ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN DISPOSITIVO IDENTIFICADOR DE BILLETES PARA PERSONAS NO VIDENTES

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

DIEGO IVAN TIPÁN MONDRAGÓN
Miguelino75@hotmail.com

DIRECTOR: ING. VIVIANA PARRAGA MSc.
viviana.parragav@epn.edu.ec

CODIRECTOR: ING. MÓNICA VINUEZ RHOR MSc.
monica.vinueza@epn.edu.ec

Quito, Febrero 2019

DECLARACIÓN

Yo, Diego Iván Tipán Mondragón, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis 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.

Diego Iván Tipán Mondragón

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Diego Iván Tipán Mondragón, bajo nuestra supervisión.

Ing. Viviana Parraga MSc.	**Ing. Mónica Vinueza MSc.**
DIRECTOR DEL PROYECTO	**CODIRECTOR DEL PROYECTO**

DEDICATORIA

Dedico este proyecto a mi familia, en especial a mi madre y a mi difunto padre, quienes han sido mi apoyo durante esta etapa universitaria y gracias a los valores de: honestidad, respeto, respondabilidad y puntualidad que me han inculcado he alcanzado este logro importante en mi vida profesional.

AGRADECIMIENTOS

A la Escuela Politecnica Nacional y a la Escuela de Formación de Tecnólogos por brindarme la experiencia del conocimiento a través de sus docentes y por haberme formado en carácter para enfrentarnos al mundo laboral.

A la Ing. Viviana Parraga por aceptar ser la auspiciante de mi proyecto, por el tiempo y la paciencia brindados para guiarme en la culminación del mismo.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto describe la construcción de un prototipo de identificación de denominación de billetes como ayuda para introducir a las personas con discapacidad visual en al ámbito laboral mediante la fundación PROCODIS.

Para el desarrollo e implementación del prototipo se tiene algunos pasos: primero, se realizó el diseño lógico o esquema circuital, a continuación, se investigó acerca de las librerías de *openCV* aplicadas a *visión artificial*, para su posterior utilización en el código de programación en *Python.* Luego, se determinó las características de cada uno de los componentes y su conexión con la *Raspberry pi*, posterior a esto, se determinó el diseño físico del mismo, tomando en cuenta características como, por ejemplo: las dimensiones de la caja de madera y la distribución de las partes electrónicas para la posterior implementación.

Finalmente, se presenta la implementación y funcionamiento del prototipo, el cual se basa en la identificación de la denominación del billete a través de una cámara conectada a la *Raspberry pi*, la cual captura una imagen, para procesarla en busca de patrones referenciales de identificación, es decir, analiza la imagen que se capturó con respecto a la base de datos de imágenes guardadas en la memoria *microSD*.

Con la ayuda de un parlante y mediante un mensaje de voz se permite informar a una persona no vidente el valor del billete consultado y de esta forma se detecta la denominación del mismo.

Palabras clave: *Raspberry pi*, programación, *visión artificial* y procesamiento de imágenes digitales.

ABSTRACT

The present project describes the construction of a prototype of identification of the denomination of the bills like the help for the introduction of people with visual disability in the occupational area through the PROCODIS foundation.

The development and implementation of the prototype there are some steps: first, we made the logical design and investigated about the libraries of OpenCV applied to artificial vision, then use them in the code of programming using Python. Then, it was determined the characteristics of each of the components of the device and its connection with the Raspberry Pi, to finally determine the physical design of the same, taking into account characteristics such as: the dimensions of the wooden box and the distribution of electronic parts for later implementation.

Finally, the implementation and operation of the prototype is presented, which is based on the identification of the denomination of the bill through a camera connected to the Raspberry pi, which captures an image, to process it in search of referential patterns of identification, therefore, it analyzes the image that was captured with respect to the database of images stored in memory.

The speaker to inform a blind person the value of the bill consulted by means of a voice message.

