

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



**“DISEÑO Y DESARROLLO DE UN
CONTROLADOR PARA QUEMADOR
INDUSTRIAL DE MEDIO POROSO”**

FRANCISCO ALEJANDRO LARA MÁRQUEZ

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO
CIVIL ELECTRÓNICO CON MENCIÓN CONTROL E
INSTRUMENTACIÓN**

PROFESOR GUIA:

MANUEL OLIVARES

PROFESIONAL CORREFERENTE:

NICOLÁS BECKER

NOVIEMBRE - 2019

Agradecimientos

Quiero agradecer a todas las personas que han tenido participación en mi proceso formativo como profesional.

Agradezco a toda mi familia, en especial a mis padres, Luciana y Alejandrino, quienes siempre me han entregado su afecto, apoyo y consejos en cada decisión que he tomado en mi vida.

A mi abuelita Olga, siempre preocupada de que no me falte nada, que no pase frío y que pueda estudiar tranquilo.

A mi tío Roberto y tía Isabel, quienes me acogieron en su hogar y han apoyado incondicionalmente durante estos últimos años de carrera.

A mi prima Carolina, a Felipe y Elisa, que siempre han estado ahí cuando los he necesitado.

A mis hermanos, Vicente e Ignacio, y a mis amigos de toda la vida, David y Ariel, todos del grupo Sus PES, por su constante apoyo, infaltables cervezas y ping pong.

A Fernanda que me ayudó a ordenar mis ideas al comenzar a escribir este trabajo y comprendió todas las veces en que me quedaba toda la noche en la universidad haciendo trabajos, tareas y estudiando.

A Innovación y Robótica Estudiantil por permitirme utilizar su espacio durante el desarrollo de este trabajo y con quienes pasé gran parte de mi vida universitaria. Gracias ustedes pude aprender mucho más de lo que habría aprendido si solo hubiera asistido a clases.

A Esteban, con quien trabajé desde el principio en el desarrollo de este trabajo que siempre pudimos sacar adelante.

A los chicos de QUEMPIN, quienes me brindaron la oportunidad de realizar este trabajo en su empresa, en especial a Nicolás y Enzo, quienes siempre tuvieron la buena disposición a ayudarme en todo lo que necesité para realizar esta memoria y también al profesor Manuel, quien siempre estuvo pendiente del proceso de desarrollo y me guió en cada etapa para realizarla correctamente.

El apoyo de cada uno de ustedes ha sido fundamental para enfrentar cada etapa de mi vida, en especial ésta que culmina en este momento.

Resumen

El presente trabajo consiste en el desarrollo e implementación de un controlador para un quemador industrial que utiliza un medio poroso inerte (MPI). Se utiliza la instrumentación implementada previamente en la memoria de Esteban Arriagada: “Instrumentación para un quemador de medios porosos inertes (2019)”, la cual adapta la mayoría de los sensores incorporados en un quemador con el fin de ser leídos por un controlador, y también se desarrolla la instrumentación pendiente de diseño. El aporte principal de esta memoria consiste en diseñar circuitos de acondicionamiento para integrar la instrumentación de un quemador de medio poroso inerte a un microcontrolador STM32F030F4, y desarrollar un programa con el fin de automatizar la secuencia de encendido del quemador cumpliendo la norma NCh3111: “Quemadores de aire forzado que utilizan combustible gaseoso”.

Palabras claves: Quemador de medios porosos inertes, automatización industrial, microcontrolador STM32F030F4.

Abstract

The present work consists in the development and implementation of a controller for an industrial burner that uses an porous inert media (PIM). The instrumentation previously implemented in Esteban Arriagada's memory: "Instrumentación para un quemador de medios porosos inertes (2019)" is used, which adapts most of the sensors incorporated in a burner in order to be read by a controller, and the pending design instrumentation is also developed. The main contribution of this work is to design conditioning circuits to integrate the instrumentation of an porous inert media burner into a STM32F030F4 microcontroller, and to develop a program in order to automate the burner ignition sequence complying with standard NCh3111 : "Quemadores de aire forzado que utilizan combustible gaseoso".

Keywords: Porous inert media burner, industrial automation, STM32F030F4 microcontroller.

Glosario

- IoT: “Internet of Things” o “Internet de las cosas” en español, corresponde a la interconexión digital de objetos cotidianos con internet.
- MPI: Medio poroso inerte.
- PWM: “Pulse Width Modulation” o “Modulación por ancho de pulso”.
- NOx: Término genérico para referirse a los óxidos de nitrógeno, conocidos por la polución del aire, tales como el óxido de nitrógeno (NO) y dióxido de nitrógeno (NO_2). Estos gases contribuyen a la formación de smog y lluvia ácida.
- PLC: Controlador lógico programable.
- Razón aire combustible: Proporción entre el volumen de aire y combustible. Este valor es relevante para obtener la máxima potencia en la combustión.

Índice general

<i>1.. Introducción</i>	4
1.1. Problema a resolver y objetivos	4
1.2. Quemadores de QUEMPIN	6
1.2.1. Llama piloto	6
1.2.2. Quemador con MPI	7
1.3. Razón Aire/Combustible	8
<i>2.. Estado del arte</i>	9
2.1. Controladores	9
2.2. Diseño de Reactores de MPI	10
2.3. Instrumentación disponible	10
2.3.1. Detector de llama piloto	11
2.3.2. Sensor de caudal de aire	11
2.3.3. Lectura de termocuplas	11
2.3.4. Operación de Servoposicionador en válvula monoblock	12
2.3.5. Presostato de gas	12
2.3.6. IoT con Raspberry Pi	12
<i>3.. Requerimientos y alternativas de solución</i>	13
3.1. Requerimientos funcionales	13
3.2. Requerimientos de diseño de hardware	14
3.3. Alternativas de solución	15
3.3.1. Criterios de evaluación	15

3.3.2.	Microcontroladores a evaluar	16
3.3.3.	Microcontrolador seleccionado	18
4..	<i>Solución Propuesta</i>	20
4.1.	Arquitectura de Hardware	20
4.1.1.	Botones	21
4.1.2.	Pantalla	21
4.1.3.	Soplador	21
4.1.4.	Activación de actuadores	22
4.2.	Desarrollo de software	22
4.2.1.	Control de velocidad del soplador	23
4.2.2.	Encendido de llama piloto	23
4.2.3.	Automatización del quemador	24
4.2.4.	Pantalla LCD	26
5..	<i>Desarrollo e Implementación</i>	27
5.1.	Circuitos de acondicionamiento	27
5.1.1.	Lectura de activación de presostato de aire	27
5.1.2.	Actuación PWM para Soplador	29
5.1.3.	Medición RPM de Soplador	30
5.1.4.	Activación de actuadores	30
5.1.5.	Lectura de otros sensores	31
5.2.	Configuración de pines en el controlador	31
5.3.	Alimentación del sistema	33
5.4.	Control PI de la velocidad del soplador	34
5.5.	Graficet de encendido de llama piloto	36
5.6.	Graficet de encendido del sistema completo	37
6..	<i>Resultados</i>	40
6.1.	Control de velocidad del soplador	40

6.2.	Lectura de activación de presostato de aire	41
6.3.	Encendido de llama piloto	41
6.4.	Encendido del sistema completo	43
7..	<i>Conclusiones</i>	45
8..	<i>Anexos</i>	49
8.1.	Imágenes	49
8.2.	Códigos utilizados	52
8.2.1.	Control_RPM_PI.ino: “Prueba de control de soplador con PI y RPM”	52
8.2.2.	Presostato_PI.ino: “Prueba de presostato de aire ante activación de soplador”	55
8.2.3.	Prueba_sensor_llama.ino: “Prueba de sensor de llama piloto” .	59
8.2.4.	final.ino: ”Prueba de automatización final del sistema	63

1. INTRODUCCIÓN

1.1. Problema a resolver y objetivos

Chile es el mayor consumidor de pan en Latinoamérica y el segundo a nivel mundial después de Alemania, con un consumo que va desde los 86 a los 90 kilos anuales por persona. El consumo de pan en nuestro país es tan relevante, que el pan corriente representa cerca del 23 % de la canasta básica familiar utilizada por el INE para calcular el IPC.

En nuestro país existen más de 12.000 panaderías y el 70 % de éstas utiliza leña como combustible principal, lo cual corresponde a una alternativa de bajo costo que permite mantener niveles productivos que den abasto a la alta demanda nacional. Sin embargo, este sistema provoca altos niveles de contaminación atmosférica y puede afectar la salud a largo plazo de las personas que trabajan con estos hornos. Al mismo tiempo, al utilizar esta fuente de energía se arriesga a tener multas de hasta 1.000 UTM y cierres en días de alerta ambiental.

La empresa QUEMPIN, spinoff de la Universidad Técnica Federico Santa María, plantea la utilización de quemadores de medios porosos inertes en hornos de panaderías chilenas, con el fin de reemplazar los hornos que usan la combustión de leña. Un horno convencional a leña utiliza en promedio un 5 % de la energía para la cocción del pan. Al utilizar quemadores de gas radiativos en la cámara de combustión de hornos tradicionales, se logra llegar a un ahorro de hasta 20 % en consumo de gas, en comparación con los quemadores a gas convencionales.[1]

Un Medio Poroso Inerte o MPI corresponde a un sólido con gran conductividad térmica a altas temperaturas que mejora las propiedades de transferencia de calor (con-

ducción, convección y radiación), permitiendo un libre desplazamiento del frente de la mezcla de gases. Estos cuentan con una alta eficiencia en la transferencia de calor por radiación en el momento de atrapar la llama que se genera en la superficie del quemador, permitiendo mantener bajas temperaturas de combustión y disminuir la emisión de gases del tipo NOx, los cuales son un subproducto de la quema de combustible para la generación de energía, de alta toxicidad para el ambiente.

Parte del proceso de encendido de los hornos de QUEMPIN se realiza de forma manual junto a un controlador automático estándar para quemadores intermitentes de gas. Sin embargo, este controlador no ofrece un monitoreo ni control de variables que permita ajustar manualmente la potencia de quemado, quedando la posibilidad de estar operando los quemadores en tasas de quemado que no son las óptimas. Además, al activar manualmente las válvulas de gas se corre un riesgo de accidentes por errores humanos.

Por lo tanto, el objetivo principal de este trabajo corresponde a desarrollar e implementar un controlador que cumpla con la automatización de la secuencia lógica de operación, considerando los sensores y actuadores ya presentes en los hornos que comercializa la empresa y, al mismo tiempo, cumplir con las normas industriales ya establecidas.

Este desarrollo utiliza la instrumentación implementada en la memoria[2] realizada para esta empresa en el programa de memorias multidisciplinarias de la UTFSM, en donde se adaptaron los sensores y actuadores necesarios para poder utilizar un controlador en el proceso de encendido y operación regular. En ese mismo trabajo, se implementó una interfaz web con un arreglo IoT, la cual muestra variables relevantes en la operación del quemador. Dado esto último, el controlador desarrollado en esta memoria se debe comunicar con este sistema.

1.2. Quemadores de QUEMPIN

Los componentes actuales de un quemador de la empresa QUEMPIN se pueden dividir en dos subsistemas: Uno que posee una llama piloto y otro que posee el quemador con MPI, como se observa en la Figura 1.1.

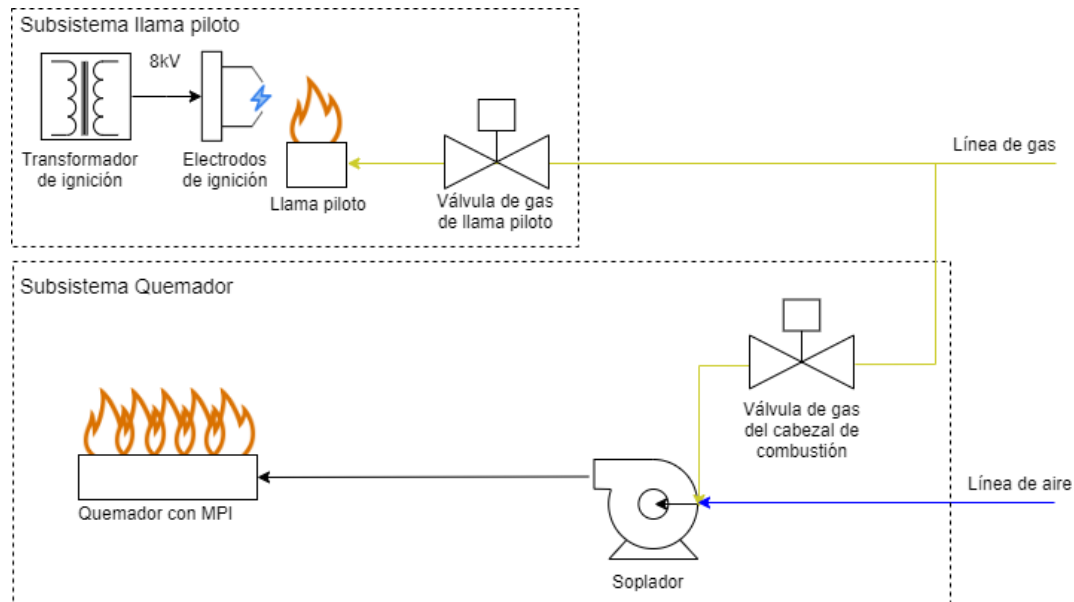


Fig. 1.1: Subsistemas que componen un quemador de la empresa QUEMPIN

1.2.1. Llama piloto

El subsistema de la llama piloto se encarga de encender el quemador. Este subsistema posee los siguientes componentes:

- Llama piloto: Utilizada para encender el quemador con MPI.
- Válvula de gas para llama piloto: Es una válvula proporcional encargada de dar el paso de gas hacia la llama piloto. La cantidad de gas que permite pasar esta válvula, es ajustada manualmente al instalar el quemador. Posteriormente, al energizar y des-energizar la válvula, se activa y desactiva el paso de gas, respectivamente.

- Transformador de ignición: Transformador que al ser conectado a 220V alterno, entrega un voltaje cercano a los 8kV.
- Electrodo de ignición: Estos electrodos son conectados al transformador de ignición, provocando un arco eléctrico al energizarlo.
- Sensor de llama: Corresponde a una sonda de ionización que detecta si la llama piloto se ha encendido.

