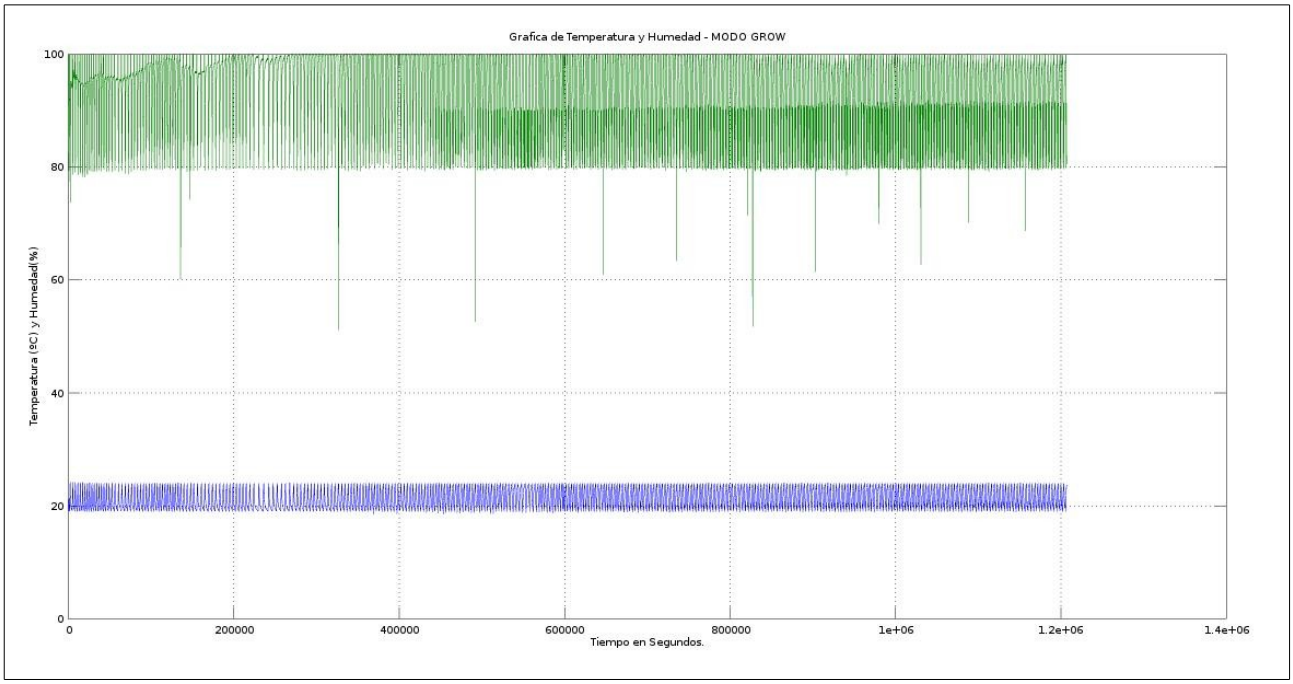


Datos evolución T y H

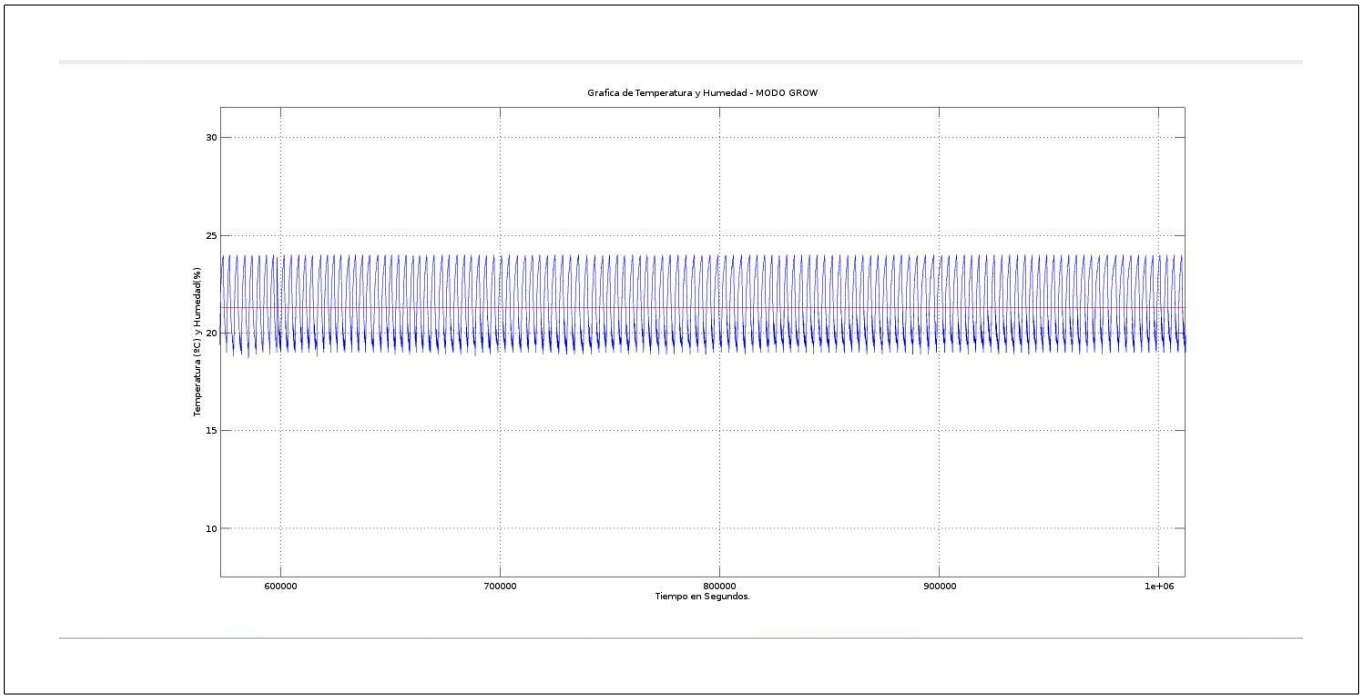
2.8.4.Generación y análisis de gráficas modo Grow

Para el análisis de los datos empleamos el Octave de nuevo (**ver código empleado en Anexos**). De este modo obtenemos las gráficas que mostramos a continuación y que nos permiten hacernos una idea más exacta a nivel visual de cual es el comportamiento real de nuestro invernadero.

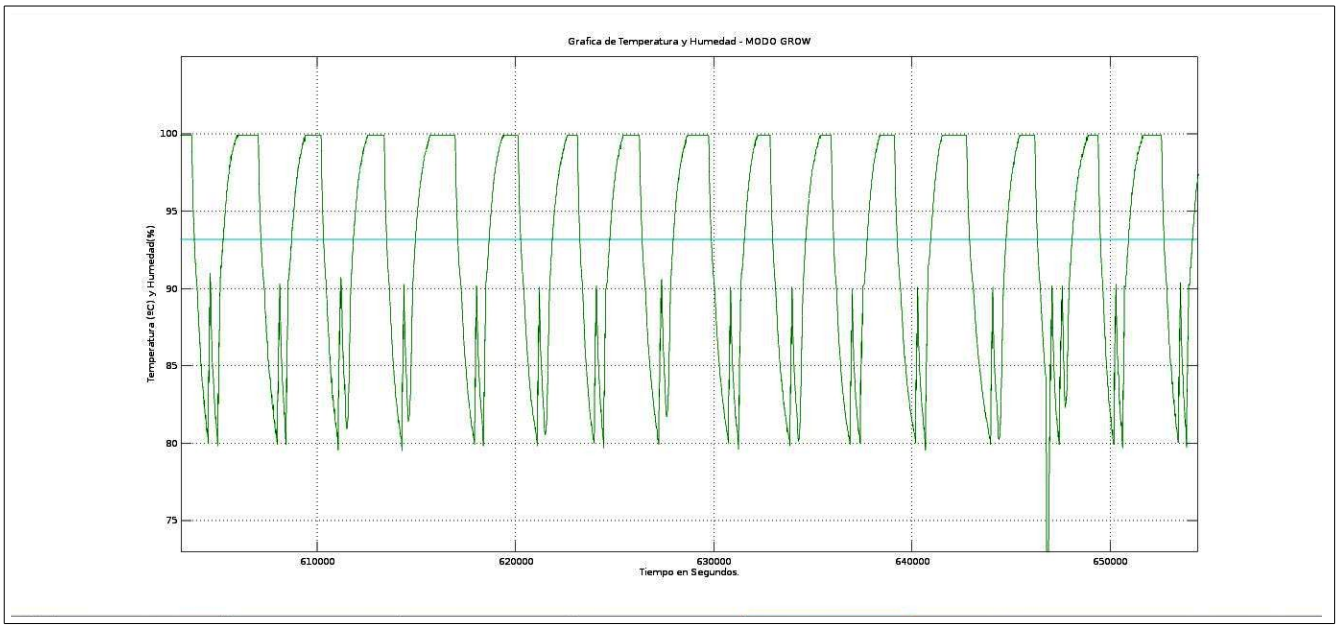
4.1 Modo Grow periodo completo



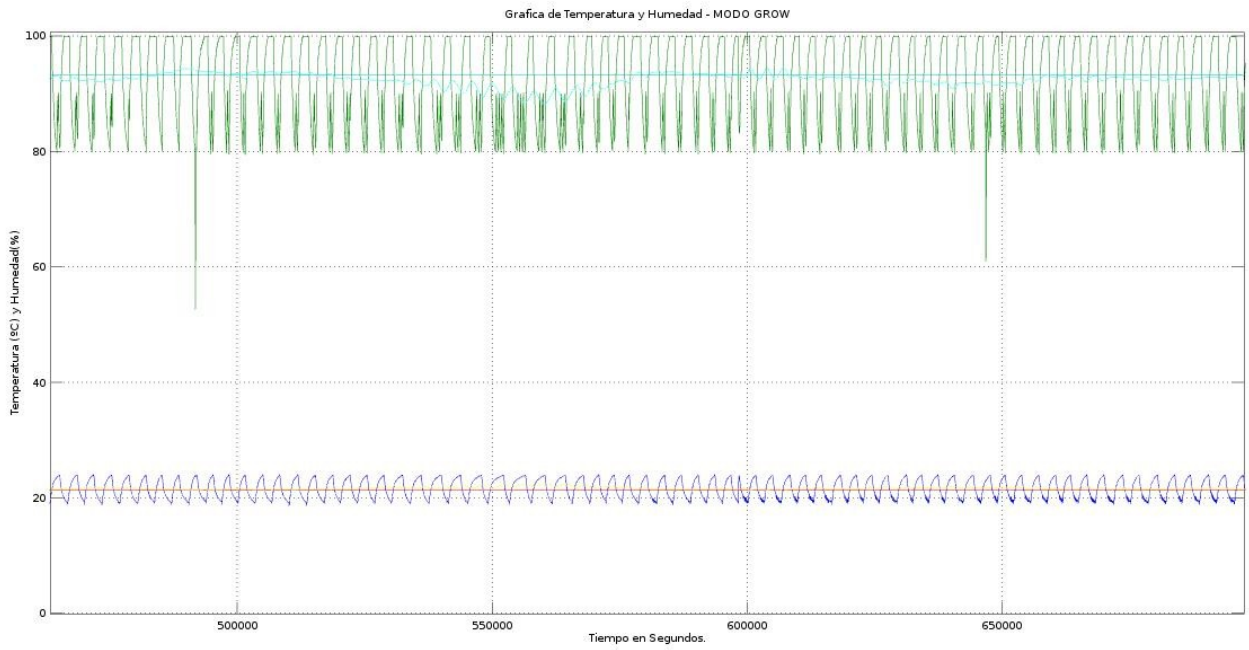
4.2 Modo grow con zoom (escala tiempo) de la temperatura en el rango intermedio del periodo, y su valor medio



4.3 Modo Grow con zoom (escala tiempo) de la humedad en el rango intermedio del periodo, y su valor medio

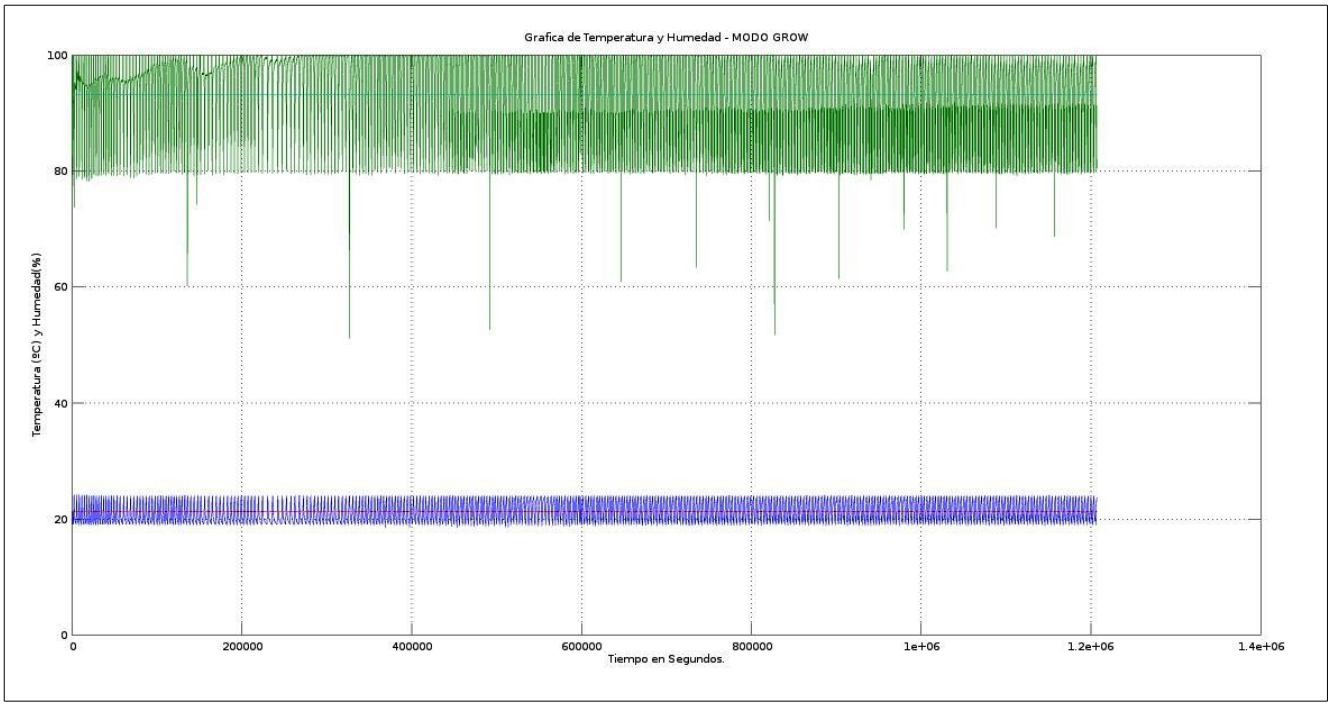


4.4 Modo Grow con zoom (escala tiempo) de la humedad y temperatura en el rango intermedio del periodo, con valores medios y filtro de tipo FIR

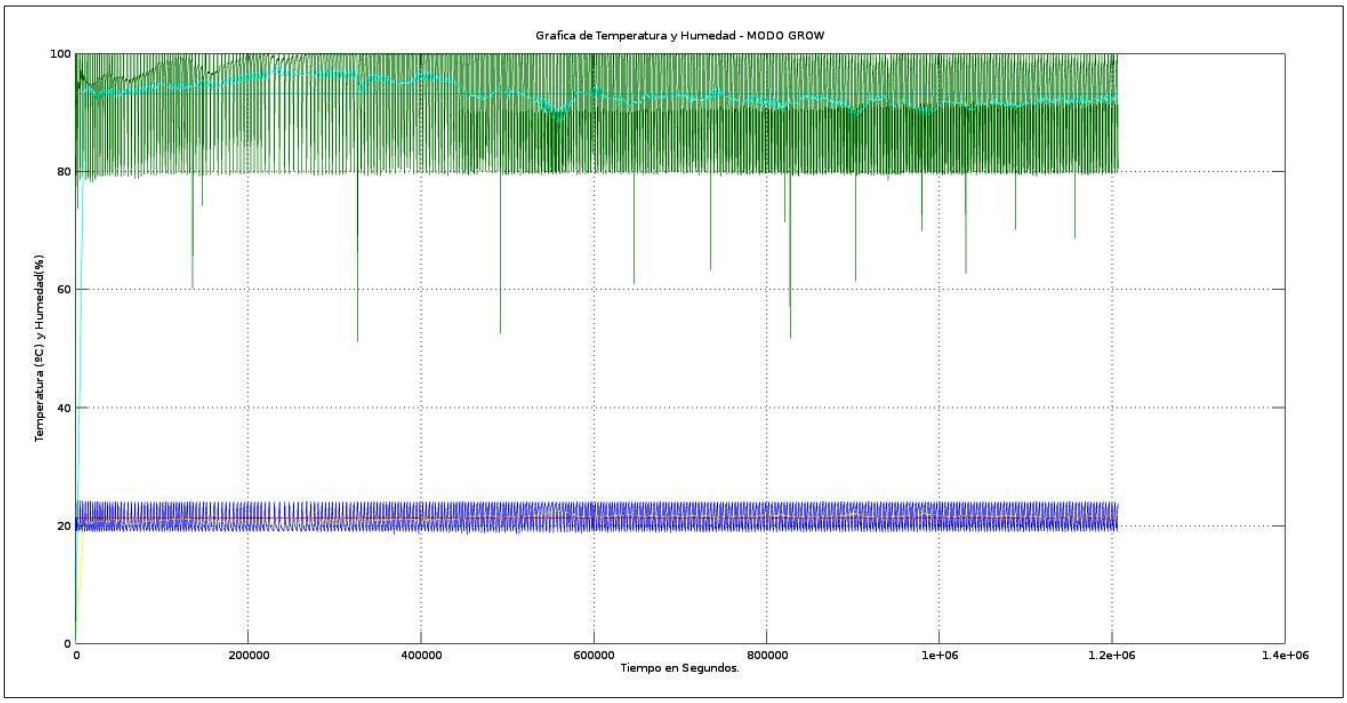


El filtro de tipo FIR nos da una muestra más real de la evolución media de ambas variables de estudio

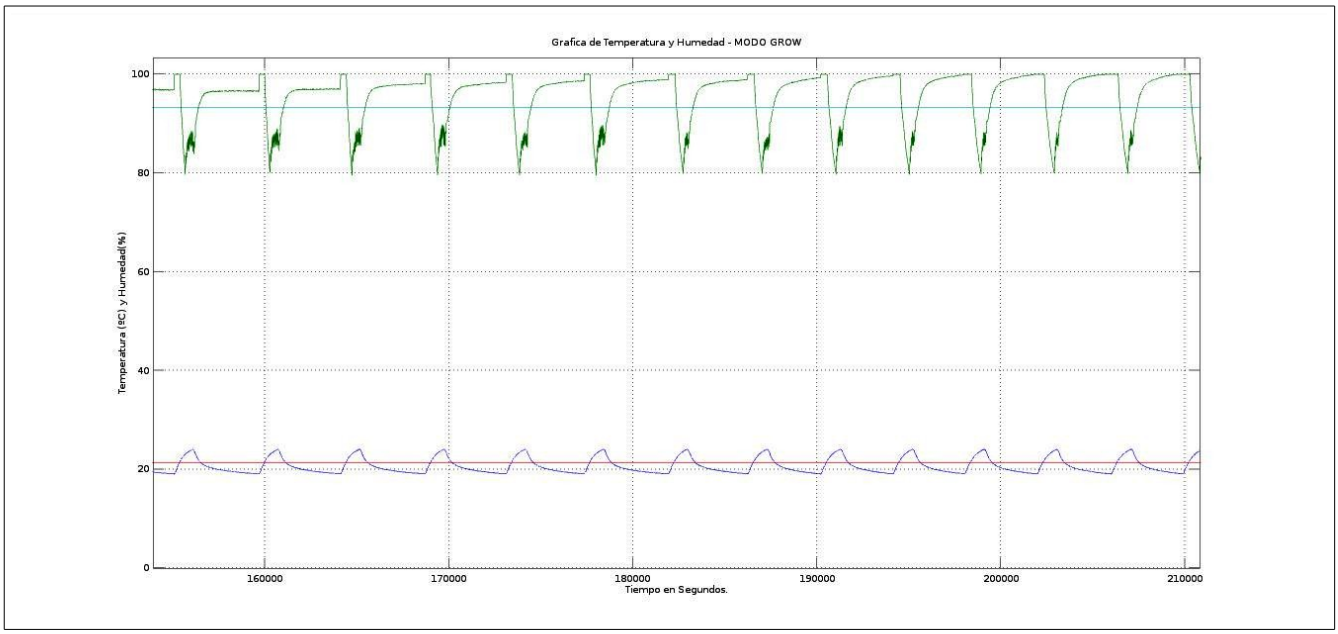
4.5 Modo Grow periodo completo con valores medios de temperatura y humedad



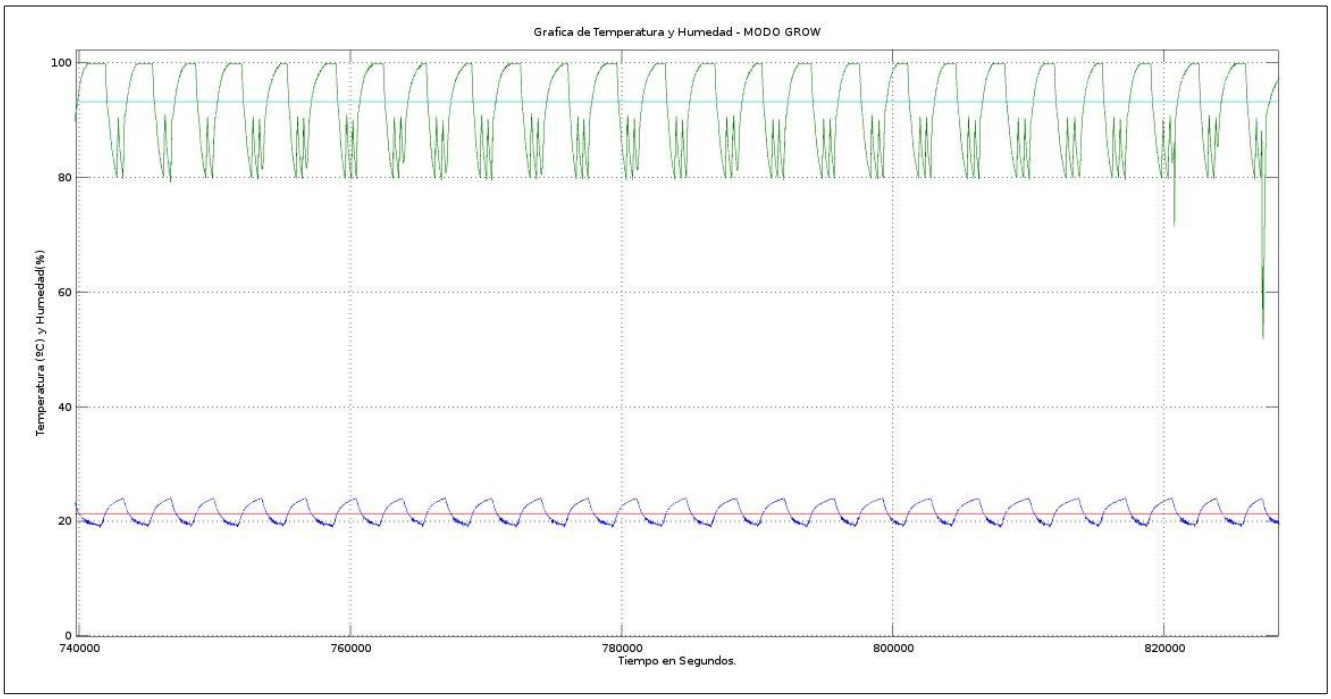
4.6 Modo Grow periodo completo con valores medios y filtro FIR de temperatura y humedad



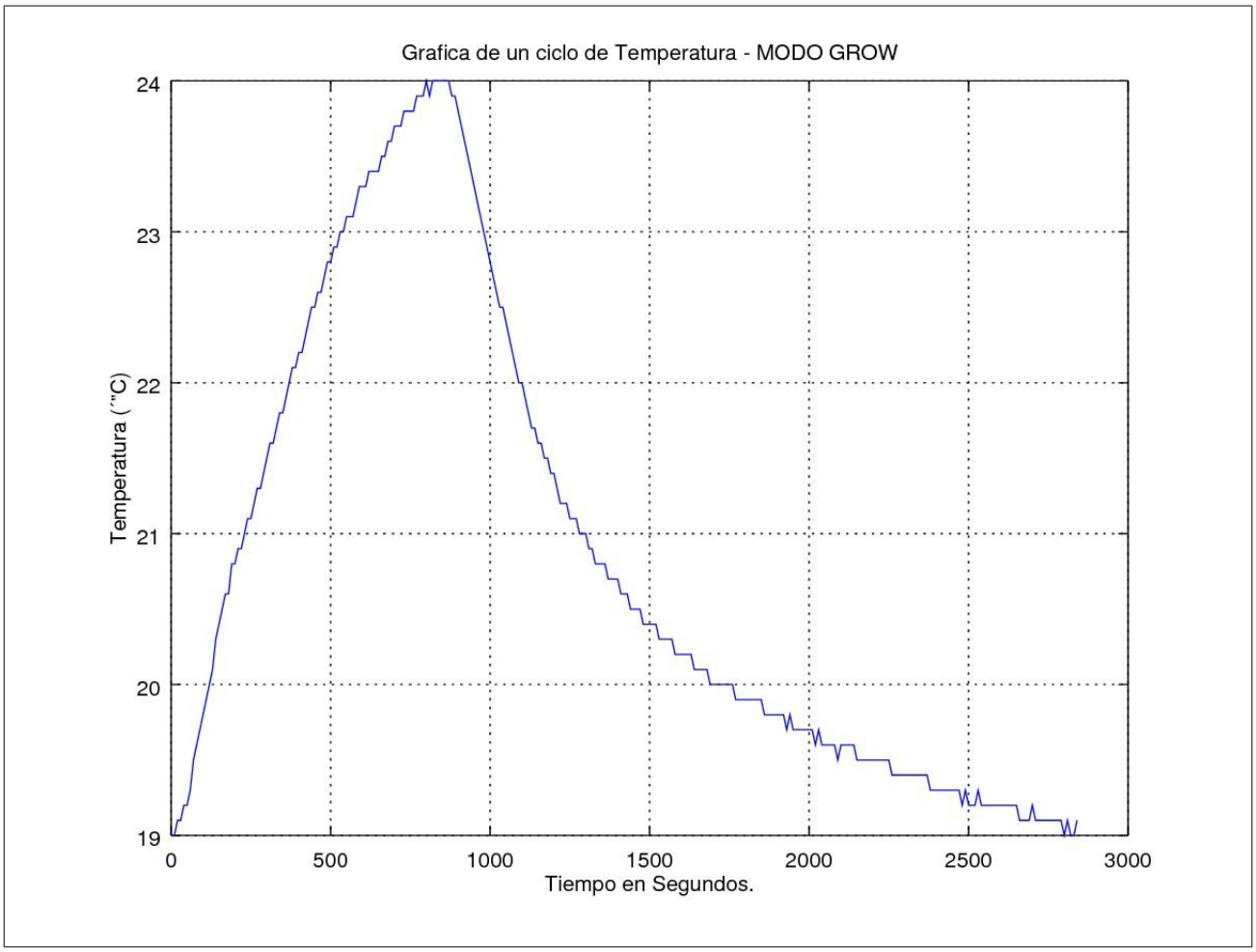
4.7 Modo Grow con zoom (escala tiempos) del principio del periodo con valores medios de temperatura y humedad



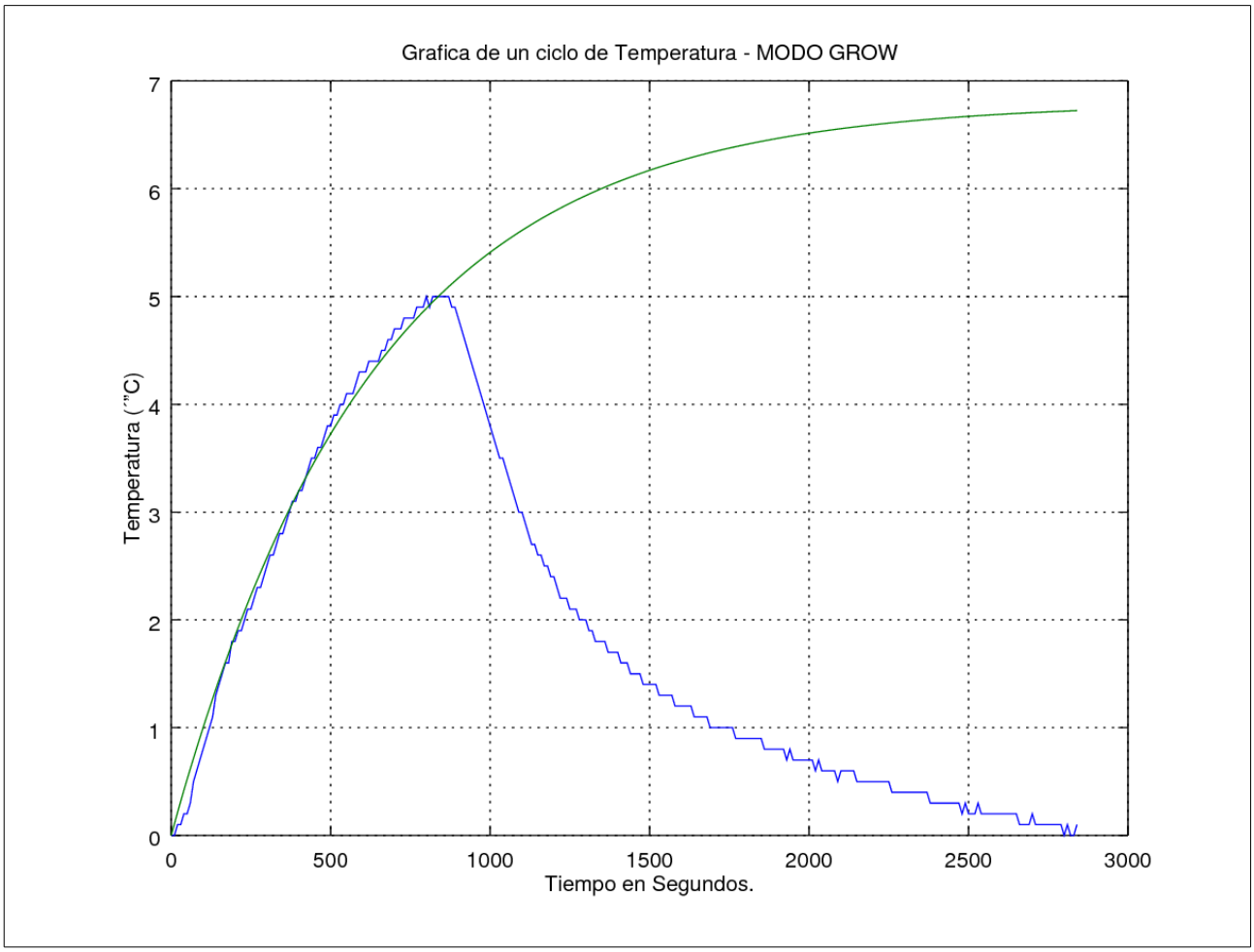
4.8 Modo Grow con zoom (escala tiempos) del final del periodo con valores medios de temperatura y humedad



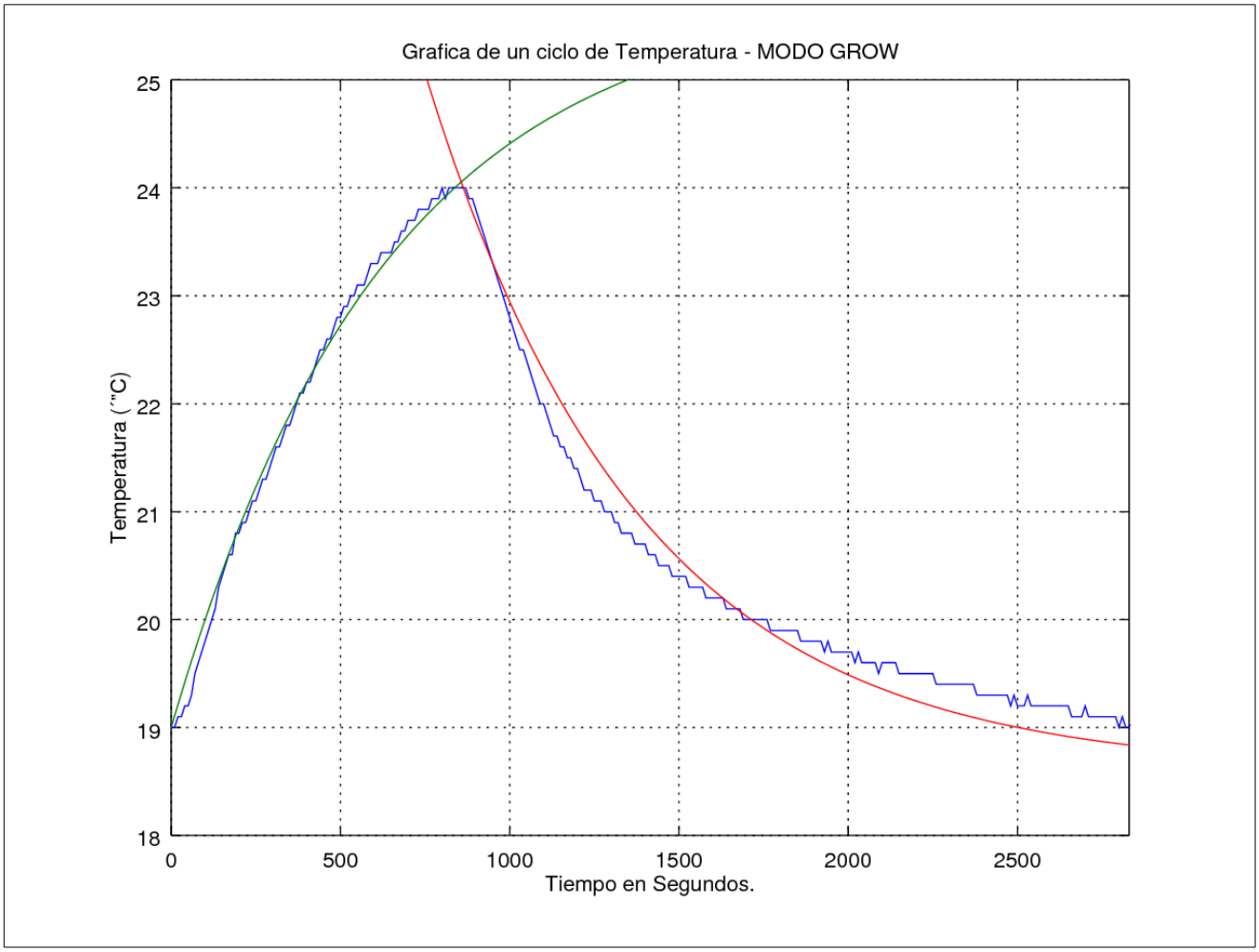
4.9 Modo Grow con la evolución de 1 ciclo de la función exponencial descrita por la variable temperatura SIN el efecto Fuzzy conectado en el invernadero



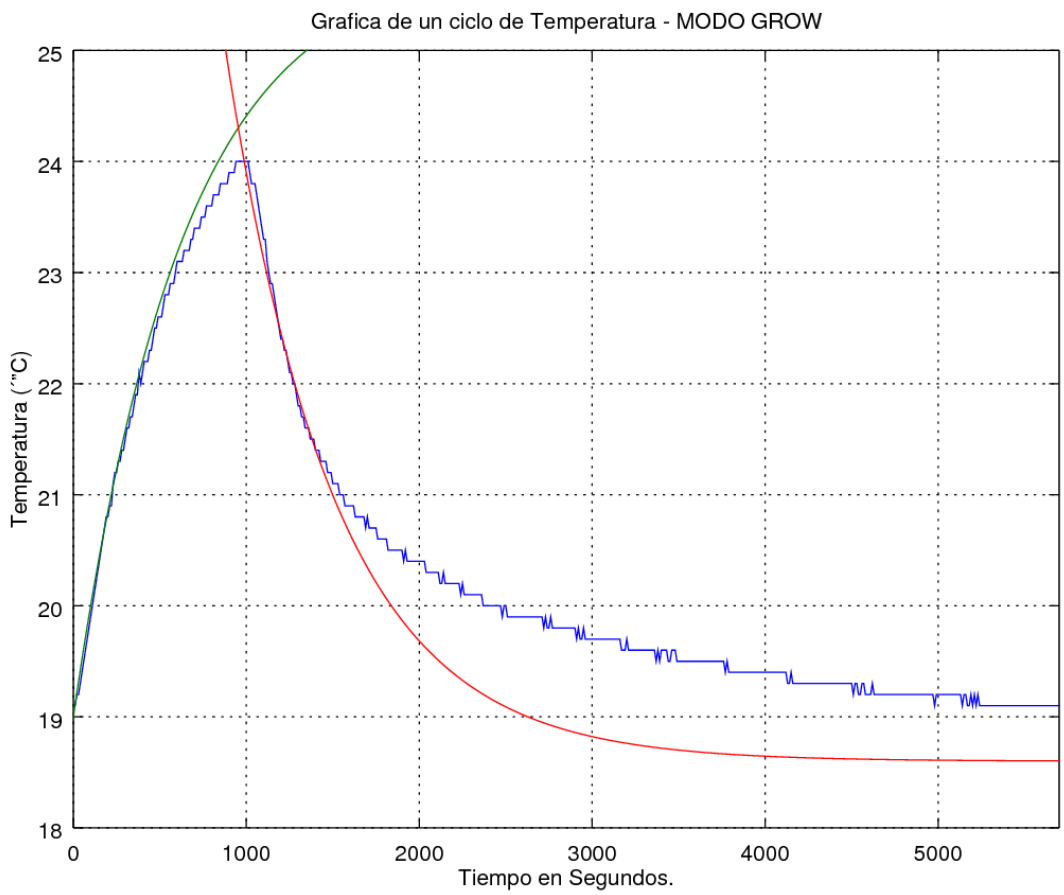
4.10 Modo Grow con la evolución de 1 ciclo de la función exponencial descrita por la variable temperatura SIN el efecto Fuzzy conectado en el invernadero y su *tau creciente* ajustada en color verde



4.11 Modo Grow con la evolución de 1 ciclo de la función exponencial descrita por la variable temperatura SIN el efecto Fuzzy conectado en el invernadero. Pintamos su *tau creciente* ajustada en color verde, y la *tau decreciente* en color rojo



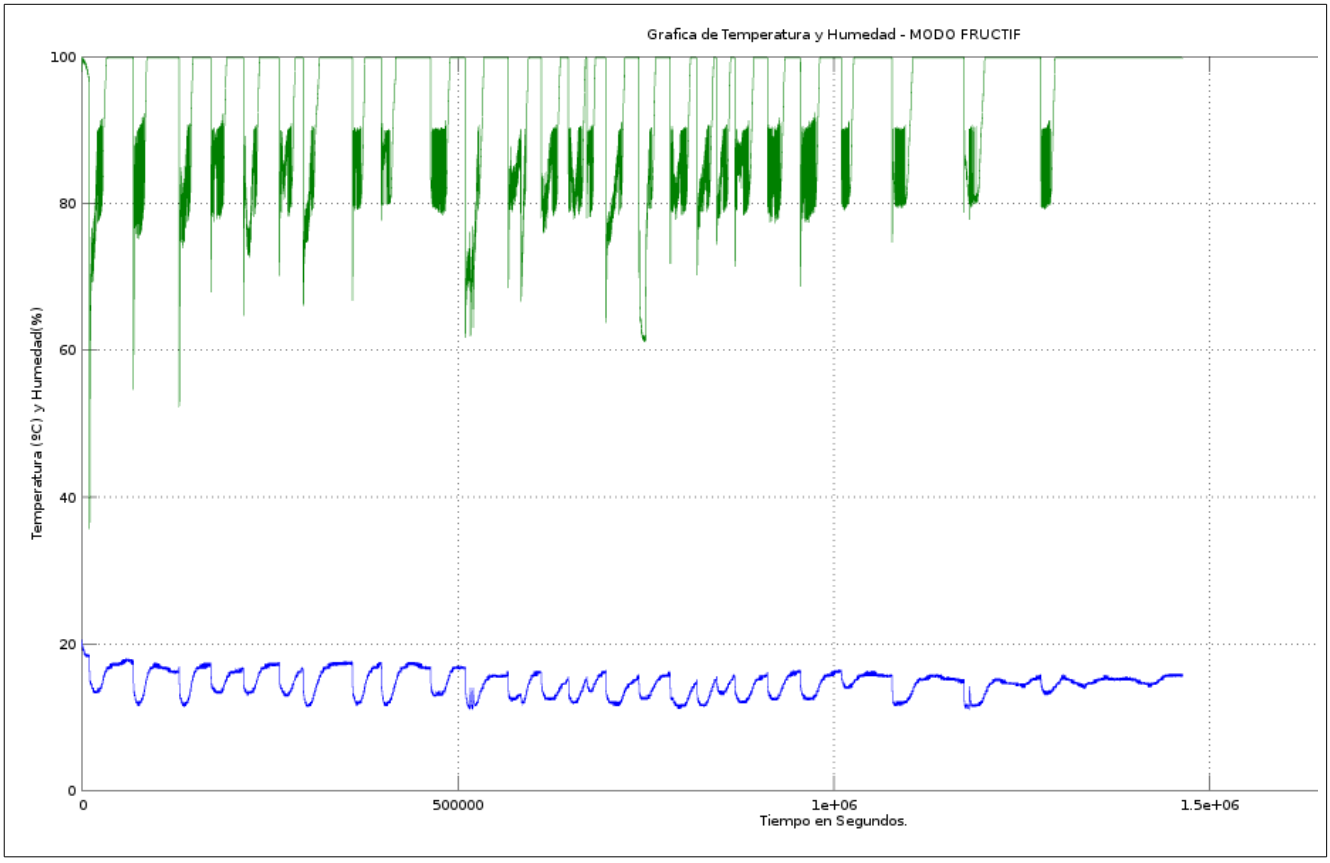
4.12 Modo Grow con la evolución de 1 ciclo de la función exponencial descrita por la variable temperatura CON el efecto Fuzzy conectado en el invernadero y su *tau* ajustada (para visualizar el efecto de dicho control)



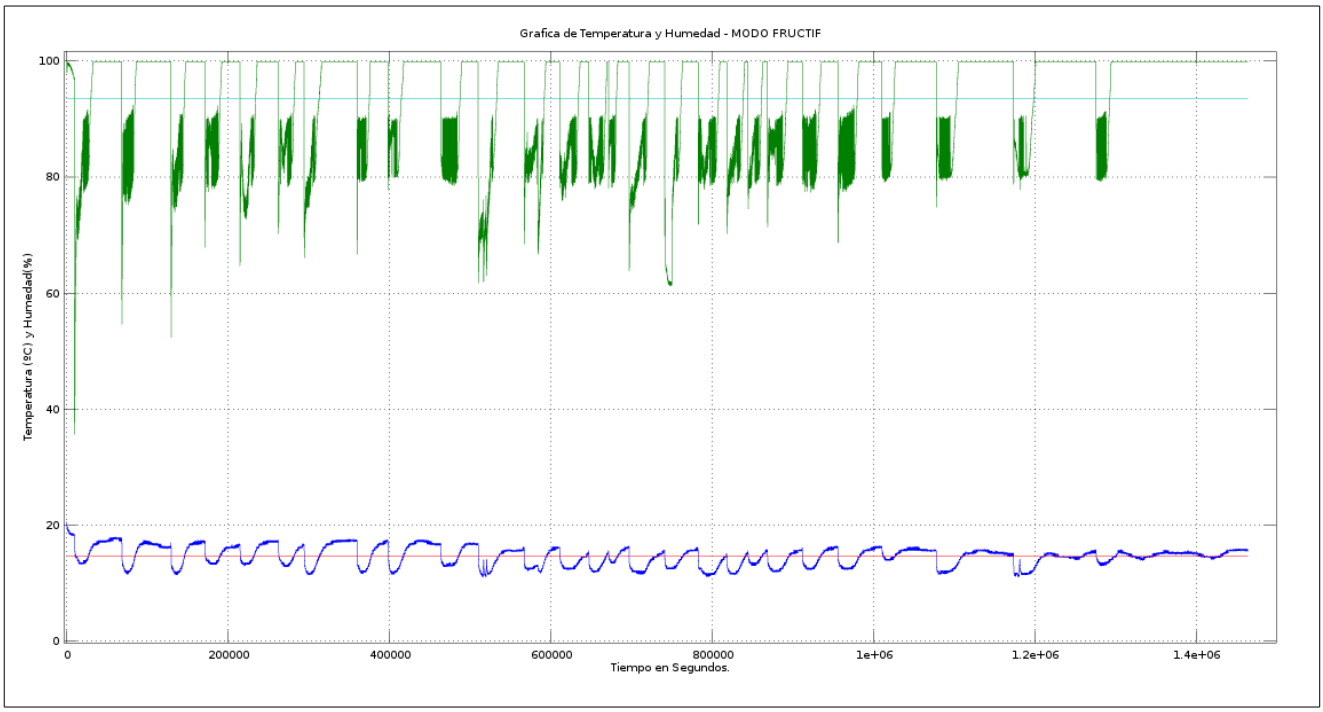
Observamos como el control Fuzzy provoca que la curva decreciente sea menos abrupta. Lo que se traduce en una oscilación de T más estable y cercana a la media deseada.

2.8.5. Generación y análisis de gráficas modo Fructif

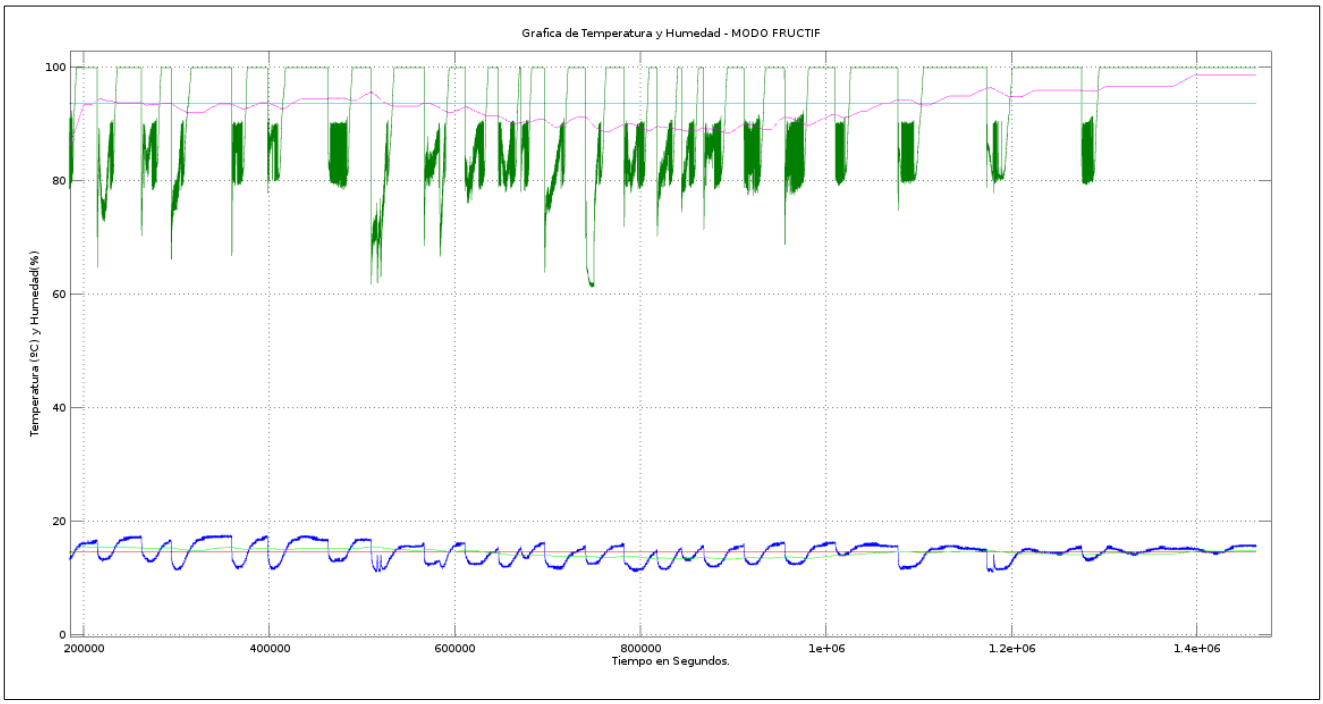
5.1 Modo Fructif periodo completo



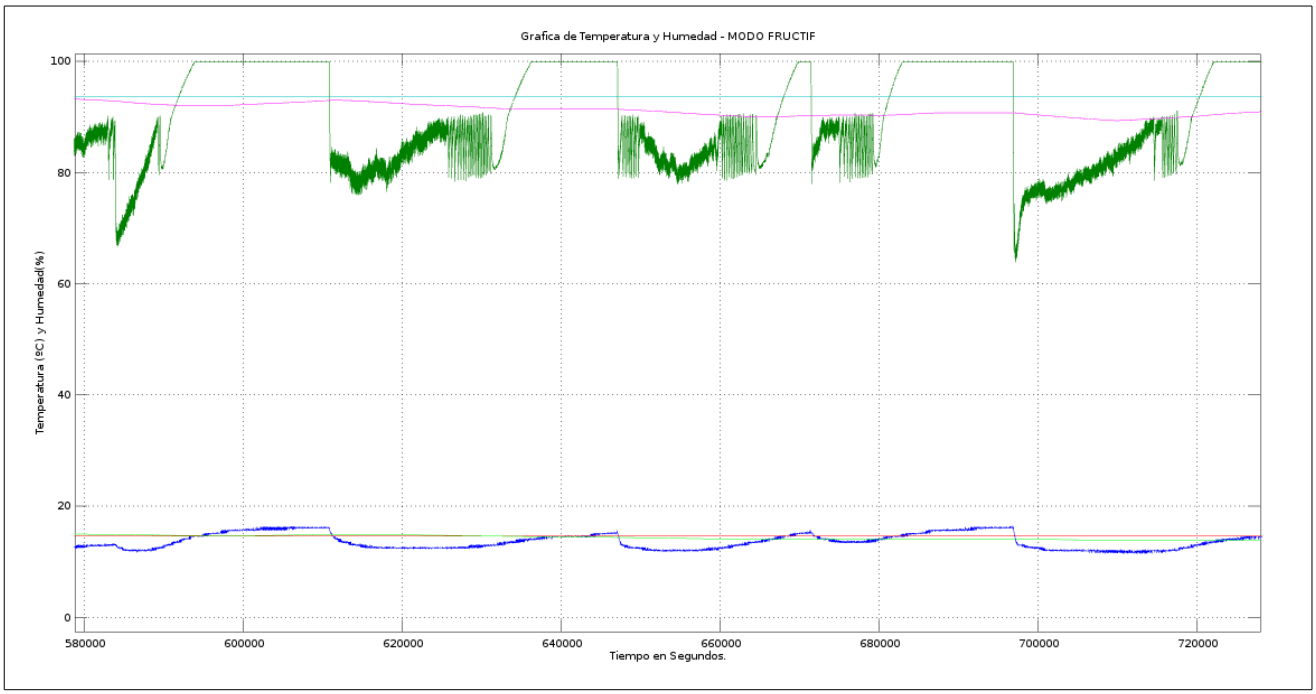
5.2 Modo Fructif periodo completo con los valores medios de temperatura y humedad



5.3 Modo Fructif periodo completo con los valores medios y filtro FIR de temperatura y humedad



5.4 Modo Fructif con zoom (escala de tiempos) de los valores medios y filtro FIR de temperatura y humedad

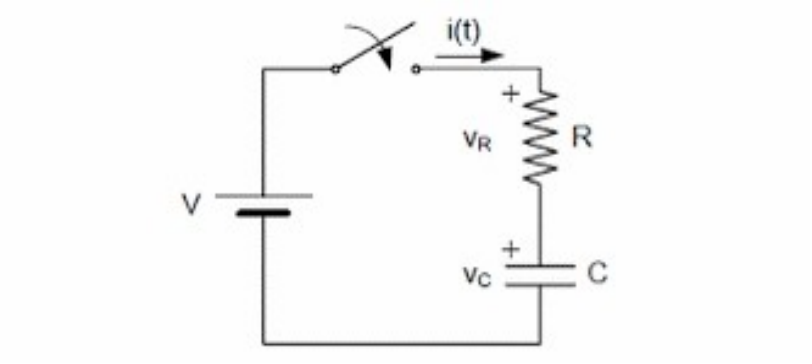


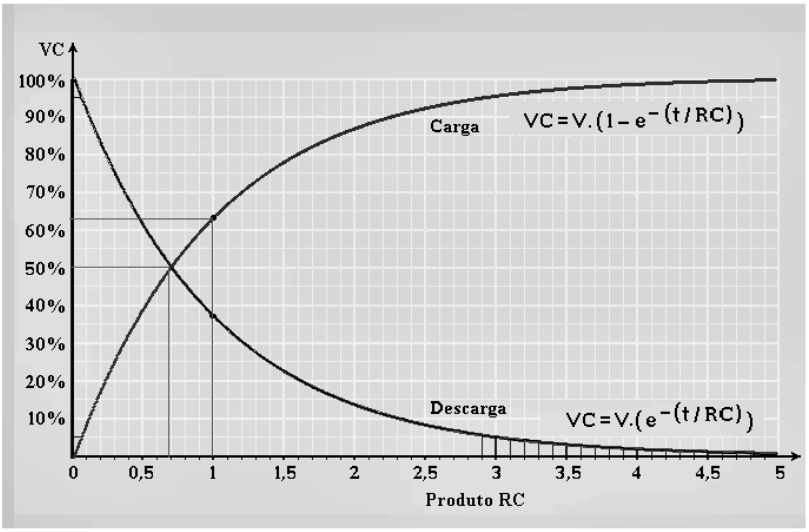
2.8.6. Modelado teórico de nuestro invernadero a partir de la respuesta experimental

Observamos que la respuesta de nuestro sistema se ajusta al comportamiento de un circuito RC serie típico.

