ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO DOMÓTICO POR MEDIO DE *RASPBERRY PI* A TRAVÉS DE UNA INTERFAZ WEB Y *TELEGRAM*

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES

Miguel Ángel Basantes Farinango
miguel.basantes@epn.edu.ec

DIRECTOR: ING. LEANDRO ANTONIO PAZMIÑO ORTIZ, MSc.
leandro.pazmino@epn.edu.ec

CODIRECTOR: ING. FABIO MATÍAS GONZÁLEZ GONZÁLEZ, MSc.
fabio.gonzalez@epn.edu.ec

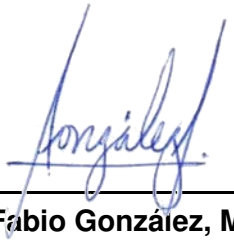
Quito, octubre 2021

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por el Sr Miguel Ángel Basantes Farinango como requerimiento parcial a la obtención del título de TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES bajo nuestra supervisión:

Ing. Leandro Pazmiño, MSc.

DIRECTOR DEL PROYECTO



Ing. Fabio González, MSc.

CODIRECTOR DEL PROYECTO

DECLARACIÓN

Yo Miguel Ángel Basantes Farinango con CI: 1727156471 declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

Sin prejuicio de los derechos reconocidos en el primer párrafo del artículo 144 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación – COESC-, soy titular de la obra en mención y otorgo una licencia gratuita, intransferible y no exclusiva de uso con fines académicos a la Escuela Politécnica Nacional.

Entrego toda la información técnica pertinente, en caso de que hubiese una explotación comercial de la obra por parte de la EPN, se negociará los porcentajes de los beneficios conforme lo establece la normativa nacional vigente.

Miguel Ángel Basantes Farinango

DEDICATORIA

A mis padres, por haberme brindado todo su apoyo durante el transcurso de mis estudios en la Escuela de Formación de Tecnólogos y en la realización del presente proyecto sin su ayuda emocional, espiritual y financiera jamás hubiera podido completar un proyecto de esta magnitud yo solo. Son un pilar fundamental en mi vida junto con mi hermana. Me enseñan constantemente valores y principios que me ayudan a ser muy dedicado, perseverante y asertivo en todos mis objetivos.

AGRADECIMIENTO

Agradezco al Ing. Leandro Pazmiño por su apoyo en el desarrollo proyecto, aprecio su confianza y tiempo. También a la Ing. Viviana Párraga por su ayuda en el planteamiento del tema y doy las gracias a todos los demás docentes que he conocido a lo largo de mi estancia en la Escuela de Formación de Tecnólogos que supieron impartirme sus conocimientos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Inicialmente se halla la introducción del escrito, en la cual se define y justifica la elaboración del proyecto, adicionalmente se destaca los objetivos. Además, se indican los fundamentos teóricos en base a los cuales se sustenta el proyecto, estos abarcan conceptos sobre: la *Raspberry Pi*, lenguajes de programación, las interfaces de comunicación e información relevante que ayudaron al desarrollo del proyecto.

Luego se tiene la metodología del proyecto, en esta sección se define el proceso que se llevó a cabo para cumplir cada uno de los objetivos específicos, asimismo, se detalla la forma en la que se construyó el prototipo de comienzo a fin.

Después se hallan los resultados y discusión, en esta se parte se establece los requerimientos que fueron imprescindibles para elaborar el proyecto como son: el *software*, el *hardware* y su configuración en la *Raspberry Pi,* también se da a conocer el desarrollo de los *scripts* con su respectivo diagrama de flujo. Adicionalmente, se establecieron los circuitos que fueron requeridos y la creación de las piezas en 3D de los componentes junto con la implementación del prototipo. Por otro lado, se añaden los resultados obtenidos al efectuar las pruebas de funcionamiento, el costo estimado del proyecto y los manuales.

A continuación, se encuentran las conclusiones y recomendaciones, las mismas que fueron definidas en base a la experiencia obtenida en todo el proceso de elaboración del prototipo. Seguidamente, se adjunta la bibliografía con la que está fundamentado el marco teórico.

Por último, se añade la documentación correspondiente a los anexos por medio de códigos QR como respaldo de lo realizado.

PALABRAS CLAVE: *Raspberry Pi*, *Python, script*

ABSTRACT

In the first place is the introduction of the document, which defines and justifies the development of the project, also in this there is additional information on the objectives. In addition, the theoretical foundations are highlighted, which include concepts on the Raspberry Pi, programming languages, communication interfaces and relevant information that helped the development of the project.

Later, there is the project methodology, in this part the process that was carried out for each of the specific objectives is described in which the way in which the prototype was elaborated from start to finish is explained in detail.

After the results and discussion are found, this section mainly establishes the requirements that were essential to implement the project in both software and hardware and its configuration on the Raspberry Pi, there is also the development of the scripts with their respective diagram flow. Additionally, the circuits that were required and the creation of the 3D parts of the components were established together with the implementation of the prototype. On the other hand, the results of the performance tests, the estimated cost of the project and the manuals are added.

Below are the conclusions and recommendations, which were raised based on the experience acquired throughout the prototype development process. Likewise, the bibliographic references by which the theoretical framework was based are attached.

Finally, the documentation of the annexes is included by means of QR codes to support what has been done.

KEYWORDS: Raspberry Pi, Python, script

1 INTRODUCCIÓN

Actualmente una gran parte de la población ha tenido que migrar a los medios tecnológicos debido al SARS.CoV-2 por lo cual varias actividades que se realizaban de forma manual han pasado a realizarse en línea. El teletrabajo ha pasado a ser uno de los medios más empleados en el país y muchas empresas lo están considerando a largo plazo. Además, se ha incrementado el anhelo de mantenernos comunicados a través de llamadas, mensajes y video llamadas; lo que ha facilitado la entrada del Internet a todos los ámbitos de la vida.

Esta situación se presenta como una oportunidad adecuada para integrar el Internet de las cosas (IoT) a objetos comunes como bombillas, computadoras, etc. De manera que simplifiquen y disminuyan las tareas que se realizan en el hogar. El control de estos objetos en un mismo lugar puede reducir y optimizar el tiempo, para así garantizar su productividad. Adicionalmente, la automatización de procesos genera la posibilidad de un mayor control a través de sensores o establecimiento de una hora fija para realizar cierta actividad repetitiva como encender o apagar la luz.

El manejo de estos procesos y la fácil integración con APIs permite crear hoy por hoy un sistema complejo, pero a la vez intuitivo con los usuarios. Dadas estas razones se plantea el prototipo, con el fin de dar una solución práctica a varias actividades como: el control de luces, cortinas, computadoras y puerta.

1.1 Objetivo general

Implementar un prototipo domótico por medio de *Raspberry Pi* a través de una interfaz *web* y *Telegram*

1.2 Objetivos específicos

- Realizar un estudio de requerimientos para el prototipo domótico.
- Diseñar el sistema de control de los dispositivos.
- Crear la interfaz *web* para la comunicación entre el usuario y el sistema.
- Instalar los elementos que componen el sistema.
- Probar el prototipo domótico

1.3 Fundamentos

Raspberry Pi

Es una computadora de bajo costo y tamaño reducido, en esta se puede conectar distintos periféricos, sin embargo, se tiene la facilidad de añadir componentes electrónicos [1].

Modelos *Raspberry Pi*

Todos los modelos *Raspberry Pi* tienen una gran cantidad de entradas y salidas digitales para conectar diversos componentes electrónicos, estos se conocen como pines *GPIO* (*General Purpose Input/Output*). En la Figura 1.1 se muestra la distribución de los pines según los modelos existentes [2].

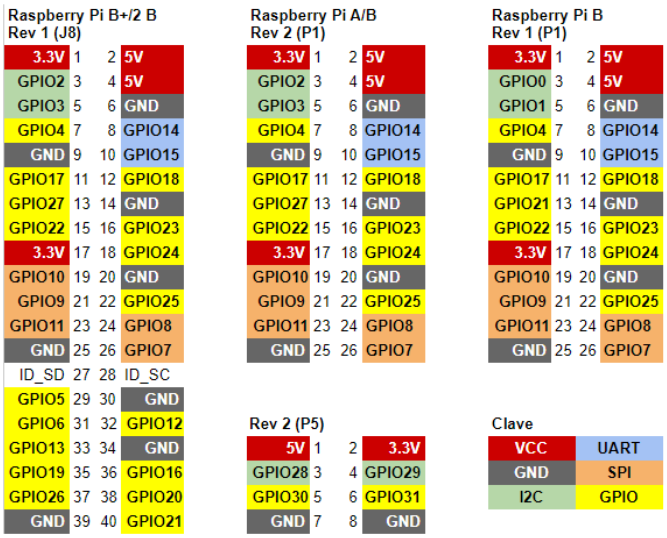


Figura 1.1 Comparación de los pines de E/S de los distintos modelos [2]

Motor a pasos

Es un dispositivo capaz de convertir una serie de pulsos eléctricos en desplazamientos angulares [3].

Engranaje

Los engranajes están constituidos por varias ruedas dentadas unidas con el propósito de transmitir la velocidad de rotación [4]. En la Figura 1.2 se puede visualizar las partes de una rueda dentada.

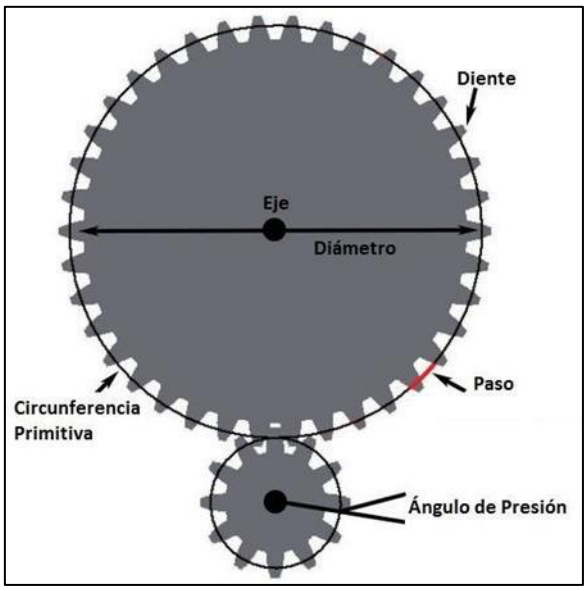


Figura 1.2 Partes de una rueda dentada [4]

Módulo relé

Es una placa que consta de uno o varios relés, además de otros elementos electrónicos como: resistencias, diodos, transistores, entre otros [5].

A4988

Es un dispositivo muy utilizado para el control de motores paso a paso bipolares, este es capaz de manejar una corriente de hasta 2 (A) de manera muy eficiente [6].

Pantalla *7inch HDMI LCD (C)*

Este componente cuenta con resolución de 1024 × 600, además posee un panel táctil capacitivo y se puede utilizar como monitor para un ordenador [7].

Interfaces de comunicación

- *Comunicación I2C*

Es un bus de comunicación que constan de dos hilos, el primer hilo es para la línea de datos (SDA) y el segundo hilo es para la señal de reloj (SCL) [2]. En la Figura 1.3 se señala a modo de ejemplo el uso de *I2C* en la *Raspberry Pi* con una pantalla LCD.

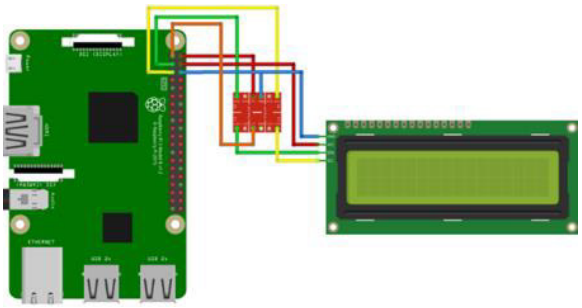


Figura 1.3 Pantalla LCD con *I2C* conectada a la *Raspberry* Pi [8]

- *Comunicación SPI*

SPI es un protocolo de comunicación que permite manejar una menor cantidad de dispositivos a diferencia de *I2C*, pero a una velocidad superior [2]. En la Tabla 1.1 se detalla la función de cada pin. Al igual que en *I2C*, hay un maestro que toma el rol principal en la comunicación y uno o varios esclavos que asumen el rol secundario, tal cual se mira en la Figura 1.4

Tabla 1.1 Pines para la comunicación *SPI* [2]

Nombre	Descripción
SCLK	Señal de reloj respecto a la que se sincronizan el resto de las señales.
MISO	Entrada de datos para el maestro, salida de datos para el esclavo.
MOSI	Salida de datos para el maestro, entrada de datos para el esclavo.
SS o *CS*	Una o varias señales de selección de destino activas a nivel bajo.

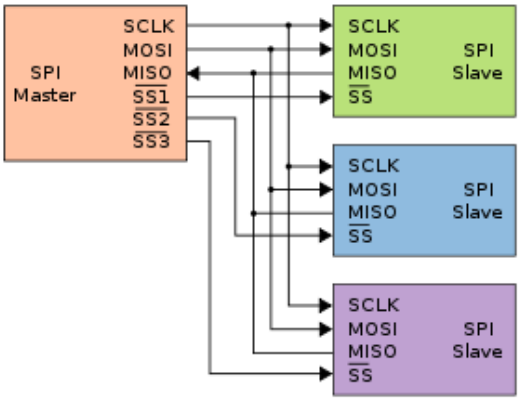


Figura 1.4 Bus *SPI* [9]

- *Comunicación UART*

La comunicación *UART* o *serial* consta de 2 señales digitales (TxD), para la transmisión de datos y (RxD) para la recepción de datos. Actualmente es casi imposible encontrar esta interfaz en un ordenador moderno, no obstante, su presencia es dominante en los microcontroladores de gama baja y en ordenadores antiguos [2].

Sin embargo, en la *Raspberry Pi* puede ser útil para mantener la conexión con un microcontrolador u otro periférico tal cual se puede apreciar en la Figura 1.5.

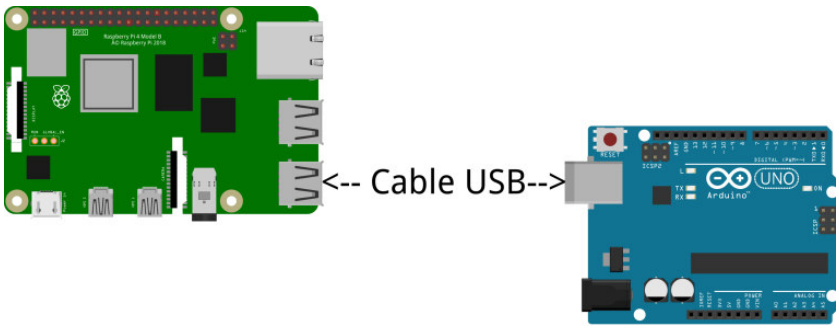


Figura 1.5 Comunicación entre *Raspberry Pi* y *Arduino* [10]

Cortafuegos *UFW*

Es un *front-end* para administrar las reglas de *firewall* en: *Arch Linux*, *Debian* o *Ubuntu*. *UFW* se usa a través de la línea de comando, aunque también tiene una *GUI* disponible, su propósito es hacer que la configuración del *firewall* sea simple [11].

Python

Es un lenguaje de programación de fácil interpretación y con una curva rápida de aprendizaje [12].

Pines en el modo *BOARD* y *BCM* en la *Raspberry Pi*

Existen dos modos de notación para controlar a los pines *GPIO* con *Python, GPIO.BOARD* y *GPIO.BCM*. La opción *GPIO.BOARD* son los números impresos en la *Raspberry Pi*, tal como se ilustra en la Figura 1.6 [13]. Por otro lado, *GPIO.BCM* corresponde a los pines según su número de (*Broadcom SOC channel*).

Figura 1.6 Pines en modo *GPIO.BOARD* [13]

Protocolo *NEC*

Es uno de los muchos protocolos de control remoto por infrarrojos, el cual utiliza pulsos para la transmisión del mensaje. Una ráfaga de pulsos dura 562,5 (µs) con una frecuencia de 38 (KHz) [14].

En la Figura 1.7 se ilustra perfectamente a modo de ejemplo el mensaje enviado al pulsar una tecla de un control remoto.

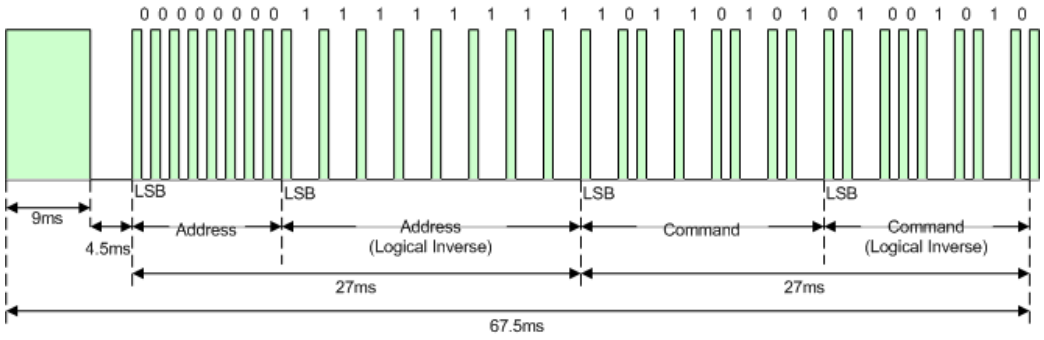


Figura 1.7 Mensaje transmitido en la pulsación de una tecla [14]

API Telegram Bot

Es una interfaz basada en *HTTP*, elaborada para desarrolladores interesados en crear *Bots* para *Telegram* [15].

Programación de tareas repetitivas del sistema (*Cron*)

Sirve para programar tareas en un sistema *Linux* por medio de los comandos colocados en el archivo *crontab* [16].

rc.local

Sirve para llevar a cabo una ejecución de un comando o *script* cada vez que se inicia el sistema operativo *Linux* [17].

Modulación de ancho de pulso (*PWM*)

Es una tecnología de control de voltaje digital eficiente, que utiliza la salida digital de un microprocesador para controlar un dispositivo. Además, permite variar la frecuencia de trabajo para ajustar el voltaje que circula a través de la carga de manera que se logra un control óptimo [18].

Apache

Está desarrollado para mantener un servidor HTTP de código abierto con el propósito que exista compatibilidad entre los distintitos sistemas operativos que lo usan como por ejemplo: *UNIX* y *Windows* [19].

phpMyAdmin

Es una herramienta diseñada en lenguaje *PHP* con el propósito de controlar la administración de *MySQL* por medio de páginas web [20].

CSS

Sirve para dar diseño a una página web. En la Figura 1.8 se nota que los documentos *HTML* y *CSS* pueden trabajar juntos para formar un solo documento. Sin embargo, si se necesita hacer modificaciones visuales se tiene la ventaja de cambiar el código en un solo lugar.

Figura 1.8 Documento *HTML* y *CSS* [21]

JavaScript

Sirve para realizar contenido de actualización dinámica, manejar multimedia, animar imágenes, entre otros [22].

JQuery

Es una librería que ayuda a programar lenguaje *JavaScript* ya que permite crear interactividad a un sitio web de manera sencilla. Esta se puede descargar del enlace https://jquery.com/, tal cual se observa en la Figura 1.9.

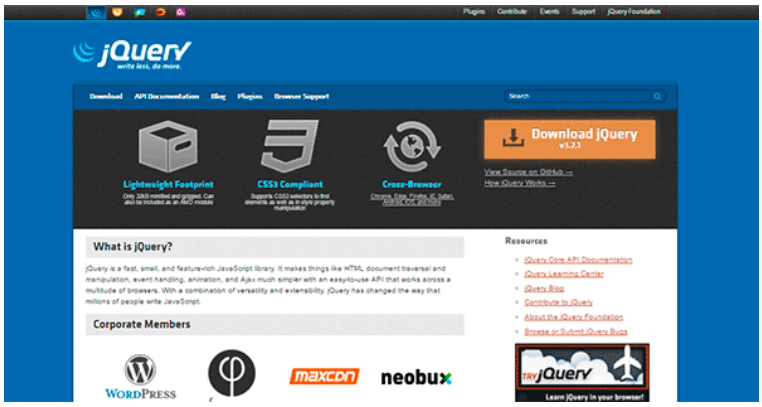


Figura 1.9 Librería *JQuery* [23]

Encriptación *MD5*

Es un algoritmo para codificar de 128 bits con 32 caracteres hexadecimales, utilizado para encriptar archivos y contraseñas en una base de datos [24].

Web Speech API

Esta *API* cuenta con dos funcionalidades distintas, el reconocimiento de voz, y la síntesis de voz (también conocido como texto a voz). Actualmente, la compatibilidad de *Web Speech API* para el reconocimiento de voz se limita a *Chrome* para escritorio y *Android Chrome* desde la versión 33 [25].

AJAX

Es ocupado en el desarrollo web para enviar y recuperar datos del servidor sin la necesidad de volver a cargar nuevamente toda la página [26].

2 METODOLOGÍA

2.1 Descripción de la metodología usada

Objetivo 1:

Para determinar los requerimientos del sistema se realizó una investigación sobre las distintas áreas que formaron el prototipo domótico. En la parte del *hardware* principalmente se empleó el microcomputador *Raspberry Pi* para el control de todos los procesos, por lo cual fue fundamental conocer sus características y modelos existentes. Para el *software* se analizó el tipo de sistema operativo, este debía ser capaz de tener una interfaz gráfica ya que se deseaba representar datos a través de una pantalla táctil, además de satisfacer todos los requerimientos y necesidades que tuvo el proyecto.

Se requirió saber más acerca de los protocolos y estándares a emplearse en la comunicación entre los diferentes dispositivos y la *Raspberry Pi*; por ende, se analizó algunas formas de hacerlo tales como *SPI*, *I2C*, *Serial*, entre otras. Se identificaron los puertos y servicios útiles para la creación del servidor *Web*, así como los que permiten comunicarse y compartir archivos. Mediante este análisis se pudo tener un mayor control del sistema y con la ayuda de un cortafuego se pudo evitar posibles brechas de seguridad.

Por último, se llevó a cabo una recopilación de todos los datos obtenidos de la investigación y se seleccionó el *hardware, software*, *APIs* y componentes electrónicos necesarios para el proyecto.

Objetivo 2:

A fin de diseñar la parte de mando se analizó primeramente las actividades que el sistema será capaz de llevar a cabo. Dando como resultado que el sistema efectuará cuatro actividades principales: control de los ordenadores, puerta, cortinas y luces. Estas tareas se llevan a cabo mediante una interfaz web la cual cuenta también con una base de datos de los usuarios que pueden ingresar, una vez dentro, el usuario tiene la posibilidad de: controlar manualmente el sistema, habilitar su uso a través de un control remoto y dejar automatizadas las cortinas, la puerta y las luces.

Por otra parte, mediante la *API* de *Telegram*, con el desarrollo de un *Bot* se tendrá el control completo de todas las funcionalidades del sistema y procesos básicos de la *Raspberry* enfocado al mantenimiento del prototipo y la seguridad.

En base a los requerimientos del prototipo se empleó varios programas residentes de *Linux,* entre los empleados están *Systemd* y *Cron.* Estos fueron muy útiles para el lanzamiento automático de la pantalla del navegador *Google Chrome* en modo quiosco de manera que nadie pueda intervenir en los archivos del sistema o cerrar la pantalla del navegador, habilitación de la ventilación del prototipo, ejecución del *bot* de *Telegram* y otros programas que se lanzan al iniciarse la *Raspberry Pi*.

Además, se construyeron los circuitos electrónicos necesarios en el programa *Fritzing.* También fue fundamental crear las piezas de los componentes, la estructura del prototipo y el sistema de engranajes. Para este fin se examinó cada elemento para obtener sus dimensiones y en el caso de los engranajes además se realizaron varios cálculos para saber las RPM que se podrían obtener mediante la caja de reducción creada. Con toda esta información se diseñó mediante *Adobe Autodesk Fusion* 360 todas las piezas y se generó los archivos .stl que más adelante servirían para la impresión 3D.

Por otra parte, para tener una visión más clara de la implementación del prototipo se diseñó en 3D un plano de un recinto domiciliario mediante *DIALux EVO* en el cual se colocó luminarias, computadoras, cortinas, puertas, el prototipo domótico, sensores, entre otros elementos para hacer más realista la simulación. El diseño elaborado en 3D en *DIALux EVO* se exportó en forma de plano a *AutoCAD* en donde se ubicó los componentes de Cableado Estructurado.

Objetivo 3:

Para crear la interfaz web, primero se instaló el sistema operativo, posteriormente se eligió un tipo de servidor compatible con esta distribución para lo cual se examinó entre los servidores *LAMPP*, *WAMPP* y *XAMPP*. Dado que el servidor *WAMPP* es netamente para sistemas operativos *Windows* quedó totalmente descartado; por otro lado, el servidor *LAMPP* es únicamente para sistemas *Linux* y ofrece mayor estabilidad en los mismos por lo cual se eligió este, aunque el servidor *XAMPP* se lo puede emplear en todos los sistemas operativos lo que fue de mucha ayuda al momento de desarrollar y probar las páginas web en *Adobe Dreamweaver*, ya que este programa se lo tenía en un ordenador.

Al momento de instalar el servidor fue necesario conocer sobre los elementos que lo componen *LAMPP= Linux + Apache + MySQL + PHP + Perl*, en donde: *Linux* es el sistema operativo, *Apache* el servidor web, el cuál es robusto y gratuito de manera que presenta una gran seguridad al momento de ejecutar los servicios en Internet, *MySQL* es la base de datos en donde se guardó la información de los usuarios y *PHP* o *Perl* los lenguajes de programación que permiten ejecutar acciones. Por otro lado, también se tuvo que agregar *phpMyAdmin* para la fácil administración de la base de datos.

La gran mayoría de las páginas se desarrollaron en *HTML5*; por otra parte, el diseño y estilo de estas fue mediante *CSS*. También se les agregó dinamismo mediante un poco de *JavaScript* con la librería *JQuery*, la cual fue fundamental para ejecutar los *scripts* de *Python* mediante *PHP*.

Dado que las imágenes en un sitio web son importantes se analizó que para el proyecto era necesario emplear *Photoshop* para optimizar su peso y ajustar su tamaño para que la página web no tarde mucho en cargar o la presentación de esta no sea mala.

Adicionalmente se requirió que la interfaz web sea compatible con cualquier dispositivo que cuente con acceso a un navegador por lo que el tamaño se adaptó principalmente para: pantallas de computadoras, *laptops*, pantalla del prototipo y *smartphones*.

Una vez acabadas las páginas web, los archivos que se generaron se pasaron a la *Raspberry Pi* en la ubicación /var/www/html mediante el programa *WinScp* el cual se vale del protocolo *SFTP*, la dirección IP del servidor, nombre de usuario, contraseña y puerto 22 para transferir la información. Otros programas ocupados fueron: *VNC Viewer* para obtener gráficamente la pantalla de la Raspberry en el ordenador con la finalidad de grabar la pantalla para los manuales y *PuTTY* que a través de *SSH* se puede ejecutar algún comando desde la terminal.

Objetivo 4:

Para instalar los elementos que conforman el prototipo; primeramente, con la ayuda de un multímetro se comprobó que existiera continuidad entre las pistas, que estuvieran correctas y entonces se soldó a las placas los componentes electrónicos. Después, se armó la estructura con las piezas impresas en 3D. Posteriormente, se posicionó de manera adecuada los sensores y los motores de las cortinas. Más adelante, se colocó a los ordenadores y al prototipo los componentes de red para tener conectividad entre dispositivos.

Objetivo 5:

Se hicieron pruebas de validación del funcionamiento del dispositivo y se identificaron las fallas de este para solventarlas. Finalmente se desarrollaron dos videos manuales en donde se tuvo que editar, modificar y crear audio, video e imágenes con el fin de hacerlos más interactivos, para este propósito se empleó principalmente la aplicación *Wondershare Filmora9* junto con otros editores de multimedia. El primero es el video manual de uso en donde se especifica como el usuario puede manejar el prototipo de manera adecuada. El segundo en cambio es el video manual de mantenimiento del prototipo en donde se puede encontrar los componentes que fueron empleados y sus características con el objetivo de reemplazarlos si existiera algún daño.

3 RESULTADOS Y DISCUSIÓN

En esta parte se detalla la forma en la que se identificaron los requerimientos del prototipo domótico, los diseños e implementación de cada elemento del proyecto tanto de *software* como *hardware*, además se halla la elaboración de los códigos de programación y por último se detallan las pruebas de funcionamiento.

3.1 Identificación de los requerimientos para implementar el proyecto

Selección del modelo *Raspberry Pi*

Para la selección del modelo a emplearse, se tomó en consideración la versatilidad del dispositivo y la complejidad del proyecto. Además, debido a los requerimientos del prototipo se necesitaron que este cuente con varias entradas y salidas para la conexión de sensores, *leds*, integrados, elementos electrónicos y *shields*. En la Tabla 3.1 se observan algunas de las principales diferencias entre los modelos existentes.

Tabla 3.1 Comparativa entre modelos [27]

Modelo	CPU	RAM	Conectividad Inalámbrica	Puertos E/S
Raspberry Pi 4B	1,5 (GHz), 4-*core* *Broadcom* BCM2711 (Cortex-A72)	2/4/8 (GB)	802.11ac / *Bluetooth* 5.0	2 x *USB* 3.0, 2 x *USB* 2.0, 1 x *Gigabit Ethernet*, 2 x micro-*HDMI*
Raspberry Pi 3B +	1,4 (GHz), 4-*core* *Broadcom* BCM2837B0 (Cortex-A53)	1 (GB)	802.11ac/ *Bluetooth* 4.2	4 x *USB* 2.0, 1 x *Ethernet*, 1 x mini-*HDMI*
Raspberry Pi Zero W	1 (GHz), 1-*core* *Broadcom* BCM2835 (ARM1176JZF-S)	512 (MB)	802.11n / *Bluetooth* 4.1	1 x micro-*USB*, 1 x mini-*HDMI*
Raspberry Pi Zero WH	1 (GHz), 1-*core* *Broadcom* BCM2835 (ARM1176JZF-S)	512 (MB)	802.11n / *Bluetooth* 4.1	1 x micro-*USB*, 1 x mini-*HDMI*
Raspberry Pi Zero	1 (GHz), 1-*core* *Broadcom* BCM2835 (ARM1176JZF-S)	512 (MB)	No posee	1 x micro-*USB*, 1 x mini-*HDMI*

En comparación, *Raspberry Pi 4B* ofrece un mejor rendimiento de su procesador, muy parecido a la de su antecesor *Raspberry Pi* 3B + pero su mayor ventaja es su velocidad de comunicación tanto por *Ethernet*, *Wi-Fi* y los puertos *USB*. El puerto *Ethernet* ya no está saturado, como el modelo anterior debido a que se tiene una conexión *Gigabit* en la que no entra en juego el cuello de botella del procesador y, sobre todo, del propio RJ45 [28].

Por otra parte, la conectividad inalámbrica es con 802.11ac de manera que es compatible con 802.11b, g y n. Esto significa que funcionará muy bien con la mayoría de los dispositivos actuales y se podrá conservar por un buen tiempo.

También, el espectro de los 5 (GHz) suele ser más silencioso, ya que sufre muchas menos interferencias de las señales *Wi-Fi* vecinas. Y debido, a que en 802.11ac los haces detectan dónde están los dispositivos, e intensifican la señal en su dirección como se muestra en la Figura 3.1 [29].

Figura 3.1 802.11n y 802.11ac [29]

En el caso de la RAM la *Raspberry Pi 4* viene con 2 (GB), 4 (GB), u 8 (GB). Pero para la mayoría de los proyectos de aficionados, y para usar como computadora de escritorio, 2 (GB) son suficientes. Desde otro punto de vista la *Raspberry Pi Zero*, *Raspberry Pi Zero W* y *Raspberry Pi Zero WH* son más pequeñas y requieren menos energía, por lo que son útiles para proyectos portátiles y en algunos casos donde no se requiera conectividad inalámbrica [30].

Como principal desventaja la *Raspberry Pi 4* alcanza una temperatura muy alta, comparado con la temperatura de la *Raspberry Pi* 3 que tras 10 minutos de ocupación es de 62,6º, en cambio en la *Raspberry Pi 4* es de 74,5º, es decir 12 grados más. De manera que es necesario comprar un disipador [28].

Por estas razones y en función de los requerimientos del prototipo se optó por la placa de desarrollo *Raspberry Pi 4B* de 2 (GB) con conector J8 de 40 pines, debido a su gran capacidad de entradas y salidas digitales. También porque sus ventajas superan la desventaja de la temperatura ya que, además, cuenta con varias interfaces de comunicación y los recursos antes mencionados.

Conexión del *hardware* de la *Raspberry Pi 4*

Para esto se clasificó toda la información encontrada en la investigación, de esta manera se logró identificar los componentes necesarios y su conexión, la cual se muestra en la Figura 3.2.

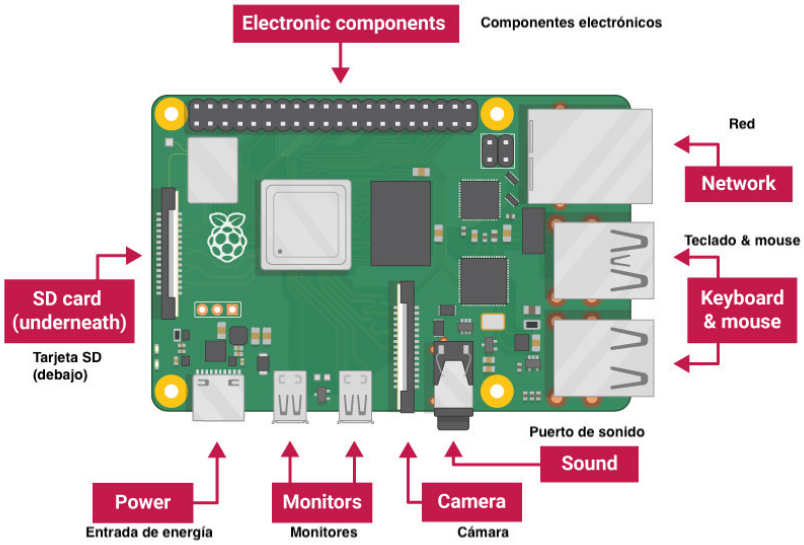


Figura 3.2 *Hardware* de la *Raspberry Pi 4* [30]

- *Fuente de alimentación*

Se eligió una fuente de alimentación que proporcione 3 (A_{DC}), con una salida tipo *USB-C* (la misma que se encuentra en muchos teléfonos móviles) como se aprecia en la Figura 3.3. En caso de usar otro modelo, emplear una fuente de alimentación de 2,5 (A_{DC}) con salida micro *USB*.

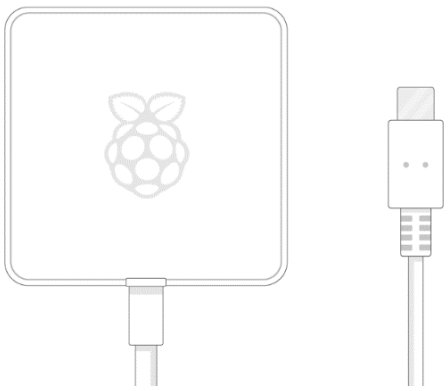


Figura 3.3 Fuente de alimentación *Raspberry Pi 4* [30]

- *Tarjeta SD*

La *Raspberry Pi* mediante la tarjeta SD almacena los archivos y el sistema operativo. Se puede emplear una tarjeta SD desde una capacidad de al menos 8 (GB) hasta una de 32 (GB) sin configuraciones adicionales. Sin embargo, una tarjeta con muy poco almacenamiento se quedará sin espacio rápidamente, por lo que se eligió una tarjeta SD de 32 (GB).

Si por algún motivo se desea usar una de mayor capacidad como por ejemplo una tarjeta de 64 (GB) se tendrá que formatear el sistema de archivos *exFAT* a *FAT16* o *FAT32* como se observa en la Figura 3.4, ya que la *Raspberry Pi* solo soporta la lectura de los 2 anteriores. Esto se puede realizar empleando el *software* fat32format [31].

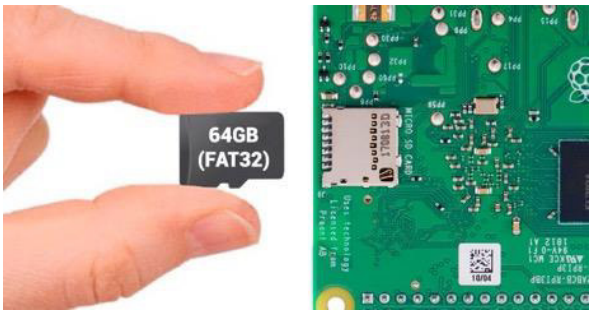


Figura 3.4 Colocación de la tarjeta microSD [31]

- *Teclado y ratón*

Se conectó un teclado y ratón *USB* para las primeras configuraciones, pero una vez establecidas, es factible utilizar un teclado y ratón *Bluetooth*. Aunque a través de *SSH* se mantuvo conexión remota con la *Raspberry Pi* desde un ordenador, por lo que es aceptable saltar su uso si no se cuenta con estos componentes.

- *Monitor*

En términos generales, para ver el entorno gráfico del sistema operativo es indispensable una pantalla, esta puede ser de un televisor, monitor de computadora, etc. Adicionalmente, si la pantalla tiene altavoces integrados, *Raspberry Pi* puede usarlos para reproducir sonido. Para la conexión de la pantalla con la *Raspberry Pi 4* asegurarse de haber utilizado el puerto *HDMI0* (el más cercano al puerto de entrada de energía) en lugar del puerto *HDMI1*, debido a que la *Raspberry Pi 4* cuenta con la opción de conectar una segunda pantalla como se indica en la Figura 3.5.

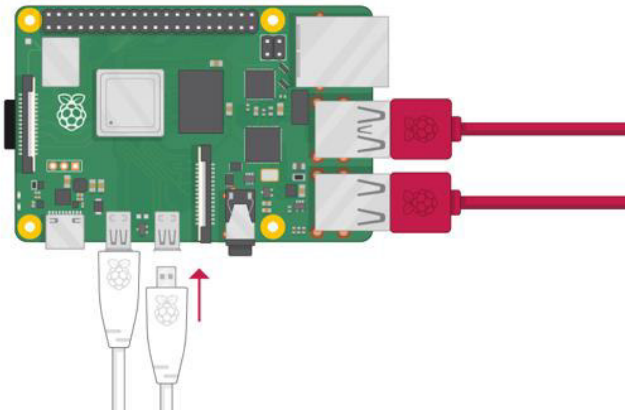


Figura 3.5 Puertos *HDMI* de la *Raspberry Pi 4* [30]

Por otro lado, puede ser útil valerse de un adaptador ya que muchos monitores tienen puertos *DVI* o *VGA*. Es importante destacar que el prototipo requería emplear una pantalla táctil por lo que se usó la pantalla *7inch HDMI LCD (C)*. Sin embargo, puede ser muy apropiado disponer de *VNC* si desea observar la interfaz gráfica.

- *Enlace de par trenzado*

Se ocupó el enlace de par trenzado UTP categoría 5e para tener acceso a *Internet* de manera cableada. A pesar de que también se configuró la conectividad inalámbrica. En la Figura 3.6 se observa la conexión de un enlace de par trenzado con la *Raspberry Pi*.

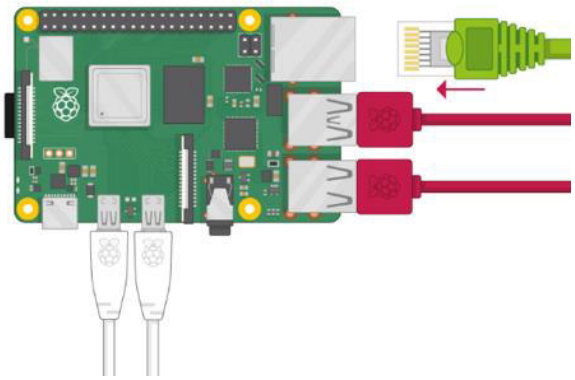


Figura 3.6 Conexión del enlace de par trenzado [30]

- *Altavoces y Cámara*

Los modelos grandes de *Raspberry Pi* (pero no *Raspberry Pi Zero* o *Raspberry Pi Zero W*) tienen un puerto de audio estándar como el que se encuentra en teléfonos inteligentes o en reproductores MP3.

Si se desea, se puede conectar los auriculares o altavoces para que la *Raspberry Pi* pueda reproducir sonido como se nota en la Figura 3.7.

En el caso de la cámara hay varias posibilidades, la primera es emplear la cámara oficial de la *Raspberry Pi* o cualquier otro tipo de cámara que funcione a través de *USB*, *Wi-Fi*, como las cámaras IP, etc.

No obstante, estos dispositivos no fueron de relevancia en el desarrollo del prototipo, pero en versiones futuras pueden ser de gran utilidad.

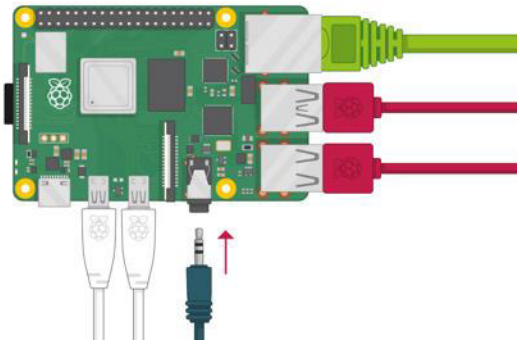


Figura 3.7 Conexión de auriculares o altavoces [30]

- *Carcasa*

Mediante la implementación de la carcasa se proporcionará protección a la *Raspberry Pi*. La carcasa que se ocupó es de aluminio, esta funciona como disipador de calor manteniendo las temperaturas de trabajo aceptables. En la Figura 3.8 se aprecia que la carcasa cuenta con 2 ventiladores pequeños los cuales funcionan a 5 (V) y se conectan a los pines 4 y 6 de la *Raspberry Pi*. Debido a que la disposición del conector *HDMI* es distinto para la *Raspberry Pi 4*, este gabinete no es compatible con versiones anteriores.

Figura 3.8 Carcasa de aluminio tipo disipador de calor para *Raspberry Pi 4* [32]

Selección del sistema operativo

Hay muchos sistemas operativos disponibles para *Raspberry Pi,* dependiendo la función que se vaya a desarrollar se puede tener sistemas operativos que funcionen como emuladores de videojuegos, reproductores de multimedia, impresión 3d, servidores, entre otros. Aunque, *Raspberry Pi OS* es el sistema operativo oficial, este cuenta con la versión *Lite* que no posee ambiente de escritorio por lo que es más ligera y la *Full* que si cuenta con interfaz gráfica. Ambas distribuciones están basadas en *Linux*, principalmente en *Debian*. Sin embargo, se pueden instalar sistemas operativos de terceros como se visualiza en la Figura 3.9. Dadas las características anteriores se prefirió el sistema *Raspberry Pi OS* que cuenta con interfaz gráfica.

Figura 3.9 Sistemas operativos para *Raspberry Pi* [33]

Instalación del sistema operativo

La forma en la que se instaló el sistema operativo *Raspberry Pi OS* en la tarjeta SD es a través del emulador de imágenes de *Raspberry Pi*. Primero se ingresó a la página oficial de *Raspberry Pi* (https://www.raspberrypi.org/software/), como se muestra en la Figura 3.10, una vez dentro se descargó el emulador para el sistema operativo *Windows.*

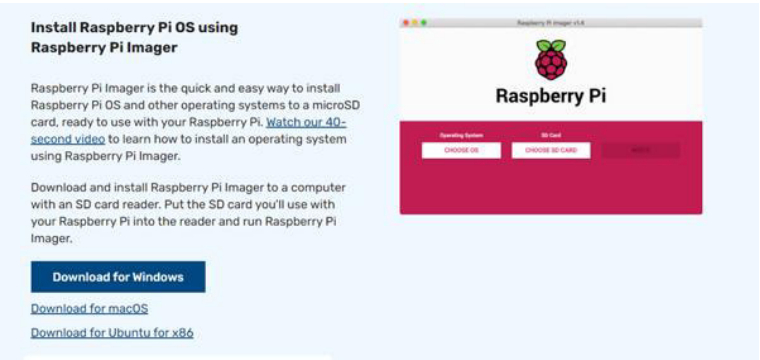


Figura 3.10 Descarga del emulador de imágenes de *Raspberry Pi*

Posteriormente, se preparó la tarjeta SD para la instalación ingresándola en el ordenador. Es importante saber que todo lo que esté almacenado en la tarjeta SD se sobrescribirá durante el formateo. Si la tarjeta SD contiene archivos, es buena idea hacer una copia de seguridad para evitar perderlos permanentemente.

Después, se ejecutó el programa descargado, lo cual se observa en la Figura 3.11. Este cuenta con tres opciones que nos permiten elegir el tipo de sistema operativo, tipo de tarjeta SD ingresada y escribir el sistema en la tarjeta. Se deberá estar conectado a Internet para que el emulador de imágenes de *Raspberry Pi* descargue el sistema operativo.

Figura 3.11 Emulador de imágenes de *Raspberry Pi*

Finalmente, una vez grabado el sistema operativo, si el proceso fue exitoso el emulador presentará el mensaje que se ilustra en la Figura 3.12 y se podrá retirar la tarjeta SD.

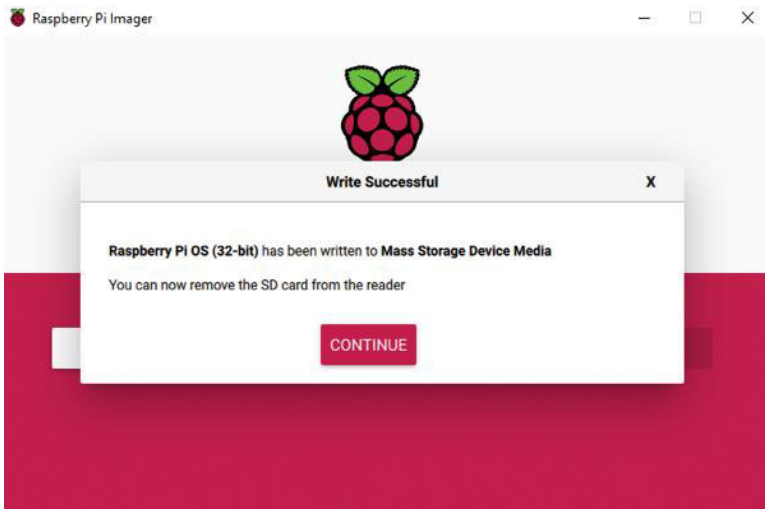


Figura 3.12 Sistema operativo grabado exitosamente en la tarjeta SD

Configuración para usar *VNC Viewer* y *SSH*

A continuación de la instalación del sistema se requirió controlar remotamente la *Raspberry Pi* para configurar sus parámetros iniciales. Por este motivo se eligió *SSH* como método de configuración a través de la terminal y *VNC* para configuración por la interfaz gráfica del sistema operativo *Raspberry Pi OS* mediante el ordenador. Inicialmente se ingresó la tarjeta SD a un ordenador y en su interior se creó un archivo vacío llamado ssh en bloc de notas como se ve en la Figura 3.13.

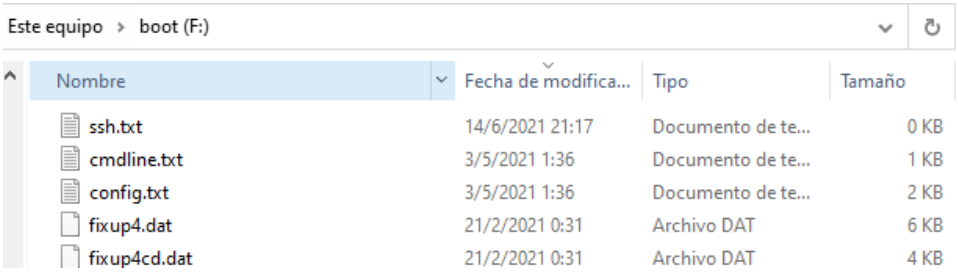


Figura 3.13 Archivo ssh.txt

Al archivo se le quitó la extensión .txt como se muestra en la Figura 3.14.

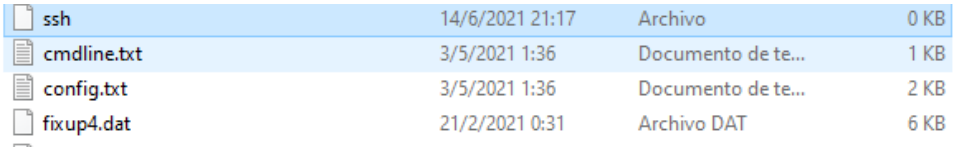


Figura 3.14 Archivo sin extensión .txt

Este archivo sirvió para que la *Raspberry Pi* pueda realizar en este, las configuraciones de *SSH*. Luego, fue necesario saber la dirección IP que el router asignó a la *Raspberry Pi,* para esto se utilizó el programa *Advanced IP Scanner*. Como se mira en la Figura 3.15 se lo puede descargar ingresando al enlace https://www.advanced-ip-scanner.com/es/.

Figura 3.15 Página de *Advanced IP Scanner*

Cuando finalizó la instalación, mediante la opción explorar se encontró que la *Raspberry Pi* estaba en la dirección IP 192.168.1.5, como se observa en la Figura 3.16.

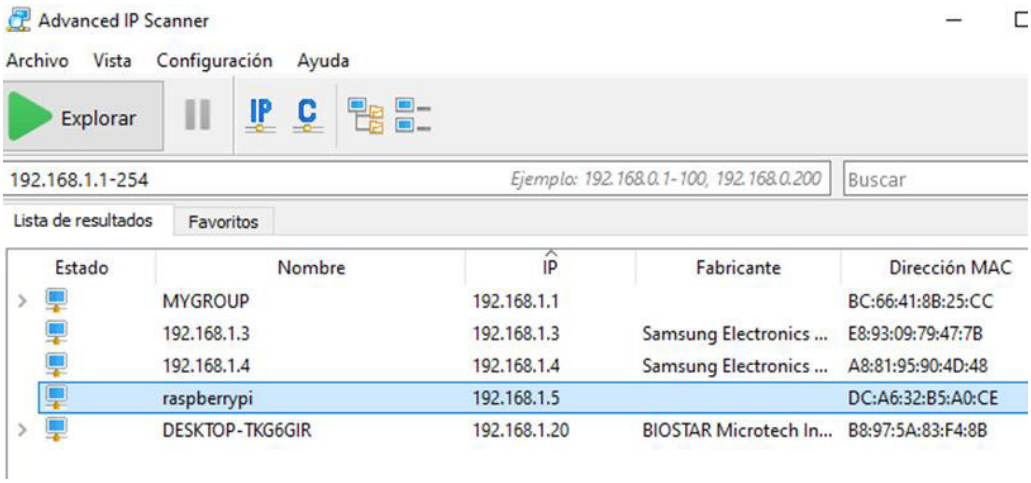


Figura 3.16 Búsqueda de dirección IP en *Advanced IP Scanner*

Más adelante, se abrió la consola de comandos (cmd) del ordenador y se digitó el comando *ssh pi@192.168.1.5*, como se ve en la Figura 3.17. Es significativo saber que por defecto la *Raspberry Pi* trae por nombre de usuario y contraseña ("pi", "*raspberry*") respectivamente, cuando recién se la adquiere.

```
C:\Users\Flia. Basantes>ssh pi@192.168.1.5
```

Figura 3.17 *SSH* a la dirección IP 192.168.1.5

Para establecer la conexión es necesario escribir que si (*yes*) e ingresar la contraseña anteriormente mencionada como se ilustra en la Figura 3.18.

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.5' (ECDSA) to the list of known hosts.
pi@192.168.1.5's password:
```

Figura 3.18 Establecimiento de conexión con la dirección IP 192.168.1.5

Una vez dentro, con el comando *sudo raspi-config* se accedió a la configuración inicial de la *Raspberry Pi* en donde se seleccionó la opción 3, Opciones de Interfaz. Justo a continuación se desplegó un submenú en el que se escogió *P3 VNC* y se habilitó *VNC*.

La configuración antes presentada se muestra en las Figura 3.19 a Figura 3.22.

```
pi@raspberrypi:~ $ sudo raspi-config
```

Figura 3.19 Comando *sudo raspi-config*

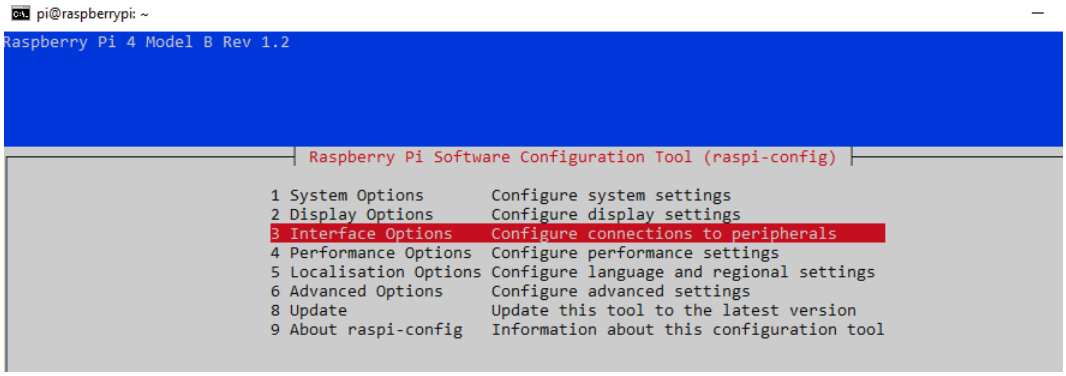


Figura 3.20 Selección de Opciones de Interfaz

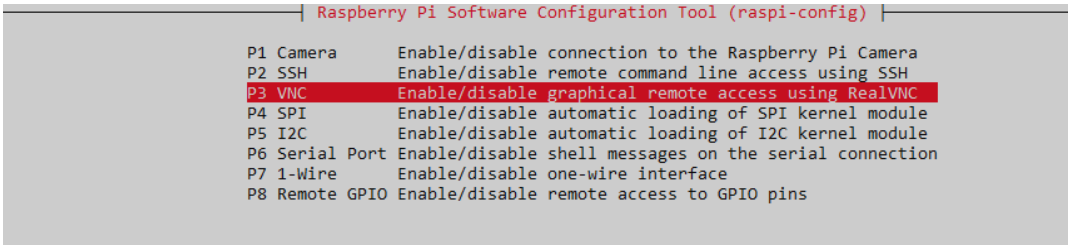


Figura 3.21 Selección de *VNC*

Figura 3.22 Habilitación del servidor *VNC*

Para que la configuración se guarde, reiniciar la *Raspberry Pi* con el comando *sudo reboot*. Después, fue necesario descargar la aplicación *VNC Viewer* en el ordenador, esta permite visualizar de forma gráfica el sistema operativo y controlarlo remotamente. La aplicación se encuentra disponible ingresando a la dirección web https://www.realvnc.com/es/connect/download/viewer/, como se observa en la Figura 3.23.

Figura 3.23 Descarga de *VNC Viewer*

Tan pronto como finalizó la instalación de *VNC Viewer* se abrió el programa, en este se ubicó la dirección IP de la *Raspberry Pi* tal cual se ve en la Figura 3.24.

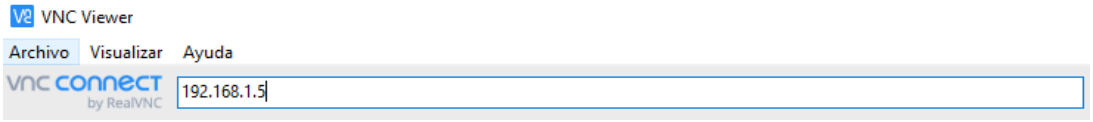


Figura 3.24 Establecimiento de la IP en *VNC* Viewer

Una vez que el servidor encontró la dirección IP se requirió ingresar el nombre de usuario y la contraseña. Si el proceso fue correcto se mostrará la interfaz gráfica en el ordenador como se destaca en la Figura 3.25.

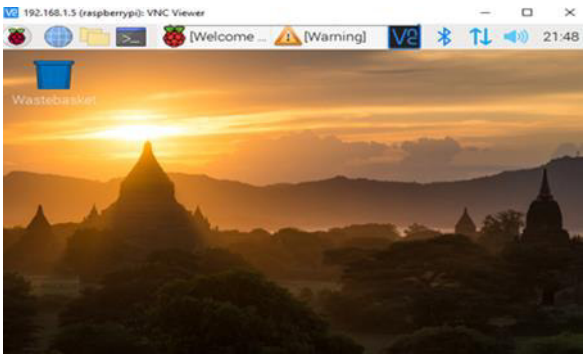


Figura 3.25 Conexión remota de la dirección IP 192.168.1.5 con *VNC Viewer*

Configuración básica de la *Raspberry Pi*

- *Modificación de contraseña*

Es indispensable cambiar la contraseña que viene por defecto para evitar que personas indeseadas ingresen a la *Raspberry Pi*. Para su modificación primero se entró como usuario *root* mediante el comando *sudo su*, segundo se escribió en la terminal el comando *passwd*, tercero se digitó la nueva contraseña y se confirmó que estuviera correcta.

Finalmente, para constatar si el cambio fue exitoso, se comprobó mediante el comando *su root* e ingresando la contraseña nueva. En la Figura 3.26 se encuentra el proceso descrito.

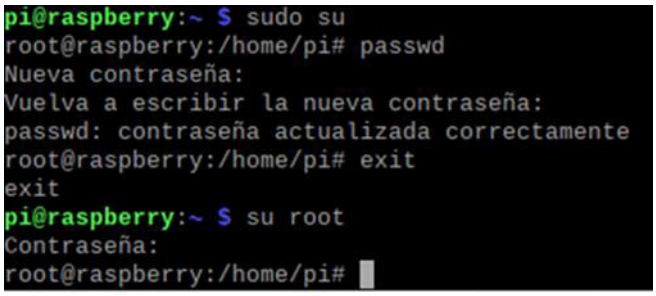


Figura 3.26 Cambio de contraseña del usuario pi

- *Selección de Zona horaria*

Se configuró la zona horaria para que la hora de la *Raspberry Pi* esté actualizada. Inicialmente se digitó el comando *sudo raspi-config* en la terminal, entonces en el menú que se desplegó se seleccionó la opción 5, Opciones de Localización. En la Figura 3.27 se destaca el menú. Dentro de esta opción se escogió L2 *Timezone*, más adelante se eligió en área geográfica América y en región Guayaquil. Por último, se reinició la *Raspberry Pi* para guardar las configuraciones con el comando *sudo reboot*. En las Figura 3.28 y Figura 3.29 se detalla el procedimiento descrito previamente.

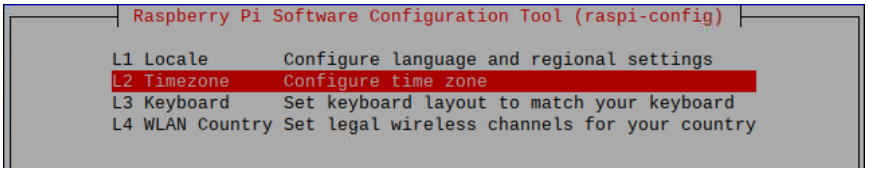


Figura 3.27 Opción L2 *Timezone*

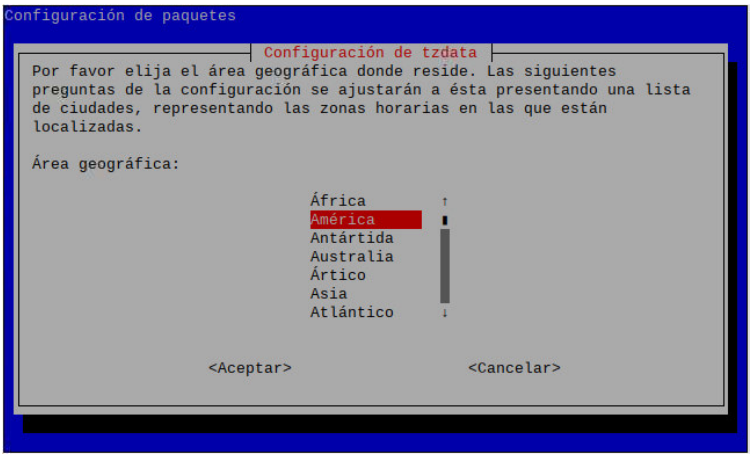


Figura 3.28 Selección del área geográfica

Figura 3.29 Selección de la región

- *Habilitación de Wireless LAN*

Con el objetivo de tener conexión inalámbrica se realizó la siguiente configuración. Para empezar, se ingresó a la terminal y se escribió el comando *sudo raspi-config*. Más adelante en el menú de configuraciones que se expone en la Figura 3.30 se eligió la opción 1, Opciones de Sistema. Luego dentro del submenú se escogió S1 *Wireless LAN* como se observa en la Figura 3.30, seguidamente fue fundamental ingresar el *SSID* (nombre de la red) y su contraseña. A continuación, se guardó los cambios reiniciando el sistema con *sudo reboot*. En definitiva, si la operación resultó exitosa se mostrará en la pantalla el símbolo de *Wi-Fi* y un visto en la red que está conectada, tal como se ve en la Figura 3.31.

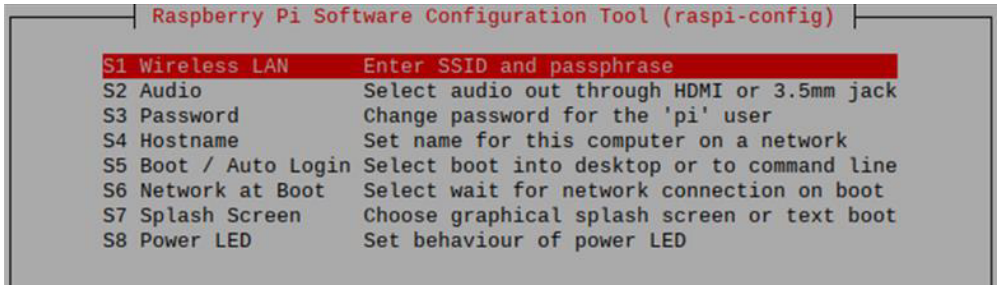


Figura 3.30 S1 *Wireless LAN*

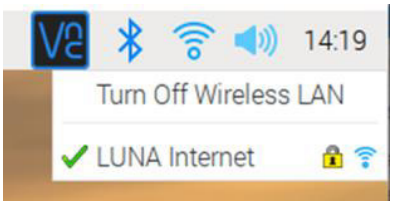


Figura 3.31 *Raspberry Pi* conectada a la red *Wi-Fi*

- *Cambio de Hostname y Auto Login*

Se cambió el nombre del *Host* para identificarlo fácilmente en la red local. Para esto se presionó sobre la opción S4 *Hostname*, en la Figura 3.30 se visualiza el menú. A continuación, se puede colocar el nombre que se prefiera como se ilustra en la Figura 3.32.

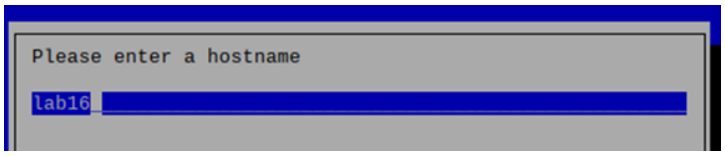


Figura 3.32 Cambio de nombre del *host*

Por otro lado, es conveniente seleccionar el tipo de arranque que se desea al iniciarse la *Raspberry Pi*, este puede ser de 2 tipos: mediante la interfaz de escritorio o con la línea de comandos. Además, se puede decidir si al iniciarse es necesario introducir el usuario y clave o a través de un *logueado* automático.

Para realizar la modificación del arranque, en el menú de la Figura 3.30 se pulsó sobre la opción S5 *Boot / Auto Login*, después se desplegó el submenú de la Figura 3.33. En este se puede escoger las alternativas mencionadas. Para el proyecto se seleccionó la posibilidad B4 para habilitar el *logueado* automático con interfaz de escritorio, debido a que se requiere que apenas encienda el dispositivo se despliegue la interfaz web. No obstante, al elegir otra opción como la B3 dificultaría el ingreso porque primero se tendría que ingresar la contraseña y el usuario tal cual se muestra en la Figura 3.34.

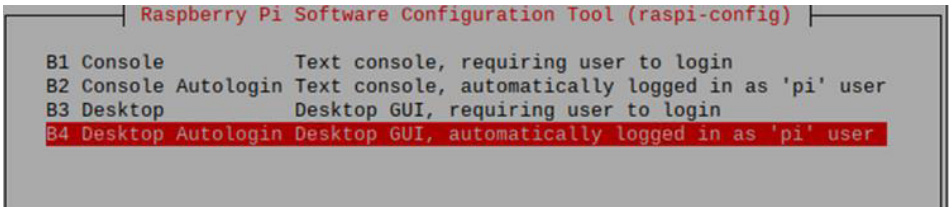


Figura 3.33 B4 *Logueado* automático con interfaz de escritorio

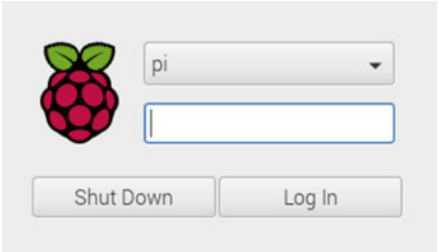


Figura 3.34 B3 *Logueado* con usuario y clave con interfaz de escritorio

Para que las configuraciones llevadas a cabo se conserven es crucial reiniciar el dispositivo con el comando *sudo reboot*.

- *Activación y desactivación de interfaces*

Existen varias interfaces de comunicación como se muestra en la Figura 3.21, entre ellas *I2C*, *SPI*, Serial, etc. Se pueden activar y desactivar eligiéndolas en el menú dependiendo la función que se vaya a realizar en la *Raspberry Pi*. Ya que para estas comunicaciones se han establecido pines específicos y solo se pueden usar como entradas o salidas si se desactivan.

En el caso de *I2C* usa los pines *GPIO* 2 y 3. Asimismo *SPI* emplea los pines *GPIO*: 10, 9, 11, 8 y 7. Por otra parte, *UART* o *Serial* se maneja en los pines *GPIO* 14 y 15.

El prototipo domótico requiere de algunos componentes electrónicos para su funcionamiento. La *Raspberry Pi* cuenta con entradas y salidas digitales suficientes para la conexión de la mayor parte del proyecto, pero carece de un conversor A/D lo que le impide obtener la lectura de sensores analógicos. Específicamente el prototipo usa una fotorresistencia para el control automático de las cortinas, se solucionó este problema a través del conversor A/D MCP3008 el cual usa comunicación *SPI*. Posteriormente, se hablará más acerca de sus características.

En base al análisis sobre las interfaces de comunicación y lo mencionado previamente, se llegó a la conclusión de que en el proyecto se habilitará la comunicación *SPI*; por consiguiente, se desactivará la *I2C* y Serial por lo que los pines se ocuparán normalmente como entradas y salidas digitales.

- *Actualización de la Raspberry Pi*

Es trascendental actualizar la *Raspberry Pi* antes de comenzar con cualquier configuración posterior a las iniciales. Con este fin se eligió la opción 8, *Update* en el menú de la Figura 3.20. Está opción actualiza todos los repositorios, por lo que es importante estar conectado a Internet. Al finalizar, saldrá un mensaje similar al de la Figura 3.35 que indica que el dispositivo se ha actualizado a la versión más reciente.

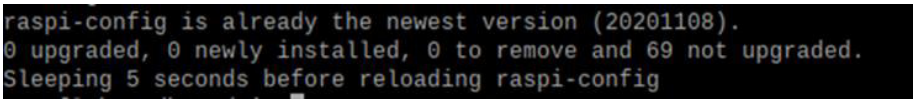


Figura 3.35 *Raspberry Pi* actualizada a la última versión

Configuración de direcciones IP fijas

Se colocó direcciones IP fijas a la conexión cableada e inalámbrica debido, a que al estar designadas por DHCP varían y se asignan aleatoriamente a otros dispositivos después de cierto tiempo, lo que dificulta en gran medida el despliegue de la interfaz web y puede llegar a ser molesto no saber en qué dirección IP se encuentra el prototipo domótico.

Antes que nada, en la terminal se tecleó el comando *ifconfig* para observar toda la configuración de red. En la Figura 3.36 se muestra que la *Raspberry Pi* cuenta con dos interfaces para acceder a Internet. La primera es eth0 para la conexión cableada por el puerto *Ethernet* y wlan0 para la conexión inalámbrica mediante el protocolo 802.11ac.

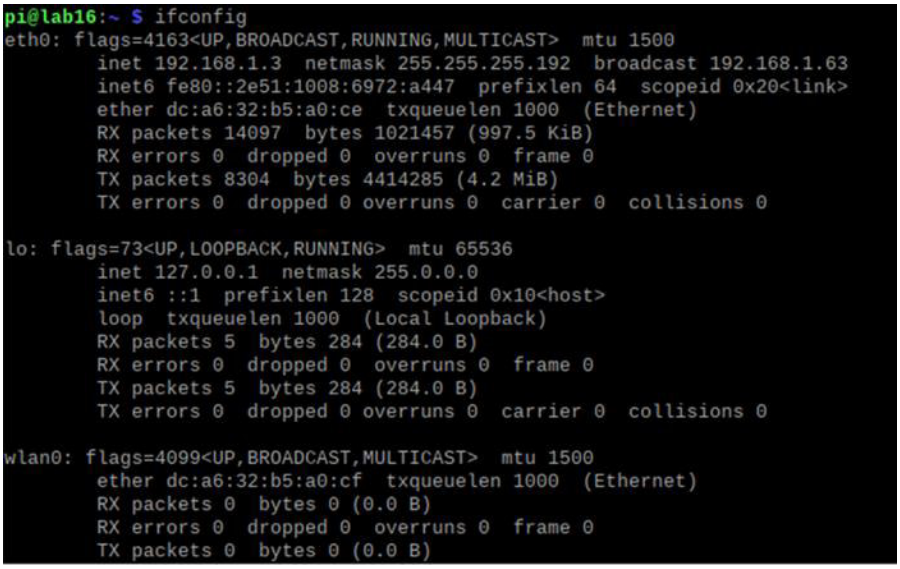


Figura 3.36 Comando *ifconfig*

A continuación, se necesitó averiguar la puerta de enlace de la red (*Gateway*) con el comando *route -n*. En la Figura 3.37 se indica que el *Gateway* es la dirección IP 192.168.1.1.

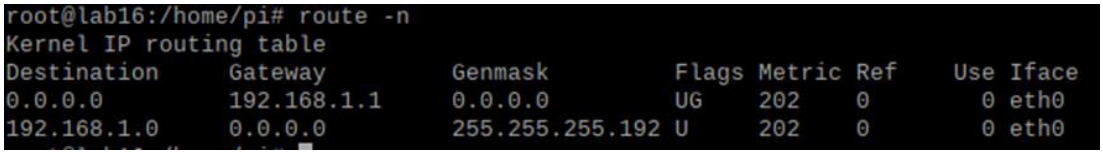


Figura 3.37 Comando *route -n*

Después, se ingresó en el archivo *dhcpcd.conf* por medio del comando *sudo nano /etc/dhcpcd.conf*.

Seguidamente en el archivo se ubicó en la interfaz eth0 la dirección IP estática 192.168.1.21, en la interfaz wlan0 la dirección IP estática 192.168.1.19 y en el *Gateway* para las dos, la dirección IP 192.168.1.1. En las Figura 3.38 y Figura 3.39 se nota más claramente lo expuesto.

