Universidad de La Laguna

ESCUELA POLITÉCNICA SUPERIOR DE INGENIERÍA

Sección Náutica, Máquinas y Radioelectrónica Naval

Trabajo presentado para la obtención del título de

GRADUADO EN TECNOLOGÍAS MARINAS

Presentado por

Ronaldo García Rodríguez

Algoritmo para la simulación de las pérdidas térmicas por aislamiento, temperatura de llama del quemador, determinación de calor transferido al agua y temperatura de gases de escape

Dirigido por

Carlos Efrén Mora Luis

Presentado en (Julio de 2021)

Resumen

Este trabajo tiene como objetivo mejorar el firmware de un simulador empleado para la simulación de una caldera de vapor saturado. Esta mejora ha consistido fundamentalmente en la inclusión del cálculo de la energía térmica perdida a través de la envolvente, además de la temperatura de llama y la temperatura de los gases de escape. Asimismo, se ha incorporado una curva de eficiencia dependiente de la carga de una caldera real para una determinación más realista de la potencia entregada por el quemador al agua durante la simulación. El algoritmo desarrollado se ejecuta en un procesador Atmega 2560, visualizando las variables de la simulación a través de un display LCD e interactuando con el algoritmo a través de potenciómetros, que simulan los controles de una caldera real. Las simulaciones que genera el Arduino emulan el funcionamiento de una caldera con el quemador a distintas cargas, de forma que la temperatura de escape, la temperatura de llama que se genera en el hogar y la potencia que es capaz de producir el quemador a tiempo real, se acercan fielmente a los valores reales de funcionamiento de una caldera. Se concluye, que la energía neta que entrega el quemador es inferior a la versión anterior del firmware debido a la incorporación de las pérdidas térmicas y un cálculo más realista de la potencia transferida por el quemador. Asimismo, se observa que a un mayor exceso de aire hay una disminución de la potencia entregada por el quemador.

Abstract

This work aims to improve a previous firmware of a steam baoiler simulator. The improvements consisted mainly in the inclusion of a more realistic calculus the thermal energy lost through the isolating shield,teh calculation of the flame temperature and the exhaust gas temperature. In addition, a load-dependent efficiency curve of a real boiler has been incorporated for a more realistic determination of the power supplied by the burner to the water during the simulation. The developed algorithm runs on an Atmega 2560 processor, visualizing the simulating variables through an LCD and interacting with the algorithm through potentiometers, that simulate the controls of a real boiler. The simulation is generated by the Arduino emulate the operation of a boiler with the burner at different loads and determines that the right exhaust temperature, the flame temperature in the furnace and the power that the burner is capable of producing in real-time. The value of these variable are very close to the values when operating a real boiler. To conclude the net power delivered by the burner is lower than the previous firmware version due to the including the effect that the excess of air has on the flame temperature in the furnace.

Índice general

Índice de figuras

Índice de tablas

Acrónimos

LED	Diodo Emisor de Luz
USB	Bus Serie Universal
LCD	Pantalla de Cristal Líquido
ICSP	In-Circuit Serial Programming
PWM	Modulación por Ancho de Pulsos
SDA	System Data
SCL	System Clock
I2C	Inter-Integrated Circuit
IDE	Entorno de Desarrollo Integrado

Simbología

Q	Calor transmitido por unidad de tiempo.
h	Coeficiente de convección.
A_s	Área del cuerpo en contacto con el fluido.
T_s	Temperatura de la superficie del cuerpo.
T_{inf}	Temperatura del fluido lejos de la superficie de intercambio.
k	Conductividad térmica.
r	Resistividad térmica.
$\dot{q}$	Flujo de calor (por unidad de tiempo y unidad de área).
A	Área de la superficie de contacto.
e	Espesor del material.
$(T_2 - T_1)$	Diferencia de temperatura entre el foco caliente y el foco frío.
$\dot{m}_{combustible}$	Caudal de combustible.
$P_{quemador}$	Potencia del quemador en kW.
$\%_{carga\ vapor}$	Carga de vapor en %
PCI	Poder calorífico inferior.
$\dot{M}_{fuel}$	Flujo moles de fuel.
$\dot{m}_{fuel}$	Flujo másico de fuel.
Pa_{fuel}	Peso atómico del fuel en (kg/mol).
$\dot{M}O_{2\ fuel}$	Flujo moles de oxígeno.
MO_2	Moles de oxígeno para una molécula de fuel.
$\dot{m}O_{2\ fuel}$	Caudal de oxígeno para el caudal de fuel.
PmO_2	Peso molecular del oxígeno en (g/mol).
I_e	Índice de exceso.
$\%exceso$	Porcentaje de exceso de aire.
$Q_{combustion}$	Calor de combustión.
$Q_{gasesdeexceso}$	Calor gases de exceso.
$\dot{m}$	Caudal másico.
Ce	Calor específico del fluido.
Δt	Diferencia de temperatura.
P_{H_2O}	Potencia calorífica contenido en el vapor de agua.
$\dot{m}_{H_2O}$	Caudal másico del vapor de agua.
Ce_{H_2O}	Calor específico del vapor de agua.
T_{adb}	Temperatura adiabática del combustible.
P_{CO_2}	Potencia calorífica contenido en el dióxido de carbono.
$\dot{m}_{CO_2}$	Caudal másico del dióxido de carbono.
Ce_{CO_2}	Calor específico del dióxido de carbono.
P_{N_2}	Potencia calorífica contenido en el nitrógeno.
$\dot{m}_{N_2}$	Caudal másico del nitrógeno.
Ce_{N_2}	Calor específico del nitrógeno.

T_{amb}	Temperatura ambiente o de entrada de gases de admisión.
P_{O_2}	Potencia calorífica contenido en el oxígeno.
$\dot{m}_{O_2}$	Caudal másico del oxígeno.
Ce_{O_2}	Calor específico del oxígeno.
Eff	Eficiencia de la caldera según la carga en %.
C_v	Carga de vapor en %.
$\cup$	Coeficiente conductividad térmica global en $\frac{W}{m^{2\circ}C}$.
K_{gas}	Coeficiente conductivo gases de escape.
K_{agua}	Coeficiente conductivo agua.
$L.T.M._I$	Temperatura media logarítmica iterada.
T_h	Temperatura del hogar de la caldera.
B_w	Temperatura del agua de caldera.
E_t	Temperatura de gases de escape.
F_w	Temperatura de agua de alimentación.
P_{SC}	Potencia térmica cedida por la superficie cilíndrica.
T_{inf1}	Temperatura del agua en un punto lejano de la superficie de intercambio.
T_{inf2}	Temperatura del aire en un punto lejano de la superficie de intercambio.
R_{ENV}	Resistencia térmica de la superficie de la envolvente.
R_{TP}	Resistencia térmica de la superficie de la tapa de la envolvente.
R_{conv1}	Resistencia convectiva de generada por el agua.
R_{cil1}	Resistencia cilíndrica de la envolvente.
R_{cil2}	Resistencia cilíndrica del aislante.
R_{cil3}	Resistencia cilíndrica del calorifugado.
R_{conv2}	Resistencia convectiva generado por el aire.
R_1	Radio interno de la envolvente.
R_2	Radio externo de la envolvente.
R_3	Radio interno del calorifugado.
R_4	Radio externo del calorifugado.
L	Longitud de la envolvente.
k_1	Conductividad térmica de la envolvente.
k_2	Conductividad térmica del aislante.
k_3	Conductividad térmica del calorifugado.
h_1	Coeficiente convectivo del agua.
h_2	Coeficiente convectivo del aire.
A_1	Área de transferencia en contacto con el agua.
A_2	Área de transferencia en contacto con el aire.
P_{SP}	Potencia térmica cedida por la tapa de la envolvente.
R_{conv1}	Resistencia convectiva generada por el agua
R_{pared1}	Resistencia de la tapa de la envolvente.
R_{pared2}	Resistencia del aislante.
R_{pared3}	Resistencia del calorifugado.
R_{conv2}	Resistencia convectiva generado por el aire.
L_1	Espesor de la tapa de la envolvente.
L_2	Espesor del aislante.
L_3	Espesor del calorifugado.
k_1	Conductividad térmica de la tapa de la envolvente.
k_2	Conductividad térmica del aislante.
k_3	Conductividad térmica del calorifugado.

A_{SP}	Área de la tapa de la envolvente.

1 Introducción y objetivos

El aprendizaje práctico del funcionamiento y operación de una caldera se facilita enormemente cuando se dispone de simuladores específicos. No obstante, no siempre es posible usar los modelos necesarios o estos resultan inaccesibles por los centros educativos, especialmente tratándose de la simulación de equipos complejos. En el caso de las calderas de vapor pirotubulares es posible simular su comportamiento partiendo, por un lado, de las características técnicas reales y, por otro lado, basándose en el comportamiento termodinámico de las distintas variables que actúan sobre su funcionamiento.

En trabajos anteriores al que se presenta, se ha realizado una simulación empleando hardware y software libre con la intención de disponer de un simulador de bajo coste económico que permita conectarse a un panel didáctico y a instrumentación realista a través del cual realizar ejercicios y entrenamiento tanto para su operación manual, como para realizar ejercicios con controladores PID para su operación automática mediante la estabilización del nivel de agua y de presión. Estos trabajos previos han conducido al desarrollo de un firmware para Arduino Mega, que se hallaba en su versión 0.2. Esta versión ya tiene funcionalidad para simular el funcionamiento del quemador y sus alarmas, la secuencia de arranque y barrido, el llenado a través de una válvula de mariposa, la pérdida de nivel de agua en función del vapor consumido, el aumento de temperatura de la caldera en función del calor transferido al agua, así como las alarmas vinculadas al funcionamiento del quemador, niveles de agua y presión de la caldera.

Partiendo de la versión previamente desarrollada, que simula una caldera modelo UMISA SMS 25 con una producción de vapor máxima de 10000 kg/h y una potencia máxima de 6563 kW, se han hecho varias mejoras en el código fuente. Estas mejoras han consistido, entre otras, programar un comportamiento más realista del quemador, la introducción de pérdidas térmicas a través de la envolvente de la caldera y la determinación de la temperatura de los gases de escape. De forma más específica, los objetivos de este trabajo se han centrado en las siguientes mejoras:

- En la versión inicial el cálculo de la potencia transferida al agua estaba simplificado, partiendo de un valor de potencia máxima al que se le aplicaba un coeficiente entre 0.0 y 1.0, dependiendo de la posición del mando de control. Se pretende calcular, de forma mucho más realista, la potencia transferida al agua.
- El desarrollo previo del simulador no tenía en cuenta las pérdidas térmicas a través de la envolvente. Se pretende determinar las pérdidas térmicas a través de la envolvente en función del estado de la caldera.

Estas mejoras han requerido la obtención de las ecuaciones necesarias, que se han integrado posteriormente en el código fuente del firmware del simulador: por un lado se ha determinado la potencia transferida al agua a partir del cálculo de la temperatura de llama junto al flujo másico del combustible y del aire aportados por el quemador; por otro lado se han

determinado las pérdidas térmicas por convección en la envolvente de la caldera, teniendo en cuenta la conductividad térmica de la envolvente de acero y del aislamiento. Como consecuencia, el simulador puede determinar de forma realista, además de los cálculos que ya se realizaban en la versión anterior.

El presente trabajo describe el desarrollo de las modificaciones del simulador a lo largo de 6 capítulos:

- Capítulo 1 ⟶ Introducción. Se describe los antecedentes del proyecto, las mejoras que se aplican para esta modificación, el funcionamiento del firmware con las nuevas implementaciones y el contenido del proyecto.
- Capítulo 2 ⟶ Fundamentación teórica. Se detalla la fundamentación teórica, en la cual está sustentada el proyecto, que contiene las citas y toda la información empleado para desarrollar los diferentes cálculos que contiene el proyecto.
- Capítulo 3 ⟶ Cálculos y ecuaciones. Se describen las diferentes ecuaciones y los pasos para obtener los diferentes resultados necesarios para la simulación.
- Capítulo 4 ⟶ Algoritmo de cálculo. Se explica el código empleado a implementar en el hardware. También se puede observar cómo se han integrado los cálculos del capítulo 3 en el código fuente.
- Capítulo 5 ⟶ Resultados. Se detallan los diferentes resultados obtenidos en la simulación, así como, los diferentes gráficos de la evolución de la caldera.
- Capítulo 6 ⟶ Discusión y conclusión. Se contrastan los resultados obtenidos y se desarrollan las conclusiones explicando algunos comportamientos, debatiendo resultados, líneas futuras y fallos durante el desarrollo .

2 Fundamentación teórica

2.1 Fueloil

El petróleo se compone básicamente de hidrocarburos, desde el metano hasta especies complejas como el fueloil $C_{20}H_{42}$ o superiores [9]. En el caso del fuel, su composición química consta de cadenas de 20 átomos o más de carbono. Cabe destacar que es el fueloil puede obtenerse por destilación bajo presión atmosférica, considerándose un residuo de este proceso [5, 9, 28, 6]. Igualmente, los fueles son de naturaleza viscosa por lo que es necesario romper y atomizar la masa de fuel en diminutas gotas que permitan la vaporización para facilitar su posterior combustión. Por otra parte, el fueloil es clasificado en diferentes grupos y estos son compuestos por: fueloil ligero, fueloil medio o fueloil pesado, dependiendo de la composición o mezcla con combustibles más ligeros.

Las calderas, exceptuando modelos de gas, utilizan fuel, ya que es un combustible económico, de gran seguridad y que tiene un fácil manejo. Asimismo, el fuel que tiene una gran viscosidad y debe reducirse para su uso. Además, en la preparación de este combustible se utilizan separadoras y calentadores que, mediante el calentamiento y la acción centrífuga de la separadora, consiguen limpiar y adecuar a condiciones óptimas el fuel.

El fuel utilizado por la caldera simulada en este trabajo, es un IFO 380 que tiene una temperatura de llama adiabática de 2102ºC, correspondiéndose a la máxima temperatura que alcanzará el fuel durante la operación del quemador [1, 28].

2.2 Combustión del fueloil

La combustión es una reacción de oxidación entre un combustible y un comburente, siendo necesaria una cierta energía de activación. Además, es siempre exotérmica [9]. La combustión se diferencia de otros procesos de oxidación lenta por resultar de una oxidación rápida con presencia de llama.

La combustión estequiométrica es aquella que se obtiene del uso de la cantidad mínima de comburente para una combustión completa. Además, cuenta con la peculiaridad de que para ser perfecta únicamente puede lograrse en un laboratorio. No obstante, la simulación, en un primer paso, calcula la energía de combustión suponiendo una reacción estequiométrica ecuación 2.1,mientras que en un segundo paso, se introduce exceso de aire para el cálculo del valor de la temperatura de llama [6, 19, 14, 25, 29], ver sección 3.2.5.

$$C_{20}H_{42} + \frac{61}{2}O_2 \rightarrow 20CO_2 + 21H_2O \tag{2.1}$$

En una combustión industrial, se emplea el contenido de oxígeno del aire como comburente. El aire está compuesto por un 21 por ciento de oxígeno, 78 por ciento de nitrógeno y 1 por ciento de dióxido de carbono y demás gases.

En la práctica, para lograr una combustión completa se necesita un exceso de aire por encima del mínimo teórico. No obstante, esta práctica conlleva ciertos problemas derivados de emplear más aire del necesario una combustión: en el mejor de los casos, el combustible se quemará totalmente sin producir inquemados. Pero a una temperatura inferior a la adiabática provoca un enfriamiento de los gases y extinción de la reacción. Hay que mencionar que, además, puede provocar humos de cierto color blanquecino que, en grandes concentraciones, es peligroso por su gran poder combustible debido a la presencia de combustible sin quemar. En definitiva, va a existir un derroche de combustible, la caldera no prestará su potencia máxima y genera posibles situaciones de peligro por explosiones. Sin embargo, trabajar sin un exceso de aire adecuado hace que la combustión sea incompleta, lo que puede originar que existan inquemados que se depositen en el hogar, y que consecuentemente, se produzca un riesgo de explosión por ignición retardada [6, 26].

2.2.1 Relación aire combustible

La relación aire combustible debe de mantenerse en un nivel tan bajo, tanto como sea posible con el fin de producir el mayor ahorro de combustible posible. Para poder garantizar una correcta combustión hay que introducir un exceso de aire en la mezcla de un 20 a un 30 por ciento, dependiendo del tipo de combustible utilizado, el estado de carga de la caldera y del tipo de quemador que se utilice [10]. Para un combustible con un determinado porcentaje en peso de carbono, hidrógeno y azufre es posible calcular el oxígeno necesario para oxidar cada elemento, como se muestra ecuación 2.2, pudiendo de esta forma calcular el aire mínimo necesario para quemar completamente una unidad del mismo [26, 14].

Para poder garantizar una combustión completa en un proceso industrial la mezcla combustible más comburente debe estar en continuo movimiento. No obstante, la mezcla anterior no es siempre homogénea y, teniendo en cuenta además que la velocidad de la reacción de combustión no es instantánea, así como el escaso tiempo de permanencia en el hogar y las variaciones de temperatura, existe el riesgo de que una fracción del combustible (β) escape sin quemarse [15, 14, 6, 9] como muestra la siguiente ecuación:

$$\begin{aligned}()\alpha+\beta)C_{20}H_{42}+\frac{61}{2}O_2+\frac{78}{21}\cdot\frac{61}{2}N_2+\frac{1}{21}\cdot\frac{61}{2}CO_2\rightarrow \\ 20\alpha CO_2+21\alpha H_2O+\beta C_{20}H_{42}+\frac{78}{21}\cdot\frac{61}{2}N_2+\frac{1}{21}\cdot\frac{61}{2}CO_2\end{aligned} \tag{2.2}$$

En la práctica, el exceso de aire de designa con la letra (n), de forma que (n>1) [9], tal y como se muestra a continuación:

$$C_{20}H_{42}+\frac{61}{2}\cdot nO_2+\frac{78}{21}\cdot\frac{61}{2}\cdot nN_2+\frac{1}{21}\cdot\frac{61}{2}\cdot nCO_2\rightarrow 20CO_2+21H_2O+\frac{78}{21}\cdot\frac{61}{2}\cdot nN_2+\frac{1}{21}\cdot\frac{61}{2}\cdot nCO_2 \tag{2.3}$$

La reducción y control de las emisiones de NOx es posible ajustarlas utilizando el exceso de aire mediante su variación para que actúe sobre la temperatura de combustión, manteniéndola por debajo de la temperatura de reacción del nitrógeno [15, 21]. No obstante, el simulador no tiene desarrollado el control de emisiones NOx y quedando como una de las posibles mejoras futuras a través del control de la temperatura de llama en el hogar.

2.2.2 Quemadores de combustibles líquidos

En el proceso de arranque de la caldera el quemador comienza realizando una purga que consiste en introducir una gran cantidad de aire al hogar durante unos minutos logrando

que los restos de inquemados, que puedan quedar dentro del hogar de la caldera, sean expulsados al exterior, evitando que puedan provocar explosiones durante el funcionamiento. A continuación, se empieza a inyectar el combustible mientras dos electrodos comienzan a producir chispas que provocan el encendido del quemador [6, 10].

Los quemadores cuentan con un sensor de tipo LDR como sistema de seguridad principal que están ubicados detrás de la cámara de combustión. Con este sistema de seguridad se consigue que si no hay suficiente intensidad lumínica de la llama, durante la combustión, el sensor cambia su resistencia interna obligando a la caldera a parar y a emitir una alarma ya que no existiría combustión [21].

En concreto, el simulador utiliza un modelo de quemador de tipo copa rotativa. También implementa en el firmware, durante el arranque inicial, el sistema de purga mencionado anteriormente.

2.3 Pérdidas Térmicas

La simulación original no disponía en su algoritmo de las pérdidas térmicas [20], porque era una caldera ideal.

En la nueva versión se han considerado estas pérdidas, calculando las correspondientes fugas térmicas a la envolvente de la caldera, el aislante y el calorifugado exterior. Estas pérdidas térmicas pertenecen a la siguiente clasificación por tipos de transferencias del calor [13, 27, 11, 22, 17]:

1. Convección.
2. Conducción.
3. Radiación.

2.3.1 Convección

La convección se produce únicamente durante la interacción de los fluidos. Dicha convección, expresada en la siguiente ecuación, es el transporte de energía calorífica por medio del movimiento del fluido, producido por la variación de la densidad de la masa de fluido en contacto con la fuente de calor [13, 30, 22, 17].

$$\frac{\Delta Q}{\Delta t} = h \cdot A_s \cdot (T_s - T_{inf}) \tag{2.4}$$

Donde:

- $Q \rightarrow$ Calor transmitido por unidad de tiempo.
- $h \rightarrow$ Coeficiente de convección.
- $A_s \rightarrow$ Área del cuerpo en contacto con el fluido.
- $T_s \rightarrow$ Temperatura de la superficie del cuerpo.
- $T_{inf} \rightarrow$ Temperatura del fluido lejano al cuerpo.

2.3.2 Conducción

La conducción de calor es un proceso de trasporte de energía calorífica a través de un cuerpo con temperatura mayor hacia otro con una temperatura menor.

Los materiales tienen una propiedad física que determinan su capacidad para conducir el calor. Esta propiedad se denomina conductividad térmica y se puede observar en la ecuación 2.5 como el coeficiente k.

$$\frac{Q}{\Delta T} = \frac{kA}{e} \cdot (T_2 - T_1) \tag{2.5}$$

A su vez, existe un propiedad, que es inversa a la anterior, denominada resistividad térmica r que es la capacidad del material a oponer resistencia al flujo de calor [13, 12, 22, 17].

$$\dot{q} = \frac{\Delta T}{R} \tag{2.6}$$

Donde:

- $Q \rightarrow$ Calor transmitido por unidad de tiempo.
- $k \rightarrow$ Conductividad térmica.
- $A \rightarrow$ Área de la superficie de contacto.
- $e \rightarrow$ Espesor del material.
- $(T_2 - T_1) \rightarrow$ Diferencia de temperatura entre el foco caliente y el frío.
- $R \rightarrow$ Resistividad térmica en $\frac{W}{mK}$.
- $\dot{q} \rightarrow$ Flujo de calor en $\frac{W}{m^2}$

2.3.3 Radiación

La radiación térmica es la capacidad que tiene un cuerpo de emitir calor por la radiación electromagnética. Esta radiación es generada por el movimiento térmico que tienen las partículas cargadas en la materia. Todos los cuerpos, salvo que la temperatura sea de cero absoluto, emiten radiación electromagnética y su intensidad es totalmente dependiente de la temperatura a la que se encuentra. A temperatura ambiente, todos los cuerpos son emisores de radiación infrarroja. Sin embargo, a temperaturas más altas, esto cambia, pasando de emitir radiación infrarroja a radiación de espectro visible, lo que permite que se pueda determinar la temperatura del cuerpo de acuerdo al color que este emita [13, 22, 17].

2.3.4 Aislamiento térmico

Para disminuir las perdidas energéticas convección, conducción y radiación, a través de la envolvente de la caldera, se utiliza el aislamiento térmico, que consiste en la instalación de diversos materiales a los que se les aplica ciertas técnicas de forma que minimizan la transferencia del flujo de calor [29]. Estos materiales deben cumplir ciertos requisitos, debiendo tener una baja conductividad térmica o una resistividad térmica alta que minimice la transferencia de calor [22, 17].

Los aislantes térmicos se pueden clasificar en cuatro tipos de materiales caracterizados por su baja conductividad térmica:

- Minerales: consiste en un material de filamentos entrelazados de materiales pétreos que forman un tejido que mantienen el aire contenido entre sus fibras en estado inmóvil ofreciendo buenos resultados térmicos y acústicos [23].
- Celulares: son los aislantes que conforman celdas cerradas o abiertas, en su estructura, que por lo general van formando una especie de tableros rígidos o flexibles pero también se pueden conformar por proyección o riego [23].
- Granulares: son materiales inorgánicos que se aglomeran en pequeñas partículas respetando una forma prefabricada o utilizadas y sueltas como la perlita o la vermiculita [23].
- Orgánicos: hacen referencia a los materiales aglomerados de naturaleza orgánica como puede ser el corcho natural [23].

En función del tipo de uso o aplicación que se le de al material se determina un tipo de aislante dependiendo de la resistencia térmica, de las propiedades mecánicas, la absorción de agua, la temperatura de trabajo, el comportamiento físico, la estabilidad, etc. [22, 17, 23].

