INSTITUTO TECNOLÓGICO SUPERIOR DE LERDO

DIVISIÓN DE POSGRADO

DISEÑO DE INTERFACES DE HARDWARE PARA PRÁCTICAS DE CONTROL DE PROCESOS UTILIZANDO RASPBERRY PI

T E S I S

PARA OBTENER EL TÍTULO DE MAESTRO EN INGENIERÍA MECATRÓNICA

PRESENTA:

ING. EDUARDO MACHADO DÍAZ

Asesor: Dr. Hesner Coto Fuentes

La excelencia académica al servicio de la sociedad

Ciudad Lerdo, Durango. diciembre 2018

Cd. Lerdo, Dgo., a 03 de Diciembre de 2018.
Oficio No. DA/425/2018

C. EDUARDO MACHADO DIAZ
ALUMNO DE LA MAESTRÍA EN ING. EN MECATRÓNICA.

Se le comunica que la comisión revisora integrada por los cuatro sinodales, reviso y aprobó en su totalidad el trabajo profesional.

"DISEÑO DE INTERFACES DE HARDWARE PARA PRÁCTICAS DE CONTROL DE PROCESOS UTILIZANDO RASPBERRY PI"

Presentado por Usted, para obtener su Título de:

MAESTRO EN INGENIERÍA MECATRÓNICA

Por lo anterior, y de acuerdo a los lineamientos profesionales, se da el trámite legal para que proceda a la impresión del trabajo presentado.

A T E N T A M E N T E
"LA EXCELENCIA ACADÉMICA AL SERVICIO DE LA SOCIEDAD"

M.C. JOSÉ ÁNGEL MÉNDEZ ORTEGA
DIRECTOR ACADÉMICO

INSTITUTO TECNOLÓGICO SUPERIOR DE LERDO
DIRECCIÓN ACADÉMICA

c.c.p. JAMO/CRM/mebe

R1/03/18

F-GE-01-006

Cd. Lerdo, Dgo., a 30 de Noviembre de 2018.

Ing. César Ríos Marmolejo
Jefe de la División de Posgrado
P r e s e n t e

Por este conducto hacemos de su conocimiento que después de haber procedido a la Revisión Rigurosa y Detallada de la Tesis del Alumno:

"C. EDUARDO MACHADO DIAZ"

Cuyo Título es:

"DISEÑO DE INTERFACES DE HARDWARE PARA PRÁCTICAS DE CONTROL DE PROCESOS UTILIZANDO RASPBERRY PI"

Este Jurado considera aprobada para la generación de los ejemplares que serán entregados de forma electrónica.

Atentamente:

M.C. Víctor Edi Manqueros Avilés
Presidente

M.C. Ernesto II Castro Juárez
Secretario

M.C. Noé Alvarado Tovar.
Vocal Propietario

M.C. Francisco Huerta Valenzuela
Vocal Suplente

RI/02/2018

F-GE-01-004

DEDICATORIA

"Es justamente la posibilidad de realizar un sueño lo que hace que la vida sea interesante."

Paulo Cohelo.

A Dios.

Por haberme permitido llegar hasta este punto y haberme dado salud para lograr mis objetivos y darme personas hermosas a mí alrededor para cumplir mis metas.

A mis padres.

Por darme la vida y ser el pilar fundamental en todo lo que soy, en mi educación, tanto académica como de la vida, por su incondicional apoyo mantenido a través del tiempo en toda mi formación académica y sobre todo por enseñarme a ser lo que soy y apoyarme en todo lo que me he propuesto, aunque nadie más lo quisiera hacer. Todo este trabajo ha sido posible gracias a ellos.

A mis hermanos.

Han sido la motivación y apoyo para lograr mis objetivos, por darme consejos y guiarme en mi vida, darme alas para darme cuenta de lo que soy capaz y puedo llegar a ser, apoyarme en mis decisiones y estar conmigo.

A mis maestros.

Sin ellos no tendría los conocimientos que tengo actualmente, por darme consejos y siempre estar dispuestos a ayudarme sin importar si se ha acabo su tiempo de trabajo, por ser en muchas ocasiones, quienes me han impulsado a trabajar en el área de la investigación.

AGRADECIMIENTOS

Al Dr. Hesner Coto Fuentes, por dejarme participar en esta investigación y brindarme su tiempo para guiarme a lo largo de la realización de este proyecto, así como parte de sus investigaciones en esta área, para facilitar el trabajo e indicarme un sinnúmero de mejoras y formas de aclarar las dudas del presente informe, sin contar todas las enseñanzas y motivaciones que me ha dado.

Al Instituto Tecnológico Superior de Lerdo, por ser el núcleo de mi aprendizaje, darnos un espacio para la realización del proyecto, así como los equipos necesarios para las pruebas y construcción del sistema. A los docentes que integran la Academia de Ingeniería Electrónica y la División de Posgrado, por su gran apoyo tanto personal como profesionalmente.

Índice General

Índice de Figuras

Índice de Tablas

CAPÍTULO 1: TEMAS PRELIMINARES

1.1 RESUMEN

La realización de sistemas mecatrónicos actualmente tiene un coste elevado y es necesario un conocimiento especializado en diferentes áreas de la ingeniería y un pago de licencia para el uso de distintos productos de automatización, por lo que se ha comenzado a buscar otras alternativas. Una de éstas es la microcomputadora Raspberry Pi® la cual es de software y hardware libre.

El siguiente trabajo presenta la elaboración de las interconexiones necesarias para comunicar la microcomputadora Raspberry Pi® utilizando el sistema operativo Raspbian®, con diferentes dispositivos con la finalidad brindarle las características necesarias para su uso como plataforma para el aprendizaje en el área control de procesos, como lo son conversores análogos – digital y el acondicionamiento necesario para la activación de los actuadores presentes en los sistemas de control, con la finalidad de desarrollar sistemas mecatrónicos de bajo costo y con independencia tecnológica, es decir, sin depender de software de terceros que pudiesen tener alto precio.

Así mismo la programación de una interfaz visual remota que permite al sistema realizar mediciones y análisis de datos mediante una conexión a internet y con controles básicos para cualquier operador. El sistema se ha realizado utilizando la plataforma Node-Red®, que es un lenguaje gráfico basado en Java Script®. En adición con una transmisión de video en tiempo real que permita la supervisión del sistema de manera visual.

1.2 ABSTRACT

The realization of mechatronic systems currently has a high cost and it is necessary a specialized knowledge in different areas of engineering and a license payment for the use of different automation products, which is why we have begun to look for other alternatives. One of these is the Raspberry Pi® microcomputer, which is free software and hardware.

The following work presents the development of the necessary interconnections to communicate the Raspberry Pi® microcomputer using the Raspbian® operating system, with different devices. in order to provide the necessary features for control learning, such as analogue converters - digital and the necessary conditioning for the activation of the most common actuators in control system design, with the aim of developing low-cost mechatronic systems with technological independence, that is, without relying on third-party software that could have a high price.

In addition, the programming of a remote visual interface that allows the system to perform measurements and data analysis through an internet connection. The system has basic controls for any operator and was create using the Node-Red® platform, which is a graphic language, based on Java Script®. A Video Streaming has been included to see in real time the system working.

1.3 INTRODUCCIÓN

En la actualidad existen una gran variedad de procesos industriales que requieren que se tenga un control de las variables necesarias en la creación de los productos como lo pueden ser la fabricación de productos alimenticios, procesos químicos, generación de energía, entro otros.

Cada uno de ellos contienen elementos que deben de ser controlados para mantener un nivel específico sin importar las circunstancias que se tengan, entre ellos la temperatura, el flujo, nivel, etc. Para ello es indispensable conocimiento en el área de control de procesos siendo ésta una de las más complicadas en el estudio de la automatización por su complejidad teórica y escasos ejemplos prácticos. Si bien existen dispositivos que sintonizan de forma automática los procesos, es indispensable conocer los aspectos básicos.

Actualmente se están desarrollando proyectos encaminados a facilitar herramientas a los estudiantes sin necesidad de que acudan físicamente a una institución educativa, abriendo paso a la educación a distancia.

La educación a distancia (ED) en México ha transitado por varias etapas, aumentando y fortaleciendo su presencia e impacto social en los últimos años. Es un modelo educativo donde los estudiantes no deben asistir a un aula y deben planificar su tiempo de estudio, prima la flexibilidad, la responsabilidad y el autoaprendizaje (Zubieta, 2016).

Es una herramienta que permite que personas sin acceso a una institución educativa en su localidad o aquellas que presenten dificultades para asistir a una, puedan cursar estudios ya sea profesionales o que complementen sus conocimientos en un área determinada.

Este tipo de educación permite que los alumnos obtengan toda la información necesaria de manera teórica sin embargo se carece, en muchos casos, de experiencias físicas que consoliden los conocimientos y apoyen el desarrollo de habilidades profesionales (Juca, 2016).

El uso cada vez más extendido de internet abre para la ED un marco de acción con grandes posibilidades, al permitir la creación y uso de herramientas que fomentan la instrucción, el intercambio de recursos a distancia y el perfeccionamiento de las competencias de los estudiantes (Falcón, 2013). Por ello es necesario crear plataformas alternativas a las actuales que permitan creas sistemas como ejemplo práctico y que puedan ser creadas de

manera fácil y económica utilizando sistemas de uso libre como lo son las microcomputadoras.

La Raspberry Pi® es un microordenador creado en Inglaterra con la finalidad de brindar de equipos de cómputo a escuelas con bajos recursos y que cuenta con entradas y salidas de propósito general para controlar pequeñas cargas. Este tipo de dispositivos se ha vuelto muy popular entre los desarrolladores de sistemas informáticos en diferentes áreas como la automatización y la domótica por la facilidad de conectarse a internet con ellas.

La popularidad de esta herramienta viene de ser un proyecto de código libre, evitando pagos de licencias y compra de equipos especializados que pueden tener un elevado costo si se compra a los fabricantes más populares.

Sin embargo, al no estar creada con fines de desarrollo en automatización o un ámbito industrial, no cuenta con los elementos más importantes en la creación de sistemas de control y sistemas de adquisición de datos. Es por ello que se deben desarrollar interconexiones entre periféricos que permitan incluir estas características en la placa para diferentes aplicaciones y que el uso de ésta se vaya incrementando.

Al ser un proyecto de código abierto se tiene una amplia documentación de desarrollos de diferentes sistemas que pueden ayudar a las personas interesadas en la implementación de ellos para llevarlos a cabo sin tener un conocimiento muy amplio en áreas de la automatización o de los equipos más populares como lo son los Controladores Lógicos Programables.

La reducción de costos por pago de licencias que se tiene utilizando las microcomputadoras debe de estar acorde con el gasto de la implementación de las interconexiones con los dispositivos que se van a emplear. Se busca agregar los periféricos mínimos necesarios para dotar de las características más esenciales a la tarjeta como lo son un conversor de señales análogas a digitales y la electrónica para el manejo de los diversos actuadores presentes en las diferentes plataformas que se pretenden crear.

Este tipo de sistemas se deben de desarrollar pensando que el usuario final no tiene un conocimiento especializado siendo necesaria la programación de una interfaz visual de fácil uso con controles sencillos que permitan la adquisición de datos y el control completo del hardware diseñado.

Debido a que la tarjeta Raspberry Pi® es muy popular en el Internet de las Cosas, se pretende implementar un sistema que pueda manejarse de manera remota y con una transmisión de video para supervisar el estado de la planta en todo momento.

En el Capítulo 1 se presentan todos los temas preliminares, se da la introducción al tema, y se describe la problemática por la cual se está desarrollando el proyecto, así como las descripciones básicas de lo que se plantea hacer.

Se desglosa la investigación previa realizada describiendo diversos proyectos que se pueden encontrar en la literatura y en la red y que brindan una idea básica que seguir en el desarrollo de las diferentes etapas de la investigación. De igual forma se da el fundamento teórico que se debe tener para el entendimiento, desarrollo, aplicación y replicación del proyecto, pues al ser una investigación aplicada, debe basarse en la teoría hecha por investigaciones anteriores a ésta.

En el Capítulo 2 se da la metodología propuesta para la realización de los objetivos que se han planteado. La planificación de las tareas permitirá que éstas se desarrollen de una manera ordenada y sea más fácil detectar errores o cambiar de rumbo. Los métodos que se han planteado se presentan tanto de forma escrita o visual en manera de diagramas de flujo o de proceso.

El Capítulo 3 brinda las tareas que se han realizado, comenzando por la creación de prototipos y puesta en marcha de ellos para conocer las deficiencias y áreas de mejora entre las diversas herramientas que se utilizaron.

Se presenta la etapa de desarrollo electrónico de los sistemas, así como los algoritmos para la implementación de las diferentes aplicaciones de software que se diseñaron para cumplir con las tareas programadas para cada uno de ellos.

El desarrollo se plantea brindando al lector del presente documento una idea general, de manera que pueda extender el conocimiento posteriormente de acuerdo a las necesidades que se tenga. Los algoritmos se presentan de forma gráfica y en algunos casos escritos dependiendo de la complejidad de los programas desarrollado.

El Capítulo 4 da los resultados obtenidos una vez que se realizaron todas las pruebas tanto de hardware y de software necesarias para la validación de la investigación, obteniendo resultados satisfactorios en las tareas que se plantearon. Se cumplieron los objetivos

específicos y el objetivo general de manera apropiada, dejando un área de investigación muy amplia que pueda ser aprovechada.

Por último, en conclusiones y recomendaciones se encuentran los comentarios finales del autor, describiendo los conocimientos adquiridos y las problemáticas que se tuvieron en la realización de la investigación. Así mismo se hacen recomendaciones para mejorar lo desarrollado desde el punto de vista del autor y personas cercanas a la creación de este documento. Dado que la investigación fue satisfactoria, se recomienda a los lectores el uso del presente escrito de manera que pueda seguir adelante los proyectos descritos.

1.4 ESTADO DEL ARTE

1.4.1 MEDIOS ACTUALES PARA EDUCACIÓN A DISTANCIA

Actualmente con las nuevas herramientas y velocidades en internet se permiten formas más directas de transmitir la enseñanza o materiales didácticos, como lo son las audioconferencias y las videoconferencias, las primeras basan su comunicación en forma de mensaje auditivo en tiempo real mediante internet, las segundas son herramientas capaces de facilitar la comunicación en directo entre tutor y estudiante. Proporcionan un estado de presencia simulada, a pesar de encontrarse a kilómetros de distancia, no solo permite el intercambio verbal si no visual.

Numerosas instituciones han tratado de estructurar servicios a través de internet que permitan la educación mediante videoconferencias que han sido un éxito en la educación a distancia, al punto que las empresas comienzan a comercializar con cursos a distancia abriendo un mercado para tener mejores profesionistas y más competentes sin la necesidad de ir presencialmente a clases. Son entornos que no exigen generalmente conocimientos especializados en informática por lo que casi cualquier persona puede utilizarlos.

“Estas plataformas y entornos difieren uno de otros en la calidad de los recursos y experiencias que le brinden al usuario” (Aretio, 2001). No todas las plataformas virtuales de aprendizaje contienen interacciones físicas para llevar a la práctica por lo que se ha comenzado a trabajar en llevar esto a las nuevas plataformas de educación mediante la instrumentación virtual.

Es importante que las nuevas plataformas virtuales mantengan la esencia de la educación a distancia: que sean fáciles de manejar y que no requieran conocimientos informáticos avanzados para su uso.

1.4.1.1 Educación a distancia en México

La educación a distancia en México se está constituyendo como una herramienta importante que atiende temas educativos que no se completan en la enseñanza presencial. La demanda en la educación en el país hace que se tenga la necesidad de hacer cambios en los modelos tradicionales de enseñanza, sobre todo teniendo en cuenta la cobertura con la que se deben de realizar.

En México la educación a distancia ha crecido en los niveles de Educación Superior a través de diferentes formas. Esta dinámica ha afectado a diferentes ramas de la educación como la pedagogía, la tecnología, lo jurídico y lo económico (Zubieta, 2016).

Todo es un proceso que está llevándose a cabo actualmente y que demanda una reflexión y análisis para conocer las ventajas y desventajas que se puedan tener al momento de implementar este sistema de enseñanza.

La educación ha evolucionado en México desde hace muchos años, comenzando desde la creación de la Secretaría de Educación Pública que, debido al alto índice de analfabetismo en el país y la poca infraestructura para cumplir con las expectativas, se decide a realizar campañas para la enseñanza utilizando el correo como método de adquisición de conocimiento a personas con bajos recursos.

En los años sesenta, se evoluciona a lo conocido como enseñanza multimedia, donde sus principales fuentes de enseñanza fueron la radio y la televisión, presentes en la mayoría de los hogares. Para la enseñanza o contacto con el tutor se incorpora el teléfono, teniendo una retroalimentación y aumentando el contacto entre guía y estudiante.

Posteriormente, se transforma a la enseñanza telemática que comienza a finales de la década de los ochenta y está conformada por la integración de las telecomunicaciones con otros medios educativos. Se apoya en el uso más generalizado de las computadoras y sistemas multimedia. Se crearon aplicaciones con material didáctico que era interactivo con el sistema de cómputo que tuviese el alumno. Algunas de las aplicaciones contenían un apartado de autoevaluación rápido y preciso pues éste estaba llevado por un programa de computadora, permitiendo conocer el avance del alumno de mejor manera (Girón, 2005).

Actualmente con las nuevas herramientas y velocidades en internet se permiten formas más directas de transmitir la enseñanza o materiales didácticos, como lo son las audioconferencias y las videoconferencias, las primeras basan su comunicación en forma de mensaje auditivo en tiempo real mediante internet, las segundas son herramientas capaces de facilitar la comunicación en directo entre tutor y estudiante. Proporcionan un estado de presencia simulada, a pesar de encontrarse a kilómetros de distancia, no solo permite el intercambio verbal si no visual (Rubio, 2010).

1.4.2 INSTRUMENTACIÓN VIRTUAL EN LA ENSEÑANZA DE LA INGENIERÍA

Los sistemas de instrumentación virtual fueron creados para procesos industriales, sin embargo, se ha venido utilizando para la enseñanza en las universidades, creando laboratorios virtuales que permiten al alumno desarrollar la simulación de procesos de una manera más rápida que si se utilizaran instrumentos tradicionales por no tener la necesidad de cambiar los circuitos para la visualización de las variables.

Los laboratorios son un elemento clave en la formación integral y actualizada de un ingeniero. “No se puede concebir un ingeniero que no haya realizado prácticas de laboratorio en su trayectoria de formación inicial” (Rugeles, 2002).

Los avances tecnológicos de los últimos años han abierto posibilidades para cambiar la estructura de los laboratorios tradicionales, por una estructura apoyada en la computación, circuitos de acondicionamiento para la adquisición de datos y el software necesario para su procesamiento.

Se han implementado laboratorios virtuales utilizando herramientas como lo son *LabVIEW*®, pues se tiene infinidad de información tanto escritas como en la red para el desarrollo en esta plataforma, para fines didácticos y en diferentes ramas de la ciencia, cada uno con diferentes propósitos ya sea para aumentar la capacidad de los alumnos o para la actualización de los docentes.

Aguirre (Aguirre, 2011) implementó un laboratorio virtual con *LabVIEW*® para la actualización de los docentes y reducir el tiempo de horas presenciales que necesitaba un alumno para adquirir las competencias.

El laboratorio consta de dos partes, la primera en donde se encuentran todas las bases teóricas necesarias para el diseño lineal de controladores acorde al nivel del estudiante que utilizará el sistema, y la segunda el apartado de simulación y de control del proceso por medio de la computadora sin llegar a utilizar hardware.

Otro ejemplo de instrumentación virtual en la enseñanza es de Chiquillo (Chiquillo, 2008) quién desarrollo un sistema que permite el análisis de máquinas eléctricas del laboratorio de la universidad donde presta sus servicios. El sistema está sobre *LabVIEW*® y permite conocer los parámetros de las máquinas eléctricas como lo son velocidad, torque, eficiencia y factor de potencia.

Los laboratorios virtuales cada vez son más comunes en la enseñanza de la ingeniería, sin embargo, a medida que cambian los modelos educativos, éstos laboratorios deben ir evolucionando pues se necesita de la supervisión y puesta en marcha por parte de los profesores o los encargados además de que obliga a la presencia física del alumno.

Algunos laboratorios virtuales que se pueden encontrar actualmente tomando como fecha de búsqueda noviembre de 2018 son:

- *PID Controller Laboratory*, www.pidlab.com que se basa en *Applet*s de Java y se utiliza para el diseño, sintonía y análisis de sistemas de control proporcional, integral y derivativo.
- *Control System and Design* www.softintegration.com/webservices/control que permite realizar el análisis en el dominio del tiempo, la respuesta a diferentes señales de prueba de control, análisis en la frecuencia y gran número de prácticas relacionadas a control.

La evolución ha llevado a crear lo que se conoce como Laboratorio Virtual Remoto, que es una modificación al Laboratorio Virtual añadiendo al sistema computacional, instrumentación, control y acceso a equipos de laboratorio reales. “Ya no hablamos de llevar a cabo prácticas en un simulador, sino de realizar actividades prácticas de forma remota a través de intranet o internet” (Medina, 2011). Esto permite la transferencia de información entre un proceso real y los estudiantes de manera unidireccional o bidireccional.

Si bien ya se encuentran laboratorios virtuales y laboratorios remotos en operación, en México todavía hay poca difusión y es necesario vincular estos laboratorios con las instituciones educativas que estén acordes para utilizarlos. Entre los que se encuentran en línea se encuentran:

- *MOOC Lab*, que es un laboratorio público y de uso masivo, de circuitos y mediciones eléctricas, mismo que está integrado al curso MOOC Energía Eléctrica: Conceptos y Principios Básicos, disponible para los usuarios a través de la plataforma de MéxicoX y que se encuentra a cargo del Tecnológico de Monterrey,
- *Laboratorio de electrólisis*: Fue creado por la Universidad Nacional Autónoma de México con la finalidad de motivar a los alumnos a realizar prácticas desde internet. Se muestran medidas de seguridad y la metodología con la que deben de realizarse este tipo de pruebas químicas mediante una simulación virtual de este proceso que hay que seguir mediante los controles de la computadora.

Algunos ejemplos de laboratorios remotos a nivel internacional son:

- *Automatic Control Telelab* http://act.dii.unisi.it/introduction.php que es un sistema de adquisición de datos y que permite la observación remota en tiempo real.
- Laboratorio Remoto de Automática industrial http://lra.unileon.es/es Es un laboratorio remoto con acceso a equipos industriales, como una planta piloto para la realización de experiencias de control de operación y supervisión remota, maquetas de procesos de control sobre variables de nivel, caudal, temperatura y otros equipos de automatización.
- http://mechanical.poly.edu/research/control/RemoteLab.html Tiene diseño de controladores PID para un sistema de tercer orden construido con circuitos RC. Permite introducir el código de control diseñado por el usuario, basado en *LabView*®.

Uno de los inconvenientes en la extensión y creación de este tipo de laboratorios puede verse mermada por el costo que tienen las licencias de los programas que se utilizan para realizar las plataformas, como las de *LabVIEW*®*.*

En la última década el incremento en las plataformas *Open Source* se ha ido desarrollando, pero la instrumentación virtual no ha sido beneficiada de este desarrollo, por lo que es necesario comenzar a ver alternativas que puedan aplicarse para el diseño de los instrumentos.

Gupta (Gupta, 2014) propone alternativas de código libre que se ofrecen en la comunidad y en el diseño y desarrollo de instrumentación virtual, tanto de hardware como algunas que incluyen programas de desarrollo de software gratuito como Arduino®, Raspberry pi® y Beagle Board®.

El software libre debe tener la capacidad de acceder a los puertos de comunicación de una tarjeta de adquisición de datos para la manipulación de datos para procesarlas, graficarlas y analizarlas, todo esto con herramientas fáciles de manejar y con documentación de libre acceso. Algunos ejemplos que enumera Gupta son: Pylab-Works®, Scilab® y GNU-Octave.

1.4.3 RASPBERRY EN LA ELECTRÓNICA

Dentro del campo de la producción industrial, desde sus inicios hasta la actualidad, los sistemas automáticos han tenido una evolución importante pasando de ser una herramienta de trabajo opcional a una indispensable para poder competir en el mercado globalizado. La automatización permite aumentar la calidad de los productos ofrecidos y reducir el tiempo

requeridos para su producción y bajar los desperdicios creados por la inexactitud que pueden dar los operadores. Así mismo una de las funciones más importantes es el poder realizar trabajos complejos sin utilización de mano de obra realizado por operadores que pueda ponerse en riesgo aumentando la rentabilidad (Alciatore, 2008).

La automatización comienza con la introducción de mecanismos que permitían aumentar las capacidades de los seres humanos o separar las actividades necesarias para realizar una tarea. Sin embargo, con la aparición de la electrónica, se comenzaron a complementar los sistemas automáticos mediante dispositivos como actuadores, transductores, sensores y sistemas programables convirtiendo estos sistemas en sistemas mecatrónicos (Bolton, Mecatrónica, 2006).

Uno de los principales componentes de los sistemas de control presentes en la mecatrónica son los sistemas de adquisición de datos que es el encargado de recopilar información de forma automática provenientes de fuentes analógicas y digitales como lo pueden ser señales de sensores o cualquier carga que se tenga. Su funcionamiento es tomar variables mesurables y transformarlas en un mundo digital para que puedan ser leídas por las computadoras. Sin embargo, para leer dichas señales es necesario un acondicionamiento o llevarlas a niveles que puedan ser tomadas por los sistemas de adquisición (Areny, 2004).

Existen en el mercado sistemas de adquisición de datos como las fabricadas por National Instrument® que tienen un alto costo. Sin embargo, existen ejemplos de sistemas de adquisición de datos con otras plataformas como los microcontroladores PIC. Ejemplo de ello es el implementado por Calzada (Calzada, 2005) que desarrolló un sistema de adquisición para medir señales senoidales con el fin de procesarlas para su análisis, creando una alternativa de bajo costo y para una aplicación en específico. Dicho sistema está comunicado con señales USB al entorno LabView®, teniendo dependencia hacia el pago de licencia del software, pero reduciendo el costo al no utilizar las tarjetas creadas por el fabricante. Una de las ventajas de los sistemas PIC es que cuentan por defecto con entradas analógicas facilitando el desarrollo de sistemas de adquisición de datos (Calzada, 2005).

Debido a que la Raspberry Pi® no cuenta con puertos análogos es necesario el uso de plataformas que brinden de estas características al sistema. Velásquez (Velásquez, 2013) combinó la microcomputadora con un conversor análogo digital MCP3002 que se comunica con la tarjeta mediante protocolo SPI (*Serial Peripheral Interface*) con la finalidad de poder

leer sensores básicos, en el caso específico el LM35 que es muy utilizado en medición de temperatura de bajo costo (Velásquez, 2013).

Se han realizado otros ejemplos de diseños para la adquisición de datos utilizando tanto hardware como software libre utilizando la microcomputadora Raspberry Pi® con el fin de realizar sistemas domóticos. Este tipo de proyectos han tomado popularidad por el uso de los teléfonos inteligentes, ya que éstos se han vuelto una parte indispensable de la vida de las personas por la infinidad de posibilidades de tareas que pueden resolverse mediante ellos (LLedó, 2012).

La domótica se refiere a la aplicación de las tecnologías de la comunicación y el control para aplicaciones domésticas ya sea para el encendido de lámparas o para medir las variables de consumo eléctrico, agua y sistemas de alarmas. Sarthak (Sarthak, 2014), diseñó un sistema domótico que permite el control de las luces utilizando Raspberry Pi®, el sistema funciona utilizando un sistema de comandos para la comunicación mediante correos electrónicos por lo que no existe una interfaz visual para visualizar las variables del sistema (Sarthak, 2014).

Los sistemas que no cuentan con interfaz visual tienen la ventaja de ser relativamente más seguros ya que solamente el programador o desarrollador conoce el sistema de comandos necesarios para su aplicación. De igual forma existen desventajas en el uso de un sistema como el desarrollado por Sarthak, una de ellas es en el caso que en la vivienda habiten más personas con desconocimiento del mismo y no puedan manejar el sistema fácilmente. Barrera (Barrera, 2016) creó un sistema de medición de variables en el hogar utilizando la Raspberry Pi® en combinación con un microcontrolador ATMEGA6 con la finalidad de obtener las prestaciones de lecturas análogas con las cuales no cuenta por defecto la microcomputadora.

La interfaz visual fue programada utilizada el entorno Codesys ®, una de las plataformas más utilizadas en la automatización industrial por los estándares que utilizan en su entorno. Cuenta con una transmisión en vivo de cámaras para la seguridad, así como el control de cargas y visualización de diferentes variables que hacen un ambiente amigable para el usuario. Sin embargo, aunque se utiliza la Raspberry Pi® se plantea una plataforma con licencia para el sistema de visualización.

Si bien una de las aplicaciones más utilizadas para la Raspberry Pi® es la domótica, se ha utilizado en otras más específicas en el ámbito mecatrónico como lo es la robótica. Muños

(Muños, 2013) desarrolló un sistema robótico utilizando como interfaz visual lenguaje el entorno Java. Fue necesario realizar un sistema para el control de los motores del robot puesto que la tarjeta no cuenta con la etapa de acondicionamiento necesaria para manejar cargas de potencia.

Existen diferentes ejemplos de etapas de potencia para microcontrolados que también pueden aplicarse en la Raspberry Pi® como los propuestos por Yime (Yime, 2016) y Rojas (Rojas, 2012). El primero propone un sistema para control de motores de corriente directa con escobillas y el segundo presenta una tarjeta genérica para el uso de motores trifásicos utilizando electrónica convencional.

Dentro del campo del control automático existen plataformas similares a las planteadas en funcionamiento. Sin embargo, utilizan sistemas costosos y de gran tamaño que, si bien cumplen el objetivo del estudio de la dinámica de sistemas, no son replicables de manera sencilla. Ejemplo de ellos es el desarrollado por Chacón que construyó una planta de prueba para la identificación de un sistema térmico que utiliza como interfaz LabVIEW®.

En México se ha trabajado con la Raspberry Pi® debido al aumento en su popularidad con proyectos de código abierto y el bajo costo que conlleva la realización de proyectos con este mini computador.

Nuñez (Nuñez, 2014), desarrollo un sistema de adquisición de datos en la web con una base de datos que permitía medir señales de las variables más utilizadas en los hogares (temperatura, consumo eléctrico, activación de cargas) combinando dos elementos de código abierto como lo son Arduino® y la Raspberry Pi®. El proyecto constaba de una página web que contenía un registro de todas las mediciones que se habían hecho en el día y elaboraba un informe con el que el operador podía comparar el consumo de diferentes días y horarios.

Flores (Flores, 2018) por su parte optó por realizar un sistema de medición de la calidad de la energía eléctrica que entrega Comisión Federal de Electricidad mediante una aplicación móvil que permitiera realizar el monitoreo del consumo eléctrico y su calidad para los usuarios que estuviesen registrados en la base de datos. En este proyecto se optó por utilizar elementos externos de hardware como conversores análogos digital que permitieran a la Raspberry Pi® adquirir mediciones sin necesidad de otra plataforma de microcontrolador.

1.5 PLANTEAMIENTO DEL PROBLEMA

La evolución de los sistemas industriales que se ha generado gracias a los avances en las tecnologías de la información así como los sistemas mecatrónicos, da como resultado la necesidad de personal con habilidades en distintas áreas de conocimiento para crear equipos multidisciplinarios que puedan realizar proyectos de alta complejidad, entre estas disciplinas una de las más importantes es el control de proceso. (Ponsa, 2001).

Anteriormente la automatización estaba enfocada principalmente en áreas como la electrónica y la mecánica. Sin embargo, esto ha ido cambiando paulatinamente con la incursión de sistemas informáticos en el control de procesos, convirtiéndolos en sistemas mecatrónicos. Debido a ello es necesario combinar los circuitos electrónicos con herramientas de computación para crear sistemas más eficientes y puedan ser monitoreados remotamente. Este proceso de evolución en los sistemas ha permitido que los países crezcan industrialmente y la producción se realice en las condiciones más idóneas. Sin embargo, este proceso presenta un problema cuando la tecnología que se utiliza requiere un conocimiento especializado en áreas del control automático (Camargo, 2015).

Ya que los sistemas de control requieren de un conocimiento profundo de diferentes áreas para su aplicación y diseño, como la matemática y la física, su aprendizaje se vuelve complejo existiendo gran cantidad de información que, en la mayoría de las ocasiones, solo presenta la parte teórica sin llevarla a la práctica. Si bien existen herramientas que permiten el análisis de sistemas y facilitan los cálculos matemáticos, las más utilizadas tienen un costo de licencia que no es accesible para la mayoría de los estudiantes y pequeños desarrolladores.

El surgimiento de las plataformas Open Source (código libre) han permitido el desarrollo tanto de hardware como software que puede ser utilizado sin la necesidad de realizar pagos de licencia. Así mismo, existen otras ventajas en la utilización de estas herramientas, como lo son las diferentes comunidades que comparten su conocimiento con la finalidad de ayudar a otros a crear nuevos proyectos basados en los ya creados reduciendo tiempos de desarrollo e información gratuita.

Actualmente se cuenta con una alternativa como la Raspberry Pi®(RPi), un mini ordenador desarrollado en Reino Unido con la finalidad de llevar herramientas informáticas de bajo costo a las escuelas de esta área. Debido a sus características, esta plataforma se ha

utilizado en el desarrollo de aplicaciones electrónicas que dependen de una computadora como lo son los sistemas domóticos y plataformas de monitoreo remoto. La tarjeta cuenta con entradas y salidas de propósito general (GPIO) digitales que la hacen versátil en la implementación de sistemas automáticos con lo que puede utilizarse en la lectura de sensores con funcionamiento “todo o nada” (Richardson, 2013).

Sin embargo, al solo tener puertos digitales no es posible la lectura de sensores analógicos cuya señal varía con el tiempo pues no tiene las características necesarias para obtener lecturas dinámicas. La falta de elementos de hardware como los convertidores análogo digital y digital – análogo son un problema cuando se pretenden realizar sistemas de adquisición de datos que permitan visualizar y procesa la información de la planta que se desea automatizar.

Así mismo, otro de los inconvenientes que se tienen en la utilización de esta plataforma, es el nivel de voltaje con el que trabaja. Los niveles aceptables de tensión para la tarjeta son de 0 a 3.3V siendo posible solamente la utilización de cargas que no sobrepasen un nivel de corriente de 50mA. Lo anterior limita el uso de la tarjeta con sus características brindadas de fábrica cuando las cargas a manejar requieren de una potencia más elevada.

Ya que la tarjeta no está diseñada para cargas de potencia o actuadores, no cuenta con protecciones que permitan reducir el riesgo de fallas o daño del equipo, como lo es el aislamiento óptico de las entradas de propósito general, así como las salidas que llevan a cargas de alto consumo en corriente. Para brindar estas funciones es necesario colocar un acondicionamiento en las entradas y salidas que permitan la interacción de este dispositivo con los diferentes elementos del sistema de control que se pretende realizar.

De acuerdo a lo anterior, es necesario crear plataformas que permitan la adquisición de conocimientos teóricos en el área de control de procesos con la finalidad de complementar las competencias de los desarrolladores que quieran crear sistemas automáticos y que éstas sean lo más económicas y accesibles posibles.

Si bien la RPi es una herramienta que puede ayudar en el desarrollo de sistemas de control de gran complejidad y bajo costo, es necesario el diseño de circuitos que permita introducirla y comunicarla con los elementos de hardware presentes en el sistema. Esto da la facilidad a desarrolladores con bajo presupuesto crear tecnología propia y disminuir la dependencia tecnológica que se tiene con los softwares que actualmente se encuentran en el mercado.

1.6 JUSTIFICACIÓN

Actualmente los sistemas mecatrónicos tienen un alto costo y complejidad de implementación pues está cerrado a un grupo de especialistas en la materia que dominen las plataformas más comunes en el mercado cerrando el camino a desarrolladores con conocimientos en otras áreas que puedan ser de utilidad. Por ello se han comenzado a utilizar otras ramas del conocimiento en la automatización como los sistemas con microcomputadoras como la Raspberry Pi® (Valera, 2014).

Las microcomputadoras utilizan frecuencias de trabajo muy por encima de los sistemas que se utilizan actualmente en la automatización, que pueden ser aprovechadas para mejorarlos y hacerlos más robustos y complejos. Así mismo, tienen la capacidad de realizar el control de cargas utilizando los periféricos contenidos en estos dispositivos como sus entradas y salidas digitales que, realizando modificaciones y circuitos de acoplamiento de señales, adquieren las características que pueden presentar otras plataformas comunes como los controladores lógicos programables (PLC) (Velásquez, 2013).

