UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA QUÍMICA

DISEÑO DEL SISTEMA DE CONTROL DE UN FERMENTADOR PARA ELABORACIÓN DOMÉSTICA DE CERVEZA

AUTOR: MIGUEL MARTÍNEZ TOMÁS

TUTOR: JAVIER SANCHÍS SÁEZ

COTUTOR: NOMBRE DEL COTUTOR

Curso Académico: 2014-15

CONTENIDO

RESUMEN

En el presente trabajo se pretende integrar la tecnología de controladores PIC de bajo costo conocidos como Arduino con el proceso tradicional de fabricación de cerveza de fermentación alta, de cara a la fabricación de reactores domésticos de fermentación que simplifiquen la entrada al mundo del homebrewing al público.

Para ello se ha diseñado un código en el lenguaje nativo de la plataforma capaz de controlar un reactor piloto fabricado a efecto de pruebas para este proyecto con capacidad para 12 litros de producto finalizado. La intención es conseguir un producto comercial con un precio de venta inferior a las alternativas que se pueden encontrar en el mercado aprovechando la versatilidad de la plataforma para integrar todos los subsistemas dentro de una misma unidad de control simplificando por tanto la solución en cuanto a hardware.

El objetivo de la aplicación y el control del proceso se ha conseguido con creces, simplificando además al máximo los procedimientos de control dado que no se requiere una precisión extrema (hay que tener en cuenta que el proceso tradicional es completamente artesano). Por otro lado, al tratarse ésta de una planta piloto, como se puede comprobar en el presupuesto no ha sido posible conseguir un producto ampliamente más económico, con lo que se necesitaría del fenómeno de economía de escala para poder entrar a competir en este mercado por precio. Aún así, el precio de salida estaría en el orden de magnitud que se maneja ahora mismo, dando otros beneficios como el control remoto de la aplicación o el potencial de crear una comunidad dada la versatilidad del equipo, todo ello gracias a la innovación en el tipo de controlador empleado.

RESUM

En aquest treball es pretén integrar la tecnologia de controladors PIC de baix cost coneguts comercialment com Arduino en el procés tradicional de fabricació de cervesa de fermentació alta, pensant en la fabricació de reactors domèstics de fermentació que simplifiquen l'accés del públic al món del homebrewing.

Per això s'ha dissenyat un codi en el llenguatge de programació natiu de la plataforma amb la capacitat de controlar una planta experimental fabricada especialment per a fer les probes pertinents, amb capacitat per a 12 litres de producte finalitzat. La intenció es aconseguir un producte comercial amb un preu de venta al públic inferior a les alternatives que es poden trobar actualment al mercat, aprofitant la versatilitat de la plataforma per a integrar tots els subsistemes dins d'una mateixa unitat de control, simplificant així la solució de hardware.

L'objectiu de la aplicació i el control del procediment de fabricació s'ha aconseguit sense cap problema, simplificant a més al màxim els procediments de control, ja que no es requereix massa precisió. Cal tindre en compte que el procés tradicional es completament artesanal. D'altra banda, al tractar-se d'una planta experimental com es pot comprovar al pressupost no ha sigut possible aconseguir un producte significativament més econòmic, seria, per tant necessari el fenomen d'economia d'escala per tal de competir per preu al mercat. Així i tot, el preu d'eixida prop dels que podem trobar al sector, tenint a més altres beneficis com la capacitat per visualitzar remotament els paràmetres de procés i la possibilitat de crear una comunitat d'usuaris col·laborativa, tot això gràcies a la innovació en la tecnologia de controladors utilitzada.

ABSTRACT

The main objective of this work is the integration between the low cost PIC controllers technology called Arduino and the traditional home brewing process for the top fermented beer, in order to manufacture home brew reactors that simplify the access to the homebrewing world to our customers.

For that, we have designed a code in the native language of the platform, that can control an experimental test plant built in order to test itself, with 12 liters of final product capacity. Our intentions was to get a commercial system with a price significantly lower than the actual alternatives in the market, increasing its functionality by integrating all the subroutines into a single control unit simplifying that way the hardware we need.

The application and control objective is well accomplished, simplifying too the control routines because of the low accuracy required in the process (the traditional one is full hand-made). In the other hand, the economic objective couldn't be satisfied while the system built for the tests is just a prototype that has not the benefits from the scale economy. Despite this, with all the manufacture hand-made, and with the code finished the selling price is in the market range right now, obtaining other benefits as the remote supervision or the potential users community thanks to the innovation in the control system used.

INTRODUCCIÓN:

1.1 OBJETIVO DEL TFG:

En el presente trabajo se pretende desarrollar un sistema de control automatizado para el proceso de fermentación de la cerveza tipo ale, de forma que cualquier persona con unos conocimientos mínimos (adquiribles mediante una guía incluida con el producto completo) se capaz de fabricar de forma satisfactoria, un lote de cerveza de calidad artesana con el mínimo esfuerzo.

Como ampliación del objetivo principal, se pretende que el automatismo sea lo suficientemente adaptable como para poder producir cualquier cerveza de fermentación superior variando unos pocos parámetros que irán codificados en archivos de texto descargables desde una comunidad online.

El ámbito de aplicación del sistema desarrollado en principio es puramente doméstico, donde el auge del home-brewing (fabricación de cerveza artesana en casa) ha creado un nicho de mercado para este tipo de equipos, donde es posible entrar actualmente e introducir ventajas competitivas, en el caso del sistema proyectado, precio y automatización.

El ámbito de aplicación de las versiones posteriores está previsto que se extienda al sector de la microcervecería, donde cada pequeño productor podrá generar su propio archivo de configuración para producir según sus estándares.

Los resultados obtenidos en este TFG, demuestran que el sistema Arduino es perfectamente capaz de controlar un reactor de este tipo, y que su fabricación en masa es perfectamente viable. Su escalado, por los recursos del sistema disponibles una vez terminada la compilación del programa parece completamente viable así mismo. El resultado obtenido en la fermentación tras la prueba del programa en una planta piloto presenta unas propiedades organolépticas comparables a cualquier cerveza artesana comercial.

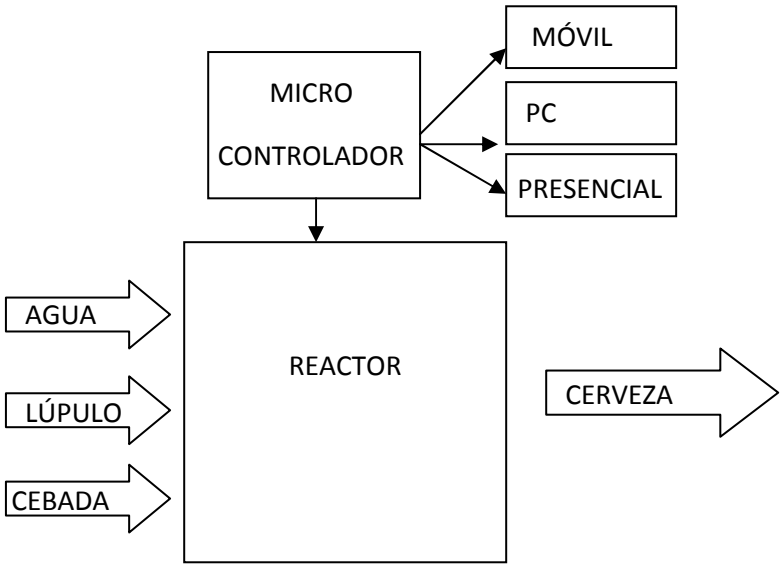


Fig. 1. Esquema general

1.2.DESCRIPCIÓN DE LA PLANTA PILOTO:

Para el estudio del comportamiento del bucle en condiciones lo más próximas a la realidad posible se ha desarrollado y construido una planta piloto consistente en un reactor de fermentación fabricado en acero AISI 304.

Para ello se ha tomado como referencia un contenedor fabricado en acero inoxidable AISI 304 con una capacidad real de 30 litros, que se ha introducido dentro de un contenedor de dimensiones superiores fabricado esta vez en chapa de acero galvanizado, debido a que no estará en condiciones normales de operación en contacto con el producto final. La unión entre ambos contenedores se realiza suspendiendo en el aire y forzando a mantener una postura centrada al tanque de inoxidable mediante tacos de madera atornillados al contenedor externo. De esta manera se consigue una cámara de aire que contribuirá al aislamiento térmico del contenedor, contando con la baja conductividad térmica de la madera como aliciente para su uso frente al metal.

Sobre ambos contenedores se instalará una tapa de acero inoxidable donde se instala parte de la instrumentación, las tomas de líquido tanto refrigerante como alimentario y los actuadores para el manejo de sólidos. Esta tapa, a su vez está divida en dos partes, de forma que es fácilmente abatible para ver el interior de la cámara de reacción, aprovechando su abertura para la entrada del cable del manejo de cebada.

El interior de la cámara de reacción cuenta además con una resistencia de 1kW para uso alimentario con una vaina fabricada en acero inoxidable, capaz de calentar todo el volumen de fluido en un tiempo prudencial, contando con la agitación para evitar la desnaturalización del material.

Junto al sistema reactor se situará una torre de disipación térmica adaptada a las dimensiones del reactor mediante un radiador de tubo aletado al que se ha dotado de un sistema de refrigeración forzada basado en dos ventiladores adosados. El sistema se completa mediante un circuito de agua como fluido refrigerante que circula en primer lugar por un serpentín de cobre alojado en el interior de la cámara de reacción, conduciendo el fluido a un tanque de expansión de acero esmaltado situado en la parte superior de la torre. Este tanque está abierto a la atmósfera para evitar presurizar el circuito debido a un posible exceso de temperatura que conlleve vaporización. Desde la base del tanque de expansión el agua circula por nivel hasta el radiador, de donde saldrá hacia una bomba inline situada en un nivel inferior que actuará a aspiración (impulsando contra el serpentín).

En esta misma torre está montada la válvula de llenado y el caudalímetro , que se conectarán a la fuente de agua para la fabricación y que regularán el nivel de agua en el depósito principal.

En la parte electrónica se ha fabricado un pequeño armario eléctrico adaptado a las dimensiones del sistema que contiene tanto el controlador como la aparamenta eléctrica de control y potencia. En el montaje se incluyen:

- Controlador Arduino Mega 2560 con tarjeta Ethernet Shield incorporada

- Placa de 8 relés accionados por optoacoplador.

- Fuente de alimentación de 12 V para control

- Transformador de 12 V de potencia

- Reductor 12-5 V

- Interruptor diferencial en la línea de 220 V de alimentación.

- Interruptor de seta de seguridad para parada de emergencia

- Interruptores magnetotérmicos adaptados a cada una de las líneas de 220, 12, 12, 5 V

- Regletero para conexiones.

Igualmente externo a este cuadro pero actuando sobre él toda la interfaz gráfica se ha desarrollado sobre la pantalla de un móvil Nokia 3310, empleando para su actuación un teclado matriz de membrana de 4x3.

Todo el conjunto se ha montado sobre una plataforma construida en chapa de acero S235JRG2 a la que se le han incorporado 6 ruedas de goma para dotar de movilidad al conjunto.

Img. 1. Planta piloto

2. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

2.1. DESCRIPCIÓN DEL PROCESO

El proceso[10] a controlar mediante este código será la fabricación de cerveza de alta fermentación (cerveza ALE) basada en el método tradicional. Este método parte de los siguientes ingredientes que pueden variar en función de las distintas recetas:

Img. 2. Ingredientes comerciales

- Cebada cruda, seca y molida.

- Cebada malteada seca y molida.

- Cebada tostada (a distintos grados) seca y molida).

- Lúpulo antiséptico (se introduce en primer lugar para aprovechar sus alfa-ácidos como antiséptico puede ser de distintas especies).

- Lúpulo saborizante (se introduce al final de proceso para aprovechar su amargor en el sabor final del producto, puede ser de la misma o distinta especie que el antiséptico).

- Levadura, generalmente seca y/o liofilizada, y de distintas especies selectas (la más extendida es la Safale S-04).

- Coagulante (generalmente musgo seco)

- Otros (miel, extractos vegetales, limón, bebidas espirituosas de alta graduación, etc).

Img. 3. Cebada artesana

El proceso a controlar consta de las siguientes etapas como denominador común de las distintas tradiciones cerveceras, por tanto serán las empleadas para la definición de los steps del bucle de control:

1- Preparación: Para optimizar la extracción sólido líquido de la que se obtiene el mosto que después se fermentará, las condiciones de extracción son críticas para darle al producto final un carácter concreto, es por esto, que antes de la introducción de los ingredientes, el lecho líquido de reacción (agua) debe ser acondicionado. Este acondicionamiento es tan simple como la adición de la cantidad correcta (en función de la receta) de agua y su llegada a una temperatura óptima para extraer los componentes de la mezcla de cereales deseados (generalmente y a estos niveles de "homebrewing, la temperatura se obtiene empíricamente de experiencias anteriores). Así pues en esta etapa se deberá llenar el reactor con agua y calentarla hasta la temperatura adecuada.

Img. 4. Mosto de cerveza artesana

2- Extracción sólido-líquido o braceado: En esta etapa se extraen los componentes que deberán ser fermentados por la levadura. Se obtienen de dos partes principales: polisacáridos de cadena más o menos larga, principalmente almidón, de los cereales como cebada (o similares) crudos o tostados y enzimas que se encargan de la obtención de monosacáridos o disacáridos lisando estas cadenas. El aporte de estas enzimas se realiza por parte de la malta (Cebada malteada), es decir, cebada con una germinación parcial paralizada mediante un secado artificial. Este componente se hace necesario en el proceso por la incapacidad de la levadura de metabolizar hidratos de carbono de elevado peso molecular.

Es fundamental en este proceso para evitar la aparición de componentes indeseados que modifiquen el sabor de nuestro producto, controlar con cierta precisión la temperatura de extracción que deberá de mantenerse constante según el proceso tradicional (a pesar de que en procesos más modernos se emplean rampas de temperatura para extraer distintos compuestos)..

3- Cocción del mosto: Para evitar la presencia de organismos indeseados que puedan estropear las propiedades organolépticas del producto final o causar problemas a la levadura seleccionada, se mantiene todo el mosto a elevada de temperatura (cercan a ebullición) durante un periodo de tiempo determinado por el maestro cervecero. Es durante esta fase cuando se introducen los distintos lúpulos y el coagulante para conseguir un caldo de cultivo óptimo para nuestra levadura.