Aunque en nuestro caso la variable de estudio es la temperatura. Por tanto definimos las siguientes analogías:

$\Delta V = \Delta T$
$I = P$
$R = R_T$
$C = C_{calorifica\_aire}$

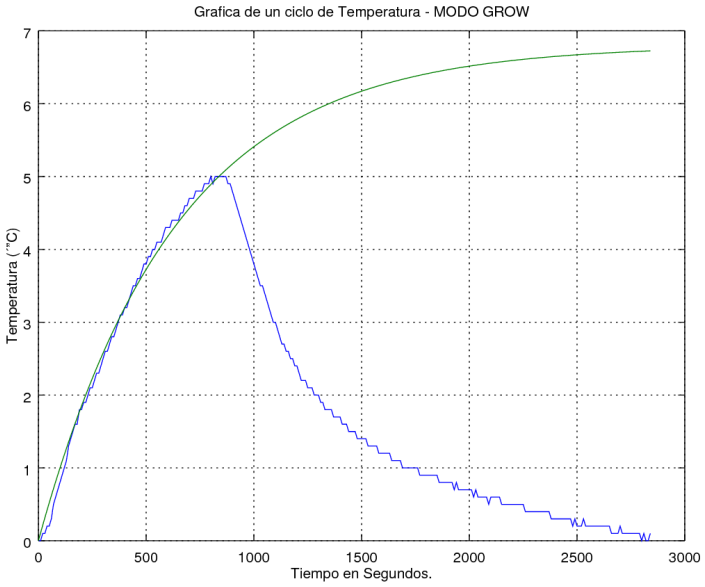




Hay que decir que esta gráfica representa un valor aleatorio escogido del estudio del modo Grow, en régimen permanente y con el invernadero en vacío.

Obtenemos nuestra τ del circuito (que representa el máximo ΔT alcanzable por nuestro invernadero) a través del estudio visual de la gráfica.

Tenemos una τ=630s



Como sabemos que τ=RC

Donde: *τ = tiempo de carga*
R=R_T= Resistencia térmica
C=$C_{calorifica\_aire}$= Capacidad calorífica del aire

Averiguamos la $C_{calorifica\_aire}$= 0,29 Kcal / m³ · ºC
Sabemos que nuestro Volumen del invernadero es:
V=0,3·0,3·0,4=0,036 m³

Calculamos: $C_{cal\_aire\_invernadero}$ = 0,29 · 4186,8 J / m³ · ºC =
= 1214,2 J / m³ · ºC
= 1214,2 ·0,036 =
$C_{cal\_aire\_invernadero}$ = 43,71 J / ºC

Ahora despejamos: R_T = τ / $C_{cal\_aire\_invernadero}$
R_T = 630 / 43,71= 14,41 ºC / W

Sabiendo estos datos podemos obtener la eficiencia térmica de nuestro invernadero con célula peltier

Sabemos que: ΔT max = 3 ºC (medida aproximada)
$P_{peltier}$ = 50W
R_T = 14,41 ºC / W

Por tanto la e=ΔT_{max} / (R_T · $P_{peltier}$) ·%

Eficiencia térmica resultante de nuestro prototipo de invernadero:

e = 0,4 %

2.8.7 Conclusiones

La eficiencia de una célula peltier ronda generalmente el 5-10% de la eficiencia de un refrigerador ideal.

Por lo que nuestro habitáculo diseñado tiene una eficiencia muy por debajo de lo que cabría esperar en un diseño ideal.
Debido seguramente a las pérdidas de temperatura producidas por los siguientes aspectos:

- Fugas de aire por la ventana del invernadero
- Transferencia de calor de los componentes metálicos que aíslan la caja y tienen un leve contacto con el exterior
- La elevada humedad del habitáculo provocado por el humidificador
- Flujo constante de aire inyectado por la bomba

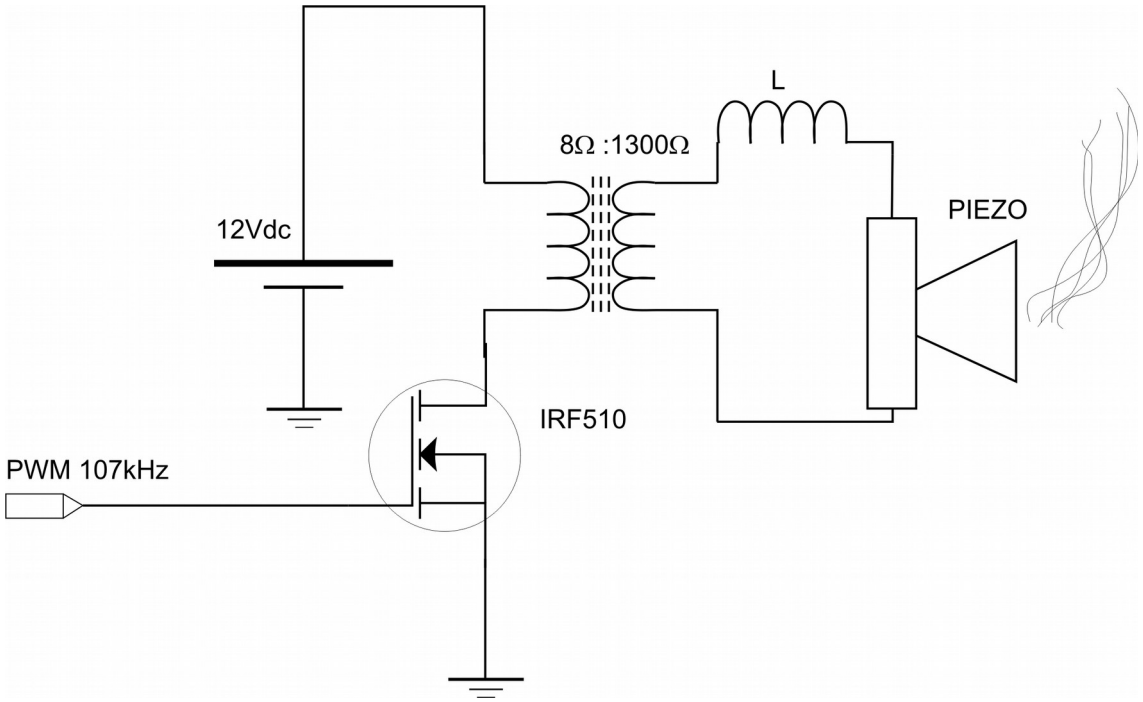
2.9 Solución de problemas surgidos durante el desarrollo

2.9.1. Resolución a nivel teórico

1.1 Cálculo del disipador del MOSFET que conmuta el Humidificador.

Observamos que el mosfet utilizado para conmutar el humidificador se calienta, así que estudiamos a nivel teórico las causas.

Esquema del circuito:



<u>Datos:</u>

Potencia del PIEZO = 2 Watios
Capacidad estática = 3’6nF
Frecuencia de resonancia = 107 kHz.

MOSFET tipo N modelo IRF510

rdsON = 0.54Ω
Rthja = 62º/W
Rthcs = 0’5º/W
Rthjc = 3’5º/W
Tjmax = 175ºC

Transformador de audio de 8 a 1300 ohmios

$$N = \frac{Vo}{Vi} = \frac{Ii}{Io} = \frac{Zo \cdot Io}{Zi \cdot Ii} = \left(\frac{Zo}{Zi}\right) \cdot \left(\frac{1}{N}\right)$$

$$N^2 = \left(\frac{Zo}{Zi}\right)$$

$$N = \sqrt{\left(\frac{Zo}{Zi}\right)} = \sqrt{\left(\frac{1300}{8}\right)} = 12.75$$

Calculamos la tensión en bornes del primario del transformador:

$$I_{prim\_trafo} = \sqrt{\left(\frac{2}{8}\right)} = 0.5A$$

Vbornes = 0.5 · 8 = 4 Voltios

Calculamos la tensión en bornes del secundario del transformador (aplicada al PIEZO):

Vpiezo = Vbornes · N = 4 · 12’75 = 51Vpp

Cálculo de la potencia disipada en el MOSFET:

rdsON = 0.54Ω
Vmosfet = rdsON · Iprim_trafo = 0.54 · 0.5 = 0’3 Voltios mínimo
Pmosfet = Vmosfet · Iprim_trafo = 0.3· 0.5 = 0’15 Watios mínimo

En realidad disipa más potencia por la reactiva que provoca el efecto capacitivo del PIEZO. Tendremos una reactiva con un cos(φ) = -1 prácticamente todo capacitivo. Por eso si el PIEZO es de 2W de potencia, entrarán al circuito casi 4W porque 2W serán de reactiva que se disiparán sobre los elementos resistivos del circuito primario si no compensamos esta reactiva. Para compensar esta reactiva colocaremos la bobina L en serie con el PIEZO en el secundario del transformador de audio. En nuestro caso hemos comprobado que si no ponemos la bobina L de compensación de reactiva nuestro mosfet se calienta un poco y hay que ponerle un disipador. La potencia disipada llega a los 2W comprobados mediante mediciones.

Cálculo de la inductancia L de compensación de reactiva:

Frecuencia de funcionamiento: 107kHz
Capacidad estática del PIEZO: 3’6nF

$$f_0 = \frac{1}{2\pi\sqrt{L \cdot C}}$$

Despejamos L:

L=1/(((2*pi*fo)^2)*C) = 615uH

Como en el modelo de los transformadores existe una inductancia serie de fugas, elegiremos una L un poco menor a la calculada. P.ej.: 470uH o 330uH.

Cálculo de la temperatura máxima de la unión sin disipador para una temperatura ambiente máxima para la pieza de 50ºC:

Tjmax < Tamb + (W · Rthja)
175 < 50 + (2 · 62)
175 < 174 → Vemos que estaríamos al límite, mejor poner un disipador aunque sea pequeño

NOTAS:

Tjmax = temperatura máxima de la unión
Tamb = temperatura ambiente del componente
W = potencia disipada en forma de calor
Rthja = Resistencia térmica unión-ambiente

Cálculo del disipador para mejorar la temperatura máxima de la unión:

Tjmax < Tamb + (W · (Rthsa + Rthjc + Rthcs))

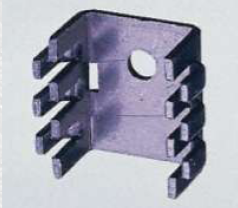
Despejamos Rthsa:
175 = 50 + (2 · (Rthsa + 3’5 + 0’5))
Rthsa = 58’5 ºW → Es decir, que como es un valor muy grande, necesitaremos un disipador muy pequeño. A mayor valor, menor es el tamaño del disipador.

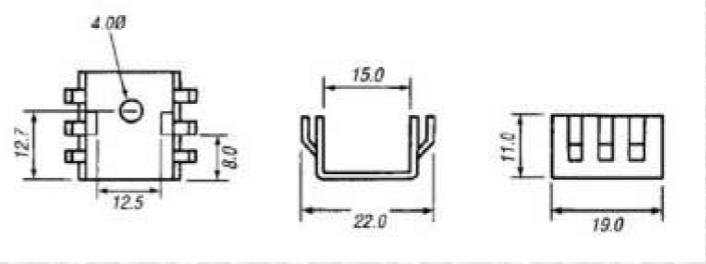
NOTAS:

Tjmax = temperatura máxima de la unión
Tamb = temperatura ambiente del componente
W = potencia disipada en forma de calor
Rthsa = Resistencia térmica disipador-ambiente
Rthjc = Resistencia térmica unión-carcasa
Rthcs = Resistencia térmica carcasa-disipador

Viendo en el catálogo de disipadores: elegimos uno tipo el HS24 que supera con creces nuestras especificaciones (30º/W).

Tipo	Información básica
HS24	Disipador de calor para encapsulados TO220.
Resistencia térmica: 30.00 ºC/W	



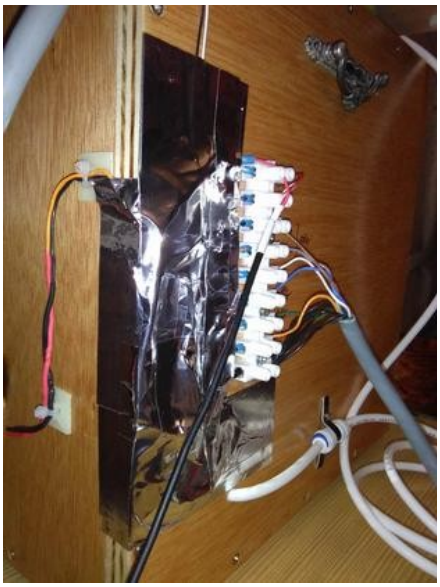
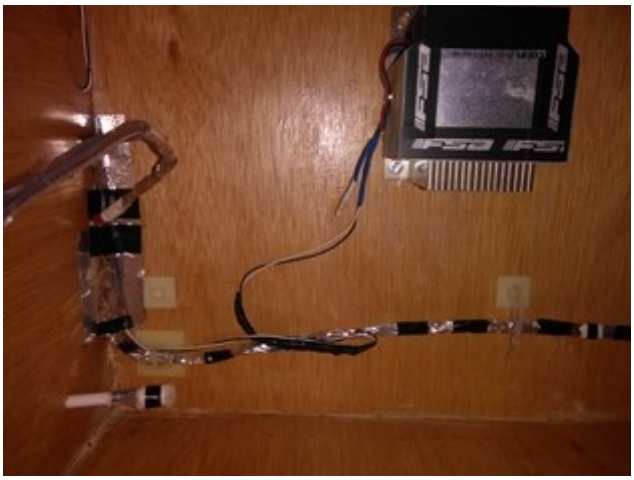


2.9.2. Resolución a nivel práctico

2.1 Interferencias

Observamos que se producen una gran cantidad de interferencias debido a la topología de la señal que alimenta al piezoeléctrico del humidificador (señal cuadrada, trafo empleado de audio pero trabajando por encima del umbral de trabajo diseñado, Vpp= 51V, f=107 Khz)

<table>
<tr><td colspan="2">Soluciones:</td></tr>
<tr><td colspan="2">Separación del cable del humidificador del resto del circuito</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Situación antes</td><td>Situación después</td></tr>
<tr><td>El trafo estaba situado dentro de la caja de control y sin ningún tipo de apantallamiento, lo cual genera interferencias.</td><td>Sacamos el trafo fuera, lo aislamos con cinta de aluminio, sustituimos el cable telefónico de dos hilos con el que estaba conectado por uno un cable de audio estéreo de dos vivos y malla. Conectamos la malla a masa.
Ademas pegamos el trafo a la carcasa metálica de la fuente de alimentación para que ayude al filtrado de interferencias.</td></tr>
</table>

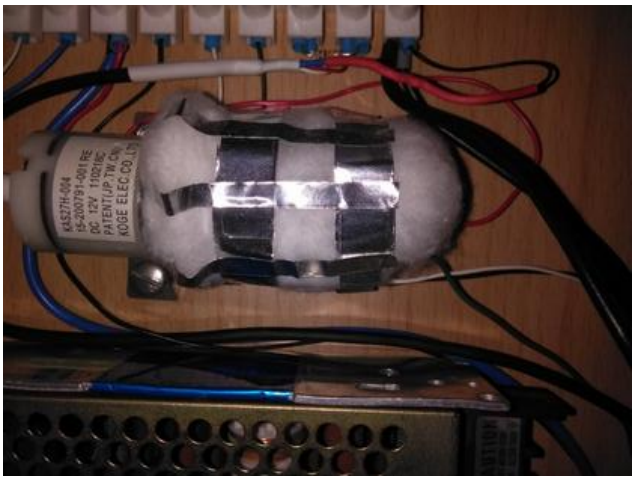
Apantallamiento del resto de cables del invernadero	
	
Cables externos	Cables internos
	
Cable del sensor	Salida cable del humidificador

Vista general del apantallamiento interior

Enrrollamos en bucles el cable del humidificador y lo pegamos con cinta de aluminio a la carcasa de la fuente.
De este modo obtenemos menos interferencias en el display LCD situado en nuestro tablero de control

2.2 Filtrado del caudal de aire suministrado por la bomba al interior del invernadero

Para evitar la contaminación de bacterias, mohos y hongos del exterior fabricamos un filtro casero con algodón y cinta de aluminio



Filtrado en la entrada de aire de la bomba situada en el tablero de control

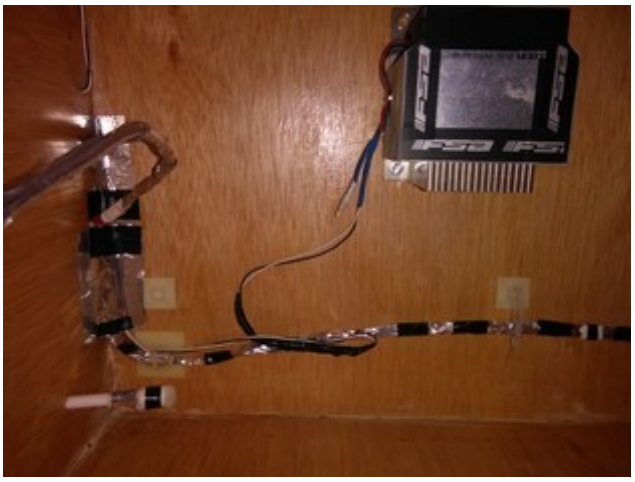
	Filtrado en el interior del invernadero. Salida del caudal de aire que inyecta la bomba al sistema de acondicionado

2.3 Aislamiento de zonas susceptibles de acumular humedad

	Ponemos cinta de aluminio en estas zonas ya que observamos la aparición de mohos blancos, verdes y negros. Esto debido a la constante acumulación de agua en estas zonas: juntas de la puerta de la ventana, visagra, etc.

	Aquí observamos con más detalle el sellado con cinta de aluminio del marco de la ventana de metacrilato, visagra y juntas varias. Ademas situamos unas tiras de cinta aislante blanca a modo de indicador del menor síntoma de aparición de moho

2.4 Sustitución del radiador del interior de la caja

	
Radiador primario	Radiador sustituido

Sustituimos el primer radiador que pusimos por uno más grande que favoreciera la disipación de calor de la peltier. Buscando por tanto una mejora en la eficiencia energética del habitáculo.

Después de estudiar los resultados concluimos que se mejora levemente el comportamiento térmico del mismo

CAPÍTULO III: CONCLUSIONES

3.1 Conclusiones generales

El presente proyecto tenía como propósito desarrollar un control climático mediante lógica Fuzzy de un prototipo de invernadero doméstico para el cultivo de setas.

A nivel general y teniendo en cuenta los objetivos planteados al comienzo del proyecto, podemos afirmar que hemos logrado alcanzarlos casi al completo.
Nos introducimos en el complejo mundo del cultivo de setas (al menos para nosotros que no somos biólogos nos supone un reto). Aprendemos a usar el Arduino satisfactoriamente. Logramos desarrollar un control fuzzy eficiente. Nos atrevemos a emplear una célula peltier como refrigerador del sistema, aunque en este caso no obtenemos las expectativas deseadas. Y por último logramos desarrollar todo el proyecto con software y hardware libre, a excepción del Matlab (Fuzzy Toolbox).

A nivel específico detallaremos ciertos aspectos que consideramos de interés.

Con respecto a la caja de cultivo:

Tenemos muchas perdidas de aire en la caja por las juntas de la puerta, bisagra y resto de componentes adheridos a ella; lo que se traduce en disipación de energía en forma de calor hacia el exterior. Esto perjudica al sistema de control e impide que se alcancen los parámetros calculados teóricamente en una situación ideal de funcionamiento.
Por tanto seria importante mejorar el aislamiento del habitáculo.

La madera no es un buen material aislante para poder emplearla como caja de cultivo de hongos. Ya que resulta un material pesado, absorbe mucha agua en concentraciones elevadas de humedad (a pesar de tener un buen hermetizado con pulimento hidrófugo), y al ser orgánico se le adhieren multitud de hongos y bacterias perjudiciales para mantener la higiene del sistema.
Resultaría mas conveniente utilizar materiales plásticos o de fibra sintética.

Analizando el sistema de control de temperatura:

La refrigeración con células Peltier no resulta eficiente para cultivar setas en condiciones óptimas. Llegamos a esa conclusión tras el análisis de los datos obtenidos en los cultivos experimentales y estudiar los márgenes de actuación en los que se mueve nuestro invernadero.
Al menos con el diseño actual, todo hay que decirlo. Ya que instalando más células por toda la caja probablemente conseguiríamos alcanzar la temperatura deseada para que la seta se desarrolle adecuadamente. El problema estaría entonces en que el diseño del circuito se complicaría considerablemente. Además el consumo de potencia del sistema se dispararía planteándonos si realmente merece la pena utilizar este método de refrigeración para enfriar el invernadero.

En cuanto al cultivo doméstico de setas:

No es bueno humidificar tanto dentro del invernadero ya que se acumula demasiada agua en los recipientes de cultivo y ello favorece la proliferación de bacterias u otros hongos ajenos al cultivo.

No es muy bueno la ventilación directa sobre los cultivos ya que reseca en exceso el sustrato de cultivo. Habría que separar el caudal de aire inyectándolo de otra manera como hacen las neveras por ejemplo.

El cultivo de setas es una disciplina más delicada de lo que pensábamos en un principio y si fallan los parámetros necesarios no se desarrolla adecuadamente el cultivo.

Pero gracias al registro de parámetros realizado y a su estudio posterior seremos capaces de mejorar las condiciones de crecimiento de la seta.

A rasgos generales citaremos lo siguiente:

Cabe destacar que durante el desarrollo de todo el proceso de elaboración del prototipo de invernadero, nos hemos visto inmersos en 7 disciplinas diferentes.

Las citamos para que se entienda:

- Mecánica
- Informática
- Control / Automatización
- Eléctrica / Electrónica
- Biología / Agroalimentación
- Compatibilidad electromagnética
- Estética / Diseño

La mecánica en cuanto al montaje del invernadero. A nivel informático en cuanto al diseño del programa de Arduino. El control y automática con respecto a la implementación del sistema fuzzy en nuestro invernadero. La parte eléctrica y electrónica debido al tablero de control. La biología o más en concreto la parte dedicada a la agroalimentación, a la hora de realizar los cultivos domésticos de seta de ostra. La compatibilidad electromagnética, en cuanto al estudio de interferencias. Y como no, ese enfoque tan importante hoy en día para vender un producto, que es realizar un diseño estéticamente agradable a la vista.

Por todo ello podemos decir que nos sentimos orgullosos del trabajo realizado y de las horas de esfuerzo aplicadas en el presente proyecto.

También somos conscientes de que se podría seguir mejorando el proyecto y por ello dejamos algunas ideas a continuación para futuros aplicaciones del prototipo.

3.2 Futuras ampliaciones y mejoras

A continuación detallamos algunas de las mejoras que incorporamos en el invernadero durante el desarrollo del mismo para optimizar su funcionamiento. Así como las futuras ampliaciones que se nos ocurren para obtener un control mas eficiente.

Mejoras en el invernadero:

- Filtramos el aire introducido por la bomba para evitar agentes de contaminación externos.
- Mejoramos las interferencias provocadas por la señal del humidificador, reubicando componentes del circuito en otro lugar y apantallando todos los cables del sistema susceptibles de generar interferencias.
- Sustituimos el radiador del interior de la caja para mejorar la disipación del calor generado por la peltier, y generar un flujo de aire que mejore la eficiencia de la misma.
- Añadimos un disipador al mosfet que conmuta el humidificador para evitar que se caliente en exceso y trabaje adecuadamente.
- Sellamos las zonas susceptibles de acumular humedad, hongos y bacterias para evitar contaminaciones del habitáculo. Lo realizamos pegando cinta de aluminio en dichas zonas.

Futuras ampliaciones:

En cuanto al funcionamiento del menú de control:

Faltaría mejorar el menú permitiendo configurar límites de temperatura y humedad. Se podrían almacenar estos datos de la configuración del menú en la memoria eeprom de Arduino.
Se podrían desarrollar funciones de test de los componentes en el menú para que el usuario tenga un control más óptimo del sistema.

En cuanto a la alimentación del tablero de control:

Se podría incorporar una batería recargable con cargador interno por si se va la luz.

En cuanto a la caja de cultivo:

Se podría cambiar el ventilador y disipador exterior que va pegado a la peltier por uno de mayor potencia. Eso provocaría una mejora de 3-4ºC en el ΔT del interior con respecto a lo que hay ahora (teniendo en cuenta los parámetros que nos da el fabricante de la peltier) en una situación ideal de funcionamiento.

Por otro lado, podríamos incorporar otro sensor de temperatura y humedad externo para mejorar el control del sistema.

En cuanto al Arduino:

Se pueden usar los pines libres de Arduino para E/S y ampliaciones futuras... se podría poner un conector para ello.

Se ha programado en Arduino Uno que está bien para prototipado pero se puede sustituir por un Arduino Pro Mini que se emplea para montaje definitivo y es mas barato (2€)

3.3 Aplicaciones del invernadero

Por último debemos hacer hincapié en que además de la utilización de nuestro prototipo de invernadero para el cultivo de setas para el cual ha sido diseñado, podemos emplearlo para otro tipo de funciones. Tan solo bastaría reprogramar el menú de control para darle otros usos.

Citamos algunos de ellos:

- Cultivo de setas de alto valor culinario, muy cotizadas en la alta cocina
- Incubación de huevos de codorniz, pato...y otras especies similares
- Elaboración de levaduras de cerveza. Se ha puesto muy de moda en los últimos años
- Fabricación de yogures caseros
- Fermentación del vino casero
- Elaboración de quesos para consumo propio
- Fermentación de levaduras de pan

Se puede controlar múltiples procesos aprovechando los parámetros de control diseñados en nuestro prototipo, lo cuál nos hace ver que disponemos de un sistema fácil de manejar y polivalente.

CAPÍTULO IV: BIBLIOGRAFÍA

[1] Altrock, C von, Krause, B. Fuzzy logic and neurofuzzy technologies in embedded automotive applications. Fuzzy Logic 93, pp.A113-9, San Francisco California, 1993

[2] Jang J.S.R., C.T. Sun, E. Mizutani. Neuro-Fuzzy and soft Computing, Prentice-Hall, 1997

[3] Mandami, E. H., Odtengaard, J. J., Lembessis, E. Use of fuzzy logic for implementing rulebased control of industrial processes. Advances in Fuzzy Sets, Possibility Theory an Applications Plenum Press, 1983.

[4] Martín del Brío, B., Sanz Molina, A. Redes neuronales y sistemas borrosos. 3a edición. Ra-Ma

[5] Reyero, R., Nicolás, F. Sistemas de control basadas en lógica borrosa. OMROM Electronics-IKERLAN,1995.

[6] Ruiz, A., Gutiérrez, J. A VLSI-CMOS programmable membership function Circuit. Fitfh IFSA World Congress, pp. 977-980, 1993

[7] Sanz Molina, A. An object oriented envelopment for complex neuro fuzzy controller. IPMU'96, pp.514-518, 1996.

[8] Sugeno J, Griffin MF, Bastian A. Fuzzy hierarchical control of an unmanned helicopter. 5Th IFSA World Congress, pp. 179-182, 1993.

[9] Terano, T., Asai, K., Sugeno, J. Fuzzy Systems Theory and its Applications. Academic Press Inc., 1992.

[10] Pérez Pueyo, Rosanna Descripción General de las Técnicas de Lógica Difusa, cap.2

[11] https://es.wikipedia.org/wiki/Lógica_difusa

[12] https://books.google.es/books

[13] https://es.wikipedia.org/wiki/Circuito_RC

[14] http://www.agromatica.es/cultivo-pleurotus-ostreatus/

[15] http://pleurotus.unpocodetodo.info/

[16] https://www.arduino.cc/

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

Control climático con lógica fuzzy de un prototipo de invernadero doméstico para el cultivo de setas

Documento Nº 2: PRESUPUESTO

AUTOR: JOSÉ CARLOS SERRANO DOMINGO

TUTOR: EDUARDO QUILES CUCARELLA

Valencia, Septiembre 2017

ÍNDICE PRESUPUESTO

1. Presupuestos parciales

A continuación, presentamos los presupuestos parciales del proyecto, que incorporan los costes de materiales, de recursos humanos y la suma total de ambos.

Presupuesto de materiales

Cantidad	Descripción	Precio Total
1	Sensor DHT 22 (AM2302) (Sensor Temperatura y humedad)	2’66€
1	Display LCD 16x2 (HD44780 retroiluminada, fondo verde)	1’79€
1	Módulo conversor I2C serie (Adaptador IIC I2C PCF8574 LCD 1602)	1’21€
1	Placa 4 Relés Optoacoplados Arduino	2’22€
2	Pulsador montaje en chasis	3’91€
1	LED 6W 12 AC/DC	0’97€
1	Célula Peltier	2’12€
1	Humidificador piezoeléctrico 107kHz	6€
1	Arduino Uno clónico	6’90€
1	Bomba inyectora de aire 12V	12’50€
1	Módulo descender DC/DC Buck 30V-3V	1’50€
2	NPN Darlington (ESM6045AV)	25€
1	MOSFET IRF510	0’30€
1	Trafo de audio de 8:1300Ω	0’10€
1	Rollo de cable para audio estéreo apantallado, 2m	1’50€
1	Rollo cable telefónico, 2m	1’30€
1	Rollo cable de pares de 0’25x12, 3m	5€
1	Rollo de cable de 1’5mm, 2m	3€
1	Caja de puntas de electricidad para crimpar	3’50€
1	Caja de conectores DuPont para Arduino	4’72€
2	Ventilador con disipador	20€
1	Chapa hierro galvanizado 0’8mm para la caja	4€
1	Tablero aglomerado para montaje electrónico	3€
1	Tablero contrachapado para invernadero	10€
1	Pulimentado y hermetizado del invernadero	20€
1	Tornillería calibre variado	3€
1	Bolsa de bridas	1€
1	Bolsa adhesivos para embridar	3€
1	Trozo de metacrilato para visor	0’50€
1	Siliconas, colas y pegamentos	7€
1	Cinta adhesiva de aluminio	3€
2	Fichas de empalme 12x2	3€
	TOTAL:	**163,70€**

Presupuesto de recursos humanos

Concepto	Número de horas	Precio (€/ud.)	Precio total (€)
Estudio y documentación	30	18	540
Horas de ingeniería (Programación y desarrollo de software en Arduino)	75	15	1125
Consultoría especializada (Control, automática, compatibilidad electromagnética, agroalimentación, desarrollo software)	20	30	600
Montaje hardware (Caja de cultivo y caja de Control)	35	12	420
Total de recursos humanos			**2685**

Suma de presupuestos parciales

Partidas	Importe(€)
Total de recursos humanos	2685
Total de componentes	163,70
Total Presupuesto	**2848,70**

2. Presupuesto total

A continuación, presentamos el presupuesto total del proyecto, que incorpora los costes anteriormente expuestos, la inclusión de un 13% de gastos generales, un 6% beneficio industrial y el 21% de Impuesto al Valor Agregado (I.V.A).

Presupuesto total de inversión

Descripción	Tasas (%)	Total (€)
Total presupuesto parcial		2848,70
Gastos generales	13	370,33
Beneficio industrial	6	170,92
Total (€)		**3389,95**

Presupuesto total base de licitación

Descripción	Impuesto (%)	Total (€)
Presupuesto sin I.V.A		3389,95
I.V.A	21	711,89
Total (€)		**4101,84**

Coste final del proyecto: 4101,84€

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

Control climático con lógica fuzzy de un prototipo de invernadero doméstico para el cultivo de setas

Documento Nº 3: MANUAL DE USO DEL INVERNADERO

AUTOR: JOSÉ CARLOS SERRANO DOMINGO

TUTOR: EDUARDO QUILES CUCARELLA

Valencia, Septiembre 2017

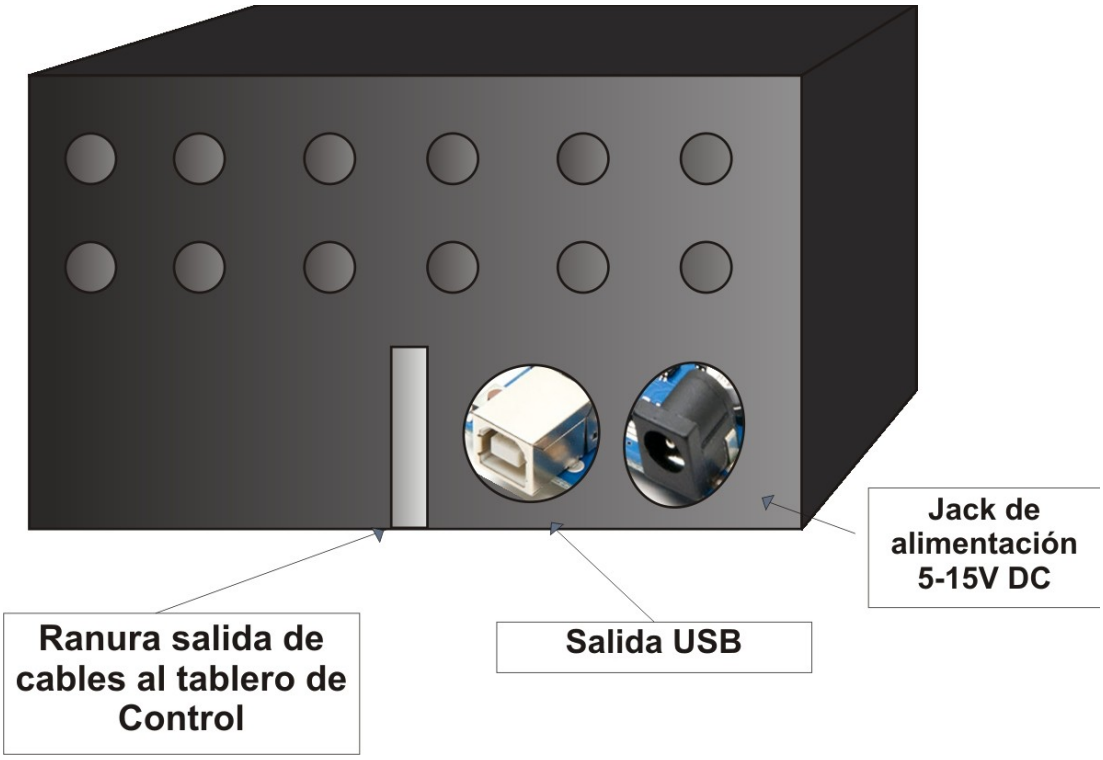
ÍNDICE MANUAL DE USO DEL INVERNADERO

MANUAL DE USO DEL INVERNADERO

CAJA DE CONTROL

Aspecto del dispositivo botones y conexiones

Botones MENU y CHANGE



Funciones de la Caja de Control

La **Caja de Control** de nuestro invernadero es el dispositivo electrónico que nos permite realizar el control climático del interior de la caja de cultivo. Conectado al resto del circuito, nos permite manipular el resto de componentes según convenga. Actuando sobre: El led, el sensor, los ventiladores, la peltier, el humidificador, sobre el display, sobre el ajuste de hora y el tiempo de los contadores.

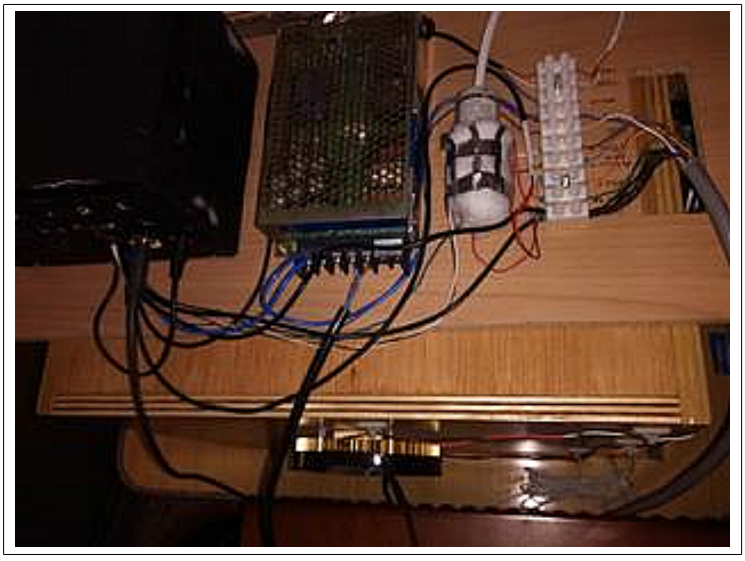
Botones del Menú de la Caja de Control

El Menú de la Caja de Control dispone de 2 botones que poseen múltiples funciones dependiendo de la opción del menú que estemos visualizando en cada momento.

BOTON MENU:
Destinado para entrar en el menú y navegar en él. Manteniéndolo presionado iremos navegando por cada una de las opciones del menú. Cada opción de menú aparecerá avanzando en el menú.

BOTON CHG:
Destinado a cambiar los valores de las opciones del menú en el que estemos. Al igual que el botón menú, las opciones irán variando al mantenerlo presionado.

Conexionado con el tablero de control



Descripción del MENU

La idea general para navegar en los menús de la C*aja de Control* es que para navegar solamente empleamos dos botones:

- El BOTÓN MENU para desplazarnos
- El BOTÓN CHG (Change) para Aceptar o Cambiar los valores.

El formato inicial del Display presenta el siguiente aspecto:

La hora del sistema en la parte superior
Y la temperatura y humedad en la parte inferior

Aspecto cuando esta un modo activo:

Al presionar el BOTÓN MENU inicialmente, podremos navegar entre las siguientes funciones:

1 -> CAMBIAR MIN

Nos permitirá configurar la hora de la C*aja de Control* para que la podamos tener de referencia como si de un reloj se tratase.

En esta opción, al presionar el BOTON CHG, nos irá incrementando el valor de los MINUTOS.

Al presionar el BOTON MENU, volveremos al menú principal para seguir navegando en él.

2 -> CAMBIAR HORA

Nos permitirá configurar la hora de la C*aja de Control* para que la podamos tener de referencia como si de un reloj se tratase.
En esta opción, al presionar el BOTON CHG, nos irá incrementando el valor de la HORA.

Al presionar el BOTÓN MENU, volveremos al menú principal para seguir navegando en él.

3 -> Humid. 05 / 10 / 15 / 20 / HUMI DESHABIL

Con esta opción podemos configurar el tiempo en segundos que el el humidificador estará en ON. El tiempo en OFF no es regulable.

Tenemos 5 opciones:
- 5s
- 10s
- 15s
- 20s
- Humi deshabil (para desconectarlo)

Al presionar el BOTON CHG, nos irá cambiando de opción
Para salir presionaremos el BOTÓN MENU.

4 -> START M AUTO

Con esta opción, podremos accionar el modo de funcionamiento automático.
Por el cual el sistema accionará el modo *Grow* (hasta que finalice su contador de horas de funcionamiento, por defecto 432h). Y cuando este haya finalizado automáticamente comenzara el modo de funcionamiento *Fructif* (hasta que finalice su contador de horas de funcionamiento, por defecto 999h).

Presionado el BOTON CHG:
Accionaremos el modo AUTO y cuando este se active apreciaremos en la parte inferior derecha del display las siglas AG/AF.
En la parte superior derecha veremos las horas de cuenta del contador, la cual apreciaremos como fluctúa en orden descendente hasta llegar a 0

Presionando el BOTON MENU:
Seguiremos navegando por el menú sin efectuar ningún cambio

5 -> START M GROW

Con esta opción, podremos accionar el modo de funcionamiento Grow.
Por el cual el sistema accionará el modo *Grow* (hasta que finalice su contador de horas de funcionamiento, por defecto 432h).

Presionando el BOTON CHG:
Accionaremos el modo GROW y cuando este se active apreciaremos en la parte inferior derecha del display la sigla G.
En la parte superior derecha veremos las horas de cuenta del contador, la cual apreciaremos como fluctúa en orden descendente hasta llegar a 0

Presionando el BOTON MENU:

Seguiremos navegando por el menú sin efectuar ningún cambio

6 -> START M FRUCT

Con esta opción, podremos accionar el modo de funcionamiento Fructif.
Por el cual el sistema accionará el modo *Fructif* (hasta que finalice su contador de horas de funcionamiento, por defecto 999h).

Presionado el BOTON CHG:
Accionaremos el modo FRUCTIF y cuando este se active apreciaremos en la parte inferior derecha del display la sigla F.
En la parte superior derecha veremos las horas de cuenta del contador, la cual apreciaremos como fluctúa en orden descendente hasta llegar a 0

Presionando el BOTON MENU:
Seguiremos navegando por el menú sin efectuar ningún cambio

7 -> STOP

Opción que nos permite detener el modo de funcionamiento activo en el invernadero en ese momento.

Presionado el BOTON CHG lo activaremos:
Desconectará la peltier, la bomba de aire y el humidificador. El resto seguirá activo.
Además reseteará el contador poniéndolo a 0 y reflejando dicha cuenta en la parte superior derecha del display (“000”).
En la parte inferior derecha aparecerá la sigla S

Presionando el BOTON MENU:
Seguiremos navegando por el menú sin efectuar ningún cambio

8 -> FUZZY HABILIT / FUZZY DESHABI

Nos permitirá accionar el control Fuzzy del invernadero.
De este modo actuaremos sobre el caudal de aire de la bomba

Al presionar el BOTON CHG lo activaremos o desactivaremos según convenga a nuestro cultivo.

Presionando el BOTON MENU:
Seguiremos navegando por el menú sin efectuar ningún cambio

9 -> INCREMEN CONT

Esta opción nos permitirá incrementar las horas del contador en intervalos de 10 en 10

unidades. Con ello conseguiremos ajustar el periodo de cultivo a nuestro antojo.

Al presionar el BOTON CHG, nos irá incrementando el valor.
Lo observaremos en la parte superior derecha.

Presionando el BOTON MENU:
Seguiremos navegando por el menú sin efectuar ningún cambio

10 -> DECREMEN CONT

Esta opción nos permitirá decrementar las horas del contador en intervalos de 10 en 10 unidades. Con ello conseguiremos ajustar el periodo de cultivo a nuestro antojo.

Al presionar el BOTON CHG, nos irá decrementando el valor.
Lo observaremos en la parte superior derecha.

Presionando el BOTON MENU:
Seguiremos navegando por el menú sin efectuar ningún cambio

11 -> SALIR

Al seleccionar esta opción saldremos de la navegación del Menú sin interferir en lo que estuviera activo en ese momento.
Se apagará el led del display y aparecerá:

- La pantalla inicial con la hora del sistema y las horas del contador en la parte superior.
- Y la temperatura y humedad en la parte inferior

Al presionar el BOTON CHG lo activaremos

Presionando el BOTON MENU:
Seguiremos navegando por el menú sin efectuar ningún cambio

Las opciones del MENU aparecen siempre en la línea superior del Display.
Mientras que en la parte inferior siempre aparecerá:
Este formato → T xx.xx H xx.xx % (x representa un dígito numérico)
Además la T cuando esté algún modo activo se tranforma en un ***** si la peltier está enfriando o **!** si la peltier está calentando.

Reset del LCD

Manteniendo presionado primero el BOTON MENU y luego sin soltar el MENU presionando BOTON CHG durante al menos 10 segundos conseguiremos resetear el LCD por si se nos hubiera quedado colgado.

Resumen del Menu de la Caja de Control

1 -> Cambiar Min
2 -> Cambiar Hora
3 -> Humid. 05/10/15/20/HUMI DESHABIL
4 -> START M AUTO
5 -> START M GROW
6 -> START M FRUCT
7 -> STOP
8 -> FUZZY HABILIT/FUZZY DESHABI
9 -> INCREMEN CONT
10 -> DECREMEN CONT
11 -> SALIR

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

Control climático con lógica fuzzy de un prototipo de invernadero doméstico para el cultivo de setas

Documento Nº 4: ESQUEMAS DE CONEXIONADO

AUTOR: JOSÉ CARLOS SERRANO DOMINGO

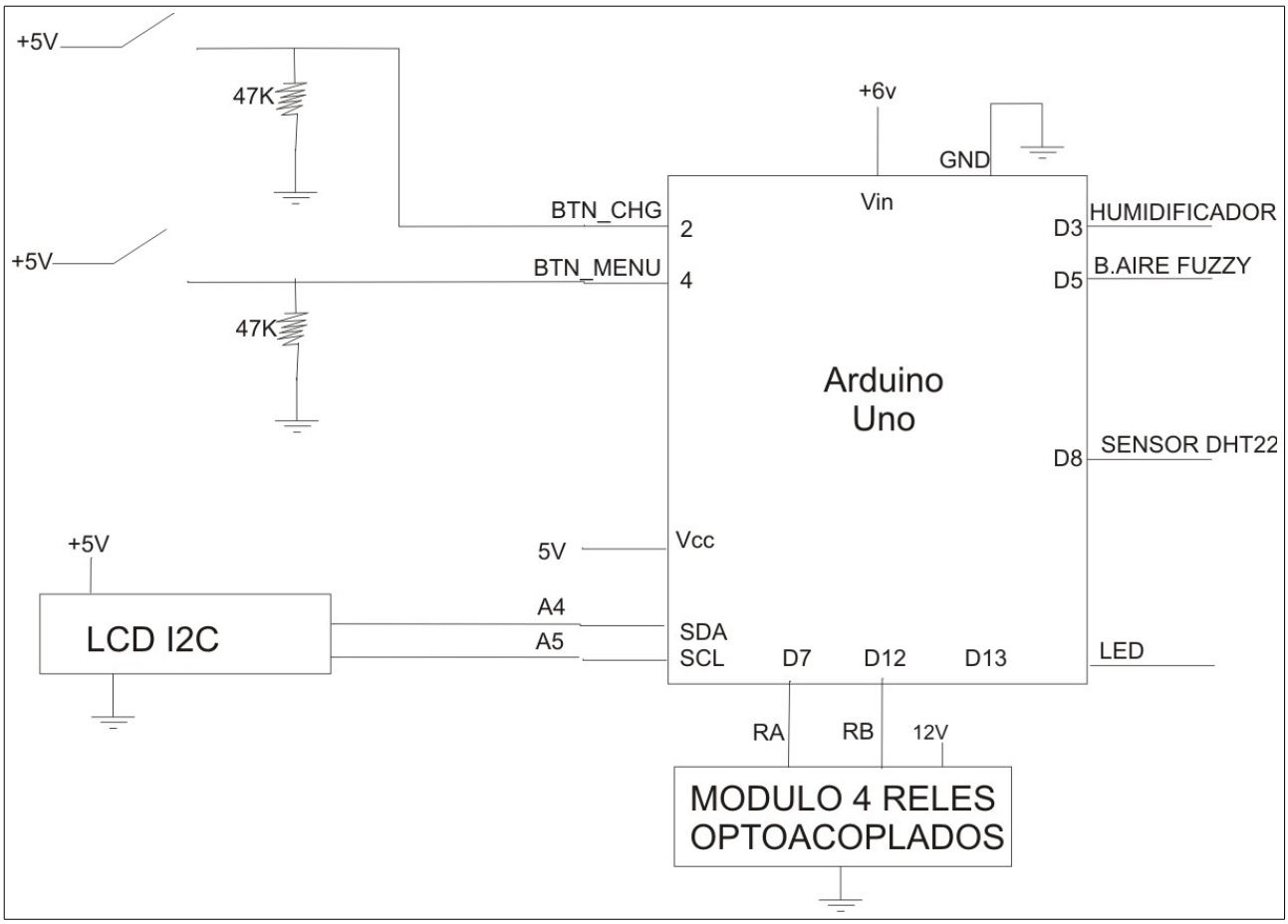
TUTOR: EDUARDO QUILES CUCARELLA

Valencia, Septiembre 2017

ÍNDICE ESQUEMAS DE CONEXIONADO

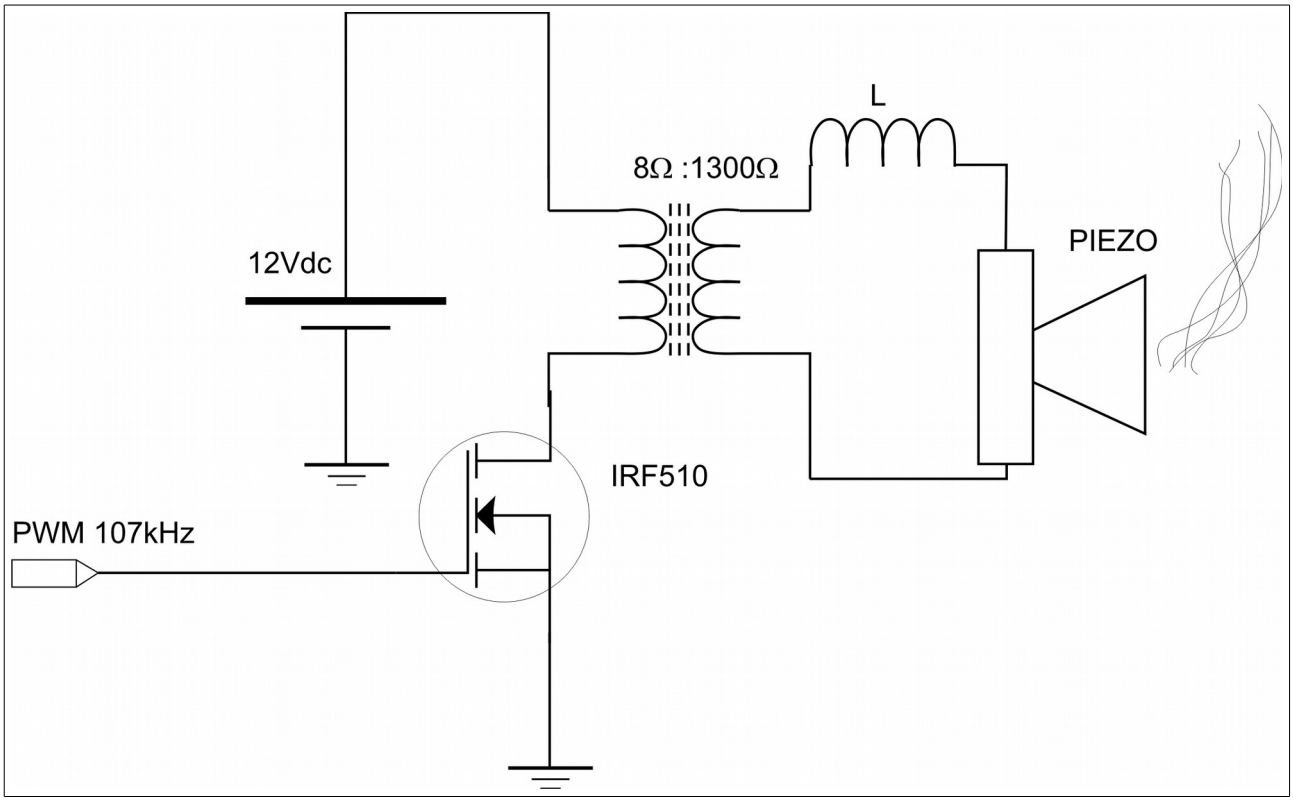
1. Esquemático de conexionado general de todos los elementos de control al Arduino