Al ser herramientas de software y hardware libre, se pueden evitar costos de licencia que, en la mayoría de las ocasiones, son innecesarias, aumentando la posibilidad de que personal sin algún sustento económico puedan incursionar en el diseño de sistemas mecatrónicos. De igual forma la mayoría de las microcomputadoras utilizan lenguajes de programación de fácil aprendizaje y basta documentación por la existencia de comunidades en la web que comparen sus conocimientos y avances, siendo una de las principales políticas y características de las plataformas Open Source con la finalidad de facilitar la tarea de desarrollar software y hardware (Stallman, 2014).

Una de las principales ventajas del uso de sistemas de microcomputadoras de software libre es la independencia tecnológica. La independencia tecnológica es la no necesidad de software de terceros o diseños de otros sistemas, es decir, no trabajar un dispositivo o programa específico, siendo posible otras opciones que puedan ser una mejor alternativa ya sea tanto en costo como en complejidad de desarrollo. Con ello se brinda al usuario la capacidad de cambiar el hardware y software de la manera que más convenga sin el permiso o realizar violaciones de derechos de autor. De igual forma este tipo de sistemas es capaz de realizar instrumentos virtuales remotos dentro de ellos al poder realizar sistemas web (Arriola, 2008),

Si bien se puede pensar en aplicaciones de tipo industrial, el desarrollo de plataformas con control remoto puede llevarse a otras áreas como puede ser la educación para ser aprovechadas en la enseñanza a nivel superior.

Siendo una de las problemáticas, aún en sistemas presenciales, la falta de prototipos, para un estudiante que se encuentre en una plataforma virtual resulta casi imposible obtener algún proceso físico que permita llevar sus conocimientos a la práctica, por lo cual es necesario en materias teórico – prácticas desarrollar nuevas alternativas con las cuales se permita interactuar a los estudiantes de estas plataformas con equipos que se encuentren en alguna institución educativa que cuente con ellos.

Con la extensión del uso del internet en casi todos los lugares donde se cuente con una computadora y el aumento de herramientas que permitan realizar transmisiones en vivo desde un lugar remoto, la educación a distancia debe evolucionar y no solo dar a los estudiantes los conocimientos teóricos sino también el darse cuenta que éstos funcionan en un mundo real, completando una parte importante del sistema de educación por competencias.

Aprovechando una de las nuevas herramientas multimedia en el internet como es el envío de imágenes, se pueden tener sistemas que permitan la interacción de una manera visual en un tiempo casi real, con retardos muy pequeños de tiempo.

Se pretende con esta investigación buscar maneras alternativas de diseño y programación que sean de uso libre y bajar el costo, siendo accesibles para casi cualquier institución educativa y para el usuario final.

1.7 OBJETIVO GENERAL

Desarrollar los elementos de hardware para la implementación de prácticas de control de procesos utilizando Raspberry Pi®.

1.7.1 OBJETIVOS ESPECÍFICOS

- Realizar interconexiones entre la Raspberry Pi® y otros dispositivos para brindarle características para la adquisición de señales y control de cargas.
- Diseñar e implementar un sistema de control de temperatura.
- Diseñar e implementar un sistema para el control de posición de un levitador de aire.
- Diseñar e implementar un control de posición para un motor de C.D.
- Programar un instrumento virtual remoto para el manejo de los sistemas propuestos.

1.8 FUNDAMENTO TEÓRICO

1.8.1 SISTEMAS DE CONTROL

El control de procesos es el área de la automatización que permite mantener un proceso o sistema en las condiciones deseadas llamadas condiciones de referencia. Dorf (Dorf, 2005) define un sistema de control como “una interconexión de componentes que se unen entre sí para tener una respuesta deseada”. Los sistemas de control ayudan a estabilizar las variables físicas que se tengan en algún proceso como lo pueden ser temperatura, presión, flujo, nivel etc.

Cuando se diseñan sistemas de control no es necesario conocer cómo funcionan sus elementos de manera interna o la interacción entre ellos, solo es necesario conocer la relación que existe entre su señal de entrada y la señal de salida. Los sistemas pueden clasificarse en lineales y no lineales (Gomáriz, 2001).

Los sistemas que se consideran lineales son los que se pueden resolver utilizando el principio de superposición. Es decir, cuando el sistema puede dividirse en subsistemas y su solución sería la suma de los resultados de cada uno. Por lo tanto, la respuesta a varias entradas puede calcularse resolviendo una entrada a la vez y sumando sus resultados. Un sistema no lineal en cambio es aquel que se caracteriza utilizando ecuaciones diferenciales que no son lineales y el principio de superposición no es aplicable. La mayoría de los sistemas físicos no son lineales, pero pueden linealizarse a tramos (Ogata, 2010).

Los sistemas de control se basan en una planta o proceso que se define como la realización de tareas de manera sucesiva que se coordinan para realizar una acción. Existen dos formas básicas en sistemas de control: los sistemas de lazo abierto y los sistemas de lazo cerrado.

Un sistema en lazo abierto no tiene retroalimentación, es decir, no se tiene conocimiento de cuál es el error que se presenta entre la lectura actual y la referencia que se propone. En este caso el control se realiza de manera experimental y en base a la experiencia del diseñador en sistemas similares por lo que es difícil llegar al punto de consigna.

En cambio, los sistemas a lazo cerrado se caracterizan por tener una retroalimentación de la medición desde la salida hasta la entrada mediante un elemento de medición, lo que permite calcular el error que se tiene en la planta (Bolton, Ingeniería de control, 2001).

La Figura 1 muestra los componentes y ubicación básicos para el diseño de un sistema de control en lazo cerrado.

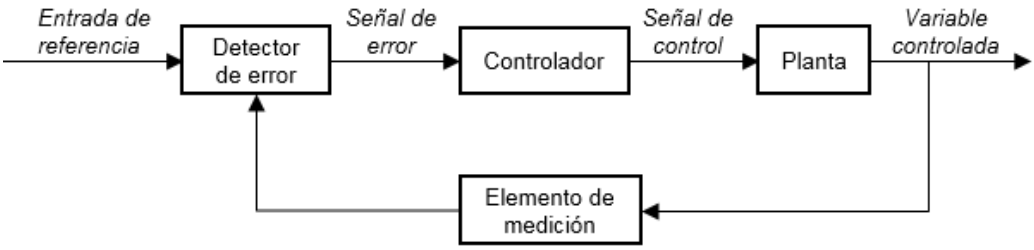

Figura 1. Elementos básicos de un sistema de control en lazo cerrado.

De acuerdo a la Figura 1 se pueden definir los diferentes elementos que componen a los sistemas de lazo cerrado (García L. , 2009).

- Entrada o referencia: Es el valor al cual el sistema debe de llegar en su valor final.
- Detector de error: Calcula el error del sistema comparando la señal de entrada con la de salida. El error puede definirse como la diferencia en valor absoluto entre el valor obtenido y el valor esperado.
- Controlador: Define que acción debe de tomar el sistema de acuerdo a la señal de error. En él se realizan las operaciones matemáticas y tratamiento de la señal para llegar el punto de consigna.
- Planta o proceso. Corresponde al sistema que se va a controlar. Para diseñar un controlador es necesario conocer el comportamiento de la planta, el cual está relacionado con su función de transferencia.
- Elemento de medición: Es el instrumento encargado de llevar la señal de retroalimentación de la salida. Se encarga de medir la variable de proceso.
- Variable controlada: Corresponde a la salida que ha sido analizada y corregida para llegar al punto propuesto por la señal de entrada.
- Perturbación: Señal externa o interna que afecta directamente al sistema.

Una de las principales ventajas que tienen estos tipos de sistemas es que la retroalimentación permite que sea menos sensibles a las perturbaciones externas y la variación que se presenta en los parámetros interiores si se comparan con los sistemas a lazo abierto.

1.8.2 MODELADO MATEMÁTICO Y FUNCIÓN DE TRANSFERENCIA

Uno de los aspectos que más importancia tienen en la ingeniería de control es la capacidad de representar los fenómenos físicos utilizando matemáticas para poder analizar el sistema y determinar cuáles son sus características de comportamiento y las limitaciones que tiene. De acuerdo a cervantes (Cervantes, 2015) un modelo matemático es la unión de las ecuaciones diferenciales con las que se puede representar al sistema y que se aproximen más a su comportamiento real. Las ecuaciones son obtenidas utilizando los métodos para resolución de los fenómenos físicos que se encuentran presentes en el sistema.

La función de transferencia de un sistema es el cociente o la relación que existe entre la entrada y la salida y puede representarse utilizando la Ecuación (1). Comúnmente se expresa utilizando la transformada de la Laplace.

$$G_{(s)} = \frac{\mathcal{L}[salida]}{\mathcal{L}[Entrada]} \tag{1}$$

Regularmente en la representación de la función de transferencia se obtiene un polinomio al cual se le conoce como polinomio característico y su forma en la transformada de Laplace es la que se ve en la Ecuación (2). Se dice que un sistema es realizable cuando el orden del denominador es mayor o igual al orden que se tiene en el numerador (Daprotis, 2011).

$$\frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \tag{2}$$

Los sistemas pueden tener diferentes representaciones, por lo que puede haber diferentes representaciones matemáticas que pueden ser igualmente válidas dependiendo de la perspectiva de quien obtuvo las ecuaciones. Cuando se obtiene un modelo matemático es necesario tomar en cuenta qué simplicidad se requiere, así como la precisión del mismo. Si se tiene un modelo muy básico se están ignorando algunas características del sistema y puede ocasionar errores entre el modelo y la planta real (Ogata, 2010).

1.8.3 RESPUESTA DINÁMICA DEL SISTEMA

La respuesta dinámica del sistema es aquella en que los valores de la magnitud son variables en el tiempo, es decir la que se presenta antes de que el sistema o la planta lleguen a su estado estable. Cuando ya se ha obtenido el modelo matemático existen varios

métodos para analizar el comportamiento del sistema. Para ello es necesario utilizar una señal de entrada como prueba y existen varios tipos de acuerdo a la entrada que pudiera tener el sistema en la realidad (Kamen, 2008).

Las señales de prueba que más se utilizan son regularmente: Función escalón, rampa e impulso. Al utilizar este tipo de señales al hacer las pruebas, se puede analizar matemáticamente el sistema para aproximar la respuesta en el tiempo pues son funciones muy simples y que se tiene conocimiento de su comportamiento.

La respuesta de un sistema que no varía en el tiempo se puede descomponer en dos partes: la respuesta dinámica o transitoria y la respuesta estacionaria mediante la Ecuación (3).

$$c(t) = ct(t) + css(t) \tag{3}$$

Donde:

- $ct(t) = Respuesta\ transitoria$
- $css(t) = Respuesta\ estacionaria$

La respuesta transitoria se origina por la dinámica del sistema y es la que determina el comportamiento de la salida cuando se ha aplicado la entrada hasta que ésta se estabiliza. La respuesta estacionaria es la respuesta que se obtiene cuando el sistema ha llegado a su estado estacionario siempre y cuando el sistema lo sea (Nise, 2008).

1.8.4 SISTEMAS DE PRIMER ORDEN

De acuerdo a (Ruíz, 2013) en los sistemas de primer orden la relación que existe entre la entrada y la salida o también llamada función de transferencia puede obtenerse mediante la Ecuación (4).

$$\frac{C(s)}{R(s)} = \frac{k}{\tau s + 1} \tag{4}$$

En donde:

K= Ganancia del sistema

τ = Constante de tiempo del sistema. Representa el tiempo en el cual la salida del sistema alcanza aproximadamente el 63% de su valor final.

Si se utiliza la función matemática el valor final es alcanzado en un tiempo infinito. Sin embargo, en la realidad lo hace en un tiempo establecido. En la práctica se considera que

el sistema alcanza el 98% de su valor final en cuatro constantes de tiempo y el 95% en tres constantes de tiempo (Boylestad, Introducción al análisis de circuitos, 2011).

Se puede representar la respuesta transitoria de un sistema de primer orden a una entrada escalón con una curva exponencial similar a la que se muestra en la Figura 2.

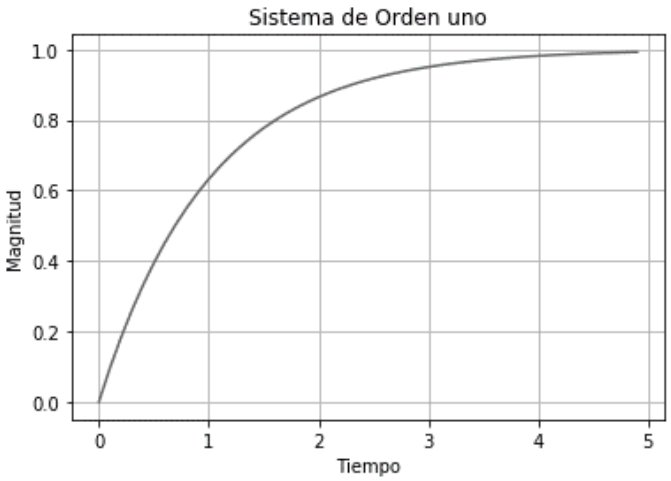

Figura 2. Respuesta dinámica de un sistema de primer orden a una entrada escalón.

1.8.5 SISTEMAS DE SEGUNDO ORDEN

Los sistemas de segundo orden poseen dos polos en la función de transferencia del sistema. Según Gaviño (Gaviño, 2010) la ecuación de un sistema de segundo orden es la que se muestra en la Ecuación (5)

$$\frac{C(s)}{R(s)} = \frac{\omega n^2}{s^2 + 2\xi\omega n + \omega n^2} \tag{5}$$

En donde:

- ξ = Factor de amortiguamiento
- ωn= Frecuencia natural

Existen diferentes tipos de sistemas que dependen del valor del factor de amortiguamiento ξ que se describen a continuación.

- Para un $\xi = 0$: Sistema oscilatorio.
- Para un $0 < \xi < 1$: Sistema subamortiguado.

- Para un $\xi = 1$: Sistema con amortiguamiento crítico.
- Para un $\xi > 1$: Sistema sobreamortiguado.

En la Figura 3 se puede observar gráficamente el comportamiento de los sistemas de segundo orden de acuerdo al valor que se tiene en el factor de amortiguamiento.

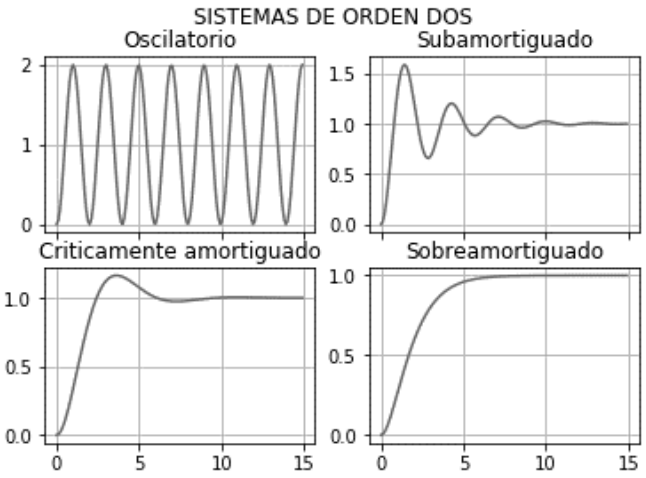

Figura 3. Representación gráfica de un sistema de segundo orden.

Las características en los sistemas de control regularmente se representan en el dominio del tiempo y en respuesta a una señal de escalón unitario pues es la más utilizada en la práctica. La respuesta transitoria de un sistema de orden superior presenta oscilaciones antes de llegar a su estado estable. Para representar un sistema transitorio es común obtener las siguientes especificaciones con el fin de analizar su comportamiento (Joglar, 2017).

- Tiempo de retardo (t_d): Es el tiempo requerido para que la salida alcance por primera vez la mitad de su valor en estado estacionario.
- Tiempo de subida (tr): Corresponde al tiempo que se necesita para que el sistema llegue por primera vez a su valor final.
- Tiempo de pico (tp): Es el tiempo requerido para que el sistema llegue a su valor máximo o valor pico.
- Máximo sobre impulso (Mp): Corresponde al valor máximo al que llega la respuesta del sistema y es la diferencia entre el valor máximo y el valor final de la curva. Se suele medir en base a porcentaje.

- Tiempo de asentamiento (ts): Es el tiempo necesario para que la curva alcance un valor cercano al valor de establecimiento y en términos de porcentaje se puede medir del 2% o 5%.

Las especificaciones mencionadas anteriormente son de vital importancia pues la mayoría de los sistemas se analizan utilizando como referencia su respuesta temporal. La Figura 4 muestra de forma gráfica cada una de las características anteriores.

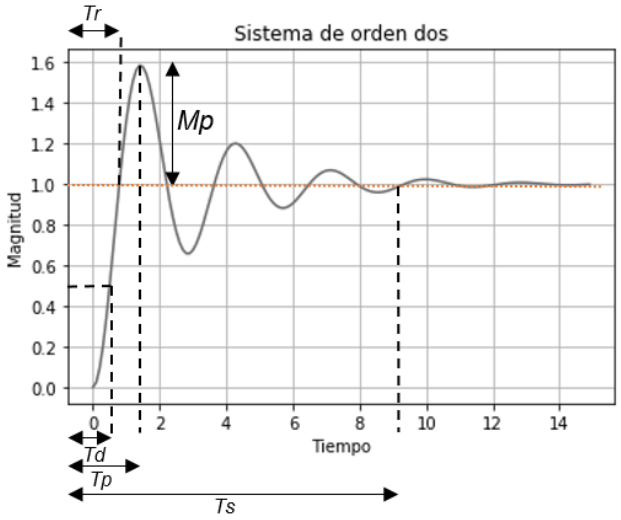

Figura 4. Características de los sistemas en su respuesta temporal.

1.8.6 MÉTODO DE ALFARO PARA IDENTIFICACIÓN DE SISTEMAS

Es posible obtener el modelo matemático de un sistema a partir de métodos experimentales siempre y cuando se pueda tener la curva de respuesta del sistema a una señal de entrada. Por lo regular los sistemas de control pueden ser modelados mediante sistemas de primer orden más tiempo muerto o segundo orden más tiempo muerto. Los métodos más utilizados para obtener los parámetros de los modelos descritos es el conocimiento de algunos puntos de la curva de respuesta al escalón (Ruíz, 2013).

Alfaro propone distintos modelos con retardo para una aproximación del modelo de la planta. El más sencillo para analizar se denomina Sistema de Primer Orden Más Tiempo Muerto (SPOMTM).

En la respuesta normal de los SPOMTM se presentan características similares a los que se pueden encontrar en un sistema de orden uno puro, con la diferencia que la señal se encuentra con un retraso respecto a la señal que se ha aplicado en la entrada conocido

como tiempo muerto. (Márquez Rubio, 2010). El sistema de primer orden más tiempo muerto se representa por la función de transferencia que se muestra en la Ecuación (6), donde k es la ganancia, tm el tiempo muerto y τ la constante de tiempo.

$$G_{p1}(s) = \frac{ke^{-tms}}{\tau s + 1} \tag{6}$$

La respuesta dinámica a una entrada escalón es la que se puede observar en la Figura 5 . Se puede observar que es muy parecida al sistema de primer orden. Sin embargo, contiene un retardo cuando se coloca la señal de prueba.

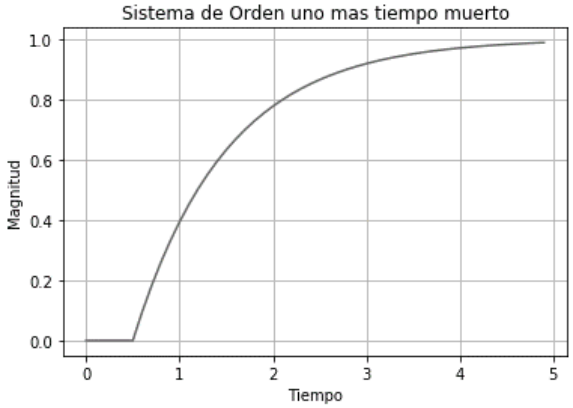

Figura 5. Respuesta al paso escalón de un SPOMTM.

De acuerdo a Alfaro (Alfaro V. M., Método de identificación de modelos de orden reducido de tres puntos 123c, 2007) se requieren dos puntos sobre la curva para identificar un SPOMT que corresponden a los valores de tiempo t_{25} y t_{75}, que representan el tiempo en el que el sistema llega al 25% y 75% de su magnitud total en estado estacionario. Una vez obtenidos estos parámetros se puede identificar el modelo con las ecuaciones (7) - (10).

$$k = \frac{\Delta y}{\Delta u} \tag{7}$$

$$\tau = 0.9102(t75 - t25) \tag{8}$$

$$t_m = 1.2620t_{25} - 0.2620t_{75} \tag{9}$$

Donde Kp corresponde al cociente entre el incremento de la señal de entrada y el incremento en la señal de salida. Si se requiere una aproximación para un sistema de segundo orden es posible utilizar su modelo con retardo cuya función de transferencia está descrita por la Ecuación (10).

$$G_{p2}(s) = \frac{ke^{-t'ms}}{(\tau' s + 1)^2} \quad (10)$$

De la misma manera que en el sistema de primer orden, se requiere conocer los tiempos en la que la respuesta alcanza tanto el 25% como el 75% sin embargo, los nuevos tiempos son calculados con las ecuaciones (11) (12)

$$\tau' = 0.5776(t_{75} - t_{25}) \quad (11)$$

$$t'm = 1.5552t_{25} - 0.5552t_{75} \quad (12)$$

1.8.7 TIPOS DE CONTROLADORES

Los controladores son elementos que se utilizan en los sistemas para hacerlos más eficientes y obtener el funcionamiento que el diseñado desea en la respuesta transitoria y en su estado estacionario.

La primera forma en la que se pueden alterar las características de la respuesta de cualquier sistema es el cambio en su ganancia. Sin embargo, al hacer esto se tiene una respuesta insatisfactoria pues otros parámetros se pueden ver afectados, siendo necesario el uso de otras herramientas para obtener la respuesta deseada. Estas herramientas dan como resultado los tres tipos más comunes de control (García L. , 2009):

- Control Proporcional-Integral (PI).
- Control Proporcional-Derivativo (PD).
- Control Proporcional-Integral-Derivativo (PID).

1.8.7.1 Control Proporcional: P

Los controladores proporcionales son aquellos en donde su salida y(t) es directamente proporcional a la entrada, que en el caso de un sistema a lazo cerrado corresponde a la señal del error siendo su función de transferencia la que se muestra en la Ecuación (13).

$$G(s) = \frac{Y(s)}{E(s)} = K_p \quad (13)$$

Las variaciones en la ganancia pueden ser aceptables cuando se está en el régimen transitorio. Sin embargo, en la respuesta estabilizada se tiene un gran error. Los

controladores de tipo proporcional son de fácil implementación y ajuste, pero no son recomendados cuando no están acompañados de otro tipo de herramienta de control como lo son el control derivativo e integral, pues no es posible llegar al punto de referencia (Gaviño, 2010).

1.8.7.2 Control Integral: I

En el control integral la salida del sistema es proporcional a la integral del error, la ganancia integral (Ki) está expresada por la Ecuación (14). La salida obtenida del sistema para cualquier instante en la respuesta temporal es proporcional a la suma de los errores que han pasado (Ogata, 2010).

$$v(t) = K_i \int e(t)dt \tag{14}$$

De acuerdo a Lozano (Lozano, 2012) en cualquier tipo de sistema de controladores, la ganancia proporcional es la más importante, por lo que es conveniente que las constante integral pueda escribirse en términos de Kp siendo representada por la Ecuación (15), donde Ti es el tiempo de integración.

$$G(s) = \frac{V(s)}{E(s)} = \frac{K_i}{s} = \frac{K_p}{T_i s} \tag{15}$$

El control integral se utiliza para eliminar el error en estado estacionario. Sin embargo, se crea una oscilación en la respuesta por la tendencia a querer eliminar el error.

1.8.7.3 Control Derivativo: D

Los controles derivativos son aquellos en donde su salida es proporcional a la derivada del error del sistema como se muestra en la Ecuación (16).

$$v(t) = K_d \frac{d\, e(t)}{dt} \tag{16}$$

Donde Kd corresponde a la ganancia derivativa. Si se escribe en términos de Kp por lo anterior mencionado, Kd se representa por la Ecuación (17).

$$K_d = K_p\, T_d \tag{17}$$

El tiempo derivativo es un factor de proporcionalidad y el equivalente en el dominio de la frecuencia del controlador de tipo derivativo es el que se muestra en la Ecuación (18).

$$G(s) = \frac{V(s)}{E(s)} = K_d s = K_p\, T_d s \tag{18}$$

El control derivativo es excitado por la rapidez con la que el error va cambiando, por lo que es posible corregir el error antes de que éste se incremente. Se dice que la acción derivativa predice el error porque se adelanta a su tendencia. Al igual que el control proporcional, debe actuar junto con otra acción pues no se ajusta a errores en el estado estacionario (Jimenez, 2014).

1.8.7.4 Control Proporcional-Integral-Derivativo: PID

Dado que los controladores proporcional, integral y derivativo que existen en forma independiente no pueden llegar a un nivel óptimo en el punto de referencia y que, en muchas ocasiones, no se tiene el modelo matemático de la planta a controlar para hacer los cálculos y tener un diseño analítico del controlador, se opta por la fusión de los tres controladores conocido como control PID, que es el algoritmo más utilizado en el control automático (Aström, 2009).

El control PID es aquel en que la salida del controlador es proporcional al error agregando la suma de los errores anteriores y la tendencia del error y se expresa mediante la Ecuación (19).

$$G(s) = \frac{V(s)}{E(s)} = K_p(1 + \frac{1}{T_i s} + T_d s) \tag{19}$$

1.8.8 MÉTODO DE ZIEGLER-NICHOLS PARA LA SINTONIZACIÓN DE CONTROLADORES PID

El primer método que se propuso para realizar la sintonización de controladores tipo PID fue desarrollado por Ziegler y Nichols (Ziegler & Nichols, 1942). Este procedimiento se basa

en el tazo de una recta tangente al punto de inflexión de la curva de respuesta del sistema al que se va a colocar el controlador.

El tiempo que transcurre entre la aplicación de la entrada y el punto en el que la tangente corta con el eje del tiempo se le conoce como tiempo muerto del sistema. El tiempo que pasa entre este instante y en el que la recta tangente toca con el valor final de la curva de respuesta corresponde a la constante de tiempo (Alfaro V. , 2006). En la Figura 6 se observa el trazo de la recta tangente sobre el punto de inflexión así como los puntos de la curva requeridos para la obtención del modelo. La función de transferencia está dada por la Ecuación (20).

$$G_{p1}(s) = \frac{k_p e^{-Ls}}{Ts + 1} \tag{20}$$

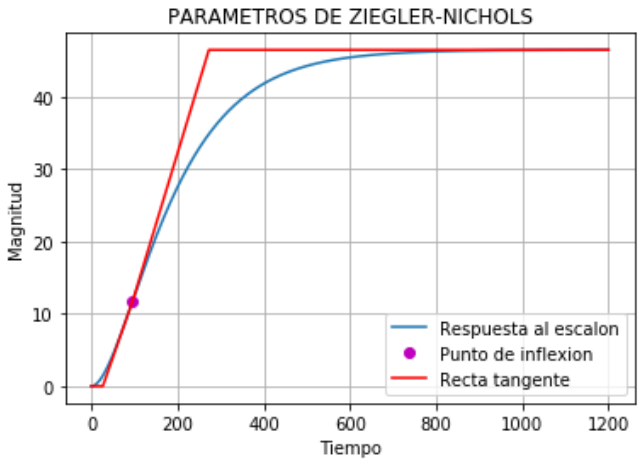

Figura 6. Parámetros de Ziegler-Nichols.

Con los valores obtenidos se procede a calcular los valores de las ganancias o tiempos del controlador. Las ecuaciones para ello son las que se muestra en la Tabla 1.

Tabla 1. Valores de sintonización para Ziegler-Nichols.

Tipo de Controlador	Kp	Ti	Td
Proporcional	$\frac{T}{L}$	∞	0
Proporcional – Integral	$0.9\,\frac{T}{L}$	$\frac{L}{0.3}$	0
Proporcional – Integral - Derivativo	$1.2\,\frac{T}{L}$	$2L$	0

Para obtener el punto de inflexión de una función (que se entiende por el momento en el que ésta pasa de ser convexa a cóncava) es necesario igualar a cero la segunda derivada de la respuesta temporal a una entrada escalón y posteriormente encontrar el máximo de la derivada de la función original. El punto de inflexión se obtiene evaluando en la función original el valor del tiempo obtenido (Gaviño, 2010).

1.8.9 ADAPACIÓN DE LAS REGLAS DE ZIEGLER-NICHOLS PARA UN SPOMTM

Debido a que es difícil determinar en donde se encuentra el punto de inflexión sobre la curva si no se tienen los datos precisos para realizar un análisis matemático, pues por lo general la señal se encuentra ruidosa o contiene perturbaciones que dificultan los cálculos, se busca identifica el sistema por otros métodos que proporcionan el modelo de la planta. Sin embargo, el método original propuesto por Ziegler-Nichols no está diseñado para operar con ellos pues fue descrito antes de que aparecieran las nuevas tecnologías, por lo que al sintonizar con él los resultados no son los esperados (Alfaro V. , 2006).

González (González, 2015) propone las ecuaciones necesarias para realizar las adaptaciones a los parámetros del método de Ziegler–Nichols para que éstos sean utilizados usando un modelo de primer orden con retardo obtenido mediante dos puntos sobre la curva de reacción. Con estas ecuaciones se pueden conseguir índices de desempeño muy aproximados al que se consiguen mediante el criterio original.

Las ecuaciones modificadas para obtener las ganancias y tiempos para el controlador PID propuestas se describen en las ecuaciones (21) -(23).

$$K_p = \frac{1.6632\tau + 0.2229}{K(0.7345t_m - 0.0020)} \tag{21}$$

$$T_i = 1.4690t_m - 0.0040 \tag{22}$$

$$T_d = 0.3672t_m - 0.0010 \tag{23}$$

1.8.10 SISTEMAS DE CONTROL EN TIEMPO DISCRETO

Los sistemas de tiempo discreto son aquellos en donde las variables tienen solamente un valor en determinado instante de tiempo. Cada instante de tiempo recibe el nombre de muestreo y éstos se representan utilizando secuencias. Las secuencias especifican en qué momento se ha realizado la muestra del sistema o se ha tomado un dato de un sistema

digital (García L. , 2009). En la Figura 7 se muestra un ejemplo de gráfica de una señal que se encuentra en tiempo discreto.

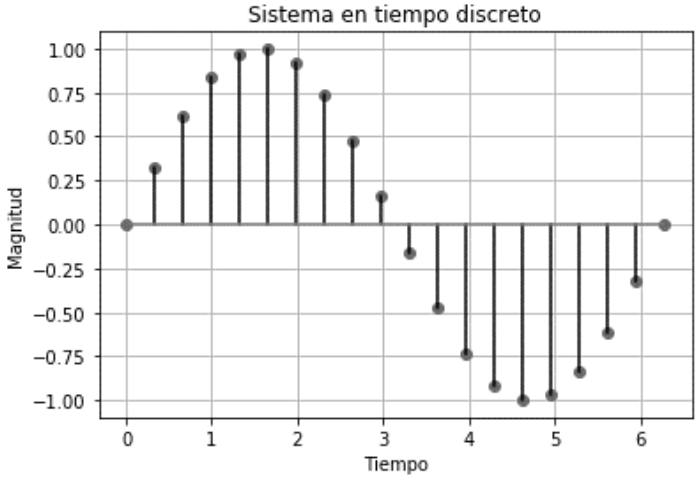

Figura 7. Ejemplo de gráfica en tiempo discreto.

En los sistemas de tiempo continuo su respuesta el análisis, cuando se analizan sistemas lineales invariantes en el tiempo, se lleva a cabo utilizando las ecuaciones diferenciales mediante la Transformada de Laplace, para convertir del dominio temporal al dominio de la frecuencia. En los sistemas digitales o discretos se utilizan las ecuaciones en diferencias y la herramienta para su resolución se conoce como Transformada Z. Es posible realizar la conversión entre sistemas de tiempo continuo y sistemas en tiempo discreto (Kuo, 2008). El diagrama a bloques básico para un sistema de control discreto según Kuo es el que se muestra en la Figura 8.

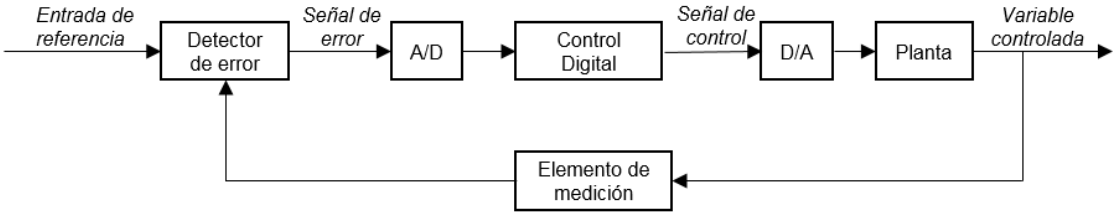

Figura 8. Diagrama a bloques básico de un sistema de control digital.

El diagrama presenta una similitud con un sistema en tiempo continuo. Sin embargo, contiene dos elementos indispensables en un sistema digital que son: el conversor análogo digital (A/D), que se encarga de convertir una señal del tiempo continuo en discreto para

poder aplicar los algoritmos de control digital, y el conversor digital análogo (D/A) que realiza la operación contraria, convertir del tiempo discreto a una señal continua con el fin de ser aplicada a la planta (Dormido, 2008).

1.8.11 MICROCONTROLADORES

1.8.11.1.1 Estructura del microcontrolador.

Un microcontrolador es un circuito integrado fabricado a alta escala de integración, es decir, está fabricado con componentes de tamaño microscópico unidos para formar uno solo. Incorpora todos los componentes de un controlador y un procesador, pero con limitadas prestaciones para realizar una tarea. La diferencia entre un microcontrolador y un microprocesador es que este último requiere de periféricos externos para realizar las tareas que le son asignadas como memoria, conversores, módulos de entradas y salidas etc; mientras que el microcontrolador contiene todos estos elementos en un solo encapsulado (Valdes, 2008).

De acuerdo a Valdés los componentes principales de un microcontrolador son los siguientes:

- Procesador o unidad central de procesos.
- Memoria RAM para el almacenamiento de datos.
- Memoria para el programa (ROM/EEPROM/Flash).
- Módulos de entradas y salidas para comunicación con el exterior.
- Módulos de control de periféricos (temporizadores, puertos de comunicación, interrupciones, mundo analógico).
- Generador de pulsos de reloj para sincronización del sistema.

1.8.11.1.2 Herramientas para el desarrollo de sistemas con microcontrolador.

Las herramientas de desarrollo con un conjunto de programas informáticos e interfaces conocidas como IDES, que permiten la realización de los proyectos de la manera más práctica para el usuario (Palacios, 2009). De acuerdo a Palacios las principales herramientas de ayuda para los sistemas que se desarrollan en microcontroladores son las que se describen a continuación.

- *Ensamblador*: Es el lenguaje máquina, se basa palabras nemotécnicas que permiten dar instrucciones al microcontrolador y que después será traducida a codificación binaria que será almacenada en la memoria Flash del microcontrolador. La programación en este lenguaje es complicada para los principiantes, pero permite el acceso de forma más eficiente a los diferentes registros que se tengan en el circuito integrado, dando dominio absoluto del sistema.
- *Complilador*: Es la herramienta informática que permite realizar programación en alto nivel (comúnmente en lenguajes como C o BASIC), permitiendo disminuir el tiempo de desarrollo de los sistemas con microcontrolador al dar al usuario un lenguaje con mejores prestaciones. Sin embargo, al ser un compilador se está limitado al uso de las instrucciones brindadas por el fabricante y la configuración de registros que éste tenga, limitando el uso de las prestaciones completas del microcontrolador.
- *Simulador/depurador*: Se trata de un software que es capaz de ejecutar en una computadora las aplicaciones del microcontrolador. Esto permite tener control de la ejecución línea a línea del código que se está trabajando, dando una potente herramienta para conocer los errores que se tengan o inconvenientes con las instrucciones en el microcontrolador.
- *Programador*: Es un dispositivo que se conecta a la computadora y permite grabar en el microcontrolador el código hexadecimal generado por el compilador. Estos dispositivos son dados por el fabricante o, en algunas ocasiones, pueden fabricarse por el mismo programador siguiendo los protocolos y circuitos recomendados por la compañía que los fabrique.
- *Cargador de arranque o bootloader*: Es un programa almacenado en la memoria del microcontrolador que permite que éste pueda borrar y volver a grabar información dentro de su memoria de arranque sin necesidad del programador. Esta característica solo está disponible en algunos microcontroladores y utilizan diferentes medios de comunicación como serie, USB, I2C, SPI etc. Son muy útiles en la etapa temprana del proceso de desarrollo pues puede programarse sin necesidad de ir al programa dado por el fabricante.
- *Placas de demostración*: Se trata de pequeños sistemas que tienen montados el microcontrolador y el dispositivo de programación (ya sea el programador dado por el fabricante o la herramienta del cargador de arranque), así como elementos de hardware que permitan realizar las acciones mínimas para el aprendizaje de este

tipo de sistemas. Pueden incluir elementos visuales como diodos emisores de luz, pantallas de cristal líquido, teclados, botones, sensores o módulos para el uso de las características analógicas etc.