Keywords: Raspberry pi, programming, artificial vision and digital image processing

1. INTRODUCCIÓN

Este proyecto ha sido desarrollado para ayudar a las personas no videntes y con baja visión de esta fundación, debido a la dificultad que tienen para reconocer la denominación de billetes. Mediante esta herramienta se da ayuda tanto a sus capacitadores, capacitados, así como al resto de personas no videntes que pertenecen a esta institución.

Una de las actividades económicas que realiza la fundación para el sustento de las personas discapacitadas es un *Cyber*, donde, uno de los principales problemas que se presentaba era la identificación de la denominación de billetes que es percibido y entregado por las personas no videntes al atender este lugar.

Aprovechando la virtud de las personas no videntes, quienes tienen el sentido del tacto más desarrollado, y debido a que los billetes tienen impresiones en relieve, que indican su *autenticidad*, estas personas son capaces de reconocer si un billete es o no falso, recorriendo dichos puntos con la yema de los dedos, razón por la cual este proyecto no contempla la parte de autenticación del billete [1]

Mediante un proyecto de vinculación entre la Escuela de Formación de Tecnólogos de la Politécnica Nacional y la fundación PROCODIS, se pudo visitar dicha fundación, donde se observó la necesidad de desarrollar el prototipo con el objeto de mejorar la calidad de vida de los discapacitados visuales, además, se busca con este proyecto introducir a las personas con discapacidad visual en al ámbito laboral.

Una de las líneas de investigación y desarrollo muy interesantes, es lo relativo a la “imitación” de nuestros órganos de los sentidos: audición y visión. En ello, no sólo se espera dotar a un computador de la capacidad de “percibir” sonidos, sino de reconocer, es decir, de “identificar” lo percibido. La *visión artificial* por computadora es una disciplina en creciente auge con multitud de aplicaciones, como inspección automática y reconocimiento de objetos.

Debido a que el reconocimiento de objetos depende crucialmente de las técnicas utilizadas, por esta razón, para la ejecución de este proyecto, principalmente se realizó una breve revisión de las técnicas más empleadas para el procesamiento de imágenes digitales, posterior a ello, se determinó los elementos a utilizar y sus respectivos requerimientos, a continuación, se investigó la tarjeta electrónica que se acople de mejor manera con las caracteristicas tanto físicas como eléctricas de estos elementos.

Para el diseño del dispositivo, se utilizó un *software* de diseño electrónico asistido por computadora, con el objetivo de determinar la distribución de los elementos, tomando en cuenta las medidas del *case* de madera en el que fue colocado, finalmente, se realizó la implementación del dispositivo y se realizaron pruebas.

Este proyecto utiliza una tarjeta electrónica, que tiene las características de un computador para poder procesar imágenes, está procesa la información generada por la cámara *Raspicam* que es la encargada de capturar las imágenes y enviar la información hacia la *tarjeta* y mediante dos parlantes se informará la denominación del billete consultado.

Este dispositivo es de fácil utilización, ya que, estas personas solo deben introducir el billete a ser consultado dentro de la caja y mediante un parlante el dispositivo informa la denominación del billete.

1.1. Marco Teórico

Tarjeta Raspberry pi 3

Desarrollada por la Universidad de *Cambridge*, cuyo objetivo era fomentar la enseñanza de las ciencias de la computación en países de escasos recursos, es una placa computadora de bajo costo con características que permiten desarrollar proyectos de investigación sobre esta plataforma, es decir, es una tarjeta diseñada sobre un microprocesador ARM (*Advanced RISC Machines*) de 1.2 Ghz con la arquitectura de 64 bits, una memoria *RAM* de 1 GHZ y un procesador gráfico (GPU)

Video Core de 400 MHZ, además, cuenta con un *Chip* que permite integrar las tecnologías *Bluetooh* 4.1 y *WIFI* 802.11 b/g/n, además, posee un puerto *Ethernet* de 10/100 Mbps para una conexión a *Internet*.

Dispone de un lector/ranura el cual permite el uso de una tarjeta *microSD* (*Secure Digital*) de 16 *GB* de almacenamiento, la cual funcionará como disco duro y en la que se instala el sistema operativo *Raspbian* para diferentes aplicaciones [2].