4- Enfriamiento: Al finalizar la cocción y previo a la siembra de levadura se debe llevar a una temperatura adecuada para el crecimiento de la levadura el sistema. Esto suele estar en torno a los 30ªC. Es fundamental que el enfriamiento se produzca los más rápido posible dado que durante la curva de temperatura se puede producir la contaminación del producto con microorganismos ambiente.

Img. 5. Residuo tras extracción

5- Siembra: Una vez a la temperatura adecuada se introducirá levadura en el sistema para su rehidratación con el mismo mosto. La curva de enfriamiento debe continuar hasta temperatura de fermentación que suele estar entre 5 y 10ºC por debajo de la de siembra. El tiempo necesario para llegar a la temperatura de fermentación será aprovechado por la levadura para consumir azúcares presentes en el medio y reproducirse hasta formar una capa que cubra casi completamente el lecho del reactor.

6- Fermentación: Es la segunda fase de actuación de la levadura, donde consume los azúcares presentes en el medio metabolizándolos hasta obtener como producto de desecho CO2 Y EtOH, componentes fundamentales de la cerveza. En esta primera etapa de fermentación en reactor, el CO2 será liberado a la atmósfera, pero el alcohol generado quedará en el seno del fluido en concentraciones que dependerán de las condiciones de fermentación y la levadura seleccionada. Es fundamental en este proceso mantener en condiciones controladas la temperatura para evitar la muerte prematura de la levadura, ya que las condiciones de actuación están limitadas a unos diez grados en torno a un valor central. Esta etapa es con diferencia la más larga del proceso y puede ir desde 5 días hasta 15 para cervezas puras de cebada.

Img. 6. Producto terminado

7- Embotellado: Una vez finalizada la fermentación la cerveza se extrae del reactor de acero y se embotella, preferiblemente en vidrio, dado que según el plástico utilizado en la botella puede interaccionar con la levadura viva que quede en el producto dando lugar a sabores indeseados. Es fundamental en la cerveza artesanal que parte de la levadura esté viva cuando se embotelle para que sea la artífice de la segunda fermentación que dará el grado de espuma de la cerveza al aprovechar el oxígeno presente en el cuello de la botella para producir CO2.

3. DESCRIPCIÓN DE LA SOLUCIÓN ACEPTADA:

3.1. POSIBLES SOLUCIONES:

De cara al control del proceso expuesto se plantean distintos sistemas disponibles comercialmente en la actualidad: PC, PLC, microcontroladores.

Hay que destacar que los requerimientos del sistema de control seleccionado era por un lado la facilidad de operación para el usuario estándar, al tratarse de un reactor de uso doméstico, luego un sistema dedicado parecía una solución más viable y menos sensible a averías así como malware. En segundo lugar, el tamaño era un elemento a tener en cuenta para conseguir un sistema compacto que se pudiera considerar relativamente portátil, o cómo mínimo que permitiera guardar el producto en una casa normal sin necesidad de tener un espacio específicamente dedicado a él. Por último el precio era un punto crítico al tener que competir en el mercado con soluciones alternativas a este reactor ya existentes.

Teniendo en cuenta todos estos parámetros la solución se hace evidente:

- El PC no cumple ninguno de los requisitos, es decir, tendría que emplear un programa funcionando sobre un sistema operativo existente, lo que lo haría al estar conectado a internet sensible a infecciones por malware. El tamaño de un PC estándar (exceptuando micropcs) lo hace inviable para una aplicación móvil, contando además con que habría que añadir una tarjeta con los pines de entrada y salida. Finalmente el precio lo sitúan como una de las soluciones menos económicas de las presentadas.

-El PLC, pese a proveer ventajas evidentes como la robustez y la estabilidad en la operación, muy por encima del PC o el microcontrolador, presenta la desventaja del precio. Se perfila como una solución demasiado industrial para este tipo de aplicaciones, contando además con que modelos más económicos tipo Zelio, no tienen suficientes entradas y salidas de cara a la operación de toda la planta.

- Finalmente el microcontrolador presenta ventaja en todos los puntos requeridos con el único inconveniente de una baja robustez, que para un entorno doméstico, sigue siendo suficiente. El precio se sitúa un orden de magnitud por debajo de las dos soluciones anteriormente citadas, además de poseer potencia más que suficiente para el control de un sistema de estas características, bastante simple, y que no requiere interacción con otros elementos en una planta industrial.

De entre los microcontroladores del mercado, quizá el más popular a día de hoy sea Arduino, con un gran número de versiones adaptadas a todas las necesidades disponibles. La versión de hardware libre de esta tarjeta controladora hace que sea posible encontrar placas a muy buen precio en el mercado asiático, unido a tener una amplísima comunidad de usuarios que aportan continuamente sus conocimientos ampliando así la cultura Maker.

Es por esto que se ha decidido emplear el Arduino Mega 2560 R3.

3.1. SOLUCIONES DE HARDWARE:

3.1.1. CONTROLADORES ARDUINO[2]:

Todo el desarrollo del sistema de control se ha basado en el empleo de controladores de bajo costo y alta flexibilidad conocidos comercialmente como Arduino, concretamente el modelo Arduino Mega.

Esta familia de hardware se basa en los chips ATMega2560 de la empresa ATMEL, un tipo de microcontroladores PIC (peripheral interface controller), es decir controladores tipo RISC (Reduced Instruction Set Controller[4]), que se plantean como perfectos para este tipo de aplicación donde los sistemas de control no son en absoluto complejos y la programación de los mismos queda simplificada por un lenguaje específico.

La placa arduino MEGA, concretamente dispone de 54 pines que pueden actuar como entrada o salida e incluso cambiar su rol a lo largo del desarrollo del bucle de funcionamiento. De estas, 16 además pueden actuar como entradas analógicas o simular una salida analógica mediante el PWM (pulse wide modulation).

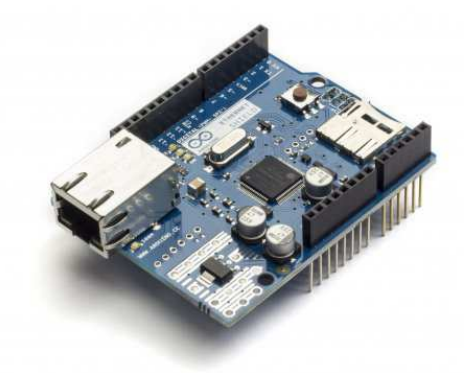
A su vez, la placa dispone del citado controlador ATMega2560, y 256 kb de memoria flash, junto a 4 kb de memoria eprom y 8 kb de ram. La frecuencia de reloj es de 16 Mhz.

El rango de alimentación de la placa es, a través del pin de alimentación, entre 7 y 12 V, según lo recomendado por el fabricante. La tensión operativa es de 5V la placa lo reduce desde la alimentación autónomamente). Cada uno de los pines actuará a 5V también, en un rango que llega hasta los 40 mA, tanto de entrada como de salida por cada pin.

Si trabajáramos con el circuito de voltaje reducido integrado en la placa a 3.3 V, el controlador podría manejar hasta 50 mA por pin.

Img. 7. Arduino MEGA 2560 R3

(store.arduino.cc/product/A000067



Img. 7. Ethernet Shield R3

(store.arduino.cc/product/A000072

3.2. SOLUCIONES DE SOFTWARE:

3.2.1. LENGUAJE DE PROGRAMACIÓN[1]:

El lenguaje empleado para el desarrollo del programa con el que manejar el controlador que controla el proceso es el específico creado por los desarrolladores del proyecto arduino para su uso. Se trata de un lenguaje de ordenes reducidas como todos los que caracterizan a los sistemas basados en controladores RISC, esta vez muy similar al lenguaje C de programación en sistemas x86.

Lo más destacable de este lenguaje es que tiene su propio editor distribuido desde la página matriz del proyecto arduino.cc, conocido como IDE Arduino, que se encarga de subir el código a la placa al conectarla mediante USB a un ordenador Windows, Mac o Android. Parte de esta capacidad multiplataforma se debe a la programación en JAVA de dicho IDE.

Como apoyo al desarrollo de proyectos basados en Arduino, se ha conformado toda una comunidad que se apoya en el fenómeno Maker, de la que surgen espontáneamente bibliotecas para simplificar la realización de tareas habituales. En el desarrollo del presente proyecto se ha hecho un uso intensivo de este tipo de aportaciones de la comunidad, como se podrá comprobar más adelante en la presentación del código.

3.2.2. ETAPAS DEL PROCESO

Atendiendo a los pasos descritos en el apartado 2.1, las etapas que deberá seguir el programa serán las siguientes:

1º- Arranque y configuración:

En el arranque deberá hacer un chequeo de la memoría para encontrar posibles recetas alojadas en la memoria de la tarjeta SD. Una vez encontradas y listadas deberá presentárselas al usuario por pantalla para que este sea capaz de encontrar la que busque en cada momento.

2º- Llenado del sistema:

Contando con que el usuario ha cargado el manejo de sólidos con los ingredientes correctos, el reactor se llenará automáticamente con el agua que indique la receta.

3º- Calentamiento previo:

El sistema llevará el fluido a la temperatura que le indique la receta y la mantendrá durante la extracción.

4º- Extracción:

El reactor gestionará la entrada de la cesta de cebada en el fluido así como la temperatura óptima para obtener los compuestos deseados.

5º- Cocción:

El reactor llevará el sistema a temperatura de cocción de mosto, y gestionará la aditivación de lúpulos y otros componentes durante el tiempo estipulado en la receta.

6º- Enfriamiento:

Se conectará el sistema de disipación de calor hasta llegar a temperatura de siembra.

7º- Siembra:

El reactor aditivará la levadura a la temperatura pertinente.

8º- Fermentación

Se deberá mantener durante el tiempo estipulado por la receta la temperatura adecuada.

ADICIONALES:

Además de estos "steps" el controlador necesitará controlar en cada vuelta que dé:

- Temperatura: debe estar comprendida en unos valores próximos a la temperatura de set point indicada en cada fase. Para ello se tomará la lectura de los sensores de temperatura y se comparará con la indicada en la receta. Se actuará conectando la resistencia y/o el disipador aerotermo, dependiendo del sentido de la corrección de ser ésta necesaria.

- Servidor: En cada iteración se deberán tomar todas las lecturas de los sensores así como del tiempo y subir los datos a la página web aojada en el mismo controlador que se actualizará.

3.2.3. FILOSOFÍA DE PROGRAMACIÓN

En el desarrollo del código se ha buscado en todo momento simular el comportamiento de un PLC, estructurando el mismo de forma de forma secuencial.

El código se compone de cinco partes:

Por un lado se realiza un set-up de todo el sistema en cada arranque (definición de variables y pines, inclusión de librerías, etc.) que se realiza fuera de las dos funciones principales.

En otro apartado existe la función void main (), en la que se realiza el set up por parte del usuario. Es decir en este apartado se incluirá todo aquello que supone una interacción con el usuario final, principalmente la selección de recetas.

En el apartado del bucle void loop() aparecen dos apartados más:

Por un lado se aprecian acciones que se deben repetir cada iteración como la lectura del reloj o de las sondas, así como el estado de los elementos (bombas, resistencias, etc), en esta categoría la actualización de la página web de estado del sistema para la consulta remota y el pulsador para parada de emergencia.

Por otro hay acciones muy concretas que se corresponden con una de las etapas (o con parte de la etapa) descritas en el capítulo 2. Estas acciones tienen unas condiciones de entrada determinadas, además de tener un indicador en forma de variable que solo permite que se ejecute una a la vez. Se ha pretendido con esto simular el funcionamiento de un PLC según la estructura de un grafcet.

Finalmente aparecen las funciones no incluidas en ninguna librería y que son necesarias para la ejecución del código. Este tipo de funciones se definen al final del código y fuera del void loop.

Para concretar en un resultado real este tipo de filosofía, los pasos a seguir serán los siguientes

- Definición de las necesidades de control del programa (Apartado 2.2.2)

- Diseño de un grafcet que satisfaga dichas necesidades (Apartado 2.2.4)

- Traducción a código de dicho grafcet.

3.2.4 GRAFCET:

Una vez definidos los pasos del proceso que el sistema debe controlar se hace necesario plasmar en un grafcet el funcionamiento del sistema de control. Al tener funciones recurrentes como el control y ajuste de temperatura o la actualización del servidor web desde el que se permite visualizar los parámetros del proceso se plantea dividir el código del bucle en tres elementos básicos:

1- Comprobación de sensores y regulación de temperatura

2- Proceso dividido por etapas con una variable que habilite o deshabilite la entrada en cada una.

3- Actualización del servidor web.

El proceso sería algo como el siguiente esquema:

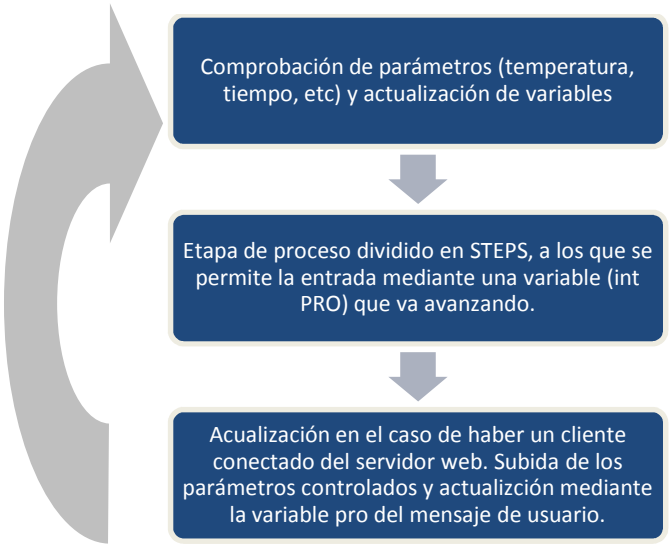


Fig. 2. Método simplificado de control

Si bien esta aproximación es válida habría que considerar más acciones que debería realizar el controlador, como una configuración inicial al arranque y la interacción con el usuario. Estas acciones deberá realizarlas antes de la entrada en el bucle y solo en el arranque, es decir, no se repetirán hasta que el equipo se reinicie.

