UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE INGENIERÍA ELECTRÓNICA

PROYECTO TÉCNICO PREVIA A LA OBTENCIÓN DEL TÍTULO DE:

INGENIERO ELECTRÓNICO

TEMA:

"DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO BASADO EN RASPBERRY PI 3 Y ARDUINO MEGA PARA CONTROL Y MONITOREO REMOTO DE UN BRAZO ESTABILIZADOR"

AUTORES:

WILLIAN CHÁVEZ LAZ

GERARDO BARZALLO MUÑOZ

TUTOR:

ING. GABRIEL SANTIAGO GARCÍA VÁSQUEZ

GUAYAQUIL, ECUADOR, 2018

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA

Nosotros, **WILLIAN CHÁVEZ LAZ** y **GERARDO BARZALLO MUÑOZ**, alumnos de la Universidad Politécnica Salesiana, declaramos que los conceptos, análisis desarrollados y conclusiones del presente trabajo son de exclusiva responsabilidad de los autores.

Guayaquil, Septiembre 2018

_____ _____

Willian Chávez Laz

C.I. 0927200352

Gerardo Barzallo Muñoz

C.I. 0940671837

CERTIFICADO DE GESTIÓN DE DERECHOS DE AUTOR

Por medio de la presente, Willian Chávez Laz y Gerardo Barzallo Muñoz, declaramos que el proyecto técnico de titulación: **"DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO BASADO EN RASPBERRY PI 3 Y ARDUINO MEGA PARA CONTROL Y MONITOREO REMOTO DE UN BRAZO ESTABILIZADOR",** que se presentó ante el Consejo de la Carrera de Ingeniería Electrónica, es cedido en su totalidad, los derechos de propiedad intelectual que nos corresponden de este proyecto técnico, enunciado forma gratuita a favor de la Universidad Politécnica Salesiana.

Guayaquil, Septiembre 2018

_____

Willian Chávez Laz

C.I. 0927200352

_____

Gerardo Barzallo Muñoz

C.I. 0940671837

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN POR EL TUTOR

En calidad de Tutor del trabajo de titulación del proyecto técnico **"DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO BASADO EN RASPBERRY PI 3 Y ARDUINO MEGA PARA CONTROL Y MONITOREO REMOTO DE UN BRAZO ESTABILIZADOR",** realizado por los señores Willian Fabricio Chávez Laz y Gerardo Sebastián Barzallo Muñoz, estudiantes de la carrera de Ingeniera electrónica, ha sido concluida y revisada en su totalidad.

_____

(f) Ing. Gabriel García Vásquez

DEDICATORIA

Dedico este trabajo a mis padres Adelaida Laz y Gilbert Chávez, pilar fundamental de mi vida académica que a pesar de todos los tropiezos presentados en el trayecto su apoyo incondicional y sus consejos son los que hoy me permiten alcanzar esta meta.

A mis hermanos Gisela, Joseline, Raphael y Nilo (┼), por impulsarme y ser mi inspiración para no decaer en lograr cada meta propuesta.

A Tamara Silvestre, ahora mi esposa por brindarme su apoyo y alentarme a seguir mi carrera universitaria.

A Beylú, mi cuñada por todo su apoyo y consejos en lo largo del camino.

A mis tías, tíos, abuelos que siempre estuvieron pendientes de mi trayectoria estudiantil y me inculcaron lo importante de lograr este propósito.

Willian Fabricio Chávez Laz

DEDICATORIA

Este trabajo se lo dedico a Dios, por haberme dado la salud, fuerza, sabiduría e inteligencia para alcanzar esta meta en mi vida, ya que he logrado concluir mi carrera gracias a él.

A mi madre Fátima Isabel Muñoz Loor, por ser pilar fundamental en mi vida y en toda mi carrera estudiantil, por todos los valores inculcados desde mi infancia, por todo el cariño, apoyo, confianza, paciencia y amor dado.

A mi hermano Miguel Alejandro Barzallo Muñoz, por ser mi hermano un pequeño amigo que siempre esta hay para brindarme su ayuda.

A mi abuelita Melchora del Socorro Loor Rojas, por ser una amiga, madre y abuela, pilar fundamental en mi vida, por el apoyo, cuidado y amor brindado.

A la mujer de mi vida Michelle Lissette Carvajal Arreaga, mi ayuda idónea por su apoyo, cariño y amor brindado para poder alcanzar esta meta y no darme por vencido.

Una mención especial para mi amigo, padre y abuelito Jorge Galo Carrasco Valencia, quien me enseñó a no darme por vencido con su ejemplo, a luchar por alcanzar cada meta quien desde el cielo me inspira, ¡Te Extraño! Papi viejo.

Gerardo Sebastián Barzallo Muñoz

AGRADECIMIENTO

Agradezco a Dios por brindarme sabiduría y fortaleza para no dejarme caer y alentarme cada día a lograr esta meta.

Gracias a mis padres Adelaida y Gilbert por su empuje y motivación en cada día de clases a lo largo de la carrera, gracias a sus consejos y por estar pendientes de mí.

Gracias a mis hermanos Gisela, Joseline y Raphael, que me motivaban a seguir siendo su ejemplo y demostrarles que se logra lo que se propone.

Gracias a mis abuelos Alfonso Chávez y Teresa Jara que estaban pendientes de mis estudios, a Tamara Silvestre, mi esposa por alentarme a terminar mis estudios, Beylú Campoverde, a mi chiquita Rebecca, Juanita Ortega por su cariño y comprensión.

Agradezco infinitamente al Ing. Gabriel García por asesorarnos en este proyecto y compartir sus conocimientos y toda la experiencia brindada, Gracias a Stalin y Alejandro Arreaga, Ángelo Viteri mis más cercanos amigos y compañeros de estudios como Jairo Gallardo, Carlos Zúñiga, Jonathan Granda, Jhonny Núñez, Enrique Villamar, Danilo Burbano, a mis primos Byron y Nelly Pullas por ser un ejemplo para mí, a mi Madrina Bertha Jara, a mis tías Elsa, Amparito, Janeth Chávez, Zoila, Olivia Jara, a Lucia Salazar, a mi tío Emilio Laz.

Gracias a todos ellos por inculcarme lo importante que es superarse en la vida y motivarme a realizar este proyecto técnico.

Willian Fabricio Chávez Laz

AGRADECIMIENTO

Agradezco a Dios, por darme la sabiduría, guía y fortaleza diaria para poder superar todo desafío y alcanzar esta meta estudiantil.

Agradezco en especial a mi mamá Fátima Isabel Muñoz Loor, el pilar fundamental en mi vida estudiantil, por su perseverancia diaria, por su esfuerzo, sacrificio, amor y por cada una de las enseñanzas impartidas que fueron formando mi personalidad y permitieron poder culminar esta etapa estudiantil.

A cada uno de mis familiares por creer, confiar y por el apoyo desinteresado. En especial a mi abuelita Melchora del Socorro Loor Rosas, por su amor y confianza, a mi tía Mayra Dayanara Carrasco Loor por su confianza, amor, apoyo y por animarme a seguir adelante, a mi tío Jorge Demetrio Carrasco Loor, por su ayuda y apoyo.

Agradezco a la Universidad Politécnica Salesiana, a cada uno de los docentes, su labor muchas veces subestimada, enfocada en cuidar y permitirnos expandir nuestros conocimientos de cada una de las materias que cursamos, por haber formado parte fundamental en nuestro desarrollo académico y profesional.

Agradezco a cada uno de mis compañeros y amigos, ya que gracias a su amistad y apoyo moral han aportado para animarme a seguir adelante y culminar esta etapa.

Un agradecimiento sinceramente a mi tutor, Ing. Gabriel Santiago García Vásquez, quien guio, oriento, apoyo y corrigió en el proceso de desarrollo del proyecto técnico a través de sus conocimientos, profesionalismo, motivación y experiencia.

Gerardo Sebastián Barzallo Muñoz

RESUMEN

AÑO	TÍTULO	ALUMNOS	TUTOR	TEMA DEL PROYECTO
2018	INGENIERÍA ELECTRÓNICA	Willian Chávez Laz Gerardo Barzallo Muñoz	Ing. Gabriel García	"Diseño e Implementación de un Módulo Didáctico basado en Raspberry PI 3 y Arduino mega para control y monitoreo remoto de un brazo estabilizador"

El presente proyecto técnico: "Diseño e Implementación de un Módulo Didáctico basado en Raspberry Pi 3 Y Arduino Mega para Control y Monitoreo Remoto de un Brazo Estabilizador" consiste en la implementación de plataformas de software y hardware libre (Raspberry Pi 3 y Arduino Mega) que permita almacenar datos seleccionados en base a las prácticas y generados por la interacción del módulo con los usuarios finales para su posterior revisión utilizando un medio de un acceso inalámbrico dentro de una red local de datos. El tipo de control para el manejo de la balanza se denomina PID (Proporcional, Integral y derivativo), por medio del cual se logra poner en la posición de balance una esfera ubicada en el brazo de la balanza por medio de un sensor infrarrojo y un motor DC, además del microcontrolador Arduino Mega que se encargará del procesamiento de los datos recibidos desde el sensor.

Las aplicaciones se diseñaron de modo que permitan al usuario final desarrollar prácticas que incentiven el aprendizaje de nuevos métodos y tecnologías actualmente disponibles en el mercado, los módulos seleccionados para realizar las actividades

descritas en el presente proyecto se pueden llevar a buen término tanto por novatos en el ámbito de la electrónica como personas que cuentan con conocimientos sólidos de la materia.

Tomando en cuenta los conceptos recibidos por medio del personal docente de la carrera de Ingeniería Electrónica se abarca temas tan básicos como elaboración de control de sensores y actuadores, programación básica de varios lenguajes tanto para la configuración de los microcontroladores como de los interfaces a los que tiene acceso el usuario final, derivando en conceptos avanzados como lo son la generación de un PID para la estabilización de una esfera en un brazo del prototipo.

Haciendo uso de los conceptos aprendidos durante la carrera de Ingeniería Electrónica en las materias respectivas se aborda temas de automatización, con mayor énfasis en manipuladores industriales abarcando nociones básicas y técnicas del funcionamiento.

La característica de este proyecto es fomentar el interés por los lenguajes de programación que llevan estas plataformas como lo es Python en la Raspberry Pi 3 y de C++ para el Arduino mega, así como el lenguaje PHP para el servidor que permite interactuar con el usuario final.

Dentro del proyecto se presentan diez prácticas que se concatenan entre ellas para logar al final que el usuario tenga una visión general de lo que Arduino y Raspberry tienen para ofrecer, además es posible plantear prácticas adicionales ya que el módulo no se limita a lo descrito dentro de este proyecto y solo se limita por el conocimiento y curiosidad de la persona que lo manipule.

Palabras claves: Raspberry Pi 3, Arduino Mega, Python, C++, PHP, PID.

ABSTRACT

YEAR	TITLE	STUDENTS	TUTOR	TOPIC
2018	ELECTRONIC ENGINEERING	Willian Chávez Laz Gerardo Barzallo Muñoz	Ing. Gabriel Garcia	"Design and Implementation of a Teaching Module based on Raspberry PI 3 and Arduino mega for control and remote monitoring of a stabilizing arm"

The present technical project: "Design and Implementation of a Didactic Module based on Raspberry Pi 3 and Arduino Mega for Remote Control and Monitoring of a Stabilizing Arm" consists of the implementation of software platforms and free hardware (Raspberry Pi 3 and Arduino Mega) that allows to store selected data based on the practices and generated by the interaction of the module with the end users for its later revision using a means of a wireless access within a local data network. The type of control for the handling of the balance is called PID (Proportional, Integral and derivative), by means of which it is possible to put in the balance position a sphere located in the arm of the balance by means of an infrared sensor and a DC motor, in addition to the Arduino Mega microcontroller that will be responsible for processing the data received from the sensor.

The applications were designed to allow the end user to develop practices that encourage the learning of new methods and technologies currently available in the market, the modules selected to carry out the activities described in this project can be carried out successfully by newcomers in the field of electronics as people who have solid knowledge of the subject.

Taking into account the concepts received by the teaching staff of the Electronics Engineering course, it covers basic topics such as the elaboration of sensors and actuators control, basic programming of several languages both for the configuration of the microcontrollers and the interfaces to which the end user has access, deriving advanced concepts such as the generation of a PID for the stabilization of a sphere in an arm of the prototype.

Making use of the concepts learned during the career of Electronic Engineering in the respective subjects addresses issues of automation, with greater emphasis on industrial manipulators covering basic notions and operational techniques. The feature of this project is to encourage interest in the programming languages that these platforms carry such as Python in the Raspberry Pi 3 and C ++ for the Arduino mega, as well as the PHP language for the server that allows interaction with the end user.

Within the project ten practices are presented that are concatenated among them to achieve the end that the user has an overview of what Arduino and raspberry have to offer, it is also possible to propose additional practices since the module is not limited to what is described inside of this project and is only limited by the knowledge and curiosity of the person who manipulates it.

Keywords: Raspberry Pi 3, Arduino Mega, Python, C++, PHP, PID.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad la Universidad Politécnica Salesiana, Sede Guayaquil, respecto a la malla curricular de la Carrera Ingeniería Electrónica se dieron a conocer materias tales como sistemas microprocesados, informática industrial y teoría de control.

En microprocesados I se da a conocer cómo controlar uno o varios elementos eléctricos o electrónicos, a través de un lenguaje de programación entendible por el mismo.

De forma real un proceso industrial requiere de una entera evaluación y supervisión para impedir fallos en el producto final, o perjuicios en los componentes eléctricos o electrónicos que lo conforman.

Con la llegada de nuevas tecnologías como Raspberry y Arduino, y la necesidad de tener un control de manera remota, que permita evaluar y a la vez controlar un proceso industrial de forma inalámbrica, nace la idea de diseñar e implementar un módulo didáctico basado en raspberry Pi 3 y arduino mega para control y monitoreo remoto de un brazo estabilizador.

Se usó el Arduino y Raspberry Pi debido a que los dos manejan software libre que permite que cualquier persona con acceso a internet y conocimientos básicos de programación puedan realizar prácticas o diseño de proyectos en poco tiempo, además es importante destacar que estos módulos pese a ser económicamente accesibles tienen gran capacidad de procesamiento.

Además, los estudiantes pueden aprender cómo funciona el lenguaje de programación y la comunicación entre los dos microcontroladores para interactuar

por medio de software y hardware con distintos aplicativos y puertos del módulo didáctico.

Este proyecto tiene como objetivo general dar a conocer a cada uno de los alumnos de la carrera de Ingeniería Electrónica sede Guayaquil, sobre diferentes tecnologías aplicables que además de utilizarse para cualquier tipo de aplicación solo limitada por la imaginación y el conocimiento del usuario, también manejo de ciertos procesos industriales, minimizando costos.

PROBLEMA

1.1 Antecedentes

Se observó y se determinó en la Universidad Politécnica Salesiana sede Guayaquil, en la carrera Ingeniería Electrónica, la necesidad para los estudiantes y profesores en realizar prácticas académicas, en el laboratorio de control automático con referencia a las materias de sistemas microprocesados, informática industrial y teoría de control acerca de un módulo didáctico basado en Raspberry y Arduino con el fin de controlar un brazo estabilizador.

En la actualidad muchas empresas poseen un sistema de control y monitoreo que permite elaborar reportes en caso de incidentes o cambios en el estado del proceso o sistema, pero también es necesario poder realizar actividades de forma remota para evitar riesgos de ejecución de algún tipo de proceso.

En el ámbito académico se comprobó que existe una carencia de conocimientos para la implementación de un control proporcional integral diferencial en los motores DC, por lo cual es necesario dejar una constancia de cómo elaborarlo de manera efectiva reduciendo tiempos de ejecución en el sistema.

Como efecto del desarrollo del proyecto es aportar de manera práctica a los conceptos impartidos en las materias anteriormente mencionadas, con este proyecto se pretende cubrir en parte la necesidad de realizar prácticas que contengan la teoría de control y comunicación.

1.2 Importancia y Alcance.

Con este trabajo de titulación se diseñó e implemento de un módulo didáctico basado en RASPBERRY PI 3 y ARDUINO mega para control y monitoreo remoto de un brazo estabilizador

Los beneficiarios del proyecto serán los estudiantes que se encuentran en las materias de materias sistemas microprocesados, informática industrial y teoría de control, en el laboratorio de control automático, así mismo los docentes al impartir sus clases.

Se aspira que los estudiantes puedan comprobar la teoría con las prácticas indicadas en el actual proyecto, ya que este sistema de comunicación es empleado constantemente a nivel industrial.

Es importante integrar conocimientos teóricos y prácticos, por lo que se elaboró un banco de pruebas en el cual el docente explicará la teoría y el estudiante pondrá en práctica lo aprendido, con éste módulo se logrará el desarrollo del conocimiento, habilidades y destrezas cognitivas en el cual ayudará a mejorar el aprendizaje tanto teórico como práctico.

1.3 Delimitación

1.3.1 Delimitación Temporal.

El actual proyecto para el diseño desarrollo y construcción se lo realizo en el año 2017 - 2018.

1.3.2 Delimitación Espacial.

El actual proyecto se lo aplicará en la Universidad Politécnica Salesiana sede Guayaquil, en el laboratorio de control automático con referencia a la materia de sistemas microprocesados, informática industrial y teoría de control II y III.

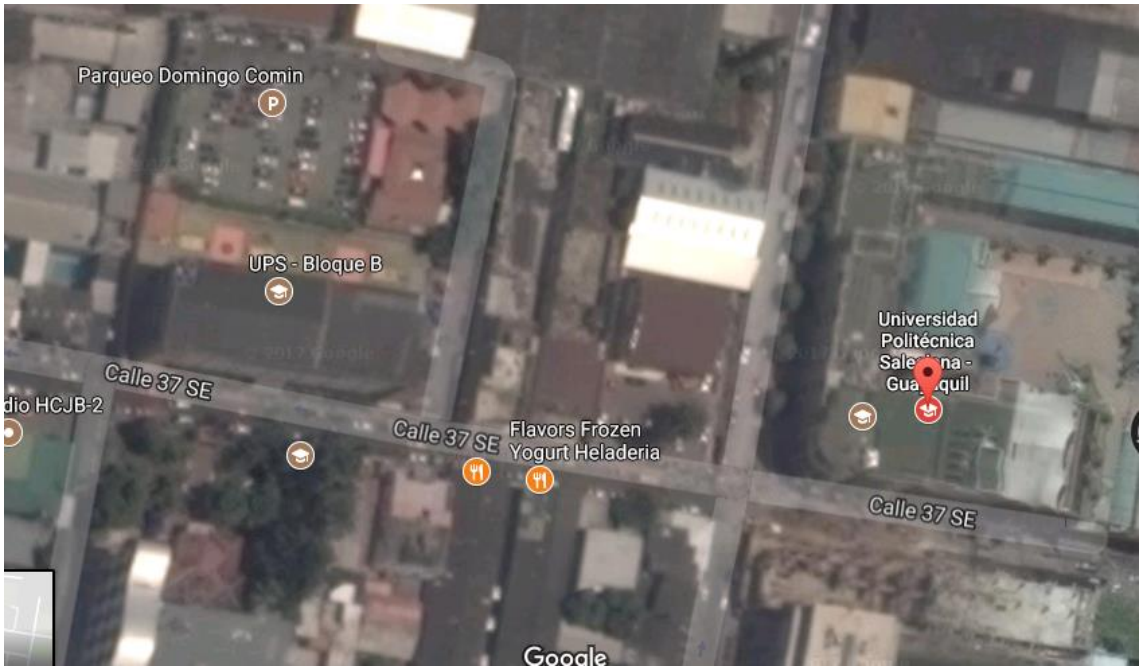


Figura 1. Ubicación Geográfica del proyecto
Google. (s.f.). [Mapa de Universidad Politécnica Salesiana, Ecuador en Google maps]. Recuperado el 3 de Junio, 2018, de: https://www.google.com/maps/search/ups/@-2.2200731,-79.8877959,233m/data=!3m1!1e3

1.3.3 Delimitación Académica.

En el presente proyecto se pondrá en práctica cada uno de los conocimientos adquiridos durante nuestra formación técnica como Ingeniero Electrónico con Mención en Sistemas Industriales dando énfasis a los temas relacionados con Automatización Industrial proyectos de control automático entre otros.

1.4 Hipótesis.

¿Qué aspectos están incidiendo para que en los laboratorios no exista un módulo didáctico basado en RASPBERRY PI 3 y ARDUINO para las prácticas de los estudiantes en las materias de sistemas microprocesados, informática industrial y teoría de control?

1.5 Justificación del proyecto

En la actualidad es necesario mejorar los aplicativos y desarrollar nuevas tecnologías reduciendo costos para afrontar las condiciones competitivas de mercado.

El presente proyecto permite conocer las ventajas de un control PID automatizado sin invertir en costosos equipos que utilizan la misma lógica para manejar procesos industriales.

Permite que los estudiantes logren una mayor comprensión de los temas a tratar en laboratorio, utilizando sus conocimientos teóricos y llevándolos a la práctica de forma estructurada y secuencial.

Del mismo modo es importante recalcar la utilización de software libre para el desarrollo de prácticas que aporten en la enseñanza de materias de control de procesos y automatización, de igual manera este tipo de herramientas solo se ven limitadas por el conocimiento y la innovación del usuario.

OBJETIVOS

2.1 Objetivo general

Diseñar e implementar un módulo didáctico basado en Raspberry Pi 3 y Arduino Mega para control y monitoreo remoto de un brazo estabilizador, para el laboratorio de control automático de la Universidad Politécnica Salesiana Sede Guayaquil.

2.2 Objetivos específicos

- Elaborar y equipar un módulo didáctico donde se encuentren los elementos de control y comunicación de un brazo estabilizador
- Diseñar la distribución interna del tablero de fuerza y control del proceso.
- Instalar los elementos de campo, tales como pulsadores, interruptores, luces piloto, sensores y actuadores, entre otros con su respectiva identificación.
- Diseñar la estructura de un brazo estabilizador para una esfera.
- Diseñar un control PID, para el brazo estabilizador de una esfera usando la Raspberry Pi 3 y Arduino Mega como un controlador.
- Diseñar la programación para la adquisición de datos, monitoreo y control para el brazo estabilizador de una esfera.
- Realizar 10 prácticas de programación, configuración, comunicación y monitoreo.
- Diseñar planos eléctricos de fuerza y control del proceso.

MARCO TEÓRICO

3.1 RASPBERRY PI

La placa consiste de una base de 85 x 54 mm, integrado por un chip Broadcom BCM2835 y un procesador de 1 GHz ARM, además de un GPU VideoCore IV y 512 de memoria RAM.

Para su funcionamiento solo se requiere una memoria SD que puede ser desde 8 a 16 GB que previamente debe ser cargado con un sistema operativo de preferencia raspbian, el cual se puede encontrar en la página oficial del producto. Para la alimentación se utiliza un cargador de tipo micro USB, más utilizado para carga de celular y tiene un costo relativamente bajo, dentro de los agregados especiales se puede encontrar también una carcasa comercial que protegerá cada uno de los complementos, además se puede elaborar una carcasa propia que cuente con ventilador e iluminación en caso de querer darle mejor presentación.

Dentro de las ventajas de esta herramienta se encuentra cada uno de los puertos de conexión con los que cuenta, de los cuales se puede destacar al menos un puerto de video HDMI, un mini Jack de audio, un puerto rj45 (Ethernet) y dos puertos USB 2.0 donde se puede conectar tanto un mouse o teclado, muy útil al momento de configurarlo.

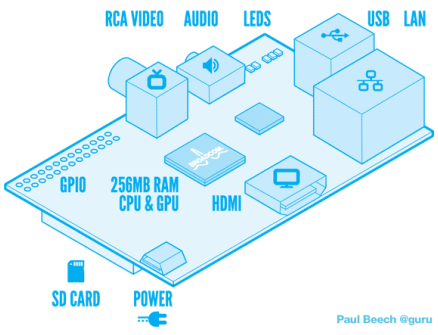


Figura 2 Diagrama RASPBERRY PI
Fuente: Bibliográfica (Adria, 2012)

Para entender mejor en que consiste una Raspberry es importante destacar que la versión 3 tiene las siguientes mejoras:

- Un Procesador ARMv8 de cuatro núcleos con 64 bits a una velocidad de 1,2 GHz.
- Si-Fi 802.11n
- Modulo Bluetooth versión 4.1 con funcionamiento en baja energía

Y mantiene de su versión anterior lo siguiente:

- Memoria RAM a partir de 1GB
- Cuatro puertos USB de alta velocidad.
- Cuarenta pines de entrada y salida.
- Un puerto de alta calidad de video digital (HDMI)
- Un puerto normalizado para la comunicación Ethernet para conector RJ-45.
- Un pin para audio de alta calidad.
- Adaptador para conexión de cámara.
- Conector para pantalla portátil.
- Ranura para Sd que contiene el sistema operativo.
- Tarjeta gráfica IV 3D.

3.1.1 Características Raspberry Pi3

La Raspberry Pi3 es similar a la Raspberry Pi2, debido a que la parte de hardware mantiene mismo tamaño y similar distribución de puertos GPIO, dentro de lo destacable es que la nueva versión varia en su construcción para procesamiento de datos como son la mejora en el procesador con la anexión de un BCM2837 ARMv8 de 64 bits con 1,2GHz de velocidad.

Además, agrega el chip BCM43143 que permite agregar conexión WIFI b/g/n y Bluetooth 4.1 de bajo consumo, también mejora el consumo y distribución de energía, lo cual permite que se pueda manejar varios puertos USB externos al mismo tiempo.

Una de las diferencias claramente definidas con su predecesora es la distribución de los leds, lo que es necesario debido a la ubicación de la nueva antena WIFI, todas las cajas la Raspberry Pi 2 son compatibles con la Raspberry Pi 3.

Dentro de los procesos de diferenciación los creadores de estas tarjetas realizaron un estudio que compara tanto la memoria, el procesamiento de CPU y medición del tiempo de arranque, estos modelos que se compararon se pueden encontrar disponibles en el mercado.

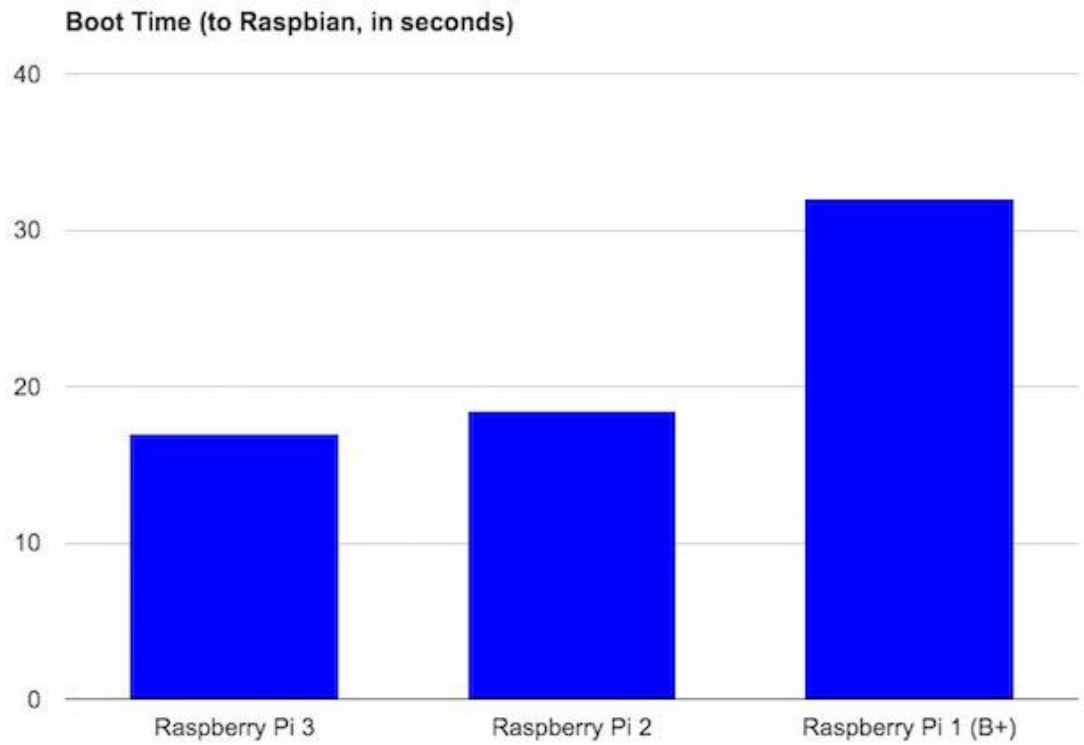


Figura 3. Tiempo de Arranque
Fuente. Bibliográfica (Klosowski, 2016)

En la Figura 3 está claro que las Raspberry Pi 2 y 3 tienen un tiempo de carga casi de la mitad del Raspberry Pi 1, aunque entre la 2 y 3 se puede observar apenas una

diferencia de 1.4 segundos que parece poco, pero en tiempo de funcionamiento y respuesta en tiempo real tiene valores a destacar.

La Raspberry Pi 3 modelo B tomando en cuenta las sugerencias de varios usuarios del producto, incorpora dentro de su misma placa dos funcionalidades, que se creían hasta hace poco, mejoras imposibles de obtener, las cuales son conectividad WIFI y Bluetooth, útiles al momento de ahorrar puertos USB que antes servían para agregar este tipo de comunicaciones.

3.1.2 Comandos básicos Raspberry PI

Como todo sistema operativo que su Kernel se basa en una distribución de Linux es importante conocer los comandos básicos de configuración, por lo que los más importantes se detallan en este apartado.

Lo más importante para configurar como recomendación propia es bueno entrar al modo súper usuario con el comando **sudo su.**

Para la navegación entre carpetas se utiliza el comando **cd** y el directorio al que se quiere acceder, a su vez dentro del directorio se utiliza **ls**.

Para creación de carpetas se utiliza **mkdir**, mientras que para crear archivo existen comandos como: **nano, vim y gedit**.

Para remoción de archivos se utiliza **rm,** para mover archivos **mv** y para copiarlos se utiliza **cp.**

3.1.3 Sistemas operativos compatibles

Raspberry pi tiene un sistema operativo propio debido a que su arquitectura ARM lo que lo vuelve incompatible con ciertos programas, pero si permite el manejo y

construcción de programas en software libre. Dentro de los sistemas operativos soportados son:

- **Raspbian:** Es el sistema operativo basado en Linux por defecto que los desarrolladores crearon para la placa.
- **Ubuntu MATE:** Una distribución que se basa en Ubuntu, pero en forma más ligera para su utilización sin consumo de exceso de recursos de CPU.
- **Kali Linux:** Es un sistema operativo para testear vulnerabilidades en los sistemas operativos.
- **CentOS:** Un sistema operativo diseñado y creado para la implementación de servidores dedicados y en software libre.
- **OpenELEC y OSMC:** Se utiliza para contenido multimedia y permite usar servidores de streaming en tiempo real.
- **RasPBX:** Es un sistema desarrollado para funcionar como central telefónica basado en asterisk.

3.2. Pines GPIO

Son pines predeterminados para la utilización de puertos entrada y salida que permiten al Raspberry interactuar con el mundo exterior, permitiendo elaborar varias prácticas de electrónica, control y monitorización.

3.2.1 Esquemas de los GPIO según el modelo

Según el modelo de Raspberry que ya va en su versión 3 la distribución de los pines de entrada y salida varían de forma no tan drástica, manejan aproximadamente el mismo número de puertos, pero la distribución por software y hardware cambian dependiendo la versión por lo cual es importante conocer cada una de sus versiones:

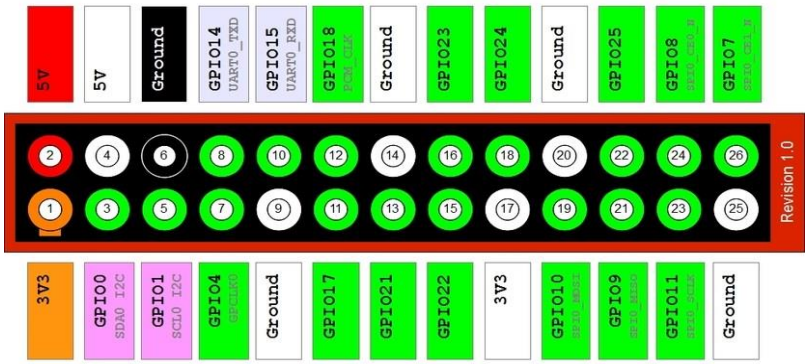


Figura 4. Distribución pines Raspberry Pi 1 Modelo B
Fuente. Bibliográfica (Isaac PE, 2015)

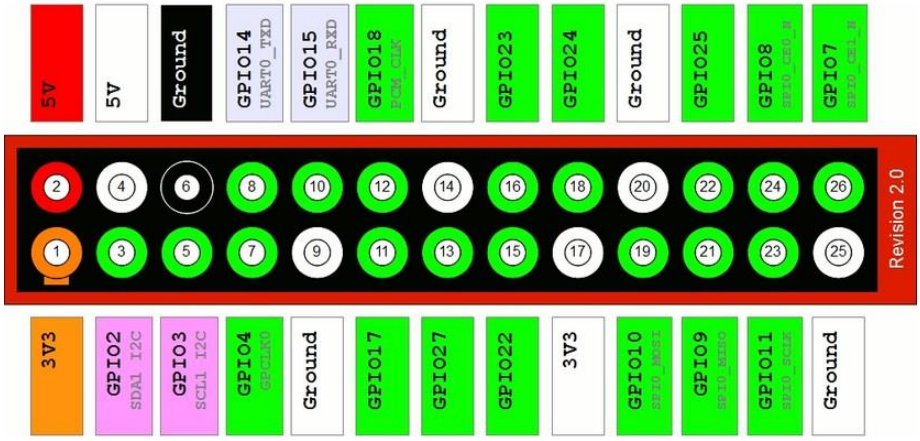


Figura 5. Distribución pines Raspberry Pi 2 Modelo A/B
Fuente. Bibliográfica (Isaac PE, 2015)

Figura 6. Distribución pines Raspberry Pi 2 Versión 2 y 3 Modelo A+ y B+
Fuente. Bibliográfica (Isaac PE, 2015)

Al observar las Figuras 4, Figura 5, Figura 6 se puede comprender que la distribución no varía demasiado, pero es necesario siempre tener en cuenta la versión con la que se está trabajando.

3.3. Conectores

3.3.1. Conector DSI

Es una interfaz con capacidades enormes que permite agregar pantallas de alto rendimiento y bajo consumo además de protecciones necesarias para evitar interferencia electromagnética, además de reducir el número de pines y agregar compatibilidad absoluta con varias marcas.

Este tipo de pantallas se utiliza en los teléfonos modernos además e dispositivos móviles de todo tipo, toda pantalla que necesita economizar espacio y consumo utiliza este tipo de tecnología, facilitando a los proveedores su manejo, ubicación y portabilidad.

Figura 7. Conector DSI
Fuente. Bibliográfica (Vis, 2014)

En la imagen se puede observar un conector DSI el cual se ubica en la porción céntrica de la placa Raspberry y permite la conexión automática de la pantalla portátil de bajo consumo.

3.3.2. Conector CSI

Este tipo de conector es usualmente utilizado para realizar conexiones a una cámara digital que permite capturar imágenes o transmitir video en tiempo real mediante su utilización.

3.3.3. Conector HDMI

Un conector HDMI permite transmisión de imagen y video digital en calidades altas de imagen y procesamiento de información, permite que el Raspberry se pueda conectar a cualquier equipo con monitor y transmitir a través del tanto audio o video.

Los puertos HDMI son los siguientes en base a su tipo:

- A: Más conocido y posee diecinueve pines. (Moreno, 2015)
- B: Posee 29 pines y se enfoca a mejor calidad de video. (Moreno, 2015)
- C: Es la versión mini del tipo A y se usa para dispositivos portátiles de menor tamaño. (Moreno, 2015)
- D: Se puede definir como la versión micro del Tipo A. (Moreno, 2015)

3.4. Puerto Ethernet

Un puerto de este tipo se puede conectar mediante un cable con un RJ-45 y permite que el Raspberry interactúe con otros equipos que se encuentran dentro de la misma red LAN de una institución, además como el acceso a le red global de datos denominada Internet.

3.4.1. Tipos de conexión

- **10BASE-T** por sus siglas se denota el ancho de banda que es 10Mbs
- **100BASE-TX** aumenta la velocidad respecto a su predecesor 100Mbs

- **1000BASE-T** este interfaz es el más utilizado en equipos de usuario final actuales.

La velocidad de las conexiones viene determinada por el interfaz de menor velocidad, es decir si conectamos un equipo con una tarjeta de red de 10Mbs a una red de 100Mbs el equipo solo transmitirá a su máximo permitido, aunque dentro de la red se permitan transmisiones a mayor ancho de banda.

En la actualidad se vienen reemplazando las redes cableadas por redes inalámbricas por su facilidad de instalación y reducción de costos, sin embargo, cuando se requiere mejorar la calidad de transmisión de datos se utiliza cable UTP lo que mejora la calidad y la seguridad en las redes de datos.

3.4.2. Características

- Tecnología que inicio con trasmisiones por medio de cable coaxial.
- Actualmente se utiliza para trenzado
- Próximamente se reemplazará por fibra óptica.
- Se utilizan equipos de red que permiten mejorar la administración de las redes de datos.
- El paquete máximo de trasmisión es de 14000 bytes por segundo.

3.4.3. Resumen de las características físicas de Ethernet.

Tabla 1.
Características físicas de Ethernet

Características	**Valor Ethernet**	**Valores IEEE 802.3**					
		10Base5	**10Base2**	**10Base5**	**10BaseT**	**100BaseTX**	**100BaseT4**

Velocidad de los datos (Mbps)	10	10	10	1	10	100	10
Longitud máxima del segmento (m)	500	500	185	250	100 con pares trenzados no blindados	100 UTP o STP	1800
Soporte	Coaxial de 50-ohmios (grueso)	Coaxial de 50-ohmios (grueso)	Coaxial de 50-ohmios (fino)	Pares trenzados sin blindaje	Pares trenzados sin blindaje	Categoría 5 Pares trenzados blindados o sin blindaje	Coaxial de 75-ohmios
Tipos de conexiones en red posibles	Bus	Bus	Bus	Estrella	Estrella	Estrella	Bus

Fuente. *Bibliográfica (Diaz & Escobar, 2014)*

3.5. Arduino mega

El Arduino Mega es un microcontrolador basado en el microcontrolador atmega 25xx. Tiene cincuenta y cuatro pines conformados por analógicos y digitales para utilización como entrada y salida de señales, 14 pines analógicos, 4 puertos de

comunicación serial, un oscilador de 16 MHz, conector para alimentación de adaptador, y un botón de reinicio.

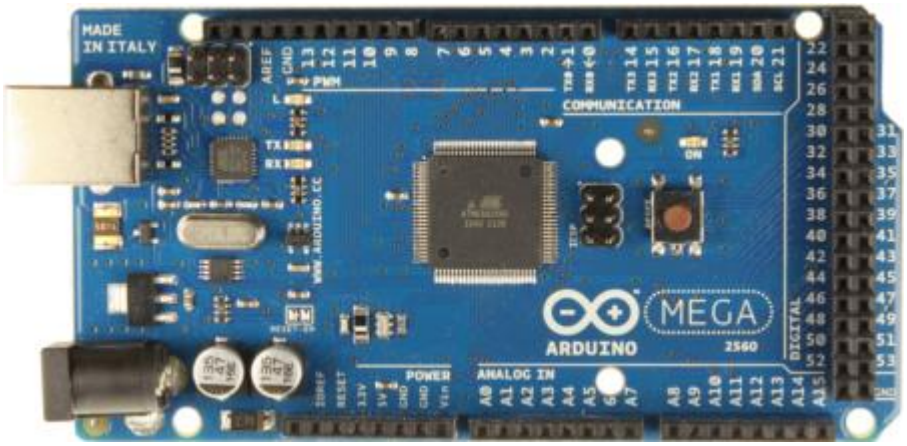


Figura 8. Arduino Mega
Fuente. Bibliográfica (Arduino, 2017)

3.5.1. Poder

Arduino se puede alimentar desde una USB, además por medio de un conector de batería y se puede alimentar por medio de los pines de VCC y GND, que puede tener voltajes entre 5 voltios y 20 voltios.

3.5.2. Entrada y salida

Estos pines se utilizan para manejar sensores y actuadores, además resulta interesante que algunos pines además de su función principal tienen funciones especiales que permiten manejar tantas interrupciones.

3.5.3. Comunicación

Este microcontrolador maneja varios puertos de comunicaciones entre los cuales se encuentran cuatro puertos UART, además se puede simular 2 puertos seriales virtuales. Además, maneja comunicación SPI y comunicación en paralelo gracias a su puerto USB para la carga de archivos.

3.5.4. Principales comandos – Arduino

IF

Para entender este comando se utiliza la pregunta ¿Si se cumple?, lo que significa en caso de cumplirse la condición se ejecuta la orden establecida.

```
if (a == b)
{
coding
}
```

ELSE

Se utiliza para indexarlo con el if, la traducción en orden se viene denotada como caso contrario, por si la orden de if no se ejecuta se realiza la sentencia o sentencias dentro del else.

```
if (a < b)
{
código 1
}
else
{
código 2
}
```

WHILE

La traducción para comprensión de esta orden es mientras, lo que indica que

mientras se cumpla la condición la sentencia se ejecuta, se lo suele utilizar muchas veces para armar el main de un programa.

```
while (a > b)

{
código
}
```

BREAK

Se utiliza para parar cualquier orden dentro de un ciclo o if lo que permite mejorar el procesamiento de información y agilizar la ejecución de órdenes.

```
while (a > b)

{
código
if(a == 5) {
break
}
}
```

FOR

Trabaja como el while pero es completo ya que integra iniciación de una variable que sirve como contador para repetir el ciclo solo las veces requeridas.

```
for (int a = 0; a < 10; a++)

{
código
}
```

SWITCH

Permite hacer varias comparaciones de código de forma rápida ahorrando que el programa pierda tiempo en realizar sentencias por lo cual mejora el procesamiento y disminuye tiempo de ejecución.

switch (variable) {

case 1:

orden 1

break;

case 2:

orden 2

break;

default:

código

}

pinMode:

Es un comando que permite seleccionar si el puerto se lo utilizara como entrada o salida.

pinMode (pin, OUTPUT/INPUT)

digitalWrite:

Escribe un valor en el registro del puerto seleccionado.

digitalWrite (pin, HIGH/LOW)

digitalRead:

Permite leer el valor que se encuentra en el puerto del Arduino.

digitalRead(pin)

analogRead:

Se utiliza para leer las entradas analógicas las cuales sirven para conectar los sensores.

analogRead(pin)

analogReference:

Se puede usar para configurar un rango de voltaje para los valores recibidos en las entradas analógicas. Estos valores se mueven desde 0 a 1024.

3.6. Sensor infrarrojo

El sensor infrarrojo se utiliza en su mayoría para delimitar distancias o como medidores en corte de luz para sistemas de seguridad, a su vez con una configuración estandarizada permite realizar comunicaciones y trabaja en los espectros de luz infrarroja, lo que genera problemas de ruido y estabilidad de los sistemas que se ven minorizados por el uso de protocolos encontrados en la normativa IEEE.

Las distancias del sensor se realizan con la utilización de un cálculo básico de trigonométrica, en donde se tiene el haz de luz y el cruce de la señal con el objeto que se interpone, permitiendo tener dos lados y un ángulo que son material más que suficiente para encontrar la distancia requerida, además el fabricante en su libro técnico establece la curva logarítmica que se puede utilizar también para el mismo propósito.

3.6.1. Sensor de infrarrojo de SHARP

Son, sin duda, uno de los sensores infrarrojos más utilizados en robótica, a destacar el GP2D12:

Figura 9. Conector DSI
Fuente. Bibliográfica (Robótica, 2016)

Este sensor maneja tres pines de los cuales se pueden describir como VCC, GND y el pin analógico el cual puede ser regulado tanto por medio de voltaje de referencia en 5v y el voltaje de 3.3v.

Para limpiar los datos recibidos por el pin analógico se puede utilizar un capacitor electrolítico de 22uF, el problema es que el transmisor genera demasiado ruido por sus características de fabricación.