1.2.2. Quemador con MPI

Los componentes del subsistema quemador corresponden a:

- Quemador con MPI: Encargado de irradiar calor en el horno.
- Soplador: Impulsa la premezcla hacia el quemador, la cual consiste en aire y gas. El soplador posee una entrada de aire y otra de gas que proviene de la válvula monobloque de gas del cabezal de combustión. Este soplador es de velocidad variable, ajustable mediante un sistema de control recomendado por el fabricante utilizando una señal PWM.
- Válvula de Gas del cabezal de combustión: Válvula Monoblock que permite el paso de gas hacia el soplador. Esta válvula cuenta con dos válvulas proporcionales de regulación de presión y caudal, además de un filtro y presostato de gas.
- Termocuplas: El quemador posee tres termocuplas ubicadas en distintos puntos, en donde es posible medir la temperatura a la que éste opera.
- Presostato de aire: Sensor encargado de comprobar que está ingresando aire al soplador, el cual corresponde a un switch que se activa al pasar un umbral de presión determinado en su entrada.

1.3. Razón Aire/Combustible

La relación Aire/Combustible corresponde al cociente entre el caudal de aire y el caudal de combustible que ingresan al quemador. Este factor se regula de forma manual, modificando la cantidad de caudales de aire y de gas que ingresan al soplador cuando se realiza el mezclado de éstos.

Para regular la razón aire/combustible, se realiza un encendido manual del quemador, el cual consiste en encender la llama piloto con el soplador en una velocidad baja. Posterior a esto, se activa la válvula de gas del cabezal de combustión permitiendo el encendido del quemador con MPI. El encendido óptimo se logra al llegar a la razón aire/combustible indicada, la cual es encontrada regulando la velocidad del soplador y manualmente la apertura de la válvula de gas de monoblock.

Una vez que la llama se estabiliza en el MPI, se eleva la velocidad del soplador en 200 RPM, llegando a la operación normal del quemador.

De esta forma, se obtiene una apertura fijada para la válvula de gas del cabezal de combustión, una velocidad de encendido del soplador y una velocidad de operación del soplador, logrando la razón de aire/combustible que permite una buena operación del quemador a través de la variación del caudal de aire por medio de la regulación de velocidad del soplador.

2. ESTADO DEL ARTE

2.1. Controladores

Uso de controladores en aplicaciones de regulación de temperatura

Un trabajo de memoria del departamento de Electrónica de la UTFSM realizado el año 2008[3], implementó el diseño de un prototipo de termocontrolador para regular la temperatura de salida de un calentador de agua para uso doméstico. La regulación de temperatura se hizo mediante la activación de válvulas que permiten el paso de más o menos agua por un calentador, controlando de esta forma esta variable. En este trabajo se evaluó el uso de un PLC, pero por sus altos costos, se optó por utilizar una plataforma de desarrollo con el microcontrolador MSP430, desarrollado en otra memoria.

Siemens LME [4]

La línea LME de controladores Siemens es usada para el encendido y supervisión de quemadores de gas de operación intermitente, con o sin soplador. Es la opción utilizada por los quemadores de QUEMPIN antes de realizar esta memoria.

Estos controladores permiten el encendido del soplador, transformador de ignición y las válvulas de los quemadores, siguiendo con el procedimiento de encendido que rige la norma. Al mismo tiempo, monitorea los sensores de llama y presión necesarios para un correcto funcionamiento del quemador, bloqueando la entrada de gas y el sistema completo si las condiciones de funcionamiento no son normales.

Pese a lo anterior, este controlador no permite una regulación de temperatura,

razón aire/combustible y solo habilita la alimentación del soplador, causando que la regulación de velocidad de este tenga que realizarse con circuitos por separado. Además, tampoco ofrece una interfaz de configuración, por lo que hace tedioso y complicado realizar cambios en variables tales como el tiempo de prebarrido.

2.2. Diseño de Reactores de MPI

Un trabajo de memoria realizado en el departamento de mecánica de la UTFSM, se ocupa de diseñar un reactor de medio poroso inerte de flujo recíproco, con suministro separado de aire y combustible. En este se analiza la combustión y sus productos, se construyó un sistema de inyección por válvulas para insertar el combustible premezclado con aire al reactor y se montó un sistema de control de flujo y registro de temperaturas. [5]

Un segundo trabajo de memoria del mismo departamento diseñó un reactor de medios porosos inertes de flujo recíproco con el fin de desarrollar el estudio experimental de generación de gas de síntesis mediante la oxidación de biogas sintético. También se obtuvo un registro de temperaturas, velocidades de filtración de mezcla aire-combustible y velocidades de frente de llama. [6]

En ambas memorias se concluye que es necesario implementar un controlador para el accionamiento de los reactores, dado que el proceso de encendido se realiza de forma manual y una válvula mal accionada pone en riesgo el funcionamiento del sistema.

2.3. Instrumentación disponible

La instrumentación implementada en el trabajo de memoria[2] previo al presente, desarrollado en el marco del programa de memorias multidisciplinarias de la UTFSM. Incorpora todos los sensores necesarios y se adaptaron los sensores que ya estaban en el quemador. Adicionalmente se integró una interfaz web utilizando una Raspberry Pi,

logrando un arreglo IoT. Esta instrumentación corresponde a:

2.3.1. *Detector de llama piloto*

El circuito detector de llama piloto entrega una señal ON-OFF de 0 a 3.3V, en donde se considera 0V para llama encendida y 3.3V llama apagada, es decir, se utiliza una lógica negativa.

El funcionamiento de este detector consiste en utilizar una sonda de ionización, la cual permite el paso de una pequeña corriente al estar en contacto con la llama. De esta forma, la llama actúa como un switch, permitiendo detectar la existencia de ésta.[2]

2.3.2. *Sensor de caudal de aire*

La medición del caudal de aire que ingresa al soplador se realiza a través de la medición de dos presiones en un tubo de pitot a la entrada de éste. Posteriormente se realiza la diferencia de estos, junto con un factor de ajuste experimental.[2] Todo lo anterior es realizado por un circuito adaptador, el cual entrega un voltaje de 0 a 3.3V el cual puede ser leído por el microcontrolador desde un pin análogo.

El voltaje y el caudal se relaciona de la siguiente forma:

$$Q[m^3/h] = -14,1932 \cdot v_m + 19137 \quad (2.1)$$

En donde Q es el caudal y v_m es el voltaje que entrega el circuito.

2.3.3. *Lectura de termocuplas*

Cada termocupla genera una tensión proporcional a la temperatura de ésta en el rango de los milivolts. Se utiliza un amplificador de instrumentación para obtener una salida amplificada, la cual posteriormente pasa por un filtro pasabajos de primer orden.[2]

El circuito acondicionador brinda una señal analógica de 0 a 3.3V, la cual puede ser leída desde el microcontrolador y se interpreta de la siguiente forma:

$$T[^{\circ}C] = 213,7092 \cdot v_m - 10,6852 \quad (2.2)$$

En donde T° corresponde a la temperatura en la termocupla y v_m corresponde al voltaje que entrega el circuito cuyo valor se encuentra entre 0 y 3.3V.

2.3.4. Operación de Servoposicionador en válvula monoblock

Con el fin de obtener distintas potencias, la memoria multidisciplinaria incorporó un controlador motorizado en la válvula de gas, permitiendo regular el paso de gas. Este control se realiza a través de un arduino que mide un potenciómetro.[2]

Para comunicarse con este monoblock se debe conectar el controlador escogido a través del puerto serial con este arduino.

2.3.5. Presostato de gas

Se utiliza un presostato de gas como switch para cerrar la conexión a 220V de todo el sistema. Es por esto que para encender toda la instrumentación del quemador, se debe dar el paso de gas a la conexión de la válvula monoblock.

2.3.6. IoT con Raspberry Pi

Para utilizar esta conexión, basta con una comunicación serial hacia una Raspberry Pi, notificando las variables: Temperatura de operación, razón aire combustible y RPM del soplador.

3. REQUERIMIENTOS Y ALTERNATIVAS DE SOLUCIÓN

3.1. *Requerimientos funcionales*

El requerimiento principal del controlador a desarrollar, es que cumpla con la norma NCh3111: “Quemadores automáticos de aire forzado que utilizan combustibles gaseosos”. En esta norma se describe el proceso de encendido, operación y apagado de los quemadores descritos.

Pese a que se trabaja en un ambiente industrial, se requiere un controlador de bajo costo con el fin de obtener una oferta atractiva de los quemadores de QUEMPIN frente a los hornos a leña. Este es un punto importante a la hora de escoger un controlador, dado que esta elección afecta directamente en el precio final de cada quemador.

También es requerido un monitoreo de variables controladas en tiempo real mostradas en una interfaz amigable para el operador del horno.

De esta forma, el trabajo a realizar en esta memoria considera un control de las variables relevantes para operar el horno utilizando quemadores de MPI, tales como el caudal de aire con respecto al caudal de gas y la revisión continua del correcto funcionamiento de los sensores y actuadores que componen el sistema, siguiendo la norma NCh3111. Al mismo tiempo se considera mostrar al usuario final variables de interés en tiempo real en una interfaz y ofrecer la posibilidad de asignar valores deseados, modificables por un técnico en mantenimiento.

Para encender la llama piloto según la norma indicada, se activa el transformador de ignición, causando un arco eléctrico entre los electrodos de ignición. Posteriormente se da el flujo de gas activando la válvula de gas, provocando un encendido en la llama piloto, la cual es detectada por el sensor de llama. Finalmente se apaga el

transformador de ignición.

Para encender el quemador cumpliendo la norma, se debe realizar un “prebarrido”, activando el soplador a una velocidad alta, con las válvulas de gas apagadas. Esto se realiza con el objetivo de eliminar cualquier resto de gas en el quemador antes de encender cualquier llama. Una vez realizado lo anterior, se enciende la llama piloto y posteriormente se disminuye la velocidad del soplador, activando al mismo tiempo la válvula de gas, generando la premezcla que ingresa al quemador. Esta premezcla, al entrar en contacto con la llama piloto, enciende el quemador y una vez asentada la llama del quemador, se aumenta ligeramente la velocidad del soplador llegando a un nivel de operación normal.

Finalmente, el controlador debe bloquearse, suspenderse o reiniciarse, dependiendo de las señales introducidas a éste y también, notificar fallos o advertencias según corresponda.

3.2. Requerimientos de diseño de hardware

Para satisfacer los requerimientos funcionales se debe diseñar un circuito controlador para el control del soplador de aire, un circuito para detectar la activación del presostato de aire desde el microcontrolador escogido y finalmente un circuito que active una alimentación de 220V controlado desde el microcontrolador para la alimentación del soplador, transformador de ignición y válvulas.

Se requiere un controlador que contenga al menos:

- 8 pines digitales
- I2C
- 1 puerto PWM
- 6 pines análogos

3.3. *Alternativas de solución*

Al tratarse de una aplicación industrial, es usual implementar un PLC, como suele escogerse en este tipo de proyectos.

Se cotiza un PLC estándar de la marca Allen Bradley, en su modelo Micrologix 1400. El valor de este PLC corresponde a \$547.400 y el precio del cable de programación de este controlador es de \$53.550. [7]

Esta alternativa es descartada dado que es necesario implementar un controlador que no suba demasiado el precio de los quemadores de la empresa. Ante este caso, se opta por utilizar un microcontrolador para cumplir esta tarea.

3.3.1. *Criterios de evaluación*

Se identifican los criterios de evaluación de microcontroladores a utilizar. Estas corresponden a:

1. **Arquitectura**

Existen distintos tipos de arquitecturas en microcontroladores, como ARM (utilizado en microcontroladores de NXP Semiconductors, Texas Instruments y ST-Microelectronics), AVR (Atmel), PIC (Microchip), entre otros. La mayor diferencia entre cada una de estas, corresponde a cómo operan a bajo nivel. Las arquitecturas AVR y PIC ofrecen operaciones de 8bits, 16 bits y 32 bits para el tamaño de una palabra. Por otro lado, ARM ofrece 32 bits y hasta 64 bits.

Estas arquitecturas se basan en dos arquitecturas fundamentales: RISC (Reduced instruction set computer) y CISC (Complex instruction set computer). Gran parte de los microcontroladores se basan en la primera arquitectura, dado que esta es capaz de generar un mayor número de instrucciones con un menor número de ciclos. [8]

Esta característica toma relevancia al momento de considerar la memoria necesaria para el proyecto.

2. Cantidad de pines de entrada/salida y PWM

Corresponde a la cantidad de pines disponibles para operar desde el microcontrolador. Es relevante tenerlo en cuenta dado que se deben conectar todos los sensores y actuadores a este. Al mismo tiempo se debe considerar la cantidad de canales PWM que posee, dado que el proyecto requiere al menos uno de ellos.

3. Precio

Como se mencionó anteriormente, el precio del controlador es un aspecto relevante dado que el objetivo del proyecto es lograr automatizar los quemadores a bajo costo.

4. Stock

Dado que se plantea producir el controlador de forma masiva, es necesario utilizar un microcontrolador que posea un alto stock y que no esté discontinuado o vaya a estarlo en un futuro cercano.

3.3.2. Microcontroladores a evaluar

Considerando los puntos anteriores, se escogen los siguientes microcontroladores para su evaluación:

1. ATmega328P[9]

El Atmel ATmega328P es un microcontrolador de baja potencia con arquitectura AVR de 8 bits. Actualmente es producido por la empresa Microchip y ha tomado popularidad dada su incorporación en las placas de desarrollo Arduino UNO R3. Pese a que es posible programar el microcontrolador utilizando lenguaje Arduino, también se puede lograr compilando el programa en lenguaje C.

Posee 23 pines de entrada/salida de propósito general, con 6 canales PWM. El voltaje de operación es de 1.8V a 5.5V y actualmente su precio unitario es de 2.15USD en su versión de montaje superficial, con 17.326 existencias. [10]

2. **ATmega2560** [11]

Este microcontrolador posee características similares al ATmega328P como la arquitectura AVR de 8 bits y el voltaje de operación. Es utilizado en las plataformas de desarrollo Arduino MEGA R3.