```
pi@lab16:~ $ sudo nano /etc/dhcpcd.conf
```

Figura 3.38 Comando *sudo nano /etc/dhcpcd.conf*

```
 GNU nano 3.2                    /etc/dhcpcd.conf

interface eth0
static ip_address=192.168.1.21
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
static domain_search=

interface wlan0
static ip_address=192.168.1.19
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
static domain_search=
```

Figura 3.39 Modificación del archivo *dhcpcd.conf*

Por último, se debe reiniciar el dispositivo para que los cambios hechos en la interfaz de red se guarden. Si este proceso fue acertado se podrá localizar las direcciones IP de red a través de *Advanced IP Scanner* o volviendo a digitar el comando i*fconfig* en la terminal tal como se observa en las Figura 3.40 y Figura 3.41.

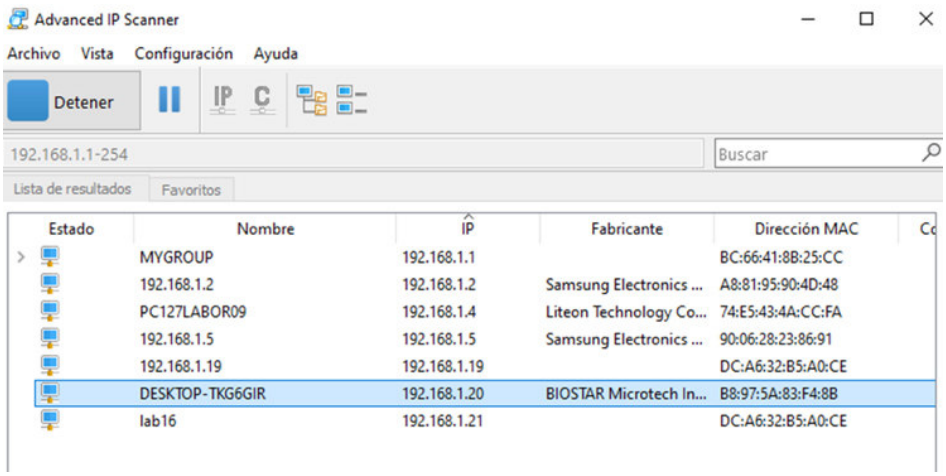


Figura 3.40 Localización de las direcciones IP modificadas

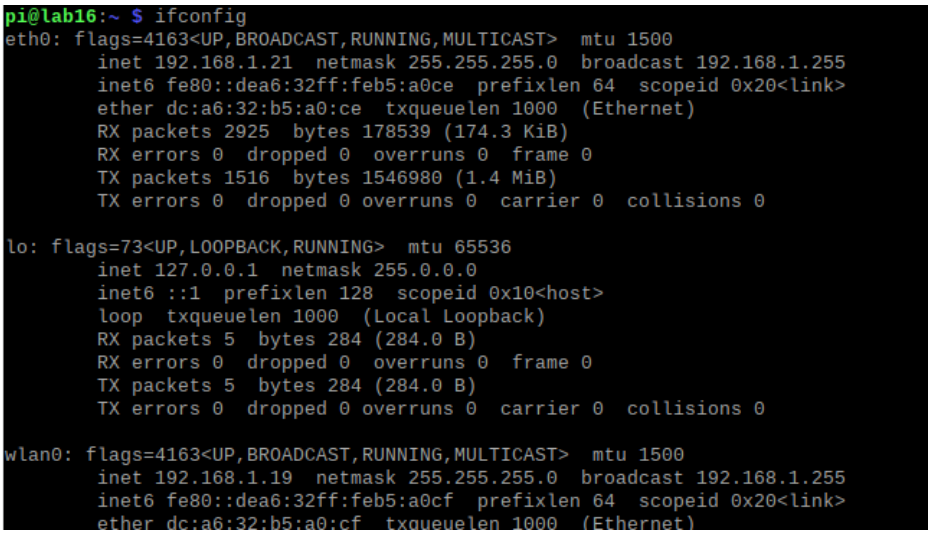


Figura 3.41 Interfaces eth0 192.168.1.21 y wlan0 192.168.1.19

Instalación y configuración del cortafuegos *UFW*

La *Raspberry Pi* está expuesta a tener posibles ataques e intrusiones que llegan desde Internet o en algunos casos desde ordenadores de la red *LAN* por lo que es fundamental contar con un cortafuegos activo [34]. De todas las posibilidades que ofrece el sistema operativo *Raspberry Pi*, se eligió *UFW* ya que es sencillo de configurar y muy práctico de usar, lo que permitió una fácil integración con la API de *Telegram*.

Se instaló el cortafuegos por medio del comando *sudo apt install ufw* tal como se muestra en la Figura 3.42.

Figura 3.42 Comando *sudo apt install ufw*

A continuación, se detallan las configuraciones del cortafuegos:

- *Puertos y servicios*

Al abrir un puerto se debe indicar el protocolo correspondiente TCP o UDP. Con el objetivo de facilitar esta actividad, el cortafuegos tiene predefinidos los servicios más usuales. Para saber la lista de todos estos servicios, se debe escribir el comando *sudo ufw app list*, justo como se observa en la Figura 3.43.

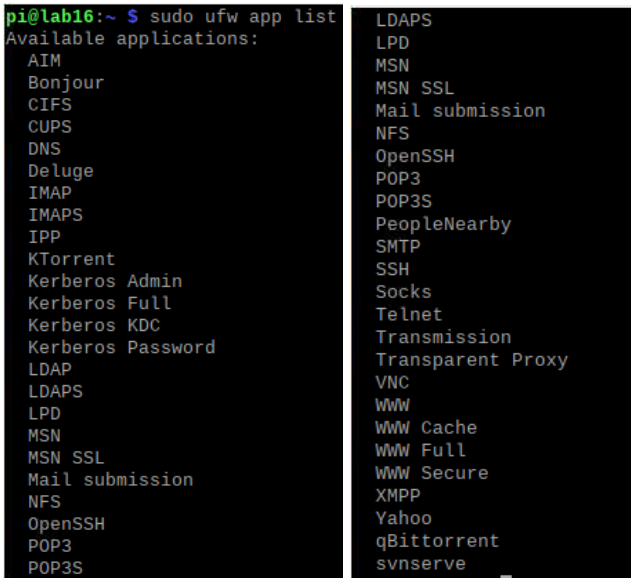


Figura 3.43 Comando *sudo ufw app list*

De manera que se puede permitir la entrada a un servicio por el número de su puerto o por su nombre. En la Tabla 3.2, a modo de ejemplo, se muestra la habilitación de dos servicios, siguiendo la misma lógica se puede habilitar cualquier otro.

Tabla 3.2 Habilitación de servicios [34]

Servicio	Número del Puerto	Nombre del servicio
Acceso remoto a un servidor	*sudo ufw allow 22/tcp*	*sudo ufw allow SSH*
Servidor web	*sudo ufw allow 80/tcp*	*sudo ufw allow WWW*

- *Servicios LAN*

Para los servicios que se van a llevar a cabo dentro de la red *LAN*, se puede habilitar todas las conexiones que vienen de los ordenadores o las provenientes de un puerto en específico, así como se muestra en la Tabla 3.3.

Tabla 3.3 Habilitación de servicios *LAN* [34]

Servicio *LAN*	Servicio *LAN* para puerto específico
sudo ufw allow from 192.168.1.0/24	*sudo ufw allow from 192.168.1.0/24 to any port 22*

- *Denegar entradas*

Al igual que se crean reglas para habilitar un servicio también, se pueden denegar si es necesario. Por ejemplo, en la Tabla 3.4 y en la Figura 3.44 se nota como se deniegan algunos servicios.

Tabla 3.4 Denegación de servicios [34]

Servicio	Número del Puerto	Nombre del servicio
Servidor web seguro (*HTTPS*)	*sudo ufw deny 443/tcp*	*sudo ufw deny WWW Secure*
Servidor web (*HTTP*)	*sudo ufw deny 80/tcp*	*sudo ufw deny WWW*

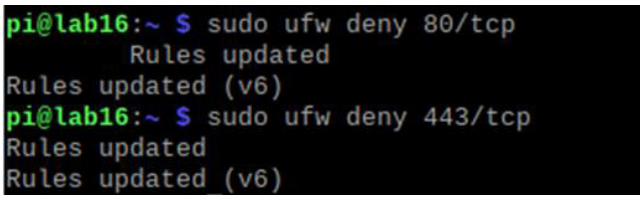


Figura 3.44 Denegación de servicios en la terminal

Asimismo, es posible denegar el acceso a un determinado host indicando su dirección IP, tanto si este proviene de la red *LAN* o *WAN*, como se muestra en las Tabla 3.5 y Tabla 3.6. Sin embargo, si ya se tienen reglas anteriormente colocadas en ese puerto será necesario que la nueva regla se lleve a cabo antes, para darle prioridad a la regla se puede utilizar el comando *insert 1*, como se observa en la columna 2 de la Tabla 3.5.

Tabla 3.5 Denegación de servicios *LAN* [34]

Servicio *LAN*	Servicio *LAN* para puerto específico
sudo ufw deny from 192.168.1.35 to any	*sudo ufw insert 1 deny from 192.168.1.20 to any port 80*

Tabla 3.6 Denegación de servicios *WAN* [34]

Servicio *WAN*	Servicio *WAN* para puerto específico
sudo ufw deny from 202.54.5.7 to any	*sudo ufw deny from 202.54.1.5 to any port 80*

- *Borrar reglas*

A la hora de borrar una regla ya establecida, lo más cómodo es emplear el comando *sudo ufw status numbered*, ya que ofrece las reglas numeradas justo como se visualiza en la Figura 3.45. De esta manera, solo se tiene que identificar su número en el listado. Por ejemplo, digitando *sudo ufw delete 2* se borraría la regla 2 que permite a la dirección IP de la red *LAN* 192.168.1.3 acceder al servidor web.

```
pi@lab16:~ $ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 80                         DENY IN     192.168.1.20
[ 2] 80                         ALLOW IN    192.168.1.3
[ 3] SSH                        ALLOW IN    Anywhere
[ 4] VNC                        ALLOW IN    Anywhere
[ 5] WWW                        DENY IN     Anywhere
[ 6] SSH (v6)                   ALLOW IN    Anywhere (v6)
[ 7] VNC (v6)                   ALLOW IN    Anywhere (v6)
[ 8] WWW (v6)                   DENY IN     Anywhere (v6)
```

Figura 3.45 Comando *sudo ufw status numbered*

- *Interfaces de red*

Las interfaces eth0 y wlan0 se pueden configurar en el cortafuego por separado, justo como se observa en la Tabla 3.7.

Tabla 3.7 Habilitación y denegación de servicios en las interfaces [34]

Interfaz eth0	Interfaz wlan0
sudo ufw allow in on eth0 to any port 80/tcp	*sudo ufw allow in on wlan0 to any port 22/tcp*
sudo ufw deny in on eth0 to any port 80/tcp	*sudo ufw deny in on wlan0 to any port 22/tcp*

- *Comandos importantes*

Una vez instalado el cortafuegos y definidas las reglas, es necesario activarlo con el comando *sudo ufw enable.* Por otro lado, para desactivarlo se emplea el comando *sudo ufw disable*. Finalmente, si se desea conocer el estado de las reglas, utilizar el comando *sudo ufw status verbose.* De la Figura 3.46 a la Figura 3.48 se muestra lo descrito anteriormente.

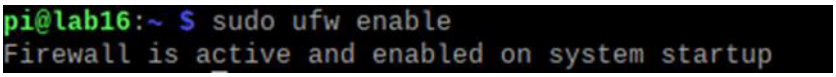


Figura 3.46 Comando *sudo ufw enable*

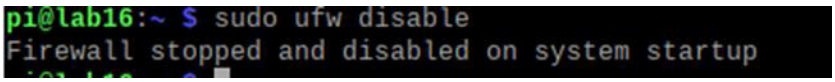


Figura 3.47 Comando *sudo ufw disable*

Figura 3.48 Comando *sudo ufw status verbose*

El prototipo domótico se diseñó para ser implementado en la red *LAN* por lo que el cortafuegos sirvió para dar acceso a los servicios como *SSH*, *VNC*, entre otros. Fundamentalmente se utilizó para activar y desactivar la interfaz web en dispositivos conectados a la red local como celulares, tabletas, *laptops* y computadoras

Instalación y configuración de *WinSCP*

Para transferir distintos archivos a la *Raspberry Pi* fácilmente desde un host de la red *LAN* se usó *WinSCP*. En primer lugar, se ingresó a la página oficial de *WinSCP* (https://winscp.net/eng/download.php) como se observa en la Figura 3.49. Posteriormente, se descargó la versión compatible para *Windows* y se instaló en el ordenador.

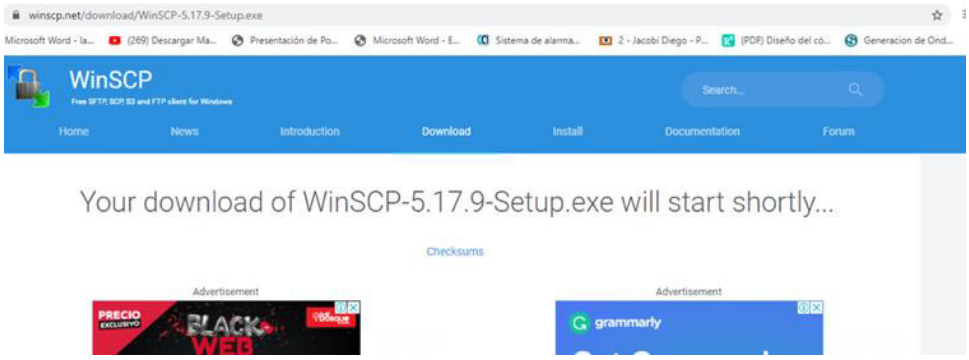


Figura 3.49 Descarga de *WinSCP*

Más adelante, se ingresó en el programa: el usuario, la contraseña, el número de puerto y la dirección IP. De manera que se logró transferir a modo de prueba algunos archivos exitosamente, justo como se muestra en las Figura 3.50 y Figura 3.51.

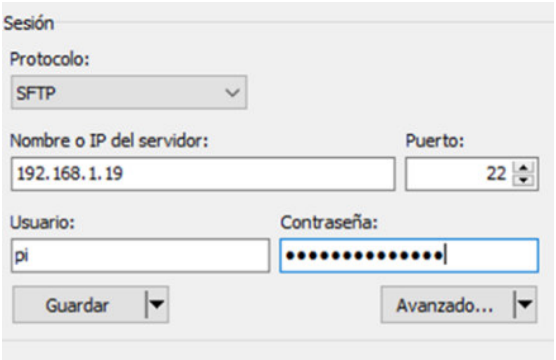


Figura 3.50 Conexión de *WinSCP* con la *Raspberry Pi*

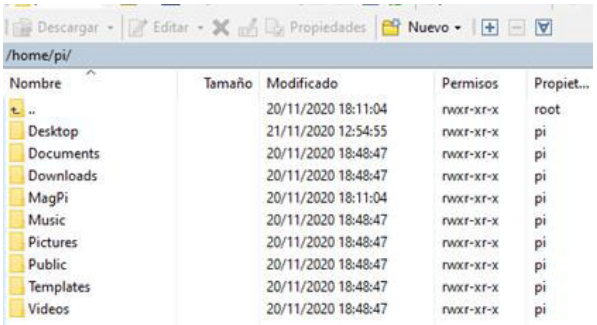


Figura 3.51 Transferencia de archivos a la *Raspberry Pi*

No obstante, si se ingresa como usuario pi solo se podrán compartir archivos con los directorios que este tenga permiso, como son: *Documents*, *Desktop*, *Music*, etc. Si se ingresa a otro directorio el cual no se tenga permiso, saldrá el mensaje de error de la Figura 3.52. Lo más acertado, es ingresar como usuario *root* para acceder a todos los directorios para este fin se efectuó las siguientes configuraciones en la *Raspberry Pi*.

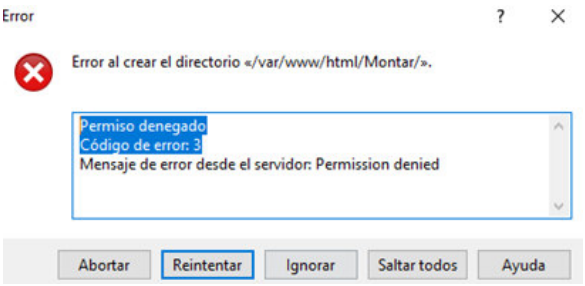


Figura 3.52 Error al transferir archivos

Se colocó el comando *sudo nano /etc/ssh/sshd_config* en la terminal para ingresar a la carpeta de configuración del *sshd_config*, justo como se observa en la Figura 3.53.

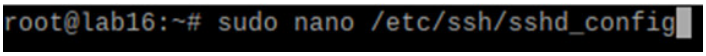


Figura 3.53 Comando *sudo nano /etc/ssh/sshd_config*

Seguidamente, se editó la línea *PermitRootLogin* tal cual se mira en la Figura 3.54.

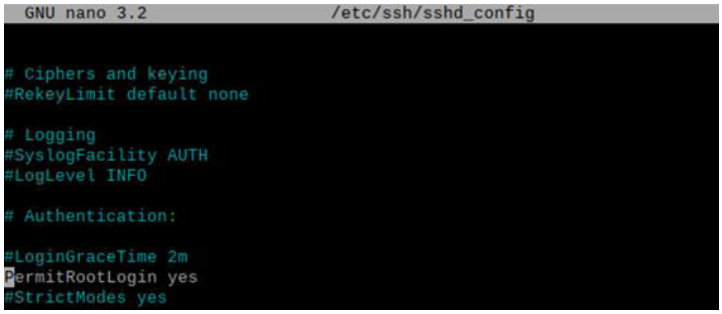


Figura 3.54 Habilitación para que el usuario *root* se pueda *loguear* remotamente

Después, se guardaron los cambios en el archivo y se agregó una contraseña al usuario *root* con el comando *sudo passwd root,* tal y como se observa en la Figura 3.55.

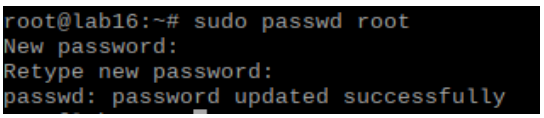


Figura 3.55 Comando *sudo passwd root*

Por último, se reinició el dispositivo y se conectó como usuario *root* con *WinSCP*, como se visualiza en la Figura 3.56.

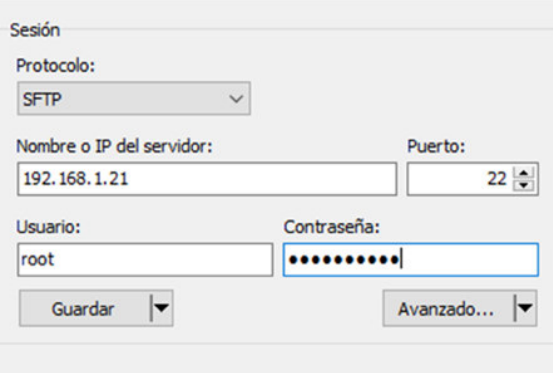


Figura 3.56 Conexión como usuario *root* con *WinSCP*

3.2 Diseño del sistema de control

Programación de los pines *GPIO*

Los pines *GPIO* son digitales, de manera que solo pueden tener dos estados, apagado o encendido. Además, permiten programarse como entradas o salidas por lo que son totalmente controlables a través de lenguajes de programación como: *Python*, *JavaScript*, *node-RED*, entre otros.

La tensión máxima a la que trabajan es de 3,3 (V_{DC}), con un consumo máximo de corriente de 16 (mA_{DC}). De manera, que se puede suministrar energía por un pin *GPIO* de forma segura a uno o dos leds con una resistencia. Lo recomendable es usar *leds* que sean de bajo consumo. En la Figura 3.57 se señala algunos *leds* clasificados por su tensión [35].

Figura 3.57 *Leds* según su consumo de tensión [13]

- *Programación desde la línea de comandos*

Raspberry Pi OS mapea la interfaz *GPIO* en */sys/class/gpio*. Este directorio contiene tres tipos de ficheros:

- Ficheros de Control de Interfaz

Estos se ocupan para pedir al núcleo de la *Raspberry Pi* el uso de cierto *GPIO* y también para liberar el pin una vez terminada su operación [35].

- Los propios *GPIOs*

Estos aparecen en forma de directorios.

- Controlador *GPIO*

Es otro directorio con ciertos ficheros que proveen información sobre el chip *GPIO* de la placa [35].

En base a lo analizado previamente se digitó los siguientes comandos para probar el encendido y apagado de un *led* desde la terminal. En la Figura 3.58 se aprecia como primero se exportó el pin *GPIO4*, luego se colocó la dirección como salida para finalmente darle un valor. Con el valor de 0 se enciende el *led* y con 1 se apaga.

Por otra parte, en la Figura 3.59 se observa físicamente implementado el circuito en un *protoboard*, este consta de una resistencia de 330 (Ω) y un *led* que está conectado al pin 7 (*GPIO4*).

```
File  Edit  Tabs  Help
pi@lab16:~ $ sudo su
root@lab16:/home/pi# echo "4" > /sys/class/gpio/export
root@lab16:/home/pi# echo "out" > /sys/class/gpio/gpio4/direction
root@lab16:/home/pi# echo "1" > /sys/class/gpio/gpio4/value
root@lab16:/home/pi# echo "1" > /sys/class/gpio/gpio4/value
root@lab16:/home/pi# echo "0" > /sys/class/gpio/gpio4/value
```

Figura 3.58 Encendido y apagado de un *led* desde la terminal

Figura 3.59 Implementación del encendido y apagado de un *led* desde la terminal

- *Programación con Python*

Se escogió *Python* para la programación de los *scripts* del prototipo, ya que es uno de los lenguajes de programación más versátiles que existen, por lo que puede ser usado fácilmente en muchos campos diferentes.

Para interactuar con los pines del puerto *GPIO* se utilizó la librería *RPi.GPIO*. Se la puede descargar del enlace https://pypi.org/project/RPi.GPIO/#files, tal cual se muestra en la Figura 3.60.

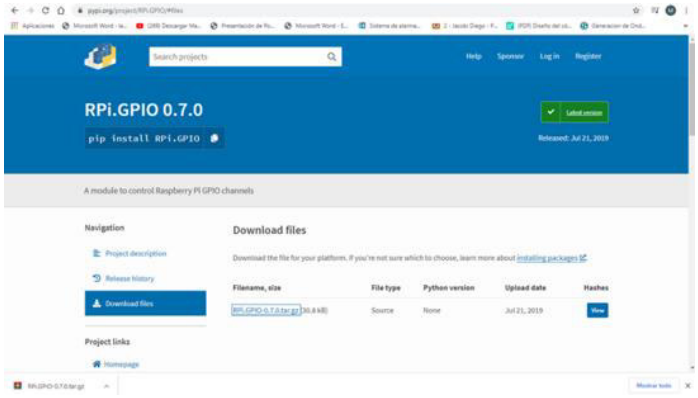


Figura 3.60 Descarga de la librería *RPi.GPIO*

Una vez que se descargó la librería, se la transfirió por medio de *WinSCP* al directorio descargas de la *Raspberry Pi*, tal como se muestra en la Figura 3.61.

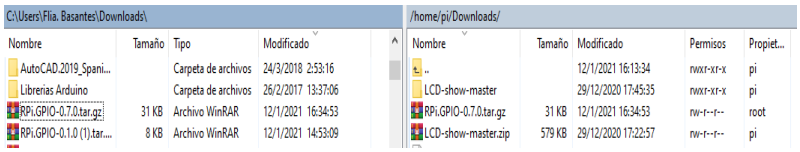


Figura 3.61 Transferencia de la librería a la *Raspberry Pi*

Posteriormente, en la terminal se ingresó a la carpeta descargas y se seleccionó el archivo de la librería para descomprimirlo con el comando *tar zxf*. Finalmente, dentro de todos los archivos de la librería se escogió *setup.py* y se instaló. Los pasos descritos previamente se los pueden examinar más a detalle en la Figura 3.62.

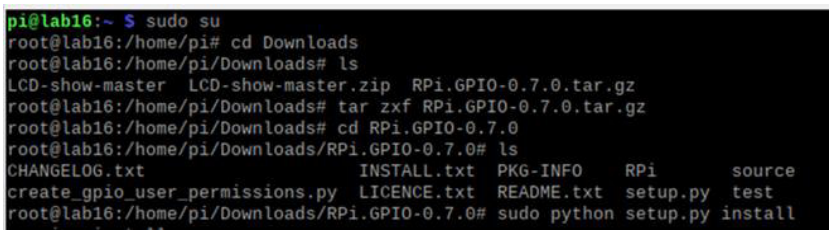


Figura 3.62 Instalación de la librería *RPi.GPIO*

Scripts para el control de las Luces

- *Esquema del código para encender las luces*

Para tener una visión más clara de la actividad de encender las luces se realizó un diagrama de flujo el cual se halla, en la Figura 3.63. Aunque, este es específicamente para el *GPIO4* que se encuentra en el pin 7, sirve de base para todas las demás áreas en donde se requería encender la luz.

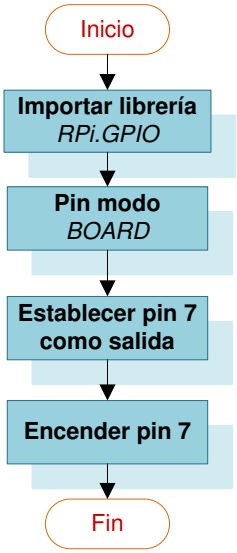


Figura 3.63 Diagrama de flujo para encender el *GPIO4*

En la Figura 3.64 por otra parte, se da a conocer el *script* creado en *Python*, mediante el comando *nano -c focon.py.* Seguidamente se creó otros con nombres distintos para las demás luces y en el archivo solo se cambió el número del pin.

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)
GPIO.output(7, True)
```

Figura 3.64 *Script* para encender el *GPIO4*

- *Esquema del código para apagar las luces*

Por el contrario, para la actividad de apagar las luces se diseñó el diagrama de flujo de la Figura 3.65, posteriormente se creó el *script* de la Figura 3.66 con el comando *nano -c focoff.py* el cual ayudó a construir los demás *scripts* en donde se deseaba apagar la luz.

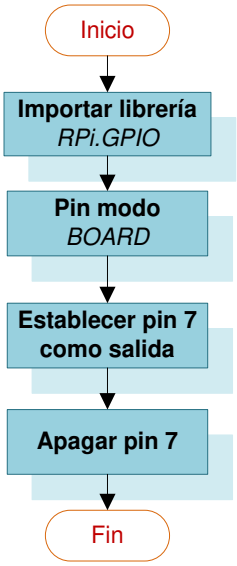


Figura 3.65 Diagrama de flujo para apagar el *GPIO4*

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)
GPIO.output(7, False)
```

Figura 3.66 *Script* para apagar el *GPIO4*

- *Esquema del código para automatizar la luz del pasillo*

En el diagrama de flujo de la Figura 3.67, se indica la forma en la que se programó el control automático de la luz del pasillo. En él, se puede observar que se emplearon varias librerías, principalmente la biblioteca *time* debido a que contiene una serie de funciones relacionadas con la medición del tiempo [36].

También, se da a conocer cómo se configuraron los pines y la iniciación de varias variables. El *script* se diseñó para funcionar al activarse desde la interfaz web, esto se logró a través de la lectura de la variable estado, si el valor leído es correcto se realiza todo el proceso del diagrama de flujo, caso contrario finaliza el programa. Adicionalmente, se requirió conocer el tiempo actual, para lo cual se empleó la función *time.localtime()* que devuelve el formato de la hora local.

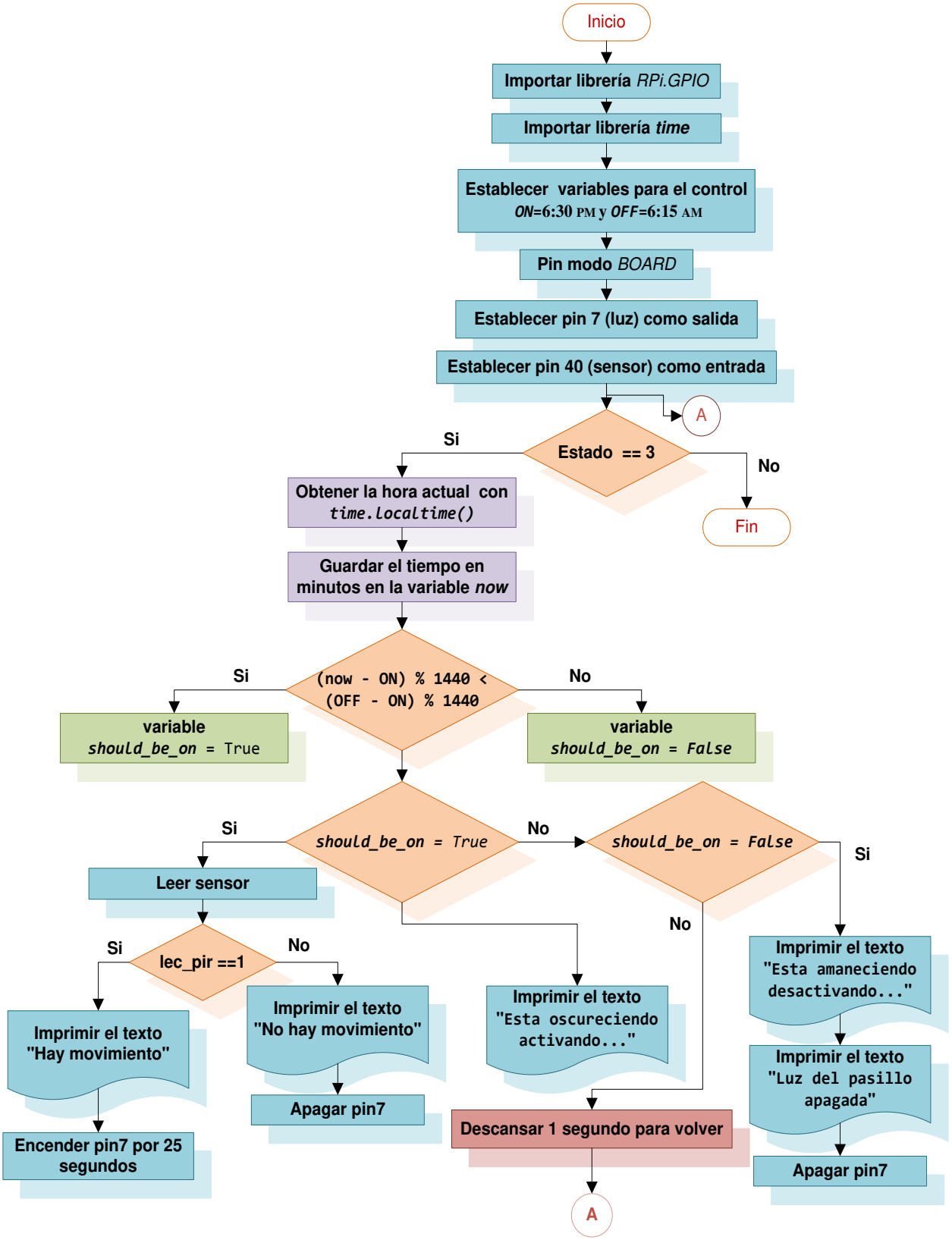


Figura 3.67 Diagrama de flujo para la luz automática del pasillo

Para la creación del *script* inicialmente, se generó el archivo llamado *pir.py* con el comando *nano -c pir.py.* A continuación, se establecieron las variables de control para habilitar (6:30 pm) y desactivar (6:15 am):

$$ON = (\mathrm{h} \cdot 60) + \mathrm{min}$$

Ecuación 3.1 Tiempo de habilitación en función de los minutos

Donde:

h : 18 (h) hora
min : 30 (min) minuto

Usando la Ecuación 3.1 se obtiene:

$$ON = 1110\ (\mathrm{min})$$

$$OFF = (\mathrm{h} \cdot 60) + \mathrm{min}$$

Ecuación 3.2 Tiempo de desactivación en función de los minutos

Donde:

h : 6 (h) hora
min : 15 (min) minuto

Empleando la Ecuación 3.2 se obtiene:

$$OFF = 375\ (\mathrm{min})$$

Por lo tanto, se podría establecer cualquier rango de hora mediante la Ecuación 3.1 y Ecuación 3.2. El problema que se debía solventar fue: encender la lectura del sensor por la noche para activar y desactivar la luz del pasillo, además de apagarlo por la mañana, por lo que se pensó en la lógica t > *on* y t < *off pero* no fue tan simple. La solución más adecuada a la que se llegó fue usar la función módulo (%) para ajustar las diferencias de tiempo con respecto al día siguiente o anterior, tal cual se muestra en la Figura 3.68.

```
stm = time.localtime()
now = stm.tm_hour * 60 + stm.tm_min #Tiempo actual
if (now - ON) % 1440 < (OFF - ON) % 1440:
        #1440/60=24 En 24 horas tenemos 1440 minutos
        should_be_on = True
else:
        should_be_on = False
```

Figura 3.68 Función para encender o apagar un dispositivo según hora predefinida

Mediante, *time.localtime ()* se devuelve una estructura, la cual se puede emplear para darle valores enteros de hora, minuto, día de la semana, día del mes, año, entre otros [37]. Sin embargo, cabe destacar también la manera en que se obtuvieron los datos del sensor, en la Figura 3.69 se visualiza esto. Así mismo, se ve como la luz del pasillo se enciende por 25 segundos por medio de la función *time.sleep()*. El método *time.sleep(secs)* detiene la ejecución del hilo de llamada durante el número de segundos indicados en el argumento *secs* [38].

```
lec_pir = GPIO.input(pir)#lectura del sensor
if lec_pir ==1 and estado_int==3:
        print("Hay movimiento")
        GPIO.output(foco1, False)
        time.sleep(25)
else:
        print("No hay movimiento")
        GPIO.output(foco1, True)
```

Figura 3.69 Obtención de los datos del sensor

***Scripts* para el control de la puerta**

- *Esquema del código para abrir la puerta*

El diagrama de flujo de la Figura 3.70 para la actividad de abrir la puerta es muy parecido al de encender la luz por no decir igual, sin embargo, se realizaron ligeras modificaciones ya que, si estaba todo el rato encendido el dispositivo de la puerta se calentaba después de un tiempo, por lo que se investigó y se llegó a la conclusión de que era mejor opción habilitar el pin 8 (*GPIO14*) por 5 segundos hasta que la persona salga y desactivarlo después de ese tiempo.

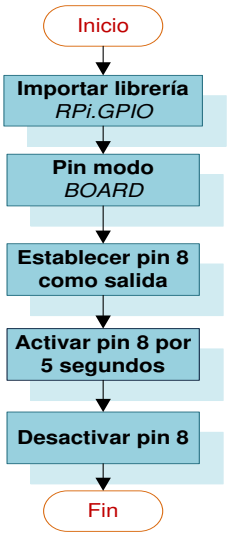


Figura 3.70 Diagrama de flujo para abrir la puerta conectada al *GPIO14*

En la Figura 3.71 por otra parte, se da a conocer el *script* creado en *Python*, mediante el comando *nano -c abrir.py.*

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT)
GPIO.output(8, False)
time.sleep(5)
GPIO.output(8, True)
```

Figura 3.71 *Script* para abrir la puerta conectada al *GPIO14*

- *Esquema del código para cerrar la puerta*

El diagrama de flujo de la Figura 3.72 y el *script* de la Figura 3.73 se diseñaron para cerrar la puerta desde la interfaz si la persona sale antes de los 5 segundos establecidos en el *script* anterior. Con el comando *nano -c cerrar.py* se creó el archivo de este *script*.

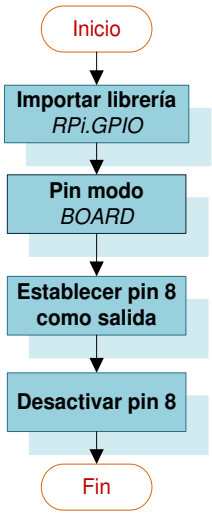


Figura 3.72 Diagrama de flujo para cerrar la puerta conectada al *GPIO14*

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT)
GPIO.output(8, True)
```

Figura 3.73 *Script* para abrir la puerta conectada al *GPIO14*

- *Esquema del código para saber el estado de la puerta y activar la alarma*

En el diagrama de flujo de la Figura 3.74, se da a conocer la forma en la que se programó el *script* para determinar el estado de la puerta y activar la alarma.

El *script* primero lee el estado que viene de la interfaz web, si este es correcto ejecuta el programa, caso contrario lo finaliza. Luego que se conoce el estado de la puerta mediante el sensor, se imprime su estado en un archivo de texto que posteriormente será representado en la interfaz. Finalmente, si la puerta está abierta se activa la alarma y se repite el ciclo si no se ha desactivado el *script* en la interfaz.

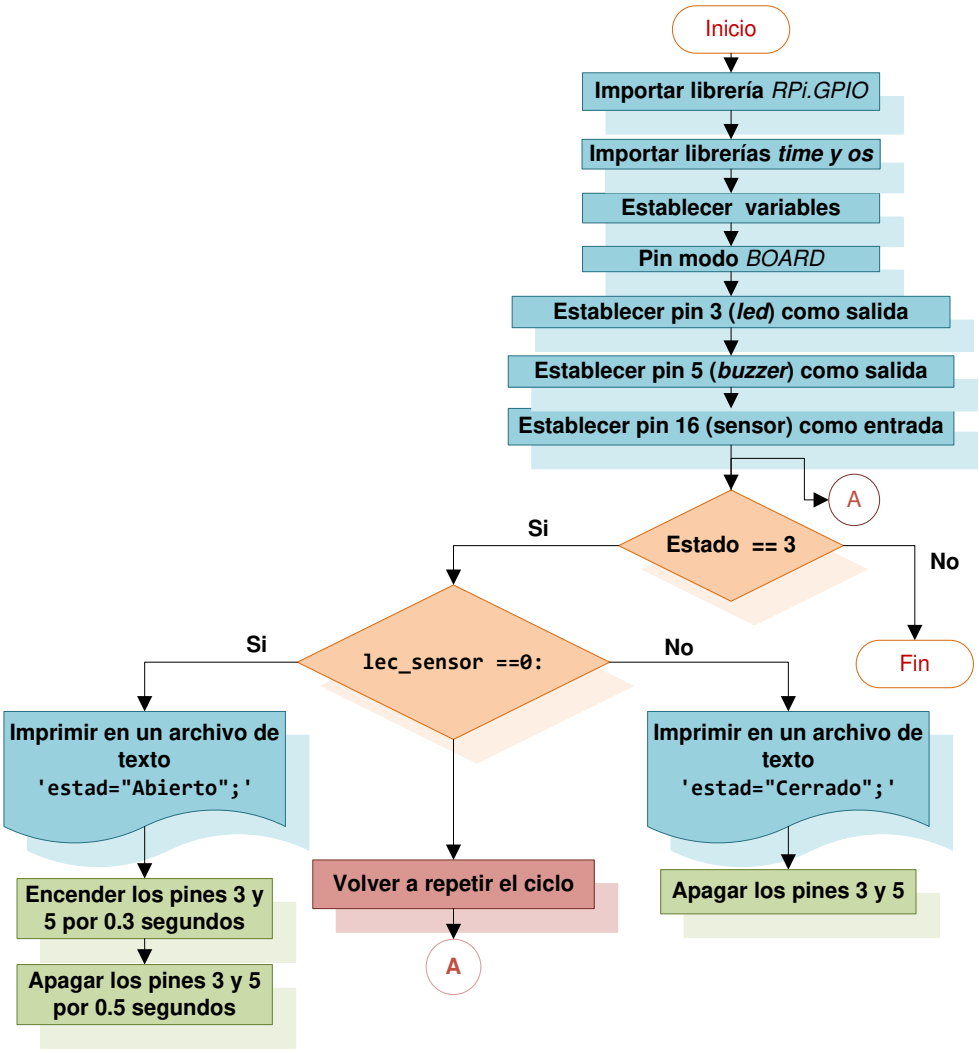


Figura 3.74 Diagrama de flujo del estado de la puerta y la alarma

En la Figura 3.75 se visualiza la parte más relevante de este *script*. En esta se nota que se empleó el módulo *os,* para esto fue necesario importar la librería al inicio del programa. Este módulo permite realizar operaciones en el Sistema Operativo como crear una carpeta, listar contenidos de una carpeta, conocer acerca de un proceso, finalizar un proceso, entre otros [39].

Específicamente se usó para llevar a cabo en el sistema la creación, modificación y eliminación del archivo de texto en el cual, se guarda el estado de la puerta.

Para esto también se utilizó junto con el módulo los comandos de *Linux* rm y mv que permiten eliminar y mover un archivo. El objetivo de esta parte es en primer lugar abrir un archivo de texto, después escribir en este el estado de la puerta, para luego moverlo al directorio de la interfaz y finalmente cada vez que el sensor cambie de lectura remover el archivo con uno nuevo de manera automática con el fin de mantenerlo actualizado. A este *script* se lo nombró como estadomag.py, y se creó en la terminal con el comando *nano -c estadomag.py.*

```
os.system ('sudo rm /var/www/html/Montar/2126/puerta/buscar/estado.txt')
archivo= ""
print("abierto")
archivo+='estad="Abierto";'
archivo2 = archivo.rstrip()
text_file = open('estado.txt','w')
text_file.write(archivo2)
text_file.close()
os.system("sudo mv estado.txt /var/www/html/Montar/2126/puerta/buscar/ ")
```

Figura 3.75 Escritura del estado de la puerta en un archivo de texto

- *Esquema del código para automatizar la puerta*

Se realizó el diagrama de la Figura 3.77, para tener un enfoque general del programa al momento de crearlo. Inicialmente el *script* lee el estado que viene de la interfaz web, si este es correcto ejecuta el programa caso contrario lo termina. Más adelante, se importa el módulo ultrasónico para posteriormente, con la ayuda del sensor de distancia poder obtener 5 muestras. Después se extrae el promedio de estas con el fin de evitar variaciones indeseadas tal cual se muestra en la Figura 3.76. El valor obtenido se imprime en un archivo de texto para luego ser representado en la interfaz web. Finalmente, si la persona está a una distancia inferior a los 30 (cm) la puerta se abrirá por 5 segundos y se cerrará después de este tiempo de forma automática.

Para efectuar el proceso anteriormente descrito se emplearon dos *scripts puertaauto.py* y *ultrasonico.py* respectivamente. Se diseñó de esta forma para probar el sensor fácilmente en los pines de la *Raspberry Pi.* De manera que en cualquier nuevo *script* diseñado solo se importe el módulo ultrasónico.

```
print("Comienzo")
suma = 0
for i in range(5):
        distancia = ultra.Distance()
        suma = distancia + suma
        time.sleep(0.3)
promedio = suma/5
```

Figura 3.76 Promedio de las lecturas del sensor de distancia

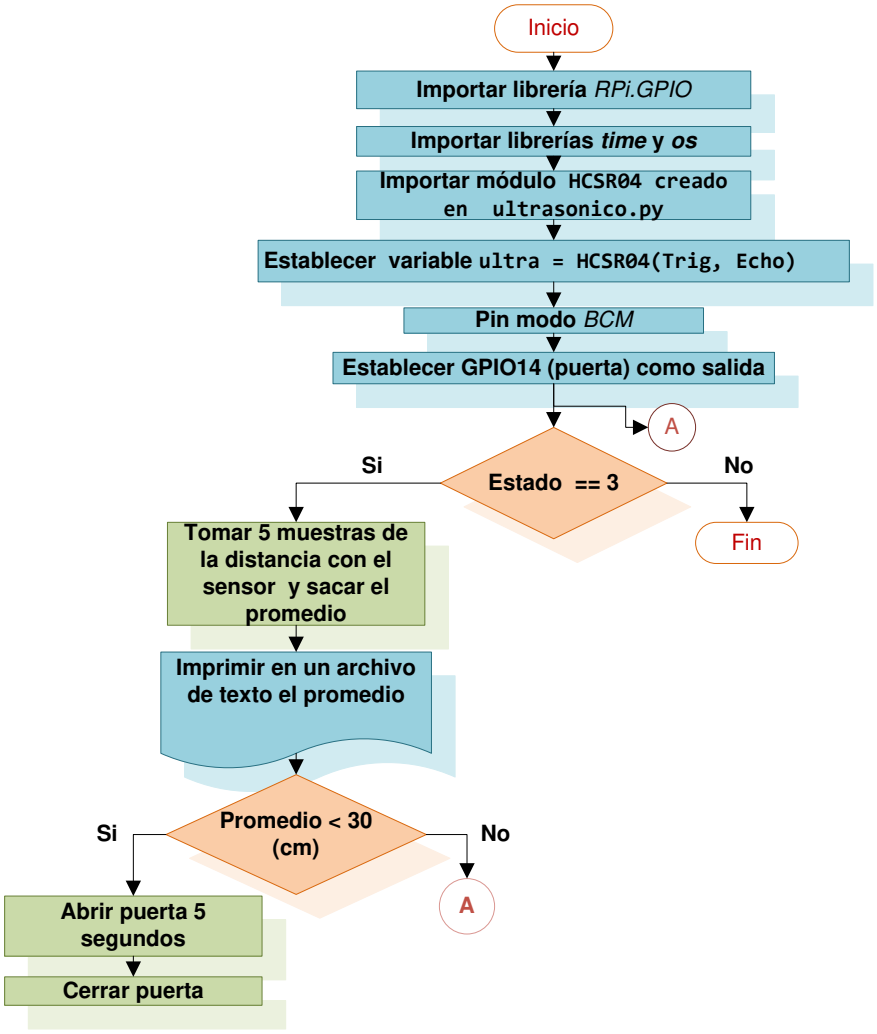


Figura 3.77 Diagrama de flujo para automatizar la puerta

En esta parte lo más significativo es el *script ultrasonico.py* y como se importó al *script puertaauto.py.* En principio en el *script ultrasonico.py* se utilizó la función clase, esta es una estructura de datos definida por el usuario, la cual contiene sus propios miembros de datos y funciones de miembros, a los que se puede acceder y utilizar realizando una instancia de esa clase [40].

La instancia de clase se define en el *script puertaauto.py con la variable ultra = HCSR04(16, 20)* después de importar el módulo con la línea de código *from ultrasonico import HCSR04.* El *GPIO16* va conectado al pin *TRIG* y el *GPIO20* al *ECHO* del sensor.

Además, se utilizó en la clase variables de instancia, estas son variables cuyo valor se asigna dentro de un constructor [40]. En la Figura 3.78 se destaca lo mencionado previamente.

```
class HCSR04:
        def __init__(self, Trig, Echo):
                self._T = Trig
                self._E = Echo
                GPIO.setup(self._T, GPIO.OUT, initial = 0)
                GPIO.setup(self._E, GPIO.IN)
        def _timing(self):
                GPIO.output(self._T, 1)
                time.sleep(0.00001)
                GPIO.output(self._T, 0)
                startTime=time.time()
                stopTime=time.time()
                while 0 ==GPIO.input(self._E):
                        startTime=time.time()
                while 1 ==GPIO.input(self._E):
                        stopTime=time.time()
                duration = stopTime - startTime
                return duration

        def Distance(self):
                duration = self._timing()
                cm = ((duration * 34300) / 2)
                return cm
```

Figura 3.78 Clase HCSR04

Para controlar el sensor y realizar las mediciones en la clase HCSR04 se efectuó lo siguiente: Inicialmente, se colocó el *TRIG* en alto por 10 (µs) que es la duración mínima del pulso de disparo del *TRIG* según las especificaciones del sensor HCSR04, inmediatamente después en bajo. Al momento de estar en bajo el sensor envía 8 pulsos ultrasónicos de 40 (kHz) y coloca *ECHO* en alto. Después fue indispensable detectar este cambio para comenzar a medir el tiempo en ese preciso instante. Una vez medido este tiempo, se ocupó la siguiente fórmula para obtener la distancia:

$$\mathrm{v} = \frac{\mathrm{d}}{\mathrm{t}}$$

Ecuación 3.3 Velocidad en función del tiempo [41]

Donde:

- v : 34300 (cm/s) velocidad del sonido
- t : Duración de cambio de ECHO (s)
- d : Distancia a la persona (cm)

Usando la Ecuación 3.3 se obtiene:

$$\mathrm{d} = \frac{34300\,.\,\mathrm{t}}{2}$$

El valor obtenido se divide para 2, dado que la medición corresponde al tiempo que se demora el pulso del sensor ultrasónico en llegar al obstáculo y regresar.

Scripts para el control de las cortinas

Para el control de las cortinas, se diseñaron dos procesos. En el primero las cortinas solo avanzaban y retrocedían. Sin embargo, si pasaban sobre el tamaño de las persianas enrollables se corría el riesgo de forzar el sistema de las persianas. Por lo cual, se llegó a la conclusión que la mejor forma era guardar el estado actual en el que se encontraban las cortinas y hacer que los motores recorran el número de pasos correspondientes a su tamaño. No obstante, se conservó los *scripts* ya que sirven para probar el control de las cortinas rápidamente desde el *Bot* de *Telegram*. Los *scripts* que se emplearon para esta función son: motor311.py, motor312.py, motor411.py y motor 412.py.

Ahora bien, no se va a detallar el funcionamiento de estos *scripts* ya que el segundo proceso abarca completamente al primero, por lo que el lector puede fácilmente entender su funcionamiento por medio del segundo proceso que se detalla a continuación:

- *Esquema del código para bajar la persiana 1 totalmente y a la mitad*

El diagrama de flujo de la Figura 3.79 se creó con la finalidad de tener una mejor visión de la actividad de bajar la persiana totalmente y a la mitad. Luego, se diseñó el *script* con el nombre motor111.py. En este, primeramente, se definieron las librerías, después se establecieron los pines *GPIO25* y *GPIO12* como salidas. Más adelante, se leyó el archivo de texto almacenar.txt para saber la posición de la cortina. La Figura 3.80 indica cómo se realizó la lectura, en principio se definió la ruta del archivo de texto. Seguidamente, se abrió el archivo como lectura con la siguiente línea de código *archivo = open (ruta, 'r')*, para posteriormente guardar el contenido del archivo de texto en la variable estado. Finalmente se cerró el archivo.

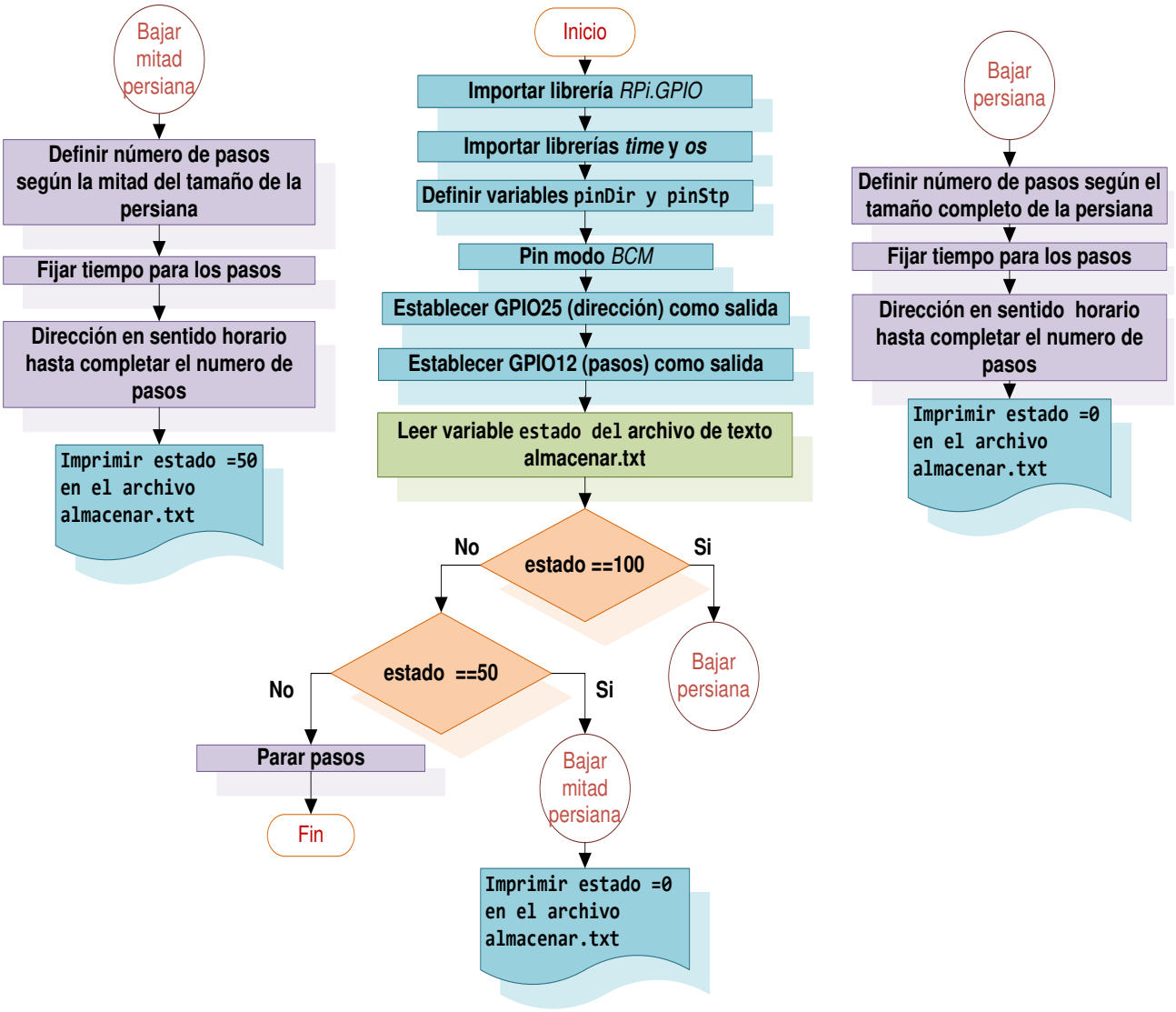


Figura 3.79 Diagrama de flujo para bajar la persiana totalmente y a la mitad

```
# lectura
ruta = '/var/www/html/cgi-enabled/almacenar.txt'
archivo = open(ruta,'r')
estado = archivo.readline()
estado_int = int(estado)
archivo.close()
print (estado_int)
```

Figura 3.80 Lectura del archivo almacenar.txt

Por otra parte, con la variable estado se efectuó un condicional, si el estado es igual a 100, se tendrá que bajar totalmente la persiana ya que la cortina está abierta. En cambio, sí es igual a 50 el programa bajará hasta la mitad de la persiana.

Si el estado es diferente a estos dos valores quiere decir que la cortina ya está en cualquiera de estas 2 posiciones, por lo que se mantendrá en el mismo lugar, de esta forma el sistema de persianas no se verá forzado.

En las Figura 3.81 y Figura 3.82 se detallan las funciones de bajar la cortina totalmente y la de bajar la persiana a la mitad. En ambos casos se especifica sobre todo el número de pasos, el tiempo de duración entre pasos y sobre todo la dirección con la línea de código *GPIO.output(pinDir, GPIO.LOW)* en sentido horario.

```
def bajaPersiana():
        numPas=6000 #(0) desde arriba(100)
        tiempo=0.005
        # Hacemos giro en sentido horario
        GPIO.output(pinDir, GPIO.LOW)
        for i in range(0,numPas):
                GPIO.output(pinStp, GPIO.HIGH)
                time.sleep(tiempo)
                GPIO.output(pinStp, GPIO.LOW)
                time.sleep(tiempo)
        estado =0
```

Figura 3.81 Función para bajar la cortina totalmente

```
def bajamitadPersiana():
        #funcion para bajar la persiana hasta la mitad(50)
        numPas=3000 #(50) desde arriba(100)
        tiempo=0.005
        # Hacemos giro en sentido horario
        GPIO.output(pinDir, GPIO.LOW)
        for i in range(0,numPas):
                GPIO.output(pinStp, GPIO.HIGH)
                time.sleep(tiempo)
                GPIO.output(pinStp, GPIO.LOW)
                time.sleep(tiempo)
        estado =50
```

Figura 3.82 Función para bajar la cortina hasta la mitad

Una vez, que se ejecuta cualquiera de las 2 funciones anteriores el nuevo estado se guarda en el archivo almacenar.txt. Esta parte, ya no se detalla porque anteriormente se explicó el cómo escribir sobre un archivo de texto.

- *Esquema del código para subir la persiana 1 totalmente y a la mitad*

Ahora bien, el diagrama de flujo de la Figura 3.83 permite conocer el proceso de subir las cortinas totalmente y a la mitad. Este proceso es similar al anterior, aunque varía en el condicional, si el estado es igual a 0, se tendrá que subir totalmente la persiana ya que la cortina está cerrada. En cambio, si es igual a 50 el programa subirá hasta la mitad de la persiana.

Si el estado es diferente a estos dos valores quiere decir que la cortina ya está en cualquiera de estas 2 posiciones, por lo que se mantendrá en el mismo lugar. Además, en las Figura 3.84 y Figura 3.85, en la línea de código *GPIO.output(pinDir, GPIO.HIGH)*, se nota que la dirección cambia a sentido antihorario.

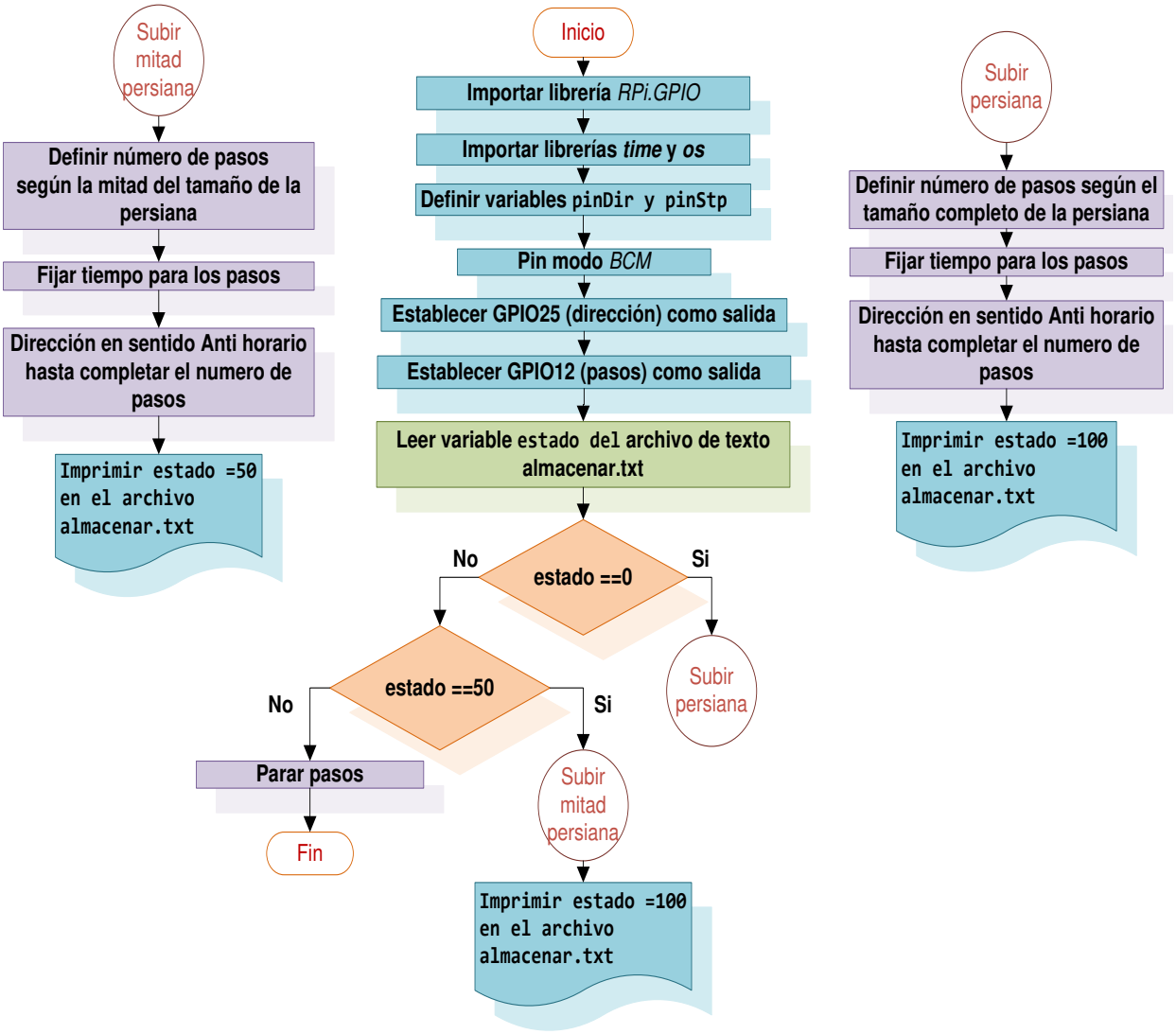


Figura 3.83 Diagrama de flujo para subir la persiana totalmente y a la mitad

```
def subePersiana():
        numPas=6000 #(100) desde abajo(0)
        tiempo=0.005
        # Hacemos giro en sentido antihorario
        GPIO.output(pinDir, GPIO.HIGH)
        for i in range(0,numPas):
                GPIO.output(pinStp, GPIO.HIGH)
                time.sleep(tiempo)
                GPIO.output(pinStp, GPIO.LOW)
                time.sleep(tiempo)
        estado =100
```

Figura 3.84 Función para subir la persiana totalmente

