Universidad de La Laguna

ESCUELA POLITÉCNICA SUPERIOR DE INGENIERÍA
Sección Náutica, Máquinas y Radioelectrónica Naval

Trabajo presentado para la obtención del tìtulo de:

GRADUADO EN TECNOLOGÍAS MARINAS

Desarrollo de una simulación básica de una caldera de vapor mediante Arduino

Presentado por

Jesús Aday Cabrera Fragiel

Presentado en "Septiembre de 2016"

ULL | Universidad de La Laguna

Resumen

Resumen Este trabajo aporta una nueva idea en lo que respecta a la ampliación de conocimientos con la creación de un programa que simule el funcionamiento de una caldera. Para ello, se necesita programar una placa de hardware libre, Arduino. En su diseño se han previsto diferentes variables que afectan al funcionamiento interno de la caldera: la alimentación de agua, los parámetros de funcionamiento del quemador, etc.

La programación de Arduino se realizó mediante un lenguaje de programación similar al C++, basado en Wiring. Una vez finalizada la programación y a través de la utilización de diodos y una pequeña pantalla LCD, puede observarse la reacción de la caldera a lo largo de un determinado período de tiempo. En este sentido, el simulador puede controlarse remotamente mediante el uso de un control manual o por un controlador PID externo.

En definitiva, este proyecto pretende ampliar el marco de conocimientos tanto del comportamiento y funcionamiento de una caldera, así como de la programación y funcionamiento de la placa conocida como Arduino.

La creación de este tipo de programas puede servir de modelo práctico de funcionamiento de una caldera “real” ya que los parámetros para la realización de las curvas de las variables a lo largo del tiempo puede ajustarse a una caldera existente.

Abstract

This work attemps to provide a new idea for knowledge acquisition, taking just the creation of a program that simulate the operation of a boiler. All this will be implemented by programming a plate free hardware Arduino. For its design will take into account different variables that affect the inner workings of the boiler feed water, the operating parameters of the burner, etc; without forgetting the time constant.

Arduino programming will be done through a programming language with some similarities with /mbox(C++), based on Wiring. With said plate, once the programming is ended, through the diodes and a small LCD screen, the reaction of the boiler will be observed during the course of time. Besides, the simulator can be externally controlled by a manual control or by an external PDI controller.

This Project aims to achieve the knowledge's acquisition of the behavior and operation of a boiler, and also the programming and operation of an Arduino board.

The creation of such programs can serve as a practical model of running for a "real" boiler because the parameters for carrying out the curves of the variables over time would be consistent with an existing boiler.

Índice general

Índice de figuras

Índice de tablas

1 Motivación

A lo largo del transcurso de mi etapa como estudiante en estos años, me he podido percatar de las dificultades que tenemos tanto los alumnos como los educadores para la adquisición o impartición de conocimientos y el desarrollo de éstos de forma práctica.

Bien sea por la crisis u otras cuestiones, lo cierto es que parece que el modelo de estudios actual se vuelca sobre una enseñanza en su gran mayoría teórica, postergando la adquisición de los conocimientos prácticos a las prácticas de empresa, las cuales, bajo mi punto de vista, no están correctamente posicionadas durante los años de duración del grado que corresponde. En este sentido suelen aglutinarse en los últimos años de carrera bajo el razonamiento de estar preparados a nivel de conocimientos y destrezas para su correcta realización. No obstante, considero que dichos conocimientos y destrezas se van aletargando e incluso olvidado precisamente al no ponerlos en práctica con una mayor frecuencia.

De hecho, uno de los pocos elementos prácticos con los que podemos interaccionar es el simulador de sala de máquinas, una herramienta muy potente que nos permite visualizar el funcionamiento y los valores como si nos encontrásemos en una instalación real, pero su aprendizaje conlleva la inversión de una gran cantidad de horas, y su explotación está limitada debido a que se sitúa dentro de la propia facultad y la accesibilidad por el número de plazas disponibles es limitada. También hay que contar con que el simulador realiza una simplificación de los sistemas en la mayoría de casos permitiéndonos actuar sobre ciertos elementos en específicos sin que podamos modificar otros.

Por todo ello, barajé la siguiente idea: ¿y si fuéramos los estudiantes los que creásemos nuestros propios simuladores? Esto nos permitiría, aprender sobre ciertos ámbitos o campos que no se enseñan explícitamente en la carrera como, por ejemplo, programar, entre otros. Y además, ampliaríamos conocimientos sobre los temas impartidos en las asignaturas obteniendo una idea más precisa de los mismos, así como de la puesta en práctica y aplicación real.

2 Introducción y Antecedentes

Arduino es una plataforma de hardware y software libre que permite realizar diseños de sistemas de forma relativamente fácil e intuitiva[1]. Al tratarse de un software libre, ha tenido una gran difusión evolucionando su hardware desde su primera versión conocida como Arduino Galileo, hasta la gran cantidad de "Arduino" que nos podemos encontrar hoy en día específicos para una gran variedad de ámbitos.

Arduino trabaja con estructura simple: un void setup, donde se localiza todo aquello que la placa necesita reconocer en un primer momento para su funcionamiento, es decir, es su configuración: conexiones de entrada, conexiones de salida, etc; y una segunda estructura llamada void loop, lugar donde se sitúa el proceso principal, donde se halla el código principal el cuál se encarga de realizar todos los procesos de forma cíclica[3] (ver fig. 2.1).

Para el manejo con Arduino hay que tener en cuenta ciertas funciones principales que sera utilizadas en repetidas ocasiones.Ver tabla 2.2.

Éstas no son las únicas funciones que existen, hay muchas otras para la realización de las diversas competencias accesibles a través de librerías descargables gratuitas.

A la hora de la creación de un software para Arduino es necesario tener claro el elemento a simular, en nuestro caso se trata de una caldera de vapor equivalente a una real en el simulador de sala de máquinas de la facultad (MC-90[4]).

Para tener una idea de cómo serán los procesos principales del programa se utiliza un diagrama de flujo, donde los definimos. Para ello nos basaremos en ciertos procesos básicos: Control de Llenado, Control del Quemador, Sistema de Llenado, Regulación Combustible/Aire, etc. Quedando otros sistemas postergados a futuras actualizaciones/revisiones del simulador.

Para su concepción es necesaria la utilización de variables como las de presión (P), volumen (V), etc. Además de constantes que fijen los niveles de los estados como por ejemplo: el volumen máximo (VM), la presión de alarma del sistema (PA)...

Tabla 2.1: Descripción de Funciones

int	Declara una variable como tipo entero.
float	Declara una variable como tipo flotante.
If(...) else(...)	If, es un estamento que se utiliza para saber si una condición determinada se ha cumplido y, en caso de no ser así, ejecutar el else.
PinMode(...)	Configura el modo de trabajo de un Pin pudiendo ser IMPUT u OUTPUT.
analogRead(pin)	Lee el valor de un pin establecido como una entrada analógica.
digitalWrite(pin,value)	Envía un PIN que se halla indicado como salida el valor HIGH o LOW dependiendo de su activación.
delay(m/s)	Mantiene en pausa la ejecución del programa durante un tiempo.

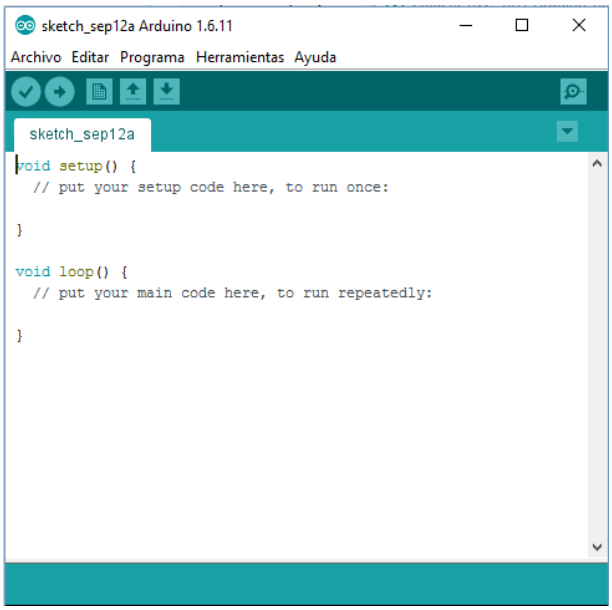

Figura 2.1: Estructura del software principal de Arduino

Tabla 2.2: Descripción de Funciones[2]

int	Declara una variable como tipo entero.
float	Declara una variable como tipo flotante.
If(...) else(...)	If, es un estamento que se utiliza para saber si una condición determinada se ha cumplido y, en caso de no ser así, ejecutar el else.
PinMode(...)	Configura el modo de trabajo de un Pin pudiendo ser IMPUT u OUTPUT.
analogRead(pin)	Lee el valor de un pin establecido como una entrada analógica.
digitalWrite(pin,value)	Envía un PIN que se halla indicado como salida el valor HIGH o LOW dependiendo de su activación.
delay(m/s)	Mantiene en pausa la ejecución del programa durante un tiempo.

Una vez obtenida la lista de variables y constantes se realiza la programación del software de Arduino, fijando todas aquellas variables y constantes que lo requieran a los pines digitales o analógicos que le aportaran la señal ya sea entrante (INPUT) o saliente (OUTPUT), siendo estos últimos indicados en el apartado de configuración del software de Arduino (void setup;).