1.1 Esquemático del circuito de Arduino

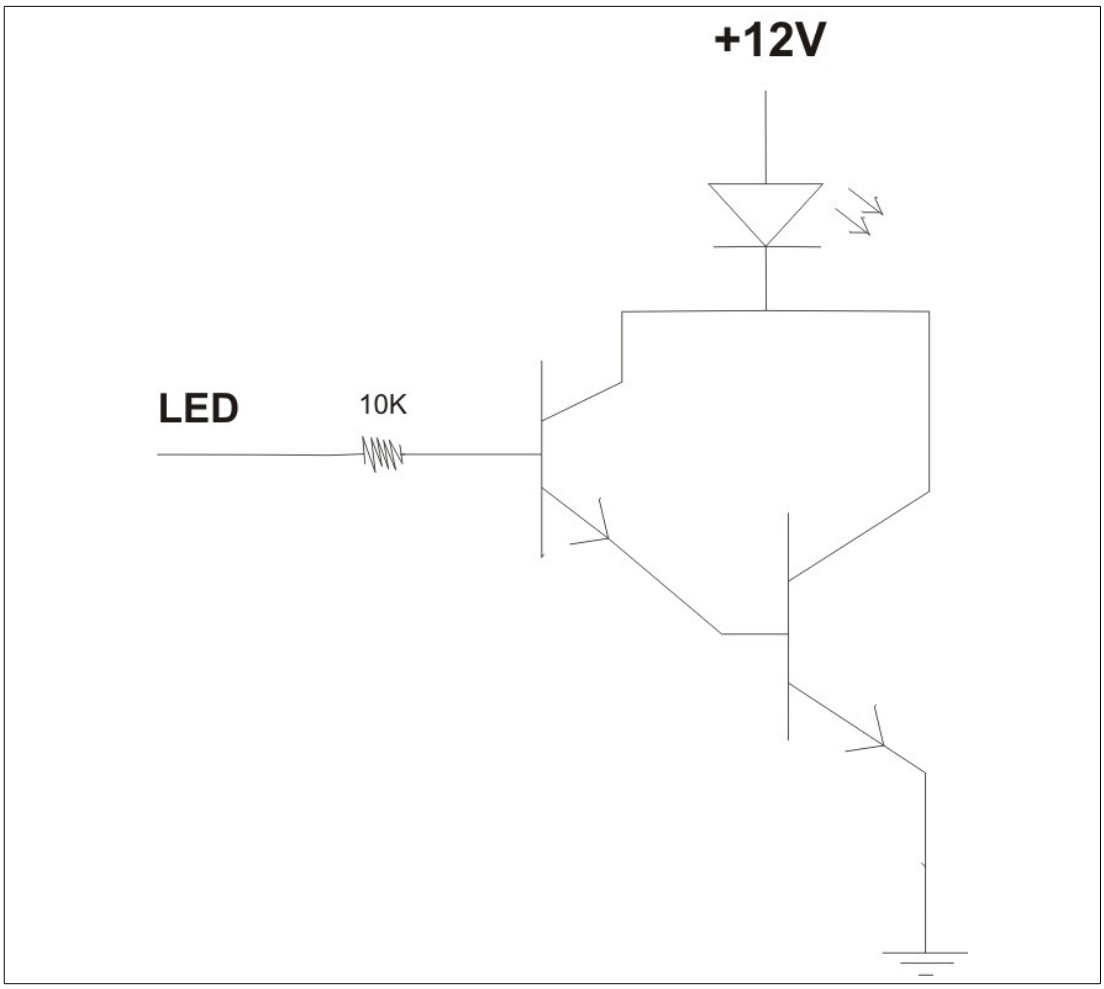


2. Esquemático de conexionado individual de los componentes

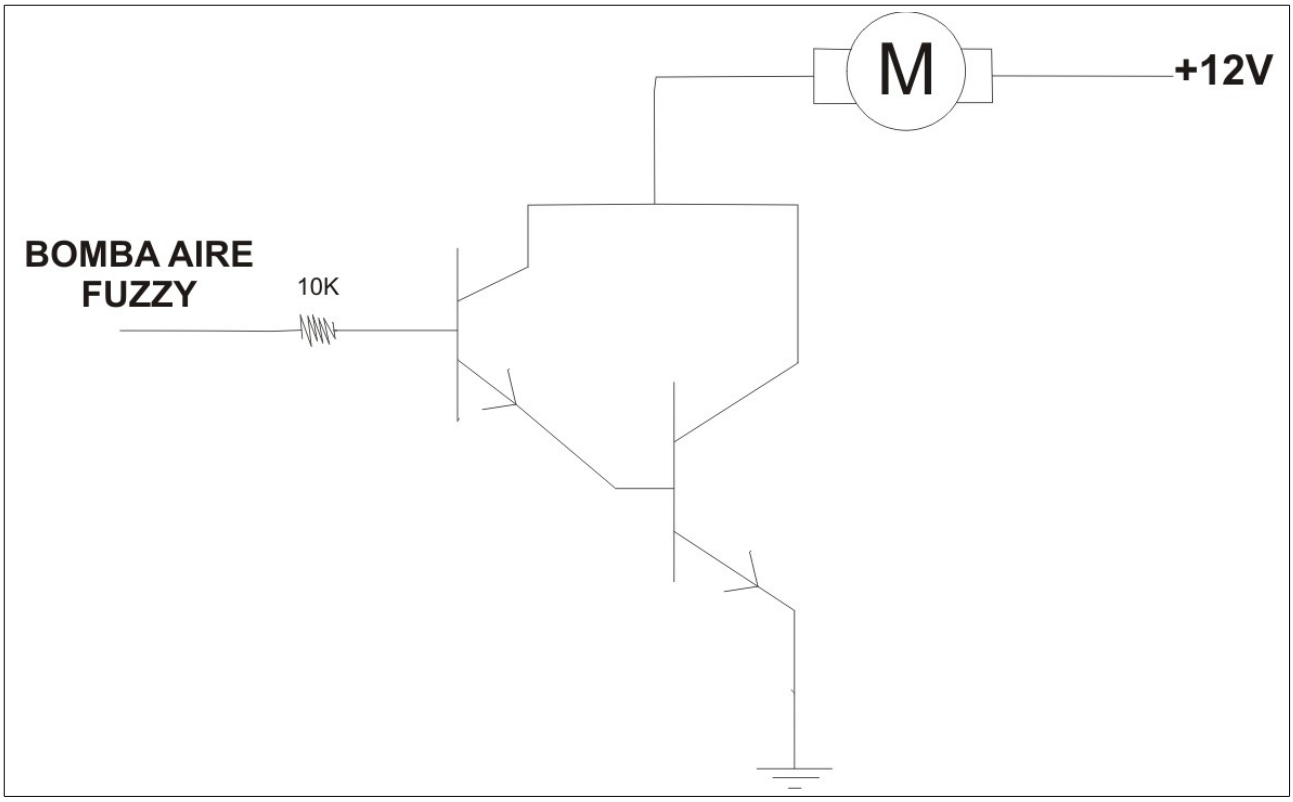
2.1 Esquemático del circuito del humidificador



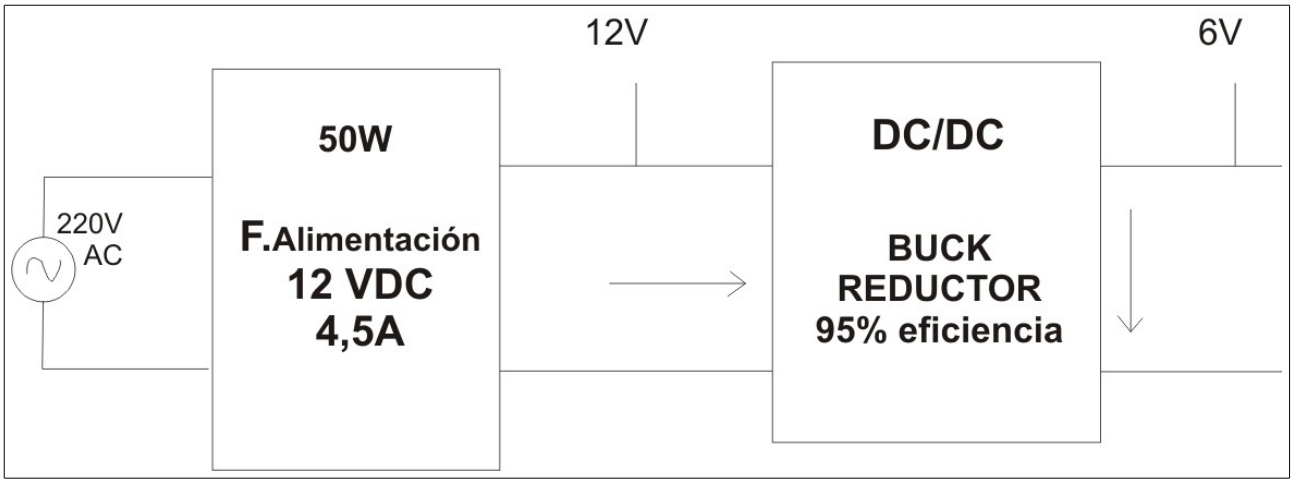
2.2 Esquemático del circuito del LED



2.3 Esquemático del circuito de la bomba de aire (control fuzzy)



2.4 Esquemático del circuito de alimentación del Arduino



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

Control climático con lógica fuzzy de un prototipo de invernadero doméstico para el cultivo de setas

Documento Nº 5: ANEXOS

AUTOR: JOSÉ CARLOS SERRANO DOMINGO

TUTOR: EDUARDO QUILES CUCARELLA

Valencia, Septiembre 2017

ÍNDICE ANEXOS

1. Código empleado

Código empleado en Maltab

2.2 Modelado teórico previo del invernadero necesario

Funciones implementadas en MATLAB:

```
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
DEW_POINT.M
////////////////////////////////////////////////////////////////////////////////

function resultado_dew_point = dew_point(celsius, humidity)

% DEW_POINT Computes the dew point.
   %   dew_point(celsius, humidity) returns temperature in celsius in which dew point is done
   %   entries in celsius and %.

A0= 373.15 ./(273.15 + celsius);
SUM = -7.90298 .* (A0-1);
SUM = SUM + (5.02808 .* log10(A0));
SUM = SUM + (-1.3816e-7 .* ((10.^(11.344 .*(1-1./A0)))-1));
SUM = SUM + (8.1328e-3 .* ((10.^(-3.49149 .*(A0-1)))-1));
SUM = SUM + log10(1013.246);
VP=(10.^(SUM-3)) .* humidity;
T = log(VP ./ 0.61078);
resultado_dew_point=(241.88 .* T) ./ (17.558-T);

end


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
FAST_DEW_POINT.M
////////////////////////////////////////////////////////////////////////////////

function resultado_dew_point = fast_dew_point(celsius, humidity)

% FAST_DEW_POINT Computes the fast dew point aproximation.
   %   fast_dew_point(celsius, humidity) returns temperature in celsius in which dew point is done
   %   entries in celsius and %.


a = 17.271;
b = 237.7;
temp = (a .* celsius) ./ (b + celsius) + log(humidity ./ 100);
resultado_dew_point = (b .* temp) ./ (a - temp);

end


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
HUMEDAD_ABSOLUTA.M
////////////////////////////////////////////////////////////////////////////////
```

```
function resultado = humedad_absoluta(celsius, humidity)

% HUMEDAD_ABSOLUTA Calcula la humedad absoluta del aire en g/m3.
   %   humedad_absoluta(celsius, humidity) returns la humedad absoluta del aire en g/m3.
   %   entries in celsius and %.

t=celsius;
f=humidity;

ai=7.45;
bi=235;
t=t.*1;
f=f.*1;
z1=(ai.*t)./(bi+t);
es=6.1.*exp(z1.*2.3025851);
e=es.*f./100;
z2=e./6.1;
z3=0.434292289.*log(z2);
dru=e.*100;
dru=floor(dru)./100;
tau=(235.*z3)./(7.45-z3).*100;
tau=floor(tau)./100;
feu=(216.7.*e)./(273.15+t).*100;
feu=round(feu)./100;

resultado = feu;

end
```

2.5 Modelado de nuestro sistema de control Fuzzy para el invernadero

2.5.2 Introducción de datos y reglas del modelo diseñado, en la aplicación Toolbox Fuzzy Logic de Matlab

Archivos .fis generados por la aplicación:

Intervalo 1

```
[System]
Name='intervalo_1_caudal_aire'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=5
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='delta_T'
Range=[-10 10]
NumMFs=3
MF1='mal':'trapmf',[-8.6 -5.4 -1.5 0]
MF2='equilibrio':'trimf',[-0.8 0 0.8]
MF3='bien':'trapmf',[0 1.5 5.4 8.6]

[Input2]
Name='delta_H'
Range=[-20 20]
NumMFs=3
MF1='mal':'trapmf',[-27.52 -17.28 -5 0]
MF2='equilibrio':'trimf',[-5 0 5]
MF3='bien':'trapmf',[0 5 17.28 27.52]

[Output1]
Name='caudal'
Range=[0 60]
NumMFs=3
MF1='poco':'trimf',[-24 0 24]
MF2='medio':'trimf',[6 30 54]
MF3='mucho':'trimf',[36 60 84]

[Rules]
1 1, 1 (1) : 1
2 1, 2 (1) : 1
3 1, 2 (1) : 1
3 2, 3 (1) : 1
3 3, 3 (1) : 1
```

Intervalo 2

```
[System]
Name='intervalo_2_caudal_aire'
Type='mamdani'
Version=2.0
NumInputs=1
NumOutputs=1
NumRules=3
```

```
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='delta_T'
Range=[-10 10]
NumMFs=3
MF1='mal':'trapmf',[-8.6 -5.4 -1.5 0]
MF2='equilibrio':'trimf',[-0.8 0 0.8]
MF3='mal2':'trapmf',[0 1.5 5.4 8.6]

[Input2]
Name='delta_H'
Range=[-20 20]
NumMFs=3
MF1='mal':'trapmf',[-27.52 -17.28 -5 0]
MF2='equilibrio':'trimf',[-5 0 5]
MF3='bien':'trapmf',[0 5 17.28 27.52]

[Output1]
Name='caudal'
Range=[0 60]
NumMFs=3
MF1='poco':'trimf',[-24 0 24]
MF2='medio':'trimf',[6 30 54]
MF3='mucho':'trimf',[36 60 84]

[Rules]
1 0, 1 (1) : 1
2 0, 2 (1) : 1
3 0, 1 (1) : 1
```

Intervalo 3

```
[System]
Name='intervalo_3_caudal_aire'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=3
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='delta_T'
Range=[-10 10]
NumMFs=3
MF1='bien':'trapmf',[-8.6 -5.4 -1.5 0]
MF2='equilibrio':'trimf',[-0.8 0 0.8]
MF3='mal':'trapmf',[0 0.8 5.4 8.6]

[Input2]
Name='delta_H'
Range=[-20 20]
NumMFs=3
MF1='mal':'trapmf',[-27.52 -17.28 -5 0]
MF2='equilibrio':'trimf',[-5 0 5]
MF3='bien':'trapmf',[0 5 17.28 27.52]

[Output1]
```

Name='caudal'
Range=[0 60]
NumMFs=3
MF1='poco':'trimf',[-24 0 24]
MF2='medio':'trimf',[6 30 54]
MF3='mucho':'trimf',[36 60 84]

[Rules]
1 2, 3 (1) : 1
2 2, 2 (1) : 1
3 0, 1 (1) : 1

Código empleado en C para Arduino

2.5 Modelado de nuestro sistema de control Fuzzy para el invernadero

2.5.4. Análisis de nuestro modelo fuzzy generado para Arduino en un entorno de programación en C

Código en C del programa empleado:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "fis_header.h"

//*********************************************************************
// Matlab .fis to arduino C converter v2.0.0.29032014
// - Karthik Nadig, USA
// Please report bugs to: karthiknadig@gmail.com
//*********************************************************************

#define XPASO 0.1
#define YPASO 0.1

#define XINICIO -20
#define XFINAL 20

#define YINICIO -10
#define YFINAL 10


/*INTERVALO 1*/
#define fis_gcI_i1 2  // Number of inputs to the fuzzy inference system
#define fis_gcO_i1 1  // Number of outputs to the fuzzy inference system
#define fis_gcR_i1 5  // Number of rules to the fuzzy inference system

/*INTERVALO 2*/
#define fis_gcI_i2 2  // Number of inputs to the fuzzy inference system
#define fis_gcO_i2 1  // Number of outputs to the fuzzy inference system
#define fis_gcR_i2 5  // Number of rules to the fuzzy inference system

/*INTERVALO 3*/
#define fis_gcI_i3 2  // Number of inputs to the fuzzy inference system
#define fis_gcO_i3 1  // Number of outputs to the fuzzy inference system
#define fis_gcR_i3 5  // Number of rules to the fuzzy inference system

#define MAXIMO_DE_RULES 5  // Number maximum of rules to the fuzzy inference system

#define min(a,b) (((a)<(b))?(a):(b))
#define max(a,b) (((a)>(b))?(a):(b))
//DEFAULT
int fis_gcI;  // Number of inputs to the fuzzy inference system
int fis_gcO;  // Number of outputs to the fuzzy inference system
int fis_gcR;  // Number of rules to the fuzzy inference system

FIS_TYPE g_fisInput_i1[fis_gcI_i1];
FIS_TYPE g_fisOutput_i1[fis_gcO_i1];

FIS_TYPE g_fisInput_i2[fis_gcI_i2];
FIS_TYPE g_fisOutput_i2[fis_gcO_i2];

FIS_TYPE g_fisInput_i3[fis_gcI_i3];
FIS_TYPE g_fisOutput_i3[fis_gcO_i3];

FIS_TYPE *g_fisInput[2];
```

```
FIS_TYPE *g_fisOutput[1];


//*********************************************************************
// Support functions for Fuzzy Inference System
//*********************************************************************
// Trapezoidal Member Function
FIS_TYPE fis_trapmf(FIS_TYPE x, FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2], d = p[3];
    FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ? 0 : ((c != d) ? ((d - x) / (d - c)) : 0)));
    FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ? 0 : ((a != b) ? ((x - a) / (b - a)) : 0)));
    return (FIS_TYPE) min(t1, t2);
}

// Triangular Member Function
FIS_TYPE fis_trimf(FIS_TYPE x, FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2];
    FIS_TYPE t1 = (x - a) / (b - a);
    FIS_TYPE t2 = (c - x) / (c - b);
    if ((a == b) && (b == c)) return (FIS_TYPE) (x == a);
    if (a == b) return (FIS_TYPE) (t2*(b <= x)*(x <= c));
    if (b == c) return (FIS_TYPE) (t1*(a <= x)*(x <= b));
    t1 = min(t1, t2);
    return (FIS_TYPE) max(t1, 0);
}

FIS_TYPE fis_min(FIS_TYPE a, FIS_TYPE b)
{
    return min(a, b);
}

FIS_TYPE fis_max(FIS_TYPE a, FIS_TYPE b)
{
    return max(a, b);
}

FIS_TYPE fis_array_operation(FIS_TYPE *array, int size, _FIS_ARR_OP pfnOp)
{
    int i;
    FIS_TYPE ret = 0;

    if (size == 0) return ret;
    if (size == 1) return array[0];

    ret = array[0];
    for (i = 1; i < size; i++)
    {
        ret = (*pfnOp)(ret, array[i]);
    }

    return ret;
}


//*********************************************************************
// Data for Fuzzy Inference System
//*********************************************************************
// Pointers to the implementations of member functions
_FIS_MF fis_gMF[] =
{
    fis_trapmf, fis_trimf
};

int fis_gIMFCount_i1[] = { 3, 3 }; // Count of member function for each Input
int fis_gOMFCount_i1[] = { 3 }; // Count of member function for each Output
```

```
int fis_gIMFCount_i2[] = { 3 }; // Count of member function for each Input
int fis_gOMFCount_i2[] = { 3 }; // Count of member function for each Output

int fis_gIMFCount_i3[] = { 3, 3 }; // Count of member function for each Input
int fis_gOMFCount_i3[] = { 3 }; // Count of member function for each Output

int *fis_gIMFCount;
int *fis_gOMFCount;

//############################################################################
##
//############################################################################
##
FIS_TYPE* fis_gMFI0Coeff[5]; // = { fis_gMFI0Coeff1_i1, fis_gMFI0Coeff2_i1, fis_gMFI0Coeff3_i1, NULL };
FIS_TYPE* fis_gMFI1Coeff[5]; // = { fis_gMFI1Coeff1_i1, fis_gMFI1Coeff2_i1, fis_gMFI1Coeff3_i1, NULL  };
FIS_TYPE** fis_gMFICoeff[2]; // = { fis_gMFI0Coeff_i1, fis_gMFI1Coeff_i1, NULL  };

FIS_TYPE* fis_gMFO0Coeff[3]; // = { fis_gMFO0Coeff1, fis_gMFO0Coeff2, fis_gMFO0Coeff3, NULL  };
FIS_TYPE** fis_gMFOCoeff[1]; // = { fis_gMFO0Coeff, NULL  };

int* fis_gMFI[2]; // = { fis_gMFI0_i1, fis_gMFI1_i1, NULL };

int* fis_gMFO[1]; // = { fis_gMFO0_i1, NULL };

FIS_TYPE fis_gRWeight[5]; // = { 1, 1, 1, 1, 1 , NULL};
int fis_gRType[5]; // = { 1, 1, 1, 1, 1 , NULL};

int* fis_gRI[5]; // = { fis_gRI0_i1, fis_gRI1_i1, fis_gRI2_i1, fis_gRI3_i1, fis_gRI4_i1, NULL  };

int* fis_gRO[5]; // = { fis_gRO0_i1, fis_gRO1_i1, fis_gRO2_i1, fis_gRO3_i1, fis_gRO4_i1, NULL  };

/*
//############################################################################
##
//############################################################################
##


FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1_i2, fis_gMFI0Coeff2_i2, fis_gMFI0Coeff3_i2, NULL  };
FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff , NULL  };

FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1_i2, fis_gMFO0Coeff2_i2, fis_gMFO0Coeff3_i2 , NULL  };
FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff , NULL  };

int* fis_gMFI[] = { fis_gMFI0_i2, NULL  };

int* fis_gMFO[] = { fis_gMFO0_i2, NULL  };

FIS_TYPE fis_gRWeight[] = { 1, 1, 1 , NULL};
int fis_gRType[] = { 1, 1, 1 , NULL};

int* fis_gRI[] = { fis_gRI0_i2, fis_gRI1_i2, fis_gRI2_i2 , NULL  };

int* fis_gRO[] = { fis_gRO0_i2, fis_gRO1_i2, fis_gRO2_i2 , NULL  };

*/
//############################################################################
##
//############################################################################
##


/*****************************************************************/
/*INTERVALO 1*/
/*****************************************************************/

// Coefficients for the Input Member Functions
FIS_TYPE fis_gMFI0Coeff1_i1[] = { -8.6, -5.4, -1.5, 0 };
```

```
FIS_TYPE fis_gMFI0Coeff2_i1[] = { -0.8, 0, 0.8 };
FIS_TYPE fis_gMFI0Coeff3_i1[] = { 0, 1.5, 5.4, 8.6 };
//FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1_i1, fis_gMFI0Coeff2_i1, fis_gMFI0Coeff3_i1, NULL };
FIS_TYPE fis_gMFI1Coeff1_i1[] = { -27.52, -17.28, -5, 0 };
FIS_TYPE fis_gMFI1Coeff2_i1[] = { -5, 0, 5 };
FIS_TYPE fis_gMFI1Coeff3_i1[] = { 0, 5, 17.28, 27.52 };
//FIS_TYPE* fis_gMFI1Coeff[] = { fis_gMFI1Coeff1_i1, fis_gMFI1Coeff2_i1, fis_gMFI1Coeff3_i1, NULL  };
//FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff_i1, fis_gMFI1Coeff_i1, NULL  };

// Coefficients for the Output Member Functions
FIS_TYPE fis_gMFO0Coeff1_i1[] = { -24, 0, 24 };
FIS_TYPE fis_gMFO0Coeff2_i1[] = { 6, 30, 54 };
FIS_TYPE fis_gMFO0Coeff3_i1[] = { 36, 60, 84 };
//FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1, fis_gMFO0Coeff2, fis_gMFO0Coeff3, NULL  };
//FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff, NULL  };

// Input membership function set
int fis_gMFI0_i1[] = { 0, 1, 0 }; //trapecio, triangulo, trapecio
int fis_gMFI1_i1[] = { 0, 1, 0 }; //trapecio, triangulo, trapecio
//int* fis_gMFI[] = { fis_gMFI0_i1, fis_gMFI1_i1, NULL };

// Output membership function set
int fis_gMFO0_i1[] = { 1, 1, 1 }; //triangulo, triangulo, triangulo
//int* fis_gMFO[] = { fis_gMFO0_i1, NULL };

// Rule Weights
//FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1 , NULL};

// Rule Type
//int fis_gRType[] = { 1, 1, 1, 1, 1 , NULL};

// Rule Inputs
int fis_gRI0_i1[] = { 1, 1 };
int fis_gRI1_i1[] = { 2, 1 };
int fis_gRI2_i1[] = { 3, 1 };
int fis_gRI3_i1[] = { 3, 2 };
int fis_gRI4_i1[] = { 3, 3 };
//int* fis_gRI[] = { fis_gRI0_i1, fis_gRI1_i1, fis_gRI2_i1, fis_gRI3_i1, fis_gRI4_i1, NULL  };

// Rule Outputs
int fis_gRO0_i1[] = { 1 };
int fis_gRO1_i1[] = { 2 };
int fis_gRO2_i1[] = { 2 };
int fis_gRO3_i1[] = { 3 };
int fis_gRO4_i1[] = { 3 };
//int* fis_gRO[] = { fis_gRO0_i1, fis_gRO1_i1, fis_gRO2_i1, fis_gRO3_i1, fis_gRO4_i1, NULL  };

/**********************************************************************/
/* FIN INTERVALO 1*/
/**********************************************************************/

/**********************************************************************/
/*INTERVALO 2*/
/**********************************************************************/
// Coefficients for the Input Member Functions
FIS_TYPE fis_gMFI0Coeff1_i2[] = { -8.6, -5.4, -1.5, 0 };
FIS_TYPE fis_gMFI0Coeff2_i2[] = { -0.8, 0, 0.8 };
FIS_TYPE fis_gMFI0Coeff3_i2[] = { 0, 1.5, 5.4, 8.6 };
//FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1_i2, fis_gMFI0Coeff2_i2, fis_gMFI0Coeff3_i2, NULL  };
//FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff , NULL  };

FIS_TYPE fis_gMFI1Coeff1_i2[] = { -27.52, -17.28, -5, 0 };
FIS_TYPE fis_gMFI1Coeff2_i2[] = { -5, 0, 5 };
FIS_TYPE fis_gMFI1Coeff3_i2[] = { 0, 5, 17.28, 27.52 };


// Coefficients for the Output Member Functions
```

```
FIS_TYPE fis_gMFO0Coeff1_i2[] = { -24, 0, 24 };
FIS_TYPE fis_gMFO0Coeff2_i2[] = { 6, 30, 54 };
FIS_TYPE fis_gMFO0Coeff3_i2[] = { 36, 60, 84 };
//FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1_i2, fis_gMFO0Coeff2_i2, fis_gMFO0Coeff3_i2 , NULL  };
//FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff , NULL  };

// Input membership function set
int fis_gMFI0_i2[] = { 0, 1, 0 };
int fis_gMFI1_i2[] = { 0, 1, 0 }; /////
//int* fis_gMFI[] = { fis_gMFI0_i2, NULL  };

// Output membership function set
int fis_gMFO0_i2[] = { 1, 1, 1 };
//int* fis_gMFO[] = { fis_gMFO0_i2, NULL  };

// Rule Weights
//FIS_TYPE fis_gRWeight[] = { 1, 1, 1 , NULL};

// Rule Type
//int fis_gRType[] = { 1, 1, 1 , NULL};

// Rule Inputs
int fis_gRI0_i2[] = { 1, 0 };
int fis_gRI1_i2[] = { 2, 0 };
int fis_gRI2_i2[] = { 3, 0 };
int fis_gRI3_i2[] = { 0, 0 };
int fis_gRI4_i2[] = { 0, 0 };
//int* fis_gRI[] = { fis_gRI0_i2, fis_gRI1_i2, fis_gRI2_i2 , NULL  };

// Rule Outputs
int fis_gRO0_i2[] = { 1 };
int fis_gRO1_i2[] = { 2 };
int fis_gRO2_i2[] = { 1 };
int fis_gRO3_i2[] = { 0 };
int fis_gRO4_i2[] = { 0 };
//int* fis_gRO[] = { fis_gRO0_i2, fis_gRO1_i2, fis_gRO2_i2 , NULL  };

/**************************************************************************/
/* FIN INTERVALO 2*/
/**************************************************************************/

/**************************************************************************/
/*INTERVALO 3*/
/**************************************************************************/
// Coefficients for the Input Member Functions
FIS_TYPE fis_gMFI0Coeff1_i3[] = { -8.6, -5.4, -1.5, 0 };
FIS_TYPE fis_gMFI0Coeff2_i3[] = { -0.8, 0, 0.8 };
FIS_TYPE fis_gMFI0Coeff3_i3[] = { 0, 1.5, 5.4, 8.6 };
//FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1_i2, fis_gMFI0Coeff2_i2, fis_gMFI0Coeff3_i2, NULL  };
//FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff , NULL  };

FIS_TYPE fis_gMFI1Coeff1_i3[] = { -27.52, -17.28, -5, 0 };
FIS_TYPE fis_gMFI1Coeff2_i3[] = { -5, 0, 5 };
FIS_TYPE fis_gMFI1Coeff3_i3[] = { 0, 5, 17.28, 27.52 };


// Coefficients for the Output Member Functions
FIS_TYPE fis_gMFO0Coeff1_i3[] = { -24, 0, 24 };
FIS_TYPE fis_gMFO0Coeff2_i3[] = { 6, 30, 54 };
FIS_TYPE fis_gMFO0Coeff3_i3[] = { 36, 60, 84 };
//FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1_i2, fis_gMFO0Coeff2_i2, fis_gMFO0Coeff3_i2 , NULL  };
//FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff , NULL  };

// Input membership function set
int fis_gMFI0_i3[] = { 0, 1, 0 };
int fis_gMFI1_i3[] = { 0, 1, 0 }; /////
//int* fis_gMFI[] = { fis_gMFI0_i2, NULL  };

// Output membership function set
```

```
int fis_gMFO0_i3[] = { 1, 1, 1 };
//int* fis_gMFO[] = { fis_gMFO0_i2, NULL  };

// Rule Weights
//FIS_TYPE fis_gRWeight[] = { 1, 1, 1 , NULL};

// Rule Type
//int fis_gRType[] = { 1, 1, 1 , NULL};

// Rule Inputs
int fis_gRI0_i3[] = { 1, 2 };
int fis_gRI1_i3[] = { 2, 2 };
int fis_gRI2_i3[] = { 3, 0 };
int fis_gRI3_i3[] = { 0, 0 };
int fis_gRI4_i3[] = { 0, 0 };
//int* fis_gRI[] = { fis_gRI0_i2, fis_gRI1_i2, fis_gRI2_i2 , NULL  };

// Rule Outputs
int fis_gRO0_i3[] = { 3 };
int fis_gRO1_i3[] = { 2 };
int fis_gRO2_i3[] = { 1 };
int fis_gRO3_i3[] = { 0 };
int fis_gRO4_i3[] = { 0 };
//int* fis_gRO[] = { fis_gRO0_i2, fis_gRO1_i2, fis_gRO2_i2 , NULL  };

/**********************************************************************/
/* FIN INTERVALO 3*/
/**********************************************************************/



// Input range Min
FIS_TYPE fis_gIMin[] = { -5, -16 };

// Input range Max
FIS_TYPE fis_gIMax[] = { 5, 16 };

// Output range Min
FIS_TYPE fis_gOMin[] = { 0 };

// Output range Max
FIS_TYPE fis_gOMax[] = { 60 };

//**************************************************************************
// Data dependent support functions for Fuzzy Inference System
//**************************************************************************
FIS_TYPE fis_MF_out(FIS_TYPE** fuzzyRuleSet, FIS_TYPE x, int o)
{
   FIS_TYPE mfOut;
   int r;

   for (r = 0; r < fis_gcR; ++r)
   {
      int index = fis_gRO[r][o];
      if (index > 0)
      {
         index = index - 1;
         mfOut = (fis_gMF[fis_gMFO[o][index]])(x, fis_gMFOCoeff[o][index]);
      }
      else if (index < 0)
      {
         index = -index - 1;
         mfOut = 1 - (fis_gMF[fis_gMFO[o][index]])(x, fis_gMFOCoeff[o][index]);
      }
      else
      {
         mfOut = 0;
```

```
        }

        fuzzyRuleSet[0][r] = fis_min(mfOut, fuzzyRuleSet[1][r]);
    }
    return fis_array_operation(fuzzyRuleSet[0], fis_gcR, fis_max);
}

FIS_TYPE fis_defuzz_centroid(FIS_TYPE** fuzzyRuleSet, int o)
{
    FIS_TYPE step = (fis_gOMax[o] - fis_gOMin[o]) / (FIS_RESOLUSION - 1);
    FIS_TYPE area = 0;
    FIS_TYPE momentum = 0;
    FIS_TYPE dist, slice;
    int i;

    // calculate the area under the curve formed by the MF outputs
    for (i = 0; i < FIS_RESOLUSION; ++i){
        dist = fis_gOMin[o] + (step * i);
        slice = step * fis_MF_out(fuzzyRuleSet, dist, o);
        area += slice;
        momentum += slice*dist;
    }

    return ((area == 0) ? ((fis_gOMax[o] + fis_gOMin[o]) / 2) : (momentum / area));
}

//************************************************************************
// Fuzzy Inference System
//************************************************************************
void fis_evaluate()
{

    FIS_TYPE fuzzyInput0[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput1[] = { 0, 0, 0 };
    FIS_TYPE* fuzzyInput[] = {fuzzyInput0, fuzzyInput1};
    //FIS_TYPE fuzzyOutput0[] = { 0, 0, 0 };
    //FIS_TYPE* fuzzyOutput[] = { fuzzyOutput0};
    FIS_TYPE fuzzyRules[MAXIMO_DE_RULES] = { 0 };
    FIS_TYPE fuzzyFires[MAXIMO_DE_RULES] = { 0 };
    FIS_TYPE* fuzzyRuleSet[] = { fuzzyRules, fuzzyFires };
    FIS_TYPE sW = 0;


FIS_TYPE g_fisInput_local[] = { *g_fisInput[0], *g_fisInput[1] };
FIS_TYPE g_fisOutput_local[] = { *g_fisOutput[0] };

    int i, j, r, o;
    // Transforming input to fuzzy Input
    for (i = 0; i < fis_gcI; ++i)
    {
        for (j = 0; j < fis_gIMFCount[i]; ++j)
        {
            fuzzyInput[i][j] =
                (fis_gMF[fis_gMFI[i][j]])(g_fisInput_local[i], fis_gMFICoeff[i][j]);
        }
    }

    int index = 0;
    for (r = 0; r < fis_gcR; ++r)
    {
        if (fis_gRType[r] == 1)
        {
            fuzzyFires[r] = FIS_MAX;
            for (i = 0; i < fis_gcI; ++i)
            {
                index = fis_gRI[r][i];
                if (index > 0)
                    fuzzyFires[r] = fis_min(fuzzyFires[r], fuzzyInput[i][index - 1]);
                else if (index < 0)
```

```
                fuzzyFires[r] = fis_min(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
            else
                fuzzyFires[r] = fis_min(fuzzyFires[r], 1);

        }
      }
      else
      {
        fuzzyFires[r] = FIS_MIN;
        for (i = 0; i < fis_gcI; ++i)
        {
            index = fis_gRI[r][i];
            if (index > 0)
                fuzzyFires[r] = fis_max(fuzzyFires[r], fuzzyInput[i][index - 1]);
            else if (index < 0)
                fuzzyFires[r] = fis_max(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
            else
                fuzzyFires[r] = fis_max(fuzzyFires[r], 0);
        }
      }

      fuzzyFires[r] = fis_gRWeight[r] * fuzzyFires[r];
      sW += fuzzyFires[r];
   }

   if (sW == 0)
   {
      for (o = 0; o < fis_gcO; ++o)
      {
        g_fisOutput_local[o] = ((fis_gOMax[o] + fis_gOMin[o]) / 2);
      }
   }
   else
   {
      for (o = 0; o < fis_gcO; ++o)
      {
        g_fisOutput_local[o] = fis_defuzz_centroid(fuzzyRuleSet, o);
        *g_fisOutput[o]=g_fisOutput_local[o];
      }
   }
}



int main()
{
   int intervalo = 0;

   float delta_temp = 0;
   float delta_humi = 0;

   float eje_y;
   float resolucion_eje_y=YPASO;
   float eje_x;
   float resolucion_eje_x=XPASO;
   FILE * fp;

   while (intervalo != 4444){

        printf("\nINTERVALO: ");
        scanf("%d", &intervalo);
        if (intervalo==4444){
           //salgo
        }else if(intervalo==1){

/*INTERVALO 1*/
fis_gcI=fis_gcI_i1;  // Number of inputs to the fuzzy inference system
```

```
fis_gcO=fis_gcO_i1;  // Number of outputs to the fuzzy inference system
fis_gcR=fis_gcR_i1;  // Number of rules to the fuzzy inference system


g_fisInput[0] = &g_fisInput_i1[0];
g_fisInput[1] = &g_fisInput_i1[1];
g_fisOutput[0] = &g_fisOutput_i1[0];
fis_gIMFCount=&fis_gIMFCount_i1[0];
fis_gOMFCount=&fis_gOMFCount_i1[0];


fis_gMFI0Coeff[0] = &fis_gMFI0Coeff1_i1[0];
fis_gMFI0Coeff[1] = &fis_gMFI0Coeff2_i1[0];
fis_gMFI0Coeff[2] = &fis_gMFI0Coeff3_i1[0];
fis_gMFI1Coeff[0] = &fis_gMFI1Coeff1_i1[0];
fis_gMFI1Coeff[1] = &fis_gMFI1Coeff2_i1[0];
fis_gMFI1Coeff[2] = &fis_gMFI1Coeff3_i1[0];
fis_gMFICoeff[0] = &fis_gMFI0Coeff[0];
fis_gMFICoeff[1] = &fis_gMFI1Coeff[0];

fis_gMFO0Coeff[0] = &fis_gMFO0Coeff1_i1[0];
fis_gMFO0Coeff[1] = &fis_gMFO0Coeff2_i1[0];
fis_gMFO0Coeff[2] = &fis_gMFO0Coeff3_i1[0];
fis_gMFOCoeff[0] = &fis_gMFO0Coeff[0];

fis_gMFI[0] = &fis_gMFI0_i1[0];
fis_gMFI[1] = &fis_gMFI1_i1[0];

fis_gMFO[0] = &fis_gMFO0_i1[0];

fis_gRWeight[0] = 1;
fis_gRWeight[1] = 1;
fis_gRWeight[2] = 1;
fis_gRWeight[3] = 1;
fis_gRWeight[4] = 1;
fis_gRType[0] = 1;
fis_gRType[1] = 1;
fis_gRType[2] = 1;
fis_gRType[3] = 1;
fis_gRType[4] = 1;

fis_gRI[0] = &fis_gRI0_i1[0];
fis_gRI[1] = &fis_gRI1_i1[0];
fis_gRI[2] = &fis_gRI2_i1[0];
fis_gRI[3] = &fis_gRI3_i1[0];
fis_gRI[4] = &fis_gRI4_i1[0];

fis_gRO[0] = &fis_gRO0_i1[0];
fis_gRO[1] = &fis_gRO1_i1[0];
fis_gRO[2] = &fis_gRO2_i1[0];
fis_gRO[3] = &fis_gRO3_i1[0];
fis_gRO[4] = &fis_gRO4_i1[0];

            printf("\nDelta de Temperatura:");
            scanf("%f", &delta_temp);
            // Read Input: delta_T
               g_fisInput_i1[0] = delta_temp;
            printf("\nDelta de Humedad:");
            scanf("%f", &delta_humi);
            // Read Input: delta_H
               g_fisInput_i1[1] = delta_humi;

            //float caudal_output = 0;
               g_fisOutput_i1[0] = 44;
               fis_evaluate();
            // Set output value: caudal
               printf("\n\nLa salida vale:\n");
               printf("---> %f\n",g_fisOutput_i1[0]);
```

```
            fp = fopen ("my_fuzzy_file_intervalo_1.txt", "w");
                    fprintf(fp, "# Created by Jose K. \n");
                    fprintf(fp, "# name: caudal_aire_1 \n");
                    fprintf(fp, "# type: matrix \n");
                    fprintf(fp, "# rows: %1.0f\n", (YFINAL - YINICIO)/YPASO);  //(Final - Inicial) / Paso
                    fprintf(fp, "# columns: %1.0f\n", (XFINAL - XINICIO)/XPASO);  //(Final - Inicial) / Paso

            for(eje_y=YINICIO;eje_y<YFINAL;eje_y=resolucion_eje_y+eje_y){
               g_fisInput_i1[0] = eje_y; //delta_temp;
                  for(eje_x=XINICIO;eje_x<XFINAL;eje_x=resolucion_eje_x+eje_x){
                     g_fisInput_i1[1] = eje_x; //delta_humi;
                     fis_evaluate();
                     fprintf(fp, " %f", g_fisOutput_i1[0]);
                  }
                     fprintf(fp, "\n");
            }
            fclose(fp);



        }else if(intervalo==2){

/*INTERVALO 2*/
fis_gcI=fis_gcI_i2;  // Number of inputs to the fuzzy inference system
fis_gcO=fis_gcO_i2;  // Number of outputs to the fuzzy inference system
fis_gcR=fis_gcR_i2;  // Number of rules to the fuzzy inference system

g_fisInput[0] = &g_fisInput_i2[0];
g_fisInput[1] = &g_fisInput_i2[1];
g_fisOutput[0] = &g_fisOutput_i2[0];
fis_gIMFCount=&fis_gIMFCount_i2[0];
fis_gOMFCount=&fis_gOMFCount_i2[0];

fis_gMFI0Coeff[0] = &fis_gMFI0Coeff1_i2[0];
fis_gMFI0Coeff[1] = &fis_gMFI0Coeff2_i2[0];
fis_gMFI0Coeff[2] = &fis_gMFI0Coeff3_i2[0];
fis_gMFI1Coeff[0] = &fis_gMFI1Coeff1_i2[0];
fis_gMFI1Coeff[1] = &fis_gMFI1Coeff2_i2[0];
fis_gMFI1Coeff[2] = &fis_gMFI1Coeff3_i2[0];
fis_gMFICoeff[0] = &fis_gMFI0Coeff[0];
fis_gMFICoeff[1] = &fis_gMFI1Coeff[0];

fis_gMFO0Coeff[0] = &fis_gMFO0Coeff1_i2[0];
fis_gMFO0Coeff[1] = &fis_gMFO0Coeff2_i2[0];
fis_gMFO0Coeff[2] = &fis_gMFO0Coeff3_i2[0];
fis_gMFOCoeff[0] = &fis_gMFO0Coeff[0];

fis_gMFI[0] = &fis_gMFI0_i2[0];
fis_gMFI[1] = &fis_gMFI1_i2[0];

fis_gMFO[0] = &fis_gMFO0_i2[0];

fis_gRWeight[0] = 1;
fis_gRWeight[1] = 1;
fis_gRWeight[2] = 1;
fis_gRWeight[3] = 1;
fis_gRWeight[4] = 1;
fis_gRType[0] = 1;
fis_gRType[1] = 1;
fis_gRType[2] = 1;
fis_gRType[3] = 1;
fis_gRType[4] = 1;

fis_gRI[0] = &fis_gRI0_i2[0];
fis_gRI[1] = &fis_gRI1_i2[0];
fis_gRI[2] = &fis_gRI2_i2[0];
fis_gRI[3] = &fis_gRI3_i2[0];
```

```
fis_gRI[4] = &fis_gRI4_i2[0];

fis_gRO[0] = &fis_gRO0_i2[0];
fis_gRO[1] = &fis_gRO1_i2[0];
fis_gRO[2] = &fis_gRO2_i2[0];
fis_gRO[3] = &fis_gRO3_i2[0];
fis_gRO[4] = &fis_gRO4_i2[0];

            printf("\nDelta de Temperatura:");
            scanf("%f", &delta_temp);
            // Read Input: delta_T
               g_fisInput_i2[0] = delta_temp;
            printf("\nDelta de Humedad:");
            scanf("%f", &delta_humi);
            // Read Input: delta_H
               g_fisInput_i2[1] = delta_humi;

            //float caudal_output = 0;
               g_fisOutput_i2[0] = 44;
               fis_evaluate();
            // Set output value: caudal
               printf("\n\nLa salida vale:\n");
               printf("---> %f\n",g_fisOutput_i2[0]);

               fp = fopen ("my_fuzzy_file_intervalo_2.txt", "w");
                        fprintf(fp, "# Created by Jose K. \n");
                        fprintf(fp, "# name: caudal_aire_2 \n");
                        fprintf(fp, "# type: matrix \n");
                        fprintf(fp, "# rows: %1.0f\n", (YFINAL - YINICIO)/YPASO);  //(Final - Inicial) / Paso
                        fprintf(fp, "# columns: %1.0f\n", (XFINAL - XINICIO)/XPASO);  //(Final - Inicial) / Paso

               for(eje_y=YINICIO;eje_y<YFINAL;eje_y=resolucion_eje_y+eje_y){
                  g_fisInput_i2[0] = eje_y; //delta_temp;
                     for(eje_x=XINICIO;eje_x<XFINAL;eje_x=resolucion_eje_x+eje_x){
                        g_fisInput_i2[1] = eje_x; //delta_humi;
                        fis_evaluate();
                        fprintf(fp, " %f", g_fisOutput_i2[0]);
                     }
                        fprintf(fp, "\n");
               }
               fclose(fp);



         }else if(intervalo==3){
/*INTERVALO 3*/
fis_gcI=fis_gcI_i3;  // Number of inputs to the fuzzy inference system
fis_gcO=fis_gcO_i3;  // Number of outputs to the fuzzy inference system
fis_gcR=fis_gcR_i3;  // Number of rules to the fuzzy inference system


g_fisInput[0] = &g_fisInput_i3[0];
g_fisInput[1] = &g_fisInput_i3[1];
g_fisOutput[0] = &g_fisOutput_i3[0];
fis_gIMFCount=&fis_gIMFCount_i3[0];
fis_gOMFCount=&fis_gOMFCount_i3[0];


fis_gMFI0Coeff[0] = &fis_gMFI0Coeff1_i3[0];
fis_gMFI0Coeff[1] = &fis_gMFI0Coeff2_i3[0];
fis_gMFI0Coeff[2] = &fis_gMFI0Coeff3_i3[0];
fis_gMFI1Coeff[0] = &fis_gMFI1Coeff1_i3[0];
fis_gMFI1Coeff[1] = &fis_gMFI1Coeff2_i3[0];
fis_gMFI1Coeff[2] = &fis_gMFI1Coeff3_i3[0];
fis_gMFICoeff[0] = &fis_gMFI0Coeff[0];
fis_gMFICoeff[1] = &fis_gMFI1Coeff[0];

fis_gMFO0Coeff[0] = &fis_gMFO0Coeff1_i3[0];
fis_gMFO0Coeff[1] = &fis_gMFO0Coeff2_i3[0];
```

```
fis_gMFO0Coeff[2] = &fis_gMFO0Coeff3_i3[0];
fis_gMFOCoeff[0] = &fis_gMFO0Coeff[0];

fis_gMFI[0] = &fis_gMFI0_i3[0];
fis_gMFI[1] = &fis_gMFI1_i3[0];

fis_gMFO[0] = &fis_gMFO0_i3[0];

fis_gRWeight[0] = 1;
fis_gRWeight[1] = 1;
fis_gRWeight[2] = 1;
fis_gRWeight[3] = 1;
fis_gRWeight[4] = 1;
fis_gRType[0] = 1;
fis_gRType[1] = 1;
fis_gRType[2] = 1;
fis_gRType[3] = 1;
fis_gRType[4] = 1;

fis_gRI[0] = &fis_gRI0_i3[0];
fis_gRI[1] = &fis_gRI1_i3[0];
fis_gRI[2] = &fis_gRI2_i3[0];
fis_gRI[3] = &fis_gRI3_i3[0];
fis_gRI[4] = &fis_gRI4_i3[0];

fis_gRO[0] = &fis_gRO0_i3[0];
fis_gRO[1] = &fis_gRO1_i3[0];
fis_gRO[2] = &fis_gRO2_i3[0];
fis_gRO[3] = &fis_gRO3_i3[0];
fis_gRO[4] = &fis_gRO4_i3[0];

            printf("\nDelta de Temperatura:");
            scanf("%f", &delta_temp);
            // Read Input: delta_T
               g_fisInput_i3[0] = delta_temp;
            printf("\nDelta de Humedad:");
            scanf("%f", &delta_humi);
            // Read Input: delta_H
               g_fisInput_i3[1] = delta_humi;

            //float caudal_output = 0;
               g_fisOutput_i3[0] = 44;
               fis_evaluate();
            // Set output value: caudal
               printf("\n\nLa salida vale:\n");
               printf("---> %f\n",g_fisOutput_i3[0]);

               fp = fopen ("my_fuzzy_file_intervalo_3.txt", "w");
                        fprintf(fp, "# Created by Jose K. \n");
                        fprintf(fp, "# name: caudal_aire_3 \n");
                        fprintf(fp, "# type: matrix \n");
                        fprintf(fp, "# rows: %1.0f\n", (YFINAL - YINICIO)/YPASO);  //(Final - Inicial) / Paso
                        fprintf(fp, "# columns: %1.0f\n", (XFINAL - XINICIO)/XPASO);  //(Final - Inicial) / Paso

               for(eje_y=YINICIO;eje_y<YFINAL;eje_y=resolucion_eje_y+eje_y){
                  g_fisInput_i3[0] = eje_y; //delta_temp;
                     for(eje_x=XINICIO;eje_x<XFINAL;eje_x=resolucion_eje_x+eje_x){
                        g_fisInput_i3[1] = eje_x; //delta_humi;
                        fis_evaluate();
                        fprintf(fp, " %f", g_fisOutput_i3[0]);
                     }
                        fprintf(fp, "\n");
               }
               fclose(fp);


         }//if else if intervalos
```

```
    }//while

    return 0;
}
```

2.6 Código desarrollo del programa en Arduino para implementar el control climático en el invernadero fabricado

2.6.1 Antecedentes y especificaciones para el desarrollo del programa

Declaración de condiciones iniciales, flags, librerias y variables globales en general

```
/*-----( Cargamos las librerias necesarias )-----*/
#include <DHT.h>   //para el sensor de temperatura y humedad
#include <Wire.h>
#include <LiquidCrystal_I2C.h>  //para el LCD
#include "./fis_header.h"   //algunas definiciones para el sistema Fuzzy


/*SISTEMA FUZZY INTERVALO 1*/
#define fis_gcI_i1 2  // Number of inputs to the fuzzy inference system
#define fis_gcO_i1 1  // Number of outputs to the fuzzy inference system
#define fis_gcR_i1 5  // Number of rules to the fuzzy inference system

/*SISTEMA FUZZY INTERVALO 2*/
#define fis_gcI_i2 2  // Number of inputs to the fuzzy inference system
#define fis_gcO_i2 1  // Number of outputs to the fuzzy inference system
#define fis_gcR_i2 5  // Number of rules to the fuzzy inference system

/*SISTEMA FUZZY INTERVALO 3*/
#define fis_gcI_i3 2  // Number of inputs to the fuzzy inference system
#define fis_gcO_i3 1  // Number of outputs to the fuzzy inference system
#define fis_gcR_i3 5  // Number of rules to the fuzzy inference system

#define MAXIMO_DE_RULES 5  // Number maximum of rules to the fuzzy inference system

//#define min(a,b) (((a)<(b))?(a):(b))   //macro para usar funcion min() en el caso que no exista
//#define max(a,b) (((a)>(b))?(a):(b))   //macro para usar funcion max() en el caso que no exista
//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm


#define DHTPIN 8     // digital pin donde tenemos conectado el sensor DHT
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321

//El pin 13 de Arduino lo dejamos para testeo porque tiene el LED de la placa y es muy util para saber si
nuestro programa funciona.
#define ledPin 13  //pin para el fotoperiodo
#define boton_CHG 2  //pin Boton CHG (Change)
#define boton_MENU 4  //pin Boton MENU (Menu)
#define humidificador 3  // pin de salida PWM para el humidificador de 107kHz
#define fuzzy_aire_bomba 5  //pin para la bomba de aire


#define  HABILITADO_FUZZY "FUZZY HABILIT"  //Texto para el menu
#define  DESHABILITADO_FUZZY "FUZZY DESHABI" //Texto para el menu

#define  DESHABILITADO_HUMIDIFICADOR "HUMI DESHABIL" //Texto para el menu

#define RELE_ACTIVO LOW  //Logica invertida en los Reles
#define RELE_APAGADO HIGH //Logica invertida en los Reles


//RETARDOS de los CONTADORES
#define RETARDO_CONEXION_RELES 4  //4 segundos de retardo para la conexion o desconexion de los reles
#define RETARDO_TIEMPO_EN_OFF_BOMBA 60  //60 segundos de retardo para mantener la bomba en OFF, el
tiempo en ON lo calculamos con el FUZZY
//#define RETARDO_TIEMPO_HUMIDIFICADOR_ON 5  //5 segundos ON para el humidificador
#define RETARDO_TIEMPO_HUMIDIFICADOR_OFF 35  //35 segundos OFF para el humidificador
```

```
#define RETARDO_HORAS_MODO_GROW 432  //18 dias de modo grow para la incubación del micelio
#define RETARDO_HORAS_MODO_FRUCTIF 999  //50 dias de modo fructificación para cosechar setas

#define RETARDO_HORAS_FOTOPERIODO_ON 12  //12 horas en ON y 12 horas en OFF de fotoperiodo
#define RETARDO_HORAS_FOTOPERIODO_OFF 12  //12 horas en ON y 12 horas en OFF de fotoperiodo


//RELES     ------>>> OJO, los relés estarán a LOW al comienzo del programa
#define rele_A 7          // pin arduino para entrada A del puente en H
#define rele_B 12         // pin arduino para entrada B del puente en H

#define PONDERACION_REAJUSTE_FUZZY 2 //dividire por 2 el valor fuzzy calculado porque sino a la peltier le
costara mucho enfriar


/*-----( Declaraciones para el LCD )-----*/
// set the LCD address to 0x27 for a 20 chars 4 line display
// Set the pins on the I2C chip used for LCD connections:
//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address
DHT dht(DHTPIN, DHTTYPE);  //cargamos configuracion del sensor DHT

char horas,minutos,segundos;
boolean bandera1=false; //cada segundo
boolean bandera2=false; //cada 10 segundos
boolean bandera3=false; //cada 2 segundos
char VARI[16]; //para almacenar un string generico y poder trabajar con el

boolean valor_boton_CHG;  //aqui almacenamos el valor leido del BOTON CHG
boolean valor_boton_MENU; //aqui almacenamos el valor leido del BOTON MENU
int contador_LED=0; //contador para el encendido del LCD y LED mientras estamos en el menu
boolean el_LED_esta_encendido=false; //flag para identificar que el LED del FOTOPERIODO estaba encendido y
que no afecte nada del menu

//FLAGS para identificar en que menu estoy y en que nivel de profundidad sobre ese menu
int bandera_nivel_menu=0;
int menu=0;
boolean boton_MENU_primera_vez=false;


//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
// CONTADORES de RETARDOS
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////

int contador_iniciado_reles=0;
int contador_iniciado_humidificador=0; //funcionara a razon de XX segundos ON
int contador_iniciado_humidificador_en_OFF=0; //funcionara a razon de XX segundos OFF
int contador_iniciado_horas_modo=0;  // Esta variable se pondra a cero con el menu Reset de contador  //
int contador_iniciado_fuzzy_aire_bomba=0;
int contador_iniciado_fuzzy_aire_bomba_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF =
RETARDO_TIEMPO_EN_OFF_BOMBA;
int contador_iniciado_fotoperiodo=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF =
RETARDO_TIEMPO_EN_OFF_BOMBA;
int contador_iniciado_fotoperiodo_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF =
RETARDO_TIEMPO_EN_OFF_BOMBA;

//////////////////////////////////////////////////////////////////
// CONDICIONES INICIALES
//////////////////////////////////////////////////////////////////

int valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON[]={5,10,15,20,99};  //5 segundos ON para el humidificador
por defecto y 99 es el valor para pararlo
int indice_valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON=0; //por defecto apuntamos al primer valor de
retardo en ON para el humidificador
int
```

```
valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON=valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON[indice_valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON];

float temperatura_inicial=27;
float humedad_inicial=60;

float temperatura_actual = 27; //leida cada dos minutos
float humedad_actual = 60; //leida cada dos minutos

float delta_temp=0;  //delta temp y delta humedad se calcularan cada vez que se termine el tiempo en off de la bomba de aire.
float delta_humedad=0;

float temperatura_10s = 27; //leida cada 10 segundos
float humedad_10s = 60; //leida cada 10 segundos

/*****************************************************************/
boolean flag_empezado_modo_grow=false;
boolean flag_empezado_modo_fructif=false;
boolean flag_empezado_modo_auto=false;
boolean flag_empezado_modo_auto_2a_fase=false;
/*****************************************************************/

boolean activado_reles_flag_frio = false;
boolean activado_reles_flag_calor = false;

// 0=parado, 1=frio, 2=calor ///el modo parado no lo implementamos y lo dejamos para ampliaciones por si el dia de mañana ponemos un sensor
////////////////////////////////externo a la incubadora para medir temp y hum exterior
////////parado estará cuando no este activo el modo//////////////////
int parado_o_frio_o_calor=0; //por defecto parada la peltier

/*****************************************************************/
boolean activado_flag_humidificador=true;
boolean flag_hay_que_humidificar = false;
boolean flag_habilitado_humidificador=true;  //por defecto arranco con el Humidificador habilitado
/*****************************************************************/
/*****************************************************************/
boolean activado_flag_bomba_aireadora=true;
/*****************************************************************/
/*****************************************************************/
boolean activado_flag_fotoperiodo=true;
/*****************************************************************/
/*****************************************************************/
boolean flag_habilitado_fuzzy=true;  //por defecto arranco con el FUZZY habilitado para la bomba de aire
/*****************************************************************/
/*****************************************************************/

int intervalo = 0; //intervalo fuzzy bomba de aire
```

2.6.3 Código completo del programa

```
/* Jose Carlos Serrano Domingo - 2016/2017                                    */
/******************************************************************************/
/******************************************************************************/
/******************************************************************************/
/*   Programa para Arduino UNO - Invernadero de Setas con Celula Peltier mediante Fuzzy Logic
Usamos:
El Timer 1 para interrupciones de 1 segundo y controlar el flujo del programa
El Timer 0 no se toca para no alterar las funciones de tiempo basicas como delay()
El Timer 2 lo usamos para el humidificador, generando una señal cuadrada de 107kHz mediante PWM

Los relés que gobiernan la celula Peltier conforman un Puente en H. Son de logica invertida
El sensor de temperatura y humedad es un DHT22.
El LCD es uno 16x2 I2C serie
La bomba de aire se ejecutara a intervalos calculados mediante algoritmo de control Fuzzy Logic
Podemos regular tambien el fotoperiodo mediante un LED de 6W.