La diferencia con el microcontrolador anterior corresponde es que posee 86 pines de entrada/salida, con 12 canales PWM de 16 bits de resolución. Su precio unitario promedio es de 12.21USD con más de 9000 existencias. [10]

3. **PIC16f628A** [12]

Corresponde a un microcontrolador fabricado por Microchip. Con arquitectura RISC mejorada de 8 bits, con un oscilador interno de 4MHz. Posee 16 pines de entrada/salida y un canal PWM. Su voltaje de operación posee un rango entre 2V a 5.5V y actualmente su precio unitario es de 1.79USD con más de 11.000 existencias. [10]

4. **MSP430F1612** [13]

Este microcontrolador es fabricado por Texas instruments. Posee una arquitectura RISC de 16 bits y 48 pines de entrada/salida con un canal PWM. Su voltaje de operación se encuentra entre 1.8V y 3.6V. Actualmente su precio unitario es de 17.91USD con 811 existencias. [10]

5. **PIC24FJ1024GA606** [14]

Microcontrolador fabricado por Microchip. Posee una arquitectura de 16 bits con un oscilador interno de hasta 32MHz. Tiene 53 puertos de entrada/salida disponibles con seis canales PWM. Su voltaje de operación se encuentra entre

2V a 3.6V y su precio promedio en forma de pastilla de montaje superficial es igual a 4.43USD con 1195 existencias. [10]

6. **STM32F030F4** [15]

Este microcontrolador es fabricado por STMicroelectronics. Tiene una arquitectura ARM de 32 bits. Posee 15 puertos de entrada/salida con tolerancia a 5V, pese a que su voltaje de operación se encuentra entre 2.4V y 3.6V, con seis canales PWM. Actualmente su precio promedio en forma de pastilla para montaje superficial corresponde a 1.36USD con 28.048 existencias.[10]

3.3.3. *Microcontrolador seleccionado*

En base a las alternativas planteadas, se da una ponderación a cada criterio descrito, según sea su relevancia para realizar la elección del controlador. Para la arquitectura y la cantidad de pines se da una ponderación de 35 % dado que estas características son las más importantes. Las características Precio y Stock son menos importantes, por lo que se le da una ponderación de 15 % a cada una.

Para evaluar con calificaciones, se fijó que el microcontrolador deseable, debe tener una arquitectura mayor a 16 bits, la cantidad de pines debe ser mayor a 12, con al menos un canal PWM y un stock mayor a 10000 unidades. Para el precio, el microcontrolador más barato obtiene 100 % y el resto obtiene una calificación menor proporcional a este porcentaje.

Siguiendo la pauta de alternativas de solución, se confecciona las tablas comparativas mostradas en las Tablas 3.1 y 3.2.

Característica	Ponderación	ATmega328P		ATmega2560		PIC16f628A	
		Valor	Calificación	Valor	Calificación	Valor	Calificación
Arquitectura	35%	AVR 8-bits	50	AVR 8-bits	50	RISC 8-bit	50
Cantidad de Pines I/O y PWM	35%	23 (6 PWM)	100	86 (12 PWM)	100	16 (1 PWM)	17
Precio (USD)	15%	2,15	63	12,21	11	1,79	76
Stock	15%	17326	100	9000	90	11000	100
Calificación Final		77		68		50	

Tab. 3.1: Cuadro comparativo de alternativas propuestas para el microcontrolador a utilizar

Característica	Ponderación	MSP430F1612		PIC24FJ1024GA606		STM32F030F4	
		Valor	Calificación	Valor	Calificación	Valor	Calificación
Arquitectura	35%	RISC 16-bits	100	RISC 16-bits	100	ARM 32-bits	100
Cantidad de Pines I/O y PWM	35%	48 (1 PWM)	17	53 (6 WPM)	100	15 (6 PWM)	100
Precio (USD)	15%	17,91	8	4,43	31	1,36	100
Stock	15%	811	8	1195	12	28048	100
Calificación Final		43		76		100	

Tab. 3.2: Continuación de cuadro comparativo de alternativas propuestas para el microcontrolador a utilizar

De esta forma, la alternativa con mejor calificación final corresponde al STM32F030F4, por lo tanto se escoge esta alternativa.

Este microcontrolador, al igual que el resto de las alternativas, viene incorporado en una placa de desarrollo con el mismo nombre del circuito integrado y es conocido usualmente como "Blue Pill". En esta placa, se incorporan los osciladores para el funcionamiento del microcontrolador, reguladores de voltaje, botón de reset y una entrada USB para alimentación. El voltaje de operación de este microcontrolador corresponde a 3,3V.

4. SOLUCIÓN PROPUESTA

4.1. Arquitectura de Hardware

La arquitectura de hardware propuesta se presenta en la Figura 4.1, la cual integra todos los elementos para la operación automática del quemador utilizando un microcontrolador, en donde se detalla la instrumentación ya implementada por la memoria multidisciplinaria[2] en color verde y los requerimientos pendientes de diseño, a realizarse en esta memoria, en color gris.

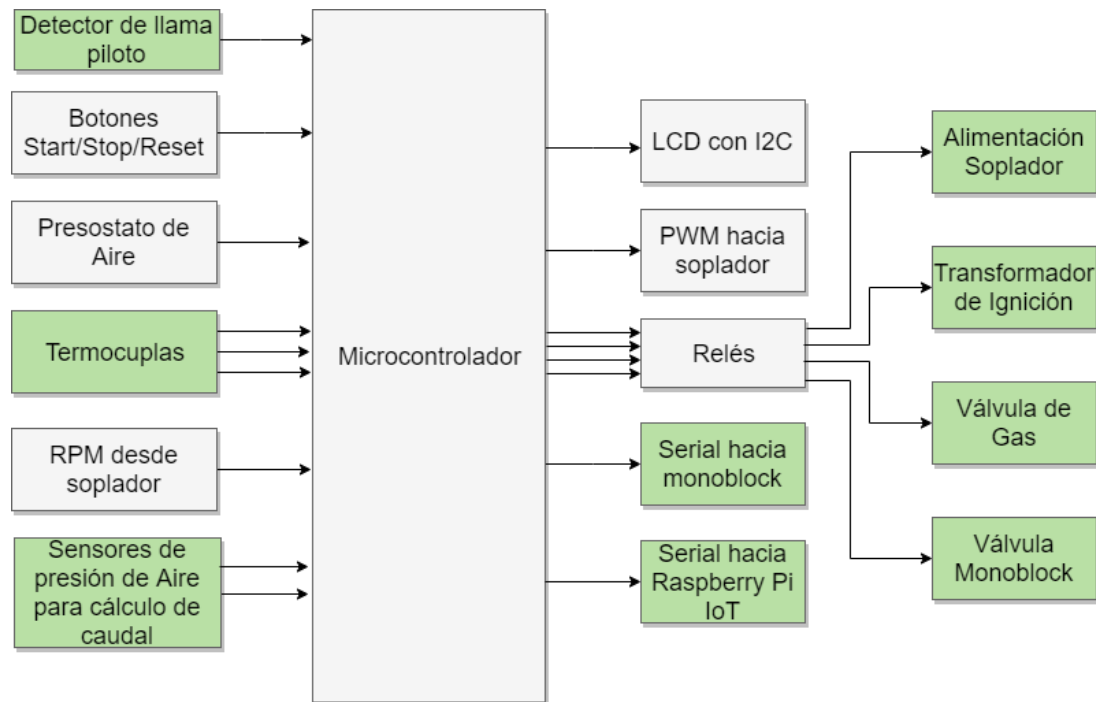


Fig. 4.1: Arquitectura de hardware propuesta

4.1.1. Botones

Se debe adaptar un circuito que lea la habilitación de pulsadores que corresponden a la Partida, Reset y Parada de emergencia, con el fin de ser detectados por el microcontrolador.

4.1.2. Pantalla

Se incorpora una pantalla Lcd de 16x2 para notificar las fases de operación y en caso de ocurrir, reportes de bloqueo en la operación. Esta pantalla utiliza un protocolo I2C, lo cual debe conectarse al microcontrolador.

4.1.3. Soplador

El soplador utilizado por QUEMPIN en sus quemadores corresponde a un Ebm Papst RG148/1200-3633. Este modelo posee una placa de control con cuatro pines, los cuales corresponden a:

- Alimentación (V_{DC}): Utilizado para energizar la placa de control. Su valor debe ir de 18V a 43V.
- Feedback de velocidad: El soplador entrega dos pulsos por cada revolución, cuyo voltaje va desde 0V a V_{DC} . Esta velocidad va desde 0 a 8500 RPM.[19]
- PWM: Para que el soplador opere, se debe introducir una señal PWM que vaya desde 0V a V_{DC} . El ciclo de trabajo de esta señal controla las RPM del soplador.
- Tierra (GND)

Dado lo anterior, para activar el soplador, es necesario adaptar un circuito que brinde entre 18 y 43V de alimentación. Por la instrumentación ya incorporada, se utiliza una alimentación de 24V.

Al mismo tiempo, se debe generar una señal PWM de 24V desde el microcontrolador. Para esto se debe amplificar una señal de 3,3V que provenga del microcontrolador, con el fin de que llegue a 24V.

Finalmente se debe adaptar la señal de pulsos de feedback de velocidad desde los 24V que entrega el soplador, a 3,3V con el fin de que esta señal pueda ser leída por el microcontrolador.

4.1.4. Activación de actuadores

Los actuadores a activar corresponden a:

- Soplador
- Transformador de Ignición
- Válvula de gas del cabezal de combustión
- Válvula de llama piloto

Cada actuador se habilita mediante su conexión a la red de alimentación de 220V.

Para controlar el encendido de cada actuador, se requiere un circuito que active un relé conectado al cable de fase de la conexión de 220V en cada actuador, mediante la activación de un pin digital del microcontrolador. Se debe aislar las conexiones del microcontrolador de la red de 220V a través de un optoacoplador.

4.2. Desarrollo de software

El lenguaje de programación escogido corresponde a Arduino. Esto dado que es posible programar el microcontrolador escogido en este lenguaje.

Si bien el lenguaje C presenta características más robustas que Arduino, se debe considerar que todo el equipo de la empresa corresponde a ingenieros mecánicos, por lo que el lenguaje escogido se comporta de manera más intuitiva al momento de requerir implementaciones posteriores.

4.2.1. Control de velocidad del soplador

Se debe desarrollar un programa que mida el tiempo entre cada pulso entregado desde el soplador con el fin de calcular las RPM a la que éste está operando. Al mismo tiempo, se debe comprobar que es posible controlar el soplador desde la señal PWM que el microcontrolador provee.

Lo anterior se puede resumir en un lazo de control simple cuya entrada corresponda a las RPM medidas y la actuación corresponda a la señal PWM. De esta forma, se comprueba la correcta operación del soplador y este desarrollo queda disponible para otros desarrollos posteriores.

4.2.2. Encendido de llama piloto

Se debe desarrollar un programa que accione el transformador de ignición y la válvula de llama piloto, mientras realiza una lectura del detector de llama piloto.

Al mismo tiempo, se debe tener en cuenta que el sistema debe bloquearse si se detecta o no se detecta la llama, cuando no corresponde.

Esta secuencia se resume en el diagrama mostrado en la Figura 4.2.

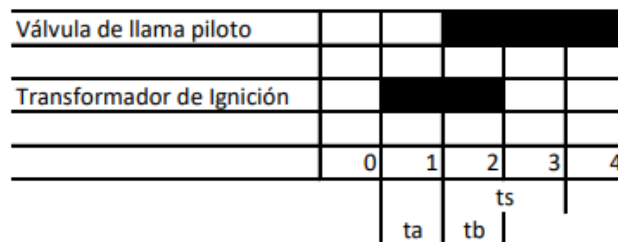


Fig. 4.2: Diagrama de fases del encendido de la llama piloto

Se identifican tres tiempos a fijar: Tiempo A (t_a), Tiempo B (t_b) y Tiempo de seguridad (t_s), con $t_b < t_s$. Este último tiempo se utiliza para que la llama piloto se estabilice mientras se enciende y se pueda medir correctamente desde el sensor de llama el encendido de esta.

Las fases corresponden a:

- ### 4.2.3. Automatización del quemador

Entradas	Detector de llama piloto	I												
	Presostato de aire	PA												
Salidas	Válvula de llama piloto	V1												
	Válvula de gas	V2												
	Transformador de Ignición	Z												
	Soplador	Velocidad 1	W1											
		Velocidad 2	W2											
		Velocidad 3	W3											
	Estado		0	1	2	3	4	5	6	7	8	9		

24

Cada estado se encuentra adaptado a la norma NCh3111, en conjunto con sus tiempos de duración que debe poseer cada fase de encendido.[18]

Estas fases corresponden a:

- Fase 0: Periodo de estabilización de sensores después de haber encendido el controlador. Luego del tiempo “t1” se pasa a la Fase 1.
- Fase 1: Fase de espera de partida. Se comprueba el correcto funcionamiento de los sensores. Dado que el soplador y la llama piloto se encuentran apagados, ambas entradas deben estar en el estado lógico 0. Si esto no ocurre, el sistema se bloquea. Al presionar el botón de partida, se avanza a la Fase 2.
- Fase 2: Se inicia el soplador en una velocidad alta. Si se detecta el estado lógico 1 en la lectura del presostato de aire, se avanza a la Fase 3. Si transcurre el tiempo “t2” y no se detecta presencia de aire, el sistema se bloquea.
- Fase 3: Fase de preventilación. El soplador expulsa aire por el quemador de forma tal de eliminar restos de combustible o premezcla que puede haber quedado de operaciones anteriores. Al comenzar la fase 3 comienza el tiempo de pre-ventilación “t3”. Al acabar este tiempo, se avanza a la Fase 4. Si durante esta operación no se detecta presión de aire a través del presostato de aire o se detecta un estado lógico 1 en el sensor de llama piloto, el sistema se bloquea.
- Fase 4: Fase de preencendido. El soplador disminuye su velocidad. Se enciende el transformador de ignición (Z). Después de transcurrido el tiempo “t4” se avanza a la Fase 5. Si se deja de detectar presencia de aire, el sistema se bloquea.
- Fase 5: Fase de postencendido. Se da el paso de gas abriendo la válvula de llama piloto (V1). Comienza a correr el tiempo de seguridad de encendido. Después de 2,5 segundos se revisa si hay llama y se avanza a la Fase 6. Si no la hay, o se termina el tiempo de seguridad de encendido sin detectar llama, o no se detecta la presencia de aire, el sistema se bloquea.

- Fase 6: Se apaga el transformador de ignición y se espera al término del tiempo de seguridad de encendido. Al terminar este tiempo, se avanza a la Fase 7.
- Fase 7: Intervalo entre el fin del tiempo de seguridad y apertura de la válvula de gas del cabezal de combustión (V2). Si se pierde presencia de aire o de llama, el sistema se bloquea. Después del fin del tiempo “t6”, se avanza a la Fase 8.
- Fase 8: Se abre la válvula V2. Después de transcurrido el tiempo “t7” se avanza a la Fase 9.
- Fase 9: Etapa de trabajo. Se aumenta la velocidad del soplador. Si se deja de detectar llama o presencia de aire, el sistema se bloquea.
- Bloqueo: Se apagan ambas válvulas, el transformador de ignición y soplador. Solo es posible volver a iniciar el sistema reiniciando el controlador.