Una vez finalizado todo lo anterior se procede a la elaboración de los algoritmos dentro del apartado de void loop, para que estos se repitan de forma cíclica. Para la estructura de los algoritmos es necesario seguir la siguiente estructura:

a) Llamar a las variables, obteniendo una lectura de su estado actual y fijándolo.

b) Llevar a cabo la realización de las condiciones necesarias mediante comandos if para la ejecución de las diferentes partes del algoritmo.

c) Señalización externa de la activación o desactivación de la variable mediante LEDs o displays LCD.

Una vez realizados los algoritmos se procede al conexionado de los elementos electrónicos necesarios para la simulación, realizando testeos y modificando los algoritmos necesarios para el cumplimiento de cada uno de los procesos.

Con ello queremos conseguir la simulación de una caldera de vapor que utilice las variables de volumen como referencia de la cantidad de agua de alimentación en la caldera, y presión como variable final a obtener. Se tendrá en cuenta tanto el control de volumen para realizar los procesos de llenado de la caldera (siendo el caudal de alimentación modificado mediante un potenciómetro para jugar con un caudal de agua entrante menor o mayor que el consumido), así como el proceso de preparación del quemador y posterior control de presión, pudiendo manejar la regulación de combustible/aire para la aparición de “Fallo de Llama” y la presión de demanda para generar mediante su variación con un potenciómetro una demanda mayor que la producida.

3 Solución Propuesta

Mi idea ha sido la creación de un sistema que reproduzca el funcionamiento básico de llenado y control del quemador de una caldera de vapor. Para ello he utilizado una placa de Arduino Mega 2560 (ver fig. 3.1). La utilización de Arduino Mega ha sido debida a la gran cantidad de puertos (pines) para entrada o salida de datos, tanto digitales como analógicos disponibles en esta placa, si bien es posible la conexión entre sí de varios Arduino (por ejemplo, UNO) para la obtención de más puertos.

Para la programación de Arduino es necesaria la utilización de un lenguaje de programación propio, similar al C++ utilizando como plataforma de escritura la versión 1.6.11. de su propio software. Además, es necesario la creación de sistemas independientes encargados de realizar individualmente una función específica del sistema principal, (ver tabla 3.1). Estos, además, deben seguir un proceso de funcionamiento determinado (ver fig. 3.2).

Cada sistema está encargado de realizar unos procesos en concreto, pero para ellos es necesario la aplicación constantes que limiten el rango de funcionamiento, (ver tabla 3.2).

Además de las constantes contamos con variables que se modificarán a lo largo del funcionamiento del programa, (ver tabla 3.3).

3.1 Procesos del Sistema

Tras el diseño del diagrama principal del que se compondrá el void loop, proseguimos a la elaboración interna de cada uno de los procesos a partir del rearme del sistema:

3.1.1 Rearme Sistema

Proceso de rearme (ver figs. 3.4 y 3.3):

a) La función lee el estado del pulsador de rearme.

b) Si el pulsador está activado, desactiva el LED de señalización, cambia el estado de la variable alarma a desactivado, e indica que el rearme se ha activado en el display LCD.

c) Si el pulsador está desactivado, se mantiene activado el LED de señalización, y se indica que el rearme está desactivado en el display LCD.

3.1.2 Purga

El sistema de "purga" representa la activación por alguna circunstancia de una válvula que conllevase la pérdida de volumen de agua de alimentación de la caldera, poniendo en peligro la misma (ver figs. 3.6 y 3.5).

Proceso de purga:

a) La función lee el estado del pulsador de purga.

b) Si el pulsador está activado, activa el LED de señalización, indica la activación de la purga en el display LCD, y activa la función de alarma.

Figura 3.1: Arduino Mega 2560

Tabla 3.1: Descripción de Sistemas

Rearme Sistema	Elemento que imposibilitará el funcionamiento del resto de algoritmos si no se encuentra activado.
Purga	Simulador de una apertura de válvulas de alimentación simulando la posible pérdida inmediata de ésta.
Control de Volumen	Sistema encargado de analizar el volumen actual de agua de alimentación de la caldera de vapor.
Control de Llenado	Sistema encargado del aumento del volumen de agua de alimentación.
Control del Quemador	Produce el encendido del quemador mediante la activación de un pulsador.
Barrido	Sistema encargado de producir el barrido para el encendido del quemador.
Control de Presión	Sistema encargado de controlar la presión actual de la caldera y responder según unos valores límites.
Producción de vapor	Se encarga de realizar los cálculos y otorgar los nuevos valores de presión y volumen.
Control de la Presión de Demanda	Sistema encargado de recoger la variación de demanda debido a un potenciómetro.
Regulador Combustible/Aire	Sistema encargado de analizar la relación Combustible/Aire aportado mediante un potenciómetro.
Potenciometro Caudal de Agua de Alimentación	Sistema encargado de recoger la variación de caudal de agua de alimentación mediante un potenciómetro.
Alarma	Sistema encargado de parar el funcionamiento del programa debido a un cumplimiento de variables límites.

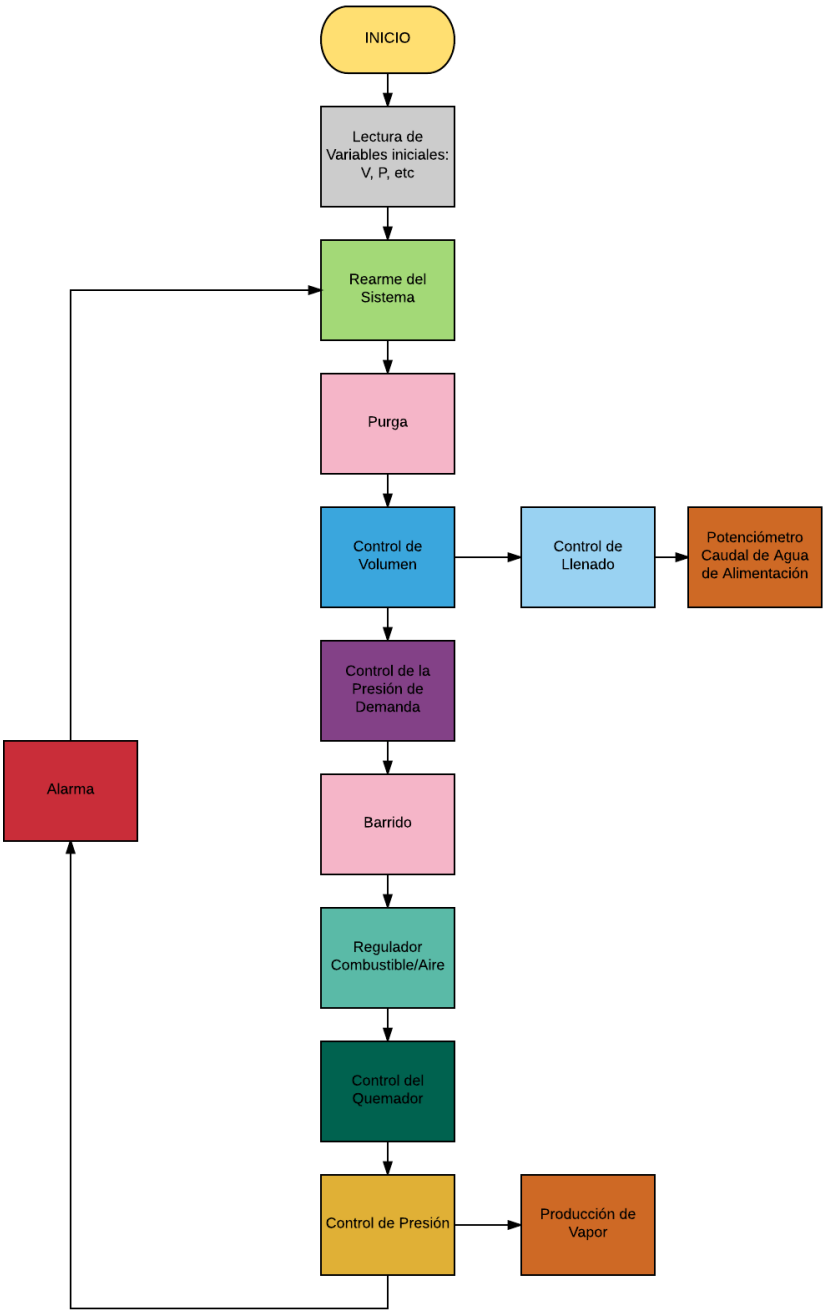

Figura 3.2: Descripción del Sistema

Tabla 3.2: Descripción de las Constantes

VA	Volumen de Alarma: indica el volumen mínimo de agua de alimentación que causara la activación del proceso de alarma.
VL	Volumen Límite: indica el volumen en el que contaríamos con un margen para el funcionamiento de la caldera.
VM	Volumen Máximo: marca un máximo para el volumen de llenado.
MAXQagua	Máximo Caudal de Agua (m3/s): fija el caudal de agua de alimentación máximo que podrían aportar las bombas.
PA	Presión de Alarma (atm): proporciona el valor máximo de presión tras el cual se activara el proceso de alarma.
VQuemador	Volumen Quemador: hace referencia al consumo de volumen de agua de alimentación por el quemador.
PQuemador	Presión Quemador: es la presión generada por el quemador.
MAXPDemanda	Máxima Presión de Demanda: fija el valor máximo de presión de demanda que puede aportar el potenciómetro.