En la simulación se ha optado por lana de roca de 100 mm de espesor, con una conductividad de 0,0410 (W/m·K) [23].

2.4 Aspectos teóricos de los intercambiadores de vapor

2.4.1 Coeficiente térmico ∪

La transmitancia térmica U es la medida del calor que fluye por unidad de tiempo y superficie, transferido a través de un sistema formado por una o más capas de material, de caras plano paralelas, cuando hay un gradiente térmico de 1°C de temperatura entre los dos ambientes que éste separa.

Es medida en la unidad W/m^2K. Su valor incluye las resistencias térmicas superficiales de las caras del intercambiador, es decir, refleja la capacidad de trasmitir calor. Cuanto menor sea el valor U, menor será el paso de la energía entre ambas caras, y por tanto mejoran las capacidades aislantes del elemento constructivo [13]. Se puede calcular el coeficiente ∪ con la ecuación 3.30.

2.4.2 Temperatura media logarítmica

La temperatura media logarítmica determina la transferencia de calor en sistemas de flujo, como es un intercambiador de calor. Es un método que analiza la temperatura del fluido frío y del fluido caliente, teniendo como un máximo de temperatura la temperatura del fluido caliente y como un mínimo la del fluido frío [13]. Y se puede calcular con la ecuación 3.31.

3 Cálculos y ecuaciones

3.1 Cálculo de la superficie total de intercambio de la caldera

La caldera, de construcción pirotubular, se modeló como un intercambiador aire-agua de grandes dimensiones que transfiere la energía calórica, contenida en los gases de combustión, hacia el agua, que se encuentra en el interior de la caldera, para producir vapor.

Mediante una imagen de construcción, figura 3.1, que se encuentra en la documentación de la caldera, se observa el número de tubos, forma y número de pasos. Sabiendo las medidas reales e interpolando con el dibujo impreso de la caldera, se pueden aproximar las dimensiones de los tubos, su longitud, diámetro y demás características.

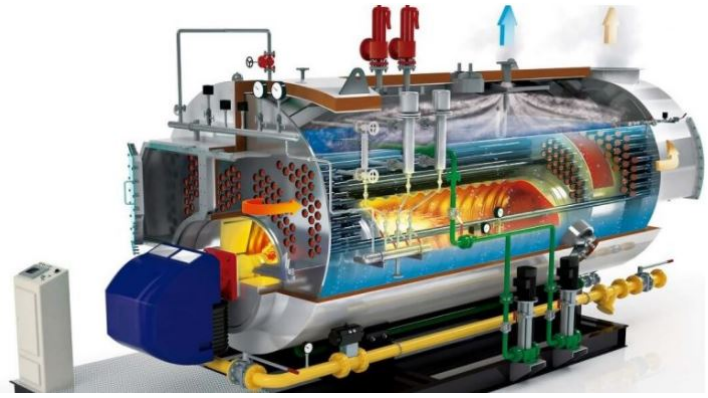

Figura 3.1: Caldera de tipo pirotubular.
Fuente: Consultoría de tratamiento de aguas de caldera [2].

Para calcular la superficie de intercambio de la caldera se calculan los tubos de misma longitud correspondientes a los de los diferentes pasos de humo de la caldera y se suman teniendo en cuenta el número de unidades que hay. Para ello, es necesario hallar el área de cada cilindro.

Este mismo cálculo se utiliza para todos los pasos de humos y para el hogar de la caldera, variando únicamente la longitud según el paso de humo o la longitud del hogar de la caldera.

Calculadas las áreas de todos los tubos, se suman el área de cada uno de los pasos de la caldera, además del hogar, según la ecuación 3.1. Si en lugar de un tubo liso, hubiera sido el caso de un tubo corrugado se aplicaría un coeficiente para realizar el cálculo con dicha característica.

$$A_{caldera} = (A_{hogar} = 2 \cdot \pi \cdot R \cdot L) + (A_{tubos\ paso1} = 2 \cdot \pi \cdot R \cdot L \cdot n) + (A_{tubos\ paso2} = 2 \cdot \pi \cdot R \cdot L \cdot n) \tag{3.1}$$

3.2 Cálculo de la temperatura del hogar

3.2.1 Peso molecular del combustible

En esta sección se detalla el método para el cálculo del peso molecular empleado en la caldera. Para este cálculo, se separan los átomos de cada molécula de manera que se pueda sumar el número de átomos por el producto de su peso atómico. (Véase la ecuación (3.2) para el combustible utilizado).

Elemento	Peso atómico
Hidrógeno	1.008
Oxígeno	16
Carbono	12

Tabla 3.1: Tabla de pesos atómicos.
Fuente: Datos extraídos del ministerio de agricultura y pesca [7].

$$\begin{aligned} C_{20} &= 12 \cdot 20 = 240 gr/Mol; \\ H_{42} &= 42 \cdot 1{,}008 gr/Mol; \\ C_{20}H_{42} &= 232,336\ gr/Mol \end{aligned} \tag{3.2}$$

Este procedimiento se realiza para todos los componentes de la reacción que son: el CO_2,el O_2, el H_2O y el N_2. Con ello se obtiene el peso atómico de cada elemento que es introducido de manera natural en la mezcla de aire-combustible.

3.2.2 Ecuación química del combustible

En el cálculo de la temperatura del hogar es necesaria la ecuación química del producto que va a quemar la caldera. En este caso, el fuel IFO 380 se compone mayoritariamente por cadenas de átomos de 20 carbonos y también se deben conocer los elementos que participan en la mezcla combustible-comburente. En el hogar entra el combustible ($C_{20}H_{42}$) y aire ($21\,\%O_2, 78\,\%N_2, 1\,\%CO_2$), pero en la ecuación química solo se tienen en cuenta aquellas sustancias que experimentan cambios químicos como se define en la siguiente expresión matemática:

$$C_{20}H_{42} + O_2 \rightarrow CO_2 + H_2O \tag{3.3}$$

Para hacer los cálculos de la cantidad necesaria de O_2 y de las cantidades formadas de los productos de la combustión (CO_2 y H_2O) se realizó el ajuste de la ecuación química introduciendo los coeficientes correspondientes a las moléculas de oxígeno necesarias para quemar una molécula de fuel, tal y como se muestra a continuación:

$$C_{20}H_{42} + \frac{61}{2}O_2 \rightarrow 20CO_2 + 21H_2O \tag{3.4}$$

Sin embargo, para poder hacer los balances de masa y energía (Entrada+Generación = Salida+Acumulación) se tuvieron en cuenta, no solo los reactivos sino todas las sustancias

que entran, salen o se acumulan en el hogar, aunque estas no participen activamente en el proceso de combustión, pero pueden experimentar cambios en su temperatura, como se expone a continuación:

$$C_{20}H_{42} + \frac{61}{2}O_2 + (N_2) + (CO_2) \rightarrow 20CO_2 + 21H_2O + (N_2) + (CO_2) \tag{3.5}$$

La ecuación anterior trabaja con unas cantidades estequiométricas de aire, es decir, con el mínimo requerido para la combustión completa del combustible, por lo que se hicieron correcciones para trabajar con un exceso de aire. Así por cada 21 moles de O_2 que entran le acompañan 78 de N_2 y 1 de CO_2. Así cada mol de O_2 le acompañan $\frac{78}{21}$ de N_2 y $\frac{1}{21}$ de CO_2. Como se sabe que entran $\frac{61}{2}$ moles de O_2, quedando las cantidades de la siguiente forma:

$$(N_2) : \frac{78}{21} \cdot \frac{61}{2} N_2 \tag{3.6}$$

$$(CO_2) : \frac{1}{21} \cdot \frac{61}{2} CO_2 \tag{3.7}$$

Por consiguiente, la ecuación queda así:

$$C_{20}H_{42} + \frac{61}{2}O_2 + \frac{78}{21} \cdot \frac{61}{2} N_2 + \frac{1}{21} \cdot \frac{61}{2} CO_2 \rightarrow 20CO_2 + 21H_2O + \frac{78}{21} \cdot \frac{61}{2} N_2 + \frac{1}{21} \cdot \frac{61}{2} CO_2 \tag{3.8}$$

En este caso entran y salen todo el N_2 y el CO_2 que acompañan al O_2 porque no se produce una transformación en el hogar, así que solo hay que tenerlos en cuenta si sufren cambios de temperatura para el balance energético.

Si se trabaja con exceso de aire, que es lo habitual, este se define como:

$$n \rightarrow \frac{Aire_{introducido}}{Aire_{minimo}} \rightarrow \frac{O_{2\,introducido}}{O_{2\,minimo}} \tag{3.9}$$

En este caso el aire mínimo y el oxígeno mínimo se refieren a los valores estequiométricos. Este índice se relaciona con el porcentaje de exceso de aire mediante: $\%_{exceso} = (n-1) \cdot 100$. Con este índice se corrigen los siguientes valores:

- Los valores de entrada y de salida de los componentes del aire, se ven afectados por n: $O_{2introducido} = n \cdot O_{2minimo}$
- Siendo el siguiente el oxígeno sobrante: $O_{2sobrante} = O_{2intoducido} - O_{2minimo} = n \cdot O_{2minimo} - O_{2minimo} = (n-1) \cdot O_{2minimo}$

La ecuación 3.10 muestra la ecuación de combustión de fuel incluyendo el exceso de aire. Si el exceso de aire toma el valor n=1 esta ecuación queda reducida a la 3.8

$$C_{20}H_{42} + \frac{61}{2} \cdot nO_2 + \frac{78}{21} \cdot \frac{61}{2} \cdot nN_2 + \frac{1}{21} \cdot \frac{61}{2} \cdot nCO_2 \rightarrow 20CO_2 + 21H_2O + \frac{78}{21} \cdot \frac{61}{2} \cdot nN_2 + \frac{1}{21} \cdot \frac{61}{2} \cdot nCO_2 \tag{3.10}$$

3.2.3 Cálculo del caudal de combustible

En la simulación, como el estado de carga está determinado por un valor expresado en tanto por ciento, es necesario calcular el caudal de combustible para poder calcular la potencia del quemador.

En la expresión 3.11, obtenida por Amell-Arrieta [8], permite calcular el caudal de combustible.

$$\dot{m}_{combustible} = \frac{P_{quemador} \cdot (\,\%\, carga\ vapor)}{PCI} \tag{3.11}$$

3.2.4 Determinación de la masa de oxígeno estequiométrico

Para poder emplear la ecuación química 3.10 es necesario realizar la conversión del caudal másico de fuel a flujo de moles de fuel como se muestra a continuación.

$$\dot{M}_{fuel} = \frac{\dot{m}_{fuel}}{Pa_{fuel}} \tag{3.12}$$

Donde:

- $\dot{M}_{fuel} \rightarrow$ Flujo moles de fuel.
- $\dot{m}_{fuel} \rightarrow$ Flujo másico de fuel.
- $Pa_{fuel} \rightarrow$ Peso atómico del fuel en (kg/mol).

Partiendo del resultado de la ecuación 3.12 se calcula el flujo de moles de oxígeno que debe acompañar al fuel para una combustión estequiométrica de la siguiente manera:

$$\dot{M}O_{2\,fuel} = \dot{M}_{fuel} \cdot MO_2 \tag{3.13}$$

Donde:

- $\dot{M}O_{2\,fuel} \rightarrow$ Flujo moles de oxígeno.
- $\dot{M}_{fuel} \rightarrow$ Flujo moles de fuel.
- $MO_2 \rightarrow$ Moles de oxígeno para una molécula de fuel.

Para trasladar este resultado a un caudal en kg/s se necesita realizar la siguiente operación:

$$\dot{m}O_{2\,fuel} = \frac{\dot{M}O_{2\,fuel} \cdot (PmO_2 \cdot 2)}{1000} \tag{3.14}$$

Donde:

- $\dot{m}O_{2\,fuel} \rightarrow$ Caudal de oxígeno para el caudal de fuel.
- $\dot{M}O_{2\,fuel} \rightarrow$ Flujo de moles de oxígeno.
- $PmO_2 \rightarrow$ Peso molecular del oxígeno en (g/mol).

3.2.5 Cálculo del índice de exceso de aire

Para determinar el índice de exceso de aire de la caldera es necesario saber que tipo de quemador se está trabajando y, según sus especificaciones, variará de un porcentaje a otro. En este caso se utilizará un 30 % de exceso de aire ya que es un quemador de tipo copa rotativa y es su porcentaje habitual de trabajo [26]. A través de la siguiente ecuación se obtiene el índice de exceso:

$$I_e = \frac{\%exceso}{100} + 1 \tag{3.15}$$

Donde:

- $I_e \rightarrow$ Índice de exceso.
- $\%exceso \rightarrow$ Porcentaje de exceso de aire.

3.2.6 Cálculo de la temperatura del hogar

Para calcular la temperatura del hogar en este apartado se tienen en cuenta estas consideraciones iniciales:

- No intervienen los gases de entrada que se emplean en la reacción completa.
- El $CO_2 + H_2O$ provenientes de la combustión están inicialmente a la temperatura adiabática y ceden energía al exceso de gases.

Con esta consideración y sabiendo que los gases resultantes de la combustión ceden calor a los gases que se introducen con el exceso se cumple en la siguiente ecuación:

$$Q_{combustion} = Q_{gases\ de\ exceso} \tag{3.16}$$

Donde:

- $Q_{combustion} \rightarrow$ Calor de combustión cedido.
- $Q_{gasesdeexceso} \rightarrow$ Calor absorbido por los gases de exceso.

Si se sustituye en cada término los gases que intervienen en cada aporte energético queda una ecuación como la siguiente:

$$Q_{CO_2+H_2O} = Q_{N_2+O_2+CO_2} \tag{3.17}$$

Donde:

- $Q_{CO_2+H_2O} \rightarrow$ Calor de los gases generado en la combustión.
- $Q_{N_2+O_2+CO_2} \rightarrow$ Calor absorbido por los gases de exceso.

Para el cálculo de la temperatura del hogar se debe comenzar formando la ecuación con la fórmula de la transferencia energética de cada gas que cede calor, para ello, se emplea la siguiente fórmula con la que se calcula la cantidad de calor necesaria para posteriormente hallar la temperatura del hogar:

$$Q = m \cdot Ce \cdot \Delta T \tag{3.18}$$

Donde:

- $Q \rightarrow$ Cantidad de calor.
- $m \rightarrow$ Masa.
- $Ce \rightarrow$ Calor específico del fluido.
- $\Delta T \rightarrow$ Diferencia de temperatura.

Las diferencias de temperatura son resultantes de restar la temperatura más alta menos la mas baja. Para ello, es necesario conocer entre qué puntos es necesario tomar nuestras temperaturas para efectuar una diferencia de temperatura correctamente. Debido a que en el firmware se ha implementado las ecuaciones de cantidad de calor de los gases de combustión y las pérdidas térmicas en una sola ecuación y esta a su vez se va a multiplicar por un diferencial de tiempo se emplea la potencia calorífica en vez de la energía o calor. La ecuación 3.19 toma el vapor de agua para realizar el cálculo, aunque también es válida para el dióxido de carbono como se ve en 3.20. La diferencia de temperatura ya mencionada, se corresponde con la diferencia de la temperatura entre la llama adiabática en el momento de la combustión (temperatura máxima de la combustión) y la temperatura que resulta una vez la masa de gases finaliza la combustión (temperatura del hogar).

$$P_{H_2O} = \dot{m}_{H_2O} \cdot Ce_{H_2O} \cdot (T_{adb} - T_h) \tag{3.19}$$

Donde:

- $P_{H_2O} \rightarrow$ Potencia calorífica contenido en el vapor de agua.
- $\dot{m}_{H_2O} \rightarrow$ Caudal másico del vapor de agua.
- $Ce_{H_2O} \rightarrow$ Calor específico del vapor de agua.
- $T_{adb} \rightarrow$ Temperatura adiabática del combustible.
- $T_h \rightarrow$ Temperatura del hogar de la caldera.

Para el CO_2 se muestra la misma ecuación pero con los valores del CO_2:

$$P_{CO_2} = \dot{m}_{CO_2} \cdot Ce_{CO_2} \cdot (T_{adb} - T_h) \tag{3.20}$$

Donde:

- $P_{CO_2} \rightarrow$ Potencia calorífica contenida en el dióxido de carbono.
- $\dot{m}_{CO_2} \rightarrow$ Caudal másico del dióxido de carbono.
- $Ce_{CO_2} \rightarrow$ Calor específico del dióxido de carbono.
- $T_{adb} \rightarrow$ Temperatura adiabática del combustible.
- $T_h \rightarrow$ Temperatura del hogar de la caldera.

A continuación, se muestran las ecuaciones de cantidad de calor de los gases que no intervienen en la combustión y que absorben al calentarse.

$$P_{N_2} = \dot{m}_{N_2} \cdot Ce_{N_2} \cdot (T_h - T_{amb}) \tag{3.21}$$

Donde:

- $P_{N_2} \rightarrow$ Potencia calorífica contenida en el nitrógeno.
- $\dot{m}_{N_2} \rightarrow$ Caudal másico del nitrógeno.
- $Ce_{N_2} \rightarrow$ Calor específico del nitrógeno.
- $T_{amb} \rightarrow$ Temperatura ambiente o de entrada de gases de admisión.
- $T_h \rightarrow$ Temperatura del hogar de la caldera.

Para el O_2 se muestra la misma ecuación pero con los valores del O_2:

$$P_{O_2} = \dot{m}_{O_2} \cdot Ce_{O_2} \cdot (T_h - T_{amb}) \tag{3.22}$$

Donde:

- $P_{O_2} \rightarrow$ Potencia calorífica contenida en el oxígeno.
- $\dot{m}_{O_2} \rightarrow$ Caudal másico del oxígeno.
- $Ce_{O_2} \rightarrow$ Calor específico del oxígeno.
- $T_{amb} \rightarrow$ Temperatura ambiente o de entrada de gases de admisión.
- $T_h \rightarrow$ Temperatura del hogar.

Para el CO_2 se muestra la misma ecuación pero con los valores del CO_2:

$$P_{CO_2} = \dot{m}_{CO_2} \cdot Ce_{CO_2} \cdot (T_h - T_{amb}) \tag{3.23}$$

Donde:

- $P_{CO_2} \rightarrow$ Potencia calorífica contenida en el dióxido de carbono.
- $\dot{m}_{CO_2} \rightarrow$ Caudal másico del dióxido de carbono.
- $Ce_{CO_2} \rightarrow$ Calor específico del dióxido de carbono.
- $T_{amb} \rightarrow$ Temperatura ambiente o de entrada de gases de admisión.
- $T_h \rightarrow$ Temperatura del hogar.

En la siguiente subsección se emplean todas estas ecuaciones 3.19, 3.20, 3.21, 3.22, 3.23 para formular la ecuación de transferencia energética total con la que se puede calcular la temperatura que tiene el hogar de la caldera.

3.2.6.1 Ecuación de transferencia energética

La siguiente expresión matemática se basa en la aplicación a la ecuación 3.16.

A continuación, se muestra el resultado general de incorporar las diferentes variables que ceden o absorben calor después del proceso de combustión, expresado en forma de potencia.

$$\begin{aligned} \dot{m}_{CO_2} \cdot Ce_{CO_2} \cdot (T_{adb} - T_h) + \dot{m}_{H_2O} \cdot Ce_{H_2O} \cdot (T_{adb} - T_h) = \\ \dot{m}_{N_2} \cdot Ce_{N_2} \cdot (T_h - T_{amb}) + \dot{m}_{O_2} \cdot Ce_{O_2} \cdot (T_h - T_{amb}) + \dot{m}_{CO_2} \cdot Ce_{CO_2} \cdot (T_h - T_{amb}) \end{aligned} \tag{3.24}$$

Si se saca el factor común a $(T_{adb} - T_h)$ y $(T_h - T_{amb})$ la ecuación queda de la siguiente manera:

$$\begin{gathered}(T_{adb} - T_h) \cdot (\dot{m}_{CO_2} \cdot Ce_{CO_2} + \dot{m}_{H_2O} \cdot Ce_{H_2O}) = \\ (T_h - T_{amb}) \cdot (\dot{m}_{N_2} \cdot Ce_{N_2} + \dot{m}_{O_2} \cdot Ce_{O_2} + \dot{m}_{CO_2} \cdot Ce_{CO_2})\end{gathered} \tag{3.25}$$

En este caso se pasa al lado izquierdo las temperaturas y al derecho lo restante quedando de la siguiente forma:

$$\frac{(T_{adb} - T_h)}{(T_h - T_{amb})} = \frac{\dot{m}_{N_2} \cdot Ce_{N_2} + \dot{m}_{O_2} \cdot Ce_{O_2} + \dot{m}_{CO_2} \cdot Ce_{CO_2}}{\dot{m}_{CO_2} \cdot Ce_{CO_2} + \dot{m}_{H_2O} \cdot Ce_{H_2O}} \tag{3.26}$$

Llegados a este punto, se deja en el lado izquierdo la temperatura del hogar y lo restante en el lado derecho como se ve a continuación:

$$T_h = \frac{T_{amb} \cdot (\frac{\dot{m}_{N_2} \cdot Ce_{N_2} + \dot{m}_{O_2} \cdot Ce_{O_2} + \dot{m}_{CO_2} \cdot Ce_{CO_2}}{\dot{m}_{CO_2} \cdot Ce_{CO_2} + \dot{m}_{H_2O} \cdot Ce_{H_2O}}) + T_{adb}}{\frac{\dot{m}_{N_2} \cdot Ce_{N_2} + \dot{m}_{O_2} \cdot Ce_{O_2} + \dot{m}_{H_2O} \cdot Ce_{H_2O} + \dot{m}_{CO_2} \cdot Ce_{CO_2} + \dot{m}_{CO_2} \cdot Ce_{CO_2}}{\dot{m}_{CO_2} \cdot Ce_{CO_2} + \dot{m}_{H_2O} \cdot Ce_{H_2O}}} \tag{3.27}$$

Operando la ecuación se obtiene esta expresión más sencilla:

$$T_h = \frac{T_{adb} \cdot (\dot{m}_{CO_2} \cdot Ce_{CO_2} + \dot{m}_{H_2O} \cdot Ce_{H_2O}) + T_{amb} \cdot (\dot{m}_{N_2} \cdot Ce_{N_2} + \dot{m}_{O_2} \cdot Ce_{O_2} + \dot{m}_{CO_2} \cdot Ce_{CO_2})}{\dot{m}_{N_2} \cdot Ce_{N_2} + \dot{m}_{O_2} \cdot Ce_{O_2} + \dot{m}_{H_2O} \cdot Ce_{H_2O} + \dot{m}_{CO_2} \cdot Ce_{CO_2} + \dot{m}_{CO_2} \cdot Ce_{CO_2}} \tag{3.28}$$

Donde:

- $\dot{m}_{CO_2} \rightarrow$ Caudal másico del dióxido de carbono.
- $Ce_{CO_2} \rightarrow$ Calor específico del dióxido de carbono.
- $\dot{m}_{H_2O} \rightarrow$ Caudal másico del vapor de agua.
- $Ce_{H_2O} \rightarrow$ Calor específico del vapor de agua.
- $\dot{m}_{N_2} \rightarrow$ Caudal másico del nitrógeno.
- $Ce_{N_2} \rightarrow$ Calor específico del nitrógeno.
- $\dot{m}_{O_2} \rightarrow$ Caudal másico del oxígeno.
- $Ce_{O_2} \rightarrow$ Calor específico del oxígeno.
- $T_{amb} \rightarrow$ Temperatura ambiente o de entrada de gases de admisión.
- $T_{adb} \rightarrow$ Temperatura adiabática del combustible.

Llegado a este punto se puede calcular la temperatura del hogar T_h sin dificultad.

3.3 Ecuación de eficiencia según la carga de vapor

Las calderas, como muchas otras máquinas térmicas, nunca trabajan en un estado de carga fijo. El rendimiento de la caldera varía según el estado de carga al que esté sometida durante el funcionamiento siendo mayor este rendimiento a mayor carga de vapor.