Integra un puerto de video *HDMI* (*High-Definition Multimedia Interface*) para conectar una pantalla, a través, de un adaptador *HDMI-VGA* (*Video Graphics Array*), además, integra un puerto *CSI* (*Camera Serial Interface*) para conectar una cámara, a través del bus de 12 hilos.

Además, posee un conector *Jack* de 3.5 mm para la salida de audio, también cuenta con cuatro puertos *USB* para conectar periféricos de entrada y salida como: teclado, mouse, memoria *flash USB*, adicionalmente, tiene puertos *GPIO* (*General Purpose Input/Output*), estos pueden ser usados como puertos de entrada y salida de datos [3].

Las partes de la tarjeta *Raspberry pi* se observa en la Figura 1.1.

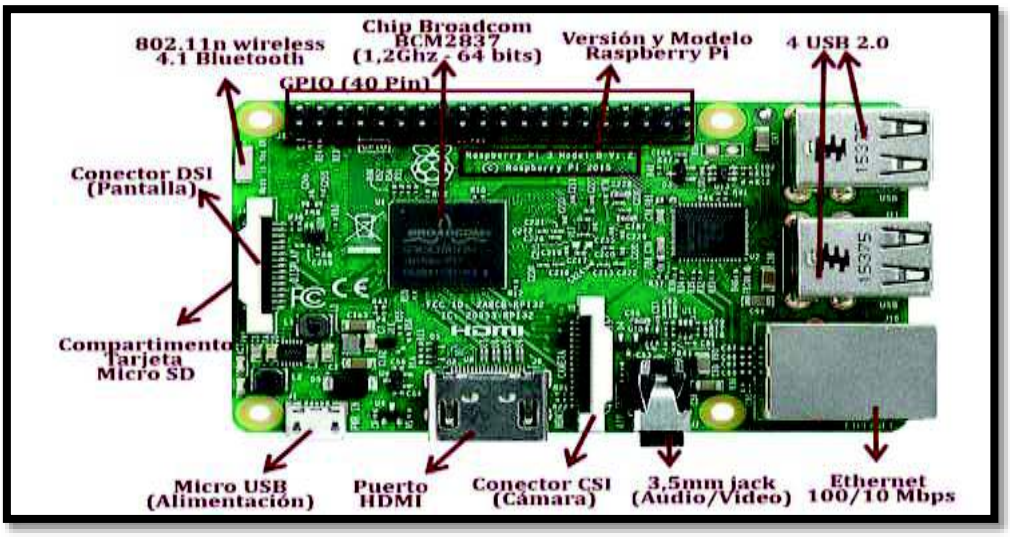

Figura 1. 1.Partes de la *Raspberry pi* [3].

La tarjeta no dispone de botón de encendido o apagado, razón por la que la energía le llega mediante un conector *microUSB* de 5 voltios (V) a 2.5 Amperios (A), el consumo de la placa es de 700 miliAmperios (m A) [2].

Puertos GPIO

GPIO (*General Purpose Input Output*) es una interfaz en la cual se puede conectar diferentes dispositivos electrónicos, leer sensores y datos de módulos a través de: comunicación *serial UART* (*Universal Asynchronous Receiver-Transmitter*). Maneja valores de 0V a 5V. En la Figura 1.2. se muestran los puertos GPIO [6].

