SEDE GUAYAQUIL

FACULTAD DE INGENIERÍAS

CARRERA: INGENIERÍA ELECTRÓNICA
INGENIERÍA EN SISTEMAS

PROYECTO TÉCNICO PREVIO A LA OBTENCIÓN DEL TÍTULO DE
INGENIERO ELECTRÓNICO
INGENIERO EN SISTEMAS

TEMA:

DISEÑO E IMPLEMENTACIÓN DE UN MÉTODO DE DESTILACIÓN DE HEXANO BASADO EN UN SISTEMA DE CALENTAMIENTO POR INDUCCIÓN CON CONTROL PID Y MONITOREO EN LÍNEA A TRAVÉS DE UNA PÁGINA WEB

AUTORES:

AGUIRRE MARIDUEÑA LUIS BRIAN
SALCEDO PONTON CRISTIAN BOLIVAR
NEIRA MELENDREZ JOSELINE ROXANA

TUTOR:

ING. NEIRA CLEMENTE LUIS
ING. TANDAZO ESPINOZA MÁXIMO

GUAYAQUIL – ECUADOR
2018

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DEL TRABAJO DE GRADO

Nosotros Joseline Roxana Neira Melendrez, Cristian Bolivar Salcedo Ponton y Luis Brian Aguirre Maridueña autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de grado y su reproducción sin fines de lucro.

Además, declaramos que los conceptos y análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad de los autores.

Joseline Roxana Neira Melendrez
C.C:

Cristian Bolivar Salcedo Ponton
C.C:

Luis Brian Aguirre Maridueña
C.C:

CESIÓN DE DERECHOS DE AUTOR

Nosotros Joseline Roxana Neira Melendrez, con documento de identificación N° , Cristian Bolivar Salcedo Ponton con documento de identificación N° , y Luis Brian Aguirre Maridueña con documento de identificación N° , manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de grado intitulado: "**DISEÑO E IMPLEMENTACIÓN DE UN MÉTODO DE DESTILACIÓN DE HEXANO BASADO EN UN SISTEMA DE CALENTAMIENTO POR INDUCCIÓN CON CONTROL PID Y MONITOREO EN LÍNEA A TRAVÉS DE UNA PÁGINA WEB**", mismo que ha sido desarrollado para optar por el título de: Ingeniero Electrónico e Ingeniero en Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada en concordancia, suscribo este documento en el momento que hacemos entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

.....	.....
Joseline Roxana Neira Melendrez	Cristian Bolivar Salcedo Ponton
C.C:	C.C:

.....

Luis Brian Aguirre Maridueña

C.C:

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Ingeniero Luis Neira, tengo a bien certificar que los estudiantes Joseline Roxana Neira Melendrez, Cristian Bolívar Salcedo Ponton y Luis Brian Aguirre Maridueña, han realizado el presente Proyecto de Titulación bajo mi supervisión; y, por lo tanto, se encuentran aptos para su correspondiente presentación.

Guayaquil,

Ing. Luis Neira Clemente

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Ingeniero Máximo Tandazo Espinoza, tengo a bien certificar que los estudiantes Joseline Roxana Neira Melendrez, Cristian Bolívar Salcedo Ponton y Luis Brian Aguirre Maridueña, han realizado el presente Proyecto de Titulación bajo mi supervisión; y, por lo tanto, se encuentran aptos para su correspondiente presentación.

Guayaquil,

Ing. Máximo Tandazo Espinoza

DEDICATORIA

Dedico este trabajo mi Dios por darme la oportunidad de vivir todos los días y darme fuerzas para culminar este proceso, a mi padre que con su apoyo incondicional ha sabido guiarme durante este camino, a mi madre ya que gracias a su paciencia y sabio consejo he podido completar esta etapa.

Joseline Roxana Neira Melendrez

DEDICATORIA

Dedico este trabajo principalmente a Dios, por permitirme día a día vivir, finalizar y llegar a este momento tan importante de mi vida como profesional. A mí querida Madre por darme su apoyo y ser el pilar fundamental de esta etapa de mi vida. A mi Padre por incentivarme y motivarme a culminar mi carrera universitaria. A mi querida hermana Martina por brindarme su inmenso cariño. A mis abuelitas, que con sus bendiciones y afecto siempre desearon lo mejor para su nieto. A mis Tíos de parte de padre y madre, por darme su ayuda y apoyo incondicional en todos los momentos de este trayecto tan lindo de estudiante. A mis compañeros de curso, a mis amigos más cercanos y primos con los que pude compartir mi etapa de vida universitaria.

Cristian Bolívar Salcedo Pontón

DEDICATORIA

Dedico este trabajo a mi Dios pues gracias a Él estoy pudiendo completar esta etapa en mi carrera profesional, a mi padre quien ha sido mi amigo con su sabio consejo y ayuda incondicional, a mi madre que con su amor, dedicación y paciencia ha sabido dirigirme hasta el final de esta etapa, a mis hermanos quienes me han ayudado en momentos difíciles, a todos mis amigos que han sido partícipes de este duro proceso y con su presencia y buen humor han hecho divertido todo este árduo proceso universitario y a mis profesores y compañeros de clases por su ayuda y colaboración durante toda la carrera.

Luis Brian Aguirre Maridueña

AGRADECIMIENTO

Agradezco a Dios por cuidarme, fortalecerme y darme siempre los ánimos para superar los obstáculos y dificultades a lo largo de mi vida.

A mi Madre, por darme su ejemplo, sus enseñanzas, sus valiosos consejos para no desfallecer ni rendirme ante nada y ser esa persona tan importante que es en mi vida.

A mi Padre, por permitirme ser la persona que soy hoy en día, darme su apoyo y motivación para seguir adelante con el duro camino de universitario.

A mi hermana Martina, por ser la niña más linda, cariñosa, afectiva, protectora que con su gran espíritu me ha brindado día a día.

A mis abuelitas Melida y Emma, por cuidar de mi desde niño, darme sus bendiciones y su infinito apoyo moral.

A mi Tía Lorena y mi Tío Diego, por estar conmigo, visitarme, cuidarme y brindarme su infinita ayuda.

A mis amigos, compañeros, primos por acompañarme y lograr este objetivo de vida tan apreciado.

Al Ing. Luis Neira, director de tesis, por su valiosa guía y asesoramiento a la realización de la misma.

Gracias a todas las personas que me ayudaron directa e indirectamente con la elaboración y realización de este proyecto.

Cristian Bolívar Salcedo Pontón
Joseline Roxana Neira Melendrez
Luis Brian Aguirre Maridueña

RESUMEN

AÑO	ALUMNOS	TUTOR DEL PROYECTO	TEMA DEL PROYECTO
2018	Joseline Roxana Neira Melendrez Cristian Bolívar Salcedo Ponton Luis Brian Aguirre Maridueña	Ing. Tandazo Espinoza Máximo Ing. Neira Clemente Luis	"DISEÑO E IMPLEMENTACIÓN DE UN MÉTODO DE DESTILACIÓN DE HEXANO BASADO EN UN SISTEMA DE CALENTAMIENTO POR INDUCCIÓN CON CONTROL PID Y MONITOREO EN LÍNEA A TRAVÉS DE UNA PÁGINA WEB"

El presente proyecto técnico de titulación: "DISEÑO E IMPLEMENTACIÓN DE UN MÉTODO DE DESTILACIÓN DE HEXANO BASADO EN UN SISTEMA DE CALENTAMIENTO POR INDUCCIÓN CON CONTROL PID Y MONITOREO EN LÍNEA A TRAVÉS DE UNA PÁGINA WEB", es aplicado en la Empresa ELICROM de la Ciudad de Guayaquil, durante el periodo comprendido entre los meses Enero y Julio del año 2018 respectivamente. La solución a la problemática fue presentada para ofrecer una alternativa tecnificada y actual a los métodos de destilación, normalmente la empresa ELICROM utiliza métodos manuales que están quedando obsoletos frente a la utilización de dispositivos electrónicos.

La metodología aplicada en este trabajo permitió establecer la solución más idónea, inciamos con la observación directa realizada en los laboratorios de destilación de ELICROM y mediante el análisis inductivo y deductivo se llega a establecer la funcionalidad del proyecto, así como las conclusiones y recomendaciones respectivas.

Mediante este proyecto, se implementó un nuevo método de destilación del hexano gracias al empleo de un sistema de calentamiento por inducción con control PID[1] el cual se encargó de procesar los datos transmitiéndolos al módulo Arduino y a su vez entrelazar esa información a la Raspberry Pi 3 lo que permitió el monitoreo de los datos en tiempo real vía servidor web mediante un protocolo de comunicación MQTT y enviados a una base de datos con control de alarmas de sobrecalentamiento, con la finalidad de tener acceso de manera eficiente en corto tiempo.

Palabras claves: tecnología, PID, Arduino, RaspBerry PI, Hexano, MQTT, servidor Web.

[1]PID Ganancia proporcional (P), Integral (I) y Derivativo (D).

SUMMARY

YEAR	STUDENTS	PROJECT TUTOR	PROJECT TITLE
2017	Joseline Roxana Neira Melendrez Cristian Bolívar Salcedo Ponton Luis Brian Aguirre Maridueña	Ing. Tandazo Espinoza Máximo Ing. Neira Clemente Luis	"DESIGN AND IMPLEMENTATION OF A HEXANE DISTILLATION METHOD BASED ON A INDUCTION HEATING SYSTEM WITH PID CONTROL AND ON-LINE MONITORING THROUGH A WEB PAGE"

The present technical project of titling: "DESIGN AND IMPLEMENTATION OF A HEXANE DISTILLATION METHOD BASED ON A INDUCTION HEATING SYSTEM WITH PID CONTROL AND ON-LINE MONITORING THROUGH A WEB PAGE", applied in the company ELICROM of the city of Guayaquil, during the period 2016 – 2017 respectively. The solution to the problem was presented due to the methods of distillation normally used in the laboratories of this company are manuals, which is why they are becoming obsolete due to the use of electronic devices.

The methodology applied in this work was of great, help because it allowed to establish the most suitable solution by means of the direct observation carried out in the distillation laboratories of ELICROM, in addition to which the technique of the experimentation was employed, the methods Inductive and deductive that managed to establish the functionalities (test and error) of the developed project.

Through this project, a new method of distillation of hexane was implemented thanks to the use of an induction heating system with PID[2], control which was responsible for processing the data trasmite to the Arduino module and in turn to interlace that information to the Raspberry Pi, which allowed real-time data monitoring via the web server throught the communication protocol MQTT and send to a data base with alarms control of overheating in order to deploy control and access networks in an efficient way in a short time, in addition to avoiding infrastructure redesigns.

Keywords: Technology, PID, Arduino, Raspberry PI, hexane. MQTT, Web Server

[2]PID Proportional Gain (P), Integral (I) and derivative (D).

INTRODUCCIÓN

En la actualidad, el desarrollo de la tecnología ha permitido la automatización de procesos, tareas o actividades que permiten utilizar las capacidades de máquinas, equipos y sistemas para realizar procesos determinados sin la intervención total o parcial de la mano de obra del hombre, lo que la hace perfecta para aplicarla en diversos sectores de producción, bienes o servicios como energía eléctrica, industria de hidrocarburos, industria química, telecomunicaciones, alimentos, entre otras, repercutiendo efectivamente en el aumento de la producción, reducción de costos, mejoramiento en la calidad del producto final entregado, aumento de la seguridad y la integración de sistemas que mejoraran la productividad de la empresa.

Una de las áreas en las que más se emplean los sistemas automatizados es en la industria química para los procesos de medición, elaboración de productos en los laboratorios y procesos de destilación de diferentes componentes (agua, hexano, alcohol crudo y etílico, etano-etileno y propano- propileno, entre otros) con la finalidad de realizar estas actividades de manera precisa y segura tanto para los empleados de la empresa como para el cliente que va a recibir el producto final.

En el presente proyecto de investigación se diseñó e implementó para la empresa ELICROM S.A de la ciudad de Guayaquil, un sistema de destilación de hexano basado en un sistema de calentamiento por inducción con control PID y monitoreo en línea mediante el empleo del módulo Arduino y la tarjeta Raspberry PI 3 que fueron enlazados para el funcionamiento del sistema de destilación y el sistema de calentamiento con el empleo de sensores, mismo que será operado a través de una página web que será diseñada bajo la plataforma del lenguaje de programación JAVA mediante el protocolo MQTT, mismo que permite que el administrador de la empresa ingrese los datos de los trabajadores que están encargados de los laboratorios de destilación. También puedan iniciar y detener los procesos de destilación y programar la temperatura correcta, con el objetivo de que ellos manipulen el sistema automatizado y se lleven a cabo las actividades de manera rápida, segura y eficiente.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. EL PROBLEMA

La empresa ELICROM se dedica al estudio de sistemas ambientales, procesos químicos y para esto en sus instalaciones utilizan equipos de laboratorio como los destiladores de hexano. Estos destiladores normalmente son manuales y tiene un controlador de temperatura básico, no cuenta con un sistema que lleve el registro, estabilización y monitoreo de la temperatura.

Adicionalmente a esto, el destilador común se calentaba con una resistencia tubular, la cual tiene un consumo considerable de energía y además representan un riesgo para los laboratoristas por el alto calentamiento que poseen.

1.1 Antecedentes

La empresa ELICROM está ubicada en Guayaquil, Ciudadela Guayaquil, Manzana 21 calle primera solar 10. Esta fue creada en el año 2001 con la finalidad de prestar servicios especializados a la Industria Ecuatoriana.

En la prestación de servicios, ELICROM cuenta con varios departamentos especializados en actividades específicas como son las áreas de Metrología, Medio Ambiente, Validaciones, Mantenimiento de Equipos de Laboratorio y Construcción/Reformas de Laboratorios.

En el año 2007, la compañía acertadamente decidió crear la división de Comercialización de Equipos de Laboratorio, y desde entonces hasta la fecha han sido seleccionados como distribuidores para el Ecuador por reconocidas marcas de trayectoria mundial, tal es el caso de Hach, Sartorius, VWR, etc., siendo actualmente esta actividad una de las principales líneas de negocio de la compañía.

Hoy en día, ELICROM cuenta con laboratorios de calibración y ensayo acreditados por la ISO 17025 del Organismo Ecuatoriano de Acreditación OAE, convirtiéndose en el laboratorio de calibración de mayor alcance (mayor número de parámetros) que existe en el Ecuador actualmente.

1.2 Importancia

La aplicación para la gestión del registro de datos obtenidos del destilador de hexano brindará a la empresa ELICROM S.A un sistema capaz de registrar su proceso de destilación, por medio de un ambiente web a su personal la cantidad de veces que desean.

El sistema de inducción implementado no se encuentra limitado solo al uso de hexano en su proceso de destilación, el usuario podrá ingresar el líquido que se desee siempre y cuando sea compatible con el material del sistema, esto se debe a que el sistema

desarrollado permite ajustar la temperatura de ebullición del líquido a destilar conocida como Setpoint o temperatura de consigna.

El método de destilación diseñado reducirá los costos de consumo energético eléctrico los cuales son muy elevados por un destilador convencional debido al uso de resistencias de calentamiento y el largo periodo de tiempo que les lleva realizar el proceso de destilación.

Mediante el sistema diseñado se podrá controlar y monitorear la temperatura de operación del sistema de destilación, obtener la cantidad de tiempo que tarda el proceso, así como la visualización de gráficas de tendencia de variables del sistema como temperatura, el voltaje y potencia.

Este proceso actualmente se realiza de forma manual en un documento de Excel, el cual lleva el registro de los datos anteriormente mencionados. Al llevar este proceso de forma manual puede resultar inseguro porque podría perderse la información y todo registro de las destilaciones realizadas y por consiguiente tendrían que volver al punto inicial. Se debe tener en cuenta que al manejar este tipo de información, no es conveniente la posibilidad de perderla, es por ello que con ayuda de la aplicación web, estos datos se mantendrán almacenados en la base de datos disminuyendo el riesgo de pérdida de los mismos.

1.3 Grupo objetivo (beneficiarios)

El primer beneficiario de este proyecto será la empresa ELICROM, porque contará con un equipo de destilación confiable, que le otorgará un historial de todas las paradas que se realizan durante el día. Adicionalmente tendrán un equipo de bajo consumo energético.

Como segundo beneficiario serán los laboratoristas porque podrán contar con un equipo de fácil operación y que no representará un riesgo por quemaduras de primer grado, debido a que se utilizará un sistema de calentamiento por inducción y no por resistencia calorífica.

1.4 Delimitación

- **Geográfica:** Guayaquil, Ciudadela Guayaquil, Manzana 21 calle primera solar 10.
- **Temporal:** El proyecto fue realizado durante el periodo académico 2017 – 2018.
- **Sectorial:** Está orientada a las áreas de Metrología, Medio Ambiente, Validaciones y procesos de destilación.
- **Institucional:** la empresa beneficiada será ELICROM S.A

1.5 Objetivos

1.5.1 Objetivo General

Diseñar e implementar un método de destilación de hexano basado en un sistema de calentamiento por inducción con control PID y monitoreo web para la empresa ELICROM.

1.5.2 Objetivos Específicos

- Investigar los sistemas de destilación de hexano con control PID existentes y su eficiencia versus otros tipos de controladores.
- Diseñar el sistema de calentamiento por inducción.
- Construir la estructura para destilación de hexano.
- Diseñar el sistema de calentamiento por inducción utilizando control PID con el software MATLAB-Simulink y con el controlador RaspBerry PI3 para la estabilización de la temperatura.
- Diseñar la aplicación web para monitorear el proceso y la temperatura del destilador.

2 MARCO TEÓRICO REFERENCIAL

2.1 Destilación

La destilación es un método comúnmente utilizado para la purificación de líquidos y la separación de mezclas con el fin de obtener sus componentes individuales.

“La destilación es una técnica de separación de sustancias que permite separar los distintos componentes de una mezcla. Esta técnica que se basa fundamentalmente en los puntos de ebullición de cada uno de los componentes de la mezcla, cuanto mayor sea la diferencia entre los puntos de ebullición de las sustancias de la mezcla, más eficaz será la separación de sus componentes; es decir, los componentes se obtendrán con un mayor grado de pureza”. (Hein & Susan, 2001)

“La técnica consiste en calentar la mezcla hasta que ésta entra en ebullición. A medida que la mezcla se calienta, la temperatura aumenta hasta que alcanza la temperatura de la sustancia con punto de ebullición más bajo mientras que los otros componentes de la mezcla permanecen en su estado original. A continuación, los vapores se dirigen hacia un condensador que los enfría y los pasa a estado líquido. El líquido destilado tendrá la misma composición que los vapores y; por lo tanto, con esta sencilla operación habremos conseguido enriquecer el líquido destilado en el componente más volátil (el de menor punto de ebullición). Por consiguiente, la mezcla sin destilar se habrá enriquecido con el componente menos volátil (el de mayor punto de ebullición)”. (Hein & Susan, 2001)

2.2 Tipos de destilación

2.2.1 Destilación simple

La destilación simple es usada para separar líquidos en el que su punto de ebullición sea inferior a 150°C que contengan impurezas no volátiles, o de otros líquidos que tengan un punto de ebullición mayor a 25°C de su punto nominal, para que la destilación sea homogénea y no existan algun tipo de proyección se coloca en el matraz un recipiente porozo. (CHANG, 2002)

2.2.2 Destilación fraccionada

Este tipo de destilación para realizar la separación de líquidos en los cuales su punto de ebullición es menor a 25°C. Cada componente que es separado se le denomina fracción. La destilación es parecida la destilación simple con la diferencia que entre el matraz y la parte superior de la destilación se instala una columna que puede tener distinto diseño comúnmente llamada columna vigreux, al calentar la mezcla origina un vapor que se va enriqueciendo y adhiriendo al componente más volátil y a su vez va ascendiendo por la columna de vigreux. (Burns, 2011)

2.2.3 Destilación a vacío

La técnica utilizada en este tipo de destilación es similar a los otros procesos con la diferencia de que todo el conjunto es conectado a una bomba de vacío. En lugar de un recipiente porozo se puede instalar adaptando a los equipos un capilar de vidrio o

algun dispositivo semejante de manera que se obtenga una ebullición homgénea, este tipo de destilación es usado para líquidos a temperaturas bajas mucho más que el caso anterior ya que la presión es menor que la atmosférica con esto se previene la descomposición térmica de los líquidos que se destilan. (Burns, 2011)

2.3 Calentamiento por Inducción

"El calentamiento por inducción consiste en una técnica utilizada muchas veces para unir, endurecer, ablandar materiales metales o materiales conductivos. En algunos procesos de fabricación esta técnica permite una combinación de velocidad de procesos, consistencia, control y eficiencia en materia enegética". (GH Electrotermia , 2011)

2.3.1 Comportamiento del calentamiento por inducción

"Cuando se aplica una corriente eléctrica alterna al primario de un transformador, se crea un campo magnético alterno. De acuerdo con la Ley de Faraday, si el secundario del transformador está ubicado dentro del campo magnético, se inducirá una corriente eléctrica". (GH Electrotermia , 2011)

"En una configuración básica de calentamiento por inducción, una fuente de alimentación genera una corriente alterna que atraviesa un inductor (normalmente una bobina de cobre) y la pieza a calentar se sitúa dentro de dicho inductor. El inductor actúa de primario del transformador y la pieza de circuito secundario. Cuando la pieza metálica es atravesada por el campo magnético, se inducen corrientes de Foucault en dicha pieza". (GH Electrotermia , 2011)

"Las corrientes de Foucault fluyen contra la resistividad eléctrica del metal, generando un calor localizado y preciso sin ningún contacto directo entre la pieza y el inductor. Este calentamiento ocurre con piezas magnéticas y no-magnéticas, y a menudo se conoce como "Efecto Joule" que hace referencia a la primera ley de Joule (fórmula científica que expresa la relación entro calor producido y corriente eléctrica a través de un conductor)". (GH Electrotermia , 2011)

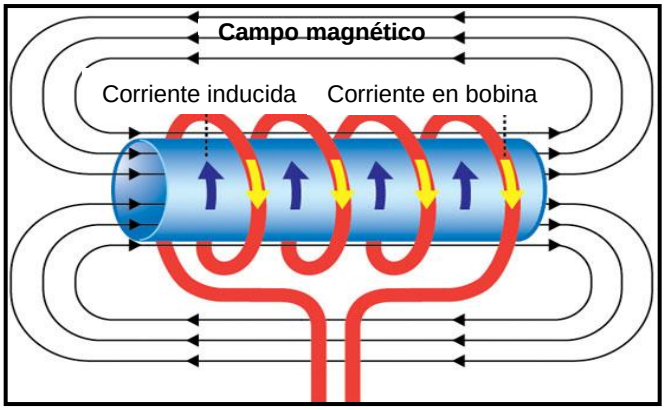


Figura 1. Inducción electromagnética

(Wilsin, j., & Bo, 2007)

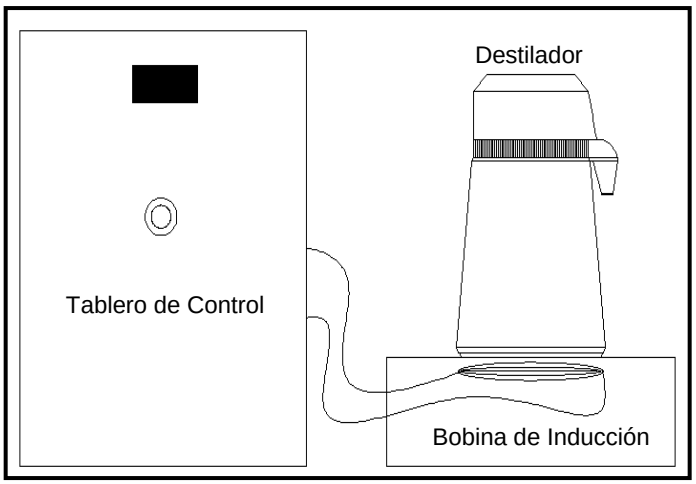


Figura 2. Esquema de destilación por inducción

(L. Aguirre, C. Salcedo, J. Neira, 2018)

2.4 Beneficios del calentamiento por inducción

2.4.1 Productividad máxima

“Las tasas de productividad pueden maximizarse porque la inducción es muy rápida: el calor se genera directa e instantáneamente en la pieza (ejemplo, más de 1000ºC en menos de un segundo en algún caso). El arranque para el calentamiento es virtualmente instantáneo, no es necesario precalentar ni enfriar. El proceso de calentamiento por inducción se completa en la planta de fabricación, próximo a la máquina de deformación fría o caliente, en lugar de enviar lotes de piezas a un horno remoto o subcontrata”. (GH Electrotermia , 2011)

2.4.2 Eficiencia energética

“Este proceso es el único realmente eficiente desde el punto de vista energético. Convierte la energía consumida en calor útil en hasta un 90%; los hornos por lotes lo hacen generalmente hasta un 45%. Además, como no necesita ni pre calentamiento ni enfriamiento en los ciclos de trabajo, las pérdidas de calor en stand-by (cuando no trabaja) se reducen a mínimos”. (GH Electrotermia , 2011)

2.5 Control y automatización del proceso

“El calentamiento por inducción elimina las inconsistencias y los problemas de calidad que se producen con llama, soplete u otros métodos. Una vez el sistema está calibrado y en marcha, no hay lugar a las desviaciones, los patrones de calentamiento son repetibles y consistentes”. (GH Electrotermia , 2011)

“Con un lazo cerrado de control de temperatura, los sistemas avanzados de calentamiento por inducción tienen la capacidad de medir la temperatura de cada pieza individualmente. La velocidad de aumento, mantenimiento y descenso de temperatura

pueden establecerse en cada caso y los datos se almacenan para cada pieza sobre la que trabaja". (GH Electrotermia , 2011)

2.5.1 Control PID

"El control PID es un mecanismo de control que a través de un lazo de retroalimentación permite regular la velocidad, temperatura, presión y flujo entre otras variables de un proceso en general. El controlador PID calcula la diferencia entre nuestra variable real contra la variable deseada. En sistemas de bombeo, regularmente nos interesa mantener la presión o flujo constante, por lo tanto, el control PID mide la diferencia entre la presión en la tubería y la presión requerida y actúa variando la velocidad del motor para que podamos tener nuestra presión o flujo constante, el algoritmo de control incluye tres parámetros fundamentales: Ganancia proporcional (P), Integral (I) y Derivativo (D)". (García, 2013)

"El parámetro Proporcional (P) mide la diferencia entre el valor actual y el set-point (en porcentaje) y aplica el cambio. (García, 2013)

"El parámetro Integral (I) se refiere al tiempo que se toma para llevar a cabo acción correctiva. Mientras el valor sea más pequeño, el ajuste es más rápido, pero puede causar inestabilidad en el sistema, oscilaciones." (García, 2013)

"El parámetro Derivativo (D) emite una acción predictiva, es decir, anticipa el error e inicia una acción oportuna. Responde a la velocidad del cambio del error y produce una corrección significativa antes de que la magnitud del error se vuelva demasiado grande". (García, 2013)

"La correcta sintonización o programación de estos parámetros nos ayuda a controlar de manera efectiva nuestra presión, temperatura o flujo deseado. Si no programamos adecuadamente estos parámetros, el sistema puede quedar inestable". (García, 2013)

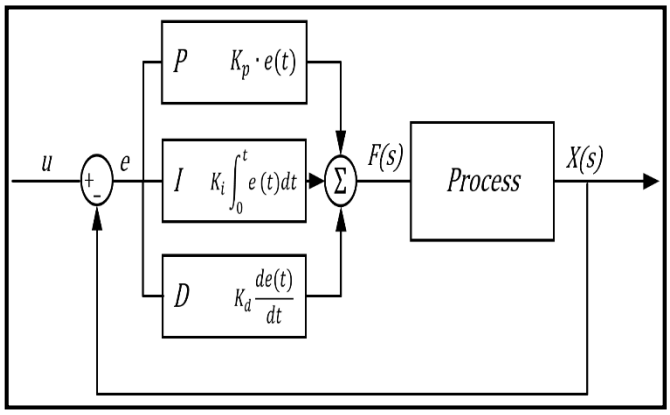


Figura 3. Diagrama de bloque de control PID

(Astrom, 2000)

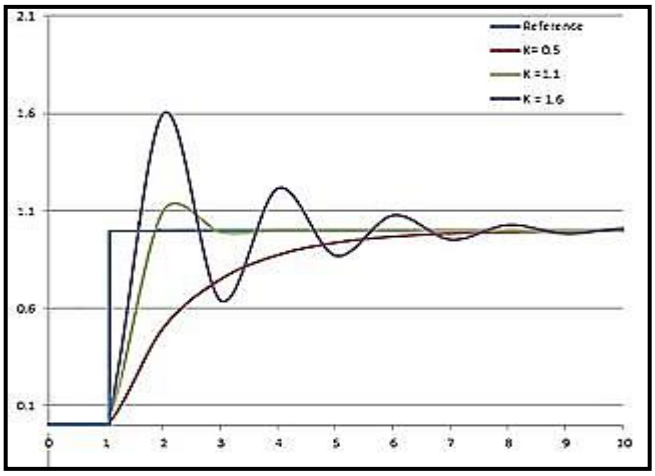


Figura 4. Forma de ondas de control PID

(Astrom, 2000)

2.6 Arduino

"Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares. El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (bootloader) que corre en la placa". (arduino.cl, 2017)

Figura 5. Logo Arduino

(Margolis, 2010)