```
def subemitadPersiana():
        #funcion para subir la persiana hasta la mitad(50)
        numPas=3000 #(50) desde abajo(0)
        tiempo=0.005
        # Hacemos giro en sentido antihorario
        GPIO.output(pinDir, GPIO.HIGH)
        for i in range(0,numPas):
                GPIO.output(pinStp, GPIO.HIGH)
                time.sleep(tiempo)
                GPIO.output(pinStp, GPIO.LOW)
                time.sleep(tiempo)
        estado =50
```

Figura 3.85 Función para subir la cortina hasta la mitad

- *Esquema del código para subir a la mitad y bajar a la mitad la persiana 1*

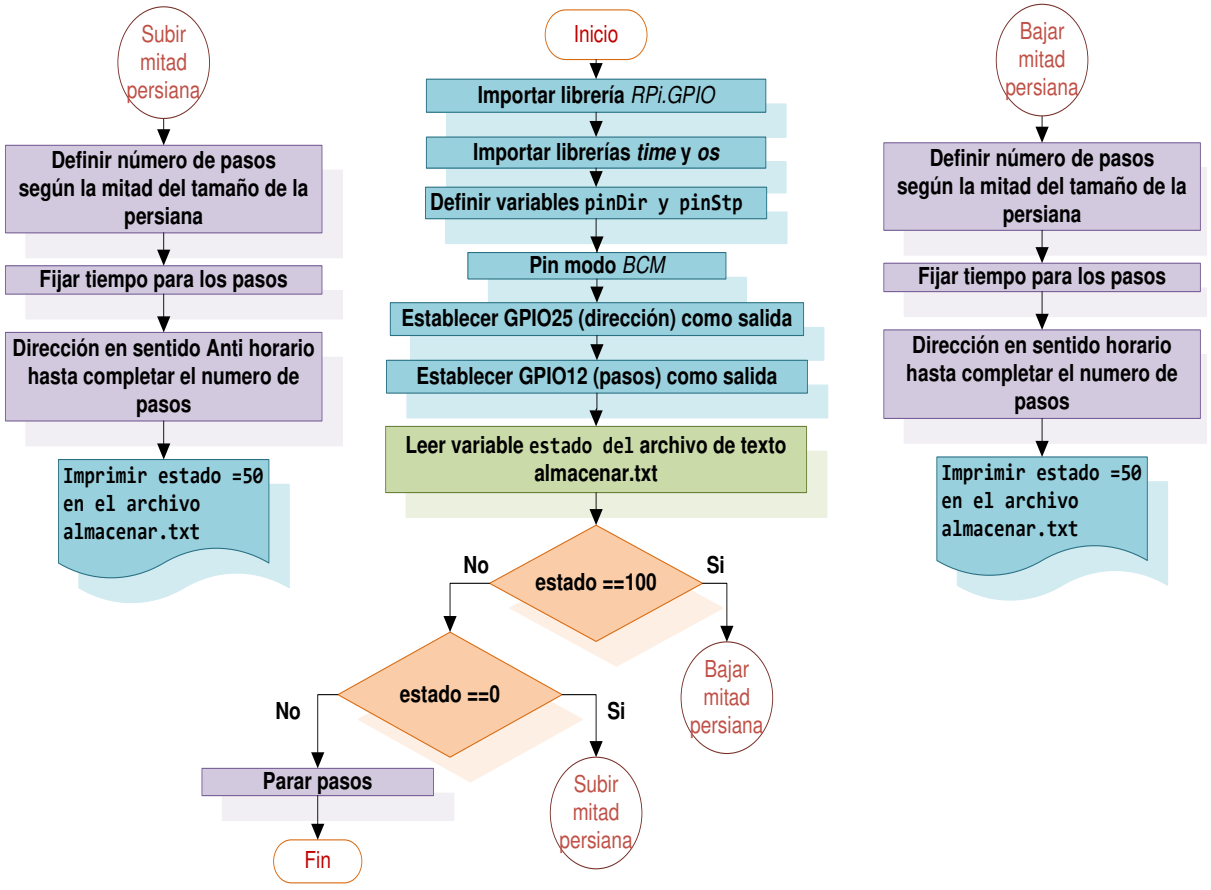


Figura 3.86 Diagrama de flujo para subir a la mitad y bajar a la mitad la persiana

El diagrama de flujo de la Figura 3.86 permitió elaborar el *script* motor113.py. El funcionamiento de este *script* es parecido a los programas anteriores.

Sin embargo, si el estado es igual a 100, se tendrá que bajar la persiana a la mitad. En cambio, sí es igual a 0 el programa subirá hasta la mitad de la persiana. Por otro lado, si el estado es diferente a estos dos valores quiere decir que la cortina ya está en cualquiera de estas 2 posiciones, por lo que se mantendrá en el mismo lugar.

Para el control de la persiana 2 se repitió el mismo proceso mediante los *scripts* motor211.py, motor212.py y motor213.py. Adicionalmente, para controlar las 2 cortinas al mismo tiempo, se pensó lanzar los *scripts* desde la interfaz juntos a través de una sentencia *if*, como las luces, pero al haber funciones de tiempo se desigualaban por lo que se diseñó los diagramas de flujo y *scripts* que se presentan más adelante.

- *Esquema del código para bajar las persianas totalmente y a la mitad*

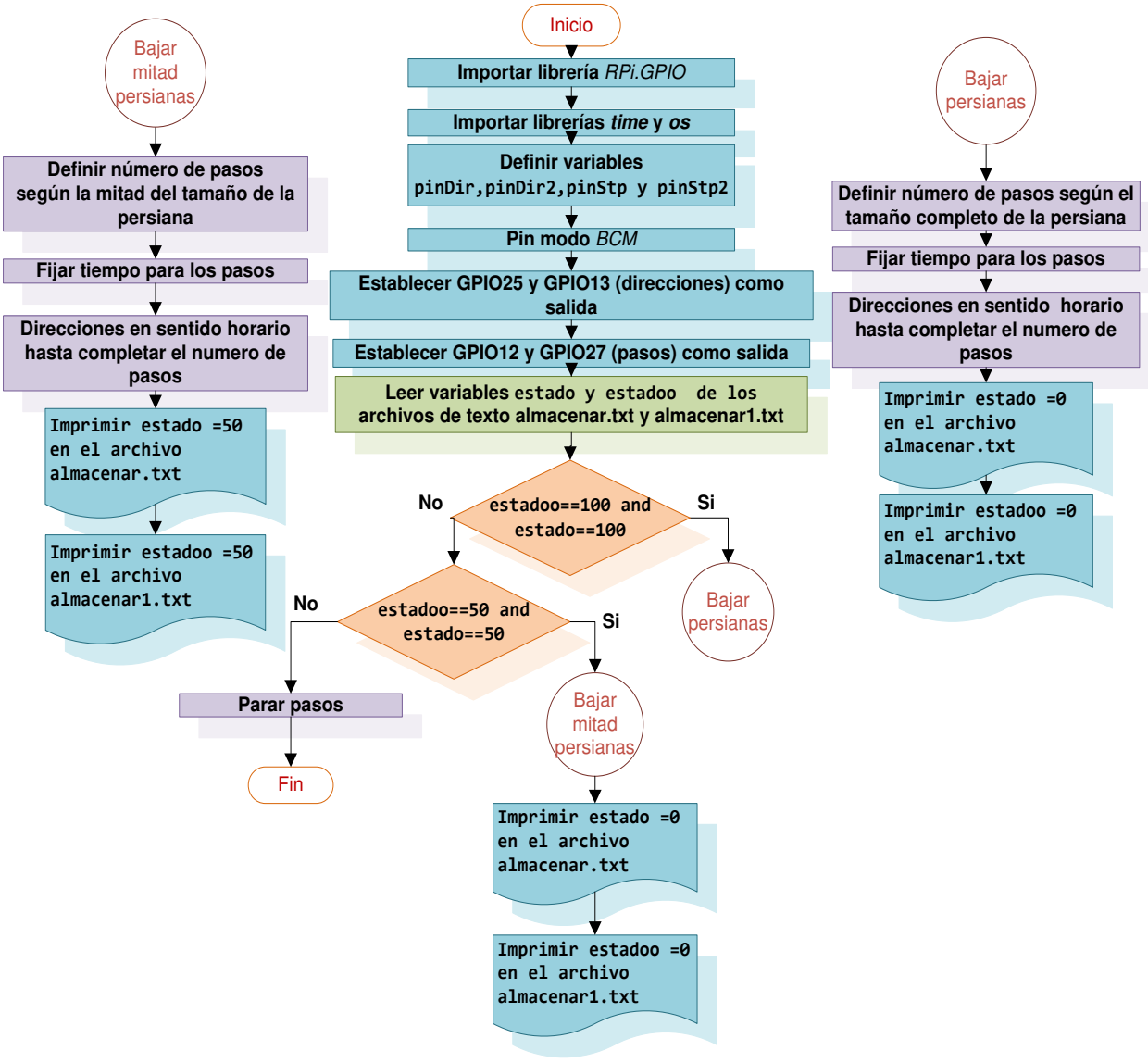


Figura 3.87 Diagrama de flujo para bajar las persianas totalmente y a la mitad

- *Esquema del código para subir las persianas totalmente y a la mitad*

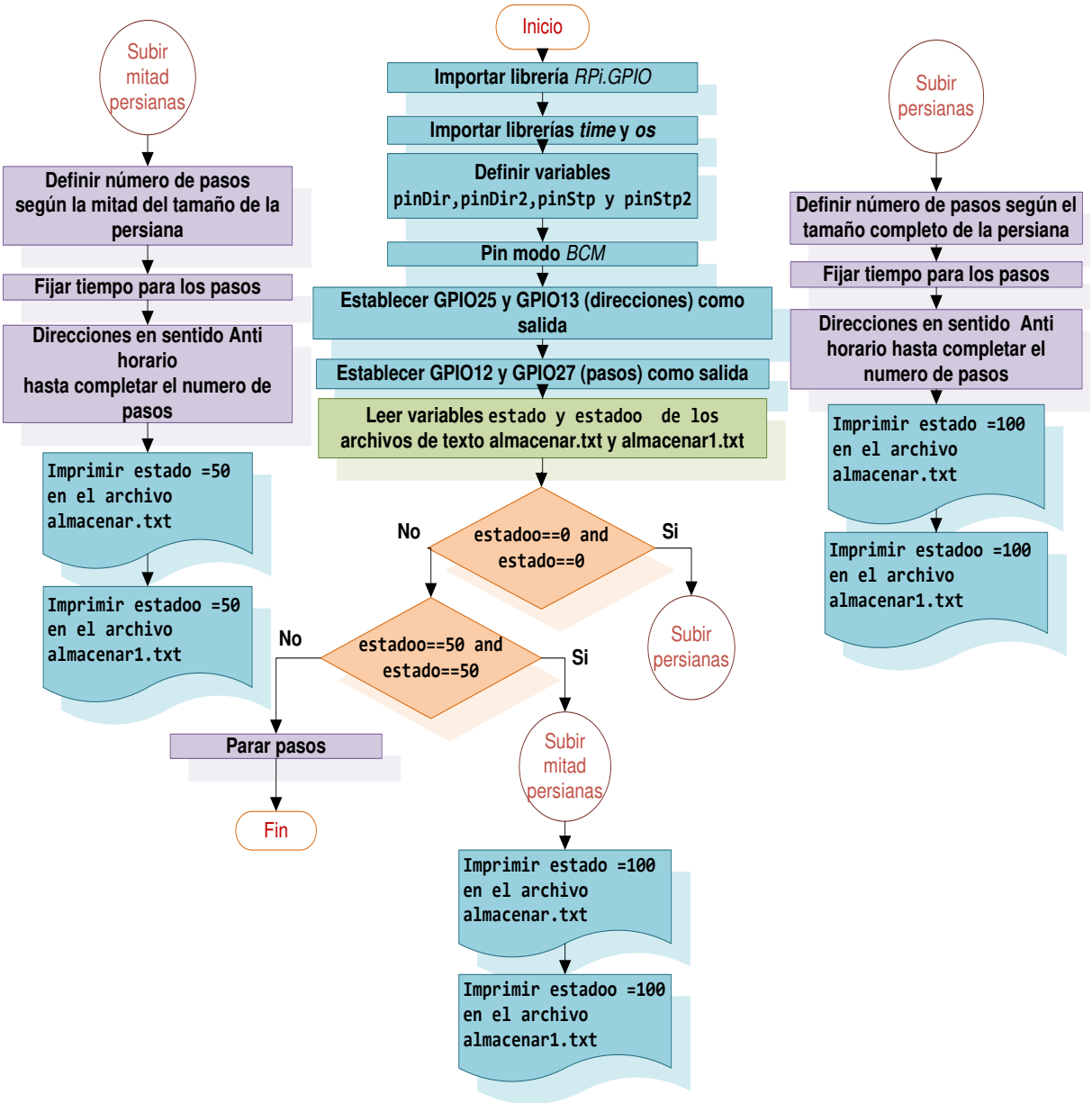


Figura 3.88 Diagrama de flujo para subir las persianas totalmente y a la mitad

- *Esquema del código para subir a la mitad y bajar a la mitad las persianas*

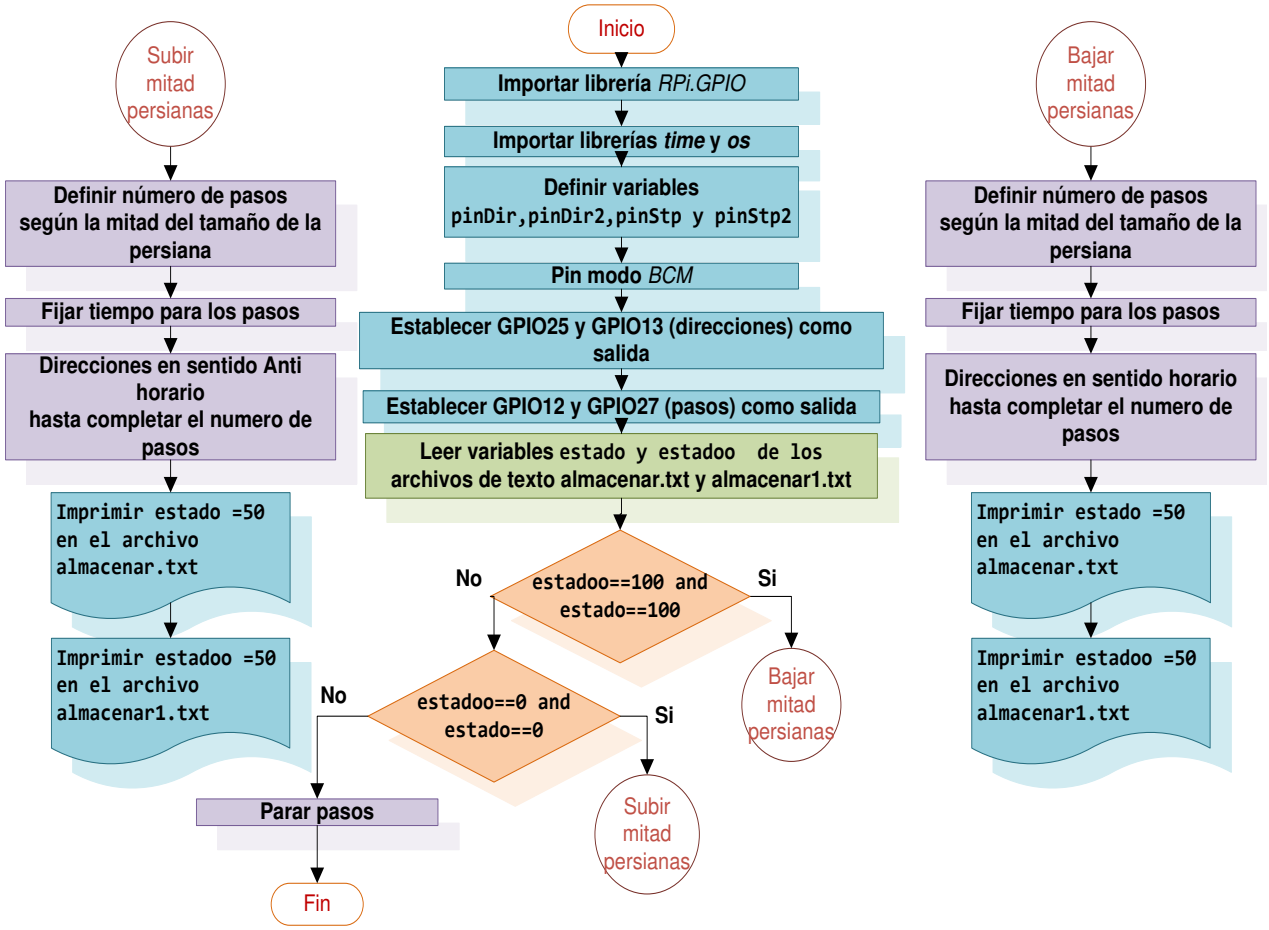


Figura 3.89 Diagrama de flujo para subir a la mitad y bajar a la mitad las persianas

Los diagramas de flujo de las Figura 3.87, Figura 3.88 y Figura 3.89 se diseñaron para el control de las dos persianas simultáneamente. A diferencia del control de una sola persiana que se leía un solo estado y se guardaba en un archivo de texto. Para el control de las dos persianas se lee dos variables de estado y se almacena en dos archivos de texto diferentes. Con estas variables de estado se realizan varios condicionales y se ejecutan determinadas funciones al igual que el control de una persiana. Sin embargo, a las dos cortinas se les da la misma dirección y los pasos se ejecutan en el mismo bucle *for*, tal cual se observa en la Figura 3.90.

```
GPIO.output(pinDir, GPIO.HIGH)
GPIO.output(pinDir2, GPIO.HIGH)
for i in range(0,numPas):
        GPIO.output(pinStp, GPIO.HIGH)
        GPIO.output(pinStp2, GPIO.HIGH)
        time.sleep(tiempo)
        GPIO.output(pinStp, GPIO.LOW)
        GPIO.output(pinStp2, GPIO.LOW)
        time.sleep(tiempo)
```

Figura 3.90 Bucle *for* para el control de las 2 persianas

Para el control de las dos persianas simultáneamente se realizaron los *scripts* motor1111.py, motor1112.py y motor1113.py.

- *Esquema del código para automatizar las cortinas*

Como se mencionó anteriormente la *Raspberry Pi* no tiene un conversor analógico-digital. Por lo cual se empleó el integrado MCP3008, que es un conversor A/D de 10 bits y 8 canales. Este es económico, fácil de conectar y no requiere ningún componente adicional. Además, permite la comunicación con la *Raspberry Pi* por medio de *SPI*.

Antes que nada, se debe instalar la librería *NumPy*. Esta es una librería de *Python* especializada en el cálculo numérico y el análisis de datos [42]. Se instaló digitando el comando *sudo apt-get install python-numpy* en la terminal tal y como se visualiza en la Figura 3.91.

```
pi@lab16:~ $ sudo apt-get install python-numpy
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
```

Figura 3.91 Instalación de la librería *NumPy*

Posteriormente, se diseñó el diagrama de flujo de la Figura 3.92 para crear el control automático de las cortinas. Después se desarrolló el *script* llamado fotores1.py. En el *script* inicialmente, se importaron las librerías. Además, se ocupó el módulo de *Python spidev* para interactuar con dispositivos *SPI* desde el espacio de usuario a través del controlador del *kernel* de *Linux*.

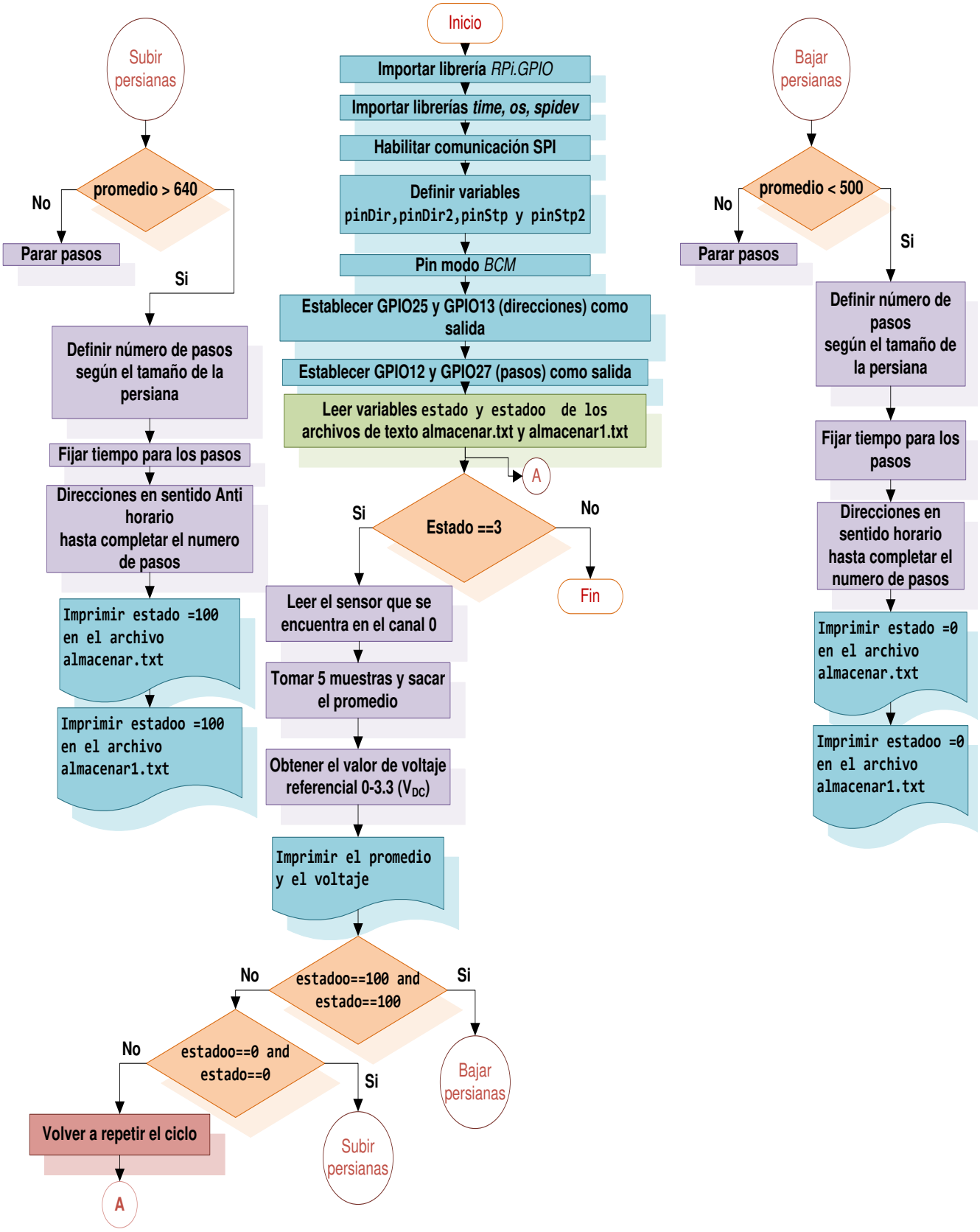


Figura 3.92 Diagrama de flujo para automatizar las cortinas

La lectura del sensor se llevó a cabo, en primer lugar habilitando el canal *SPI* con la línea de código *spi.open (0,0)*, luego se realizó la función llamada *analogRead ()* para leer los canales del conversor A/D MCP3008, tal cual se observa en la Figura 3.93.

```
spi = spidev.SpiDev()
spi.open(0,0)


def analogRead(pin):
        spi.max_speed_hz = 1350000
        adc = spi.xfer2([1,(8+pin) << 4,0])
        lec = ((adc[1]&3) << 8) + adc[2]
        return lec
```

Figura 3.93 Función para leer los canales del conversor A/D MCP3008

La función de lectura de los canales del integrado se compone de*:*

- *spi.max_speed_hz*

Es la velocidad máxima de bus *SPI*, se estableció el valor de 1350000

- *adc = spi.xfer2([1,(8+pin) << 4,0])*

La función **spi.xfer2()** manda tres bytes y retorna igualmente tres *bytes*. El primer *byte* que se envió es siempre **1**, es el *bit* de inicio, el segundo *byte* es la configuración del chip de acuerdo con la Figura 3.94 [43].

Además, *Single/Diff* son los modos en que el conversor trabaja. Para el *script* se empleó el modo simple. Con D0, D1 y D2 se puede realizar 8 combinaciones que son las correspondientes al número de canales del conversor. Con estos *bits* se elige el canal **(8+channel)**, 8 es el valor del modo simple más el canal leído, luego se corre todo cuatro posiciones para organizar los datos **<<4**, por otra parte, el tercer *byte* siempre es **0**.

En el caso del *script* se colocó en *channel* la variable pin para llamarlo más adelante en el programa, pero se puede colocar directamente el canal que se va a emplear del 0-7. Los datos leídos llegan a la *Raspberry Pi* a través del pin *MISO* según el ritmo de la señal de reloj *CLK*. Los datos recibidos se almacenan en la cadena de tres *bytes* adc [43].

Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7
					CH1 = IN-
0	0	0	1	differential	CH0 = IN- CH1 = IN+
0	0	1	0	differential	CH2 = IN+ CH3 = IN-
0	0	1	1	differential	CH2 = IN- CH3 = IN+
0	1	0	0	differential	CH4 = IN+ CH5 = IN-
0	1	0	1	differential	CH4 = IN- CH5 = IN+
0	1	1	0	differential	CH6 = IN+ CH7 = IN-
0	1	1	1	differential	CH6 = IN- CH7 = IN+

Figura 3.94 Configuración de bits del MCP3008 [43]

- lec = ((adc[1]&3) << 8) + adc[2]

Observando la Figura 3.95 se puede conocer la manera en que llegan los 3 *bytes* que son enviados por el conversor. Recordando la información mencionada en la función **(adc [1] & 3) << 8)**, el elemento 1 viene de la cadena de 3 *bytes* contenida en adc [] y se efectúa la operación *and* con el valor 3 para eliminar los dos primeros *bits*, moviendo 8 lugares a la izquierda con el fin de dejar espacio a los *bits* bajos **+ adc[2]** que se encuentran en el vector dos en la cadena de 3 *bytes* [43].

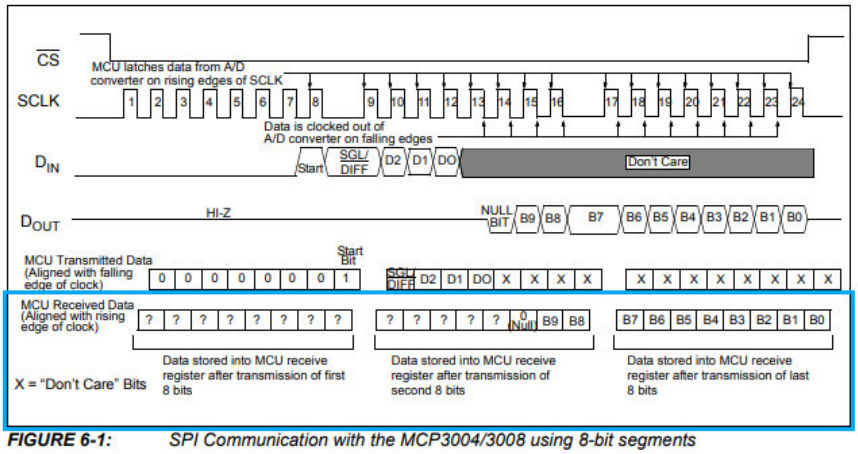


FIGURE 6-1: *SPI Communication with the MCP3004/3008 using 8-bit segments*

Figura 3.95 Trama de comunicación *SPI* del MCP3008 [43]

En la Figura 3.96 se observa cómo se efectuó la lectura del sensor que se encuentra ubicado en el canal 0 a través de la siguiente línea de código *lectura = analogRead(0)*, una vez obtenido el valor se realizaron 5 lecturas para obtener el promedio con el fin de hacer más exacta la medida y evitar inconsistencias.

```
suma = 0
for i in range(5):
        lectura = analogRead(0)
        suma = lectura + suma
        time.sleep(0.3)
promedio = suma/5
volts = (promedio * 3.3) / float(1023)
volts = round(volts,2)
print (volts)
```

Figura 3.96 Lectura del sensor de luz

Para conseguir el valor de voltaje se debió tener en cuenta que una lectura de 1023 se obtiene en 3,3 (V_{DC}), por otro lado, a un valor de 0 se consigue 0 (V_{DC}). Con esta premisa se generó la siguiente fórmula de la Ecuación 3.4:

$$\mathrm{Vs} = \frac{3{,}3}{1023} \,.promedio$$

Ecuación 3.4 Voltaje de salida en función de la lectura del sensor

Donde:

Vs : Voltaje de salida en función de la lectura del sensor (V_{DC})

$3{,}3$: Voltaje de alimentación del conversor (V_{DC})

1023 : Valor máximo del conversor 10 bits

promedio : Valor promedio que se obtiene de la lectura de 5 muestras por medio del sensor

Los valores de voltaje y la lectura promedio del sensor se guardan en un archivo de texto para más tarde ser representados en la interfaz web. Finalmente dependiendo si el promedio < 500 o el promedio > 640 se cierran o abren las cortinas respectivamente tal cual se indicó en los *scripts* anteriores.

***Scripts* para el control de las computadoras**

- *Esquema del código para encender un ordenador*

En primer lugar, se instaló el módulo *wakeonlan* mediante el comando *sudo apt-get install wakeonlan.* Sin embargo, al importar la librería al *script*, *Python* no la reconoció por lo que también se usó el comando *sudo pip install wakeonlan. PIP* es un administrador de paquetes de *Python* que permite la fácil instalación de módulos y librerías [44].

Más tarde, se realizó el diagrama de flujo de la Figura 3.97 para facilitar la creación del *script* llamado *index.py*.

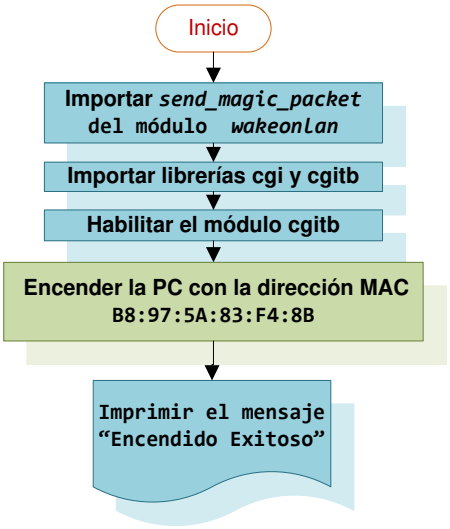


Figura 3.97 Diagrama de flujo para encender un ordenador

En la Figura 3.98 se observa, como del módulo *wakeonlan* se importó la función *send_magic_packet,* en esta función se ingresó la dirección *MAC* de la *PC* que se deseaba encender. Después que el ordenador recibe este paquete, la tarjeta de red le envía una señal a la placa base del equipo, que inicia el proceso para arrancarlo. Por otra parte, se utilizó el módulo *CGI* para probar el *script* desde el servidor web, para más luego imprimir un pequeño mensaje en formato *HTML* si el proceso de encendido resultó exitoso. En cambio, se empleó el módulo *cgitb* para detectar algún error inesperado que pueda ocurrir en la ejecución del *script* en el navegador. Este módulo se habilitó con la línea de código *cgitb.enable().*

```
from wakeonlan import send_magic_packet
import cgi
import cgitb
cgitb.enable()
send_magic_packet('B8:97:5A:83:F4:8B')
print 'Content-type: text/html\n\n'
print '<h1>Encendido Exitoso </h1>'
```

Figura 3.98 *Script* para encender la *PC* con la dirección *MAC* B8-97-5A-83-F4-8B

- *Esquema del código para apagar un ordenador*

Para apagar un ordenador se utilizó varios comandos del paquete *samba-common* por lo cual; primeramente, antes de desarrollar el *script* se lo instaló con el comando *sudo apt-get install samba-common.*

Seguidamente, se desarrolló el diagrama de flujo de la Figura 3.99, que sirvió para elaborar el *script* apagar.py.

Figura 3.99 Diagrama de flujo para apagar un ordenador

Después, se importó la librería *commands.* Esta librería permite ejecutar comandos de *Linux* a través de la función *commands.getoutput().*

```
import cgi
import cgitb
import commands
cgitb.enable()
commands.getoutput('net RPC SHUTDOWN -I 192.168.1.20 -U domotica%ma170939 -f -t 30 -C "Apagado en 30 segundos" ')
print 'Content-type: text/html\n\n'
print '<h1>Apagado Exitoso </h1>'
```

Figura 3.100 *Script* para apagar la *PC* con la dirección IP 192.168.1.20

En la Figura 3.100 se nota, que para apagar de manera remota un equipo *Windows* desde un sistema *Linux* se debe emplear el comando:

net RPC SHUTDOWN -I direcciónIP -U nombreUsuario%contraseña

Donde:

- *Dirección IP*: Es la IP del ordenador que se desea apagar de forma remota.
- Nombre de usuario: Nombre de algún usuario administrador
- %: Es el símbolo para separar el nombre del usuario de su contraseña.
- Contraseña: Es la contraseña del usuario administrador

El comando puede realizar varias funciones añadiendo la siguiente estructura al mismo [45]:

- -r: Para reiniciar el ordenador.
- -f: Para cerrar todas las aplicaciones abiertas de manera forzada.
- -t: Para especificar la duración en segundos
- -C: Para mostrar un mensaje en el ordenador a modo de aviso al usuario

Por lo tanto, con la línea de código *net RPC SHUTDOWN -I 192.168.1.20 -U domotica%ma170939 -f -t 30 -C "Apagado en 30 segundos"* significaría que: un ordenador *Windows* que usa la dirección IP 192.168.1.20 y posee un usuario administrador de nombre domótica con la contraseña ma170939, está forzado a cerrar todas las aplicaciones abiertas y en un tiempo de 30 segundos se apagará según se indicó en un mensaje en la pantalla al usuario del ordenador.

- *Esquema del código para reiniciar un ordenador*

Cuando se instalan varios programas, distribuciones, etc., por lo general, siempre se requiere reiniciar el ordenador, para facilitar esta labor se diseñó el diagrama de flujo de la Figura 3.101 y el *script* llamado reiniciar.py de la Figura 3.102. Este *script* se realizó modificando el programa anterior ya que en la línea de código *net RPC SHUTDOWN -I 192.168.1.20 -U domotica%ma170939 -f -r -t 30 -C "Reinicio en 30 seg"* se agregó la opción -r que en vez de apagar la máquina la reinicia, además también se cambió el mensaje que se muestra al usuario del ordenador.

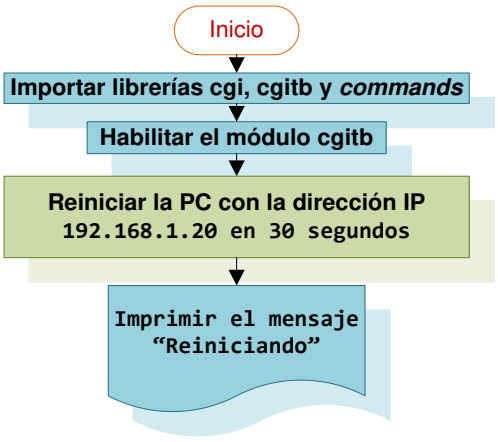


Figura 3.101 Diagrama de flujo para reiniciar un ordenador

```
import cgi
import cgitb
import commands
cgitb.enable()
commands.getoutput('net RPC SHUTDOWN -I 192.168.1.20 -U domotica%ma170939 -f -r  -t 30 -C "Reinicio en 30 seg" ')
print 'Content-type: text/html\n\n'
print '<h1>Reiniciando </h1>'
```

Figura 3.102 *Script* para reiniciar la *PC* con la dirección IP 192.168.1.20

- *Esquema del código para cancelar el reinicio o apagado de un ordenador*

Con el fin de cancelar la actividad de reinicio o apagado de un ordenador en los *scripts* previamente diseñados se dejó un margen de tiempo de 30 segundos en los cuales se podrá ejecutar este *script* cancelar.py. Esto se hizo, ya que el usuario del ordenador podría olvidarse de guardar algún trabajo importante o tal vez se presionaron mal los botones en la interfaz web. Para comenzar, inicialmente se estableció el diseño del *script* mediante el diagrama de flujo de la Figura 3.103.

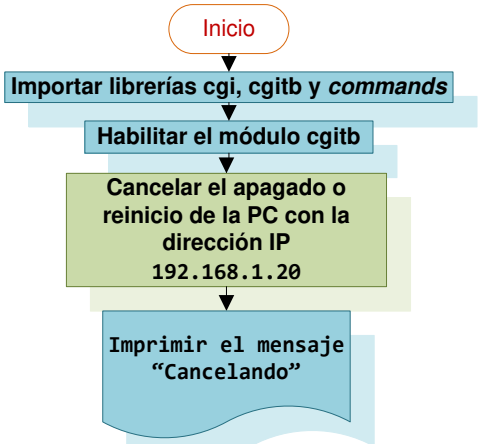


Figura 3.103 Diagrama de flujo para cancelar el apagado o reinicio de un ordenador

Acto seguido, se empleó la siguiente línea de código *net rpc abortshutdown -I 192.168.1.20 -U domotica%ma170939.* Esta permite abortar la ejecución del apagado o reinicio del ordenador tal cual se observa en la Figura 3.104.

```
import cgi
import cgitb
import commands
cgitb.enable()
commands.getoutput('net rpc abortshutdown  -I 192.168.1.20 -U domotica%ma170939 ')
print 'Content-type: text/html\n\n'
print '<h1>Cancelando </h1>'
```

Figura 3.104 *Script* para cancelar el apagado o reinicio de la *PC*

Es importante, dar a conocer que el usuario del ordenador también puede ingresar al cmd y digitar el comando *shutdown -a* para realizar la misma actividad mencionada anteriormente, así como se muestra en la Figura 3.105.

Figura 3.105 Cancelar la ejecución del apagado o reinicio desde el cmd

- *Esquema del código para conocer el estado de un ordenador*

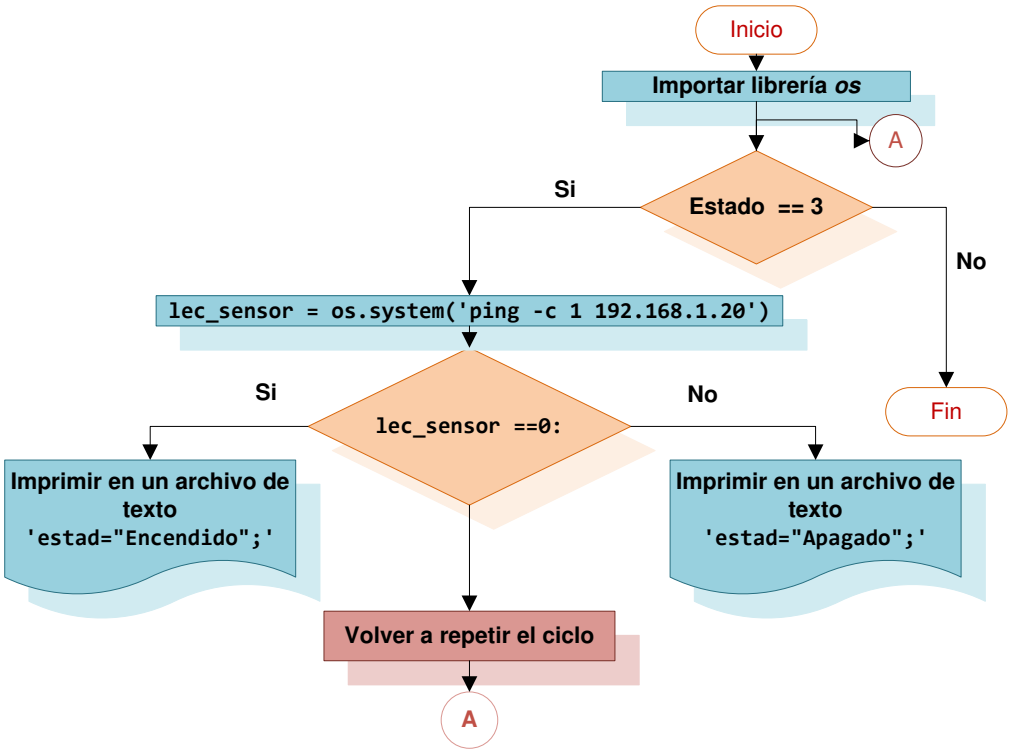


Figura 3.106 Diagrama de flujo para conocer el estado de un ordenador

En el diagrama de flujo de la Figura 3.106 se puede conocer el proceso que lleva a cabo el *script* estado.py. Principalmente, en el diagrama se destaca la manera en que se sabe si el ordenador se encuentra activo o no. Para determinar esto se usó el comando *ping* justo como se visualiza en la Figura 3.107. Mediante la opción -c se específica el número de solicitudes de eco que se desea ejecutar.

```
lec_sensor = os.system('ping -c 1 192.168.1.20')
```

Figura 3.107 *Ping* para determinar el estado del ordenador

Más adelante, si existió respuesta por parte del ordenador se imprime el mensaje encendido, o apagado si no la hubo. Estos estados se guardan en un archivo de texto para finalmente, ser representados en la interfaz *web*. Los *scripts* anteriores además se emplearon como base para crear el control de todos los demás ordenadores.

Scripts para el control del prototipo a través de un control remoto

Esta funcionalidad se agregó al prototipo primordialmente para que los usuarios que sean adultos mayores y personas que tengan alguna discapacidad que les impida desplazarse logren emplear el proyecto más fácilmente.

En primer lugar, se realizó el diagrama de flujo de la Figura 3.109 para más adelante desarrollar el *script ir.py*. En el *script* inicialmente se importó la librería para el control de los pines *GPIO*. Después se empleó el módulo *datetime*, este proporciona clases para manipular fechas y horas de forma sencilla y compleja [46].

Posteriormente, se definió las variables y se estableció el pin en el que se conectaría el receptor infrarrojo. Además, se crearon 3 listas, en la primera lista llamada *Buttons* se almacenó los valores en formato hexadecimal de las teclas del control remoto que se ocupó. Para obtener estos valores se empleó la aplicación *IR-Code-Decoder—master*, esta se puede descargar del enlace https://github.com/Lime-Parallelogram/IR-Code-Decoder--, tal cual se mira en la Figura 3.108.

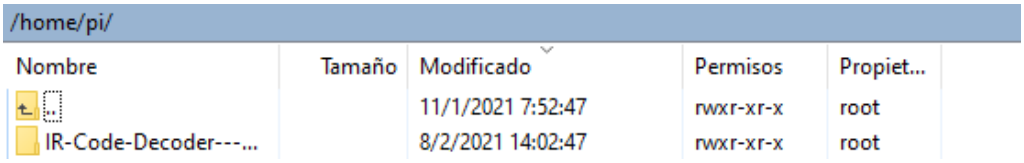


Figura 3.108 Aplicación *IR-Code-Decoder--master*

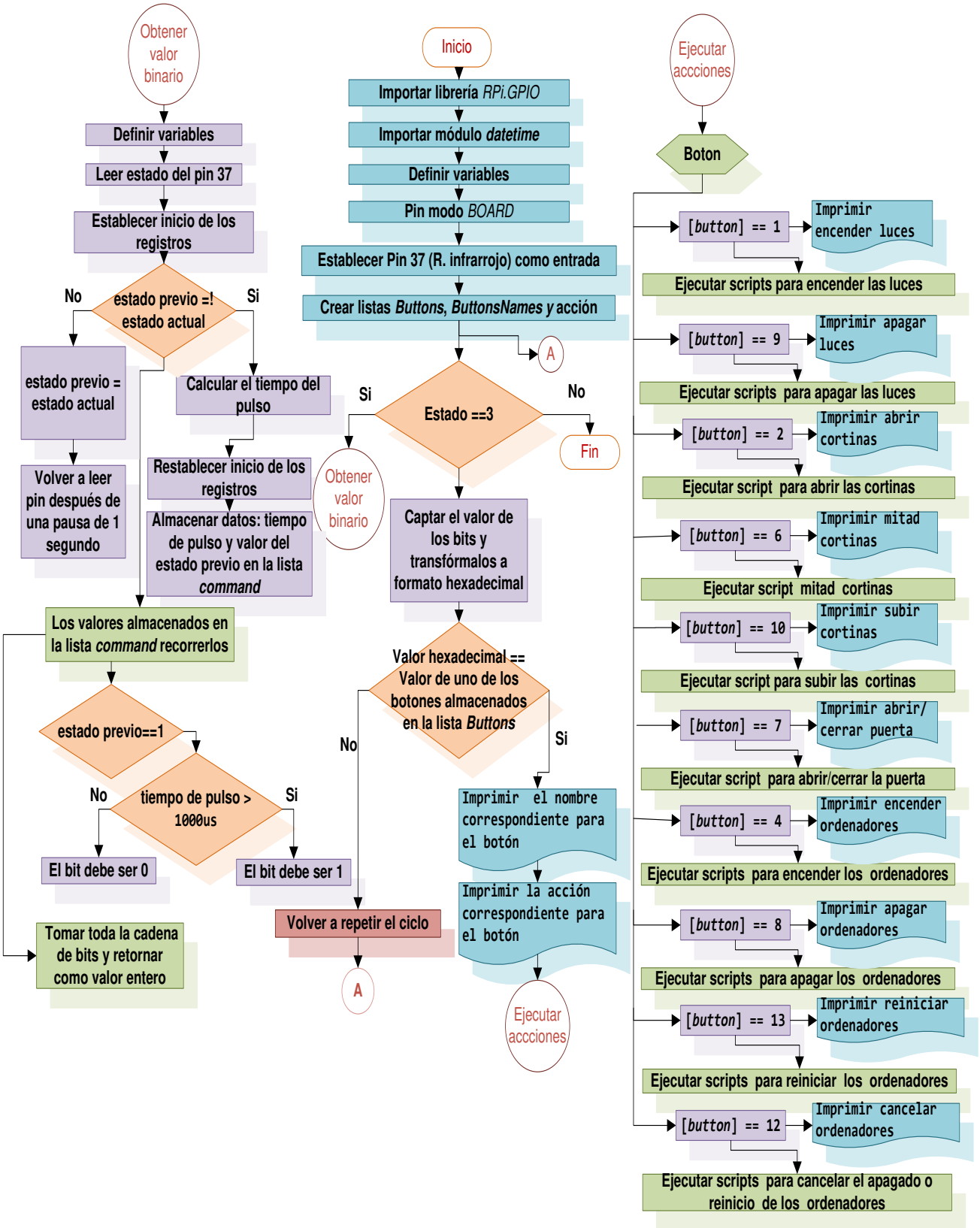


Figura 3.109 Diagrama de flujo para el control del prototipo a través de un control

Una vez descargada la aplicación se ingresó a la capeta *IR-Code-Decoder---master* y se ejecutó el archivo *GUI.py* tal cual se visualiza en la Figura 3.110. Más tarde, en el programa se digitó el pin en donde se conectó el receptor infrarrojo y también se dio un nombre al archivo en donde se desea guardar los códigos de cada tecla. En las Figura 3.111 y Figura 3.112 se aprecia lo mencionado previamente.

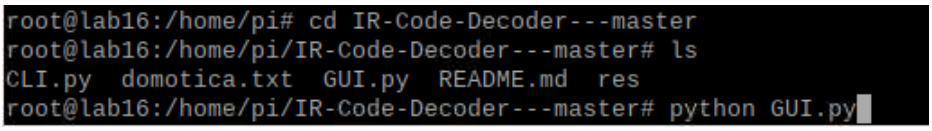


Figura 3.110 Ejecución de *GUI.py*

Figura 3.111 Ingreso del pin donde está conectado el receptor infrarrojo

Figura 3.112 Ingreso del nombre del archivo donde se desea guardar los datos

A continuación, en el menú de la Figura 3.113 se pulsó sobre la opción *Start Test* para empezar con el proceso de recolección de los datos, seguidamente se presionó sobre una tecla del control remoto y finalmente con la opción *Save Command* se almacenó su valor hexadecimal.

Figura 3.113 Inicio de la recolección de datos de las teclas

Al finalizar todo este proceso se obtuvieron los códigos de las teclas con su valor correspondiente en el archivo de texto domotica.txt como se nota en la Figura 3.114. El control remoto del cual se leyó los valores es el de la Figura 3.115, sin embargo, se puede usar cualquier otro, por ejemplo, el de la TV.

/home/pi/IR-Code-Decoder---master/domotica.txt - root@192.168.1.21 - Editor - WinSCP

Codificación Color

```
Button codes regarding domotica IR controller:
Button Code - tomateoscuro: 0x300f710efL
Button Code - tomatemedio: 0x300f730cfL
Button Code - tomateclaro: 0x300f708f7L
Button Code - amarillo: 0x300f728d7L
Button Code - verde: 0x300f7906fL
Button Code - azulmarino: 0x300f7b04fL
Button Code - azulmedio: 0x300f78877L
Button Code - verdeoscuro: 0x300f7a857L
Button Code - azulcielo: 0x300f750afL
Button Code - moradooscuro: 0x300f7708fL
Button Code - moradoclaro: 0x300f748b7L
Button Code - rosado: 0x300f76897L
```

Figura 3.114 Archivo con los valores hexadecimales de las teclas

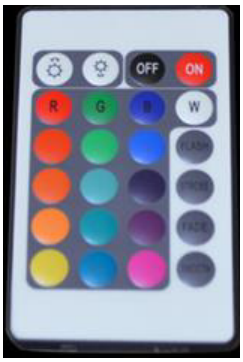


Figura 3.115 Control remoto utilizado [47]

En la segunda lista llamada *ButtonsNames* se guardó los nombres que se asignó a las teclas del control, por otra parte, en cambio en la tercera lista, nombrada acción, se colocó una numeración a los botones. Esto se realizó con la finalidad que cuando se active la lectura del receptor infrarrojo desde la interfaz web y se pulse sobre una tecla del control remoto, el programa compare el valor de la tecla pulsada con los valores hexadecimales almacenados, si al recorrer la lista *Buttons* encuentra que coincide con uno de estos valores imprimirá el nombre del botón y ejecutará el *script* correspondiente.

```
inData = convertHex(getBinary()) #obtener el valor hexadecimal entrante
for button in range(len(Buttons)):
        # Recorre todos los valores de la lista
        if hex(Buttons[button]) == inData: #Comprueba esto contra el ingreso
                print(ButtonsNames[button]) # Imprime el nombre correspondiente para el boton
                print(accion[button]) # Imprime la accion correspondiente para el boton
                if accion[button] == 1:
                        print("encender luces")
                        os.system ('sudo python /var/www/html/cgi-enabled/focoff.py')
                        os.system ('sudo python /var/www/html/cgi-enabled/focoff1.py')
                        os.system ('sudo python /var/www/html/cgi-enabled/focoff5.py')
                        os.system ('sudo python /var/www/html/cgi-enabled/focoff4.py')
                        os.system ('sudo python /var/www/html/cgi-enabled/focoff3.py')
```

Figura 3.116 Comparación del botón pulsado con los valores almacenados

En la Figura 3.116 se puede observar que los *scripts* que se van a ejecutar son los mismos que previamente se diseñaron y se llevan a cabo a través del comando *os.system ('sudo python nombre-del-script.py')* más no, nuevamente se realiza la programación de estas actividades.

La parte nueva en el *script* por así decirlo es la lectura de los valores hexadecimales de las teclas a través del receptor infrarrojo. A diferencia de *Arduino* que cuenta con la librería *IRremote* que facilita la lectura IR y programación, no se encontró una librería que realice la misma función en la *Raspberry Pi* por lo cual, se diseñó la lectura en el *script* a través de líneas de código.

Dicho lo anterior, en el *script* se estableció el inicio de la trama con la línea de comando *startTime = datetime.now(),* para después si el estado que se lee del receptor infrarrojo cambia calcular este tiempo y almacenarlo en la lista *command* junto con el tiempo. Este proceso se va realizando cada 1 segundo y constantemente se va posicionando en la parte final de la lista *command* con la función *append()* tal cual se ve en la Figura 3.117.

```
if previousValue != value:
        now = datetime.now()
        pulseTime = now - startTime # Calcula el tiempo del pulso
        startTime = now #Restablecer la hora de inicio
        command.append((previousValue, pulseTime.microseconds)) # Almacenar datos registrados
```

Figura 3.117 Cálculo y almacenamiento del tiempo del pulso de la tecla presionada

Como se mencionó anteriormente, el tiempo de transmisión de un bit 0 es alrededor de 1000 (µs) por lo que un valor superior a este, específicamente 2,25 (ms) será un bit 1 esto se plasma en las líneas de código de la Figura 3.118.

```
for (typ, tme) in command:
        if typ == 1: #Si mirando el periodo de descanso
                if tme > 1000: #Si pulso mayor a 1000us
                        binary = binary *10 +1 # Debe ser 1
                else:
                        binary *= 10 # Debe ser 0
```

Figura 3.118 Establecimiento de los bits de la transmisión

Cuando se obtiene toda esta cadena de caracteres se debe transformar primero a entero en base 2 y luego a hexadecimal con la función *hex* () tal como está en la Figura 3.119.

```
#Convertir valor a hexadecimal
def convertHex(binaryValue):
        tmpB2 = int(str(binaryValue),2) # Base propia 2 del equipo
        return hex(tmpB2)
```

Figura 3.119 Conversión de la cadena de bits a formato hexadecimal

Scripts para el control del prototipo a través de Telegram

Para crear el *script bot_pihole.py* se empleó la *API Telegram Bot* esta una herramienta poderosa para realizar múltiples funcionalidades como: automatización de acciones, trabajo con usuarios, tiendas *online*, juegos y mucho más. Un *Bot* es una cuenta especial que se crea en *Telegram* y no necesita asociarse a ningún teléfono en concreto a diferencia de *WhatsApp*.

En el caso del prototipo se utilizó para que el usuario autorizado pueda mandar mensajes a la *Raspberry Pi*, ejecutar *scripts* y conocer el estado de algunas funciones básicas de esta. La *Raspberry Pi* a su vez contestará las peticiones que se realicen al *Bot* e interactuará con las cuentas de Telegram a través de su *ID*. Existen 2 modos de programar el *script*, a través de *Polling* y *Webhooks*. La técnica de *Polling* consiste en realizar un sondeo largo a la espera de mensajes y peticiones para procesarlos. En cambio, *Webhooks*, supone configurar el *Bot* con una dirección de *Internet*, en la que se coloca el *script* y posteriormente se invoca cada vez que llega un mensaje o petición [48].

El método óptimo es el sistema de *Webhooks*, pero requiere de más infraestructura. Para su uso se debe tener una dirección IP pública de Internet, con un certificado seguro (*SSL*). Sin embargo, la técnica de *Polling* puede hacerse perfectamente en la *Raspberry Pi* simplemente es necesario tener conexión a Internet. No obstante, en un periodo prolongado de uso el método *Polling* comienzan a devolver periódicamente un error 504(*Gateway Timeout*) y el *Bot* se cae. Aún más si se ejecutan varios *Bots* al mismo tiempo, aumenta la probabilidad de encontrar errores.

Si bien es molestoso sondear periódicamente el servidor, pagar por este proceso va más allá de los alcances del proyecto, por lo cual se ocupó el método *Polling* pero se trató de solventar el problema de la caída del *Bot*, lanzando automáticamente el *script* cada cierto tiempo con el fin que siempre se encuentre en línea y resulte casi imperceptible al usuario del prototipo. Además, porque en el prototipo existe un solo *Bot*, no varios. Primeramente, se diseñó el diagrama de flujo de la Figura 3.120.

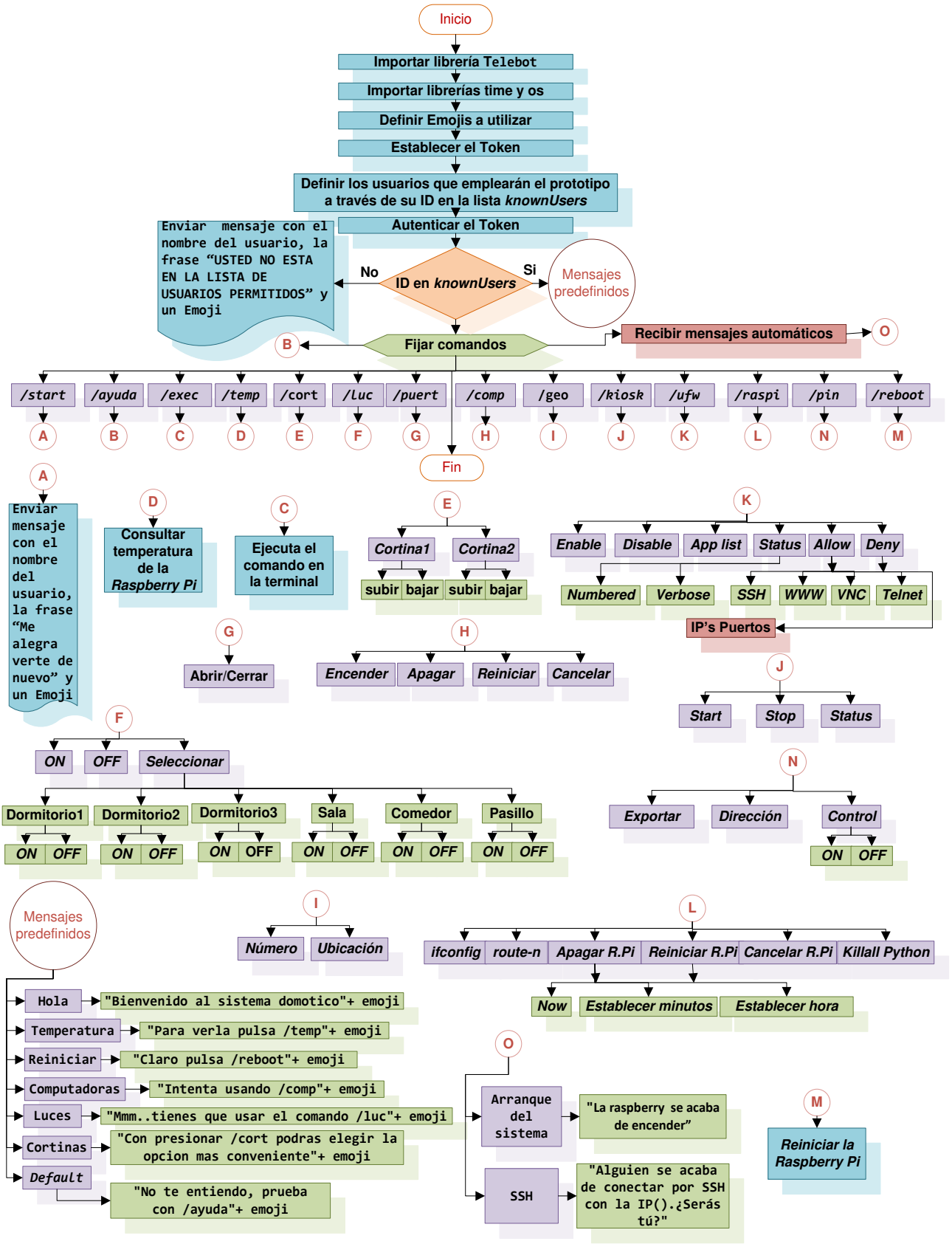


Figura 3.120 Diagrama de flujo para el control del prototipo a través de *Telegram*

Después, en la *Raspberry Pi,* se creó la carpeta aplicaciones con el comando *mkdir* dentro del directorio *root,* para almacenar los archivos que utiliza el *script bot_pihole.py*, tal cual se visualiza en la Figura 3.121.

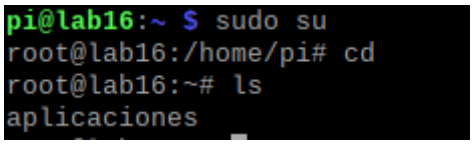


Figura 3.121 Creación de la carpeta para guardar los archivos de *bot_pihole.py*

Más tarde, se instaló la *API Telegram Bot* para esto primero se utilizó el comando *sudo apt-get install git*, luego dentro de la carpeta aplicaciones se efectuó la descarga de la librería con el comando *git clone* https://github.com/eternnoir/pyTelegramBotAPI, después se tecleó en la terminal *git checkout 3.6.6* para evitar errores. Si el proceso fue exitoso se debería haber creado dentro de aplicaciones otra carpeta con el nombre *pyTelegramBotAPI* así como se muestra en la Figura 3.122.

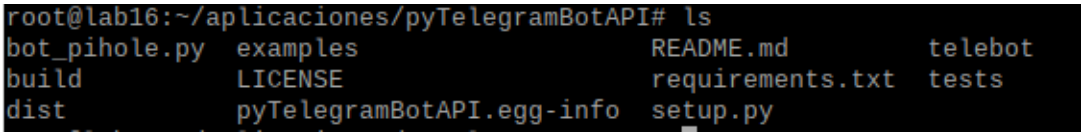


Figura 3.122 Descarga de *pyTelegramBotAPI*

Ahora bien, para que la librería funcione con *Python* se necesita instalar un añadido a este, esto se llevó a cabo tecleando el comando *sudo apt-get install python-setuptools.* Finalmente, se digitó el comando *sudo python setup.py install* para completar la instalación. Posteriormente, se creó el *script* con el comando *sudo nano bot_pihole.py* en la carpeta *pyTelegramBotAPI.*

A continuación, se detalla el proceso que se empleó para generar y configurar el *Bot.* Para empezar, se buscó en la aplicación de *Telegram* en la barra de búsqueda *@BotFather* como se nota en la Figura 3.123.

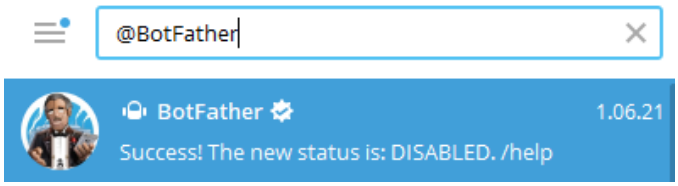


Figura 3.123 Búsqueda de *@BotFather*

BotFather es un *Bot* que permite generar nuevas cuentas de *Bots,* además de configurar las existentes. Para utilizarlo se debe presionar sobre el botón *start*, después este devolverá una lista de comandos que se pueden usar, tal cual se ilustra en la Figura 3.124.

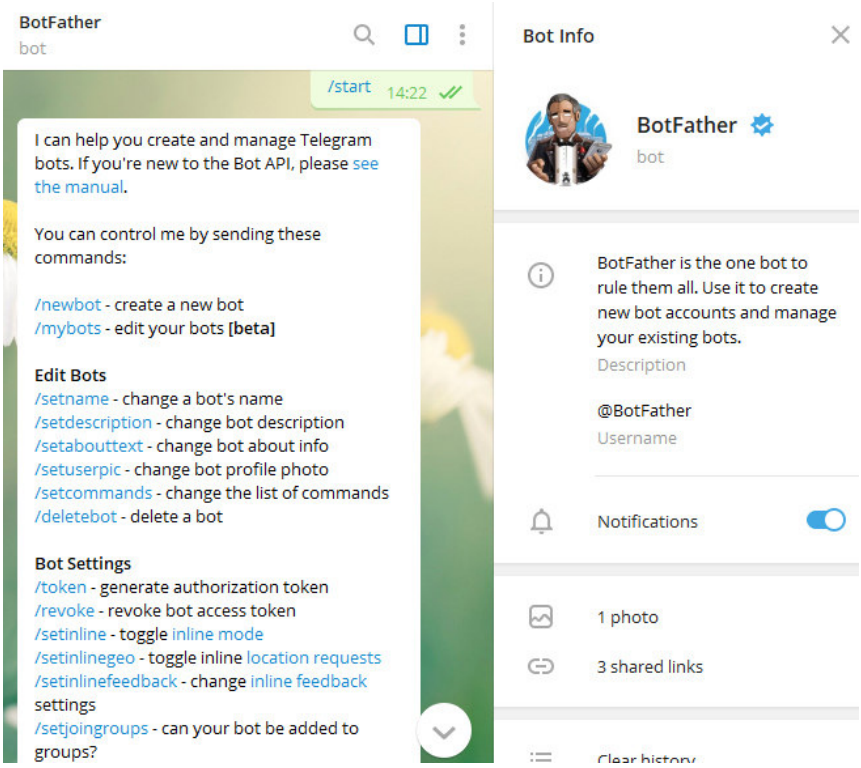


Figura 3.124 Comandos de *@BotFather*

Seguidamente, se presionó sobre la opción /newbot, y se nombró al Bot como *raspibot*. Entonces *BotFather* pidió que se ingrese un nombre de usuario que termine en "*bot*", cabe mencionar que *BotFather* no dejará realizar el registro si el usuario ya existe. El nombre de usuario que se colocó fue *esfotbot*, inmediatamente luego de este proceso, se puede buscar en la aplicación de *Telegram* el *Bot* con su respectivo nombre de usuario anteponiendo el símbolo "@" por ejemplo: *@esfotbot*.

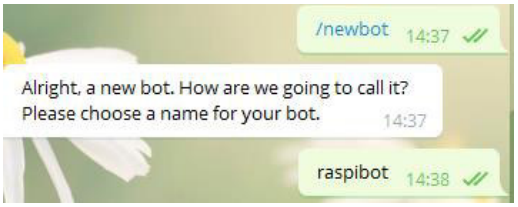


Figura 3.125 Nombre del *Bot*

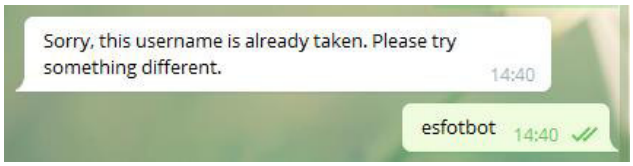


Figura 3.126 Registro del nombre de usuario del *Bot*

En las Figura 3.125 y Figura 3.126 se encuentra lo mencionado previamente, más adelante *BotFather* envió un *Token* tal cual se observa en la Figura 3.127. Este es el número de identificación del *Bot*, que se empleó para programar el *script* en *Python* y llamarlo a través de la *Raspberry Pi*.

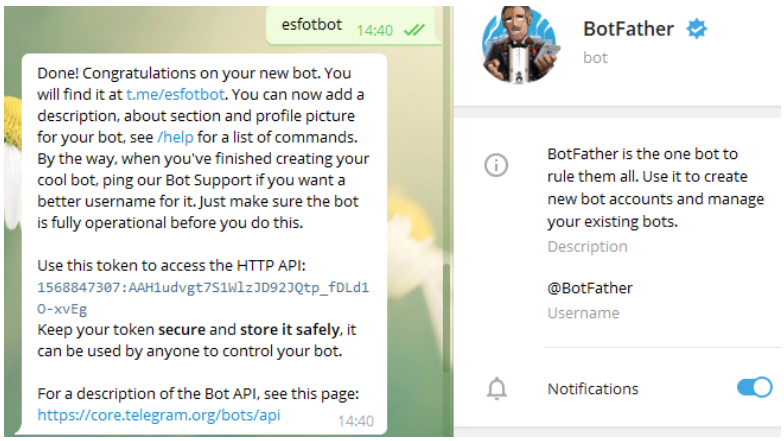


Figura 3.127 Número de identificación del *Bot*

Como configuraciones básicas del *Bot* se añadió una foto con el comando */setuserpic* y una pequeña descripción con la opción */setdescription*. Esto se muestra en la Figura 3.128. En *BotFather* hay más opciones que el lector puede emplear, como, por ejemplo: cambiar el *Token*.

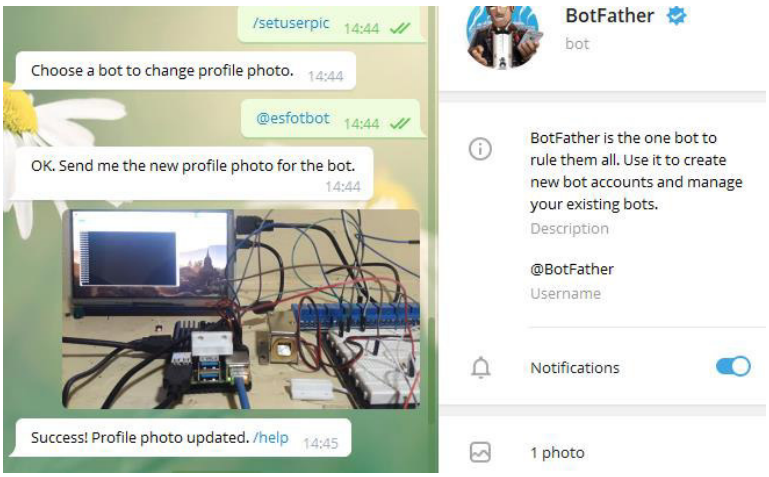


Figura 3.128 Configuraciones básicas del *Bot*

Otro aspecto que se tomó en cuenta en esta parte es la obtención del *ID* de los usuarios que utilizarán el prototipo domótico, para este fin se ocupó el *Bot @userinfobot* tal como se logra ver en la Figura 3.129. Este *Bot* envía al usuario de *Telegram* su número único de identificación en la aplicación y sirvió para controlar el acceso al prototipo. En la Figura 3.130 se observa cómo se obtiene el *ID* de un usuario de *Telegram*.

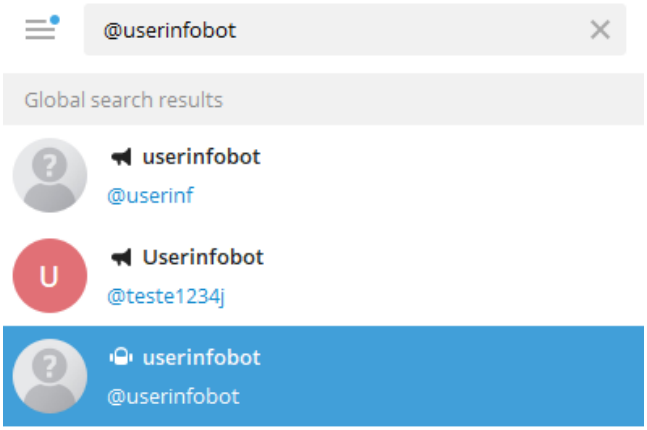


Figura 3.129 *Bot @userinfobot*

Figura 3.130 *ID* de un usuario de *Telegram*

Después de importar todas las librerías, se creó varias variables, en estas se almacenó en formato *Unicode* el valor de los emojis que se ocuparon tal cual se mira en la Figura 3.131.

```
# emojis
carasonri = u'\U0001F605'
comenzar = u'\U0000231A'
ayud = u'\U0001F46E'
coman = u'\U0001F4CB'
```

Figura 3.131 Emojis en formato *Unicode*

Acto seguido, se definió la lista *knownUsers* = [*IDs* de los usuarios] en la cual se debe ingresar los *IDs* de los usuarios que se desea que empleen el prototipo, también en otra variable que se llamó *Token* se guardó la identificación del *Bot* y a través de la función *telebot.TeleBot(TOKEN)* se lo autenticó. En la Figura 3.132 se visualiza lo dicho.

```
#autentificacion del bot
bot = telebot.TeleBot(TOKEN)
bot.set_update_listener(listener) # registro listener
```

Figura 3.132 Autentificación del *Bot*

Más tarde, en la lista *commands* se definió los comandos que emplea el *Bot*, con una descripción de su funcionalidad para que el usuario pueda manejar fácilmente el prototipo. Los comandos que se colocaron al *Bot* se presentan en la Figura 3.133.

```
commands = {
            'start': 'Empieza el Bot'+' '+comenzar,
            'ayuda': 'Da informacion sobre los comandos disponibles'+' '+ ayud,
            'exec': 'Ejecuta un comando'+' '+ coman,
            'temp': 'Comprueba la temperatura de la raspberry'+' '+ temp,
            'cort': 'Controla las cortinas'+' '+ cort,
            'luc': 'Controla las luces'+' '+ lu,
            'puert': 'Controla la puerta'+' '+ puer,
            'comp': 'Controla las computadoras'+' '+ comp,
            'geo': 'Puede obtener su ubicacion y su # de contacto '+' '+ pin,
            'kiosk':'Configuracion de Google Chrome Kiosk'+' '+ panya,
            'ufw':'Configuracion del cortafuegos ufw'+' '+ fuego,
            'raspi':'Configuracion basica de la Raspberry Pi'+' '+ raspib,
            'pin':'Controla la salida de un pin de la Raspberry por Terminal'+' '
            'reboot': 'Reinicia el servidor'+' '+ rei
```

Figura 3.133 Comandos del *Bot*

Enseguida se definió la función *listener* que se usa para escuchar los mensajes que llegan al *Bot* e imprime el nombre del usuario, su *ID* y el texto que se recibió. En la Figura 3.134 se encuentra la parte de código que efectúa lo mencionado.