1.8.12 INSTRUMENTACIÓN

Los procesos que requieren ser automatizados exigen que se tenga un control de diversas variables y magnitudes como lo son la presión, el flujo, el nivel, la temperatura, etc. Para ello es necesario utilizar instrumentos de medición y control que permitan que se tenga una regulación de las variables y mantenerlas en condiciones idóneas.

La instrumentación trata los sistemas que tienen como objetivo medir las magnitudes físicas en el entorno, recabar información sobre ellas y presentarlas de manera visual para un operador (Pérez M. , 2014). Los sistemas que permiten realizar estas tareas se conocen comúnmente como sistemas de medida. Un sistema de medida puede resumirse de acuerdo a la estructura que se muestra en la Figura 9.

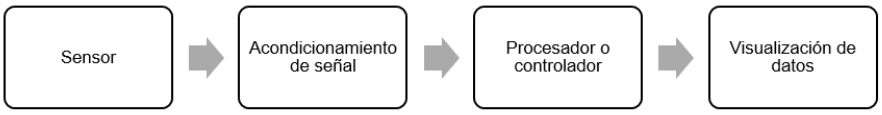

Figura 9. Estructura general de un sistema de medición.

Un sensor es el elemento que se encuentra en contacto con la parte física que se requiere medir y que al interactuar con él tiene cambios en su estructura interna proporcional a la magnitud y que comúnmente está relacionada con una señal eléctrica. Es importante diferenciar el término sensor con el de transductor pues se tiende a utilizar la expresión para ambos, un transductor es cualquier elemento que permite convertir de un tipo de energía a otra. Si bien un sensor contiene un transductor para realizar la conversión de la magnitud a electricidad, un transductor como un motor no puede llamarse sensor (Drake, 2015). La Figura 10 muestra la estructura de un sensor.

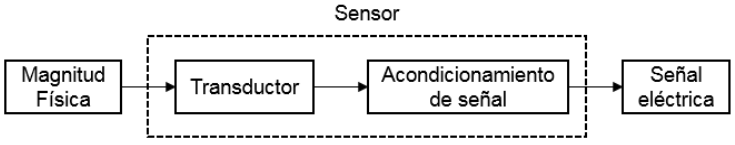

Figura 10. Estructura básica de un sensor.

1.8.12.1 Acondicionamiento de señales

Las señales que provienen de los sensores y sistemas de medición regularmente no tienen los niveles óptimos para el procesamiento de ellas por lo que es necesario realizar modificaciones que permitan al diseñador establecer las señales en los parámetros requeridos para su correcto procesamiento. Estas modificaciones toman el nombre de acondicionamiento de señal (Areny, 2004) .

De acuerdo a Areny, los acondicionadores de señal desempeñan las siguientes funciones principales:

1. Protección: Para evitar el daño de los elementos, puede ser por niveles de voltaje no adecuados o corrientes elevadas.
2. Convertir de un tipo de señal a otra: Es el caso en el que se necesita convertir una señal de corriente directa a alterna o viceversa. Ejemplo de ellos es cambiar un valor resistivo a tensión.
3. Obtención de un nivel adecuado de señal: Un ejemplo de ello son los termopares, en donde la salida se da en niveles de voltaje muy pequeños que no pueden ser leídos adecuadamente.
4. Eliminación o reducción de ruido: Eliminar señales no deseadas mediante filtrado.
5. Manipulación de la señal: Se utiliza cuando es necesario linealizar señales que no lo son para realizar un mejor procesamiento.

1.8.13 AMPLIFICADORES OPERACIONALES

Un amplificador operacional es un amplificador diferencial que tiene una ganancia muy alta, así como gran impedancia en la entrada y baja en la salida.

Los amplificadores operacionales se utilizan para proporcionar cambios en la amplitud del voltaje, para hacer osciladores, filtros y en la mayoría de los circuitos de instrumentación. Un amplificador operacional contiene varias etapas de amplificadores diferenciales para poder tener ganancias de voltajes muy grandes (Boylestad, Electrónica: Teoría de circuitos y dispositivos electrónicos, 2009).

La Figura 11, muestra un amplificador operacional básico que tiene dos entradas y una salida. Cada entrada produce una salida de la misma polaridad y la otra en opuesta, dependiendo a cuál de ellas se introduce la señal de ingreso.

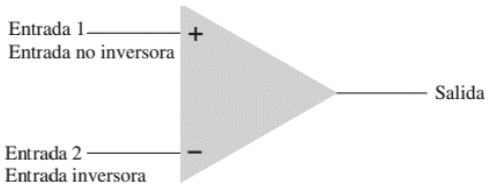

Figura 11. Amplificador operacional básico.

1.8.13.1 Entrada sencilla

Los amplificadores de entrada sencilla se tienen cuando la señal de entrada se conecta a una señal y la otra a tierra para crear una diferencia de potencial. Si la señal se conecta a la entrada positiva y tierra a negativa se va a producir una señal con la misma polaridad a la ingresada. Si se realiza de forma inversa la señal tendrá una polaridad invertida. Un ejemplo de funcionamiento se muestra en la Figura 12.

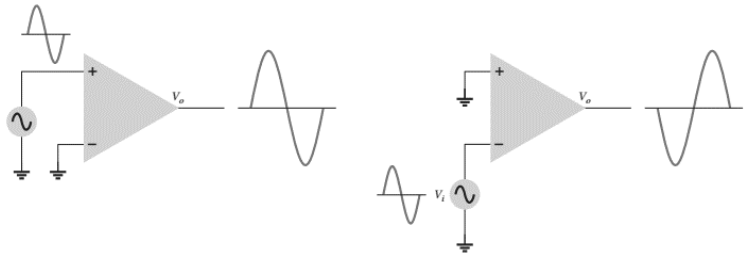

Figura 12. Amplificador operacional en entrada sencilla.

1.8.13.2 Entrada doble

En los amplificadores de entrada doble es posible ingresar señales en cada una de las entradas, lo que se conoce como operación de doble entrada. Si se ingresa un voltaje diferencial entre las dos terminales, la salida será la diferencia que exista entre la entrada positiva y la entrada negativa como se ve en la Figura 13.

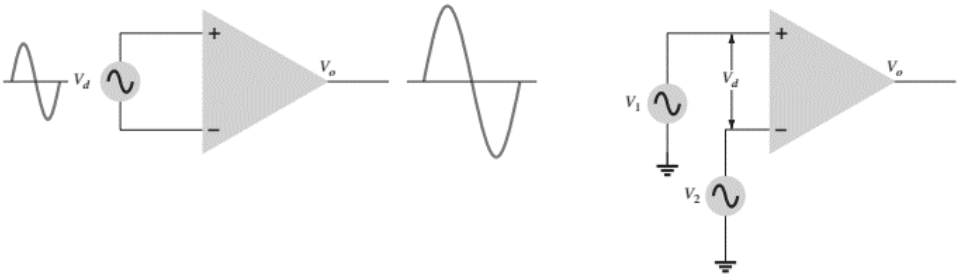

Figura 13. Amplificador operacional en doble entrada.

1.8.13.3 Amplificador inversor

El amplificador inversor tiene una salida que es proporcional a la ganancia establecida por el cociente entre la resistencia de retroalimentación (Rf) y la resistencia en la entrada (R1) y la polaridad es invertida a la señal de ingreso. La configuración de un amplificador operacional en modo inversor, así como su ecuación se pueden ver en la Figura 14.

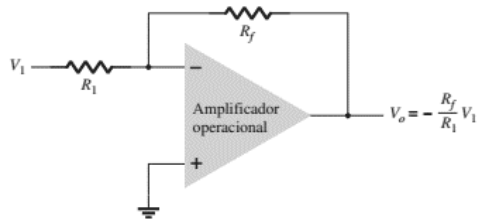

Figura 14. Amplificador operacional inversor.

1.8.13.4 Amplificador no inversor

También conocido como multiplicador de ganancia constante tiene un funcionamiento parecido al amplificador inversor. Sin embargo, en este la ganancia resulta de aumentar la unidad al cociente que forman las resistencias del circuito. En esta configuración se mantiene la polaridad que se tenga en la señal de entrada siendo la principal diferencia entre el funcionamiento de ambas configuraciones, el circuito de un amplificador no inversor es el que se presenta en la Figura 15.

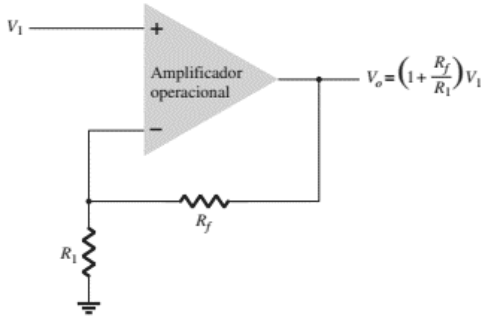

Figura 15. Amplificador operacional no inversor.

1.8.14 SISTEMAS DE ADQUISICIÓN DE DATOS

Los sistemas de adquisición de datos (DAQ) son aquellos que realizan el proceso de medir un fenómeno físico utilizando una computadora o un sistema digital. Los sistemas DAQ contienen diferentes elementos como sensores, hardware para realizar la medición de las

variables y una computadora con un software que pueda ser programable (National Instrument, 2018).

Si se compara con los sistemas antiguos de medición, los sistemas DAQ digitales permiten aprovechar la potencia de los procesadores para aumentar la productividad, tener una mejor visualización y habilidad para comunicar los elementos con otros sistemas. Regularmente este tipo de elementos se encuentran estandarizados para su mejor funcionamiento.

En la Figura 16 se muestran los elementos principales que componen a un sistema básico de adquisición de datos.

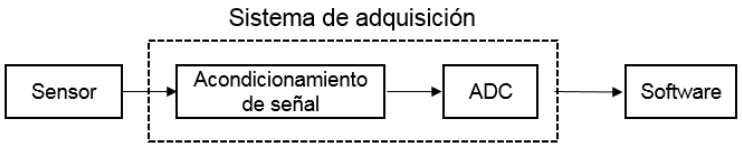

Figura 16. Partes de un sistema de adquisición de datos.

1.8.14.1 Conversor análogo – digital.

Los dispositivos de conversión análogo a digital (ADC) convierten un nivel de tensión analógico en su equivalente en palabra digital. Si n es el número de bit obtenidos de la palabra, esto significa que habrá 2^n niveles de tensión diferentes. Todo convertidor ADC debe procurar que el conjunto de bit obtenidos a la salida sea un reflejo lo más exacto posible del valor analógico correspondiente. Se usan un gran número de métodos para convertir señales analógicas a la forma digital, los más utilizados son: Rampa de escalera, aproximaciones sucesivas, paralelo y doble rampa (García L. , 2009). El ejemplo de funcionamiento de un conversor ADC se muestra en la Figura 17.

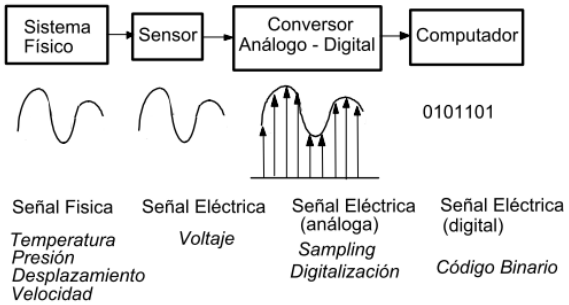

Figura 17. Descripción de conversión análoga - digital.

Para una aplicación efectiva de los conversores digital – analógicos es preciso conocer y saber las especificaciones de éstos, pues ponen en manifiesto las limitaciones y las prestaciones verdaderas que pueden brindar. Entre las principales características de estos dispositivos se encuentran (Miyara, 2004):

- *Resolución*: Es la cantidad de bits o dígitos binarios que acepta en su entrada. Se pueden expresar como el porcentaje del valor nominal máximo.
- *Exactitud*: Es la máxima desviación respecto a la línea recta que une el mínimo y el máximo valor ideales. Se expresa en bits menos significativos (LSB) lo cual significa que se usa el salto mínimo nominal por unidad. La descripción de esta característica se ilustra en la Figura 18.

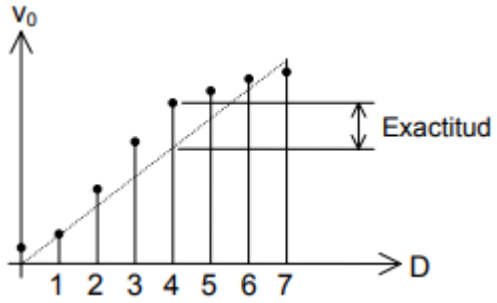

Figura 18. Error de exactitud en un ADC.

- *Error de escala*: Es el error que se obtiene a fondo de escala con respecto al valor ideal como se ve en la Figura 19. Se debe en general a errores de ganancia, en la referencia o en la red resistiva del conversor. Se expresa también en LSB y se busca que este error sea lo más cercano a cero.

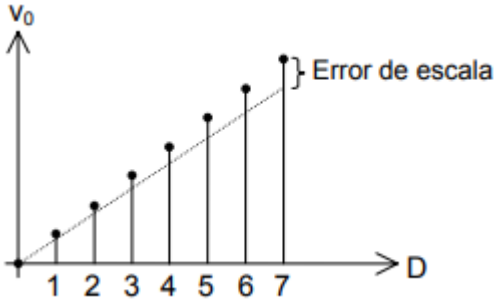

Figura 19. Error de escala en un ADC.

- *Tiempo de establecimiento*: Corresponde al máximo tiempo transcurrido luego de un cambio de código de entrada arbitrario para alcanzar el valor analógico correspondiente, es decir, cuando se tarda en obtener una lectura el conversor cuando la señal de entrada ha cambiado. El tiempo de establecimiento depende de la calidad de los amplificadores operacionales y su comportamiento lineal.

1.8.15 INSTRUMENTACIÓN VIRTUAL

Los instrumentos virtuales nacen debido a la necesidad de utilizar los sistemas informáticos para sustituir los instrumentos analógicos en la realización de mediciones y el análisis de variables físicas. Los instrumentos análogos tradicionales están hechos para cumplir una función específica y el hardware que tienen está diseñado para esta única función por lo que resultan poco versátiles si el sistema que se está creando sufre un cambio. Debido a esto se comenzaron a diseñar los instrumentos virtuales como alternativa (Ugurlu, 2011).

Para que los instrumentos virtuales cumplan su función deben de realizar las tareas de adquisición, almacenaje, análisis, visualización y comunicación de los datos. Así mismo deben de ser capaces de realizar acciones al proceso de acuerdo a las necesidades que sean colocadas por el usuario. Los instrumentos virtuales no deben considerarse una simulación de un sistema, el término virtual se refiere a que el instrumento está creado en un software que tiene una apariencia virtual de un instrumento real (Quiñones, 2011).

La instrumentación virtual está basada en la creación de interfaces gráficas de usuario *graphical user interfaces* o GUIs). La interacción entre el hardware y el software se realiza mediante lenguajes de programación que permiten realizar las funciones que deben cumplir los instrumentos.

Actualmente, los sistemas de monitorización y control de los procesos tienen en su mayoría instrumentos virtuales para su correcto funcionamiento, esto gracias al avance tecnológico de la informática y la electrónica. Las ventajas de utilizar instrumentos virtuales vienen de la mano con las ventajas de utilizar sistemas informáticos: incrementar la velocidad de procesamiento, comunicación entre diferentes plataformas de hardware, altas velocidades, toma de acciones en función de las tareas que se han programado y sistemas de alarma (Arrieta, 2014).

1.8.16 PROTOCOLOS DE COMUNICACIÓN

Lo fundamental en cualquier tipo de comunicación es resolver el problema de cómo llevar el mensaje desde un punto A hacia un punto B sin errores o con la mínima probabilidad de ellos, esto se logra mediante la codificación del mensaje de la mejor manera posible. Este tipo de codificación recibe el nombre de canal, que será el encargado de establecer la unión entre los dos puntos. Un ejemplo de lo mencionado anteriormente es el que se muestra en la Figura 20 (Coto, 2008).

Figura 20. Esquema básico de la comunicación entre dos puntos.

El protocolo se define como las reglas para la transmisión de la información a través del canal. Un protocolo de red de comunicación es un conjunto de reglas que gobiernan el intercambio ordenado de datos dentro del canal.

Los elementos básicos de un protocolo de comunicación son: Conjunto de símbolos o caracteres, conjunto de reglas para la secuencia y sincronización de los mensajes construidos a partir de los símbolos y los procedimientos para determinar cuándo se ha enviado el mensaje correctamente o se ha detectado un error, para que exista comunicación entre ambos puntos por el canal se debe enviar la misma configuración en cada uno de ellos (Pérez C. , 2012).

Según Pérez para el establecimiento de reglas es necesario identificar los siguientes puntos: Un emisor y receptor, método de comunicación acordado, idioma común (tipo de cifrado), velocidad y momento de entrega y requisitos de confirmación o acuse de recibo. Una vez establecidos los puntos anteriores el diagrama de la comunicación una vez realizado el protocolo deberá ser parecido al que se presenta en la Figura 21.

Figura 21. Comunicación con protocolo.

1.8.17 SOFTWARE LIBRE

Se le conoce como software libre a todo aquel que respeta la libertad para los usuarios y la comunidad, es tener la libertad para poder ejecutar, realizar copias, estudiar y distribuir el software con la finalidad de realizar modificaciones que permitan mejorarlo (GNU, 2018). De acuerdo al *Free Software Foundation* las cuatro libertades esenciales del software libre son:

1. Libertad de ejecución del programa para cualquier propósito.
2. Libertad para el estudio del funcionamiento y cambiarlo para que realice las funciones que el programador desee.
3. Libertad de distribución de copias con la finalidad de ayudar a otros.
4. Libertad de distribuir copias de las versiones modificadas por la comunidad con la finalidad de que ésta se beneficie, siendo necesario el acceso al código fuente para que pueda ser remodificado.

Software libre no significa que no pueda ser comercial, es posible obtener los programas habiendo pagado por la copia. Sin embargo, esto no debe impedir que se tenga la libertad para copiar y modificarlo e incluso, en algunos casos, vender las copias creadas.

1.8.17.1 Raspbian®

Raspbian es un sistema operativo basado en la distribución Debian Jessie GNU/Linux y que es el recomendado para utilizar en las microcomputadoras Raspberry Pi®, pues se encuentra optimizado para el hardware que ésta contiene y que son utilizados comúnmente en la enseñanza de sistemas informáticos.

El sistema operativo viene precargado con algunos programas para la educación la programación y otros de propósito general. Entre ellos se encuentra Python®, Scratch®, Sonic Pi®, Java®, Mathemática® (Raspberry Foundation, 2018).

Al ser una distribución GNU/Linux tiene la filosofía de un software libre por lo que se puede ejecutar, copiar, modificar y distribuir sin la necesidad de realizar pagos en su licencia. Todo software que se realice en este sistema operativo puede ser recompilado en la propia placa Raspberry y contiene gran cantidad de repositorios (bibliotecas) donde se pueden descargar programas como se tiene en cualquier otra distribución de Linux en los equipos de escritorio (Salcedo, 2016).

Estas características han permitido que las Raspberry Pi® puedan utilizarse para infinidad de proyectos gratuitos e incrementado la extensión de sistemas informáticos programables de bajo costo aplicados en países en vías de desarrollo.

1.8.17.2 Python®

Python® es un lenguaje interpretado de programación que se basa en la filosofía de software libre y fácil pues intenta que la sintaxis sea lo más sencilla posible y que los códigos sean legibles. Tiene una curva de aprendizaje muy sencilla por la basta documentación. Sin embargo, cuenta con estructuras potentes de los lenguajes de algo nivel que utilizan programación orientada a objetos (POO).

Es un lenguaje que soporta diversos estilos de programación como lo es la POO y el uso de la programación imperativa. Utiliza un lenguaje dinámico y funciona en Windows®, MacOs® y Linux® (Python Software Foundation, 2018).

Este lenguaje permite que los programas sean divididos en bloques o módulos que pueden reutilizarse desde otros programas dentro de la misma distribución. Como es un lenguaje interpretado, no es necesario realizar una compilación y se puede hacer una depuración del código en tiempo real en el modo interactivo, lo que da lugar a la exploración de las instrucciones del lenguaje o probar fragmentos de programa de manera rápida dentro de la plataforma de desarrollo (Rossum, 2009).

Se tiene una licencia de software libre llamada *Python Software Foundation License* por lo que se tiene acceso sin costo a los intérpretes, módulos o bloques de función, basta documentación gratuita, siendo uno de los lenguajes más utilizados dentro del desarrollo de plataformas de *Open Source.*

Python® tiene una filosofía para transferencia y legalidad que deben de tenerse en todos los programas realizados en la plataforma y que es necesario seguir si se quiere ser programador en la comunidad de desarrollo. Estos principios fueron expuestos por uno de los desarrolladores más conocidos dentro de la comunidad de Python®. La filosofía según Peters (Peters T. , 2016) de este lenguaje de programación se basa en:

- Los casos especiales no son tan especiales como para quebrantar las reglas.
- Lo práctico gana a lo puro.
- Los errores nunca deberían dejarse pasar silenciosamente.

- A menos que hayan sido silenciados explícitamente.
- Frente a la ambigüedad, rechaza la tentación de adivinar.
- Debería haber una -y preferiblemente sólo una- manera obvia de hacerlo.
- Ahora es mejor que nunca.
- Aunque nunca es a menudo mejor que ya mismo.
- Si la implementación es difícil de explicar, es una mala idea.
- Si la implementación es fácil de explicar, puede que sea una buena idea

Debido a su naturaleza es fácil encontrar módulos creados por la comunidad para encontrar la solución a muchos problemas presentes en diversas ramas de la programación como lo son diseño web, matemático, comunicación entre dispositivos y uso de elementos de hardware, convirtiéndolo en una herramienta versátil para el uso que el usuario quiera darle (Halterman, 2011).

1.8.18 SCYPI

Es una colección de algoritmos matemáticos y funciones construidas con la extensión *Numpy* para el lenguaje de programación Python® que permite el manejo de arreglos matemáticos o vectores (ScipyOrg, 2016).

Agrega herramientas a Python que proveen al usuario de comandos y clases de alto nivel para la manipulación y visualización de información, debido a esto, convierte este lenguaje de programación en un ambiente interactivo y fácil de utilizar para el procesamiento de datos al mismo nivel que otras herramientas como MATLAB® o Scilab®.

Este módulo es una combinación de diferentes paquetes creados por la comunidad de Python® y que tienen la finalidad de realizar investigaciones científicas, brindando módulos que permitan el desarrollo de la ciencia mediante lenguaje de programación. El paquete contiene los siguientes módulos, cada uno con amplia documentación en sus sitios web, así como por la comunidad y desarrolladores que utilizan esta herramienta.

- *Numpy*: Principal paquete para cálculos numéricos por medio de la computadora en Python, brinda vectores y matrices así como operaciones básicas que se realizan entre ellos (NumpyDevelopers, 2016).
- *Scipy Library*: Colección de algoritmos y *toolkits* de propósito específico como el procesamiento de señales, optimización, estadística entre otros.

- *Matplotlib*: Un paquete que permite realizar gráficas con calidad de publicaciones en 2D y rudimentarias gráficas en tercera dimensión. Permite introducir comandos de Latex® para realizar gráficos más presentables, es uno de los paquetes más utilizados por la comunidad (MatplotlibOrg, 2016).
- *Sympy*: Permite realizar operaciones matemáticas simbólicas y algebra computarizada (SymPy Developers, 2016).

Al igual que Python® todos los módulos anteriores son bajo la filosofía *Open Source*, permitiendo modificar los módulos y el código fuente de acuerdo a las necesidades del programador, siempre y cuando sean publicadas las mejoras.

1.8.19 PROGRAMACIÓN WEB

La programación de los sitios web es una de las disciplinas dentro del mundo de la programación que más se ha desarrollado, abriendo un mundo de posibilidades para crear nuevos sistemas de diferentes tipos.

De acuerdo al Departamento de Desarrollo web (Universidad de Chile, 2008), los tres pilares básicos sobre los que se sustenta la arquitectura lógica de la Web son:

1. Identificadores únicos (URI): Para poderse referenciar o describir un objeto se necesita que éstos tengan un nombre para identificarlos. En la web estos nombres propios se llaman Identificadores Universales de Recursos, que corresponda a una dirección en la Web.
2. Lenguaje Universal: En la comunicación universal debe tenerse un lenguaje único que sea entendible por todos. Para ello se diseñó el lenguaje *Hyper Text Markup Lenguage* (HTML), que es un lenguaje de hipertexto, es decir, permite redirigir al lector desde un punto cualquiera de texto a otro, a esto se le conoce como enlaces web.
3. Protocolo de Transmisión de datos HTTP: Se requiere de un protocolo que permita enviar y recibir información en HTML de un sitio a otro. El protocolo *Hyper Text Transfer Protocol* (HTTP) es un protocolo de transmisión entre clientes y servidores. “Las peticiones HTTP pueden realizarse usando dos métodos” (Mateu, 2004). El método GET, envía parámetros con la petición codificados en la URL y el método POST, lo hace como parte del cuerpo de la petición, no es visible para el usuario, solamente puede verse desde el servidor.

1.8.20 RASPBERRY PI ®

La Raspberry Pi® es un microcomputador u ordenador de placa reducida desarrollada en el Reino Unido por la Fundación Raspberry que ha crecido en popularidad por su habilidad para realizar funciones comunes en las computadoras de escritorio convencionales como lo es el procesar texto, utilizar hojas de cálculo, navegación y programación web así como el uso de juegos (RPiFundation, 2017).

Surgió en los Laboratorios de informática de la Universidad de Cambridge con la intención de realizar ordenadores con bajo costo en la enseñanza de la informática entre los niños y jóvenes.

El concepto de la Fundación Raspberry es hacerla accesible para todos, por lo que se ha dotado a la placa con lo básico para utilizar ambientes de programación, así como puertos de conexión de hardware para realizar proyectos electrónicos de una manera sencilla. Esta última cualidad es lo que la ha vuelto popular en el uso de sistemas del Internet De las Cosas (IoT) (Asadi, 2014).

La diferencia principal entre la microcomputadora Raspberry Pi® y una computadora común, aparte del tamaño, es que utiliza un sistema operativo diferente basado en Linux® llamado Raspbian®, por lo que no es necesario pagar licencias o trabajar en ambientes cerrados y de códigos restringidos como lo son los sistemas más populares de Windows® y Apple®, creando un ambiente libre para cualquier usuario común que no tiene fondos financieros (Upton, 2012).

El gran éxito del proyecto Raspberry radica en la basta documentación y la comunidad que se ha creado alrededor de ella, disponiendo tutoriales para principiantes sin conocimientos informáticos. La Figura 22 muestra una placa Raspberry Pi® 3 Modelo B.

Figura 22. Placa Raspberry Pi® 3.

CAPÍTULO 2: METODOLOGÍA

2.1 TIPO DE INVESTIGACIÓN

El enfoque dado para la resolución del problema planteado es la investigación aplicada. La investigación aplicada también puede llamarse como investigación práctica o empírica y se caracteriza en la búsqueda de aplicaciones o utilización de conocimientos adquiridos para implementar y sistematizar la práctica basada en las investigaciones (Vargas, 2009). Algunas de las ideas de este tipo de enfoque son:

- Incluir esfuerzos sistemáticos para resolver problemas o situaciones como lo pueden ser la innovación técnica, artesanal industrial y científica.
- Considera los estudios de teorías científicas que han sido validadas para la solución de problemáticas y el control de situaciones.
- Requiere de un marco teórico para basar la generación de la solución al problema que se quiere resolver.
- Se centra en el análisis y solución a las problemáticas de la vida real.
- Debe nutrirse de los avances científicos actuales y anteriores.

La investigación aplicada guarda relación con la investigación básica que es donde se obtienen los conocimientos previos, pues depende de ella para sustentar las investigaciones realizadas. Pero la característica más destacada de la investigación aplicada es su interés en la aplicación y en las consecuencias prácticas de los conocimientos que se han obtenido. El objetivo de la investigación aplicada es predecir un comportamiento específico en una situación definida (Ander, 2011).

Esta investigación también es conocida como empírica, dado que busca la aplicación del conocimiento adquirido con la idea de consolidar el saber para resolver una situación.

2.2 DESCRIPCIÓN DEL PROYECTO

El proyecto ha sido dividido en diferentes etapas que se explican a continuación. Esto con la finalidad de realizar diferentes experimentos y analizar sus resultados para optar por la solución más óptima para el problema, y dando paso a futuras investigaciones que puedan basarse en los problemas y soluciones que se presentan en la realización del proyecto.

2.2.1 ESQUEMA GENERAL DEL PROYECTO

Con la finalidad de entender a manera general las tareas que se pretenden realizar en este proyecto se presenta el diagrama de la Figura 23. En él se pueden observar los elementos con los que contarán los sistemas una vez que sean montados, así como las fases correspondientes a cada uno.

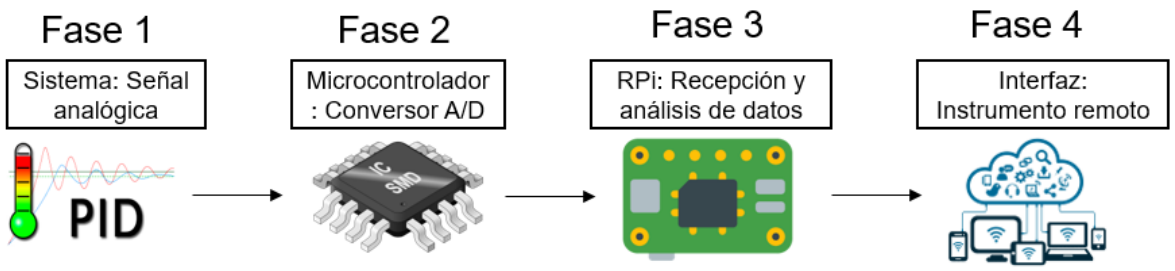

Figura 23. Diagrama general del proyecto.

El sistema consiste en un sistema físico de tamaño reducido que pueda ser replicable fácilmente y contenga una variable con un amplio uso en diferentes ámbitos. Posteriormente es necesario recolectar los datos de la variable física (tipo analógico) y transformarlos a palabras digitales, tarea que se optó fuera realizada mediante un microcontrolador utilizando sus periféricos para señales analógicas. Una vez realizada la conversión los datos van a ser recibidos y analizados dentro de la mini computadora Raspberry Pi® para posteriormente visualizar los datos de manera numérica y gráfica utilizando un instrumento virtual en un servidor web.

2.2.1.1 Fase 1: Evaluación de plantas de prueba

Como primer punto es necesario la evaluación de diferentes plantas de prueba, utilizando como referencia trabajos anteriores en la implementación de sistemas de control automático, así como aquellos que son comunes como proyectos en la enseñanza de la ingeniería.

Esta fase del proyecto requiere de investigación del estado del arte, así como recolección de datos con docentes del área de sistemas de control. De igual forma tomar en cuenta la facilidad con la que los sistemas pueden ser construidos y su capacidad de ser replicados sin el uso de elementos especializados que no puedan adquirirse fácilmente.

2.2.1.2 Fase 2: Experimentación de sistema con microcontrolador

En esta etapa se pretende la experimentación con diferentes plataformas para la implementación de sistemas con microcontroladores, con la finalidad de utilizar aquel que cumpliese todas las características necesarias para el buen funcionamiento de los diferentes sistemas a utilizar tomando como guía el seguimiento que se muestra en la Figura 24.

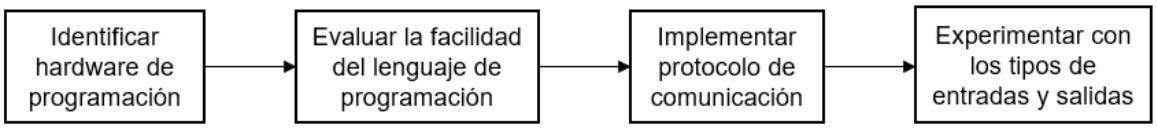

Figura 24. Diagrama de flujo para pruebas de microcontroladores.

Las tareas descritas anteriormente permiten evaluar cada uno de los microcontroladores y así comprobar cuál de ellos cumple los requerimientos que se requieren entre los que se encuentran:

- Facilidad de programación sin necesidad de hardware adicional.
- Existencia de librerías que faciliten la programación en general, así como el uso de los diferentes dispositivos de hardware, así como la implementación de algoritmos para el manejo de cadenas de caracteres y la comunicación por protocolo serie.
- Recursos de hardware analógico y digital que permitan el uso de periféricos necesarios para el manejo de cargas de potencia.

2.2.1.3 Fase 3: Programación del sistema para el análisis de datos.

El ordenador Raspberry Pi® cuenta con diferentes lenguajes de programación y herramientas matemáticas que permitan el análisis matemático de los datos recabados de la planta de prueba, por ello se requiere evaluar cuál de ellos tiene las características principales para el cumplimiento de esta tarea, entre éstas se encuentran:

- Librerías para el manejo de sistemas de ecuaciones diferenciales.
- Recepción de datos por puerto serie.
- Generación de gráficos.
- Generación de entregables en un formato multiplataforma (xls).

La metodología a utilizar para la evaluación de los sistemas de análisis matemático y recepción de datos se muestran en la Figura 25.

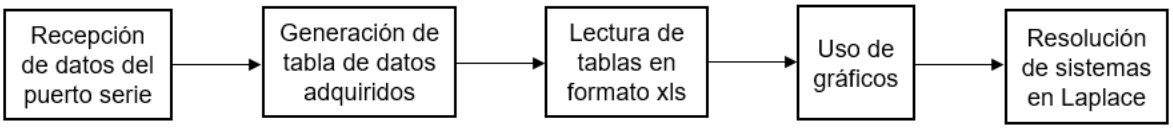

Figura 25. Diagrama de flujo para prueba de análisis de datos.

2.2.1.4 Fase 4: Implementación del instrumento virtual remoto

La última fase del proyecto consiste en la realización de las pruebas con diferentes plataformas de programación web que permitan la interacción del usuario con el sistema propuesto, con la finalidad de tener un manejo de los diferentes parámetros del controlador y visualizar los gráficos de la respuesta de éste. La plataforma a utilizar debe realizar lo siguiente:

- Poder implementarse dentro de la Raspberry Pi®.
- Debe ser de uso libre.
- Existencia de basta documentación para su programación.
- Realizar transmisiones de video para el monitoreo en tiempo real de la planta.
- Uso dinámico de los elementos de la interfaz.

El diagrama de las tareas a realizar para conocer si se cumplen los requerimientos descritos anteriormente se presentan en la Figura 26.

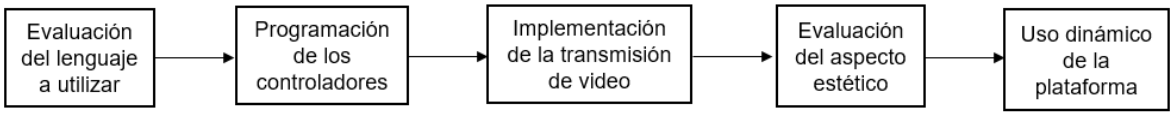

Figura 26. Diagrama de flujo para la evaluación de la plataforma web.

Tomando como base la metodología propuesta se propone la realización de un prototipo inicial para la evaluación de los primeros elementos seleccionados para cada fase para su evaluación. Una vez que se ha realizado se recopilará la información obtenida de la retroalimentación de las diferentes pruebas realizadas para solucionar fallas y mejorar los puntos que se requieran para obtener los elementos óptimos para la implementación del proyecto.

CAPÍTULO 3: DESARROLLO DEL PROYECTO.

3.1 PROTOTIPO DEL LEVITADOR CON FLASK® Y ARDUINO®

La primera prueba se realizó utilizando Arduino como microcontrolador con la idea de utilizar un software libre y su gran comunidad con documentación dada evaluando el desempeño de procesamiento tanto en la comunicación como en la parte matemática.

3.1.1 DISEÑO DEL SISTEMA DE MEDICIÓN

El sistema tiene como variable controlada la distancia. Se optó por el uso de un sensor ultrasónico HC-SR04 que es un elemento de bajo costo y con una programación sencilla en comparación con otros que se encuentran en el mercado tiene un funcionamiento que no depende en gran medida de factores externos pues regresa un valor en alto con una duración acorde al tiempo que tarda la señal ultrasónica en regresar y ser captada por el receptor. La conexión del sensor y la placa Arduino® se muestra en la Figura 27.

Para realizar la medición se debe entregar un pulso con duración de 10 microsegundos por el pin de disparo para que se envíe la onda ultrasónica, una vez hecho esto, el sensor mantendrá el pin de *Echo* en alto hasta que la onda regrese.