El flujo del programa se ejecuta en paralelo mediante semaforos que bloquean o no la ejecucion de cada hilo
Basicamente tenemos los siguientes hilos:
        El Menu del LCD
        Modo Grow
                Humidificador
                Peltier
                Fuzzy Bomba Aire
        Modo Fructif
                Humidificador
                Peltier
                Fuzzy Bomba Aire
                Fotoperiodo
        Modo Auto
                Modo Grow
                Modo Fructif
        ISR
                Gestion de los contadores para cada funcion de forma periodica a razon de 1 segundo
*/
/******************************************************************************/
/******************************************************************************/
/******************************************************************************/


/*-----( Cargamos las librerias necesarias )-----*/
#include <DHT.h>   //para el sensor de temperatura y humedad
#include <Wire.h>
#include <LiquidCrystal_I2C.h>  //para el LCD
#include "./fis_header.h"   //algunas definiciones para el sistema Fuzzy


/*SISTEMA FUZZY INTERVALO 1*/
#define fis_gcI_i1 2  // Number of inputs to the fuzzy inference system
#define fis_gcO_i1 1  // Number of outputs to the fuzzy inference system
#define fis_gcR_i1 5  // Number of rules to the fuzzy inference system

/*SISTEMA FUZZY INTERVALO 2*/
#define fis_gcI_i2 2  // Number of inputs to the fuzzy inference system
#define fis_gcO_i2 1  // Number of outputs to the fuzzy inference system
#define fis_gcR_i2 5  // Number of rules to the fuzzy inference system

/*SISTEMA FUZZY INTERVALO 3*/
#define fis_gcI_i3 2  // Number of inputs to the fuzzy inference system
#define fis_gcO_i3 1  // Number of outputs to the fuzzy inference system
#define fis_gcR_i3 5  // Number of rules to the fuzzy inference system

#define MAXIMO_DE_RULES 5  // Number maximum of rules to the fuzzy inference system

//#define min(a,b) (((a)<(b))?(a):(b))   //macro para usar funcion min() en el caso que no exista
//#define max(a,b) (((a)>(b))?(a):(b))   //macro para usar funcion max() en el caso que no exista
//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
```

```
//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm


#define DHTPIN 8     // digital pin donde tenemos conectado el sensor DHT
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321

//El pin 13 de Arduino lo dejamos para testeo porque tiene el LED de la placa y es muy util para saber si nuestro programa funciona.
#define ledPin 13  //pin para el fotoperiodo
#define boton_CHG 2  //pin Boton CHG (Change)
#define boton_MENU 4  //pin Boton MENU (Menu)
#define humidificador 3  // pin de salida PWM para el humidificador de 107kHz
#define fuzzy_aire_bomba 5  //pin para la bomba de aire


#define  HABILITADO_FUZZY "FUZZY HABILIT"  //Texto para el menu
#define  DESHABILITADO_FUZZY "FUZZY DESHABI" //Texto para el menu

#define  DESHABILITADO_HUMIDIFICADOR "HUMI DESHABIL" //Texto para el menu

#define RELE_ACTIVO LOW  //Logica invertida en los Reles
#define RELE_APAGADO HIGH //Logica invertida en los Reles


//RETARDOS de los CONTADORES
#define RETARDO_CONEXION_RELES 4  //4 segundos de retardo para la conexion o desconexion de los reles
#define RETARDO_TIEMPO_EN_OFF_BOMBA 60  //60 segundos de retardo para mantener la bomba en OFF, el tiempo en ON lo calculamos con el FUZZY
//#define RETARDO_TIEMPO_HUMIDIFICADOR_ON 5  //5 segundos ON para el humidificador
#define RETARDO_TIEMPO_HUMIDIFICADOR_OFF 35  //35 segundos OFF para el humidificador


#define RETARDO_HORAS_MODO_GROW 432  //18 dias de modo grow para la incubación del micelio
#define RETARDO_HORAS_MODO_FRUCTIF 999  //50 dias de modo fructificación para cosechar setas

#define RETARDO_HORAS_FOTOPERIODO_ON 12  //12 horas en ON y 12 horas en OFF de fotoperiodo
#define RETARDO_HORAS_FOTOPERIODO_OFF 12  //12 horas en ON y 12 horas en OFF de fotoperiodo


//RELES     ------>>> OJO, los relés estarán a LOW al comienzo del programa
#define rele_A 7          // pin arduino para entrada A del puente en H
#define rele_B 12         // pin arduino para entrada B del puente en H

#define PONDERACION_REAJUSTE_FUZZY 2 //dividire por 2 el valor fuzzy calculado porque sino a la peltier le costara mucho enfriar


/*-----( Declaraciones para el LCD )-----*/
// set the LCD address to 0x27 for a 20 chars 4 line display
// Set the pins on the I2C chip used for LCD connections:
//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address
DHT dht(DHTPIN, DHTTYPE);  //cargamos configuracion del sensor DHT

char horas,minutos,segundos;
boolean bandera1=false; //cada segundo
boolean bandera2=false; //cada 10 segundos
boolean bandera3=false; //cada 2 segundos
char VARI[16]; //para almacenar un string generico y poder trabajar con el

boolean valor_boton_CHG;  //aqui almacenamos el valor leido del BOTON CHG
boolean valor_boton_MENU; //aqui almacenamos el valor leido del BOTON MENU
int contador_LED=0; //contador para el encendido del LCD y LED mientras estamos en el menu
boolean el_LED_esta_encendido=false; //flag para identificar que el LED del FOTOPERIODO estaba encendido y que no afecte nada del menu

//FLAGS para identificar en que menu estoy y en que nivel de profundidad sobre ese menu
```

```
int bandera_nivel_menu=0;
int menu=0;
boolean boton_MENU_primera_vez=false;


////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
// CONTADORES de RETARDOS
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

int contador_iniciado_reles=0;
int contador_iniciado_humidificador=0; //funcionara a razon de XX segundos ON
int contador_iniciado_humidificador_en_OFF=0; //funcionara a razon de XX segundos OFF
int contador_iniciado_horas_modo=0;  // Esta variable se pondra a cero con el menu Reset de contador  //
int contador_iniciado_fuzzy_aire_bomba=0;
int contador_iniciado_fuzzy_aire_bomba_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF =
RETARDO_TIEMPO_EN_OFF_BOMBA;
int contador_iniciado_fotoperiodo=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF =
RETARDO_TIEMPO_EN_OFF_BOMBA;
int contador_iniciado_fotoperiodo_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF =
RETARDO_TIEMPO_EN_OFF_BOMBA;

////////////////////////////////////////////////////////////////
// CONDICIONES INICIALES
////////////////////////////////////////////////////////////////

int valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON[]={5,10,15,20,99};  //5 segundos ON para el humidificador
por defecto y 99 es el valor para pararlo
int indice_valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON=0; //por defecto apuntamos al primer valor de
retardo en ON para el humidificador
int
valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON=valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON[indice_
valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON];

float temperatura_inicial=27;
float humedad_inicial=60;

float temperatura_actual = 27; //leida cada dos minutos
float humedad_actual = 60; //leida cada dos minutos

float delta_temp=0;  //delta temp y delta humedad se calcularan cada vez que se termine el tiempo en off de
la bomba de aire.
float delta_humedad=0;

float temperatura_10s = 27; //leida cada 10 segundos
float humedad_10s = 60; //leida cada 10 minutos

/*****************************************************************/
boolean flag_empezado_modo_grow=false;
boolean flag_empezado_modo_fructif=false;
boolean flag_empezado_modo_auto=false;
boolean flag_empezado_modo_auto_2a_fase=false;
/*****************************************************************/

boolean activado_reles_flag_frio = false;
boolean activado_reles_flag_calor = false;

// 0=parado, 1=frio, 2=calor ///el modo parado no lo implementamos y lo dejamos para ampliaciones por si el
dia de mañana ponemos un sensor
//////////////////////////////externo a la incubadora para medir temp y hum exterior
////////parado estará cuando no este activo el modo//////////////////
int parado_o_frio_o_calor=0; //por defecto parada la peltier

/*****************************************************************/
boolean activado_flag_humidificador=true;
boolean flag_hay_que_humidificar = false;
boolean flag_habilitado_humidificador=true;  //por defecto arranco con el Humidificador habilitado
/*****************************************************************/
```

```
/*************************************************************************/
boolean activado_flag_bomba_aireadora=true;
/*************************************************************************/
/*************************************************************************/
boolean activado_flag_fotoperiodo=true;
/*************************************************************************/
/*************************************************************************/
boolean flag_habilitado_fuzzy=true;  //por defecto arranco con el FUZZY habilitado para la bomba de aire
/*************************************************************************/
/*************************************************************************/

int intervalo = 0; //intervalo fuzzy bomba de aire



////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCIONES
////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION COMIENZA RETARDO
// funcion que la usaremos para pasar de forma generica un valor de retardo a una variable contador
////////////////////////////////////////////////////////////////////////////////////////////////////////
void comienza_retardo(int cuantos_segundos, int *mi_variable){
     *mi_variable = cuantos_segundos;
}


////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION PINTAR CONTADOR
// funcion para mostrar en el LCD el valor restante de horas del modo que este iniciado
////////////////////////////////////////////////////////////////////////////////////////////////////////
void pintar_contador(){
              sprintf(VARI,"%03d",contador_iniciado_horas_modo);
              lcd.setCursor(13, 0);
              lcd.print(VARI);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION PINTAR ESPACIO VACIO CONTADOR
// funcion que la usaremos para borrar del LCD el valor del contador de horas del modo que estuviese activo
////////////////////////////////////////////////////////////////////////////////////////////////////////
void pintar_espacio_vacio_contador(){
              lcd.setCursor(12, 0);
              lcd.print(F("    "));
}

////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_MODO_GROW
// funcion que la usaremos para configurar los flags adecuadamente para que comience el MODO GROW en el
loop()
////////////////////////////////////////////////////////////////////////////////////////////////////////
void start_modo_grow(){
   int *mi_variable = &contador_iniciado_horas_modo;

     flag_empezado_modo_grow=true;
     flag_empezado_modo_fructif=false;
          parado_o_frio_o_calor=1; //frio al inicio
     activado_reles_flag_frio = true; //comienzo enfriando que será lo más normal
          activado_reles_flag_calor = false; //comienzo enfriando que será lo más normal

          comienza_retardo(RETARDO_HORAS_MODO_GROW, mi_variable);

}//start modo grow

////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_MODO_GROW
```

```
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el MODO GROW en el
loop()
////////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_modo_grow(){
    flag_empezado_modo_grow=false;
    lcd.setCursor(14, 1);
    lcd.print(F(" S"));

          activado_reles_flag_frio = false; //dejo de enfriar
          activado_reles_flag_calor = false; //dejo de calentar
          parado_o_frio_o_calor=0;
          contador_iniciado_horas_modo=0;
          contador_iniciado_humidificador=0;
          contador_iniciado_humidificador_en_OFF=0;
          activado_flag_humidificador=true;
          flag_hay_que_humidificar = false;

         digitalWrite(humidificador, LOW);
         digitalWrite(fuzzy_aire_bomba, LOW);

          contador_iniciado_fuzzy_aire_bomba=0;
          contador_iniciado_fuzzy_aire_bomba_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF
= RETARDO_TIEMPO_EN_OFF_BOMBA;
          activado_flag_bomba_aireadora=true;

          contador_iniciado_fotoperiodo=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF =
RETARDO_TIEMPO_EN_OFF_BOMBA;
          contador_iniciado_fotoperiodo_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF =
RETARDO_TIEMPO_EN_OFF_BOMBA;
          activado_flag_fotoperiodo=true;
          el_LED_esta_encendido=false;

        pintar_contador();
  desconecto_reles();

}//stop modo grow

////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_MODO_FRUCTIF
// funcion que la usaremos para configurar los flags adecuadamente para que comience el MODO FRUCTIF en el
loop()
////////////////////////////////////////////////////////////////////////////////////////////////////////
void start_modo_fructif(){
  int *mi_variable = &contador_iniciado_horas_modo;

    flag_empezado_modo_fructif=true;
    flag_empezado_modo_grow=false;
       parado_o_frio_o_calor=1; //frio al inicio
          activado_reles_flag_frio = true; //comienzo enfriando que será lo más normal
          activado_reles_flag_calor = false; //comienzo enfriando que será lo más normal

         comienza_retardo(RETARDO_HORAS_MODO_FRUCTIF, mi_variable);

}//start modo fructif

////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_MODO_FRUCTIF
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el MODO FRUCTIF en el
loop()
////////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_modo_fructif(){
    flag_empezado_modo_fructif=false;
    lcd.setCursor(14, 1);
    lcd.print(F(" S"));
          activado_reles_flag_frio = false; //dejo de enfriar
          activado_reles_flag_calor = false; //dejo de calentar
          parado_o_frio_o_calor=0;
          contador_iniciado_horas_modo=0;
  digitalWrite(ledPin, LOW); //apago el LED interior
```

```
    digitalWrite(humidificador, LOW);
          digitalWrite(fuzzy_aire_bomba, LOW);

           contador_iniciado_humidificador=0;
           contador_iniciado_humidificador_en_OFF=0;
           activado_flag_humidificador=true;
           flag_hay_que_humidificar = false;

           contador_iniciado_fuzzy_aire_bomba=0;
           contador_iniciado_fuzzy_aire_bomba_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF
= RETARDO_TIEMPO_EN_OFF_BOMBA;
           activado_flag_bomba_aireadora=true;

           contador_iniciado_fotoperiodo=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF =
RETARDO_TIEMPO_EN_OFF_BOMBA;
           contador_iniciado_fotoperiodo_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF =
RETARDO_TIEMPO_EN_OFF_BOMBA;
           activado_flag_fotoperiodo=true;
           el_LED_esta_encendido=false;

          pintar_contador();
    desconecto_reles();

}//stop modo fructif

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_MODO_AUTO
// funcion que la usaremos para configurar los flags adecuadamente para que funcione el MODO AUTO en el
loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void start_modo_auto(){
    flag_empezado_modo_auto=true;
    lcd.setCursor(14, 1);
    lcd.print(F("AG"));
    start_modo_grow(); //comienzo primero con el modo GROW
    //start_modo_fructif(); //comienzo despues con el modo GROW
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_MODO_AUTO
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el MODO AUTO en el
loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_modo_auto(){
    stop_modo_grow();
    stop_modo_fructif();
     flag_empezado_modo_auto=false;
           flag_empezado_modo_auto_2a_fase=false;
     lcd.setCursor(14, 1);
     lcd.print(F(" S"));
}//stop modo fructif

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_TOTAL
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el MODO que estuviera
activo en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_total(){

  int *mi_variable = &contador_iniciado_reles;
  comienza_retardo(RETARDO_CONEXION_RELES, mi_variable);  //comienzo el retardo de los reles para que si
empiezo otro modo no conmute inmediatamente la peltier.

    stop_modo_grow();
    stop_modo_fructif();
    stop_modo_auto();
}//stop modo fructif
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION MODO_GROW
// funcion que que se corresponde al hilo de ejecucion MODO_GROW en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////////
void modo_grow(){
    int *mi_variable = &contador_iniciado_reles;
//para que el micelio crezca ha de estar a una temperatura optima de 25ºC. la temperatura de la sala ha de estar entre 18 y 24ºC

if( flag_empezado_modo_grow==true){
pintar_contador();

  //esta funcion se llama cada 2 segundos  con la bandera 3
     //comienzo calculando el valor para activar el frio, calor o parado
     //comienzo calculando el valor para activar el humidificador o no (que habiamos acordado de hacerlo a rafagas de 15 segundos en on y 15 segundos en off)
     //NO HAY LUZ EN ESTE MODO

                if ((temperatura_10s <= 24) && (activado_reles_flag_frio == true) && (activado_reles_flag_calor == false)){
                        parado_o_frio_o_calor=2; //estaba enfriando y debo pasar a calentar porque si no me paso de frio
                                comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo retardo porque habra un cambio de estado
                                desconecto_reles();
                }else if((temperatura_10s >= 27) && (activado_reles_flag_frio == false) && (activado_reles_flag_calor == true)){
                        parado_o_frio_o_calor=1; //estaba calentando y debo pasar a enfriar porque si no me paso de calor
                                comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo retardo porque habra un cambio de estado
                                desconecto_reles();
                }else if((activado_reles_flag_frio == true) && (activado_reles_flag_calor == true)){
                        activado_reles_flag_frio = true; //esta condicion no debe darse y por eso volvemos a resetearlo como si comenzaramos enfriando
                        activado_reles_flag_calor = false;
                        parado_o_frio_o_calor=1;
                                comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo retardo porque habra un cambio de estado
                                desconecto_reles();
                }else if( (temperatura_10s > 24) && (temperatura_10s < 27) ){
                        //no cambio nada porque estoy funcionando en la zona optima
                }


/*************************************************************************/
////////////////////////////////////////////////////////////////////////////
/// DE SI HAY QUE PONER FRIO O CALOR ///////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

     if (parado_o_frio_o_calor == 1){
        activo_flag_peltier_frio();
     }else if (parado_o_frio_o_calor == 2){
        activo_flag_peltier_calor();
     }else if (parado_o_frio_o_calor == 0){
        desactivo_flag_peltier();
        desconecto_reles();
     }

////////////////////////////////////////////////////////////////////////////
/*************************************************************************/
/*************************************************************************/
////////////////////////////////////////////////////////////////////////////
/// DE SI HAY QUE HUMIDIFICAR O NO /////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

//Activamos el humidificador siempre que la humedad esté por debajo de 80% y dejamos de humidificar si
```

```
estamos por encima de 90%
                    if (humedad_10s <= 70){
                            flag_hay_que_humidificar = true;
                    }else if (humedad_10s >= 80){
                            flag_hay_que_humidificar = false;
                    }

////////////////////////////////////////////////////////////////////////////
/*************************************************************************/

  peltier(); //activara o desactivara el frio o el calor
  humidifica(); //activa o desactiva el humidificador

////////////////////////////////////////////////////////////////////////////
/*************************************************************************/

////////////////////////////////////////////////////////////////////////////
/*************************************************************************/
/*************************************************************************/
/*************************************************************************/
          if(temperatura_actual <= 23){
                    intervalo = 1;
          }else if((temperatura_actual > 23) &&(temperatura_actual < 27)){
                    intervalo = 2;
          }else if(temperatura_actual >= 27){
                    intervalo = 3;
          }
/*************************************************************************/

    bomba_aireadora(); //activa o desactiva la bomba aireadora
/*************************************************************************/
/*************************************************************************/
////////////////////////////////////////////////////////////////////////////
/*************************************************************************/
}

if((flag_empezado_modo_grow==true)&&(contador_iniciado_horas_modo==0)){
          stop_modo_grow();
}//if modo grow

}//modo grow
/*************************************************************************/
/*************************************************************************/
/*************************************************************************/

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION MODO_FRUCTIF
// funcion que que se corresponde al hilo de ejecucion MODO_FRUCTIF en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void modo_fructif(){
    int *mi_variable = &contador_iniciado_reles;
//para que el micelio fructifique ha de estar a una temperatura optima de 12ºC. la temperatura de la sala ha de
estar entre 12 y 18ºC

if( flag_empezado_modo_fructif==true){
pintar_contador();

  //esta funcion se llama cada 2 segundos  con la bandera 3
      //comienzo calculando el valor para activar el frio, calor o parado
      //comienzo calculando el valor para activar el humidificador o no (que habiamos acordado de hacerlo a
rafagas de 15 segundos en on y 15 segundos en off)
      //SI HAY LUZ EN ESTE MODO

                    if ((temperatura_10s <= 11) && (activado_reles_flag_frio == true) &&
(activado_reles_flag_calor == false)){
                            parado_o_frio_o_calor=2; //estaba enfriando y debo pasar a calentar porque si no me
paso de frio
```

```
                                comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo
retardo porque habra un cambio de estado
                                desconecto_reles();
                        }else if((temperatura_10s >= 14) && (activado_reles_flag_frio == false) &&
(activado_reles_flag_calor == true)){
                                parado_o_frio_o_calor=1; //estaba calentando y debo pasar a enfriar porque si no me
paso de calor
                                comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo
retardo porque habra un cambio de estado
                                desconecto_reles();
                        }else if((activado_reles_flag_frio == true) && (activado_reles_flag_calor == true)){
                                activado_reles_flag_frio = true; //esta condicion no debe darse y por eso volvemos a
resetearlo como si comenzaramos enfriando
                                activado_reles_flag_calor = false;
                                parado_o_frio_o_calor=1;
                                comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo
retardo porque habra un cambio de estado
                                desconecto_reles();
                        }else if( (temperatura_10s > 11) && (temperatura_10s < 14) ){
                                //no cambio nada porque estoy funcionando en la zona optima
                        }


/*****************************************************************************/
//////////////////////////////////////////////////////////////////////////////
/// DE SI HAY QUE PONER FRIO O CALOR ////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////

    if (parado_o_frio_o_calor == 1){
       activo_flag_peltier_frio();
    }else if (parado_o_frio_o_calor == 2){
       activo_flag_peltier_calor();
    }else if (parado_o_frio_o_calor == 0){
       desactivo_flag_peltier();
       desconecto_reles();
    }

//////////////////////////////////////////////////////////////////////////////
/*****************************************************************************/
/*****************************************************************************/
//////////////////////////////////////////////////////////////////////////////
/// DE SI HAY QUE HUMIDIFICAR O NO //////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////

//Activamos el humidificador siempre que la humedad esté por debajo de 80% y dejamos de humidificar si
estamos por encima de 90%
                        if (humedad_10s <= 70){
                                flag_hay_que_humidificar = true;
                        }else if (humedad_10s >= 80){
                                flag_hay_que_humidificar = false;
                        }

//////////////////////////////////////////////////////////////////////////////
/*****************************************************************************/

  peltier(); //activara o desactivara el frio o el calor
  humidifica(); //activa o desactiva el humidificador

//////////////////////////////////////////////////////////////////////////////
/*****************************************************************************/

//////////////////////////////////////////////////////////////////////////////
/*****************************************************************************/
/*****************************************************************************/
/*****************************************************************************/
        if(temperatura_actual <= 11){
                intervalo = 1;
        }else if((temperatura_actual > 11) &&(temperatura_actual < 14)){
                intervalo = 2;
```

```
        }else if(temperatura_actual >= 14){
                intervalo = 3;
        }
/*****************************************************************************/

   bomba_aireadora(); //activa o desactiva la bomba aireadora
/*****************************************************************************/
/*****************************************************************************/
//////////////////////////////////////////////////////////////////////////////
/*****************************************************************************/
        fotoperiodo();

}

if((flag_empezado_modo_fructif==true)&&(contador_iniciado_horas_modo==0)){
        stop_modo_fructif();
}//if modo fructif

}//modo fructif
/*****************************************************************************/
/*****************************************************************************/
/*****************************************************************************/

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION MODO_AUTO
// funcion que que se corresponde al hilo de ejecucion MODO_AUTO en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void modo_auto(){

/
***##################################################################
########***/
   //1º modo_auto_grow;
/
***##################################################################
########***/
if( (flag_empezado_modo_grow==true) && ( flag_empezado_modo_auto==true) &&
(flag_empezado_modo_auto_2a_fase==false)){

////////
         modo_grow();
////////
}else if( (flag_empezado_modo_grow==false) && (flag_empezado_modo_fructif==false) &&
( flag_empezado_modo_auto==true) && (flag_empezado_modo_auto_2a_fase==false)){
        stop_modo_grow();
        flag_empezado_modo_auto_2a_fase=true;
    lcd.setCursor(14, 1);
    lcd.print(F("AF"));
        start_modo_fructif();
}//if modo auto grow



/
***##################################################################
########***/
   //2º modo_auto_fructif();
/
***##################################################################
########***/

if((flag_empezado_modo_fructif==true) && ( flag_empezado_modo_auto==true) &&
(flag_empezado_modo_auto_2a_fase==true) ){

////////
         modo_fructif();
////////
```

```
}else if( (flag_empezado_modo_fructif==false) && (flag_empezado_modo_grow==false) &&
( flag_empezado_modo_auto==true) && (flag_empezado_modo_auto_2a_fase==true) ){
        stop_modo_auto(); //fin de modo auto, incluye fin de modo grow y fructif
}//if modo auto fructif

}//modo auto

/*****************************************************************************/
/*****************************************************************************/
/*****************************************************************************/



////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_FUZZY_AIRE_BOMBA
// funcion que la usaremos para configurar los flags adecuadamente para que comience a funcionar la bomba
de aire en el loop()
// tambien activa el retardo correspondiente al ON
////////////////////////////////////////////////////////////////////////////////////////////////
void start_fuzzy_aire_bomba(){
/////////////////////////////////////////////////////////////
  int *mi_variable = &contador_iniciado_fuzzy_aire_bomba;
  float retardo_calculado_fuzzy=0;

if(contador_iniciado_fuzzy_aire_bomba!=0){

}else{
      digitalWrite(fuzzy_aire_bomba, HIGH);
         //contador_iniciado_fuzzy_aire_bomba = intervalos(intervalo, delta_temp, delta_humedad, Fuzzy*
fuzzy, Fuzzy* fuzzy2, Fuzzy* fuzzy3)
      retardo_calculado_fuzzy=intervalos(intervalo, delta_temp,
delta_humedad)/PONDERACION_REAJUSTE_FUZZY;

      if (int(retardo_calculado_fuzzy) == 0 ){
        retardo_calculado_fuzzy=1;  //nunca debe valer cero sino la bomba no se apagaria porque siempre
estaria haciendo el start
      }
         comienza_retardo(int(retardo_calculado_fuzzy), mi_variable);
         activado_flag_bomba_aireadora=false;
}//if
}//start_fuzzy_aire_bomba


////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_FUZZY_AIRE_BOMBA
// funcion que la usaremos para configurar los flags adecuadamente para que finalice la bomba de aire en el
loop()
// tambien activa el retardo correspondiente al OFF
////////////////////////////////////////////////////////////////////////////////////////////////
void stop_fuzzy_aire_bomba(){
  int *mi_variable = &contador_iniciado_fuzzy_aire_bomba_en_OFF;
/////////////////////////////////////////////////////////////
if(contador_iniciado_fuzzy_aire_bomba_en_OFF!=0){

}else{
         digitalWrite(fuzzy_aire_bomba, LOW);
         //contador_iniciado_fuzzy_aire_bomba=0;
         //contador_iniciado_fuzzy_aire_bomba = intervalos(intervalo, delta_temp, delta_humedad, Fuzzy*
fuzzy, Fuzzy* fuzzy2, Fuzzy* fuzzy3)
      comienza_retardo(RETARDO_TIEMPO_EN_OFF_BOMBA, mi_variable);
      activado_flag_bomba_aireadora=true;
}//if
}//stop_fuzzy_aire_bomba


////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION BOMBA_AIREADORA
// funcion que se corresponde al hilo de ejecucion de la bomba de aire dentro del loop()
// si el hilo esta habilitado la bomba funciona, sino la bomba esta desconectada
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////
void bomba_aireadora(){  //activara la bomba en el estado que indiquen los flags
if(flag_habilitado_fuzzy==false){
          digitalWrite(fuzzy_aire_bomba, LOW);
       contador_iniciado_fuzzy_aire_bomba=0;
          contador_iniciado_fuzzy_aire_bomba_en_OFF = 0;
          activado_flag_bomba_aireadora=true;
}else{

   if((activado_flag_bomba_aireadora==true)&&(contador_iniciado_fuzzy_aire_bomba_en_OFF==0)){
        start_fuzzy_aire_bomba();
   }else if(( activado_flag_bomba_aireadora==false)&&(contador_iniciado_fuzzy_aire_bomba==0)){
        stop_fuzzy_aire_bomba();
   }
}//if fuzzy habilitado
}


////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_FOTOPERIODO
// funcion que la usaremos para configurar los flags adecuadamente para que comience a funcionar el
FOTOPERIODO en el loop()
// tambien activa el retardo correspondiente al ON
////////////////////////////////////////////////////////////////////////////////////////////////////////
void start_fotoperiodo(){
//////////////////////////////////////////////////////////////////
  int *mi_variable = &contador_iniciado_fotoperiodo;

if(contador_iniciado_fotoperiodo!=0){

}else{

          digitalWrite(ledPin, HIGH);
          el_LED_esta_encendido=true;
          //contador_iniciado_fotoperiodo = RETARDO_HORAS_FOTOPERIODO_ON;
          comienza_retardo(RETARDO_HORAS_FOTOPERIODO_ON, mi_variable);
          activado_flag_fotoperiodo=false;
  }//if
}//start_fuzzy_aire_bomba



////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_FOTOPERIODO
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el FOTOPERIODO en el
loop()
// tambien activa el retardo correspondiente al OFF
////////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_fotoperiodo(){
int *mi_variable = &contador_iniciado_fotoperiodo_en_OFF;
//////////////////////////////////////////////////////////////////
  if(  contador_iniciado_fotoperiodo_en_OFF != 0){

  }else{
       digitalWrite(ledPin, LOW);
                    el_LED_esta_encendido=false;
       comienza_retardo(RETARDO_HORAS_FOTOPERIODO_OFF, mi_variable);
       activado_flag_fotoperiodo=true;
  }
}//stop_fuzzy_aire_bomba


////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION FOTOPERIODO
// funcion que se corresponde al hilo de ejecucion del fotoperiodo dentro del loop()
////////////////////////////////////////////////////////////////////////////////////////////////////////
void fotoperiodo(){  //activara el fotoperiodo en el estado que indiquen los flags
```

```
    if((activado_flag_fotoperiodo==true)&&(contador_iniciado_fotoperiodo_en_OFF==0)){
        start_fotoperiodo();
    }else if((activado_flag_fotoperiodo==false)&&(contador_iniciado_fotoperiodo==0)){
        stop_fotoperiodo();
    }
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_HUMIDIFICADOR
// funcion que la usaremos para configurar los flags adecuadamente para que comience a funcionar el
HUMIDIFICADOR en el loop()
// tambien activa el retardo correspondiente al ON
////////////////////////////////////////////////////////////////////////////////////////////////////
void start_humidificador(){
//////////////////////////////////////////////////////////////////
// PWM
//////////////////////////////////////////////////////////////////
  int *mi_variable = &contador_iniciado_humidificador;

if(contador_iniciado_humidificador!=0){

}else{


  pinMode(humidificador, OUTPUT);// output pin for OCR2B

  // Set up the 107kHz output
  TCCR2A = _BV(COM2A1) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
  TCCR2B = _BV(WGM22) | _BV(CS20);
  OCR2A = 149; //159
  OCR2B = 74; //79

  comienza_retardo(valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON, mi_variable);
  //contador_iniciado_humidificador=RETARDO_TIEMPO_HUMIDIFICADOR;
  activado_flag_humidificador=false;
}

}//start_humidificador


////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_HUMIDIFICADOR
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el HUMIDIFICADOR en el
loop()
// tambien activa el retardo correspondiente al OFF
////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_humidificador(){
  int *mi_variable = &contador_iniciado_humidificador_en_OFF;

  if(contador_iniciado_humidificador_en_OFF != 0){

  }else{

    digitalWrite(humidificador, LOW);
    comienza_retardo(RETARDO_TIEMPO_HUMIDIFICADOR_OFF, mi_variable);
          //contador_iniciado_humidificador=RETARDO_TIEMPO_HUMIDIFICADOR;
          activado_flag_humidificador=true;
  }

}//stop_humidificador


////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION HUMIDIFICADOR
// funcion que se corresponde al hilo de ejecucion del humidificador dentro del loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void humidifica(){  //activara el humidificador en el estado que indiquen los flags

if(flag_habilitado_humidificador==false){
```

```
          digitalWrite(humidificador, LOW);
          contador_iniciado_humidificador=0;
          contador_iniciado_humidificador_en_OFF=0;
          activado_flag_humidificador=true;
}else{
  if (flag_hay_que_humidificar == true){

    if((activado_flag_humidificador==true) && (contador_iniciado_humidificador_en_OFF==0)){
        start_humidificador();
    }else if((activado_flag_humidificador==false) && (contador_iniciado_humidificador==0)){
        stop_humidificador();
    }


  }else{
    digitalWrite(humidificador, LOW);
    activado_flag_humidificador=true;
    contador_iniciado_humidificador_en_OFF=0;
    contador_iniciado_humidificador=0;
  }
}//if habilitado humidif
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION DESCONECTO_RELES
// funcion para apagar los reles directamente en el puente en H
////////////////////////////////////////////////////////////////////////////////////////////////////////////
void desconecto_reles(){
    digitalWrite(rele_A, RELE_APAGADO);
    digitalWrite(rele_B, RELE_APAGADO);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION ACTIVO_FLAG_PELTIER_FRIO
// funcion para indicar a la funcion peltier que podemos enfriar
////////////////////////////////////////////////////////////////////////////////////////////////////////////
void activo_flag_peltier_frio(){
    activado_reles_flag_frio = true;
    activado_reles_flag_calor = false;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION ACTIVO_FLAG_PELTIER_CALOR
// funcion para indicar a la funcion peltier que podemos calentar
////////////////////////////////////////////////////////////////////////////////////////////////////////////
void activo_flag_peltier_calor(){
    activado_reles_flag_frio = false;
    activado_reles_flag_calor = true;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION DESACTIVO_FLAG_PELTIER_FRIO
// funcion para indicar a la funcion peltier que desconecte la peltier
// Ademas activa un pequeño retardo para que la peltier se recupere antes de volver a conectarla
////////////////////////////////////////////////////////////////////////////////////////////////////////////
void desactivo_flag_peltier(){
    int *mi_variable = &contador_iniciado_reles;
    comienza_retardo(RETARDO_CONEXION_RELES, mi_variable);
    activado_reles_flag_frio = false;
    activado_reles_flag_calor = false;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION PELTIER
// funcion para poner a la peltier en frio, en calor o desconectada, siempre que el retardo de la funcion
desconexion haya finalizado
```

```
// MUY IMPORTANTE: No hay que activar nunca el RELE_A y el RELE_B a la vez, poque hacemos un corto en la fuente de 12V.
// Siempre asegurarse del orden de conexion o desconectar todo primero. Primero apagar un rele y luego activar el otro.
////////////////////////////////////////////////////////////////////////////////////////////////////
void peltier(){  //activara la peltier en el estado que indiquen los flags
   if(contador_iniciado_reles!=0){
   }else if((activado_reles_flag_frio==true) && (activado_reles_flag_calor==false)){

         digitalWrite(rele_B, RELE_APAGADO);
         digitalWrite(rele_A, RELE_ACTIVO);

   }else if((activado_reles_flag_frio==false) && (activado_reles_flag_calor==true)){

         digitalWrite(rele_A, RELE_APAGADO);
         digitalWrite(rele_B, RELE_ACTIVO);

   }else if((activado_reles_flag_frio==false) && (activado_reles_flag_calor==false)){

         digitalWrite(rele_A, RELE_APAGADO);
         digitalWrite(rele_B, RELE_APAGADO);

   }
}

/
**********************************************************************************************************/
////////////////////////////////////////////////////////////////////////////////////////////////////
/
**********************************************************************************************************/
/
**********************************************************************************************************/
/
**********************************************************************************************************/
//   INICIO DE FUNCIONES Y DATOS FUZZY EMPLEADOS EN NUESTRO SISTEMA DE CONTROL
//      VARIABLE SOBRE LA QUE ACTUA: Caudal de aire de la Bomba Aireadora
/
**********************************************************************************************************/
/
**********************************************************************************************************/
/
**********************************************************************************************************/
////////////////////////////////////////////////////////////////////////////////////////////////////
//INICIALIZACION DE OBJETOS FUZZY
////////////////////////////////////////////////////////////////////////////////////////////////////
int fis_gcI;  // Numero de entradas al fuzzy inference system
int fis_gcO;  // Numero de outputs al fuzzy inference system
int fis_gcR;  // Numero de reglas al fuzzy inference system

FIS_TYPE g_fisInput_i1[fis_gcI_i1];
FIS_TYPE g_fisOutput_i1[fis_gcO_i1];

FIS_TYPE g_fisInput_i2[fis_gcI_i2];
FIS_TYPE g_fisOutput_i2[fis_gcO_i2];

FIS_TYPE g_fisInput_i3[fis_gcI_i3];
FIS_TYPE g_fisOutput_i3[fis_gcO_i3];

FIS_TYPE *g_fisInput[2];
FIS_TYPE *g_fisOutput[1];

//*****************************************************************
```

```
// Funciones de soporte para Fuzzy Inference System
//*********************************************************************
// Trapezoidal Member Function
FIS_TYPE fis_trapmf(FIS_TYPE x, FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2], d = p[3];
    FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ? 0 : ((c != d) ? ((d - x) / (d - c)) : 0)));
    FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ? 0 : ((a != b) ? ((x - a) / (b - a)) : 0)));
    return (FIS_TYPE) min(t1, t2);
}

// Triangular Member Function
FIS_TYPE fis_trimf(FIS_TYPE x, FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2];
    FIS_TYPE t1 = (x - a) / (b - a);
    FIS_TYPE t2 = (c - x) / (c - b);
    if ((a == b) && (b == c)) return (FIS_TYPE) (x == a);
    if (a == b) return (FIS_TYPE) (t2*(b <= x)*(x <= c));
    if (b == c) return (FIS_TYPE) (t1*(a <= x)*(x <= b));
    t1 = min(t1, t2);
    return (FIS_TYPE) max(t1, 0);
}

//Funcion MIN
FIS_TYPE fis_min(FIS_TYPE a, FIS_TYPE b)
{
    return min(a, b);
}

//Funcion MAX
FIS_TYPE fis_max(FIS_TYPE a, FIS_TYPE b)
{
    return max(a, b);
}

FIS_TYPE fis_array_operation(FIS_TYPE *array, int size, _FIS_ARR_OP pfnOp)
{
    int i;
    FIS_TYPE ret = 0;

    if (size == 0) return ret;
    if (size == 1) return array[0];

    ret = array[0];
    for (i = 1; i < size; i++)
    {
        ret = (*pfnOp)(ret, array[i]);
    }

    return ret;
}


//*********************************************************************
// Data for Fuzzy Inference System
//*********************************************************************
// Pointers to the implementations of member functions
_FIS_MF fis_gMF[] =
{
    fis_trapmf, fis_trimf
};

int fis_gIMFCount_i1[] = { 3, 3 }; // Count of member function for each Input
int fis_gOMFCount_i1[] = { 3 }; // Count of member function for each Output

int fis_gIMFCount_i2[] = { 3 }; // Count of member function for each Input
```

```
int fis_gOMFCount_i2[] = { 3 }; // Count of member function for each Output

int fis_gIMFCount_i3[] = { 3, 3 }; // Count of member function for each Input
int fis_gOMFCount_i3[] = { 3 }; // Count of member function for each Output

int *fis_gIMFCount;
int *fis_gOMFCount;

//#################################################################################
##
//#################################################################################
##
FIS_TYPE* fis_gMFI0Coeff[5]; // = { fis_gMFI0Coeff1_i1, fis_gMFI0Coeff2_i1, fis_gMFI0Coeff3_i1, NULL };
FIS_TYPE* fis_gMFI1Coeff[5]; // = { fis_gMFI1Coeff1_i1, fis_gMFI1Coeff2_i1, fis_gMFI1Coeff3_i1, NULL  };
FIS_TYPE** fis_gMFICoeff[2]; // = { fis_gMFI0Coeff_i1, fis_gMFI1Coeff_i1, NULL  };

FIS_TYPE* fis_gMFO0Coeff[3]; // = { fis_gMFO0Coeff1, fis_gMFO0Coeff2, fis_gMFO0Coeff3, NULL  };
FIS_TYPE** fis_gMFOCoeff[1]; // = { fis_gMFO0Coeff, NULL  };

int* fis_gMFI[2]; // = { fis_gMFI0_i1, fis_gMFI1_i1, NULL };

int* fis_gMFO[1]; // = { fis_gMFO0_i1, NULL };

FIS_TYPE fis_gRWeight[5]; // = { 1, 1, 1, 1, 1 , NULL};
int fis_gRType[5]; // = { 1, 1, 1, 1, 1 , NULL};

int* fis_gRI[5]; // = { fis_gRI0_i1, fis_gRI1_i1, fis_gRI2_i1, fis_gRI3_i1, fis_gRI4_i1, NULL  };

int* fis_gRO[5]; // = { fis_gRO0_i1, fis_gRO1_i1, fis_gRO2_i1, fis_gRO3_i1, fis_gRO4_i1, NULL  };


/*****************************************************************************/
/*INTERVALO 1*/
/*****************************************************************************/

// Coefficients for the Input Member Functions
FIS_TYPE fis_gMFI0Coeff1_i1[] = { -8.6, -5.4, -1.5, 0 };
FIS_TYPE fis_gMFI0Coeff2_i1[] = { -0.8, 0, 0.8 };
FIS_TYPE fis_gMFI0Coeff3_i1[] = { 0, 1.5, 5.4, 8.6 };
//FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1_i1, fis_gMFI0Coeff2_i1, fis_gMFI0Coeff3_i1, NULL };
FIS_TYPE fis_gMFI1Coeff1_i1[] = { -27.52, -17.28, -5, 0 };
FIS_TYPE fis_gMFI1Coeff2_i1[] = { -5, 0, 5 };
FIS_TYPE fis_gMFI1Coeff3_i1[] = { 0, 5, 17.28, 27.52 };
//FIS_TYPE* fis_gMFI1Coeff[] = { fis_gMFI1Coeff1_i1, fis_gMFI1Coeff2_i1, fis_gMFI1Coeff3_i1, NULL  };
//FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff_i1, fis_gMFI1Coeff_i1, NULL  };

// Coefficients for the Output Member Functions
FIS_TYPE fis_gMFO0Coeff1_i1[] = { -24, 0, 24 };
FIS_TYPE fis_gMFO0Coeff2_i1[] = { 6, 30, 54 };
FIS_TYPE fis_gMFO0Coeff3_i1[] = { 36, 60, 84 };
//FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1, fis_gMFO0Coeff2, fis_gMFO0Coeff3, NULL  };
//FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff, NULL  };

// Input membership function set
int fis_gMFI0_i1[] = { 0, 1, 0 }; //trapecio, triangulo, trapecio
int fis_gMFI1_i1[] = { 0, 1, 0 }; //trapecio, triangulo, trapecio
//int* fis_gMFI[] = { fis_gMFI0_i1, fis_gMFI1_i1, NULL };

// Output membership function set
int fis_gMFO0_i1[] = { 1, 1, 1 }; //triangulo, triangulo, triangulo
//int* fis_gMFO[] = { fis_gMFO0_i1, NULL };

// Rule Weights
//FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1 , NULL};

// Rule Type
//int fis_gRType[] = { 1, 1, 1, 1, 1 , NULL};

// Rule Inputs
```

```
int fis_gRI0_i1[] = { 1, 1 };
int fis_gRI1_i1[] = { 2, 1 };
int fis_gRI2_i1[] = { 3, 1 };
int fis_gRI3_i1[] = { 3, 2 };
int fis_gRI4_i1[] = { 3, 3 };
//int* fis_gRI[] = { fis_gRI0_i1, fis_gRI1_i1, fis_gRI2_i1, fis_gRI3_i1, fis_gRI4_i1, NULL  };

// Rule Outputs
int fis_gRO0_i1[] = { 1 };
int fis_gRO1_i1[] = { 2 };
int fis_gRO2_i1[] = { 2 };
int fis_gRO3_i1[] = { 3 };
int fis_gRO4_i1[] = { 3 };
//int* fis_gRO[] = { fis_gRO0_i1, fis_gRO1_i1, fis_gRO2_i1, fis_gRO3_i1, fis_gRO4_i1, NULL  };

/**********************************************************************/
/* FIN INTERVALO 1*/
/**********************************************************************/

/**********************************************************************/
/*INTERVALO 2*/
/**********************************************************************/
// Coefficients for the Input Member Functions
FIS_TYPE fis_gMFI0Coeff1_i2[] = { -8.6, -5.4, -1.5, 0 };
FIS_TYPE fis_gMFI0Coeff2_i2[] = { -0.8, 0, 0.8 };
FIS_TYPE fis_gMFI0Coeff3_i2[] = { 0, 1.5, 5.4, 8.6 };
//FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1_i2, fis_gMFI0Coeff2_i2, fis_gMFI0Coeff3_i2, NULL  };
//FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff , NULL  };

FIS_TYPE fis_gMFI1Coeff1_i2[] = { -27.52, -17.28, -5, 0 };
FIS_TYPE fis_gMFI1Coeff2_i2[] = { -5, 0, 5 };
FIS_TYPE fis_gMFI1Coeff3_i2[] = { 0, 5, 17.28, 27.52 };


// Coefficients for the Output Member Functions
FIS_TYPE fis_gMFO0Coeff1_i2[] = { -24, 0, 24 };
FIS_TYPE fis_gMFO0Coeff2_i2[] = { 6, 30, 54 };
FIS_TYPE fis_gMFO0Coeff3_i2[] = { 36, 60, 84 };
//FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1_i2, fis_gMFO0Coeff2_i2, fis_gMFO0Coeff3_i2 , NULL  };
//FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff , NULL  };

// Input membership function set
int fis_gMFI0_i2[] = { 0, 1, 0 };
int fis_gMFI1_i2[] = { 0, 1, 0 }; /////
//int* fis_gMFI[] = { fis_gMFI0_i2, NULL  };

// Output membership function set
int fis_gMFO0_i2[] = { 1, 1, 1 };
//int* fis_gMFO[] = { fis_gMFO0_i2, NULL  };

// Rule Weights
//FIS_TYPE fis_gRWeight[] = { 1, 1, 1 , NULL};

// Rule Type
//int fis_gRType[] = { 1, 1, 1 , NULL};

// Rule Inputs
int fis_gRI0_i2[] = { 1, 0 };
int fis_gRI1_i2[] = { 2, 0 };
int fis_gRI2_i2[] = { 3, 0 };
int fis_gRI3_i2[] = { 0, 0 };
int fis_gRI4_i2[] = { 0, 0 };
//int* fis_gRI[] = { fis_gRI0_i2, fis_gRI1_i2, fis_gRI2_i2 , NULL  };

// Rule Outputs
int fis_gRO0_i2[] = { 1 };
int fis_gRO1_i2[] = { 2 };
```

```
int fis_gRO2_i2[] = { 1 };
int fis_gRO3_i2[] = { 0 };
int fis_gRO4_i2[] = { 0 };
//int* fis_gRO[] = { fis_gRO0_i2, fis_gRO1_i2, fis_gRO2_i2 , NULL  };

/*****************************************************************/
/* FIN INTERVALO 2*/
/*****************************************************************/

/*****************************************************************/
/*INTERVALO 3*/
/*****************************************************************/
// Coefficients for the Input Member Functions
FIS_TYPE fis_gMFI0Coeff1_i3[] = { -8.6, -5.4, -1.5, 0 };
FIS_TYPE fis_gMFI0Coeff2_i3[] = { -0.8, 0, 0.8 };
FIS_TYPE fis_gMFI0Coeff3_i3[] = { 0, 1.5, 5.4, 8.6 };
//FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1_i2, fis_gMFI0Coeff2_i2, fis_gMFI0Coeff3_i2, NULL  };
//FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff , NULL  };

FIS_TYPE fis_gMFI1Coeff1_i3[] = { -27.52, -17.28, -5, 0 };
FIS_TYPE fis_gMFI1Coeff2_i3[] = { -5, 0, 5 };
FIS_TYPE fis_gMFI1Coeff3_i3[] = { 0, 5, 17.28, 27.52 };


// Coefficients for the Output Member Functions
FIS_TYPE fis_gMFO0Coeff1_i3[] = { -24, 0, 24 };
FIS_TYPE fis_gMFO0Coeff2_i3[] = { 6, 30, 54 };
FIS_TYPE fis_gMFO0Coeff3_i3[] = { 36, 60, 84 };
//FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1_i2, fis_gMFO0Coeff2_i2, fis_gMFO0Coeff3_i2 , NULL  };
//FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff , NULL  };

// Input membership function set
int fis_gMFI0_i3[] = { 0, 1, 0 };
int fis_gMFI1_i3[] = { 0, 1, 0 }; /////
//int* fis_gMFI[] = { fis_gMFI0_i2, NULL  };

// Output membership function set
int fis_gMFO0_i3[] = { 1, 1, 1 };
//int* fis_gMFO[] = { fis_gMFO0_i2, NULL  };

// Rule Weights
//FIS_TYPE fis_gRWeight[] = { 1, 1, 1 , NULL};

