UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Trabajo Fin de Grado

Desarrollo de un sistema para control de microclimas en cultivo de plantas

Autor: Carlos Reyes Guerola

Director: Dr. Álvaro Tormos Ferrando
Dr. Antonio Guill Ibáñez

AGRADECIMIENTOS:

Al Dr. Álvaro Tormos Ferrando y al Dr. Antonio Guill Ibáñez, directores del Trabajo Fin de Grado, por su confianza y apoyo.

A mi familia y en especial a D. Juan Ramón Reyes Guerola por su apoyo y por hacer posible que pueda realizar este trabajo.

Índice

1. Justificación

El desarrollo óptimo de las plantas requiere unas condiciones de cultivo diferentes para cada especie en cuestión. El presente Trabajo Fin de Grado constituye una propuesta tecnológica para posibilitar que el desarrollo vegetal de cada especie se realice en las condiciones que permiten obtener un mejor producto y un mayor rendimiento del cultivo. El sistema podría tener una aplicación a nivel científico, facilitando por ejemplo el estudio básico de las necesidades hídricas, térmicas y lumínicas de cada planta, algo que a su vez podría repercutir en la producción en masa de un cultivo, o el desarrollo de aditivos para el mismo.

Por otra parte, el trabajo desarrollado puede resultar útil en el ámbito educativo, ya que los alumnos podrían observar el desarrollo de diferentes plantas y aprender de una forma visual y, siempre acompañado de las explicaciones del tutor, cómo funciona el crecimiento de una especie, los nutrientes a añadir al sustrato para mejorar el cultivo y las condiciones meteorológicas que se necesitan para el desarrollo de ésta.

2. Introducción

En este apartado se van a dar unas pequeñas nociones relativas a la creación y control de microclimas, así como su aplicación al cultivo de plantas para comprender mejor el marco del Trabajo de Fin de Grado.

2.1. Creación de microclimas

En este apartado se van a tratar los aspectos más relevantes para entender mejor cuando se habla de microclima en este trabajo.

2.1.1. ¿Qué es un microclima?

Un microclima (1) es un clima que afecta a una zona de espacio reducido en el que sus características climatológicas difieren de la zona en la que se sitúa. Se trata de una serie de variables atmosféricas que distinguen un espacio de dimensiones reducidas.

2.1.2. Tipos de microclimas

Generalmente pueden existir microclimas naturales en los que se dan unas determinadas condiciones debidas a diferentes factores o microclimas artificiales en los que se adaptan las condiciones según diferentes criterios establecidos. Estos últimos suelen ser generados en zonas urbanas debido a la emisión de gases contaminantes que producen el efecto invernadero.

Siendo más específicos, dentro de estos dos grupos, pueden existir diferentes microclimas. Se van a mencionar algunos tipos (2).

- Microclimas urbanos: Las aglomeraciones de centros urbanos generan islas calientes. Este fenómeno es favorecido por la emisión de energía de los edificios conduciendo a aumento considerable de la temperatura de la zona.
- Microclimas costeros: La presencia de masas de agua de gran volumen generan un efecto amortiguador de la temperatura debido a la inercia térmica entre estas masas y el aumento de la presión atmosférica. Estas diferencias producidas entre el agua y la costa se invierten entre el día y la noche.

- Microclimas de montaña: En estas zonas se pueden dar dos condiciones en función de la dirección del viento: El viento que desciende de la ladera que contiene aire húmedo con días cubiertos y con abundantes precipitaciones que en consecuencia, generan pocas irradiaciones solares con una gran amplitud térmica.

2.1.3. Magnitudes posibles a controlar

Para poder cultivar una planta determinada, el control climático se condiciona por cuatro factores:

- Temperatura: La temperatura es uno de los parámetros más importantes a considerar para ajustar un microclima artificial debido a que es la magnitud que más influye en el crecimiento y desarrollo de las plantas. Normalmente, una temperatura óptima para un cultivo está entre los 10 °C y 20 °C.

  Para ajustar la temperatura a cada cultivo se deben conocer las diferentes limitaciones de cada especie. Para ello se definen una serie de parámetros que determinan las diferentes temperaturas en las que una planta puede estar o viceversa.

  - Temperatura mínima letal: Aquella en la que por debajo se producen daños en el cultivo.
  - Temperaturas máximas y mínimas biológicas: Indican valores por encima o por debajo respectivamente del cual, no es posible que una planta alcance una determinada fase vegetativa.
  - Temperatura óptima: Indica el margen de valores óptimos para el desarrollo de un cultivo.

  En la tabla 1 se detallan los valores óptimos de diferentes especies.

Tabla 1: Exigencias de temperatura en grados para distintas especies.

	Tomate	Pimiento	Berenjena	Pepino	Melón	Sandía
Temp. mínima letal	0-2	-1	0	-1	0-1	0
Temp. mínima biológica	10-12	10-12	10-12	10-12	13-15	11-13
Temp. óptima	13-16	16-18	17-22	18-18	18-21	17-20
Temp. máxima biológica	21-27	23-27	22-27	20-25	25-30	23-28
Temp. máxima letal	33-38	33-35	43-53	31-35	33-37	33-37

- Humedad relativa: La humedad relativa es la cantidad de agua contenida en el aire, en relación con la máxima admisible a la misma temperatura. Existe una relación inversa con la temperatura ya que, a temperaturas altas, aumenta la capacidad de contener vapor de agua lo que produce que la humedad relativa sea menor. En cambio, a temperaturas bajas el contenido de humedad relativa aumenta.

  Cada especie tiene una humedad relativa idónea para que la planta pueda crecer en perfectas condiciones. En la tabla 2 se detallan algunos ejemplos.

Tabla 2: Tabla con la humedad relativa óptima para diferentes especies.	
Especie	Humedad relativa
Tomate	50-60 %HR
Pimiento	50-60 %HR
Berenjena	50- 60 %HR
Pepino	70-90 %HR
Melón	50-60 %HR
Sandía	50-60 %HR

La humedad relativa es un factor climático que puede modificar el rendimiento del cultivo. Cuando es excesiva, las plantas reducen su transpiración con lo que se pueden producir abortos florales por el apelmazamiento del polen y hay una mayor posibilidad de desarrollar enfermedades criptogámicas. Por el contrario, si es muy baja, las plantas transpiran en exceso pudiendo deshidratarse produciéndose problemas comunes de mal cuaje.

- Iluminación: A una mayor luminosidad se debe aumentar la temperatura, la humedad relativa y el dióxido de carbono para que la fotosíntesis sea de una eficiencia máxima, por lo contrario, si hay poca luz pueden descender las necesidades de otros factores. Para mejorar la luminosidad natural en un microclima artificial se pueden utilizar los siguientes medios:
    - Materiales con buena transparencia.
    - Una buena orientación de la zona del microclima.
    - Materiales que reduzcan el mínimo las sombras interiores.
    - Aumento del ángulo de incidencia de las radiaciones solares.
    - Acolchado del suelo con plástico blanco.
- CO_2: El anhídrido carbónico de la atmósfera es la materia prima imprescindible de la función clorofílica de los cultivos. El enriquecimiento de la atmósfera del microclima artificial a tratar con CO_2 es muy interesante ya que se puede aprovechar la máxima actividad fotosintética de las plantas. La concentración en la atmósfera de CO_2 es de un 0.03%, pudiendo aumentar esta concentración en un microclima artificial hasta un límite entre un 0.1 y 0.2%. Aumentar por encima de estos valores puede resultar tóxico para el cultivo.

  Los niveles de CO_2 dependen de la especie cultivada, de la radiación solar, de la ventilación, de la temperatura y de la humedad. El nivel óptimo de asimilación está entre los 18 y 23 °C. Con respecto a la luminosidad y humedad, cada planta tiene su valor óptimo. El enriquecimiento con CO_2 puede aumentar el rendimiento de un cultivo hasta un 30%, además de aumentar la calidad del cultivo y de la cosecha del fruto.

2.1.4. Control de un microclima

El control de un microclima está basado en adecuar todos los sistemas instalados en el microclima artificial: Sistema de calefacción que regula la temperatura, la ventilación que reduce la temperatura e introduce oxígeno dentro del habitáculo y el suministro de fertilización carbónica para mantener adecuados los niveles de radiación, temperatura, humedad relativa y el nivel de CO_2, y así poder obtener la máxima respuesta del cultivo y, por tanto, mejorar el rendimiento, la calidad del producto y obtener un cultivo de mayor calidad.

Actualmente son numerosos los sistemas de automatización que existen en el mercado para controlar los diferentes parámetros climáticos que se han mencionado en el apartado 2.1.3 de los microclimas artificiales. Estos sistemas están basados en el uso de un ordenador central al que se conectan diferentes sensores que recogen las variaciones de las diferentes magnitudes con respecto a los unos valores de referencia programados inicialmente. Se trata de una estación meteorológica que registra los valores de temperatura, humedad relativa, iluminación, velocidad del viento, etc.

Estos sistemas pueden conectarse a diferentes sistemas de fertirriego y de regulación climática. Los diferentes sensores o automatismos son distribuidos en diferentes sectores, pudiendo cada uno captar datos de forma autónoma. En controlador central se encarga de recoger toda la información captada por los sensores, que hace coordinar las diferentes actuaciones y se envían las órdenes necesarias a cada uno de los sectores.

2.2. El invernadero

Un claro ejemplo de un microclima artificial es el invernadero. En este apartado se va a explicar el concepto del invernadero y sus características más destacadas.

2.2.1. ¿Qué es un invernadero?

Un invernadero (3) es un recinto cerrado destinado a la horticultura, dotado de una cubierta translúcida de vidrio o material plástico que permite el control de la humedad y otros factores ambientales con el objetivo de favorecer el desarrollo óptimo del cultivo.

2.2.2. Detalles de un invernadero

Actualmente se utilizan los invernaderos para el cultivo de diferentes tipos de plantas u hortalizas. Al usar un invernadero, se puede proteger el cultivo de las variaciones del clima permitiendo una mejor cosecha. Algunos invernaderos incluyen sistemas de control automático de diferentes magnitudes climatológicas para mantener estable el interior del invernadero. En la figura 1 se puede apreciar un ejemplo de un invernadero.

Figura 1: Ejemplo de un invernadero situado en la ciudad de Londres.

En los invernaderos se aprovecha el efecto producido por la radiación solar que, al atravesar la cubierta del invernadero, calienta los objetos que hay dentro y, a su vez, emiten radiación infrarroja (de longitud de onda mayor que la solar) que no pueden atravesar la cubierta, quedando atrapados y produciendo un calentamiento en el habitáculo.

3. Objetivo

El objetivo principal del proyecto es controlar las condiciones ambientales de cultivo de cualquier tipo de planta que se tenga identificada y catalogada en una base de datos. Los requerimientos de temperatura, humedad ambiental y del suelo e iluminación habrán sido previamente almacenados. La adaptación automática de estas variables a las necesidades de la planta permitirá el crecimiento óptimo de la misma. La planta será ubicada en un pequeño habitáculo que permita controlar las condiciones ambientales, mediante el uso de diferentes sensores y todo esto controlado con un microcontrolador junto con una aplicación para la gestión del sistema.

4. Funcionamiento básico del sistema

El sistema se implementará mediante sensores y la plataforma Arduino. La base de datos se programará en MYSQL, mientras que una aplicación realizada en Visual Basic realizará la comunicación entre sistema y base de datos, controlará las variables y monitorizará las magnitudes físicas.

Mediante la aplicación de Visual Basic se añadirán a la base de datos los diferentes tipos de plantas que se desea cultivar. También se puede modificar los datos introducidos en cualquier planta para actualizar sus condiciones idóneas.

Una vez elegida la planta, se configura la conexión bluetooth por el puerto serie que posea dicho módulo con el pc. Una vez configurada la conexión y realizada la misma, se transmiten los datos al sistema.

El siguiente paso es elegir la planta que hay en el habitáculo para transferirla al sistema por el módulo bluetooth y que el arduino reciba los datos.

Cuando el arduino reciba los datos, se pondrá el sistema en funcionamiento para controlar las diferentes magnitudes que se mencionarán en el apartado 3.2. Éste se pondrá a realizar el control del invernadero modificando las magnitudes adaptando el medio del habitáculo a la planta colocada.

Finalmente se puede monitorizar el sistema desde la misma aplicación o desde un terminal táctil ubicado en el mismo invernadero.

5. Diseño del sistema

En este apartado se van a tratar todas las partes detalladamente que conforman el sistema diseñado.

5.1. Diagrama de bloques

En la figura 2 se pueden apreciar los bloques que definen el sistema.

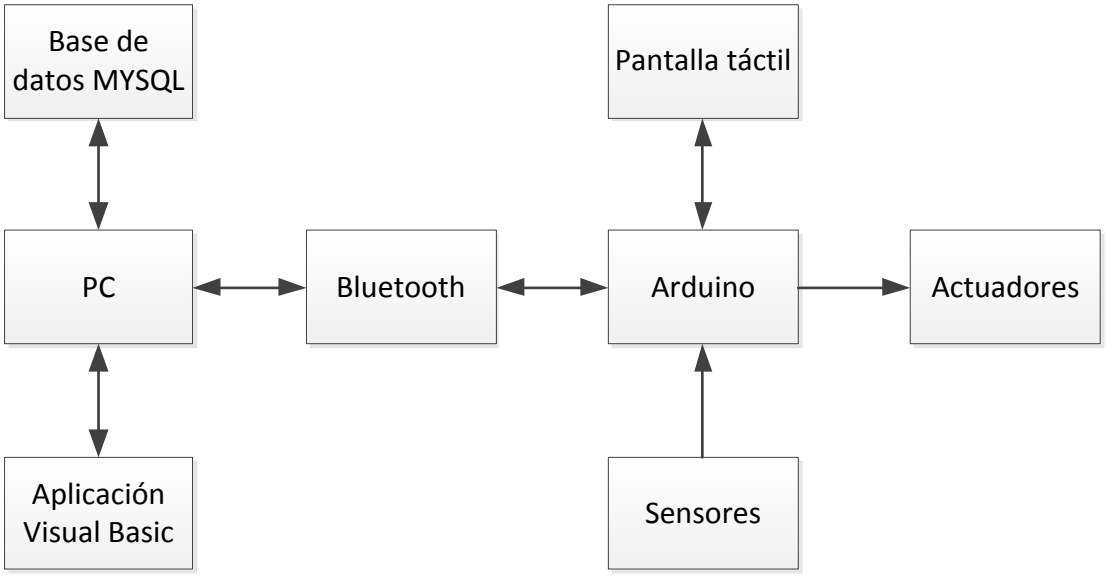


Figura 2: Diagrama de bloques del sistema.

- Base de datos MYSQL: Base que contiene toda la información respectiva a las plantas a cultivar.
- PC: Ordenador que gestiona la aplicación de Visual Basic.
- Aplicación Visual Basic: Aplicación desarrollada que gestiona la base de datos ubicada en el PC y que mantiene comunicación con el sistema.
- Bluetooth: Estándar de comunicación utilizado entre la aplicación de Visual Basic y la planta de trabajo.
- Arduino: Microcontrolador encargado de controlar todo el sistema de forma ininterrumpida. Se utilizará el Arduino Mega 2560 (4) (Arduino, Italia).
- Pantalla táctil: Dispositivo gráfico con el que poder interactuar para monitorizar la planta de trabajo sin tener que utilizar el PC.
- Actuadores: Elementos electrónicos que van a permitir modificar las condiciones del microclima.
- Sensores: Elementos electrónicos que van a servir para captar las magnitudes que se van a medir para realizar su correspondiente regulación.

5.2. Magnitudes a medir

Las magnitudes que se van a controlar en este sistema son las siguientes:

- Temperatura: Ajustar la temperatura a la planta cultivada para evitar que se queme por un exceso de calor o se congele debido a la baja temperatura. Se va a utilizar el sensor digital DHT22 (5) (Aosong electronics, China).
- Humedad relativa del aire: ajustar la humedad del aire para que no sea demasiado seco y la planta se asfixie o que la planta tenga demasiada humedad y muera por sobre hidratación. Se utilizará el mismo sensor digital que el empleado para la temperatura, el sensor DHT22, que mide ambas magnitudes.
- Humedad del suelo: Ajustar la cantidad de agua que sea necesaria en la tierra para que no se seque ni esté demasiado saturada. Se utilizará el sensor analógico SEN92355P (6) (Seed Technology, China).

- Iluminación: controlar la cantidad de luz que necesite la planta durante las horas sin luz o para suplir la luz natural de forma que sea constante su producción. Se va a utilizar un fotodiodo BPW34 (7) (Vishay, Estados Unidos).

5.3. Especificaciones

Se pretende mantener las siguientes especificaciones:

- Control de la temperatura del ambiente con una histéresis de 2 grados.
- Control de la humedad relativa del aire con una histéresis de un 2 %HR.
- Control de la humedad del terreno con una histéresis del 2 %HR.
- Control de la iluminación para compensar la falta de luz natural en ambientes oscuros o cuando cae la noche.
- Tener un sistema autónomo una vez introducidos los datos de las referencias.
- Utilizar alimentación simple para la electrónica analógica a implementar.
- Filtrar todas las señales de ruidos procedentes del exterior que puedan perturbar y afectar a la medida del sistema.
- Tener la potencia aislada de la parte de control.

5.4. Comunicaciones

En este apartado se va a detallar como se configuran las diferentes comunicaciones que tiene el sistema para su funcionamiento completo.

5.4.1. Bluetooth

Se va a utilizar un módulo bluetooth HC-06 (8) (Guangzhou HC Technology, China) que permite la comunicación serie entre dispositivos. Para este caso, se configura una conexión entre el PC que contiene la base de datos y la aplicación de Visual Basic y el arduino que realiza el control del microclima a tratar.

El dispositivo HC-06 sólo puede actuar como esclavo, es decir, no puede iniciar una comunicación, por lo que el maestro siempre va a ser el PC. Esto es para que el arduino no pueda enviar datos sin el consentimiento del usuario que contenga el PC ya que debe ser éste mismo quién, a partir del ordenador, decida cuando consultar el estado del sistema o cuando introducir nuevos parámetros.

Para poder establecer una comunicación entre el módulo y otro dispositivo, el dispositivo en cuestión debe configurar su comunicación. En este caso, es un ordenador el que va a establecer la comunicación, por lo que debe conectarse a la misma. El PC crea una un puerto serie virtual por el cual se transmiten los datos entre el módulo y éste. Éste puerto COM virtual permite actuar sobre el módulo bluetooth del PC y gestiona la transmisión por el mismo. El módulo HC-06 recibe la información y la transmite por su pin de recepción (Rx) y, en el caso de generar una respuesta, el módulo bluetooth transmite en el sentido opuesto por el pin de transmisión (Tx) En la figura 3 se puede apreciar de forma esquemática como es la transmisión entre el equipo y el módulo.

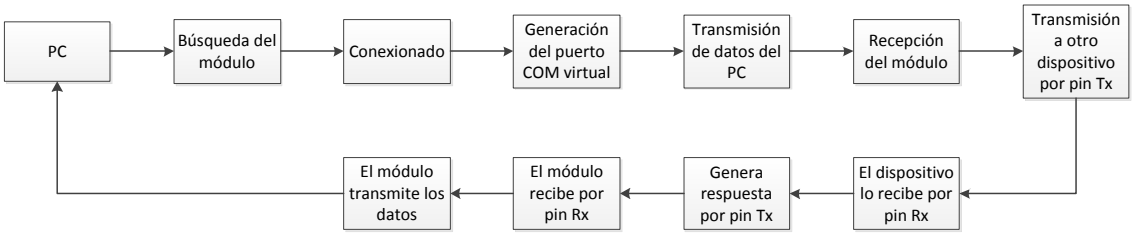


Figura 3: Pasos en la primera comunicación por bluetooth.

Una vez se ha realizado la primera comunicación, en las posteriores comunicaciones ya no se busca el módulo y no se genera un nuevo puerto serie virtual. Se utiliza el mismo ya que se ha quedado guardado en la primera conexión. A continuación, se procederá a explicar cómo se ha realizado toda esta misma configuración explicada adaptada al proyecto y de una forma más detallada.

El primer paso es configurar la comunicación entre el PC y el módulo bluetooth. Para ello se alimenta el módulo a 5 voltios y desde el PC se busca el periférico. Tal y como se indica en la figura 4 se ha encontrado el dispositivo y se introduce la contraseña que tiene el módulo.

Figura 4: Emparejamiento e introducción de la contraseña de seguridad.

Tanto el nombre como contraseña pueden ser modificados mediante comandos AT, además de otros parámetros que pueden configurar la comunicación con el módulo. En este caso, se dejan los parámetros por defecto. Una vez emparejado el dispositivo aparecerán dos puertos COM virtuales que permiten actuar sobre el módulo bluetooth. Aunque se actúe sobre un

puerto, se está actuando sobre los dos a la vez, esto es debido a que uno actúa sobre la transmisión y otro sobre la recepción. A efectos visuales, se está trabajando sobre un único puerto. En el administrador de dispositivos deben aparecer dichos puertos COM y se configurarán tal y como se muestra en la figura 5.

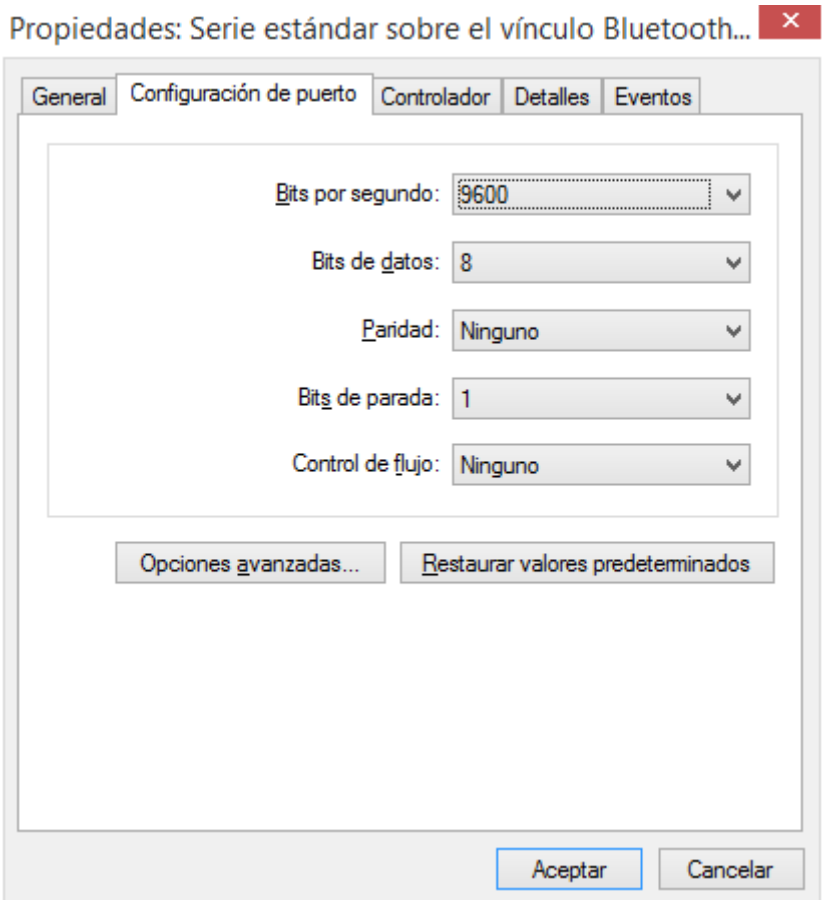


Figura 5: Configuración de los puertos COM para la comunicación bluetooth.

Como se ha dicho anteriormente, esta configuración puede ser cambiada a placer con los comandos AT, pero se va a utilizar la configuración por defecto en el módulo.

Debido a que el pin de recepción del módulo bluetooth (RX) soporta un voltaje máximo de 3,3 V se debe implementar un pequeño divisor de tensión para emplear las salidas digitales a 5 V del Arduino. A partir de la ecuación del divisor de tensión (Ec. 5.4.1.1) se van a obtener los valores de las resistencias a colocar.

$$V_{Rx} = \frac{R2}{R1} * V_{Tx} \qquad (Ec. 5.4.1.1)$$

$$Si\ R2 = 10\ k\Omega$$

$$3.3 = \frac{10 * 10^3}{R1 + 10 * 10^3} * 5$$

$$R1 = 5.15\ k\Omega$$

En la figura 6 se puede apreciar el resultado de dicho divisor de tensión.

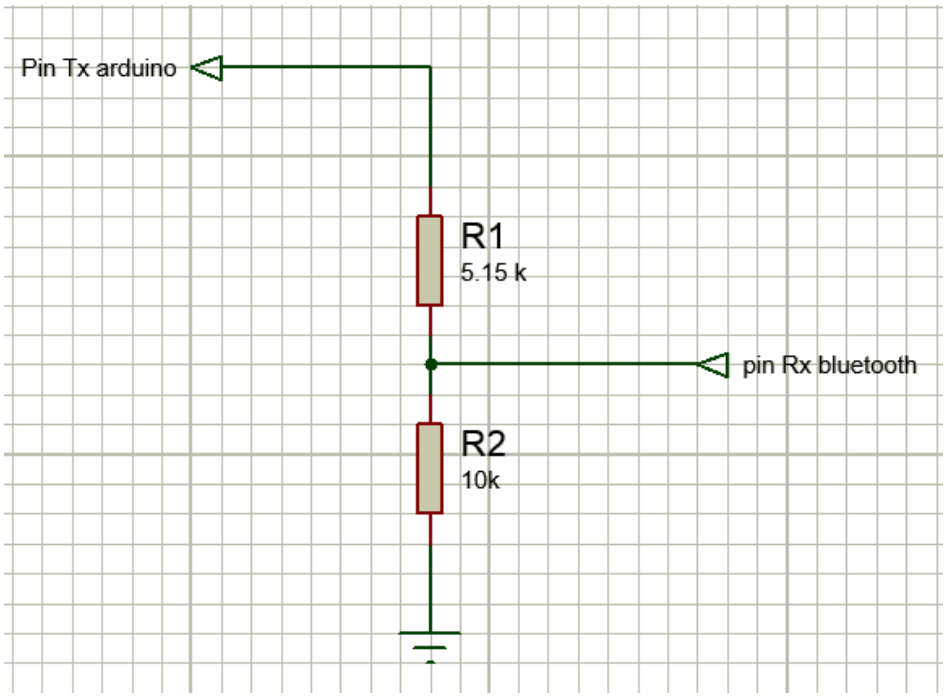


Figura 6: Esquema del divisor de tensión para la recepción en el módulo bluetooth.

Para transmitir los datos, tanto desde la aplicación de Visual Basic al Arduino y viceversa, se ha realizado una pequeña trama en la que se distinguen los datos de lo que es el final de la cadena. En la tabla 3 se observa el formato de dicha trama.

Tabla 3: Formato de la cadena de caracteres a transmitir por bluetooth.	
Dato	Dato
Cadena de datos	f

Esta trama se compone de los datos a transmitir y de un carácter de final de trama (f) que simboliza que no se van a transmitir más datos.

También se ha diseñado una pequeña tabla de códigos de funciones para que el sistema sepa que acción debe hacer cuando se le transmita una orden desde la aplicación de Visual Basic. En la tabla 4 se detalla cada código.

Tabla 4: Tabla de códigos para la transmisión por bluetooth.	
Código	Función
110	Inicia el sistema
111	Para el sistema
112	Inicia la consulta de los datos actuales del sistema. Esto es para cuando se desea monitorizar el sistema
113	Permite transferir los datos de una planta desde la aplicación de Visual Basic al Arduino.
114	Permite saber si el sistema está en marcha o está parado para que la sincronización por bluetooth sea correcta.

5.4.2. Base de datos MYSQL

Mediante la aplicación Xampp (9) se puede crear un servidor tanto local como en red para tener una base de datos y poder consultarla, modificarla o añadir nuevos parámetros según las necesidades que puedan surgir. En este proyecto, se va a implementar una nueva tabla en la que se almacenarán los parámetros que aparecen en la tabla 5.

Tabla 5: Formato de la tabla en la base de datos MYSQL					
Código planta	Nombre	Temperatura	Humedad aire	Humedad suelo	Iluminación

En este caso, la comunicación será entre la base de datos ubicada en el PC y la aplicación de Visual Basic, que también estará en el mismo PC. En el correspondiente apartado para la aplicación de Visual Basic, se explicará cómo se obtienen o guardan los datos.

En este apartado se procederá a explicar cómo se ha creado la base de datos y cómo se añaden parámetros nuevos, así como el tipo de sentencias que se utilizarán.

El primer paso es crear la base a utilizar en cuestión, ya que un mismo PC puede albergar diferentes bases de datos con datos que no tengan relación alguna entre sí. En la figura 7 se muestra cómo se va a nombrar la base de datos perteneciente a este proyecto.

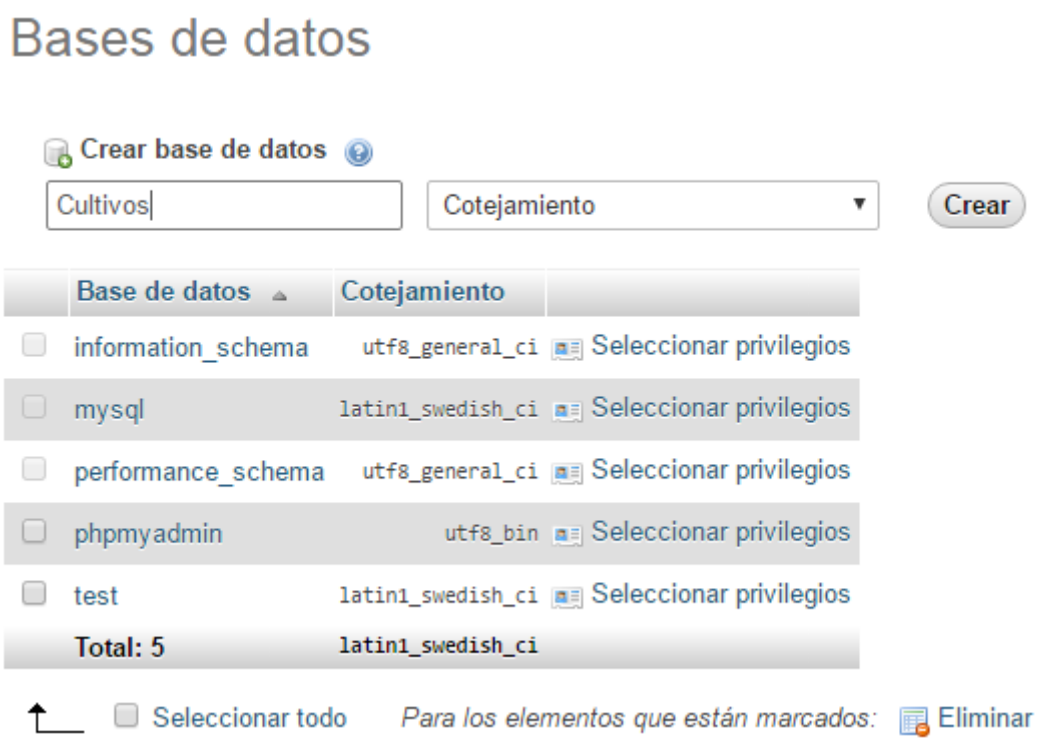


Figura 7: Creación de la nueva base de datos para la gestión de parámetros.

Una vez creada la base, se va a crear una nueva tabla donde se albergarán los datos de cada planta a cultivar. En la figura 8 se puede apreciar cómo se crea una nueva tabla.

Figura 8: Captura con la nueva tabla y el número de columnas que va a tener dicha tabla.

Una vez se crea la tabla, se va a nombrar cada parámetro de la pertinente tabla, el tipo de datos, márgenes, valores por defecto, etc. En la figura 9 se aprecia cómo van a ser cada uno de los parámetros de la nueva tabla creada.

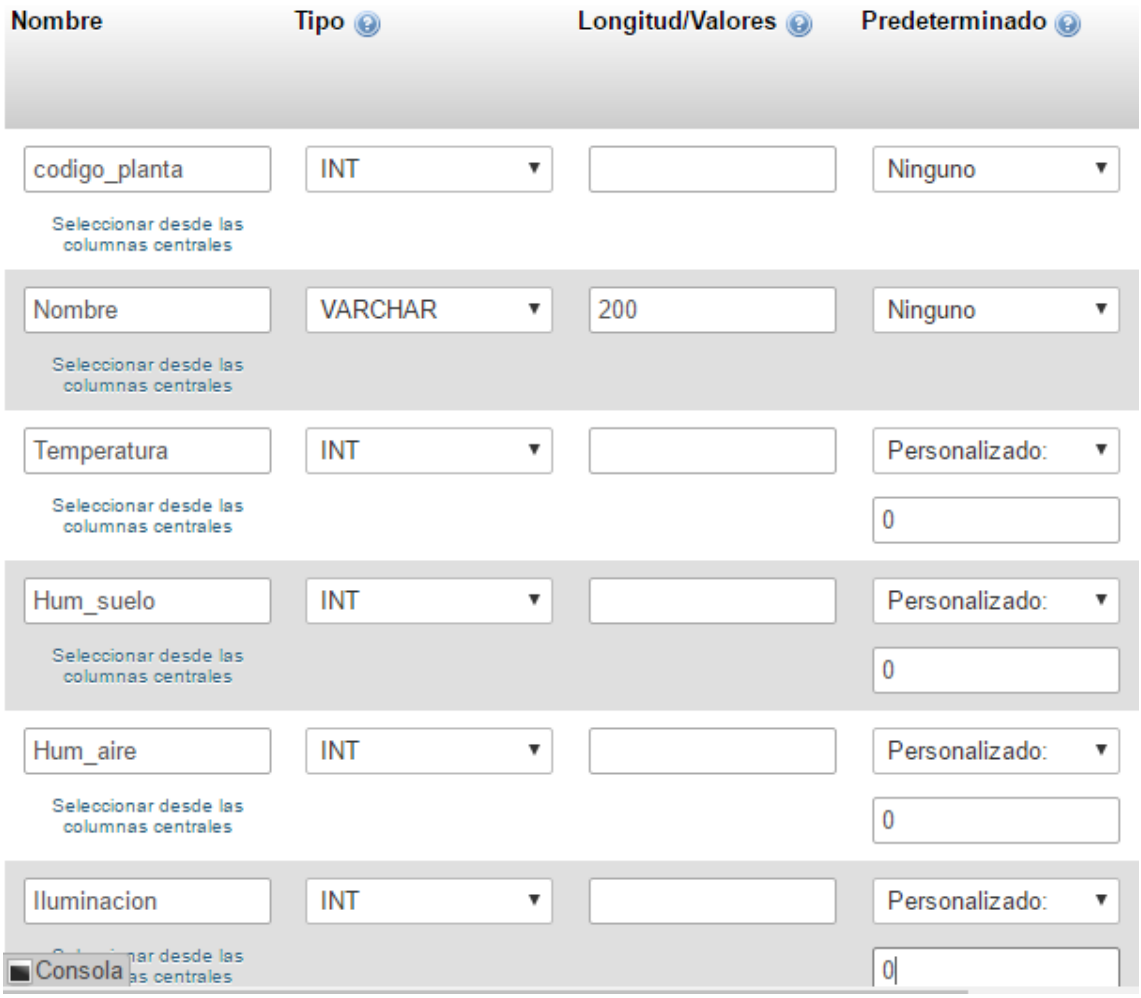


Figura 9: Figura con las características de los parámetros de la nueva tabla creada.