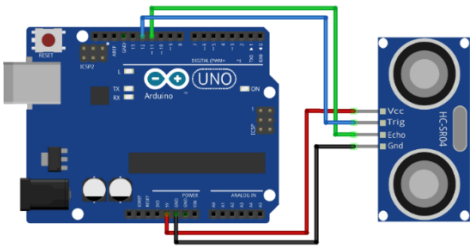

Figura 27. Conexión del sensor HC-SR04.

Se utiliza la función *pulseIn* de Arduino® que permite medir el tiempo en milisegundos que dura una entrada en alto. Con el tiempo obtenido se utiliza la constante de velocidad del sonido en el aire de 340m/s en la ecuación 27, donde v es la velocidad del sonido y t el tiempo en milisegundos medido. Se divide el resultado entre 2 dado que el tiempo recibido es el tiempo de ida y vuelta quedando la ecuación 28.

$$d = \frac{v\,t}{2} \tag{24}$$

$$d = \frac{0.034t}{2} = 0.017t \tag{25}$$

El fragmento de código para la programación del funcionamiento del sensor se muestra en la Figura 28.

```
digitalWrite(trig, LOW);
delayMicroseconds(5);
digitalWrite(trig, HIGH);
delayMicroseconds(10);    //Se envía el pulso necesario
tiempo = pulseIn(echo, HIGH); //se mide el tiempo en alto del echo
distancia =0.017*tiempo;//Se calcula la distancia
Serial.print("Distancia: ");
Serial.println(distancia);
delay(20);
```

Figura 28. Código de lectura de distancia en Arduino®.

3.1.2 CREACIÓN DEL PROTOCOLO DE COMUNICACIÓN EN ARDUINO®

Aunque se utiliza el protocolo serial para el envío de información, es necesario crear un protocolo propio de comandos que permitan enviar instrucciones al Arduino® y que se guarden los parámetros necesarios o se inicie alguna acción ya que pueden perderse datos en la comunicación entre dispositivos. Para la programación de éste se requieren comandos para enviar el número de muestras, el tiempo de muestreo y la instrucción para que el Arduino® comience a enviar los datos por el puerto serie.

La trama se diseñó con un separador entre el comando y su parámetro (@) y un caracter de final de trama (/), por lo que el sistema de comunicación quedaría de la siguiente manera: *comando@parámetro/*

En la programación de estos comandos se utiliza la clase *String* de Arduino® que permite manipular y usar cadenas de texto en una manera más compleja que en arreglos de caracteres, se puede concatenar, agregar datos, buscar y reemplazar sub-cadenas (Arduino , 2016). Si bien esta clase toma más memoria dentro del microcontrolador, es una herramienta sumamente útil y fácil de utilizar. Se determinaron los comandos a utilizar y que serían interpretados por la tarjeta Arduino® en la se muestra la declaración de los comandos.

```
int DATA[10];
const String CMD[]={"SetPWM", "SetSamples", "SetTime", "StartSample", "C/","SendData","SendFreq"};
byte NUM_CMD = 7;
```

Figura 29. Declaración de comandos en Arduino®.

Ya que el sistema está pensado para poderse modificar, los comandos y parámetros se colocaron dentro de dos arreglos, puede observarse que el arreglo de los parámetros es de tipo entero, pues ahí se guardaran los números que representan el tiempo y las muestras, mientras que los comandos al ser cadena, el arreglo debe ser de tipo *String*.

```
String Read_dato()
{
  String Serial_Dato;//Cadena de recepción
  int IncomingBytes;//Numero de bytes en puerto serie
  IncomingBytes   = Serial.available();
  Serial_Dato= "";

  if (IncomingBytes>0) //Si existen datos en el buffer
  {
     for (int i=1 ; i<=IncomingBytes; i++)
     {
       char c = Serial.read(); //Se asigna el byte a 'c'
       if (c == '/') FLAG = 1;//Bandera de trama completa

       Serial_Dato += c; //Se concatenan todos los bytes
      }
    }
   return Serial_Dato; //Se regresa la cadena obtenida
  }
```

Figura 30. Función de lectura del puerto serie en Arduino®.

La programación fue realizada mediante funciones con la finalidad de hacer el código legible y poder ser reutilizadas si el usuario quiere modificarlo. La primera función programada fue la recepción de datos que se muestra en la Figura 30

Se crean las variables necesarias, la cadena que será devuelta por la función y el número de bytes disponibles en el buffer de puerto serie. Posteriormente se pregunta si existen datos en el buffer para ser leídos y los concatena en la cadena declarada. Si se encuentra el final de trama se activa la bandera de que el dato llego completo y se retorna la cadena que se ha concatenado.

Una vez obtenida la cadena, es necesario conocer si el comando es válido, por lo que se busca en el arreglo de comandos, la función se ve en la Figura 31.

```
void Find_dato(String BUFFER)
{
  int anchoCadena;//Largo de cadena
  int PosIndex;//Posición del separador
  int Pos_Slash;//Posición del final de trama
  String Temp_CMD;//Cadena de comando
  String Temp_Par;//Cadena de parámetro

  anchoCadena = BUFFER.length();//Largo de cadena
  PosIndex = BUFFER.indexOf('@');//Encuentra el separador
  Temp_CMD= BUFFER.substring(0,PosIndex);//Se extrae el comando
  Temp_Par = BUFFER.substring(PosIndex+1, anchoCadena-1);//Parámetro

     for (int i=0; i<=NUM_CMD; i++)
     {
      //Barrido en busca de comando
        if(Temp_CMD.equals(CMD[i])==1)
        {
          DATA[i] = Temp_Par.toInt();
          //Guarda el parámetro en el arreglo
          }
     }
}
```

Figura 31. Función para encontrar comandos en Arduino®.

La función recibe como argumento una cadena para procesarla, se crean variables para guardar el largo de la cadena, la posición del separador (@), la posición del fin de trama (/) y dos cadenas donde se almacenarán temporalmente el comando y el parámetro. Para analizar la cadena se utilizan los métodos de la clase *string*, el método *length* permite conocer el ancho de una cadena, *indexOf* permite conocer la posición de un carácter en la cadena, y *substring* permite sustraer una sub – cadena que se encuentre en los índices indicados.

Es necesario encontrar dos sub - cadenas, la correspondiente al comando que se encuentra desde el índice cero a la posición del separador, y la del parámetro que se encuentra después del separador hasta el final de trama. Al validar el comando su parámetro se guarda en la posición del arreglo correspondiente y lo transforma a entero para poder usarlo en operaciones en Arduino®.

3.1.3 COMUNICACIÓN SERIAL ARDUINO® – RASPBERRY PI®

La comunicación serial es un protocolo muy utilizado a la fecha, por ello la mayoría de los lenguajes de programación contienen librerías para el uso de ésta

Sin embargo, en algunos casos, no vienen por defecto. Python® contiene un módulo que debe ser instalado para el manejo de dispositivos mediante protocolo serial. Una vez que se ha instalado el paquete se procede a realizar la programación utilizando Python 2.7®.

La primera prueba que se realizó fue enviar dos caracteres, de manera que al enviar una "A" el puerto 13 del Arduino® estuviera en alto y al enviar una "B" pasara a bajo. El script de Python® que ejecuta la función se muestra en la Figura 32.

```
import serial#Se importa la libreria
#Se configura el puerto serie
arduino = serial.Serial('/dev/ttyACM0',9600)
while True:
    #Se envìa el caracter recibido por el teclado
    arduino.write(raw_input('Ingresa la letra:'))
```

Figura 32. Envío de datos de Python® a Arduino®.

El código importa las librerías para el puerto serie, posteriormente se crea un objeto de la clase Serial y se define el puerto serie y la velocidad en baudios. A diferencia de Windows®, en Linux® los puertos seriales toman diferente nombre en este caso se coloca el directorio donde se almacena la configuración del puerto.

Se crea un lazo infinito en el cual mediante la instrucción *raw_input()* se lee una cadena proveniente del teclado y se escribe por puerto serie hacia Arduino®. En la Figura 33 se muestra un fragmento del código que realiza la lectura en Arduino®.

```
if(Serial.available()>0)
{
 dato = Serial.read();
}
if (dato == 'A')
digitalWrite(13, HIGH);
if (dato == 'B')
digitalWrite(13, LOW);
```

Figura 33. Recepción de datos con Arduino®.

Comprobando el correcto funcionamiento del envío de datos desde Python® al dispositivo, se realizó la comunicación inversa, es decir, enviar datos desde Arduino® y ser leídos por la terminal de Python. El código que se muestra en la Figura 34, muestra un contador que envía su valor por puerto serie desde Arduino® a la Raspberry pi®.

Es importante tomar en cuenta que la impresión de los datos en Arduino® debe contener un salto de línea, pues el método de recepción utiliza ese carácter como separador de datos.

```
void loop()
{
  Serial.println(contador);
  delay(1000);
  contador++;
}
```

Figura 34. Envío de datos de Arduino® a Python®.

Como se mencionó anteriormente, el método *readline()* lee los datos del puerto serie y los separa en donde encuentre un salto de línea. El fragmento de código de la Figura 35 es el utilizado.

```
import serial#Se importa la librerìa
#Se configura el puerto serie
arduino = serial.Serial('/dev/ttyACM0',9600)
while True:
    print arduino.readline()
```

Figura 35. Recepción de datos por salto de línea en Python®.

El resultado de la comunicación fue satisfactorio y dio paso a observar el comportamiento de la recepción de datos. Como se puede ver en la Figura 36, el dato es recibido y enviado a la terminal de Python de manera correcta, pero se puede observar que junto a la cuenta, se imprime un signo que no ha sido programado en el envío con Arduino®.

Este símbolo representa un problema ya que, al momento de realizar el algoritmo de recepción de datos, si se quiere convertir éste en un valor numérico, el lenguaje de programación no lo permite pues no es un número solamente, si no la combinación de diferentes caracteres ascii, siendo necesario la clasificación de los datos para el buen funcionamiento del sistema de adquisición de datos.

Figura 36. Impresión de datos en la terminal de Python®.

Al inspeccionar el código de ambas plataformas se encontró que el símbolo es la interpretación de la terminal de Python® al conjunto de salto de línea (/n) y retorno de carro (/r) que se envían al utilizar el método *Serial.println()*

3.1.4 ADQUISICIÓN DE MUESTRAS UTILIZANDO LA TERMINAL DE PYTHON®

Una vez identificado el símbolo se procedió a realizar la prueba para conocer si el protocolo de comandos programado en Arduino® funcionaba correctamente si se envían desde la Raspberry Pi®.

Los datos recibidos serán ahora lecturas del sensor HC-SR04, para ello se utilizaron las funciones definidas anteriormente para el procesado de las cadenas recibidas en el buffer del puerto.

El fragmento de código para la medición de distancia ha sido introducido, así como una función que permite realizar el muestreo con las características que el usuario coloque en la terminal de Python®. En la Figura 37 se muestra la función para realizar el muestreo.

Consiste en un ciclo for, en el cual el número de iteraciones que se realizan corresponden al tiempo de muestreo almacenado en el arreglo de parámetros (DATA[1]), posteriormente se manda llamar a la función *Read_Distance()* que devuelve la distancia medida por el sensor y es enviada por puerto serie. Se tiene un retardo que toma el valor del parámetro de tiempo de muestreo para realizar la siguiente muestra.

Una vez terminado el ciclo se imprime un caracter como señal a la Raspberry de que se han enviado las muestras solicitadas, pues si no se envía no se puede tener la seguridad de que la colección de datos fue enviada correctamente.

Para la realización de esta prueba no se utiliza el final de trama en el script de Python® pues solo se requería saber si los comandos eran reconocidos de manera satisfactoria.

```
for (int i=1; i<=DATA[1]; i++)
{
  //Se envía la distancia por el puerto serie
  Serial.println(Read_Distance());
  //Espera el tiempo establecido para hacer nueva medición
  delay(DATA[2]);
  }
  //Se imprime el final de la trama
  Serial.println("$");
  DATA[3]=0;
```

Figura 37. Ciclo FOR de muestreo en Arduino®.

Se modificó el script de recepción para almacenar los datos en una lista. Una lista en Python® es un conjunto de valores que pueden o no ser del mismo tipo, funciona similar a los arreglos en C++, sin embargo, se tienen métodos para el manejo de ellas.

Los comandos se enviaron utilizando las cadenas leídas del teclado y se enviaron con un retardo de medio segundo entre ellos para permitir al Arduino® leerlos de mejor manera. La modificación que realiza la lista con los datos enviados se muestra en la Figura 38.

```
#Escribir las muestras colocadas
while len(datos) <= int(muestras):
        dato = arduino.readline();
        #Se ingresan los datos uno tras otro
        datos.append(dato)

print datos
```

Figura 38. Toma de muestras en Python®.

El método *append()* permite introducir datos en una lista, uno detrás de otro sin necesidad de conocer el índice, siendo una herramienta necesaria para el análisis de muestras, pues permite guardar colecciones de datos de una manera sencilla y con la ventaja de que existen métodos para procesarlos de diferentes maneras. Las muestras tomadas se ven en la Figura 39. Puede observarse que efectivamente el símbolo correspondía al retorno de carro e impresión de línea.

```
Ingresa muestras: 20
Ingresa el tiempo de muestreo:100
['0.00\r\n', '98.81\r\n', '101.05\r\n', '99.88\r\n', '99.17\r\n', '99.19\r\n', '
100.34\r\n', '99.97\r\n', '99.88\r\n', '99.17\r\n', '98.81\r\n', '101.12\r\n', '
101.12\r\n', '100.34\r\n', '99.55\r\n', '102.22\r\n', '100.62\r\n', '100.64\r\n'
, '99.55\r\n', '100.71\r\n']
```

Figura 39. Lista con las muestras obtenidas en Python®.

3.1.5 PROGRAMACIÓN DEL SERVIDOR WEB CON FLASK®

3.1.5.1 Montaje del servidor

Con la utilización de Flask® no se requiere de un servidor externo que necesite ser montado en la Raspberry Pi®, pues Flask® cuenta con su propio servidor que se instala automáticamente junto con el *framework* y que puede funcionar tanto para hacer pruebas como para su uso común. Para el uso del framework es necesario crear un directorio de trabajo donde se leerán los archivos que se necesiten en las peticiones.

Dicho directorio consta básicamente de la raíz (lugar para los scripts de Python®, una carpeta para los archivos HTML que serán renderizados (*templates*), que obligatoriamente debe tener el nombre para que Python lo reconozca, y una carpeta para los archivos dinámicos utilizados en el código HTML (*static*).

El *framework* basa su funcionamiento en la lectura de las URL y se asocia una función de Python a ella que corresponde al código que se ejecutará cuando se realice dicha petición, retornando ya sea una respuesta dentro del HTML actual o una página nueva, sin embargo, no solamente pueden retornar HTML si no variables del *Script.*

Existen diversas funciones que contiene Flask® para la realización de contenido dinámico y estático de acuerdo a las necesidades que se tengan, por lo que es recomendable incluir aquellas que se utilicen en el código. La importación de las librerías *render_template* y *request*, permiten renderizar páginas HTML como respuesta y leer las peticiones del cliente.

Se debe definir la función de todas las peticiones GET que se tengan del cliente mediante el método *@app.route()*, pues si se realiza una petición que no tenga una definida, el servidor entrará en error y no realizará ninguna opción. Una vez que se han programado todas las funciones es necesario correr el servidor con el método run, el cual puede contener distintos argumentos: *debug*, permite ejecutar el servidor cada que se realiza un cambio al código principal sin necesidad de correr la aplicación desde la terminal de la Raspberry Pi®.

3.1.5.2 Integración de Formularios de Flask® al HTML

Comúnmente los formularios de las páginas web se crean desde el código HTML con las etiquetas especiales que contiene, sin embargo dichos formularios, aunque pueden ser leídos por Flask®, no son los que suelen utilizarse cuando se utiliza este lenguaje de programación, se tienen formularios especiales optimizados para su uso mediante una extensión del *framework* llamado *WTForms* que incluye opciones de validación de formularios y su renderización de una manera muy fácil por medio de objetos de Python®.

El script de Python® mostrado en Figura 40, presenta el código para la declaración de formularios utilizando *WTForms*, dicho script se recomienda escribirlo en un archivo Python® diferente para que sea llamado como una librería desde el código principal y que éste no contenga un gran número de declaraciones.

```
from wtforms import Form
from wtforms import IntegerField

class CommentForm(Form):
        Tiempo = IntegerField("Tiempo")
        Muestras = IntegerField("Muestras")
```

Figura 40. Clase de formularios en Python®.

Dentro del código HTML se pueden colocar variables u objetos que se creen en Python® colocando el nombre entre dobles llaves ({{variable}}) incluidos los formularios, que deberán ser llamados como atributos de la clase *CommentForm()* que deben ser declarados desde el programa principal. La manera de introducir formularios en el código HTML se muestra en la Figura 41.

```
<td width="195">Tiempo de Muestreo(ms)</td>
<td width="175"> {{form.Tiempo}}</td>
```

Figura 41. Introducción de formularios al HTML mediante Flask®.

El archivo HTML modificado genera la página web que se muestra en la Figura 42, es una página sencilla que se utilizó para realizar las pruebas de formularios y renderización de páginas. Para una mejor presentación pueden utilizarse tablas de HTML si se quiere que todos los campos que se utilicen en los formularios contengan las mismas medidas.

Figura 42. Página web de prueba.

Para leer el contenido de los formularios, es necesario que llegue al servidor una petición tipo POST, encargada de enviar peticiones que no son visibles al cliente, es decir, no generan una URL distinta si no está programada.

Las funciones definidas en Python® se encuentran programadas de manera automática para peticiones tipo GET, siendo necesario indicarles que se puede acceder mediante otro método a dicha función como se ve en la Figura 43.

```
@app.route('/',methods= ['GET', 'POST'])
def index():
        form = forms.CommentForm(request.form)
        if request.method == 'POST':
                time = form.Tiempo.data
                sample = form.Muestras.data
                print time
                print sample

        return render_template("index.html",form = form)
```

Figura 43. Lectura de formularios en Python®.

Se observa en la imagen anterior, que debe agregarse como argumento de la ruta el tipo de peticiones que puede leer, si llega una petición POST, es decir se ha enviado el formulario, el contenido de éstos se guarda en variables que pueden ser utilizadas por el script de Python® con la necesidad de cambiar de ésta a la que se requiera

3.1.6 ADAPTACIÓN DE LA TRANSMISIÓN DE VIDEO A LA PÁGINA WEB

Una de las partes más importantes en la realización del proyecto es la transmisión de video del proceso en todo momento que se estén realizando las pruebas con él. Se buscaron aplicaciones que permitieran la inclusión de la transmisión sin un gran retardo en la imagen y sin aplicaciones externas o difíciles de instalar.

Debido a que existen proyectos de domótica para el uso de vigilancias y alarmas se encuentran en la red diferentes aplicaciones, siendo MJPG - Streamer la aplicación que se ajustaba mejor a las necesidades del proyecto. MJPG - Streamer es una aplicación desarrollada para la transmisión de video en un servidor web, sin embargo, no se transmite video como tal, la aplicación genera un tren de imágenes en formato jpg que se irán refrescando dentro de la página lo que permite que se vea con fluidez.

Utilizando la cualidad de ser trenes de imágenes, el código HTML para utilizar la transmisión es bastante sencillo, se agrega una etiqueta de imagen con sus dimensiones y se coloca en la fuente la URL que contiene al *streaming* de video. El ejemplo de transmisión se presenta en la Figura 44.

Figura 44. Ejemplo de transmisión de video.

La adaptación del código HTML para ingresar las imágenes generadas por MJGP-Streamer se muestra en la Figura 45.

```
<p>Bienvenido al streaming de video del proceso donde puedes visualizar el sistema mientras se realizan las pruebas
</p>
<center>
  <img src="http://172.16.37.176:8083/?action=stream" width="640" height="480"/>

</center>
```

Figura 45. Adaptación del HTML para el ingreso de la transmisión.

3.1.7 DISEÑO DE LA ESTRUCTURA

El levitador de aire consiste básicamente en un tubo plástico con una fuente de aire debajo que permite elevar una esfera de unicel o algún material que pueda flotar con la presión de aire que se genera.

Se realizó un plano en 3D con la herramienta *SketchUp*® de google para diseñar un prototipo de bajo costo y que pudiera ser fabricado son necesidad de herramientas especiales y con materiales de fácil accesibilidad.

El diseño consta básicamente de un cubo en la parte inferior con dos de sus caras descubiertas para la circulación de aire y dentro de él, el ventilador y el tubo plástico. La parte superior consta de una placa donde reposa el tubo y el sensor de distancia. En la Figura 46 se muestra el prototipo diseñado.

Si bien las piezas necesarias pueden fabricarse con madera u otro material, se recomienda que sea un material rígido que pueda soportar el peso del tubo y la base de la parte superior para que la estructura se mantenga firme en la superficie.

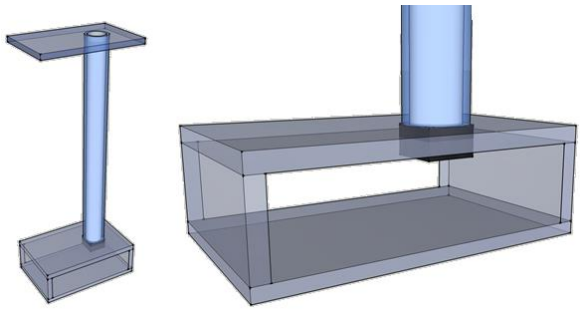

Figura 46. Diseño en 3D del levitador de aire en Sketchup®.

La elaboración del proceso se realizó por medio de placas de acrílico de 7mm de espesor. Para la base se utilizaron dos placas de 140mm de largo y 90mm de ancho y dos soportes de 90mm de largo y 45mm de ancho.

El tubo plástico utilizado fue un sifón de pecera de 80cm de largo y con un ancho de una pulgada debido al alto costo de tubos de materiales como polietileno o acrílico, obteniendo resultados satisfactorios pues al no tener mucho peso no afecta en la estabilidad de la estructura. En la parte superior de colocó una placa de acrílico con las mismas dimensiones que la base. Se puede observar el prototipo fabricado en la Figura 47.

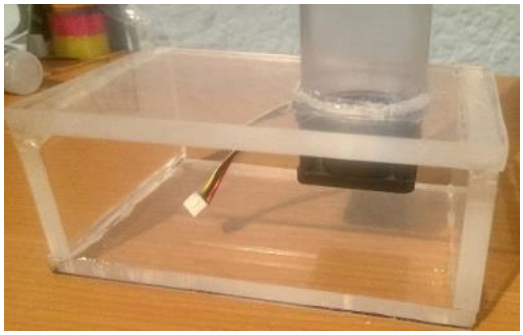

Figura 47. Prototipo construido.

Como fuente de aire se utilizó un motor *brushless* de refrigeración de equipo electrónico de 24VCD SANYO® Fine ACE 20 que se muestra en la Figura 48.

Figura 48. Motor para la fuente de aire.

3.1.8 PÁGINA WEB DEL SERVIDOR

3.1.8.1 Contenido estático

Para la realización de la página web se utilizó Dreamweaver® que permite hacer plantillas para personas que no tienen conocimientos avanzados de diseño web. Se diseñó una página sencilla y de fácil navegación con la información teórica básica necesaria para el modelado y sintonización del controlador PID.

En el index se encuentra la introducción de la plataforma junto con los apartados que contienen los componentes teóricos y el desarrollo de las prácticas como se muestra en la Figura 49. Se agregaron también algunos elementos como los son los logotipos de las instituciones educativas a las que se pretende brindar la herramienta para la realización de las prácticas, así como la insignia de la investigación de Instrumentación Virtual Remota.

Figura 49. Index de la página web.

El diseño de la página se basó en los colores del Instituto Tecnológico Superior de Lerdo y fue diseñada con el fin de que no fuese complicado navegar dentro de ella.

3.1.8.2 Contenido Dinámico de la Página Web

Para el contenido dinámico se optó por realizar una página dividida en dos partes ya que las pruebas realizadas dieron como resultado el impedimento de tener los formularios y la transmisión en una sola página web, ya que al adquirir los datos la transmisión se pausaba por lo que no podía verse el funcionamiento del proceso. En la parte superior presenta los formularios para el envío de las muestras y el tiempo de muestreo, mientras que debajo se observa la transmisión de video como se puede ver en la Figura 50, ambas páginas, tanto superior como inferior, son independientes sin embargo se observan en una sola lo que permite que no se afecten entre ellas.

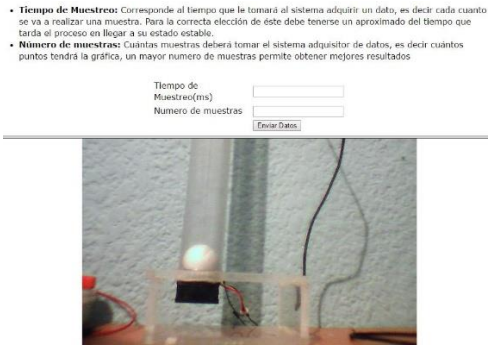

Figura 50. Sistema en transmisión.

3.1.9 PRUEBA DE ADQUISICIÓN DE DATOS.

Una vez realizado el muestreo, los resultados se presentan al usuario de dos maneras distintas, la primera una gráfica de datos contra el tiempo de muestreo que puede utilizarse

para métodos al tanteo de los controladores. La segunda entrega un archivo xls (Excel®) con los datos muestreados y el tiempo de muestreo que se ha utilizado para que éstos sean utilizados para realizar análisis de datos con un software especializado para el diseño de sistemas como MATLAB® o Scilab®. Para realizar lo anterior se utilizó la librería para Python XLS que permite generar archivos de Excel a partir de listas de Python®.

Se tuvieron errores al momento de ingresar las gráficas y el archivo xls generadas en la página web, pues se almacenaba la última imagen en el caché del navegador por lo que las imágenes no se refrescaban, así como tampoco los archivos de Excel®, siendo necesario implementar un código que permitiera que cada uno de los entregables tuviera un nombre diferente.

El método propuesto y con el que se tuvieron resultados satisfactorios fue nombrar los archivos con la fecha y hora en la que fueron generados, permitiendo generar ficheros diferentes cada una de las muestras y que éstas se mantengan almacenadas en el servidor. Al utilizar este método es necesario decirle al HTML que cargue la imagen y fichero correspondiente a la muestra que se ha generado.

Fue realizada una prueba utilizando el levitador, para observar el comportamiento del mismo y que las gráficas fueran generadas correctamente, el experimento se muestra en la Figura 51. Con un muestreo cada 100 milisegundos y tomando 100 muestras.

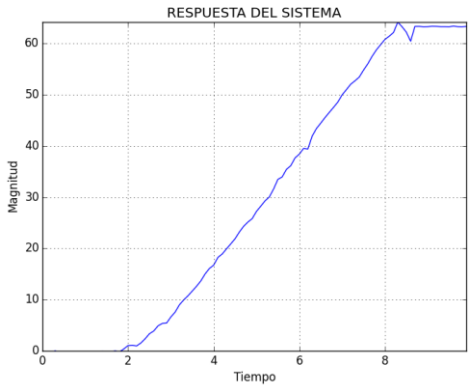

Figura 51. Prueba de muestreo con el levitador

Las pruebas se realizaron con diversas pelotas de unicel de varios tamaños y una pelota plástica, siendo esta última demasiado pesada para que la presión del ventilador pudiera levantarla a voltajes bajos.

Utilizando el servidor web, se realizaron las pruebas para verificar el funcionamiento del sistema con el levitador. Fueron tomadas 4 muestras para ser promediadas mediante los entregables que genera la página web y obtener los datos necesarios para obtener el modelo.

El promedio de los de los datos lo muestra la Figura 52, dicho promedio fue utilizado utilizando Excel® ya que todos los entregables se entregan en este formato. Posteriormente fueron analizadas utilizando le programa en Matlab para obtener el modelo matemático, se realizó en esa plataforma para poder utilizar las herramientas se sintonización y observar de mejor manera los lazos de control. Sin embargo, debido a la falla en el modelo matemático aplicar los algoritmos representaba una tarea difícil.

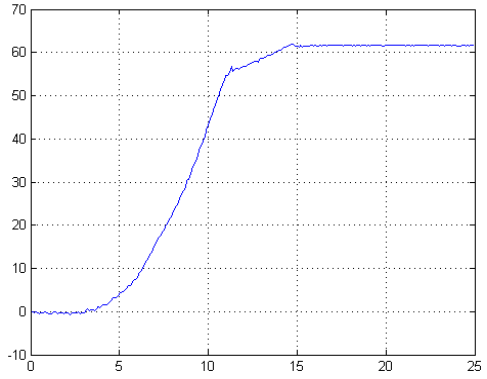

Figura 52. Promedio de muestras del Levitador

Fue utilizada la herramienta de sintonización de Simulink® con la cual se obtuvieron los valores de las constantes del PID, sin embargo, se modificaron manualmente para obtener la respuesta deseada basándose en los datos obtenidos que, posteriormente, fueron ingresadas al código de Python® que calcula el valor de la variable de control. Fue colocada una cinta métrica como referencia.

La Figura 53 muestra el controlador PID funcionando llegando de forma muy aproximada a la referencia colocada, el error observable es debido a la desviación que existe entre el modelo real de la planta y el modelo matemático aproximado, dicho error genera un sobreimpulso demasiado alto que no es deseable en la planta. La referencia colocada fue de 20cm, siendo muy cercano el valor obtenido físicamente.

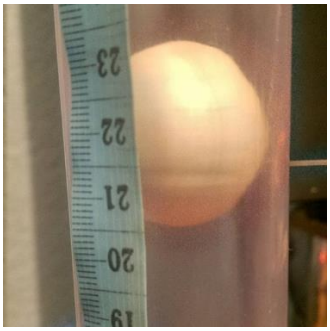

Figura 53. Referencia de 20cm.

Las pruebas fueron realizadas a varias referencias obteniendo resultados satisfactorios, se crearon perturbaciones externas como impedir el flujo de aire en la parte superior e inferior del levitador para observar si llegaba a la referencia a pesar de ellas y se obtuvieron buenos resultados como se muestra en la Figura 54, donde se colocó una referencia de 30cm.

Figura 54. Referencia a 30cm.

Habiendo realizado el análisis de la metodología utilizada, se determinó que los elementos de hardware y software no eran lo más óptimo para la plataforma. Entre las razones de software es que, si bien la plataforma entrega las gráficas y resultados de las pruebas, así como la transmisión, la página web no es dinámica y no permite ver en tiempo real la adquisición de datos, así mismo cada que se requiere un formulario o un dato la página tiene un refrescamiento que puede ocasionar problemas en el sistema en un futuro.

En el hardware se concluyó que la velocidad de procesamiento de datos del Arduino® no permite que los controladores funcionen de manera óptima y no pueden utilizarse todos los recursos del microcontrolador.

3.2 LEVITADOR DE AIRE CON NODE-RED® Y PIC18F2550®

De acuerdo a la retroalimentación del prototipo anterior con las plataformas de hardware y software descritas se optó por el cambio de ambas, dado que, si bien funcionan correctamente de manera conjunta, éstas no tienen las características óptimas para dar una buena experiencia al usuario final. Los cambios de hardware que se realizan se enlistan a continuación:

- Elemento de control (motor): se realiza el cambio de un brushless para refrigeración de equipos de cómputo a un motor de corriente directa, esto debido a que el control de este tipo de motor es más sencillo que el anterior.
- Sensor de distancia: Si bien el sensor HC-SR04 cumple con los requerimientos necesarios, tu tamaño no permite que el diseño estructural pueda optimizarse, por lo que se cambia por un sensor láser VL53l0x que es utilizado para vehículos no tripulados. El elemento de medición es el que se muestra en la Figura 55.

Figura 55. Sensor VL5310x.

- Microcontrolador: Al hacerse las pruebas con la plataforma Arduino®, se encontraron limitaciones al momento de utilizarse recursos que son importantes en la realización de controladores como lo son las interrupciones y los *timers*, por lo que se decide el uso del microcontrolador PIC18F2550®, su distribución puede verse en la Figura 56.

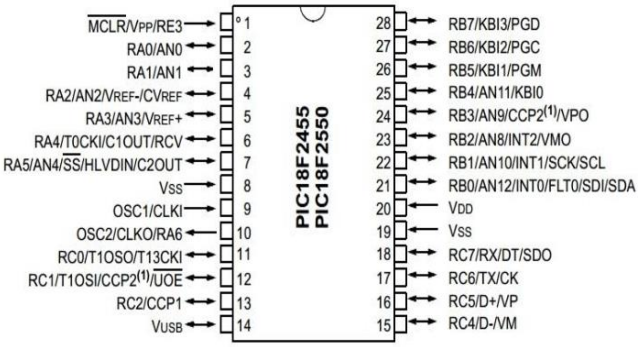

Figura 56. Distribución PIC18F2550®.

3.2.1 DISEÑO ELECTRÓNICO

Una de las desventajas al momento de crear el primer prototipo del levitador de aire fue la elección del elemento final de control que era un motor de ventilación de equipos de cómputo tipo brushless, esto derivaba en poco rango útil de uso de acuerdo al ciclo de trabajo de la modulación por ancho de pulsos.

Debido a esto se optó por cambiar la plataforma para que trabajara con motores de corriente directa de escobillas que tiene un comportamiento más lineal y se puede controlar de diferentes formas.

Para ello se diseñó un circuito basado en el microcontrolador PIC18F2550® y un puente H L298 que es el más utilizado para el cambio de giro y control de potencia del motor. Este circuito integrado permite controlar la velocidad del motor de acuerdo al nivel de la modulación de pulsos que entre a sus terminales de habilitación.

Así mismo cuenta con protecciones a las cuales deben agregarse elementos externos como diodos de recuperación rápida. El circuito diseñado se muestra en la Figura 57.

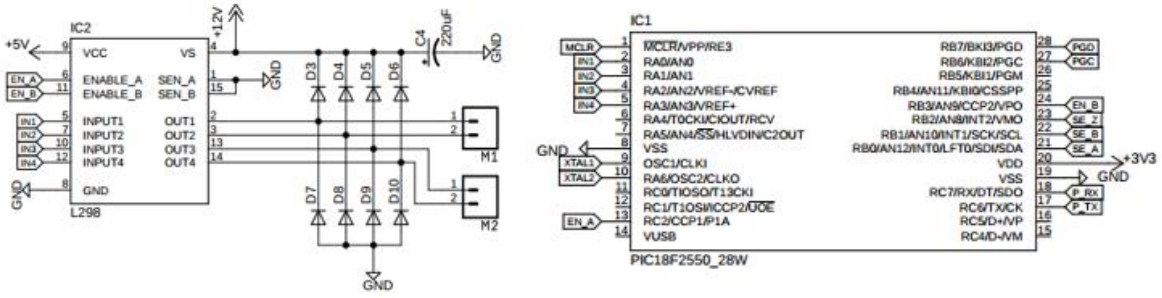

Figura 57. Circuito para el control de motores de C.D.

Las salidas de PWM que tiene el microcontrolador PIC18F2550® están conectadas a la entrada de habilitación del puente H para poder controlar hasta dos motores de corriente directa. Mientras que las entradas de selección de giro están a pines digitales del microcontrolador. En adición a esto el microcontrolador tiene las salidas y entradas de transmisión de puerto serie hacia la Raspberry Pi® para realizar la comunicación bidireccional.

Para el sensor láser se utilizaron dos conectores, uno que será la alimentación y otro que corresponde a los pines de la comunicación i2c utilizada para la comunicación con el microcontrolador.

Con los circuitos diseñados se realizó la fabricación del circuito impreso en tecnología de montaje superficial, para que fuera apilable con las placas Raspberry Pi®® modelos B y no existieran problemas de conexión y el tamaño fuera reducido. La placa de circuito impreso montada es la que se muestra en la Figura 58.

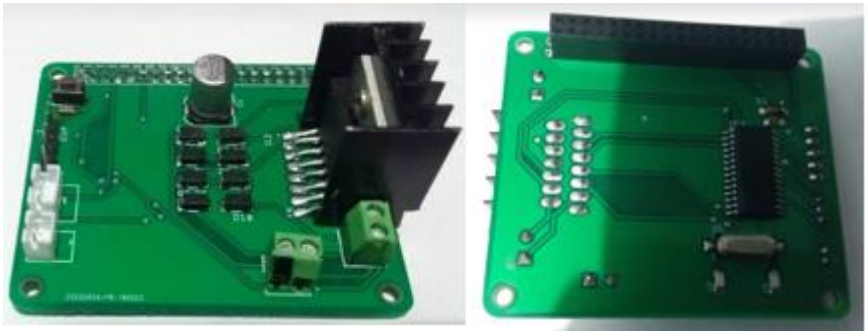

Figura 58. Circuito montado para el control del levitador.