Tabla 3.3: Descripción de las Variables

RearmeSistema	Guarda el estado del sistema de Rearme (Activado/Desactivado).
Purga	Guarda el estado del sistema de Purga (Activado/Desactivado).
Alarma	Guarda el estado del sistema de Alarma (Activado/Desactivado).
V	Volumen: marca el volumen actual del programa.
ValvulaEnt	Válvula de Entrada: permite saber si se encuentra activada o desactivada.
SensorLlama	Permite saber el estado del quemador (Activado/Desactivado).
Barrido	Nos permite saber si se ha realizado la maniobra de barrido (Activado/Desactivado).
Quemador	Indica el estado del quemador (Activado/Desactivado).
Automatico	Indica el estado del sistema automático de arranque/parada del quemador
P	Presión: es la presión actual del sistema.
PDemanda	Presión de Demanda: es la presión que se demanda mediante el manejo del potenciómetro.
ReguladorPresionD	Regulador Presión de Demanda: recoge el dato del estado del potenciómetro de demanda.
PresionD	Presión Demandada: modifica el valor del Regulador PresionD para llevarlo a un rango utilizable.
ReguladorQAgua	Regulador Caudal de Agua: recoge el dato del estado del potenciómetro de caudal de agua.
QAguaF	Caudal de Agua Final: es el caudal de agua entrante final que se obtiene mediante el manejo del potenciómetro.
QAgua	Caudal de Agua: modifica el valor del Regulador QAgua para llevarlo a un rango utilizable.
ReguladorCA	Regulador Combustible/Aire: recoge el dato del estado del potenciómetro de la mezcla de combustible Aire.
CA	Combustible/Aire: modifica el valor del Regulador CA para llevarlo a un rango utilizable.
CAQuemador	Combustible/Aire Quemador: Interfiere en la activación/desactivación del quemador debido a una mala mezcla de combustible/Aire.

```
void REARME_DEL_SISTEMA() {
RearmeSistema = digitalRead(PulsadorRearme);

 if (RearmeSistema == HIGH) {
    digitalWrite(LedRearme, LOW);
    lcd.setCursor(5,1);
    lcd.print("ON ");
    Alarma = LOW;
         lcd.setCursor(0,3);
     lcd.print("                    ");
}

else{
    digitalWrite(LedRearme,HIGH);
    lcd.setCursor(5,1);
    lcd.print("OFF");
    delay(1000);
}
}
```

Figura 3.3: Programación: Rearme del Sistema

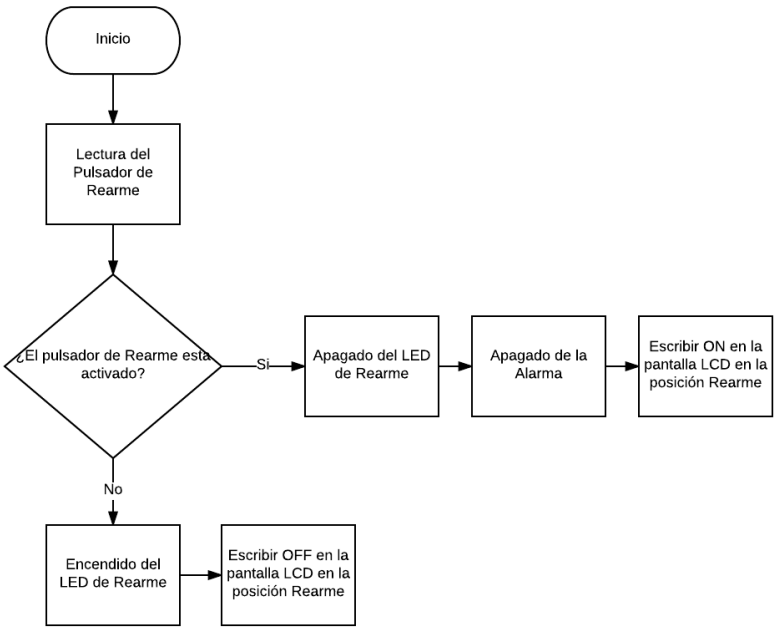

Figura 3.4: Diagrama de flujo: Rearme del Sistema

```
void PURGA(){

 if (Purga == LOW){
    digitalWrite(LedPurga, LOW);
     lcd.setCursor(6,2);
      lcd.print("OFF");

}
  else{
    digitalWrite(LedPurga, HIGH);
    lcd.setCursor(6,2);
    lcd.print("ON ");

    Alarma = HIGH;

}
}
```

Figura 3.5: Programación: Purga

c) Si el pulsador esta desactivado, mantiene desactivado el LED de señalización e indica la desactivación de la purga en el display LCD.

3.1.3 Control de Volumen

Se trata de uno de los procesos principales, en él el programa recogerá el estado actual de la variable V y lo comparará con los valores predeterminado que tienen VA (1000m3), VL (2000m3), y VM (3000m3), dicha comparación comenzara primero por el estado de V¡VA. Esto es así para optimizar el proceso y en el caso de que el sistema se encuentre en alarma esta se active lo antes posible (ver figs. 3.8 y 3.7).

Proceso del control de llenado:

a) La función lee el valor de la variable V.

b) Si la variable V es menor que la constante VA, se activa el LED de señalización y se activa la función de alarma.

c) Si la variable V es menor que la constante VL, se activa el LED de señalización, se activa la variable ValvulaEnt, se indica su activación en el display LCD.

d) Si la variable V es menor que la constante VM, se activa el LED de señalización, se activa la variable ValvulaEnt, se indica su activación en el display LCD.

e) Si la variable V es mayor o igual que VM se desactivan todos los LEDs de señalización, se desactiva la variable ValvulaEnt, se indica su desactivación en el display LCD.

En contraposición a otros procesos anteriores el de Rearme y el de Purga no son de necesario cumplimiento para el funcionamiento de los demás aunque sí que se realizará debido a la repetición cíclica de los procesos localizados en el void loop;.

3.1.4 Control de Llenado

Proceso del control de Llenado (ver figs. 3.10 y 3.9):

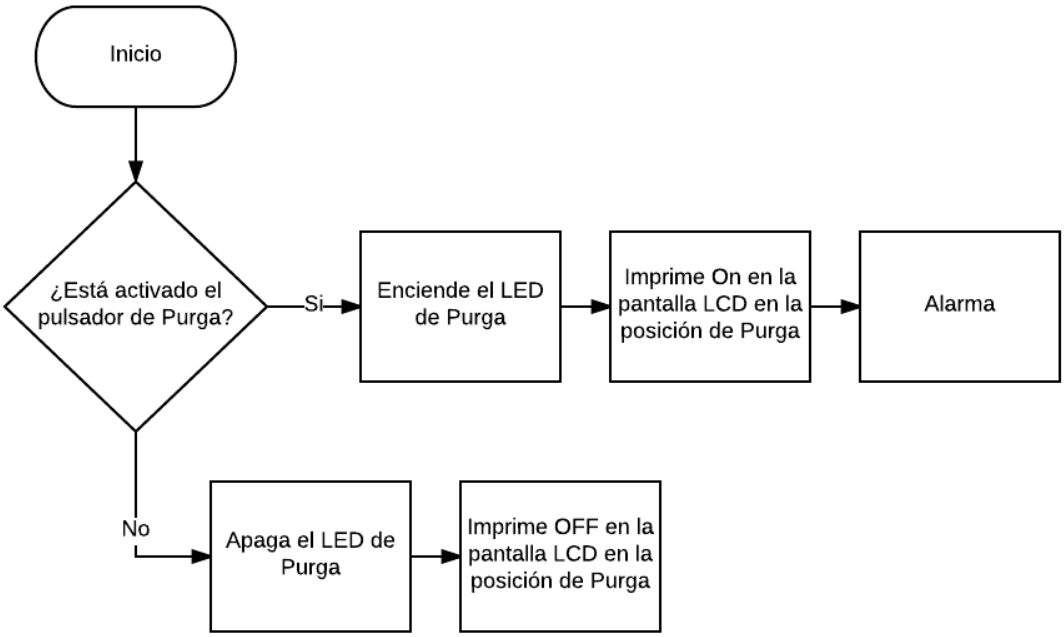

Figura 3.6: Diagrama de flujo: Purga

```
void CONTROL_DE_VOLUMEN(){
    if (V < VA) {
      digitalWrite(LedVA, HIGH);
      Alarma = HIGH; }
    else{
      if ( V < VL) {
        digitalWrite(LedVL, HIGH);
        ValvulaEnt = HIGH;
           lcd.setCursor(15,1);
           lcd.print("ON ");
        digitalWrite(LedVA, LOW); }
     else{
        if( V < VM) {
          digitalWrite(LedVM, HIGH);
          ValvulaEnt = HIGH;
           lcd.setCursor(15,1);
           lcd.print("ON "); }
     else{
       if (V >= VM) {
         digitalWrite(LedVA, LOW);
         digitalWrite(LedVL, LOW);
         digitalWrite(LedVM, LOW);
         digitalWrite(LedValvulaEnt, LOW);
         lcd.setCursor(15,1);
         lcd.print("OFF");
         ValvulaEnt = LOW;
       }}}}}
```

