ÍNDICE GENERAL

1 INTRODUCCIÓN

1.1 Antecedentes 1
1.2 Problemática 1
1.3 Justificación 2
1.4 Objetivo 4
1.4.1 Objetivo general 4
1.4.2 Objetivos específicos 4
1.5 Hipótesis 4
1.6 Organización de la tesis 4

2 ANTECEDENTES

2.1 Introducción 6
2.2 Requisitos del sistema en la Facultad de Ingeniería 6
2.3 Casos de estudio 7
2.3.1 Caso 1. Control de estacionamiento en Móstoles 7
2.3.2 Caso 2. Acceso residencial por reconocimiento dactilar 8
2.3.3 Caso 3. Acceso a la Facultad de Perú por RFID y cámara 9
2.3.4 Caso 4. Caseta de cobro por RFID 10
2.4 Comparativa 11

3 METODOLOGÍA

3.1 Introducción 14
3.2 Descripción del sistema 14
3.2.1 Configuración de la Raspberry 15
3.2.1.1 Instalación de la pila LAMP, PuTTy y FileZilla 15
3.2.1.2 Configuración de los GPIO y comunicación SPI 17
3.2.1.3 Instalación de OpenCV 18
3.2.1.4 Otras configuraciones 19
3.2.2 Enfoque de estudio sobre etiquetas RFID 20
3.2.3 Protocolo de comunicaciones SPI 20
3.2.4 Propuesta del reconocimiento óptico de placas vehiculares con la Raspberry 24
3.3 Propuestas de diseño 27

4 RESULTADOS

4.1 Configuración del sistema para la Raspberry 36
4.2 Composición y armado del sistema físico 40
4.3 Programación y algoritmos del sistema 42
4.4 Funcionamiento del sistema 47
4.5 Costo del sistema diseñado 49

CONCLUSIONES 50

BIBLIOGRAFIA 51

GLOSARIO 53

1. INTRODUCCIÓN

1.1 Antecedentes

El transportarse por medio de automóviles se ha convertido en un medio que los seres humanos utilizan día a día. Por lo que, los estacionamientos juegan un papel importante en este estilo de vida. Esto ha provocado que, con el paso del tiempo, aumente la demanda por los espacios dentro de estacionamientos. Sánchez (2020) afirma: “En noviembre de 2020, se exportaron desde México 287 mil 703 automóviles, un incremento de 4.7 por ciento en comparación con el mismo mes del año pasado”. Esto significa que existe una demanda por los automóviles y que sin duda alguna el uso de los estacionamientos es indispensable.

Con el tiempo, han surgido varias tecnologías de autoidentificación. Entre las numerosas aplicaciones podemos mencionar la gestión del acceso del personal. La protección del acceso a establecimientos privados, actualmente gestionada por sus respectivas unidades administrativas, genera problemas en el correcto control del registro de los usuarios que tienen acceso a sus establecimientos y los horarios de atención que establecen.

En algunos lugares se utilizan infraestructuras costosas, lo que supone un grave problema para la instalación en algunos accidentes. La necesidad de implementar alternativas que permitan solucionar el problema del control de acceso aprovechando la tecnología actual para maximizar su capacidad.

1.2 Problemática

La cantidad de espacios para estacionamiento en la Facultad de Ingeniería no resulta suficiente para la demanda que existe entre miembros de la institución y miembros de instituciones vecinas. Encontrar un espacio al momento de llegar, resulta un poco tardío y esto implica que los alumnos, personal docente y administrativos de la institución deban llegar con anticipación con los horarios de entrada preestablecidos.

Esto ocasiona que tanto alumnos como personal de la institución pierdan tiempo desplazándose a otras facultades para hacer uso de sus estacionamientos. Al igual es una

pérdida de dinero para la Universidad del Estado de México (UAEMex) si se traduce en horas-humanos.

El precio de un sistema de acceso vehicular con lectura de tarjetas RFID comercial tiene un costo excesivamente elevado. La UAEMex tiene contratado servicios de empresas particulares, por lo que, incrementan aún más los costos.

1.3 Justificación

El uso de automóviles en zonas urbanas ha tomado una gran relevancia, como consecuencia, la Facultad de Ingeniería al encontrarse inmersa dentro de una, posee un número considerable de usuarios de vehículos automotor. En ese sentido, existen espacios designados y localizados cerca de cada institución para facilitar el acceso. Sin embargo, hay un gran número de estudiantes que ocupan los espacios de otras facultades, al igual, alumnos que ingresan a los espacios de los profesores.

De acuerdo con información obtenida del departamento de Control Escolar de la Facultad de Ingeniería (Ingeniería, 2022), se obtiene la Figura 1 que demuestra el número de alumnos y profesores.

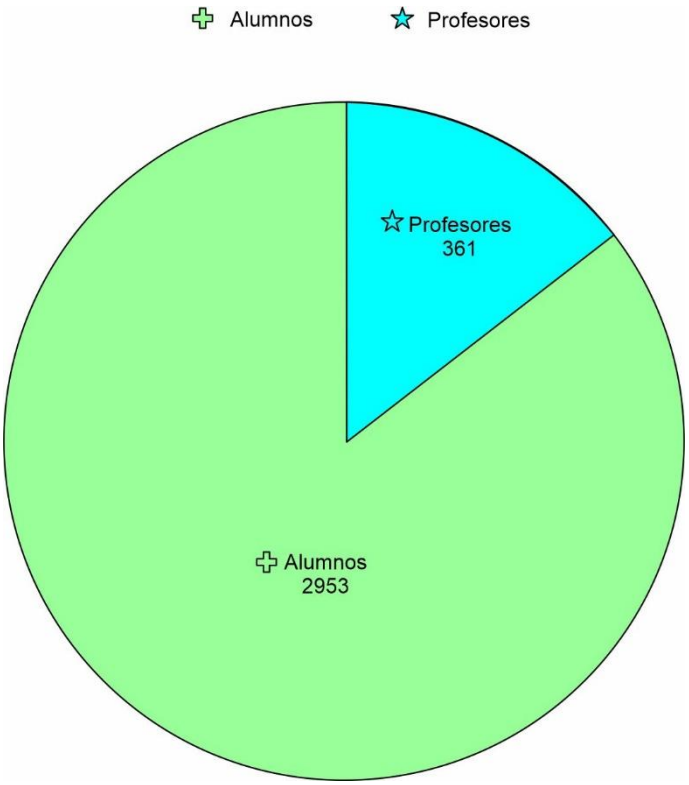

Figura 1 *Esquema de la población en la Facultad de Ingeniería tomando la información de Alumnos / Profesores. (Elaboración propia)*

El estacionamiento de la Facultad de Ingeniería cuenta con 105 espacios para el área de alumnos y 70 espacios para el área de profesores. Suponiendo, que 614 alumnos y 270 profesores utilizan automóviles, estos valores los dividimos, uno para el turno matutino y otro para el turno vespertino con un valor cada uno de 307 alumnos y 135 profesores. Se hace la siguiente suposición representada en la Figura 2. Se tomarán en cuenta dos casos ideales en donde no hay vehículos de otras facultades en el estacionamiento de la Facultad de Ingeniería.

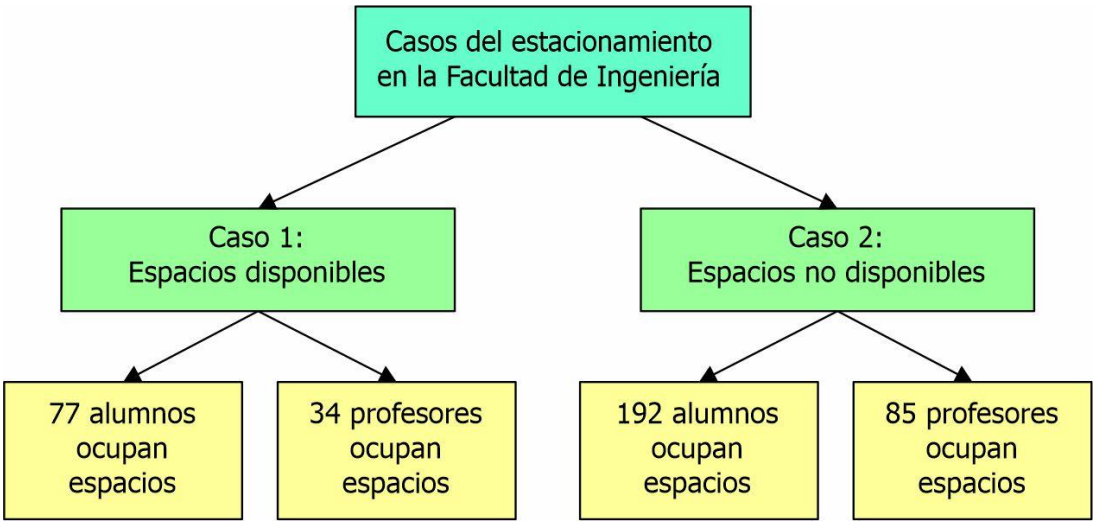

Figura 2 *Esquema de los casos que se puede presentar el estado del estacionamiento de la Facultad de Ingeniería. (Elaboración propia)*

En el caso uno se presenta las primeras y últimas horas del día (se ocupa ¼ del estacionamiento). El caso 2, donde, interceptan cambios de turno (hora pico), se pudieran juntar ambos turnos, con un poco más de una cuarta parte de cada turno (192 alumnos y 85 profesores), ya no existe espacio suficiente dentro de CU (Ciudad Universitaria), es ahí donde 38 alumnos y 15 profesores utilizan las calles como estacionamiento esperando un lugar dentro de CU.

Es por eso la importancia de tener un control del acceso vehicular a la Facultad de Ingeniería, ya que en la vida real existe un gran número de personas que no respetan sus espacios designados por facultades, aumentando las cifras y el riesgo de golpes y robos fuera de las instalaciones de la universidad.

El proyecto de un sistema de acceso vehicular involucra una parte de detección de tarjetas RFID (procesamiento de la información y gestión de la base de datos) y una parte de potencia con la parte electromecánica para el levantamiento de la pluma. Este trabajo de titulación solo abarca desde la lectura de la tarjeta RFID hasta la gestión de la base de

datos y así enviar una señal digital que se acoplara a una posterior etapa de potencia. La descripción de esa etapa de potencia queda más allá de los límites de esta tesis.

1.4 Objetivos

1.4.1 Objetivo general

Diseñar y construir un prototipo de la etapa de control para el acceso vehicular en la Facultad de Ingeniería para gestionar el acceso exclusivo.

1.4.2 Objetivos específicos

- Desarrollar un sistema que permita realizar el control de acceso de vehicular, utilizando la tecnología de identificación por radiofrecuencia.
- Involucrar en el sistema tarjetas de radiofrecuencia para identificar al personal.
- Utilizando Raspberry Pi, implementar un conjunto de aplicaciones cliente-servidor que le permitan controlar y gestionar el funcionamiento del sistema de acceso.
- Obtener un sistema funcional y de bajo costo.
- Desarrollar un acceso en la web que permita explotar los datos generados, de forma remota.

1.5 Hipótesis

El precio de la etapa de lectura de tarjetas RFID y gestión de una base de datos para el acceso vehicular de este trabajo es al menos 6 veces más barato con respecto a un sistema comercial de características similares.

1.6 Organización de la tesis

Esta tesis contiene cuatro capítulos, en los cuales se propone un diseño y analiza los costos que ofrece la tecnología RFID aplicado a la gestión del acceso vehicular a un estacionamiento. Se presentan los elementos que participan en un proyecto de este tipo y aunque en esta ocasión está orientado al control de acceso, la mayoría de las cuestiones estudiadas son aplicables a muchos casos. Se analiza el ciclo completo de este tipo de sistemas, comenzando desde el momento en que se genera la información desde el lector RFID; para el procesamiento de datos, desde servidores de clientes y aplicaciones web.

Es por lo que esta tesis pretende ser una guía de los elementos que se deben considerar en un proyecto RFID. A continuación, se describe como cada capítulo contribuye a alcanzar este objetivo.

2. ANTECEDENTES

2.1 Introducción

A medida que pasa el tiempo han surgido distintas tecnologías orientadas a la identificación y control de vehículos en estacionamientos, este capítulo proporciona un análisis de algunas tecnologías de identificación existentes y las compara con algunos casos de estudio en los que se menciona primero una descripción de sus características así como sus ventajas y desventajas para después compararlas con los requisitos específicos del estacionamiento en la Facultad de Ingeniería de la Universidad Autónoma del Estado de México.

Existen distintos sistemas de contabilización de vehículos en estacionamientos que se pueden clasificar según su método en tarjetas magnéticas, sistemas biométricos, código de barras, tarjetas de radiofrecuencia, RFID y memorias de contacto que se describen a continuación. El acceso con sistemas biométricos realiza a través del análisis y medición de características físicas tales como el reconocimiento del iris, la reflexión retinal, la geometría de la mano o geometría facial, etcétera. En lo que respecta al acceso con tarjetas magnéticas se basan en la lectura de una banda magnética mientras que al acceso con tarjetas de código de barras consiste en decodificar los datos contenidos en una imagen formada por combinaciones de barras y espacios el acceso por la identificación por radiofrecuencia consiste en asignar un código de información a un producto proceso persona para identificar o acceder a información adicional al respecto (Alvarado, 2008).

2.2 Requisitos del sistema en la Facultad de Ingeniería

El diseño del sistema debe satisfacer las condiciones del espacio designado en la Facultad de Ingeniería; las cuales, debe aplicarse en un espacio abierto y no techado, debe contar con acceso RFID. El acceso del vehículo es permitido por una barrera vehicular o pluma, la cual sólo da acceso mediante el registro WEB de alumnos y personal laboral de la institución, también debe existir la opción de registrar altas temporales (conferencistas, profesores invitados, personal de rectoría, proveedores, etc.). El sistema debe comunicarse con un servidor mediante Raspberry Pi, usando el protocolo de comunicación SPI.

2.3 Casos de estudio

2.3.1 Caso 1. Control de estacionamiento en Móstoles

En la tesis titulada “Sistema de control de estacionamientos de la policía de Móstoles” describe un sistema de acceso vehicular para la policía mediante el uso de una plataforma en la nube a través de Microsoft Azure usando la tecnología de un servicio de dominio de las smart cities como servidor. Con el fin de que las ciudades modernas puedan orientarse a mejorar el confort (servicio) de los ciudadanos, mientras respetan los aspectos del medioambiente (Sánchez, 2016).

El objetivo de este sistema es que se coloque un chip en cada auto de la ciudad y un sensor en cada parqueadero, y por medio de una aplicación se busquen los lugares disponibles, ocupados y los que invaden lugares que no les correspondan como para autos de carga o minusválidos, facilitando el trabajo del personal de policía.

Los datos son en tiempo real y se almacenan en la nube, lo cual ayuda a un mejor monitoreo de lo que ocurre en la ciudad. Los policías tendrán acceso a esta información y recibirán una notificación para acudir al parqueadero cuando ocurra un problema, como, cuando alguien este mal estacionado o invada un parqueadero especial.

Ventajas:

- Servidor en la nube (reduce costos).
- Optimizar el trabajo de los policías.
- Mayor seguridad y control en los parqueaderos.
- Tiene ayuda del sistema de una Smart Citie.

Desventajas:

- Instalación de sensores y chips.
- Requiere mantenimiento
- Sistema disponible solo para Smart Cities.
- El sistema pertenece a la nube por lo que puede ser hackeado.
- Uso de sensores (aumenta costos).
- Caída del sistema.

Como en este sistema está desarrollado por proveedores que ofrecen una plataforma (Microsotf Azure) de aplicaciones para la nube y que esta alojada en los data centers de

Microsoft tiene una rentabilidad. Los planes de App Service admiten 5 niveles de precios diferentes: Gratis, Compartido, Básico, Estándar y Premium.

2.3.2 Caso 2. Acceso residencial por reconocimiento dactilar

En la tesis que lleva por título “Sistema web de registro y control vehicular por medio de autenticación biométrica con acceso móvil” se describe un sistema que permite gestionar el acceso de vehículos en un conjunto residencial por medio de una aplicación móvil. El sistema fue desarrollado en Android Studio usando servidores de Firebase en tiempo real con almacenamiento en la nube el cual se puede acceder por reconocimiento dactilar para realizar el ingreso del usuario.

El objetivo de dicha tesis fue que mediante una base de datos se permita el acceso a una residencia (propietarios, visitantes y vehículos) en la cual se debe realizar un previo registro mediante una página web, con ayuda de una aplicación móvil que permite controlar el acceso y un módulo que refleja la información sobre los propietarios, visitantes y vehículos. Todo esto con la huella dactilar. También se implementa un sistema de encriptación de información para la seguridad del sistema (Rincón y Laguna, 2018).

Una de las ventajas más importantes que se refleja es que al sistematizar el acceso se llevará un control más organizado en cuanto al manejo de entradas y salidas. Con una plataforma de fácil manejo, cómodo y seguro.

El administrador puede manejar el sistema para realizar modificaciones de datos que los propietarios dispongan. Y los guardias de seguridad podrán tener total seguridad de los datos que se muestran en pantalla, los cuales son los que permiten el acceso manual a la residencia.

Una desventaja de este sistema es que cualquier versión de Android inferior a la 6.0 no podrá hacer uso del sistema de la aplicación móvil. Esto nos lleva a algunas limitantes, como contar con un equipo móvil, y este equipo móvil deberá tener forzosamente sistema Android 6.0 y superiores. Otra limitante es que su celular o equipo móvil debe contar con un sensor de huellas, ya que sin este el principal uso de la aplicación no servirá. Además, se debe estar conectado a internet para poder acceder.

Este proyecto plantea una solución económica para la seguridad de un conjunto residencial.

Los gastos para la implementación de este sistema son (Rincón y Laguna, 2018):

1. Recursos técnicos (computadores, servicios de energía, servicio de internet, celular, papelería e impresiones): MX$2,790.00
2. Recursos humanos (asesoría para realizar el proyecto, programadores): MX$14,000.00
3. Costos imprevistos: MX$1,591.00

Total, de: MX$18,381.00

2.3.3 Caso 3. Acceso a la Facultad de Perú por RFID y cámara

En la tesis que lleva por nombre "Diseño de un sistema de acceso vehicular a la PUCP en tecnología RFID y detección de placas vehiculares" se describe un sistema de acceso vehicular para la Universidad Católica de Perú para la gestión de los espacios disponibles y el acceso único a personal de la universidad. En este trabajo de tesis se plantean dos tecnologías, RFID y detección de placas vehiculares. Todas estas tecnologías fueron unificadas en una base de datos por medio de una plataforma basada en lenguaje de programación Java mediante librerías de conexión y una interfaz final para la gestión de accesos, que sería utilizada por el personal de seguridad encargados del acceso vehicular. El error obtenido fue por debajo del 7% en detección de placas vehiculares, que afirma que se puede identificar correctamente a los usuarios que ingresen a la universidad (Gomero, 2017).

El sistema de este proyecto cuenta con una cámara "DCS-3715[103]" con un precio de US$1090.00, una barrera o pluma "LiftMaster - Mega Arm" con un precio de US$1520.00. La barrera Mega Arm tiene un sistema de baterías, por lo que, si hay un problema de red, se corta la energía y las baterías de 24V pueden seguir funcionando. Se utiliza un relevador "G3MB-202P" de la marca OMRON y tiene un costo de US$4.00, adicionando un transistor 2N222 con resistencia de protección.

En la barrera se colocaron dispositivos LEDs para indicar al usuario el acceso a la PUCP, también se acondiciono un parlante en donde el personal de seguridad puede estar

en comunicación con el usuario. El microcontrolador utilizado para el sistema es el "ATmega328" con un precio de MX$10.63, una fuente de 2.5 A de la marca Thermaltake con un precio de MX$100.00. Y un lector de RFID "MFRC522", entre otros dispositivos.

La ventaja de utilizar un sistema con aplicación en RFID y detección de placas vehiculares es mejorar el sistema para gestionar el acceso del personal, automatizando el acceso y registro de las entradas y salidas. La desventaja de utilizar estos sistemas se halla en el constante mantenimiento adecuado que requiere para todos los sensores y sistemas digitales.

2.3.4 Caso 4. Caseta de cobro por RFID

El informe universitario "Control de acceso mediante FPGA y RFID" presenta el diseño e implementación de un sistema de control de acceso mediante identificación por radiofrecuencia (RFID) controlado por una matriz de computadoras programables (FPGA). El comportamiento de una FPGA es descrito utilizando el lenguaje VHDL, lenguaje de descripción y modelado diseñado para describir la funcionalidad y la organización de sistemas digitales en hardware, placas de circuitos y componentes (Ortiz, Ibarra, Andrade y Almanza, 2012).

El sistema esta aplicado a una caseta de cobro para carretera, donde, los usuarios (coches) contarán con una tarjeta transponder y cuando se acerquen a la caceta no tengan que detenerse por completo para poder hacer el pase de caseta. Cuenta con dos dispositivos de adquisición RFID, uno de baja frecuencia Texas Instrument (TI) modelo RI-STU-MB2A con un costo de MX$2,944.29 y uno de alta frecuencia Alien Technology modelo ALR9800 con un costo de MX$3,951.95. También se utilizó un par de antenas modelo RI-ANT-S01C con un costo de MX$2,335.68, con esta, el dispositivo tiene un alcance de no mayor a 30cm para el transponder pasivo. Para el dispositivo de adquisición ALR9800 se utilizan dos antenas modelo ALR-9610-BC. Otro componente clave de este sistema es la tarjeta de desarrollo Spartan 3E STARTER KIT de Xilinx, valorada en MX$2,999.00, y que se utiliza como unidad central de procesamiento.

Todo el sistema fue desarrollado en LabView, un entorno de programación gráfica, la comunicación propuesta fue en modo serie. El sistema se controla a través de un panel frontal que muestra los datos del usuario.

La desventaja de utilizar un sistema de acceso vehicular son los altos costos al utilizar los materiales propuestos. La ventaja de usar materiales costosos y de calidad es que su durabilidad y confiabilidad son mayores a otros en el mercado. El sistema propuesto y la programación son muy prácticos y funcionales.

2.4 Comparativa

Se realizó un estudio, cotizando sistemas de acceso vehicular por RFID en el mercado, de los cuales a continuación, se darán a conocer los costos y ciertas características que posee cada uno.

Por la empresa (BARTEK ID System, 2023), se contactó un proveedor el cual dio dos opciones de sistemas:

- Opción 1. Lector de presentación casi contacto
  1 unidad de control de acceso: Barrera vehicular (Brazo de 3 mts de aluminio)
  1 lector de tarjeta RFID (Lector alcance 2-3 cm)
  1 panel de control
  100 tarjetas RFID tipo tarjeta de crédito (lectura de 2-3 cm)
  Costo: USD$ 2,235 / MX$ 43,710.05

- Opción 2. Lector largo alcance
  1 unidad de control de acceso: Barrera vehicular (Brazo de 3 mts de aluminio)
  1 lector de tarjeta RFID largo alcance (Lector alcance 4 metros)
  1 panel de control
  100 tag vehicular RFID largo alcance (se adhiere al parabrisas del auto)
  Costo: USD$ 4,285 / MX$ 83,802.04

Por la empresa (JP CRD Accesos y Controles, 2023), se contactó un proveedor el cual dio dos opciones de sistemas:

- Opción 1. SCA Vehicular Barreras de Uso Rudo Largo Alcance
  2 antenas de RFID largo alcance 6 metros
  2 soporte de antenas
  1 controlador
  200 tags de largo alcance (6/12 metros)
  2 barreras Access Pro (brazo 4/6 metros, 2,500 ciclos diarios)

2 sensores de masa para cerrado automático de puertas
2 lazos sensores de presencia vehicular
1 instalación, obra civil, configuración y capacitación en barrera y lectores
Costo: USD$ 5,936 / MX$ 115, 958.93

- Opción 2. SCA Vehicular Barreras de Uso Rudo Proximidad
  2 antenas de proximidad
  2 soporte de antenas
  1 controlador
  200 tags de proximidad
  2 barreras Access Pro (brazo 4/6 metros, 2,500 ciclos diarios)
  2 sensores de masa para cerrado automático de puertas
  2 lazos sensores de presencia vehicular
  1 instalación, obra civil, configuración y capacitación en barrera y lectores
  Costo: USD$ 4,928 / MX$ 96,295.29

Por la empresa (AUTOSISTEMAS, 2023), se contactó un proveedor el cual dio una opción de sistema:

- Opción1. Paquete 3 (Precio sin IVA)
  1 controlador para 2 lectores
  2 lectores largo alcance (6 metros)
  2 barreras vehiculares
  2 brazos con led
  4 controles remotos
  2 radio receptores para controles remotos
  2 detectores de vehículos
  2 llaves de apertura manual
  2 kit de anclaje
  Costo: MXN$ 66,800

Los casos de estudio mencionados anteriormente en la sección 2.3, no satisfacen los requisitos del sistema requeridos para el presente trabajo de titulación, ya que no cuentan con los parámetros de costos, equipo (hardware) y métodos de comunicación (software). En esta sección los casos de estudio para la cotización puede que cumpla con

algunos objetivos o requisitos, pero no satisfacen al 100% lo que se quiere lograr con este trabajo.

De acuerdo con los objetivos mencionados en el capítulo 1, se tiene que obtener un sistema de bajo costo el cual la mayoría de los proveedores no cumple con esta característica. Un sistema que utilice como base de datos una Raspberry Pi, donde, en ningún caso cumple. El sistema debe estar controlado por un sistema de programación en php con un protocolo de comunicación SPI. Por este motivo se plantea en esta tesis un sistema que satisfaga los requisitos, como se plantea a continuación.

3. METODOLOGÍA

3.1 Introducción

El sistema desarrollado intenta cubrir la mayoría de los puntos o tecnologías que podrían jugar un papel en proyectos o implementaciones de este tipo. La idea es sentar las bases de todos los problemas que hay que resolver si se quiere implementar un sistema con tecnología RFID. Aunque en este trabajo se analiza un problema específico como lo es el control de acceso. La solución propuesta también podría resolver otros casos de estudio con determinadas variantes.

3.2 Descripción del sistema

El sistema implementado está constituido por una Raspberry Pi, la cual, es una computadora basada en ARM con pines GPIO, USB, ethernet, entre otras cosas. Misma que cuenta con diversas ventajas pues es una computadora muy económica y tiene casi el mismo tamaño que una tarjeta de crédito.

La placa Raspberry Pi 4 cuenta con un procesador ARMv8 de 64 bits de cuatro núcleos con modelos de 2GB, 4GB u 8GB. Aparte del nuevo procesador y más opciones de RAM. Los puertos de pantalla utilizan puertos micro-HDMI y pueden admitir una resolución de 4K. Además, 2 de los 4 puertos USB de la placa son USB3.0 para admitir transferencias de datos más rápidas (Mallari J. 2020).

Una buena alternativa a un servidor web económico es utilizar Raspberry Pi4, que se puede configurar para que actúe como servidor HTTP para la aplicación web.

3.2.1 Configuración de la Raspberry

Se analizará la instalación de lo que es la pila LAMP (Linux, Apache, MySQL, PHP), PuTTy y FileZilla que son las herramientas y programas que llevan a la Raspberry Pi4 al funcionamiento completo como ordenador y base de dato para el desarrollo de este trabajo de tesis. Posteriormente se analizará la configuración de los puertos de entrada y salida de propósito general y de la comunicación SPI, así como, una librería para el reconocimiento de imágenes.