2.6.1 Arduino UNO

Es una placa electrónica basada en el microcontrola ATmega328 Cuenta con 14 entradas y salidas digitales, de las cuales 6 se pueden utilizar como salidas PWM (modulación por ancho de pulsos) y otras 6 son entradas analógicas. (Perez, 2015)

Además, incluye un resonador cerámico de 16Mhz, un conector USB, un conector de alimentación, una cabecera ICSP y un botón de reseteado. La placa incluye todo lo necesario para que el microcontrolador haga su trabajo, basta conectarla a un ordenador con un cable USB o a la corriente eléctrica a través de un transformador. (Perez, 2015)

2.6.2 Características Generales del Arduino Uno

Microcontrolador:	ATmega32u4
Tensión de funcionamiento:	5V
Alimentación recomendada:	7-12V
Pines I/O Digitales:	20
Canales PWM:	7
Entradas analógicas:	12
Corriente Maxima de los pines I/O:	40 mA
Corriente Maxima de los pines 3.3V	50 mA
Memoria Flash:	32 KB (4 KB usados para el bootloader)
SRAM:	2.5 KB
EEPROM interna:	1 KB
Velocidad:	16 MHz

Tabla 1. Características de la tarjeta controladora Arduino Uno

(ELECTRONICA EMBAJADORES, S.L, 2017)

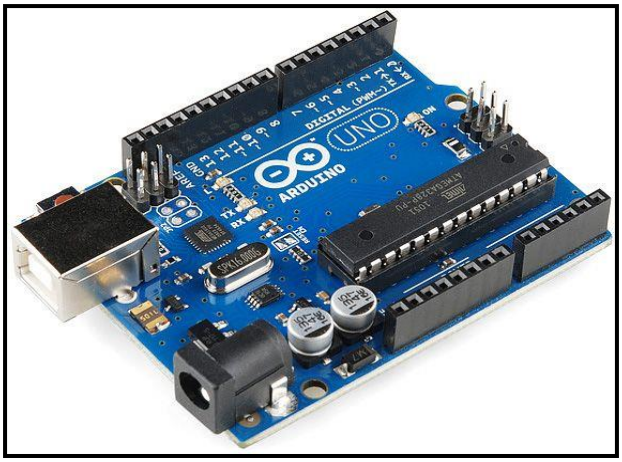


Figura 6. Tarjeta Arduino Uno

(ELECTRONICA EMBAJADORES, S.L, 2017)

2.7 Raspberry

Es un pequeño ordenador que se conecta a su monitor y un teclado, es una placa que soporta varios componentes necesarios en un ordenador común, capaz de ser utilizado por muchas de las cosas que su PC de escritorio hace, como hojas de cálculo, procesadores de texto y juegos (ABC Tecnología, 2013)

Tiene varios puertos y entradas, dos USD, uno de Ethernet y salida HDMI. Estos puertos permiten conectar el miniordenador a otros dispositivos, teclados, ratones y pantallas. También posee un System on Chip que contiene un procesador ARM que corre a 700 Mhz, un procesador gráfico VideoCore IV y hasta 512 MG de memoria RAM. Es posible instalar sistema operativo libre a través de una tarjeta SD (ABC Tecnología, 2013)

2.7.1 Raspberry PI3

La Raspberry Pi 3 es una placa base con unas dimensiones tan llamativas como sus especificaciones y la larga lista de aplicaciones que este pequeño artilugio puede tener. (ABC Tecnología, 2013)

2.7.2 Características generales de la Raspberry Pi 3

Procesador	Chipset Broadcom BCM2387.
GPU	Dual Core VideoCore IV ® Multimedia Co-procesador. Proporciona Open GL ES 2.0, OpenVG

Memoria RAM	1 Gb LPDDR2
Conectividad	Ethernet socket Ethernet 10/100 BaseT
Salida de video	HDMI rev 1.3 y 1.4
Salida de audio	Jack de 3,5 mm de salida de audio, HDMI
Conector GPIO	40-clavijas de 2,54 mm (100 milésimas de pulgada) de expansión: 2x20 tira

Tabla 2. Características de la tarjeta RaspBerry PI3

(Kurniawan, 2015)

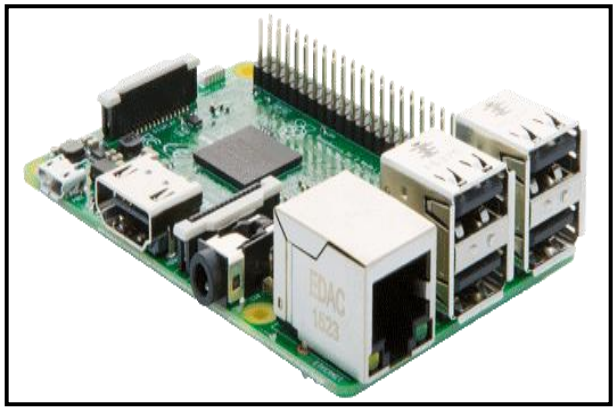


Figura 7. Tarjeta controladora RaspBerry PI 3

(Kurniawan, 2015)

2.8 Protocolo MQTT

MQTT (Message Queue Telemetry Transport), un protocolo usado para la comunicación machine-to-machine (M2M) en el "Internet of Things". Este protocolo está orientado a la comunicación de sensores, debido a que consume muy poco ancho de banda y puede ser utilizado en la mayoría de los dispositivos empotrados con pocos recursos (CPU, RAM). (Yébenez Gálvez , 2017)

Un ejemplo de uso de este protocolo es la aplicación de Facebook Messenger tanto para android e Iphone. La arquitectura de MQTT sigue una topología de estrella, con un nodo central que hace de servidor o "broker" con una capacidad de hasta 10000

clientes. El broker es el encargado de gestionar la red y de transmitir los mensajes, para mantener activo el canal, los clientes mandan periódicamente un paquete (PINGREQ) y esperan la respuesta del broker (PINGRESP). La comunicación puede ser cifrada entre otras muchas opciones. (Yébenez Gálvez , 2017)

2.8.1 Características de Protocolo MQTT

MQTT es un protocolo abierto, sencillo, ligero y fácil de implantar, es ideal para responder a las siguientes necesidades: (Digital Dimension, 2015)

- Está especialmente adaptado para utilizar un ancho de banda mínimo
- Es ideal para utilizar redes inalámbricas
- Consume muy poca energía
- Es muy rápido y posibilita un tiempo de respuesta superior al resto de protocolos web actuales
- Permite una gran fiabilidad si es necesario
- Requiere pocos recursos procesadores y memorias (Digital Dimension, 2015)

2.9 Destilador convencional

En la actualidad existen equipos de destilación utilizados a nivel industrial y a nivel doméstico uno de estos equipos que se utiliza para ambos casos es el destilador de Rosson como se aprecia en la siguiente figura:

Figura 8. Destilador Convencional

(L. Aguirre, C. Salcedo, J. Neira, 2018)

2.9.1 Características del destilador convencional

Tensión de Alimentación	110 – 220 Vac
Consumo	750 Watts 7.2 Amps.
Velocidad de destilación	1 Litro por hora
Frecuencia	50/60 Hz

Material Interno	Acero inoxidable

Tabla 3. Características de un destilador convencional

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Ventajas:

- Fácil Uso
- Poco consumo de energía

Desventajas

- Poca capacidad de destilación
- Tiempo extenso de destilación
-

Funcionamiento:

El destilador de Rosson funciona de la siguiente manera, cuenta con una resistencia en la parte inferior, la cual una vez suministrada la energía eléctrica a través de sus terminales de alimentación empieza a generar un calentamiento en el metal, el cual es material de acero inoxidable provocando así que el líquido presente en dentro del recipiente suba su temperatura hasta que esta alcance su punto de ebullición.

Una vez que el líquido alcanza su punto de ebullición procede a convertirse en vapor provocando la destilación o separación de sus componentes químicos; en la parte superior del equipo se encuentra un evaporador el cual se encarga de convertir el vapor en líquido nuevamente, con la diferencia de que este se encuentra con propiedades químicas diferentes a las que obtenía en un principio.

3 MARCO METODOLOGÍCO

Para la realizar el diseño de control del sistema de inducción, se necesita identificar cuantas señales (sensores y actuadores) se van a controlar, de esta forma se podrá realizar la programación a utilizar en el sistema

3.1 Análisis del sistema

Lo primero que se realizó fue la identificación de las etapas del sistema, las cuales son las siguientes:

- Preparación de la materia prima
- Consigna de temperatura
- Ejecución del sistema
- Monitoreo y registro de datos
- Destilación de la materia prima
- Destilación completa

MATERIA PRIMA	• HEXANO
PROCESO 1	• PLACA DE INDUCCION • SETPOINT • CALENTAMIENTO POR INDUCCIÓN
PROCESO 2	• CONTROL PID
POCESO 3	• MONITOREO WEB • REGISTRO DE DATOS • USUARIOS
PROCESO 4	• DESTILACION DEL HEXANO • TEMPERATURA ESTABLE
PRODUCTO TERMINADO	• PROCESO DE DESTILACION TERMINADO

Tabla 4. Proceso de destilación

(L. Aguirre, C. Salcedo, J. Neira, 2018)

3.2 Diseño del sistema de calentamiento por inducción

A continuación se muestra el diagrama de bloques del sistema diseñado para realizar la destilación del producto mediante el calentamiento por inducción

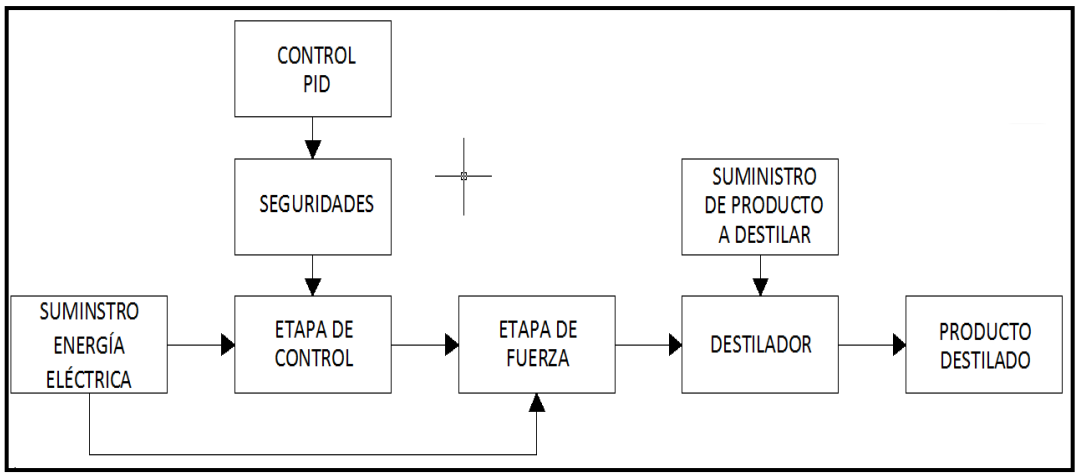


Figura 9. Diagrama de bloques del sistema de destilación.

(L. Aguirre, C. Salcedo, J. Neira, 2018)

3.3 Circuito de Fuerza

Una vez que se identificaron las etapas se procedió a realizar el circuito de fuerza del sistema de inducción el cual contiene los siguientes elementos:

- Microcontrolador MC58HC705P9
- Triacs
- Transformador 220V/110V
- Resistencias 100K
- Disipador de Calor FT5695
- Ventilador 220Vac

Luego de realizar el listado de los elementos de la etapa de fuerza del sistema se procedió a realizar el circuito para su funcionamiento. Ver Anexo 18.

El diseño de la placa contenedora del circuito de fuerza se elaboró en el software Altium designer el cual nos permite elegir y dimensionar el tamaño de la tarjeta con la correcta ubicación de sus componentes.

Acontinuación observamos las dos vistas del circuito de fuerza del sistema.

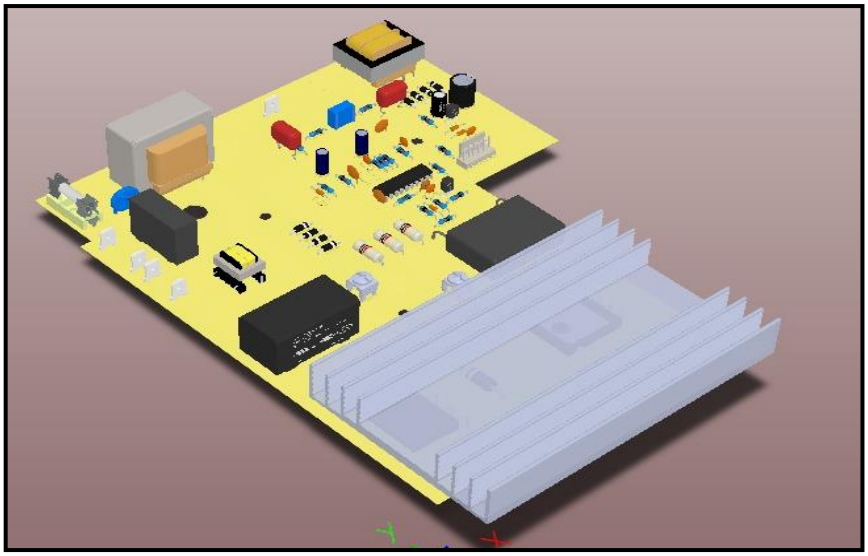


Figura 10. Vista superior en 3D de la placa de fuerza del sistema.

(L. Aguirre, C. Salcedo, J. Neira, 2018)

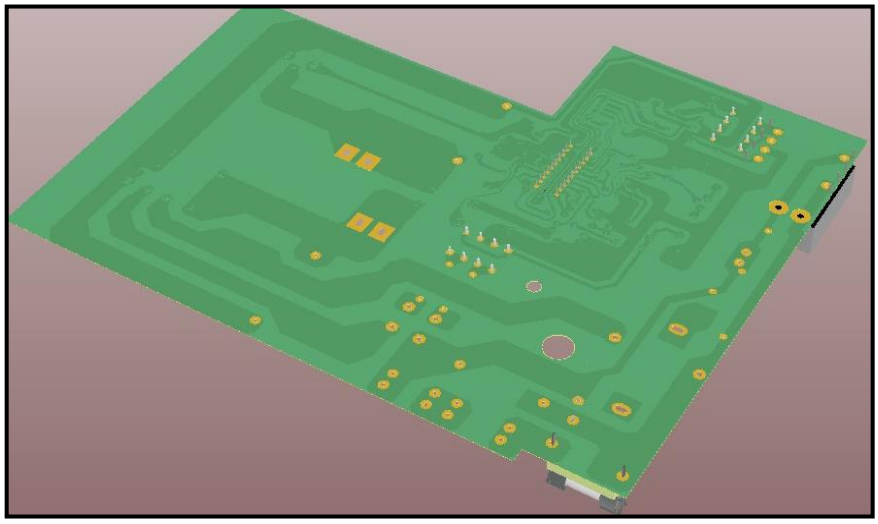


Figura 11. Vista inferior en 3D de la placa de fuerza el sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Luego de realizar el diseño en el software Altiun designer se procedió a realizar la implementación del circuito diseñado con el montaje de los elementos en la placa de fuerza del sistema.

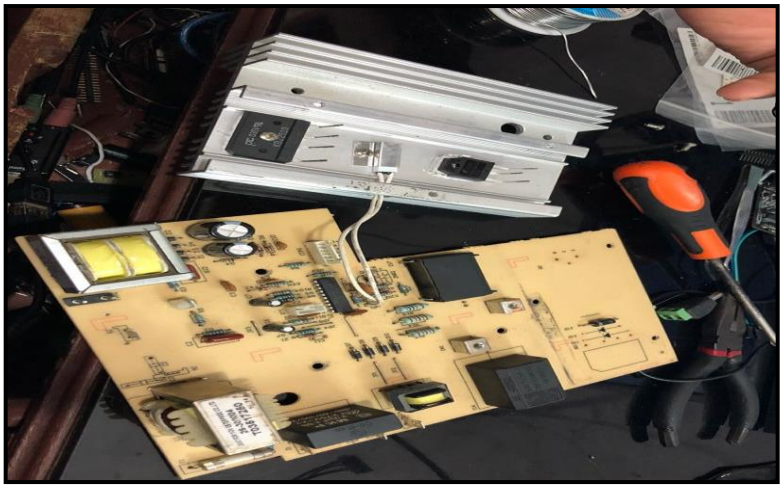


Figura 12. Montaje de elementos en la placa de fuerza del sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Figura 13. Montaje de elementos de en disipador de calor en placa de potencia.

(L. Aguirre, C. Salcedo, J. Neira, 2018)

3.4 Circuito de Control

Realizado el diseño del circuito de fuerza, se procedió a realizar el diseño del circuito de control el cual contendrá como elementos principales los siguientes equipos:

- RaspBerry PI 3
- Arduino Uno

Donde se procede a realizar las conexiones de los equipos y nos queda un circuito como el que se aprecia en el Anexo 19.

El diseño de la placa del circuito de control se elaboró en el software Altium designer el cual nos permite elegir y dimensionar el tamaño de la tarjeta con la correcta ubicación de sus componentes, y con el fin de optimizar los espacios en el tablero eléctrico se ubica las tarjetas controladoras tanto esclavo como maestro en un mismo circuito como se aprecia acontinuación.

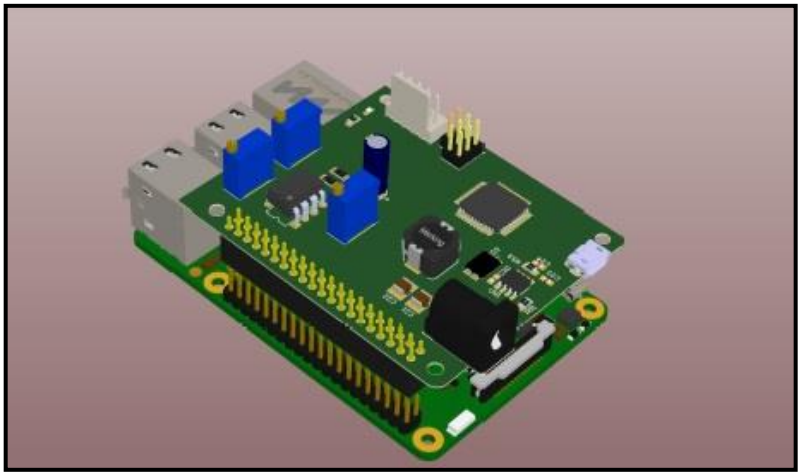


Figura 14. Enlaces de tarjetas de control

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Luego de realizar el diseño en el software Altiun designer se procedió a realizar la implementación del circuito diseñado con el montaje de los elementos en la placa de control del sistema.

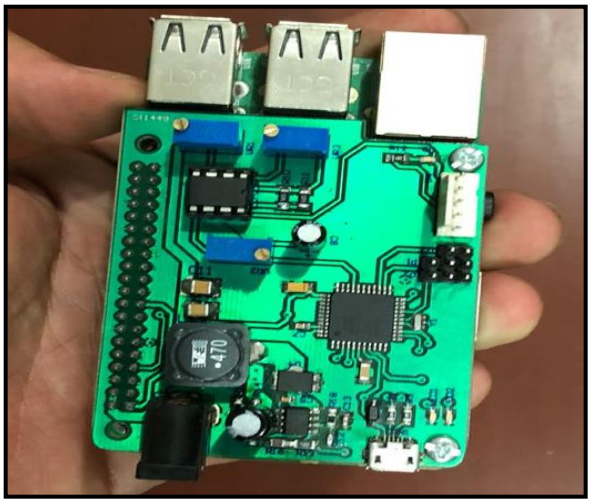


Figura 15. Elementos montados en la tarjeta de control del sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

3.4.1 Arquitectura de control

Con los elementos mencionados en la sección 3.3 se procede a realizar la arquitectura de control del sistema para identificar cual es el desempeño de cada tarjeta, donde la tarjeta RaspBerry Pi 3 actúa como módulo maestro y la tarjeta Arduino Uno como esclavo.

La arquitectura de control nos queda de la siguiente forma:

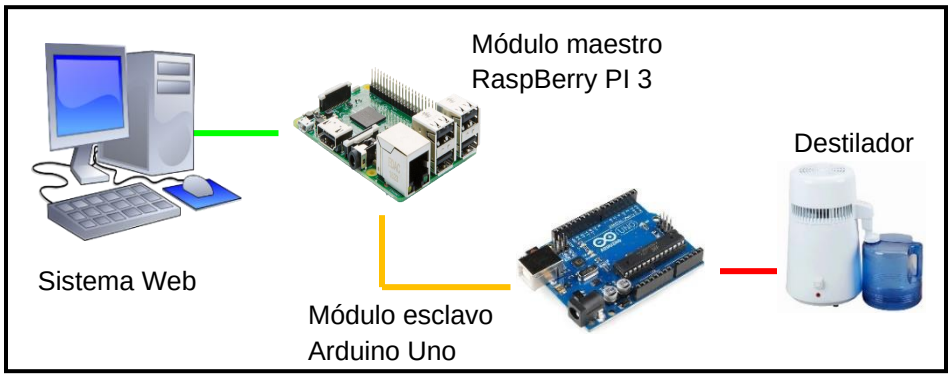


Figura 16. Arquitectura de control del sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Donde la tarjeta controladora raspberry Pi3 es la encargada de recibir los datos de consignias desde el sistema Web y enviar el voltaje adecuado al circuito de potencia, a su vez recibe la señal de temperatura del arduino quien es el que recibe la señal de temperaturay la envía a través de comunicación serial hacia la raspberry, esto lo realiza mediante la aplicación del lazo de control PID como se aprecia en la sección 2.5.1

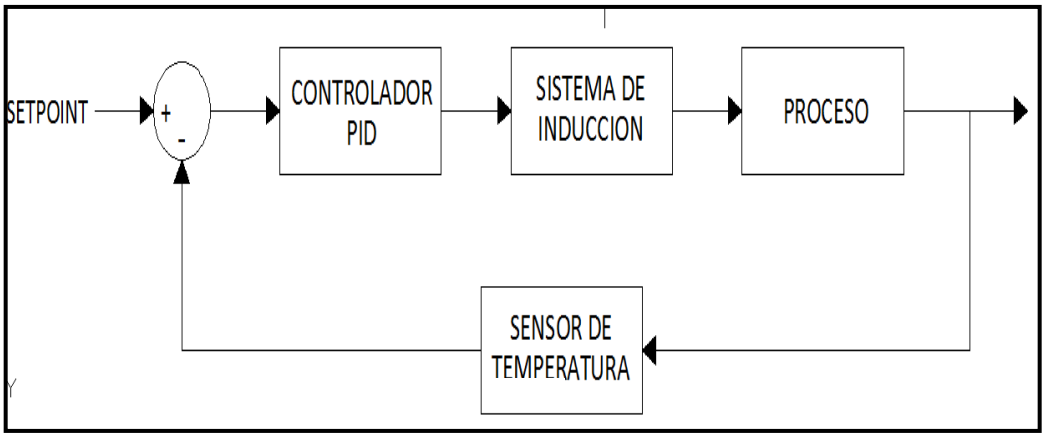


Figura 17. Lazo de control PID cerrado

(L. Aguirre, C. Salcedo, J. Neira, 2018)

3.4.2 Tablero de fuerza y control

Una vez diseñados los circuitos de fuerza y control se procedió a realizar el diseño del enlace entre ambas etapas para su correcto funcionamiento, el diseño del tablero de control nos queda de la siguiente forma.

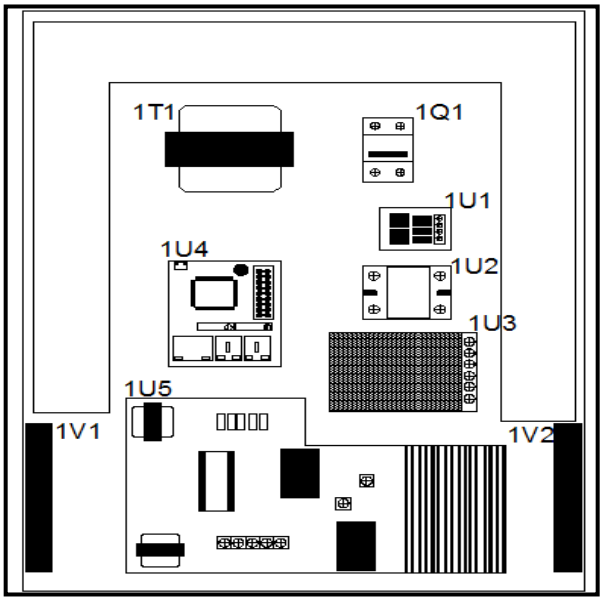


Figura 18. Vista interna de elementos de fuerza y control dentro de tablero eléctrico

(L. Aguirre, C. Salcedo, J. Neira, 2018)

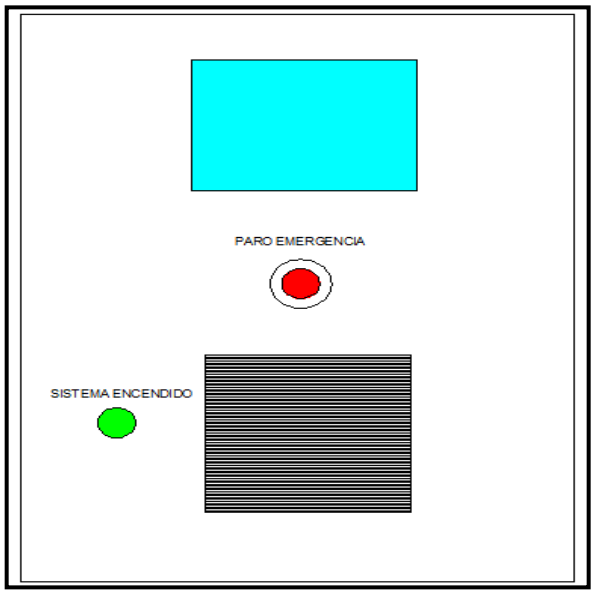


Figura 19. Vista externa de tablero eléctrico

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Figura 20. Vista Real interna de tablero eléctrico

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Figura 21. Vista externa de tablero eléctrico

(L. Aguirre, C. Salcedo, J. Neira, 2018)

3.5 Programación de Controladores

Después de realizar la arquitectura de control se procede a realizar el código de programación que ejecutará cada tarjeta controladora para que cumpla con el destilamiento adecuado

3.5.1 Programación en Raspberry PI 3

Para que realice el proceso de destilación según lo requerido se procede a a crear 3 rutinas de programación dentro de la tarjeta controladora RapBerry PI 3 para el cual se utilizó el lenguaje de programación llamado PHYTON, en este tipo de lenguaje se realizaron 3 rutinas de programación las cuales se describen a continuación:

Rutina de programación Principal

Es la encargada de realizar el análisis de las configuraciones iniciales de la tarjeta y de realizar el llamado a las otras rutinas de programación.

```
# -*- coding: utf-8 -*-
import numpy
import matplotlib.pyplot as plot
import portCom
import pid

plot.ion()
serial=portCom.PortCom(port='com12',baud_rate=9600)

set_point=100
pid=pid.Pid(set_point=set_point,kp=40.2,ki=0.001,kd=23.7)

try:
    t=0
    while True:
        medition=ord(serial.read())
        power=pid.compute(value=medition)
        serial.write(str(power))
        plot.pause(0.01)
except KeyboardInterrupt as exception:
    pass
```

Figura 22. Rutina de Programación Principal

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Rutina de programación PID

En esta rutina de programación el programa ejecuta el lazo de control PID con los valores KP, KI, KD, para realizar el correcto control del sistema.

```
# -*- coding: utf-8 -*-
class Pid:
    lastProportional=0
    integrative=0
    def __init__(self,set_point,kp=0,ki=0,kd=0,max=255,min=0):
        self.set_point=set_point
        self.kp=kp
        self.ki=ki
        self.kd=kd
        self.max=max
        self.min=min
    def compute(self,value):
        proportional=self.set_point-value
        derivative=proportional-self.lastProportional
        self.integrative=self.integrative+proportional
        if self.integrative>self.max:
            self.integrative=self.max
        if self.integrative<self.min:
            self.integrative=self.min
        self.lastProportional=proportional
        power=proportional*self.kp+self.integrative*self.ki+derivative*self.kd
        if power>self.max:
            power=self.max
        if power<self.min:
            power=self.min
        return power
```

Figura 23. Rutina de Programación PID

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Rutina de programación Comunicación

Aquí es donde se ejecuta el código para la transmisión de datos entre el sistema Web y la tarjeta controladora.

```
#-*- coding: utf-8 -*-
import serial
class PortCom:
    def __init__(self,port,baud_rate):
        self.port=serial.Serial(port,baud_rate)
    def read(self):
        return self.port.read()
    def read_line(self):
        character=self.port.read()
        data=""
        while character!="\r":
            data=data+character
            character=self.port.read()
        self.port.read()
        return data
    def write(self,data):
        self.port.write(data)
```

Figura 24. Rutina de Programación Comunicación

(L. Aguirre, C. Salcedo, J. Neira, 2018)

3.5.2 Programación en Arduino Uno

La tarjeta controladora Arduino Uno es la encargada de receptar las señales que genera el sistema de destilación y enviarlos al módulos maestro RaspBerry PI 3 a través de su puerto serial, el cual queda de la siguiente forma:

```
#define BAUD_RATE 9600

#define FROM_MIN 0
#define FROM_MAX 1023
#define TO_MIN 0
#define TO_MAX 255

#define PIN_OUT 10

int out=0;

void setup()
{
   Serial.begin(BAUD_RATE);
   pinMode(PIN_OUT,OUTPUT);
}

void loop()
{
   out=analogRead(A0);
   out=map(out,FROM_MIN,FROM_MAX,TO_MIN,TO_MAX);
   Serial.write(out);
   if(Serial.available())
   {
      String in="";
      while(Serial.available())
      {
         in+=(char)Serial.read();
      }
      analogWrite(PIN_OUT,in.toInt());
   }
   delay(20);
}
```

Figura 25. Programación en Arduino Uno

(L. Aguirre, C. Salcedo, J. Neira, 2018)

3.6 Programación de Matlab

Una vez realizado el control mediante la raspberry PI 3 y con el fin de realizar el control del sistema mediante un software más didáctico, se procedió a realizar la programación para controlar el sistema mediante el software Matlab.