```
def listener(messages):
#Escuchamos los mensajes que nos llegan de los usuarios e imprimimos por consola su id y
    for m in messages:
        if m.content_type == 'text':
            # imprime el mensaje por consola
            print str(m.chat.first_name) + " [" + str(m.chat.id) + "]: " + m.text
```

Figura 3.134 Función para escuchar los mensajes que llegan al *Bot*

Cuando un usuario ejecuta cualquiera de los comandos definidos anteriormente, si su *ID* está almacenado en la lista *knownUsers* el *Bot* efectúa una acción como enviar un mensaje o poner en marcha un *script*. Caso contrario, impide el acceso y le informa que no se encuentra en la lista de usuarios permitidos junto con un emoji.

En gran parte del código se utilizó la función *message_handler()*, por lo que es importante saber que esta sirve como un controlador de mensajes. Los controladores de mensajes constan de uno o varios filtros. Cada filtro devuelve *True* para un determinado mensaje, para que sea elegible por el manejador de mensajes.

Un controlador de mensajes se declara de la siguiente manera (siempre que *Bot* sea una instancia de *TeleBot*):

@ bot . message_handler (filtros)

def funcion_nombre (mensaje):

    bot . reply_to (mensaje , "Este es un controlador de mensajes")

Donde:

funcion_nombre no está sujeto a ninguna restricción. Se permite cualquier nombre de función con los manejadores de mensajes. La función debe aceptar como máximo un argumento, que será el mensaje que la función debe manejar. Filtros es una lista de argumentos de palabras clave. Un filtro se declara en la forma: Nombre = argumento. Un controlador puede tener varios filtros. *TeleBot* admite los filtros que se presentan en la Tabla 3.8.

Tabla 3.8 Filtros de la función *message_handler()* [49]

Nombre	Descripción
content_types	Lista de cadenas *['text'], ['document'], ['audio'],* etc.
regexp	Una expresión regular como cadena.
commands	Un comando que está en la lista de cadenas
func	Una función (*lambda* o referencia de función)

Habiendo explicado lo anterior, se ocupó de varias formas distintas la función message_*handler*(), primeramente para ejecutar los comandos que se definieron en la lista *commands*. En la Figura 3.135 se visualiza cómo se empleó con el comando *start*, sin embargo, este método se aplicó a todos los demás comandos.

```
@bot.message_handler(commands=['start'])
def command_start(m):
    cid = m.chat.id
    if cid not in knownUsers:
        bot.send_message(cid,str(m.from_user.first_name)+"\n"+"USTED NO ESTA EN LA LISTA DE USUARIOS PERMITIDOS")
        bot.send_message(cid,cat)

    else:
        bot.send_message(cid, 'Muy buenas, ' + str(m.from_user.first_name) + '. Me alegra verte de nuevo.'+sonri+"
```

Figura 3.135 Filtro para manejar todos los mensajes de texto del comando *start*

La segunda forma en que se empleó esta función, fue para identificar un mensaje recurrente que forma parte de una cadena de texto, en la Figura 3.136 se nota que a través del argumento *regexp* se define la expresión Hola, lo que significa que cuando el usuario ingrese este mensaje en el *Bot* puedo hacerlo de manera exacta “Hola” o puede formar parte de una oración más extensa como “hola prototipo” de igual manera el programa identifica la estructura y ejecuta una acción como por ejemplo: de responder con un mensaje de Bienvenida. Si se desea también se puede hacer que el *Bot* realice determinada función con la palabra específica a través de *message.text* == palabra deseada.

```
# Filtros para un mensaje que coincida con las palabras
#@bot.message_handler(func=lambda message: message.text == "Hola")
@bot.message_handler( regexp = "Hola")
def command_text_hi(m):
    cid = m.chat.id
    if cid not in knownUsers:
        bot.send_message(m.chat.id, "PERMISO DENEGADO"+adver)
        bot.send_message(m.chat.id, enon)
    else:
        bot.send_message(m.chat.id,"Bienvenido al sistema domotico")
        bot.send_message(m.chat.id, carasonri)
```

Figura 3.136 Filtro para un mensaje regular y específico

Otra manera de emplear la función *message_handler()* fue a través de definir una función y un tipo de contenido. En la Figura 3.137 se aprecia, como cada vez que el usuario ingresa texto por *default* es decir sin sentido, le envía un mensaje automático dándole una posible ayuda a sus requerimientos. El contenido que se puede usar no se limita solo a texto, se puede detectar documentos, videos, audios, imágenes, entre otros.

```python
@bot.message_handler(func=lambda message: True, content_types=['text'])
def command_default(m):
    cid = m.chat.id
    if cid not in knownUsers:
        bot.send_message(m.chat.id, "PERMISO DENEGADO"+adver)
        bot.send_message(m.chat.id, enon)
    else:
        # Este comando se repite en un mensaje normal
        bot.send_message(m.chat.id, "No te entiendo, prueba con /ayuda")
        bot.send_message(m.chat.id, ayuud)
```

Figura 3.137 Filtro para cuando se ingresa contenido tipo texto

En las figuras anteriores, se utilizó el comando *send_message* que permite enviar un mensaje, no obstante, todos los métodos *API* se encuentran en la clase *TeleBot*. Solo que se les cambia el nombre para seguir las convenciones de nomenclatura comunes de *Python*. Por ejemplo: *getMe* se cambia el nombre a *get_me* y *sendMessage* a *send_message.* En la Tabla 3.9 se colocó los principales comandos de la librería *Telebot*.

Tabla 3.9 Principales comandos de la librería *Telebot* [50]

Nombre	Descripción
send_message	Envía un mensaje
send_photo	Envía una foto
send_document	Envía un documento (.doc, .txt, .pdf, etc.)
send_video	Envía un video
send_audio	Envía un audio
send_location	Envía la localización de un usuario
forward_message	Reenvía un mensaje
send_chat_action	Envía acciones de chat

Para crear los menús y submenús del prototipo se empleó 2 modos de programación mediante la librería *Telebot*. El primer modo es *ReplyKeyboardMarkup()* que crea un menú reemplazando el teclado del usuario en la aplicación de *Telegram*, lo que resultó muy útil para colocar comandos de *Linux* ordenados por secciones y dar mayor facilidad al usuario ya que, de otra forma se tendría que recordar el comando completo y escribirlo después. Este modo se aplicó a los comandos *kiosk, geo, ufw, raspi y pin*. En la Figura 3.138 se nota como se creó las columnas del menú *kiosk* y se nombró los botones, esta misma manera se utilizó en los demás comandos mencionados anteriormente.

```
markup = types.ReplyKeyboardMarkup(row_width=1, resize_keyboard=True)
#itembtna = types.KeyboardButton("Cortina1"+" "+vi)
itembtnv = types.KeyboardButton('/exec sudo systemctl start kiosk.service')
itembtnc = types.KeyboardButton('/exec sudo systemctl stop kiosk.service')
itembtnd = types.KeyboardButton('/exec sudo systemctl status kiosk.service')
#markup.row(itembtna, itembtne)
#markup.row(itembtnv, itembtnc, itembtnd)
markup.row(itembtnv)
markup.row(itembtnc)
markup.row(itembtnd)
bot.send_message(m.chat.id, "Seleccione por favor:", reply_markup=markup)
```

Figura 3.138 Menú que reemplaza el teclado del usuario

Por otra parte, el modo *InlineKeyboardMarkup()* permite mostrar al usuario el menú en forma de botones en la sección de mensajes de *Telegram*. Esta forma es más eficiente si se desea ejecutar *scripts* o programas en la *Raspberry Pi* ya que devuelve una función de llamada cada vez que se presiona sobre un botón del menú. Este modo se utilizó con los comandos *cort, luc, puert* y *comp*. En las Figura 3.139 y Figura 3.140 se da a conocer la forma cómo se realizó la programación del comando *cort*, no obstante, sirve de base para comprender la estructura de los demás comandos que se nombraron previamente.

```
key = types.InlineKeyboardMarkup()
but_1 = types.InlineKeyboardButton(text=" "+"Cortina1"+" "+vi, callback_data="NumberOne")
but_2 = types.InlineKeyboardButton(text=" "+"Cortina2"+" "+jo, callback_data="NumberTwo")
key.add(but_1, but_2)
bot.send_photo(message.chat.id,photo=open('/var/www/html/Montar/2126/luz/images/corti.png', 'rb'))
bot.send_message(message.chat.id, "Iniciando el sistema de las cortinas", reply_markup=key)
```

Figura 3.139 Menú en línea

Cabe resaltar que los *scripts* que se ejecutan en estos menús y submenús son los mismos que ya se estructuraron en las secciones anteriores.

```
@bot.callback_query_handler(func=lambda c:True)
def inline(c):
        if c.data == 'NumberOne':
                key = types.InlineKeyboardMarkup()
                but_30 = types.InlineKeyboardButton(arr, callback_data="Number30")
                but_31 = types.InlineKeyboardButton(ab, callback_data="Number31")
                key.add(but_30, but_31)
                bot.send_message(c.message.chat.id, 'Avance o Retroceda', reply_markup=key)

        if c.data == 'Number30':
                os.system ('sudo python /var/www/html/cgi-enabled/motor312.py')
                time.sleep(0.3)
                bot.send_message(c.message.chat.id, 'Siga pulsando para avanzar'+" "+av)
```

Figura 3.140 Función de llamada del menú en línea a los *scripts*

A diferencia de los comandos anteriores, *exec* y *temp* no necesitaron de un menú, en las Figura 3.141 y Figura 3.142 se visualizan las líneas de código de estos. En el caso del comando *exec* se ocupó la función *len()* para obtener la longitud de los caracteres del mensaje. Por otro lado, en los 2 comandos se utilizó la función *os.popen(),* este método se empleó para abrir una tubería al ejecutar los comandos de *Linux* o con el *script*. Esta interferencia permite traer de vuelta la respuesta para presentarla al usuario en *Telegram*.

```
# Ejecuta un comando
@bot.message_handler(commands=['exec'])
def command_long_text(m):
    cid = m.chat.id
    if cid not in knownUsers:
        bot.send_message(m.chat.id, "PERMISO DENEGADO"+adver)
        bot.send_message(m.chat.id, enon)
    else:
        bot.send_message(cid, "Ejecutando: "+m.text[len("/exec"):])
        bot.send_chat_action(cid, 'typing') # show the bot "typing" (max. 5 secs)
        time.sleep(2)
        f = os.popen(m.text[len("/exec"):])
        result = f.read()
        bot.send_message(cid, "Resultado: "+result)
```

Figura 3.141 Función para ejecutar comandos de Linux

```
# Mira temperaturas
@bot.message_handler(commands=['temp'])
def command_long_text(m):
    cid = m.chat.id
    if cid not in knownUsers:
        bot.send_message(m.chat.id, "PERMISO DENEGADO"+adver)
        bot.send_message(m.chat.id, enon)
    else:
        bot.send_message(cid, "Vamos a comprobar si has puesto caliente a tu equipo...")
        bot.send_chat_action(cid, 'typing') # show the bot "typing" (max. 5 secs)
        time.sleep(2)
        f = os.popen("sudo sh /root/aplicaciones/temperaturas.sh")
        result = f.read()
        bot.send_message(cid, ""+result)
```

Figura 3.142 Función para obtener la temperatura de la *Raspberry Pi*

Finalmente, en el *script bot_pihole.py* en su parte final se colocó la línea de código *bot.polling(none_stop=True)*, esta es la encargada de realizar el sondeo largo al ejecutar el *script*.

Ahora bien, hay 3 *scripts* .sh que complementan el funcionamiento del *Bot*. El primero permite conocer la temperatura e información importante de la *Raspberry Pi*, el segundo envía un mensaje al usuario cada vez que el sistema arranca y el tercero indica si alguien se conectó por medio de *SSH*.

- *Script temperaturas.sh*

El *script* temperaturas.sh es un archivo de tipo *bash*. Los archivos *bash* son muy útiles ya que son archivos de texto en los que se puede colocar una secuencia de comandos de *Linux* e imprimir fácilmente sus valores por medio de la función *echo*.

En la Figura 3.143 se nota como se obtiene tanto la temperatura del *CPU (ARM)* como la del *GPU (Raspberry Pi)*. En el caso de la temperatura del *CPU* se extrae del archivo */sys/class/thermal/thermal_zone0/temp* por medio del comando *cat*, más tarde se guardó en la variable *CPU* y se dividió por 1000 para obtener la temperatura en grados centígrados con la finalidad de tener un formato más legible para el usuario [51].

```
echo "Temp.CPU => $((cpu/1000))'C°"
echo "Temp.GPU => $(/opt/vc/bin/vcgencmd measure_temp)"
```

Figura 3.143 Temperaturas del *CPU* y *GPU* de la *Raspberry Pi*

Por otro lado, para la temperatura del *GPU* y demás información que se encuentra en la Figura 3.144 se empleó la función *vcgencmd* junto con los comandos de la Tabla 3.10.

```
echo "$(vcgencmd measure_clock arm) Hz"
echo "$(vcgencmd measure_volts core)"
echo "$(vcgencmd get_config  total_mem)M"
echo "Mem. del Sistema $(vcgencmd get_mem arm)"
echo "Mem. de la $(vcgencmd get_mem gpu)"
```

Figura 3.144 Información de la *Raspberry Pi*

Tabla 3.10 Principales comandos de la función *vcgencmd* [52]

Nombre	Descripción
vcgencmd measure_clock	Devuelve la frecuencia actual del reloj especificado
vcgencmd measure_volts	Muestra los voltajes actuales utilizados dependiendo un bloque específico.
vcgencmd get_config	Muestra el valor de la configuración especificada
vcgencmd get_mem	Informa sobre la cantidad de memoria direccionable
vcgencmd measure_temp	Devuelve la temperatura del *SoC* medida por el sensor de temperatura integrado.

También en el *script* se usó el archivo */proc/meminfo,* este almacena estadísticas sobre el uso de memoria libre y utilizada (tanto física como de intercambio) en el sistema, así como la memoria compartida y los *buffers* utilizados por el *kernel*. [53] Para ver el archivo se utilizó el comando *egrep* tal cual se observa en la Figura 3.145.

```
echo "$(egrep --color 'Mem|Cache|Swap' /proc/meminfo)"
```

Figura 3.145 Impresión de la memoria libre y utilizada

Finalmente, además se consideró significativo que el usuario conozca el espacio disponible de la tarjeta SD, esto se logró a través del comando *df -h* tal cual se ve en la Figura 3.146.

```
echo "$(df -h)"
```

Figura 3.146 Comando para saber el espacio disponible de la tarjeta SD

- *Script reinicio.sh*

Mediante el *script* reinicio.sh el usuario conoce automáticamente si la *Raspberry Pi* ha realizado un arranque del sistema operativo. En la Figura 3.147 se encuentra el *script,* en este se nota que se empleó el *Token* para autentificar el *Bot* y también el *ID* del usuario encargado de controlar el prototipo. Además, se envía el mensaje a la aplicación de *Telegram* a través del método *POST*.

```
 GNU nano 3.2                    reinicio.sh

#!/bin/bash

TOKEN="1568847307:AAH1udvgt7S1WlzJD92JQtp_fDLd1O-xvEg"
ID="1246757745"
MENSAJE="La raspberry  se acaba de encender"
URL="https://api.telegram.org/bot$TOKEN/sendMessage"

curl -s -X POST $URL -d chat_id=$ID -d text="$MENSAJE"
```

Figura 3.147 *Script* reinicio.sh

Para programar la tarea del *script* reinicio.sh se ocupó el *daemon Cron.* Con este propósito, primeramente, se ingresó al archivo *crontab* digitando el comando *sudo crontab -e* en la terminal. Después en la parte inferior del archivo se añadió la siguiente línea de código: *@reboot (sleep 30 ; sh /home/pi/aplicaciones/reinicio.sh) >/dev/null 2>&1*, tal cual se visualiza en la Figura 3.148.

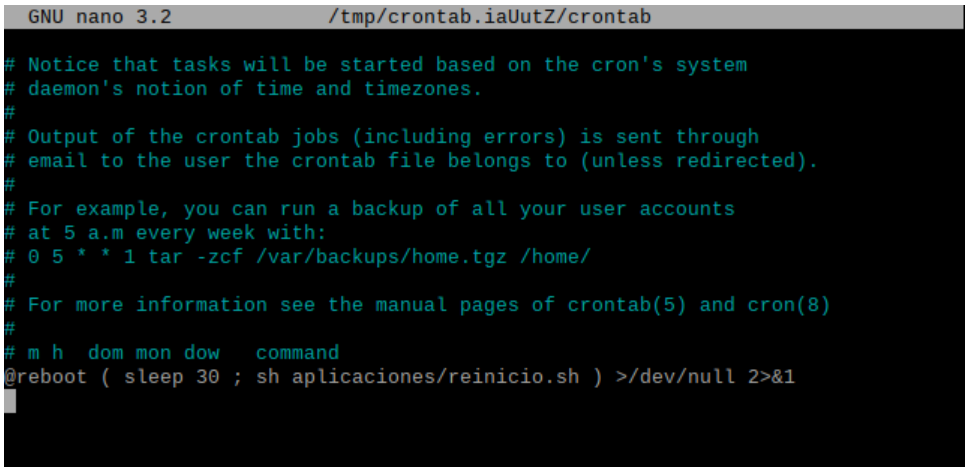


Figura 3.148 Programación del *script* reinicio.sh en el archivo *crontab*

Donde:

- *@reboot*: Permite efectuar una tarea al arrancar el sistema. En la Tabla 3.11 se encuentran los principales comandos del archivo *crontab* con su sintaxis para definir actividades en un tiempo determinado.
- *sleep 30*: Tiempo de espera para ejecutar el comando. Este tiempo es importante para que el sistema operativo pueda cargar todo lo necesario para que el comando se ejecute bien.
- *sh /home/pi/aplicaciones/reinicio.sh:* Comando que ejecuta el *script*.
- *>/dev/null 2>&1*: Para que el *script* trabaje en segundo plano y no interrumpa al usuario.

Tabla 3.11 Programar tareas en *crontab* [54]

Sintaxis	Palabras especiales y su equivalencia
minuto (0-60) hora (0-23) día del mes (1-31) mes del año (1-12) día de la semana (0-7)	*@hourly* → 0 * * * * *@daily* → 0 0 * * * *@mounthly* → 0 0 1 * * *@yearly* → 0 0 1 1 * *@reboot* → cuando arranca el sistema

- *Shell script*

El tener *SSH* activado en el prototipo podría presentar un riesgo. Ya que si alguien averiguará la contraseña de *SSH* podría hacer lo que quisiera con el equipo. Por este motivo, se necesitó que la *Raspberry Pi* envíe un mensaje al *Bot* cada vez que se efectúa una conexión remota por *SSH*.

A diferencia de la tarea anterior que era programada en *Cron*, la actividad de detectar si alguien se conectó a la *Raspberry Pi* por *SSH* es un evento que puede ocurrir de manera repentina e inesperada. La solución más apropiada a la que se llegó fue emplear los archivos *shell scripts*, estos son ficheros que el sistema operativo ejecuta de forma automática cuando se da una cierta condición. En la Tabla 3.12 se encuentran algunos de estos archivos que se consideraron para realizar esta tarea.

Tabla 3.12 Archivos *Shell scripts* [55]

Para todos los usuarios (Se necesita permisos de root para editar/modificar estos archivos)	**Para un usuario específico**
/etc/profile → Se ejecuta cuando cualquier usuario inicia la sesión.	*~/.bash_profile* → Se ejecuta el *.bash_profile* cuando un usuario específico inicia su sesión
/etc/bashrc → Se ejecuta cada vez que cualquier usuario ejecuta el programa *bash*	*~/.bashrc* → Se ejecuta el *.bashrc* cuando un usuario específico ejecuta el programa *bash*

Inicialmente se empleó el fichero *bashrc,* pero se descartó inmediatamente, ya que el mensaje no solo se enviaba cuando se conectaba el usuario remotamente, sino que cada vez que la terminal se abría, lo que llegaba a ser muy molesto. Además, también se deseaba conocer el ingreso por *SSH* tanto de un usuario específico como del usuario *root* por lo que el archivo utilizado finalmente fue el *profile*.

Seguidamente, se ingresó al archivo *profile* como usuario *root* tal como se muestra en la Figura 3.149.

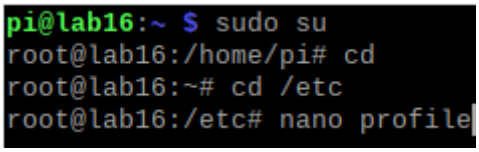


Figura 3.149 Ingreso al archivo *profile*

Una vez dentro, en la parte final de este archivo se colocó las siguientes líneas de código:

curl -s -X POST

https://api.telegram.org/bot"1568847307:AAH1udvgt7S1WlzJD92JQtp_fDLd1O-xvEg"/sendMessage -d chat_id="1246757745" -d text="⚠ Alguien se acaba de conectar a la Raspberry por ssh con la IP $(echo $SSH_CLIENT | awk '{ print $1}'). Se ingresó como usuario $(echo $USER) ¿Serás tú? ">/dev/null 2>&1

En estas líneas se puede apreciar cómo se ingresó la dirección del *Bot* de *Telegram* y el *ID* del usuario que va a recibir el mensaje a través del método *POST*. El mensaje que recibe el usuario del *Bot* le indica que ha existido una conexión remota, además en este se añade la dirección IP de la máquina que accedió a la conexión con su usuario.

Esto se realizó imprimiendo la función *SSH_CLIENT*, esta devuelve varios parámetros juntos como: la dirección IP, el número de puerto, etc. De manera que solo para imprimir la dirección IP se utilizó el comando *awk*. Este comando en específico proporciona un lenguaje de *scripting* para el procesamiento de texto. Con el lenguaje de *scripting awk* se puede: definir variables, generar reportes con formato, entre otros. [56].

Por otra parte, para devolver el nombre del usuario que ingresó remotamente se ocupó la función *USER*. Esta función devuelve el nombre de usuario actual.

Scripts para el control de la ventilación del prototipo

Con el propósito de mantener el prototipo en óptimas condiciones y darle mayor vida útil, se agregó la ventilación ya que este permanecerá encendido la mayor parte del tiempo.

El *script* venti.py es el encargado de esta función, fue elaborado a través del diagrama de flujo que se muestra en la Figura 3.150. Al principio se pensó en solo activar y desactivar la ventilación, pero al trabajar el ventilador a su máxima capacidad siempre, llegaba a ser muy ruidoso.

Por lo cual, además de activarse de manera automática cada vez que se enciende el prototipo, la velocidad del ventilador varía a través de *PWM*, dependiendo de la temperatura de la *Raspberry Pi*. Para la obtención de la temperatura se empleó la misma lógica del *script* temperaturas.sh, específicamente la sección de código de la Figura 3.143.

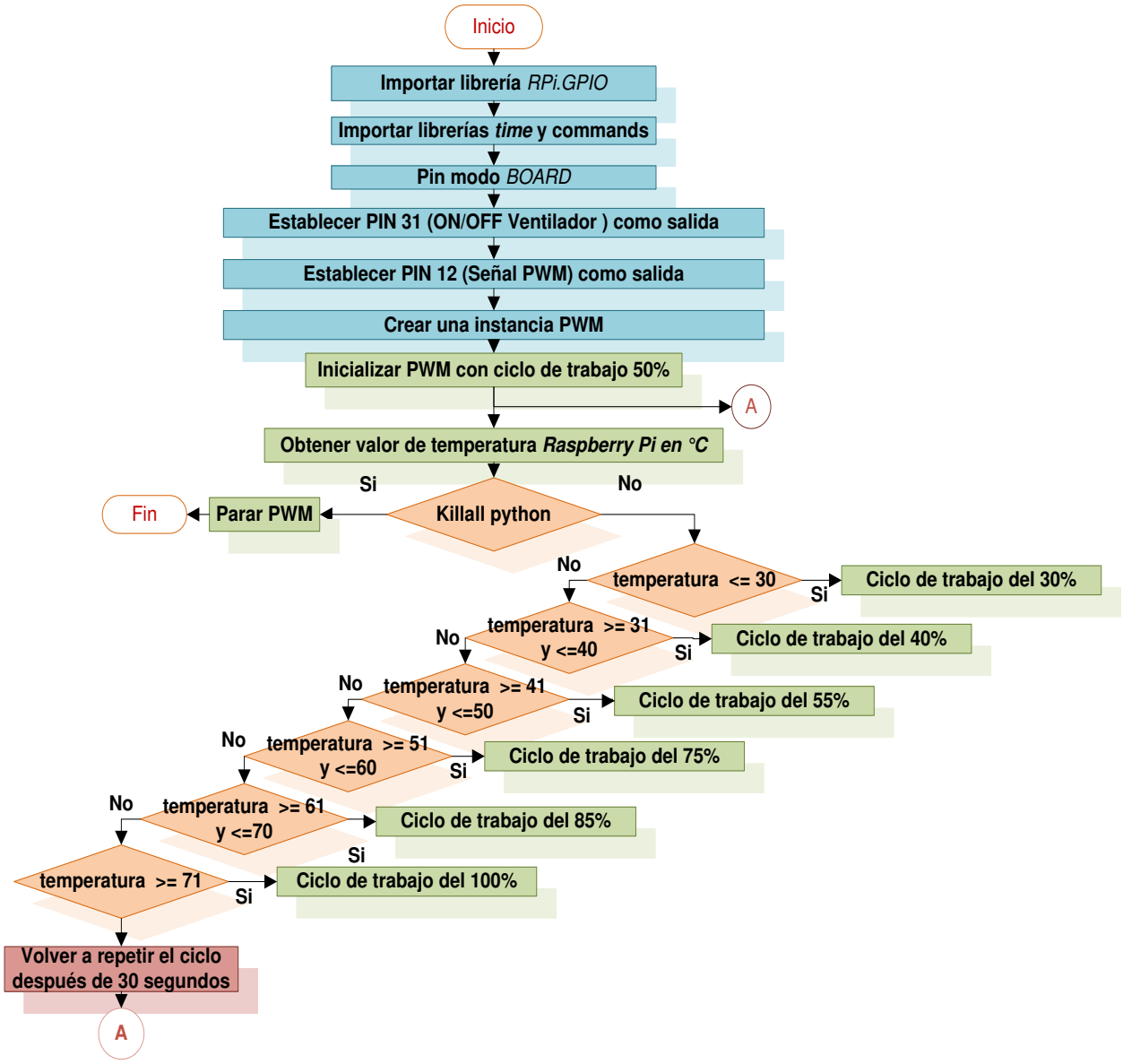


Figura 3.150 Diagrama de flujo para la ventilación del prototipo

Por otra parte, para el control *PWM* primeramente se debió tener en consideración que la *Raspberry Pi* posee solo dos canales *PWM* disponibles y que estos canales solo se pueden asignar a ciertos pines (*PWM0* en *GPIO12* y *GPIO18*, *PWM1* en *GPIO13* y *GPIO19*). En el caso del *script* la señal *PWM*, que se ocupó fue la *PWM0* a través del *GPIO18* (Pin12). La biblioteca *RPi.GPIO* proporciona comandos específicos que se pueden utilizar en el control *PWM*, estos se encuentran en la Tabla 3.13.

Tabla 3.13 Comandos para el control *PWM* [18]

Comandos	Sintaxis	Significados
pwm = GPIO.PWM(channel, frequency)	canal: Pin de la señal *PWM* frecuencia: Frecuencia inicial de la señal *PWM*, en (Hz)	Crea una instancia de la señal *PWM*
pwm.start(dc)	dc: Ciclo de trabajo inicial de la señal *PWM* (0 -100)	Habilita la señal *PWM*
pwm.ChangeFrequency(freq)	freq: Nueva frecuencia de la señal *PWM* en (Hz)	Cambia la frecuencia de la señal *PWM*
pwm.ChangeDutyCycle(dc)	dc: Nuevo ciclo de trabajo de *PWM* (0-100)	Cambia el ciclo de trabajo de la señal *PWM*

La frecuencia de trabajo se definió en 1000 (Hz) tal cual se nota en la Figura 3.151, los valores de frecuencia deben estar entre 1 y 2000.

```
#Salida PWM
GPIO.setup(12, GPIO.OUT)
pwm = GPIO.PWM(12, 1000)
```

Figura 3.151 Definición de la instancia *PWM* en el *script* venti.py

Una vez definida la frecuencia de trabajo, se estableció que la señal inicie con un ciclo del 50% como se observa en la Figura 3.152.

```
#Iniciar PWM
pwm.start(50)
```

Figura 3.152 Inicio de la señal *PWM* con un ciclo del 50%

Más tarde, se emplearon las Ecuación 3.5 y Ecuación 3.6 junto con la Figura 3.153 para obtener el tiempo que la señal permanece en alto y bajo además del voltaje que se obtendrá a la salida.

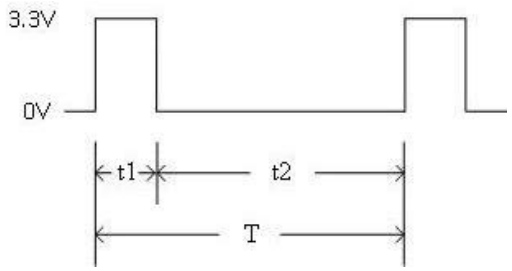


Figura 3.153 Ciclo de trabajo de la señal *PWM* [18]

$$\text{Ciclo de Trabajo} = \frac{t1}{\text{T}} = \frac{t1}{\text{t1} + \text{t2}}$$

Ecuación 3.5 Fórmula del ciclo de trabajo de la señal *PWM* [18]

Donde:

$Ciclo\ de\ Trabajo$: 50 %

T : $\frac{1}{1000} = 1000$ (µs) Periodo de la señal *PWM*

$t1$: Tiempo en parte positiva

$t2$: Tiempo en parte negativa

Usando la Ecuación 3.5 se obtiene:

$$\text{t1 y t2} = 500\ (\mu\text{s})$$

Por lo tanto, al trabajar el ventilador en un ciclo del 50% este permanecerá la misma cantidad de tiempo tanto en la parte positiva como en la negativa.

$$\text{Vo} = (\text{Vmax} - \text{Vmin}) * Ciclo\ de\ Trabajo$$

Ecuación 3.6 Fórmula del voltaje de salida [18]

Donde:

Vo	:	Voltaje de salida (V_{DC})
$Vmax$	:	3,3 (V_{DC})
$Vmin$	:	0 (V_{DC})
$Ciclo\ de\ trabajo$	:	50 %

Empleando la Ecuación 3.6 se obtiene:

$$\mathrm{Vo} = 1{,}65\ (V_{DC})$$

Al empezar a funcionar la ventilación del prototipo el voltaje que le llegará al ventilador será $1{,}65\ (V_{DC})$.

Finalmente, cada vez que la *Raspberry Pi* eleve su temperatura, el ciclo de trabajo también aumentará, lo que hará que llegue mayor voltaje al ventilador y este girará más rápido de manera que la temperatura disminuirá automáticamente. En la Figura 3.154 se puede apreciar cómo se colocó la variación del ciclo de trabajo de la señal *PWM* en el *script* según la temperatura de la *Raspberry Pi*, por otro lado, en la Figura 3.155 se destaca la señal *PWM* trabajando en diferentes ciclos con su respectiva salida de voltaje.

```
while True:
    if get_cpu_temp() <= 30:
            pwm.ChangeDutyCycle(30)
    elif get_cpu_temp() >= 31 and get_cpu_temp() <= 40:
            pwm.ChangeDutyCycle(40)
    elif get_cpu_temp() >= 41 and get_cpu_temp() <= 50:
            pwm.ChangeDutyCycle(55)
    elif get_cpu_temp() >= 51 and get_cpu_temp() <= 60:
            pwm.ChangeDutyCycle(75)
    elif get_cpu_temp() >= 61 and get_cpu_temp() <= 70:
            pwm.ChangeDutyCycle(85)
    elif get_cpu_temp() >= 71:
            pwm.ChangeDutyCycle(100)
    time.sleep(30)
```

Figura 3.154 Ciclo de trabajo según la temperatura de la *Raspberry Pi*

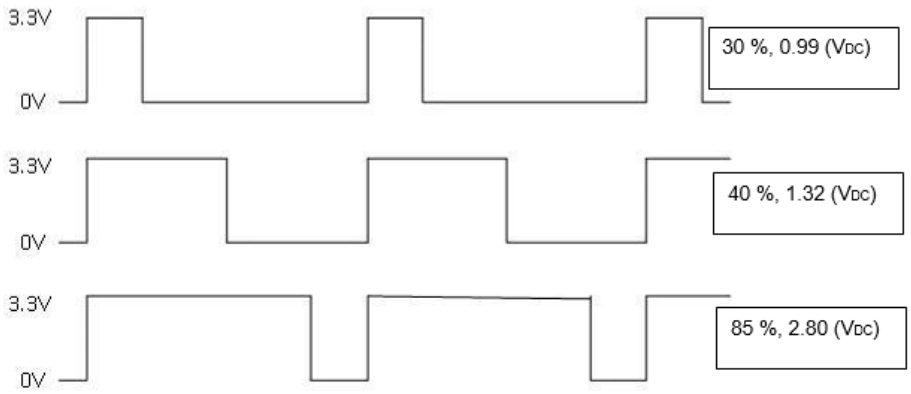


Figura 3.155 Señales *PWM* que llegan al ventilador según su ciclo de trabajo [18]

Servicio pantalla en modo quiosco

- *Script para lanzar la pantalla en modo quiosco*

Primeramente, se instaló el paquete *sed* en la *Raspberry Pi* con el comando *sudo apt-get install sed*. Luego se creó el archivo kiosk.sh digitando el comando *nano /home/pi/kiosk.sh* en la terminal de *Linux*. Más tarde, se ingresó en el archivo las líneas de comando de la Figura 3.156.

```
  GNU nano 3.2                          /home/pi/kiosk.sh

#!/bin/bash
xset s noblank
xset s off
xset -dpms


sed -i 's/"exited_cleanly":false/"exited_cleanly":true/' /home/pi/.config/chromium/Default/Preferences
sed -i 's/"exit_type":"Crashed"/"exit_type":"Normal"/' /home/pi/.config/chromium/Default/Preferences

/usr/bin/chromium-browser --noerrdialogs --disable-infobars --kiosk localhost:8080 &
```

Figura 3.156 *Script para lanzar la pantalla en modo quiosco*

Las líneas de código *xset s noblank, xset s off y xset -dpms* ayudan a evitar que el sistema de administración de energía de la pantalla de la *Raspberry Pi* se active y apague la pantalla. Por otra parte, en las dos líneas que siguen a continuación se usa *sed* para buscar el archivo de preferencias de *Chrome* y borrar los indicadores que harían aparecer la barra de advertencia, lo cual evita un comportamiento que realmente no se desea que ocurra en la *Raspberry Pi.*

Finalmente, con la línea de *código /usr/bin/chromium-browser --noerrdialogs --disable-infobars --kiosk "dirección del prototipo" &*, lo que hace es configurar *Chrome* para operar en modo quiosco, de manera que solo se permite el acceso limitado tanto al navegador web como a cualquier otra funcionalidad del sistema operativo. Por otra parte, *noerrdialogs* impide mostrar cuadros de diálogo de error al usuario final. En esta línea se debe colocar la dirección que se desea que se muestre al ejecutar el *script.*

- *Configuración del quiosco de Raspberry Pi para que se inicie en el arranque*

Antes de nada, se utilizó el comando *eco $ DISPLAY* para calcular el valor de visualización actual, justo como se observa en la Figura 3.157.

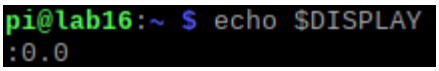


Figura 3.157 Comando para saber el valor de visualización actual

Este valor se usa para que el sistema operativo sepa en qué pantalla mostrar el quiosco de *Chrome*, sin él, el quiosco no se cargará o se cargará en la pantalla incorrecta si se tienen 2 pantallas conectadas a la *Raspberry Pi.*

Seguidamente, para que el quiosco se inicie en el arranque se creó un servicio con el comando: *sudo nano /lib/systemd/system/kiosk.service.* Dentro de este archivo, se ingresó las líneas de texto de la Figura 3.158.

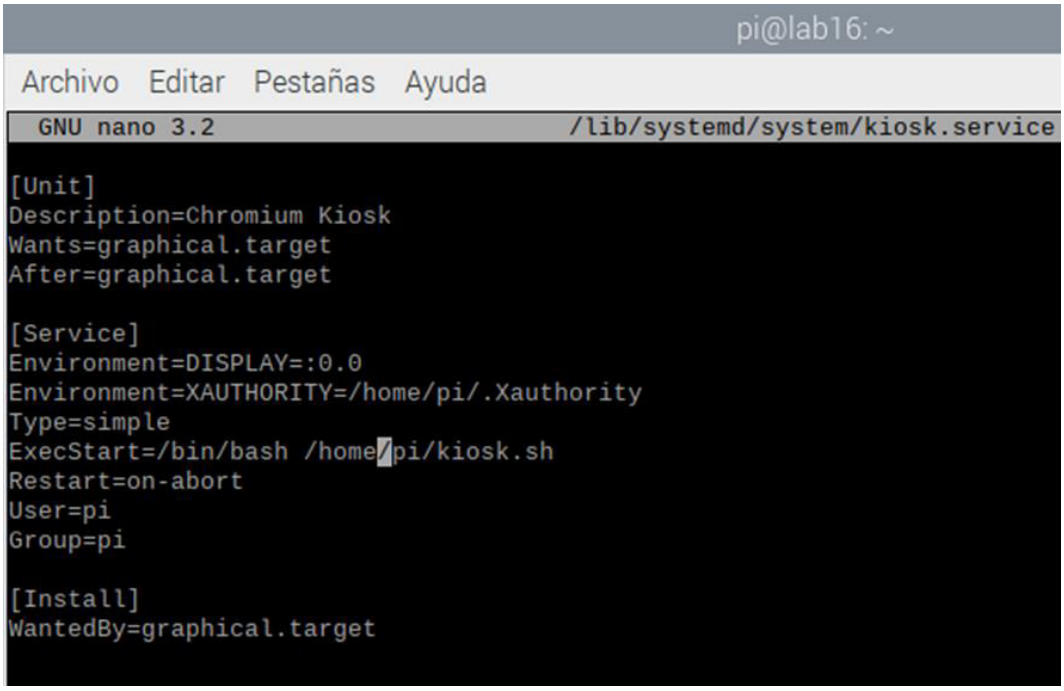


Figura 3.158 Servicio para lanzar la pantalla en modo quiosco

Por otro lado, se debe habilitar el servicio, para que la pantalla en modo quiosco se inicie en el arranque automáticamente con el comando *sudo systemctl enable kiosk.service,* en cambio ya con el servicio habilitado, se puede elegir reiniciar la *Raspberry Pi* o iniciar el servicio inmediatamente ejecutando el comando *sudo systemctl start kiosk.service.* También se puede verificar el estado del servicio con el comando *sudo systemctl status kiosk.service.*

Si se desea detener la ejecución del servicio se debe utilizar el comando *sudo systemctl stop kiosk.service*. Este comando detendrá la ejecución del *script* kiosk.sh y cerrará el navegador *Chrome* abierto. Por último, si alguna vez se necesita deshabilitar el servicio, emplear el comando: *sudo systemctl disable kiosk.service.*

Lanzamiento automático del *script* del *Bot* de *Telegram* y otros *scripts*

Para el lanzamiento automático al iniciar el arranque del sistema operativo de los *scripts* del *Bot* de *Telegram*, la ventilación del prototipo y del led que indica que el prototipo se encendió se empleó el archivo *rc.local.* Para esto en primer lugar, se ingresó al directorio *etc* con el comando *cd /etc* y se digitó en la terminal el comando *sudo nano rc.local.*

Una vez en el archivo se colocó los *scripts* tal cual se muestra en la Figura 3.159. Es importante que los *scripts* se coloquen antes de la línea de código "*exit 0".*

```
  GNU nano 3.2                                          rc.local

#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

sleep   30
setsid /usr/bin/python  /var/www/html/cgi-enabled/telegramapi.py &
setsid /usr/bin/python  /var/www/html/cgi-enabled/venti.py &
setsid /usr/bin/python  /var/www/html/cgi-enabled/led.py &

exit 0
```

Figura 3.159 Lanzamiento automático con rc.local

Por otra parte, en el caso del *Bot* de *Telegram* como se tenía problemas porque el *script* se caía debido al método *Polling*, lo que se hizo fue calcular el tiempo en que el *Bot* permanecía activo y con este tiempo a través del *script* de la Figura 3.160 volver a habilitar el servicio del *Bot* de *Telegram.*

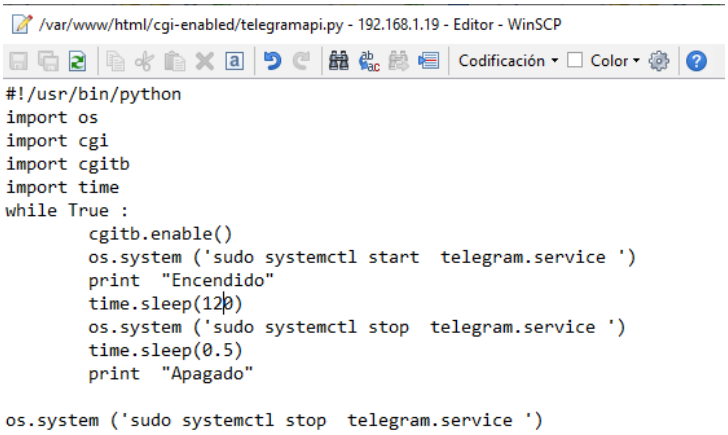


Figura 3.160 *Script* que lanza periódicamente el *Bot* de *Telegram*

Diseño de los circuitos del prototipo

En primer lugar, se estableció el consumo de corriente de los diferentes componentes. Dado que los elementos requieren de distintos voltajes se los organizó según la fuente de alimentación que ocupan. En la Tabla 3.14 se puede observar el consumo de corriente y voltaje de los dispositivos que utilizarán la fuente de alimentación 5 (V_{DC}) 2 (A_{DC}).

Tabla 3.14 Consumo de la fuente de alimentación 5 (V_{DC}) 2 (A_{DC})

Cantidad	Elemento	Consumo de voltaje	Consumo de corriente	TOTAL Consumo de corriente
8	Módulo relé	5 (V_{DC})	90 (mA_{DC})	0,72 (A_{DC})
1	Ventilador de PC	5 (V_{DC}) – 12 (V_{DC})	0,45 (A_{DC})	0,45 (A_{DC})
1	Receptor Infrarrojo (KY-022)	2,7 (V_{DC}) – 5,5 (V_{DC})	1,5 (mA_{DC})	1,5 (mA_{DC})
1	Sensor *PIR* (HC-SR501)	4,5 (V_{DC}) – 20 (V_{DC})	60 (uA_{DC})	60 (uA_{DC})
1	Sensor Ultrasónico (HC-SR04)	5 (V_{DC})	15 (mA_{DC})	15 (mA_{DC})
TOTAL Corriente				**1,18 (A_{DC})**

A continuación, se obtuvo el consumo de potencia mediante la Ecuación 3.7

$$P = V.I$$

Ecuación 3.7 Fórmula de la Potencia [57]

Donde:

I : 1,18 (A_{DC}) corriente

V : 5 (V_{DC}) voltaje

P : (W) potencia

Usando la Ecuación 3.7 se obtiene:

$$P = 5{,}9\ (W)$$

Luego, se procedió a colocar los elementos de la Tabla 3.14 en el programa *Fritzing* tal como se muestra en la Figura 3.161 con el propósito de tener una guía al momento de armar los circuitos del prototipo.

Dado que los componentes de la Tabla 3.14 se manejan a 5 (V_{DC}) y los pines *GPIO* de la *Raspberry Pi* no son tolerantes a esta tensión pues están diseñados para 3,3 (V_{DC}) se empleó un conversor de nivel lógico.

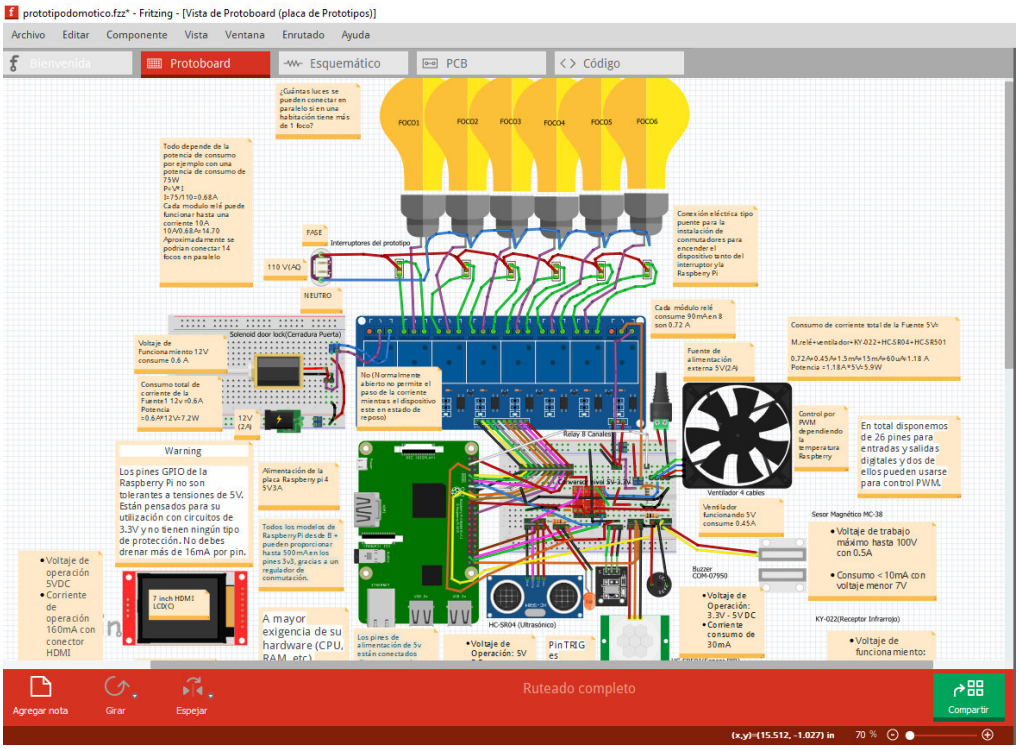


Figura 3.161 Esquemático del prototipo domótico

Por otra parte, en la Tabla 3.15 se destaca el consumo de corriente y voltaje de los motores a pasos que están conectados a la fuente de alimentación 12 (V_{DC}) 2 (A_{DC}) número 1. Además, de la potencia que consumen.

Tabla 3.15 Consumo de la fuente de alimentación 12 (V_{DC}) 2 (A_{DC}) número 1

Cantidad	Elemento	Consumo de voltaje	Consumo de corriente	TOTAL Consumo de corriente
2	Nema17	12 (V_{DC})	0,8 (A_{DC})	1,6 (A_{DC})
TOTAL Corriente				**1,6 (A_{DC})**

Donde:

I : 1,6 (A_{DC}) corriente
V : 12 (V_{DC}) voltaje
P : (W) potencia

Empleando la Ecuación 3.7 se obtiene:

$$P = 19{,}2\ (W)$$

Posteriormente, en el diseño del circuito del motor a pasos fue necesario calcular la intensidad que proporcionará el controlador A4988 al motor. Para esto se tuvo en cuenta que la intensidad se regula a través del potenciómetro Rs que trae el A4988 y esta cambia dependiendo del integrado que se utilice, por lo que las fórmulas varían tal como se indica en la Figura 3.162.

Modelo	Rs	Fórmula reducida
A4988	50	I_max = 0,625 * V_ref
A4988	100	I_max = 1,25 * V_ref
A4988	200	I_max = 2 ,2* V_ref

Figura 3.162 Fórmulas según el valor Rs del integrado A4988 [58]

Como se adquirió el integrado A4988 con un valor de Rs igual a 100 (Ω) se ocupó la fórmula de la Ecuación 3.8.

$$I_{max} = 1{,}25\ .\ V_{ref}$$

Ecuación 3.8 Fórmula de la corriente máxima que entrega el integrado A4988 al motor a pasos [58]

Entonces:

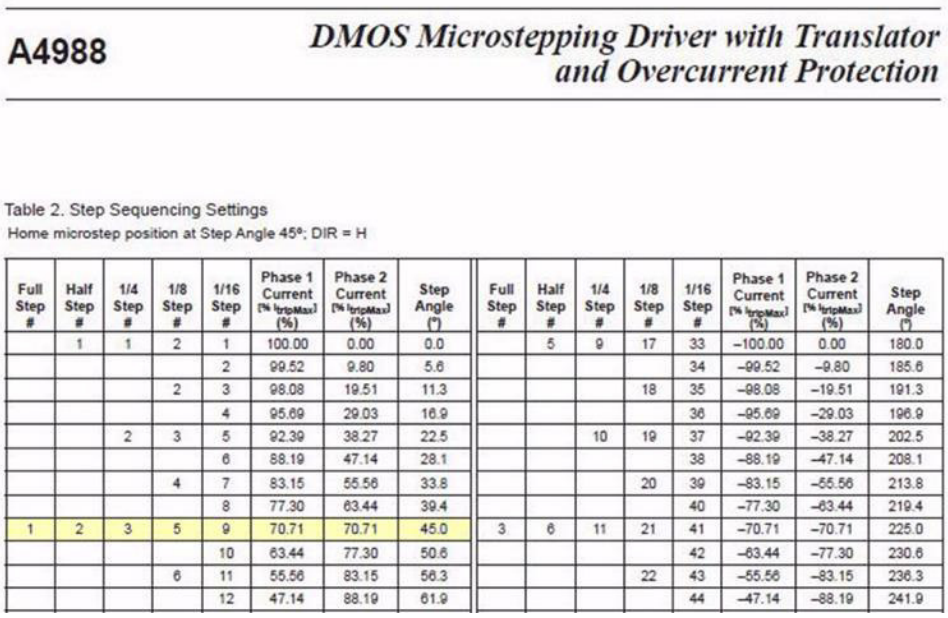
I_{max} : 0,8 (A_{DC}) corriente

V_{ref} : (V_{DC}) voltaje

Por lo tanto:

$$V_{ref} = 0{,}64\ (V_{DC})$$

El valor calculado es cuando el integrado trabaja con medios pasos, tal como se visualiza en la Figura 3.163.



A4988 *DMOS Microstepping Driver with Translator and Overcurrent Protection*

Table 2. Step Sequencing Settings

Home microstep position at Step Angle 45°; DIR = H

Full Step #	Half Step #	1/4 Step #	1/8 Step #	1/16 Step #	Phase 1 Current (% $I_{TripMax}$) (%)	Phase 2 Current (% $I_{TripMax}$) (%)	Step Angle (°)	Full Step #	Half Step #	1/4 Step #	1/8 Step #	1/16 Step #	Phase 1 Current (% $I_{TripMax}$) (%)	Phase 2 Current (% $I_{TripMax}$) (%)	Step Angle (°)
	1	1	2	1	100.00	0.00	0.0		5	9	17	33	–100.00	0.00	180.0
				2	99.52	9.80	5.6					34	–99.52	–9.80	185.6
			2	3	98.08	19.51	11.3				18	35	–98.08	–19.51	191.3
				4	95.69	29.03	16.9					36	–95.69	–29.03	196.9
		2	3	5	92.39	38.27	22.5			10	19	37	–92.39	–38.27	202.5
				6	88.19	47.14	28.1					38	–88.19	–47.14	208.1
			4	7	83.15	55.56	33.8				20	39	–83.15	–55.56	213.8
				8	77.30	63.44	39.4					40	–77.30	–63.44	219.4
1	2	3	5	9	70.71	70.71	45.0	3	6	11	21	41	–70.71	–70.71	225.0
				10	63.44	77.30	50.6					42	–63.44	–77.30	230.6
			6	11	55.56	83.15	56.3				22	43	–55.56	–83.15	236.3
				12	47.14	88.19	61.9					44	–47.14	–88.19	241.9

Figura 3.163 *Datasheet* del integrado A4988 [59]

En cambio, cuando se trabaja a paso completo la corriente aplicada es siempre del 71%, por lo que al configurar el integrado con el multímetro se empleó el valor de 0,454 (V_{DC}).

Más tarde, con la información obtenida se estableció el diagrama de la Figura 3.164, cabe aclarar que solo se ocupó una *Raspberry Pi,* esta es la misma de la Figura 3.161.

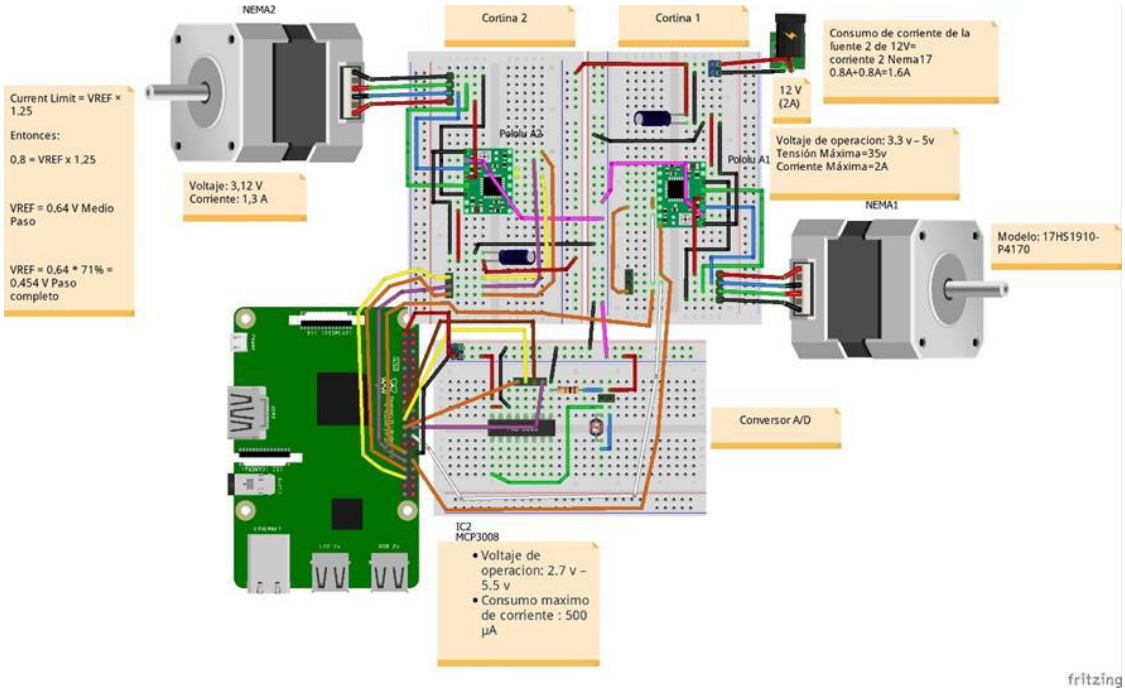


Figura 3.164 Esquemático del sistema de las cortinas

Por otro lado, en la Tabla 3.16 se puede ver el consumo de corriente y voltaje de la cerradura eléctrica con solenoide que funciona con la fuente de alimentación 12 (V_{DC}) 2 (A_{DC}) número 2. Asimismo, se obtuvo el consumo de potencia del dispositivo

Tabla 3.16 Consumo de la fuente de alimentación 12 (V_{DC}) 2 (A_{DC}) número 2

Cantidad	Elemento	Consumo de voltaje	Consumo de corriente	TOTAL Consumo de corriente
1	Cerradura eléctrica con Solenoide	12 (V_{DC})	0,6 (A_{DC})	0,6 (A_{DC})
TOTAL Corriente				**0,6 (A_{DC})**

Donde:

I : 0,6 (A_{DC}) corriente
V : 12 (V_{DC}) voltaje
P : (W) potencia

Por medio de la Ecuación 3.7 se obtiene:

$$P = 7{,}2\ (W)$$

Del mismo modo, se realizaron las Tabla 3.17 y Tabla 3.18 para los componentes que utilizan las fuentes internas de la *Raspberry Pi* y se calculó su potencia de consumo.

Tabla 3.17 Consumo de la fuente interna de la *Raspberry Pi* 5 (V_{DC})

Cantidad	Elemento	Consumo de voltaje	Consumo de corriente	TOTAL Consumo de corriente
2	Ventiladores de la carcasa	5 (V_{DC})	0,2 (A_{DC})	0,4 (A_{DC})
1	Pantalla *7inch HDMI LCD (C)*	5 (V_{DC})	160 (mA_{DC})	160 (mA_{DC})
TOTAL Corriente				**0,56 (A_{DC})**

Tabla 3.18 Consumo de la fuente interna de la *Raspberry Pi* 3,3 (V_{DC})

Cantidad	Elemento	Consumo de voltaje	Consumo de corriente	TOTAL Consumo de corriente
1	Conversor A/D (MCP3008)	2,7 (V_{DC}) – 5,5 (V_{DC})	500 (uA_{DC})	500 (uA_{DC})
2	Controlador de motor a pasos (A4988)	3,3 (V_{DC}) – 5 (V_{DC})	15 (mA_{DC})	30 (mA_{DC})
1	Sensor magnético (MC-38)	3,3 (V_{DC}) – 5 (V_{DC})	10 (mA_{DC})	10 (mA_{DC})
TOTAL Corriente				**40,5 (mA_{DC})**

La fuente de alimentación de la *Raspberry Pi* que se utilizó es de 5 (V_{DC}) 3 (A_{DC}). Por lo que los pines de la fuente de alimentación interna de 5 (V_{DC}) son capaces de suministrar la corriente del adaptador, menos la utilizada por la propia *Raspberry Pi* que según la documentación oficial de *Raspberry*, el modelo *Raspberry Pi 4* usa 0,6 (A_{DC}) en reposo, 0,85 (A_{DC}) reproduciendo vídeo, y hasta 1,25 (A_{DC}) en estrés máximo.

En cambio, la fuente interna de 3,3 (V_{DC}) puede proporcionar hasta 500 (mA). Teniendo en cuenta lo mencionado y que por cada pin de la *Raspberry Pi* se puede drenar 16 (mA_{DC}) se planteó las siguientes ecuaciones:

$$\mathrm{I}_{Raspberry} = \mathrm{I}_{pines\ ocupados} + \mathrm{I}_{consumo}$$

Ecuación 3.9 Fórmula del consumo de corriente de la *Raspberry Pi*

Entonces:

$\mathrm{I}_{pines\ ocupados}$: 26 pines de 16 (mA_{DC}) = 0,4 (A_{DC})

$\mathrm{I}_{consumo}$: 1,25 (A_{DC}) en estrés máximo

$\mathrm{I}_{Raspberry}$: (A_{DC}) corriente

Por lo tanto, usando la Ecuación 3.9:

$$\mathrm{I}_{Raspberry} = 1{,}66\ (A_{DC})$$

$$\mathrm{I}_{consumofuentePi} = \mathrm{I}_{consumofuente\ interna\ 5\ (v)} + \mathrm{I}_{consumofuenteinterna\ 3.3(v)} + \mathrm{I}_{Raspberry}$$

Ecuación 3.10 Fórmula del consumo de corriente de la fuente de la *Raspberry Pi*

Donde:

$\mathrm{I}_{consumofuente\ interna\ 5\ (v)}$: 0,56 (A_{DC})

$\mathrm{I}_{consumofuenteinterna\ 3.3(v)}$: 40,5 (mA_{DC})

$\mathrm{I}_{Raspberry}$: 1,66 (A_{DC})

$\mathrm{I}_{consumofuentePi}$: (A_{DC}) corriente

Ocupando la Ecuación 3.10 se obtiene:

$$\mathrm{I}_{consumofuentePi} = 2{,}26\ (A_{DC})$$

I : 2,26 (A_{DC}) corriente

V : 5 (V_{DC}) voltaje

P : (W) potencia

Empleando la Ecuación 3.7 se obtiene:

$$\mathrm{P} = 11{,}3\ (\mathrm{W})$$

En base al valor calculado, se obtiene que la potencia de consumo de la fuente *Raspberry Pi* es de 11,3 (W).

Seguidamente, como parte del diseño se calculó la potencia total del prototipo domótico para esto se sumó las potencias de las fuentes de alimentación dando como resultado que el proyecto consumirá 43,6 (W).Tomar en cuenta que esta potencia se generará solo si todos los aparatos trabajan juntos a la vez.

Finalmente, si se mantiene el prototipo trabajando a su capacidad máxima en un año habrá consumido 381,93 (kWh) y estableciendo el precio de un 1 (kWh) como referencia a un valor de 0,08 centavos. El precio de tener el prototipo funcionando a su capacidad máxima todo el año será de 30,55$.

Armado de los circuitos en *Protoboard*

Después, del diseño del esquemático se armaron los circuitos en *Protoboard*, tal como se ilustra en la Figura 3.165.

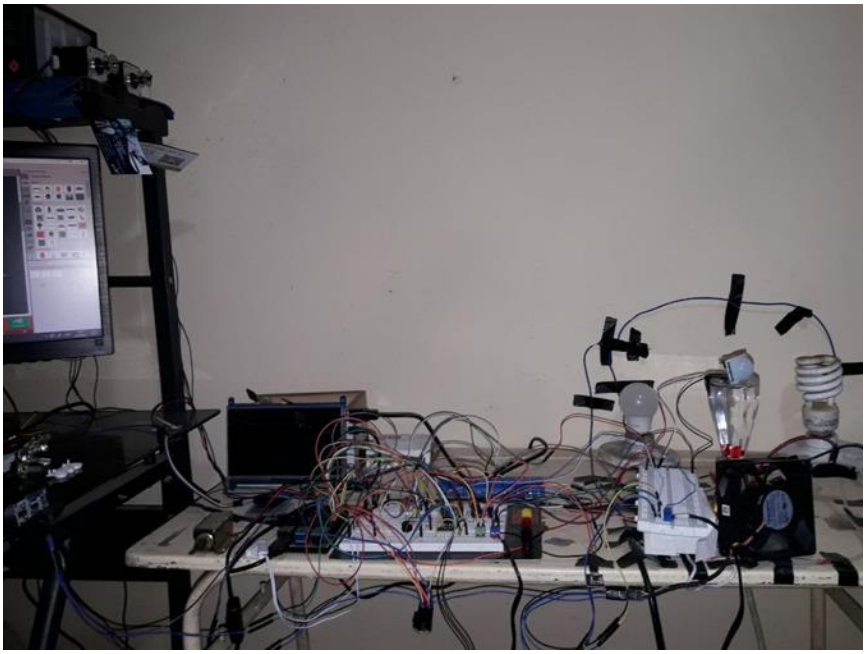


Figura 3.165 Armado de los circuitos en *Protoboard*

Conexión de la pantalla táctil y configuración

Inicialmente, para la conexión de la pantalla táctil se ocupó el diagrama de la Figura 3.166.

Figura 3.166 Esquema de conexión de la pantalla *7inch HDMI LCD(C)*

Más adelante, para la configuración del *touch* se descargó la configuración de las pantallas *Waveshare* en la *Raspberry Pi* del enlace https://github.com/goodtft/LCD-show. A continuación, se descomprimió la carpeta descargada con el comando *unzip* tal como se destaca en la Figura 3.167.

```
pi@raspberrypi:~/Downloads $ unzip LCD-show-master.zip
```

Figura 3.167 Descompresión de la carpeta de configuración de la pantalla

Después, se ingresó a la carpeta justo como se aprecia en la Figura 3.168, seguidamente, en la terminal de *Linux* se ejecutó la configuración de la pantalla por medio del comando *sudo ./LCD7C-show,* tal cual se observa en la Figura 3.169.