Finalmente también habrá que tener en cuenta que será necesario un pulsador de parada de emergencia para garantizar la seguridad de la operación, lo que deberá ir en una línea de proceso independiente. El esquema final de funcionamiento (muy simplificado) sería el siguiente:

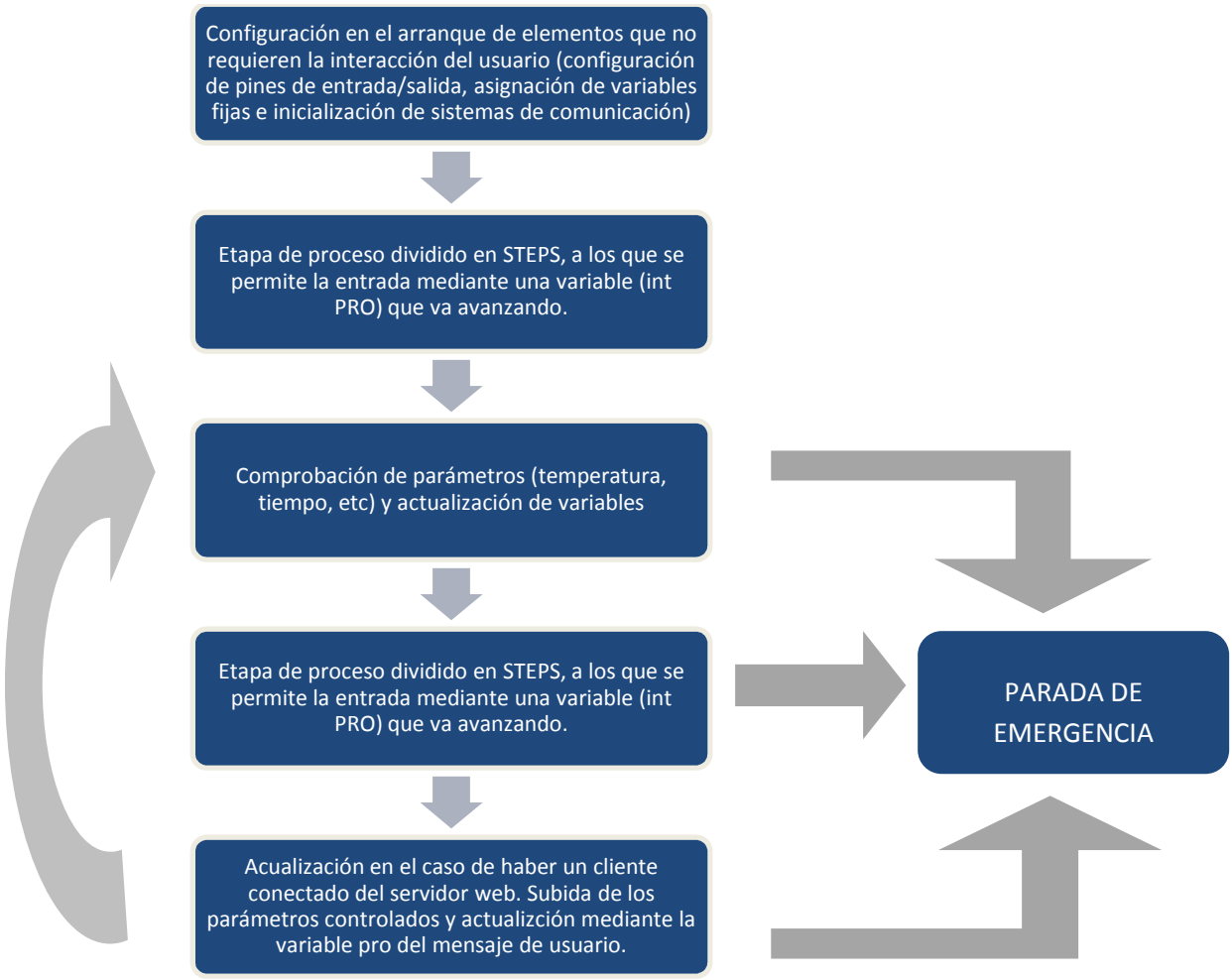


Fig. 3. Método completo de control

De cara a conseguir un grafcet empleable habría que definir concretamente cada uno de los steps por los que deberá de pasar la variable PRO en el bucle, lo que constituirá la parte horizontal del grafcet. Siguiendo los pasos del punto 2.2.2:

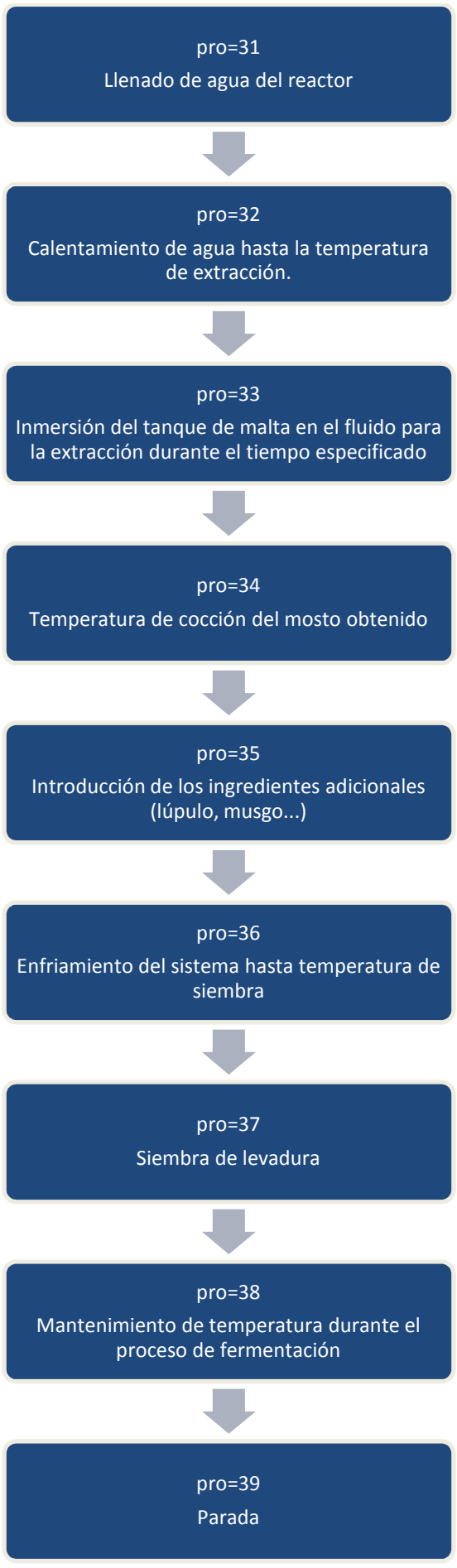


Fig. 4. Steps en los que se divide el programa

Uniendo estos esquemas se llega al grafcet que se puede consultar en el Anexo 1 (Plano 1).

3.2.5 DISEÑO DE DETALLE Y TRADUCCIÓN A CÓDIGO:

El grafcet desarrollado en el punto 2.2.4 y plasmado en el plano 1 da una idea general de como debe funcionar el sistema de control, pero para la correcta operación del equipo es necesario un nivel de concreción mucho mayor. Es por esto que antes de escribir el código hay que estudiar qué y cómo van a hacer cada uno de los steps que se programen, cabe destacar que la solución final con el código comentado está en el anexo 3 Manual del programador.:

A.- DEFINICIÓN DE VARIABLES

Se procede en primer lugar a definir los parámetros que permanecerán fijos a lo largo, estos son imprescindibles para el correcto funcionamiento de la librería que controlará los motores paso a paso para el manejo de sólidos.

#define STEPS_PER_REVOLUTION 512

#define MOTOR_SPEED 1200

#define MSECS 2000

B.- INCLUSIÓN DE LIBRERÍAS

Justo después de la definición de variables y aún fuera del código funcional (fuera de las funciones principales) del programa se incluyen las librerías que se van a emplear para simplificar el código, a continuación se presentan:

Las librerías como ya se ha comentado anteriormente, son archivos de texto diseñados por usuarios de la comunidad para el beneficio de la misma donde se automatizan procesos habituales en los controladores en forma de funciones que facilitan la programación al resto de usuarios de la comunidad y simplifican en gran medida los códigos que estos generan.

En el caso concreto de este TFG, se han empleado librerías para el control de elementos de hardware comerciales empleados en el desarrollo de la planta piloto, para de esta forma, poder llegar a una solución efectiva en forma de programa sin necesidad de conocer los detalles técnicos de los componentes electrónicos.

Las librerías empleadas han sido las siguientes:

Para la lectura de las mediciones de temperatura mediante el sensor ds18b20 de Maxim Integrated es recomendable el uso de dos librerías:

DallasTemperature.h[5] --> Descargada de la página panamahitek.com, contiene las ordenes de lectura y conversión de la señal proporcionada por el integrado ds18b20 para adaptarlas a la interfaz del controlador.

OneWire.h[6] --> Es la librería que permite al controlador trabajar con elementos OneWire, es decir, que le permite distinguir por medio de un código unitario para cada sensor de 64-bits, las distintas señales conectadas a un único pin. Este protocolo es importante de cara al escalado de los reactores a nivel industrial, ya que un único pin de la placa controladora es capaz de manejar todos los sensores de temperatura de la instalación.

Como interfaz de entrada de usuario se ha empleado un teclado de matriz/membrana 4x3. Este teclado se conecta mediante 7 pines al controlador, de forma que cuando se pulsa una de las teclas se conectan dos de los pines, dando lugar a una combinación singular. Para la asignación de valores a cada una de las combinaciones así como para la lectura y traducción de estas entradas se emplea la librería Keypad, publicada bajo GNU por el grupo Wiring[7] de la Universidad de los Andes.

Como interfaz de salida de usuario se ha empleado la pantalla de un teléfono móvil Nokia 3310, comprada montada sobre pcb y con las adaptaciones necesarias para su control con Arduino la tienda online Banggood. Su funcionamiento se compone de una matriz de puntos y una comunicación por puerto de serie, para el manejo de cada uno de estos elementos se emplean las siguientes librerías.

NokiaLCD.h[8] --> Publicada por Scott Daniels, está preparada para traducir directamente código ASCII a la matriz de puntos, se apoya en la librería distribuida con el IDE SPI. Recibe órdenes similares a las necesarias para la comunicación por pantallas de puerto serie de Arduino (print, etc) simplificando enormemente el mostrar información al usuario final.

SPI.h --> Viene incluida directamente en el IDE de Arduino.cc, por tanto se considera una de las librerías oficiales, sirve para facilitar el envío y recepción de datos mediante los pines de conexión serial RX - TX, sirviendo como traductor de los pulsos enviados y recibidos. Se incluye en este TFG tanto como apoyo a la librería NokiaLCD.h como para el debugging del código mediante la conexión a una pantalla serial simulada desde el IDE.

El sistema de control web requiere de dos funciones por un lado, el controlador debe actuar como un servidor que reciba las conexiones de los dispositivos empleados por el cliente y les muestre una página web alojada en la memoria EPROM. Por otro lado se necesitará gestionar la conexión en el sentido más estricto de la palabra (asignación estática de IP, MAC, etc). Ambas necesidades quedan simplificadas por la librería Ethernet.h[9] integrada en el IDE. Para su funcionamiento será necesario el empleo de una shield prefabricada según los esquemas de la comunidad que simplificará además el prototipado de la planta piloto.

Finalmente, integrado en este mismo shield se encuentra un lector de tarjetas SD que se empleará como interfaz para la introducción de nuevos programas (nuevas recetas) siguiendo un formato normalizado entendible por el bucle de control que modificará los parámetros clave de la fabricación de producto final descritos en el siguiente apartado.

C.- DEFINICIÓN DE OTROS ELEMENTOS

Se definirán ahora los elementos correspondientes a los pines y otros ajustes que va a utilizar cada uno de los elementos controlados por las librerías externas, esto es:

- Definición de la MAC, IP y puerto del servidor web integrado.

- Definición de la correspondencia en el teclado de cada tecla con su señal en pines.

- Definición de los pines necesarios para operar la pantalla.

- Definición de todas las variables necesarias para el funcionamiento del lector de tarjetas microSD integrado en el módulo ethernet, así como para la lectura y almacenamiento de valores de los archivos de texto en que consistirán las recetas.

- Definición de todas las variables de proceso.

- Definición de todos los pines según cada elemento, asignándole un nombre de variable que haga fácil su posterior identificación en el código.