3.6.2. Sensor infrarrojo SHARP GP2Y0A21YK0F

Es muy útil describir el sensor seleccionado para el presente proyecto el cual permite calcular la distancia en un rango entre 10 y 80 centímetros. Tomando en cuenta también los rangos inestables que se encuentran en sus puntos máximo y mínimo. Se encuentra integrado por un detector de posición, un diodo emisor y un micro circuito que permite procesar las señales para presentarla en valores previamente denotados dentro del cuaderno técnico. (NayLampmechatronics, 2015)

No se ve afectado ni por calor ni voltaje o señales externas debido a su robusto diseño electrónico y su método de triangulación para encontrar las distancias requeridas, además es importante destacar que el sensor envía señales en voltios que se relacionan con la distancia que se encuentra entre el haz de luz y el objeto. Dentro del datashet se puede encontrar la relación voltaje y distancia necesaria para formar la curva logarítmica de posición. (NayLampmechatronics, 2015)

La fórmula para obtener los centímetros necesarios se lo puede realizar de la siguiente forma:

- Centímetros = 27.86 * [voltaje según el número de bits]^-1.15

Figura 10. Sensor infrarrojo SHARP GP2Y0A21YK0F
Fuente. Bibliográfica (NayLampmechatronics, 2015)

Conectores:

- Voltaje de salida
- Tierra (gnd)
- Alimentación que puede ser tres o cinco voltios. (NayLampmechatronics, 2015)

3.7. Motor DC

Son máquinas capaces de convertir la energía eléctrica en mecánica y de forma inversamente proporcional, todo esto debido a las leyes del electromagnetismo.

El funcionamiento de corriente continua se define en base al rechazo que se ejerce en los polos magnéticos de un imán en base a las leyes de Lorentz, el electroimán se llama rotor y su eje le permite girar entre cada uno de los polos.

Este tipo de motores tienen que ser revisados periódicamente ya que por la utilización se generan desgastes en los carbones y a su vez en el bobinado lo que hace que la teoría electromagnética pierda su estabilidad.

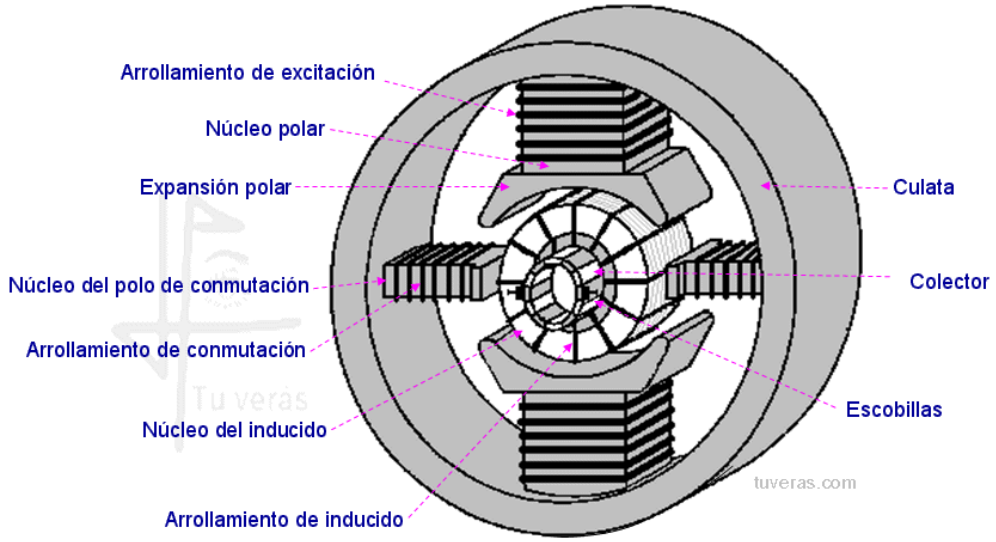


Figura 11. Motor DC
Fuente. Bibliográfica (Pernía, 2011)

3.7.1. Motor con emg49

Es un motor utilizado en varias aplicaciones de robótica por la capacidad y su costo relativamente bajo, dentro de las ventajas encontradas en este tipo de motor es la adición de un encoder de cuatro etapas con dos canales, lo que permite recuperar tanto la velocidad como la dirección del motor en tiempo real.

Por sus características de diseño y construcción permite soportar grandes pesos de entre 5 y 25kg, de lo que se puede recalcar es que el encoder permite recuperar 588 pulsos por cada vuelta, además permite tomar 2000 pulsos por segundo.

Características:

- **Tensión para óptimo funcionamiento:** entre 18 y 24 voltios.
- **Fuerza soportada:** aproximadamente 16 kilogramos por centímetro.
- **Velocidad con carga:** 122 revoluciones por minuto.
- **Corriente necesaria para funcionar:** 2.1 amperios.
- **Velocidad con sistema libre:** 143 revoluciones por minuto.
- **Corriente con sistema libre:** 0.5 amperios.
- **Corriente óptima:** 13 Amperios.

- **Potencia de funcionamiento:** 34,7 Watts.
- **Pulsos del encoder por vuelta:** 588.
- **Diámetro:** 45 milímetros.
- **Conectores:** Tiene un cable de cuatro conectores los cuales integran el encoder con su cable de VCC, GND, Canal A y Canal B, además de esto cuenta con dos cables extra para la alimentación y movimiento del motor.

Figura 12. Motor DC reductor 24V 122 RPM con ENCODER EMG49
Fuente. Bibliográfica (SuperRobotica.com, 2017)

MARCO METODOLÓGICO

4.1. Metodología

Para la parte metodológica de este proyecto se realizará un proceso experimental que parte de un punto propuesto a un escenario practico mediante el cual se procederá a realizar pruebas de funcionamiento de cada una de las prácticas. Las diez prácticas propuestas generan un proceso de retroalimentación entre ellas ocupando los conocimientos previamente adquiridos en su predecesora.

Se define el campo de aplicación del desarrollo de este proyecto es los estudiantes de carreras afines a Electrónica o redes de la información.

Se procede a elaborar cada práctica de forma detallada destacando los objetivos, diagramas, materiales a utilizar, código y programas necesarios para el funcionamiento de cada actividad del módulo didáctico para de esta forma demostrar su funcionalidad.

A partir de este punto se realizará el desarrollo de la metodología con las prácticas, su desarrollo y resultado por medio de evidencia fotográfica que demuestra su funcionamiento.

4.2. Practicas

Dentro de este apartado se procede a la elaboración de 10 prácticas que permiten visualizar la funcionalidad del módulo didáctico y las herramientas que contiene, las actividades a desarrollar son las siguientes:

- Manejo de puerto I/0
- Elaboración de un interfaz web
- Elaboración de base de datos

- Exportación de archivos Excel desde el servidor web
- Elaboración de interfaz de monitoreo grafico
- Elaboración de servidor web
- Identificación del sistema
- Diseño de un control proporcional derivativo (PD)
- Diseño de un control proporcional integral (PI)
- Control proporcional integral derivativo (PID) con monitoreo

4.2.1. Práctica 1

	**UNIVERSIDAD POLITÉCNICA SALESIANA** **FACULTAD DE INGENIERÍA** **SEDE GUAYAQUIL**

CARRERA	PRÁCTICA N^0	NOMBRE DE LA PRÁCTICA
ELECTRÓNICA	1	Manejo de puerto I/0

OBJETIVOS
Utilizar los Puertos GPIO del Raspberry. Comprender la funcionalidad del lenguaje Python, y de esta manera generar un programa que permita el manejo de puertos de GPIO integrados y el enlace con la placa Arduino Mega. Desarrollar las prácticas mediante el uso de la pantalla Touch que está conectada con el Raspberry Pi 3, además de un teclado inalámbrico.

EQUIPOS A UTILIZAR	
• Módulo Didáctico	• Pantalla LCD

• Cable de poder del Módulo didáctico • Router Inalámbrico • Cable de red	• Teclado Inalámbrico • Módulo didáctico

INSTRUCCIONES
1) Conectar todos los módulos necesarios para la elaboración de la práctica 2) Utilizar la pantalla touch para poder manipular las herramientas necesarias para la práctica 3) Verificar las conexiones de los cables de conexión entre cada módulo 4) Comprobar funcionamiento de los módulos 5) Elaborar un programa utilizando Python y C++ que permita manipular los puertos salida del Raspberry, además del Arduino 6) Elaborar un programa en Python que permita manejar las luces o salidas por medio de las entradas o botones ubicados en el módulo 7) Integrar una alerta que permita verificar si la práctica está en proceso o ha terminado

DIAGRAMAS

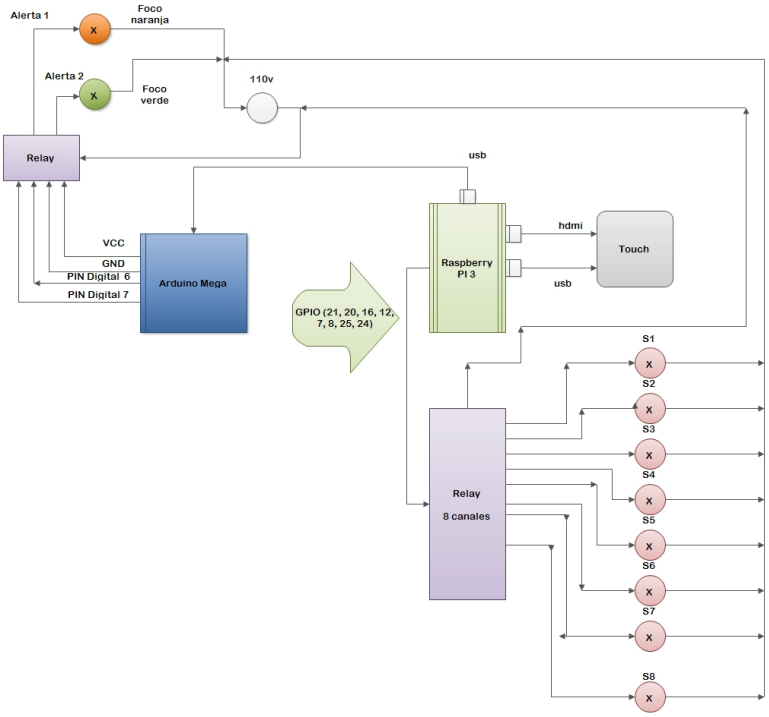


Figura 13. Diagrama Luces intermitentes
Fuente: Realizado por el autor

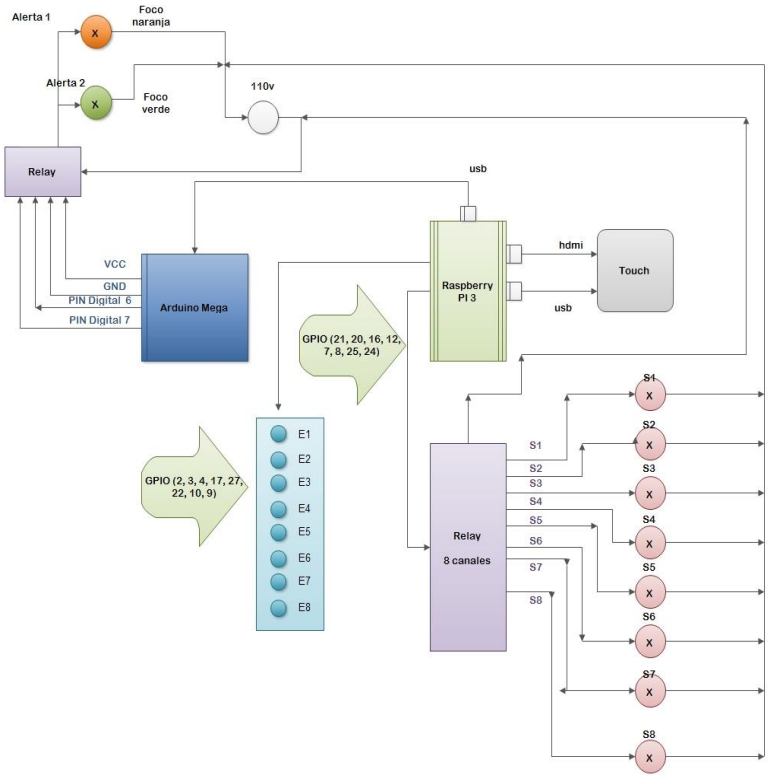


Figura 14. Diagrama puertos entrada/salida
Fuente: Realizado por el autor

PROCESO

1. Primero se verifica las conexiones necesarias para el funcionamiento del módulo didáctico las cuales son:

    a. Conexión del cable HDMI hacia la pantalla.

Figura 15. Conector HDMI en pantalla LCD touch
Fuente: Realizado por el autor

Figura 16. Conector HDMI en pantalla Raspberry Pi 3
Fuente: Realizado por el autor

b. Conexión del conector USB a mini USB de la pantalla Touch

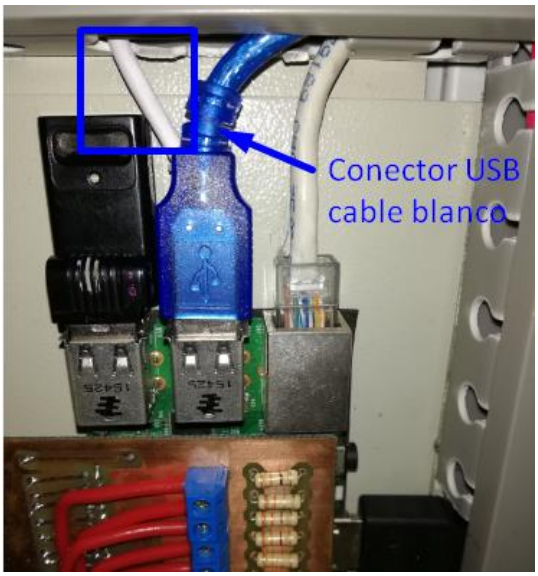


Figura 17. Conector USB en Raspberry Pi 3
Fuente: Realizado por el autor

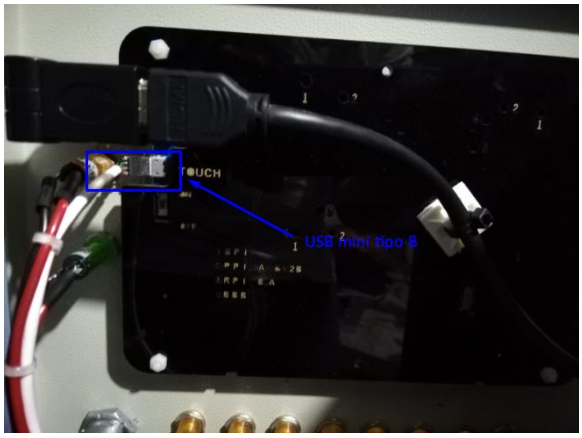


Figura 18. Conector USB mini tipo B en la pantalla LCD touch
Fuente: Realizado por el autor

c. Conexión por conector USB a USB tipo B hacia el Arduino Mega

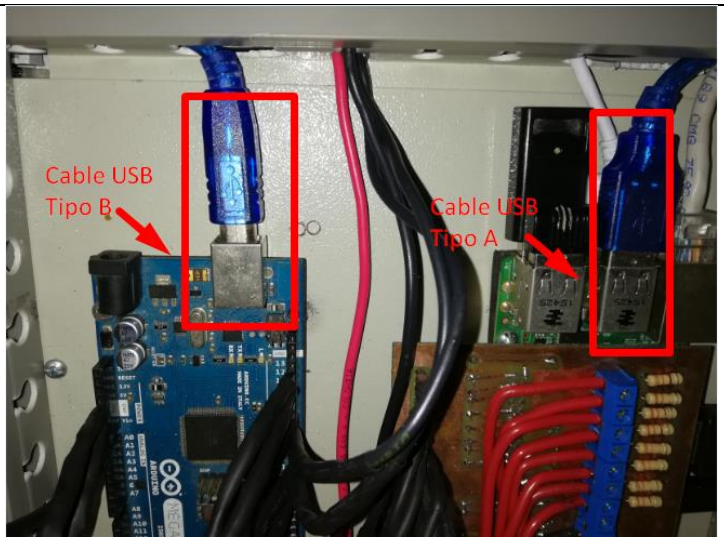


Figura 19. Conector USB tipo A y tipo B desde Arduino hacia Raspberry Pi .
Fuente: Realizado por el autor

Como se visualiza en las Figuras posteriores es necesario que los cables de comunicación estén correctamente conectados.

2. Verificar el funcionamiento de la pantalla LCD touch, para lo cual se realiza lo siguiente:

    a. Verificar que la pantalla se encuentre encendida, revisando en su parte posterior el botón de encendido se encuentre en la posición de ON.

Figura 20. Parte posterior LCD touch interruptor de encendido/apagado
Fuente: Realizado por el autor

    b. Por medio de la utilización de un dedo en la pantalla táctil se verifica

mediante una prueba fácil que es arrastre de algún icono del escritorio

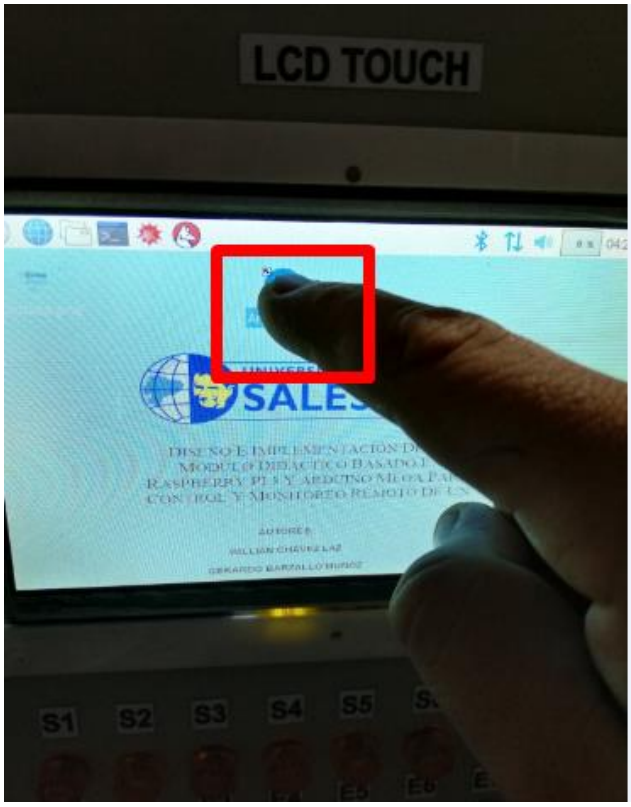


Figura 21. Arrastre táctil de un icono en pantalla LCD touch
Fuente: Realizado por el autor

Realizado el proceso de verificación del funcionamiento de las herramientas necesarias para la práctica actual se procede a realizar las siguientes tareas, las cuales consisten en realizar un manejo de los puertos GPIO, y además verificar el proceso por medio de Arduino.

ARDUINO

3. Ingresar por medio de la LCD touch al Arduino Idle que se encuentra ubicación en Inicio, Electronics, Arduino Idle como podemos observar en la Figura 22.

Figura 22. Abrir Arduino IDLE desde pantalla LCD touch
Fuente: Realizado por el autor

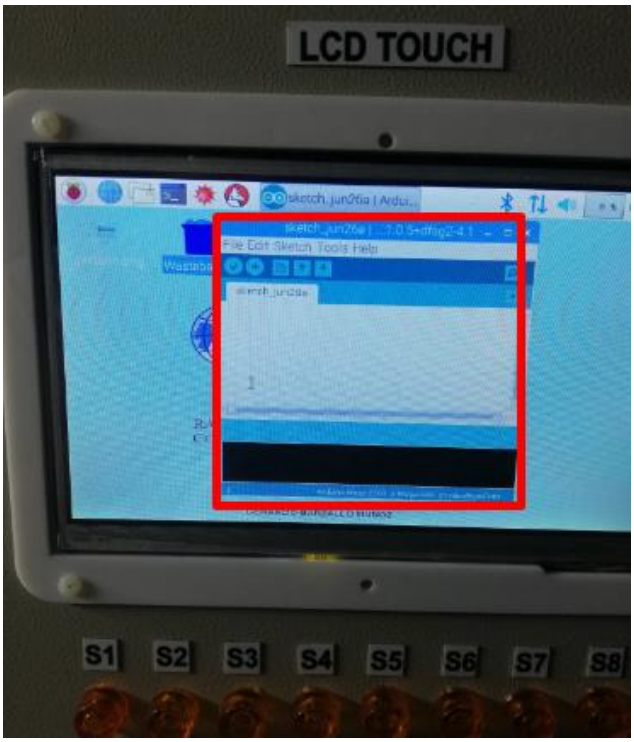


Figura 23. Arduino IDLE
Fuente: Realizado por el autor

Dentro del interfaz de Arduino es necesario cargar un código que permita monitorizar el funcionamiento de la práctica, el código se encuentra en el Anexo 7.

Figura 24. Arduino IDLE con el código para la práctica 1
Fuente: Realizado por el autor

Como podemos observar en la figura 24, el panel de alerta se mantiene apagado, mientras el código no se ha cargado, dentro de esto se tiene que el foco naranja funcionará mientras no empiece la práctica de GPIO y se detendrá el momento que empiece a funcionar, para accionar el foco verde.

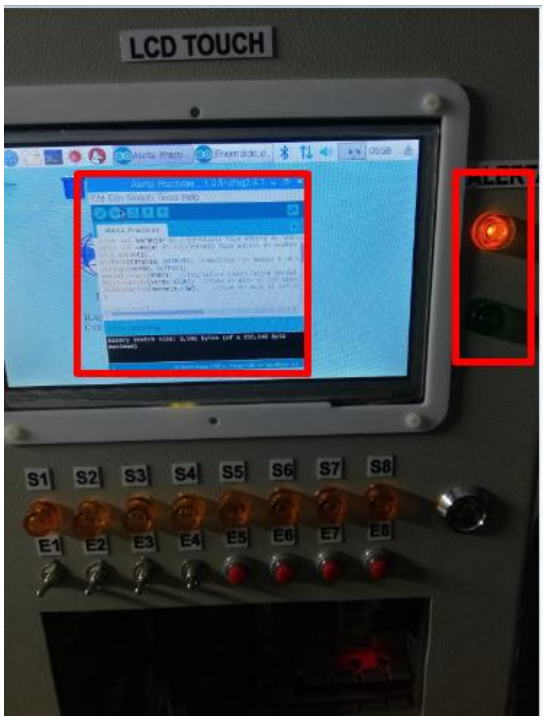


Figura 25. Código cargado en Arduino Mega práctica no iniciada
Fuente: Realizado por el autor

RASPBERRY

Después es necesario agregar el código para manejo de puertos de salida en raspberry Pi 3 que permita manejar cada uno de los focos, para lo cual se procede a realizar la comprobación de 2 escenarios.

4. **Luces intermitentes:** que demuestren la funcionalidad de los puertos de salida.

    a. Se procede a abrir el terminal en el raspberry Pi 3 desde la LCD touch, como se puede observar en la figura haciendo click en la barra superior, en el icono de terminal.

Figura 26. Terminal en interfaz Raspberry Pi 3
Fuente: Realizado por el autor

    b. Abierto el terminal, por medio del teclado es necesario ingresar como super usuario mediante la utilización del comando **sudo su**, se procede a la creación de un programa python que ejecutara ordenes que permitirán realizar la prueba de programa de encendido de todas las luces intermitentes, por medio de la utilización del comando nano

salida.py.

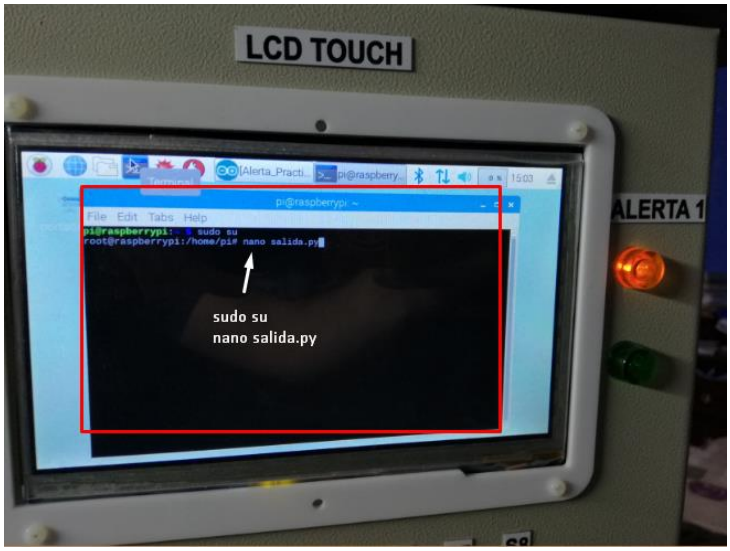


Figura 27. Creando un programa para control salida intermitente
Fuente: Realizado por el autor

c. Dentro del archivo salida.py es necesario agregar el código que se encuentra en el Anexo 8 para el manejo de luces intermitentes.

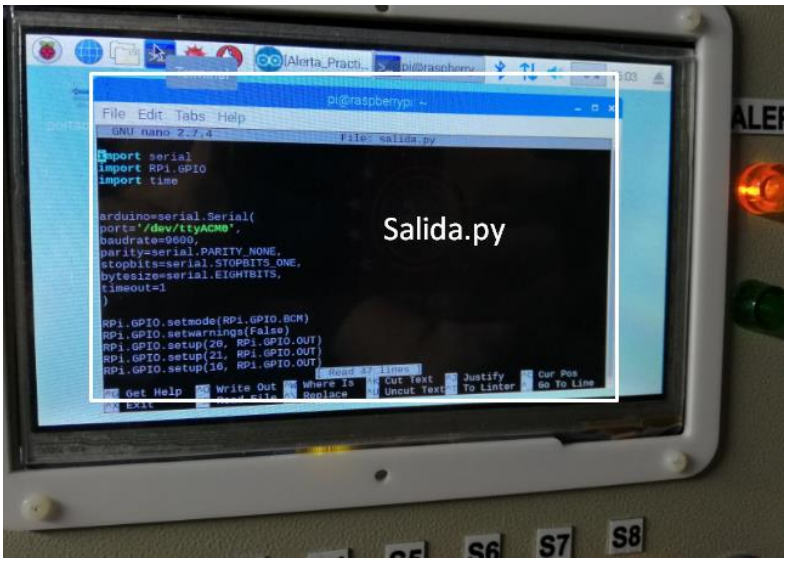


Figura 28. Programa salida.py
Fuente: Realizado por el autor

d. Para ejecutar el programa simplemente se ingresa dentro del terminal el comando **python3 <nombre del archivo>,** en este caso **python 3 salida.py.**

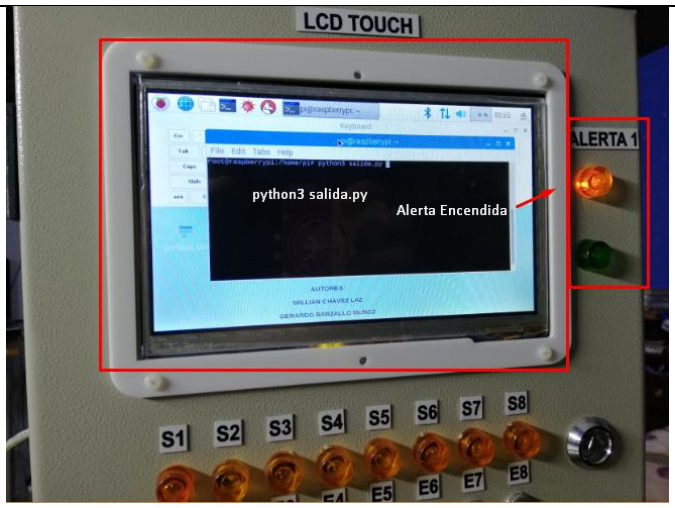


Figura 29. Programa salida.py
Fuente: Realizado por el autor

Al momento de ejecutar el programa con el botón de enter en el teclado se puede ver como parpadean los leds y la alerta del foco cambia a verde.

Figura 30. Programa salida.py luces encendidas
Fuente: Realizado por el autor

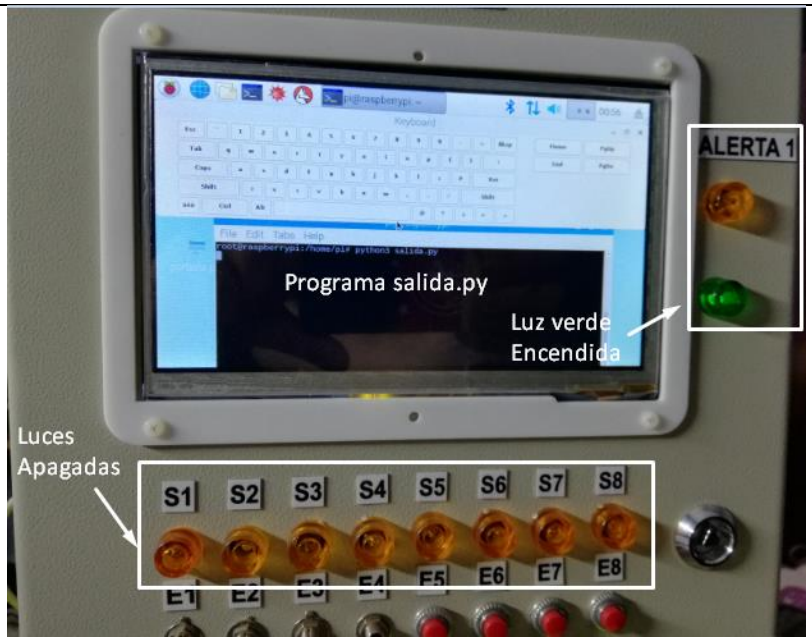


Figura 31. Programa salida.py luces apagadas
Fuente: Realizado por el autor

Al momento que se detiene el programa se visualiza como la alerta enciende el foco naranja, y la salida del terminal.

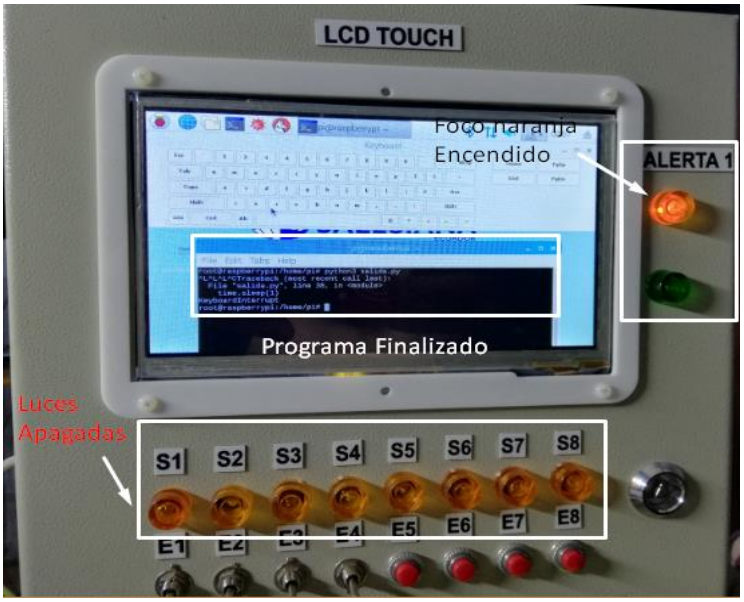


Figura 32. Programa finalizado luz alerta naranja encendida
Fuente: Realizado por el autor

Mientras el programa funcione la luz verde de alerta se enciende, cuando no esta ejecutando el codigo, se enciende la luz naranja para indicar que no esta funcionando.

5. **Puertos GPIO:** Mediante este programa se busca interactuar entre los

botones como puerto de entrada y las luces como interfaces de salida.

a. Es necesario crear un archivo de nombre entrada_salida utilizando el procedimiento realizado en el paso 4 y sus anexos, de los cuales es necesario agregar el código que se encuentra en el Anexo 9.

Figura 33. Programa entrada_salida.py
Fuente: Realizado por el autor

b. Cargado el código se lo ejecuta para ver su funcionamiento por medio del comando **python3 entrada_salida.py**

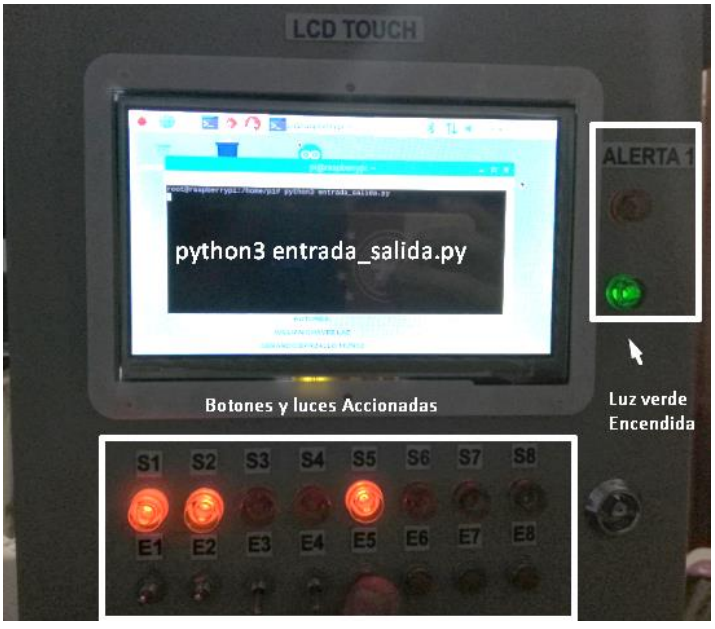


Figura 34. Programa entrada_salida.py ejecutandose
Fuente: Realizado por el autor

Como se visualiza en la Figura 34, las luces que se encuentran encendidas fueron accionados por los switch en posicion de prendido, y el botón pulsado tambien el resto como estan apagados o no pulsados se mantienen apagados, ademas la luz verde de alerta de funcionamiento esta activa mientras el programa funcione, caso contrario se puede observar en la Figura 35.

Figura 35. Programa entrada_salida.py finalizado
Fuente: Realizado por el autor

Para finalizar los programas ejecutados por medio de python se utiliza el ctrl+c.

4.2.2. Práctica 2

	UNIVERSIDAD POLITÉCNICA SALESIANA **FACULTAD DE INGENIERÍA** **SEDE GUAYAQUIL**

CARRERA	PRÁCTICA N^0	NOMBRE DE LA PRÁCTICA
ELECTRÓNICA	2	Elaboración de un interfaz web

OBJETIVOS
• Utilizar Raspberry Pi 3 para la implementación de un servidor web • Comprobar la funcionalidad del servidor web • Interconectar los interfaces web mediante HTML, php y Python • Realizar manejo de puertos de salida por medio del interfaz web

EQUIPOS A UTILIZAR	
• Cable de poder del Módulo Didactico • Router Inalámbrico • Cable de red	• Pantalla LCD • Teclado Inalámbrico • Módulo didáctico • Arduino Mega

INSTRUCCIONES
1) Utilizar la pantalla touch para poder manipular las herramientas necesarias para la práctica 2) Comprobar las conexiones de los cables de conexión entre cada módulo 3) Comprobar funcionamiento de los módulos necesarios para la práctica 4) Configurar cada uno de los ficheros esenciales para la realización de la práctica. 5) Desarrollar un programa que permita interactuar al usuario de forma directa y remota con el interfaz web almacenado dentro de la Raspberry pi 3. 6) Visualizar de forma directa el funcionamiento del módulo que integra tanto el Raspberry Pi 3 y Arduino Mega. 7) Interconectar varios lenguajes de programación para llegar a un objetivo deseado.

DIAGRAMAS

Diagrama conexión de pines Raspberry Pi y Relay

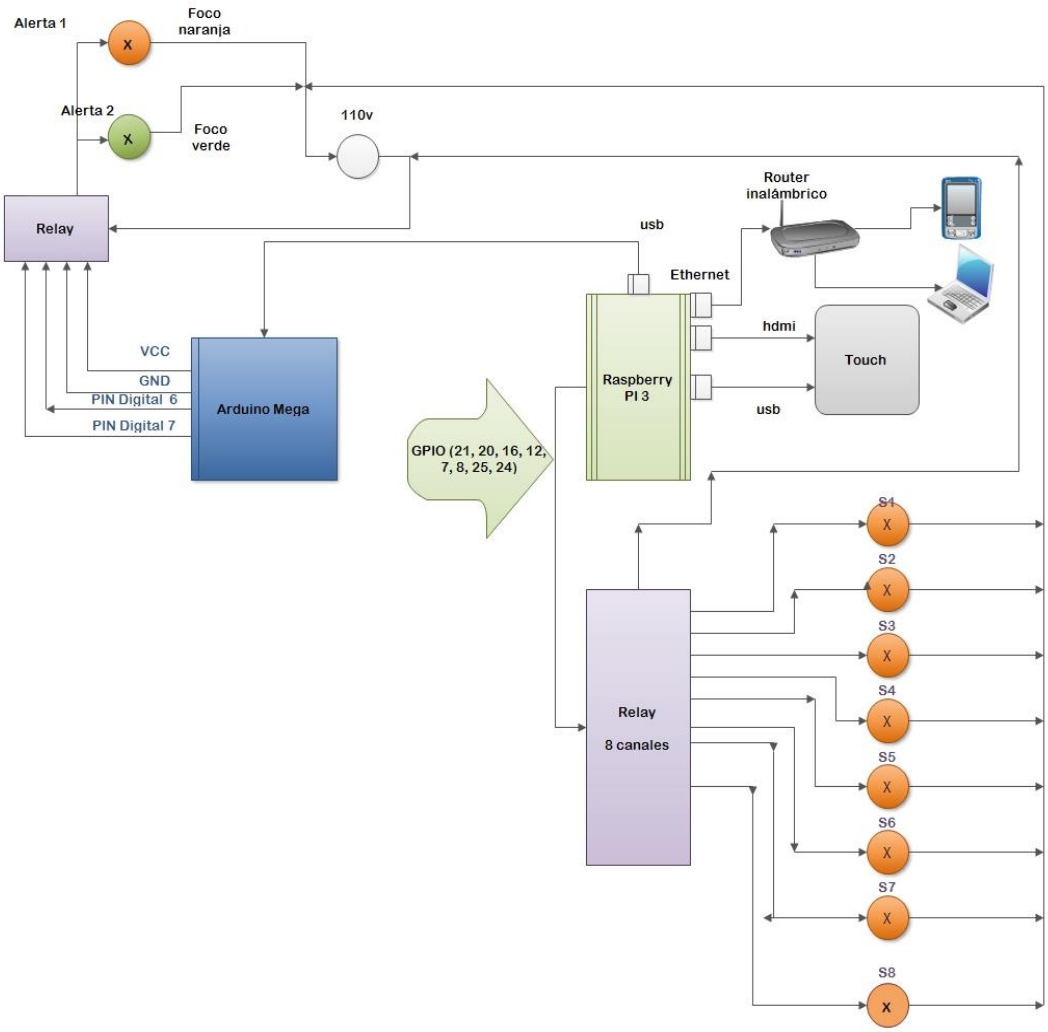


Figura 36. Diagrama conexión página Web
Fuente: Realizado por el autor

PROCESO

1. Se realiza el proceso de verificación como se puede observar en la práctica.1.
2. Se procede a instalar la paquetería necesaria para el manejo de un interfaz web como se puede observar en el Anexo.
3. Una vez concluido los procesos anteriores es necesario por medio de

comandos ingresar en la ubicación /var/www/HTML, es donde se almacenará nuestra página web, para esto se ingresa al terminal desde la pantalla LCD y se utiliza el siguiente comando cd /var/www/HTML mediante la utilización del teclado USB.

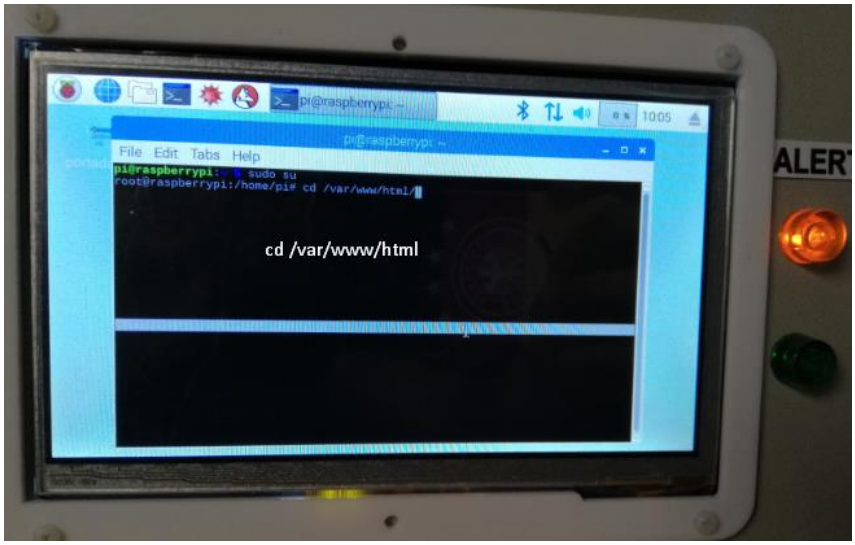


Figura 37. Terminal acceso a ruta html
Fuente: Realizado por el autor

Como se visualiza una vez ejecutada la orden por medio de la tecla Enter en el teclado se accede al fichero.

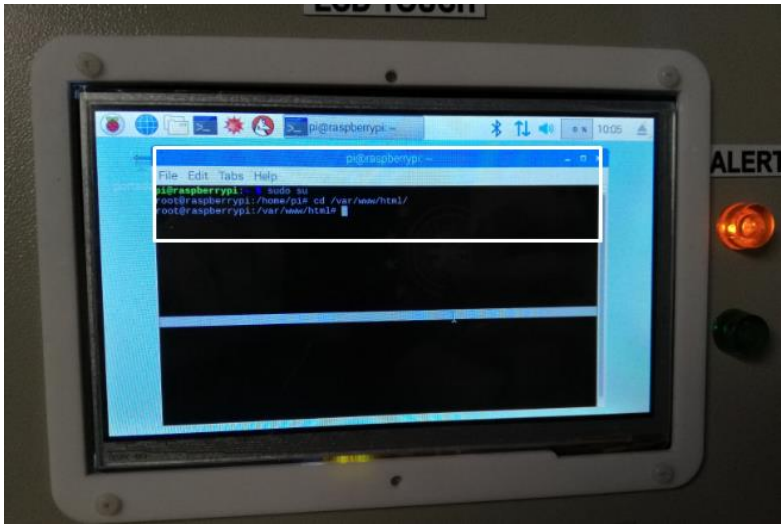


Figura 38.Terminal dentro de la carpeta html
Fuente: Realizado por el autor

4. Dentro de la carpeta es necesario ubicar la imagen que será la portada, para

por medio de código HTML generar el index y anexarlo en el interior de la página web. Para este caso se diseñó una portada que se puede ver en la Figura 39.