// Rule Type
//int fis_gRType[] = { 1, 1, 1 , NULL};

// Rule Inputs
int fis_gRI0_i3[] = { 1, 2 };
int fis_gRI1_i3[] = { 2, 2 };
int fis_gRI2_i3[] = { 3, 0 };
int fis_gRI3_i3[] = { 0, 0 };
int fis_gRI4_i3[] = { 0, 0 };
//int* fis_gRI[] = { fis_gRI0_i2, fis_gRI1_i2, fis_gRI2_i2 , NULL  };

// Rule Outputs
int fis_gRO0_i3[] = { 3 };
int fis_gRO1_i3[] = { 2 };
int fis_gRO2_i3[] = { 1 };
int fis_gRO3_i3[] = { 0 };
int fis_gRO4_i3[] = { 0 };
//int* fis_gRO[] = { fis_gRO0_i2, fis_gRO1_i2, fis_gRO2_i2 , NULL  };

/*****************************************************************/
/* FIN INTERVALO 3*/
/*****************************************************************/


// Input range Min
```

```
FIS_TYPE fis_gIMin[] = { -5, -16 };

// Input range Max
FIS_TYPE fis_gIMax[] = { 5, 16 };

// Output range Min
FIS_TYPE fis_gOMin[] = { 0 };

// Output range Max
FIS_TYPE fis_gOMax[] = { 60 };

//************************************************************************
// Data dependent support functions for Fuzzy Inference System
//************************************************************************
FIS_TYPE fis_MF_out(FIS_TYPE** fuzzyRuleSet, FIS_TYPE x, int o)
{
    FIS_TYPE mfOut;
    int r;

    for (r = 0; r < fis_gcR; ++r)
    {
        int index = fis_gRO[r][o];
        if (index > 0)
        {
            index = index - 1;
            mfOut = (fis_gMF[fis_gMFO[o][index]])(x, fis_gMFOCoeff[o][index]);
        }
        else if (index < 0)
        {
            index = -index - 1;
            mfOut = 1 - (fis_gMF[fis_gMFO[o][index]])(x, fis_gMFOCoeff[o][index]);
        }
        else
        {
            mfOut = 0;
        }

        fuzzyRuleSet[0][r] = fis_min(mfOut, fuzzyRuleSet[1][r]);
    }
    return fis_array_operation(fuzzyRuleSet[0], fis_gcR, fis_max);
}

//Funcion para el calculo del centroide en el proceso de defuzzy
FIS_TYPE fis_defuzz_centroid(FIS_TYPE** fuzzyRuleSet, int o)
{
    FIS_TYPE step = (fis_gOMax[o] - fis_gOMin[o]) / (FIS_RESOLUSION - 1);
    FIS_TYPE area = 0;
    FIS_TYPE momentum = 0;
    FIS_TYPE dist, slice;
    int i;

    // calculate the area under the curve formed by the MF outputs
    for (i = 0; i < FIS_RESOLUSION; ++i){
        dist = fis_gOMin[o] + (step * i);
        slice = step * fis_MF_out(fuzzyRuleSet, dist, o);
        area += slice;
        momentum += slice*dist;
    }

    return ((area == 0) ? ((fis_gOMax[o] + fis_gOMin[o]) / 2) : (momentum / area));
}

//************************************************************************
// Fuzzy Inference System
// Calcula el valor de salida
//************************************************************************
void fis_evaluate()
```

```
{

   FIS_TYPE fuzzyInput0[] = { 0, 0, 0 };
   FIS_TYPE fuzzyInput1[] = { 0, 0, 0 };
   FIS_TYPE* fuzzyInput[] = {fuzzyInput0, fuzzyInput1};
   //FIS_TYPE fuzzyOutput0[] = { 0, 0, 0 };
   //FIS_TYPE* fuzzyOutput[] = { fuzzyOutput0};
   FIS_TYPE fuzzyRules[MAXIMO_DE_RULES] = { 0 };
   FIS_TYPE fuzzyFires[MAXIMO_DE_RULES] = { 0 };
   FIS_TYPE* fuzzyRuleSet[] = { fuzzyRules, fuzzyFires };
   FIS_TYPE sW = 0;


FIS_TYPE g_fisInput_local[] = { *g_fisInput[0], *g_fisInput[1] };
FIS_TYPE g_fisOutput_local[] = { *g_fisOutput[0] };

   int i, j, r, o;
   // Transforming input to fuzzy Input
   for (i = 0; i < fis_gcI; ++i)
   {
      for (j = 0; j < fis_gIMFCount[i]; ++j)
      {
         fuzzyInput[i][j] =
            (fis_gMF[fis_gMFI[i][j]])(g_fisInput_local[i], fis_gMFICoeff[i][j]);
      }
   }

   int index = 0;
   for (r = 0; r < fis_gcR; ++r)
   {
      if (fis_gRType[r] == 1)
      {
         fuzzyFires[r] = FIS_MAX;
         for (i = 0; i < fis_gcI; ++i)
         {
            index = fis_gRI[r][i];
            if (index > 0)
               fuzzyFires[r] = fis_min(fuzzyFires[r], fuzzyInput[i][index - 1]);
            else if (index < 0)
               fuzzyFires[r] = fis_min(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
            else
               fuzzyFires[r] = fis_min(fuzzyFires[r], 1);

         }
      }
      else
      {
         fuzzyFires[r] = FIS_MIN;
         for (i = 0; i < fis_gcI; ++i)
         {
            index = fis_gRI[r][i];
            if (index > 0)
               fuzzyFires[r] = fis_max(fuzzyFires[r], fuzzyInput[i][index - 1]);
            else if (index < 0)
               fuzzyFires[r] = fis_max(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
            else
               fuzzyFires[r] = fis_max(fuzzyFires[r], 0);
         }
      }

      fuzzyFires[r] = fis_gRWeight[r] * fuzzyFires[r];
      sW += fuzzyFires[r];
   }

   if (sW == 0)
   {
      for (o = 0; o < fis_gcO; ++o)
      {
         g_fisOutput_local[o] = ((fis_gOMax[o] + fis_gOMin[o]) / 2);
```

```
      }
   }
   else
   {
      for (o = 0; o < fis_gcO; ++o)
      {
         g_fisOutput_local[o] = fis_defuzz_centroid(fuzzyRuleSet, o);
         *g_fisOutput[o]=g_fisOutput_local[o];
      }
   }
}

////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION INTERVALOS
// funcion que nos devuelve el valor calculado FUZZY del sistema de control.
// PARAMETROS DE ENTRADA: El intervalo en el que haremos el calculo FUZZY (1, 2 o 3) y el DELTA_T y el
DELTA_H para
// la tendencia hacia arriba o hacia abajo de la temperatura y de la humedad
////////////////////////////////////////////////////////////////////////////////////////////////////////
int intervalos(int intervalo, float delta_temp, float delta_humi){

//float caudal_output=0;
if (flag_habilitado_fuzzy==true){
   switch(intervalo){
   case 1:

/*INTERVALO 1*/
fis_gcI=fis_gcI_i1;  // Number of inputs to the fuzzy inference system
fis_gcO=fis_gcO_i1;  // Number of outputs to the fuzzy inference system
fis_gcR=fis_gcR_i1;  // Number of rules to the fuzzy inference system


g_fisInput[0] = &g_fisInput_i1[0];
g_fisInput[1] = &g_fisInput_i1[1];
g_fisOutput[0] = &g_fisOutput_i1[0];
fis_gIMFCount=&fis_gIMFCount_i1[0];
fis_gOMFCount=&fis_gOMFCount_i1[0];


fis_gMFI0Coeff[0] = &fis_gMFI0Coeff1_i1[0];
fis_gMFI0Coeff[1] = &fis_gMFI0Coeff2_i1[0];
fis_gMFI0Coeff[2] = &fis_gMFI0Coeff3_i1[0];
fis_gMFI1Coeff[0] = &fis_gMFI1Coeff1_i1[0];
fis_gMFI1Coeff[1] = &fis_gMFI1Coeff2_i1[0];
fis_gMFI1Coeff[2] = &fis_gMFI1Coeff3_i1[0];
fis_gMFICoeff[0] = &fis_gMFI0Coeff[0];
fis_gMFICoeff[1] = &fis_gMFI1Coeff[0];

fis_gMFO0Coeff[0] = &fis_gMFO0Coeff1_i1[0];
fis_gMFO0Coeff[1] = &fis_gMFO0Coeff2_i1[0];
fis_gMFO0Coeff[2] = &fis_gMFO0Coeff3_i1[0];
fis_gMFOCoeff[0] = &fis_gMFO0Coeff[0];

fis_gMFI[0] = &fis_gMFI0_i1[0];
fis_gMFI[1] = &fis_gMFI1_i1[0];

fis_gMFO[0] = &fis_gMFO0_i1[0];

fis_gRWeight[0] = 1;
fis_gRWeight[1] = 1;
fis_gRWeight[2] = 1;
fis_gRWeight[3] = 1;
fis_gRWeight[4] = 1;
fis_gRType[0] = 1;
fis_gRType[1] = 1;
```

```
fis_gRType[2] = 1;
fis_gRType[3] = 1;
fis_gRType[4] = 1;

fis_gRI[0] = &fis_gRI0_i1[0];
fis_gRI[1] = &fis_gRI1_i1[0];
fis_gRI[2] = &fis_gRI2_i1[0];
fis_gRI[3] = &fis_gRI3_i1[0];
fis_gRI[4] = &fis_gRI4_i1[0];

fis_gRO[0] = &fis_gRO0_i1[0];
fis_gRO[1] = &fis_gRO1_i1[0];
fis_gRO[2] = &fis_gRO2_i1[0];
fis_gRO[3] = &fis_gRO3_i1[0];
fis_gRO[4] = &fis_gRO4_i1[0];

            // Read Input: delta_T
               g_fisInput_i1[0] = delta_temp;
            // Read Input: delta_H
               g_fisInput_i1[1] = delta_humi;
            //float caudal_output = 0;
               g_fisOutput_i1[0] = 44;
               fis_evaluate();
            // Set output value: caudal
      return (int(g_fisOutput_i1[0]));
   break;
   case 2:

/*INTERVALO 2*/
fis_gcI=fis_gcI_i2;  // Number of inputs to the fuzzy inference system
fis_gcO=fis_gcO_i2;  // Number of outputs to the fuzzy inference system
fis_gcR=fis_gcR_i2;  // Number of rules to the fuzzy inference system

g_fisInput[0] = &g_fisInput_i2[0];
g_fisInput[1] = &g_fisInput_i2[1];
g_fisOutput[0] = &g_fisOutput_i2[0];
fis_gIMFCount=&fis_gIMFCount_i2[0];
fis_gOMFCount=&fis_gOMFCount_i2[0];

fis_gMFI0Coeff[0] = &fis_gMFI0Coeff1_i2[0];
fis_gMFI0Coeff[1] = &fis_gMFI0Coeff2_i2[0];
fis_gMFI0Coeff[2] = &fis_gMFI0Coeff3_i2[0];
fis_gMFI1Coeff[0] = &fis_gMFI1Coeff1_i2[0];
fis_gMFI1Coeff[1] = &fis_gMFI1Coeff2_i2[0];
fis_gMFI1Coeff[2] = &fis_gMFI1Coeff3_i2[0];
fis_gMFICoeff[0] = &fis_gMFI0Coeff[0];
fis_gMFICoeff[1] = &fis_gMFI1Coeff[0];

fis_gMFO0Coeff[0] = &fis_gMFO0Coeff1_i2[0];
fis_gMFO0Coeff[1] = &fis_gMFO0Coeff2_i2[0];
fis_gMFO0Coeff[2] = &fis_gMFO0Coeff3_i2[0];
fis_gMFOCoeff[0] = &fis_gMFO0Coeff[0];

fis_gMFI[0] = &fis_gMFI0_i2[0];
fis_gMFI[1] = &fis_gMFI1_i2[0];

fis_gMFO[0] = &fis_gMFO0_i2[0];

fis_gRWeight[0] = 1;
fis_gRWeight[1] = 1;
fis_gRWeight[2] = 1;
fis_gRWeight[3] = 1;
fis_gRWeight[4] = 1;
fis_gRType[0] = 1;
fis_gRType[1] = 1;
fis_gRType[2] = 1;
fis_gRType[3] = 1;
fis_gRType[4] = 1;
```

```
fis_gRI[0] = &fis_gRI0_i2[0];
fis_gRI[1] = &fis_gRI1_i2[0];
fis_gRI[2] = &fis_gRI2_i2[0];
fis_gRI[3] = &fis_gRI3_i2[0];
fis_gRI[4] = &fis_gRI4_i2[0];

fis_gRO[0] = &fis_gRO0_i2[0];
fis_gRO[1] = &fis_gRO1_i2[0];
fis_gRO[2] = &fis_gRO2_i2[0];
fis_gRO[3] = &fis_gRO3_i2[0];
fis_gRO[4] = &fis_gRO4_i2[0];

            // Read Input: delta_T
               g_fisInput_i2[0] = delta_temp;
            // Read Input: delta_H
               g_fisInput_i2[1] = delta_humi;
            //float caudal_output = 0;
               g_fisOutput_i2[0] = 44;
               fis_evaluate();
            // Set output value: caudal
      return (int(g_fisOutput_i2[0]));
   break;
   case 3:

/*INTERVALO 3*/
fis_gcI=fis_gcI_i3;  // Number of inputs to the fuzzy inference system
fis_gcO=fis_gcO_i3;  // Number of outputs to the fuzzy inference system
fis_gcR=fis_gcR_i3;  // Number of rules to the fuzzy inference system


g_fisInput[0] = &g_fisInput_i3[0];
g_fisInput[1] = &g_fisInput_i3[1];
g_fisOutput[0] = &g_fisOutput_i3[0];
fis_gIMFCount=&fis_gIMFCount_i3[0];
fis_gOMFCount=&fis_gOMFCount_i3[0];


fis_gMFI0Coeff[0] = &fis_gMFI0Coeff1_i3[0];
fis_gMFI0Coeff[1] = &fis_gMFI0Coeff2_i3[0];
fis_gMFI0Coeff[2] = &fis_gMFI0Coeff3_i3[0];
fis_gMFI1Coeff[0] = &fis_gMFI1Coeff1_i3[0];
fis_gMFI1Coeff[1] = &fis_gMFI1Coeff2_i3[0];
fis_gMFI1Coeff[2] = &fis_gMFI1Coeff3_i3[0];
fis_gMFICoeff[0] = &fis_gMFI0Coeff[0];
fis_gMFICoeff[1] = &fis_gMFI1Coeff[0];

fis_gMFO0Coeff[0] = &fis_gMFO0Coeff1_i3[0];
fis_gMFO0Coeff[1] = &fis_gMFO0Coeff2_i3[0];
fis_gMFO0Coeff[2] = &fis_gMFO0Coeff3_i3[0];
fis_gMFOCoeff[0] = &fis_gMFO0Coeff[0];

fis_gMFI[0] = &fis_gMFI0_i3[0];
fis_gMFI[1] = &fis_gMFI1_i3[0];

fis_gMFO[0] = &fis_gMFO0_i3[0];

fis_gRWeight[0] = 1;
fis_gRWeight[1] = 1;
fis_gRWeight[2] = 1;
fis_gRWeight[3] = 1;
fis_gRWeight[4] = 1;
fis_gRType[0] = 1;
fis_gRType[1] = 1;
fis_gRType[2] = 1;
fis_gRType[3] = 1;
fis_gRType[4] = 1;
```

```
fis_gRI[0] = &fis_gRI0_i3[0];
fis_gRI[1] = &fis_gRI1_i3[0];
fis_gRI[2] = &fis_gRI2_i3[0];
fis_gRI[3] = &fis_gRI3_i3[0];
fis_gRI[4] = &fis_gRI4_i3[0];

fis_gRO[0] = &fis_gRO0_i3[0];
fis_gRO[1] = &fis_gRO1_i3[0];
fis_gRO[2] = &fis_gRO2_i3[0];
fis_gRO[3] = &fis_gRO3_i3[0];
fis_gRO[4] = &fis_gRO4_i3[0];

            // Read Input: delta_T
               g_fisInput_i3[0] = delta_temp;
            // Read Input: delta_H
               g_fisInput_i3[1] = delta_humi;
            //float caudal_output = 0;
               g_fisOutput_i3[0] = 44;
               fis_evaluate();
            // Set output value: caudal
      return (int(g_fisOutput_i3[0]));
   break;
   }//switch
}//if fuzzy habilitado

//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm

}//intervalos

////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// FIN DE FUNCIONES Y DATOS FUZZY
////////////////////////////////////////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// CONFIGURACION INICIAL DEL ARDUINO
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void setup()
{
 cli();

 // initialize serial communication at 9600 bits per second:

  Serial.begin(9600);
  dht.begin();

  pinMode(fuzzy_aire_bomba, OUTPUT);// output pin for OCR2B
  pinMode(rele_A, OUTPUT);// output pin for OCR2B
  pinMode(rele_B, OUTPUT);// output pin for OCR2B
  digitalWrite(fuzzy_aire_bomba, LOW);
  digitalWrite(rele_A, RELE_APAGADO);
  digitalWrite(rele_B, RELE_APAGADO);

  pinMode(humidificador, OUTPUT);// output pin for OCR2B
  digitalWrite(humidificador, LOW);

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);

/*   El reloj de tiempo real se basa en interrupciones internas.

    Cuando el Timer1 llega a  la cifra 15624, genera una
    interrupción y se pone a cero el contador volviendo a contar
    otra vez.

    Este numero se saca de la siguiente forma:

     Frecuencia del procesador 16Mhz dividido por el prescaler
```

```
    puesto a 1024 nos da una frecuencia de 15625, y a esta cifra le
    debemos restar 1, con lo cual nos da la el numero 15624.
*/


  TCCR1A=0;
  TCCR1B=0;
  OCR1A=15624;
  TCCR1B |= (1<<WGM12);
  TCCR1B |= (1<<CS10);
  TCCR1B |= (1<<CS12);
  TIMSK1=(1<<OCIE1A);
  sei();
 horas=0;
 minutos=0;
 segundos=0;
 lcd.begin(16,2);        // initialize the lcd for 16 chars 2 lines, turn on backlight
 lcd.backlight();
////////////////////////////////////////////////////////////////////////////
//TEST LCD Y LED INCUBADORA
// 3 segundos encendido y con el texto Probando...
////////////////////////////////////////////////////////////////////////////
                                digitalWrite(ledPin, HIGH);

  // Print a message to the LCD.
  lcd.setCursor(0, 0);
  lcd.print(F("Probando...     "));
  delay(3000);
  lcd.noBacklight();

                                digitalWrite(ledPin, LOW);


////////////////////////////////////////////////////////////////////////////
//TEST DEL HUMIDIFICADOR
// 3 veces ON-OFF en 3 segundos
////////////////////////////////////////////////////////////////////////////
//start_humidificador();
  pinMode(humidificador, OUTPUT);// output pin for OCR2B

  // Set up the 107kHz output
  TCCR2A = _BV(COM2A1) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
  TCCR2B = _BV(WGM22) | _BV(CS20);
  OCR2A = 149; //159
  OCR2B = 74; //79

  delay(500);


//stop_humidificador();
  digitalWrite(humidificador, LOW);
  delay(500);



//start_humidificador();
  pinMode(humidificador, OUTPUT);// output pin for OCR2B

  // Set up the 107kHz output
  TCCR2A = _BV(COM2A1) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
  TCCR2B = _BV(WGM22) | _BV(CS20);
  OCR2A = 149; //159
  OCR2B = 74; //79

  delay(500);
```

```
//stop_humidificador();
  digitalWrite(humidificador, LOW);
  delay(500);


//start_humidificador();
  pinMode(humidificador, OUTPUT);// output pin for OCR2B

  // Set up the 107kHz output
  TCCR2A = _BV(COM2A1) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
  TCCR2B = _BV(WGM22) | _BV(CS20);
  OCR2A = 149; //159
  OCR2B = 74; //79

  delay(500);


//stop_humidificador();
  digitalWrite(humidificador, LOW);
////////////////////////////////////////////////////////////////////////
//FIN DEL TEST HUMIDIFICADOR
////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////
//TEST DE LECTURA DEL SENSOR DHT
// Primera lectura de Temperatura y de Humedad en el Display con los modos apagados
////////////////////////////////////////////////////////////////////////
//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm

    lcd.setCursor(0, 1);
    lcd.print(F("T"));
                  temperatura_inicial = (float)dht.readTemperature();
    lcd.print(temperatura_inicial);

    lcd.print(F(" H"));
                  humedad_inicial = (float)dht.readHumidity();
    lcd.print(humedad_inicial);
    lcd.print(F("%"));
////////////////////////////////////////////////////////////////////////
//FIN DEL TEST DEL SENSOR DHT
////////////////////////////////////////////////////////////////////////

}//setup

/*************************************************************************/
/*************************************************************************/
/*************************************************************************/


////////////////////////////////////////////////////////////////////////
//INTERRUPCIONES CADA SEGUNDO Y CONTADORES RETARDOS
////////////////////////////////////////////////////////////////////////
ISR(TIMER1_COMPA_vect)
{

  segundos++;


  ////////////// RETARDO RELES  /////////////////
  //////////////////////////////////////////////
      if(contador_iniciado_reles<=0){
          contador_iniciado_reles = 0;
      }else{
          contador_iniciado_reles--;
      }
```

```
////////////////////////////////////////////////
////////////////////////////////////////////////

//////// TIEMPO EN ON BOMBA DE AIRE //////////
////////////////////////////////////////////////
    if(contador_iniciado_fuzzy_aire_bomba<=0){
       contador_iniciado_fuzzy_aire_bomba = 0;
    }else{
       contador_iniciado_fuzzy_aire_bomba--;
    }
////////////////////////////////////////////////
////////////////////////////////////////////////

//////// TIEMPO EN OFF BOMBA DE AIRE //////////
////////////////////////////////////////////////
    if(contador_iniciado_fuzzy_aire_bomba_en_OFF<=0){
       contador_iniciado_fuzzy_aire_bomba_en_OFF = 0;
    }else{
       contador_iniciado_fuzzy_aire_bomba_en_OFF--;
                    if(contador_iniciado_fuzzy_aire_bomba_en_OFF<=0){
                                temperatura_actual = (float)dht.readTemperature();
                                humedad_actual = (float)dht.readHumidity();
                                delta_temp = temperatura_actual - temperatura_inicial;
                                delta_humedad = humedad_actual - humedad_inicial;
                                temperatura_inicial = temperatura_actual;
                                humedad_inicial = humedad_actual;
                        }//hago los deltas
    }
////////////////////////////////////////////////
////////////////////////////////////////////////

////// TIEMPO DEL HUMIDIFICADOR EN ON    ////////
////////////////////////////////////////////////
    if(contador_iniciado_humidificador<=0){
       contador_iniciado_humidificador = 0;
    }else{
       contador_iniciado_humidificador--;
    }
////////////////////////////////////////////////
////////////////////////////////////////////////

/////// TIEMPO DEL HUMIDIFICADOR EN OFF   ///////
////////////////////////////////////////////////
    if(contador_iniciado_humidificador_en_OFF<=0){
       contador_iniciado_humidificador_en_OFF = 0;
    }else{
       contador_iniciado_humidificador_en_OFF--;
    }
////////////////////////////////////////////////
////////////////////////////////////////////////



bandera1=true;
/*************************************************/
/* La bandera la hago cada 10 segundos ************/
/* La usare para la lectura de los sensores *******/
if((segundos%10) == 1){
  bandera2=true;
}//if
/*************************************************/
/*************************************************/
/* La bandera la hago cada 2 segundos ************/
/* La usare para la lectura de los sensores *******/
if((segundos%2) == 1){
```

```
  bandera3=true;
 }//if
 /*************************************************/



 if(segundos>59)
   {
    segundos=0;
    minutos++;
    if(minutos>59)
       {
        minutos=0;
        horas++;

 //////// TIEMPO MODO FUNCIONAMIENTO /////////
 ////////////////////////////////////////////////
      if(contador_iniciado_horas_modo<=0){
         contador_iniciado_horas_modo = 0;
      }else{
         contador_iniciado_horas_modo--;
      }
 ////////////////////////////////////////////////
 ////////////////////////////////////////////////


 ////////////// TIEMPO FOTOPERIODO ON    //////////
 ////////////////////////////////////////////////
      if(contador_iniciado_fotoperiodo<=0){
         contador_iniciado_fotoperiodo = 0;
      }else{
         contador_iniciado_fotoperiodo--;
      }
 ////////////////////////////////////////////////
 ////////////////////////////////////////////////

 ////////////// TIEMPO FOTOPERIODO OFF    //////////
 ////////////////////////////////////////////////
      if(contador_iniciado_fotoperiodo_en_OFF<=0){
         contador_iniciado_fotoperiodo_en_OFF = 0;
      }else{
         contador_iniciado_fotoperiodo_en_OFF--;
      }
 ////////////////////////////////////////////////
 ////////////////////////////////////////////////



         if(horas>23)
           {
            segundos=0;
            minutos=0;
            horas=0;
           }
       }
   }
}



////////////////////////////////////////////////////////////////////////////////////////////////////////////
// INICIO  LOOP - MAIN
////////////////////////////////////////////////////////////////////////////////////////////////////////////

void loop()
{


////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
// MENU DEL LCD
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//CADA SEGUNDO

if(bandera1==true){


      valor_boton_CHG=digitalRead(boton_CHG);
      valor_boton_MENU=digitalRead(boton_MENU);

  if ((valor_boton_CHG==HIGH) && (valor_boton_MENU==HIGH)){
          //no hago nada porque he bloqueado el programa y lo que quiero es que si estoy presionando 10
segundos los dos botones entonces se resetee el display
          //se supone que si hago esto es porque el display LCD hay que reinicializarlo
  }else{

      if(bandera_nivel_menu==0){
          sprintf(VARI,"%02d:%02d:%02d    ",horas,minutos,segundos);
          //Serial.println(VARI); //pintar la HORA
          lcd.setCursor(0, 0);
          lcd.print(VARI);
          //pintar_espacio_vacio_contador();


          if (valor_boton_MENU==HIGH){

              menu=1;
              bandera_nivel_menu=1;
              boton_MENU_primera_vez=true;
              lcd.backlight();

               //Serial.println("Menu Cambiar Min");
               lcd.setCursor(0, 0);
               lcd.print(F("Cambiar Min  "));


          }else if (valor_boton_CHG==HIGH){
              digitalWrite(ledPin,HIGH);
              lcd.backlight();
              contador_LED=0;
          }

      }else if(bandera_nivel_menu==1 && menu==1){

          if (valor_boton_CHG==HIGH){
            //Serial.println("Menu Cambiar Minutos");
/*
              lcd.setCursor(0, 0);
              lcd.print(F("Cambiar Min  "));
*/
              bandera_nivel_menu=2;
          }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

              bandera_nivel_menu=1;
              boton_MENU_primera_vez=true;
              menu=(menu+1)%12; //siguiente Menu
              //Serial.println("Menu Cambiar Horas");
              lcd.setCursor(0, 0);
              lcd.print(F("Cambiar Hora "));
          }

      }else if(bandera_nivel_menu==1 && menu==2){

          if (valor_boton_CHG==HIGH){
            //Serial.println("Menu Cambiar Horas");
/*
              lcd.setCursor(0, 0);
```

```
          lcd.print(F("Cambiar Hora "));
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=1;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("Menu Resetear Contador");
                                        if(valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON==99){
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(DESHABILITADO_HUMIDIFICADOR));
                                        }else{
           lcd.setCursor(0, 0);
           lcd.print(F("Humid. "));
                                sprintf(VARI,"%02d", valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON);
                                lcd.print(VARI);
           lcd.print(F("    "));
                                        }
                        }

    }else if(bandera_nivel_menu==1 && menu==3){

        if (valor_boton_CHG==HIGH){
          //Serial.println("Menu Resetear Contador");
/*
                if(valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON==99){
                                lcd.setCursor(0, 0);
                                lcd.print(F(DESHABILITADO_HUMIDIFICADOR));
                }else{
                                        lcd.setCursor(0, 0);
                                        lcd.print(F("Humid. "));
                                        sprintf(VARI,"%02d",
valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON);
                                        lcd.print(VARI);
                                        lcd.print(F("    "));
                }
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=1;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("Menu START Modo AUTO");
           lcd.setCursor(0, 0);
           lcd.print(F("START M AUTO "));
        }


    }else if(bandera_nivel_menu==1 && menu==4){

        if (valor_boton_CHG==HIGH){
          //Serial.println("Menu START Modo AUTO");
/*
          lcd.setCursor(0, 0);
          lcd.print(F("START M AUTO "));
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=1;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("Menu START Modo GROW");
           lcd.setCursor(0, 0);
           lcd.print(("START M GROW "));
        }
```

```
    }else if(bandera_nivel_menu==1 && menu==5){

        if (valor_boton_CHG==HIGH){
          //Serial.println("Menu START Modo GROW");
/*
          lcd.setCursor(0, 0);
          lcd.print(F("START M GROW "));
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=1;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("Menu START Modo Fructif");
           lcd.setCursor(0, 0);
           lcd.print(F("START M FRUCT"));
        }


    }else if(bandera_nivel_menu==1 && menu==6){

        if (valor_boton_CHG==HIGH){
          //Serial.println("Menu START Modo Fructif");
/*
          lcd.setCursor(0, 0);
          lcd.print(F("START M FRUCT"));
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=1;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("STOP");
           lcd.setCursor(0, 0);
           lcd.print(F("STOP        "));
        }

    }else if(bandera_nivel_menu==1 && menu==7){

        if (valor_boton_CHG==HIGH){
          //Serial.println("STOP");
/*
          lcd.setCursor(0, 0);
          lcd.print(F("STOP        "));
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=2;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("HABILITADO/DESHABILITADO FUZZY");
                        if (flag_habilitado_fuzzy==true){
                                        lcd.setCursor(0, 0);
                                        lcd.print(F(HABILITADO_FUZZY));
                        }else{
                                        lcd.setCursor(0, 0);
                                        lcd.print(F(DESHABILITADO_FUZZY));
                        }
        }
    }else if(bandera_nivel_menu==1 && menu==8){ //menu HABILITAR / DESHABILITAR FUZZY


          //Serial.println("HABILITADO/DESHABILITADO FUZZY");
        if (valor_boton_CHG==HIGH){
```

```
/*
                                if (flag_habilitado_fuzzy==true){
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(HABILITADO_FUZZY));
                                }else{
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(DESHABILITADO_FUZZY));
                                }
*/
                                bandera_nivel_menu=2;

        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=2;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("INCREMENTAR CONTADOR");
           lcd.setCursor(0, 0);
                lcd.print(F("INCREMEN CONT"));
        }
    }else if(bandera_nivel_menu==1 && menu==9){ //incrementar


        if (valor_boton_CHG==HIGH){
           //Serial.println("INCREMENTAR CONTADOR");
/*
                lcd.setCursor(0, 0);
                lcd.print(F("INCREMEN CONT"));
*/
          bandera_nivel_menu=2;

        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=2;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("DECREMENTAR_CONTADOR");
           lcd.setCursor(0, 0);
                lcd.print(F("DECREMEN CONT"));
        }
    }else if(bandera_nivel_menu==1 && menu==10){ //incrementar


        if (valor_boton_CHG==HIGH){
           //Serial.println("DECREMENTAR_CONTADOR");
/*
                lcd.setCursor(0, 0);
                lcd.print(F("DECREMEN CONT"));
*/
          bandera_nivel_menu=2;

        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=2;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("SALIR");
           lcd.setCursor(0, 0);
           lcd.print(F("SALIR       "));
        }

    }else if(bandera_nivel_menu==1 && menu==11){ //menu SALIR


       if (valor_boton_CHG==HIGH){
          //Serial.println("Fuera de Menu");
          bandera_nivel_menu=2; //OJO AQUI PODRIAMOS SALIR YA DEL MENU poniendo menu=0 y
bandera_nivel_menu=0
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){
```

```
            bandera_nivel_menu=1;
            boton_MENU_primera_vez=true;
            menu=1; //siguiente Menu
            //Serial.println("Menu Cambiar Minutos");
            lcd.setCursor(0, 0);
            lcd.print(F("Cambiar Min  "));
          }
/*#####################################################*/
/*#####################################################*/
/*#####################################################*/

     }else if(bandera_nivel_menu==2 && menu==1){
          if (valor_boton_MENU==HIGH){

            bandera_nivel_menu=1;

            //Serial.println("Menu Cambiar Minutos");
/*
            lcd.setCursor(0, 0);
            lcd.print(F("Cambiar Min  "));
 */
                                    // no toco la variable menu, porque mantiene el valor del menu en el que
estaba
          }else if (valor_boton_CHG==HIGH){
            minutos++;
            if(minutos>59)
            {
              minutos=0;
            }
            sprintf(VARI,"%02d:%02d:%02d     ",horas,minutos,segundos);
            //Serial.println(VARI);
            lcd.setCursor(0, 0);
            lcd.print(VARI);
          }

     }else if(bandera_nivel_menu==2 && menu==2){
          if (valor_boton_MENU==HIGH){
            bandera_nivel_menu=1;
            //Serial.println("Menu Cambiar Horas");
/*
            lcd.setCursor(0, 0);
            lcd.print(F("Cambiar Hora "));
*/
            // no toco la variable menu, porque mantiene el valor del menu en el que estaba
          }else if (valor_boton_CHG==HIGH){
            horas++;
            if(horas>23)
            {
              horas=0;
            }
            sprintf(VARI,"%02d:%02d:%02d     ",horas,minutos,segundos);
            //Serial.println(VARI);
            lcd.setCursor(0, 0);
            lcd.print(VARI);
          }
     }else if(bandera_nivel_menu==2 && menu==3){
          if (valor_boton_MENU==HIGH){
            bandera_nivel_menu=1;
            //Serial.println("Menu Resetear Contador");
/*
                        if(valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON==99){
                                lcd.setCursor(0, 0);
                                lcd.print(F(DESHABILITADO_HUMIDIFICADOR));
                        }else{
                                        lcd.setCursor(0, 0);
                                        lcd.print(F("Humid. "));
```

```
                                sprintf(VARI,"%02d", valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON);
                                lcd.print(VARI);
                                        lcd.print(F("    "));
                        }
*/
          // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){


indice_valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON=(indice_valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON
+1)%5;


valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON=valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON[indice_
valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON];
                                        if(valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON==99){
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(DESHABILITADO_HUMIDIFICADOR));
                                                flag_habilitado_humidificador=false;
                                                stop_humidificador();
                                        }else{
          lcd.setCursor(0, 0);
          lcd.print(F("Humid. "));
                                sprintf(VARI,"%02d", valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON);
                                lcd.print(VARI);
          lcd.print(F("    "));
                                                flag_habilitado_humidificador=true;
                                        }
        }

    }else if(bandera_nivel_menu==2 && menu==4){
        if (valor_boton_MENU==HIGH){
          bandera_nivel_menu=1;
          //Serial.println("Menu START Modo AUTO");
/*
          lcd.setCursor(0, 0);
          lcd.print(F("START M AUTO "));
*/
          // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
                  stop_total();
             start_modo_auto();
        }

    }else if(bandera_nivel_menu==2 && menu==5){
        if (valor_boton_MENU==HIGH){
          bandera_nivel_menu=1;
          //Serial.println("Menu START Modo GROW");
/*
          lcd.setCursor(0, 0);
          lcd.print(F("START M GROW "));
*/
          // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
                  stop_total();
                  lcd.setCursor(14, 1);
                  lcd.print(F(" G"));
             start_modo_grow();
        }

    }else if(bandera_nivel_menu==2 && menu==6){
        if (valor_boton_MENU==HIGH){
          bandera_nivel_menu=1;
          //Serial.println("Menu START Modo Fructif");
/*
          lcd.setCursor(0, 0);
          lcd.print(F("START M FRUCT"));
*/
```

```
            // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
                    stop_total();
                    lcd.setCursor(14, 1);
                    lcd.print(F(" F"));
            start_modo_fructif();
        }
    }else if(bandera_nivel_menu==2 && menu==7){
        if (valor_boton_MENU==HIGH){
            bandera_nivel_menu=1;
            //Serial.println("STOP");
/*
            lcd.setCursor(0, 0);
            lcd.print(F("STOP        "));
*/
            // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
            stop_total();
        }
    }else if(bandera_nivel_menu==2 && menu==8){
        if (valor_boton_MENU==HIGH){
            bandera_nivel_menu=1;
            //Serial.println("HABILITADO/DESHABILITADO FUZZY");
/*
                if (flag_habilitado_fuzzy==true){
                                lcd.setCursor(0, 0);
                                lcd.print(F(HABILITADO_FUZZY));
                }else{
                                lcd.setCursor(0, 0);
                                lcd.print(F(DESHABILITADO_FUZZY));
                }
*/
            // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
            //Serial.println("HABILITADO/DESHABILITADO FUZZY");
                                if (flag_habilitado_fuzzy==true){
                                        flag_habilitado_fuzzy=false;
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(DESHABILITADO_FUZZY));
                                                stop_fuzzy_aire_bomba();
                                }else{
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(HABILITADO_FUZZY));
                                        flag_habilitado_fuzzy=true;
                                }
        }

    }else if(bandera_nivel_menu==2 && menu==9){
        if (valor_boton_MENU==HIGH){
            bandera_nivel_menu=1;

            //Serial.println("INCREMENTAR CONTADOR");
/*
                lcd.setCursor(0, 0);
                lcd.print(F("INCREMEN CONT"));
*/
            // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
                                if(contador_iniciado_horas_modo!=0){

        contador_iniciado_horas_modo=contador_iniciado_horas_modo+10;
                                                if(contador_iniciado_horas_modo>=990){
                                    contador_iniciado_horas_modo=999;
                                                }
                                                pintar_contador();
                        }//if algun modo activo
```

```
        }
    }else if(bandera_nivel_menu==2 && menu==10){
        if (valor_boton_MENU==HIGH){
           bandera_nivel_menu=1;

           //Serial.println("DECREMENTAR CONTADOR");
/*
                lcd.setCursor(0, 0);
                lcd.print(F("DECREMEN CONT"));
*/
           // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
                                if(contador_iniciado_horas_modo!=0){

        contador_iniciado_horas_modo=contador_iniciado_horas_modo-10;
                                                if(contador_iniciado_horas_modo<10){
                                    contador_iniciado_horas_modo=1;
                                                }
                                                pintar_contador();
                        }//if algun modo activo

        }

/************************** SALGO DE LOS MENUS *********************************/
/*******************************************************************************/
/*******************************************************************************/
  }//if cuelgue LCD
     if((contador_LED<10)&&(menu==0)&&(bandera_nivel_menu==0)){  //encendere el LED solo si no esta
encendido por alguno de los modos
           contador_LED++;
    }else if((menu==0)&&(bandera_nivel_menu==0)){
           if (el_LED_esta_encendido==true){ //si el LED esta encendido por el fotoperiodo no debo apagarlo
aunque salga del menu.
                                }else{
                                        digitalWrite(ledPin,LOW);
                                }
           //Serial.println(contador_LED);
           lcd.noBacklight();
           contador_LED=0;
    }

   bandera1=false;
   boton_MENU_primera_vez=false;
}//if  bandera1

////////////////////////////////////////////////////////////////////////////////////////////////////
// FIN DEL MENU DEL LCD
////////////////////////////////////////////////////////////////////////////////////////////////////


//CADA DIEZ SEGUNDOS
if(bandera2==true){




////////////////////////////////////////////////////////////////////////////////////////////////////
//RESET DEL LCD
////////////////////////////////////////////////////////////////////////////////////////////////////

//si mantengo 10 segundos los 2 botones entonces reseteo el display
  if ((valor_boton_CHG==HIGH) && (valor_boton_MENU==HIGH)){
     lcd.begin(16,2);   // initialize the lcd for 16 chars 2 lines, turn on backlight
     lcd.backlight();
     lcd.setCursor(0, 0);
```

```
      //lcd.print(F("RST LCD"));
      menu=0;
      bandera_nivel_menu=0;
      contador_LED=0; //enciendo el LCD 10 segundos para que se vea que lo he reseteado
            //pinto el contador de horas modo si es distinto de 0
            if(contador_iniciado_horas_modo!=0){
                  pintar_contador(); //pinto contador modo
                  //Ahora tengo que refrescar la letra que indica el modo de funcionamiento si la hay
              lcd.setCursor(14, 1);
                  if (flag_empezado_modo_auto_2a_fase==true){
                     lcd.print(F("AF"));
                  }else if(flag_empezado_modo_auto==true){
                     lcd.print(F("AG"));
                  }else if(flag_empezado_modo_fructif==true){
                     lcd.print(F(" F"));
                  }else if(flag_empezado_modo_grow==true){
                     lcd.print(F(" G"));
                  }

            }//fin de if para refrescar el display

  }//fin de cuelgue
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FIN DEL RESET DEL LCD
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

     lcd.setCursor(0, 1);
     switch (parado_o_frio_o_calor){
       case 0:
            lcd.print(F("T")); //simbolo para temperatura con la peltier apagada.
            break;
       case 1:
            lcd.print(F("*")); //simbolo para saber que la peltier esta enfriando.
            break;
       case 2:
            lcd.print(F("!")); //simbolo para saber que la peltier esta calefactando.
            break;
     }//switch

     //Pintar_Lecturas_Sensores();
           temperatura_10s = (float)dht.readTemperature();
     lcd.print(temperatura_10s);
     lcd.print(F(" H"));
           humedad_10s = (float)dht.readHumidity();
     lcd.print(humedad_10s);
     lcd.print(F("%"));
     //Enviar_Lecturas_Sensores() por el puerto serie al PC para su captura y/o presentacion
   Serial.print(temperatura_10s,2);
   Serial.print(" ");
   Serial.println(humedad_10s,2);

     bandera2=false;
}//if bandera2

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// INICIO CALCULO DE MODOS DE FUNCIONAMIENTO
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//CADA DOS SEGUNDOS
if(bandera3==true){
     /*********************************/
     /***** LEO MODOS ACTIVOS *****/
            modo_grow();
            modo_fructif();
            modo_auto();
     /*********************************/
     /*********************************/
```

```
    bandera3=false;
}//if bandera3
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// FIN de CALCULO DE MODOS DE FUNCIONAMIENTO
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

}// fin del loop
```

4.1 Código dedicado a la ejecución del menú LCD

```
void loop()
{

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// MENU DEL LCD
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//CADA SEGUNDO

if(bandera1==true){


     valor_boton_CHG=digitalRead(boton_CHG);
     valor_boton_MENU=digitalRead(boton_MENU);

  if ((valor_boton_CHG==HIGH) && (valor_boton_MENU==HIGH)){
         //no hago nada porque he bloqueado el programa y lo que quiero es que si estoy presionando 10
segundos los dos botones entonces se resetee el display
         //se supone que si hago esto es porque el display LCD hay que reinicializarlo
  }else{

     if(bandera_nivel_menu==0){
          sprintf(VARI,"%02d:%02d:%02d    ",horas,minutos,segundos);
          //Serial.println(VARI); //pintar la HORA
          lcd.setCursor(0, 0);
          lcd.print(VARI);
          //pintar_espacio_vacio_contador();


          if (valor_boton_MENU==HIGH){

              menu=1;
              bandera_nivel_menu=1;
              boton_MENU_primera_vez=true;
              lcd.backlight();

                //Serial.println("Menu Cambiar Min");
                lcd.setCursor(0, 0);
                lcd.print(F("Cambiar Min  "));


          }else if (valor_boton_CHG==HIGH){
              digitalWrite(ledPin,HIGH);
              lcd.backlight();
              contador_LED=0;
          }

     }else if(bandera_nivel_menu==1 && menu==1){

          if (valor_boton_CHG==HIGH){
            //Serial.println("Menu Cambiar Minutos");
/*
            lcd.setCursor(0, 0);
            lcd.print(F("Cambiar Min  "));
*/
            bandera_nivel_menu=2;
          }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

              bandera_nivel_menu=1;
              boton_MENU_primera_vez=true;
```

```
            menu=(menu+1)%12; //siguiente Menu
            //Serial.println("Menu Cambiar Horas");
            lcd.setCursor(0, 0);
            lcd.print(F("Cambiar Hora "));
        }

    }else if(bandera_nivel_menu==1 && menu==2){

        if (valor_boton_CHG==HIGH){
          //Serial.println("Menu Cambiar Horas");
/*
          lcd.setCursor(0, 0);
          lcd.print(F("Cambiar Hora "));
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

            bandera_nivel_menu=1;
            boton_MENU_primera_vez=true;
            menu=(menu+1)%12; //siguiente Menu
            //Serial.println("Menu Resetear Contador");
                                        if(valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON==99){
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(DESHABILITADO_HUMIDIFICADOR));
                                        }else{
            lcd.setCursor(0, 0);
            lcd.print(F("Humid. "));
                                        sprintf(VARI,"%02d", valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON);
                                        lcd.print(VARI);
            lcd.print(F("    "));
                                        }
                                }

    }else if(bandera_nivel_menu==1 && menu==3){

        if (valor_boton_CHG==HIGH){
          //Serial.println("Menu Resetear Contador");
/*
                if(valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON==99){
                                        lcd.setCursor(0, 0);
                                        lcd.print(F(DESHABILITADO_HUMIDIFICADOR));
                }else{
                                        lcd.setCursor(0, 0);
                                        lcd.print(F("Humid. "));
                                        sprintf(VARI,"%02d",
valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON);
                                        lcd.print(VARI);
                                        lcd.print(F("    "));
                }
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

            bandera_nivel_menu=1;
            boton_MENU_primera_vez=true;
            menu=(menu+1)%12; //siguiente Menu
            //Serial.println("Menu START Modo AUTO");
            lcd.setCursor(0, 0);
            lcd.print(F("START M AUTO "));
        }


    }else if(bandera_nivel_menu==1 && menu==4){

        if (valor_boton_CHG==HIGH){
          //Serial.println("Menu START Modo AUTO");
/*
```

```
          lcd.setCursor(0, 0);
          lcd.print(F("START M AUTO "));
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=1;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("Menu START Modo GROW");
           lcd.setCursor(0, 0);
           lcd.print(("START M GROW "));
        }

    }else if(bandera_nivel_menu==1 && menu==5){

        if (valor_boton_CHG==HIGH){
          //Serial.println("Menu START Modo GROW");
/*
          lcd.setCursor(0, 0);
          lcd.print(F("START M GROW "));
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=1;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("Menu START Modo Fructif");
           lcd.setCursor(0, 0);
           lcd.print(F("START M FRUCT"));
        }


    }else if(bandera_nivel_menu==1 && menu==6){

        if (valor_boton_CHG==HIGH){
          //Serial.println("Menu START Modo Fructif");
/*
          lcd.setCursor(0, 0);
          lcd.print(F("START M FRUCT"));
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=1;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("STOP");
           lcd.setCursor(0, 0);
           lcd.print(F("STOP        "));
        }

    }else if(bandera_nivel_menu==1 && menu==7){

        if (valor_boton_CHG==HIGH){
          //Serial.println("STOP");
/*
          lcd.setCursor(0, 0);
          lcd.print(F("STOP        "));
*/
          bandera_nivel_menu=2;
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

           bandera_nivel_menu=2;
           boton_MENU_primera_vez=true;
           menu=(menu+1)%12; //siguiente Menu
           //Serial.println("HABILITADO/DESHABILITADO FUZZY");
                                if (flag_habilitado_fuzzy==true){
```

```
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(HABILITADO_FUZZY));
                                    }else{
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(DESHABILITADO_FUZZY));
                                    }
            }
        }else if(bandera_nivel_menu==1 && menu==8){ //menu HABILITAR / DESHABILITAR FUZZY


            //Serial.println("HABILITADO/DESHABILITADO FUZZY");
            if (valor_boton_CHG==HIGH){
/*
                                    if (flag_habilitado_fuzzy==true){
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(HABILITADO_FUZZY));
                                    }else{
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(DESHABILITADO_FUZZY));
                                    }
*/
                                    bandera_nivel_menu=2;

            }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

               bandera_nivel_menu=2;
               boton_MENU_primera_vez=true;
               menu=(menu+1)%12; //siguiente Menu
               //Serial.println("INCREMENTAR CONTADOR");
               lcd.setCursor(0, 0);
                     lcd.print(F("INCREMEN CONT"));
            }
        }else if(bandera_nivel_menu==1 && menu==9){ //incrementar


            if (valor_boton_CHG==HIGH){
               //Serial.println("INCREMENTAR CONTADOR");
/*
                     lcd.setCursor(0, 0);
                     lcd.print(F("INCREMEN CONT"));
*/
              bandera_nivel_menu=2;

            }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

               bandera_nivel_menu=2;
               boton_MENU_primera_vez=true;
               menu=(menu+1)%12; //siguiente Menu
               //Serial.println("DECREMENTAR_CONTADOR");
               lcd.setCursor(0, 0);
                     lcd.print(F("DECREMEN CONT"));
            }
        }else if(bandera_nivel_menu==1 && menu==10){ //incrementar


            if (valor_boton_CHG==HIGH){
               //Serial.println("DECREMENTAR_CONTADOR");
/*
                     lcd.setCursor(0, 0);
                     lcd.print(F("DECREMEN CONT"));
*/
              bandera_nivel_menu=2;

            }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

               bandera_nivel_menu=2;
               boton_MENU_primera_vez=true;
```

```
            menu=(menu+1)%12; //siguiente Menu
            //Serial.println("SALIR");
            lcd.setCursor(0, 0);
            lcd.print(F("SALIR       "));
        }

    }else if(bandera_nivel_menu==1 && menu==11){ //menu SALIR


        if (valor_boton_CHG==HIGH){
           //Serial.println("Fuera de Menu");
           bandera_nivel_menu=2; //OJO AQUI PODRIAMOS SALIR YA DEL MENU poniendo menu=0 y
bandera_nivel_menu=0
        }else if(valor_boton_MENU==HIGH && boton_MENU_primera_vez==false){

            bandera_nivel_menu=1;
            boton_MENU_primera_vez=true;
            menu=1; //siguiente Menu
            //Serial.println("Menu Cambiar Minutos");
            lcd.setCursor(0, 0);
            lcd.print(F("Cambiar Min  "));
        }
/*############################################################*/
/*############################################################*/
/*############################################################*/

    }else if(bandera_nivel_menu==2 && menu==1){
        if (valor_boton_MENU==HIGH){

           bandera_nivel_menu=1;

            //Serial.println("Menu Cambiar Minutos");
/*
            lcd.setCursor(0, 0);
            lcd.print(F("Cambiar Min  "));
 */
                                    // no toco la variable menu, porque mantiene el valor del menu en el que
estaba
        }else if (valor_boton_CHG==HIGH){
            minutos++;
            if(minutos>59)
            {
              minutos=0;
            }
            sprintf(VARI,"%02d:%02d:%02d     ",horas,minutos,segundos);
            //Serial.println(VARI);
            lcd.setCursor(0, 0);
            lcd.print(VARI);
        }

    }else if(bandera_nivel_menu==2 && menu==2){
        if (valor_boton_MENU==HIGH){
            bandera_nivel_menu=1;
            //Serial.println("Menu Cambiar Horas");
/*
            lcd.setCursor(0, 0);
            lcd.print(F("Cambiar Hora "));
*/
            // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
            horas++;
            if(horas>23)
            {
              horas=0;
            }
            sprintf(VARI,"%02d:%02d:%02d     ",horas,minutos,segundos);
            //Serial.println(VARI);
            lcd.setCursor(0, 0);
            lcd.print(VARI);
```

```
        }
    }else if(bandera_nivel_menu==2 && menu==3){
        if (valor_boton_MENU==HIGH){
           bandera_nivel_menu=1;
           //Serial.println("Menu Resetear Contador");
/*
                        if(valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON==99){
                                lcd.setCursor(0, 0);
                                lcd.print(F(DESHABILITADO_HUMIDIFICADOR));
                        }else{
                                        lcd.setCursor(0, 0);
                                        lcd.print(F("Humid. "));
                                sprintf(VARI,"%02d", valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON);
                                lcd.print(VARI);
                                        lcd.print(F("    "));
                        }
*/
           // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){



indice_valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON=(indice_valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON
+1)%5;


valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON=valor_RETARDO_TIEMPO_HUMIDIFICADOR_ON[indice_v
alor_RETARDO_TIEMPO_HUMIDIFICADOR_ON];
                                        if(valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON==99){
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(DESHABILITADO_HUMIDIFICADOR));
                                                flag_habilitado_humidificador=false;
                                                stop_humidificador();
                                        }else{
           lcd.setCursor(0, 0);
           lcd.print(F("Humid. "));
                                sprintf(VARI,"%02d", valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON);
                                lcd.print(VARI);
           lcd.print(F("    "));
                                                flag_habilitado_humidificador=true;
                                        }
        }

    }else if(bandera_nivel_menu==2 && menu==4){
        if (valor_boton_MENU==HIGH){
           bandera_nivel_menu=1;
           //Serial.println("Menu START Modo AUTO");
/*
           lcd.setCursor(0, 0);
           lcd.print(F("START M AUTO "));
*/
           // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
                    stop_total();
              start_modo_auto();
        }

    }else if(bandera_nivel_menu==2 && menu==5){
        if (valor_boton_MENU==HIGH){
           bandera_nivel_menu=1;
           //Serial.println("Menu START Modo GROW");
/*
           lcd.setCursor(0, 0);
           lcd.print(F("START M GROW "));
*/
           // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
```