Procedimos a abrir la herramienta Simulink la cual nos ayuda a realizar el diagrama de control del sistema para su correcto comportamiento.

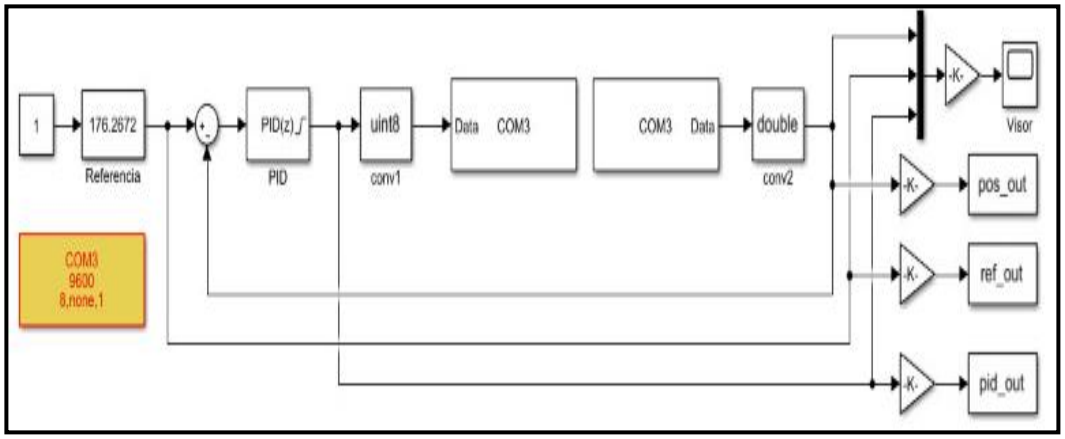


Figura 26. Diseño de programación en Simulink

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Primero definimos los parámetros de comunicación del puerto donde se asignan los baudios y el tipo de paridad con la que se va a comunicar el software Matlab con la interfaz arduino.

Figura 27. Parametros de comunicación de puerto serial entre Matlab y Arduino

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Luego agregamos un bloque de entrada el cual lo llamamos referencia el cual contendrá el valor de setpoint del sistema.

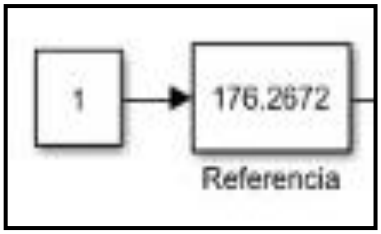


Figura 28. Asignación de Setpoint en el sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Para continuar con el diagrama de control de un lazo PID se procede a agregar el bloque de comparación el cuál dará como resultado el error entre el setpoint y la temperatura actual del sistema, seguido de este comparador se procede a colocar el control bloque PID el cual se encargará de corregir el error presente después del comparador.

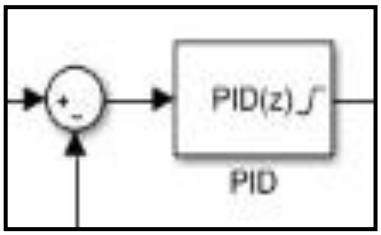


Figura 29 Asignación de bloque PID en el sistema.

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Luego se realiza el envío del valor de consigna de temperatura al sistema una vez realizada la corrección del error del bloque PID, para esto se añade los bloques de transferencia de datos indicando el puerto de comunicación por el cual va a enviar los datos a la interfaz de señales Arduino, el cual a su vez actúa sobre el sistema para lograr alcanzar la temperatura de consigna.

Figura 30 Envío de datos de temperatura através del puerto serial.

(L. Aguirre, C. Salcedo, J. Neira, 2018)

3.6.1 Cálculos de los parámetros PID a través de Matlab

Para hallar los valores KP, KI, KD que se utilizarán para afinar el control de temperatura en el sistema de destilación se realizó de la siguiente manera:

Creación del vector de entrada

Se procede a crear el vector de entrada v1 gracias a los valores capturados desde la Raspberry pi 3 el cual permite almacenar en un archivo de texto dichos valores, este vector contendrá los datos de temperartura obtenidos durante las pruebas de ensayo, las pruebas consistieron en realizar un proceso de destilación con valores PID aleatorios para poder obtener valores y que el sistema nos arroje los valores óptimos.

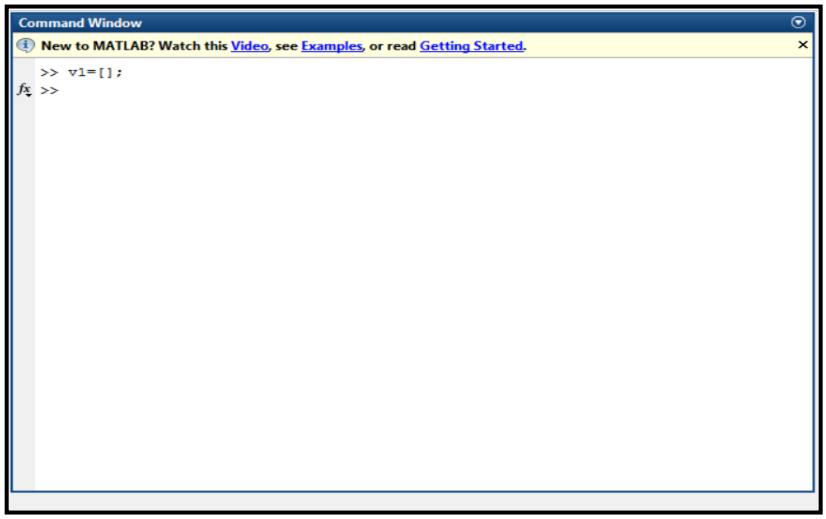


Figura 31. Creación del vector de entrada

(L. Aguirre, C. Salcedo, J. Neira, 2018)

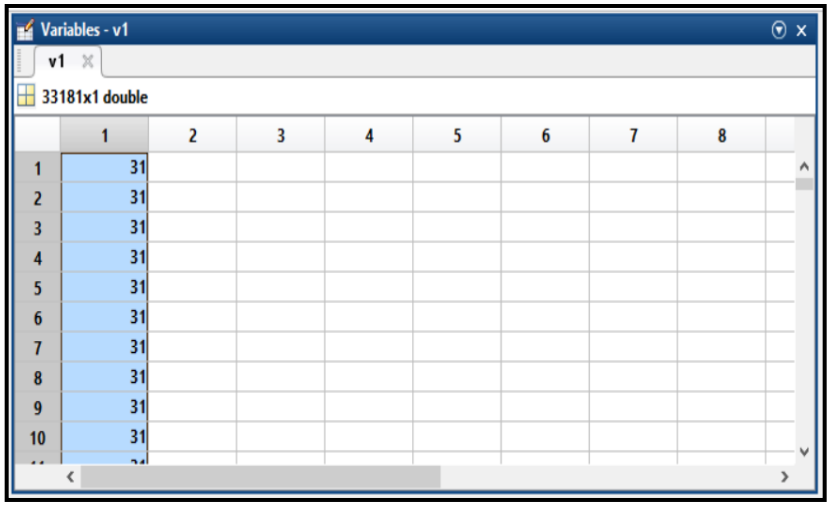


Figura 32. Ingreso de datos de temperatura

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Creación del vector de consigna

Se procede a crear el vector de entrada r1, el cual es el valor que contendrá el valor de consigna del sistema y tiene que ser del mismo tamaño del vector de entrada.

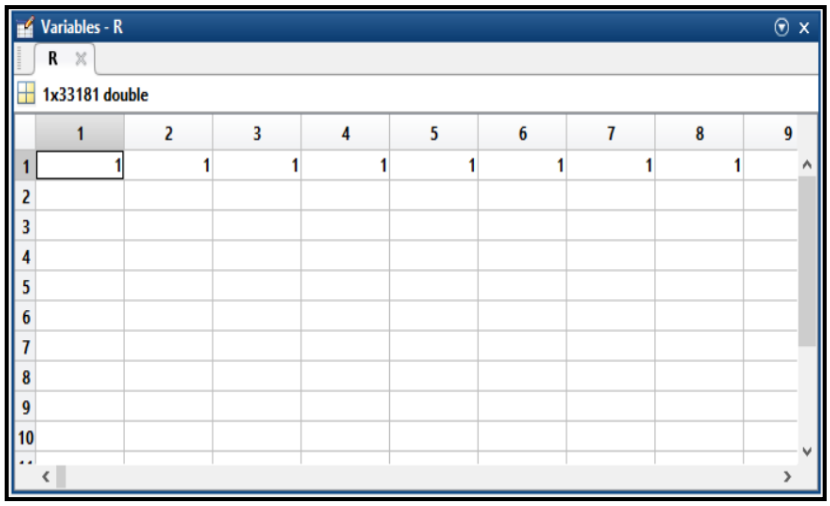


Figura 33. Ingreso de datos de consigna

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Este mismo vector se procede a sacarle la traspuesta y multiplicarlo por 80 el cual es el valor de consigna del sistema para que quede del mismo tamaño que el vector de entrada v1.

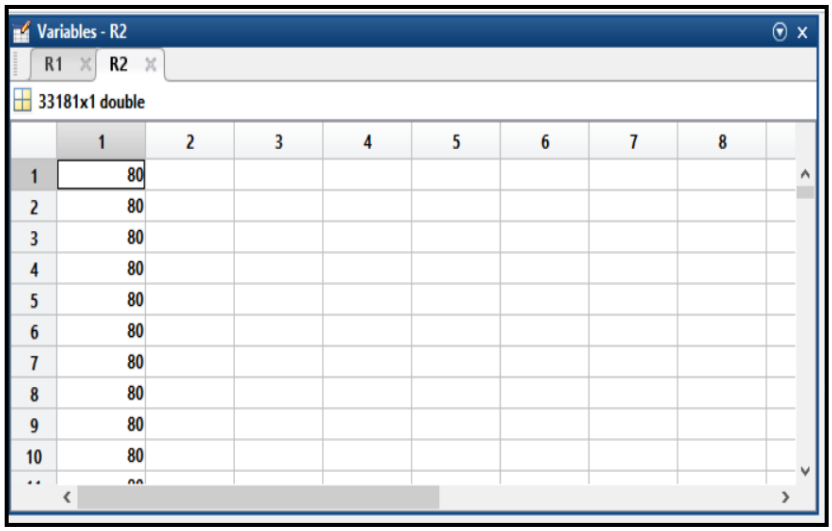


Figura 34. Traspuesta del vector consigna

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Una vez creados el vector de entrada y el vector de consigna se procede a utilizar el comando Ident de Matlab el cual nos da apertura al software para estimar la función de transferencia el cual System Identification Tool la proporcionará

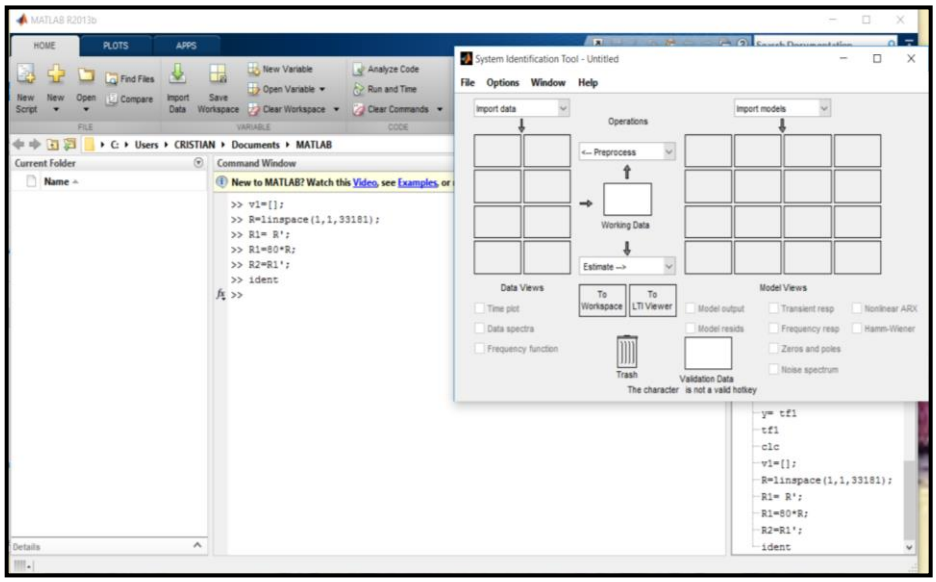


Figura 35. Herramienta System Identification Tool

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Se llama a los vectores creados de entrada (R2) y salida (v1), para asi importa los datos a la función del ident del Matlab.

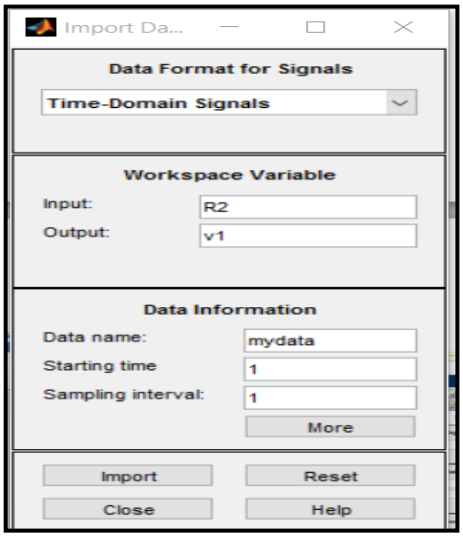


Figura 36. Importación de los vectores

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Estimación de la función de transferencia

Una vez creado el sistema en el System Identification Tool se procede a estimar la función de transferencia en el dominio del tiempo.

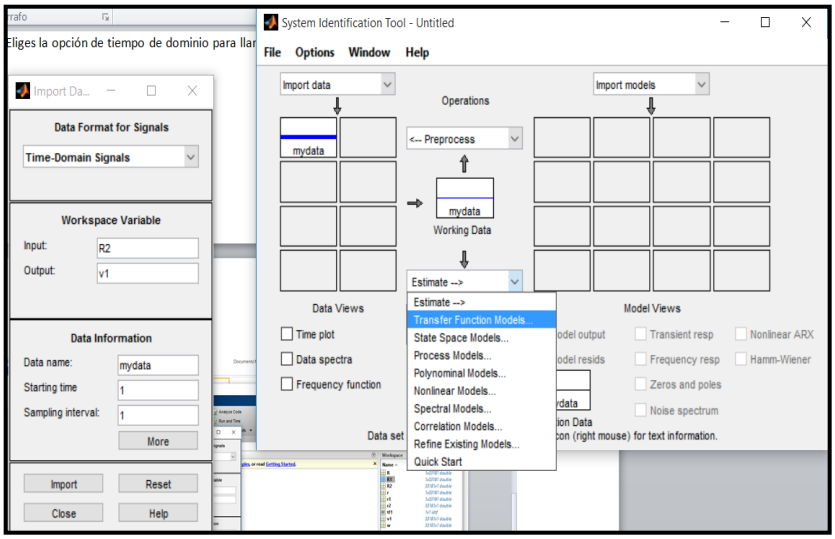


Figura 37. Estimación de la función de transferencia

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Luego de estimar la función de transferencia, se deberá tener seleccionada la misma para arrastrarla al workspace del software y obtener nuestra función de transferencia.

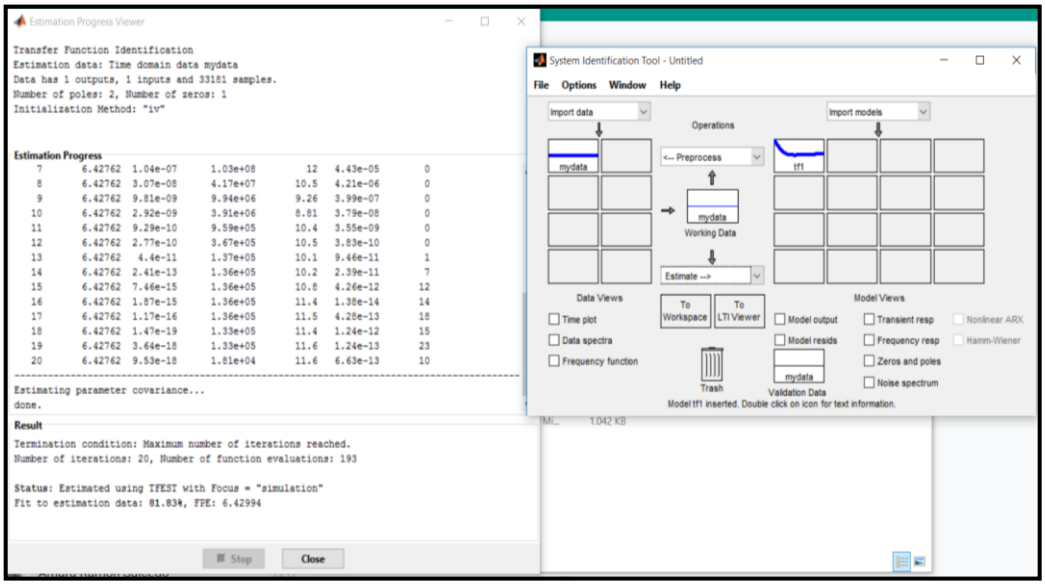


Figura 38. Traslado de la función de transferencia

(L. Aguirre, C. Salcedo, J. Neira, 2018)

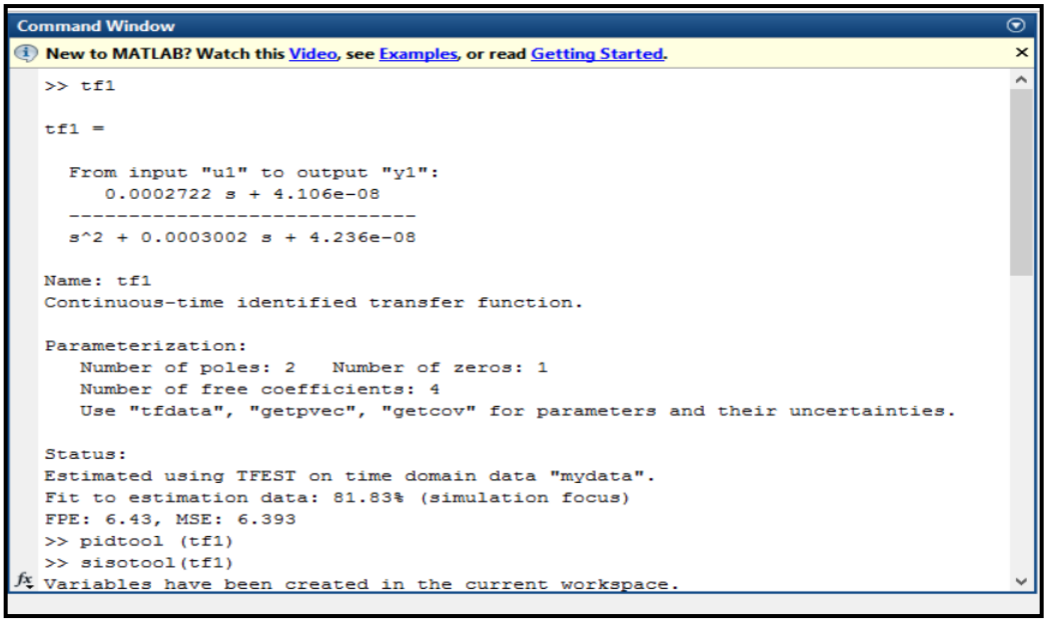


Figura 39. Función de transferencia

(L. Aguirre, C. Salcedo, J. Neira, 2018)

PID tool

Luego de encontrar la función de transferencia se procede a utilizar la herramienta del software Matlab PID Tool el cual nos permitirá ajustar los tiempos de respuesta, overshoots para sintonizar de mejor manera los parámetros P. I. D.

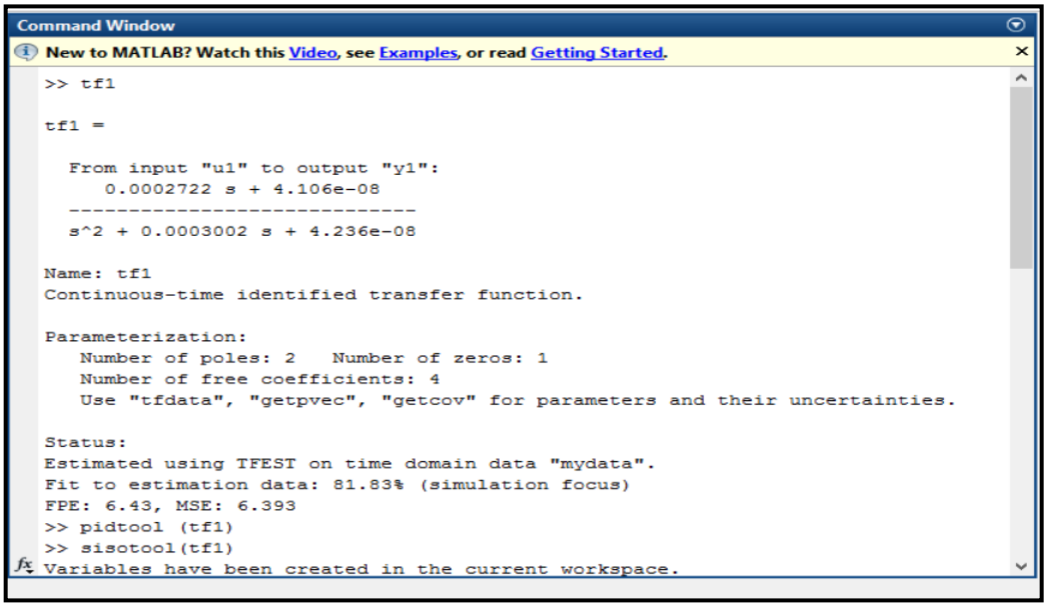


Figura 40. PID tool

(L. Aguirre, C. Salcedo, J. Neira, 2018)

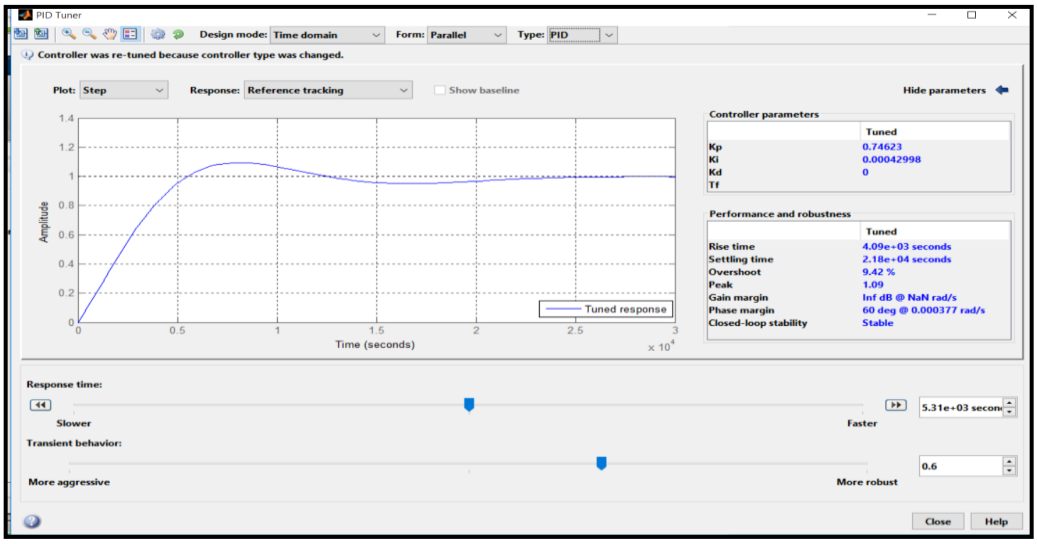


Figura 41. Salida análoga del sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Pese a que se nota el comportamiento correcto del sistema se realiza una mejora en la salida para optimizar los valores de tiempo de respuesta, overshoots y encontrar los mejores valors P. I. D.

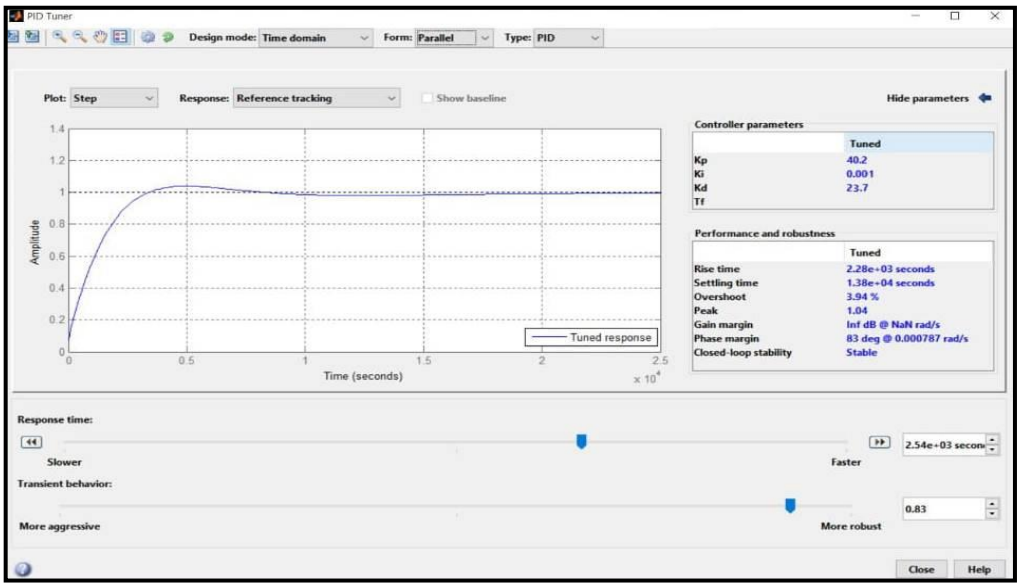
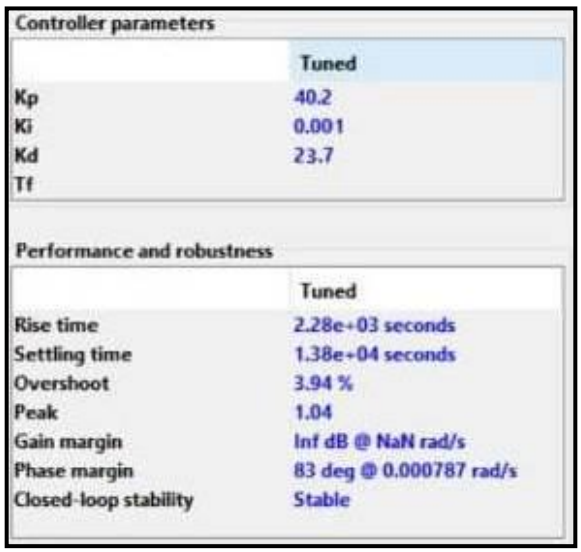


Figura 42. Salida análoga del sistema mejorada

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Luego de verificar que el comportamiento de la salida del sistema se ajuste a lo requerido los valores finales P. I. D. obtenidos son los siguientes



Controller parameters

	Tuned
Kp	40.2
Ki	0.001
Kd	23.7
Tf	

Performance and robustness

	Tuned
Rise time	2.28e+03 seconds
Settling time	1.38e+04 seconds
Overshoot	3.94 %
Peak	1.04
Gain margin	Inf dB @ NaN rad/s
Phase margin	83 deg @ 0.000787 rad/s
Closed-loop stability	Stable

Figura 43. Valores P.I.D. obtenidos

(L. Aguirre, C. Salcedo, J. Neira, 2018)

3.6.2 Comportamiento del sistema desde Simulink

Una vez realizado la programación en Simulink y obtenidos los parámetros KP, KI, KD se procede a poner en marcha el sistema.

Observamos el comportamiento del sistema mediante el bloque colocado en la programación realizada en Simulink Scope, el cual nos permite visualizar las variables:

- Referencia o Setpoint
- Temperatura actual
- Salida del sistema

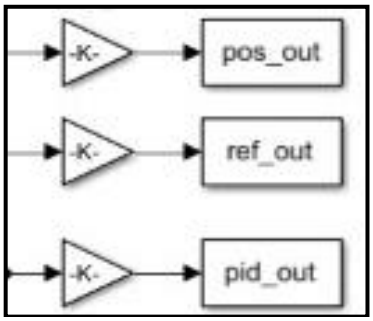


Figura 44. Variables de visualización de comportamiento del sistema.