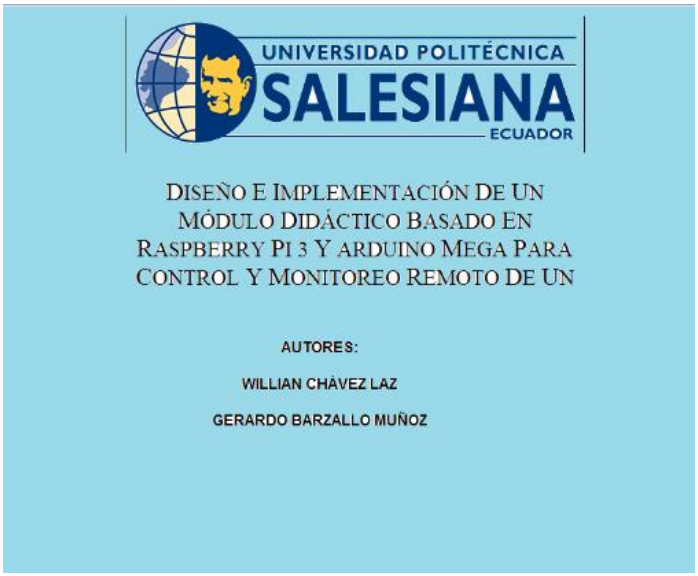


Figura 39. Portada pagina web
Fuente: Realizado por el autor

5. Una vez ubicada dentro de la carpeta HTML la portada como se puede ver abriendo el navegador de archivos dentro de Raspberry, ubicado en la parte superior y dirigiendo mediante el uso de la parte touch hacia la ruta especificada como se ve en la figura 40 está ubicado el archivo portada.png

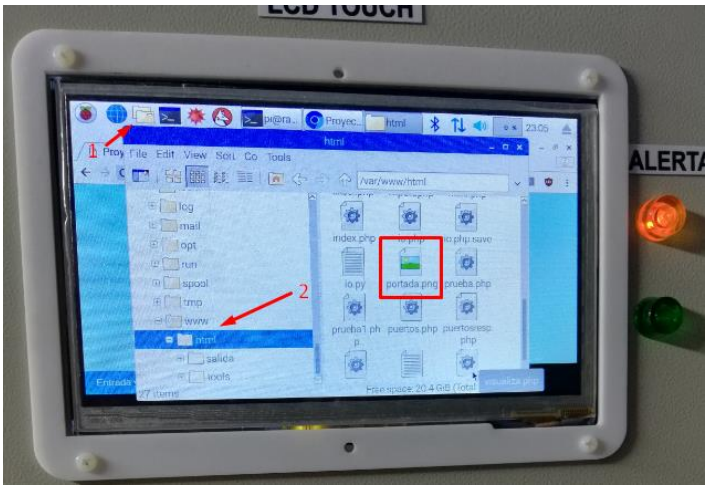


Figura 40. Carpeta html archivo portada.png
Fuente: Realizado por el autor

6. Ubicado el archivo .png ubicado en la carpeta php, se procede a realizar un

archivo index, dentro del terminal se escribe **nano index.php** para tener el archivo de origen que enlazara el resto de funciones y actividades.

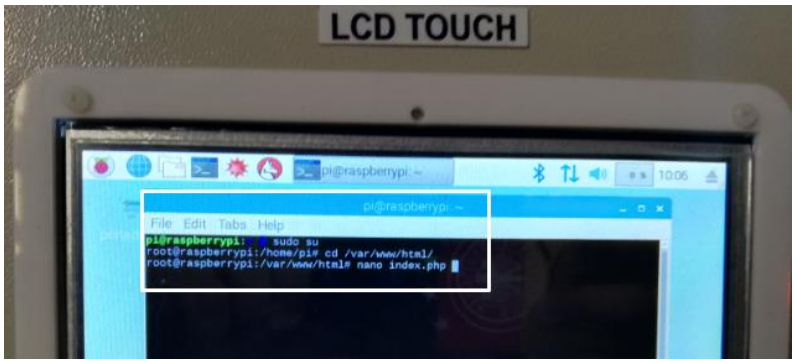


Figura 41. Terminal creación de index.php
Fuente: Realizado por el autor

7. Creado se agrega el código que se encuentra dentro del anexo 10 A para generar la portada.

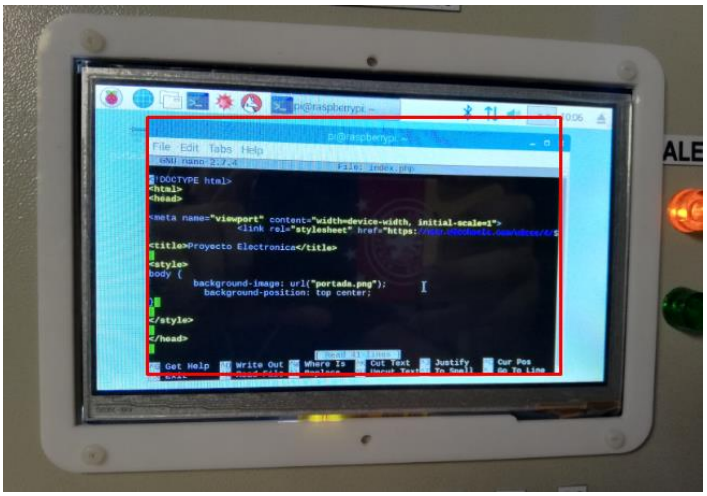


Figura 42. Terminal creación de index.php
Fuente: Realizado por el autor

Para finalizar se utiliza los comandos: **control + o** para guardar y **control + x** para salir.

8. Se ingresa en el terminal el comando **ifconfig** para saber cuál es la dirección IP asignada.

Figura 43. Terminal comando ifconfig
Fuente: Realizado por el autor

Como se visualiza en la respuesta del comando se encuentra la dirección ip asignada al equipo Raspberry Pi 3.

9. Se accede a cualquier navegador que este enlazado a la red generada por el Access point denominada Módulo Didáctico, desde cualquier dispositivo conectado a la red se abre un navegador y se escribe en la barra de navegación la dirección IP obtenida en el paso 8 del procedimiento.

Figura 44. Red inalámbrica del módulo didáctico
Fuente: Realizado por el autor

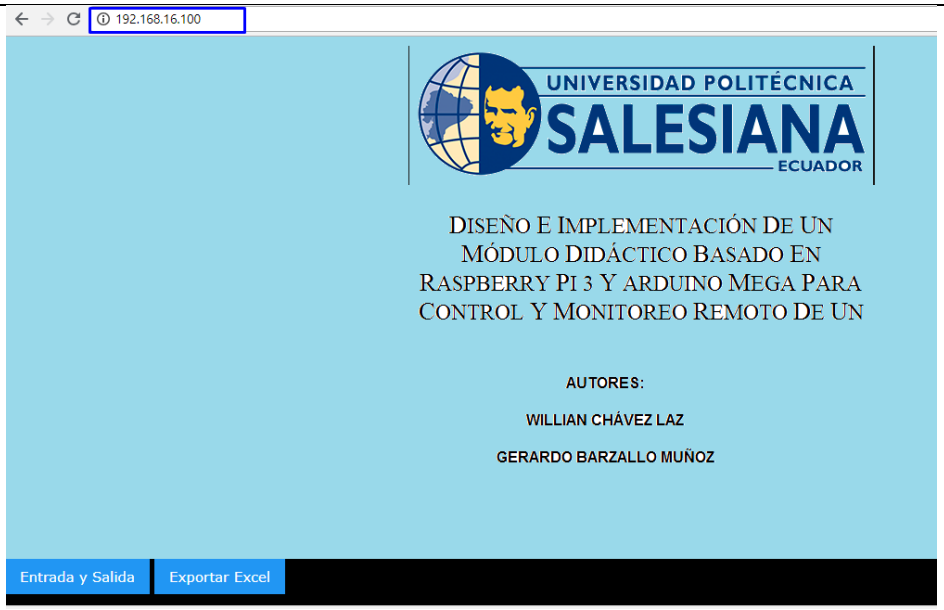


Figura 45. Página Web
Fuente: Realizado por el autor

Tal como se encuentra en la Figura 45 la página web de enlace ha sido satisfactoriamente creada, el siguiente paso es realizar una interfaz web que interactúe con el usuario final y el módulo didáctico controlando las salidas del Raspberry Pi 3.

10. Para que funcione es necesario crear un nuevo archivo que permita visualizar los botones para accionar las salidas del Raspberry, para esto se crea dentro de la carpeta HTML un archivo php con el nombre que se prefiera sin olvidar que para anexarlo al botón de la portada hay que tenerlo presente, para el caso el archivo se denominara puertos.php, se utiliza el comando nano puertos.php y el código se encuentra en el anexo 10 B.

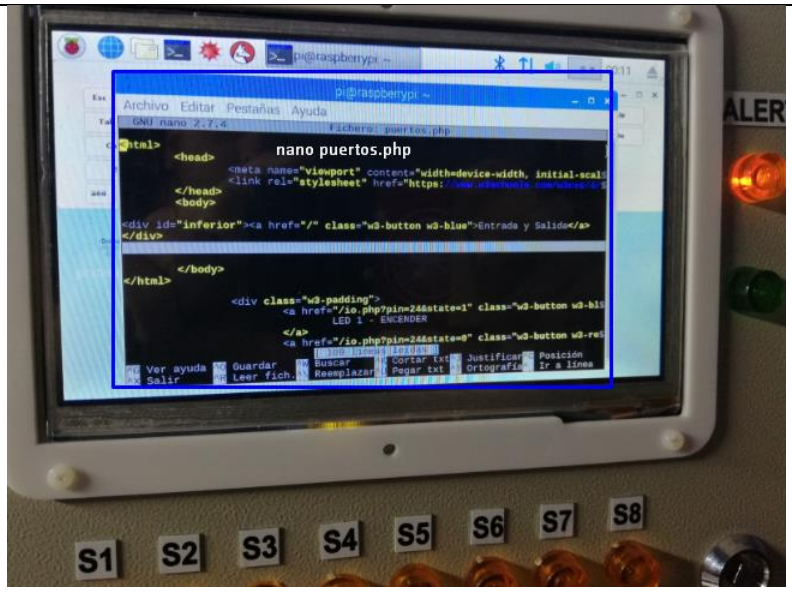


Figura 46. Puerto.php en terminal
Fuente: Realizado por el autor

Cargado el código es necesario enlazar por medio de código php sentencias que permitan ejecutar programas en Python para control de las luces del panel inferior a la pantalla LCD touch, estos programas se encuentran en el Anexo 11A y 11B con nombres preestablecidos como io.php e io.py respectivamente, donde el primero ejecuta el llamado y asignación de puerto por botón de la página web.

11. Para crear el archivo io.php se utiliza el comando nano io.php.

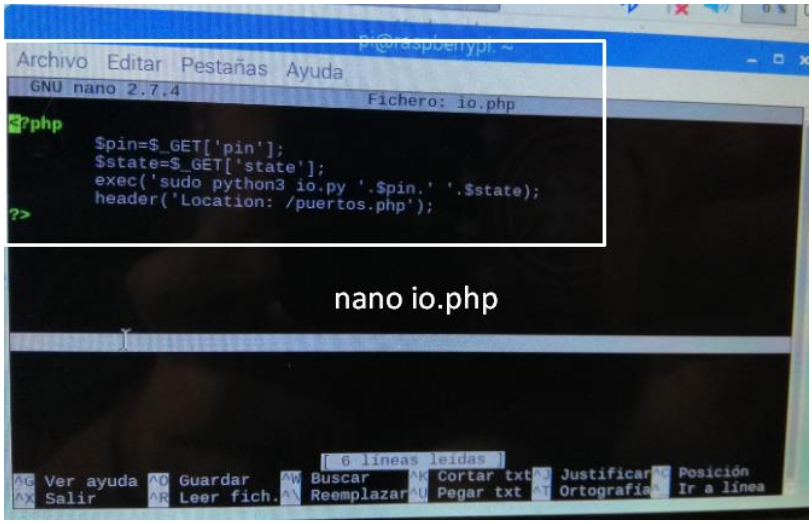


Figura 47. Nano io.php en terminal
Fuente: Realizado por el autor

Este código se encarga de enlazar la página web con el programa en Python para

manejo de puertos de salida, utilizando el interfaz gráfico que en este caso es un portal web.

12. Es necesario realizar un programa denominado io.py que ejecutara las órdenes hacia los puertos GPIO de salida del Raspberry Pi 3.

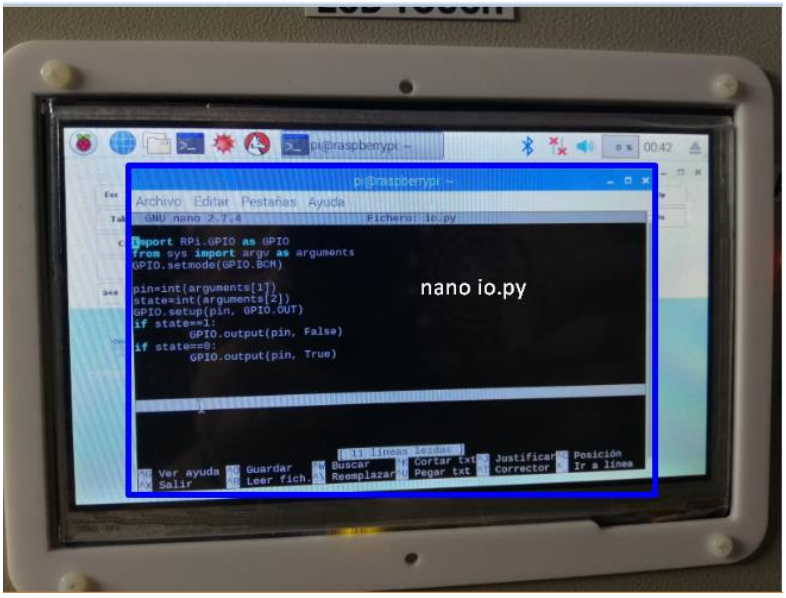


Figura 48. Nano io.py mediante terminal
Fuente: Realizado por el autor

Como se visualiza en la figura 48, el programa lo único que hace es recibir el dato desde el interfaz web con el número de puerto y si lo ubica en 0 lógico o en 1 lógico.

Para comprobar el funcionamiento como se menciona en los pasos 8 y 9 de la práctica 2, y hacer clic sobre el botón que tiene como nombre entrada y salida.

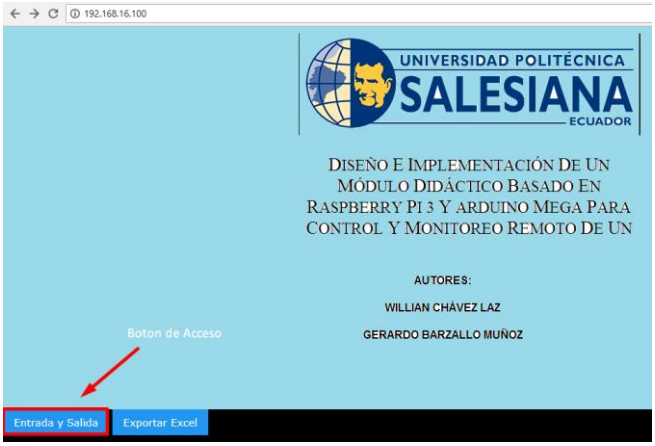


Figura 49. Página web y selección de botón entrada y salida
Fuente: Realizado por el autor

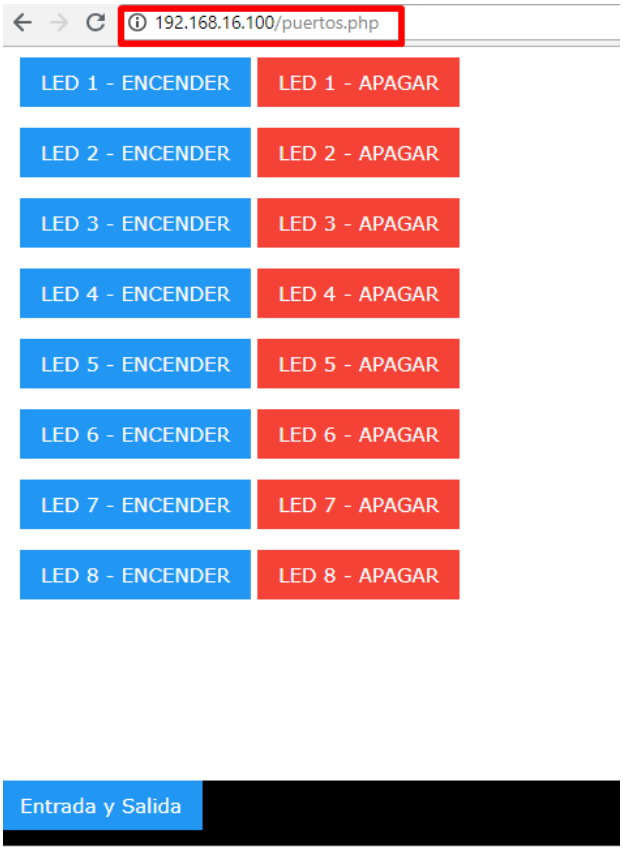


Figura 50. Interfaz puertos.php manejo de salidas
Fuente: Realizado por el autor

Como se visualiza en la Figura 50 y Figura 51, se puede identificar que para cada salida o foco existe un botón que enciende o apaga alguna de los focos ubicados en el tablero.

Para comprobarlo se procede a encender todos los botones de módulo por medio del interfaz Web.

Una vez realizada es necesario apagar todos los botones y comprobar que funciona el botón de apagado.

Figura 51. Luces encendidas por medio de interfaz web
Fuente: Realizado por el autor

Figura 52. Luces apagadas por medio de interfaz web
Fuente: Realizado por el autor

Con esto se puede dar por finalizada la práctica número 2 aplicando todo lo requerido.

4.2.3. Práctica 3

	UNIVERSIDAD POLÍTECNICA SALESIANA **FACULTAD DE INGENIERÍA** **SEDE GUAYAQUIL**

CARRERA	PRÁCTICA N^0	NOMBRE DE LA PRÁCTICA
ELECTRÓNICA	3	Elaboración de base de datos

OBJETIVOS
• Instalar la paquetería que permita utilizar una base de datos • Desarrollar las funcionalidades de MySQL o MariaDB para la realización de una base de datos. • Utilizar las herramientas ofrecidas por el raspberry pi 3 para almacenar los valores de los puertos GPIO en una base de datos mediante la utilizacion de python. • Visualizar los resultados recopilados en la base de datos.

EQUIPOS A UTILIZAR	
• Raspberry Pi 3	• Salidas digitales

• Teclado USB Inalámbrico • Pantalla LCD touch	• Relay de 8 canales • Entradas Digitales

INSTRUCCIONES
1) Instalar en Raspberry Pi la paquetería necesaria para el uso de una base de datos mediante MariaDB. 2) Configurar un programa que permita generar una base de datos y almacenar los valores de los puertos GPIO 3) Visualizar la base de datos por medio de la consola y por medio de un programa en python de lectura

PROCEDIMIENTO
1. Para la elaboración de la práctica se empieza por la instalación de la paquetería necesaria para utilizar una base de datos la cual solo se realiza por una sola vez, que además se encuentra en el manual de usuario del proyecto realizado. a. Primero se instala la paquetería necesaria accediendo al terminal y utilizando **sudo (super usuario) apt-get (tomar de los repositorios) install (instalar el software siguiente) mysql-server**

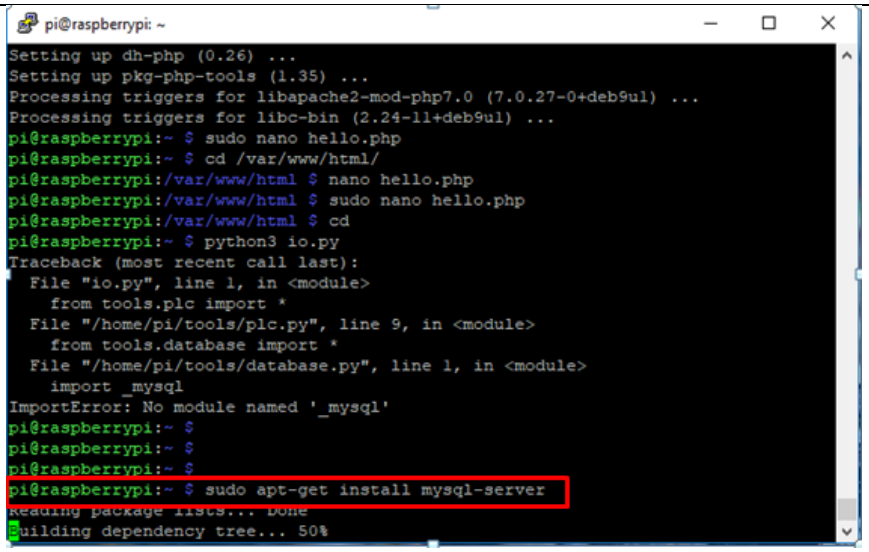


Figura 53. Instalación paquetería mysql
Fuente: Realizado por el autor

b. Dentro del proceso de instalación se solicita una contraseña la cual por defecto para todos las paqueterías es la misma que en root **"plcpi2018"**

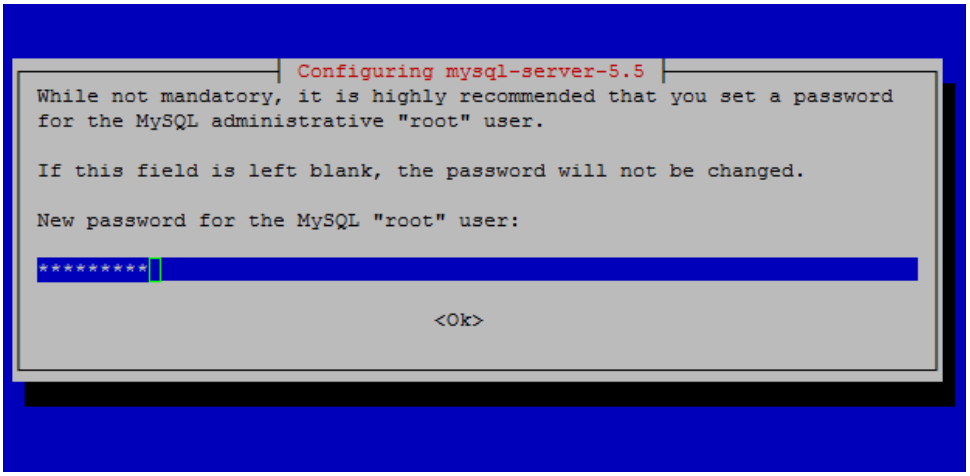


Figura 54. Contraseña mysql
Fuente: Realizado por el autor

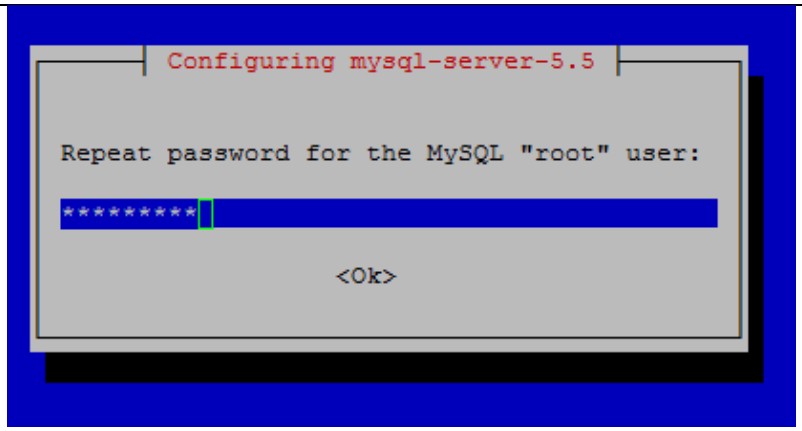


Figura 55. Confirmación contraseña
Fuente: Realizado por el autor

Finalizado el proceso no es necesario realizarlo otra vez, solo en caso de perder información o realizar un respaldo de información.

2. Adicional para el control de bases de datos es necesario instalar la paqueteria que lo permita.

    a. Para instalar las dependencias el procedimiento es similar, solo se utiliza el comando **sudo apt-get install python3-mysqldb.**

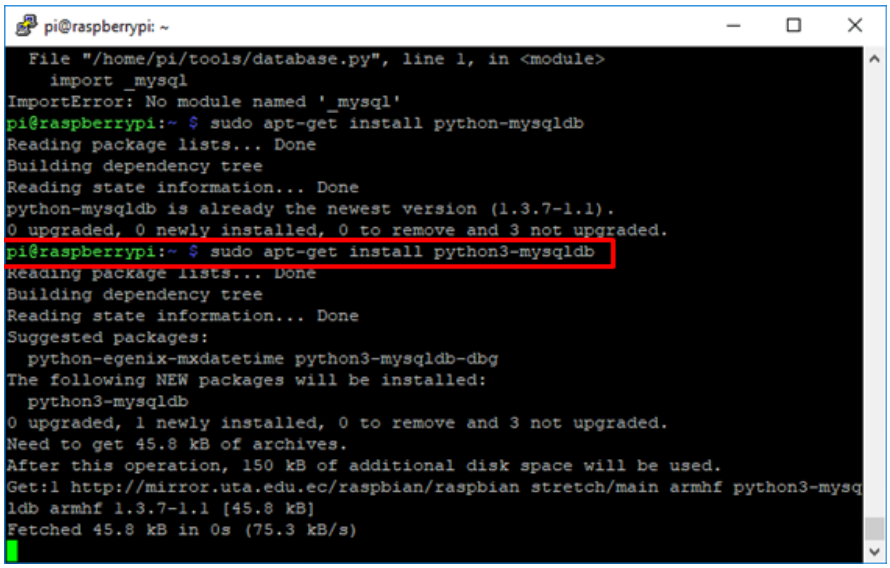


Figura 56. Instalación de paquetes para manejo de mysql con python
Fuente: Realizado por el autor

3. Una vez instalado lo necesario para el manejo de base de datos lo que procede es a realizar la base de datos para lo cual a traves de la terminal se realiza lo siguiente:

a. Se introduce el comando **sudo mysql –u root –p**

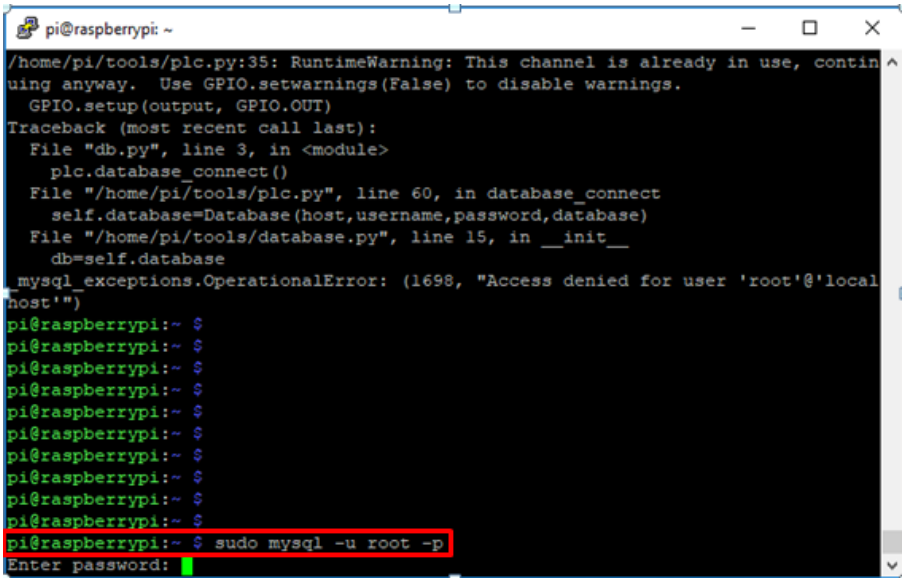


Figura 57. Acceso al gestor mysql para manipulación de base de datos
Fuente: Realizado por el autor

Como se observa el programa solicita una contraseña la cual se agrego en el paso 1 seccion b.

b. Se procede a la creación de una base de datos utilizando los siguientes comandos.

CREATE DATABASE puertos;

CREATE USER 'puertos'@'localhost' IDENTIFIED BY 'plcpi2018';

GRANT ALL PRIVILEGES ON puertos . * TO 'puertos'@'localhost';

FLUSH PRIVILEGES;

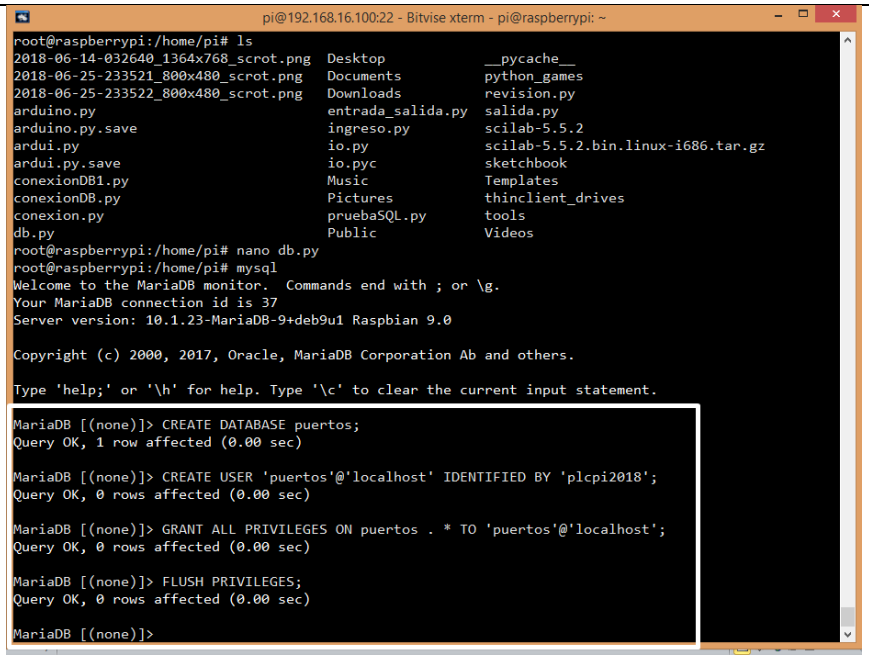


Figura 58. Creación de base de datos
Fuente: Realizado por el autor

Cuando la base de datos se creó exitosamente se tiene una respuesta de **Query OK**, como se puede observar en la figura 58.

c. Para el control de la base de datos por medio de python se creará un programa de extension .py, lo cual se vino explicando como realizar en las prácticas posteriores, y se cargará el código que se encuentra dentro del Anexo 12.

   Como breve recordario el comando a utilizar es **sudo nano BaseD.py** para dejar un nombre predeterminado.

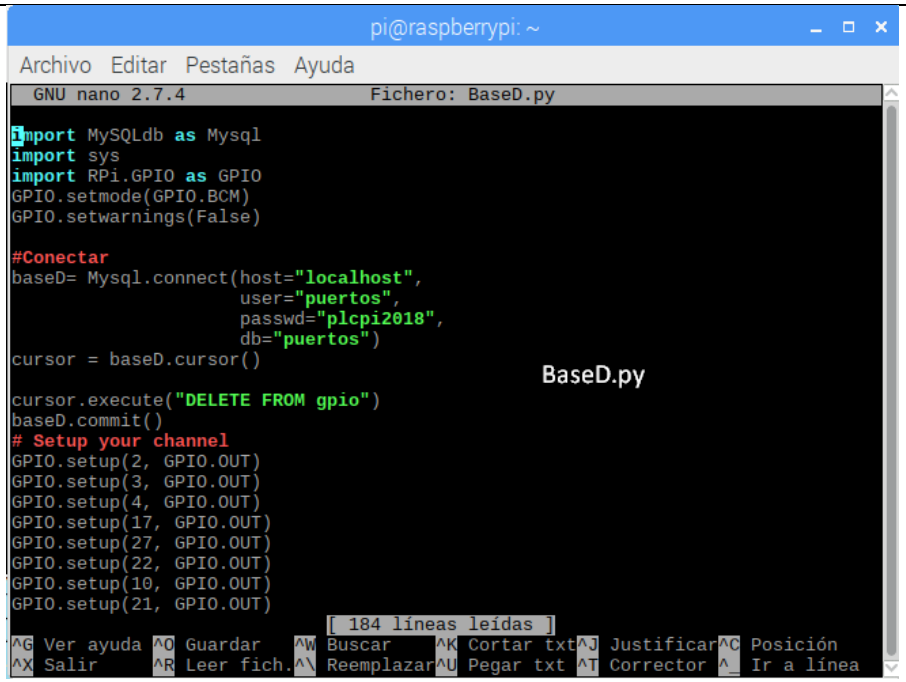


Figura 59. Creación del archivo BaseD.py
Fuente: Realizado por el autor

Creado se procede a guardar por medio de ctrl+o y luego para salir ctrl+x.

d. Para comprobar su funcionamiento se ejecuta el programa con el comando **python3 BaseD.py**

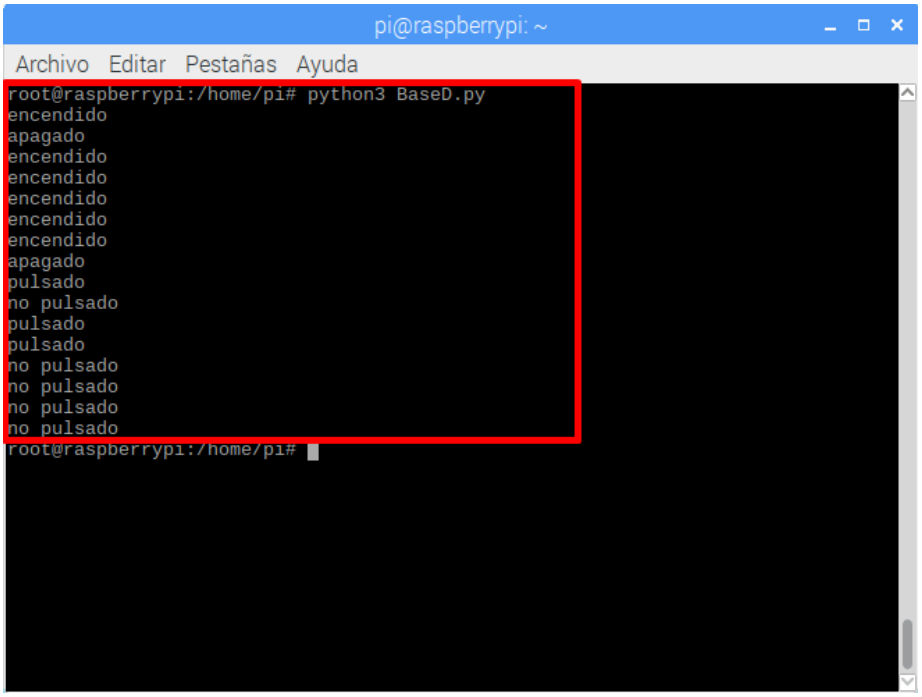


Figura 60. Ejecución de programa BaseD.py
Fuente: Realizado por el autor

Como se visualiza este programa se encarga de subir valores a la base de datos.

4. Para comprobar si en la base de datos se agregaron los valores existen las siguientes formas:
    a. Primero la forma directa es accediendo al terminal y escribiendo **mysql** seguido de la tecla enter, una vez dentro de la base de datos se ingresa los siguientes comandos: use puertos; SELECT * FROM gpio

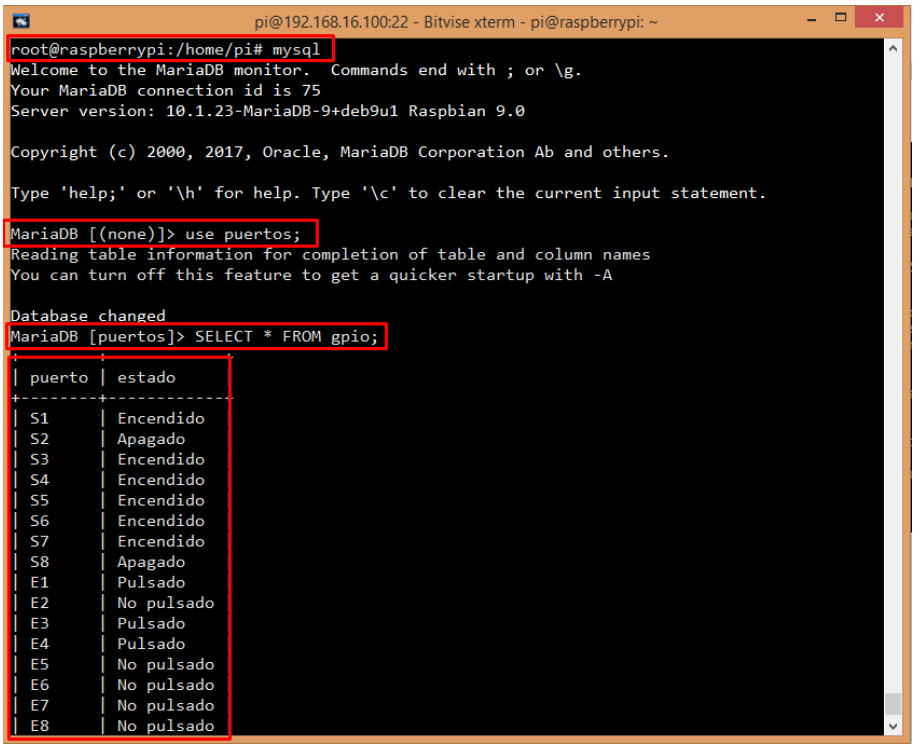


Figura 61. Visualización de base de datos por medio de mysql
Fuente: Realizado por el autor

    b. Por medio de un programa para la lectura de la base de datos, para esto es fundamental crear un archivo definido con el nombre, visualizer.py utilizando el comando nano visualizer.py, una vez creado se carga el código disponible en el anexo 13.

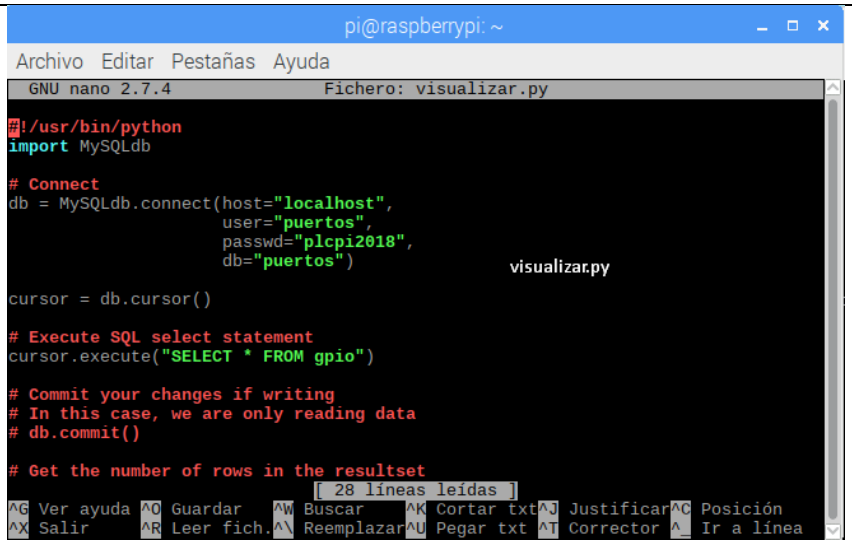


Figura 62. Creación del archivo visualizar.py
Fuente: Realizado por el autor

Copiado correctamente el código se utiliza ctrl + o para guardar y ctrl + x para salir.

Es necesario ejecutar el programa por medio del comando python3 visualizar.py en el terminal de raspbian.

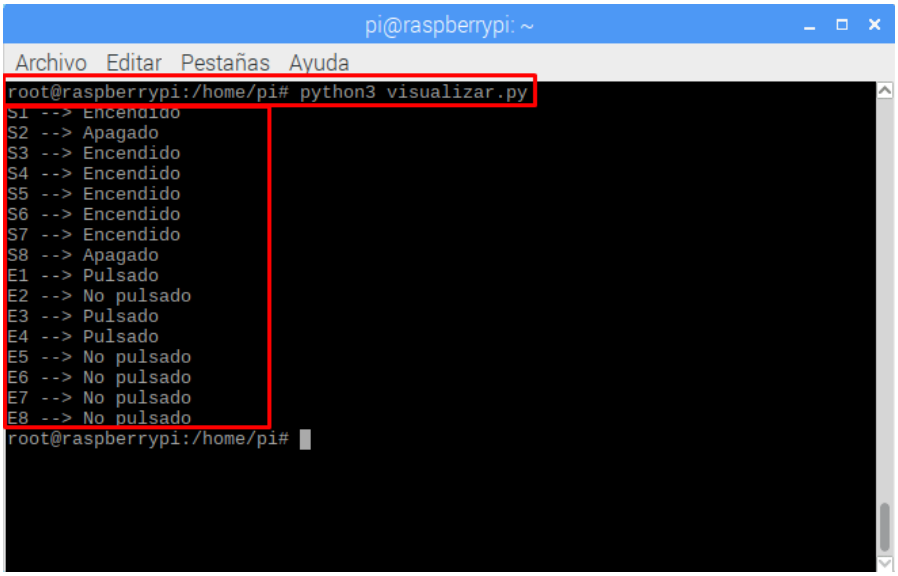


Figura 63. Ejecución del programa visualizar.py
Fuente: Realizado por el autor

5. Para comprobar que la base de datos almacene los valores de los puertos entrada y salida de forma correcta se procede a verificar el módulo y mirar la posición y distribución de las luces demostrando que se encuentran en la misma posición almacenada.

Figura 64. Panel de puertos GPIO
Fuente: Realizado por el autor

Se puede observar todos los puertos coinciden en posición y activación con la base de datos.

DIAGRAMA

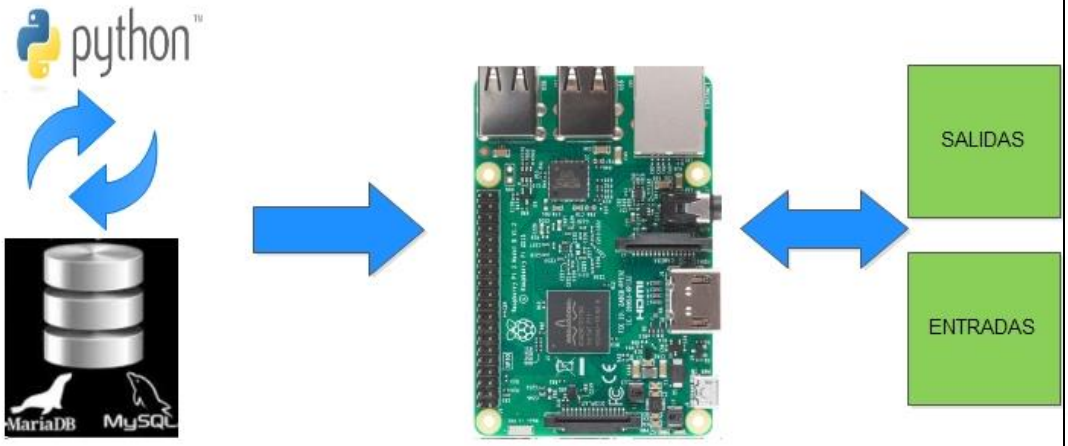


Figura 65. Ejecución del programa visualizar.py
Fuente: Realizado por el autor

4.2.4. Práctica 4

	**UNIVERSIDAD POLITÉCNICA SALESIANA** **FACULTAD DE INGENIERÍA** **SEDE GUAYAQUIL**

CARRERA	PRÁCTICA N^0	NOMBRE DE LA PRÁCTICA
ELECTRÓNICA	4	Exportar en formato Excel

OBJETIVOS
Exportar la base de datos en formato xls a través de un interfaz web Explorar las capacidades de programación web con el uso de las librerías php para manejo de base de datos Conocer las capacidades de exportación de una base de datos hacia un formato reconocible en cualquier sistema Visualizar en Excel la base de datos

EQUIPOS A UTILIZAR	
• Módulo Didáctico • Access Point Inalámbrico • Raspberry Pi 3	• Pantalla LCD touch • Teclado USB inalámbrico • Computador portátil o SmartPhone

INSTRUCCIONES
1) Crear un programa en php que permita ser agregado como función de un

botón para el envío de la base de datos desarrollada en mysql a un formato comprensible en cualquier Sistema operativo.

2) Utilizar las librerias disponibles para manejo de base de datos por medio de php
3) Verificar que la base de datos se exportó de forma correcta y coincide con los valores registrados en la base de datos en sus respectivas filas y columnas.

PROCEDIMIENTO

1. Para esta parte del proceso lo que se requiere es la descarga de la base de datos en un formato que permita compartirlo hacia cualquier medio, por este motivo se utilizo el formato de un archivo excel, el cual tiene compatibilidad tanto como para windows y linux.
2. Lo primero que se debe realizar es la agregación de un botón dentro de la pagina principal que permita ejecutar el programa en php que exporte y genere el archivo tipo .xls.
    a. Es necesario dirigirse a la ubicación html, para lo cual se utilizara el siguiente comando descrito en previas prácticas que es cd /<ubicación>. En este caso el comando dentro del terminal es: cd /var/www/html.

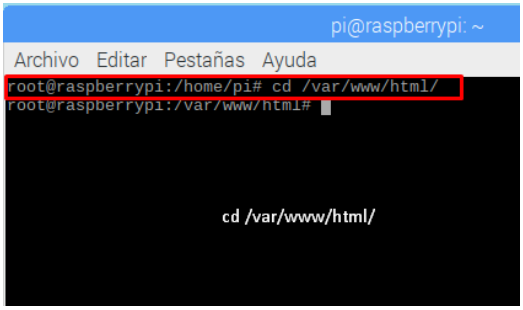


Figura 66. Accediendo directorio html
Fuente. Realizado por el autor

    b. Dentro de la misma es necesario editar el archivo index.php agregando el código siguiente dentro de la division del botón de Entrada y salida

```
<a href="export.php" class="w3-button w3-
blue"> Exportar Excel</a>
```

c. Para esto se utiliza el comando nano index.php, dentro del botón se debe poner el enlace hacia el archivo export.php que se generará en los pasos posteriores.