```
                stop_total();
                lcd.setCursor(14, 1);
                lcd.print(F(" G"));
            start_modo_grow();
        }

    }else if(bandera_nivel_menu==2 && menu==6){
        if (valor_boton_MENU==HIGH){
            bandera_nivel_menu=1;
            //Serial.println("Menu START Modo Fructif");
/*
            lcd.setCursor(0, 0);
            lcd.print(F("START M FRUCT"));
*/
            // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
                stop_total();
                lcd.setCursor(14, 1);
                lcd.print(F(" F"));
            start_modo_fructif();
        }
    }else if(bandera_nivel_menu==2 && menu==7){
        if (valor_boton_MENU==HIGH){
            bandera_nivel_menu=1;
            //Serial.println("STOP");
/*
            lcd.setCursor(0, 0);
            lcd.print(F("STOP        "));
*/
            // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
            stop_total();
        }
    }else if(bandera_nivel_menu==2 && menu==8){
        if (valor_boton_MENU==HIGH){
            bandera_nivel_menu=1;
            //Serial.println("HABILITADO/DESHABILITADO FUZZY");
/*
                if (flag_habilitado_fuzzy==true){
                                lcd.setCursor(0, 0);
                                lcd.print(F(HABILITADO_FUZZY));
                }else{
                                lcd.setCursor(0, 0);
                                lcd.print(F(DESHABILITADO_FUZZY));
                }
*/
            // no toco la variable menu, porque mantiene el valor del menu en el que estaba
        }else if (valor_boton_CHG==HIGH){
            //Serial.println("HABILITADO/DESHABILITADO FUZZY");
                                if (flag_habilitado_fuzzy==true){
                                        flag_habilitado_fuzzy=false;
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(DESHABILITADO_FUZZY));
                                                stop_fuzzy_aire_bomba();
                                }else{
                                                lcd.setCursor(0, 0);
                                                lcd.print(F(HABILITADO_FUZZY));
                                        flag_habilitado_fuzzy=true;
                                }
        }

    }else if(bandera_nivel_menu==2 && menu==9){
        if (valor_boton_MENU==HIGH){
            bandera_nivel_menu=1;

            //Serial.println("INCREMENTAR CONTADOR");
/*
                lcd.setCursor(0, 0);
                lcd.print(F("INCREMEN CONT"));
```

```
*/
            // no toco la variable menu, porque mantiene el valor del menu en el que estaba
          }else if (valor_boton_CHG==HIGH){
                                    if(contador_iniciado_horas_modo!=0){

          contador_iniciado_horas_modo=contador_iniciado_horas_modo+10;
                                                    if(contador_iniciado_horas_modo>=990){
                                       contador_iniciado_horas_modo=999;
                                                    }
                                                    pintar_contador();
                          }//if algun modo activo

          }
      }else if(bandera_nivel_menu==2 && menu==10){
          if (valor_boton_MENU==HIGH){
             bandera_nivel_menu=1;

             //Serial.println("DECREMENTAR CONTADOR");
/*
                    lcd.setCursor(0, 0);
                    lcd.print(F("DECREMEN CONT"));
*/
             // no toco la variable menu, porque mantiene el valor del menu en el que estaba
          }else if (valor_boton_CHG==HIGH){
                                    if(contador_iniciado_horas_modo!=0){

          contador_iniciado_horas_modo=contador_iniciado_horas_modo-10;
                                                    if(contador_iniciado_horas_modo<10){
                                       contador_iniciado_horas_modo=1;
                                                    }
                                                    pintar_contador();
                          }//if algun modo activo

          }

/************************** SALGO DE LOS MENUS **********************************/
/********************************************************************************/
/********************************************************************************/
/********************************************************************************/
/********************************************************************************/
  }//if cuelgue LCD
      if((contador_LED<10)&&(menu==0)&&(bandera_nivel_menu==0)){  //encendere el LED solo si no esta
encendido por alguno de los modos
             contador_LED++;
      }else if((menu==0)&&(bandera_nivel_menu==0)){
             if (el_LED_esta_encendido==true){ //si el LED esta encendido por el fotoperiodo no debo apagarlo
aunque salga del menu.
                                    }else{
                                              digitalWrite(ledPin,LOW);
                                    }
             //Serial.println(contador_LED);
             lcd.noBacklight();
             contador_LED=0;
      }

   bandera1=false;
   boton_MENU_primera_vez=false;
}//if  bandera1

////////////////////////////////////////////////////////////////////////////////////////////////////
// FIN DEL MENU DEL LCD
////////////////////////////////////////////////////////////////////////////////////////////////////


//CADA DIEZ SEGUNDOS
if(bandera2==true){
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//RESET DEL LCD
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//si mantengo 10 segundos los 2 botones entonces reseteo el display
  if ((valor_boton_CHG==HIGH) && (valor_boton_MENU==HIGH)){
     lcd.begin(16,2);   // initialize the lcd for 16 chars 2 lines, turn on backlight
     lcd.backlight();
     lcd.setCursor(0, 0);
     //lcd.print(F("RST LCD"));
     menu=0;
     bandera_nivel_menu=0;
     contador_LED=0; //enciendo el LCD 10 segundos para que se vea que lo he reseteado
           //pinto el contador de horas modo si es distinto de 0
           if(contador_iniciado_horas_modo!=0){
                 pintar_contador(); //pinto contador modo
                 //Ahora tengo que refrescar la letra que indica el modo de funcionamiento si la hay
             lcd.setCursor(14, 1);
                 if (flag_empezado_modo_auto_2a_fase==true){
                    lcd.print(F("AF"));
                 }else if(flag_empezado_modo_auto==true){
                    lcd.print(F("AG"));
                 }else if(flag_empezado_modo_fructif==true){
                    lcd.print(F(" F"));
                 }else if(flag_empezado_modo_grow==true){
                    lcd.print(F(" G"));
                 }

           }//fin de if para refrescar el display

  }//fin de cuelgue
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FIN DEL RESET DEL LCD
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

4.2 Código dedicado a la ejecución de los modos de funcionamiento

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_MODO_GROW
// funcion que la usaremos para configurar los flags adecuadamente para que comience el MODO GROW en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void start_modo_grow(){
  int *mi_variable = &contador_iniciado_horas_modo;

    flag_empezado_modo_grow=true;
    flag_empezado_modo_fructif=false;
        parado_o_frio_o_calor=1; //frio al inicio
    activado_reles_flag_frio = true; //comienzo enfriando que será lo más normal
        activado_reles_flag_calor = false; //comienzo enfriando que será lo más normal

        comienza_retardo(RETARDO_HORAS_MODO_GROW, mi_variable);

}//start modo grow

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_MODO_GROW
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el MODO GROW en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_modo_grow(){
    flag_empezado_modo_grow=false;
    lcd.setCursor(14, 1);
    lcd.print(F(" S"));

        activado_reles_flag_frio = false; //dejo de enfriar
        activado_reles_flag_calor = false; //dejo de calentar
        parado_o_frio_o_calor=0;
        contador_iniciado_horas_modo=0;
        contador_iniciado_humidificador=0;
        contador_iniciado_humidificador_en_OFF=0;
        activado_flag_humidificador=true;
        flag_hay_que_humidificar = false;

       digitalWrite(humidificador, LOW);
       digitalWrite(fuzzy_aire_bomba, LOW);

        contador_iniciado_fuzzy_aire_bomba=0;
        contador_iniciado_fuzzy_aire_bomba_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF = RETARDO_TIEMPO_EN_OFF_BOMBA;
        activado_flag_bomba_aireadora=true;

        contador_iniciado_fotoperiodo=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF = RETARDO_TIEMPO_EN_OFF_BOMBA;
        contador_iniciado_fotoperiodo_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF = RETARDO_TIEMPO_EN_OFF_BOMBA;
        activado_flag_fotoperiodo=true;
        el_LED_esta_encendido=false;

      pintar_contador();
  desconecto_reles();

}//stop modo grow

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_MODO_FRUCTIF
// funcion que la usaremos para configurar los flags adecuadamente para que comience el MODO FRUCTIF en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void start_modo_fructif(){
  int *mi_variable = &contador_iniciado_horas_modo;
```

```
    flag_empezado_modo_fructif=true;
    flag_empezado_modo_grow=false;
       parado_o_frio_o_calor=1; //frio al inicio
          activado_reles_flag_frio = true; //comienzo enfriando que será lo más normal
          activado_reles_flag_calor = false; //comienzo enfriando que será lo más normal

         comienza_retardo(RETARDO_HORAS_MODO_FRUCTIF, mi_variable);

}//start modo fructif

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_MODO_FRUCTIF
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el MODO FRUCTIF en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_modo_fructif(){
    flag_empezado_modo_fructif=false;
    lcd.setCursor(14, 1);
    lcd.print(F(" S"));
          activado_reles_flag_frio = false; //dejo de enfriar
          activado_reles_flag_calor = false; //dejo de calentar
          parado_o_frio_o_calor=0;
          contador_iniciado_horas_modo=0;
   digitalWrite(ledPin, LOW); //apago el LED interior
   digitalWrite(humidificador, LOW);
         digitalWrite(fuzzy_aire_bomba, LOW);

          contador_iniciado_humidificador=0;
          contador_iniciado_humidificador_en_OFF=0;
          activado_flag_humidificador=true;
          flag_hay_que_humidificar = false;

          contador_iniciado_fuzzy_aire_bomba=0;
          contador_iniciado_fuzzy_aire_bomba_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF = RETARDO_TIEMPO_EN_OFF_BOMBA;
          activado_flag_bomba_aireadora=true;

          contador_iniciado_fotoperiodo=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF = RETARDO_TIEMPO_EN_OFF_BOMBA;
          contador_iniciado_fotoperiodo_en_OFF=0; //int contador_iniciado_fuzzy_aire_bomba_en_OFF = RETARDO_TIEMPO_EN_OFF_BOMBA;
          activado_flag_fotoperiodo=true;
          el_LED_esta_encendido=false;

         pintar_contador();
   desconecto_reles();

}//stop modo fructif

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_MODO_AUTO
// funcion que la usaremos para configurar los flags adecuadamente para que funcione el MODO AUTO en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void start_modo_auto(){
   flag_empezado_modo_auto=true;
   lcd.setCursor(14, 1);
   lcd.print(F("AG"));
   start_modo_grow(); //comienzo primero con el modo GROW
   //start_modo_fructif(); //comienzo despues con el modo GROW
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_MODO_AUTO
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el MODO AUTO en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_modo_auto(){
   stop_modo_grow();
```

```
    stop_modo_fructif();
     flag_empezado_modo_auto=false;
           flag_empezado_modo_auto_2a_fase=false;
     lcd.setCursor(14, 1);
     lcd.print(F(" S"));
}//stop modo fructif

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_TOTAL
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el MODO que estuviera activo en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_total(){

  int *mi_variable = &contador_iniciado_reles;
  comienza_retardo(RETARDO_CONEXION_RELES, mi_variable);  //comienzo el retardo de los reles para que si empiezo otro modo no conmute inmediatamente la peltier.

    stop_modo_grow();
    stop_modo_fructif();
    stop_modo_auto();
}//stop modo fructif


////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION MODO_GROW
// funcion que que se corresponde al hilo de ejecucion MODO_GROW en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
void modo_grow(){
    int *mi_variable = &contador_iniciado_reles;
//para que el micelio crezca ha de estar a una temperatura optima de 25ºC. la temperatura de la sala ha de estar entre 18 y 24ºC

if( flag_empezado_modo_grow==true){
pintar_contador();

  //esta funcion se llama cada 2 segundos  con la bandera 3
     //comienzo calculando el valor para activar el frio, calor o parado
     //comienzo calculando el valor para activar el humidificador o no (que habiamos acordado de hacerlo a rafagas de 15 segundos en on y 15 segundos en off)
     //NO HAY LUZ EN ESTE MODO

                  if ((temperatura_10s <= 24) && (activado_reles_flag_frio == true) && (activado_reles_flag_calor == false)){
                        parado_o_frio_o_calor=2; //estaba enfriando y debo pasar a calentar porque si no me paso de frio
                              comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo retardo porque habra un cambio de estado
                              desconecto_reles();
                  }else if((temperatura_10s >= 27) && (activado_reles_flag_frio == false) && (activado_reles_flag_calor == true)){
                        parado_o_frio_o_calor=1; //estaba calentando y debo pasar a enfriar porque si no me paso de calor
                              comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo retardo porque habra un cambio de estado
                              desconecto_reles();
                  }else if((activado_reles_flag_frio == true) && (activado_reles_flag_calor == true)){
                        activado_reles_flag_frio = true; //esta condicion no debe darse y por eso volvemos a resetearlo como si comenzaramos enfriando
                        activado_reles_flag_calor = false;
                        parado_o_frio_o_calor=1;
                              comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo retardo porque habra un cambio de estado
                              desconecto_reles();
                  }else if( (temperatura_10s > 24) && (temperatura_10s < 27) ){
                        //no cambio nada porque estoy funcionando en la zona optima
                  }
```

```
/*****************************************************************************/
/////////////////////////////////////////////////////////////////////////////
/// DE SI HAY QUE PONER FRIO O CALOR ////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////

    if (parado_o_frio_o_calor == 1){
       activo_flag_peltier_frio();
    }else if (parado_o_frio_o_calor == 2){
       activo_flag_peltier_calor();
    }else if (parado_o_frio_o_calor == 0){
       desactivo_flag_peltier();
       desconecto_reles();
    }

/////////////////////////////////////////////////////////////////////////////
/*****************************************************************************/
/*****************************************************************************/
/////////////////////////////////////////////////////////////////////////////
/// DE SI HAY QUE HUMIDIFICAR O NO //////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////

//Activamos el humidificador siempre que la humedad esté por debajo de 80% y dejamos de humidificar si
estamos por encima de 90%
                if (humedad_10s <= 70){
                        flag_hay_que_humidificar = true;
                }else if (humedad_10s >= 80){
                        flag_hay_que_humidificar = false;
                }

/////////////////////////////////////////////////////////////////////////////
/*****************************************************************************/

  peltier(); //activara o desactivara el frio o el calor
  humidifica(); //activa o desactiva el humidificador

/////////////////////////////////////////////////////////////////////////////
/*****************************************************************************/

/////////////////////////////////////////////////////////////////////////////
/*****************************************************************************/
/*****************************************************************************/
/*****************************************************************************/
        if(temperatura_actual <= 23){
                intervalo = 1;
        }else if((temperatura_actual > 23) &&(temperatura_actual < 27)){
                intervalo = 2;
        }else if(temperatura_actual >= 27){
                intervalo = 3;
        }
/*****************************************************************************/

   bomba_aireadora(); //activa o desactiva la bomba aireadora
/*****************************************************************************/
/*****************************************************************************/
/////////////////////////////////////////////////////////////////////////////
/*****************************************************************************/
}

if((flag_empezado_modo_grow==true)&&(contador_iniciado_horas_modo==0)){
        stop_modo_grow();
}//if modo grow

}//modo grow
/*****************************************************************************/
/*****************************************************************************/
/*****************************************************************************/
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION MODO_FRUCTIF
// funcion que que se corresponde al hilo de ejecucion MODO_FRUCTIF en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////////////
void modo_fructif(){
    int *mi_variable = &contador_iniciado_reles;
//para que el micelio fructifique ha de estar a una temperatura optima de 12ºC. la temperatura de la sala ha de estar entre 12 y 18ºC

if( flag_empezado_modo_fructif==true){
pintar_contador();

  //esta funcion se llama cada 2 segundos  con la bandera 3
      //comienzo calculando el valor para activar el frio, calor o parado
      //comienzo calculando el valor para activar el humidificador o no (que habiamos acordado de hacerlo a rafagas de 15 segundos en on y 15 segundos en off)
      //SI HAY LUZ EN ESTE MODO

                    if ((temperatura_10s <= 11) && (activado_reles_flag_frio == true) && (activado_reles_flag_calor == false)){
                                parado_o_frio_o_calor=2; //estaba enfriando y debo pasar a calentar porque si no me paso de frio
                                        comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo retardo porque habra un cambio de estado
                                        desconecto_reles();
                    }else if((temperatura_10s >= 14) && (activado_reles_flag_frio == false) && (activado_reles_flag_calor == true)){
                                parado_o_frio_o_calor=1; //estaba calentando y debo pasar a enfriar porque si no me paso de calor
                                        comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo retardo porque habra un cambio de estado
                                        desconecto_reles();
                    }else if((activado_reles_flag_frio == true) && (activado_reles_flag_calor == true)){
                                activado_reles_flag_frio = true; //esta condicion no debe darse y por eso volvemos a resetearlo como si comenzaramos enfriando
                                activado_reles_flag_calor = false;
                                parado_o_frio_o_calor=1;
                                        comienza_retardo(RETARDO_CONEXION_RELES, mi_variable); //activo retardo porque habra un cambio de estado
                                        desconecto_reles();
                    }else if( (temperatura_10s > 11) && (temperatura_10s < 14) ){
                                //no cambio nada porque estoy funcionando en la zona optima
                    }


/*************************************************************************************/
////////////////////////////////////////////////////////////////////////////
/// DE SI HAY QUE PONER FRIO O CALOR ///////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

      if (parado_o_frio_o_calor == 1){
          activo_flag_peltier_frio();
      }else if (parado_o_frio_o_calor == 2){
          activo_flag_peltier_calor();
      }else if (parado_o_frio_o_calor == 0){
          desactivo_flag_peltier();
          desconecto_reles();
      }

////////////////////////////////////////////////////////////////////////////
/*************************************************************************************/
/*************************************************************************************/
////////////////////////////////////////////////////////////////////////////
/// DE SI HAY QUE HUMIDIFICAR O NO /////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

//Activamos el humidificador siempre que la humedad esté por debajo de 80% y dejamos de humidificar si
```

```
estamos por encima de 90%
                if (humedad_10s <= 70){
                        flag_hay_que_humidificar = true;
                }else if (humedad_10s >= 80){
                        flag_hay_que_humidificar = false;
                }

////////////////////////////////////////////////////////////////////////////
/*************************************************************************/

  peltier(); //activara o desactivara el frio o el calor
  humidifica(); //activa o desactiva el humidificador

////////////////////////////////////////////////////////////////////////////
/*************************************************************************/

////////////////////////////////////////////////////////////////////////////
/*************************************************************************/
/*************************************************************************/
/*************************************************************************/
        if(temperatura_actual <= 11){
                intervalo = 1;
        }else if((temperatura_actual > 11) &&(temperatura_actual < 14)){
                intervalo = 2;
        }else if(temperatura_actual >= 14){
                intervalo = 3;
        }
/*************************************************************************/

   bomba_aireadora(); //activa o desactiva la bomba aireadora
/*************************************************************************/
/*************************************************************************/
////////////////////////////////////////////////////////////////////////////
/*************************************************************************/
        fotoperiodo();

}

if((flag_empezado_modo_fructif==true)&&(contador_iniciado_horas_modo==0)){
        stop_modo_fructif();
}//if modo fructif

}//modo fructif
/*************************************************************************/
/*************************************************************************/
/*************************************************************************/

////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION MODO_AUTO
// funcion que que se corresponde al hilo de ejecucion MODO_AUTO en el loop()
////////////////////////////////////////////////////////////////////////////////////////////////
void modo_auto(){

/
***********************************************************************
********/
   //1º modo_auto_grow;
/
***********************************************************************
********/
if( (flag_empezado_modo_grow==true) && ( flag_empezado_modo_auto==true) &&
(flag_empezado_modo_auto_2a_fase==false)){

////////
         modo_grow();
////////
}else if( (flag_empezado_modo_grow==false) && (flag_empezado_modo_fructif==false) &&
( flag_empezado_modo_auto==true) && (flag_empezado_modo_auto_2a_fase==false)){
        stop_modo_grow();
```

```
          flag_empezado_modo_auto_2a_fase=true;
    lcd.setCursor(14, 1);
    lcd.print(F("AF"));
          start_modo_fructif();
}//if modo auto grow



/
**********************************************************************
*********/
   //2º modo_auto_fructif();
/
**********************************************************************
*********/

if((flag_empezado_modo_fructif==true) && ( flag_empezado_modo_auto==true) &&
(flag_empezado_modo_auto_2a_fase==true) ){

////////
          modo_fructif();
////////
}else if( (flag_empezado_modo_fructif==false) && (flag_empezado_modo_grow==false) &&
( flag_empezado_modo_auto==true) && (flag_empezado_modo_auto_2a_fase==true) ){
          stop_modo_auto(); //fin de modo auto, incluye fin de modo grow y fructif
}//if modo auto fructif

}//modo auto

/*************************************************************************************/
/*************************************************************************************/
/*************************************************************************************/




void loop()
////////////////////////////////////////////////////////////////////////////////////////////////////
// INICIO CALCULO DE MODOS DE FUNCIONAMIENTO
////////////////////////////////////////////////////////////////////////////////////////////////////
//CADA DOS SEGUNDOS
if(bandera3==true){
    /*********************************/
    /***** LEO MODOS ACTIVOS *****/
          modo_grow();
          modo_fructif();
          modo_auto();
    /*********************************/
    /*********************************/

    bandera3=false;
}//if bandera3
////////////////////////////////////////////////////////////////////////////////////////////////////
// FIN de CALCULO DE MODOS DE FUNCIONAMIENTO
////////////////////////////////////////////////////////////////////////////////////////////////////


}//loop
```

4.3 Código dedicado a la ejecución de interrupciones ISR

```
////////////////////////////////////////////////////////////////////////////
//INTERRUPCIONES CADA SEGUNDO Y CONTADORES RETARDOS
////////////////////////////////////////////////////////////////////////////
ISR(TIMER1_COMPA_vect)
{

  segundos++;


  /////////////// RETARDO RELES  ////////////////////
  ////////////////////////////////////////////////
      if(contador_iniciado_reles<=0){
         contador_iniciado_reles = 0;
      }else{
         contador_iniciado_reles--;
      }
  ////////////////////////////////////////////////
  ////////////////////////////////////////////////
  ///////// TIEMPO EN ON BOMBA DE AIRE //////////
  ////////////////////////////////////////////////
      if(contador_iniciado_fuzzy_aire_bomba<=0){
         contador_iniciado_fuzzy_aire_bomba = 0;
      }else{
         contador_iniciado_fuzzy_aire_bomba--;
      }
  ////////////////////////////////////////////////
  ////////////////////////////////////////////////
  ///////// TIEMPO EN OFF BOMBA DE AIRE //////////
  ////////////////////////////////////////////////
      if(contador_iniciado_fuzzy_aire_bomba_en_OFF<=0){
         contador_iniciado_fuzzy_aire_bomba_en_OFF = 0;
      }else{
         contador_iniciado_fuzzy_aire_bomba_en_OFF--;
                        if(contador_iniciado_fuzzy_aire_bomba_en_OFF<=0){
                                 temperatura_actual = (float)dht.readTemperature();
                                 humedad_actual = (float)dht.readHumidity();
                                 delta_temp = temperatura_actual - temperatura_inicial;
                                 delta_humedad = humedad_actual - humedad_inicial;
                                 temperatura_inicial = temperatura_actual;
                                 humedad_inicial = humedad_actual;
                           }//hago los deltas
      }
  ////////////////////////////////////////////////
  ////////////////////////////////////////////////
  ////// TIEMPO DEL HUMIDIFICADOR EN ON    ////////
  ////////////////////////////////////////////////
      if(contador_iniciado_humidificador<=0){
         contador_iniciado_humidificador = 0;
      }else{
         contador_iniciado_humidificador--;
      }
  ////////////////////////////////////////////////
  ////////////////////////////////////////////////
  /////// TIEMPO DEL HUMIDIFICADOR EN OFF   ///////
  ////////////////////////////////////////////////
      if(contador_iniciado_humidificador_en_OFF<=0){
         contador_iniciado_humidificador_en_OFF = 0;
      }else{
         contador_iniciado_humidificador_en_OFF--;
      }
  ////////////////////////////////////////////////
  ////////////////////////////////////////////////

  bandera1=true;
  /*************************************************/
  /* La bandera la hago cada 10 segundos ************/
```

```
  /* La usare para la lectura de los sensores *******/
  if((segundos%10) == 1){
    bandera2=true;
  }//if
  /*************************************************/
  /*************************************************/
  /* La bandera la hago cada 2 segundos ************/
  /* La usare para la lectura de los sensores *******/
  if((segundos%2) == 1){
    bandera3=true;
  }//if
  /*************************************************/


  if(segundos>59)
    {
     segundos=0;
     minutos++;
     if(minutos>59)
        {
         minutos=0;
         horas++;

  //////// TIEMPO MODO FUNCIONAMIENTO /////////
  ////////////////////////////////////////////////
       if(contador_iniciado_horas_modo<=0){
          contador_iniciado_horas_modo = 0;
       }else{
          contador_iniciado_horas_modo--;
       }
  ////////////////////////////////////////////////
  ////////////////////////////////////////////////
  ////////////// TIEMPO FOTOPERIODO ON    //////////
  ////////////////////////////////////////////////
       if(contador_iniciado_fotoperiodo<=0){
          contador_iniciado_fotoperiodo = 0;
       }else{
          contador_iniciado_fotoperiodo--;
       }
  ////////////////////////////////////////////////
  ////////////////////////////////////////////////

  ////////////// TIEMPO FOTOPERIODO OFF    //////////
  ////////////////////////////////////////////////
       if(contador_iniciado_fotoperiodo_en_OFF<=0){
          contador_iniciado_fotoperiodo_en_OFF = 0;
       }else{
          contador_iniciado_fotoperiodo_en_OFF--;
       }
  ////////////////////////////////////////////////
  ////////////////////////////////////////////////

          if(horas>23)
            {
             segundos=0;
             minutos=0;
             horas=0;
            }
        }
    }
}
```

5. Funciones y parámetros para la implementacion del fuzzy en el control del caudal de aire del invernadero

```
////////////////////////////////////////////////////////////////////////////////////////////////////
**************************************************************************************
//   INICIO DE FUNCIONES Y DATOS FUZZY EMPLEADOS EN NUESTRO SISTEMA DE CONTROL
//      VARIABLE SOBRE LA QUE ACTUA: Caudal de aire de la Bomba Aireadora
/**************************************************************************************
////////////////////////////////////////////////////////////////////////////////////////////////////
//INICIALIZACION DE OBJETOS FUZZY
////////////////////////////////////////////////////////////////////////////////////////////////////
int fis_gcI;  // Numero de entradas al fuzzy inference system
int fis_gcO;  // Numero de outputs al fuzzy inference system
int fis_gcR;  // Numero de reglas al fuzzy inference system

FIS_TYPE g_fisInput_i1[fis_gcI_i1];
FIS_TYPE g_fisOutput_i1[fis_gcO_i1];

FIS_TYPE g_fisInput_i2[fis_gcI_i2];
FIS_TYPE g_fisOutput_i2[fis_gcO_i2];

FIS_TYPE g_fisInput_i3[fis_gcI_i3];
FIS_TYPE g_fisOutput_i3[fis_gcO_i3];

FIS_TYPE *g_fisInput[2];
FIS_TYPE *g_fisOutput[1];

//************************************************************************
// Funciones de soporte para Fuzzy Inference System
//************************************************************************
// Trapezoidal Member Function
FIS_TYPE fis_trapmf(FIS_TYPE x, FIS_TYPE* p)
{
   FIS_TYPE a = p[0], b = p[1], c = p[2], d = p[3];
   FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ? 0 : ((c != d) ? ((d - x) / (d - c)) : 0)));
   FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ? 0 : ((a != b) ? ((x - a) / (b - a)) : 0)));
   return (FIS_TYPE) min(t1, t2);
}

// Triangular Member Function
FIS_TYPE fis_trimf(FIS_TYPE x, FIS_TYPE* p)
{
   FIS_TYPE a = p[0], b = p[1], c = p[2];
   FIS_TYPE t1 = (x - a) / (b - a);
   FIS_TYPE t2 = (c - x) / (c - b);
   if ((a == b) && (b == c)) return (FIS_TYPE) (x == a);
   if (a == b) return (FIS_TYPE) (t2*(b <= x)*(x <= c));
   if (b == c) return (FIS_TYPE) (t1*(a <= x)*(x <= b));
   t1 = min(t1, t2);
   return (FIS_TYPE) max(t1, 0);
}

//Funcion MIN
FIS_TYPE fis_min(FIS_TYPE a, FIS_TYPE b)
{
   return min(a, b);
}

//Funcion MAX
FIS_TYPE fis_max(FIS_TYPE a, FIS_TYPE b)
{
   return max(a, b);
}

FIS_TYPE fis_array_operation(FIS_TYPE *array, int size, _FIS_ARR_OP pfnOp)
{
   int i;
   FIS_TYPE ret = 0;
```

```
    if (size == 0) return ret;
    if (size == 1) return array[0];

    ret = array[0];
    for (i = 1; i < size; i++)
    {
        ret = (*pfnOp)(ret, array[i]);
    }

    return ret;
}


//***********************************************************************
// Data for Fuzzy Inference System
//***********************************************************************
// Pointers to the implementations of member functions
_FIS_MF fis_gMF[] =
{
    fis_trapmf, fis_trimf
};

int fis_gIMFCount_i1[] = { 3, 3 }; // Count of member function for each Input
int fis_gOMFCount_i1[] = { 3 }; // Count of member function for each Output

int fis_gIMFCount_i2[] = { 3 }; // Count of member function for each Input
int fis_gOMFCount_i2[] = { 3 }; // Count of member function for each Output

int fis_gIMFCount_i3[] = { 3, 3 }; // Count of member function for each Input
int fis_gOMFCount_i3[] = { 3 }; // Count of member function for each Output

int *fis_gIMFCount;
int *fis_gOMFCount;

//########################################################################
##
//########################################################################
##
FIS_TYPE* fis_gMFI0Coeff[5]; // = { fis_gMFI0Coeff1_i1, fis_gMFI0Coeff2_i1, fis_gMFI0Coeff3_i1, NULL };
FIS_TYPE* fis_gMFI1Coeff[5]; // = { fis_gMFI1Coeff1_i1, fis_gMFI1Coeff2_i1, fis_gMFI1Coeff3_i1, NULL  };
FIS_TYPE** fis_gMFICoeff[2]; // = { fis_gMFI0Coeff_i1, fis_gMFI1Coeff_i1, NULL  };

FIS_TYPE* fis_gMFO0Coeff[3]; // = { fis_gMFO0Coeff1, fis_gMFO0Coeff2, fis_gMFO0Coeff3, NULL  };
FIS_TYPE** fis_gMFOCoeff[1]; // = { fis_gMFO0Coeff, NULL  };

int* fis_gMFI[2]; // = { fis_gMFI0_i1, fis_gMFI1_i1, NULL };

int* fis_gMFO[1]; // = { fis_gMFO0_i1, NULL };

FIS_TYPE fis_gRWeight[5]; // = { 1, 1, 1, 1, 1 , NULL};
int fis_gRType[5]; // = { 1, 1, 1, 1, 1 , NULL};

int* fis_gRI[5]; // = { fis_gRI0_i1, fis_gRI1_i1, fis_gRI2_i1, fis_gRI3_i1, fis_gRI4_i1, NULL  };

int* fis_gRO[5]; // = { fis_gRO0_i1, fis_gRO1_i1, fis_gRO2_i1, fis_gRO3_i1, fis_gRO4_i1, NULL  };


/*************************************************************************/
/*INTERVALO 1*/
/*************************************************************************/

// Coefficients for the Input Member Functions
FIS_TYPE fis_gMFI0Coeff1_i1[] = { -8.6, -5.4, -1.5, 0 };
FIS_TYPE fis_gMFI0Coeff2_i1[] = { -0.8, 0, 0.8 };
FIS_TYPE fis_gMFI0Coeff3_i1[] = { 0, 1.5, 5.4, 8.6 };
//FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1_i1, fis_gMFI0Coeff2_i1, fis_gMFI0Coeff3_i1, NULL };
```

```
FIS_TYPE fis_gMFI1Coeff1_i1[] = { -27.52, -17.28, -5, 0 };
FIS_TYPE fis_gMFI1Coeff2_i1[] = { -5, 0, 5 };
FIS_TYPE fis_gMFI1Coeff3_i1[] = { 0, 5, 17.28, 27.52 };
//FIS_TYPE* fis_gMFI1Coeff[] = { fis_gMFI1Coeff1_i1, fis_gMFI1Coeff2_i1, fis_gMFI1Coeff3_i1, NULL  };
//FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff_i1, fis_gMFI1Coeff_i1, NULL  };

// Coefficients for the Output Member Functions
FIS_TYPE fis_gMFO0Coeff1_i1[] = { -24, 0, 24 };
FIS_TYPE fis_gMFO0Coeff2_i1[] = { 6, 30, 54 };
FIS_TYPE fis_gMFO0Coeff3_i1[] = { 36, 60, 84 };
//FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1, fis_gMFO0Coeff2, fis_gMFO0Coeff3, NULL  };
//FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff, NULL  };

// Input membership function set
int fis_gMFI0_i1[] = { 0, 1, 0 }; //trapecio, triangulo, trapecio
int fis_gMFI1_i1[] = { 0, 1, 0 }; //trapecio, triangulo, trapecio
//int* fis_gMFI[] = { fis_gMFI0_i1, fis_gMFI1_i1, NULL };

// Output membership function set
int fis_gMFO0_i1[] = { 1, 1, 1 }; //triangulo, triangulo, triangulo
//int* fis_gMFO[] = { fis_gMFO0_i1, NULL };

// Rule Weights
//FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1 , NULL};

// Rule Type
//int fis_gRType[] = { 1, 1, 1, 1, 1 , NULL};

// Rule Inputs
int fis_gRI0_i1[] = { 1, 1 };
int fis_gRI1_i1[] = { 2, 1 };
int fis_gRI2_i1[] = { 3, 1 };
int fis_gRI3_i1[] = { 3, 2 };
int fis_gRI4_i1[] = { 3, 3 };
//int* fis_gRI[] = { fis_gRI0_i1, fis_gRI1_i1, fis_gRI2_i1, fis_gRI3_i1, fis_gRI4_i1, NULL  };

// Rule Outputs
int fis_gRO0_i1[] = { 1 };
int fis_gRO1_i1[] = { 2 };
int fis_gRO2_i1[] = { 2 };
int fis_gRO3_i1[] = { 3 };
int fis_gRO4_i1[] = { 3 };
//int* fis_gRO[] = { fis_gRO0_i1, fis_gRO1_i1, fis_gRO2_i1, fis_gRO3_i1, fis_gRO4_i1, NULL  };

/**********************************************************************/
/* FIN INTERVALO 1*/
/**********************************************************************/

/**********************************************************************/
/*INTERVALO 2*/
/**********************************************************************/
// Coefficients for the Input Member Functions
FIS_TYPE fis_gMFI0Coeff1_i2[] = { -8.6, -5.4, -1.5, 0 };
FIS_TYPE fis_gMFI0Coeff2_i2[] = { -0.8, 0, 0.8 };
FIS_TYPE fis_gMFI0Coeff3_i2[] = { 0, 1.5, 5.4, 8.6 };
//FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1_i2, fis_gMFI0Coeff2_i2, fis_gMFI0Coeff3_i2, NULL  };
//FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff , NULL  };

FIS_TYPE fis_gMFI1Coeff1_i2[] = { -27.52, -17.28, -5, 0 };
FIS_TYPE fis_gMFI1Coeff2_i2[] = { -5, 0, 5 };
FIS_TYPE fis_gMFI1Coeff3_i2[] = { 0, 5, 17.28, 27.52 };


// Coefficients for the Output Member Functions
FIS_TYPE fis_gMFO0Coeff1_i2[] = { -24, 0, 24 };
FIS_TYPE fis_gMFO0Coeff2_i2[] = { 6, 30, 54 };
FIS_TYPE fis_gMFO0Coeff3_i2[] = { 36, 60, 84 };
//FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1_i2, fis_gMFO0Coeff2_i2, fis_gMFO0Coeff3_i2 , NULL  };
//FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff , NULL  };
```

```
// Input membership function set
int fis_gMFI0_i2[] = { 0, 1, 0 };
int fis_gMFI1_i2[] = { 0, 1, 0 }; /////
//int* fis_gMFI[] = { fis_gMFI0_i2, NULL  };

// Output membership function set
int fis_gMFO0_i2[] = { 1, 1, 1 };
//int* fis_gMFO[] = { fis_gMFO0_i2, NULL  };

// Rule Weights
//FIS_TYPE fis_gRWeight[] = { 1, 1, 1 , NULL};

// Rule Type
//int fis_gRType[] = { 1, 1, 1 , NULL};

// Rule Inputs
int fis_gRI0_i2[] = { 1, 0 };
int fis_gRI1_i2[] = { 2, 0 };
int fis_gRI2_i2[] = { 3, 0 };
int fis_gRI3_i2[] = { 0, 0 };
int fis_gRI4_i2[] = { 0, 0 };
//int* fis_gRI[] = { fis_gRI0_i2, fis_gRI1_i2, fis_gRI2_i2 , NULL  };

// Rule Outputs
int fis_gRO0_i2[] = { 1 };
int fis_gRO1_i2[] = { 2 };
int fis_gRO2_i2[] = { 1 };
int fis_gRO3_i2[] = { 0 };
int fis_gRO4_i2[] = { 0 };
//int* fis_gRO[] = { fis_gRO0_i2, fis_gRO1_i2, fis_gRO2_i2 , NULL  };

/*************************************************************************/
/* FIN INTERVALO 2*/
/*************************************************************************/

/*************************************************************************/
/*INTERVALO 3*/
/*************************************************************************/
// Coefficients for the Input Member Functions
FIS_TYPE fis_gMFI0Coeff1_i3[] = { -8.6, -5.4, -1.5, 0 };
FIS_TYPE fis_gMFI0Coeff2_i3[] = { -0.8, 0, 0.8 };
FIS_TYPE fis_gMFI0Coeff3_i3[] = { 0, 1.5, 5.4, 8.6 };
//FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1_i2, fis_gMFI0Coeff2_i2, fis_gMFI0Coeff3_i2, NULL  };
//FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff , NULL  };

FIS_TYPE fis_gMFI1Coeff1_i3[] = { -27.52, -17.28, -5, 0 };
FIS_TYPE fis_gMFI1Coeff2_i3[] = { -5, 0, 5 };
FIS_TYPE fis_gMFI1Coeff3_i3[] = { 0, 5, 17.28, 27.52 };


// Coefficients for the Output Member Functions
FIS_TYPE fis_gMFO0Coeff1_i3[] = { -24, 0, 24 };
FIS_TYPE fis_gMFO0Coeff2_i3[] = { 6, 30, 54 };
FIS_TYPE fis_gMFO0Coeff3_i3[] = { 36, 60, 84 };
//FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1_i2, fis_gMFO0Coeff2_i2, fis_gMFO0Coeff3_i2 , NULL  };
//FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff , NULL  };

// Input membership function set
int fis_gMFI0_i3[] = { 0, 1, 0 };
int fis_gMFI1_i3[] = { 0, 1, 0 }; /////
//int* fis_gMFI[] = { fis_gMFI0_i2, NULL  };

// Output membership function set
int fis_gMFO0_i3[] = { 1, 1, 1 };
//int* fis_gMFO[] = { fis_gMFO0_i2, NULL  };
```

```
// Rule Weights
//FIS_TYPE fis_gRWeight[] = { 1, 1, 1 , NULL};

// Rule Type
//int fis_gRType[] = { 1, 1, 1 , NULL};

// Rule Inputs
int fis_gRI0_i3[] = { 1, 2 };
int fis_gRI1_i3[] = { 2, 2 };
int fis_gRI2_i3[] = { 3, 0 };
int fis_gRI3_i3[] = { 0, 0 };
int fis_gRI4_i3[] = { 0, 0 };
//int* fis_gRI[] = { fis_gRI0_i2, fis_gRI1_i2, fis_gRI2_i2 , NULL  };

// Rule Outputs
int fis_gRO0_i3[] = { 3 };
int fis_gRO1_i3[] = { 2 };
int fis_gRO2_i3[] = { 1 };
int fis_gRO3_i3[] = { 0 };
int fis_gRO4_i3[] = { 0 };
//int* fis_gRO[] = { fis_gRO0_i2, fis_gRO1_i2, fis_gRO2_i2 , NULL  };

/*************************************************************************/
/* FIN INTERVALO 3*/
/*************************************************************************/



// Input range Min
FIS_TYPE fis_gIMin[] = { -5, -16 };

// Input range Max
FIS_TYPE fis_gIMax[] = { 5, 16 };

// Output range Min
FIS_TYPE fis_gOMin[] = { 0 };

// Output range Max
FIS_TYPE fis_gOMax[] = { 60 };

//************************************************************************
// Data dependent support functions for Fuzzy Inference System
//************************************************************************
FIS_TYPE fis_MF_out(FIS_TYPE** fuzzyRuleSet, FIS_TYPE x, int o)
{
   FIS_TYPE mfOut;
   int r;

   for (r = 0; r < fis_gcR; ++r)
   {
      int index = fis_gRO[r][o];
      if (index > 0)
      {
         index = index - 1;
         mfOut = (fis_gMF[fis_gMFO[o][index]])(x, fis_gMFOCoeff[o][index]);
      }
      else if (index < 0)
      {
         index = -index - 1;
         mfOut = 1 - (fis_gMF[fis_gMFO[o][index]])(x, fis_gMFOCoeff[o][index]);
      }
      else
      {
         mfOut = 0;
      }

      fuzzyRuleSet[0][r] = fis_min(mfOut, fuzzyRuleSet[1][r]);
   }
   return fis_array_operation(fuzzyRuleSet[0], fis_gcR, fis_max);
```

```
}

//Funcion para el calculo del centroide en el proceso de defuzzy
FIS_TYPE fis_defuzz_centroid(FIS_TYPE** fuzzyRuleSet, int o)
{
    FIS_TYPE step = (fis_gOMax[o] - fis_gOMin[o]) / (FIS_RESOLUSION - 1);
    FIS_TYPE area = 0;
    FIS_TYPE momentum = 0;
    FIS_TYPE dist, slice;
    int i;

    // calculate the area under the curve formed by the MF outputs
    for (i = 0; i < FIS_RESOLUSION; ++i){
        dist = fis_gOMin[o] + (step * i);
        slice = step * fis_MF_out(fuzzyRuleSet, dist, o);
        area += slice;
        momentum += slice*dist;
    }

    return ((area == 0) ? ((fis_gOMax[o] + fis_gOMin[o]) / 2) : (momentum / area));
}

//************************************************************************
// Fuzzy Inference System
// Calcula el valor de salida
//************************************************************************
void fis_evaluate()
{

    FIS_TYPE fuzzyInput0[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput1[] = { 0, 0, 0 };
    FIS_TYPE* fuzzyInput[] = {fuzzyInput0, fuzzyInput1};
    //FIS_TYPE fuzzyOutput0[] = { 0, 0, 0 };
    //FIS_TYPE* fuzzyOutput[] = { fuzzyOutput0};
    FIS_TYPE fuzzyRules[MAXIMO_DE_RULES] = { 0 };
    FIS_TYPE fuzzyFires[MAXIMO_DE_RULES] = { 0 };
    FIS_TYPE* fuzzyRuleSet[] = { fuzzyRules, fuzzyFires };
    FIS_TYPE sW = 0;


FIS_TYPE g_fisInput_local[] = { *g_fisInput[0], *g_fisInput[1] };
FIS_TYPE g_fisOutput_local[] = { *g_fisOutput[0] };

    int i, j, r, o;
    // Transforming input to fuzzy Input
    for (i = 0; i < fis_gcI; ++i)
    {
        for (j = 0; j < fis_gIMFCount[i]; ++j)
        {
            fuzzyInput[i][j] =
                (fis_gMF[fis_gMFI[i][j]])(g_fisInput_local[i], fis_gMFICoeff[i][j]);
        }
    }

    int index = 0;
    for (r = 0; r < fis_gcR; ++r)
    {
        if (fis_gRType[r] == 1)
        {
            fuzzyFires[r] = FIS_MAX;
            for (i = 0; i < fis_gcI; ++i)
            {
                index = fis_gRI[r][i];
                if (index > 0)
                    fuzzyFires[r] = fis_min(fuzzyFires[r], fuzzyInput[i][index - 1]);
                else if (index < 0)
                    fuzzyFires[r] = fis_min(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
```

```
            else
               fuzzyFires[r] = fis_min(fuzzyFires[r], 1);

         }
      }
      else
      {
         fuzzyFires[r] = FIS_MIN;
         for (i = 0; i < fis_gcI; ++i)
         {
            index = fis_gRI[r][i];
            if (index > 0)
               fuzzyFires[r] = fis_max(fuzzyFires[r], fuzzyInput[i][index - 1]);
            else if (index < 0)
               fuzzyFires[r] = fis_max(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
            else
               fuzzyFires[r] = fis_max(fuzzyFires[r], 0);
         }
      }

      fuzzyFires[r] = fis_gRWeight[r] * fuzzyFires[r];
      sW += fuzzyFires[r];
   }

   if (sW == 0)
   {
      for (o = 0; o < fis_gcO; ++o)
      {
         g_fisOutput_local[o] = ((fis_gOMax[o] + fis_gOMin[o]) / 2);
      }
   }
   else
   {
      for (o = 0; o < fis_gcO; ++o)
      {
         g_fisOutput_local[o] = fis_defuzz_centroid(fuzzyRuleSet, o);
         *g_fisOutput[o]=g_fisOutput_local[o];
      }
   }
}

////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION INTERVALOS
// funcion que nos devuelve el valor calculado FUZZY del sistema de control.
// PARAMETROS DE ENTRADA: El intervalo en el que haremos el calculo FUZZY (1, 2 o 3) y el DELTA_T y el
DELTA_H para
// la tendencia hacia arriba o hacia abajo de la temperatura y de la humedad
////////////////////////////////////////////////////////////////////////////////////////////////////
int intervalos(int intervalo, float delta_temp, float delta_humi){

//float caudal_output=0;
if (flag_habilitado_fuzzy==true){
   switch(intervalo){
   case 1:

/*INTERVALO 1*/
fis_gcI=fis_gcI_i1;  // Number of inputs to the fuzzy inference system
fis_gcO=fis_gcO_i1;  // Number of outputs to the fuzzy inference system
fis_gcR=fis_gcR_i1;  // Number of rules to the fuzzy inference system


g_fisInput[0] = &g_fisInput_i1[0];
g_fisInput[1] = &g_fisInput_i1[1];
g_fisOutput[0] = &g_fisOutput_i1[0];
fis_gIMFCount=&fis_gIMFCount_i1[0];
fis_gOMFCount=&fis_gOMFCount_i1[0];
```

```
fis_gMFI0Coeff[0] = &fis_gMFI0Coeff1_i1[0];
fis_gMFI0Coeff[1] = &fis_gMFI0Coeff2_i1[0];
fis_gMFI0Coeff[2] = &fis_gMFI0Coeff3_i1[0];
fis_gMFI1Coeff[0] = &fis_gMFI1Coeff1_i1[0];
fis_gMFI1Coeff[1] = &fis_gMFI1Coeff2_i1[0];
fis_gMFI1Coeff[2] = &fis_gMFI1Coeff3_i1[0];
fis_gMFICoeff[0] = &fis_gMFI0Coeff[0];
fis_gMFICoeff[1] = &fis_gMFI1Coeff[0];

fis_gMFO0Coeff[0] = &fis_gMFO0Coeff1_i1[0];
fis_gMFO0Coeff[1] = &fis_gMFO0Coeff2_i1[0];
fis_gMFO0Coeff[2] = &fis_gMFO0Coeff3_i1[0];
fis_gMFOCoeff[0] = &fis_gMFO0Coeff[0];

fis_gMFI[0] = &fis_gMFI0_i1[0];
fis_gMFI[1] = &fis_gMFI1_i1[0];

fis_gMFO[0] = &fis_gMFO0_i1[0];

fis_gRWeight[0] = 1;
fis_gRWeight[1] = 1;
fis_gRWeight[2] = 1;
fis_gRWeight[3] = 1;
fis_gRWeight[4] = 1;
fis_gRType[0] = 1;
fis_gRType[1] = 1;
fis_gRType[2] = 1;
fis_gRType[3] = 1;
fis_gRType[4] = 1;

fis_gRI[0] = &fis_gRI0_i1[0];
fis_gRI[1] = &fis_gRI1_i1[0];
fis_gRI[2] = &fis_gRI2_i1[0];
fis_gRI[3] = &fis_gRI3_i1[0];
fis_gRI[4] = &fis_gRI4_i1[0];

fis_gRO[0] = &fis_gRO0_i1[0];
fis_gRO[1] = &fis_gRO1_i1[0];
fis_gRO[2] = &fis_gRO2_i1[0];
fis_gRO[3] = &fis_gRO3_i1[0];
fis_gRO[4] = &fis_gRO4_i1[0];

            // Read Input: delta_T
               g_fisInput_i1[0] = delta_temp;
            // Read Input: delta_H
               g_fisInput_i1[1] = delta_humi;
            //float caudal_output = 0;
               g_fisOutput_i1[0] = 44;
               fis_evaluate();
            // Set output value: caudal
      return (int(g_fisOutput_i1[0]));
   break;
   case 2:

/*INTERVALO 2*/
fis_gcI=fis_gcI_i2;  // Number of inputs to the fuzzy inference system
fis_gcO=fis_gcO_i2;  // Number of outputs to the fuzzy inference system
fis_gcR=fis_gcR_i2;  // Number of rules to the fuzzy inference system

g_fisInput[0] = &g_fisInput_i2[0];
g_fisInput[1] = &g_fisInput_i2[1];
g_fisOutput[0] = &g_fisOutput_i2[0];
fis_gIMFCount=&fis_gIMFCount_i2[0];
fis_gOMFCount=&fis_gOMFCount_i2[0];

fis_gMFI0Coeff[0] = &fis_gMFI0Coeff1_i2[0];
```

```
fis_gMFI0Coeff[1] = &fis_gMFI0Coeff2_i2[0];
fis_gMFI0Coeff[2] = &fis_gMFI0Coeff3_i2[0];
fis_gMFI1Coeff[0] = &fis_gMFI1Coeff1_i2[0];
fis_gMFI1Coeff[1] = &fis_gMFI1Coeff2_i2[0];
fis_gMFI1Coeff[2] = &fis_gMFI1Coeff3_i2[0];
fis_gMFICoeff[0] = &fis_gMFI0Coeff[0];
fis_gMFICoeff[1] = &fis_gMFI1Coeff[0];

fis_gMFO0Coeff[0] = &fis_gMFO0Coeff1_i2[0];
fis_gMFO0Coeff[1] = &fis_gMFO0Coeff2_i2[0];
fis_gMFO0Coeff[2] = &fis_gMFO0Coeff3_i2[0];
fis_gMFOCoeff[0] = &fis_gMFO0Coeff[0];

fis_gMFI[0] = &fis_gMFI0_i2[0];
fis_gMFI[1] = &fis_gMFI1_i2[0];

fis_gMFO[0] = &fis_gMFO0_i2[0];

fis_gRWeight[0] = 1;
fis_gRWeight[1] = 1;
fis_gRWeight[2] = 1;
fis_gRWeight[3] = 1;
fis_gRWeight[4] = 1;
fis_gRType[0] = 1;
fis_gRType[1] = 1;
fis_gRType[2] = 1;
fis_gRType[3] = 1;
fis_gRType[4] = 1;

fis_gRI[0] = &fis_gRI0_i2[0];
fis_gRI[1] = &fis_gRI1_i2[0];
fis_gRI[2] = &fis_gRI2_i2[0];
fis_gRI[3] = &fis_gRI3_i2[0];
fis_gRI[4] = &fis_gRI4_i2[0];

fis_gRO[0] = &fis_gRO0_i2[0];
fis_gRO[1] = &fis_gRO1_i2[0];
fis_gRO[2] = &fis_gRO2_i2[0];
fis_gRO[3] = &fis_gRO3_i2[0];
fis_gRO[4] = &fis_gRO4_i2[0];

            // Read Input: delta_T
               g_fisInput_i2[0] = delta_temp;
            // Read Input: delta_H
               g_fisInput_i2[1] = delta_humi;
            //float caudal_output = 0;
               g_fisOutput_i2[0] = 44;
               fis_evaluate();
            // Set output value: caudal
      return (int(g_fisOutput_i2[0]));
   break;
   case 3:

/*INTERVALO 3*/
fis_gcI=fis_gcI_i3;  // Number of inputs to the fuzzy inference system
fis_gcO=fis_gcO_i3;  // Number of outputs to the fuzzy inference system
fis_gcR=fis_gcR_i3;  // Number of rules to the fuzzy inference system


g_fisInput[0] = &g_fisInput_i3[0];
g_fisInput[1] = &g_fisInput_i3[1];
g_fisOutput[0] = &g_fisOutput_i3[0];
fis_gIMFCount=&fis_gIMFCount_i3[0];
fis_gOMFCount=&fis_gOMFCount_i3[0];


fis_gMFI0Coeff[0] = &fis_gMFI0Coeff1_i3[0];
fis_gMFI0Coeff[1] = &fis_gMFI0Coeff2_i3[0];
fis_gMFI0Coeff[2] = &fis_gMFI0Coeff3_i3[0];
```