En conclusión, cada vez que se inicialice el sistema, el controlador Arduino realizará las siguientes acciones antes de comenzar a ejecutar las funciones que se le carguen:

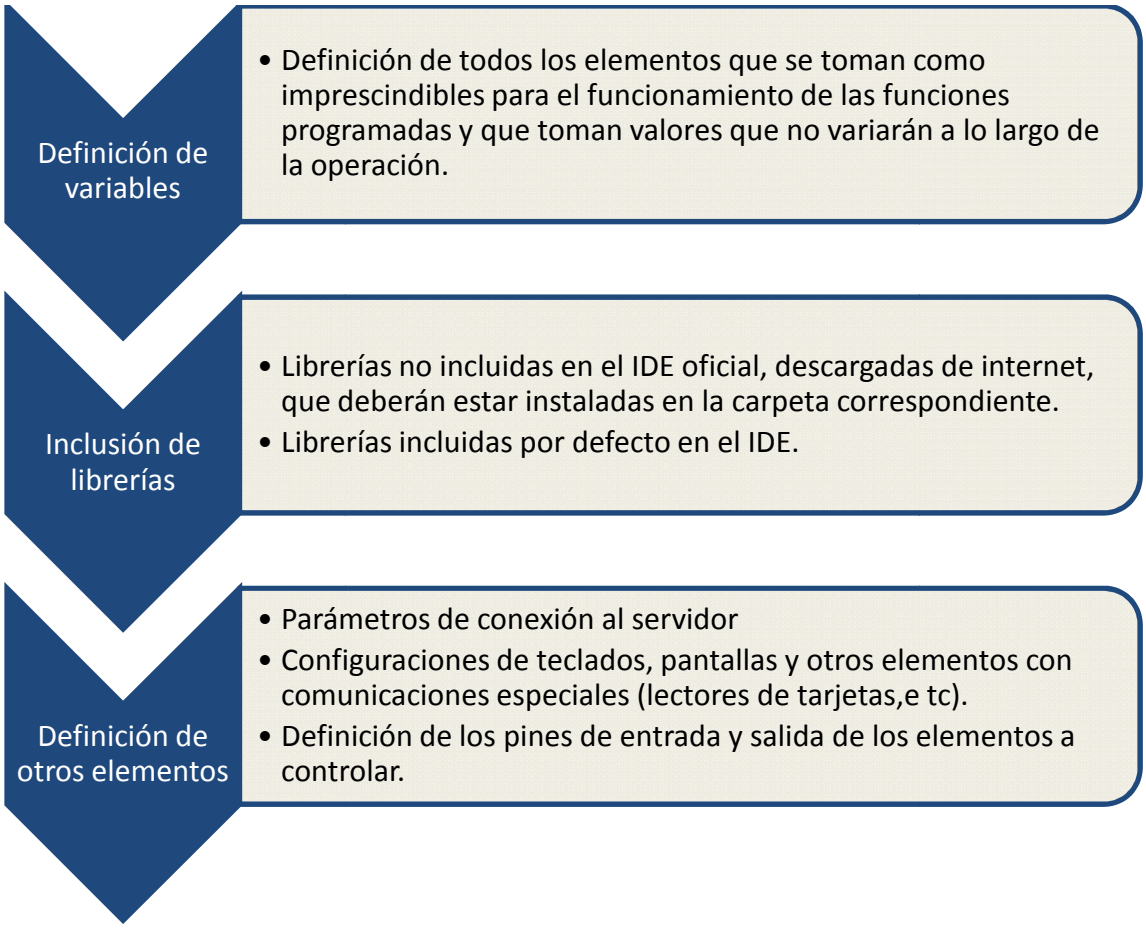


Fig. 5. Acciones en el arranque

D.- VOID SET-UP

Es la primera función definida en cada proyecto de arduino y sirve para configurar, como su propio nombre indica todos los elementos que serán necesarios para el correcto funcionamiento del bucle.

En la función de SET-UP del programa de control del reactor realizaremos las siguientes aciones:

- Inicialización de la comunicación Ethernet según los parámetros establecidos en el apartado anterior.

- Asignación del estado inicial de cada uno de los pines (entrada/salida), aprovechando para ello los nombres definidos en el apartado de definición de otros elementos (punto c de este capítulo)

- Inicialización de la comunicación serial (puertos RX, TX) imprescindibles para el correcto funcionamiento de la pantalla.

- Limpieza de la pantalla y primer mensaje al usuario.

- Inicialización de la tarjeta micro SD

- Lectura de los archivos en la tarjeta de memoria.

- Presentación por pantalla del listado de archivos.

- Inicialización de la variable puntero que irá marcando en que etapa del bucle debe entrar, dándole el valor de la primera entrada.

Es decir, la finalidad de la función void set-up en el programa de este TFG es doble, por un lado inicializa todas las comunicaciones "levantando" tanto el servidor como la pantalla, tareas que como ya se ha indicado antes solo deben realizarse una vez, por el otro, sirve para el volcado a memoria de toda la información relativa a las recetas que el usuario podrá seleccionar luego para servir como set-points de los distintos parámetros a controlar en la elaboración de la cerveza. Esquemáticamente el funcionamiento de esta función es el siguiente.

Fig 6. void setup ()

E.- VOID LOOP (1)

La función VOID LOOP es la que se repetirá una y otra vez durante el funcionamiento del dispositivo hasta que se desconecte. Es por tanto donde se encuentra el grueso del código y donde se realizan la gran mayoría de las acciones necesarias para el funcionamiento del sistema, es por eso que para su explicación se va a dividir en cuatro partes, siendo esta la primera de ellas.

La parte más importante de void loop() es la variable pro (de programa), que puede tomar valores desde el 30 hasta el 39. Esta variable es la que determinará que partes del código se expresan en la operación del dispositivo y cuales quedarán anuladas, segmentando el código mediante condiciones if().

En este primer apartado se va a tratar la parte que continúa directamente con el void set up, donde se lee la selección del usuario y se definen los parámetros que actuarán como set-point.

El bucle sólo entrará a esta parte de código si el valor de PRO es 30. Este valor solo lo adquiere en el void setup() luego es imposible que vuelva a entrar a este if una vez que ha cambiado el valor de pro.

Las operaciones que se llevan a cabo son por este orden:

1 - Toma del valor de entrada introducido por el usuario al usar la función getKey() de la librería Keypad.h. Este valor se introduce en la variable key.

2 - Conversión de este valor en su código ASCII correspondiente (restando 49 unidades). Este valor se recoge en la variable kai.

3- Una vez que se ha tomado un valor a la variable key (que recoge el valor) se procede a hacer una impresión por pantalla que informe al usuario que el proceso de fabricación está comenzando y se pasa a pro=31.

Una vez que se ha asignado un entero a kai, se tomará como primer coordenada de la matriz que recoge los valores numéricos de los parámetros de cada una de las recetas, es decir una vez fijado ese valor, int recipar [][], pasa de ser un array a ser un vector int recipar [kai][].

Fig. 7. Sistema selección recetas

F.- VOID LOOP (2)

En el segundo apartado de void loop se va a describir la comprobación de la temperatura, una subrutina que se repetirá en cada uno de los ciclos, exceptuando valores de pro de 30 (etapa descrita en E.), 31 (llenado), 38(fin de la fermentación) y 39(parada).

Para realizar el control de temperatura, pese a haber planteado el uso de un PID, se ha llegado a la conclusión de que un controlador todo o nada tanto hacia frío como hacia caliente (tipo termostato) con una tolerancia de +-1 grado centígrado respecto a los distintos set-points de

las distintas etapas es más que suficiente, considerando que en el proceso artesano tradicional, los controles tienen una tolerancia de hasta +-5 ºC.

El funcionamiento de este sistema es el siguiente:

1- En cada uno de los ciclos del bucle el controlador toma el valor de temperatura proporcionado por la sonda y lo aloja en una variable.

2- En caso de no estar en ninguna de las etapas vetadas para el control, compara esta medida con el setpoint establecido en ese momento en una variable que se actualiza según el tiempo.

3- En caso de ser menor a la temperatura mínima, se conectará la resistencia y se dará señal de apagado del sistema de disipación térmica.

4- En caso de ser mayor, se conectará el sistema de disipación térmica y se desconectará la resistencia.

5- En caso de estar dentro de la tolerancia, se desconectarán ambos sistemas.

Esquemáticamente sería de la siguiente manera:

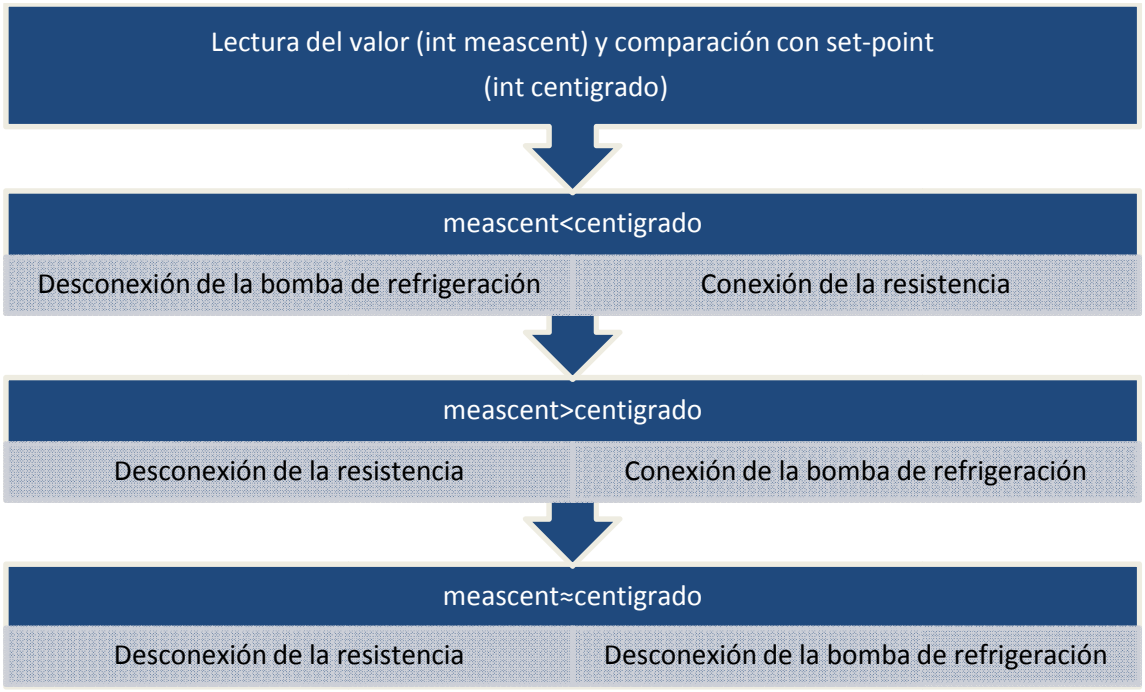


Fig. 8. Sistema de control temperatura

G.- VOID LOOP (3)

En este apartado se abordará el fragmento del bucle en el que se comprenden las acciones que controlan realmente el proceso según va avanzando la fabricación de cerveza. Para seguir una progresión lógica se nombrarán según el valor de la variable int PRO en cada momento de la ejecución:

pro=31

Es el ciclo de llenado con agua del reactor. El sistema toma el valor en litros de agua del vector de parámetros y lo mete a una variable. En otra variable cuenta los pulsos del caudalímetro efecto hall, considerando que según el fabricante, cada 450 pulsos es un litro de agua. Cuando se comprueba que se han dado los pulsos correspondientes, se cierra la valvula y se pasa al siguiente step.

pro=32

En esta etapa solamente se cambia el set-point de temperatura y se mantiene el valor de pro

hasta que se alcanza el valor deseado.

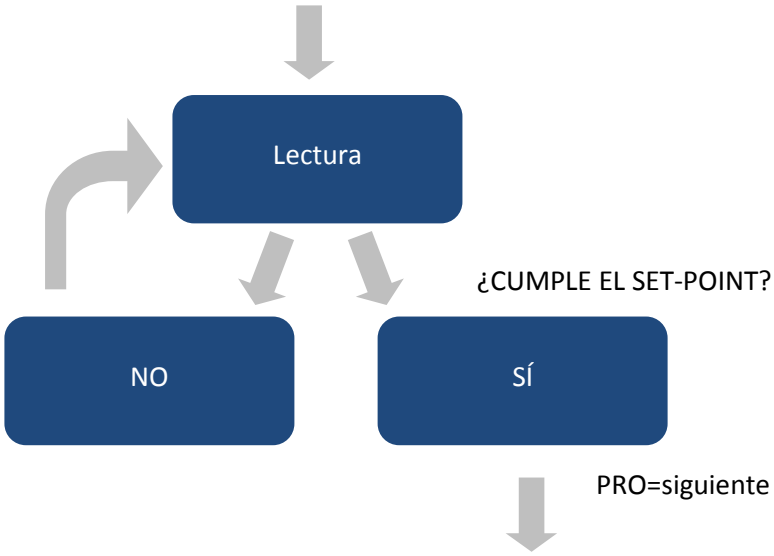


Fig. 9. Esquema general de subprocesos

pro=33

En esta etapa con la temperatura controlada se introduce la cebada malteada en el fluido para la extracción de los azúcares que posteriormente se fermentarán en el proceso. Para ello se ha empleado un motor de 12 volts de corriente contínua al que se han conectado 4 relés con sus correspondientes salidas dos a dos, de forma que se puede realizar una inversión del giro del mismo por software modificando la salida de unos pines a otros y por tanto invirtiendo la polaridad de la alimentación.

La etapa en sí consiste en la introducción del contenedor de cebada y su mantenimiento durante un tiempo predeterminado.

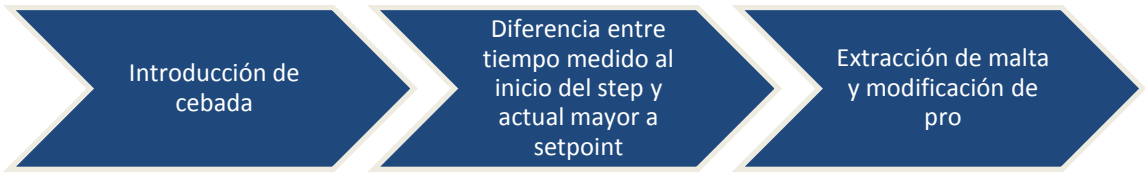


Fig. 10. Esquema pro=33

pro=34

Es una repetición del procedimiento en 32, pero con un set-point de temperatura distinto.

pro=35

Este step es muy similar al paso 33, con la diferencia de que añadirá dos ingredientes en lugar de uno solo, esto es, manteniendo en primer lugar durante un tiempo prudencial el mosto a temperatura de cocción, añadirá el primer lúpulo, empleado como antiséptico. Dejando pasar el tiempo definido en el vector de parámetros introducirá el segundo, empleado como saborizante y finalmente pasado el último tramo de tiempo extraerá los dos a la vez.

Fig. 11. Esquema pro=35

pro=36

Step igual al 32 y 34, con set-point distinto y con la excepción de aditivar la levadura cuando la temperatura cumple el set-point.

pro=37

Control de temperatura a lo largo del tiempo.

pro=38

Parada del sistema

pro=39

Bucle vacío para dejar el sistema en espera.

H.- VOID LOOP (4)

Como entrada final de la función loop(), queda hablar sobre el servidor web integrado en el controlador que permite el control del proceso desde cualquier punto por internet.

Los valores mostrados por pantalla son tiempo desde la puesta en marcha, temperatura del lecho del fluido y un mensaje describiendo la etapa llevada a cabo por el equipo.

El funcionamiento es el siguiente:

Una vez finalizada la etapa de control correspondiente, el sistema comprobará si hay algún cliente conectado. De ser así generará una página html introduciendo los distintos parámetros que se han descrito. El mensaje generado se obtendrá por medio de un selector tipo select-case, en el que la variable de selección será la variable pro antes empleada.