4.2.4. Pantalla LCD

Durante el programa de automatización del sistema completo, se debe mostrar en la pantalla LCD las variables medidas por el microcontrolador y, en caso de bloqueo, se debe mostrar la etapa en que se bloqueó el sistema y a causa de qué sensor.

De esta forma, al encenderse el microcontrolador y entrar en la fase 0, se debe mostrar la palabra ”QUEMPIN.” en la pantalla. Posteriormente en la fase 1, se debe mostrar la frase ”Presione Start”, con el fin de informar que el sistema se encuentra a la espera de la partida accionada por el operador. Finalmente en las fases posteriores, se debe informar el caudal de aire y temperatura de las termocuplas.

5. DESARROLLO E IMPLEMENTACIÓN

De acuerdo a los requerimientos de diseño descritos en los capítulos anteriores, se desarrolla la solución propuesta.

5.1. Circuitos de acondicionamiento

5.1.1. Lectura de activación de presostato de aire

El presostato de aire del quemador funciona como un switch que se cierra al momento en que la presión de aire que entra en éste supera un umbral fijado.

Este presostato corresponde a un Krom Schroder DL4E-1 30Z.[16] Se diseña el circuito mostrado en la Figura 5.1, el cual utiliza un optoacoplador 4N26 para aislar los 24V, de la alimentación del microcontrolador que corresponde a 3.3V.

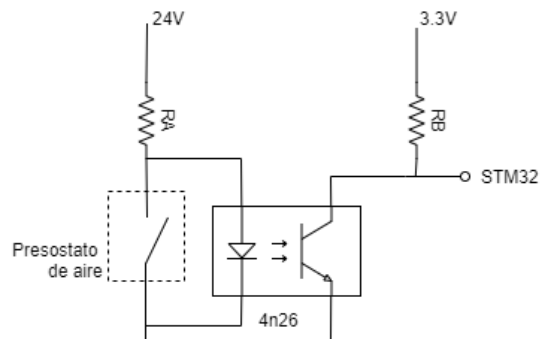


Fig. 5.1: Circuito para lectura de presostato desde STM32

Así al momento en que la presión de aire que está en el presostato se encuentra por debajo del umbral configurado en éste, el switch se comporta como circuito abierto, activando el optoacoplador, el cual conduce desde su fototransistor brindando una sa-

lida de 0V. Por otro lado, cuando se activa el presostato, el optoacoplador no conduce, provocando una salida de 3.3V.

Desde la hoja de datos del presostato, se tiene que a corriente nominal de entrada, la cual corresponde a 10mA, el voltaje en el led es de 1.2V. Tomando esto en cuenta, se tiene que el valor de la resistencia R_A está dada por:

$$R_A = \frac{24 - 1,2}{0,010} = 2280 \quad (5.1)$$

Se recomienda que si el valor de la resistencia no se encuentra en un valor comercial, se debe escoger un valor mayor cercano a este. Por lo tanto se escoge $R_A = 2,4k\Omega$

Por otro lado, la resistencia R_B actúa como una resistencia en configuración pull up. Esto es conveniente, dado que el microcontrolador utilizado tiene una configuración de lectura de pines de entrada llamada “INPUT_PULLUP”, con la cual la STM32 conecta una resistencia interna a VCC de la forma mostrada en la Figura 5.2, cuyo valor se encuentra entre 30k Ω a 50k Ω . [15]

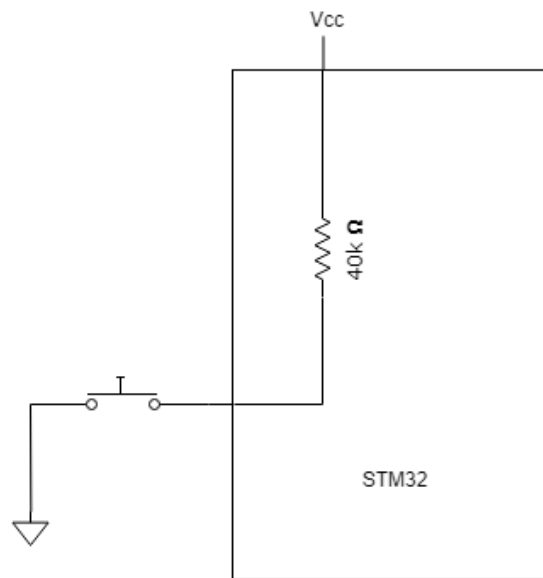


Fig. 5.2: Configuración input_pullup en STM32

Así, el diagrama planteado anteriormente queda como el mostrado en la Figura 5.3.

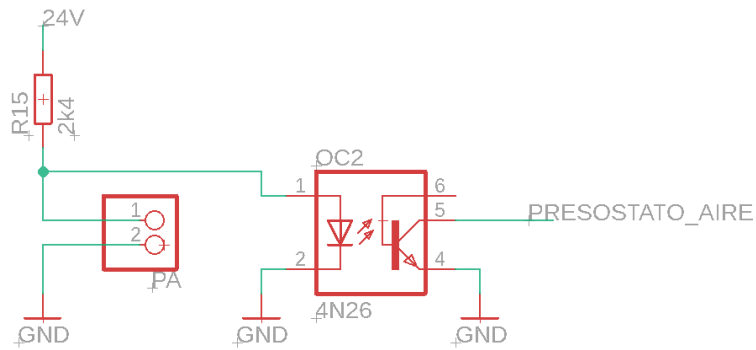


Fig. 5.3: Circuito para lectura de presostato desde STM32 utilizando la configuración “INPUT_PULLUP”

En donde el bloque PA representa los terminales del presostato de aire y el emisor del fototransistor del optoacoplador se conecta al microcontrolador.

5.1.2. Actuación PWM para Soplador

Como se mencionó en el capítulo anterior, se debe adaptar una señal digital PWM de 3,3V desde el microcontrolador hacia la placa de control del soplador que trabaja con 24V.

Considerando esto, se diseña el circuito mostrado en la Figura 5.4.

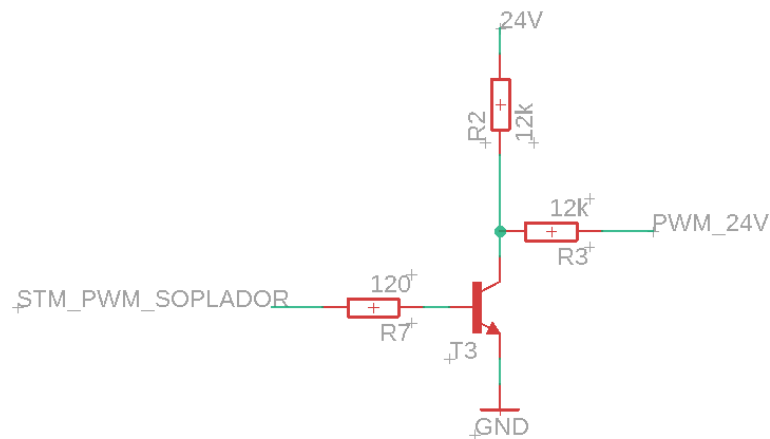


Fig. 5.4: Circuito adaptador de PWM de 3.3V a 24V para soplador

Este circuito se encuentra basado en la adaptación sugerida por el fabricante del soplador, el cual consiste en un transistor en corte/saturación con las resistencias R2 y

R3 mostradas.[19] A este circuito se le agrega la resistencia R7 mostrada en la Figura, cuyo valor corresponde a $120\ \Omega$ de forma tal que la razón entre los valores de R2 y R7 sea igual a la constante $h_{fe} = 100$ del transistor 2N2222.

Así, al conectar la entrada a 0V se obtiene una salida de 24V y al conectar la entrada a 3.3V se obtiene una salida de 0V.

5.1.3. Medición RPM de Soplador

Para la lectura de RPM desde el microcontrolador, se utiliza el circuito mostrado en la Figura 5.5.

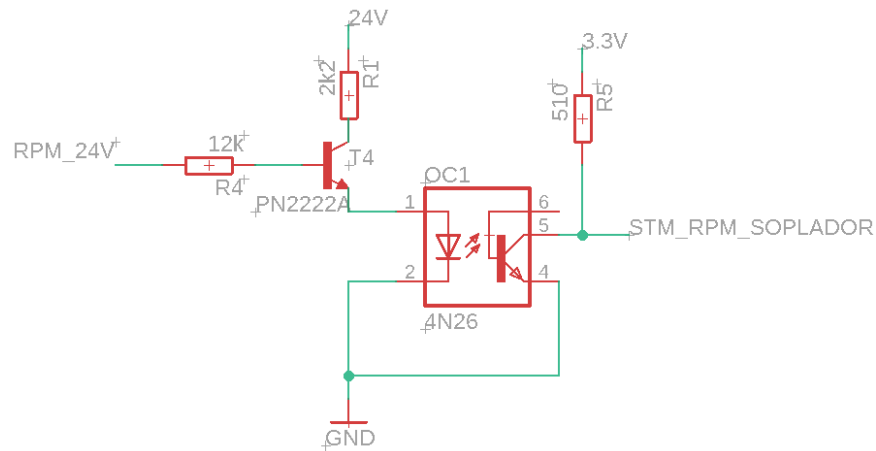


Fig. 5.5: Circuito adaptador de RPM de 24V a 3.3V para soplador

Este circuito también se basa en la recomendación del fabricante, la cual consiste en la resistencia R1, R4, un transistor NPN y un optoacoplador. A este último se le agregó una resistencia pull up de $330\ \Omega$, con el fin de que la corriente que pasa por ésta y que ingresa a la STM32 sea igual a 10mA.

5.1.4. Activación de actuadores

La activación de la alimentación del soplador, el transformador de ignición y las válvulas de gas para la llama piloto y el cabezal de combustión, se realiza utilizando un módulo de cuatro relés disponible en el mercado. La alimentación de este módulo

corresponde a 5V. Cada relé es activado por un pin digital conectado al microcontrolador de la forma que muestra el circuito de la Figura 5.6.

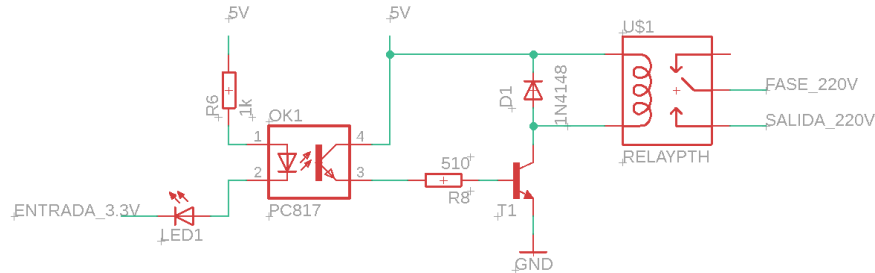


Fig. 5.6: Circuito de módulo de relés

Es posible dividir el circuito en dos partes: Una que habilita el optoacoplador y otra que activa el relé desde este último, aislando de esta forma la alimentación del microcontrolador, de la conexión de 220V.

La activación del relé ocurre solo cuando la entrada del circuito está en 0V. De esta forma cualquier otro voltaje presente en esta entrada, apaga el actuador. Es por lo anterior, que es posible alimentar el circuito con 5V y controlar el relé desde el microcontrolador con lógica de 3.3V. [20].

5.1.5. Lectura de otros sensores

Para la lectura de las termocuplas, caudal de aire y caudal de gas, se conecta el circuito desarrollado en la memoria de Esteban Arriagada[2] a los pines analógicos del microcontrolador. Para la lectura del sensor de llama, se conecta la salida del circuito desarrollado en esa memoria, a un pin digital del microcontrolador.

5.2. Configuración de pines en el controlador

Se utilizan tres pulsadores conectados a tierra, los cuales son leídos desde el microcontrolador. Es posible leer este estado gracias a la configuración de pines digitales como INPUT_PULLUP, de la misma forma en que se utiliza la lectura del estado del

presostato de aire, por lo que se conecta un pulsador que será leído como botón de partida y otro como botón de parada de emergencia, ambos utilizando la configuración mostrada en la Figura 5.2.

De esta forma, el pin digital es leído como un 1 lógico cuando el pulsador no se encuentra activado y es leído como un 0 lógico, cuando el pulsador se presiona.

El tercer pulsador es conectado en el pin RESET, para resetear el microcontrolador.

Al mismo tiempo, se conecta la pantalla LCD en un puerto I2C del microcontrolador, junto con la alimentación de ésta.

Así, la distribución de conexiones en el microcontrolador queda dada por lo mostrado en la Figura 5.7.

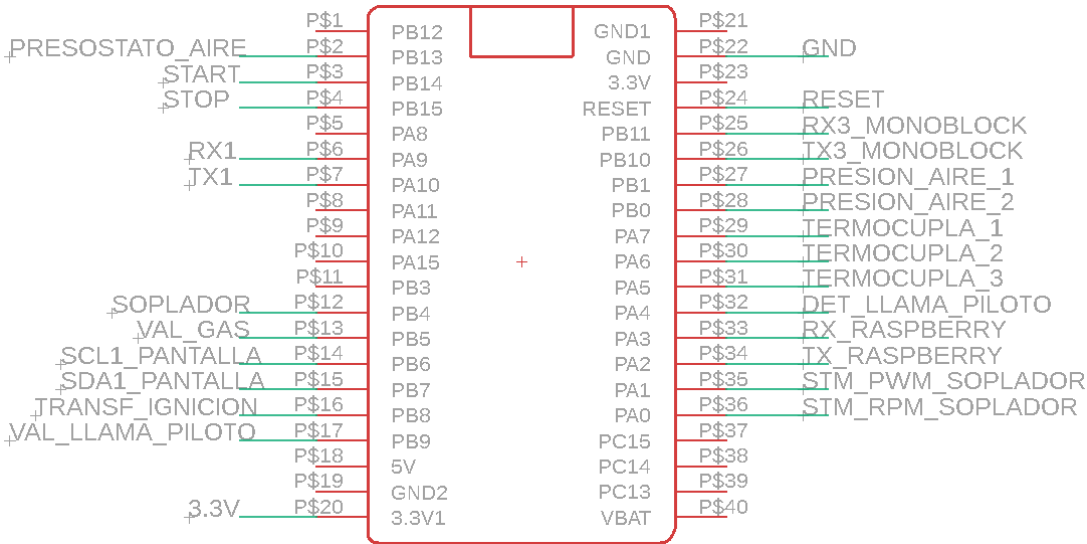


Fig. 5.7: Conexiones en el microcontrolador

Resumiendo lo anterior, se llega a la tabla de símbolos mostrado en la Tabla 5.1.