```
pi@raspberrypi:~/Downloads/LCD-show-master $ ls
boot             MHS40C-show
etc              MHS40-show
LCD101H-show     MIS35-show
LCD24-3A+-show   MPI3508_480_320-show
LCD24-show       MPI3508-show
LCD28-show       MPI3510-show
LCD32-show       MPI4008-show
LCD35-show       MPI5001-show
LCD55-show       MPI5094-show
LCD5-show        README.md
LCD7B-show       rotate.sh
LCD7C-show       system_backup.sh
LCD7H-show       system_restore.sh
LCD7S-show       usr
LCD-hdmi         xinput-calibrator_0.7.5-1_armhf.deb
MHS24-show       xserver-xorg-input-evdev_1%3a2.10.3-1_armhf.deb
MHS32-show       xserver-xorg-input-evdev_1%3a2.10.5-1_armhf.deb
MHS35B-show      xserver-xorg-input-evdev_1%3a2.10.6-1+b1_armhf.deb
MHS35-show       xserver-xorg-input-evdev_2.10.5-1_armhf.deb
```

Figura 3.168 Ingreso a la carpeta LCD-show-master

```
pi@raspberrypi:~/Downloads/LCD-show-master $ sudo ./LCD7C-show
```

Figura 3.169 Ejecución de la configuración de la pantalla

Por último, se reinició la *Raspberry Pi* para que los cambios se guarden. En la Figura 3.170 se puede ver la pantalla *7inch HDMI LCD(C)* configurada.

Figura 3.170 *Touch* de la pantalla *7inch HDMI LCD(C)* configurado

Diseño de las placas electrónicas

Primeramente, en el programa *Fritzing* en la sección PCB se desarrolló las placas, en la capa superior se colocó la identificación de los componentes, en cambio en la capa inferior se ubicó las pistas conductoras. Luego, se procedió a realizar el enrutamiento manualmente y se comprobó que estuviera correcto mediante el controlador de reglas de diseño (DCR). En la Figura 3.171 se muestra el diseño de la PCB que corresponde al sistema domótico, por otra parte, en la Figura 3.172 se destaca la PCB del sistema de las cortinas.

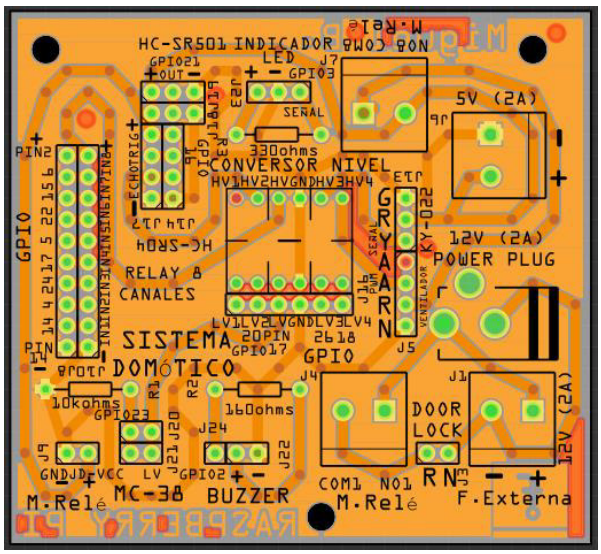


Figura 3.171 PCB del prototipo domótico

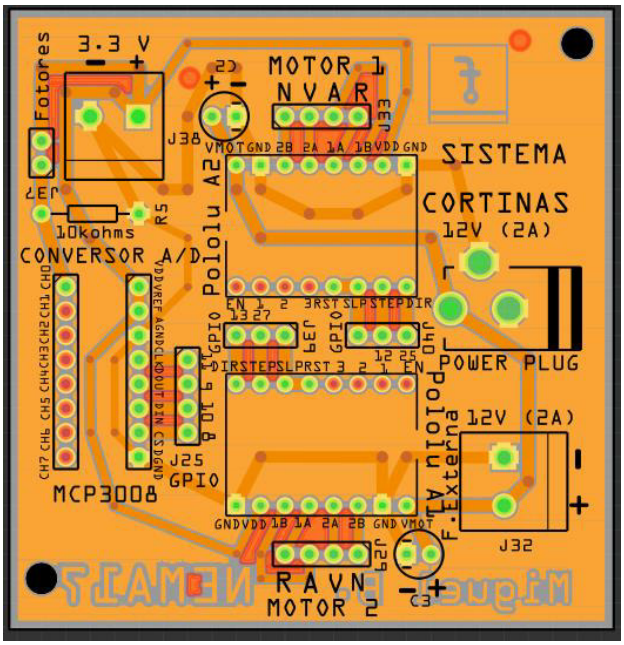


Figura 3.172 PCB del sistema de las cortinas

Diseño de la simulación en *DIALux Evo*

Principalmente, se realizó la simulación en *DIALux Evo* para tener una guía al momento de la implementación de los dispositivos. Para su diseño en primer lugar, se creó un plano de un domicilio, luego se extruyó las paredes y se dio forma a las distintas áreas. Más tarde, se agregó las texturas, tal cual se puede notar en la Figura 3.173.

Figura 3.173 Diseño 3D de un domicilio por medio de un plano

Figura 3.174 Luz recomendada según la estancia [60]

Acto seguido, a cada estancia de la vivienda se le agregó el nivel de luz adecuado en el programa según la Figura 3.174. Posteriormente, se ubicaron las luminarias en la simulación y se realizó el cálculo de nivel luminoso. En la Figura 3.175 se muestran los colores falsos.

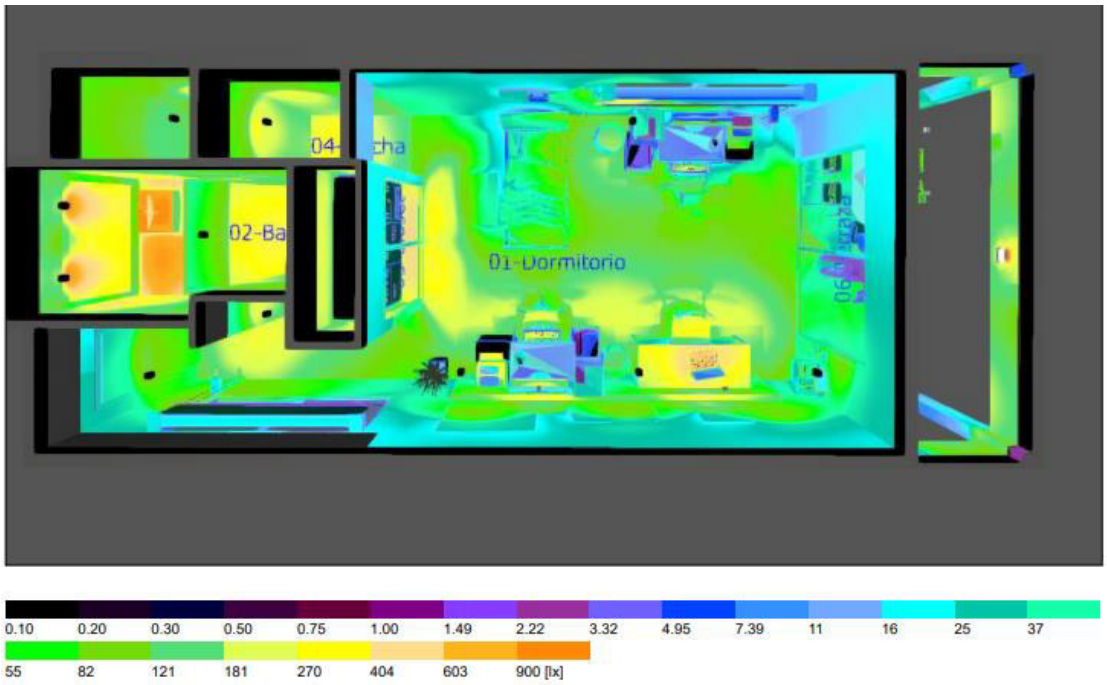


Figura 3.175 Representación gráfica de los niveles de iluminación

También es importante destacar que se añadieron en la simulación varios elementos como: computadoras, *laptops,* interruptores, el prototipo, los sensores, las persianas, entre otros. De la Figura 3.176 a la Figura 3.179 se encuentra lo dicho.

Figura 3.176 Ubicación del prototipo, el sensor ultrasónico y la cerradura

Figura 3.177 Ubicación de las persianas, los motores y el módem

Figura 3.178 Área del dormitorio

Figura 3.179 Área del pasillo

Estudio de Carga

Tal como se visualiza en la Figura 3.173 de la simulación, en un área puede haber más de una luminaria para cumplir el nivel de iluminación establecido. Para determinar cuántas luces se pueden conectar en paralelo a cada módulo relé se realizó el siguiente cálculo:

Donde:

I : (A) corriente
V : 110 (V_{AC}) voltaje
P : 75 (W) potencia de un foco

Por medido, de la Ecuación 3.7 se obtiene:

$$I = 0{,}68\ (A)$$

En base al valor calculado, tomando en consideración que cada módulo relé puede funcionar hasta una corriente de 10 (A), se obtiene que aproximadamente se podrían conectar 14 focos en paralelo a cada módulo relé con una potencia de consumo de 75 (W). Por otro lado, para efectuar el estudio de carga, se obtuvo el consumo de potencia de los aparatos que se ocuparon en el proyecto y se consideró las luces con una potencia de consumo de 100 (W) que corresponde a una luminaria del tipo incandescente, pero al momento de la instalación, se utilizaron del tipo led ya que su potencia es mucho menor. En la Tabla 3.19 se encuentra lo mencionado previamente.

Tabla 3.19 Estudio de carga

Descripción	**Cant.**	**Potencia**	**Potencia Total**	**FFUn(%) Factor frecuencia de uso**	**CIR Carga Instalada**	**FSn Factor de simulta-neidad**	**DMU Potencia máxima demandada**
Iluminación	17	100 (W)	1700 (W)	100 %	1700 (W)	50 %	850 (W)
PC	2	270 (W)	540 (W)	100 %	540 (W)	60 %	324 (W)
Laptop	2	50 (W)	100 (W)	50 %	50 (W)	50 %	25 (W)
Prototipo	1	43,6 (W)	43,6 (W)	100 %	43,6 (W)	100 %	43,6 (W)
Total:			**2383,6 (W)**		**2333,6 (W)**		**1242,6 (W)**

Se recomienda considerar todos los dispositivos del hogar para que la instalación eléctrica monofásica funcione adecuadamente si el prototipo se implementa en la vida real.

Entonces:

I : (A) corriente
V : 110 (V_{AC}) voltaje
P : 1242,6 (W)

Usando la Ecuación 3.7 se obtiene:

$$\mathrm{I} = 11{,}29\ (\mathrm{A})$$

Por lo tanto, con una corriente de 11,29 (A) la protección que se debe ocupar es de 14,11 (A), redondeando 15 (A). Después, se estableció los circuitos justo como se indica en la Tabla 3.20.

Tabla 3.20 Circuitos de la instalación eléctrica monofásica

Circuito	Potencia Total	FFU	CIR	Corriente	Protección
Iluminación	1700 (W)	100 %	1700 (W)	15,45 (A)	19,31 (A) redondeando 20 (A)
Aparatos	683,6 (W)	100 %	683,6 (W)	6,21 (A)	7,76 (A) redondeando 8 (A)

Finalmente, por medio de la Figura 3.180 se seleccionó el tipo de cable, además se creó el diagrama unifilar de la Figura 3.181.

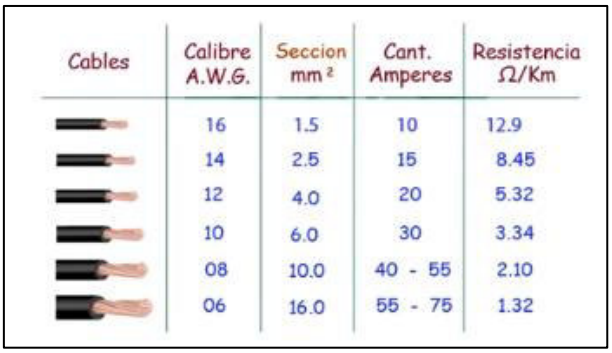


Figura 3.180 Tipos de cables según la cantidad de corriente [61]

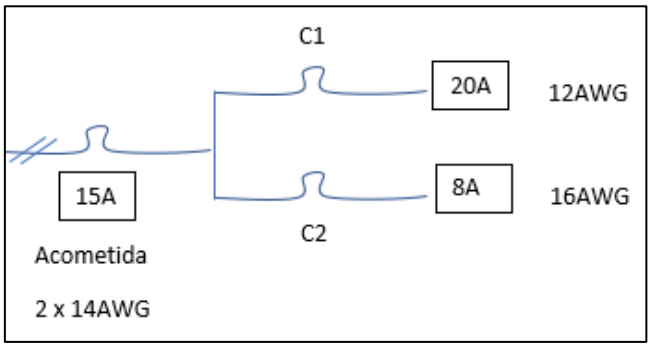


Figura 3.181 Diagrama unifilar

Cálculo de la horizontal

Inicialmente, la simulación de *DIALux Evo* se exportó en forma de plano a *AutoCAD* tal como se puede apreciar en la Figura 3.182. Más tarde, en este se agregaron los puntos de red.

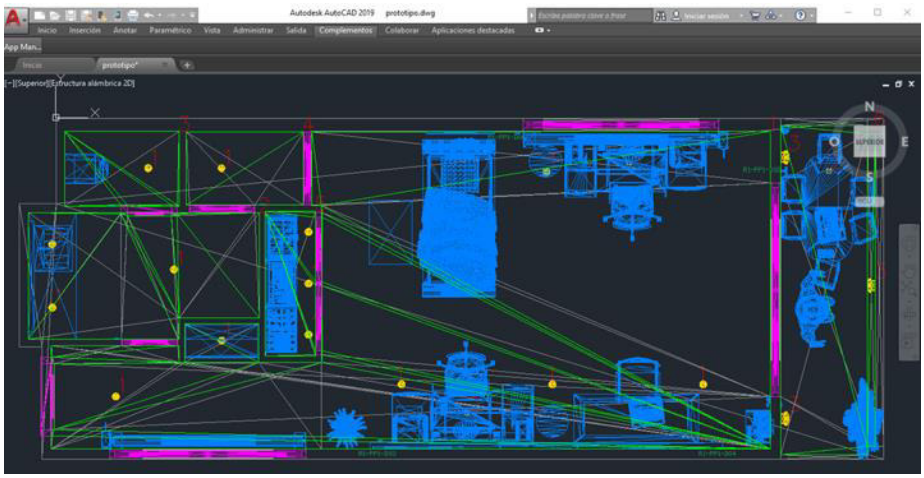


Figura 3.182 Plano *AutoCAD*

A continuación, por medio de la Ecuación 3.11 se calculó la longitud promedio del cable, tomando en consideración la distancia del punto más lejano y el más cercano.

$$Longitud\ promedio\ cable = \frac{distancia\ al\ pto\ lejano\ +\ distancia\ al\ pto\ cercano}{2}$$

Ecuación 3.11 Fórmula de la longitud promedio del cable [62]

Donde:

$distancia\ al\ pto\ lejano$: 5 (m)

$distancia\ al\ pto\ cercano$: 1 (m)

$Longitud\ promedio\ cable$: Metros

Usando la Ecuación 3.11 se obtiene:

$$\mathrm{L}ongitud\ promedio\ cable = 3\ (\mathrm{m})$$

Después, se calculó la distancia promedio a través de la Ecuación 3.12

$$Distancia\ promedio = longitud\ promedio\ cable + 10\% + 2{,}5m$$

Ecuación 3.12 Fórmula de la distancia promedio [62]

Donde:

$Distancia\ promedio$	:	Metros
$longitud\ promedio\ cable$	:	3 (m)

Por medio de la Ecuación 3.12 se obtiene:

$$Distancia\ promedio = 5{,}8\ (\mathrm{m})$$

Luego, se calculó el número de bobinas de cable que se necesitarían para hacer el cableado.

$$N\acute{u}mero\ bobinas\ cable = \frac{distancia\ promedio\ \ * \#puntos}{305}$$

Ecuación 3.13 Fórmula para calcular el número de bobinas [62]

Donde:

$\#puntos$	:	7
$distancia\ promedio$	:	5,8 (m)
$N\acute{u}mero\ bobinas\ cable$	:	cantidad de bobinas

Empleando la Ecuación 3.13 se obtiene:

$$N\acute{u}mero\ bobinas\ cable = 0{,}13 \quad 1Bobina$$

En base al valor calculado, se puede notar que debido a que se tiene pocos dispositivos la cantidad requerida para el prototipo es menor a 1 bobina. Sin embargo, ya que la interfaz web se diseñó para funcionar hasta con 20 dispositivos al momento de implementar, por ejemplo, este sistema en un laboratorio o centro de cómputo, en el diseño se debe tomar en cuenta estos dispositivos en el cálculo de la horizontal.

Posteriormente, se organizaron los elementos que se ocuparían en el cableado estructurado para tenerlos a la mano al momento de la instalación, justo como se puede observar en la Tabla 3.21.

Tabla 3.21 Componentes del cableado estructurado

Descripción	Cantidad		
Punto simple	1	1pto	
Puntos dobles	3	6pto	
Total puntos	7ptos		
Face Plate	Simple	Doble	
	1	3	
Jack RJ-45	14		
Patch panel	$\frac{\#Puntos}{48\ o\ 24}$	$\frac{7}{24} = 0{,}29$	1 patch panel
Switch	1 switch		
Patch cord	7 patch cords de 3ft (1m) para el área trabajo		

Por otro lado, también se realizó un diagrama del *Rack* en donde se deberían colocar los componentes de la Tabla 3.21. En la Figura 3.183 se muestra lo dicho.

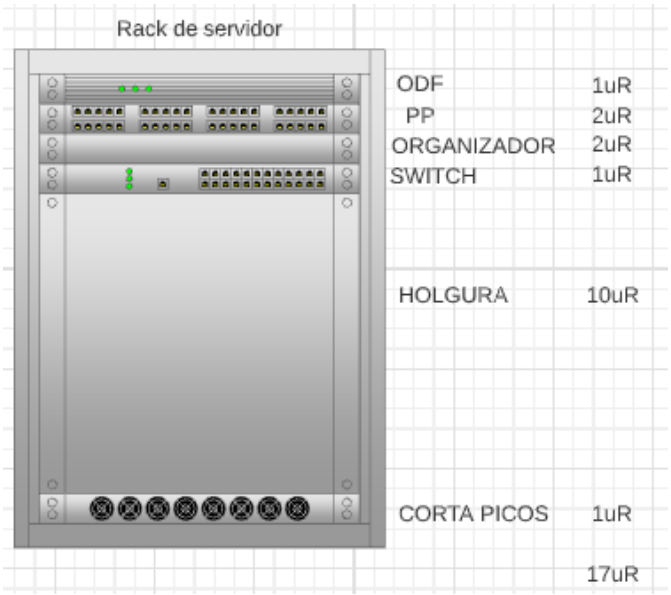


Figura 3.183 Rack en donde se deberían colocar los elementos

Finalmente, se diseñó el diagrama de la Figura 3.184 para tener una guía al conectar los componentes de red al prototipo.

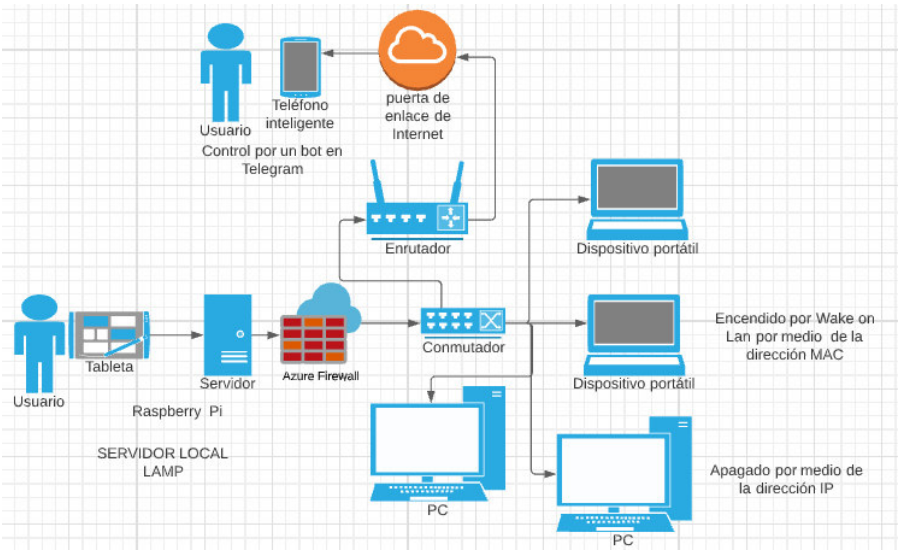


Figura 3.184 Diseño de red de los componentes del prototipo

Diseño de las piezas en 3D del prototipo en *Fusion 360*

- *Diseño de la estructura y piezas de los componentes*

En primer lugar, para el diseño de la estructura del prototipo y piezas de los componentes se estableció su tamaño, según los elementos que van a contener. Seguidamente se modeló las piezas en el programa *Fusion 360* tal como se destaca en la Figura 3.185. Por otra parte, también en las piezas se dejó el espacio para los tornillos y en algunos casos se añadió relieves de letras e imágenes.

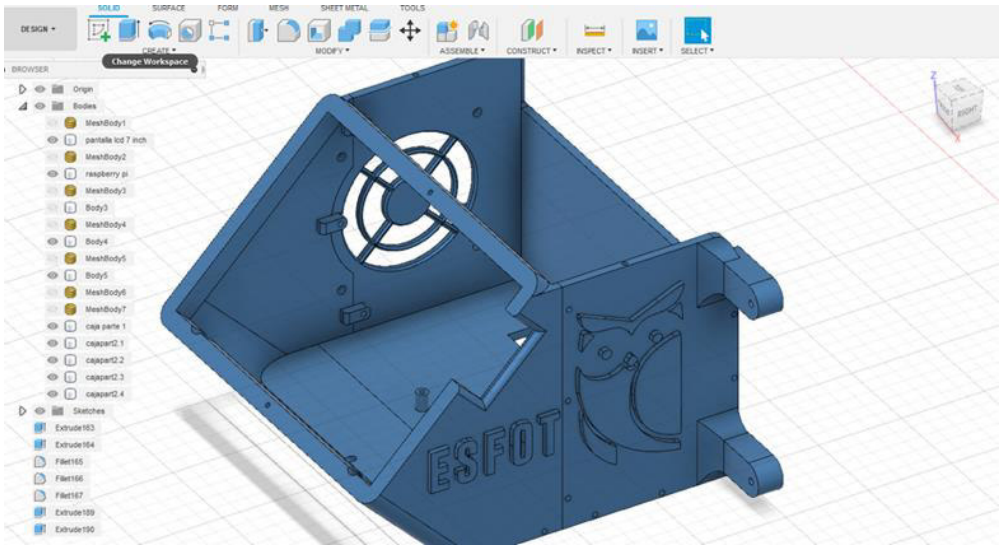


Figura 3.185 Diseño de las piezas en *Fusion 360*

De la Figura 3.186 a la Figura 3.189 se puede apreciar el modelado de las piezas que componen la estructura del prototipo y la de los componentes.

Figura 3.186 Diseño de la estructura del prototipo

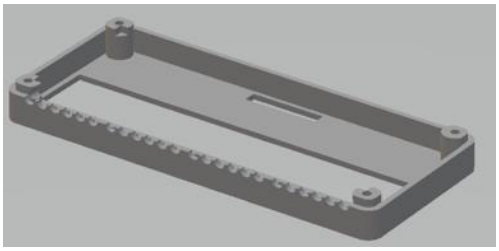


Figura 3.187 Diseño de la estructura del módulo relé

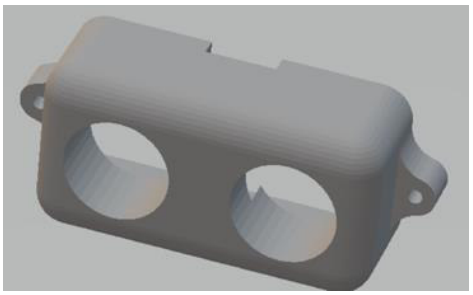


Figura 3.188 Diseño de la estructura del sensor ultrasónico

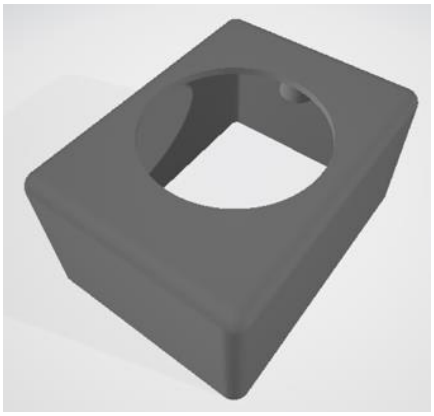


Figura 3.189 Diseño de la estructura del sensor *PIR*

- *Diseño del sistema de engranajes de las cortinas*

Primeramente, para el diseño del sistema de engranajes se debió tener en cuenta que cuando un engranaje reduce su velocidad gana fuerza (par motor). Para la velocidad del sistema de las cortinas se diseñó el diagrama de la Figura 3.190 y se calculó en base a la Ecuación 3.14.

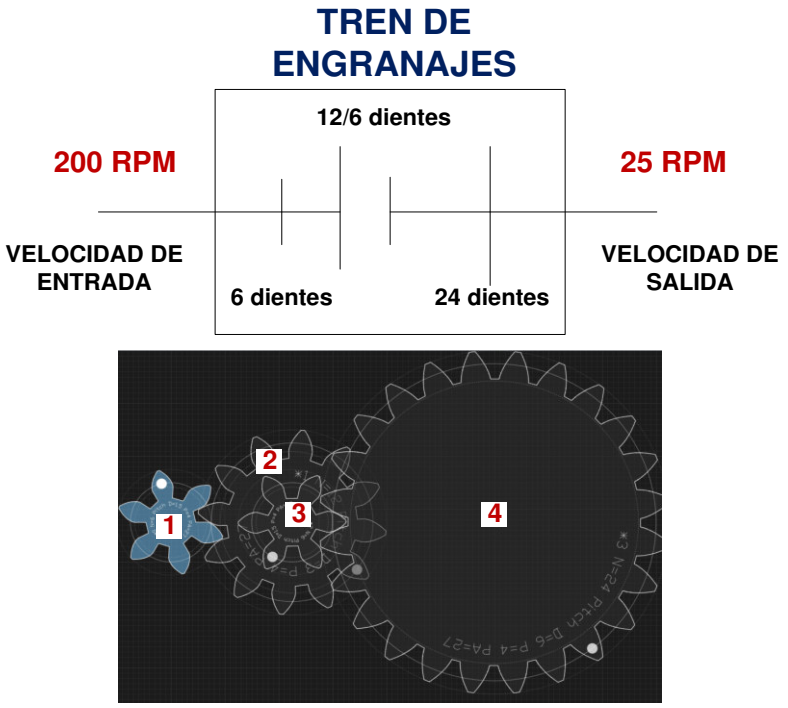


Figura 3.190 Tren de engranajes del sistema de las cortinas

$$Z_1 . N_1 = Z_2 . N_2$$

Ecuación 3.14 Fórmula para el cálculo de los engranajes [62]

Donde:

Z : número de dientes

N : velocidad (rpm)

Según las características del motor 17HS1910-P4170 este tiene una velocidad nominal de 200 (rpm). Ahora si se observa la Figura 3.190 se nota que tanto la rueda 2 como la 3 están en el mismo eje, de manera que giran con la misma velocidad en RPM.

Usando la Ecuación 3.14 para los engranajes 1-2 se obtiene:

$$\mathrm{N}_2 = 100\ (\mathrm{RPM})\ N2\ será\ la\ misma\ que\ la\ N3.$$

Empleando nuevamente la Ecuación 3.14 para los engranajes 3-4 se obtiene:

$$\mathrm{N}_4 = 25\ (\mathrm{RPM})$$

En base al valor calculado, la salida del tren de engranaje será de $25\ (\mathrm{RPM})$, por otra parte, para determinar la relación de los engranajes se ocupó la Ecuación 3.15.

$$\mathrm{R}_V = \mathrm{N}_s \,/\, \mathrm{N}_e$$

Ecuación 3.15 Fórmula para obtener la relación de los Engranajes [62]

Donde:

N_s : velocidad de salida 25 (RPM)

N_e : velocidad de entrada 200 (RPM)

R_V : relación de velocidad del engranaje

Por medio, de la Ecuación 3.15 se obtiene:

$$\mathrm{R}_V = \frac{1}{8}$$

Tomando en cuenta el valor calculado, el tren de engranaje va 8 veces más lento a la salida que a la entrada, por lo que la velocidad se reduce. Más tarde, para desarrollar la pieza que mueve a la cadena se tomó como referencia la Figura 3.191, específicamente la pieza que corresponde a la cadena de 6 (mm) – 6.6 (mm) del dispositivo *Brunt WiFi Blind Engine.*

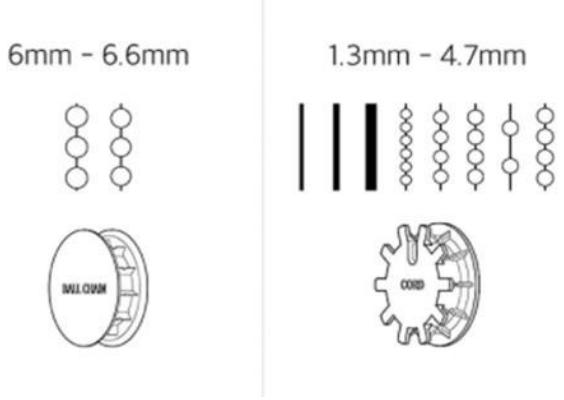


Figura 3.191 Pieza que mueve la cadena en el dispositivo *Brunt WiFi Blind Engine* [63]

Una vez con todos los datos a la mano se empezó con el diseño en *Fusion 360.* En el caso de los engranajes se empleó la herramienta *SPUR GEAR* tal cual se visualiza en la Figura 3.192. Esta herramienta permite la creación rápida de engranajes, a modo de ejemplo en la Figura 3.193 se muestra el engranaje de 24 dientes diseñado.

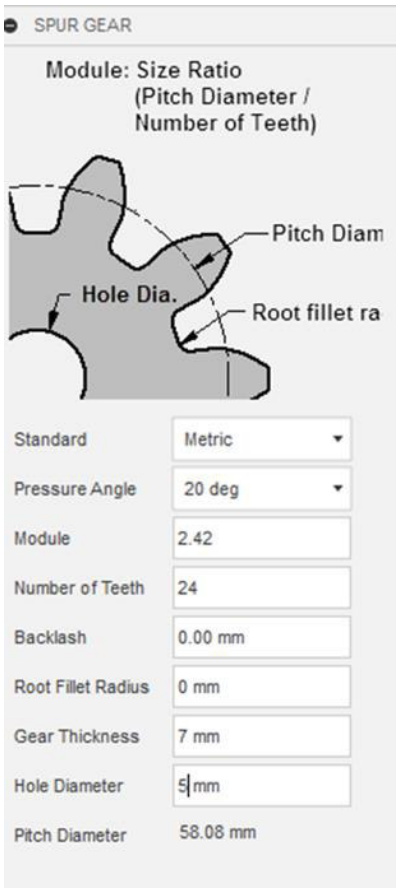


Figura 3.192 Creación de los engranajes

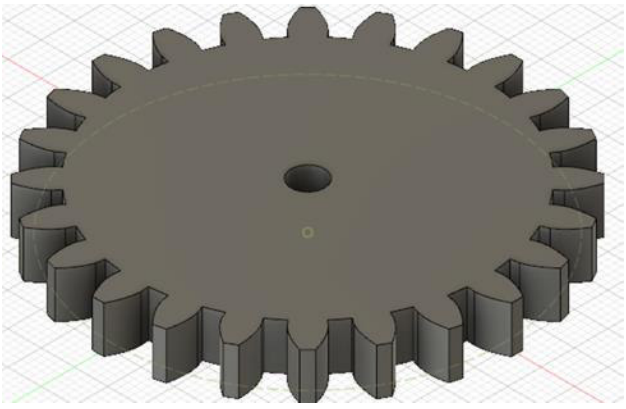


Figura 3.193 Engranaje de 24 dientes diseñado

Posteriormente, se combinó las piezas para obtener el engranaje doble y el engranaje que mueve la cadena de las persianas, justo como se observa en la Figura 3.194. Cabe destacar que en versiones futuras puede ser buena idea modelar también la pieza que corresponde a la cadena de 1,3 (mm) – 4,7 (mm) de la Figura 3.191 para tener mayor compatibilidad con varios tipos de cadenas.

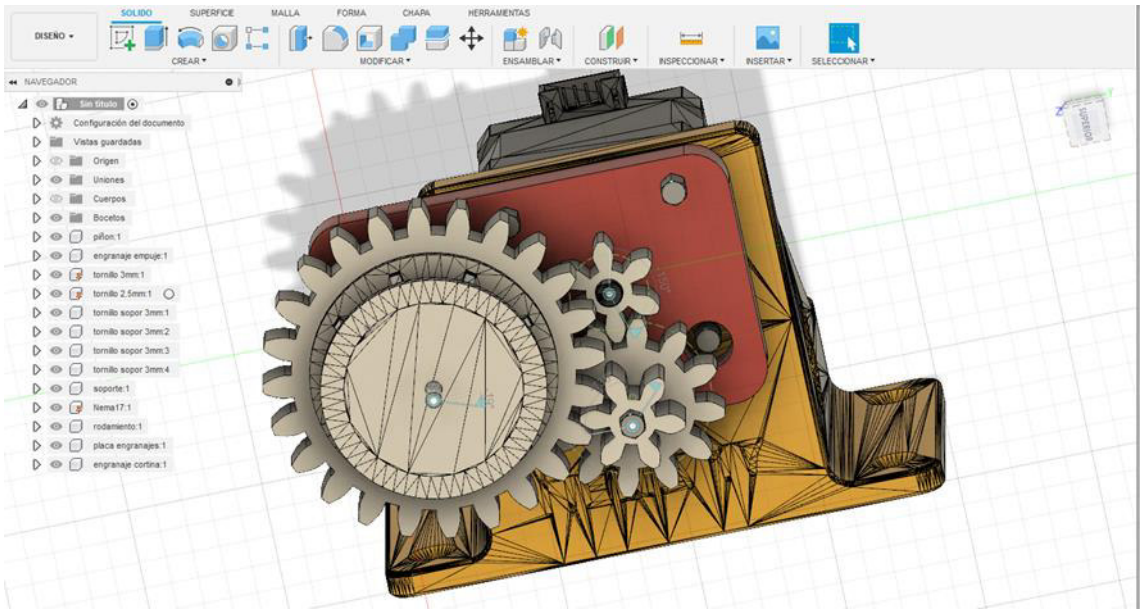


Figura 3.194 Diseño de las piezas del sistema de engranajes en *Fusion 360*

A continuación, se modeló el soporte del motor por medio de la Figura 3.195. Por otra parte, en la Figura 3.196 se puede apreciar el diseño de las cortinas terminado.

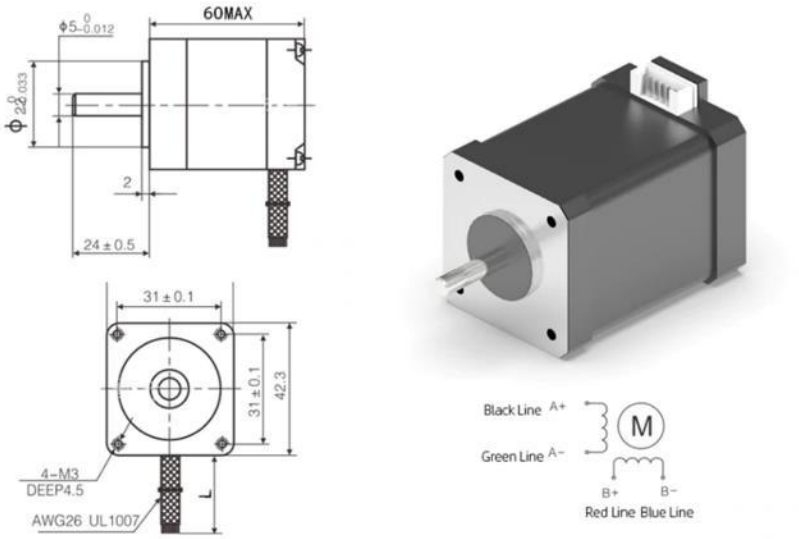


Figura 3.195 Características del motor a pasos elegido [3]

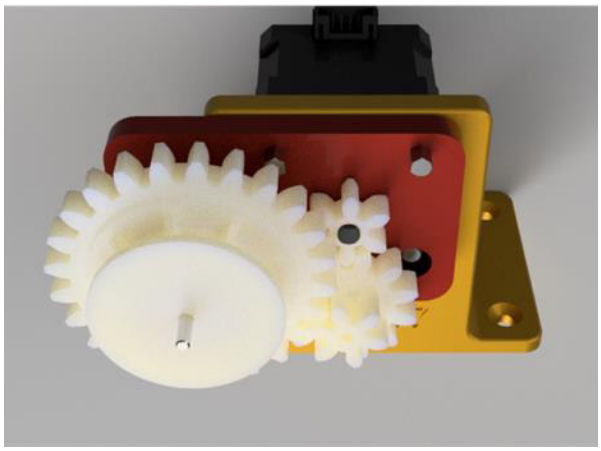


Figura 3.196 Diseño del sistema de las cortinas finalizado

Finalmente, cuando se acabó de modelar todas las piezas en 3D estas se exportaron a formato STL tal cual se ilustra a modo de ejemplo en la Figura 3.197.

engranaje cortina.stl	7/4/2021 12:35	Archivo STL	369 KB
engranaje empuje.stl	7/4/2021 12:33	Archivo STL	118 KB
piñon.stl	7/4/2021 12:31	Archivo STL	88 KB
placa engranajes.stl	7/4/2021 12:34	Archivo STL	91 KB
soporte.stl	7/4/2021 12:34	Archivo STL	502 KB

Figura 3.197 Piezas en formato STL

3.3 Creación de la interfaz web

Instalación de servidores

- *Instalación del servidor XAMPP*

Primeramente, se descargó *XAMPP* de la página oficial (https://www.apachefriends.org/es/index.html) tal como se muestra en la Figura 3.198.

Figura 3.198 Instalación del servidor *XAMPP*

Después, de la instalación del programa se habilitó los servicios de *Apache* y *MySQL.* En la Figura 3.199 se puede observar lo dicho.

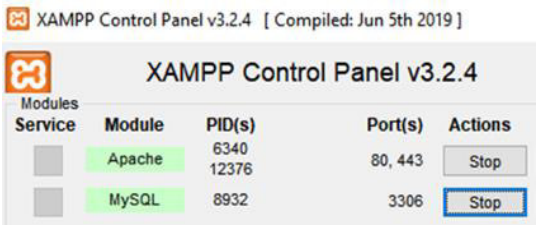


Figura 3.199 Habilitación de los servicios de *Apache* y *MySQL*

Más tarde, se probó que *phpMyAdmin* estuviera funcionando correctamente para esto se ingresó en el navegador y se digitó la dirección *localhost/phpmyadmin*. Como dirección IP de localhost se configuró la dirección IP 192.168.1.20. En la Figura 3.200 se visualiza lo mencionado anteriormente.

Figura 3.200 *phpMyAdmin* del ordenador con la dirección IP 192.168.1.20

- *Instalación del servidor LAMPP*

Antes de instalar *LAMPP* es buena idea actualizar el repositorio y los paquetes de *software*. Para este propósito se escribió en la terminal el comando *sudo apt update && sudo apt upgrade.*

- Instalación de *Apache*

Para instalar *Apache* se ocupó el comando *sudo apt install apache2.* Luego de su instalación Apache debería iniciarse automáticamente. Se puede comprobar su estado por medio del comando *systemctl status apache2,* como se observa en la Figura 3.201.

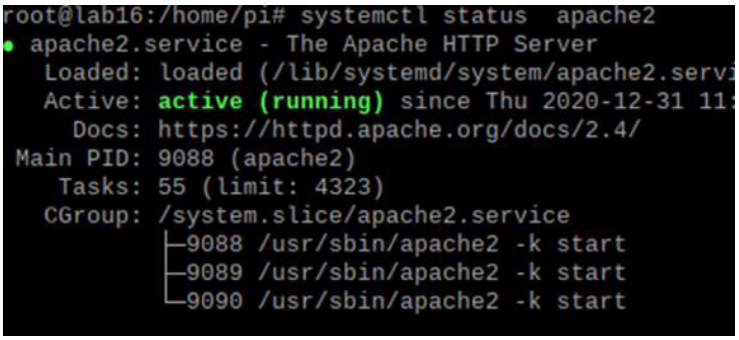


Figura 3.201 Comprobación del funcionamiento de Apache

Además, también se puede comprobar que *Apache* esté corriendo, ingresando la dirección IP que se configuró en la *Raspberry Pi* en el navegador, tal cual se mira en la Figura 3.202.

Figura 3.202 Comprobación del funcionamiento de Apache en el navegador

- Instalación de *php*

Para instalar *php* se utilizó el comando *sudo apt install php*. Ahora bien, para observar que *php* esté funcionando se creó el archivo *info.php* mediante el comando *sudo nano /var/www/html/info.php*. Después, dentro del archivo se colocó la línea de código *<?php phpinfo(); ?>*. En la Figura 3.203 se encuentra lo mencionado.

Figura 3.203 Archivo *info.php*

Más tarde, el archivo *info.php* se abrió en el navegador tal como se destaca en la Figura 3.204. En esta página se puede encontrar información acerca de la versión de *php* que se instaló y su compatibilidad con el sistema operativo.

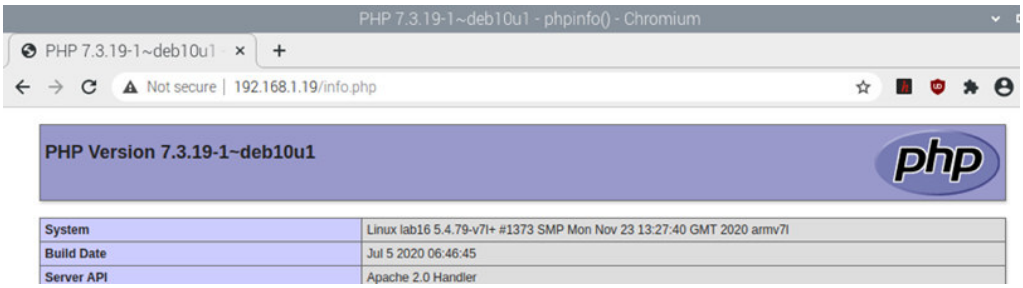


Figura 3.204 Comprobación del funcionamiento de *php* en el navegador

- Instalación de *MySQL*

Se empleó *MariaDB* como reemplazo directo de *MySQL*, para instalar el servidor de base de datos *MariaDB* en primer lugar se digitó el comando *sudo apt install mariadb-server php-mysql* en la terminal, a continuación, se reinició el servicio de *Apache* con el comando *sudo service apache2 restart.*

Después, se comprobó que *MariaDB* estuviera habilitado con el comando *systemctl status mariadb,* justo como se observa en la Figura 3.205. Si no se ha ejecutado *MariaDB*, se lo puede iniciar con el comando *systemctl start mariadb.*

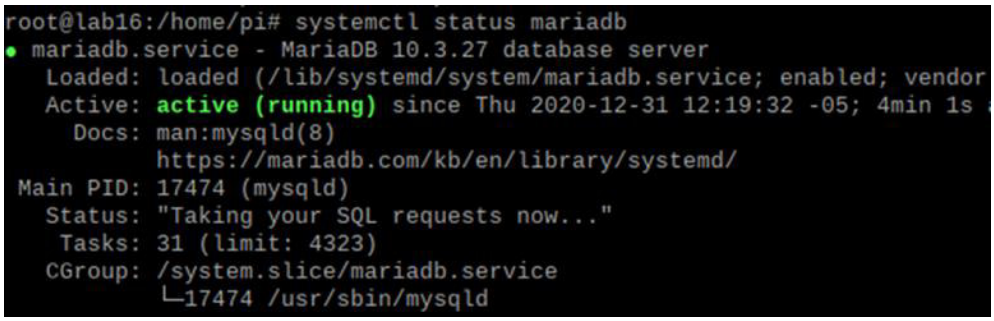


Figura 3.205 Comprobación del funcionamiento de *MariaDB*

Más adelante, se configuró el *script* de seguridad para lo cual se ingresó el comando *mysql_secure_installation*. Este comando pedirá crear una contraseña de *root* para *MariaDB*. Una vez que se estableció la contraseña, también se debe configurar una serie de parámetros como: remover los usuarios anónimos, desalojar la conexión remota del usuario *root*, recargar los privilegios de las tablas, entre otros. La configuración que se ocupó para cada parámetro va de la Figura 3.206 a la Figura 3.208.

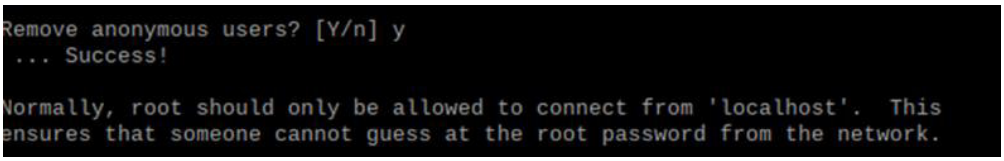


Figura 3.206 Configuración de *mysql_secure_installation* parte 1

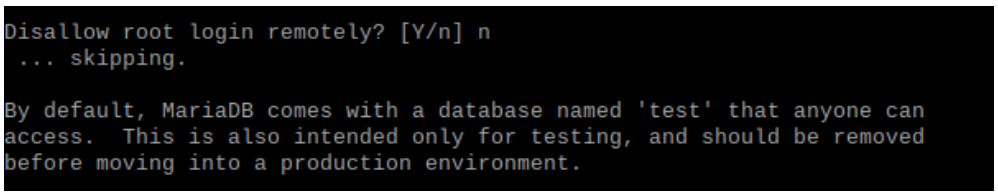


Figura 3.207 Configuración de *mysql_secure_installation* parte 2

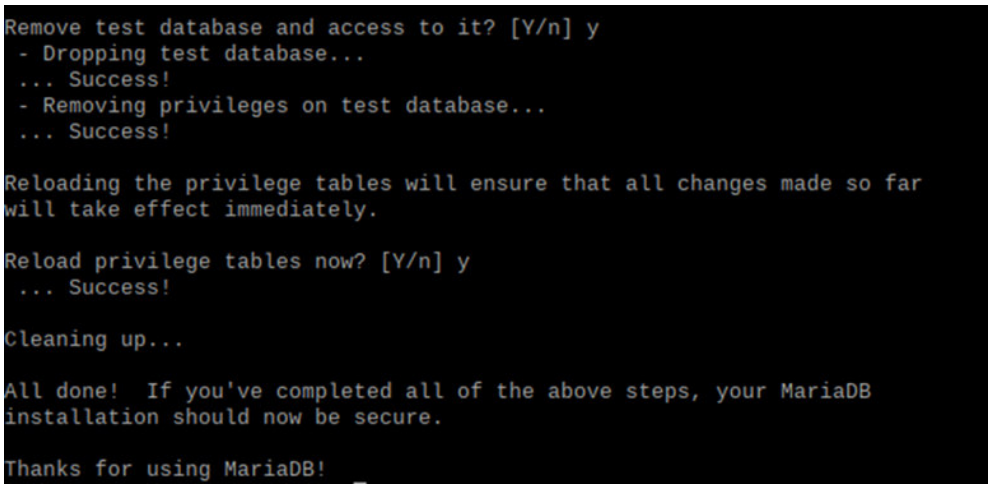


Figura 3.208 Configuración de *mysql_secure_installation* parte 3

Posteriormente se procedió a crear un nuevo usuario para darle aún más seguridad a la base de datos, para este fin se empleó el comando *sudo mysql --user=root –password*. Seguidamente, se ingresó las siguientes líneas de código:

> create user admin@'%' identified by 'Raspberry';

*> grant all privileges on *.* to admin@'%';*

> FLUSH PRIVILEGES;

> exit;

Donde:

- *create user*: Permite crear un usuario, en este caso el usuario es admin y la contraseña para ingresar a la base de datos es *Raspberry*
- '%'*:* Significa que el usuario puede conectarse desde *localhost* (equipo *Raspberry*) y desde afuera, por ejemplo: un ordenador de la red *LAN*.
- *grant all privileges*: Garantiza todos los privilegios al usuario
- *FLUSH PRIVILEGES:* Refresca los privilegios
- *exit:* Sale de la configuración

- Instalación de *phpMyAdmin*

Inicialmente, se usó el comando *sudo apt install phpmyadmin* para empezar la instalación. Más o menos en la mitad de la configuración el programa preguntará el tipo de servicio que se va a usar con *phpMyAdmin,* tal cual se ilustra en la Figura 3.209. Entonces se seleccionó *Apache 2*.

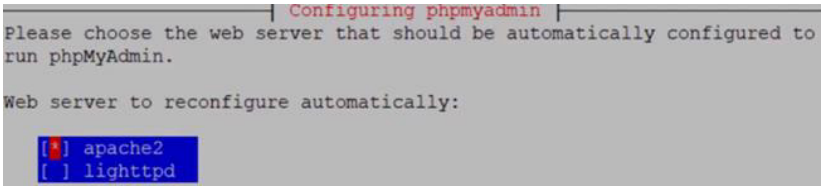


Figura 3.209 Selección del tipo de servicio que se va a emplear con *phpMyAdmin*

Tras esto, también se habilitó la configuración *dbconfig-common* en el menú de la Figura 3.210.

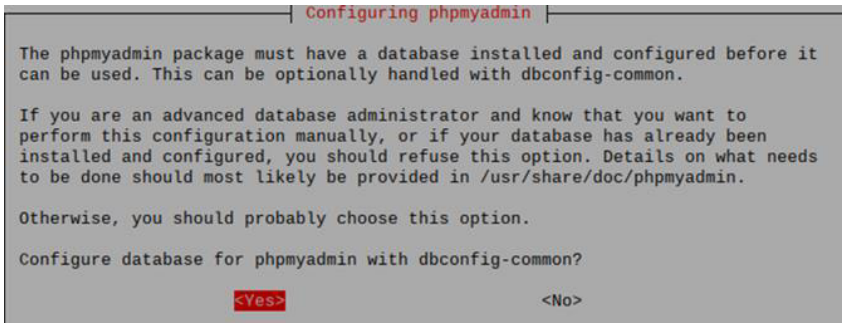


Figura 3.210 Habilitación de la configuración *dbconfig-common*

Acto seguido, se ingresó una contraseña para *phpMyAdmin.* Asimismo, acabada la instalación se enlazó *phpMyAdmin* con *MySQL* por medio del comando *sudo phpenmod mysqli*.

A continuación, se restauró el servidor *Apache* con el comando *sudo service apache2 restart* para que los cambios se guarden.

Finalmente se validó que *phpMyAdmin* estuviera funcionando, digitando la dirección web http://192.168.1.19/phpmyadmin/ en el navegador. En la Figura 3.211 se muestra lo dicho, en esta página se debe ingresar el usuario y contraseña establecidos anteriormente.

Figura 3.211 Página de ingreso a *phpMyAdmin*

Habilitación de *MySQL* para admitir conexiones remotas

En primer lugar, se ingresó en el archivo *50-server.cnf* con el comando *sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf*. Una vez dentro, se buscó *bind-address = 127.0.0.1* en el archivo de configuración, luego se cambió *a bind-address = 0.0.0.0* tal como se visualiza en la Figura 3.212. Es importante mencionar que la dirección IP 127.0.0.1 es la de *localhost*, por lo que se modifica para permitir conexiones remotas. Por último, para que los cambios surtan efecto se debe ejecutar el comando *sudo service mysql restart*.

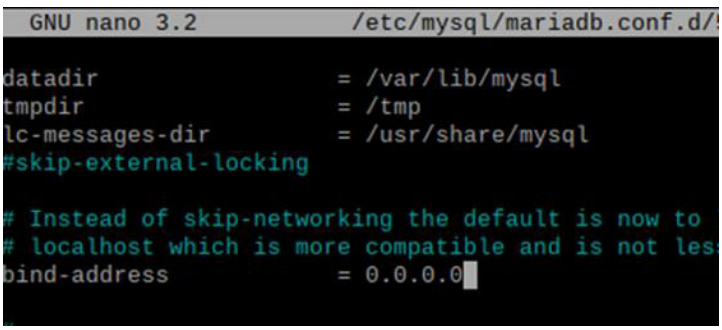


Figura 3.212 Archivo de configuración *50-server.cnf*

En la Figura 3.213 se encuentra abierto el *phpMyAdmin* de la *Raspberry Pi* en el navegador de una computadora de la red *LAN*.

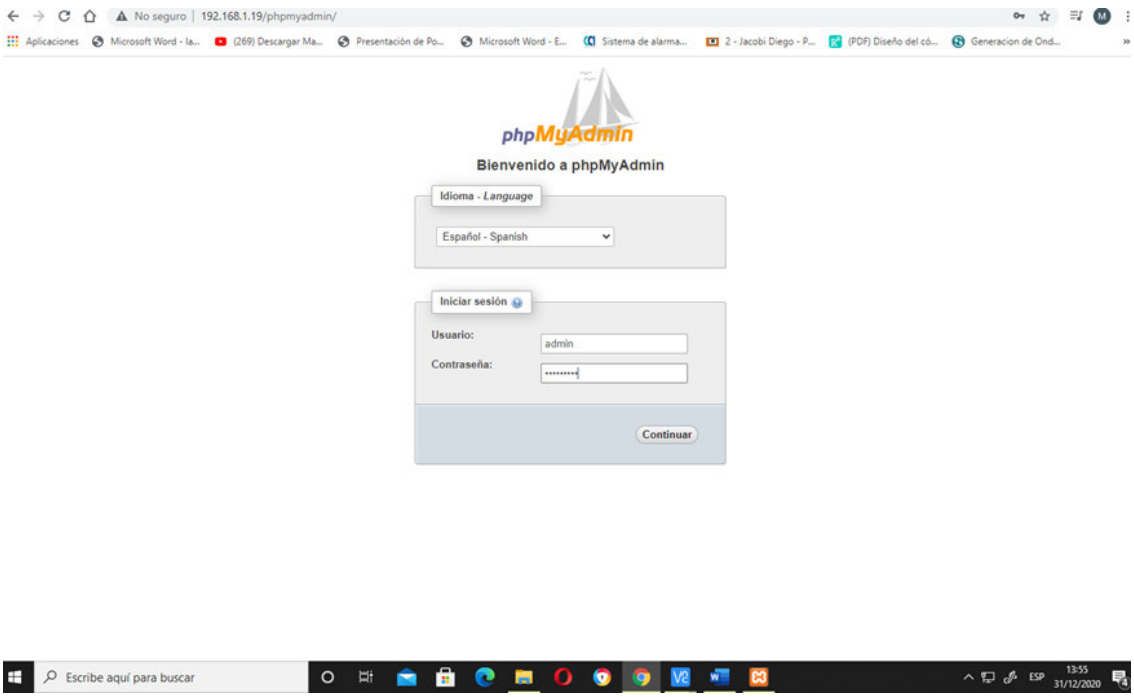


Figura 3.213 Conexión remota a *phpMyAdmin* desde un ordenador de la red *LAN*

Ahora si se abre el *phpMyAdmin* del servidor *XAMPP* en la *Raspberry Pi,* no se va a cargar, ya que este viene predefinido para trabajar localmente, en la Figura 3.214 se muestra lo dicho.

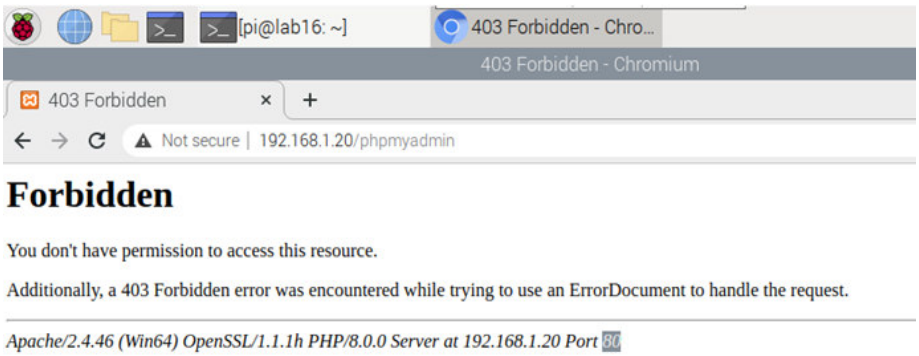


Figura 3.214 Conexión remota a *phpMyAdmin* del servidor *XAMPP*

Configuración básica de *phpMyAdmin*

Para realizar varias configuraciones como cambiar el tamaño de la base de datos o aumentar el tiempo de ejecución de los *scripts* se modificó el archivo *php.ini*.

Antes de nada, se digitó el comando */etc/php/7.3/apache2/php.ini* en la terminal de *Linux*. El archivo *php.ini* es el de la Figura 3.215.

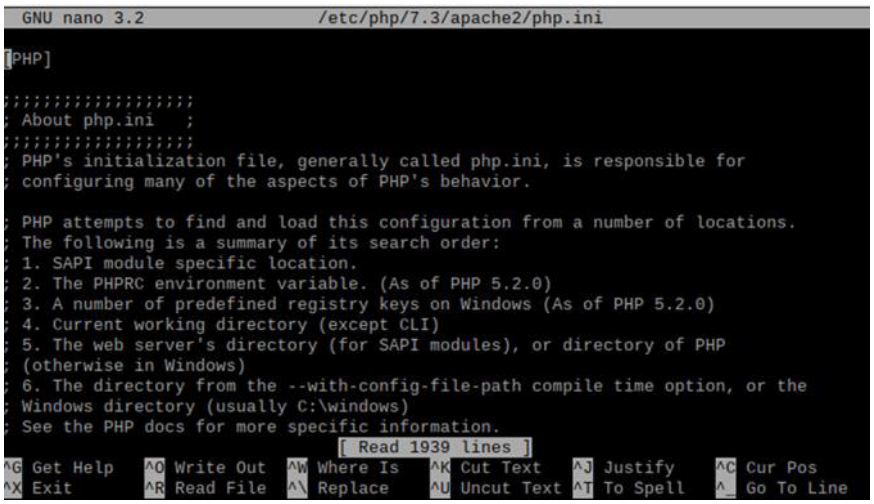


Figura 3.215 Archivo de configuración *php.ini*

Más adelante, en el archivo se aumentó los ficheros de entrada a 3000 mediante la línea de *código – max_execution_time = 3000*, también se aumentó el tamaño de datos del *POST* con la línea de código – *post_max_size = 16M*, el valor que se elija depende del tamaño de la base de datos, 16 (MB) es un valor bastante deseable, sin embargo, si la base de datos es muy grande se puede dejar en 128 (MB).

Por otra parte, el tamaño del archivo que se puede importar a la base de datos tenía un máximo permitido de 2 (MB) por lo que era muy pequeño y se cambió a 16 (MB) con la línea de código – *upload_max_filesize = 16M*. Seguidamente, también se aumentó el tiempo de datos de entrada a 1000 con la línea de código – *max_input_time = 1000*. Finalmente se guardó los cambios en el archivo y se digitó el comando *sudo service apache2 restart* para que las modificaciones surtan efecto. En la Figura 3.216 se puede ver que se ingresó a *phpMyAdmin y* los cambios resultaron exitosos, ya que ahora se puede importar archivos de hasta 16 (MB).

Figura 3.216 Archivo que se puede importar a la base de datos de 16 (MB)

Instalación de Dreamweaver y creación de los directorios de la interfaz web

Primeramente, se descargó e instaló el *software Dreamweaver,* tal como se muestra en la Figura 3.217.

Figura 3.217 Instalación de *Dreamweaver 2019*

Para la creación de un nuevo código se ingresó en el menú archivo y se seleccionó la opción nuevo, después se desplegó una ventana similar a la Figura 3.218. En esta ventana se eligió el tipo de código que se va a desarrollar y se presionó sobre la opción crear. Los códigos que se ocuparon en la interfaz web son: *HTML5*, *CSS*, *PHP* y *JavaScript*. Dependiendo del lenguaje de programación aparecerán parte de los códigos correspondientes.

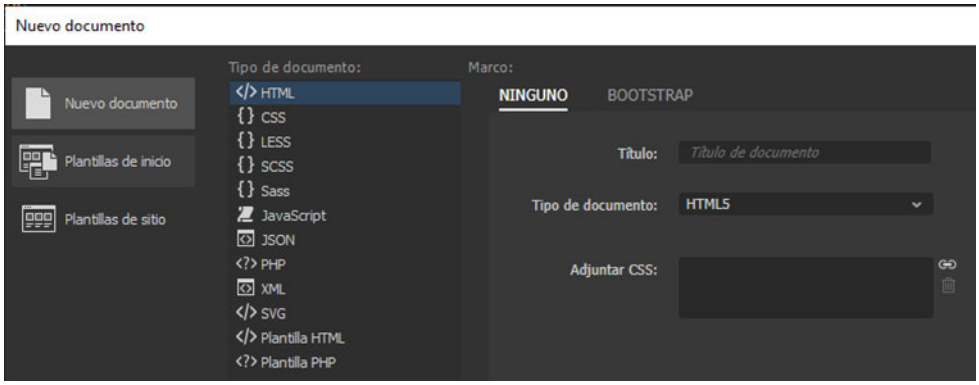


Figura 3.218 Creación de un nuevo documento

Por otro lado, para crear un nuevo sitio se seleccionó la opción nuevo sitio, más adelante se desplegó la ventana de la Figura 3.219, en esta se ingresó un nombre para el sitio.

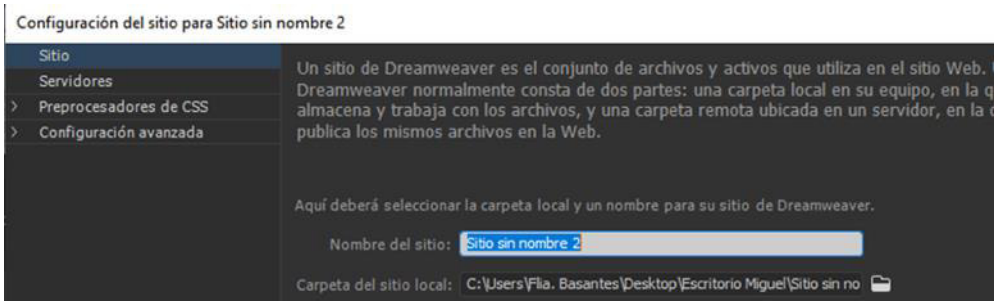


Figura 3.219 Creación de un nuevo sitio

En la carpeta del sitio web se crearon varias subcarpetas para almacenar distintos tipos de documentos como: imágenes, *scripts*, librerías, entre otros. Con el fin de tener organizado los archivos tal como se visualiza en la Figura 3.220. Luego, en *Dreamweaver* se abrió los archivos del sitio para comenzar a editarlos. Es importante mencionar que la carpeta del sitio debe estar como una subcarpeta de *htdocs* dentro del directorio del programa *XAMPP* para poder probar las páginas web en el navegador.

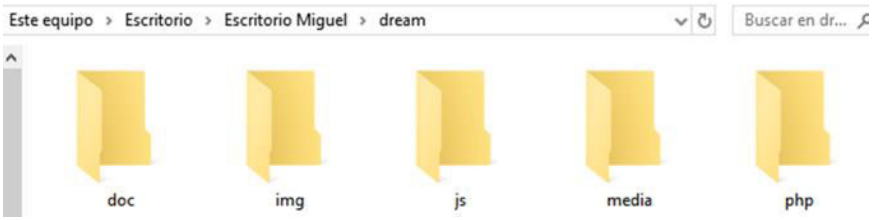


Figura 3.220 Creación del directorio de la interfaz web

Finalmente, se creó 2 archivos uno en *HTML5* y otro en *PHP* para constatar que el sitio estuviera corriendo adecuadamente. De la Figura 3.221 a la Figura 3.224 se halla lo mencionado anteriormente.

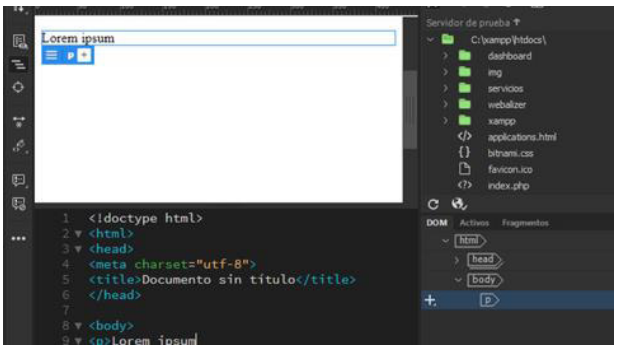


Figura 3.221 Archivo de *HTML5*

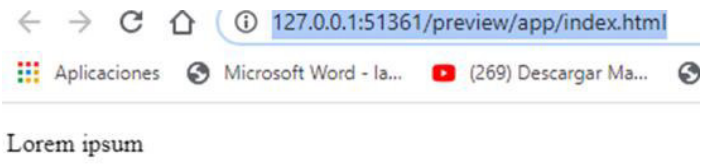


Figura 3.222 Vista previa del archivo de *HTML5*

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
</head>

<body>
<?php
echo "Hola Mundo";
    ?>

</body>
</html>
```

Figura 3.223 Archivo de *PHP*

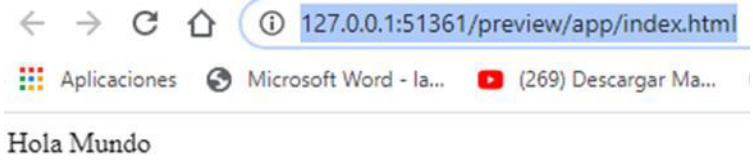


Figura 3.224 Vista previa del archivo de *PHP*

Instalación de *Photoshop* y edición de las imágenes para la interfaz web

Ya que las imágenes son muy necesarias en cualquier sitio web, la interfaz web del prototipo no es la excepción por este motivo se instaló el programa *Photoshop* tal cual se observa en la Figura 3.225.

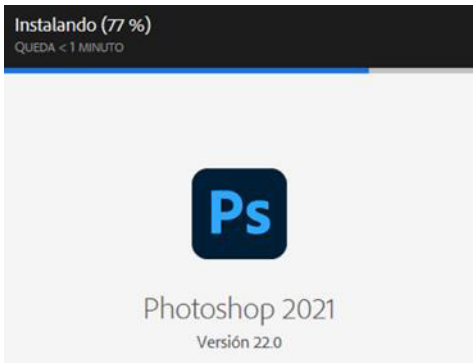


Figura 3.225 Instalación de *Photoshop* 2021

Para la edición de una imagen, se dio doble clic sobre esta y se seleccionó la opción abrir con *Photoshop* tal como se destaca en la Figura 3.226.

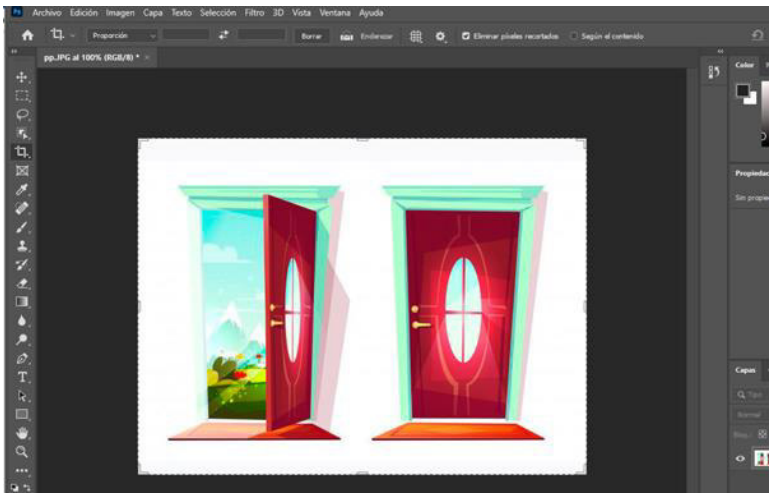


Figura 3.226 Abrir una imagen en *Photoshop*

Una vez en el programa se puede editar la imagen como mejor se desee con las herramientas del programa. Principalmente, se ocupó para cambiar el tamaño de las imágenes, crear el *banner* del inicio del video y ajustar el peso de estas para que no demoren mucho en cargar en la interfaz web. A modo de ejemplo, se nota como en la imagen de la Figura 3.227 se ajustó la anchura a 147 pixeles y la altura a 198 pixeles.

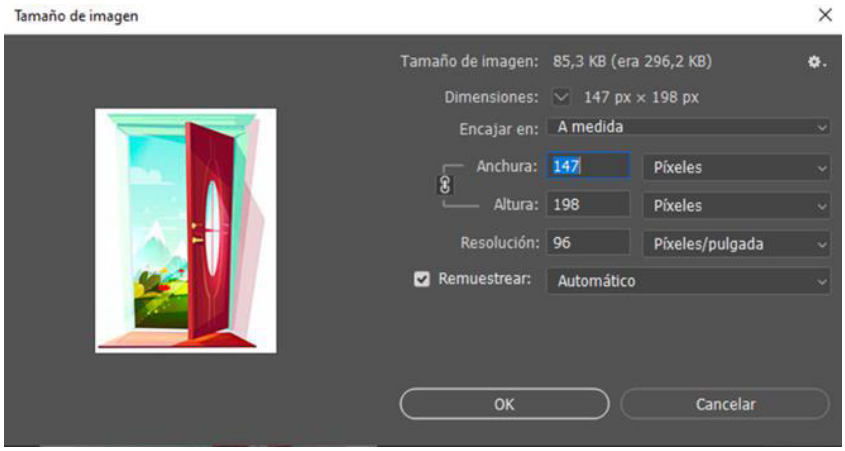


Figura 3.227 Edición del tamaño de una imagen de la interfaz web

A su vez para modificar el peso de la imagen se seleccionó el menú archivo del programa y se pulsó sobre la opción Exportar. Después dentro del submenú se escogió la opción Guardar para web (heredado). Más adelante, se mostró las configuraciones de la Figura 3.228. En esta configuración, se puede elegir la calidad de la imagen, esto influirá en su peso.

En todas las imágenes se usó la opción alta no obstante se puede escoger otra como: baja o muy alta. Finalmente, la imagen se guardó en la carpeta creada para almacenar las imágenes que se crearon previamente.

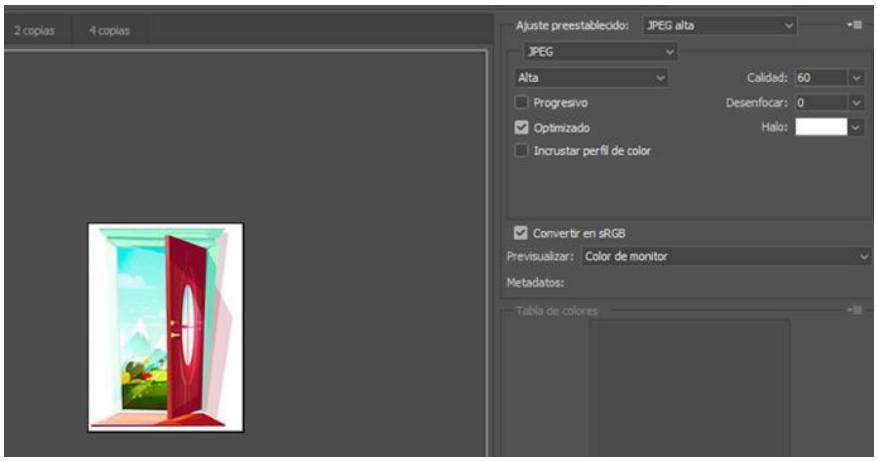


Figura 3.228 Edición de la calidad de la imagen

Después, de concluir todos los *scripts* de la interfaz en el editor de códigos *Dreamweaver,* toda la carpeta del sitio se traspasó a la *Raspberry Pi*. Es importante que esta carpeta, se guarde específicamente en la dirección */var/www/html* para que Apache muestre el sitio web en el navegador. En la Figura 3.229 se puede observar todos los directorios de la interfaz web.

/var/www/html/Montar/2126/

Nombre	Tamaño	Modificado	Permisos	Propiet...
..		10/3/2021 11:31:16	rwxr-xr-x	root
users		19/7/2021 21:38:48	rwxr-xr-x	root
puerta		1/7/2021 19:40:58	rwxr-xr-x	root
luz		1/7/2021 13:31:53	rwxr-xr-x	root
computadoras		30/6/2021 14:46:32	rwxr-xr-x	root
cortina		21/5/2021 19:38:55	rwxr-xr-x	root
vistas		10/3/2021 11:32:09	rwxr-xr-x	root
sistema		10/3/2021 11:31:27	rwxr-xr-x	root
plugins		10/3/2021 11:31:25	rwxr-xr-x	root
js		10/3/2021 11:31:24	rwxr-xr-x	root
jquery		10/3/2021 11:31:24	rwxr-xr-x	root
img		10/3/2021 11:31:24	rwxr-xr-x	root
css		10/3/2021 11:31:21	rwxr-xr-x	root
bootstrap		10/3/2021 11:31:17	rwxr-xr-x	root
procesa.php	1 KB	13/4/2021 20:19:25	rw-r--r--	root
conexion.php	1 KB	1/3/2021 20:19:37	rw-r--r--	root
login.sql	5 KB	1/3/2021 20:12:23	rw-r--r--	root
index.php	6 KB	28/2/2021 13:52:56	rw-r--r--	root
sesion2.php	9 KB	26/2/2021 13:50:58	rw-r--r--	root

Figura 3.229 Traspaso del sitio web a la dirección */var/www/html*

Configuración de *Apache2* para ejecutar *scripts* de *Python*

Primeramente, se instaló el *software* necesario con el comando apt-*get -y install perl libcgi-pm-perl ruby python curl,* más adelante se habilitó el módulo *CGI* con el comando *a2enmod cgid.* Seguidamente, se reinició el servidor web mediante el comando *systemctl restart apache2.* En la Figura 3.230 se encuentra lo mencionado anteriormente.

```
root@lab16:/home/pi# a2enmod cgid
Enabling module cgid.
To activate the new configuration, you need to run:
  systemctl restart apache2
root@lab16:/home/pi#  systemctl restart apache2
```

Figura 3.230 Habilitación del módulo *CGI*

Luego, de habilitar el módulo *CGI*, los *scripts* solo se podrán ejecutar desde el directorio */usr/lib/cgi-bin* de forma predeterminada. Sin embargo, para ejecutar los *scripts* desde otro directorio se modificó el archivo de configuración *cgi-enabled.conf* con el comando *nano -c /etc/apache2/conf-available/cgi-enabled.conf,* en este archivo se ingresó las líneas de código de la Figura 3.231.