```
fis_gMFI1Coeff[0] = &fis_gMFI1Coeff1_i3[0];
fis_gMFI1Coeff[1] = &fis_gMFI1Coeff2_i3[0];
fis_gMFI1Coeff[2] = &fis_gMFI1Coeff3_i3[0];
fis_gMFICoeff[0] = &fis_gMFI0Coeff[0];
fis_gMFICoeff[1] = &fis_gMFI1Coeff[0];

fis_gMFO0Coeff[0] = &fis_gMFO0Coeff1_i3[0];
fis_gMFO0Coeff[1] = &fis_gMFO0Coeff2_i3[0];
fis_gMFO0Coeff[2] = &fis_gMFO0Coeff3_i3[0];
fis_gMFOCoeff[0] = &fis_gMFO0Coeff[0];

fis_gMFI[0] = &fis_gMFI0_i3[0];
fis_gMFI[1] = &fis_gMFI1_i3[0];

fis_gMFO[0] = &fis_gMFO0_i3[0];

fis_gRWeight[0] = 1;
fis_gRWeight[1] = 1;
fis_gRWeight[2] = 1;
fis_gRWeight[3] = 1;
fis_gRWeight[4] = 1;
fis_gRType[0] = 1;
fis_gRType[1] = 1;
fis_gRType[2] = 1;
fis_gRType[3] = 1;
fis_gRType[4] = 1;

fis_gRI[0] = &fis_gRI0_i3[0];
fis_gRI[1] = &fis_gRI1_i3[0];
fis_gRI[2] = &fis_gRI2_i3[0];
fis_gRI[3] = &fis_gRI3_i3[0];
fis_gRI[4] = &fis_gRI4_i3[0];

fis_gRO[0] = &fis_gRO0_i3[0];
fis_gRO[1] = &fis_gRO1_i3[0];
fis_gRO[2] = &fis_gRO2_i3[0];
fis_gRO[3] = &fis_gRO3_i3[0];
fis_gRO[4] = &fis_gRO4_i3[0];

            // Read Input: delta_T
               g_fisInput_i3[0] = delta_temp;
            // Read Input: delta_H
               g_fisInput_i3[1] = delta_humi;
            //float caudal_output = 0;
               g_fisOutput_i3[0] = 44;
               fis_evaluate();
            // Set output value: caudal
      return (int(g_fisOutput_i3[0]));
   break;
   }//switch
}//if fuzzy habilitado

//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
//mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm

}//intervalos

////////////////////////////////////////////////////////////////////////////////////////////////////////
// FIN DE FUNCIONES Y DATOS FUZZY
////////////////////////////////////////////////////////////////////////////////////////////////////////
```

6. Funciones para el control del resto de componentes

Funciones para actuar sobre la bomba aireadora:

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_FUZZY_AIRE_BOMBA
// funcion que la usaremos para configurar los flags adecuadamente para que comience a funcionar la bomba
de aire en el loop()
// tambien activa el retardo correspondiente al ON
////////////////////////////////////////////////////////////////////////////////////////////////////
void start_fuzzy_aire_bomba(){
//////////////////////////////////////////////////////////////////
  int *mi_variable = &contador_iniciado_fuzzy_aire_bomba;
  float retardo_calculado_fuzzy=0;

if(contador_iniciado_fuzzy_aire_bomba!=0){

}else{
       digitalWrite(fuzzy_aire_bomba, HIGH);
          //contador_iniciado_fuzzy_aire_bomba = intervalos(intervalo, delta_temp, delta_humedad, Fuzzy*
fuzzy, Fuzzy* fuzzy2, Fuzzy* fuzzy3)
       retardo_calculado_fuzzy=intervalos(intervalo, delta_temp,
delta_humedad)/PONDERACION_REAJUSTE_FUZZY;

       if (int(retardo_calculado_fuzzy) == 0 ){
         retardo_calculado_fuzzy=1;  //nunca debe valer cero sino la bomba no se apagaria porque siempre
estaria haciendo el start
       }
          comienza_retardo(int(retardo_calculado_fuzzy), mi_variable);
          activado_flag_bomba_aireadora=false;
}//if
}//start_fuzzy_aire_bomba


////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_FUZZY_AIRE_BOMBA
// funcion que la usaremos para configurar los flags adecuadamente para que finalice la bomba de aire en el
loop()
// tambien activa el retardo correspondiente al OFF
////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_fuzzy_aire_bomba(){
  int *mi_variable = &contador_iniciado_fuzzy_aire_bomba_en_OFF;
//////////////////////////////////////////////////////////////////
if(contador_iniciado_fuzzy_aire_bomba_en_OFF!=0){

}else{
          digitalWrite(fuzzy_aire_bomba, LOW);
          //contador_iniciado_fuzzy_aire_bomba=0;
          //contador_iniciado_fuzzy_aire_bomba = intervalos(intervalo, delta_temp, delta_humedad, Fuzzy*
fuzzy, Fuzzy* fuzzy2, Fuzzy* fuzzy3)
       comienza_retardo(RETARDO_TIEMPO_EN_OFF_BOMBA, mi_variable);
       activado_flag_bomba_aireadora=true;
}//if
}//stop_fuzzy_aire_bomba


////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION BOMBA_AIREADORA
// funcion que se corresponde al hilo de ejecucion de la bomba de aire dentro del loop()
// si el hilo esta habilitado la bomba funciona, sino la bomba esta desconectada
////////////////////////////////////////////////////////////////////////////////////////////////////
void bomba_aireadora(){  //activara la bomba en el estado que indiquen los flags
if(flag_habilitado_fuzzy==false){
          digitalWrite(fuzzy_aire_bomba, LOW);
       contador_iniciado_fuzzy_aire_bomba=0;
          contador_iniciado_fuzzy_aire_bomba_en_OFF = 0;
          activado_flag_bomba_aireadora=true;
}else{
```

```
    if((activado_flag_bomba_aireadora==true)&&(contador_iniciado_fuzzy_aire_bomba_en_OFF==0)){
        start_fuzzy_aire_bomba();
    }else if(( activado_flag_bomba_aireadora==false)&&(contador_iniciado_fuzzy_aire_bomba==0)){
        stop_fuzzy_aire_bomba();
    }
}//if fuzzy habilitado
}
```

Funciones para actuar sobre el fotoperiodo:

```
////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_FOTOPERIODO
// funcion que la usaremos para configurar los flags adecuadamente para que comience a funcionar el
FOTOPERIODO en el loop()
// tambien activa el retardo correspondiente al ON
////////////////////////////////////////////////////////////////////////////////////////////////////////
void start_fotoperiodo(){
//////////////////////////////////////////////////////////////////
  int *mi_variable = &contador_iniciado_fotoperiodo;

if(contador_iniciado_fotoperiodo!=0){

}else{

          digitalWrite(ledPin, HIGH);
          el_LED_esta_encendido=true;
          //contador_iniciado_fotoperiodo = RETARDO_HORAS_FOTOPERIODO_ON;
          comienza_retardo(RETARDO_HORAS_FOTOPERIODO_ON, mi_variable);
          activado_flag_fotoperiodo=false;
  }//if
}//start_fuzzy_aire_bomba


////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_FOTOPERIODO
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el FOTOPERIODO en el
loop()
// tambien activa el retardo correspondiente al OFF
////////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_fotoperiodo(){
int *mi_variable = &contador_iniciado_fotoperiodo_en_OFF;
//////////////////////////////////////////////////////////////////
  if(  contador_iniciado_fotoperiodo_en_OFF != 0){

  }else{
      digitalWrite(ledPin, LOW);
                el_LED_esta_encendido=false;
      comienza_retardo(RETARDO_HORAS_FOTOPERIODO_OFF, mi_variable);
      activado_flag_fotoperiodo=true;
  }
}//stop_fuzzy_aire_bomba


////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION FOTOPERIODO
// funcion que se corresponde al hilo de ejecucion del fotoperiodo dentro del loop()
////////////////////////////////////////////////////////////////////////////////////////////////////////
void fotoperiodo(){  //activara el fotoperiodo en el estado que indiquen los flags
    if((activado_flag_fotoperiodo==true)&&(contador_iniciado_fotoperiodo_en_OFF==0)){
        start_fotoperiodo();
    }else if((activado_flag_fotoperiodo==false)&&(contador_iniciado_fotoperiodo==0)){
        stop_fotoperiodo();
    }
}
```

Funciones para actuar sobre el humidificador:

```
////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION START_HUMIDIFICADOR
// funcion que la usaremos para configurar los flags adecuadamente para que comience a funcionar el
HUMIDIFICADOR en el loop()
// tambien activa el retardo correspondiente al ON
////////////////////////////////////////////////////////////////////////////////////////////////////////
void start_humidificador(){
//////////////////////////////////////////////////////////////////
// PWM
//////////////////////////////////////////////////////////////////
  int *mi_variable = &contador_iniciado_humidificador;

if(contador_iniciado_humidificador!=0){

}else{


  pinMode(humidificador, OUTPUT);// output pin for OCR2B

  // Set up the 107kHz output
  TCCR2A = _BV(COM2A1) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
  TCCR2B = _BV(WGM22) | _BV(CS20);
  OCR2A = 149; //159
  OCR2B = 74; //79

  comienza_retardo(valor_fijado_RETARDO_TIEMPO_HUMIDIFICADOR_ON, mi_variable);
  //contador_iniciado_humidificador=RETARDO_TIEMPO_HUMIDIFICADOR;
  activado_flag_humidificador=false;
}

}//start_humidificador


////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION STOP_HUMIDIFICADOR
// funcion que la usaremos para configurar los flags adecuadamente para que finalice el HUMIDIFICADOR en el
loop()
// tambien activa el retardo correspondiente al OFF
////////////////////////////////////////////////////////////////////////////////////////////////////////
void stop_humidificador(){
  int *mi_variable = &contador_iniciado_humidificador_en_OFF;

  if(contador_iniciado_humidificador_en_OFF != 0){

  }else{

    digitalWrite(humidificador, LOW);
    comienza_retardo(RETARDO_TIEMPO_HUMIDIFICADOR_OFF, mi_variable);
          //contador_iniciado_humidificador=RETARDO_TIEMPO_HUMIDIFICADOR;
          activado_flag_humidificador=true;
  }

}//stop_humidificador


////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION HUMIDIFICADOR
// funcion que se corresponde al hilo de ejecucion del humidificador dentro del loop()
////////////////////////////////////////////////////////////////////////////////////////////////////////
void humidifica(){  //activara el humidificador en el estado que indiquen los flags

if(flag_habilitado_humidificador==false){
          digitalWrite(humidificador, LOW);
          contador_iniciado_humidificador=0;
          contador_iniciado_humidificador_en_OFF=0;
          activado_flag_humidificador=true;
}else{
  if (flag_hay_que_humidificar == true){
```

```
    if((activado_flag_humidificador==true) && (contador_iniciado_humidificador_en_OFF==0)){
        start_humidificador();
    }else if((activado_flag_humidificador==false) && (contador_iniciado_humidificador==0)){
        stop_humidificador();
    }


  }else{
    digitalWrite(humidificador, LOW);
    activado_flag_humidificador=true;
    contador_iniciado_humidificador_en_OFF=0;
    contador_iniciado_humidificador=0;
  }
}//if habilitado humidif
}
```

Funciones para actuar sobre los relés:

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION DESCONECTO_RELES
// funcion para apagar los reles directamente en el puente en H
////////////////////////////////////////////////////////////////////////////////////////////////////////////
void desconecto_reles(){
    digitalWrite(rele_A, RELE_APAGADO);
    digitalWrite(rele_B, RELE_APAGADO);
}
```

Funciones para actuar sobre la peltier:

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION ACTIVO_FLAG_PELTIER_FRIO
// funcion para indicar a la funcion peltier que podemos enfriar
////////////////////////////////////////////////////////////////////////////////////////////////////////////
void activo_flag_peltier_frio(){
    activado_reles_flag_frio = true;
    activado_reles_flag_calor = false;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION ACTIVO_FLAG_PELTIER_CALOR
// funcion para indicar a la funcion peltier que podemos calentar
////////////////////////////////////////////////////////////////////////////////////////////////////////////
void activo_flag_peltier_calor(){
    activado_reles_flag_frio = false;
    activado_reles_flag_calor = true;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION DESACTIVO_FLAG_PELTIER
// funcion para indicar a la funcion peltier que desconecte la peltier
// Ademas activa un pequeño retardo para que la peltier se recupere antes de volver a conectarla
////////////////////////////////////////////////////////////////////////////////////////////////////////////
void desactivo_flag_peltier(){
    int *mi_variable = &contador_iniciado_reles;
    comienza_retardo(RETARDO_CONEXION_RELES, mi_variable);
    activado_reles_flag_frio = false;
    activado_reles_flag_calor = false;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION PELTIER
// funcion para poner a la peltier en frio, en calor o desconectada, siempre que el retardo de la funcion
desconexion haya finalizado
```

```
// MUY IMPORTANTE: No hay que activar nunca el RELE_A y el RELE_B a la vez, poque hacemos un corto en la fuente de 12V.
// Siempre asegurarse del orden de conexion o desconectar todo primero. Primero apagar un rele y luego activar el otro.
////////////////////////////////////////////////////////////////////////////////////////////////////////
void peltier(){  //activara la peltier en el estado que indiquen los flags
   if(contador_iniciado_reles!=0){
   }else if((activado_reles_flag_frio==true) && (activado_reles_flag_calor==false)){

         digitalWrite(rele_B, RELE_APAGADO);
         digitalWrite(rele_A, RELE_ACTIVO);

   }else if((activado_reles_flag_frio==false) && (activado_reles_flag_calor==true)){

         digitalWrite(rele_A, RELE_APAGADO);
         digitalWrite(rele_B, RELE_ACTIVO);

   }else if((activado_reles_flag_frio==false) && (activado_reles_flag_calor==false)){

         digitalWrite(rele_A, RELE_APAGADO);
         digitalWrite(rele_B, RELE_APAGADO);

   }
}
```

Funciones adicionales:

```
////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION COMIENZA RETARDO
// funcion que la usaremos para pasar de forma generica un valor de retardo a una variable contador
////////////////////////////////////////////////////////////////////////////////////////////////////////
void comienza_retardo(int cuantos_segundos, int *mi_variable){
    *mi_variable = cuantos_segundos;
}


////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION PINTAR CONTADOR
// funcion para mostrar en el LCD el valor restante de horas del modo que este iniciado
////////////////////////////////////////////////////////////////////////////////////////////////////////
void pintar_contador(){
            sprintf(VARI,"%03d",contador_iniciado_horas_modo);
            lcd.setCursor(13, 0);
            lcd.print(VARI);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////
//FUNCION PINTAR ESPACIO VACIO CONTADOR
// funcion que la usaremos para borrar del LCD el valor del contador de horas del modo que estuviese activo
////////////////////////////////////////////////////////////////////////////////////////////////////////
void pintar_espacio_vacio_contador(){
            lcd.setCursor(12, 0);
            lcd.print(F("    "));
}
```

Código empleado en Octave

2.5 Modelado de nuestro sistema de control Fuzzy para el invernadero

2.5.5. Tratamiento de datos generados con Octave

Comandos empleados con Octave para generar nuestras gráficas de los intervalos fuzzy en 3D:

```
load 'my_fuzzy_file_intervalo_1.txt'
load 'my_fuzzy_file_intervalo_2.txt'
load 'my_fuzzy_file_intervalo_3.txt'
delta_humedad = linspace (-20, 20, 400);
delta_temperatura = linspace (-10, 10, 200);
mesh(delta_humedad, delta_temperatura, caudal_aire_1)
mesh(delta_humedad, delta_temperatura, caudal_aire_2)
mesh(delta_humedad, delta_temperatura, caudal_aire_3)
xlabel("Delta de Humedad (%)")
ylabel("Delta de Temperatura (ºC)")
zlabel("Caudal de Aire en segundos")
```

2.8 Registro con Raspberry o PC externo de los datos experimentales de cultivos de seta realizados, y el posterior tratamiento con Octave para su estudio teórico

Para sacar graficas en 2D de la evolución temporal de T y H de los datos experimentales de cultivo de setas:

```
t=[0:10:120687*10];

plot(t,arduino(:,1), t,arduino(:,2))

title("Grafica de Temperatura y Humedad - MODO GROW")
xlabel("Tiempo en Segundos.")
ylabel("Temperatura (ºC) y Humedad(%)")
grid


t_media=linspace(mean(arduino(:,1)), mean(arduino(:,1)),120688);
h_media=linspace(mean(arduino(:,2)), mean(arduino(:,2)),120688);
plot(t,arduino(:,1), t,arduino(:,2), t,t_media, t,h_media)
title("Grafica de Temperatura y Humedad - MODO GROW")
ylabel("Temperatura (ºC) y Humedad(%)")
xlabel("Tiempo en Segundos.")

disp('Valor medio de la temperatura');
mean(arduino(:,1))
disp('Valor medio de la Humedad');
mean(arduino(:,2))


figure

t_fig_2=[0:10:((3038-2754))*10];
f_aprox_creciente=(0-6.8)*exp(-t_fig_2/630)+6.8;
plot(t_fig_2,arduino([2754:3038],1)-19, t_fig_2, f_aprox_creciente)

title("Grafica de un ciclo de Temperatura - MODO GROW")
xlabel("Tiempo en Segundos.")
ylabel("Temperatura (ºC)")
grid


figure

t_fig_3=[0:10:((3038-2754))*10];
f_aprox_creciente=(0-6.8)*exp(-t_fig_3/630)+6.8;
f_aprox_decreciente=(5+0.4)*exp(-(t_fig_3-863)/630)-0.4;

plot(t_fig_3,arduino([2754:3038],1)-19+19, t_fig_3, f_aprox_creciente+19, t_fig_3, f_aprox_decreciente+19)
axis([0 ((3038-2754)-1)*10 18 25])
title("Grafica de un ciclo de Temperatura - MODO GROW")
xlabel("Tiempo en Segundos.")
ylabel("Temperatura (ºC)")


figure

t_fig_4=[0:10:((23510-22939))*10];
f_aprox_creciente_fuz=(0-6.8)*exp(-t_fig_4/630)+6.8;
f_aprox_decreciente_fuz=(5+0.4)*exp(-(t_fig_4-988)/630)-0.4;

plot(t_fig_4,arduino([22939:23510],1)-19+19, t_fig_4, f_aprox_creciente_fuz+19, t_fig_4,
f_aprox_decreciente_fuz+19)
```

```
axis([0 ((23510-22939)-1)*10 18 25])
title("Grafica de un ciclo de Temperatura - MODO GROW")
xlabel("Tiempo en Segundos.")
ylabel("Temperatura (ºC)")


figure(1)
coeff = ones(1, 1000)/1000; %filtro FIR 1000 muestras
avg1000Temp = filter(coeff, 1, arduino(:,1));
avg1000Humi = filter(coeff, 1, arduino(:,2));
hold on;
plot(t, avg1000Temp, 'g', t, avg1000Humi, 'c');
grid on;


figure(1)
coeff = ones(1, 20000)/20000; %filtro FIR 20000 muestras
avg1000Temp = filter(coeff, 1, arduino(:,1));
avg1000Humi = filter(coeff, 1, arduino(:,2));
hold on;
plot(t, avg1000Temp, 'g', t, avg1000Humi, 'c');
grid on;
```

2. Datasheet componentes

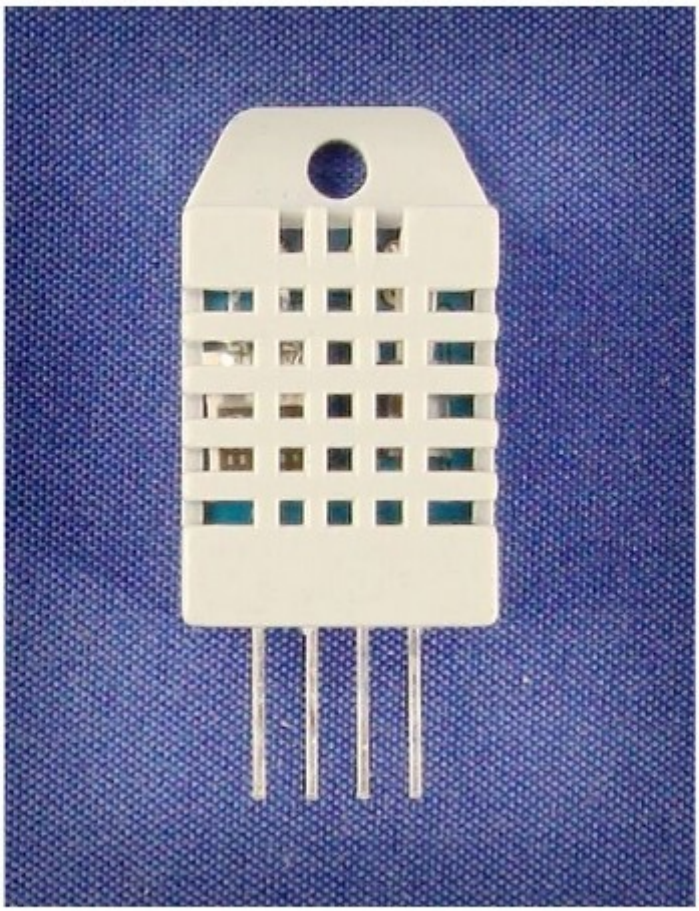
Sensor DHT22

Aosong Electronics Co.,Ltd

Your specialist in innovating humidity & temperature sensors

Digital-output relative humidity & temperature sensor/module

DHT22 (DHT22 also named as AM2302)



Capacitive-type humidity and temperature module/sensor

Aosong Electronics Co.,Ltd

Your specialist in innovating humidity & temperature sensors

1. Feature & Application:

* Full range temperature compensated * Relative humidity and temperature measurement
* Calibrated digital signal *Outstanding long-term stability *Extra components not needed
* Long transmission distance * Low power consumption *4 pins packaged and fully interchangeable

2. Description:

DHT22 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability.Its sensing elements is connected with 8-bit single-chip computer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

Small size & low consumption & long transmission distance(20m) enable DHT22 to be suited in all kinds of harsh application occasions.

Single-row packaged with four pins, making the connection very convenient.

3. Technical Specification:

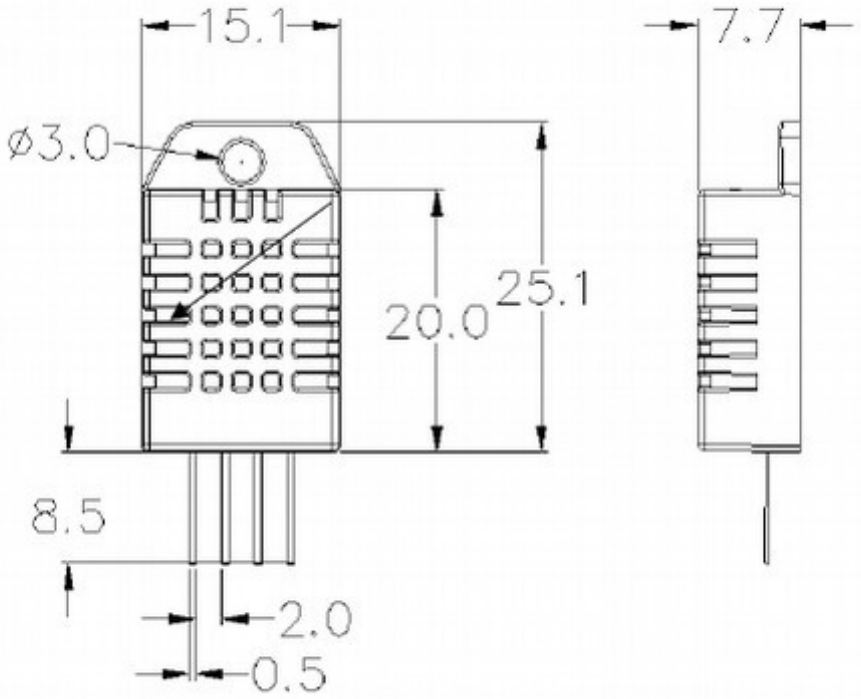
Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

4. Dimensions: (unit----mm)

1) Small size dimensions: (unit----mm)

Aosong Electronics Co.,Ltd

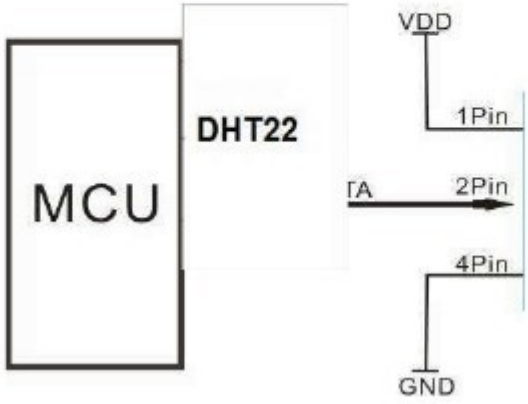
Your specialist in innovating humidity & temperature sensors



Pin sequence number: 1 2 3 4 (from left to right direction).

Pin	Function
1	VDD----power supply
2	DATA--signal
3	NULL
4	GND

5. Electrical connection diagram:



3Pin---NC, AM2302 is another name for DHT22

6. Operating specifications:

(1) Power and Pins

Power's voltage should be 3.3-6V DC. When power is supplied to sensor, don't send any instruction to the sensor within one second to pass unstable status. One capacitor valued 100nF can be added between VDD and GND for wave filtering.

(2) Communication and signal

Single-bus data is used for communication between MCU and DHT22, it costs 5mS for single time communication.

Data is comprised of integral and decimal part, the following is the formula for data.

DHT22 send out higher data bit firstly!

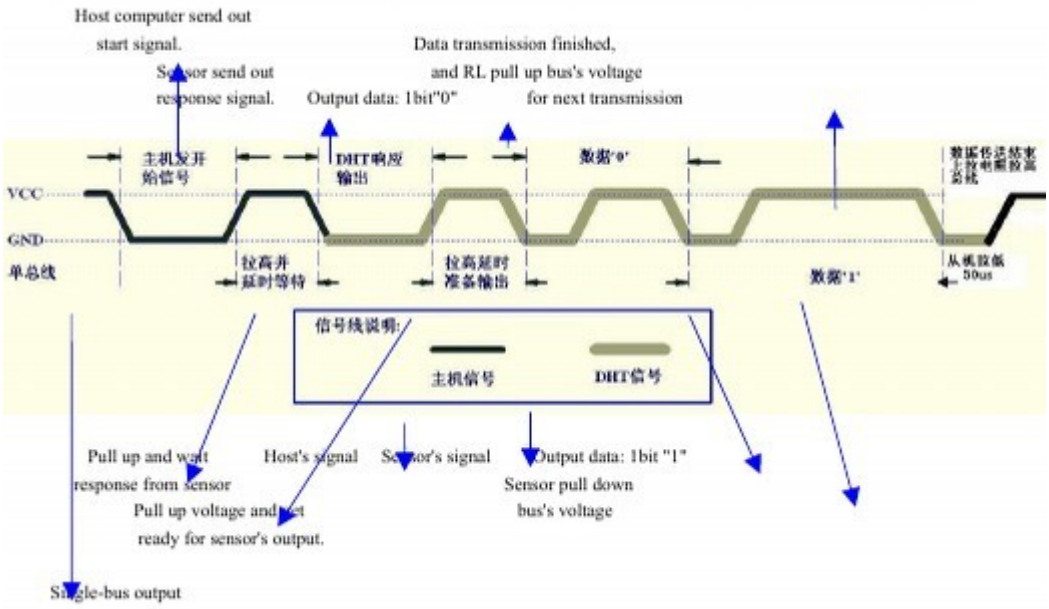
DATA=8 bit integral RH data+8 bit decimal RH data+8 bit integral T data+8 bit decimal T data+8 bit check-sum

If the data transmission is right, check-sum should be the last 8 bit of "8 bit integral RH data+8 bit decimal RH data+8 bit integral T data+8 bit decimal T data".

When MCU send start signal, DHT22 change from low-power-consumption-mode to running-mode. When MCU finishs sending the start signal, DHT22 will send response signal of 40-bit data that reflect the relative humidity

and temperature information to MCU. Without start signal from MCU, DHT22 will not give response signal to MCU. One start signal for one time's response data that reflect the relative humidity and temperature information from DHT22. DHT22 will change to low-power-consumption-mode when data collecting finish if it don't receive start signal from MCU again.

1) Check bellow picture for overall communication process:



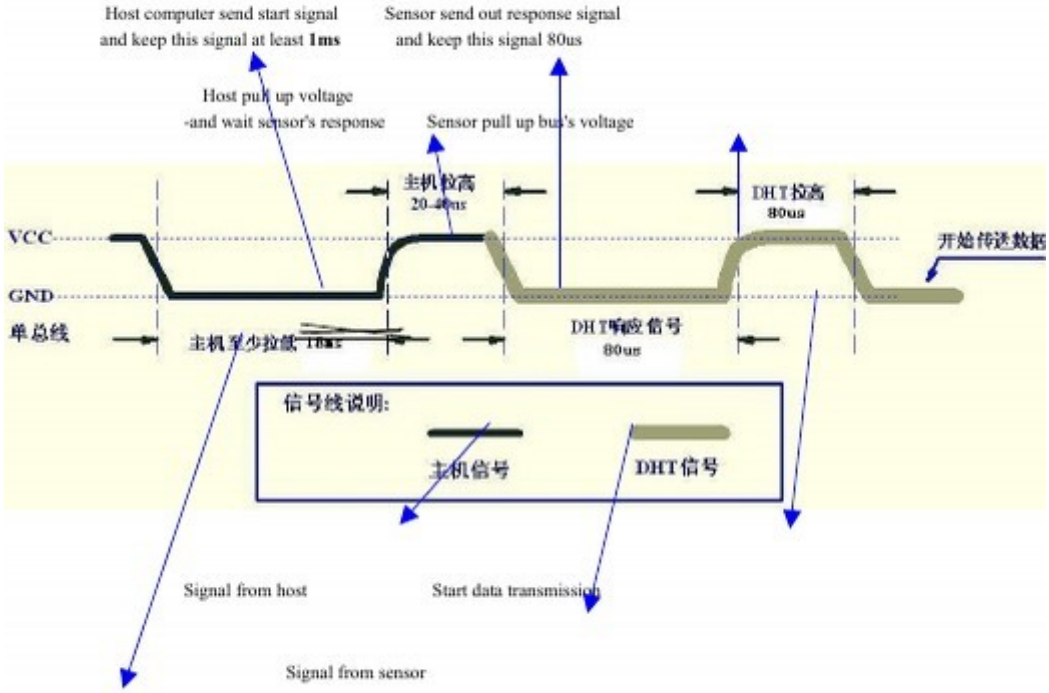
2) Step 1: MCU send out start signal to DHT22

Data-bus's free status is high voltage level. When communication between MCU and DHT22 begin, program of MCU will transform data-bus's voltage level from high to low level and this process must beyond at least 1ms to ensure DHT22 could detect MCU's signal, then MCU will wait 20-40us for DHT22's response.

Check bellow picture for step 1:

Aosong Electronics Co.,Ltd

Your specialist in innovating humidity & temperature sensors



Single-bus signal

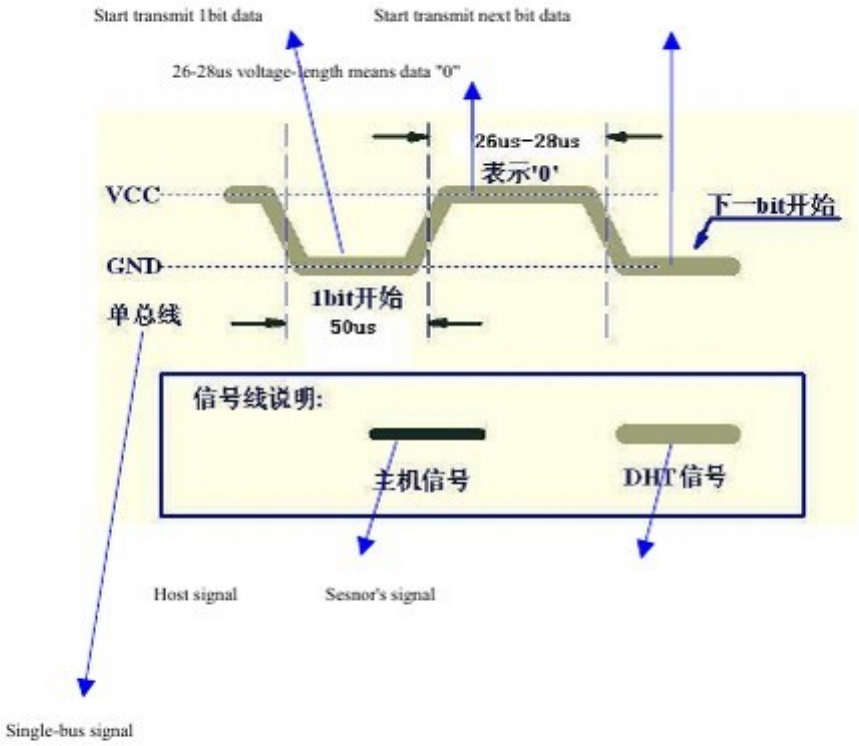
Step 2: DHT22 send response signal to MCU

When DHT22 detect the start signal, DHT22 will send out low-voltage-level signal and this signal last 80us as response signal, then program of DHT22 transform data-bus's voltage level from low to high level and last 80us for DHT22's preparation to send data.

Check bellow picture for step 2:

Aosong Electronics Co.,Ltd

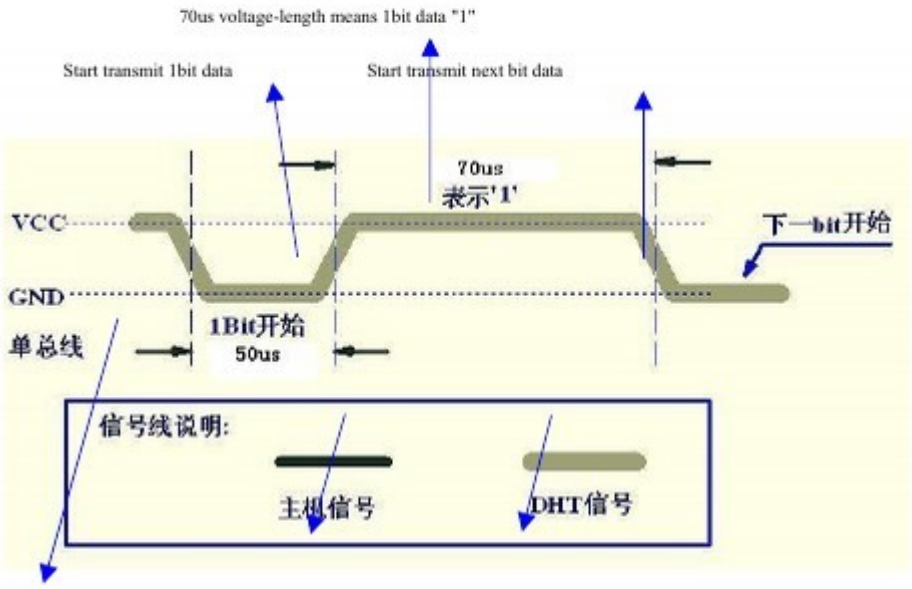
Your specialist in innovating humidity & temperature sensors



Step 3: DHT22 send data to MCU

When DHT22 is sending data to MCU, every bit's transmission begin with low-voltage-level that last 50us, the following high-voltage-level signal's length decide the bit is "1" or "0".

Check bellow picture for step 3:



Host signal Sesnor's signal

Single-bus signal

If signal from DHT22 is always high-voltage-level, it means DHT22 is not working properly, please check the electrical connection status.

7. Electrical Characteristics:

Item	Condition	Min	Typical	Max	Unit
Power supply	DC	3.3	5	6	V
Current supply	Measuring	1		1.5	mA
	Stand-by	40	Null	50	uA
Collecting period	Second		2		Second

*Collecting period should be : >2 second.

8. Attentions of application:

(1) Operating and storage conditions

We don't recommend the applying RH-range beyond the range stated in this specification. The DHT22 sensor can recover after working in non-normal operating condition to calibrated status, but will accelerate sensors' aging.

(2) Attentions to chemical materials

Vapor from chemical materials may interfere DHT22's sensitive-elements and debase DHT22's sensitivity.

(3) Disposal when (1) & (2) happens

Step one: Keep the DHT22 sensor at condition of Temperature 50~60Celsius, humidity <10%RH for 2 hours;

Step two: After step one, keep the DHT22 sensor at condition of Temperature 20~30Celsius, humidity >70%RH for 5 hours.

(4) Attention to temperature's affection

Relative humidity strongly depend on temperature, that is why we use temperature compensation technology to ensure accurate measurement of RH. But it's still be much better to keep the sensor at same temperature when sensing.

DHT22 should be mounted at the place as far as possible from parts that may cause change to temperature.

(5) Attentions to light

Long time exposure to strong light and ultraviolet may debase DHT22's performance.

(6) Attentions to connection wires

The connection wires' quality will effect communication's quality and distance, high quality shielding-wire is recommended.

(7) Other attentions

* Welding temperature should be bellow 260Celsius.
* Avoid using the sensor under dew condition.
* Don't use this product in safety or emergency stop devices or any other occasion that failure of DHT22 may cause personal injury.

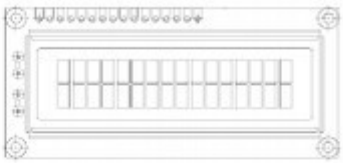
Display LCD

VISHAY

LCD-016M002B

Vishay

16 x 2 Character LCD



FEATURES

- 5 x 8 dots with cursor
- Built-in controller (KS 0066 or Equivalent)
- + 5V power supply (Also available for + 3V)
- 1/16 duty cycle
- B/L to be driven by pin 1, pin 2 or pin 15, pin 16 or A,K (LED)
- N.V. optional for + 3V power supply

MECHANICAL DATA

ITEM	STANDARD VALUE	UNIT
Module Dimension	80.0 x 36.0	mm
Viewing Area	66.0 x 16.0	mm
Dot Size	0.56 x 0.66	mm
Character Size	2.96 x 5.56	mm

ABSOLUTE MAXIMUM RATING

ITEM	SYMBOL	STANDARD VALUE			UNIT
		MIN.	TYP.	MAX.	
Power Supply	VDD-VSS	- 0.3	–	7.0	V
Input Voltage	VI	- 0.3	–	VDD	V

NOTE: VSS = 0 Volt, VDD = 5.0 Volt

ELECTRICAL SPECIFICATIONS

ITEM	SYMBOL	CONDITION		STANDARD VALUE			UNIT
				MIN.	TYP.	MAX.	
Input Voltage	VDD	VDD = + 5V		4.7	5.0	5.3	V
		VDD = + 3V		2.7	3.0	5.3	V
Supply Current	IDD	VDD = 5V		–	1.2	3.0	mA
Recommended LC Driving Voltage for Normal Temp. Version Module	VDD - V0	- 20 °C		–	–	–	V
		0°C		4.2	4.8	5.1	
		25°C		3.8	4.2	4.6	
		50°C		3.6	4.0	4.4	
		70°C		–	–	–	
LED Forward Voltage	VF	25°C		–	4.2	4.6	V
LED Forward Current	IF	25°C	Array	–	130	260	mA
			Edge	–	20	40	
EL Power Supply Current	IEL	Vel = 110VAC:400Hz		–	–	5.0	mA

DISPLAY CHARACTER ADDRESS CODE:

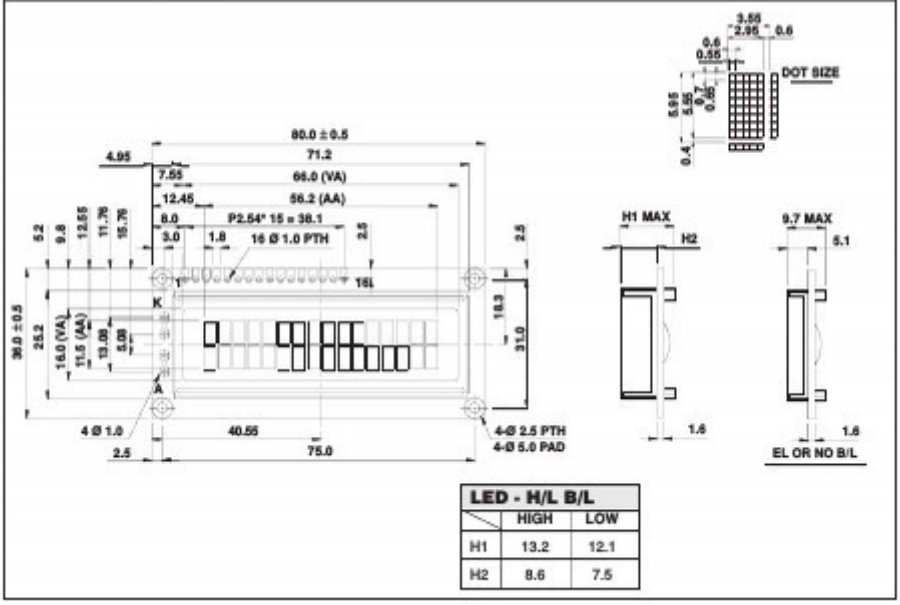
Display Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DD RAM Address	00	01														0F
DD RAM Address	40	41														4F

LCD-016M002B

Vishay 16 x 2 Character LCD

PIN NUMBER	SYMBOL	FUNCTION
1	Vss	GND
2	Vdd	+ 3V or + 5V
3	Vo	Contrast Adjustment
4	RS	H/L Register Select Signal
5	R/$\overline{W}$	H/L Read/Write Signal
6	E	H →L Enable Signal
7	DB0	H/L Data Bus Line
8	DB1	H/L Data Bus Line
9	DB2	H/L Data Bus Line
10	DB3	H/L Data Bus Line
11	DB4	H/L Data Bus Line
12	DB5	H/L Data Bus Line
13	DB6	H/L Data Bus Line
14	DB7	H/L Data Bus Line
15	A/Vee	+ 4.2V for LED/Negative Voltage Output
16	K	Power Supply for B/L (0V)

DIMENSIONS in millimeters



LED - H/L B/L		
	HIGH	LOW
H1	13.2	12.1
H2	8.6	7.5

Placa 4 Relés Optoacoplados Arduino

La salidas de las placas Arduino son perfectamente útiles para controlar cargas que no consuman demasiada corriente, como un Led, pero son insuficientes para cargas mayores. ¿Cómo hacemos para controlar por ejemplo una lámpara o un motor que se alimentan de 220 voltios? Una forma es emplear un módulo de relés como el que analizo en este artículo.

Antes de comenzar con el análisis: como sucede con muchos módulos para Arduino que se consiguen en el mercado, la placa que utilicé no tiene una marca que identifique a su fabricante ni información "oficial" sobre su funcionamiento. Seguramente provenga de alguna factoría china que produce estos productos que luego son vendidos por numerosas tiendas a través de la red y del cual pueden existir distintas variedades, con diferentes características. Antes de conectarlo según las indicaciones que se dan a continuación, por favor, asegúrense de que el módulo que tienen entre sus manos sea igual al que se describe aquí.

Hecha esta salvedad, comencemos con una descripción general de la placa. Se trata de un módulo de 4 relés (o relays) que funcionan a 5 Voltios, capaces de manejar cargas de hasta 10 Amperes en 250 Voltios, convenientemente aislados mediante optoacopladores de las entradas, las que cuentan con leds individuales que sirven como indicadores de estado.

Los distintos componentes del módulo pueden verse en la siguiente imagen:

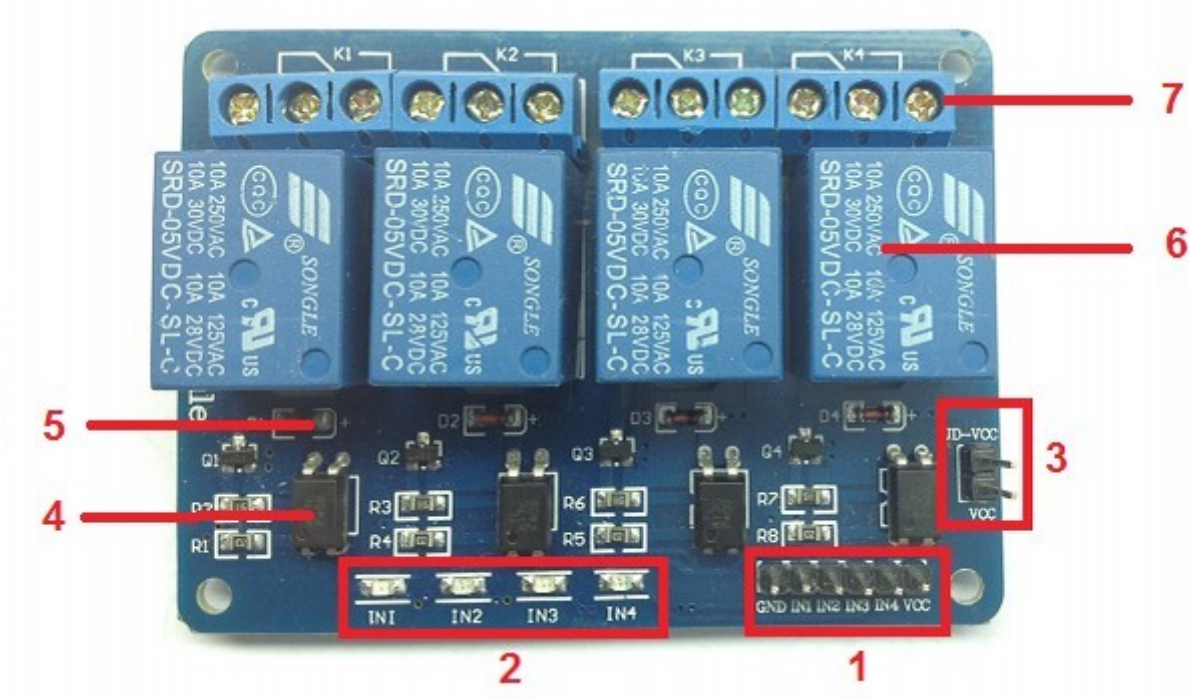


Fig. 1. Partes del módulo

Como se puede apreciar, la placa tiene un conector de entradas (IN1 a IN4) y alimentación (GND es masa o negativo y Vcc es el positivo) [1], cuatro leds que indican el estado de la entradas [2], un jumper selector para la alimentación de los relés [3], cuatro optoacopladores del tipo FL817C [4],

cuatro diodos de protección [5], cuatro relés marca SONGLE con bobinas de 5V y contactos capaces de controlar hasta 10 Amperes en una tensión de 250V [6] y cuatro borneras, con tres contactos cada una (Común, Normal abierto y Normal cerrado), para las salidas de los relés [7].

Esquemático

En la imagen de mas abajo se puede apreciar el circuito esquemático de un canal, el resto de los canales repite la misma configuración.

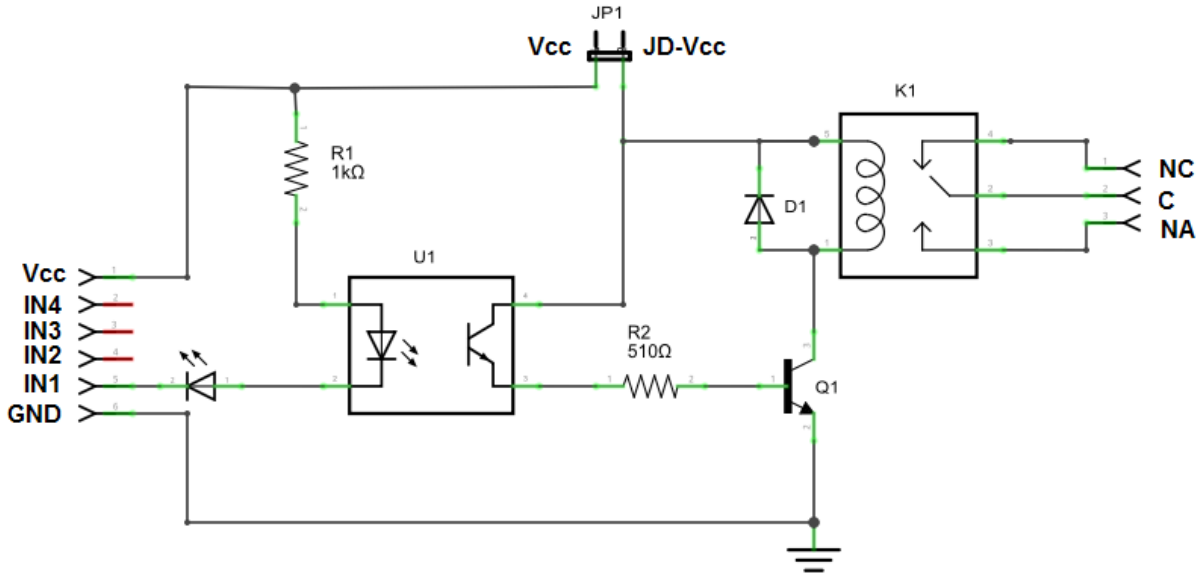


Fig. 2. Circuito esquemático

Funcionamiento

A partir del circuito de la Fig. 2 analicemos el funcionamiento del circuito: la entrada IN1 está conectada al cátodo del diodo del optoacoplador a través del led indicador. El ánodo del diodo del optoacoplador se conecta a Vcc (positivo) por intermedio de R1, una resistencia de 1000 ohms. Estos tres componentes, el diodo indicador, el diodo del opto y la R1 forman un circuito serie por el cual circula la corriente cuando la entrada está a un nivel BAJO (conectada a GND) y no circula si la entrada está a un nivel ALTO (conectada a Vcc).

El transistor del opto tiene su colector a JD-Vcc y su emisor conectado a Q1 a través de una resistencia de 510 ohms. Este es otro circuito serie por el cual circula corriente cuando el transistor del opto conduce al ser “iluminado” por su diodo, con lo que se introduce corriente en la base de Q1 a través de R2.

Finalmente, Q1 está conectado en una típica configuración emisor común, con su emisor a masa (GND) y la bobina del relé como carga en el colector. Cuando circula corriente por la base desde el opto, Q1 se satura permitiendo el paso de la corriente a través de la bobina del relé, lo que produce que se cierren los contactos del mismo (común con normal abierto). El diodo D1 protege al transistor de la tensión que aparece en la bobina del relé cuando deja de circular corriente por la misma.

En síntesis, al ponerse la entrada **a nivel BAJO** se pone a la saturación el transistor Q1 a través del

optoacoplador con lo que se cierra el contacto normal abierto del relé.

Alimentación y consumo

La forma mas sencilla de alimentar este módulo es desde Vcc y GND de la placa Arduino, manteniendo el Jumper en su lugar, con lo que JD-Vcc = Vcc. Esta conexión tiene dos limitaciones importantes:

- Se pierde la aislación eléctrica que brindan los optoacopladores, lo que aumenta la posibilidad de daño al Arduino si hay algún problema con las cargas de los relés.
- La corriente consumida por las bobinas de los relés debe ser provista por la placa Arduino. Cada bobina consume unos 90 mA y las cuatro juntas suman 360 mA. Si a esto le sumamos los consumos que pueden tener otras salidas, estamos muy cerca de los 500 mA que puede suministrar un puerto USB. En este caso se debería alimentar al Arduino con una fuente externa, lo que aumenta el limite de corriente a 1 A (en el caso de la Arduino UNO).

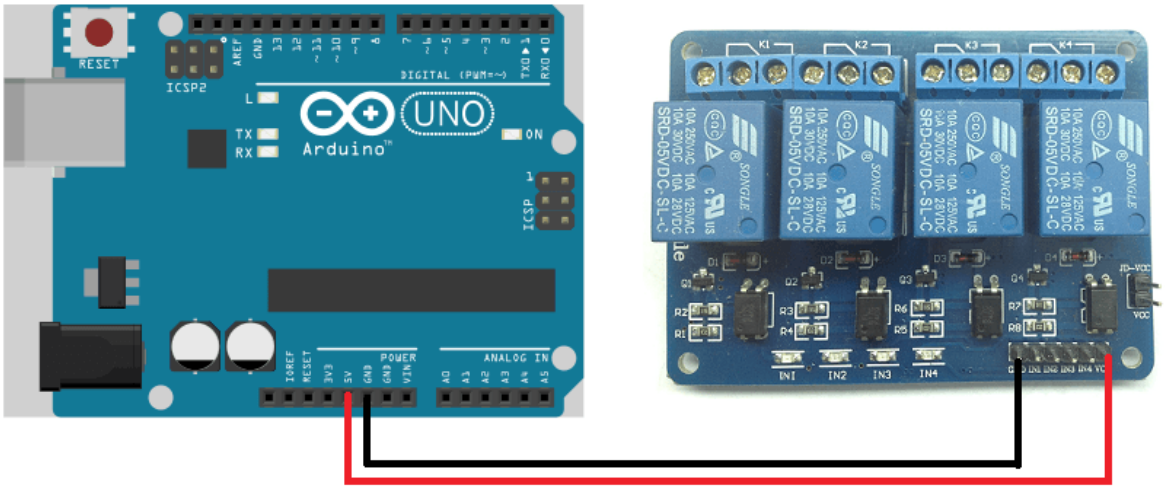


Fig. 3. Alimentación con una sola fuente

La forma mas segura es remover el jumper y alimentar la placa de relés con dos fuentes: la de la placa Arduino conectada a Vcc y una segunda fuente, con el positivo a JD-Vcc y el negativo a GND, sin estar éste unido a la placa Arduino. Esta conexión tiene como ventajas:

- Hay completa aislación entre la carga y el Arduino.
- Todo el consumo de los relés es tomado de la segunda fuente y no del Arduino o el puerto USB.