(L. Aguirre, C. Salcedo, J. Neira, 2018)

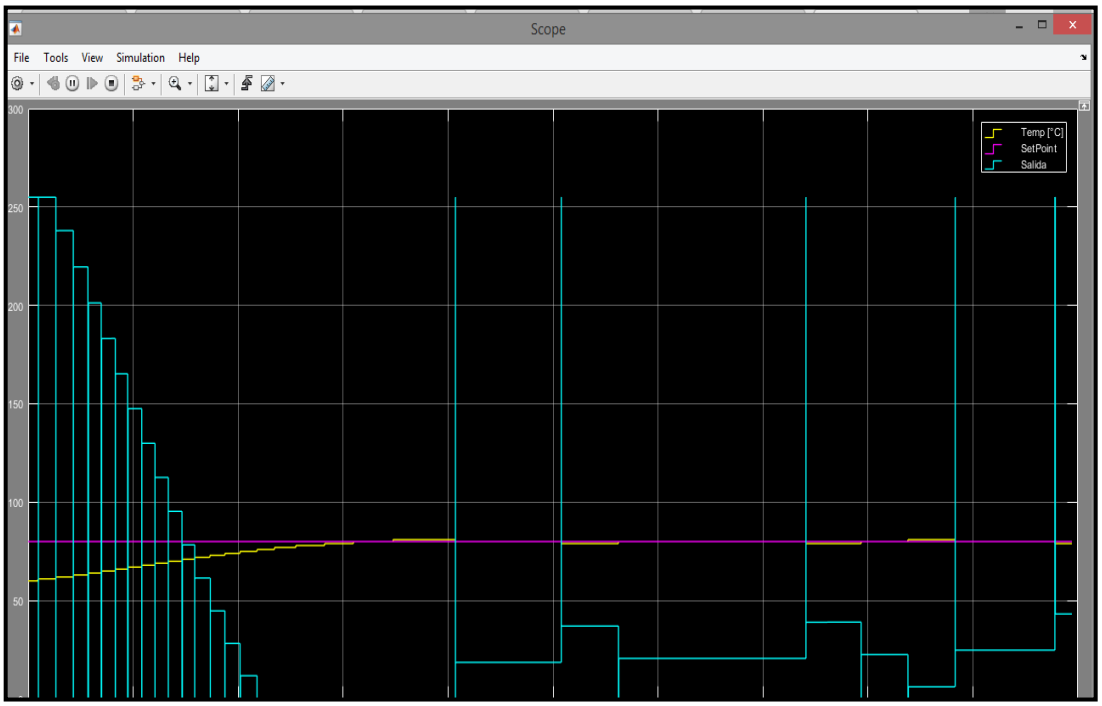


Figura 45. Comportamiento del sistema en Simulink

(L. Aguirre, C. Salcedo, J. Neira, 2018)

En la gráfica podemos obervar el comportamiento del sistema y las variables:

- Setpoint
- Salida
- Temperatura

Con lo cual validamos el correcto comportamiento del controlador PID ya que la temperatura logra alcanzar y mantenerse en el valor de referencia o Setpoint.

3.7 Programación de Sistemas

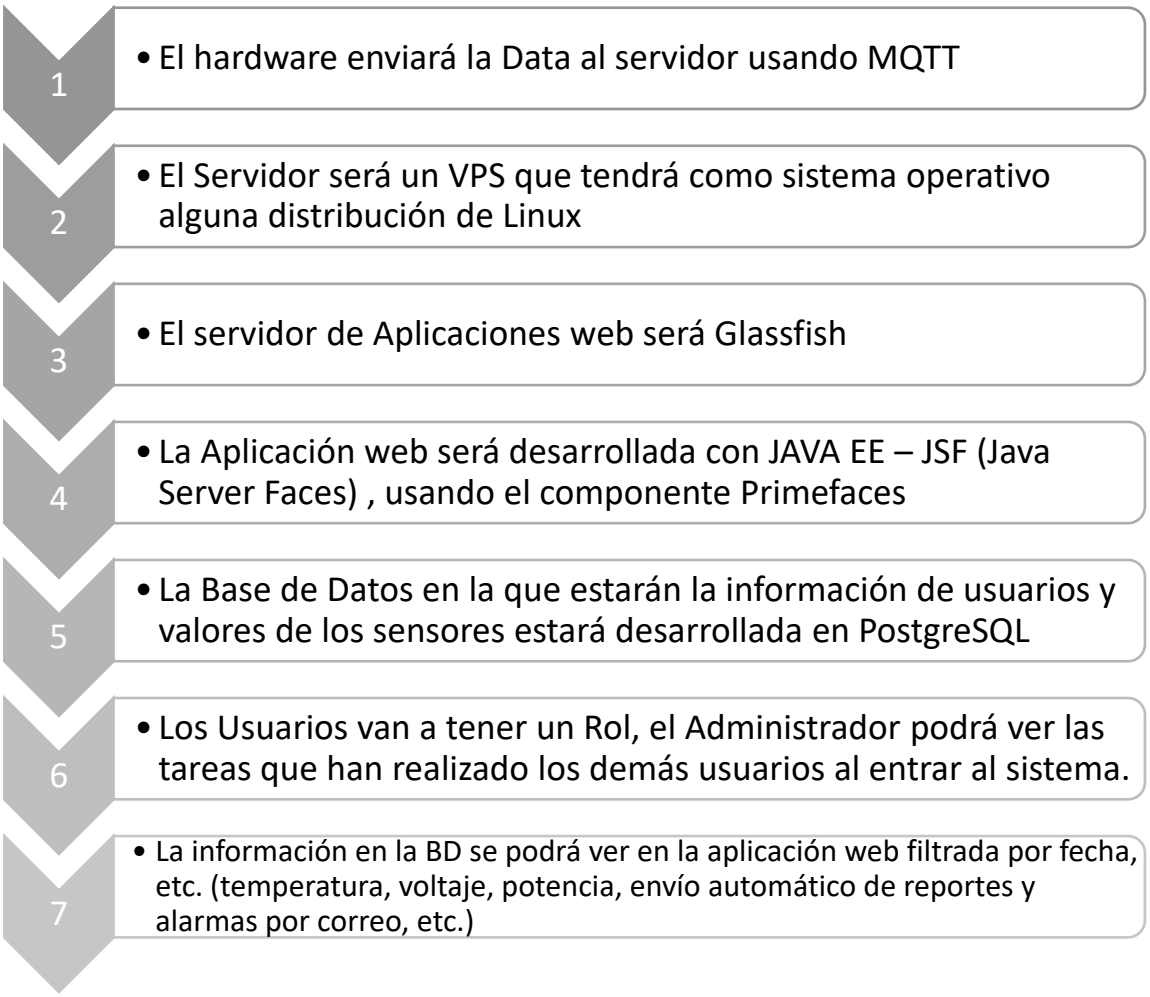
Una vez realizada la parte electrónica se procedió a realizar la programación del sistema de registro de datos y visualización en el sistema Web, el cual mostrará la siguiente información:

- Nombre del Operador
- Encendido / Apagado del equipo
- Numero de Bachs diarios, mensuales o anuales
- Registro de temperatura del Hexano
- Datos estadísticos de temperatura como valores máximos y mínimos, media aritmética, desviación estándar, etc.
- Mensajes como Inicio de destilación y Final de destilación

- Al finalizar la destilación se genera un reporte en donde se mostrará el nombre del operador, el tiempo total del Batch, los datos estadísticos de temperatura, etc.

3.7.1 Pasos para el envío de datos

Para mostrar esta información el sistema realizará los siguientes pasos:



3.7.2 Roles a implementar

Administrador

- Puede ver quienes han operado el sistema
- Tiempo de duración del proceso
- Valores máximos y mínimos de temperatura que alcanzaron
- Cantidad de hexano que se obtuvo

Visitante

- Puede colocar nombre de operador

- Puede ingresar número de veces que hexano se destilará
- Puede ingresar temperatura en la que quiere que se mantenga para la destilación

Esto se lo realizará con la ayuda de una tarjeta Raspberry (sistema embebido), se realizará el código en Python y por este medio se leen los valores del sensor y se hace un código para aplicar el control PID y toda la data se envía al servidor usando el protocolo MQTT ya que ese protocolo se usa para el internet de las cosas.

3.7.3 Programación de la página web en Java

El Método con el cual se está vinculado el evento del botón encender/apagar el equipo, envía por UDP un carácter que será recibido por el servidor UDP que está corriendo en el mismo servidor, para que se haga la debida comunicación con la tarjeta raspberry por medio de una conexión (protocolo) TCP/IP.

```
    public void encenderEquipo() {

        EquipoDAO.actualizarEstado(estadoEquipo);
        setEstadoEquipo(EquipoDAO.obtenerEstadoEquipo());

        if (estadoEquipo) {
            insertarLog("encendio Equipo");
            enviarUDP("C");

        } else {
            insertarLog("Apago Equipo");
            enviarUDP("D");
}
    }
```

Figura 46. Comunicación mediante Protocolo TCP/IP

(L. Aguirre, C. Salcedo, J. Neira, 2018)

El presente Método está vinculado al evento del botón que establece el inicio de la destilación, toma los parámetros que sean necesarios para realizar él envió de datos vía
UDP para así poder enviar al raspberry el setpoint de la destilación.

```
    public void destilar() {
System.out.println(destilando);
        if (!destilando) {
Object[] datos = EquipoDAO.obtenerIdBach();
            EquipoDAO.actualizarDestilacion("B");
            enviarUDP("B");
            insertarLog("Detuvo Destilacion");

            if (datos != null) {
int id = (int) datos[0];
                String hora = String.valueOf(datos[1]);
                String tiempo = String.valueOf(datos[2]);
                String numDestilaciones =
String.valueOf(EquipoDAO.obtenerNumeroDestilaciones(getCurrentDate()));
                String tiempoAhora = new SimpleDateFormat("HH:mm:ss").format(new
Date());

//  idDestilacion , numeroDestilacion, Hora , tiempo
                enviarUDP(id + ";" + numDestilaciones + ";" + hora + ";" + tiempo + ";" +
tiempoAhora);
            }

        } else {
            EquipoDAO.actualizarDestilacion("B");

            BachDao.insertarBach(sep, tiempo);
            boolean b = EquipoDAO.obtenerEstadoDestilacion();
            setDestilando(b);
            if (b) {
                enviarUDP("A," + sep.trim());
                insertarLog("Inicio Destilacion");
            }
        }
```

Figura 47. Envío de Datos Vía UDP - 1

(L. Aguirre, C. Salcedo, J. Neira, 2018)

En este método vinculado al botón de verificación del correo contrasta los hilos cada 5 segundos si se muestra una destilación activa que no haya finalizada y su tiempo de

destilación ya haya excedido, en el caso que se confirme una destilación se envía un correo notificando a los administradores que es hora de apagar la destilación.

```
    public static void verificarCorreo() {
        new Thread(new Runn   able() {
            @Override
            public void run() {
                while (true) {
                    try {
                        Thread.sleep(5000);
Object[] datos = EquipoDAO.obtenerDatosBach();

                        if (datos != null) {
int id = (int) datos[0];
                            String hora = String.valueOf(datos[1]);
                            String tiempo = String.valueOf(datos[2]);

                            SimpleDateFormat df = new SimpleDateFormat("HH:mm:ss");
Date d = df.parse(hora);
Calendar cal = Calendar.getInstance();
                            cal.setTime(d);
                            String timpos[] = tiempo.split(":");
                            cal.add(Calendar.HOUR, Integer.parseInt(timpos[0]));
                            cal.add(Calendar.MINUTE, Integer.parseInt(timpos[1]));
                            cal.add(Calendar.SECOND, Integer.parseInt(timpos[2]));
                            String tiempoTotal = df.format(cal.getTime());

                            DateFormat dfx = new SimpleDateFormat("HH:mm:ss");
Date horaTerminar = dfx.parse(tiempoTotal);
String tiempoInicio = new SimpleDateFormat("HH:mm:ss").format(new Date());
Date horaInicio = dfx.parse(tiempoInicio);
String tiempoInicio = new SimpleDateFormat("HH:mm:ss").format(new Date());
Date horaInicio = dfx.parse(tiempoInicio);
                            if (horaInicio.compareTo(horaTerminar) > 0) {
                                if (!yaSeEnvio(id)) {
                                    String mensaje = "<p><span style=\"color:
#ff0000;\"><strong>DEBE DETENER LA DESTILACION QUE INICIO A LAS " +
hora + " HORAS</strong></span></p>";
                                    GmailSendEmailSSL.enviarMensaje(mensaje, "Alerta!
```

```
Detener Destilacion");
                                listaEnviados.add(id);
                            }

                        }

                    }
                } catch (Exception e) {
}

            }

        }
    }).start();
  }
```

Figura 48. Envío de Datos Vía UDP - 2

(L. Aguirre, C. Salcedo, J. Neira, 2018)

En este método vinculado al botón iniciar conexión por protocolo de comunicación TCP/IP para iniciar el servidor TCP que se comunicará con el Raspberry para enviarle caracteres de encendido/apagado, iniciar/detener destilación.

```
  public static void iniciarTCP() {
      new Thread(new Runnable() {
          @Override
          public void run() {
              try {
                  while (true) {
                      try {
                          ServerSocket welcomeSocket = new ServerSocket(5005);

                          Socket connectionSocket = welcomeSocket.accept();

                          raspberry = new
DataOutputStream(connectionSocket.getOutputStream());
System.out.println("Se Conectó el Equipo");
} catch (Exception e) {
                      }

                  }

              } catch (Exception e) {
```

Figura 49. Inicio de Servidor TCP

(L. Aguirre, C. Salcedo, J. Neira, 2018)

En las presentes líneas de código se inicia con el servidor UDP que estará en comunicación con la página web enviando datos a la raspberry mediante el protocolo de comunicación tcp/ip, también se procede a enviar al correo cuando la destilación haya sido detenida, de los cuales se obteniendo el cálculo de los siguientes valores: Temperatura Máxima, mínima, desviación estándar etc.

```
public static void udpInterno() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    DatagramSocket serverSocket = new DatagramSocket(9876);
                    byte[] receiveData = new byte[1024];

while (true) {
                        receiveData = new byte[1024];
                        DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
                        serverSocket.receive(receivePacket);
                        String sentence = new String(receivePacket.getData());
                        System.out.println("Data: " + sentence);
                        String datos[] = sentence.trim().split(";");
                        if (datos.length == 5) {
                            String idDestilacion = datos[0];
String numeroDestilaciones = datos[1];
                            String hora = datos[2];
                            String tiempo = datos[3];
                            String tiempoAhora = datos[4];
                            List<Registro> list = EquipoDAO.obtenerRegistros(idDestilacion);
String temMax = EquipoDAO.obtenerTemMaxBach(idDestilacion);
String temMin = EquipoDAO.obtenerTemMinBach(idDestilacion);
                            String media = EquipoDAO.obtenerMediaAritmetica(list);
                            String desviacion = EquipoDAO.obtenerDesvia(list);
                            String infoCorreo = "<p><span style=\"color:
#0000ff;\"><strong>Se detuvo la Destilacion #" + numeroDestilaciones + " a las " +
tiempoAhora + " horas, que inicio a las " + hora + " horas y debio estsar destilando
durante el tiempo de " + tiempo + "</strong></span></p>\n"
+ "<table style=\"height: 104px; width: 400px;\" border=2>\n"
                                    + "<tbody>\n"
                                    + "<tr>\n"
                                    + "<td style=\"width: 192px;\"
bgcolor=\"#00ff00\">Temperatura Maxima:</td>\n"
```

```
bgcolor=\"#00ff00\">Temperatura Minima:</td>\n"
                            + "<td style=\"width: 196px;\" bgcolor=\"#bfff00\">" + temMin
+ "</td>\n"
                            + "</tr>\n"
                            + "<tr>\n"
                            + "<td style=\"width: 192px;\" bgcolor=\"#00ff00\">Media
Aritmetica:</td>\n"
                            + "<td style=\"width: 196px;\" bgcolor=\"#bfff00\">" + media +
"</td>\n"
                            + "</tr>\n"
                            + "<tr>\n"
                            + "<td style=\"width: 192px;\"
bgcolor=\"#00ff00\">Desviacion estandar:</td>\n"
                            + "<td style=\"width: 196px;\" bgcolor=\"#bfff00\">" +
desviacion + "</td>\n"
                            + "</tr>\n"
                            + "</tbody>\n"
                            + "</table>";
GmailSendEmailSSL.enviarMensaje(infoCorreo, "Destilacion Detenida");

} else {
  public static void mqtt() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                MqttSubscribe.main(null);
            }
        }).start();
    }
```

Figura 51. Inicio de Suscriptor MQTT

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Estas líneas de código presenta la Clase que contiene el Subscriptor MQTT, esté recibirá y almacenará datos que obtendrá el raspberry con los sensores mientras haya una destilación activa.

```
public class MqttSubscribe {

	public static void main(String[] args) {

		String topic = "Tesis";
		int qos = 2;
		String broker = "tcp://localhost:1883";
		String clientId = "SubscribeSample";
		MemoryPersistence persistence = new MemoryPersistence();

		try {
			MqttClient sampleClient = new MqttClient(broker, clientId,
					persistence);
			sampleClient.setCallback(new MqttCallback() {

				@Override
				public void messageArrived(String topic,
MqttMessage msg)
						throws Exception {
					System.out.println("topic:" + topic);
					System.out.println("msg:" + new

String(msg.getPayload()));
                                String data= new String(msg.getPayload());
                                String datos[]= data.trim().split(";");
                                float temperatura=Float.parseFloat(datos[0]);
                                float voltaje=Float.parseFloat(datos[1]);
                                float potencia=Float.parseFloat(datos[2]);
                                if(datos.length==3){
                                  int idB= EquipoDAO.obtenerIdBach();
if(idB!=-1){
                                    EquipoDAO.insertarRegistro(idB, temperatura,
voltaje, potencia);
}
```

Figura 52. Contenido de Suscriptor MQTT

(L. Aguirre, C. Salcedo, J. Neira, 2018)

3.7.4 Diseño de la página web de control de destilación

Inicio de sesión de los administradores del laboratorio

En esta pantalla iniciamos sesión según el rol que nosotros tengamos como se puede apreciar en la sección 3.6.2

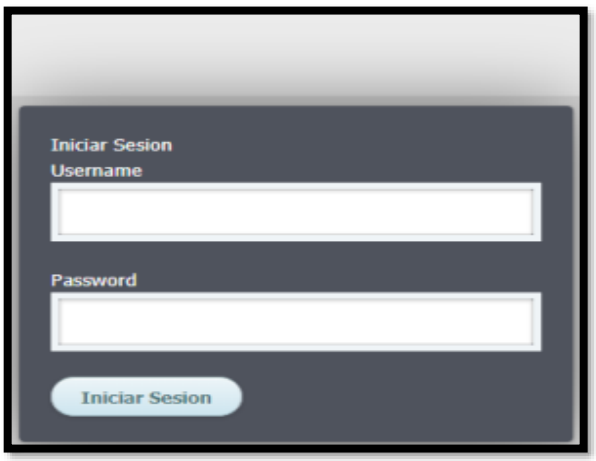


Figura 53. Inicio de Sesión

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Tiempo de control de nuestro sistema de destilación

Estamos en la ventana de control donde se pueden establecer la mayoría de las funciones principales de nuestro sistema dentro de las cuales están las siguientes configuraciones:

- Tiempo de duración de un proceso de destilación
- Valor de consigna o setpoint de la materia prima
- El tiempo en que se debe parar la destilación
- Las temperaturas máximas y mínimas por seguridad del sistema
- Destilaciones por día que se desee hacer
- Encender o apagar un proceso de destilación

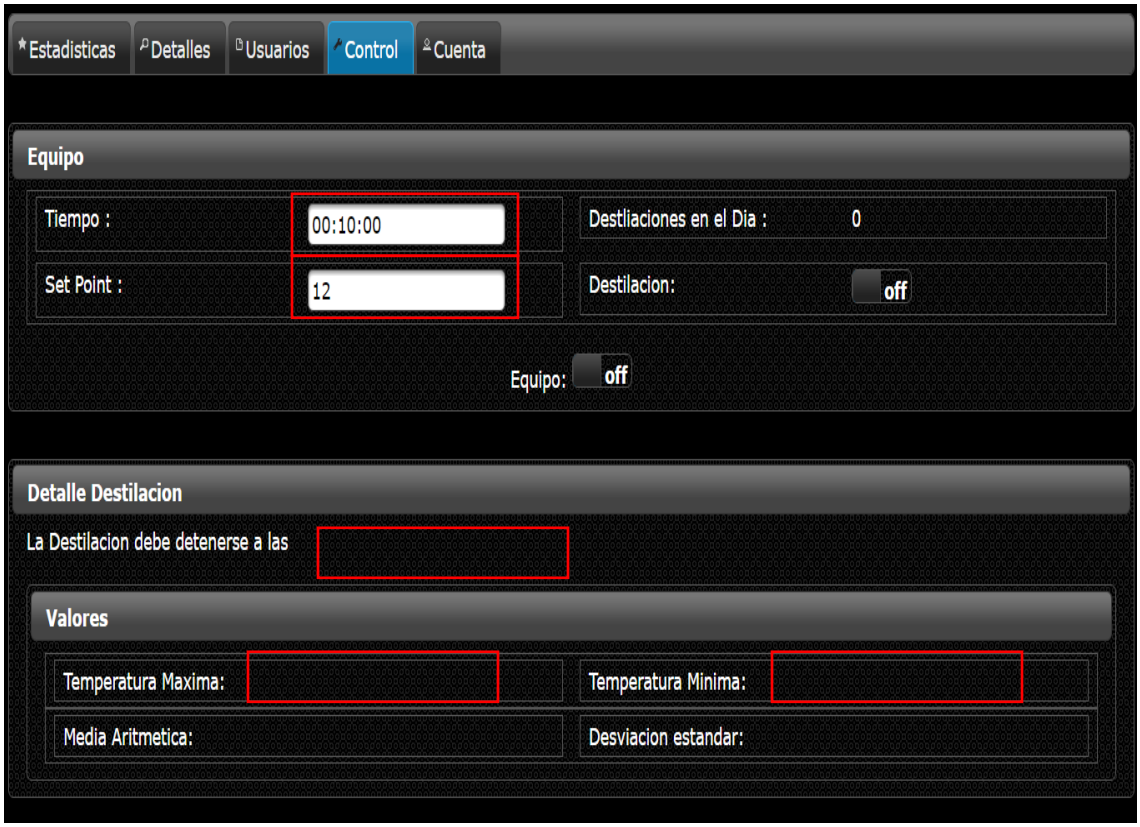


Figura 54. Ventana de Control del Sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Reporte

La ventana estadística nos muestra un reporte previo mientras el sistema está en funcionamiento donde podemos apreciar la fecha en la que esta operando y el número de batch que está destilando.

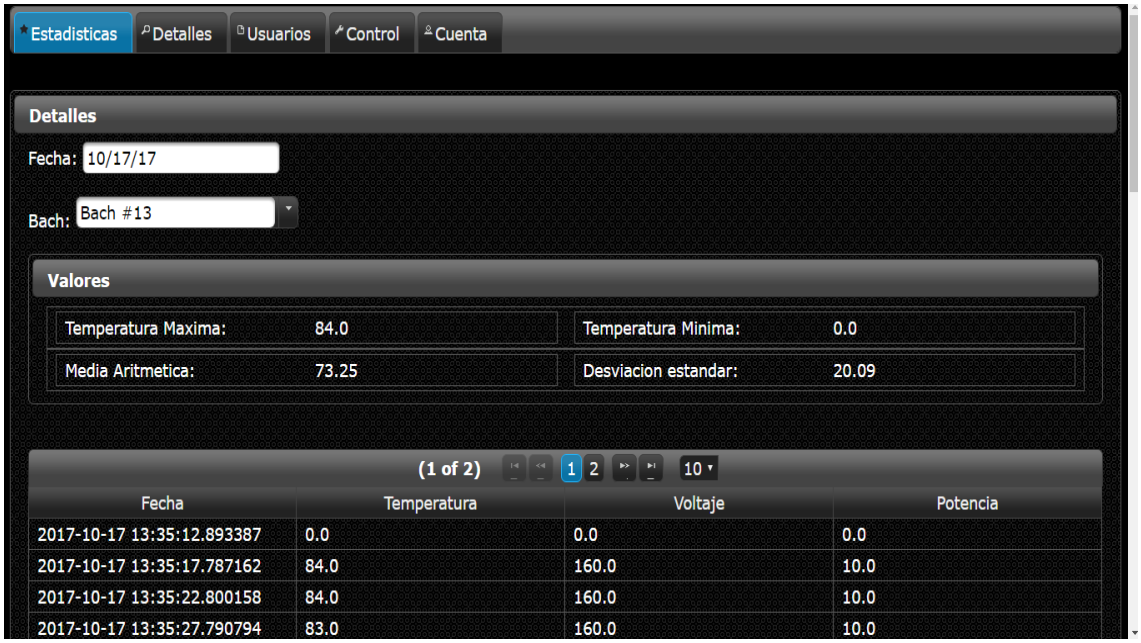


Figura 55. Ventana de estadísticas del sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Tendencias

En las siguientes gráficas vemos el comportamiento de temperatura, potencia y setpoint del sistema en función del tiempo.

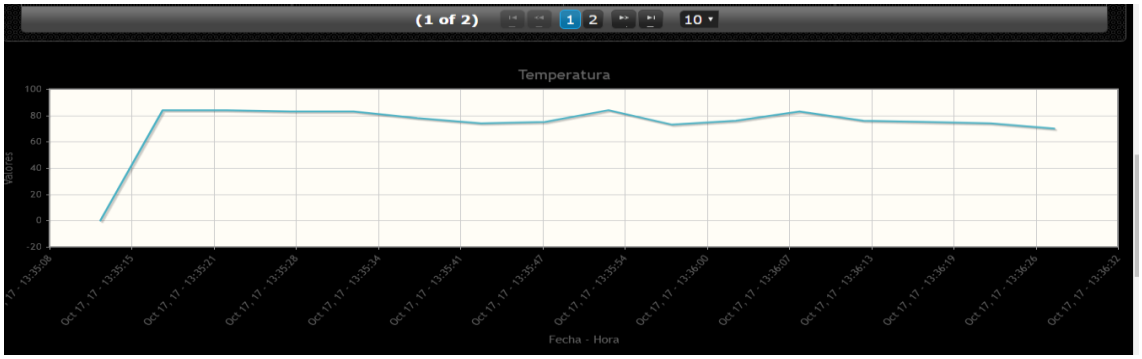


Figura 56. Señal de temperatura en función del tiempo

(L. Aguirre, C. Salcedo, J. Neira, 2018)

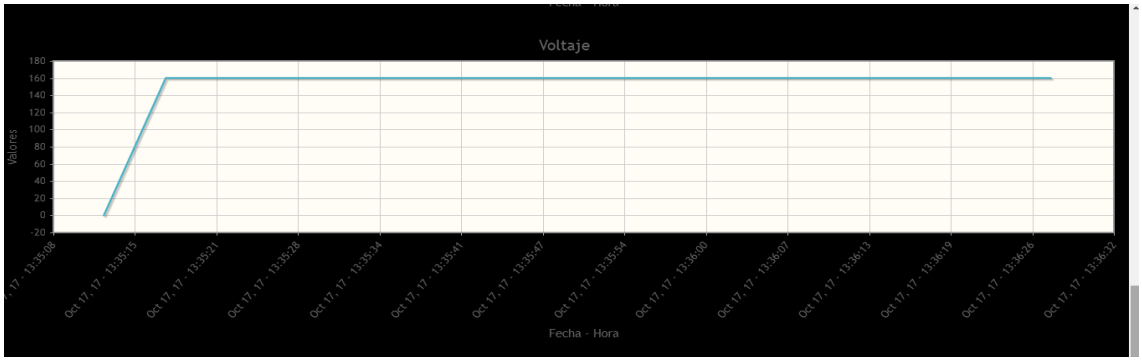


Figura 57. Señal de Setpoint del sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

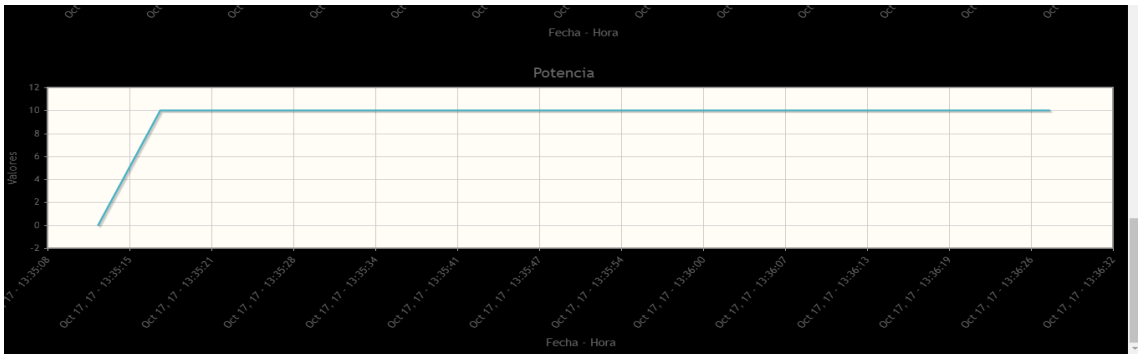


Figura 58. Señal de potencia del sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Usuarios

En la pestaña usuarios nos presenta un pequeño formulario el cual es llenado por la persona que esté utilizando el sistema en esos momentos.