Finalmente queda introducir un parámetro para poder realizar pruebas con la aplicación de Visual Basic. Las sentencias que se van a utilizar en este proyecto se pueden observar en la tabla 6.

Tabla 6: Tabla con las principales sentencias a utilizar en el proyecto.	
Sentencia	Función
SELECT	Permite leer de la base de datos.
UPDATE	Permite modificar datos ya añadidos.
INSERT INTO	Permite añadir nuevos datos

Para Insertar un nuevo parámetro, tan sólo se debe ir a la pestaña SQL y, mediante una de las sentencias mencionadas en la tabla 3, se puede añadir un nuevo parámetro. En la figura 10, se puede apreciar el formato de una sentencia para añadir un nuevo parámetro a partir de los datos consultados en diferentes páginas web (10) (11).

```
1 INSERT INTO `Plantas`(`codigo_planta`, `Nombre`, `Temperatura`, `Hum_suelo`, `Hum_aire`, `Iluminacion`) VALUES
  ('1','Tomate','14','55','55','20000')
```

Figura 10: Ejemplo de sentencia a introducir en la nueva tabla creada.

5.5. Adaptación de señales

En este apartado se va a tratar la adaptación de las señales de los sensores analógicos que se van a utilizar al rango de las entradas analógicas del arduino.

5.5.1. Medida de humedad del suelo

Tal y como se ha indicado en el apartado 5.2 se va a utilizar el sensor SEN92355P para obtener la humedad del suelo de la planta. Debido a que dicho sensor no puede ofrecer una señal analógica dentro del rango de medida de las entradas analógicas del arduino a utilizar, se debe implementar un pequeño amplificador en configuración no inversor para ajustar la amplitud al rango necesario. Este tipo de amplificador es de alimentación simple y Rail to Rail, es decir, que puede ofrecer prácticamente el mismo nivel de tensión máximo que el de alimentación. El amplificador a utilizar es el LMC6042 (12) (Texas Instruments, Estados Unidos).

El rango de la salida del sensor es de 0 a 4.67 V, mientras que el rango que se desea a la salida del amplificador es de 0 a 5 V, por lo que se obtiene la ganancia a partir de la ecuación 5.5.1.1 para obtener la ganancia del amplificador según las condiciones necesarias.

$$G = \frac{V_s}{V_e} \qquad Ec.\,5.5.1.1$$

$$G = \frac{5}{4.67} = 1.07\ V/V$$

Una vez obtenida la ganancia del amplificador se pueden obtener los valores de las resistencias necesarias para que esta ganancia sea efectiva. En la ecuación 5.5.1.2 se muestra la fórmula para obtener los valores resistivos necesarios en la configuración no inversor del amplificador.

$$G = 1 + \frac{R4}{R3} \qquad Ec\ 5.5.1.2$$

$$Si\ R3 = 10\ k\Omega$$

$$1.07 = 1 + \frac{R4}{10 * 10^3}$$

$$R4 = 706.64\ \Omega \rightarrow 680\ \Omega$$

Finalmente en la figura 11 se puede apreciar el resultado del amplificador con el conjunto de resistencias.

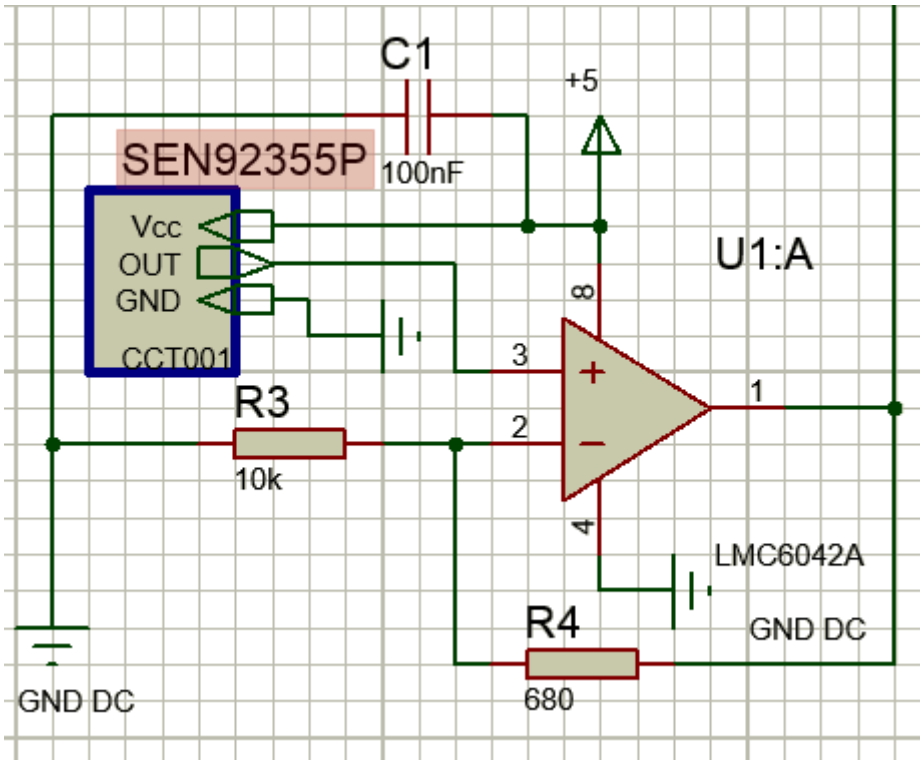


Figura 11: esquema del amplificador no inversor para el sensor de humedad del suelo.

El condensador C1 que aparece en la figura 10 es un condensador de desacoplo que es utilizado para evitar los ruidos procedentes de la fuente de alimentación.

5.5.2. Medida del nivel de iluminación

Para la medida del nivel de iluminación se va a utilizar como sensor un fotodiodo. Este tipo de sensor tiene la ventaja de que es un sensor muy lineal en su rango de funcionamiento. A su vez, la señal que proporciona con respecto a la cantidad de luz que incide sobre el fotodiodo es muy baja, por lo que se debe amplificar y tratar para que se pueda a justar al rango de medida de las entradas analógicas que contiene el arduino. Este tipo de señal es en corriente, ya que este sensor variaría su corriente de luminosidad y de oscuridad ($I_p\ e\ I_{dark}$) dependiendo de la luz que incida sobre el mismo.

Como primera parte, se realiza un tratamiento para amplificar ligeramente la señal del fotodiodo ya que, como se ha mencionado anteriormente, éste presenta una ganancia muy baja. Para ello se utiliza un amplificador operacional con una resistencia (Rf) de valor elevado para aumentar la señal y reducir los posibles ruidos que pueda ofrecer el mismo, además de convertir la ganancia en corriente del fotodiodo a una ganancia en tensión para poder amplificarla y leerla con el arduino. Se va a utilizar una resistencia de valor 180 kΩ para dicho tratamiento, ya que, con esta resistencia, la ganancia queda aumentada ligeramente y no se sobrepasa el nivel máximo de la entrada analógica. En la figura 12 se puede apreciar el circuito resultante de esta parte.

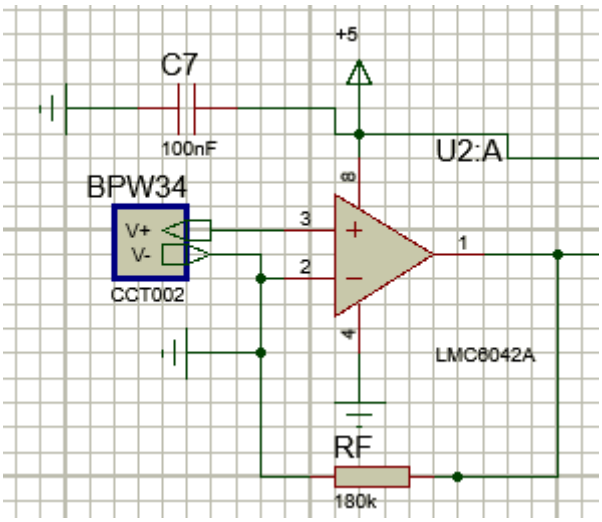


Figura 12: Circuito que trata la baja ganancia del fotodiodo y la convierte a ganancia en tensión.

El condensador C7 hace función de condensador de desacoplo para evitar ruidos procedentes de las fuentes de alimentación.

El rango que ofrece el fotodiodo a máxima incidencia de luz dentro del habitáculo es de 1.55 V. Este valor es cuando la iluminación artificial está incidiendo sobre el diodo. No se considera el efecto por la luz natural debido a que ésta es considerada un ruido que no se puede reducir mediante el uso de componentes eléctricos. Por lo que la medida efectiva es cuando se suprime la luz natural, es decir, en un ambiente oscuro.

Una vez obtenido éste valor se procede a calcular la ganancia necesaria para ajustar al rango de las entradas del arduino (0-5 V) mediante la ecuación 5.5.2.1.

$$G = \frac{V_s}{V_e} \qquad Ec\ 5.5.2.1$$

$$G = \frac{5}{1{,}55} = 3{,}226\ V/V$$

Con este valor ya calculado se procede a obtener el valor de las resistencias de la configuración seleccionada con el amplificador operacional. En este caso, se va a utilizar una configuración no inversora que se puede calcular a partir de la ecuación 5.5.2.2.

$$G = 1 + \frac{R8}{R9} \qquad Ec\ 5.5.2.2$$

$$Si\ R8 = 10\ k\Omega$$

$$3{,}226 = 1 + \frac{10 * 10^3}{R9}$$

$$R9 = 4{,}49\ k\Omega$$

Finalmente, el resultado del circuito completo es el mostrado en la figura 13.

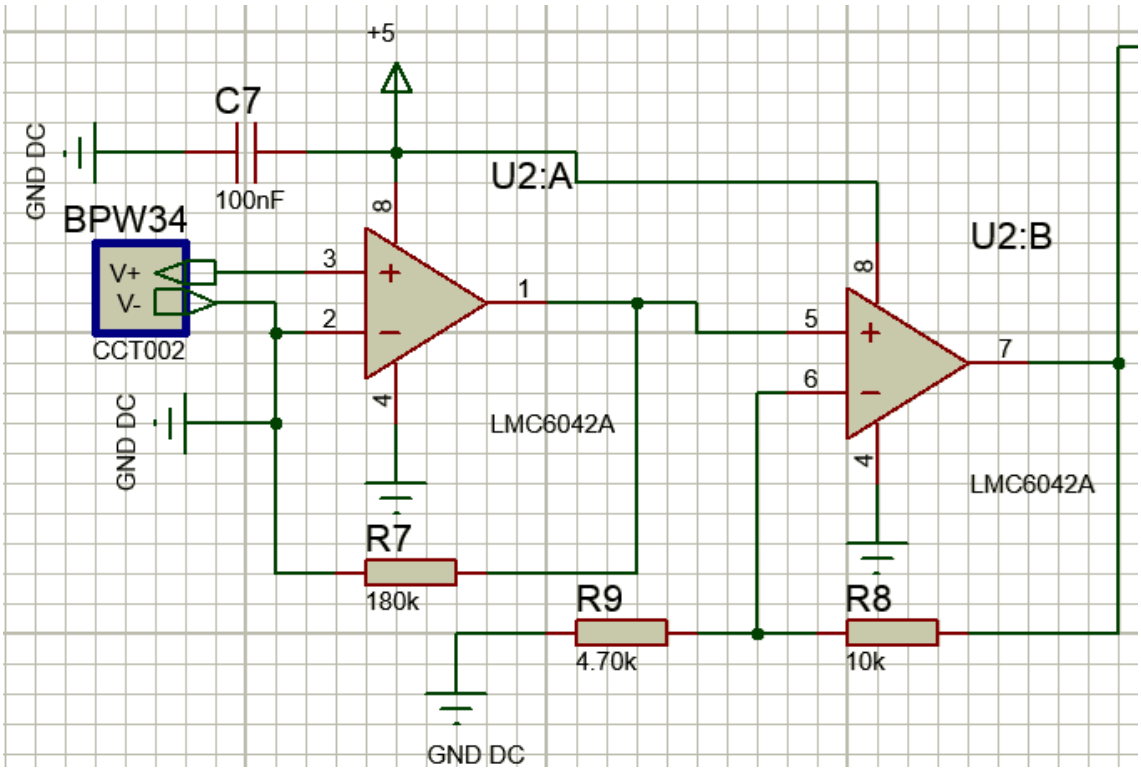


Figura 13: Circuito con todas las etapas para amplificar la señal del fotodiodo.

5.6. Filtrado

Como parte del filtrado de señales para evitar ruidos procedentes del exterior, se ha decidido implementar un filtrado paso-bajo para eliminar interferencias de alta frecuencia. Para ello se ha diseñado un filtro digital único para todas las magnitudes. Esto permite ahorrar en componentes electrónicos tales como amplificadores, resistencias y condensadores, y disponer de una mayor versatilidad a la hora de realizar modificaciones.

El primer paso es calcular el orden del filtro. Para ello se va a estimar un ruido de alterna de 50 Hz (ruido procedente de la señal eléctrica) de 1 V voltio de amplitud a la entrada del filtro. Se desea atenuar en 20 dB, es decir, que éste ruido sea 10 veces menor a la salida del filtro. Se determina que la frecuencia de corte del filtro es de 20 Hz, lo suficientemente baja para eliminar cualquier ruido procedente. El filtro además no debe atenuar ni amplificar la señal de medida, es decir, que debe tener ganancia unitaria.

A partir de las condiciones mencionadas anteriormente, se calcula la pendiente del filtro a partir de la ecuación 5.6.1.

$$Pendiente = \frac{G_f - G_i}{\log\left(\frac{f_f}{f_i}\right)} \qquad Ec\ 5.6.1$$

$$Pendiente = \frac{-20 - 0}{\log\left(\frac{50}{20}\right)} = -50\ dB \rightarrow 60\ dB$$

A partir del resultado de la ecuación 5.6.1 se puede determinar el orden mediante la ecuación 5.6.2.

$$Orden = \frac{|Pendiente|}{20} \qquad Ec\ 5.6.2$$

$$Orden = \frac{|-60|}{20} = 3$$

Ya con el orden necesario para la implementación del filtro digital, se va a obtener el polinomio Butterworth que corresponde a un filtro de tercer orden. El polinomio es el siguiente:

$$G_{filtro}(s) = s^3 + 2 * s^2 + 2 * s + 1$$

Sabiendo que la función de transferencia de un filtro paso-bajo es $\frac{1}{G_{filtro}}$ y siendo s igual $\frac{S}{\omega_c}$, con lo que ω_c es la frecuencia de corte en radianes, se puede obtener dicha función de transferencia. Por lo que la función de transferencia del filtro queda de la siguiente forma:

$$fdt_{filtro}(s) = \frac{1}{\left(\frac{s}{2*\pi*20}\right)^3 + 2*\left(\frac{s}{2*\pi*20}\right)^2 + 2*\left(\frac{s}{2*\pi*20}\right) + 1}$$

Ya con la función de transferencia del filtro en continuo obtenida, se va a proceder a comprobar el resultado del filtro y discretizarlo con la herramienta de Matlab, *sisotool*. En la figura 14 se aprecia el diagrama de bode resultante del filtro.

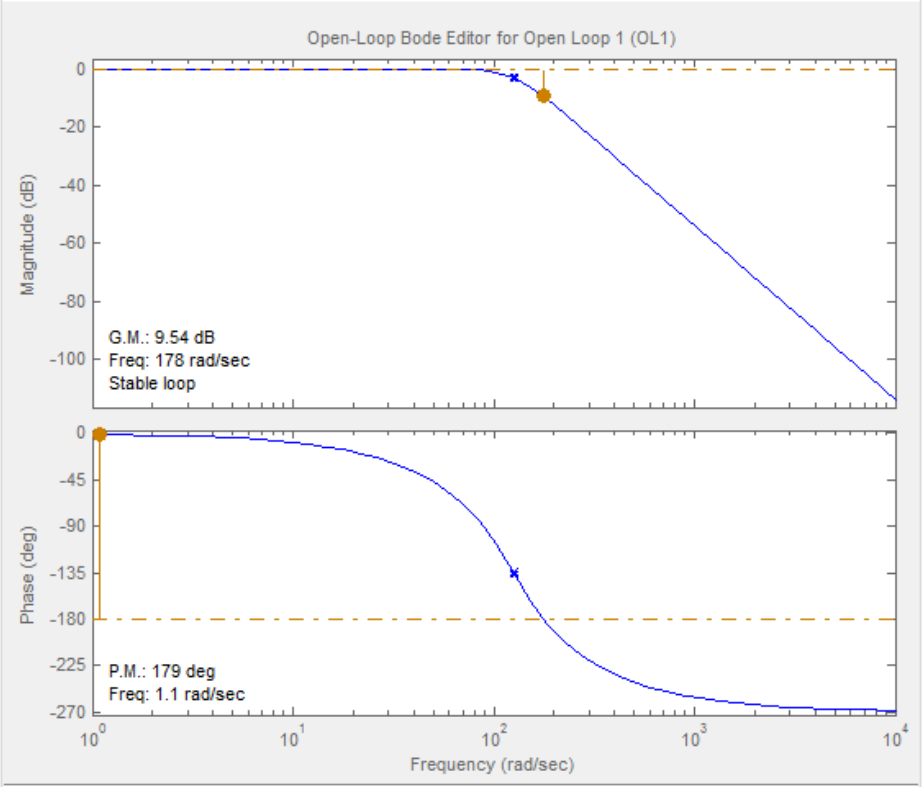


Figura 14: Diagrama de bode del filtro digital obtenido.

Para discretizar el filtro se debe tener en cuenta el teorema de muestreo, ya que la frecuencia de muestreo de la señal debe ser dos veces mayor que la frecuencia original. Por lo que en la ecuación 5.6.3 se va a calcular la frecuencia mínima posible para poder muestrear con el filtro.

$$f_{muestreo} \geq 2 * f_{original} \qquad Ec\ 5.6.3$$

$$f_{muestreo} \geq 2 * 20$$

$$f_{muestreo} \geq 40\ Hz \rightarrow 100\ Hz$$

Mediante la herramienta de sisotool de Matlab se disctretiza el filtro obtenido para obtener su ecuación discreta. En la figura 15 se observa el método utilizado para la discretización y el tiempo de muestreo.

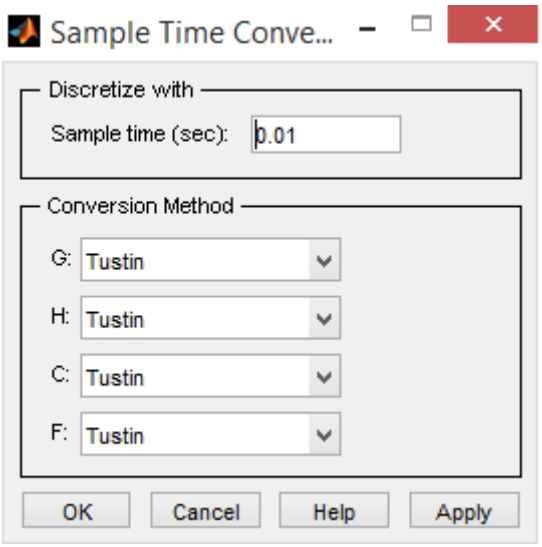


Figura 15: Discretizado del filtro con el tiempo de muestreo deseado.

Finalmente la función de transferencia discreta resultante es la siguiente:

$$G_{discreta}(Z) = \frac{0.0753 * Z^3 + 0.2259 * Z^2 + 0.2259 * Z + 0.0753}{Z^3 - 0.8266 * z^2 + 0.5154 * Z - 0.08648}$$

Como en el arduino no se puede colocar una función de transferencia y operar con la misma, se va a proceder a obtener la ecuación en diferencias para obtener una ecuación que iguale la salida del filtro con los parámetros necesarios a la entrada del mismo. Sabiendo que $G_{discreta(Z)} = \frac{y(Z)}{x(Z)}$ siendo y(Z) la salida del filtro y x(Z) la entrada del mismo, se obtiene la siguiente ecuación.

$$\frac{y(Z)}{x(Z)} = \frac{0.0753 * Z^3 + 0.2259 * Z^2 + 0.2259 * Z + 0.0753}{Z^3 - 0.8266 * z^2 + 0.5154 * Z - 0.08648}$$

Se va a proceder a despejar para dejar los parámetros con y(z) a un lado y los parámetros con x(Z) al otro lado de la ecuación.

$$y(Z) * Z^3 - y(Z) * 0.8266 * Z^2 + y(Z) * 0.5154 * Z * y(Z) * 0.08648 = x(Z) * 0.0753 * Z^3 + x(Z) * 0.2259 * Z^2 + x(Z) * 0.2259 * Z + x(Z) * 0.0753$$

Sabiendo que tanto x(Z) e y(Z) es igual a x(k) e y(k), se procede a sustituir en la ecuación por sus parámetros finitos.

$$y(k+3) - 0.8266 * y(k+2) + 0.5154 * y(k+1) - 0.08648 * y(k) = 0.0753 * x(k+3) + 0.2259 * x(k+2) + 0.2259 * x(k+1) + 0.0753 * x(k)$$

Como no se pueden tener parámetros futuros a parámetro actual se sustituye x(k+3) e y(k+3) por x(k) e y(k).

$$y(k) - 0.8266 * y(k-1) + 0.5154 * y(k-2) - 0.08648 * y(k-3) = 0.0753 * x(k) + 0.2259 * x(k-1) + 0.2259 * x(k-2) + 0.0753 * x(k-3)$$

Finalmente sólo queda despejar para que la ecuación en diferencias tenga su salida igualada a los parámetros anteriores necesarios.

$$y(k) = 0.0753 * x(k) + 0.2259 * x(k-1) + 0.2259 * x(k-2) + 0.0753 * x(k-3) + 0.8266 * y(k-1) - 0.5154 * y(k-2) + 0.08648 * y(k-3)$$

Todo esto se ha compilado en un fichero de Matlab del tipo .m que permite implementar todos los pasos realizados anteriormente. En la figura 16 se detalla dicho fichero de Matlab.

```
s=tf('s');
f=20; %frecuencia de corte
rad=f*2*pi; %obtener la frecuencia en radianes
pb=s/rad;
p=pb^3+2*pb^2+2*pb+1; %polinomio de tercer orden butterworth adaptado a paso bajo
Gc=1/p; %Se obtiene la fdt del filtro
sisotool(Gc); %Se abre el sisotool para discretizar el filtro y ver su comportamiento
Gd=(0.0753*z^3+0.2259*z^2+0.2259*z+0.0753)/(z^3-0.8266*z^2+0.5154*z-0.08648); %se obtiene la fdt discreta

```

Figura 16: Fichero de Matlab en el que se implementa el proceso del diseño del filtro digital.

5.7. Medida de humedad del aire y temperatura

Como se ha mencionado en el apartado 5.2, se va a utilizar el sensor digital DHT22 (13). Este sensor permite obtener la medida de la temperatura y de la humedad del ambiente a la vez y lo traduce a una cadena de 8 bits en forma de pulsos. La cadena de pulsos tiene la forma que se muestra en la figura 17.

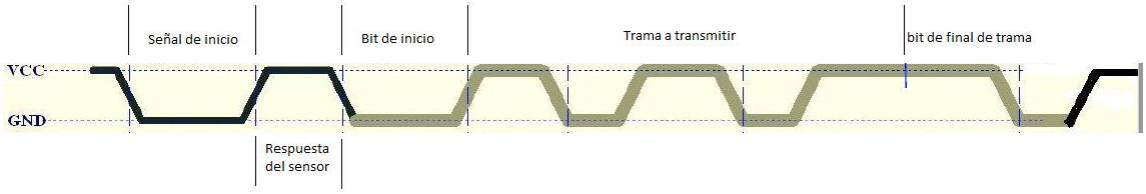


Figura 17: Forma de la cadena de pulsos entre el Arduino y el sensor.

Mediante el uso de una librería para Arduino, se puede interpretar dicha cadena de pulsos por un pin digital cualquiera sin utilizar ninguna entrada analógica. Esto contribuye a reducir el número de componentes de la adaptación de la señal y del filtrado de la misma.

Para que el sensor funcione correctamente se debe colocar una resistencia de pull-up para que la entrada digital del arduino pueda interpretar la señal correctamente sin tener falsos estados bajos o estados altos.

El módulo que se ha utilizado ya lleva dicha resistencia, siendo ésta de un valor resistivo de 4,7 kΩ. Además incluye un condensador de filtrado en paralelo a la resistencia para reducir el ruido de capacidad 100 nF. En la figura 18 se detalla el resultado de la conexión al Arduino.

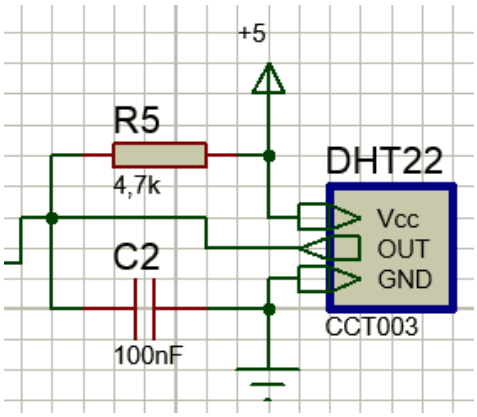


Figura 18: Esquema del conexionado del sensor DHT22 al Arduino.

En este caso, como se ha mencionado anteriormente, R5 y C2 son la resistencia y el condensador que ya van incorporados en el módulo que se ha decidido utilizar.

Como pequeño defecto, este sensor tiene un tiempo de muestreo de 2 segundos que hace que sea algo lento, pero como las magnitudes de la temperatura y humedad son procesos lentos, es decir, que una gran variación se produce tras un largo periodo de tiempo.

Para configurar el sensor y poder obtener datos de la temperatura y humedad del ambiente debe usar la librería correspondiente al sensor. En la figura 19 se aprecia la configuración que se debe implementar en el código del arduino para poder utilizar el sensor.

```
#include <DHT.h>//librería para el uso del sensor DHT22


#define DHTPIN 46//se define el pin del sensor DHT22
#define DHTTYPE DHT22 //Se define el tipo de sensor utilizado(librería válida para varios sensores de la serie DHT)

DHT dht(DHTPIN, DHTTYPE,6);//se inicia el objeto del sensor DHT para el manejo de las funciones.

void setup() {
dht.begin();//Se inicia el sensor DHT
}
```

Figura 19: Código de configuración del sensor DHT en el Arduino.

Primero se incluye la librería para que el programa pueda encontrar las diferentes funciones a utilizar. Se define el pin de entrada que leerá el arduino del sensor. A continuación, se define el tipo de sensor a utilizar. Esto es debido a que la librería puede utilizar los diferentes sensores de la serie DHT.

Una vez realizadas las diferentes definiciones, se llama al constructor del objeto para poder utilizar las diferentes funciones que contiene la librería. Este constructor tiene como parámetros las definiciones que se han declarado anteriormente y la velocidad de las cuentas del sensor. Éste último parámetro se deja con el valor seis ya que es el valor por defecto.

Cuando el constructor del objeto ya está definido, sólo queda iniciar el mismo dentro de la función de configuración inicial del Arduino.

Finalmente sólo se deben llamar a las funciones readHumidity() y readTemperature() para poder leer la humedad y la temperatura respectivamente.

5.8. Salidas del sistema

En este apartado se van a detallar cómo se han implementado las diferentes salidas para poder modificar las condiciones del microclima a regular.

5.8.1. Resistencias térmicas y ventilador

En este apartado se pretende dimensionar la resistencia total necesaria que sea capaz de disipar el suficiente calor para mantener la temperatura del habitáculo. Para ello se precisa obtener la potencia térmica a disipar. En este caso la potencia eléctrica y la potencia térmica van a ser la misma debido a que toda la potencia consumida por las resistencias va a ser repercutido en producción de calor.

Como las condiciones de cada planta van a ser diferentes, se va a suponer un caso medio para el dimensionado de la resistencia total necesaria. Suponiendo los siguientes datos:

- $T_{ambiente}$: 20 °C
- T_{Final}: 25 °C
- Tiempo de establecimiento: 20 minutos

Se va a obtener el calor disipado total a partir de la ecuación 5.8.1.1.

$$Q = \frac{m * c_e * \Delta T}{t} \qquad Ec\ 5.8.1.1$$

Siendo Q el calor disipado en Kilocalorías por hora (kcal/h), m la masa del aire del habitáculo en kilogramos (kg), c_e el calor específico en kilojulios por kilogramo Kelvin (kJ/kgK), ΔT el incremento de temperatura en grados Kelvin (K) y t el tiempo transcurrido en horas (h). Suponiendo que la masa del aire son 1.293 gramos y el calor específico del aire son 1012 kJ/kgK, se puede obtener el calor a disipar.

$$Q = \frac{m * c_e * \Delta T}{t}$$

$$Q = \frac{\frac{1.293}{1000} * 1012 * (298 - 293)}{\frac{20}{60}} = 19.62\ kcal/h$$

Una vez obtenido el calor disipado, sólo queda convertir esta magnitud a potencia eléctrica. Sabiendo que 1 kilovatio son 860 Kcal/h, se puede determinar mediante una simple regla de tres la potencia eléctrica a disipar.

$$\left.\begin{array}{r} 1\ kW - - - - - - - 860\ kcal/h \\ X\ kW - - - - - - - 19.62\ kcal/h \end{array}\right\} \quad X = 22.81 \cdot 10^{-3}\ kW \rightarrow 22.81\ W$$

Se usa el relé SRD-05Vdc-SL-C (14) (Songle Relay, China) para poder alimentar las resistencias en tensión alterna para minimizar el consumo debido a que la potencia a disipar es elevada y que el arduino no puede ofrecer una corriente elevada a sus salidas digitales.

Una vez obtenida la potencia total a disipar y especificado el uso de tensión alterna, se procede a obtener la corriente a consumir y a su vez la resistencia necesaria. Para ello se va a utilizar la fórmula de la potencia activa y la ley de Ohm para obtener la corriente consumida y la resistencia necesaria respectivamente. En la ecuación 5.8.1.2 se obtiene la corriente consumida.

$$P = V * I * \cos(\varphi) \qquad Ec\ 5.8.1.2$$

En este caso el factor de potencia es unitario ya que la resistencia no produce desfase alguno entre la tensión y la corriente.

$$P = V * I * \cos(\varphi)$$

$$22.81 = 230 * I * 1$$

$$I = 99.17 * 10^{-3}\ A \rightarrow 99.17\ mA$$

Con la corriente consumida ya obtenida, mediante la fórmula de la ley de Ohm, se puede obtener la resistencia necesaria. En la ecuación 5.8.1.3 se detalla la fórmula de la ley de Ohm.

$$R = \frac{V}{I} \qquad Ec\ 5.8.1.3$$

$$R = \frac{230}{99.17 * 10^{-3}} = 2310\ \Omega \rightarrow 2.31\ k\Omega \rightarrow 2.5\ k\Omega$$

Se elige 2.5 kΩ debido a que es el valor nominal más cercano al valor obtenido.

Debido a que se desea distribuir el calor por todo el habitáculo, se van a instalar dos resistencias a los extremos del habitáculo para mantener con mayor uniformidad el calor de la zona a calentar. Para ello se utilizan dos resistencias de 5 kΩ, conectadas en paralelo, para distribuir la potencia térmica a disipar en dos partes iguales y siendo la resistencia total la calculada mediante la ecuación 5.8.1.3. Por lo tanto la potencia a disipar por cada resistencia debe ser la mitad.

Finalmente, se instala un ventilador para refrigerar el ambiente cuando se sobrepase la temperatura de objetivo y poder reducir la temperatura de una forma más rápida y mantener el sistema mejor controlado. En la figura 20 se detalla el esquema final implementado.

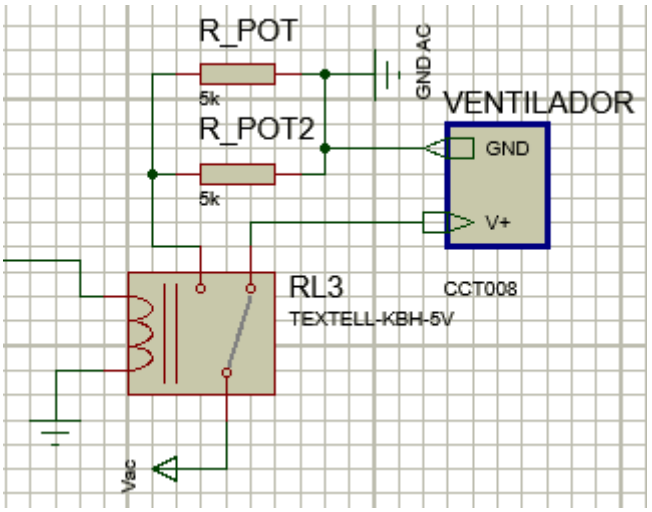


Figura 20: Esquema final de la salida para el control de la temperatura.

5.8.2. Electroválvula

Se va a utilizar la electroválvula ASC3 (15) (Emerson, Alemania) de tipo normalmente abierta para controlar la humedad del suelo. En este caso el control utilizado para el control de la humedad del suelo es del tipo todo o nada con histéresis ya que es una magnitud que no ofrece grandes variaciones en un gran intervalo de tiempo.

Debido a que el arduino a sus salidas digitales no puede ofrecer la corriente necesaria y, además, no puede ofrecer una señal senoidal para la electroválvula elegida, se debe utilizar un relé para activar y desactivar la misma. Este relé debe soportar la corriente que necesite la electroválvula para su funcionamiento nominal. Según las especificaciones, consume 220 mA. Por lo tanto se elige el relé SRD-05Vdc-SL-C que soporta una corriente máxima de 10 A, muy superior a la requerida, lo que asegura que ningún pico procedente de la señal de la red eléctrica pueda dañar el relé. Además, al usar el relé, se aísla la parte de control de la parte de maniobra.

El conexionado de la electroválvula se puede apreciar en la figura 21.

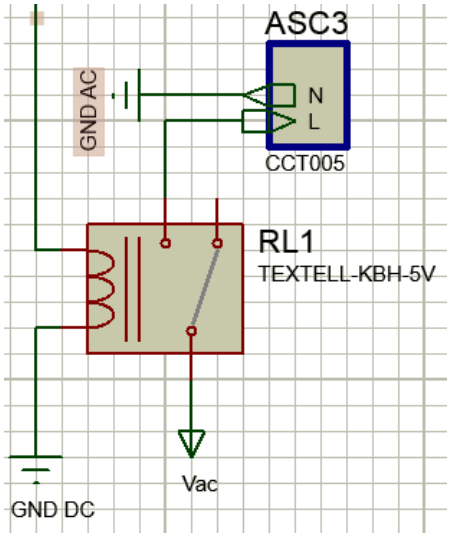


Figura 21: Conexionado del conjunto relé y electroválvula.

5.8.3. Humidificador

Se va a utilizar un pequeño humidificador de tipo usb (16) para mantener la humedad relativa del microclima. Este humidificador necesita un pequeño depósito de agua para que pueda soltar agua en estado gaseoso. Dadas las especificaciones del dispositivo, el humidificador consume 300 ml por cada hora, por lo que se diseñará un depósito con una capacidad superior para no tener que estar aportando agua al mismo cada periodo de tiempo corto. Siguiendo con las especificaciones, este dispositivo consume 300 mA, una corriente muy superior a la que puede aportar una salida digital del Arduino. Para ello se va a necesitar un relé que active el humidificador para que éste pueda ser alimentado desde una fuente de alimentación externa. Por lo tanto, se elige el mismo relé que en los apartados 5.8.2 y 5.8.1, el relé SRD-05Vdc-SL-C que puede soportar el paso de la corriente que necesita el humidificador para su correcto funcionamiento. En la figura 22 se puede apreciar el resultado del circuito diseñado para la salida del control de la humedad relativa del aire.