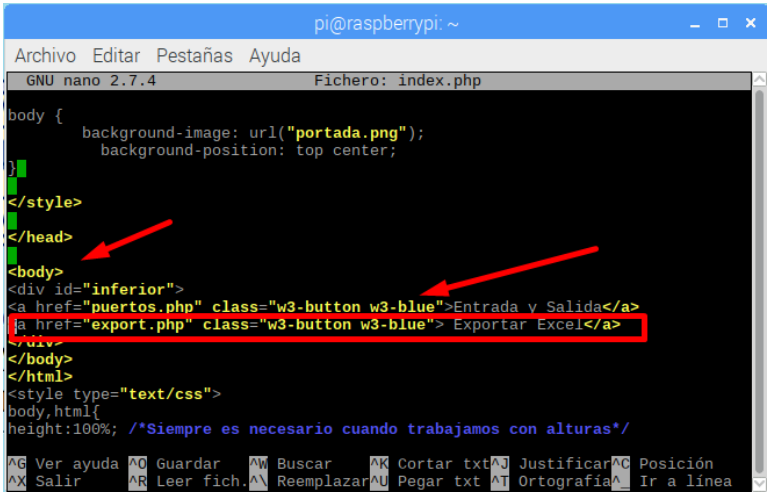


Figura 67. Dentro del archivo index.php
Fuente. Realizado por el autor

d. Cargado el botón aparecerá en la pagina principal del interfaz web, para poder visualizarlo, primero es necesario verificar que direccion ip tiene el raspberry con el comando ifconfig, luego de esto solo basta con usar un navegador web que se encuentre conectado a la red Módulo Didactico.

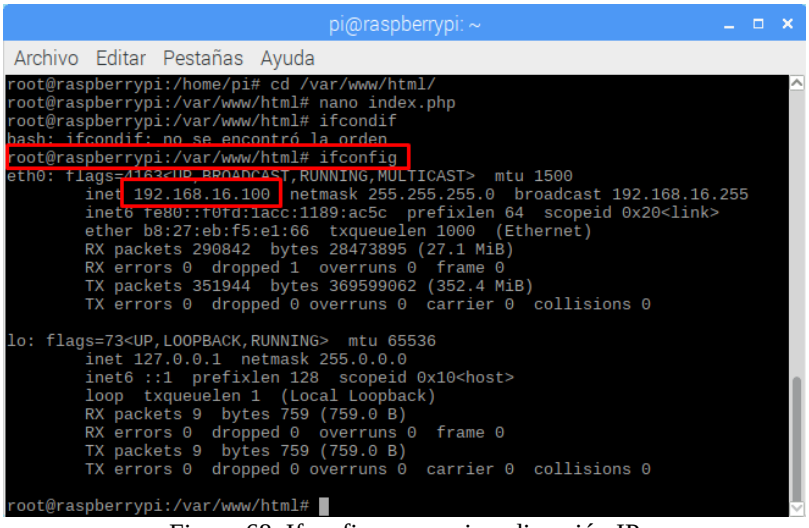


Figura 68. Ifconfig para revisar dirección IP
Fuente. Realizado por el autor

Figura 69. Portada con el botón "Exportar Excel" agregado
Fuente. Realizado por el autor

Se puede observar en la Figura 69, el botón se agrego dentro de la portada

e. Es necesario generar el código que ejecute la consulta y creación del fichero para su previa descarga, por esto se debe crear un archivo de nombre export.php mediante el comando **nano export.php** y agregar el código que se encuentra en el Anexo 14.

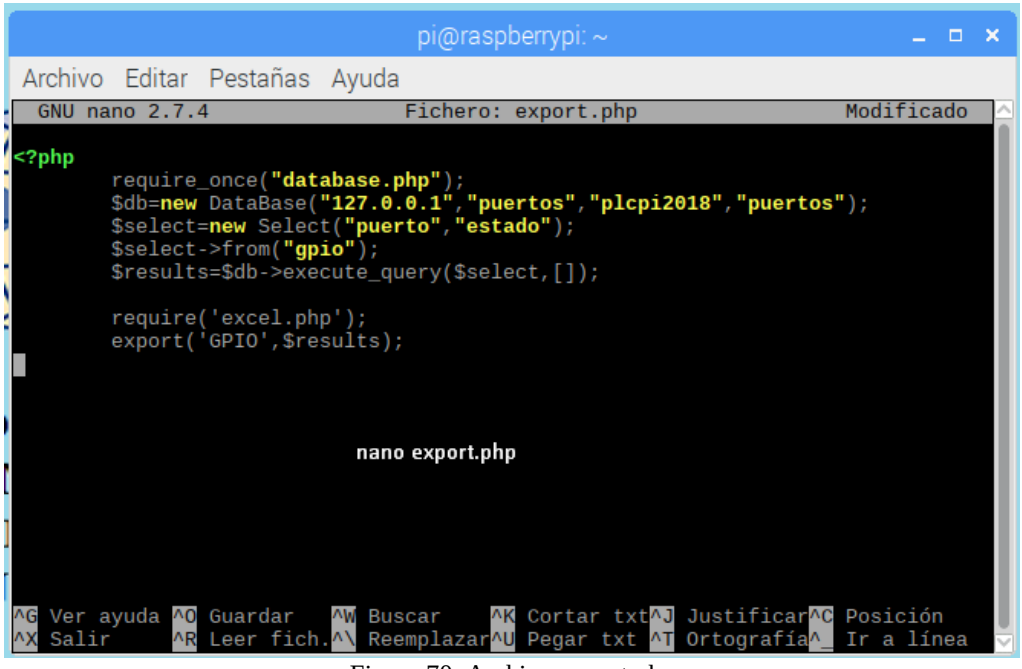


Figura 70. Archivo export.php
Fuente. Realizado por el autor

Completado el código se utiliza **ctrl + o** para guardar y **ctrl + x** para salir.

3. Para el enlace dentro del código se utilizará dos librerias predeterminadas recuperadas de los ejemplos instalados dentro de los repositorios de mysql, para esto se utilizara la librería database.php que se encarga de enlazar y recuperar los registros de la base de datos, mientras que para el formato en modo excel se usará la librería excel.php que se encarga de trasladar los datos a formato excel, cada uno de las librerias y sus códigos se encuentran en el Anexo 15.
    a. Para cada una de las librerias es necesario crearlos y cargar el código utilizando el comando nano y todo debe ser creado dentro de la carpeta html.

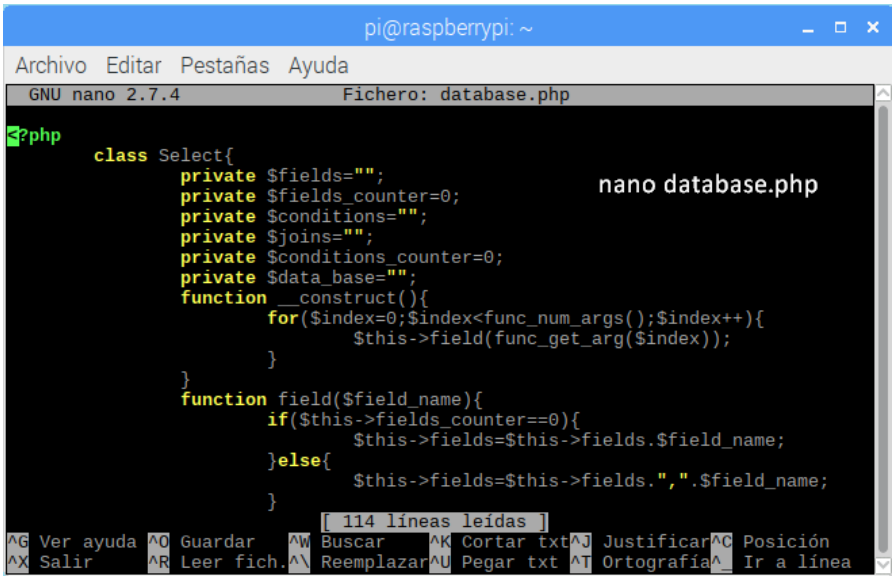


Figura 71. Librería database.php
Fuente. Realizado por el autor

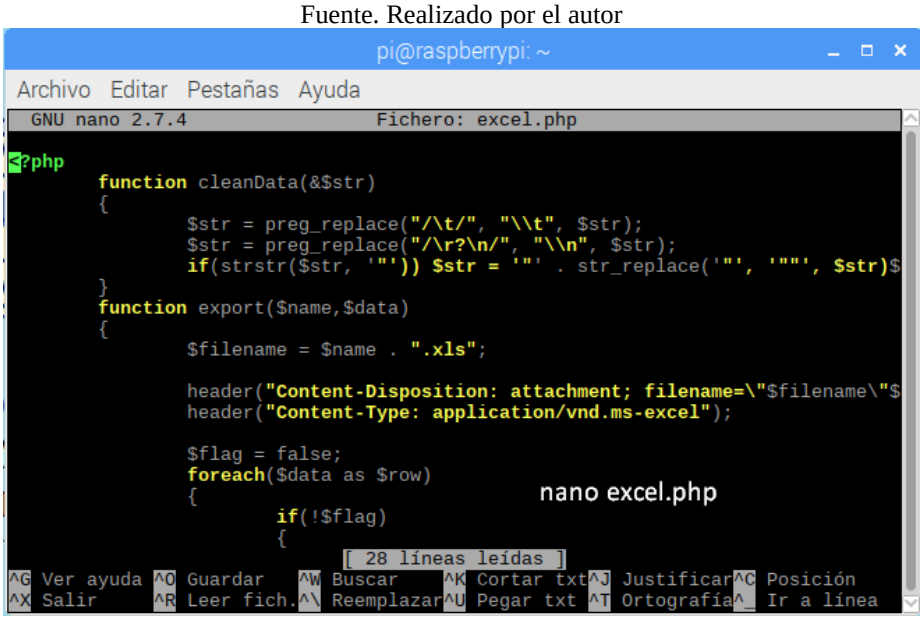


Figura 72. Librería excel.php
Fuente. Realizado por el autor

4. Para verificar su funcionamiento lo único que se debe realizar es ingresar a la pagina web desde cualquier navegador conectado a la red Inalámbrico Módulo Didactico y dar click en el botón Exportar Excel.

Figura 73. Portada con botón de exportar a excel.php
Fuente. Realizado por el autor

Luego simplemente se abre el archivo excel descargado se da click en continuar de todos modos lo unico que indica es que el formato es un formato antiguo de excel al poner continuar lo abre normalmente.

Figura 74. Abrir archivo descargado GPIO.xls
Fuente. Realizado por el autor

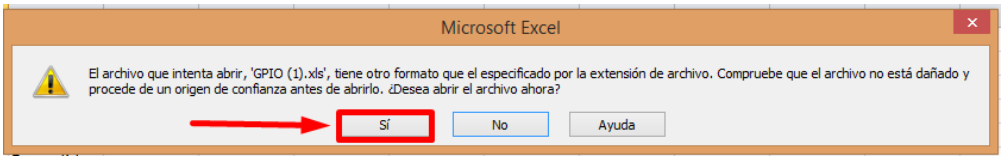


Figura 75. Aceptación de apertura forzada
Fuente. Realizado por el autor

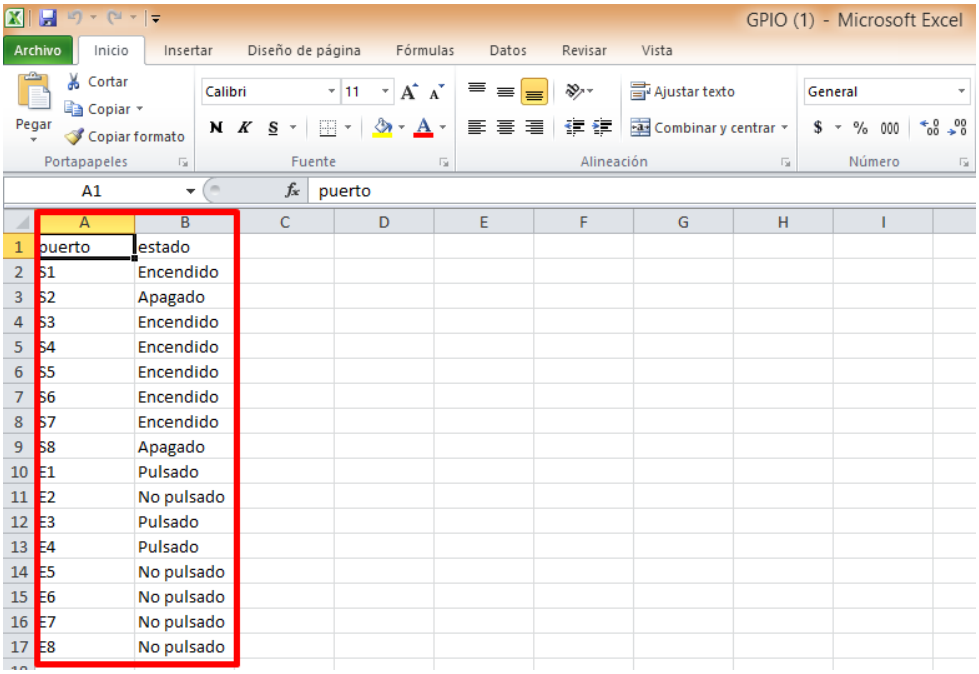


Figura 76. Visualización del archivo en excel
Fuente. Realizado por el autor

Como se visualiza en la Figura 76 la base de datos esta exportada de forma satisfactoria, para su comparacion se puede abrir la base de datos como se indico en la práctica 2 y mirar que es identica la respuesta.

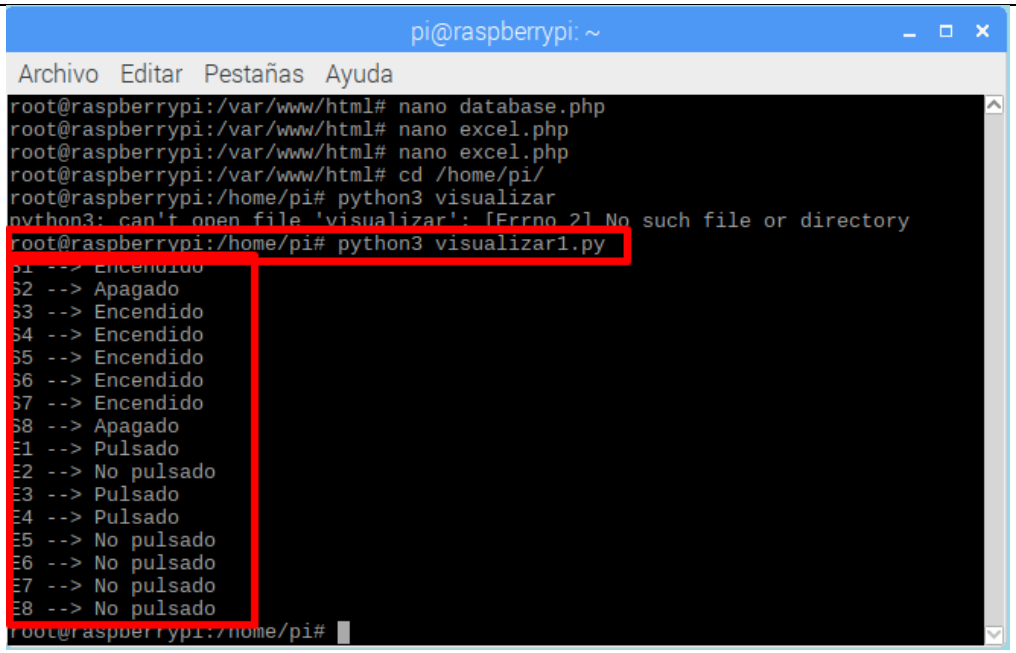


Figura 77. Visualización por medio de programa visualizar.py
Fuente. Realizado por el autor

DIAGRAMAS

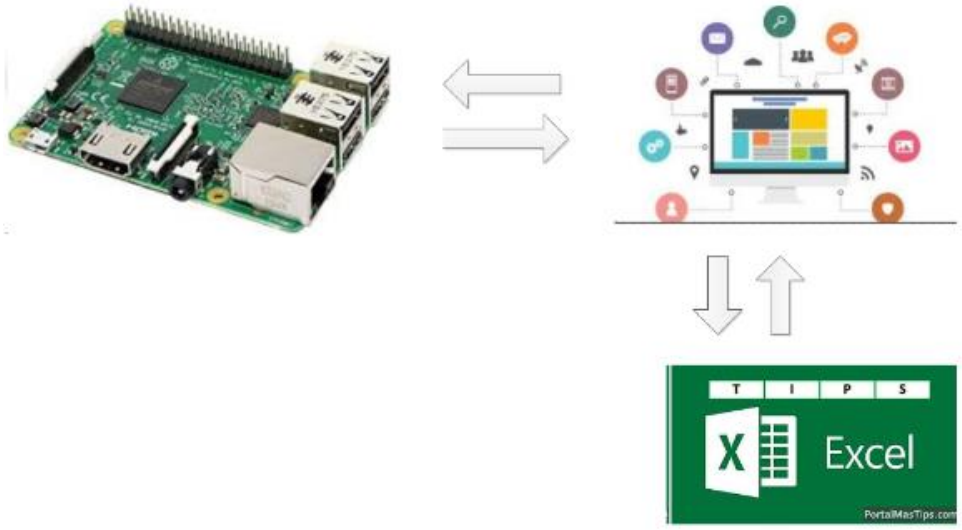


Figura 78. Diagrama de bloques Práctica 4
Fuente: Realizado por el autor

4.2.5. Práctica 5

	**UNIVERSIDAD POLITÉCNICA SALESIANA** **FACULTAD DE INGENIERÍA** **SEDE GUAYAQUIL**

CARRERA	**PRÁCTICA N^0**	**NOMBRE DE LA PRÁCTICA**
ELECTRÓNICA	5	Elaboración de un interfaz de monitoreo gráfico

OBJETIVOS
• Utilizar Raspberry Pi 3 para el diseño de un interfaz web que permita visualizar la base de datos o datos recibidos • Anexarlo al index • Revisar la funcionalidad de las prácticas anteriores

EQUIPOS A UTILIZAR	
• Módulo Didáctico • Router • Cable de red	• Laptop • Pantalla touch • Software Putty

INSTRUCCIONES

1) Verificar las conexiones del Raspberry con el relay
2) Revisar el código necesario en php que permita visualizar los datos
3) Utilizar la conexión SSH para comunicarse con el Raspberry de forma remota
4) Realizar una página que este indexada a la portada o página web

1. PROCEDIMIENTO

2. Creación de la interfaz web

a. Vamos a la carpeta raíz del servidor web

```
cd /var/www/html
```

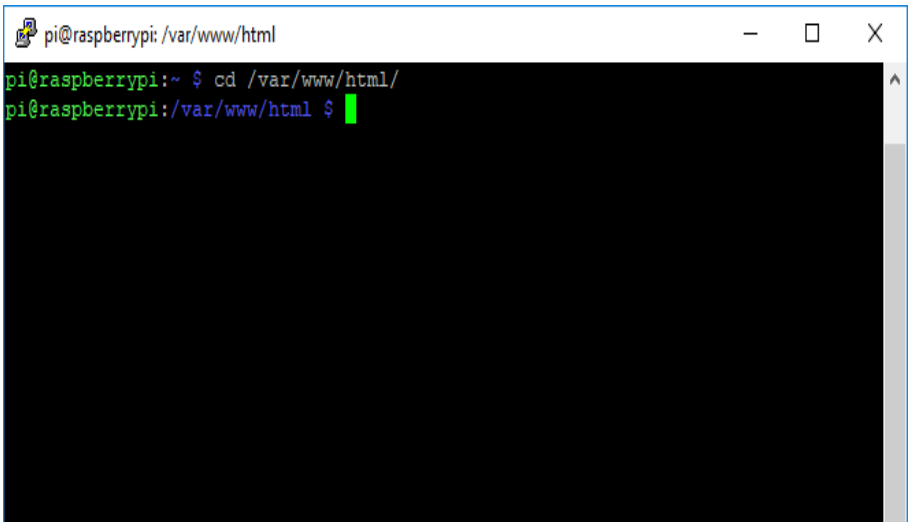


Figura 79. Ingreso a la carpeta del servidor

Fuente: Realizado por el autor

b. Creamos un archivo chart.php

```
sudo nano chart.php
```

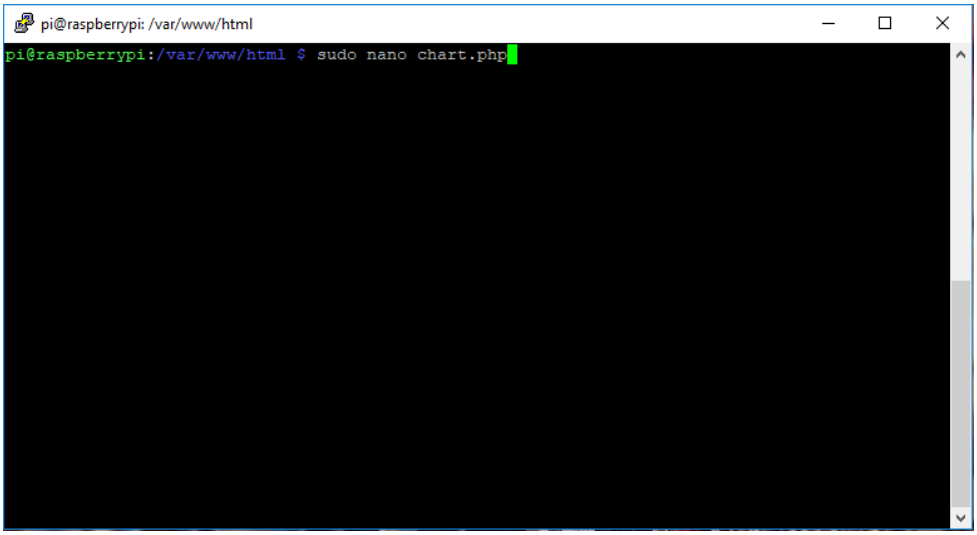


Figura 80. Creación del archivo chart.php

Fuente: Realizado por el autor

c. Como puede notarse usaremos la misma plantilla de la práctica anterior

```
<html>
        <head>
                <meta name="viewport" content="width=device-width,
initial-scale=1">
                <link rel="stylesheet"
href="https://www.w3schools.com/w3css/4/w3.css">
        </head>
        <body class="w3-light-grey">
                ...
        </body>
</html>
```

d. En el interior del “body” colocamos el título de la página.

```
<h1 class="w3-center">
        Gráfica de estados de I0
</h1>
```

e. Además, el espacio donde se dibujará la gráfica

```
<div class="w3-card-4 w3-margin w3-round">
        <div id="chart_div"></div>
</div>
```

f. A continuación, importamos la librería de javascript para hacer las

gráficas

```
<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
```

g. Luego creamos un nuevo “script” para definir qué se debe mostrar en la gráfica.

```
<script type="text/javascript">
        ...
</script>
```

h. Importamos los módulos necesarios de la librería

```
google.charts.load('current', {'packages':['corechart']});
```

i. Definimos la función que se debe ejecutar cuando el navegador esté listo para hacer el render de la gráfica.

```
google.charts.setOnLoadCallback(drawChart);
```

j. Dentro de la función “drawChart” definimos un array con la información de la gráfica.

```
var data = google.visualization.arrayToDataTable([
```

k. Abrimos un tag Php para cargar la información de la base de datos. De forma similar a la práctica anterior vamos a levantar una conexión con la base de datos y extraer los datos de la tabla “io”.

```
<?php
        require_once("database.php");
        $db=new DataBase("127.0.0.1","plcpi","plcpi2018","plcpi");
        $select=new Select("name","value");
        $select->from("io");
        $select->where("name",">");
        $conditions=array("I0");
        $results=$db->execute_query($select,$conditions);
```

l. Hacemos un “foreach” para iteractuar sobre cada uno de los elementos extraídos de la base de datos y los guardamos en dos arreglos diferentes

separados por cada nombre del puerto.

```
$i1=[];
$i2=[];

foreach($results as $result)
{
        if($result['name']=='I1') $i1[]=$result['value'];
        if($result['name']=='I2') $i2[]=$result['value'];
}
```

m. Con los datos correctamente organizados procedemos a “imprimirlos” al documento (página web actual)

```
        echo "['Nombre','I0','I1'],";
         for($index=0;$index<count($i1);$index++)
         {
                 echo
"[".$index.",".$i1[$index].",".$i2[$index]."],";
         }

?>
```

n. Luego creamos la variable “options” donde se definen los parámetros de la gráfica como el título (title) el nombre y el color del eje horizontal (hAxis) y el valor mínimo del ejve vertical (vAxis).

```
        var options = {
          title: ' grafica del IO del PLC',
          hAxis: {title: 'Tiempo',  titleTextStyle: {color:
'#333'}},
          vAxis: {minValue: 0}
        };
```

o. Creamos una nueva gráfica, con los parámetros previamente definidos.

```
        var chart = new
google.visualization.AreaChart(document.getElementById('chart_div'))
;
        chart.draw(data, options);
      }
```

3. Registro de resultados

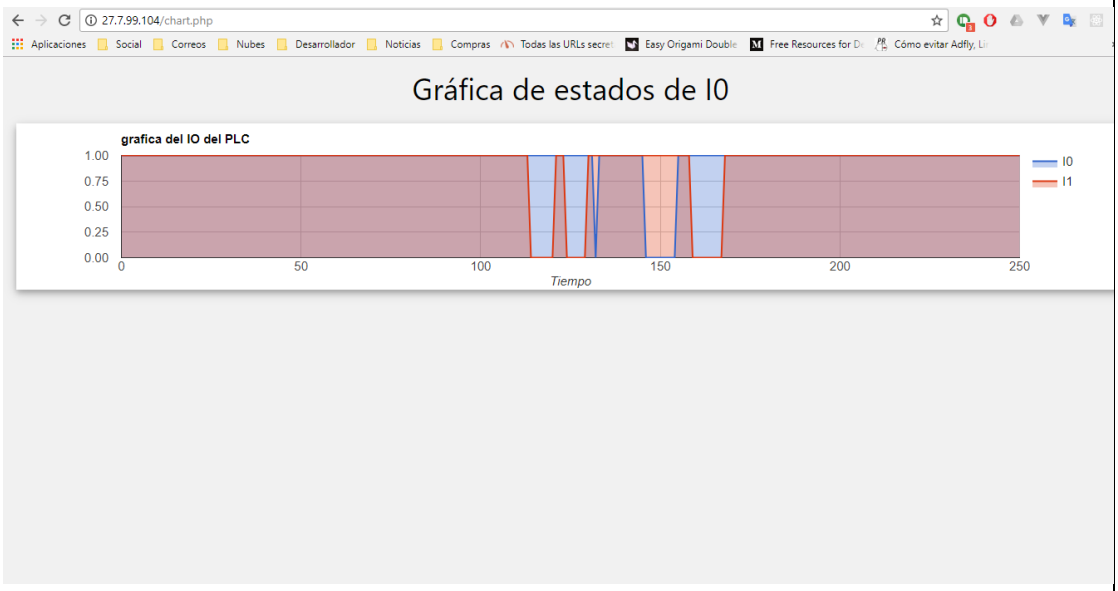


Figura 81.Interfaz Monitorización

Fuente: Realizado por el autor

DIAGRAMAS

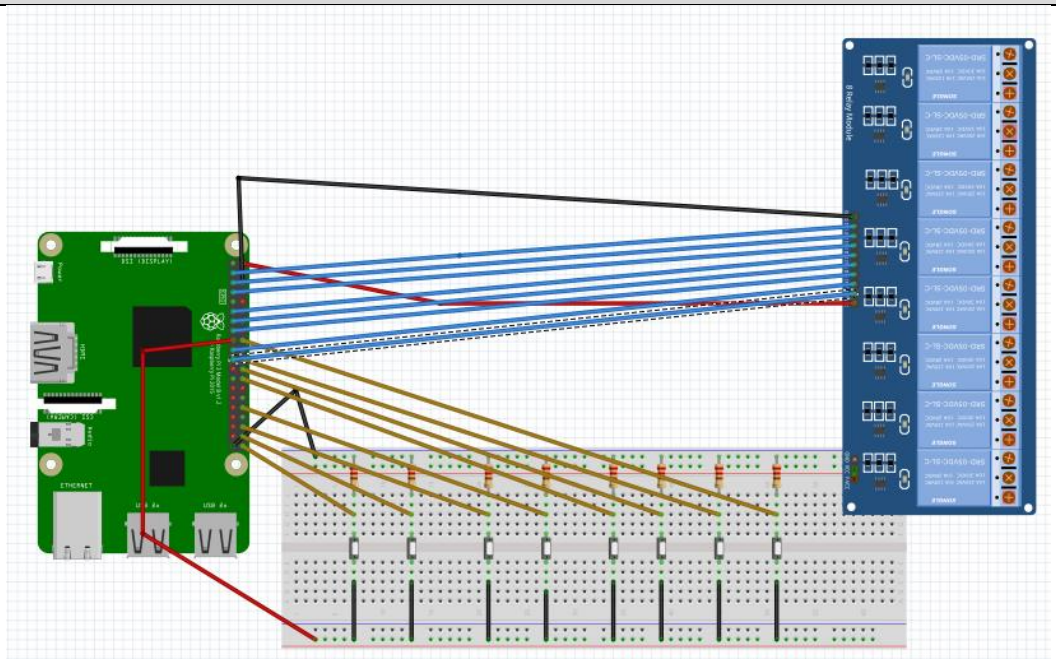


Figura 82.Diagrama Pines en Fritzing
Fuente: Realizado por el autor

4.2.6. Práctica 6

	UNIVERSIDAD POLITÉCNICA SALESIANA **FACULTAD DE INGENIERÍA** **SEDE GUAYAQUIL**

CARRERA	PRÁCTICA N^0	NOMBRE DE LA PRÁCTICA
ELECTRÓNICA	6	Elaboración de un servidor web

OBJETIVOS
• Utilizar Raspberry Pi 3 para la implementación de un servidor web • Interconectar los interfaces web mediante HTML, php y Python • Comprobar la funcionalidad del servidor web • Generar enlaces direccionados a las prácticas elaboradas.

EQUIPOS A UTILIZAR	
• Cable de poder del Módulo Didactico • Router Inalámbrico • Cable de red	• Pantalla LCD • Teclado Inalámbrico • Módulo didáctico • Arduino Mega

INSTRUCCIONES

1) Utilizar la pantalla touch para poder manipular las herramientas necesarias para la práctica
2) Comprobar las conexiones de los cables de conexión entre cada módulo
3) Comprobar funcionamiento de los módulos necesarios para la práctica
4) Configurar cada uno de los ficheros indispensables para la realización de la práctica
5) Desarrollar un software libre que permita interactuar al usuario de forma directa y remota con el interfaz web almacenado dentro de la Raspberry pi 3.
6) Agregar botones para cada práctica a realizar que contemple el interfaz web
7) Generar vínculos que generen una plataforma dinámica para el usuario

DIAGRAMAS

Flujograma Servidor WEB

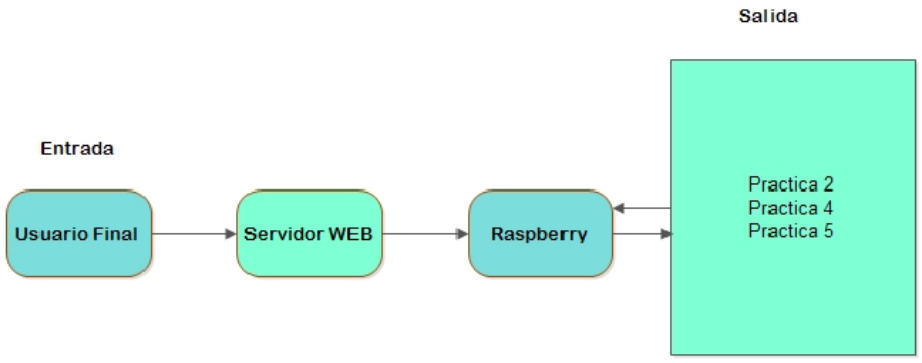


Figura 83. Flujo grama Servidor Web
Fuente: Realizado por el autor

PROCESO

1. Se realiza el proceso de verificación como se puede observar en la práctica 1.
2. Es necesario generar los enlaces para cada una de las prácticas elaboradas anteriormente.
    a. Primero se genera el enlace entre la práctica 2, para esto se requiere agregar dentro de la página puertos.php el siguiente código:

```
<div id="inferior"><a href="index.php" class="w3-button w3-blue">
                    Inicio
                </a>
            </div>
```

Como se puede observar en la figura 84, el enlace a la página principal se encuentra disponible.

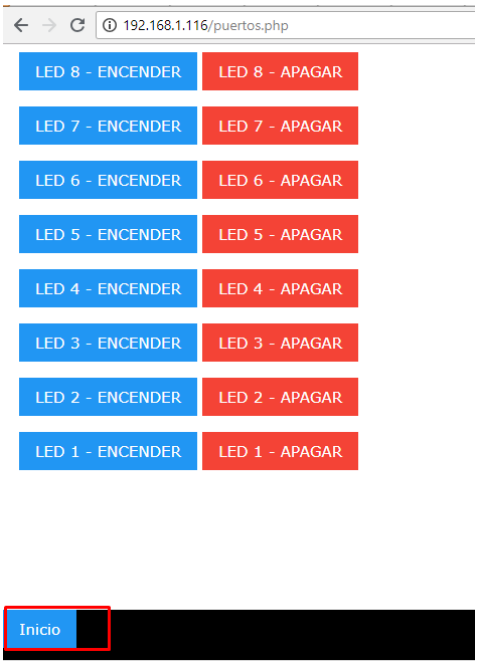


Figura 84. Botón retorno Pagina Principal en Práctica 1
Fuente: Realizado por el autor

b. Ahora se procede a enlazar la Práctica 3 para descargar de la base de

datos en formato Excel.

Para esto se procede a utilizar el comando en el interior de la carpeta /var/www/HTML y se escribe nano index.php, y se agrega las siguientes líneas de código.

```
<a href="export,php" class="w3-button w3-blue">

                    Práctica 4

                </a>
```

Como se puede ver en la figura 85, se agregó el enlace para descargar directamente desde la portada la base de datos almacenada de cada posición de los puertos del Raspberry.

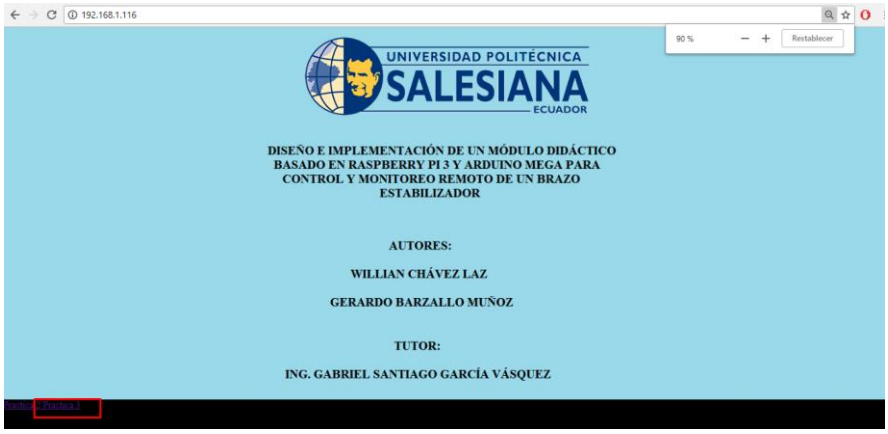


Figura 85. Botón de acceso a la práctica 4
Fuente: Realizado por el autor

c. También es necesario agregar la práctica 5 que se trata de visualización de estados por medio de una gráfica, para esto primero se agrega el botón en la portada agregando en index.php la siguiente línea de código:

```
<a href="chart.php" class="w3-button w3-blue">
```

Práctica 5

Al recargar la página y se podrá visualizar el enlace generado.

Figura 86. Botón de acceso a la práctica 5
Fuente: Realizado por el autor

Además, se necesita dentro de la página chart.php que es la práctica 5 un botón de enlace para retornar al inicio por medio del siguiente código, utilizando en comando **nano chart.php.**

Inicio

Figura 87. Botón de regreso a portada en práctica 5

Fuente: Realizado por el autor

Con este proceso se puede apreciar que el interfaz web enlaza todas las prácticas hacia la portada y al interfaz web generado, con lo que se demuestra que se logró un servidor web interactivo.

4.2.7. Práctica 7

	**UNIVERSIDAD POLITÉCNICA SALESIANA** **FACULTAD DE INGENIERÍA** **SEDE GUAYAQUIL**

CARRERA	PRÁCTICA N^0	NOMBRE DE LA PRÁCTICA
ELECTRÓNICA	7	Identificación del sistema

OBJETIVOS
Identificar el tipo de sistema que se empleara en el proyecto para de esa forma En base a la identificación del sistema encontrar el modelo matemático para el motor emg49 Desarrollar una simulación en el programa Scilab y su herramienta xcos para sacar los valores de funcionamiento necesarios para la identificación de la función de transferencia.

EQUIPOS A UTILIZAR	
• Arduino MEGA • Conector USB	• Laptop • Arduino IDLE

• Motor DC • Scilab • Puente H	• Fuente 110 a 24V • Sensor Infrarrojo

INSTRUCCIONES

1) Interconectar el Motor EMG49 hacia el puente H de 43a y hacia el transformador de 120VAC a 24V
2) Conectar el encoder del motor al Arduino MEGA
3) Conectar PWM 7 y 8 del Puente H 43a según se indique
4) Configurar Scilab para los escenarios necesarios
5) En base a los datos del datashet y obtenidos por medio de scilab encontrar la función de transferencia

PROCESO

1. Dentro de los primeros pasos para la identificación del sistema es necesario interconectar la placa Arduino Mega con el Encoder y la fuente, como se puede ver en la figura 88.

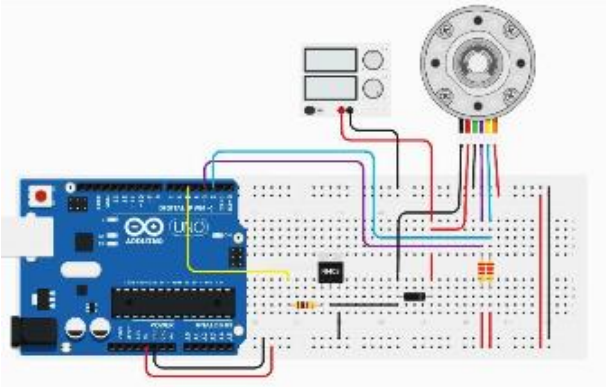


Figura 88. Conexión del motor y Arduino Mega
Fuente: Realizado por el autor

En la Figura 88 se puede observar dos resistencias que van desde el borde positivo del Arduino hacia el encoder y con unas dos resistencias las cuales tienen un valor de 2.2k mientras que el mosfet que se puede ver se utiliza para regular el giro del motor y mantenerlo estable, pero al contar con un puente H de 43 amperios se lo utilizará en su lugar.

Una vez realizada la conexión es necesario configurar la herramienta Scilab que permitirá obtener los valores para la identificación del sistema.

Lo primero es cargar el código necesario dentro de Arduino el cual se encuentra almacenado en el escritorio de Raspberry Pi 3, se encuentra con el nombre de **toolbox_arduino_v3.**

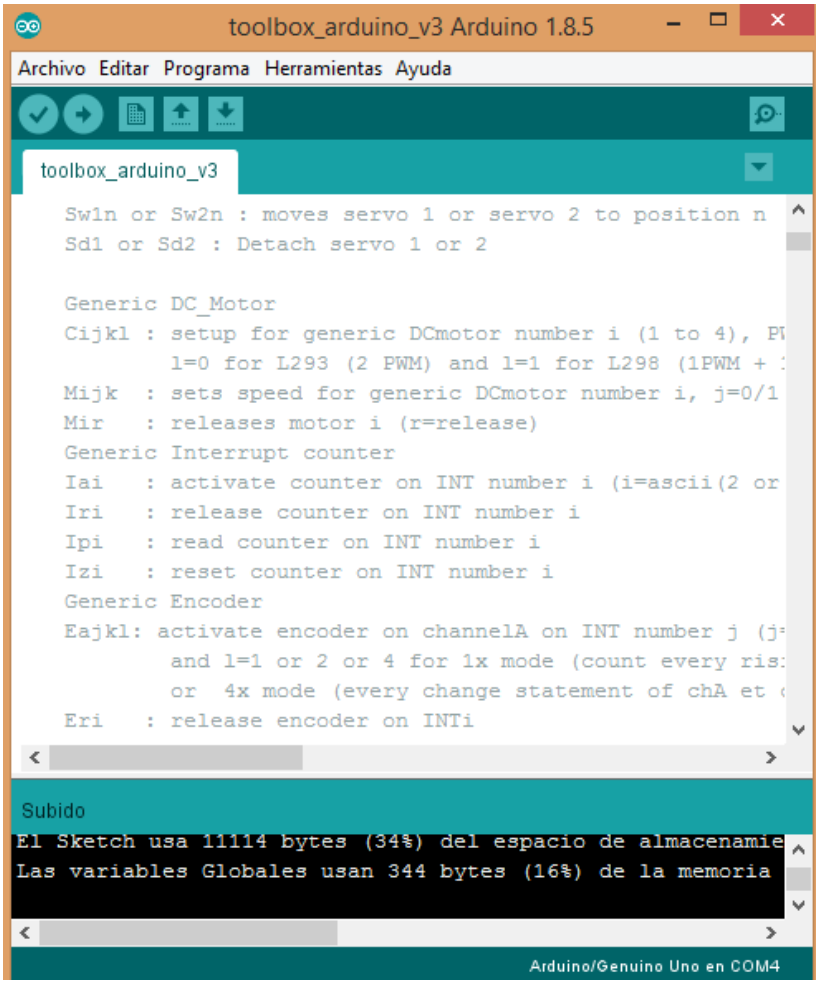


Figura 89. Cargando librería para manejo de scilab
Fuente: Realizado por el autor

SCILAB

2. Es necesario instalar la version 5.5 no superior debido a que la paqueteria de arduino no es compatible con versiones superiores, una vez realizado dentro del interfaz del Scilab para configurarlo se realiza lo siguiente:

    a. Ingresamos al menu y seleccionamos open, de esta forma cargamos las herramientas de arduino previamente descargadas.

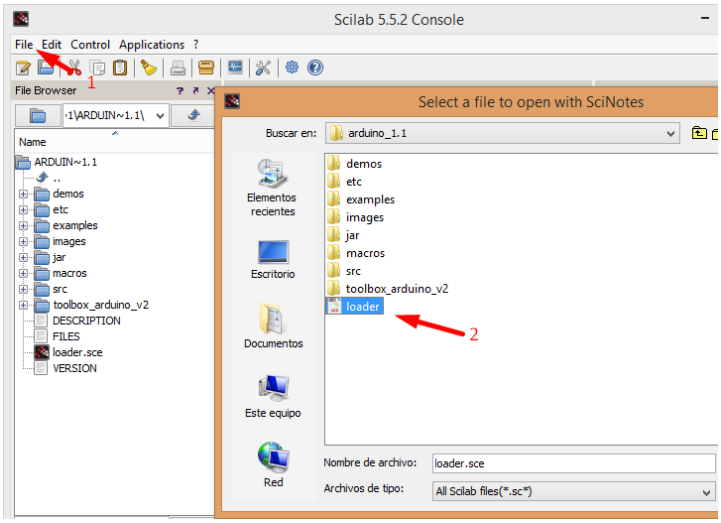


Figura 90. Cargando herramientas de Arduino en Scilab
Fuente: Realizado por el autor

Al momento de abrir el cargador se visualizará la ventana que se observa en la Figura 90 y lo único que se necesita es usar el botón de ejecutar y el programa de arduino estará cargado. Este proceso se lo realiza cada que se utilizará arduino en scilab.