Nombre	# Pin	Tipo de pin	Descripción
PRESOSTATO_AIRE	PB13	Digital de entrada	Pin para lectura del estado del presostato de aire.
START	PB14	Digital de entrada	Pin para lectura del botón de Start.
STOP	PB15	Digital de entrada	Pin para lectura del botón de parada de emergencia.
STM_RPM.SOPLADOR	PA0	Digital de entrada	Pin para lectura de RPM del soplador.
SOPLADOR	PB4	Digital de salida	Pin para control de relé en alimentación del soplador.
VAL_GAS	PB5	Digital de salida	Pin para control de relé para activar la válvula de gas.
TRANSF_IGNICION	PB8	Digital de salida	Pin para control de relé para activar el transformador de ignición.
VAL_LLAMA_PILOTO	PB9	Digital de salida	Pin para control de relé para activar la válvula de gas de llama piloto.
STM_PWM.SOPLADOR	PA1	Digital de salida PWM	Pin para enviar la señal PWM de control del soplador.
SCL1_PANTALLA	PB6	I2C SCL	Pin de SCL para pantalla LCD.
SDA1_PANTALLA	PB7	I2C SDA	Pin de SDA para pantalla LCD.
TERMOCUPLA1	PA7	Análogo	Pin para lectura de temperatura.
TERMOCUPLA2	PA6	Análogo	Pin para lectura de temperatura.
TERMOCUPLA3	PA5	Análogo	Pin para lectura de temperatura.
RX1	PA9	Serial 1 RX	Pin serial de conexión con FTDI para programación del microcontrolador.
TX1	PA10	Serial 1 TX	Pin Serial de conexión con FTDI para programación del microcontrolador.
RX_RASPBERRY	PA3	Serial 2 RX	Pin serial para comunicación con Raspberry Pi.
TX_RASPBERRY	PA2	Serial 2 TX	Pin serial para comunicación con Raspberry Pi.
RX3_MONOBLOCK	PB11	Serial 3 RX	Pines seriales para comunicación con arduino en válvula Monoblock.
TX3_MONOBLOCK	PB10	Serial 3 TX	Pines seriales para comunicación con arduino en válvula Monoblock

Tab. 5.1: Tabla de símbolos para las conexiones en el microcontrolador

5.3. Alimentación del sistema

Para alimentar todo el desarrollo se necesitan 3 niveles de voltaje: 24V, 5V y 3.3V.

Para lograr lo anterior, se conecta una fuente switching de 24V 1A, con dos módulos en cascada: un módulo DC-DC step down 2596 el cual es un circuito del tipo buck, cuya salida es regulada a 5V. Posteriormente esta salida es conectada a otro módulo DC-DC step down MP1584, el cual también es un circuito buck, que convierte la

entrada de 5V a 3.3V.

5.4. Control PI de la velocidad del soplador

Para el control del soplador se realiza una lectura de las RPM utilizando las interrupciones que realiza este pin en el microcontrolador. Se configuran las interrupciones en el pin de lectura de las RPM de forma tal que se activen al registrar un cambio en el nivel lógico (0 a 1, o viceversa). Al ocurrir una interrupción, se ejecuta la siguiente función:

```
1 void interruptFunction() {  
2   t1 = t2;  
3   t2 = micros();  
4   medicion = 30000000 / (t2 - t1);  
5 }
```

Ésta toma el tiempo en microsegundos entre cada interrupción, correspondiente a la mitad de un período. Es por esto que se asigna la variable “t” para guardar el valor del periodo, que correspondería al doble de lo medido y se realiza la conversión a RPM de la forma:

$$RPM = \frac{30000000}{t} \quad (5.2)$$

Para regular la velocidad del soplador, se genera una señal PWM cuyo ciclo de trabajo es inversamente proporcional a esta. Por lo que en un ciclo de trabajo de 100 % el soplador se detiene y en 0 % trabaja a su máxima velocidad.

Considerando lo anterior, se incorpora un controlador PI de la forma:

$$D = k_p error + k_i integral \quad (5.3)$$

Donde:

$$error = sepoint - medicion \quad (5.4)$$

$$integral = \sum error \cdot 0,000001 \cdot T \quad (5.5)$$

en donde T corresponde al tiempo transcurrido entre la actuación anterior y la actuación actual.

Así, D corresponde a la actuación que se resta de un ciclo de trabajo de 50 %. Es por esto que se fija una saturación de $\pm 50\%$ en la actuación.

Para generar una PWM desde el pin escogido en el microcontrolador, se configura el timer que éste posee con el fin de fijar una frecuencia de 4kHz, dado que el fabricante del soplador recomienda una frecuencia de PWM para control que se encuentre entre 1kHz y 6kHz[19]. Así, el código en Arduino para esta configuración corresponde a:

```
1 HardwareTimer pwmtimer(2); //Se escoge el timer 2
2 pwmtimer.pause(); //Se pausa el timer
3 period = 250;
4 pwmtimer.setPeriod(period); //Configuracion de periodo
5 pwmtimer.refresh();
6 pwmtimer.resume(); //Habilitacion de timer
7 pinMode(pwmMotor, PWM); //Configuracion de pin
```

Para generar una PWM con un ciclo de trabajo D, se utiliza la siguiente línea de código:

```
1 analogWrite(pwmMotor,D);
```

Dado el cambio del valor del periodo del timer, la magnitud que puede poseer D va entre 0 a 18000, en donde 0 corresponde a 0 % de ciclo de trabajo y 18000 corresponde

a 100 %. Es por esto que se realiza un ajuste lineal dado que se requiere 50 % de ciclo de trabajo al encontrar error 0, obteniendo:

```
1 analogWrite (pwmMotor, 9000-180*D) ;
```

5.5. Grafcet de encendido de llama piloto

Se resume lo descrito en el capítulo anterior en el diagrama GRAFCET mostrado en la Figura 5.8.

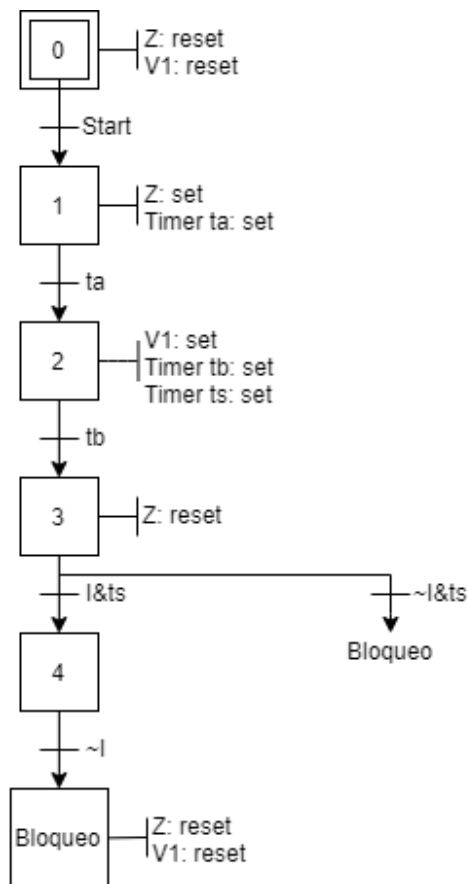


Fig. 5.8: Diagrama grafcet de encendido de llama piloto

Donde la tabla de símbolos corresponde a la mostrada la Tabla 5.2.

Y la tabla de transiciones y acciones de cada estado se observa en la Tabla 5.3.

Símbolo	Tipo	Descripción
Start	Entrada	Botón de partida
I	Entrada	Sensor de llama piloto
V1	Salida	Válvula de gas de llama piloto
Z	Salida	Transformador de Ignición
ta	Timer	Timer a (2 segundos)
tb	Timer	Timer b (2,5 segundos)
ts	Timer	Tiempo de seguridad (5 segundos)

Tab. 5.2: Tabla de símbolos de diagrama grafcet de encendido de llama piloto

Estado	Acción	Transición
0	Desactiva Z y V1	Activación de botón Start
1	Activa Z y Timer ta	Fin de Timer Ta
2	Activa V1, timer tb y timer ts	Fin de timer tb
3	Desactiva Z	Activación de I y fin de timer ts
4	-	Desactivación I
Bloqueo	Desactiva Z y V1	-

Tab. 5.3: Tabla de transiciones y acciones del diagrama grafcet de encendido de llama piloto

Posteriormente este diagrama grafcet se programa en Arduino, el cual es mostrado en la sección “Códigos utilizados” en los anexos, con el nombre “Prueba_sensor_llama.ino”.

En este código se define SET como un estado de 0V y RESET como un estado de 3.3V. Esto dado que la activación de los relés se realiza llevando la entrada a tierra. También se destaca el uso de interrupciones para el botón de partida.

5.6. Grafcet de encendido del sistema completo

Al igual que en la sección anterior, se resume las fases de operación en un diagrama GRAFCET, mostrado en la Figura 5.9.

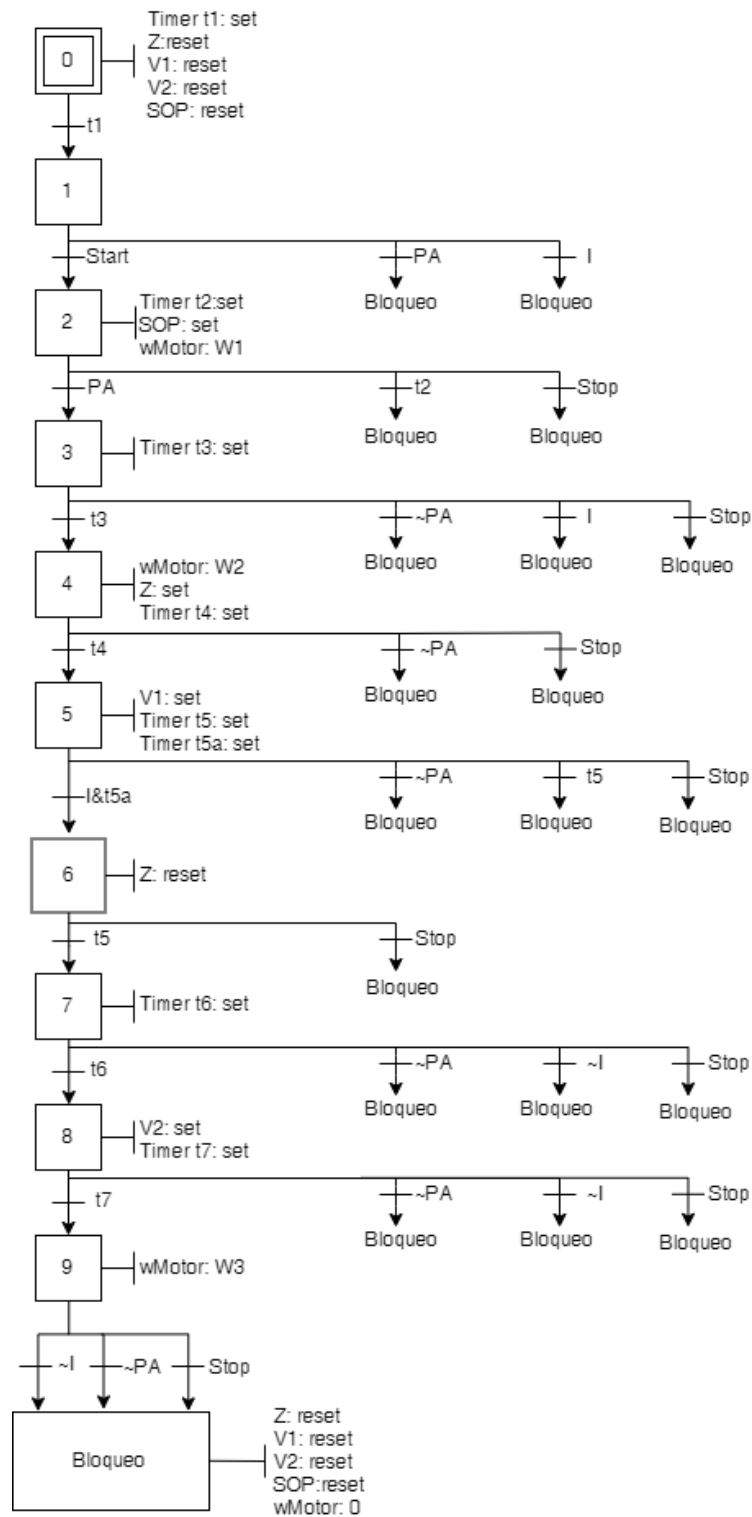


Fig. 5.9: Diagrama grafcet del encendido y operación del quemador

Cuya tabla de símbolos se observa en la Tabla 5.4.

Símbolo	Tipo	Descripción
Start	Entrada	Botón de partida
Stop	Entrada	Botón de parada de emergencia
PA	Entrada	Presostato de aire
I	Entrada	Sensor de llama piloto
V1	Salida	Válvula de gas de llama piloto
V2	Salida	Válvula de gas del cabezal de combustión
SOP	Salida	Alimentación del soplador
Z	Salida	Transformador de Ignición
t1	Timer	Timer 1 (5 segundos)
t2	Timer	Timer 2 (5 segundos)
t3	Timer	Timer 3 (180 segundos)
t4	Timer	Timer 4 (2 segundos)
t5	Timer	Timer 5, tiempo de seguridad (7 segundos)
t5a	Timer	Timer 5 auxiliar (2,5 segundos)
t6	Timer	Timer 6 (3 segundos)
t7	Timer	Timer 7 (30 segundos)

Tab. 5.4: Tabla de símbolos de diagrama grafcet de encendido y operación del quemador

Y la tabla de transiciones y acciones de cada estado se observa en la Tabla 5.5.

Estado	Acción	Transición
0	Desactiva Z, V1, V2, SOP y Activa Timer t1	Fin de t1
1	-	Activación de Start
2	Activa SOP, Timer t2 y asigna a wMotor el valor W1	Activación de PA
3	Activa timer t3	Fin de t3
4	Activa Z, timer t4 y asigna a wMotor el valor W2	Fin de t4
5	Activa V1, timer t5 y timer t5a	Activación de I y Timer t5a
6	Desactiva Z	Fin de t5
7	Activa timer t6	Fin de t6
8	Activa V2 y timer t7	Fin de t7
9	Asigna a wMotor el valor W3	Desactivación de I, PA o activación de Stop
Bloqueo	Desactiva Z, V1, V2, SOP y asigna a wMotor el valor 0	-

Tab. 5.5: Tabla de transiciones y acciones del diagrama grafcet de encendido y operación del quemador

6. RESULTADOS

Se realizan las pruebas implementadas en el microcontrolador, obteniendo los resultados mostrados a continuación.