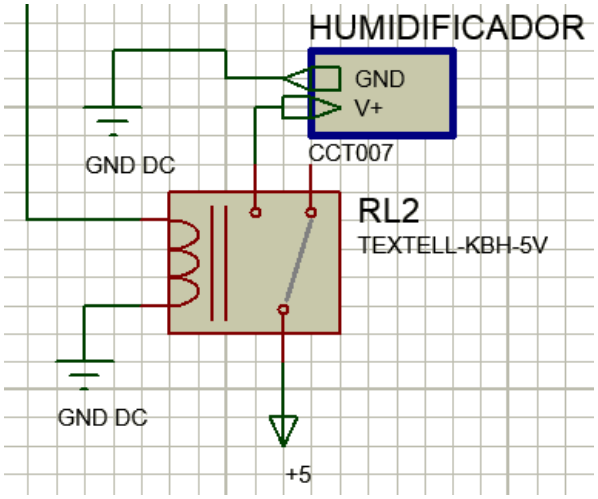


Figura 22: Diseño del circuito que alimenta el humidificador.

5.8.4. Placas de LEDS

Se van a utilizar una serie de placas led de color blanco cálido (17). Estas placas en su interior llevan 24 LEDs que funcionan a 12 V con una potencia de disipación de 3 W. Como se va a regular la cantidad de iluminación del habitáculo es necesario regular el paso de corriente por las placas LED.

Para ello se decide utilizar el transistor bd243C (18) (Fairchild semiconductor, Estados Unidos) para poder regular la corriente y para poder ofrecer la corriente que la salida PWM del Arduino no puede ofrecer. El primer paso es tratar de regular la corriente de base para que al máximo ancho de pulso (100%) sature el transistor. A partir del datasheet del transistor se obtiene que tiene una beta de 100 y una tensión base emisor de 0.6 V y, a partir de las especificaciones de los LEDs, se obtiene que la corriente de colector es de 220 mA. En la Ecuación 5.8.4.1 se tiene el cálculo necesario.

$$I_c = \beta * I_b \qquad Ec\ 5.8.4.1$$

$$0.220 = 100 * I_b$$

$$I_b = 1.5\ mA$$

Una vez obtenida la corriente necesaria para saturar el transistor, se tiene que obtener el valor de la resistencia necesaria en la base para poder regular la corriente del colector y tener la regulación por corriente. Para ello se analiza la parte base-emisor, sabiendo que la tensión en la base para saturar el transistor va a ser 5 V. En la ecuación 5.8.4.2 se detalla dicho cálculo.

$$V_b = R_b * I_b + V_{be} \qquad Ec\ 5.8.4.2$$

$$5 = R_b * 1.5 \cdot 10^{-3} + 0.6$$

$$R_b = 2.93\ k\Omega \rightarrow 3.3\ k\Omega$$

Debido a que la frecuencia del PWM es demasiado lenta y la velocidad de captura de la iluminación del fotodiodo empleado es muy rápida, éste último capta los anchos de pulso de estado bajo de la señal PWM y no se puede regular el sistema de iluminación de forma correcta. Para subsanar este error se utiliza el integrado 74HC04 (19) (NXP, Países Bajos) que permite invertir la señal para obtener siempre el estado opuesto al PWM y que el fotodiodo no capte los anchos de pulso bajo. De esta forma, dos placas de diodos LED tienen el ancho de pulso correspondiente a la salida PWM del arduino, y las otras dos placas de diodos LED restantes, tienen su inversa. En la figura 23 se tiene el resultado del circuito implementado.

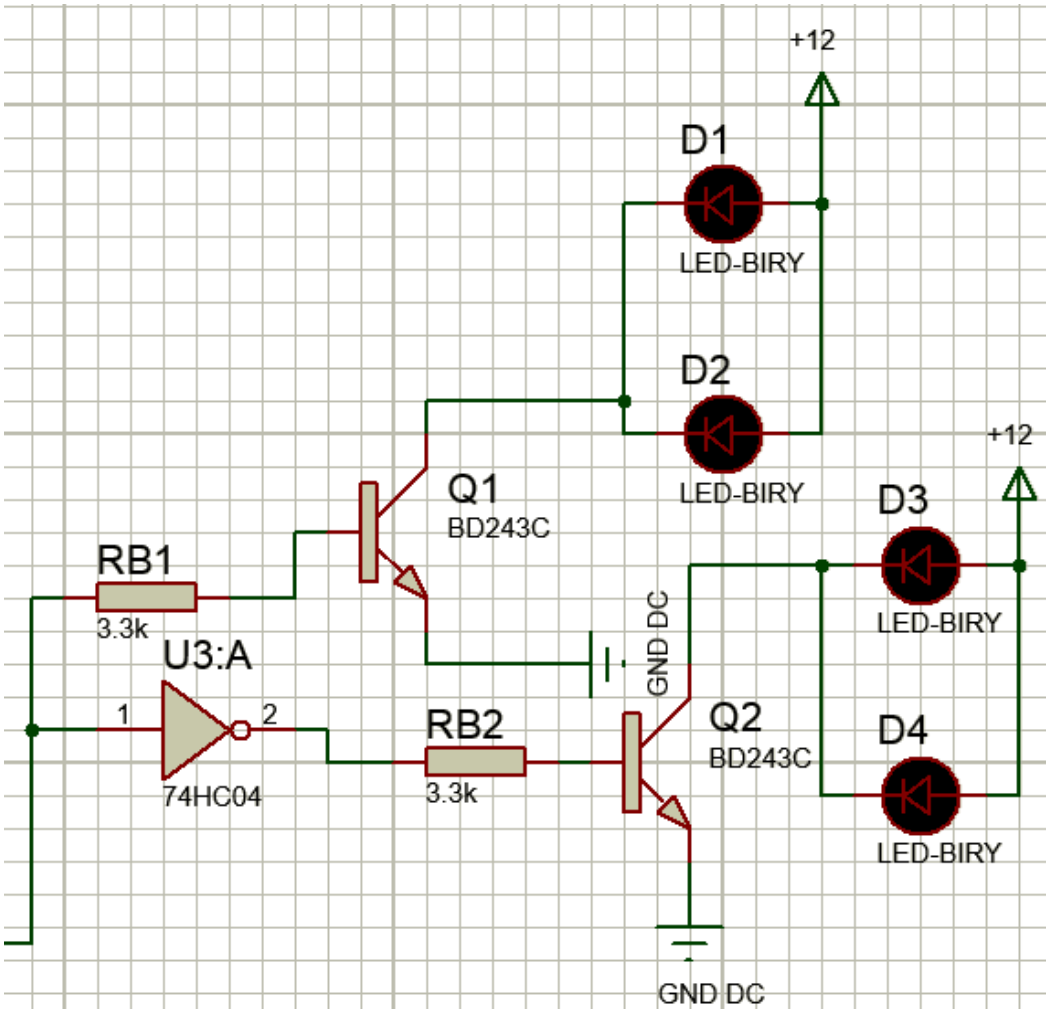


Figura 23: Circuito obtenido para el control de la corriente de las placas LED.

5.9. Tipos de control de las magnitudes

En este apartado se van a explicar cómo se han planteado los diferentes controles que se van a utilizar para el control de las diferentes magnitudes que se van a tratar en el sistema.

5.9.1. Todo o nada con histéresis

Para controlar la temperatura y las humedades del suelo y del aire se ha decidido utilizar un control todo o nada con histéresis debido a que estas magnitudes son de respuesta lenta, es decir, para apreciar una variación grande en estas magnitudes debe pasar un periodo de tiempo largo.

Para ello se fijan dos valores de referencia a alcanzar, uno para la referencia superior y otro para la referencia inferior. En este caso se ajusta en la base de datos el valor de la referencia superior y se fija la referencia inferior en dos unidades menos para intentar obtener una respuesta prácticamente lineal.

Al utilizar este tipo de control la acción sobre el correspondiente actuador es de tipo interruptor, es decir, cuando el control debe activar la salida, lo hace siempre al máximo posible, y cuando se debe apagar la salida, lo hace por completo. Cuando se esté por debajo de la referencia inferior, se activará el actuador y, en el momento que se sobrepase la referencia superior, se apaga por completo el actuador. Esto provoca unas pequeñas sobreoscilaciones debido a que no hay una regulación de la salida conforme el valor actual de la magnitud se acerca a cualquiera de las diferentes referencias existentes. En la figura 24 se puede apreciar una pequeña explicación gráfica de todo lo mencionado anteriormente.

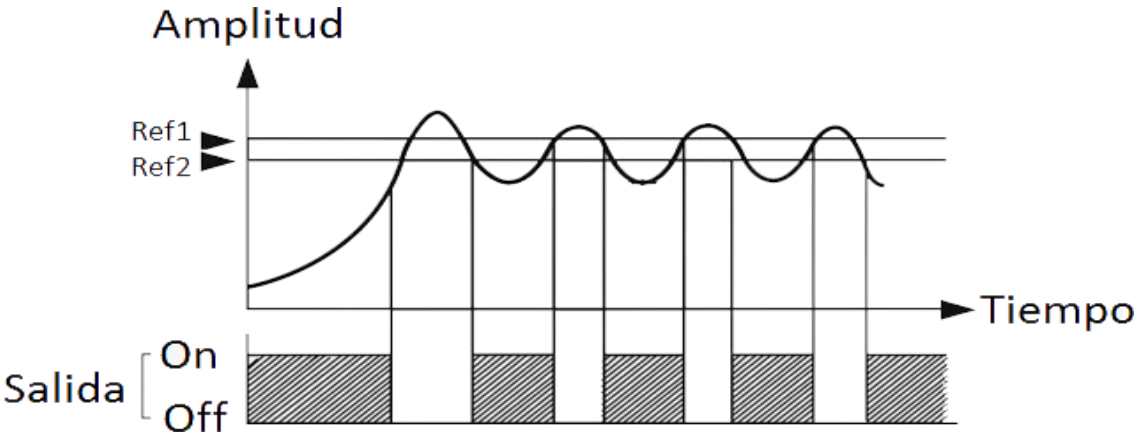


Figura 24: Representación de una salida con un control todo o nada con histéresis.

5.9.2. PID

En este apartado se va a explicar el proceso para obtener el regulador PID necesario para poder regular la iluminación del habitáculo ya que esta magnitud es de acción rápida. Como primera parte se va a explicar la obtención del modelo necesario para poder diseñar el regulador necesario.

5.9.2.1. Obtención del modelo de las placas LED

Para obtener el modelo se implementa en Matlab un fichero .m para capturar los datos obtenidos en el Arduino. Además se implementa un pequeño código en Arduino para generar un escalón para ver cómo reacciona el fotodiodo y poder obtener esta reacción en Matlab y

poder obtener el correspondiente modelo. En la figura 25 se tiene el código de Arduino que genera el escalón mencionado.

```
const int analogOutPin = 45; // Analog output pin
int lectura_analog=0;         //valor lectura pin analógico
byte outputValue = 125;        // valor del PWM para el pulso
int x[4]={0,0,0,0}; //vector del filtro de entrada de los últimos 3 valores
int y[4]={0,0,0,0}; //vector a la salida del filtro de los últimos 3 valores
int z[4]={0,0,0,0}; //vector a la salida del segundo filtro de los últimos 3 valores
int a[4]={0,0,0,0};//vector a la salida del tercer filtro
int e[2]={0,0};//vector del error pid
int s[2]={0,0};//vector salida pid
int k=0;
int salida_filtro=0;
void setup() {
  Serial.begin(9600);       //Inicializar la comunicación serie pc- arduino
}

void loop() {
  analogWrite(analogOutPin, outputValue); //escritura pwm en el pin
  analogWrite(analogOutPin, (255-outputValue));
  //delay(1000);
  x[k]=analogRead(A10);         //lectura analógica
  filtro(x,y);
  filtro(y,z);
  filtro(z,a);
  Serial.println(a[0]);         //envío por el puerto com del arduino(USB)
  outputValue=pid(a[0],1000);
}
void filtro(int entrada[], int salida[])
{
  salida[k]=0.0753*entrada[k]+0.2259*entrada[k+1]+0.2259*entrada[k+2]+0.0753*entrada[k+3]_
  +0.8266*salida[k+1]-0.5154*salida[k+2]+0.08648*salida[k+3]; //ecuacion en diferencias del filtro de 3 er orden
  salida[k+3]=salida[k+2];
  salida[k+2]=salida[k+1];
  salida[k+1]=salida[k];
  entrada[k+3]=entrada[k+2];
  entrada[k+2]=entrada[k+1];
  entrada[k+1]=entrada[k];
}
```

Figura 25: Código de Arduino que genera el escalón necesario para obtener el modelo necesario.

Este código contiene la inicialización de las variables necesarias y la función que inicializa el puerto serie para poder transmitir los datos obtenidos al fichero de Matlab y poder obtener el modelo. En la función sin fin del sistema, se escribe la salida PWM del escalón y su inversa a continuación para mantener el escalón de forma constante. A continuación se lee la entrada analógica que corresponde al fotodiodo y se llama a la función del filtro paso bajo tres veces para evitar que se note el ancho de pulso bajo en la captura del modelo. Finalmente se tiene la función que implementa el filtro paso bajo mencionado en el apartado 5.6. Ésta contiene la ecuación en diferencias que corresponde al filtro y la actualización de valores de entrada y de salida.

A continuación se va a explicar el funcionamiento del fichero .m de Matlab el cual permite guardar los datos transferidos por el puerto serie del Arduino en una variable que a su vez se transportarán estos datos a un fichero de texto el cual se podrá obtener el modelo. En la figura 26 se tiene el correspondiente fichero .m.

```
delete(instrfind({'Port'},{'COM3'}));
%crear objeto serie
s = serial('COM3','BaudRate',9600,'Terminator','CR/LF');
warning('off','MATLAB:serial:fscanf:unsuccessfulRead');
%abrir puerto
fopen(s);% parámetros de medidas
tmax = 7; % tiempo de captura en s
rate = 5; % resultado experimental (comprobar)
% preparar la figura
f = figure('Name','Captura');
a = axes('XLim',[0 tmax],'YLim',[0 1024.1]);
l1 = line(nan,nan,'Color','r','LineWidth',2);

xlabel('Tiempo (s)')
ylabel('AI10')
title('Captura del porcentaje en tiempo real con Arduino')
grid on
hold on
% inicializar
v1 = zeros(1,tmax*rate);
i = 1;
t = 0;
temp= zeros(1,tmax*rate);
% ejecutar bucle cronometrado
tic
while t<tmax
    t = toc;
    % leer del puerto serie
    a = fscanf(s,'%f,%f')';
    a1=a;
    v1(i)=a1(1);
    temp(i)=t(1);
    % dibujar en la figura
    x = linspace(0,i/rate,i);
    set(l1,'YData',v1(1:i),'XData',x);
    drawnow
    % seguir
    i = i+1;
end
% resultado del cronometro
clc;
fprintf('%g s de captura a %g cap/s \n',t,i/t);
%% Limpiar la escena del crimen
fclose(s);
delete(s);
clear s;
```

Figura 26: Código del fichero .m que captura los datos procedentes del Arduino.

La primera parte de código configura el puerto serie, lo abre para poder capturar los datos procedentes del Arduino y se configuran las diferentes variables que gestionan la velocidad de

captura, el tiempo de captura, las variables en la que se van a guardar los datos y la configuración de la ventana en la que se mostrará el gráfico de captura. Después se tiene un bucle while que conforma el gráfico a partir de cómo se van leyendo los datos y, a su salida cuando se ha terminado el tiempo de graficado, se muestra la gráfica y la velocidad de captura y se cierra todo.

Para la captura del modelo, se determina un valor determinado del escalón en el código de arduino y se ejecuta el fichero .m de Matlab obteniendo un sistema como el de la figura 27.

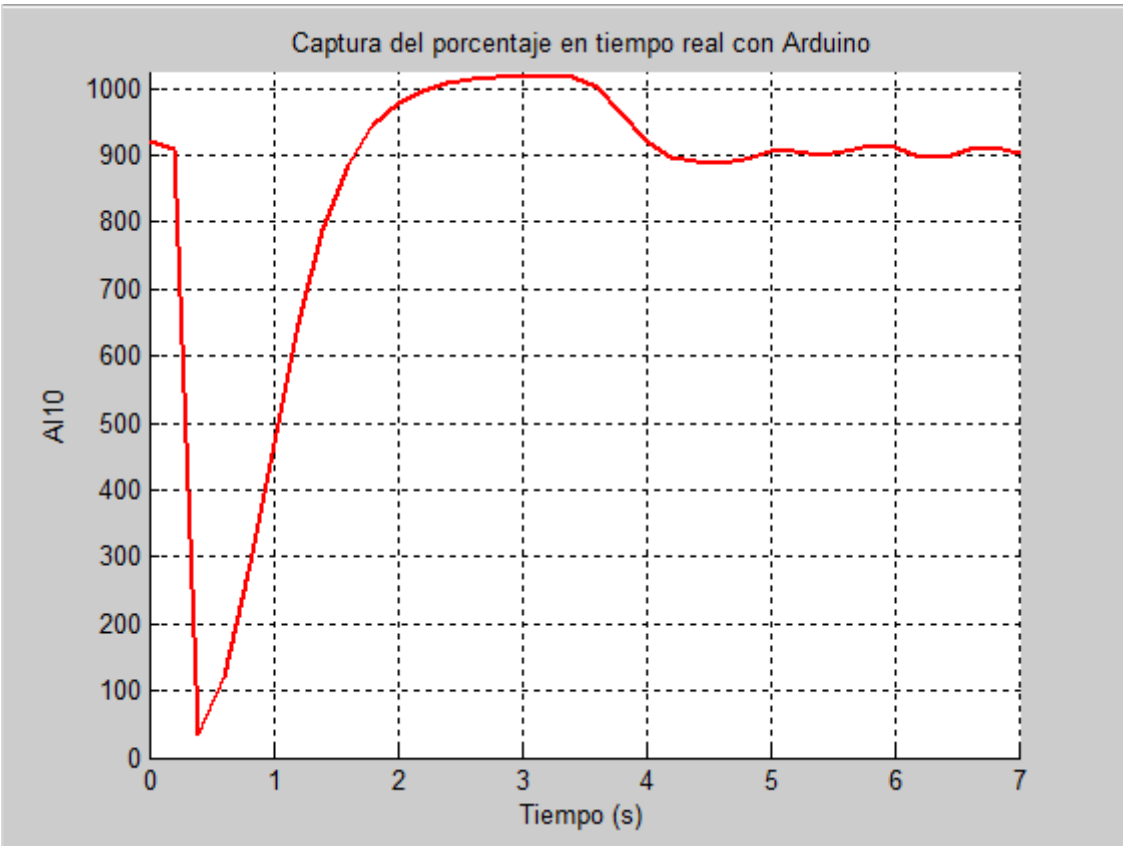


Figura 27: Gráfica resultante de la captura del modelo con el fichero. m de Matlab.

Se puede percibir que la primera caída de valor corresponde a que el sistema en arduino ya está funcionando y se hace un reset al activar la captura de datos sufriendo un pequeño retraso. A partir de este reset se considera correcta la captura de datos en la que se puede percibir claramente que el sistema tiene una pequeña sobreoscilación antes de estabilizarse teniendo una ligera variación. Esta ligera variación corresponde a lo mencionado en el apartado 5.8.4.

Una vez obtenidos los datos, se guardarán en un fichero de texto para poder utilizarlos en el siguiente archivo .m de Matlab en el que se ejecuta la herramienta que obtiene el modelo.

Este fichero tiene dos partes, una en la que se obtiene el modelo y la otra en la que se obtiene el regulador PID. En la figura 28 se tiene la parte de código correspondiente a la captura del modelo.

```
load grafica.txt
%grafica
temp=grafica(:,1); % definimos la columna de tiempo
entrada=grafica(:,2); % definimos la columna de entrada
salida=grafica(:,3);
%ploteamos
subplot(211),plot(temp,entrada)
subplot(212),plot(temp,salida)
incr_salida=salida-0;    %el comando iden va por incrementos sobre 0
incr_entrada=entrada-0;
% comando de identificacion
ident;
%modelo capturado a partir del comando ident
s=tf('s');
kp=-0.00013869;
Tp1=399.65;
Tp2=8.1472;
Tp3=8.287;
Tz=33151;
g=kp*((1+Tz*s)/(s*(1+Tp1*s)*(1+Tp2*s)*(1+Tp3*s)));
```

Figura 28: Código correspondiente a la obtención del modelo.

La primera parte carga el fichero y se guardan las diferentes columnas del mismo en diferentes variables y se plotean en una figura para comprobar si la captura está correcta. A continuación se ejecuta el comando *ident* que permite calcular el modelo obtenido y se guardan las variables resultantes y la función correspondiente al modelo.

Cuando se ejecuta el comando *ident* aparece una ventana como la de la figura 29, la cual se importan los datos y se busca el modelo resultante más aproximado.

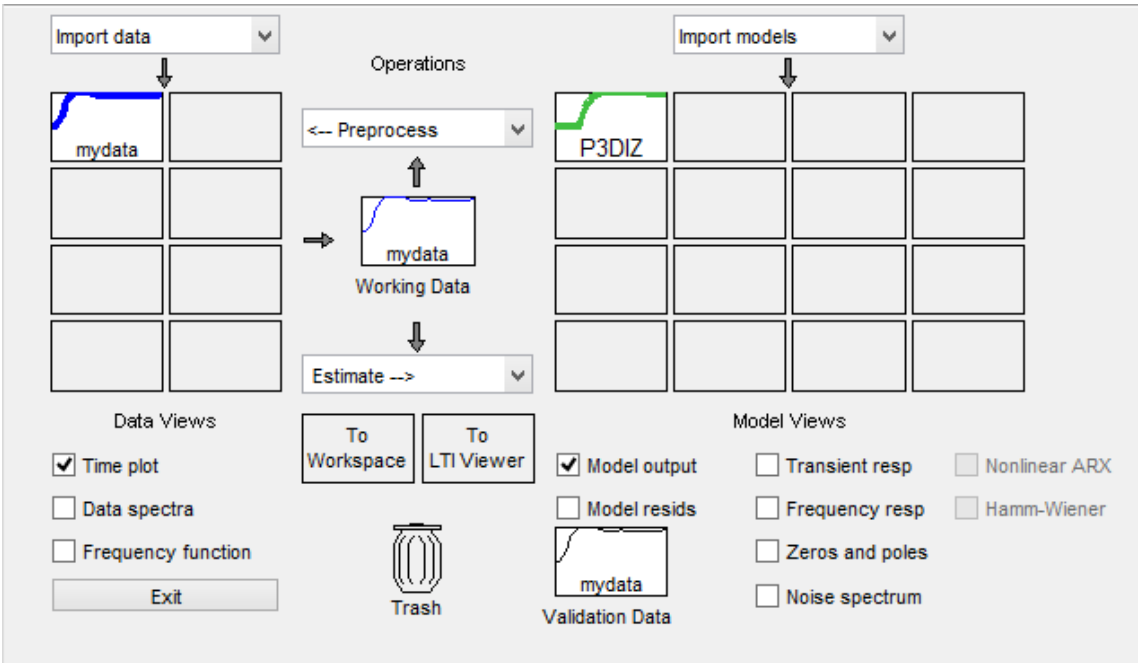


Figura 29: Pantalla del comando de obtención del modelo.

A partir de los datos se estima el modelo más aproximado. Para este caso se obtiene que el modelo más aproximado sea el de un sistema de 3 polos, un integrador y un zero. En la figura 30 se muestran los datos correspondientes de la función correspondiente al modelo obtenido.

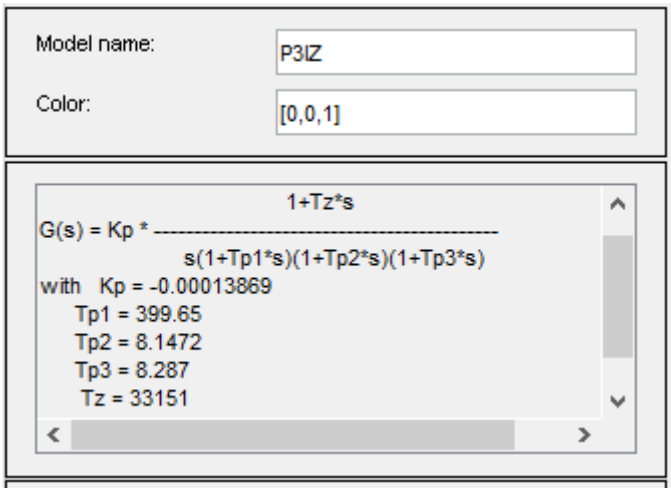


Figura 30: Captura de la determinación de la función del modelo obtenido.

Además se puede mostrar la respuesta ante un escalón idéntico al utilizado para comparar los datos capturados con la función del modelo aproximado. Esto sirve para determinar qué configuración de polos y zeros es la más aproximada al modelo real. En la figura 31 se dispone del gráfico obtenido con el modelo aproximado elegido.

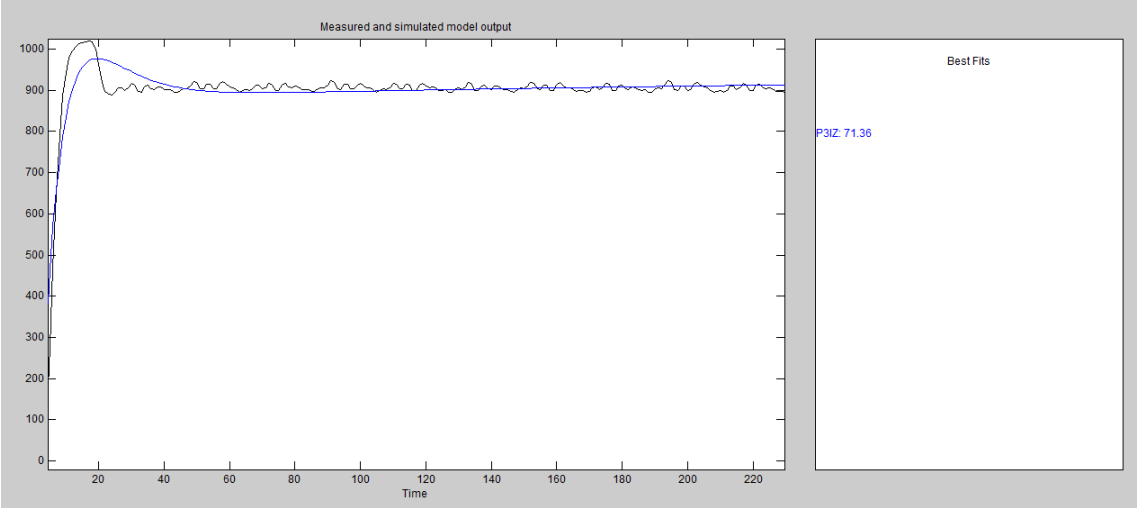


Figura 31: Gráfica en la que se comparan los datos obtenidos con el modelo aproximado y su porcentaje de similitud.

La similitud del modelo aproximado del real es de un 71.36%. Es un valor bastante elevado y que supone una gran certeza de proximidad, por lo que el modelo aproximado se puede considerar válido.

Finalmente la función de transferencia del modelo resultante obtenido es la siguiente.

$$G(s) = 0.00013869 * \frac{1 + 33151 * s}{s * (1 + 339.65 * s) * (1 + 8.1472 * s) * (1 + 8.287 * s)}$$

5.9.2.2. Obtención del regulador PID

Para obtener el regulador PID necesario se debe utilizar la herramienta de *sisotool* de Matlab que permite ver el lugar de las raíces de la planta obtenida y poder implementar un regulador, así como discretizarlo para obtener una ecuación en diferencias que pueda ser introducida en el Arduino.

Como primera parte se determinan las siguientes especificaciones:

- Tiempo de establecimiento inferior a 38 segundos.
- Sobreoscilación inferior al 5%.

Una vez determinadas las especificaciones, toca obtener el regulador con la herramienta *sisotool*. Para ello se tiene la otra parte del código del fichero .m mencionado en el apartado 5.9.1.1. En la figura 32 se tiene esta última parte.

```
%Diseño del PID
sisotool(g);
%PID obtenido
Gpid=0.072337*(s/(s+0.00862));
%PID discreto obtenido
Zpid=0.072305*((z-1)/(z-0.999));
```

Figura 32: Parte de código en la que se obtiene el regulador PID deseado.

Esta última parte tiene la llamada a la herramienta *sisotool* con respecto a la función del modelo obtenido y la anotación del PID en continuo obtenido y su PID discreto.

Dentro del *sisotool* se añaden los diferentes polos y ceros que van a modificar el lugar de las raíces para entrar dentro de las especificaciones deseadas. En la figura 33 se tiene el lugar de las raíces obtenido a partir de añadir polos y ceros.

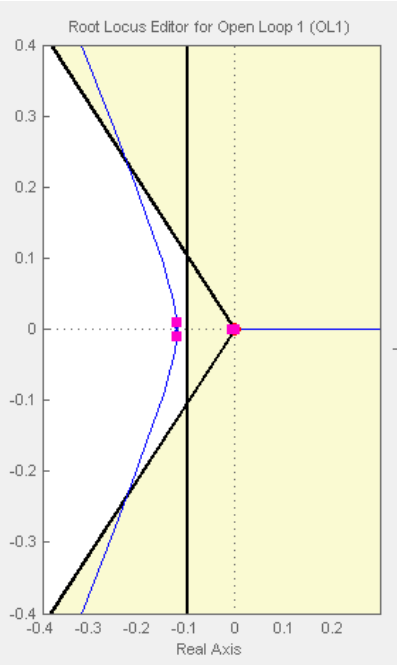


Figura 33: Lugar de las raíces del modelo con el PID en continuo.

Con el que se obtiene la siguiente función de transferencia del regulador PID.

$$Gpid(s) = 0.072337 * \frac{s}{s + 0.00862}$$

El siguiente paso es discretizar este regulador para poder obtener la ecuación en diferencias necesaria para poder pasarlo al código del Arduino. En la figura 34 se tiene el método de discretizado utilizado, así como su tiempo de muestreo.

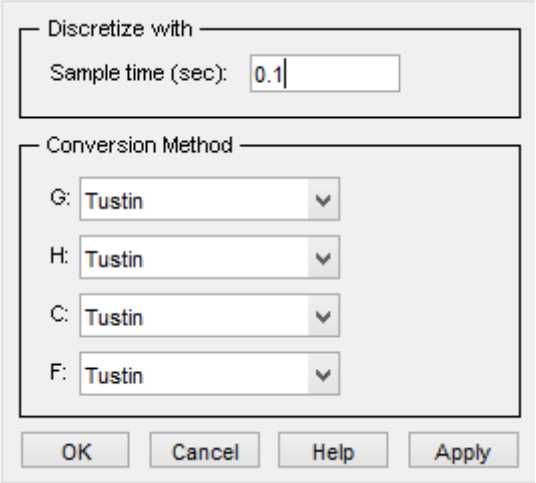


Figura 34: Captura del discretizado del regulador PID.

Una vez discretizado el regulador se obtiene la siguiente función de transferencia discreta.

$$G(z) = 0 * 072305 * \frac{z - 1}{z - 0.999}$$

A partir de esta función de transferencia se puede obtener la ecuación en diferencias del mismo método que en el apartado 5.6.

$$\frac{y(z)}{x(z)} = 0.072305 * \frac{z - 1}{z - 0.999}$$

$$\frac{y(z)}{x(z)} = \frac{0.072305 * z - 0.082305}{z - 0.999}$$

$$y(z) * z - y(z) * 0.999 = 0.072305 * z * x(z) - 0.072305 * x(z)$$

$$y(k + 1) - 0.999 * y(k) = 0.072305 * x(k + 1) - 0.072305 * x(k)$$

$$y(k) - 0.999 * y(k - 1) = 0.072305 * x(k) - 0.072305 * x(k - 1)$$

$$y(k) = 0.072305 * x(k) - 0.072305 * x(k - 1) + 0.999 * y(k - 1)$$

Con la ecuación en diferencias ya obtenida sólo queda implementarla en el código del arduino dentro de una función llamada PID. En la figura 35 se dispone de esta función implementada.

```
int pid (int entrada, int referencia)
{
 int sal_255=0;
 e[k]=referencia-entrada;
 s[k]=0.072305*e[k]-0.072305*e[k+1]+0.999*s[k+1];
 s[k+1]=s[k];
 e[k+1]=e[k];
 sal_255=(s[k]*255)/1024;
 Serial.println(s[k]);
 return sal_255;
}
```

Figura 35: Función que implementa el regulador PID diseñado.

A esta función se le debe introducir la referencia deseada y la entrada, siendo ésta última la salida del filtro digital implementado. A partir de la entrada y la referencia, se obtiene el error actual del sistema. Este error es el que se utiliza para obtener la salida en la ecuación en diferencias y obtener la salida regulada. Finalmente se actualizan los valores para la nueva entrada a la función y se escala la salida para la salida PWM.

5.10. Aplicación Visual Basic

En este apartado se van a detallar cómo se ha implementado la aplicación de Visual Basic que permite la comunicación entre la base de datos y el Arduino, así como la monitorización del sistema y el tratamiento de la base de datos ubicada en el mismo equipo informático.

5.10.1. Configuración inicial

Para poder utilizar todos los componentes como el acceso a la base de datos y poder utilizar las funciones que gestionan la conexión y la llamada a la misma, se deben importar una serie de "librerías" o en este caso, referencias. Para ello, se debe acceder a las propiedades del proyecto creado y en la pestaña de referencias se agregan las referencias de *MySql.Data* y *Microsoft.VisualBasic.PowerPacks*. En la figura 36 se muestran las referencias agregadas al proyecto.

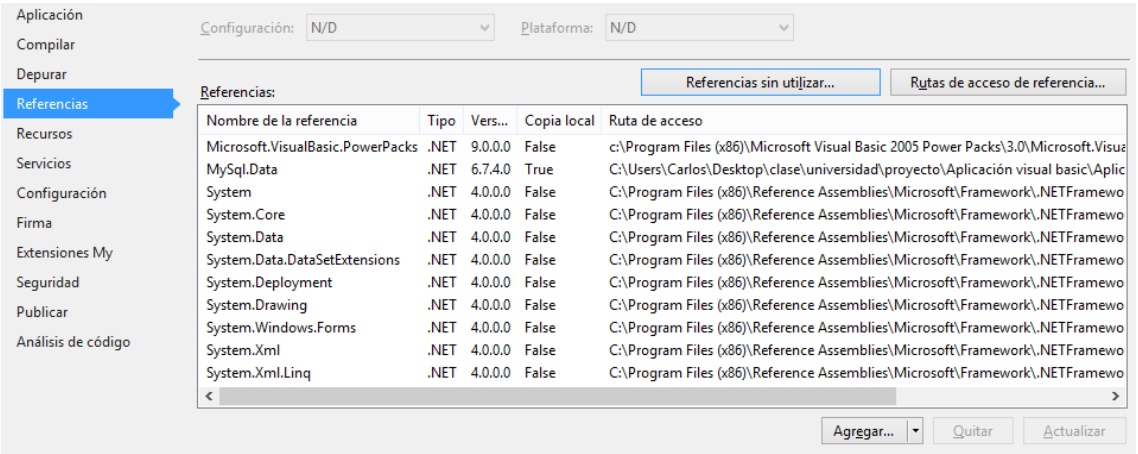


Figura 36: Lista de las diferentes referencias agregadas a la aplicación de Visual Basic.