Figura 3.7: Programación: Control de Volumen

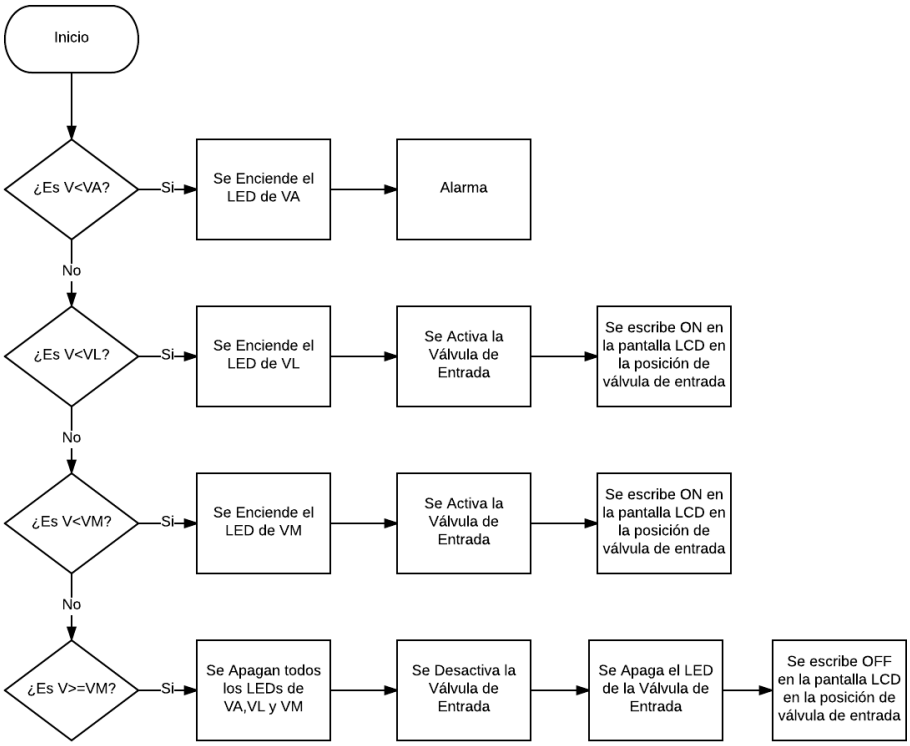

Figura 3.8: Diagrama de flujo: Control de Volumen

a) la función lee el estado de la variable ValvulaEnt.

b) Si está activada, se activa el LED de señalización, se indica su activación en el display LCD, re realiza la operación V=V+QAguaF y se indica su valor en el display LCD.

c) si está desactivado la función se mantendrá a la espera de que se active.

3.1.5 Control del Quemador

Proceso del control del quemador (ver figs. 3.12 y 3.11):

a) La función lee el estado del pulsador del quemador.

b) Si está activado, se activa el Sensor de Llama, se indica su activación en el display LCD y se activa el sistema automático.

```
void CONTROL_DE_LLENADO(){


   if ( ValvulaEnt = HIGH){
          digitalWrite(LedValvulaEnt, HIGH);
          lcd.setCursor(15,1); //LCD VENT
           lcd.print("ON ");
         V = V + QAguaF;
          lcd.setCursor(2,0);
          lcd.print(V);
        }
}
```

Figura 3.9: Programación: Control de Llenado

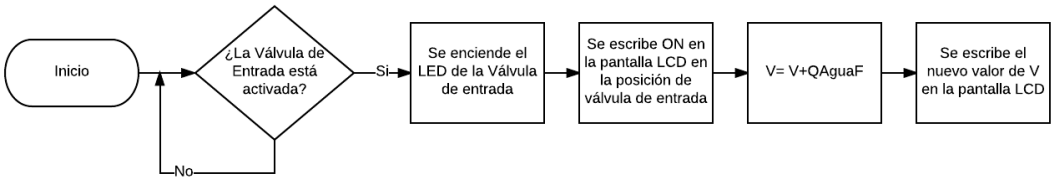

Figura 3.10: Diagrama de flujo: Control de Llenado

```
void CONTROL_DEL_QUEMADOR(){
  Quemador = digitalRead(PulsadorAQuemador);
  if(Quemador == HIGH){
    SensorLlama = HIGH;
lcd.setCursor(16,2);
lcd.print("ON ");
Automatico = HIGH;
lcd.setCursor(0,3);
lcd.print("                    ");
  }

}
```

Figura 3.11: Programación: Control del Quemador

c) Si está desactivado la función se mantendrá a la espera de que se active.

3.1.6 Barrido

Proceso del barrido (ver figs. 3.14 y 3.13):

a) La función lee el estado de la variable Automático.

b) Si la variable automático está desactivada, se activa el LED de señalización, y se lee el estado del pulsador de barrido.

c) Si el pulsador de barrido está activado se indica su activación en el display de la pantalla LCD y se desconecta el LED de señalización.

d) Si el pulsador de barrido está desactivado, se activa el LED de señalización y se queda a la espera de su activación.

3.1.7 Control de Presión

El control de presión está encargado en todo momento de controlar la variable de presión en el sistema, esto lo hará mediante la comparación con una presión de alarma prefijada en el programa y una máxima presión de demanda que tiene fijada el sistema. Ambas son completamente variables cambiando el valor en el programa.

Proceso del control de presión (ver figs. 3.16 y 3.15):

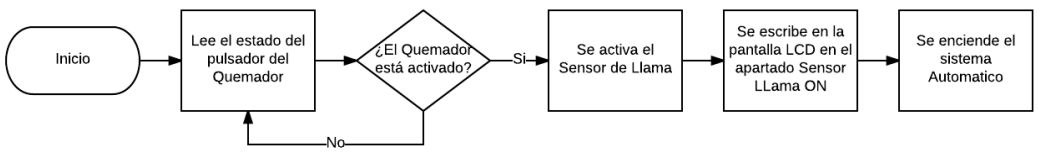

Figura 3.12: Diagrama de flujo: Control del Quemador

```
void CONTROL_DE_BARRIDO(){
  if(Automatico == LOW){
    Barrido = digitalRead(PulsadorBarrido);
    digitalWrite(LedBarrido, HIGH);
     if(Barrido == HIGH){
    lcd.setCursor(0,3);
    lcd.print("       BARRIDO      ");
      digitalWrite(LedBarrido, LOW);


  }
  else{
     digitalWrite(LedBarrido, LOW);
  }
}
}
```

Figura 3.13: Programación: Barrido

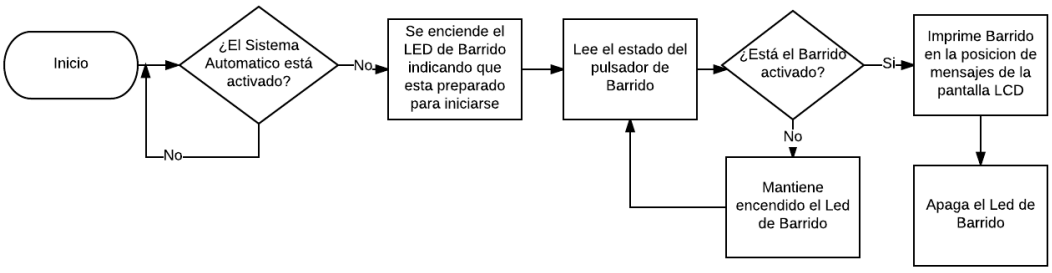

Figura 3.14: Diagrama de flujo: Barrido

```
void CONTROL_DE_PRESION(){

if(P >= PA){
   Alarma = HIGH;
}
else {
  if(P > MAXPDemanda){
    Automatico = HIGH;
    SensorLlama = LOW;
      lcd.setCursor(16,2); //LCD sensor llama
      lcd.print("OFF");
    Quemador = LOW;
  }
  else {
    if(P <= MAXPDemanda){
      if(Automatico == HIGH){
      SensorLlama = HIGH;
      lcd.setCursor(16,2); //LCD sensor llama
      lcd.print("ON ");
      Quemador = HIGH;
      }
    }
  }
}

}
```

Figura 3.15: Programación: Control de Presión

a) La función lee el estado de la variable P.

b) Si la variable P es mayor o igual que la constante PA se activa el proceso de alarma.

c) Si la variable P es mayor que la contante MAXPDemanda se desactiva el sensor de Llama, se indica su desactivación en el display LCD, se activa el sistema Automático y se desactiva el quemador.

d) Si la variable P es menor o igual a la constante MAXPDemanda la función lee el estado de la variable Automático.

e) Si la variable automático está activado, se activa el sensor de llama, se indica su activación en el display LCD y se activa el quemador.