3.2.2 DISEÑO ESTRUCTURAL

De los principales problemas al momento de realizar el prototipo del levitador fue que se realizó con herramientas manuales, siendo difícil la construcción y montaje. Así mismo dichos desperfectos alteraban el comportamiento del sistema. Por lo que la nueva estructura se realizó mediante un torno de control numérico y utilizando como programa de diseño Fusion 360 de Autodesk® en su versión con licencia para estudiantes.

Como primer punto se realizó el diseño de los elementos de hardware con los que ya se contaba, es decir el ventilador y el sensor láser con la finalidad de realizar una simulación de montaje dentro del software que facilitara su ensamble físicamente. El diseño se ve en la Figura 59.

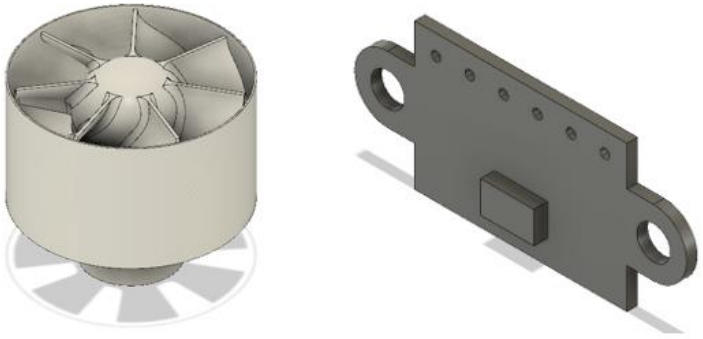

Figura 59. Diseño 3D del ventilador y sensor.

A partir de los modelos anteriores se comenzó con la creación de los demás elementos que fijarán a la estructura. Se creó una base para el tubo de acrílico y debajo de él unas arandelas que fijarán el motor a la base para que se pierda la menor cantidad de aire entre una unión y otra. En la parte superior se realizó una tapa para el tubo con una abertura donde se encuentra montado el sensor láser. El diseño terminado es el que se muestra en la Figura 60.

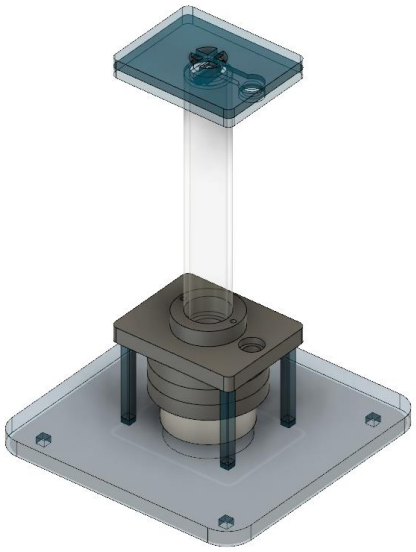

Figura 60. Modelo 3D del levitador.

Con el modelo terminado se definió como material para la elaboración de la estructura física el polietileno de alta densidad (HDPE) ya que es blando pero rígido y fácil de maquinar con el torno. La lámina de HDPE cuenta con un grosor de 0.6mm por lo que todas las piezas diseñadas anteriormente tomaron en cuenta esta consideración. Posteriormente se maquinaron cada una de las piezas utilizando el CNC que se ve en la Figura 61.

Figura 61. Fabricación de las piezas en CNC.

3.2.3 IMPLEMENTACIÓN DEL PROTOCOLO DE COMUNICACIÓN EN PIC18F2550®.

Al cambiar de microcontrolador es necesario realizar nuevamente la programación del protocolo de comunicación, ya que Arduino® maneja un lenguaje orientado a objetos con diferentes estructuras de programa que la programación en C que utilizan los microcontroladores PIC de Microchip®. De acuerdo a las pruebas realizadas en el prototipo se establece el protocolo de comunicación que se muestra en la Figura 62.

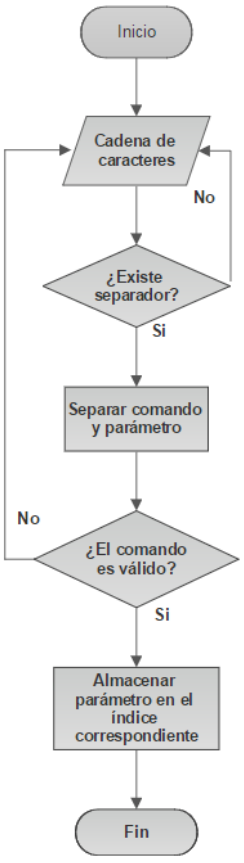

Figura 62. Diagrama de flujo del protocolo de comunicación.

El diagrama anterior es el que se implementará tanto en el microcontrolador como en la plataforma de recepción de datos. Se realizó de manera que pudiera ser replicado en cualquier lenguaje de programación que se quiera utilizar y siendo lo más sencillo posible.

Es de tomar en cuenta que el protocolo es diseñado e implementado por el programador y puede ser cambiado de acuerdo a las necesidades que se tengan durante la creación del programa final.

El lenguaje de programación utilizado para la programación del microcontrolador en PIC-C compiler o CCS® que, si bien no es el oficial de la marca, es de los más utilizados por las funciones que ya se encuentran definidas dentro del compilador.

Para la implementación se definen primero las variables a utilizar que corresponden al carácter del separador, un arreglo de caracteres que va a almacenar los datos que provienen del puerto serie y dos punteros que corresponderán a las cadenas de caracteres que se van a separar. Por último, un índice que va a permitir encontrar los datos buscados dentro del arreglo. La Figura 63 muestra las variables declaradas.

```
char separador[] = "@";
char rxBuffer[32];
char *comando;
char *parametro;
int  rxIndex =0;
```

Figura 63. Variables para el protocolo de comunicación en CCS®.

Posteriormente se hace la matriz de comandos de la Figura 64 que enlista todos los comandos que el usuario desea implementar. Se pueden tener tantos comandos como elementos puedan introducirse en esta matriz dando una gran variedad de combinaciones y comandos que pueden hacer el protocolo tan extenso como el programador lo requiera.

```
#define CMD_SIZE 15
#define NUM_CMD  13

char CMDS[NUM_CMD][CMD_SIZE]=
{
   "SetSamples", // DATA[0]
   "SetTime",    // DATA[1]
   "SetPWM",     // DATA[2]
   "GetDist",    // DATA[3]
   "SendData",   // DATA[4]
   "LedOn",      // DATA[5]
   "LedOff",     // DATA[6]
   "pwmnow",     // DATA[7]
   "setkp",      // DATA[8]
   "setti",      // DATA[9]
   "settd",      // DATA[10]
   "setpoint",   // DATA[11]
   "mode"        // DATA[12]
};

float DATA[NUM_CMD];
```

Figura 64. Declaración de comandos en CCS®.

Así mismo se crea un arreglo de tipo flotante que va a almacenar todos los parámetros o valores de cada uno de los comandos a implementar si es que éstos lo requieren. Es importante definir desde un principio cuantos comandos se van a utilizar, así como el tamaño de los mismos, pues de esto depende que el protocolo funcione correctamente al momento de realizar las iteraciones de búsqueda.

Una de las diferencias entre Arduino® y el PIC18F2550® es el uso de diferentes interrupciones. Cuando se implementa el protocolo en Arduino® no se tiene con certeza una interrupción que índice la recepción de un dato por el puerto serie, lo que puede originar pérdida de información y por ende un mal funcionamiento de la comunicación. Dado que la comunicación es prioridad en el programa, se realiza utilizando esta característica del microcontrolador PIC18F2550®, que detiene la ejecución del programa no importando donde se encuentre y se dirige a recibir el carácter proveniente del puerto serie, disminuyendo las fallas. La función de interrupción se ve en la Figura 65.

```
#INT_RDA
void RDA_ISR()
{
   rxBuffer[rxIndex]=getc();
   if(rxBuffer[rxIndex] == '/')
   {
     comando = strtok(rxBuffer,separador);
     parametro = strtok(NULL,separador);
     valor = atof(parametro);
     rxIndex=0;
   }
   else
   rxIndex++;
}
```

Figura 65. Función de interrupción por puerto serie en CCS®.

La función consiste en guardar el carácter que se lee del buffer de recepción, posteriormente se pregunta si éste es una diagonal invertida (fin de la trama), si es así se busca el separador y se clasifica el comando y el parámetro y reinicia el conteo del tamaño del arreglo. En caso contrario se va incrementando el índice hasta que se encuentren las letras especificadas.

Cuando se han clasificado el parámetro y el comando, el primero se transforma a su equivalente en valor numérico para posteriormente ser utilizado en operaciones matemáticas o asignado a un valor de un periférico del microcontrolador.

Si bien se han clasificado las dos tramas, es necesario saber si éstas son comandos válidos o no. Se creó la función FindCommand para que realizara esta tarea y es la que se presenta en la Figura 66.

```
int FindCommand(char *cmd, char *par)
{
   char buf[CMD_SIZE];
   int i;

   for (i=0; i<NUM_CMD; i++)
   {
      strcpy(buf, &CMDS[i][0]);
      if (stricmp(buf, cmd)==0)
      {
         DATA[i]=atof(par);
         return(i);
      }
   }
   return(0xFF);
}
```

Figura 66. Función de búsqueda de comando en CCS®.

La función pide como argumentos de entrada el comando y el parámetro que anteriormente se ha clasificado. Se crea un arreglo temporal que permitirá trabajar con los datos que se tienen.

El bucle for es el encargado de realizar las iteraciones (tareas continuas) y es por ello que anteriormente se deben de definir las características de los comandos como lo es el tamaño de éste.

La función copia en el buffer temporal el contenido de la matriz de comandos de acuerdo a la posición del índice. Una vez realizada la copia se hace la comparación de ésta con lo que se ha colocado como argumento a la función de búsqueda.

Si la comparación es exitosa se trata de un comando válido y se guarda el parámetro en la posición correspondiente y se devuelve el índice o la posición del comando que se ha encontrado para su posterior identificación. Si no es un comando válido se devuelve un 255.

Con el índice devuelto basta con realizar una comparación entre éste y la posición en la matriz del comando asignado para saber cuál es éste y que tareas se deben de realizar con la información que se ha recabado, ya sea activar algún elemento discreto y asignar valores numéricos a los diferentes elementos que se encuentren programados.

3.2.4 FUNCIONES DE MUESTREO Y MEDICIÓN DE DISTANCIA

Una vez establecido el protocolo de comunicación en el microcontrolador, es necesario ahora realizar las funciones que harán el intercambio de datos entre los elementos del sistema de control con la Raspberry Pi®.

En primer lugar, se programó el sensor láser utilizando una librería modificada de la que el fabricante brinda libremente ya que ésta se encuentra basada en un lenguaje orientado a objetos.

Dicha modificación tiene como nombre *vl53l0x-ccs* del autor Arnan Roger Sipitakiat y se encuentra como uso libre en la red en plataformas como GitHub en donde se realizan intercambios de información con licencia de *open source*. Sin embargo, se hicieron modificaciones debido a la versión del programa que se está utilizando.

El código para obtener la distancia es la que se presenta en la Figura 67, en ella ya se utilizan la función *readRangeSingleMillimeters* que se encuentra definida en la librería y que retorna el valor en milímetros de lo que mide el sensor láser y posteriormente se realiza la conversión a centímetros.

El valor que retorna la función es la diferencia que existe entre la distancia del tubo de acrílico y la medida pues se tiene que tomar en cuenta que el sensor se encuentra en la parte superior de la estructura.

```
float Get_Data()
{
  int16 m_data;
  float distancia;

  m_data = readRangeSingleMillimeters();
  distancia = (float)m_data;
  distancia = distancia/10.0;
  distancia = 79.0-distancia;
  return distancia;
}
```

Figura 67. Función para la obtención de distancia en CCS®.

El sistema debe de ser capaz de realizar un número de muestras determinadas a un tiempo establecido por el usuario, así como controlar el incremento en el escalón de acuerdo a las pruebas que se requiera realizar. Por ello se definen instrucciones que permitan, mediante los comandos, establecer dichos parámetros y que se devuelva la información necesaria.

El código para la función de muestreo que se ha establecido es el que se observa en la Figura 68, en la función se establece el ciclo de trabajo del PWM al establecido por los parámetros indicados y se utiliza un ciclo for con iteración hasta completarlas muestras dadas y que envía el parámetro de la distancia y del PWM con la finalidad de hacer los gráficos como son debidos.

Para el tiempo de muestreo se crea la función millis que permite realizar retardos en milisegundos dando como argumento un número entero, tarea que no puede realizar la función de retardo establecida de forma predeterminada por el compilador. Una vez terminada la muestra se imprime un carácter de final de envío.

```
void Get_Samples(int16 muestras, int16 tiempo)
{
   int16 i=0;
   float distancia;

   set_pwm2_duty((int16)DATA[2]);

   for(i=1;i<=muestras;i++)
   {
     printf("dist@%.2f/\r\n",Get_Data());
     printf("pwm@%ld/\r\n",(int16)DATA[2]);
     delay_milis(tiempo);
   }
   printf("&\r\n");
   set_pwm2_duty((int16)DATA[7]);
}
```

Figura 68. Función para obtención de muestras en CCS®.

Para el uso del sensor es necesario crear una rutina de conteo de milisegundos pues se va a tomar como referencia para los cálculos internos en la programación del sensor láser. Se realiza mediante interrupción ya que la precisión es prioridad en esa tarea para obtener mediciones más exactas a las reales. La interrupción programada se ve en la Figura 69. Y está calculada con una frecuencia de reloj de 20MHZ utilizada en el microcontrolador.

```
#INT_TIMER1                    //1ms interrupt
void  TIMER1_isr(void) {
   msTimer++;
   set_timer1(5000);
}
```

Figura 69. Retardo por interrupción en CCS®.

3.2.5 PROTOCOLO DE COMUNICACIÓN EN NODE-RED®

A partir de los resultados obtenidos en el prototipo del levitador utilizando como herramienta de interfaz la API Flask® para Python®, se determinó que, si bien funciona de manera conjunta con Arduino® en la adquisición de datos, presenta desventajas en la experiencia al usuario, ya que todos los controles, así como gráficas son estáticas y no cambian hasta que exista una recarga de la página web. Por lo anterior, se cambió a la plataforma Node-Red® que permite el uso de controles dinámicos y da una mejor apariencia a la página web donde se van a realizar las prácticas.

Node-Red® es una plataforma de programación visual de software libre que se basa en la interconexión de nodos de programación. Permite interactuar hardware y software fácilmente. Cada nodo se comunica con el otro mediante un mensaje llamado *msg.payload* que puede describirse como el valor retornado de una función. Se basa en NodeJs® y en el lenguaje JavaScript®.

Así como en las demás plataformas, es necesario realizar el protocolo de comunicación para decodificar los datos recibidos del microcontrolador, así como realizar la codificación para la comunicación bidireccional. Los nodos para realizar la lectura del puerto serie se pueden ver en la Figura 70.

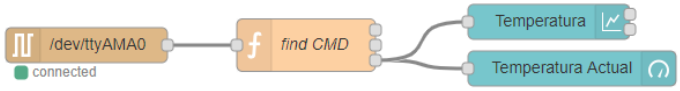

Figura 70. Nodos para lectura de puerto serie en Node-Red®.

El primer nodo de la izquierda corresponde a la librería para el manejo de puerto serial con Node-Red®, el segundo es una función en JavaScript® de autoría propia para la decodificación de los mensajes provenientes del puerto serial. Dentro de la función es necesario crear las variables que se utilizan en el protocolo de comunicación y se ven en la Figura 71.

```
var cadena = msg.payload;
var indice,cmd,par,len;

var msg={};
var msg1={};
var msg2={};
```

Figura 71. Asignación de variables de protocolo en Node-Red®.

En cadena se almacena el mensaje proveniente del nodo de puerto serie, es decir, el mensaje que ha sido enviado desde el PIC18F2550®. Se crean tres mensajes, que serán los encargados de clasificar los parámetros de los comandos encontrados, como se ha mencionado anteriormente, Node-Red® se basa en el intercambio de mensajes a través de los nodos, es por esto que se mandan en arreglos ordenados, de manera que el nodo siguiente pueda leerlos y decodificar la información que ha recibido.

Con las variables asignadas se procede a realizar el algoritmo de búsqueda siguiendo el diagrama de flujo descrito anteriormente, comprobando que a partir de él se puede realizar el protocolo en cualquier lenguaje de programación que contenga funciones o métodos parecidos para el manejo de cadena. El algoritmo implementado en JavaScript® para la decodificación se ve en la Figura 72.

```
len = cadena.length;
indice = cadena.indexOf("@");
cmd=cadena.substring(0,indice);
par=cadena.substring(indice+1,len-1)
```

Figura 72. Código de decodificación de mensaje en JavaScript®.

Una vez que se han separado los comandos y los parámetros se realiza la clasificación para la creación de cada uno de los mensajes como se ve en Figura 73.

```
if(cmd == "dist")
{
    msg.payload = par;
    msg.topic="Temp";
    return [ null,null, msg ];
}
else if(cmd == "pwm")
{
    msg1.payload = par;
    msg1.topic="PWM";
    return [null, msg1, null];
}
else if(cmd == "error")
{
    msg2.payload = par;
    msg2.topic="ERROR";
    return [ msg2, null,null];
}
```

Figura 73. Clasificación de parámetros en JavaScript®.

De acuerdo al comando que se ha enviado desde el microcontrolador, es el orden que en que se ordenan los parámetros en el arreglo de salida.

Para el envío de datos desde la interfaz hacia el dispositivo, se diseñó el *dashboard* que se muestra en la Figura 74, en donde el usuario colocará los parámetros que requiere para realizar el muestreo de las señales provenientes del levitador.

Figura 74. Control para muestreo de datos.

Los controles devuelven un valor numérico en su mensaje, por lo que es necesario transformarlos en valores de cadena. Para ello primero se transforman los mensajes provenientes de los nodos de control numérico y se asignan a variables globales con la finalidad de poder ser leídas en funciones de otros nodos sin que éstos se encuentren conectados entre sí.

Los nodos requeridos para la asignación de los valores en variables globales se ve en la Figura 75, así como los nodos utilizados para la creación de la interfaz mostrada anteriormente. Los nodos en color azul corresponden a los elementos visuales del *dashboard* y los naranjas a las funciones que son programadas por el usuario en JavaScript®.

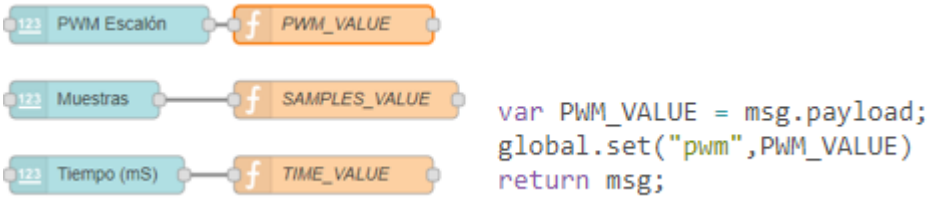

Figura 75. Ejemplo de asignación de variable global en Node-Red®.

Una vez con las variables globales asignadas, se procede a realizar las tramas necesarias para el protocolo de comunicación a partir de los valores recién obtenidos en los nodos anteriores. Para ello se utilizan los nodos que se ven en la Figura 76.

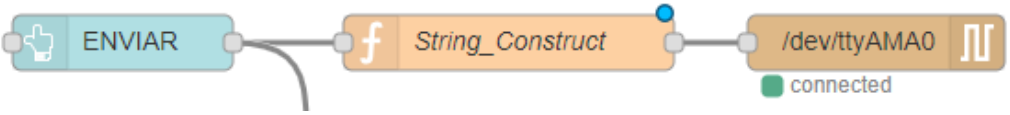

Figura 76. Nodos para construcción de tramas en Node-Red®.

El nodo Enviar corresponde al botón que se encuentra en el *dashboard* y de acuerdo al valor de este se ejecutarán las acciones programadas en la función *String_Construct*. El código implementado en la función se presenta en la Figura 77.

```
var test = msg.payload;
var tiempo = global.get("time");
var muestras = global.get("samples");
var pwm_valor = global.get("pwm")
var cadena_tiempo,cadena_muestras,cadena_pwm;

if (test == true)
{
    cadena_tiempo = "settime@"+ tiempo.toString()+"/";
    cadena_muestras = "setsamples@"+ muestras.toString()+"/"
    cadena_pwm = "setpwm@"+ pwm_valor.toString()+"/"
    msg.payload = cadena_tiempo+cadena_muestras+cadena_pwm+"senddata@/";
}
return msg;
```

Figura 77. Código para construcción de tramas en JavaScript®.

En primer lugar, se lee el valor del botón que se asigna a la variable *test*, a partir de ella se determina si se envían o no las tramas que se van a construir.

Para la construcción de tramas se leen los valores de las variables globales asignados anteriormente. Posteriormente se crean las tramas con los comandos que se han colocado en la matriz del microcontrolador, es importante remarcar que ambos deben de ser los mismos, de lo contrario el protocolo no funcionará. Se observa que en primer lugar en envía el comando y su separador y enseguida el valor del parámetro proveniente de la interfaz visual en Node-Red®. Con esto se termina con la programación básica de la comunicación bidireccional entre la Raspberry Pi® y el microcontrolador PIC18F2550®.

3.3 DISEÑO DEL SISTEMA DE CONTROL DE TEMPERATURA

Con la finalidad de realizar diversos sistemas se optó por la creación de un sistema de control de temperatura, ya que se acuerdo a la literatura y proyectos vistos en la investigación previa a la creación de este proyecto, es el más utilizado en la enseñanza de sistemas de control por ser de los más lineales y estables.

El sistema de control de temperatura será una simulación de un horno, utilizando como elemento calefactor una lámpara incandescente automotriz que, al calentarse, elevará su temperatura en proporción al voltaje que pase a través de sus terminales. La Raspberry Pi® solamente funcionará como interfaz realizando la comunicación con el controlador mediante el uso del protocolo serie.

3.3.1 DISEÑO ELECTRÓNICO DEL SISTEMA

Con el fin de evaluar diferentes plataformas el sistema se basó en un microcontrolador PIC18F2550® de Microchip® que presenta características más avanzadas al Atmega328p® utilizado anteriormente. Soluciona la mayoría de las deficiencias vistas en el prototipo como la velocidad de procesamiento, el acceso a las interrupciones y timer de manera completa y sin limitaciones. La Figura 78 muestra el diagrama esquemático y algunas funciones del microcontrolador PIC18F2550®.

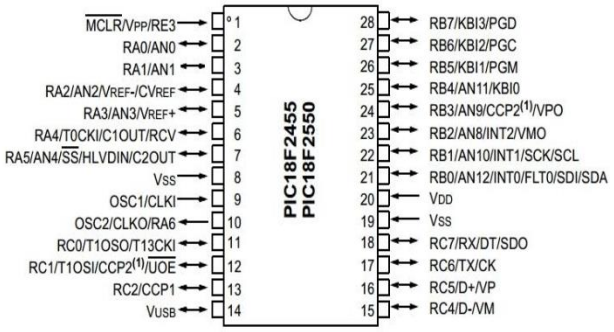

Figura 78. Diagrama del Microcontrolador PIC18F2550®.

El elemento de medición seleccionado es un sensor de temperatura semiconductor LM35 que proporciona una salida proporcional y lineal a la temperatura de 10mV por grado centígrado. Si bien ya tiene su salida en voltaje es necesario amplificarlo para tener una mejor resolución del conversor análogo digital.

Tomando como referencia que la temperatura máxima de lectura del sensor es de 150°C, el voltaje máximo que entrega será de 1.5V. Dado que el voltaje de entrada máximo del microcontrolador es de 5V se realizó una división con la finalidad de conocer la ganancia que debería entregar el amplificador no inversor dando el valor de 3.33.

Posteriormente se realizó el diseño del amplificador no inversor utilizando la ecuación 26.

$$G = \left(1 + \left(\frac{R_2}{R_1}\right)\right) \tag{26}$$

Proponiendo un valor para la resistencia R2 de 4.7k, se despeja el valor de R1 dando como resultado 2.2K, obteniendo una ganancia de 3.16 que es muy cercana a la requerida para el uso total del conversor. El circuito diseñado para el sistema de acondicionamiento de la señal se puede ver en la Figura 79.

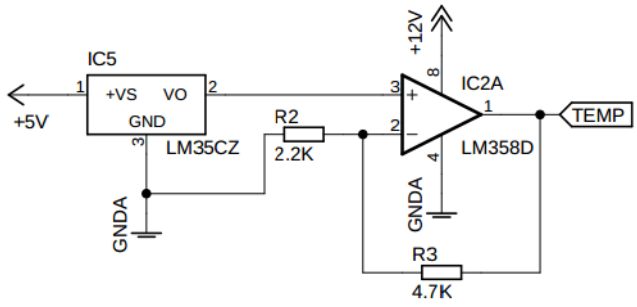

Figura 79. Acondicionamiento de señal para el sensor LM35.

El microcontrolador no puede manejar cargas de más de 5V y de un consumo de corriente cercano a los 20mA, se requiere de una etapa de potencia para el manejo de la lámpara incandescente, pues al ser un elemento calefactor debido a la disipación de potencia del filamento, tiene un consumo de corriente elevado.

Para el diseño del circuito de potencia se toma en cuenta que la salida variable que se tiene con el microcontrolador es una señal de modulación de ancho de pulsos que por sí sola no puede realizar la tarea de controlar la corriente que pasa por la lámpara siendo necesario agregar más elementos al diseño con la finalidad de obtener el voltaje más estable y que no influyera en gran medida al momento de realizar el diseño del controlador.

El diagrama general de la etapa de potencia que se ha diseñado es el que se puede ver en la Figura 80.

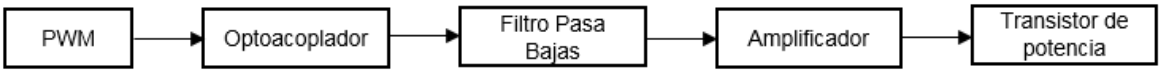

Figura 80. Diagrama general de la etapa de potencia.

El control del nivel de intensidad de la lámpara se realizará utilizando los módulos de PWM (*Pulse width modulation*). Como protección al microcontrolador es necesario colocar un optoacoplador con la finalidad de aislar el control con la potencia.

Una salida PWM es uno de los mecanismos más utilizados en los autómatas con la finalidad de emular una señal analógica. Consiste en proporcionar un tren de pulsos a una frecuencia determinada cuyo valor promedio es el valor analógico que se requiera. Sin embargo, la señal varía entre cero y el voltaje de salida como se ve en la Figura 81.

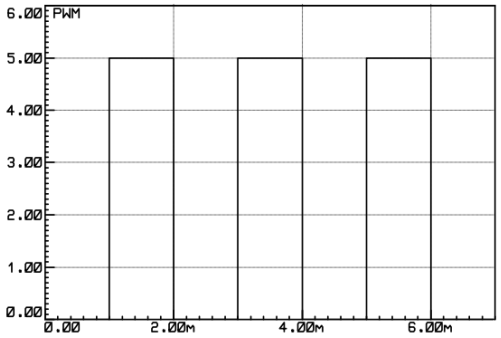

Figura 81. Ejemplo de PWM.

Si bien el PWM es muy utilizado para emular la señal analógica puede presentar diversos problemas. Entre ellos que si se quiere medir el voltaje que se está colocando en la carga se estará midiendo la tensión promedio del tren de pulsos haciendo la medición errónea.

Para mejorar la respuesta de la salida analógica se coloca un filtro pasa bajas a la salida del modulador mediante un circuito Resistencia – Capacitor (RC). La Figura 82 presenta el funcionamiento del circuito pasa bajas en respuesta al PWM. Se observa como a partir del tren de pulsos se genera un voltaje más estable y cercano a una señal analógica

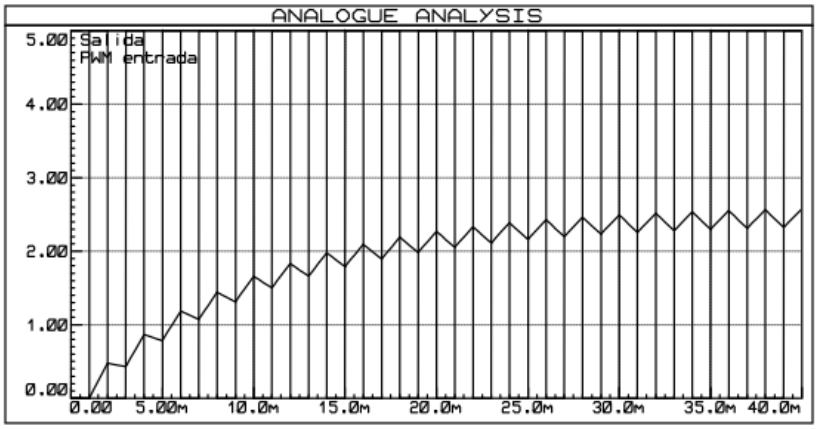

Figura 82. Funcionamiento del filtro pasa bajas.

El filtro diseñado es el que se muestra en la Figura 83 así como la respuesta que se tendrá. Los valores de la resistencia y el capacitor son valores cercanos recomendados en la documentación para la implementación de estos filtros a la frecuencia de trabajo del PWM del dispositivo que son aproximadamente 500Hz.

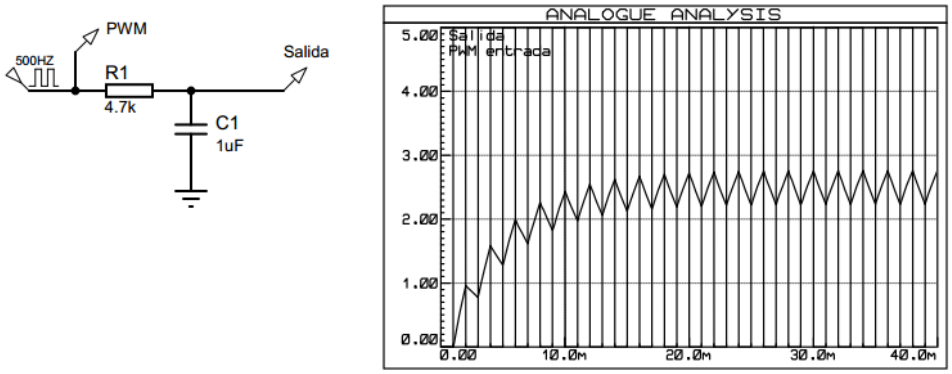

Figura 83. Circuito y respuesta del Filtro Pasa Bajas.

La frecuencia de corte del filtro está dada por la ecuación (27).

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi(4700)(1x10^{-6})} = 33.86Hz \qquad (27)$$

El comportamiento del filtro está regido por un sistema de primer orden por lo que la constante de tiempo y el tiempo de establecimiento están dado por las ecuaciones (28) y (29)

$$\tau = RC = 4700(1x10^{-6}) = 4.7mS \quad (28)$$

$$T_s = 3\tau = 4.7ms(3) = 14.1mS \quad (29)$$

Se ha diseñado el circuito de manera que tenga un tiempo de establecimiento alto. Al hacer esto se aumenta el rizado (picos de voltaje) generados que no afectan el funcionamiento de la carga a utilizar en este proyecto. El rizado va decreciendo al mismo tiempo que aumenta el ciclo de trabajo del PWM.

Una vez convertida la señal cuadrada a voltaje directo, es necesario amplificar el voltaje siendo esta tarea realizada por medio de un amplificador operacional. Habiendo amplificado el voltaje éste debe de manejarse por un elemento que pueda manejar el consumo de corriente, agregando un transistor de potencia que será el encargado de brindar la potencia necesaria en la lámpara.

El diseño electrónico de la etapa de potencia se observa en la Figura 84. Se ha agregado un optoacoplador con el fin de aislar eléctricamente la señal proveniente del microcontrolador con la señal de potencia y así tener una protección si se presenta un corto circuito, así como algunos elementos de visualización como lo es el led en la salida del PWM para tener un elemento visual de la señal cuadrada.

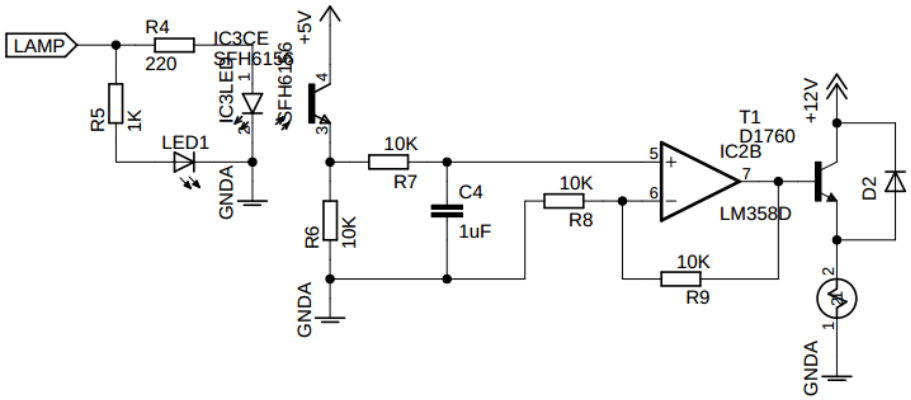

Figura 84. Diseño electrónico de la etapa de potencia.

Una vez teniendo el elemento calefactor se debe de tener un sistema de enfriamiento, para ello se realizó el circuito de potencia para el manejo de un ventilador tipo brushless utilizado en la refrigeración de equipos de cómputo. Como el ventilador está diseñado para trabajar mediante modulación de ancho de pulso, no es necesario colocar filtros u otros elementos

en el control más que el dispositivo que va a entregar la carga de potencia. El circuito se ve en la Figura 85.

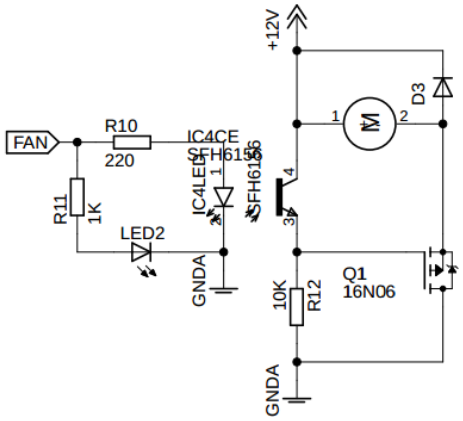

Figura 85. Circuito de potencia del ventilador.

Habiendo diseñado todos los circuitos para el control de las cargas, así como la adquisición de datos se procedió a hacer la parte de comunicación y los otros elementos que requiere el microcontrolador para trabajar y comunicarse con la Raspberry Pi®. El circuito del microcontrolador y de la Raspberry Pi® es el que se muestra en la Figura 86.

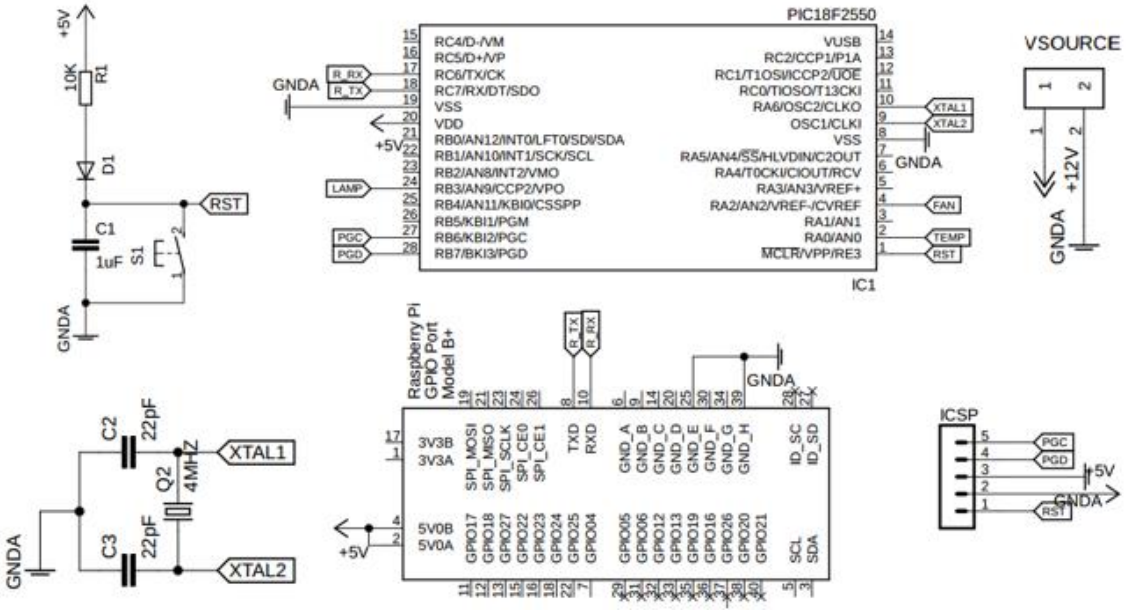

Figura 86. Circuito del microcontrolador y comunicación.