Una vez añadidas las referencias necesarias, se crearán dos módulos para albergar las diferentes variables públicas y diferentes funciones para que se puedan utilizar en cualquier

formulario de la aplicación. Estos módulos se llamarán Funciones base de datos y Variables. El primero, como su nombre indica, contendrá todas las funciones respectivas al tratamiento de la base de datos y el segundo, contiene las variables de acceso público y funciones de propósito general.

Con los módulos creados, se va a especificar cómo van a ser los diferentes formularios o pantallas que contiene la aplicación. En la figura 37 se muestran la configuración de los diferentes parámetros que van a tener los formularios.

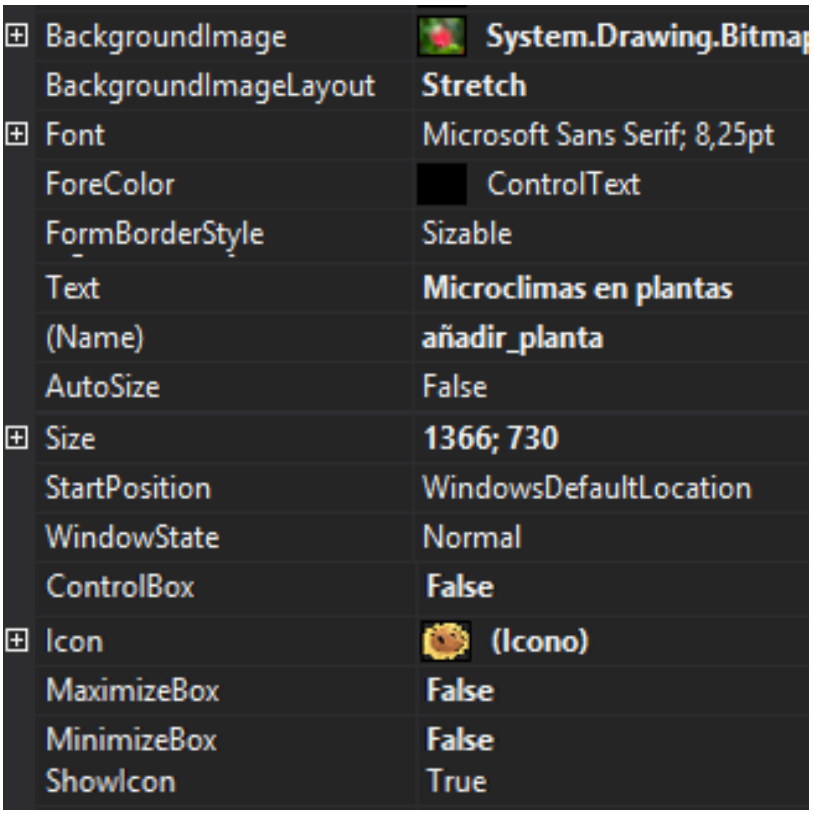


Figura 37: Propiedades en común para todos los formularios existentes.

En la tabla 7 se muestra la explicación a cada parámetro mostrado en la figura 24.

Tabla 7: Tabla con las explicaciones a cada parámetro utilizado en los formularios.	
Parámetro	Definición
BackgroundImage	Permite la colocación de una imagen de fondo del formulario.
BackgroundImageLayout	Se define el formato de la imagen. Se pone en Stretch para ajustar la imagen a la zona visible.
Font	Se configura la fuente del título de la pantalla.
ForeColor	Color que tendrá el texto del título de la aplicación
FormBorderStyle	Formato de la barra del título de la aplicación. Se ajusta a Sizable para que no tenga el icono de cerrar.
Text	Texto que se ubicará en el título de la aplicación.
(Name)	Nombre para referenciar al formulario en el código.
AutoSize	Permite el autoajuste de la ventana. Se ajusta en False para que no se ajuste automáticamente.
Size	Se ajusta el tamaño de la ventana. En este caso se ajusta a la resolución de la pantalla.
StartPosition	Posición en la que se va a iniciar el formulario. Se deja por defecto ya que el tamaño del formulario es la resolución de la pantalla.
WindowState	Formato en el que se va a mostrar el formulario. Se deja en por defecto para que no aparezca minimizado o maximizado.
ControlBox	Muestra la barra de control. Se deja en false para que no aparezca dicha barra.
Icon	Muestra el icono de la aplicación.
MaximizeBox	Permite visualizar el icono de maximizar. Se configura en False para que no aparezca.
MinimizeBox	Permite visualizar el icono de minimizar. Se configura en False para que no aparezca.
ShowIcon	Muestra el icono de la aplicación en la barra del título de la aplicación. Se configura en True para que se muestre dicho icono.

5.10.2. Apariencia

La aplicación dispone de un total de 6 pantallas en las que se podrá interactuar de diferentes formas para poder tratar específicamente cada parte por separado.

La primera pantalla que aparece una vez se abre la aplicación es la pantalla principal. Esta pantalla es el menú por el cual se accederán a todas las partes que forma esta aplicación. En la figura 38 se puede apreciar la pantalla principal de la aplicación implementada con Visual Basic.

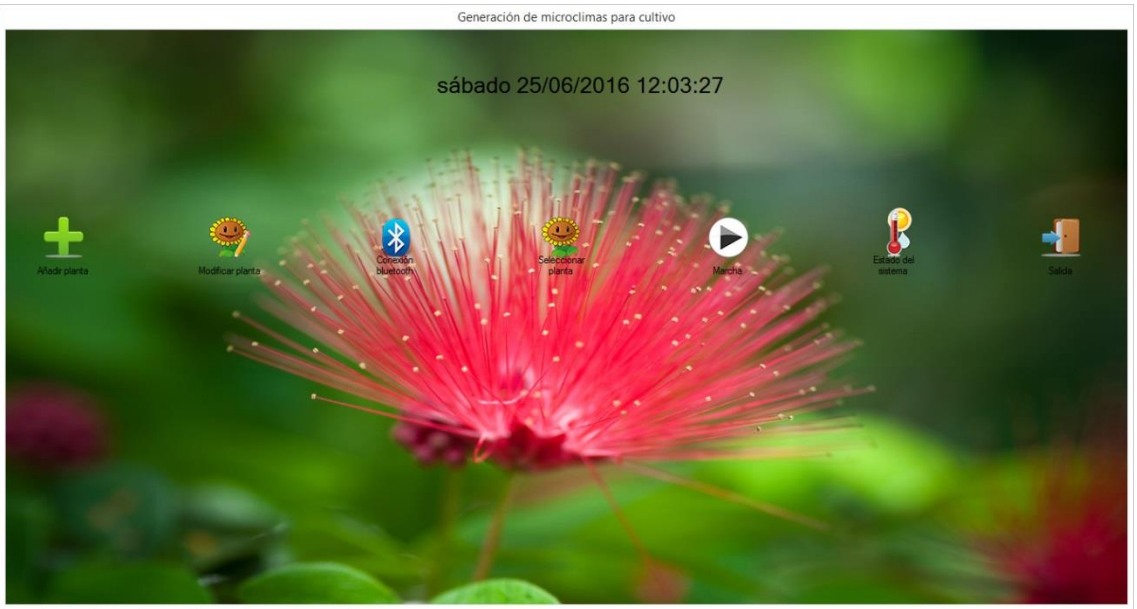


Figura 38: Apariencia de la pantalla principal.

Esta pantalla dispone de un pequeño cuadro de texto que muestra la fecha completa actual actualizada a cada segundo con el formato día de la semana, fecha y hora. En la tabla 8 se explica el funcionamiento de cada botón.

Tabla 8: Tabla con la descripción de cada botón de la pantalla principal.

Elemento	Descripción
 Añadir planta	Se accede a la pantalla que permite agregar una nueva planta a la base de datos.
 Modificar planta	Accede a la pantalla para modificar los datos de una planta agregada a la base de datos.
 Conexión bluetooth	Permite acceder a la pantalla que configura la conexión del módulo bluetooth bluetooth.
 Seleccionar planta	Se accede a la pantalla para seleccionar la planta que se desea controlar para transferir los datos al habitáculo.
 Marcha	Permite inicializar y parar el control del sistema de una forma simple y rápida. Esto sirve en el caso de que el sistema ya tiene datos transferidos.

 Paro	
 Estado del sistema	Accede a la pantalla de monitorización del estado del sistema.
 Salida	Permite salir de la aplicación. Al pulsar el botón muestra un mensaje para certificar la salida de la misma.

El funcionamiento de esta pantalla está previsto para que se realice un acceso de izquierda a derecha, es decir, que para realizar un funcionamiento correcto y completo se debe acceder a las diferentes pantallas pulsando en los botones de izquierda a derecha. De la misma forma se van a explicar las diferentes pantallas a las que se puede acceder desde la pantalla principal.

Al acceder a la pantalla de añadir planta se va a poder colocar los datos necesarios para el control de una planta en concreto y transferir los datos a la base de datos que contiene el equipo informático. En la figura 39 se aprecia la apariencia de esta pantalla.

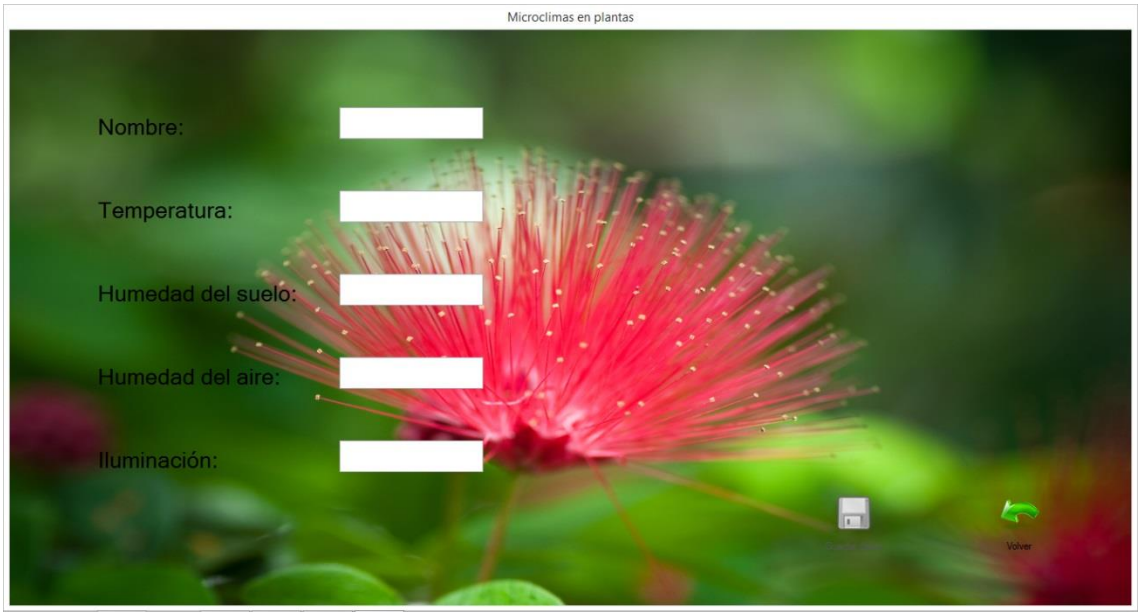


Figura 39: Apariencia de la pantalla de añadir una nueva planta a la base de datos.

Esta pantalla dispone de una serie de cuadros de texto en los que se van a introducir los datos correspondientes a la planta y se podrán guardar, una vez estén correctamente completados los campos de texto, en la base de datos. En la tabla 9 se explica el funcionamiento de cada botón.

Tabla 9: Tabla con la descripción de cada botón de la pantalla de añadir una nueva planta.	
Elemento	Descripción
 Guardar	Botón que permite guardar los datos introducidos de la planta deseada a la base de datos. Éste botón sólo está activo cuando los datos introducidos en los campos de texto son del formato adecuado.
 Volver	Permite volver a la pantalla principal.

En la pantalla de modificar planta se puede modificar los datos de una planta ya existente en la base de datos en caso de error o de querer actualizar los datos y volver a transmitir estos nuevos datos en la base de datos. En la figura 40 se puede apreciar la apariencia de dicha pantalla.

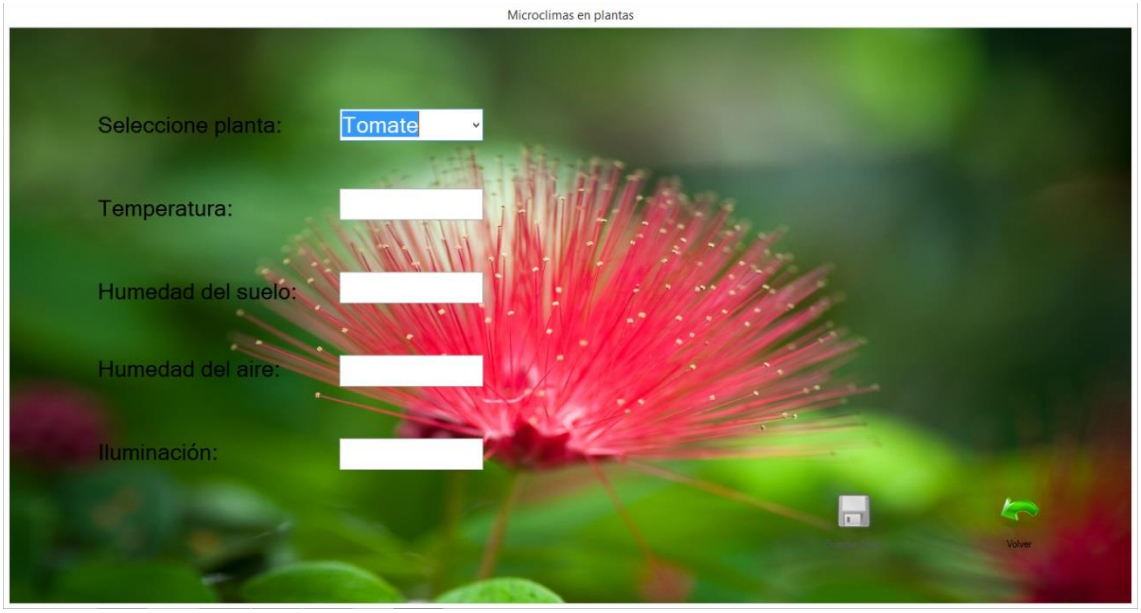


Figura 40: Apariencia de la pantalla de modificar los datos de una planta ya guardada.

En esta pantalla primero sólo aparece un cuadro desplegable para seleccionar la planta que se desee modificar sus datos. Cuando se seleccione una, aparecerán los respectivos campos de texto para completar los datos y poder transferirlos a la base de datos. En la tabla 10 se explica el funcionamiento de cada botón de esta pantalla.

Tabla 10: Tabla con la descripción de cada botón de la pantalla de modificar datos.	
Elemento	Descripción
 Guardar	Botón que permite guardar los datos introducidos de la planta deseada a la base de datos. Éste botón sólo está activo cuando los datos introducidos en los campos de texto son del formato adecuado.
 Volver	Permite volver a la pantalla principal.

En la pantalla de conexión bluetooth se va a configurar el puerto serie asociado al módulo bluetooth enlazado como se ha explicado en el apartado 5.4.1 y se procederá a conectarse al módulo para poder enviar y recibir datos. En la figura 41 se puede apreciar la apariencia de esta pantalla.

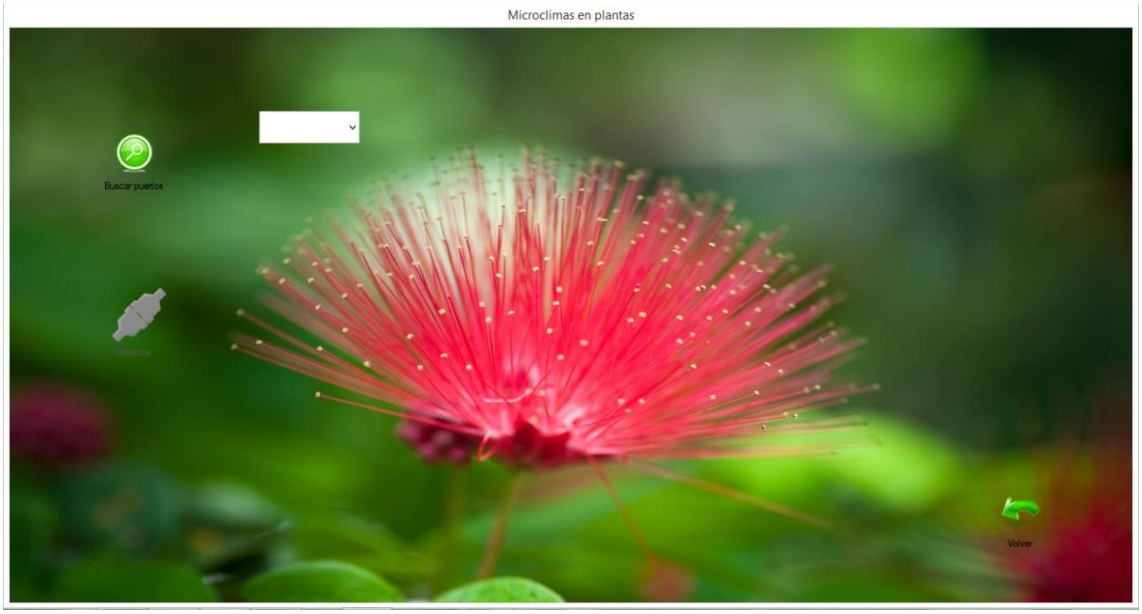
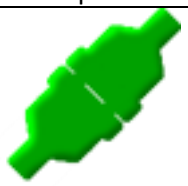


Figura 41: Apariencia de la pantalla para configurar la conexión bluetooth.

En esta pantalla se tiene un cuadro desplegable en el que se selecciona el puerto serie correspondiente al módulo bluetooth asociado y, una vez seleccionado dicho puerto, se puede conectar y poder mantener una comunicación con el Arduino. En la tabla 11 se explica el funcionamiento de cada botón de la pantalla.

Tabla 11: Tabla con la descripción de cada botón de la pantalla de conexión por bluetooth.	
Elemento	Descripción
 Buscar puertos	Permite buscar todos los puertos serie disponibles en el equipo informático.
 Conectar  Desconectar	Permite conectar y desconectar con el módulo bluetooth a partir del puerto serie seleccionado en el desplegable.
 Volver	Botón que permite volver a la pantalla principal una vez conectado o desconectado el módulo bluetooth.

En la pantalla de seleccionar la planta que se tiene en el habitáculo para transferir los datos de las diferentes magnitudes a controlar, se seleccionará la planta deseada y se transmitirán los datos por el módulo bluetooth configurado anteriormente. En la figura 42 se puede apreciar la apariencia de esta pantalla.

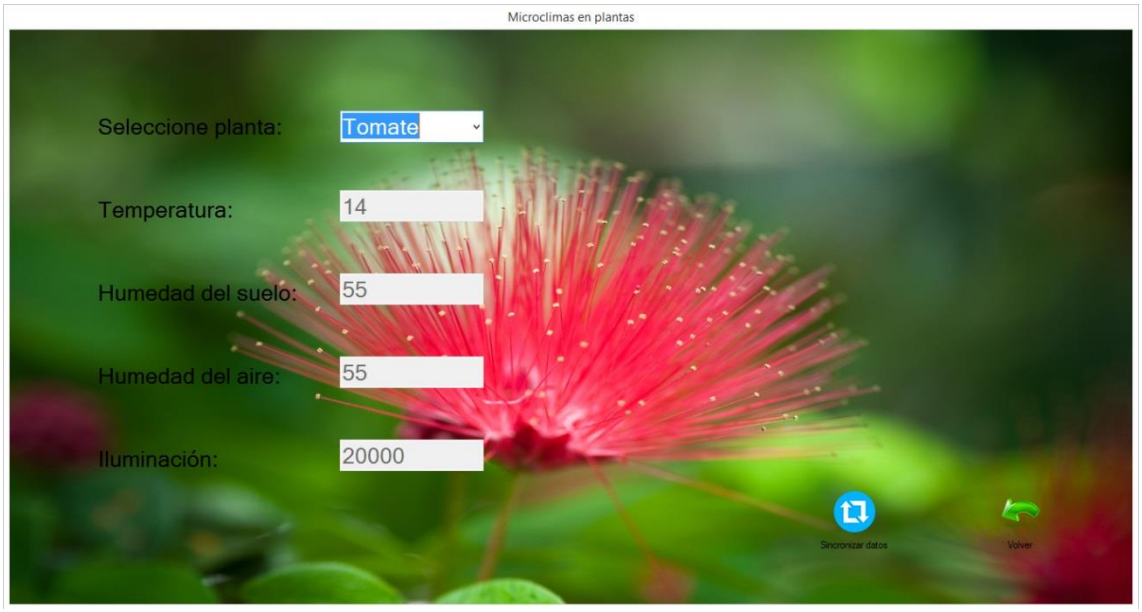


Figura 42: Apariencia de la pantalla de selección de planta para transferir los datos al habitáculo.

En la pantalla primero sólo aparece el desplegable para seleccionar la planta que se desea controlar. Una vez seleccionada la planta deseada, aparecerán los diferentes campos de texto

para poder visualizar si los datos son correctos y poder transferir dichos datos al Arduino por bluetooth. En la tabla 12 se describe el funcionamiento de los diferentes botones de la pantalla.

Tabla 12: Tabla con la descripción de los botones de la pantalla de selección de planta.	
Elemento	Descripción
 Sincronizar datos	Permite sincronizar los datos en el Arduino para poder realizar el control. En caso de no estar conectado por bluetooth, no se podrá realizar dicha sincronización.
 Volver	Botón que permite volver a la pantalla principal.

Finalmente se tiene la pantalla de monitorización de datos en la que se pueden visualizar los valores actuales de cada magnitud a medir y el valor objetivo o de referencia que debe alcanzar dicha magnitud. En la figura 43 se puede apreciar la apariencia de esta pantalla.

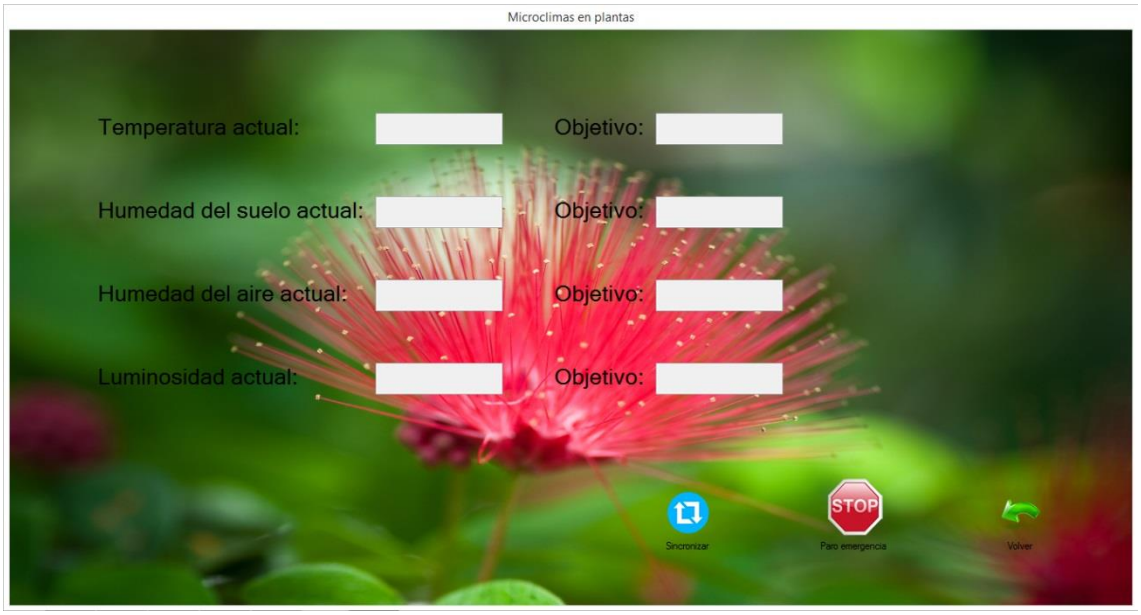


Figura 43: Apariencia de la pantalla de monitorización del sistema.

Esta pantalla dispone de una serie de campos de texto en los que se puede visualizar el valor actual y el de objetivo de cada magnitud cada vez que se desean sincronizar los datos. Se dispone de un botón de parada de emergencia que para automáticamente todo el sistema en caso de detectar un posible mal funcionamiento. En la tabla 13 se describe el funcionamiento de cada uno de los botones presentes en esta pantalla.

Tabla 13: Tabla con la descripción de cada botón de la pantalla de monitorización.	
Elemento	Descripción
 Sincronizar	Permite actualizar los datos actuales de cada magnitud. Esto sólo es posible si hay una conexión bluetooth activa.
 Paro emergencia	Botón que permite realizar una parada del sistema en caso de mal funcionamiento del sistema.
 Volver	Permite acceder a la pantalla principal.

5.10.3. Código

En este apartado se van a detallar las partes más importantes del código que contiene la aplicación de Visual Basic. El código completo de dicha aplicación se puede encontrar en el apartado de anexos.

Siguiendo el orden del apartado 5.9.2 para explicar los diferentes formularios, se va a proceder a explicar los aspectos más importantes de la pantalla principal. El aspecto más importante de dicha pantalla es el botón de marcha/paro del sistema, ya que deben cumplirse una serie de condiciones para que el sistema pueda funcionar correctamente. El resto de botones se encargan sólo de abrir el correspondiente formulario y cerrar el actual. En la figura 44 se tiene la parte de código del mismo botón.

```
Private Sub cmd_marcha_Click(sender As Object, e As EventArgs) Handles cmd_marcha.Click
    If bluetooth.sp_bluetooth.IsOpen = True Then
        If (estado = False) Then
            estado = True
            cmd_estado.Enabled = True
            Call enviar_datos("110") 'se envía el comando para iniciar el sistema
            'enviar estado marcha a arduino
            cmd_marcha.Image = Image.FromFile(Application.StartupPath & "\imagenes visual basic\stop.png") 'se modifica la imagen a mostra en el botón.
            cmd_marcha.Text = "Paro"
        Else
            estado = False
            Call enviar_datos("111") 'se envía el comando para parar el sistema
            'enviar estado paro a arduino
            cmd_marcha.Image = Image.FromFile(Application.StartupPath & "\imagenes visual basic\play.png") 'se modifica la imagen a mostra en el botón.
            cmd_marcha.Text = "Marcha"
        End If
    Else
        MsgBox("El módulo bluetooth no está activado, actívelo antes", MsgBoxStyle.Critical, "Advertencia")
    End If

End Sub
```

Figura 44: Código respectivo al botón de marcha/paro del sistema.

Primero se debe considerar si el puerto correspondiente al módulo bluetooth está abierto, en caso de no estarlo, no se puede iniciar y se muestra un mensaje de error. En el caso de ya estar abierto dicho puerto, se debe enviar el código de función correspondiente explicado en el apartado 5.4.1 para iniciar el sistema o pararlo. Además se cambiará el icono del botón así como su nombre.

En la pantalla de añadir una nueva planta la parte más importante del código es cuando se desean guardar los datos de la planta nueva a añadir y de un timer interno que permite desbloquear el botón de guardado. En la figura 45 se tiene el código correspondiente al guardado de una nueva planta.

```
    Private Sub cmd_guardar_Click(sender As Object, e As EventArgs) Handles cmd_guardar.Click
        Dim planta_existente As Boolean 'variable para determinar si existe la planta a guardar
        respuestamsg = MsgBox("¿Desea guardar la planta " & txt_nombre.Text & "?", MsgBoxStyle.YesNo, "Advertencia") 'se lanza un mensaje para asegurarse
        'del deseo de introducir esta planta
        If respuestamsg = MsgBoxResult.Yes Then 'si la respuesta es afirmativa se guardarán los datos en la base de datos.
            conex.ConnectionString = ("data source=localhost; user id=root; password=''; database=cultivos") 'se configura la conexión a la base de datos
            sql = "SELECT `Nombre` FROM `plantas`"
            Call lectura_base()
            For i = 0 To dt.Rows.Count - 1 'se recorren todos los nombres para comprobar si existe la planta que se quiere añadir
                If dt.Rows(0).Item(i) = txt_nombre.Text Then
                    planta_existente = True 'existe una planta con ese nombre
                Else
                    planta_existente = False 'no existe una planta con ese nombre
                End If
            Next
            If planta_existente = True Then
                MsgBox("Planta ya existente en la base de datos.", MsgBoxStyle.Critical, "Error")
            Else
                sql = "INSERT INTO `plantas`(`codigo_planta`, `Nombre`, `Temperatura`, `Hum_suelo`, `Hum_aire`, `Iluminacion`)" & _
                "VALUES ('" & dt.Rows.Count + 1 & "','" & txt_nombre.Text & "','" & txt_temperatura.Text & "','" & txt_humedad_suelo.Text & _
                 "','" & txt_humedad_aire.Text & "','" & txt_iluminacion.Text & "')" 'se guarda la consulta a realizar
                Call escritura_base() 'se escribe la consulta en la base de datos
                txt_nombre.Text = "" 'se borra el texto de todos los recuadros para introducir otra planta
                txt_humedad_aire.Text = ""
                txt_humedad_suelo.Text = ""
                txt_iluminacion.Text = ""
                txt_temperatura.Text = ""
                txt_nombre.Focus() 'Se selecciona el cuadro de texto del nombre de la planta para comenzar la introducción de otra planta
            End If
        End If
    End Sub
End Class
```

Figura 45: Código correspondiente al botón de guardado de una planta en la base de datos.

Primero se pide una confirmación para asegurarse de querer guardar dicha planta, en caso de no hacerlo, no se produce ninguna acción. En el caso opuesto, se comprueba si el nombre de la planta ya existe, es decir, si se ha guardado ya dicha planta en la base de datos. En el caso de no estar, se procederá a guardar los datos en la base de datos y a borrar todos los campos de texto y a poner el cursor sobre el primer campo de texto para iniciar un nuevo proceso de añadir otra planta. Si la planta ya existe, se muestra un mensaje de error para que el usuario sepa que la planta ya existe en la base de datos.

El botón timer sólo se tiene una condición en la que se activa el botón de guardado siempre que se tengan todos los campos de texto con el formato adecuado. En la figura 46 se puede apreciar el código del timer.

```
Private Sub Timer_estado_Tick(sender As Object, e As EventArgs) Handles Timer_estado.Tick
    If txt_nombre.Text <> "" And IsNumeric(txt_humedad_aire.Text) = True And IsNumeric(txt_humedad_suelo.Text) = True And IsNumeric(txt_iluminacion.Text) = True _
        And IsNumeric(txt_temperatura.Text) = True Then 'se comprueba si los cuadros están correctamente rellenados
        cmd_guardar.Enabled = True 'si los recuadros están rellenados con el nombre de la planta y sus valores, se activa el botón de guardado de datos
    Else
        cmd_guardar.Enabled = False ' en caso contrario, se desabilita dicho botón
    End If
End Sub
```

Figura 46: Código respectivo al timer que activa el botón de guardado.

En la pantalla de modificación de una planta se tiene la inicialización del formulario y el botón de guardado de los datos a modificar. En la figura 47 se puede apreciar el código respectivo a la inicialización del formulario.

```
Private Sub modificar_planta_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    cmd_guardar.Enabled = False 'inicialmente se desabilita el botón de guardado.
    txt_humedad_aire.Visible = False 'se ponen invisibles los textos y los cuadros inicialmente para poder seleccionar primero el item deseado
    txt_humedad_suelo.Visible = False
    txt_iluminacion.Visible = False
    txt_temperatura.Visible = False
    lb_humedad_aire.Visible = False
    lb_humedad_suelo.Visible = False
    lb_iluminacion.Visible = False
    lb_temperatura.Visible = False
    conex.ConnectionString = ("data source=localhost; user id=root; password=''; database=cultivos") 'se configura la conexión
    'a la base de datos
    sql = "SELECT `Nombre` FROM `plantas`" 'se consultan los nombres de las plantas
    Call lectura_base() 'se extraen los datos de la base de datos
    For i = 0 To dt.Rows.Count - 1
        cmb_planta.Items.Add(dt.Rows(i).Item(0)) 'se añaden los nombres de las plantas al desplegable.
    Next
End Sub
```

Figura 47: Código que inicializa la pantalla de actualización de los datos de una planta.

Como es la parte de inicialización del formulario, se procede a ocultar los respectivos campos de texto para que el usuario tenga que seleccionar primero la planta que desea modificar. Se consulta en la base de datos los respectivos nombres guardados y se añaden a un desplegable para que se pueda seleccionar, de una forma sencilla, la planta que se desea actualizar.

Una vez introducidos los nuevos datos se debe pulsar en el botón de guardado para introducir estos datos en la base de datos. En la figura 48 se puede observar el código que permite guardar los nuevos datos introducidos.

```
Private Sub cmd_guardar_Click(sender As Object, e As EventArgs) Handles cmd_guardar.Click
    respuestamsg = MsgBox("¿Desea actualizar los datos de la planta " & cmb_planta.SelectedItem & "?", MsgBoxStyle.YesNo, "Advertencia") 'se
    ' lanza un mensaje para asegurarse del deseo de introducir esta planta
    If respuestamsg = MsgBoxResult.Yes Then 'si la respuesta es afirmativa se guardarán los datos en la base de datos.
        conex.ConnectionString = ("data source=localhost; user id=root; password=''; database=cultivos") 'se configura la conexión
        ' a la base de datos
        sql = "UPDATE `plantas` SET `Temperatura`='" & txt_temperatura.Text & "',`Hum_suelo`='" & txt_humedad_suelo.Text & _
            "',`Hum_aire`='" & txt_humedad_aire.Text & "',`Iluminacion`='" & txt_iluminacion.Text & "' WHERE `Nombre`='" & _
            cmb_planta.SelectedItem & "'" 'se guarda la consulta a realizar
        Call actualizar_base() 'se actualiza la consulta en la base de datos
        txt_humedad_aire.Text = "" 'se borra el texto de todos los recuadros para introducir otra planta
        txt_humedad_suelo.Text = ""
        txt_iluminacion.Text = ""
        txt_temperatura.Text = ""
        cmb_planta.SelectedIndex = -1 'se pone el índice vacío del desplegable para poder volver a iniciar la secuencia de modificación
        cmb_planta.SelectedText = "" 'se borra el contenido del desplegable
    End If
End Sub
```

Figura 48: Código que permite guardar los nuevos datos introducidos en la base de datos.

Primero se muestra un mensaje para confirmar que se desean guardar los nuevos datos. En caso de no querer hacerlo, no se va a ocurrir nada, pero en el caso de si hacerlo, se actualizarán los nuevos datos y se borrará el contenido de los campos de texto y se deselecciona el desplegable para poder volver a actualizar los datos de una nueva planta y seleccionar otra planta.

En la pantalla de la conexión por bluetooth se tiene un objeto que gestiona toda la conexión y los botones para la búsqueda de los puertos serie y del botón que abre y cierra la conexión. En la figura 49 se observa el código del botón que busca los diferentes puertos serie.

```
Private Sub cmd_buscarpuertos_Click(sender As Object, e As EventArgs) Handles cmd_buscarpuertos.Click
    cmb_puertos.Items.Clear() 'limpia el desplegable
    For Each puertodisponible As String In My.Computer.Ports.SerialPortNames 'se buscan todos los puertos existentes en el PC
        cmb_puertos.Items.Add(puertodisponible) 'se añaden todos los puertos disponibles al desplegable
    Next
    If (cmb_puertos.Items.Count > 0) Then 'en caso de tener un puerto seleccionado un puerto se habilita el botón de conexión
        cmb_puertos.Text = cmb_puertos.Items(0)
        cmd_conectar.Enabled = True
    Else  'en caso contrario, se muestra un mensaje de error y se mantiene desactivado el botón de conexión
        MsgBox("No encontrado ningún puerto serie", MsgBoxStyle.Exclamation, "Advertencia")
        cmd_conectar.Enabled = False
        cmb_puertos.Items.Clear()
    End If
End Sub
```

Figura 49: Código del botón que permite buscar los diferentes puertos existentes en el PC.