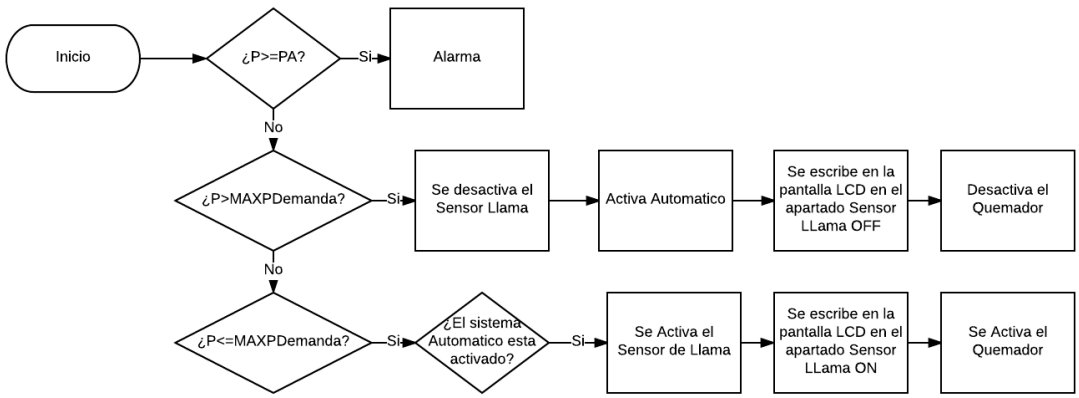

Figura 3.16: Diagrama de flujo: Control de Presión

```
void PRODUCCION_VAPOR(){

  if(SensorLlama == HIGH){
   V= V - VQuemador;
   P= P + PQuemador;

  }
    P= P - PDemanda;
   lcd.setCursor(12,0);
   lcd.print(P);
}
```

Figura 3.17: Programación: Producción de Vapor

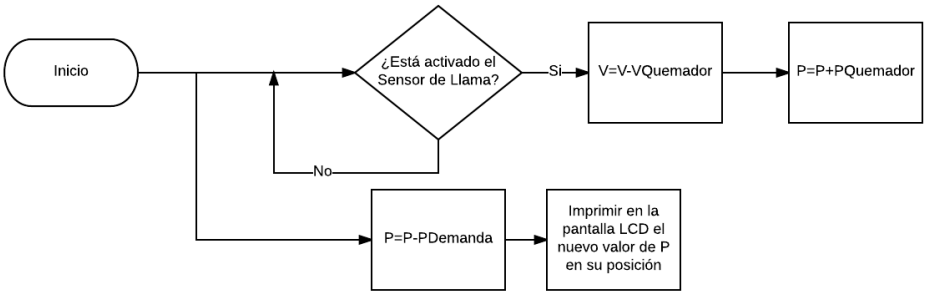

Figura 3.18: Diagrama de flujo: Producción de Vapor

3.1.8 Producción de Vapor

Este es el proceso encargado de los cambios en las variables de Presión y Volumen provenientes del funcionamiento del quemador. Proceso de producción de vapor (ver figs. 3.18 y 3.17):

a) La función realiza la operación P=P-PDemanda e indica su nuevo valor en el display de la pantalla LCD, ademas lee el estado de la variable SensorLlama.

b) Si la variable SensorLlama está activa, se realiza las operaciones V=V-Vquemador y O=P+PQuemador.

3.1.9 Control de la Presión de Demanda

Es el proceso encargado de saber mediante la variación de posición de un potenciómetro, la demanda de vapor exigida. Proceso de control de la presión de demanda (ver figs. 3.20 y 3.19):

a) La función lee el estado de la variable P.

b) Si la variable P es mayor que cero se lee la posición del PotenciometroPDemanda, se realiza la operación PresionD=ReguladorPresionD/1023 y se ejecuta el comando Pdemanda=MAXPDemanda-PresiónD.

3.1.10 Regulador Combustible/Aire

Se trata de un proceso importante para el control del quemador, pues con el simularemos la relación entre combustible y aire, la cual es de especial importancia para la creación de llama. Si dicha

```
void POTENCIOMETRO_PDEMANDA(){
if(P > 0){
ReguladorPresionD = analogRead(PotenciometroPDemanda);
float PresionD = float(ReguladorPresionD)/1023;
PDemanda= MAXPDemanda * PresionD;
}
}
```

Figura 3.19: Programación: Control de la Presión de Demanda

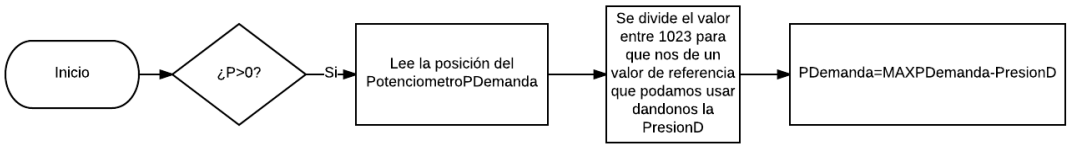

Figura 3.20: Diagrama de flujo: Control de la Presión de Demanda

relación no se encuentra en los parámetros adecuados se producirá un Fallo de Llama que puede hacer en cualquier momento que el Quemador se desactive. Proceso del Regulador Combustible/Aire (ver figs. 3.22 y 3.21):

a) La función lee la posición del PotenciometroCA, se realiza la siguiente operación CA=ReguladorCA* $\frac{100}{1023}$.

b) Si la variable CA es menor que 30 se indicara "Fallo de Llama" en el display LCD, se desactiva las variables: quemador, sensor de llama, automático, barrido y CAQuemador e indicándose el desactivado del sensor de llama en el display LCD.

c) Si la variable CA es mayor que 70 se indicara "Fallo de Llama" en el display LCD, se desactiva las variables: quemador, sensor de llama, automático, barrido y CAQuemador e indicándose el desactivado del sensor de llama en el display LCD.

d) En caso de que no se cumplan los dos casos anteriores se activa la variable CAQuemador.

3.1.11 Potenciometro Caudal de Agua de Alimentación

Es el proceso que se encarga de regular mediante un potenciómetro el caudal de agua de alimentación entrante a través de la válvula de entrada. Aumentando de esta manera el Volumen de agua de alimentación en la caldera. Proceso Potenciómetro caudal de agua de alimentación (ver figs. 3.24 y 3.23):

a) La función lee la posición del PotenciometroQAgua, realiza la operación ReguladorQAgua/1023 y la operación QaguaF=MAXQagua*Qagua.

b) Si la variable ReguladorQAgua es menor o igual que cero se desactiva la válvula de entrada e indica su desactivado en el display de la pantalla LCD.

c) Si la variable ReguladorQagua es mayor que cero se activa la válvula de entrada e indica su activado en el display LCD.

3.1.12 Alarma

Es un proceso ejecutado antes de la realización de los demás procesos con el fin de que salte inmediatamente a este en caso de cumplirse su condición de Alarma Activada. Al contrario que otros procesos, sin contar el de purga, se trata de uno que no podremos invertir ya que se ha posibilitado la realización de un trabajo manual para su mejora. Con la ejecución de este proceso se llevará a cabo la desactivación del resto de procesos, desde el control de llenado al quemador cada una de las variables operacionales de estos será desactivada para imposibilitar, en la medida de posible, un daño a la caldera u operarios de ésta. Proceso de alarma (ver figs. 3.26 y 3.25):

a) La función lee el estado de la variable Alarma.

```
void REGULADOR_COMBUSTIBLE_AIRE(){
ReguladorCA = analogRead(PotenciometroCA);
float CA = float (ReguladorCA)*100/1023;
if( CA < 30 ){
   lcd.setCursor(0,3);
   lcd.print("   FALLO DE LLAMA  ");
       Quemador = LOW;
    SensorLlama = LOW;
 lcd.setCursor(16,2); //LCD sensor llama
 lcd.print("OFF");
    Automatico = LOW;
    CAQuemador = LOW;
    Barrido = LOW;}
else{
 if( CA > 70 ){
   lcd.setCursor(0,3);
   lcd.print("   FALLO DE LLAMA  ");
       Quemador = LOW;
    SensorLlama = LOW;
 lcd.setCursor(16,2); //LCD sensor llama
 lcd.print("OFF");
    Automatico = LOW;
    CAQuemador = LOW;
    Barrido = LOW; }
CAQuemador = HIGH;
}}
```