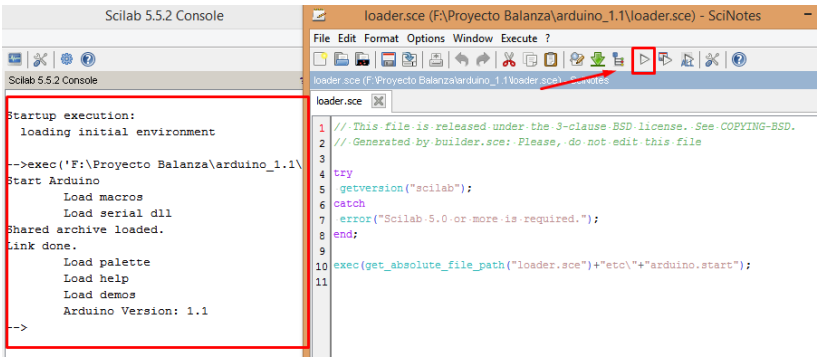


Figura 91. Ejecutando loader de Arduino
Fuente: Realizado por el autor

b. Efectuado este proceso se procede a abrir la herramienta xcos que permite interactuar de manera dinámica con el arduino y el motor para obtener los valores, esto se encuentra en el panel dentro de aplicaciones.

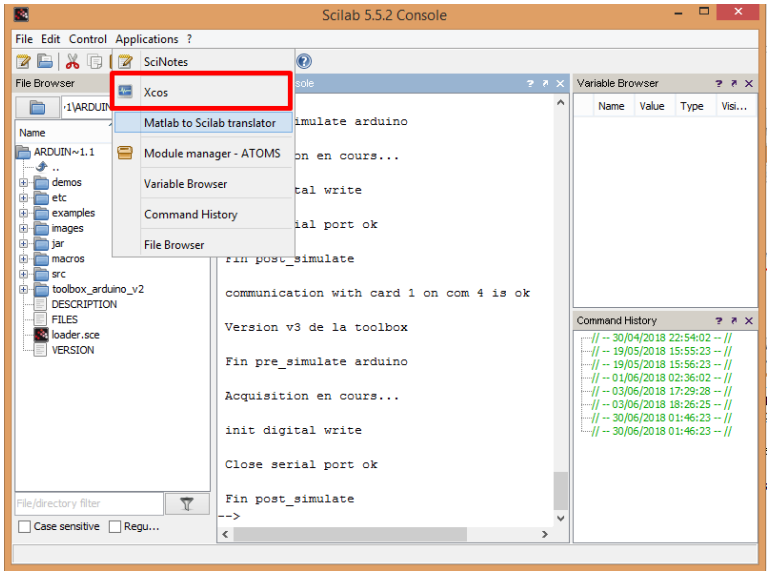


Figura 92. Abriendo xcos
Fuente: Realizado por el autor

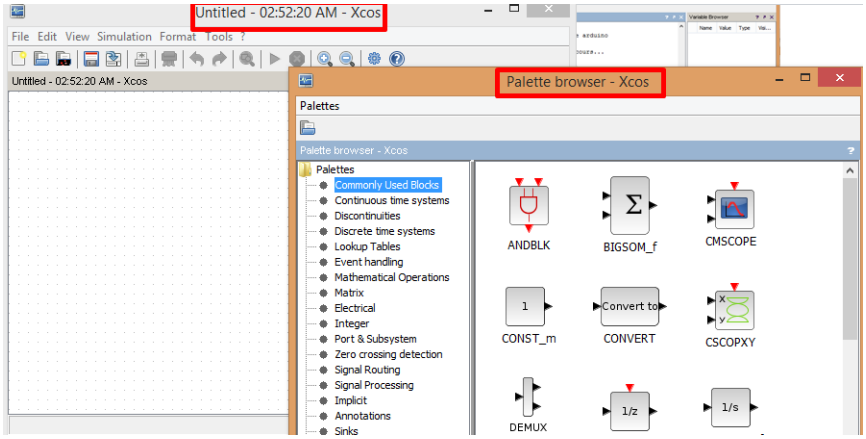


Figura 93. Interfaz xcos
Fuente: Realizado por el autor

Como se visualiza en la figura 93 existe una lamina para las herramientas y una paleta de componentes que se pueden agregar e interactuar entre ellos.

c. El primer proceso se lo realiza por medio de la obtención de los

pulsos necesarios en un determinado tiempo con la utilización del escalón unitario, para esto se debe realizar un programa que se denominará prueba motor y sus componentes se pueden ver interconectados en la Figura 94.

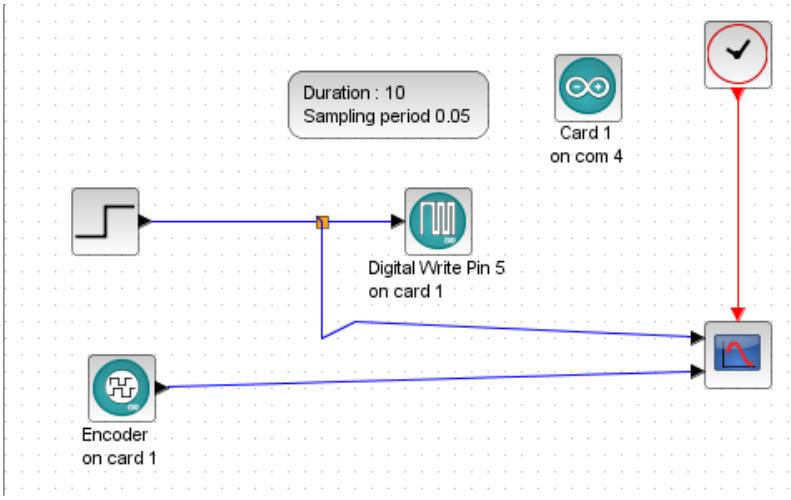


Figura 94. Función Escalón unitario calculo pulsos
Fuente: Realizado por el autor

Se visualiza que se interconecta el encoder para los pines 2 y 3, canal A y B respectivamente, además para la salida del motor será controlada por el pin 5 o cualquier pin digital que se requiera por conexión del módulo es recomendable usar el pin 10 y la comunicación se la realiza por medio del COM4 o cualquier puerto reconocido donde se ubique el arduino, se tomará muestras cada 0.05s en tiempo de 10 segundos.

La respuesta que se obtiene se puede ver reflejada en la Figura 95.

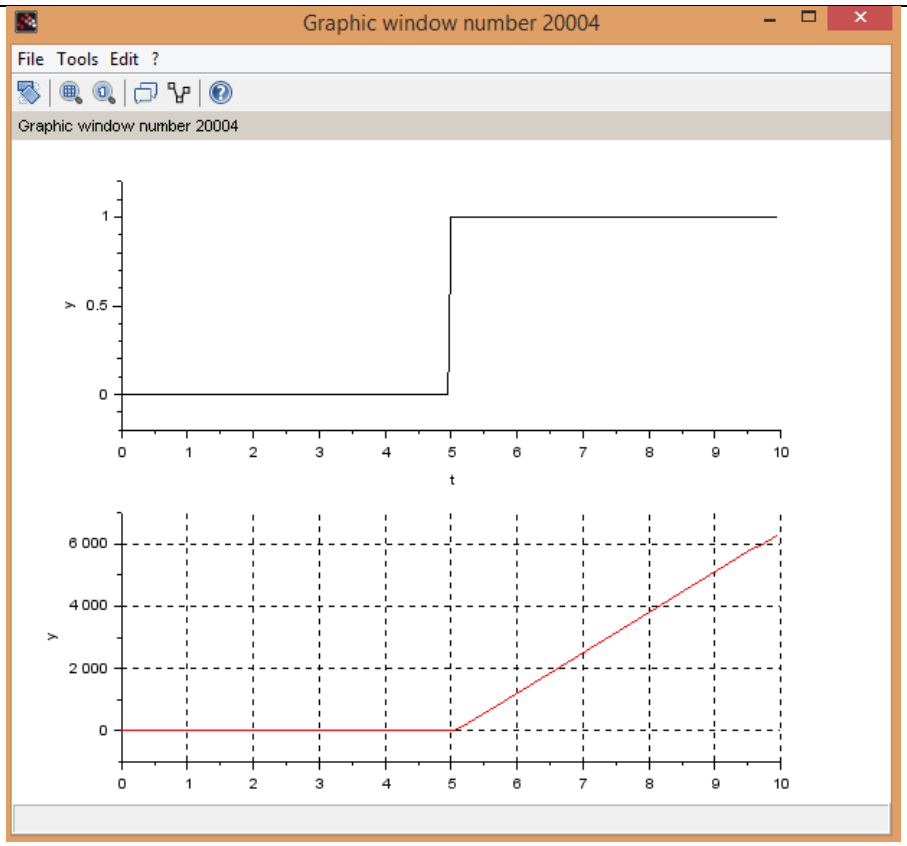


Figura 95. Respuesta Función Rampa pulsos por tiempo
Fuente: Realizado por el autor

Se puede visualizar que se tiene alrededor de 1200 pulsos por segundo para su mejor visualización se realiza un acercamiento en la Figura 96.

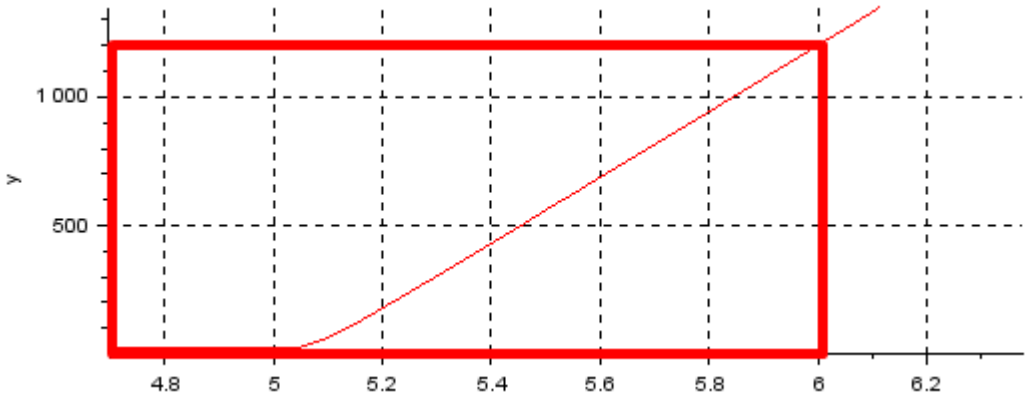


Figura 96. Número de pulsos en 1 segundo
Fuente: Realizado por el autor

Las variaciones que se ven con respecto al datasheet se deben a las caracteristicas de la fuente con la que se trabaja pero se mantienen dentro del margen de error para el calculo de la función de transferencia.

d. El siguiente paso es obtener las revoluciones por minuto del motor, tomando en cuenta el datashet se puede tener valores aproximados de lectura y para poder obtenerlos se modifica las variables en scilab utilizando la función que se encuentra en la figura.

Para la elaboración del diagrama se toma en cuenta los valores de pulsos por muestra que toma lo que se encuentra en el datashet como 980 por muestra.

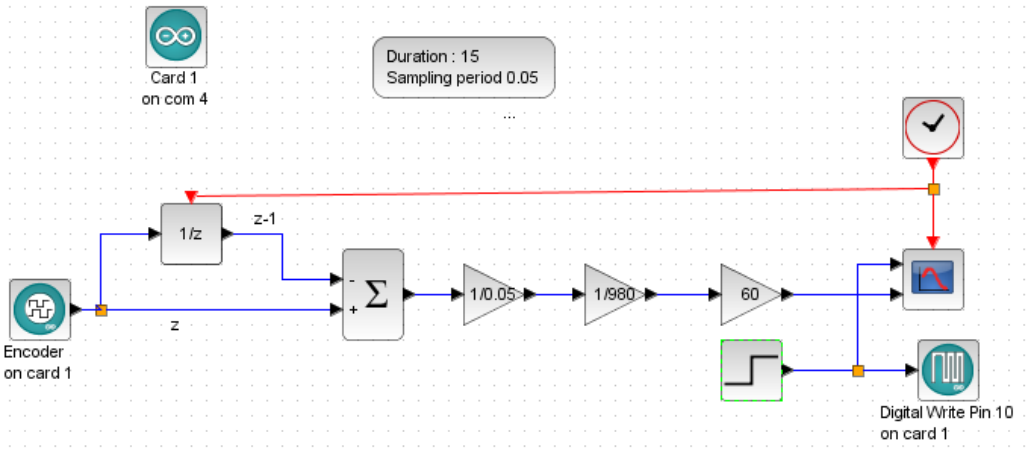


Figura 97. Diagrama para la obtención de rpm
Fuente: Realizado por el autor

Realizado el diagrama es primordial solo cambiar el pin del motor al pin 10 de la placa Arduino Mega y darle al botón de inicio para adquirir los valores que se muestran en la figura 98.

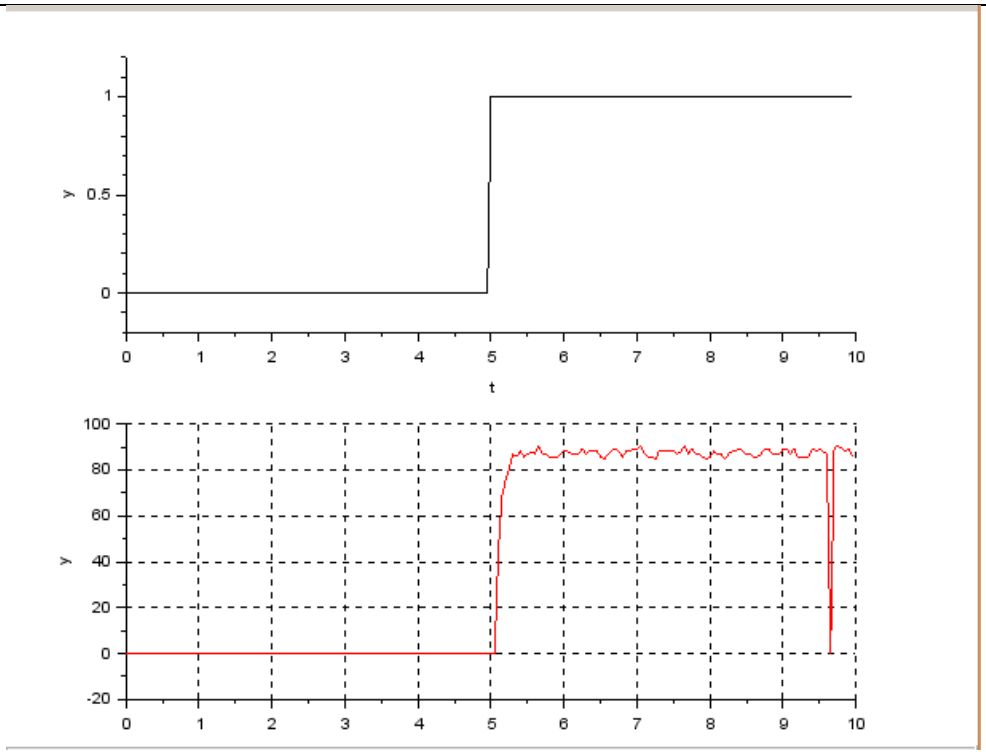


Figura 98. Respuesta del motor en rpm
Fuente: Realizado por el autor

Revisando la respuesta se puede obtener que la velocidad del motor es aproximada a la descrita en el datashet con un valor de 87 rpm, con este valor se procede a realizar una estimación matemática que la represente.

Modelado matemático

Como se observa en la gráfica se puede analizar que se asemeja a una respuesta de un sistema de primer orden con retardo

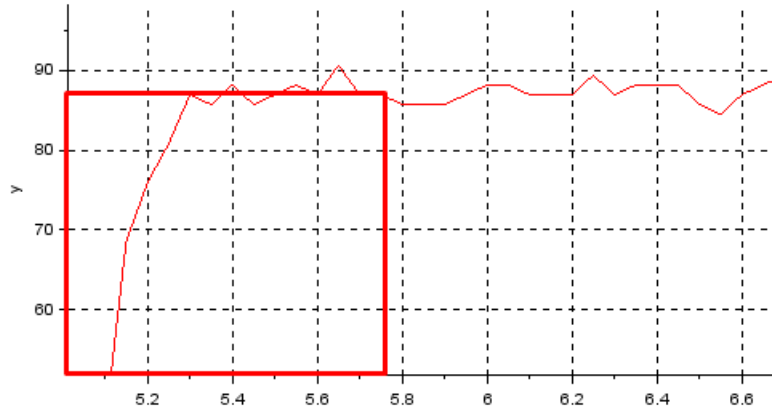


Figura 99. Lectura del primer pico
Fuente: Realizado por el autor

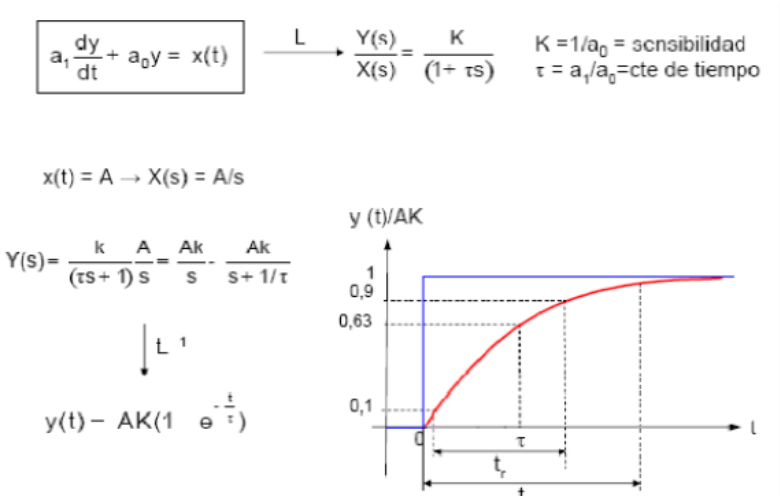


Figura 100. Sistema de primer orden
Fuente: Realizado por el autor

De los sistemas de primer orden se utiliza las ecuaciones para encontrar los valores necesarios para definir la función de transferencia.

Es necesario calcular la ganancia que viene dada por la siguiente función

$$\lim_{t\to\infty}(t) \quad = K = y(\infty)$$

Si acercamos los valores obtenidos por medio de Scilab podemos obtener el punto máximo de la pendiente y resultado es K=87.

Y la constante de tiempo se obtiene gracias a la siguiente función:

$$y(\propto) = 0.632K$$

Se reemplaza los valores de la siguiente forma y se obtiene el siguiente resultado:

$$y(\propto) = 0.632K$$

$$y(\propto) = 0.632(87)$$

$$y(\propto) = 54.98$$

El siguiente paso es obtener el valor de $\propto$ siguiendo el proceso anteriormente

realizado de la figura de respuesta del motor se obtiene la diferencia de tiempo en el arranque del motor el cual se obtiene ubicando el punto y(∝) lo que se puede observar en la Figura 101.

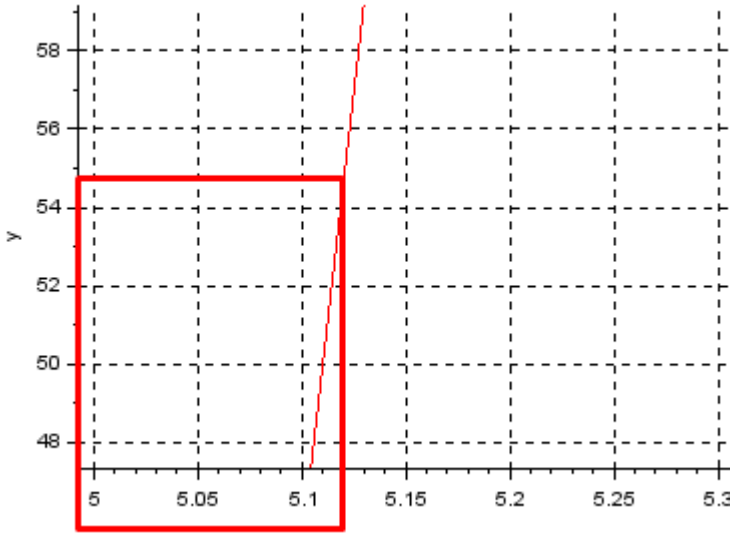


Figura 101. Ubicando y(∝)
Fuente: Realizado por el autor

De la grafica se define que el valor es 5.143 al cual se resta 5 por el tiempo de muestreo o el periodo que se puso al momento de la simulación teniendo lo siguiente:

$$\propto = 0.143 - 0.05$$

$$\propto = 0.93$$

Con estos valores en base a un sistema de primer orden se reemplaza los valores de la siguiente forma:

$$G(s) = 87\frac{e^{-0,05s}}{0.93s + 1}$$

e. Es necesario comparar mediante la respuesta del motor con la simulación, por lo cual en scilab se elabora el programa que lo permita.

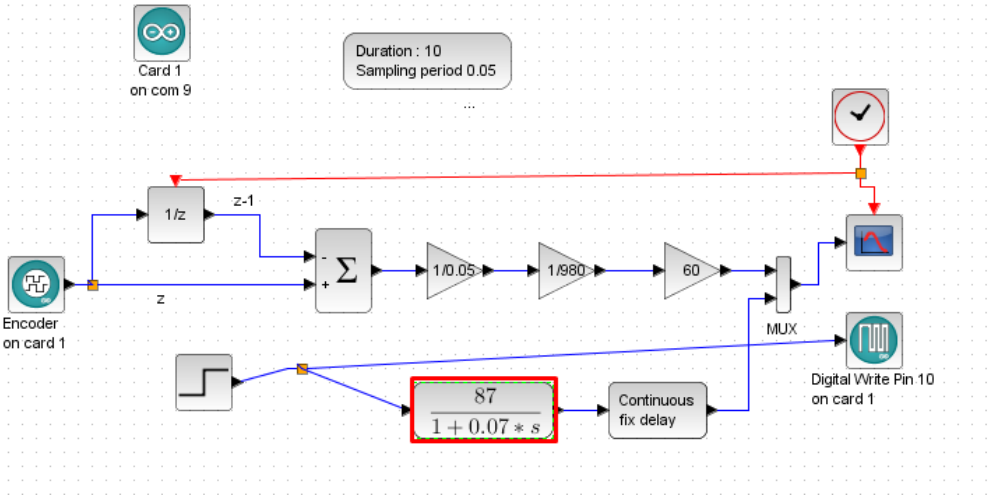


Figura 102. Función de transferencia
Fuente: Realizado por el autor

De lo cual se obtiene la siguiente respuesta:

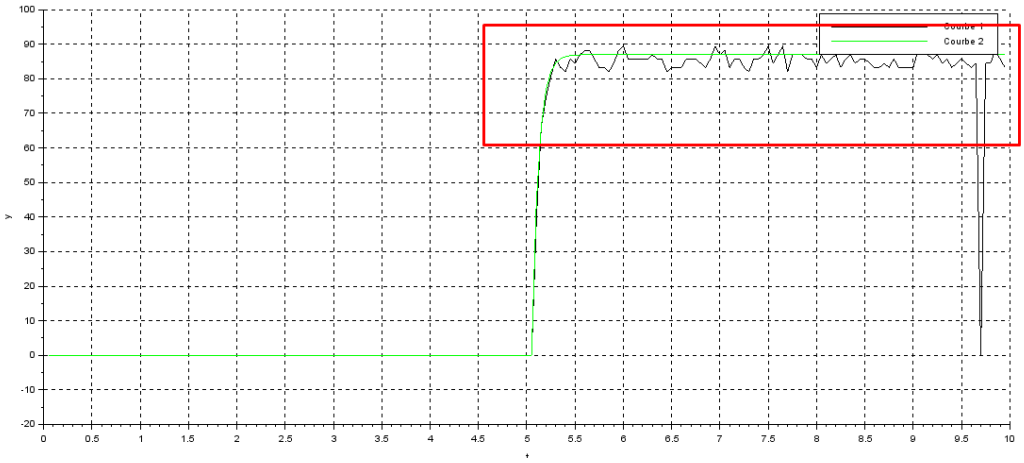


Figura 103. Función de transferencia
Fuente: Realizado por el autor

Como se visualiza tanto la respuesta del motor como la función de transferencia manejan la misma amplitud en su onda de salida, por lo cual se puede definir que el modelado matemático se ha realizado con exito.

Para encontrar el modelo matemático del motor se utiliza un sistema abierto con una señal de entrada que excite la entrada del motor mediante una función unitario, todo esto gracias a scilab y sus herramientas, lo importante al momento de encontrar una función de transferencia es tratar de reducir el cálculo a una ecuación de primer orden asi se simplifica el proceso.

4.2.8. Práctica 8

	**UNIVERSIDAD POLITÉCNICA SALESIANA** **FACULTAD DE INGENIERÍA** **SEDE GUAYAQUIL**

CARRERA	PRÁCTICA N^0	NOMBRE DE LA PRÁCTICA
ELECTRÓNICA	8	Diseño de un control proporcional (P)

OBJETIVOS
Identificar el proceso para la implementación del control proporcional Establecer los controladores y actuadores que intervienen en el proceso Establecer un valor proporcional tomando en cuenta algún tipo de método de estudio planteado para sistemas de control. Elaborar el código que permita que proceso funcione de acuerdo a las estimaciones matemáticas.

EQUIPOS A UTILIZAR	
• Arduino MEGA • Conector USB	• Laptop • Arduino IDLE

• Motor DC • Scilab • Puente H	• Fuente 110 a 24V • Sensor Infrarrojo

INSTRUCCIONES

1) Interconectar el Motor EMG49 hacia el puente H de 43a y hacia el transformador de 120VAC a 24V
2) Conectar el encoder del motor al Arduino MEGA
3) Conectar PWM 9 y 10 del Puente H 43a según se indique
4) Cargar el código con el metodo proporcional
5) Elaborar las pruebas de funcionamiento en la balanza.

DIAGRAMAS

Diagrama de conexión balanza

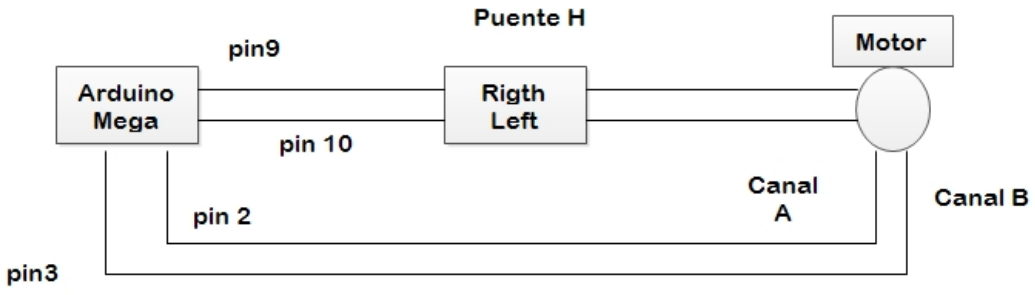


Figura 104. Diagrama conexión Arduino y Balanza
Fuente: Realizado por el autor

PROCESO

1. Para empezar el modelado del sistema es necesario identificar las señales de control, el actuador y la variable de medición para relacionarlas todas.
2. Para el diseño del proceso es necesario estimar las variables de control que en este caso se pueden identificar como la posición de la esfera entregada por el sensor Sharp infrarrojo, además también se tiene como señal de control la posición del motor la cual se recibe por medio del encoder con su canal a y b conectados al pin 2 y 3 respectivamente.
3. El actuador en el caso establecido viene representado por el motor DC, como se pudo observar en la práctica 7 la función de transferencia ya se determinó.
4. Y por último el componente amortiguado proporcional viene determinado por Kp multiplicado por la posición de la esfera indicando que dependiendo donde se encuentre se procederá a enviar un pulso o para mover el motor tanto a la derecha como la izquierda dependiendo de lo solicitado
5. Para identificar las variables se procede a detallar las variables necesarias a continuación:
    a. Posición de la esfera: Es necesario recordar que antes de utilizar por problemas de ruido y resistencia del cableado es necesario calibrar el sensor poniendo como punto de referencia el sitio donde queremos ubicar la esfera. Elaborando una tabla que ubique como punto cero la posición deseada y una escala que permita relacionar la distancia a cada extremo con valores similares como se visualiza en la Tabla 2.

Tabla 2.
Relación sensor y distancia

Lectura sensor	100	140	180	220	260	0	300	392	485	577	670
Relación	-20	-15	-10	-5	0	0	0	5	10	15	20

Fuente: Elaborada por el autor

Para obtener los valores indicados se procede a realizar una lectura del sensor por el puerto Analógico 0 del Arduino y realizando una comparación utilizando el código que se encuentra descrito a continuación:

```
int distancia(){ //método para obtener la distancia
long suma=0; //valor que almacenara el dato obtenido del sensor
for(int i=0;i<25;i++) //Filtrado de ruido con varias muestras por lectura
{
suma=suma+analogRead(A0); //Asignar valor de lectura del sensor y sumarlo al anterior
}
return(suma/25); //se divide el valor obtenido por el número de muestras
}
```

Como se puede observar los valores del sensor se observan en la Figura 105, 106 y 107 respectivamente y sirven como punto medio y máximo que se utilizaran para identificar la posición de la esfera.

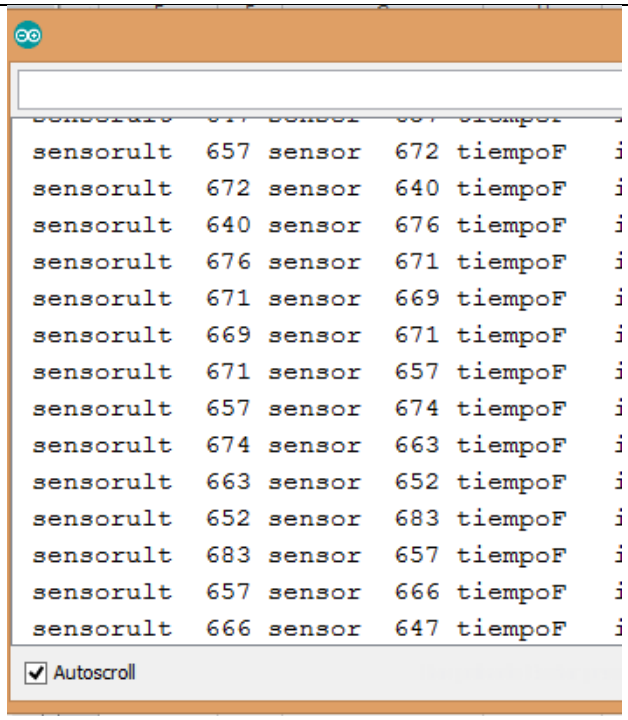


Figura 105. Lectura sensor Derecha
Fuente. Realizado por el autor

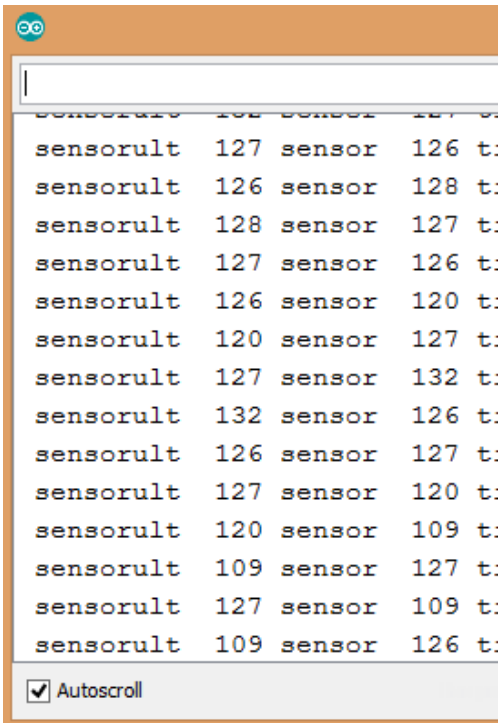


Figura 106. Lectura sensor Izquierda
Fuente. Realizado por el autor

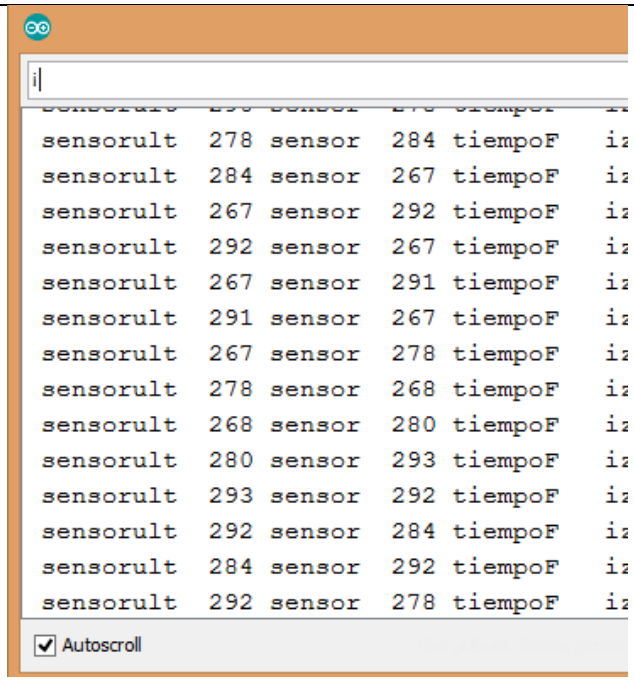


Figura 107. Lectura sensor punto de consigna
Fuente. Realizado por el autor

Considerando como base la tabla 2 se generan vectores que permitirán establecer en base a la medida del sensor la posición dentro de la balanza con la utilización del siguiente código:

```
int calibrando(int sensor3){ //Comparativa entre vectores
 while(i<=11){ //Numero de posiciones en el vector
  int j=i+1; // Variables para desplazarse entre el vector que almacena las distancias
  if((sensor3>sensor[i])&&(sensor3<sensor[j])){ //Establece en que rango se encuentra la lectura
    rango1=Rango[i]; // Asigna en base al rango la distancia con referencia al pinto 0
     return(rango1); //regresa el valor al programa principal y lo imprime
  }
  i++;
```

}

Una vez realizado se puede obtener la lectura del sensor y la posición del motor Dc como se visualiza en la Figura 108.

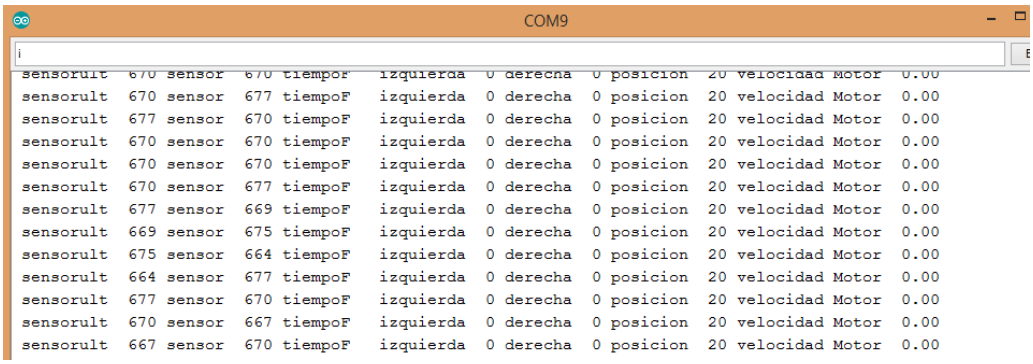


Figura 108. Lectura sensor y posición establecida
Fuente. Realizado por el autor

6. Es importante recordar también que otro punto de consigna es la posición del motor la cual debe mantenerse en 0 cuando la esfera este en la posición 0. Para esto se genera un control en base a interrupciones para contar los pasos que da el motor tanto para arriba como abajo, el cual es:

```
void EncodeA()
{
    if (digitalRead(canalA) == digitalRead(canalB))
    {
      //IsCW = true;
      if (derecha + 1 <= limite){
       derecha++;
       //izquierda--;
      }
    }
     if (digitalRead(canalA) != digitalRead(canalB))
```

```
    {
      //IsCW = true;
      if (derecha + 1 <= limite){
       //izquierda++;
       derecha--;
      }
    }
}
```

Lo cual se entiende que si el motor va hacia la derecha el contador suma y si va a la izquierda resta. Por tal virtud el motor va a tratar por medio del encoder mantener la posición 0 que es la estable mientras la esfera se forzará que llegue al centro.

Es importante elaborar un diagrama del control PID implementado en este caso son dos controles que se concatenan para formar un proceso y sistema de lazo cerrado como se visualiza en la *Figura 109.*

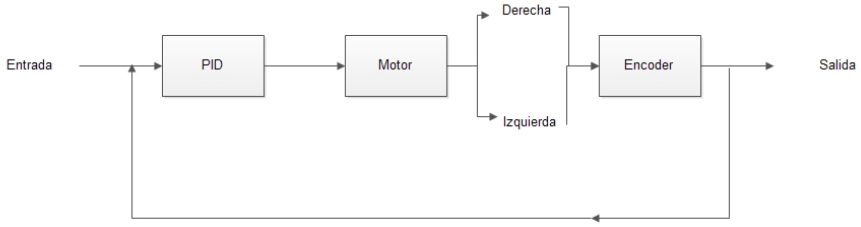


Figura 109. Diagrama proceso encoder
Fuente. Realizado por el autor

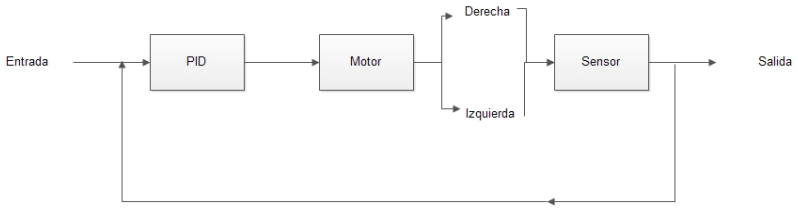


Figura 110. Diagrama proceso sensor
Fuente. Realizado por el autor

Como se visualiza en la Figura 111 y el proceso viene establecido por dos PID

independientes que al sumarse se complementan entre sí.

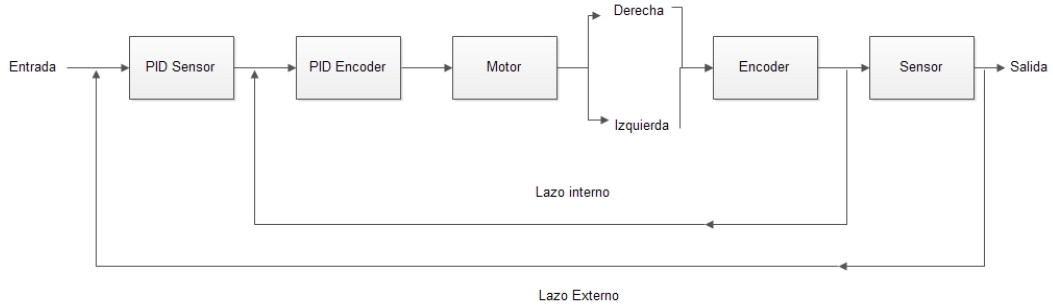


Figura 111. Diagrama proceso completo
Fuente. Realizado por el autor

El método utilizado es un proceso con PID en cascada debido a que se tiene dos variables de control que se identifican como la respuesta del encoder y del sensor, que juntos complementan para formar el control final del proceso.

Una vez expuesto el proceso es necesario identificar el valor proporcional para cada una de los procesos independientes.

Encoder

En el caso del encoder se utilizará el método de asignación de polos para lo cual en la práctica 7 se identificó el modelado matemático se procede a descomponer en base a la función de transferencia los valores necesarios en este caso el valor proporcional para el encoder:

a. Se procede a definir variables necesarias para por medio de asignación de polos encontrar la respuesta solicitada, para obtener un concepto de lo que se necesita, visualizar la Figura 112, identifica el método y las variables que se puede obtener.

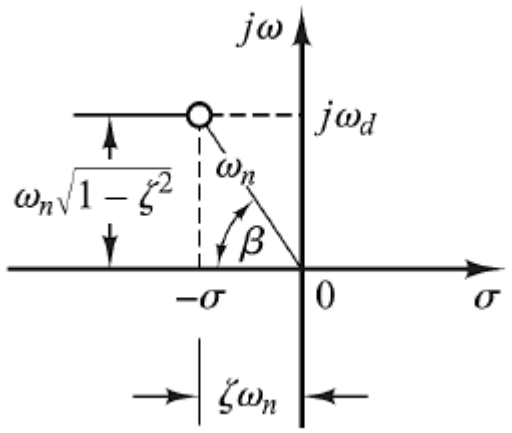


Figura 112. Plano S
Fuente. Realizado por el autor

b. En base al plano s, se tiene que utilizar la ecuación o función de transferencia y encontrar las ecuaciones necesarias que por descomposición simple se muestra a continuación:

$$\frac{Y(s)}{R(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)H(s)}$$

c. De donde se puede identificar que P(s) es la función de transferencia de nuestra planta y C(s) es la función característica del controlador reemplazando se obtiene como resultado:

$$\frac{Y(s)}{R(s)} = \frac{87(KpS + Ki)}{0.07s^2 + (1 + 87kp)s + 87Ki}$$

De lo expuesto es necesario descomponer los valores con ayuda del plano s obteniendo para el valor proporcional lo siguiente:

$$Kp = \frac{1 + 87}{0.07} = 2\sigma$$

$$\sigma = 7$$

$$Kp = 0.011$$

Sensor

Para obtener el valor Proporcional del sensor es necesario realizar una aproximación por método de Ziegler-Nichols en donde se ubican extremos y de ellos se regula a un punto de consigna que permita su funcionamiento, en este caso solo para la porción proporcional se llevó a la conclusión que el valor optimo es 4.2, el mismo que será implementado dentro del código de la manera siguiente:

Proporcional = error x KP

Donde el error es la lectura del sensor cuando esta fuera del punto de consigna.

Para comprobar el funcionamiento es necesario cargar en el Arduino el código que se encuentra en el anexo 17.

De lo que se puede concluir que el sistema trata de estabilizarse pero la esfera no se detiene en el centro de la balanza más bien en uno de sus extremos por lo cual es necesario agregar la componente integral y derivativa.

4.2.9. Práctica 9

	UNIVERSIDAD POLITÉCNICA SALESIANA **FACULTAD DE INGENIERÍA** **SEDE GUAYAQUIL**

CARRERA	PRÁCTICA N^0	NOMBRE DE LA PRÁCTICA
ELECTRÓNICA	9	Diseño de un control Proporcional Integral

OBJETIVOS
Según la identificación control proporcional agregar la porción integral del control Establecer los controladores y actuadores que intervienen en el proceso Establecer un valor integral tomando en cuenta algún tipo de método de estudio planteado para sistemas de control. Elaborar el código que permita que proceso funcione de acuerdo a las estimaciones matemáticas.

EQUIPOS A UTILIZAR	
• Arduino MEGA	• Laptop

• Conector USB • Motor DC • Scilab • Puente H	• Arduino IDLE • Fuente 110 a 24V • Sensor Infrarrojo

INSTRUCCIONES

1) Interconectar el Motor EMG49 hacia el puente H de 43a y hacia el transformador de 120VAC a 24V
2) Conectar el encoder del motor al Arduino MEGA
3) Conectar PWM 9 y 10 del Puente H 43a según se indique
4) Cargar el código con el metodo proporcional intergral
5) Elaborar las pruebas de funcionamiento en la balanza.

PROCESO

1. Revisar el controlador Realizado en la Práctica 8.
2. Dentro de lo que se vio en la parte anterior es necesario agregar el componente integral tanto en el PID del encoder como del sensor.
3. El motor tiene dos entradas de control, todo el proceso de identificación se lo realizo en la práctica 7 y 8.
4. Para identificar el componente integral se lo realiza por separado.

Encoder:

a. En la Práctica 7 se determinó la función de transferencia del motor, mientras que en la práctica 8 el componente proporcional.

b. Para encontrar el componente integral es necesario mantener el método de asignación de polos, el cual se descompone de la siguiente manera utilizando los métodos algebraicos necesarios.

$$\frac{87Ki}{0.07} = \sigma^2 + Wd^2$$

$$Ki = \frac{(7^2 + 7^2)(0.07)}{87}$$

$$Ki = 0.078$$

Sensor:

a. En la práctica anterior se utilizó el método de aproximación de ziegler y Nichols de donde se determinó el componente proporcional, para el componente integral es necesario relacionar la variable por la velocidad de la siguiente forma:

Integral = Kd x Velocidad.