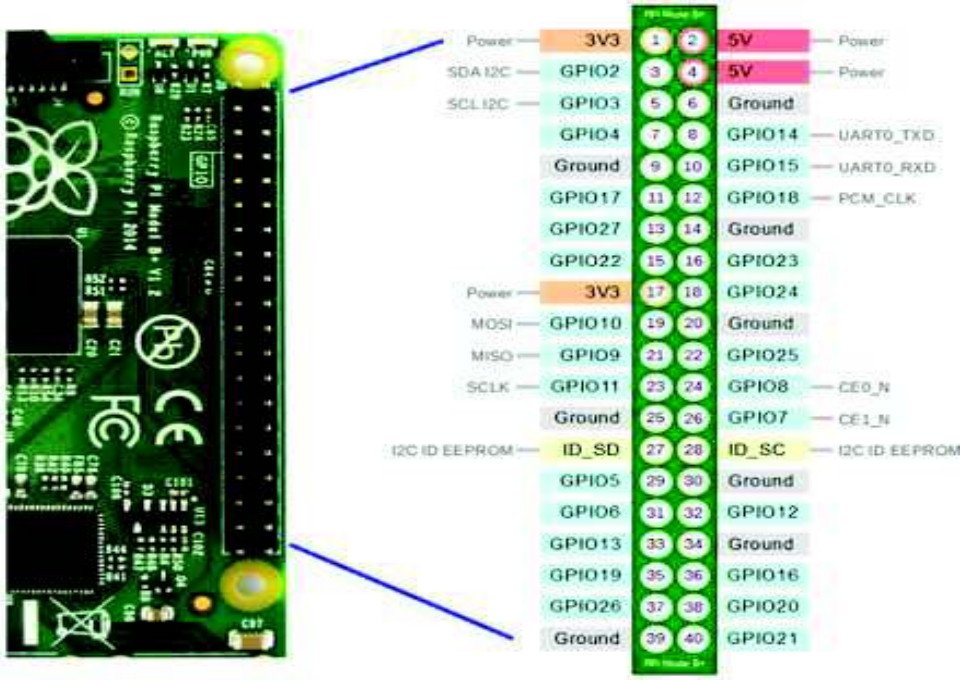

Figura 1. 2. Pines GPIO de la tarjeta *Raspberry pi 3* [6].

Sistema Operativo *Raspbian*

Es uno de los sistemas operativos adaptados a la plataforma *Raspberry pi*, la característica principal es que éste puede ser cargado en una *micro SD*. Desarrollado a partir de *Debian* de *Linux*, es de libre distribución y proporciona un manejo de la

plataforma, mediante un escritorio y el uso de programas básicos, como en una *PC*. Proporciona un entorno de fácil manejo para la expansión del aprendizaje en proyectos de aplicación de *software* y *hardware*, Figura 1.3. [7].

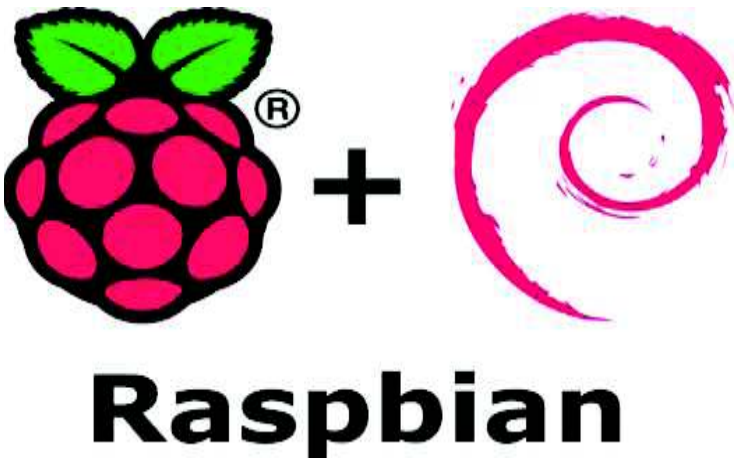

Figura 1. 3. *Raspbian_Logo* [6].

Lenguaje de Programación *Python*

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel con lo que permite la integración de sistemas, de manera más eficiente y rápida. Maneja programación orientada a objetos y también es compatible con un gran número de librerías [6].

OpenCV

Es una biblioteca informática de código abierto, desarrollada originalmente por *Intel*, la biblioteca es multiplataforma y compatible con *Linux.* Contiene varias funciones que abarcan una gran gama de áreas en el proceso de visión, como se puede destacar: reconocimiento de objetos, reconocimiento facial, calibración de cámaras, visión estéreo y visión robótica [7].

OpenCV fue originalmente escrita en C y esta interfaz C hace a *OpenCV* portátil para algunas plataformas específicas y busca reducir el número de líneas de código necesarias al código de la funcionalidad de la visión, así como reducir los errores comunes de programación tales como procesadores de señal digital [8].