Figura 3.21: Programación: Regulador Combustible/Aire

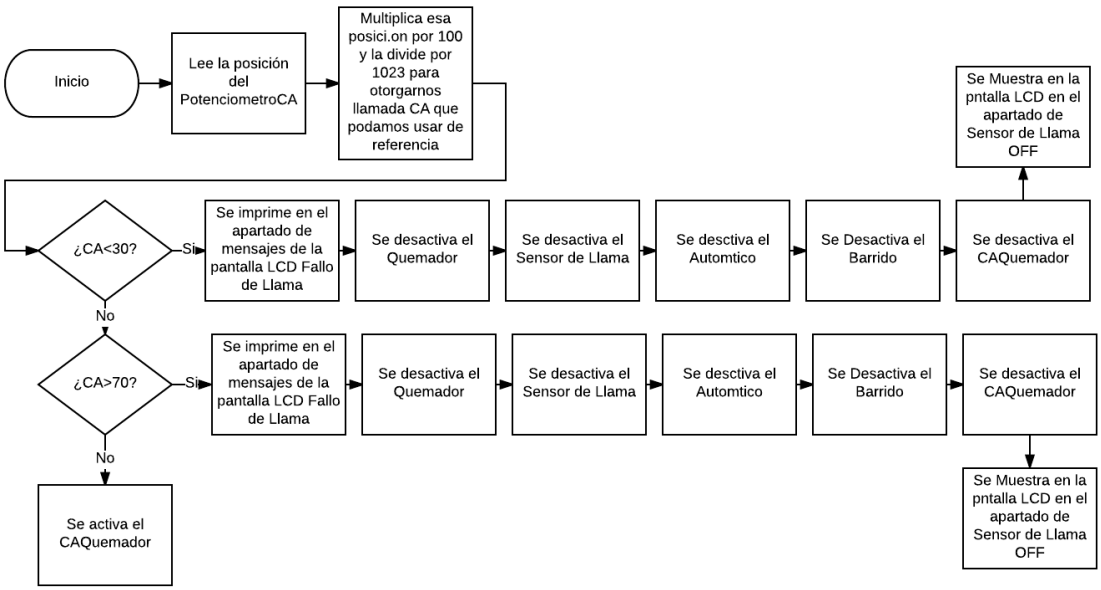

Figura 3.22: Diagrama de flujo: Regulador Combustible/Aire

```
void POTENCIOMETRO_QAGUA(){

ReguladorQAgua = analogRead(PotenciometroQagua);
float QAgua = float(ReguladorQAgua)/1023;
QAguaF= MAXQagua * QAgua;
 if(ReguladorQAgua <=0){
   ValvulaEnt = LOW;
    lcd.setCursor(15,1); //LCD VALVULA DE ENTRADA
    lcd.print("OFF");
 }
 if(ReguladorQAgua > 0){
  ValvulaEnt = HIGH;
    lcd.setCursor(15,1); //LCD VALVULA DE ENTRADA
    lcd.print("ON ");
 }
}
```

Figura 3.23: Programación: Potenciómetro Caudal de Agua de Alimentación

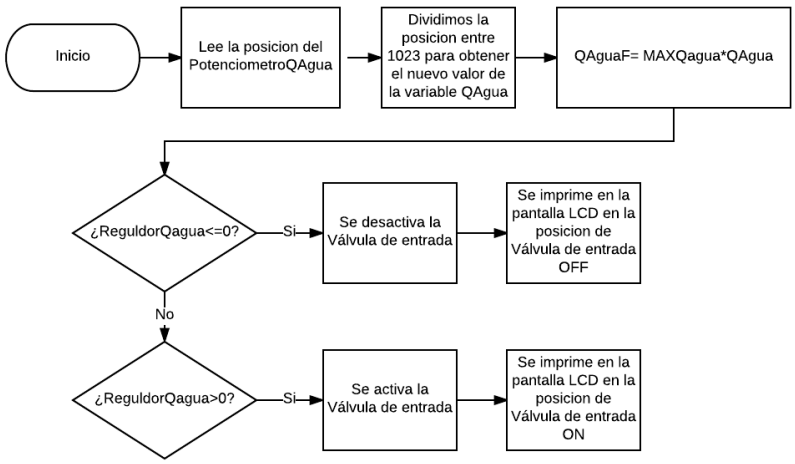

Figura 3.24: Diagrama de flujo: Potenciómetro Caudal de Agua de Alimentación

```
void ALARMA(){

 if (Alarma == HIGH)
 lcd.setCursor(0,3);
 lcd.print("       ALARMA       ");
    digitalWrite(LedVA, LOW);
    digitalWrite(LedVL, LOW);
    digitalWrite(LedVM, LOW);
    RearmeSistema = LOW;
    lcd.setCursor(5,1); //LCD REARME SISTEMA
    lcd.print("OFF");
    ValvulaEnt = LOW;
    lcd.setCursor(15,1); //LCD VALVULA DE ENTRADA
    lcd.print("OFF");
    Quemador= LOW;
    SensorLlama=LOW;
 lcd.setCursor(16,2); //LCD sensor llama
 lcd.print("OFF");
    Automatico=LOW;
    Barrido=LOW;
     digitalWrite(LedAlarma, HIGH);
    delay(1000);
    digitalWrite(LedAlarma, LOW);
    delay(1000);
}
```

Figura 3.25: Programación: Alarma

b) Si la variable está activada, se indica "ALARMA" en el display LCD, se desconectan los LEDs de señalización de VA,VL,y VM, y se desactivan: Sensor de Llama, Quemador, Válvula de entrada, Rearme de sistema, Automático, Barrido, indicándose aquellos que dispongan su desconexión en el display LCD.

Cabe señalar que tras la realización de estos procesos se realizará un delay (1000) ; con la intención de que el programa se ejecute cada segundo produciéndose así el cambio de variables de P y V cuya evolución será por segundo (Atm/s, y m^3/s).

3.2 Descripción del Simulador

El simulador consta de un hardware Arduino Mega 2560, interconectado al ordenador mediante un USB 2.0 - 1.0. Además dispone de un display LCD de 4x20 con el que se visualizará la gran mayoría de los cambios en las variables (V, P, RearmeSistema, Purga, ValvulaEnt, SensorLlama), así como

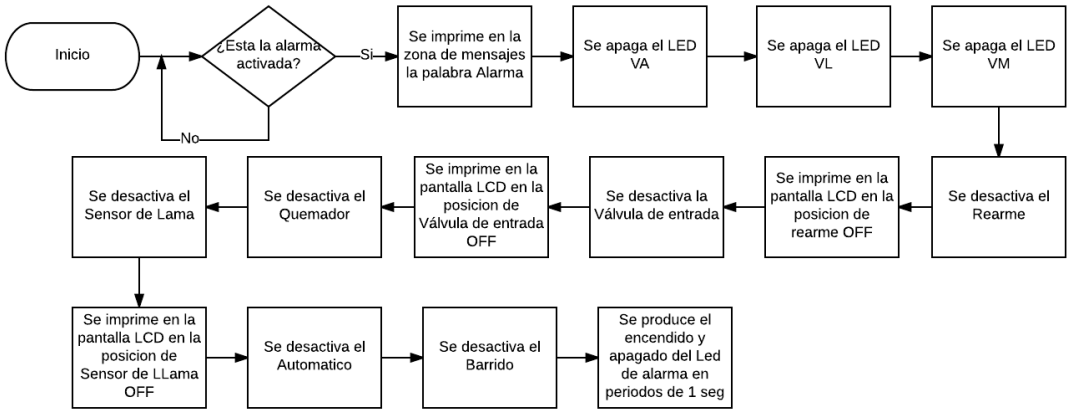

Figura 3.26: Diagrama de flujo: Alarma

Tabla 3.4: Variables Fijadas a Pines

```
const int PulsadorRearme = 22
const int LedRearme = 2
const int PulsadorPurga = 24
const int LedPurga = 4
const int LedAlarma = 5
const int LedVA = 6
const int LedVL = 7
const int LedVM = 8
const int LedValvulaEnt = 9
const int LedBarrido = 10
const int PulsadorBarrido = 26
const int LedLlama = 12
const int PulsadorAQuemador = 28
const int PotenciometroPDemanda = A1
const int PotenciometroQagua = A2
const int PotenciometroCA = A3
```

los mensajes (Alarma, Fallo de Llama, Barrido) , quedando las demás señalizadas de forma luminosa mediante LEDs (VA,VL,VM, etc.), cabe destacar que ciertas variables también disponen de diodos LEDs para su señalización, siendo en estos casos un sistema redundante.

Para aquellas variables que necesitan una señal de entrada (RearmeSistema, Purga, Barrido, Quemador), se ha conectado un pulsador entre uno de los pines proporcionados por Arduino que nos otorga 5v y el pin prefijado en cada una de las variables para recibir dicha señal de entrada.

Además contamos con potenciómetros para variar las señales de entrada de: regulación combustible/aire, regulación del caudal de agua de alimentación y la presión de demanda. Con lo que será posible realizar una regulación de dichos sistemas manualmente. Dichos potenciómetros están conectados a un pin de salida de 5V y un pin de entrada GND proporcionada por el hardware de Arduino, la salida de regulación del potenciómetro se conecta al pin de entrada correspondiente a cada variable, (ver fig. 3.27).

3.3 Conexionado

A la hora de realizar el conexionado debemos tener en cuenta que cada uno de los puertos o pines de entrada y salida con los que cuenta Arduino tiene una numeración, y en el caso de algunos en especial una función en específico, como por ejemplo los puertos encargados de otorgar un voltaje continuo de 5v, los puertos para la conexión a tierra del sistema (GND), etc. Es importante no equivocarse en la elección de estos para el conexionado.

Para la elección de pines en Arduino, se prefijaron anteriormente con una variable mediante la programación en el software, (ver tabla 3.4).

Cada una de estas variables está vinculada con un pin en específico, destacando aquellas con la numeración de A1, A2, A3, las cuales son entradas analógicas, que si bien pueden ser utilizados como digitales. Los que son digitales únicamente, no pueden utilizarse como analógicos. Además de otorgarle un Pin a cada variable, se indica si ese pin es de entrada o salida, para el reconocimiento por parte del hardware de Arduino pues las entradas están preparadas para recibir una señal de voltaje y las salidas proporcionara un voltaje, (ver tabla 3.5).