Con los circuitos diseñados se procedió a la realización del circuito impreso utilizando el Software AutoDesk® Eagle en su versión 9.2. Las limitaciones del impreso deberían de ser las que se enlistan a continuación:

- Debe de ser apilable en la Raspberry Pi® (Tipo Shield) con la finalidad de reducir espacio y que sea compacto y compatible con las tarjetas que tengan el puerto de entradas y salidas en la misma distribución.
- La mayoría de los componentes serán de montaje superficial para reducir espacio y tener una mejor presentación.

A partir de las especificaciones de diseño anterior, se crearon los circuitos impresos que pueden verse en la Figura 87 y corresponden a la parte inferior y superior. El circuito se realizó en doble cara para reducir espacio y aprovechando todo el espacio de la placa fenólica

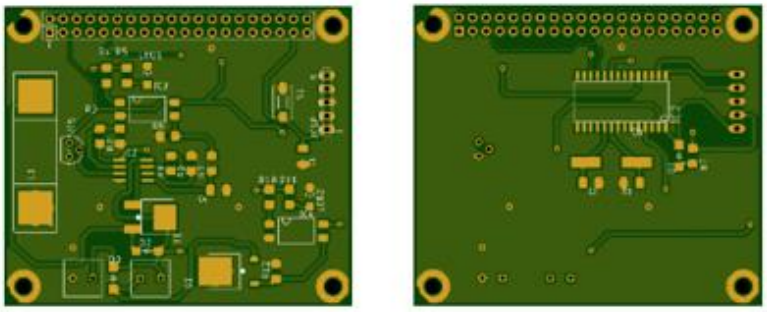

Figura 87. Caras superior e inferior del PCB.

Debido a la complejidad del circuito impreso por causa de su tamaño, la fabricación estuvo a cargo de una empresa especializada en la fabricación de PCB, obteniendo como resultado la placa de la Figura 88, una vez que fueron montados todos los componentes.

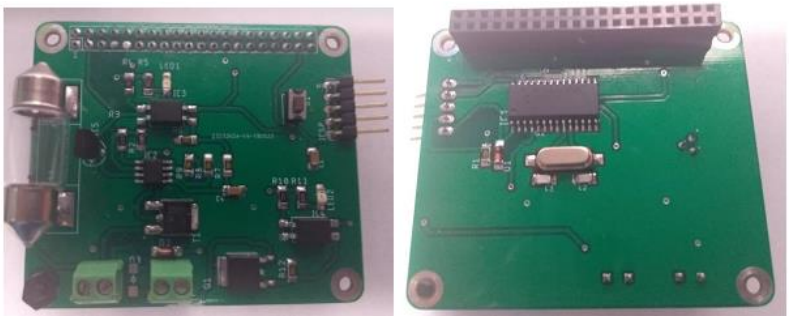

Figura 88. Montaje terminado del PCB.

3.4 DISEÑO DE SISTEMA ENTRENADOR DE PIC18F4550®

Durante la investigación del estado del arte, se encontraron ejemplos de programación de microcontroladores basándose en la cualidad de algunos de ellos de programar su propia memoria flash sin necesidad de utilizar el dispositivo específico de la marca para ello. Se encontró un programa de arranque (bootloader) que permite enviar el programa mediante el puerto serie llamado Tiny Bootloader.

El Tiny Bootloader es un arrancador que se almacena en la parte baja de la memoria flash del microcontrolador, y tiene un soporte en PIC de la serie 16, 18 y algunos Dspic®. Es de uso libre y fue diseñado por Claudiu Chiculita (Chiculita, 2018). Sin embargo, la versión original del bootloader se encuentra para la plataforma Windows que no puede instalarse en la Raspberry Pi®.

Siendo la comunidad de Python® una de las que más librerías crea, se encontró en la red una versión del mismo arrancador que funciona en Raspbian®, evitando el problema de compatibilidad entre las diversas plataformas.

3.4.1 DISEÑO ELECTRÓNICO.

Para realizar una comparación de sistemas con microcontrolador, se cambió de plataforma de hardware y software con el propósito de mejorar la experiencia de programación de los controladores, así como una interfaz visual más dinámica. Para ello se propuso un sistema entrenador de microcontroladores con el cual los alumnos puedan realizar las prácticas de su preferencia y programar el sistema de forma virtual.

El hardware incluido debe de cumplir con las siguientes características:

- Basarse en componentes de montaje superficial.
- Contener los elementos básicos para interactuar con la mayoría de las prestaciones con las que cuenta el microcontrolador (conversor ADC, PWM, DAC, I2C, SPI).
- Los elementos deben de ser de tamaño reducido y tener, en medida de lo posible, cambios que puedan observarse fácilmente mediante la cámara web.

A partir de lo anterior se procedió al diseño de los diferentes circuitos de prueba, como primer punto se colocaron diodos LED que puedan ser programados y entender el funcionamiento de las entradas digitales, así mismo se conectaron salidas de la tarjeta Raspberry Pi con entradas del microcontrolador PIC184550® para leerlas, con ello se

cumple el conocimiento básico de entradas y salidas. La Figura 89 muestra el circuito utilizado, se agregaron diodos de protección con la finalidad de que no fluya corriente en dos sentidos, pues si no se tiene un conocimiento avanzado puede hacerse un programa que dañe el sistema.

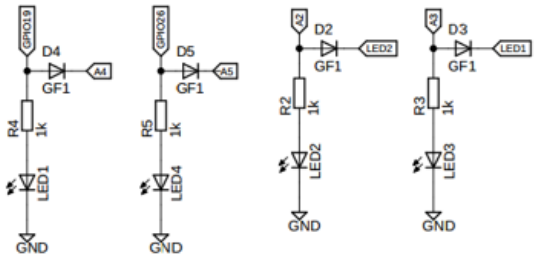

Figura 89. Circuito de entradas y salidas digitales.

Continuando con el uso de salidas digitales, se agregó un elemento de visualización que corresponde a un display de 7 segmentos de tipo burbuja con la finalidad de observar valores numéricos cuando sea necesario en combinación con los demás elementos de la plataforma.

El display cuenta con 4 caracteres que se multiplexaron con las salidas del microcontrolador obteniendo el circuito que se ve en la Figura 90.

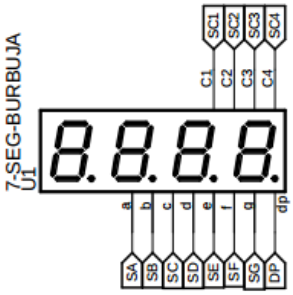

Figura 90. Circuito del display de siete segmentos.

En el apartado de comunicación se buscaron elementos de fácil adquisición y de uso común en la enseñanza de microcontroladores. Se optó por colocar un reloj de tiempo real DS1307 que tiene una comunicación I2C.

Existe numerosa documentación para su programación, así como ejemplos en la literatura pues es muy común encontrar relojes de este tipo en sistemas digitales. El integrado requiere de resistencias externas y un cristal de cuarzo encargado de darle los pulsos para la señal de salida, siendo el montaje el presentado en la Figura 91.

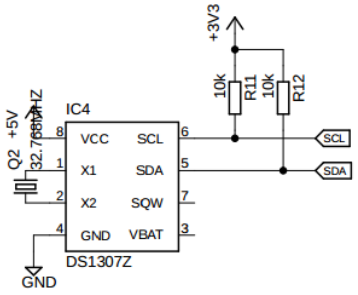

Figura 91. Circuito del RTC DS1307.

El segundo protocolo más utilizado es el SPI el cual se encuentra en numerosos elementos como los son los conversores análogo digital, digital análogo, pantallas LCD, sensores etc.

Con la finalidad de dotar al microcontrolador con un elemento con el que no cuenta por defecto y que tenga un cambio observable se colocó un conversor digital análogo TLV5616 con comunicación SPI. Su salida se encuentra conectada a un diodo emisor de luz para observar los cambios de voltaje mediante la intensidad luminosa de éste.

El integrado requiere la conexión de los pines de comunicación a los de uso específico del microcontrolador y una referencia de voltaje equivalente a la mitad del voltaje de entrada el esquemático utilizado se encuentra en la Figura 92.

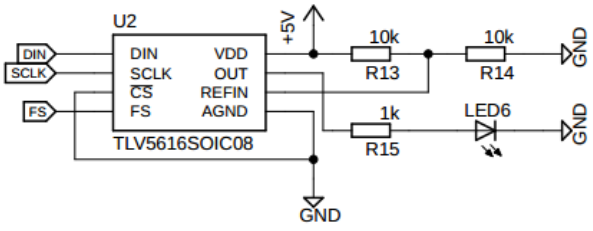

Figura 92. Circuito del DAC TLV5616.

Aprovechando el uso del LED conectado al conversor, como elemento análogo de entrada se utilizó una resistencia dependiente de luz (LDR) el cual tendrá un valor de salida

dependiente de la luminosidad del LED utilizado anteriormente, lo que permitirá a los alumnos tener un pequeño control de los elementos que se encuentra en la placa para aprovecharlos en manera conjunta. La resistencia dependiente de luz se encuentra en un divisor de tensión para obtener un voltaje de salida y posteriormente a un amplificador para mejorar la señal y que pueda ser leída por el microcontrolador como se ve en la Figura 93.

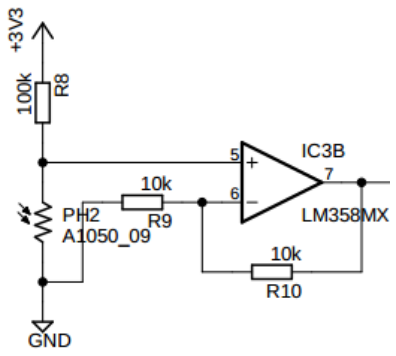

Figura 93. Circuito del LDR.

Como último elemento se colocó un sensor de temperatura LM35 que, de acuerdo a la investigación realizada, se encuentra entre los más usados en el aprendizaje del uso del conversor análogo digital del microcontrolador. Si bien puede ser leído directamente se optó por amplificar la señal el doble para utilizar más el rango del conversor de 8 bits del microcontrolador. El circuito utilizado se muestra en la Figura 94.

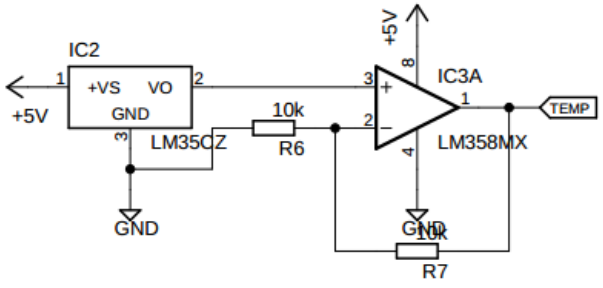

Figura 94. Circuito del sensor LM35.

Si bien el microcontrolador deberá interactuar con los elementos mostrados anteriormente, también debe hacerlo con la Raspberry Pi®. El bootloader (programa guardado en la memoria del microcontrolador que permite reescribirlo internamente) funciona mediante puerto serial, siendo necesaria la comunicación por este tipo de comunicación con la placa

Raspberry Pi®. Las conexiones del puerto serie, así como los puertos digitales utilizadas son las que se ven en la Figura 95.

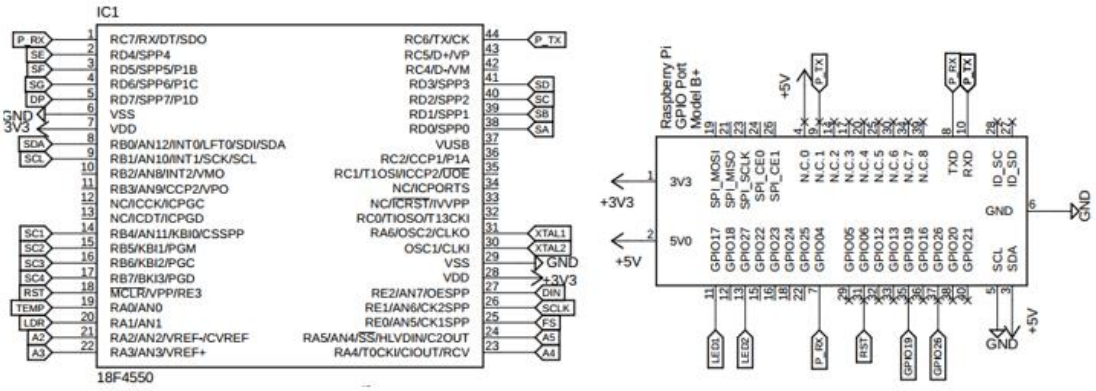

Figura 95. Conexión entre el PIC18F4550 y la Raspberry Pi®.

Se puede observar en la imagen que el microcontrolador se encuentra alimentado a 3.3V, pues es el voltaje de trabajo de la microcomputadora. Utilizar un voltaje más alto (como los 5V comúnmente usados en los microcontroladores de este tipo) corresponde un riesgo de daño al sobrecargar los puertos.

Una vez realizado el diseño de la electrónica, se procedió a la creación de la placa del circuito impreso utilizando el software AutoDesk® Eagle 9.2 en montaje superficial. Obteniendo los resultados que se pueden ver en la Figura 96.

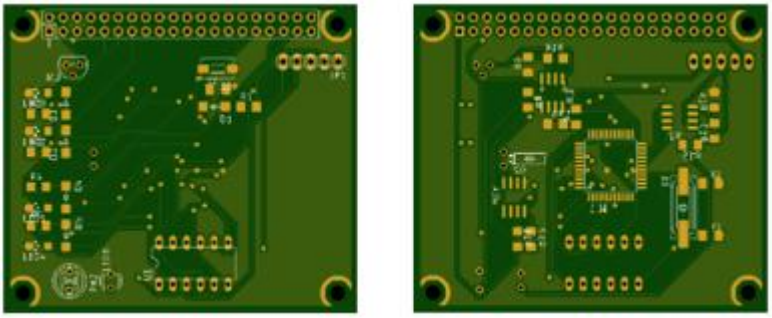

Figura 96. Diseño del PCB del entrenador.

Debido a la complejidad del circuito impreso por causa de su tamaño, la fabricación estuvo a cargo de una empresa especializada en la fabricación de PCB, obteniendo como resultado la placa de la Figura 97, una vez que fueron montados todos los componentes.

Figura 97. Circuito montado del entrenador.

La figura de la parte izquierda corresponde a la vista superior en donde se colocaron todos los elementos visuales que van a interactuar con el usuario de manera que tenga una mejor presentación. Mientras que en la parte inferior se instalaron los circuitos integrados y elementos que no son relevantes visualmente en el uso de la plataforma.

3.4.2 INTERFAZ VISUAL PARA CARGA DE PROGRAMA EN NODE-RED®

Si bien se encontró el programa para cargar el archivo hexadecimal desde Python, se debe recordar que éste no es el lenguaje con el que se está trabajando la interfaz visual, por ello, se debió encontrar la manera de ejecutar los scripts desde las páginas del Node-Red®.

En primer lugar, se creó una página web sencilla a la cual se accediera desde una dirección distinta al *dashboard* donde se encuentren los controles de la interfaz de usuario. Se utilizó el paquete de Node-Red® que permite ejecutar peticiones GET en donde se encontrarán los botones que subirán el archivo hexadecimal y lo almacenarán en la memoria de la Raspberry Pi®, para posteriormente ser leída desde el script de Python®. Los nodos que realizan la tarea de crear la página web se pueden ver en la Figura 98.

Figura 98. Nodos para creación de peticiones GET en Node-Red®.

Mediante el nodo Template se puede realizar un dibujo mediante código HTML que es el utilizado comúnmente en la programación Web. El primer nodo del algoritmo crea la petición Get utilizando la ruta HTTP que se le coloque dentro de su configuración.

El código HTML utilizado para realizar la página web de carga de programa se muestra en la Figura 99. En el código solo se colocan dos botones, uno para subir el archivo hacia la memoria SD y otro para activar la carga del hexadecimal.

```
<h1>Subir Programa (HEX):</h1>

<form action="/uploadsimple_post" method="POST" enctype="multipart/form-data">
    <input type="file" name="myFile" />
    <input type="submit" value="Submit">
</form>
```

Figura 99. Código HTML para carga de programa.

Con la página creada, se procede a la conexión de nodos que permitan ejecutar scripts o acceder a los comandos propios de Raspbian®. El algoritmo comienza leyendo la petición POST proveniente de la página web creada anteriormente como se ve en la Figura 100.

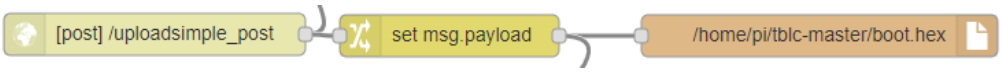

Figura 100. Nodo de lectura de petición POST en Node-Red®.

El primer nodo permite leer las peticiones POST que se han hecho, el segundo nodo realiza una lectura de los datos que se han enviado una vez que se ha utilizado el método. Por último, todo lo recabado se guarda en la ruta especificada en el último nodo y corresponde al archivo que será leído desde la terminal.

Los nodos que se ven en la Figura 101 realizan la carga del programa hacia el microcontrolador. Se manda el pin de RESET del micro y se ejecuta el script de Python®, una vez que el programa ha sido cargado se devuelve el microcontrolador a su funcionamiento y se realiza una revisión de que la carga haya sido exitosa, dando al usuario seguridad de que la tarea se ejecutó.

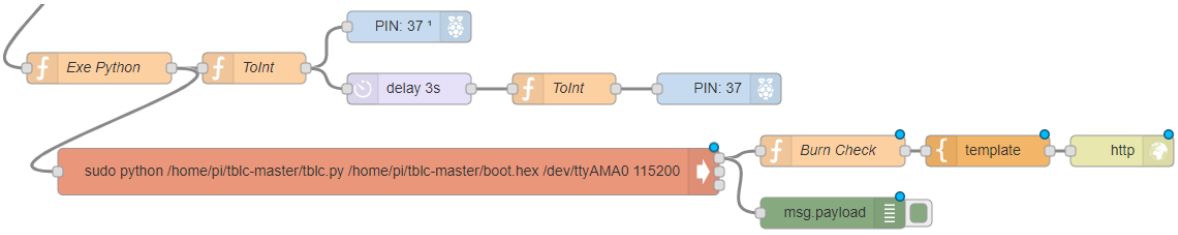

Figura 101. Nodos para ejecución del bootloader mediante terminal en Node-Red.

3.5 IDENTIFICACIÓN DEL MODELO MATEMÁTICO EN PYTHON®

Con la finalidad de tener una herramienta que permitiera hacer pruebas teóricas acerca del comportamiento matemático de la planta, así como su simulación al momento de aplicar las normas de control correspondientes, se optó por implementar algoritmos de identificación de modelo matemáticos descritos anteriormente como lo son los métodos de Alfaro y de Ziegler – Nichols.

Teniendo en cuenta que se quiere trabajar con herramientas de uso libre que los usuarios comunes que no cuentan con software de pago como lo puede ser Matlab®, o semejante, se programó utilizando las librerías de ciencia y matemática para Python®, que contienen funciones y métodos muy parecidos a los softwares antes mencionados.

Se buscó que el programa fuera aplicable sin necesidad de utilizar la herramienta visual en Node-Red®, es decir si la adquisición de datos se realizó de una manera diferente, por lo que su funcionamiento se basa en la lectura de datos provenientes de un archivo .xls de Excel®. La metodología para la implementación de este script se muestra en Figura 102

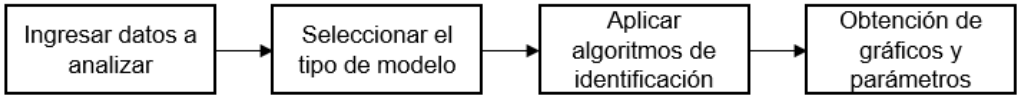

Figura 102. Metodología para script de identificación.

Python® no cuenta con las herramientas de lectura de archivos tipo xls ni con las librerías para el manejo de matemática, ciencia y control por defecto, siendo necesario su instalación mediante los diferentes paquetes que se encuentran libres por la comunidad de este lenguaje. En este caso se utilizó la plataforma Anaconda 2.7®, que trae por defecto algunas de éstas herramientas. Sin embargo, debe de completarse la instalación de las faltantes. Las librerías utilizadas en la implementación de este script se muestran en la Figura 103.

```
import matplotlib.pyplot as plt
import xlwt
import xlrd
import scipy
import numpy as np
from scipy import interpolate
from scipy import signal
```

Figura 103. Librerías para uso de herramientas matemáticas en Python®.

El programa permite encontrar el modelo de primer orden más tiempo muerto, así como el de segundo orden más tiempo muerto. El primero ya que es el más sencillo de modelar en donde se pueden calcular los parámetros del PID de diferentes maneras. El segundo para poder utilizar los métodos de Ziegler – Nichols y encontrar las ganancias del controlador.

El diagrama de flujo de la Figura 104 resume el algoritmo que se utiliza para encontrar los valores de tiempo que se necesitan para el modelo de primer orden más tiempo muerto utilizando el método de Alfaro.

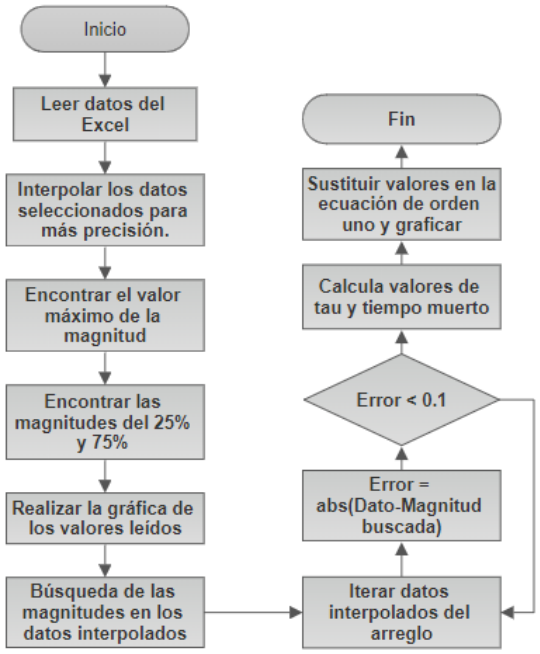

Figura 104. Algoritmo para programar método de Alfaro.

Los datos que provienen del Excel® no son infinitos, se encuentran en tiempo discreto y, en la mayoría de los casos, si no se hace una adquisición con muchas muestras, no es suficiente para poder aplicar algoritmos de búsqueda, por ello se hace una interpolación de los datos para aproximar una curva con más resolución y que permita tener datos más aproximados a una curva analógica con más puntos en el tiempo.

La interpolación ayuda a que los datos obtenidos al momento de encontrar los valores de las magnitudes correspondientes sean más cercanos a los que se tuvieran si se tuvieran un número infinito de puntos sobre la curva. El error puede ajustarse de acuerdo a las necesidades que se requieran, es decir, si se requiere un modelo preciso o no.

Para encontrar los parámetros de Ziegler – Nichols se sigue un procedimiento parecido, cambiando la cantidad de parámetros que deben de ser obtenidos. El diagrama de flujo para el algoritmo de Ziegler – Nichols se ve en la Figura 105.

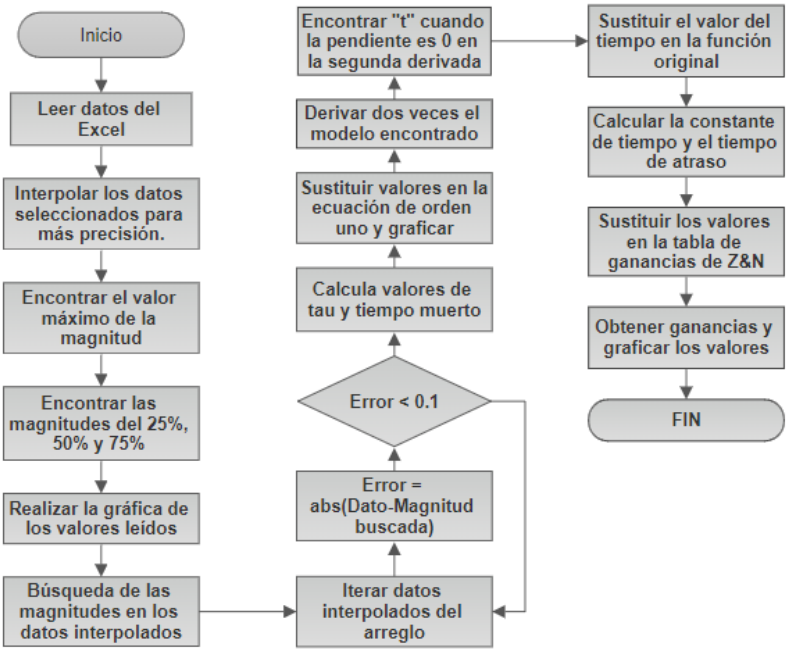

Figura 105. Algoritmo para programar Ziegler- Nichols.

El algoritmo es una combinación del Método de Alfaro con las reglas anteriormente descritas. Se debe de recordar que el método Z&N es un método comúnmente realizado gráficamente por lo que se debió encontrar una metodología matemática para la obtención de los parámetros. Un ejemplo de gráfico obtenido se ve en la Figura 106.

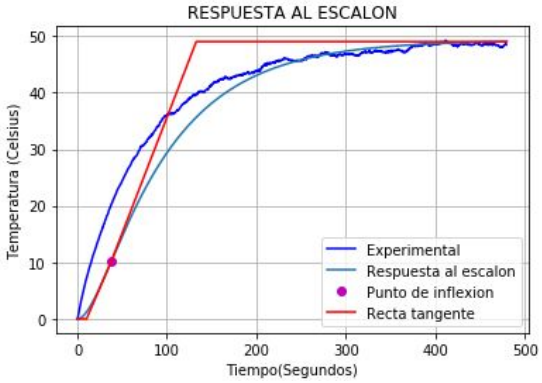

Figura 106. Ejemplo de gráfico del algoritmo de modelado.

CAPÍTULO 4: RESULTADOS

4.1 DISEÑO DEL CONTROLADOR PID

Para el ejemplo del diseño del controlador se optó por el de temperatura, ya que es el sistema más fácil de modelar y el que tiene elementos lineales que facilitan que el algoritmo implementado en Python funcione de manera correcta.

4.1.1 METODOLOGÍA

Siguiendo el diagrama a bloques de los sistemas de control digital es necesario realizar una metodología para obtener todos los parámetros necesarios para la sintonización del controlador. El diagrama a bloques de la Figura 107 presenta la metodología utilizada para el diseño del controlador.

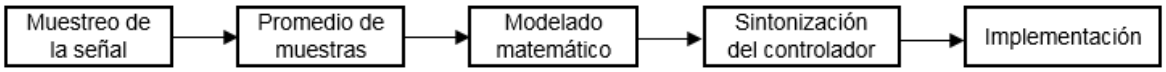

Figura 107. Metodología para el diseño del controlador PID.

Antes de sintonizar el controlador es necesario conocer el modelo de la planta (si es que se utilizan algoritmos de control clásico). Para ello es necesario realizar un muestreo de la respuesta del sistema y obtener su curva de reacción a un paso escalón. Debido a que el sistema térmico varía de acuerdo a las condiciones ambientales es necesario realizar pruebas a diferentes temperaturas ambientes con la finalidad de promediarlas y obtener una curva aproximada que pueda adaptarse a las condiciones que se tengan presentes al momento de la implementación.

Una vez obtenidas las pruebas se obtiene el modelo matemático utilizando los métodos propuestos en diferentes proyectos y fuentes de información. A partir del modelo matemático es posible sintonizar y obtener las constantes necesarias para la implementación del control PID.

4.1.2 OBTENCIÓN DEL MODELO MATEMÁTICO

Habiendo realizado el sistema de comunicación, se procedió a realizar los experimentos para la obtención del modelo matemático. En primer lugar, es necesario conocer cuál es la

relación entre el ciclo de trabajo de la modulación por ancho de pulsos y la temperatura que se genera con éste.

La resolución del PWM del PIC18F2550® es de 8 bits por lo que el valor del ciclo de trabajo se encuentra entre 0 y 255. Con estos valores se realizaron pruebas con incrementos de 500 en la cuenta del ciclo de trabajo midiendo tanto la temperatura como el voltaje en la lámpara incandescente. Las pruebas se repitieron 5 veces y se obtuvo el promedio de valores de cada una de ellas. En la Tabla 3 pueden observarse los resultados promediados.

Tabla 2. Valores de la relación ciclo de trabajo – temperatura.

Ciclo de trabajo	Voltaje generado	Temperatura
20	1.86V	49°C
60	3.1V	58°C
100	5.65V	65°C
140	7.11V	71°C
180	8.2V	75°C
220	8.83V	78°C

Ya que la temperatura ambiente es cambiante, se definió que el valor mínimo que puede tener el ciclo de trabajo sea de 15 con la finalidad de que la temperatura inicial del sistema sea lo más constante posible. La gráfica generada por los valores anteriores se puede observar en la Figura 108.

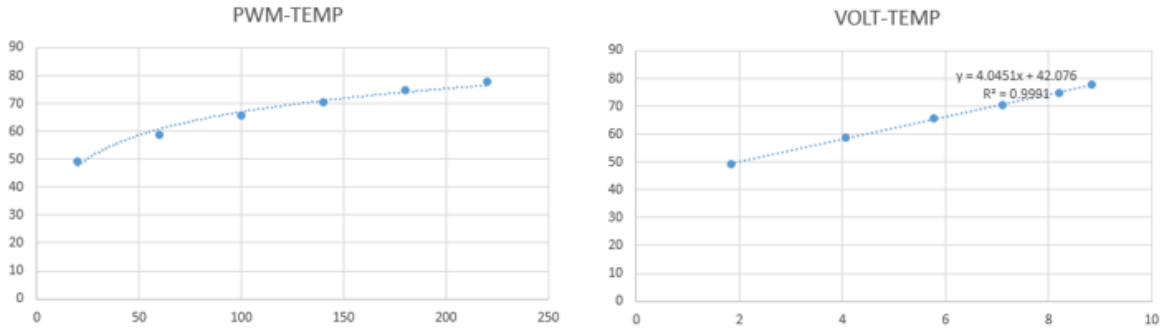

Figura 108. Relaciones de variables.

Puede observarse que el cambio de voltaje es proporcional al cambio de temperatura, sin embargo, esto no es así con el ciclo de trabajo.

Para realizar las pruebas de la respuesta al escalón se colocó un valor inicial de 60 en el ciclo del trabajo del PWM y posteriormente se le aplicó un incremento a 220 con la finalidad de obtener la curva de respuesta. El número de muestras fue de 960 con un tiempo de muestreo de 500mS. La primera prueba es la que se observa en la Figura 109

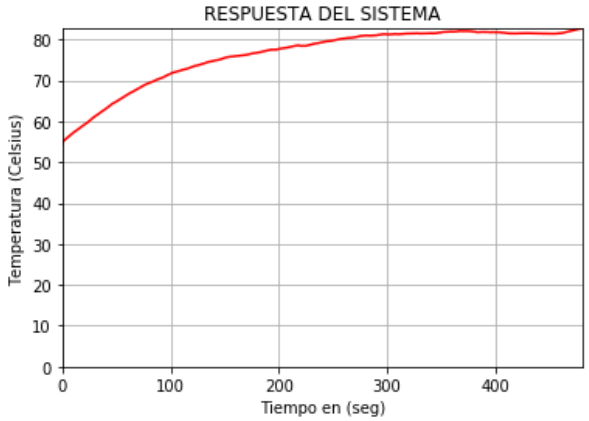

Figura 109. Respuesta experimental al paso escalón del sistema.

Se puede observar que las mediciones varían en ciertas partes de la gráfica debido a las perturbaciones existentes en el ambiente en el que se estaban realizando las pruebas. Por ello se hicieron 6 muestreos y fueron promediados así mismo se le realizó un Offset a la gráfica para que el valor inicial fuera cero y no tener problemas al momento de aplicar los algoritmos de modelado. La gráfica generada por el promedio de pruebas una vez aplicado el Offset es la que se muestra en la Figura 110.

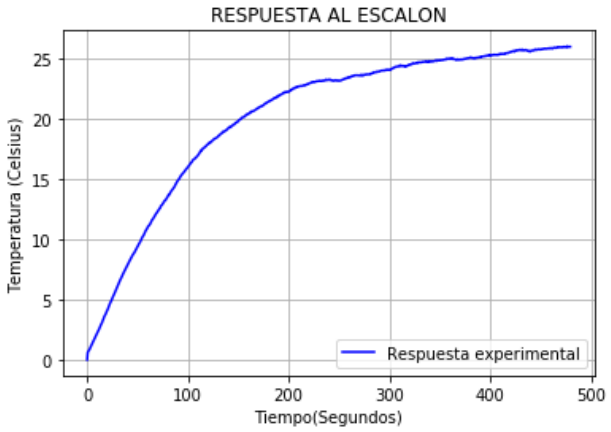

Figura 110. Respuesta al paso escalón promediada.

Si bien la gráfica generada parece continua esto es por términos de visualización ya que se debe recordar que es un sistema muestreado y solo existen los puntos que se han tomado. Por ello se realizó una interpolación con la cual se pudiese crear un espacio con un gran número de valores estimados entre cada una de las muestras. Se puede observar en la Figura 111 como la interpolación está demasiado cercana a la curva obtenida.

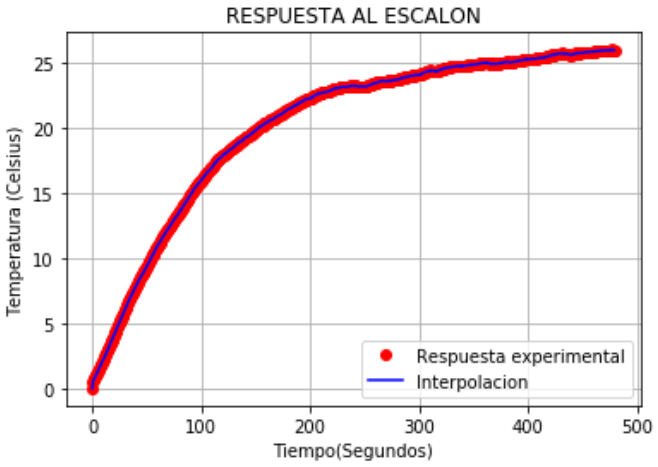

Figura 111. Interpolación de las muestras.

De manera gráfica puede concluirse que el sistema es de primer orden. A pesar de ello, y con la finalidad de probar los algoritmos de modelado de sistemas por medio de la curva de respuesta, se realizó la identificación de la planta por el método de dos puntos propuesto por Alfaro.

El primer paso fue obtener el valor máximo del sistema y a partir de él encontrar el tiempo en el que la magnitud de temperatura se encuentra a un 25% y a un 75%, en el caso de un sistema de primer orden más tiempo muerto, y de un 50% en el caso de un sistema de segundo orden más tiempo muerto.

Para encontrar los tiempos en los que se alcanzan los porcentajes se utilizó un algoritmo de búsqueda en los datos de la interpolación, comparando la magnitud a buscar con todos los valores en la lista de datos. Es importante tomar en cuenta que este método puede o no ser exacto pues es muy poco probable que se obtenga la medición exacta siendo necesario colocar un error al momento de comparar.

En la gráfica de la Figura 112 pueden observarse los tiempos obtenidos así como su ubicación en la curva de respuesta.

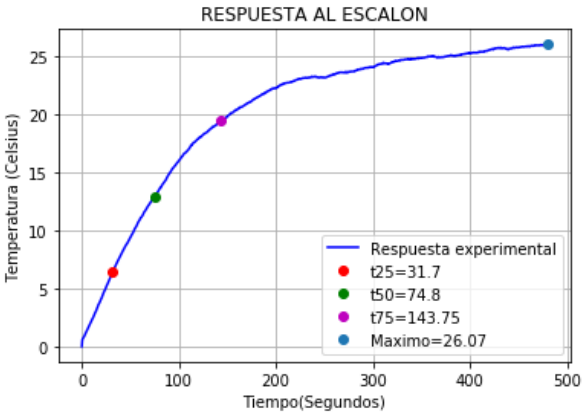

Figura 112. Obtención de los tiempos para el método de Alfaro.

Utilizando las ecuaciones propuestas para un modelo de primer orden más tiempo muerto se obtuvieron los siguientes parámetros.

$$k_p = \frac{\Delta y}{\Delta u} = \frac{26.07}{26.07} = 1$$

$$\tau = 0.9102(t75 - t25) = 0.9102(143.75 - 31.7) = 101.98$$

$$t_m = 1.2620t_{25} - 0.2620t_{75} = 1.262(31.7) - 0.262(143.75) = 2.34$$

Por lo que el modelo de primer orden más tiempo muerto utilizando el método de Alfaro es el que se presenta en la ecuación (30)

$$G(s) = \frac{e^{-2.34s}}{101.98s + 1} \tag{30}$$

De acuerdo a la teoría de los sistemas de primer orden puro, la constante de tiempo es aquella en la que la respuesta alcanza el 63.21% por lo que el modelo de primer orden puro está dado por la ecuación (31)

$$G(s) = \frac{1}{103s + 1} \tag{31}$$

Se puede observar que ambos modelos de primer orden son muy parecidos y por lo tanto válidos, siendo útiles para utilizar los métodos de sintonización correspondientes para cada uno.