Primero se limpia el desplegable y a continuación se buscan los diferentes puertos existentes y se añaden a este desplegable. Si hay algún puerto disponible, el botón de conexión se desbloqueará, en caso contrario, el botón seguirá bloqueado y se mostrará un mensaje de error.

Una vez encontrados los diferentes puertos disponibles, se podrá pulsar el botón de conexión. En la figura 50 se aprecia el código respectivo a este botón.

```
Private Sub cmd_conectar_Click(sender As Object, e As EventArgs) Handles cmd_conectar.Click
    If (cmd_conectar.Text = "Conectar") Then 'caso para la conexión
        Try
            With sp_bluetooth 'configura la conexión del puerto serie asociado al módulo bluetooth
                .BaudRate = 9600
                .DataBits = 8
                .Parity = IO.Ports.Parity.None
                .StopBits = IO.Ports.StopBits.One
                .PortName = cmb_puertos.Text
                .Open() 'se abre el puerto
            End With
            cmd_conectar.Text = "Desconectar" ' se cambia el texto del botón de conexión y la imagen a mostrar
            cmd_conectar.Image = Image.FromFile(Application.StartupPath & "\imagenes visual basic\desconectar.png")
        Catch ex As Exception ' en caso de existir un error, se muestra dicho error y no se hace nada.
            MsgBox(ex.Message, MsgBoxStyle.Critical, "Error")
        End Try
        Call enviar_datos("114") 'se envia el código de estado del sistema
        timer_estado.Enabled = True 'se habilita el timer que procederá a leer el estado del sistema.
    ElseIf (cmd_conectar.Text = "Desconectar") Then 'caso de desconexión
        cmd_conectar.Text = "Conectar" 'se cambia el texto y la imagen del botón de conexión
        cmd_conectar.Image = Image.FromFile(Application.StartupPath & "\imagenes visual basic\conectar.png") 'se modifica la imagen a mostrar
        ' en el botón.
        sp_bluetooth.Close() ' se finaliza la conexión bluetooth
    End If
End Sub
```

Figura 50: Código que permite conectar la aplicación con el sistema por bluetooth.

En la pantalla de selección de planta se dispone del objeto que inicia el proceso de envío de los datos correspondientes a una planta seleccionada en el desplegable y del timer que realiza el envío según van llegando los datos al Arduino. En la figura 51 se puede apreciar el código correspondiente al botón que inicia la sincronización de datos.

```
Private Sub cmd_enviar_Click(sender As Object, e As EventArgs) Handles cmd_enviar.Click
    respuestamsg = MsgBox("¿Desea enviar los datos de la planta " & cmb_planta.SelectedItem & "?", MsgBoxStyle.YesNo, "Advertencia") 'se lanza
    ' un mensaje para asegurarse del deseo de introducir esta planta
    If respuestamsg = MsgBoxResult.Yes Then 'si la respuesta es afirmativa se guardarán los datos en la base de datos.
        If (bluetooth.sp_bluetooth.IsOpen = True) Then
            Call enviar_datos("113") 'se envía el código para que el arduino proceda a guardar las consignas
            cmb_planta.SelectedIndex = -1 'se pone el índice vacío del desplegable para poder volver a iniciar la secuencia de modificación
            cmb_planta.SelectedText = "" 'se borra el contenido del desplegable
        Else
            MsgBox("El módulo bluetooth no está activado, actívelo antes", MsgBoxStyle.Critical, "Advertencia")
        End If
    End If
End Sub
```

Figura 51: Código correspondiente al botón de sincronización de datos.

Una vez que se pulsa este botón, el sistema devolverá un mensaje para asegurarse de que se desea controlar la planta que se haya seleccionado en el desplegable, si se responde que no, el sistema no hará nada. En caso de responder si, se detectará si la conexión bluetooth está

activa. Si no está activa, se devuelve un mensaje de error, en caso de estar activada, se transmite el código de acción correspondiente y se reinicia el desplegable para seleccionar otra planta y poder volver a realizar este proceso.

Mediante el timer, se comprobará cada cierto tiempo si se ha recibido en el arduino el dato correspondiente mediante la respuesta del mismo. En la figura 52 se aprecia el código de dicho temporizador.

```
Private Sub Timer_estado_Tick(sender As Object, e As EventArgs) Handles Timer_estado.Tick
    If cmb_planta.SelectedIndex > -1 And IsNumeric(txt_humedad_aire.Text) = True And IsNumeric(txt_humedad_suelo.Text) = True _
        And IsNumeric(txt_iluminacion.Text) = True And IsNumeric(txt_temperatura.Text) = True Then 'se comprueba si los
        ' cuadros están correctamente rellenados
        cmd_enviar.Enabled = True 'si los recuadros están rellenados con el nombre de la planta y sus valores, se
        ' activa el botón de enviado de datos
    Else
        cmd_enviar.Enabled = False ' en caso contrario, se desabilita dicho botón
    End If
    Select Case cadena_total(0)
        Case 1
            Call enviar_datos(codigo_planta) 'se envía el código de la planta elegida
        Case 2
            Call enviar_datos(txt_temperatura.Text) 'se envía la temperatura objetivo
            txt_temperatura.Text = "" 'se borra el texto del recuadros
        Case 3
            Call enviar_datos(txt_humedad_suelo.Text) 'se envía la humedad del suelo objetivo
            txt_humedad_suelo.Text = "" 'se borra el texto del recuadros
        Case 4
            Call enviar_datos(txt_humedad_aire.Text) 'se envía la humedad del aire objetivo
            txt_humedad_aire.Text = "" 'se borra el texto del recuadros
        Case 5
            Call enviar_datos(txt_iluminacion.Text) 'se envía la iluminación objetivo
            txt_iluminacion.Text = "" 'se borra el texto del recuadros
    End Select
End Sub
```

Figura 52: Código respectivo al timer que permite la transmisión de datos al Arduino.

La primera condición analiza si el formato de los datos es correcto y desbloquea el poder pulsar el botón de la sincronización de los datos entre el PC y el Arduino. En caso de ser correctos, el botón se puede pulsar, y, en caso de no ser correctos, no se tendrá acceso a dicho botón.

La segunda condición es para comprobar si se ha recibido en el arduino el dato recibido. Cuando se recibe un dato, el arduino responde con un número para determinar el orden de envío y que la aplicación pueda continuar el proceso de envío. Además se borra el campo de texto correspondiente al carácter enviado.

En la pantalla de monitorización se tiene el botón de sincronización que inicia la recepción de datos, el temporizador que permite recibir los datos y el botón de parada del sistema en caso de una emergencia. En la figura 53 se puede apreciar el código correspondiente al botón de sincronización de datos.

```
Private Sub cmd_sincronizar_Click(sender As Object, e As EventArgs) Handles cmd_sincronizar.Click
    If bluetooth.sp_bluetooth.IsOpen = True Then
        cad = ""
        Call enviar_datos("112") ' se envía el comando para proceder a la recepción de datos
        i = 1
        Call enviar_datos("1")
        Timer_recepcion.Enabled = True
    Else
        MsgBox("El módulo bluetooth no está activado, actívelo antes", MsgBoxStyle.Critical, "Advertencia")
    End If
End Sub
```

Figura 53: Código correspondiente al botón de sincronización de datos.

Primero se detecta si la conexión por bluetooth está activa, en caso de no estarlo, se devuelve un mensaje de error. En el caso opuesto, se transmite el código para que el Arduino se ponga a transmitir los datos del sistema y se transmite un valor para que éste sepa que puede transmitir el primer dato.

El temporizador permite saber si se ha recibido el correspondiente dato y responder al Arduino como que se ha recibido correctamente el dato. En la figura 54 se muestra el código del respectivo timer.

```
Private Sub Timer_recepcion_Tick(sender As Object, e As EventArgs) Handles Timer_recepcion.Tick
    Select Case i
        Case 1

            If (cad = "f") Then
                temperatura = cadena_total(0) 'se guarda el dato recibido
                Call enviar_datos("2") ' se envía un comando de confirmación de recepción
                i = i + 1
            End If
        Case 2
            If (cad = "f") Then
                humedad_suelo = cadena_total(0) 'se guarda el dato recibido
                Call enviar_datos("3") ' se envía un comando de confirmación de recepción
                i = i + 1
            End If
        Case 3
            If (cad = "f") Then
                humedad_aire = cadena_total(0) 'se guarda el dato recibido
                Call enviar_datos("4") ' se envía un comando de confirmación de recepción
                i = i + 1
            End If
        Case 4
            If (cad = "f") Then
                iluminacion = cadena_total(0) 'se guarda el dato recibido
                Call enviar_datos("5") ' se envía un comando de confirmación de recepción
                i = i + 1
            End If
        Case 5
            If (cad = "f") Then
                codigo_planta = cadena_total(0) 'se guarda el dato recibido
                i = i + 1
                txt_temperatura.Text = temperatura
                txt_humedad_suelo.Text = humedad_suelo
                txt_humedad_aire.Text = humedad_aire
                txt_luz.Text = iluminacion
                conex.ConnectionString = ("data source=localhost; user id=root; password=''; database=cultivos") 'se configura
                ' la conexión a la base de datos
                sql = "SELECT `Temperatura`, `Hum_suelo`, `Hume_aire`, `Iluminacion` FROM `plantas` WHERE `codigo_planta`='" & _
                    codigo_planta & "'" 'se guarda la consulta a realizar
                If codigo_planta = 0 Then
                    MsgBox("No hay datos de ninguna planta en el sistema de control", MsgBoxStyle.Information, "Advertencia")
                Else
                    Call lectura_base()
                    txt_obj_temp.Text = dt.Rows(0).Item(0)
                    txt_obj_hr_suelo.Text = dt.Rows(0).Item(1)
                    txt_obj_hr_aire.Text = dt.Rows(0).Item(2)
                    txt_obj_luz.Text = dt.Rows(0).Item(3)
                End If
                Timer_recepcion.Enabled = False
            End If
    End Select
End Sub
```

Figura 54: Código correspondiente al temporizador que controla la recepción de datos.

Se tiene un select case que corresponde con la respuesta de la aplicación de Visual Basic siempre y cuando se haya recibido la cadena de datos completa, es decir, cuando se haya recibido el carácter de final de cadena explicado en el apartado 5.4.1. Una vez se confirma la recepción de este carácter, se guarda el dato en la correspondiente variable y se responde al arduino para que pueda seguir con la transmisión de los datos. Cuando se finaliza la

transmisión de los datos, se llama a la base de datos para obtener los datos de referencia para poder comparar con los datos recibidos del Arduino. Finalmente se para el temporizador para optimizar la aplicación.

El botón de parada de emergencia permite parar el sistema en caso de que el sistema no funcione correctamente y poder salvar a la planta que está creciendo dentro del microclima. En la figura 55 se dispone del código de dicho botón.

```
Private Sub cmd_paro_Click(sender As Object, e As EventArgs) Handles cmd_paro.Click
    If bluetooth.sp_bluetooth.IsOpen = True Then
        Call enviar_datos("111")
        estado = False
    Else
        MsgBox("El módulo bluetooth no está activado, actívelo antes", MsgBoxStyle.Critical, "Advertencia")
    End If

End Sub
```

Figura 55: Código correspondiente al botón de para de emergencia.

Primero, como en todos los botones que van a transmitir un dato por el módulo bluetooth, se va a comprobar si dicho módulo está activado o no. En caso de no estarlo, se va a mostrar un mensaje de error. Y en el caso opuesto, se transmite el correspondiente comando para que el Arduino inicie la parada del sistema.

5.11. Pantalla Táctil

Se va a utilizar la pantalla táctil ILI9341 (20) (Ili Technology, Taiwán) que permitirá implementar un sistema de monitorización en la propia planta del sistema para cuando no se disponga del equipo informático con la aplicación de Visual Basic para realizar la misma acción. Para poder controlar esta pantalla en el Arduino se utilizan las siguientes librerías: Adafruit_GFX que permite el control de la parte gráfica de la pantalla, es decir, la parte en la que diseñan las formas de los botones, indicadores, etc que se puedan necesitar para el diseño de cualquier aplicación, y la librería Adafruit_TFTLCD que va a permitir configurar el tipo de pantalla, pines del arduino que va a utilizar y las dimensiones que contendrá el panel táctil para determinar las coordenadas máximas del panel.

Como primera parte de configuración de este panel en el Arduino para poder utilizarla, aparte de incluir las librerías mencionadas, es definir los colores básicos para el diseño de la parte gráfica. Cada color tiene un valor en hexadecimal de 16 bits que permite la representación gráfica del mismo. En la figura 56 se muestra la parte de código correspondiente a la configuración de colores básicos para poder utilizar la pantalla.

```
#define BLACK   0x0000  // Se definene los colores
#define BLUE    0x001F  // que se puedan utilizar para
#define RED     0xF800  // el texto y los elementos graficos
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF
```

Figura 56: Definiciones de los diferentes colores básicos que se pueden utilizar en la pantalla.

Después se realiza la configuración de pines, se definen los límites del panel táctil en coordenadas X, Y y el rango de detección de presión que determina que se ha tocado la pantalla y así evitar falsas pulsaciones. Además se configuran las instancias de inicio de la pantalla determinando las anteriores dimensiones y pines analógicos que utiliza el panel táctil. En la figura 57 se puede apreciar la parte de código que configura los pines y dimensiones del panel táctil.

```
 // Pines necesarios para los 4 pines del panel tactil
#define YP A3  // Pin analogico A1 para ADC
#define XM A2  // Pin analogico A2 para ADC
#define YM 9
#define XP 8
short TS_MINX = 190; // Coordenadas del panel tactil para delimitar
short TS_MINY = 161; // el tamaño de la zona donde se puede presionar
short TS_MAXX = 920; // y que coincida con el tamaño del LCD
short TS_MAXY = 820;
// Se define la presion máxima y minima que podemos realizar sobre el panel
#define MINPRESSURE 500
#define MAXPRESSURE 1000
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300); // Instancia del panel tactil (Pin XP, YP, XM, YM, Resistencia del panel)
#define LCD_CS A3   // Se define los pines del LCD
#define LCD_CD A2   // para poder visualizar elementos graficos
#define LCD_WR A1
#define LCD_RD A0
#define LCD_RESET A4 //Pin de reset de programa
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET); // Instancia LCD
```

Figura 57: Código que configura las dimensiones y pines del panel táctil acorde al panel obtenido.

Una vez llamadas a las instancias que inicializan el objeto de la pantalla táctil, se inicializa el dispositivo desde la función de setup del arduino. Esta función permite inicializar todos los dispositivos y realizar una configuración inicial cada vez que el arduino se reinicie o se ponga en marcha. En la figura 58 se puede apreciar el código de inicialización correspondiente a la pantalla táctil.

```
void setup()
{
  pinMode(13,OUTPUT); //led indicador de que se ha presionado en la pantalla táctil
  tft.begin(0x9341); // Se inicia el LCD especificando el controlador ILI9341.
  tft.setRotation(1); // Establecemos la posición de la pantalla Vertical u Horizontal
  tft.setTextSize(2);    // Se especifica el tamaño del texto
  tft.setTextColor(BLACK); // Se define el color del texto
  tft.fillScreen(WHITE); // Se pinta la pantalla de blanco
  apariencia_pantalla(); //se carga la apariencia de la pantalla
}
```

Figura 58: Función de inicialización de parámetros y objetos del Arduino.

En esta parte se define el pin 13 que activa el LED interno del Arduino para detectar las pulsaciones en la pantalla. Se inicia el panel con el driver ILI9341 correspondiente a la pantalla usada. Se ajusta la rotación de la pantalla para tener el panel apaisado. Se ajusta el tamaño del texto y el color que se va a utilizar en la apariencia gráfica para indicar los valores de los parámetros medidos. Se pone el panel en blanco para limpiarlo en caso de arrancar con la pantalla en otro color. Finalmente, se arranca la función de inicialización de la pantalla, es decir, una primera pantalla que se inicia con la apariencia gráfica que va a contener el panel.

Esta función carga las formas y textos de la pantalla para que el sistema se inicie ya con toda la apariencia y, de esta forma, sólo se debe modificar el objeto de la pantalla perteneciente

durante la ejecución del código en el Arduino. En la figura 59 se aprecia el código de esta función.

```
//Nombre: apariencia_pantalla
//Función: Carga la apariencia de la pantalla
//Necesita valores: No
//Devuelve valores: No
void apariencia_pantalla()
{
  tft.setCursor(150,5);  // Se coloca el cursor en la posición deseada (x,y)
  tft.println("Real"); // Se escribe por pantalla
  tft.setCursor(220,5);  // Se coloca el cursor en la posición deseada (x,y)
  tft.println("Objetivo"); // Se escribe por pantalla
  tft.setCursor(5,35);  // Se coloca el cursor en la posición deseada (x,y)
  tft.println("Temperatura"); // Se escribe por pantalla
  tft.setCursor(140,35);  // Se coloca el cursor en la posición deseada (x,y)
  tft.println(temp); // Se escribe por pantalla
  tft.setCursor(230,35);  // Se coloca el cursor en la posición deseada (x,y)
  tft.println(temp_c); // Se escribe por pantalla
  tft.setCursor(7,65);  // Se coloca el cursor en la posición deseada (x,y)
  tft.println("Hum. aire"); // Se escribe por pantalla
  tft.setCursor(140,65);  // Se coloca el cursor en la posición deseada (x,y)
  tft.println(hum_aire); // Se escribe por pantalla
  tft.setCursor(230,65);  // Se coloca el cursor en la posición deseada (x,y)
  tft.println(hum_aire_c); // Se escribe por pantalla
  tft.setCursor(7,95);  // Se coloca el cursor en la posición deseada (x,y)
  tft.println("Hum. suelo"); // Se escribe por pantalla
  tft.setCursor(140,95);  // Se coloca el cursor en la posición deseada (x,y)
  tft.println(hum_suelo); // Se escribe por pantalla
 tft.setCursor(230,95);  // Se coloca el cursor en la posición deseada (x,y)
 tft.println(hum_suelo_c); // Se escribe por pantalla
 tft.setCursor(5,125);  // Se coloca el cursor en la posición deseada (x,y)
 tft.println("Iluminacion"); // Se escribe por pantalla
 tft.setCursor(140,125);  // Se coloca el cursor en la posición deseada (x,y)
 tft.println(ilum); // Se escribe por pantalla
 tft.setCursor(230,125);  // Se coloca el cursor en la posición deseada (x,y)
 tft.println(ilum_c); // Se escribe por pantalla
 //dibujar botón marcha/paro
 tft.drawRect(100, 170 , 120, 60, BLACK); // Se dibuja un "boton"
 tft.drawRect(101, 171, 118, 58, BLACK); // Se dibuja un "boton" para dar mayor grosor a las líneas
 if(marcha_paro==true)//se dibujar el considera el estadodel sistema para dibujar el color adecuado
 {
   tft.fillRect(102, 172, 116, 56, RED); //Se rellema el fondo de color rojo
   tft.setCursor(135,190);  // Se coloca el cursor en la posición deseada (x,y)
   tft.println("Paro"); // Se escribe por pantalla
 }
 else
 {
   tft.fillRect(102, 172, 116, 56, GREEN); //Se rellema el fondo de color rojo
   tft.setCursor(125,190);  // Se coloca el cursor en la posición deseada (x,y)
   tft.println("Marcha"); // Se escribe por pantalla
 }
}
```

Figura 59: Código de la función de inicialización de la apariencia de la pantalla.

Como primer parámetro se define la posición X e Y del cursor donde se va a escribir cada campo de texto y posteriormente el texto a escribir, ya sea texto informativo o el valor de una variable. Una vez escritos todos los campos de texto, se procede a dibujar el botón de

marcha/paro del sistema. Este botón va a tener un doble borde para detallar el límite del mismo de una forma adecuada. Además se va a condicionar el color del mismo si el sistema ya ha sido inicializado desde la aplicación de Visual Basic o no. Este caso no suele suceder debido a que es la inicialización del Arduino, pero se ha implementado por seguridad y para que la pantalla funcione de forma correcta cuando se pulse el botón diseñado. En caso de que el sistema esté parado se, va a mostrar el botón en color rojo, y en el caso opuesto, se va a mostrar el botón de color verde.

Una vez ya configurada la apariencia de la pantalla, se va a proceder a implementar la función que permita leer las coordenadas en los diferentes ejes que determinen la posición donde el usuario ha pulsado sobre el panel táctil. En la figura 60 se puede apreciar el código que pertenece a dicha función.

```
//Nombre: lectura_panel
//Función: realiza la lectura de un punto de contacto en la pantalla
//Necesita valores: No
//Devuelve valores: No
void lectura_panel()
{
    digitalWrite(13, HIGH);
    TSPoint p = ts.getPoint(); // Se realiza la lectura de las coordenadas
    digitalWrite(13, LOW);

    pinMode(XM, OUTPUT); // La librería utiliza estos pines como entrada y salida
    pinMode(YP, OUTPUT); // por lo que es necesario declararlos como salida justo
                         // despues de realizar una lectura de coordenadas.

    // Se mapean los valores analogicos leidos del panel tactil (0-1023)
    // y se convierten en valores correspondientes a la medida del LCD 320x240
    Y = map(p.x, TS_MAXX, TS_MINX, tft.width(), 0); //Se hace un remapeado a la inversa debido
    X = map(p.y, TS_MINY, TS_MAXY, tft.height(), 0);//a que el driver reconoce las coordenadas alrevés
    Z = p.z;
}
```

Figura 60: Código que pertenece a la función que obtiene las coordenadas de pulsación en la pantalla.

Primero se activa el LED indicador de que se ha presionado el panel y se obtienen los parámetros correspondientes a las coordenadas de pulsación. Se desactiva el LED indicador de la pulsación detectada y se determinan los pines de salida XM y YP que sirven para determinar las salidas digitales que tiene conectadas el arduino al panel táctil. Es necesario ubicar la declaración en este punto ya que, de otra forma, no se obtienen las coordenadas correctamente. Una vez declarados estos pines, se procede a mapear las diferentes coordenadas correspondiente al eje de coordenadas deseado. En este caso, el eje está ubicado en la parte superior izquierda de la pantalla. En la figura 61 se puede ver cómo son el eje X e Y en el panel de una forma visual.

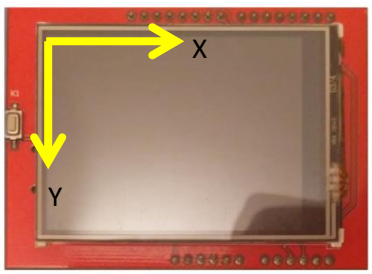


Figura 61: Distribución del mapeado de los ejes para el uso del panel táctil.

Finalmente queda detallar cómo funciona la pantalla cuando se presiona un botón o para que se actualicen los nuevos valores medidos en el sistema. Para ello, se implementa una función para atender las actualizaciones de los valores y si se presiona el botón de parada. En la figura 62 se puede apreciar la función desarrollada.

```
//Nombre: accion_pantalla
//Función: Funcionamiento de la pantalla
//Necesita valores: No
//Devuelve valores: No
void accion_pantalla()
{
  lectura_panel();//se lee el punto de contacto
  if((X>77&&X<167)&&(Y>230&&Y<303)&&(Z>MINPRESSURE && Z < MAXPRESSURE))//condición para volver atrás
  {
    if(marcha_paro==true)
    {
      marcha_paro=false;
      tft.fillRect(102, 172, 116, 56, GREEN); //Se rellema el fondo de color verde
      tft.setCursor(125,190);  // Se coloca el cursor en la posición deseada (x,y)
      tft.println("Marcha"); // Se escribe por pantalla
    }
    else
    {
      marcha_paro=true;
      tft.fillRect(102, 172, 116, 56, RED); //Se rellema el fondo de color rojo
      tft.setCursor(135,190);  // Se coloca el cursor en la posición deseada (x,y)
      tft.println("Paro"); // Se escribe por pantalla
    }
  }
  else if ((Z>MINPRESSURE && Z < MAXPRESSURE))
  {
    tft.fillRect(135, 30, 180, 120, WHITE); //Se rellema el fondo de color blanco
     tft.setCursor(140,35);  // Se coloca el cursor en la posición deseada (x,y)
     tft.println(temp); // Se escribe por pantalla
     tft.setCursor(230,35);  // Se coloca el cursor en la posición deseada (x,y)
     tft.println(temp_c); // Se escribe por pantalla
     tft.setCursor(140,65);  // Se coloca el cursor en la posición deseada (x,y)
     tft.println(hum_aire); // Se escribe por pantalla
     tft.setCursor(230,65);  // Se coloca el cursor en la posición deseada (x,y)
     tft.println(hum_aire_c); // Se escribe por pantalla
     tft.setCursor(140,95);  // Se coloca el cursor en la posición deseada (x,y)
     tft.println(hum_suelo); // Se escribe por pantalla
     tft.setCursor(230,95);  // Se coloca el cursor en la posición deseada (x,y)
     tft.println(hum_suelo_c); // Se escribe por pantalla
     tft.setCursor(140,125);  // Se coloca el cursor en la posición deseada (x,y)
     tft.println(ilum); // Se escribe por pantalla
     tft.setCursor(230,125);  // Se coloca el cursor en la posición deseada (x,y)
     tft.println(ilum_c); // Se escribe por pantalla

  }//fin if/else if

}
```

Figura 62: Función que permite accionar el botón de parada del sistema y actualizar los datos mostrados.

En esta función se considera primero si se ha pulsado el botón de parada. Esto se averigua obteniendo las coordenadas X e Y de la zona pulsada y si la cantidad de presión ejercida sobre la pantalla es la adecuada para detectar que alguien ha pulsado en ella. En caso de que la zona pulsada esté dentro de la zona de actuación del botón de parada, se procederá a cambiar su estado, de marcha a parada o viceversa.

La siguiente condición representa la actualización de los valores obtenidos por los diferentes sensores del sistema. Éste se activa si se ha presionado sobre cualquier zona de la pantalla, es decir, si se presiona sobre la pantalla, se actualizarán los datos obtenidos.

Finalmente, el resultado gráfico de dicha pantalla se puede apreciar en la figura 63.

Figura 63: Captura de la apariencia final de la pantalla táctil.

5.12. Configuración pines arduino

En este apartado se van a tratar la asignación de pines para las diferentes entradas y salidas de los diferentes elementos que va a tener el sistema según las limitaciones de pines y de memoria del Arduino elegido.

Los pines que van a contener las entradas digitales son representados en la tabla 14.

Tabla 14: Tabla con las diferentes asignaciones para las entradas digitales.	
Nombre	Pin
LCD_D2	2
LCD_D3	3
LCD_D4	4
LCD_D5	5
LCD_D6/ TOUCH XP	6
LCD_D7/ TOUCH YM	7
LCD_D0	8
LCD_D1	9
SD_DI	11
Entrada sensor DHT 22	46
Pin RX Arduino (TX módulo bluetooth)	52

Una vez detalladas las entradas digitales, se va a proceder a detallar las salidas digitales del arduino que van a permitir actuar sobre el microclima. En la tabla 15 se pueden apreciar las diferentes asignaciones de los pines de salida.

Tabla 15: Tabla con las diferentes asignaciones de los pines de salida digital.	
Nombre	Pin
Salida electro válvula	22
Salida humidificador de aire	23
Salida resistencias térmicas	24
Salida PWM placas LED	45
Salida TX Arduino (RX módulo bluetooth)	53
SD_CS	10
SD_DO	12
SD_SCK	13

Finalmente sólo queda definir los diferentes pines que corresponden a las entradas analógicas del Arduino que conectan a los diferentes sensores analógicos y a diferentes pines del panel táctil. En la tabla 16 se muestran las diferentes asignaciones de pines.

Tabla 16: Tabla con las diferentes asignaciones de los pines de entrada analógica.	
Nombre	Pin
LCD_RD	A0
LCD_WR/ TOUCH_YP	A1
LCD_RS/ TOUCH_XM	A2
LCD_CS	A3
LCD_RST	A4
Entrada sensor de luz BPW34	A10
Entrada sensor de humedad del suelo SEN92355P	A11

5.13. Código arduino

En este apartado se van a detallar los aspectos más relevantes del código implementado en el Arduino, el código entero está ubicado en el apartado de anexos.

La primera parte del código es la introducción de las diferentes librerías a utilizar. En la figura 64 se dispone de todas las librerías incluidas.

```
//importar librerías
#include <SoftwareSerial.h>//librería para el uso de las comunicaciones serie
#include <DHT.h>//librería para el uso del sensor DHT22
#include <Adafruit_GFX.h>    // Libreria de graficos
#include <Adafruit_TFTLCD.h> // Libreria de LCD
#include <stdint.h>
#include "TouchScreen.h"//libreria táctil
#include <TimerOne.h> //librería pra poder utilizar interrupciones temporizadas
```

Figura 64: Figura con todas las librerías incluidas en el código de Arduino.

Las librerías que se disponen son las siguientes:

- SoftwareSerial.h: Permite utilizar las comunicaciones serie para utilizar el módulo bluetooth.
- DHT.h: Permite utilizar los sensores DHT para el control de la temperatura y la humedad relativa del aire.
- Adafruit_GFX.h: Sirve para utilizar las diferentes funciones de control de gráficos en la pantalla táctil.

- Adafruit_TFTLCD.h: Permite configurar los diferentes modelos de pantallas táctiles.
- Stdint.h: Sirve para las funciones genéricas del Arduino.
- TouchScreen.h: Permite el manejo de la pantalla táctil.
- TimerOne.h: Permite el uso de interrupciones temporizadas para el muestreo de los datos de los diferentes sensores.

Una vez incluidas todas las librerías necesarias, se procede a inicializar los diferentes objetos y variables necesarias. Con todo esto ya declarado, se procede a la función de configuración del sistema que permite inicializar todos los objetos y periféricos necesarios. En la figura 65 se dispone del código de la función de inicialización.

```
//Nombre: setup
//Función: Configurar módulos y pines
//Necesita valores: No
//Devuelve valores: No
void setup()
{
  pinMode(13,OUTPUT); //led indicador de que se ha presionado en la pantalla táctil
  pinMode(22,OUTPUT); //Salida de la electro válvula.
  pinMode(23,OUTPUT); //Salida del humidificador del aire.
  pinMode(24,OUTPUT); //Salida de las resistencias térmicas para el control de la temperatura.
  tft.begin(0x9341); // Se inicia el LCD especificando el controlador ILI9341.
  tft.setRotation(1); // Establecemos la posición de la pantalla Vertical u Horizontal
  tft.setTextSize(2);    // Se especifica el tamaño del texto
  tft.setTextColor(BLACK); // Se define el color del texto
  tft.fillScreen(WHITE); // Se pinta la pantalla de blanco
  apariencia_pantalla(); //se carga la apariencia de la pantalla
  BT.begin(9600); //Velocidad del puerto del módulo Bluetooth
  dht.begin();//Se inicia el sensor DHT
  Timer1.initialize(2000000); //inicializa la interrupción temporizada a 2 segundos.
  Timer1.attachInterrupt(control_sistema);// se asigna la interrupción a la función que realiza el control del sistema.
}//fin void setup
```

Figura 65: Función de la inicialización de los periféricos y objetos.

Primero se configuran los pines de salida digital que corresponden a los diferentes actuadores y al LED que indica que se ha pulsado en la pantalla táctil. A continuación está toda la configuración de la pantalla táctil, empezando por el modelo de pantalla, la rotación de la pantalla, el tamaño del texto y su color y una limpieza de la pantalla dejándola de color blanco y finalmente, se carga la apariencia inicial de la pantalla. Una vez cargada la pantalla, se inicializa el módulo bluetooth a la velocidad deseada, se inicializa el sensor DHT y se inicializa la interrupción temporizada y se asigna a la función que la gestiona.

Una vez inicializado el sistema, se procede a la parte del código que se va a repetir infinitas veces. En la figura 66 se puede apreciar dicho código.

```
//Nombre: loop
//Función: programa principal en bucle repetitivo
//Necesita valores: No
//Devuelve valores: No
void loop()
{
  while (BT.available()>0)
  {
    codigo=lectura_bt();//se procede a leer primero el código de envío para saber que función se debe realizar
    accion_realizar(codigo);//llamada a la función que averigua el código recibido
    codigo=0;//se resetea el código de envio para evitar fallos.
  }//fin if BT.available
  ilum=lectura_porcentual_dir(A7);//se lee la iluminación. Se coloca en el loop debido a que la luz es una variable de acción rápida.
  x_ilum[0]=ilum;//ajuste variable filtro.
  filtro(x_ilum,y_ilum);//filtrado
  ilum=y_ilum[0];//ajuste magnitud despues filtro.
  accion_pantalla(); //atiende a la lectura de la pantalla y su interacción.
}//fin void loop
```

Figura 66: Función que repite el código principal infinitas veces.

En la primera parte se dispone de un bucle while que permite mantenerse dentro del mismo mientras que se disponga de información en el buffer de recepción del mismo. Si se reciben datos, se gestionará el código transmitido y se procederá a realizar la respuesta. En caso de no disponer información en el buffer, no se mantendrá en dicho bucle.

Fuera de este bucle se dispone de la lectura del sensor de iluminación y su filtrado. Se coloca en esta parte del código debido a que es una magnitud de acción rápida y necesita ser muestreada lo más rápido posible.

Finalmente se dispone de la llamada de la interactuación de la pantalla táctil en la que se van a actualizar los datos y se atenderá a si se ha pulsado el botón de parada de emergencia.

Ahora se van a explicar las funciones de lectura y escritura por el módulo bluetooth. La parte de la lectura se encarga de recibir todos los datos y separar toda la cadena recibida del carácter de fin de cadena. En la figura 67 se dispone la función de la lectura por bluetooth.

```
//Nombre: lectura_bt
//Función: realizar la lectura bluetooth
//Necesita valores: No
//Devuelve valores: valor integer del parámetro necesario
int lectura_bt()
{
  int valor=0; //variable para la devolución de la función
  String cadena; //variable para la cadena de datos completa
  char dato;//variable para la recepción de caracteres
  do //hace el bucle necesario para leer toda la cadena da datos
  {
   dato=BT.read();//lectura de los datos enviados del pc
   if(dato=='f')//mientras que el dato recibido no sea f se añaden caracteres a la cadena
   {
     valor=cadena.toInt();//se convierte la cadena de caracteres a un valor integer
   }
   else
   {
     cadena=cadena+dato;//se añade caracter a la cadena
   }//fin if/else dato=='f'
  }//fin do
  while(dato!='f');//hasta que no se reciba el dato f no se sale del bucle
  return valor;//devuelve el parámetro para el control
}//fin lectura bt
```

Figura 67: Función de la lectura de caracteres procedentes del módulo bluetooth.