3.2.1.1 Instalación de la pila LAMP, PuTTy y FileZilla

Para que el sistema de estacionamiento se ejecute continuamente en la Raspberry manteniéndose en la espera de peticiones de ejecución que le hagan los usuarios del estacionamiento, se requiere configurar la Raspberry como servidor WEB. El servidor es responsable de responder adecuadamente a estas solicitudes, dando como resultado que la página o la información se entregue de acuerdo con los comandos solicitados.

Se determina usar una Raspberry porque su costo es mucho menor a la de una computadora de escritorio o una laptop y puede utilizarse como un servidor WEB en el ambiente en el que se diseñó la pluma del estacionamiento. La forma más sencilla de utilizar Raspberry Pi como servidor web es instalar uno de los servidores web tradicionales como Apache o lighttpd.

Se empleó la pila LAMP (combinación de Linux, Apache, MySQL y PHP), para ejecutar aplicaciones que van desde sistemas de gestión de contenidos hasta foros interactivos. Todo esto se puede hacer con una Raspberry Pi, siempre y cuando no esperes un rendimiento similar al de un potente servidor comercial.

Se utiliza el sistema operativo Raspbian para instalar y configurar la pila LAMP, ya que es el sistema operativo por defecto para la Raspberry Pi. Para eso desde el navegador se ingresa a la página oficial de Raspberry (Raspberrypi.org/software/), posteriormente, en la sección de descargas se descarga la imagen del sistema, a continuación, el archivo descargado se ejecuta como administrador y se instala. Una vez instalado el programa se abre, se selecciona el sistema operativo y la tarjeta (microSD) en la que se grabara el sistema de la Raspberry. El usuario y contraseña por default en la Raspberry es “pi” y “raspberry” respectivamente, se pueden cambiar y configurar ambos por seguridad.

Para usar los puertos GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General) se deben asignar permisos para su ejecución remota. Para ello, se debe ingresar a “preferencias”, después en “RaspberryPiConfiguration”, enseguida se abre una ventana en donde se coloca Hostname e Interface (SSH) y se habilita. Asimismo, para utilizar los pines GPIO se habilitan y se reinicia la Raspberry. Para obtener la IP local se puede visualizar en un recuadro arrastrando el cursor en la parte superior derecha del monitor o abriendo la consola y con el comando “ifconfig”.

Posteriormente en la terminal o consola se utilizan los comandos “sudo apt-get update”, “sudo apt-get install apache2” y “sudo apt install php libapache2-mod-php php-mysql” para instalar los paquetes necesarios. Como MySQL ya ha sido remplazado por otra base de datos llamada MariaDB, se ingresa el comando “sudo apt-get install mariadb-server” y se quitan las opciones de MySQL escribiendo el comando “sudo mysql_secure_installation”. Lo siguiente es instalar PhpMyAdmin con el comando “sudo apt install phpmyadmin php-mbstring php-gettext” y “sudo apt install phpmyadmin”. Revisar el apéndice en la sección 1 para mayores detalles de la instalación.

Una vez que se complete la instalación, tanto el servidor MariaDB como el servidor web Apache se ejecutarán en segundo plano. Para verificar que el servidor esté funcionando correctamente, use cualquier computadora conectada a la misma red que la Raspberry Pi e ingrese la dirección IP de la Raspberry Pi en la barra de direcciones de un navegador web. El navegador debería mostrar la página predeterminada.

El último paso es confirmar que el módulo PHP se ha instalado correctamente en Apache. Este módulo es importante porque permite que el servidor web ejecute scripts PHP para entregar contenido dinámico; de lo contrario, el servidor web solo podría entregar páginas estáticas. Para probar el correcto funcionamiento de este módulo, se creó un nuevo archivo de script PHP, utilizando el siguiente comando escrito en una línea: sudo sh -c 'echo "<php phpinfo ();?>"> /var /www /phptest.php’. Este comando crea un nuevo archivo llamado “phptest.php” en el directorio /var/www, que le dice a PHP que cree una página de información con fines de diagnóstico. Para ver su contenido utiliza un navegador en cualquier dispositivo conectado a la red e ingresa la URL http://dir-IP-Raspberry/phptest.php, también puedes probar en la misma RaspberryPi escribiendo http://localhost/phptest.php en la barra de direcciones de su navegador. Para más

información acerca de la instalación de la Pila LAMP consultar el apéndice en la sección 1.

Para ejecutar los comandos en la terminal se emplea un cliente SSH telnet en el que se ingresa la IP de la Raspberry (IP local) y el puerto. El programa que se utiliza para el desarrollo de esta tesis como cliente SSH es PuTTy, mismo que se descarga en: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html, toda la instalación de los comandos a los que se refiera los siguientes apartados se ingresan en la terminal de dicho cliente. Para más información consultar: https://documentation.help/PuTTY/.

Para la transferencia de los archivos entre la computadora en la que se realizó la programación y la Raspberry se usa el programa FileZilla que emplea un protocolo de comunicaciones SSH, que se descarga en: https://filezilla-project.org/download.php. Dicho programa consiste en una interfaz gráfica en la que se tiene que ingresar la dirección IP, el puerto y el tipo de protocolo con el que se establecerá la comunicación. Dentro del programa aparecen dos ventanas, una en la parte izquierda, la cual contiene los archivos de forma local y en la parte derecha son los archivos en la Raspberry. Para más información consultar: https://www.greengeeks.com/blog/the-ultimate-guide-to-using-filezilla/.

3.2.1.2 Configuración de los GPIO y comunicación SPI

Una vez lista la Raspberry Pi, con fuente de alimentación y la memoria con el sistema operativo, se necesita configurar los pines GPIO a utilizar. Existen distintas formas de configurar un solo puerto para poder transmitir datos y se pueden utilizar casi todos los pines como pines de propósito general (entrada/salida).

Antes de configurar los puertos se debe crear la conexión con el modem designado para la red. Para ello, primero en la PC se ingresa a la página de administrador del módem y se selecciona firewall, después al enrutamiento de puertos, en Host interno se introduce la dirección IP de la Raspberry (IP local). Para encontrar la dirección IP local se utiliza PuTTy. Dentro de PuTTy pedirá el usuario y contraseña de la Raspberry. Enseguida se ingresa el comando “ifconfig” el cual da la dirección IP local, ese número se escribe en la página anteriormente buscada del módem en el apartado de Host interno, posteriormente deberá seleccionar el dispositivo que está conectado al módem y deseemos utilizar. También se tiene que ingresar el protocolo utilizado (TCP) y el número

de puerto interno que corresponde al de la Raspberry y a menos que se cambie, por default será el número 22, al igual que el numero externo es el 22.

Volviendo a PuTTy se ingresa el comando “python3”, ahí se escriben las declaraciones para acceder a los puertos de entrada/salida de la Raspberry. En el caso de declarar los GPIO de acuerdo con su número de pin se ingresa el comando “import RPi.GPIO as GPIO”, “GPIO.setmode(GPIO.BOARD)”, “GPIO.setup(n, GPIO.OUT)” refiriéndose a “n” como el número del GPIO designado. En este caso se designó el pin “n” como salida, para designarlo como entrada cambiaria el ultimo comando a: “GPIO.setup(n, GPIO.IN)”.

3.2.1.3 Instalación de OpenCV

La instalación de OpenCV es un poco compleja, se recomienda acceder a la página: https://programarfacil.com/blog/vision-artificial/opencv-raspberry-pi/ , la cual es la que se utilizó como tutorial para la instalación.

A continuación, se mostrarán comandos utilizados para la instalación de OpenCV, para ver el proceso completo los scripts se encontrarán en la sección 2 del apéndice. Se sugiere revisar el apéndice ya que existen modificaciones dentro de los scrips que no se mostraran a continuación.

Comandos utilizados para instalación de OpenCV y otras librerías:

- **pip –version**
- **sudo cat /etc/os-release**
- **sudo pip3 install virtualenv virtualenvwrapper**
- **sudo nano ~/.bashrc**
  Dentro de este último comando se debe agregar al final del texto lo siguiente:

  ```
  export WORKON_HOME=$HOME/.virtualenvs
  export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
  source /usr/local/bin/virtualenvwrapper.sh
  ```

- **source ~/.bashrc.**
- **mkvirtualenv opencv4-5-tesis**
- **sudo pip3 install opencv-contrib-python==4.5.4.60**
- **sudo apt install libgfortran5 libatlas3-base**
- **sudo pip3 install numpy==1.22.3**
- **sudo pip3 install pytesseract==0.3.9**

3.2.1.4 Otras configuraciones

Configuración para la detección de tarjetas RFID en RaspberryPi

La Raspberry Pi necesita configurarse para que pueda detectar los comandos y librerías necesarias al ejecutar el programa. A continuación, se mostrarán algunos comandos para la configuración. Para más información consultar el apéndice en la sección 3, ya que se presentan más comandos e imágenes.

- **`sudo nano /boot/config.txt`**
- **`device_tree_peram = spi = on`**
- **`dtoverlay = spi-bcm2708`**
- **`sudo raspi-coonfig`**
- **`dmesg | grep spi`**

Habilitar Cámara

Para tomar fotos con la cámara seleccionada, en este caso se trata de la cámara Raspberry NoIR, se ocupa habilitarla en el sistema de la RaspberryPi. Para esto es necesario seguir los pasos indicados en la sección 4 del apéndice.

Base de datos

Para la base de datos se ocupan los programas previamente instalados que son MariaDB y phpMyAdmin que son los responsables para crear las tablas y la interfaz. Para esto se tiene que seguir una serie de pasos que se encuentra en el apéndice sección 5. Ahí se mostrarán todos los comandos y señales para poder crear su base de datos.

3.2.2 Enfoque de estudio sobre etiquetas RFID

La tecnología RFID ha estado en auge en los últimos años gracias a la relativa reducción de costes en el mercado, el aumento de sus capacidades y los beneficios que muestra frente a otras tecnologías de autoidentificación.

Con base en lo anterior, la tecnología RFID promete revolucionar la vida de las personas debido a sus diferentes aplicaciones. Al involucrarse en la cadena de producción y distribución de las fábricas, generará enormes beneficios como: descripción de la línea de producción, verificación de la calidad de los productos, preparación de inventos automáticos, desde el momento que ingresan al almacén de la tienda; comprensión del momento de suministro y pago automático del carrito al pasarlo a caja.

Los beneficios también se integran en las actividades diarias. El desarrollo de frigoríficos con tecnología RFID no sólo detectará el momento en el que un producto está a punto de caducar, además, si se solicita un suministro, se informará al cliente de esta condición. Las lavadoras, por ejemplo, identificarán el período de lavado que corresponde a un tipo de ropa definido.

Al final, otra aplicación es en las plantas de reciclaje de basura porque las máquinas emitirán informes sobre el material en el que están fabricados determinados productos, facilitando así su división y agrupación. En definitiva, las aplicaciones de esta tecnología son prometedoras, sin embargo, aún se encuentra en proceso de evolución.

3.2.3 Protocolo de comunicaciones SPI

Para que los circuitos entre el módulo con el que se comunica la tarjeta (Raspberry) se comuniquen, se necesita de un protocolo de comunicaciones, el que se utiliza en la mayoría de las tarjetas comerciales es el SPI, el cual se describe a continuación:

SPI pertenece a los protocolos más reconocidos para trabajar con comunicación serial gracias a su velocidad de transmisión, simplicidad, manejo y además porque varios dispositivos en el mercado como pantallas LCD (Liquit Crystal Display), sensores, microcontroladores tienen la posibilidad de trabajar con él.

SPI es un protocolo síncrono que funciona en modo full duplex para recibir y transmitir información, permitiendo que 2 dispositivos se comuniquen entre sí simultáneamente usando diferentes canales o diferentes líneas en el mismo cable. Al ser

un protocolo síncrono, el sistema dispone de una línea adicional a la línea de datos delegada para realizar el proceso de sincronización. De la cual se muestra su funcionamiento en el inciso a) de la figura 3.1.

Hasta la Figura 3.4 se muestra el funcionamiento de la conexión con la Raspberry Pi y un solo lector RFID.

Para ejemplificarlo mejor, en este protocolo se define un maestro que será aquel dispositivo delegado para transmitir información a sus esclavos, en esta situación será la Raspberry Pi. Los esclavos van a ser aquellos dispositivos que se encargan de recibir y enviar información al maestro (lector de tarjetas RFID). Cabe señalar que el maestro también puede recibir información de sus esclavos. Para que este proceso se haga realidad es imprescindible disponer de 2 registros de movimiento, uno para el maestro y otro para el esclavo respectivamente. Los registros de movimiento están delegados para almacenar bits en paralelo para realizar una conversión en paralelo a serie para la transmisión de información.

Existen cuatro líneas lógicas encargadas de realizar todo el proceso:

- **MOSI (Master Out Slave In):** Línea utilizada para llevar los bits que provienen del maestro hacia el esclavo.
- **MISO (Master In Slave Out):** Línea utilizada para llevar los bits que provienen del esclavo hacia el maestro.
- **CLK (Clock):** Línea proveniente del maestro encarga de enviar la señal de reloj para sincronizar los dispositivos.
- **SS (Slave Select):** Línea encargada de seleccionar y a su vez, habilitar un esclavo.

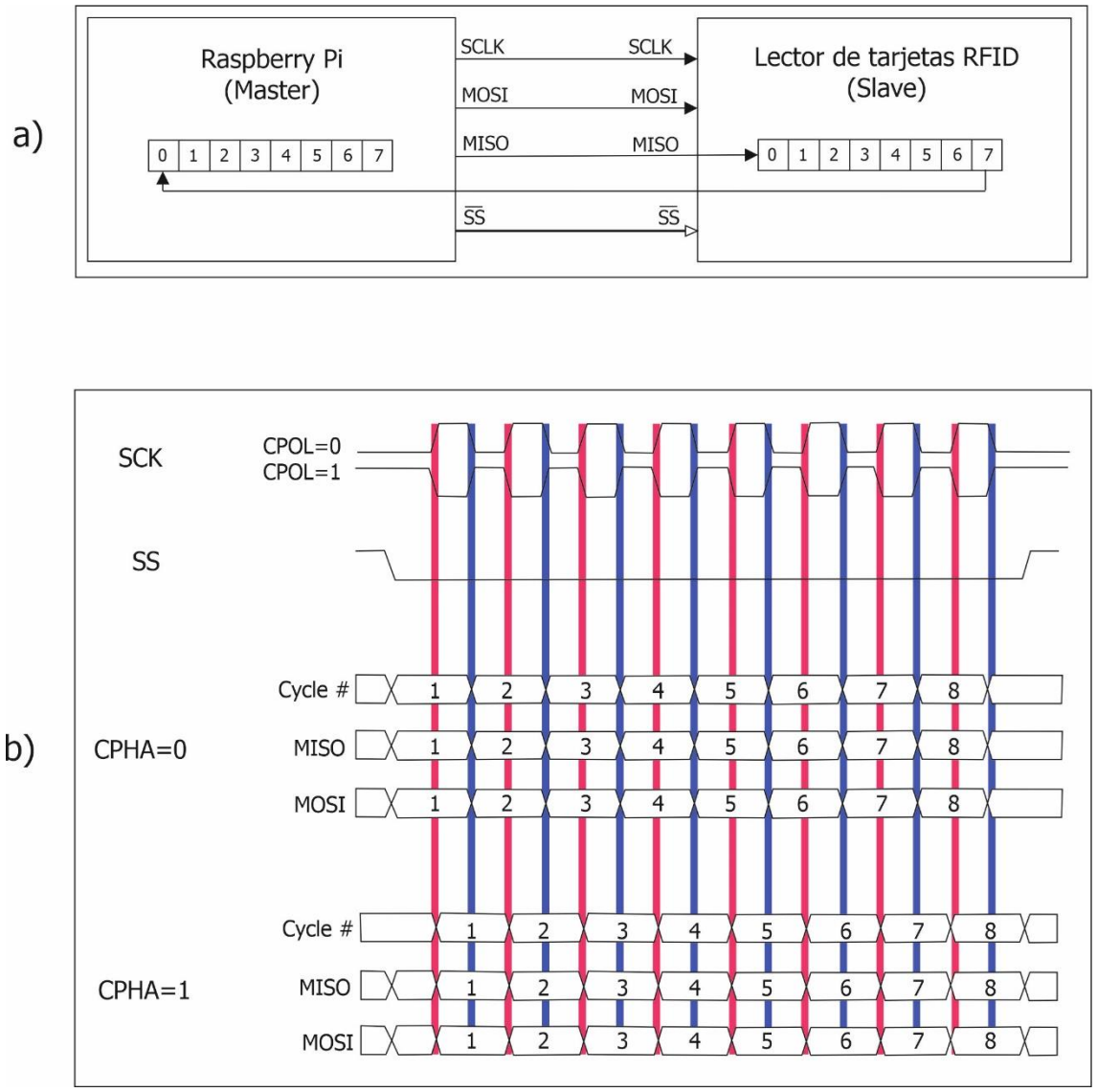

Figura 3.1 *Estructura general del protocolo SPI. En a) se muestra la conexión Master/Slave donde se tienen todas estas líneas con sus respectivos registros de desplazamiento y su dirección de flujo, mientas que, en b) distintos modos en SPI dependiendo del estado del SCK. (Elaboración propia)*

Existen 4 métodos en los que se puede enviar información dependiendo de 2 límites basados en la señal del reloj, de la misma forma que se muestra en la Figura 3.1. b). El primero de ellos es la polaridad y el segundo es el escenario. Al tener 2 límites donde cada uno puede tomar 2 estados, tendremos entonces 4 métodos diferentes de poder realizar el proceso de transmisión y envío de información.

- Modo 0: CPOL = 0 y CPHA = 0. Modo en el cual el estado del reloj permanece en estado lógico bajo y la información se envía en cada transición de bajo a alto, es decir alto activo.

- Modo 1: CPOL = 0 y CPHA = 1. Modo en el cual el estado del reloj permanece en estado lógico bajo y la información se envía en cada transición de alto a bajo, es decir bajo activo.

- Modo 2: CPOL = 1 y CPHA = 0. Modo en el cual el estado del reloj permanece en estado lógico alto y la información se envía en cada transición de bajo a alto, es decir alto activo.

- Modo 3: CPOL = 1 y CPHA = 1. Modo en el cual el estado del reloj permanece en estado lógico alto y la información se envía en cada transición de alto a bajo, es decir bajo activo.

La configuración de métodos es libre para cada esclavo, esto significa que cada esclavo puede tener una configuración de CPOL (Clock Polarity) y CPHA (Phase Polarity) diferente a otros esclavos, incluso una frecuencia de trabajo diferente y luego Para esto, el maestro tendrá para ajustarse a la configuración de cada esclavo. Por este motivo, es aconsejable que el sistema intente funcionar dentro de los mismos límites si es viable. Todo esto en el caso de tener a más de un esclavo (Digi-Key Electronics, 2019).

En esta tesis la comunicación entre la Raspberry Pi y las tarjetas lectoras de RFID se realizó con un maestro (Raspberry) y dos esclavos (Tarjeta RFID). La conexión que se utilizó fue directa, tal como se presenta en la Figura 3.1. a). Uno de los beneficios que brinda este protocolo es la velocidad de transmisión debido a que es configurable a través de un programa y también dependerá de los dispositivos utilizados en el sistema. SPI tiene velocidades de transmisión mucho más altas que otros protocolos de comunicación ya que funciona en modo full duplex. Los límites configurables son la frecuencia del reloj, la configuración de etapa (CPHA) y la polaridad (CPOL). Si sólo hay un esclavo, la línea

SS fija se puede posicionar si el esclavo lo permite. No se limitan a trabajar con palabras de 8 bits. Es muy utilizado cuando es necesario comunicarse con grupos en distancias cortas.

3.2.4 Propuesta del reconocimiento óptico de placas vehiculares con la Raspberry

El reconocimiento de las placas vehiculares para permitir o no el acceso se realiza a través de cámaras de vigilancia dadas las condiciones del estacionamiento de la Facultad de Ingeniería, pues dicho estacionamiento está a la intemperie y se requiere que la cámara tenga capacidad para captar las imágenes de las placas en diferentes condiciones de iluminación.

Las cámaras especialmente diseñadas para las Raspberry Pi se caracterizan por un cable plano de 15 contactos, el cual la Rasberry Pi tiene módulos de cámara que se conectan por el puerto llamado "CAMERA" para poder realizar la conexión con la cámara y su cable plano. A continuación, en la Tabla 1 se plasman las características de 3 modelos de cámaras con sus características.

Tabla.1 *Características de las cámaras nativas para Raspberry Pi*

Modelo	Visión nocturna	Resolución de video / imagen	Compatible con RaspberryPi 4
Pi NoIR	Sí	1080p30, 720p60 y VGA90 / 8MP	Sí
Pi	No	1080p30, 720p60 y VGA90 / 8MP	Sí
Rev 1.3	No	1080p HD a 30fps / 5MP	No, compatible a RB Pi 3

La discusión de esta sección se basa en la hoja de especificaciones de los médelos de cámaras presentados (Pi NoIR, Pi, rev 1.3). De acuerdo con las Tabla 2 la cámara que satisface las necesidades de este proyecto es la "Cámara Rasberry Pi NoIR" ya que cuenta con visión nocturna y compatible con la Rasberry Pi 4 que es la placa utilizada. El software que se utiliza para el reconocimiento de imágenes, en particular, placas vehiculares es Open CV.

Procesamiento de las imágenes empleando Open CV

Las imágenes (JPG) capturadas por la cámara Rasberry Pi NoIR, las cuales son procesadas por el programa Open CV dándonos una cadena con las placas que se configuran por distintos factores que dependen de las características de las imágenes (día, noche, números y letras similares, calidad de imagen, etc.).

3.3 Propuestas de diseño

A continuación, en la Figura 3.2 se presenta una imagen de las locaciones en donde se encontrarán los accesos de las plumillas en el estacionamiento de la Facultad de Ingeniería.

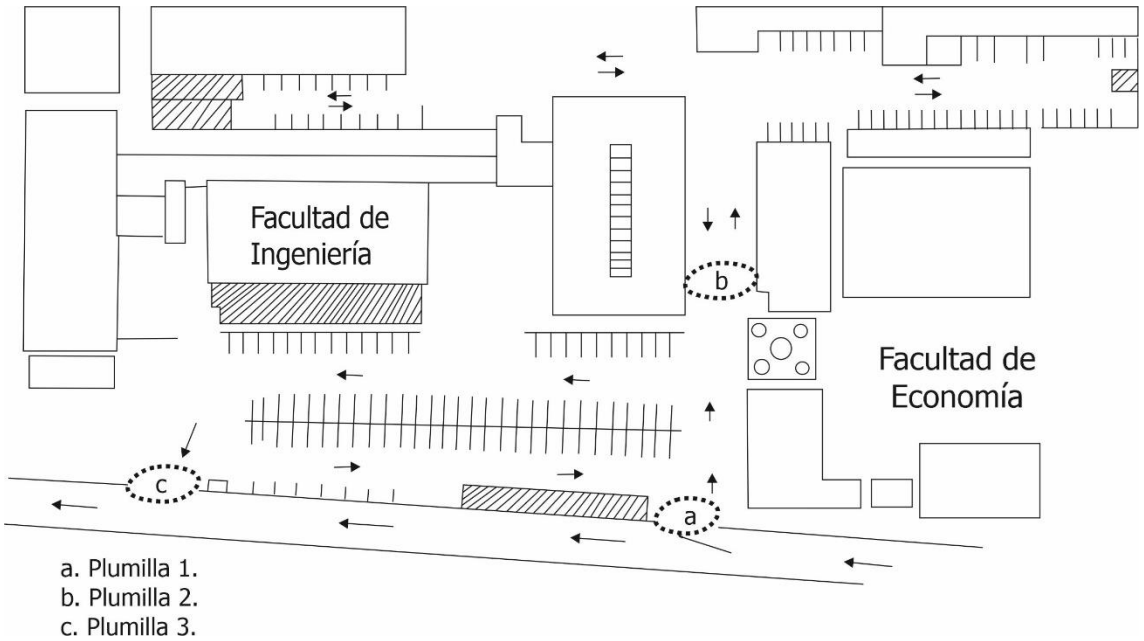

Figura 3.2 *Trazo aéreo de la Facultad de Ingeniería, marcando el sentido vehicular y locación de las plumillas, así como los espacios académicos. (Elaboración propia)*

Como se muestra en la Figura 3.2. existen 3 accesos en los cuales cada uno cuenta con identificadores RFID y una videocámara. La plumilla 1 tiene la función de entrada al estacionamiento de alumnos, la plumilla 3 tiene la función de salida al estacionamiento de alumnos. Mientras la plumilla 2 tiene la función de entrada/salida al estacionamiento de profesores dentro del estacionamiento de alumnos.

El diseño de la plumilla 2 tiene un grado mayor de complejidad ya que con un solo ordenador y una sola videocámara debe realizar la función de entrada/salida. Es así como cuestiones económicas y satisfaciendo los objetivos de esta tesis de posgrado solo se necesita la realización de una de las tres plumillas indicadas en la Figura 3.2. Se tomará como objeto de estudio la plumilla 2 para la realización del trabajo presente.

1. Lector RFID 1.
2. Plumilla y videocámara.
3. Lector RFID 2.

Figura 3.3 *Diseño de la locación para la plumilla 2 que dará acceso en ambos sentidos al estacionamiento de Profesores. (Elaboración propia)*

Para acceder al estacionamiento de profesores se cuenta con un lector de RFID y una videocámara, tal como se muestra en la Figura 3.3., al igual que para la salida. El detalle de este diseño es que se ocupan dos lectores RFID, uno para entrada y otro para la salida. Para el sistema de detección de placas se utiliza una sola videocámara la cual apunta hacia la entrada y por medio de un espejo reflector la videocámara captará los vehículos que deseen salir. Permitiendo que el sistema opere adecuadamente.

En la figura 3.4 se observa un diagrama a bloques que en general representa el funcionamiento y dirección de como se mandan los datos, al igual que un circuito representando las conexiones generales del sistema físico real (Relevadores)

En la Figura 3.5 se representa mediante un diagrama de circuitos las conexiones que se utilizaron para el proyecto utilizando los pines de la Raspberry Pi y todos los adicionales, como los botones para simular las placas metálicas y los servomotores que representan las plumillas

En la Figura 3.6 se muestran las dimensiones y diseño de la caja protectora donde se colocará la Raspberry Pi y todas las conexiones.

Al igual que en la Figura 3.7, Figura 3.8 y Figura 3.9 son los diseños para la construcción de la en tamaño escala.