El directorio que se ocupó en lugar de */usr/lib/cgi-bin*, se creó con el comando *mkdir /var/www/html/cgi-enabled*. Por otra parte, en la Figura 3.232 se nota que en la dirección */var/www/html* se almacenó tanto el directorio de la interfaz web como el de los *scripts* de *Python*. En cambio, en la Figura 3.233 se puede ver que en el directorio *cgi-enabled* se guardó todos los *scripts* diseñados anteriormente, excepto los que están relacionados al *Bot* de *Telegram* ya que estos se colocaron en el directorio aplicaciones, tal como se indicó en la Figura 3.121.

```
 GNU nano 3.2      /etc/apache2/conf-available/cgi-enable

<Directory "/var/www/html/cgi-enabled">
    Options +ExecCGI
    AddHandler cgi-script .cgi .pl .rb .py
</Directory>
```

Figura 3.231 Habilitación del módulo *CGI* en otro directorio

```
root@lab16:/var/www/html# ls
cgi-enabled  index.html  info.php  Montar
```

Figura 3.232 Creación del directorio *cgi-enabled*

```
root@lab16:/var/www/html/cgi-enabled# ls
abrir.py        encender1.py   focoff5.py   led.py        pir3.py
almacenar1.txt  estado2.py     focoff.py    motor1111.py  pir.py
almacenar2.txt  estado3.py     focon1.py    motor1112.py  prueba2.py
almacenar3.txt  estado4.py     focon2.py    motor1113.py  prueba.py
almacenar4.txt  estado4u.py    focon3.py    motor111.py   puertaauto2.py
almacenar5.txt  estado5.py     focon4.py    motor112.py   puertaauto3.py
almacenar6.txt  estado6.py     focon5.py    motor113.py   puertaauto.py
almacenar7.txt  estadomag2.py  focon.py     motor11.py    reiniciar2.py
almacenar8.txt  estadomag3.py  fotores1.py  motor211.py   reiniciar.py
almacenar9.txt  estadomag4.py  fotores2.py  motor212.py   salida.out
almacenar.txt   estadomag.py   fotores3.py  motor213.py   telegramapi.py
apagar2.py      estado.py      fotores.py   motor21.py    ultrasonico.py
apagar.py       estadou.py     index2.py    motor311.py   ultrasonico.pyc
cancelar2.py    focoff1.py     index.py     motor312.py   venti1.py
cancelar.py     focoff2.py     ir2.py       motor411.py   venti.py
cerrar.py       focoff3.py     ir3.py       motor412.py
ejem.py         focoff4.py     ir.py        pir2.py
```

Figura 3.233 Archivos y *scripts* de *Python* del directorio *cgi-enabled*

Posteriormente, se probó que el módulo *CGI* estuviera funcionando con el nuevo directorio *cgi-enabled*. Para esto, se creó el archivo llamado index.py tal cual, se observa en la Figura 3.234. En este *script*, se colocó un pequeño mensaje "Prueba de *Script* con *Python*" a través de lenguaje *HTML*. Más tarde, se le dio los permisos necesarios con el comando *chmod 705 /var/www/html/cgi-enabled/index.py*. Acto seguido, se probó el *script* en el navegador accediendo a la dirección web http://localhost/cgi-enabled/index.py, tal como se destaca en la Figura 3.235. También se puede probar el *script* digitando el comando *curl http://localhost/cgi-enabled/index.py* en la terminal.

```
 GNU nano 3.2                          index.py

#!/usr/bin/env python

print "Content-type: text/html\n\n"
print "<html>\n<body>"
print "<div style=\"width: 100%; font-size: 40px; font-weight: bold; text-align: center;\$
print "Prueba de script con Python"
print "</div>\n</body>\n</html>"
```

Figura 3.234 *Script* de prueba *index.py*

Figura 3.235 Ejecución del *script* de prueba *index.py* en el navegador

Tal como el *script* de prueba anterior, a todos los archivos del directorio *cgi-enabled* se les otorgó el permiso necesario con el comando *chmod 705*, pero además para evitar que existan errores fue necesario darles otro permiso a los *scripts* ya que puede pasar que solo se ejecute en la terminal y no en el navegador, debido a que el servidor *Web Apache* en *Linux* usa habitualmente el usuario *www-data.*

Para esto se ingresó a la carpeta *cgi-enabled* y a cada *script* se le dio el permiso de usuario y grupo *www-data* con el comando *chown www-data:www-data* ,tal cual se observa en la Figura 3.236.

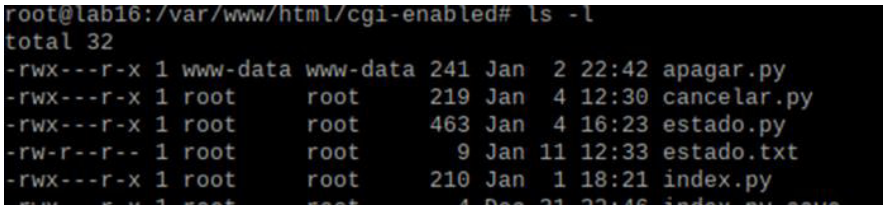


Figura 3.236 Añadiendo el permiso de usuario y grupo *www-data*

A continuación, se configuró el comando *sudo* para el grupo *www-data* para esto, se ingresó al archivo *sudoers* por medio del comando *nano /etc/sudoers*. En este archivo se ingresó la línea de código *www-data ALL=(ALL) NOPASSWD:ALL* tal como se mira en la Figura 3.237. Finalmente, se digitó el comando *service apache2 restart* para que los cambios se guarden

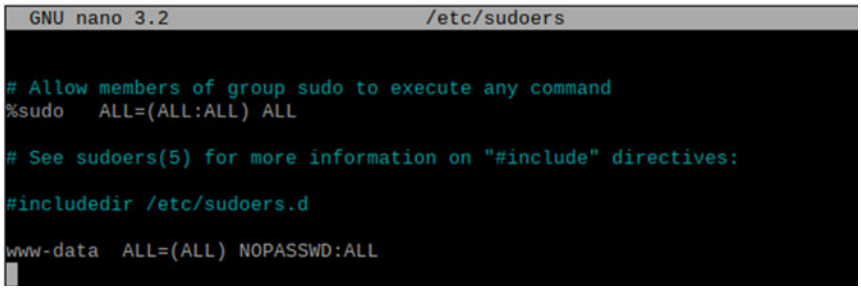


Figura 3.237 Configuración del archivo *sudoers*

Diseño de la interfaz web

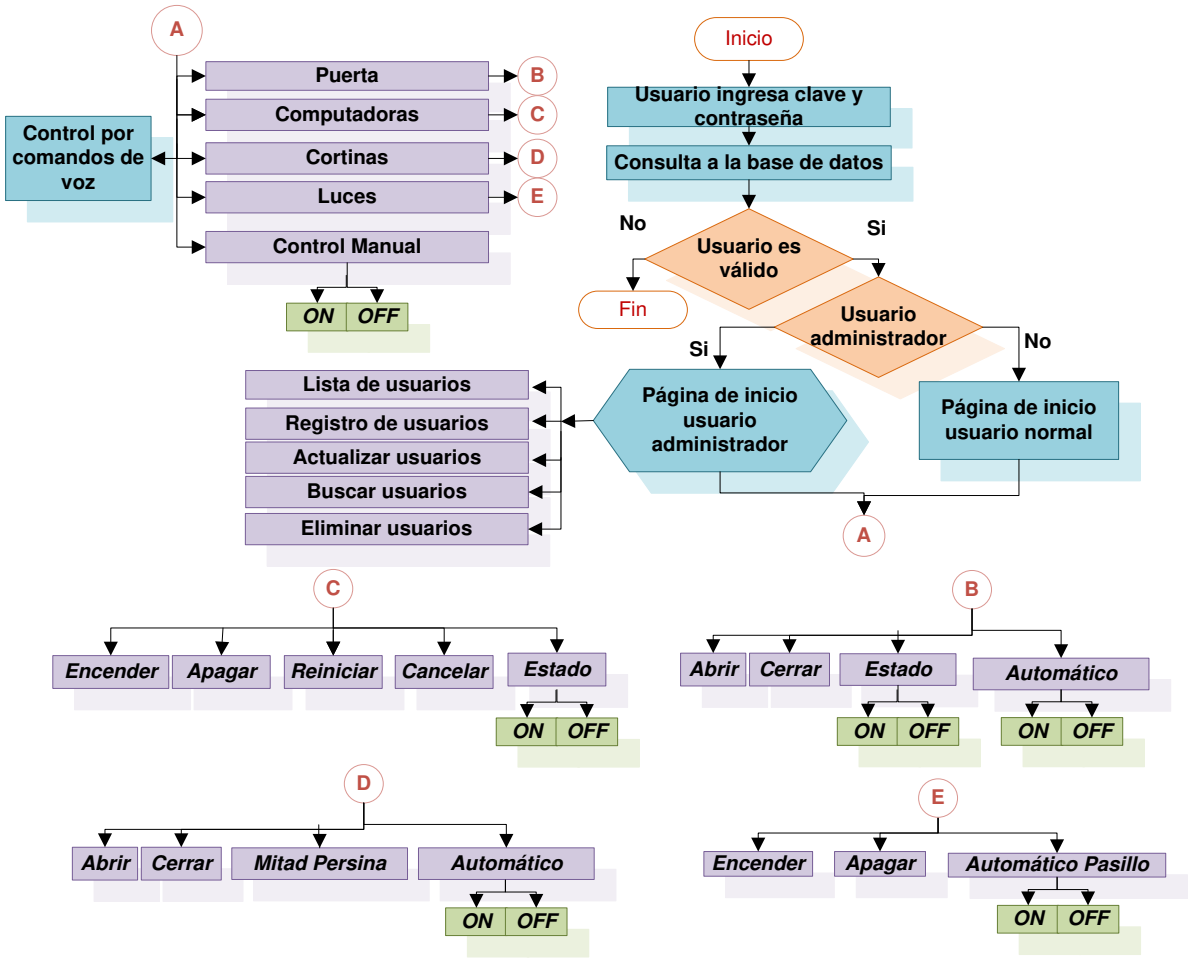


Figura 3.238 Diseño de la interfaz web

Por medio del diagrama de flujo de la Figura 3.238 se realizó la programación de los *scripts* que conforman la interfaz web. A continuación, se detallan las partes más relevantes del proceso de diseño:

- *Creación de la base de datos*

Primeramente, se asignó el nombre a la base de datos. Más adelante, se creó 2 tablas, en la primera tabla llamada rol se generó 2 columnas idrol y rol. Por otra parte, en la segunda tabla llamada usuario se generó 6 columnas idusuario, nombre, correo, usuario, clave, rol y estatus tal cual se mira en la Figura 3.239.

Luego, se relacionó las 2 tablas por medio del rol, es decir, si en la tabla usuario consta el rol = 1 el usuario es de tipo administrador en cambio si el rol = 2 el usuario es de tipo usuario normal, dependiendo el rol del usuario este tendrá distintos privilegios en la interfaz web.

Por otro lado, la columna estatus sirve para dar de baja o alta a un usuario si el estatus es 0 el usuario ya no podrá ingresar a la interfaz, aunque esté registrado en la base de datos. Por el contrario, si el estatus es 1 el usuario puede ingresar normalmente. De esta manera se garantiza el restablecimiento del usuario si se borró por error el registro de un usuario en la interfaz.

Figura 3.239 Creación de la base de datos

- *Conexión con la base de datos*

En primer lugar, se creó el *script* conexion.php, dentro de este, se empleó el comando *@mysqli_connect($host, $user, $password, $db);*. Este comando, es el que permite la conexión con la base de datos, en él se debe ingresar el nombre del host, usuario, contraseña y nombre de la base de datos. En caso de que la conexión falle se colocó un pequeño mensaje para conocer este error. Lo mencionado anteriormente se encuentra en la Figura 3.240.

conexion.php

```php
<?php

    $host = 'localhost';
    $user = 'root';
    $password = 'Raspberry';
    $db = 'login';

    $conection = @mysqli_connect($host,$user,$password,$db);

    if(!$conection){
        echo "Error en la conexión";
    }
    /* Aqui agregamos el host , el susuario , contraseña y el nombre de
    la base de datos y con el comando @mysqli_connect nos conectamos*/


?>
```

Figura 3.240 *Script* para la conexión con la base de datos

- *Página de login con HTML5, CSS, PHP y JavaScript*

Para la página de *login* se creó varios archivos. En el primer archivo se utilizó *PHP* combinado con el leguaje *HTML5* para la realización de un formulario el cual permite enviar el usuario y la contraseña. Por otra parte, se diseñó otro archivo en lenguaje *CSS* para dar el estilo a la página de inicio de sesión.

En cambio, a través de un archivo en *JavaScript* y por medio de la librería *JQuery* se generó la animación de los campos de usuario y contraseña.

Primeramente, a los campos de usuario y contraseña se les agregó la clase *input*, luego se creó 2 funciones tal como se muestra en la Figura 3.241. En la primera función se añadió la clase *focus* y en la segunda se removió esta clase con la finalidad, de que cada vez que el usuario pulse sobre los campos se ejecute la animación de la Figura 3.242.

```
const inputs = document.querySelectorAll(".input");


function addcl(){
    let parent = this.parentNode.parentNode;
    parent.classList.add("focus");
}

function remcl(){
    let parent = this.parentNode.parentNode;
    if(this.value == ""){
        parent.classList.remove("focus");
    }
}


inputs.forEach(input => {
    input.addEventListener("focus", addcl);
    input.addEventListener("blur", remcl);
});
/*Creamos un evento para que cada vez que pulsemos
sobre la entrada se genere la animacion*/
```

Figura 3.241 *Script* para la conexión con la base de datos

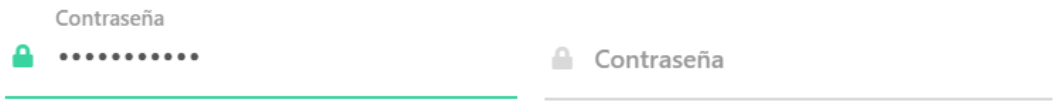


Figura 3.242 Animación del campo contraseña

Por otro lado, para enviar los campos al servidor se empleó el método *POST*. Además, en estos campos se agregó algunos iconos por medio de la librería *Font Awesome* (https://fontawesome.com/v4.7.0/icons/), en la Figura 3.243 se puede ver a modo de ejemplo el icono y la clase que se empleó para el campo usuario.

Figura 3.243 Animación del campo contraseña

Más tarde, se ocupó algunas ilustraciones en formato svg del banco de imágenes *unDRAW* (https://undraw.co/illustrations), para el diseño de la página de *login* tal como se visualiza en la Figura 3.244

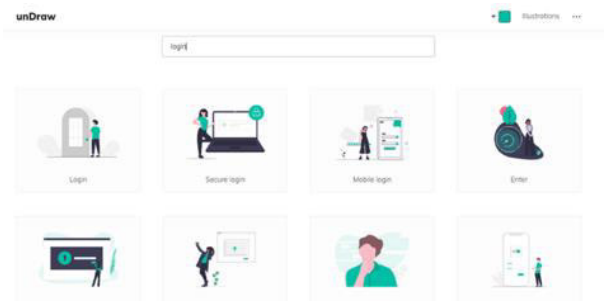


Figura 3.244 Ilustraciones en formato svg

Posteriormente, en la página de *login*, se agregó el *script* de la Figura 3.245 para generar varias alertas, la primera alerta se mostrará cuando la variable *jsVar* tenga un valor de 1 con el mensaje *"Debe ingresar un usuario y/o password",* en cambio cuando esta tenga un valor de 3 se indicará el mensaje *"Usuario y/o password incorrecta".* Las alertas diseñadas se encuentran en la Figura 3.246 y Figura 3.247.

```
<script type="text/javascript">
var jsVar = "<?php echo $retorna; ?>";
if(jsVar == 1){

            Swal.fire({
            type:'warning',
            title:'Debe ingresar un usuario y/o password',
          });
           }
if(jsVar == 3){

             Swal.fire({
                   type:'error',
                   title:'Usuario y/o password incorrecta',
               });
}

</script>
```

Figura 3.245 *Script* para mostrar alertas

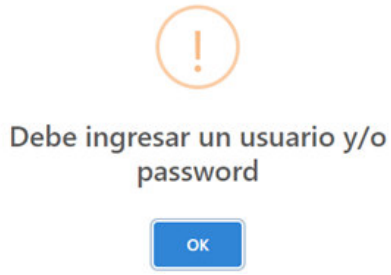


Figura 3.246 Alerta cuando el usuario trata de ingresar sin usuario y contraseña

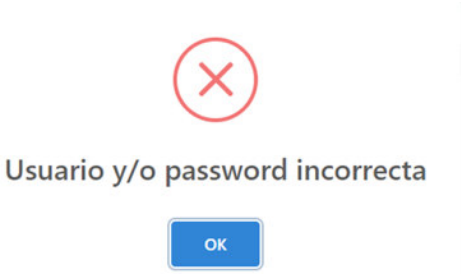


Figura 3.247 Alerta cuando el usuario ingresó mal el usuario o contraseña

La variable *jsVar* que se mencionó previamente se obtuvo de la variable retorna del *script* en *PHP* de la Figura 3.248. Para pasar esta variable desde *PHP* se imprimió el valor de la variable retorna en el código *JavaScrip*t mientras se genera la página.

```php
if(empty($_POST['usuario']) || empty($_POST['clave']))
{

    $retorna=1;

}else{

    require_once "conexion.php";

    $user = mysqli_real_escape_string($conection,$_POST['usuario']);
    $pass = md5(mysqli_real_escape_string($conection,$_POST['clave']));

    $query = mysqli_query($conection,"SELECT * FROM usuario WHERE usuario= '$user' AND clave =
    '$pass'");
    mysqli_close($conection);
    $result = mysqli_num_rows($query);

    if($result > 0)
    {
        $data = mysqli_fetch_array($query);
        $_SESSION['active'] = true;
        $_SESSION['idUser'] = $data['idusuario'];
        $_SESSION['nombre'] = $data['nombre'];
        $_SESSION['email']  = $data['email'];
        $_SESSION['user']   = $data['usuario'];
        $_SESSION['rol']    = $data['rol'];
```

Figura 3.248 *Script* para la autentificación de la página de *login*

Por otra parte, en el *script* de la Figura 3.248 en primer lugar se inicializó la sesión con la línea de código *session_start();* si la sesión se encuentra activa, es decir si un usuario ya ha ingresado este se redirigirá automáticamente a la página de inicio con el comando

header('location: vistas/pag_inicio.php'); Caso contrario, el *script* revisará que los datos de usuario y clave del formulario no estén vacíos, si lo están el *script* procederá a indicar la primera alerta creada en *SweetAlert,* colocando la variable retorna en 1.

No obstante, si los campos del formulario están llenos se establecerá la conexión con la base de datos con el comando *require_once "conexion.php";* y se procederá a enviar los datos de usuario y contraseña al servidor.

A estos datos se les agregó la línea de código *mysqli_real_escape_string*, ya que ayuda a evitar que ciertos caracteres especiales se puedan ingresar, con la finalidad de prevenir la inyección *SQL*.

Además, la contraseña se encriptó con el sistema *MD5*, ya que en la base datos se estableció este parámetro al momento de crear las tablas, tal cual se ilustra en la Figura 3.249. El algoritmo proporcionará un código asociado a la clave. De manera que, con el código generado por el algoritmo (*hash)*, la base de datos podrá comprobar que la clave sea la correcta y no haya podido llegar de manera distinta a como era originalmente.

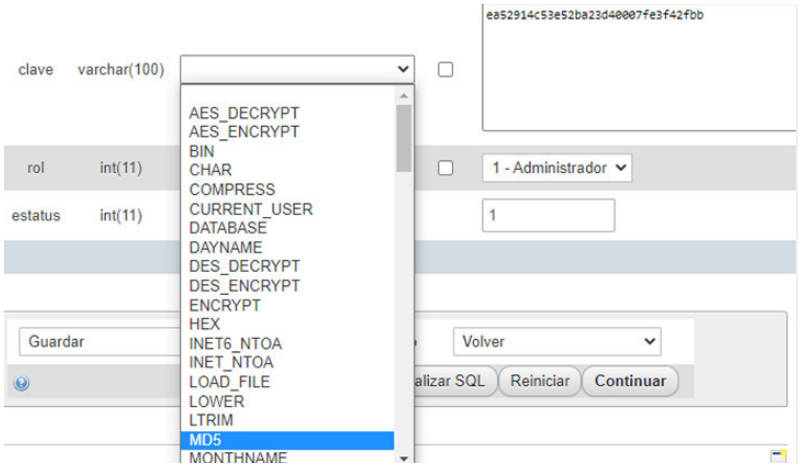


Figura 3.249 Encriptación de la clave con MD5

Si los datos del formulario son correctos, y el estatus del usuario es diferente de 0 el *script* redirigirá al usuario a la página de inicio caso contrario destruirá la sesión. Finalmente, si los datos del formulario son incorrectos, el *script* mostrará la alerta número 2 colocando la variable retorna en 3 e igualmente se destruirá la sesión.

Otra forma en que se le agregó seguridad al formulario fue a través del atributo de patrón para campos, mediante la línea de código *pattern="[A-Za-z0-9_-]{1,15}"*. Esta línea significa que el formulario solo permitirá la entrada de letras mayúsculas y minúsculas de la A-Z, guiones bajos y medios.

No obstante, los caracteres especiales como las comillas simples no se podrán ingresar, también la longitud de entrada mínima será de 1 y la máxima de 15, es decir caracteres mayores a estos no se podrán enviar a la base de datos. En la Figura 3.250 se halla lo dicho.

```
            <h5>Usuario</h5>
            <input  class="input" type="text" id="usuario"
            pattern="[A-Za-z0-9_-]{1,15}" name="usuario"  >
           <!-- utilizamos el atributo pattern para evitar ingreso de
           caracteres especiales solo los que estan en parentesis
            entraran
            La longitud minima sera de 1 y la maxima 15
            -->
        </div>
    </div>
<div class="input-div pass">
        <div class="i">
            <i class="fas fa-lock"></i>
            <!--logotipo de candado-->
        </div>
        <div class="div">
            <h5>Contraseña</h5>
            <input  class="input" type="password" id="clave"
         pattern="[A-Za-z0-9_-]{1,15}"  name="clave">
```

Figura 3.250 Agregación del atributo de patrón para campos de entrada

En la Figura 3.251 se observa cómo al ingresar caracteres especiales al formulario este impide su ingreso.

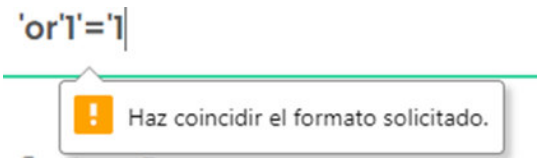


Figura 3.251 Ingreso de caracteres no permitidos

Finalmente, a la página de *login* en el archivo *CSS* se colocó los márgenes de la página, el tipo de letra, la posición de las imágenes, el tamaño del formulario, el formato de la entrada del usuario y contraseña, el estilo del botón de envío y se ajustó el diseño para que pueda funcionar en distintos tipos de pantalla.

- *Cerrar sesión*

Para finalizar la sesión del usuario se creó el *script* en *PHP* de la Figura 3.252, este *script* destruye la sesión y se redirige a la página de *login*

```php
<?php

    session_start();
    session_destroy();

    header('location: ../');

?>
```

Figura 3.252 *Script* para cerrar la sesión

- *Páginas de inicio*

Se diseñó 2 páginas de inicio, la primera página es para el usuario normal y la segunda es para el usuario de tipo administrador, dependiendo el rol del usuario este ingresará a su página correspondiente. En estas páginas se mostrará el nombre del usuario que ha ingresado y el permiso que tiene. Tal como se mira en las Figura 3.253 y Figura 3.254.

Figura 3.253 Página de inicio del usuario normal

Figura 3.254 Página de inicio del usuario administrador

Para las funciones descritas se empleó un archivo *PHP* junto con *HTML5* en el cual se colocó los mensajes, así como los botones para entrar y cerrar sesión. El botón de cerrar sesión llama al *script* de *PHP* de la Figura 3.252 con la línea de código *href="../sistema/salir.php"*, al contrario el botón entrar redirige a la interfaz web correspondiente a cada usuario con la línea de código *href="../users/index.php".*

Para la impresión del nombre del usuario que ingresó, se utilizó la siguiente línea de código <?*php echo $_SESSION['user'].' -'.$_SESSION['rol'];?>*, en esta se puede observar que se imprime la sesión con el comando *echo* y también el rol.

Para el diseño de las 2 páginas se utilizó *Bootstrap* modificando sus librerías *CSS* y J*avaScript.*

- *Páginas de control de la interfaz web*

Se elaboró 2 páginas para el control del prototipo tal cual se observa en las Figura 3.255 y Figura 3.256. La primera página podrá controlar las funcionalidades básicas del prototipo tal como se ilustra en el diagrama de flujo de la Figura 3.238.

Sin embargo, la segunda página del administrador de la Figura 3.256 además podrá agregar, buscar, editar y eliminar usuarios del prototipo domótico.

Figura 3.255 Página de control del usuario normal

Figura 3.256 Página de control del usuario administrador

Para la creación de las páginas del usuario administrador se empleó una plantilla de *Dashboard,* en esta primeramente se agregó el nombre del usuario que ingresó a la interfaz web con el comando *echo $_SESSION['user].' -'.$_SESSION['rol'];* tal cual se muestra en la Figura 3.257.

Figura 3.257 Impresión en la página de control del usuario que ingresó

Más adelante, se añadió un *script* en *PHP* para que imprima el día, el mes y el año automáticamente tal como se ilustra en la Figura 3.258.

Figura 3.258 Impresión en la página de control de la fecha

Para realizar lo dicho, se empleó el comando *date_default_timezone_set()*, en este se colocó la zona horaria y luego se ubicó en la cabecera del *Dashboard* mediante la línea de código * Quito, <?php echo fechaC();?> .* En la Figura 3.259 se encuentra el *script* encargado de realizar la obtención de la fecha.

```php
<?php
    date_default_timezone_set('America/Guayaquil');

    function fechaC(){
        $mes = array("","Enero",
                       "Febrero",
                       "Marzo",
                       "Abril",
                       "Mayo",
                       "Junio",
                       "Julio",
                       "Agosto",
                       "Septiembre",
                       "Octubre",
                       "Noviembre",
                       "Diciembre");
        return date('d')." de ". $mes[date('n')] . " de " . date('Y');
    }


 ?>
```

Figura 3.259 *Script* para obtener la fecha

Posteriormente, se agregó las demás páginas que conforman el menú de control manteniendo el diseño del *Dashboard*, para esto se utilizó las líneas de código: *<?php require_once "vistas/parte_superior.php"?>* y *<?php require_once "vistas/parte_inferior.php"?>* al comenzar una nueva página y al finalizarla respectivamente.

- *Registro de usuario*

En primer lugar, se elaboró el formulario de la Figura 3.260, este consta de los campos de nombre, correo, usuario, clave y selección del tipo de usuario.

Figura 3.260 Formulario de registro de usuarios

El formulario mostrará 4 alertas, la primera alerta indica al usuario que no puede dejar los campos vacíos, por otro lado, la segunda alerta se activa cuando ya existe otro usuario con el mismo correo o usuario, en cambio la tercera alerta es para indicar si existen errores al momento de guardar el formulario y finalmente la última alerta sirve para avisar que el nuevo usuario fue creado con éxito. En la Figura 3.261 se muestra un ejemplo de alerta.

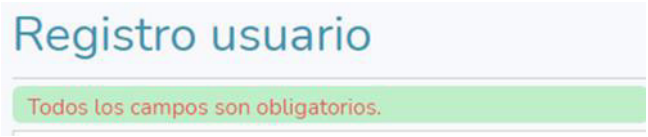


Figura 3.261 Ejemplo de una alerta

Algunos comandos empleados en esta parte son similares al formulario de *login* por lo cual, se menciona a continuación solo lo más relevante. Para insertar un nuevo usuario se utilizó el comando *$query_insert = mysqli_query ($conection,"INSERT INTO usuario (nombre,correo,usuario,clave,rol) VALUES ('$nombre','$email','$user','$clave','$rol')");*. Este comando significa que se insertará en la tabla usuario en los campos de nombre, correo, usuario, clave y rol los valores de las variables que se llenen en el formulario. Otra parte importante, es que el tipo usuario es seleccionable en el formulario de registro, para esto se usó las líneas de código de la Figura 3.262.

```
<?php

    $query_rol = mysqli_query($conection,"SELECT * FROM
    rol");
    mysqli_close($conection);
    $result_rol = mysqli_num_rows($query_rol);

 ?>

<select name="rol" id="rol">
    <?php
        if($result_rol > 0)
        {
            while ($rol =
            mysqli_fetch_array($query_rol)) {
    ?>
            <option value="<?php echo $rol["idrol"]; ?
            >"><?php echo $rol["rol"] ?></option>
    <?php
                |
            }

        }
     ?>
```

Figura 3.262 Líneas de código para seleccionar el tipo de usuario

En la Figura 3.262 se observa cómo se seleccionó el rol de Figura 3.263 y se almacenó en la variable resultado, después si la operación fue correcta se imprimió constantemente el valor del idrol en el rol para que se muestre como opción desplegable en el formulario.

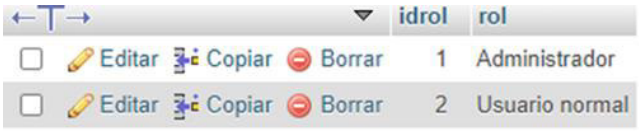


Figura 3.263 Tabla del idrol y el rol

- *Lista de usuarios*

Para la lista de usuarios se creó la tabla de la Figura 3.264, en esta tabla se ubicó los campos: *ID*, Nombre, Correo, Usuario, Rol y Acciones. En la columna acciones se colocó la opción de editar o eliminar los campos. También se añadió un buscador para encontrar a los usuarios y un paginador para que cada 5 usuarios se muestre una nueva página.

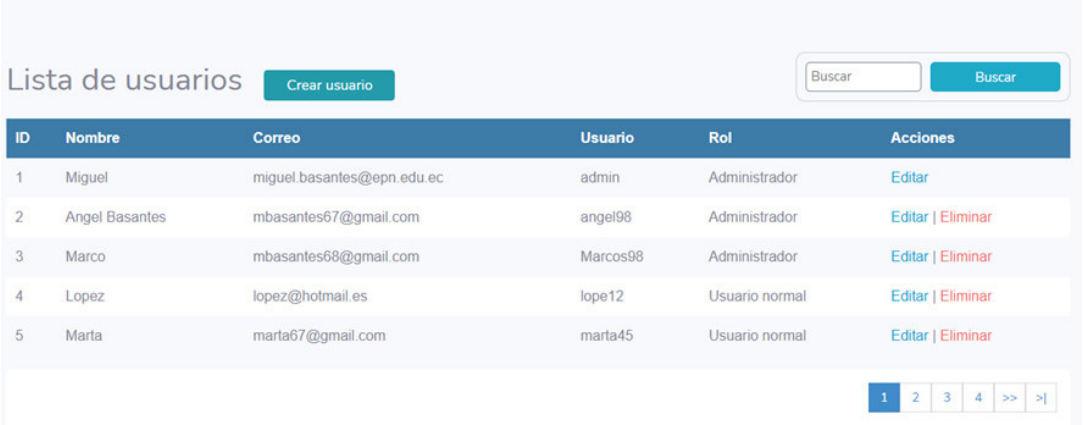


Figura 3.264 Lista de usuarios

- *Paginador*

Para el paginador primeramente se creó el *script* de la Figura 3.265. Seguidamente, se obtuvo el total de los registros existentes de la base de datos donde el estatus sea 1 mediante la línea de código *$sql_registe = mysqli_query($conection,"SELECT COUNT(*) as total_registro FROM usuario WHERE estatus = 1 ");*.

```php
<?php
    //Paginador
    $sql_registe = mysqli_query($conection,"SELECT COUNT(*) as
    total_registro FROM usuario WHERE estatus = 1 ");
    $result_register = mysqli_fetch_array($sql_registe);
    $total_registro = $result_register['total_registro'];

    $por_pagina = 5;

    if(empty($_GET['pagina']))
    {
        $pagina = 1;
    }else{
        $pagina = $_GET['pagina'];
    }

    $desde = ($pagina-1) * $por_pagina;
    $total_paginas = ceil($total_registro / $por_pagina);

    $query = mysqli_query($conection,"SELECT u.idusuario,
    u.nombre, u.correo, u.usuario, r.rol FROM usuario u INNER
    JOIN rol r ON u.rol = r.idrol WHERE estatus = 1 ORDER BY
    u.idusuario ASC LIMIT $desde,$por_pagina
        ");

    mysqli_close($conection);

    $result = mysqli_num_rows($query);
    if($result > 0){

        while ($data = mysqli_fetch_array($query)) {

    ?>
```

Figura 3.265 *Script* del paginador

A continuación, se almacenó el valor total de registros en un arreglo y se generó una variable para que por página se muestre 5 registros.

Acto seguido, se evaluó el valor que se está enviando en el paginador. En la Figura 3.266 se puede notar a modo de ejemplo que al presionar sobre el número de la página 2 se envía la variable página = 2, con esta variable se creó un condicional para que al entrar en la lista de usuarios al iniciar el paginador se sitúe en la página 1 y luego según se presione en cualquier número del paginador este cambie de página.

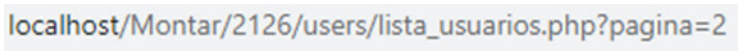


Figura 3.266 Paginador ubicado en la página número 2

En cambio con la línea de código *$query = mysqli_query($conection,"SELECT u.idusuario, u.nombre, u.correo, u.usuario, r.rol FROM usuario u INNER JOIN rol r ON u.rol = r.idrol WHERE estatus = 1 ORDER BY u.idusuario ASC LIMIT $desde,$por_pagina "*, lo que se realizó es el establecimiento de la conexión con la base de datos y la selección de los datos de los usuarios con un límite ascendente establecido por las variables desde y por_pagina.

- *Buscador*

Para el buscador, primeramente, se creó un formulario para el envío de la palabra que se está buscando a la base de datos tal como se ilustra en la Figura 3.267.

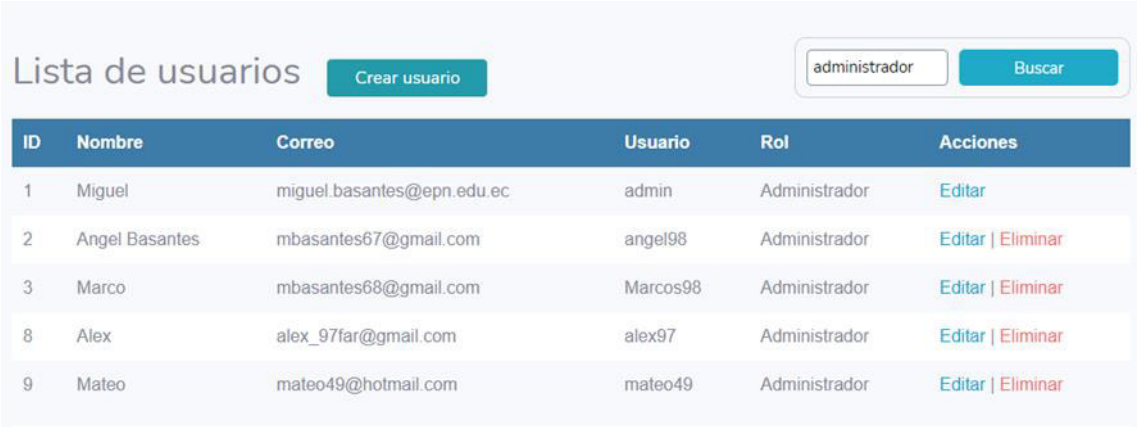


Figura 3.267 Formulario para el envío de la palabra que se está buscando

Luego, tal como se muestra en la Figura 3.268 se estableció la conexión con la base de datos y por medio de la función *LIKE* se buscó la variable búsqueda en la tabla usuario en la base de datos, siempre y cuando el estatus del usuario sea 1, es decir que esté activo.

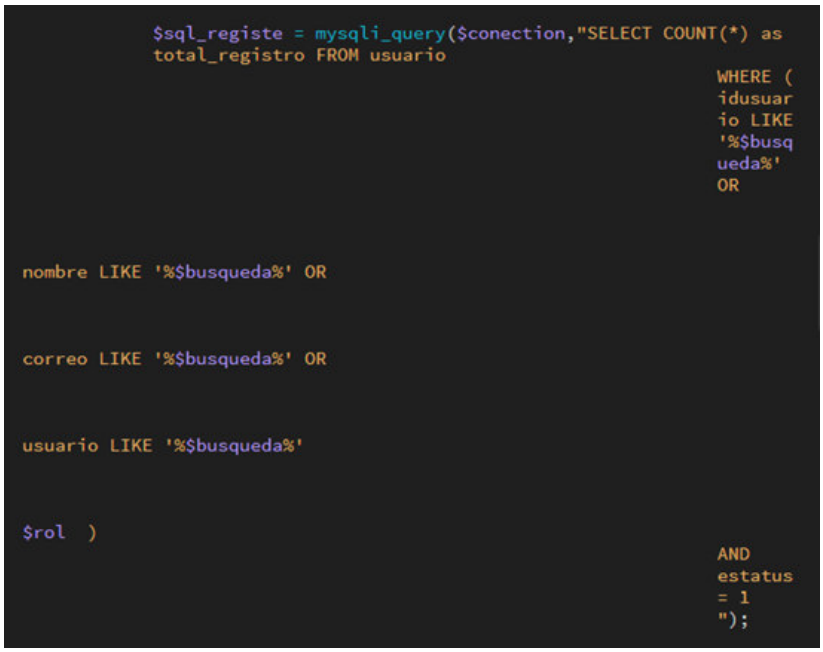


Figura 3.268 Función para buscar una palabra en la base de datos

Después, se estableció la función de la Figura 3.269, para que cuando se realice la búsqueda solo aparezca el número de registro resultante. Más tarde, esto también se aplicó al paginador de modo que si solo se encuentra un registro se tendría una página, si hay más de 5 registros en donde coincida la búsqueda se mostrarán en otra página y así sucesivamente.

```
$query = mysqli_query($conection,"SELECT u.idusuario,
u.nombre, u.correo, u.usuario, r.rol FROM usuario u INNER
JOIN rol r ON u.rol = r.idrol
                        WHERE
                        ( u.idusuario LIKE
                        '%$busqueda%' OR
                            u.nombre LIKE '%$busqueda%'
                            OR
                            u.correo LIKE '%$busqueda%'
                            OR
                            u.usuario LIKE
                            '%$busqueda%' OR
                            r.rol    LIKE
                            '%$busqueda%' )
                        AND
                        estatus = 1 ORDER BY
                        u.idusuario ASC LIMIT
                        $desde,$por_pagina
    ");
```

Figura 3.269 Función para mostrar los registros resultantes de la búsqueda

- *Editar usuario*

Para la actualización de un usuario, se elaboró un formulario similar al de registro de usuario tal como se presenta en la Figura 3.270. Este formulario se muestra al presionar sobre el botón editar en la columna acciones tal cual se observa en la Figura 3.271.

Figura 3.270 Formulario de actualización de usuario

Figura 3.271 Edición de un usuario al presionar el botón editar

Una vez que se modifican los datos y envían con el botón actualizar usuario, en la parte superior se mostrarán algunas alertas. La primera alerta indicará que la actualización fue exitosa, la segunda si los datos de correo o usuario ya existen y en cambio la tercera si ocurrió un error al conectarse con la base de datos.

La parte que destaca en esta sección se encuentra en la Figura 3.272, en esta se nota como se estableció la conexión con la base de datos y con la función *update* se actualizó las columnas de nombre, correo, usuario, clave y rol de la tabla usuario.

```
$sql_update = mysqli_query($conection,"UPDATE
usuario
                                        SET nombre
                                        =
                                        '$nombre',
                                        correo='$em
                                        ail',usuari
                                        o='$user',c
                                        lave='$clav
                                        e',
                                        rol='$rol'
                                        WHERE
                                        idusuario=
                                        $idUsuario
                                        ");
```

Figura 3.272 Función para actualizar un usuario de la base de datos

- *Borrar usuario*

Para borrar un usuario en la tabla lista de usuarios en la columna acciones se colocó el botón eliminar. Cuando se presiona sobre este botón aparecerá una ventana similar a la Figura 3.273. En esta ventana se colocó 2 botones aceptar y cancelar. Si se elige la opción cancelar redirigirá nuevamente a la tabla de usuarios, en cambio con la opción aceptar se enviará un formulario con el método *POST* para proceder a la eliminación del registro.

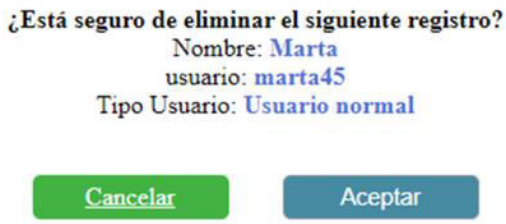


Figura 3.273 Ventana para eliminar el registro de un usuario

En la Figura 3.274 se observa el *script* que se diseñó para eliminar un usuario. En primer lugar, con el comando *$query_delete = mysqli_query($conection,"UPDATE usuario SET estatus = 0 WHERE idusuario = $idusuario ");* se actualizó el estatus del usuario a 0 para darle de baja, de manera que en la tabla de usuarios ya no se mostrará el usuario eliminado, ni este podrá *loguearse* en la interfaz web, pero se conservarán los datos en *phpMyAdmin* tal como se ilustra en la Figura 3.275. Por otro lado, si lo que se desea es eliminar el usuario de la base de datos definitivamente, emplear el comando *$query_delete = mysqli_query($conection,"DELETE FROM usuario WHERE idusuario =$idusuario ");*.

```php
<?php
    session_start();
    if($_SESSION['rol'] != 1)
    {
        header("location: ./");
    }
    include "../conexion.php";

    if(!empty($_POST))
    {
        if($_POST['idusuario'] == 1){
            header("location: lista_usuarios.php");
            mysqli_close($conection);
            exit;
        }
        $idusuario = $_POST['idusuario'];

        //$query_delete = mysqli_query($conection,"DELETE FROM usuario
        WHERE idusuario =$idusuario ");
        $query_delete = mysqli_query($conection,"UPDATE usuario SET
        estatus = 0 WHERE idusuario = $idusuario ");
        mysqli_close($conection);
        if($query_delete){
            header("location: lista_usuarios.php");
        }else{
            echo "Error al eliminar";
        }

    }
```

Figura 3.274 *Script* para borrar un usuario

☐	Editar	Copiar	Borrar	4	Lopez	lopez@hotmail.es	lope12	ea52914c53e52ba23d40007fe3f42fbb	2	1
☐	Editar	Copiar	Borrar	5	Marta	marta67@gmail.com	marta45	ea52914c53e52ba23d40007fe3f42fbb	2	0

Figura 3.275 Actualización del estatus de un usuario

- *Ejecución de scripts*

Para la ejecución de todos los *scripts* desde la interfaz web se empleó la estructura que se detalla a continuación:

En primer lugar, se creó varias casillas de verificación con los identificadores y clases correspondientes. En la Figura 3.276 se muestra el código empleado para crear una casilla de verificación. Luego por medio de código *CSS* se les dio diseño, estilo y animación en forma de botón o *switch*. En la Figura 3.280 se aprecia a modo de ejemplo el diseño de 2 casillas de verificación que se utilizan en el sistema de la puerta.

```html
<label id="lab8">
 <input type="checkbox" id="switch8" name="switch8">
 <span class="btn8"></span>

 <i class="fa fa-power-off"></i>
 </label>
```

Figura 3.276 Creación de una casilla de verificación en *HTML5*

Más tarde, se agregó funcionalidad a cada casilla de verificación. En la Figura 3.277 se puede notar el código para dar funcionalidad a una casilla de verificación.

En el *script* se observa que si la casilla está marcada mostrará una determinada imagen y también un texto en algunos casos, en cambio si no está marcada los datos mostrados serán otros.

Sin embargo, la principal función es para, que al marcar y desmarcar la casilla por medio de *Ajax* a través del método *POST* se envíe el valor fijado de la variable estado a un archivo de *PHP* para que se ejecute el *script* de *Python* deseado sin necesidad de recargar la página.

```
<script>
    $("#switch8").change(function()
    {
        var checkeado=$("#switch8").attr("checked");
        if(checkeado) {

            $("#switch8").removeAttr("checked");

            console.log("estado 17");
            var estado=17;
            $("#campo3").text("Automático Apagado");
            $("#campo3").removeClass('alert alert-info').addClass( "alert alert-danger" );
            $("#imagen_bombillo8").html('<img src="../images/img5.jpg" alt="Album Cover" clas

            $.ajax({
                data:{valor_estado: estado},
                url:'../procesa.php',
                type:'POST',
                success: function(response){
                }
            });

        } else {
            $("#switch8").attr("checked","checked");
            console.log("estado 18");
            var estado=18;
            $("#campo3").text("Automático Encendido");
            $("#campo3").removeClass('alert alert-danger').addClass( "alert alert-info" );
            $("#imagen_bombillo8").html('<img src="../images/img6.jpg" alt="Album Cover" clas
```

Figura 3.277 *Script* para agregar funcionalidad a la casilla de verificación

En las Figura 3.278 y Figura 3.279 se destacan algunos de los *scripts* de *PHP* que recibirán el valor de la variable estado.

procesa.php

```
1  <?php
2  $valor_estado = $_POST['valor_estado'];
3
4  if($valor_estado==1)
5  {
6      exec('sudo python /var/www/html/cgi-enabled/abrir.py');
7
8
```

Figura 3.278 *Script* que ejecuta los *scripts* de *Python* por medio de la función *if*

```php
<?php
$valor_estado=$_POST['valor_estado'];
switch ($valor_estado) {
    case 1:
        exec('sudo python /var/www/html/cgi-enabled/motor112.py');
        break;
    case 5:
        exec('sudo python /var/www/html/cgi-enabled/motor113.py');
        break;
```

Figura 3.279 *Script* que ejecuta los *scripts* de *Python* con la función *switch*

En la Figura 3.280 se puede visualizar de mejor manera todo lo dicho, ya que en la imagen se muestra como al presionar el botón deslizante hacia la derecha en el sistema de la puerta el programa envía el estado 1 y ejecuta el *script* de *Python* para abrir la puerta, por el contrario, cuando se presiona el botón deslizante hacia la izquierda se envía el estado 2 y se ejecuta el *script* para cerrar la puerta.

Este proceso se realizó con todos los botones que tiene la interfaz web solo que con otros valores de la variable estado. Cabe recalcar que todos los *scripts* de *Python* que se ejecutan son los mismos que se diseñaron en las secciones anteriores.

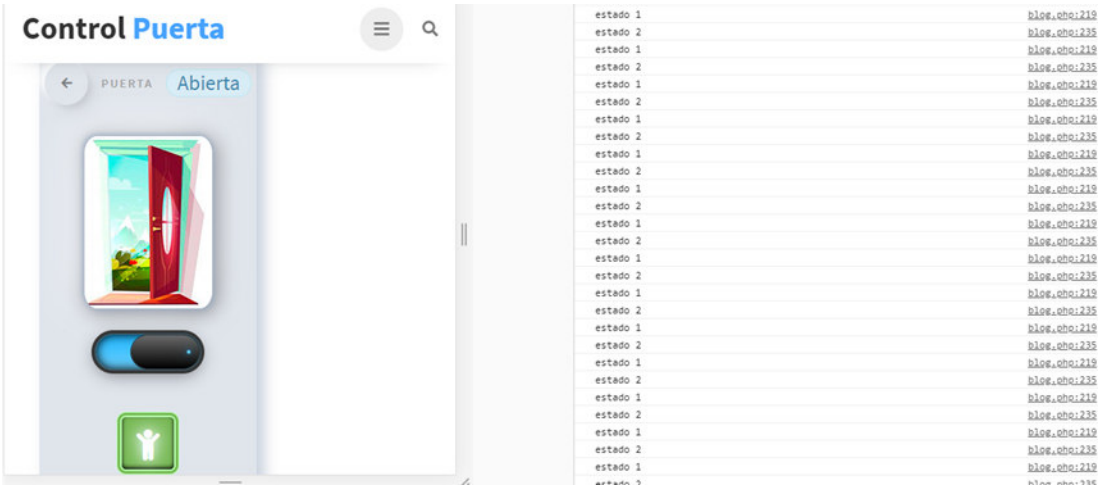


Figura 3.280 Impresión en consola de los estados al pulsar un botón

- *Lectura de los archivos de texto y representación de los datos en la interfaz*

Algunos de los *scripts* de *Python* que se crearon imprimen automáticamente el valor de los sensores o mensajes importantes en un archivo de texto. Estos datos indican el funcionamiento actual del prototipo y se representaron en la interfaz web de la siguiente manera:

Inicialmente, el archivo de texto se colocó en un archivo de *HTML5* en forma de *script* tal como se ilustra en la Figura 3.281.

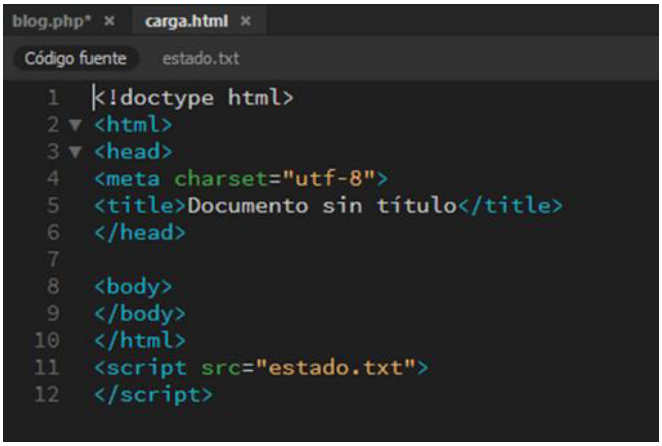


Figura 3.281 Archivo de texto en forma de *script*

Seguidamente, por medio de la función *setInterval ()* se cargó el archivo de *HTML5* en un determinado tiempo tal cual se observa en la Figura 3.282. Esta función continuará llamando a la función *cargarDatos* hasta que se llame a la función *clearInterval ()* o se cierre la ventana. Esto es muy útil ya que, si se coloca un tiempo adecuado, los datos cargados serán lo más cercano a la realidad.

```
setInterval(function(){
$("#div1").load("carga.html");//cargo el archivo estado.txt
cargarDatos();//Imprimo los datos
},16);//60 fps 60 veces por segundo velocidad de un juego normal 1000ms equivalen a 1 segundo
1000/60=16.666
function cargarDatos(){

//console.log("cargando");
document.getElementById("campo1").innerText=estad;
        $("#campo1").addClass( "alert alert-info" );
}
```

Figura 3.282 Función para cargar los datos a la interfaz web

Para determinar este valor se debió tener en cuenta que la velocidad está determinada por los *Frames* por segundo (fps) y que el cerebro humano a partir de 12 (fps), empieza a ver una consecución de fotos como una imagen en movimiento por lo cual el valor más adecuado era 60 (fps) que es valor que se utiliza en un juego normal [64].

Por lo tanto, si 1000 (ms) equivalen a 1 segundo y los datos se cargarán 60 veces en un segundo se obtuvo que 16 (ms) es el tiempo óptimo. En contraste, si se desea ejecutar la función solo una vez, después de un número específico de milisegundos, se puede optar por el método *setTimeout ()*.

- *Control por comandos de voz*

Primeramente, se diseñó la entrada en donde se mostrará el texto ingresado por la voz y también se colocó las imágenes que aparecerán dependiendo de la función que se esté llevando a cabo tal como se ilustra en la Figura 3.283.

```
            <h2 class="titulo">Control por voz</h2>
                <div class="voz">
 <div   id="voces"  >
   <input name="q" id="q" type="text"  placeholder="Habla ahora..." autocomplete="off" autofocus>
   <!-- <button type="button"><i class="fas fa-microphone"></i></button> -->
 </div>
 <p class="info"></p>
</div>

            <div class="contenido-textos">


                 <div id="bombi"><img src="../images/img3.jpg"></div>
                 <div id="bombi1"><img src="../images/img4.jpg"></div>
```

Figura 3.283 Diseño de la entrada para la voz

Después se elaboró un *script* de *JavaScript*, en el cual se integró *Web Speech API* con los comandos propios de la aplicación. Principalmente lo que se hizo fue crear un botón en forma de micrófono y darle funcionalidad, de modo que si se presiona sobre este empieza a funcionar el reconocimiento de voz. En cambio, si no se presiona el botón el reconocimiento de voz para. Lo mencionado se encuentra en la Figura 3.284

```
const micBtn = searchForm.querySelector("button");
const micIcon = micBtn.firstElementChild;

micBtn.addEventListener("click", micBtnClick);
function micBtnClick() {
  if(micIcon.classList.contains("fa-microphone-slash")) {
    recognition.start();
  }
  else {
    recognition.stop();
  }
}
```

Figura 3.284 Funcionalidad del botón micrófono

Por otra parte, también se dio funcionalidad a ciertas palabras. Por ejemplo, con la palabra parar, el reconocimiento de voz se detiene, al contrario, con la palabra borrar se borra la entrada del texto ingresado por la voz. En las Figura 3.285 y Figura 3.286 se destaca lo dicho.

Después, la palabra que *Web Speech API* reconoce se guardó en un variable. Más tarde, a través de *POST* y por medio de *Ajax* se envió esta variable al *script* de *PHP* de la Figura 3.287 tal como se observa en la Figura 3.285, con el propósito de verificar la variable con las palabras almacenadas y ejecutar el *script* de *Python* correspondiente.

Si este proceso es exitoso el *script* de *PHP* devolverá un valor a la función *Ajax* que servirá para mostrar la imagen y texto correspondiente al *script* al usuario en forma de animación. Sin embargo, si la palabra no coincide con ningún valor se mostrará un mensaje al usuario indicando que siga intentando. Finalmente, todo el proceso descrito anteriormente, se llevó a cabo para cada parte que compone el sistema con diferentes comandos de voz que activan sus correspondientes *scripts* de *Python*.

```
if(transcript.toLowerCase().trim()==="parar") {
  recognition.stop();
}
else if(!searchFormInput.value) {
  searchFormInput.value = transcript;
    var valor = transcript;
    $("#caja").text(valor);
        $.ajax({
    type: "POST",
    url: "procesa.php",
    data: "valor="+valor,
    success: function(datos){
    console.log( "Salida: " + datos);
      switch(datos)
      {
      case "1":
         $("#bombi1").hide('slow');
         $("#bombi").show('slow');
         $("#caja").text('Abierto');
         console.log("Abierto");
        break;
```

Figura 3.285 Función para enviar los datos a través de *POST* mediante *Ajax*

```
else {
 if(transcript.toLowerCase().trim()==="borrar") {
    searchFormInput.value = "";
  }
  else {
    searchFormInput.value = transcript;
  }
```

Figura 3.286 Función para borrar la entrada del texto ingresado por la voz

```
<?php
$valor=$_POST['valor'];

if($valor=='abrir' || $valor=='activar' || $valor=='abre' || $valor=='abrir puerta')
{
 exec('sudo python /var/www/html/cgi-enabled/cerrar.py');
    $retorna=1;
}
```

Figura 3.287 *Scripts* de *Python* por medio de comandos de voz

3.4 Implementación de los elementos que componen el sistema

Implementación de las placas electrónicas

Después del diseño de las placas PCB en *Fritzing* se procedió a imprimir los archivos de las pistas conductoras y la señalización de los componentes electrónicos. Más adelante, se planchó la baquelita junto con la impresión. A continuación, se quemó la baquelita con cloruro férrico. Seguidamente, se quitó los residuos con una lija fina y *thinner*.

Posteriormente, se procedió a realizar los orificios para encajar los elementos electrónicos, luego se midió continuidad entre las pistas para constatar que no existiera ningún error. Finalmente, con un cautín y estaño se procedió a soldar los componentes, el resultado final de las placas se observa en la Figura 3.288.

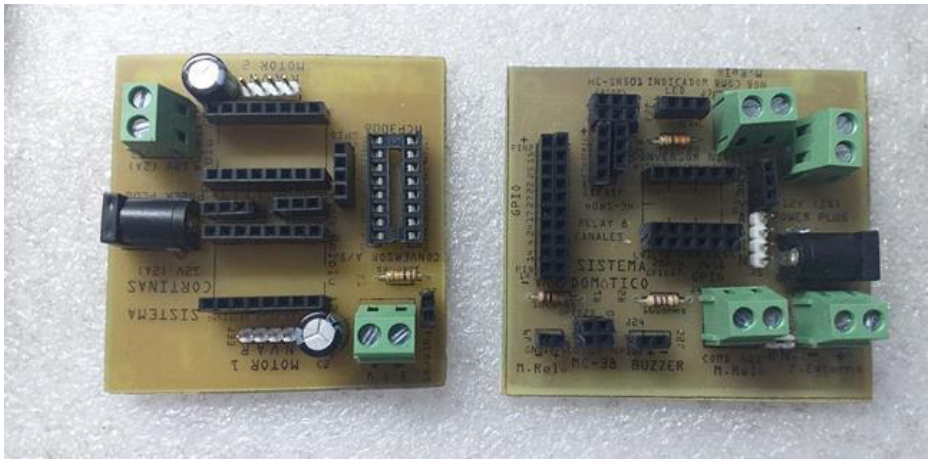


Figura 3.288 Implementación de las placas electrónicas

Impresión 3D de los componentes y la estructura del prototipo

En primer lugar, los archivos .stl elaborados en el programa *Fusion* 360 se exportaron a la impresora 3D. Luego se ubicaron en el programa de la impresora de acuerdo con el tamaño de la bandeja con la finalidad de que no se comentan errores al momento de la impresión. En la Figura 3.289 se muestra a modo de ejemplo, la ubicación de los componentes que conforman el sistema de las cortinas.

A continuación, se imprimió las piezas empleando el material PLA. Sin embargo, los tipos de relleno difieren ya que para los componentes que forman el sistema de las cortinas como es el caso de los engranajes se empleó un relleno del 70% debido al trabajo que van a realizar por lo cual requieren mayor resistencia.

Por otra parte, para la estructura del prototipo y demás se ocupó un relleno del 30%.

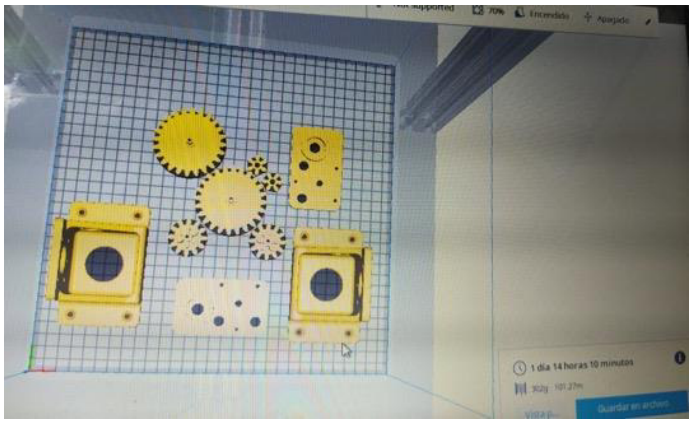


Figura 3.289 Ubicación de los componentes en la impresora

Finalmente, después de algunos días de impresión se obtuvo todas las piezas del proyecto tal cual se aprecia en las Figura 3.290, Figura 3.291 y Figura 3.292.

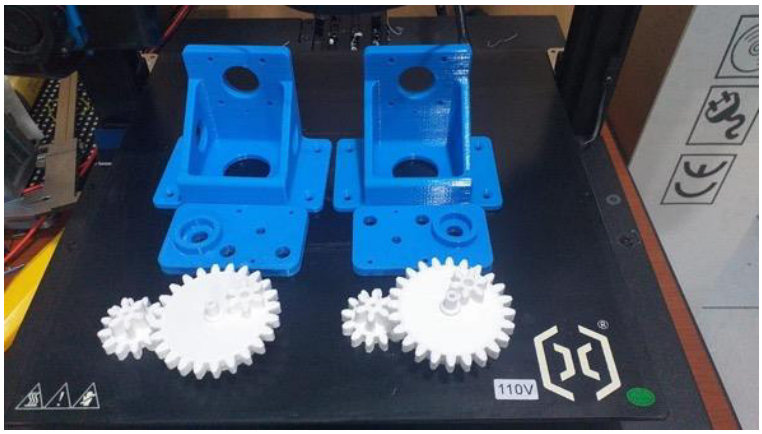


Figura 3.290 Piezas del sistema de las cortinas

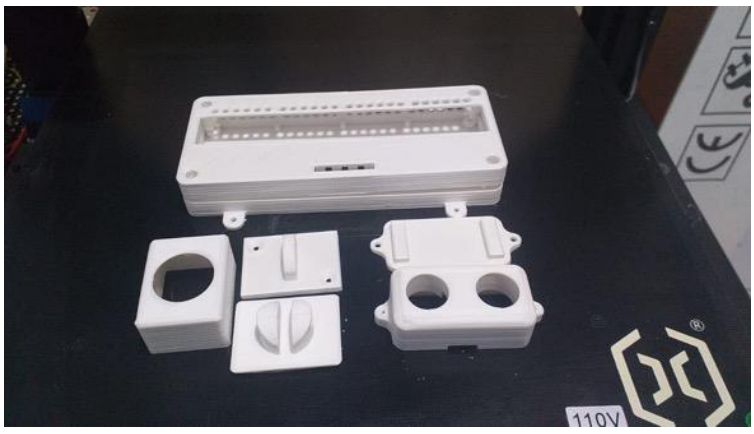


Figura 3.291 Piezas para los sensores y el módulo relé

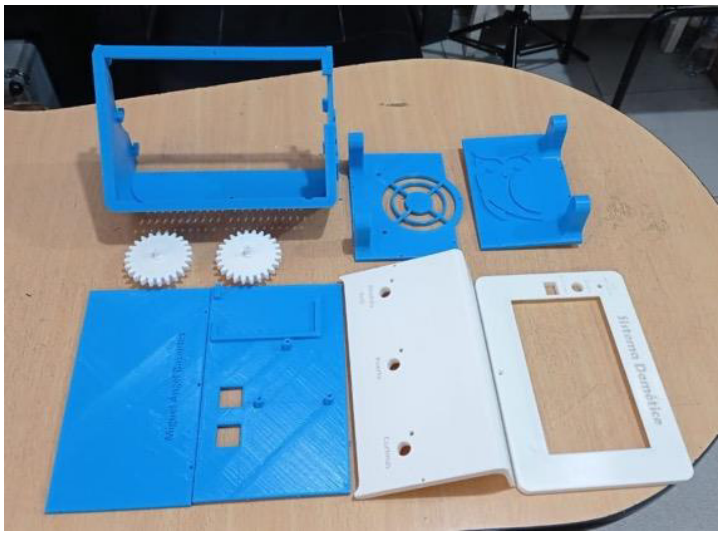


Figura 3.292 Piezas para la estructura del prototipo

Armado de las piezas impresas

Antes de nada, se armó la estructura del prototipo con las piezas de la Figura 3.292 por medio de tornillos que tenían las siguientes características: el diámetro de la cabeza de 0,5 (cm), el diámetro del cuerpo 0,3 (cm), altura de la cabeza de 0,1 (cm) y altura del cuerpo 2 (cm). En la Figura 3.293 se puede apreciar el armado de la estructura del prototipo.

Figura 3.293 Armado de la estructura del prototipo

Seguidamente, se armó la estructura que contendrá al sensor ultrasónico con las piezas de la Figura 3.291. Para esto primero se ingresó el sensor y se presionó las piezas para que encajen, además se colocó espadines hembra sobre los pines del sensor para poder conectar fácilmente este con los circuitos electrónicos. En la Figura 3.294 se encuentra armada la estructura del sensor.

Figura 3.294 Armado de las piezas que contendrán al sensor ultrasónico

Luego, se armó la estructura que contendrá al módulo relé con las piezas de la Figura 3.291. Para esto se ingresó el dispositivo en la estructura creada y se ajustó por medio de 4 tornillos con las siguientes características: el diámetro de la cabeza de 0,5 (cm), el diámetro del cuerpo 0,3 (cm), altura de la cabeza de 0,1 (cm) y altura del cuerpo 1 (cm). En la Figura 3.295 se aprecia el armado de la estructura del módulo.

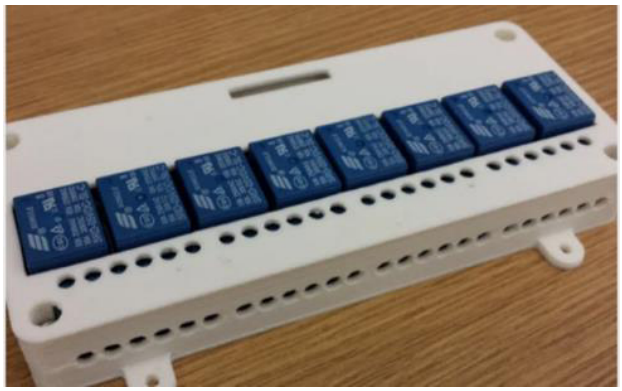


Figura 3.295 Armado de las piezas que contendrán al módulo relé

Acto seguido, con las piezas de la Figura 3.291 se armó la estructura que contendrá al sensor *PIR*. Para esto se ingresó el componente en la estructura creada y se presionó las piezas para ajustarlas una sobre otra. También fue necesario emplear un tornillo de las mismas dimensiones de la estructura del prototipo y una tuerca para solventar el movimiento del sensor dependiendo en donde se vaya a ubicar. En la Figura 3.296 se visualiza la estructura que contiene el sensor *PIR*.

Figura 3.296 Armado de las piezas que contendrán al sensor *PIR*

Finalmente, se armó el sistema que permite el movimiento a las 2 cortinas con las piezas de la Figura 3.290. Para esto se ingresó los 2 motores Nema 17 en el soporte y se ajustó con tornillos de las mismas dimensiones de la estructura del prototipo.

A continuación, se colocaron las placas, en donde van montados los engranajes y sobre estas los rodamientos 626ZZ. Más tarde, se ubicaron todos los engranajes, para esto se empleó un pedazo pequeño de lija para ajustar a los que iban en el motor, en cambio para los engranajes de soporte se utilizó tornillos con las siguientes características: el diámetro de la cabeza de 0,4 (cm), el diámetro del cuerpo 0,2 (cm), altura de la cabeza de 0,1 (cm) y altura del cuerpo 3 (cm). Por otro lado, los engranajes que son los encargados de mover el sistema se instalaron sobre los rodamientos a presión además para que no se salgan se ajustó con tornillos de las mismas dimensiones que se ocuparon en la estructura del prototipo y tuercas.

Puede ser útil también lijar un poco las piezas para quitar algunos vestigios que quedan de la impresión y engrasar los engranajes para que estos corran de mejor manera y no se traben.