Para calcular la velocidad es necesario realizar una relación de los valores medidos en la distancia inicial – la distancia final de lo que se puede denotar como error actual y error pasado, respectivamente.

Para comprobar su funcionamiento se debe cargar el código que se encuentra en el Anexo 18

Como se puede observar en la balanza la respuesta más estable que en la práctica anterior.

4.2.10. Práctica 10

	**UNIVERSIDAD POLITÉCNICA SALESIANA** **FACULTAD DE INGENIERÍA** **SEDE GUAYAQUIL**

CARRERA	PRÁCTICA N^0	NOMBRE DE LA PRÁCTICA
ELECTRÓNICA	10	Diseño de un control Proporcional Integral y Derivativo

OBJETIVOS
Completar el control PID del proceso en caso de ser necesario para estabilizar la respuesta. Establecer los controladores y actuadores que intervienen en el proceso. Elaborar el código que permita que proceso funcione de acuerdo a las estimaciones matemáticas.

EQUIPOS A UTILIZAR	
• Arduino MEGA • Conector USB	• Laptop • Arduino IDLE

• Motor DC • Scilab • Puente H	• Fuente 110 a 24V • Sensor Infrarrojo

INSTRUCCIONES

1) Interconectar el Motor EMG49 hacia el puente H de 43a y hacia el transformador de 120VAC a 24V
2) Conectar el encoder del motor al Arduino MEGA
3) Conectar PWM 9 y 10 del Puente H 43a según se indique
4) Cargar el código con el método proporcional
5) Elaborar las pruebas de funcionamiento en la balanza.

PROCESO

1. Revisar el controlador Realizado en la Práctica 9.
2. Verificar los valores del proceso y determinar la porción integral para el proceso

Encoder:

a. En la práctica 7, 8 y 9 se tomó en cuenta la función de transferencia y la asignación de polos para la resolución del control del proceso, pero por la misma estimación matemática y pruebas realizadas en scilab, se puede comprobar que un parámetro derivativo dentro del proceso determinado por el encoder solo vuelve inestable y errático el controlador.

Sensor:

a. Para la determinación del valor derivativo se es necesario agregar un valor que incremente mientras más cerca este del punto de consigna de forma que se pueda realizar una aproximación en el punto 0, por ziegler y nichols se necesita hacer una estimación probando con la utilización del código encontrado en el anexo 19, y se asigna valores altos medios y bajos, de los cuales se obtiene que es necesario un valor para Kd = 0.09.

3. Una vez cargado los valores es necesario agregar un botón dentro del interfaz web tanto como del módulo con la pantalla LCD que permita inicializar y finalizar el funcionamiento de la práctica número 10.
4. Además, para cumplir con el propósito de monitorizar el controlador se utilizará la herramienta Serial Ploter de arduino Idle, lo cual se puede visualizar en la Figura 113.

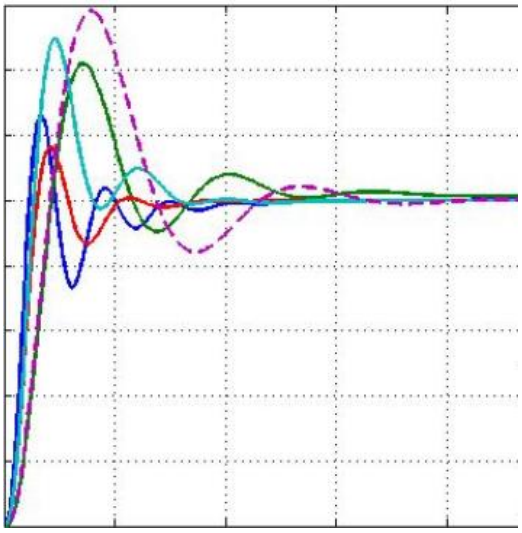


Figura 113. Monitor Serial PID
Fuente. Realizado por el autor

4.3. Análisis de resultados

Para evaluar que tan rentable y beneficioso resulta el desarrollo del proyecto técnico, se procedió a realizar una presentación del módulo didáctico a los alumnos de la materia de teoría de control III, explicando el funcionamiento, ventajas, aplicaciones y limitaciones.

Al concluir la exposición se procedió a realizar una encuesta según el siguiente detalle:

Tabla 3.
Encuesta estudiantes

1. ¿Qué tal le pareció el diseño del módulo didáctico?
Excelente Bueno Regular Malo
2. ¿Qué tan útil le parece las diferentes funciones que posee el módulo didáctico?
Muy útil Útil Inútil
3. ¿Qué tan sencillo le pareció el trabajar una práctica con el módulo didáctico?
Muy difícil Difícil Sencillo Muy sencillo
4. ¿Estaría usted interesado en aprender mucho más de lo explicado en la

exposición acerca del módulo didáctico?
Si No
5. ¿Cómo calificaría la exposición dada por nosotros?
Excelente Buena Regular Mala
6. En su opinión que se debería mejorar al momento de exponer y ¿qué podría mejorar en el módulo didáctico?
…… …… ……

Fuente: Elaborado por el autor

Una vez realizada la encuesta a 20 alumnos de las materias de la materia de, los resultados obtenidos fueron los siguientes:

- Con referencia al diseño del módulo:

  - 18 alumnos lo consideraron excelente.
  - 2 alumnos los consideraron bueno.
  - 0 alumnos lo consideraron regular.
  - 0 alumnos lo consideraron malo.

- Con referencia a la utilidad de las diferentes funciones que posee el modulo:

- 20 alumnos lo consideraron muy útil.
- 0 alumnos los consideraron útil.
- 0 alumnos lo consideraron inútil

- Con referencia a la simplicidad de trabajar con los módulos didácticos:

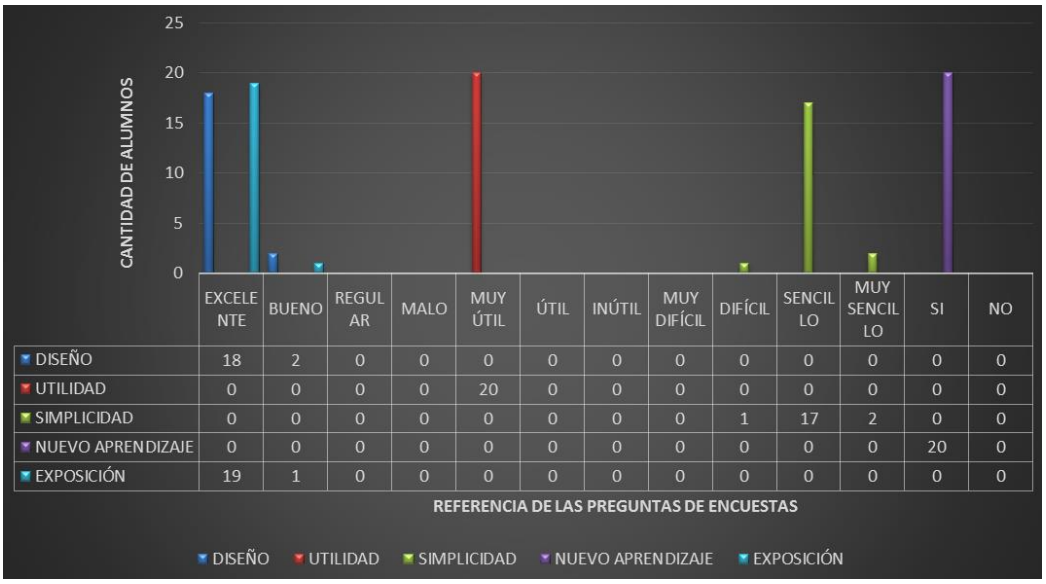
  - 0 alumnos lo consideraron muy difícil.
  - 1 alumnos los consideraron difícil.
  - 17 alumnos lo consideraron sencillo.
  - 2 alumnos lo consideraron muy sencillo.

- Con referencia a si estarían dispuestos a aprender más acerca de los módulos didácticos:

  - 20 alumnos respondieron SI.
  - 0 alumnos respondieron NO

- Con referencia a la exposición:

  - 19 alumnos lo consideraron excelente.
  - 1 alumnos los consideraron buena.
  - 0 alumnos lo consideraron regular.
  - 0 alumnos lo consideraron mala.



	EXCELENTE	BUENO	REGULAR	MALO	MUY ÚTIL	ÚTIL	INÚTIL	MUY DIFÍCIL	DIFÍCIL	SENCILLO	MUY SENCILLO	SI	NO
DISEÑO	18	2	0	0	0	0	0	0	0	0	0	0	0
UTILIDAD	0	0	0	0	20	0	0	0	0	0	0	0	0
SIMPLICIDAD	0	0	0	0	0	0	0	0	1	17	2	0	0
NUEVO APRENDIZAJE	0	0	0	0	0	0	0	0	0	0	0	20	0
EXPOSICIÓN	19	1	0	0	0	0	0	0	0	0	0	0	0

Figura 114. Tabulación Encuesta
Fuente. Realizado por el autor

Figura 115. Exposición Modulo didáctico a estudiantes
Fuente. Realizado por el autor

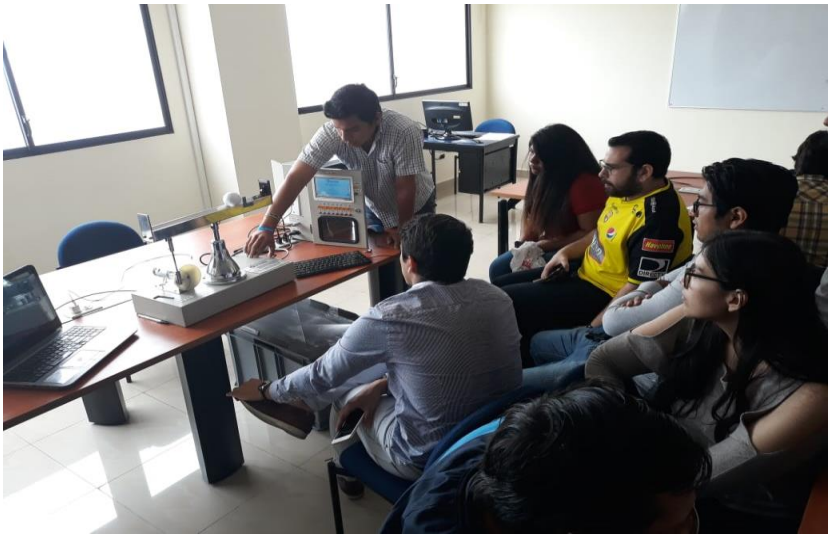


Figura 116. Exposición brazo estabilizador estudiantes
Fuente. Realizado por el autor

4.4. Diseño del módulo didáctico

En el desarrollo del proyecto se tomó en cuenta especificaciones de funcionamiento del sistema, para posterior no tener ningún inconveniente y lograr los resultados óptimos

Los elementos físicos que se diseñaron se detallaran a lo largo del presente capitulo.

4.4.1. Diseño de la balanza en AutoCAD

Se procedió a realizar el diseño de la balanza en la herramienta AutoCAD, con las medidas específicas de los elementos que conforman la balanza, así mismo se puede ver en 3D cada uno de las piezas que a continuación se verán en las figuras respectivas.

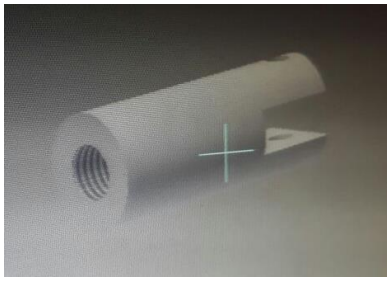


Figura 117. Tubo de conexión entre la balanza el brazo
Fuente: Realizado por el autor

En la figura 117, podemos visualizar la pieza diseñada la cual empata con la base y el brazo de la balanza permitiendo movilidad del cable que se dirige hacia el sensor.

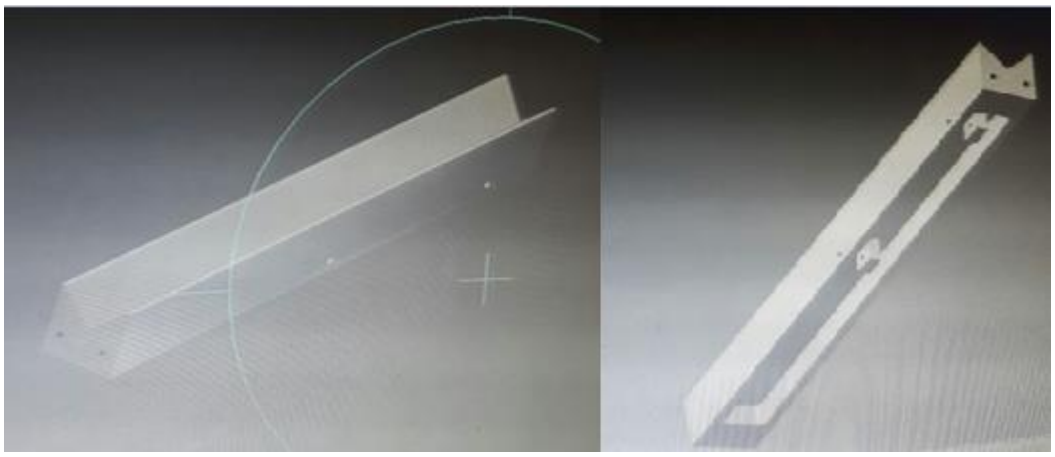


Figura 118. Brazo de la balanza
Fuente: Realizado por el autor

De lo que podemos visualizar en la Figura 118, se elaboro el diseño del brazo que permita el soporte y desplazamiento de la esfera, ademas de tene unas caracteristicas 50cm de largo que para parte práctica permite una longitud de lectura del sensor de 40cm esto debido el rango ciego del sensor infrarrojo seleccionado.

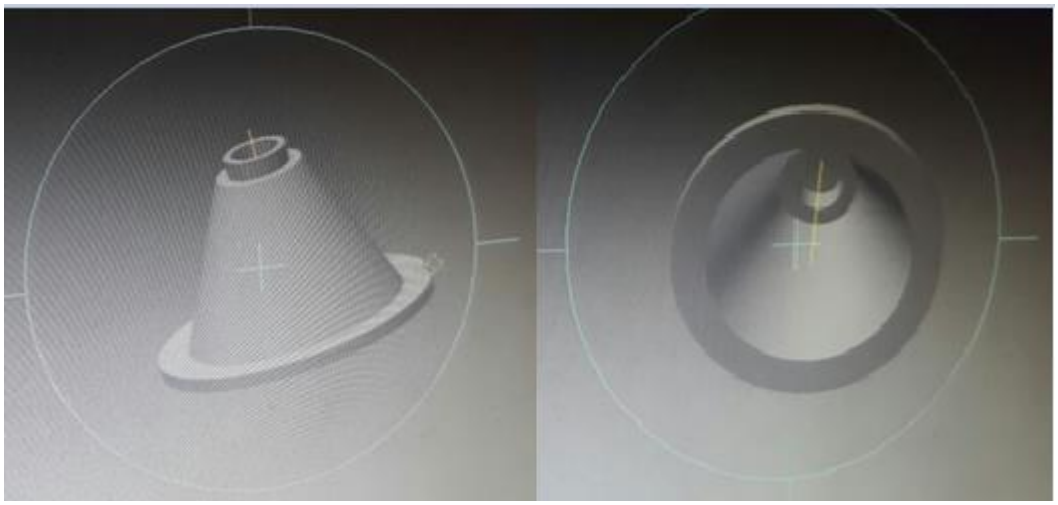


Figura 119. Soporte en forma cónica de la balanza
Fuente: Realizado por el autor

Podemos visualizar en la Figura 119 el diseño conico para la base que permita sostener el peso de la riel mas la esfera y los soportes del sensor, mismo que se ubicara sobre la base de aluminio que contiene todos los módulos necesarios, los cuales por detallar son fuente 110V a 20V, el puente H de 34 amperios y el motor DC.

En la Figura 120 podemos visualizar el diseño de la balanza con las medidas que se espera tener al final del ensamblaje.

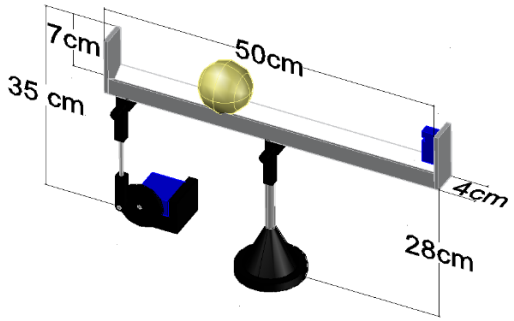


Figura 120. Vista lateral y medidas de la balanza
Fuente: Realizado por el autor

4.4.2. Construcción de la estructura de la balanza

Se procedió a la realización de las piezas de la balanza que anteriormente fueron diseñadas en AutoCAD. Cada una de las partes fue hecha a las medidas requeridas en aluminio.

Para la parte de la base de la balanza se le realizo en acero galvanizado.

Figura 121. Brazo Balanza
Fuente: Realizado por el autor

Podemos visualizar en la Figura 121, la construccion del riel con un abertura en tipo V, el cual sera el brazo de la balanza y por donde pasara la esfera.

Figura 122. Soporte balanza
Fuente: Realizado por el autor

Podemos visualizar en la Figura 122, la base de la balanza de forma cónica.

Figura 123. Brazo balanza finalizado
Fuente: Realizado por el autor

Esta es una pieza de unión entre el brazo y la base de la balanza que se encarga de dar soporte y permitir el paso del cable.

Figura 124. Unión balanza
Fuente: Realizado por el autor

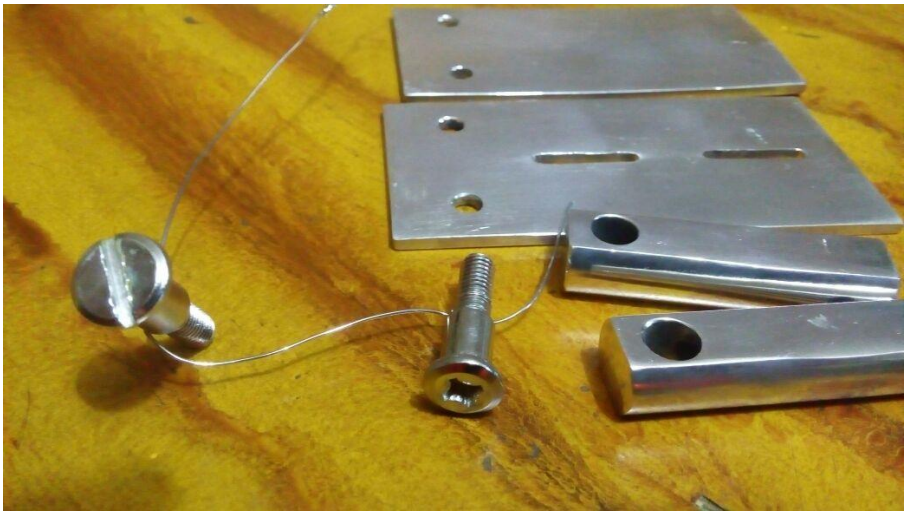


Figura 125. Topes y soportes
Fuente: Realizado por el autor

Podemso visualizar en la Figura 125, los limites que se construyo para cada uno de los bordes del brazo estabilizador y que se conectaran por medio de dos pernos exagonales a la barra principal.

4.4.3. Ensamble de la estructura del brazo estabilizador

Se procede a ensamblar cada una de las partes previamente elaboradas, de esta manera obtenemos nuestro brazo estabilizado ensamblado figura 126.

Figura 126. Ensamblaje balanza
Fuente: Realizado por el autor

Para el ensamble de nuestro brazo estabilizador tendremos como base las estructuras previamente fabricadas Figura 127.

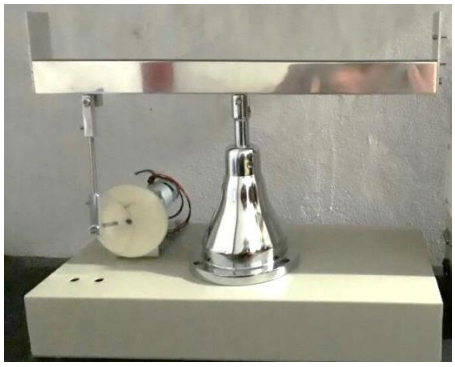


Figura 127. Estructura Balanza Finalizada
Fuente: Realizado por el autor

Los elementos usados para la implementación de nuestro brazo estabilizador son los que se presentan en la Tabla 4.

Tabla 4.
Elementos utilizados en brazo estabilizador

Ítem	Equipos
1	Motor Dc 24V 13A, Motor /Encoder EMG49
2	Sharp IR Range Sensor - 10cm to 80cm
3	Drive Motor DC
4	Fuente de poder 24V – 13 A
5	Cable eléctrico #18 flexible – metros
6	Canaleta Ranurada 25x25mm

7	Baquelitas para distribución
8	Conector JACK-PODER-3IN1
9	Conector DB25

Fuente. Realizado por el autor

4.4.4. Colocación de los elementos en el brazo estabilizador

Se colocaron los conectores DB25 y JACK-PODER-3IN1 como se muestra en la Figura 128.

Figura 128. Conectores Balanza
Fuente: Realizado por el autor

Se realiza la instalacion del motor Dc 24V 13A /Encoder EMG49 y del Sensor Sharp IR Range como se muestra en la Figura 129.

Figura 129. Sensor y motor balanza
Fuente: Realizado por el autor

Se realizo la instalacion de fuente de poder 24V – 13 A y del Drive Motor DC, la conexión interna de cada de los elementos electrónicos y se ordenó el cableado usando una canaleta Ranurada 25x25mm como se muestra en la Figura 130.

Figura 130. Estructura interna modulo balanza
Fuente: Realizado por el autor

Finalizada la conexión interna del módulo se procede a energizarlo para examinar el correcto funcionamiento del mismo y se procede con el etiquetado de cada uno de los elementos del módulo como se muestra en la Figura 131.

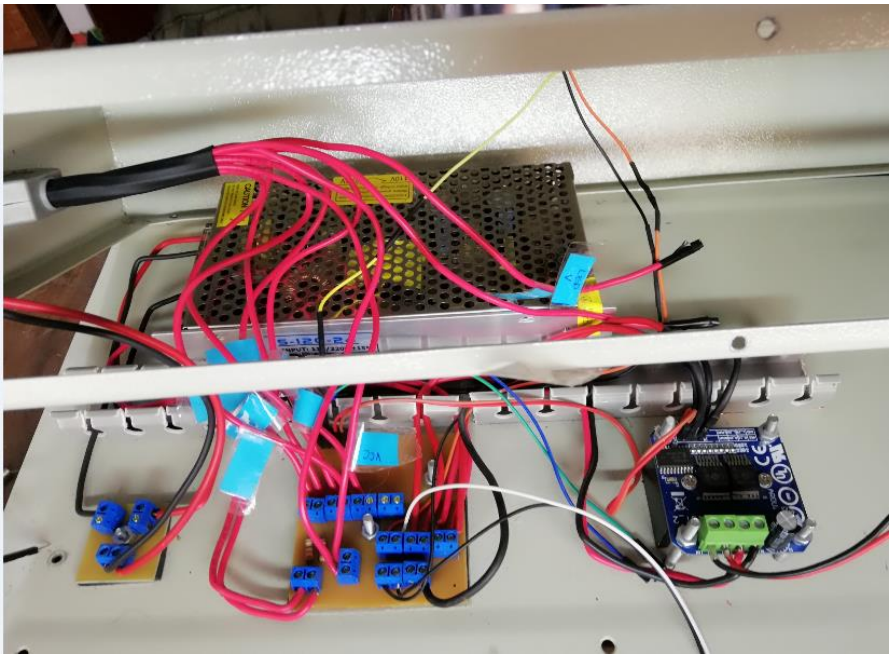


Figura 131. Cableado y etiquetado Balanza
Fuente: Realizado por el autor

4.4.5. Diseño y construcción de baquelitas

Para la distribución interna del nuestro módulo de control se procedió con el diseño de las baquelitas en el programa Eagle

En la figura 132 se puede observar la baquelita para a la distribución de los pines de nuestra Raspberry la cual permitirá controlar las entradas y salidas de nuestro módulo de control, Anexo 20.

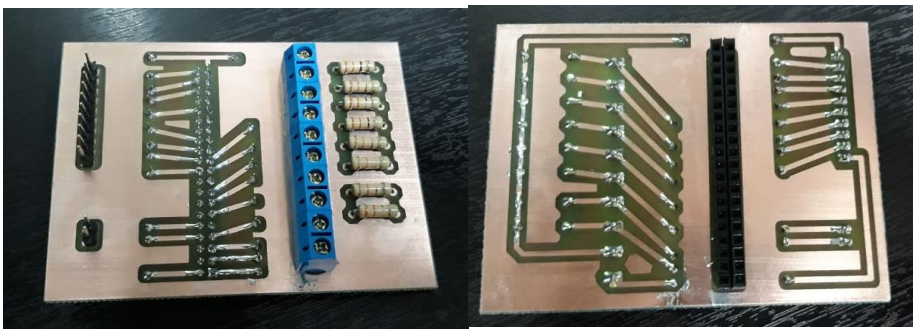


Figura 132. Baquelita distribución de pines
Fuente: Realizado por el autor

En la figura 133 se observa la baquelita de distribución interna para la alimentación de nuestras luces pilotos de 120V. Anexo 20.

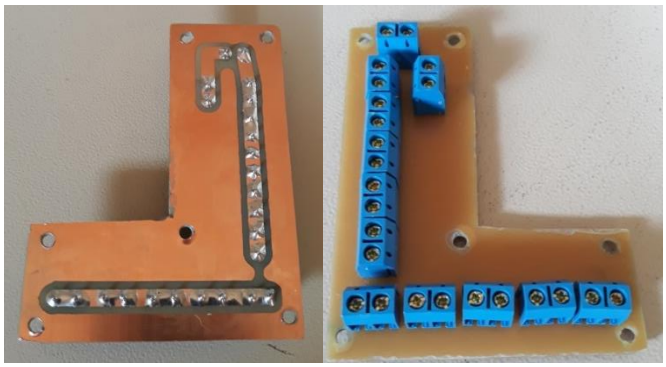


Figura 133. Baquelita alimentación
Fuente: Realizado por el autor

En la figura 134 se visualiza la baquelita de distribución interna para la alimentación de 120V. Anexo 20.

Figura 134. Baquelita alimentación
Fuente: Realizado por el autor

4.4.6. Ensamble del módulo didáctico

Para el ensamble de nuestro modulo tendremos como base las estructuras previamente fabricadas Figura 135.

Figura 135. Modulo luces
Fuente: Realizado por el autor

Los elementos usados para la implementación de nuestro módulo de control son los que se exponen en la Tabla 5.

Tabla 5.
Elementos utilizados en modulo luces

Ítem	Equipos
1	Raspberry Pi 3 Model B Motherboard
2	SanDisk Ultra 32GB microSDHC UHS-I Card
3	Arduino/Genuino Mega 2560
4	Pantalla LCD táctil de 7 pulgadas (B)
5	Módulo Relay 2CH 5VDC
6	Módulo Relay 8CH 5VDC
7	BL-WR2000 300Mbps Wireless N Router
8	Baquelitas para distribución
9	Conector HDMI en L
10	Cable HDMI
11	Conector JACK-PODER-3IN1

12	Conector DB25
13	Conector RJ45
14	Conector HDMI en L
15	Luces piloto 120V
16	Switch On/Off
17	Pulsador abiertos
18	Cable eléctrico #18 flexible
19	Conector micro USB
20	Canaleta Ranurada 25x25mm
21	Cable USB Tipo A -Tipo B

Fuente: Realizado por el autor

4.4.7. Colocación de los elementos en el módulo de control.

Se colocaron los conectores DB25, RJ45 y JACK-PODER-3IN1 como se muestra en la Figura 136.

Figura 136. Conectores módulo didáctico
Fuente: Realizado por el autor

Se colocaron las luces piloto de 120V, los Switch On/Off y los pulsadores como se muestra en la Figura 137.

Figura 137. Luces y botones módulo didáctico
Fuente: Realizado por el autor

Se realizo la instalacion de la pantalla LCD táctil de 7 pulgadas (B) y la conexión de cada de los elementos electrónicos colocados en la tapa frontal de nuestro modulo. Figura 138.

Figura 138. Instalación pantalla LCD
Fuente: Realizado por el autor

Finalizado la conexión de todos elementos con el cable eléctrico #18 flexible se procede a ordenarlos correctamente usando amarras plásticas como se muestra en la Figura 139.

Figura 139. Cableado interno
Fuente: Realizado por el autor

Se realizó la distribución interna del modelo usando la Raspberry Pi 3, Arduino Mega 2560, relay de 8 y 2 canales, la baquelita de distribución y la canaleta ranurada de 25x25mm como se muestra en la Figura 140.

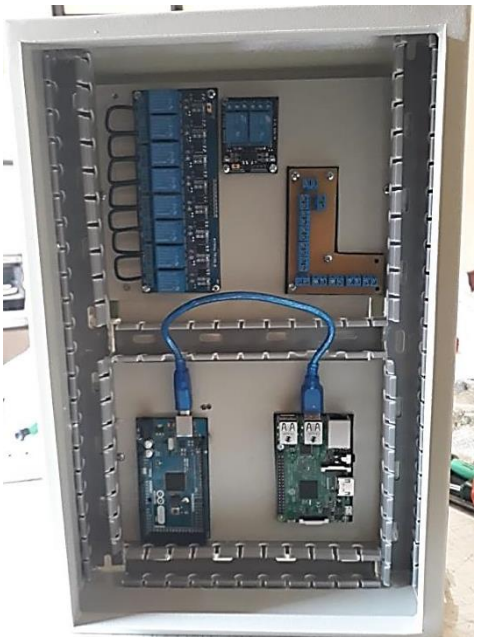


Figura 140. Distribución módulos
Fuente: Realizado por el autor

Una vez realizada la distribucion de los elementos dentro del modulo se procede con la conexión interna como se muestra en al Figura 141, tanto de la parte frontal como posterior del modulo.

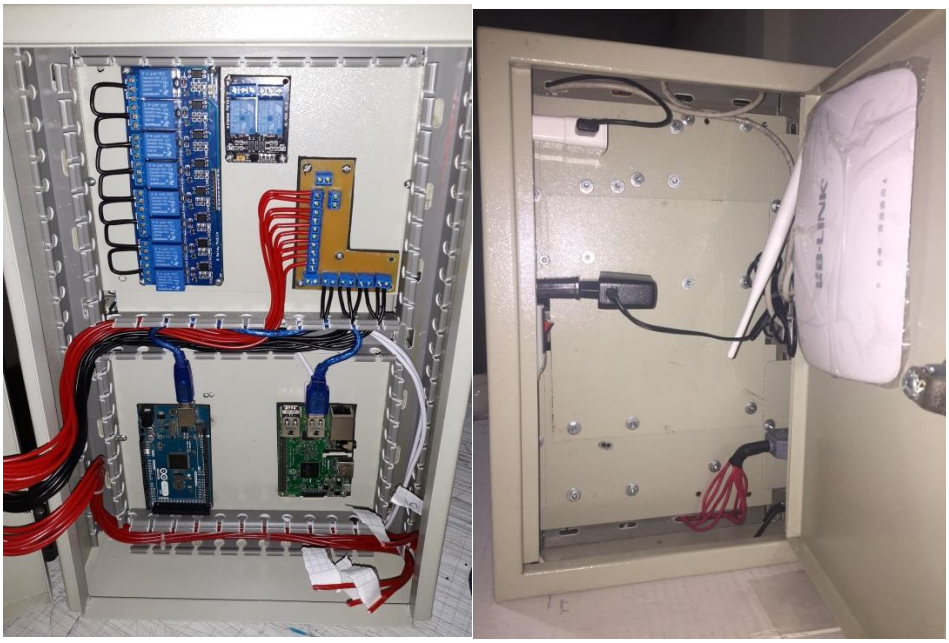


Figura 141. Distribución interna del modulo
Fuente: Realizado por el autor

Finalizada la conexión interna del módulo se procede a energizarlo para verificar el correcto funcionamiento Figura 142 y se procede con el etiquetado de cada uno de los elementos del módulo como se muestra en la Figura 142 y 143.

Figura 142. Etiquetado externo modulo
Fuente: Realizado por el autor

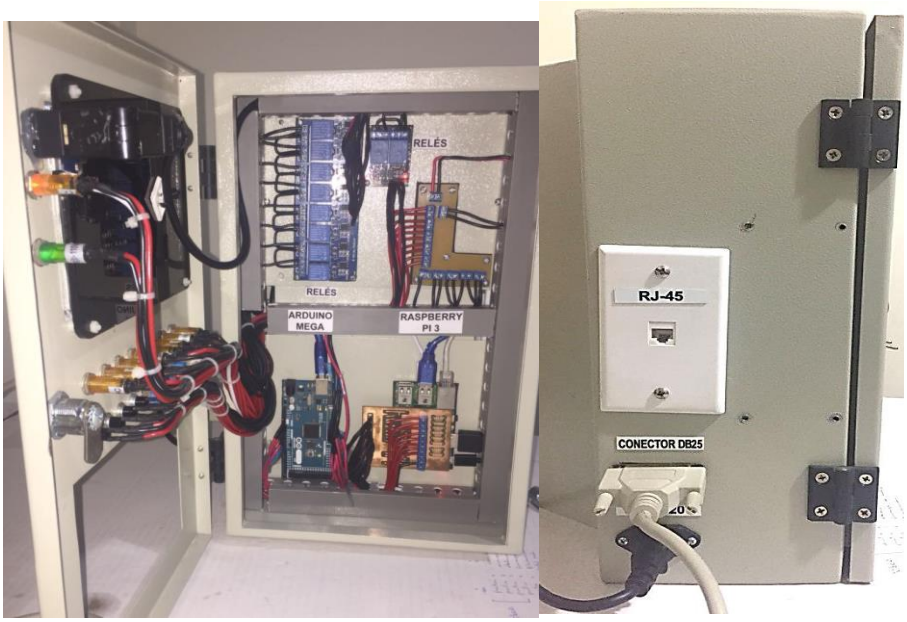


Figura 143. Etiquetado interno módulo didáctico
Fuente: Realizado por el autor

FUNCIONALIDAD

El presente proyecto exhibe un aporte importante para la carrera de ingeniería electrónica y la comunidad universitaria, porque logra integrar los conocimientos recibidos en materias de control automático impartidas por los docentes y demuestra de manera teórica – práctica, los diferentes tipos de sistemas de control analizadas en clase.

Dentro de cada una de sus prácticas existen conocimientos generales y específicos acerca de varios temas como pueden ser: electrónica, control, programación básica e intermedia, cableado, diseño de circuitos y manejo de microcontroladores.

Además, se puede destacar que el módulo didáctico no se ve limitado por las diez prácticas implementadas y permite que el usuario final sea tanto el estudiante como el docente pueda generar prácticas nuevas en base a las necesidades cambiantes que la temática requiera.

Dentro del funcionamiento del módulo se encuentran practicas interactivas desarrolladas en base a software y hardware libre que permiten visualizar las ventajas presentadas por los módulos raspberry y arduino, además de su interacción entre ellos para el control de funciones tanto de entrada y salida e interacción con el usuario final de una manera interactiva y didáctica de aprendizaje.

INNOVACIÓN

El proyecto de titulación abarca además de métodos de control de procesos que se utilizan en la industria, también tecnologías de la información que permiten interactuar con el usuario final de manera remota, cada una de estas prácticas permite al usuario final comprender el funcionamiento de un interfaz web que interactúa con un módulo didáctico electrónico, aplicando programación, electrónica, matemática y manejo de microcontroladores además de sensores y actuadores.

La tecnología sigue avanzando de manera constante por lo cual es necesario que los proyectos electrónicos se actualicen en base a las necesidades mercado laboral y educativo, por lo cual se utiliza microcontroladores robustos los cuales son arduino y raspberry, que no se ven limitados por el tipo de aplicación y son además compatibles con todo tipo de módulos para recibir toda clase de señales tanto analógicas como digitales según la situación lo amerite.

IMPACTO

El presente proyecto aporta tecnológicamente al laboratorio de control automático, con un módulo didáctico que permite a los estudiantes que cursan la carrera de ingeniería electrónica, mejor la comprensión y análisis de los sistemas de control automático, de este modo forman un sistema que converge entre varios lenguajes de programación y prácticas de nivel básico, intermedio y avanzado.

El impacto se refleja en la mejora del aprendizaje de las enseñanzas impartidas en clase por medio de prácticas interactivas dirigidas al estudiante y docente, además de generar curiosidad académica para encaminar las aspiraciones formativas de los futuros profesionales.

La integración de las prácticas por medio de un interfaz web que le permita visualizar al usuario final ciertos aspectos de lo realizado y de esta forma también controlar ciertas señales que interactúan con el módulo didáctico.

CONCLUSIONES

El diseño e implementación del actual proyecto es un aporte muy importante para los estudiantes de la Universidad Politécnica Salesiana Sede Guayaquil en la carrera de Ingeniería Electrónica.

Como se redactó en la documentación, este proyecto está basado en RASPBERRY PI 3 y ARDUINO MEGA para control y monitoreo remoto de un brazo estabilizador, de las cuales la Raspberry Pi3 se encargará de controlar los puertos

Y el Arduino Mega se encargará de la captura y análisis de datos recibidos del sensor Sharp para tomar las decisiones correctas que permitan estabilizar la esfera.

Arduino permite manejar procesos de nivel industrial de manera económica y flexible para todo tipo de aplicativos.

El control PID es una herramienta útil al momento de mejorar la precisión de un proceso para obtener una salida deseada con cualquier tipo de motor, permitiendo realizar tareas de automatización tanto en el campo educativo como industrial.

RECOMENDACIONES

Es importante mantener una correcta conexión de puertos sensores y actuadores dentro de la estructura del módulo didáctico y de esta forma asegurar que todos las practicas funcionen de manera adecuada.

Para mejorar la estabilidad del servidor web es recomendable utilizar la conexión hacia la red local o internet por medio del conector Rj-45 hacia un puerto Ethernet con acceso a la red.

El mantenimiento preventivo debe ser realizado cada seis meses debido a que los componentes electrónicos pueden verse afectados por la utilización constante.

Verificar la alimentación de cada una de las partes del módulo didáctico se encuentren conectados correctamente según el diagrama.