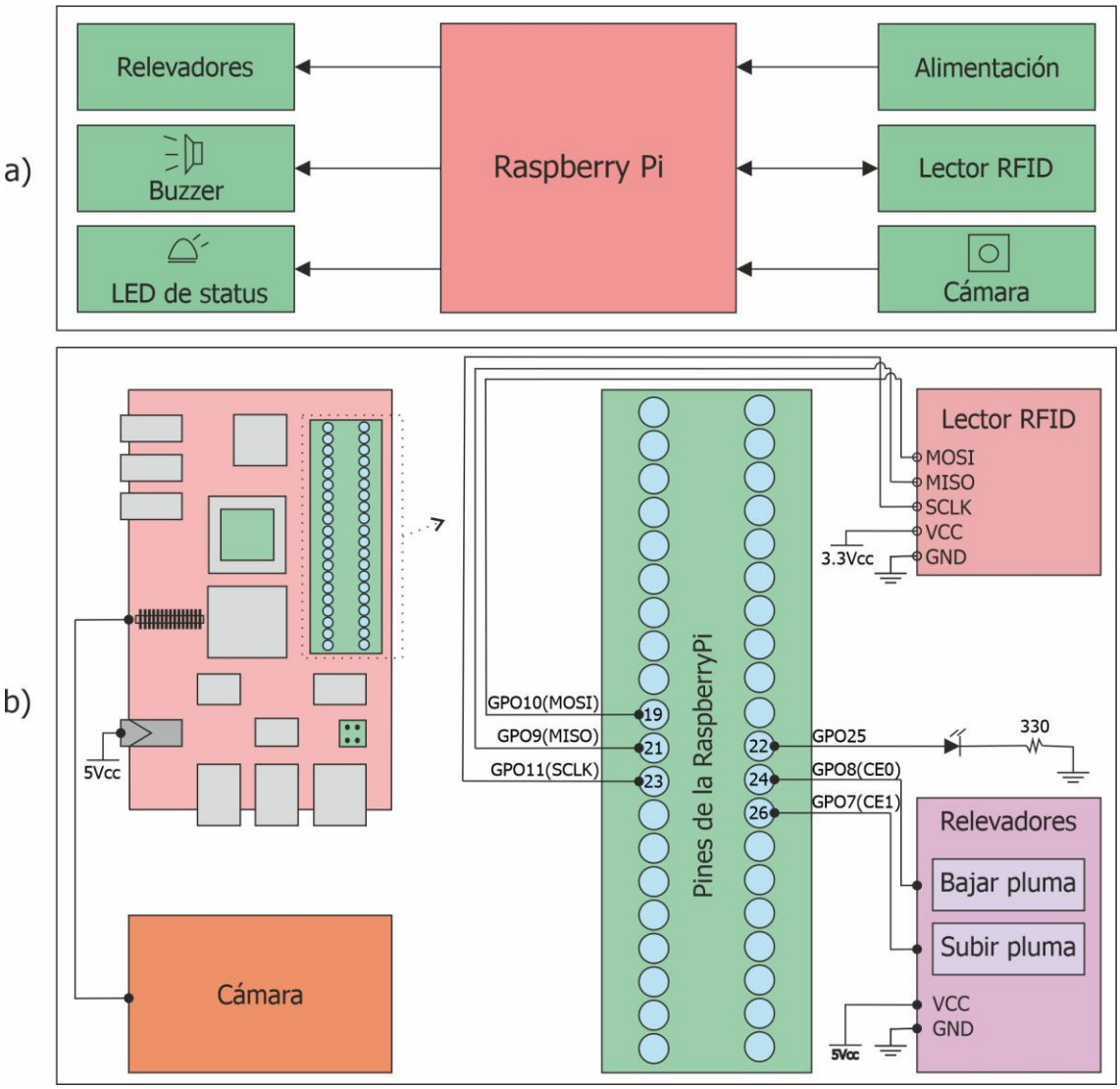

Figura 3.4 *Diagrama a bloques y circuito eléctrico de las conexiones a la Raspberry Pi. En a) se muestra un diagrama a bloques con el sentido en que los datos se envían según los elementos conectados a la Raspberry Pi. En b) se muestra el circuito eléctrico de cómo deben ser conectados los elementos a la Raspberry Pi con sus voltajes respectivos y las conexiones a los GPO, esto es para el caso donde se tenga la pluma real. (Elaboración propia)*

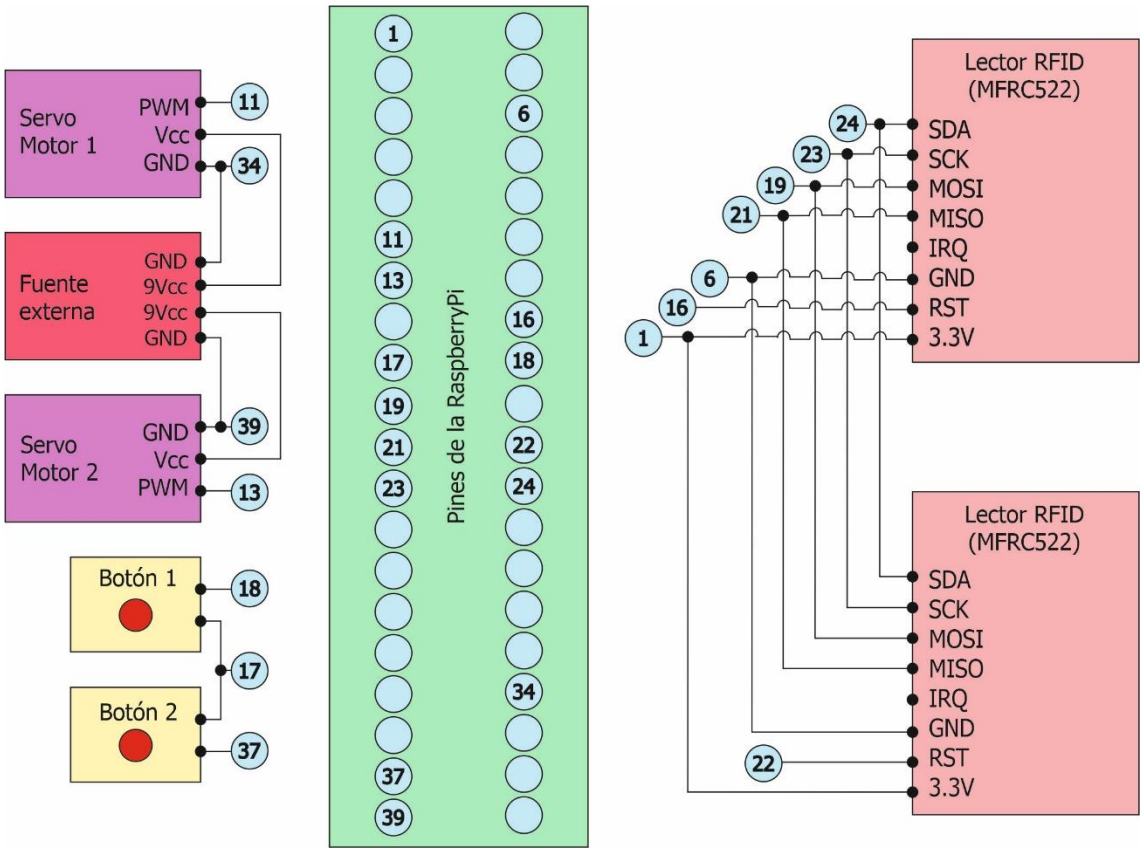

Figura 3.5 *Circuito eléctrico de las conexiones a la Raspberry Pi (pines). Diagrama que se utiliza para la conexión de los dos lectores RFID y la Raspberry Pi, junto a dos servomotores y dos botones que se utilizaron para hacer la representación en maqueta de cómo es el funcionamiento. Recordando que este trabajo de investigación es solo el prototipo. (Elaboración propia)*

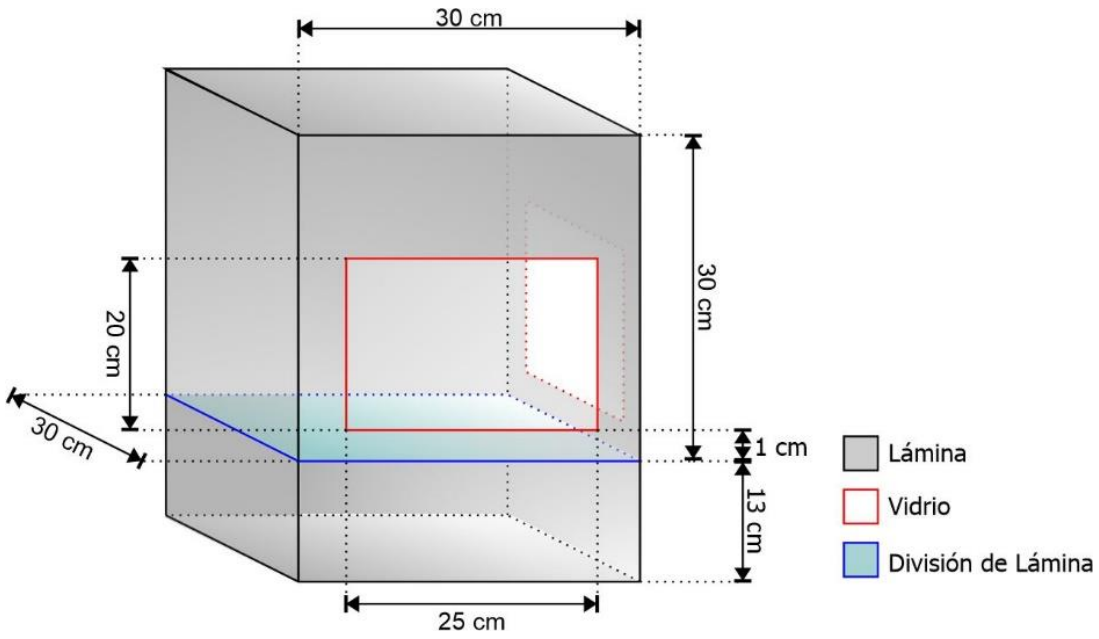

Figura 3.6 *Diseño de la caja protectora para el circuito eléctrico. Para que la Raspberry, cámara y el circuito eléctrico estén protegidos tanto en el ámbito de la seguridad (robo, daño, extravíos, manipulación, etc.), así como el daño y deterioro que pudiera llegar a sufrir por circunstancias o factores externos producidos por la naturaleza. Se muestra la propuesta de diseño la cual se implementó para hacer las tomas correspondientes a la detección de placas, representando el material correspondiente y las medidas reales. (Elaboración propia)*

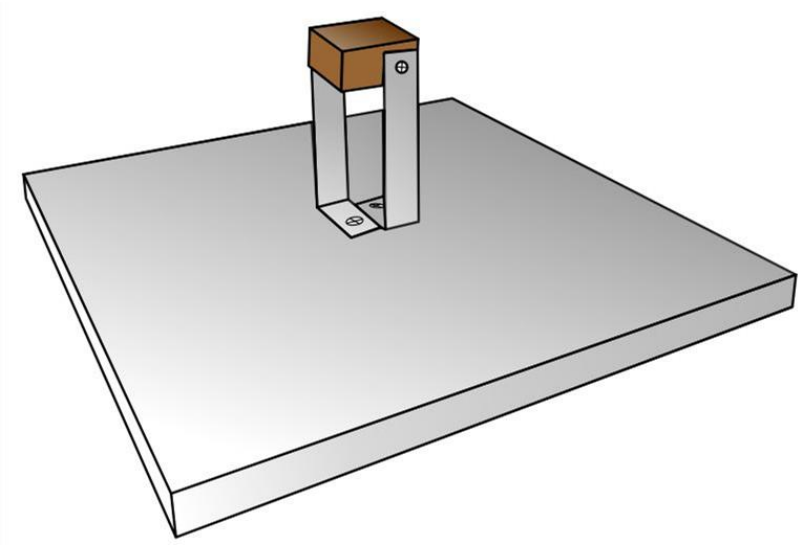

Figura 3.7 *Diseño de la base para la estructura del servo y la cámara. La base sostendrá la estructura del servo y la cámara. Esta se localizará dentro de la caja protectora que es la Figura 3.6. Tiene una elevación de 15cm, necesarios para que la cámara pueda tener una visión a través de los cristales de la caja protectora. Esta pieza sostendrá la base que sostendrá el servomotor. (Elaboración propia)*

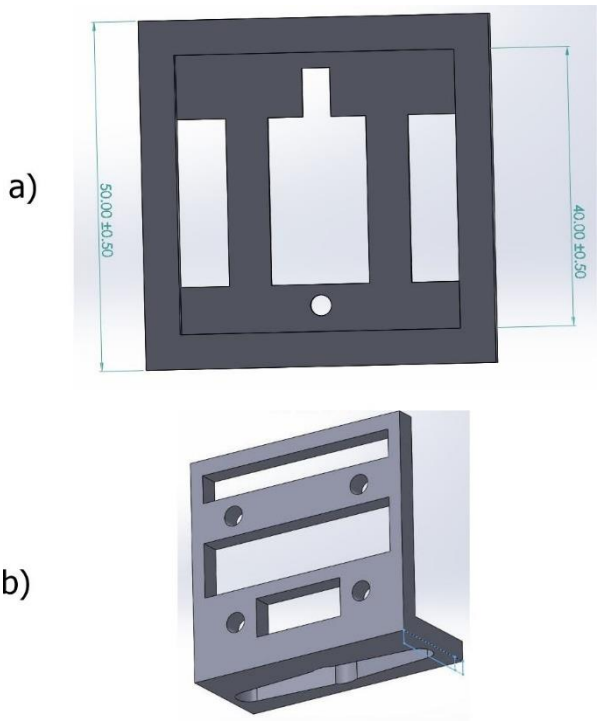

Figura 3.8 *Diseño de la base de la cámara para la Raspberry Pi. En a) se muestra el diseño de la base para el servomotor, que embonara en la parte superior del diseño de la Figura 3.7. Esta pieza está diseñada a las medidas exactas del servomotor y creada por una impresora 3D, al igual que la pieza del inciso b) que es la pieza diseñada para embonar en la rondana del servomotor y uniendo la cámara de Raspberry utilizada en este proyecto. (Elaboración propia)*

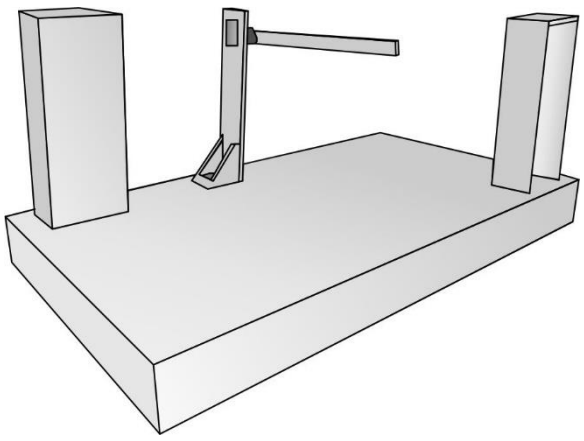

Figura 3.9 *Diseño de la estructura de la maqueta, en la que se representa las funciones principales de este proyecto de tesis. La detección de los RFID que son los dos rectángulos verticales y la simulación de una plumilla con un servomotor. (Elaboración propia)*

4. RESULTADOS

4.1 Configuración del sistema para la Raspberry Pi

La configuración principal del sistema y el inicio de este trabajo de titulación es la instalación de la pila LAMP el cual tiene como resultado las páginas oficiales de cada paquete en la dirección designada de la Raspberry Pi, tal y como se muestra en la Figura 4.1, Figura 4.2 y Figura 4.3, para ello se ejecutan de la siguiente manera.

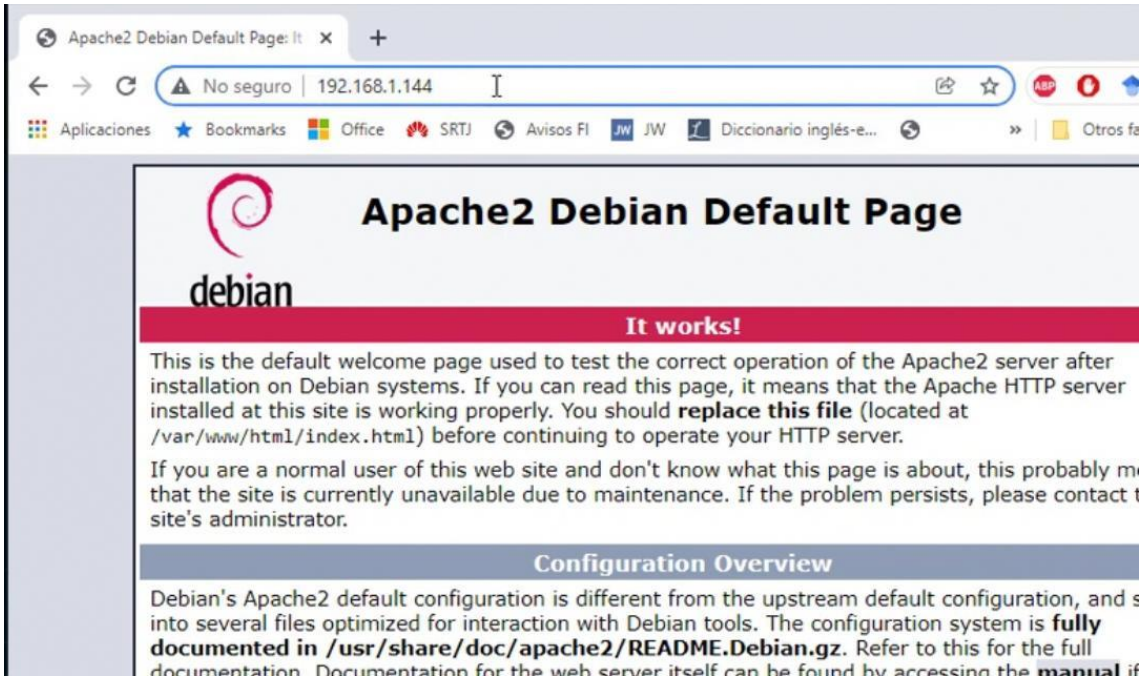

Figura 4.1 *Página de Apache2. Para dirigirse a la página de Apache 2 se ingresa la dirección IP de la RaspberryPi.*

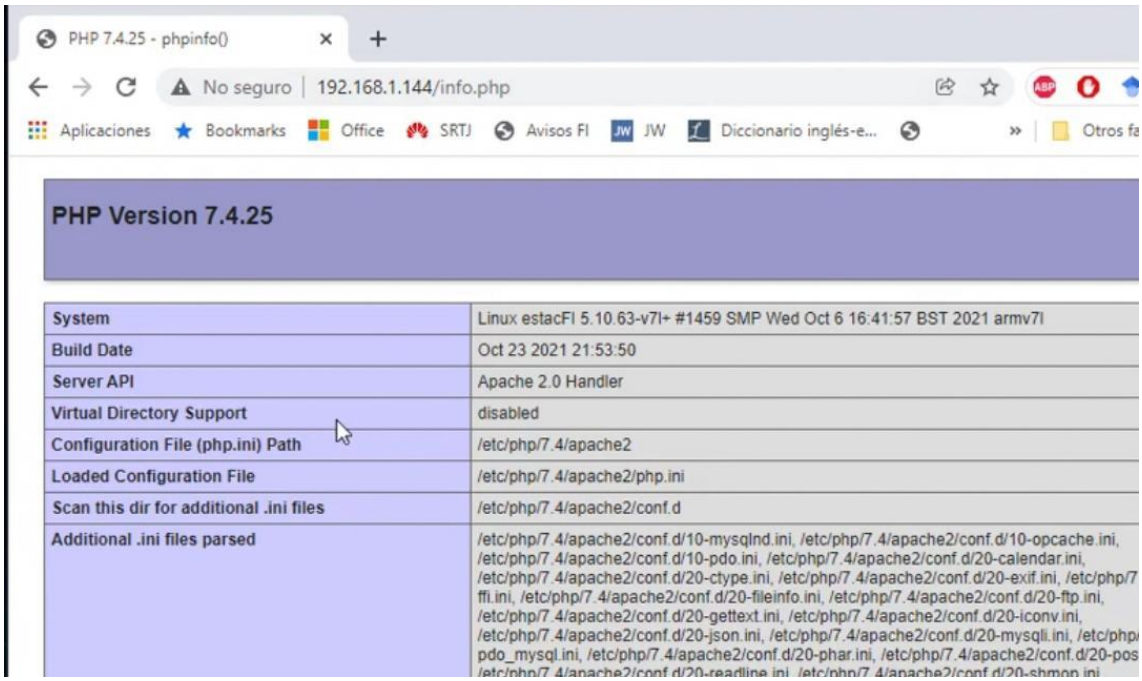

Figura 4.2 *Página de PHP. Para verificar que php se instaló correctamente se ingresa el comando* ***sudo chmod -R 777 /var/www/html****, se crea un archivo con* ***nano /var/www/html/info.php*** *, en donde se abrirá una nueva pestaña y se ingresa el comando* ***<?php phpinfo(); ?>*** *y en el navegador se ingresa la dirección IP/info.php*

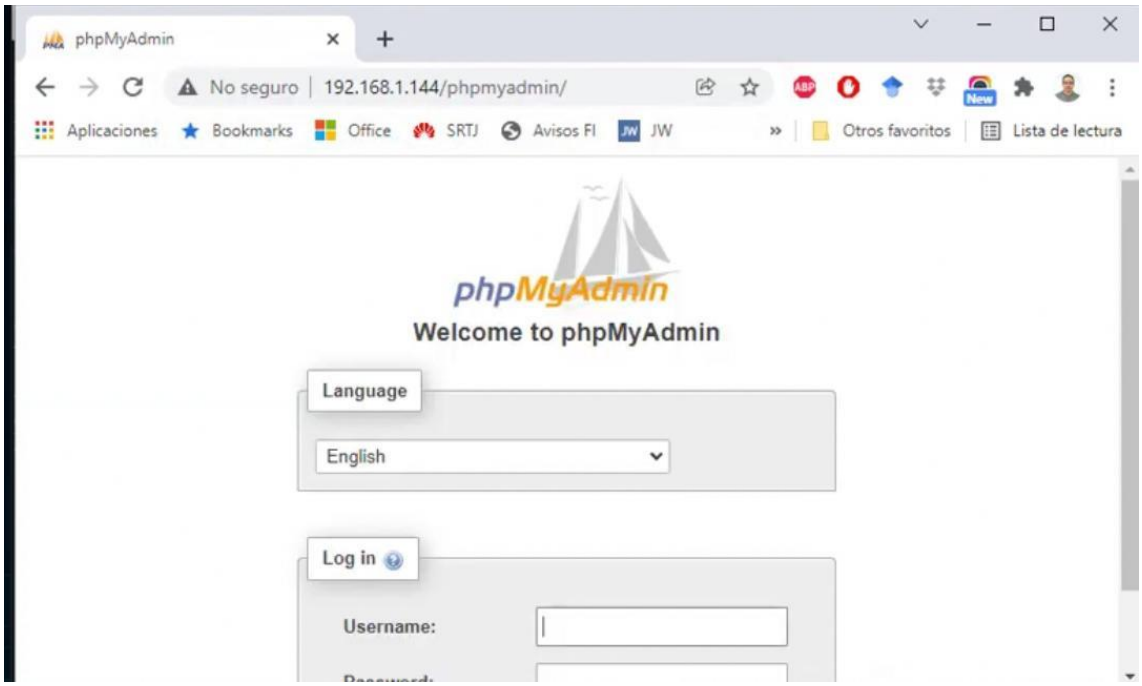

Figura 4.3 *Pagina PHPMyAdmin. Para probar la instalación escribir en el navegador la dirección IP/phpmyadmin, al ver la página por defaul significa que se instaló correctamente*

Otro de los programas instalados dentro de la Raspberry Pi que deben comprobar su instalación es OpenCV, la cual viene con otras librerías que ayudan a la detección de imágenes. Para verificar su instalación se ingresan los siguientes comandos dentro de Python en la consola, como se muestra en la siguiente Figura 4.4.

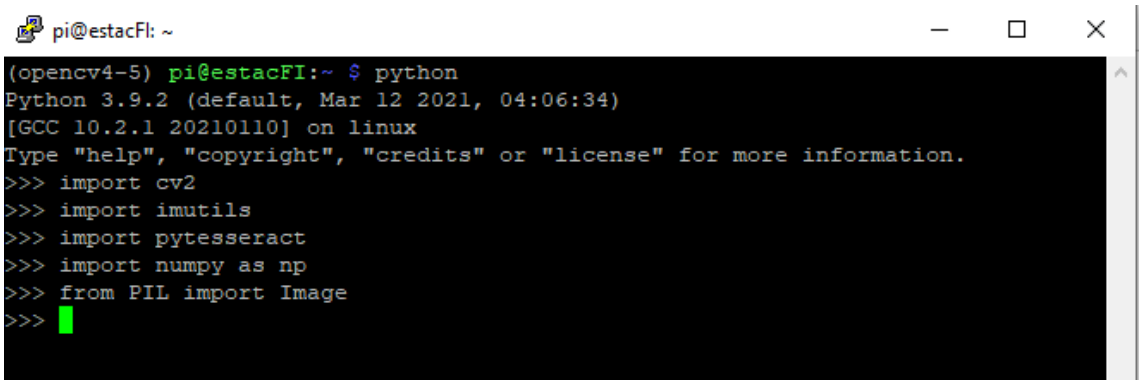

Figura 4.4 *OpenCV. Para probar la instalación de OpenCV se ingresa a Python desde la consola, al ingresar los comandos y no presentar ningún error se verifica que se instaló correctamente.*

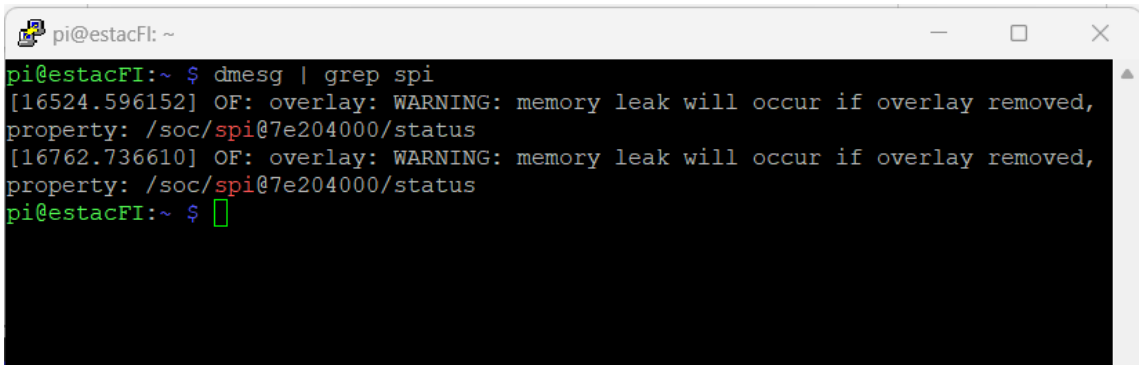

Figura 4.5 *Configurar Raspberry con protocolo SPI. Para comprobar que existe la comunicación con ese puerto se escribe el siguiente comando* ***`dmesg | grep spi`*** *y debe aparecer la siguiente lectura como en la Figura.*

Figura 4.6 *Conexiones de la Raspberry Pi. Se muestran las distintas conexiones que se emplearon en la RaspberryPi, desde las comunicaciones SPI con las tarjetas RFID, los servomotores hasta los botones de activación.*

4.2 Composición y armado del sistema físico

Las imágenes que se mostraran a continuación son el resultado de todas las propuestas de diseño que aparecen en el apartado 3.3.

Figura 4.7 *Caja protectora con cámara y RaspberryPi. Se muestra la caja protectora con los dos cristales que permiten que la cámara pueda tener la visión hacia ambos sentidos, junto con la RaspberryPi y una carcasa. Todo esto sostenido por la base diseñada. (Elaboración propia)*

Figura 4.8 *Parte inferior de la caja protectora. Debajo de la caja protectora como se muestra en la Figura 4.2. se encuentra la fuente de poder que alimenta a los servos motores, al igual se encuentra un orificio por el cual todos los cables ocupados conectan con la parte superior hacia la RaspberryPi. (Elaboración propia)*

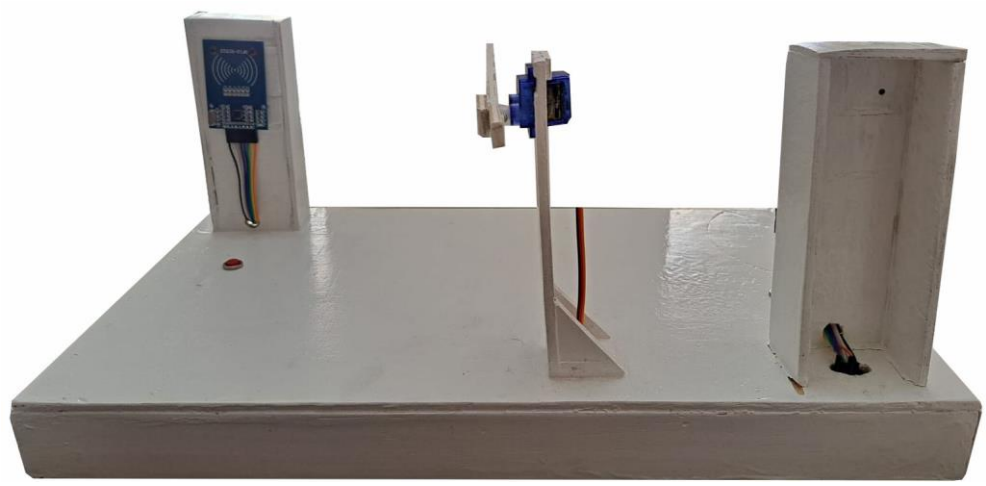

Figura 4.9 *Maqueta representativa. Se muestra la maqueta que actúa como la representación física de la plumilla y los detectores RFID. Para llevar a cabo esta maqueta se utilizaron dos lectores RFID y en la parte que representa el concreto o suelo se insertaron los dos botones que al ser presionados hacen la función que un auto se encuentra ahí. Todos los cables salientes de la maqueta conectan con la RaspberryPi. (Elaboración propia)*

4.3 Programación y algoritmos del sistema

Son 4 los programas que se utilizan para ejercitar el funcionamiento del proyecto, que llevan por nombre: “MFRC522.py”, “RFID.py”, “addusuario.py” y “Main.py”. El programa con nombre “MFRC522.py” se puede encontrar en páginas de internet, ya que es el programa predeterminado al usar una tarjeta de este tipo.

Un ejemplo de cómo obtener este programa es ingresando en el enlace: https://github.com/pat-odoo/TwoRC522_RPi2-3/blob/master/module/MFRC522.py. Desde ahí se puede copiar y pegar el código a un documento nuevo dentro de nuestra consola para así tener el código dentro de la RaspberryPi. Este programa se encarga de crear las conexiones de entrada/salida de las tarjetas RFID y la RaspberryPi a los puertos que se designan por medio de protocolos de comunicación.

El programa con nombre “RFID.py” es un programa que se diseñó para hacer la selección de dos tarjetas RFID. Este programa manda a llamar al programa “MFRC522.py” para que así pueda hacer la conexión con las tarjetas y obtener los datos. A continuación, se dará una breve explicación y el código completo se presenta en la sección 6 del apéndice.

```
1  #RFID.py
2  import RPi.GPIO as GPIO
3  import MFRC522
4  import time
```

Figura 4.10 *En esta sección del código “RFID.py” se importan librerías para activar los GPIO y una variable en cuestión del tiempo, así como el programa antes mencionado “MFRC522.py”.*

```
31      while leyendo and conTiempo:
32
33        tiempoActual = time.time()
34
35        if ( abs(tiempoActual - tiempoInicial)>=30.0):
36
37          conTiempo = False
38          print("Se ha acabado el tiempo")
39          GPIO.output(rst,True)
```

Figura 4.11 *En esta sección del código “RFID.py”, crea una diferencia de tiempo que al ejecutar el programa y no pasar la tarjeta RFID cancela el programa con el mensaje “Se ha acabado el tiempo”.*

```
41        else:
42
43          (status, TagType) = self.__reader.MFRC522_Request(self.__reader.PICC_REQIDL)
44
45          if status == self.__reader.MI_OK:
46
47            (status, uid) = self.__reader.MFRC522_SelectTagSN()
48
49            if status == self.__reader.MI_OK:
50              self.uid = uid
51              print("UID de la tarjeta: %s" % self.uidToString(uid))
52
53              leyendo = False
54              GPIO.output(rst,True)
```

Figura 4.12 *En esta sección del código "RFID.py" al pasar la tarjeta en el rango de tiempo determinado reconoce el código y lo ingresa en la variable "self.uidToString(uid)", que luego es importado a los programas que ejecutan el acceso y registro de los autos.*

El programa con nombre "addusuario.py" es un programa que se diseñó para agregar usuarios a la base de datos. Este programa manda a llamar al programa "RFID.py" para que así pueda hacer la conexión con la variable que contiene el número o clave de cada tarjeta RFID. A continuación, se dará una breve explicación y el código completo se presenta en la sección 6 del apéndice.

```
1 #addusuario.py
2 import RPi.GPIO as GPIO
3 from datetime import datetime
4 import mysql.connector
5 import subprocess
6 import time
7 from RFID import RFID
```

Figura 4.13 *En esta sección del código "addusuario.py" se importan todas las librerías necesarias para ejecutar el programa de acuerdo con las necesidades requeridas, de igual modo se importa el programa "RFID.py" y se activan los puertos GPIO.*

```
9  GPIO.setmode(GPIO.BOARD)
10 GPIO.setwarnings(False)
11
12 buttonEntry = 37
13 GPIO.setup(37, GPIO.IN, GPIO.PUD_DOWN)
14 buttonExit = 18
15 GPIO.setup(18, GPIO.IN, GPIO.PUD_DOWN)
16
17 pino = 16
18 GPIO.setup(16,GPIO.OUT)
19 pint = 22
20 GPIO.setup(22,GPIO.OUT)
```

Figura 4.14 *En esta sección del código "addusuario.py" las líneas 9 y 10 indica la forma de escritura en como se usan los puertos GPIO, las líneas 12 a 15 se define la entrada de los botones que sirven como sensores al momento de presentarse un coche en la entrada o salida del estacionamiento, las líneas 12 a 20 dice que los pines 16 y 22 de la RaspberryPi se declaran salidas.*

```
22 def leer(pino):
23     lector = RFID()
24     lector.read(pino)
25     return lector.uidToString(lector.uid)
26
27 def leeer(pint):
28     lector = RFID()
29     lector.read(pint)
30     return lector.uidToString(lector.uid)
31
32 mydb = mysql.connector.connect(
33     host='localhost',
34     user='root',
35     password='holamundo',
36     database='Estacionamiento'
37 )
38
39 mycursor = mydb.cursor()
```

Figura 4.15 *En esta sección del código "addusuario.py" las líneas 22 a 30 indica que en la variable "pino" y "pint" se guardara el código de la tarjeta RFID, de la línea 32 a 39 permite hacer la conexión con la base de datos colocando los datos de la cuenta vinculada en PhpMyAdmin.*

```
button1 = GPIO.input(buttonEntry)
button2 = GPIO.input(buttonExit)

if button1 == True:
    print("Nuevo usuario para el estacionamiento")
    nombre = input("Nombre:")
    nivel = input("Nivel de acceso. 1.-maestro, 2.-alumno, 3.-visitante):")
    clave = input("Numero de Cuenta:")
    tarjeta = leer(pino)
    nivel = int(nivel)
    clave = int(clave)
    sql = "INSERT INTO usuarios (Nombre, Tarjeta, Nivel, Clave) VALUES (%s, %s, %s, %s)"
    val = (nombre, tarjeta, nivel, clave)
    mycursor.execute(sql, val)
    mydb.commit()
    print(mycursor.rowcount, "registro insertado.")

elif button2 == True:
    print("Nuevo usuario para el estacionamiento")
    nombre = input("Nombre:")
    nivel = input("Nivel de acceso. 1.-maestro, 2.-alumno, 3.-visitante):")
    clave = input("Numero de Cuenta:")
    tarjeta = leeer(pint)
    nivel = int(nivel)
    clave = int(clave)
    sql = "INSERT INTO usuarios (Nombre, Tarjeta, Nivel, Clave) VALUES (%s, %s, %s, %s)"
    val = (nombre, tarjeta, nivel, clave)
    mycursor.execute(sql, val)
    mydb.commit()
    print(mycursor.rowcount, "registro insertado.")

else:
    print("no hay coche detectado")

```

Figura 4.16 *El resto del código "addusuario.py" dice que existen tres casos. Dos de ellos es cuando se oprime un botón indicando que un coche está en la entrada o salida, es donde se corre el programa "addusuario" ingresan los datos para dar de alta y pasan la tarjeta RFID, el tercer caso es cuando al terminar el tiempo determinado para agregar un usuario no se ingresa ningún dato y finaliza el programa. En esta parte se envían todos los datos a la base de datos.*

El programa con nombre "Main.py" es un programa que se diseñó para hacer todo el funcionamiento principal del trabajo de tesis. Este programa es el que permite la entrada/salida al estacionamiento dependiendo si esta dado de alta en la base de datos. A continuación, se dará una breve explicación y el código completo se presenta en la sección 6 del apéndice.

```
42  def buscar_usuario(Tarjeta):
43      query = ("SELECT Tarjeta FROM usuarios WHERE Tarjeta=%s")
44      mycursor.execute(query,(Tarjeta,))
45      aux = 0
46      result = mycursor.fetchall()
47      for x in result:
48          aux=x[0]
49      return aux
50
51  i=0
52  servo1 = 11
53  GPIO.setup(servo1, GPIO.OUT)
54  p = GPIO.PWM(servo1, 100)
55  p.start(10)
```

Figura 4.17 *En esta sección del código "Main.py" las líneas 42 a 49 tiene la función de buscar en la base de datos el código que pasa la tarjeta RFID. De la línea 51 a 55 dice que el servomotor se conecta en el pin 11 de la RaspberryPi y sus condiciones iniciales de arranque.*

```
62  while True:
63
64      button1 = GPIO.input(buttonEntry)
65      button2 = GPIO.input(buttonExit)
66
67      if button1 == True:
68          p.ChangeDutyCycle(6.5)
69          i+=1
70          print("Acceso para Entrada")
71          tarjeta = leer(pino)
72          subprocess.call(['raspistill','-o','/home/pi/Pictures/imag%s.jpg' % i])
73          a = buscar_usuario(tarjeta)
74          if tarjeta == a:
75              print("Tarjeta encontrada")
76              q.ChangeDutyCycle(3.5)
77              direccion = 1
78              sql = "INSERT INTO registros (Tarjeta, Lector, Fecha) VALUES (%s, %s,NOW())"
79              val = (tarjeta, direccion)
80              mycursor.execute(sql, val)
81              mydb.commit()
82              print(mycursor.rowcount, "registro insertado.... PASE!")
83              time.sleep(10)
84              q.ChangeDutyCycle(10)
85          else:
86              print("Acceso denegado")
```

Figura 4.18 *En esta sección del código "Main.py" y el resto, se crea un ciclo con el comando "while True:" y todo lo que este dentro de él se repetirá. De la línea 64 a la 86 indica que al oprimir alguno de los botones se seleccionara la entrada o salida la cual tiene algunas indicaciones diferentes, dentro de estas líneas se lee la tarjeta RFID que pasa por el lector y la variable "tarjeta" busca el valor dentro de la base de datos. Si el valor se encuentra en la base permite el acceso y si no lo encuentra aparecerá un mensaje con "Acceso denegado". Esto para los dos botones. Al final se registra la entrada/salida.*

4.4 Funcionamiento del sistema.

Para que el sistema funcione, siempre tiene que estar operando el programa “Main.py”, tiene que a ver una conexión a internet y debe existir un operador capacitado para que, en caso de ser necesario por algunas fallas como apagones de luz, reiniciar el sistema.

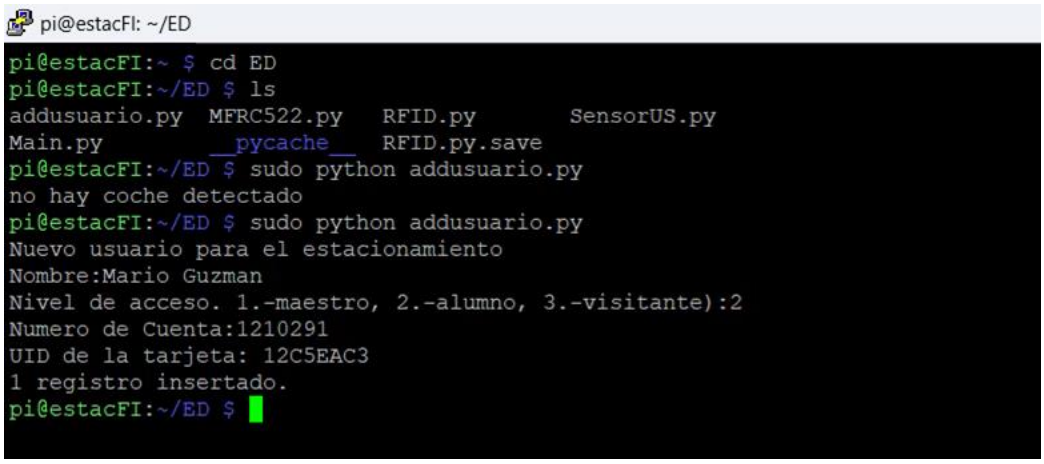

Figura 4.10 *Función del programa “addusuario.py”. Aparece el cómo se ve cuando se registra a un usuario a la base de datos. Todo esto se hace desde la consola mediante PuTTy conectado a la RaspberryPi. Para que el programa funcione se debe mantener un botón de los que se encuentran en la maqueta y así poder realizar el registro. En caso de que no sea oprimido el botón mandara un mensaje como se muestra se muestra en la figura con el contenido de “no hay coche detectado”.*

Figura 4.11 *Usuarios registrados en base de datos. Se muestran todos los usuarios registrados exitosamente en la base de datos de phpmyadmin.*

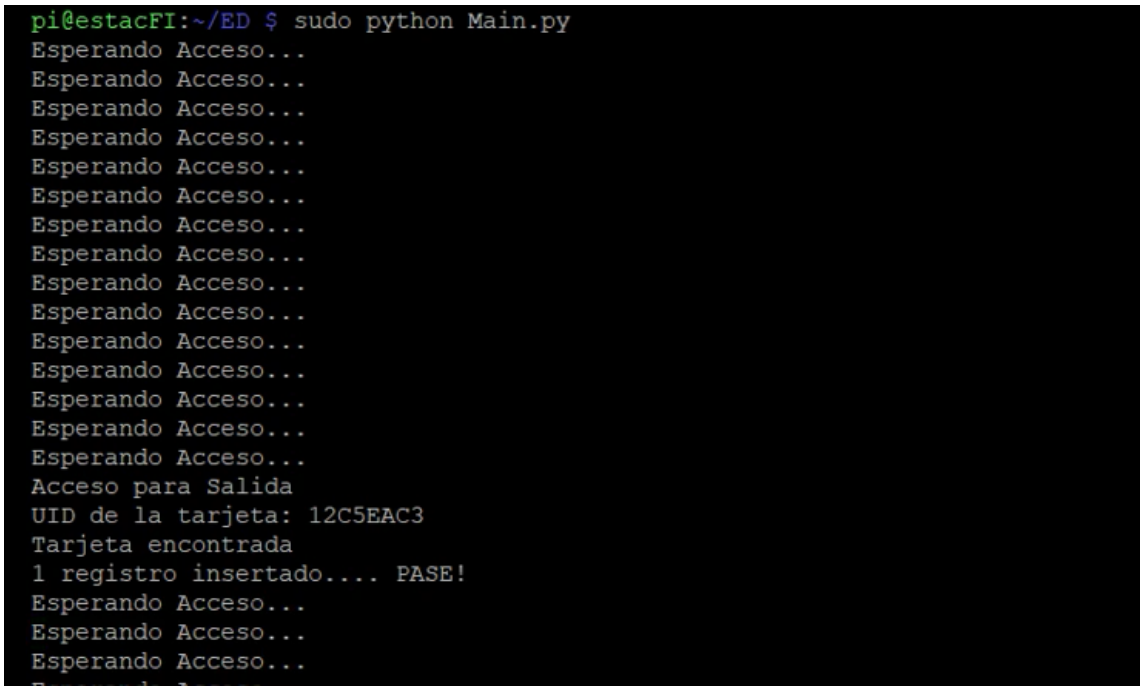

Figura 4.12 *Función del programa "Main.py". Se muestra como se ve el programa "Main.py" funcionando. Todo esto se hace desde la consola mediante PuTTy conectado a la Raspberry Pi. Este programa es el que permite la entrada/salida de los autos que previamente se registraron en la base de datos. Si la tarjeta RFID fue registrada, el programa buscará su código y permitirá el paso. Después de eso el programa seguirá corriendo hasta que otro auto requiera el paso.*

Figura 4.13 *Registros de entrada/salida en base de datos. Se muestran todos los registrados de entrada/salida con fecha y sentido, si en el apartado de "Lector" aparece un "1" significa que se permitió la entrada al estacionamiento y si aparece un "2" significa que se permitió la salida del estacionamiento.*

Para más información de cómo es el funcionamiento del sistema se realizó un video en el siguiente enlace explicando detalladamente los comandos y funcionamiento de la tesis: https://www.youtube.com/watch?v=l6bUrnNIkfM

4.5 Costos del sistema diseñado

A continuación, se presenta un listado de los materiales, dispositivos y herramientas utilizados para la realización del prototipo.

Tabla.2 *Precios utilizados del sistema desarrollado*

Cantidad	Descripción	Precio
28 pzs	Jumpers h-h	MX$45
1 rollo	Soldadura 60/40 100g	MX$175
20 m	Cable UTP	MX$80
1	Fuente de computadora 500W	MX$450
1	Cable HDMI a micro HDMI de 1.5 metros	MX$60
1	MicroSD 16GB	MX$110
1 rollo	Cinta de aislar	MX$20
1	Kit lector RFID	MX$150
2	Servo motor	MX$120
1	Raspberry Pi4 con funda protectora	MX$3500
1	Maqueta	MX$400
1	Protoboard	MX$90
1	Caja de acero para introducir el sistema	MX$500
20 m	Cable coaxial	MX$20

CONCLUSIONES

Una de las observaciones que se notaron al realizar las pruebas con el lector MFRC522 es que la distancia media para obtener una buena lectura es entre 2 y 3 cm desde la base. Así que para la realización de la instalación real de la base para el lector de los tags es una desventaja ya que el lector debe estar lo más cerca posible del automóvil el cual accederá con su tag.

El costo total del prototipo es de MX$ 5,740.00 sin incluir pluma y poste. Para comprobar la hipótesis se deben considerar estos valores al valor aproximado de los sistemas comerciales proporcionado por los proveedores, ya que al realizar la comparativa con los valores cotizados de los proveedores incluye el valor de estos. Un brazo mecánico de alto rendimiento junto con un poste para sostener la caja protectora y el precio del prototipo tiene un precio aproximado de MX$45,500.00. Esto comprueba que el sistema de acceso vehicular propuesto en esta tesis no es 6 veces más barato que un sistema comercial de MX$81,313.262 promedio. Por lo tanto, la hipótesis no es verdadera.

En la parte de instalación de programas como OpenCV y otros, se recomienda ver más formas de instalación ya que cada versión de los programas tiene distintas formas de instalación.

En la parte del hardware se observó que los lectores RFID pueden tener algunos falsos contactos, se recomienda tener una buena conexión e instalación de los dispositivos para prevenir fallas.

Una desventaja del trabajo propuesto es que si existieras un error del sistema el operador capacitado tardaría al menos 10 minutos en solucionar el problema, ocasionando la obstrucción y una fila de personal que desee acceder al estacionamiento.

Finalmente, estos diseños podrán ser utilizados para una mejora del sistema de acceso vehicular actual en la Facultad de Ingeniería. El objetivo de este proyecto final es proponer alternativas para mejorar los procesos de admisión que se puedan aplicar en el futuro en las instalaciones de la CU de la Universidad del Estado de México.

BIBLIOGRAFIA

SICDE (2020). Información de control escolar, Facultad de Ingeniería, UAEMex.

Sánchez, A. (2020, 07 de diciembre). 'Acelera' 4.7% la exportación de autos en México en noviembre. El financiero. Recuperado de: https://www.elfinanciero.com.mx/empresas/acelera-4-7-la-exportacion-de-autos-en-mexico-en-noviembre/

Alvarado, J. (2008). Sistema de control de acceso con RFID (Tesis de pregrado). Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional. México D.F. Recuperado de: http://profesores.sanvalero.net/~arnadillo/Documentos/Apuntes/Tecnicas/UD9_Comunicaci%F3n inal%E1mbrica/RFID/Sistema de control de acceso con RFID.pdf

Sánchez, O. (2016). Servicios en la Nube con Microsoft Azure "Sistema de Control de Estacionamientos de la Policía Municipal de Móstoles" (Tesis de pregrado). Universidad Politécnica de Madrid. España. Recuperado de: https://oa.upm.es/42505/10/TFG_OSCAR_MORENO_SANCHEZ.pdf

Rincón, J., Laguna, C. (2018). Sistema de registro y control vehicular por medio de autenticación biométrica con acceso móvil (Tesis de pregrado). Universidad Distrital Francisco José de Calcas. Colombia. Recuperado de: https://repository.udistrital.edu.co/bitstream/handle/11349/15956/Rinc%C3%B3nL%C3%B3pezJuli%C3%A1nAndr%C3%A9s2019.pdf?sequence=1&isAllowed=y

Gomero, L. (2017). Diseño de un sistema de acceso vehicular a la PUCP basado en tecnología RFID y detección de placas vehiculares (Tesis de pregrado). Pontificia Universidad Católica de Perú. Perú. Recuperado de: https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/9388/GOMERO_LUIS_ACCESO_VEHICULAR_RFID_PLACAS_VEHICULARES.pdf?sequence=1&isAllowed=y

Ortiz, E., Ibarra, M., Andrade, J. y Almanza, D. (2012). Control de acceso usando FPGA y RFID (Acta universitaria). Universidad de Guanajuato. México. Recuperado de: https://www.redalyc.org/pdf/416/41624636005.pdf

Enríquez C., Fernández R. (2007), CONFIGURANDO LA COMPUTADORA RASPBERRY PI COMO SERVIDOR WEB. (Articulo) Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas (UPIITA). Instituto Politécnico Nacional. Recuperado de: http://www.boletin.upiita.ipn.mx/index.php/ciencia/510-cyt-numero-39/375-configurando-la-computadora-raspberry-pi-como-servidor-web

Digi-Key Electronics. (2019), Por qué y cómo usar la interfaz periférica serial para simplificar las conexiones entre distintos dispositivos. (Articulo) Recuperado de: https://www.digikey.com.mx/es/articles/why-how-to-use-serial-peripheral-interface-

simplify-connections-between-multiple-devices -:~:text=CPOL%20%3D%200%20establece%20el%20reloj%20en%20inactivo,reloj%20en%20el%20borde%20de%20rastreo%20%28Figura%203%29.

AUTOSISTEMAS (2023). Barreras Vehiculares (Página de ventas).
Recuperado de: https://www.autosistemas.com/barreras

JP CRD Accesos y Controles (2023). Accesos y Controles (Página de ventas).
Recuperado de: https://www.accesosycontroles.com/

BARTEK System (2023). Estacionamiento y Acceso Vehicular (Página de ventas)
Recuperado de: https://bartek.com.mx/

GLOSARIO

RFID - Radio Frequency Identification en español, Identificación por Radio Frecuencia.

FPGA - matriz de puertas lógicas programable en campo. Conjunto de circuitos integrados, como puede ser cualquier chip

Linux - Es un sistema operativo completamente libre y, por lo tanto, gratuito.

GPIO - Es un pin genérico en un chip.

Script - Es un término usado en programación para hablar de los fragmentos de código.

MariaDB - Es un sistema de gestión de bases de datos relacionales (RDBMS) gratuito y de código abierto.

OpenCV - Es una biblioteca libre de visión artificial originalmente desarrollada por Intel. OpenCV significa Open Computer Vision (Visión Artificial Abierta).

Relevador - Son dispositivos electromagnéticos que se encargan de abrir y cerrar el paso de la corriente eléctrica.

JPG - Es un formato de archivo de imagen que se utiliza para almacenar imágenes y fotografías en formato digital.

PHPMyAdmin - Es una aplicación web que sirve para administrar bases de datos MySQL de forma sencilla.

Jumper - Es un elemento que permite cerrar el circuito eléctrico del que forma parte dos conexiones.

Cable UTP - Es un tipo de cable de cobre.

Cable Coaxial - Cable eléctrico constituido por dos conductores concéntricos aislados entre sí.

Python - Es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos.

Raspberry Pi - Ordenadores de placa reducida.

Servidor WEB - Programa informático que procesa una aplicación del lado del servidor.

Sistemas Biométricos - Sistemas automatizados de identificación y verificación de un individuo.

Tag - Conjunto de palabras claves.

RF – Radiofrecuencia.

EAN - Escuela de Administración de Negocios.

UCC – Universidad Cristóbal Colon.

ID - Se llama el nombre de usuario con el que accedemos a una página o sistema.

ISO - International Organization for Standardization en español, Organización Internacional de Normalización.

EPC - Engineering, Procurement and Construction en español, Ingeniería, Compras y Construcción.

Frecuencia - Es el número de repeticiones por unidad de tiempo de cualquier evento periódico.

ARM - Advanced RISC Machine.

HTTP - Protocolo de Transferencia de Hipertexto.

Pila LAMP - Paquete de aplicaciones y herramientas open source.

Dirección IP - Conjunto de números que identifica, de manera lógica y jerárquica, a una interfaz en la red de un dispositivo que utilice el protocolo o, que corresponde al nivel de red del modelo TCP/IP.

Protoboard - Es un instrumento que permite probar el diseño de un circuito sin la necesidad de soldar o desoldar componentes.

PHP - Lenguaje de programación de uso general que se adapta especialmente al desarrollo web.

Configuración - Clock Polarity en español, Polaridad de Reloj.

Configuración CPHAL -Clock Phase en español, Reloj de Fase.

MOSI - Microprocessor Operating System Interface.

SPI - Serial Peripherical Inerface

APÉNDICE

ÍNDICE

1 **Instalación de la pila LAMP** A2

2 **Instalación de OpenCV** A10

3 **Configuración de las RFID** A17

4 **Habilitación de la cámara** A21

5 **Base de datos** A23

6 **Códigos de línea** A27

1. Instalación de la pila LAMP

Se presentan las capturas de los comandos y como se deben instalar los programas de la pila LAMP y Open CV.

El primer paso es ingresar el usuario y contraseña. Después se escribe el comando **`sudo apt-get update`** para buscar alguna actualización, se escribe para instalar apache2 **`sudo apt-get install apache2`** seguido de la **`s`**, como se indica en la Figura 1.

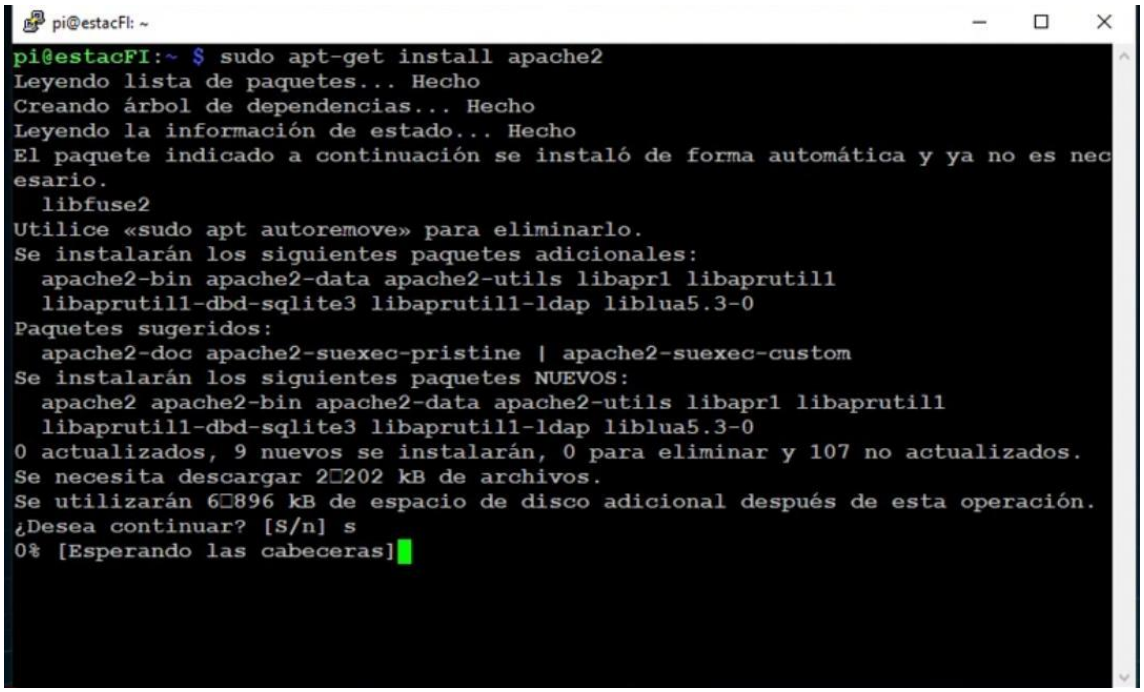

Figura 1. *Instalación de apache2*

Para comprobar que Apache se instala correctamente se ingresa el comando **`ifconfig`** para obtener la dirección IP local como se muestra en la Figura 2 y se ingresa en un navegador, donde, te lleva a la página predeterminada de Apache como se muestra en la Figura 3.

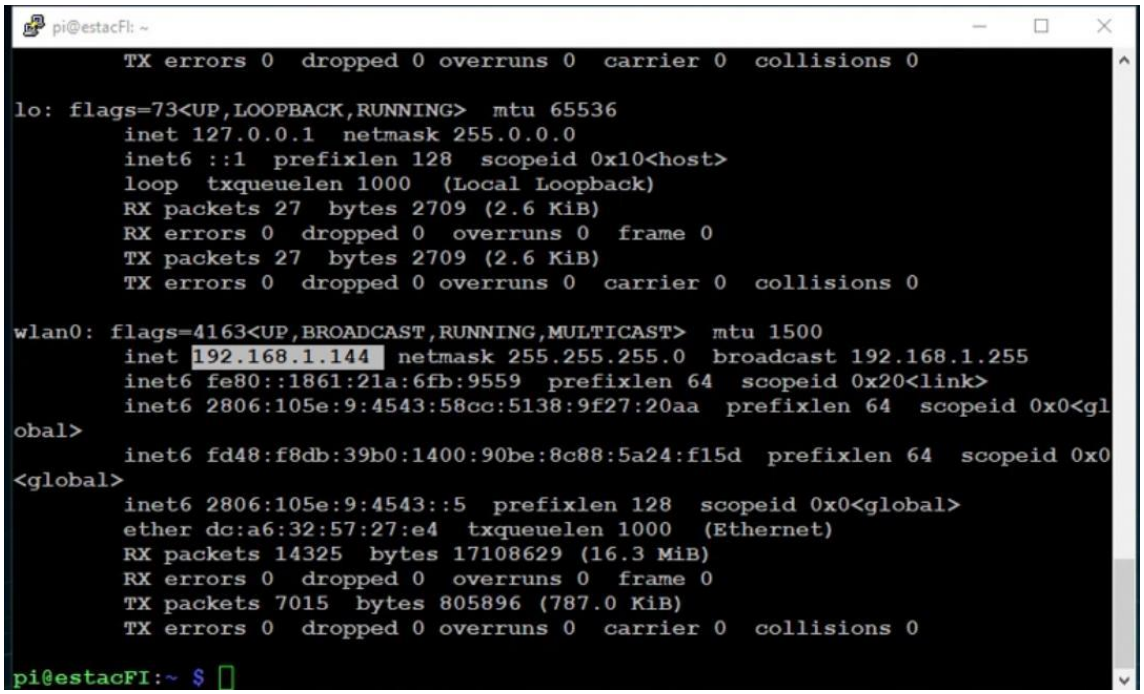

Figura 2. *IP local*

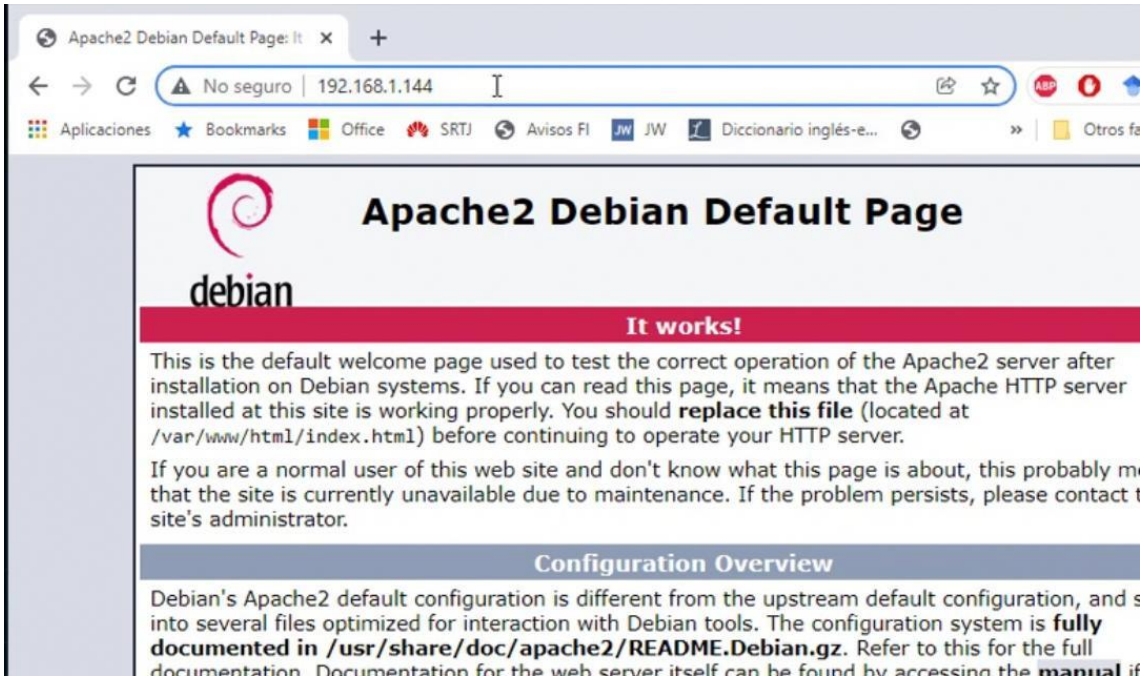

Figura 3. *Página de Apache2*

Para la instalación de PHP se ingresa el comando **`sudo apt install php libapache2-mod-php php-mysql`** y acepta con **`s`** como se muestra en la Figura 4.

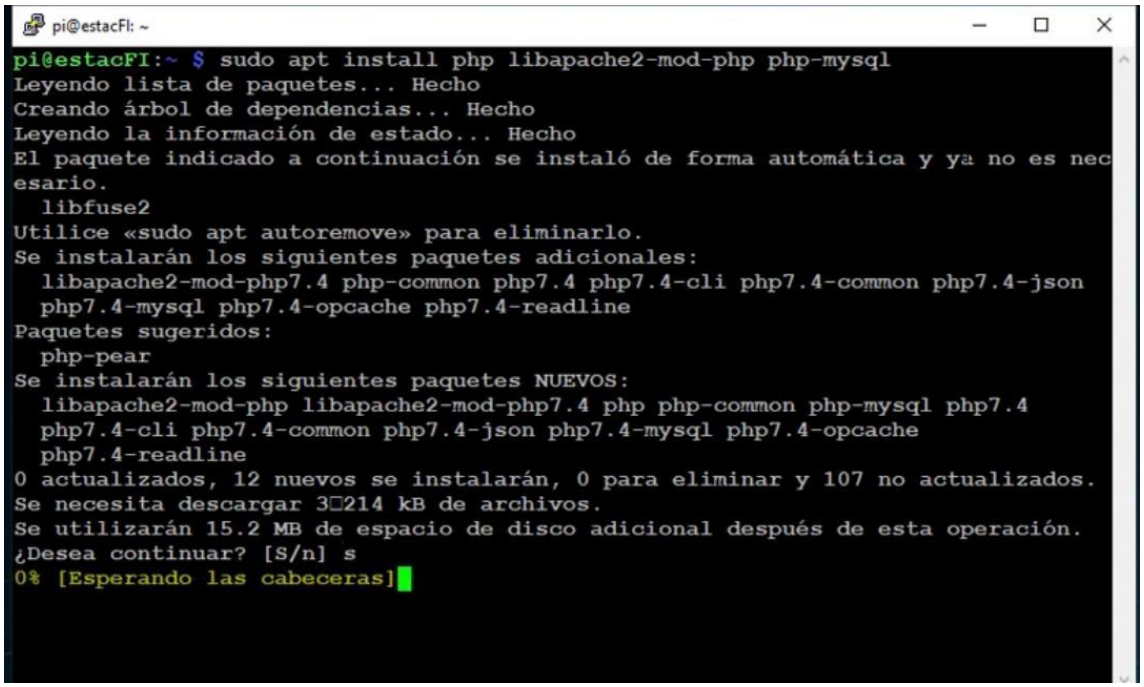

Figura 4. *Instalación de PHP*

Para verificar que php se instaló correctamente se ingresa el comando **`sudo chmod -R 777 /var/www/html`**, se crea un archivo con **`nano /var/www/html/info.php`** , en donde se abrirá una nueva pestaña y se ingresara **`<?php phpinfo(); ?>`** y en el navegador se ingresa la dirección IP/info.php como se muestra en la Figura 5.

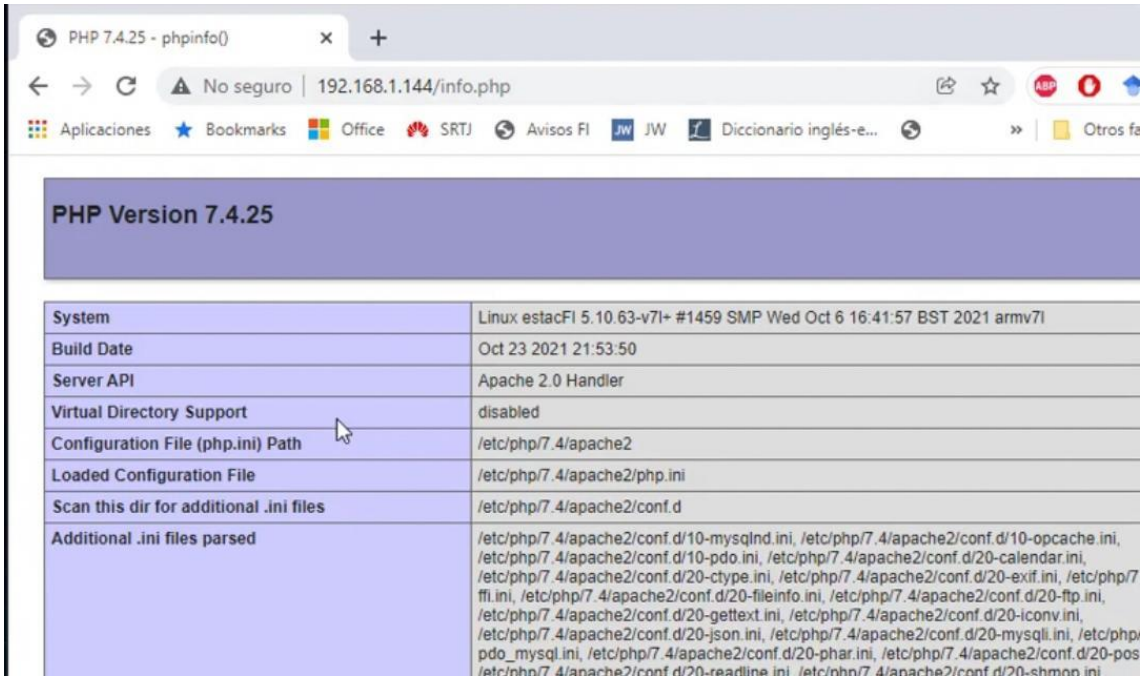

Figura 5. *Página de PHP*

Para la instalación de MariaDB se ingresa el comando **`sudo apt install mariadb-server`** y acepta con **`s`** como se muestra en la Figura 6.

Figura 6. *Instalación de MariaDB*

Ahora se cambian los parámetros que se encuentran por defecto por riesgos de seguridad. Se ingresa el comando **`sudo mysql_secure_installation`**, en la primera opción damos en **`n`**. Continua con **`Y`** para cambiar la contraseña del root y se escribe, confirmar la contraseña como se muestra en la Figura 6.1.

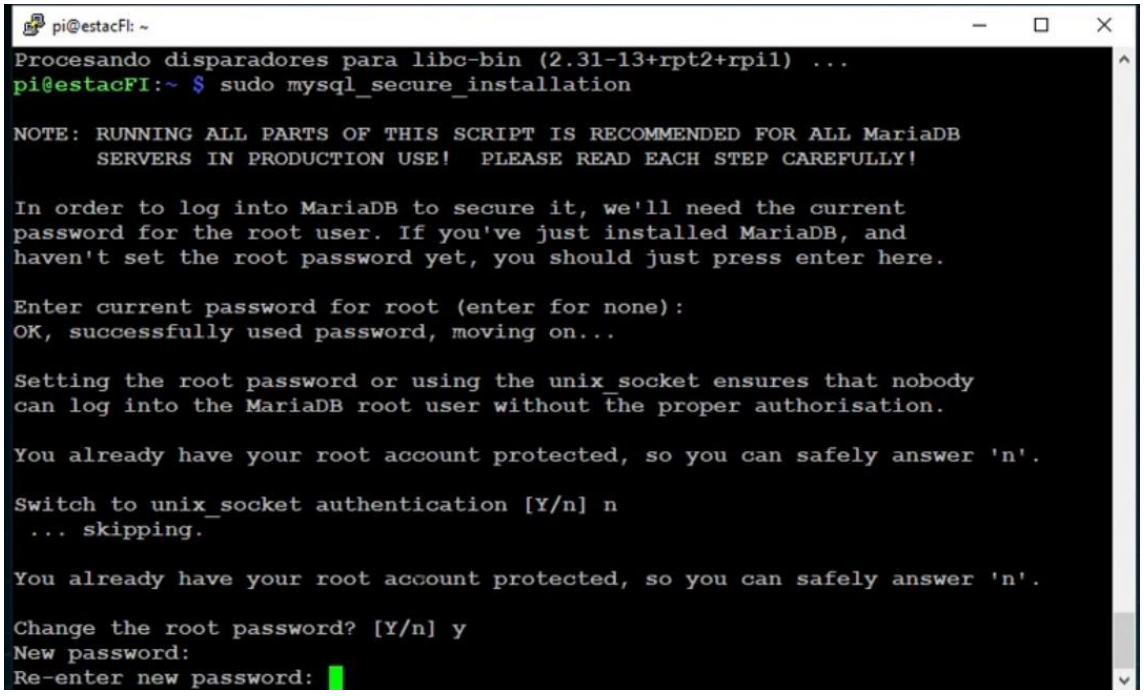

Figura 6.1. *Instalación de MariaDB*

Y continuamos con **`Y`** para remover el usuario anónimo por defaul como se muestra en la Figura 6.2.

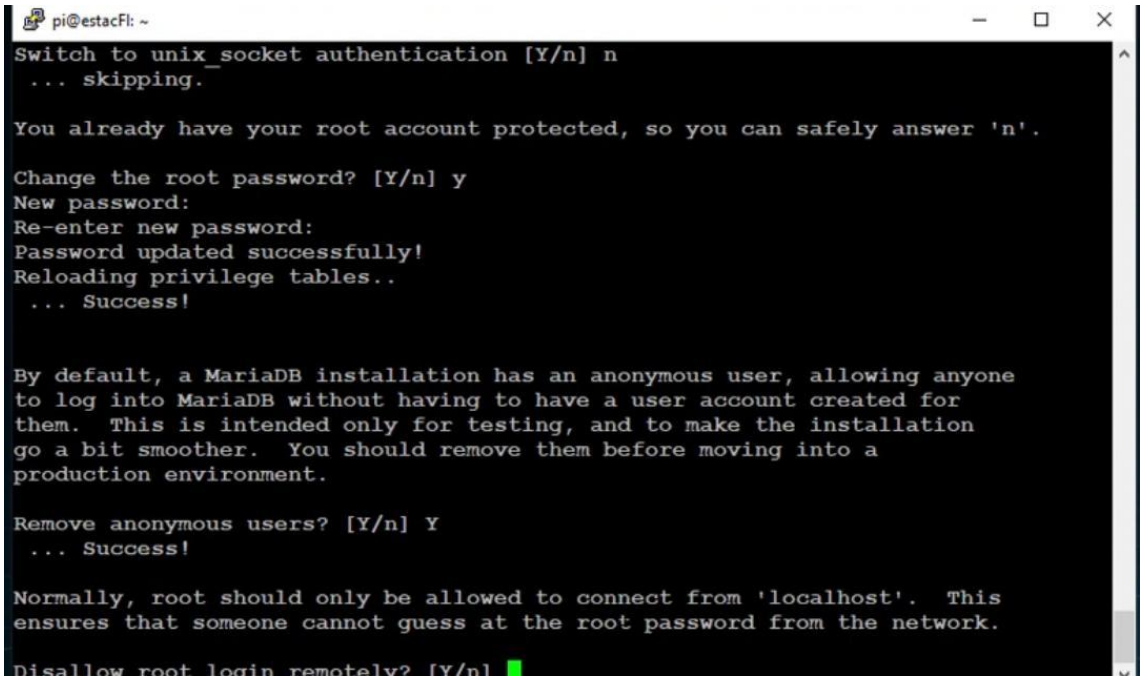

Figura 6.2. *Instalación de MariaDB*

Para las siguientes opciones continua con **`n`**, **`n`** y **`Y`** como se muestra en la Figura 6.3.

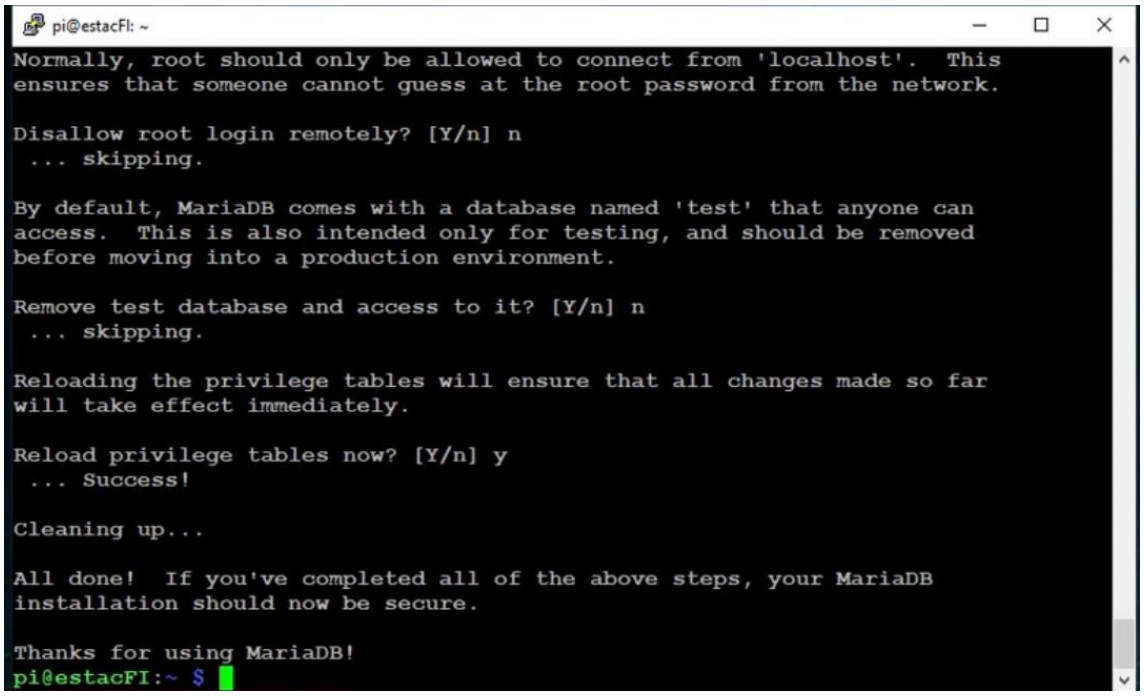

Figura 6.3. *Instalación de MariaDB*

Para la instalación de PHP myadmin con el comando **`sudo apt install phpmyadmin php-mbstring`** y **`sudo apt install phpmyadmin`**, continua con **`s`** como se muestra en la Figura 7.

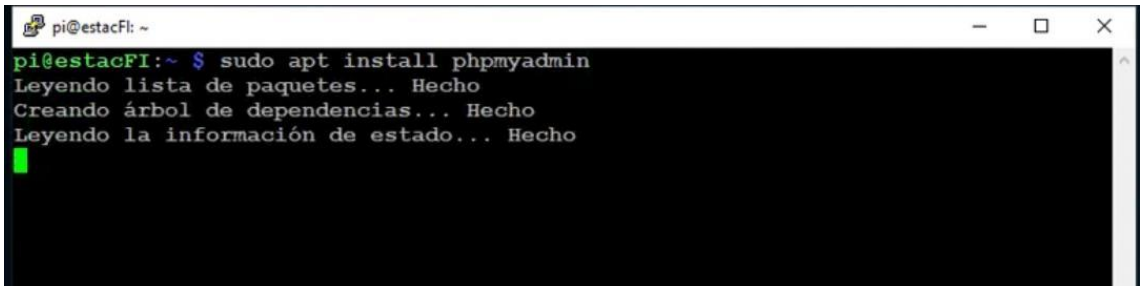

Figura 7. *Instalación de PHPMyAdmin*

El programa mostrara una pantalla azul y selecciona **apache2** con la tecla de espacio aceptar como se muestra en la Figura 7.1.

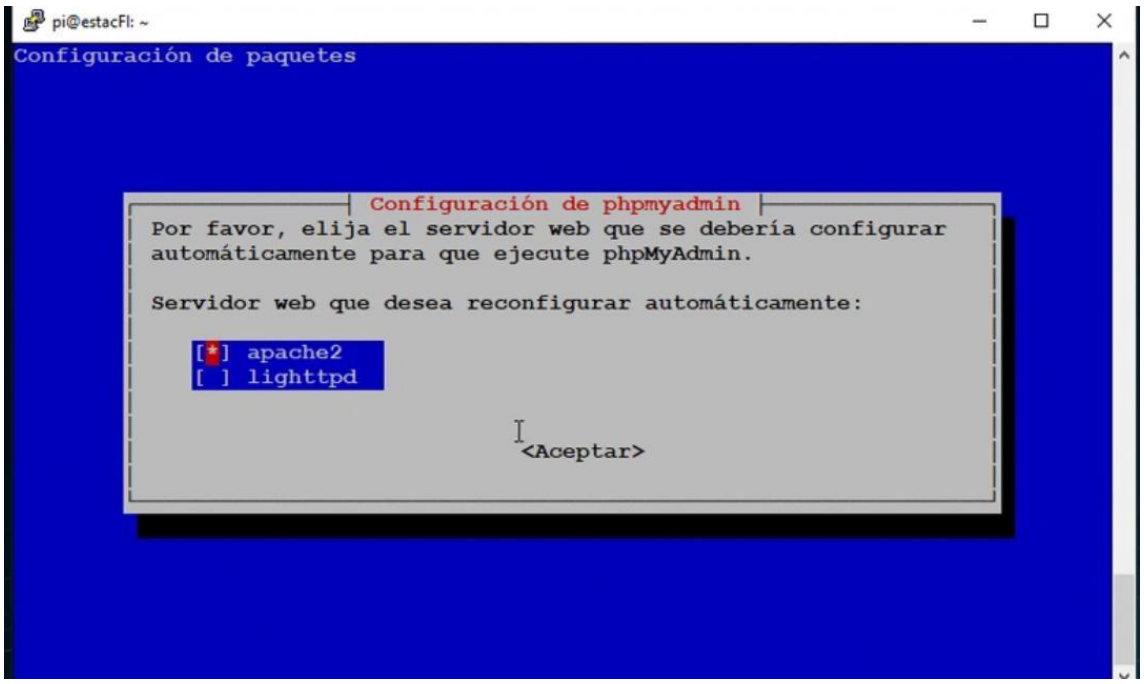

Figura 7.1. *Instalación de PHPMyAdmin*

Se mostrará otra pantalla azul para la configuración de paquetes y selecciona **si** como se muestra en la Figura 7.2.

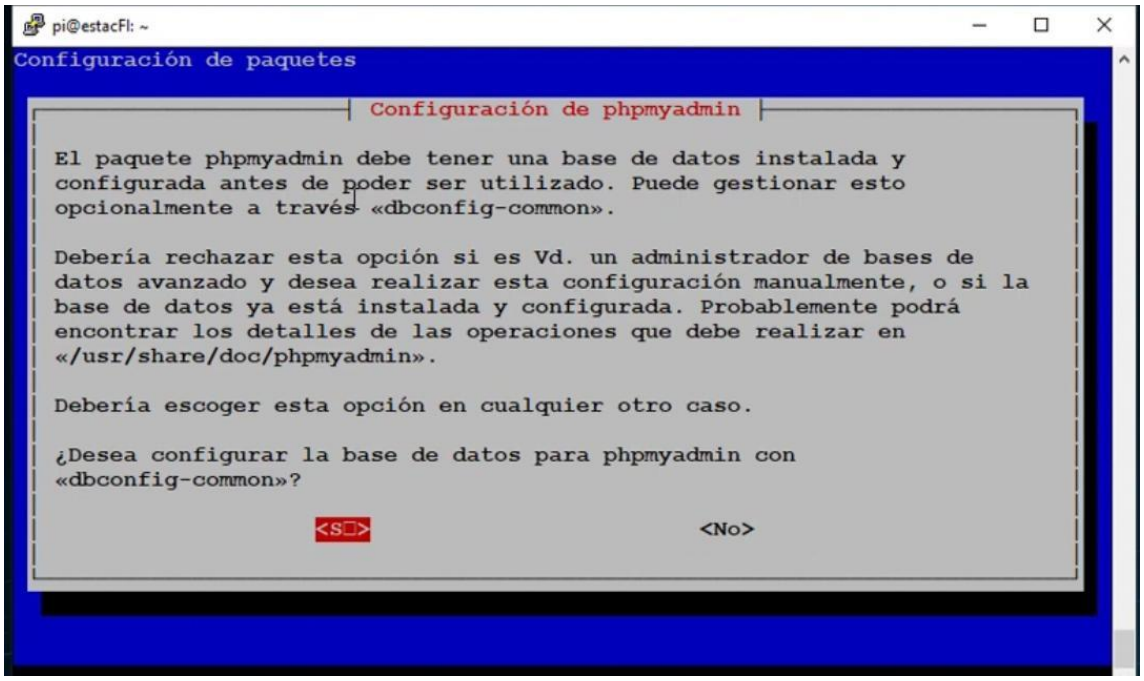

Figura 7.2. *Instalación de PHPMyAdmin*

Se mostrará otra pantalla azul pidiendo la contraseña de MySQL que se había puesto anteriormente y confirmar contraseña como se muestra en la Figura 7.3.

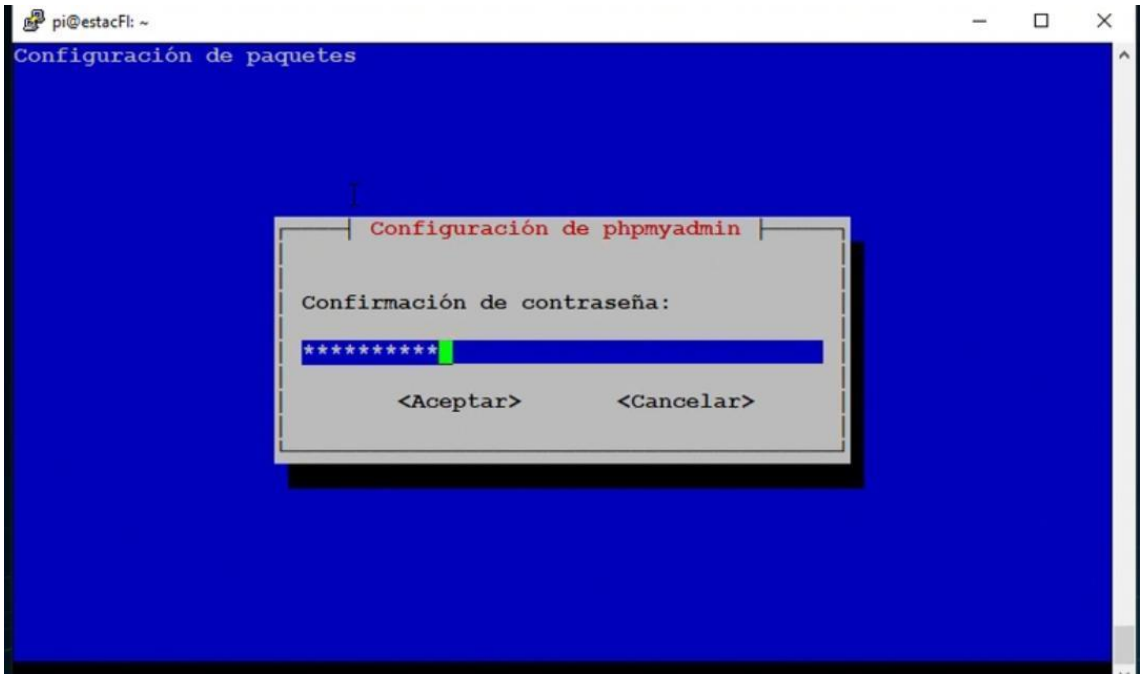

Figura 7.3. *Instalación de PHPMyAdmin*

Para probar la instalación escribir en el navegador la dirección IP/phpmyadmin, al ver la página por defaul significa que se instaló correctamente como se muestra en la figura 8.

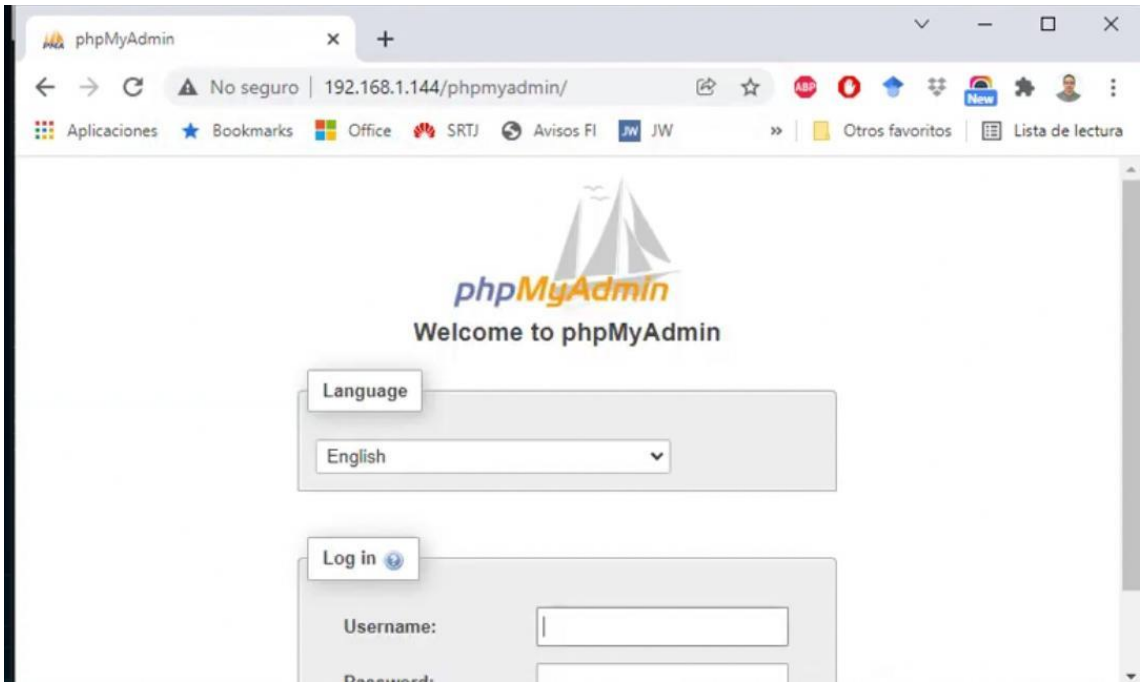

Figura 8. *Pagina PHPMyAdmin*

2. Instalación de la pila OpenCV

La instalación de OpenCV se realizara mediante el gestor de paquetes pip, el cual está instalado por defecto en la RaspberryPi, para comprobar la versión se ingresa el comando **`pip --version`** como se muestra en la figura 9.

Figura 9. *Versión del gestor de paquetes pip.*

El siguiente paso es entrar a la página oficial de piwheels: https://www.piwheels.org/ . El cual proporciona paquetes binarios recompilados específicos para Raspberry Pi, es decir, cada paquete es un traje hecho a medida para tu Raspberry Pi.

Para saber que versión de OpenCV se debe instalar, dar clic en la opción “Search” y escribir “opencv”, el paquete que se desea porque es el más completo es: “opencv-contrib-python” y damos clic como se muestra en la figura 10.1.

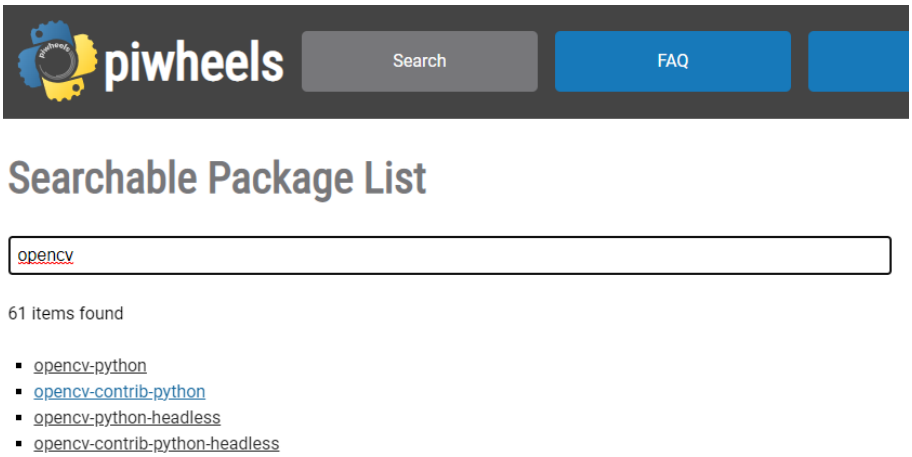

Figura 10.1 *Instalación de OpenCV.*

En seguida se abre una página con las versiones de opencv-contrib-python, donde existen 3 formatos de instalación (stretch, búster, bullseye), para saber qué tipo utilizar

se ingresa en la consola el comando **`sudo cat /etc/os-release`** como se muestra en la figura 10.2.

Figura 10.2 *Instalación de OpenCV.*

Sabiendo el tipo de formato (bullseye) se puede saber que versión de OpenCV instalar.

En este caso se instaló la versión 4.5.5.62 ya que la RaspberriPi cumple con los requisitos necesarios para esta versión. Se abre la pestaña y se da clic en “How to install this version”, se copia el comando como se muestra en la figura 10.3.

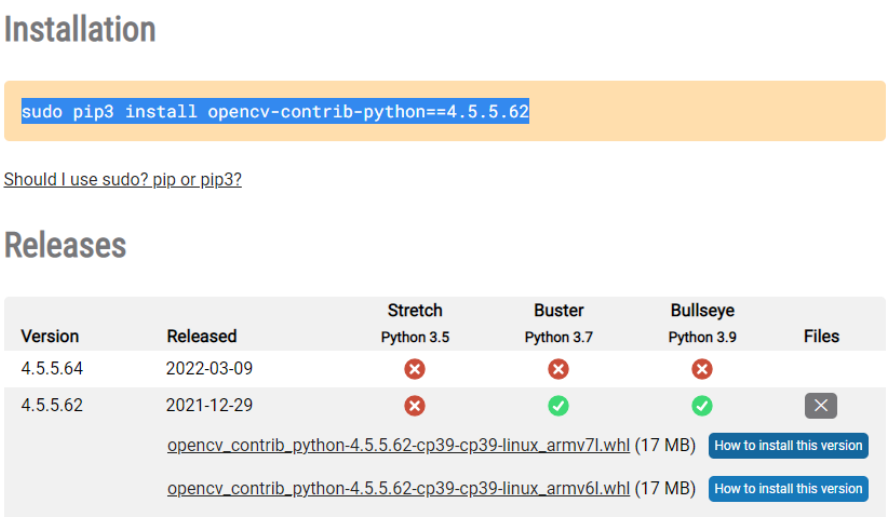

Figura 10.3 *Instalación de OpenCV.*

Antes de ejecutar el comando en la consola, se debe crear un entorno virtual de python. Para ello se deben instalar dos programas con el comando **`sudo pip3 install virtualenv virtualenvwrapper`** como se muestra en la figura 10.4.

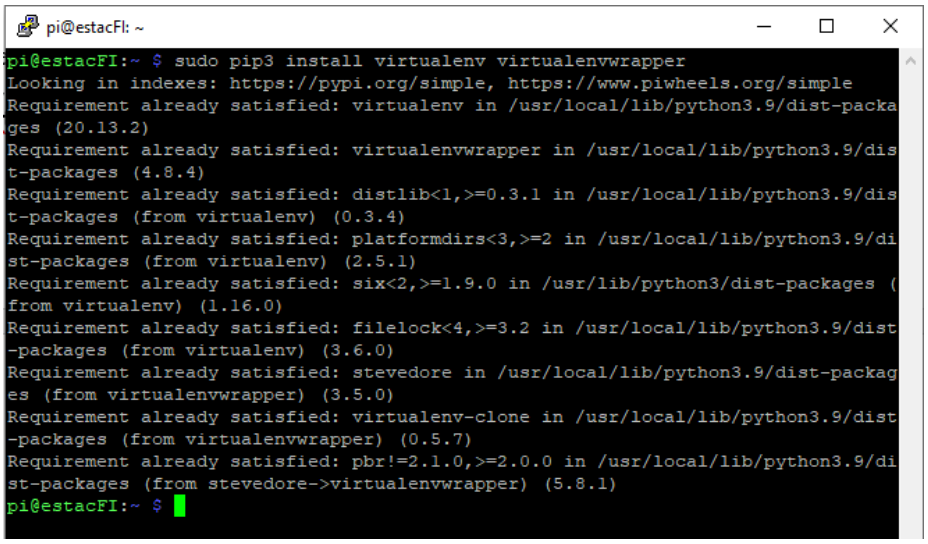

Figura 10.4 *Instalación de OpenCV.*

Esto instalará los paquetes necesarios para crear entornos virtuales de Python. Ahora se debe editar el archivo "*~/.bashrc"* ejecutando el comando **`sudo nano ~/.bashrc`**, El comando anterior abrirá el archivo en el editor *nano*. Ir al final del archivo y añadir las siguientes líneas:

```
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```

como se muestra en la figura 10.5.

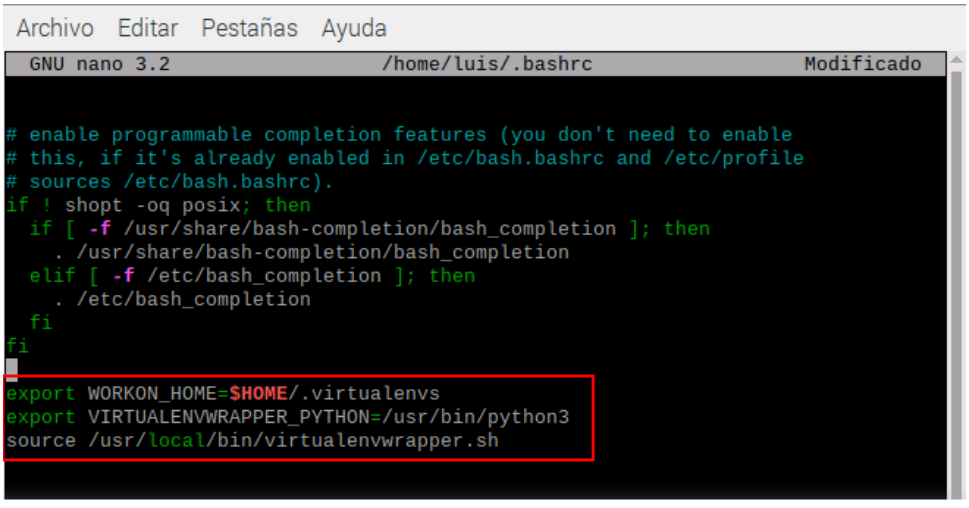

Figura 10.5 *Instalación de OpenCV.*

Ahora pulsar las teclas CTRL + x para salir. Aparecerá un mensaje preguntando si quieres guardar los cambios como se muestra en la figura 10.6.

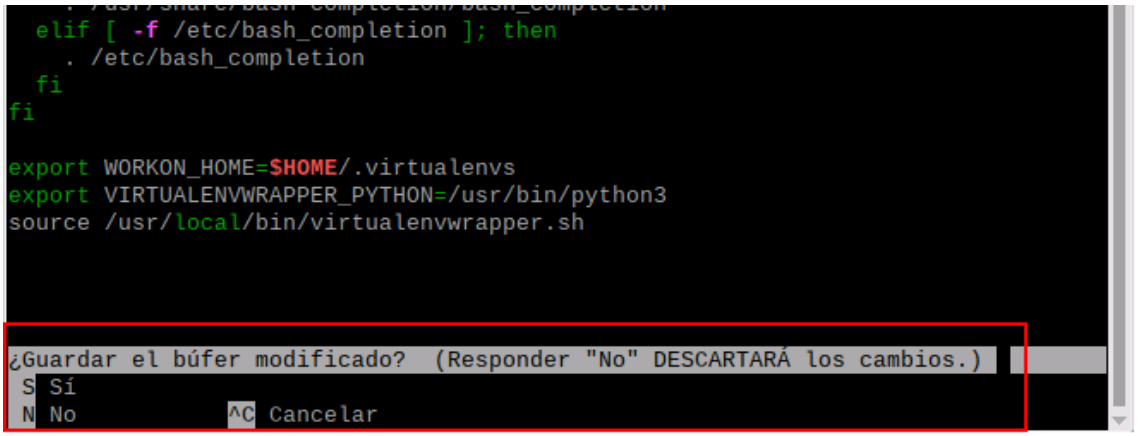

Figura 10.6 *Instalación de OpenCV.*

Presionar la tecla “s” y luego aparecerá otro mensaje preguntando por el nombre de archivo como se muestra en la figura 10.7.

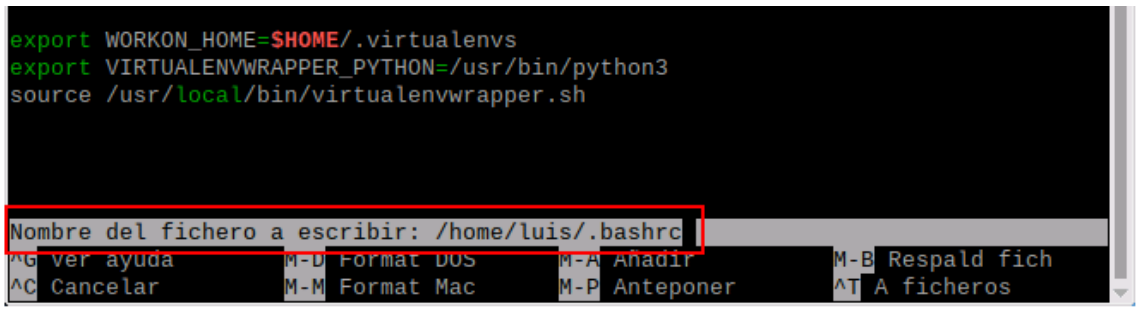

Figura 10.7 *Instalación de OpenCV.*

Dejar el nombre del archivo original. *bashrc*. Solo falta actualizar el terminal ejecutando el siguiente comando **`source ~/.bashrc`**.

Para comenzar a crear el entorno virtual se ingresa el comando **`mkvirtualenv`** + **`(El nombre del entorno virtual)`** que en este saco seria: **`mkvirtualenv opencv4-5-tesis`** como se muestra en la figura 10.8.

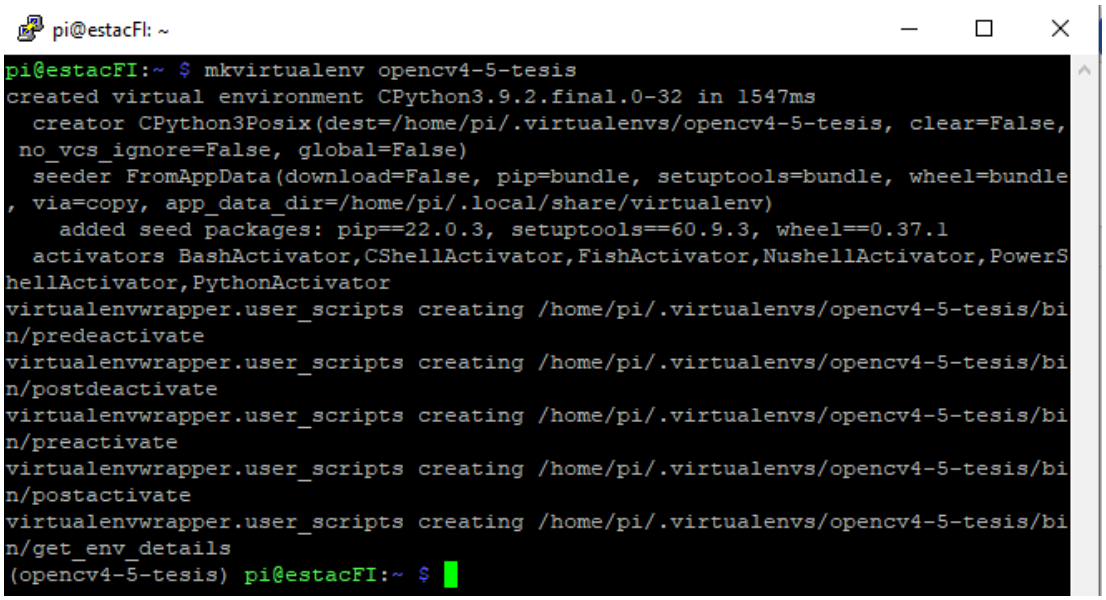

Figura 10.8 *Instalación de OpenCV.*

Para entrar al entrono virtual ya creado se utiliza el comando **`workon opencv4-5-tesis`**, para salir del entorno se utiliza en comando **`deactivate`** como se muestra en la figura 10.9.

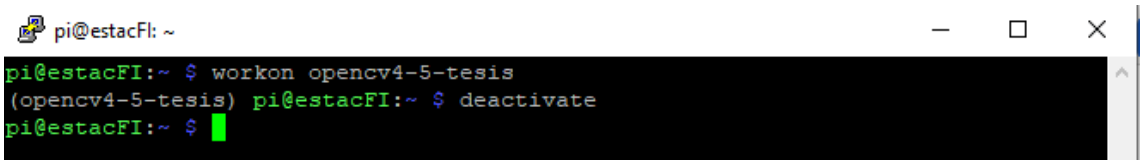

Figura 10.9 *Instalación de OpenCV.*

Dentro del entorno virtual pegamos el comando que obtuvimos de piwheels que es: **`sudo pip3 install opencv-contrib-python==4.5.4.60`** como se muestra en la figura 10.10.

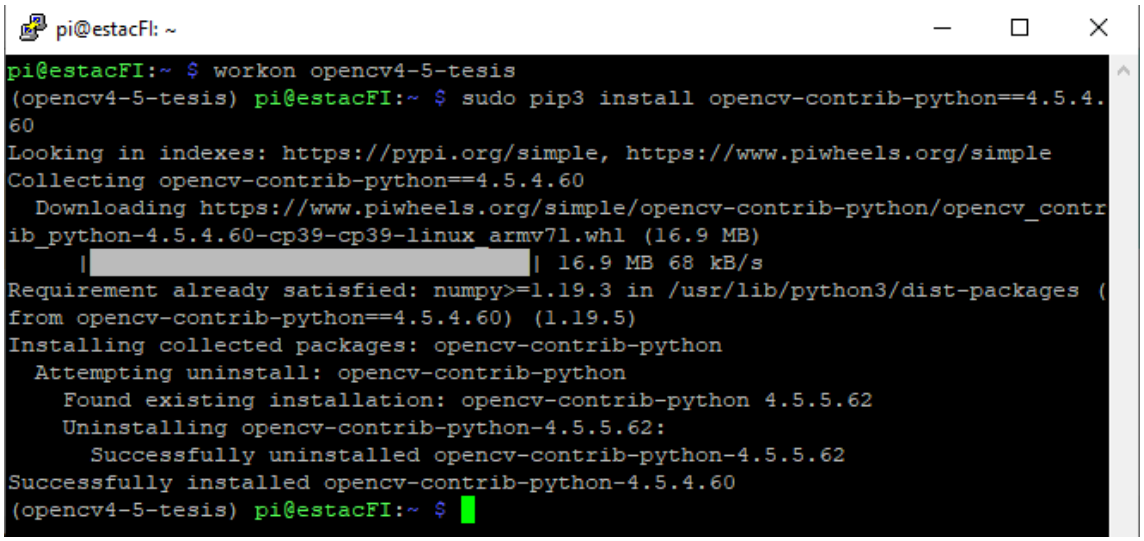

Figura 10.10 *Instalación de OpenCV.*

Al igual se instalan otras librerías que servirán para esta tesis con los siguientes comandos que fueron recopilados de piwheels de la misma manera en que se buscó OpenCV:

```
sudo apt install libgfortran5 libatlas3-base
sudo pip3 install numpy==1.22.3
sudo pip3 install pytesseract==0.3.9
```

Para verificar su instalación se ingresan los siguientes comandos, como se muestra en la figura 10.11:

```
python
import cv2
import imutils
import pytesseract
import numpy as np
from PIL import Image
```

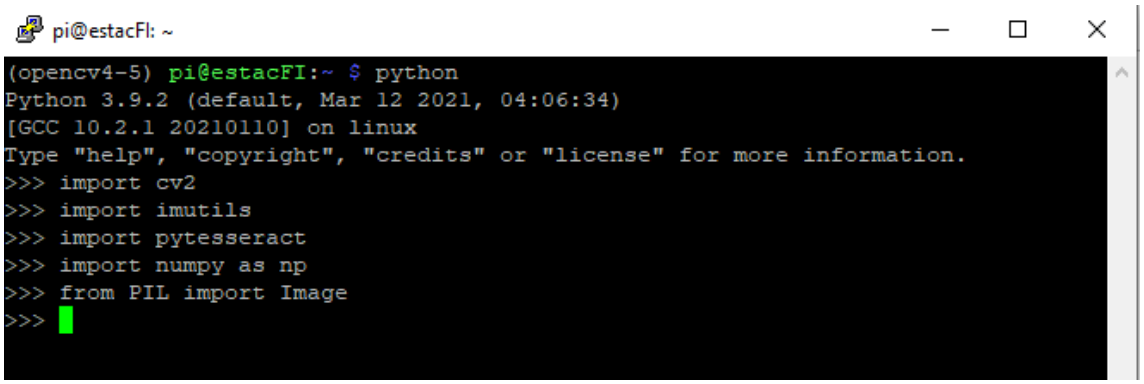

Figura 10.11 *Instalación de OpenCV.*

3. Configuración de las RFID

La interfaz entre la Raspberry Pi y la tarjeta RFID es mediante el protocolo SPI, por lo tanto, se debe habilitar la Raspberry Pi editar el archivo config.txt con el comando **`sudo nano /boot/config.txt`** como se muestra en la figura 11.1.

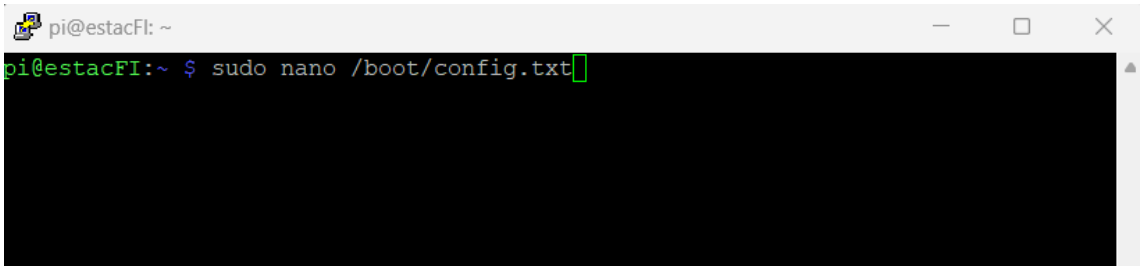

Figura 11.1. *Configurar Raspberry con protocolo SPI*

Al final del archivo config.txt agregar las siguientes líneas como se muestra en la figura 11.2.:

```
device_tree_peram = spi = on
dtoverlay = spi-bcm2708
```

Lo guardamos y salimos a la parte principal de la consola.

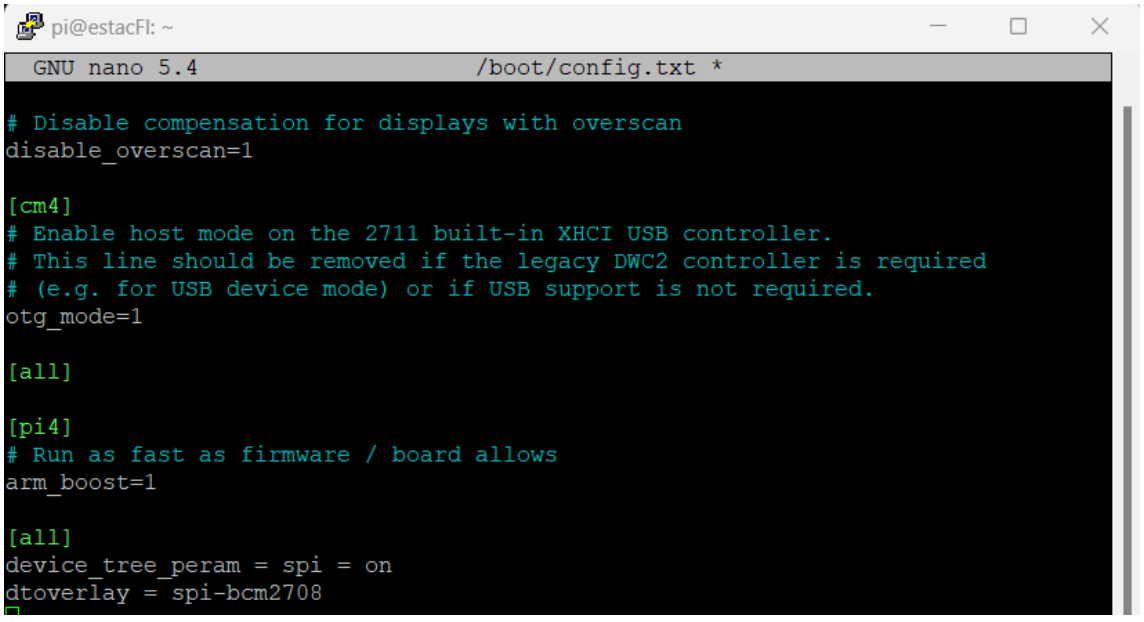

Figura 11.2. *Configurar Raspberry con protocolo SPI*

A continuación, ingresamos el siguiente comando para entrar a la configuración de la Raspberry con el comando **`sudo raspi-coonfig`** como se muestra en la figura 11.3.

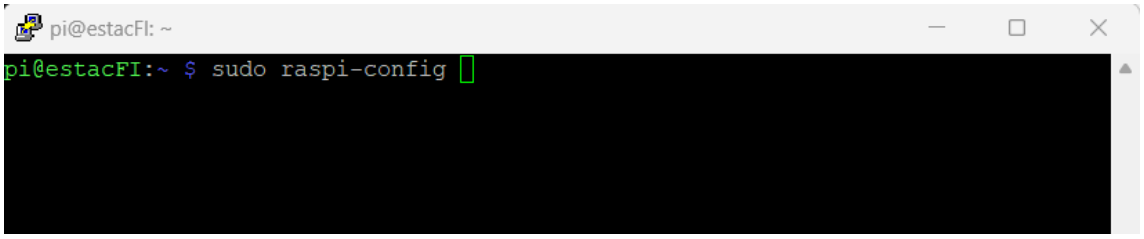

Figura 11.3. *Configurar Raspberry con protocolo SPI*

Aparecerá una ventana con varias opciones y se seleccionará la que diga “Interface Opciones” como se muestra en la figura 11.4.

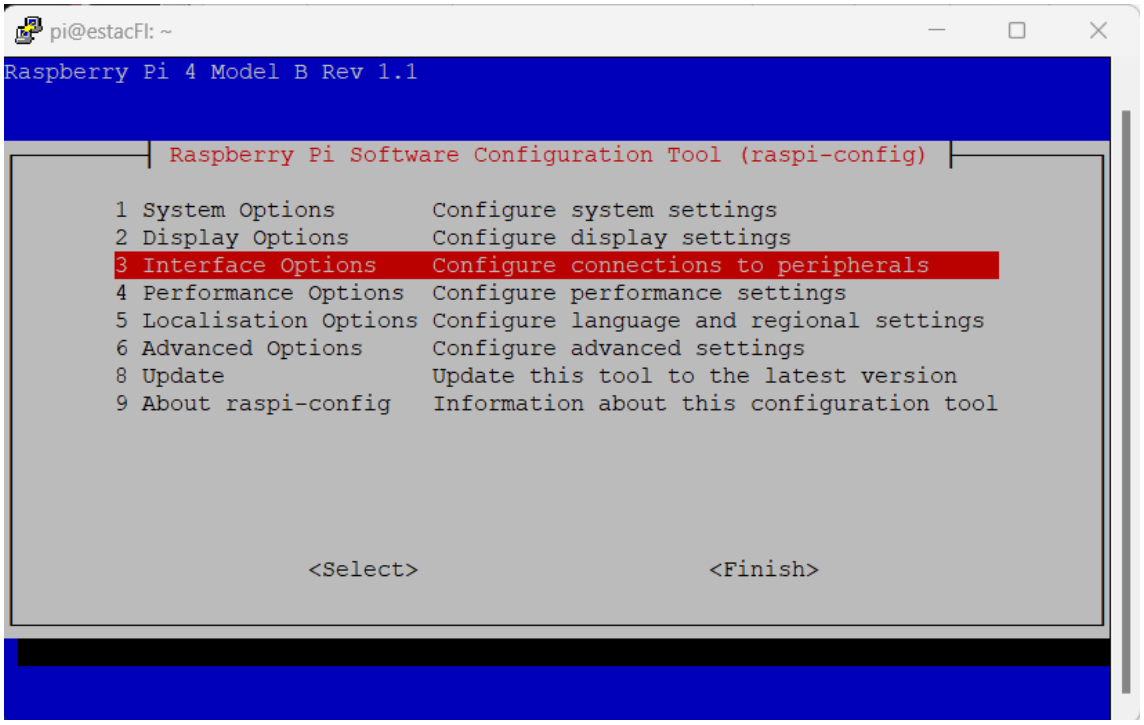

Figura 11.4. *Configurar Raspberry con protocolo SPI*

Después escogemos la opción SPI como se muestra en la Figura 11.5.

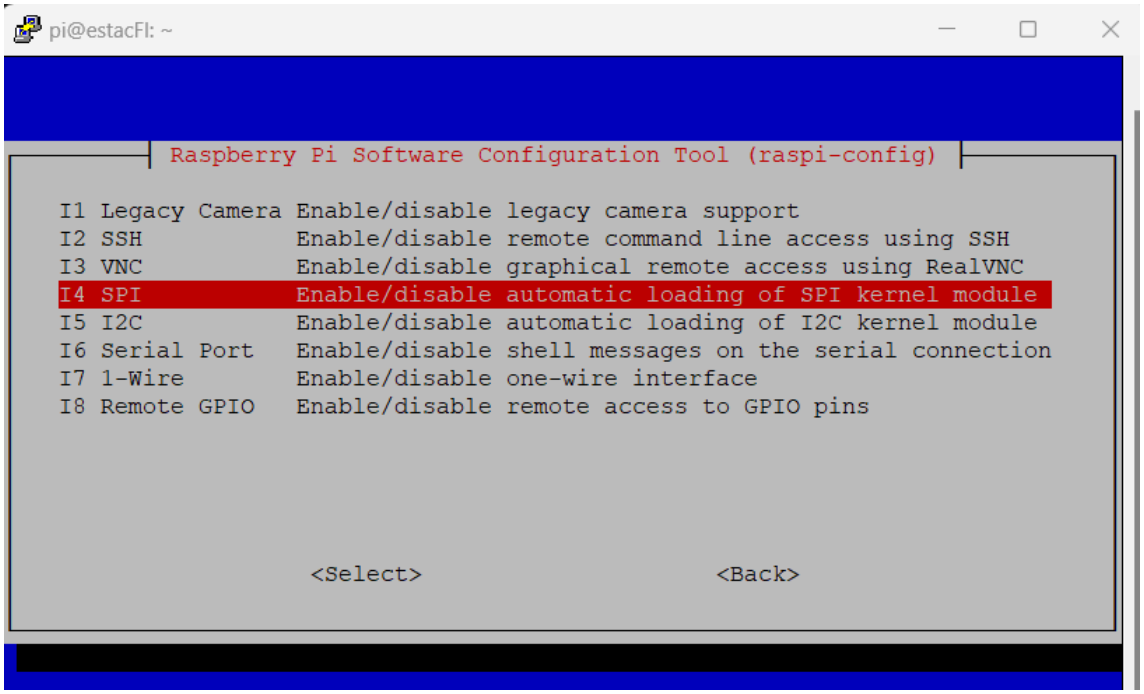

Figura 11.5. *Configurar Raspberry con protocolo SPI*

Aparecerá una ventana pregunrando si se desea habilitar la interfaz SPI y se seleeciona la opción “sí” como se muestra en la figura 11.6.

Figura 11.6. *Configurar Raspberry con protocolo SPI*

Para comprobar que existe la comunicación con ese puerto se escribe el siguiente comando **`dmesg | grep spi`** y debe aparecer la siguiente lectura como en la Figura 11.7.

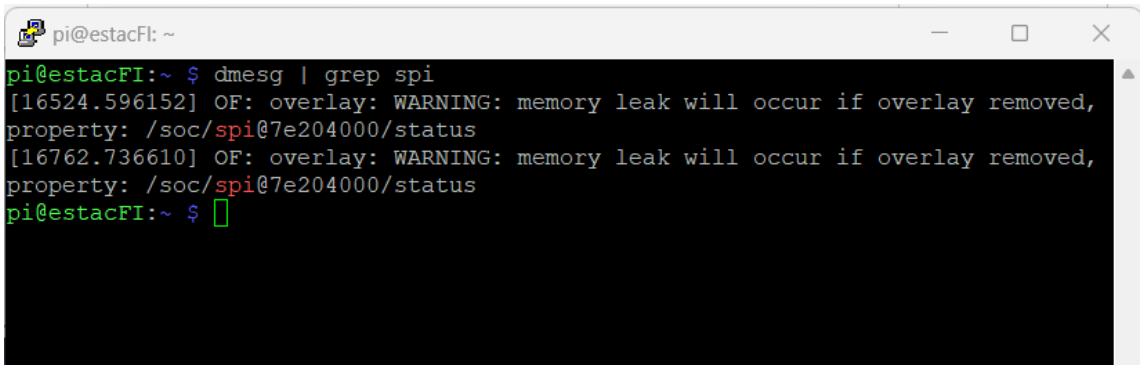

Figura 11.7. *Configurar Raspberry con protocolo SPI*

4. Habilitación de la Cámara

Activar la cámara en la Raspberry es muy sencillo. Se debe ingresar el siguiente comando **`sudo raspi-config`** como se muestra en la Figura 12.1.

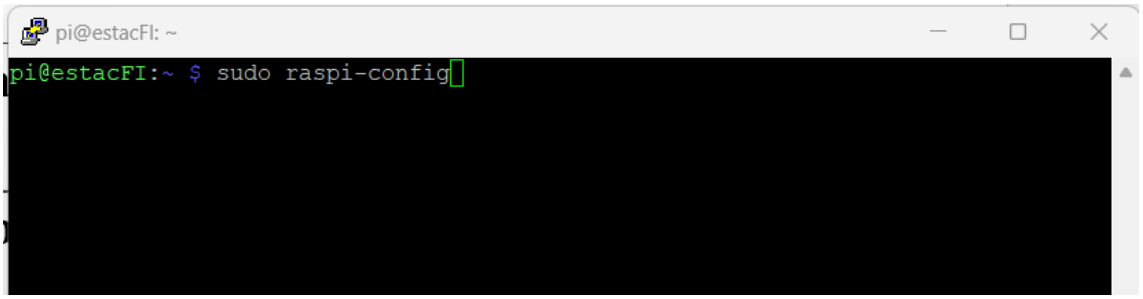

Figura 12.1. *Habilitar cámara*

Enseguida se abrirá una ventana azul con varias opciones, se debe seleccionar la que diga "Interface Opciones" como se muestra en la Figura 12.2.

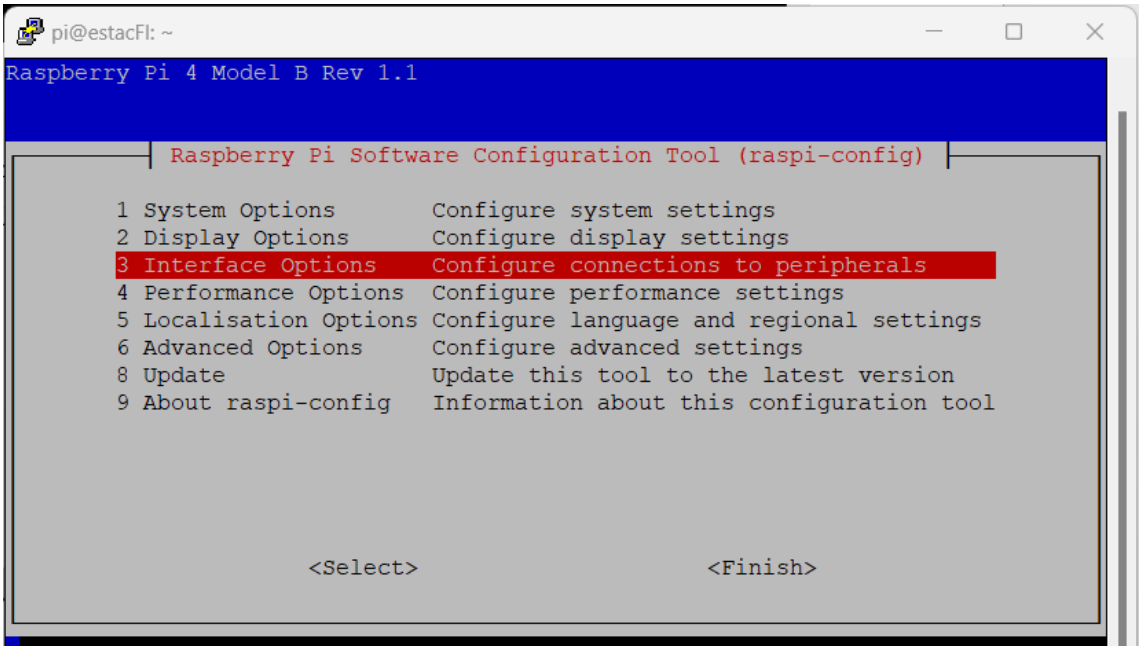

Figura 12.2. *Habilitar cámara*

Se abrirá otra ventana azul y se selecciona "Legacy Camera" como se muestra en la Figura 12.3.

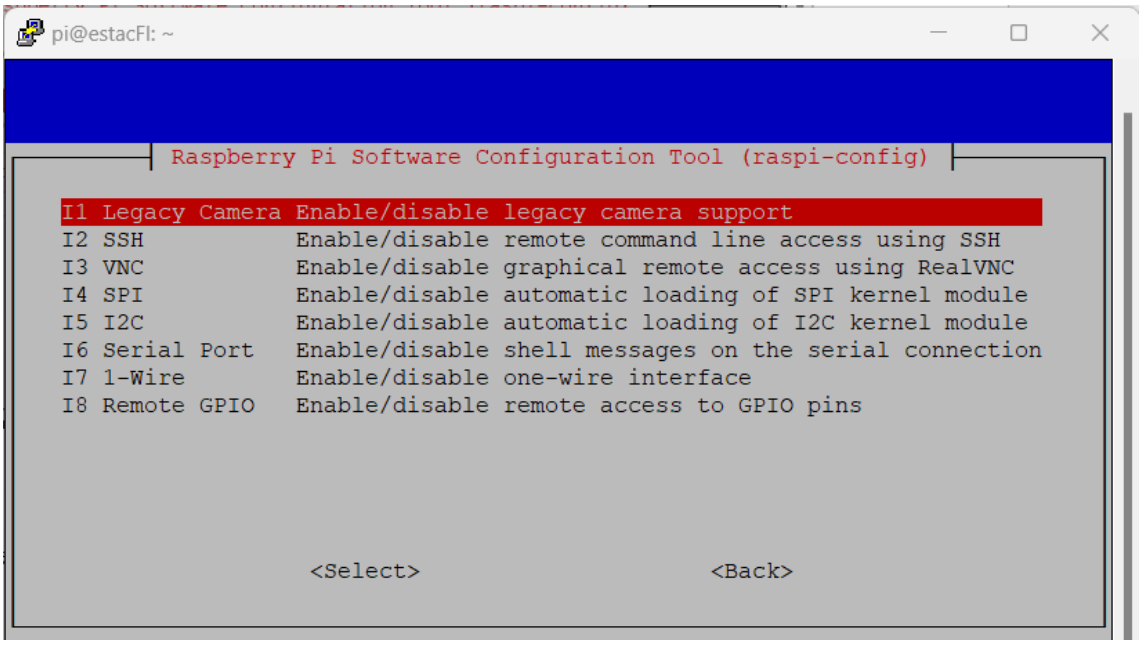

Figura 12.3. *Habilitar cámara*

Final mente aceptamos y salimos a la consola principal.

5. Base de datos

Utilizando el navegador ingresar a phpMyAdmin con la dirección IP + /phpmyadmin/ y tiene que aparecer una ventana como se muestra en la Figura 12.1. Se iniciará sesión con el usuario “root” y la contraseña que se utilizó para crear la base de datos que en este caso es “holamundo”.

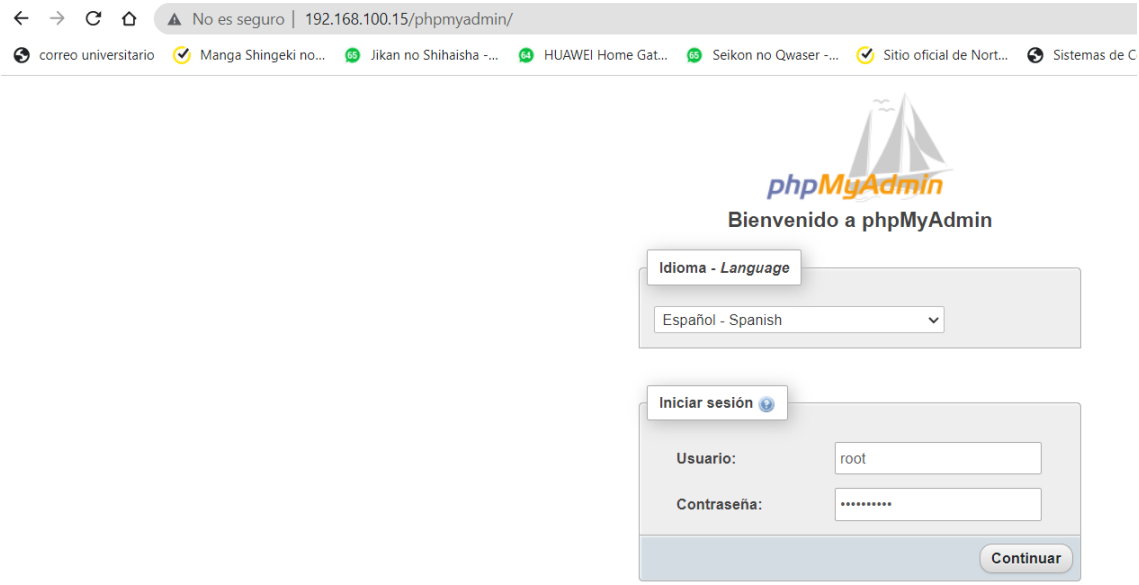

Figura 13.1. *Crear base de datos*

A continuación, se creará la base de datos dando clic a la opción que dice “Nueva” en la parte superior izquierda de la página que apareció al iniciar sesión, como se muestra en la Figura 13.2., esta opción mostrará la página llamada “Base de datos”, ahí se pondrá el nombre de la nueva base de datos que en este caso se llama “Estacionamiento” y se selecciona la opción “utf8_spanish2_ci” y se oprime en crear.

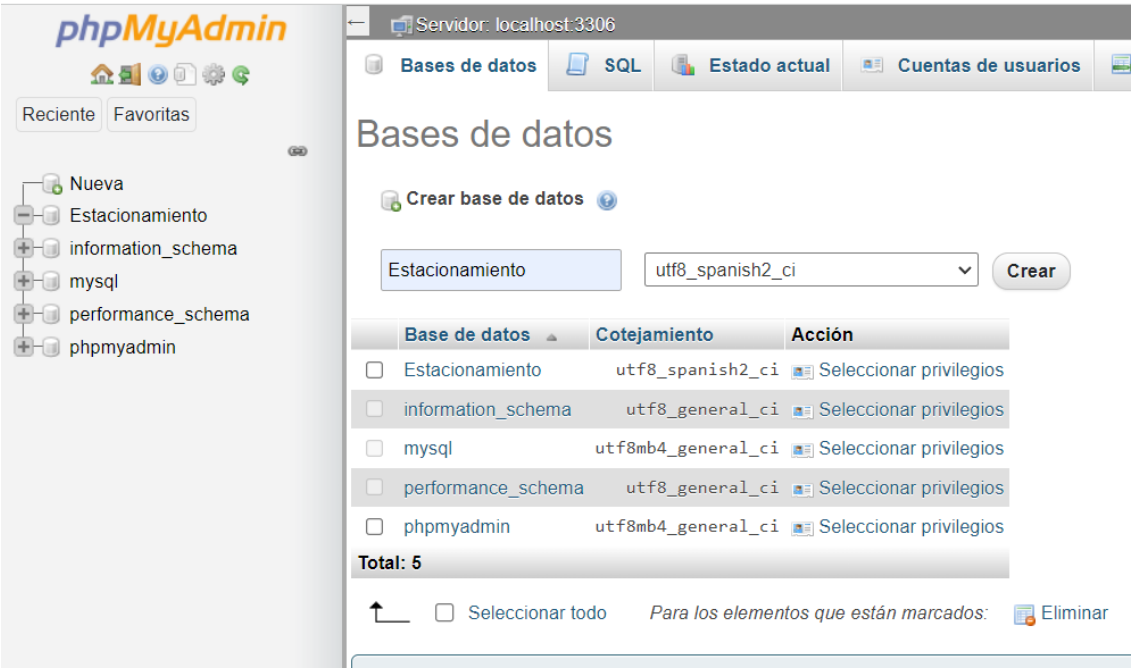

Figura 13.2. *Crear base de datos*

Al dar clic en crear aparecerá una página en la cual se crearán las tablas que existirán en la base de datos, en este caso se crearán dos tablas, la primera se llamara “usuarios” y la segunda “registros”. Como se muestra en la Figura 13.3. donde se crea la tabla llamada “usuario” con 5 columnas y se da clic en “continuar”

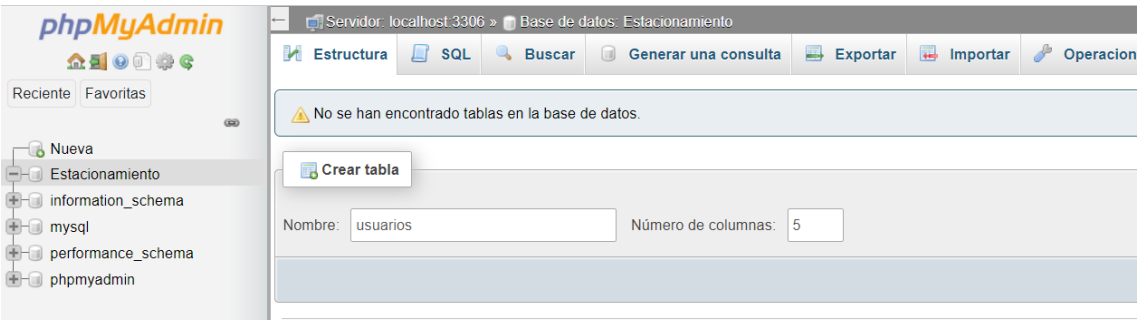

Figura 13.3. *Crear base de datos*

Al dar clic en continuar aparecerá una tabla y se llenará como se muestra en la Figura 13.4. Y en la parte inferior derecha dar clic en “Guardar”

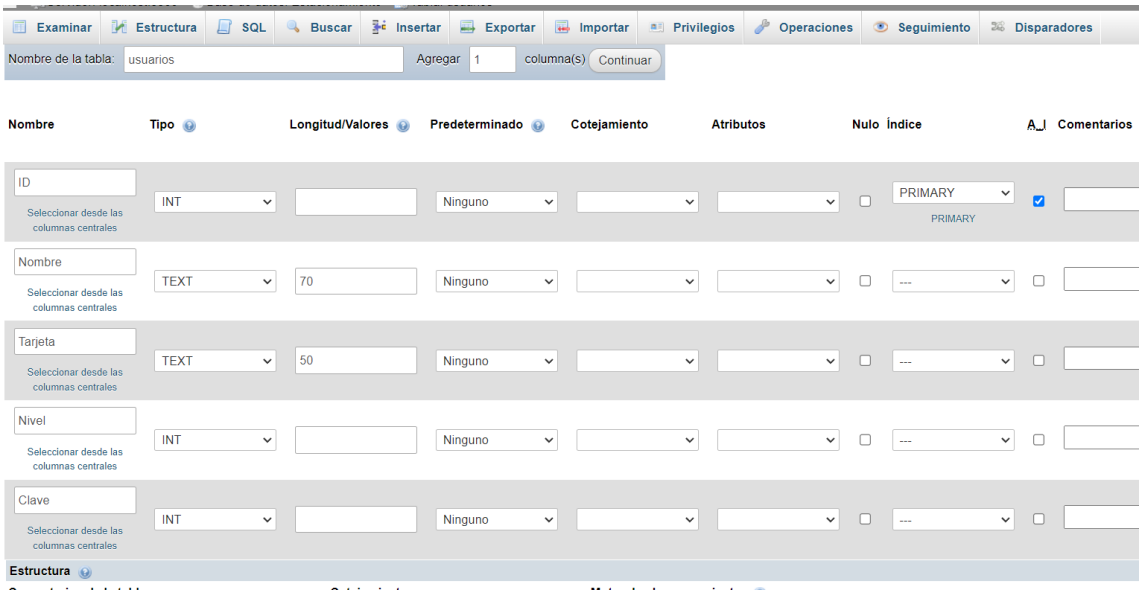

Figura 13.4. *Crear base de datos*

Se repite el mismo procedimiento para crear una table con el nombre de “registros” y la tabla se llena como se muestra en la Figura 13.5.

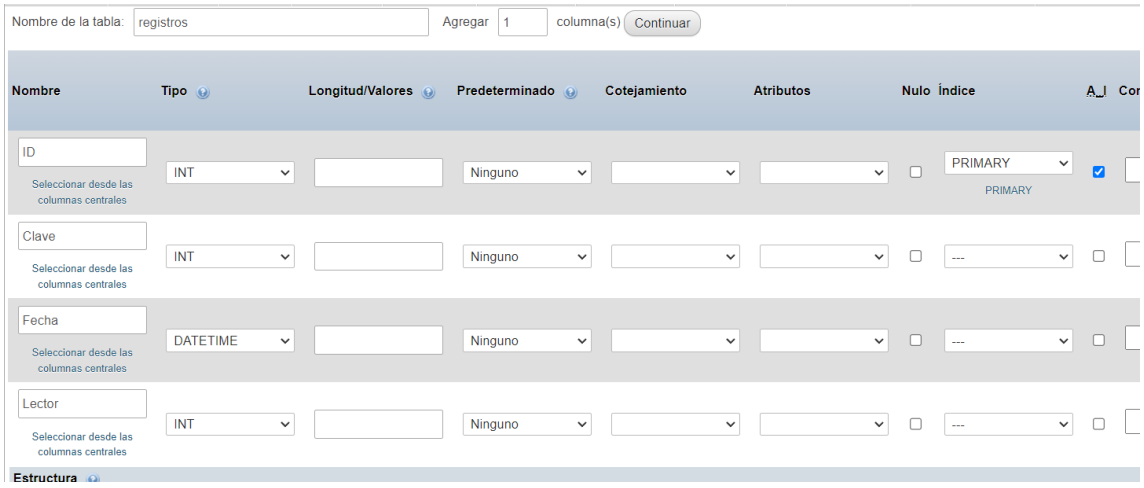

Figura 13.5. *Crear base de datos*

De esta forma queda la base de datos por parte de phpMyAdmin.

Para poder crear la conexión entre MSQL y Python se debe ingresar el comando **`python -m pip install mysql-connector-python`** como se muestra en la Figura 14.1.

Figura 14.1. *Librerías extras para base de datos*

Tambien la librería scanf que se busco en piwheels **`sudo pip3 install scanf`** como se muestra en la Figura 14.2.

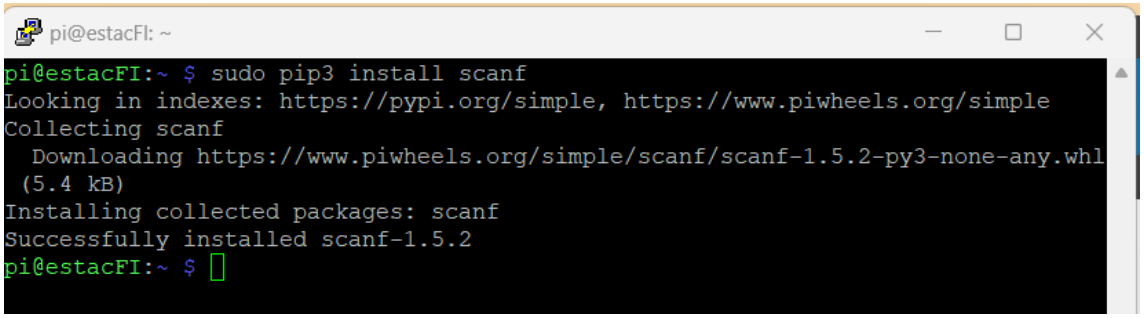

Figura 14.2. *Librerías extras para base de datos*

6. Códigos de línea

El siguiente código con el nombre “MFRC522.py” se utiliza para hacer la decodificación del lector RFID. Y se puede encontrar por internet.

```
#MFRC522.py
import spidev
import signal
import time

DEBUG = False

class MFRC522:
  MAX_LEN = 16

  PCD_IDLE       = 0x00
  PCD_AUTHENT    = 0x0E
  PCD_RECEIVE    = 0x08
  PCD_TRANSMIT   = 0x04
  PCD_TRANSCEIVE = 0x0C
  PCD_RESETPHASE = 0x0F
  PCD_CALCCRC    = 0x03

  PICC_REQIDL    = 0x26
  PICC_REQALL    = 0x52
  PICC_ANTICOLL1  = 0x93
  PICC_ANTICOLL2  = 0x95
  PICC_ANTICOLL3  = 0x97
  PICC_AUTHENT1A = 0x60
  PICC_AUTHENT1B = 0x61
  PICC_READ      = 0x30
  PICC_WRITE     = 0xA0
  PICC_DECREMENT = 0xC0
  PICC_INCREMENT = 0xC1
  PICC_RESTORE   = 0xC2
  PICC_TRANSFER  = 0xB0
  PICC_HALT      = 0x50

  MI_OK       = 0
  MI_NOTAGERR = 1
  MI_ERR      = 2

  Reserved00     = 0x00
  CommandReg     = 0x01
  CommIEnReg     = 0x02
  DivlEnReg      = 0x03
  CommIrqReg     = 0x04
  DivIrqReg      = 0x05
  ErrorReg       = 0x06
  Status1Reg     = 0x07
  Status2Reg     = 0x08
  FIFODataReg    = 0x09
  FIFOLevelReg   = 0x0A
  WaterLevelReg  = 0x0B
  ControlReg     = 0x0C
  BitFramingReg  = 0x0D
  CollReg        = 0x0E
  Reserved01     = 0x0F

  Reserved10     = 0x10
  ModeReg        = 0x11
  TxModeReg      = 0x12
  RxModeReg      = 0x13
  TxControlReg   = 0x14
  TxAutoReg      = 0x15
  TxSelReg       = 0x16
  RxSelReg       = 0x17
  RxThresholdReg = 0x18
  DemodReg       = 0x19
  Reserved11     = 0x1A
  Reserved12     = 0x1B
```

```
    MifareReg      = 0x1C
    Reserved13     = 0x1D
    Reserved14     = 0x1E
    SerialSpeedReg = 0x1F

    Reserved20        = 0x20
    CRCResultRegM     = 0x21
    CRCResultRegL     = 0x22
    Reserved21        = 0x23
    ModWidthReg       = 0x24
    Reserved22        = 0x25
    RFCfgReg          = 0x26
    GsNReg            = 0x27
    CWGsPReg          = 0x28
    ModGsPReg         = 0x29
    TModeReg          = 0x2A
    TPrescalerReg     = 0x2B
    TReloadRegH       = 0x2C
    TReloadRegL       = 0x2D
    TCounterValueRegH = 0x2E
    TCounterValueRegL = 0x2F

    Reserved30      = 0x30
    TestSel1Reg     = 0x31
    TestSel2Reg     = 0x32
    TestPinEnReg    = 0x33
    TestPinValueReg = 0x34
    TestBusReg      = 0x35
    AutoTestReg     = 0x36
    VersionReg      = 0x37
    AnalogTestReg   = 0x38
    TestDAC1Reg     = 0x39
    TestDAC2Reg     = 0x3A
    TestADCReg      = 0x3B
    Reserved31      = 0x3C
    Reserved32      = 0x3D
    Reserved33      = 0x3E
    Reserved34      = 0x3F

    serNum = []

    def __init__(self, bus=0,dev=0, spd=1000000):
      self.spi=spidev.SpiDev()
      self.spi.open(bus=bus,device=dev)
      self.spi.max_speed_hz=spd
      self.MFRC522_Init()

    def MFRC522_Reset(self):
      self.Write_MFRC522(self.CommandReg, self.PCD_RESETPHASE)

    def Write_MFRC522(self, addr, val):
      self.spi.writebytes(((addr<<1)&0x7E,val))

```

```
120   def Read_MFRC522(self, addr):
121     val = self.spi.xfer2((((addr<<1)&0x7E) | 0x80,0))
122     return val[1]
123
124   def SetBitMask(self, reg, mask):
125     tmp = self.Read_MFRC522(reg)
126     self.Write_MFRC522(reg, tmp | mask)
127
128   def ClearBitMask(self, reg, mask):
129     tmp = self.Read_MFRC522(reg);
130     self.Write_MFRC522(reg, tmp & (~mask))
131
132   def AntennaOn(self):
133     temp = self.Read_MFRC522(self.TxControlReg)
134     if(~(temp & 0x03)):
135       self.SetBitMask(self.TxControlReg, 0x03)
136
137   def AntennaOff(self):
138     self.ClearBitMask(self.TxControlReg, 0x03)
139
140   def MFRC522_ToCard(self,command,sendData):
141     backData = []
142     backLen = 0
143     status = self.MI_ERR
144     irqEn = 0x00
145     waitIRq = 0x00
146     lastBits = None
147     n = 0
148     i = 0
149
150     if command == self.PCD_AUTHENT:
151       irqEn = 0x12
152       waitIRq = 0x10
153     if command == self.PCD_TRANSCEIVE:
154       irqEn = 0x77
155       waitIRq = 0x30
156
157     self.Write_MFRC522(self.CommIEnReg, irqEn|0x80)
158     self.ClearBitMask(self.CommIrqReg, 0x80)
159     self.SetBitMask(self.FIFOLevelReg, 0x80)
160
161     self.Write_MFRC522(self.CommandReg, self.PCD_IDLE);
162
163     while(i<len(sendData)):
164       self.Write_MFRC522(self.FIFODataReg, sendData[i])
165       i = i+1
166
167     self.Write_MFRC522(self.CommandReg, command)
168
169     if command == self.PCD_TRANSCEIVE:
170       self.SetBitMask(self.BitFramingReg, 0x80)
171
```

```
    i = 2000
    while True:
      n = self.Read_MFRC522(self.CommIrqReg)
      i = i - 1
      if ~((i!=0) and ~(n&0x01) and ~(n&waitIRq)):
        break

    self.ClearBitMask(self.BitFramingReg, 0x80)

    if i != 0:
      if (self.Read_MFRC522(self.ErrorReg) & 0x1B)==0x00:
        status = self.MI_OK

        if n & irqEn & 0x01:
          status = self.MI_NOTAGERR

        if command == self.PCD_TRANSCEIVE:
          n = self.Read_MFRC522(self.FIFOLevelReg)
          lastBits = self.Read_MFRC522(self.ControlReg) & 0x07
          if lastBits != 0:
            backLen = (n-1)*8 + lastBits
          else:
            backLen = n*8

          if n == 0:
            n = 1
          if n > self.MAX_LEN:
            n = self.MAX_LEN

          i = 0
          while i<n:
            backData.append(self.Read_MFRC522(self.FIFODataReg))
            i = i + 1;
      else:
        status = self.MI_ERR

    return (status,backData,backLen)


  def MFRC522_Request(self, reqMode):
    status = None
    backBits = None
    TagType = []

    self.Write_MFRC522(self.BitFramingReg, 0x07)

    TagType.append(reqMode);
    (status,backData,backBits) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE, TagType)

    if ((status != self.MI_OK) | (backBits != 0x10)):
      status = self.MI_ERR
```

```
223
224     return (status,backBits)
225
226
227   def MFRC522_Anticoll(self,anticolN):
228     backData = []
229     serNumCheck = 0
230
231     serNum = []
232
233     self.Write_MFRC522(self.BitFramingReg, 0x00)
234
235     serNum.append(anticolN)
236     serNum.append(0x20)
237
238     (status,backData,backBits) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE,serNum)
239
240     if(status == self.MI_OK):
241       i = 0
242       if len(backData)==5:
243         while i<4:
244           serNumCheck = serNumCheck ^ backData[i]
245           i = i + 1
246         if serNumCheck != backData[i]:
247           status = self.MI_ERR
248       else:
249         status = self.MI_ERR
250
251     return (status,backData)
252
253   def MFRC522_Anticoll1(self):
254     return  self.MFRC522_Anticoll(self.PICC_ANTICOLL1)
255
256   def MFRC522_Anticoll2(self):
257     return  self.MFRC522_Anticoll(self.PICC_ANTICOLL2)
258
259   def MFRC522_Anticoll3(self):
260     return  self.MFRC522_Anticoll(self.PICC_ANTICOLL3)
261
262
263   def CalulateCRC(self, pIndata):
264     self.ClearBitMask(self.DivIrqReg, 0x04)
265     self.SetBitMask(self.FIFOLevelReg, 0x80);
266     i = 0
267     while i<len(pIndata):
268       self.Write_MFRC522(self.FIFODataReg, pIndata[i])
269       i = i + 1
270     self.Write_MFRC522(self.CommandReg, self.PCD_CALCCRC)
271     i = 0xFF
272     while True:
273       n = self.Read_MFRC522(self.DivIrqReg)
274       i = i - 1
275       if not ((i != 0) and not (n&0x04)):
276         break
277     pOutData = []
278     pOutData.append(self.Read_MFRC522(self.CRCResultRegL))
279     pOutData.append(self.Read_MFRC522(self.CRCResultRegM))
280     return pOutData
```

```
281
282   def MFRC522_PcdSelect(self, serNum,anticolN):
283     backData = []
284     buf = []
285     buf.append(anticolN)
286     buf.append(0x70)
287     i = 0
288     while i<5:
289       buf.append(serNum[i])
290       i = i + 1
291     pOut = self.CalulateCRC(buf)
292     buf.append(pOut[0])
293     buf.append(pOut[1])
294     (status, backData, backLen) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE, buf)
295     if (status == self.MI_OK) and (backLen == 0x18):
296       if DEBUG:
297         print("Size: " + str(backData[0]))
298         print("PcdSelect {} {}".format(anticolN,backData))
299       return  1
300     else:
301       return 0
302
303   def MFRC522_PcdSelect1(self, serNum):
304     return  self.MFRC522_PcdSelect(serNum,self.PICC_ANTICOLL1)
305
306   def MFRC522_PcdSelect2(self, serNum):
307     return  self.MFRC522_PcdSelect(serNum,self.PICC_ANTICOLL2)
308
309   def MFRC522_PcdSelect3(self, serNum):
310     return  self.MFRC522_PcdSelect(serNum,self.PICC_ANTICOLL3)
311
312
313
314
315   def MFRC522_Auth(self, authMode, BlockAddr, Sectorkey, serNum):
316     buff = []
317
318     # First byte should be the authMode (A or B)
319     buff.append(authMode)
320
321     # Second byte is the trailerBlock (usually 7)
322     buff.append(BlockAddr)
323
324     # Now we need to append the authKey which usually is 6 bytes of 0xFF
325     i = 0
326     while(i < len(Sectorkey)):
327       buff.append(Sectorkey[i])
328       i = i + 1
329     i = 0
330
331     # Next we append the first 4 bytes of the UID
332     while(i < 4):
333       buff.append(serNum[i])
334       i = i +1
335
336     # Now we start the authentication itself
```

```
337     (status, backData, backLen) = self.MFRC522_ToCard(self.PCD_AUTHENT,buff)
338 
339     # Check if an error occurred
340 
341 
342     if DEBUG:
343       if not(status == self.MI_OK):
344         print("AUTH ERROR!!")
345       if not (self.Read_MFRC522(self.Status2Reg) & 0x08) != 0:
346         print("AUTH ERROR(status2reg & 0x08) != 0")
347 
348     # Return the status
349     return status
350 
351   def MFRC522_StopCrypto1(self):
352     self.ClearBitMask(self.Status2Reg, 0x08)
353 
354   def MFRC522_Read(self, blockAddr):
355     recvData = []
356     recvData.append(self.PICC_READ)
357     recvData.append(blockAddr)
358     pOut = self.CalulateCRC(recvData)
359     recvData.append(pOut[0])
360     recvData.append(pOut[1])
361     (status, backData, backLen) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE, recvData)
362     if not(status == self.MI_OK):
363       print("Error while reading!")
364     i = 0
365     if len(backData) == 16:
366       print("Sector "+str(blockAddr)+" "+str(backData))
367 
368   def MFRC522_Write(self, blockAddr, writeData):
369     buff = []
370     buff.append(self.PICC_WRITE)
371     buff.append(blockAddr)
372     crc = self.CalulateCRC(buff)
373     buff.append(crc[0])
374     buff.append(crc[1])
375     (status, backData, backLen) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE, buff)
376     if not(status == self.MI_OK) or not(backLen == 4) or not((backData[0] & 0x0F) ==
0x0A):
377         status = self.MI_ERR
378 
379     print("%s backdata &0x0F == 0x0A %s" % (backLen, backData[0]&0x0F))
380     if status == self.MI_OK:
381         i = 0
382         buf = []
383         while i < 16:
384             buf.append(writeData[i])
385             i = i + 1
386         crc = self.CalulateCRC(buf)
387         buf.append(crc[0])
388         buf.append(crc[1])
389         (status, backData, backLen) = self.MFRC522_ToCard(self.PCD_TRANSCEIVE,buf)
390         if not(status == self.MI_OK) or not(backLen == 4) or not((backData[0] & 0x0F) ==
0x0A):
```

```
391             print("Error while writing")
392         if status == self.MI_OK:
393             print("Data written")
394 
395     def MFRC522_DumpClassic1K(self, key, uid):
396       i = 0
397       while i < 64:
398           status = self.MFRC522_Auth(self.PICC_AUTHENT1A, i, key, uid)
399           # Check if authenticated
400           if status == self.MI_OK:
401               self.MFRC522_Read(i)
402           else:
403               print("Authentication error")
404           i = i+1
405 
406     def MFRC522_Init(self):
407 
408       self.MFRC522_Reset();
409 
410 
411       self.Write_MFRC522(self.TModeReg, 0x8D)
412       self.Write_MFRC522(self.TPrescalerReg, 0x3E)
413       self.Write_MFRC522(self.TReloadRegL, 30)
414       self.Write_MFRC522(self.TReloadRegH, 0)
415 
416       self.Write_MFRC522(self.TxAutoReg, 0x40)
417       self.Write_MFRC522(self.ModeReg, 0x3D)
418       self.AntennaOn()
419 
420 
421     def MFRC522_SelectTagSN(self):
422         valid_uid=[]
423         (status,uid)= self.MFRC522_Anticoll1()
424         if status != self.MI_OK:
425              return  (self.MI_ERR,[])
426 
427         if DEBUG:   print("anticol1() {}".format(uid))
428         if self.MFRC522_PcdSelect1(uid) == 0:
429             return (self.MI_ERR,[])
430         if DEBUG:   print("pcdSelect1() {}".format(uid))
431 
432         #check if first byte is 0x88
433         if uid[0] == 0x88 :
434           #ok we have another type of card
435            valid_uid.extend(uid[1:4])
436            (status,uid)=self.MFRC522_Anticoll2()
437            if status != self.MI_OK:
438              return (self.MI_ERR,[])
439            if DEBUG: print("Anticol2() {}".format(uid))
440            rtn =  self.MFRC522_PcdSelect2(uid)
441            if DEBUG: print("pcdSelect2 return={} uid={}".format(rtn,uid))
442            if rtn == 0:
443              return (self.MI_ERR,[])
444            if DEBUG: print("PcdSelect2() {}".format(uid))
445            #now check again if uid[0] is 0x88
446            if uid[0] == 0x88 :
447              valid_uid.extend(uid[1:4])
448              (status , uid) = self.MFRC522_Anticoll3()
449              if status != self.MI_OK:
450                return (self.MI_ERR,[])
451              if DEBUG: print("Anticol3() {}".format(uid))
452              if self.MFRC522_PcdSelect3(uid) == 0:
453                return (self.MI_ERR,[])
454              if DEBUG: print("PcdSelect3() {}".format(uid))
455         valid_uid.extend(uid[0:4])
456 
457         return (self.MI_OK,valid_uid)
```

El siguiente código con el nombre “RFID.py” se utiliza para obtener el código en una variable. Este código manda a llamar al programa “MFRC522.py”

```python
#RFID.py
import RPi.GPIO as GPIO
import MFRC522
import time

class RFID:

  def __init__(self):

    self.__reader = MFRC522.MFRC522()

  def uidToString(self, uid):

    stringUID = ""

    for i in uid:
      stringUID = format(i, '02X') + stringUID

    return stringUID

  def read(self,rst):

    leyendo = True
    GPIO.setup(rst,GPIO.OUT)
    GPIO.output(rst,False)

    conTiempo = True

    tiempoInicial = time.time()

    while leyendo and conTiempo:

      tiempoActual = time.time()

      if ( abs(tiempoActual - tiempoInicial)>=30.0):

        conTiempo = False
        print("Se ha acabado el tiempo")
        GPIO.output(rst,True)

      else:

        (status, TagType) = self.__reader.MFRC522_Request(self.__reader.PICC_REQIDL)

        if status == self.__reader.MI_OK:

          (status, uid) = self.__reader.MFRC522_SelectTagSN()

          if status == self.__reader.MI_OK:
            self.uid = uid
            print("UID de la tarjeta: %s" % self.uidToString(uid))

            leyendo = False
            GPIO.output(rst,True)

          else:
            print("Error de autenticacion")

```

El siguiente código con el nombre “addusuario.py” se utiliza para agregar nuevos usuarios a la base de datos. Este código manda a llamar al programa “RFID.py”

```python
#addusuario.py
import RPi.GPIO as GPIO
from datetime import datetime
import mysql.connector
import subprocess
import time
from RFID import RFID

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

buttonEntry = 37
GPIO.setup(37, GPIO.IN, GPIO.PUD_DOWN)
buttonExit = 18
GPIO.setup(18, GPIO.IN, GPIO.PUD_DOWN)

pino = 16
GPIO.setup(16,GPIO.OUT)
pint = 22
GPIO.setup(22,GPIO.OUT)

def leer(pino):
    lector = RFID()
    lector.read(pino)
    return lector.uidToString(lector.uid)

def leeer(pint):
    lector = RFID()
    lector.read(pint)
    return lector.uidToString(lector.uid)

mydb = mysql.connector.connect(
    host='localhost',
    user='root',
    password='holamundo',
    database='Estacionamiento'
)

mycursor = mydb.cursor()

button1 = GPIO.input(buttonEntry)
button2 = GPIO.input(buttonExit)

if button1 == True:
    print("Nuevo usuario para el estacionamiento")
    nombre = input("Nombre:")
    nivel = input("Nivel de acceso. 1.-maestro, 2.-alumno, 3.-visitante):")
    clave = input("Numero de Cuenta:")
    tarjeta = leer(pino)
    nivel = int(nivel)
    clave = int(clave)
    sql = "INSERT INTO usuarios (Nombre, Tarjeta, Nivel, Clave) VALUES (%s, %s, %s, %s)"
    val = (nombre, tarjeta, nivel, clave)
    mycursor.execute(sql, val)
    mydb.commit()
    print(mycursor.rowcount, "registro insertado.")
```

```
57
58  elif button2 == True:
59      print("Nuevo usuario para el estacionamiento")
60      nombre = input("Nombre:")
61      nivel = input("Nivel de acceso. 1.-maestro, 2.-alumno, 3.-visitante):")
62      clave = input("Numero de Cuenta:")
63      tarjeta = leeer(pint)
64      nivel = int(nivel)
65      clave = int(clave)
66      sql = "INSERT INTO usuarios (Nombre, Tarjeta, Nivel, Clave) VALUES (%s, %s, %s, %s)"
67      val = (nombre, tarjeta, nivel, clave)
68      mycursor.execute(sql, val)
69      mydb.commit()
70      print(mycursor.rowcount, "registro insertado.")
71
72  else:
73      print("no hay coche detectado")
74
```

El siguiente código con el nombre “Main.py” este se encarga de todo el funcionamiento del sistema. Este código manda a llamar al programa “RFID.py”

```
1   #Main.py
2   import RPi.GPIO as GPIO
3   import time
4   import subprocess
5   from datetime import datetime
6   import mysql.connector
7   from RFID import RFID
8   from picamera import PiCamera
9
10  GPIO.setmode(GPIO.BOARD)
11  GPIO.setwarnings(False)
12
13  buttonEntry = 37
14  GPIO.setup(37, GPIO.IN, GPIO.PUD_DOWN)
15  buttonExit = 18
16  GPIO.setup(18, GPIO.IN, GPIO.PUD_DOWN)
17
18  pino = 16
19  GPIO.setup(16,GPIO.OUT)
20  pint = 22
21  GPIO.setup(22,GPIO.OUT)
22
23  mydb = mysql.connector.connect(
24      host='localhost',
25      user='root',
26      password='holamundo',
27      database='Estacionamiento'
28  )
29
30  mycursor = mydb.cursor()
31
32  def leer(pino):
33      lector = RFID()
34      lector.read(pino)
35      return lector.uidToString(lector.uid)
36
37  def leeer(pint):
38      lector = RFID()
39      lector.read(pint)
40      return lector.uidToString(lector.uid)
```

```
41
42  def buscar_usuario(Tarjeta):
43      query = ("SELECT Tarjeta FROM usuarios WHERE Tarjeta=%s")
44      mycursor.execute(query,(Tarjeta,))
45      aux = 0
46      result = mycursor.fetchall()
47      for x in result:
48          aux=x[0]
49      return aux
50
51  i=0
52  servo1 = 11
53  GPIO.setup(servo1, GPIO.OUT)
54  p = GPIO.PWM(servo1, 100)
55  p.start(10)
56
57  servo2 = 13
58  GPIO.setup(servo2, GPIO.OUT)
59  q = GPIO.PWM(servo2, 100)
60  q.start(10)
61
62  while True:
63
64      button1 = GPIO.input(buttonEntry)
65      button2 = GPIO.input(buttonExit)
66
67      if button1 == True:
68          p.ChangeDutyCycle(6.5)
69          i+=1
70          print("Acceso para Entrada")
71          tarjeta = leer(pino)
72          subprocess.call(['raspistill','-o','/home/pi/Pictures/imag%s.jpg' % i])
73          a = buscar_usuario(tarjeta)
74          if tarjeta == a:
75              print("Tarjeta encontrada")
76              q.ChangeDutyCycle(3.5)
77              direccion = 1
78              sql = "INSERT INTO registros (Tarjeta, Lector, Fecha) VALUES (%s, %s,NOW())"
79              val = (tarjeta, direccion)
80              mycursor.execute(sql, val)
81              mydb.commit()
82              print(mycursor.rowcount, "registro insertado.... PASE!")
83              time.sleep(10)
84              q.ChangeDutyCycle(10)
85          else:
86              print("Acceso denegado")
87          time.sleep(3)
88      elif button2 == True:
89          p.ChangeDutyCycle(14.5)
90          i+=1
91          print("Acceso para Salida")
92          tarjeta = leeer(pint)
93          subprocess.call(['raspistill','-o','/home/pi/Pictures/imag%s.jpg' % i])
94          b = buscar_usuario(tarjeta)
95          if tarjeta == b:
96              print("Tarjeta encontrada")
```

```
            q.ChangeDutyCycle(3.5)
            direccion = 2
            sql = "INSERT INTO registros (Tarjeta, Lector, Fecha) VALUES (%s, %s, NOW())"
            val = (tarjeta, direccion)
            mycursor.execute(sql, val)
            mydb.commit()
            print(mycursor.rowcount, "registro insertado.... PASE!")
            time.sleep(10)
            q.ChangeDutyCycle(10)
        else:
            print("Acceso denegado")
        time.sleep(3)
    else:
        p.ChangeDutyCycle(10)
        print("Esperando Acceso...")
        time.sleep(0.2)

```