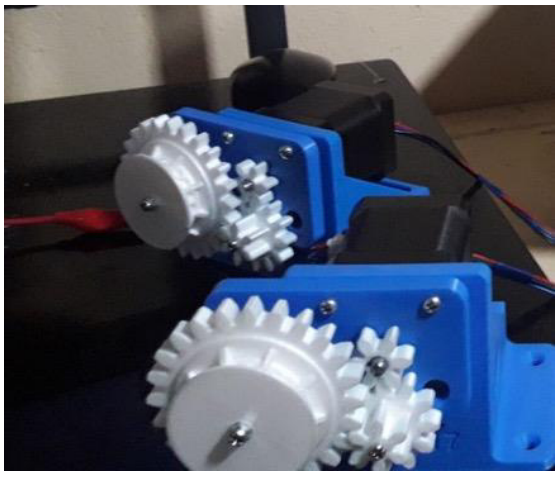


Figura 3.297 Armado de las piezas del sistema de las cortinas

Cabe resaltar que la mayor parte de las estructuras que se armaron tienen el propósito de aislar la parte electrónica y eléctrica del usuario. Además, permitir una fácil manipulación para el mantenimiento como es el caso del módulo relé. Por otra parte, las demás piezas armadas cumplen una función específica como dar movimiento a las cortinas o permitir la movilidad al sensor.

Colocación de los componentes electrónicos en la estructura del prototipo

Una vez que la estructura del prototipo se encontraba armada, se colocó sobre esta primeramente la pantalla *7inch HDMI LCD (C)*, luego el ventilador de *PC* se ajustó a la estructura por medio de 4 tornillos con las siguientes medidas: el diámetro de la cabeza de 0,4 (cm), el diámetro del cuerpo 0,2 (cm), altura de la cabeza de 0,1 (cm) y altura del cuerpo 4 (cm) y tuercas. En la Figura 3.298 se muestra lo dicho.

Figura 3.298 Colocación de la pantalla *7inch HDMI LCD (C)* y el ventilador

Después, se agregó la primera placa con tornillos de las mismas dimensiones que se ocuparon en la estructura y se conectó los componentes a través de cables para *protoboard* tal cual se ilustra en la Figura 3.299.

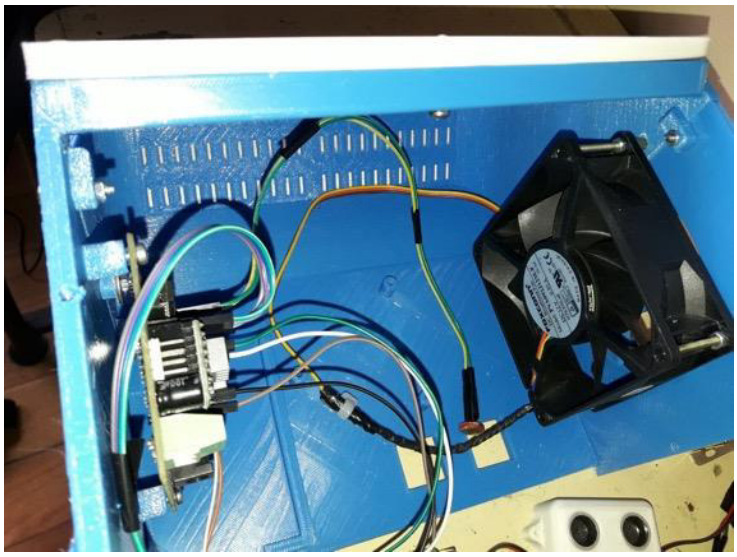


Figura 3.299 Colocación de la placa que controla los motores

Más adelante, se ubicó la *Raspberry Pi* y la segunda placa tal como se ve en la Figura 3.300. Además, se dio conexión a las placas, módulos, pantalla, ventilador, fuente de alimentación y otros componentes con la *Raspberry Pi*.

Figura 3.300 Colocación de la *Raspberry Pi* y la segunda placa

Posteriormente, en la parte delantera de la estructura se colocó el led indicador, el *buzzer* y el receptor infrarrojo. Finalmente se elaboró un cable *patch cord* para que el prototipo cuente con conexión cableada *Ethernet*. En la Figura 3.301 se muestra lo mencionado previamente, cabe indicar que el cable utilizado fue categoría 5e y la longitud del cable era de 3m además que se ponchó los RJ45 empleando la asignación de pines T568A y T568B.

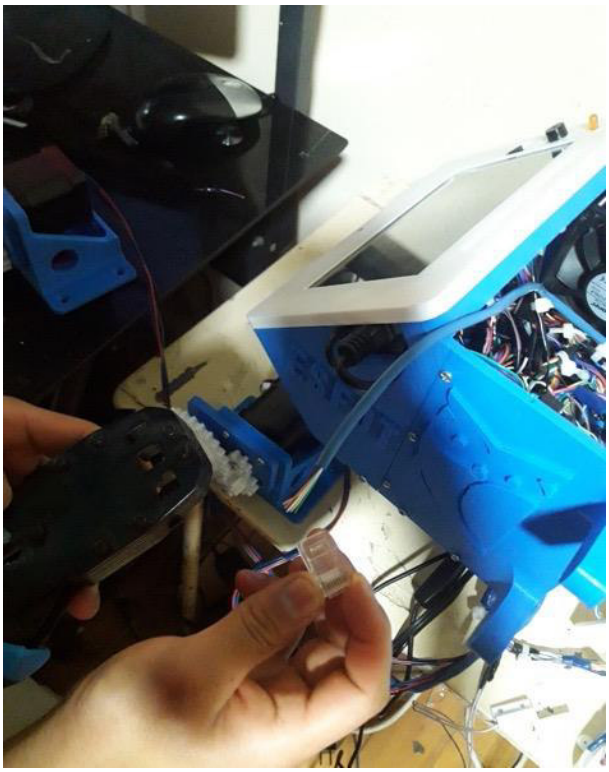


Figura 3.301 Ponchado de cable UTP categoría 5e

Implementación del sistema de cortinas

Para implementar el sistema de las cortinas se adquirió 2 persianas enrollables de 64 (cm) de alto y 80 (cm) de largo con cuerda de bolas de 6 (mm) de plástico. Tal como se muestra en la Figura 3.302.

Figura 3.302 Cortinas enrollables

A continuación, se elaboraron 4 soportes para las dos cortinas con el fin de colocarlas en la pared fácilmente. Los soportes se crearon en madera y se les dio color con pintura blanca tal cual se aprecia en la Figura 3.303.

Figura 3.303 Soportes para las cortinas

Más tarde, se fijó los soportes en la pared con las persianas justo como se muestra en la Figura 3.304. Entonces se probó las persianas manualmente para comprobar que estuvieran correctamente fijadas y el sistema corriera adecuadamente.

Figura 3.304 Instalación de las persianas enrollables

Asimismo, se perforó la pared teniendo en cuenta las dimensiones del soporte diseñado para el motor y que la cuerda de la cortina estuviera lo suficientemente templada. Sin embargo, tampoco es adecuado estirar demasiado la cuerda porque puede romperse. En la Figura 3.305 se observa que la perforación de la pared se realizó con la ayuda de un taladro.

Figura 3.305 Perforación de la pared para colocar los soportes del motor

Luego, a los huecos perforados, se les agregó varios tacos presionándolos con un martillo con el propósito de que al atornillar el sistema de las cortinas estas no se descuadren y no causen que la caja de reducción se trabe. En la Figura 3.306 se destaca la colocación del sistema de las cortinas.

Figura 3.306 Colocación del sistema de las cortinas

Finalmente, se comprobó que la instalación del sistema sea el adecuado haciendo funcionar a los motores. En la Figura 3.307 se observa la implementación del sistema en las 2 cortinas.

Figura 3.307 Implementación del sistema en las 2 cortinas

Implementación del sistema de la puerta

Con el fin de hacer más didáctica la implementación del sistema se construyó una maqueta de una puerta completamente funcional tal cual se observa en la Figura 3.308. Para esto se empleó madera y 2 bisagras.

Figura 3.308 Maqueta de una puerta completamente funcional

Tras lo anterior, se desarmó la cerradura eléctrica solenoide para variar la posición del eje del cerrojo para que coincida con la posición de la entrada de la puerta. En la Figura 3.309 se muestra lo dicho.

Figura 3.309 Cambio de la posición del eje del cerrojo

Más adelante, se ajustó la cerradura eléctrica solenoide a la puerta, así como se indica en la Figura 3.310 por medio de 4 tornillos y un desarmador.

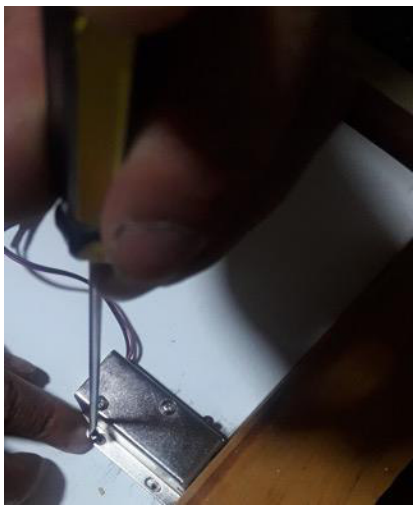


Figura 3.310 Montaje de la cerradura eléctrica solenoide en la puerta

Por último, tal cual se ilustra en la Figura 3.311 se pintó la maqueta de la puerta y se añadió a esta los sensores ultrasónico y magnético. El sensor ultrasónico se ubicó en la parte frontal de la puerta. En cambio, del sensor magnético se colocó una parte en la puerta y otra en el marco, con el fin de que al abrir la puerta el estado del sensor se modifique. Además, se realizó la conexión a los sensores por medio de canaleta.

Figura 3.311 Implementación del sistema en la puerta

Implementación del sistema de las luces

Para la implementación del sistema de las luces se ocupó la conexión eléctrica tipo puente con el propósito de encender el dispositivo tanto del interruptor como de la *Raspberry Pi*. También se utilizó 6 focos led de 9 (W), 6 conmutadores simples, 6 cajas para ubicar los conmutadores, cable calibre 12 AWG y canaleta. En la Figura 3.312 se destacan algunos de los materiales mencionados previamente.

Figura 3.312 Materiales para la implementación del sistema de las luces

A continuación, se procedió a armar en los conmutadores las líneas de fase y los retornos a los pines normalmente abiertos y normalmente cerrados del módulo relé. En cambio, en las boquillas se conectó el neutro y los retornos a los pines de común del módulo relé. En la Figura 3.313 se destaca la conexión con los componentes.

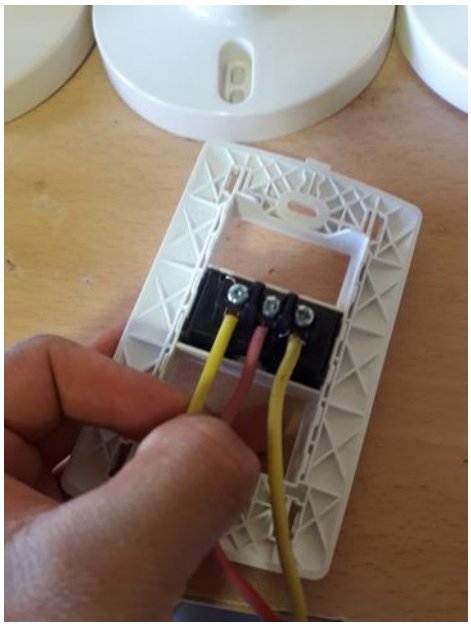


Figura 3.313 Conexión de los componentes

Más tarde, se ubicó los interruptores sobre las cajas y se ajustó con tornillos tal cual se puede apreciar en la Figura 3.314.

Figura 3.314 Ubicación de los interruptores sobre las cajas

Al final se colocó la canaleta sobre los cables para darle una mejor presentación al proyecto tal como se observa en la Figura 3.315.

Figura 3.315 Implementación del sistema de las luces

Implementación del sistema de los ordenadores

En primer lugar, se elaboró varios cables *patch cords* para que los ordenadores tengan conexión cableada *Ethernet* de la misma manera que se mostró en la Figura 3.301. Seguidamente, se realizaron varias configuraciones en los ordenadores que se detallan a continuación:

- *Configuración del encendido remoto de los ordenadores*

Tal como se indicó en el diagrama de flujo de la Figura 3.97, para el encendido de los ordenadores se ocupó el módulo *wakeonlan* por lo cual es necesario que este módulo se encuentre activo en el computador.

Para esto se ingresó en el BIOS presionando la tecla DEL (Supr) al momento del arranque, sin embargo, esto dependerá mucho del tipo de ordenador, aunque las teclas más comunes para esto son: DEL, ESC y F2. Una vez en el BIOS en la sección *Power* se activó la opción *Wake Up by Integrated LAN* y se guardó los cambios.

Seguidamente, en el computador se entró a la ventana Propiedades de *Ethernet* tal cual se muestra en la Figura 3.316 y en la sección Funciones de red se pulsó sobre la opción Configurar. Luego se desplegó la ventana de la Figura 3.317, en esta se seleccionó la pestaña Opciones avanzadas y se habilitó *Wake on magic packet.*

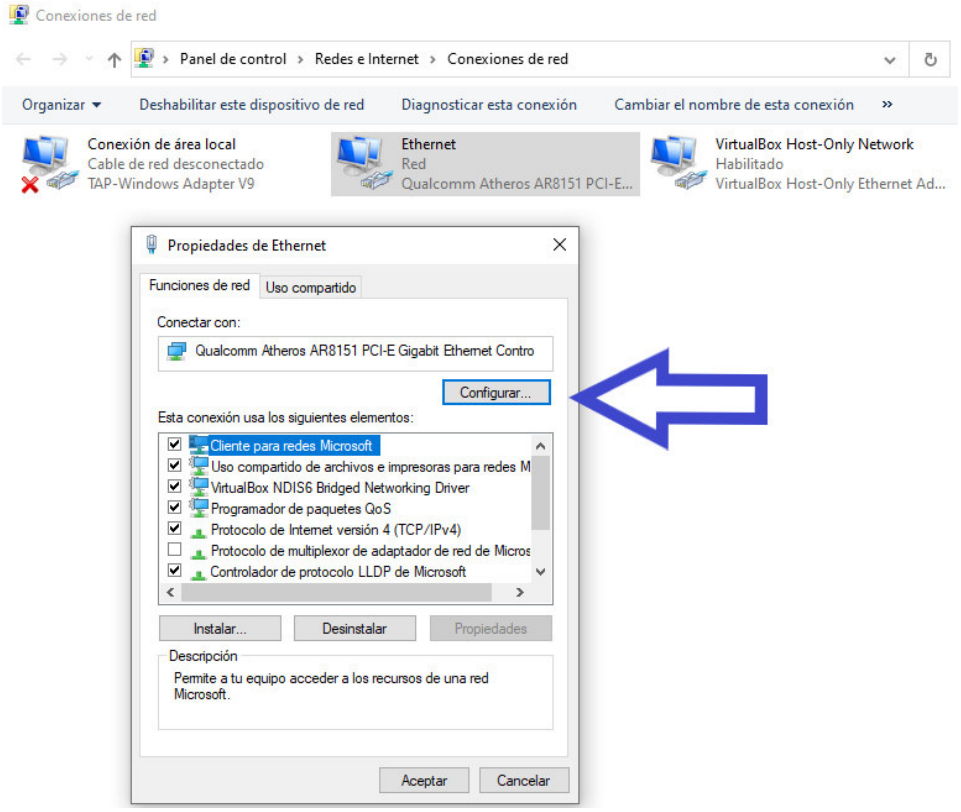


Figura 3.316 Propiedades de *Ethernet*

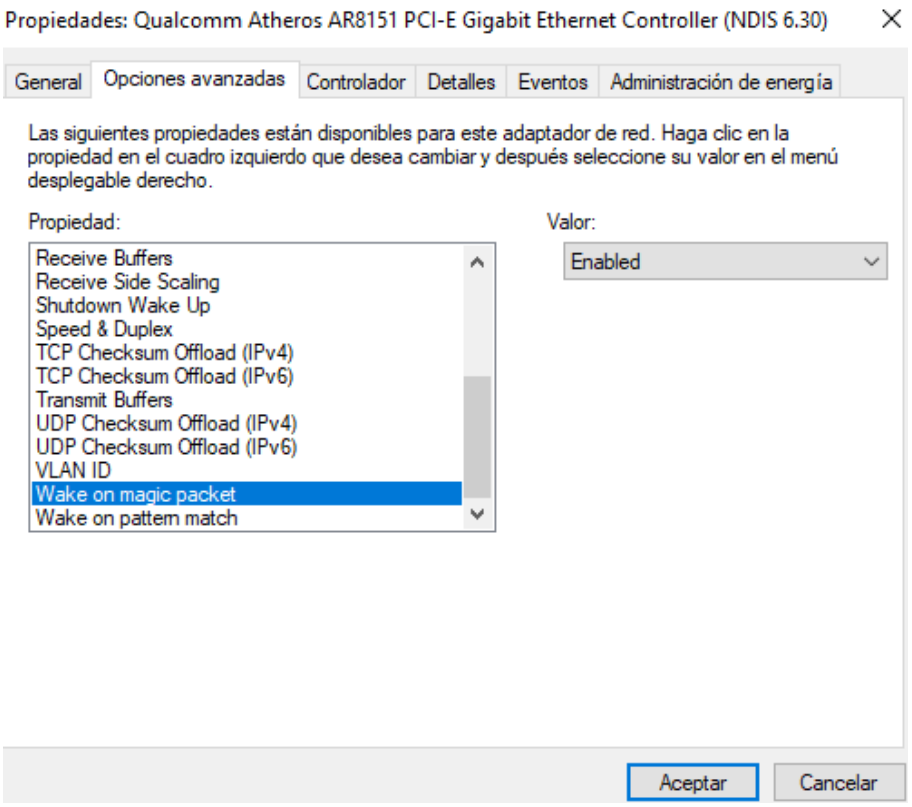


Figura 3.317 Habilitación de *Wake on magic packet*

Luego, en la pestaña Administración de energía, se marcó la casilla Permitir solo un *Magic Packet* para reactivar el equipo justo como se indica en la Figura 3.318 y se guardó la configuración pulsando sobre el botón aceptar.

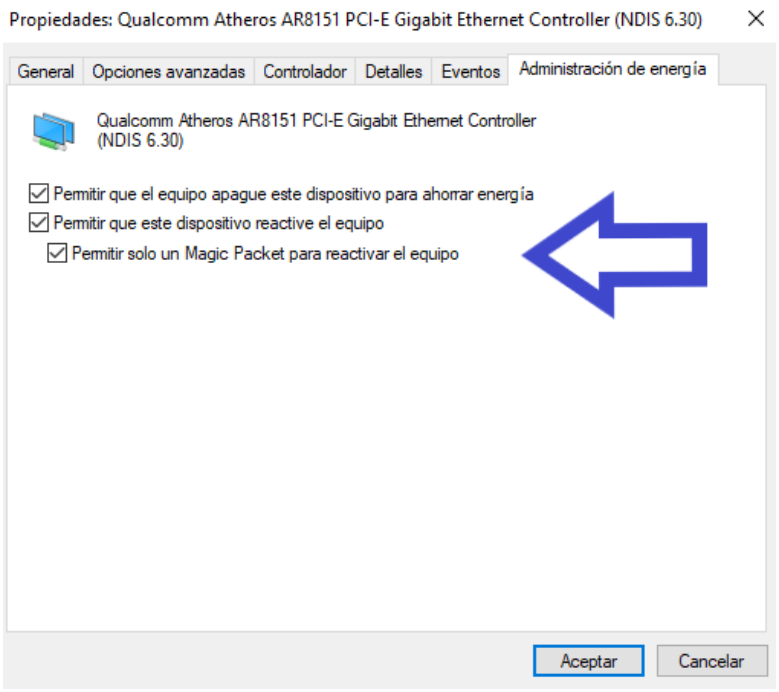


Figura 3.318 Permitir solo un *Magic Packet* para reactivar el equipo

Por último, se deshabilitó el inicio rápido al desmarcar la casilla Activar inicio rápido (recomendado) así como se visualiza en la Figura 3.319.

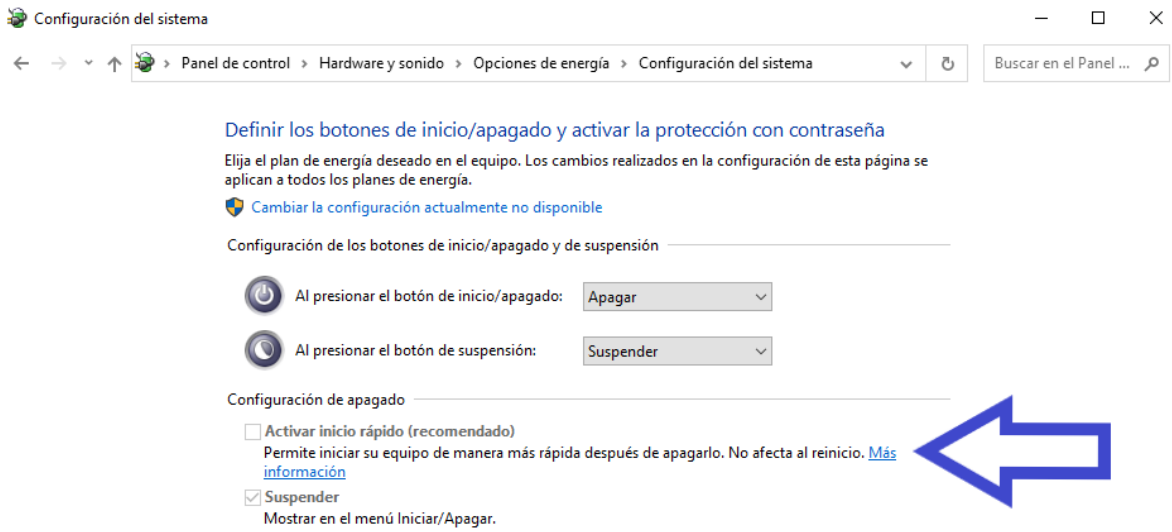


Figura 3.319 Deshabilitación del inicio rápido

- *Configuración para el apagado, reinicio y cancelado remoto de los ordenadores*

Básicamente estas 3 funciones se llevaron a cabo, a través de la dirección IP de los ordenadores por lo que no es adecuado que estas cambien dinámicamente, por esto se estableció direcciones IP fijas a cada ordenador. En la Figura 3.320 se destaca como se configuró una dirección IP estática en un ordenador.

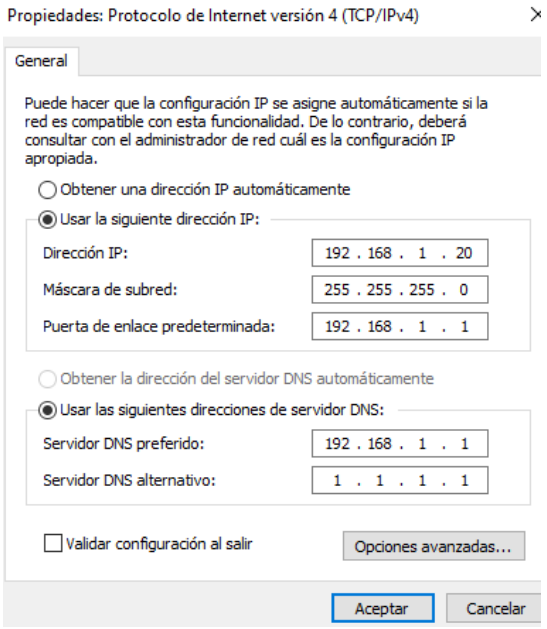


Figura 3.320 Configuración de una dirección IP fija en un ordenador

Después de lo anterior, para evitar el problema de acceso denegado se cambió una directiva de seguridad en el ordenador. Para esto inicialmente se accedió en herramientas administrativas y se pulsó sobre la opción Directiva de seguridad local justo como se aprecia en la Figura 3.321.

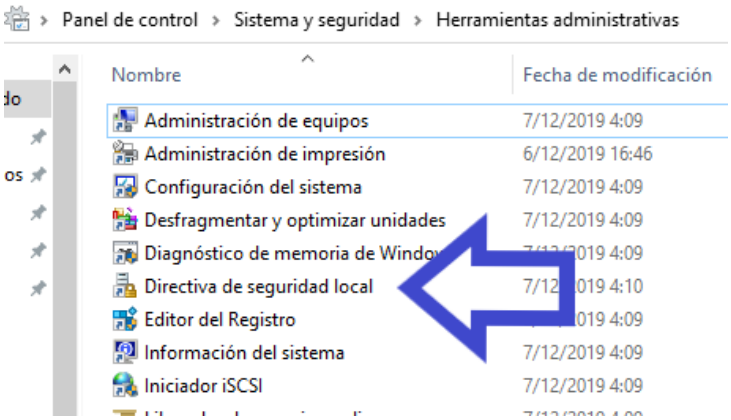


Figura 3.321 Directiva de seguridad local

Acto seguido, se ingresó en el directorio Directivas Locales en la subcarpeta Asignación de derechos de usuario, así como se observa en la Figura 3.322. Entonces, se seleccionó la directiva Forzar cierre desde un sistema remoto. A continuación, en la ventana que se desplegó se presionó el botón Agregar usuario o grupo y se añadió el usuario administrador que se configuró en el *script* de *Python*.

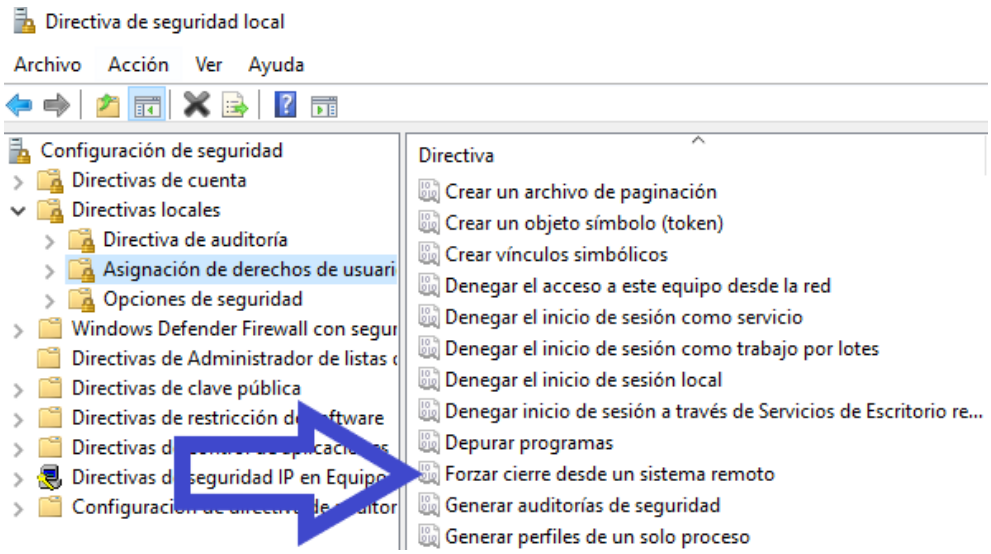


Figura 3.322 Directiva Forzar cierre desde un sistema remoto

También se puede escribir en la ventana de agregar usuario o grupo la palabra “Todos” para que todos puedan realizar esta tarea tal cual se observa en la Figura 3.323.

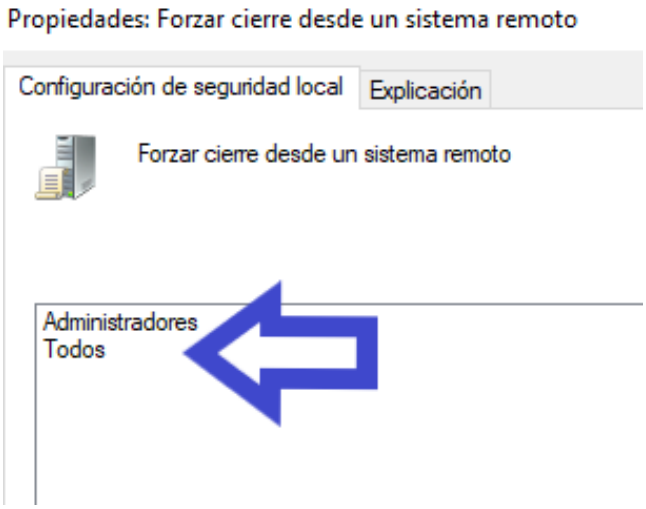


Figura 3.323 Configuración de la directiva forzar cierre desde un sistema remoto

Posteriormente, se modificó en el Editor de Registro el directorio *HKEY_LOCAL_MACHINE,* en este se almacena información sobre la configuración del equipo local tanto a nivel de *hardware* como de *software* [65]. Para esto primeramente se entró en el editor de registro digitando las teclas *Windows* + r y colocando la palabra *regedit* en la ventana que se despliega, justo como se observa en la Figura 3.324. Otra forma válida de ingresar es escribiendo el comando *regedit.exe* en la terminal cmd.

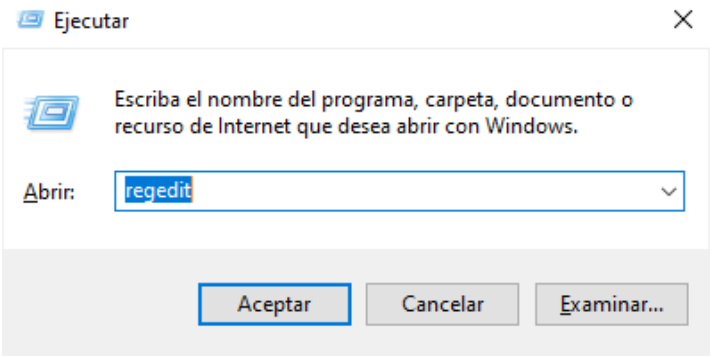


Figura 3.324 Ingreso al editor de registros

Una vez dentro del editor de registro se accedió a la ruta *Equipo\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System.* Se recomienda que antes de modificar cualquier parámetro se realice primero un *backup* para poder restaurar todo a las condiciones iniciales si ocurre algún error. En la Figura 3.325 se muestra la exportación del archivo y en la Figura 3.326 el respaldo creado.

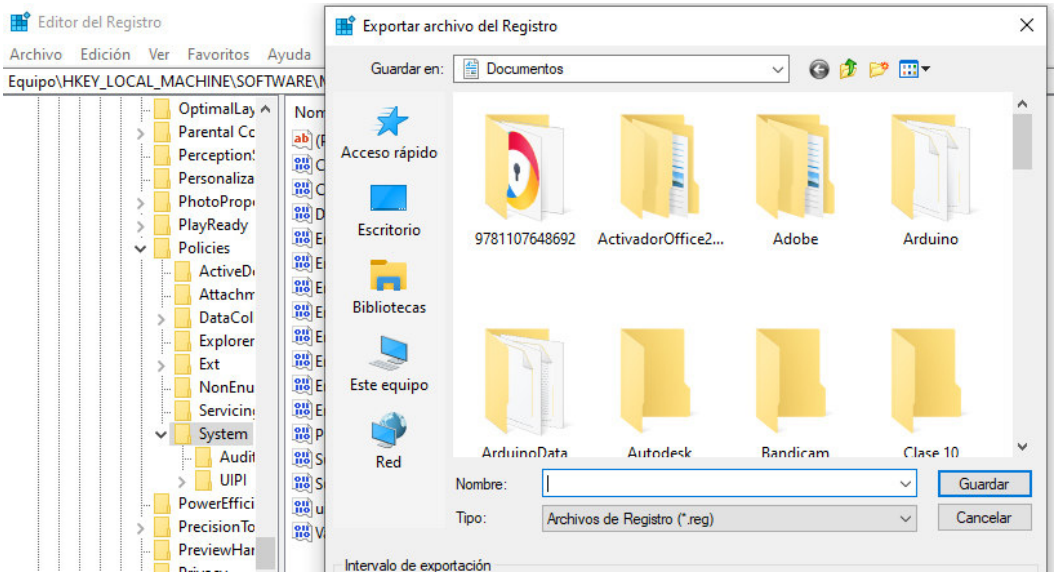


Figura 3.325 Exportación del archivo

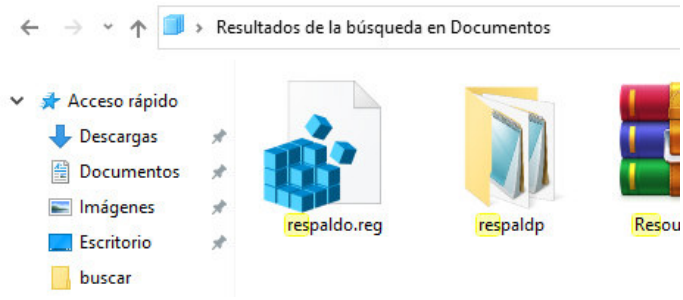


Figura 3.326 Archivo de *backup* respaldo.reg

Enseguida, en la ruta anterior se creó un registro *REG_DWORD* con el nombre *LocalAccountTokenFilterPolicy* con el valor de 1 tal como se aprecia en la Figura 3.327.

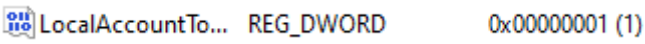


Figura 3.327 *Registro LocalAccountTokenFilterPolicy* con el valor de 1

Tal como se indicó en la Figura 3.100 se ocupó varios comandos del paquete *samba-common* para ejecutar las actividades de apagar, reiniciar y cancelar los ordenadores remotamente. Por lo cual, es importante permitir el acceso a este paquete en el computador.

Para esto, se ingresó en el cortafuegos de *Windows* y en reglas de entrada se escogió la opción nueva regla justo como se observa en la Figura 3.328.

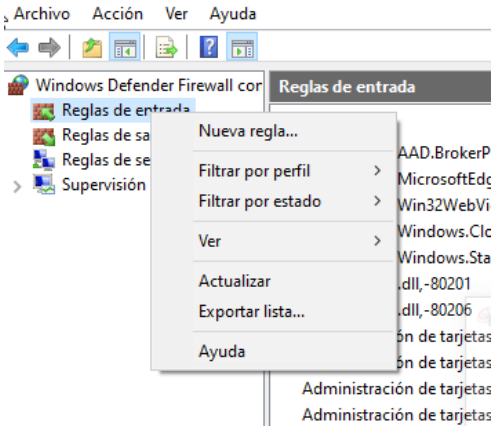


Figura 3.328 Cortafuegos de *Windows*

Entonces, se creó 2 reglas, la primera regla se nombró *SAMBA_TCP* con los puertos 139 y 445. En cambio, la segunda regla se llamó *SAMBA_UDP* con los puertos 137 y 138. En la Figura 3.329 se puede visualizar lo dicho.

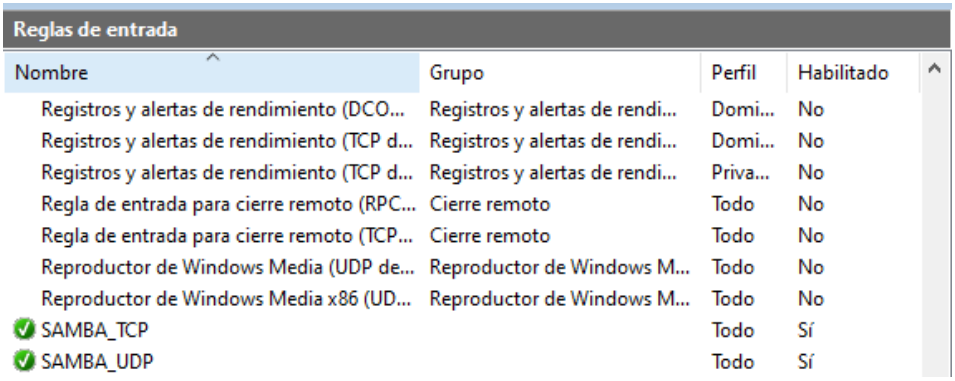


Figura 3.329 Reglas de entrada para el paquete *samba-common*

- Creación de un usuario administrador en *Windows*

Un problema importante para la ejecución de los comandos del paquete *samba-common* fue que estos tenían que emplear un usuario administrador con su clave. Sin embargo, para el desarrollo del proyecto los ordenadores se pidieron prestados y los dueños no recordaban las claves.

Por este motivo, se solventó este inconveniente creando un nuevo usuario del tipo administrador con su contraseña, ya que los comandos funcionarán perfectamente, aunque se esté en otra sesión diferente al usuario creado.

Para esto, en primer lugar, se ejecutó la terminal cmd como administrador. Luego con el comando *net user* se observó los usuarios que tenía el equipo justo como se muestra en la Figura 3.330.

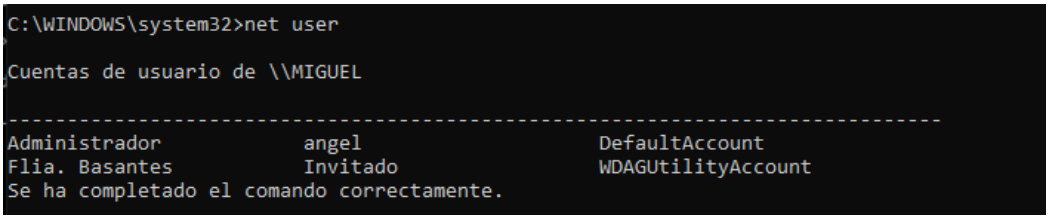


Figura 3.330 Comando *net user*

Más adelante, se procedió a crear el nuevo usuario con el comando *net user "nombre del usuario" /add*. En la Figura 3.331 se destaca a modo de ejemplo la creación exitosa del usuario prueba.

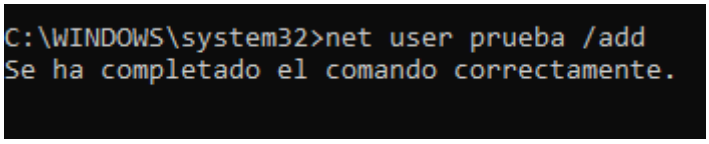


Figura 3.331 Creación del usuario prueba

A continuación, se examinaron los grupos locales del ordenador con el comando *net localgroup* tras esto se agregó el usuario al grupo Administradores con el comando *net localgroup Administradores "nombre del usuario" /add.* En las Figura 3.332 y Figura 3.333 se puede ver lo mencionado. Si algún rato se desea eliminar el usuario del grupo, digitar el comando *net localgroup Administradores "nombre del usuario" /delete* o si se quiere borrar por completo al usuario, emplear el comando net user *"nombre del usuario"* /delete.

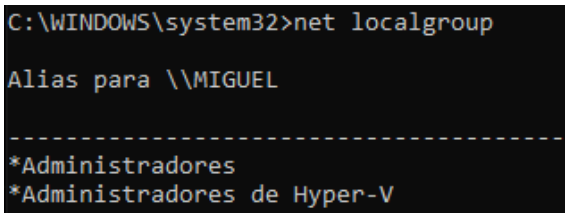


Figura 3.332 Grupos locales del ordenador

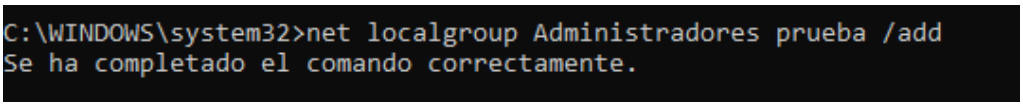


Figura 3.333 Agregación del usuario al grupo Administradores

Finalmente, se añadió la contraseña al usuario con el comando *net user "nombre del usuario" ** tal cual se observa en la Figura 3.334.

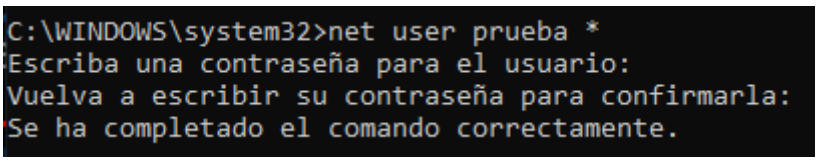


Figura 3.334 Agregación de la clave del usuario

- Expiración de la contraseña

Después de un tiempo, específicamente 42 días si no se modifica nada en las Directivas de Seguridad Local, la contraseña caducará y se presentará el error de la Figura 3.335 al ejecutar los *scripts* de *Python* de los ordenadores en la *Raspberry Pi*.

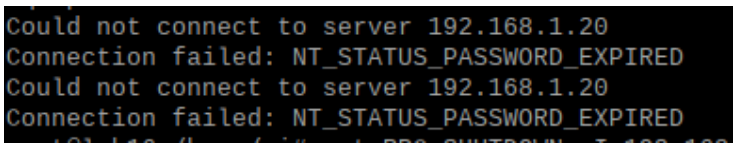


Figura 3.335 Error en la conexión contraseña expirada

Antes de nada, se debe aclarar que el suceso que la contraseña expire es algo bueno ya que le indica al usuario que debe modificarla periódicamente y esta opción se debe activar por defecto [66]. Ahora bien, si el lector lo considera como una medida de seguridad innecesaria, puede desactivarse de la siguiente manera:

En primer lugar, en la barra de búsqueda de Windows buscar lusrmgr.msc. Una vez dentro, seleccionar en el directorio Usuarios al usuario deseado y en la ventana que se despliega marcar la opción La contraseña nunca expira. En las Figura 3.336 y Figura 3.337 se destaca lo mencionado anteriormente.

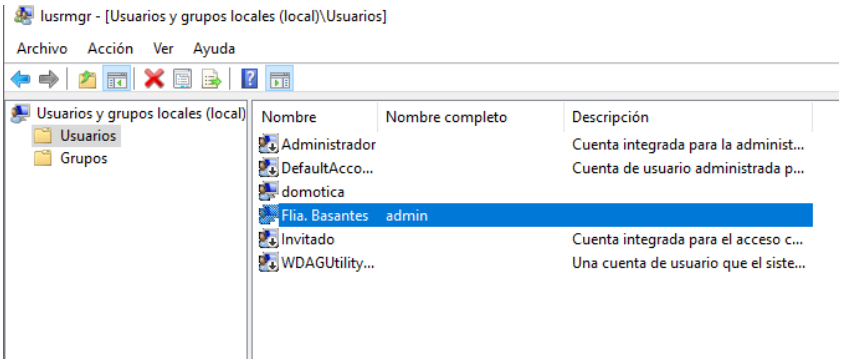


Figura 3.336 lusrmgr.msc

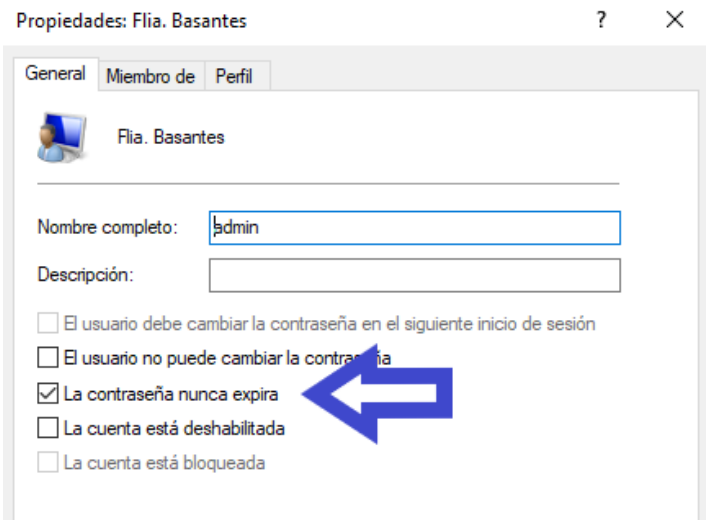


Figura 3.337 Configuración para que la contraseña del usuario nunca expire

Finalización del prototipo

Como parte de la implementación se respaldó todo el sistema, para esto se ingresó la tarjeta SD de 32 GB que contenía todas las configuraciones y *scripts* en el ordenador. A continuación, por medio del programa *USB Image Tool* se creó un *backup* de toda la imagen ISO.

En las Figura 3.338 y Figura 3.339 se puede observar lo dicho. También, es importante destacar que este proceso se realiza ya que un error muy común es que la tarjeta SD se corrompa por un mal manejo de la *Raspberry Pi.*

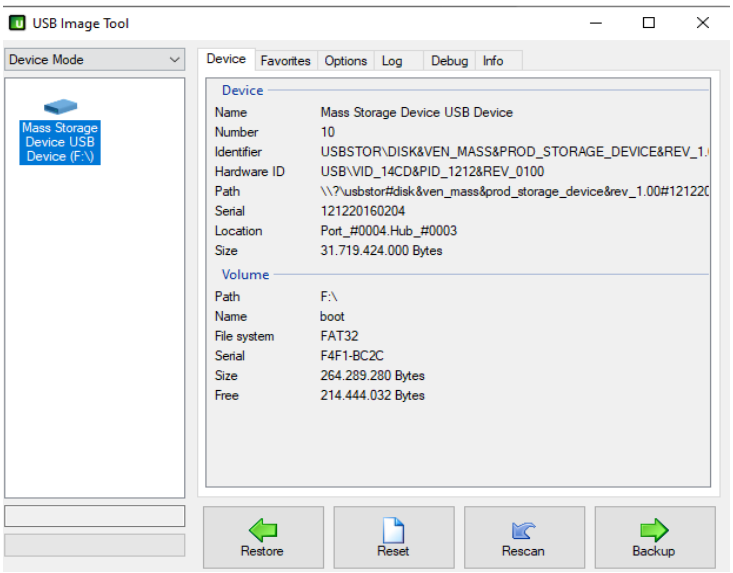


Figura 3.338 Programa *USB Image Tool*

Figura 3.339 *Backup* de todo el sistema

Por último, en la Figura 3.340 se puede ver el prototipo ya terminado y listo para efectuar las correspondientes pruebas de funcionamiento.

Figura 3.340 Prototipo finalizado

3.5 Realización de pruebas de funcionamiento

Prueba de lanzamiento de los *scripts* automáticos al iniciar el prototipo

Primeramente, al encender el prototipo el led indicador se enciende de manera automática justo como se observa en la Figura 3.341. Más tarde, cuando se hayan cargado todos los componentes de red en la *Raspberry Pi,* la pantalla se lanza en modo quiosco y se desactiva el led indicador tal cual se ilustra en la Figura 3.342.

Figura 3.341 Led indicador del encendido del prototipo

Figura 3.342 Pantalla en modo quiosco

A continuación, el usuario encargado del prototipo recibe el mensaje de la Figura 3.343 en el *Bot* de *Telegram.* Finalmente, se activa el módulo relé que controla el ventilador justo como en la Figura 3.344 y este comienza a funcionar tal cual se muestra en la Figura 3.345. Todo el proceso descrito se ejecutó de manera correcta por lo que no se tuvo que hacer modificaciones.

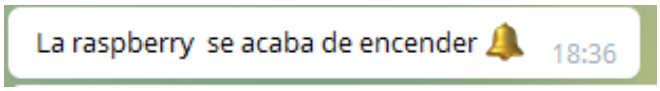


Figura 3.343 Mensaje en el *Bot* de *Telegram* al encender la *Raspberry Pi*

Figura 3.344 Relé que controla el ventilador encendido

Figura 3.345 Lanzamiento automático del ventilador

Prueba de envío de mensaje automático al conectarse por *SSH* al prototipo

Para probar esta función se ingresó por *SSH* al prototipo a través del programa *PuTTY* tal cual se aprecia en la Figura 3.346*,* previamente se abrió el puerto a través del cortafuegos *UFW* en el *Bot* de *Telegram*.

Una vez establecida la conexión el usuario encargado del prototipo recibe el mensaje de la Figura 3.347. Este mensaje varía de acuerdo con el usuario que se empleó para establecer la conexión por *SSH*, justo como se destaca en la Figura 3.348.

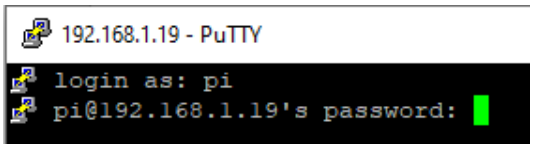


Figura 3.346 Conexión por *SSH*

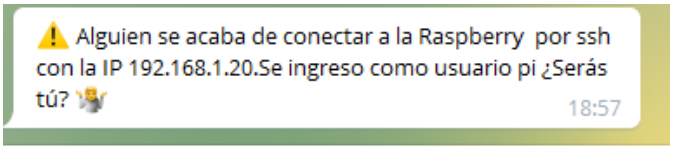


Figura 3.347 Mensaje en el *Bot* de *Telegram* al establecer la conexión por *SSH*

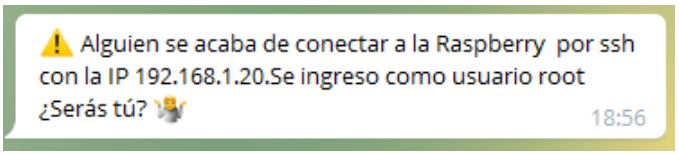


Figura 3.348 Mensaje al establecer la conexión por *SSH* como usuario *root*

Prueba de funcionamiento de la interfaz web

En primer lugar, se probó el registro de nuevo usuario, para esto se entró en la interfaz web como usuario administrador y se creó el usuario de la Figura 3.349.

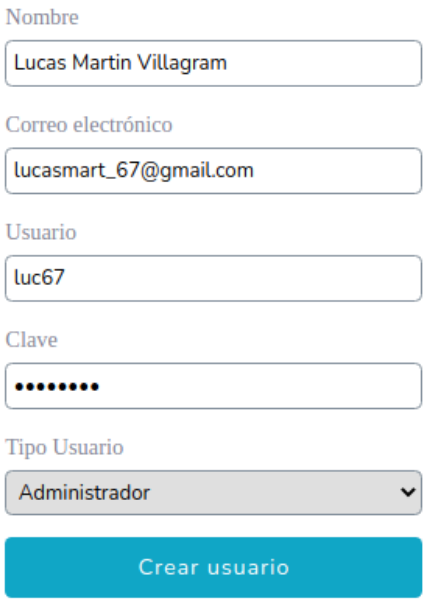


Figura 3.349 Registro de nuevo usuario

Posteriormente, el proceso resultó exitoso ya que saltó la alerta de la Figura 3.350. Más tarde, en la lista de usuarios, tal cual se muestra en la Figura 3.351, se buscó al usuario creado y se constató que todos los parámetros llenados en el registro de usuario estuvieran correctos en la tabla, además que el paginador correspondiera con el valor de página según el número de registros.

Figura 3.350 Alerta de usuario creado correctamente

Figura 3.351 Lista de usuarios

A continuación, con el usuario creado se procedió a iniciar sesión tal como se visualiza en la Figura 3.352.

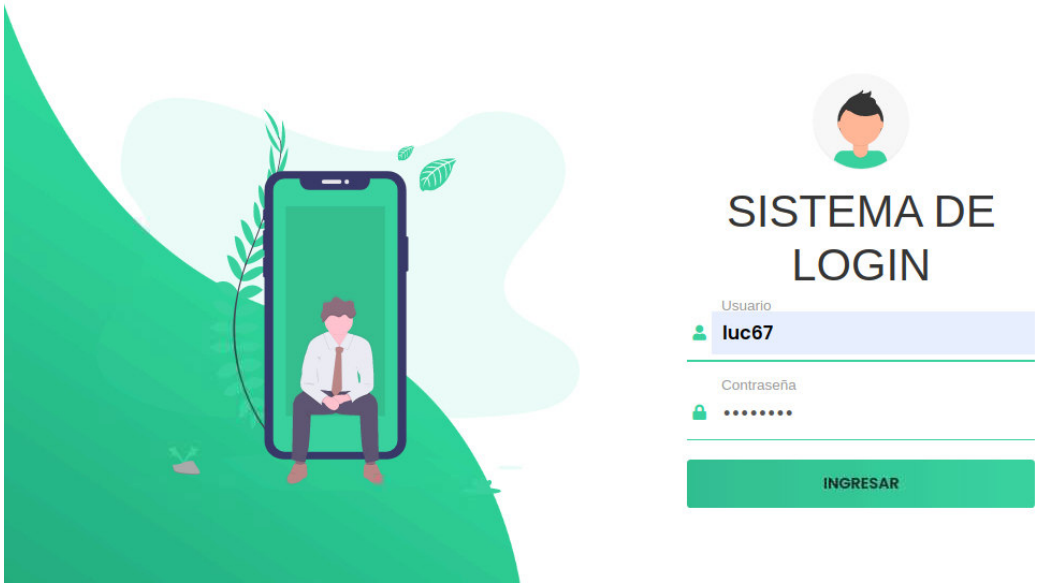


Figura 3.352 Inicio de sesión con el usuario creado

Después, se constató que la página de inicio de sesión y el panel de control correspondieran al usuario creado previamente. También que el año, día y mes de la fecha estuvieran correctos según el día que ingresó el usuario. Lo dicho se encuentra desde la Figura 3.353 a la Figura 3.355.

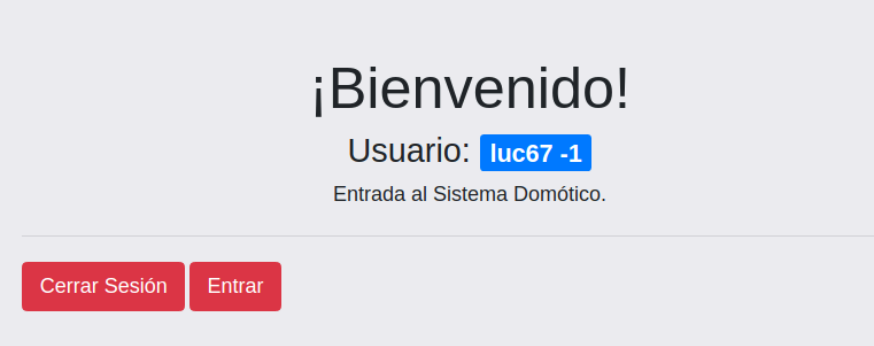


Figura 3.353 Inicio de sesión del usuario luc67

Figura 3.354 Fecha y nombre del usuario luc67

Figura 3.355 Panel de control del usuario administrador luc67

Luego, se verificó que el formulario de actualización de usuario trabajara de forma adecuada. Para esto se modificó el parámetro de tipo de usuario del usuario luc67 cambiándolo de administrador a usuario normal tal cual se observa en la Figura 3.356. Este proceso se completó correctamente tal como indica la alerta de la Figura 3.357. Seguidamente se buscó al usuario modificado en la lista de usuarios para confirmar que los cambios en la tabla también tuvieron efecto. En la Figura 3.358 se nota lo dicho.

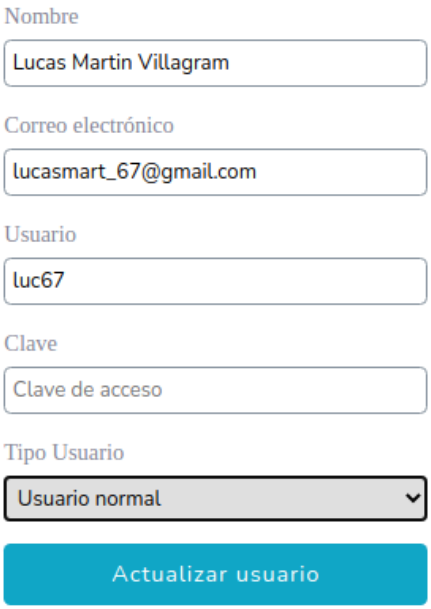


Figura 3.356 Actualización del usuario luc67

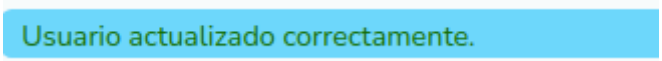


Figura 3.357 Alerta de usuario actualizado correctamente

Figura 3.358 Búsqueda del usuario modificado

Tras lo anterior se volvió a ingresar sesión con el usuario modificado para corroborar que efectivamente su rol sea el de un usuario normal. En las Figura 3.359 y Figura 3.360 se muestra lo mencionado.

Figura 3.359 Inicio de sesión del usuario luc67

Figura 3.360 Panel de control del usuario normal luc67

Finalmente, se examinó la función de eliminación de usuario. Para esto se eliminó el usuario luc67 tal como se destaca en la Figura 3.361. Además, se volvió a iniciar sesión con este usuario para comprobar que esté dado de baja y ya no pueda ingresar al prototipo.

Figura 3.361 Eliminación del usuario normal luc67

Pruebas de funcionamiento del sistema domótico

Inicialmente, se probó el sistema domótico presionando sobre las opciones de la Figura 3.362 y se constató que redirigieran a las páginas de control correspondientes a cada función.

Figura 3.362 Sistema domótico

- *Pruebas del control de las computadoras*

En la página de control de las computadoras se comprobó que el menú seleccionable de la Figura 3.363 estuviera funcionando adecuadamente.

Figura 3.363 Página de control de las computadoras

- Prueba de encendido de las computadoras

Para verificar el encendido de los ordenadores primeramente se eligió el PC2 del menú de la Figura 3.363. Una vez en esta sección se presionó sobre el botón encender justo como se destaca en la Figura 3.364, seguidamente se visualizó la animación y el ordenador que corresponde a la PC2, que se observa en la Figura 3.365, se encendió de manera correcta.

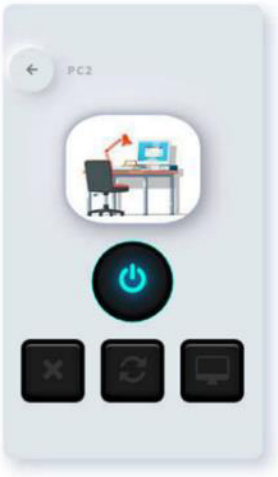


Figura 3.364 Encendido de la PC2 desde la interfaz web

Figura 3.365 Ordenador que corresponde a la PC2 encendido

- Prueba de apagado de las computadoras

Para comprobar el apagado de las computadoras se volvió a presionar sobre el mismo botón de encendido tal cual se ilustra en la Figura 3.366. Entonces la animación cambió y el ordenador que se ve en la Figura 3.367 empezó el proceso de apagado de manera adecuada.

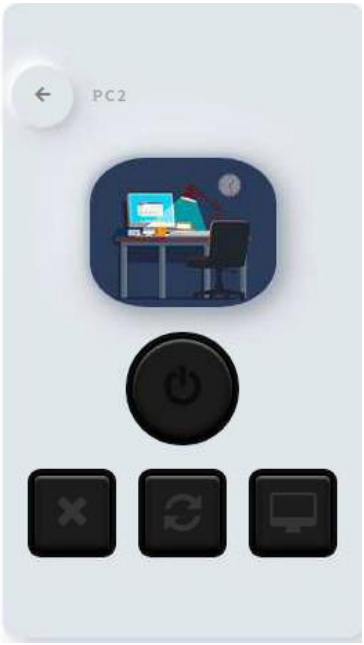


Figura 3.366 Apagado de la PC2 desde la interfaz web

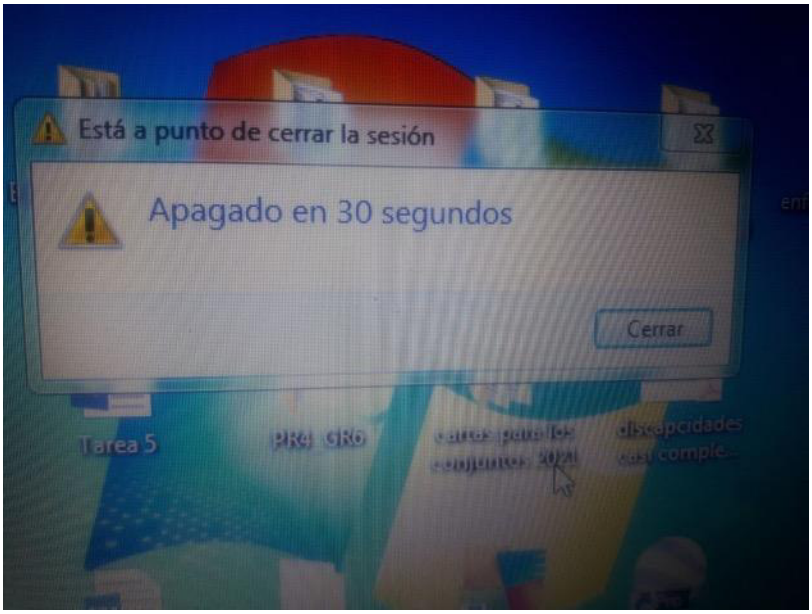


Figura 3.367 Ordenador que corresponde a la PC2 apagándose

- Prueba de reinicio de las computadoras

Para poner a prueba el reinicio de los ordenadores se pulsó sobre el botón reinicio tal como se observa en la Figura 3.368. Luego, que la animación se efectuara el ordenador de la PC2 comenzó el proceso de reinicio justo como se aprecia en la Figura 3.369. Finalmente, después del tiempo establecido la PC2 se reinició de forma exitosa.

Figura 3.368 Reinicio de la PC2 desde la interfaz web

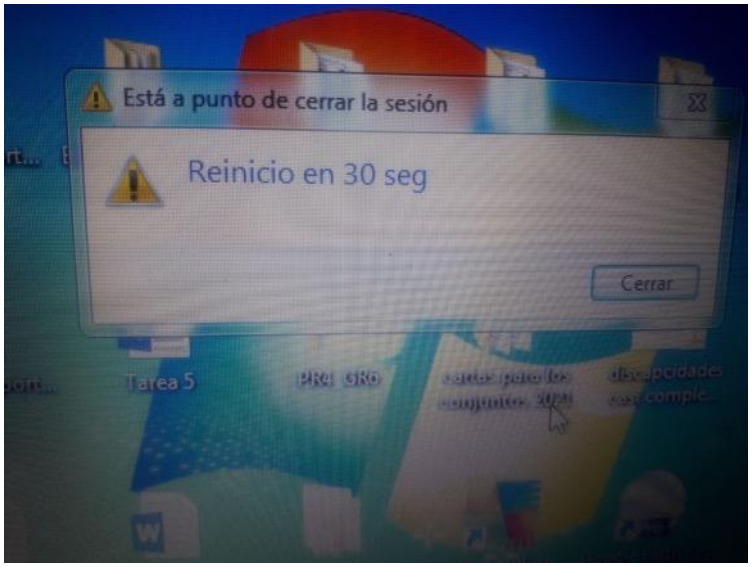


Figura 3.369 Ordenador que corresponde a la PC2 reiniciándose

- Prueba de cancelado del apagado o reinicio de los ordenadores

Por otro lado, para constatar el cancelado de los ordenadores justo al momento que se está efectuando el reinicio o apagado, es decir antes de que el tiempo establecido finalice, se pulsó sobre el botón cancelar tal cual se mira en la Figura 3.370. Más tarde se indicó al usuario del ordenador de la PC2 que el cierre de sesión se canceló tal como se visualiza en la Figura 3.371.

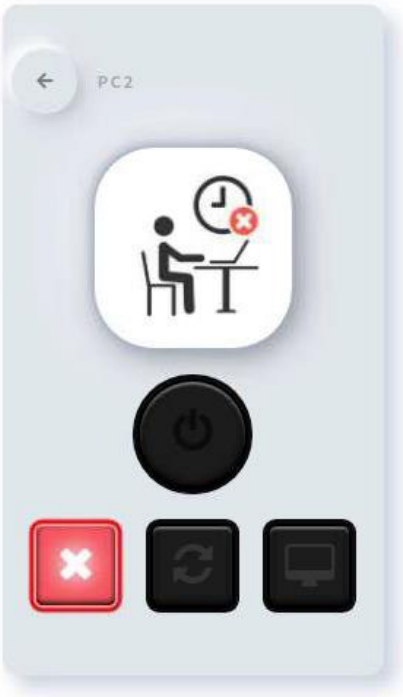


Figura 3.370 Cancelado del reinició o apagado de la PC2 desde la interfaz web

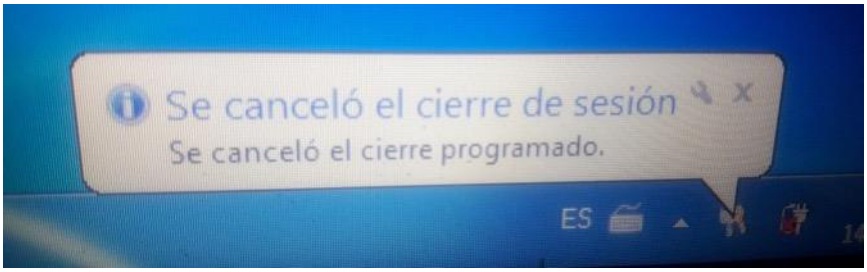


Figura 3.371 Ordenador que corresponde a la PC2 cancelando el cierre de sesión

- Prueba del estado de los ordenadores

Para la prueba del estado de los ordenadores en la sección del PC2 se pulsó sobre el botón estado tal cual se ve en la Figura 3.372. Este estado indica al usuario en tiempo real el funcionamiento del ordenador, según se estableció en el diagrama de flujo de la Figura 3.106, siempre y cuando en el *PC* esté conectado el cable de red *Ethernet*.

Las mismas pruebas anteriores se llevaron a cabo para determinar que el *PC1* funcionara correctamente. Principalmente, la *PC1* y *PC2* están integradas al sistema tal cual se puede notar en la Figura 3.373.

Sin embargo, la interfaz web se probó para manejar 20 ordenadores tal como se ilustra en el menú de la Figura 3.363, de manera individual y grupal, es decir todos a la vez.

Figura 3.372 Estado de la PC2

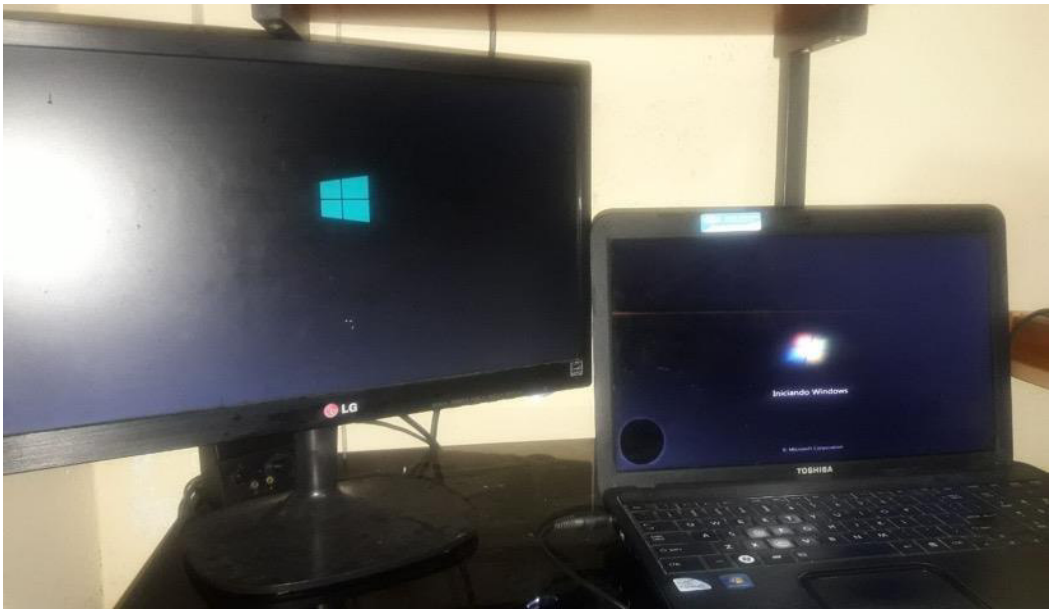


Figura 3.373 Pruebas en los ordenadores

- *Pruebas del control de la puerta*

Antes que nada, en la página de control de la puerta se verificó el menú seleccionable de la Figura 3.374 y se determinó que está operando de forma correcta.

Figura 3.374 Página de control de la puerta

- Prueba de abrir la puerta y estado

Para la prueba de abrir la puerta se pulsó sobre el botón deslizante de la Figura 3.375 hacia la derecha. Entonces la animación se mostró y la maqueta de la puerta de la Figura 3.376 se abrió. No obstante, también se presionó sobre el botón estado y según lo establecido en el diagrama de flujo de la Figura 3.74, el estado en tiempo real de la puerta se mostró y la alarma se activó como es debido, hasta que el usuario presione nuevamente sobre el mismo botón estado para desactivarla.

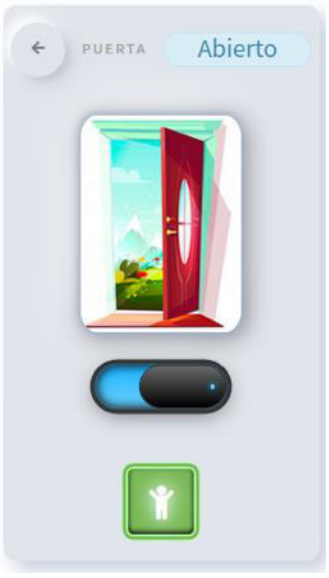


Figura 3.375 Puerta abierta desde la interfaz web

Figura 3.376 Puerta de la maqueta abierta

- Prueba de cerrar la puerta y estado

Para la prueba de cerrar la puerta se presionó sobre el botón deslizante de la Figura 3.377 hacia la derecha. Más tarde, la animación cambió y la maqueta de la puerta de la Figura 3.378 se cerró. Seguidamente, como el botón estado estaba activo también se reflejó el nuevo estado de la puerta en la interfaz web.