Figura 1. 4. *OpenCV_Logo* [8].

Cámara *Raspicam*

Uno de los motivos principales de la elección de esta cámara, es ser propia de *Raspberry pi*. Se pudo haber elegido una cámara con conexión por *USB* para el proyecto, pero las pocas cámaras que ofrecen este tipo de conexión son las *webcams*, pero estas son de muy poca resolución de video y robustez [8].

La *Raspicam* por otro lado, tiene una resolución de 8 *Megapixeles* aproximadamente y la capacidad de capturar video a 3280 x 2464 *pixeles* y 90 fps (*cuadros por segundo*) , para ello, esta cámara utiliza un sensor de imagen *Sony IMX219*, diseñado a medida para la *Raspberry pi*, además, cuenta con un lente de foco fijo y se conecta directamente a la *Raspberry pi* mediante el cable de 15 *centímetros* (cm) fijado a las ranuras en su puerto serie (*CSI*) [9].

Su reducido tamaño como se observa en la Figura 1.5, permite que ocupe poco espacio en una *carcasa* que proteja a la *Raspberry pi* y a la *Raspicam*, pues la unión entre el puerto *CSI* de la *Raspberry* y la cámara es flexible, lo que permite colocar la

cámara casi en cualquier posición, por otra parte, su precio se mantiene igual que las cámaras *Webcams*, pero la *Raspicam* ofrece mayor resolución [8].

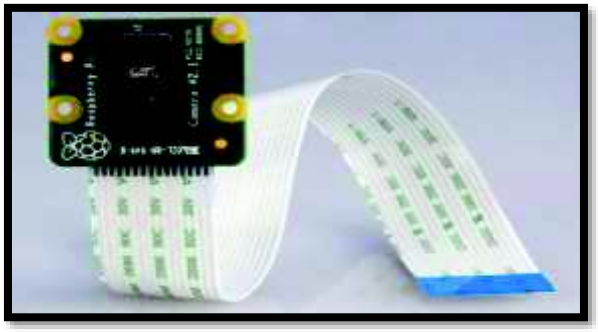

Figura 1. 5. Cámara *Raspicam* [8].

Parlantes *USB*

Estos parlantes son los que mejor se acoplan para el presente proyecto, debido a su reducido tamaño como se puede observar en la Figura 1.6, adecuado para colocarlos dentro del case de madera.

Además, cuentan con control de volúmen en el lateral y simplemente se conecta el terminal usb de la *Raspberry pi* para obtener la energía, también, posee altavoces de 50mm x 50mm (*milímetros*) [2].

Figura 1. 6. Parlantes *USB* [2].

Visión Artificial

La *visión artificial* es un procedimiento de adquisición de imágenes, mediante sistemas ópticos, donde se realiza el análisis automático de las mismas [10].

La entrada de un sistema de *visión artificial* es una imagen obtenida por un elemento de adquisición, mientras que su salida es una descripción de la escena obtenida a partir de la imagen.

Las etapas de un sistema de *visión artificial* son:

Captura: es la adquisición de imágenes digitales mediante algún tipo de sensor óptico [11].

Preprocesado: prepara la imagen convirtiendo las imágenes a color en escala de grises, eliminando las partes no útiles y realzando las zonas de interés [11].

Segmentación: proceso mediante el cual una imagen se descompone en regiones o elementos para su posterior interpretación, el proceso de segmentación evalúa si cada pixel de la imagen pertenece o no al objeto de interés [11].

Reconocimiento: distingue los diferentes objetos segmentados en función de sus características [11].

Adquisición e interpretación de imágenes

La imagen digital es una matriz bidimensional, cada elemento de la matriz se llama *pixel* (*picture element*), en el pixel se almacena la luminancia de las imágenes en niveles de gris (256 niveles), o intensidad de cada componente de una base de color *red, Green and blue* (RGB) en imágenes a color [11].