BIBLIOGRAFÍA

Adria, J. (2012, Mayo 28). *Raspberry Pi on stage*. From Joel Adria Technical Services: http://joeladria.com/posts/raspberry-pi-on-stage/

Arduinga!, C. (2012). *Principales comandos (C/C++) de Arduino.* From Taringa: https://www.taringa.net/comunidades/arduinga/4518720/Principales-comandos-C-C-de-Arduino.html

Arduino. (2017). *Arduino Mega.* From Arduino: https://www.arduino.cc/en/Main/arduinoBoardMega

Castro, A. (2014, Enero 23). *¿Qué es Raspberry Pi, dónde comprarla y cómo usarla?* From ComputerHoy: http://computerhoy.com/noticias/hardware/que-es-raspberry-pi-donde-comprarla-como-usarla-8614?amp=

CAVSI. (2015, Noviembre 03). *¿Qué es Puerto Ethernet? Definición*. From http://www.cavsi.com/preguntasrespuestas/que-es-puerto-ethernet/

Diaz, J. M., & Escobar, L. (2014). *TECNOLOGÍA ETHERNET*. From UNIVERSIDAD DEL VALLE : http://www.galeon.com/redesljd/ethernet.htm

dvioleau. (2013). *forge.scilab.org*. Retrieved Mayo 12, 2018 from https://forge.scilab.org/index.php/p/arduino/source/tree/61//macros/toolbox_arduino_v3/toolbox_arduino_v3.ino

Isaac PE. (2015, Mayo 26). *Todo sobre los GPIO Raspberry Pi*. From https://comohacer.eu/gpio-raspberry-pi/

Klosowski, T. (2016, Marzo 03). *El rendimiento de la Raspberry Pi 3, comparado contra los modelos anteriores*. From Gizmodo: http://es.gizmodo.com/el-rendimiento-de-la-raspberry-pi-3-comparado-contra-l-1762657409

Luz, S. d. (2016, Septiembre 24). *Recopilatorio de Raspberry Pi: sistemas operativos, herramientas software y mods*. From RedesZone: https://www.redeszone.net/2016/09/24/recopilatorio-raspberry-pi-sistemas-operativos-herramientas-software-mods/

Moreno, S. F. (2015, Agosto 25). *¿Qué es HDMI? ¿Qué significa y para qué sirve?* From Valor TOP: http://www.valortop.com/blog/que-es-hdmi-que-significa-y-para-que-sirve

NayLampmechatronics. (2015). *Sensor Infrarrojo de distancia SHARP GP2Y0A21*. From NayLampmechatronics: http://www.naylampmechatronics.com/sensores-proximidad/203-sensor-infrarrojo-de-distancia-sharp-gp2y0a21.html

Pernía, M. A. (2011, Septiembre). *Conceptos Básicos de Máquinas de corriente continua.* From Universidad Nacional Experimental del Táchira: https://es.scribd.com/document/70376756/Conceptos-Basicos-de-Motores-DC

Profesional, B. (2016, Febrero 29). *RaspBerry Pi 3, el ordenador más barato.* From Revista Digital: http://bibliotecaprofesional.com/raspberry-pi-3-el-ordenador-mas-barato/

Ramírez, J. L. (2014, Agosto 14). *15 comandos para Raspberry Pi que todo el mundo debería conocer*. From Rootear: https://rootear.com/ubuntu-linux/15-comandos-para-raspberry-pi

Raspberry. (2012). *Rasoberry* . Retrieved Mayo 3, 2018 from https://www.raspberryshop.es/

Robótica, W. d. (2016). *Sensor de infrarrojos*. From Wiki de Robótica: http://wiki.robotica.webs.upv.es/wiki-de-robotica/sensores/sensores-proximidad/sensor-infrarrojos/

shop, R. (2017, Diciembre 17). *Raspberry shop*. From RASPBERRY SHOP: https://www.raspberryshop.es/raspberry-pi-3.php

SuperRobotica.com. (2017, Enero 5). *MOTOR DC REDUCTOR 24V 122 RPM CON ENCODER EMG49.* From Motores de Corriente Continua: http://www.superrobotica.com/S330105.htm

Ulloa, M. U. (2015, 03 15). *Controlador Lógico Programable PLC*. From Control Real Español: http://controlreal.com/es/controlador-logico-programable-plc/

Vis, P. J. (2014). *Conector DSI*. From Raspberry_PI: https://www.petervis.com/Raspberry_PI/Raspberry_Pi_LCD/Raspberry_Pi_LCD_DSI_Display_Connector.html

ANEXOS

Anexo 1 Planos Eléctricos

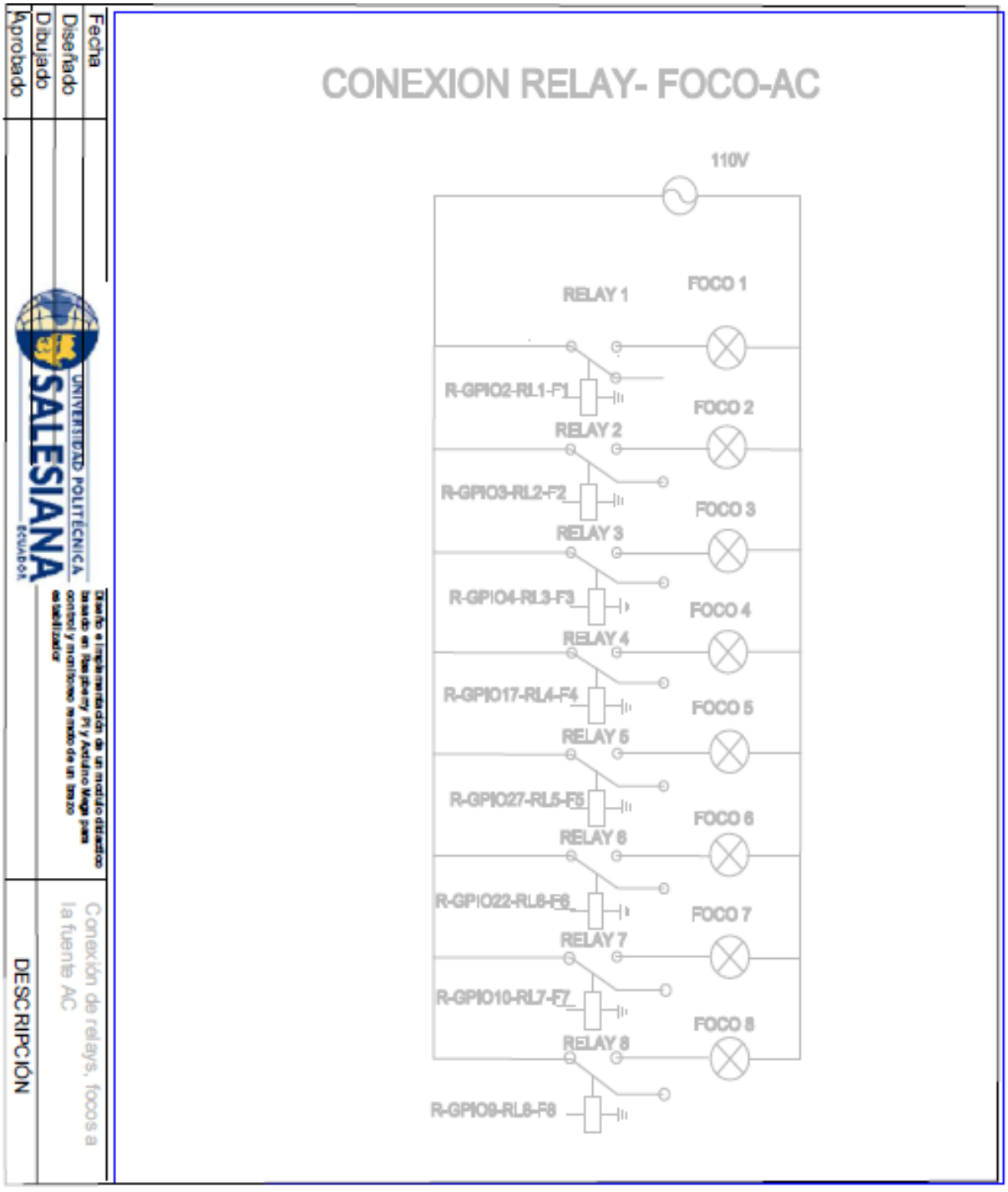


Figura 144. Diagrama Relays
Fuente: Realizado por el autor

Figura 145. Simbología Eléctrica
Fuente: Realizado por el autor

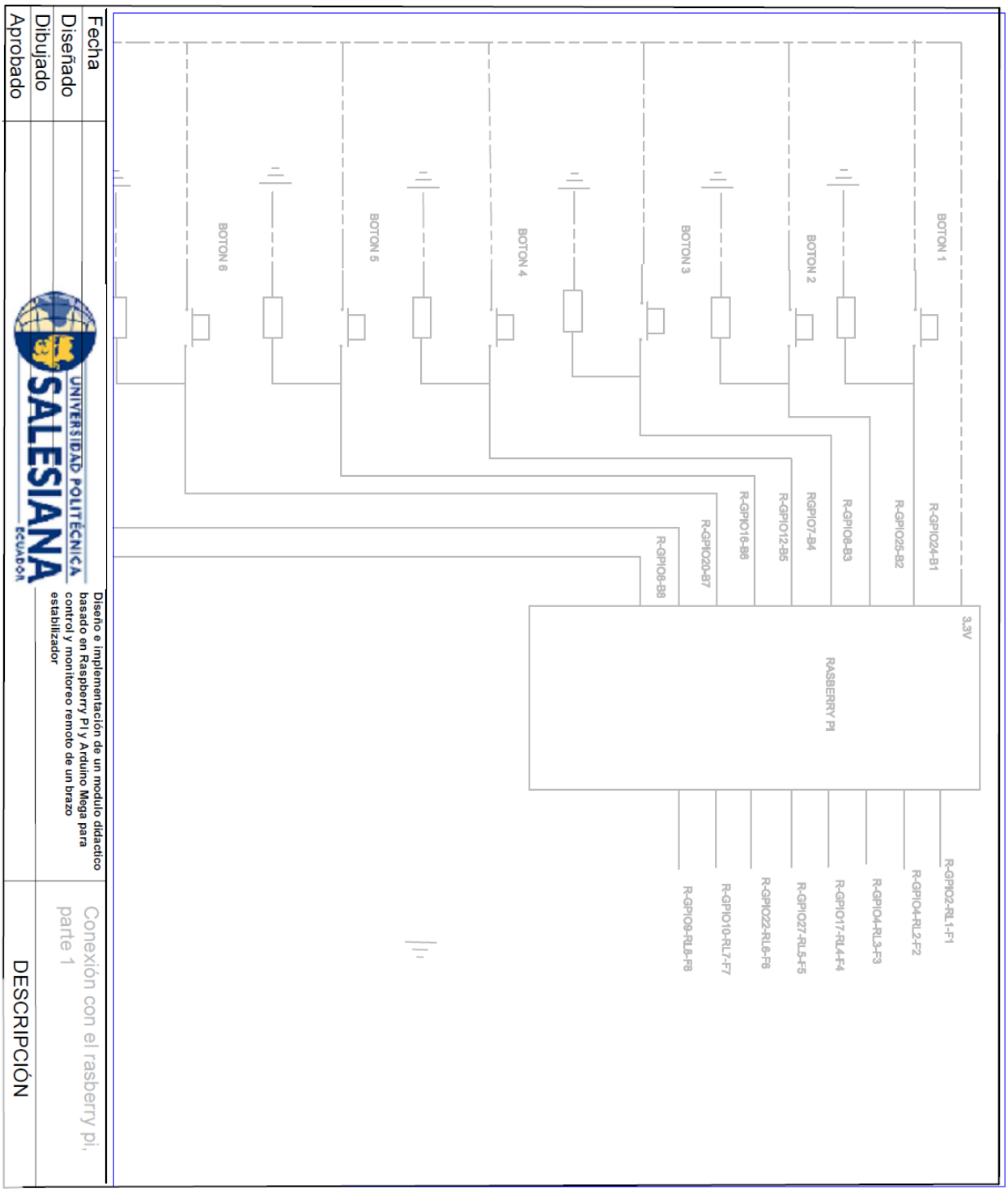


Figura 146. Conexión raspberry
Fuente: Realizado por el autor

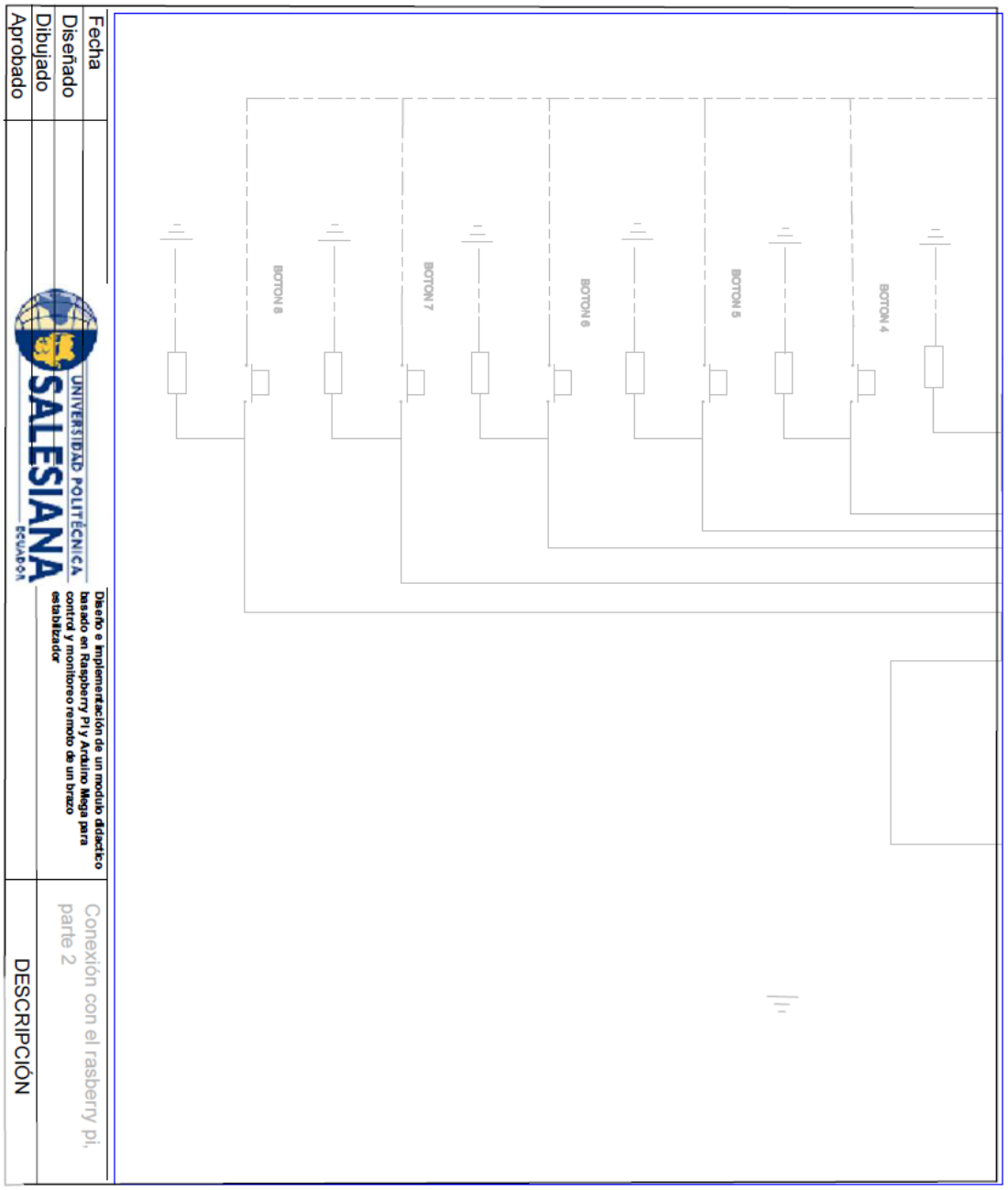


Figura 147. Diagrama botones
Fuente: Realizado por el autor

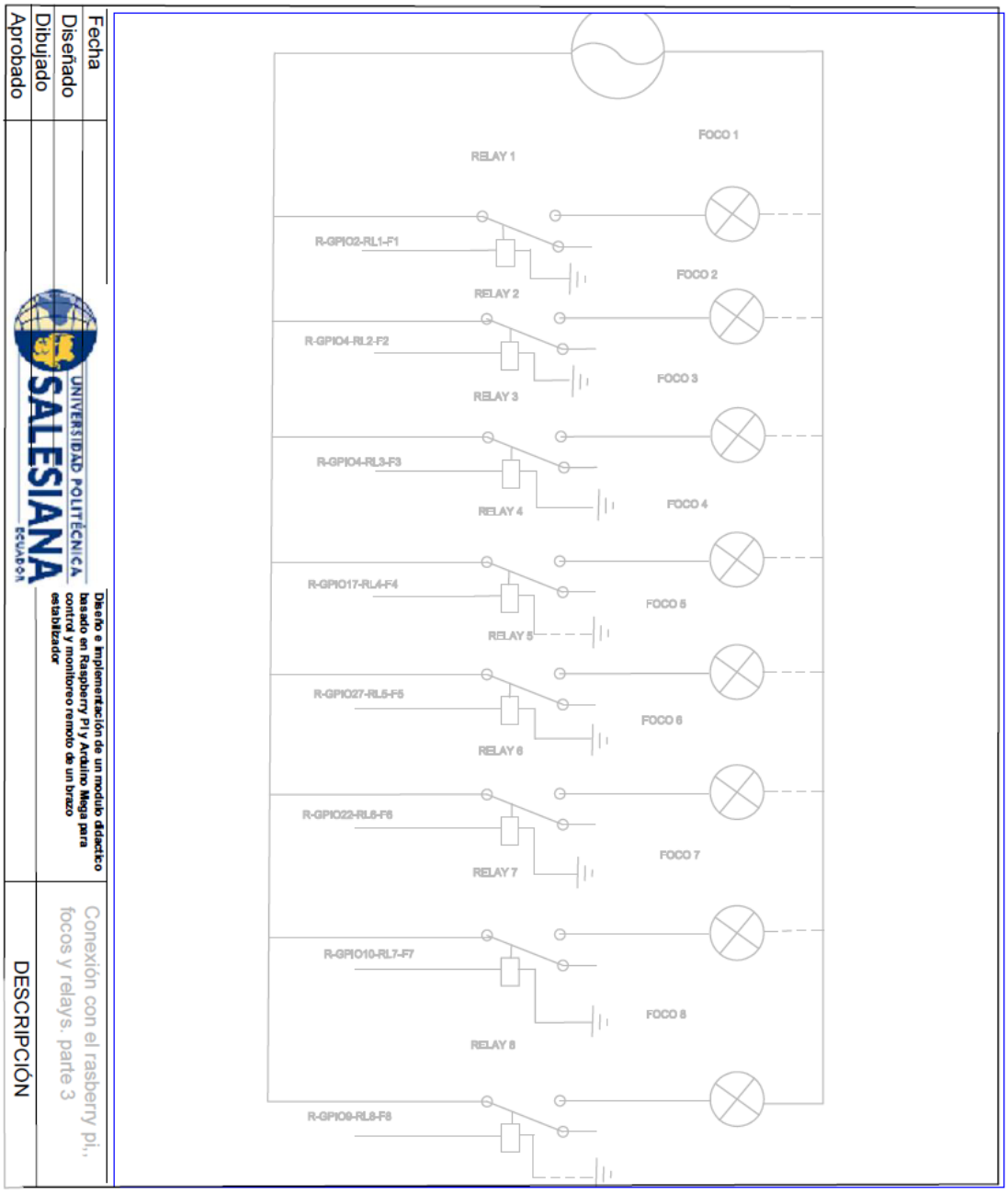


Figura 148. Diagrama relays-focos
Fuente: Realizado por el autor

Anexo 2 Presupuesto

Tabla 6.
Presupuesto

Ítem	Equipos	Cant.	Costo unitario	Costo total
1	Raspberry Pi 3 Model B Motherboard	1	$ 60,00	$ 60,00
2	SanDisk Ultra 32GB microSDHC UHS-I Card with Adapter, Grey/Red, Standard Packaging (SDSQUNC-032G-GN6MA)	1	$ 15,00	$ 15,00
3	Arduino/Genuino Mega 2560 R3 con 56 E/S digital y 16 entradas analógicas	1	$ 25,00	$ 25,00
4	Motor Dc 24V 13A, Motor /Encoder EMG49	1	$ 107,00	$ 107,00
5	Sharp IR Range Sensor - 10cm to 80cm W/ Cable M: GP2Y0A21YK0F	1	$ 20,00	$ 20,00
6	Pantalla LCD táctil de 7 pulgadas (B) Interfaz HDMI Raspbian Angstrom para Raspberry Pi B / B + / 2 B / 3 B / 3B + BB Plátano negro Pi	1	$ 62,00	$ 62,00
7	Módulo Relay 2CH 5VDC	1	$ 5,00	$ 5,00
8	Módulo Relay 8CH 5VDC	1	$ 18,00	$ 18,00
9	BL-WR2000 300Mbps Wireless N Router	1	$ 15,00	$ 15,00
10	Drive Motor DC	1	$ 24,00	$ 24,00
11	Diseño e impresión de baquelitas	6	$ 10,00	$ 60,00
12	Diseño y construcción del brazo estabilizador	1	$ 500,00	$ 500,00
13	Diseño y construcción de tablero metálico	1	$ 200,00	$ 200,00
14	Conector HDMI en L	2	$ 1,00	$ 2,00
15	Luces piloto 110V	8	$ 0,60	$ 4,80
16	Switch On/Off	4	$ 0,30	$ 1,20
17	Pulsador abiertos	4	$ 0,30	$ 1,20
18	Fuente de poder 25v	1	$ 25,00	$ 25,00
19	Cable eléctrico #18 flexible - metros	60	$ 0,60	$ 36,00
20	Conector micro USB	1	$ 2,00	$ 2,00
21	Cable paralelo DB25 macho	1	$ 7,00	$ 7,00
22	Cable HDMI	1	$ 5,00	$ 5,00
23	Materiales y herramientas varios	1	$ 150,00	$ 150,00
24	Capacitación Raspberry y arduino	1	$ 300,00	$ 300,00
25	Asesoría Técnica	1	$ 150,00	$ 150,00
			TOTAL	**$ 1.795,20**

Fuente: Realizado por el autor

Anexo 3 Portada interfaz web

Figura 149. Portada Raspberry
Fuente: Realizado por el autor

Anexo 4 Portada interfaz web manejo de puertos salida

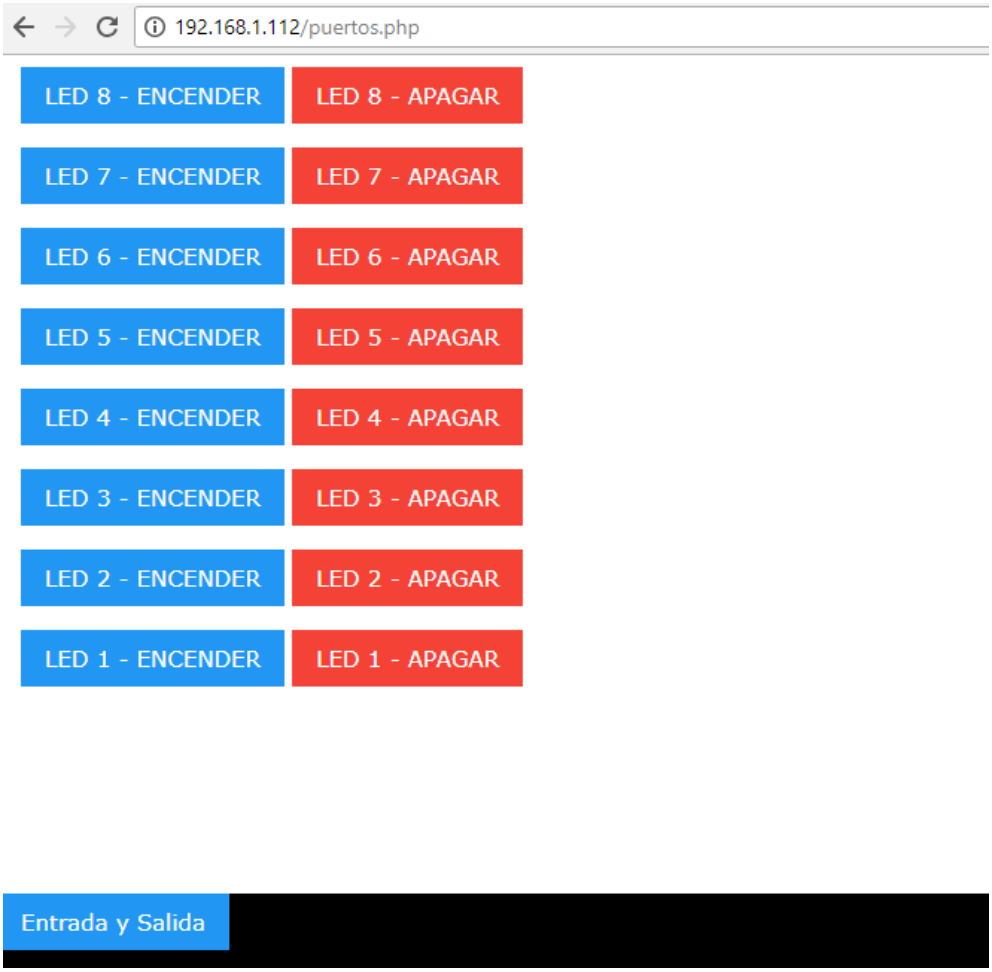


Figura 150. Interfaz manejo entradas y salidas
Fuente: Realizado por el autor

Anexo 5 Diseño baquelita eagle

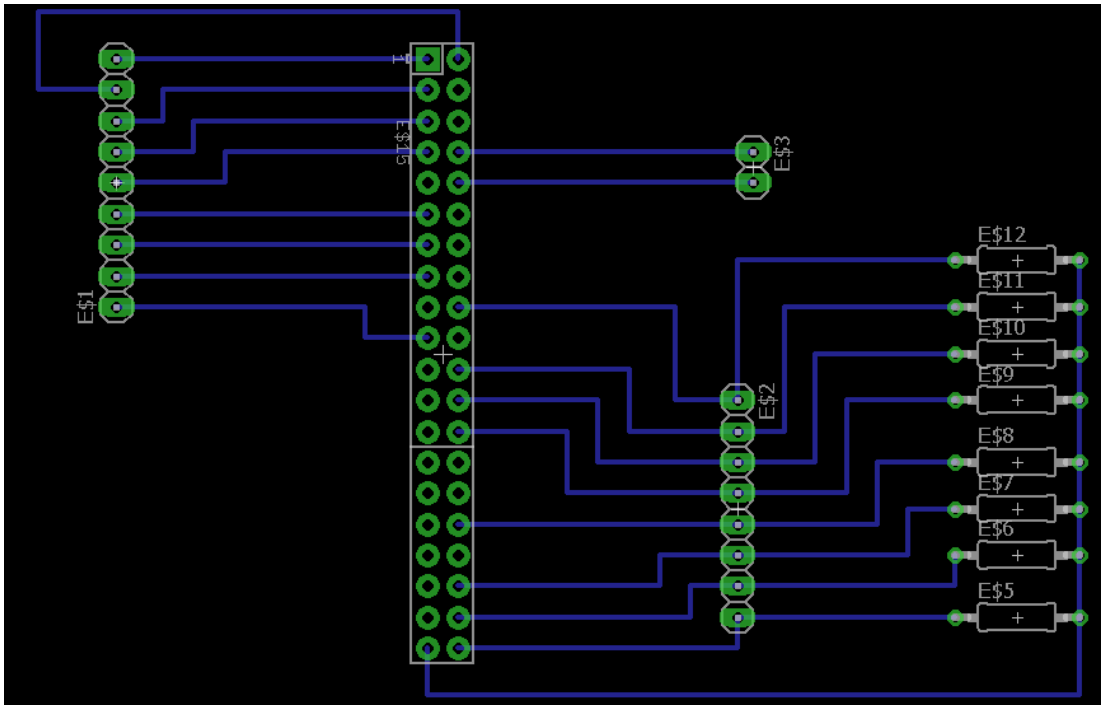
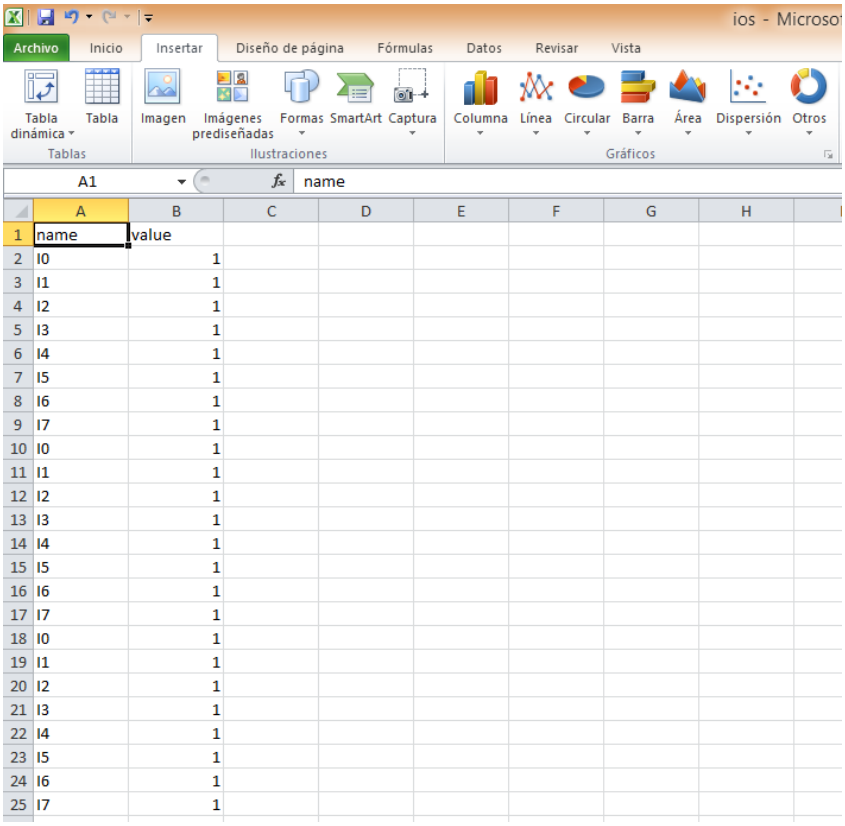


Figura 151. Diseño raspberry baquelita
Fuente: Realizado por el autor

Anexo 6 Documento base de datos exportada a Excel



	A	B
1	name	value
2	I0	1
3	I1	1
4	I2	1
5	I3	1
6	I4	1
7	I5	1
8	I6	1
9	I7	1
10	I0	1
11	I1	1
12	I2	1
13	I3	1
14	I4	1
15	I5	1
16	I6	1
17	I7	1
18	I0	1
19	I1	1
20	I2	1
21	I3	1
22	I4	1
23	I5	1
24	I6	1
25	I7	1

Figura 152. Base de datos Excel
Fuente: Realizado por el autor

Anexo 7: Código Arduino – Alerta Practicas

```
const int naranja= 6; //Constante tipo entero de nombre led con valor 6

const int verde= 7; //Constante tipo entero de nombre led con valor 6

void setup(){

pinMode(naranja, OUTPUT); //Habilita Pin número 6 de Arduino como salida

pinMode(verde, OUTPUT);

Serial.begin(9600);    //Inicializa comunicación Serial a una velocidad de 9600 baudios

digitalWrite(verde,HIGH);  //Pone en alto el led alerta1 verde que por lógica de relay apaga el led

digitalWrite(naranja,LOW);    //Pone en bajo el led alerta1 naranja que por lógica del relay lo enciende

}

void loop(){

  while(Serial.available()){ //Mientras la comunicación serial se encuentre activa

    if(Serial.read()=='H'){  //Si el valor recibido es H

      digitalWrite(naranja,HIGH); //el led naranja se apaga

      digitalWrite(verde,LOW); //el foco verde se enciende

    }

    else{

      digitalWrite(verde,HIGH); //el foco verde se apaga
```

```
    digitalWrite(naranja,LOW);  //el foco verde se enciende
    }
  }
  delay(1000);
  digitalWrite(verde,HIGH);
  digitalWrite(naranja,LOW);
}
```

Anexo 8: Código Raspberry salidas intermitentes

```
import serial    #Utiliza la librería para comunicación serial del Raspberry
import RPi.GPIO  #Utiliza librería para manejo de puertos entrada y salida
import time  #Utiliza librería para manejo de retardos en la ejecución del código
arduino=serial.Serial(
port='/dev/ttyACM0',
baudrate=9600,
parity=serial.PARITY_NONE,
stopbits=serial.STOPBITS_ONE,
bytesize=serial.EIGHTBITS,
timeout=1
)   #Inicializa la comunicación serial y especifica los parámetros de configuración y
los valores los cuales son el puerto, la velocidad, pariedad, bits de parada, tamaño del
envio, tiempo de respuesta
RPi.GPIO.setmode(RPi.GPIO.BCM)    #Utiliza la distribución de puertos por
software y no numero de pines
```

```
RPi.GPIO.setwarnings(False)     #Elimina las alertas que se generan al cambiar de
estado un puerto
RPi.GPIO.setup(20, RPi.GPIO.OUT)   #inicializa los puertos como salidas
RPi.GPIO.setup(21, RPi.GPIO.OUT)
RPi.GPIO.setup(16, RPi.GPIO.OUT)
RPi.GPIO.setup(12, RPi.GPIO.OUT)
RPi.GPIO.setup(7, RPi.GPIO.OUT)
RPi.GPIO.setup(8, RPi.GPIO.OUT)
RPi.GPIO.setup(25, RPi.GPIO.OUT)
RPi.GPIO.setup(24, RPi.GPIO.OUT)
arduino.close()  #Cierra el puerto serial en caso de encontrarse activo
arduino.open()  #Activa el puerto serial
txt="H"   #Genera una variable con el valor “H”
while True: #Programa principal
    RPi.GPIO.output(21, True) #Pone 1 logico en el puerto seleccionado
    RPi.GPIO.output(20, True)
    RPi.GPIO.output(16, True)
    RPi.GPIO.output(12, True)
    RPi.GPIO.output(7, True)
    RPi.GPIO.output(8, True)
    RPi.GPIO.output(25, True)
    RPi.GPIO.output(24, True)
    arduino.write(txt.encode()) #envía el valor de txt por el puerto serial
    time.sleep(1) #maneja un retardo de un segundo entre envío de datos y
encendido de leds
```

```
RPi.GPIO.output(21, False) #Pone 0 lógico en los puertos de salida
RPi.GPIO.output(20, False)
RPi.GPIO.output(16, False)
RPi.GPIO.output(12, False)
RPi.GPIO.output(7, False)
RPi.GPIO.output(8, False)
RPi.GPIO.output(25, False)
RPi.GPIO.output(24, False)
time.sleep(1) #Da un tiempo de 1 segundo de retardo para seguir con el código
```

Anexo 9: Código manejo GPIO entrada_salida Raspberry

```
import serial
import RPi.GPIO as GPIO
import time
arduino=serial.Serial(
port='/dev/ttyACM0',
baudrate=9600,
parity=serial.PARITY_NONE,
stopbits=serial.STOPBITS_ONE,
bytesize=serial.EIGHTBITS,
timeout=1
)
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(2, GPIO.IN) #Indica los puertos  que se configuran como salida
GPIO.setup(3, GPIO.IN)
```

```
GPIO.setup(4, GPIO.IN)
GPIO.setup(17, GPIO.IN)
GPIO.setup(27, GPIO.IN)
GPIO.setup(22, GPIO.IN)
GPIO.setup(10, GPIO.IN)
GPIO.setup(9, GPIO.IN)
GPIO.setup(21, GPIO.OUT)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)
GPIO.setup(7, GPIO.OUT)
GPIO.setup(8, GPIO.OUT)
GPIO.setup(25, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)
arduino.close()
arduino.open()
txt="H"
while True:
  arduino.write(txt.encode())
  if GPIO.input(2)==GPIO.LOW:   #Condición mientras el botón está apagado
    GPIO.output(21, True) #La salida se ubica en 1 lógico
  if GPIO.input(2)==GPIO.HIGH:  #Condición mientras el botón está encendido o
presionado
    GPIO.output(21, False)  #La salida se ubica en 0 lógico
  if GPIO.input(3)==GPIO.LOW:
```

```
    GPIO.output(20, True)
if GPIO.input(3)==GPIO.HIGH:
    GPIO.output(20, False)
if GPIO.input(4)==GPIO.LOW:
    GPIO.output(16, True)
if GPIO.input(4)==GPIO.HIGH:
    GPIO.output(16, False)
if GPIO.input(17)==GPIO.LOW:
    GPIO.output(12, True)
if GPIO.input(17)==GPIO.HIGH:
    GPIO.output(12, False)
if GPIO.input(27)==GPIO.LOW:
    GPIO.output(7, True)
if GPIO.input(27)==GPIO.HIGH:
    GPIO.output(7, False)
if GPIO.input(22)==GPIO.LOW:
    GPIO.output(8, True)
if GPIO.input(22)==GPIO.HIGH:
    GPIO.output(8, False)

if GPIO.input(10)==GPIO.LOW:
    GPIO.output(25, True)

if GPIO.input(10)==GPIO.HIGH:
    GPIO.output(25, False)
```

```
  if GPIO.input(9)==GPIO.LOW:
    GPIO.output(24, True)

  if GPIO.input(9)==GPIO.HIGH:
    GPIO.output(24, False)
  arduino.write(txt.encode())
  time.sleep(1)
GPIO.cleanup() #borra cualquier puerto en caso de encontrarse activo al finalizar el
ciclo
```

Anexo 10 A: Código index.php

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
            <link rel="stylesheet"
href="https://www.w3schools.com/w3css/4/w3.css">
<title>Proyecto Electronica</title> <!—Titulo de la página web  -->
<style>
body {
      background-image: url("portada.png"); <!—Imagen de fondo -->
          background-position: top center;  <!—posición de la imagen de fondo
superior centro  -->
}
</style>
```

```
</head>
<body>
<div id="inferior"> <!—Division de botónes -->
<a href="puertos.php" class="w3-button w3-blue">Entrada y Salida</a> <!—botón
con enlace a puertos.php -->
<a href="export.php" class="w3-button w3-blue"> Exportar Excel</a> <!—botón
con enlace a export.php -->
</div>
</body>
</html>
<style type="text/css">
body,html{
height:100%; /*Siempre es necesario cuando trabajamos con alturas*/
}
 #inferior{
color: #FFF;
background: #000;
position:absolute; /*posición inferior según la pantalla*/
left:0px; /*Espacio derecho*/
right:0px; /*Espacio izquierda*/
bottom:0px; /*espacio del botón*/
height:50px; /*alto del botón*/
z-index:0;
 }
</style>
```

Anexo 10 B: Código puertos.php

```
<HTML> <!-- Llamar método HTML -->
    <head> <!— Inicio encabezado  -->
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet"
href="https://www.w3schools.com/w3css/4/w3.css">
    </head>
    <body>

<div id="inferior"><a href="/" class="w3-button w3-blue">Entrada y Salida</a>
</div>

    </body>
</html>

        <div class="w3-padding"> <!— creación fila 1  -->
            <a href="/io.php?pin=2&state=1" class="w3-button w3-blue"> <!—
creación de botón columna 1 con referencia a io.php y envió de parámetros  -->
                LED 8 – ENCENDER <!— Texto del botón -->
            </a> <!— Fin  botón columna 1  -->
            <a href="/io.php?pin=2&state=0" class="w3-button w3-red"> <!—
creación de botón columna 2 con referencia a io.php y envió de parámetros  -->
                LED 8 - APAGAR
            </a> <!— Fin  botón columna 2 -->
```

```
</div>

<div class="w3-padding"> <!— creación fila 2 -->
        <a href="/io.php?pin=3&state=1" class="w3-button w3-blue">
            LED 7 - ENCENDER
        </a>
        <a href="/io.php?pin=3&state=0" class="w3-button w3-red">
            LED 7 - APAGAR
        </a>

</div>

<div class="w3-padding"> <!— creación fila 3 -->
        <a href="/io.php?pin=4&state=1" class="w3-button w3-blue">
            LED 6 - ENCENDER
        </a>
        <a href="/io.php?pin=4&state=0" class="w3-button w3-red">
            LED 6 - APAGAR
        </a>

</div>

<div class="w3-padding"> <!— creación fila 4 -->

        <a href="/io.php?pin=17&state=1" class="w3-button w3-blue">
```

```
            LED 5 - ENCENDER
        </a>
        <a href="/io.php?pin=17&state=0" class="w3-button w3-red">
            LED 5 - APAGAR
        </a>

</div>

<div class="w3-padding"> <!— creación fila 5 -->

        <a href="/io.php?pin=27&state=1" class="w3-button w3-blue">
            LED 4 - ENCENDER
        </a>
        <a href="/io.php?pin=27&state=0" class="w3-button w3-red">
            LED 4 - APAGAR
        </a>

</div>

<div class="w3-padding"> <!— creación fila 6 -->

        <a href="/io.php?pin=22&state=1" class="w3-button w3-blue">
            LED 3 - ENCENDER
        </a>
        <a href="/io.php?pin=22&state=0" class="w3-button w3-red">
```

```
            LED 3 - APAGAR
        </a>

</div>

<div class="w3-padding"> <!— creación fila 7 -->

        <a href="/io.php?pin=10&state=1" class="w3-button w3-blue">
            LED 2 - ENCENDER
        </a>
        <a href="/io.php?pin=10&state=0" class="w3-button w3-red">
            LED 2 - APAGAR
        </a>

</div>
<div class="w3-padding"> <!— creación fila 8 -->

        <a href="/io.php?pin=9&state=1" class="w3-button w3-blue">
            LED 1 - ENCENDER
        </a>
        <a href="/io.php?pin=9&state=0" class="w3-button w3-red">
            LED 1 - APAGAR
        </a>

</div>
```

```
<style type="text/css"> /*Método style tipo texto y css*/
body,html{ /*ubicación del cuerpo en la página web*/
height:100%; /*Definir altura*/
}
 #inferior{ /* Método denominado inferior para el botón*/
color: #FFF; /* Determinación de un color*/
background: #000; /* Color de Fondo*/
position:absolute; /*Posición con relación a la página*/
left:0px; /* Espacio derecha */
right:0px; /*Espacio izquierda*/
bottom:0px; /*Espacio del botón*/
height:50px; /*Espacio en altura*/
z-index:0; /*ajuste en la página*/
 }
</style> /*Fin método*/
```

Anexo 11 A: Código io.php

```
?php
        $pin=$_GET['pin'];
        $state=$_GET['state'];
        exec('sudo python3 io.py '.$pin.' '.$state);
        header('Location: /puertos.php');
?>
```

Anexo 11 B: Código io.py

```
import RPi.GPIO as GPIO #Se agrega la librería de manejo de puertos y se la nombra como GPIO
from sys import argv as arguments # De la librería sys se importa argv y se lo nombra como arguments
GPIO.setmode(GPIO.BCM) # Se designa la utilización de los puertos con numeración por software de puertos

pin=int(arguments[1]) #Se recibe primer dato recibido y se lo asigna a pin
state=int(arguments[2]) #Se recibe segundo dato y se lo asigna a la variable state
GPIO.setup(pin, GPIO.OUT) #Asignamos el puerto determinado en pin como salida
if state==1: #Si el dato recibido en state es 1
    GPIO.output(pin, False) #Se designa como apagado al número recibido en pin de puerto
if state==0: # Si el dato recibido en state es 0
    GPIO.output(pin, True) # Se designa como encendido al número recibido en pin de puerto
```

Anexo 12: Código BaseD.py

```
import MySQLdb as Mysql  #Importa librería para manejo de base de datos y se la nombra como Mysql
import sys #Librería de control de programa
import RPi.GPIO as GPIO #Librería para manejo de puertos y se la nombra como GPIO
GPIO.setmode(GPIO.BCM)  #Se utiliza la distribución por software de los puertos
GPIO.setwarnings(False) #Desactiva las alertas por defecto en puerto
```

```
#Conectar
baseD= Mysql.connect(host="localhost",
                     user="puertos",
                     passwd="plcpi2018",
                     db="puertos") #Asigna a una variable denominada baseD la
información necesaria para comunicarse con la base de datos
cursor = baseD.cursor() #Designa una variable para ejecución de órdenes mysql
cursor.execute("DELETE FROM gpio") #Deja en cero los valores de la base de
datos
baseD.commit() #Ejecuta la sentencia anterior
#Habilita todos los puertos que manejan tanto salidas y entradas del módulo
GPIO.setup(2, GPIO.OUT)
GPIO.setup(3, GPIO.OUT)
GPIO.setup(4, GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
GPIO.setup(10, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)
GPIO.setup(7, GPIO.OUT)
GPIO.setup(8, GPIO.OUT)
GPIO.setup(25, GPIO.OUT)
```

```
GPIO.setup(24, GPIO.OUT)
GPIO.setup(9, GPIO.OUT)
# Asigna el valor retornado por el puerto a una variable “0 si se encuentra en 0L o 1 si se encuentra en 1L”
S8 = GPIO.input(21)
S7 = GPIO.input(20)
S6 = GPIO.input(16)
S5 = GPIO.input(12)
S4 = GPIO.input(7)
S3 = GPIO.input(8)
S2 = GPIO.input(25)
S1 = GPIO.input(24)
E8 = GPIO.input(2)
E7 = GPIO.input(3)
E6 = GPIO.input(4)
E5 = GPIO.input(17)
E4 = GPIO.input(27)
E3 = GPIO.input(22)
E2 = GPIO.input(10)
E1 = GPIO.input(9)

if S1==1:   #Si la variable tiene un valor asignado igual a 1
   print("apagado") #Significa que está apagado por lo cual se imprime en el promt apagado
```

```
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S1','Apagado'))
#Inserta el nombre del puerto y su estado en la base de datos denominada gpio
   baseD.commit() #Ejecuta la orden anterior
else: #caso contrario la variable tiene un valor diferente a 1 en este caso 0
   print("encendido")
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S1','Encendido'))
   baseD.commit()
if S2==1:
   print("apagado")
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S2','Apagado'))
   baseD.commit()
else:
   print("encendido")
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S2','Encendido'))
   baseD.commit()
if S3==1:
   print("apagado")
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S3','Apagado'))
   baseD.commit()
else:
   print("encendido")
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S3','Encendido'))
   baseD.commit()
if S4==1:
   print("apagado")
```

```
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S4','Apagado'))
    baseD.commit()
  else:
    print("encendido")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S4','Encendido'))
    baseD.commit()
  if S5==1:
    print("apagado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S5','Apagado'))
    baseD.commit()
  else:
    print("encendido")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S5','Encendido'))
    baseD.commit()
  if S6==1:
    print("apagado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S6','Apagado'))
    baseD.commit()
  else:
    print("encendido")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S6','Encendido'))
    baseD.commit()
  if S7==1:
    print("apagado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S7','Apagado'))
```

```
    baseD.commit()
else:
    print("encendido")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S7','Encendido'))
    baseD.commit()
if S8==1:
    print("apagado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S8','Apagado'))
    baseD.commit()
else:
    print("encendido")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('S8','Encendido'))
    baseD.commit()
if E1==0:
    print("no pulsado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E1','No pulsado'))
    baseD.commit()
else:
    print("pulsado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E1','Pulsado'))
    baseD.commit()
if E2==0:
    print("no pulsado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E2','No pulsado'))
    baseD.commit()
```

```
else:
   print("pulsado")
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E2','Pulsado'))
   baseD.commit()
if E3==0:
   print("no pulsado")
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E3','No pulsado'))
   baseD.commit()
else:
   print("pulsado")
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E3','Pulsado'))
   baseD.commit()
if E4==0:
   print("no pulsado")
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E4','No pulsado'))
   baseD.commit()
else:
   print("pulsado")
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E4','Pulsado'))
   baseD.commit()
if E5==0:
   print("no pulsado")
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E5','No pulsado'))
   baseD.commit()
else:
```

```
    print("pulsado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E5','Pulsado'))
    baseD.commit()
if E6==0:
    print("no pulsado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E6','No pulsado'))
    baseD.commit()
else:
    print("pulsado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E6','Pulsado'))
    baseD.commit()
if E7==0:
    print("no pulsado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E7','No pulsado'))
    baseD.commit()
else:
    print("pulsado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E7','Pulsado'))
    baseD.commit()
if E8==0:
    print("no pulsado")
    cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E8','No pulsado'))
    baseD.commit()
else:
    print("pulsado")
```

```
   cursor.execute("""INSERT INTO gpio VALUES (%s,%s)""",('E8','Pulsado'))
   baseD.commit()
sys.exit() #Sale del programa una vez finalizado la carga de los estados de todos los puertos de entrada y salida
```

Anexo 13: Código visualizar.py

```
import MySQLdb as Mysql
# Conecta a la base de datos
baseD = Mysql.connect(host="localhost",
                      user="puertos",
                      passwd="plcpi2018",
                      db="puertos")

cursor = baseD.cursor()

cursor.execute("SELECT * FROM gpio") #Ejecuta un comando sql que requiere todos los valores de la tabla gpio
Nfilas = cursor.rowcount # Asigna el valor de filas que existe en la BD a una variable
for x in range(0, Nfilas):   #mientras x se mantenga dentro de un valor inicial de 0 hasta el número de las filas
    fila = cursor.fetchone() #El cursor se desplaza entre filas
    print(fila[0], "-->", fila[1]) #Se imprimen todas las filas que tiene la BD
baseD.close() #Se cierra la conexión con la BD
```

Anexo 14: Código export.php