Figura 59. Venta de usuarios del sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

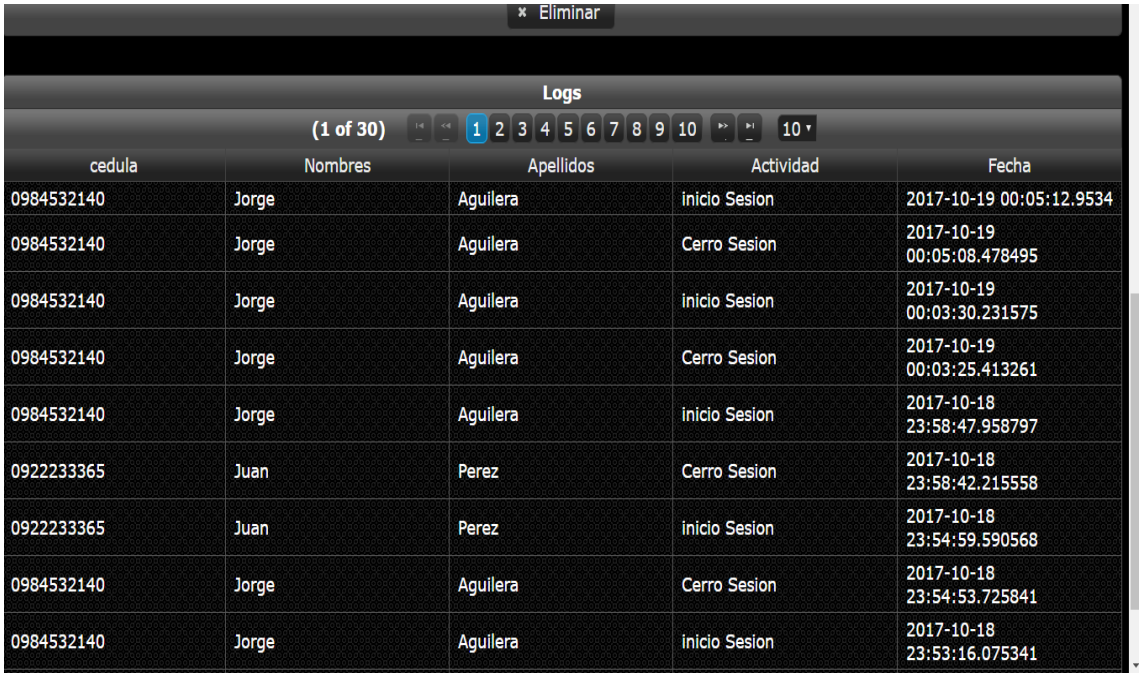


Figura 60. Listado de usuarios registrados

(L. Aguirre, C. Salcedo, J. Neira, 2018)

4 RESULTADOS

A continuación se detallan cada uno de los objetivos planteados al inicio:

4.1 Diseño de un sistema de calentamiento por inducción

Se logró cumplir la estructura del sistema diseñado cumpliendo con cada etapa del sistema.

- Etapa de fuerza
- Etapa de control
- Etapa de monitoreo Web
- Pruducto terminado

Se logró a realizar el diseño de cada uno de los elementos de fuerza y control para poder lograr el calentamiento por inducción los elementos que se utilizaron para realizar el calentamiento por inducción son los siguientes:

- Bobina de inducción
- Tablero eléctrico de control
- Arduino
- RaspBerrry PI 3
- Evaporador
- Recipiente de hexano

Figura 61. Sistema de calentamiento por inducción

(L. Aguirre, C. Salcedo, J. Neira, 2018)

4.2 Proceso de destilación

El proceso de destilación comienza de la siguiente manera se procede a colocar en el recipiente de acero inoxidable el líquido a destilar el cual en este caso es el hexano, la cantidad máxima que puede contener el recipiente de la materia prima es de 1 litro.

4.2.1 Inicio de cesión en el sistema

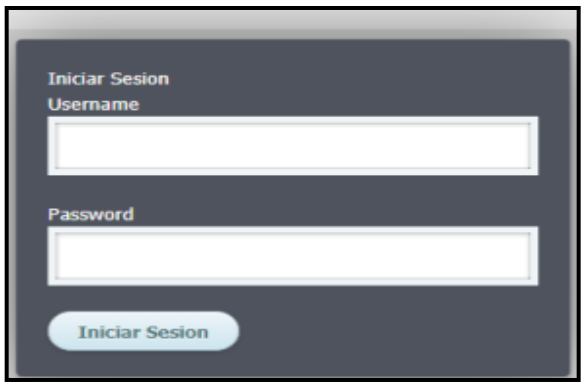


Figura 62. Inicio de sesión en el sistema

(L. Aguirre, C. Salcedo, J. Neira, 2018)

4.2.2 Validación de los parámetros iniciales

Se procede a conectar el voltaje de alimentación del sistema, una vez suministrada la energía se procede a encender el sistema web, donde se tendrán que validar los valores iniciales en los parámetros de configuración, los valores principales a verificar en el sistema web antes de iniciar el proceso de destilación son:

- Consigna de temperatura o Setpoint
- Duración del proceso de destilación
- Limites de temperaturas
- Nombre del Analista de laboratorio

Estos tres valores son los principales ya que, al colocar un valor de consigna incorrecto sucederán dos escenarios; en caso de colocar un valor de consigna por debajo del punto de ebullición no se conseguirá la destilación del mismo; y en caso de colocar un valor de consigna por encima del punto de ebullición, habrá pérdidas del material a destilar.

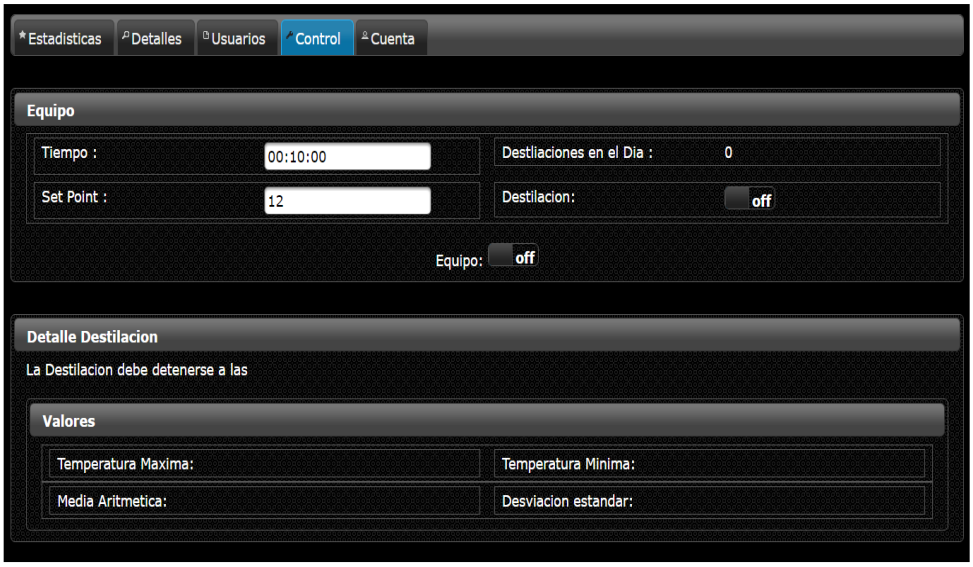


Figura 63. Establecimiento de parámetros inciales

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Si se establece un tiempo de destilación superior al tiempo medido en las pruebas, ocacionará que el sistema siga encendido pudiendo provocar daños en el recipiente y a su vez aumentará el consumo de energía eléctrica elevando los costos energéticos de los mismos.

En caso de colocar un límite de temperatura superior al establecido, el cual es un porcentaje de +/- 10% del punto de ebullición, ocasionará daños en el líquido a destilar y pudiendo ocasionar también daños en los equipos de control.

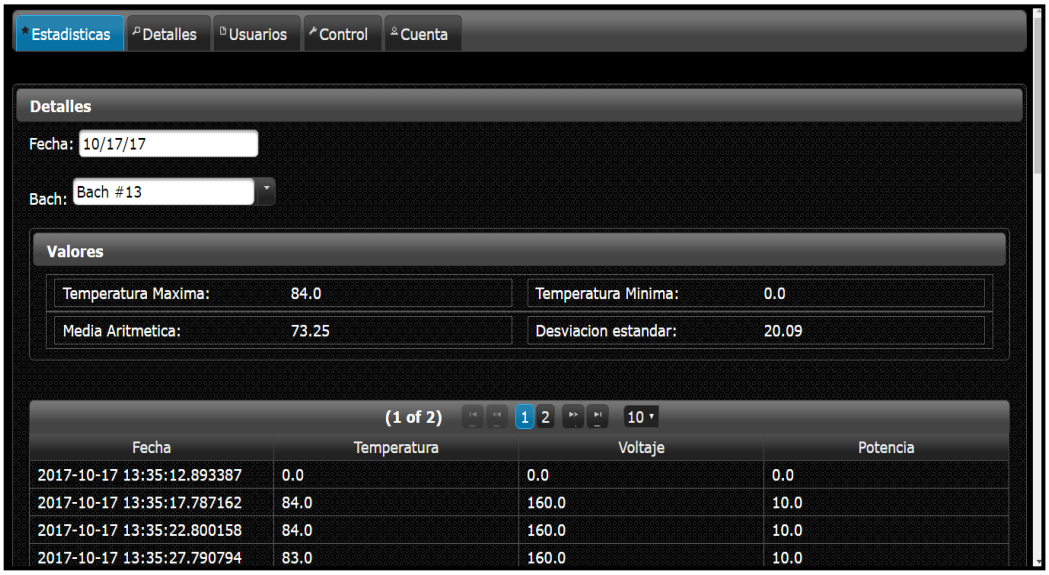


Figura 64. Parámetros de operación del sistema.

(L. Aguirre, C. Salcedo, J. Neira, 2018)

4.2.2 Inicio de destilación

El inicio del proceso se dará cuando el operador a través del sistema web de la orden de marcha al sistema, la tarjeta controladora Rapsberry PI 3 recibe esta orden de marcha del sistema y comienza a ejecutar los códigos de programación vistos en la sección 3.4.

Este proceso de destilación dura el tiempo establecido en el sistema web controlando que se cumplan todos los parámetros de configuración para garantizar la correcta destilación y seguridad de los equipos de control.

Se procede a realizar la obtención de datos del sistema una vez inciado el proceso de destlación, para lo cual se realizan pruebas de ensayo la cual podemos apreciar a continuación.

Para validar que el sistema funcionara de manera se procede a realizar la captura de datos desde el controlador Raspberry pi3 el cual nos da la facilidad de almacenar las variables en un archivo de texto, y con estos datos realizar las pruebas para afinar el controlador.

Se lograron capturar datos durante el proceso de destilación y se obtuvieron 32767 datos los cuales se aprecian en las gráficas acontinuación.

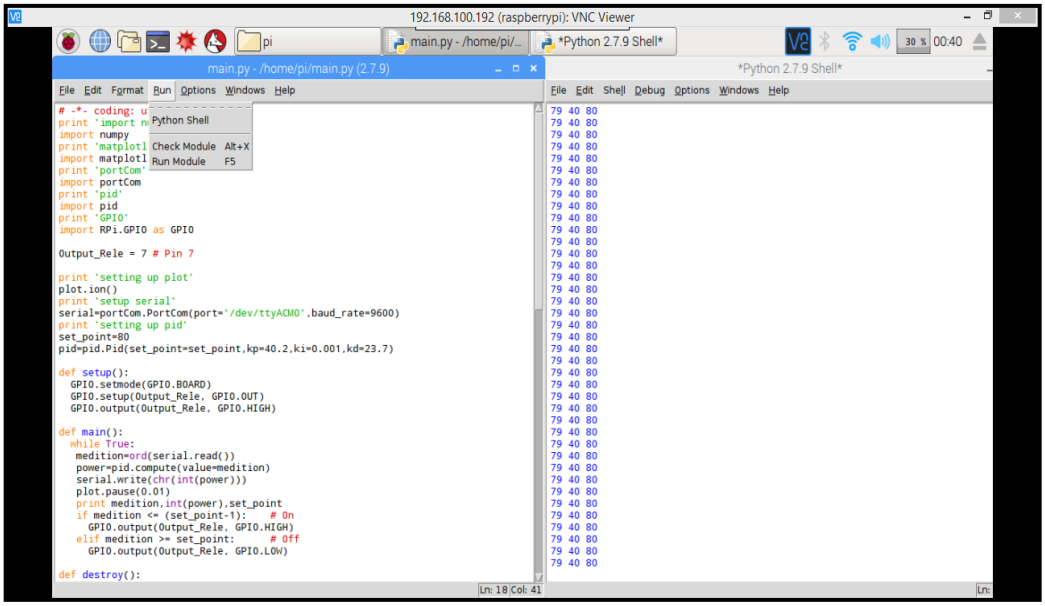


Figura 65. Datos en tiempo real del sistema obtenidos desde Raspberry pi3

(L. Aguirre, C. Salcedo, J. Neira, 2018)

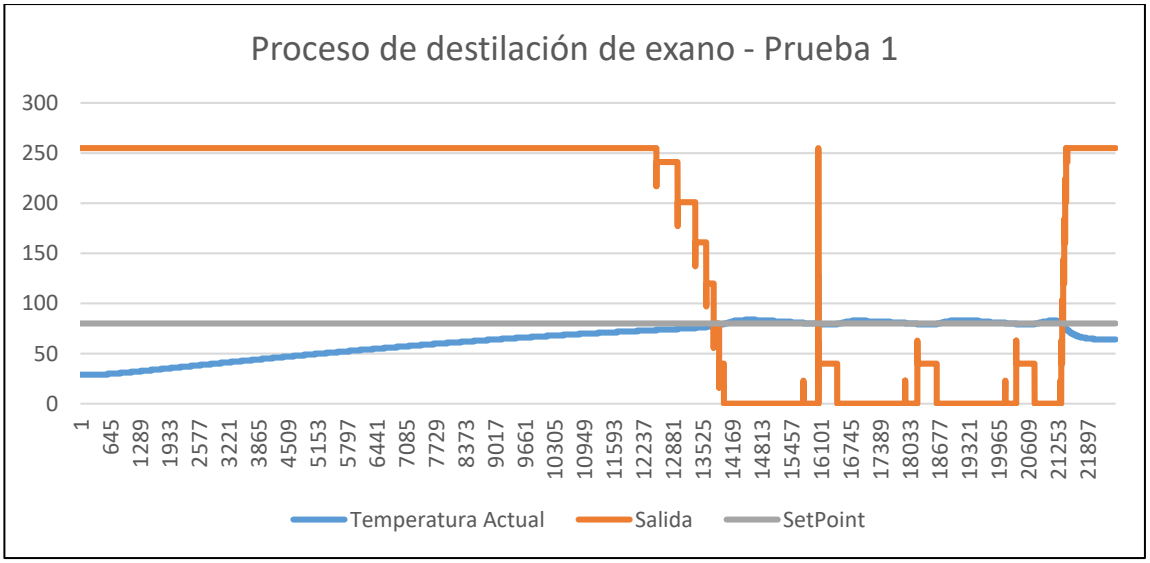


Figura 66. Comportamiento de señales del sistema prueba 1

(L. Aguirre, C. Salcedo, J. Neira, 2018)

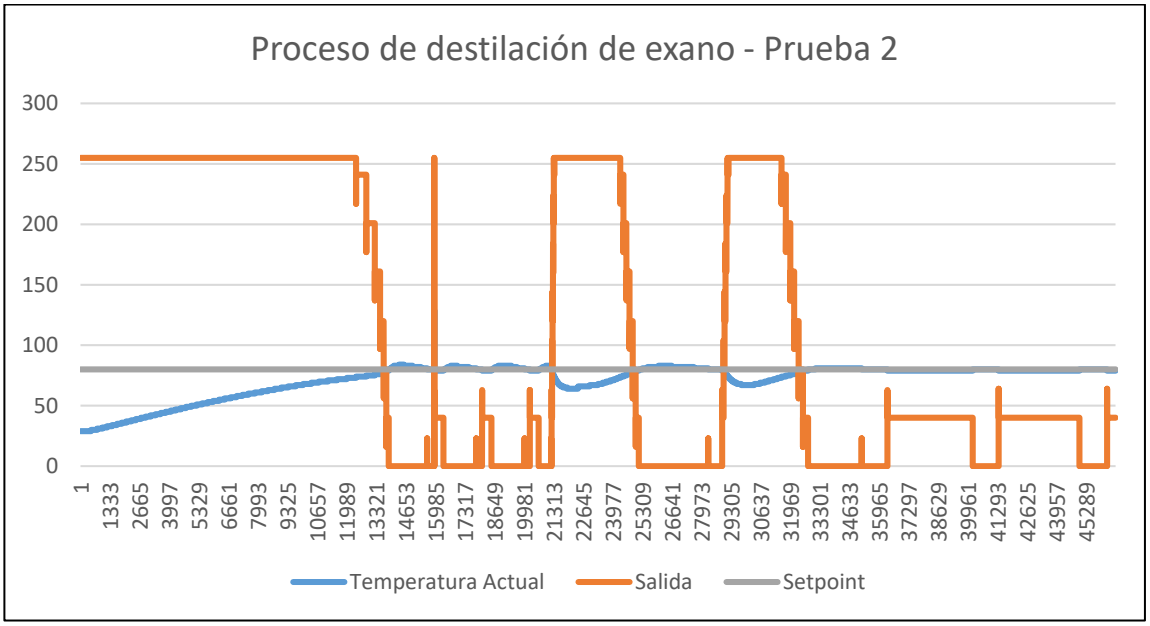


Figura 67. Comportamiento de señales del sistema prueba 2

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Luego de realizar las dos pruebas de ensayo y al notar que el sistema no se encuentra calibrado de una manera correcta se procede a colocar los valores P.I.D. encontrados en la sección 3.5.1 Cálculo de los parámetros P.I.D los cuales son colocados en el control realizado por la Raspberry pi3 y también colocados en la herramienta simulink del software Matlab.

4.2.3 Culminación del proceso

Luego de las pruebas realizadas y una vez ajustado todos los parámetros de control del sistema según los datos obtenidos una vez que terminó el proceso de destilación son los siguientes:
Con un muestro de de 32267 muestreos de variables de temperatura, setpoint y salida la gráfica del sistema nos queda de la siguiente forma.

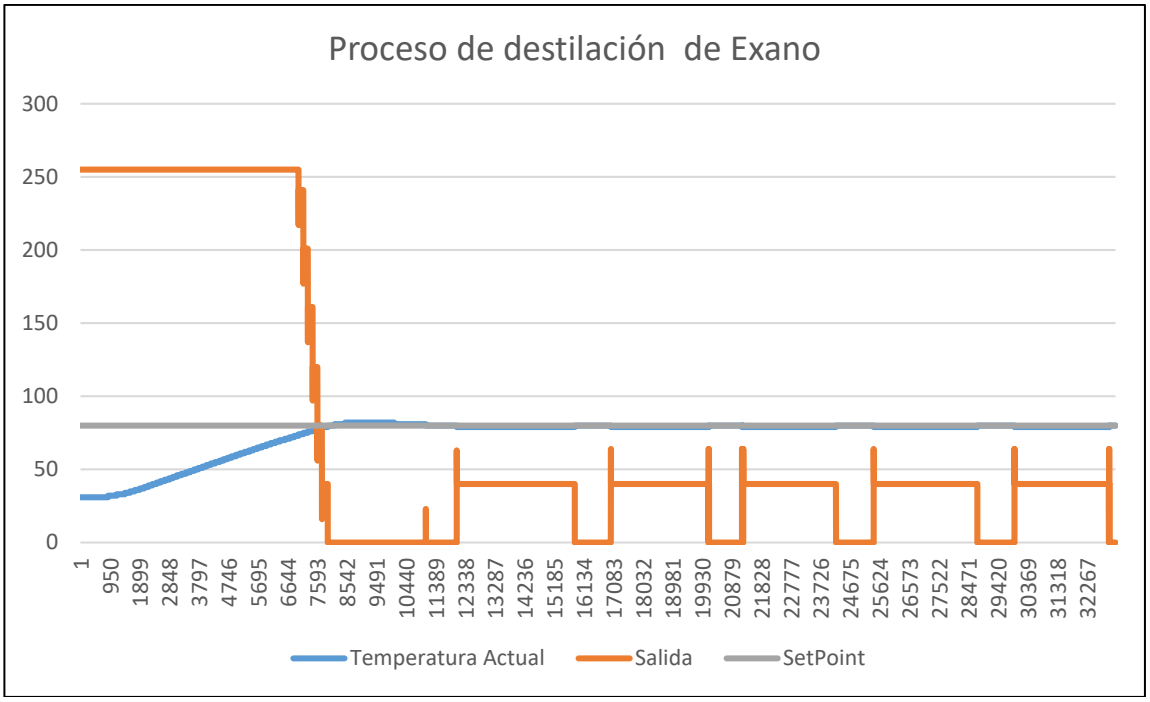


Figura 68. Comportamiento de señales del sistema final

(L. Aguirre, C. Salcedo, J. Neira, 2018)

Donde podemos apreciar que la señal de temperatura tiende al valor de consigna el cual es el punto de ebullición del líquido hexano, también observamos el comportamiento de la salida del sistema que se va acoplando hasta que el hexano alcance su punto de ebullición.

Luego de haber realizado la destilación el sistema procede a enviar un mensaje de culminación del proceso vía correo, además de esto durante el proceso de destilación en caso de ocurrir algun evento fuera de los rangos de protección este envía una alarma al sistema y es notificado vía correo.

Todos estos eventos son almacenados en una base de datos con el fin de que el operador pueda llevar un control de los procesos de destilación realizados durante el día, semana o mes, para poder ejecutar un plan de acción y llevar a cabo el cumplimiento del cronograma de actividades emitido por parte de Elicrom. Con esto se logró cumplir con todos los objetivos planteados para el desarrollo del proyecto.

CONCLUSIONES

Se logró diseñar un método de destilación cumpliendo con todas las seguridades debidas del líquido a destilar el cual es el hexano, ya que es un líquido que se usa en muchas ocasiones como combustible, pese a ellos se logró realizar el proceso de destilación con total seguridad.

Debido a la dificultada de encontrar un sensor de temperatura pt100 que coincida con el tamaño del destilador se procede a cambiar con un sensor ds1820 que se adapta al sistema tanto en tamaño y facilidad de usar sus datos.

El método de destilación de hexano diseñado supera la capacidad de destilación a un destilador convencional, un destilador convencional tiene una capacidad de destilación de 1 litro / hora, el método de destilación por inducción aplicado obtuvo una capacidad de destilación de 2 litros / hora.

Se logró realizar la destilación de 2 litros de hexano en 1 hora a 80°C el tiempo en que se estabiliza el sistema de 7 minutos.

Con la implementación del sistema web y HMI el personal que opere el destilador puede tener acceso a los datos de producción de la máquina, a histórico de alarmas y puede visualizar en tiempo real lo que está ocurriendo en el proceso.

Con el registro en la base de datos de todas las acciones que ocurren en el destilador se facilita el control de los batch realizados, ya que se evita la anotación manual de las anomalías presentes en el destilador y de las cantidades de líquido destiladas, facilitando así la visualización de estos registros tanto del operador como de las personas que tenga acceso a esta información.

Las ventajas del destilador frente a los destiladores convencionales se evidencian en muchos aspectos, ahora se cuenta con un tablero de control donde sus elementos y cables están debidamente etiquetados esto facilita la revisión en cualquier momento que se lo necesite, estas etiquetas son las mismas a las que se hace referencia en los planos eléctricos por ende el técnico tiene como guiarse y pude resolver cualquier falla en menor tiempo.

Para el desarrollo de la programación de los equipos de control y el sistema web se procedió a realizar los comentarios adecuados para poder realizar futuros cambios en caso de que se requiera.

La interfaz web desarrollada en Java es de muy fácil comprensión y legibilidad, para lo cual el operador podrá desarrollar los procesos de destilación de manera rápida y fácil.

RECOMENDACIONES

Revisar que el voltaje de alimentación para operar el destilador sea el adecuado el cual es de 220 Vac ya que al suministrar un voltaje por debajo del valor mencionado podría ocasionar daños en la bobina de inducción.

Realizar mantenimiento preventivo en la parte eléctrica, electrónica y mecánica cada 2 meses para evitar acumulaciones de polvo o desgastes en la máquina.

Antes de dar marcha al proceso es importante haber validado los parámetros de configuración, límites de alarma de temperatura, y consignas de ejecución del proceso para evitar daños en el líquido.

En caso de encontrar alguna anomalía en el sistema se recomienda revisar el manual de usuario del destilador para proceder a verificar el fallo.

Se recomienda que la persona que operará la máquina tenga los conocimientos mínimos requeridos para operar el destilador, ya que si se pasa por alto una de las recomendaciones podría ocasionar daños tanto a los equipos como al operador.

BIBLIOGRAFÍA

ABC Tecnología. (21 de julio de 2013). *Raspberry pi3*. Recuperado el 18 de agosto de 2017, de ABC: http://www.abc.es/tecnologia/informatica-hardware/20130716/abci-raspberry-como-201307151936.html

arduino.cl. (18 de Agosto de 2017). *Qué es Arduino*. Recuperado el 18 de agosto de 2017, de arduino.cl: http://arduino.cl/que-es-arduino/

Astrom, K. J. (2000). *PID Advance.* Alemania.

Burns, R. A. (2011). *Fundamento de Quimica.* Mexico: Pearson Educacion.

Digital Dimension. (Febrero de 2015). *MQTT: un protocolo específico para el internet de las cosas* . Obtenido de Digital Dimension: http://www.digitaldimension.solutions/es/blog-es/opinion-de-expertos/2015/02/mqtt-un-protocolo-especifico-para-el-internet-de-las-cosas/

ELECTRONICA EMBAJADORES, S.L. (19 de marzo de 2017). *ELECTRONICA EMBAJADORES*. Recuperado el 18 de agosto de 2017, de ELECTRONICA EMBAJADORES: https://www.electronicaembajadores.com/es/Productos/Detalle/LCA1008/modulos-electronicos/modulos-arduino/modulo-arduino-leonardo-r3

García, P. (5 de Septiembre de 2013). *Franklin Electric*. Obtenido de ¿Qué es control PID?: https://franklinelinkmx.wordpress.com/2013/09/05/que-es-el-control-pid/

GH Electrotermia . (2011). *Calentamiento por Inducción*. Obtenido de GH Electrotermia : http://www.ghinduction.com/about-induction-heating/

Hein, M., & Susan, A. (2001). *Fundamentos de Quimica.* Mexico D. F. 11560: THOMSON LEARNING.

Kurniawan, A. (2015). *Getting Started RaspBerry PI3.* EEUU.

Margolis, M. (2010). *Arduino Cookbook / Edition 2.* USA: O' Reilly.

Perez, R. (07 de 2015). *Placa Arduino Uno.* Obtenido de escuelatecnica.roqueperez.net: http://escuelatecnica.roqueperez.net/wp-content/uploads/2015/07/Placa-Arduino-Uno.pdf

Wilsin, J., j., B. A., & Bo, L. (2007). *Fisica.* Mexico: Pearson Educación.