Al contrario que la especificación del pin para cada una de las variables, la determinación de entradas o salidas debe disponerse en el void setup; para su correcto funcionamiento, realizándose la lectura de éstos sólo al inicio de la ejecución de la simulación. Para el conexionado de la pantalla LCD, se usa el BUS I2C a través de los pines SDA y SCL, y otros dos de alimentación (5V y GND).

A la hora del conexionado de LEDs, Pulsadores y Potenciómetros hemos utilizado una protoboard de fácil conexionado.

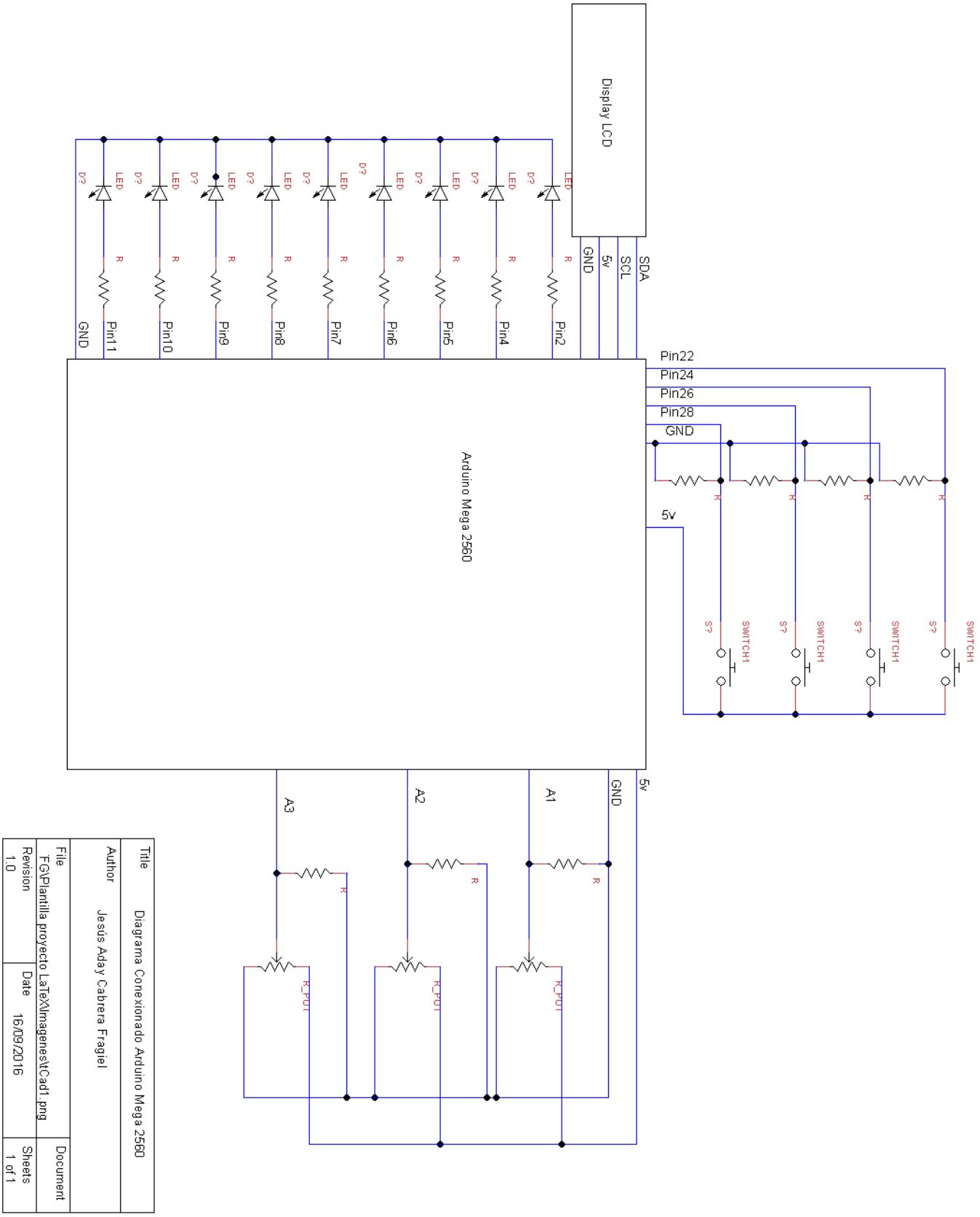

Figura 3.27: Esquema del Simulador

Tabla 3.5: Constantes INPUT/OUTPUT

pinMode(PulsadorRearme, INPUT)
pinMode(LedRearme, OUTPUT)
pinMode(PulsadorPurga, INPUT)
pinMode(LedPurga, OUTPUT)
pinMode(LedAlarma, OUTPUT)
pinMode(LedVA, OUTPUT)
pinMode(LedVL, OUTPUT)
pinMode(LedVM, OUTPUT)
pinMode(LedValvulaEnt, OUTPUT)
pinMode(LedBarrido, OUTPUT)
pinMode(PulsadorBarrido, INPUT)
pinMode(LedLlama, OUTPUT)
pinMode(PulsadorAQuemador, INPUT)
pinMode(PotenciometroPDemanda, INPUT)
pinMode(PotenciometroQagua, INPUT)

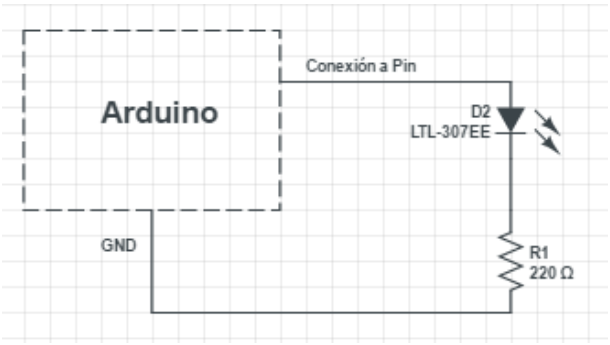

Figura 3.28: Conexionado: Conexión LEDs

En el caso de los LEDs hay que tener en cuenta que el voltaje proporcionado puede llegar a dañarlos así que entre su conexión al pin de la variable que le corresponda al LED, es necesario la colocación de una resistencia. En este caso se ha conectado en serie una resistencia e 220ohms que será suficiente para evitar un posible fundido del LED. Tras el LED este debe conectarse a GND (ver fig. 3.28).

En lo referente a los pulsadores y, teniendo en cuenta el voltaje de 5v, se ha conectado en serie una resistencia de 10k ohms debido a que se creaba una corriente entre las puntas del pulsador que falseaba la señal, esta resistencia se encuentra en serie entre el pulsador y el GND como en el caso del conexionado de los LEDs (ver fig. 3.29).

Para el conexionado de los potenciómetros debemos tener en cuenta que estos cuentan con 3 conexiones 2 que pueden ser o la entrada de voltaje o la tierra indistintamente variando únicamente hacia donde es el + y el – a la hora de realizar el movimiento del potenciómetro y otra que recibe

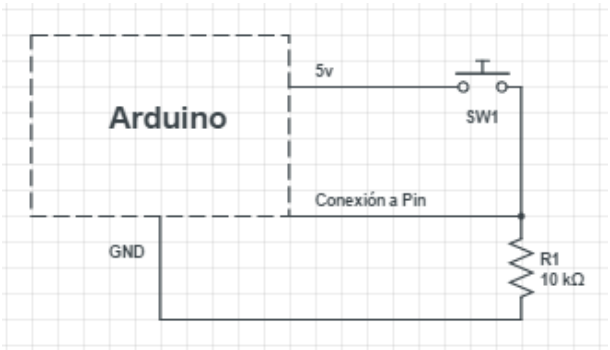

Figura 3.29: Conexionado: Conexión Pulsador

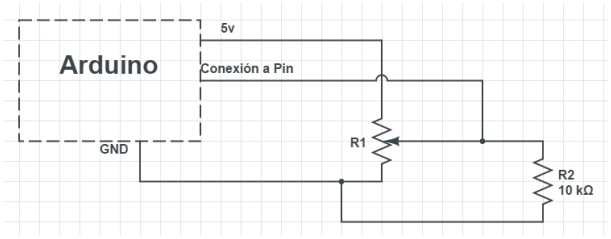

Figura 3.30: Conexionado: Conexión Potenciómetro

la variación de voltaje. Se debe destacar que en la que se produce la variación de voltaje para una correcta lectura, sobre todo en el caso de 0, es necesario conectarle una resistencia en serie conectado a GND. Dicha resistencia al igual que en el caso de los pulsadores es de 10k ohms 5 (ver fig. 3.30).

4 Posibles Mejoras

4.1 Utilización de valores reales

En este diseño del prototipo no se han tenido en cuenta valores reales sino se ha procedido con la búsqueda de una correcta simulación en el funcionamiento de los algoritmos, por lo que queda sujeto a posibles modificaciones posteriores. En este sentido, la inclusión de datos reales que logren su funcionamiento conllevará un mayor realismo. Todo ello puede ser mediante la obtención de gráficas de rendimiento de la caldera que nos proporcionen datos sobre la evaporación en función del rendimiento, consumo de combustible, etc. Además se podrían tener en cuenta las pérdidas de carga en la línea debido al aislamiento con el que se ha simulado la instalación.