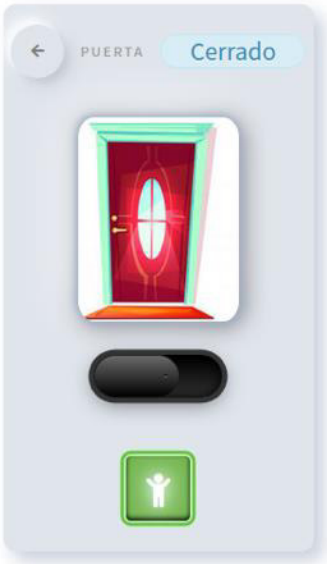


Figura 3.377 Puerta cerrada desde la interfaz web

Figura 3.378 Puerta de la maqueta cerrada

- Prueba del automático de la puerta

Por otro lado, la prueba del control automático de la puerta se llevó a cabo comprobando la lectura en tiempo real del sensor al presionar sobre el botón de la Figura 3.379. Cuando la distancia era mayor a la que se estableció en el diagrama de la Figura 3.77 la puerta permanecía cerrada tal cual se aprecia en la Figura 3.378, en cambio a una lectura inferior a la establecida la puerta se abría automáticamente, justo como se observa en Figura 3.376 de manera exitosa. Los datos del sensor que se representan en la Figura 3.379 fueron colocados con el propósito de que el usuario administrador pueda determinar fácilmente que el sensor ultrasónico esté trabajando.

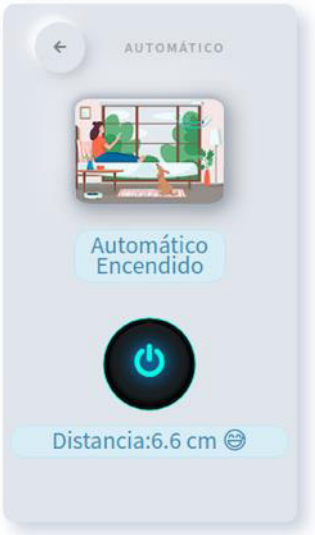


Figura 3.379 Activación del sistema automático de la puerta en la interfaz web

- *Pruebas del control de las luces*

En un principio, en la página de control de las luces se comprobó el menú seleccionable de la Figura 3.380 y se estableció que está operando de forma correcta.

Figura 3.380 Página de control de las luces

- Prueba de encender las luces

Para la prueba de encender las luces se escogió del menú de la Figura 3.380 la opción Dormitorio3 que corresponde al foco número 3. Entonces en esta sección se pulsó sobre el botón en forma de *switch* que se muestra en la Figura 3.381, seguidamente el Foco 3 de la Figura 3.382 se encendió. Además, se probó que este también encendía desde el conmutador.

Figura 3.381 Encendido del Foco 3 desde la interfaz web

Figura 3.382 Foco 3 encendiéndose

- Prueba de apagar las luces

Para la prueba de apagar las luces se presionó nuevamente sobre el botón en forma de *switch* justo como se destaca en la Figura 3.383. Más adelante, la animación cambió y el Foco 3 de la Figura 3.384 se apagó. Posteriormente, se verificó que el foco también se apague desde el conmutador.

Figura 3.383 Apagado del Foco 3 desde la interfaz web

Figura 3.384 Foco 3 apagándose

Cabe recalcar que estas pruebas se volvieron a efectuar sobre todos los demás focos que componen el sistema domótico, además que desde la interfaz web se pueden controlar los focos de manera individual y grupal, es decir, todos a la vez tal como se mira en la Figura 3.385.

Figura 3.385 Pruebas en los focos

- Prueba del automático de la luz del pasillo

Para la prueba del control automático de la luz del pasillo se tuvo en cuenta el diseño establecido en el diagrama de flujo de la Figura 3.67. Para empezar, se activó el botón del automático. Si la hora no es la establecida y se activó el sensor justo como se puede apreciar en la Figura 3.386, la luz del pasillo se mantiene apagada. Por otro lado, si la hora es la establecida, el sensor empieza a funcionar detectando el movimiento y encendiendo la luz del pasillo o apagándola si no hay movimiento.

Figura 3.386 Activación antes de la hora establecida

Además, se verificó que los datos del movimiento que se representa en la interfaz web, tal como se observan en la Figura 3.387, correspondan con la lectura del sensor en tiempo real. El proceso descrito se realizó de manera adecuada y no se hizo mayores cambios.

Figura 3.387 Activación durante la hora establecida

- *Pruebas del control de las cortinas*

  - Prueba de subir las persianas

Para la prueba de subir las persianas, primeramente, en la página de control de las cortinas en el menú se seleccionó la cortina 2, a continuación, en esta sección se presionó sobre el botón subir tal cual se puede notar en la Figura 3.388.

Figura 3.388 Subida de la cortina 2 desde la interfaz web

Acto seguido, la cortina comenzó a abrirse tal como se destaca en la Figura 3.389 y se completó esta tarea con éxito. Además, también se constató que el programa estuviera almacenando el estado de la cortina, volviendo a presionar nuevamente el botón de subir, como la cortina ya estaba abierta no se abrió más y permaneció en la misma posición.

Figura 3.389 Cortina 2 subiendo

- Prueba de posicionar las persianas en la mitad

Para la prueba de posicionar las persianas en la mitad, en primer lugar, en la sección de la cortina 2 se pulsó sobre el botón del centro tal como se visualiza en la Figura 3.390.

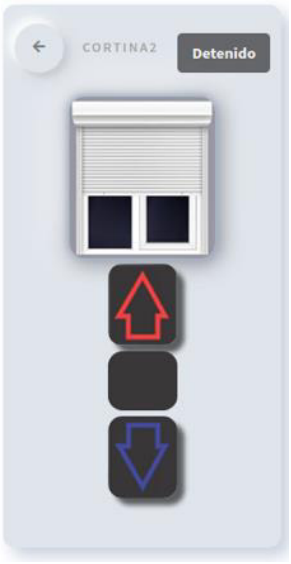


Figura 3.390 Posicionamiento de la cortina 2 en la mitad desde la interfaz web

Después la persiana de la Figura 3.391 se posicionó de manera adecuada en la mitad y al volver a pulsar sobre el botón del centro se corroboró que el estado si se está almacenando porque no cambió de posición.

Figura 3.391 Cortina 2 posicionada en la mitad

- Prueba de bajar las persianas

Para la prueba de bajar las persianas en la sección de la cortina 2 se presionó sobre el botón bajar justo como se ve en la Figura 3.392.

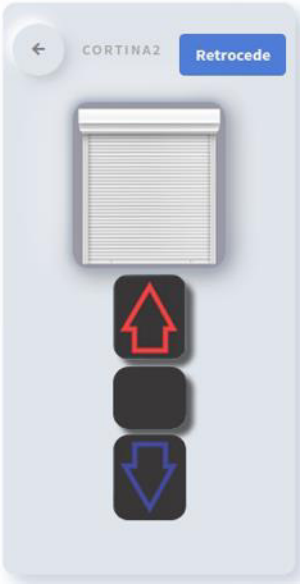


Figura 3.392 Bajada de la cortina 2 desde la interfaz web

Luego la persiana de la Figura 3.393 comenzó a bajar y completó esta función exitosamente según el número de pasos establecidos. Finalmente, se volvió a oprimir el mismo botón de bajar y está ya no bajó más, lo que indicó que el estado si se estaba guardando.

Figura 3.393 Cortina 2 bajando

Es importante dar a conocer que estas pruebas se volvieron a efectuar sobre la cortina 1, además que desde la interfaz web se pueden controlar las cortinas de manera individual y grupal, es decir, todas a la vez tal como se observa en la Figura 3.340.

- Prueba del automático de las cortinas

Por otra parte, para la prueba del automático de las cortinas se examinó nuevamente el diagrama de flujo de la Figura 3.92 con el propósito de determinar que el proceso realizado sea el adecuado y que los valores del sensor correspondan con los mostrados en la interfaz web tal cual se ilustra en la Figura 3.394.

Cuando existe suficiente nivel de luz las cortinas se abren automáticamente, justo como se observa en la Figura 3.389, en cambio sí hay bajo nivel de luz se cierran tal cual se nota en la Figura 3.393. Los valores del sensor se presentaron al usuario administrador ya que con estos puede determinar rápidamente si el sensor está trabajando o sucedió algún error.

Figura 3.394 Prueba del automático de las cortinas

- *Prueba del control manual del prototipo*

Para probar el control manual del prototipo, inicialmente se activó el botón del control manual. Seguidamente, se procedió a pulsar sobre las teclas que correspondían a las funciones de las cortinas, las luces, la puerta y los ordenadores en el control remoto de la Figura 3.115 según lo establecido en el diagrama de flujo de la Figura 3.109. Todo se desarrolló de acuerdo con lo previsto por lo que no se tuvo que hacer correcciones en esta parte. Finalmente se desactivó el control manual volviendo a presionar sobre el botón tal como se muestra en la Figura 3.395.

Figura 3.395 Prueba del control manual del prototipo

- *Prueba del control por comandos de voz*

Para la prueba del control del prototipo por comandos de voz, inicialmente se entró a la interfaz web desde el navegador del celular tal cual se observa en la Figura 3.397 ya que este cuenta con un micrófono. Además, esto fue muy útil porque también se verificó la compatibilidad de la interfaz web con otros dispositivos como son los celulares. Sin embargo, para realizar la conexión desde otros dispositivos debe habilitarse el acceso en el cortafuegos *UFW* por medio del *Bot* de *Telegram.*

En el cortafuegos se habilitó a la dirección IP 192.168.1.3 el acceso a la interfaz *web* y luego se configuró de forma estática la dirección IP en el dispositivo tal como se mira en la Figura 3.396.

Figura 3.396 Configuración de dirección IP estática en el celular

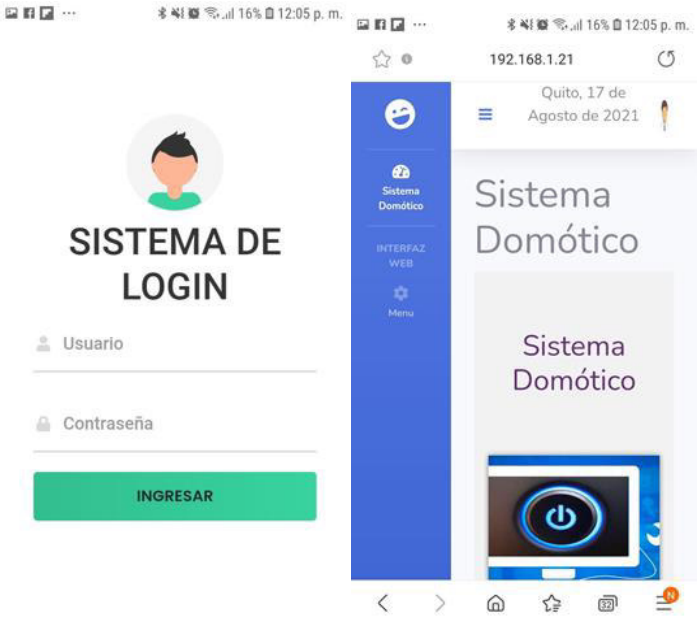


Figura 3.397 Prueba de compatibilidad con otros dispositivos

Una vez dentro de la interfaz web, se entró a las páginas de control del prototipo y en estas se habilitó el botón de micrófono y se ingresó con la voz los comandos correspondientes a cada función justo como se destaca, a modo de ejemplo, en la Figura 3.398. Entonces la animación se llevó a cabo y los dispositivos respondieron adecuadamente a lo diseñado.

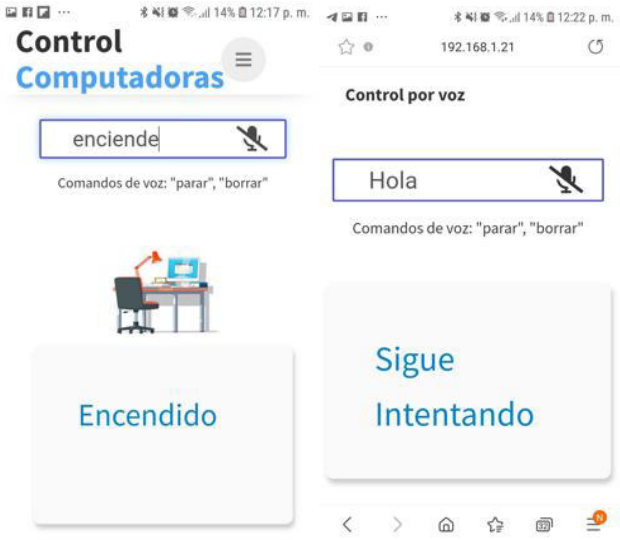


Figura 3.398 Prueba del control del prototipo por comandos de voz

Pruebas de funcionamiento del *Bot* de *Telegram*

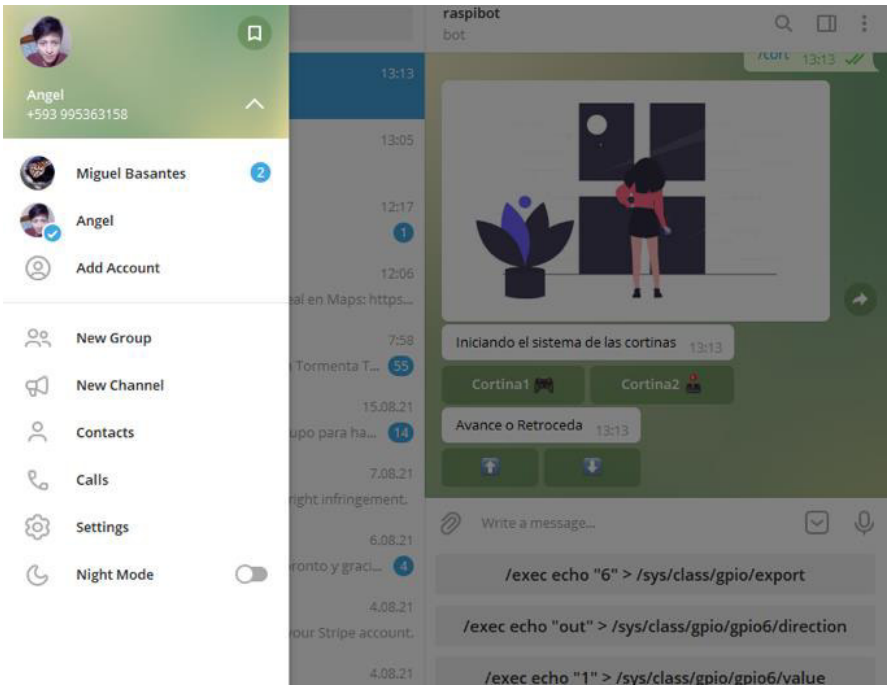


Figura 3.399 Usuarios de la aplicación *Telegram*

- *Prueba de inicio del Bot de Telegram*

Esencialmente, se ocupó dos usuarios de *Telegram* para probar el inicio del *Bot* tal cual se ilustra en la Figura 3.399. El primer usuario es Miguel este no tiene acceso al prototipo, por lo cual al iniciar el *Bot* este recibe el mensaje de la Figura 3.400. En cambio, el segundo usuario es Ángel el cual es el encargado del prototipo y este si tiene acceso, por lo que recibe el mensaje de la Figura 3.401. Sin embargo, si el encargado del prototipo considera necesario puede agregar al usuario Miguel, añadiéndolo en el *script* que se diseñó en base al diagrama de flujo de la Figura 3.120.

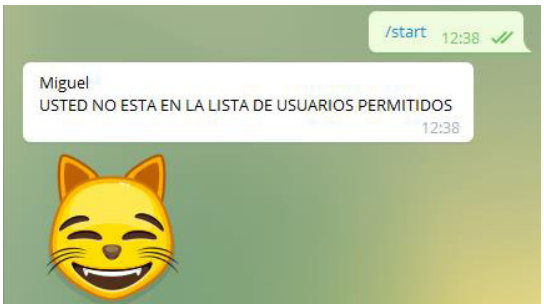


Figura 3.400 Mensaje que recibe el usuario no registrado en el *script* del *Bot*

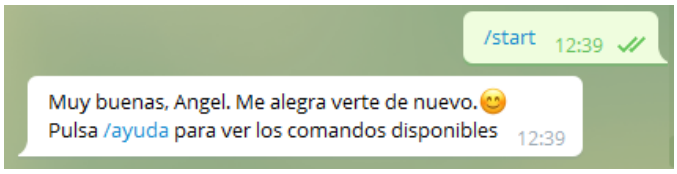


Figura 3.401 Mensaje que recibe el usuario registrado en el *script* del *Bot*

- *Prueba de mensajes predeterminados y por default*

Más tarde, se probó en el *Bot* de *Telegram* los mensajes predeterminados por ejemplo al ingresar la palabra “Hola”, “hola” o frases que incluyan está en una oración, el *Bot* responde correctamente con el mensaje de la Figura 3.402. Posteriormente, también se verificó con las demás palabras definidas en el diseño del *Bot*.

Figura 3.402 Mensajes predefinidos

También se ingresó frases que no estaban definidas en el *script* tal como se nota en la Figura 3.403 para constatar que el *Bot* responda de manera correcta con el mensaje por default.

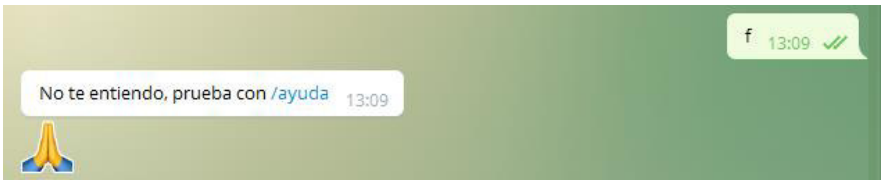


Figura 3.403 Mensaje por *default*

- *Prueba del menú de comandos*

Por otro lado, se determinó que el menú de comandos funcionara adecuadamente, para esto se ingresó en el *Bot* de *Telegram* el comando /ayuda y este respondió correctamente con el mensaje de la Figura 3.404.

Figura 3.404 Menú de comandos

- *Prueba de configuración de la pantalla en modo quiosco*

Para la prueba de configuración de la pantalla en modo quiosco, primeramente, se debe tener en cuenta que mientras el servicio esté ejecutándose no se puede cerrar el navegador, por lo cual el encargado del prototipo tiene dos formas de detener el servicio. La primera manera, es habilitar el puerto desde el cortafuegos para *VNC* y teclear remotamente Alt + F11.

En cambio, la segunda forma es a través del *Bot* de *Telegram,* para esto se debe emplear el comando /kiosk justo como se aprecia en la Figura 3.405. Seguidamente, el teclado cambiará a las principales opciones de la pantalla en modo quiosco.

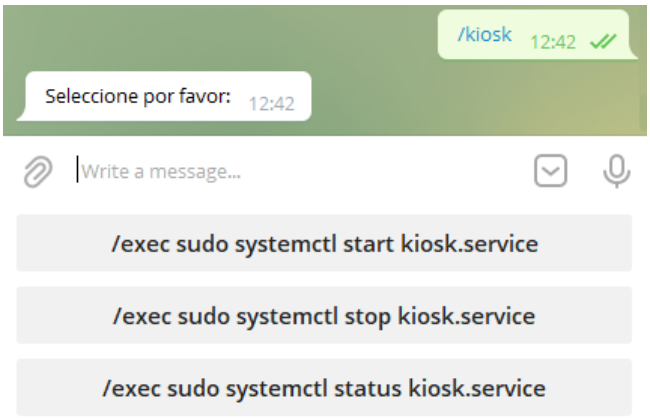


Figura 3.405 Menú de la configuración de la pantalla en modo quiosco

En la Figura 3.406 se destaca como de ejecutó correctamente la opción de detener la pantalla en modo quiosco. A su vez en la Figura 3.407 se nota la pantalla del prototipo con el servicio detenido.

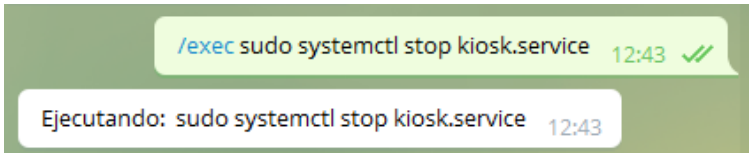


Figura 3.406 Ejecución de la opción modo quiosco detenido

Figura 3.407 Modo quiosco de la pantalla detenido

Por otra parte, en la Figura 3.408 se muestra cómo se llevó a cabo la opción de iniciar la pantalla en modo quiosco de manera adecuada. Además, en la Figura 3.409 se observa la pantalla del prototipo con el servicio ejecutándose.

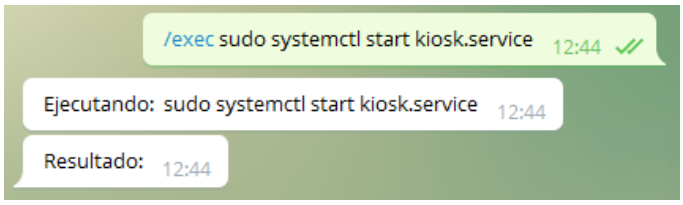


Figura 3.408 Ejecución de la opción modo quiosco iniciado

Figura 3.409 Modo quiosco de la pantalla iniciado

- *Prueba de configuración del cortafuegos UFW*

Para la prueba de configuración del cortafuegos en el *Bot* de *Telegram* se ocupó el comando */ufw*. Entonces, el teclado se modificó con las opciones principales del cortafuego justo como se ve en la Figura 3.410 de manera exitosa. Básicamente para utilizar el menú se debe comprender la información de la sección Instalación y configuración del cortafuegos *UFW* ya que los comandos y la función que estos desempeñan son las mismas, solo que evitan el tiempo de configurarlo por la terminal o remotamente por *SSH*.

Figura 3.410 Menú de la configuración del cortafuegos

- *Prueba de obtención de la temperatura e información relevante del prototipo*

Principalmente, se realizó la prueba de obtención de la temperatura de la *Raspberry Pi* e información importante de la misma por medio del comando /temp, justo como se observa en la Figura 3.411.

Figura 3.411 Comando /temp

Luego se mostró la información actual en un mensaje exitosamente tal como se mira en la Figura 3.412. Con toda esta documentación el usuario encargado puede llevar un registro del comportamiento de la *Raspberry Pi* y tomar acción sobre los parámetros no deseados.

Figura 3.412 Información importante de la *Raspberry Pi*

- *Prueba de la configuración básica de la Raspberry Pi desde el Bot de Telegram*

Para efectuar esta prueba se usó el comando /raspi tal cual se visualiza en la Figura 3.413. Después, se desplegó el menú en el teclado correctamente. Los comandos colocados en el menú son los que más se emplearon en el desarrollo del proyecto y el encargado del prototipo puede usarlos para conocer la dirección IP de *Gateway*, ver las direcciones IP colocadas en las interfaces de red eth0 y wlan0, reiniciar y apagar la *Raspberry Pi* inmediatamente o en un tiempo establecido, cancelar la ejecución del apagado o reinicio del prototipo y finalmente matar todos los *scripts* de *Python* que se están ejecutando.

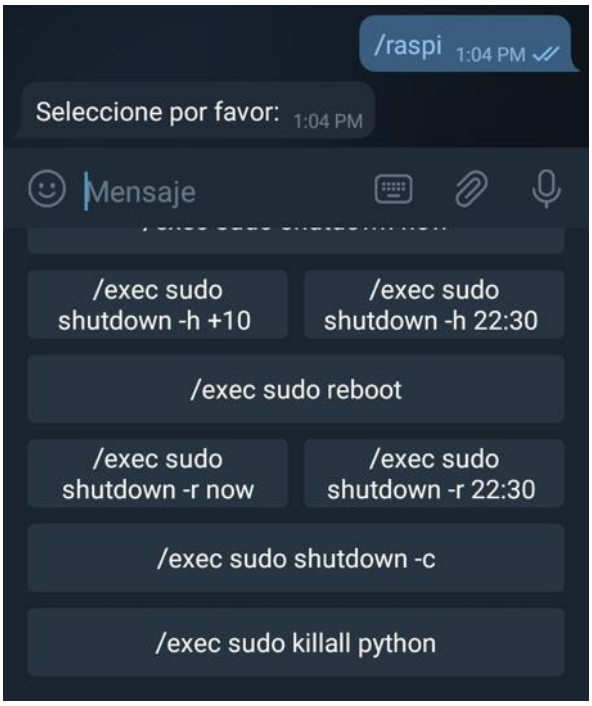


Figura 3.413 Menú de la configuración básica de la *Raspberry Pi*

A modo de ejemplo, se destaca en la Figura 3.414 la ejecución de una de las opciones mencionadas previamente, como es la de saber la dirección IP de *Gateway.*

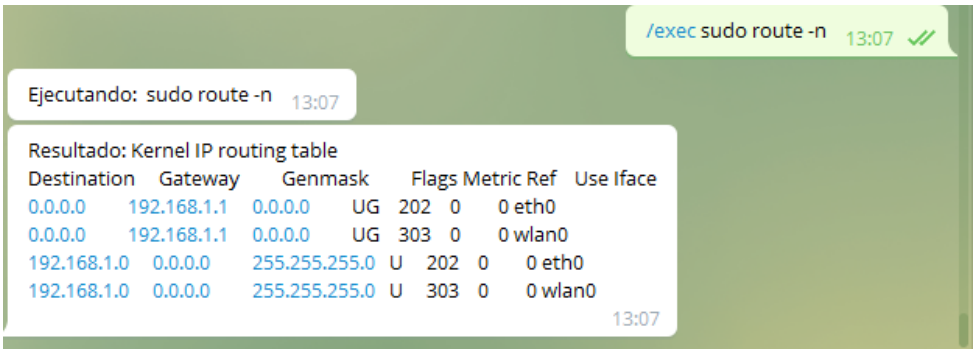


Figura 3.414 Ejecución de la opción para saber la dirección IP de *Gateway*

- *Prueba del control de la salida de un pin de la Raspberry Pi desde el Bot de Telegram*

Para desarrollar esta prueba se empleó el comando /pin en el *Bot* de *Telegram* justo como se puede observar en Figura 3.415. Más adelante se desplegó de manera exitosa el teclado. Este contiene principalmente los comandos necesarios para el control del pin en el que se encuentra el módulo que controla la ventilación del prototipo.

Sin embargo, cambiando el valor de este, se puede utilizar para el control de cualquier pin que el usuario encargado necesite. Tal como se indicó en la sección Programación desde la línea de comandos en la Figura 3.58.

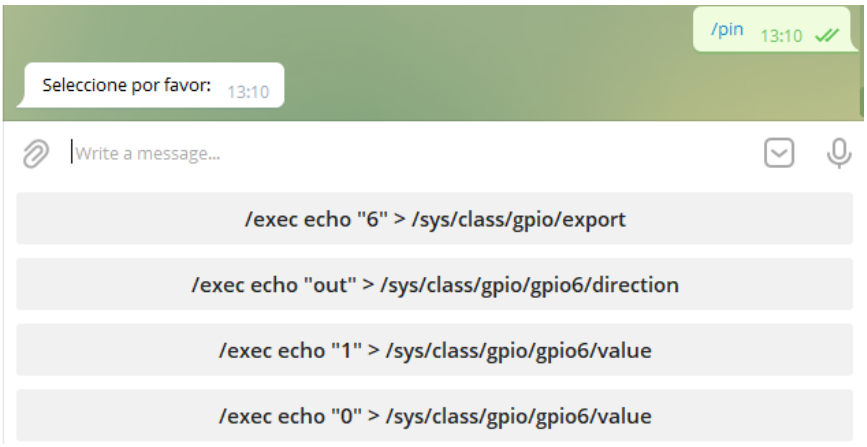


Figura 3.415 Menú para el control de la salida de un pin de la *Raspberry Pi*

- *Prueba de otras funcionalidades del Bot de Telegram*

Por otro lado, para probar las demás funcionalidades se utilizaron los comandos /comp, /luc, /puert y /cort. Entonces los teclados en línea de cada tarea aparecieron correctamente. De la Figura 3.416 a la Figura 3.419 se muestra lo dicho.

Figura 3.416 Menú en línea para probar rápidamente los ordenadores

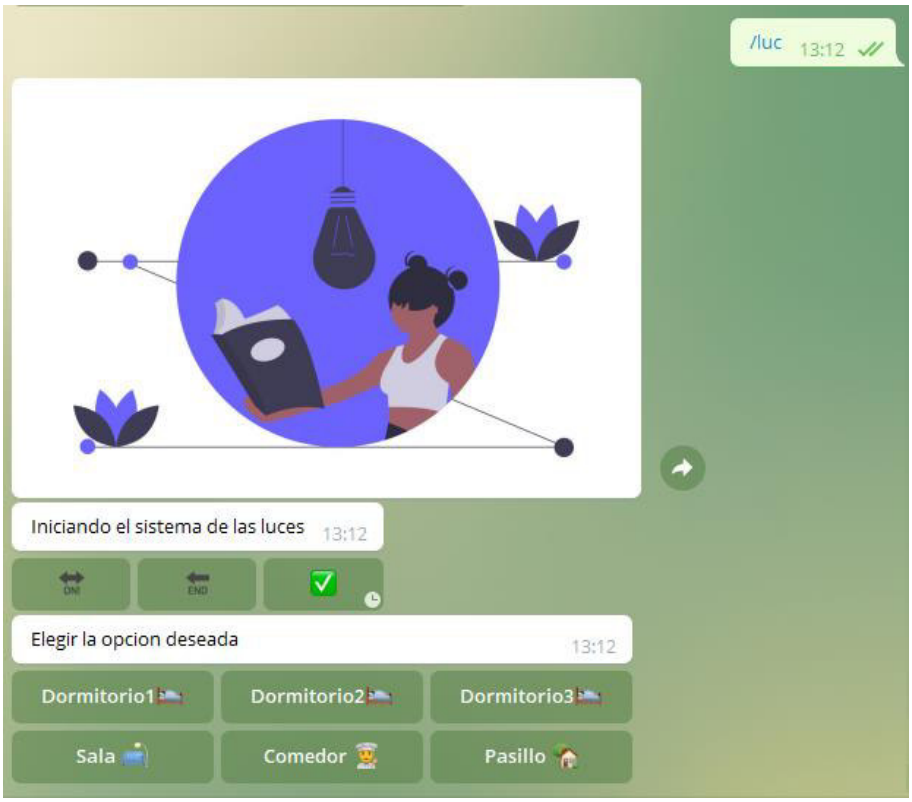


Figura 3.417 Menú en línea para probar rápidamente las luces

Figura 3.418 Menú en línea para probar rápidamente la puerta

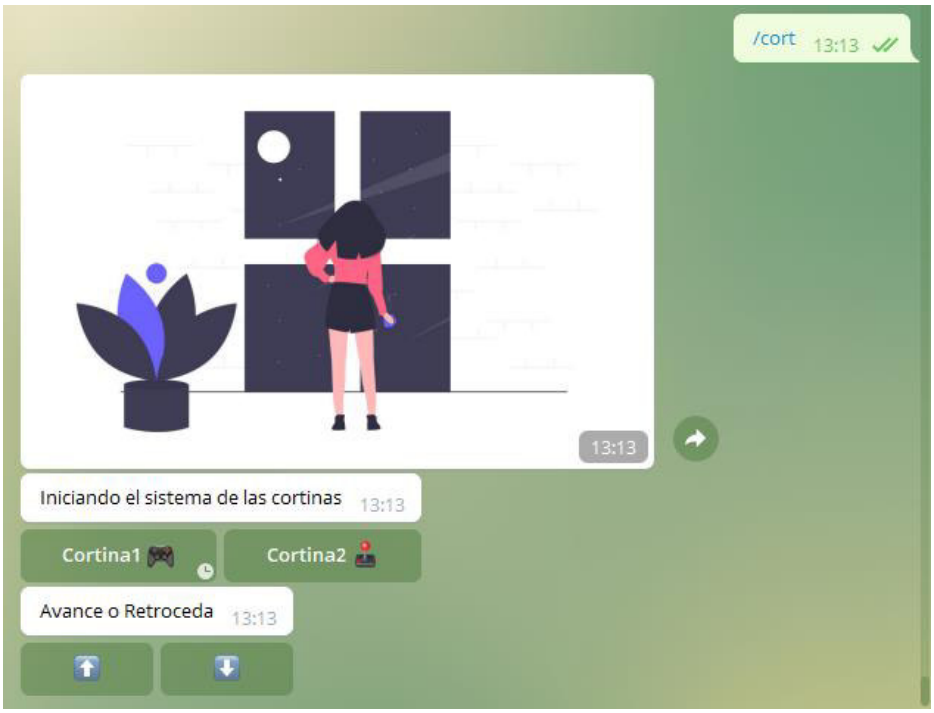


Figura 3.419 Menú en línea para probar rápidamente las persianas

Es importante destacar que, aunque se controla básicamente las 4 actividades como son: la puerta, las luces, las computadoras y las cortinas, estas se colocaron en el *Bot* para que el usuario encargado del sistema pueda realizar pruebas rápidas del funcionamiento de los dispositivos.

Además, en las cortinas, se puede utilizar para que cuando exista un corte de la luz inesperado y se detenga el sistema antes de completar el número de pasos, igualar a la posición deseada, caso contrario se tendría que desmontar todo el sistema e igualar la persiana manualmente.

3.6 Presupuesto referencial del proyecto

En primer lugar, en la Tabla 3.22 se presenta el presupuesto referencial de los materiales empleados para el desarrollo del prototipo. Por otra parte, en la Tabla 3.23 se colocó el costo estimado de la mano de obra en base al tiempo ocupado en la realización del prototipo, el cual incluye el diseño, el desarrollo, la instalación y las pruebas de funcionamiento. En cambio, en la Tabla 3.24 se encuentra el valor total aproximado de todo el prototipo.

Tabla 3.22 Presupuesto referencial de materiales

Cantidad	Detalle	Precio unidad	Precio Total
1	Kit *Raspberry Pi 4B* 2 (GB) RAM Case Fuente SD 32 (GB) *HDMI*	$ 125,00	$ 130,00 por costo de envío
1	Módulo relé 8 Canales	$ 10,51	$ 10,51
1	Sensor *PIR* (HC-SR501)	$ 2,50	$ 2,50
1	Sensor Ultrasónico (HC-SR04)	$ 2,50	$ 2,50
1	Sensor Magnético (MC-38)	$ 3,00	$ 3,00
1	KY-022 (Receptor Infrarrojo)	$ 2,50	$ 2,50
1	Control Remoto	$ 1,50	$ 1,50
1	Buzzer	$ 1,00	$ 1,00
1	Ventilador para *PC* 4 pines	$ 5,00	$ 5,00
1	Pantalla *7inch HDMI LCD (C)*	$ 99,99	$ 99,99
1	Conversor nivel 5 (V_{DC}) – 3,3 (V_{DC})	$ 1,95	$ 1,95
4	*Led*	$ 0,10	$ 0,40
7	Resistencia	$ 0,10	$ 0,70
3	*Switch*	$ 0,50	$1,50
1	Madera para piezas	$ 10,00	$ 10,00
1	Pintura para la maqueta puerta	$ 3,50	$ 3,50
3	Bisagras para puerta	$ 0,37	$ 1,11
1	Impresión PCB	$ 1,00	$ 1,00
2	Baquelita	$ 2,50	$ 5,00
1	Cloruro férrico	$ 1,25	$ 1,25
1	Cerradura Eléctrica con Solenoide	$ 9,95	$ 9,95
6	Foco	$ 1,00	$ 6,00
6	Conmutador simple	$ 1,83	$ 10,98
6	Boquilla	$1,50	$ 9,00
6	Cajetín Rectangular Plástico	$0,34	$2,04
1	Canaleta Dexson Lisa Blanca Adhesivo 13x7 (mm)	$2,02	$2,02
2	Nema Modelo: 17HS1910-P4170	$ 22,40	$ 44,80
2	Driver A4988	$ 2,50	$ 5,00

Cantidad	Detalle	Precio unidad	Precio Total
2	Capacitor	$ 0,20	$ 0,40
1	Conversor A/D MCP3008	$ 2,50	$ 2,50
1	Fotorresistencia (LDR)	$ 1,00	$ 1,00
6	RJ45	$ 0,25	$ 1,50
1	Cable UTP categoría 5e 9 (m)	$ 4,50	$ 4,50
1	Cable calibre 12 AWG 3 (m)	$ 3,00	$ 3,00
1	Jack hembra	$ 0,50	$ 0,50
30	Tornillo y tuerca	$ 0,05	$ 1,50
2	Rodamiento 626ZZ	$ 3,00	$ 6,00
1	Paquete de cables para la conexión	$ 8,00	$ 8,00
2	Persiana enrollable 64x80 (cm) con cadena de bolas de 6 (mm)	$ 35,00	$ 70,00
1	Fuente de alimentación 12 (V_{DC}) 5 (A_{DC})	$ 17,00	$ 17,00
1	Fuente de alimentación 5 (V_{DC}) 2 (A_{DC})	$ 10,00	$ 10,00
1	Enchufe	$ 1,50	$ 1,50
2	Costo de impresión del sistema de la cortina	$ 42,50	$ 85,00
1	Costo de impresión de la estructura módulo relé	$ 35,00	$ 35,00
1	Costo de impresión de la estructura sensor *PIR*	$ 12,00	$ 12,00
1	Costo de impresión de la estructura sensor ultrasónico	$ 12,00	$ 12,00
1	Costo de la impresión de la estructura prototipo	$ 240,00	$ 240,00
TOTAL MATERIAL			**$ 886,10**

Tabla 3.23 Presupuesto referencial de mano de obra

Cantidad	Detalle	Precio unidad	Precio Total
40	Tiempo de diseño (horas)	$ 8,00	$ 320
110	Tiempo de desarrollo (horas)	$ 8,00	$ 880
24	Tiempo de instalación (horas)	$ 5,00	$ 120
24	Tiempo de pruebas (horas)	$ 5,00	$ 120
TOTAL DE MANO DE OBRA			**$ 1440**

Tabla 3.24 Presupuesto referencial del prototipo

Detalle	Precio Total
Total material	$ 886,10
Total mano de obra	$ 1440
TOTAL PROTOTIPO	**$ 2326,10**

Como se puede observar en la Tabla 3.24, el valor estimado para el desarrollo del prototipo es de $ 2326,10, esto se debe a las diversas funcionalidades que el prototipo maneja por lo que requiere de una gran cantidad de componentes, principalmente por la impresión 3D. Sin embargo, a largo plazo, al replicar el proyecto nuevamente, el costo disminuirá considerablemente, porque ya se tendrán los diseños y el desarrollo.

3.7 Manual de Uso y Mantenimiento

Se diseñaron dos videos manuales, para ingresar a estos escanear el código QR de la Figura 3.420.

Figura 3.420 Código QR de los video manuales

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Al desarrollar el prototipo se concluyó que es fundamental estar en constante búsqueda de nuevas tecnologías y lenguajes de programación ya que estos permiten ampliar el conocimiento adquirido en la universidad. Además, este proceso ayuda a generar nuevas ideas que facilitan la vida de las personas.

- Para determinar los requerimientos y necesidades del prototipo se investigó a fondo los dispositivos con sus características técnicas como: el nivel de voltaje que manejan, las funciones que llevan a cabo, las interfaces de comunicación que usan, diagramas de conexión e integración con otros componentes. De esta forma se consiguió que estos funcionaran adecuadamente. Además, se evitó posibles errores y daños.

- Al emplear la *Raspberry Pi* como dispositivo de control de todo el prototipo se logró controlar varios *scripts* y programas de manera independiente en un mismo lugar debido a que la *Raspberry Pi* es básicamente una computadora y como tal tiene todas las características de esta, como puertos de salida *HDMI*, *USB*, Audio, entre otros. Por lo que resultó ser un dispositivo muy versátil, adaptable, accesible y escalable a las necesidades del proyecto.

- Al diseñar el sistema de control de los dispositivos se concluyó que el lenguaje de programación *Python* es muy útil, ya que posee una gran cantidad de librerías disponibles que permiten realizar casi cualquier tipo de tarea, además que tiene una curva de aprendizaje rápida e información disponible en línea, la cual se ocupó en gran parte del código de los *scripts* del prototipo.

- Al construir el sistema de las cortinas se concluyó que es útil elaborar las piezas en 3D en el programa *Fusion 360* porque, aunque requiere más tiempo de desarrollo, a la final este tiempo se ve recompensado al momento de la implementación, ya que las piezas que se imprimieron no fueron solo para una cortina sino para dos. De esta forma, a largo plazo el sistema puede replicarse varias veces según el número de cortinas que se coloquen y reemplazar las piezas dañadas rápidamente, lo que no se podría hacer si se hubieran elaborado de manera manual.

- Al realizar la programación de los *scripts* que componen el prototipo domótico se concluyó que es necesario elaborar diagramas de flujo para tener una visión más clara de lo que se va a llevar a cabo, de otro modo se puede cometer errores en el código. Además, la programación se vuelve más rápida ya que se sabe de antemano lo que se quiere desarrollar. Adicionalmente, conocer los comandos de Linux ayudó a integrar la programación de los pines *GPIO* con el sistema operativo, debido a que se necesitó instalar paquetes, configurar el sistema, crear, eliminar, mover y modificar tanto documentos como archivos.

- Al elaborar la interfaz web se concluyó que el lenguaje de programación *HTML5* es la base de las páginas web, por otra parte, con CSS se da estilos, sin embargo, el ocupar el *framework Bootstrap* facilita en gran medida el diseño ya que este consta de códigos ya establecidos para formularios, tablas, botones, alertas, entre otros. Adicionalmente, con el lenguaje *JavaScript* con la librería *JQuery* se dio funcionalidad a los botones y se envió los datos a *PHP* mediante *POST* para que se ejecuten los *scripts* de *Python*.

- La herramienta *phpMyAdmin* ayudó a la administración de la base de datos ya que en esta se almacenó la información de los usuarios y sus roles. Así mismo con las consultas a la base de datos se logró mostrar la información en los campos que se necesitaba, como es el caso de la tabla de usuarios, el formulario de actualización de usuario, formulario de *login,* entre otros, además de enviar la información del formulario de registro de nuevos usuarios.

- Se optó por el control a través de *Telegram* para el usuario encargado del mantenimiento del prototipo, ya que es una de las aplicaciones de mensajería más ocupadas en el mundo y destaca de otras por su transparencia en la información. Además, que hoy en día las aplicaciones de mensajería se ocupan diariamente para actividades cotidianas como chatear con amigos, enviar fotos, audios, videos, documentos, interactuar en grupos, realizar video llamadas, entre otros. Por ende, para el encargado del prototipo no le será difícil ni complicado conocer el funcionamiento del prototipo e interactuar con este.

- Al instalar los elementos que componen el sistema se concluyó que es importante realizar un boceto o simulación para definir la posición en que posiblemente se ubiquen los elementos, asimismo facilita para tener a la mano las herramientas a utilizar, por ejemplo, un taladro, desarmadores, tacos, martillo, ponchadora, canaleta, entre otros.

- Al realizar las pruebas de funcionamiento se concluyó que para lograr un buen funcionamiento del prototipo este debe tener tanto conexión cableada *Ethernet* como conexión inalámbrica ya que si la una falla la otra mantiene la comunicación. También se determinó que la seguridad de un cortafuegos nunca está por demás, a pesar de que el proyecto principalmente está pensado para trabajar en la red LAN.

- El diseño del prototipo se realizó de tal forma que con pequeñas mejoras pueda implementarse en la vida real o por otra parte servir de guía para las personas interesadas en conocer el funcionamiento de la *Raspberry Pi* con diferentes componentes electrónicos como: sensores, actuadores, motores, dispositivos de representación de datos, entre otros.

4.2 Recomendaciones

- Si se desea que la interfaz web funcione en cualquier lugar por medio de Internet, se recomienda emplear un proveedor de alojamiento web para tener un dominio. Sin embargo, si no se tiene esa posibilidad se puede usar *InfinityFree* el cual es un proveedor de alojamiento web gratuito con ciertas restricciones, aunque para el prototipo sería suficiente, ya que en si el prototipo no tiene la finalidad de ser un sitio web para todas las personas, más bien es de uso personal.

- Se recomienda ser cuidadoso al instalar el sistema de las cortinas y ajustar bien las piezas para que los engranajes no se traben. También es importante recalcar que la cadena de bolas debe ser de plástico ya que se utilizó al principio una cadena de bolas de metal y el sistema se trababa, aunque los motores tenían la fuerza suficiente para levantar la cortina esta no se movía.

- Si el prototipo se conecta a Internet se sugiere incrementar aún más la seguridad cambiando los puertos, asegurando el servidor *SSH*, implementando *Fail2ban* el cual bloquea las direcciones IP que hayan intentado entrar en el sistema de forma ilegal. Además, se podría crear un túnel *SSH* a través de un servidor *VPN* para conectarse remotamente a la *Raspberry Pi* de manera segura a través de Internet.

- Se recomienda usar el optoacoplador HCPL-3700 para construir un sensor de voltaje AC para el sistema de las luces, con la finalidad de que al apagar los focos desde los conmutadores se refleje el estado en la interfaz web en tiempo real. También puede ser útil hacer un circuito de *dimmer* con un TRIAC y un optoacoplador MOC3021 para regular la intensidad luminosa de ciertas áreas como pueden ser los dormitorios. De modo que en la interfaz web se tenga una barra deslizante que al moverla también varíe la intensidad del foco o desde el control remoto usar los botones para aumentar o disminuir el brillo del foco.

- Si se desea diferentes horarios de encendido y apagado durante los siete días de la semana, se tendría que crear una lista de horarios de encendido y apagado, determinar la hora de la semana actual y ajustar con la función módulo, los minutos en una semana. Por el momento, el sistema de control a través de una hora predeterminada se lo realizó solo en la luz del pasillo y se establece la hora a través del *script*. No obstante, en versiones futuras se podría automatizar más actividades con el manejo del tiempo y establecer la hora desde la interfaz web a modo de un calendario, reloj, entre otros.

- Se puede agregar cámaras IP, *USB*, entre otras, para crear un sistema más complejo en el cual se pueda almacenar las imágenes, videos y enviarlos a través del *Bot* de *Telegram*. Además, si se ocupa la biblioteca OpenCV en *Python*, se puede utilizar la inteligencia artificial para la detección de rostros y objetos. Asimismo, con esta información se podría ejecutar los *scripts* para, por ejemplo, habilitar o denegar el acceso de la puerta.

- En el conversor A/D se empleó solo 1 de los 8 canales, en los otros se pueden agregar potenciómetros, otra fotorresistencia, sensores de temperatura como el LM35 o TMP36, sensores de humedad y cualquier otro tipo de sensor analógico. Por otra parte, si con el módulo relé se quiere manejar dispositivos con mayores corrientes se puede conectar a este un contactor para reestablecer o interrumpir la corriente eléctrica de la carga. Adicionalmente sería útil instalar un suministro de energía alternativo a todo el prototipo para evitar que el sistema deje de funcionar cuando exista un corte de corriente de la red eléctrica pública.

- Al establecer los valores de la frecuencia de la señal *PWM* tener en cuenta que, si se utiliza valores muy pequeños de frecuencia, la señal estará intermitente y si son muy altos la medida será ligeramente diferente de la frecuencia proporcionada como parámetro de la *Raspberry Pi*. Por lo tanto, se recomienda 100 (Hz), si se emplea la señal en un *LED* y 1000 (Hz) para otros dispositivos como el ventilador y servomotores. Aunque la *Raspberry Pi* tenga solo dos canales *PWM* estos se pueden expandir a través de *I2C*, por ejemplo, con el módulo PCA9685 el cual controla 16 salidas *PWM*.

- Es importante en el desarrollo de la interfaz web no olvidarse de destruir la sesión con el comando *session_destroy();* y cerrar la conexión con el comando *mysqli_close();* cada vez que se establece conexión con la base de datos ya que esto evita posibles brechas de seguridad.

- Para ampliar el sistema actual se puede ocupar el modelo maestro-esclavo, siendo el maestro la *Raspberry Pi* encargada de representar los datos y los esclavos los microcontroladores. Adicionalmente, la conexión se podría establecer con estos dispositivos de manera inalámbrica por *Bluetooth* para funciones de corto alcance o XBEE para largo alcance. Sin embargo, si se realiza de manera cableada, puede realizarse la comunicación con los dispositivos por *Ethernet* o por las interfaces *SPI*, *I2C*, Serial, entre otras.

- En una nueva versión del *Bot* de *Telegram* se puede añadir la función de almacenar información del usuario mediante una base de datos como, por ejemplo, *MongoDB Atlas*. También cambiar el método *Polling* del *script* del *Bo*t al de *Webhooks* y alojar el *Bot* en *Heroku* la cual es una plataforma que ayuda al despliegue de proyectos de *software.*

- Sería recomendable que se continúe desarrollando este proyecto a futuro en otras actividades que vayan más allá de ser un sistema domótico, como desarrollar un sistema de vivero inteligente, un sistema de detección de incendios, un sistema de tienda comercial con el *Bot* de *Telegram*, un sistema para una institución educativa en la cual se requiera compartir información o documentos de manera automática, entre otros.

- Un tema interesante que se podría considerar es utilizar el *framework Django* el cual permite el desarrollo web en lenguaje *Python* de manera que se podría integrar la interfaz web, el *Bot* de *Telegram*, y todos los *scripts* en un mismo lugar.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] E. Rodrígues, «Xataka,» 15 05 2018. [En línea]. Available: https://www.xataka.com/makers/cero-maker-todo-necesario-para-empezar-raspberry-pi. [Último acceso: 04 08 2021].

[2] M. Fernández, «Taller de Raspberry Pi,» 17 05 2017. [En línea]. Available: https://franciscomoya.gitbooks.io/taller-de-raspberry-pi/content/es/. [Último acceso: 11 06 2021].

[3] P. Villegas, «Mecafenix,» 19 06 2019. [En línea]. Available: https://www.ingmecafenix.com/electricidad-industrial/motor-paso-a-paso/. [Último acceso: 27 08 2021].

[4] L. Valles, «Área tecnología,» 18 05 2020. [En línea]. Available: https://www.areatecnologia.com/mecanismos/engranajes.html. [Último acceso: 27 08 2021].

[5] L. Macas, «Solectro,» 19 04 2020. [En línea]. Available: https://solectroshop.com/es/blog/guia-para-principiantes-sobre-modulos-de-reles-en-los-proyectos-de-arduino-n28. [Último acceso: 27 08 2021].

[6] M. Almagro, «Naylamp,» 17 09 2020. [En línea]. Available: https://naylampmechatronics.com/drivers/63-driver-pap-pololu-a4988.html. [Último acceso: 27 08 2021].

[7] Waveshare, «7inch HDMI LCD (C),» 12 05 2020. [En línea]. Available: https://www.waveshare.com/wiki/7inch_HDMI_LCD_(C). [Último acceso: 27 08 2021].

[8] R. Lozano, «Talos Electronics,» 19 08 2020. [En línea]. Available: https://www.taloselectronics.com/blogs/tutoriales/como-usar-pantalla-lcd-con-i2c-con-raspberry. [Último acceso: 18 06 2021].

[9] M. F. Perez, «Aprendiendo Arduino,» 13 11 2016. [En línea]. Available: https://aprendiendoarduino.wordpress.com/2016/11/13/bus-spi/#comments. [Último acceso: 18 06 2021].

[10] S. García, «330ohms,» 07 07 2020. [En línea]. Available: https://blog.330ohms.com/2020/07/07/como-conectar-arduino-y-raspberry-pi-por-comunicacion-serial/. [Último acceso: 18 06 2021].

[11] M. Chalco, «Programador clic,» 10 05 2020. [En línea]. Available: https://programmerclick.com/article/3482327083/. [Último acceso: 04 08 2021].

[12] M. Zuñiga, «Programo Ergo Sum,» 12 02 2021. [En línea]. Available: https://www.programoergosum.com/cursos-online/raspberry-pi/244-iniciacion-a-python-en-raspberry-pi/que-es-python. [Último acceso: 04 08 2021].

[13] L. Villanueva, «MNP,» 27 07 2016. [En línea]. Available: https://www.mnp.cl/es/post/que-es-board-bcm-raspberry-pi. [Último acceso: 27 06 2021].

[14] D. López, «Electrónica de Invierno,» 20 09 2011. [En línea]. Available: https://electronicavm.wordpress.com/2011/09/20/recepcion-ir-con-arduino-protocolo-nec/. [Último acceso: 12 07 2021].

[15] Telegram, «Telegram.org,» 12 02 2021. [En línea]. Available: https://core.telegram.org/bots/api. [Último acceso: 04 08 2021].

[16] O. Solaris, «Guía de administración del sistema: administración avanzada,» 14 05 2011. [En línea]. Available: https://docs.oracle.com/cd/E38897_01/html/E23086/sysrescron-1.html. [Último acceso: 23 07 2021].

[17] J. Vallas, «Zeppelinux,» 18 01 2019. [En línea]. Available: https://www.zeppelinux.es/como-ejecutar-un-script-o-programa-al-iniciar-linux-con-rc-local/. [Último acceso: 27 08 2021].

[18] W. Jianshu, «Programmerclick,» 12 04 2019. [En línea]. Available: https://programmerclick.com/article/69571078066/. [Último acceso: 27 07 2021].

[19] T. A. S. Foundation, «Apache,» 12 01 2021. [En línea]. Available: https://httpd.apache.org/. [Último acceso: 28 07 2021].

[20] L. Franco, «Web Hosting,» 17 05 2020. [En línea]. Available: https://www.inc.cl/blog/hosting/que-es-phpmyadmin. [Último acceso: 28 07 2021].

[21] L. Manz, «Lenguaje CSS,» 14 06 2017. [En línea]. Available: https://lenguajecss.com/css/introduccion/que-es-css/. [Último acceso: 29 07 2021].

[22] M. Valdiviezo, «MDN Web Docs,» 19 06 2020. [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Último acceso: 29 07 2021].

[23] L. Chuburu, «Laurachuburu_Tutoriales,» 12 07 2019. [En línea]. Available: https://www.laurachuburu.com.ar/tutoriales/que-es-jquery-y-como-implementarlo.php. [Último acceso: 03 08 2021].

[24] L. Vargas, «Infranetworking,» 12 04 2019. [En línea]. Available: https://www.infranetworking.com/md5. [Último acceso: 03 08 2021].

[25] M. a. i. contributors, «MDN Web Docs Store,» 16 01 2021. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API. [Último acceso: 10 08 2021].

[26] G. Borja, «Hostinger tutoriales,» 20 05 2021. [En línea]. Available: https://www.hostinger.es/tutoriales/que-es-ajax. [Último acceso: 10 08 2021].

[27] E. R. D. LUIS, «Xataka,» 02 08 2020. [En línea]. Available: https://www.xataka.com/seleccion/que-modelo-raspberry-pi-comprar-repaso-a-principales-placas-proyectos-habituales-para-dar-mejor. [Último acceso: 10 06 2021].

[28] A. A. Huertos, «Computer Hoy,» 06 02 2020. [En línea]. Available: https://computerhoy.com/listas/tecnologia/debes-saber-raspberry-pi-4-antes-lanzarte-comprar-446889. [Último acceso: 10 06 2021].

[29] A. Rosero, «Mejor Antivirus Ahora,» 28 09 2016. [En línea]. Available: https://mejorantivirusahora.com/802-11ac-vs-802-11n-cul-es-la-diferencia/. [Último acceso: 10 06 2021].

[30] C. Y. y. A. Calderín, «Proyectos Raspberry Pi,» Raspberry Pi Foundation , 12 04 2019. [En línea]. Available: https://projects.raspberrypi.org/es-LA/projects/raspberry-pi-setting-up/6. [Último acceso: 11 06 2021].

[31] R. Larcos, «Mecatronica Uno,» 12 01 2019. [En línea]. Available: https://mecatronicauno.com/usar-raspberry-pi-con-memoria-sd-de-64gb-o-mas/. [Último acceso: 11 06 2021].

[32] L. Moya, «GeekFactory,» 10 01 2021. [En línea]. Available: https://www.geekfactory.mx/tienda/raspberry-pi/carcasa-de-aluminio-disipador-de-calor-raspberry-pi-4/. [Último acceso: 14 06 2021].

[33] L. Diéguez, «Kolwidi,» 28 03 2019. [En línea]. Available: https://kolwidi.com/blogs/blog-kolwidi/los-10-mejores-sistemas-operativos-para-tu-raspberry-pi. [Último acceso: 14 06 2021].

[34] L. Villanueva, «Raspberry Pi web info,» 12 05 2019. [En línea]. Available: https://rpi.uroboros.es/segurid.html. [Último acceso: 21 06 2021].

[35] L. Matías, «DescubreArduino,» 12 05 2018. [En línea]. Available: https://descubrearduino.com/pines-gpio-de-la-raspberry-pi-4/. [Último acceso: 22 06 2021].

[36] Python, «Introducción a la programación con Python,» 29 03 2019. [En línea]. Available: https://www.mclibre.org/consultar/python/lecciones/python-biblioteca-time.html#:~:text=Esta%20funci%C3%B3n%20nos%20permite%20cronometrar,%22%22)%20final%20%3D%20time.. [Último acceso: 29 06 2021].

[37] L. Villagram, «W3big,» 14 06 2019. [En línea]. Available: http://www.w3big.com/es/python/att-time-localtime.html. [Último acceso: 29 06 2021].

[38] E. Narvaez, «DelftStack,» 03 30 2021. [En línea]. Available: https://www.delftstack.com/es/howto/python/python-sleep-ms/. [Último acceso: 29 06 2021].

[39] L. Leopoldo, «Covantec,» 12 05 2019. [En línea]. Available: https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion7/archivos.html. [Último acceso: 01 07 2021].

[40] N. Valdiviezo, «GeeksforGeeks,» 10 06 2021. [En línea]. Available: https://www.geeksforgeeks.org/python-classes-and-objects/. [Último acceso: 02 07 2021].

[41] A. Ortiz, «EduPython,» 05 03 2016. [En línea]. Available: http://edupython.blogspot.com/2016/07/midiendo-distancias-el-sensor.html. [Último acceso: 02 07 2021].

[42] S. Alarcón, «Manual de Python,» 16 07 2019. [En línea]. Available: https://aprendeconalf.es/docencia/python/manual/numpy/. [Último acceso: 06 07 2021].

[43] Microchip, «Converters with SPI Serial Interface,» 14 05 2008. [En línea]. Available: https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf. [Último acceso: 06 07 2021].

[44] D. Pérez, «Tecnonucleous,» 18 01 2018. [En línea]. Available: https://tecnonucleous.com/2018/01/28/como-instalar-pip-para-python-en-windows-mac-y-linux/. [Último acceso: 07 08 2021].

[45] F. Yépez, «KoalaSoft,» 25 05 2009. [En línea]. Available: https://koalasoft.wordpress.com/2009/07/24/comando-net-rpc-para-apagar-o-reiniciar-una-maquina-remota-de-windows-desde-una-terminal-unix/. [Último acceso: 08 07 2021].

[46] A. Boncut, «W3resource,» 01 09 2020. [En línea]. Available: https://www.w3resource.com/python-exercises/python-basic-exercise-3.php. [Último acceso: 12 07 2021].

[47] L. Jaramillo, «Mercotecnia,» 12 05 2020. [En línea]. Available: https://www.mercotecnia.cl/tienda/control-remoto-24-botones-solo-control/. [Último acceso: 12 07 2021].

[48] P. Sandoval, «Revista Española Electrónica,» 02 01 2021. [En línea]. Available: https://www.redeweb.com/articulos/conexion-de-telegram-con-raspbery-pi/. [Último acceso: 13 07 2021].

[49] Python, «PythonRepo,» 07 13 2021. [En línea]. Available: https://pythonrepo.com/repo/eternnoir-pyTelegramBotAPI-python-third-party-apis-wrappers. [Último acceso: 15 07 2021].

[50] S. Martínez, «Geek Theory,» 12 05 2016. [En línea]. Available: https://geekytheory.com/telegram-programando-un-bot-en-python. [Último acceso: 15 07 2021].

[51] V. Gite, «NixCraft,» 30 05 2021. [En línea]. Available: https://www.cyberciti.biz/faq/linux-find-out-raspberry-pi-gpu-and-arm-cpu-temperature-command/. [Último acceso: 16 07 2021].

[52] R. P. FOUNDATION, «Raspberrypi,» 12 03 2020. [En línea]. Available: https://www.raspberrypi.org/documentation/raspbian/applications/vcgencmd.md. [Último acceso: 16 07 2021].

[53] V. Gite, «nixCraft,» 19 07 2020. [En línea]. Available: https://www.cyberciti.biz/faq/linux-check-memory-usage/. [Último acceso: 16 07 2021].

[54] D. Garcia, «Sistemes14.,» 02 01 2013. [En línea]. Available: https://www.sistemes14.com/2013/01/crontab-ejemplos-programar-tareas-linux/. [Último acceso: 23 07 2021].

[55] M. Coloma, «Institut Puig Castellar,» 12 04 2020. [En línea]. Available: https://elpuig.xeill.net/Members/rborrell/articles/los-archivos-bashrc-bash_profile-etc-bashrc-etc-profile-los-archivos-bashrc-bash_profile-etc-bashrc-etc-profile-cual-utilizar. [Último acceso: 26 07 2021].

[56] M. Ebrahim, «LikeGeeks,» 02 21 2017. [En línea]. Available: https://likegeeks.com/es/comando-awk/. [Último acceso: 26 07 2021].

[57] J. Vélez, «Electrónica Unicrom,» 12 03 2019. [En línea]. Available: https://unicrom.com/ley-de-ohm-potencia-electrica/. [Último acceso: 24 08 2021].

[58] L. Llamas, «Tutoriales arduino,» 26 08 2016. [En línea]. Available: https://www.luisllamas.es/motores-paso-paso-arduino-driver-a4988-drv8825/. [Último acceso: 25 08 2021].

[59] MicroSystems.LLC, «Datasheet A4988,» 12 02 2014. [En línea]. Available: https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf. [Último acceso: 25 08 2021].

[60] J. C. Estrada, «La casa de la lámpara,» 19 04 2021. [En línea]. Available: https://www.lacasadelalampara.com/que-es-en-iluminacion-un-lumen-o-lux-iluminacion-por-zonas/. [Último acceso: 25 08 2021].

[61] P. Villamizar, «Electrotec,» 12 04 2020. [En línea]. Available: https://electrotec.pe/blog/TiposDeCables. [Último acceso: 26 08 2021].

[62] J. Osorio, «Cableado Estructurado,» 06 09 2010. [En línea]. Available: https://es.slideshare.net/jorge_613/cableado-estructurado-5142635. [Último acceso: 26 08 2021].

[63] B. W. B. Engine, «User Manual,» 16 02 2021. [En línea]. Available: https://fccid.io/2AODVBEAKR1601/User-Manual/User-Manual-3773972. [Último acceso: 27 08 2021].

[64] Y. Fernández, «Xataka Basics,» 05 05 2020. [En línea]. Available: https://www.xataka.com/basics/que-fps-fotogramas-segundo-sirven-videojuegos. [Último acceso: 10 08 2021].

[65] S. Sistemas, «Solvetic.com,» 26 05 2021. [En línea]. Available: https://www.solvetic.com/tutoriales/article/3500-como-abrir-editor-registro-en-windows-10-8-7/. [Último acceso: 13 08 2021].

[66] R. Santos, «rafaelsantos.es,» 18 02 2021. [En línea]. Available: https://www.rafaelsantos.es/web/informatica/windows/55-cancelar-la-caducidad-de-la-contrasena-de-windows. [Último acceso: 14 08 2021].

[67] M. Legña, «FLOPY,» 21 07 2019. [En línea]. Available: https://www.flopy.es/crea-un-bot-de-telegram-para-tu-raspberry-ordenale-cosas-y-habla-con-ella-a-distancia/. [Último acceso: 13 07 2021].

[68] A. Olmos, «Overant,» 03 10 2018. [En línea]. Available: https://www.overant.com/blog/diferencias-entre-mysql-y-mariadb/. [Último acceso: 28 07 2021].

[69] T. P. Group, «PHP,» 14 03 2021. [En línea]. Available: https://www.php.net/manual/es/intro-whatis.php. [Último acceso: 28 07 2021].

[70] D. Jaramillo, «Cuaderno de Clase,» 23 03 2017. [En línea]. Available: http://janda1617smr2curro.blogspot.com/2017/03/que-es-xampp-y-para-que-sirve.html. [Último acceso: 28 07 2021].

[71] F. Ángeles, «Universidad autónoma del estado de Hidalgo,» 03 05 2019. [En línea]. Available: https://www.uaeh.edu.mx/scige/boletin/prepa3/n8/m1.html. [Último acceso: 29 07 2021].

[72] J. D. P. Jiménez, «Openwebinars,» 20 01 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-html5/. [Último acceso: 29 07 2021].

[73] T. Pérez, «NeoAttack,» 17 02 2019. [En línea]. Available: https://neoattack.com/neowiki/photoshop/. [Último acceso: 29 07 2021].

[74] D. Lázaro, «GET y POST en PHP,» 14 05 2018. [En línea]. Available: https://diego.com.es/get-y-post-en-php. [Último acceso: 03 08 2021].

[75] J. Montaño, «Aqui hay dominios,» 23 05 2014. [En línea]. Available: https://www.aquihaydominios.com/blog/font-awesom-que-es-y-como-se-usa/. [Último acceso: 03 08 2021].

[76] H. Russo, «Geek's Room,» 13 07 2020. [En línea]. Available: https://geeksroom.com/2018/01/undraw-ilustraciones/115150/?cn-reloaded=1. [Último acceso: 03 08 2021].

[77] A. Ramos, «EVILNAPSIS,» 16 08 2018. [En línea]. Available: https://evilnapsis.com/2018/08/16/jquery-instalar-y-usar-el-plugin-sweetalert-2/. [Último acceso: 03 08 2021].

[78] M. Rivas, «winscp.net,» 17 09 2014. [En línea]. Available: https://winscp.net/eng/docs/lang:es. [Último acceso: 04 08 2021].

[79] M. Cruz, «Oficina de ceguridad internauta,» 20 01 2019. [En línea]. Available: https://www.osi.es/es/herramientas-gratuitas/vnc-viewer. [Último acceso: 04 08 2021].

[80] J. Alba, «Can I use,» 05 06 2021. [En línea]. Available: https://caniuse.com/?search=pattern. [Último acceso: 05 08 2021].

[81] L. Guest, «Rockcontent.,» 12 04 2020. [En línea]. Available: https://rockcontent.com/es/blog/bootstrap/. [Último acceso: 05 08 2021].

[82] M. Villagram, «Workana,» 12 03 2019. [En línea]. Available: https://i.workana.com/glosario/que-es-un-dashboard/. [Último acceso: 05 08 2021].

[83] M. Hernández, «Oficina de software,» 30 05 2014. [En línea]. Available: https://oshl.umh.es/2014/05/30/fritzing-un-programa-open-source-para-el-diseno-electronico/. [Último acceso: 11 08 2021].

[84] A. Malta, «3D Natives,» 29 04 2020. [En línea]. Available: https://www.3dnatives.com/es/fusion-360-software-290420202/#!. [Último acceso: 11 08 2021].

[85] A. Bravo, «uptodown,» 12 05 2018. [En línea]. Available: https://usb-image-tool.uptodown.com/windows. [Último acceso: 14 08 2021].

[86] F. Rivas, «Electricaaplicada,» 12 07 2019. [En línea]. Available: https://www.electricaplicada.com/potencia-consumo-equipos-electricos/. [Último acceso: 26 08 2021].

[87] G. M. Smith, «Dewesoft,» 12 04 2020. [En línea]. Available: https://dewesoft.com/es/daq/que-es-un-sensor. [Último acceso: 27 08 2021].

[8 L. Hidalgo, «Solvetic,» 09 03 2020. [En línea]. Available:
8] https://www.solvetic.com/tutoriales/article/1773-%C2%BFqu%C3%A9-es-systemd/.
[Último acceso: 27 08 2021].

[8 M. Cajas, «Uptodown,» 19 04 2020. [En línea]. Available: https://dialux-
9] evo.uptodown.com/windows. [Último acceso: 27 08 2021].

ANEXOS

ANEXO 1: CÓDIGOS DE PROGRAMACIÓN DE LOS DISPOSITIVOS

ANEXO 2: CÓDIGOS DE PROGRAMACIÓN DEL *BOT* DE *TELEGRAM*

ANEXO 3: CÓDIGOS DE PROGRAMACIÓN DE LA INTERFAZ WEB

ANEXO 4: INFORME DE LA SIMULACIÓN DE *DIALUX EVO*