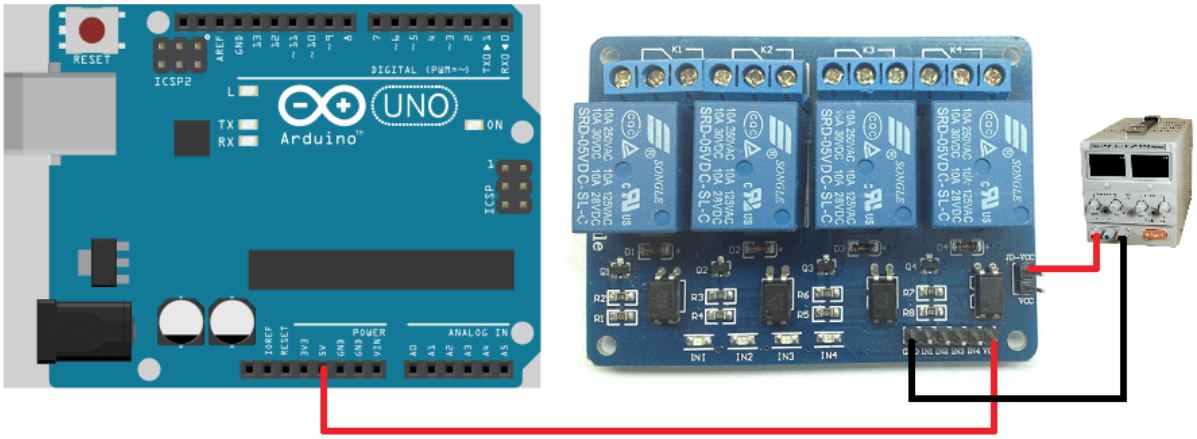


Fig. 4. Alimentación con dos fuentes

Entradas

Las entradas a la placa puede conectarse directamente a las salidas digitales de la placa Arduino. La única precaución a tener en cuenta es que cuando Arduino arranca al ser alimentado, los pines se configuran como entradas automáticamente y puede ocurrir que, por un brevísimo lapso de tiempo entre el arranque y la correcta configuración de estos pines como salidas, las entradas de control al módulo de relé queden en un estado indeterminado. Esto se puede evitar conectando en cada entrada un pull-up con una resistencia de 10K a Vcc, lo que asegura un estado ALTO durante el arranque.

Software

Este programa de ejemplo controla los 4 relés conectando las entradas IN1, IN2, IN3 e IN4 del módulo a los pines digitales 2, 3, 4 y 5 de una placa Arduino UNO. La alimentación se toma directamente desde el Arduino como en el primero de los casos vistos mas arriba.

```
//Definiciones
//Recordar que los reles se activan con nivel BAJO (0)

#define RELAY_ON 0
#define RELAY_OFF 1


void setup () {

//Inicialización

//Asegurar nivel ALTO en cada entrada de rele
digitalWrite (2, RELAY_OFF);
digitalWrite (3, RELAY_OFF);
digitalWrite (4, RELAY_OFF);
digitalWrite (5, RELAY_OFF);

//Definir los pines como salida
pinMode (2, OUTPUT);
pinMode (3, OUTPUT);
pinMode (4, OUTPUT);
```

```
pinMode (5, OUTPUT);

}

void loop () {

digitalWrite (2, RELAY_ON); //Activa relé 1
delay (2000);
digitalWrite (2, RELAY_OFF); //Desactiva relé 1
delay (2000);

}
```

Célula Peltier

Hebei I.T. (Shanghai) Co., Ltd.

Thermoelectric Cooler

TEC1-12706

Performance Specifications

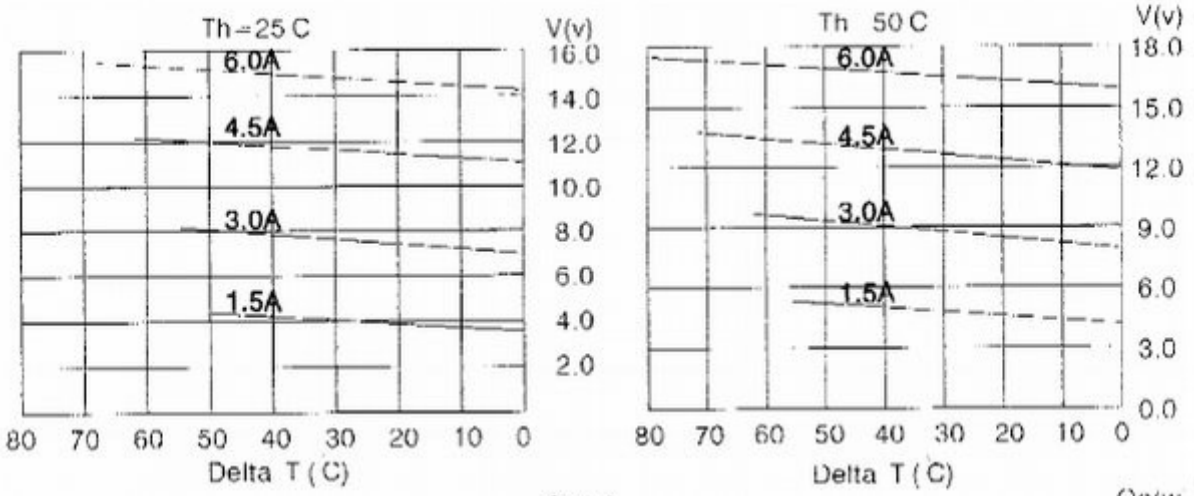
Hot Side Temperature (º C)	25º C	50º C
Qmax (Watts)	50	57
Delta Tmax (º C)	66	75
Imax (Amps)	6.4	6.4
Vmax (Volts)	14.4	16.4
Module Resistance (Ohms)	1.98	2.30

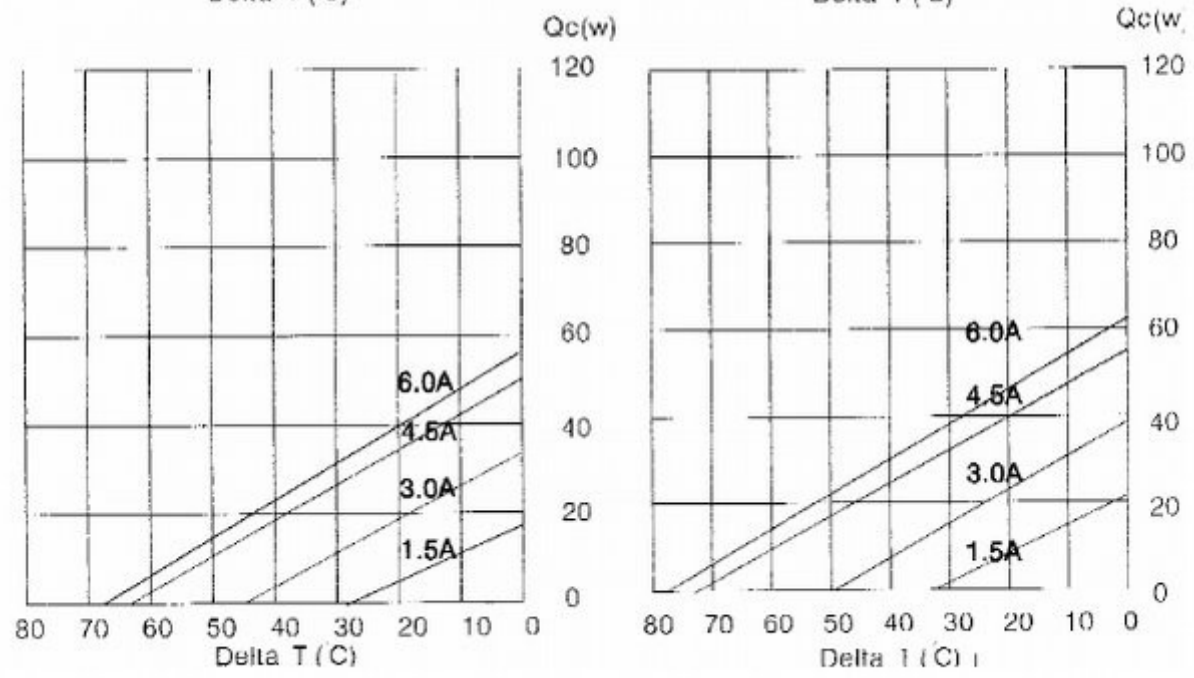
Hebei I.T. (Shanghai) Co., Ltd.

Thermoelectric Cooler

TEC1-12706

Performance curves:

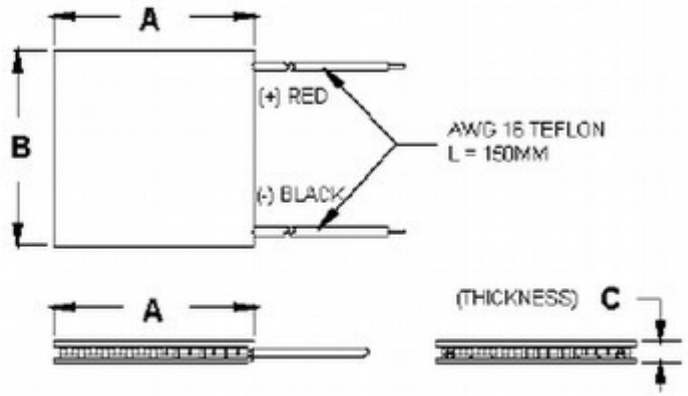




Hebei I.T. (Shanghai) Co., Ltd.

Thermoelectric Cooler

TEC1-12706



Ceramic Material: Alumina (Al_2O_3)
Solder Construction: 138° C, Bismuth Tin (BiSn)

Size table:

A	B	C			
40	40	3.9			

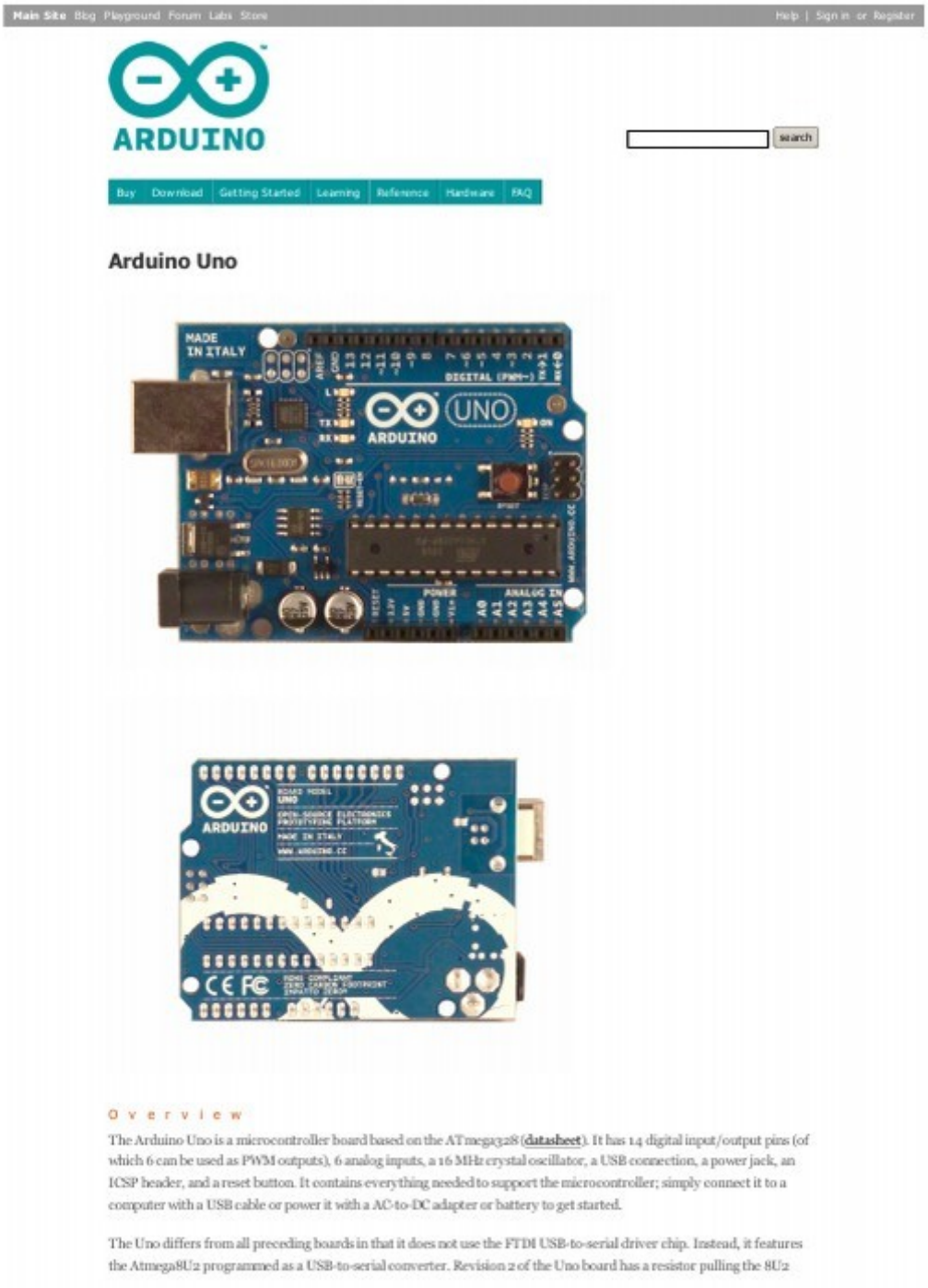
Operating Tips

- Max. Operating Temperature: 138°C
- Do not exceed Imax or Vmax when operating module.
- Life expectancy: 200,000 hours
- Please consult HB for moisture protection options (seeling).
- Failure rate based on long time testings: 0.2%.

Humidificador piezoeléctrico 107kHz

Especificaciones generales del humidificador:	
Material	ABS + PP
Método	Transductor Piezoeléctrico 107kHz
Potencia	2W
Humidification	10-35ml / h
Dimensión	13.5 x 7 cm

Arduino Uno


Main Site Blog Playground Forum Labs Store
Help | Sign in or Register
ARDUINO
search
Buy Download Getting Started Learning Reference Hardware FAQ
Arduino Uno
MADE IN ITALY
UNO
ARDUINO
POWER
ANALOG IN
Overview
The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. Revision 2 of the Uno board has a resistor pulling the 8U2

HWB line to ground, making it easier to put into DFU mode.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Schematic & Reference Design

EAGLE files: arduino-uno-reference-design.zip

Schematic: arduino-uno-schematic.pdf

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the **attachInterrupt()** function for details.

- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the **analogWrite()** function.

- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the **SPI library**.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the **analogReference()** function. Additionally, some pins have specialized functionality:

- **TWI: A4 (SDA) and A5 (SCL).** Support TWI communication using the **Wire library**.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with **analogReference()**.

- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the **mapping between Arduino pins and ATmega328 ports**[2].

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, **on Windows, a .inf file is required.** The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A **SoftwareSerial library** allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the **documentation** for details. For SPI communication, use the **SPI library**.

Programming

The Arduino Uno can be programmed with the Arduino software (**download**). Select "Arduino Uno from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the **reference** and **tutorials**.

The ATmega328 on the Arduino Uno comes preburned with a **bootloader** that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (**reference**, **C header files**).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see **these instructions** for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use **Atmel's FLIP software** (Windows) or the **DFU programmer** (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See **this user-contributed tutorial** for more information.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see **this forum thread** for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

Bomba de aire 12V

Tensión nominal	DC 6.0 V	DC 12.0 V	DC 24.0 V
Corriente nominal	≪600mA	≪450mA	≪350mA
Flujo de aire	≫ 1.2LPM	≫ 1.2LPM	≫ 1.2LPM
Presión máxima	≫ 60Kpa		
Presión de vacío	≪-30Kpa		
Ruido	≪63bB		
Tubería	Φ 4.2mm		
Dimensión	Φ 27*71mm		
Tiempo de la vida	50000 veces (10 s en 7 s off)		

Módulo descender DC/DC Buck 30V-3V

Descripción del producto

Este circuito te permite tener un voltaje regulado a partir de una fuente de alimentación con un voltaje mayor, por ejemplo si tienes una fuente de 12V puedes regularlos a 5V, 3.3V, 2.2V, etc, para el uso con microcontroladores, Arduino, PICs, Raspberry Pi, fuentes variables, drivers para leds, etc.

Este módulo esta basado en el Regulador DC-DC Step Down LM2596 que es un circuito integrado monolítico adecuado para el diseño fácil y conveniente de una fuente de conmutación tipo buck. Es capaz de conducir una corriente de hasta 3A. Maneja una carga con excelente regulación de línea y bajo voltaje de rizado. Este dispositivo está disponible con voltaje de salida ajustable. El módulo reduce al mínimo el uso de componentes externos para simplificar el diseño de fuentes de alimentación.

El módulo convertidor LM2596 es una fuente de alimentación conmutada, así que su eficiencia es significativamente mayor en comparación con los populares reguladores lineales de tres terminales, especialmente con tensiones de entrada superiores.

Características

- Basada en el regulador LM2596, salida entre 1,5 y 35Vdc
- Voltaje de entrada: 4.5-40V
- Voltaje de salida: 1.5-35V (Adjustable)
- Corriente de salida: Maxima 3A
- Dimensiones: 43*20*14mm
- Frecuencia de switching: 150 KHz

NPN Darlington (ESM6045AV)

SGS-THOMSON MICROELECTRONICS

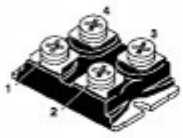
ESM6045DV

NPN DARLINGTON POWER MODULE

- HIGH CURRENT POWER BIPOLAR MODULE
- VERY LOW R_{th} JUNCTION CASE
- SPECIFIED ACCIDENTAL OVERLOAD AREAS
- ULTRAFAST FREEWHEELING DIODE
- ISOLATED CASE (2500V RMS)
- EASY TO MOUNT
- LOW INTERNAL PARASITIC INDUCTANCE

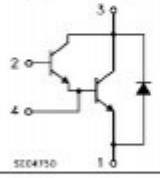
INDUSTRIAL APPLICATIONS:

- MOTOR CONTROL
- SMPS & UPS
- DC/DC & DC/AC CONVERTERS
- WELDING EQUIPMENT



ISOTOP

INTERNAL SCHEMATIC DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CEV}	Collector-Emitter Voltage (V_{BE} = -5 V)	600	V
$V_{CEO(sus)}$	Collector-Emitter Voltage (I_B = 0)	450	V
V_{EBO}	Emitter-Base Voltage (I_C = 0)	7	V
I_C	Collector Current	84	A
I_{CM}	Collector Peak Current (t_p = 10 ms)	126	A
I_B	Base Current	8	A
I_{BM}	Base Peak Current (t_p = 10 ms)	16	A
P_{tot}	Total Dissipation at T_c = 25 °C	250	W
T_{stg}	Storage Temperature	-55 to 150	°C
T_j	Max. Operating Junction Temperature	150	°C
V_{ISO}	Insulation Withstand Voltage (AC-RMS)	2500	°C

THERMAL DATA

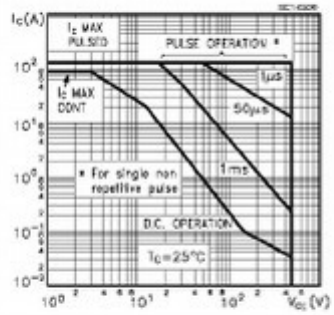
$R_{th(j-case)}$	Thermal Resistance Junction-case (transistor)	Max	0.5	°C/W
$R_{th(j-case)}$	Thermal Resistance Junction-case (diode)	Max	1.2	°C/W
R_{thc-h}	Thermal Resistance Case-heatsink With Conductive Grease Applied	Max	0.05	°C/W

ELECTRICAL CHARACTERISTICS (T_{case} = 25 °C unless otherwise specified)

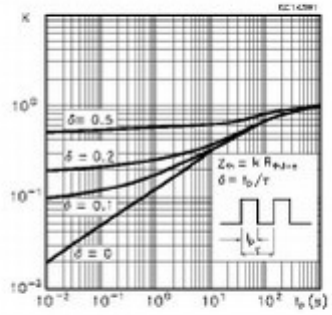
Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
I_{CER} #	Collector Cut-off Current (R_{BE} = 5 Ω)	V_{CE} = V_{CEV} V_{CE} = V_{CEV} T_j = 100 °C			1.5 22	mA mA
I_{CEV} #	Collector Cut-off Current (V_{BE} = -5)	V_{CE} = V_{CEV} V_{CE} = V_{CEV} T_j = 100 °C			1 15	mA mA
I_{EBO} #	Emitter Cut-off Current (I_C = 0)	V_{EB} = 5 V			1	mA
$V_{CEO(SUS)}$*	Collector-Emitter Sustaining Voltage	I_C = 0.2 A L = 25 mH V_{clamp} = 450 V	450			V
h_{FE}*	DC Current Gain	I_C = 70 A V_{CE} = 5 V		120		
$V_{CE(sat)}$*	Collector-Emitter Saturation Voltage	I_C = 50 A I_B = 1 A I_C = 50 A I_B = 1 A T_j = 100 °C I_C = 70 A I_B = 4 A I_C = 70 A I_B = 4 A T_j = 100 °C		1.2 1.6 1.35 1.7	 2 2	V V V V
$V_{BE(sat)}$*	Base-Emitter Saturation Voltage	I_C = 70 A I_B = 4 A I_C = 70 A I_B = 4 A T_j = 100 °C		2.3 2.4	 3	V V
dI_C/dt	Rate of Rise of On-state Collector	V_{CC} = 300 V R_C = 0 t_p = 3 µs I_{B1} = 1.5 A T_j = 100 °C	375	450		A/µs
V_{CE}(3 µs)•	Collector-Emitter Dynamic Voltage	V_{CC} = 300 V R_C = 6 Ω I_{B1} = 1.5 A T_j = 100 °C		6	9	V
V_{CE}(5 µs)•	Collector-Emitter Dynamic Voltage	V_{CC} = 300 V R_C = 6 Ω I_{B1} = 1.5 A T_j = 100 °C		3	4.5	V
t_s t_f t_c	Storage Time Fall Time Cross-over Time	I_C = 50 A V_{CC} = 50 V V_{BB} = -5 V R_{BB} = 0.3 Ω V_{clamp} = 450 V I_{B1} = 1 A L = 0.05 mH T_j = 100 °C		3.5 0.3 0.8	5.5 0.5 1.7	µs µs µs
V_{CEW}	Maximum Collector Emitter Voltage Without Snubber	I_{CWoff} = 84 A I_{B1} = 4 A V_{BB} = -5 V V_{CC} = 50 V L = 0.03 mH R_{BB} = 0.3 Ω T_j = 125 °C	450			V
V_F*	Diode Forward Voltage	I_F = 70 A T_j = 100 °C		1.6	1.9	V
I_{RM}	Reverse Recovery Current	V_{CC} = 200 V I_F = 70 A dI_F/dt = -375 A/µs L < 0.05 µH T_j = 100 °C		38	45	A

* Pulsed: Pulse duration = 300 µs, duty cycle 1.5 %
# See test circuits in databook introduction
To evaluate the conduction losses of the diode use the following equations:
V_F = 1.5 + 0.0055 I_F P = 1.5 $I_{F(AV)}$ + 0.0055 $I^2_{F(RMS)}$

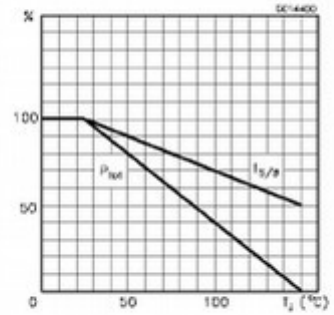
Safe Operating Areas



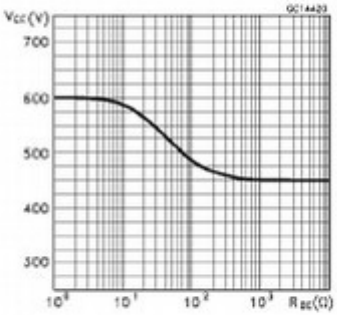
Thermal Impedance



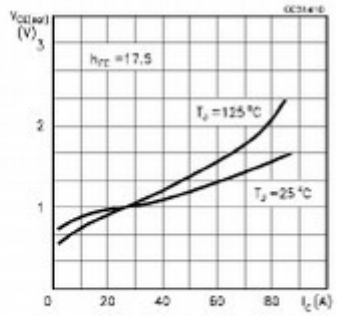
Derating Curve



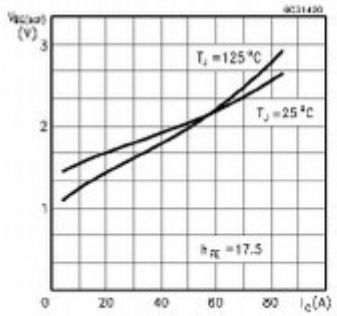
Collector-emitter Voltage Versus base-emitter Resistance



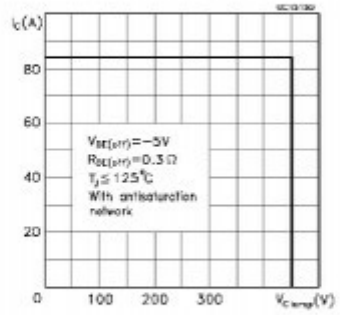
Collector Emitter Saturation Voltage



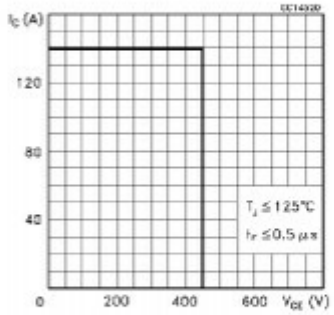
Base-Emitter Saturation Voltage



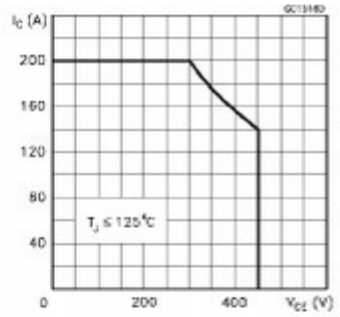
Reverse Biased SOA



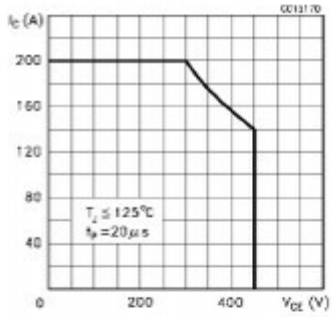
Foward Biased SOA



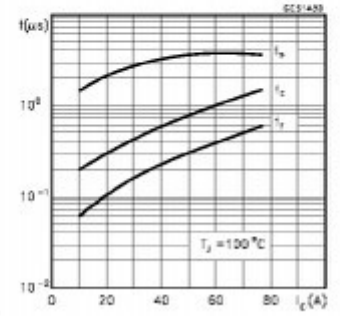
Reverse Biased AOA



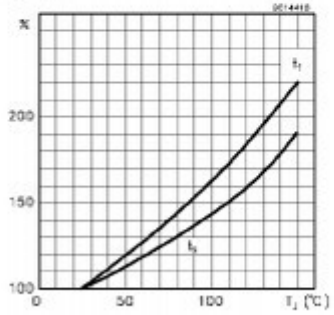
Forward Biased AOA



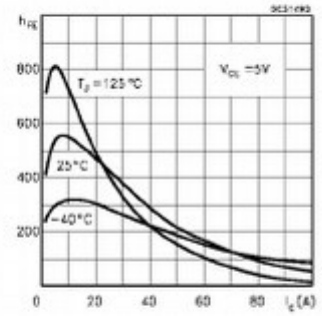
Switching Times Inductive Load



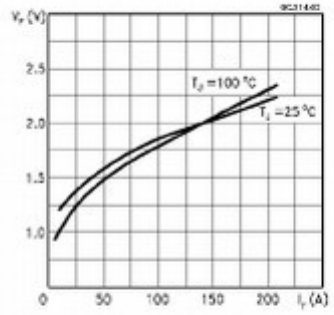
Switching Times Inductive Load Versus Temperature



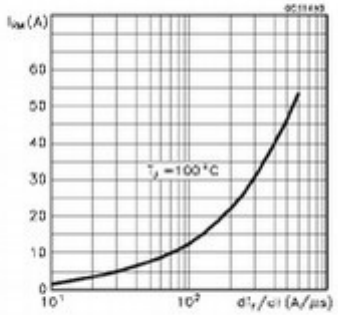
Dc Current Gain



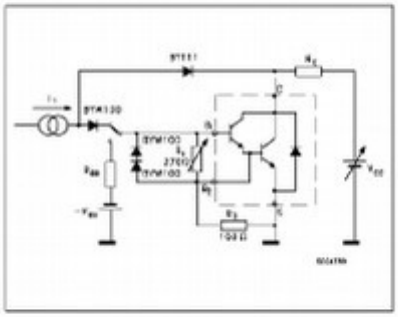
Typical V_F Versus I_F



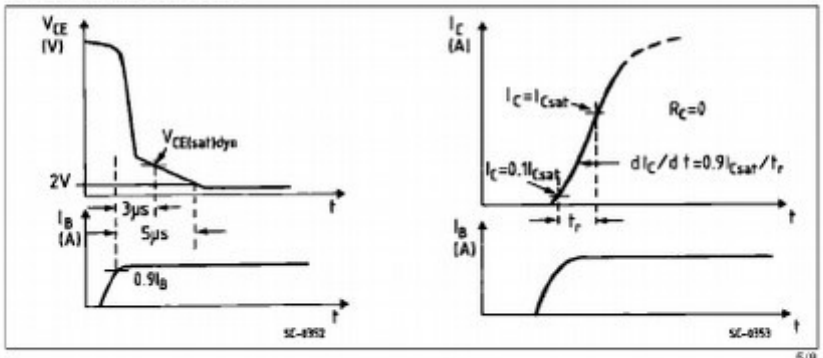
Peak Reverse Current Versus dI_F/dt



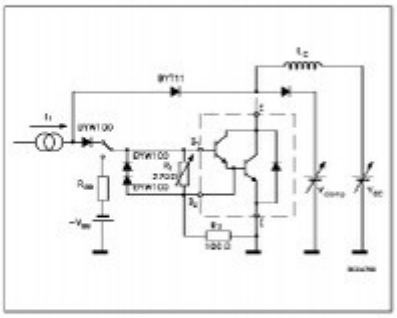
Turn-on Switching Test Circuit



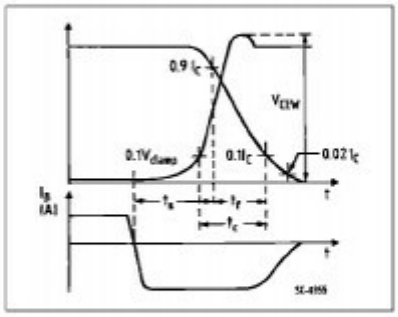
Turn-on Switching Waveforms



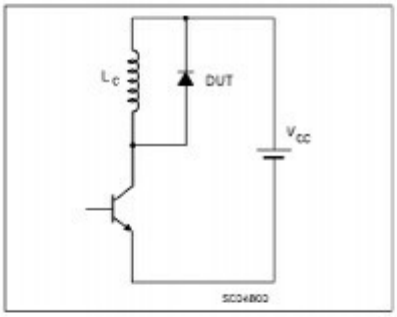
Turn-on Switching Test Circuit



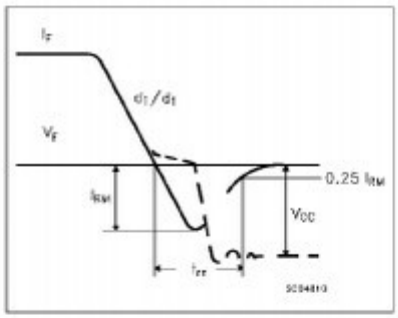
Turn-off Switching Waveforms



Turn-off Switching Test Circuit of Diode

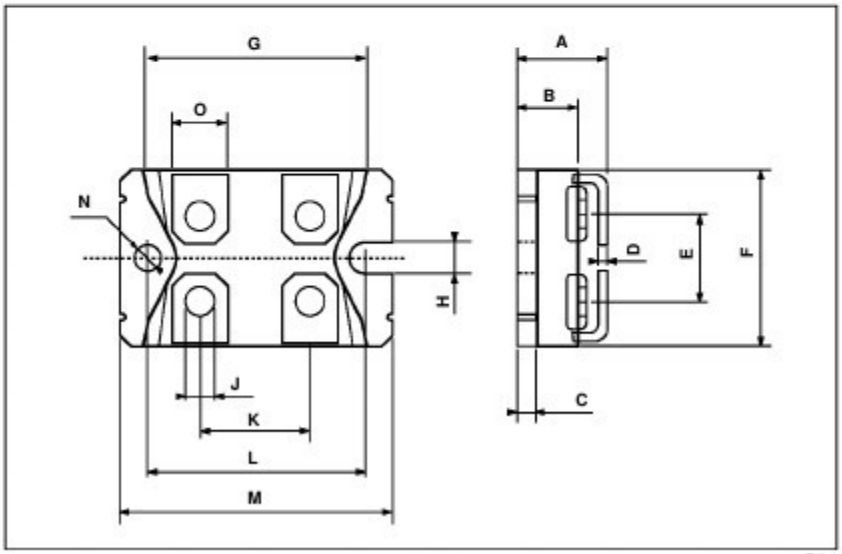


Turn-off Switching Waveform of Diode



ISOTOP MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A	11.8		12.2	0.466		0.480
B	8.9		9.1	0.350		0.358
C	1.95		2.05	0.076		0.080
D	0.75		0.85	0.029		0.033
E	12.6		12.8	0.496		0.503
F	25.15		25.5	0.990		1.003
G	31.5		31.7	1.240		1.248
H	4			0.157		
J	4.1		4.3	0.161		0.169
K	14.9		15.1	0.586		0.594
L	30.1		30.3	1.185		1.193
M	37.8		38.2	1.488		1.503
N	4			0.157		
O	7.8		8.2	0.307		0.322



MOSFET IRF510

FAIRCHILD SEMICONDUCTOR™

IRF510

Data Sheet *January 2002*

5.6A, 100V, 0.540 Ohm, N-Channel Power MOSFET

This N-Channel enhancement mode silicon gate power field effect transistor is an advanced power MOSFET designed, tested, and guaranteed to withstand a specified level of energy in the breakdown avalanche mode of operation. All of these power MOSFETs are designed for applications such as switching regulators, switching convertors, motor drivers, relay drivers, and drivers for high power bipolar switching transistors requiring high speed and low gate drive power. These types can be operated directly from integrated circuits.

Formerly developmental type TA17441.

Ordering Information

PART NUMBER	PACKAGE	BRAND
IRF510	TO-220AB	IRF510

NOTE: When ordering, include the entire part number.

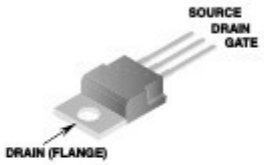
Features

- 5.6A, 100V
- $r_{DS(ON)} = 0.540\Omega$
- Single Pulse Avalanche Energy Rated
- SOA is Power Dissipation Limited
- Nanosecond Switching Speeds
- Linear Transfer Characteristics
- High Input Impedance
- Related Literature
  - TB334 "Guidelines for Soldering Surface Mount Components to PC Boards"

Symbol

Packaging

JEDEC TO-220AB



Absolute Maximum Ratings $T_C = 25^oC$, Unless Otherwise Specified

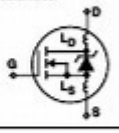
		IRF510	UNITS
Drain to Source Voltage (Note 1)	V_{DS}	100	V
Drain to Gate Voltage (R_{GS} = 20kΩ) (Note 1)	V_{DGR}	100	V
Continuous Drain Current	I_D	5.6	A
$T_C = 100^oC$	I_D	4	A
Pulsed Drain Current (Note 3)	I_{DM}	20	A
Gate to Source Voltage	V_{GS}	±20	V
Maximum Power Dissipation	P_D	43	W
Linear Derating Factor		0.29	W/^oC
Single Pulse Avalanche Energy Rating (Note 4)	E_{AS}	19	mJ
Operating and Storage Temperature Range	T_J, T_{STG}	-55 to 175	^oC
Maximum Temperature for Soldering			
Leads at 0.063in (1.6mm) from Case for 10s	T_L	300	^oC
Package Body for 10s, See Techbrief 334	T_{pkg}	260	^oC

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

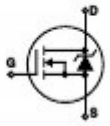
NOTE:

1. $T_J = 25^oC$ to 150^oC.

Electrical Specifications $T_C = 25^oC$, Unless Otherwise Specified

PARAMETER	SYMBOL	TEST CONDITIONS		MIN	TYP	MAX	UNITS
Drain to Source Breakdown Voltage	BV_{DSS}	V_{GS} = 0V, I_D = 250µA, (Figure 10)		100	-	-	V
Gate to Threshold Voltage	$V_{GS(TH)}$	$V_{GS} = V_{DS}$, I_D = 250µA		2.0	-	4.0	V
Zero-Gate Voltage Drain Current	I_{DSS}	V_{DS} = 95V, V_{GS} = 0V		-	-	25	µA
		V_{DS} = 0.8 x Rated BV_{DSS}, V_{GS} = 0V, $T_J = 150^oC$		-	-	250	µA
On-State Drain Current (Note 2)	$I_{D(ON)}$	$V_{DS} > I_{D(ON)} \times r_{DS(ON)MAX}$, V_{GS} = 10V (Figure 7)		5.6	-	-	A
Gate to Source Leakage Current	I_{GSS}	V_{GS} = ±20V		-	-	±100	nA
Drain to Source On Resistance (Note 2)	$r_{DS(ON)}$	V_{GS} = 10V, I_D = 3.4A (Figures 8, 9)		-	0.4	0.54	Ω
Forward Transconductance (Note 2)	g_{fs}	V_{GS} = 50V, I_D = 3.4A (Figure 12)		1.3	2.0	-	S
Turn-On Delay Time	$t_{d(ON)}$	I_D = 5.6A, R_{GS} = 24Ω, V_{DD} = 50V, R_L = 9Ω, V_{DD} = 50V, V_{GS} = 10V MOSFET switching times are essentially independent of operating temperature		-	8	12	ns
Rise Time	t_r			-	25	63	ns
Turn-Off Delay Time	$t_{d(OFF)}$			-	15	7	ns
Fall Time	t_f			-	12	59	ns
Total Gate Charge (Gate to Source + Gate to Drain)	$Q_{g(TOT)}$	V_{GS} = 10V, I_D = 5.6A, V_{DS} = 0.8 x Rated BV_{DSS}, $I_{G(REF)}$ = 1.5mA (Figure 14) Gate charge is essentially independent of operating temperature.		-	5.0	30	nC
Gate to Source Charge	Q_{gs}			-	2.0	-	nC
Gate to Drain "Miller" Charge	Q_{gd}			-	3.0	-	nC
Input Capacitance	C_{ISS}	V_{GS} = 0V, V_{DS} = 25V, f = 1.0MHz (Figure 11)		-	135	-	pF
Output Capacitance	C_{OSS}			-	80	-	pF
Reverse-Transfer Capacitance	C_{RSS}			-	20	-	pF
Internal Drain Inductance	L_D	Measured From the Contact Screw On Tab To Center of Die	Modified MOSFET Symbol Showing the Internal Devices Inductances	-	3.5	-	nH
		Measured From the Drain Lead, 6mm (0.25in) From Package to Center of Die		-	4.5	-	nH
Internal Source Inductance	L_S	Measured From The Source Lead, 6mm (0.25in) From Header to Source Bonding Pad		-	7.5	-	nH
Junction to Case	$R_{\theta JC}$			-	-	3.5	$^oC/W$
Junction to Ambient	$R_{\theta JA}$	Free air operation		-	-	80	$^oC/W$

Source to Drain Diode Specifications

PARAMETER	SYMBOL	Test Conditions		MIN	TYP	MAX	UNITS
Continuous Source to Drain Current	I_{SD}	Modified MOSFET Symbol Showing the Integral Reverse P-N Junction Diode		-	-	5.6	A
Pulse Source to Drain Current (Note 3)	I_{SDM}			-	-	20	A
Source to Drain Diode Voltage (Note 2)	V_{SD}	$T_J = 25^oC$, $I_{SD} = 5.6A$, $V_{GS} = 0V$ (Figure 13)		-	-	2.5	V
Reverse Recovery Time	t_{rr}	$T_J = 25^oC$, $I_{SD} = 5.6A$, $dI_{SD}/dt = 100A/\mu s$		4.6	96	200	ns
Reverse Recovered Charge	Q_{RR}	$T_J = 25^oC$, $I_{SD} = 5.6A$, $dI_{SD}/dt = 100A/\mu s$		0.17	0.4	0.83	µC

NOTES:

2. Pulse test: pulse width ≤ 300µs, duty cycle ≤ 2%.
3. Repetitive rating: pulse width limited by max junction temperature. See Transient Thermal Impedance curve (Figure 3).
4. $V_{DD} = 25V$, start $T_J = 25^oC$, $L = 910\mu H$, $R_G = 25\Omega$, peak $I_{AS} = 5.6A$.

Typical Performance Curves Unless Otherwise Specified

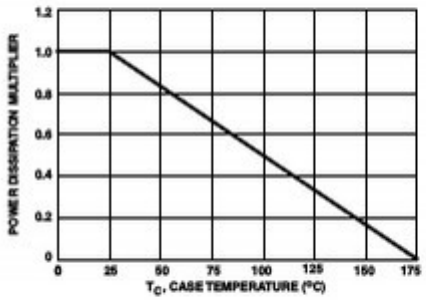


FIGURE 1. NORMALIZED POWER DISSIPATION vs CASE TEMPERATURE

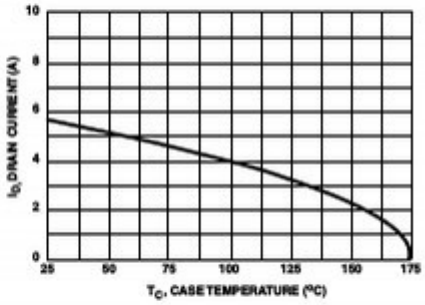


FIGURE 2. MAXIMUM CONTINUOUS DRAIN CURRENT vs CASE TEMPERATURE

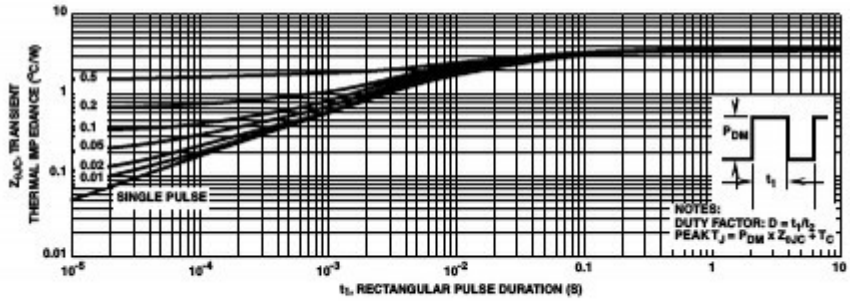


FIGURE 3. MAXIMUM TRANSIENT THERMAL IMPEDANCE

Typical Performance Curves Unless Otherwise Specified (Continued)

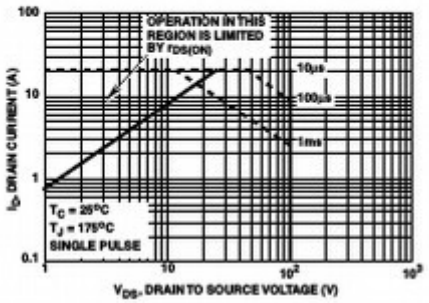


FIGURE 4. FORWARD BIAS SAFE OPERATING AREA

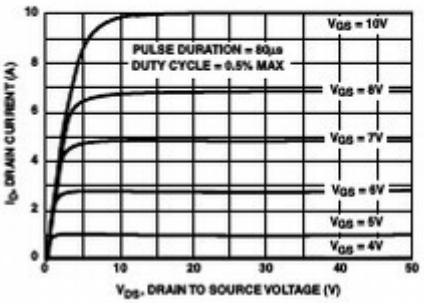


FIGURE 5. OUTPUT CHARACTERISTICS

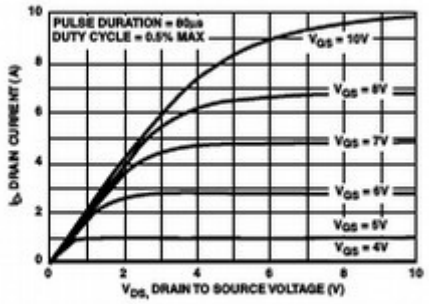


FIGURE 6. SATURATION CHARACTERISTICS

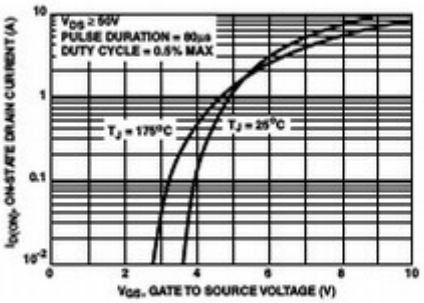


FIGURE 7. TRANSFER CHARACTERISTICS

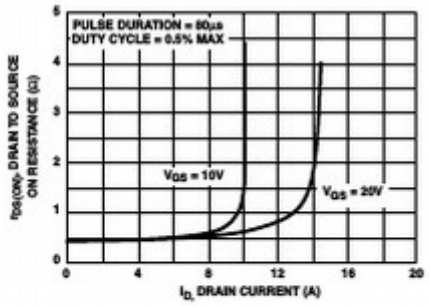


FIGURE 8. DRAIN TO SOURCE ON RESISTANCE vs GATE VOLTAGE AND DRAIN CURRENT

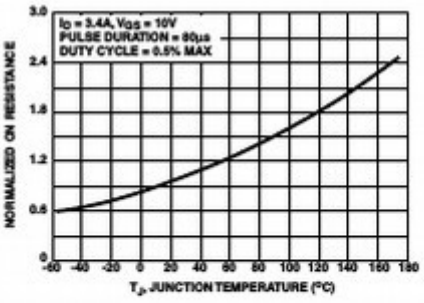


FIGURE 9. NORMALIZED DRAIN TO SOURCE ON RESISTANCE vs JUNCTION TEMPERATURE

Typical Performance Curves Unless Otherwise Specified (Continued)

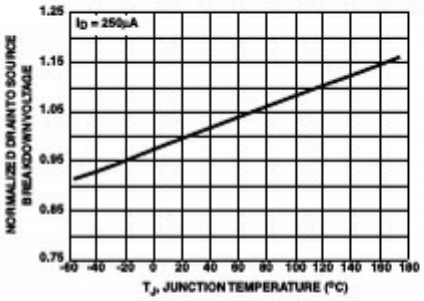


FIGURE 10. NORMALIZED DRAIN TO SOURCE BREAKDOWN VOLTAGE vs JUNCTION TEMPERATURE

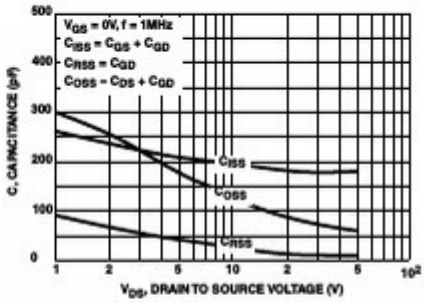


FIGURE 11. CAPACITANCE vs DRAIN TO SOURCE VOLTAGE

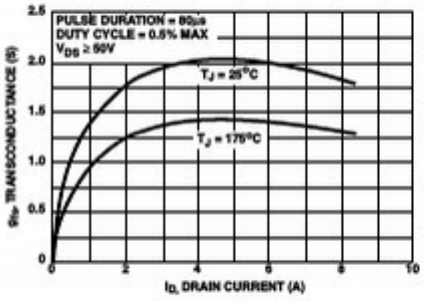


FIGURE 12. TRANSCONDUCTANCE vs DRAIN CURRENT

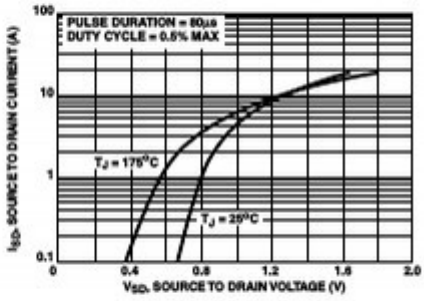


FIGURE 13. SOURCE TO DRAIN DIODE VOLTAGE

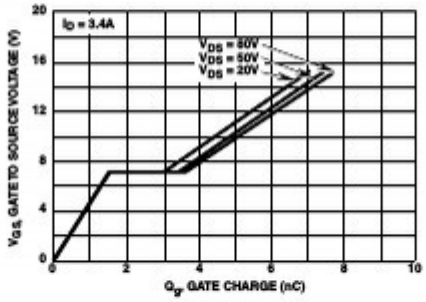


FIGURE 14. GATE TO SOURCE VOLTAGE vs GATE CHARGE

Test Circuits and Waveforms

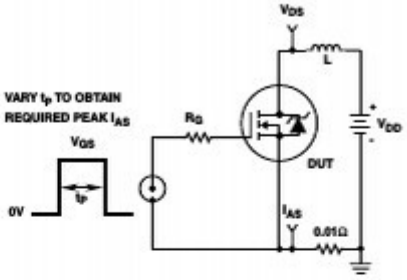


FIGURE 15. UNCLAMPED ENERGY TEST CIRCUIT

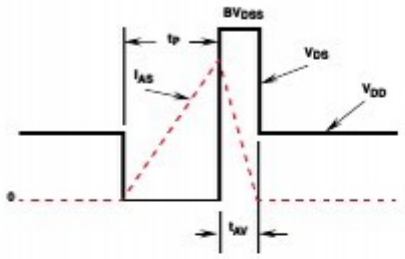


FIGURE 16. UNCLAMPED ENERGY WAVEFORMS

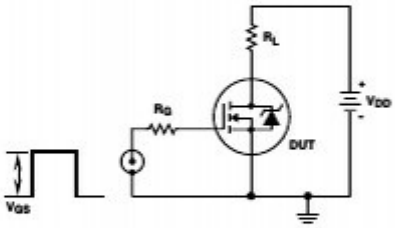


FIGURE 17. SWITCHING TIME TEST CIRCUIT

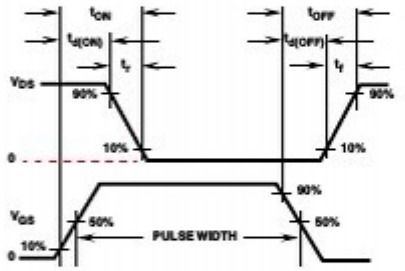


FIGURE 18. RESISTIVE SWITCHING WAVEFORMS

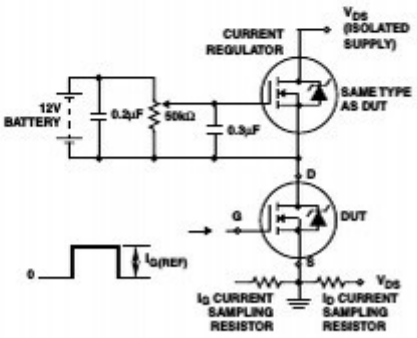


FIGURE 19. GATE CHARGE TEST CIRCUIT

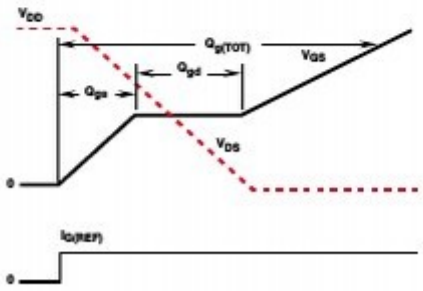


FIGURE 20. GATE CHARGE WAVEFORM

TRADEMARKS

The following are registered and unregistered trademarks Fairchild Semiconductor owns or is authorized to use and is not intended to be an exhaustive list of all such trademarks.

ACEx™	FAST®	OPTOLOGIC™	SMART START™	VCX™
Bottomless™	FASTr™	OPTOPLANAR™	STAR*POWER™	
CoolFET™	FRFET™	PACMAN™	Stealth™	
CROSSVOLT™	GlobalOptoisolator™	POP™	SuperSOT™-3	
DenseTrench™	GTO™	Power247™	SuperSOT™-6	
DOME™	HiSeC™	PowerTrench®	SuperSOT™-8	
EcoSPARK™	ISOPLANAR™	QFET™	SyncFET™	
E²CMOS™	LittleFET™	QS™	TinyLogic™	
EnSigna™	MicroFET™	QT Optoelectronics™	TruTranslation™	
FACT™	MicroPak™	Quiet Series™	UHC™	
FACT Quiet Series™	MICROWIRE™	SILENT SWITCHER®	UltraFET®	

STAR*POWER is used under license

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR CORPORATION.
As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, or (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

PRODUCT STATUS DEFINITIONS

Definition of Terms

Datasheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This datasheet contains preliminary data, and supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
No Identification Needed	Full Production	This datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
Obsolete	Not In Production	This datasheet contains specifications on a product that has been discontinued by Fairchild semiconductor. The datasheet is printed for reference information only.