6.1. Control de velocidad del soplador

Se ejecuta el programa mostrado en “Control_RPM_PI.ino”, incluido en los anexos del presente informe y se obtiene el gráfico mostrado en la Figura 6.1.

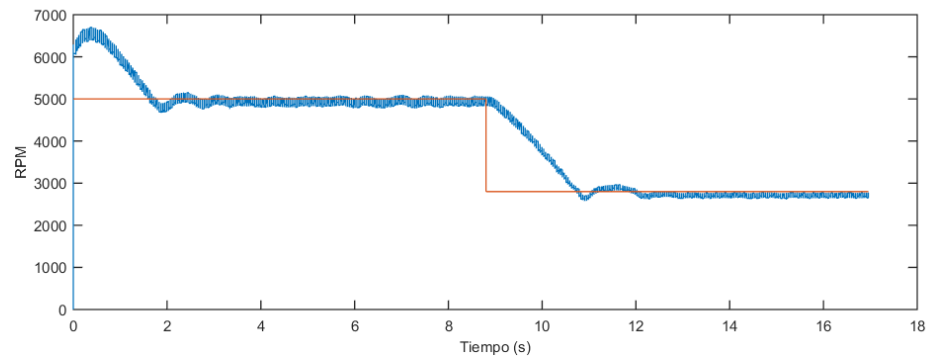


Fig. 6.1: RPM medidas en el soplador, controladas con un PI, en donde las curvas en azul corresponden a las RPM medidas y las rectas rojas corresponden a la referencia

En esta prueba, se utilizan las siguientes constantes del controlador PI:

- $k_p = 0,3$
- $k_i = 0,2$

las cuales fueron obtenidas iterativamente utilizando métodos de oscilación y posteriormente corrección de error estacionario. Se observa un buen seguimiento de la referencia, por lo que las constantes del PI son correctas.

6.2. Lectura de activación de presostato de aire

Para comprobar la buena lectura del circuito del presostato de aire, se ejecuta el código “Presostato_PI.ino” también incluido en los anexos. El cual realiza un pulso de 5 segundos en el setpoint del PI incorporado anteriormente, mientras mide la señal del presostato de aire desde el microcontrolador.

El resultado de lo anterior se observa en la Figura 6.2.

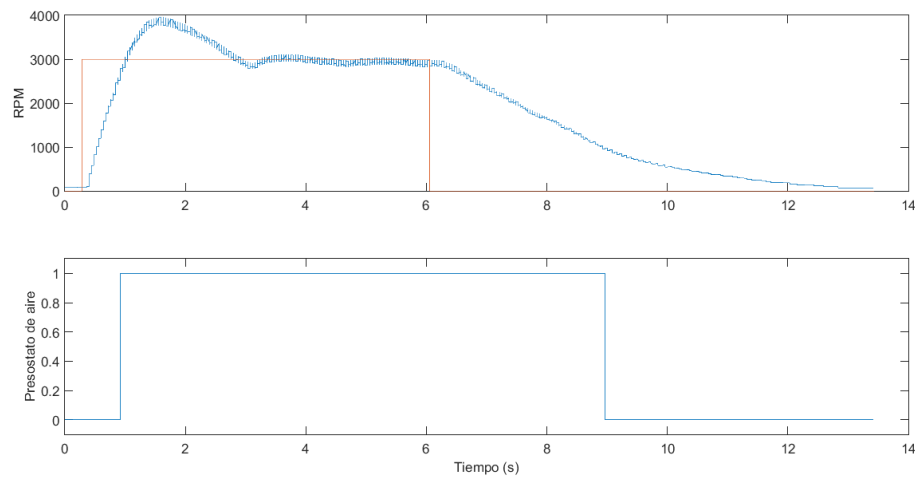


Fig. 6.2: Prueba del circuito de lectura de la activación del presostato de aire.

En el gráfico superior se realiza un pulso en la referencia del lazo de control de velocidad, mostrado en color rojo y al mismo tiempo se muestra la lectura en color azul. En el gráfico inferior se muestra la respuesta del presostato de aire leída desde el microcontrolador.

Se observa una correcta medición de la activación del presostato de aire en el microcontrolador.

6.3. Encendido de llama piloto

Se realiza el encendido de la llama piloto utilizando la instrumentación implementada y el código “Prueba_sensor_llama.ino”, incluido en los anexos, obteniendo los

gráficos de la Figura 6.3. Con la finalidad de comparar el avance de cada estado en todos los gráficos, se incluyeron guías en color azul, las cuales separan cada estado del programa.

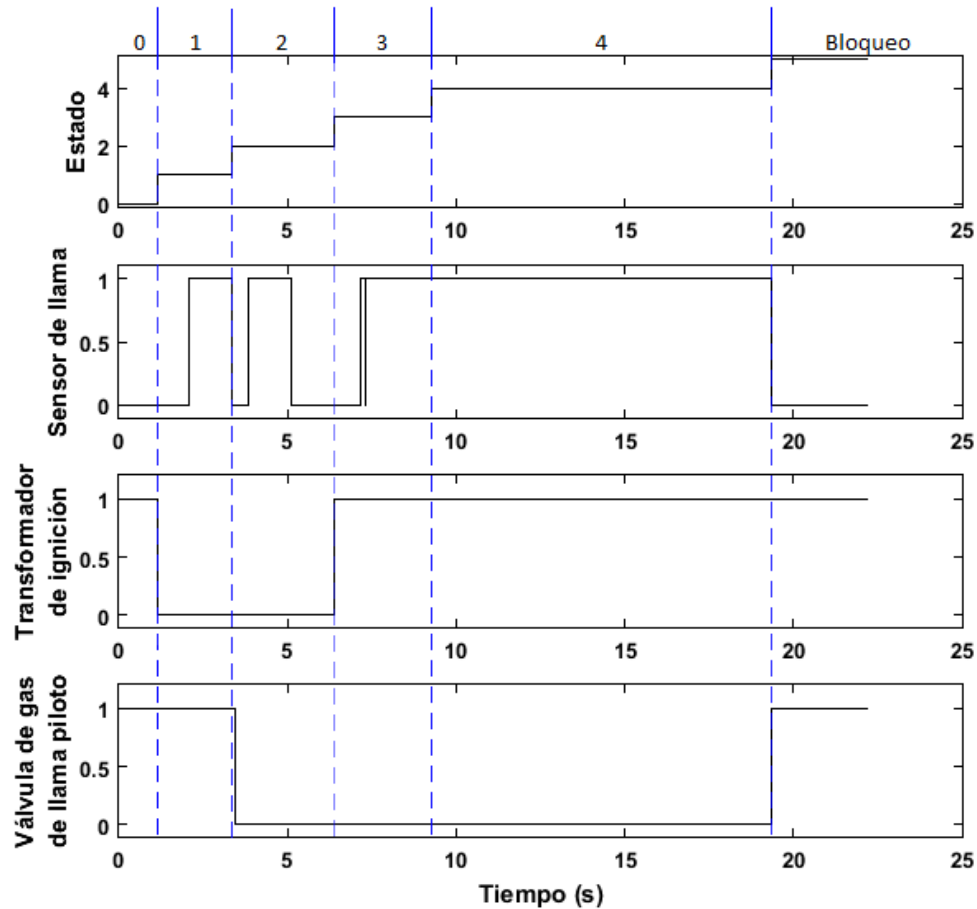


Fig. 6.3: Medición del sensor de llama, estado del programa, estado del transformador de ignición y válvula de gas, con respecto al tiempo.

Los tiempos utilizados en esta prueba fueron:

- $t_a = 2$ segundos
- $t_b = 2.5$ segundos
- $t_s = 5$ segundos

Como se explicó en capítulos anteriores, los actuadores se activan si la salida de los puertos digitales a los cuales están conectados se encuentran en un 0 lógico.

Se observa un comportamiento efectivo en los actuadores, respecto del avance de los estados. El botón de partida fue presionado, avanzando al estado 1, en donde se activa el transformador de ignición. Transcurrido el tiempo “ta”, el microcontrolador avanza al estado número 2, en donde se activa la válvula de llama piloto. Luego de transcurrir el tiempo “tb”, se avanza al estado 3 apagando el transformador de ignición. A partir de este punto, comienza a correr el tiempo de seguridad “ts”, el cual da un momento para que la llama se estabilice luego de su ignición. Al finalizar el tiempo de seguridad, se revisa que el sensor de llama está activado y se avanza al estado 4.

Dada la programación realizada, el microcontrolador se encontrará monitoreando constantemente la llama piloto y bloqueará el sistema si no la detecta. Esto ocurre previo al segundo 20, en donde se apagó intencionalmente la llama piloto y la STM32 programada entró en el estado de bloqueo, apagando ambos actuadores.

Es importante destacar el hecho de que al encender el transformador de ignición, se produce una baja de voltaje en el sensor de llama, resultando en una medición de llama encendida. Esto no es relevante para los bloqueos, dado que en los estados en que el transformador de ignición se encuentra encendido, no se revisa el encendido de llama.

Posteriormente es posible observar que al entrar al estado 3, el sensor de llama debe estabilizar su medición. Esto no toma más de la mitad del tiempo de seguridad, por lo que es posible reducir este tiempo a 2.5 segundos.

6.4. Encendido del sistema completo

Utilizando el programa anterior, se logra encender la llama piloto y posteriormente se regula el paso de gas de la válvula de gas del cabezal de combustión. De esta forma, se realiza el proceso de encendido manual llegando a una velocidad de encendido para el soplador de 1600 RPM y una velocidad de operación de 1800 RPM, las cuales

garantizan una razón de aire/combustible ideales para un buen encendido y operación.

Se programa el graficet mostrado en el capítulo anterior, en el archivo “final.ino”, incluido en los anexos.

Se obtiene de esta forma las mediciones y activaciones mostradas en la Figura 6.4.

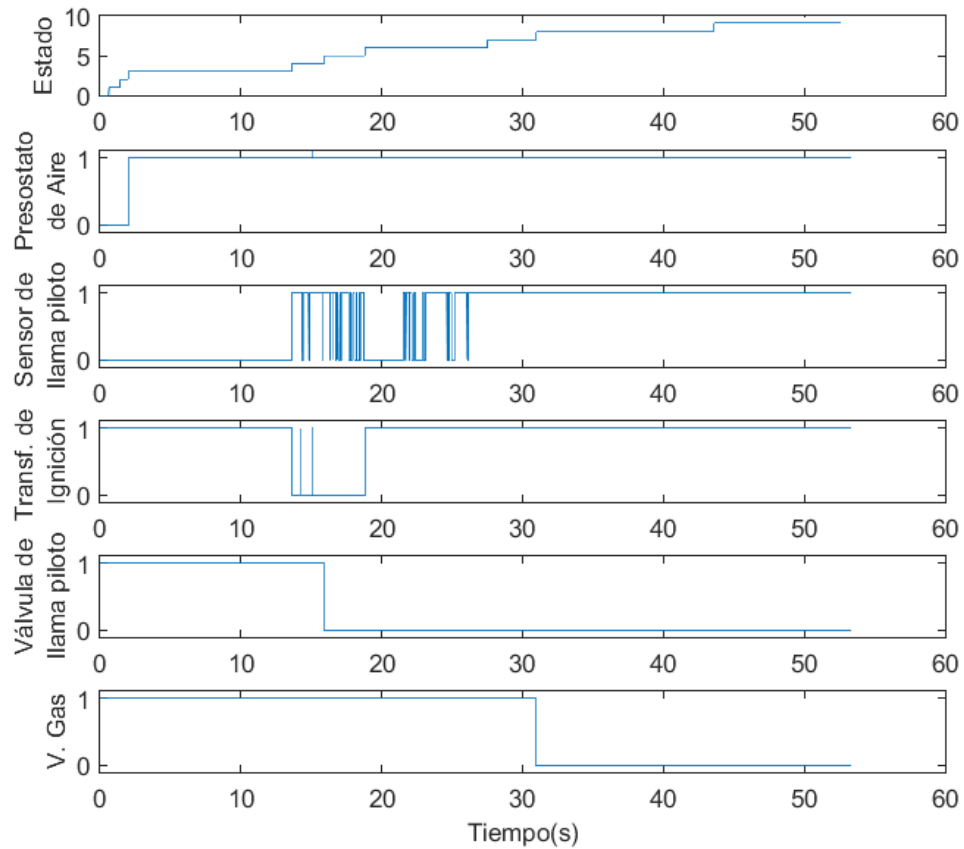


Fig. 6.4: Secuencia de encendido completa

De esta forma se observa que se cumple la secuencia de encendido diseñada y que rige la norma.

7. CONCLUSIONES

- Fue posible encender el quemador automáticamente siguiendo la secuencia de encendido logrando llegar a la operación deseada siguiendo la norma NCh3111.
- La estabilización de la medición del sensor de llama piloto ocurre en un tiempo que no supera la mitad del tiempo de seguridad escogido para las pruebas. Por lo tanto es posible disminuir este tiempo a la mitad, sin afectar al buen funcionamiento y toma de mediciones en el proceso de encendido del quemador.
- Uno de los aportes más importantes de este trabajo corresponde al desarrollo del lazo de control para la regulación de velocidad del soplador, lo cual a la fecha se ha implementado ya en seis quemadores comercializados por la empresa QUEMPIN, utilizando un microcontrolador AtTiny con las mismas funciones y circuito adaptador desarrollados en la presente memoria. Esto presenta una mejora en el desempeño de los quemadores dado que anteriormente el control de velocidad de los sopladores era a lazo abierto y presentaba caídas o aumentos de velocidad ante perturbaciones en la red eléctrica con la que se alimentaba el soplador.
- El controlador desarrollado en este trabajo presenta mejoras respecto al controlador utilizado previamente en los quemadores de QUEMPIN dado que si ocurre un bloqueo en el controlador desarrollado, se informa la fase en la que estaba el quemador y el sensor que falló, lo cual es un gran aporte a la resolución de problemas de operación. También se agrega un botón de parada de emergencia, el cual resulta de gran importancia para la seguridad de operación con los quemadores.

- Una posible mejora al controlador desarrollado puede ser la de incorporar una conexión a internet a través de algún módulo disponible, con el fin de realizar un arreglo IoT desde el mismo controlador. Otro aspecto a mejorar puede ser el poder lograr un lazo de control desde el caudal de aire y combustible obteniendo una tasa de quemador fijada por el microcontrolador.