En esta parte se declaran las variables necesarias para la lectura de la cadena, para extraer la parte deseada y para convertir el dato obtenido en un valor numérico con el que tratar a posteriori. Se declara un bucle do while en que se repetirá la lectura siempre que carácter enviado no sea el carácter de final de cadena ya que éste carácter siempre será el último a transmitir siempre. Dentro del do, se tiene la condición en la que si el último carácter leído es el carácter de final de carrera, se obtiene la cadena ya leída hasta el momento actual y se convierte a un valor numérico para poder tratarlo, en caso opuesto se añade el carácter leído a la cadena que se va componiendo conforme se van leyendo los diferentes caracteres.

La parte de escritura es la encargada de componer la cadena junto con el carácter de final de cadena y transmitir todo el conjunto por el módulo bluetooth. En la figura 68 se dispone de la función de escritura.

```
//Nombre: escritura_bt
//Función: realizar la escritura bluetooth
//Necesita valores: cadena de caracteres string
//Devuelve valores: No
void escritura_bt(String cad)
{
  cad=cad+'f';// se añade a la cadena el parámetro de fin de cadena
  BT.println(cad);//proceso de envío de la trama al pc
}//fin escritura bt
```

Figura 68: Función que permite transmitir datos por el módulo bluetooth.

Esta función coge el parámetro introducido que es la cadena de caracteres a transmitir y le añade el carácter de final de cadena. Una vez añadido lo transmite por el módulo bluetooth.

Cuando se han recibido datos, siempre se recibe primero el código identificador que permite saber que acción se va a realizar. Para ello, se ha implementado una función que determina el tipo de acción a realizar una vez recibidos los datos por bluetooth. En la figura 69 se dispone de dicha función.

```
//Nombre: accion_realizar
//Función: realiza las correspondientes acciones a partir de la lectura bt
//Necesita valores: Entero sin signo que indica el código de acción
//Devuelve valores: Ninguno
void accion_realizar(unsigned int cod)
{
    switch (cod)
    {
      case 110://caso en el que el sistema se pone en funcionamiento
      marcha_paro=true;//se pone a true la variable que determina el inicio o paro del sistema
      break;
      case 111://caso en el que el sistema se para
      marcha_paro=false;//se pone a false la variable que determina el inicio o paro del sistema
      break;
      case 112://caso en el que se procede a leer el estado del sistema
      estado_sistema();//se llama a la función que realiza el envío de datos del estado del sistema
      break;
      case 113://caso en el que recbien las consignas y código de planta desde la app en VB
      lectura_consignas();//se llama a la función lectura_consignas para la lectura de todas las consignas
      break;
      case 114: //caso en el que se averigua el estado del sistema cuando hay una vinculación con la app en VB
      sistema_marcha_paro();
      break;
      default://caso diferente
      break;
    }//fin switch
}//fin accion_realizar
```

Figura 69: Función que permite clasificar el tipo de acción a realizar en el sistema.

Esta función está compuesta por una sentencia switch que permite englobar diferentes casos según el valor de una variable. En este caso se engloban los casos según el valor de la tabla de códigos ubicada en el apartado 5.4.1. Una vez se acceda a uno de los diferentes casos, se accederán a diferentes funciones o se cambiará el valor de la variable que pone en funcionamiento el sistema o viceversa. En la figura 70 se aprecia dicha función.

```
//Nombre: estado_sistema
//Función: Enviar los datos necesarios para observar el estado del sistema en la app de VB
//Necesita valores: Ninguno
//Devuelve valores: Ninguno
void estado_sistema()
{
  i=1;
     while (i<=5)
     {
      while(BT.available()<1);//mientras que no haya algo en el buffer de entrada, no se procede a recibir el dato
      codigo=lectura_bt();//se lee el comando de confirmación para vaciar el buffer de entrada
       switch (codigo)
       {
         case 1:
            envio="";//se deja en blanco el string para poder añadir la siguiente variable a enviar
            envio=envio+temp;//se añade al string el valor de la variable a enviar.
            escritura_bt(envio);//se envia el valor deseado
            i++;
            break;
         case 2:
            envio="";
            envio=envio+hum_suelo;//se añade al string el valor de la variable a enviar.
            escritura_bt(envio);//se envia el valor deseado
            i++;
            break;
          case 3:
             envio="";
             envio=envio+hum_aire;//se añade al string el valor de la variable a enviar.
             escritura_bt(envio);//se envia el valor deseado
             i++;
             break;
          case 4:
             envio="";
             envio=envio+ilum;//se añade al string el valor de la variable a enviar.
             escritura_bt(envio);//se envia el valor deseado
             i++;
             break;
          case 5:
             envio="";
             envio=envio+cod_planta;//se añade al string el valor de la variable a enviar.
             escritura_bt(envio);//se envia el valor deseado
             i++;
             break;
        }//fin switch
     }//fin while
}
```

Figura 70: Función que permite implementar la acción correspondiente al código enviado.

En esta función se va recibir los datos en un cierto orden para saber qué dato corresponde a cada magnitud. Para ello se utiliza una variable para identificar el estado en el que se está. Además, se tiene un bucle while para no salir de esta función hasta que no se reciban todos los datos. Dependiendo de la posición en la que se esté dentro de la sentencia switch, se transmite una variable y se incrementa el valor para acceder al siguiente caso. Todo esto es para que la comunicación entre el arduino y la aplicación de Visual Basic estén sincronizadas y se puedan entender.

En el caso de que el código recibido sea el de lectura de consignas, se accederá a una función que permite obtener todos los valores de referencia desde la aplicación de Visual Basic. En la figura 71 se dispone de dicha función.

```
//Nombre: lectura_consignas
//Función: realizar la lectura de las consignas desde la app en vb
//Necesita valores: Ninguno
//Devuelve valores: Ninguno
void lectura_consignas()
{
    escritura_bt("1");
    while(BT.available()==0);//mientras que no haya algo en el buffer de entrada, no se procede a recibir el dato
    cod_planta=lectura_bt();//se procede a leer las consignas y código de planta
    escritura_bt("2");//se responde al sistema como que ha llegado el primer dato
    while(BT.available()<1);
    temp_c=lectura_bt();
    escritura_bt("3");
    while(BT.available()<1);
    hum_suelo_c=lectura_bt();
    escritura_bt("4");
    while(BT.available()<1);
    hum_aire_c=lectura_bt();
    escritura_bt("5");
    while(BT.available()<1);
    ilum_c=lectura_bt();
    escritura_bt("0");// Se envía un valor fuera de rango para no recibir siempre el mismo dato debido al timer de envío en la app de VB
}//fin lectura_consignas
```

Figura 71: Función que permite recibir los valores de referencia para el control del microclima.

El funcionamiento de esta función es similar al de la función anterior pero de forma inversa. En este caso es el Arduino el que debe recibir la información y transmitir el valor de la posición para tener la sincronización. Para ello, el Arduino transfiere el dato de posición y espera a recibir los datos. Una vez recibidos, vuelve a transmitir el siguiente dato de posición. De esta forma se implementa para todos los datos y, para terminar, se envía un último valor para saber que la sincronización entre dispositivos ha sido correcta.

Finalmente, si se inicia el sistema, se pondrá en marcha el control del sistema cada 2 segundos ya que, la función implementada, funciona bajo una interrupción temporizada. Mientras que no se active el sistema, la función no realiza nada aunque la interrupción esté funcionando. En la figura 72 se dispone de esta función.

```
//Nombre: control_sistema
//Función: realizar el control del microclima
//Necesita valores: Ninguno
//Devuelve valores: Ninguno
void control_sistema()
{
  //lectura de magnitudes
  hum_aire=dht.readHumidity();//se lee la humedad del aire
  temp=dht.readTemperature();//se lee la temperatura
  hum_suelo=lectura_porcentual_inv(A6);//se lee la humedad del suelo
  x_hum[0]=hum_suelo;
  filtro(x_hum,y_hum);//filtrado
  hum_suelo=y_hum[0];
  if(marcha_paro==true)
  {
    //control todo o nada con histéresis de la temperatura
    if(temp>=temp_c)
    {
      salida_digital_bajo(24);//parar resistencias
    }
    else if(temp<=(temp_c-1))//se considera una histéresis de un grado ya que se encuentra entre el márgen óptimo de las plantas
    {
      salida_digital_alto(24);//activar resistencias
    }//fin if/else if temp
    //control todo o nada con histéresis de la humedad del aire
     if(hum_aire>=hum_aire_c)
     {
      salida_digital_bajo(23);//parar humidificador
     }
     else if(hum_aire<=(hum_aire_c-3))
     {
       salida_digital_alto(23);//activar humidificador
     }//fin if/else if hum_aire
     //control todo o nada con histéresis de la humedad del suelo
     if(hum_suelo>=hum_suelo_c)
     {
      salida_digital_bajo(22);//parar electrovalvula
     }
     else if(hum_suelo<=(hum_suelo_c-3))
     {
       salida_digital_alto(22);//activar electrovalvula
     }//fin if/else if hum_suelo
     SalidaPWM=pid(ilum,ilum_c);
     analogWrite(analogOutPin, SalidaPWM); //escritura pwm en el pin
     analogWrite(analogOutPin, (255-SalidaPWM));
  }//while
}//fin void control_sistema
```

Figura 72: Código correspondiente al control del sistema.

En la primera parte se procede a leer los diferentes sensores y a realizar su filtrado en caso de ser un sensor analógico. A continuación, se considera el estado del sistema, es decir, si el sistema está en funcionamiento, se procederá a controlar las diferentes magnitudes que intervienen en el sistema y, en caso contrario, no se realiza ninguna acción. En caso de estar activo el sistema, se procede a las condiciones que activan y paran todos los controles todo o nada con histéresis. Todas estas condiciones funcionan de la misma forma, si se supera la consigna superior, se para la salida y, si se reduce la magnitud por debajo de la histéresis, se activa la salida digital correspondiente. Finalmente se tiene la llamada del control PID para regular la iluminación del sistema y la activación de la salida PWM con respecto al control PID realizado anteriormente.

Para el filtrado se ha implementado una función en la que se coloca la ecuación en diferencias obtenida en el apartado 5.6. En la figura 73 se puede apreciar la función mencionada.

```
//Nombre: Filtro
//Función: Filtro paso bajo de tercer orden
//Necesita valores: las últimas 4 entradas y salidas
//Devuelve valores: Ninguno
void filtro(int entrada[], int salida[])
{
  salida[k]=0.0753*entrada[k]+0.2259*entrada[k+1]+0.2259*entrada[k+2]+_
  0.0753*entrada[k+3]+0.8266*salida[k+1]-0.5154*salida[k+2]+0.08648*salida[k+3]; //ecuacion en
                                                          //diferencias del filtro de 3 er orden
  salida[k+3]=salida[k+2];
  salida[k+2]=salida[k+1];
  salida[k+1]=salida[k];
  entrada[k+3]=entrada[k+2];
  entrada[k+2]=entrada[k+1];
  entrada[k+1]=entrada[k];
}
```

Figura 73: Función que implementa el filtro paso bajo.

Esta función tiene dos parámetros de entrada en los cuales forman las últimas cuatro entradas y salidas que ha tenido el filtro para poder calcular el último valor. Como primer paso, se calcula el valor actual a la salida del filtro y, después se procede a actualizar la línea temporal en las variables de entrada y salida para el próximo cálculo.

6. Diseño del habitáculo

En este apartado se exponen los diferentes planos que forman el habitáculo separados por diferentes habitáculos que conforman todo el montaje. En el anexo se podrán encontrar los planos completos.

6.1. Habitáculo

Esta parte es la que conforma el habitáculo donde se ubicará el cultivo a colocar, el espacio para la tierra a cultivar y los diferentes sensores y actuadores que van a controlar el sistema.

El habitáculo está compuesto por dos partes:

- Parte aérea: Esta es la parte en la que se puede apreciar la tierra y se va a controlar la temperatura, la humedad relativa del aire y la iluminación del microclima. Tiene forma cúbica de 50 cm de lado. Dentro de habitáculo se ubicarán los correspondientes actuadores y sensores de las tres magnitudes a controlar dentro de esta zona. En la parte de fuera, se dispone de la pantalla conectada al arduino que, a su vez, están tapados por una tapa que sólo permite apreciar la pantalla. Finalmente se dispone de una serie de bisagras y cierre para poder abrir y cerrar el habitáculo cuando se desee. Cabe resaltar que el material a utilizar para esta parte es el metacrilato ya que es un material que permite observar el desarrollo de la panta, permite retener el calor dentro del habitáculo y puede dejar pasar la luz natural del exterior.
- Parte de tierra y patas: Esta parte está compuesta por la zona en la que se va a ubicar la tierra necesaria para la planta y el actuador y sensor para el control de la humedad del suelo. Se va a disponer de una superficie cuadrada de 50 cm de lado con una altura de 25 cm haciendo que se tenga un volumen de 0.0625 m^3 para albergar la tierra que será necesaria para el cultivo de la planta deseada. Esta zona será de aluminio ya que es un material poco pesado y a su vez resistente.

En la figura 74 se puede apreciar el resultado de las diferentes perspectivas del habitáculo.

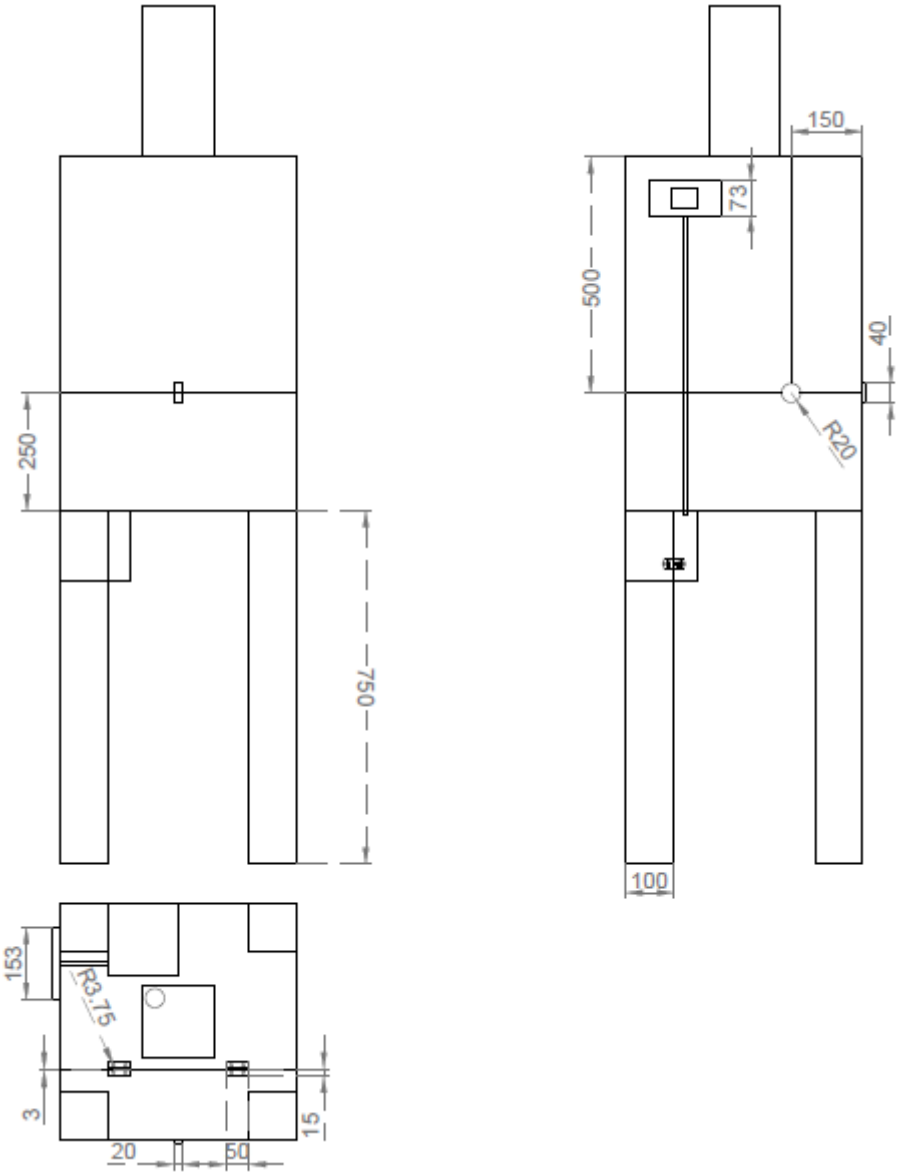


Figura 74: Diseño del habitáculo completo en sus diferentes perspectivas.

6.2. Caja de componentes

Esta parte es la que conforma la caja donde se ubicará toda la parte de la electrónica y la fuente de alimentación que permite que todos los componentes electrónicos puedan funcionar.

La caja será de forma cúbica de 15 cm de lado, la que albergará la placa de componentes electrónicos y la fuente de alimentación. El Arduino estará donde se ubique la pantalla táctil para aislar la parte de potencia de la parte digital. La caja dispondrá de una pequeña abertura por la que saldrán los cables que irán a las diferentes entradas del Arduino. Finalmente la caja tendrá una conexión a red para conectar el sistema a la red eléctrica. En la figura 75 se muestra el resultado del diseño de la caja.

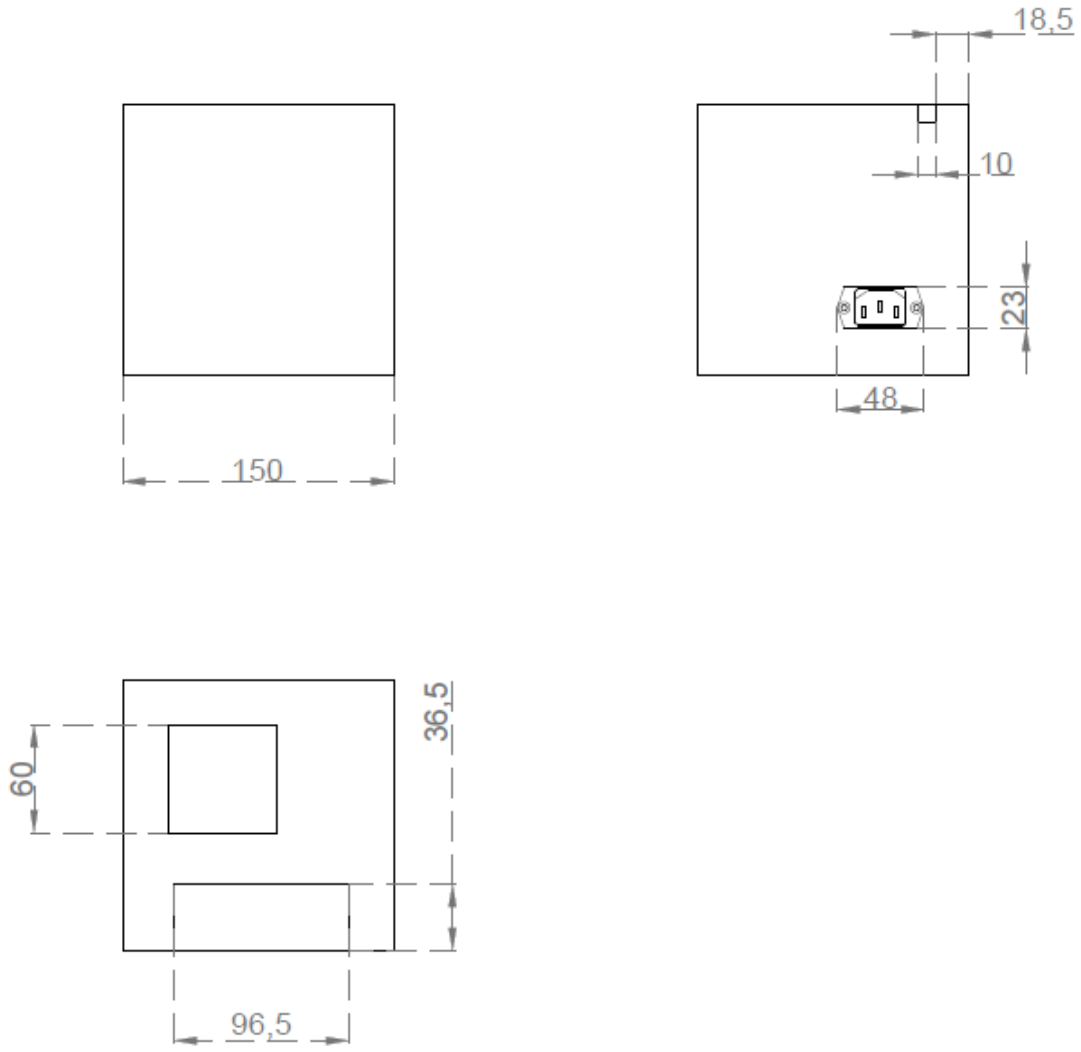


Figura 75: Diseño del plano de la caja de componentes en las diferentes perspectivas.

6.3. Habitáculo humidificador

Esta parte es el habitáculo del depósito del humidificador donde se va a alojar el actuador que permita modificar la humedad relativa del aire.

Sabiendo que el humificador consume 300 ml por hora se desea tener una autonomía de 24 horas sin tener que volver a llenar el depósito de agua se va a obtener el volumen necesario para dicha autonomía. En la ecuación 6.3.1 se detalla cómo obtener el volumen necesario.

$$Volumen\ total = \frac{Volumen\ por\ hora * horas}{1000} \qquad (Ec\ 6.3.1)$$

$$Volumen\ total = \frac{\frac{300}{1000} * 24}{1000} = 0{,}0072\ m^3$$

Una vez obtenido el volumen necesario se procede al diseño del habitáculo. Se supone que el habitáculo será de superficie cuadrada, es decir, que dos lados van a ser iguales. Si se define que el lado va a ser de 15 cm, sólo queda por determinar la altura del habitáculo. En la ecuación 6.3.2 se obtiene la altura necesaria.

$$Volumen = lado^2 * altura \qquad (Ec\ 6.3.2)$$

$$0.0072 = \left(\frac{15}{100}\right)^2 * altura$$

$$altura = 0.32\ m \rightarrow 32\ cm$$

Una vez obtenidas las dimensiones del depósito de agua, sólo queda implementar su diseño en autocad.

El material a utilizar para el diseño del depósito va a ser el metacrilato para seguir la línea del habitáculo, además de poder observar de forma manual el nivel de agua restante. Se utilizarán selladores para que el agua no pueda filtrar por ningún lado del depósito ni por la abertura para el humidificador que convierte el agua del cubículo en vapor de agua para actuar sobre la humedad del aire.

En la Figura 76 se puede apreciar el resultado del diseño implementado.

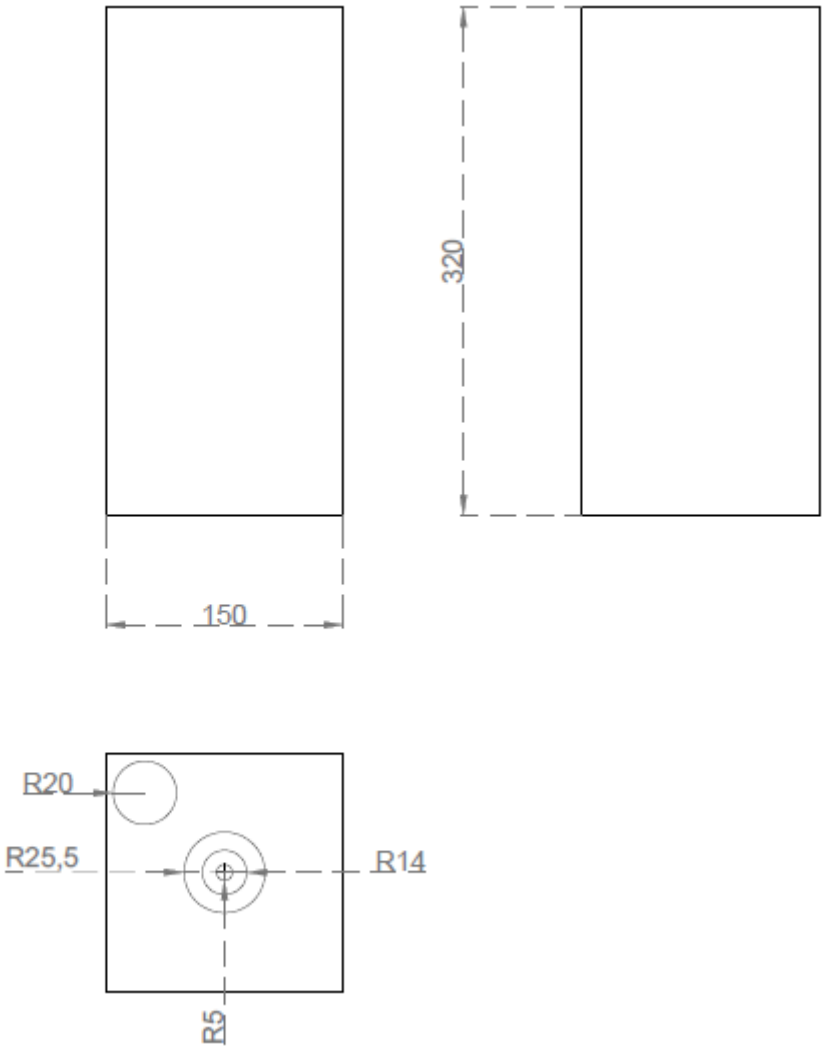


Figura 76: Diseño del habitáculo del humidificador en todas sus perspectivas.

7. Presupuesto

En este apartado se van a detallar el valor de cada elemento utilizado para poder implementar este Trabajo de Fin de Grado junto con su precio aproximado. Cabe destacar que este apartado es orientativo debido a que el precio de los materiales utilizados varía con el paso el tiempo y con el ajuste de impuestos correspondiente, en este caso el 21%.

Para la realización del prototipo se estima el empleo de dos trabajadores, un ingeniero o jefe de producción para el desarrollo del prototipo y un electrónico de primera para el montaje del mismo. En la tabla 17 se puede apreciar el coste por día trabajado de los trabajadores necesarios.

Tabla 17: Coste salarial de los trabajadores necesarios para el desarrollo del prototipo.	
Trabajador	Salario (€/día)
Jefe de operación/Ingeniero	113.60
Electrónico de primera	61.34

El gasto de la programación de los diferentes lenguajes utilizados, corre a cargo de las horas realizadas por el ingeniero empleado. En la tabla 18 se puede apreciar el coste del desarrollo del código.

Tabla 18: Coste del desarrollo y verificación de los diferentes lenguajes de programación.			
Actividad	Categoría	Tiempo (horas)	Precio (€)
Diseño de la base de datos	Ingeniero	2	28.40
Código Arduino	Ingeniero	100	1420
Código Visual Basic	Ingeniero	90	1278
Depuración	Ingeniero	20	284
Total presupuesto sin IVA			**2487.93**
Total presupuesto con IVA (21%)			**3010.40**

Una vez implementados los gastos de programación de código, se van a estimar los costes para el diseño y montaje del habitáculo, así como toda la electrónica diseñada e implementada. Para esta labor, se van a precisar los servicios del ingeniero y del electrónico de primera mencionados en la tabla 17. En la tabla 19 se muestran los costes del diseño y montaje del prototipo.

Tabla 19: Coste del desarrollo y montaje del prototipo.			
Actividad	Categoría	Tiempo (horas)	Precio (€)
Desarrollo habitáculo	Ingeniero	50	710
Desarrollo electrónica	Ingeniero	40	568
Desarrollo reguladores	Ingeniero	20	284
Desarrollo filtros	Ingeniero	3	42.60
Gestión de compras	Electrónico de primera	4	30.67
Montaje	Electrónico de primera	15	115.01
Verificación y testeo	Ingeniero	4	56.80
Verificación y testeo	Electrónico de primera	4	30.67
Total presupuesto sin IVA			**1518.64**
Total presupuesto con IVA (21%)			**1837.55**

Finalmente se detalla el coste de cada componente utilizado para el montaje del prototipo, así como la cantidad suministrada y el proveedor utilizado. En la tabla 20 se muestra el presupuesto de los elementos utilizados.

Tabla 20: Tabla con el presupuesto de los materiales utilizados.				
Componente	Precio (€)	Cantidad	Proveedor	Total (€)
Arduino mega	42.98	1	Arduino	42.98
Panel táctil	3.51	1	Aliexpress	3.51
Humidificador usb	5.77	1	Aliexpress	5.77
Fuente de alimentación 12 V y 5V	12.46	1	Farnell	12.46
Módulo Bluetooth HC-06	2.43	1	Aliexpress	2.43
Placas LED 3W 12 V	0.79	4	Banggod	3.16
Sensor DHT22	4.06	1	Aliexpress	4.06
BD243C	0.50	2	Farnell	1.00
74HC04	0.32	1	Farnell	0.32
LMC6042A	2.97	2	Farnell	5.94

BPW34	1.08	1	Farnell	1.08
Relé SRD-05Vdc-SL-C	1.95	3	Farnell	5.85
Sensor SEN92335P	4.41	1	Digikey	4.41
Resistencia 3.3 kΩ	0.20	2	Farnell	0.40
Resistencia 5.6 kΩ	0.48	1	Farnell	0.48
Resistencia 10 kΩ	0.32	3	Farnell	0.96
Resistencia 680 Ω	0.38	1	Farnell	0.38
Resistencia 4.7 kΩ	0.21	1	Farnell	0.21
Resistencia 180 kΩ	0.48	1	Farnell	0.48
Condensador 100 nF	0.09	1	Farnell	0.09
Resistencia 5 kΩ 12W	9.72	2	Farnell	19.44
Electroválvula ASC3	34.56	1	Ebay	34.56
Conector alimentación	1.76	1	Farnell	1.76
Cable 2.5 mm^2 5 m (Azul, marrón y TT)	2.25	3	Leroy Merlín	6.75
Cable 0.25 mm^2 25 m (Rojo y negro)	20.10	2	Leroy Merlín	40.20
Placa PCB 10x16 mm	3.53	1	Farnell	3.53
Plancha metacrilato 1510x1015x10 mm	159.40	1	Macoglass	159.40
Plancha aluminio 900x1500x10 mm	9.40	1	Motedis	9.40
Bisagras	1.95	2	Leroy Merlín	3.90
Pasador	1.85	1	Leroy Merlín	1.85
Ventilador 230 V	31.26	1	Farnell	31.26
Tornillos M3x12 rosca chapa	0.03	20	Leroy Merlín	0.55
Silicona selladora	6.95	3	Leroy Merlín	20.85
Total presupuesto sin IVA				**354.55**
Total presupuesto con IVA (21%)				**429.42**

Hay que destacar que las planchas de aluminio, metacrilato y la placa PCB son de medidas estándar, por lo que va a sobrar una pequeña parte durante el corte de las respectivas piezas que conforman el diseño del microclima.

8. Propuestas de mejora

En este apartado se van a redactar las posibles mejoras a este Trabajo de Fin de Grado para se pueda obtener un sistema aún más completo y con menor mantenimiento.

- Implementación de un sistema de enriquecimiento de CO_2: tal y como se ha explicado en el apartado 2.1.3, aportar un enriquecimiento de CO_2 puede aportar hasta un 30% de aumento en el rendimiento del cultivo, lo que supone obtener un producto de mayor calidad.
- Sustitución del arduino por un DSP: Al utilizar un DSP se puede obtener una mayor velocidad del proceso ya que el procesador del Arduino funciona a una frecuencia de 16 MHz y un DSP puede alcanzar los 200 MHz de frecuencia de trabajo. Esto supone aumentar la velocidad del proceso en más de 10 veces. El único inconveniente es que supone adaptar las diferentes librerías implementadas en el código de arduino al DSP y ajustar los rangos de las entradas analógicas a 3.3 V.
- Sustitución del sensor DHT22 por un higrómetro y una PT100: Estos dos sensores son más precisos que el sensor utilizado ya que ofrecen una respuesta más lineal y con un error más reducido. Al no ser sensores de tipo digital como es el caso del sensor

utilizado para la temperatura y la humedad relativa del aire, se debe implementar una etapa de amplificación para ajustar al rango de medida deseado.

- Implementación de reguladores de tipo PID a todas las magnitudes: Este tipo de regulador permite ajustar la referencia de una forma más exacta y sin tener una sobreoscilación excesiva, cosa que el control todo o nada con histéresis implementado si la tiene y es difícil poder reducirla modificando el funcionamiento de los diferentes actuadores.
- Control del nivel del habitáculo del humidificador: Implementar un sensor de nivel para cuando el nivel del depósito de agua del humidificador descienda de un nivel mínimo, abra una electroválvula que rellene el depósito hasta cierto nivel máximo que será controlado por otro sensor de nivel.
- Implementar una aplicación móvil que permita acceder a la base de datos de forma remota y se pueda comunicar con el Arduino realizando las mismas funciones que la aplicación de Visual Basic ya implementada.

9. Conclusiones

- Se ha desarrollado un sistema para poder crear un microclima dado una serie de condiciones específicas que son diferentes para cada especie a cultivar.
- Se ha logrado diseñar un prototipo que tiene un gran abanico de usos, tanto a nivel educativo, comercial como experimental. Además se han añadido una serie de mejoras pueden permitir obtener un producto de mayor calidad que pueden permitir llegar a comercializar una versión mejorada a la diseñada en este Trabajo de Fin de Grado. Se ha logrado implementar un sistema interactivo, de fácil uso, que permite introducir los datos y controlar el habitáculo sin tener grandes nociones de agricultura, cosa que permite tener huertos urbanos más eficientes
- El precio es un factor bastante determinante a la hora de implementar el prototipo para su comercialización ya que se han utilizado componentes y sensores de precios bastante reducidos que obtienen un gran rendimiento para un uso en el que no se requiera una precisión extrema.
- Actualmente no existen modelos comerciales que implementen la misma lógica para el control de un microclima, por lo que no hay ningún producto muy similar que se pueda ajustar a las características de este prototipo. Los posibles modelos que puedan existir, suelen ser invernaderos para el cultivo industrial, lo que implica que el microclima a controlar es de grandes dimensiones y se necesita una gran superficie, por lo que no se adapta a las características de este prototipo.

10. Bibliografía

En este apartado se enumeran las referencias bibliográficas utilizadas en la realización del Trabajo Fin de Grado. Se presentan ordenadas por orden de aparición en el trabajo.

1. **Guerrero, Pablo.** La guía. [En línea] 9 de 03 de 2012. [Citado el: 27 de 07 de 2016.] http://geografia.laguia2000.com/general/microclima.

2. **Rodríguez, Enrique Reyes.** Wordpress. [En línea] 29 de 05 de 2013. [Citado el: 27 de 07 de 2016.] https://reyesrodriguez.files.wordpress.com/2013/06/tipos-de-climas.pdf.

3. Wikipedia. [En línea] 30 de 07 de 2016. [Citado el: 31 de 07 de 2016.] https://es.wikipedia.org/wiki/Invernadero.

4. Arduino. [En línea] [Citado el: 10 de 06 de 2016.] https://www.arduino.cc/en/Main/ArduinoBoardMega2560.

5. **Liu, Thomas.** Aosong Electronics Co. [En línea] [Citado el: 10 de 06 de 2016.] https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf.

6. Datasheetspdf. [En línea] 13 de 09 de 2011. [Citado el: 10 de 06 de 2016.] http://www.datasheetspdf.com/PDF/SEN92355P/837193/1.

7. **semiconductors, Vishay.** Vihay. [En línea] 23 de 08 de 2011. [Citado el: 10 de 06 de 2016.] http://www.vishay.com/docs/81521/bpw34.pdf.

8. [En línea] [Citado el: 05 de 06 de 2016.] http://diymakers.es/arduino-bluetooth/ .

9. **friends, Apache.** Xampp. [En línea] 2016. [Citado el: 09 de 06 de 2016.] https://www.apachefriends.org/es/index.html.