Los parámetros para un modelo de segundo orden más tiempo muerto están dados por las siguientes ecuaciones.

$$\tau' = 0.5776(143.75 - 31.7) = 64.72$$

$$t_m'' = 1.5552(31{,}7) - 0.5552(143.75) = -30.51$$

$$a = \frac{t_{50} - t_m' - 1.4362\tau'}{1.9844\tau' - t_{50} + t_m'} = \frac{74.8 - 2.34 - 1.4362(64.72)}{1.9844(64.72) - 74.8 + 2.34} = 0.534$$

$$\tau_1 = \tau'' = 84.35$$

$$\tau_2 = a\tau'' = (0.534)(84.35) = 45.09$$

$$\tau'' = \frac{2\tau'}{1 + a} = 84.35$$

Siendo el modelo matemático de un sistema de orden dos más tiempo muerto el que se presenta en la ecuación (32). Debido a que con este modelo existirían tiempos negativos debido al tiempo muerto, éste no se tomará en cuenta.

$$G(s) = \frac{e^{30.51s}}{(84.35s + 1)(45.09s + 1)} \tag{32}$$

La Figura 113 presenta la comparación de los dos modelos con la curva experimental.

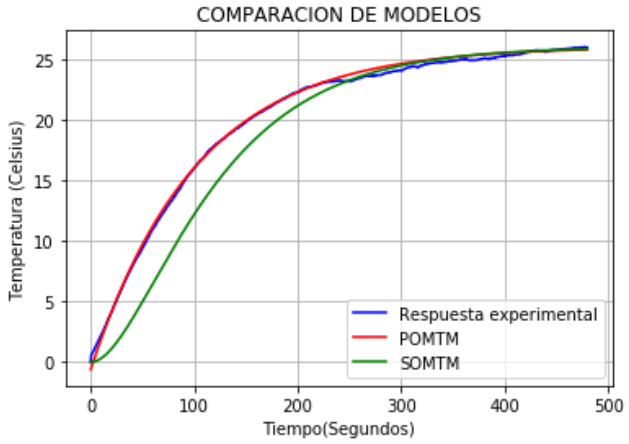

Figura 113. Comparación de modelos matemáticos.

Donde la señal roja corresponde al modelo de primer orden más tiempo muerto (POMTM), siendo la curva color verde al modelo de segundo orden (SOMTM) ignorando el tiempo muerto. Se puede observar que aun ignorando el retraso el modelo de orden dos es muy

cercano a la respuesta real de la planta. Sin embargo, el modelo de primer orden se ajusta con mucha exactitud a los datos experimentales.

4.1.3 SINTONIZACIÓN DEL CONTROLADOR PID

Habiendo obtenido el modelo matemático se procedió a la sintonización del controlador PID, es decir, la obtención de las constantes que permitirán cambiar la respuesta del sistema de acuerdo a los parámetros que el desarrollador desee.

4.1.4 SINTONIZACIÓN POR ZIEGLER-NICHOLS

El método de Ziegler-Nichols es uno de los más antiguos y también de los más utilizados en la sintonización de controladores PID pues permiten dar valores cercanos de las constantes que posteriormente pueden ser ajustados de manera experimental por el diseñador para obtener la respuesta necesaria para el control del proceso.

Para la sintonización por este método se utilizó el modelo de segundo orden ignorando el tiempo muerto. Los pasos seguidos para encontrar el punto de inflexión se describen a continuación.

1. Encontrar la primera derivada de la respuesta del sistema.
2. Derivar una segunda vez y encontrar el valor del tiempo cuando la derivada sea cero.
3. El punto de inflexión se encontrará al sustituir el tiempo en la respuesta del sistema.
4. Evaluar el valor del tiempo en la primera derivada para obtener la pendiente de la recta.
5. Con el valor de la pendiente es posible calcular dos puntos de la recta para obtener su ecuación.

En la Figura 114 se observan las gráficas generadas utilizando la librería *Numpy*, que permite realizar operaciones con vectores, facilitando el cálculo de las derivadas. Mediante algoritmos de búsqueda se encontraron los valores de tiempo establecidos.

Todas las gráficas fueron generadas usando algoritmos de programación en Python® por lo que pueden utilizarse para cualquier ejemplo en general y no solo para el propuesto en este trabajo. Sin embargo, se tiene que tener en cuenta que los datos ingresados corresponden a datos reales por lo que se pueden tener problemas al momento de querer generar gráficas si es que éstos no están bien definidos.

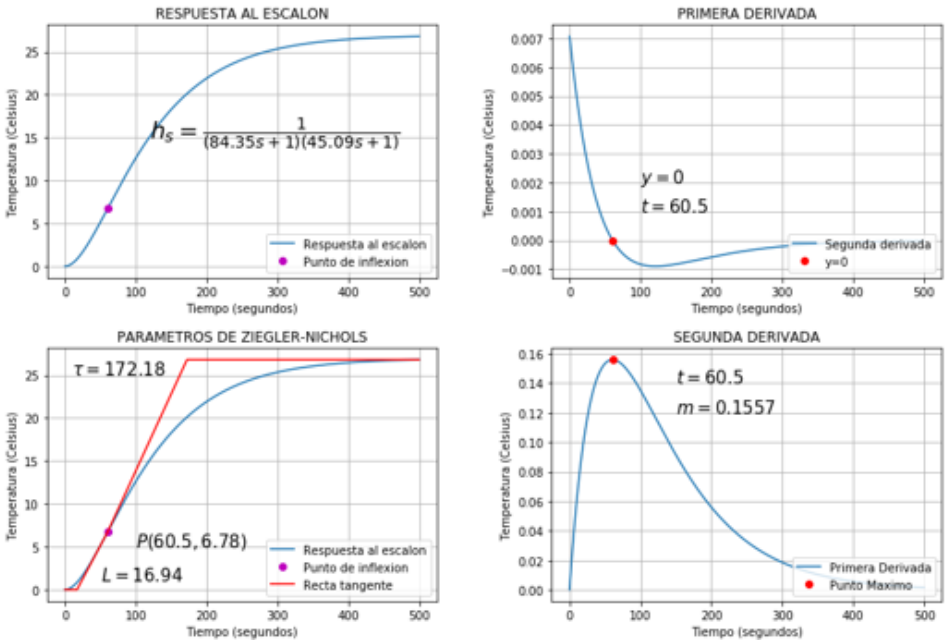

Figura 114. Obtención del punto de inflexión y parámetros de Ziegler – Nichols.

En la gráfica superior derecha se observa la gráfica correspondiente a la primera derivada, en la cual es necesario encontrar el valor de tiempo donde la variable dependiente es igual a cero siendo t=60.5 el valor más cercano a cero encontrado. Posteriormente en la gráfica inferior derecha se muestra la segunda derivada evaluando el valor del tiempo en la función, obteniendo el punto máximo de ésta y con ello el valor de la pendiente “m”.

Una vez que se ha encontrado el valor del tiempo necesario, se evalúa en la sigmoidea de la respuesta al escalón obteniendo el punto de inflexión (P) que se muestra en la gráfica superior izquierda.

Sabiendo que el tiempo de atraso es aquel en el que la variable dependiente es igual a cero, y habiendo obtenido el punto de inflexión, se puede conocer su valor mediante la ecuación de la recta al solo tener una incógnita. El tiempo de atraso (L) está dado por la ecuación (33).

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{6.78 - 0}{60.5 - L} \tag{33}$$

$$0.1557 = \frac{6.78}{60.5 - L}$$

$$L = \frac{9.4198 - 6.78}{0.1557} = 16.94$$

El valor de la constante de tiempo está dado por la ecuación (34):

$$T = \frac{h(s)_{max}}{m} = \frac{26.81}{0.1557} = 172.18 \tag{34}$$

Por lo que el modelo de primer orden más tiempo según Ziegler-Nichols es el que se muestra en la ecuación (35).

$$h(s) = \frac{e^{-16.94t}}{172.18s + 1} \tag{35}$$

Con ayuda de las ecuaciones de la Tabla 1 se calculan los valores de las constantes para el controlador tipo PID. Los resultados se pueden ver a continuación en las ecuaciones (36)–(38).

$$k_p = 1.2\ \frac{T}{L} = 1.2\frac{172.18}{16.94} = 12.19 \tag{36}$$

$$T_i = 2L = 2(16.94) = 33.8 \tag{37}$$

$$T_d = \frac{L}{2} = \frac{16.94}{2} = 8.47 \tag{38}$$

Y expresado en términos de ganancias por las ecuaciones (39),(40).

$$k_i = \frac{k_p}{T_i} = \frac{12.19}{33.8} = 0.38 \tag{39}$$

$$k_d = k_p\, T_d = 12.19(8.47) = 103.24 \tag{40}$$

4.1.5 SINTONIZACIÓN POR ADAPTACIÓN DE ZIEGLER-NICHOLS

La adaptación hecha a las reglas de Ziegler-Nichols está basada en un modelo de primer orden más tiempo muerto obtenido por el método de Alfaro. Si bien ambos modelos tienen similitud distan en la manera en que son obtenidos, pues Ziegler – Nichols está basado en la recta tangente al punto de inflexión y Alfaro en puntos sobre la curva de respuesta.

Utilizando el modelo de POMTM anteriormente obtenido y siguiendo las ecuaciones (41)-(43) se calculan los parámetros para el controlador.

$$K_p = \frac{1.6632\tau + 0.2229}{K(0.7345t_m - 0.0020)} = \frac{1.6632(101.98) + 0.2229}{(0.7345 * 2.34 - 0.0020)} = 98.92 \quad (41)$$

$$T_i = 1.4690t_m - 0.0040 = 1.4690(2.34) - 0.004 = 3.43 \quad (42)$$

$$T_d = 0.3672t_m - 0.0010 = 0.3672(2.34) = 0.8592 \quad (43)$$

4.2 DISEÑO DEL CONTROLADOR PID DISCRETO

Debido a que el controlador se va a implementar en un microcontrolador es necesario convertir del tiempo continuo al tiempo discreto. Para es necesario pasar al dominio de Z las funciones de transferencia en el dominio S. En primer lugar, es necesario establecer un tiempo de muestreo. Ziegler Nichols proponen su cálculo mediante la ecuación (44).

$$T < \frac{t_m}{4} \quad (44)$$

De acuerdo a lo anterior se ha definido un tiempo de muestreo de 500mS ya que este ha sido el tiempo aplicado al momento de realizar la toma de muestras de la señal.

Utilizando la aproximación de Tustin y las ecuaciones presentadas por García (García E. , 2008) el PID discreto está dado por la ecuación (45).

$$C(z^{-1}) = \frac{u(k)}{e(k)} = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 - z^{-1}} \quad (45)$$

Donde:

$$q_0 = k_p\left[1 + \frac{T}{2t_i} + \frac{t_d}{T}\right]$$

$$q_1 = -k_p\left[1 - \frac{T}{2t_i} + \frac{2t_d}{T}\right]$$

$$q_2 = \frac{k_p t_d}{T}$$

En el diagrama de la Figura 115 se muestra el gráfico de bloques del control digital. La ecuación presentada corresponde a la función de transferencia del controlador, siendo necesario despejar la señal de control u(k) que es la que ingresará al proceso.

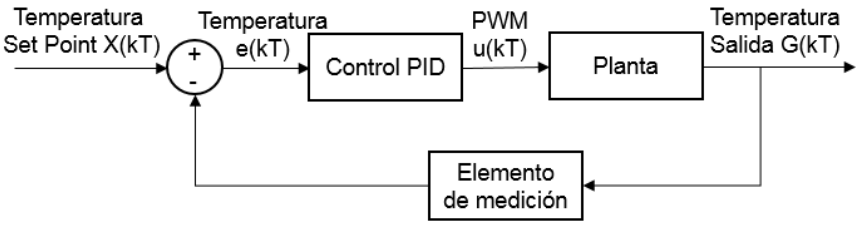

Figura 115. Diagrama bloques control PID.

Sin embargo, es necesario recordar que el equivalente al tiempo continuo en tiempo discreto son las ecuaciones en diferencias por lo que se calcula la transformada Z inversa

$$u(k)(1 - z^{-1}) = q_0 e(k) + q_1 z^{-1} e(k) + q_2 z^{-2} e(k)$$
$$u(k) - u(k)z^{-1} = q_0 e(k) + q_1 z^{-1} e(k) + q_2 z^{-2} e(k)$$
$$u(k) = u(k)z^{-1} + q_0 e(k) + q_1 z^{-1} e(k) + q_2 z^{-2} e(k)$$

La transformada Z inversa es la que se muestra en la ecuación (46):

$$u(kT) = u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 (k-2) \quad (46)$$

Así, u(kT) quiere decir la ley de control en el instante de tiempo actual, u(k-1) corresponde a la ley de control un muestreo atrás, e(k) es el error en el estado actual y e(k-1) corresponde al error un tiempo atrás y e (k-2) el error retrasado dos tiempos.

4.3 IMPLEMENTACIÓN DEL CONTROLADOR DIGITAL

Con la ecuación en diferencias obtenida se procedió a la programación del controlador en el microcontrolador PIC18F2550®. Sin embargo, la ecuación es válida para cualquier microcontrolador que tenga la capacidad de realizar las operaciones de manera eficaz y asegurando el tiempo de muestreo establecido.

Como se observa en el diagrama a bloques de la Figura 115, el punto de referencia se da en unidades de Temperatura y así mismo la señal de error, por otro lado una vez que entra en juego el controlador, las unidades deben de ser convertidas a resolución de PWM pues esta es la señal que hará que el proceso cambie su estado.

Para ello se utilizan los valores de la Tabla 2. Valores de la relación ciclo de trabajo – temperatura con el fin de hacer un ajuste de curvas que dé una ecuación que relacione la temperatura con el PWM. Usando el comando polyfit de Python® de una ecuación cuadrática se obtuvieron los resultados que se ven en la Figura 116.

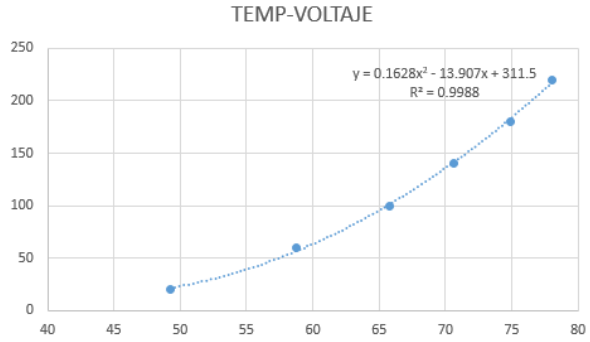

Figura 116. Ajuste de curvas relación Temperatura – PWM.

Habiendo calculado la ecuación que relaciona la temperatura y el PWM ya no es necesario volver a transformarla puesto que las lecturas que da el elemento de medición serán en grados centígrados. Si bien se puede observar un ajuste muy cercano a la curva real, puede que se presenten errores por el grado del polinomio de grado dos. De acuerdo a los cálculos el factor r^2=0.998. El diagrama de flujo de la Figura 117 resume el algoritmo del control PID que se implementó en el PIC18F2550®.

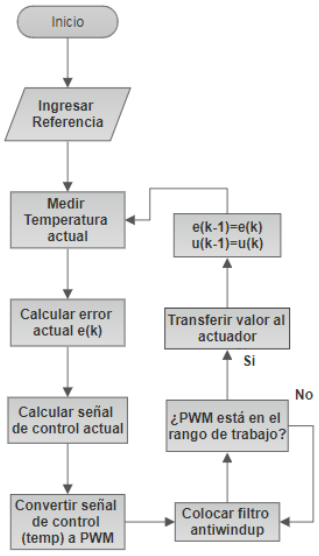

Figura 117. Diagrama de flujo de programación del PID.

La referencia debe de estar ingresada en grados centígrados. En la primera iteración los errores anteriores deben de estar igualados a cero. Una vez hecha la medición es necesario calcular el error actual que será ingresado a la ecuación en diferencias calculada que devolverá la señal de control en grados centígrados. Posteriormente se deben de convertir a resolución de ciclo de trabajo y colocar un filtro *anti windup*, que consiste en colocar límites tanto inferiores como superiores para la variable de control.

Este filtro es de vital importancia para el buen funcionamiento del controlador, pues si en el cálculo de la variable de control ésta supera la resolución admitida por el microcontrolador las cuentas empezarán a fallar y el control se saldrá de control.

Cuando el PWM se ha limitado se transfiere al actuador y por último se guardan los valores anteriores para la siguiente iteración.

4.4 PROGRAMACIÓN DE ALGORITMOS DE IDENTIFICACIÓN EN PYTHON®

El código hecho en Python® permite, a través de los archivos de Excel® de muestreo, identificar y sintonizar los valores del control de acuerdo a los métodos propuestos. En la Figura 118 se muestra el menú principal de la aplicación para identificación.

```
Ingrese la ruta del archivo excel a leer:C:\Users\DesktopMADI\Documents\TEST_4\M3.xls

Ingrese el numero de muestras tomadas: 960

Ingrese la columna del vector de datos(empezando de cero): 0

Ingrese la columna del vector de tiempo(empezando de cero): 1
Ingrese el tipo de sistema:
 1)POMTM  2)PDMTM
```

Figura 118. Menú de la aplicación para identificación

Para la lectura se debe de ingresar la ruta completa del archivo Excel® a leer con su respectiva extensión. Posteriormente se deben ingresar el número de muestras o filas a leer, así como sus ubicaciones en la tabla empezando del cero, una vez ingresados se elige que modelo se va a identificar, primero orden más tiempo muerto (POMTM) o segundo orden más tiempo muerto (SOMTM). Cuando se ha hecho la elección la aplicación

devolverá los parámetros que ha calculado a partir de las ecuaciones descritas. La Figura 119 muestra el ejemplo cuando se hace la elección de un sistema de segundo orden

```
Los parametros del modelo de segundo orden mas tiempo muerto son:
El valor de Tau1 es: 84.350116
El valor de Tau2 es: 45.090044
El valor del tm es: -30.510160
El valor maximo es: 26.070000
Los parametros del modelo de Ziegler-Nichols son:
L = 16.940903
Tau= 171.920812
Las ganancias del PID segun Ziegler-Nichols son:
Kp= 12.177921
Ki= 0.359424
Kd=103.152487
Ti= 33.881805
Td=8.470451
```

Figura 119. Parámetros dados por la aplicación para un sistema de SOMTM.

Así mismo la aplicación retornará las gráficas correspondientes a la respuesta experimental, de acuerdo a las muestras, y la respuesta del modelo calculado con la finalidad de observar la diferencia entre cada uno. En el caso del sistema de segundo orden se visualiza de igual forma la ubicación del punto de inflexión y el trazo de la recta tangente como lo muestra la Figura 120.

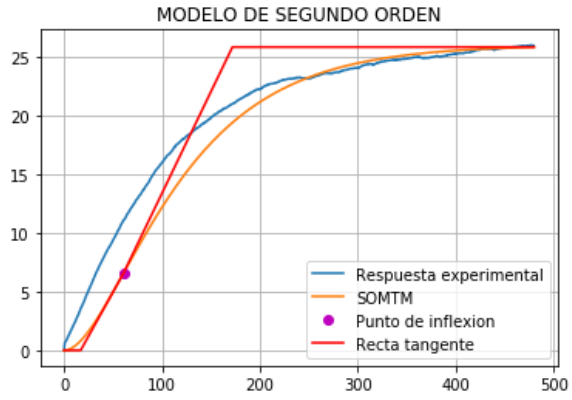

Figura 120. Ejemplo de gráfica para identificación de un modelo SOMTM.

4.5 SIMULACIÓN DEL CONTROL PID UTILIZANDO PYTHON®

Utilizando otro script de Python se ha simulado la respuesta del sistema una vez ingresados las ganancias del controlador. En la Figura 121 se muestra la respuesta del sistema al

ingresar el control y las funciones de transferencia de cada bloque utilizando el método de Ziegler-Nichols.

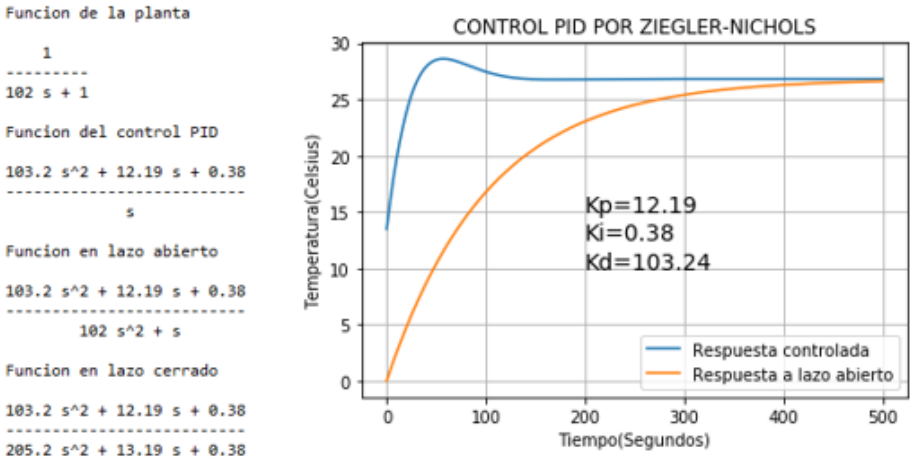

Figura 121. Simulación de la respuesta con el control PID por Ziegler-Nichols.

Se ha utilizado como modelo de la planta para la simulación el modelo de primer orden ignorando el retardo ya que éste es mínimo y es la más cercana a la respuesta experimental obtenida mediante las muestras. La Figura 122 corresponde a la simulación del controlador utilizando la adaptación a las reglas de Ziegler-Nichols.

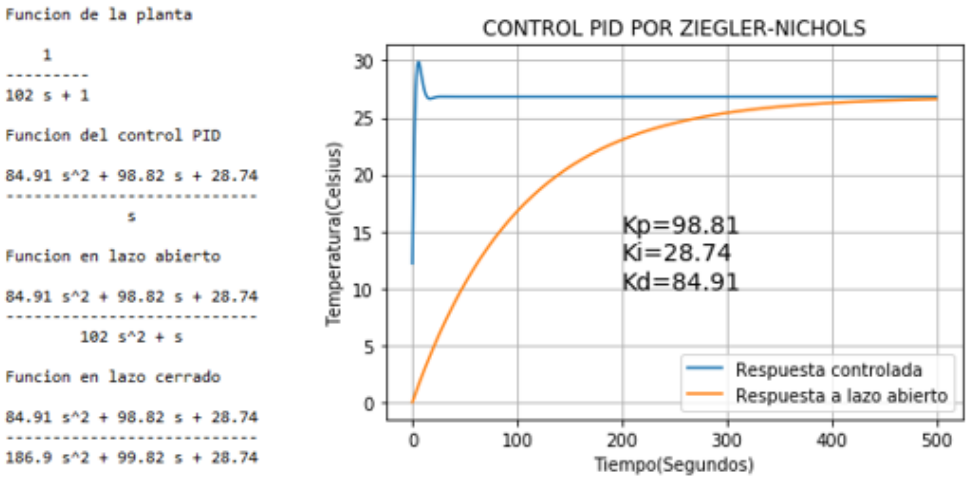

Figura 122. Simulación de control PID por adaptación de Ziegler-Nichols.

Así mismo se pueden ver las funciones de transferencia del controlador en el espacio de Laplace con la finalidad de que puedan analizarse matemáticamente.

4.6 IMPLEMENTACIÓN DE LA INTERFAZ DEL CONTROL PID EN NODE-RED®

Habiendo hecho la metodología necesaria para realizar el control PID se procedió a programar la interfaz visual utilizando la herramienta Node-Red® instalada de forma predeterminada en la tarjeta Raspberry Pi®.

Para el apartado del manejo de tramas se utilizaron los nodos que presenta la Figura 123 que contienen las funciones de transformación de valores numéricos a cadenas y así como la interfaz para la adquisición de datos.

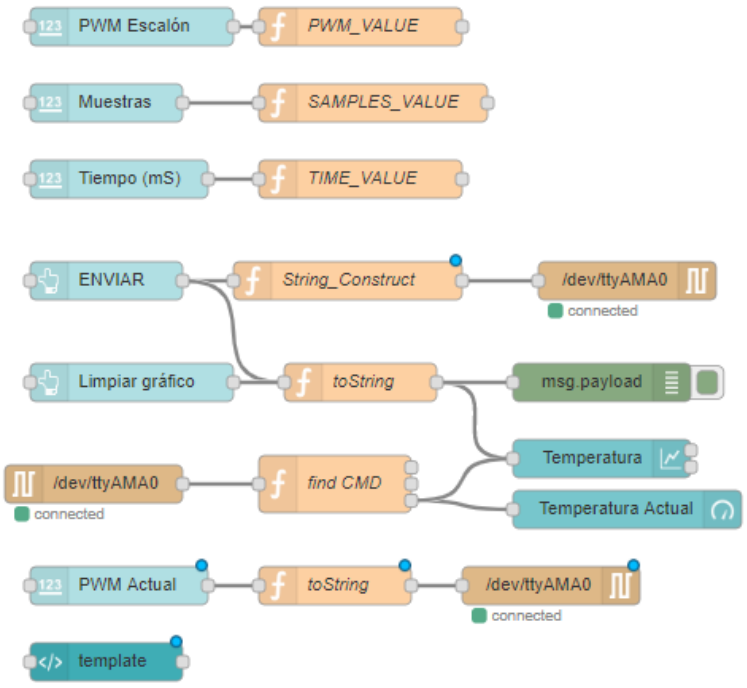

Figura 123. Nodos para la creación de tramas completo en Node-Red®.

En el nodo *Template* se carga la dirección IP correspondiente a la transmisión de video que, al funcionar de manera satisfactoria, no hubo cambios en su metodología de implementación en comparación con la utilizada en Flask®. Con la configuración anterior se montó la interfaz visual que se presenta en la Figura 124. La interfaz contiene, a diferencia del servidor de Flask®, controles interactivos que pueden ser cambiados de forma dinámica sin necesidad de refrescar la página web. Las gráficas son dinámicas y puede verse rápidamente el cambio en la experiencia del usuario en comparación de otras alternativas.

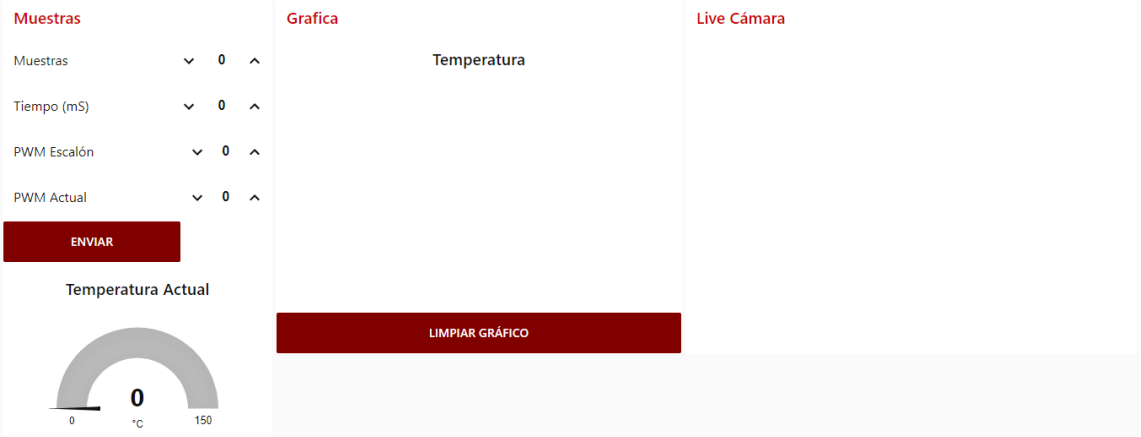

Figura 124. Interfaz para la adquisición de datos.

Una vez que se ha programado la interfaz para la adquisición de datos, se diseñó y programó otro apartado donde se encontrarán todos los parámetros para el cambio de ganancias del controlador PID con la finalidad de que pueda utilizarse cualquier valor y experimentar con los cambios que se tienen al cambiar cada uno de las variables. La Figura 125 muestra la programación en Node-Red®.

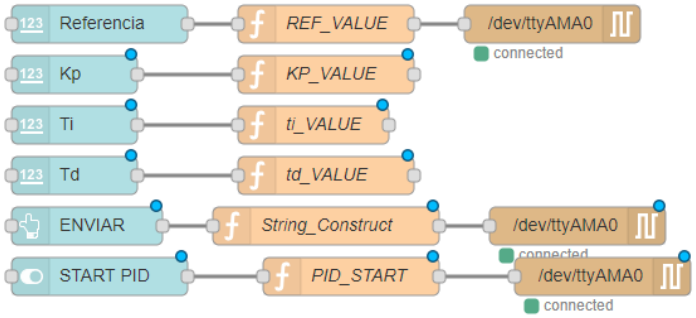

Figura 125. Programación de la interfaz para el controlador en Node-Red®.

El programa anterior genera como resultado la interfaz visual que se ve en la Figura 126. Contiene los controles para cambiar las ganancias, así como la información necesaria para observar los parámetros más utilizados en el análisis de sistemas dinámicos y poder comparar los resultados prácticos con los que se puedan obtener en la simulación. Las gráficas pueden “limpiarse” cada que el usuario lo desee y cambiar de modo entre adquisición de datos y el uso del controlador.

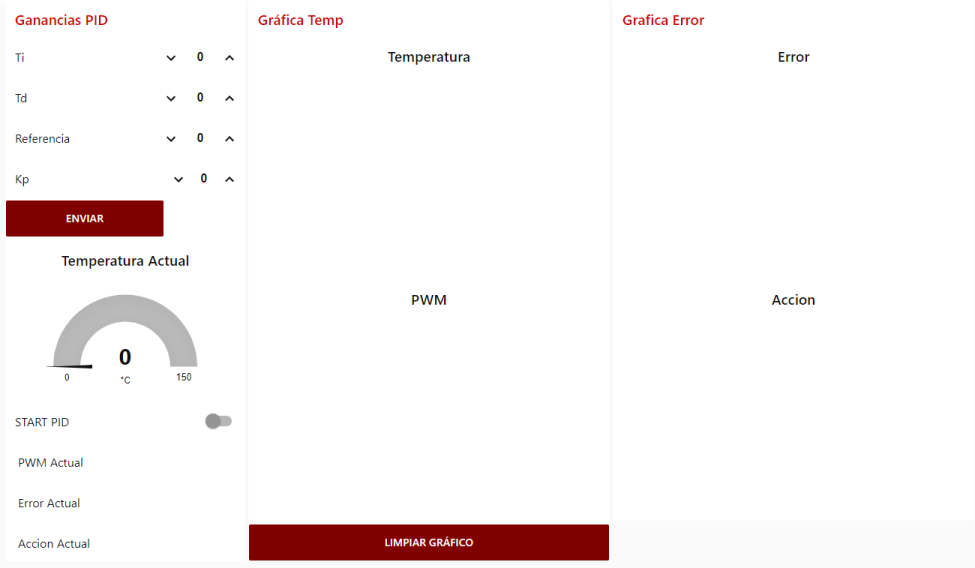

Figura 126. Interfaz para el controlador.

Con las interfaces programadas se realizaron las pruebas para su correcto funcionamiento obteniendo los resultados de la Figura 127 con la adquisición de datos.

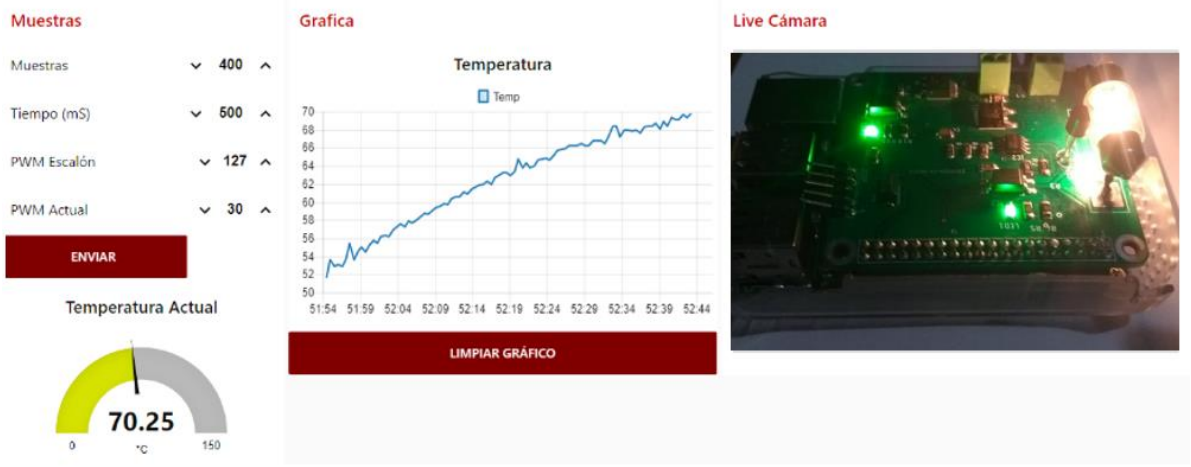

Figura 127. Ejemplo de adquisición con la plataforma.

Posteriormente se utilizó para probar la ganancia de los controladores mediante pruebas aleatorias para observar la generación de las gráficas que se colocaron para su análisis siendo la Figura 128 un ejemplo de ello.

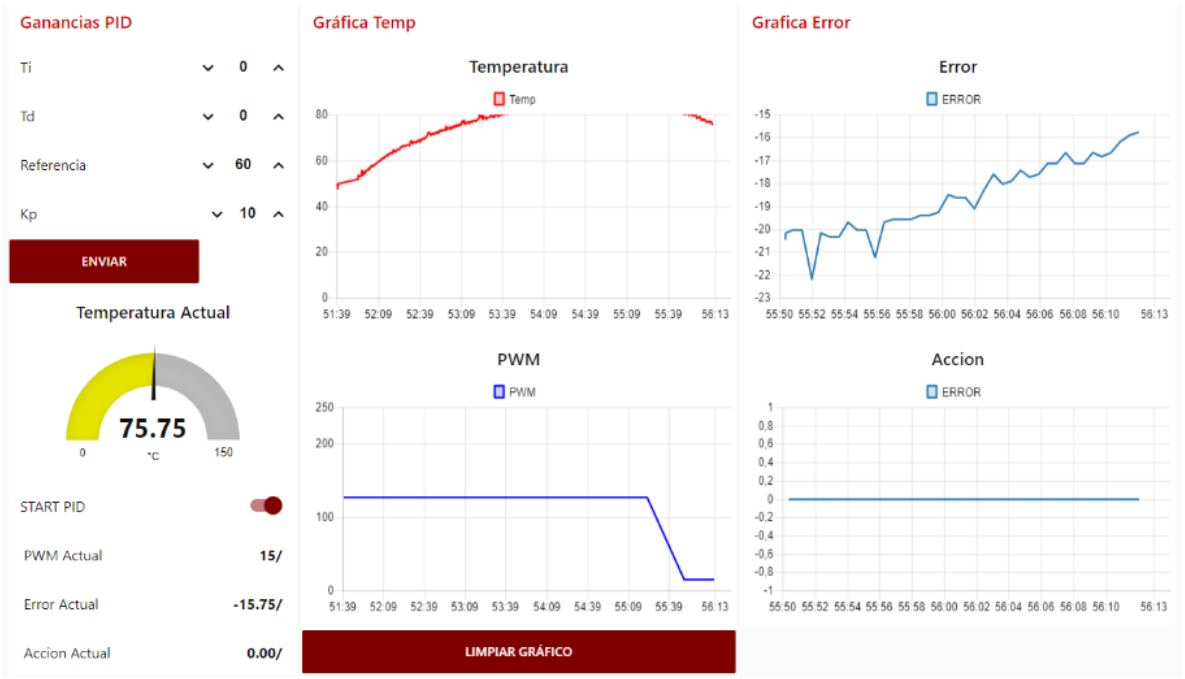

Figura 128. Ejemplo del uso de la interfaz del controlador.

4.7 PRUEBAS CON EL ENTRENADOR DE PIC18F2550®

Con la finalidad de comprobar el funcionamiento de la plataforma se realizó un código de prueba utilizando el reloj de tiempo real en combinación con el display de siete segmentos obteniendo los resultados de la Figura 129.

La ventaja del uso de display en miniatura es que la resolución que se obtiene en conjunto con la cámara es bastante buena, teniendo visualización de todos los números de manera clara

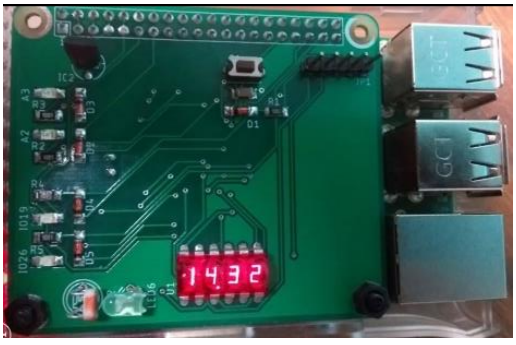

Figura 129. Circuito de prueba utilizando el reloj de tiempo real.