```
<?php
	require_once("database.php");  /*Solicita la librería para manejo de DB*/
	$db=new DataBase("127.0.0.1","puertos","plcpi2018","puertos"); /* Se crea una variable para asignar los valores de conexión a la BD*/
	$select=new Select("puerto","estado"); /*Se selecciona las columnas a recuperar*/
	$select->from("gpio"); /*se identifica de que tabla se recupera las columnas*/
	$results=$db->execute_query($select,[]); /*Se ejecuta la coneccion y la orden de selección de los parámetros previamente indicados*/

	require('excel.php'); /*Se solicita que la librería Excel.php genere el archivo en formato xls*/
	export('GPIO',$results); /*Se exporta el archivo con un nombre GPIO y se lo descarga*/

?>
```

Anexo 15: Librería database.php

```
<?php
	class Select{
		private $fields="";
		private $fields_counter=0;
		private $conditions="";
		private $joins="";
		private $conditions_counter=0;
```

```
private $data_base="";
function __construct(){
	for($index=0;$index<func_num_args();$index++){
		$this->field(func_get_arg($index));
	}
}
function field($field_name){
	if($this->fields_counter==0){
		$this->fields=$this->fields.$field_name;
	}else{
		$this->fields=$this->fields.",".$field_name;
	}
	$this->fields_counter++;
}
function from($data_base){
	if($data_base==""){
		throw new Exception('Data base already selected.');
	}
	$this->data_base=$data_base;
}
public static $INNER_JOIN="INNER JOIN";
function join($type,$table,$field1,$field2){
	$join=$type." ".$table." ON ".$field1."=".$field2;
	$this->joins=$this->joins." ".$join;
}
```

```
function where(){
    switch(func_num_args()){
        case 2:
            $this->first_where(func_get_arg(0),func_get_arg(1));
            break;
        case 3:
            $this->lasts_where(func_get_arg(0),func_get_arg(1),func_get_arg(2));
            break;
        default:
            throw new Exception('Invalid parameters to where clause.');
            break;
    }
}
function first_where($condition,$clause){
    if($this->conditions_counter==0){
        $this->conditions=$this->conditions.$condition.$clause.":v".$this->conditions_counter;
    }else{
        throw new Exception('To separe the where statements must indicate the logic field.');
    }
    $this->conditions_counter++;
```

```
		}
		function lasts_where($separator,$condition,$clause){
			if($this->conditions_counter==0){
				throw new Exception('The fisrt statement do not need
the logic field.');
			}else{
				$this->conditions=$this->conditions." ".$separator."
".$condition.$clause.":v".$this->conditions_counter;
			}
			$this->conditions_counter++;
		}
		function toString(){
			if($this->fields_counter==0){
				throw new Exception('At least one field must be
selected.');
			}
			if($this->data_base==""){
				throw new Exception('The table was not specified.');
			}
			$conditions="";
			if($this->conditions_counter>0){
				$conditions=" WHERE ".$this->conditions;
			}
			$joins="";
			if($this->joins!=""){
```

```
                $joins=$this->joins;
            }
            return "SELECT ".$this->fields." FROM ".$this->data_base.$joins.$conditions;
        }
    }

    class DataBase{
        private $SERVER="localhost";
        private $USER="root";
        private $PASSWORD="";
        private $DATA_BASE="cursos";
        private $connection;
        function __construct($server,$user,$password,$data_base){
            $this->connection=new PDO("mysql:host=".$server.";dbname=".$data_base,$user,$password);
            $this->connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        }
        function execute_query($query,$conditions){
            $structure=$query->toString();jgchg
            $statement=$this->prepared_query($structure,$conditions);
            return $statement;
        }
        function query($query){
```

```
                return $this->connection->query($query);
        }
        function get_statement($query){
                return $this->connection->prepare($query);
        }
        function prepared_query($query,$conditions){
                $statement=$this->get_statement($query);
                for($index=0;$index<count($conditions);$index++){
                        $statement-
>bindParam(":v".$index,$conditions[$index]);
                }
                $statement->execute();
                $statement->setFetchMode(PDO::FETCH_ASSOC);
                return $statement->fetchAll();
        }
        function get_last_id(){
                return $this->connection->lastInsertId();
        }
}
```

Anexo 16: Excel.php

```
<?php
        function cleanData(&$str)
        {
                $str = preg_replace("/\t/", "\\t", $str);
                $str = preg_replace("/\r?\n/", "\\n", $str);
```

```
        if(strstr($str, '"')) $str = '"' . str_replace('"', '""', $str) . '"';
    }
    function export($name,$data)
    {
        $filename = $name . ".xls";

        header("Content-Disposition: attachment; filename=\"$filename\"");
        header("Content-Type: application/vnd.ms-excel");

        $flag = false;
        foreach($data as $row)
        {
            if(!$flag)
            {
                echo implode("\t", array_keys($row)) . "\r\n";
                $flag = true;
            }
            array_walk($row, __NAMESPACE__ . '\cleanData');
            echo implode("\t", array_values($row)) . "\r\n";
        }
        exit;
    }
?>
```

Anexo17. Proporcional

```
#include<PID_v1.h>

int Sensor=0;

float calibrar, distancia1;

float Kp = 0.003106;                                    //Initial Proportional Gain

float Ki = 0.098;                                       //Initial Integral Gain

float Kd = 0;

float KpS=3.1;

float proporcional, integral, derivativo, movimiento;

int rango1=0;

int i=0;

const int canalA = 3;

const int canalB = 2;

const int limite = 150;

volatile int derecha=0;

volatile int izquierda=0;

int sensor [] ={64,108,152,196,240,280,385,490,595,700};

int Rango [] = {-20,-15,-10,-5,0,5,10,15,20};

int Pin;

int valor=0;

int rango_pid;

//PID

float error,erroract,errorpas;

double Setpoint, Input, Output, ServoOutput;

float setpoint=0;
```

```
float velocidad=180;
float pid;
int estable=7;
int ciclo=0;
int inestable=6;
//Variables
int analogo;
unsigned long timeI=0;
unsigned long timeF=0;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);
void setup() {
  pinMode(estable,OUTPUT);
  pinMode(inestable,OUTPUT);
  pinMode(canalA, INPUT_PULLUP);
  Serial.begin(9600);
  attachInterrupt(digitalPinToInterrupt(canalA), EncodeA, CHANGE);
  analogReference(EXTERNAL);
  myPID.SetMode(AUTOMATIC);
  myPID.SetOutputLimits(-180,180);
}
void loop() {
 timeF=millis();
 int ultimaD=analogo;
 analogo=distancia();
```

```
timeI=timeF;
i=0;
rango_pid=calibrando(analogo);
Serial.print(" sensorult  ");
Serial.print(ultimaD);
Serial.print(" sensor  ");
Serial.print(analogo);
Serial.print(" tiempoF  ");
Serial.print(" izquierda  ");
Serial.print(izquierda);
Serial.print(" derecha  ");
Serial.print(derecha);
Serial.print(" posicion  ");
Serial.print(rango_pid);
Serial.print(" velocidad Motor  ");
Serial.println(movimiento);
error=rango_pid;
erroract=setpoint-error;
proporcional=KpS*error;
movimiento=proporcional+error;
motor();
while(ciclo<=2){
  Setpoint=valor;
Input=derecha;
```

```
 myPID.Compute();
if(derecha>valor){
   analogWrite(10,abs(Output));
   analogWrite(9,0);
 }
 if(derecha<valor){
   analogWrite(9,abs(Output));
   analogWrite(10,0);
 }
 if(derecha==valor){
   analogWrite(10,0);
   analogWrite(9,0);
 }
 ciclo++;
 }
 }
 if((rango_pid<0)){
     analogWrite(9,abs(movimiento));
    analogWrite(10,0);
 }
 if((rango_pid>0)){
    analogWrite(10,abs(movimiento));
    analogWrite(9,0);
```

```
  }
  if(izquierda>70) analogWrite(10,0);
  if(derecha>70) analogWrite(9,0);

}

int distancia(){
  long suma=0;
  for(int i=0;i<25;i++)
  {
    suma=suma+analogRead(A0);
  }
  return(suma/25);
}

void EncodeA()
{
      if (digitalRead(canalA) == digitalRead(canalB))
      {
         if (derecha + 1 <= limite){
          derecha++;
         }
      }
       if (digitalRead(canalA) != digitalRead(canalB))
```

```
      {
        if (derecha + 1 <= limite){
         derecha--;
        }

      }
}
int calibrando(int sensor3){
  while(i<=9){
    int j=i+1;
    if((sensor3>sensor[i])&&(sensor3<sensor[j])){
      rango1=Rango[i];
      return(rango1);
    }
    i++;

  }
  //i=0;
}
void motor(){
  if(movimiento<0){
    analogWrite(9,abs(movimiento));
    analogWrite(10,0);
  }
```

```
 if(movimiento>0){
   analogWrite(10,abs(movimiento));
   analogWrite(9,0);
 }
 if(movimiento==0){
   analogWrite(10,0);
   analogWrite(9,0);
 }
 ciclo=0;
 delay(12);

}
```

Anexo 18. Código Práctica 9

```
#include<PID_v1.h>
int Sensor=0;
float calibrar, distancia1;
float Kp = 0.0011;
float Ki = 0.078;
float Kd = 0;
float KpS=3.2;
float KiS=2.3;
float proporcional, integral, derivativo, movimiento;
int rango1=0;
int i=0;
```

```
const int canalA = 3;
const int canalB = 2;
const int limite = 1000;
volatile int derecha=0;
volatile int izquierda=0;
int sensor [] ={120,145,190,235,280,310,440,570,697};
int Rango [] = {-20,-15,-10,-5,0,5,10,15,20};
int Pin;
int valor=0;
int rango_pid;
//PID
float error,erroract,errorpas;
double Setpoint, Input, Output, ServoOutput;
float setpoint=0;
float velocidad=0;
float pid;
int estable=7;
int ciclo=0;
int inestable=6;
int final1=0;
//Variables
int analogo;
unsigned long timeI=0;
unsigned long timeF=0;
```

```
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);
void setup() {
   pinMode(estable,OUTPUT);
   pinMode(inestable,OUTPUT);
   pinMode(canalA, INPUT_PULLUP);
   Serial.begin(9600);
   attachInterrupt(digitalPinToInterrupt(canalA), EncodeA, CHANGE);
   analogReference(EXTERNAL);
   myPID.SetMode(AUTOMATIC);
   myPID.SetOutputLimits(-180,180);
  // put your setup code here, to run once:

}

void loop() {
  timeF=millis();
  int ultimaD=analogo;
  analogo=distancia();
  timeI=timeF;
  i=0;
  rango_pid=calibrando(analogo);
  Serial.print(" sensorult  ");
  Serial.print(ultimaD);
  Serial.print(" sensor  ");
```

```
Serial.print(analogo);
Serial.print(" tiempoF  ");
Serial.print(" izquierda  ");
Serial.print(izquierda);
Serial.print(" derecha  ");
Serial.print(derecha);
Serial.print(" posicion  ");
Serial.print(rango_pid);
Serial.print(" final ");
Serial.print(final1);
Serial.print(" posicion F  ");
Serial.print(erroract);
Serial.print(" velocidad ");
Serial.print(velocidad);
Serial.print(" velocidad Motor  ");
Serial.println(movimiento);
error=rango_pid;
erroract=setpoint-error;
proporcional=KpS*abs(error);
integral=KiS*abs(velocidad);
movimiento=proporcional+integral;
motor();
Setpoint=valor;
Input=derecha+error;
```

```
  myPID.Compute();
  if(derecha>valor){
    analogWrite(10,abs(Output));
    analogWrite(9,0);
  }
  if(derecha<valor){
    analogWrite(9,abs(Output));
    analogWrite(10,0);
  }
  if(derecha==valor){
    analogWrite(10,0);
    analogWrite(9,0);
  }
  final1=distancia();
  i=0;
  erroract=calibrando(final1);
  velocidad=error-erroract;
}

int distancia(){
  long suma=0;
  for(int i=0;i<25;i++)
  {
    suma=suma+analogRead(A0);
```

```
  }
  return(suma/25);
}
void EncodeA()
{

     if (digitalRead(canalA) == digitalRead(canalB))
     {
       if (derecha + 1 <= limite){
        derecha++;
       }


     }
      if (digitalRead(canalA) != digitalRead(canalB))
     {
       if (derecha + 1 <= limite){
        derecha--;
       }

     }
}

int calibrando(int sensor3){
```

```
  while(i<=8){
    int j=i+1;
    if((sensor3>=sensor[i])&&(sensor3<sensor[j])){
      rango1=map(sensor3,sensor[i],sensor[j],Rango[i],Rango[j]);
      return(rango1);
    }
    i++;

  }
}
void motor(){
    if((velocidad<=4)&&(velocidad>=-4)){
    if(error<-5){
    analogWrite(9,abs(movimiento));
    analogWrite(10,0);
  }
  if(error>5){
    analogWrite(10,abs(movimiento));
    analogWrite(9,0);
  }
  }
  if(velocidad<-4){
    analogWrite(10,abs(movimiento));
    analogWrite(9,0);
```

```
 }
 if(velocidad>4){
  analogWrite(9,abs(movimiento));
  analogWrite(10,0);
 }
 if((error>=-5)&&(error<=5)){
  analogWrite(10,0);
  analogWrite(9,0);
 }
 ciclo=0;
 delay(9);
 return;
}
```

Anexo 19. Código Práctica 10

```
#include<PID_v1.h>
int Sensor=0;
float calibrar, distancia1;
float Kp = 0.0011;                                  //Initial Proportional Gain
float Ki = 0.078;                                   //Initial Integral Gain
float Kd = 0;
float KpS=3.2;
float KiS=2.3;
float KdS=0.7;
float proporcional, integral,  movimiento;
float derivativo=0;
```

```
int rango1=0;
int i=0;
const int canalA = 3;
const int canalB = 2;
const int limite = 1000;
volatile int derecha=0;
volatile int izquierda=0;
int sensor [] ={120,145,190,235,280,310,440,570,697};
int Rango [] = {-20,-15,-10,-5,0,5,10,15,20};
int Pin;
int valor=0;
int rango_pid;
//PID
float error,erroract,errorpas;
double Setpoint, Input, Output, ServoOutput;
float setpoint=0;
float velocidad=0;
float pid;
int estable=7;
int ciclo=0;
int inestable=6;
int final1=0;
int acumulativo=0;
//Variables
```

```
int analogo;
unsigned long timeI=0;
unsigned long timeF=0;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);
void setup() {
   pinMode(estable,OUTPUT);
   pinMode(inestable,OUTPUT);
   pinMode(canalA, INPUT_PULLUP);
   Serial.begin(9600);
   attachInterrupt(digitalPinToInterrupt(canalA), EncodeA, CHANGE);
   analogReference(EXTERNAL);
   myPID.SetMode(AUTOMATIC);
   myPID.SetOutputLimits(-180,180);
}

void loop() {
  timeF=millis();
  int ultimaD=analogo;
  analogo=distancia();
  timeI=timeF;
  i=0;
  rango_pid=calibrando(analogo);
  Serial.print(" sensorult  ");
  Serial.print(ultimaD);
```

```
Serial.print(" sensor  ");
Serial.print(analogo);
Serial.print(" tiempoF  ");
Serial.print(" izquierda  ");
Serial.print(izquierda);
Serial.print(" derecha  ");
Serial.print(derecha);
Serial.print(" posicion  ");
Serial.print(rango_pid);
Serial.print(" final ");
Serial.print(final1);
Serial.print(" posicion F  ");
Serial.print(erroract);
Serial.print(" velocidad ");
Serial.print(velocidad);
Serial.print(" velocidad Motor  ");
Serial.println(movimiento);
error=rango_pid;
erroract=setpoint-error;
proporcional=KpS*abs(error);
integral=KiS*abs(velocidad);
derivativo=KdS*abs(acumulativo);
movimiento=proporcional+integral+derivativo;
motor();
```

```
 Setpoint=valor;
 Input=derecha+error;
 myPID.Compute();
if(derecha>valor){
  analogWrite(10,abs(Output));
  analogWrite(9,0);
 }
 if(derecha<valor){
  analogWrite(9,abs(Output));
  analogWrite(10,0);
 }
 if(derecha==valor){
  analogWrite(10,0);
  analogWrite(9,0);
 }

 final1=distancia();
 i=0;
 erroract=calibrando(final1);
 velocidad=error-erroract;
}

int distancia(){
 long suma=0;
```

```
  for(int i=0;i<25;i++)
  {
    suma=suma+analogRead(A0);
  }
  return(suma/25);
}

void EncodeA()
{

     if (digitalRead(canalA) == digitalRead(canalB))
     {
       if (derecha + 1 <= limite){
        derecha++;
       }



     }
      if (digitalRead(canalA) != digitalRead(canalB))
     {
       if (derecha + 1 <= limite){
        derecha--;
       }
```

```
      }




}

int calibrando(int sensor3){
  while(i<=8){
    int j=i+1;
    if((sensor3>=sensor[i])&&(sensor3<sensor[j])){
      rango1=map(sensor3,sensor[i],sensor[j],Rango[i],Rango[j]);
      return(rango1);
    }
    i++;


  }
}
void motor(){
    if((velocidad<=4)&&(velocidad>=-4)){
    if(error<-5){
    analogWrite(9,abs(movimiento));
    analogWrite(10,0);
  }
  if(error>5){
```

```
  analogWrite(10,abs(movimiento));
  analogWrite(9,0);
 }
 }
 acumulativo++;

 if(velocidad<-4){
  analogWrite(10,abs(movimiento));
  analogWrite(9,0);
 }
 if(velocidad>4){
  analogWrite(9,abs(movimiento));
  analogWrite(10,0);
 }
 if((error>=-5)&&(error<=5)){
  analogWrite(10,0);
  analogWrite(9,0);
  acumulativo=0;
 }
 delay(13);
 return;
}
```

Anexo 20. Método por igualación de polos

La planta viene representada por la ecuación de transferencia obtenida en la Practica 7, mientras que el control a implementar en el encoder está representado por:

$$C(s) = \frac{Kps + Ki}{s}$$

Para despejar los valores y obtener una solución de forma algebraica se debe ubicar los polos en lazo cerrado, para esto es necesario tener en cuenta los siguientes valores de la forma transitoria que son:

ts(2%) < 1 seg ; tiempo de establecimiento

tp < 0.5 seg; tiempo pico en segundos

Mp < 20%; Sobrepaso menor

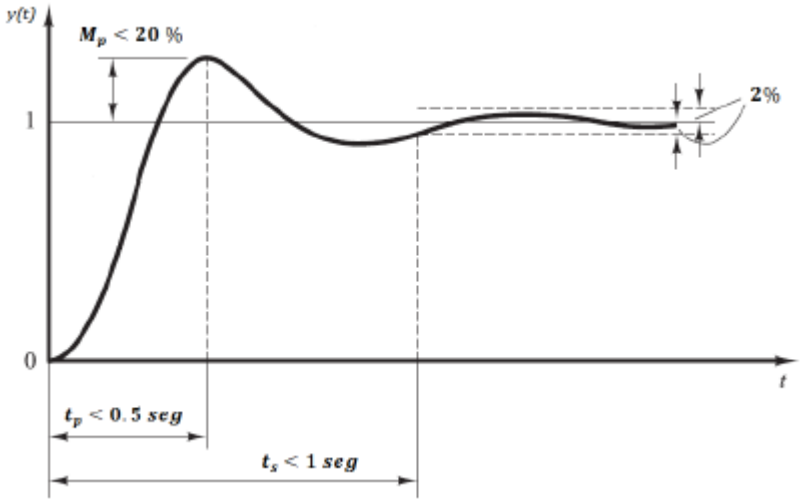


Figura 153. Gráfico de amortiguamiento
Fuente: Realizado por el autor

En base a la gráfica para la obtención de las formulas y valores es necesario utilizar la siguiente tabla:

Parámetro característico	Valor
Sobrepaso	$M_p = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}}; \qquad \zeta = \sqrt{\frac{ln^2 M_p}{\pi^2 + ln^2 M_p}};$
Tiempo de sobrepaso	$t_p = \frac{\pi}{\omega_d};$
Envolvente de la respuesta	$\sigma = \zeta\omega_0;$
Frecuencia Amortiguada	$\omega_d = \omega_0\sqrt{1-\zeta^2};$
Tiempo de Subida	$t_r = \frac{1}{\omega_d} arctan\left(\frac{\omega_d}{\sigma}\right);$
Tiempo de establecimiento	$t_s = \begin{cases} \frac{4}{\zeta\omega_0}; & (2\%) \\ \frac{3}{\zeta\omega_0}; & (5\%) \end{cases}$

Figura 154. Formulas ecuaciones amortiguadas
Fuente: Realizado por el autor

En base a la figura 154 se puede obtener todos los valores necesarios para el cálculo, una vez revisado todos los conceptos necesarios se procede a relacionar la salida y entrada para un proceso de lazo cerrado.

$$\frac{Y(s)}{R(s)} = \frac{C(s)F(s)}{1 + C(s)P(s)A(s)}$$

Se reemplaza en la formula anterior los valores de la función de transferencia y la función del controlador:

$$\frac{Y(s)}{R(s)} = \frac{\left(\frac{Kps + Ki}{s}\right)\left(\frac{87}{0.075s + 1}\right)}{1 + \left(\frac{Kps + Ki}{s}\right)\left(\frac{83}{0.075s + 1}\right)} \quad (1)$$

$$\frac{Y(s)}{R(s)} = \frac{\frac{87(Kps + Ki)}{s(0.075s + 1)}}{\frac{0.087s^2 - s + (Kps + Ki)87}{s(0.075s + 1)}}$$

$$\frac{Y(s)}{R(s)} = \frac{87(Kps + Ki)}{0.075s^2 - s + 87Kps + 87Ki}$$

$$\frac{Y(s)}{R(s)} = \frac{87(Kps + Ki)}{0.075s^2 - s(1 + 87Kp) + 87Ki}$$

Como se puede observar después del despeje y agrupación la ecuación obtenida se asemeja a un sistema de segundo orden:

$$G(s) = \frac{W^2 n}{s^2 + 22Wns + W^2 n}$$

Por lo cual es necesario reemplazar los valores en un sistema de segundo orden:

$$G(s) = \frac{87(Kps + Ki)}{s^2 + \frac{s(1 + 87Kp)}{0.075s} + \frac{87}{0.075}}$$

Es necesario relacionar la ganancia de control y la ubicación geográfica de polos:

$$\frac{1 + 87Kp}{0.075} = 2\zeta Wn = 2\sigma$$

$$\frac{87Ki}{0.075} = Wn^2 = \sigma^2 + Wd^2$$

Para obtener una respuesta deseada es necesario establecer los valores de establecimiento al 2% en menos de 1 segundo:

$$ts = \frac{4}{\sigma} < 1$$

$$\sigma > 4$$

El tiempo pico se relaciona en base al tiempo pico que se alcanza a los 0.5 segundos:

$$tp = \frac{\pi}{Wd} < 0.5$$

$$Wd < 6.28$$

El sobrepaso debe ser menor al 20%:

$$Wd = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} < 0.20$$

$$\zeta > \sqrt{\frac{(\ln(0.20))^2}{\pi^2 + (\ln(0.20))^2}}$$

$$\zeta > 0.45$$

Una vez obtenido los valores se procede a obtener el ángulo β:

$$\zeta = \cos\beta$$

$$\beta < \cos^{-1} 0.45$$

$$\beta = 62.3°$$

Una vez obtenidos todos los valores necesarios se lo debe graficar en el eje z:

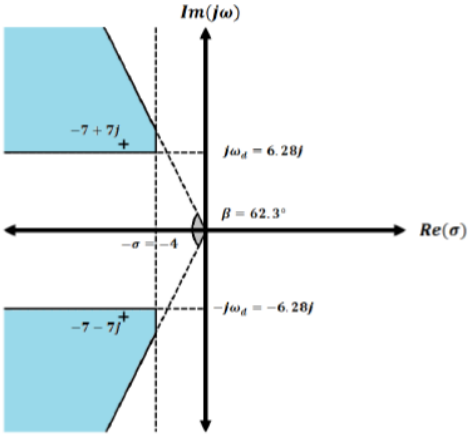


Figura 155. Asignación de polos
Fuente: Realizado por el autor

El área azul representa los valores que cumplen con las especificaciones de la transitoria , para el despeje de los valores del controlador se utiliza los valores enteros -7-7j que se utilizaran en las ecuaciones mencionadas con anterioridad para obtener Kp y Ki, donde **σ**=7 y Wd=7.

Entonces para Kp:

$$\frac{1 + 87Kp}{0.075} = 2\boldsymbol{\sigma}$$

$$Kp = \frac{2\sigma(0.093) - 1}{87}$$

$$Kp = \frac{2x7(0.093) - 1}{87}$$

$$Kp = \frac{2x7(0.093) - 1}{87}$$

$$Kp = 0.0035$$

Mientras que para Ki:

$$\frac{87Ki}{0.093} = \boldsymbol{\sigma^2 + Wd^2}$$

$$Ki = \frac{(\boldsymbol{\sigma^2 + Wd^2})(0.093)}{87}$$

$$Ki = \frac{(\mathbf{7^2 + 7^2})(0.093)}{87}$$

$$Ki = 0.105$$

Anexo 21. Diseño baquelita Eagle

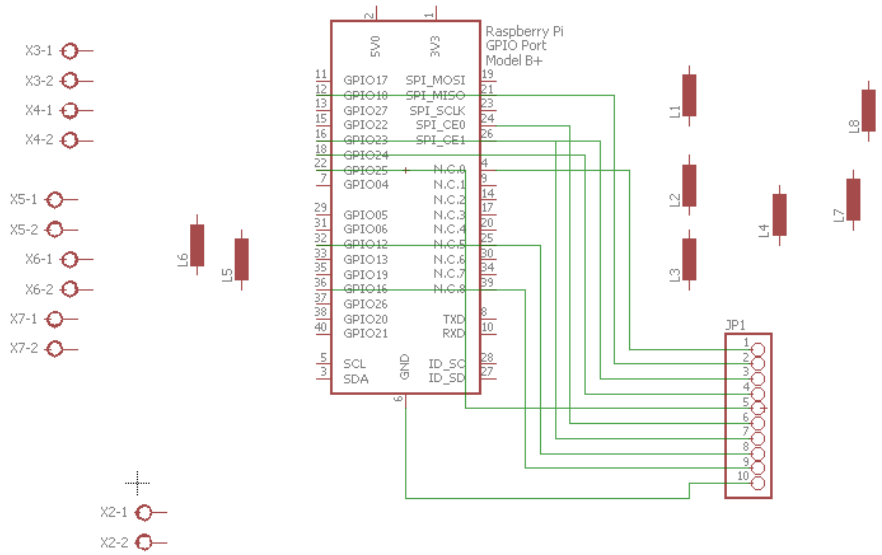


Figura 156. Diseño esquema raspberry eagle
Fuente: Realizado por el autor

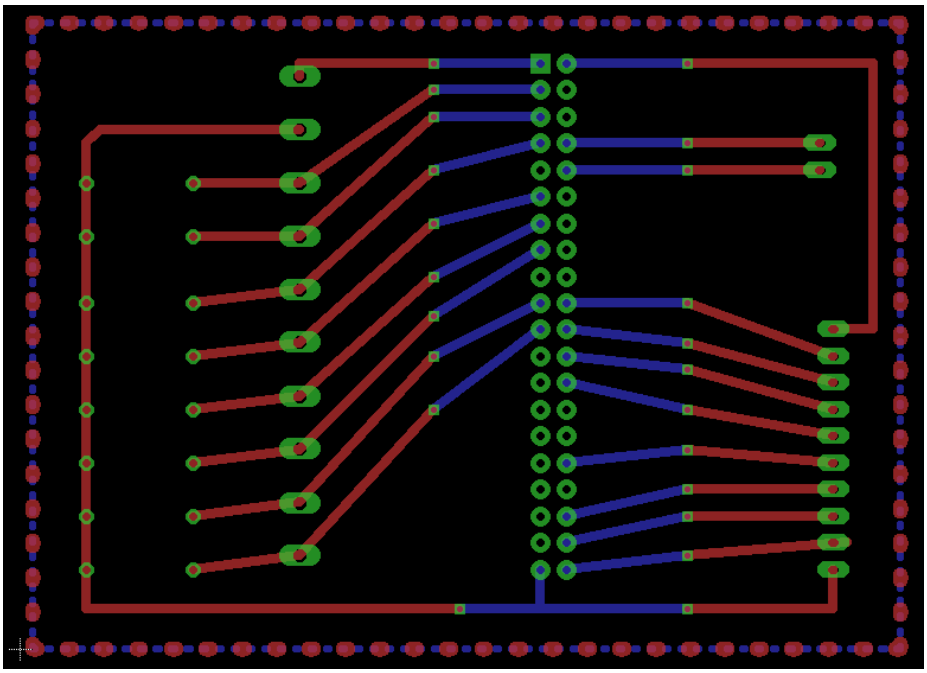


Figura 157. Diseño board raspberry eagle
Fuente: Realizado por el autor

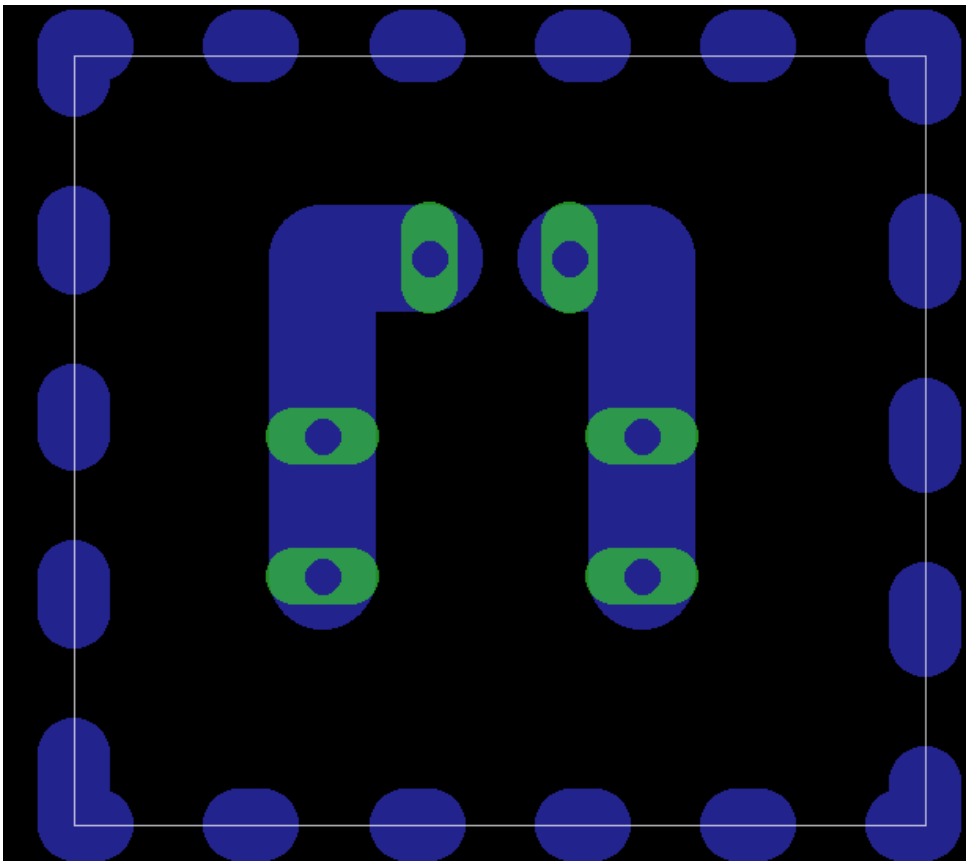


Figura 158. Diseño board conexiones
Fuente: Realizado por el autor

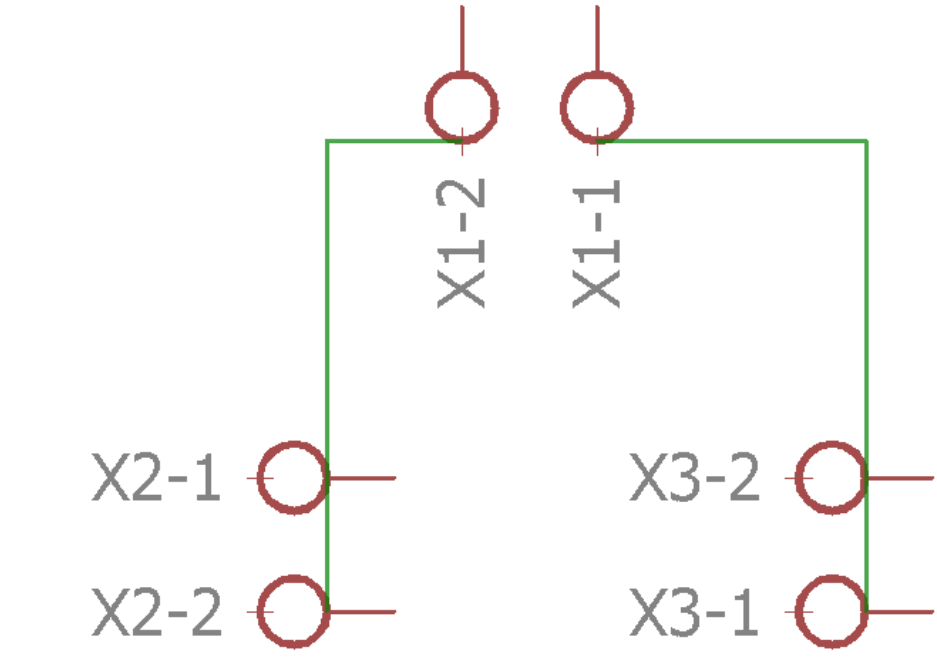


Figura 159. Diseño esquema conexiones
Fuente: Realizado por el autor

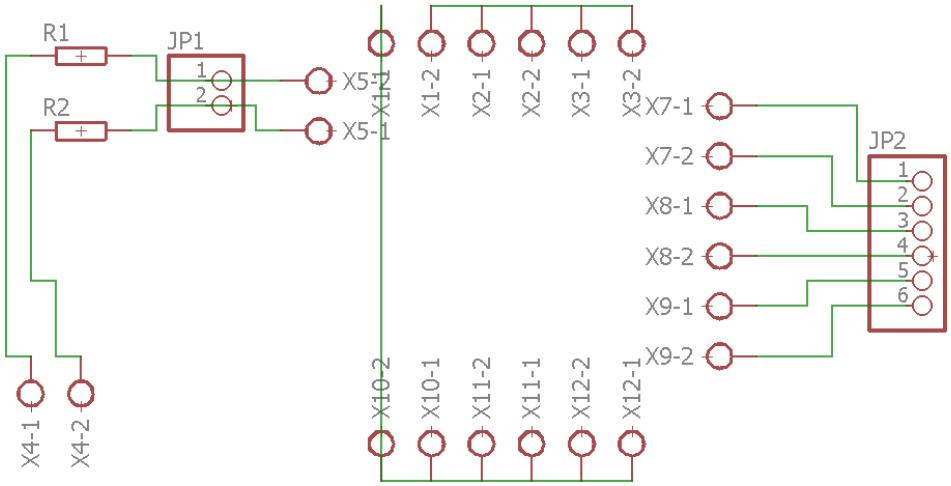


Figura 160. Diseño conexión balanza eagle
Fuente: Realizado por el autor

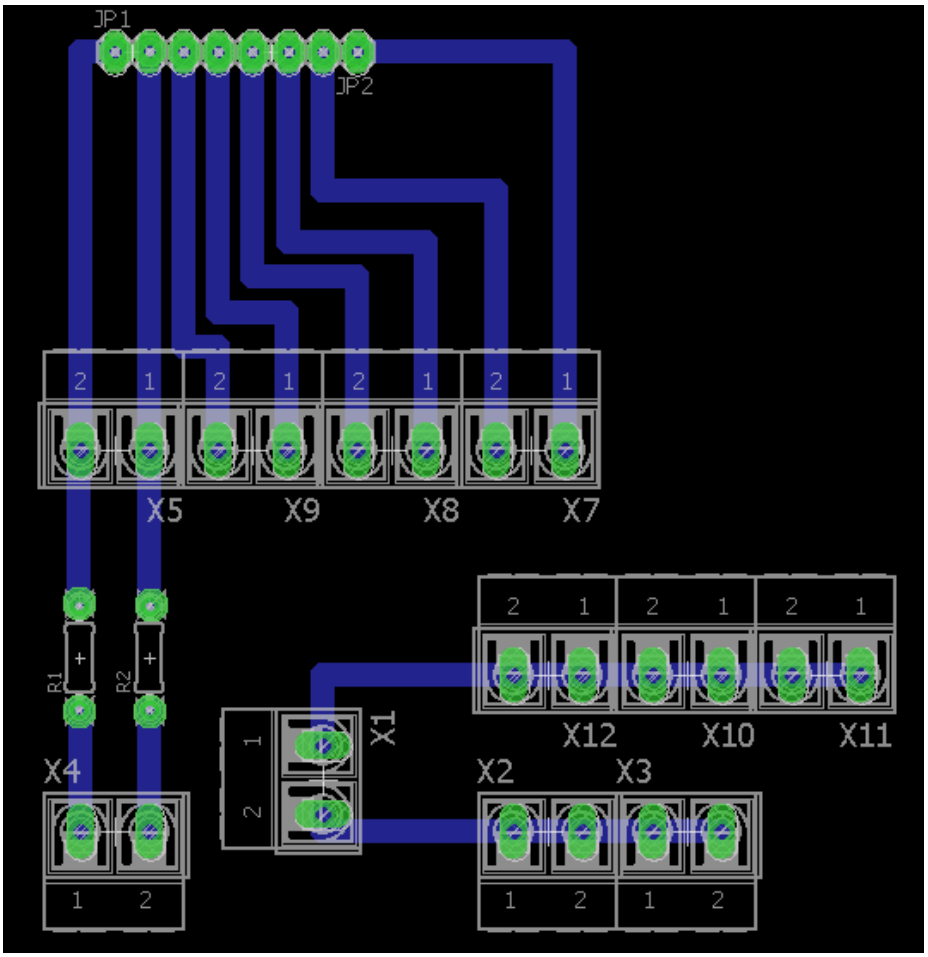


Figura 161. Diseño board conexiones
Fuente: Realizado por el autor

Anexo 22. Cronograma de actividades

Tabla 7.

Cronograma de Actividades

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR	CRONOGRAMA DE PROYECTO TÉCNICO																		
TEMA:	**DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO BASADO EN RASPBERRY PI 3 Y ARDUINO MEGA PARA CONTROL Y MONITOREO REMOTO DE UN BRAZO ESTABILIZADOR**																		
ACTIVIDADES	**MES 1**	**MES 2**	**MES 3**	**MES 4**	**MES 5**	**MES 6**	**MES 7**	**MES 8**	**MES 9**	**MES 10**	**MES 11**	**MES 12**	**MES 13**	**MES 14**	**MES 15**	**MES 16**	**MES 17**	**MES 18**
Capacitación plataforma Arduino y Raspberry PI 3																		
Compra y adquisición de equipos																		
Diseño y construcción de modulo didáctico																		
Desarrollo de prácticas 1,2,3																		
Desarrollo de prácticas 4,5,6																		
Acoplamiento de equipos en el modulo																		
Cableado y montaje de equipos en el módulo didáctico																		
Configuración de equipos																		
Pruebas de funcionamiento practicas 1,2,3,4,5,6																		
Diseño y construcción brazo estabilizador																		
Cableado y montaje brazo estabilizador																		
Desarrollo de prácticas 7,8																		
Desarrollo de prácticas 9,10																		
Pruebas de funcionamiento practicas 7,8,9,10																		
Realización de correcciones y modificaciones																		
Pruebas de funcionamiento finales																		
Desarrollo de documentación																		
Presentación del funcionamiento general de módulo																		
Reunión con tutor																		

Fuente: Realizado por el autor

Anexo 23. PID utilizando Ziegler-Nichols

Dentro del modelado matemático es necesario tomar en cuenta las variables necesarias para la planta, en este caso se cuenta con el sensor de distancia y el movimiento del motor. Por lo cual es necesario generar una escala de valores que se puedan controlar en referencia al sensor que nos entrega voltajes entre 0 a 5 voltios, permitiendo obtener medidas entre 0 y 1024 dependiendo de la distancia donde se encuentre el sensor, por lo cual en base a una tabla de valores predeterminados se obtiene una tabla de distancias que variaran de -20 a 20, siendo 0 el punto de consigna deseado.

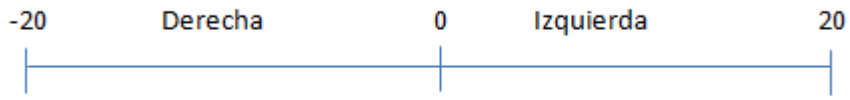


Figura 162. Tabla de valores lectura sensor vs dirección
Fuente: Realizado por el autor

Dentro de los valores se obtiene que al momento que el punto de consigna es negativo se desplaza el motor hacia la derecha y cuando es positivo hacia la izquierda.

Una vez obtenidos los valores es necesario tomar en cuenta que el método Ziegler-Nichols indica el cálculo por medio de aproximación de valores tomando en cuenta la Tabla.

Tabla 8.
Valores PID según Ziegler-Nichols

Tipo de control	Kp	Ki	Kd
P	0,50 K c	0	0
PI	0,45 K c	1,2 K p dT / P c	0
PID	0,60 K c	2 K p dT / P c	K p P / c (8DT)

Fuente: Realizado por el autor

Según se puede observar el primer paso para identificar los componentes proporcional integral y derivativo son igualar a 0 tanto Ki como Kd y asignar a Kp el máximo valor para la velocidad del motor que se requiere para la planta en este caso por ser un motor DC se tendrá valores de entre 0 a 255, que por motivos de estabilidad, se selecciona 80. El valor Kc es igual a Kp en su punto máximo, es decir tomando en cuenta la máxima velocidad del motor requerida y el error máximo que se podrá tener en este caso (-20, 20). Obteniendo lo siguiente:

Kp= (80)/(20)

Kp= 4 //se toma el valor absoluto

Con esto se puede elaborar un valor proporcional para el movimiento del motor que vendrá dado por la polaridad sea negativa para izquierda y positiva para la derecha, el movimiento viene dado por:

Movimiento = Kp*error_actual

Tomando en cuenta el error actual como la variación de la posición de la esfera del punto de consigna o 0.

Una vez obtenido el valor proporcional se procede a obtener el valor integral que va en referencia a la acumulación del error en el tiempo del proceso de la siguiente forma:

Integral = integral +error_actual

Para la obtención de estos valores se procede a obtener tres valores de error variables en el tiempo los cuales son, recordando que el signo solo indica la dirección por lo cual se usara el valor absoluto:

1er error_actual=20

2do error_actual=-17

Por lo cual el integral es la sumatoria de los errores, entonces:

Integral=20+17

Integral = 37

Para despejar Ki se toma en cuenta la formula general de PID que es la sumatoria de todos los componentes, así:

Movimiento= Kp*error_actual+Ki*integral

Tomando en cuenta que kp=4; error_actual=10; integral=13, y el movimiento del primer error el cual es 80 , y se reemplaza:

80 = 4*17+Ki*37

80-68=37ki

12=37ki

Ki=0,32

De esta forma se obtiene los valores proporcional e integral por lo cual en base al método de ziegler se procede a realizar el cálculo del componente derivativo que se toma en base a predecir el error futuro, algo sumamente complejo pero simplificable para el caso de la siguiente forma tomando en cuenta valores pasados para predecir un valor futuro:

Derivada = error_actual-error_pasado

Derivada=17-20

Donde se tiene que el error actual es 17 y el error pasado es 20

Derivada = -3

Ahora es necesario encontrar el error siguiente con la siguiente formula:

Error_futuro=error_actual+derivada

Error_futuro=17-3

Error_futuro=14

Una vez calculado el error futuro es necesario reemplazar los valores, así:

Derivada=error_futuro-error_actual

Derivada=14-17

Derivada=-3

De donde se reemplaza en la fórmula de Movimiento y se obtiene el valor de Kd:

Movimiento= kp*error_actual+ki*integral+kd*derivada

80=4*17+0,32*37+kd*-3

12=11,84-3kd

0,16/3=kd

Kd=0,053

En caso de que estos valores no se adapten al sistema Ziegler-Nichols indica que se realizara una aproximación a un punto de error mínimo o medio para de esta forma encontrar el valor perfecto de oscilación en base a las formulas presentadas tomando en cuenta un error_actual igual a su máximo 20, medio 10 o mínimo 1.