Bibliografía

- [1] QUEMPIN Sectores: Panaderías [en línea]
<www.quempin.cl/sectores/panaderias>[consulta: 10 abril 2019]
- [2] ARRIAGADA, Esteban. “Instrumentación para un quemador de medios porosos inertes”. Memoria de Titulación. UTFSM, Departamento de Electrónica, 2019.
- [3] QUIJADA, Diego. “Diseño y construcción de un prototipo comercial de termocontrolador de uso doméstico”. Memoria de Titulación. UTFSM, Departamento de Electrónica, 2008.
- [4] SIEMENS “Burner Control LME...” Datasheet
- [5] SUBIABRE, Guido. “Diseño e implementación de un reactor de medio poroso inerte de flujo recíproco con suministro separado de aire y combustible”. Memoria de Titulación. UTFSM, Departamento de Mecánica, 2016.
- [6] ROSAS, Joaquín. “Aplicación de la tecnología de combustión en medios porosos inertes en oxicomustión y generación de gas de síntesis”. Memoria de Titulación. UTFSM, Departamento de Mecánica, 2016.
- [7] PLCChile, Allen Bradley [en línea] <<https://www.plcchile.com/allen-bradley/>>
[consulta: 19 abril 2019]
- [8] Octopart.com: How to choose a microcontroller [en línea]
<<https://octopart.com/blog/archives/2017/08/how-to-choose-a-microcontroller>> [consulta: 19 abril 2019]

- [9] ATmega328/P Datasheet
- [10] Mouser electronics [en línea] <<https://www.mouser.cl/>> [consulta: 19 abril 2019]
- [11] ATmega2560 Datasheet
- [12] PIC16F628A Datasheet
- [13] MSP430F1612 Datasheet
- [14] PIC24FJ1024GA606 Datasheet
- [15] STM32F030F4 Datasheet
- [16] Krom Schroeder “Pressure switch for air DL, Technical information GB” 4a edición, 2003.
- [17] STEPHANOPOULOS, George. “Chemical process control: An introduction to theory and practice”, 1984.
- [18] INSTITUTO Nacional de Normalización (Chile). Quemadores automáticos de aire forzado que utilizan combustibles gaseosos. NCH3111: Of. 2007. Santiago, 2007. 79 p
- [19] Ebm-Papst “EC centrifugal fan RG148/1200-3633” Datasheet
- [20] Handson Technology, “Channel 5V Optical Isolated Relay Module” Datasheet.

8. ANEXOS

8.1. Imágenes



Fig. 8.1: Contenedor del controlador implementado

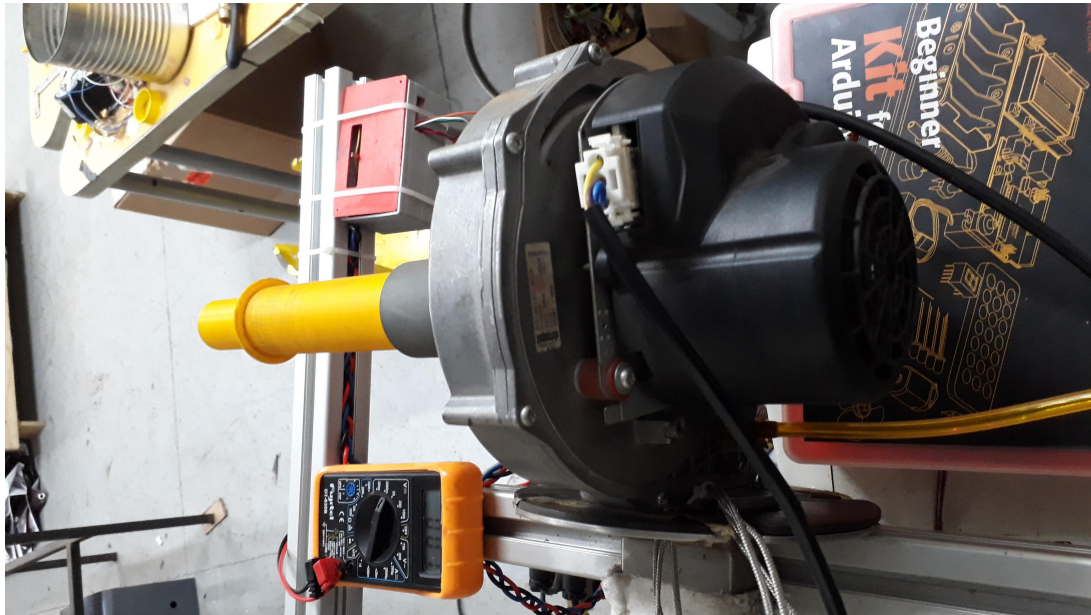


Fig. 8.2: Soplador



Fig. 8.3: Llama piloto

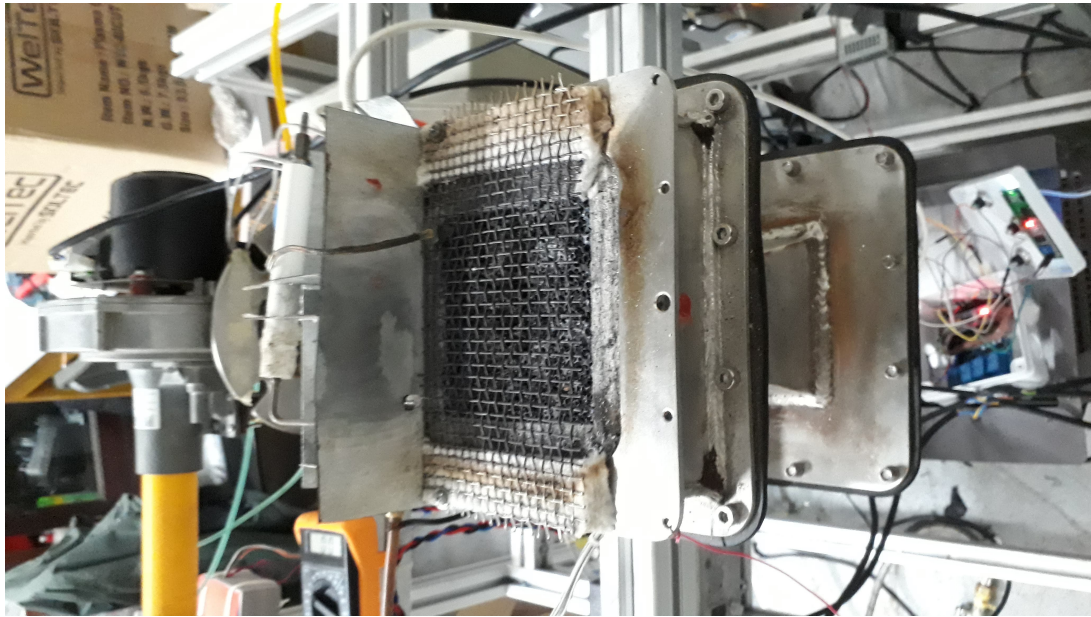


Fig. 8.4: Quemador con MPI apagado

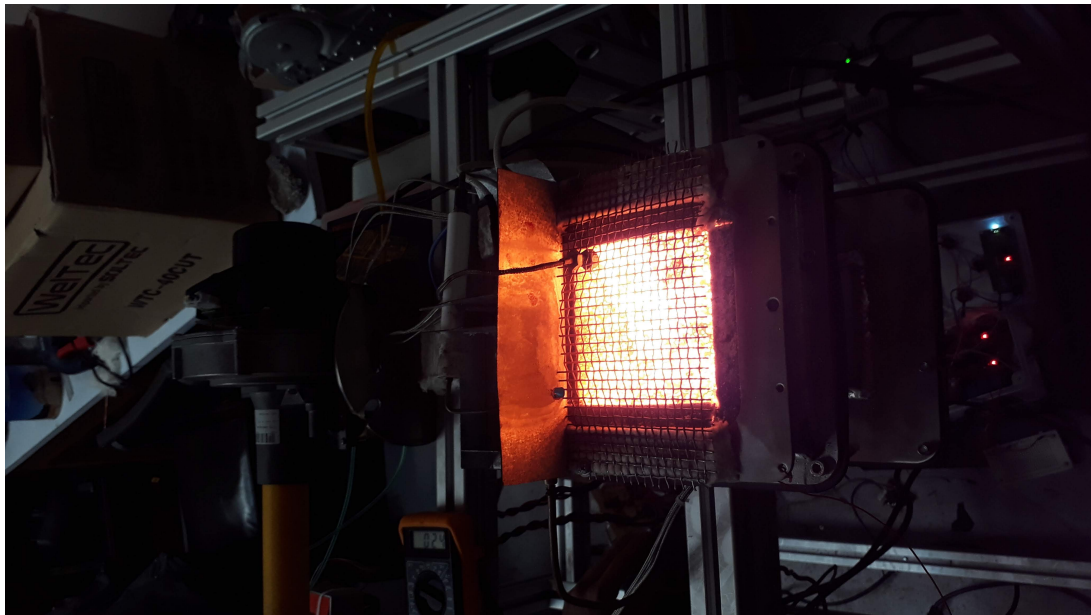


Fig. 8.5: Quemador con MPI encendido

8.2. Códigos utilizados

8.2.1. *Control_RPM.PI.ino*: “Prueba de control de soplador con PI y RPM”

```
1 // % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
2 //          Prueba de control de soplador con PI y RPM
3 // % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
4 HardwareTimer pwmtimer(2); //Timer 2
5
6 //Librerias para uso de LCD por I2C
7 #include <Wire.h>
8 #include "LiquidCrystal_I2C.h"
9 LiquidCrystal_I2C lcd(0x27, 16, 2);
10
11 const int PWM_SOPLADOR = PA1; //Puerto digital de salida
12 const int RPM_SOPLADOR = PA0; //Puerto digital de
    entrada
13
14 //Constante para RPM
15 int32 period;
16
17 //Variables para medicion de RPM
18 int32 t1, t2, medicion, t;
19
20 //Constantes PI
21 float error, integral;
22 float kp = 0.3;
23 float ki = 0.2;
```

```

24
25 //Actuacion de PI
26 int32 duty;
27 int32 act;
28
29 //Configuracion de timer para RPM
30 void beginTimer() {
31     pwmtimer.pause();
32     period = 250;
33     pwmtimer.setPeriod(period);
34     pwmtimer.refresh();
35     pwmtimer.resume();
36     pinMode(PWM_SOPLADOR, PWM);
37 }
38
39 void interruptFunction() {
40     t1 = t2;
41     t2 = micros();
42     t = 2*(t2-t1);
43     medicion = (30000000 / t <10000)?30000000 / t :
        medicion;
44 }
45
46 void PI_(float setpoint){
47     error = setpoint - medicion;
48     integral += error*0.000001*t;
49     duty = kp*error+ki*integral;
50     if(duty <= -50 || setpoint < 10){

```

```

51     act = 18000;
52     integral = 0;
53 }
54 else if(duty >= 50){
55     act = 0;
56     integral = 0;
57 }
58 else{
59     act = (int)(9000-180*duty);
60 }
61 analogWrite(PWM_SOPLADOR, act);
62 }
63
64 void setup() {
65     lcd.begin();
66     lcd.print("RPM:");
67     pinMode(PWM_SOPLADOR, OUTPUT);
68     beginTimer();
69     attachInterrupt(RPM_SOPLADOR, interruptFunction,
70                     CHANGE);
71     Serial.begin(115200);
72 }
73
74 void loop() {
75     if(millis()<10000){
76         PI_(5000);
77         Serial.print(5000);

```



```

7 const int RPM_SOPLADOR = PA0; //Puerto digital de
    entrada
8 const int PRESOSTATO_AIRE = PB13; //Puerto digital de
    entrada
9
10 //Constante para RPM
11 int32 period;
12
13 //Variables para medicion de RPM
14 int32 t1, t2, medicion, t;
15
16 //Constantes PI
17 float error, integral;
18 float kp = 0.3;
19 float ki = 0.2;
20
21 //Actuacion de PI
22 int32 duty;
23 int32 act;
24
25 //Configuracion de timer para RPM
26 void beginTimer() {
27     pwmtimer.pause();
28     period = 250;
29     pwmtimer.setPeriod(period);
30     pwmtimer.refresh();
31     pwmtimer.resume();
32     pinMode(PWM_SOPLADOR, PWM);

```

```

33 }
34
35 void interruptFunction() {
36     t1 = t2;
37     t2 = micros();
38     t = 2*(t2-t1);
39     medicion = (30000000 / t <10000)?30000000 / t :
        medicion;
40 }
41
42 void PI_(float setpoint){
43     error = setpoint - medicion;
44     integral += error*0.000001*t;
45     duty = kp*error+ki*integral;
46     if(duty <= -50 || setpoint < 10){
47         act = 18000;
48         integral = 0;
49     }
50     else if(duty >= 50){
51         act = 0;
52         integral = 0;
53     }
54     else{
55         act = (int) (9000-180*duty);
56     }
57     analogWrite(PWM_SOPLADOR, act);
58 }
59

```

```

60 void setup() {
61   pinMode(PRESOSTATO_AIRE, INPUT_PULLUP);
62   pinMode(PWM_SOPLADOR, OUTPUT);
63   attachInterrupt(RPM_SOPLADOR, interruptFunction,
        CHANGE);
64   Serial.begin(115200);
65 }
66
67 void loop() {
68   //primeros 2 segundos
69   if(millis()<2000){
70     PI_(0);
71     Serial.print(0);
72   }
73   else if(millis()<7000){ //entre segundo 2 y 7
74     PI_(3000);
75     Serial.print(3000);
76   }
77   else{ //posterior a 7 segundos
78     PI_(0);
79     Serial.print(0);
80   }
81   Serial.print(",");
82   Serial.print(medicion);
83   Serial.print(",");
84   Serial.println(digitalRead(PRESOSTATO_AIRE));
85 }

```

8.2.3. Prueba_sensor_llama.ino: “Prueba de sensor de llama piloto”

```

1 // %%%%%%%%%%%%%%% Prueba de sensor de llama piloto
2 //
3 // %%%%%%%%%%%%%%% Definiciones para SET y RESET
4 //Definiciones para SET y RESET
5 #define SET LOW
6 #define RESET HIGH
7
8 //Puertos Entradas digitales
9 const int partida = PB14;
10 const int parada = PB15;
11 const int detectorLlamaPiloto = PA4;
12
13
14 //Puertos Salidas
15 const int valvulaLlamaPiloto = PB9;
16 const int trafoIgnicion = PB8;
17
18 //Timers
19 uint32 comienzoFase= 0;
20 uint32 ta =2000; //Tiempo A
21 uint32 tb =2500; //Tiempo B
22 uint32 ts =5000; //Tiempo de Seguridad
23
24 //Variables
25 int I, V1, Z;
26