10. **Infoagro.** Infoagro. [En línea] 27 de 01 de 2015. [Citado el: 09 de 06 de 2016.] http://www.infoagro.com/industria_auxiliar/control_climatico.htm.

11. **Environment, Hydro.** Hydroenv. [En línea] 2016. [Citado el: 09 de 06 de 2016.] http://hydroenv.com.mx/catalogo/index.php?main_page=page&id=221.

12. **Instruments, Texas.** Ti. [En línea] 01 de 03 de 2013. [Citado el: 10 de 06 de 2016.] http://www.ti.com.cn/cn/lit/ds/symlink/lmc6042.pdf.

13. **Rduinostar.** Rduinostar. [En línea] [Citado el: 23 de 06 de 2016.] http://rduinostar.com/documentacion/datasheets/dht22-caracteristicas-am2302/.

14. **Relay, Songle.** datasheetcafe. [En línea] [Citado el: 29 de 06 de 2016.] http://datasheetcafe.databank.netdna-cdn.com/wp-content/uploads/2015/10/SRD-05VDC-SL-C-Datasheet.pdf.

15. **Emerson.** Emerson Climate Technologies. [En línea] 13 de 04 de 2016. [Citado el: 29 de 06 de 2016.] http://www.emersonclimate.com/europe/ProductDocuments/AlcoLiterature/RU_IT_FR_ES_DE_CS_EN_ASC3_Coils_OI.pdf.

16. **Aliexpress.** Aliexpress. [En línea] [Citado el: 16 de 07 de 2016.] http://es.aliexpress.com/item/Mini-Humidifier-DC-5V-Home-Air-Diffuser-USB-Portable-cute-design/32631136501.html?spm=2114.13010608.0.53.26oaWL.

17. **Banggood.** Banggood. [En línea] [Citado el: 15 de 08 de 2016.] http://www.banggood.com/3W-24led-COB-LED-Chip-220mA-WhiteWarm-White-For-DIY-DC-12V-p-959064.html.

18. **Semiconductor, Fairchild.** micropik. [En línea] 01 de 02 de 2000. [Citado el: 13 de 08 de 2016.] http://www.micropik.com/PDF/BD243[1].pdf.

19. **NXP.** NXP. [En línea] 27 de 11 de 2015. [Citado el: 13 de 08 de 2016.] https://www.nxp.com/documents/data_sheet/74HC_HCT04.pdf.

20. **Ilitek.** Electronicavm. [En línea] 20 de 07 de 2011. [Citado el: 10 de 06 de 2016.] https://electronicavm.files.wordpress.com/2015/03/datasheet-chip-lcd-ili9341.pdf.

11. Anexos

En este apartado se incluye toda la información adicional complementaria a la ya explicada en el presente documento. Se adjuntan diferentes planos del prototipo, esquemas y códigos que componen todo el proyecto.

11.1. Código arduino

```
//Programa de control de microclimas en cultivo de plantas
//TFG curso 2015/2016
//Universitat Politècnica de València
//Autor: Carlos Reyes Guerola

//importar librerías
#include <SoftwareSerial.h>//librería para el uso de las
comunicaciones serie
#include <DHT.h>//librería para el uso del sensor DHT22
#include <Adafruit_GFX.h>    // Libreria de graficos
#include <Adafruit_TFTLCD.h> // Libreria de LCD
#include <stdint.h>
#include "TouchScreen.h"//libreria táctil
#include <TimerOne.h> //librería pra poder utilizar interrupciones
temporizadas

//definición de constantes
#define DHTPIN 46//se define el pin del sensor DHT22
#define DHTTYPE DHT22 //Se define el tipo de sensor utilizado(librería
válida para varios sensores de la serie DHT)
#define     BLACK   0x0000  // Se definene los colores
#define     BLUE    0x001F  // que se puedan utilizar para
#define     RED     0xF800  // el texto y los elementos graficos
#define     GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF
 // Pines necesarios para los 4 pines del panel tactil
#define YP A3  // Pin analogico A1 para ADC
#define XM A2  // Pin analogico A2 para ADC
#define YM 9
#define XP 8
short TS_MINX = 190; // Coordenadas del panel tactil para delimitar
```

```
short TS_MINY = 161; // el tamaño de la zona donde se puede presionar
short TS_MAXX = 920; // y que coincida con el tamaño del LCD
short TS_MAXY = 820;
// Se define la presion máxima y minima que podemos realizar sobre el
panel
#define MINPRESSURE 500
#define MAXPRESSURE 1000
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300); // Instancia del
panel tactil (Pin XP, YP, XM, YM, Resistencia del panel)
#define LCD_CS A3   // Se define los pines del LCD
#define LCD_CD A2   // para poder visualizar elementos graficos
#define LCD_WR A1
#define LCD_RD A0
#define LCD_RESET A4 //Pin de reset de programa
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET); //
Instancia LCD

SoftwareSerial BT(52,53); //52 RX, 53 TX. Se determinan los pines de
comunicación bluetooth

//declaración de variables
int X=0; // Variables que almacenaran la coordenada
int Y=0; // X, Y donde se presione y la variable Z
int Z=0; // almacenara la presión realizada
unsigned int i=0;//variable para recorrer matrices
int temp_c=0;//variable para la lectura de la temperatura(consigna)
int hum_suelo_c=0;//variable para la lectura de la humedad del
suelo(consigna)
int hum_aire_c=0;//variable para la lectura de la humedad del
aire(consigna)
int ilum_c=0;//variable para la lectura de la iluminación(consigna)
int cod_planta=0;//variable del identificador de planta
int codigo=0;//variable para la lectura de código de función
float temp=0.0;//variable para la medida de la temperatura
float hum_suelo=0.0;//variable para la medida de la humedad del suelo
float hum_aire=0.0;//variable para la medida de la humedad del aire
float ilum=0.0;//variable para la medida de la iluminación
bool marcha_paro=false;//variable para controlar si el sistema esta en
funcionamiento
String envio="";//variable para el envío de cadenas
int SalidaPWM=0;//variable para la salida PWM
const int analogOutPin = 45; // Analog output pin
int k=0; //variable para recorrer ec. diferencias
int x_hum[4]={0,0,0,0}; //vector del filtro de entrada de los últimos
3 valores
int y_hum[4]={0,0,0,0}; //vector a la salida del filtro de los últimos
3 valores
int x_ilum[4]={0,0,0,0}; //vector del filtro de entrada de los últimos
3 valores
int y_ilum[4]={0,0,0,0}; //vector a la salida del filtro de los
últimos 3 valores
int e[2]={0,0};//vector del error pid
int s[2]={0,0};//vector salida pid
DHT dht(DHTPIN, DHTTYPE,6);//se inicia el objeto del sensor DHT para
el manejo de las funciones.

//Nombre: setup
//Función: Configurar módulos y pines
//Necesita valores: No
//Devuelve valores: No
void setup()
```

```
{
  pinMode(13,OUTPUT); //led indicador de que se ha presionado en la
pantalla táctil
  pinMode(22,OUTPUT); //Salida de la electro válvula.
  pinMode(23,OUTPUT); //Salida del humidificador del aire.
  pinMode(24,OUTPUT); //Salida de las resistencias térmicas para el
control de la temperatura.
  tft.begin(0x9341); // Se inicia el LCD especificando el controlador
ILI9341.
  tft.setRotation(1); // Establecemos la posición de la pantalla
Vertical u Horizontal
  tft.setTextSize(2);    // Se especifica el tamaño del texto
  tft.setTextColor(BLACK); // Se define el color del texto
  tft.fillScreen(WHITE); // Se pinta la pantalla de blanco
  apariencia_pantalla(); //se carga la apariencia de la pantalla
  BT.begin(9600); //Velocidad del puerto del módulo Bluetooth
  dht.begin();//Se inicia el sensor DHT
  Timer1.initialize(2000000); //inicializa la interrupción temporizada
a 2 segundos.
  Timer1.attachInterrupt(control_sistema);// se asigna la interrupción
a la función que realiza el control del sistema.
}//fin void setup

//Nombre: loop
//Función: programa principal en bucle repetitivo
//Necesita valores: No
//Devuelve valores: No
void loop()
{
  while (BT.available()>0)
  {
    codigo=lectura_bt();//se procede a leer primero el código de envío
para saber que función se debe realizar
    accion_realizar(codigo);//llamada a la función que averigua el
código recibido
    codigo=0;//se resetea el código de envio para evitar fallos.
  }//fin if BT.available
  ilum=lectura_porcentual_dir(A7);//se lee la iluminación. Se coloca
en el loop debido a que la luz es una variable de acción rápida.
  x_ilum[0]=ilum;//ajuste variable filtro.
  filtro(x_ilum,y_ilum);//filtrado
  ilum=y_ilum[0];//ajuste magnitud despues filtro.
  accion_pantalla(); //atiende a la lectura de la pantalla y su
interacción.
}//fin void loop

//////////Funciones para las comunicaciones série
bluetooth//////////////////
//Nombre: lectura_bt
//Función: realizar la lectura bluetooth
//Necesita valores: No
//Devuelve valores: valor integer del parámetro necesario
int lectura_bt()
{
  int valor=0; //variable para la devolución de la función
  String cadena; //variable para la cadena de datos completa
  char dato;//variable para la recepción de caracteres
  do //hace el bucle necesario para leer toda la cadena da datos
  {
   dato=BT.read();//lectura de los datos enviados del pc
```

```
   if(dato=='f')//mientras que el dato recibido no sea f se añaden
caracteres a la cadena
   {
     valor=cadena.toInt();//se convierte la cadena de caracteres a un
valor integer
   }
   else
   {
     cadena=cadena+dato;//se añade caracter a la cadena
   }//fin if/else dato=='f'
  }//fin do
  while(dato!='f');//hasta que no se reciba el dato f no se sale del
bucle
  return valor;//devuelve el parámetro para el control
}//fin lectura bt

//Nombre: escritura_bt
//Función: realizar la escritura bluetooth
//Necesita valores: cadena de caracteres string
//Devuelve valores: No
void escritura_bt(String cad)
{
  cad=cad+'f';// se añade a la cadena el parámetro de fin de cadena
  BT.println(cad);//proceso de envío de la trama al pc
}//fin escritura bt

//Nombre: accion_realizar
//Función: realiza las correspondientes acciones a partir de la
lectura bt
//Necesita valores: Entero sin signo que indica el código de acción
//Devuelve valores: Ninguno
void accion_realizar(unsigned int cod)
{
    switch (cod)
    {
      case 110://caso en el que el sistema se pone en funcionamiento
      marcha_paro=true;//se pone a true la variable que determina el
inicio o paro del sistema
      break;
      case 111://caso en el que el sistema se para
      marcha_paro=false;//se pone a false la variable que determina el
inicio o paro del sistema
      break;
      case 112://caso en el que se procede a leer el estado del
sistema
      estado_sistema();//se llama a la función que realiza el envío de
datos del estado del sistema
      break;
      case 113://caso en el que recbien las consignas y código de
planta desde la app en VB
      lectura_consignas();//se llama a la función lectura_consignas
para la lectura de todas las consignas
      break;
      case 114: //caso en el que se averigua el estado del sistema
cuando hay una vinculación con la app en VB
      sistema_marcha_paro();
      break;
      default://caso diferente
      break;
    }//fin switch
}//fin accion_realizar
```

```
//Nombre: lectura_consignas
//Función: realizar la lectura de las consignas desde la app en vb
//Necesita valores: Ninguno
//Devuelve valores: Ninguno
void lectura_consignas()
{
    escritura_bt("1");
    while(BT.available()==0);//mientras que no haya algo en el buffer
de entrada, no se procede a recibir el dato
    cod_planta=lectura_bt();//se procede a leer las consignas y código
de planta
    escritura_bt("2");//se responde al sistema como que ha llegado el
primer dato
    while(BT.available()<1);
    temp_c=lectura_bt();
    escritura_bt("3");
    while(BT.available()<1);
    hum_suelo_c=lectura_bt();
    escritura_bt("4");
    while(BT.available()<1);
    hum_aire_c=lectura_bt();
    escritura_bt("5");
    while(BT.available()<1);
    ilum_c=lectura_bt();
    escritura_bt("0");// Se envía un valor fuera de rango para no
recibir siempre el mismo dato debido al timer de envío en la app de VB
}//fin lectura_consignas

//Nombre: estado_sistema
//Función: Enviar los datos necesarios para observar el estado del
sistema en la app de VB
//Necesita valores: Ninguno
//Devuelve valores: Ninguno
void estado_sistema()
{
  i=1;
     while (i<=5)
     {
      while(BT.available()<1);//mientras que no haya algo en el buffer
de entrada, no se procede a recibir el dato
      codigo=lectura_bt();//se lee el comando de confirmación para
vaciar el buffer de entrada
       switch (codigo)
       {
         case 1:
            envio="";//se deja en blanco el string para poder añadir
la siguiente variable a enviar
            envio=envio+temp;//se añade al string el valor de la
variable a enviar.
            escritura_bt(envio);//se envia el valor deseado
            i++;
            break;
         case 2:
            envio="";
            envio=envio+hum_suelo;//se añade al string el valor de la
variable a enviar.
            escritura_bt(envio);//se envia el valor deseado
            i++;
            break;
         case 3:
```

```
            envio="";
            envio=envio+hum_aire;//se añade al string el valor de la
variable a enviar.
            escritura_bt(envio);//se envia el valor deseado
            i++;
            break;
         case 4:
            envio="";
            envio=envio+ilum;//se añade al string el valor de la
variable a enviar.
            escritura_bt(envio);//se envia el valor deseado
            i++;
            break;
         case 5:
            envio="";
            envio=envio+cod_planta;//se añade al string el valor de la
variable a enviar.
            escritura_bt(envio);//se envia el valor deseado
            i++;
            break;
       }//fin switch
    }//fin while
}

//Nombre: sistema_marcha_paro
//Función: escribe por bluetooth el dato que acutaliza el botón de
marcha paro en la aplicación de VB
//Necesita valores: Ninguno
//Devuelve valores: Ninguno
void sistema_marcha_paro()
{
  if (marcha_paro==true)
  {
    escritura_bt("1");
  }
  else
  {
    escritura_bt("0");
  }
}
//////////Funciones para la lectura de los
sensores//////////////////////////
//Nombre: lectura_porcentual_inv
//Función: realizar la lectura de los sensores que se miden en
porcentaje inversamente proporcional
//Necesita valores: valor integer que define el pin que leer
//Devuelve valores: valor porcentual de tipo decimal
float lectura_porcentual_inv(int pin)
{
  unsigned int lectura=0; //variable que almacena la lectura digital
realizada
  float porcentaje=0;//variable que almacena el valor digital
convertido a porcentaje
  int inv=0;
  lectura=analogRead(pin);//lectura del pin analógico seleccionado
(entre 0 y 1023)
  inv=map(lectura,0,1023,1023,0);//se invierte la medida para que la
relación humedad medida digtal sera directa
  porcentaje=inv*100.0/1023;//si un 100% equivale a 1023 muestras, la
lectura invertida sera X%
  return porcentaje;// se devuelve el porcentaje obtenido.
```

```
}//fin lectura porcentual_inv

//Nombre: lectura_porcentual_dir
//Función: realizar la lectura de los sensores que se miden en
porcentaje directamente proporcional
//Necesita valores: valor integer que define el pin que leer
//Devuelve valores: valor porcentual de tipo decimal
float lectura_porcentual_dir(int pin)
{
  unsigned int lectura=0; //variable que almacena la lectura digital
realizada
  float porcentaje=0;//variable que almacena el valor digital
convertido a porcentaje
  lectura=analogRead(pin);//lectura del pin analógico seleccionado
(entre 0 y 1024)
  porcentaje=lectura*10000/1024;//Si un 10000 lux equivale a 1024
muestras, la lectura directa sera x%
  return porcentaje;// se devuelve el porcentaje obtenido.
}//fin lectura porcentual_dir

//////////Función que realiza el control del sistema/////////////////
//Nombre: control_sistema
//Función: realizar el control del microclima
//Necesita valores: Ninguno
//Devuelve valores: Ninguno
void control_sistema()
{
  //lectura de magnitudes
  hum_aire=dht.readHumidity();//se lee la humedad del aire
  temp=dht.readTemperature();//se lee la temperatura
  hum_suelo=lectura_porcentual_inv(A6);//se lee la humedad del suelo
  x_hum[0]=hum_suelo;
  filtro(x_hum,y_hum);//filtrado
  hum_suelo=y_hum[0];
  if(marcha_paro==true)
  {
    //control todo o nada con histéresis de la temperatura
    if(temp>=temp_c)
    {
      salida_digital_bajo(24);//parar resistencias
    }
    else if(temp<=(temp_c-1))//se considera una histéresis de un grado
ya que se encuentra entre el márgen óptimo de las plantas
    {
      salida_digital_alto(24);//activar resistencias
    }//fin if/else if temp
    //control todo o nada con histéresis de la humedad del aire
    if(hum_aire>=hum_aire_c)
    {
     salida_digital_bajo(23);//parar humidificador
    }
    else if(hum_aire<=(hum_aire_c-3))
    {
      salida_digital_alto(23);//activar humidificador
    }//fin if/else if hum_aire
    //control todo o nada con histéresis de la humedad del suelo
    if(hum_suelo>=hum_suelo_c)
    {
     salida_digital_bajo(22);//parar electrovalvula
    }
    else if(hum_suelo<=(hum_suelo_c-3))
```

```
    {
      salida_digital_alto(22);//activar electrovalvula
    }//fin if/else if hum_suelo
    SalidaPWM=pid(ilum,ilum_c);
    analogWrite(analogOutPin, SalidaPWM); //escritura pwm en el pin
    analogWrite(analogOutPin, (255-SalidaPWM));
  }//while
}//fin void control_sistema

//Nombre: Filtro
//Función: Filtro paso bajo de tercer orden
//Necesita valores: las últimas 4 entradas y salidas
//Devuelve valores: Ninguno
void filtro(int entrada[], int salida[])
{

salida[k]=0.0753*entrada[k]+0.2259*entrada[k+1]+0.2259*entrada[k+2]+0.
0753*entrada[k+3]+0.8266*salida[k+1]-
0.5154*salida[k+2]+0.08648*salida[k+3]; //ecuacion en
                                                        //diferencias
del filtro de 3 er orden
  salida[k+3]=salida[k+2];
  salida[k+2]=salida[k+1];
  salida[k+1]=salida[k];
  entrada[k+3]=entrada[k+2];
  entrada[k+2]=entrada[k+1];
  entrada[k+1]=entrada[k];
}

//Nombre: PID
//Función: Control PID de la iluminación
//Necesita valores: Entrada y consigna
//Devuelve valores: Salida en valor PWM.
int pid (int entrada, int referencia)
 {
  int sal_255=0; //variable para escalado de la salida
  e[k]=referencia-entrada; //se calcula el error
  s[k]=0.072305*e[k]-0.072305*e[k+1]+0.999*s[k+1];//Ec. diferencias
PID
  s[k+1]=s[k]; //ajuste de variables
  e[k+1]=e[k];
  sal_255=(s[k]*255)/10000;// escalado
  return sal_255;
 }
////////////Funciones para la actuvación de las salidas
digitales/analógicas///////
//Nombre: Salida_digital_alto
//Función: activa una salida digital
//Necesita valores: pin a activar
//Devuelve valores: Ninguno
void salida_digital_alto(int pin)
{
  pinMode(pin,OUTPUT);
  digitalWrite(pin,HIGH);
}

//Nombre: Salida_digital_bajo
//Función: desactiva una salida digital
//Necesita valores: pin a activar
//Devuelve valores: Ninguno
void salida_digital_bajo(int pin)
```

```
{
  pinMode(pin,OUTPUT);
  digitalWrite(pin,LOW);
}
////////////Funciones para el control del display
táctil//////////////////////////

//Nombre: apariencia_pantalla
//Función: Carga la apariencia de la pantalla
//Necesita valores: No
//Devuelve valores: No
void apariencia_pantalla()
{
  tft.setCursor(150,5);  // Se coloca el cursor en la posición deseada
(x,y)
  tft.println("Real"); // Se escribe por pantalla
  tft.setCursor(220,5);  // Se coloca el cursor en la posición deseada
(x,y)
  tft.println("Objetivo"); // Se escribe por pantalla
  tft.setCursor(5,35);  // Se coloca el cursor en la posición deseada
(x,y)
  tft.println("Temperatura"); // Se escribe por pantalla
  tft.setCursor(140,35);  // Se coloca el cursor en la posición
deseada (x,y)
  tft.println(temp); // Se escribe por pantalla
  tft.setCursor(230,35);  // Se coloca el cursor en la posición
deseada (x,y)
  tft.println(temp_c); // Se escribe por pantalla
  tft.setCursor(7,65);  // Se coloca el cursor en la posición deseada
(x,y)
  tft.println("Hum. aire"); // Se escribe por pantalla
  tft.setCursor(140,65);  // Se coloca el cursor en la posición
deseada (x,y)
  tft.println(hum_aire); // Se escribe por pantalla
  tft.setCursor(230,65);  // Se coloca el cursor en la posición
deseada (x,y)
  tft.println(hum_aire_c); // Se escribe por pantalla
  tft.setCursor(7,95);  // Se coloca el cursor en la posición deseada
(x,y)
  tft.println("Hum. suelo"); // Se escribe por pantalla
  tft.setCursor(140,95);  // Se coloca el cursor en la posición
deseada (x,y)
  tft.println(hum_suelo); // Se escribe por pantalla
  tft.setCursor(230,95);  // Se coloca el cursor en la posición
deseada (x,y)
  tft.println(hum_suelo_c); // Se escribe por pantalla
  tft.setCursor(5,125);  // Se coloca el cursor en la posición deseada
(x,y)
  tft.println("Iluminacion"); // Se escribe por pantalla
  tft.setCursor(140,125);  // Se coloca el cursor en la posición
deseada (x,y)
  tft.println(ilum); // Se escribe por pantalla
  tft.setCursor(230,125);  // Se coloca el cursor en la posición
deseada (x,y)
  tft.println(ilum_c); // Se escribe por pantalla
  //dibujar botón marcha/paro
  tft.drawRect(100, 170 , 120, 60, BLACK); // Se dibuja un "boton"
  tft.drawRect(101, 171, 118, 58, BLACK); // Se dibuja un "boton" para
dar mayor grosor a las líneas
  if(marcha_paro==true)//se dibujar el considera el estadodel sistema
para dibujar el color adecuado
```

```
  {
    tft.fillRect(102, 172, 116, 56, RED); //Se rellema el fondo de
color rojo
    tft.setCursor(135,190);  // Se coloca el cursor en la posición
deseada (x,y)
    tft.println("Paro"); // Se escribe por pantalla
  }
  else
  {
    tft.fillRect(102, 172, 116, 56, GREEN); //Se rellema el fondo de
color rojo
    tft.setCursor(125,190);  // Se coloca el cursor en la posición
deseada (x,y)
    tft.println("Marcha"); // Se escribe por pantalla
  }
}

//Nombre: accion_pantalla
//Función: Funcionamiento de la pantalla
//Necesita valores: No
//Devuelve valores: No
void accion_pantalla()
{
  lectura_panel();//se lee el punto de contacto
  if((X>77&&X<167)&&(Y>230&&Y<303)&&(Z>MINPRESSURE && Z <
MAXPRESSURE))//condición para volver atrás
  {
    if(marcha_paro==true)
    {
      marcha_paro=false;
      tft.fillRect(102, 172, 116, 56, GREEN); //Se rellema el fondo de
color verde
      tft.setCursor(125,190);  // Se coloca el cursor en la posición
deseada (x,y)
      tft.println("Marcha"); // Se escribe por pantalla
    }
    else
    {
      marcha_paro=true;
      tft.fillRect(102, 172, 116, 56, RED); //Se rellema el fondo de
color rojo
      tft.setCursor(135,190);  // Se coloca el cursor en la posición
deseada (x,y)
      tft.println("Paro"); // Se escribe por pantalla
    }
  }
  else if ((Z>MINPRESSURE && Z < MAXPRESSURE))
  {
    tft.fillRect(135, 30, 180, 120, WHITE); //Se rellema el fondo de
color blanco
    tft.setCursor(140,35);  // Se coloca el cursor en la posición
deseada (x,y)
    tft.println(temp); // Se escribe por pantalla
    tft.setCursor(230,35);  // Se coloca el cursor en la posición
deseada (x,y)
    tft.println(temp_c); // Se escribe por pantalla
    tft.setCursor(140,65);  // Se coloca el cursor en la posición
deseada (x,y)
    tft.println(hum_aire); // Se escribe por pantalla
    tft.setCursor(230,65);  // Se coloca el cursor en la posición
deseada (x,y)
```

```
    tft.println(hum_aire_c); // Se escribe por pantalla
    tft.setCursor(140,95);  // Se coloca el cursor en la posición
deseada (x,y)
    tft.println(hum_suelo); // Se escribe por pantalla
    tft.setCursor(230,95);  // Se coloca el cursor en la posición
deseada (x,y)
    tft.println(hum_suelo_c); // Se escribe por pantalla
    tft.setCursor(140,125);  // Se coloca el cursor en la posición
deseada (x,y)
    tft.println(ilum); // Se escribe por pantalla
    tft.setCursor(230,125);  // Se coloca el cursor en la posición
deseada (x,y)
    tft.println(ilum_c); // Se escribe por pantalla

  }//fin if/else if

}

//Nombre: lectura_panel
//Función: realiza la lectura de un punto de contacto en la pantalla
//Necesita valores: No
//Devuelve valores: No
void lectura_panel()
{
    digitalWrite(13, HIGH);
    TSPoint p = ts.getPoint(); // Se realiza la lectura de las
coordenadas
    digitalWrite(13, LOW);

    pinMode(XM, OUTPUT); // La librería utiliza estos pines como
entrada y salida
    pinMode(YP, OUTPUT); // por lo que es necesario declararlos como
salida justo
                         // despues de realizar una lectura de
coordenadas.

    // Se mapean los valores analogicos leidos del panel tactil (0-
1023)
    // y se convierten en valores correspondientes a la medida del LCD
320x240
    Y = map(p.x, TS_MAXX, TS_MINX, tft.width(), 0); //Se hace un
remapeado a la inversa debido
    X = map(p.y, TS_MINY, TS_MAXY, tft.height(), 0);//a que el driver
reconoce las coordenadas alrevés
    Z = p.z;
}
```