Al realizarse las pruebas con el bootloader se tuvieron resultados satisfactorios siempre y cuando los códigos no incluyeran rutinas de interrupción. Este problema si bien no afecta las prácticas básicas, al momento de utilizar este tipo de recursos el microcontrolador no funciona correctamente. Analizando el código fuente del arrancador, si bien se encuentra en la parte baja de la memoria, no remapea los vectores de interrupción. Para remapear estos vectores se utiliza el código que se muestra en la Figura 130.

```
#include <18F2550.h>
#device adc = 10
#fuses HS,MCLR,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOUSBDIV,PLL1,CPUDIV1,NOVREGEN,CCP2B3
#use delay(clock=20M)
#define MAX_FLASH 0x8000 //for PICs with 32KB of mem
#define LOADER_SIZE   0xFF   //tinybld size + a bit more (200 bytes is enough)
#org MAX_FLASH-LOADER_SIZE , MAX_FLASH-1 void boot_loader(void) {}
```

Figura 130. Configuración del microcontrolador para Tiny Btld

También se muestra la configuración de los fuses mínima que se requieren para el funcionamiento del microcontrolador, como lo es el tipo de oscilador externo que se está usando, la división de frecuencia que se tiene y la frecuencia de trabajo para ejecutar las instrucciones. Todas las configuraciones pueden cambiarse modificando el archivo ensamblador del bootloader de acuerdo a las necesidades que tengan los usuarios.

4.7.1 INTERFAZ VISUAL EN NODE-RED

Aparte de la interfaz visual con la cámara web que se utilizó en la creación del sistema de control de temperatura, se tiene un apartado diferente con la página web que tiene los algoritmos para la transferencia de los códigos hexadecimal desde internet.

Todos los sistemas de la plataforma cuentan con esta interfaz en caso de que se quiera cargar códigos propios de cada usuario y no limitarse solamente a los que se tienen en la interfaz. La página web para carga de archivos es sencilla pues la tarea no requiere una estética muy alta y es la que se ve en la Figura 131.

Figura 131. Página de carga de archivo

4.8 LEVITADOR DE AIRE

Con las piezas fabricadas se procedió a realizar el montaje de las mismas para obtener la estructura final del levitador. Si bien las piezas estaban hechas a la medida, debido al tamaño de la lámina de HDPE fue necesario utilizar tornillos para que las piezas quedaran fijas unas con otras. Las piezas creadas son las que se ven en la Figura 132.

Figura 132. Piezas del levitador fabricadas

Para el flujo de aire se montó un tubo de acrílico de 75cm de alto y un diámetro de 25mm, altura a la que se realizaron las mejores pruebas para el levitador. Como elemento levitante se fabricó un disco de HDPE de color azul para realizar un contraste con la estructura general. El disco tiene diámetro de 24.5mm con la finalidad de que quedara justo y no tuviera muchos movimientos o giros evitando problemas al momento de modelar. La estructura completa y montada es la que se ve en la Figura 133.

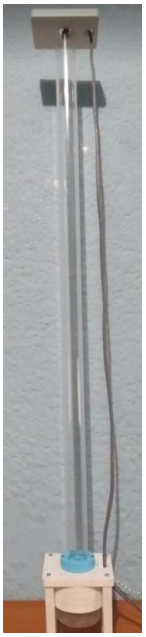

Figura 133. Levitador montado

Siguiendo los algoritmos programados anteriormente se realizaron las pruebas de modelado haciendo un paso al escalón. El voltaje de alimentación del sistema está en 18V, que son los necesarios para que el disco se eleve de forma suave y sin presentar impulsos muy bruscos.

La resolución del PWM se encuentra a 10 bits en esta ocasión. Tomando en cuenta que este sistema es difícil de modelar pues sus características son distintas al sistema de temperatura ya que influyen en él muchos aspectos físicos, las pruebas realizadas fueron utilizando métodos experimentales para dar con los parámetros justos para el modelado.

Se realizaron cuatro pruebas en paso al escalón con 200 muestras en un tiempo de muestreo de 250ms a una magnitud de 600 cuentas. Este número se obtuvo a partir de las pruebas para saber en qué número el disco comenzaba a elevarse y mantenerse estable. Posteriormente se hizo un promedio de muestras y se obtuvo la curva de respuesta que se ve en la Figura 134.

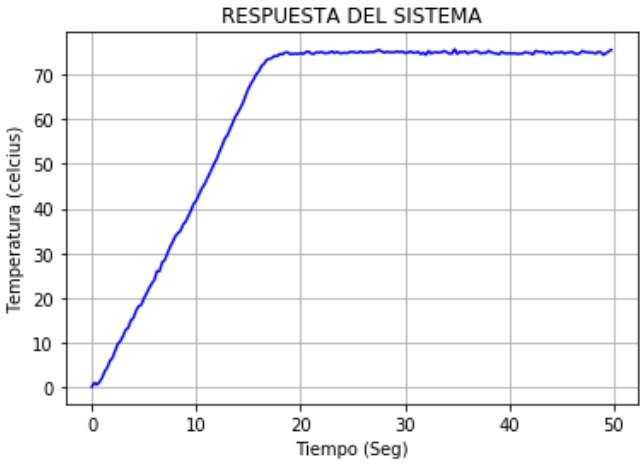

Figura 134. Curva de respuesta del levitador.

Como se puede observar, la curva no es de orden uno, por lo que sintonizar el controlador con los métodos descritos es una tarea difícil de realizar. Sin embargo, se puso a prueba el script de Python® para modelado matemático obteniendo resultados ligeramente cercanos a los que podrían tenerse con un sistema de primer orden puro por naturaleza.

Habiendo realizado las pruebas, se concluyó que modelar con un sistema de primer orden utilizando el método de Alfaro no era posible pues las curvas no están en nada aproximadas

a las que se desean. La gráfica de la Figura 135 muestra el resultado si se intenta modelar con un sistema de primer orden con retardo.

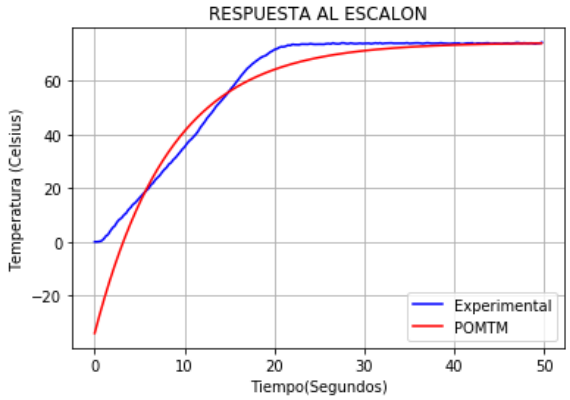

Figura 135. Prueba de modelado del levitador con un sistema POMTM

Repitiendo el experimento con un sistema de orden dos e intentando sintonizar el controlador por los métodos de Ziegler-Nichols se obtienen resultados que pueden aproximarse a la sintonización real del sistema, pues la curva no se encuentra tan distante de la que se tiene al momento de aplicar los algoritmos. La Figura 136 presenta la gráfica de la sintonización utilizando Z&N así como los valores de las constantes del tiempo del modelo y las ganancias del controlador PID.

A partir de las pruebas se concluye que para sintonizar este tipo de sistema puede aplicarse el método experimental a prueba y error, pues sacar al sistema de su estado estable no presenta riesgo o daño a la estructura del levitador, permitiendo al usuario aplicar los métodos que mejor convenga para su control.

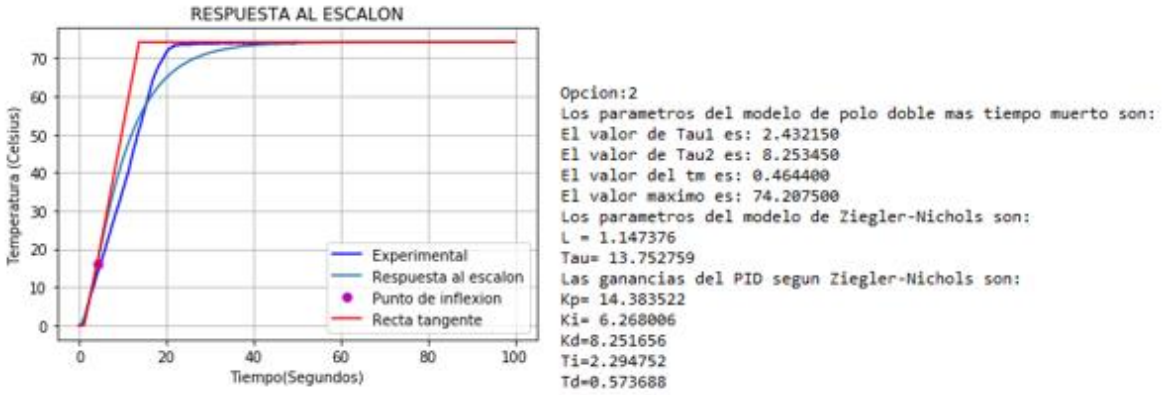

Figura 136. Modelado del levitador con un sistema de orden dos.

Habiendo realizado la interfaz para el manejo del sistema de levitación de aire se realizó una prueba con la pantalla del controlador PID de la plataforma. Una vez ingresados los parámetros del controlador que se arrojaron en el código de diseño se obtuvieron resultados favorables.

La curva que presentó el sistema al colocar una referencia es la que se muestra en la Figura 137. La ganancia Kp del sistema fue ajustada pues si bien las ganancias son una aproximación es difícil garantizar que todas ellas funcionen de manera correcta una vez que se aplican físicamente.

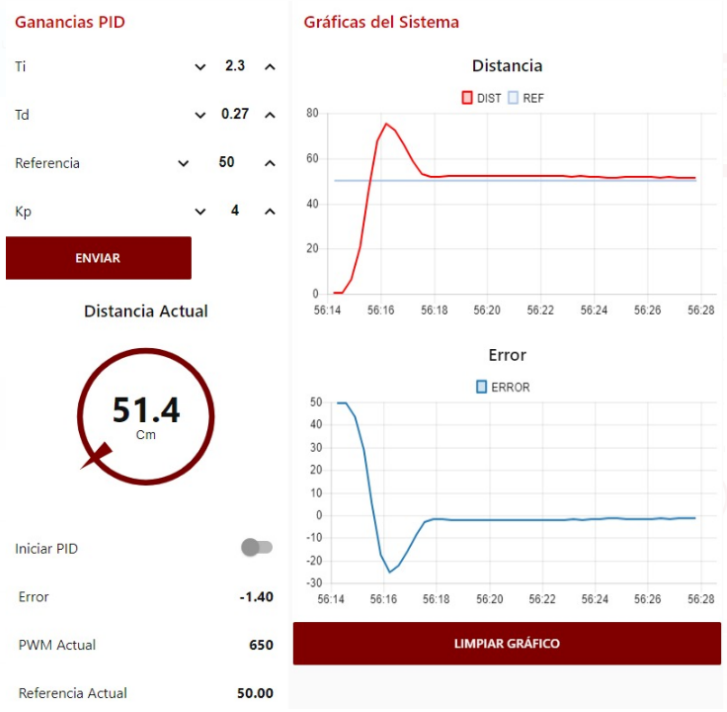

Figura 137. Prueba del controlador PID en la plataforma

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES.

5.1 CONCLUSIONES

Los sistemas automáticos están en constante evolución, los desarrolladores tanto de hardware y de software requieren de mejores herramientas para incrementar el número de productos que puedan mejorar la calidad de vida de las personas, ya sea dándoles confort, creando productos que ayuden en aspectos de la salud y elaborar proyectos más complejos que permitan una mayor producción industrial a un menor costo y con mejores ganancias.

Una de las partes más importantes para poder tener todas las mejoras mencionadas es el aprendizaje, estudio y aplicación de sistemas de control automático, ya que de ellos depende que todos los equipos que se diseñen desempeñen su tarea de la manera más óptima. Ejemplos de sistemas de control automático que se encuentren actualmente en desarrollo pueden verse en los vehículos no tripulados, sistemas agrícolas para el control de todas las variables necesarias en el cultivo de diversos productos y que permiten su cosecha, aunque no se tengan las condiciones meteorológicas necesarias.

Si bien la tecnología se está aplicando en entornos industriales, también se están desarrollando productos que permitan a las personas realizar las tareas domésticas de una mejor manera con el fin de tener un desarrollo sustentable en sus hogares, monitoreando, controlando y reduciendo el consumo de los servicios primarios como lo es la luz eléctrica, el gas y el consumo de agua. En este tipo de aplicaciones el uso de sistemas automáticos es de vital importancia para solo utilizar los recursos que sean necesarios para cumplir con las tareas diarias de las personas.

Por todo lo anterior es necesario que los ingenieros electrónicos y todos los involucrados en el desarrollo de éstas tecnologías tengan los conocimientos básicos en el diseño, desarrollo e implementación de sistemas mecatrónicos. De acuerdo a lo realizado en este proyecto, actualmente no es posible solamente tener conocimientos en el área de electrónica para crear sistemas automáticos óptimos, los desarrolladores deben de ser multidisciplinarios teniendo bases de informática, electrónica y mecánica, pues son los principales elementos con los que cuenta un proyecto tecnológico que deberá de ser implementado.

Es importante remarcar que, si bien existe el conocimiento teórico de fácil acceso, este tipo de teoría no es fácil de comprender si no es aplicada en elementos físicos reales que permitan contrastar la teoría con la práctica.

Las instituciones educativas deben de buscar diferentes metodologías que permitan a los docentes crear herramientas para que los alumnos obtengan la combinación de teoría y práctica que los brinde de mejores competencias que podrán utilizar cuando lleguen al entorno laboral o de investigación de acuerdo a lo que ellos deseen.

Se deben aprovechar todas las ventajas que nos brindan las tecnologías de la información y la comunicación y la basta información con la que las personas pueden hacerse ingresando a las diferentes plataformas que existen para compartir el conocimiento y que son de uso libre.

Uno de los principales problemas de los que se tuvieron en la realización de este proyecto fue encontrar herramientas o plataformas que permitieran el desarrollo del sistema sin necesidad de utilizar software de paga. Como estudiantes o emprendedores, no se cuentan con los recursos económicos para adquirir los softwares o plataformas más utilizados para diseñar, analizar e implementar las diferentes etapas de los proyectos. Es por ello que se buscaron alternativas que fueran de uso libre y en las que se tuviera una documentación suficiente para tener el conocimiento de forma autodidacta y de manera rápida.

Habiendo diseñado e implementado el proyecto se puede constatar que existen plataformas de uso libre que son lo suficientemente potente para el desarrollo de aplicaciones ya sea muy básicas o complejas y que la comunidad del Open Source va incrementando día a día permitiendo que los ingenieros o desarrolladores tengan independencia tecnológica y no dependan de terceros para implementar y crear aplicaciones.

Así mismo, es importante el compartir el conocimiento y la base de las aplicaciones que se crean utilizando este tipo de sistemas, pues como ha sucedido en este proyecto, muchos de los problemas se solucionaron utilizando herramientas creadas por personas comunes e investigadores que tienen sus propias plataformas para compartir conocimiento y que se encuentran en la disponibilidad de liberar su trabajo con la finalidad de facilitar la tarea a futuros desarrolladores.

Los métodos para la sintonización de controladores han avanzado, sin embargo, los principales métodos que fueron creados en el siglo pasado se siguen aplicando por la sencillez y el buen funcionamiento que se han tenido en su implementación. El más utilizado es el método de Ziegler y Nichols que hoy en día se encuentran presentes en los equipos industriales de alta gama por su extenso estudio sobre ellos, por esta razón este proyecto se basó en él.

Al ser un método que fue creado tiempo atrás, no está planeado para su desarrollo con los sistemas informáticos actuales, por lo que fue necesario encontrar una metodología teórica que permita encontrar sus parámetros de una manera más exacta sin utilizar métodos visuales.

En conclusión, este proyecto permitió fortalecer competencias en las diferentes áreas de la mecatrónica, pues fue un desarrollo conjunto utilizando las áreas que forman este conocimiento como lo son la electrónica (presente en el desarrollo de los circuitos, cálculo de variables y aplicación de la teoría de control), la mecánica (que ayudó al desarrollo de la estructura física de los proyectos así como el análisis de los sistemas y la creación de los prototipos diseñados) y la informática (creando las interfaces visuales y explotando los recursos informático en aras de crear los programas que facilitan el cálculo, diseño y fabricación de las diferentes partes).

Se creó una plataforma práctica, útil y sencilla son elementos libres y sin gastos de licencia o permisos de algún tercero que pudieran interferir al momento de su venta o inclusive de su implementación, por lo que se afirma que las herramientas de uso libre pueden ser una opción viable, si bien son sencillas y pueden tener errores pues son creadas generalmente por una sola persona, cada usuario puede alterarlas para su interés propio.

5.2 RECOMENDACIONES

Este proyecto se centró en el diseño del hardware necesario para el uso de diferentes elementos físicos mediante el mini ordenador Raspberry Pi®. Sin embargo, al momento de comprobar su funcionamiento se constató que el diseño puede mejorarse pero que pueden elevar el costo de fabricación. Ejemplo de ello es el cambio de la señal de control, que se basa en modulación de ancho de pulso, siendo más óptimo la implementación de un sistema de conversión digital – análogo, ya que el primero no es lineal y presenta un gasto de procesamiento al tener que realizar más operaciones matemáticas.

Los algoritmos presentados son los más comúnmente utilizados en la enseñanza de los sistemas de control y que se basan en el uso experimental, pero no son únicos y pueden implementarse diferentes técnicas de control como lo pueden ser la lógica difusa, redes neuronales o controles predictivos. Queda a gusto del usuario el método a utilizar de acuerdo a las necesidades que tenga.

Los modelos matemáticos y la sintonización de los controladores se realizaron de manera experimental, pues es lo que normalmente se va a encontrar en la práctica, en caso contrario pueden analizarse los fenómenos físicos presentes en los sistemas para la obtención de su función de transferencia de manera analítica y contrastar ambos resultados en búsqueda de llevar la teoría a la práctica, comprobando si ambas se complementan de manera correcta.

Existen diferentes herramientas que actualmente se están implementando y que cuentan con mejores recursos de hardware y de procesamiento que pudiesen aplicarse en conjunto con la Raspberry Pi® en busca de un funcionamiento más óptimo de los sistemas, se recomienda su puesta a prueba con la finalidad de incrementar las posibilidades de plataformas y no quedarse estancado en el uso de una sola.

La implementación de plataformas web es un área inmensa del conocimiento y que una persona con formación en ingeniería electrónica no tiene como especialidad, por lo que se aconseja mejorar los algoritmos de programación web y buscar el uso de lenguajes de programación más potentes que permitan mejorar los recursos con los que se cuenta. Ejemplo de ello es que no se tiene actualmente un sistema de gestión de usuarios que se encuentre en una base de datos. Sin embargo, actualmente se trabaja en su creación mediante la participación de desarrolladores de software que realicen esta tarea de una mejor manera y en menor tiempo.

El trabajo a futuro de este proyecto es la mejora de la plataforma web para mejorar la experiencia del usuario y evitar los fallos que puedan tenerse en el servidor. Así mismo se está trabajando en la creación de más sistemas físicos a los que pueda aplicarse teoría de control, entre los proyectos que se han planificado se encuentran:

- Sistema de control de velocidad y posición de un motor de corriente directa.
- Sistema de balancín con hélice utilizando un giroscopio.
- Péndulo invertido.
- Sistemas de llenado de tanque – cisterna.

Todos los proyectos anteriores pueden encontrarse en trabajos previos, siendo de los más desarrollados por los investigadores en esta área del conocimiento y por ende existe una amplia documentación sobre su diseño, análisis y fabricación.

5.3 PUBLICACIONES Y PARTICIPACIONES EN CONGRESOS.

De acuerdo a los temas abordados en el presente documento se realizaron las siguientes publicaciones y participaciones en eventos:

- Ponencia: "*Desarrollo de interconexiones de periféricos para Raspberry Pi*" en el marco de la Semana de I+D del Instituto Tecnológico Superior de Lerdo, junio de 2017.
- Concurso: 4to Encuentro de Jóvenes Investigadores del Estado de Durango con el proyecto "*Plataforma Web para el desarrollo de prácticas de control de procesos utilizando instrumentación virtual remota*", septiembre 2017.
- Ponencia: "*Sistema de adquisición de datos con Python y Arduino*" en el marco del Tercer Congreso Internacional de Ciencias de la ingeniería, octubre de 2017. Y publicación en la revista Ciencia, Ingeniería y Desarrollo Tec Lerdo 2017, ISSN2448-623X Volumen 1, Número 3,
- Exposición: "*Sistema de adquisición de datos con Python y Arduino*" en el marco de la Semana de I+D del Instituto Tecnológico Superior de Lerdo, diciembre de 2017.
- Publicación: "*Identificación e implementación de un controlador PID de temperatura basado en la curva de respuesta*" en la revista Ciencia, Ingeniería y Desarrollo Tec Lerdo 2018, ISSN2448-623X Volumen 1, Número 4.

BIBLIOGRAFÍA

Aguirre, I. J. (2011). Instrumentación Virtual aplicada al diseño de sistemas digitales de control. *IEEE-RITA*, 40-48.

Alciatore, D. (2008). *Introducción a la mecatrónica y los sistemas de medición.* México: McGraw-Hill.

Alfaro, V. (2006). Identificación de procesos sobreamortiguados utilizando técnicas de lazo abierto. *Ingeniería*, 11-25.

Alfaro, V. M. (2006). Identificación del modelo de orden reducido a partir de la curva de reacción del proceso. *Ciencia y tecnología*, 197-216.

Alfaro, V. M. (2007). Método de identificación de modelos de orden reducido de tres puntos 123c.

Ander, E. (2011). *Aprender a investigar: nociones básicas.* Argentina: Brujas.

Arduino . (20 de Noviembre de 2016). *Arduino Referencias*. Obtenido de Arduino Referencias: https://www.arduino.cc/en/Reference/StringObject

Areny, R. (2004). *Sensores y acondicionadores de señal.* México : Marcombo.

Aretio, L. G. (2001). *La educación a distancia; ayer y hoy.* Madrid: UNED.

Arrieta, Á. (2014). Sistema multipotenciostato basado en instrumentación virtual. *Ingeniería, Investigación y Tecnología*, 321-337.

Arriola, Ó. (2008). Sistemas integrales para la automatización de bibliotecas basados en software libre. *Acimed*.

Asadi, A. (2014). *Raspberry Pi for Beginners.* UK: Imagine Publishing.

Aström, K. (2009). *Control PID Avanzado.* España: Pearson Educación.

Barrera, L. (2016). Sistema domótico básico utilizando la tarjeta Raspberry Pi. *Universidad Distrital Francisco José de Caldas*.

Bolton, W. (2001). *Ingeniería de control.* Mexico, DF: Alfaomega.

Bolton, W. (2006). *Mecatrónica.* México: Alfaomega.

Boylestad, R. (2009). *Electrónica: Teoría de circuitos y dispositivos electrónicos.* México: Pearson Educación.

Boylestad, R. (2011). *Introducción al análisis de circuitos.* México: Pearson Educación.

Calzada, M. (2005). Sistema de medición y adquisición de datos mediante microcontroladores. *Instituto Tecnológico de Veracruz*.

Camargo, C. (2015). Metodología para la Transferencia Tecnológica en La industria basada en Software Libre. *Universidad Nacional de Colombia* .

Cervantes, L. (2015). *Modelización matemática: Principios y aplicaciones .* Puebla: Benemérita Universidad Autónoma de Puebla.

Chiculita, C. (10 de 11 de 2018). *etc*. Obtenido de etc: http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm

Chiquillo, J. C. (2008). Aplicaciones de la instrumentación virtual en la educación tecnológica. *ITCA-FEPADE*, 18-20.

Coto, A. (2008). *Protocolos y comunicaciones de Red.* México: Cisco Networking Academy.

Creus, A. (2010). *Instrumentación Industrial.* México: Alfaomega.

Daprotis, M. (2011). *Función de Transferencia y respuesta impulsiva.* Argentina: Universidad Nacional del Sur.

Dorf, R. (2005). *Sistemas de Control Moderno.* Madrid: Pearson Educación.

Dormido, S. (2008). Muestreo, Control y Comunicación basados en eventos. *Revista Iberoamericana de Automática e Informática Industrial*, 5-26.

Drake, J. (2015). *Instrumentación Electrónica de comunicaciones.* Santander: Universidad de Cantabria.

Falcón, M. (2013). La educación a distancia y su relación con las nuevas tecnologías de la información y las comunicaciones. *Medisur*.

Flores, J. (2018). Diseño de un medidor de energía eléctrica bidireccional y monitoreo mediante aplicación móvil. *Revista Iberoamericana de las ciencias computacionales e informáticas*.

García, E. (2008). *Compilador CCS y Simulador Proteus para microcontroladores PIC.* México: Alfaomega.

García, L. (2009). *Control Digita, Teória y Práctica.* Medellín: Politécnico de Colombia.

Gaviño, R. H. (2010). *Introducción a los sistemas de control: Conceptos, aplicaciones y simulación con MATLAB.* Mexico: Pearson Educación.

Girón, G. (2005). Origen y desarrollo de la educación a distancia en México. *Encuentro internacional de Educación Superior*.

GNU. (04 de Abril de 2018). *GNU*. Obtenido de https://www.gnu.org/philosophy/free-sw.es.html

Gomáriz, S. (2001). *Teoría de control: Diseño electrónico.* Barcelona: UPC.

González, J. A. (2015). Adaptación de las reglas de Ziegler - Nichols a los parámetros de un sistema de primer Orden. *CID Tec Lerdo*, 73-80.

Gupta, N. (2014). Virtual Instruments in the Open Source world. *International Journal of Science and Research*, 617-621.

Halterman, R. (2011). *Learning to program with Python.*

Hidalgo, A. (2008). *Introducción a los sistemas de control y modelado matemático de sistemas invariantes en el tiempo.* Argentina: Universidad Nacional de San Juan.

Huircán, J. (s.f.). Conversores Análogo - Digital y Digital - Análogo: Conceptos básicos. *Core AC*.

Jimenez, O. (2014). Controlador PID-PD para sistemas lineales inestables de segundo orden con retardo. *Convención Científica de Ingeniería y Arquitectura*.

Joglar, J. (2017). Interpolación inversa de características de respuesta transitoria temporal a parámetros de la función de transferencia típica de segundo orden. *Revista Iberoamericana de Automática e Informática Industrial*, 44-55.

Juca, X. (2016). La educación a distancia, una necesidad para la formación de los profesionales. *Universidad y sociedad*.

Kamen, E. (2008). *Fundamentos de señales y sistemas usando la Web y MATLAB.* México: Pearson Educación.

Kuo, B. (2008). *Sistemas de Control Digital.* México: Grupo Editorial Patria.

LLedó, E. (2012). Diseño de un sistema de control domótico basado en la plataforma Arduino. *Universidad Politécnica de Valencia*.

Lozano, L. (2012). Diseño, Implementación y Validación de un controlador PID autosintonizado. *Tecno Lógicas*, 33-53.

Márquez Rubio, J. (2010). CONTROL BASADO EN UN ESQUEMA OBSERVADOR PARA SISTEMAS DE PRIMER ORDEN. *Revista Mexicana de Ingeniería Química*, 43-52.

Martínez, H. H. (2006). Evolución de la instrumentación electrónica programable. *Temas de ciencia y tecnología*, 33-42.

Mateu, C. (2004). *Desarrollo de aplicaciones Web.* Catalunya: FUOC.

MatplotlibOrg. (13 de Noviembre de 2016). *matplotlib.org*. Obtenido de matplotlib.org: matplotlib.org

Medina, P. L. (2011). Los Laboratorios Virtuales y remotos en la enseñanza de la ingeniería. *Revista Internacional de Educación en Ingeniería*, 24-30.

Miyara, F. (2004). *Conversores D/A y A/D.* Argentina: Universidad Nacional de Rosario.

Muños, R. (2013). *Distrital Francisco José de Caldas.* Madrid: Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación.

National Instrument. (11 de Mayo de 2018). *NI*. Obtenido de http://www.ni.com/data-acquisition/what-is/esa/

Nise, N. (2008). *Sistema de control para ingeniería.* México: Continental.

NumpyDevelopers. (13 de Noviembre de 2016). *numpy.org*. Obtenido de numpy.org: www.numpy.org

Nuñez, J. (2014). *Diseño e integración de un sistema de adquisición de datos con Arduino y Raspberry Pi .* México: Universidad Nacional Autónoma de México.

Ogata, K. (2010). *Ingeniería de control moderna.* Madrid: Pearson Educación.

Palacios, E. (2009). *Microcontrolador PIC16F84.* México: RA-MA Editorial.

Pérez, C. (2012). Plataforma embebida multipropósito para comunicación mediante protocolo MIL-STD. *CASE*, 237-242.

Pérez, E. M. (1995). *Instrumentación Electrónica.* Barcelona: Marcombo.

Pérez, M. (2014). *Instrumentación Electrónica.* España: Paraninfo.

Peters, T. (11 de Noviembre de 2016). *PEP 20 -- The Zen of Python*. Obtenido de PEP 20 -- The Zen of Python: https://github.com/python/peps/blob/master/pep-0020.txt

Peters, T. (10 de Mayo de 2018). *PEP 20-The Zen of Python*. Obtenido de https://github.com/python/peps/blob/master/pep-0020.txt

Ponsa, P. (2001). Actividades Docentes en Mecatrónica. *Universidad Politécnica de Catalunya*.

Python Software Foundation. (10 de Mayo de 2018). *Python*. Obtenido de www.python.org

Quiñones, C. (2011). Labview y la instrumentación virtual aplicadas a la docencia y la investigación en ciencias básicas. *Elementos*, 115-121.

Raspberry Foundation. (11 de Mayo de 2018). *Raspberrypi*. Obtenido de https://www.raspberrypi.org/documentation/raspbian/

Richardson, M. (2013). *Getting Started with Raspberry Pi.* United States: Maker Media.

Rojas, A. (2012). Diseño de tarjeta electrónica genérica para el control de motores trifásicos. *Ingeniería Investigación y Tecnología*, 21-31.

Ronacher, A. (12 de Noviembre de 2016). *Flask.org*. Obtenido de Flask.org: http://flask.pocoo.org/

Rossum, G. v. (2009). *El tutorial de Python.* Argentina: Python Software Fundation.

RPiFundation. (18 de Octubre de 2017). *RaspberryPi*. Obtenido de RaspberryPi: https://www.raspberrypi.org/documentation/linux/software/

Rubio, A. (2010). *Internet y enseñanza: La educación Virtual.* Universidad Complutense de Madrid.

Rugeles, R. C. (2002). La instrumentación virtual en la enseñanza de la ingeniería. *Acción pedagógica*, 74-84.

Ruíz, Á. (2013). Control Basado en Eventos de Sistemas de Primer Orden con Retardo. *Revista Iberoamericana de Automática e Informática Industrial*, 302-312.

Salcedo, M. (2016). Uso del microcomputador Raspberry Pi en estaciones meteorológicas. *Télématique*.

Sarthak, J. (2014). Raspberry Pi based interactive Home automation. *International Conference on Reliability, Optimization and Information Technology*.

ScipyOrg. (13 de Noviembre de 2016). *Scipy.org*. Obtenido de Scipy.org: https://docs.scipy.org/doc/scipy/reference/tutorial/general.html

Stallman, R. (2014). *Software Libre para una Sociedad Libre.* Madrid: Traficantes de Sueños.

SymPy Developers. (13 de Noviembre de 2016). *Sympy.org*. Obtenido de Sympy.org: www.sympy.org

Ugurlu, Y. (2011). Measuring the Impact of Virtual Instrumentation for. *IEEE Global Engineering Education Conference* , 152-158.

Universidad de Chile. (2008). *Como funciona la Web.* Chile: CIW.

Upton, E. (2012). *Raspberry Pi: User guide.* UK: Google Drive.

Valdes, F. (2008). *Microcontroladores: Fundamentos y amplifaciones con PIC.* México: Alfaomega.

Valera, A. (2014). Plataformas de bajo coste para la realización de trabajos prácticos de Mecatrónica y Robótica. *Revista Iberoamericana de Automática e Informática Industrial*, 363-376.

Vargas, Z. (2009). La investigación aplicada: una forma de conocer las realidades con evidencia científica. *Educación*, 155-165.

Velásquez, S. (2013). Monitoreo de Variables Analógicas usando Raspberry Pi. *Universidad, Ciencia y Tecnología*, 170-175.

Yime, E. (2016). Diseño mecatrónico de una Shield de arduino para el control de motores DC con escobillas. *Prospect*, 73-79.

Ziegler, J., & Nichols, N. B. (1942). Optimum Settings for Automatic Controller. *ASME Transactions*, 759-768.

Zubieta, J. (2016). *La educación a distancia en Mexico: una nueva realidad.* México: Virtual Educa.

ANEXOS

5.4 PREPARACIÓN DE LA RASPBERRY PI®

5.4.1 CONFIGURACIÓN DEL SISTEMA OPERATIVO

Si se utiliza la pantalla táctil original del fabricante lo más probable es que se tenga problemas con la posición de la pantalla. Abrir pantalla de comandos e ingresar lo siguiente:

5.4.1.1 Actualizar el sistema operativo:

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```

5.4.1.2 Rotar pantalla y Sistema táctil

- Abrir el archive de configuración de arranque

```
sudo nano /boot/config.txt
```

- Una vez abierto el archivo de configuración ingresar alguna de las siguientes líneas al final de éste de acuerdo a la rotación que se desee hacer.

```
lcd_rotate=1 #90 grados
lcd_rotate=2 #180 grados
lcd_rotate=1 #270 grados
```

5.4.1.3 Solución de problema con puerto serie en Raspberry Pi modelos 3

Las Raspberry Pi con bluetooth integrado utilizan los pines de la UART para su uso, por lo que deben de ser desactivados si se requiere hacer la comunicación por puerto serie mediante el hardware interno de la placa.

- Abrir el archive de configuración de arranque

```
sudo nano /boot/config.txt
```

- Agregar las siguientes líneas al archivo de texto de configuración de arranque

```
dtoverlay=pi3-disable-bt
enable_uart=1
```

- Ingresar el siguiente comando en la terminal

```
sudo systemctl disable hciuart
sudo reboot
```

5.5 INSTALACIÓN DE DEPENDENCIAS PARA LA PLATAFORMA

5.5.1 INSTALACIÓN DEL MANEJADOR DE LIBRERÍAS DE NODE-RED

- Descargar el paquete npm para node-red ingresando en la terminal de comandos:

```
sudo apt-get install npm
```

5.5.2 INSTALACIÓN DEL DASHBOARD EN NODE-RED

- Abrir el software node-red siguiendo la ruta *menú>Programación>Node-RED*. También es posible mediante la terminal utilizando el comando:

```
node-red-start
```

- Ingresar al host local en el navegador de internet mediante la url:

```
localhost:1880/
```

- En caso de utilizar acceso remoto, ingresar al puerto 1880 de la dirección IP de la Raspberry Pi. Si se desconoce se puede ingresar el siguiente comando para arrojarla desde la terminal

```
hostname -I
```

- Dentro de la interfaz de node-red ingresar al menú principal *Manage pallete>Palette>Install>node-red-dashboard>install*

5.5.3 INSTALACIÓN DEL STREAMING DE VIDEO

- Instalar la aplicación motion mediante el comando:

```
sudo apt-get install motion
```

- Modificar el archivo de configuración ingresando a la ruta:

```
sudo nano /etc/motion/motion.conf
```

- Buscar dentro del archivo las siguientes líneas y cambiarlas por las siguientes:

```
daemon on
width 480
height 320
framerate 30
output_pictures off
stream_localhost off
stream_maxrate 30
```

- Conectar cámara USB a la Raspberry Pi y revisar si ésta es detectada colocando en la pantalla de comandos:

```
lsusb
```

- Los siguientes comandos son los necesarios se ingresan en la terminal de comandos de acuerdo a la función que se quiera realizar.

```
sudo motion #iniciar
sudo service motion stop #detener transmisión
sudo service motion reload #reiniciar la transmisión de video
```

5.5.4 INSTALACIÓN DE LIBRERÍAS DE PYTHON 2.7

- Instalar la librería para puerto serie de Python 2.7 mediante el siguiente comando:

```
sudo apt-get install python-serial
```

- Instalar dependencia wx:

```
sudo apt-get install python-wxgtk2.8
```

- Instalar paquete de Excel mediante:

```
sudo apt-get install python-xlwt
sudo apt-get install python-xlrd
```

- Instalar librerías de matemáticas

```
sudo apt-get install python-numpy
sudo apt-get install python-matplotlib
sudo apt-get install python-scipy
```