Yébenez Gálvez , J. (2017). *¿Qué es MQTT?* Obtenido de Geeky Theory : https://geekytheory.com/que-es-mqtt

ANEXOS

ANEXO 1

ESTRUCTURA PRINCIPAL DE INICIO DE PROGRAMACIÓN DEL RASPBERRY

```
import thread
import time
import paho.mqtt.client as mqtt
from random import randint
import socket
import RPi.GPIO as GPIO

import numpy
import matplotlib.pyplot as plot
import portCom
import pid

def conectarMQTT():
   global client;
   client=mqtt.Client()
   client.connect("165.227.183.174",1883,60)

data = "0;0;0";
destilar = 1;
encendido = 0;
arduino=38;

def enviarDataMQTT():
   while True:
global client;
      global data;
      global destilar;
      global encendido;
if destilar==1 and encendido==1:
         client.publish("Tesis",data)
         time.sleep(5)
      else:
         destilar=0;
         break;


def iniciarLecturas(sp):
   global pid;
global arduino;
   plot.ion()
   serial=portCom.PortCom(port='/dev/ttyUSB0',baud_rate=9600)

   set_point=sp
   print "set point "
   print set_point
   pid=pid.Pid(set_point=set_point,kp=0.1,ki=0,kd=0)
```

```
try:
      thread.start_new_thread( enviarDataMQTT, ( ) )
      t=0
      while True:
         global data;
         global encendido;
         global destilar;
         if destilar==1 and encendido==1:
            medition=ord(serial.read())
            power=pid.compute(value=medition)
            if power < 20 :
               GPIO.output(arduino, GPIO.LOW)
            else:
               GPIO.output(arduino, GPIO.HIGH)
            serial.write(str(power))
            plot.pause(0.01)
temperatura=medition;
            voltaje=power;
            if voltaje <= 255 or voltaje >= 191:
               voltaje=160;
            elif voltaje <=191 or voltaje >=128:
               voltaje = 158;
            elif voltaje<=128:
               voltaje=152;
            potencia=randint(1, 800);
            if potencia <= 255 or potencia >= 191:
               potencia=10;
            elif potencia <=191 or potencia >=128:
               potencia=7.5
            elif potencia<=128:
               potencia=5
            data = str(temperatura) + ";"+ str(voltaje)+
";"+ str(potencia);
else:
            destilar=0;
            break;
   except Exception as e:
print(e)
      print "Error"
```

```
def metodoPrinicipal():
   global arduino;
   GPIO.setmode(GPIO.BOARD)
   GPIO.setup(arduino, GPIO.OUT)
   GPIO.output(arduino, GPIO.LOW)
TCP_IP = '165.227.183.174'
   TCP_PORT = 5005
   BUFFER_SIZE = 1024

   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   s.connect((TCP_IP, TCP_PORT))

   while True:
      global destilar;
      global encendido;
      msg = s.recv(BUFFER_SIZE)
      if not data: break
      s.send("n")
      if msg.startswith('A'):
         GPIO.output(arduino, GPIO.HIGH)
         conectarMQTT();
         a,n=msg.split(",")
         destilar = 1;
         thread.start_new_thread( iniciarLecturas,
(int(n), ) )
print "iniciar destilacion";
      elif msg == "B":
        GPIO.output(arduino, GPIO.LOW)
        destilar = 0;
        print "detener destilacion";
      elif msg == "C":
        conectarMQTT();
        GPIO.output(arduino, GPIO.LOW)
        encendido=1;
        print "encender Equipo";
      elif msg == "D":
        GPIO.output(arduino, GPIO.LOW)
        encendido=0;
        print "apagar Equipo";
   #s.close()
try:
   while True:
      try:
         conectarMQTT();
         metodoPrinicipal();

      except Exception:
         pass;
except Exception:
print "Error";
```

ANEXO 2

CONTROL PID

```
   # -*- coding: utf-8 -*-
class Pid:
    lastProportional=0
    integrative=0
    def
__init__(self,set_point,kp=0,ki=0,kd=0,max=255,min=0):
        self.set_point=set_point
        self.kp=kp
        self.ki=ki
        self.kd=kd
        self.max=max
        self.min=min
    def compute(self,value):
        proportional=self.set_point-value
        derivative=proportional-self.lastProportional
        self.integrative=self.integrative+proportional
        if self.integrative>self.max:
            self.integrative=self.max
        if self.integrative<self.min:
            self.integrative=self.min
        self.lastProportional=proportional


power=proportional*self.kp+self.integrative*self.ki+deriv
ative*self.kd
        if power>self.max:
            power=self.max
        if power<self.min:
            power=self.min
return power
```

ANEXO 3

LECTURA DE PUERTOS A UTILIZAR

```
# -*- coding: utf-8 -*-
import serial
class PortCom:
    def __init__(self,port,baud_rate):
        self.port=serial.Serial(port,baud_rate)
    def read(self):
        return self.port.read()
    def read_line(self):
        character=self.port.read()
        data=""
        while character!="\r":
            data=data+character
            character=self.port.read()
        self.port.read()
        return data
    def write(self,data):
self.port.write(data)
```

ANEXO 4

CONEXIÓN ARDUINO CON RASPBERRY

```
#define BAUD_RATE 9600
#define FROM_MIN 0
#define FROM_MAX 1023
#define TO_MIN 0
#define TO_MAX 254
#define PIN_OUT 10

int out=0;
void setup()
{
   Serial.begin(BAUD_RATE);
   pinMode(PIN_OUT,OUTPUT);
}
void loop()
{
   out=analogRead(A0);
   out=map(out,FROM_MIN,FROM_MAX,TO_MIN,TO_MAX);
   Serial.write(out);
   if(Serial.available())
   {
      String in="";
      while(Serial.available())
      {
         in+=(char)Serial.read();
                                                    }
                                          analogWrite(PIN_OUT,in.toInt());
                                                    }
                                               delay(20);
                                                    }
```

ANEXO 5

CONEXIÓN ARDUINO MATLAB

```
#define BAUD_RATE 9600

#define FROM_MIN 0
#define FROM_MAX 1023
#define TO_MIN 0
#define TO_MAX 255

#define PIN_OUT 10

int out=0;
byte in=0;

void setup()
{
   Serial.begin(BAUD_RATE);
   pinMode(PIN_OUT,OUTPUT);
}

void loop()
{
   out=analogRead(A0);
   out=map(out,FROM_MIN,FROM_MAX,TO_MIN,TO_MAX);
   Serial.write(out);
   if(Serial.available())
   {
      in=Serial.read();
      analogWrite(PIN_OUT,in);
}
   delay(20);
}
```

ANEXO 6
LOGIN DE LA PAGINA WEB EN JAVA

```
package com.tesis.jsf.beans;

import java.io.Serializable;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;
import javax.servlet.http.HttpSession;

import com.tesis.jsf.dao.LoginDAO;
import com.tesis.jsf.dao.LogsDAO;
import com.tesis.jsf.dao.UserDAO;
import com.tesis.jsf.util.SessionUtils;
import javax.annotation.PostConstruct;

import org.primefaces.model.chart.AxisType;
import org.primefaces.model.chart.LineChartModel;
import org.primefaces.model.chart.DateAxis;
import org.primefaces.model.chart.LineChartSeries;

@ManagedBean
@SessionScoped
public class Login implements Serializable {

   private static final long serialVersionUID = 1094801825228386363L;

   private String pwd;
   private String msg;
   private String user;




   public String getPwd() {
      return pwd;
   }

   public void setPwd(String pwd) {
      this.pwd = pwd;
   }

   public String getMsg() {
      return msg;
   }

   public void setMsg(String msg) {
```

```
        this.msg = msg;
    }

    public String getUser() {
        return user;
    }

    public void setUser(String user) {
        this.user = user;
    }

    void insertarLog(String detalle) {
        HttpSession session = SessionUtils.getSession();
        String userActual = session.getAttribute("username").toString();

String idUsuario = UserDAO.obtenerIdUsuario(userActual);

        LogsDAO.insertarLog(idUsuario, detalle);

    }

    //validate login
public String validateUsernamePassword() {
        boolean valid = LoginDAO.validate(user, pwd);
        if (valid) {
            HttpSession session = SessionUtils.getSession();
            session.setAttribute("username", user);
            insertarLog("inicio Sesion");
            return "control";
        } else {
            FacesContext.getCurrentInstance().addMessage(
                    null,
                    new FacesMessage(FacesMessage.SEVERITY_WARN,
                            "Incorrect Username and Passowrd",
                            "Please enter correct username and Password"));
            return "login";
        }
    }

    //logout event, invalidate session
    public String logout() {
        insertarLog("Cerro Sesion");
        HttpSession session = SessionUtils.getSession();
session.invalidate();

        return "login";
    }
```

ANEXO 7
PESTAÑA USUARIO

```
package com.tesis.jsf.beans;

import com.tesis.jsf.dao.LogsDAO;
import com.tesis.jsf.dao.UserDAO;
import com.tesis.jsf.entities.Log;
import com.tesis.jsf.entities.Operador;
import com.tesis.jsf.util.SessionUtils;
import java.io.Serializable;
import java.util.List;
import javax.annotation.PostConstruct;
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;
import javax.servlet.http.HttpSession;
import org.primefaces.context.RequestContext;
import org.primefaces.event.SelectEvent;
import org.primefaces.event.UnselectEvent;

@ManagedBean(name = "user")
@SessionScoped
public class Usuarios implements Serializable {

    private String name;
    private String lastname;
    private String email;
    private String phone;
    private String nickname;
    private String password;
    private String cid;
    private String rol;
    private boolean editarCedula = false;
    private List<Log> logs;

    @PostConstruct
    public void init() {

setOperadores(UserDAO.obtenerOperadores());
        setLogs(LogsDAO.obtenerLogs());

}

    public void carga() {

        try {
```

```
            HttpSession session = SessionUtils.getSession();
            String userActual = session.getAttribute("username").toString();

            String rol = UserDAO.obtenerRolUsuario(userActual);
            if (rol.equalsIgnoreCase("A")) {

            } else {
                session = SessionUtils.getSession();
                session.invalidate();
                FacesContext.getCurrentInstance().getExternalContext().redirect("/error");
            }

        } catch (Exception e) {
        }
    }

    public void buttonAction(ActionEvent actionEvent) {
        FacesContext context = FacesContext.getCurrentInstance();

        if (UserDAO.comprobarExistencia(cid)) {
            if (UserDAO.comprobarExistenciaUusuario(nickname)) {
                context.addMessage(null, new FacesMessage("Error", "El nickname ya esta
en uso!!"));
            } else if (UserDAO.actualizarOperador(cid, rol, name, lastname, email, phone,
nickname, password)) {
                context.addMessage(null, new FacesMessage("Proceso", "Correcto!!"));
            }

        } else if (UserDAO.comprobarExistenciaUusuario(nickname)) {
            context.addMessage(null, new FacesMessage("Error", "El nickname ya esta en
uso!!"));
        } else if (UserDAO.insertarUsuario(cid, rol, name, lastname, email, phone,
nickname, password)) {
            context.addMessage(null, new FacesMessage("Proceso", "Correcto!!"));

        } else {
            context.addMessage(null, new FacesMessage("Error", "Errroooor!!"));
}

        setOperadores(UserDAO.obtenerOperadores());
        RequestContext contextx = RequestContext.getCurrentInstance();
contextx.update("formTablaUsuarios:singleDT");

    }

    public void eliminarUser(ActionEvent actionEvent) {
        FacesContext context = FacesContext.getCurrentInstance();
        context.addMessage(null, new FacesMessage("Proceso", "Correcto!!"));
        System.out.println("Clickk eliminar XD");
        HttpSession session = SessionUtils.getSession();
```

```
        String userActual = session.getAttribute("username").toString();
if (!userActual.equals(getOperadorSeleccionado().getNickname())) {
            if (UserDAO.eliminarUsuario(getOperadorSeleccionado().getIdOperador())) {
                System.out.println("Se elimino el usuario");
                setOperadores(UserDAO.obtenerOperadores());
                RequestContext contextx = RequestContext.getCurrentInstance();
contextx.update("formTablaUsuarios:singleDT");
                //
FacesContext.getCurrentInstance().getPartialViewContext().getRenderIds().add("formT
ablaUsuarios:singleDT");

} else {
                System.out.println("No se pudo eliminar");
}
        }

    }

    /**
     * @return the name
     */
    public String getName() {
        return name;
    }

    /**
     * @param name the name to set
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * @return the lastname
     */
    public String getLastname() {
        return lastname;
    }

    /**
     * @param lastname the lastname to set
     */
    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    /**
     * @return the email
     */
    public String getEmail() {
        return email;
```

```
}

/**
 * @param email the email to set
 */
public void setEmail(String email) {
    this.email = email;
}

/**
 * @return the phone
 */
public String getPhone() {
    return phone;
}

/**
 * @param phone the phone to set
 */
public void setPhone(String phone) {
    this.phone = phone;
}

/**
 * @return the nickname
 */
public String getNickname() {
    return nickname;
}

/**
 * @param nickname the nickname to set
 */
public void setNickname(String nickname) {
    this.nickname = nickname;
}

/**
 * @return the password
 */
public String getPassword() {
    return password;
}

/**
 * @param password the password to set
 */
public void setPassword(String password) {
    this.password = password;
}
```

```
    /**
     * @return the cid
     */
    public String getCid() {
        return cid;
    }

    /**
     * @param cid the cid to set
     */
    public void setCid(String cid) {
        this.cid = cid;
    }

    /**
     * @return the rol
     */
    public String getRol() {
        return rol;
    }

    /**
     * @param rol the rol to set
     */
    public void setRol(String rol) {
this.rol = rol;
    }

    private List<Operador> operadores;

    private Operador operadorSeleccionado;

public void onRowSelect(SelectEvent event) {
        System.out.println("Se selecciono " + ((Operador) event.getObject()).getCedula());
        setOperadorSeleccionado(((Operador) event.getObject()));

        setName(((Operador) event.getObject()).getNombres());
        setLastname(((Operador) event.getObject()).getApellidos());
        setEmail(((Operador) event.getObject()).getCorreos());
        setPhone(((Operador) event.getObject()).getTelefono());
        setNickname(((Operador) event.getObject()).getNickname());
        setPassword(((Operador) event.getObject()).getPassword());
        setCid(((Operador) event.getObject()).getCedula());

        RequestContext contextx = RequestContext.getCurrentInstance();
        contextx.update("formulario:name");
        contextx.update("formulario:apellidos");
        contextx.update("formulario:email");
        contextx.update("formulario:phone");
        contextx.update("formulario:nickname");
        contextx.update("formulario:password");
```

```
            contextx.update("formulario:regex");

        }

        /**
         * @return the operadores
         */
        public List<Operador> getOperadores() {
            return operadores;
        }

        /**
         * @param operadores the operadores to set
    */
        public void setOperadores(List<Operador> operadores) {
            this.operadores = operadores;
        }

        /**
         * @return the operadorSeleccionado
         */
        public Operador getOperadorSeleccionado() {
            return operadorSeleccionado;
        }

        /**
         * @param operadorSeleccionado the operadorSeleccionado to set
         */
        public void setOperadorSeleccionado(Operador operadorSeleccionado) {
            this.operadorSeleccionado = operadorSeleccionado;
        }

        /**
         * @return the editarCedula
    */
        public boolean isEditarCedula() {
            return editarCedula;
        }

        /**
         * @param editarCedula the editarCedula to set
         */
        public void setEditarCedula(boolean editarCedula) {
            this.editarCedula = editarCedula;
        }

        /**
         * @return the logs
         */
        public List<Log> getLogs() {
            return logs;
```

```
    }

    /**
     * @param logs the logs to set
     */
    public void setLogs(List<Log> logs) {
this.logs = logs;
    }

}
```

ANEXO 8

PESTAÑA CONTROL

```
package com.tesis.jsf.beans;

import com.tesis.jsf.dao.BachDao;
import com.tesis.jsf.dao.EquipoDAO;
import com.tesis.jsf.dao.LogsDAO;
import com.tesis.jsf.dao.RegistrosDAO;
import com.tesis.jsf.dao.UserDAO;
import com.tesis.jsf.entities.Registro;

import com.tesis.jsf.util.SessionUtils;

import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.Date;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;

import javax.annotation.PostConstruct;
import javax.servlet.http.HttpSession;

import org.primefaces.event.SelectEvent;

import java.io.*;
import java.net.*;
import java.util.Calendar;
import java.util.List;

@ManagedBean(name = "control")
@SessionScoped
public class Control implements Serializable {

    private String tiempo;
    private String sep;
    private String totalDestilaciones;
    private boolean estadoEquipo;
    private boolean destilando;
    private String tempMaxima;
    private String temMinima;
    private String mediaAr;
    private String desviacion;
    private boolean rolAdmin;
    private String horade;
private List<Registro> registros;
```

```
    @PostConstruct
    public void init() {
        setEstadoEquipo(EquipoDAO.obtenerEstadoEquipo());
        setDestilando(EquipoDAO.obtenerEstadoDestilacion());
        System.out.println("Antes de tener numero destilaciones");

setTotalDestilaciones(String.valueOf(EquipoDAO.obtenerNumeroDestilaciones(getCurr
entDate())));
        setRolAdmin(verificarAdmin());

    }

    public void carga() {

        try {

setTotalDestilaciones(String.valueOf(EquipoDAO.obtenerNumeroDestilaciones(getCurr
entDate())));
            Object[] datos = EquipoDAO.obtenerIdBach();

            if (datos != null) {
int id = (int) datos[0];
                String hora = String.valueOf(datos[1]);
                String tiempo = String.valueOf(datos[2]);
                SimpleDateFormat df = new SimpleDateFormat("HH:mm:ss");
Date d = df.parse(hora);
Calendar cal = Calendar.getInstance();
                cal.setTime(d);
                String timpos[] = tiempo.split(":");
                cal.add(Calendar.HOUR, Integer.parseInt(timpos[0]));
                cal.add(Calendar.MINUTE, Integer.parseInt(timpos[1]));
                cal.add(Calendar.SECOND, Integer.parseInt(timpos[2]));
                String tiempoTotal = df.format(cal.getTime());
setHorade(tiempoTotal);
                setRegistros(RegistrosDAO.obtenerRegistrosPorIdBach(id));
                setTempMaxima(RegistrosDAO.obtenerTemperaturaMaximaPorIdBach(id));
                setTemMinima(RegistrosDAO.obtenerTemperaturaMinimaPorIdBach(id));
                setMediaAr(RegistrosDAO.obtenerMediaAritmetica(registros));
                setDesviacion(RegistrosDAO.obtenerDesvia(registros));
} else {
                setHorade("");

                setTempMaxima("");
                setTemMinima("");
                setMediaAr("");
                setDesviacion("");

            }

        } catch (Exception e) {
        }
```

```
    }

    boolean verificarAdmin() {
        HttpSession session = SessionUtils.getSession();
        String userActual = session.getAttribute("username").toString();

        String rol = UserDAO.obtenerRolUsuario(userActual);
        if (rol.equalsIgnoreCase("A")) {
            return true;
        } else {
            return false;
        }

    }

    void enviarUDP(String dato) {
        try {

            DatagramSocket clientSocket = new DatagramSocket();
            InetAddress IPAddress = InetAddress.getByName("localhost");
            byte[] sendData = new byte[1024];
            sendData = dato.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 9876);
            clientSocket.send(sendPacket);
            clientSocket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public String getCurrentDate() {
        return new SimpleDateFormat("yyyy-MM-dd").format(new Date());
    }

    public void onDateSelect(SelectEvent event) {
        FacesContext facesContext = FacesContext.getCurrentInstance();
        SimpleDateFormat format = new SimpleDateFormat("HH:mm:ss");
        String fecha = format.format(event.getObject());
System.out.println(fecha);
        tiempo = fecha;

    }

    public void encenderEquipo() {

        EquipoDAO.actualizarEstado(estadoEquipo);
        setEstadoEquipo(EquipoDAO.obtenerEstadoEquipo());

        if (estadoEquipo) {
            insertarLog("encendio Equipo");
```

```
            enviarUDP("C");

        } else {
            insertarLog("Apago Equipo");
            enviarUDP("D");
}
    }

    void insertarLog(String detalle) {
        HttpSession session = SessionUtils.getSession();
        String userActual = session.getAttribute("username").toString();

String idUsuario = UserDAO.obtenerIdUsuario(userActual);

        LogsDAO.insertarLog(idUsuario, detalle);

    }

    public void destilar() {
System.out.println(destilando);
        if (!destilando) {
Object[] datos = EquipoDAO.obtenerIdBach();
            EquipoDAO.actualizarDestilacion("B");
            setDestilando(EquipoDAO.obtenerEstadoDestilacion());
            enviarUDP("B");
            insertarLog("Detuvo Destilacion");

            if (datos != null) {
int id = (int) datos[0];
                String hora = String.valueOf(datos[1]);
                String tiempo = String.valueOf(datos[2]);
                String numDestilaciones =
String.valueOf(EquipoDAO.obtenerNumeroDestilaciones(getCurrentDate()));
                String tiempoAhora = new SimpleDateFormat("HH:mm:ss").format(new
Date());

//  idDestilacion , numeroDestilacion, Hora , tiempo
                enviarUDP(id + ";" + numDestilaciones + ";" + hora + ";" + tiempo + ";" +
tiempoAhora);
            }

        } else {
            EquipoDAO.actualizarDestilacion("B");

            BachDao.insertarBach(sep, tiempo);
            boolean b = EquipoDAO.obtenerEstadoDestilacion();
            setDestilando(b);
            if (b) {
                enviarUDP("A," + sep.trim());
                insertarLog("Inicio Destilacion");
            }
```

```
            }
            // EquipoDAO.actualizarEstado(estadoEquipo);
            System.out.println("se empieza a destilar XD");

        }

        /**
         * @return the totalDestilaciones
         */
        public String getTotalDestilaciones() {
            return totalDestilaciones;
        }

        /**
         * @param totalDestilaciones the totalDestilaciones to set
         */
        public void setTotalDestilaciones(String totalDestilaciones) {
            this.totalDestilaciones = totalDestilaciones;
        }

        /**
         * @return the estadoEquipo
         */
        public boolean isEstadoEquipo() {
            return estadoEquipo;
        }

        /**
         * @param estadoEquipo the estadoEquipo to set
         */
        public void setEstadoEquipo(boolean estadoEquipo) {
            this.estadoEquipo = estadoEquipo;
    }

        /**
         * @return the destilando
         */
        public boolean isDestilando() {
            return destilando;
        }

        /**
         * @param destilando the destilando to set
         */
        public void setDestilando(boolean destilando) {
            this.destilando = destilando;
        }

        /**
         * @return the sep
         */
```

```
public String getSep() {
    return sep;
}

/**
 * @param sep the sep to set
 */
public void setSep(String sep) {
    this.sep = sep;
}

/**
 * @return the tempMaxima
 */
public String getTempMaxima() {
    return tempMaxima;
}

/**
 * @param tempMaxima the tempMaxima to set
 */
public void setTempMaxima(String tempMaxima) {
    this.tempMaxima = tempMaxima;
}

/**
 * @return the temMinima
 */
public String getTemMinima() {
    return temMinima;
}

/**
 * @param temMinima the temMinima to set
 */
public void setTemMinima(String temMinima) {
    this.temMinima = temMinima;
}

/**
 * @return the mediaAr
 */
public String getMediaAr() {
    return mediaAr;
}

/**
 * @param mediaAr the mediaAr to set
 */
public void setMediaAr(String mediaAr) {
    this.mediaAr = mediaAr;
```

```
    }

    /**
     * @return the desviacion
     */
    public String getDesviacion() {
        return desviacion;
    }

    /**
     * @param desviacion the desviacion to set
     */
    public void setDesviacion(String desviacion) {
this.desviacion = desviacion;
    }

    /**
     * @return the rolAdmin
*/
    public boolean isRolAdmin() {
        return rolAdmin;
    }

    /**
     * @param rolAdmin the rolAdmin to set
     */
    public void setRolAdmin(boolean rolAdmin) {
this.rolAdmin = rolAdmin;
    }

    /**
     * @return the registros
     */
    public List<Registro> getRegistros() {
        return registros;
    }

    /**
     * @param registros the registros to set
     */
    public void setRegistros(List<Registro> registros) {
        this.registros = registros;
    }

    /**
     * @return the horade
     */
    public String getHorade() {
        return horade;
    }
```

```
    /**
     * @param horade the horade to set
     */
    public void setHorade(String horade) {
this.horade = horade;
    }

}
```

ANEXO 9

PESTAÑA ESTADISTICA

```
package com.tesis.jsf.beans;

import com.tesis.jsf.dao.BachDao;
import com.tesis.jsf.dao.RegistrosDAO;
import com.tesis.jsf.dao.UserDAO;
import com.tesis.jsf.entities.Registro;
import com.tesis.jsf.util.SessionUtils;
import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import javax.annotation.PostConstruct;
import javax.ejb.Stateless;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import org.primefaces.model.chart.AxisType;
import org.primefaces.model.chart.DateAxis;
import org.primefaces.model.chart.LineChartModel;
import org.primefaces.model.chart.LineChartSeries;
import java.util.List;
import javax.faces.bean.ManagedProperty;
import javax.faces.context.FacesContext;
import org.primefaces.event.SelectEvent;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import javax.servlet.http.HttpSession;

@ManagedBean(name = "est")
@SessionScoped
public class Estadisticas implements Serializable {

    private String idBach;
private List<Registro> registros;
    private Date fechaSeleccionada;
private Map<String, String> baches;
    private String tempMaxima;
    private String temMinima;
    private String mediaAr;
    private String desviacion;
    private boolean rolAdmin;
    private LineChartModel dateModel;
    private LineChartModel dateModelVoltaje;
    private LineChartModel dateModelPotencia;
```

```
    @PostConstruct
    public void init() {
        createDateModel();

        baches = new HashMap<String, String>();
         try {
             setRolAdmin(verificarAdmin());
        } catch (Exception e) {
        }

    }

      boolean verificarAdmin() {
        HttpSession session = SessionUtils.getSession();
        String userActual = session.getAttribute("username").toString();

        String rol = UserDAO.obtenerRolUsuario(userActual);
        if(rol.equalsIgnoreCase("A")){
           return true;
        }else {
            return false;
        }

    }

    void llenarBaches(String fecha) {
        baches = BachDao.obtenerBaches(fecha);
        System.out.println("se llenaron xd");

    }

    public void eventoCombo() {
        System.out.println("Evento XD");
System.out.println(getIdBach());
        setRegistros(RegistrosDAO.obtenerRegistros(getIdBach()));
        setMediaAr(RegistrosDAO.obtenerMediaAritmetica(registros));
        setDesviacion(RegistrosDAO.obtenerDesvia(registros));
        setTempMaxima(RegistrosDAO.obtenerTemMaxBach(getIdBach()));
        setTemMinima(RegistrosDAO.obtenerTemMinBach(getIdBach()));
if (getRegistros() == null) {
            createDateModel();
        } else if (getRegistros().size() < 1) {
            createDateModel();
        } else {
            graficar();
        }

    }

    public void onDateSelect(SelectEvent event) {
        FacesContext facesContext = FacesContext.getCurrentInstance();
```

```
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        String fecha = format.format(event.getObject());

        llenarBaches(fecha);
        List<Registro> list = new ArrayList<Registro>();
        setRegistros(list);
        createDateModel();
        setTemMinima(null);
        setTempMaxima(null);
        setMediaAr(null);
        setDesviacion(null);

    }



    private void graficar() {
        System.out.println("entre a graf");
        setDateModel(new LineChartModel());
        setDateModelVoltaje(new LineChartModel());
        setDateModelPotencia(new LineChartModel());

        LineChartSeries series1 = new LineChartSeries();
        series1.setLabel("Temperatura");
        series1.setShowMarker(false);
        //  series1.set("2017-09-22 07:02:45.861427", 50);
        // series1.set("2017-09-22 24:03:09.948609", 22);
        LineChartSeries series2 = new LineChartSeries();

        series2.setLabel("Voltaje");
    series2.setShowMarker(false);
        LineChartSeries series3 = new LineChartSeries();
        series3.setLabel("Potencia");
        series3.setShowMarker(false);
        System.out.println("antes del for");
        for (int i = 0; i < registros.size(); i++) {
            System.out.println(registros.get(i).getFecha() + "fecha ");
            series1.set(registros.get(i).getFecha(), registros.get(i).getTemperatura());
            series2.set(registros.get(i).getFecha(), registros.get(i).getVoltaje());
            series3.set(registros.get(i).getFecha(), registros.get(i).getPotencia());
        }

        getDateModel().addSeries(series1);
        getDateModelVoltaje().addSeries(series2);
        getDateModelPotencia().addSeries(series3);

        getDateModel().setTitle("Temperatura");
        getDateModel().setZoom(true);

        getDateModelVoltaje().setTitle("Voltaje");
        getDateModelVoltaje().setZoom(true);
```

```
    getDateModelPotencia().setTitle("Potencia");
    getDateModelPotencia().setZoom(true);
    //  dateModel.setShowPointLabels(true);
    getDateModel().getAxis(AxisType.Y).setLabel("Valores");
    getDateModelVoltaje().getAxis(AxisType.Y).setLabel("Valores");
    getDateModelPotencia().getAxis(AxisType.Y).setLabel("Valores");
    DateAxis axis = new DateAxis("Fecha - Hora");
    axis.setTickAngle(-50);
    //  axis.setMax("2017-09-22 19:03:09.948609");
    String max = RegistrosDAO.obtenerMaximo(getIdBach());
    System.out.println("MAx " + max);
    axis.setMax(max);
    axis.setTickFormat("%b %#d, %y - %H:%#M:%S");

    getDateModel().getAxes().put(AxisType.X, axis);
    getDateModelVoltaje().getAxes().put(AxisType.X, axis);
    getDateModelPotencia().getAxes().put(AxisType.X, axis);
}

private void createDateModel() {
    setDateModel(new LineChartModel());
    setDateModelVoltaje(new LineChartModel());
    setDateModelPotencia(new LineChartModel());

    LineChartSeries series1 = new LineChartSeries();
    series1.setLabel("Series 1");

    series1.set("2017-09-22 07:02:45", 60);
    series1.set("2017-09-22 24:03:09", 80);
    /* series1.set("2014-01-01", 51);
    series1.set("2014-01-06", 22);
    series1.set("2014-01-12", 65);
    series1.set("2014-01-18", 74);
    series1.set("2014-01-24", 24);
    series1.set("2014-01-30", 51);*/

    LineChartSeries series2 = new LineChartSeries();
    series2.setLabel("Series 2");

    series2.set("2017-09-22 18:02:45", 20);
    series2.set("2017-09-22 18:03:09", 50);
    /*  series2.set("2014-01-12", 24);
    series2.set("2014-01-18", 12);
    series2.set("2014-01-24", 74);
    series2.set("2014-01-30", 62);*/

    LineChartSeries series3 = new LineChartSeries();
    series3.setLabel("Series 3");

    series3.set("2017-09-22 18:02:45", 80);
```

```
        series3.set("2017-09-22 18:03:09", 10);

        getDateModel().addSeries(series1);
        getDateModelVoltaje().addSeries(series2);
        getDateModelPotencia().addSeries(series3);
      //  getDateModel().addSeries(series2);
        //getDateModel().addSeries(series3);

        getDateModel().setTitle("Temperatura");
        getDateModel().setZoom(true);
        getDateModelVoltaje().setTitle("Voltaje");
        getDateModelVoltaje().setZoom(true);
        getDateModelPotencia().setTitle("Potencia");
        getDateModelPotencia().setZoom(true);
        //  dateModel.setShowPointLabels(true);
        getDateModel().getAxis(AxisType.Y).setLabel("Valores");
         getDateModelVoltaje().getAxis(AxisType.Y).setLabel("Valores");
          getDateModelPotencia().getAxis(AxisType.Y).setLabel("Valores");
        DateAxis axis = new DateAxis("Fecha - Hora");
        axis.setTickAngle(-50);
        axis.setMax("2017-09-26");
        axis.setTickFormat("%b %#d, %y - %H:%#M:%S");

        getDateModel().getAxes().put(AxisType.X, axis);
        getDateModelVoltaje().getAxes().put(AxisType.X, axis);
        getDateModelPotencia().getAxes().put(AxisType.X, axis);
    }

    /**
     * @return the dateModel
     */
    public LineChartModel getDateModel() {
        return dateModel;
    }

    /**
     * @param dateModel the dateModel to set
     */
    public void setDateModel(LineChartModel dateModel) {
        this.dateModel = dateModel;
    }

    /**
     * @return the registros
     */
    public List<Registro> getRegistros() {
        return registros;
    }

    /**
     * @param registros the registros to set
```

```
*/
    public void setRegistros(List<Registro> registros) {
        this.registros = registros;
    }

    /**
     * @return the fechaSeleccionada
     */
    public Date getFechaSeleccionada() {
        return fechaSeleccionada;
    }

    /**
     * @param fechaSeleccionada the fechaSeleccionada to set
     */
    public void setFechaSeleccionada(Date fechaSeleccionada) {
        this.fechaSeleccionada = fechaSeleccionada;
    }

    /**
     * @return the idBach
*/
    public String getIdBach() {
        return idBach;
    }

    /**
     * @param idBach the idBach to set
     */
    public void setIdBach(String idBach) {
        this.idBach = idBach;
    }

    /**
     * @return the baches
     */
    public Map<String, String> getBaches() {
        return baches;
    }

    /**
     * @param baches the baches to set
     */
    public void setBaches(Map<String, String> baches) {
        this.baches = baches;
    }

    /**
     * @return the tempMaxima
     */
    public String getTempMaxima() {
```

```
        return tempMaxima;
    }

    /**
     * @param tempMaxima the tempMaxima to set
     */
    public void setTempMaxima(String tempMaxima) {
        this.tempMaxima = tempMaxima;
    }

    /**
     * @return the temMinima
     */
    public String getTemMinima() {
        return temMinima;
    }

    /**
     * @param temMinima the temMinima to set
     */
    public void setTemMinima(String temMinima) {
        this.temMinima = temMinima;
    }

    /**
     * @return the mediaAr
     */
    public String getMediaAr() {
        return mediaAr;
    }

    /**
     * @param mediaAr the mediaAr to set
     */
    public void setMediaAr(String mediaAr) {
        this.mediaAr = mediaAr;
    }

    /**
     * @return the desviacion
     */
    public String getDesviacion() {
        return desviacion;
    }

    /**
     * @param desviacion the desviacion to set
     */
    public void setDesviacion(String desviacion) {
this.desviacion = desviacion;
    }
```

```
    /**
     * @return the rolAdmin
*/
    public boolean isRolAdmin() {
        return rolAdmin;
}

    /**
     * @param rolAdmin the rolAdmin to set
     */
    public void setRolAdmin(boolean rolAdmin) {
        this.rolAdmin = rolAdmin;
    }

    /**
     * @return the dateModelVoltaje
     */
    public LineChartModel getDateModelVoltaje() {
        return dateModelVoltaje;
    }

    /**
     * @param dateModelVoltaje the dateModelVoltaje to set
     */
    public void setDateModelVoltaje(LineChartModel dateModelVoltaje) {
        this.dateModelVoltaje = dateModelVoltaje;
    }

    /**
     * @return the dateModelPotencia
     */
    public LineChartModel getDateModelPotencia() {
        return dateModelPotencia;
    }

    /**
     * @param dateModelPotencia the dateModelPotencia to set
     */
    public void setDateModelPotencia(LineChartModel dateModelPotencia) {
        this.dateModelPotencia = dateModelPotencia;
    }

}
```