3.3.1 Obtención de la ecuación de la eficiencia

El simulador basa la variación de su rendimiento en función del comportamiento de la caldera como se observa en la siguiente gráfica obtenida de Rodríguez (2015)[27]:

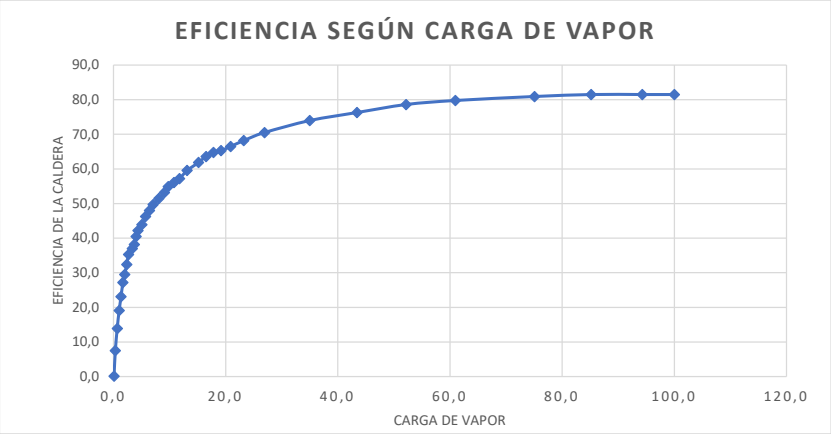

Figura 3.2: Comportamiento de la eficiencia con respecto a la carga.
Fuente: Gráfica Reproducida de Rodríguez (2015) [27].

El análisis de la curva muestra que la disminución de la eficiencia es una tendencia marcada en los generadores de vapor a medida que se ve reducido el porcentaje de carga. Este factor, que es debido a la utilización de la caldera, se hace más evidente cuando se trabaja con valores inferiores al 50 %.

Inicialmente, con ayuda del ordenador y el software "Get Data Graphics"se extrae una tabla de datos de los puntos que se han situado sobre la curva. A continuación, se muestra un ejemplo gráfico donde se ve el proceso:

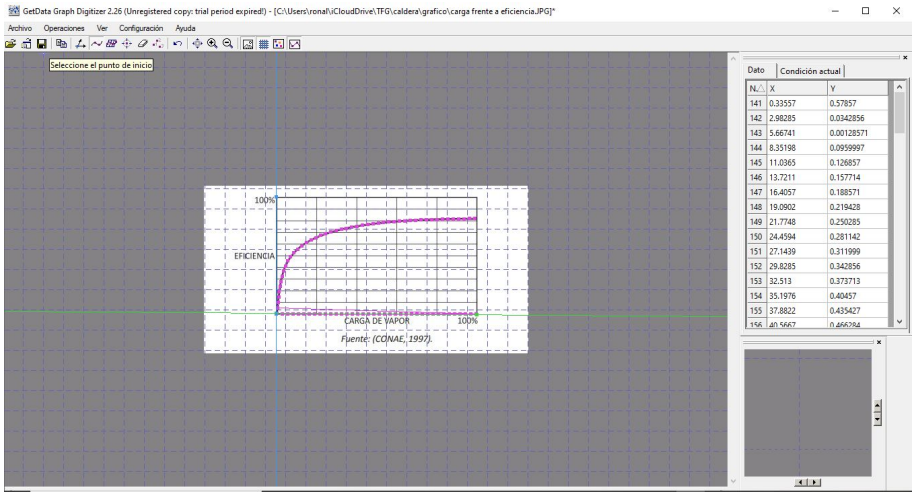

Figura 3.3: Obtención de la tabla de valores mediante el software Get Data.
Fuente: Elaboración propia.

Una vez obtenida la tabla de valores se hace uso de una hoja de cálculo para generar una gráfica con la que se obtenga la ecuación de la línea de tendencia.

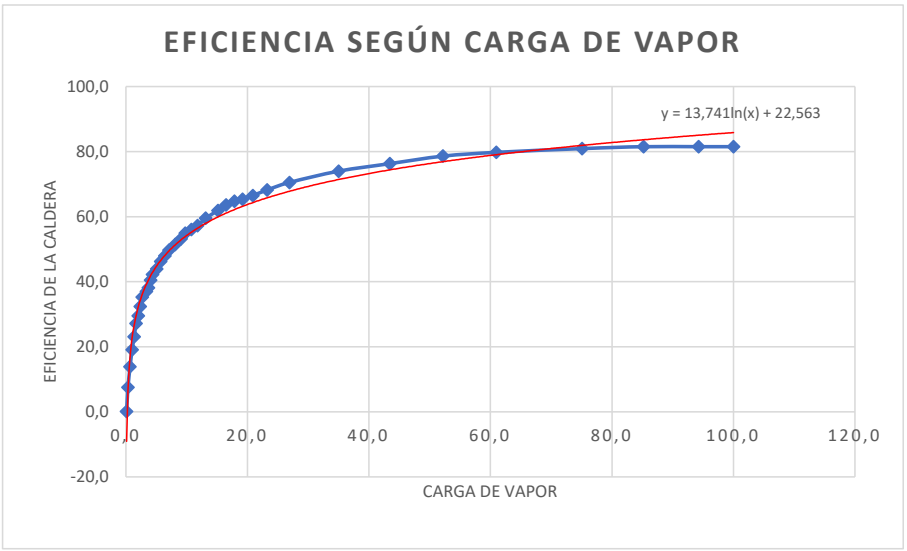

Figura 3.4: Obtención de la línea de tendencia y su ecuación mediante excel.
Fuente: Elaboración propia.

Gracias a la función de línea de tendencia se puede ejecutar la opción de que devuelva una ecuación aproximada a esa línea quedando como resultante la ecuación:

$$Eff = 13,741 \ln(C_v) + 22,563 \tag{3.29}$$

Donde:

- $Eff \rightarrow$ Eficiencia de la caldera según la carga en %.
- $C_v \rightarrow$ Carga de vapor en %.

3.4 Cálculo de la temperatura de gases de escape

Esta sección describe la obtención de la temperatura de los gases de escape. Para ello se hace uso del cálculo de la temperatura media logarítmica, reduciendo la caldera a un intercambiador de calor.

3.4.1 Cálculo del coeficiente térmico ∪

El cálculo del coeficiente térmico del intercambiador,requiere seleccionar los valores de conductividad para los gases y para el agua según el estado en el que se encontrarán dentro de la caldera. Estos valores son obtenidos del archivo Universitat Politècnica de València [**?**]:

- $K_{gas} \rightarrow 280 \frac{W}{m^{2\circ}C}$
- $K_{agua} \rightarrow 20000 \frac{W}{m^{2\circ}C}$

Seguidamente se puede calcular el coeficiente ∪ de la siguiente manera cenguel, 2011[13]:

$$\cup = \frac{1}{\frac{1}{K_{gas}} \cdot \frac{1}{K_{agua}}} \tag{3.30}$$

Donde:

- $\cup \rightarrow$ Coeficiente conductividad térmica global en $\frac{W}{m^{2\circ}C}$.
- $K_{gas} \rightarrow$ Coeficiente conductivo gases de escape.
- $K_{agua} \rightarrow$ Coeficiente conductivo agua.

3.4.2 Cálculo de la temperatura media logarítmica real

Se emplea la siguiente ecuación para el cálculo de la temperatura media logarítmica real:

$$L.T.M = \frac{P_{cald}}{SI \cdot \cup} \tag{3.31}$$

Donde:

- $L.T.M. \rightarrow$ Temperatura media logarítmica.
- $P_{cald} \rightarrow$ Potencia de la caldera en kW.
- $S_I \rightarrow$ Superficie total de intercambio.
- $\cup \rightarrow$ Coeficiente de transmisión del intercambiador en $\frac{kW}{m^{2\circ}C}$

3.4.3 Temperatura de gases de escape mediante el cálculo iterativo

El cálculo de la temperatura de los gases de escape no puede hacerse de forma directa, sino mediante cálculo iterativo partiendo de un valor estimado inicial, empleando para ello la siguiente ecuación:

$$L.T.M_I = \frac{(T_h - B_w) - (E_t - F_w)}{\ln((T_h - B_w) - (E_t - F_w))} \tag{3.32}$$

Donde:

- $L.T.M._I \rightarrow$ Temperatura media logarítmica iterada.
- $T_h \rightarrow$ Temperatura del hogar de la caldera.
- $B_w \rightarrow$ Temperatura del agua de caldera.
- $E_t \rightarrow$ Temperatura de gases de escape.
- $F_w \rightarrow$ Temperatura de agua de alimentación.

Esta operación se realiza tantas veces como sea necesaria hasta que el valor de la temperatura media logarítmica iterada sea aproximadamente igual a la temperatura media logarítmica real con la precisión que se requiera. Este cálculo se realiza en el simulador con la ayuda de un bucle tipo do-while programado de forma que en cuanto se alcance la precisión requerida se detiene continuando con el resto del cálculo.

3.5 Pérdida térmicas a través del aislamiento

El cálculo de la transferencia de calor que se fuga a través del aislante térmico que contiene la caldera y requiere de su descomposición geométrica en una superficie lateral de un cilindro y en dos tapas circulares. El empleo de las ecuaciones 3.33 3.34 3.35 3.36 permiten calcular la pérdida térmica total.

3.5.1 Cálculo de pérdidas térmicas en la superficie lateral de la caldera

La transferencia de calor estacionaria a través de estas capas de aislamiento se pueden tratar como si fuese una pared plana multicapa por la que va a fluir el calor calculado mediante la siguiente ecuación:

$$P_{SC} = \frac{T_{inf1} - T_{inf2}}{R_{ENV}} \tag{3.33}$$

Donde:

- $P_{SC} \rightarrow$ Potencia térmica cedida por la superficie cilíndrica.
- $T_{inf1} \rightarrow$ Temperatura del agua en un punto lejano de la superficie de intercambio.
- $T_{inf2} \rightarrow$ Temperatura del aire en un punto lejano de la superficie de intercambio.

- $R_{ENV} \rightarrow$ Resistencia térmica de la superficie de la envolvente.

Véase que la R_{ENV} contiene la resistencia de las múltiples capas, por lo que, se tiene en consideración los materiales que componen el conglomerado del cilindro.

$$\begin{aligned} R_{ENV} &= R_{conv1} + R_{cil1} + R_{cil2} + R_{cil3} + R_{conv2} \\ &= \frac{1}{h_1 A_1} + \frac{ln(\frac{r_2}{r_1})}{2\pi L k_1} + \frac{ln(\frac{r_3}{r_2})}{2\pi L k_2} + \frac{ln(\frac{r_4}{r_3})}{2\pi L k_3} + \frac{1}{h_2 A_4} \end{aligned} \tag{3.34}$$

Donde:

- $R_{ENV} \rightarrow$ Resistencia térmica de la superficie de la envolvente.
- $R_{conv1} \rightarrow$ Resistencia convectiva de agua.
- $R_{cil1} \rightarrow$ Resistencia cilíndrica de la envolvente.
- $R_{cil2} \rightarrow$ Resistencia cilíndrica del aislante.
- $R_{cil3} \rightarrow$ Resistencia cilíndrica del calorifugado.
- $R_{conv2} \rightarrow$ Resistencia convectiva del aire.
- $R_1 \rightarrow$ Radio interno de la envolvente.
- $R_2 \rightarrow$ Radio externo de la envolvente.
- $R_3 \rightarrow$ Radio interno de la calorifugado.
- $R_4 \rightarrow$ Radio externo de la calorifugado.
- $L \rightarrow$ Longitud de la envolvente.
- $k_1 \rightarrow$ Conductividad térmica de la envolvente.
- $k_2 \rightarrow$ Conductividad térmica de la aislante.
- $k_3 \rightarrow$ Conductividad térmica de la calorifugado.
- $h_1 \rightarrow$ Coeficiente convectivo del agua.
- $h_2 \rightarrow$ Coeficiente convectivo del aire.
- $A_1 \rightarrow$ Área de transferencia en contacto con el agua.
- $A_2 \rightarrow$ Área de transferencia en contacto con el aire.

3.5.2 Cálculo de pérdidas térmicas en las tapas circulares de la caldera

Para las tapas se usa el flujo de calor por paredes planas quedando las ecuaciones de manera muy similar a las del apartado anterior. Para el flujo de calor se utiliza la ecuación:

$$P_{SP} = \frac{T_{inf1} - T_{inf2}}{R_{TP}} \tag{3.35}$$

Donde:

- $P_{SP} \rightarrow$ Potencia térmica cedida por la tapa de la envolvente.
- $T_{inf1} \rightarrow$ Temperatura del agua en un punto lejano de la superficie de intercambio.
- $T_{inf2} \rightarrow$ Temperatura del aire en un punto lejano de la superficie de intercambio..
- $R_{TP} \rightarrow$ Resistencia térmica de la tapa de la envolvente.

En cambio, para la resistencia térmica total la ecuación expresada si que varía quedando de la siguiente manera:

$$\begin{aligned} R_{TP} &= R_{conv1} + R_{pared1} + R_{pared2} + R_{pared3} + R_{conv2} \\ &= \frac{1}{h_1 A_{SP}} + \frac{L_1}{A_{SP} k_1} + \frac{L_2}{A_{SP} k_2} + \frac{L_3}{A_{SP} k_3} + \frac{1}{h_2 A_{SP}} \end{aligned} \tag{3.36}$$

Donde:

- $R_{TP} \rightarrow$ Resistencia térmica de la tapa de la envolvente.
- $R_{conv1} \rightarrow$ Resistencia convectiva generada por el agua.
- $R_{pared1} \rightarrow$ Resistencia de la tapa envolvente.
- $R_{pared2} \rightarrow$ Resistencia del aislante.
- $R_{pared3} \rightarrow$ Resistencia del calorifugado.
- $R_{conv2} \rightarrow$ Resistencia convectiva generada por el aire.
- $L_1 \rightarrow$ Espesor tapa de la envolvente.
- $L_2 \rightarrow$ Espesor del aislante.
- $L_3 \rightarrow$ Espesor del calorifugado.
- $k_1 \rightarrow$ Conductividad térmica de la tapa de la envolvente.
- $k_2 \rightarrow$ Conductividad térmica del aislante.
- $k_3 \rightarrow$ Conductividad térmica del calorifugado.
- $h_1 \rightarrow$ Coeficiente convectivo del agua.
- $h_2 \rightarrow$ Coeficiente convectivo del aire.
- $A_{SP} \rightarrow$ Área de la tapa de la envolvente.

4 Hardware y algoritmo de cálculo

4.1 Hardware y lenguaje de programación empleado

La electrónica que se utiliza para ejecutar este simulador se compone de una placa Arduino modelo Mega 2560 que integra un microcontrolador del fabricante Atmel modelo ATmega2560 [3, 4].

Figura 4.1: Arduino Mega 2560.
Fuente: Imagen extraída de[4].

Es una placa diseñada para disponer de una variedad de entradas y salidas: 54 son salidas/entradas digitales, y 14 de ellas salidas de modulación por ancho de pulso, también conocido por las siglas "Modulación por Ancho de Pulsos (PWM)", mientras 16 son entradas de tipo analógico más 4 puertos tipo serie. También contiene un oscilador que trabaja a una frecuencia de 16MHz, además, de un puerto Bus Serie Universal (USB) 2.0 tipo B, pines de alimentación, pulsador de RESET y un conector In-Circuit Serial Programming (ICSP) [3, 4, 24].

A continuación, se muestran brevemente los datos técnicos de la electrónica Mega 2560 [3]:

- Microcontrolador: ATmega 2560.
- Memoria de programación: 256kb.
- Tamaño del bootloader:8kb.

- Regulación de tensión: Si, 5Vdc
- Entradas/salidas disponibles: 54 (14 tipo PWM)
- Entradas analógicas: 16.
- Tensión de alimentación: (se recomienda) 7-12.5V.
- Máxima corriente en las entradas: 40mA.
- Tensión en las salidas: 5V, 50mA.
- Dimensiones: 100 x 50 mm.
- Velocidad de reloj: 16MHz

El diagrama siguiente muestra el mapa de conexiones usados en el proyecto:

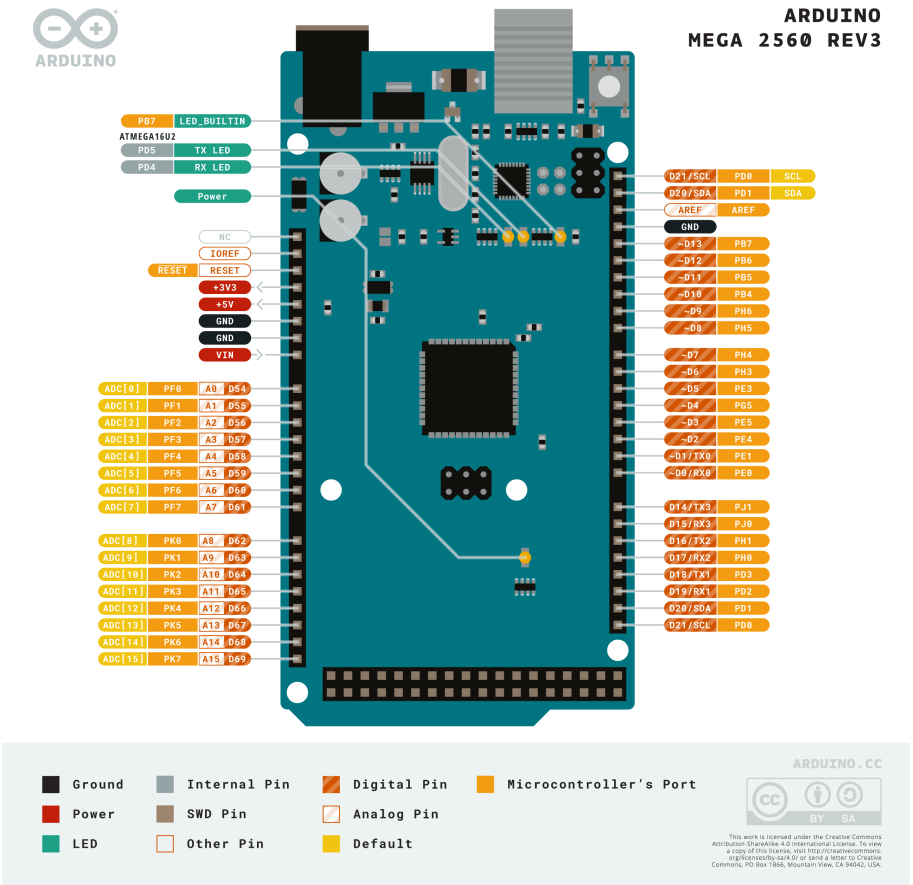

Figura 4.2: Diagrama de conexiones Arduino Mega 2560.
Fuente: Imagen extraída de web de Arduino[4].

El lenguaje de programación empleado para diseñar el firmware es C++, aunque no es C puro que proviene de AVR-libc, el cual provee una librería C de alta calidad lista para usar en los controladores de Atmel [16].

4.2 Conexiones y diagramas

En la realización del simulador no todas las entradas/salidas de la placa son utilizadas. Haciendo uso de la conexión USB se alimenta la placa, la pantalla Pantalla de Cristal Líquido (LCD) y el resto de componentes del simulador. Los Diodo Emisor de Luz (LED) que se emplean en este proyecto son de tipo bicolor con el fin de ocupar un LED para dos señales visuales y son conectados a las salidas digitales para su alimentación.

Identificación de los diferentes LEDs y su función:

- LED 1, conexión a pin número 22 → Alarma alto nivel de agua en la caldera (rojo).
- LED 1, conexión a pin número 23 → Nivel correcto de agua (verde).
- LED 2, conexión a pin número 24 → Alarma alta presión de vapor (rojo).
- LED 2, conexión a pin número 25 → Presión de vapor de trabajo correcta (verde).
- LED 3, conexión a pin número 26 → Bloqueo de la caldera y aviso RESET (rojo).
- LED 3, conexión a pin número 27 → Caldera en operación (verde).
- LED 4, conexión a pin número 28 → Estado del quemador encendido-apagado (verde).

Los LEDs por norma general trabajan con un voltaje de entre 2 y 3,5 voltios a una corriente nominal de unos 20mA, por lo que, al conectarlo a la placa se le aplicaría un voltaje mayor del que necesita. Esto provoca que la vida útil del diodo LED se vea afectada produciendo prematuramente una rotura del mismo. Para evitarlo se limita la corriente con ayuda de una resistencia.

Como se menciona anteriormente, se usan potenciómetros de 10kΩ su función es la de producir la modificación de variables de la simulación. Los potenciómetros se conectan por lo general a las entradas analógicas que cuentan con una resolución de 0 a 1024 y en algunos casos se deriva a masa la señal de los potenciómetros para evitar que existan valores de entrada estando el potenciómetro a 0.

- Potenciómetro 1 → Modifica la potencia del quemador.
- Potenciómetro 2 → Control de la válvula de agua de alimentación.
- Potenciómetro 3 → Regula la válvula de la línea de vapor.
- Potenciómetro 4 → Regulación de la retroiluminación del LCD.

La pantalla LCD recibe alimentación de la placa electrónica mediante los pines de alimentación y mediante cables que están conectados a los pines 20 (System Data (SDA)) y 21 (System Clock (SCL)), que conforman el bus de tipo Inter-Integrated Circuit (I2C), se comunican los datos a la pantalla desde la placa electrónica. Para ello, es necesario el uso de la librería Wire.

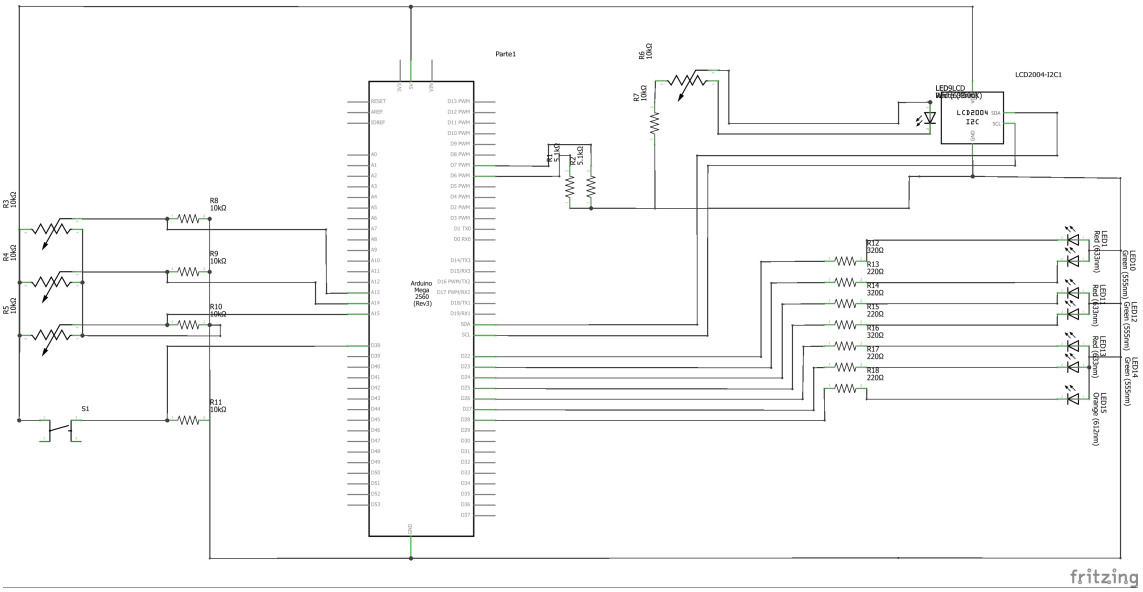

Figura 4.3: Conexiones del simulador elaborado con Fritzing.
Fuente: Elaboración propia.

Se ha realizado un esquema electrónico con ayuda del software fritzing para la representación eléctrica del simulador [18]. En el diagrama se pueden ver los diferentes componentes que forman el simulador y el cableado del mismo. Destacar que, la pantalla que se utiliza es de 4 líneas horizontales LCD.

4.3 Desarrollo del firmware

4.3.1 Código

4.3.1.1 Variables y constantes

Las variables y constantes son un conjunto de valores y definiciones. En las variables se puede almacenar por tipo de variable, modificable en todo momento que se utilice. Según el tipo de variables que se utilice tendrá definida una capacidad de almacenamiento en bit u otra, indicando su nombre detrás del tipo de variable. Las constantes son espacios de almacenamientos no modificables durante la ejecución del código, donde se almacena información para su posterior uso, solo se puede modificar en el código fuente, usadas normalmente por constantes como puede ser el número π o un valor de conductividad térmica. El listado de variables y constantes pertenecen a la configuración del firmware. En la siguiente sección de código se puede observar las constantes empleado en el firmware:

```
1         //Potencia perdida por aislamiento
2         //#define LosePower                              0
3
4
5         //Definición de las características de la instalación
6         //Tuberias
7         //Diámetro interior de la tubería (mm)
8         #define Di                 220.0
9         //Presión nominal de la tubería: 20bar
10        //Longitud equivalente (m)
11        #define LE                 250.0
12
```

```
13          //Dimensiones de la caldera
14          //Número de tubos del segundo paso de humos
15          #define nTubSecStep   150
16          //Numero de tubos del tercer paso de humos
17          #define nTubThiStep   84
18          //Diámetro del quemador (m)
19          #define FurnaceDiameter   1.31
20          //Diámetro de la caldera (m)
21          #define boilerDiameter    3.2
22          //Diámetro de los tubos de paso de humos (m)
23          #define tubeDiameter    0.061
24          //Longitud del hogar de la caldera (m)
25          #define furnaceLength   5.638
26          //Longitud de los tubos del segundo paso de humos (m)
27          #define secStepLength   4.724
28          //Longitud de los tubes del tercer paso de humos (m)
29          #define thiStepLength   5.94
30          //espesor del acero de la caldera (m)
31          #define MaterialThickness   0.008
32          //Longitud de la caldera (m)
33          #define BoilerLength    6.2
34          //Internal boiler wall radius (mm)
35          #define internalBoilerWRad   1.491
36          //External boiler wall radius (mm)
37          #define externalBoilerWRad   1.499
38          //External boiler thermal isolation radius (mm)
39          #define externalBoilerThermalIsoRad   1.599
40          //External boiler radius (mm)
41          #define externalBoilerRad   1.6
42          //Front and rear cover radius (m)
43          #define CoverRadius   1.6
44
45          //Fuel data
46          #define LowCalVal  39765//kJ/kg
47
48          //Adiabatic temperatures
49          //Light fuel oil (Â°C)
50          #define L_FuelAdiabaticTemperature  2104
51          //Medium fuel oil (Â°C)
52          #define M_FuelAdiabaticTemperature  2101
53          //Heavy fuel oil (Â°C)
54          #define H_FuelAdiabaticTemperature  2102
55
56
57          //Chemical Data
58          //Carbon (g/mol)
59          #define C_AtomicWeight  12
60          //Hydrogen (g/mol)
61          #define H_AtomicWeight  1.008
62          //Nitrogen (g/mol)
63          #define N_AtomicWeight  14
64          //Oxygen (g/mol)
65          #define O_AtomicWeight  16
66          //Fuel (kg/mol)
67          #define Fuel_molecWeight  0.28233
68          //Carbon dioxide (kg/mol)
69          #define CO2_molecWeight  0.044
70          //Water (kg/mol)
71          #define H2O_molecWeight  0.018016
72
73          //Specifics heats
74          //Nitrogen (kJ/kgÂ°C)
75          #define N2SH  1.04
```

```
76          //Oxygen (kJ/kgÂºC)
77          #define O2SH  0.918
78          //Fuel (kJ/kgÂºC)
79          #define FuelSH  1.5899
80          //Carbon dioxide  (kJ/kgÂºC)
81          #define CO2SH  0.8439
82          //Water (kJ/kgÂºC)
83          #define H2OSH  4.18
84
85          //Definición de coeficientes termicos
86          //Water Thermal Coefficient (W/m2Â·K)
87          #define K_H2O   62
88          //Air Thermal Coefficient (W/m2Â·K)
89          #define K_Air   18
90          //Exhaust gas Thermal Coefficient (W/m2Â·K)
91          #define KExhaustGas   280
92          //Steel Thermal Coefficient (W/mÂ·K)
93          #define SteelThermalCoefficient   50.20
94          //Rock Wool Thermal Coefficient (W/mÂ·K)
95          #define RockWoolThermalCoefficient   0.0410
96          //Aluminiun Thermal Coefficient (W/mÂ·K)
97          #define AluminiumlThermalCoefficient   209.3
98
99          //Definición de constantes
100         //numero pi
101         #define pi    3.1416
```

Y seguidamente se muestran las variables de control en la siguiente sección de código:

```
1   //Definición de variables de control
2
3   //Potencia perdidas por aislamiento
4   float LosePower = 0.0;
5   //Rendimiento del quemador
6   float etaBurner = 0.0;
7   //Carga de la caldera
8   float BoilerLoad = 0;
9   //Potencia transferida al agua (kW)
10  float PowerTransferred = 0.0;
11
12  //Variables cálculo de pérdidas térmicas
13  //pi number pow 2
14  float pi2 = pow(pi, 2);
15  //Cilinder area
16  float BoilerCylindricalSurface = 0.0;
17  //Circunference area
18  float BoilerCoverSurface = 0.0;
19  //Cilinder area
20  float BoilerExternalCylindricalrSurface = 0.0;
21  //Boiler steel thickness
22  float BoilerSteelThickness = 0.0;
23  //Boiler isolation thickness
24  float BoilerIsolationThickness = 0.0;
25  //Boiler isolation protector thickness
26  float BoilerIsolationProtectorThickness = 0.0;
27
28  //Boiler body thermal resistance (W)
29  float Rtotalcyl = 0.0;
30  //Lost heat boiler body insulation
31  float Qperd1 = 0.0;
32  //Boiler front cover thermal resistance (ÂºC/W)
33  float Rtotalfront = 0.0;
34  //Boiler rear cover thermal resistance (ÂºC/W)
35  float Rtotalrear = 0.0;
```

```
36 //Lost heat insulationr front cover (W)
37 float Qperd2 = 0.0;//(W)
38 //Lost heat insulation rear cover (W)
39 float Qperd3 = 0.0;//(W)
40 //Boiler thermal resistance (ÂºC/W)
41 float Rboiler = 0.0;
42
43 //Temperatura de la caldera
44 float BoilerTemperature = 0.0;
45 //Temperatura tras el aporte energético del quemador
46 float IntermediateTemperature = 0.0;
47 //Energía neta
48 double NetEnergy = 0.0;
49 //Energía de calentamiento a Ts
50 float HeatEnergy = 0.0;
51 //Energía sobrante
52 float SurplusEnergy = 0.0;
53 //Masa de vapor generado
54 float SteamMass = 0.0;
55 //Volumen de vapor generado
56 float SteamVolume = 0.0;
57 //Calculo de la presion generada por la masa de vapor
58 float SteamPressurecreate = 0.0;
59 //Numero de moles
60 float n = 0.0;
61
62 //Combustión, variables cálculo temperatura del hogar
63 //Fuel Flow
64 float FuelFlow = 0.0;
65 //Value n of Air Excess
66 float nExcess = 0.0;
67 //Reactive combustion products flow
68 //Moles of fuel for current fuel flow
69 float MOLFlowFuel = 0.0;
70 //Moles of oxygen for stoichiometric combustion
71 float MOLO2stoichiometric = 0.0;
72 //Moles of oxygen flow for stoichiometric combustion
73 double MOLO2stoichiometricFlow = 0.0;
74 //No reactive exhaust gases flow with air excess
75 //Excess oxigen flow
76 double O2Flow = 0.0;
77 //Excess nitrogen flow
78 double N2Flow = 0.0;
79 //Excess carbon dioxide flow
80 double CO2Flow = 0.0;
81 //Reactive exhaust gases flow with air excess
82 //Reactive carbon dioxide flow of exhaust gas
83 double CO2Flowreactive = 0.0;
84 //Reactive water steam flow of exhaust gas
85 double H2OFlowreactive = 0.0;
86 //Non-reactive nitrogen flow of exhaust gas
87 double N2Flownonreactive = 0.0;
88 //Non-reactive carbon dioxide flow of exhaust gas
89 double CO2Flownonreactive = 0.0;
90 //Non-reactive oxygen flow of exhaust gas
91 double O2Flownonreactive = 0.0;
92 //Boiler furnace temperature
93 float FurnaceTemp = 0.0;
94
95 //Efficiency according to boiler load
96 //Curve Efficiency
97 float Efficiency = 0.0;
98 //U coefficient
```

```
99  float CoeffU = 0.0;
100 //Real logarithmic mean temperature
101 float LMTreal = 0.0;
102 //Iterated logarithmic mean temperature
103 float LMTiterated = 0.0;
104 //Exhaust Temp
105 float ExhaustTemp = 0.0;
106
107 //variables cálculo de superficie
108 //Radio interno del hogar (m)
109 double Internalfurnrad = 0.00;
110 //Radio externo del hogar (m)
111 double Externalfurnrad = 0.00;
112 //Radio interno de los tubos del segundo paso de humos (m)
113 double IntsecSteprad = 0.00;
114 //Radio Externo de los tubos del segundo paso de humos (m)
115 double ExtsecSteprad = 0.00;
116 //Radio interno de los tubos del tercer paso de humos (m)
117 double IntthiSteprad = 0.00;
118 //Radio externo de los tubos del tercer paso de humos (m)
119 double ExtthiSteprad = 0.00;
120 //Superficie del hogar (m2)
121 double Furnacesurface = 0.00;
122 //Superficie del segundo paso de humos (m2)
123 double secStepsurface = 0.00;
124 //Superficie del tercer paso de humos (m2)
125 double thiStepsurface = 0.00;
126 //Superficie total de intercambio de la caldera (m2)
127 double ExchangeSurface = 0.00;
```

4.3.1.2 Cálculo de pérdidas térmicas

El siguiente diagrama muestra el proceso que sigue el firmware para recopilar datos y reproducir los cálculos.

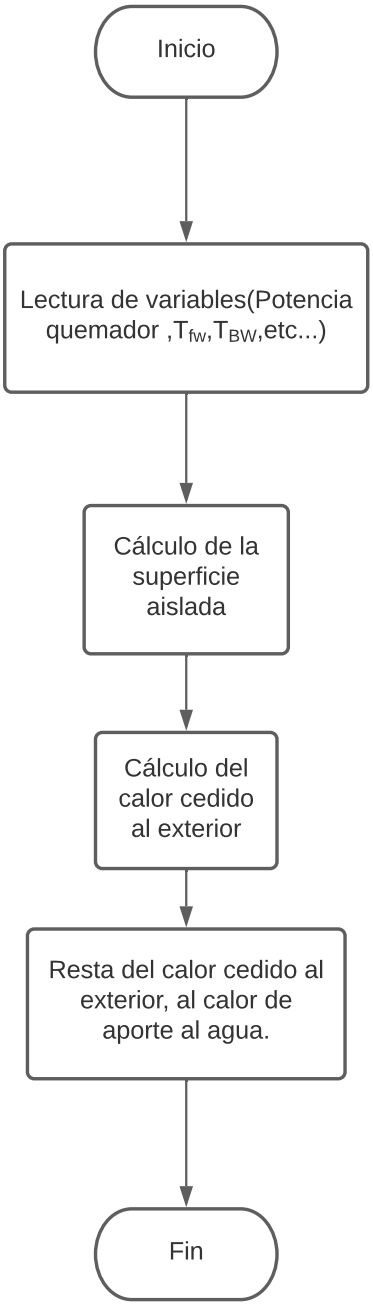

Figura 4.4: Diagrama de flujo para el cálculo de las perdidas térmicas.
Fuente: Elaboración propia.

```
/*===> Calculates thermal losses in insulation */
void calculateThermalLosses(){
        /*===> Calculates thermal loses */
        // pi number pow 2
        pi2 = pow(pi, 2);
        //cilinder area
        BoilerCylindricalSurface = 2*pi*internalBoilerWRad*BoilerLength;
        //cover circunference area
        BoilerCoverSurface= pi*pow(CoverRadius,2);
        //boiler cilinder area
```

```
11          BoilerExternalCylindricalrSurface= 2*pi*externalBoilerRad*
                BoilerLength;
12          //boiler steel thickness
13          BoilerSteelThickness= externalBoilerWRad - internalBoilerWRad;
14          //boiler rockwool thickness
15          BoilerIsolationThickness= externalBoilerThermalIsoRad -
                externalBoilerWRad;
16          //boiler isolation protector (aluminium) thickness
17          BoilerIsolationProtectorThickness= externalBoilerRad -
                externalBoilerThermalIsoRad;
18
19          //boiler body thermal resistance
20          Rtotalcyl = ((1/(K_H2O*BoilerCylindricalSurface)) + ((log(
                externalBoilerWRad/internalBoilerWRad)))/(2*pi*BoilerLength*
                SteelThermalCoefficient)) + ((log(externalBoilerThermalIsoRad/
                externalBoilerWRad))/(2*pi*BoilerLength*
                RockWoolThermalCoefficient)) + ((log(externalBoilerRad/
                externalBoilerThermalIsoRad))/(2*pi*BoilerLength*
                Aluminium1ThermalCoefficient)) + (1/(K_Air*
                BoilerExternalCylindricalrSurface)); //(Â ºC/W)A1=A4=2*pi*r*l
21          //lost heat boiler body insulation
22          Qperd1 = (WaterTemperature-EnvironmentTemperature)/Rtotalcyl;//(W)
23          //boiler front cover thermal resistance
24          Rtotalfront = (1/(K_H2O*BoilerCoverSurface)) + (BoilerSteelThickness
                /(SteelThermalCoefficient*BoilerCoverSurface)) + (
                BoilerIsolationThickness/(RockWoolThermalCoefficient*
                BoilerCoverSurface)) + (BoilerIsolationProtectorThickness/(
                Aluminium1ThermalCoefficient*BoilerCoverSurface)) + (1/(K_Air*
                BoilerCoverSurface)); //(Â ºC/W)
25          //boiler rear cover thermal resistance is the same as the front
                cover
26          Rtotalrear = Rtotalfront; //(Â ºC/W)
27          //lost heat insulationr front cover
28          Qperd2 = (WaterTemperature-EnvironmentTemperature)/Rtotalfront;//(W)
29          //lost heat insulation rear cover
30          Qperd3 = Qperd2;//(W)
31          //total heat lost through insulation
32          LosePower = Qperd1 + Qperd2 + Qperd3; // power losses.
33          //boiler thermal resistance
34          Rboiler = Rtotalcyl + Rtotalfront + Rtotalrear;
35
36
37
38  }
```

En el fragmento de código anterior se han utilizado las ecuaciones 3.33, 3.34, 3.35, 3.36

Se realiza primeramente el cálculo de la superficie exterior de la caldera que es la que va a producir intercambio de calor con el exterior y va a reducir el calor que está contenido en el agua. Se calculan las resistencias térmicas en las tapas de los pasos de humos, además de la resistencia térmica de la propia caldera, que es considerada un cilindro. A continuación, se calcula el calor cedido al ambiente mediante la fórmula matemática que también es utilizada para las cubiertas frontales.

4.3.1.3 Cálculo de la superficie de intercambio

El proceso de cálculo de la superficie de intercambio está descrito en el diagrama de flujo que se muestra en la figura 4.4 y su código de programación se muestra a continuación:

```
1
2   /*===> Calculate exchange surface */
```

```
3  void exchangeSurface(){
4          //Calculate some dimensions of the boiler
5          //Internal furnace radius
6          Internalfurnrad = FurnaceDiameter / 2;
7          //External furnace radius
8          Externalfurnrad = Internalfurnrad + MaterialThickness;
9          //Internal radius of the pipes of the second fume pass
10         IntsecSteprad = tubeDiameter / 2;
11         //Extternal radius of the pipes of the second fume pass
12         ExtsecSteprad = IntsecSteprad + MaterialThickness;
13         //Internal radius of the pipes of the third fume pass
14         IntthiSteprad = IntsecSteprad;
15         //External radius of the pipes of the third fume pass
16         ExtthiSteprad = ExtsecSteprad;
17         //Furnace surface
18         Furnacesurface = 2 * pi * Internalfurnrad * furnaceLength;
19         //Second fume pass surface
20         secStepsurface = 2 * pi * IntsecSteprad * secStepLength;
21         //Third fume pass surface
22         thiStepsurface = 2 * pi * IntthiSteprad * thiStepLength;
23         //Total boiler exchange surface
24         ExchangeSurface = Furnacesurface + (nTubSecStep*secStepsurface) + (
               nTubThiStep*thiStepsurface);
25
26 }
```

Se calcula mediante el número de tubos y las dimensiones de la caldera, la superficie interna de intercambio de calor de la caldera. Esta superficie es de utilidad para cálculos posteriores.

4.3.2 Cálculo de la temperatura del hogar

Como se muestra a continuación el diagrama de flujo muestra el proceso de cálculo y las decisiones del firmware para obtener la temperatura del hogar.

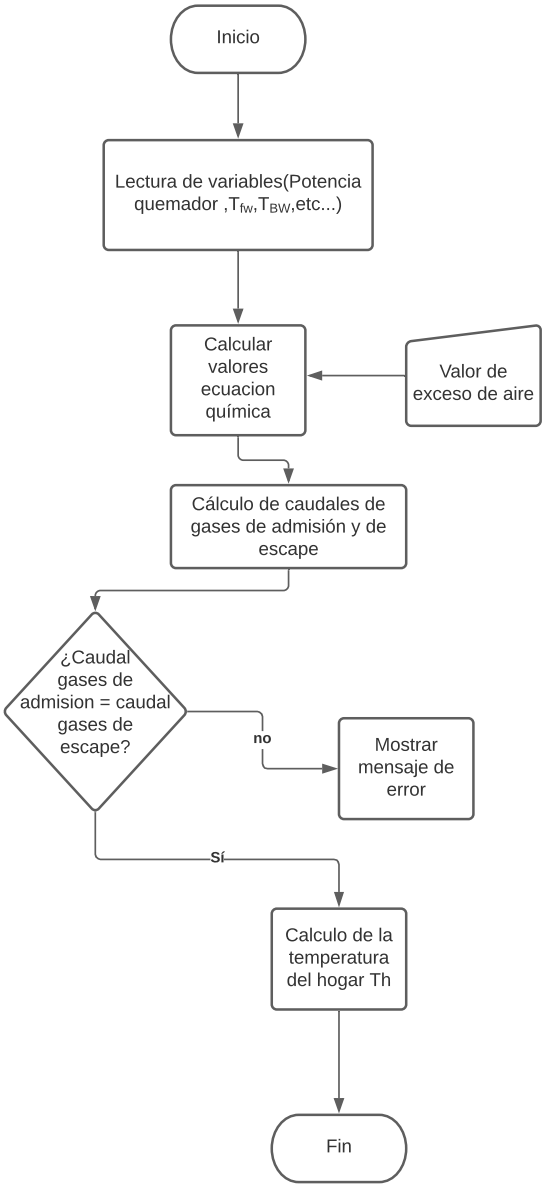

Figura 4.5: Diagrama de flujo para el cálculo de las pérdidas térmicas.
Fuente: Elaboración propia.

```
1
2  /*===> Calculate the boiler furnace temperature */
3  void calculateFurnaceTemperature(){
4
5          BoilerLoad = etaBurner;
6          if(BoilerLoad>0){
7                  float ExcessAir = 30;//Potenciometro
8                  //Calculate fuel flow
9                  FuelFlow = ((PMax*(BoilerLoad/100))/LowCalVal);
10                 //Calculate air excess
11                 nExcess = (ExcessAir/100)+1;
12                 //Reactive combustion products flow
13                 MOLFlowFuel = (FuelFlow/Fuel_molecWeight);
14                 MOLO2stoichiometric = MOLFlowFuel*(((61/2)*nExcess)/nExcess)
                       ;
```

```
15                  MOLO2stoichiometricFlow = (MOLO2stoichiometric *(
                        O_AtomicWeight*2)/1000);
16                  //Calculate no reactive exhaust gases flow with air excess
17                  O2Flow = ((MOLFlowFuel*((61/2)*nExcess)*(O_AtomicWeight*2))
                        /1000);
18                  N2Flow = (((78.06/20.98)*((61/2)*nExcess))*MOLFlowFuel*(
                        N_AtomicWeight*2)/1000);
19                  CO2Flow = ((((0.04/20.98)*((61/2)*nExcess))*MOLFlowFuel*(
                        CO2_molecWeight*1000))/1000);
20                  //Calculate reactive exhaust gases flow with air excess
21                  CO2Flowreactive = 20*MOLFlowFuel*CO2_molecWeight;
22                  H2OFlowreactive = 21*MOLFlowFuel*H2O_molecWeight;
23                  N2Flownonreactive = N2Flow;
24                  CO2Flownonreactive = CO2Flow;
25                  O2Flownonreactive = (((61/2)*(nExcess-1))*MOLFlowFuel*((
                        O_AtomicWeight*2))/1000);
26                  //Equation for calculate furnace temperature
27                  FurnaceTemp = (((((CO2Flowreactive*CO2SH)+(H2OFlowreactive*
                        H2OSH))*H_FuelAdiabaticTemperature)+(((N2Flownonreactive
                        *N2SH)+(O2Flownonreactive*O2SH)+(CO2Flownonreactive*
                        CO2SH))*EnvironmentTemperature))/((CO2Flowreactive*CO2SH
                        )+(H2OFlowreactive*H2OSH)+(N2Flownonreactive*N2SH)+(
                        O2Flownonreactive*O2SH)+(CO2Flownonreactive*CO2SH)));
28          }else {FurnaceTemp = WaterTemperature;}
29  }
```

En este proceso se han utilizado las ecuaciones 3.11, 3.9, 3.12, 3.13, 3.14, 3.28.

En el cálculo de la temperatura del hogar se debe disponer en el sketch correspondiente las constantes químicas y datos que necesita el código para su funcionamiento. Inicialmente, se calculan los moles de fuel y oxígeno en una reacción estequiométrica. A continuación, los gases de admisión o gases que entran con el aire del quemador y se obtienen los caudales para la cantidad de fuel inyectada. Por último, el cálculo de los caudales de gases de escape que se generan tras la combustión incluyendo los propios de la reacción exotérmica. Finalmente, tras estos cálculos se procede mediante la ecuación 3.28 al cálculo de la temperatura del hogar.

4.3.2.1 Cálculo de la eficiencia según la carga de vapor

Se ha implementado en la siguiente sección de código el proceso para obtener la eficiencia de la caldera, este procedimiento está recogido en el diagrama de flujo de la figura 4.6.

```
1   /*===> Calculate boiler efficiency */
2   void calculateBoilerEfficiency(){
3           BoilerLoad = etaBurner;
4           //calculate efficiency
5           if(BoilerLoad > 0.194){
6                   Efficiency = 13.741*log(BoilerLoad)+22.563;//Curve
                        Efficiency
7
8           }else {Efficiency = 0;}
9
10  }
```

En esta sección de código se ha dispuesto la ecuación 3.29 que se obtuvo de la gráfica de eficiencia según carga de vapor obteniendo el valor directamente.

4.3.2.2 Cálculo de la temperatura de escape

En el diagrama de flujo 4.6 se muestra la ejecución que realiza el firmware para la obtención de los resultados de los cálculos.

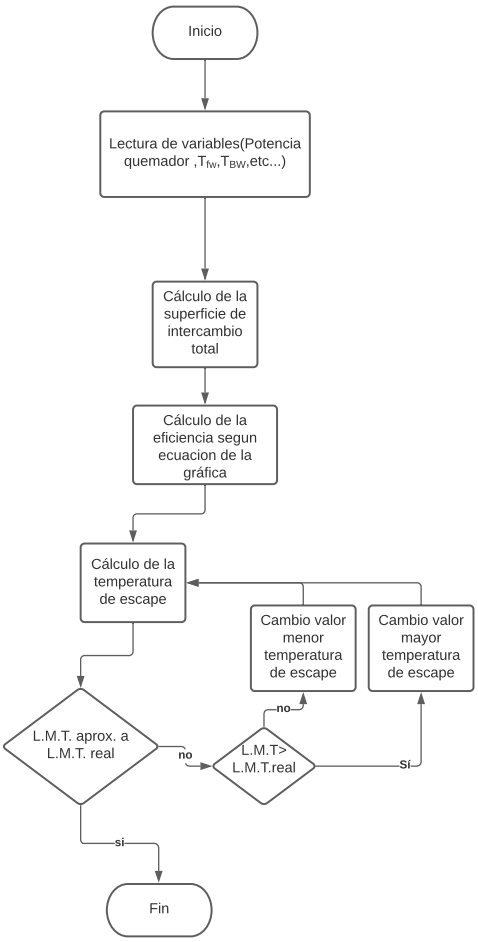

Figura 4.6: Diagrama de flujo para el cálculo de las temperaturas de escape.
Fuente: Elaboración propia.

```
1
2  /*===> Calculate exhaust temperature */
3  void calculateExhaustTemperature(){
4          // Calculates U coeff
5          CoeffU = (0.050760);//kW/m2ÂºC
6          //Temperatura media logarítmica
7          LMTreal = (PowerTransferred/(ExchangeSurface*CoeffU));//ÂºC
8          //Configuration of Do-While
9          float LowValue = WaterTemperature;
10         float HighValue = FurnaceTemp;
11         //Determines initial Exhaust temp
12         ExhaustTemp = (FurnaceTemp - WaterTemperature) / 2;
13         //Operation variables
14         float DeltaTemp = 0.0;
15         float ValidOperation = 0;
16
```

```
17          // The first range is iterated and reduced by using the sign-change
                method until
18          // the calculated difference is less than 0.05.
19          if((FurnaceTemp>(WaterTemperature+10))&& (etaBurner>23)){
20
21                  do {
22                          LMTreal = (PowerTransferred/(ExchangeSurface*CoeffU)
                                );//Â°C
23                          LMTiterated= (((FurnaceTemp-WaterTemperature)-(
                                ExhaustTemp-FeedWaterTemperature))/
24                          (log((FurnaceTemp-WaterTemperature)/(ExhaustTemp-
                                FeedWaterTemperature))));//Â°C
25
26                          if (LMTiterated < LMTreal) {LowValue = ExhaustTemp;}
27                          else{ HighValue = ExhaustTemp;}
28                          DeltaTemp = LMTreal-LMTiterated;
29
30                          ExhaustTemp = LowValue + (HighValue - LowValue) / 2;
31
32                  }while ((abs(DeltaTemp) > 0.05));
33          }else{ExhaustTemp = WaterTemperature;}
34
35
36  }
```

Para finalizar, en esta sección de código se ha calculado mediante las ecuaciones 3.30, 3.31, 3.32, de manera iterativa la temperatura de los gases de escape hasta aproximar la temperatura a un valor de +-0.05℃a la temperatura real de gases de escape. La búsqueda de la temperatura de gases de escape fue posible gracias al uso del bucle do-while el cual aproxima el valor repitiendo el cálculo tantas veces como sean necesarias.

5 Resultados

Se han simulado diversas condiciones de funcionamiento con variaciones en el estado de carga del quemador y el porcentaje de exceso de aire. Los distintos métodos de depuración del código han sido los siguientes: mediante lectura de variables en LCD, por extracción de datos por el puerto serie y por lectura del código en el Entorno de Desarrollo Integrado (IDE) de Arduino. Se han corregido diversos errores de funcionamiento modificando fracciones de código por las correcciones necesarias que corrige el defecto en funcionamiento.

5.1 Datos de la simulación

Con el fin de observar el comportamiento de la caldera, se han realizado diversas simulaciones. Los datos obtenidos han servido tanto para observar el comportamiento del simulador como para detectar errores en la ejecución del firmware. En las siguientes secciones se muestra un representación gráfica de dichos datos, una vez depurado el código fuente.

5.1.1 Simulación: Comportamiento del aislamiento térmico según la temperatura del agua

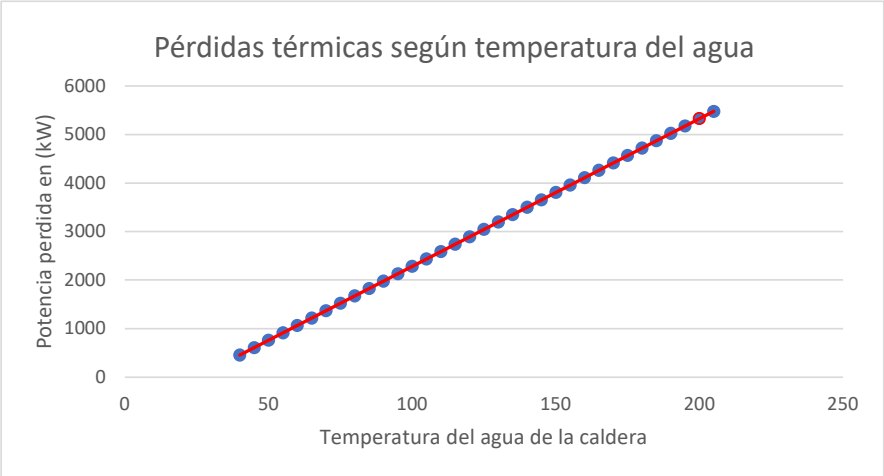

Figura 5.1: Temperatura de agua frente a la potencia perdida.
Fuente: Elaboración propia.

La figura 5.1 en la que se ha comparado los valores de la temperatura de agua que se encuentra en el interior de la caldera frente a la potencia perdida a través del aislante de la envolvente. En esta se aprecia que la línea de tendencia de los datos extraídos de la simulación define una trayectoria lineal con una pendiente positiva. Se observa que a mayor temperatura del agua de la caldera van a existir unas pérdidas térmicas mayores llegando, en este caso, a un valor de 5500 W.

5.1.2 Simulación: Comportamiento de la temperatura del hogar frente al exceso de aire

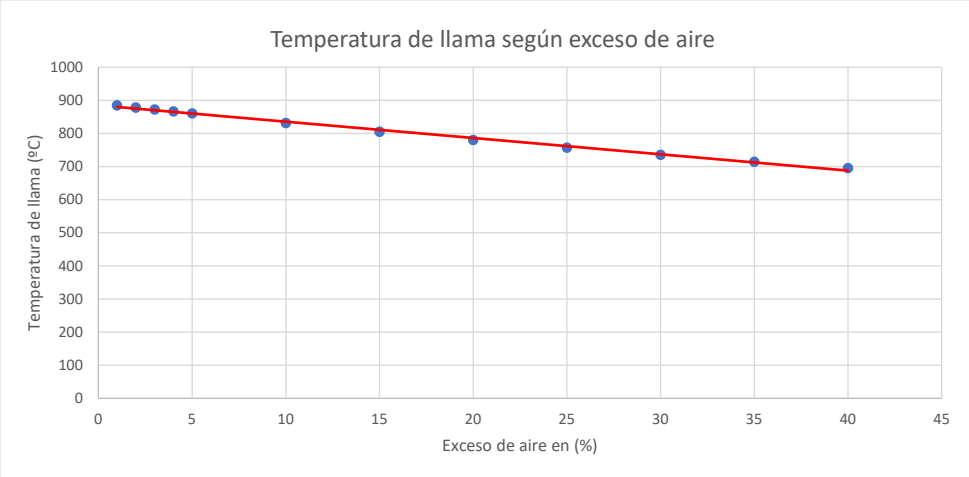

Figura 5.2: Temperatura del hogar frente al exceso de aire.
Fuente: Elaboración propia.

La figura 5.2 muestra el comportamiento de la temperatura del hogar frente al exceso de aire. Se aprecia que la línea de tendencia, respecto al aumento del exceso de aire, es decreciente, adoptando un comportamiento lineal con una pendiente negativa no acusada. De manera, que un aumento del exceso de aire va a disminuir la temperatura de llama de forma progresiva desde un máximo de 870ºC hasta 700ºC.

5.1.3 Simulación: Temperatura de escape y carga del quemador

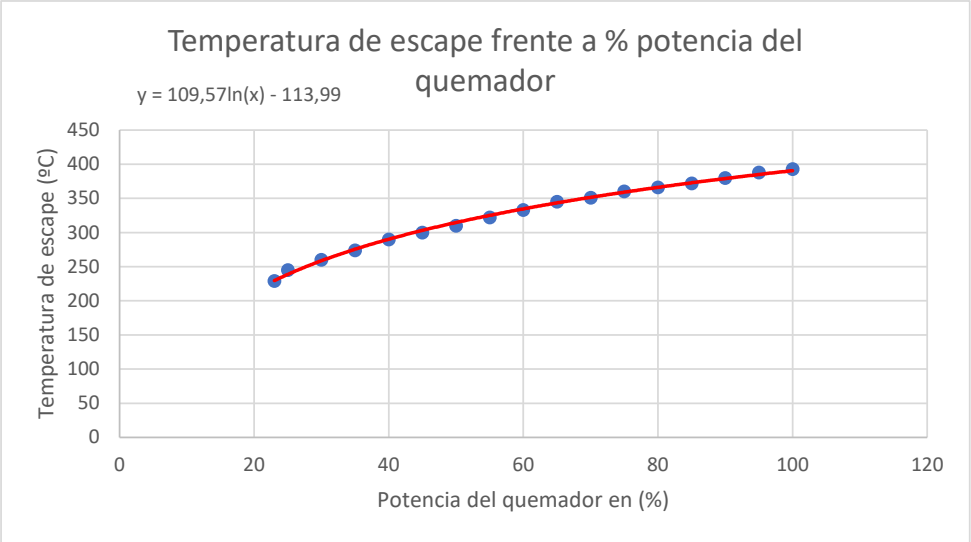

Figura 5.3: Temperatura de gases de escape frente a la carga del quemador.
Fuente: Elaboración propia.

La figura 5.3 muestra el comportamiento de la temperatura de los gases de escape frente a la carga del quemador, mostrando una variación de temperatura de escape comprendidos entre los 230℃y los 393℃. En la gráfica se puede visualizar una línea de tendencia de tipo logarítmica que no sigue un comportamiento lineal respondiendo desde una carga del quemador del 23 % hasta una carga del 100 % y una temperatura de agua durante el funcionamiento de 190℃ a 200℃.

5.1.4 Simulación: Calentamiento del agua de 45°C a 99°Cvisualización de tres variables

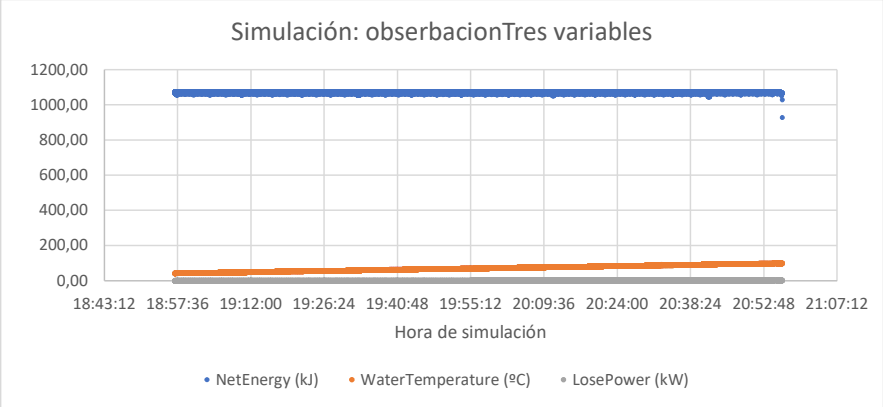

Figura 5.4: Simulación del calentamiento de agua.
Fuente: Elaboración propia.

En la figura 5.4 se puede observar como la energía neta mantiene una tendencia lineal durante las dos horas de la simulación porque, aunque existen pérdidas, solo representan un 1 % del total por ello no se aprecia ninguna tendencia en la línea de la variable NetEnergy de tipo decreciente. Como se puede observar la línea descrita por la variable LosePower tiene una pendiente positiva que no se alcanza a apreciar en este gráfico debido al aumento de las pérdidas de manera tan progresiva. Respecto a la variable WaterTemperature como se observa si describe una pendiente positiva visible en el gráfico que muestra que para calentar el agua desde los 45°Chasta los 99,60°Cemplea aproximadamente unas 2 horas y 15 minutos.

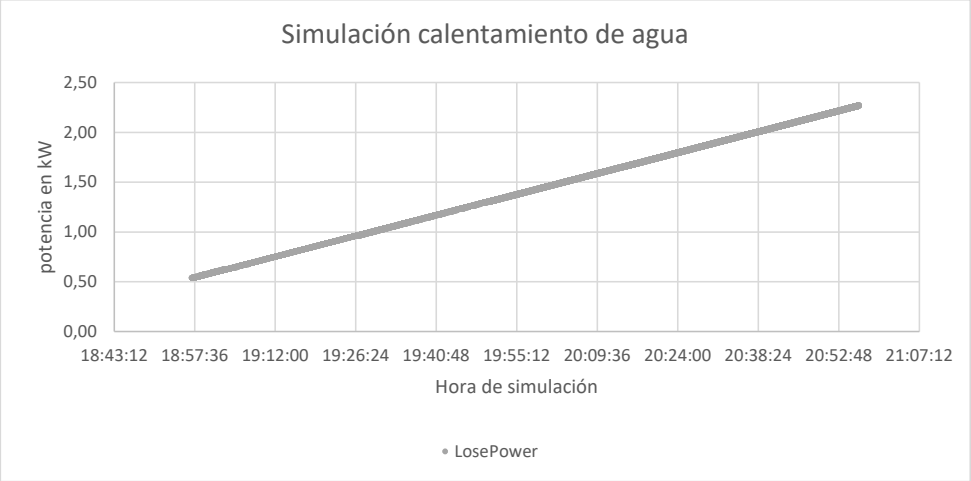

Figura 5.5: Visualización de la variable LosePower.
Fuente: Elaboración propia.

En la figura 5.5 se observa mejor el aumento de las pérdidas térmicas según aumenta la temperatura en el calentamiento las cuales estan comprendidas entre 0,5kW para 45℃y 2,25kW para 99,61℃.

5.2 Depuración del código

5.2.1 Fallos de la programación

Se muestran los diversos fallos explicando la corrección que se realiza.

- Error de los datos introducidos → Repetición de variables y uso de variables incorrectas.
- Error de sintaxis → Mala utilización de la sintaxis de la función.
- Integración de variables con el resto de código → Las variables no se integraban con el código ya realizado por lo que no modificaba el funcionamiento del resto del programa.
- Falta de terminadores en las funciones → Falta de elementos terminadores tal como ; }
- Falta de elementos condicionantes tipo if() → El código se ejecutaba siempre sin distinguir ciertos casos induciendo errores de cálculo.
- Error de programación del bucle Do-While → El bucle no realiza el cálculo bloqueando el programa porque no termina de ejecutar los comandos.
- Alta temperatura del micro-controlador → Debido a la gran cantidad de cálculos repetitivos el microprocesador sufre una elevada temperatura de trabajo dificultando la operación incluso llegando al bloqueo y reinicio.

5.2.2 Fallo en la ecuación NetEnergy

En el trabajo anterior [20] se llevaron a cabo cálculos para obtener la energía neta pero no se contempló una caldera con pérdidas térmicas sino a una caldera ideal sin estas pérdidas, es por ello, que en la ecuación el valor LosePower es 0. Como en este trabajo se han implementado los cálculos de las pérdidas térmicas esta variable tomará un valor obtenido en ese momento por los cálculos hechos. Si se da el caso de que el quemador está apagado y la variable PowerTransferred es 0 el resultado dará un valor negativo que a su vez será multiplicado por un diferencial de tiempo que también será negativo. La solución, por tanto, es aplicar el comando if-else para que cuando el valor NetEnergy sea menor a 0 lo corrija manteniéndolo en 0 hasta que sea positivo.

5.2.3 Fallo en la variable SteamMass

En la simulación la caldera responde normalmente realizando un calentamiento desde la temperatura en frío del agua hasta los 99,61℃ donde se produce una anomalía. La cual consiste en que la caldera cuando alcanza esta temperatura de 99,61℃ la variable HeatEnergy se va a un valor de 0 produciendo que los diferentes cálculos que se hallan a partir de esta variable muestren error o alcancen valores de infinito. Por lo tanto, una vez ocurre este fallo la simulación se detiene ya que todos los valores son incorrectos. El código ha sido revisado en búsqueda de problemas sin éxito, por lo que, la solución de este fallo queda como una propuesta para para la futura depuración del código.

5.3 Dispositivo simulador

El siguiente dispositivo se ha construido para este proyecto con una caja de registro eléctrica en la que se ha instalado la placa electrónica, el display, los diversos potenciómetros, LEDs indicadores y un interruptor para seleccionar el modo de pantalla, el cual, permite elegir entre dos pantallas que exponen las variables y otra que permite visualizar las alarmas. En la imagen 5.6 se observa el simulador terminado y en operación.

Figura 5.6: Dispositivo usado para la simulación.
Fuente: Imagen de elaboración propia.

6 Discusión y conclusión

6.1 Discusión

Tras la realización de este trabajo se puede concluir que un mayor exceso de aire produce un mayor enfriamiento de los gases de combustión. Esto sucede dado que la cantidad de exceso de aire que se está añadiendo a la combustión extrae una cantidad de calor determinada, por lo que, al aumentar el volumen de exceso de aire que se introduce para la combustión la cantidad de calor que este es capaz de extraer es mayor lo que conlleva a que la temperatura de llama sea menor.

Por otro lado, se concluye que la velocidad de calentamiento del agua va disminuyendo mientras la temperatura del agua aumenta lo que sucede de manera casi lineal y es debido a la aplicación de las pérdidas térmicas en la ecuación de la energía neta.

También, durante el calentamiento del agua, se observa que la energía neta que calcula la electrónica se mantiene alrededor de los mismos valores debido a que aunque existen pérdidas estas no supone mas de un 1 % por lo que no modifica mucho los valores de la variable NetEnergy. Con la variable LosePower vemos un comportamiento creciente del valor según aumenta la temperatura del agua como se apreció en las gráficas 5.4 y 5.5 que indica que se tendrá un valor máximo de pérdidas a la temperatura de servicio como muestra la gráfica5.1.

Por último, se obtiene la conclusión de que con la caldera en un estado de régimen la temperatura de gases va a ser más cercana a la temperatura del agua en el interior de la caldera cuando la carga del quemador sea mínima. Por el contrario, se obtendrá la máxima temperatura de gases con la caldera en estado de servicio y una carga del quemador cercana al 100 %.

6.2 Conclusión

Como objetivo general de este trabajo, se pretendía lograr un comportamiento más realista del simulador a partir de la introducción de las pérdidas térmicas de la envolvente de la caldera, así como mediante la determinación de la temperatura de los gases de escape. De forma específica, se deseaba determinar la potencia neta transferida al agua, teniendo en cuenta las pérdidas térmicas a través de la envolvente de la caldera. Estos objetivos se han cumplido, al haber desarrollado las modificaciones necesarias en el código fuente del simulador para la determinación en tiempo real de la temperatura de llama, pudiendo precisar la cantidad de calor transferida al agua, así como la temperatura de los gases de escape. Esto ha sido posible gracias a la introducción del cálculo de la potencia del proceso de combustión, teniendo en cuenta la variación del índice de exceso de aire, el cual puede ser modificado por el usuario del simulador y computado en tiempo real. Asimismo, la

introducción de una función específica para el cálculo de las pérdidas térmicas ha permitido determinar con mayor realismo la temperatura del agua en el interior de la caldera. La nueva versión del simulador permite mostrar el comportamiento, además de las variables que anteriormente simulaba, de la temperatura de llama, la temperatura de los gases de escape, así como el flujo de energía neta que es transferido al agua, teniendo en cuenta tanto la potencia procedente de la llama en el hogar, como la potencia perdida a través de la envolvente.

Se ha alcanzado una mejora notable en la visualización de los parámetros que permiten conocer mejor el comportamiento de la caldera bajo cualquier estado de funcionamiento. Debido a que se puede emular esa situación y ver que ocurre. En el ámbito académico, este simulador es útil para mostrar el funcionamiento de una caldera de vapor real a los estudiantes y que éstos puedan interactuar con la misma, pudiendo fabricar un regulador PID para controlar automáticamente los variables de nivel de agua y potencia de llama. Con este simulador se consiguen emular situaciones de peligro o de funcionamiento anormal, por lo que este firmware es útil para entrenar este tipo de situaciones de forma segura.

6.2.1 Limitaciones y líneas futuras

Durante el desarrollo de este trabajo se han encontrado ciertas limitaciones. La primera de ellas es la inexistencia de un sistema de purga de agua, ya que durante el calentamiento de la caldera el nivel de agua aumenta por encima del nivel máximo debido a la dilatación térmica. La segunda se refiere a la limitación del hardware empleado para la simulación, el cual esta al límite de su capacidad, calentándose en exceso, siendo recomendable la migración a un hardware más potente como una Raspberry Pi. Vinculado a esta mejora de hardware, es recomendable portar el código a Python para aprovechar al máximo el potencial que ofrece esta arquitectura.

En cuanto a las líneas futuras se plantea la fabricación de una consola impresa en 3d en la que se pueda disponer de varias pantallas de tipo LCD, incorporando los potenciómetros y los puertos necesarios para interaccionar con la instrumentación real.

6.3 Valoración personal

Este trabajo ha supuesto un gran reto a nivel teórico, ya que he tenido que adquirir conocimientos sobre reacciones químicas de combustión, calderas de producción de vapor y los procesos físicos sobre transferencia de calor. Además, he aprendido a utilizar de forma avanzada el LaTeX descubriendo diferentes configuraciones que se pueden emplear, así como mejorar mis habilidades en la redacción de textos académicos.

Con respecto a la caldera, he aprendido y ha supuesto para mí una manera de entender mejor cómo funciona, así como todos los factores que se tienen en cuenta en el proceso para que se ponga en marcha y la utilidad que tiene disponer de un simulador de bajo coste para realizar diferentes pruebas sobre un proyecto existente, lo cual considero muy útil a la hora de encarar diferentes fallos que puedan llegar a producirse en una situación real.

Bibliografía

[1] "Temperatura de llama adiabática." [Página web], 2008 [consultado 12 de diciembre de 2020]. URL: https://web.archive.org/web/20080112141325/http://www.doctorfire.com/flametmp.html.

[2] "Tratamiento de agua de calderas." [Página web], 2015 [consultado 3 de mayo de 2021]. URL: https://tratamientodeaguass.com/tratamiento-de-aguas-para-calderas/.

[3] "Arduino." [Página web], [consultado 10 de Enero de 2021]. URL: https://www.arduino.cc/.

[4] "Arduino, ARDUINO MEGA 2560." [Página web], [consultado 10 de Enero de 2021]. URL: https://store.arduino.cc/arduino-mega-2560-rev3.

[5] "Fueloil." [Página web], [consultado 15 de diciembre de 2020]. URL: https://es.wikipedia.org/wiki/Fueloil.

[6] "Combustión, Quemadores, Controles Y Sistemas de Seguridad de Llama. Proceso Básico De Combustión." [Página web], [consultado 16 de diciembre de 2020]. URL: http://recursosbiblio.url.edu.gt/publicjlg/biblio_sin_paredes/fac_ing/Manu_cald/cap/11.pdf.

[7] "Tablas de pesos atomico." [Página web], [consultado 28 de diciembre de 2020]. URL: https://www.mapa.gob.es/ministerio/pags/biblioteca/fondo/pdf/46691_16.pdf.

[8] Amell-Arrieta, A.; Agudelo, J. R.; Cortés, J., "Verificación experimental del efecto de la altitud sobre la potencia térmica de un quemador atmosférico". *Revista Facultad de Ingeniería Universidad de Antioquia*, (25), 2002: 26–35.

[9] Añorbe Díaz, B., *Fundamentos químicos aplicados al buque.* Polígono Industrial Costa Sur 38009 Santa Cruz de Tenerife: GRAFIEXPRESS, S.L., Octubre 2011.

[10] Bahamondes, P. A., "Descripción de calderas y generadores de vapor". 2006.

[11] Bermúdez Vásquez, E. A.; *et al.*, "Eficiencia energética de la caldera pirotubular con máquinas de 250 BHP de un laboratorio farmacéutico". 2005.

[12] Bragado, N.; Tapia, D.; Sutil, J., "Estimando la Conducción Térmica en Sólidos".

[13] Çengel, Y. A.; Ghajar, A. J., *Transferencia de calor y masa: fundamentos y aplicaciones, cuarta edición.* McGraw-Hill Interamericana, 2011.

[14] Glassman, I.; Yetter, R. A.; Glumac, N. G., *Combustion.* Academic press, 2014.

[15] Goldstein, H. L.; Siegmund, C. W., "Influence of heavy fuel oil composition and boiler combustion conditions on particulate emissions". *Environmental science & technology*, 10(12), 1976: 1109–1114.

[16] Herrador, R. E., "Guía de usuario de Arduino". *Universidad de Córdoba*, 13, 2009.

[17] Holman, J. P.; Valenzuela, R., *Transferencia de calor*. 660.28427 H65 1986., McGraw-Hill São Paulo, 1998.

[18] Knörig, A.; Wettach, R.; Cohen, J., "Fritzing: a tool for advancing electronic prototyping for designers". En: *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, 2009, pp. 351–358.

[19] Law, C. K.; Makino, A.; Lu, T., "On the off-stoichiometric peaking of adiabatic flame temperature". *Combustion and Flame*, 145(4), 2006: 808–819.

[20] Álvaro Luis Luis, "Algoritmo para la simulación de la transferencia de energía y determinación de la presión y temperatura del vapor en una caldera de vapor saturado orientado a su ejecución en un hardware libre." 2017. URL: http://riull.ull.es/xmlui/handle/915/6856.

[21] Martínez, M. M., *Combustión y quemadores*. 1, Marcombo, 2005.

[22] Mills, A. F.; Régules, S., *Transferencia de calor*, vol. 542. Irwin México, 1995.

[23] Palomo Cano, M., "Aislantes TÉrmicos, Criterios De SelecciÓn Por Requisitos EnergÉticos". Escuela Técnica Superior de Madrid. Universidad Politécnica de Madrid.

[24] Pedrera, A. C., *Arduino para Principiantes: 2ª Edición*. IT Campus Academy, 2017.

[25] Rivera, A. d. P. F.; *et al.*, "Modelo matemático de una cámara de combustión de una caldera pirotubular utilizando la herramienta matlab-simulink". *Matéria (Rio de Janeiro)*, 25, 2020.

[26] Rivero Rodríguez, P., "Optimización de la combustión." 2016. Dpto. de Ingeniería Marítima. Universidad de La Laguna.

[27] Rodríguez, M. L.; Moya, D. V.; Monzón, J. M., "Funcionamiento y pérdidas en calderas pirotubulares, estudios de casos". *Cuba: Universo Sur.*, 2015.

[28] Schmidt, P. F., *Fuel oil manual*. Industrial Press Inc., 1985.

[29] Torres Gómez, J. L., "Estudio energético para el re dimensionamiento de una caldera de vapor". 2015.

[30] Vilte, M. d. S.; De Paul, I., "Transferencia térmica por convección natural en un recinto cerrado en condiciones de equilibrio térmico y dinámico". *Avances en Energías Renovables y Medio Ambiente*, 4, 2000.

Anexos

A Tablas de datos

A.1 Tabla de datos de la gráfica temperatura de agua y calor perdido

Tª agua	Perdida térmica en (kW)
40	457
45	609
50	761
55	913
60	1065
65	1218
70	1370
75	1522
80	1674
85	1826
90	1979
95	2131
100	2283
105	2435
110	2588
115	2740
120	2892
125	3044
130	3196
135	3349
140	3501
145	3653
150	3805
155	3957
160	4110
165	4262
170	4414
175	4566
180	4718
185	4871
190	5023
195	5175
200	5327
205	5479

Tabla A.1: Tabla de datos gráfica temperatura de agua y calor perdido

A.2 Tabla de datos de la gráfica temperatura del hogar y exceso de aire

% Exceso de aire	Temperatura del hogar (℃)
1	884,77
2	878,56
3	872,44
4	866,4
5	860,46
10	831,97
15	805,39
20	780,53
25	757,24
30	735,37
35	714,8
40	695,41

Tabla A.2: Tabla de datos gráfica temperatura del hogar y exceso de aire

A.3 Tabla de datos de la gráfica temperatura de escape y carga del quemador

Carga	Temperatura de escape (℃)
23	229
25	245
30	260
35	274
40	290
45	300
50	310
55	322
60	333
65	345
70	351
75	360
80	366
85	372
90	380
95	388
100	393

Tabla A.3: Tabla de datos gráfica temperatura de escape y carga del quemador

B Código fuente del simulador

B.1 Código fuente

```
#include <math.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <PinChangeInterrupt.h>
#include "PhysicalProperties.h"
#include "BoilerConstants.h"
#include "BoilerControlVariables.h"

//PWM interrupt control variables
const byte channel_pin[] = {6, 7}; // feedwater, burner
volatile unsigned long rising_start[] = {0,0}; // feedwater, burner
volatile long channel_length[] = {0,0}; //feedwater, burner

//I2C address and size of LCD display
LiquidCrystal_I2C lcd(0x3F, 20, 4);


/*
-------------------------------------
MAIN ARDUINO FUNCTIONS
-------------------------------------
*/

void setup() {
        //Initialises LCD
        lcd.init();

        //Welcome message
        lcd.backlight();
        lcd.setCursor(5, 1);
        lcd.print("Open Boiler");
        lcd.setCursor(5, 2);
        lcd.print("Ver. 0.3");
        delay(1500);

        //Initialises digital PINs
        pinMode(LedBurnerOn, OUTPUT);
        pinMode(LedLowLevel, OUTPUT);
        pinMode(LedRightLevel, OUTPUT);
        pinMode(LedHighPressure, OUTPUT);
        pinMode(LedPressureOk, OUTPUT);
        pinMode(LedBoilerAlarm, OUTPUT);
        pinMode(LedBoilerNormal, OUTPUT);
        pinMode(InputReset, INPUT);
        pinMode(Display_1, OUTPUT);
        pinMode(Display_2, OUTPUT);
        pinMode(InputAutoManual, INPUT);
```

```
48                  pinMode(OutLevel, OUTPUT);
49
50                  // Initialises PWM PINs
51                  pinMode(channel_pin[0], INPUT);
52                  pinMode(channel_pin[1], INPUT);
53
54                  //Initialises serial communications
55                  Serial.begin (9600);
56
57                  //Interruption management of PWM signal
58                  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(
                        channel_pin[0]), onRising0, CHANGE);
59                  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(
                        channel_pin[1]), onRising1, CHANGE);
60
61                  //Screen clear
62                  lcd.clear();
63          }
64
65          void loop() {
66
67                  //Stores loop start time
68                  t_init = millis();
69
70                  //Reads water volume at start time
71                  v_init = WaterVolume;
72
73                  //Reads digital inputs
74                  readDigitalPorts();
75
76                  //Calculates water and steam thermodynamic properties at
                        actual pressure
77                  calculateThermodynamicVariables();
78
79                  //Calculates water level in boiler
80                  calculateLevel();
81
82                  //Calculates boiler pressure increase caused by pumping
                        water
83                  deltaVolumePressureVariation();
84
85                  //Rests blockages after pressing reset button
86                  controlReset();
87
88                  //Controls burner
89                  burnerControl();
90
91                  //Calculates heat transfer to water
92                  calculatePower();
93
94                  //Calculates Boiler efficiency according to efficiency curve
95                  calculateBoilerEfficiency();
96
97                  //Calculates boiler furnace temperature according to air
                        excess
98                  calculateFurnaceTemperature();
99
100                 //Calculates heat exchange surface
101                 exchangeSurface();
102
103                 //Calculates heat transfer to water
104                 calculateThermalLosses();
105
```

```
106                 //Calculates the amount of steam generated
107                 steamGeneration();
108
109                 //Calculates steam flow when main valve is open
110                 calculateSteamFlow();
111
112                 //Calculates boiler pressure caused by steam
113                 calculateSteamPressure();
114
115                 //Blocks boiler in case of too low/high water level, or too
                        highe pressure
116                 blockageControl();
117
118                 //Calculates Exhaust gas temperature
119                 calculateExhaustTemperature();
120
121                 //Writes information on LCD
122                 LCDSelect();
123
124
125                 //Writes parameters to digital ports for both human-machine
                        interface and an external control
126                 writeDigitalPorts();
127
128                 //Calculates water volumen diference at loop end
129                 v_delta = WaterVolume - v_init;
130
131                 //Calculates loop elapsed time
132                 t_delta =millis() - t_init;
133
134                 //Serial Print for output values
                        ------------------------------------------------------------------------

135                 Serial.print("\t");
136                 Serial.print("Net energy: ");
137                 Serial.print(NetEnergy);
138                 Serial.print("\t");
139                 Serial.print("Steam Mass: ");
140                 Serial.print(SteamMass);
141                 Serial.print("\t");
142                 Serial.print("Power Transferred: ");
143                 Serial.print(PowerTransferred);
144                 Serial.print("\t");
145                 Serial.print("Lose Power: ");
146                 Serial.print(LosePower);
147                 Serial.print("\t");
148                 Serial.print("Water Temperature: ");
149                 Serial.print(WaterTemperature);
150                 Serial.print("\t");
151                 Serial.print("Heat Energy: ");
152                 Serial.print(HeatEnergy);
153                 Serial.print("\n");
154         }
155
156
157         /*
158         ------------------------------------
159         AUXILIAR CONTROL FUNCTIONS
160         ------------------------------------
161         */
162
163         /*===> Reads control command inputs from PWM or manual
                potentiometers */
```

```
164         void readDigitalPorts() {
165                 int PWMControlBurner, PWMControlLevel;
166 
167                 FlagAutoManual = digitalRead(InputAutoManual);
168 
169                 if(FlagAutoManual = digitalRead(InputAutoManual)) {
170                         /* ==>Â¡Â¡Pending conversion of PWM HIGH signals
                               into controlvalues!!*/
171                         //Burner PWM command signal (auto command from
                               external controller)
172                         PWMControlBurner = channel_length[1];
173                         FlagBurner = HIGH; //Flag is HIGH on auto control
174 
175                         /* ==>Â¡Â¡Pending conversion of PWM HIGH signals
                               into controlvalues!!*/
176                         //Water valve PWM command signal (auto commad from
                               external controller)
177                         PWMControlLevel = channel_length[0];
178                         FlagWater = HIGH; //Flag is HIGH on auto control
179                 } else {
180                         FlagBurner = LOW; //Flag is LOW on manual control
181                         FlagWater = LOW;  //Flag is LOW on manual control
182                         //Manual burner command signal 0-100%
183                         ControlBurner = 100.0/1024 * float(analogRead(
                               InputBurnerManual));
184                         //Manual water valve command signal 0-90 degrees
185                         ControlLevel = 90.0/1024.0 * float(analogRead(
                               InputWaterManual));
186                 }
187 
188                 //Main steam valve command signal
189                 MainValve = analogRead(InputMainValve);
190                 //Minimum command value to open main steam valve
191                 if(MainValve<10) MainValve = 0;
192                 //Translates main steam valve command signal to aperture
                       percentage
193                 Valve = float(MainValve) * 100 / 1023;
194 
195         }
196 
197         /*===> Writes data on LCD panel */
198         //LCD Select
199         void LCDSelect(){
200                 Value1 = digitalRead(Display_1);  //lectura digital de pin
201                 Value2 = digitalRead(Display_2);  //lectura digital de pin
202                 if (Value1 == HIGH) {
203                         writeLCD();
204                 }
205                 else if (Value2 == HIGH){
206                         writeLCD2();
207                 }else  {writeLCD3();}
208 
209         }
210         void writeLCD() {
211                 //Prints burner power %
212                 if (FlagBurner) { //FlagBurner is HIGH on auto control
213                         lcd.setCursor(0, 0);
214                         lcd.print("B ");
215                         if (Burner < 9.95) lcd.print("  ");
216                         else if (Burner < 100.0) lcd.print(" ");
217                         lcd.print(Burner, 1);
218                 } else {        //FlagBurner is LOW on manual control
219                         lcd.setCursor(0, 0);
```

```
220                         lcd.print("B*");
221                         if (Burner < 9.95) lcd.print("  ");
222                         else if (Burner < 99.95) lcd.print(" ");
223                         lcd.print(Burner, 1);
224                 }
225
226                 //Prints feedwater aperture in %
227                 if (FlagWater) { //FlagWater is HIGH on auto control
228                         lcd.setCursor(0, 1);
229                         lcd.print("W ");
230                         if (ButterflyValve < 9.95) lcd.print(" ");
231                         lcd.print(ButterflyValve, 1);
232                         lcd.print((char)223);
233                 } else {      //FlagWater is LOW on auto control
234                         lcd.setCursor(0, 1);
235                         lcd.print("W*");
236                         if (ButterflyValve < 9.95) lcd.print(" ");
237                         lcd.print(ButterflyValve, 1);
238                         lcd.print((char)223);
239                 }
240
241                 //Prints main steam valve aperture in %
242                 lcd.setCursor(0, 2);
243                 lcd.print("MV");
244                 if (Valve < 9.95) lcd.print("  ");
245                 else if (Valve < 100.0) lcd.print(" ");
246                 lcd.print(Valve, 1);
247
248                 //Prints feedwater volumetric flow
249                 lcd.setCursor(8, 0);
250                 lcd.print("F");
251                 if (WaterInletFlow < 9.95) lcd.print(" ");
252                 lcd.print(WaterInletFlow, 1);
253
254                 //Prints total water volumen
255                 lcd.setCursor(8, 1);
256                 lcd.print("V");
257                 if (WaterVolume < 9.95) lcd.print(" ");
258                 lcd.print(WaterVolume, 1);
259
260                 //Prints total water level from bottom
261                 lcd.setCursor(8, 2);
262                 lcd.print("L");
263                 if (WaterLevel < 9.95) lcd.print(" ");
264                 lcd.print(WaterLevel, 1);
265
266                 //Prints level indicator sensor signal
267                 lcd.setCursor(0, 3);
268                 lcd.print("WL");
269                 if (WaterLevelIndicator > -9.5 && WaterLevelIndicator < 0.0)
                        lcd.print(" ");
270                 if (WaterLevelIndicator >= 0.0 && WaterLevelIndicator < 9.5)
                        lcd.print("  ");
271                 if (WaterLevelIndicator >= 9.5) lcd.print(" ");
272                 lcd.print(WaterLevelIndicator, 0);
273                 lcd.print("mm");
274
275                 //Prints boiler pressure
276                 lcd.setCursor(8, 3);
277                 lcd.print("P");
278                 if (SteamPressure < 9.95) lcd.print(" ");
279                 lcd.print(SteamPressure, 1);
280
```

```
281                 //Prints boiler water temperature
282                 lcd.setCursor(14,0);
283                 lcd.print("T");
284                 if (WaterTemperature < 9.95) lcd.print("   ");
285                 if (WaterTemperature >= 9.95 && WaterTemperature <= 99.5)
                        lcd.print(" ");
286                 lcd.print(WaterTemperature, 1);
287
288                 //Prints mass steam flow in kg/h
289                 lcd.setCursor(14,1);
290                 lcd.print("F");
291                 if (SteamFlow*3600 < 9.95) lcd.print("    ");
292                 if (SteamFlow*3600 >= 9.95 && SteamFlow*3600 < 99.5) lcd.
                        print("   ");
293                 if (SteamFlow*3600 >= 99.5 && SteamFlow*3600 < 999.5) lcd.
                        print("  ");
294                 if (SteamFlow*3600 > 999.5 && SteamFlow*3600 < 9999.5) lcd.
                        print(" ");
295                 lcd.print(SteamFlow*3600,0);
296
297                 //Auto/Manual display
298                 if(FlagAutoManual) {
299                         lcd.setCursor(14, 2);
300                         lcd.print("  AUTO");
301                 } else {
302                         lcd.setCursor(14, 2);
303                         lcd.print("ALARM");
304                 }
305
306                 //Prints ALARM or NORMAL signal
307                 if (Alarm) {
308                         lcd.setCursor(14, 3);
309                         lcd.print("MANUAL");
310                 } else {
311                         lcd.setCursor(14, 3);
312                         lcd.print("NORMAL");
313                 }
314                 //Blank spaces
315                 lcd.setCursor(7, 0);
316                 lcd.print(" ");
317                 lcd.setCursor(7, 1);
318                 lcd.print(" ");
319                 lcd.setCursor(7, 2);
320                 lcd.print(" ");
321                 lcd.setCursor(7, 3);
322                 lcd.print(" ");
323                 lcd.setCursor(13, 0);
324                 lcd.print(" ");
325                 lcd.setCursor(13, 1);
326                 lcd.print(" ");
327                 lcd.setCursor(13, 2);
328                 lcd.print(" ");
329                 lcd.setCursor(13, 3);
330                 lcd.print(" ");
331         }
332         void writeLCD2() {
333
334                 //Prints mass steam flow in kg/h
335                 lcd.setCursor(0,0);
336                 lcd.print("FT");
337                 if (FurnaceTemp < 9.95) lcd.print("    ");
338                 if (FurnaceTemp >= 9.95 && FurnaceTemp < 99.5) lcd.print("
                        ");
```

```
339                 if (FurnaceTemp >= 99.5 && FurnaceTemp < 999.5) lcd.print("
                        ");
340                 if (FurnaceTemp > 999.5 && FurnaceTemp < 9999.5) lcd.print("
                        ");
341                 lcd.print(FurnaceTemp,0);
342
343                 //Prints burner power %
344                 if (FlagBurner) { //FlagBurner is HIGH on auto control
345                         lcd.setCursor(0, 1);
346                         lcd.print("B ");
347                         if (Burner < 9.95) lcd.print("  ");
348                         else if (Burner < 100.0) lcd.print(" ");
349                         lcd.print(Burner, 1);
350                 } else {        //FlagBurner is LOW on manual control
351                         lcd.setCursor(0, 1);
352                         lcd.print("B*");
353                         if (Burner < 9.95) lcd.print("  ");
354                         else if (Burner < 99.95) lcd.print(" ");
355                         lcd.print(Burner, 1);
356                 }
357
358                 //Prints boiler water temperature
359                 lcd.setCursor(0,2);
360                 lcd.print("T ");
361                 if (WaterTemperature < 9.95) lcd.print("   ");
362                 if (WaterTemperature >= 9.95 && WaterTemperature <= 99.5)
                        lcd.print(" ");
363                 lcd.print(WaterTemperature, 1);
364
365                 //Print Exhaust temperature in (Â ºC)
366                 lcd.setCursor(0,3);
367                 lcd.print("ET");
368                 if (FurnaceTemp < 9.95) lcd.print("    ");
369                 if (FurnaceTemp >= 9.95 && FurnaceTemp < 99.5) lcd.print("
                        ");
370                 if (FurnaceTemp >= 99.5 && FurnaceTemp < 999.5) lcd.print("
                        ");
371                 if (FurnaceTemp > 999.5 && FurnaceTemp < 9999.5) lcd.print("
                        ");
372                 lcd.print(ExhaustTemp,0);
373
374                 //Prints boiler pressure
375                 lcd.setCursor(8, 0);
376                 lcd.print("P");
377                 if (SteamPressure < 9.95) lcd.print(" ");
378                 lcd.print(SteamPressure, 1);
379
380                 //Prints main steam valve aperture in %
381                 lcd.setCursor(8, 1);
382                 lcd.print("MV");
383                 if (Valve < 9.95) lcd.print("  ");
384                 else if (Valve < 100.0) lcd.print(" ");
385                 lcd.print(Valve, 1);
386
387                 //Prints ALARM or NORMAL signal
388                 if (Alarm) {
389                         lcd.setCursor(8, 2);
390                         lcd.print("ALARM ");
391                 } else {
392                         lcd.setCursor(14, 3);
393                         lcd.print("NORMAL");
394                 }
395
```

```
396
397                 //Blank spaces
398                 lcd.setCursor(7, 0);
399                 lcd.print(" ");
400                 lcd.setCursor(7, 1);
401                 lcd.print(" ");
402                 lcd.setCursor(7, 2);
403                 lcd.print(" ");
404                 lcd.setCursor(7, 3);
405                 lcd.print("              ");
406
407                 lcd.setCursor(13, 0);
408                 lcd.print("       ");
409                 lcd.setCursor(15, 1);
410                 lcd.print("     ");
411                 lcd.setCursor(14, 2);
412                 lcd.print("      ");
413                 //lcd.setCursor(13, 3);
414                 //lcd.print(" ");
415
416         }
417         void writeLCD3() {
418
419                 // Alarm Display
420                 //if(LedBoilerAlarm == LOW){
421                         if(PowerTransferred>0){
422                                 lcd.setCursor(12, 0);
423                                 lcd.print("Brn. ON ");
424                         }else {
425                                 lcd.setCursor(12, 0);
426                                 lcd.print("Brn. OFF");
427                         }
428                         lcd.setCursor(0, 0);
429                         lcd.print("ALARMS-LIST ");
430                         if (WaterLevelIndicator<-20){
431                                 lcd.setCursor(0, 1);
432                                 lcd.print("Low Water Level     ");
433                         }else if (WaterLevelIndicator>20){
434                                 lcd.setCursor(0, 1);
435                                 lcd.print("High Water Level    ");
436                         }else {
437                                 lcd.setCursor(0, 1);
438                                 lcd.print("Normal Water Level  ");
439                         }
440                         if (SteamPressure>20){
441                                 lcd.setCursor(0, 2);
442                                 lcd.print("Boiler High Pressure");
443                         }else if (SteamPressure<20&&SteamPressure>12){
444                                 lcd.setCursor(0, 2);
445                                 lcd.print("Normal Pressure     ");
446                         } else {
447                                 lcd.setCursor(0, 2);
448                                 lcd.print("Low Pressure        ");
449                         }
450                         if ((SteamPressure<20&&SteamPressure>12)&&(
                                WaterLevelIndicator<20&&WaterLevelIndicator>-20)
                                ){
451                                 lcd.setCursor(0, 3);
452                                 lcd.print("Boiler Normal Work  ");
453                         }else {
454                                 lcd.setCursor(0, 3);
455                                 lcd.print("Boiler Out of Param.");}
456                 }
```

```
457
458         /*===> Calculates steam flow for steam pressure,
459         internal diameter and equivalen length of steam pipe */
460         void calculateSteamFlow(){
461                 /*The function uses two nested loops to solve the White-
                        Colebrook equation
462                 and then equalize head losses with the difference of
                        pressure*/
463                 double f = 0.008, R = 1000.0, a = 0.0, b = 0.0, SteamVolFlow
                         = 0.0, hf = 0.0;
464
465                 do { // External loop iterates Reynolds data with Darcy-
                        Weisbach equation
466
467                         do { // Inner loop calculates factor f with White-
                                Colebrook equation
468                                 a = 1/sqrt(f);
469                                 b = 2 * log10(R * sqrt (f)) - 0.8;
470                                 f+= 0.001;
471                         } while (a-b > 0.00001);
472
473                         // Calculates initial volumetric flow with
                                continuity equation
474                         SteamVolFlow = 3.14159 * R * viscosity_steam * Di
                                /1000 / 4 / density_steam;
475                         // Calculates head losses with Darcy Weisbach
476                         hf = f * LE / pow(Di/1000,5) * 8 * pow(SteamVolFlow
                                ,2) / 9.8696 / 9.81;
477                         // Reynolds is increased logaritmically for next
                                iteration
478                         R = pow(10,log10(R)+0.15);
479
480
481                 } while (SteamPressure - hf/10.2 > 0.5);
482
483                 // Final mass steam flow is calculated with Darcy-Weisbach
                        equation
484                 SteamFlow = Valve / 100 * density_steam * sqrt(
                        SteamPressure * 10.2 * 9.8696 * pow(Di/1000,5) * 9.81 /
                        f / LE / 8);
485
486                 // Steam mass is decreased taking into account SteamFlow and
                         time elapsed at the end of each loop
487                 SteamMass -= SteamFlow * t_delta/1000;
488                 // To avoid errors in other functions, SteamMass under 0.5
                        is not considered
489                 if(SteamMass < 0.5) SteamMass = 0.0;
490         }
491
492         /*===> Calculates steam pressure by using Van der Vaals equation */
493         void calculateSteamPressure() {
494                 float mol = 0.0;
495
496                 // Volume of steam camera
497                 SteamVolume = v[32] - WaterVolume;
498                 // Mass to steam mol conversion
499                 mol = SteamMass *55.55;
500
501                 //Steam pressure calculated with Van der Vaals equation
502                 SteamPressure = (mol * R * (WaterTemperature+273.15) / (
                        SteamVolume - mol*b)
503                 - a * pow(mol,2) / pow(SteamVolume,2))  /100000;
504         }
```

```
505
506         /*===> Calculates any gas mol by iterating Van der Vaals equation */
507         float calculateMol(float T, float P, float V, float a, float b,
                float R) {
508               // Uses perfect gasses equation to find a seed value for n
509               float n = P*V/R/T, y = 0.0, bb = 0.0, m = 0.0;
510
511               // Calculates n mol by iterating Van de Vaals equation with
                      Newton method
512               do {
513                     m = 3*pow(n,2)*a*b/(pow(n,2)) - 2*n*a/V + R*T-P*b;
514                     y = n*R*T - (P+pow(n,2)*a/(pow(V,2)))*(V-n*b);
515                     bb = y-m*n;
516                     n = -bb/m;
517               } while (fabs(n*R*T - (P+pow(n,2)*a/(pow(V,2)))*(V-n*b)) >
                      100);
518
519               return n;
520         }
521
522         /*===> Calculates water level */
523         void calculateLevel() {
524               // Calculates volumetric flow of inlet water (m^3/h)
525               WaterInletFlow = WaterFlow(SteamPressure, ControlLevel);
526
527               // Calculates mass flow of inlet water (kg/h)
528               FeedWaterMass = ((WaterInletFlow/3600) * (float(t_delta)
                      /1000.0))*FeedWaterDensity;
529
530               // Totalizes boiler water+steam mass (kg)
531               WaterMass += (FeedWaterMass - SteamFlow * float(t_delta)
                      /1000.0);
532
533               // Calculates liquid water volume in boiler
534               WaterVolume = WaterMass/density_liquid;
535
536               // Calculates water level in boiler through FindWaterLevel()
537               WaterLevel = FindWaterLevel(WaterVolume);
538
539               // Calculates indicator level of water level sensor
540               WaterLevelIndicator = WaterLevel * 1000 - WaterLevelMinimum;
541
542               // Simulates water level sensor saturated signal under -30
                      and +30 mm
543               if (WaterLevelIndicator < -30.0) WaterLevelIndicator =
                      -30.0;
544               if (WaterLevelIndicator > 30.0) WaterLevelIndicator = 30.0;
545         }
546
547         /*===> Calculates pressure variation caused when filling the boiler
                */
548         void deltaVolumePressureVariation() {
549               // Volume variation when caused by inlet water is considered
                       Isothermal
550               double V1 = v[32] - WaterVolume;
551               double V2 = V1 -v_delta;
552               double P2 = (SteamPressure + 1) * V1 / V2; //1 atm ~ 1 bar
553
554               if (abs(V2-V1) < 0.1 && V2 > 0.0) { // Avoids errors when
                      calculating volume after reset
555                     SteamPressure = SteamPressure + (float(V1 / V2) - 1)
                            * (SteamPressure + 1);
556               }
```

```
557         }
558
559         /*===> Calculates feedwater flow */
560         float WaterFlow(double Pressure, double ValvePosition) {
561                 float instantValve = ValvePosition;
562
563                 // Simulates a servo action of feedwater valve with an
                        exponential response
564                 if (abs(instantValve - ButterflyValve) < 0.05) {
565                         timeButterflyValve = 0;
566                 } else timeButterflyValve = timeButterflyValve + t_delta;
567                 ButterflyValve = instantValve -
568                 (instantValve - ButterflyValve) *
569                 exp(-1 / K_ButterflyValve * timeButterflyValve);
570
571                 // Total head steam is steam pressure (bar)
572                 double H_steam = SteamPressure;
573
574                 // Equalizes Backpressure and pump-pressure by iterating the
                        intersection
575                 // between pump's and system's H-Q curves. A first range
                        between SteamPressure
576                 // and max pump pressure is used.
577                 double HeadLow = H_steam;
578                 double HeadHigh = FeedWaterMaxPressure; //bar
579                 double HeadMiddle = (HeadHigh - HeadLow) / 2;
580                 double DeltaFlow_d = 0.0;
581                 float Flow = 0.0;
582
583                 // The first range is iterated and reduced by using the sign
                        -change method until
584                 // the calculated flow difference is less than 0.05.
585                 do {
586                         DeltaFlow_d = DeltaFlow(HeadMiddle, H_steam); //Uses
                                DeltaFlow() to find the difference
587                         if (DeltaFlow_d > 0) HeadLow = HeadMiddle;
588                         if (DeltaFlow_d < 0) HeadHigh = HeadMiddle;
589                         HeadMiddle = HeadLow + (HeadHigh - HeadLow) / 2;
590                 } while (abs(DeltaFlow_d) > 0.05);
591
592                 //Calculates volumetric feedwater flow
593                 Flow = -0.0000167886871355758 * pow(HeadMiddle, 5) +
                        0.0010251972 * pow(HeadMiddle, 4) - 0.0223995353 * pow(
                        HeadMiddle, 3) +
594                 0.192711983 * pow(HeadMiddle, 2) - 1.0778854747 * HeadMiddle
                         + 33.152;
595                 if (Flow < 0.5) Flow = 0.0; //If flow is very low, 0 is
                        assigned to ensure valve closing.
596
597                 return float(Flow);
598         }
599
600         /*===> Calculates feed water flow diference for butterfly valve
601         and pump H-Q curves */
602         double DeltaFlow(float HeadPump, float H_steam) { //HeadPump &
                H_steam in bar
603                 // Calculates the headloss constant K for butterfly
                        feedwater valve
604                 float K = 0.1561 * exp(0.1095 * (90 - ButterflyValve));
605
606                 double Flow_v = 0.0, Flow_p = 0.0, Delta_F = 0.0;
607
608                 if (ButterflyValve == 0.0) {
```

```
609                     Flow_v = 0;
610             } else if ((HeadPump - H_steam) > 0.0) { //Ensures
                    difference is positive to avoid errors in reset
611                     // Calculates flow with butterfly valve H-Q curve
612                     Flow_v = 3.14159 / 4 * pow(PipeDiameter, 2) * sqrt((
                        HeadPump - H_steam)) * 2 * 9.81 / K * 3600;
613             } else Flow_v = 0.0;
614
615             // Calculates flow for pump H-Q curve
616             Flow_p = -0.0000167886871355758 * pow(HeadPump, 5) +
                    0.0010251972 * pow(HeadPump, 4) - 0.0223995353 *
617             pow(HeadPump, 3) + 0.192711983 * HeadPump * HeadPump -
                    1.0778854747 * HeadPump + 33.152;
618
619             return Flow_p - Flow_v;
620     }
621
622     /*===> Calculates water height in boiler by interpolating data in a
623     lookup table (LUT) */
624     float FindWaterLevel(float Volume) {
625             int i = 0;
626
627             // LUT defines levels in m a loop is used to find a range
                    volume for
628             // actual volume of water in boiler.
629             if (Volume) { // Loop is only started if Volume > 0
630                     for (i = 0; i < 33; i++) {
631                             if ((v[i] < Volume) && (v[i + 1] > Volume))
                                    break;
632                     }
633
634                     // Uses a linear interpolation to estimate water
                            height
635                     if (i > 32) return h[32];
636                     else {
637                             return interpolate(Volume, h[i], h[i + 1], v
                                    [i], v[i + 1]);
638                     }
639             } else return 0.0;
640     }
641
642     /*===> Controls burner status and simulates a purging cycle */
643     void burnerControl() {
644             if (!Alarm && FlagBurnerON != 3 && ControlBurner > 0  && !
                    digitalRead(InputReset)) {
645                     if (t_burnerStart < 1000 * burnerStartupTime) { //
                            Simulates purging
646                             FlagBurnerON = 1;
647                             t_burnerStart = t_burnerStart + t_delta;
648                     } else { // Starts burner after purge
649                             FlagPurgeStart = 3;
650                             t_burnerStart = 0;
651                     }
652             }
653
654             // If alarm raises or power reaches minimum position, burner
                     stops
655             if (Alarm || ControlBurner < burnerMinimum) {
656                     FlagBurnerON = 0; // Resets burner flag
657                     t_burnerStart = 0; // Resets purging time
658                     FlagPurgeStart = 0; // Resets purging flag
659             }
660     }
```

```
661
662         /*===> Calculates power transferred to water and thermal losses */
663         void calculatePower() {
664                 float instantBurner;
665                 int K; // Multiplier of time constant to simulate delay of
                        burner servo-actuator
666
667                 // Moves burner control to 0% when purge process is
                        activated
668                 if (FlagBurnerON == 1) {
669                         if (FlagPurgeStart == 0) {
670                                 instantBurner = 0.0;
671                                 K = 1; // Slower
672                                 // Purge process is only started if burner
                                        control is at minimum
673                                 if (Burner < 0.05) FlagPurgeStart = 1;
674                         }
675                         // Moves burner control to 100% to start purge
676                         if (FlagPurgeStart == 1) {
677                                 instantBurner = 100.0;
678                                 K = 4; // Faster
679                         }
680                         // Burner control moves to minimum power to start
                                burner
681                         if (FlagPurgeStart == 3) {
682                                 instantBurner = burnerMinimum;
683                                 K = 1; // Slower
684                                 // Purge sequence is finished when burner is
                                        started
685                                 if (abs(Burner - burnerMinimum) < 0.05) {
686                                         FlagBurnerON = 2;
687                                 }
688                         }
689                 }
690
691                 if (FlagBurnerON == 2) { // Controls ignition timing
692                         K = 1; // Slower
693                         t_burnerIgnition = t_burnerIgnition + t_delta;
694                         // Control is back to user when ignition time ends
695                         if (t_burnerIgnition >= tBurnerIgnition * 1000) {
696                                 FlagBurnerON = 3;
697                                 t_burnerIgnition = 0;
698                         }
699                 }
700
701                 // Transferes burner control to potentiometer or external
702                 // controller when burner is ignited
703                 if (FlagBurnerON == 3) {
704                         instantBurner = ControlBurner;
705                         K = 1; // Slower
706                 }
707
708                 // Transferes burner control to potentiometer or external
709                 // controller when burner off
710                 if (FlagBurnerON == 0) {
711                         instantBurner = ControlBurner;
712                         K = 1; // Slower
713                 }
714
715                 // Simulates a servo delay of burner position by following
                        an
716                 // exponential behaviour when difference between command and
717                 // real position is more than 0.05%
```

```
718                 if (abs((instantBurner - Burner)) < 0.05) {
719                         timeBurner = 0;
720                 } else  timeBurner = timeBurner + t_delta;
721                 Burner = instantBurner - (instantBurner - Burner) * (exp(-1
                        / (K * K_Burner) * timeBurner));
722
723                 // Calculates power transferred if burner is ignited
724                 // This version does not apply an efficiency curve
725                 if (FlagBurnerON >= 2) {
726                         etaBurner = Burner;
727                 } else etaBurner = 0.0;
728                 // Thermal losses are not calculated in this version
729                 PowerTransferred = Efficiency/100 * PMax;
730
731         }
732
733
734         /*===> Calculates thermal losses in insulation */
735         void calculateThermalLosses(){
736                 /*===> Calculates thermal loses */
737                 // pi number pow 2
738                 pi2 = pow(pi, 2);
739                 //cilinder area
740                 BoilerCylindricalSurface = 2*pi*internalBoilerWRad*
                        BoilerLength;
741                 //cover circunference area
742                 BoilerCoverSurface= pi*pow(CoverRadius,2);
743                 //boiler cilinder area
744                 BoilerExternalCylindricalrSurface= 2*pi*externalBoilerRad*
                        BoilerLength;
745                 //boiler steel thickness
746                 BoilerSteelThickness= externalBoilerWRad -
                        internalBoilerWRad;
747                 //boiler rockwool thickness
748                 BoilerIsolationThickness= externalBoilerThermalIsoRad -
                        externalBoilerWRad;
749                 //boiler isolation protector (aluminium) thickness
750                 BoilerIsolationProtectorThickness= externalBoilerRad -
                        externalBoilerThermalIsoRad;
751
752                 //boiler body thermal resistance
753                 Rtotalcyl = ((1/(K_H2O*BoilerCylindricalSurface)) + ((log(
                        externalBoilerWRad/internalBoilerWRad)))/(2*pi*
                        BoilerLength*SteelThermalCoefficient)) + ((log(
                        externalBoilerThermalIsoRad/externalBoilerWRad))/(2*pi*
                        BoilerLength*RockWoolThermalCoefficient)) + ((log(
                        externalBoilerRad/externalBoilerThermalIsoRad))/(2*pi*
                        BoilerLength*AluminiumlThermalCoefficient)) + (1/(K_Air*
                        BoilerExternalCylindricalrSurface)); //(ÂºC/W)A1=A4=2*pi
                        *r*l
754                 //lost heat boiler body insulation
755                 Qperd1 = (WaterTemperature-EnvironmentTemperature)/Rtotalcyl
                        ;//(W)
756                 //boiler front cover thermal resistance
757                 Rtotalfront = (1/(K_H2O*BoilerCoverSurface)) + (
                        BoilerSteelThickness/(SteelThermalCoefficient*
                        BoilerCoverSurface)) + (BoilerIsolationThickness/(
                        RockWoolThermalCoefficient*BoilerCoverSurface)) + (
                        BoilerIsolationProtectorThickness/(
                        AluminiumlThermalCoefficient*BoilerCoverSurface)) + (1/(
                        K_Air*BoilerCoverSurface)); //(ÂºC/W)
758                 //boiler rear cover thermal resistance is the same as the
                        front cover
```

```
759                 Rtotalrear = Rtotalfront; //(ÂºC/W)
760                 //lost heat insulationr front cover
761                 Qperd2 = (WaterTemperature-EnvironmentTemperature)/
                        Rtotalfront;//(W)
762                 //lost heat insulation rear cover
763                 Qperd3 = Qperd2;//(W)
764                 //total heat lost through insulation
765                 LosePower = ((Qperd1 + Qperd2 + Qperd3)/1000);// power
                        losses. More simply Qtot = (tinf1-tinf2)/rtotal;
766                 //boiler thermal resistance
767                 Rboiler = Rtotalcyl + Rtotalfront + Rtotalrear; // es
                        necesario? resistencia termica
768
769         }
770         /*===> Calculate exchange surface */
771         void exchangeSurface(){
772                 //Calculate some dimensions of the boiler
773                 //Internal furnace radius
774                 Internalfurnrad = FurnaceDiameter / 2;
775                 //External furnace radius
776                 Externalfurnrad = Internalfurnrad + MaterialThickness;
777                 //Internal radius of the pipes of the second fume pass
778                 IntsecSteprad = tubeDiameter / 2;
779                 //Extternal radius of the pipes of the second fume pass
780                 ExtsecSteprad = IntsecSteprad + MaterialThickness;
781                 //Internal radius of the pipes of the third fume pass
782                 IntthiSteprad = IntsecSteprad;
783                 //External radius of the pipes of the third fume pass
784                 ExtthiSteprad = ExtsecSteprad;
785                 //Furnace surface
786                 Furnacesurface = 2 * pi * Internalfurnrad * furnaceLength;
787                 //Second fume pass surface
788                 secStepsurface = 2 * pi * IntsecSteprad * secStepLength;
789                 //Third fume pass surface
790                 thiStepsurface = 2 * pi * IntthiSteprad * thiStepLength;
791                 //Total boiler exchange surface
792                 ExchangeSurface = Furnacesurface + (nTubSecStep*
                        secStepsurface) + (nTubThiStep*thiStepsurface);
793
794         }
795         /*===> Calculate the boiler furnace temperature */
796         void calculateFurnaceTemperature(){
797
798                 BoilerLoad = etaBurner;
799                 if(BoilerLoad>0){
800                         float ExcessAir = 30;//Potenciometro
801                         //Calculate fuel flow
802                         FuelFlow = ((PMax*(BoilerLoad/100))/LowCalVal);
803                         //Calculate air excess
804                         nExcess = (ExcessAir/100)+1;
805                         //Reactive combustion products flow
806                         MOLFlowFuel = (FuelFlow/Fuel_molecWeight);
807                         MOLO2stoichiometric = MOLFlowFuel*(((61/2)*nExcess)/
                                nExcess);
808                         MOLO2stoichiometricFlow = (MOLO2stoichiometric *(
                                O_AtomicWeight*2)/1000);
809                         //Calculate no reactive exhaust gases flow with air
                                excess
810                         O2Flow = ((MOLFlowFuel*((61/2)*nExcess)*(
                                O_AtomicWeight*2))/1000);
811                         N2Flow = (((78.06/20.98)*((61/2)*nExcess))*
                                MOLFlowFuel*(N_AtomicWeight*2)/1000);
```

```
812                         CO2Flow = ((((0.04/20.98)*((61/2)*nExcess))*
                                MOLFlowFuel*(CO2_molecWeight*1000))/1000);
813                         //Calculate reactive exhaust gases flow with air
                                excess
814                         CO2Flowreactive = 20*MOLFlowFuel*CO2_molecWeight;
815                         H2OFlowreactive = 21*MOLFlowFuel*H2O_molecWeight;
816                         N2Flownonreactive = N2Flow;
817                         CO2Flownonreactive = CO2Flow;
818                         O2Flownonreactive = (((61/2)*(nExcess-1))*
                                MOLFlowFuel*((O_AtomicWeight*2))/1000);
819                         //Equation for calculate furnace temperature
820                         FurnaceTemp = (((((CO2Flowreactive*CO2SH)+(
                                H2OFlowreactive*H2OSH))*
                                H_FuelAdiabaticTemperature)+(((N2Flownonreactive
                                *N2SH)+(O2Flownonreactive*O2SH)+(
                                CO2Flownonreactive*CO2SH))*
                                EnvironmentTemperature))/((CO2Flowreactive*CO2SH
                                )+(H2OFlowreactive*H2OSH)+(N2Flownonreactive*
                                N2SH)+(O2Flownonreactive*O2SH)+(
                                CO2Flownonreactive*CO2SH)));
821                 }else {FurnaceTemp = WaterTemperature;}
822
823         }
824
825         /*===> Calculate boiler efficiency */
826         void calculateBoilerEfficiency(){
827                 BoilerLoad = etaBurner;
828                 //calculate efficiency
829                 if(BoilerLoad > 0.194){
830                         Efficiency = 13.741*log(BoilerLoad)+22.563;//Curve
                                Efficiency
831                 }else {Efficiency = 0;}
832
833         }
834
835         /*===> Calculate exhaust temperature */
836         void calculateExhaustTemperature(){
837                 //variables de funcionamiento
838
839
840                 CoeffU = (0.050760);//kW/m2Â°C
841                 //Temperatura media logarítmica
842                 LMTreal = (PowerTransferred/(ExchangeSurface*CoeffU));//Â°C
843
844
845                 float LowValue = WaterTemperature; //double
846                 float HighValue = FurnaceTemp;  //double
847                 ExhaustTemp = (FurnaceTemp - WaterTemperature) / 2;
848
849                 float DeltaTemp = 0.0;//double
850                 float ValidOperation = 0;
851                 /* if (FurnaceTemp > WaterTemperature){ ValidOperation = (
                        FurnaceTemp>WaterTemperature)&&(ExhaustTemp>FurnaceTemp)
                        ;
852                 }else{ExhaustTemp = WaterTemperature;}*/
853
854                 // The first range is iterated and reduced by using the sign
                        -change method until
855                 // the calculated difference is less than 0.05.
856                 if((FurnaceTemp>(WaterTemperature+10))&& (etaBurner>23)){
857
858
859
```

```
860                         do {
861                                 LMTreal = (PowerTransferred/(ExchangeSurface
                                        *CoeffU));//Âºc
862                                 LMTiterated= (((FurnaceTemp-WaterTemperature
                                        )-(ExhaustTemp-FeedWaterTemperature))/
863                                 (log((FurnaceTemp-WaterTemperature)/(
                                        ExhaustTemp-FeedWaterTemperature))));//
                                        Âºc
864
865                                 if (LMTiterated < LMTreal) {LowValue =
                                        ExhaustTemp;}
866                                 else{ HighValue = ExhaustTemp;}
867                                 DeltaTemp = LMTreal-LMTiterated;
868
869                                 ExhaustTemp = LowValue + (HighValue -
                                        LowValue) / 2;
870
871                         }while ((abs(DeltaTemp) > 0.05));
872                 }else{ExhaustTemp = WaterTemperature;}
873
874
875
876         }
877
878         /*===> Calculate thermodynamic variables for actual boiler pressure
                ***/
879         void calculateThermodynamicVariables(){
880                 H_water_b = h_water;
881
882                 // Finds minimum and maximum values to interpolate
883                 int i=0;
884                 for(i=0; i<67; i++) {
885                         if ((Pa[i]<=SteamPressure+1) && (Pa[i+1]>
                                SteamPressure+1)) break;
886                 }
887                 //Thermodynamic variables calculated from actual pressure
888                 if((Pa[i]<=SteamPressure+1) && (Pa[i+1]>SteamPressure+1)) {
                        //Manometric pressure +1
889                         T_saturate = interpolate(SteamPressure+1,Ts[i],Ts[i
                                +1],Pa[i],Pa[i+1]);
890                         density_steam = interpolate(SteamPressure+1,dv[i],dv
                                [i+1],Pa[i],Pa[i+1]);
891                         viscosity_steam = interpolate(SteamPressure+1,vd[i],
                                vd[i+1],Pa[i],Pa[i+1]);
892                         h_water = interpolate(SteamPressure+1,Hl[i],Hl[i+1],
                                Pa[i],Pa[i+1]);
893                         h_steam = interpolate(SteamPressure+1,Hv[i],Hv[i+1],
                                Pa[i],Pa[i+1]);
894                         L_ws = interpolate(SteamPressure+1,L[i],L[i+1],Pa[i
                                ],Pa[i+1]);
895                         c_water = interpolate(SteamPressure+1,Ce[i],Ce[i+1],
                                Pa[i],Pa[i+1]);
896                 }
897
898
899                 //Thermodynamic variables calculated from actual temperature
900                 for(i=0; i<67; i++) {
901                         if ((Ts[i]<=WaterTemperature) && (Ts[i+1]>
                                WaterTemperature)) break;
902                 }
903                 density_liquid = interpolate(WaterTemperature,dl[i],dl[i+1],
                        Ts[i],Ts[i+1]);
```

```
904                 P_saturate = interpolate(WaterTemperature,Pa[i],Pa[i+1],Ts[i
                        ],Ts[i+1]);
905         }
906
907         /*===> Calculates the mass of steam generated,
908         and pressure and temperature variations */
909         void steamGeneration(){
910
911                 float SteamMassSaturation = 0.0, FlashSteam = 0.0,
                        SteamMassBefore = 0.0;
912                 // double deltaTemp = 0.0;
913
914                 //NetEnergy transferred to water (KJ)
915
916                 NetEnergy = (float(PowerTransferred) - float(LosePower)) *
                        float(t_delta)/1000.0;
917
918
919                 // En caso de NetEnergy ser negativo limita la salida a 0 ya
                         que si no se transfiere potencia pero hay perdidas el
                        programa se bloquea
920
921                 if (NetEnergy < 0){
922                         NetEnergy = 0;
923                 }
924
925                 // Case A: Temperature of water < Saturation temperature at
                        actual pressure
926                 if (WaterTemperature <= T_saturate) {
927                         // Calculates energy required to reach saturation
                            temperature
928                         if((T_saturate - WaterTemperature) < 0.05)
                            HeatEnergy = 0;
929                         else HeatEnergy = (float(T_saturate) - float(
                            WaterTemperature)) *
930                         float(c_water) * float(WaterMass);
931
932                         if (NetEnergy>HeatEnergy) { // When there is more
                            energy than required to saturation
933                                 WaterTemperature=T_saturate; // Temperature
                                    reaches saturation temperature
934                                 SurplusEnergy = NetEnergy-HeatEnergy; //
                                    Excess of energy is calculated
935                                 SteamMass += SurplusEnergy/L_ws; // Excess
                                    of energy adds steam mass
936                                 WaterVolume -= SurplusEnergy/L_ws/
                                    density_liquid; // Liquid water
                                    decreases
937                         } else {  // When there is less energy than required
                             for saturation
938                                 if (WaterMass) { // Only if there is water
939                                         WaterTemperature += NetEnergy/(
                                            c_water*WaterMass); //
                                            Temperature increases
940                                         // Feedwater affects water
                                            temperature
941                                         WaterTemperature = (WaterMass *
                                            c_water * (WaterTemperature +
                                            273.15) +
942                                         FeedWaterMass * C_FeedWater * (
                                            FeedWaterTemperature + 273.15))
                                            /
```

```
943                                         (WaterMass * c_water + FeedWaterMass
                                                * C_FeedWater) - 273.15;
944                                 } else WaterTemperature =
                                        FeedWaterTemperature;
945                         }
946                 }
947
948                 // Case B: Temperature of water > Saturation temperature at
                        actual pressure
949                 if (h_water<H_water_b){
950                         //Mass of steam generated
951                         FlashSteam = (H_water_b-h_water) / L_ws * WaterMass;
952                         SteamMassSaturation = calculateMol(WaterTemperature
                            + 273,
953                         P_saturate*100000, SteamVolume, a, b, R) * 18 /
                            1000;
954                         // Total steam mass cannot overcome
955                         // the corresponding amount of steam at actual
                            temperature
956                         if((SteamMass + FlashSteam) < SteamMassSaturation) {
957                                 FlashSteam = (H_water_b-h_water) / L_ws *
                                    WaterMass + NetEnergy / L_ws;
958                                 SteamMass += FlashSteam; // All excess of
                                    energy produces steam
959                         } else if (WaterMass){ // Maximum mass of steam is
                            generated.
960                                 SteamMassBefore = SteamMass;
961                                 SteamMass = SteamMassSaturation; //Steam
                                    mass reaches max mass at saturation
962                                 FlashSteam = SteamMass - SteamMassBefore;
963                                 //Excess of energy from burner is used to
                                    heat water
964                                 WaterTemperature += NetEnergy/c_water/
                                    WaterMass;
965                         }
966
967                         // Water temperature is recalculated
968                         WaterTemperature = ((WaterMass - FlashSteam)*
                            c_water * (WaterTemperature + 273.15) +
969                         FeedWaterMass * C_FeedWater * (FeedWaterTemperature
                            + 273.15)) /
970                         (WaterMass * c_water + FeedWaterMass * C_FeedWater)
                            - 273.15;
971                 }
972         }
973
974         /*===> Controls boiler blockage caused by too high pressure or very
                low water level */
975         void blockageControl() {
976                 if (WaterLevelIndicator < -30.0 || WaterLevelIndicator >
                    29.9 || SteamPressure > SteamMaxPressure) {// modificado
                     para simulación
                    ------------------------------------------------------
977                         Alarm = HIGH;
978                 }
979         }
980
981         /*===> Controls reset operation */
982         void controlReset() {
983                 int resetValue = 0, blinkCount = 0;
984
985                 // Tries to reset blockage when pushed shortly
986                 if (resetValue = digitalRead(InputReset)) {
```

```
987                         Alarm = LOW;
988                         t_reset = t_reset + t_delta;
989                 } else {
990                         t_reset = 0;
991                 }
992
993                 if (t_reset > 5000) { // Changes operating condition when
                        pushed more than 5s
994                         do { // Blinks alarm LED 5 times
995                                 digitalWrite(LedBoilerAlarm, LOW);
996                                 digitalWrite(LedBoilerNormal, HIGH);
997                                 delay(100);
998                                 digitalWrite(LedBoilerNormal, LOW);
999                                 digitalWrite(LedBoilerAlarm, HIGH);
1000                                delay(100);
1001                                blinkCount++;
1002                        } while (blinkCount < 6);
1003                        // Sets variables to working operating condition
1004                        WaterVolume = WaterWorkingVolume;
1005                        SteamPressure = SteamWorkingPressure;
1006                        WaterTemperature = WaterWorkingTemperature;
1007                        calculateThermodynamicVariables();
1008                        WaterMass = WaterVolume*density_liquid;  //
                            Recalculates water mass
1009                        FeedWaterMass = ((WaterInletFlow/3600) * (float(
                            t_delta)/1000.0))*FeedWaterDensity;
1010                        //Masa de vapor (kg)
1011                        SteamMass = SteamWorkingMass; // Recalculates steam
                            mass
1012                        lcd.clear();
1013                        Alarm = HIGH; //Sets alarm to HIGH to avoid burner
                            start operation
1014                }
1015        }
1016
1017        /*===> Writes data to digital PINs */
1018        void writeDigitalPorts() {
1019                // Alarm LED status
1020                digitalWrite(LedBoilerAlarm, Alarm);
1021
1022                // Water-level LED status
1023                if (WaterLevelIndicator < -20.0) {
1024                        digitalWrite(LedLowLevel, HIGH);
1025                        digitalWrite(LedRightLevel, LOW);
1026                } else if (WaterLevelIndicator >= -20.0 &&
                        WaterLevelIndicator < 20.0) {
1027                        digitalWrite(LedLowLevel, LOW);
1028                        digitalWrite(LedRightLevel, HIGH);
1029                } else if (WaterLevelIndicator >= 20.0) {
1030                        digitalWrite(LedHighLevel, HIGH);
1031                        digitalWrite(LedRightLevel, LOW);
1032                }
1033
1034                // Boiler pressure LED status
1035                if (SteamPressure > SteamMaxPressure) {
1036                        digitalWrite(LedPressureOk, LOW);
1037                        digitalWrite(LedHighPressure, HIGH);
1038                } else if (SteamPressure < SteamMinPressure) {
1039                        if (t_flashP >= 0 && t_flashP < 1000) { //El LED
                            parpadea cada segundo
1040                                digitalWrite(LedPressureOk, HIGH);
1041                                digitalWrite(LedHighPressure, LOW);
1042                                t_flashP = t_flashP + t_delta;
```

```
1043                        } else if (t_flashP > 1000 && t_flashP < 2000) {
1044                                digitalWrite(LedPressureOk, LOW);
1045                                t_flashP = t_flashP + t_delta;
1046                        } else t_flashP = 0;
1047                } else {
1048                        digitalWrite(LedPressureOk, HIGH);
1049                        digitalWrite(LedHighPressure, LOW);
1050                }
1051
1052                // Burner LED status
1053                if (FlagBurnerON == 0) {
1054                        digitalWrite(LedBurnerOn, LOW);
1055                } else if (FlagBurnerON == 1) {
1056                        if (t_flashB < 500) { //El LED parpadea cada segundo
1057                                digitalWrite(LedBurnerOn, HIGH);
1058                                t_flashB = t_flashB + t_delta;
1059                        } else if (t_flashB > 500 && t_flashB < 1000) {
1060                                digitalWrite(LedBurnerOn, LOW);
1061                                t_flashB = t_flashB + t_delta;
1062                        } else t_flashB = 0;
1063                } else digitalWrite(LedBurnerOn, HIGH);
1064
1065
1066
1067                //Sends PWM Level Signal to OutLevel PIN (-30 ~ +30)
1068                analogWrite(OutLevel, int((WaterLevelIndicator + 30) / 60 *
                        254));
1069                //Sends PWM Pressure Signal to OutPressure (0 ~ 20)
1070                analogWrite(OutPressure, int(SteamPressure/20*254));
1071        }
1072
1073        /*===> Interpolates data */
1074        float interpolate(float value, float LL, float LH, float VL, float
                VH) {
1075                return LL + (value - VL) * (LH - LL) / (VH - VL);
1076        }
1077
1078        /*===> Process PWM signals */
1079        void processPin(byte pin) {
1080                uint8_t trigger = getPinChangeInterruptTrigger(
                        digitalPinToPCINT(channel_pin[pin]));
1081
1082                if(trigger == RISING) {
1083                        rising_start[pin] = micros();
1084                } else if(trigger == FALLING) {
1085                        channel_length[pin] = micros() - rising_start[pin];
1086                }
1087        }
1088
1089        /*===> Detects ascending PWM signal on channel_pin[0] */
1090        void onRising0(void) {
1091                processPin(0);
1092        }
1093
1094        /*===> Detects ascending PWM signal on channel_pin[1] */
1095        void onRising1(void) {
1096                processPin(1);
1097        }
```

B.2 Archivos de configuración

B.2.1 Parámetros y constantes

```
1  //PIN definition
2
3  //Auto-manual switch
4  #define InputBurnerManual        A15
5  //Feed water manual control potentiometer
6  #define InputWaterManual         A14
7  //Puerto entrada apertura de la válvula principal de vapor
8  #define InputMainValve           A13
9  //Puerto salida presión
10 #define OutPressure              8
11 //Puerto salida Nivel
12 #define OutLevel                          9
13 //Puerto salida Quemador ON
14 #define LedBurnerOn                       28
15 //Puerto salida nivel correcto
16 #define LedRightLevel            23
17 //Puerto salida Alarma nivel
18 #define LedLowLevel                       22
19 #define LedHighLevel             22
20 //Puerto salida Alarma presión
21 #define LedHighPressure          24
22 //Puerto salida presión correcta
23 #define LedPressureOk            25
24 //Puerto salida Bloqueo caldera
25 #define LedBoilerAlarm           26
26 //Puerto salida caldera operativa
27 #define LedBoilerNormal          27
28 //Puerto entrada Reset
29 #define InputReset                        38
30 #define Display_1        53
31 #define Display_2        52
32 //Puerto entrada Auto/Man
33 #define InputAutoManual          39
34
35
36
37 //Parámetros para el reseteo en condiciones de operación
38 //Volumen normal de operación de la caldera (para la función de
       reseteo)
39 #define WaterWorkingVolume     27.36 //0 mm
40 //#define WaterWorkingVolume  26.8 // -30 mm
41 #define SteamWorkingPressure  16.0
42 #define WaterWorkingTemperature 204.0
43 #define SteamWorkingMass    63.0
44
45
46 //Definición de parámetros de la caldera
47 //Caudal máximo de vapor en T/h
48 #define SteamMaxFlow                      10.0
49 //Potencia máxima kW
50 #define PMax                                    6563
51 //Nivel mínimo de agua (mm)
52 #define WaterLevelMinimum                 2500.0
53 //Densidad del agua de alimentación (kg/m3)
54 #define FeedWaterDensity                  992.25
55 //Temperatura del agua de alimentación (Â ºC)
56 #define FeedWaterTemperature     40
57 //Temperatura ambiente (Â ºC)
```

```
58          #define EnvironmentTemperature  25.0
59          //Diámetro de la tubería de agua de alimantación (m)
60          #define PipeDiameter                    0.065
61          //Presión máxima de la bomba de agua de alimentación (bar)
62          #define FeedWaterMaxPressure    28.715
63          //Presión máxima de trabajo de la caldera (bar)
64          #define SteamMaxPressure                20.0
65          //Presión mínima de trabajo de la caldera (bar)
66          #define SteamMinPressure                12.0
67          //Tiempo del proceso de barrido (s)
68          #define burnerStartupTime               30
69          //Constante de tiempo para el control del quemador (ms)
70          #define K_Burner                                    50000.0
71          //Posición mínimo para el arranque del quemador (%)
72          #define burnerMinimum                   5.0
73          //Tiempo de ignición (s)
74          #define tBurnerIgnition                 3.0
75          //Constante de tiempo de la válvula de agua de alimentación (ms)
76          #define K_ButterflyValve                20000.0
77          //Potencia perdidas por aislamiento
78          //#define LosePower                                 0 {ha sido
                modificada a variable en BoilerControlVariables.h}
79
80
81          //Definición de las características de la instalación
82          //Tuberias
83          //Diámetro interior de la tubería (mm)
84          #define Di              220.0
85          //Presión nominal de la tubería: 20bar
86          //Longitud equivalente (m)
87          #define LE              250.0
88
89          //Dimensiones de la caldera
90          //Número de tubos del segundo paso de humos
91          #define nTubSecStep   150
92          //Numero de tubos del tercer paso de humos
93          #define nTubThiStep   84
94          //Diámetro del quemador (m)
95          #define FurnaceDiameter   1.31
96          //Diámetro de la caldera (m)
97          #define boilerDiameter    3.2
98          //Diámetro de los tubos de paso de humos (m)
99          #define tubeDiameter    0.061
100         //Longitud del hogar de la caldera (m)
101         #define furnaceLength   5.638
102         //Longitud de los tubos del segundo paso de humos (m)
103         #define secStepLength   4.724
104         //Longitud de los tubes del tercer paso de humos (m)
105         #define thiStepLength   5.94
106         //espesor del acero de la caldera (m)
107         #define MaterialThickness   0.008
108         //Longitud de la caldera (m)
109         #define BoilerLength    6.2
110         //Internal boiler wall radius (mm)
111         #define internalBoilerWRad   1.491
112         //External boiler wall radius (mm)
113         #define externalBoilerWRad   1.499
114         //External boiler thermal isolation radius (mm)
115         #define externalBoilerThermalIsoRad   1.599
116         //External boiler radius (mm)
117         #define externalBoilerRad   1.6
118         //Front and rear cover radius (m)
119         #define CoverRadius   1.6
```

```
120
121         //Fuel data
122         #define LowCalVal  39765//kJ/kg
123
124         //Adiabatic temperatures
125         //Light fuel oil (ÂºC)
126         #define L_FuelAdiabaticTemperature  2104
127         //Medium fuel oil (ÂºC)
128         #define M_FuelAdiabaticTemperature  2101
129         //Heavy fuel oil (ÂºC)
130         #define H_FuelAdiabaticTemperature  2102
131
132
133         //Chemical Data
134         //Carbon (g/mol)
135         #define C_AtomicWeight  12
136         //Hydrogen (g/mol)
137         #define H_AtomicWeight  1.008
138         //Nitrogen (g/mol)
139         #define N_AtomicWeight  14
140         //Oxygen (g/mol)
141         #define O_AtomicWeight  16
142         //Fuel (kg/mol)
143         #define Fuel_molecWeight  0.28233
144         //Carbon dioxide (kg/mol)
145         #define CO2_molecWeight  0.044
146         //Water (kg/mol)
147         #define H2O_molecWeight  0.018016
148
149         //Specifics heats
150         //Nitrogen (kJ/kgÂºC)
151         #define N2SH  1.04
152         //Oxygen (kJ/kgÂºC)
153         #define O2SH  0.918
154         //Fuel (kJ/kgÂºC)
155         #define FuelSH  1.5899
156         //Carbon dioxide  (kJ/kgÂºC)
157         #define CO2SH  0.8439
158         //Water (kJ/kgÂºC)
159         #define H2OSH  4.18
160
161         //Definición de coeficientes termicos
162         //Water Thermal Coefficient (W/m2Â·K)
163         #define K_H2O   62
164         //Air Thermal Coefficient (W/m2Â·K)
165         #define K_Air   18
166         //Exhaust gas Thermal Coefficient (W/m2Â·K)
167         #define KExhaustGas   280
168         //Steel Thermal Coefficient (W/mÂ·K)
169         #define SteelThermalCoefficient   50.20
170         //Rock Wool Thermal Coefficient (W/mÂ·K)
171         #define RockWoolThermalCoefficient   0.0410
172         //Aluminiun Thermal Coefficient (W/mÂ·K)
173         #define AluminiumlThermalCoefficient   209.3
174
175         //Definición de constantes
176         //numero pi
177         #define pi    3.1416
```

B.2.2 Variables

```
1
```

```
2           //Definición de variables de control
3           //Valor del control del quemador
4           float ControlBurner = 0.0;
5           //Valor del control de nivel
6           float ControlLevel = 0.0;
7           //Control del bloqueo de la caldera
8           int Alarm = HIGH;
9           //Estado del pulsador de rearme
10          int Reset = LOW;
11          //LCD change function
12          int Value1 = LOW;
13          int Value2 = LOW;
14          //Apertura válvula principal de vapor
15          int MainValve = 0;
16          //Indicador automático/manual HIGH:Auto, LOW:Manual
17          int FlagBurner = LOW, FlagWater = LOW;
18          //Indicador del estado del quemador
19          int FlagBurnerON = 0; //0:Apagado  1:Barrido  2:Encendido
20          //Indicador del estado de la purga de gases
21          int FlagPurgeStart = 0; //0:Requiere reposicionase  1:Puede
                iniciarse  3:Barrido finalizado
22          //Indicador estado Auto/Manual
23          int FlagAutoManual = 0; //0: Manual  1: Auto
24
25
26          //Definición de variables de funcionamiento
27          //Volumen de agua (m3)
28          float WaterVolume = 26.8;//0.0 modificado
29          //Nivel del agua (m)
30          float WaterLevel = 0.0;
31          //Nivel indicador (mm)
32          float WaterLevelIndicator = 0.0;
33          //Temperatura del agua (ÂºC)
34          double WaterTemperature = 25.;
35          //Presión de vapor (bar)
36          float SteamPressure = 0.0;
37          //Caudal del agua de alimentación
38          float WaterInletFlow = 0.0;
39          //Caudal de vapor (m3/s)
40          float SteamOutletFlow = 0.0;
41          //Potencia perdidas por aislamiento
42          float LosePower = 0.0;
43          //Rendimiento del quemador
44          float etaBurner = 0.0;
45          //Carga de la caldera
46          float BoilerLoad = 0;
47          //Potencia transferida al agua (kW)
48          float PowerTransferred = 0.0;
49          //Calor transferido al agua (kWs)
50          float HeatTransferred = 0.0;
51          //Flujo de vapor
52          float SteamFlow = 0.0;
53
54          //Variables cálculo de pérdidas térmicas
55          //pi number pow 2
56          float pi2 = pow(pi, 2);
57          //Cilinder area
58          float BoilerCylindricalSurface = 0.0;
59          //Circunference area
60          float BoilerCoverSurface = 0.0;
61          //Cilinder area
62          float BoilerExternalCylindricalrSurface = 0.0;
63          //Boiler steel thickness
```

```
64          float BoilerSteelThickness = 0.0;
65          //Boiler isolation thickness
66          float BoilerIsolationThickness = 0.0;
67          //Boiler isolation protector thickness
68          float BoilerIsolationProtectorThickness = 0.0;
69
70          //Boiler body thermal resistance (W)
71          float Rtotalcyl = 0.0;
72          //Lost heat boiler body insulation
73          float Qperd1 = 0.0;
74          //Boiler front cover thermal resistance (Â ºC/W)
75          float Rtotalfront = 0.0;
76          //Boiler rear cover thermal resistance (Â ºC/W)
77          float Rtotalrear = 0.0;
78          //Lost heat insulationr front cover (W)
79          float Qperd2 = 0.0;//(W)
80          //Lost heat insulation rear cover (W)
81          float Qperd3 = 0.0;//(W)
82          //Boiler thermal resistance (Â ºC/W)
83          float Rboiler = 0.0;
84
85
86
87          //Definición de variables de simulación
88          //Tiempo de inicio de bucle
89          unsigned long t_init = 0;
90          //Tiempo de bucle
91          unsigned long t_delta = 0;
92          //Tiempo de reset
93          unsigned long t_reset = 0;
94          //Tiempo de parpadeo LED presión
95          unsigned long t_flashP = 0;
96          //Tiempo parpadeo LED quemador
97          unsigned long t_flashB = 0;
98          //Tiempo de barrido
99          unsigned long t_burnerStart = 0;
100         //Tiempo del encendido
101         unsigned long t_burnerIgnition = 0;
102         //Volumen de agua en la caldera al inicio del ciclo
103         double v_init = 0.0;
104         //Incremento del volumen en cada ciclo
105         double v_delta = 0.0;
106         //Temperatura de saturación (Â ºC)
107         float T_saturate = 0.0;
108         //Presión correspondiente a la temperatura de saturación
109         float P_saturate = 0.0;
110         //Densidad de líquido (kg/m3)
111         float density_liquid = 0.0;
112         //Densidad de vapor (kg/m3)
113         float density_steam = 0.0;
114         //Entalpía específica del agua líquida (kJ/kg)
115         float h_water = 0.0;
116         //Entalpía específica del vapor (kJ/kg)
117         float h_steam = 0.0;
118         //Entalpía específica del agua en el bucle anterior (kJ/kg)
119         float H_water_b = 0.0;
120         //Calor latente de vaporización (kJ/kg)
121         float L_ws = 0.0;
122         //Calor específico del agua líquida (kJ/kgK)
123         float c_water = 0.0;
124         //Temperatura de la caldera
125         float BoilerTemperature = 0.0;
126         //Temperatura tras el aporte energético del quemador
```

```
127         float IntermediateTemperature = 0.0;
128         //Energía neta
129         double NetEnergy = 0.0;
130         //Energía de calentamiento a Ts
131         float HeatEnergy = 0.0;
132         //Energía sobrante
133         float SurplusEnergy = 0.0;
134         //Masa de vapor generado
135         float SteamMass = 0.0;
136         //Volumen de vapor generado
137         float SteamVolume = 0.0;
138         //Calculo de la presion generada por la masa de vapor
139         float SteamPressurecreate = 0.0;
140         //Numero de moles
141         float n = 0.0;
142
143         //Combustión, variables cálculo temperatura del hogar
144         //Fuel Flow
145         float FuelFlow = 0.0;
146         //Value n of Air Excess
147         float nExcess = 0.0;
148         //Reactive combustion products flow
149         //Moles of fuel for current fuel flow
150         float MOLFlowFuel = 0.0;
151         //Moles of oxygen for stoichiometric combustion
152         float MOLO2stoichiometric = 0.0;
153         //Moles of oxygen flow for stoichiometric combustion
154         double MOLO2stoichiometricFlow = 0.0;
155         //No reactive exhaust gases flow with air excess
156         //Excess oxigen flow
157         double O2Flow = 0.0;
158         //Excess nitrogen flow
159         double N2Flow = 0.0;
160         //Excess carbon dioxide flow
161         double CO2Flow = 0.0;
162         //Reactive exhaust gases flow with air excess
163         //Reactive carbon dioxide flow of exhaust gas
164         double CO2Flowreactive = 0.0;
165         //Reactive water steam flow of exhaust gas
166         double H2OFlowreactive = 0.0;
167         //Non-reactive nitrogen flow of exhaust gas
168         double N2Flownonreactive = 0.0;
169         //Non-reactive carbon dioxide flow of exhaust gas
170         double CO2Flownonreactive = 0.0;
171         //Non-reactive oxygen flow of exhaust gas
172         double O2Flownonreactive = 0.0;
173         //Boiler furnace temperature
174         float FurnaceTemp = 0.0;
175
176         //Efficiency according to boiler load
177         //Curve Efficiency
178         float Efficiency = 0.0;
179         //U coefficient
180         float CoeffU = 0.0;
181         //Real logarithmic mean temperature
182         float LMTreal = 0.0;
183         //Iterated logarithmic mean temperature
184         float LMTiterated = 0.0;
185         //Exhaust Temp
186         float ExhaustTemp = 0.0;
187
188         //variables cálculo de superficie
189         //Radio interno del hogar (m)
```

```
190         double Internalfurnrad = 0.00;
191         //Radio externo del hogar (m)
192         double Externalfurnrad = 0.00;
193         //Radio interno de los tubos del segundo paso de humos (m)
194         double IntsecSteprad = 0.00;
195         //Radio Externo de los tubos del segundo paso de humos (m)
196         double ExtsecSteprad = 0.00;
197         //Radio interno de los tubos del tercer paso de humos (m)
198         double IntthiSteprad = 0.00;
199         //Radio externo de los tubos del tercer paso de humos (m)
200         double ExtthiSteprad = 0.00;
201         //Superficie del hogar (m2)
202         double Furnacesurface = 0.00;
203         //Superficie del segundo paso de humos (m2)
204         double secStepsurface = 0.00;
205         //Superficie del tercer paso de humos (m2)
206         double thiStepsurface = 0.00;
207         //Superficie total de intercambio de la caldera (m2)
208         double ExchangeSurface = 0.00;
209
210
211         //Calor especifico del agua de alimentación a 40Â ºC (kJ/kgK)
212         float C_FeedWater = 4.179;
213         //Masa de agua de la caldera (kg)
214         float WaterMass = 0.0;
215         //Masa de agua de alimentación en cada ciclo (kg)
216         float FeedWaterMass = 0.0;
217         //Viscosidad dinámica del vapor saturado kg/ms
218         float viscosity_steam = 0.0;
219         //Número de Reynolds
220         float Re = 0.0;
221
222
223
224         //Conversión de variables analógicas
225         float Burner = 0.0;
226         //Constante tiempo de reacción del quemador
227         unsigned long timeBurner = 0;
228         //Posición de la válvula de agua de alimentación
229         float ButterflyValve = 0.0;
230         //Tiempo de reacción de la válvula de agua de alimentación
231         unsigned long timeButterflyValve = 0;
232         //Posición de la válvula principal de vapor
233         float Valve = 0.0;
```