Así pues el esquema general de funcionamiento del bucle sería:

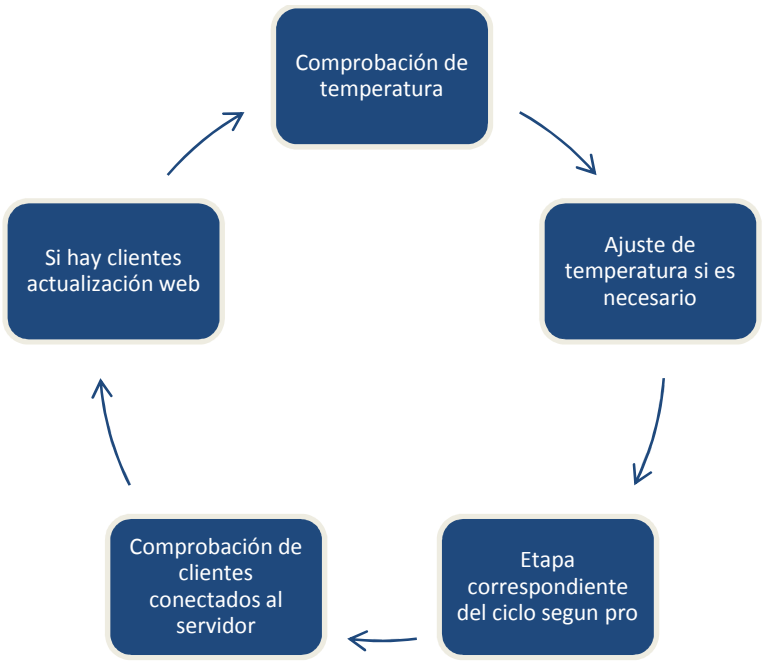


Fig. 12. Esquema general de control

I.- FUNCIONES AUXILIARES

A falta de librerías específicas para el control de los motores paso a paso (de fabricación china y sin ninguna documentación por parte del fabricante) y la necesidad de automatizar el proceso de lectura de archivos para poder tener un bucle for manejable en la función void setup(), ha habido que añadir unas cuantas funciones tras void loop(), que pueden ser invocadas desde esta según el funcionamiento de la tecnología arduino. Pueden consultarse junto con el resto del código en el anexo 3 - Manual del programador.

4. CONCLUSIONES DE LA MEMORIA:

La principal conclusión que se obtiene de esta memoria es que la potencia de las plataformas libres, tanto en hardware como en software, hacen posible el desarrollo de productos a un coste realmente económico, como se puede comprobar en el apartado 5.2 (presupuesto) por gente con unos conocimientos en electrónica y programación, que pese a ser suficientes no alcanzan la categoría de expertos.

La comunidad ha sido clave para el desarrollo de este trabajo, como se puede comprobar de la gran cantidad de librerías empleadas, con lo que se ha conseguido reducir la complejidad del código hasta niveles que lo hacen comprensible incluso para auténticos programadores noveles con una ligera explicación de lo que se hace en cada parte.

Por otro lado es remarcable como conclusión de este trabajo, una hipótesis planteada durante el planteamiento del mismo, la potencia de una placa controladora de 11 € para controlar un reactor completo automatizado. Este punto, según mi opinión ha quedado claramente demostrado, pues el control de este proceso es completamente escalable (con la consiguiente multiplicidad de sensores) a nivel industrial, consiguiendo así, un sistema de control "low-cost" para una aplicación industrial completamente funcional en explotación real.

Finalmente y como conclusión personal, me gustaría remarcar la interdisciplinariedad del currículum del grado en ingeniería química, que permite a los graduados enfocar su carrera profesional a sectores tan dispares como la electrónica automática o el control de procesos abriendo a los egresados un abanico de posibilidades que se hace efectivamente inabarcable.

5. BIBLIOGRAFÍA:

1- IDE Arduino. (s.f.) Recuperado el 21/04/2014, de https://www.arduino.cc/en/Main/Software

2- Arduino/Genuino MEGA (s.f.) Recuperado el 21/04/2014 de https://www.arduino.cc/en/Main/ArduinoBoardMega2560

3- Modulación por ancho de pulsos Recuperado el 2/06/2014 de https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos

4- Reduced instruction set computing (s.f.) Recuperado el 6/8/2014 de https://es.wikipedia.org/wiki/Reduced_instruction_set_computing

5- García González, Antony, Aprendiendo a utilizar el sensor de temperatura DS18B20, Panama Hitek, 09/01/2014. Consultado el 08/08/2014. Se puede consultar en http://panamahitek.com/aprendiendo-utilizar-el-sensor-de-temperatura-ds18b20/

6- Dallas Semin¡conductor's 1-Wire Protocol, (s.f.) Recuperado el 08/08/2014 de http://playground.arduino.cc/Learning/OneWire

7- Stanley, Mark and Brevig, Alexander, Wiring distribution, wiring.org.co, 12/07/2012. Consultado el 10/08/2015, se puede consultar en http://wiring.uniandes.edu.co/source/trunk/wiring/

8- Daniels, Scott, NokiaLCD Library, provideyourown.com, 12/11/2012. Consultado el 15/8/2014. Se puede consultar en https://github.com/provideyourown/NokiaLCD/blob/master/nokiaLCD.h

9- Arduino Ethernet Shield (s.f.) Recuperado el 15/08/2014 de https://www.arduino.cc/en/Main/ArduinoEthernetShield

10- Elaboración en casa (s.f.) Recuperado el 3/2/2014 de http://www.cervezas.info/proceso-cervecero/elaboracion-en-casa/

11- Arduino, Función lectura en línea en SD (13/6/2014) Recuperado el 4/9/2014 de http://domotica-arduino.es/arduino-funcion-lectura-linea-en-sd/

12- Antonio Castro, CAO-11: El uso de relés en Arduino. Blog de Acuariofília, Biología y Medioambiente 14/06/2013. Consultado el 02/05/2014. Se puede consultar en http://ciberdroide.com/AcuBioMed/cao-11-el-uso-de-reles-en-arduino/

6. PRESUPUESTO (ANEXO 1):

6.1. ESTRUCTURA DEL PRESUPUESTO:

Para la valoración del presupuesto correspondiente al trabajo se ha separado en cuatro partidas principales, esto es así porque se ha pretendido dar un valor al objeto del trabajo, es decir el desarrollo de un programa para el control del proceso de fabricación de cerveza artesana, y por otro, la mano de obra correspondiente al desarrollo de la planta piloto, que no entraría dentro del trabajo en sí, pero que se ha desarrollado para realizar las pruebas que se han considerado oportunas, así es posible distinguir por un lado el dinero que se presupuestaría si se solicitará el desarrollo de este trabajo como ingeniería del mismo y por otro el dinero real gastado en lo que correspondería al cliente que lo solicitaría (en el hipotético caso de realizar este trabajo como un profesional en el libre ejercicio de su profesión):

Mano de obra de programación: Como su propio nombre indica se corresponde a las horas totales que se han dedicado a la recopilación de librerías de la comunidad y a la composición del código.

Materiales de programación: Se corresponde al coste de los materiales mínimos necesarios para el desarrollo de proyectos arduino a cualquier escala.

Mano de obra planta piloto: la combinación de las horas de ingeniería, electricidad y mecánica para la fabricación de la planta piloto.

Materiales planta piloto: Los componentes tanto electrónicos como de cualquier clase empleados.

6.2: PRESUPUESTO:

Desarrollo

Materiales programación

	Ud.	€/Ud.	Total
Placa de prototipado rápido	1	0.6	0.60 €
Kit de cables	1	1	1.00 €
Arduino Nano	1	2	2.00 €
Kit de iniciación Arduino	1	16	16.00 €

Mano de obra programación

Horas técnico programador	50	12	600.00 €

Materiales planta piloto

Plancha acero 6mm	1	40	40.00 €
Ruedas	4	3	12.00 €
Reactor acero AISI 304 L	1	70	70.00 €
Camisa Galvanizada	1	20	20.00 €
Serpentín Cobre	1	25	25.00 €
Resistencia 1 Kw	1	7	7.00 €
Disipador aluminio	1	25	25.00 €
Tanque expansión	1	10	10.00 €
Conductos Cobre	1	20	20.00 €
Paquete de perfiles	1	15	15.00 €
Kit de cableado hasta instrumentos	3	2	6.00 €
Manguitos y grifería	1	20	20.00 €
Servomotores Stepper	3	1	3.00 €
Controladoras arduino mega	1	11	11.00 €
Placa 8 relays	1	4	4.00 €

Shield Ethernet Arduino	1	4	4.00 €
Bomba de agua	1	12	12.00 €
Motores bomba de agua y polea	2	12	24.00 €
Motor agitador	1	0.6	0.60 €
Sonda de temperatura	1	1	1.00 €
Poleas	3	1	3.00 €
Kit de regleta	1	5	5.00 €
Armario eléctrico	1	20	20.00 €
Transformador AC/DC 220/12 v	1	12	12.00 €
Rectificador AC/DC	1	6	6.00 €
Magnetotérmico 20A	1	5	5.00 €
Magnetotérmico 10A	3	6	18.00 €
Diferencial Monofásico 40A/30mA	1	14	14.00 €
Regletero 35 Ud.	35	0.2	7.00 €
Pulsador parada emergencia	1	6	6.00 €
Fuente alimentación AC/DC 220/12 10A	1	11	11.00 €
Mano de obra planta piloto			
Oficial 1º Electricista	20	15	300.00 €
Oficial 1º Mecánica	30	10	300.00 €
Técnico programador	3	12	36.00 €

1,692.20 €

6.3. COMENTARIOS SOBRE EL PRESUPUESTO

Teniendo en cuenta que el valor comercial de este tipo de dispositivos con un nivel de automatismo similar se encuentra ahora mismo en el entorno de los 2000€ (Brewie) o en el de los 1200 para reactores mucho más rudimentarios. Teniendo además en cuenta que en este trabajo se está hablando de un prototipo en el que se han empleado materiales comprados al por menor y se ha manufacturado de forma completamente artesanal, es mi apreciación personal que el objetivo de rebajar el precio de mercado de estos reactores podría conseguirse con una correcta economía de escala.

Si se considera que se pueden reducir por la fabricación en cadena los costes en aproximadamente un 30% (separación de la fabricación del reactor en puestos de trabajo), y se toma en cuenta además que la mano de obra de programación es un coste que no se va a repetir, el coste de una unidad comercial EXWORK sería igual a:

$$Cew = 0.7 \cdot (1692 - 600) = 764.4\ €$$

Un margen industrial adecuado podría ser el 40%, considerando que la distribución se haría directamente a minoristas, es decir, 1070 € para el distribuidor. Si se tomara un valor de referencia para PVP, de un incremente del 80% sobre el costo 1375.92€.

Esto dejaría para el distribuidor un margen comercial de aproximadamente el 30%, con lo que se obtendría un precio de venta al público válido considerando las prestaciones del equipo y considerando además la situación del mercado.

Enfocándolo desde el punto de vista opuesto, copiando los precios de la competencia más directa, se podría tener un PVP de 2000 € aproximadamente (se toma 1956.864 € para trabajar con márgenes exactos). Si se suponen unos márgenes equitativos para proveedor y distribuidor del 60% del costo de cada uno, el reparto se haría de la siguiente manera:

Costo: 764.4 €

Beneficio neto que genera el producto: 1956.86-764.4=1192.46 €

Precio de venta a distribuidores EXWORKS = 764.4x1.6= 1.223 €

Beneficio neto para la empresa = 458.64 €/Ud.

Precio de venta al público= 1956.86€

Beneficio bruto para el distribuidor= 733.86 € (a falta de costes de transporte, logística y marketing)

Con lo que a precio de mercado el negocio podría ser muy interesante para los distribuidores, facilitando la entrada en el mercado del producto, como mínimo a nivel nacional.

7. ÍNDICE DE PLANOS (ANEXO 2)

Adjuntos al presente TFG se podrán encontrar los siguientes planos:

8. MANUAL DEL PROGRAMADOR (ANEXO3):

8.1- DEFINICIÓN DE VARIABLES:

En este punto se procede a la definición de las variables necesarias para la adaptación de la librería de control de los stepper al modelo concreto empleado en el proyecto, en este caso 28BYJ-48 de 5 voltios marca Banggood:

#define STEPS_PER_REVOLUTION 512

#define MOTOR_SPEED 1200

#define MSECS 2000

8.2- INCLUSIÓN DE LIBRERIAS:

En este apartado se incluyen todas las librerías necesarias para el funcionamiento del programa que no vienen incluidas por defecto en el IDE de arduino.cc, como ya se ha expuesto en puntos anteriores la fortaleza de los sistemas open hardware y open software y en especial del proyecto arduino es la extensa comunidad que produce contenidos libres para el sistema, en este caso se han tomado las siguientes (descritas con mayor detalle en el apartado 2.2 de esta memoria):

#include <DallasTemperature.h> --> De Panamahitek.com, para los sensores ds18b20

#include <OneWire.h> --> De Panamahitek.com, para el uso de sensores con protocolo 1-Wire de Dallas Semiconductors, inc.

#include <Keypad.h> --> Del grupo Wiring de la Universidad de los Andes, controla el teclado matriz.

#include <NokiaLCD.h> --> De Scott Daniels (provideyourown.com) - controla la matriz de led que conforman la pantalla, en este caso de un Nokia 3310.

#include <SPI.h>--> Integrada con el IDE de Arduino.cc, sirve para manejar la comunicación serial por los puertos tx, rx, necesaria para el funcionamiento de la pantalla.

#include <SD.h> --> Librería integrada con el IDE de Arduino.cc, maneja un lector de tarjetas SD, en este caso microSD (mismos protocolos de funcionamiento) integrado en el shield prefabricado incorporado para la conexión ethernet.

#include <Ethernet.h> --> Integrada también con el IDE de Arduino.cc para el manejo de una conexión a internet mediante Ethernet, tanto como cliente como servidor.