ANEXO 10

PESTAÑA DETALLES

```
package com.tesis.jsf.beans;

import com.tesis.jsf.dao.RegistrosDAO;
import com.tesis.jsf.dao.UserDAO;
import com.tesis.jsf.entities.Registro;
import com.tesis.jsf.util.SessionUtils;
import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import javax.annotation.PostConstruct;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;
import javax.servlet.http.HttpSession;
import org.primefaces.event.SelectEvent;

@ManagedBean(name = "det")
@SessionScoped
public class Detalles implements Serializable {

    private Date fechaDesde;
    private Date fechaHasta;
    private String tempMaxima;
    private String temMinima;
    private String mediaAr;
    private String desviacion;
    private List<Registro> registros;
    private boolean rolAdmin;

    @PostConstruct
    public void init() {

        setRolAdmin(verificarAdmin());

    }

    boolean verificarAdmin() {
        HttpSession session = SessionUtils.getSession();
        String userActual = session.getAttribute("username").toString();

        String rol = UserDAO.obtenerRolUsuario(userActual);
        if (rol.equalsIgnoreCase("A")) {
            return true;
        } else {
```

```
            return false;
        }

    }

    public void onDateSelect(SelectEvent event) {
        FacesContext facesContext = FacesContext.getCurrentInstance();
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        String fecha = format.format(event.getObject());

if (fechaHasta == null) {
            System.out.println(fecha + " Solo es Fecha Desde");
} else {
            SimpleDateFormat formatx = new SimpleDateFormat("dd/MM/yyyy");

setRegistros(RegistrosDAO.obtenerRegistrosRangos(format.format(fechaDesde),
format.format(fechaHasta)));

setTempMaxima(RegistrosDAO.obtenerTemperaturaMaxima(format.format(fechaDesd
e), format.format(fechaHasta)));

setTemMinima(RegistrosDAO.obtenerTemperaturaMinima(format.format(fechaDesde),
format.format(fechaHasta)));
            setMediaAr(RegistrosDAO.obtenerMediaAritmetica(registros));
            setDesviacion(RegistrosDAO.obtenerDesvia(registros));
            System.out.println(fecha + " Fecha Desde y Hasta");
}

    }

    public void onDateSelectHasta(SelectEvent event) {
        FacesContext facesContext = FacesContext.getCurrentInstance();
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        String fecha = format.format(event.getObject());
if (fechaDesde == null) {
            System.out.println(fecha + " Solo es Fecha Hasta");
} else {
            SimpleDateFormat formatx = new SimpleDateFormat("dd/MM/yyyy");

setRegistros(RegistrosDAO.obtenerRegistrosRangos(format.format(fechaDesde),
format.format(fechaHasta)));

setTempMaxima(RegistrosDAO.obtenerTemperaturaMaxima(format.format(fechaDesd
e), format.format(fechaHasta)));

setTemMinima(RegistrosDAO.obtenerTemperaturaMinima(format.format(fechaDesde),
format.format(fechaHasta)));
            setMediaAr(RegistrosDAO.obtenerMediaAritmetica(registros));
            setDesviacion(RegistrosDAO.obtenerDesvia(registros));
            System.out.println(fecha + " Fecha Desde y Hasta");
        }
```

```
        // System.out.println(fecha + " Fecha Hasta");

    }

    /**
     * @return the fechaDesde
     */
    public Date getFechaDesde() {
        return fechaDesde;
    }

    /**
     * @param fechaDesde the fechaDesde to set
     */
    public void setFechaDesde(Date fechaDesde) {
        this.fechaDesde = fechaDesde;
    }

    /**
     * @return the fechaHasta
     */
    public Date getFechaHasta() {
        return fechaHasta;
    }

    /**
     * @param fechaHasta the fechaHasta to set
     */
    public void setFechaHasta(Date fechaHasta) {
        this.fechaHasta = fechaHasta;
    }

    /**
     * @return the registros
*/
    public List<Registro> getRegistros() {
        return registros;
    }

    /**
     * @param registros the registros to set
     */
    public void setRegistros(List<Registro> registros) {
        this.registros = registros;
    }

    /**
     * @return the tempMaxima
     */
    public String getTempMaxima() {
        return tempMaxima;
```

```
    }

    /**
     * @param tempMaxima the tempMaxima to set
     */
    public void setTempMaxima(String tempMaxima) {
        this.tempMaxima = tempMaxima;
    }

    /**
     * @return the temMinima
     */
    public String getTemMinima() {
        return temMinima;
    }

    /**
     * @param temMinima the temMinima to set
     */
    public void setTemMinima(String temMinima) {
        this.temMinima = temMinima;
    }

    /**
     * @return the mediaAr
     */
    public String getMediaAr() {
        return mediaAr;
    }

    /**
     * @param mediaAr the mediaAr to set
     */
    public void setMediaAr(String mediaAr) {
this.mediaAr = mediaAr;
    }

    /**
     * @return the desviacion
     */
    public String getDesviacion() {
        return desviacion;
    }

    /**
     * @param desviacion the desviacion to set
     */
    public void setDesviacion(String desviacion) {
this.desviacion = desviacion;
    }
```

```
    /**
     * @return the rolAdmin
*/
    public boolean isRolAdmin() {
        return rolAdmin;
    }

    /**
     * @param rolAdmin the rolAdmin to set
     */
    public void setRolAdmin(boolean rolAdmin) {
this.rolAdmin = rolAdmin;
    }

}
```

ANEXO 11

CÓDIGO FUENTE DE LA BASE DE DATOS

```
-- Database: sistemaMonitor

-- DROP DATABASE "sistemaMonitor";

CREATE DATABASE "sistemaMonitor"
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'Spanish_Ecuador.1252'
    LC_CTYPE = 'Spanish_Ecuador.1252'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;


-- Table: public."Operador"

-- DROP TABLE public."Operador";

CREATE TABLE public."Operador"
(
    "idOperador" integer NOT NULL DEFAULT
nextval('"Operador_idOperador_seq"'::regclass),
    cedula character varying(10) COLLATE pg_catalog."default" NOT NULL,
    rol "char" NOT NULL,
    nombres character varying COLLATE pg_catalog."default" NOT NULL,
    apellidos character varying COLLATE pg_catalog."default" NOT NULL,
    correo character varying COLLATE pg_catalog."default",
    telefono character varying COLLATE pg_catalog."default",
    nicakname character varying(10) COLLATE pg_catalog."default" NOT NULL,
    password character varying(15) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Operador_pkey" PRIMARY KEY ("idOperador")
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public."Operador"
    OWNER to postgres;



-- Table: public.bach

-- DROP TABLE public.bach;
```

```
CREATE TABLE public.bach
(
    idbach bigint NOT NULL DEFAULT nextval('bach_idbach_seq'::regclass),
    estado character varying(1) COLLATE pg_catalog."default" NOT NULL,
    setpoint double precision,
    tiempo time without time zone,
    fecha date DEFAULT now(),
    hora time without time zone DEFAULT now(),
    CONSTRAINT bach_pkey PRIMARY KEY (idbach)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.bach
    OWNER to postgres;


-- Table: public.estadoequipo

-- DROP TABLE public.estadoequipo;

CREATE TABLE public.estadoequipo
(
    encendido boolean
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.estadoequipo
    OWNER to postgres;


-- Table: public.logs

-- DROP TABLE public.logs;

CREATE TABLE public.logs
(
    idregistro bigint NOT NULL DEFAULT nextval('logs_idregistro_seq'::regclass),
    idoperador integer,
    fecha_registro timestamp without time zone DEFAULT now(),
    detalle character varying(50) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT logs_pkey PRIMARY KEY (idregistro),
    CONSTRAINT logs_idoperador_fkey FOREIGN KEY (idoperador)
REFERENCES public."Operador" ("idOperador") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
```

```
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.logs
    OWNER to postgres;



-- Table: public.registros

-- DROP TABLE public.registros;

CREATE TABLE public.registros
(
    idregistro bigint NOT NULL DEFAULT nextval('registros_idregistro_seq'::regclass),
    idbach integer,
    temperaruta double precision NOT NULL,
    voltaje double precision NOT NULL,
    potencia double precision NOT NULL,
    fecha_registro timestamp without time zone DEFAULT now(),
    CONSTRAINT registros_pkey PRIMARY KEY (idregistro),
    CONSTRAINT registros_idbach_fkey FOREIGN KEY (idbach)
        REFERENCES public.bach (idbach) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.registros
    OWNER to postgres;
```

ANEXO 12

CONEXIÓN A BASE DE DATOS

```
package demoniosistemamonitor.database;

import java.sql.Connection;
import java.sql.DriverManager;

public class DataConnect {

	public static Connection getConnection() {
		try {
			Class.forName("org.postgresql.Driver");
			Connection con = DriverManager.getConnection(

	"jdbc:postgresql://localhost:5432/sistemaMonitor", "postgres", "admin");
			return con;
		} catch (Exception ex) {
			System.out.println("Database.getConnection() Error -->"
					+ ex.getMessage());
			return null;
		}
	}

	public static void close(Connection con) {
		try {
			con.close();
		} catch (Exception ex) {
		}
	}
}
```

ANEXO 13

BASE DE DATOS

```
package demoniosistemamonitor.database;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
public class EquipoDAO {

    public static String obtenerMediaAritmetica(List<Registro> registros) {
        float suma = 0;
        for (int i = 0; i < registros.size(); i++) {
            suma += registros.get(i).getTemperatura();
}
        float media = suma / registros.size();
String resultado = String.valueOf(media);
        return resultado;
    }

    public static String obtenerDesvia(List<Registro> registros) {
        float suma = 0;
        for (int i = 0; i < registros.size(); i++) {
            suma += registros.get(i).getTemperatura();
}
        float media = suma / registros.size();

double sum = 0;
        float varianza = 0;
        for (int i = 0; i < registros.size(); i++) {
            sum = Math.pow(registros.get(i).getTemperatura() - media, 2);
varianza += sum;
        }
        varianza = varianza / (registros.size() - 1);

double desvi = Math.sqrt(varianza);
        double resul = Math.rint(desvi * 100) / 100;
        return String.valueOf(resul);
    }

    public static String obtenerTemMaxBach(String idBach) {
        Connection con = null;
        PreparedStatement ps = null;

        try {
            con = DataConnect.getConnection();
```

```
            ps = con.prepareStatement("select * from public.\"registros\" where idbach = " +
idBach + " order by temperaruta desc limit 1;");

            ResultSet rs = ps.executeQuery();

            System.out.println(ps);

            while (rs.next()) {

                float temperatura = rs.getFloat("temperaruta");
                return String.valueOf(temperatura);

            }
        } catch (SQLException ex) {
            System.out.println("Login error -->" + ex.getMessage());
            return null;
        } finally {
            DataConnect.close(con);
        }
        return null;
    }

    public static String obtenerTemMinBach(String idBach) {
        Connection con = null;
        PreparedStatement ps = null;

        try {
            con = DataConnect.getConnection();
            ps = con.prepareStatement("select * from public.\"registros\" where idbach = " +
idBach + " order by temperaruta asc limit 1;");

            ResultSet rs = ps.executeQuery();

            System.out.println(ps);

            while (rs.next()) {

                float temperatura = rs.getFloat("temperaruta");
                return String.valueOf(temperatura);

            }
        } catch (SQLException ex) {
            System.out.println("Login error -->" + ex.getMessage());
            return null;
        } finally {
            DataConnect.close(con);
        }
        return null;
    }

    public static List<Registro> obtenerRegistros(String idBach) {
```

```
        Connection con = null;
        PreparedStatement ps = null;
        List<Registro> list = new ArrayList<Registro>();

        try {
            con = DataConnect.getConnection();
            ps = con.prepareStatement("select * from public.\"registros\" where idbach = " +
idBach + " order by idregistro asc;");

            ResultSet rs = ps.executeQuery();

            System.out.println(ps);

            while (rs.next()) {
                String fecha = rs.getString("fecha_registro");
                float temperatura = rs.getFloat("temperaruta");
                float voltaje = rs.getFloat("voltaje");
                float potencia = rs.getFloat("potencia");
                list.add(new Registro(fecha, temperatura, voltaje, potencia));
            }
        } catch (SQLException ex) {
            System.out.println("Login error -->" + ex.getMessage());
            return null;
        } finally {
            DataConnect.close(con);
        }
        return list;
    }

    public static Object[] obtenerDatosBach() {
        Connection con = null;
        PreparedStatement ps = null;
        Object datos[] = new Object[3];

        try {
            con = DataConnect.getConnection();
            ps = con.prepareStatement("select * from public.\"bach\" where estado = 'A';");

            ResultSet rs = ps.executeQuery();

            System.out.println(ps);

            while (rs.next()) {

                datos[0] = rs.getInt("idbach");
datos[1] = rs.getString("hora");
                datos[2] = rs.getString("tiempo");
return datos;

            }
        } catch (SQLException ex) {
```

```
            System.out.println("Login error -->" + ex.getMessage());
            return null;
        } finally {
            DataConnect.close(con);
        }
        return null;
    }

    public static boolean actualizarEstado(boolean estado) {
        try {

            Connection conn = DataConnect.getConnection();

            String query = "update public.\"estadoequipo\" set encendido = ? ;";
            PreparedStatement preparedStmt = conn.prepareStatement(query);
            preparedStmt.setBoolean(1, estado);

            preparedStmt.executeUpdate();

            conn.close();
            return true;
        } catch (Exception e) {
            System.err.println("Got an exception! ");
            System.err.println(e.getMessage());
        }
        return false;
    }

    public static boolean actualizarDestilacion(String estado) {
        try {

            Connection conn = DataConnect.getConnection();

            String query = "update public.\"bach\" set estado = ? ;";
            PreparedStatement preparedStmt = conn.prepareStatement(query);
            preparedStmt.setString(1, estado);

            preparedStmt.executeUpdate();

            conn.close();
            return true;
        } catch (Exception e) {
            System.err.println("Got an exception! ");
            System.err.println(e.getMessage());
        }
        return false;
    }

    public static boolean obtenerEstadoEquipo() {
        Connection con = null;
        PreparedStatement ps = null;
```

```
        try {
            con = DataConnect.getConnection();
            ps = con.prepareStatement("select * from public.\"estadoequipo\";");

            ResultSet rs = ps.executeQuery();

            System.out.println(ps);

            while (rs.next()) {

                boolean encendido = rs.getBoolean("encendido");
                return encendido;

            }
        } catch (SQLException ex) {
            System.out.println("Login error -->" + ex.getMessage());
            return false;
        } finally {
            DataConnect.close(con);
        }
        return false;
    }

    public static int obtenerNumeroDestilaciones(String fecha) {
        Connection con = null;
        PreparedStatement ps = null;

try {
            System.out.println("Numero Destilaciones");
con = DataConnect.getConnection();
            ps = con.prepareStatement("SELECT count(*) as numero FROM public.bach
where fecha = '" + fecha + "'");

            ResultSet rs = ps.executeQuery();

            System.out.println(ps);

            while (rs.next()) {

                int numero = rs.getInt("numero");
                return numero;

            }
        } catch (SQLException ex) {
            System.out.println("Login error -->" + ex.getMessage());
            return 0;
        } finally {
            DataConnect.close(con);
        }
        return 0;
```

```
    }

    public static int obtenerIdBach() {
        Connection con = null;
        PreparedStatement ps = null;

        try {
            con = DataConnect.getConnection();
            ps = con.prepareStatement("select * from public.\"bach\" where estado = 'A';");

            ResultSet rs = ps.executeQuery();

            System.out.println(ps);

            while (rs.next()) {

                return rs.getInt("idbach");

            }
        } catch (SQLException ex) {
            System.out.println("Login error -->" + ex.getMessage());
            return -1;
        } finally {
            DataConnect.close(con);
}
        return -1;
    }

    public static boolean insertarRegistro(int idBach, float temperatura, float voltaje, float
potencia) {
        Connection con = null;
        PreparedStatement ps = null;
        //insert into bach (estado,setPoint,tiempo) values ('A',25,'00:20:00');
        try {
            con = DataConnect.getConnection();
            ps = con.prepareStatement("insert into registros
(idBach,temperaruta,voltaje,potencia) values ( " + idBach + " ,  " + temperatura + ", " +
voltaje + ", " + potencia + "); ");

            ps.execute();
            return true;

        } catch (Exception ex) {
            System.out.println("Login error -->" + ex.getMessage());
            return false;
        } finally {
            DataConnect.close(con);
}

    }
}
```

ANEXO 14

REGISTRO DE BASE DE DATOS

```
package demoniosistemamonitor.database;

import java.io.Serializable;

public class Registro implements Serializable {

    private String fecha;
    private float temperatura;
    private float voltaje;
    private float potencia;

    public Registro() {
    }

    public Registro(String fecha, float temperatura, float voltaje, float potencia) {
        this.fecha = fecha;
        this.temperatura = temperatura;
        this.voltaje = voltaje;
        this.potencia = potencia;
}


    /**
     * @return the fecha
     */
    public String getFecha() {
        return fecha;
    }

    /**
     * @param fecha the fecha to set
     */
    public void setFecha(String fecha) {
this.fecha = fecha;
    }

    /**
     * @return the temperatura
*/
    public float getTemperatura() {
        return temperatura;
    }

    /**
     * @param temperatura the temperatura to set
     */
```

```
    public void setTemperatura(float temperatura) {
this.temperatura = temperatura;
    }

    /**
     * @return the voltaje
     */
    public float getVoltaje() {
        return voltaje;
    }

    /**
     * @param voltaje the voltaje to set
     */
    public void setVoltaje(float voltaje) {
this.voltaje = voltaje;
    }

    /**
     * @return the potencia
*/
    public float getPotencia() {
        return potencia;
    }

    /**
     * @param potencia the potencia to set
*/
    public void setPotencia(float potencia) {
        this.potencia = potencia;
    }



}
```

ANEXO 15

DEMONIO SISTEMA

```
package demoniosistemamonitor;

import demoniosistemamonitor.database.EquipoDAO;
import demoniosistemamonitor.database.Registro;
import java.io.*;
import java.net.*;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

public class DemonioSistemaMonitor {

    static ArrayList<Integer> listaEnviados = new ArrayList<Integer>();
    static DataOutputStream raspberry;

    public static void main(String[] args) {
try {
            verificarCorreo();
            mqtt();
            iniciarTCP();
            udpInterno();
while (true) {
                try {
                    Thread.sleep(20000000);
                } catch (Exception e) {
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void verificarCorreo() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                while (true) {
                    try {
                        Thread.sleep(5000);
Object[] datos = EquipoDAO.obtenerDatosBach();

                        if (datos != null) {
int id = (int) datos[0];
```

```
                    String hora = String.valueOf(datos[1]);
                    String tiempo = String.valueOf(datos[2]);
                    SimpleDateFormat df = new SimpleDateFormat("HH:mm:ss");
Date d = df.parse(hora);
Calendar cal = Calendar.getInstance();
                    cal.setTime(d);
                    String timpos[] = tiempo.split(":");
                    cal.add(Calendar.HOUR, Integer.parseInt(timpos[0]));
                    cal.add(Calendar.MINUTE, Integer.parseInt(timpos[1]));
                    cal.add(Calendar.SECOND, Integer.parseInt(timpos[2]));
                    String tiempoTotal = df.format(cal.getTime());

                    DateFormat dfx = new SimpleDateFormat("HH:mm:ss");
Date horaTerminar = dfx.parse(tiempoTotal);

String tiempoInicio = new SimpleDateFormat("HH:mm:ss").format(new Date());
Date horaInicio = dfx.parse(tiempoInicio);
                    if (horaInicio.compareTo(horaTerminar) > 0) {
                        if (!yaSeEnvio(id)) {
                            String mensaje = "<p><span style=\"color:
#ff0000;\"><strong>DEBE DETENER LA DESTILACION QUE INICIO A LAS " + hora +
" HORAS</strong></span></p>";
                            GmailSendEmailSSL.enviarMensaje(mensaje, "Alerta! Detener
Destilacion");
                            listaEnviados.add(id);
                        }

                    }

                }
            } catch (Exception e) {
}

        }

    }
    }).start();
  }

  public static boolean yaSeEnvio(int n) {
    int t = listaEnviados.size();
    for (int i = 0; i < t; i++) {
      if (listaEnviados.get(i) == n) {
        return true;
      }
    }
    return false;
  }

  public static void mqtt() {
    new Thread(new Runnable() {
```

```
            @Override
            public void run() {
                MqttSubscribe.main(null);
            }
        }).start();
    }

    public static void iniciarTCP() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    while (true) {
                        try {
                            ServerSocket welcomeSocket = new ServerSocket(5005);

                            Socket connectionSocket = welcomeSocket.accept();

                            raspberry = new
DataOutputStream(connectionSocket.getOutputStream());
System.out.println("Se Conectó el Equipo");
} catch (Exception e) {
                        }

                    }

                } catch (Exception e) {
                }
            }
        }).start();
    }

    public static void udpInterno() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    DatagramSocket serverSocket = new DatagramSocket(9876);
                    byte[] receiveData = new byte[1024];

                    while (true) {
                        receiveData = new byte[1024];
                        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                        serverSocket.receive(receivePacket);
                        String sentence = new String(receivePacket.getData());
                        System.out.println("Data: " + sentence);
                        String datos[] = sentence.trim().split(";");
                        if (datos.length == 5) {

                            String idDestilacion = datos[0];
```

```
String numeroDestilaciones = datos[1];
                        String hora = datos[2];
                        String tiempo = datos[3];
                        String tiempoAhora = datos[4];
                        List<Registro> list = EquipoDAO.obtenerRegistros(idDestilacion);
String temMax = EquipoDAO.obtenerTemMaxBach(idDestilacion);
String temMin = EquipoDAO.obtenerTemMinBach(idDestilacion);
                        String media = EquipoDAO.obtenerMediaAritmetica(list);
                        String desviacion = EquipoDAO.obtenerDesvia(list);

                        String infoCorreo = "<p><span style=\"color: #0000ff;\"><strong>Se
detuvo la Destilacion #" + numeroDestilaciones + " a las " + tiempoAhora + " horas, que
inicio a las " + hora + " horas y debio estsar destilando durante el tiempo de " + tiempo
+ "</strong></span></p>\n"
+ "<table style=\"height: 104px; width: 400px;\" border=2>\n"
                                + "<tbody>\n"
                                + "<tr>\n"
                                + "<td style=\"width: 192px;\" bgcolor=\"#00ff00\">Temperatura
Maxima:</td>\n"
                                + "<td style=\"width: 196px;\" bgcolor=\"#bfff00\">" + temMax +
"</td>\n"
                                + "</tr>\n"
                                + "<tr>\n"
                                + "<td style=\"width: 192px;\" bgcolor=\"#00ff00\">Temperatura
Minima:</td>\n"
                                + "<td style=\"width: 196px;\" bgcolor=\"#bfff00\">" + temMin +
"</td>\n"
                                + "</tr>\n"
                                + "<tr>\n"
                                + "<td style=\"width: 192px;\" bgcolor=\"#00ff00\">Media
Aritmetica:</td>\n"
                                + "<td style=\"width: 196px;\" bgcolor=\"#bfff00\">" + media +
"</td>\n"
+ "</tr>\n"
                                + "<tr>\n"
                                + "<td style=\"width: 192px;\" bgcolor=\"#00ff00\">Desviacion
estandar:</td>\n"
                                + "<td style=\"width: 196px;\" bgcolor=\"#bfff00\">" + desviacion +
"</td>\n"
                                + "</tr>\n"
                                + "</tbody>\n"
                                + "</table>";
GmailSendEmailSSL.enviarMensaje(infoCorreo, "Destilacion Detenida");

} else {
                        raspberry.write(sentence.trim().getBytes());
                    }

                }
            } catch (Exception e) {
                e.printStackTrace();
```

```
}

            }
        }
        ).start();

    }

}
```

ANEXO 16

ENVÍO DE CORREOS

```
package demoniosistemamonitor;

import javax.mail.*;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import java.util.Date;
import java.util.Properties;

public class GmailSendEmailSSL {

   private static final String USERNAME = "tesisinformacionr";
   private static final String PASSWORD = "*****";

   public static void enviarMensaje(String mensaje, String tema) {

      try {
         String mailFrom = "tesisinformacionr@gmail.com";
         String mailTo = "correoInf@gmail.com";
         String mailSubject = tema;
         String mailText = mensaje;

         GmailSendEmailSSL gmail = new GmailSendEmailSSL();
         gmail.sendMail(mailFrom, mailTo, mailSubject, mailText);
      } catch (Exception e) {
      }
   }

   private void sendMail(String mailFrom, String mailTo, String mailSubject,
         String mailText) throws Exception {

      Properties config = createConfiguration();

      Session session = Session.getInstance(config, new Authenticator() {
         @Override
         protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(
                  GmailSendEmailSSL.USERNAME,
                  GmailSendEmailSSL.PASSWORD
            );
         }
      });


      Message message = new MimeMessage(session);
      message.setSentDate(new Date());
      message.setFrom(new InternetAddress(mailFrom));
```

```
        message.setRecipient(Message.RecipientType.TO, new InternetAddress(mailTo));
        message.setSubject(mailSubject);
        String htmlText = (mailText);
        message.setContent(htmlText, "text/html; charset=utf-8");


        Transport.send(message, GmailSendEmailSSL.USERNAME,
GmailSendEmailSSL.PASSWORD);
    }

    private Properties createConfiguration() {
        return new Properties() {
            {
                put("mail.smtp.host", "smtp.gmail.com");
                put("mail.smtp.auth", "true");
                put("mail.smtp.port", "465");
                put("mail.smtp.socketFactory.port", "465");
                put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
}
        };
    }
}
```