11.2. Código Visual Basic

```
Public Class Pantalla_principal

    Private Sub Pantalla_principal_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
        lb_fecha.Text = WeekdayName(Weekday(Now,
FirstDayOfWeek.Monday), False, FirstDayOfWeek.Monday) & " " & Date.Now
'se carga la fecha actual en la carga del programa
        If estado = True Then
            cmd_marcha.Image = Image.FromFile(Application.StartupPath
& "\imagenes visual basic\stop.png") 'se modifica la imagen a mostra
en el botón.
            cmd_marcha.Text = "Paro"
        Else
```

```
            cmd_marcha.Image = Image.FromFile(Application.StartupPath
& "\imagenes visual basic\play.png") 'se modifica la imagen a mostra
en el botón.
            cmd_marcha.Text = "Marcha"
        End If
    End Sub

    Private Sub Timer_fecha_Tick(sender As Object, e As EventArgs)
Handles Timer_fecha.Tick    'Timer para controlar la función horaria
actual en el programa.
        lb_fecha.Text = WeekdayName(Weekday(Now,
FirstDayOfWeek.Monday), False, FirstDayOfWeek.Monday) & " " & Date.Now
'carga en el label la fecha actual cada 1 segundo
    End Sub

    Private Sub cmd_marcha_Click(sender As Object, e As EventArgs)
Handles cmd_marcha.Click
        If bluetooth.sp_bluetooth.IsOpen = True Then
            If (estado = False) Then
                estado = True
                cmd_estado.Enabled = True
                Call enviar_datos("110") 'se envía el comando para
iniciar el sistema
                'enviar estado marcha a arduino
                cmd_marcha.Image =
Image.FromFile(Application.StartupPath & "\imagenes visual
basic\stop.png") 'se modifica la imagen a mostra en el botón.
                cmd_marcha.Text = "Paro"
            Else
                estado = False
                Call enviar_datos("111") 'se envía el comando para
parar el sistema
                'enviar estado paro a arduino
                cmd_marcha.Image =
Image.FromFile(Application.StartupPath & "\imagenes visual
basic\play.png") 'se modifica la imagen a mostra en el botón.
                cmd_marcha.Text = "Marcha"
            End If
        Else
            MsgBox("El módulo bluetooth no está activado, actívelo
antes", MsgBoxStyle.Critical, "Advertencia")
        End If

    End Sub

    Private Sub cmd_salir_Click(sender As Object, e As EventArgs)
Handles cmd_salir.Click
        respuestamsg = MsgBox("¿Desea salir de la aplicación?",
MsgBoxStyle.YesNo, "Advertencia") 'se muestra mensaje para preguntar
si se desea salir. El resultado se guarda en la variable respuestamsg
        If respuestamsg = MsgBoxResult.Yes Then 'si se da la condición
de que se desea salir, se procederá a cerrar el formulario.
            bluetooth.sp_bluetooth.Close() 'Se cierra la conexión
bluetooth existente
            bluetooth.Close() 'Cierra el formulario que mantiene la
configuración bluetooth
            Me.Close() 'cierra el formulario actual
            End 'finaliza el programa
        End If
    End Sub
```

```
    Private Sub cmd_añadir_Click(sender As Object, e As EventArgs)
Handles cmd_añadir.Click
        añadir_planta.Show() 'Se muestra el formulario principal
        añadir_planta.Size = Me.Size 'Se ajusta el tamaño del
formulario actual al que se va a cambiar
        añadir_planta.Location = Me.Location 'Se ajusta la
localización del formulario actual al que se va a cambiar
        Me.Close()
    End Sub

    Private Sub cmd_seleccion_Click(sender As Object, e As EventArgs)
Handles cmd_seleccion.Click
        Seleccionar_planta.Show() 'Se muestra el formulario principal
        Seleccionar_planta.Size = Me.Size 'Se ajusta el tamaño del
formulario actual al que se va a cambiar
        Seleccionar_planta.Location = Me.Location 'Se ajusta la
localización del formulario actual al que se va a cambiar
        Me.Close() 'Se cierra el formulario actual
    End Sub

    Private Sub cmd_estado_Click(sender As Object, e As EventArgs)
Handles cmd_estado.Click
        estado_sistema.Show() 'Se muestra el formulario principal
        estado_sistema.Size = Me.Size 'Se ajusta el tamaño del
formulario actual al que se va a cambiar
        estado_sistema.Location = Me.Location 'Se ajusta la
localización del formulario actual al que se va a cambiar
        Me.Close() 'Se cierra el formulario actual
    End Sub

    Private Sub cmd_config_Click(sender As Object, e As EventArgs)
Handles cmd_config.Click
        bluetooth.Show() 'Se muestra el formulario principal
        bluetooth.Size = Me.Size 'Se ajusta el tamaño del formulario
actual al que se va a cambiar
        bluetooth.Location = Me.Location 'Se ajusta la localización
del formulario actual al que se va a cambiar
        Me.Close() 'Se cierra el formulario actual
    End Sub

    Private Sub cmd_modificar_Click(sender As Object, e As EventArgs)
Handles cmd_modificar.Click
        modificar_planta.Show() 'Se muestra el formulario principal
        modificar_planta.Size = Me.Size 'Se ajusta el tamaño del
formulario actual al que se va a cambiar
        modificar_planta.Location = Me.Location 'Se ajusta la
localización del formulario actual al que se va a cambiar
        Me.Close()
    End Sub
End Class

Public Class añadir_planta

    Private Sub cmd_volver_Click(sender As Object, e As EventArgs)
Handles cmd_volver.Click
        Pantalla_principal.Show() 'Se muestra el formulario principal
        Pantalla_principal.Size = Me.Size 'Se ajusta el tamaño del
formulario actual al que se va a cambiar
        Pantalla_principal.Location = Me.Location 'Se ajusta la
localización del formulario actual al que se va a cambiar
        Me.Close() 'Se cierra el formulario actual
```

```
    End Sub

    Private Sub Timer_estado_Tick(sender As Object, e As EventArgs)
Handles Timer estado.Tick
        If txt_nombre.Text <> "" And IsNumeric(txt_humedad_aire.Text)
= True And IsNumeric(txt_humedad_suelo.Text) = True And
IsNumeric(txt_iluminacion.Text) = True _
            And IsNumeric(txt_temperatura.Text) = True Then 'se
comprueba si los cuadros están correctamente rellenados
            cmd_guardar.Enabled = True 'si los recuadros están
rellenados con el nombre de la planta y sus valores, se activa el
botón de guardado de datos
        Else
            cmd_guardar.Enabled = False ' en caso contrario, se
desabilita dicho botón
        End If
    End Sub

    Private Sub añadir_planta_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        cmd_guardar.Enabled = False 'inicialmente se desabilita el
botón de guardado.
    End Sub

    Private Sub cmd_guardar_Click(sender As Object, e As EventArgs)
Handles cmd_guardar.Click
        Dim planta_existente As Boolean 'variable para determinar si
existe la planta a guardar
        respuestamsg = MsgBox("¿Desea guardar la planta " &
txt_nombre.Text & "?", MsgBoxStyle.YesNo, "Advertencia") 'se lanza un
mensaje para asegurarse
        'del deseo de introducir esta planta
        If respuestamsg = MsgBoxResult.Yes Then 'si la respuesta es
afirmativa se guardarán los datos en la base de datos.
            conex.ConnectionString = ("data source=localhost; user
id=root; password=''; database=cultivos") 'se configura la conexión a
la base de datos
            sql = "SELECT `Nombre` FROM `plantas`"
            Call lectura_base()
            For i = 0 To dt.Rows.Count - 1 'se recorren todos los
nombres para comprobar si existe la planta que se quiere añadir
                If dt.Rows(0).Item(i) = txt_nombre.Text Then
                    planta_existente = True 'existe una planta con ese
nombre
                Else
                    planta_existente = False 'no existe una planta con
ese nombre
                End If
            Next
            If planta_existente = True Then
                MsgBox("Planta ya existente en la base de datos.",
MsgBoxStyle.Critical, "Error")
            Else
                sql = "INSERT INTO `plantas`(`codigo_planta`,
`Nombre`, `Temperatura`, `Hum_suelo`, `Hum_aire`, `Iluminacion`)" & _
                "VALUES ('" & dt.Rows.Count + 1 & "','" &
txt_nombre.Text & "','" & txt_temperatura.Text & "','" &
txt_humedad_suelo.Text & _
                 "','" & txt_humedad_aire.Text & "','" &
txt_iluminacion.Text & "')" 'se guarda la consulta a realizar
```

```
                Call escritura_base() 'se escribe la consulta en la
base de datos
                txt_nombre.Text = "" 'se borra el texto de todos los
recuadros para introducir otra planta
                txt_humedad_aire.Text = ""
                txt_humedad_suelo.Text = ""
                txt_iluminacion.Text = ""
                txt_temperatura.Text = ""
                txt_nombre.Focus() 'Se selecciona el cuadro de texto
del nombre de la planta para comenzar la introducción de otra planta
            End If
        End If
    End Sub
End Class

Public Class bluetooth

    Private Sub cmd_volver_Click(sender As Object, e As EventArgs)
Handles cmd_volver.Click
        Pantalla_principal.Show() 'Se muestra el formulario principal
        Pantalla_principal.Size = Me.Size 'Se ajusta el tamaño del
formulario actual al que se va a cambiar
        Pantalla_principal.Location = Me.Location 'Se ajusta la
localización del formulario actual al que se va a cambiar
        Me.Hide() 'se oculta el formulario actual para que la
comunicación bluetooth siga en funcionamiento
    End Sub



    Private Sub bluetooth_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        cadena = ""
        strbufferentrada = "" 'Vaciar la variable del bufer de entrada
para recibir sólo lo deseado
        strbuffersalida = "" 'Vaciar la variable del bufer de salida
para recibir sólo lo deseado
        cmd_conectar.Enabled = False 'desabilitar el botón de conectar
por bluetooth ya que no hay un puerto seleccionado

    End Sub

    Private Sub cmd_buscarpuertos_Click(sender As Object, e As
EventArgs) Handles cmd_buscarpuertos.Click
        cmb_puertos.Items.Clear() 'limpia el desplegable
        For Each puertodisponible As String In
My.Computer.Ports.SerialPortNames 'se buscan todos los puertos
existentes en el PC
            cmb_puertos.Items.Add(puertodisponible) 'se añaden todos
los puertos disponibles al desplegable
        Next
        If (cmb_puertos.Items.Count > 0) Then 'en caso de tener un
puerto seleccionado un puerto se habilita el botón de conexión
            cmb_puertos.Text = cmb_puertos.Items(0)
            cmd_conectar.Enabled = True
        Else  'en caso contrario, se muestra un mensaje de error y se
mantiene desactivado el botón de conexión
            MsgBox("No encontrado ningún puerto serie",
MsgBoxStyle.Exclamation, "Advertencia")
            cmd_conectar.Enabled = False
            cmb_puertos.Items.Clear()
```

```
        End If
    End Sub

    Private Sub cmd_conectar_Click(sender As Object, e As EventArgs)
Handles cmd_conectar.Click
        If (cmd_conectar.Text = "Conectar") Then 'caso para la
conexión
            Try
                With sp_bluetooth 'configura la conexión del puerto
serie asociado al módulo bluetooth
                    .BaudRate = 9600
                    .DataBits = 8
                    .Parity = IO.Ports.Parity.None
                    .StopBits = IO.Ports.StopBits.One
                    .PortName = cmb_puertos.Text
                    .Open() 'se abre el puerto
                End With
                cmd_conectar.Text = "Desconectar" ' se cambia el texto
del botón de conexión y la imagen a mostrar
                cmd_conectar.Image =
Image.FromFile(Application.StartupPath & "\imagenes visual
basic\desconectar.png")
            Catch ex As Exception ' en caso de existir un error, se
muestra dicho error y no se hace nada.
                MsgBox(ex.Message, MsgBoxStyle.Critical, "Error")
            End Try
            Call enviar_datos("114") 'se envia el código de estado del
sistema
            timer_estado.Enabled = True 'se habilita el timer que
procederá a leer el estado del sistema.
        ElseIf (cmd_conectar.Text = "Desconectar") Then 'caso de
desconexión
            cmd_conectar.Text = "Conectar" 'se cambia el texto y la
imagen del botón de conexión
            cmd_conectar.Image =
Image.FromFile(Application.StartupPath & "\imagenes visual
basic\conectar.png") 'se modifica la imagen a mostrar
            ' en el botón.
            sp_bluetooth.Close() ' se finaliza la conexión bluetooth
        End If
    End Sub

    Public Sub datorecibido(sender As Object, e As
IO.Ports.SerialDataReceivedEventArgs) Handles
sp_bluetooth.DataReceived
        strbufferentrada = Me.sp_bluetooth.ReadExisting 'cuando el
buffer de datos contiene información, se guardan dichos datos en
        'la variable de entrada.
        Call lectura_datos() 'se procede a leer los datos recibidos.
    End Sub
    Private Sub timer_estado_Tick(sender As Object, e As EventArgs)
Handles timer_estado.Tick
        If cadena_total(0) = "1" Then 'caso en el que el sistema está
en marcha
            estado = True 'se habilita la variable que modifica el
estado del botón en la pantalla principal
            timer_estado.Enabled = False ' se para el timer
        ElseIf (cadena_total(0) = "0") Then 'caso en el que el sistema
está parado
            estado = False 'se deshabilita la variable que modifica el
estado del botón en la pantalla principal
```

```
            timer_estado.Enabled = False ' se para el timer
        End If
    End Sub
End Class

Public Class estado_sistema

    Private Sub cmd_volver_Click(sender As Object, e As EventArgs)
Handles cmd_volver.Click
        Pantalla_principal.Show() 'Se muestra el formulario principal
        Pantalla_principal.Size = Me.Size 'Se ajusta el tamaño del
formulario actual al que se va a cambiar
        Pantalla_principal.Location = Me.Location 'Se ajusta la
localización del formulario actual al que se va a cambiar
        Me.Close() 'Se cierra el formulario actual
    End Sub


    Private Sub cmd_sincronizar_Click(sender As Object, e As
EventArgs) Handles cmd_sincronizar.Click
        If bluetooth.sp_bluetooth.IsOpen = True Then
            cad = ""
            Call enviar_datos("112") ' se envía el comando para
proceder a la recepción de datos
            i = 1
            Call enviar_datos("1")
            Timer_recepcion.Enabled = True
        Else
            MsgBox("El módulo bluetooth no está activado, actívelo
antes", MsgBoxStyle.Critical, "Advertencia")
        End If
    End Sub

    Private Sub cmd_paro_Click(sender As Object, e As EventArgs)
Handles cmd_paro.Click
        If bluetooth.sp_bluetooth.IsOpen = True Then
            Call enviar_datos("111")
            estado = False
        Else
            MsgBox("El módulo bluetooth no está activado, actívelo
antes", MsgBoxStyle.Critical, "Advertencia")
        End If

    End Sub

    Private Sub Timer_recepcion_Tick(sender As Object, e As EventArgs)
Handles Timer_recepcion.Tick
        Select Case i
            Case 1

                If (cad = "f") Then
                    temperatura = cadena_total(0) 'se guarda el dato
recibido
                    Call enviar_datos("2") ' se envía un comando de
confirmación de recepción
                    i = i + 1
                End If
            Case 2
                If (cad = "f") Then
                    humedad_suelo = cadena_total(0) 'se guarda el dato
recibido
```

```
                    Call enviar_datos("3") ' se envía un comando de
confirmación de recepción
                    i = i + 1
                End If
            Case 3
                If (cad = "f") Then
                    humedad_aire = cadena_total(0) 'se guarda el dato
recibido
                    Call enviar_datos("4") ' se envía un comando de
confirmación de recepción
                    i = i + 1
                End If
            Case 4
                If (cad = "f") Then
                    iluminacion = cadena_total(0) 'se guarda el dato
recibido
                    Call enviar_datos("5") ' se envía un comando de
confirmación de recepción
                    i = i + 1
                End If
            Case 5
                If (cad = "f") Then
                    codigo_planta = cadena_total(0) 'se guarda el dato
recibido
                    i = i + 1
                    txt_temperatura.Text = temperatura
                    txt_humedad_suelo.Text = humedad_suelo
                    txt_humedad_aire.Text = humedad_aire
                    txt_luz.Text = iluminacion
                    conex.ConnectionString = ("data source=localhost;
user id=root; password=''; database=cultivos") 'se configura
                    ' la conexión a la base de datos
                    sql = "SELECT `Temperatura`, `Hum_suelo`,
`Hume_aire`, `Iluminacion` FROM `plantas` WHERE `codigo_planta`='" & _
                        codigo_planta & "'" 'se guarda la consulta a
realizar
                    If codigo_planta = 0 Then
                        MsgBox("No hay datos de ninguna planta en el
sistema de control", MsgBoxStyle.Information, "Advertencia")
                    Else
                        Call lectura_base()
                        txt_obj_temp.Text = dt.Rows(0).Item(0)
                        txt_obj_hr_suelo.Text = dt.Rows(0).Item(1)
                        txt_obj_hr_aire.Text = dt.Rows(0).Item(2)
                        txt_obj_luz.Text = dt.Rows(0).Item(3)
                    End If
                    Timer_recepcion.Enabled = False
                End If
        End Select
    End Sub
End Class

Public Class modificar_planta

    Private Sub cmd_volver_Click(sender As Object, e As EventArgs)
Handles cmd_volver.Click
        Pantalla_principal.Show() 'Se muestra el formulario principal
        Pantalla_principal.Size = Me.Size 'Se ajusta el tamaño del
formulario actual al que se va a cambiar
        Pantalla_principal.Location = Me.Location 'Se ajusta la
localización del formulario actual al que se va a cambiar
```

```
        Me.Close() 'Se cierra el formulario actual
    End Sub

    Private Sub modificar_planta_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
        cmd_guardar.Enabled = False 'inicialmente se desabilita el
botón de guardado.
        txt_humedad_aire.Visible = False 'se ponen invisibles los
textos y los cuadros inicialmente para poder seleccionar primero el
item deseado
        txt_humedad_suelo.Visible = False
        txt_iluminacion.Visible = False
        txt_temperatura.Visible = False
        lb_humedad_aire.Visible = False
        lb_humedad_suelo.Visible = False
        lb_iluminacion.Visible = False
        lb_temperatura.Visible = False
        conex.ConnectionString = ("data source=localhost; user
id=root; password=''; database=cultivos") 'se configura la conexión
        'a la base de datos
        sql = "SELECT `Nombre` FROM `plantas`" 'se consultan los
nombres de las plantas
        Call lectura_base() 'se extraen los datos de la base de datos
        For i = 0 To dt.Rows.Count - 1
            cmb_planta.Items.Add(dt.Rows(i).Item(0)) 'se añaden los
nombres de las plantas al desplegable.
        Next
    End Sub

    Private Sub Timer_estado_Tick(sender As Object, e As EventArgs)
Handles Timer_estado.Tick
        If cmb_planta.SelectedIndex > -1 And
IsNumeric(txt_humedad_aire.Text) = True And
IsNumeric(txt_humedad_suelo.Text) = True And
IsNumeric(txt_iluminacion.Text) = True And
IsNumeric(txt_temperatura.Text) = True Then 'se comprueba si los
cuadros están correctamente rellenados
            cmd_guardar.Enabled = True 'si los recuadros están
rellenados con el nombre de la planta y sus valores, se activa el
botón de guardado de datos
        Else
            cmd_guardar.Enabled = False ' en caso contrario, se
desabilita dicho botón
        End If
    End Sub

    Private Sub cmd_guardar_Click(sender As Object, e As EventArgs)
Handles cmd_guardar.Click
        respuestamsg = MsgBox("¿Desea actualizar los datos de la
planta " & cmb_planta.SelectedItem & "?", MsgBoxStyle.YesNo,
"Advertencia") 'se
        ' lanza un mensaje para asegurarse del deseo de introducir
esta planta
        If respuestamsg = MsgBoxResult.Yes Then 'si la respuesta es
afirmativa se guardarán los datos en la base de datos.
            conex.ConnectionString = ("data source=localhost; user
id=root; password=''; database=cultivos") 'se configura la conexión
            ' a la base de datos
            sql = "UPDATE `plantas` SET `Temperatura`='" &
txt_temperatura.Text & "',`Hum_suelo`='" & txt_humedad_suelo.Text & _
```

```
                "',`Hum_aire`='" & txt_humedad_aire.Text &
"',`Iluminacion`='" & txt_iluminacion.Text & "' WHERE `Nombre`='" & _
                cmb_planta.SelectedItem & "'" 'se guarda la consulta a
realizar
            Call actualizar_base() 'se actualiza la consulta en la
base de datos
            txt_humedad_aire.Text = "" 'se borra el texto de todos los
recuadros para introducir otra planta
            txt_humedad_suelo.Text = ""
            txt_iluminacion.Text = ""
            txt_temperatura.Text = ""
            cmb_planta.SelectedIndex = -1 'se pone el índice vacío del
desplegable para poder volver a iniciar la secuencia de modificación
            cmb_planta.SelectedText = "" 'se borra el contenido del
desplegable
        End If
    End Sub

    Private Sub cmb_planta_SelectedIndexChanged(sender As Object, e As
EventArgs) Handles cmb_planta.SelectedIndexChanged
        If cmb_planta.SelectedIndex = -1 Then 'si el índice es nulo no
se muestran los textos y campos de texto
            txt_humedad_aire.Visible = False
            txt_humedad_suelo.Visible = False
            txt_iluminacion.Visible = False
            txt_temperatura.Visible = False
            lb_humedad_aire.Visible = False
            lb_humedad_suelo.Visible = False
            lb_iluminacion.Visible = False
            lb_temperatura.Visible = False
        Else ' en caso contrario, es decir, se ha seleccionado una
planta, se muestran los textos y campos de texto
            txt_humedad_aire.Visible = True
            txt_humedad_suelo.Visible = True
            txt_iluminacion.Visible = True
            txt_temperatura.Visible = True
            lb_humedad_aire.Visible = True
            lb_humedad_suelo.Visible = True
            lb_iluminacion.Visible = True
            lb_temperatura.Visible = True
        End If

    End Sub
End Class

Public Class Seleccionar_planta

    Private Sub cmd_volver_Click(sender As Object, e As EventArgs)
Handles cmd_volver.Click
        Pantalla_principal.Show() 'Se muestra el formulario principal
        Pantalla_principal.Size = Me.Size 'Se ajusta el tamaño del
formulario actual al que se va a cambiar
        Pantalla_principal.Location = Me.Location 'Se ajusta la
localización del formulario actual al que se va a cambiar
        Me.Close() 'Se cierra el formulario actual
    End Sub

    Private Sub Seleccionar_planta_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
        cmd_enviar.Enabled = False 'inicialmente se desabilita el
botón de guardado.
```

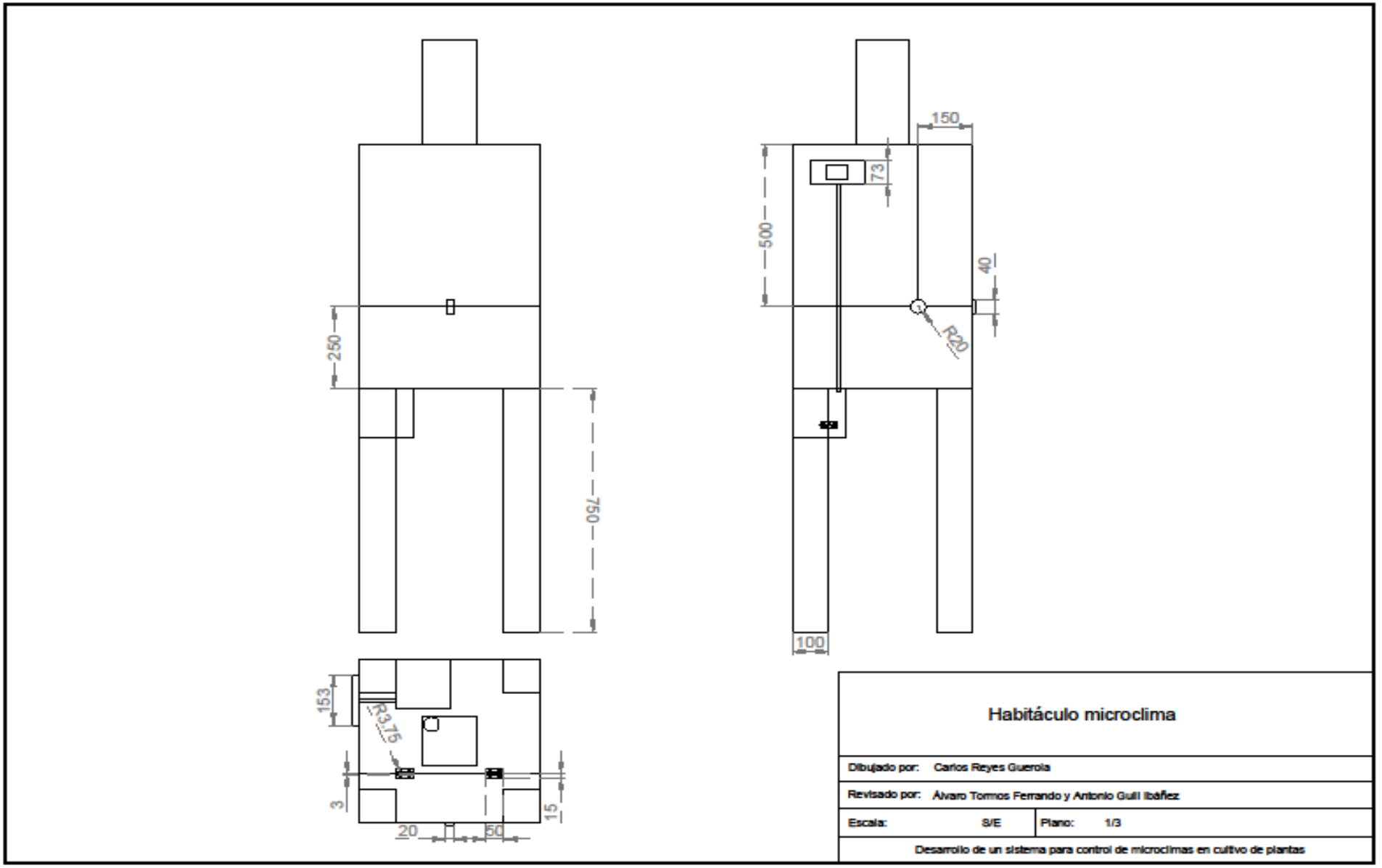
```
        txt_humedad_aire.Visible = False 'se ponen invisibles los
textos y los cuadros inicialmente para poder seleccionar primero el
item deseado, además de inabilitarlos
        txt_humedad_suelo.Visible = False
        txt_iluminacion.Visible = False
        txt_temperatura.Visible = False
        lb_humedad_aire.Visible = False
        lb_humedad_suelo.Visible = False
        lb_iluminacion.Visible = False
        lb_temperatura.Visible = False
        txt_humedad_aire.Enabled = False
        txt_humedad_suelo.Enabled = False
        txt_iluminacion.Enabled = False
        txt_temperatura.Enabled = False
        conex.ConnectionString = ("data source=localhost; user
id=root; password=''; database=cultivos") 'se configura la conexión a
la base de datos
        sql = "SELECT `Nombre` FROM `plantas`" 'se consultan los
nombres de las plantas
        Call lectura_base() 'se extraen los datos de la base de datos
        For i = 0 To dt.Rows.Count - 1
            cmb_planta.Items.Add(dt.Rows(i).Item(0)) 'se añaden los
nombres de las plantas al desplegable.
        Next
    End Sub

    Private Sub cmb_planta_SelectedIndexChanged(sender As Object, e As
EventArgs) Handles cmb_planta.SelectedIndexChanged
        If cmb_planta.SelectedIndex = -1 Then 'si el índice es nulo no
se muestran los textos y campos de texto
            txt_humedad_aire.Visible = False
            txt_humedad_suelo.Visible = False
            txt_iluminacion.Visible = False
            txt_temperatura.Visible = False
            lb_humedad_aire.Visible = False
            lb_humedad_suelo.Visible = False
            lb_iluminacion.Visible = False
            lb_temperatura.Visible = False
        Else ' en caso contrario, es decir, se ha seleccionado una
planta, se muestran los textos, campos de texto y se obtienen los
datos de la planta elegida
            txt_humedad_aire.Visible = True
            txt_humedad_suelo.Visible = True
            txt_iluminacion.Visible = True
            txt_temperatura.Visible = True
            lb_humedad_aire.Visible = True
            lb_humedad_suelo.Visible = True
            lb_iluminacion.Visible = True
            lb_temperatura.Visible = True
            conex.ConnectionString = ("data source=localhost; user
id=root; password=''; database=cultivos") 'se configura la conexión a
la base de datos
            sql = "SELECT `codigo_planta`, `Temperatura`, `Hum_suelo`,
`Hum_aire`, `Iluminacion` FROM `plantas` WHERE `Nombre`='" &
cmb_planta.SelectedItem & "'" 'se consultan los datos de la planta
seleccionada
            Call lectura_base() 'se procede a leer la base de datos
            codigo_planta = dt.Rows(0).Item(0)
            txt_temperatura.Text = dt.Rows(0).Item(1) 'se asigna la
temperatura de la planta seleccionada
```

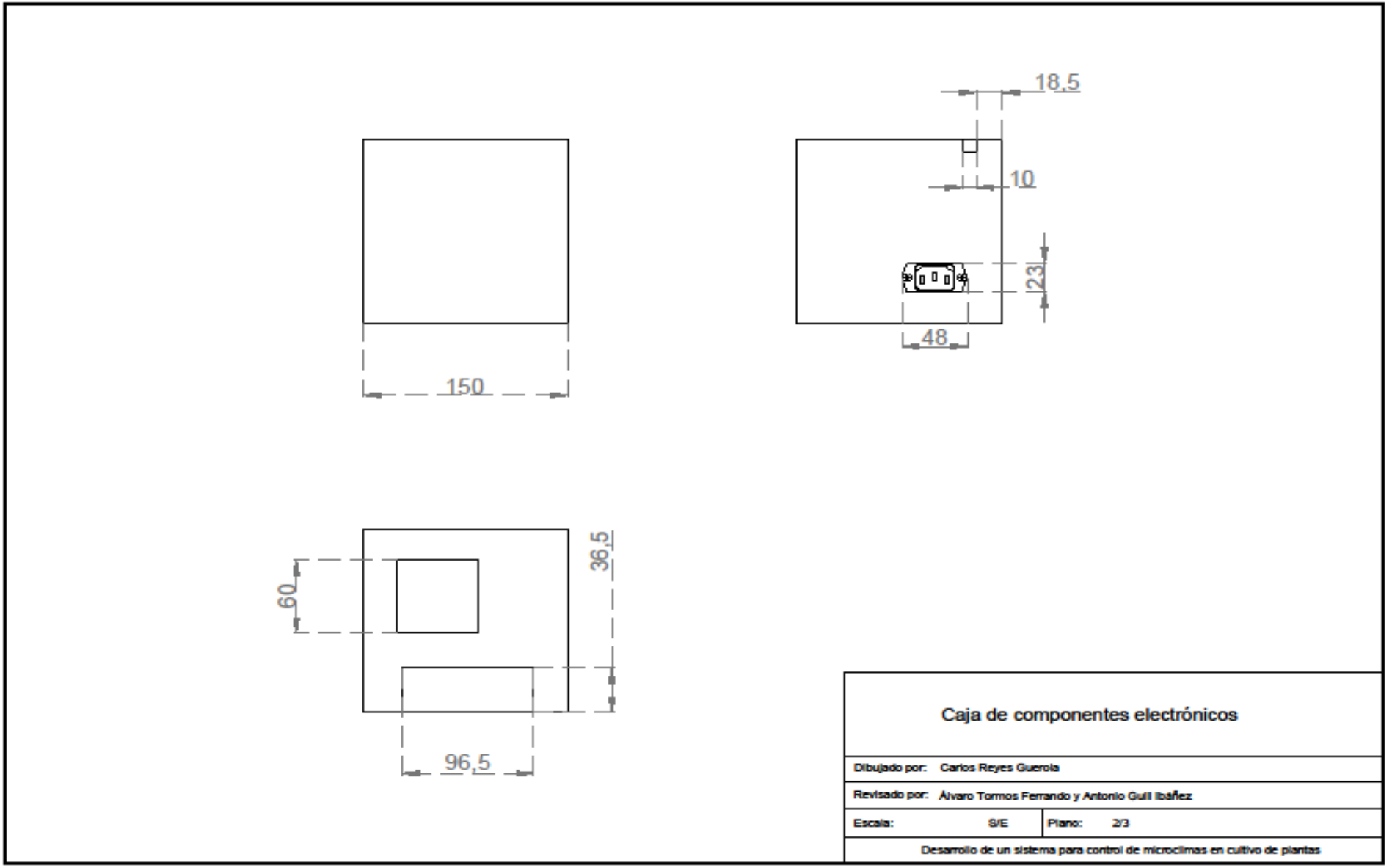
```
            txt_humedad_suelo.Text = dt.Rows(0).Item(2) 'se asigna la
humedad del suelo de la planta seleccionada
            txt_humedad_aire.Text = dt.Rows(0).Item(3) 'se asigna la
humedad del aire de la planta seleccionada
            txt_iluminacion.Text = dt.Rows(0).Item(4) 'se asigna la
iluminación de la planta seleccionada
        End If
    End Sub

    Private Sub Timer_estado_Tick(sender As Object, e As EventArgs)
Handles Timer_estado.Tick
        If cmb_planta.SelectedIndex > -1 And
IsNumeric(txt_humedad_aire.Text) = True And
IsNumeric(txt_humedad_suelo.Text) = True _
            And IsNumeric(txt_iluminacion.Text) = True And
IsNumeric(txt_temperatura.Text) = True Then 'se comprueba si los
            ' cuadros están correctamente rellenados
            cmd_enviar.Enabled = True 'si los recuadros están
rellenados con el nombre de la planta y sus valores, se
            ' activa el botón de enviado de datos
        Else
            cmd_enviar.Enabled = False ' en caso contrario, se
desabilita dicho botón
        End If
        Select Case cadena_total(0)
            Case 1
                Call enviar_datos(codigo_planta) 'se envía el código
de la planta elegida
            Case 2
                Call enviar_datos(txt_temperatura.Text) 'se envía la
temperatura objetivo
                txt_temperatura.Text = "" 'se borra el texto del
recuadros
            Case 3
                Call enviar_datos(txt_humedad_suelo.Text) 'se envía la
humedad del suelo objetivo
                txt_humedad_suelo.Text = "" 'se borra el texto del
recuadros
            Case 4
                Call enviar_datos(txt_humedad_aire.Text) 'se envía la
humedad del aire objetivo
                txt_humedad_aire.Text = "" 'se borra el texto del
recuadros
            Case 5
                Call enviar_datos(txt_iluminacion.Text) 'se envía la
iluminación objetivo
                txt_iluminacion.Text = "" 'se borra el texto del
recuadros
        End Select
    End Sub

    Private Sub cmd_enviar_Click(sender As Object, e As EventArgs)
Handles cmd_enviar.Click
        respuestamsg = MsgBox("¿Desea enviar los datos de la planta "
& cmb_planta.SelectedItem & "?", MsgBoxStyle.YesNo, "Advertencia") 'se
lanza
        ' un mensaje para asegurarse del deseo de introducir esta
planta
        If respuestamsg = MsgBoxResult.Yes Then 'si la respuesta es
afirmativa se guardarán los datos en la base de datos.
            If (bluetooth.sp_bluetooth.IsOpen = True) Then
```

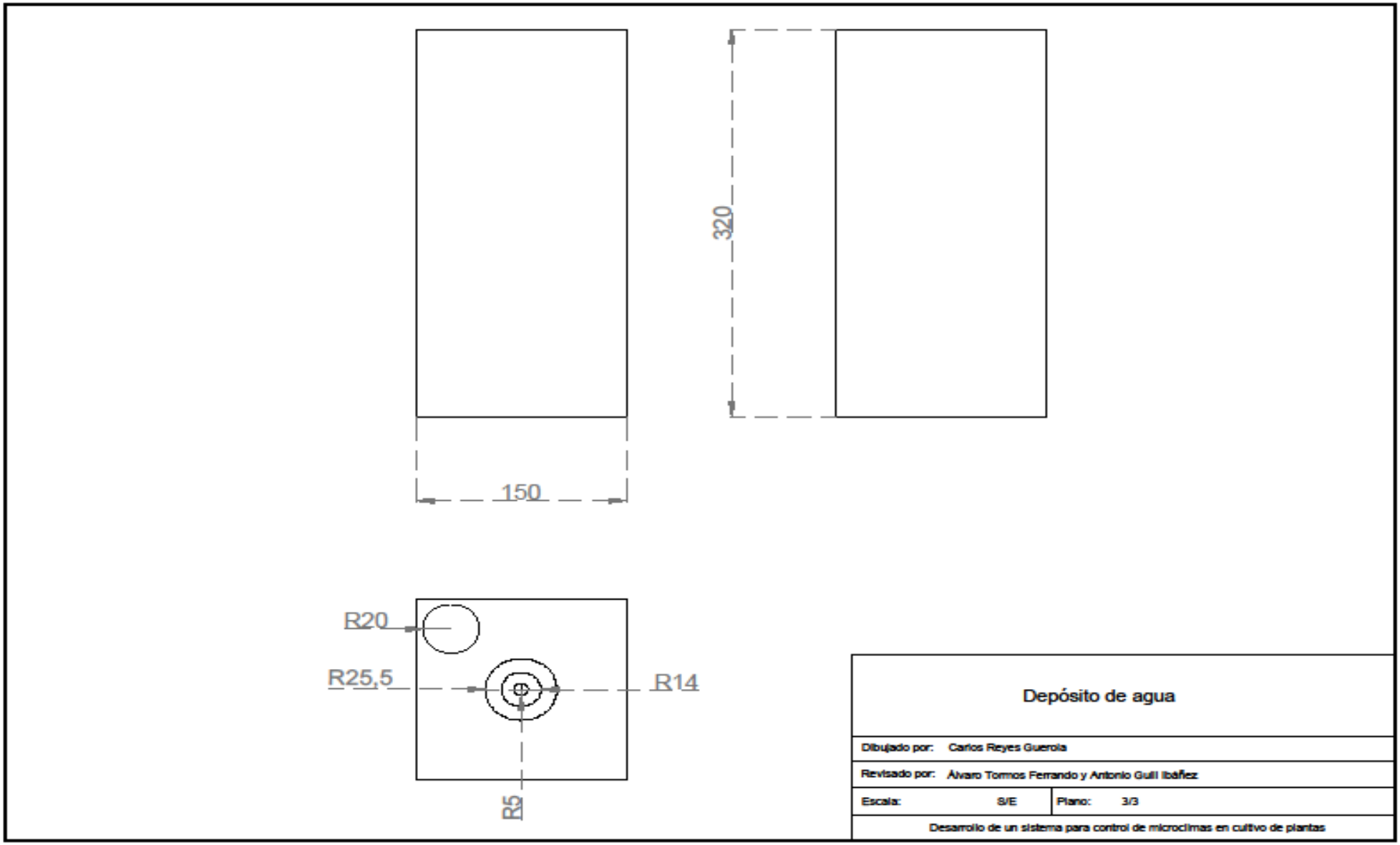
```
                Call enviar_datos("113") 'se envía el código para que
el arduino proceda a guardar las consignas
                cmb_planta.SelectedIndex = -1 'se pone el índice vacío
del desplegable para poder volver a iniciar la secuencia de
modificación
                cmb_planta.SelectedText = "" 'se borra el contenido
del desplegable
            Else
                MsgBox("El módulo bluetooth no está activado, actívelo
antes", MsgBoxStyle.Critical, "Advertencia")
            End If
        End If
    End Sub
End Class

Module Variables
    Public respuestamsg As MsgBoxResult 'Variable utilizada para
obtener una respuesta cuando se pregunta algo con un msgbox
    Public strbufferentrada As String 'Variable en la que se guardan
los datos recibidos en el buffer del módulo bluetooth
    Public strbuffersalida As String 'variable en la que se guardan
los datos a enviar por bluetooth
    Public cadena As String 'variable utilizada para componer toda la
cadena recibida por bluetooth
    Public cadena_total(2) As String 'matriz destinada a separar el
final de cadena de la cadena
    Public i As Integer 'variable para recorrer matrices
    Public temperatura As String 'variable que guarda la temperatura
actual del sistema
    Public humedad_suelo As String 'variable que guarda la humedad del
suelo actual del sistema
    Public humedad_aire As String 'variable que guarda la humedad del
aire actual del sistema
    Public iluminacion As String 'variable que guarda la iluminación
actual del sistema
    Public codigo_planta As String 'variable que guarda el código de
la planta actual en el sistema
    Public cad As String 'variable para almacenar el carácter de final
de trama
    Public estado As Boolean 'Variable para el estado del sistema

    Public Sub lectura_datos()
        cadena = cadena & strbufferentrada
        cad = ""
        If (cadena.Contains("f")) Then
            cadena_total = Split(cadena, "f", 2)
            cad = "f"
            cadena = ""
            strbufferentrada = ""
        End If
    End Sub
    Public Sub enviar_datos(cad As String)
        bluetooth.sp_bluetooth.DiscardOutBuffer()
        strbuffersalida = cad
        strbuffersalida = strbuffersalida & "f"
        bluetooth.sp_bluetooth.Write(strbuffersalida)
        strbuffersalida = ""
    End Sub
End Module
```

```
Imports MySql.Data.MySqlClient 'Importa la librería de mysql para
poder utilizar los tipos de datos
Module Funciones_base_datos
    Public conex As New MySqlConnection 'estructura conexión base de
datos("data source=localhost;user id=root; password='';
database=españa") formato ejemplo
    Public da As MySqlDataAdapter 'estructura de la consulta y la
conexión de la base de datos
    Public dt As DataTable 'variable para la lectura de las tablas
    Public dt2 As DataTable 'variable para la lectura de las tablas
    Public sql As String 'variable que contiene la esctructura de la
consulta
    Public Sub escritura_base()
        Try
            da = New MySqlDataAdapter(sql, conex) 'se configura la
estructura de la consulta y conexión
            dt = New DataTable 'se crea la tabla a subir datos
            da.Fill(dt) 'realiza la conexión con la estructura de la
consulta a la base de datos
        Catch ex As Exception 'en caso de error se muestra un mensaje
            If ex.Message = "No hay ninguna fila en la posición 0."
Then
            Else
                MsgBox(ex.Message)
            End If
        End Try
    End Sub
    Public Sub lectura_base()
        Try
            da = New MySqlDataAdapter(sql, conex) 'se configura la
estructura de la consulta y conexión
            dt = New DataTable 'se crea la tabla a leer datos
            dt2 = New DataTable 'se crea una segunda tabla para evitar
que se borre la tabla leída para poder escribirla de nuevo
            da.Fill(dt2) 'realiza la conexión con la estructura de la
consulta a la base de datos
            da.Fill(dt) 'realiza la conexión con la estructura de la
consulta a la base de datos
        Catch ex As Exception 'en caso de error se muestra un mensaje
            MsgBox(ex.Message)
        End Try
    End Sub
    Public Sub actualizar_base()
        Try
            da = New MySqlDataAdapter(sql, conex) 'se configura la
estructura de la consulta y conexión
            dt = New DataTable 'se crea la tabla a subir datos
            da.Fill(dt) 'realiza la conexión con la estructura de la
consulta a la base de datos
        Catch ex As Exception 'en caso de error se muestra un mensaje
            MsgBox(ex.Message)
        End Try
    End Sub
End Module
```

11.3. Planos en Autocad del habitáculo


150
73
500
40
R20
250
750
100
153
R375
3
15
20
50
Habitáculo microclima
Dibujado por: Carlos Reyes Guerola
Revisado por: Álvaro Tormos Ferrando y Antonio Gull Ibáñez
Escala: S/E
Plano: 1/3
Desarrollo de un sistema para control de microclimas en cultivo de plantas


18,5
10
23
48
150
36,5
60
96,5
Caja de componentes electrónicos
Dibujado por: Carlos Reyes Guerola
Revisado por: Álvaro Tormos Ferrando y Antonio Guill Ibáñez
Escala: S/E
Plano: 2/3
Desarrollo de un sistema para control de microclimas en cultivo de plantas


320
150
R20
R25,5
R14
R5
Depósito de agua
Dibujado por: Carlos Reyes Guerola
Revisado por: Álvaro Tormos Ferrando y Antonio Gull Ibáñez
Escala: S/E
Plano: 3/3
Desarrollo de un sistema para control de microclimas en cultivo de plantas

11.4. Esquema eléctrico en Proteus