8.3 - CONFIGURACIÓN DE A LIBRERÍA DE CONEXIÓN ETHERNET:

En este apartado se configura por un lado la MAC que va a presentar el dispositivo cuando se conecte a una red, la dirección IP que tomará (este parámetro habrá que variarlo en función del formato de la red en el punto de instalación) y el puerto que se empleará para conectarse al actuar como servidor.

```
byte mac[] = {  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

IPAddress ip(192,168,0,177);

EthernetServer server(80);
```

8.4- DEFINICIÓN DEL TECLADO:

Se ha optado por un teclado numérico de 12 dígitos (de 0 a 9, asterisco y almohadilla). Se ha tomado esta decisión por considerarlo más que suficiente para las necesidades de control del dispositivo, al tener este un diseño en los menús inspirado en los teléfonos móviles de años atrás. Para la configuración, y gracias a la librería keypad.h, sólo es necesario definir en la matriz el número de filas y de columnas y en una tabla como se observa a continuación, especificar qué carácter ANSII se corresponde a qué pulsación. Es muy importante tener en cuenta que la codificación que va a devolver la librería es tipo CHAR, de forma que no podremos emplearlo como valor numérico sin convertirlo antes.

```
const byte ROWS = 4; //cuatro filas

const byte COLS = 3; //tres columnas

char keys[ROWS][COLS] = {

 {  '1','2','3' } ,

 {  '4','5','6' } ,

 {  '7','8','9' } ,

 {  '*','0','#' }

};
```

Una vez configurado se debe especificar qué pines del controlador se corresponde cada una de las columnas y filas, diferenciables en el cableado de conexión por estar agrupado de forma distinta (tres a cuatro) en el bus. Esta definición de pines, se podría mover a la definición general, pero por comodidad en cuanto a la gestión de librerías se ha decidido dejarlo en la configuración del teclado.

```
byte rowPins[ROWS] = {  14,15,16,17};
```

byte colPins[COLS] = { 18,19,20};

Finalmente con la función siguiente se inicia la lectura de datos:

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

8.5- DEFINICIÓN DE LA PANTALLA

Mediante la siguiente función contenida en la librería específica para las pantallas Nokia 3310 y Nokia 3510 se configuran los pines del controlador a utilizar por cada uno de los pines de la pantalla necesarios para su correcto funcionamiento, identificados en el orden marcado en el comentario según vienen especificados en la placa soporte de la pantalla.

NokiaLCD NokiaLCD(41,39,37,35,33); // (SCK, MOSI, DC, RST, CS) //

8.6- DEFINICIÓN DEL LECTOR DE TARJETAS MICROSD:

Los pines, al tratarse de una librería estándar vienen definidos por defecto (se refiere a una shield comercial, al igual que el controlador de hardware y software libre). Al tratarse éste de un prototipo se ha admitido como válido y por tanto no se ha modificado.

En esta configuración sin embargo lo que se inicializa es un contador que conformará el menú de recetas en la fase de selección de programa.

Se define la raíz de la tarjeta SD y, en ella, un archivo genérico donde luego se abrirán todas las recetas introducidas junto a la tarjeta. Inicializa a 0 el contador de recetas y después almacena las variables de proceso en un vector junto a un numero de posición que, en este caso, va de 1 a 5 por tratarse de un prototipo (máximo de líneas de la pantalla con la que se está trabajando).

Cuando el usuario seleccione una receta según el número mostrado en pantalla, las variables almacenadas en la matriz se emplearán como variables para el funcionamiento del bucle.

File root;

File archivo;

int recipecount=0;

int recipar [5][9];

String lin;

8.7 - VARIABLES DE PROCESO:

Los siguientes valores son las variables que controlarán el proceso:

long unsigned chrono=0; --> esta variable se tomará como referencia del tiempo, igualándola al valor de la función millis() (tiempo transcurrido desde el arranque del controlador) y restándolo en cada ciclo del controlador para tomar el tiempo absoluto que transcurre desde el inicio de cada paso del proceso.

long unsigned chronocont=0; --> Almacena el valor del tiempo transcurrido en cada vuelta de controlador para cada etapa de proceso que deba ser controlada por tiempo.

long unsigned templupa=0; --> Temperatura de extracción del lúpulo empleado como antiséptico del proceso.

long unsigned templupb=0; --> Temperatura de extracción del lúpulo empleado como saborizante final de la cerveza.

int rescont=0;//resistencia control --> Controla si la resistencia está encendida (1) o apagada (0).

int bombcont=0;// controla si la bomba de recirculación está en marcha

Los siguientes dos pares identifican los pines que alimentan, mediante sendos relés, el motor de continua reversible encargado del movimiento del saco de cebada/malta. Se debe configurar de este modo para evitar cortocircuitar el sistema:

int mup=23;//malta arriba positivo

int mun=25;//malta arriba negativo

int mdp=24;//malta abajo positivo

int mdn=26;//malta abajo negativo

int maltcon=0; --> Este valor binario controla si la malta está abajo (sumergida) o arriba.

int pro=30; --> Este valor controla en que paso del proceso está.

int levacont=0; --> Este valor binario controla si ya se ha realizado la adición de la levadura.

int bomb=27; --> Este valor define el pin desde el que se controla el arranque de la bomba de refrigeración.

int valv=28; --> En este pin se conecta la electroválvula que ajusta el llenado.

int res=22; --> En este pin se conecta el relay que controla el encendido de la resistencia de calentamiento.

int agi=29; --> En este pin se conecta el relé que controla la agitación del sistema.

int watpul=0; --> Esta variable acumula los pulsos producidos por el caudalímetro de efecto Hall para obtener los litros de agua admitidos mediante una relación.

int caud=49; --> En este pin se conecta el caudalímetro hall.

int EMERGENCIA=51; --> En este pin se conecta el pulsador de parada de emergencia.

int motorPins[4] = { 21,30,32,34}; --> En estos pines se conecta el stepper que controla el lúpulo antiséptico (Stepper A).

int stepcona=0; --> Este valor binario controla la posición del stepper A.

int motorPinsb[4] = { 36,38,40,42}; --> En estos pines se conecta el stepper que controla el lúpulo saborizante (Stepper B)

int stepconb=0; Este valor binario controla la posición del stepper B.

int motorPinsc[4] = { 43,44,45,46}; --> En estos pines se conecta el stepper que gira la clapeta para la adición de la levadura.

int lookup[8] = { B01000, B01100, B00100, B00110, B00010, B00011, B00001, B01001}; Almacena las combinaciones de pulsos para que se muevan los stepper.

int temp=47; --> En este pin se conecta la sonda de temperatura.

OneWire ourWire(temp); --> Definición de la variable anterior según el formato de la librería OneWire.

int centigrado=0; --> Esta variable evoluciona a lo largo del proceso adoptando el valor de las temperaturas necesarias para cada paso.

DallasTemperature sensors(&ourWire);

int meascent=0;

int kai=0; --> Variable necesaria para la conversión de los valores entregados por la función keypad.

Los siguientes valores adoptan valores para tiempos de extracción según las recetas:

int chronomalt=0;

int chronolup=0;

int chronoleva=0;

8.8- SET UP DEL PROGRAMA:

En la siguiente función predefinida en todos los proyectos arduino se realizan las acciones que configurarán el funcionamiento del bucle, pero solo se ejecutarán una vez. En este caso lo que se hace es una lectura de la tarjeta microSD para identificar las recetas, se asignan los valores necesarios para el funcionamiento de cada una de las funciones implicadas en el bucle y se ejecuta el menú de selección de receta para el usuario. A continuación y sobre el código se describen en profundidad.

```
void setup() {
```

En este apartado se inicializa el server que permitirá la supervisión del proceso online:

```
 Ethernet.begin(mac, ip);
 server.begin();
```

A continuación se definen los pines como salida o como entrada dependiendo de la función que vayan a realizar en el bucle.

```
 pinMode(pinout, OUTPUT);
 pinMode(pinin, INPUT);
 pinMode(temp,INPUT);
 pinMode(caud,INPUT);
 pinMode(EMERGENCIA,INPUT);
 pinMode(res,OUTPUT);
 pinMode(mup,OUTPUT);
 pinMode(mun,OUTPUT);
 pinMode(mdp,OUTPUT);
 pinMode(mdn,OUTPUT);
 pinMode(bomb,OUTPUT);
 pinMode(valv,OUTPUT);
 pinMode(agi,OUTPUT);
```

Para realizar la misma función en los motores, al ir en conjuntos se condensa la operación mediante bucles for:

```
for (int i = 0; i <= 3; i++) {
pinMode(motorPins[i], OUTPUT);
}
for (int i = 0; i <= 3; i++) {
  pinMode(motorPinsb[i], OUTPUT);
}
for (int i = 0; i <= 3; i++) {
  pinMode(motorPinsc[i], OUTPUT);
}
```

Se inicializan las comunicaciones con los sensores y el puerto serie a 9600 baudios:

```
sensors.begin();
Serial.begin(9600);
```

Se inicializa el funcionamiento de la pantalla para presentar el menú:

```
NokiaLCD.init();   // Init screen.
NokiaLCD.clear();  // Clear screen.
```

Se inicializa el lector de tarjetas microSD, se leen los archivos ubicados en su directorio raíz, se procede a numerarlos e introducirlos en un vector en función de su posición y a presentarlos por pantalla según el nombre del archivo.

```
pinMode(4, OUTPUT);
if(SD.begin(4)){
  // Serial.println("SD Iniciado correctamente.");
}
else{
  // Serial.println("Error al iniciar SD.");
}
```

```
root = SD.open("/");

do{
  archivo = (root.openNextFile());
  if(archivo && !archivo.isDirectory()){
    //Si no hay archivo siguiente
    char recipechar=recipecount+49; //cambio a ASCII
    NokiaLCD.character(recipechar);  //Imprimo el NÚMERO
    NokiaLCD.print("- ");  //Imprimo el GUION
    NokiaLCD.print(archivo.name());  //Imprimo el nombre
      for (int x=0; x <= 8; x++){
      lin = ReadFile(x ,archivo.name());

      recipar [recipecount][x] = lin.toInt();
      delay(10);
    }
    recipecount++;
  }
}
while(archivo);
```

Aquí ya ha impreso en pantalla todas las recetas (hasta un máximo de 5) presentes como archivos de texto plano en la microSD.

Se procede a la inicialización del bucle de control.

En los proyectos arduino el proceso se realiza en una función llamada void loop, esta se ejecutará una y otra vez hasta que se desconecte el controlador de su fuente de alimentación.

Para asemejar lo más posible el funcionamiento de este controlador al de uno de nivel industrial, y dada la linealidad del proceso se han definido una serie de STEPS o pasos, que avanzan según lo hace la variable "pro" y que se corresponden con el grafcet del proceso.

A continuación y sobre el mismo programa se describe la función, condiciones de entrada y de salida de cada "step" del programa.

pro=30; --> Se inicializa el valor inicial, antes de arrancar el bucle para asegurarnos que entra al primer paso.

}

void loop() { --> Se define la función

1- Parada de emergencia --> Según las buenas costumbres en la programación de controladores siempre debe de haber una señal que detenga completamente la operación en caso de ser necesario, en este caso es la lectura en el pin llamado EMERGENCIA. Si el pin lee señal, pasará automáticamente el programa al step 39 que detiene la operación.

Como se puede comprobar este paso no tiene condición de entrada, luego la comprobación se realiza cada vez que el bucle pasa por el punto cero.

```
if (digitalRead(EMERGENCIA)==HIGH){
  pro=39;
  }
```

En el Step 30 se espera a que el usuario pulse un botón que se corresponde a una de las 5 recetas presentadas en el Set Up por pantalla. Una vez la haya seleccionado mediante la pulsación, se convertirá a un valor entero el carácter ASCII que devuelve la librería keypad.h.

Ese valor se tomará como el primer valor de la matriz sobre la que se han volcado las variables de proceso de cada una de las recetas, seleccionando pues el vector que las contiene.

Para finalizar se avanza al step 31.

```
if (pro=30){
  int key = keypad.getKey();
  kai = key-49;
  if (key != NO_KEY){
    NokiaLCD.clear();
    NokiaLCD.setCursor(1,1);
    NokiaLCD.print("Comenzando proceso");
    pro=31;
```

```
  }
 }
```

if(pro !=30){

Justo al finalizar el primer paso, en el que todavía no hay variables de sistema, y cada vez que pase el bucle por este punto se miden todas las variables de proceso que se están controlando en este prototipo, es decir, tiempo, temperatura, posición de todos los elementos y sistema de refrigeración. Se ajustan las variables necesarias y se sigue con el bucle. Este paso es muy importante porque lee los valores con los que se actualiza el sistema de vigilancia web.

En esta rutina se lleva a cabo también el sistema de regulación de temperatura conectando o desconectando la calefacción o disipación en función de las necesidades.

```
 //Control de temperatura
 sensors.requestTemperatures();
 meascent =  sensors.getTempCByIndex(0);
 //aquí se controla que la temperatura se mantenga dentro de los valores que se buscan
 if (pro!=30 && pro!=31 && pro!=38 && pro!=39){
  digitalWrite(agi, HIGH);
  if (meascent < centigrado){
   if(rescont==0){
    digitalWrite(res, HIGH);
    rescont=1;
   }
  }
  else{
   if(rescont==1){
    digitalWrite(res, LOW);
    rescont=0;
   }
  }
```

```
    if (meascent > centigrado){
    if(bombcont==0){
      digitalWrite(bomb, HIGH);
      bombcont=1;
    }
   }
   else{
    if(rescont==1){
      digitalWrite(bomb, LOW);
      bombcont=1;
    }
   }
  }
```

A partir de este punto cada uno de los steps se corresponde con una parte del proceso artesano de preparación de cerveza descrito en puntos anteriores.

STEP 31 --> En este paso se llena el reactor con el agua indicada en la receta, para ello se abre la electroválvula de llenado y se empiezan a contar ciclos del caudalímetro efecto hall. Cada 450 pulsos del caudalímetro se corresponden a un litro de agua.

Se mantiene abierta la válvula mientras el valor del contador de pulsos sea inferior al valor alojado en el punto cero del vector de variables del sistema multiplicado por 450, posteriormente se cierra y se modifica la variable pro a 32.

```
  if (pro ==31){
   NokiaLCD.clear();
   NokiaLCD.setCursor(1,1);
   NokiaLCD.print("Llenando el reactor con agua");
   digitalWrite(valv, HIGH);
```

```
do{
    if ( digitalRead(caud) == HIGH )
    {
      if ( digitalRead(caud) == LOW )
      {
        watpul++;            //Incrementa el contador
        delay (10);         // Retardo
      }
    }
  }
  while(watpul<=450*recipar[kai][0]);
  digitalWrite(valv, LOW);
  pro=32;
 }
```

En el proceso de elaboración de cerveza es muy importante respetar las temperaturas para cada proceso, sobretodo en la fermentación y extracciones, ya que de no ser así podría aparecer en nuestro producto final algún componente indeseado, tanto a nivel orgánico (contaminación bacteriana de la cerveza) como a nivel molecular, que alteraría el sabor, aroma e incluso salubridad de nuestro producto.