ANEXO 17

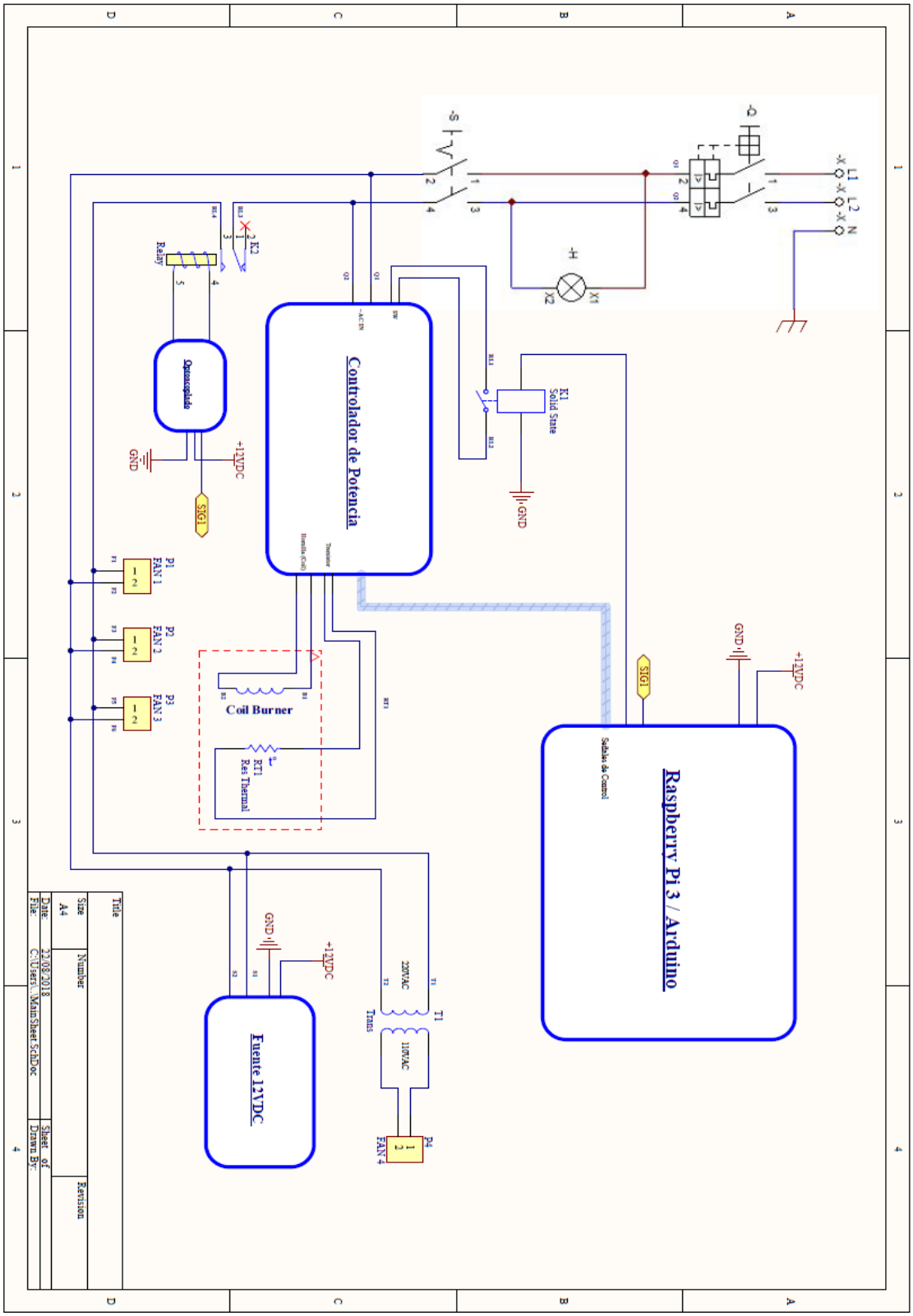
MQTT

```
package demoniosistemamonitor;

import demoniosistemamonitor.database.EquipoDAO;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

public class MqttSubscribe {

        public static void main(String[] args) {

                String topic = "Tesis";
                int qos = 2;
                String broker = "tcp://localhost:1883";
                String clientId = "SubscribeSample";
                MemoryPersistence persistence = new MemoryPersistence();

                try {
                        MqttClient sampleClient = new MqttClient(broker, clientId,
                                        persistence);
                        sampleClient.setCallback(new MqttCallback() {

                                @Override
                                public void messageArrived(String topic,
MqttMessage msg)
                                                throws Exception {
                                        System.out.println("topic:" + topic);
                                        System.out.println("msg:" + new
String(msg.getPayload()));
                        String data= new String(msg.getPayload());
                        String datos[]= data.trim().split(";");
                        float temperatura=Float.parseFloat(datos[0]);
                        float voltaje=Float.parseFloat(datos[1]);
                        float potencia=Float.parseFloat(datos[2]);
                        if(datos.length==3){
                          int idB= EquipoDAO.obtenerIdBach();
if(idB!=-1){
                            EquipoDAO.insertarRegistro(idB, temperatura, voltaje,
potencia);
}

                        }
```

```
                                        }

                                        @Override
                                        public void deliveryComplete(IMqttDeliveryToken
token) {

                                        }

                                        @Override
                                        public void connectionLost(Throwable cause) {
                                                cause.printStackTrace();
                                        }
                                });
                                MqttConnectOptions options = new MqttConnectOptions();

                                sampleClient.connect(options);
                                sampleClient.subscribe(topic, qos);

                        } catch (MqttException me) {

                                me.printStackTrace();
                        }
                }

}
```

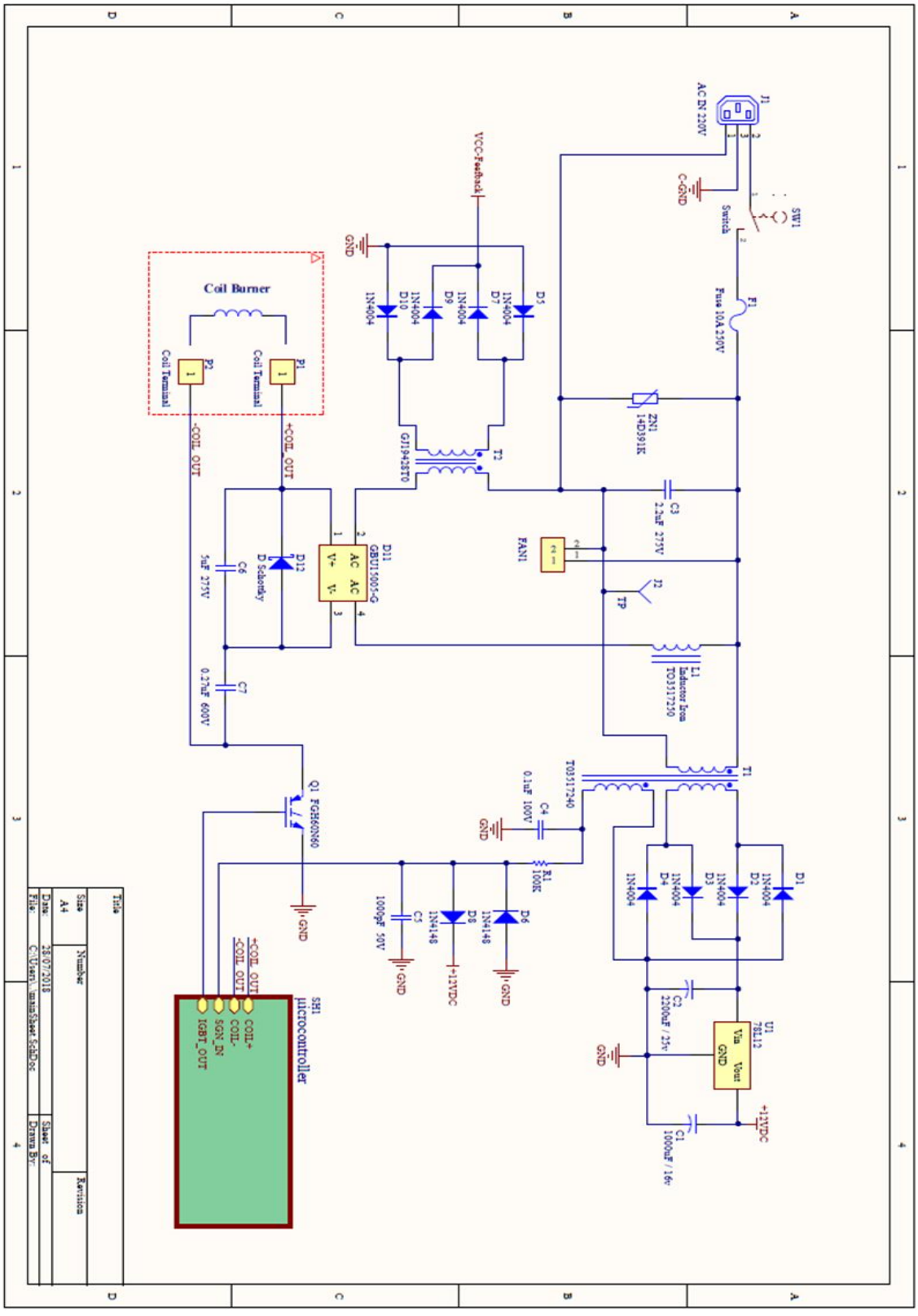
ANEXO 18

Circuito Electrico



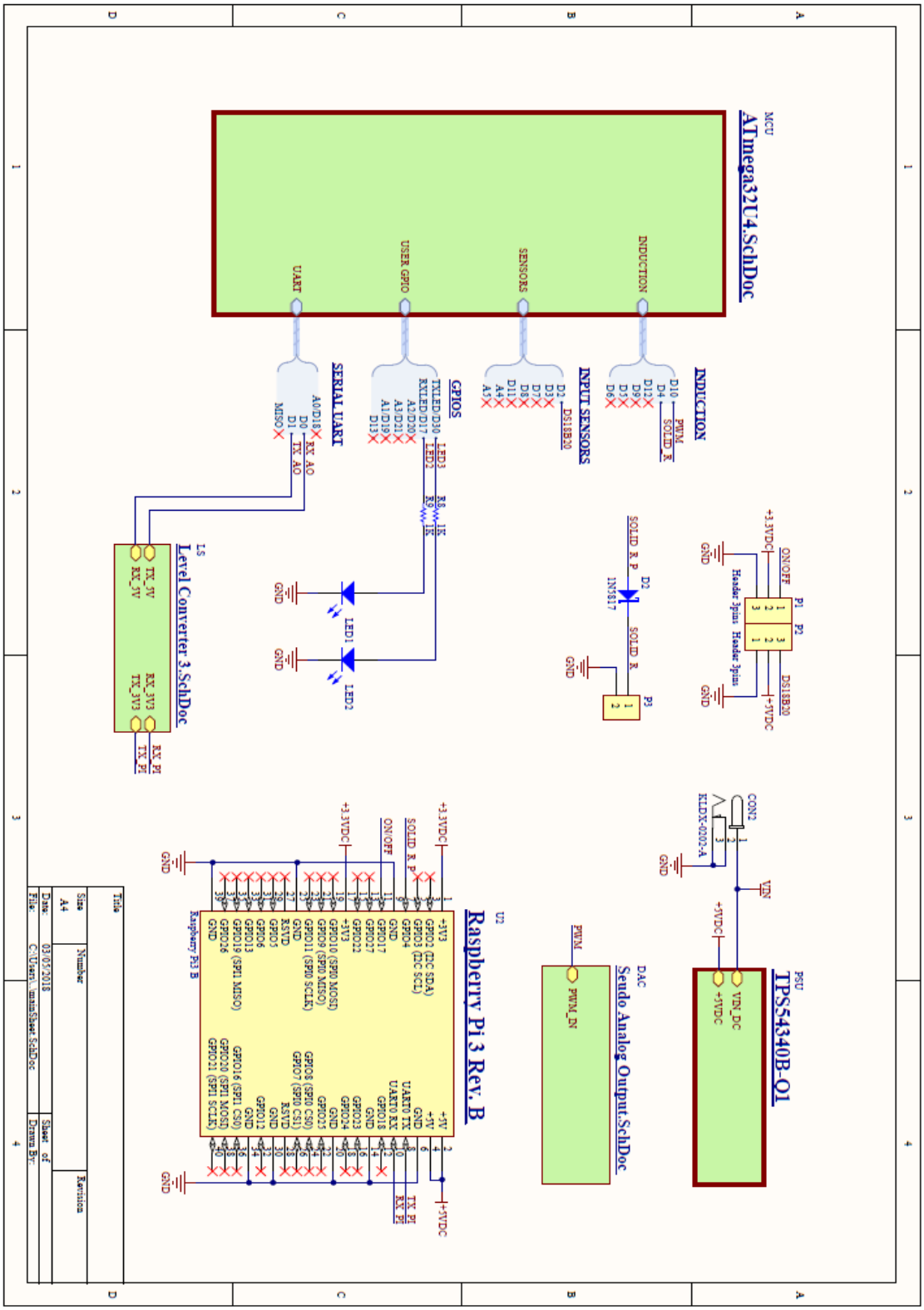
ANEXO 19

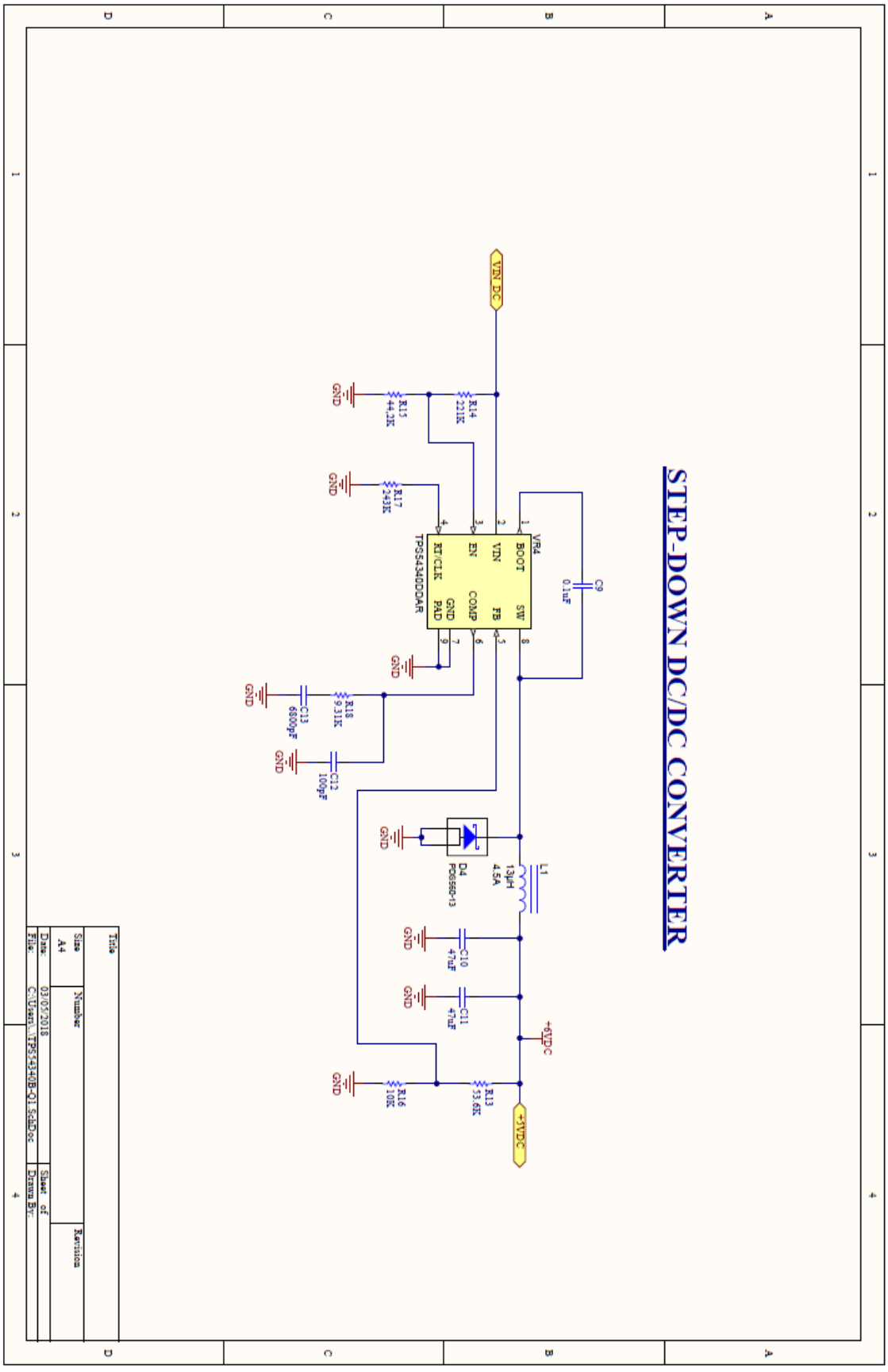
Circuito de potencia

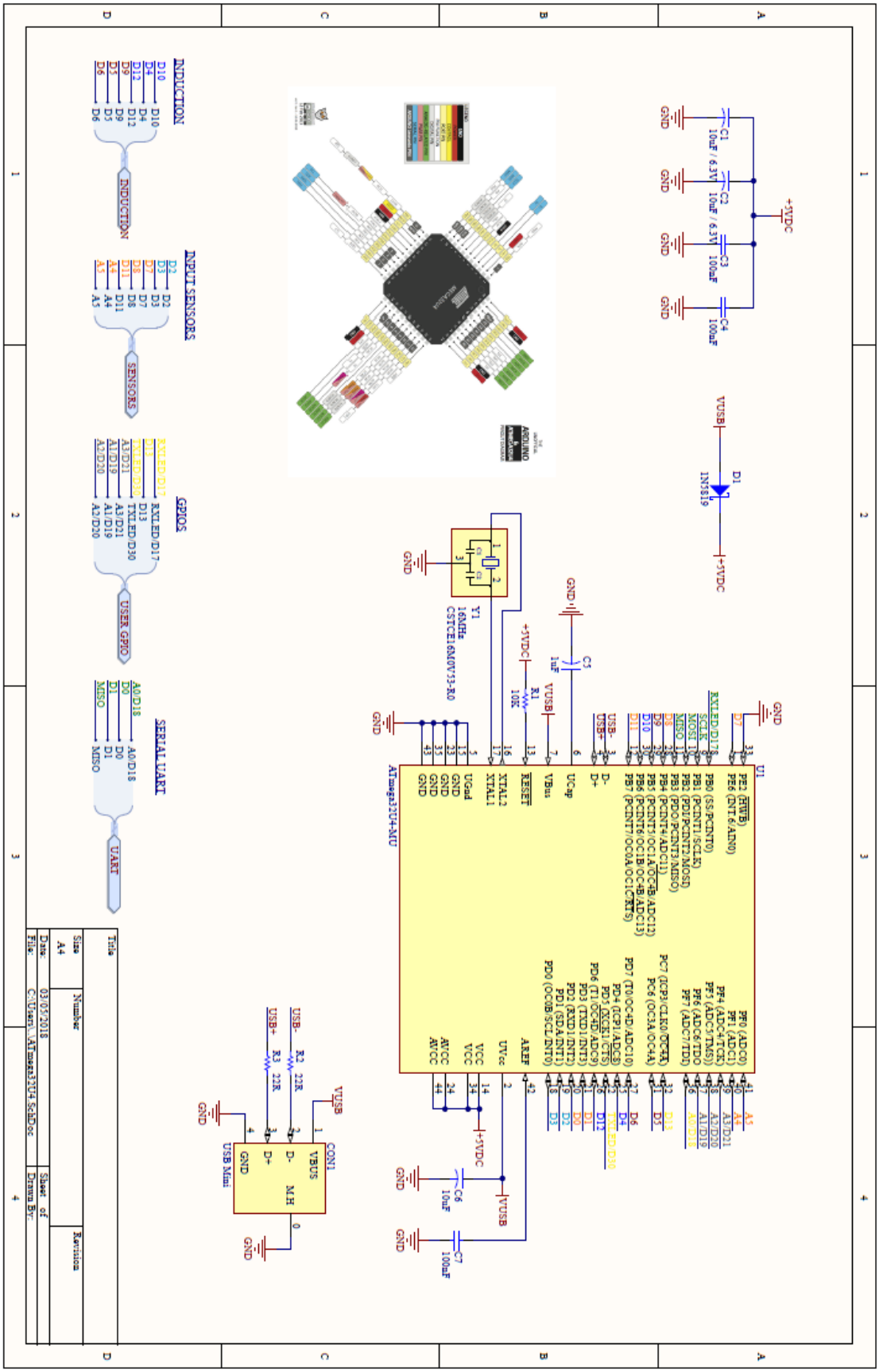


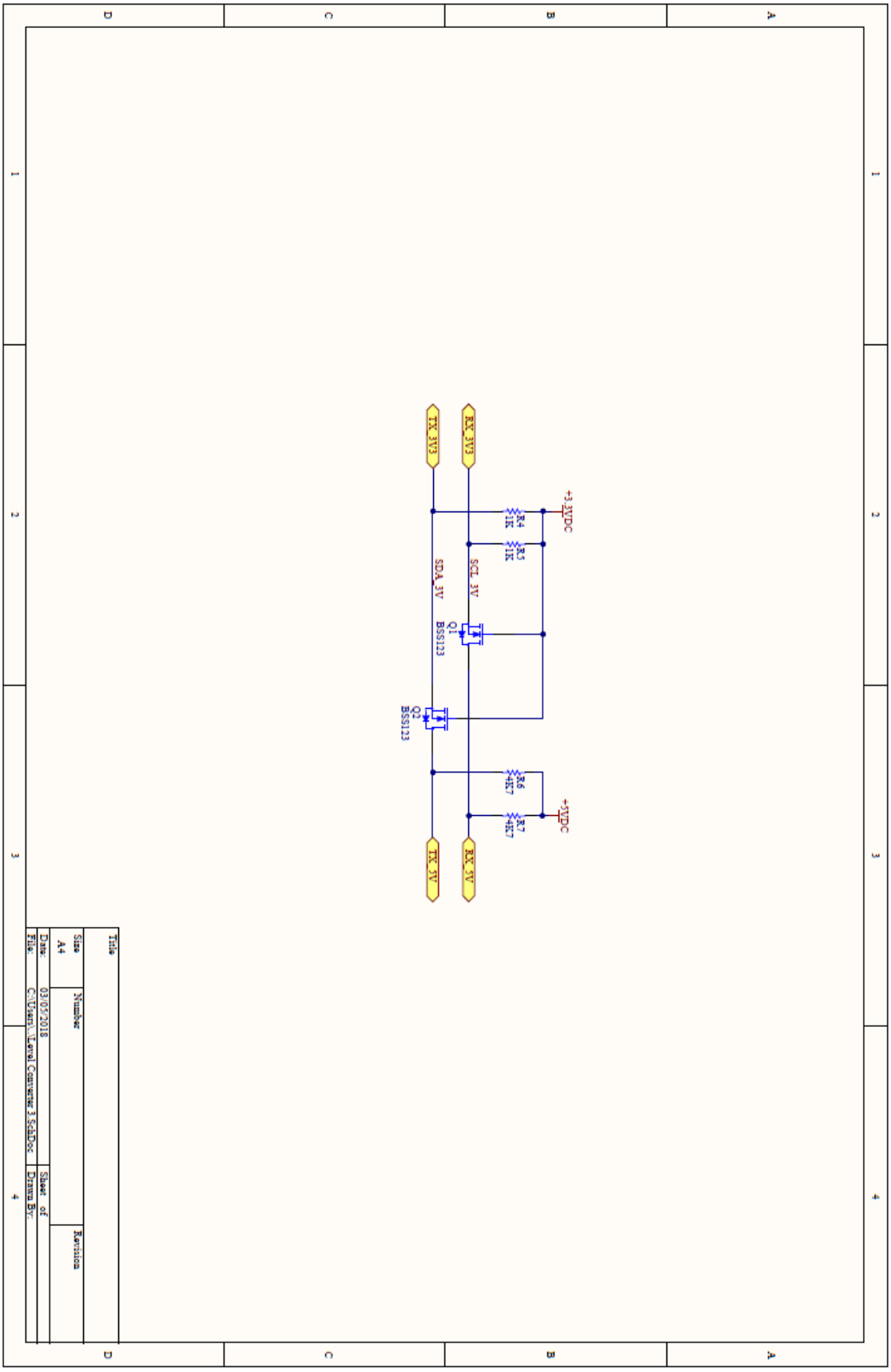
ANEXO 20

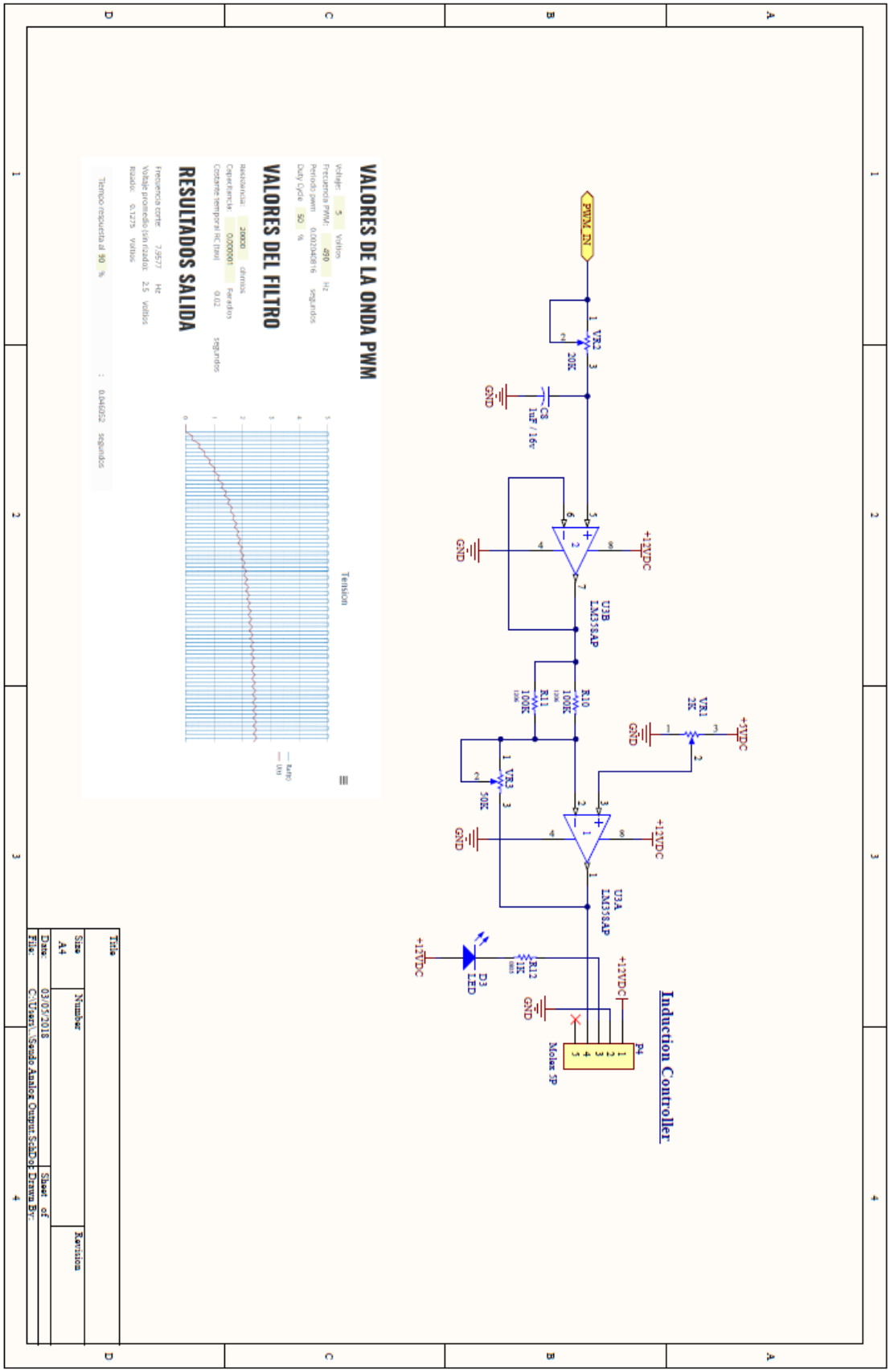
Circuitos de Control




STEP-DOWN DC/DC CONVERTER
VIN_DC
VR4
TPS54340DDAR
BOOT
VIN
EN
SW
RT/CLK
GND PAD
COMP
FB
C9
0.1uF
R14
221K
R15
44.2K
R17
243K
C13
6800pF
R18
9.31K
C12
100pF
D4
PDS560-13
L1
13µH
4.5A
C10
47uF
C11
47uF
R16
10K
R13
53.6K
+6VDC
+3VDC
GND
Title
Size
A4
Number
Revision
Date:
03/05/2018
File:
C:\Users\...TPS54340B-Q1.SchDoc
Sheet of
Drawn By:


INDUCTION
INPUT SENSORS
GPIOS
SERIAL UART
Title
Size
A4
Number
Revision
Date:
03/05/2018
Sheet of
File:
C:\Users\...\ATmega32U4 SchDoc
Drawn By:


TX_3V3
RX_3V3
+3.3VDC
R4
1K
R5
1K
SCL_3V
SDA_3V
Q1
BSS123
Q2
BSS123
R6
4K7
R7
4K7
+5VDC
TX_5V
RX_5V
Title
Size
A4
Number
Revision
Date:
03/05/2018
Sheet of
File:
C:\Users\...\Level Converter 3.SchDoc
Drawn By:


Induction Controller
PWM_IN
VR2
20K
C8
1uF / 16v
GND
U3B
LM358AP
+12VDC
VR1
2K
R10
100K
R11
100K
VR3
50K
U3A
LM358AP
D3
LED
R12
1K
P4
Molex 5P
VALORES DE LA ONDA PWM
VALORES DEL FILTRO
RESULTADOS SALIDA
Tension
Title
Size
A4
Number
Revision
Date:
03/05/2018
Sheet of
File:
C:\Users\...\Senido Analog Output.SchDoc
Drawn By: