ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE MÓDULOS PARA PROCESAMIENTO DE VOZ, IMAGEN Y DATOS UTILIZANDO RASPBERRY PI PARA EL LABORATORIO DE MICROPROCESADORES DE LA ESFOT

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

MIGUEL ANGEL CAMUÉS BUITRÓN
angelitsmcb_1994@hotmail.com

ADRIÁN ALEJANDRO CISNEROS VELÁSQUEZ
adrian.cisneros@epn.edu.ec

DIRECTOR: ING. FANNY FLORES, MSc.
fanny.flores@epn.edu.ec

CODIRECTOR: ING. MONICA VINUEZA, MSc.
monica.vinueza@epn.edu.ec

Quito, Junio 2018

DECLARACIÓN

Nosotros, CAMUÉS BUITRÓN MIGUEL ANGEL y CISNEROS VELÁSQUEZ ADRIÁN ALEJANDRO, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Camués Buitrón Miguel Angel

Cisneros Velásquez Adrián A.

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por CAMUÉS BUITRÓN MIGUEL ANGEL y CISNEROS VELÁSQUEZ ADRIÁN ALEJANDRO, bajo nuestra supervisión.

Ing. Fanny Flores Estévez, MSc.	Ing. Mónica Vinueza Rhor, MSc.
DIRECTORA DE PROYECTO	**CODIRECTORA DE PROYECTO**

DEDICATORIA

A todas aquellas personas importantes en el transcurso de mi vida que siempre estuvieron ahí en las buenas y en las malas, cuya confianza me fortalecía, permitiéndome seguir sin importar las dificultades que se presentaran en el camino.

"Adrián"

A mis padres Miguel y Cecilia, quienes me apoyaron incondicionalmente en cada momento de mi carrera estudiantil, por los valores que me han inculcado, y sus consejos que me han ayudado a salir adelante ante cualquier adversidad. Y sobre todo por ser un ejemplo de lucha y constancia ante la vida.

A mi hermano Carlos por compartir todos sus conocimientos cuando los necesité, sus anécdotas y experiencias que me ayudaron a no cometer muchos errores.

A mis hermanas Margarita y Angélica, por darme todo su cariño y estar siempre con una sonrisa a mi lado en los buenos y malos momentos.

"Miguel"

AGRADECIMIENTOS

A mi familia, mis amigos y todas las personas que me brindaron el apoyo necesario para seguir avanzando, inspirándome para poder realizar este trabajo y cumplir con el objetivo de llegar más lejos.

"Adrián"

Agradezco a toda mi familia porque de una u otra manera estuvieron siempre apoyándome, guiándome y viendo mi completo bienestar personal y en mi carrera.

A Angélica Banda quien llegó en momentos difíciles de mi vida y con su amor logró darme las fuerzas para terminar mis estudios, pasando malas noches a mi lado, ayudándome académica y personalmente.

A mis amigos Miguel Banda, Jonathan Martínez, Byron Tipanluisa y Ronie Rosillo con quienes compartimos muchas vivencias y entre risas, juegos y lágrimas formamos una fuerte amistad que hizo de mi etapa estudiantil una experiencia única.

A mi directora de tesis, Ing. Fanny Flores quien me brindó su confianza, tiempo, paciencia y consideración para poder lograr una meta más en mi vida.

A mi codirectora de tesis, Ing. Mónica Vinueza por compartir sus conocimientos, por sus palabras de motivación que me ayudaron a ser un buen profesional.

A todos quienes hicieron posible la realización de este proyecto y personas que talvez paso por alto, pero les debo mis agradecimientos.

"Miguel"

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La finalidad del presente proyecto es la implementación de nueve módulos basados en tarjetas Raspberry Pi 3, que sean capaces de cumplir funciones de procesamiento de imágenes, voz y bases de datos. Los códigos y el funcionamiento de los programas serán controlados y gestionados por el usuario final. Dichos módulos serán utilizados por los estudiantes del Laboratorio de Microprocesadores.

Los fundamentos teóricos para este proyecto se los detalla de manera rápida y legible para el lector con una descripción breve de los dispositivos que forman parte del módulo implementado; así mismo, del software utilizado para el funcionamiento de los programas. El contenido consta de resúmenes, gráficos detallados de las conexiones, configuraciones, programación y funcionamiento de los módulos que hacen de este documento una fuente de consulta e introducción a la programación de módulos Raspberry Pi 3 con Python 2.

El módulo está conformado principalmente por el microcomputador Raspberry Pi 3, que es la base para el procesamiento de la información recopilada por dispositivos externos. Además, cuenta con dispositivos adicionales como son: micrófono USB para la captación de los comandos de voz, cámara compatible con el módulo Raspberry Pi 3 que se encargará de receptar las imágenes para su procesamiento y una pequeña placa con dos LEDs, uno de color azul y otro de color verde conectados a los pines GPIO, en los cuales se visualizarán su encendido o apagado controlados por comandos de voz.

Los programas son ejecutados desde la terminal de comandos, introduciendo la ruta de ubicación del archivo para abrirlo. Para el funcionamiento del procesamiento de imágenes y voz, basta con los archivos creados en Python. Por otro lado, el procesamiento de base de datos hace uso de software complementario como: Apache, MySQL y PhPMyAdmin.

Los registros completos de las tres prácticas estarán en informes individuales en forma de manual de usuario, en los cuales se indican los comandos básicos de Linux para la actualización de software y ejecución de los programas

ABSTRACT

The objective of this project is the deployment of nine modules based on Raspberry Pi 3. The modules are capable of processing images, voice and databases. The codes of the programs and operations will be controlled and managed by final users. The modules will be used by the Microprocessors Laboratory students.

The theoretical basics for this project are detailed in a fast and understandable mode to the reader. Moreover, a brief description of devices that take part of the implemented module is included; as well as the software for the program's operation. The contents consist of summaries, connection diagrams, settings, programming, and module's operation; making this document a Raspberry Pi 3 with Python 2 searching and learning source.

The module is mainly made up of the Raspberry Pi 3 microcomputer, which is the base for acquired information processing from external devices. Also, it has additional devices such as: a USB microphone to receive voice commands, a dedicated Raspberry Pi 3 camera to capture images for processing, and an electronic circuit with two blue and green LEDs connected to GPIO pins to show the ON/OFF status of them controlled by voice commands.

Programs are executed from the terminal, by inserting the file's location to open it. For images and voice processing, it is enough with creating files in Python. Instead, for database processing, it is necessary to use complementary software like Apache, MySQL and PhPMyAdmin.

The complete registry of the three practices is included in individual reports as user guides, featuring the Linux basic commands for software update and programs running.

1. INTRODUCCIÓN

El PIC 16F870 constituye el elemento base de la asignatura de Microprocesadores dictada actualmente en la Escuela de Formación de Tecnólogos. Dicho dispositivo ha sido considerado por su versátil aplicación en el ámbito electrónico [1]. Sin embargo, al ser un elemento electrónico de limitadas características, hoy en día se encuentra obsoleto en comparación con otros microcontroladores y módulos electrónicos que ofrecen funciones más avanzadas y modernas [1].

La desactualización del Programa de Estudios por Asignatura (PEA) de la asignatura de Microprocesadores, hace que la enseñanza se base en el uso del lenguaje de programación Ensamblador o “Assembler”. El desarrollo de programas en bajo nivel resulta altamente complejo para los estudiantes, puesto que requiere muchas líneas de código para realizar incluso un programa sencillo; esto no ocurriría al usar un lenguaje de alto nivel, como lenguaje de programación en C.

Assembler presenta otras desventajas; como por ejemplo: demanda un considerable tiempo de programación, los comandos son complejos de utilizar, el software de programación no es didáctico ni muy amigable con el programador, se debe tener una secuencia muy estricta de las instrucciones, una programación inadecuada pudiera bloquear o dañar el PIC. Adicionalmente, el conocimiento del programador no debe limitarse a la programación; además se debe conocer el funcionamiento interno del microcontrolador, lo cual resulta complicado debido al tiempo limitado de la asignatura de Microprocesadores.

Por otro lado, los estudiantes hacen uso de protoboards para realizar sus prácticas y muchos de estos no funcionan correctamente o fallan en su totalidad. Lo mismo ocurre con dispositivos electrónicos, ocasionando que los estudiantes presenten prácticas incompletas o no las alcancen a presentar.

En base a estos problemas, se busca dar una solución más eficiente, para lo cual se implementarán nueve módulos didácticos basados en tarjetas Raspberry Pi 3, ideales para un aprendizaje más didáctico y contemporáneo.

Este proyecto tiene como objetivo proporcionar material interactivo útil y por sobre todo actualizado al Laboratorio de Microprocesadores de la ESFOT. Este material estará dedicado a los estudiantes que cursen la carrera de Electrónica y Telecomunicaciones, así como Electromecánica, mejorando su aprendizaje e incentivando el interés y la creatividad de los mismos hacia un nuevo ámbito de programación y elaboración de proyectos más innovadores.

Nueve módulos Raspberry Pi estarán a disposición de los estudiantes, los cuales son equipos dirigidos al aprendizaje, enfocados a la programación y electrónica, y que incluyen múltiples recursos con el propósito de facilitar la introducción a la tecnología actual [2].

Los módulos Raspberry Pi son considerados prácticamente mini computadoras. El lenguaje de programación Python es relativamente mucho más sencillo y comprensible, e incluso posee librerías que sirven de apoyo para el programador, simplificando enormemente las líneas de código que disminuirán la complejidad del proyecto [3].

El módulo Raspberry Pi es usado en un sinnúmero de aplicaciones, de acuerdo a las necesidades de las personas. En robótica, se puede construir desde robots sencillos hasta robots con inteligencia artificial; ha revolucionado en el mercado de la domótica ya que tiene la capacidad suficiente de hacer la función de servidor en el cual convergen los servicios de audio, video en tiempo real, automatización de las cosas, bases de datos y seguridades con controles de acceso [4].

1.1 MARCO TEÓRICO

Raspberry PI

Introducción

Raspberry Pi es una fundación encaminada a la aplicación de tecnologías digitales de forma interactiva y educativa, dándoles a las personas la posibilidad de aprender, desenvolverse e incluso crear soluciones de la misma índole por medio del conocimiento impartido y las herramientas que ofrece [5].

La base fundamental del programa que apoya al saber tecnológico radica en el hardware llamado del mismo modo, el cual se puede definir como una pequeña computadora de bajo costo, pero de alto rendimiento en la que se puede aprender temas de actual interés como programación y robótica llevada a diversos campos. [5]

La idea que dio forma a este singular dispositivo data en el 2006 cuando el Dr. Eben Upton y sus colegas en la Universidad de Cambridge, propusieron un modelo computacional de bajo costo que pudiera resolver el declive existente en cuanto al conocimiento y habilidades técnicas de las nuevas generaciones venideras de esa época. Enfocados, principalmente en las jóvenes generaciones. [4]

Hoy en día, la comunidad de Raspberry cuenta con miles de participantes entre jóvenes y adultos en todo el mundo que comparten el interés por la tecnología, la cual sigue creciendo cada vez más.

Características

Los módulos Raspberry Pi precisan de ser dispositivos prácticos a pesar de su sencillez en cuanto a características, además de ser costeables tanto en adquisición como en mantenimiento debido a su bajo consumo de energía. [6]

También al complementarse con software de uso libre, su rentabilidad en cuanto a libertad de investigación y experimentación, lo convierte en una herramienta muy eficiente y al alcance de todos. En la Tabla 1.1 se presenta un cuadro comparativo de las características de Raspberry Pi 2 y Raspberry Pi 3.

Tabla 1.1 Características de hardware de RasPi 2 y RasPi 3. [7]

	Raspberry Pi 2 modelo B	***Raspberry Pi 3 modelo B***
SoC	BCM2836	BCM2837
CPU	Quad Cortex A7 @ 900 MHz	Quad Cortex A53 @ 1.2GHz
Set de Instrucciones	ARMv7-A	ARMv8-A
GPU	250MHz VideoCore IV	400MHz VideoCore IV
RAM	1GB SDRAM	1GB SDRAM
Almacenamiento	micro SD	micro SD
Ethernet	10/100	10/100
USB	4 Puertos	4 Puertos
Inalámbrico	Ninguno	802.11n/Bluetooth 4.0
Salida de Video	HDMI / Compuesto	HDMI / Compuesto
Salida de Audio	HDMI / Auriculares	HDMI / Auriculares
GPIO	40 Pines	40 Pines

Para un mayor detalle técnico de la estructura y composición base de las tarjetas Raspberry, diríjase al *Anexo A1.*

Hardware

Procesador

Como se muestra en la Figura 1.1, el módulo Raspberry Pi 3 posee un microprocesador Broadcom BCM2837 de 64 bits, siendo éste la versión más actual y mejorada comparada con su antecesor BCM2836. La diferencia radica solamente en el reemplazo de la arquitectura ARMv7 por una arquitectura de mejores prestaciones (ARMv8), cuyos núcleos trabajan a una frecuencia de 1.2Ghz, haciendo a Raspberry Pi 3 un 50% más rápido que su versión anterior. Para más información sobre la información de la arquitectura ARMv8 diríjase al Anexo A2.

Figura 1.1 Tarjeta Raspberry Pi 3 modelo B. [8]

El GPU incluido (Graphics Processing Unit), trabaja a una frecuencia de 400Mhz. Debido a que el microchip gestiona múltiples funciones dentro de sí, se lo considera un SoC (*System on a Chip*) [7] [9]. Para un mayor detalle sobre la información del procesador y subpropiedades complementarias diríjase al *Anexo A3.*

Memoria

Las memorias utilizadas por la tarjeta Raspberry Pi 3 consisten en dos partes. La memoria RAM del tipo SDRAM incluida en el circuito del dispositivo con capacidad de 1GB, la cual se encarga de entregar un rendimiento eficiente en la ejecución de los programas. La memoria SD externa al sistema, la cual cumple la función de

disco duro que además de sostener el sistema operativo, será el medio de almacenamiento masivo de información en ésta. [7] [4]

Tarjeta de red

Incluye en su placa el módulo de red Broadcom BCM43438 con disponibilidad de conexión WiFi bajo el estándar 802.11n para velocidades de transmisión cercanas a 600 Mbps. Además, soporta Bluetooth 4.1 de bajo consumo (*Bluetooth Low Energy*), que permite una tasa de transferencia de información viable, así como una mejor capacidad de conexión entre equipos [7] [8]. Para más información sobre el módulo de red utilizado, diríjase al *Anexo A4.*

GPIO

Los pines GPIO (*General Purpose Input/Output*) son esenciales en la tarjeta Raspberry Pi debido a que es la interfaz física entre los programas y el mundo exterior, los cuales se pueden apreciar en la Figura 1.2. A partir de estos, se puede controlar simples acciones como encender o apagar un LED por medio de órdenes provenientes del software (salidas), o realizar una instrucción por medio de señales externas enviadas a los pines del módulo para que suceda alguna acción específica en el software enlazado (entradas). [10]

Figura 1.2 Pines GPIO de la tarjeta Raspberry. [10]

En la Figura 1.3 se observa que la tarjeta Raspberry Pi 3 está constituida de 40 pines, de los cuales 26 son pines de propósito general (GPIO), 8 pines corresponden a tierra (GND), 2 pines a alimentación de 5 voltios (5V), 2 pines a alimentación de 3.3 voltios (3.3V) y 2 pines de uso específico para aplicaciones avanzadas (ID EEPROM). [10]

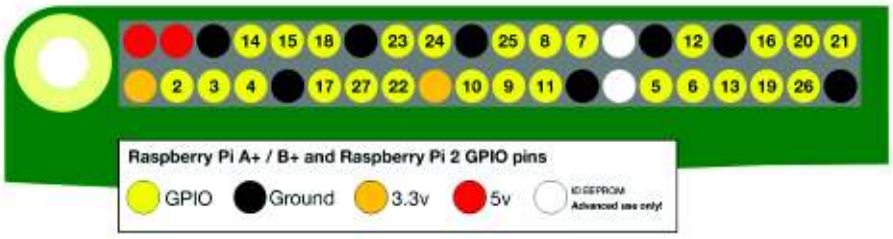


Figura 1.3 Distribución y numeración de los pines GPIO. [10]

USB

El modelo B de Raspberry Pi contiene dos pares de puertos USB 2.0 (4 puertos USB en total); a diferencia del modelo A que solo tiene un puerto. [6]

Ethernet

Dispone de un puerto LAN Ethernet para conexión mediante cable RJ45 con una capacidad de transferencia de 10/100 Mbps. [7] [8]

Salida Analógica

Al igual que su versión anterior, posee un puerto de audio minijack de 3.5mm para conectar dispositivos compatibles. [8]

HDMI

Incluye un puerto full HDMI, el cual permite la conexión directa a monitores o televisores que dispongan del mismo puerto por medio del cable HDMI. [11]

Puerto CSI

Permite la conexión de la cámara Raspberry bajo el estándar CSI (*Camera Serial Interface*) a la tarjeta Raspberry Pi anfitrión mediante un bus serial. [12]

Puerto DSI

Permite la conexión de un display bajo las especificaciones DSI (*Display Serial Interface*), compatible con el sistema Raspberry Pi, por medio de un bus serial. [13]

Energización

La energización del equipo se lleva a cabo mediante una fuente de alimentación conectada al puerto micro USB designado. La fuente debe cumplir con los valores recomendados de voltaje y corriente, según el modelo de tarjeta Raspberry. Para

el caso de Raspberry Pi 3 modelo B, se requiere una fuente de 5 voltios DC con un mínimo de corriente de 2 amperios (2.5 A recomendado) [4] [8]. El sistema no cuenta con un botón de encendido/apagado, por lo que el proceso de alimentación se da de manera directa al momento de acoplar la fuente.

Tarjeta SD

La Raspberry Pi, al no tener integrado un disco duro, requiere una microSD como indica la Figura 1.4, la cual contiene el sistema operativo, así como los programas y la información que se vaya añadiendo al momento de trabajar en la plataforma visual de Raspberry. [4]

Figura 1.4 Tarjeta microSD Raspberry con NOOBS preinstalado

Un punto a considerar es la capacidad de la microSD que se desea acoplar al módulo Raspberry Pi. Para ello, lo más importante es saber el espacio que el sistema operativo ocupará en ésta. Actualmente, la capacidad mínima recomendada es de 8GB, pero puede incluirse memorias de mayor capacidad para un funcionamiento más estable. [14]

También debe tomarse en cuenta la clase de microSD para una mayor velocidad de escritura de la misma. Una tarjeta clase 4 tendrá una capacidad de escritura de 4MB/s, mientras que una clase 10 será capaz de lograr una escritura de 10MB/s. Sin embargo, esto no significa que una tarjeta clase 10 superará a una tarjeta clase 4, en términos de uso general, ya que a menudo estas velocidades de escritura se logran al costo de velocidad de lectura y tiempos de búsqueda aumentados. [14]

Software

Raspbian

Raspbian es el sistema operativo de uso común recomendado para trabajar con tarjetas Raspberry Pi, siendo éste un software libre con arquitectura basado en Debian, bajo optimización de funcionamiento y compatibilidad con el hardware de las tarjetas anteriormente mencionadas [15]. Además, el sistema operativo Raspbian trae consigo más de 35.000 paquetes, los cuales se definen como software precompilado para una fácil instalación de los mismos en la tarjeta Raspberry Pi. [15]

phpMyAdmin

Es una herramienta de software libre que permite manejar la administración de MySQL sobre la Web. Para ello, phpMyAdmin ofrece una amplia gama de operaciones; entre las más frecuentes se encuentran el manejo de bases de datos, tablas, columnas, relaciones, usuarios, permisos, entre otras. [16]

VNC Connect

Es un software de acceso remoto multiplataforma que permite realizar conexiones a distancia, en tiempo real con dispositivos programados y conectados en la red. Tiene la finalidad de visualizar, monitorear y controlar los equipos que se desee analizar [17]. Con este programa, se puede observar el escritorio de la Raspberry Pi dentro de una ventana generada por el mismo software en otro equipo computacional e incluso en un dispositivo móvil.

De este modo, es capaz de controlar el entorno del módulo como si se trabajase directamente con éste [18]. La aplicación VNC Connect viene incluida con el sistema operativo Raspbian. VNC Connect consiste en dos partes: *VNC Server* que permite la configuración de parámetros para el acceso remoto al módulo y *VNC Viewer* que permite visualizar e interactuar con el espacio de trabajo de Raspbian a través de otros equipos configurados para el acceso. [18]

Python

Python es un lenguaje de programación eficaz debido a su estructura de datos de alto nivel a su enfoque en la programación dirigida a objetos. Además, es sencillo y de fácil aprendizaje ya que ofrece una estructura gramatical ordenada. Incorpora un tipado dinámico e interpretación natural, lo cual lo convierte en un lenguaje ideal para scripting (archivos de órdenes) y desarrollo eficiente de aplicaciones. [19]

OpenCV

OpenCV (*Open Source Computer Vision Library*) es un software de código abierto con librerías referentes a visión por computadora y aprendizaje de máquinas. Fue creada para proveer una infraestructura común dedicada a proyectos de la misma índole y acelerar el uso de la percepción artificial en productos comerciales. [20]

NumPy

NumPy es un paquete fundamental para la informática científica en Python, en el cual se encuentran funciones como procesador de matrices N-dimensionales, funciones de radiodifusión sofisticadas, herramientas para codificación C/C++ y Fortran. Además, presenta varias utilidades en cuanto a Álgebra lineal, transformadas de Fourier, entre otras capacidades numéricas. [21]

Julius

Es un software de reconocimiento de voz de alto rendimiento para investigadores y desarrolladores con la capacidad de realizar decodificaciones, casi en tiempo real, en pruebas de dictado de palabras. [22]

Cámara Raspberry PI

Introducción

La cámara Raspberry Pi, como se observa en la Figura 1.5, es un complemento multimedia oficial de la Fundación Raspberry Pi. El modelo original de 5 megapíxeles estuvo disponible al público en 2013 y una nueva versión de 8 megapíxeles fue presentada en 2016. Ambos modelos poseen versiones de luz visible e infrarroja. [23]

Figura 1.5 Cámara Raspberry Pi v2. [23]

La cámara Raspberry Pi puede ser usada para grabar videos en alta definición, así como capturar fotografías. Existe una gran cantidad de proyectos que incluyen su funcionalidad en ambición al ingenio del usuario, como grabación en cámara lenta (*slow motion*), o en cámara rápida (*time lapse*). Además, se pueden encontrar librerías adjuntas al módulo visual, con las cuales se pueden añadir efectos a las capturas [23]. Para mayor información diríjase al *Anexo A5.*

Aplicaciones

Las aplicaciones para la cámara Raspberry Pi contemplan varios fines, como por ejemplo en aplicaciones de seguridad para el hogar o incluso para captar fenómenos en la naturaleza [23]. En el área técnica, se destina su funcionalidad en proyectos de reconocimiento de objetos, con respuesta según lo visualizado; principalmente para seguimiento, que es parte fundamental de la industria y la robótica aplicada.

2. METODOLOGÍA

Los métodos utilizados en el proceso de diseño, ensamblaje, programación y pruebas de los módulos didácticos Raspberry Pi son los siguientes:

2.1 MÉTODO COMPARATIVO

Este método se tomó a manera de debate con el propósito de evaluar el alcance y la capacidad del módulo Raspberry Pi, así como de otros sistemas de similar funcionalidad en el ámbito tecnológico. Se consideraron los beneficios que cada

una de las soluciones presenta, teniendo en cuenta que siempre se buscará la opción más rentable y que cumpla satisfactoriamente con el objetivo.

Del mismo modo, el método comparativo permitió indagar la forma de desarrollar una programación distinta y variada que llegara al mismo resultado, ya sea ésta más corta o más extensa, más simple o más compleja, demostrando que existe más de una posible respuesta a un problema planteado.

2.2 MÉTODO ANALÍTICO

Mediante este método fue posible identificar, conocer y aprender todas las partes esenciales que componen el hardware de la tarjeta Raspberry Pi. Se analizó cada componente de su arquitectura, la función que desempeña, los estándares que cumple, entre otros factores relevantes de interés.

Adicionalmente, el método analítico fue aplicado en el reconocimiento de cada librería y línea de código perteneciente a los programas desarrollados en Python. De este modo fue posible conocer el propósito que cubre cada una de éstas dentro de las prácticas de laboratorio desarrolladas. Al mismo tiempo, como resultado del análisis, se identificaron librerías y comandos fundamentales para la elaboración de futuros proyectos que abarquen el mismo campo.

2.3 MÉTODO EXPERIMENTAL

El método experimental se utilizó para elaborar el diseño del módulo didáctico con los recursos y materiales adquiridos. Los elementos complementarios se han colocado de manera que sea más fácil su utilización; además de cómoda y de bajo riesgo de averías si se manipula constantemente.

De igual manera, se empleó el método experimental al realizar las prácticas de laboratorio mediante el software Python, con el fin de analizar las respuestas obtenidas. Así, fue factible incluir y simplificar funciones que mejoren los resultados

hasta obtener un producto final satisfactorio y ejecutable que cumpla con los objetivos planteados.

2.4 MÉTODO DEDUCTIVO

Gracias a este método se pueden obtener conclusiones y puntos clave a tomar en cuenta, resultado de las pruebas realizadas con las prácticas de laboratorio desarrolladas. Así, es posible esclarecer un rendimiento más eficiente, como una buena iluminación al utilizar la cámara o una clara pronunciación al trabajar con el micrófono.

3. RESULTADOS Y DISCUSIÓN

3.1 DISEÑO DE LOS MÓDULOS

El proyecto considera el desarrollo de 9 módulos didácticos basados en tecnología Raspberry Pi. El diseño de los 9 módulos es el mismo, y se describe a continuación.

Se consideró un case plástico rectangular de color negro, dentro del cual se encuentra la tarjeta Raspberry Pi 3 para su seguridad, protegiendo el procesador y puntos sensibles del contacto directo y daños por estática. Las dimensiones del case son 6 cm de largo x 2,5 cm de ancho x 9 cm de profundidad, adaptándose de forma precisa a la tarjeta Raspberry Pi 3 y a sus periféricos.

En la **Figura 3.1** se presentan los puertos desde una vista frontal y son los siguientes:

- 1 Puerto CSI para la conexión de la cámara dedicada para Raspberry.
- 1 Puerto DSI para la conexión de un display táctil dedicado para Raspberry.
- 1 Puerto Ethernet 10/100 para conectarse físicamente a la Red.
- 4 Puertos USB para la conexión de periféricos.
- 1 Puerto GPIO de 40 pines utilizados en proyectos de electrónica.
- 1 Puerto DSI para la conexión de un display táctil dedicado para Raspberry.

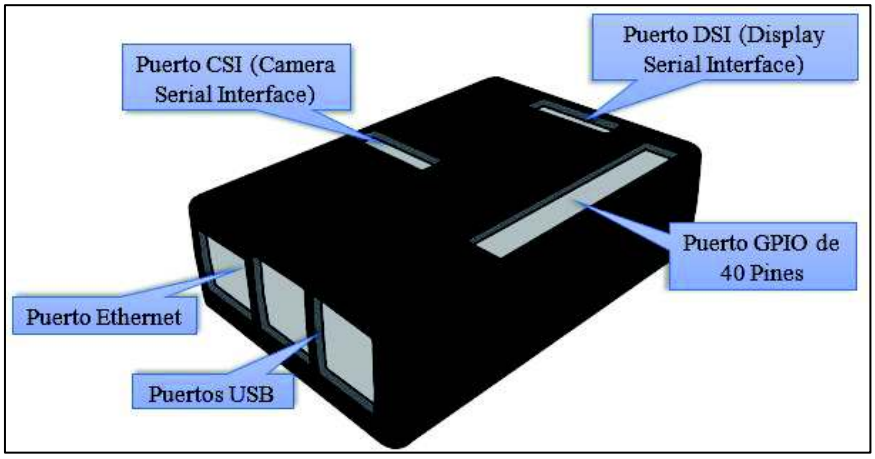


Figura 3.1 Perspectiva frontal del Case

En la **Figura 3.2** se presentan los puertos desde una vista posterior y son los siguientes:

- Puerto GPIO de 40 pines.
- Puerto para display táctil.
- Puerto para tarjeta microSD, la cual cumplirá con las funciones de disco duro del módulo.
- Puerto mini USB para la conexión de una fuente de energía de 5v-2A.
- Puerto HDMI dedicado a la salida multimedia hacia dispositivos compatibles (pantallas).
- Salida de audio analógico estéreo de 4 polos.
- Puerto CSI para cámara dedicada.

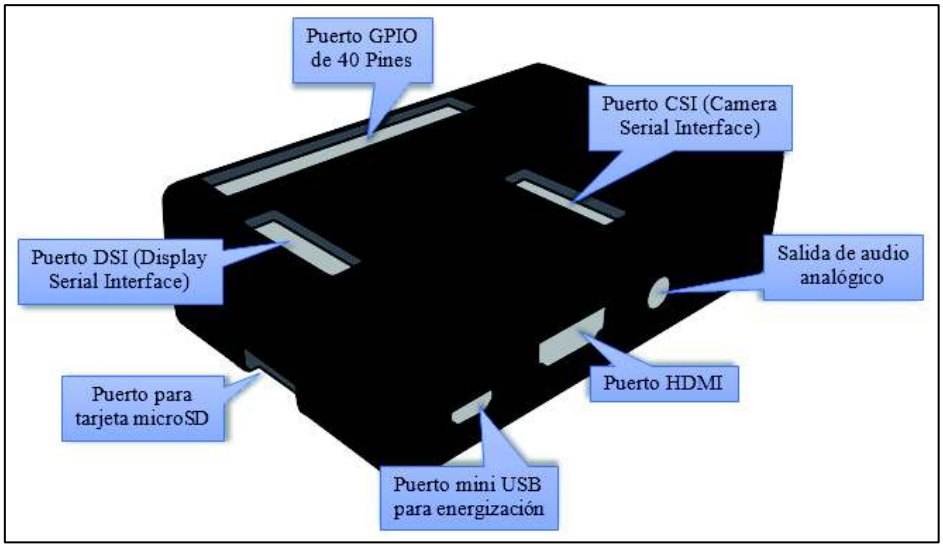


Figura 3.2 Perspectiva Posterior del Case

El diseño de las placas de LEDs fue realizado en el software Proteus 7.8, el cual permite el diseño y simulación de circuitos electrónicos, logrando así una mejor estructura y ubicación de los elementos en la placa de 2,5 cm x 4 cm. Los elementos de la placa son:

- 2 Resistencias de 330 Ω.
- 1 LED color azul.
- 1 LED color verde.
- 2 Pares de espadines en los que se conectan los LEDs.
- 3 Cables para protoboard (Dupont).

El valor de las resistencias a utilizar se lo determinó con la siguiente fórmula [24]:

$$R = \frac{Vfuente\ -\ Vdiodo}{I}$$

Teniendo en cuenta que el voltaje de luminiscencia tanto del diodo verde, como del diodo azul es 3.4v, con corriente de 5mA a un voltaje de alimentación de 5v, se tiene el siguiente resultado:

$$R = \frac{5v\ -\ 3.4v}{5mA} = 320\Omega$$

Considerando que 320Ω no corresponde a un valor estándar, se eligió el valor más próximo de 330Ω, permitiendo un óptimo funcionamiento y mayor protección.

Para escoger los elementos a utilizar en las placas y realizar las conexiones entre cada uno de ellos, se utilizó Isis 7.8, herramienta de Proteus que permite el diseño y simulación de circuitos electrónicos, como se lo puede ver en la **Figura 3.3**.

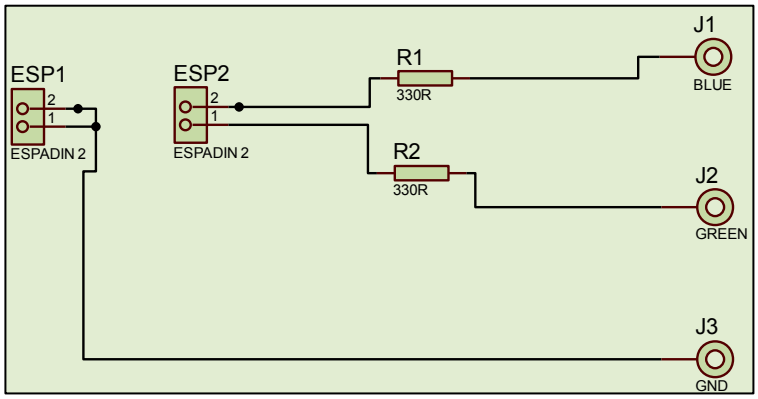


Figura 3.3 Diseño placas de LEDs realizado en Isis-Proteus

Por otro lado, una vez realizado el diseño de la placa de LEDs en Isis, se procedió al enrutamiento de las pistas, ubicación de elementos, distancias de separación entre elementos y dimensionamiento de la placa. Para ello, se usó Ares 7.8, que es otra herramienta de Proteus que permite realizar los diseños de las placas para circuitos impresos. El diseño del circuito con sus dimensiones se lo puede observar en la **Figura 3.4**

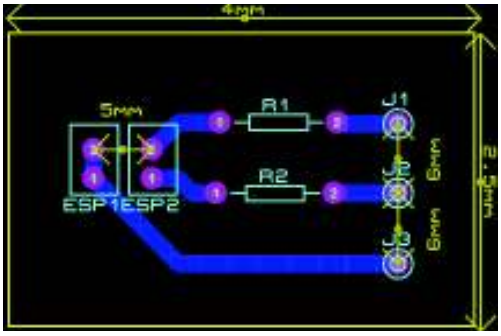


Figura 3.4 Diseño placa de LEDs realizado en Ares- Proteus

Para tener una mejor perspectiva de cómo quedará el circuito terminado, Ares brinda una opción de visualización en 3D, que resulta muy útil teniendo en cuenta que muestra todos los elementos físicos que conforman el circuito. En la **Figura 3.5** se puede observar la placa de LEDs terminada.

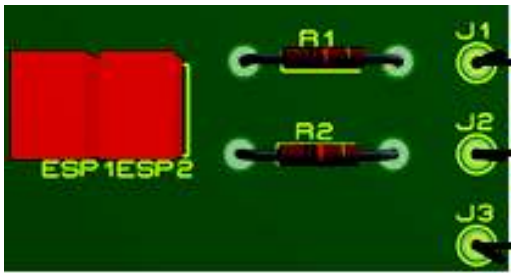


Figura 3.5 Visualización 3D de la placa de LEDs

Adicionalmente, se agrega la fuente de poder de 5V-2A para la energización y el convertidor de señal de HDMI a VGA. El convertidor fue necesario ya que la señal de video propia de la tarjeta Raspberry es HDMI y la entrada de los monitores que se encuentran en el laboratorio de Microprocesadores de la ESFOT es VGA.

Para la fijación de la cámara, se elaboró una cubierta plástica de medidas afines al dispositivo a modo de contención, así como de protección con el propósito de evitar la manipulación directa de sus elementos electrónicos. Adicionalmente, se instaló un soporte flexible entre la cubierta plástica de la cámara y el case del módulo, con el fin de facilitar el manejo del módulo cámara.

Por último, el mini micrófono USB estará conectado en uno de los 4 puertos USB integrados en el módulo Raspberry Pi 3.

Con todas las consideraciones mencionadas, el diseño final del módulo es el que se muestra en la **Figura 3.6**.

Figura 3.6 Diseño Final del Módulo

3.2 CONSTRUCCIÓN DE LOS MÓDULOS

Para la construcción de los módulos se tomaron la tarjeta Raspberry Pi 3 y la base del case de protección, como se muestra en la **Figura 3.7**, orientándolos en la dirección correcta de colocación para su posterior ensamblaje y sujeción.

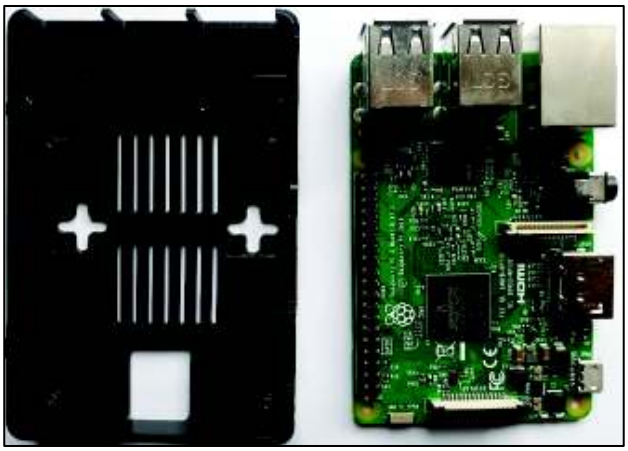


Figura 3.7 Base de case y tarjeta Raspberry Pi 3

Determinada la orientación de colocación, se sobrepuso la tarjeta Raspberry dentro de la base del case y se la aseguró de forma firme y cuidadosa, como se observa en la **Figura 3.8**. Para una correcta sujeción, se colocaron cuatro tornillos de tamaño estándar que coinciden con los orificios de la tarjeta y de la base.

Figura 3.8 Tarjeta Raspberry Pi en la base

Una vez lista la base, se procedió a colocar los accesorios en la tapa del case, estos accesorios son la base flexible que sujeta la cámara y la placa de LEDs.

Como puede observarse en la **Figura 3.9**, la base flexible para la cámara consta de un brazo flexible, cilíndrico y metálico de 15 cm de largo que permite una mejor manipulación y estabilidad de la cámara. Este se encuentra sujeto a una pequeña caja plástica de color negro fabricada mediante impresión 3D de 3 cm x 3cm, dentro de la cual se ubicará la cámara.

Figura 3.9 Base flexible para la cámara

Los siguientes accesorios son las 9 placas de LEDs de 2,5 cm x 4 cm, una por módulo, las cuales fueron fabricadas con el método tradicional basado en la impresión del diseño del circuito realizado en Proteus en una hoja de papel termotransferible. Una vez impreso el circuito, se procedió a su posterior planchado sobre una baquelita A4 de cobre previamente tratada y libre de impurezas; con esto se consigue la transferencia del circuito a la baquelita.

Una vez que los circuitos impresos fueron adheridos a la baquelita, se recortaron por sus márgenes y se desecharon los sobrantes para continuar al proceso de quemado. Este proceso consiste en retirar el exceso de cobre que no se encuentra cubierto por la impresión, dejando únicamente las pistas conductoras necesarias.

Se colocó agua caliente en un recipiente, se sumergieron las placas y sobre ellas ácido cloruro férrico, se dejó reposar por un lapso de media hora y las baquelitas estaban listas para ser perforadas.

Para finalizar el proceso de construcción, se perforó en los espacios fijados para los elementos; posteriormente, se soldaron los mismos con estaño y se colocaron los LEDs sobre los espadines. El resultado final se puede apreciar en la **Figura 3.10**.

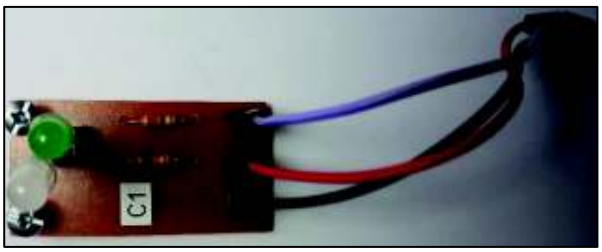


Figura 3.10 Placa de LEDs

En la tapa del case se realizaron perforaciones exactas en los espacios para la base flexible de la cámara y la placa de LEDs, tomando en cuenta que ninguno interfiera con el otro o con algún periférico del módulo. Se puede observar lo realizado en la **Figura 3.11**.

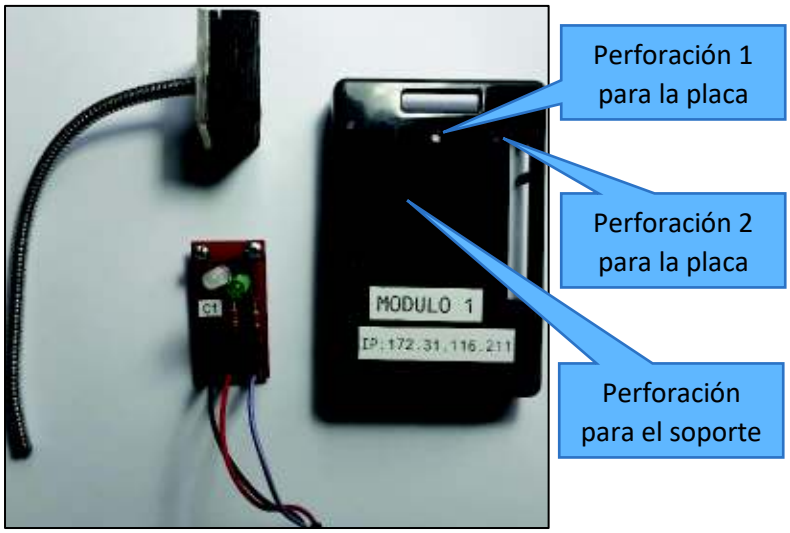


Figura 3.11 Tapa del case y accesorios

En la **Figura 3.12** se puede ver que la base flexible de la cámara fue sujeta desde la parte inferior a la tapa del case, quedando de forma vertical, fija y en posición frontal al usuario, permitiendo un manejo adecuado de la misma.

Figura 3.12 Ubicación de la base flexible

Como muestra la **Figura 3.13**, la placa de LEDs se la ubicó entre la base flexible para la cámara y el puerto GPIO, sujetándola con tornillos de ½ x 4 en los orificios previamente realizados. En esta posición se tiene una fácil conexión de los cables de la placa a los pines del puerto GPIO.

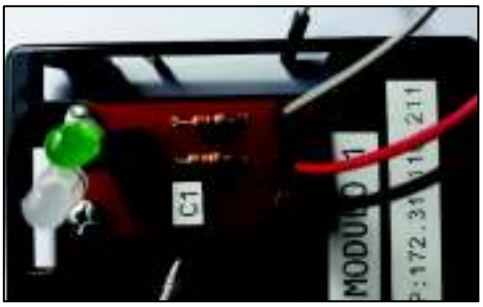


Figura 3.13 Ubicación de la placa de LEDs

Como siguiente paso, se introdujo el bus de datos de la cámara por el espacio del puerto dedicado a la cámara, ubicado en la tapa del case, y se conectó con pequeñas pinzas a su socket CSI (Camera Serial Interface) en la tarjeta Raspberry Pi. La conexión se observa en la **Figura 3.14**.

Figura 3.14 Conexión de la Cámara

Se procedió con la conexión de la placa de LEDs y GND a los pines GPIO de la tarjeta Raspberry Pi (Pines 9, 11 y GND). Cuidadosamente, se los introdujo por el espacio del puerto GPIO de la tapa, y uno a uno se los conectó a los pines, como se muestra en la **Figura 3.15**.

Figura 3.15 Conexión de LEDs a pines GPIO

Se colocó la cámara en su respectiva caja protectora, como se muestra en la **Figura 3.16**.

Figura 3.16 Cámara en su case

Finalmente, se conectó el micrófono en un puerto USB, como se ve en la **Figura 3.17**. Además, se introdujo la memoria microSD con el sistema operativo Raspbian cargado, en su respectivo socket, como se muestra en la **Figura 3.18**.

Figura 3.17 Conexión de micrófono

Figura 3.18 Ubicación de memoria microSD

Los módulos y cada uno de sus elementos, han sido etiquetados para su reconocimiento.

3.3 DESARROLLO DE LAS PRÁCTICAS

En el Anexo C (Hojas Guía Para Instructor), se detalla el proceso de preparación del módulo Raspberry Pi, incluyendo su energización y la configuración del puerto de video.

PRÁCTICA 1:

Procesamiento de imágenes mediante el módulo Raspberry Pi 3.

Esta práctica consiste en el reconocimiento de los colores rojo, verde y amarillo. Cuando la cámara distinga un objeto de uno de los colores indicados, se desplegará en el monitor las palabras “rojo”, “verde”, o “amarillo”, dependiendo el caso.

La identificación de los colores designados parte del código RGB dentro de los parámetros de configuración en el programa, cuya descripción se puede encontrar en páginas web, facilitando esta información. La mayoría de las páginas nos mostrarán un amplio espectro de tonalidades de todos los colores, desde el más claro hasta el más oscuro. Desde este punto queda bajo criterio propio elegir un rango de tonalidad de los colores aplicados que se adapte a lo que se desea identificar por medio de la cámara.

El hardware y software utilizados en esta práctica incluyen:

- Hardware: Módulo Raspberry Pi, Cámara.
- Software: Python 2.
- Extensiones y Librerías: Open CV, NumPy.

La extensión Open CV permitirá añadir funciones personalizadas a la cámara para que ésta pueda realizar un reconocimiento personalizado hacia los elementos dentro de su campo de visión.

La extensión NumPy permitirá realizar análisis matemático que determinará los valores de identificación de los colores antes mencionados.

En la **Figura 3.19** se puede observar el correspondiente Diagrama de Flujo. Como se puede apreciar, el programa comienza con la importación de las librerías "*cv2"* para OpenCV, "*time"* para sentencias de control de tiempos, y "*numpy"* para el programa de elaboración de matrices NumPy.

Luego, el programa continúa con la asignación de la cámara Raspberry Pi, que será la cámara por defecto para el análisis y captura de las imágenes.

Al haberse configurado la respectiva funcionalidad de la cámara, el programa procede con la modificación de los parámetros correspondientes a la calidad y nitidez de la imagen, como la resolución y los FPS, indispensables para obtener un claro reconocimiento y visualización. Una vez configurada y preparada, la cámara se encenderá.

Luego de que la cámara se encienda, el programa proporciona una ventana en donde se puede observar lo que esta capta, de forma que la persona se percate de lo que toma el dispositivo al presentarse un color en el área objetivo.

En este caso, el programa está diseñado para captar el color rojo, amarillo y verde, para lo cual se incluye una respuesta tanto visual como textual al momento de identificar cualquiera de los colores mencionados anteriormente.

En el **Anexo B1** se incluye la Hoja Guía para el estudiante, y en el **Anexo C1** se presenta la Hoja Guía para el instructor, incluyendo el código Python correspondiente.

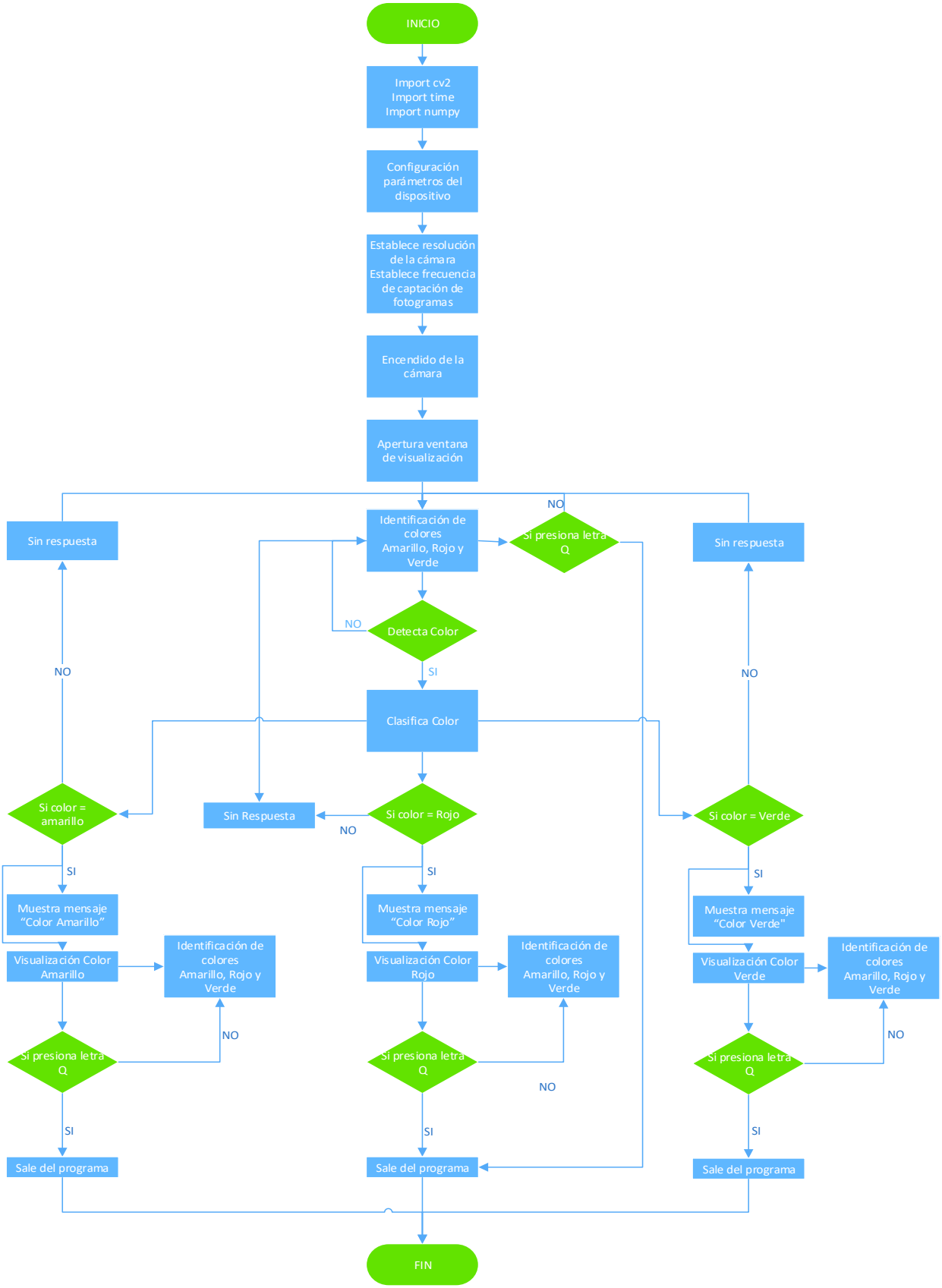


Figura 3.19 Diagrama de Flujo Procesamiento de imágenes

PRÁCTICA 2:

Procesamiento de base de datos utilizando módulos Raspberry Pi 3

Esta práctica consiste en la elaboración de una base de datos, así como de una "guía" que indicará paso a paso el proceso de edición de la misma, mediante las opciones disponibles en ésta.

El hardware y software utilizados en esta práctica incluyen:

- Hardware: Módulo Raspberry Pi, parlantes o audífonos.
- Software: Python 2, Apache, MySQL, PhpMyAdmin
- Extensiones y Librerías: PyGame

Apache es un servidor web que permitirá almacenar y transferir las bases de datos, las tablas, y la información ingresada en el programa como hipertexto; es decir, se escribirán en una página web con todos los componentes que la conforman.

MySQL funciona en conjunto con Apache como gestor de bases de datos, el cual se utilizará para crear la base *"Registro"* y la tabla *"Base1",* con la finalidad de ingresar, eliminar y modificar la información respectiva. Por defecto se incluye una contraseña de ingreso.

El software PhpMyAdmin facilitará la administración de las bases de datos y las tablas creadas en la práctica, a partir de una página web previamente configurada. De este modo se proporcionará una interfaz gráfica de fácil manejo para el usuario.

La librería de PyGame será fundamental en la creación de la guía auditiva, la cual dirá las opciones disponibles según se proceda en la edición y generación de la base de datos.

En la **Figura 3.20** se puede observar el correspondiente Diagrama de Flujo. Como se puede apreciar, el programa comienza con la importación de las librerías "*mysql.connector"* para realizar la conexión automática con la base de datos en MySQL, "*sys"* que proporciona las variables y funciones de Python, "*pygame"* para la lectura y reproducción de audios en formato wav, y "time" para el control de funcionalidades relacionadas con tiempo.

Una vez importadas todas las librerías, se realiza una primera prueba de conexión con la base de datos con el fin de enlazarse a ésta. Si todo está correctamente

configurado, se establece la conexión y se despliega el menú principal en forma de texto y audio; caso contario, el programa no iniciará.

El menú principal contiene las siguientes opciones: mostrar datos = 1, ingresar datos = 2, eliminar datos = 3, buscar datos = 4 y salir = 5. Si selecciona la opción 1 muestra todos los datos que existen en la tabla de la base de datos y muestra un submenú que permite ingresar o eliminar datos y salir del programa.

Si selecciona la opción 2, el programa solicita los siguientes detalles para ingresarlos: nombre, apellido, cédula, edad, género, dirección, teléfono y correo. Al final, indicará que los datos se ingresaron y muestra un submenú que permite mostrar o ingresar datos, volver al menú principal o salir del programa.

Si se selecciona la opción 3, se despliegan todos los datos de la tabla y una opción para ingresar el número de registro que se desea eliminar. Si el número de registro existe solicitará una confirmación de eliminación; caso contrario, indicará que no existe y muestra un submenú con las siguientes opciones: eliminar otro registro, volver al menú principal y salir del programa.

Si se selecciona la opción 4, se visualiza un menú con las opciones de búsqueda, las cuales son: por nombre, apellido o cédula y la opción de salir del programa sin realizar ninguna acción. Cualquier otra opción diferente a las mencionadas, cerrará el programa.

Si se selecciona la opción 5, el programa terminará su ejecución luego de mostrar un mensaje de agradecimiento por usar la base de datos.

Todos los menús y submenús del programa cuentan con la guía audible.

En el **Anexo B2** se incluye la Hoja Guía para el estudiante y en el **Anexo C2** se presenta la Hoja Guía para el instructor, incluyendo el código Python correspondiente.

Figura 3.20 Diagrama de Flujo Base de datos

PRÁCTICA 3:

Procesamiento de voz y control de puertos GPIO utilizando módulos Raspberry Pi 3.

Esta práctica consiste en el control de puertos GPIO para encender o apagar LEDs por medio de comandos de voz. Cuando se pronuncie la respectiva orden, a través del micrófono, se encenderá o se apagará el LED azul o el LED verde incluidos en el módulo, dependiendo de los tipos de comandos asignados para cada uno de estos.

El hardware y software utilizados en esta práctica incluyen:

- Hardware: Módulo Raspberry Pi, Micrófono, Puertos GPIO, Placa de LEDs.
- Software: Python 2, Julius
- Extensiones y Librerías: Rythmbox

El software Julius es un programa de reconocimiento de voz que se encargará de interpretar e identificar toda palabra receptada por el micrófono con el fin de ser utilizadas por el equipo a nivel de software.

Las librerías de Rhythmbox cuentan en su base de datos con una cantidad limitada de comandos en inglés, pero permiten definir comandos de control para ejecutar interacciones de manera externa, utilizando dos puertos GPIO que servirán para variar el estado de los LEDs (Encendido, Apagado) que conforman el módulo.

Para registrar comandos en la base de datos de Rhythmbox hay que editar dos archivos propios del software, ingresar el comando y su pronunciación para que pueda interpretar la palabra en la entrada del micrófono.

Para conseguir un buen resultado en el desarrollo del programa, se debe tomar en cuenta que Julius y Rhythmbox son considerados programas complementarios; es decir, sin una configuración adecuada de sus parámetros descritos en la práctica no se obtendrá el control sobre las salidas que son los LEDs conectados a los pines GPIO.

En la **Figura 3.21** se puede observar el correspondiente Diagrama de Flujo. Como se puede apreciar, el programa comienza con la importación de las librerías "*sys",*

que proporcionan las variables y funciones de Python, "*os"* proporciona funciones de interacción con el sistema operativo, y "RPi.GPIO" para el control de los pines GPIO.

Una vez importadas las librerías, verifica que Julius y Rythmbox estén operativos. En Julius comprueba que la entrada de voz, en este caso el micrófono USB, esté funcionando, mientras que en Rythmbox comprueba que los comandos estén registrados correctamente para poder ser utilizados.

Si todas las configuraciones y componentes están correctos, se visualiza un mensaje de bienvenida al programa. Los comandos que se pueden utilizar con la función que cumple cada uno y la línea <<<*please speak*>>>*,* la cual es propia del software Julius. Así queda listo el programa para el reconocimiento de los comandos de voz.

Si el comando es *"Computer blue"*, el LED azul se enciende y si el comando es "Computer pause", el LED azul se apaga.

Del mismo modo para el LED Verde, si el comando es *"Computer green",* el LED verde se enciende y si el comando es *"Computer next",* el LED verde se apaga.

Para cerrar el programa se tienen dos opciones: Presionar "CTRL + C" o cerrarlo desde la opción de la ventana del programa.

En el **Anexo B3** se incluye la Hoja Guía para el estudiante, y en el **Anexo C3** se presenta la Hoja Guía para el instructor incluyendo el código Python correspondiente.

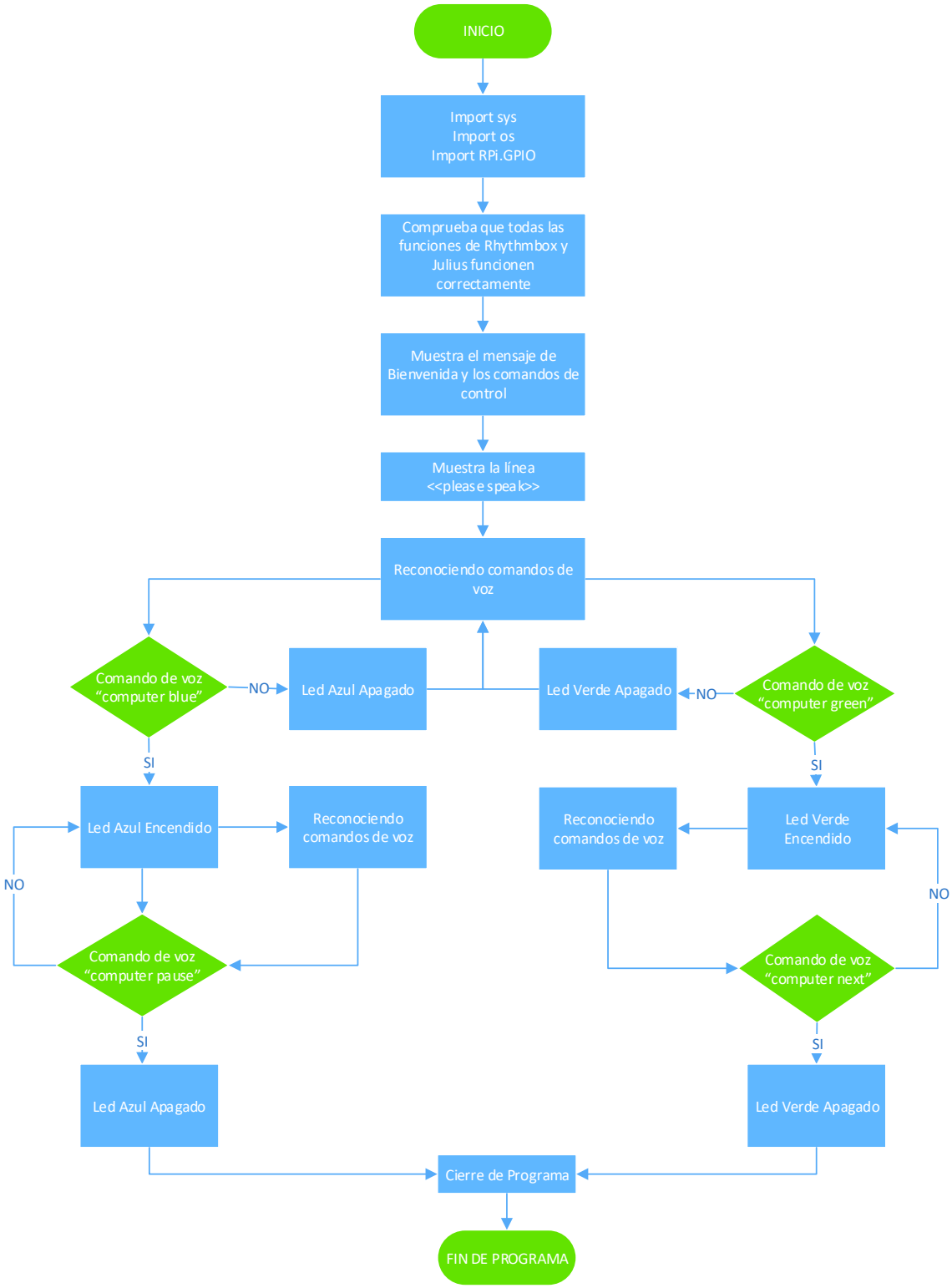


Figura 3.21 Diagrama de Flujo Procesamiento de voz

3.4 PRUEBAS DE FUNCIONAMIENTO

PRÁCTICA 1

Una vez guardado el código, iniciar con su ejecución. En la misma ventana de Python abrir la pestaña Run, dar clic a la opción Run Module, como se ve en la **Figura 3.22**. El programa empezará a correr y se observará el reconocimiento de los colores a medida que se lo ubique frente a la cámara.

Figura 3.22 Opción de ejecución del programa

Para este caso, se puede observar el reconocimiento del color rojo (**Figura 3.23**), del color amarillo (**Figura 3.24**) y del color verde (**Figura 3.25**), comprobando que el programa está funcionando correctamente.

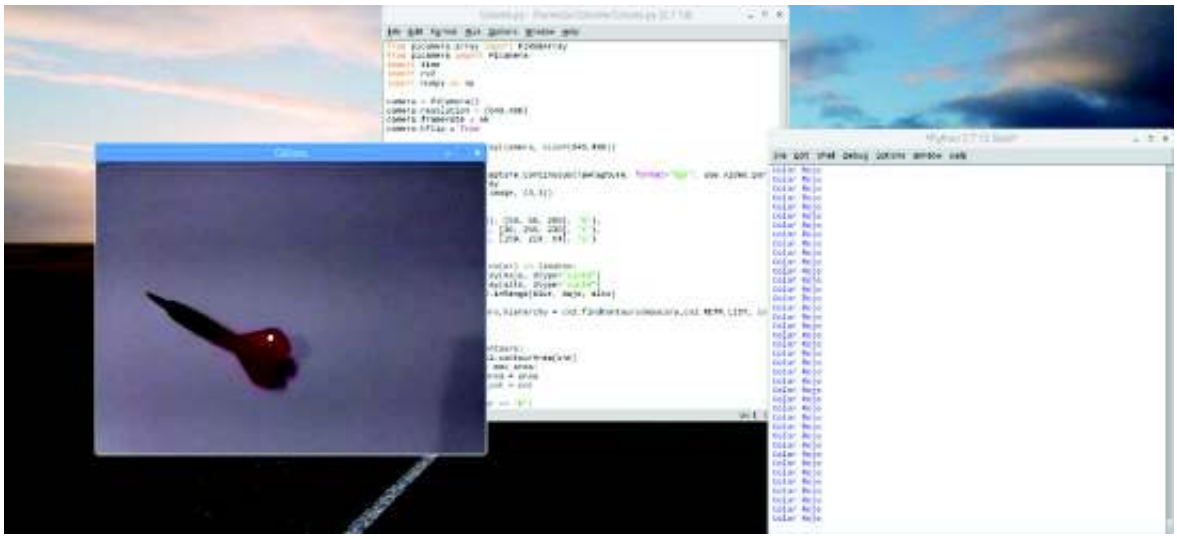


Figura 3.23 Detección del Color Rojo

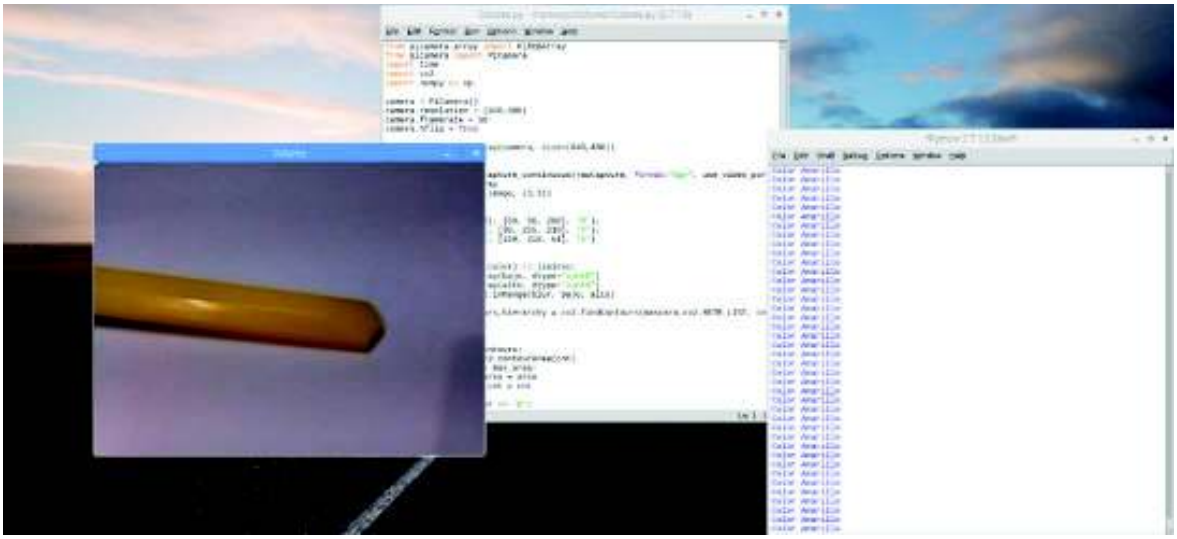


Figura 3.24 Detección del Color Amarillo

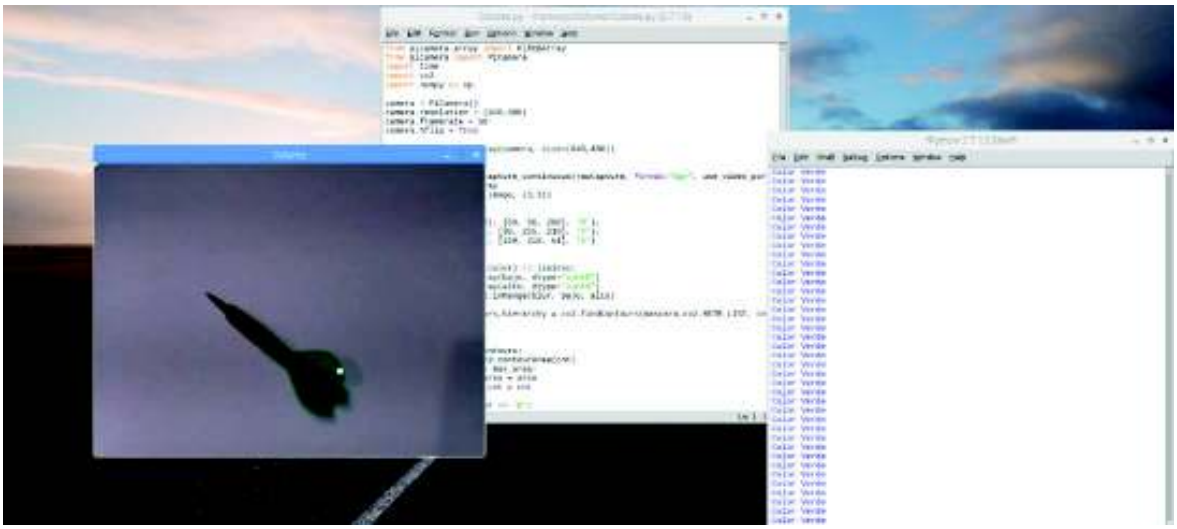


Figura 3.25 Detección del Color Verde

PRÁCTICA 2

Para ejecutar el programa, se lo puede realizar de dos maneras, abriéndolo directamente con el ejecutor de Python o desde LXTerminal.

Desde Python, ir a la pestaña Run (**Figura 3.26**) y seleccionar Run Module, o presionando F5 el programa se iniciará. Observar en la **Figura 3.27** que el programa se inicia correctamente.

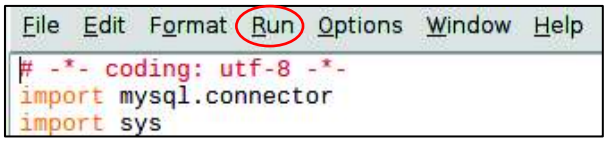


Figura 3.26 Ejecutar Programa

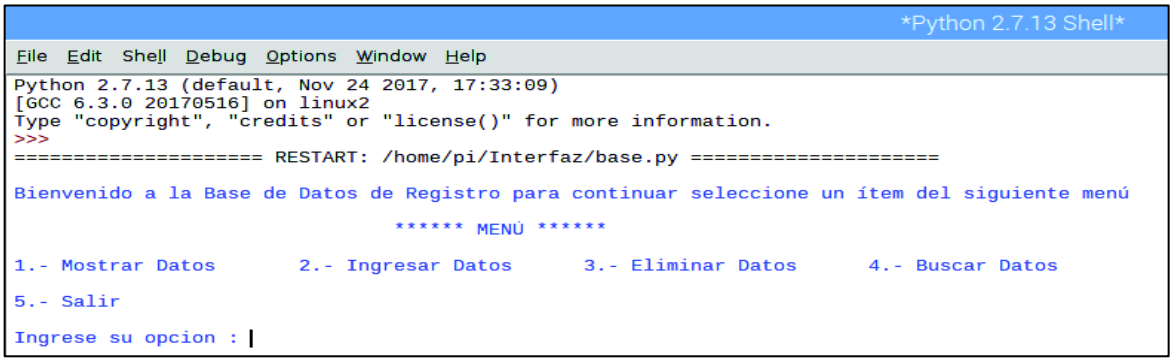


Figura 3.27 Ejecución del Programa

Desde LXTerminal, ingresar la ruta en la cual se encuentra el archivo.

El comando **cd** permite ingresar en carpetas, en este caso se ingresará a la carpeta Programas [25].

La palabra **Programas** en letras azules, indica que se ha colocado bien la ruta y se ha ingresado en la carpeta para ejecutar el programa [25]. Colocar Python con la versión en la que se hizo el programa, en este caso 2 seguido del nombre del archivo; presionar Enter y automáticamente se ejecutará. (**Figura 3.28**).

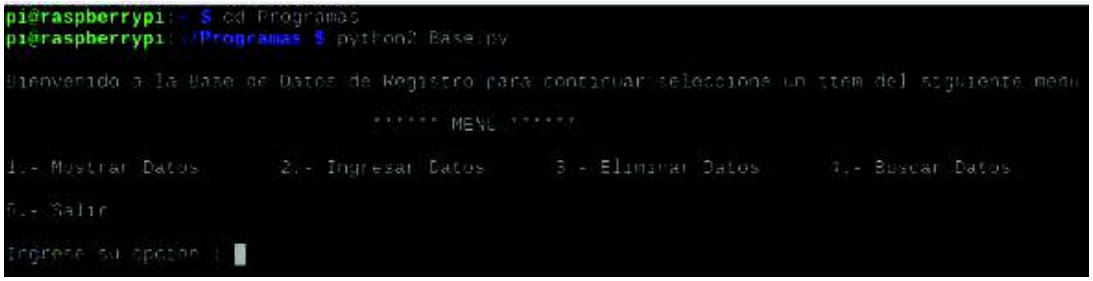


Figura 3.28 Programa en LXTerminal

Para comprobar que el programa funciona, se gestionarán datos en la base de datos. Primero mostrar datos, para esto presionar 1 y observar que la tabla está vacía. Ingresar nuevamente a phpMyAdmin mediante el navegador Chromium y verificar que la tabla en la base de datos esté igualmente vacía. (**Figura 3.29**).

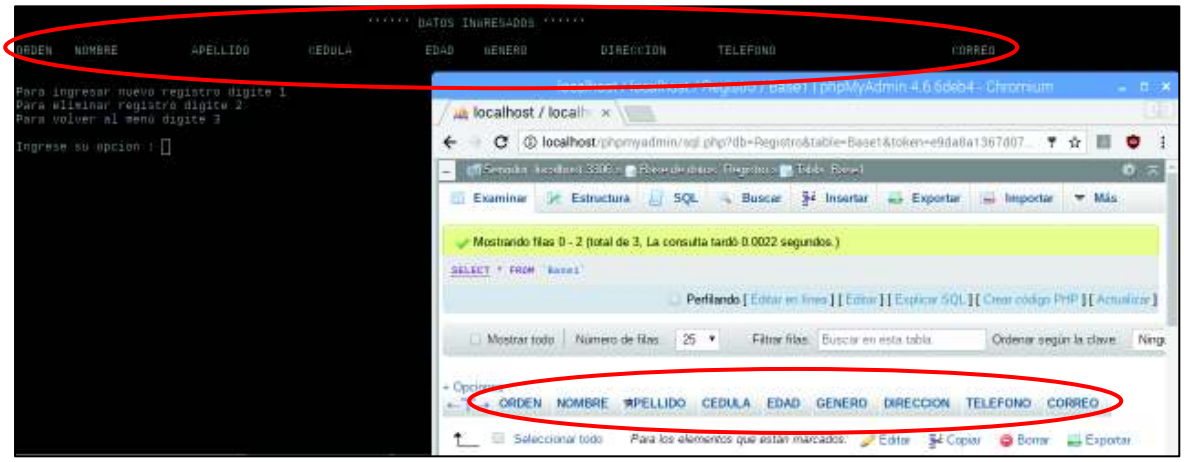


Figura 3.29 Comparación del programa y PhpMyAdmin

Al ingresar nuevos registros, se observa que los datos se ingresaron con éxito, así se ingresa más registros (**Figura 3.30**)

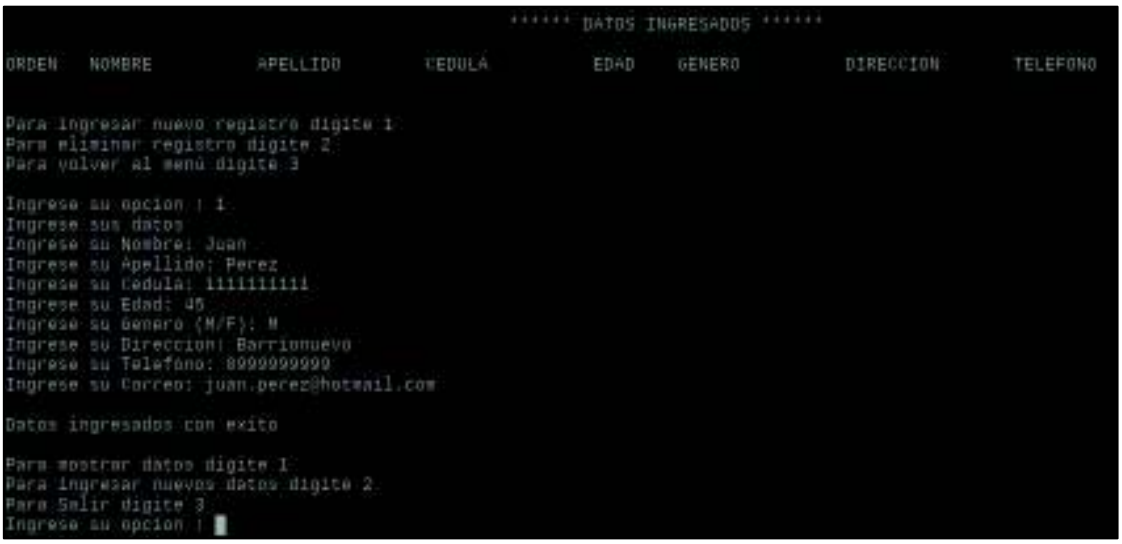


Figura 3.30 Prueba Ingresar datos

Revisar los registros y comprobar que se visualizan, tanto en la pantalla de comandos como en la base de datos de PhpMyAdmin (**Figura 3.31**).

Figura 3.31 Prueba Comprobar datos

Se elimina el registro número 8 y se observa que se realizó con éxito la eliminación (**Figura 3.32**).

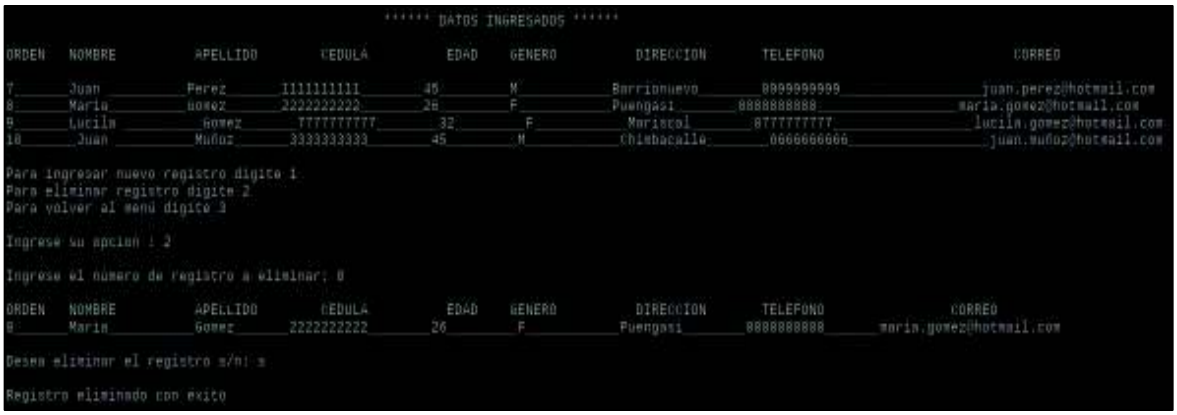


Figura 3.32 Prueba Eliminar Datos

El registro se eliminó de la base de datos como se esperaba (**Figuras 3.33 y 3.34**).

Figura 3.33 Prueba eliminar datos

Figura 3.34 Comparación con PhpMyAdmin

Para finalizar, se seleccionará buscar datos. Buscar por nombre e ingresar el dato Juan; se observa que devuelve dos coincidencias con todos sus datos (**Figura 3.35**).

Figura 3.35 Prueba Buscar

Con todas las pruebas anteriores, se comprueba que el programa está funcionando correctamente.

PRÁCTICA 3

Para ejecutar el programa, únicamente se da doble clic sobre el archivo **Control.sh**. Aparece la ventana que se observa en la **Figura 3.36**, se selecciona **Ejecutar en Terminal** y el programa se abrirá automáticamente, como se muestra en la **Figura 3.37.**

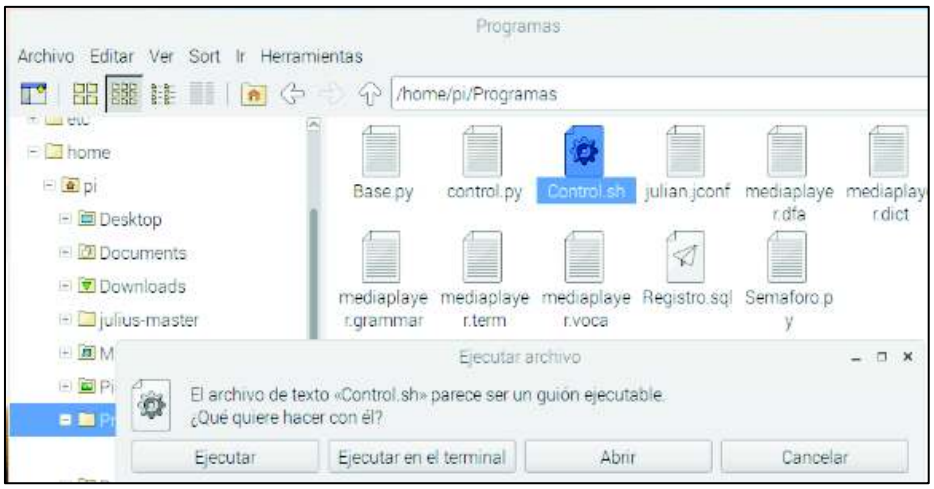


Figura 3.36 Ejecutar Programa

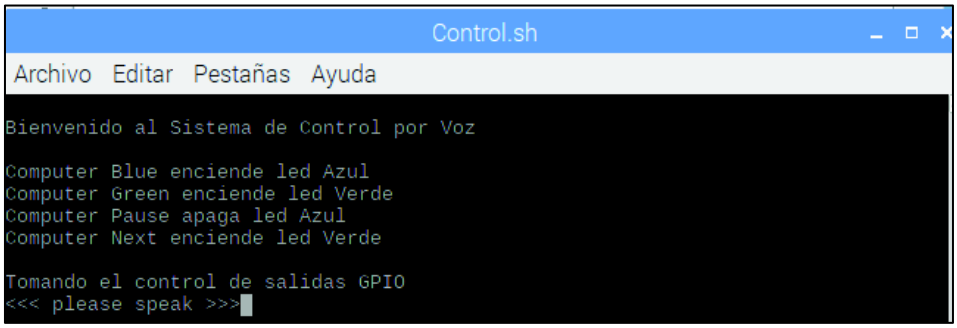


Figura 3.37 Ejecución del Programa

La línea “<<< please speak >>>” es propia de Julius y aparecerá automáticamente para realizar la captación y el reconocimiento de la voz. La estructura de pronunciación de Julius es computer más el comando con el que se ejecutará la acción (computer + comando).

Según se pronuncie el comando, aparecerá en la pantalla lo reconocido por Julius. Tal y como se muestra en la **Figura 3.38,** los comandos identificados son “computer blue” y “computer green”, en cuyo caso encenderá los LEDs azul y verde, respectivamente (**Figura 3.39**).

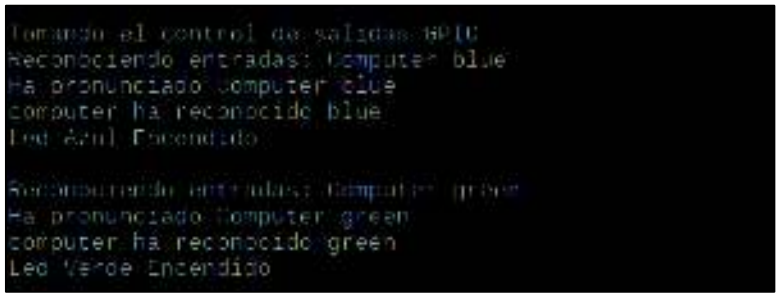


Figura 3.38 Encender LEDs azul y verde

Figura 3.39 LEDs azul y verde encendidos

De igual manera, al pronunciar los comandos “computer pause” y “computer next”, como se indica en la **Figura 3.40**, Julius los reconocerá y habilitará el apagado de los LEDs azul y verde, respectivamente (**Figura 3.41**).

```
Reconociendo entradas: Computer next
Ha pronunciado Computer next
computer ha reconocido next
Led Verde Apagado

Reconociendo entradas: Computer pause
Ha pronunciado Computer pause
computer ha reconocido pause
Led Azul Apagado
```

Figura 3.40 Apagar LEDs azul y verde

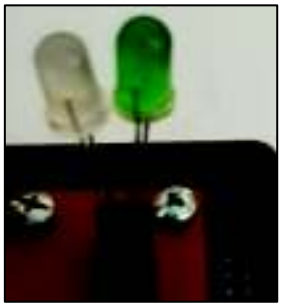


Figura 3.41 LEDs azul y verde apagados

Al realizar estos pasos, se puede ver que la práctica funciona de manera óptima, demostrando que el módulo y sus complementos funcionan correctamente.

3.5 COSTOS DE IMPLEMENTACIÓN

Costo unitario de los Módulos

CANTIDAD	DESCRIPCIÓN	V.UNITARIO
1	Rasberry Pi 3	$70,00
1	Cámara para Raspberry Pi 3	$30,00
1	Micrófono USB	$6,00
1	Case para Raspberry Pi 3	$7,00
1	Case para Cámara	$2,00
1	Soporte Flexible para Cámara	$3,50
1	Memoria MicroSD 16 GB clase 10	$11,00
1	Fuente de Alimentación 5v-2A	$6,00
1	Cable VGA 1.5 m	$5,00
1	Placa de Leds	$4,00
1	Convertidor HDMI a VGA	$10,00
1	Etiquetación de Módulo + Rollo de Etiquetas	$5,00
	TOTAL	**$159,50**

Tabla 1.2 Costo unitario de los Módulos

Costo Total del Proyecto

CANTIDAD	DESCRIPCIÓN	V.UNITARIO	V.TOTAL
9	Raspberry Pi 3	$70,00	$630,00
9	Cámaras para Raspberry Pi 3	$30,00	$270,00
9	Micrófonos USB	$6,00	$54,00
9	Cases para Raspberry Pi 3	$7,00	$63,00
9	Cases para Cámaras	$2,00	$18,00
9	Soportes Flexibles para Cámara	$3,50	$31,50
9	Memorias microSD 16 GB clase 10	$11,00	$99,00
9	Fuentes de Alimentación 5v-2A	$6,00	$54,00
9	Cables VGA 1.5 m	$5,00	$45,00
9	Placas de Leds	$4,00	$36,00
9	Convertidores HDMI a VGA	$10,00	$90,00
9	Ensamblaje de los módulos	$5,00	$45,00
9	Etiquetación de Módulos + Rollos de Etiquetas	$5,00	$45,00
		TOTAL	**$1.480,50**

Tabla 1.3 Costo total del Proyecto

NOTA: En el presente presupuesto se han considerado únicamente costos de hardware. No se incluyen rubros de desarrollo intelectual.

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- El desarrollo de los módulos contribuirá a que los estudiantes del Laboratorio de Microprocesadores adquieran destreza y conocimiento con la tecnología Raspberry, procesamiento de voz, imágenes y bases de datos.

- Los módulos Raspberry Pi 3 han demostrado poseer un rendimiento considerable, tomando en cuenta sus características de fábrica. Además, su funcionalidad es muy estable al momento de ejecutar programas y utilizar los complementos externos, siempre y cuando se tomen las precauciones necesarias en el momento de utilizarlos.

- La cámara Raspberry Pi es, sin duda alguna, un complemento esencial para la investigación. Su versatilidad en captura y grabación de imágenes permite obtener resultados que pueden ser añadidos a un conjunto de instrucciones, correspondientes a programas cuya función esté predispuesta a analizar datos y entornos de índole gráfica.

- El mini micrófono USB, junto con el módulo Raspberry Pi, hacen posible el reconocimiento de voz, siendo ésta muy importante en la ejecución de programas que incluyan factores de control por voz. Así, es factible generar archivos en un formato legible para la mayoría de dispositivos y que pueden ser agregados a las líneas de programación de proyectos afines a su utilidad.

- El reconocimiento de imágenes cumplió con su función satisfactoriamente, desde su ejecución hasta el cumplimiento de su tarea en base a los tres colores previamente registrados en el programa. De primera instancia, se pudo observar cómo el proyecto iniciaba tal y como se planteó, con la apertura de dos ventanas cuyo propósito sería el de indicar tanto de manera visual como textual el reconocimiento de los colores.

Luego, la identificación se efectuaba a través de las ventanas mencionadas con anterioridad por medio de los centroides que son puntos blancos visibles en la primera ventana que corresponde a la vista de la cámara y que sirven como indicadores al encontrar uno de los colores que forma parte del código; además se muestra el mensaje correspondiente del color reconocido en la segunda ventana.

- El funcionamiento del control de voz con respuesta visual mediante LEDs, utilizando pines del puerto GPIO, cumplió con lo esperado de forma efectiva y sin errores. A partir de la ejecución del programa mediante el script elaborado, se observó la apertura de la ventana de terminal en la que seguidamente se visualizó las instrucciones del cómo interactuar con el proyecto para que se pudieran encender o apagar los LEDs. Al pronunciar los comandos correspondientes para el encendido y apagado de las luces, se comprobó que el sistema funciona correctamente tanto en hardware como en software ya que se agregó una respuesta textual que indique lo que se pronuncia y lo que se realiza en el instante en que se menciona las líneas de control.

- La ejecución de la base de datos generada se dio sin inconvenientes por medio de la ventana de Python, así como por la ventana de Terminal. Se observó la matriz de la base de datos con sus respectivas opciones de edición y manejo, al mismo tiempo que se desarrollaba la introducción y guía de forma sonora. Además, se constató que la edición de la información continuó fluidamente al igual que el sistema guía. Luego, se observó que los registros ingresados se conservaban en los respectivos parámetros donde se añadió información con anterioridad. Por último, se comprobó que la finalización del programa se realizaba correctamente junto con su mensaje de audio.

- El desconocimiento de la diferencia en estructura de los comandos entre las versiones de Python 2 y Python 3 conllevó a varios errores y problemas al

momento de ejecutar los programas para las pruebas, se presentaron errores de sintaxis o fallas en el funcionamiento del programa que se solucionaron con tiempo de investigación.

- Los programas realizados en este proyecto son base para el aprendizaje de nuevas técnicas de programación y motivación para iniciar la investigación, diseño y aplicación de proyectos aún más complejos en varias áreas y campos no solo técnicos, sino siendo de gran utilidad para diferentes áreas en las que se requiera tecnología de punta.

4.2 RECOMENDACIONES

- Es indispensable utilizar una fuente de alimentación acorde al módulo Raspberry Pi con sus correspondientes valores de voltaje y corriente asignados al mismo para prevenir un mal funcionamiento, así como sobrecargas que podrían perjudicar al dispositivo de forma permanente.

- Mantener un espacio de trabajo ordenado e impecable para el manejo del módulo Raspberry Pi favorece a un mayor tiempo de vida útil con respecto al mismo, de modo que su uso sea adecuado y sin problemas para todos aquellos estudiantes que vayan avanzando en la carrera y tengan interés en trabajar con estos.

- Para trabajar con la cámara Raspberry Pi de manera más eficiente, se debe estar en un entorno bien iluminado y que la luz sea perceptible para la misma, de modo que ésta pueda mantener un correcto margen de tonalidades y que la visualización en tiempo real sea impecable.

- Al momento de utilizar el micrófono del módulo Raspberry Pi, se debe tomar en cuenta que al momento de hablar hacia éste, las palabras deben ser pronunciadas de manera clara y fuerte debido a que su nivel de reconocimiento no es tan bueno comparado con otros micrófonos de mejores prestaciones. Sin embargo, si se aplica en grabación y creación de archivos

de audio, se pueden encontrar configuraciones adicionales para poder mejorar la calidad y por lo tanto obtener un sonido más claro.

- Se debe tener especial cuidado en el manejo de los puertos GPIO, especialmente si se trata de manipularlos, puesto que entre ciertos pines existe un flujo ininterrumpido de corriente mientras el módulo permanezca conectado a la fuente de alimentación, aún si éste fue apagado mediante el "Shutdown" del sistema. Por esta razón pueden ocurrir cortocircuitos y perjudicar la integridad de la tarjeta Raspberry Pi.

- Cabe mencionar que una parte muy importante de trabajar con los módulos Raspberry Pi es que estos se mantengan actualizados a nivel de software; por desgracia dichas actualizaciones no se realizan de manera automática. Es por eso que mediante los comandos *"sudo apt-get upgrade"* y *"sudo apt-get update",* escritos dentro de la ventana de Terminal, se podrá actualizar el sistema. Es recomendable realizar este paso al menos una vez por mes.

- La programación realizada en Python necesita mantener un debido orden entre las funciones utilizadas dentro de la misma ya que en el momento de ejecutar el programa se tomará en cuenta dicha jerarquía de funciones y sus pertenencias en ésta para realizar lo planteado.

- Los módulos Raspberry Pi son una herramienta muy versátil, además de completa en múltiples formas, ofreciendo la posibilidad de trabajar a gusto con estos elementos. Sin embargo, si se desea un máximo rendimiento en los mismos, dependiendo del tipo de labor que se necesite aplicar, es aconsejable añadir un medio de enfriamiento para el dispositivo como un disipador de calor o un electroventilador compatibles.

  Mientras más trabajo se le ordene hacer, existe la posibilidad de presentarse un sobrecalentamiento que en el peor de los casos podría comprometer la parte integrada de la tarjeta.

- Investigar las diferencias en cuanto a estructura de comandos entre las distintas versiones de Python, y tener en cuenta que al momento de realizar la compilación y ejecución de un programa sin considerar los cambios entre éstos, ya sea de una menor versión a una mayor o viceversa, puede ocasionar errores de compilación y el programa no funcionará correctamente.

- Continuar de manera constante con la investigación y aplicación de los módulos Raspberry Pi, ya que han demostrado ser dispositivos que pueden llegar a cubrir muchas expectativas en diferentes áreas ligadas a la tecnología, haciendo posibles tareas complejas o cumpliendo funciones que tiempo atrás no se las creía realizables.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] Microchip Technology, «PIC 16F870-Microcontrollers and Processors,» 2 Mayo 2013. [En línea]. Available: http://www.microchip.com/wwwproducts/en/PIC16F870.

[2] Raspberry Pi Foundation, «Raspberry Pi Documentation,» Abril 2014. [En línea]. Available: https://www.raspberrypi.org/documentation/.

[3] W. Donat, Learn Raspberry Pi Programming with Python, Technology in Action, 2014.

[4] D. Norris, Raspberry Pi for the Evil Genius, New York: McGraw-Hill Education, 2014.

[5] Raspberry Pi Foundation, «About Us,» 2018. [En línea]. Available: https://www.raspberrypi.org/about/.

[6] Raspberry Pi Foundation, «Raspberry Pi Documentation - Datasheet,» 2016. [En línea]. Available: https://www.raspberrypi.org/documentation/hardware/computemodule/RPI-CM-DATASHEET-V1_0.pdf.

[7] A. Guerrero, «Raspberry Pi 3: Mas potencia y conectividad,» Facultad de Informática, Universidad Nacional de la Plata, 7 Marzo 2016. [En línea]. Available: https://sl.linti.unlp.edu.ar/category/hardware/.

[8] Raspberry Pi Foundation, «Raspberry Pi 3 Model B,» Febrero 2016. [En línea]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[9] Raspberry Pi Foundation, «BCM2837,» 2018. [En línea]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2837/README.md.

[10] Raspberry Pi Foundation, «GPIO: Models A+, B+, Raspberry Pi 2 and Raspberry Pi 3,» 2018. [En línea]. Available: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/README.md.

[11] Raspberry Pi Foundation, «Monitor Connection,» 2018. [En línea]. Available: https://www.raspberrypi.org/documentation/setup/monitor-connection.md.

[12] MIPI Alliance, «Camera and Imaging,» 2018. [En línea]. Available: https://mipi.org/specifications/camera-and-imaging.

[13] MIPI Alliance, «Display and Touch,» 2018. [En línea]. Available: https://www.mipi.org/specifications/display-interface.

[14] Raspberry Pi Foundation, «SD Cards,» 2018. [En línea]. Available: https://www.raspberrypi.org/documentation/installation/sd-cards.md.

[15] Raspberry Pi Foundation, «Raspbian,» 2018. [En línea]. Available: https://www.raspberrypi.org/documentation/raspbian/.

[16] PhpMyAdmin, «PhpMyAdmin: Bringing MySQL to the Web,» 2018. [En línea]. Available: https://www.phpmyadmin.net/.

[17] Real VNC Limited, «Real VNC,» 2018. [En línea]. Available: https://www.realvnc.com/es/connect/docs/faq/philosophy.html.

[18] Raspberry Pi Foundation, «VNC (Virtual Network Computing),» 2018. [En línea]. Available: https://www.raspberrypi.org/documentation/remote-access/vnc/.

[19] Python Software Foundation, «Tabla de Contenidos - Tutorial de Python 3.6.3 documentation,» 2017. [En línea]. Available: http://docs.python.org.ar/tutorial/3/real-index.html.

[20] OpenCV Team, «OpenCV,» 2018. [En línea]. Available: https://opencv.org/.

[21] NumPy Developers, «Numpy,» 2017. [En línea]. Available: http://www.numpy.org/.

[22] Julius Development Team, «Julius,» 2014. [En línea]. Available: http://julius.osdn.jp/en_index.php?q=index-en.html#about_julius.

[23] Raspberry Pi Foundation, «Camera Module v2,» 2018. [En línea]. Available: https://www.raspberrypi.org/products/camera-module-v2/.

[24] A. Malvino y D. Bates, «Teoría de Diodos,» de *Principios de Electrónica*, Aravaca, McGraw-Hill, 2007, p. 73.

[25] P. Membrey y H. David, Learn Raspberry Pi with Linux, New York: Springer, 2013.

[26] The Raspberry Pi Foundation, «Datasheet - Raspberry Pi Compute Module,» Octubre 2016. [En línea]. Available: https://www.raspberrypi.org/documentation/hardware/computemodule/RPI-CM-DATASHEET-V1_0.pdf.

[27] J. Goodacre, «Technology Preview: The ARMv8 Architecture,» Noviembre 2011. [En línea]. Available: https://www.arm.com/files/downloads/ARMv8_white_paper_v5.pdf.

[28] Broadcom Corporation, «VideoCore IV Architecture Guide,» 2013. [En línea]. Available: https://docs.broadcom.com/docs/12358545.

[29] M. Batz, «Raspberry Pi Geek,» Linux New Media USA, 2018. [En línea]. Available: http://www.raspberry-pi-geek.com/Archive/2016/17/Raspberry-Pi-3-Model-B-in-detail.

[30] The Raspberry Pi Foundation, «The Camera Module,» *The MagPi - The Magazine for Raspberry Pi Users,* vol. 1, nº 14, 2013.

[31] Raspberry Pi Foundation, «AN INTRODUCTION TO GPIO AND PHYSICAL COMPUTING ON THE RASPBERRY PI,» [En línea]. Available: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/. [Último acceso: 12 Enero 2017].

[32] Raspberry Pi Foundation, «Installing operating system images,» 2018. [En línea]. Available: https://www.raspberrypi.org/documentation/installation/installing-images/README.md.

[33] Raspberry Pi Foundation, «Installing Raspbian with NOOBS,» Projects - Raspberry Pi, 2018. [En línea]. Available: https://projects.raspberrypi.org/en/projects/noobs-install.

[34] A. Rosebrock, «Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi,» Pyimagesearch, 4 Septiembre 2017. [En línea]. Available: https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/.

[35] Prometec, «Crear un servidor en la Raspberry Pi - Tutoriales Arduino,» 2 Abril 2017. [En línea]. Available: https://www.prometec.net/raspberry-pi-servidor/.

[36] R. P. Foundation, «Julius 4.4.2 Reconocimiento de voz,» 2 Septiembre 2016. [En línea]. Available: https://www.raspberrypi.org/forums/viewtopic.php?f=76&t=158821&sid=0ddc0bc66c9f68a378cc74db0c166f1f. [Último acceso: 12 Enero 2017].

ANEXOS

ANEXO A: TEORÍA COMPLEMENTARIA

ANEXO B: HOJAS GUÍA PARA EL ESTUDIANTE

ANEXO C: HOJAS GUÍA PARA EL INSTRUCTOR

ANEXO D: INSTALACIÓN DE SOFTWARE Y LIBRERÍAS

ANEXO A
TEORÍA COMPLEMENTARIA

ANEXO A1: MÓDULOS RASPBERRY PI

Introducción

Especificaciones Mecánicas

ANEXO A2: ARQUITECTURA ARM

Introducción

Fundamentos de la Arquitectura ARMv8

ANEXO A3: ARQUITECTURA VIDEOCORE IV 3D

Introducción

ANEXO A4: WIFI Y BLUETOOTH

Descripción

Función de Radio Inhabilitado

ANEXO A5: CÁMARA RASPBERRY PI

Historia

Instalación de la Cámara

ANEXO A1: MÓDULOS RASPBERRY PI

INTRODUCCIÓN

El módulo computacional (CM1), módulo computacional 3 (CM3), y módulo computacional 3 Lite (CM3L) de Raspberry Pi son Sistemas en Módulos (SoMs), provistos de procesador, memoria, tarjeta multimedia Flash (para CM1 y CM3) y circuitos de energización secundarios. Estos módulos permiten a un diseñador obtener un aprovechamiento al máximo del hardware y software de las Raspberry Pi, elaborando sus propios sistemas personalizados. Además, estos módulos poseen interfaces IO (entrada/salida) extra sobre la placa y bajo la misma, los cuales están disponibles en las tarjetas Raspberry Pi modelo A/B, ampliando más opciones para el diseñador. [26]

El CM1 (**Figura A1**), contiene un procesador BCM2835 (usado en los modelos originales de Raspberry Pi y en los modelos Raspberry Pi B+). Además, una memoria RAM LPDDR2 de 512Mbytes y una tarjeta multimedia Flash (eMMC) de 4Gbytes. El CM3 contiene un procesador BCM2837 (usado en la Raspberry Pi 3), memoria RAM LPDDR2 de 1Gbyte y una tarjeta multimedia Flash (eMMC) de 4Gbytes. Finalmente, el módulo CM3L (**Figura A2**), tiene las mismas características que el CM3, con la diferencia que no incluye una tarjeta multimedia Flash, y los pines de la interfaz SD/eMMC están disponibles para el usuario con el fin de que pueda conectar su propio dispositivo SD/eMMC. [26]

Nótese que el procesador BCM2837 es una evolución del procesador BCM2835. Las únicas diferencias reales son que el BCM2837 puede soportar más RAM (hasta 1Gbyte) y la constitución del CPU de arquitectura ARM ha sido actualizada desde un simple núcleo ARM11 en BCM2835 a un Quad Core Cortex A53 con 512Kbytes dedicados de memoria caché 1.2 en BCM2837. Todas las interfaces IO y periféricos son las mismas y por lo tanto ambos chips son, en gran parte, compatibles en software y en hardware. [26]

Los pines de salida de los CM1 y CM3 son idénticos. Fuera de la mejora del CPU y el incremento en RAM, las otras diferencias significativas en hardware a tomar en cuenta son que el CM3 ha pasado a medir de 30mm a 31mm de alto, el suministro

de voltaje de la batería ahora puede extraer significativamente más energía bajo pesadas cargas del CPU, y también un mejor desempeño en el manejo de puertos (HDMI_HPD_N_1V8 y EMMC_EN_N_1V8), sustentados mediante las IO en lugar del procesador para aligerar su carga. [26]

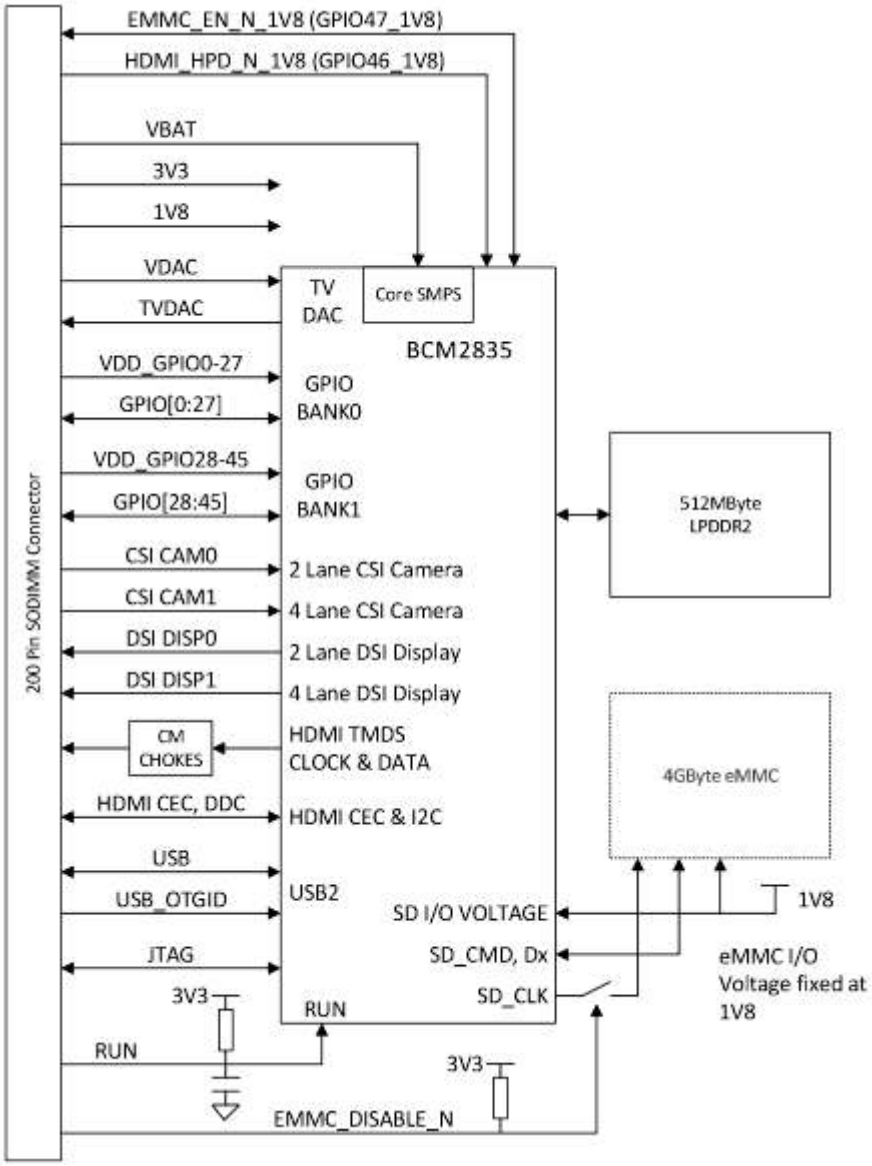


Figura A1: Diagrama de Bloques del CM1 [26]

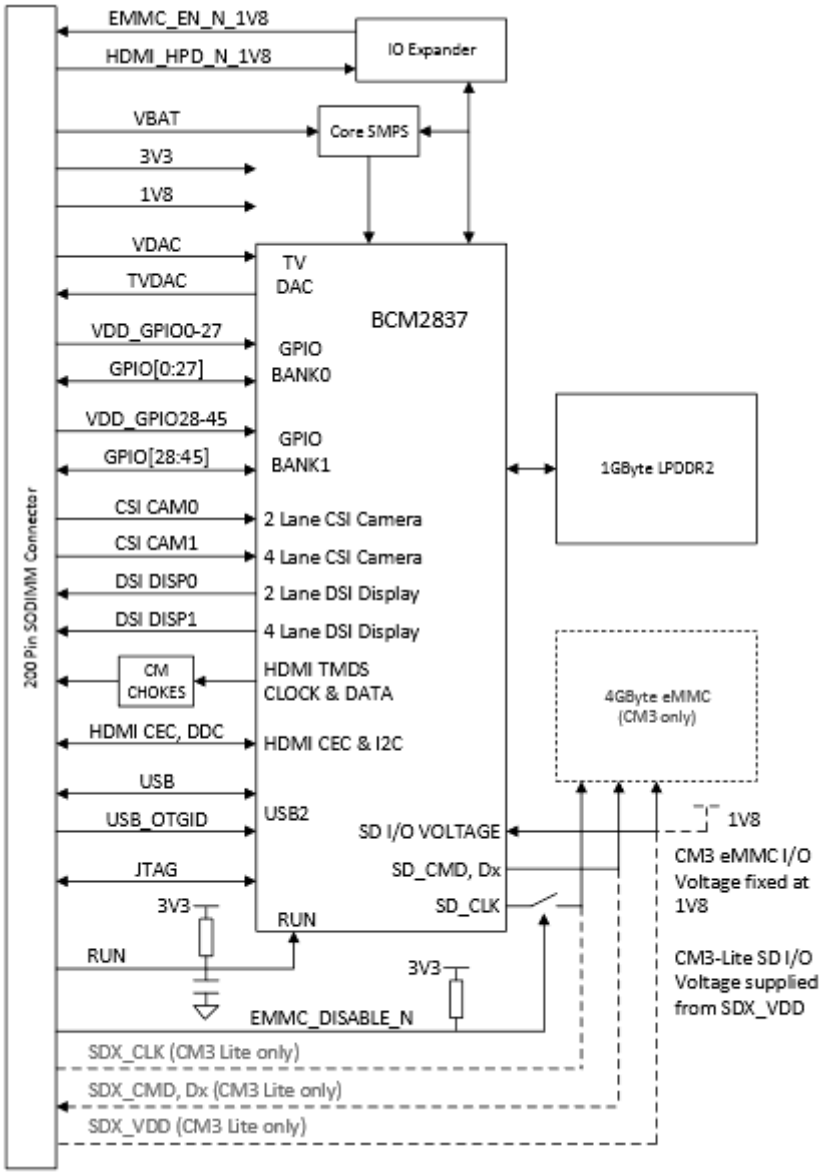


Figura A2: Diagrama de Bloques del CM3/CM3L [26]

Especificaciones Mecánicas

El módulo computacional conforma las especificaciones mecánicas JEDEC MO-224 para módulos de 200 pines DDR2 (1.8V) SODIMM (con la excepción de que los módulos CM3 y CM3L de la **Figura A4** son 31mm más altos a comparación de los 30mm del módulo CM1 de la **Figura A3**), y por lo tanto deberían funcionar con la mayoría de sockets DDR2 SODIMM disponibles en el mercado. [26]

El factor de forma SODIMM fue elegido como una manera de proveer las conexiones de los 200 pines, usando un estándar fácilmente disponible, además de conectores compatibles con manufacturas PCB, de bajo coste. [26]

- La altura máxima de componentes bajo el módulo computacional es de 1.2mm.
- La altura máxima de componentes sobre el módulo computacional es 1.5mm.
- El grosor de la placa PCB del módulo computacional es 1.0mm +/- 0.1mm.

Nótese que la ubicación y el arreglo de los componentes en el módulo computacional pueden estar sujetos a ligeros cambios con el tiempo debido a revisiones en consideración al costo y fabricación. Aun así, la altura máxima de los componentes y el grosor de la placa de circuitos se mantendrán según lo especificado. [26]

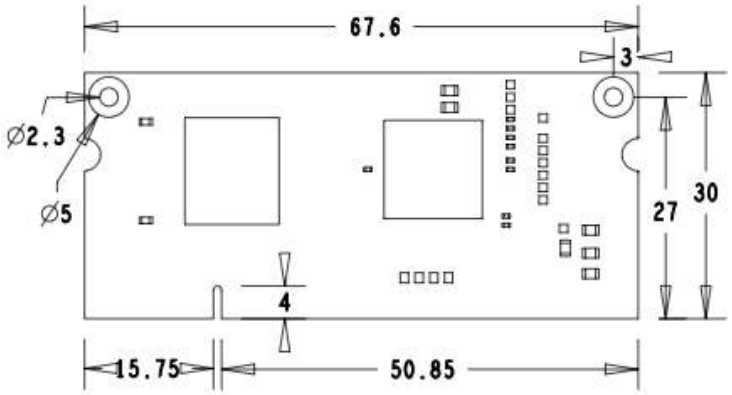


Figura A3: Especificaciones Mecánicas del CM1 [26]

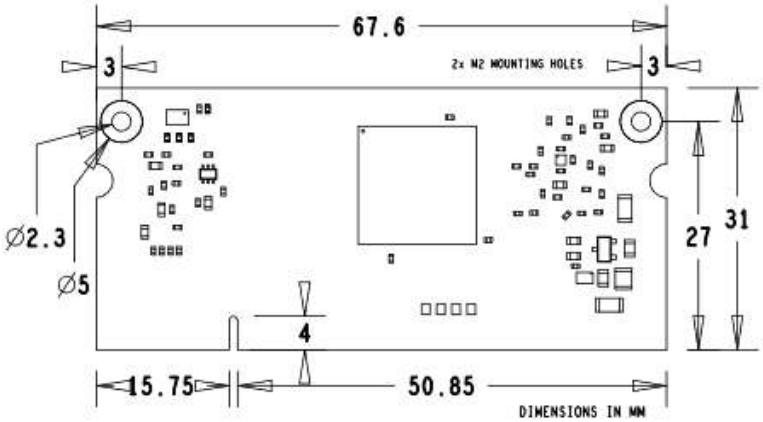


Figura A4: Especificaciones Mecánicas del CM3/CM3L [26]

ANEXO A2: ARQUITECTURA ARM

Introducción

La arquitectura ARM® RISC ha evolucionado en los últimos 20 años, ganando un amplio reconocimiento con la variante ARMv4 hasta la introducción de la familia de procesadores ARM7™. Fue diseñado desde el inicio para ser optimizado en base a un bajo consumo de energía. En ese entonces, no había coma flotante, ni instrucciones matemáticas complejas, ni SIMD (*Simple Instruction, Multiple DATA*), mucho menos la capacidad para dividir dos números enteros. Desde el 2011 y en la actualidad, toda la familia de procesadores ARM Cortex™ son diseñados usando la arquitectura ARMv7. [27]

El amplio mercado de aprobación de los procesadores ARM, ha empezado a extenderse mucho más allá de los dispositivos móviles (**Figura A5**), tanto que la arquitectura ARMv7 a pesar de ser totalmente consistente, ha sido separado en tres secciones que coinciden de mejor manera a los requerimientos del mercado como los entornos de "*Aplicación*" Sección-A, "*Tiempo Real*" Sección-R y "*Microcontroladores*" Sección-M. [27]

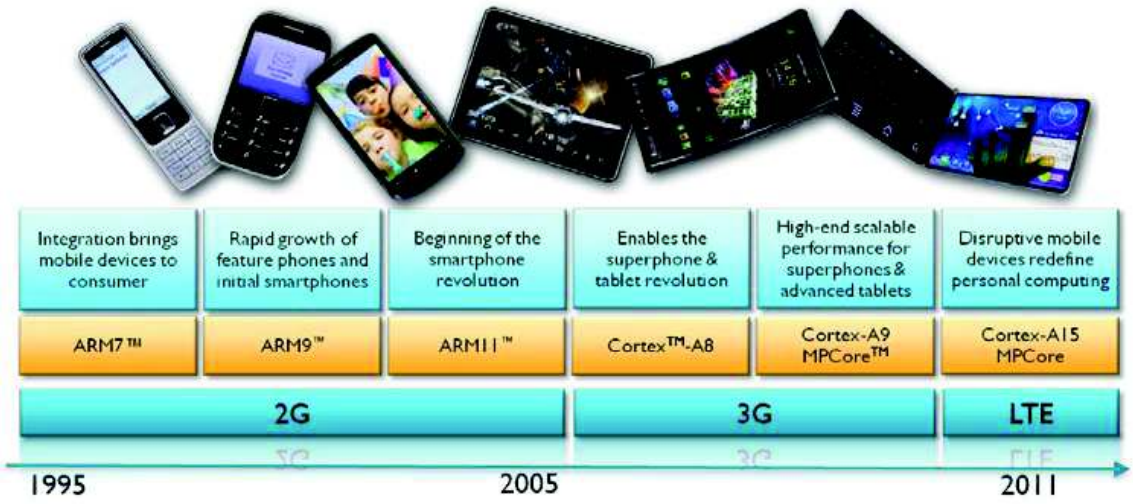


Figura A5: Evolución de la Arquitectura ARM [27]

Los procesadores Cortex, rápidamente se han convertido en la elección primaria en toda la industria. Dentro de la Sección-A, introducido por el Cortex-A8, el Cortex-A9 trajo el multiprocesamiento SMP (*Multiprocesamiento Simétrico*) a las masas, y

el Cortex-A5 lo trajo a dispositivos con conexión a Internet clase ARM926 de menor costo y de bajo consumo de energía. Además, Cortex-A5 comenzó la oportunidad para utilizar las últimas y más valiosas características de arquitectura ARM al más bajo precio puntual relacionado al Internet. Todos estos procesadores Cortex, son en esencia la base para la mayoría de los nuevos negocios de ARM. En octubre de 2011, ARM introdujo el procesador Cortex-A7 y el esquema de eficiencia energética big.LITTLE para cambiar nuevamente el enfoque de la industria móvil, con el propósito de entregar Smartphones de nivel excepcional con baterías de larga duración. [27]

Estaba claro para ARM, desde el nivel de aceptación de la arquitectura ARMv7 y en el creciente acompañamiento en el ecosistema asociado y en tiendas de aplicaciones, que cualquier nueva mejora a la arquitectura tendría que ser totalmente compatible. De igual manera, estaba claro que se debía contar con soporte para cualquier migración hacia una nueva arquitectura. La nueva arquitectura debía traer una ventaja significativa sobre la arquitectura existente, aún cuando el software ejecutado no haga uso de ninguna de las nuevas características. [27]

Históricamente en la industria de la computación, tales mejoras han sido cumplidas añadiendo nuevas funcionalidades que superen lo antiguo. Intentos en la industria por remover problemas legales y empezar de cero quedaron atrás con la llegada de nuevas arquitecturas, intentando solucionar los problemas de lo descontinuado. En consecuencia, el fundamento crítico de ARMv8 había sido establecido. [27]

Los dispositivos basados en procesadores ARM habían sido observados siguiendo las capacidades del mercado de los PC hace unos cuantos años. Recientemente, el perfil de desempeño de los dispositivos basados en procesadores Cortex-A9, usados en casi todos los dispositivos del tipo Tablet, ha establecido incuestionablemente que el rendimiento de una solución ARM puede ofrecer la experiencia que la gran mayoría de usuarios de PC requieren con un consumo muy bajo de energía. Este punto de rendimiento significa que ARM está siendo aceptado por muchas otras clases de dispositivos, con mayor notoriedad en dispositivos empresariales y servidores donde el consumo de energía por unidad de rendimiento es aumentado por dispositivos basados en ARM. [27]

También existen algunas tendencias básicas que han avivado requerimientos hacia la siguiente versión de arquitectura como se observa en la **Figura A6**. Sistemas en un Chip (SoC) se han vuelto más y más integrados. Las tecnologías de fabricación subnanométrica permiten un inimaginable nivel de integración en un solo dispositivo. Tales dispositivos tienen el desempeño y la necesidad de asistir software más complejo. La demanda de las memorias para sostener estas aplicaciones está en continuo crecimiento. La complejidad de una sola aplicación está empezando a necesitar la dirección de más memoria de la que puede ser dirigida por una arquitectura simple de 32 bits. [27]

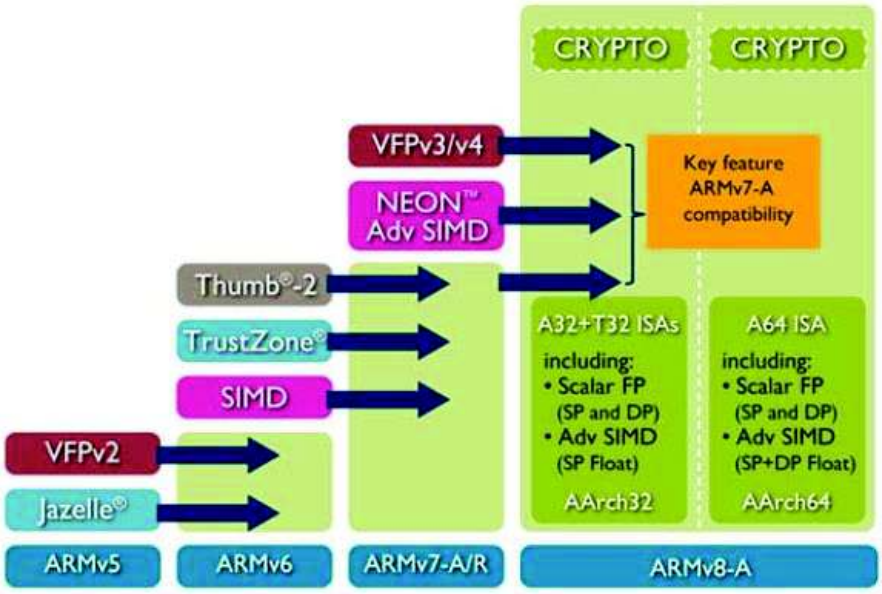


Figura A6: Conjunto de mejoras añadidas con cada generación [27]

ARMv8 hasta el día de hoy es uno de los más extensos programas a ser manejado dentro de ARM. [27]

Fundamentos de la Arquitectura ARMv8

El primer aspecto a tomar en cuenta acerca de la arquitectura ARMv8, hasta este punto, es que ha sido definida por los mercados de procesadores como un perfil Cortex-A. La arquitectura ARMv7 aún tiene muchas generaciones por delante, dirigiendo los mercados de ámbito "*Tiempo Real*" y "*Microcontroladores*". De hecho, muchos de los mercados correspondientes a la Sección-A, que necesiten una extensión más allá del espacio de direccionamiento de 32 bits, están siendo direccionados por el Cortex-A15 con su capacidad de LPAE (*Large Phisical*

Address Extensions). LPAE define un formato de cuadro de página para permitir el mapeo de páginas virtuales a un espacio de direccionamiento físico de 40 bits. Esto fue hecho de tal forma que nos permita cubrir tanto el extenso espacio de direccionamiento físico como el virtual. [27]

Como parte fundamental a ARMv8, debe serlo también el nuevo set de instrucciones conocido como A64; para que la codificación de instrucciones sea permitida al utilizar una aplicación en el momento de usar una máquina de 64 bits. ARM tomó la decisión para introducir 64 bits, mediante un nuevo conjunto de instrucciones, antes que la extensión de un set de instrucciones existentes por múltiples buenas razones. Lo más notable, y probablemente no tan sorprendente, sea que podríamos entregar un nuevo conjunto de instrucciones independientes para ejecutar códigos en modo de bajo consumo que añadiendo instrucciones a un conjunto de estos existente.

Por supuesto, por razones de compatibilidad, se continúa dando soporte a toda la mecánica del ARMv7 en la nueva arquitectura ARMv8, pero al momento de correr un software de 64 bits, esta parte del sistema no está siendo utilizada, y el área de residuos complejos que ha sido acumulado no necesita estar activo cuando se está corriendo en la arquitectura ISA de 64 bits, a diferencia de otras arquitecturas donde la extensión de 64 bits fue simplemente añadida a la complejidad histórica y legado de su modo de 32 bits.

La nueva arquitectura ISA se trazó con los años de experiencia de construir diferentes implementaciones de microarquitectura, por lo que nuevamente está definido que estos nuevos procesadores pueden ser optimizados para bajo consumo de operación de forma más sencilla, siendo una oportunidad desde la primera máquina ARMv4 que resultó en los ahora legendarios procesadores ARM7 de bajo consumo. [27]

La nueva arquitectura ARMv8-A también presentó la oferta que permitirá optimizar el modelo de software, así como mejorar la utilización del silicón en sus procesadores, ofreciendo la oportunidad de bajar aún más el consumo de energía del sistema. [27]

Claramente, esta nueva arquitectura necesita permitir software que dé soporte a la herencia de 32 bits, pero a diferencia de los recientes conjuntos de instrucciones, fue decidido que las transiciones entre los estados de ejecución entre 32 bits y 64 bits solamente deberían ocurrir en límites de excepción conocidos como interprocesamiento, y no mediante el ARM tradicional para Internetworking. Esto permite una jerarquía entre los múltiples niveles de privilegios de ejecución soportados. [27]

ANEXO A3: ARQUITECTURA DE VIDEOCORE IV 3D

Introducción

La segunda generación de sistemas 3D en VideoCore® VI es un gran paso desde la primera generación de hardware 3D en VideoCore III. La arquitectura VideoCore IV 3D es escalable, basado en múltiples procesadores especializados en flotantes-puntos de sombra denominados QPUs (**Figura A7**) [28].

Para evitar el ancho de banda de memoria Escalante en altas resoluciones y alto rendimiento, la nueva arquitectura utiliza renderización de pixeles basado en cuadros. Esto reduce el ancho de banda de regulación de capas por un orden de magnitud comparado a una renderización de forma inmediata, y permite un antialiasing multimuestreo 4 veces mayor para ser usado con una baja penalización en rendimiento o en consumo de memoria. Un sistema totalmente configurado soportará resoluciones de 720p con un multimuestreo 4x en una buena tasa de cuadros con contenido de juegos de la siguiente generación. [28]

Las especificaciones más importantes para un sistema totalmente configurado a 250 MHz son:

- 25M de triángulos renderizados por segundo.
- 1G de pixeles por segundo con texturización bilineal simple, sombreado simple y 4x de multimuestreo.
- Soporta 16x máscaras de cobertura antialiasing para renderización en 2D a una tasa completa de pixeles.

- Resolución estándar de 720p con 4x multimuestreo.
- Soporta una renderización HDR de 16 bits.
- Soporte completo de OpenGL-ES 1.1/2.0 y OpenVG 1.1

El hardware VideoCore IV 3D es autónomo y altamente automatizado, requiriendo poco ancho de banda de procesamiento o intervención en tiempo real por parte de drivers de software. Esto, en conjunto con la escalabilidad, hace una arquitectura 3D adecuada para su uso en una gran variedad de sistemas SoC. [28]

La escalabilidad es proporcionada principalmente por múltiples instancias de un procesador de sombreado de punto-flotante de propósito especial, denominado un procesador Quad (QPU). El QPU es un procesador SIMD de 16 vías. Cada procesador tiene dos vectores de punto-flotante ALUs, el cual lleva operaciones tanto múltiples como no múltiples en paralelo con una latencia de ciclos de instrucción simple. Internamente, el QPU es un procesador SIMD de 4 vías multiplexado 4 veces sobre 4 ciclos, haciéndolo particularmente adecuado para la transmisión del procesamiento de cuadros de pixeles. [28]

Los QPUs están organizados en grupos de hasta 4, denominados particiones, los cuales comparten ciertos recursos comunes. Cada partición comparte un caché de instrucciones, una Unidad de Funciones Especiales, una o dos Unidades de Búsqueda de Texturas y Memoria (TMU), y unidades de interpolación para la interpolación de variantes (VRI). [28]

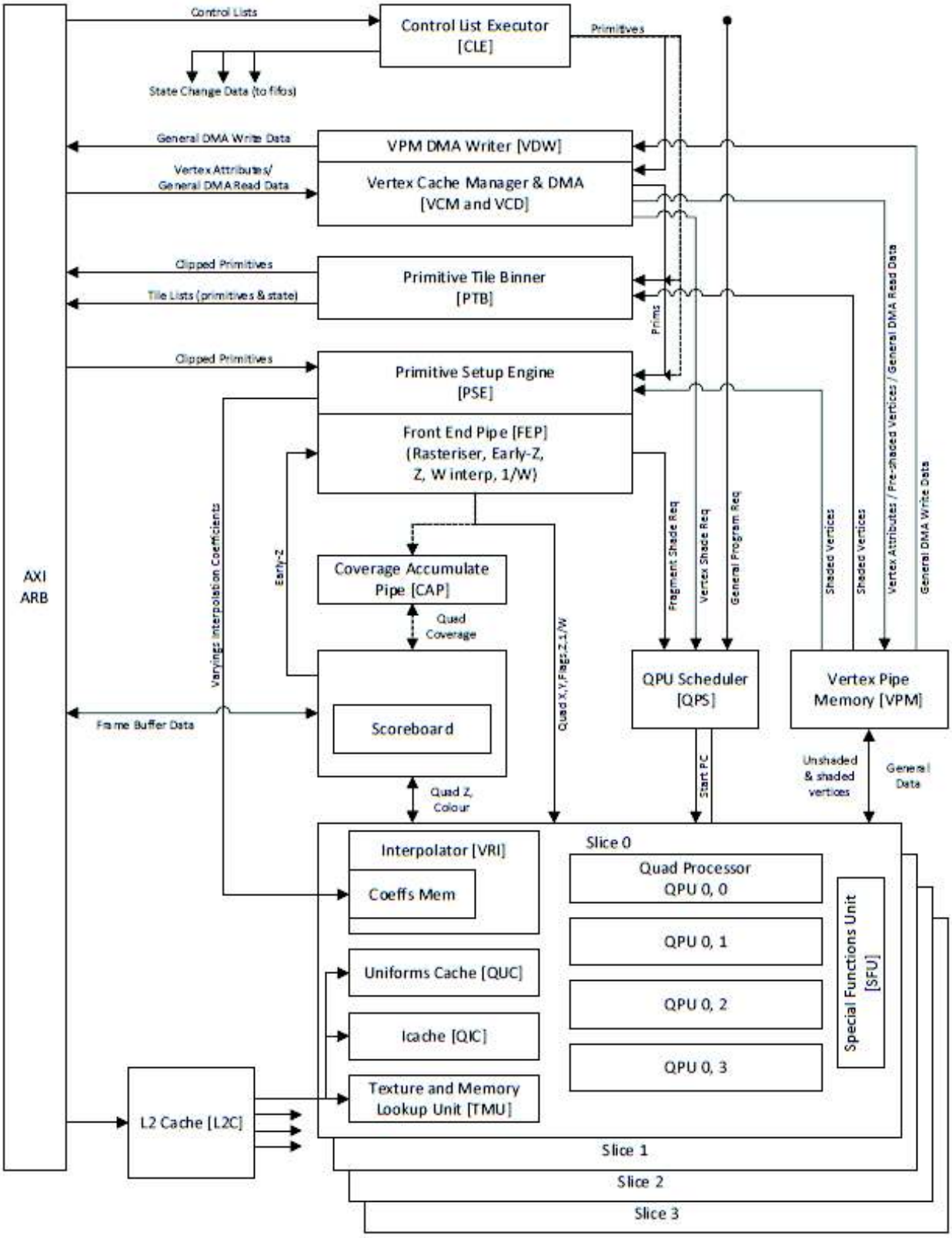


Figura A7: Diagrama de Bloques del VideoCore® IV 3D [28]

ANEXO A4: WIFI Y BLUETOOTH

Descripción

El kit combinado de Broadcom que incluye WiFi/Bluetooth, correspondiente al BCM 43438, transmite solamente en el espectro de 2.4 GHz y provee la función de WiFi (estándar 802.11b/g/n con un rendimiento de red de hasta 72.2Mbps), Bluetooth 4.1 (de bajo consumo), e incluso un receptor de radio FM el cual no se encuentra habilitado en el sistema Pi. WiFi y Bluetooth son implementados en un chip con capacidades completamente independientes. La única conexión entre estas dos funciones tiene que ver con la negociación por el acceso a la antena. El chip soporta solamente una antena, por tal motivo no mantiene una configuración MIMO. [29]

Función de Radio Inhabilitada

El chip BCM43438 de Broadcom posee un módulo de radio FM; sin embargo, no se puede activar esta función en la Raspberry Pi 3. El pin de la antena necesario para su activación, se encuentra en una ubicación inaccesible en la parte inferior del chip. El pin está rodeado por otros pines mucho más importantes que definitivamente requieren conexión. [29]

MIMO (*Multiple Input, Multiple Output*) es un proceso que hace uso de múltiples antenas de emisión y recepción para la comunicación inalámbrica. Los sistemas MIMO pueden transmitir considerablemente más información por Hertz de ancho de banda, usado a comparación de un sistema de antena única. [29]

ANEXO A5: CÁMARA RASPBERRY PI

Historia

Desde su primer lanzamiento, como se muestra en la **Figura A8**, la Raspberry Pi ha tenido un puerto en su estructura para acoplar una cámara al GPU (la unidad de procesamiento gráfico VideoCore 4 que forma parte de la tarjeta). Esta conexión

utiliza el protocolo eléctrico CSI-2, el cual es un estándar utilizado en la mayoría de equipos móviles. Dicha conexión es extremadamente rápida, mediante la cual la Raspberry Pi es capaz de enviar imágenes con resolución de 1080p (1920x1080 x10bpp) a 30 FPS, o a una menor resolución con FPS mucho mayores.

Siempre se ha propuesto, en algún punto, lanzar un módulo cámara que pudiera usar esta conexión, con la habilidad de transmitir datos de video en alta velocidad a través del GPU sin ninguna clase de interacción con el procesador ARM, lo que haría más eficiente el trabajo de la cámara incluso más que una cámara acoplada por medio de USB. Además, también podría habilitar el uso de la habilidad del GPU para codificar video en formato H264, o imágenes JPEG en hardware. [30]

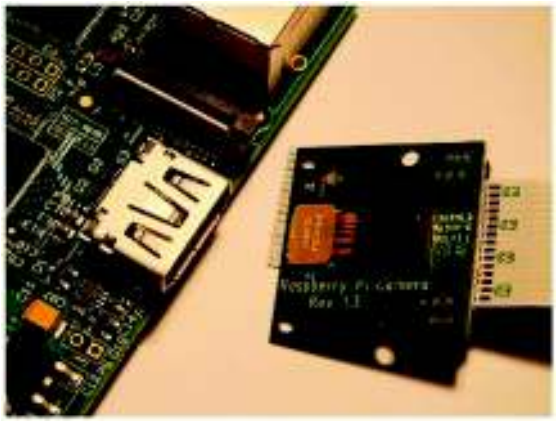


Figura A8: Cámara Raspberry Pi junto al puerto de conexión CSI [30]

Resulta que la producción de un pequeño PCB, como el de la cámara, no es tarea fácil. El prototipo fue rediseñado con motivo de sustraer algunos componentes innecesarios; pero más importante, mover el lente de la cámara, usado para temporización, a la propia PCB. [30]

Las pruebas de compatibilidad electromagnética (EMC) han mostrado que el temporizador de 25MHz provisto por el GPU causaba demasiada interferencia al momento de pasar por el cable a la PCB. Al añadir un cristal a la misma PCB, dicha interferencia se detuvo. Un par de placas se diseñaron más tarde para ayudar con la fabricación de la línea de producción y pruebas; y eventualmente, cerca de un año después de los primeros prototipos, las placas de producción estaban listas. [30]

Mientras tanto, la labor ha ido progresando con el propósito de escribir un par de aplicaciones para hacer uso de la cámara; así como actualizar el firmware del GPU para la compatibilidad con la cámara, y mejorar los ajustes de la cámara como los ajustes básicos que ya se encontraban en su lugar, pero con unos pocos defectos [30]. Los ajustes de la cámara son un reto complicado, los cuales han sido cubiertos por parte del sitio web de Raspberry Pi.

Instalación de la cámara

Este tipo de cámaras poseen sensibilidad a la estática, por lo que se debe tomar como prioridad la manipulación de la misma, la cual puede ser sujetada por medio de su placa para no correr riesgos de avería. [30]

Solamente se debe realizar un par de conexiones, en las cuales el cable debe ir acoplado tanto en la placa de la cámara como en la tarjeta Raspberry Pi. Se debe considerar la posición correcta en la que estará acoplado el cable; de otro modo, no funcionará.

Sobre la placa de la cámara, la marca azul que se encuentra en el extremo del cable, debe ir en sentido contrario a ésta, y en la tarjeta Raspberry Pi la misma marca ubicada en el otro extremo del cable debe ir en sentido hacia el puerto Ethernet, como se indica en la **Figura A9**. A partir de este punto, solamente hay que empujar un poco el cable en las correspondientes ranuras y luego presionar gentilmente las pestañas incluidas en dichas ranuras. [30]

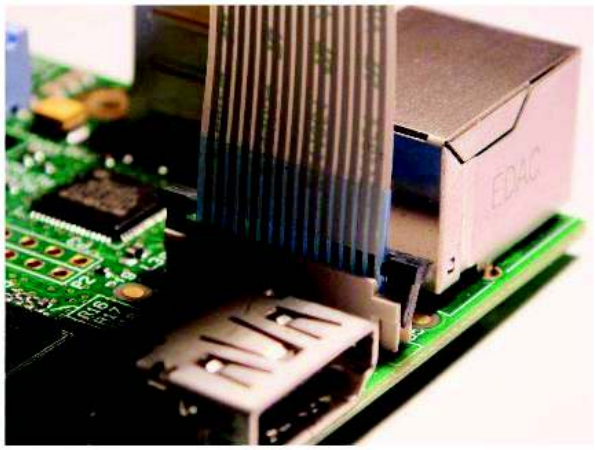


Figura A9: Conexión del cable al puerto CSI de la Raspberry [30]

ANEXO B
HOJAS GUÍA PARA EL ESTUDIANTE

ANEXO B1

HOJA GUÍA PARA EL ESTUDIANTE PRÁCTICA N°1

"Procesamiento de imágenes mediante el módulo Raspberry Pi 3"

ANEXO B2

HOJA GUÍA PARA EL ESTUDIANTE PRÁCTICA N°2

"Procesamiento de base de datos utilizando módulos Raspberry Pi 3"

ANEXO B3

HOJA GUÍA PARA EL ESTUDIANTE PRÁCTICA N°3

"Procesamiento de voz y control de puertos GPIO utilizando módulos Raspberry Pi 3"

ANEXO B1

HOJA GUÍA PARA EL ESTUDIANTE PRÁCTICA N°1

"Procesamiento de imágenes mediante el módulo Raspberry Pi 3"

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

HOJA GUÍA PARA EL ESTUDIANTE

PRÁCTICA 1

1. TEMA: Procesamiento de imágenes mediante el módulo Raspberry Pi 3.

2. OBJETIVOS

- Familiarizar al estudiante con el uso del módulo Raspberry Pi.
- Conocer la utilidad de trabajar con la cámara del módulo Raspberry Pi y conocer el alcance de sus prestaciones en el ámbito de la electrónica.
- Realizar un programa en Python que permita el reconocimiento de tres colores utilizando la cámara integrada en el módulo Raspberry Pi.

3. TRABAJO PREPARATORIO

- Consultar los principales puntos acerca de la *"Raspberry Pi Camera"* a ser utilizada (Rev 1.3), relacionados con su hardware y la calidad de imagen que ésta puede procesar.
- Investigar las líneas de comandos para la comprobación del funcionamiento de la cámara Raspberry Pi v1.3, tomando en cuenta los parámetros personalizables para obtener distintos resultados en cuanto a la salida de imágenes y video procedentes de la misma.
- Buscar acerca de los Códigos de Colores y tomar al menos siete colores básicos y sus diferentes tonalidades en código RGB, esencial para el reconocimiento.

4. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRACTICA

- El desarrollo de la práctica será realizado mediante el uso del software de programación Python, el cual viene incluido en el SO de Raspbian en el módulo Raspberry Pi.
- Una vez dentro del software, se elaborará el programa que permitirá reconocer los tres colores elegidos, desde la importación de las librerías necesarias a la activación y configuración de la cámara e incluso la declaración de los eventos de respuesta necesarios para comprobar su buen funcionamiento.
- Al terminar de realizar la programación correspondiente, se ejecutará el programa y se llevará a cabo la revisión y comprobación de la inicialización de la cámara, así como el reconocimiento del color designado.

ANEXO B2

HOJA GUÍA PARA EL ESTUDIANTE PRÁCTICA N°2

"Procesamiento de base de datos utilizando módulos Raspberry Pi 3"

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

HOJA GUÍA PARA EL ESTUDIANTE

PRÁCTICA 2

1. TEMA: Procesamiento de base de datos utilizando módulos Raspberry Pi 3.

2. OBJETIVOS

- Relacionar al estudiante con el lenguaje de programación Python y su interfaz gráfica.
- Comprender el funcionamiento de una base de datos y su software complementario.
- Realizar un programa en Python que permita la conexión y administración de una base de datos.

3. TRABAJO PREPARATORIO

- Investigar las diferencias y semejanzas de funcionamiento y sintaxis de comandos entre Python 2 y Python 3.
- Investigar acerca de phpMyAdmin, Apache2 y MySQL-Server, su funcionamiento, su instalación y la relación que mantienen estos tres programas.
- Familiarizarse con la creación de las bases de datos utilizando phpMyAdmin, apoyándose en fuentes didácticas y observando los parámetros más comunes al momento de elaborarlas.
- Generar archivos de audio (.WAV) utilizando cualquier software de grabación. Dichos audios serán utilizados como mensajes de aviso anexos según los parámetros considerados en el punto anterior (los más comunes), dentro de la elaboración de la base de datos.

4. DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA

- Crear una nueva carpeta dentro del directorio */home/pi*, en la cual se crearán una hoja de programación de Python llamada Base.py y una carpeta llamada audios en donde se colocarán los audios solicitados al estudiante.
- Ingresar a phpMyAdmin con las credenciales Usuario = root y Contraseña = root y crear una nueva base de datos con su respectiva tabla y parámetros dentro de la misma.
- Abrir la hoja de programación de Python 2 y realizar el código de conexión a la base de datos de phpMyAdmin y verificar su funcionamiento.
- Ya comprobada la conexión, realizar el programa de procesamiento de la base de datos; esto hacerlo a continuación del código de conexión.
- Al terminar el programa, ejecutarlo para llevar a cabo la comprobación de su funcionamiento.

ANEXO B3

HOJA GUÍA PARA EL ESTUDIANTE PRÁCTICA N°3

"Procesamiento de voz y control de puertos GPIO utilizando módulos Raspberry Pi 3"

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

HOJA GUÍA PARA EL ESTUDIANTE

PRÁCTICA 3

1. **TEMA:** Procesamiento de voz y control de puertos GPIO utilizando módulos Raspberry Pi 3.

2. **OBJETIVOS**

- Relacionar al estudiante con el lenguaje de programación Python y su interfaz gráfica.
- Realizar un programa en Python que controle el encendido y apagado de puertos GPIO mediante comandos de voz.

3. **TRABAJO PREPARATORIO**

- Consultar el funcionamiento de los pines GPIO del módulo Raspberry Pi, su configuración y sus librerías en Python.
- Consultar qué son las nomenclaturas BCM y BOARD de los pines GPIO y cómo utilizarlos al momento de realizar una sentencia, coloque ejemplos.
- Consultar acerca del software de reconocimiento de voz Julius y familiarizarse con su funcionalidad en módulos Raspberry Pi.
- Investigar la función, sintaxis y colocar un ejemplo de los siguientes comandos de Python: class(), parse(), self(), def(), if(),if_in, _init_,return(), os,system, print, sys.exit(), readline(), lower(), startswith(), strip(), split(), sys.argv, join(). capitalize y try_except.

4. **DESCRIPCIÓN DE LAS ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA**

- Crear una nueva carpeta dentro del directorio “/home/pi” que servirá posteriormente para el desarrollo de la práctica y luego copiar en ésta los archivos que se encuentran en la carpeta Reproductor ubicada en el mismo directorio.
- Modificar la base de datos de Julius (Archivos mediaplayer.voca y mediaplayer.dict) para que pueda reconocer los nuevos comandos en el control de los LEDs.
- Dentro de Nueva carpeta, crear una nueva hoja de programación en Python con el nombre control.py y realizar el programa para el control de los puertos GPIO mediante comandos de voz.
- Abrir una hoja de tipo script (.sh) con la cual se exportará la variable de entorno ALSADEV y se ejecutará el programa control.py.
- Ejecutar el archivo .sh y realizar la comprobación del correcto funcionamiento del programa.

ANEXO C
HOJAS GUÍA PARA EL INSTRUCTOR

ANEXO C1

HOJA GUÍA PARA EL INSTRUCTOR PRÁCTICA N°1

"Procesamiento de imágenes mediante el módulo Raspberry Pi 3"

ANEXO C2

HOJA GUÍA PARA EL INSTRUCTOR PRÁCTICA N°2

"Procesamiento de base de datos utilizando módulos Raspberry Pi 3"

ANEXO C3

HOJA GUÍA PARA EL INSTRUCTOR PRÁCTICA N°3

"Procesamiento de voz y control de puertos GPIO utilizando módulos Raspberry Pi 3"

ANEXO C1

HOJA GUÍA PARA EL INSTRUCTOR PRÁCTICA N°1

"Procesamiento de imágenes mediante el módulo Raspberry Pi 3"

1. **TEMA:** Procesamiento de imágenes mediante el módulo Raspberry Pi 3.

2. DESARROLLO DE LA PRÁCTICA

NOTA: La duración promedio de la presente práctica es 1 hora.

CÓDIGO COMPLETO

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import numpy as np

camera = PiCamera()
camera.resolution = (640,480)
camera.framerate = 50
camera.hflip = True

rawCapture = PiRGBArray(camera, size=(640,480))
time.sleep(0.1)

for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    image = frame.array
    blur = cv2.blur (image, (3,3))

    limites = [
        ([17, 15, 100], [50, 56, 200], 'R'),
        ([16, 76, 72], [30, 255, 210], 'Y'),
        ([34, 77, 23], [159, 218, 64], 'G')
        ]

    for (bajo, alto, color) in limites:
        bajo = np.array(bajo, dtype="uint8")
        alto = np.array(alto, dtype="uint8")
        mascara = cv2.inRange(blur, bajo, alto)

        image, contours,hierarchy = cv2.findContours(mascara,cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
        max_area = 0
        best_cnt = 1

        for cnt in contours:
            area = cv2.contourArea(cnt)
            if area > max_area:
                max_area = area
                best_cnt = cnt

            elif color == 'R':
                print "Color Rojo"
                continue

            elif color == 'Y':
                print "Color Amarillo"
                continue

            elif color == 'G':
                print "Color Verde"
                continue

        M = cv2.moments(best_cnt)
        cx,cy = int(M['m10']/M['m00']), int(M['m01']/M['m00'])

        cv2.circle(blur,(cx,cy),5,(255,255,255),-1)

        cv2.imshow("Colores", blur)

        key = cv2.waitKey(1) & 0xFF

        rawCapture.truncate(0)

    if key == ord ("q"):
        break
```

Se recomienda que al momento de realizar el programa, se tomen en cuenta el orden (tabulaciones) de las líneas de programación usadas en cada una de las funciones aplicadas, ya que al momento de correr el mismo, quizá advierta sobre errores de sintaxis o de no tener correspondencia hacia alguna función.

PREPARACIÓN DEL MÓDULO RASPBERRY

Conectar la fuente asignada al módulo para su energización. La fuente debe cumplir con los parámetros de voltaje y corriente indicados. (5V – 2Amp) para el correcto funcionamiento al conectar varios dispositivos USB. Su entrada de energía se encuentra en el lado izquierdo junto a la salida HDMI (Cable blanco), como se observa en la **Figura C1**.

Figura C1: Energización

Una vez energizado el módulo Raspberry, realizar la conexión del mismo a una de las pantallas del laboratorio usando la salida HDMI del módulo y el puerto VGA disponible en la parte posterior del monitor. Para esta conexión se debe utilizar el convertidor HDMI a VGA en conjunto con el cable VGA que se les proporcionará junto con la fuente de alimentación. Una vez realizada la conexión, cambiar la configuración predeterminada de entrada de video de DVI a VGA, de la siguiente manera:

En el lado derecho del monitor se encuentra el botón de **MENU**. Presiónelo y mostrará las opciones que se observan en la **Figura C2.**

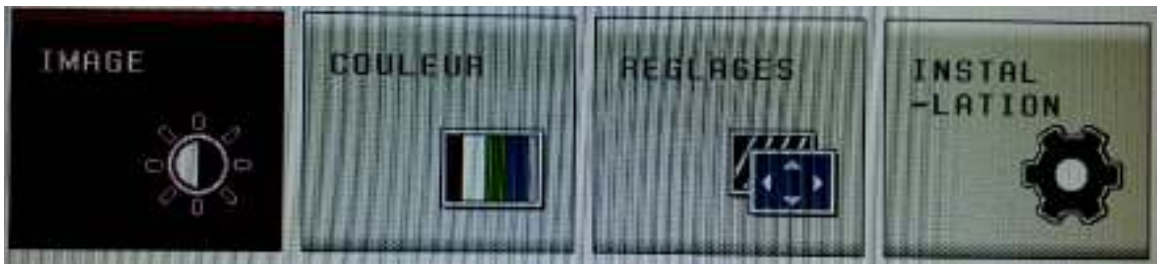


Figura C2: Configuración de Pantalla (Monitor)

Con los botones (flechas) del monitor, dirigirse hasta la opción **INSTALLATION** (**Figura C3**).

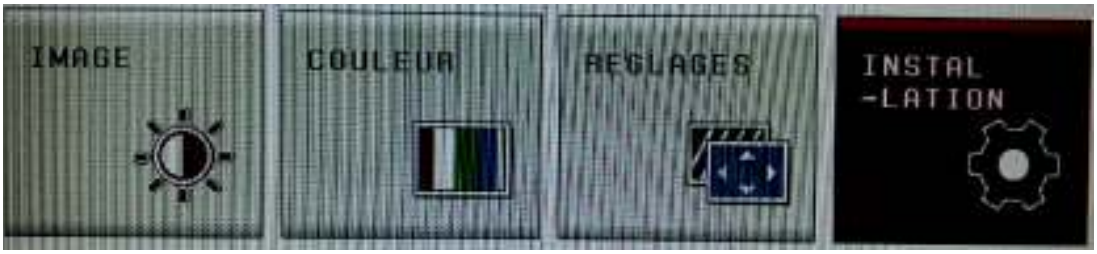


Figura C3: Opción INSTALLATION

Presionando el botón Menú, ingresar a las opciones de INSTALLATION y con las flechas dirigirse hasta **INPUT SELECT** (**Figura C4**).

FiguraC4: Installation

Con el botón Menú, ingresar a las opciones de INPUT SELECT y con las flechas seleccionar **VGA**. Cuando se selecciona la opción VGA, automáticamente se cambia su señal y se observa en el monitor la pantalla del Raspberry Pi (**Figura C5**).

Figura C5: Escritorio de Raspberry

Como punto importante a tomar en cuenta, tanto en esta práctica como en las siguientes, una vez realizada la configuración señalada anteriormente sobre la preparación del módulo para su utilización, se debe conectar los periféricos necesarios para trabajar en éste como son el teclado y el mouse. Para ello se recomienda utilizar dichos dispositivos que correspondan al CPU del lugar en el que se hayan ubicado en el laboratorio.

Desde el escritorio, abrir el Gestor de Archivos ubicado en la parte inferior izquierda de la pantalla con el ícono , el cual abrirá una ventana como la mostrada en la **Figura C6.**

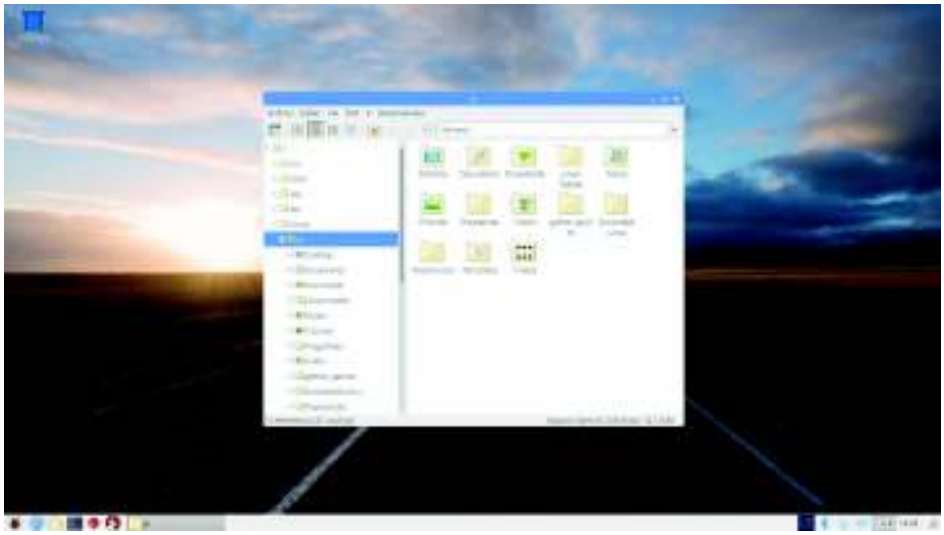


Figura C6: Ventana del Gestor de Archivos

En la parte derecha de la ventana del Gestor de Archivos, se puede ver múltiples carpetas como *Desktop, Documents*, entre otros. En ese mismo espacio se crea una carpeta para el programa con el nombre de *"Colores", dando clic derecho y seleccionando la opción de "crear nuevo"* y luego seleccionando *"Carpeta".* Se nombra a la misma y se generará la carpeta como se ve en la **Figura C7.**

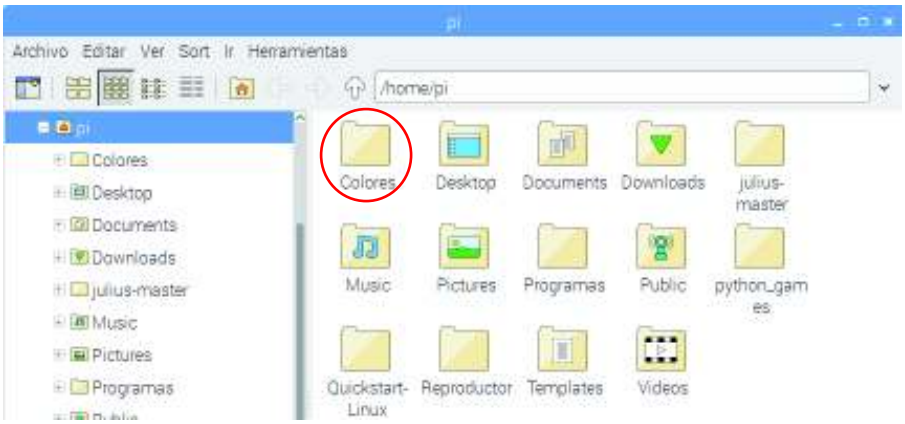


Figura C7: Visualización de la carpeta creada

Hacer doble clic en la carpeta para entrar en ésta. Una vez dentro, se procede a crear lo que será el archivo para realizar el programa. Se da clic derecho, y en la parte de "crear nuevo", elegir la opción *"Archivo vacío".* Nuevamente aparecerá el campo que permitirá dar nombre al archivo, como se muestra en la **Figura C8,** tomando en cuenta que al final del nombre se añade la nomenclatura ".py" para darle el formato de Python. Al seleccionar Aceptar, se visualiza el archivo dentro de la carpeta. **Figura C9**.

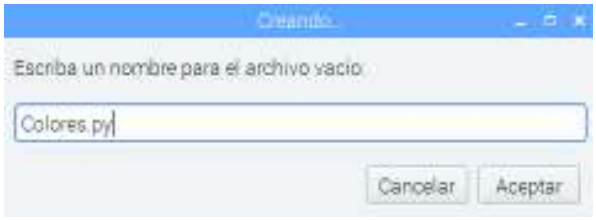


Figura C8: Creación del archivo "Colores.py"

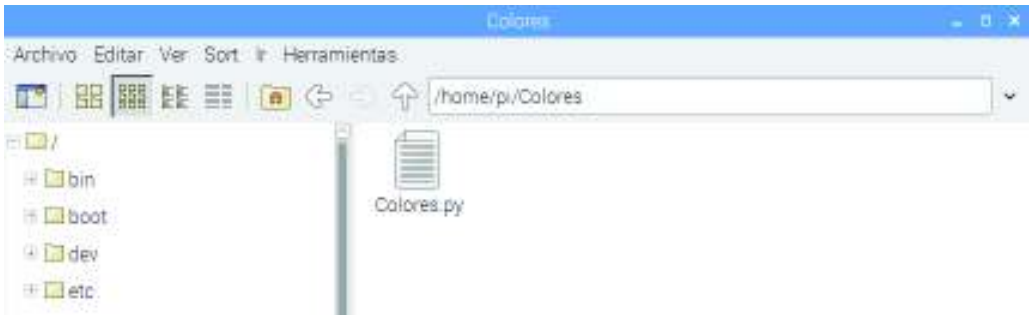


Figura C9: Archivo de "Colores.py" en la carpeta creada

Una vez generado el archivo *"Colores.py",* dar clic derecho y se mostrará un menú desplegable. Para abrir el nuevo archivo, se selecciona *"Abrir con…",* lo cual abrirá una nueva ventana con una lista de aplicaciones disponibles. Buscar la opción *"Programación",* dar clic y aparecerá una nueva lista en la que se selecciona el programa "*Python2 (IDLE)",* como se indica en la **Figura C10,** clic en *Aceptar.*

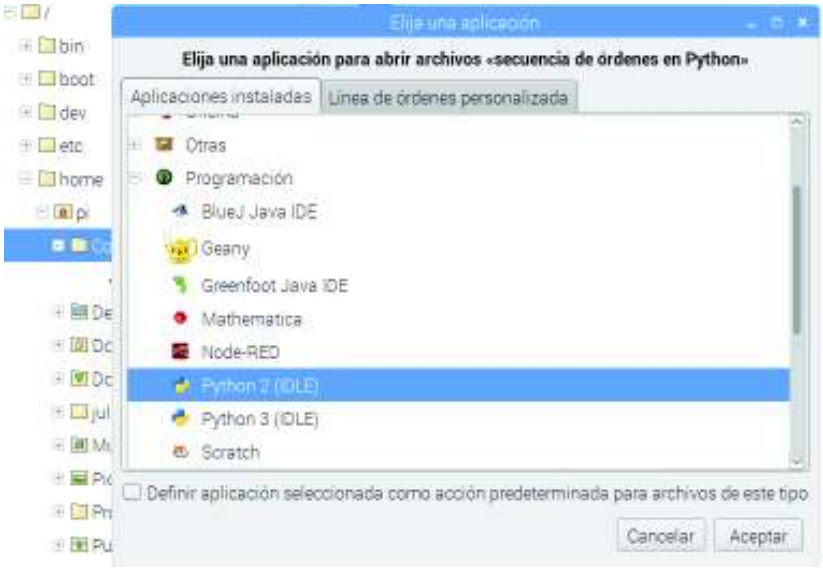


Figura C10: Opciones para abrir el archivo

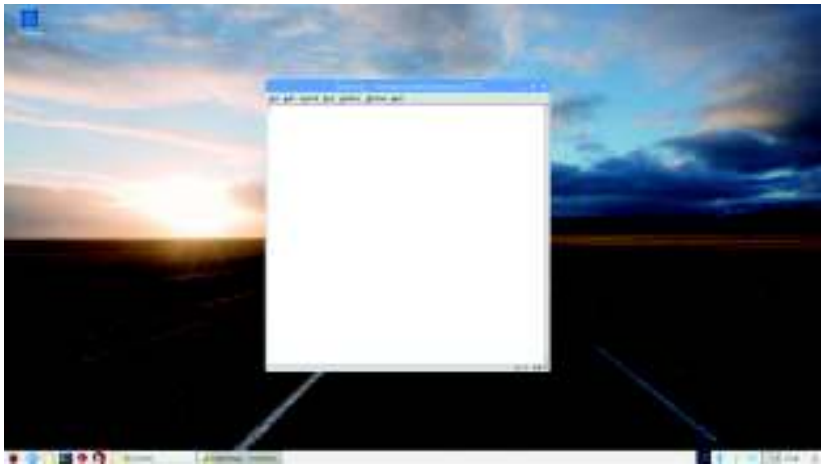


Figura C11: Visualización de la ventana de Python

Una vez abierto el software, como se muestra en la ***Figura C11,*** *se* procede a realizar el programa de reconocimiento de imágenes agregando las siguientes líneas:

```
from picamera.array import PiRGBArray
```

"Genera matrices RGB tridimensionales de cualquier captura RGB"

```
from picamera import PiCamera
```

"Se elige el dispositivo que se utilizará como periférico de entrada"

```
import time
```

"Permite el uso de sentencias de tiempo dentro del programa"

```
import cv2
```

"Se habilita el uso de funciones del programa OpenCV"

"OpenCV (Open Source Computer Vision Library) es un software de código abierto con librerías referentes a visión por computadora y aprendizaje de máquinas, la cual fue creada para proveer una infraestructura común dedicada a proyectos de la misma índole y acelerar el uso de la percepción artificial en productos comerciales – OpenCV.org"

```
import numpy as np
```

"Ejecuta el complemento matemático NumPy para Python"

"NumPy es un paquete fundamental para la informática científica en Python, en el cual se encuentran funciones como procesador de matrices N-dimensionales, funciones de radiodifusión sofisticadas, herramientas para codificación C/C++ y Fortran e incluso varias utilidades en cuanto a álgebra lineal, transformadas de Fourier, entre otras más capacidades numéricas – NumPy.org"

```
camera = PiCamera()
```
"Permite la configuración de los parámetros del dispositivo a usar (Pi Cámara)"

```
camera.resolution = (640,480)
```
"Se detalla la resolución a ser utilizada y visualizada por la cámara" – Escala variable.

```
camera.framerate = 50
```
"Se especifica la cantidad de fotogramas por segundo para el procesamiento de imagen – (framerate: frecuencia de capas)" – Valor Variable.

```
camera.hflip = True
```
"Gira la perspectiva de la cámara de manera horizontal – (hflip: horizontal flip)"

```
rawCapture = PiRGBArray(camera, size=(640,480))
```
"Realiza una captura de imagen en baja calidad como base para el proceso de recopilación de imágenes, detallando en ésta el formato de reconocimiento (PiRGBArray), el dispositivo predeterminado (camera) y el tamaño de la imagen (size)".

```
time.sleep (0.1)
```
"Tiempo de precalentamiento de la cámara"

```
for frame in camera.capture_continuous (rawCapture, format="bgr", use_video_port=True):
```
"En esta línea se define el análisis subsecuente de las capturas, tomando como fuente parámetros como son la imagen base (rawCapture), el formato de lectura de colores (bgr o RGB – Red, Green, Blue) y el puerto de video habilitado para el proceso (use_video_port) en cuyo caso es el de la Pi Camera.

```
image = frame.array
```
"Define la matriz analizada por el NumPy por medio del nombre (image) como la imagen fuente".

```
blur = cv2.blur (image, (3,3))
```
"Permite utilizar la función de OpenCV para filtrar los ruidos en la captura de imágenes (blur: borroso), por medio de un filtro normalizado, dándole a conocer los factores (image) y el tamaño de la matriz que lo constituye (3,3 – 3x3)".

```
limites = [
     ([17, 15, 100], [50, 56, 200], 'R'),
     ([16, 76, 72], [30, 255, 210], 'Y'),
     ([34, 77, 23], [159, 218, 64], 'G')
    ]
```
"En este punto se crea un parámetro (limites) que abarca los valores de los tres colores elegidos para el análisis (Rojo, Amarillo, Verde – para este caso) en formato RGB insertados

de la manera que se muestra junto con un comentario (R, Y, G) al final de cada color para su posterior uso en declaraciones".

```
for (bajo, alto, color) in limites:
```

"Se genera una función para el análisis de los valores insertados en el parámetro anterior".

```
bajo = np.array(bajo, dtype="uint8")
alto = np.array(alto, dtype="uint8")
```

"Define la tonalidad más alta y más baja de los colores a analizar por medio de los valores ingresados, los cuales entrarán en una matriz que será procesada por NumPy (np.array: numpy array – matriz numpy) a través del formato de datos descrito (dtype: data type; uint8: unsigned integer – entero sin signo (2^8 – 0 a 255))".

```
mascara = cv2.inRange(blur, bajo, alto)
```

"Llama a la función de OpenCV para condición de límite de imagen (cv2.inRange) en base a los parámetros borroso (blur), bajo y alto de la misma mediante el nombre designado (mascara)".

```
image, contours,hierarchy = cv2.findContours(mascara,cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

"Describe lo que se desea a la salida del análisis: imagen (image), contornos (contours) y jerarquía (hierarchy). Para lograrlo, se especifican parámetros para la detección de contornos mediante una función de OpenCV (cv2.findContours), tomando la condición de límites (mascara), el tipo de recuperación de contornos (cv2.RETR_LIST – para obtener los contornos en lista (de forma sencilla), sin dependencia de jerarquías) y el método de aproximación de contornos (cv2.CHAIN_APPROX_SIMPLE – para obtener un trazado de contorno con puntos de análisis en las cuatro esquinas del recuadro de la imagen para ahorro de memoria)".

```
max_area = 0
best_cnt = 1
```

"Parámetros para definir la detección del área máxima de los contornos (max_area) y almacenarlos como una identidad (best_cnt: best contour)".

```
for cnt in contours:
    area = cv2.contourArea(cnt)
```

"Función de OpenCV para la detección e identificación de los contornos (cv2.contourArea) definida mediante el área (area) y el contorno (cnt)".

```
    if area > max_area:
    max_area = area
    best_cnt = cnt
```

"Función condicionante para la detección del área y el contorno de la imagen analizada".

```
elif color == 'R': #Respuesta al color rojo
    print "Color Rojo"
    continue

elif color == 'Y': #Respuesta al color amarillo
    print "Color Amarillo"
    continue

elif color == 'G': #Respuesta al color verde
    print "Color Verde"
    continue
```

"Condiciones para obtener una respuesta de la detección de cada color por medio de los comentarios adjuntos a los códigos de colores descritos anteriormente en el parámetro (limites), cuya visualización se indicará en la ejecución del programa".

```
M = cv2.moments(best_cnt)
cx,cy = int(M['m10']/M['m00']), int(M['m01']/M['m00'])
```

"Describe la función de OpenCV para el análisis de características de los objetos detectados (cv2.moments), como su área o su centro de masa y la detección de centroides en ésta (cx = int (M['m10']/M['m00']); cy = int (M['m01']/M['m00']) – Calculados por la relación $C_x = \frac{M_{10}}{M_{00}}$ *y la relación* $C_y = \frac{M_{01}}{M_{00}}$ *)".*

```
cv2.circle(blur,(cx,cy),5,(255,255,255),-1)
```

"Función de OpenCV para la visualización gráfica de los centroides (cv2.circle) mediante los parámetros de imagen (blur, centroides (cx, cy), diámetro gráfico del centroide ((5) – valor variable), color del centroide gráfico ((255, 255, 255) – Blanco (valor variable)), relleno ((-1) – valor recomendado))".

```
cv2.imshow("Colores", blur)
```

"Función de OpenCV que permite abrir una ventana para visualizar lo que la cámara observa (cv2.imshow) con la opción de darle nombre a la ventana ("Colores") y habilitar el filtro de ruido en la misma (blur).

```
key = cv2.waitKey(1) & 0xFF
```

"Función de OpenCV para crear un enlace de respuesta con el teclado (cv2.waitKey) generando un parámetro con el nombre designado (key) con un valor de retorno dado en ASCII (0xFF).

```
rawCapture.truncate(0)
```

"Permite la reutilización de la salida para un nuevo procesamiento de las matrices"

```
if key == ord ("q"):
    break
```

"Condición que relaciona el parámetro anterior (key) para detener el proceso de reconocimiento de imagen mediante una tecla ("q"), terminando la ejecución del programa"

EJECUCIÓN DEL PROGRAMA

Luego de haber realizado el programa, se procede a guardarlo de tal manera que esté listo para su ejecución. Para guardar el proyecto, dirigirse a la pestaña *File* de la ventana de Python y seleccionar la opción *Save,* como se indica en la **Figura C12**.

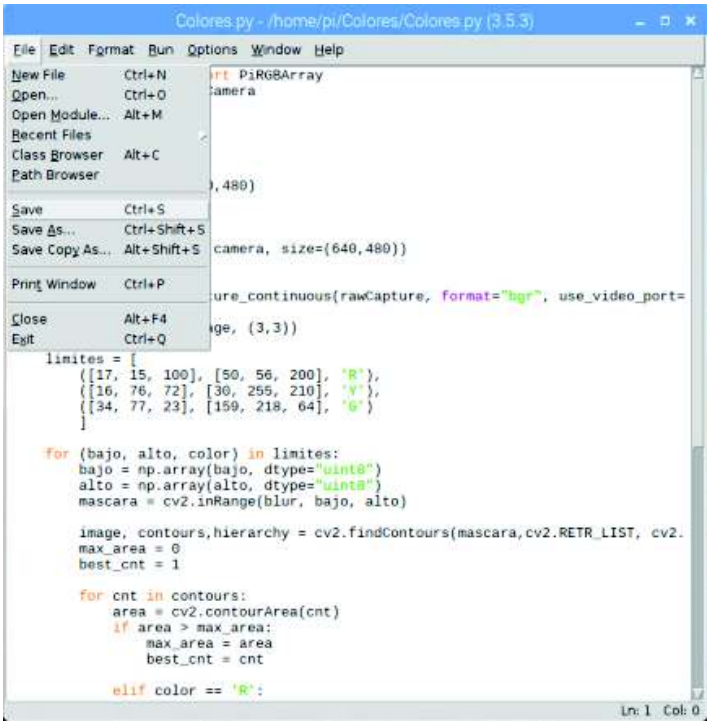


Figura C12: Opción de guardado

Una vez guardado el proyecto, continuar con su ejecución. En la misma ventana de Python, dirigirse a la pestaña *Run,* dar clic a la opción *Run Module,* como se ve en la **Figura C13** y observar que el programa empezará a reconocer los colores a medida que se ubiquen objetos de color rojo, verde o amarillo frente a la cámara.

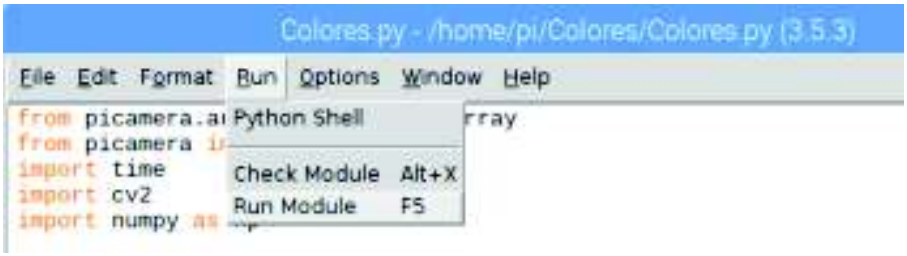


Figura C13: Opción de ejecución del programa

Para este caso, se puede observar el reconocimiento del color rojo (**Figura C14**), del color amarillo (**Figura C15**) y del color verde (**Figura C16**), comprobando que el programa está funcionando correctamente.

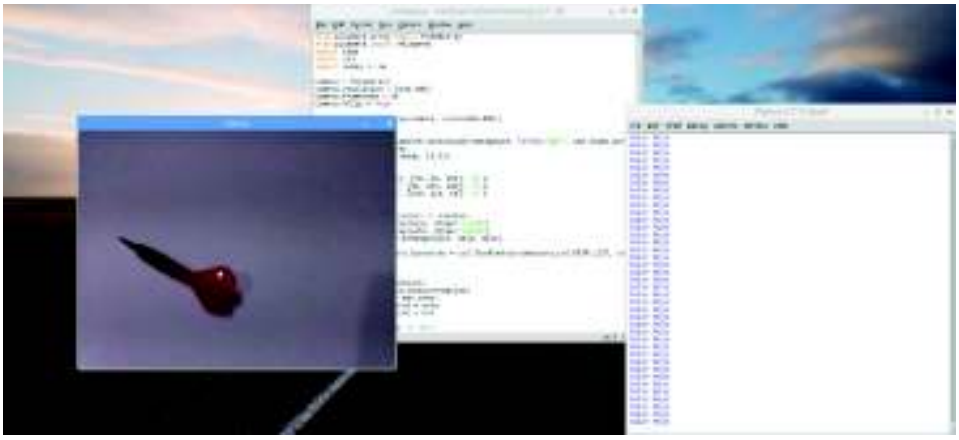


Figura C14: Detección del Color Rojo

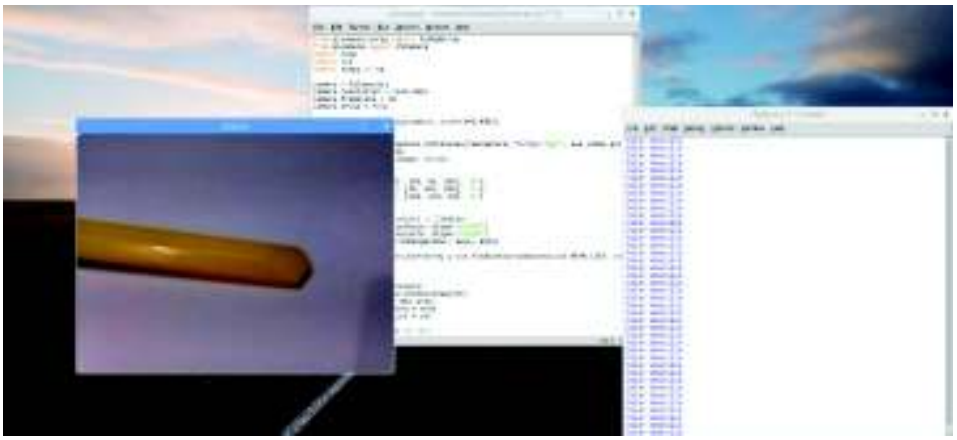


Figura C15: Detección del Color Amarillo

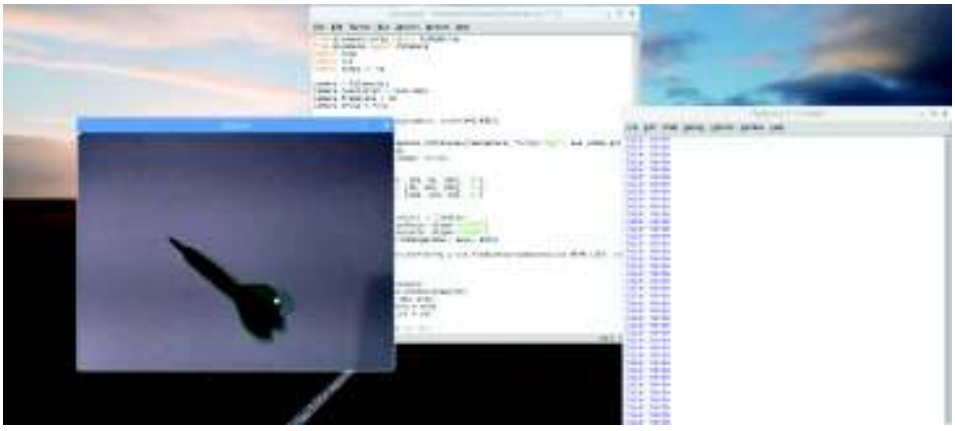


Figura C16: Detección del Color Verde

3. RECOMENDACIONES

- Al momento de ejecutar el programa es recomendable que la habitación tenga una buena iluminación para mejorar la etapa de reconocimiento del color, debido a que las cámaras Raspberry incluidas no tienen una resolución tan clara como las tradicionales cámaras Web.

- En caso de presentarse problemas al identificar un color, se pueden realizar estas dos opciones.

    - Colocar un fondo blanco (hoja de papel) detrás del elemento a reconocer puede ser de gran ayuda para que el reconocimiento sea más preciso.
    - Buscar una nueva escala de tonalidades del color y reemplazarla en el parámetro "límites", descrito anteriormente, hasta que se solucione el problema.

- Como parte importante del uso de los módulos Raspberry Pi, se aconseja realizar la actualización del software de los mismos al menos una vez al mes ya que el sistema y sus librerías están en constante desarrollo. Para ello, escribir las siguientes líneas en LXTerminal, el cual se encuentra junto al Gestor de Archivos con el ícono 

    - $ sudo apt-get upgrade
    - $ sudo apt-get update

ANEXO C2

HOJA GUÍA PARA EL INSTRUCTOR PRÁCTICA N°2

"Procesamiento de base de datos utilizando módulos Raspberry Pi 3"

1. **TEMA:** Procesamiento de Base de Datos utilizando módulos Raspberry Pi 3.

2. DESARROLLO DE LA PRÁCTICA

CÓDIGO COMPLETO

```
def menu():
    print ("\n\tIngresar nuevos datos digite 1")
    print ("\tBuscar datos digite 2")
    print ("\tSalir al menu digite 3")
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/menu.wav")
    pygame.mixer.music.play()
    opc = raw_input("\n\t Ingrese su opcion : ")

    if opc == '1':
        ingresardatos()
    elif opc == '2':
        borrar()
    elif opc == '3':
        menu()
def nombre():
    cursor = db.cursor()
    pygame.mixer.init()
    pygame.mixer.music.load("Audios\nombre.wav")
    pygame.mixer.music.play()
    nombre=raw_input("\n\tIngrese el Nombre: ")
    print ("\nNOMBRE\t\tAPELLIDO\tCEDULA\t\tEDAD\tGENERO\t\tDIRECCION\tTELEFONO\t\tCORREO")
    cursor.execute ("SELECT * FROM Datos WHERE NOMBRE = '" + nombre + "'")
    datos=cursor.fetchall()
    if datos:
        for z in datos:
            print str(z[0]) + "\t\t" + z[1] + "\t\t" + z[2] + "\t" + z[3] + "\t\t" + z[4] + "\t\t" + z[5] + "\t\t" + z[6] + "\t\t" + z[7] + "\t\t" + z[8]
    else:
        pygame.mixer.init()
        pygame.mixer.music.load("Audios\intentar.wav")
        pygame.mixer.music.play()
        print("\nnombre o intentarlo")
        buscar()
def apellido():
    cursor = db.cursor()
    pygame.mixer.init()
    pygame.mixer.music.load("Audios\apellido.wav")
    pygame.mixer.music.play()
    apellido=raw_input("\n\tIngrese el apellido: ")
    print ("\nNOMBRE\t\tAPELLIDO\tCEDULA\t\tEDAD\tGENERO\t\tDIRECCION\tTELEFONO\t\tCORREO")
    cursor.execute ("SELECT * FROM Datos WHERE APELLIDO = '" + apellido + "'")
    datos=cursor.fetchall()
    if datos:
        for z in datos:
            print str(z[0]) + "\t\t" + z[1] + "\t\t" + z[2] + "\t" + z[3] + "\t\t" + z[4] + "\t\t" + z[5] + "\t\t" + z[6] + "\t\t" + z[7] + "\t\t" + z[8]
    else:
        pygame.mixer.init()
        pygame.mixer.music.load("Audios\intentar.wav")
        pygame.mixer.music.play()
        print("\nnombre o intentarlo")
        buscar()
def cedula():
    cursor = db.cursor()
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/cedula.wav")
    pygame.mixer.music.play()
    cedula=raw_input("\n\tIngrese el Numero de cedula: ")
    print ("\nNOMBRE\t\tAPELLIDO\tCEDULA\t\tEDAD\tGENERO\t\tDIRECCION\tTELEFONO\t\tCORREO")
    cursor.execute ("SELECT * FROM Datos WHERE CEDULA = '" + cedula + "'")
    datos=cursor.fetchall()
    if datos:
        for z in datos:
            print str(z[0]) + "\t\t" + z[1] + "\t\t" + z[2] + "\t" + z[3] + "\t\t" + z[4] + "\t\t" + z[5] + "\t\t" + z[6] + "\t\t" + z[7] + "\t\t" + z[8]
    else:
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/intentar.wav")
        pygame.mixer.music.play()
        print("\nnombre o intentarlo")
        buscar()
def buscar():
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/buscar.wav")
    pygame.mixer.music.play()
    print ("\nBuscar por Nombre digite 1\n")
    print ("Buscar por Apellido digite 2\n")
    print ("Buscar por Cedula digite 3\n")
    print ("Salir digite 4")
    buscar = raw_input("\nIngrese su opcion : ")
    if buscar == '1':
        nombre()
        cursor = db.cursor()
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/menubuscar.wav")
        pygame.mixer.music.play()
        print ("\nEliminar registro digite 1\n")
        print ("Volver al menu digite 2")
        buscar = raw_input("\nIngrese su opcion : ")
        if buscar == '1':
            borrar()
        elif buscar == '2':
            menu()
        else:
            print("\nGracias por utilizar nuestra Base de Datos")
            pygame.mixer.init()
            pygame.mixer.music.load("Audios/gracias.wav")
            pygame.mixer.music.play()
            time.sleep(2.5)
    elif buscar == '2':
        apellido()
        cursor = db.cursor()
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/menubuscar.wav")
        pygame.mixer.music.play()
        print ("\nEliminar registro digite 1\n")
        print ("Volver al menu digite 2")
        buscar = raw_input("\nIngrese su opcion : ")
        if buscar == '1':
            borrar()
        elif buscar == '2':
            menu()
        else:
            print("\nGracias por utilizar nuestra Base de Datos")
            pygame.mixer.init()
            pygame.mixer.music.load("Audios/gracias.wav")
            pygame.mixer.music.play()
            time.sleep(2.5)
    elif buscar == '3':
        cedula()
        cursor = db.cursor()
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/buscar.wav")
        pygame.mixer.music.play()
        print ("\nEliminar registro digite 1\n")
        print ("Volver al menu digite 2")
        buscar = raw_input("\nIngrese su opcion : ")
        if buscar == '1':
            borrar()
        elif buscar == '2':
            menu()
        else:
            print("\nGracias por utilizar nuestra Base de Datos")
            pygame.mixer.init()
            pygame.mixer.music.load("Audios/gracias.wav")
```

El código completo que se presenta ayudará de guía para saber a qué altura se encuentran los comandos del programa ya que las tabulaciones son importantes para su correcto funcionamiento.

PREPARACIÓN DEL MÓDULO RASPBERRY

Conectar la fuente asignada al módulo para su energización. La fuente debe cumplir con los parámetros de voltaje y corriente indicados. (5V – 2Amp) para el correcto funcionamiento al conectar varios dispositivos USB. Su entrada de energía se encuentra en el lado izquierdo junto a la salida HDMI (Cable blanco), como se observa en la **Figura C17**.

Figura C17: Energización

Una vez energizado el módulo Raspberry, realizar la conexión del mismo a una de las pantallas del laboratorio usando la salida HDMI del módulo y el puerto VGA disponible en la parte posterior del monitor. Para esta conexión se debe utilizar el convertidor HDMI a VGA en conjunto con el cable VGA que se les proporcionará junto con la fuente de alimentación. Una vez realizada la conexión, cambiar la configuración predeterminada de entrada de video de DVI a VGA, de la siguiente manera:

En el lado derecho del monitor se encuentra el botón de Menú, presionarlo y mostrará las opciones que se observan en la **Figura C18**.

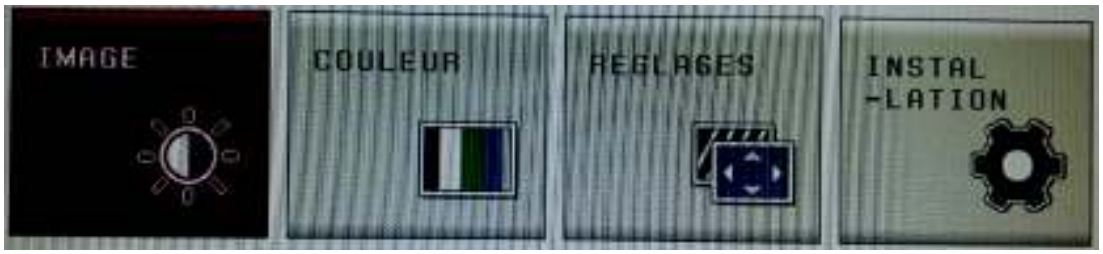


Figura C18: Configuración de Pantalla (Monitor)

Con los botones (flechas) del monitor, dirigirse hasta la opción **INSTALLATION** (**Figura C19**).

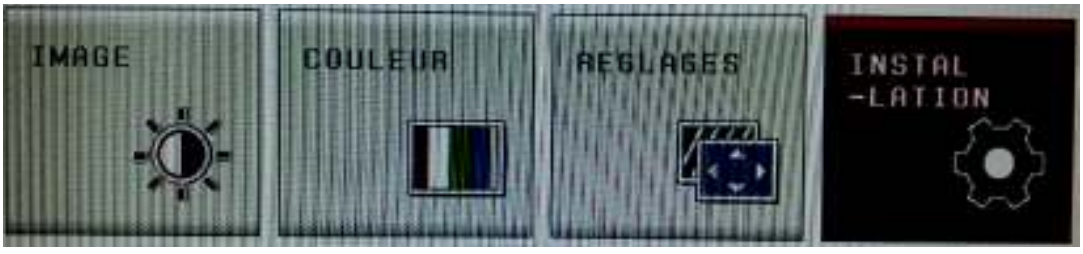


Figura C19: Opción INSTALLATION

Presionando el botón Menú, ingresar a las opciones de INSTALLATION y con las flechas dirigirse hasta INPUT SELECT (**Figura C20**).

Figura C20: Installation

Con el botón Menú, ingresar a las opciones de INPUT SELECT y con las flechas seleccionar VGA. Cuando se selecciona la opción VGA, automáticamente se cambia su señal y se observa en el monitor la pantalla del Raspberry Pi (**Figura C21**).

Figura C21: Interfaz de Raspberry

CREACIÓN DE LA BASE DE DATOS

Ingresar a PhpMyAdmin en el navegador Chromium , de la siguiente manera como se muestra en la **Figura C22**.

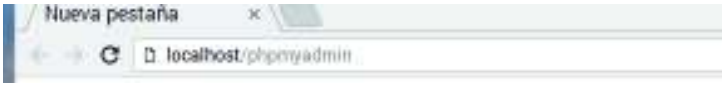


Figura C22: Ingreso a PhpMyAdmin

En las **Figuras C23 y C24** se muestra cómo crear una nueva base de datos, la cual se denominará Registro.

Figura C23: Nueva base de datos

Figura C24: Nombre de la nueva base de datos

Ahora, se creará una tabla y se elegirá el número de columnas dependiendo del número de datos a ingresar, como se observa en la **Figura C25**. Nombre de la tabla **Base1** y número de columnas **9**.

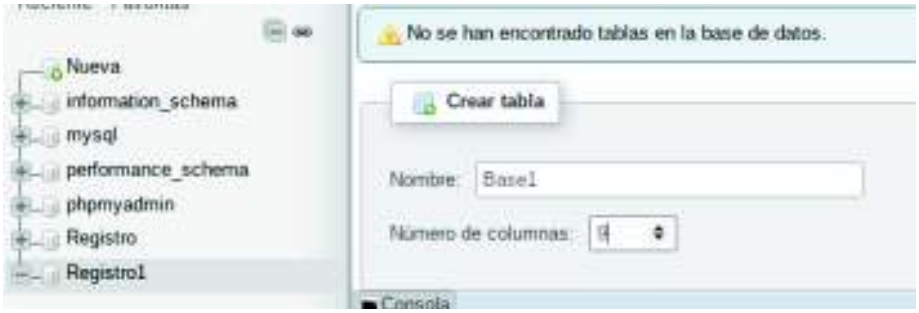


Figura C25: Tabla de la base de datos

En la tabla creada se llenarán los campos y se elegirá el tipo de Dato que se desea ingresar; los nombres de las columnas son:

ORDEN, NOMBRE, APELLIDO, CÉDULA, EDAD, GÉNERO, DIRECCIÓN, TELÉFONO Y CORREO

Solo el primer dato llamado ORDEN será de tipo INT (Dato de tipo Entero), número de índice PRIMARY lo cual habilitará las opciones de edición de las filas ingresadas y auto incrementable. Del dato NOMBRE al dato CORREO serán de tipo VARCHAR (Carácter de Longitud Variable). Todos los datos tienen una longitud de 30, que es la máxima longitud que puede alcanzar el dato.

En las **Figuras C26 y C27** se presenta la tabla creada, en la cual se ingresarán los datos y se debe crear un programa mediante el cual se conectará a la base de datos y se la gestionará.

Figura C26: Datos de la tabla

Figura C27: Tabla phpMyAdmin creada

PROGRAMACIÓN

Abrir el directorio Pi (Gráfico de dos carpetas) y crear una nueva carpeta, la cual se denominará con cualquier nombre, en este caso se la llamará **Programas** (**Figura C28**).

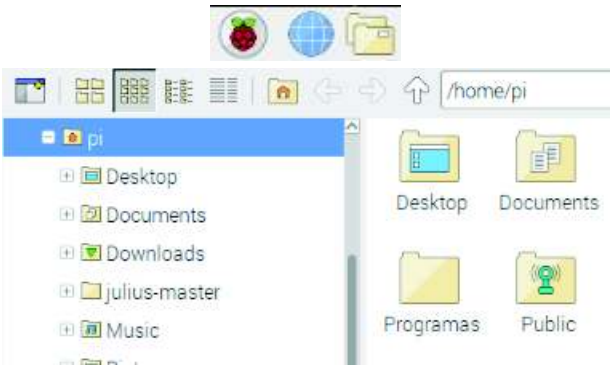


Figura C28: Carpeta Principal (Programas)

Dentro de la carpeta **Programas,** crear una nueva hoja llamada **Base.py** y una carpeta llamada **Audios,** dentro de la cual se colocarán los audios solicitados al estudiante (**Figura C29**).

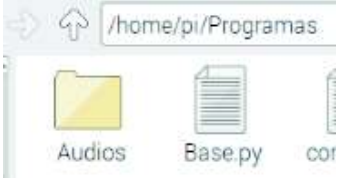


Figura C29: Carpeta de audios y hoja de Python

Abrir la hoja **Base.py** dando clic derecho sobre la hoja y seleccionando Python 2 (IDLE) para iniciar la programación. Véase **Figura C30**.

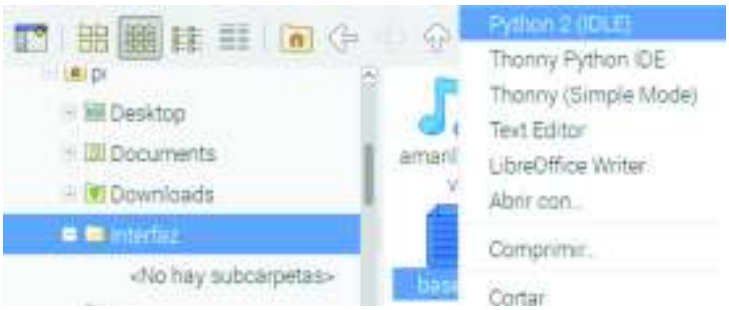


Figura C30: Hoja de Python

En la **Figura C31** se observa la importación de librerías y funcionalidades necesarias para el programa.

```
# -*- coding: utf-8 -*-
import mysql.connector
import sys
import pygame
import time
```

Figura C31: Importación de funciones

1.- El comando de la primera línea sirve para la codificación de caracteres.
2.- Importar la librería de conexión de Python a MySQL-Server.
3.- Importar el módulo que provee las variables y funcionalidades de sistema.
4.- Pygame contiene módulos de Python para la creación de juegos; sin embargo, estos módulos se usaran en este caso solamente para reproducir audios.
5.- Import time provee de funciones relacionadas con el tiempo, es necesaria para poder utilizar **time.sleep(),** más adelante.

La **Figura C32** indica cómo establecer la conexión con la base de datos.

```
try:
    db = mysql.connector.connect(
        host = "localhost",
        user = "root",
        password = "root",
        database = "Registro"
)
except Exception as e:
        sys.exit('No se puede conectar con la Base de Datos')
```

Figura C32: Conexión a base de datos

1.- Se abre la conexión al servidor MySQL con mysql.connector.connect y se guarda en variable db.

2.- Se leen los datos Host, User, Password y Database; si los datos coinciden con los registrados, se establece la conexión, caso contrario no lo hará.

Ahora se creará la función **borrar,** como lo indica la **Figura C33**.

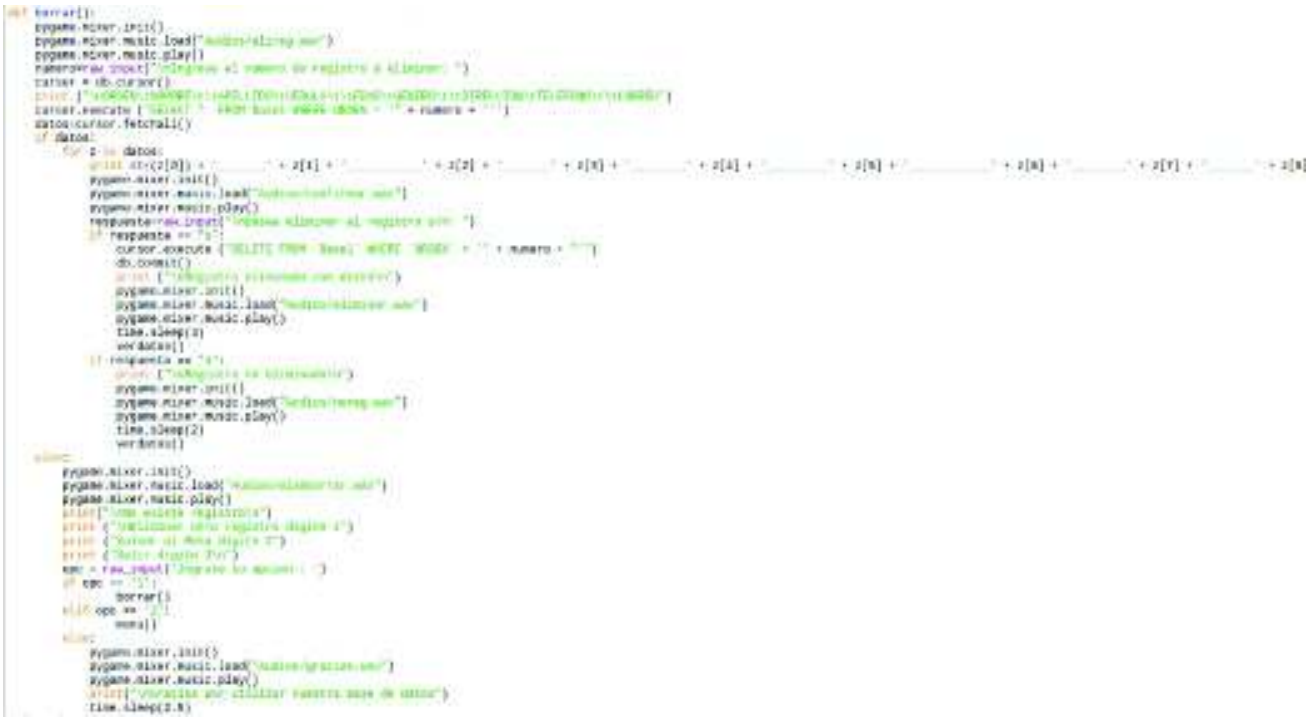


Figura C33: Función Borrar

1.- Se define la función **borrar.**

2.- La línea .init inicializa pygame, Music.load carga el archivo y music.play reproduce el archivo elireg.wav.

3.- Inicia un **raw_input** que se almacena en la variable **número** y se imprime "Ingrese el número de registro a eliminar".

4.- **Cursor = db.cursor()** crea un cursor que es el que permite ejecutar la acción de borrar datos de la tabla Base1 de la base de datos, dependiendo del valor que tenga la columna **ORDEN**.

5.- Define la variable **numero** dentro de la cual se encuentra **raw_input**, el cual toma la entrada del usuario y devuelve una cadena de caracteres y se imprimirá "Ingrese el número de registro a eliminar". El **(\n)** que se encuentra detrás de la palabra **Ingrese,** es un espacio entre filas; es decir, espacios hacia abajo. El dato que ingrese el usuario se almacenará en la variable **numero.**

6.- La línea **print** muestra en pantalla lo que se encuentre en su interior; en este caso, las columnas que se nombraron en la base de datos desde ORDEN hasta CORREO.

7.- Para ejecutar una acción, se ingresa **cursor.execute**; lo que se está haciendo es borrar la tabla Base1 creada en phpMyAdmin, específicamente la columna ORDEN y lo comparará con el dato ingresado por el usuario.

8.- **Datos=cursor.fetchall()** recupera las filas de un conjunto de datos que será la tabla Base1 y si existe devolverá el resultado, de no existir devolverá una lista vacía.

9.- Se reproduce el archivo confirmar.wav.

10.- De coincidir el dato ingresado por el usuario con uno de la tabla, irá a esta línea **if datos;** caso contario, saltará a la línea else que se encuentra al mismo nivel, la cual se explicará posteriormente.

11.- El resultado que se encuentre en **datos** se imprimirá mediante **z.**

12.- Imprimirá un string de datos (**str**) de **z** el cual va incrementando a medida que se lo coloque, aquí el string va desde z[0] hasta z[8]; es decir, nueve columnas que coinciden con el número de columnas de la tabla Base1. El string z[0] se imprime pero no se lo ingresa debido a que en PhpMyAdmin se lo configura como auto incrementable, por lo cual aumentará su valor de forma automática.

13.- Se reproduce el archivo confirmar.wav.

14.- Seguido imprime "Desea eliminar el registro s/n:", el dato ingresado se guarda en la variable **respuesta** y se inicia un nuevo condicional.

15.- Compara y si variable **respuesta** es igual a **s**,

16.- Se ejecutará la eliminación del dato seleccionado tomado de la columna **ORDEN** de la tabla Base1.

17.- **Db.commit()** confirma la ejecución de la línea anterior, la cual es de eliminar dato y se puedan ver los cambios en PhpMyAdmin; sin este comando no se ejecutarán los cambios.

18.- Después de eliminar el dato, se imprimirá "Registro eliminado con éxito". Se reproducirá el archivo eliminar.wav y se ejecutará la función **verdatos** que se explicará más adelante. El **time.sleep** permite una pausa en la función de acuerdo al tiempo en segundos que se coloque en los paréntesis de éste.

19.- Un nuevo condicional **if** indica que si el valor ingresado es **n,** se imprimirá "Registro no eliminado", reproduciendo el archivo noreg.wav y se suspenderá la secuencia por dos segundos.

20.- Se ejecutará la función **verdatos.**

21.- Si el dato que se ingresó en la parte inicial no existe en la tabla Base1, se ejecutará **else.**

22.- Se reproduce el archivo elseborrar.wav.

23.- Se imprimen "No existe registro", "Para eliminar otro registro digite 1", "Para volver al Menú digite 2" y "Para salir digite 3".

24.- Inicia un nuevo **raw_input,** que se almacena en la variable **opc** y se imprime "Ingrese su opción".

25.- Compara y si **opc** es igual a **1,** se ejecuta la función **borrar,** si **opc** es igual a **2 s**e ejecuta la función **menu** y de no cumplirse las condiciones anteriores, con **else** reproducirá el archivo gracias.wav e imprimirá "Gracias por utilizar nuestra Base de Datos". Tendrá una pausa de 2.5 segundos y saldrá del programa.

La **Figura C34** muestra la función **ver datos**.

Figura C34: Función Ver datos

1.- Define la función **verdatos**.

2.- Imprime "DATOS INGRESADOS", los backslash + t que se encuentran en la línea son espacios en esa misma línea.

3.- **Cursor = db.cursor()** crea un cursor que es el que permite ejecutar la acción de seleccionar datos de la tabla Base1 de la base de datos.

4.- Se selecciona todos los datos que contenga la tabla Base1.

5.- **Datos=cursor.fetchall()** recuperará los datos de la tabla y permitirá visualizarlos.

6.- Imprime las columnas desde ORDEN hasta CORREO.

7.- Imprime los datos que se encuentran en la variable **datos** mediante **z** con un string de datos desde z[0] hasta z[8], correspondientes a las nueve columnas de la tabla Base1.

8.- Se ejecuta la función **menu1**.

Para crear la función **ingresar datos,** escribir las líneas que se muestran en la **Figura C35**.

```
def ingresardatos():
    print("\nIngrese sus datos\n")
    cursor = db.cursor()
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/nombre.wav")
    pygame.mixer.music.play()
    nombre = raw_input("Ingrese su Nombre: ")
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/apellido.wav")
    pygame.mixer.music.play()
    apellido = raw_input("Ingrese su Apellido: ")
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/cedula.wav")
    pygame.mixer.music.play()
    cedula = raw_input("Ingrese su Cedula: ")
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/edad.wav")
    pygame.mixer.music.play()
    edad = raw_input("Ingrese su Edad: ")
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/genero.wav")
    pygame.mixer.music.play()
    genero = raw_input("Ingrese su Genero (M/F): ")
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/direccion.wav")
    pygame.mixer.music.play()
    direccion = raw_input("Ingrese su Direccion: ")
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/telefono.wav")
    pygame.mixer.music.play()
    telefono = raw_input("Ingrese su Telefono: ")
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/correo.wav")
    pygame.mixer.music.play()
    correo = raw_input("Ingrese su Correo: ")
    cursor.execute(" INSERT INTO Base1 (NOMBRE,APELLIDO,CEDULA,EDAD,GENERO,DIRECCION,TELEFONO,CORREO) VALUES ('" + nombre + "','" + apellido + "
    db.commit()
    print ("\nDatos ingresados con exito\n")
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/ingresar.wav")
    pygame.mixer.music.play()
    time.sleep(3)
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/menuingresar.wav")
    pygame.mixer.music.play()
    print ("Mostrar datos digite 1")
    print ("Ingresar datos digite 2")
    print ("Volver al menú digite 3")
    print ("Salir digite 4\n")
    opc = raw_input('Ingrese su opcion : ')

    if opc == '1':
            verdatos()
    elif opc == '2':
            ingresardatos()
    elif opc == '3':
        menu()
    elif opc == '4':
        print("\nGracias por utilizar nuestra Base de Datos")
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/gracias.wav")
        pygame.mixer.music.play()
        time.sleep(2.5)
```

Figura C35: Función Ingresar datos

1.- Define la función **ingresardatos**.

2.- Imprime "Ingrese sus datos"

3.- **Cursor = db.cursor()** crea un cursor que es el que permite ejecutar la acción de ingresar datos de la tabla Base1 de la base de datos.

4.- Se ingresa el nombre luego de ejecutarse la función **raw_input,** la cual almacenará el dato en la variable **nombre,** se imprime "Ingrese su Nombre:" y reproduce el archivo nombre.wav

5.- Se ingresa el apellido luego de ejecutarse la función **raw_input,** la cual almacenará el dato en la variable **apellido,** se imprime "Ingrese su Apellido:" y reproduce el archivo apellido.wav.

6.- Se ingresa el número de cédula luego de ejecutarse la función **raw_input,** la cual almacenará el dato en la variable **cedula,** se imprime "Ingrese su Cédula:" y reproduce el archivo cedula.wav

7.- Se ingresa la edad luego de ejecutarse la función **raw_input,** la cual almacenará el dato en la variable **edad,** se imprime "Ingrese su Edad:" y reproduce el archivo edad.wav

8.- Se ingresa el género luego de ejecutarse la función **raw_input,** la cual almacenará el dato en la variable **genero,** se imprime "Ingrese su Género (M/F):" y reproduce el archivo genero.wav

9.- Se ingresa la dirección luego de ejecutarse la función **raw_input,** la cual almacenará el dato en la variable **direccion,** se imprime "Ingrese su Dirección:" y reproduce el archivo direccion.wav

10.- Se ingresa el teléfono luego de ejecutarse la función **raw_input,** la cual almacenará el dato en la variable **telefono,** se imprime "Ingrese su Teléfono:" y reproduce el archivo telefono.wav.

11.- Se ingresa el correo luego de ejecutarse la función **raw_input,** la cual almacenará el dato en la variable **correo,** se imprime "Ingrese su Correo:" y reproduce el archivo correo.wav

12.- Se ejecutará la inserción de los datos en las columnas **NOMBRE**, **APELLIDO**, **CEDULA**, **EDAD**, **GENERO**, **DIRECCION**, **TELEFONO** y **CORREO** de la tabla Base1. Primero, se debe colocar los nombres de las columnas de la misma manera como están nombradas en PhpMyAdmin, seguido los valores que se encuentran en las variables antes declaradas.

13.- **Db.commit()** confirma la ejecución de la línea anterior, la cual es de insertar datos y se puedan ver los cambios en phpMyAdmin.

14.- Imprime "Datos ingresados con éxito".

15.- Reproduce el archivo ingresar.wav; hay una espera de 3 segundos y reproduce el archivo menuingresar.wav.

16.- Imprime "Para mostrar datos digite 1", "Para ingresar nuevos datos digite 2" y "Para salir digite 3".

17.- Inicia un nuevo **raw_input** que se almacena en la variable **opc** y se imprime "Ingrese su opción".

18.- Compara y si **opc** es igual a **1,** se ejecuta la función **verdatos,** si **opc** es igual a **2** se ejecuta la función **ingresardatos**; caso contrario, con **else** imprimirá "Gracias por utilizar nuestra Base de Datos", reproduce el archivo gracias.wav con una pausa de 2.5 segundos y saldrá del programa.

La **Figura C36** muestra la creación de la función menú 1.

```
def menu1():
    print ('\n'"Ingresar nuevos datos digite 1")
    print ("Eliminar datos digite 2")
    print ("Volver al menú digite 3")
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/menu1.wav")
    pygame.mixer.music.play()
    opc = raw_input('\n''Ingrese su opcion : ')

    if opc == '1':
        ingresardatos()
    elif opc == '2':
        borrar()
    elif opc == '3':
        menu()
```

Figura C36: Función Menú 1

1.- Define la función **menu1**.

2.- Imprime "Para ingresar nuevo registro digite 1", "Para eliminar registro digite 2" y "Para volver al menú digite 3" y reproduce el archivo menu1.wav.

3.- Inicia un nuevo **raw_input** que se almacena en la variable **opc** y se imprime "Ingrese su opción".

4.- Compara y si **opc** es igual a **1** se ejecuta la función **ingresardatos,** si **opc** es igual a **2** se ejecuta la función **borrar** y si **opc** es igual a **3** se ejecuta la función **menu()**.

Crear la función **nombre** al igual que la **Figura C37**.

```
def nombre():
    cursor = db.cursor()
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/inombre.wav")
    pygame.mixer.music.play()
    nombre=raw_input("\nIngrese el Nombre: ")
    print ("\nORDEN\tNOMBRE\t\tAPELLIDO\tCEDULA\t\tEDAD\tGENERO\t\tDIRECCION\tTELEFONO\t\tCORREO")
    cursor.execute ("SELECT *  FROM Base1 WHERE NOMBRE = '" + nombre + "'")
    datos=cursor.fetchall()
    if datos:
        for z in datos:
            print str(z[0]) + '_______' + z[1] + '___________' + z[2] + '_______' + z[3] + '________' + z[4] +
    else:
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/intentar.wav")
        pygame.mixer.music.play()
        print("\nVuelva a Intentarlo")
        buscar()
```

Figura C37: Función Nombre

1.- Define la función **nombre**.

2.- **Cursor = db.cursor()** crea un cursor que es el que permite ejecutar la acción de seleccionar datos de la columna **NOMBRE** de la tabla Base1 y se reproduce el archivo inombre.wav.

3.- Define la variable **nombre,** dentro de la cual se encuentra **raw_input**, el cual toma la entrada del usuario y devuelve una cadena de caracteres y se imprimirá "Ingrese el Nombre:".

4.- **Print** imprime lo que se encuentre en su interior; en este caso, las columnas que se nombraron en la base de datos desde ORDEN hasta CORREO.

5.- Para ejecutar una acción se ingresa **cursor.execute**, lo que se está haciendo es leer la tabla Base1 creada en phpMyAdmin, específicamente la columna **NOMBRE** y lo comparará con el dato ingresado por el usuario.

6.- **Datos=cursor.fetchall()** recupera las filas de un conjunto de datos que será la tabla Base1; devolverá el resultado que puede ser una sola fila o un conjunto de filas, dependiendo de las coincidencias encontradas, de no existir devolverá una lista vacía.

7.- De coincidir el dato ingresado por el usuario con uno de la tabla, irá a esta línea **if datos**; caso contario, saltará a la línea **else** para ejecutar las instrucciones de su interior.

8.- El resultado que se encuentre en **datos** se imprimirá mediante **z.**

9.- Imprimirá un string de datos (**str**) de **z**, el string va desde z[0] hasta z[8] que son las columnas de la tabla Base1.

10.- Si no se cumplen las condiciones anteriores, con **else** reproducirá el archivo intentar,wav. Imprimirá "Vuelva a Intentarlo" y se ejecutará la función **buscar**.

Crear la función **apellido,** como se muestra en la **Figura C38**.

```
def apellido():
    cursor = db.cursor()
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/iapellido.wav")
    pygame.mixer.music.play()
    apellido=raw_input("\nIngrese el Apellido: ")
    print ("\nORDEN\tNOMBRE\t\tAPELLIDO\tEDAD\t\tGENERO\t\tDIRECCION\tTELEFONO\t\tCORREO")
    cursor.execute ("SELECT * FROM Base1 WHERE APELLIDO = '" + apellido + "'")
    datos=cursor.fetchall()
    if datos:
        for z in datos:
            print str(z[0]) + "\t\t" + z[1] + "\t\t" + z[2] + "\t\t" + z[3] + "\t\t" + z[4]
    else:
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/intentar.wav")
        pygame.mixer.music.play()
        print("\nVuelva a Intentarlo")
        buscar()
```

Figura C38: Función Apellido

1.- Define la función **apellido**.

2.- **Cursor = db.cursor() c**rea un cursor que es el que permite ejecutar la acción de seleccionar datos de la columna **APELLIDO** de la tabla Base1 y se reproduce el archivo iapellido.wav.

3.- Define la variable **apellido** dentro de la cual se encuentra **raw_input**, el cual toma la entrada del usuario y devuelve una cadena de caracteres y se imprimirá "Ingrese el Apellido:".

4.- **Print** imprime lo que se encuentre en su interior; en este caso, las columnas que se nombraron en la base de datos desde ORDEN hasta CORREO.

5.- Para ejecutar una acción, se ingresa **cursor.execute**, lo que se está haciendo es leer la tabla Base1 creada en phpMyAdmin, específicamente la columna **APELLIDO** y lo comparará con el dato ingresado por el usuario.

6.- **Datos=cursor.fetchall()** recupera las filas de un conjunto de datos que será la tabla Base1; devolverá el resultado que puede ser una sola fila o un conjunto de filas, dependiendo de las coincidencias encontradas. De no existir coincidencias, devolverá una lista vacía.

7.- De coincidir el dato ingresado por el usuario con uno de la tabla, irá a esta línea **if datos**; caso contario, saltará a la línea **else** para ejecutar las instrucciones de su interior.

8.- El resultado que se encuentre en **datos,** se imprimirá mediante **z.**

9.- Imprimirá un string de datos (**str**) de **z**; el string va desde z[0] hasta z[8] que son las columnas de la tabla Base1.

10.- Si no se cumplen las condiciones anteriores con **else,** reproducirá el archivo intentar,wav; imprimirá "Vuelva a Intentarlo" y se ejecutará la función **buscar**.

Crear la función cédula, como se muestra en la **Figura C39**.

```
def cedula():
    cursor = db.cursor()
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/cedula.wav")
    pygame.mixer.music.play()
    cedula=raw_input("\nIngrese el Número de cédula: ")
    print ("\nORDEN\tNOMBRE\t\tAPELLIDO\tCEDULA\t\tEDAD\tGENERO\t\tDIRECCION\tTELEFONO\t\tCORREO")
    cursor.execute ("SELECT *  FROM Base1 WHERE CEDULA = '" + cedula + "'")
    datos=cursor.fetchall()
    if datos:
        for z in datos:
            print str(z[0]) + '_______' + z[1] + '_____________' + z[2] + '_______' + z[3] + '_________' + z[4] +
    else:
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/intentar.wav")
        pygame.mixer.music.play()
        print("\nVuelva a Intentarlo")
        buscar()
```

Figura C39: Función Cédula

1.- Define la función **cedula**.

2.- **Cursor = db.cursor() c**rea un cursor que es el que permite ejecutar la acción de seleccionar datos de la columna **CEDULA** de la tabla Base1 y se reproduce el archivo iacedula.wav.

3.- Define la variable **cedula,** dentro de la cual se encuentra **raw_input**, el cual toma la entrada del usuario y devuelve una cadena de caracteres y se imprimirá "Ingrese el Número de cédula:".

4.- **Print** imprime lo que se encuentre en su interior; en este caso, las columnas que se nombraron en la base de datos desde ORDEN hasta CORREO.

5.- Para ejecutar una acción, se ingresa **cursor.execute**, lo que se está haciendo es leer la tabla Base1 creada en phpMyAdmin, específicamente la columna **CEDULA** y la comparará con el dato ingresado por el usuario.

6.- **Datos=cursor.fetchall()** recupera las filas de un conjunto de datos que será la tabla Base1; devolverá el resultado que puede ser una sola fila o un conjunto de filas, dependiendo de las coincidencias encontradas. De no existir coincidencias, devolverá una lista vacía.

7.- Si el dato ingresado por el usuario coincide con uno de la tabla, dicho dato se registrará dentro de la función **if datos**; caso contario, saltará a la línea **else** para ejecutar las instrucciones de su interior.

8.- El resultado que se encuentre en **datos** se imprimirá mediante **z.**

9.- Imprimirá un string de datos (**str**) de **z**, el string va desde z[0] hasta z[8], que son las columnas de la tabla Base1.

10.- Si no se cumplen las condiciones anteriores con **else,** reproducirá el archivo intentar,wav. Imprimirá "Vuelva a Intentarlo" y se ejecutará la función **buscar**.

Con ayuda de la **Figura C40,** se creará la función buscar.

```
def buscar():
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/buscar.wav")
    pygame.mixer.music.play()
    print ("\nBuscar por Nombre digite 1\n")
    print ("Buscar por Apellido digite 2\n")
    print ("Buscar por Cédula digite 3\n")
    print ("Salir digite 4")
    buscar = raw_input('\nIngrese su opcion : ')
    if buscar =='1':
        nombre()
        cursor = db.cursor()
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/menubuscar.wav")
        pygame.mixer.music.play()
        print ("\nEliminar registro digite 1\n")
        print ("Volver al menu digite 2")
        buscar = raw_input('\nIngrese su opcion : ')
        if buscar == '1':
            borrar()
        elif buscar == '2':
            menu()
        else:
            print("\nGracias por utilizar nuestra Base de Datos")
            pygame.mixer.init()
            pygame.mixer.music.load("Audios/gracias.wav")
            pygame.mixer.music.play()
            time.sleep(2.5)
    elif buscar =='2':
        apellido()
        cursor = db.cursor()
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/menubuscar.wav")
        pygame.mixer.music.play()
        print ("\nEliminar registro digite 1\n")
        print ("Volver al menu digite 2")
        buscar = raw_input('\nIngrese su opcion : ')
        if buscar == '1':
            borrar()
        elif buscar == '2':
            menu()
        else:
            print("\nGracias por utilizar nuestra Base de Datos")
            pygame.mixer.init()
            pygame.mixer.music.load("Audios/gracias.wav")
            pygame.mixer.music.play()
            time.sleep(2.5)
    elif buscar =='3':
        cedula()
        cursor = db.cursor()
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/buscar.wav")
        pygame.mixer.music.play()
        print ("\nEliminar registro digite 1\n")
        print ("Volver al menu digite 2")
        buscar = raw_input('\nIngrese su opcion : ')
        if buscar == '1':
            borrar()
        elif buscar == '2':
            menu()
        else:
            print("\nGracias por utilizar nuestra Base de Datos")
            pygame.mixer.init()
            pygame.mixer.music.load("Audios/gracias.wav")
            pygame.mixer.music.play()
            time.sleep(2.5)
    elif buscar =='4':
        print("\nGracias por utilizar nuestra Base de Datos")
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/gracias.wav")
        pygame.mixer.music.play()
        time.sleep(2.5)
    else:
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/inicie.wav")
        pygame.mixer.music.play()
        print("\nInicie nuevamente\n")
        menu()
```

Figura C40: Función Buscar

1.- Define la función **buscar.**

2.- Se reproduce el archivo buscar.wav e imprime "Buscar por Nombre digite 1", "Buscar por Apellido digite 2", "Buscar por Cédula digite 3" y "Salir digite 4".

3.- Se ingresa la opción luego de ejecutarse la función **raw_input,** que se almacena en la variable **buscar** y se imprime "Ingrese su opción".

4.- Ejecuta el **if**, compara. Si **buscar** es igual a **1,** se ejecuta la función **nombre.**

5.- Dentro de la función, se crea uncursor para que se puedan realizar instrucciones y reproduce el archivo menubuscar.wav.

6.- Imprime "Eliminar registro digite 1" y "Volver al menú digite 2".

7.- Se ingresa la opción luego de ejecutarse la función **raw_input,** que se almacena en la variable **buscar** y se imprime "Ingrese su opción".

8.- Compara y si **buscar** es igual a **1,** se ejecuta la función **borrar.**

9.- Compara y si **buscar** es igual a **2,** se ejecuta la función **menú.**

10.- Si no se cumplen las condiciones anteriores, se ejecuta el **else** de las condiciones internas; reproduce el archivo gracias.wav e imprime "Gracias por usar nuestra Base de Datos".

11.- Ejecuta el primer **elif**, compara y si **buscar** es igual a **2** se ejecuta la función **apellido.**

12.- Dentro de la función se crea un cursor para que se puedan realizar instrucciones y reproduce el archivo menubuscar.wav.

13.- Imprime "Eliminar registro digite 1" y "Volver al menú digite 2".

14.- Inicia **raw_input** que se almacena en la variable **buscar** y se imprime "Ingrese su opción".

15.- Compara y si **buscar** es igual a **1** se ejecuta la función **borrar.**

16.- Compara y si **buscar** es igual a **2** se ejecuta la función **menu.**

17.- Si no se cumplen las condiciones anteriores, se ejecuta el **else** de las condiciones internas. Reproduce el archivo gracias.wav e imprime "Gracias por usar nuestra Base de Datos".

18.- Ejecuta el segundo **elif**, compara y si **buscar** es igual a **3,** se ejecuta la función **cedula.**

19.- Dentro de la función se crea un cursor para que se puedan realizar instrucciones y reproduce el archivo menubuscar.wav.

20.- Imprime "Eliminar registro digite 1" y "Volver al menú digite 2".

21.- Se ingresa la opción luego de ejecutarse la función **raw_input,** que se almacena en la variable **buscar** y se imprime "Ingrese su opción".

22.- Compara y si **buscar** es igual a **1,** se ejecuta la función **borrar.**

23.- Compara y si **buscar** es igual a **2,** se ejecuta la función **menu.**

24.- Si no se cumplen las condiciones anteriores, se ejecuta el **else** de las condiciones internas. Reproduce el archivo gracias.wav e imprime "Gracias por usar nuestra Base de Datos".

25.- Ejecuta el tercer **elif**, compara y si **buscar** es igual a **4,** reproduce el archivo gracias.wav e imprime "Gracias por usar nuestra Base de Datos".

26.- Si no se cumplen las condiciones anteriores en su **else,** se reproduce el archivo inicie,wav. Se imprime "Inicie nuevamente" y se ejecuta la función **menu**.

La última función de la **Figura C41** que se definirá es **menú** y esta es la función principal, la primera que se ejecutará al iniciar el programa y dentro de la cual se ejecutarán las demás funciones, conforme se las vaya llamando.

```
def menu():
    print ("\nBienvenido a la Base de Datos de Registro para continuar seleccione un item del siguiente menu\n")
    print ("\t\t\t\t****** MENU ******\t\t\t\n")
    print ("1.- Mostrar Datos\t"),
    print ("2.- Ingresar Datos\t"),
    print ("3.- Eliminar Datos\t"),
    print ("4.- Buscar Datos\t\n")
    print ("5.- Salir\n")
    pygame.mixer.init()
    pygame.mixer.music.load("Audios/menup.wav")
    pygame.mixer.music.play()
    opc = raw_input('Ingrese su opcion : ')

    if opc == '1':
        verdatos()
    elif opc =='2':
        ingresardatos()
    elif opc == '3':
        borrar()
    elif opc == '4':
        buscar()
    else:
        pygame.mixer.init()
        pygame.mixer.music.load("Audios/gracias.wav")
        pygame.mixer.music.play()
        print("\nGracias por utilizar nuestra Base de Datos")
        time.sleep(2.5)
```

Figura C41: Función Menú

1.- Define la función **menu.**

2.- Imprime "Bienvenido a la Base de Datos de Registro para poder continuar seleccione un ítem del siguiente menú".

3.- Imprime "1.- Mostrar Datos", "2- Ingresar Datos", "3.- Eliminar Datos", "4.- Buscar Datos", "5.- Salir" y reproduce el archivo menup.wav.

4.- Se ingresa la opción luego de ejecutarse la función **raw_input,** que se almacena en la variable **buscar** y se imprime "Ingrese su opción".

5.- Ejecuta el **if**, compara y si **opc** es igual a **1,** se ejecuta la función **verdatos**; si es igual a **2,** se ejecuta la función **ingresardatos**, si **opc** es igual a **3** se ejecuta la función **borrar** y si **opc** es igual a **4,** se ejecuta la función **buscar**.

6.- De no cumplirse las condiciones anteriores, se ejecuta **else.** Reproduce el archivo gracias.wav e imprime "Gracias por usar nuestra Base de Datos" y sale del programa.

```
menu()
exit
```

Figura C42: Inicio de Programa

En esta última línea de la **Figura C42,** se ejecuta el programa llamando a la función principal **menu(),** mostrando el mensaje y las opciones principales y **exit** nos indica la finalización del programa.

Para ejecutar el programa, se lo puede hacer de dos maneras: abriéndolo directamente con ejecutor de Python o desde LXTerminal.

EJECUCIÓN DEL PROGRAMA

Desde Python, dirigirse a la pestaña Run (**Figura C43**) y seleccionar Run Module, o presionando F5 el programa se iniciará. Se observa que el programa funciona correctamente (**Figura C44**).

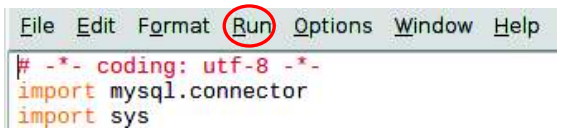


Figura C43: Ejecutar Programa

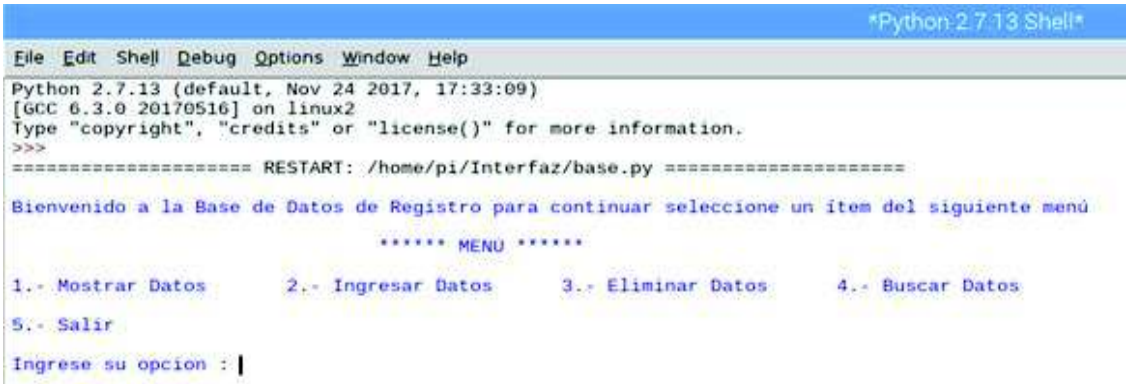


Figura C44: Ejecución del Programa

Desde LXTerminal, ingresar la ruta en la cual se encuentra el archivo.

El comando **cd** permite ingresar en carpetas; en este caso, ingresar a la carpeta Programas.

La palabra **Programas** en letras azules indica que se ha colocado bien la ruta y se ha entrado en la carpeta para ejecutar el programa. Se coloca Python con la versión en la que se hizo el programa; en este caso, Python 2; seguido del nombre del archivo y abrir (**Figura C45**).

Figura C45: Programa en LXTerminal

Para comprobar que el programa funciona, se gestiona datos en la base de datos. Primero, mostrar datos, para lo cual presionar 1 y observar que la tabla está vacía. Ingresar nuevamente a PhpMyAdmin, mediante el navegador Chromium y verificar que la tabla en la base de datos esté igualmente vacía. (**Figura C46**).

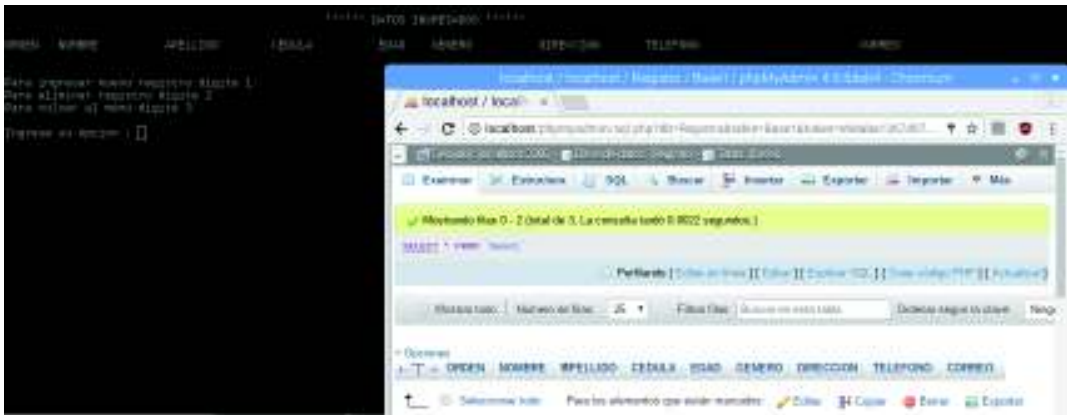


Figura C46: Comparación del programa y PhpMyAdmin

Al ingresar nuevos registros, se observa que los datos se ingresaron con éxito, entonces se debe continuar ingresando más registros (**Figura C47**).

Figura C47: Prueba Ingresar datos

Los registros se visualizan tanto en la pantalla de comandos, como en la base de datos de PhpMyAdmin (**Figura C48**).

Figura C48: Prueba Ingresar datos

Eliminar el registro número 8 y observar que se realizó con éxito la eliminación (**Figura C49**).

Figura C49: Prueba Eliminar Datos

El registro se eliminó de la base de datos como se esperaba (**Figuras C50 y C51**).

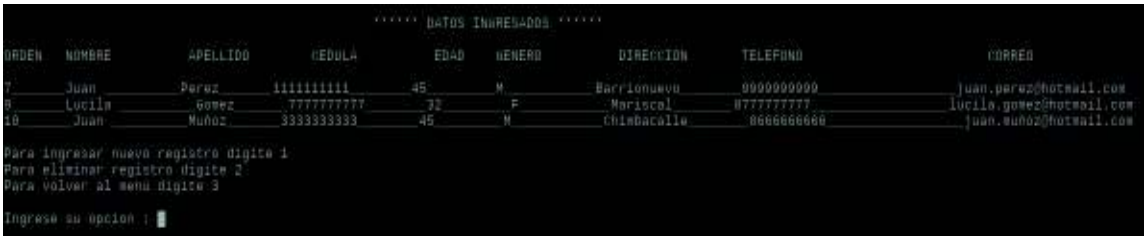


Figura C50: Prueba Eliminar datos

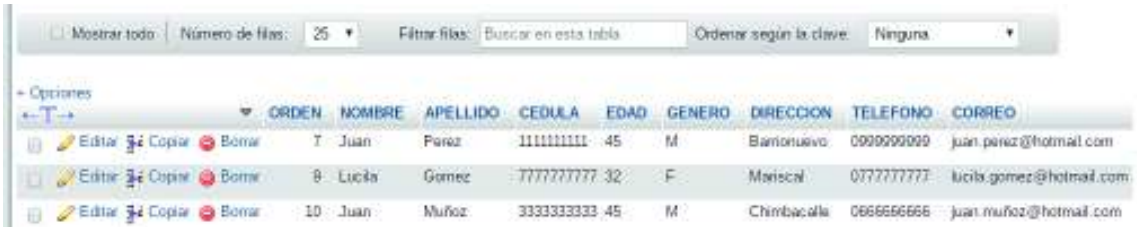


Figura C51: Comparación con PhpMyAdmin

Para finalizar, buscar datos por nombre. Ingresar Juan, notar que devuelve dos coincidencias con todos sus datos (**Figura C52**).

Figura C52: Prueba Buscar

Con esto, se comprueba que el programa está funcionando correctamente.

3. RECOMENDACIONES

- Se recomienda investigar la sintaxis de comandos que usa Python 2 ya que ésta varía en comparación a Python 3. Si se usan comandos en una versión diferente, presentarán errores al momento de compilar.
- Realizar en Python solo el código que permita la conexión a la base de datos de phpMyAdmin y verificar su funcionamiento correcto; de no hacerlo, identificar el error y solucionarlo antes de continuar.
- Prestar mucha atención al momento de realizar la programación, ya que el mínimo error de escritura evita su compilación y funcionamiento.

ANEXO C3

HOJA GUÍA PARA EL INSTRUCTOR PRÁCTICA N°3

"Procesamiento de voz y control de puertos GPIO utilizando módulos Raspberry Pi 3"

1. **TEMA:** Procesamiento de Voz y control de puertos GPIO utilizando módulos Raspberry Pi 3.

2. DESARROLLO DE LA PRÁCTICA

CÓDIGO COMPLETO

```
#! /usr/bin/python -u

import sys
import os
import RPi.GPIO as GPIO

class Rhythmbox:
        name = "Rhythmbox"

        commands = {
                        'play': 'play',
                        'pause': 'pause',
                        'next': 'next',
                        'prev': 'previous',
                        'show': 'notify',
                        'pause': 'pause',
                        'silence': 'pausa',
        }
        def parse(self, word):
                if word in self.commands:
                        return 'rhythmbox-client --%s' % self.commands[word]

class CommandAndControl:
        def __init__(self, file_object):

                if os.system('ps xa | grep -v grep | grep rhythmbox >/dev/null') == 0:
                        self.mediaplayer = Rhythmbox()
                elif os.system('which rhythmbox >/dev/null') == 0:
                        self.mediaplayer = Rhythmbox()
                else:
                        print 'Comando no reconocido. ' \
                                'Favor intente de nuevo'
                        sys.exit(1)

                print '\r\nBienvenido al Sistema de Control por Voz\n'
                print 'Computer Blue enciende led Azul'
                print 'Computer Green enciende led Verde'
                print 'Computer Pause apaga led Azul'
                print 'Computer Next enciende led Verde\n'
                print 'Tomando el control de salidas GPIO'

                startstring = 'sentence1: <s> '
                endstring = ' </s>'

                while 1:
                        line = file_object.readline()
                        if not line:
                                break
                        if 'missing phones' in line.lower():
                                print 'Error: Fonemas no encontrados en la base de datos.'
                                sys.exit(1)
                        if line.startswith(startstring) and line.strip().endswith(endstring):
                                self.parse(line.strip('\n')[len(startstring):-len(endstring)])

        def parse(self, line):
                # Parse the input
                params = [param.lower() for param in line.split() if param]
                if not '-q' in sys.argv and not '--quiet' in sys.argv:
                        print 'Reconociendo entradas:', ' '.join(params).capitalize()

                command = self.mediaplayer.parse(params[1])

                if params[1] == 'blue':

                        GPIO.setmode(GPIO.BCM)

                        GPIO.setup(26, GPIO.OUT)

                        GPIO.output(26, True)

                        print 'Ha pronunciado', ' '.join(params).capitalize()
                        print ' ha reconocido '.join(params)
                        print 'Led Azul Encendido\n'
                if params[1] == 'pause':

                        GPIO.setmode(GPIO.BCM)

                        GPIO.setup(26, GPIO.OUT)

                        GPIO.output(26, False)

                        print 'Ha pronunciado', ' '.join(params).capitalize()
                        print ' ha reconocido '.join(params)
                        print 'Led Azul Apagado\n'
                if params[1] == 'green':

                        GPIO.setmode(GPIO.BCM)

                        GPIO.setup(19, GPIO.OUT)

                        GPIO.output(19, True)

                        print 'Ha pronunciado', ' '.join(params).capitalize()
                        print ' ha reconocido '.join(params)
                        print 'Led Verde Encendido\n'
                if params[1] == 'next':

                        GPIO.setmode(GPIO.BCM)

                        GPIO.setup(19, GPIO.OUT)

                        GPIO.output(19, False)

                        print 'Ha pronunciado', ' '.join(params).capitalize()
                        print ' ha reconocido '.join(params)
                        print 'Led Verde Apagado\n'

if __name__ == '__main__':
        try:
                CommandAndControl(sys.stdin)
        except KeyboardInterrupt:
                sys.exit(1)
```

El código completo que se presenta, ayudará de guía para saber a qué altura se encuentran los comandos del programa ya que las tabulaciones son importantes para su correcto funcionamiento.

PREPARACIÓN DEL MÓDULO RASPBERRY

Conectar la fuente asignada al módulo para su energización. La fuente debe cumplir con los parámetros de voltaje y corriente indicados, (5V – 2Amp) para el correcto funcionamiento al conectar varios dispositivos USB. Su entrada de energía se encuentra en el lado izquierdo junto a la salida HDMI (Cable blanco), como se observa en la **Figura C53**.