En este step se realiza un precalentamiento del agua antes de proceder a realizar la extracción sólido líquido de los componentes, la temperatura vendrá determinada por la receta a emplear, ya que para cada tipo de cereal es distinta.

En primer lugar, el programa limpia la pantalla del mensaje del step anterior, posteriormente procede a informar al usuario de en qué parte del proceso se encuentra y finalmente toma el valor del segundo elemento de la matriz de variables del sistema y lo introduce en la variable centígrado, de forma que al pasar por el punto de control de variables en cada ciclo del controlador, se mantendrá a ésta.

```
if (pro ==32 && pro!=31){
  NokiaLCD.clear();
  NokiaLCD.setCursor(1,1);
  NokiaLCD.print("Calentando  hasta      temperatura optima de   extraccion");
  centigrado = recipar[kai][1];
  if(meascent>=centigrado){
    pro=33;
  }
 }
```

En el siguiente paso, y una vez la temperatura se ha estabilizado en el valor óptimo según el usuario, se procede a comprobar si el saco de malta ya se ha sumergido, esto se realiza mediante un valor binario que cambia cuando se activa el sistema de grúa.

Si es la primera vez que se pasa por el bucle, en primer lugar se introducirá el valor de millis() en la variable chrono y se activará la variable binaria de control para no volver a hacerlo hasta el siguiente step. Posteriormente se conectará el motor del sistema de cebada en sentido de bajada y se cambiará el valor a 1 del binario de control.

En cada ciclo del controlador se comparará el tiempo mientras la temperatura queda controlada por el programa fuera de step. Al finalizar el tiempo se volverá a activar el motor del manejo de sólidos, esta vez en sentido contrario y se devolverá el valor de control a 0.

```
if (pro ==33 && pro!=32){
  NokiaLCD.clear();
  NokiaLCD.setCursor(1,1);
  NokiaLCD.print("Realizando  extraccion  solido      liquido de  la malta");
  if (chronomalt==0){
    chrono=millis();
    chronomalt=1;
  }
  if (maltcon==0){
    digitalWrite(mup, HIGH);
```

```
    digitalWrite(mun, HIGH);
    delay(1000);
    maltcon=1;
    chronocont = millis()-chrono;
   }
   if(chronocont<recipar [kai][2]){
    if(maltcon!=0){
     digitalWrite(mdp, HIGH);
     digitalWrite(mdn, HIGH);
     delay(1000);
     maltcon=0;
    }
    pro=34;
   }
  }
```

En el siguiente paso, se lleva el sistema a la temperatura de extracción del primer lúpulo, al que llamaremos lúpulo aséptico, pues se empleará en la mezcla para prevenir la aparición de bacterias en nuestro producto. Esta temperatura es próxima al valor de ebullición, aunque sin alcanzarlo, ya que no es necesario para la eliminación de los patógenos. Cabe destacar en este punto que aunque la presión de vapor aumentará al trabajar a esta temperatura, la pérdida de producto no será tan significativa al no alcanzar la ebullición, aún así, deberá ser prevista en el llenado del sistema al programar las recetas.

Este bucle sigue el mismo procedimiento que el paso 32, fija un nuevo set point en la variable centígrado y deja que el programa lleve el sistema hasta él.

```
  if (pro ==34 && pro!=33){
   NokiaLCD.clear();
   NokiaLCD.setCursor(1,1);
   NokiaLCD.print("Calentando a segunda   temperatura de extraccio");
   centigrado = recipar[kai][3];
```

```
  if(meascent>=centigrado){
    pro=35;
  }
 }
```

El step 35, sigue la misma filosofía que en el paso 33, con la diferencia que aquí hay tres eventos que controlar mediante tiempo. En primer lugar se mide el tiempo que debe permanecer la infusión de cereal a temperatura, al finalizar el cuál, se introduce el primer lúpulo en el sistema. En este momento se introduce el primer lúpulo en el sistema y se vuelve a poner a contar el tiempo. Una vez finalizado, y manteniendo esta carga de lúpulo dentro del fluido se introduce la segunda y se mide el tiempo final.

Pasado este tiempo se retiran ambos lúpulos simultáneamente y se puede proceder al enfriado en el siguiente paso.

```
 if (pro ==35 && pro!=34){
  NokiaLCD.clear();
  NokiaLCD.setCursor(1,1);
  NokiaLCD.print("Realizando  extraccion  del lupulo");
  if (chronolup==0){
   chrono=millis();
   chronolup=1;
  }
  chronocont = millis()-chrono;
  if(chronocont>=recipar [kai][4] && stepcona==0){
   moveSteps(STEPS_PER_REVOLUTION);
   stepcona = 1;
  }
  if(chronocont>=recipar [kai][5] && stepconb==0){
   moveStepsb(STEPS_PER_REVOLUTION);
   stepconb=1;
```

```
  }
  if(chronocont>recipar [kai][6]){
   stepcona = 0;
   moveSteps(-STEPS_PER_REVOLUTION);
   stepconb=0;
   moveStepsb(-STEPS_PER_REVOLUTION);
   pro=36;
  }
 }
```

En el paso 36 se debe reducir la temperatura hasta una que sea adecuada para la adición de la levadura sin que muera. Para ello se conecta el disipador de calor, consistente en un circuito de cobre insertado en el sistema, que circula agua intercambiando calor hasta un disipador aletado de aluminio con ventilación forzada.

Para ello modifica la variable centigrado y deja que el programa gestione el sistema de refrigeración. Una vez la temperatura es adecuada, adiciona la levadura girando un 25% la clapeta, de forma que además, el hueco queda cubierto.

```
 if (pro ==36 && pro !=35){
  NokiaLCD.clear();
  NokiaLCD.setCursor(1,1);
  NokiaLCD.print("Enfriando   hasta      temperatura de siembra");
  centigrado = recipar[kai][7];
  if(meascent<=centigrado){
   if (levacont ==0){
    moveStepsc(STEPS_PER_REVOLUTION/4);
    levacont=1;
   }
   pro=37;
  }
 }
```

A partir de aquí solo queda controlar las variables del proceso, manteniendo la temperatura de fermentación lo más estable posible para que el producto final tenga una reproducibilidad máxima. El tiempo de fermentación también vendrá determinado por la receta.

```
if (pro ==37 && pro!=38){
  NokiaLCD.clear();
  NokiaLCD.setCursor(1,1);
  NokiaLCD.print("Fermentando:Tu cerveza  esta casi   lista");
  if (chronoleva ==0){
   chrono=millis();
   chronoleva=1;
  }
  chronocont = millis()-chrono;
  centigrado = recipar[kai][8];
  if(chronocont>recipar [kai][9]){
   pro=38;
  }
}
```

Una vez finalizado el proceso, la cerveza deberá ser filtrada y embotellada. Estos pasos ya no son realizados por el dispositivo, pero suponen un tiempo similar al de carga y por el diseño específico del grifo de vaciado y su posición estratégica se ven todavía más favorecidos. El step 38 es pues solo un mensaje por pantalla y conduce incondicionalmente al step 39.

```
if (pro ==38 && pro !=37){
  NokiaLCD.clear();
  NokiaLCD.setCursor(1,1);
  NokiaLCD.print("Cerveza     lista para  filtrar y   embotellar  ENHORABUENA");
  pro=39;
}
```

El último step no hace nada, además su condición de entrada desactiva los controles de temperatura.

```
if (pro ==39 && pro != 38){

}
```

8.9 SUPERVISION WEB

Independientemente de los pasos numerados propios del grafcet y de las subrutinas de control de temperatura, una de las ventajas de este código es que se puede supervisar a distancia mediante una conexión Ethernet. En ese caso, nuestro controlador actuará también como un servidor web alojado en el local de producción. Para la conexión desde dentro de la red bastará con introducir la ip definida en la configuración en cualquier navegador web. Desde un punto externo habrá que definir en el router un puerto de conexión predeterminado, escribir la ip pública del router y especificar el puerto en el navegador (pero es igualmente posible).

En sistemas con ip dinámica se recomienda usar servicios de DNS dinámica, como noip.com, que además es gratuito.

El programa comprueba en primer lugar que haya un cliente conectado, y de ser así y mientras dure la conexión escribe en el cliente una página HTML, donde cada println se corresponde con una línea del código web. Para identificar el punto del proceso en que se encuentra emplea un selector switch case, donde la condición es la variable pro, empleada para la selección de steps. Justo después, se produce una impresión por pantalla de las distintas variables del proceso que se muestran también al cliente.

```
EthernetClient client = server.available();
if (client) {
  boolean currentLineIsBlank = true;
  while (client.connected()) {
    if (client.available()) {
      char c = client.read();
      if (c == '\n' && currentLineIsBlank) {
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println("Connection: close");
```

```
client.println("Refresh: 15");

client.println();

client.println("<!DOCTYPE HTML>");

client.println("<html>");

client.println("<h1 style=\"background-color:black;font-family:arial;color:white;font-size:30px;\" >");

client.print("POLIBREW web control service");

client.println("</h1>");

client.println("<br />");

client.println("<h2 style=\"background-color:white;font-family:arial;color:black;font-size:20px;\" >");

client.print("Proceso: ");

switch (pro) {

case 31:

 client.print("Llenando el reactor con agua");

 client.println("<br />");

 break;

case 32:

 client.print("Calentando hasta temperatura óptima de extracción");

 client.println("<br />");

 break;

case 33:

 client.print("Realizando extracción sólido líquido de la malta y la cebada");

 client.println("<br />");

 break;

case 34:

 client.print("Calentando hasta temperatura de extracción del lúpulo");

 client.println("<br />");
```

```
  break;
 case 35:
  client.print("Realizando extracción del lúpulo");
  client.println("<br />");
  break;
 case 36:
  client.print("Enfriando hasta temperatura de siembra");
  client.println("<br />");
  break;
 case 37:
  client.print("Fermentando: Tu cerveza está casi lista");
  client.println("<br />");
  break;
 case 38:
  client.print("Tu cerveza está lista para filtrar y embotellar ENHORABUENA BREWMASTER!!");
  client.println("<br />");
  break;
 default:
  client.print("No se está produciendo ningún proceso");
  client.println("<br />");
 }
 client.println("<br />");
 client.print("Variables del proceso: ");
 client.println("<br />");
 client.print("Temperatura medida:  ");
 client.print(meascent);
 client.println("<br />");
```

```
        client.print("Temperatura buscada: ");
        client.print(centigrado);
        client.println("<br />");
        client.println("</h2>");
        client.println("<br />");
        client.println("<br />");
        client.println("<br />");
        client.print("Versión ALPHA para el TFG:  Diseño del sistema de control de un fermentador para elaboración doméstica de cerveza");
        client.println("<br />");
        client.print("Agradeciendo la inestimable colaboración de mi tutor D. Javier Sanchis y la formación de la UPV");
        client.println("<br />");
        client.print("Miguel Martínez -- 2015");
        client.println("</html>");
        break;
      }
      if (c == '\n') {
        currentLineIsBlank = true;
      }
      else if (c != '\r') {
        currentLineIsBlank = false;
      }
    }
  }
  delay(1);
  client.stop();
  Serial.println("client disconnected");
```

```
 }
}
}
```

8.10 FUNCIONES ADICIONALES PARA LOS STEPPER

El código previsto a continuación se emplea para el correcto funcionamiento de los stepper. Se debe a que al tratarse de motores de manufactura china no cumplen el estándar de este tipo de motores y se ha tenido que improvisar una pseudo-librería que los active. Por cuestiones de comodidad, se ha repetido la función para cada uno de los stepper implicados en la operación del reactor.

```
void moveSteps(int steps) {
 int i;
 if (steps > 0) {
  for (i = 0; i < steps; i++) {
   moveClockWise();
  }
 }
 if (steps < 0) {
  for (i = steps; i <= 0; i++) {
   moveAntiClockWise();
  }
 }
}


void moveClockWise() {
 for (int j = 7; j >= 0; j--) {
  setMotor(j);
  delayMicroseconds(MOTOR_SPEED);
 }
}
```

```
void moveAntiClockWise() {
 for (int j = 0; j <= 7; j++) {
  setMotor(j);
  delayMicroseconds(MOTOR_SPEED);
 }
}

void setMotor(int j) {
 for (int i = 0; i < 4; i++) {
  digitalWrite(motorPins[i], bitRead(lookup[j], i));
 }
}

//funciones para los stepperb
void moveStepsb(int steps) {
 int i;
 if (steps > 0) {
  for (i = 0; i < steps; i++) {
   moveClockWiseb();
  }
 }
 if (steps < 0) {
  for (i = steps; i <= 0; i++) {
   moveAntiClockWiseb();
  }
 }
}
```

```
void moveClockWiseb() {
 for (int j = 7; j >= 0; j--) {
   setMotorb(j);
   delayMicroseconds(MOTOR_SPEED);
 }
}

void moveAntiClockWiseb() {
 for (int j = 0; j <= 7; j++) {
   setMotorb(j);
   delayMicroseconds(MOTOR_SPEED);
 }
}

void setMotorb(int j) {
 for (int i = 0; i < 4; i++) {
   digitalWrite(motorPinsb[i], bitRead(lookup[j], i));
 }
}

//funciones para los stepperc
void moveStepsc(int steps) {
 int i;
 if (steps > 0) {
   for (i = 0; i < steps; i++) {
     moveClockWiseb();
```