Técnicas de detección de imágenes

Para determinar la denominación de billetes se utilizó una de las técnicas de detección de imágenes digitales, las más usadas son: Contar el número de *pixeles* en blanco de la imagen**,** comprobar el *histograma* de la imagen y la detección de círculos mediante la transformada de *Hough.*

Despúes de revisar brevemente, se determinó que la técnica para reconocer la denominación de billetes que mejor se acopla al proyecto propuesto es:

Comprobar el *histograma* de la imagen: el objetivo de este método, es el de encontrar los valores característicos de la imagen que establecen la separación del objeto de interés, con respecto a las regiones que no pertenecen al mismo.

Entonces, si se divide el número de *pixeles* de la imagen, puede interpretarse como la verosimilitud de que un determinado nivel de gris aparezca en la imagen; se establece un mínimo número de pixeles en nivel de gris (0) y cuando se supere este valor (1) es que se ha excedido el umbral.

Éste método detecta solo *pixeles* en nivel de gris, no la forma, ni donde están ubicados, sin embargo, no son inconveniente para el presente proyecto [10].

2. METODOLOGÍA

Se realizó una breve investigación de las diferentes técnicas de *visión artificial*, utilizadas actualmente para el procesamiento de imágenes digitales que permiten determinar la denominación de billetes, y así hacer referencia a la que resulte ser más práctica y confiable.

Se investigó el uso, manejo y la relación entre: la cámara , tarjeta *Raspberry pi* y el parlante; que fueron los requerimientos necesarios para la implementación del dispositivo. También, la alimentación de corriente de los elementos, el lenguaje de

programación a ser usado en la tarjeta *Raspberry pi;* la pótencia necesaria, respuesta en frecuencia y sensibilidad para el correcto funcionamiento del parlante; la resolución, el rango focal, la luminosidad y la calidad del lente que precisa la cámara y así obtener los mejores resultados.

Para el diseño del prototipo se determinó usar una tarjeta *Raspberry pi* por la gran cantidad de información disponible en la *web*, su fácil uso, su costo económico y debido a que, para aplicaciones de visión artificial se requiere un sistema embebido que tenga capacidad de procesamiento de video, capacidad de procesamiento en una RAM externa y que permita almacenar gran cantidad de datos, es por estas características que se utiliza la tarjeta *Raspberry pi* que mediante las librerías de *OpenCV* se realizó el procesamiento de la información, para luego, poder ser transmitida hacia el usuario en forma de audio a través del parlante.

A continuación, tomando en cuenta las dimensiones de los elementos se procedió a desarrollar el diseño real considerando aspectos como:

- El espacio requerido para el montaje de los componentes electrónicos.
- Elementos complementarios que necesita la estructura, tales como una fuente constante de iluminación y una fuente de poder.
- Factores externos que podrían afectar al dispositivo como: la iluminación natural, temperatura y humedad.

Se realizó el diseño del diagrama circuital, fue necesaria la realización del esquema electrónico, programación del *software* y la simulación de todo el conjunto.

Luego, se desarrolló el código de programación en *Phyton*, utilizando librerías de *visión artificial* del *software* libre *OpenCV* de acuerdo a los requerimientos que va a tener el dispositivo, ya que este proyecto se basa en esta tecnología.

OpenCV es multiplataforma y proporciona un entorno de desarrollo fácil de utilizar y altamente eficiente, ya que su programación se basa en código *C y C++* optimizados [12].

Una vez diseñado el dispositivo, se procedió con la implementación del mismo, tomando consideraciones como: la altura de la cámara para conseguir un ángulo focal óptimo, la distribución de la fuente de iluminación alrededor de la cámara, el nivel adecuado de pótencia de salida de los parlantes, lugar de instalación del prototipo, la facilidad para su conexión y desconexión haciéndolo *plug and play*, para que al colocar la alimentación este entre en funcionamiento de manera inmediata.

Finalmente, ya implementado el dispositivo se realizó las pruebas de funcionamiento respectivas, en varias condiciones y con billetes de diferente denominación, con el fin de identificar posibles factores que pueden afectar el dispositivo.

3. RESULTADOS Y DISCUSIÓN

3.1. Requerimientos

Para tener una idea clara del diseño del prototipo que se deseaba implementar, fue necesario determinar las necesidades y los requerimientos asociadas a éste.