Figura C53: Energización

Una vez energizado el módulo Raspberry, realizar la conexión del mismo a una de las pantallas del laboratorio usando la salida HDMI del módulo y el puerto VGA disponible en la parte posterior del monitor. Para esta conexión se debe utilizar el convertidor HDMI a VGA en conjunto con el cable VGA que se les proporcionará junto con la fuente de alimentación. Una vez realizada la conexión, cambiar la configuración predeterminada de entrada de video de DVI a VGA, de la siguiente manera:

En el lado derecho del monitor se encuentra el botón de Menú, presionarlo y mostrará las opciones que se observan en la **Figura C54**.

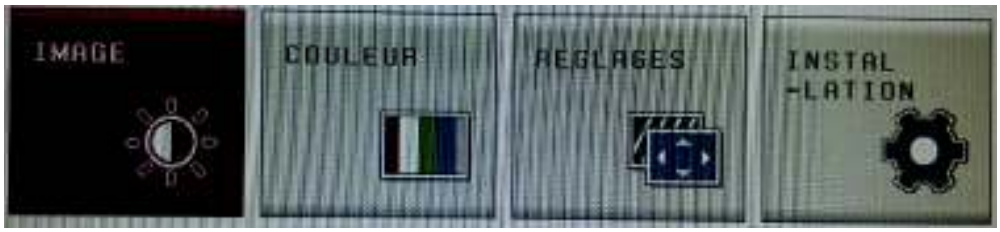


Figura C54: Configuración de Pantalla (Monitor)

Con los botones (flechas) del monitor, dirigirse hasta la opción **INSTALLATION** (**Figura C55**).

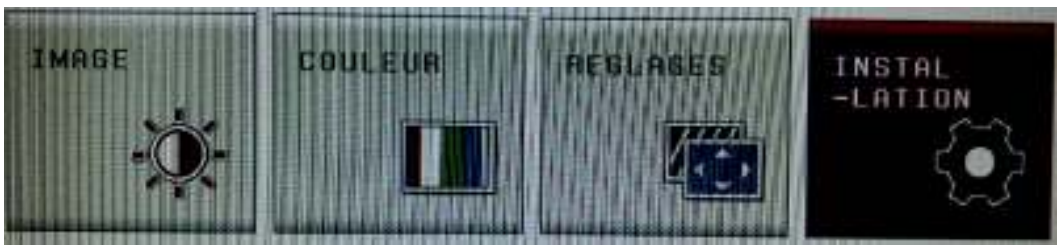


Figura C55: Opción INSTALLATION

Presionando el botón Menú, ingresar a las opciones de INSTALLATION y con las flechas dirigirse hasta INPUT SELECT (**Figura C56**).

Figura C56: Installation

Con el botón Menú, ingresar a las opciones de INPUT SELECT y con las flechas seleccionar VGA. Cuando se selecciona la opción VGA, automáticamente se cambia su señal y se observa en el monitor la pantalla del módulo Raspberry Pi (**Figura C57**).

Figura C57: Interfaz de Raspberry

PROGRAMACIÓN

Abrir el directorio Pi (Gráfico de dos carpetas) y crear una nueva carpeta, la cual se denominará con cualquier nombre; en este caso, **Programas** (**Figura C58**).

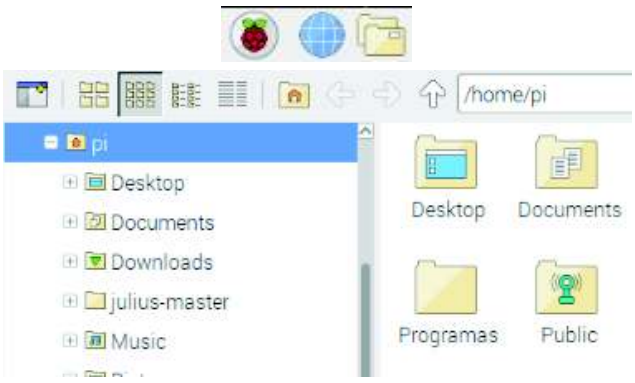


Figura C58: Carpeta Principal (Programas)

Copiar los archivos de la carpeta Reproductor, que se encuentra en el mismo directorio **Pi,** dentro de la nueva carpeta (**Figura C59**).

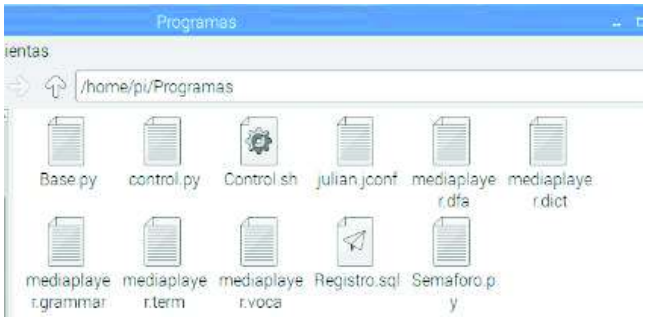


Figura C59: Copia de Archivos

Modificar los archivos mediaplayer.dict y mediaplayer.voca de la carpeta **Programas,** añadiendo los nuevos comandos con su pronunciación la cual se puede consultar en Google como Fonetica de palabras; en este caso, **Blue y Green,** como se presenta en las **Figuras C60 y C61**.

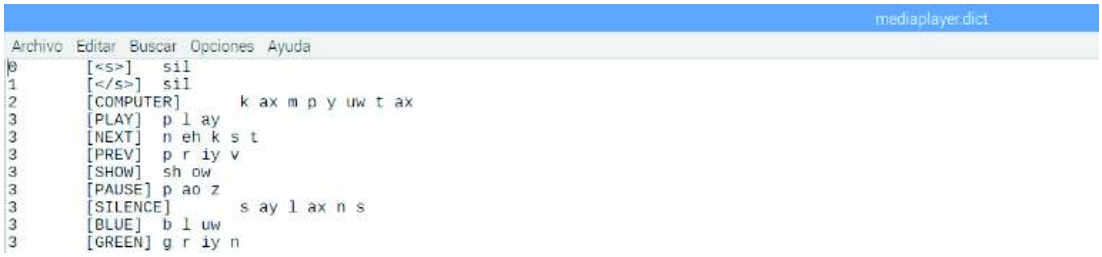


Figura C60: Edición archivo Mediaplayer.dict

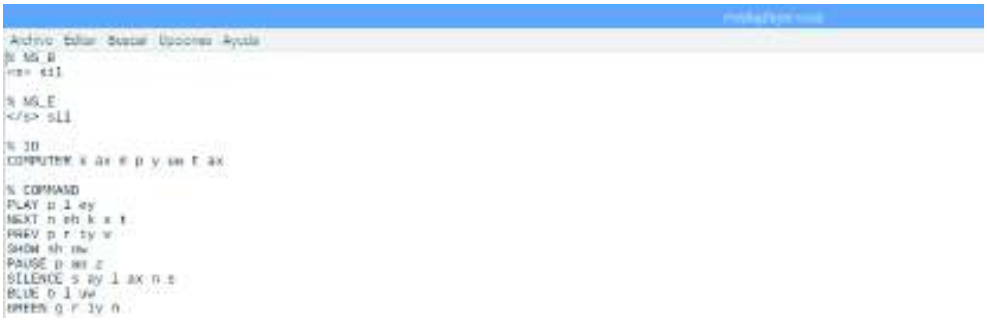


Figura C61: Edición archivo Mediaplayer.voca

Dentro de la nueva carpeta, crear dos nuevas hojas, una de programación de Python llamada **control.py** y otra un script llamado **Control.sh** (**Figura C62**).

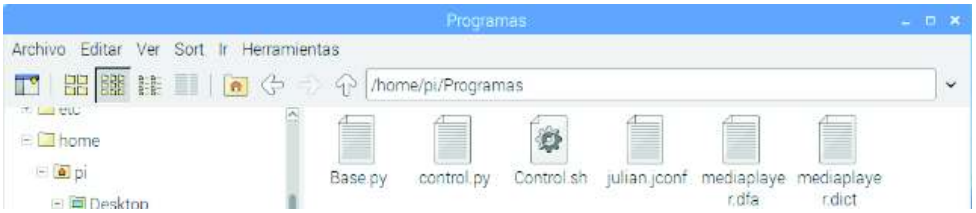


Figura C62: Hoja de programación y Script

Abrir la hoja **control.py,** dando clic derecho sobre la hoja y seleccionando Python 2 (IDLE), como en la **Figura C63** y empezar la programación.

Figura C63: Hoja de Python

Esta línea, conocida como proceso, indica la ruta al intérprete para que pueda reconocer los comandos que vienen a continuación del programa (#! /usr/bin/python -u).

```
#! /usr/bin/python -u
```

La **Figura C64** muestra la importación de funciones necesarias para ejecutar el programa.

```
import sys
import os
import RPi.GPIO as GPIO
```

Figura C64: Importación de Funciones

1.- **Import sys** importa el módulo que provee las variables y funcionalidades de sistema.

2.- **Import os** habilita todas las funciones de sistema para interactuar con el sistema operativo.

3.- **Import RPi.GPIO as GPIO** contiene el módulo RPI.GPIO y lo nombra como GPIO.

Class crea la plantilla llamada Rhythmbox, dentro de la cual se encuentran los comandos soportados por defecto en el programa Rhythmbox (**Figura C65**).

```
class Rhythmbox:

        name = "Rhythmbox"

        commands = {
                        'play': 'play',
                        'pause': 'pause',
                        'next': 'next',
                        'prev': 'previous',
                        'show': 'notify',
                        'pause': 'pause',
                        'silence': 'pause',
        }
```

Figura C65: Comandos Predeterminados

La **Figura C66** muestra las líneas de comando para el análisis de Rhythmbox

```
def parse(self, word):
        if word in self.commands:
                return 'rhythmbox-client --%s' % self.commands[word]
```

Figura C66: Análisis de Rhythmbox

Def parse define un análisis gramatical de una palabra (word), donde self define que el análisis es a nivel de clase.

Si la variable "word" está en comandos a nivel de clase; es decir que se encuentra en la plantilla de Rhythmbox, retorna la respuesta del programa con la palabra de comandos.

La **Figura C67** muestra las funciones para determinar la clase CommandAndControl

```
class CommandAndControl:

        def __init__(self, file_object):


                if os.system('ps xa | grep -v grep | grep rhythmbox >/dev/null') == 0:
                        self.mediaplayer = Rhythmbox()
                elif os.system('which rhythmbox >/dev/null') == 0:
                        self.mediaplayer = Rhythmbox()
                else:
                        print 'Comando no reconocido. ' \
                              'Favor intente de nuevo'
                        sys.exit(1)
```

Figura C67: Clase CommandAndControl

1.- Crea la clase CommandAndControl, dentro del cual se encuentran las órdenes de ejecución.

2.- Define INIT, el cual es el primer comando a ejecutarse al llamarse automáticamente e inicializando sus atributos internos.

3.- En las dos condiciones IF y ELIF, se analiza que no existan errores internos del sistema. Si el valor es igual a cero, quiere decir que todo está correcto y se ejecuta Rhythmbox.

4.- Caso contrario, se imprimirá "Comando no reconocido", y el sistema no se ejecutará. (**Ver Figura C67**)

La **Figura C68** muestra los mensajes iniciales y qué función realizará cada comando

```
print '\t\nBienvenido al Sistema de Control por Voz\n'
print 'Computer Blue enciende led Azul'
print 'Computer Green enciende led Verde'
print 'Computer Pause apaga led Azul'
print 'Computer Next apaga led Verde\n'
print 'Tomando el control de salidas GPIO'
```

Figura C68: Mensajes de Inicio

1.- Imprime “Bienvenido al Sistema de Control por Voz”. El backslash más t (\t) significa un espacio en la misma línea y el back slash más n (\n) significa espacios entre filas.

2.- Imprime “Computer Blue enciende LED Azul”

3.- Imprime” Computer Green enciende LED Verde”

4.- Imprime “Computer Pause apaga LED Azul”

5.- Imprime “Computer Next apaga LED Verde”

Starstring permite iniciar la secuencia de datos str y endstring la finaliza.

```
startstring = 'sentence1: <s> '
endstring = ' </s>'
```

La **Figura C69** muestra el bucle que determinará la comparación y ejecución del programa.

```
while 1:
        line = file_object.readline()
        if not line:
                break
        if 'missing phones' in line.lower():
                print 'Error: Fonemas no encontrados en la base de datos.'
                sys.exit(1)
        if line.startswith(startstring) and line.strip().endswith(endstring):
                self.parse(line.strip('\n')[len(startstring):-len(endstring)])
```

Figura C69: Comparación y ejecución del programa

1.- Entrará en un bucle infinito con While 1

2.- **Line = file_object.readline(),** define una variable llamada **line,** en la que se leerá una línea completa con readline del archivo file_object interno de Julius.

3.- IF NOT LINE indica que si el valor de la variable definida no existe, saldrá de la ejecución.

4.- Si existen fonemas desconocidos en **line_lower** que convierte cualquier letra en minúscula, imprimirá “Error: Fonemas no encontrados en la base de datos” y terminará el programa.

5.- Startswith analiza si la línea comienza con string (str) de starstring y endstring; mientras que line.strip evita espacios en blanco al principio y final de la línea, mostrando en pantalla únicamente el texto.

6.- Se realiza un análisis gramatical dentro del cual se devuelven los valores en blanco de la línea y devuelve las filas existentes en startstring y endstring (**Figura C69**).

Def parse define un análisis gramatical de **line**, donde self define que el análisis es a nivel de clase.

Define la variable params en un nivel bajo y devolviendo los parámetros del string, separados usando al str como el separador con **line.split()**.

Sys.argv es un agumento de líneas de comandos de Python. Para que pueda funcionar, primero se debe importar sys (import sys), se forma como una matriz de todos los comandos del programa y los lee de manera ordenada. Si todo está en orden, imprimirá “Reconociendo entradas” (**Figura C70**).

```
def parse(self, line):
        # Parse the input
        params = [param.lower() for param in line.split() if param]
        if not '-q' in sys.argv and not '--quiet' in sys.argv:
                print 'Reconociendo entradas:', ' '.join(params).capitalize()

        command = self.mediaplayer.parse(params[1])
```

Figura C70: Reconocimiento de entradas

En la **Figura C71** se muestran los pasos para encender el LED Azul.

```
if params[1] == 'blue':

        GPIO.setmode(GPIO.BCM)

        GPIO.setup(26, GPIO.OUT)

        GPIO.output(26, True)

        print 'Ha pronunciado', ' '.join(params).capitalize()
        print ' ha reconocido '.join(params)
        print 'Led Azul Encendido\n'
```

Figura C71: Encendido LED azul

1.- Si la función parámetros (params) es igual a BLUE, la función entrará en un estado activo, la cual habilitará físicamente el pin del puerto GPIO con su respectiva configuración.

2.- Los pines GPIO a usarse en la función (params) se identificarán por su número "Broadcom SOC Channel o BCM" (BCM 26) y no por su número físico de PIN (pin 37), como se observa en la **Figura C72**. Para utilizar el número físico de los pines en la función, se escribirá GPIO.BOARD en lugar de GPIO.BCM como se mostró en la sentencia dentro de la función parámetros (params).

Figura C72: Nomenclatura de pines [31]

3.- El BCM 26 (pin 37) se configura como salida (GPIO.OUT).

4.- El BCM 26 (pin 37) en modo de salida se enciende (True).

5.- Se muestra el mensaje “Ha pronunciado” junto con la función .**join(params)** correspondiente a lo que sería la pronunciación de “computer blue” en la pantalla. Además, con capitalize, el valor se devuelve con la primera letra en mayúscula (Computer blue).

6.- Se muestra el mensaje “Ha reconocido” junto con la función .**join(params),** correspondiente a lo que sería la pronunciación de “computer blue” en la pantalla.

7.- Se muestra el mensaje “LED Azul Encendido” en la pantalla.

En la **Figura C73** se muestran los pasos para apagar el LED Azul.

```
if params[1] == 'pause':
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(26, GPIO.OUT)
        GPIO.output(26, False)
        print 'Ha pronunciado', ' '.join(params).capitalize()
        print ' ha reconocido '.join(params)
        print 'Led Azul Apagado\n'
```

Figura C73: Apagado LED azul

1.- Si la función parámetros (params) es igual a PAUSE, la función entrará en un estado activo, la cual habilitará físicamente el pin del puerto GPIO con su respectiva configuración.

2.- Lo pines GPIO están seteados para ser reconocidos por su número "Broadcom SOC Channel o BCM" (BCM 26) y no por su número físico de PIN (pin 37).

3.- El BCM 26 (pin 37) se configura como salida (GPIO.OUT).

4.- El BCM 26 (pin 37) en modo de salida, apaga (False).

5.- Se muestra el mensaje “Ha pronunciado” junto con la función .**join(params)** correspondiente a lo que sería la pronunciación de “computer pause” en la pantalla. Además, con capitalize, el valor se devuelve con la primera letra en mayúscula (Computer pause).

6.- Se muestra el mensaje “Ha reconocido” junto con la función .**join(params),** correspondiente a lo que sería la pronunciación de “computer pause” en la pantalla.

7.- Se muestra el mensaje “LED Azul Apagado” en la pantalla.

En la **Figura C74** se muestran los pasos para encender el LED Verde.

```
if params[1] == 'green':
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(19, GPIO.OUT)
        GPIO.output(19, True)
        print 'Ha pronunciado', ' '.join(params).capitalize()
        print ' ha reconocido '.join(params)
        print 'Led Verde Encendido\n'
```

Figura C74: Encendido LED verde

1.- Si la función parámetros (params) es igual a GREEN, la función entrará en un estado activo, la cual habilitará físicamente el pin del puerto GPIO con su respectiva configuración.

2.- Lo pines GPIO están seteados para ser reconocidos por su número "Broadcom SOC Channel o BCM" (BCM 19) y no por su número físico de PIN (pin 35).

3.- El BCM 19 (pin 35) se configura como salida (GPIO.OUT).

4.- El BCM 19 (pin 35) en modo de salida, enciende (True).

5.- Se muestra el mensaje “Ha pronunciado” junto con la función .**join(params)** correspondiente a lo que sería la pronunciación de “computer green” en la pantalla. Además, con capitalize, el valor se devuelve con la primera letra en mayúscula (Computer green).

6.- Se muestra el mensaje “Ha reconocido” junto con la función .**join(params),** correspondiente a lo que sería la pronunciación de “computer green” en la pantalla.

7.- Se muestra el mensaje “LED Verde Encendido” en la pantalla.

En la **Figura C75** se muestran los pasos para apagar el LED Verde.

```
if params[1] == 'next':
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(19, GPIO.OUT)
        GPIO.output(19, False)
        print 'Ha pronunciado', ' '.join(params).capitalize()
        print ' ha reconocido '.join(params)
        print 'Led Verde Apagado\n'
```

Figura C75: Apagado LED verde

1.- Si la función parámetros (params) es igual a NEXT, la función entrará en un estado activo, la cual habilitará físicamente el pin del puerto GPIO con su respectiva configuración.

2.- Lo pines GPIO están seteados para ser reconocidos por su número "Broadcom SOC Channel o BCM" (BCM 19) y no por su número físico de PIN (pin 35).

3.- El BCM 19 (pin 35) se configura como salida (GPIO.OUT).

4.- El BCM 19 (pin 35) en modo de salida, apaga (False).

5.- Se muestra el mensaje “Ha pronunciado” junto con la función .**join(params)** correspondiente a lo que sería la pronunciación de “computer next” en la pantalla. Además, con capitalize, el valor se devuelve con la primera letra en mayúscula (Computer next).

6.- Se muestra el mensaje “Ha reconocido” junto con la función .**join(params),** correspondiente a lo que sería la pronunciación de “computer next” en la pantalla.

7.- Se muestra el mensaje “LED Verde Apagado” en la pantalla.

La línea if _name_ == '_main_' analiza si el módulo del programa está siendo ejecutado como programa principal o si está siendo importado desde otro módulo. Es decir, si se lo ejecuta como programa principal con Pyhton o en la terminal de comandos el atributo name pasará a main; caso contrario, si se lo importa como se lo hace mediante el script main, toma en sí el nombre del archivo control.py para ejecutarlo.

Si cumple con la condición, intenta y ejecuta lo que está en su interior, la clase CommandAndControl definida anteriormente; caso contrario, (except) termina el proceso si existe algo mal o el usuario termina el programa abruptamente presionando Control + C (**Figura C76**).

```
if __name__ == '__main__':
        try:
                CommandAndControl(sys.stdin)
        except KeyboardInterrupt:
                sys.exit(1)
```

Figura C76: Programa

Una vez finalizado el programa dentro del Script **Control.sh** (**Figura C77**) creado al inicio, escribir las líneas de la imagen a continuación.

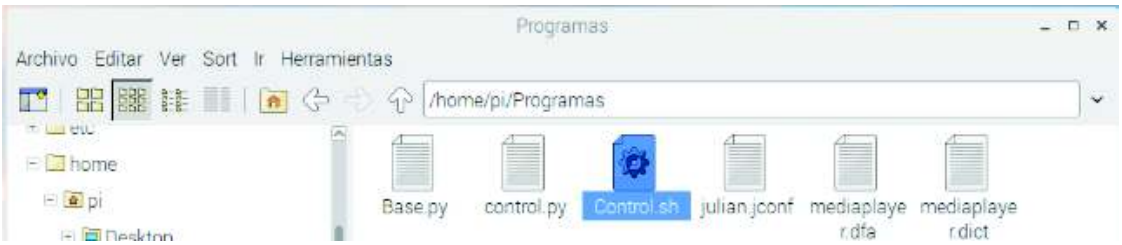


Figura C77: Archivo Control.sh

NOTA: Verificar que los cambios realizados en las bases de datos Mediaplayer.dict y Mediaplayer.voca se hayan guardado para que el programa pueda reconocerlos.

ALSADEV es una variable de entorno que usa Julius para su funcionamiento, la cual hay que exportarla cada vez que el módulo Raspberry Pi 3 se reinicie. Para evitar su exportación manual, se crea el script en el cual se hace la exportación de la variable y la ejecución de programa directamente **(Figura C78)**.

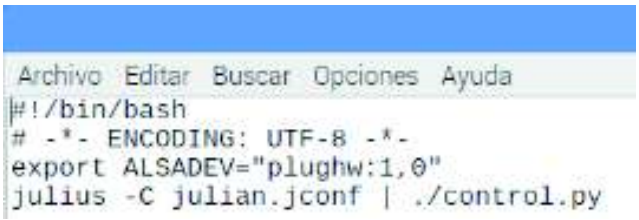


Figura C78: Exportación ALSADEV y Programa

EJECUCIÓN DEL PROGRAMA

Para ejecutar el programa, únicamente dar doble clic sobre el archivo **Control.sh**. Aparece la ventana que se observa en la **Figura C79**, seleccionar **Ejecutar en el Terminal** y el programa se abrirá automáticamente.

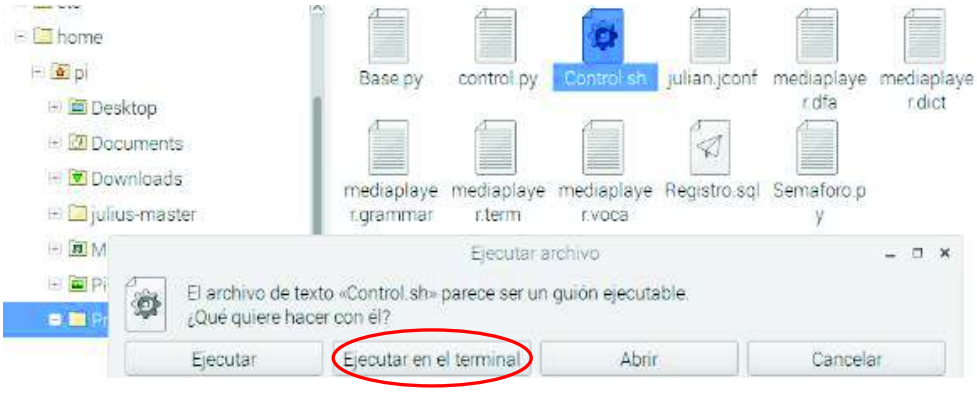


Figura C79: Ejecutar Programa

La línea "<<< please speak >>>" es propia de Julius y aparecerá automáticamente para realizar la captación y el reconocimiento de la voz. La estructura de pronunciación de Julius es computer más el comando con el que se ejecutará la acción (computer + comando). Ver **Figura C80**.

Figura C80: Ejecución del Programa

Se observa en las **Figuras C81 – C84** la comprobación del funcionamiento de los módulos, el cual es correcto.

Figura C81: Encender LEDs azul y verde

Figura C82: LEDs azul y verde encendidos

```
Reconociendo entradas: Computer next
Ha pronunciado Computer next
computer ha reconocido next
Led Verde Apagado

Reconociendo entradas: Computer pause
Ha pronunciado Computer pause
computer ha reconocido pause
Led Azul Apagado
```

Figura C83: Apagar LEDs azul y verde

Figura C84: LEDs azul y verde apagados

3. RECOMENDACIONES

- Se recomienda prestar mucha atención en los espacios o tabulaciones que se deben ingresar en el programa para que éste las ejecute en el orden correcto y el programa funcione.
- Pronunciar adecuadamente los comandos tomando en cuenta que requieren de dos partes en su estructura.

ANEXO D
INSTALACIÓN DE SOFTWARE Y LIBRERÍAS

ANEXO D1

INSTALACIÓN DEL SISTEMA OPERATIVO RASPBIAN

ANEXO D2

INSTALACIÓN DE OPENCV Y NUMPY

ANEXO D3

INSTALACIÓN DE APACHE, MYSQL Y PHPMYADMIN

ANEXO D4

INSTALACIÓN DE JULIUS Y RYTHMBOX

ANEXO D1

INSTALACIÓN DEL SISTEMA OPERATIVO RASPBIAN

SISTEMA OPERATIVO RASPBIAN

Para instalar el sistema operativo Raspbian, se puede seguir cualquiera de estos dos métodos.

Instalación de Raspbian por medio de imagen

El sistema operativo Raspbian puede ser instalado en la microSD por medio de un archivo de imagen que lo contiene. Para esto se necesitará un computador con lector de SD que permita instalar la imagen [32].

Descarga de la imagen

Las imágenes oficiales de los sistemas operativos recomendados están disponibles para su descarga desde el sitio web de Raspberry. https://www.raspberrypi.org/

Una vez en la página principal de Raspberry, dirigirse a la pestaña de Descargas (Downloads).

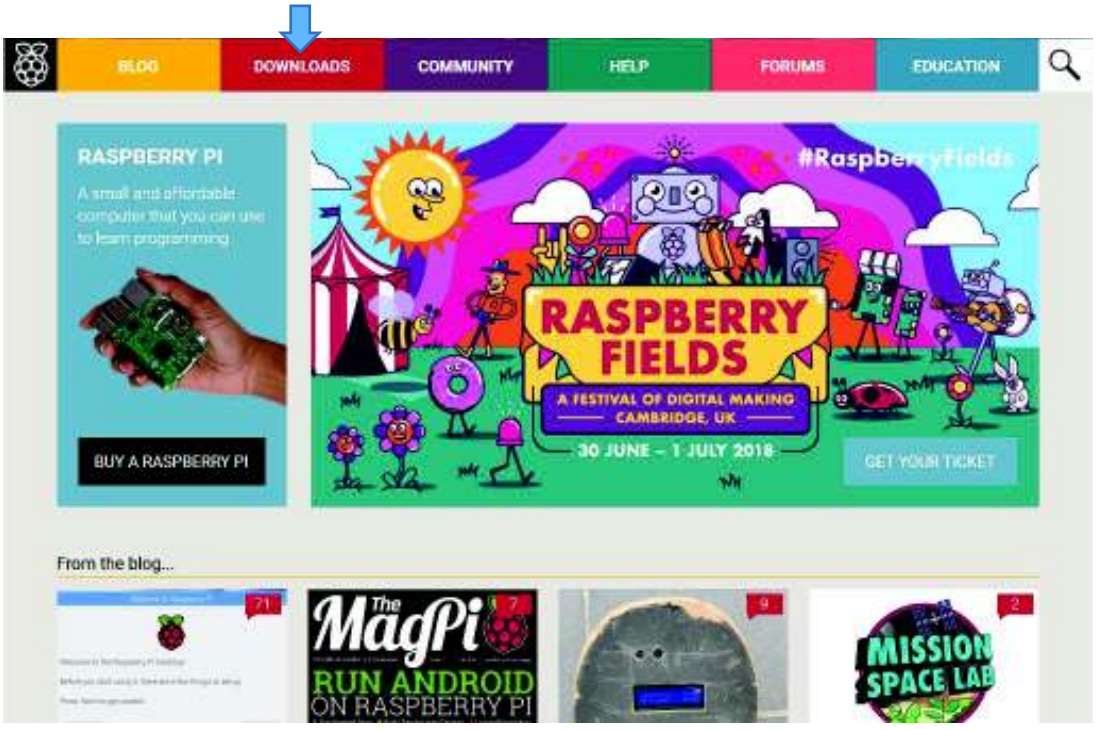


Figura D1: Página web de Raspberry Pi [32]

Al hacer clic en la pestaña, mostrará dos opciones. Se debe elegir la opción RASPBIAN.

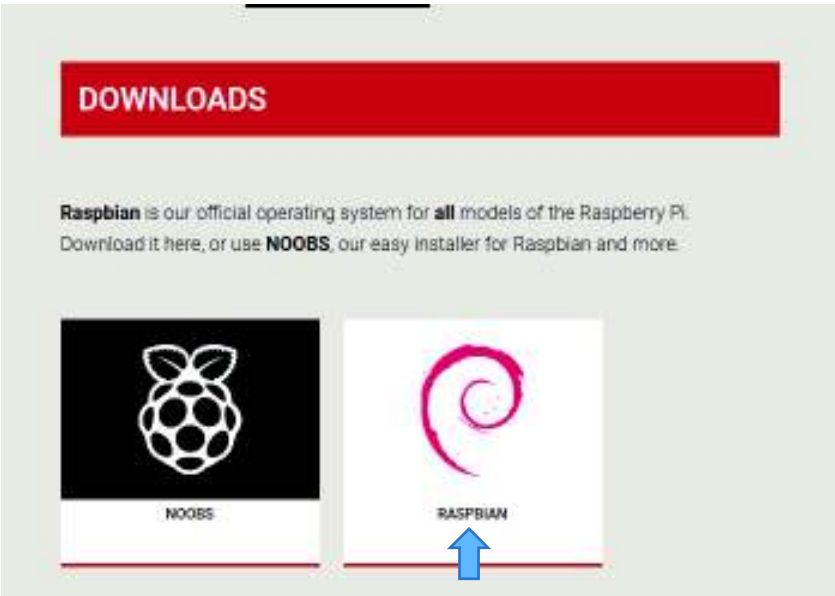


Figura D2: Tipos de instalación del SO [32]

Luego de hacer clic en la opción RASPBIAN, se verá dos opciones que contienen el sistema operativo. Elegir la opción de la izquierda con el tipo de descarga preferido.

Figura D3: Opciones de descarga del SO [32]

Nota: La descarga que contiene la imagen de Raspbian en un archivo ZIP debe ser descomprimido por una herramienta de descompresión de archivos que soporte el formato ZIP64. El software de descompresión recomendado es:

- Para Windows: 7-Zip
- Para Mac: The Unarchiver
- Para Linux: Unzip

Grabar la imagen en la microSD

Se necesitará utilizar una herramienta de escritura de archivos de imagen para instalar el sistema operativo descargado en la microSD.

Etcher es una herramienta de escritura para grabar en microSD, la cual es compatible para Windows, Linux y Mac, además de ser fácil de utilizar. Incluso permite la grabación de imágenes directamente del archivo ZIP sin necesidad de realizar la descompresión del mismo. Para escribir la imagen con Etcher se procede de la siguiente manera:

- Descargar Etcher y luego instalarlo. https://etcher.io/
- Ingresar la microSD en el lector de tarjetas del equipo.
- Abrir Etcher y seleccionar desde la carpeta contenedora el archivo de imagen (.img o .zip) que se desee escribir en la microSD.

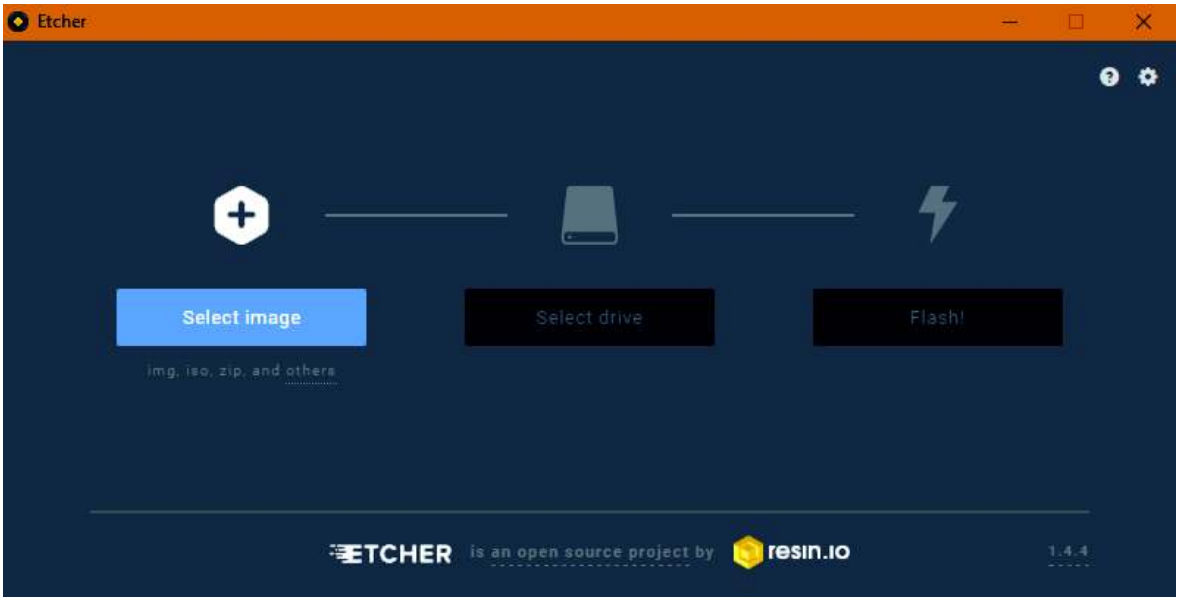


Figura D4: Ventana de Etcher

- Seleccionar la tarjeta SD en la que se desee escribir la imagen.
- Revisar nuevamente las opciones seleccionadas y dar clic en Flash! para que la escritura en la tarjeta SD comience.

Una vez realizada la escritura de la imagen en la tarjeta SD, se procede con su extracción del equipo. Después se introduce la tarjeta SD en la Raspberry Pi para

luego energizarla. Luego de haber realizado esto, la Raspberry Pi inmediatamente empezará a cargar el sistema operativo Raspbian. Tomará un poco de tiempo mientras inicia el sistema.

Instalación de Raspbian por medio de NOOBS

NOOBS es una forma más sencilla e interactiva de instalar el sistema operativo Raspbian en la microSD [33]. Para ello dirigirse a la pestaña Descargas de la página principal de Raspberry. Una vez abierto se visualizará las dos opciones de descargas anteriormente observadas. Seleccionar la opción NOOBS.

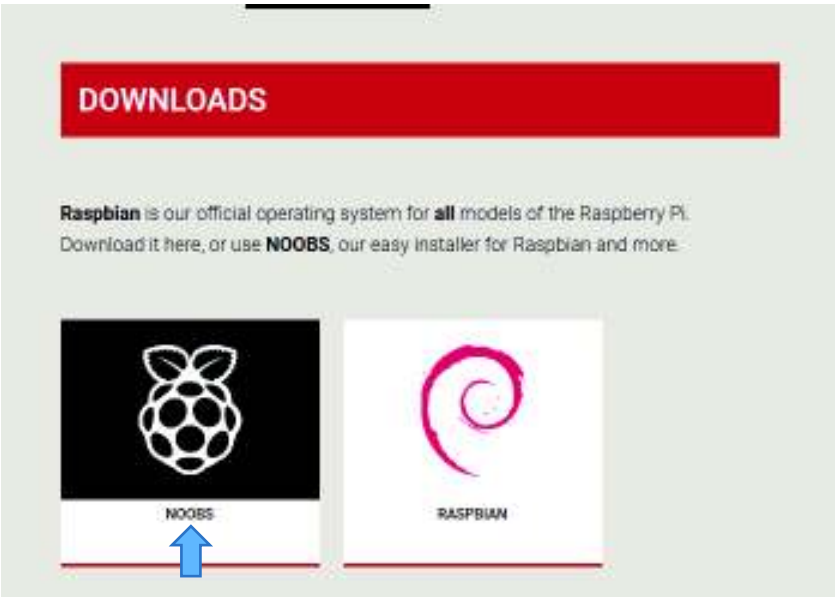


Figura D5: Tipos de instalación del SO [33]

Al hacer clic en la opción NOOBS se desplegaran dos opciones. Elegir la opción de la izquierda con el tipo de descarga de preferencia en formato ZIP.

Figura D6: Opciones de descarga del SO [33]

Formateo de tarjeta SD

Antes de instalar NOOBS en la microSD es recomendable formatear la tarjeta en la que se realizará el proceso. Se recomienda utilizar la herramienta de formateo *SD Card Formatter*. https://www.sdcard.org/downloads/formatter_4/index.html

Extracción de NOOBS del archivo ZIP

- Luego de haber realizado el formateo de la tarjeta SD, ir a la carpeta donde se descargó el archivo.
- Al encontrarlo, descomprimir dicho archivo en una nueva carpeta.
- Después de descomprimir el archivo ZIP, copiar los archivos resultantes de la descompresión directamente en la tarjeta SD.

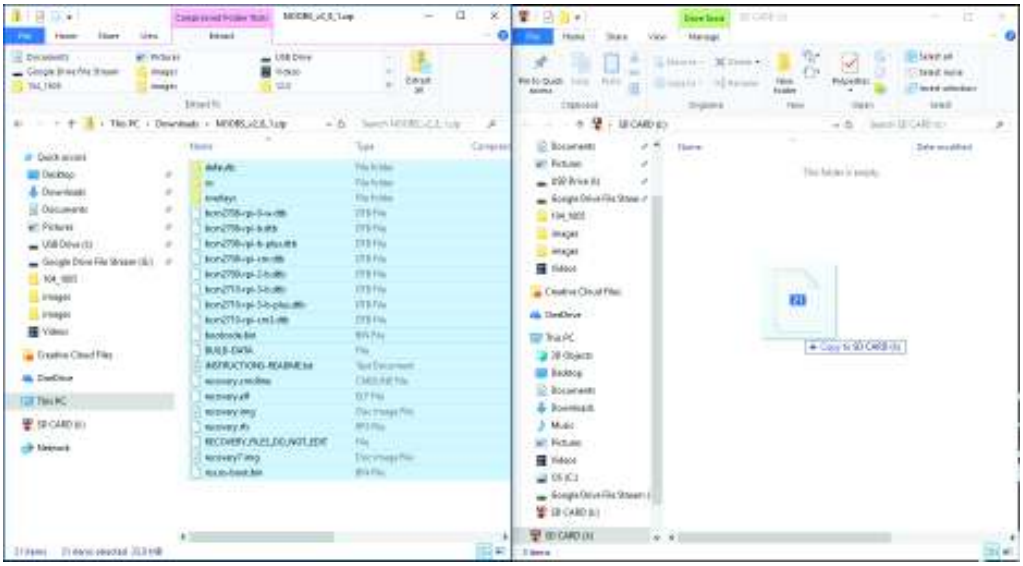


Figura D7: Copia de archivos a la SD [33]

- Una vez copiados los archivos, expulsar la tarjeta SD del equipo.