4.2 Añadir un sistema de tiempo para el barrido

La realización del proceso de barrido no es algo que se pueda hacer una vez y dejarlo pasar hasta la ejecución de la maniobra del quemador, sino que cuenta con un tiempo limitado en el que es efectivo. Por ello, podría ser efecto de mejora la inclusión de un contador que contase el tiempo que pasa desde que se pulsa el pulsador de barrido y provoque la desactivación de este en caso de no activarse en el tiempo estipulado.

4.3 Cálculo de porcentaje de gases de combustión

Mediante el potenciómetro Combustible/Aire, regulamos el porcentaje de estos a la hora de la combustión, pero en una caldera también debemos tener en cuenta los porcentajes de CO^2, CO, etc; que se producen ya que hay limitaciones legales las cuales han de cumplirse.

4.4 Simulación de procesos termodinámicos

La simulación puede mejorarse añadiendo procesos de cálculo que tengan en cuenta el proceso de calentamiento del agua real, siguiendo los diagramas de Mollier y teniendo en cuenta los sucesivos estados de equilibrio según aumenta la temperatura.

5 Conclusiones

Una vez realizado el desarrollo del trabajo se llega a las siguientes conclusiones(ver figs. 5.1 y 5.2):

Los actuales modelos de actividades practicas son insuficientes para la impartición y obtención de conocimientos por parte de profesorado y alumnado.

Tras la realización de los algoritmos se han adquirido conocimientos sobre programación, visualización de errores, comprensión de esquemas electrónicos, etc.

La creación del prototipo ha repercutido en el aumento de la comprensión del manejo de elementos electrónicos (LEDs, Potenciómetros, etc.).

Además de proporcionar posibilidades infinitas para crear nuevos sistemas, se muestra como una forma de profundizar sobre aquellos cuyos simuladores no son detallados.

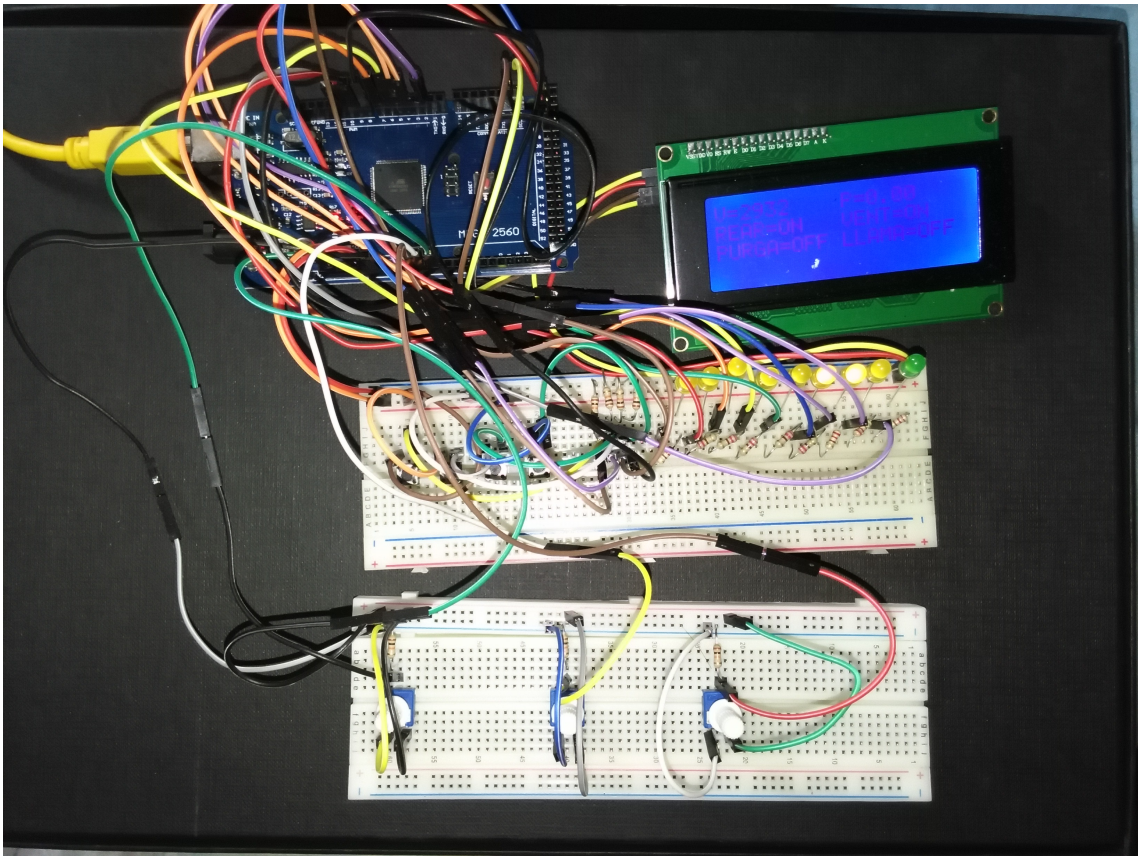

Figura 5.1: Simulación, Ejemplo 1

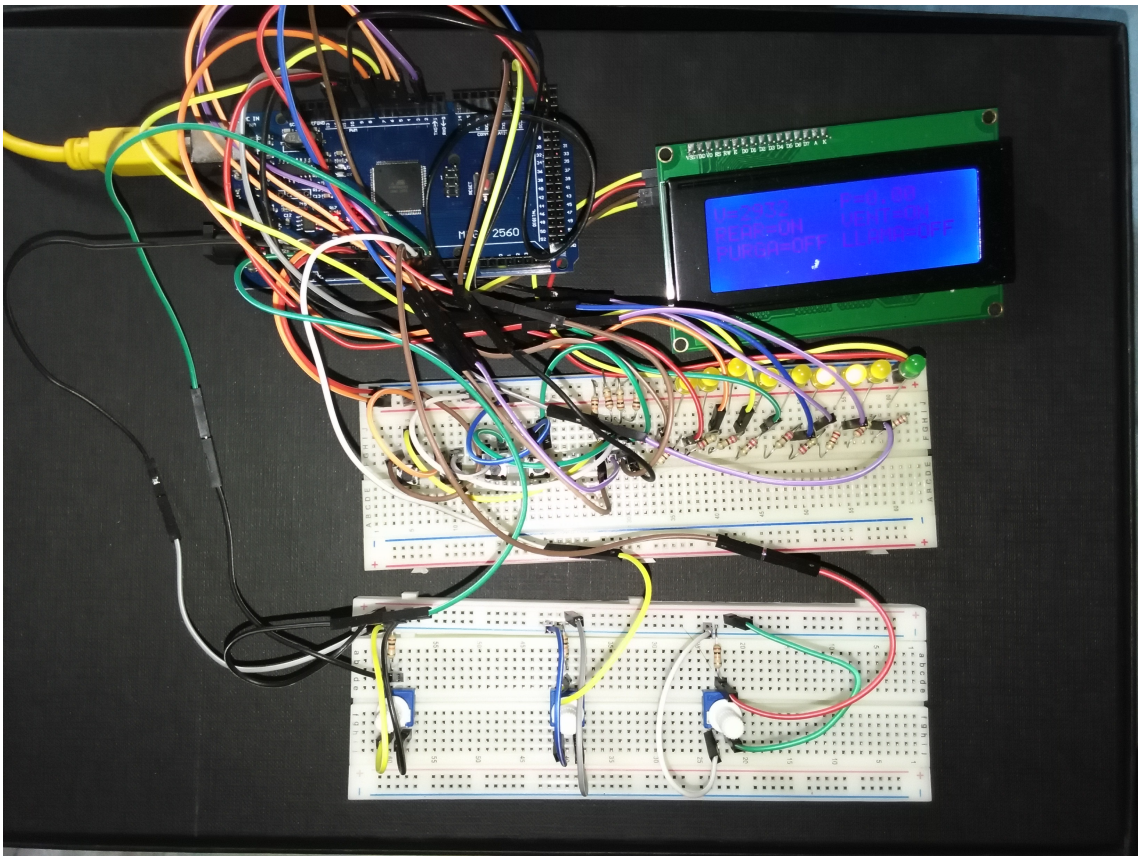

Figura 5.2: Simulación Ejemplo 2

Anexos

A Primer anexo

A.1 Lista de Materiales

Para la creación del prototipo ha sido necesaria la utilización de ciertos materiales que expongo a continuación.Ver tabla 3.1.

Tabla A.1: Descripción de Materiales

1	Arduino Mega 2560
1	Cable USB 2.0-1.0
4	Pulsadores
3	Potenciometros
9	LEDs
1	Pantalla LCD de 20x4
2	Protoboard
57	Cables de colores con pines para Protoboard
7	Resistencias de 10k Ohms (Marron, Negro, Naranja, Oro)
9	Resistencias de 220 Ohms (Rojo, Rojo, Marron, Oro)

Bibliografía

[1] "Arduino". [Página web], 2016 [consultado 16 de septiembre de 2016]. URL: https://www.arduino.cc/.

[2] José Manuel Ruiz Gutiérrez , *Manual de Programación Arduino*. 2016.

[3] German Tojeiro Calaza , *Taller de Arduino un enfoque práctico para estudiantes*. 1º, MARCOM BO, S.A., 2014.

[4] Harald Kluken, *Manual del Simulador MC-90*. KONGSBERG MARITIME AS, 2006.