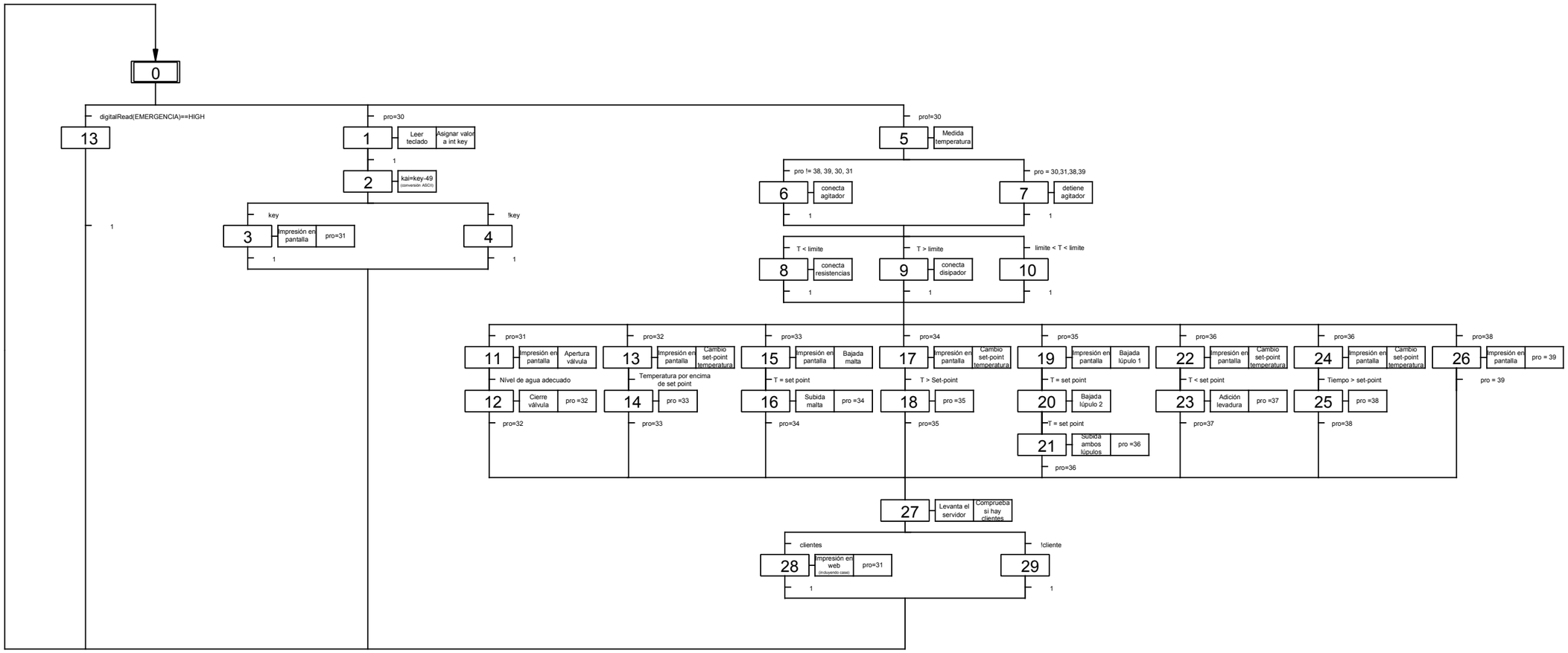
```
    }
  }
  if (steps < 0) {
    for (i = steps; i <= 0; i++) {
      moveAntiClockWiseb();
    }
  }
}

void moveClockWisec() {
  for (int j = 7; j >= 0; j--) {
    setMotorb(j);
    delayMicroseconds(MOTOR_SPEED);
  }
}

void moveAntiClockWisec() {
  for (int j = 0; j <= 7; j++) {
    setMotorb(j);
    delayMicroseconds(MOTOR_SPEED);
  }
}
void setMotorc(int j) {
  for (int i = 0; i < 4; i++) {
    digitalWrite(motorPinsb[i], bitRead(lookup[j], i));
  }
}
```

8.11 FUNCIÓN ADICIONAL PARA LECTURA DE ARCHIVOS

Para la función de lectura de archivos también ha sido necesario emplear un código añadido a la librería estándar, concretamente esta función ReadFile, extraída del blog domotica-arduino.es

```
String ReadFile(int Linea,char Ruta[]){
 int Lin=0;
 String Resultado;
 File myFile;
 byte Bin;
 myFile = SD.open(Ruta);
 if (myFile) {
  while (myFile.available()) {
   Bin=myFile.read();
   if (Bin==13){
    Lin++;
    myFile.read();
   }
   else
   {
    if (Lin==Linea){
     Resultado=Resultado+(char(Bin));
    }
    if (Lin>Linea){
     myFile.close();
     return Resultado;
    }
   }
 }
```

```
    myFile.close();
  }
  else {
    return"NOFILE!!";
  }
}
```



TRABAJO FINAL DE GRADO EN INGENIERÍA EN QUÍMICA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

Proyecto: DISEÑO DEL SISTEMA DE CONTROL DE UN FERMENTADOR PARA ELABORACIÓN DOMÉSTICA DE CERVEZA

Plano: GRAFCET

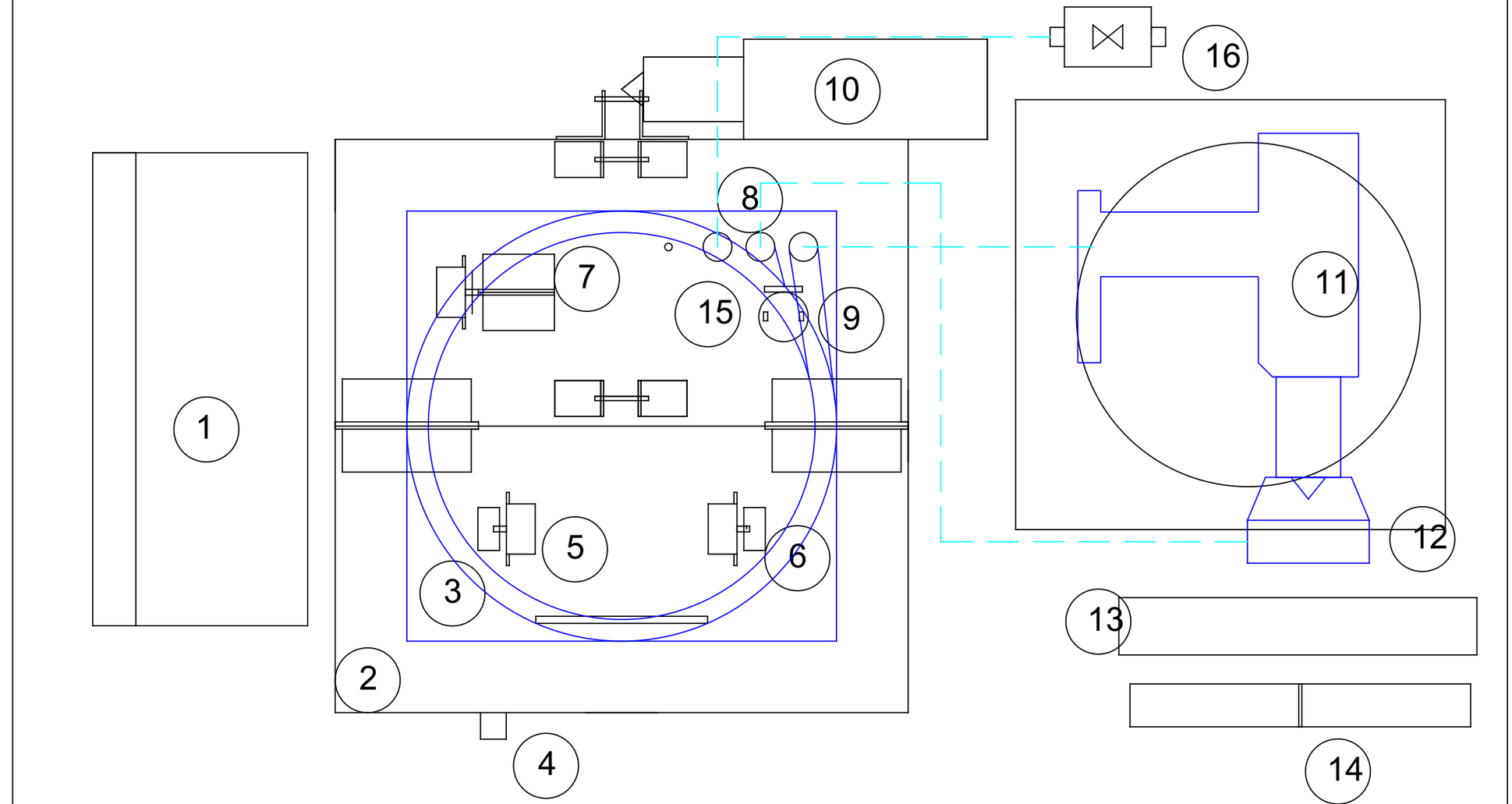
Septiembre 2015

Fecha: 31/08/2015

Nº Plano: 1

Autor: MIGUEL MARTÍNEZ TOMÁS

Escala: n/a



1 - CUADRO ELÉCTRICO

2 - CÁMARA DE AIRE

3- REACTOR

4 - SALIDA DE PRODUCTO

5 - SISTEMA DE ADICIÓN LÚPULO 1

6 - SISTEMA DE ADICIÓN LÚPULO 2

7 - SISTEMA DE ADICIÓN DE LEVADURA

8 - TOMAS DE AGUA DE REACTOR Y REFRIGERACIÓN

9 - AGITADOR

10 - SISTEMA DE ADICIÓN DE CEBADA Y MALTA

11 - VASO DE EXPANSIÓN DEL SISTEMA DE REFRIGERACIÓN

12 - SISTEMA DE CIRCULACIÓN DEL SISTEMA DE DISIPACIÓN

13 - DISIPADOR AEROTERMO

14 - VENTILACIÓN FORZADA DEL DSIPADOR

15 - SERPENTÍN DE DISIPACIÓN

16 - ELECTROVÁLVULA DE ALIMENTACIÓN

LÍNEAS DE AGUA

ELEMENTOS OCULTOS

TRABAJO FINAL DE GRADO EN INGENIERÍA EN QUÍMICA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

Proyecto: DISEÑO DEL SISTEMA DE CONTROL DE UN FERMENTADOR PARA ELABORACIÓN DOMÉSTICA DE CERVEZA

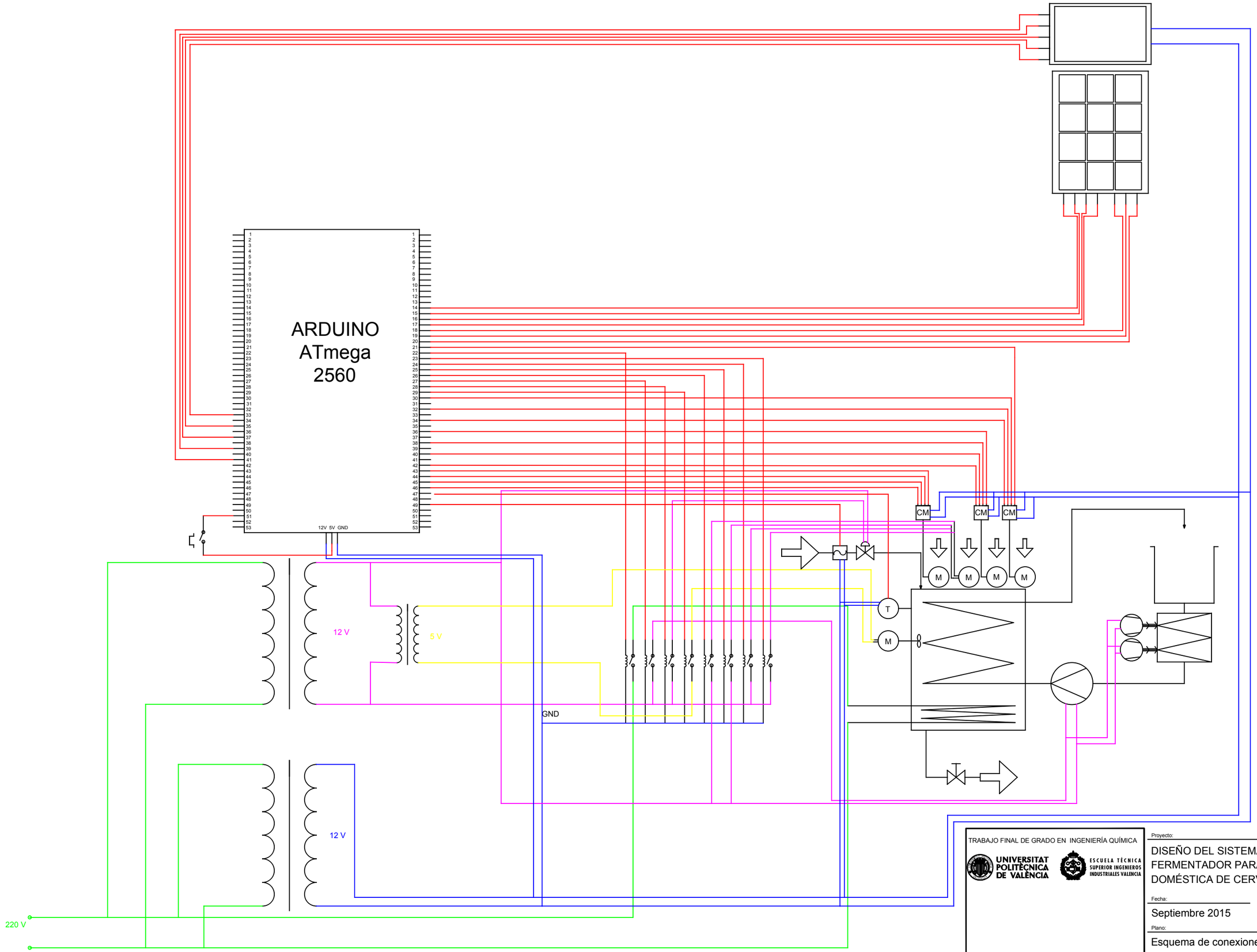
Plano: IMPLANTACIÓN PLANTA PILOTO

Fecha: Septiembre 2015 — 31/08/2015

Autor: MIGUEL MARTÍNEZ TOMÁS

Escala: n/a

Nº Plano: 2



TRABAJO FINAL DE GRADO EN INGENIERÍA QUÍMICA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

Proyecto:

DISEÑO DEL SISTEMA DE CONTROL DE UN FERMENTADOR PARA ELABORACIÓN DOMÉSTICA DE CERVEZA

Fecha: Septiembre 2015

Escala: N/A

Plano: Esquema de conexiones eléctricas

Nº Plano:

Miguel Martínez Tomás

Autor proyecto