Inicio desde NOOBS

- Luego de haber extraído la tarjeta SD del equipo se debe insertar ésta en la Raspberry Pi.
- Después de haber insertado la tarjeta SD en la Raspberry Pi se procede a energizarla y a conectar los periféricos necesarios para trabajar en ésta.

- Al iniciarse la Raspberry Pi, mostrará una ventana con opciones de instalación. Seleccionar la opción *Raspbian* y luego clic en la opción *Install.*

Figura D8: Ventana de instalación de NOOBS [33]

- En la ventana emergente dar clic en Si para que el proceso de instalación continúe. Tomará un tiempo en realizar la instalación.

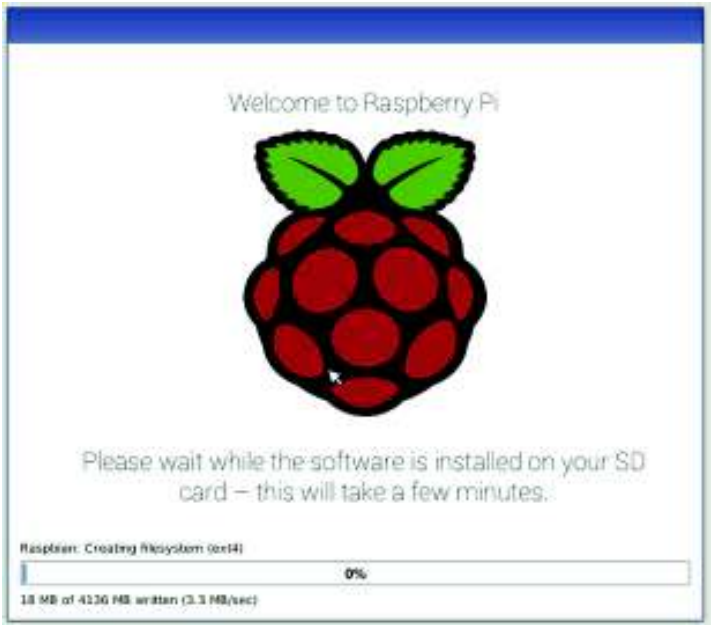


Figura D9: Ventana de progreso de instalación de Raspbian [33]

- Cuando Raspbian haya terminado de instalarse, dar clic en Ok en la ventana emergente y la Raspberry Pi se reiniciará para iniciar el sistema operativo.

ANEXO D2

INSTALACIÓN DE OPENCV Y NUMPY

OPENCV Y NUMPY

El proceso de instalación de OpenCV es quizá el más extenso de todos los programas incluidos en la Raspberry Pi. Por tal motivo se recomienda seguir cuidadosamente cada paso para evitar errores en el proceso de instalación [34].

Instalación

La instalación de OpenCV se realizará a partir de la plataforma de LXTerminal incluida en el sistema operativo de Raspbian. Se deberá ingresar los respectivos comandos explicados a continuación.

Paso 1: Expansión del sistema de archivos

Si recientemente se instaló el sistema operativo Raspbian en la Raspberry Pi, es necesario expandir el sistema de archivos para incluir todo el espacio disponible de la microSD en la instalación de OpenCV. Para ello se escribe el siguiente comando:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi
1 $ sudo raspi-config
```

Luego de escribir el comando, dar enter y aparecerá una ventana. Seleccionar de la lista el parámetro de opciones avanzadas (*Advanced Options*).

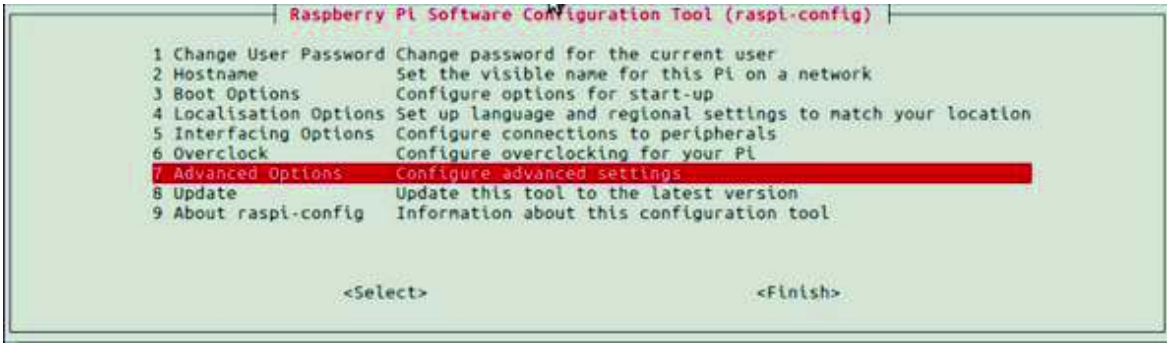


Figura D10: Ventana de configuración de Raspberry Pi [34]

Después, seleccionar el parámetro de expansión del sistema de archivos (*Expand Filesystem*)

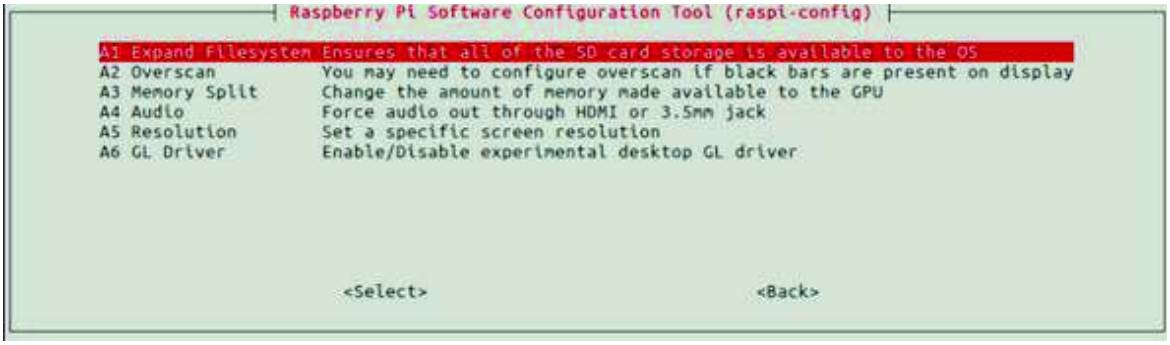


Figura D11: Ventana de opciones avanzadas de Raspberry Pi [34]

Una vez que se da enter en la opción de expansión del sistema de archivos, seleccionar más abajo la opción finalizar (Finish) para que luego la Raspberry Pi ejecute un reinicio. Si no se da el reinicio, hacerlo mediante el siguiente código:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi  Shell
1 $ sudo reboot
```

Después de realizarse el reinicio, el sistema de archivos debió haberse expandido para poder incluir todo el espacio disponible de la microSD.

Paso 2: Instalación de dependencias

El primer paso consiste en actualizar todos los paquetes del sistema:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi  Shell
1 $ sudo apt-get update && sudo apt-get upgrade
```

A continuación se necesitará instalar algunas herramientas de desarrolladores, como CMake, el cual ayudará en el proceso de instalación de OpenCV:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi  Shell
1 $ sudo apt-get install build-essential cmake pkg-config
```

Luego, se debe instalar algunos paquetes de entrada/salida de imágenes que permitirá cargar archivos con distintos formatos de imagen desde el disco. Ejemplos de formatos incluidos son JPEG, PNG, TIFF, entre otros:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi  Python
1 $ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
```

Así como se necesita paquetes de entrada/salida de imágenes, se necesitará paquetes de entrada/salida de video. Estas librerías permitirán leer archivos de video de distintos formatos desde el disco:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
1 $ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
2 $ sudo apt-get install libxvidcore-dev libx264-dev
```

La librería de OpenCV viene con un submódulo llamado *highgui,* el cual se usa para mostrar imágenes en la pantalla y construir interfaces gráficas de usuario básicas.

Para compilar el módulo *highgui* se necesita instalar la librería de desarrollo GTK:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
1 $ sudo apt-get install libgtk2.0-dev libgtk-3-dev
```

Varias operaciones dentro de OpenCV (llamadas operaciones matrices) pueden ser optimizadas instalando algunas dependencias extra:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
1 $ sudo apt-get install libatlas-base-dev gfortran
```

Estas librerías de optimización son importantes, especialmente para dispositivos tales como la Raspberry Pi.

Por último, se instala los archivos de encabezado para Python 2.7 y Python 3 de tal forma que se pueda compilar en OpenCV con enfoque en Python:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
1 $ sudo apt-get install python2.7-dev python3-dev
```

Si se trabaja con una versión recién instalada de Raspbian, es posible que estas versiones de Python ya se encuentren en su versión más reciente.

Si se omite este paso, es probable que se presente un error de archivo no encontrado relacionado con el archivo de cabecera *Python.h* cuando se ejecute el comando *make* para la compilación de OpenCV.

Paso 3: Descarga del código fuente de OpenCV

Ya con las dependencias instaladas, se debe continuar con la adquisición del archivo de OpenCV desde su repositorio oficial. (***Nota:*** *Para futuras versiones de*

*OpenCV, solamente se debe reemplazar **3.3.0** con el número de la última versión*)
https://opencv.org/

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi
1 $ cd ~
2 $ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip
3 $ unzip opencv.zip
```

Al necesitar la versión completa de OpenCV para tener acceso a todas sus características, se deberá adquirir el repositorio *opencv_contrib*:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry P
1 $ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip
2 $ unzip opencv_contrib.zip
```

***Nota:** Asegurarse de que las versiones de opencv y opencv_contrib sean las mismas (en este caso 3.3.0). Si las versiones no coinciden, pueden ocurrir errores de ejecución o sincronización en el proceso de compilación*.

Paso 4: Extensiones de Python

Antes de empezar a compilar el OpenCV en la Raspberry Pi 3 se necesitará instalar *pip*, un gestor de paquetes de Python:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi
1 $ wget https://bootstrap.pypa.io/get-pip.py
2 $ sudo python get-pip.py
3 $ sudo python3 get-pip.py
```

Quizá se presente un mensaje emergente indicando que *pip* se encuentra actualizado, pero es recomendable no omitir este paso.

Instalación de NumPy

A continuación se realiza la instalación correspondiente a NumPy, el cual es un programa utilizado para procesamiento numérico:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi
1 $ pip install numpy
```

La instalación de NumPy podría tomar un tiempo en completarse.

Paso 5: Compilación e instalación de OpenCV

Luego de completar los pasos anteriores, se procederá con la compilación e instalación de OpenCV. Se utilizará el comando CMake para el proceso:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
1 $ cd ~/opencv-3.3.0/
2 $ mkdir build
3 $ cd build
4 $ cmake -D CMAKE_BUILD_TYPE=RELEASE \
5     -D CMAKE_INSTALL_PREFIX=/usr/local \
6     -D INSTALL_PYTHON_EXAMPLES=ON \
7     -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules \
8     -D BUILD_EXAMPLES=ON ..
```

Antes de dirigirse al paso principal de compilación, asegurarse de examinar la salida de CMake tal y como se muestra a continuación (en la misma ventana de instalación, luego de realizar el paso anterior se desplegarán varias líneas, regresar un poco para verlo). Para que el proceso de compilación de OpenCV en Python 2 como en Python 3 sea posible, hay que fijarse que las secciones marcadas incluyan rutas válidas hacia los parámetros *Interpreter*, *Libraries*, *numpy* y *packages path*:

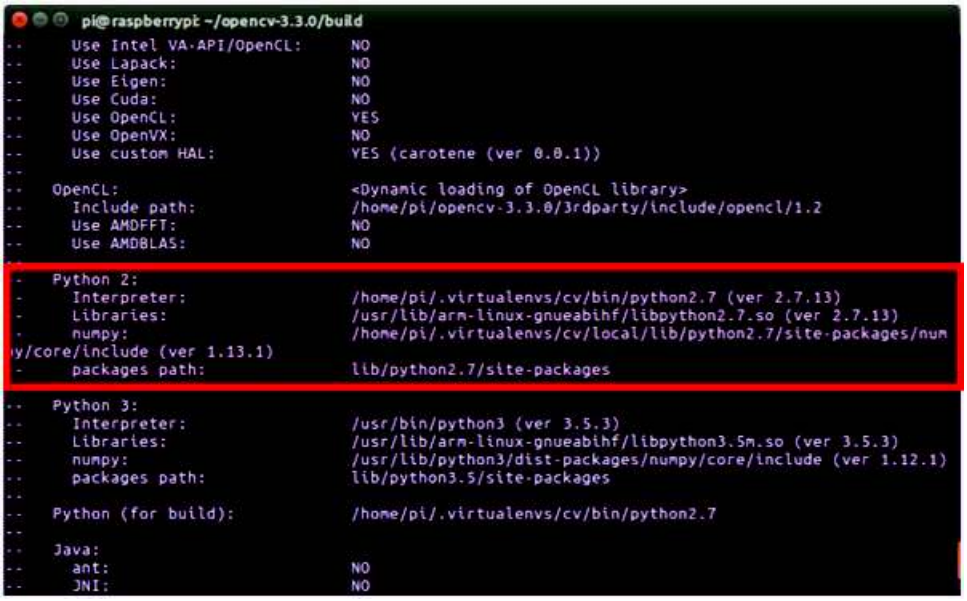


Figura D12: Sección de parámetros para Python 2 [34]

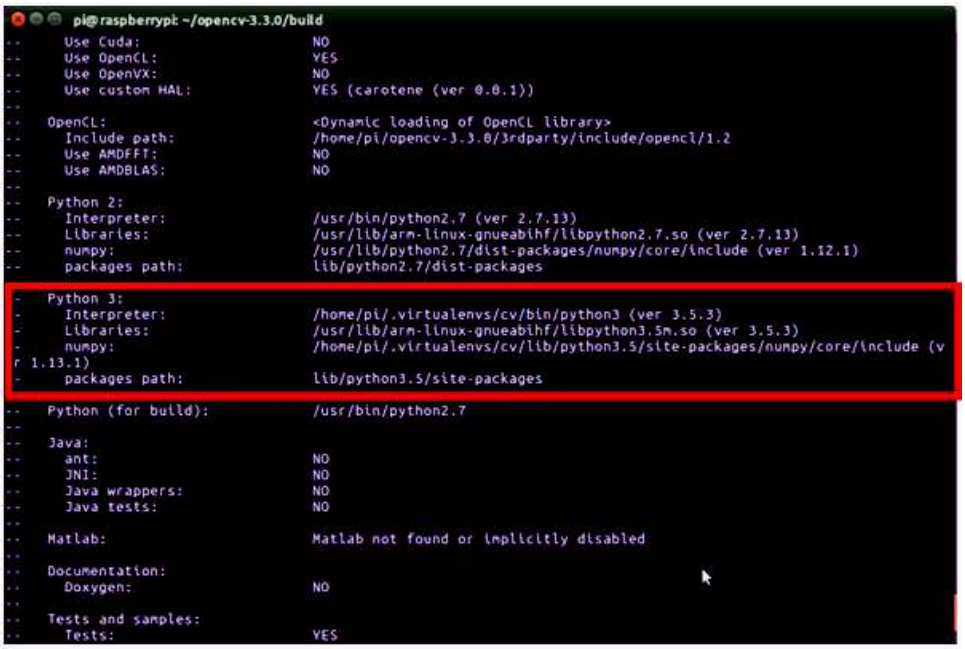


Figura D13: Sección de parámetros para Python 3 [34]

Finalmente se procederá a la compilación de OpenCV, para ello se debe ingresar el siguiente comando:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi    Shell
1 $ make -j4
```

Nota: El término *–j4* hace referencia a la cantidad de núcleos del procesador que se utilizará para la compilación (en este caso 4 núcleos). Con ello se determina una mayor rapidez del proceso.

Este punto sin duda alguna es el más tardío de la instalación (alrededor de 4 horas), por lo que solamente quedará esperar hasta que el proceso haya terminado y se muestre de esta forma:

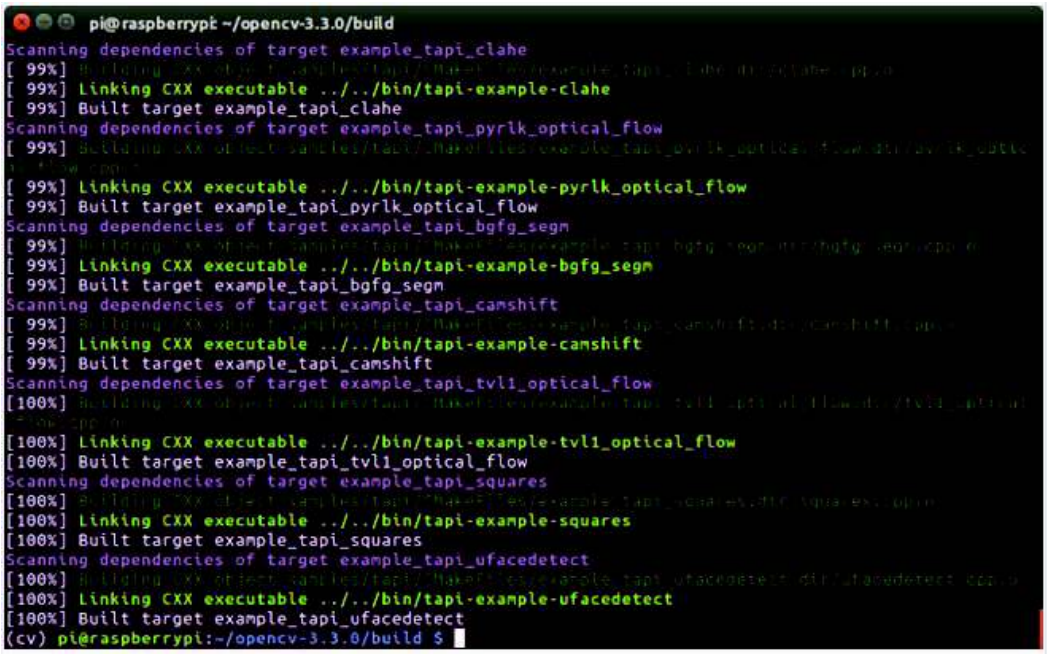


Figura D14: Fin del proceso de compilación [34]

Como nota adicional, es probable que la compilación se detenga debido a errores que se dan en medio del proceso. Si se da el caso, se debe volver a ingresar el comando *make –j4* para empezar nuevamente la compilación hasta que ésta se complete exitosamente.

Una vez terminada la compilación, se procede con la instalación:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi        Shell
1 $ sudo make install
2 $ sudo ldconfig
```

Para poder comprobar si se instaló tanto OpenCV como NumPy, se ingresa en una nueva ventana de LXTerminal la siguiente línea: ***dpkg --get-selections***

Con este comando se realizará el pedido para mostrar una lista de todos los programas instalados en la Raspberry Pi. Una vez ingresado el comando solamente habrá que buscar en la lista que consten los nombres OpenCV y NumPy.

ANEXO D3

INSTALACIÓN DE APACHE, MYSQL Y PHPMYADMIN

APACHE, MYSQL Y PHPMYADMIN

Con ayuda de estos tres programas se busca crear un servidor web en la Raspberry Pi, el cual será utilizado para el manejo de lo que son bases de datos. [35]

Instalación

Nuevamente dirigirse a una ventana de LXTerminal para ingresar los comandos respectivos para la instalación.

Primeramente, se debe configurar una IP estática en el equipo de acuerdo al punto de acceso a la red (Ethernet o WiFi). Para el ejemplo se tomarán los siguientes parámetros:

```
address 192.168.0.100
netmask 255.255.255.0
gateway 192.168.0.1
```

"Para cambiar la configuración de red en la Raspberry Pi, se debe dar clic derecho sobre el ícono de red ubicado en la parte inferior derecha de la pantalla y seleccionar la primera opción de la lista emergente".

Luego, se debe realizar una actualización de repositorios y programas de la Raspberry Pi. Para ello se ingresan los comandos:

```
sudo apt-get update
sudo apt-get upgrade
```

Una vez actualizada la Raspberry Pi, se procede con la instalación de Apache escribiendo el código:

```
sudo apt-get install apache2
```

Para realizar la comprobación de que el proceso de instalación va por buen camino, abrir una ventana del explorador web Chromium y escribir la dirección IP fija establecida con anterioridad en el buscador. Debe mostrarse lo siguiente:

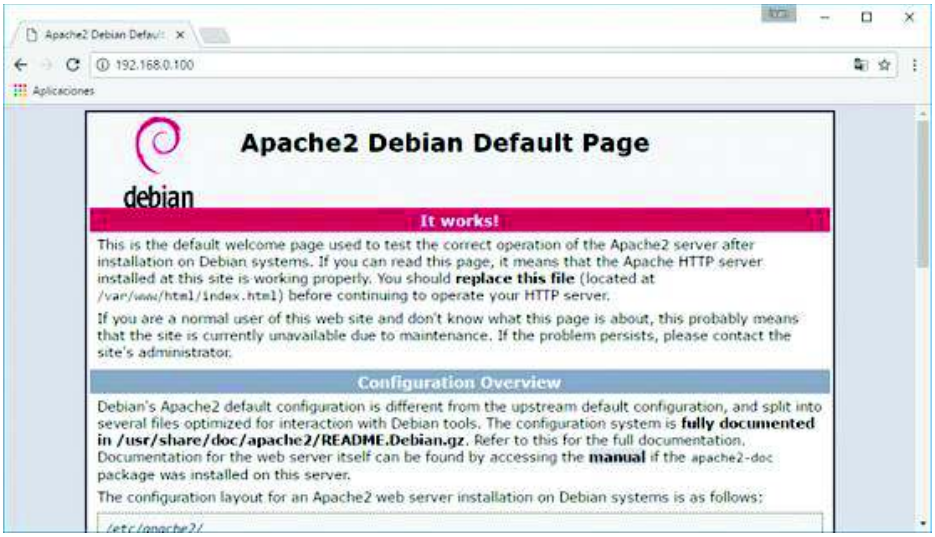


Figura D15: Ventana de demostración del funcionamiento de Apache [35]

Después de comprobar que el procedimiento va por buen camino, continuar con la instalación de PHP, cuyo propósito es la elaboración de sitios web dinámicos. Se escribe en LXTerminal:

```
sudo apt-get install php5 libapache2-mod-php5
```

Luego de terminarse la instalación, reiniciar el equipo.

Al haberse reiniciado el equipo, comprobar el correcto funcionamiento de PHP ingresando en LXTerminal la línea:

```
sudo nano /var/www/html/info.php
```

A continuación quedará así:

```
<?php
      phpinfo();
?>
```

Lo siguiente que se debe hacer es abrir Chromium e ingresar nuevamente la IP fija añadiendo */info.php* luego de ésta. Debe verse de la siguiente manera:

Figura D16: Ventana de demostración del funcionamiento de PHP [35]

A partir de aquí se continuará con la instalación de MySQL, programa conocido por ser un potente gestor de base de datos. Para ello se ingresa en el terminal el comando:

```
sudo apt-get install mysql-server mysql-client php5-mysql
```

Tomar en cuenta que en medio de la instalación se solicitará una contraseña, para lo cual se debe ingresar una contraseña que debe ser recordada para usarse posteriormente. Al concluir la instalación, reiniciar la Raspberry.

Una vez que se enciende el equipo, abrir nuevamente LXTerminal para seguir con la instalación de PhPMyAdmin. Este programa permite un fácil manejo de la administración de MySQL por medio de Internet. Se ingresa la siguiente línea:

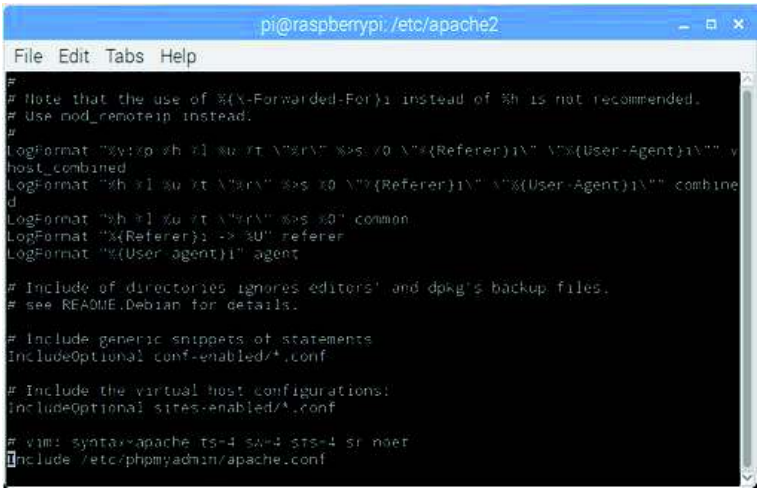
```
sudo apt-get install php5-mysql phpmyadmin
```

Mientras se instala, aparecerá una pregunta referente a qué tipo de servidor se está utilizando. Se debe marcar la opción Apache. Luego, mostrará una opción en la que señalará si se desea configurar una base de datos, para lo cual se deberá introducir la contraseña anteriormente creada en el momento de instalar MySQL. Seguido, se deberá crear una contraseña para ingresar a PhPMyAdmin. Se recomienda utilizar una contraseña común para todo el proceso de instalación.

Al terminar la instalación, escribir el siguiente código:

```
sudo nano /etc/apache2/apache2.conf
```

Al ejecutar este código se abrirá un fichero en el cual se debe agregar al final del mismo la línea de texto ***Include /etc/phpmyadmin/apache.conf*** como se muestra a continuación:



Realizado todo el proceso, dirigirse a Chromium, escribir en el buscador la dirección IP asignada seguido de /phpmyadmin. Si todo ha salido bien se visualizará una pantalla en donde se solicitará credenciales tales como el nombre de usuario y contraseña. Por defecto el usuario es *root* y la contraseña será la que se haya configurado en pasos anteriores. Si no reconoce alguno de los dos parámetros, intentar nuevamente después de reiniciar la Raspberry.

Figura D17: Cuadro de inicio de sesión de PhPMyAdmin [35]

Figura D18: Ventana de PhPMyAdmin [35]

Una vez dentro ya se puede comenzar a crear bases de datos según sea necesario.

ANEXO D4

INSTALACIÓN DE JULIUS Y RHYTHMBOX

JULIUS Y RHYTHMBOX

Esta instalación incluye todas las librerías y paquetes necesarios, tanto para el funcionamiento como para el trabajo conjunto entre ambos programas [36].

Instalación

Del mismo modo que se instaló OpenCV, se utilizará LXTerminal para ingresar los comandos necesarios para dar lugar a la instalación del software previamente mencionado.

Primeramente, instalar un paquete con el siguiente comando:

```
sudo apt install libasound2-dev
```

La instalación de este paquete es fundamental debido a que si se realiza la compilación sin éste, mostrará el siguiente error:

```
configure: error: no ALSA header!
configure: error: ./configure failed for libsent
```

Para continuar, proceder con la descarga de Julius mediante el siguiente enlace, el cual es la fuente oficial del desarrollador del programa.

```
wget https://github.com/julius-speech/julius/archive/master.zip
```

Luego de descargado el programa, renombrar su carpeta contenedora para una mejor identificación del mismo. Se realiza con el comando:

```
mv master.zip julius-master.zip
```

A continuación, descomprimir el archivo:

```
unzip julius-master.zip
```

Después, ingresar en el directorio donde se encuentra el programa:

```
cd julius-master
```

En este punto, compilar e instalar Julius con ayuda de los comandos:

```
sudo ./configure --with-mictype=alsa
sudo make
sudo make install
```

Para comprobar que el procedimiento se está realizando correctamente, escribir *julius* y dar enter para ejecutar. Debe visualizarse:

```
pi@raspberrypi:~/julius-master $ julius
Julius rev.4.4.2 - based on
JuliusLib rev.4.4.2 (fast)  built for armv7l-unknown-linux-gnueabi

Copyright (c) 1991-2016 Kawahara Lab., Kyoto University
Copyright (c) 1997-2000 Information-technology Promotion Agency, Japan
Copyright (c) 2000-2005 Shikano Lab., Nara Institute of Science and Technology
Copyright (c) 2005-2016 Julius project team, Nagoya Institute of Technology

Try '-setting' for built-in engine configuration.
Try '-help' for run time options.
```

Como siguiente paso, comprobar la detección del micrófono por parte de la Raspberry. En este caso se utiliza un micrófono USB por lo que éste deberá verse en la lista de dispositivos conectados solicitada con el comando *lsusb:*

```
pi@raspberrypi:~/julius-master $ lsusb
Bus 001 Device 005: ID 248a:8566
Bus 001 Device 004: ID 0d8c:013c C-Media Electronics, Inc. CM108 Audio Controller
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Notar que en *Device 004* se muestra la identificación del micrófono. Ahora, comprobar el número de tarjeta, así como el número de dispositivo asignado por el equipo. Para ello, usar el comando *arecord –l:*

```
pi@raspberrypi:~/julius-master $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: Device [USB PnP Sound Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Se puede visualizar que en este caso el número de tarjeta (*card*) es 1 y el número de dispositivo (*device*) es 0. Dichos parámetros son importantes para más adelante.

Luego, configurar la tarjeta designada en el paso anterior con el comando *amixer sget Mic –c 1* para ser usada:

```
pi@raspberrypi:~/julius-master $ amixer sget Mic -c 1
Simple mixer control 'Mic',0
  Capabilities: cvolume cvolume-joined cswitch cswitch-joined
  Capture channels: Mono
  Limits: Capture 0 - 16
  Mono: Capture 0 [0%] [0.00dB] [on]
```

Seguidamente, ajustar el volumen de recepción del micrófono al 100% con ayuda del comando *amixer sset Mic 16 –c 1*:

```
pi@raspberrypi:~/julius-master $ amixer sset Mic 16 -c 1
Simple mixer control 'Mic',0
  Capabilities: cvolume cvolume-joined cswitch cswitch-joined
  Capture channels: Mono
  Limits: Capture 0 - 16
  Mono: Capture 16 [100%] [23.81dB] [on]
```

Después de realizar las configuraciones anteriores, instalar ALSADEV, el cual es una variable de entorno que permite identificar el dispositivo que se usará como elemento receptor, en este caso el micrófono. Ingresar uno por uno los siguientes comandos:

```
export ALSADEV="plughw:1,0"
echo $ ALSADEV
```

Como siguiente paso, probar Julius. Para ello, salir del directorio actual (julius-master) con el comando *cd ../*

Para verificar que está fuera del directorio de julius-master, escribir el comando que se muestra:

```
pi@raspberrypi:~ $ pwd
/home/pi
```

Como se observa, el directorio actual es */home/pi*. En este mismo directorio, crear un nuevo directorio correspondiente al paquete Quickstart-Linux que se utilizará posteriormente:

```
mkdir Quickstart-Linux
```

Para revisar la creación del nuevo directorio, escribir *ls* y dar enter. Se mostrará:

```
pi@raspberrypi:~ $ ls
julius-master  Quickstart-Linux
```

Luego, ingresar en el directorio de Quickstart-Linux para seguir con la descarga de más paquetes necesarios:

```
cd Quickstart-Linux
```

Se procede con la descarga del paquete Julius-4.3.1-Quickstart-Linux_(0.9.0).tgz que permitirá probar la instalación y el reconocimiento del programa Julius:

```
wget http://www.repository.voxforge1.org/downloads/Main/Tags/Releases/0.9.0/Julius-4.3.1-Quickstart-Linux_\(0.9.0\).tgz
```

Seguido, descomprimir el contenido adquirido:

```
tar -xvzf Julius-4.3.1-Quickstart-Linux_\(0.9.0\).tgz
```

Si se presenta algún error en la ejecución de la línea, suprimir las barras invertidas dentro y fuera de los paréntesis.

Al haber terminado la descompresión, escribir *ls –la* para observar los archivos adquiridos. Dentro de lo conseguido deberá constar el archivo *Sample.jconf*, el cual servirá para probar Julius:

```
pi@raspberrypi:~/Quickstart-Linux $ ls -la
acoustic_model_files
GRAMMAR_NOTES
LICENSE
etc
julius
README
grammar
Julius-4.3.1-Quickstart-Linux_(0.9.0).tgz
Sample.jconf
```

Después de constatar que el archivo *Sample.jconf* está en lo mostrado, se procede con su ejecución. Utilizar la siguiente línea:

```
julius -input mic -C Sample.jconf
```

Una vez ejecutado el comando, aparecerá una serie de información para luego mostrar la línea que indicará si Julius está funcionando:

```
<<< please speak >>>
```

A partir de aquí, solamente resta pronunciar palabras para interactuar con la Raspberry. Palabras del software previamente configuradas se pueden ver dentro de los archivos *sample.dict* y *sample.voca* que se encuentran en el directorio *grammar* de la carpeta de Julius.

Por defecto, la ejecución de Julius está enlazada principalmente por el comando a continuación, el cual siempre debe ser ejecutado previamente. Caso contrario dará un error. El punto a tomar en cuenta es que este comando debe ser escrito cada vez que la Raspberry sea reiniciada o encendida luego de estar apagada:

```
export ALSADEV="plughw:1,0"
```

Para evitar repetir el mismo proceso múltiples veces, se recomienda crear un *script* (archivo de ejecución) que evite el trabajo de escribir todos los comandos para abrir Julius y que solamente tenga que darse doble clic a dicho archivo para ejecutar el software.

Un ejemplo de este script se lo puede encontrar en la segunda práctica del uso de los módulos Raspberry Pi, correspondiente al control por voz.

La siguiente parte de la instalación está dirigida a Rithmbox, siguiendo estos pasos con el fin de asociarlo con el control por voz. Para ello se procederá a la descarga de un ejemplo que aplique estas condiciones.

Comenzar con la creación de un directorio de trabajo con el nombre de *reproductor*:

```
mkdir reproductor
```

Luego, ingresar en el directorio creado:

```
cd reproductor
```

Una vez dentro del directorio, escribir la siguiente línea que será para descargar el ejemplo mencionado anteriormente:

Después de haber descargado el ejemplo, se procede con su instalación:

```
sudo dpkg --install julius-voxforge_0.1.1~daily20130206-0ubuntu1_all.deb
```

Para liberar espacio, luego de que la instalación haya sido realizada, borrar la descarga:

```
rm julius-voxforge_0.1.1~daily20130206-0ubuntu1_all.deb
```

Seguidamente, realizar el copiado de archivos necesarios; en este caso, el archivo de configuración:

```
cp /usr/share/doc/julius-voxforge/examples/julian.jconf.gz ./
```

A continuación, desempaquetar el archivo:

```
gunzip julian.jconf.gz
```

Posteriormente, copiar dos archivos más correspondientes a la gramática y a la aplicación en Python:

```
cp /usr/share/doc/julius-voxforge/examples/controlapp/mediaplayer* ./
cp /usr/share/doc/julius-voxforge/examples/controlapp/command.py ./
```

Realizado todos los pasos anteriores podrá observarse que dentro del directorio *reproductor* se encuentran 4 archivos:

- *command.py* = Script que toma la salida de Julius para darle funciones personalizadas.
- *julian.jconf* = Configuración básica de Julius.
- *mediaplayer.grammar* = Fichero de gramática.
- *mediaplayer.voca* = Fichero de vocabulario.

Una vez comprobada la ubicación de los archivos, proceder con la compilación. Para ello, utilizar el comando *mkdfa* junto con el nombre del archivo sin extensión. Tomar en cuenta que los archivos ***.grammar*** y ***.voca*** deben tener el mismo nombre.

```
mkdfa.pl mediaplayer
```

Luego de ser ejecutado el comando, mostrará lo siguiente:

```
pi@raspberrypi:~/reproductor $ mkdfa.pl mediaplayer
mediaplayer.grammar has 1 rules
mediaplayer.voca    has 4 categories and 9 words
---
Now parsing grammar file
Now modifying grammar to minimize states[-1]
Now parsing vocabulary file
Now making nondeterministic finite automaton[5/5]
Now making deterministic finite automaton[5/5]
Now making triplet list[5/5]
4 categories, 5 nodes, 4 arcs
-> minimized: 5 nodes, 4 arcs
---
generated: mediaplayer.dfa mediaplayer.term mediaplayer.dict
```

Al terminar la compilación se podrá observar que dentro del directorio reproductor se han creado 3 archivos más. Estos son: ***mediaplayer.dfa***, ***mediaplayer.dict*** y ***mediaplayer.term***.

Después, editar el archivo de configuración julian.jconf de modo que pueda ser preparado para su uso. Se debe realizar el reemplazo de las líneas 36 y 37 dentro de éste como se indica a continuación:

Las líneas a ser cambiadas originalmente son:

```
-dfa sample.dfa
-v sample.dict
```

Por lo que una vez ubicadas deben ser reemplazadas por las líneas:

```
-dfa mediaplayer.dfa
-v mediaplayer.dict
```

Además, debe eliminarse el símbolo de comentario (**#**) de las líneas 162 (*-input mic*) y 242 (*-quiet*).

Concluido con todo el proceso, continuar con la ejecución de Julius. Puede darse un problema de privilegios en el momento de hacerlo funcionar, para lo cual se escribe la siguiente línea cuya función es la de corregir los permisos:

```
sudo chmod 777 -R ./
```

Como se mencionó en pasos anteriores, se debe ingresar el comando que permitirá el funcionamiento de la tarjeta al igual que el micrófono. Dicho comando es:

```
export ALSADEV="plughw:1,0"
```

Finalmente, continuar con la ejecución y utilización de Julius:

```
julius -C julian.jconf | ./command.py
```

Como se puede observar, todos estos procesos de instalación de software fueron realizados para que el módulo Raspberry Pi cumpla con su cometido de estar en las mejores condiciones para realizar las prácticas de laboratorio incluidas con éste.

Como nota adicional y para fines prácticos, se entregará un respaldo de software al laboratorio de Microcontroladores, el cual contendrá el sistema operativo Raspbian junto con el resto de programas mencionados en esta sección en el caso de que se necesite reinstalar en una microSD nueva o existente.