```

```

27 //Estado
28 int estado = 0;
29
30 //Funcion para interrupcion de boton start
31 void start() {
32     estado = 1;
33     comienzoFase =0;
34     detachInterrupt(partida);
35 }
36
37 void setup() {
38     Serial.begin(115200);
39     //Configuracion para puertos digitales de salida
40     pinMode(valvulaLlamaPiloto, OUTPUT);
41     pinMode(trafoIgnicion, OUTPUT);
42
43     //Configuracion para puertos digitales de entrada
44     pinMode(detectorLlamaPiloto, INPUT);
45     //Configuracion para puertos digitales de entrada con
        input pullup
46     pinMode(partida, INPUT_PULLUP);
47     pinMode(parada, INPUT_PULLUP);
48     //Inicializacion de estados de salidas digitales
49     Z = RESET;
50     V1 = RESET;
51 }
52
53 void loop() {

```

```

54 //Escritura de salidas digitales
55 digitalWrite(trafoIgnicion, Z);
56 digitalWrite(valvulaLlamaPiloto, V1);
57
58 //Lectura de entradas digitales
59 I = !(digitalRead(detectorLlamaPiloto));
60
61 //Envio por puerto serial de las mediciones obtenidas
62 Serial.print(I);
63 Serial.print(",");
64 Serial.print(Z);
65 Serial.print(",");
66 Serial.print(V1);
67 Serial.print(",");
68
69 switch (estado) {
70     case 0: //Fase 0
71         if (comienzoFase == 0) {
72             Z = RESET;
73             V1 = RESET;
74             comienzoFase = millis();
75             attachInterrupt(partida, start, FALLING); //
              Habilitacion de partida por se al de inicio
76         }
77         Serial.println(0);
78         break;
79     case 1: //Fase 1
80         Z = SET;

```

```

81     if (comienzoFase == 0)comienzoFase = millis();
82     if (millis() - comienzoFase > ta) {
83         estado = 2;
84         comienzoFase = 0;
85     }
86     Serial.println(1);
87     break;
88 case 2: //Fase 2
89     if (comienzoFase == 0)comienzoFase = millis();
90     V1 = SET;
91     if (millis()-comienzoFase > tb)estado = 3;
92     Serial.println(2);
93     break;
94 case 3: //Fase 3
95     Z = RESET;
96     if (millis() - comienzoFase > ts){
97         if(I == HIGH)estado = 4;
98         else estado =99;
99     }
100    Serial.println(3);
101    break;
102 case 4: //Fase 4
103     if(I==LOW)estado=99;
104     Serial.println(4);
105     break;
106 case 99: //Estado de bloqueo
107     Z = HIGH;
108     V1 = HIGH;

```

```

109     Serial.println(99);
110     break;
111     default: //Estado default
112         Z = HIGH;
113         V1 = HIGH;
114         break;
115 }
116 delay(10);
117 }

```

8.2.4. *final.ino: "Prueba de automatización final del sistema"*

```

1 HardwareTimer pwmtimer(2);
2 #include <Wire.h>
3 #include "LiquidCrystal_I2C.h"
4 LiquidCrystal_I2C lcd(0x27, 16, 2);
5
6 //Definiciones para SET y RESET
7 #define SET LOW
8 #define RESET HIGH
9
10 //Entradas digitales
11 const int entradaRPMmotor = PA0;
12 const int partida = PB14;
13 const int parada = PB15;
14 const int presostatoAire = PB13;
15 const int detectorLlamaPiloto = PA4;
16

```

```

17 //Entradas analogicas
18 const int termocupla_1 = PA2;
19 const int presion_aire_1 = PB1;
20
21
22 //Salidas digitales
23 const int pwmMotor = PA1;
24 const int alim_soplador = PA7;
25 const int valvulaLlamaPiloto = PB9;
26 const int trafoIgnicion = PB8;
27 const int valvulaGas = PB5;
28
29 //Timers
30 uint32 comienzoTmr = 0;
31 uint32 Tmr1 = 5000;
32 uint32 Tmr2 = 5000; //Tiempo sin presion de aire
33 uint32 Tmr3 = 180000; // Prebarrido
34 uint32 Tmr4 = 2000; //Tiempo chispero
35 uint32 Tmr5 = 7000; //Ts
36 uint32 Tmr5a = 2500; //Tiempo de chispero
37 uint32 Tmr6 = 3000;
38 uint32 Tmr7 = 30000;
39
40 //Constantes
41 uint32 W1 = 3000;
42 uint32 W2 = 1600;
43 uint32 W3 = 1800;
44

```

```

45
46 //Variables para operacion
47 int wMotor = 0;
48 int I, PA;
49 int V1, V2, Z, SOP;
50 int temperatur1, caudal1;
51
52 //Estados
53 int estado = 0;
54 int ult_estado = 0;
55 int bl_ll = 0;
56
57 //Variables para medicion de rpm
58 int32 duty;
59 int32 period;
60 int32 t1 = 0;
61 int32 t2 = 0;
62
63 //Variables para PID
64 float medicion, med_ant;
65 float error;
66 float integral = 0;
67 float derivativo = 0;
68
69 //Constantes PID
70 float kp = 0.5;
71 float ki = 6.8;
72 float kd = 0.00003;

```

```

73
74 //Funcion para lectura de RPM
75 void interruptFunction() {
76     t1 = t2;
77     t2 = micros();
78     medicion = 30000000 / (t2 - t1);
79 }
80
81 //Funcion para fijar frecuencia de RPM en 4kHz
82 void beginTimer() {
83     pwmtimer.pause();
84     period = 250;
85     pwmtimer.setPeriod(period);
86     pwmtimer.refresh();
87     pwmtimer.resume();
88     pinMode(pwmMotor, PWM);
89 }
90
91 //Funcion PID
92 void PID(float setpoint){
93     error = setpoint - medicion;
94     integral += error * 0.01;
95     derivativo = (medicion - med_ant) / 0.01;
96     duty = kp * error + ki * integral + kd * derivativo;
97     if (duty < -50) {
98         pwmWrite(pwmMotor, 180*95);
99         integral = 0;
100     }

```

```

101  else if (duty > 50) {
102      pwmWrite(pwmMotor, 0);
103      integral = 0;
104  }
105  else {
106      pwmWrite(pwmMotor, 9000 - 180 * duty);
107  }
108  med_ant = medicion;
109 }
110
111 //Funcion para interrupcion de boton start
112 void start() {
113     estado = 2;
114     comienzoTmr =0;
115     detachInterrupt(partida);
116     attachInterrupt(entradaRPMmotor, interruptFunction,
117                     CHANGE);
118     SOP = SET;
119 }
120 void setup() {
121     lcd.begin();
122     pinMode(valvulaLlamaPiloto, OUTPUT);
123     pinMode(valvulaGas, OUTPUT);
124     pinMode(trafoIgnicion, OUTPUT);
125     pinMode(alim_soplador, OUTPUT);
126
127     pinMode(entradaRPMmotor, INPUT_PULLUP);

```

```

128  pinMode(partida, INPUT_PULLUP);
129  pinMode(parada, INPUT_PULLUP);
130  pinMode(presostatoAire, INPUT_PULLUP);
131  pinMode(detectorLlamaPiloto, INPUT_PULLUP);
132  Z = RESET;
133  V1 = RESET;
134  V2 = RESET;
135  SOP = RESET;
136  ult_estado = 0;
137  beginTimer();
138
139  Serial.begin(115200);
140 }
141
142 void loop() {
143  //Parada de emergencia
144  if(digitalRead(parada)==LOW) {
145    ult_estado = estado;
146    estado=99;
147    bl_ll = 2;
148  }
149
150  //Escritura de salidas digitales
151  digitalWrite(trafoIgnicion, Z);
152  digitalWrite(valvulaLlamaPiloto, V1);
153  digitalWrite(valvulaGas, V2);
154  digitalWrite(alim_soplador, SOP);
155  //Lectura de entradas digitales

```

```

156 PA = (digitalRead(presostatoAire));
157 I = !(digitalRead(detectorLlamaPiloto));
158
159 //Lectura de entradas analogicas
160 temperatur1 = analogRead(termocupla_1);
161 caudal1 = analogRead(presion_aire_1);
162
163 //Conversion de voltaje a unidades medidas
164 temperatur1 = 213.7092*temperatur1-10.6852;
165 caudal1 = -14.1932*caudal1+19137;
166
167 //Control PID
168 PID(wMotor);
169
170 switch(estado){
171     case 0:
172         if (comienzoTmr == 0)comienzoTmr = millis();
173         if (millis() - comienzoTmr > Tmr1) {
174             estado = 1;
175             comienzoTmr = 0;
176         }
177         break;
178     case 1: //Espera de partida
179         if ( PA == HIGH || I == HIGH){
180             estado = 99;
181             if(PA == HIGH) bl_l1 = 0;
182             else bl_l1 = 1;
183             ult_estado = 1;

```

```

184     }
185     attachInterrupt(partida, start, CHANGE); //
        Habilitacion de partida por senal de inicio
186     break;
187 case 2:
188     wMotor = W1;
189     if (comienzoTmr == 0)comienzoTmr = millis();
190     if (millis() - comienzoTmr > Tmr2)estado = 99;
191
192     if (PA == HIGH) {
193         comienzoTmr = 0;
194         estado = 3;
195     }
196     break;
197 case 3: //preventilacion
198     if ((PA == LOW && millis() - comienzoTmr > 2000 )
        || I == HIGH){
199         estado = 99;
200         if(PA == LOW) bl_ll = 0;
201         else bl_ll = 1;
202         ult_estado = 3;
203     }
204
205     if (comienzoTmr == 0)comienzoTmr = millis();
206     if (millis() - comienzoTmr > Tmr3) {
207         estado = 4;
208         comienzoTmr = 0;
209     }

```

```

210     break;
211 case 4:
212     if (PA == LOW ) {
213         estado = 99;
214         bl_ll = 0;
215         ult_estado = 4;
216     }
217     wMotor = W2;
218     Z = SET;
219     if (comienzoTmr == 0)comienzoTmr = millis();
220     if (millis() - comienzoTmr > Tmr4) {
221         estado = 5;
222         comienzoTmr = 0;
223     }
224     break;
225
226 case 5: //Encendido llama piloto
227     if (comienzoTmr == 0)comienzoTmr = millis();
228     if (PA == LOW) {
229         estado = 99;
230         bl_ll = 0;
231         ult_estado = 5;
232     }
233     V1 = SET;
234     if (millis()-comienzoTmr > Tmr5a)estado = 6;
235     break;
236 case 6:

```

```

237     if(PA == LOW || ((millis() - comienzoTmr > Tmr5)&&
        I == LOW)) {
238         estado = 99;
239         if(PA == LOW) bl_11 = 0;
240         else bl_11 = 1;
241         ult_estado = 6;
242     }
243     Z = RESET;
244     if (I == HIGH && (millis() - comienzoTmr > Tmr5))
        {
245         estado = 7;
246         comienzoTmr = 0;
247     }
248     break;
249 case 7:
250     if (comienzoTmr == 0) comienzoTmr = millis();
251     if (millis() - comienzoTmr > Tmr6) {
252         estado = 8;
253         comienzoTmr = 0;
254     }
255     if (PA == LOW || ( I == LOW&&(millis() -
        comienzoTmr > 5000))) {
256         estado = 99;
257         if(PA == LOW) bl_11 = 0;
258         else bl_11 = 1;
259         ult_estado = 7;
260     }
261     break;

```

```

262     case 8: //Encendido MPI
263         if (PA == LOW || I == LOW) {
264             estado = 99;
265             if(PA == LOW) bl_ll = 0;
266             else bl_ll = 1;
267             ult_estado = 8;
268         }
269         if (comienzoTmr == 0)comienzoTmr = millis();
270         if (millis() - comienzoTmr > Tmr7) {
271             estado = 9;
272             comienzoTmr = 0;
273         }
274         V2 = SET;
275         break;
276
277     case 9: //Operacion
278         wMotor = W3;
279         if (PA == LOW || I == LOW) {
280             estado = 99;
281             if(PA == LOW) bl_ll = 0;
282             else bl_ll = 1;
283             ult_estado = 9;
284         }
285         break;
286     case 99: //Estado de bloqueo
287         wMotor = 0;
288         digitalWrite(trafoIgnicion, RESET);
289         digitalWrite(valvulaLlamaPiloto, RESET);

```

```

290     digitalWrite(valvulaGas, RESET);
291     digitalWrite(alim_soplador, RESET);
292     lcd.setCursor(0,0);
293     lcd.print("_____");
294     lcd.setCursor(0,1);
295     lcd.print("_____");
296     lcd.setCursor(0,0);
297     if(bl_ll==1) lcd.print("Blq._S.Llama");
298     if(bl_ll==0) lcd.print("Blq._Prst._Aire");
299     if(bl_ll==2) lcd.print("P._emergencia");
300     lcd.setCursor(0,1);
301     if(ult_estado==1) lcd.print("Espera_partida");
302     else if(ult_estado<=3) lcd.print("Preventilacion")
        ;
303     else if(ult_estado <= 5) lcd.print("Encendido_
        piloto");
304     else if(ult_estado == 6) lcd.print("Piloto_no_
        encendido");
305     else if(ult_estado <= 8) lcd.print("Encendido_MPI")
        ;
306     else if(ult_estado == 9) lcd.print("Operacion");
307     delay(10000);
308     break;
309
310     default:
311         break;
312
313 }

```

```

314
315 //Escritura en pantalla cada 400 milisegundos
316 if(millis() %400 == 0){
317     lcd.setCursor(0,0);
318     lcd.print("_____");
319     lcd.setCursor(0,0);
320     lcd.print("RPM:");
321     lcd.print((int)medicion);
322     lcd.print("_T:");
323     lcd.print(temperatural);
324     if(estado == 0){
325         lcd.setCursor(0,1);
326         lcd.print("_____");
327         lcd.setCursor(0,1);
328         lcd.print("QUEMPIN");
329     }
330     else if(estado == 1){
331         lcd.setCursor(0,1);
332         lcd.print("_____");
333         lcd.setCursor(0,1);
334         lcd.print("Presione_Start");
335     }
336     else if(estado >2){
337         lcd.setCursor(0,1);
338         lcd.print("_____");
339         lcd.setCursor(0,1);
340         lcd.print("Caudal:");
341         lcd.print(caudal1);

```

342 }

343

344 }

345 }