Para esto se entrevistó a la persona que acompaña a Walker Verdesoto, una persona no vidente y presidente ejecutivo de PROCODIS que atiende junto a él en el *Cyber* de la fundación.

Él explicó que generalmente los usuarios cancelan el servicio de *Cyber* con billetes de $1 (un dólar) y $ 5 (cinco dólares) principalmente.

Además, que el lugar donde se encuentra ubicada la ventanilla de atención al cliente tiene muy poca luz tanto natural como eléctrica.

En base a esto, se estableció los siguientes requerimientos para el prototipo:

- Resolución de la cámara.
- Identificación de billetes.
- Fuente constante de iluminación en la caja para mejorar las capacidades de captura e identificación.
- Nivel de audio de los parlantes.
- Ubicación del prototipo en un lugar estratégico y seguro.

Resolución de la cámara: debido a que el lugar donde se va a colocar el prototipo carece de buena iluminación tanto natural como eléctrica, además, como se pretende identificar un objeto en este caso los billetes, se consideró que la resolución de la cámara debía ser de 8 *Megapíxeles* aproximadamente, dado que, la resolución define el tamaño de la foto, y a mayor tamaño el sistema se hace mas lento [13].

Identificación de billetes: de acuerdo a los requerimientos se tomaron muestras de billetes tanto de un dólar, cinco dólares principalmente, como diez y veinte dólares secundariamente, lo que se pretende es que la identificación de cada uno de los billetes tenga un margen de error del 0.2 % como máximo para que el prototipo sea útil.

Fuente constante de iluminación: dado que, el lugar donde se va a colocar el prototipo carece de buena iluminación, pero es necesario colocarla ahí por tema de seguridad, por esta razón es esencial que el prototipo tenga un sistema de luz constante, por lo que se colocaron 6 *LEDS* distribuidos de manera uniforme alrededor de la cámara.

Ubicación estratégica del prototipo: debido a que el prototipo será utilizado para consultar la denominación de los billetes, por seguridad es necesario colocarlo en la

parte posterior de la persona que atiende el *Cyber*, esto es para evitar robos o que los usuarios puedan acceder a la caja y tomar los billetes de manera no autorizada.

3.2. Selección de Dispositivos

En base a los requerimientos del prototipo, los componentes electrónicos seleccionados fueron los siguientes:

Tarjeta *Raspberry pi*

Las características técnicas de la *Raspberry pi* se detallan en la Tabla 3.1.

Tabla 3. 1.Características técnicas de la *Raspberry pi* [5].

Procesador	Broadcom BCM2837, Cortex-A53 (ARMv8) 64-bit SoC
Frecuencia de reloj	1,2 GHz
GPU	Video Core IV 400 MHz
Memoria RAM	1GB LPDDR2 SDRAM
Capacidad de almacenamiento	Micro SD de 16 GB hasta 32 GB
Puertos	- GPIO 40 pines - HDMI - 4 puertos USB 2.0 - CSI (cámara Raspberry Pi) - DSI - Jack 3.5 para salida de audio
Alimentación	5 Voltios a 2.5 Amperios

Cámara Raspicam

A continuación, en la Tabla 3.2. se detallan las características técnicas de la *Raspicam*:

Tabla 3. 2.Características técnicas de la cámara *Raspicam* [9].

Resolución en pixeles	3280 x 2464
Alimentación	5 Voltios a 100 miliAmperios
Tamaño óptico del lente	¼ ¨
Imagen	Alta velocidad, alta sensibilidad y poca contaminación por ruido de patrón y borrones

Parlantes USB

Las características técnicas deben tener los parlantes se presentan en la Tabla 3.3.

Tabla 3. 3.Características técnicas de los parlantes *USB* [2].

Alimentación	5 voltios a 250 miliAmperios
Potencia de salida	1.5 watts
Compatibilidad	Windows, Mac o superior

3.3. Diseño Lógico

Partes del prototipo implementado

En la Figura 3.1, se muestra el diagrama de bloques de las conexiones del prototipo, como se detalla a continuación:

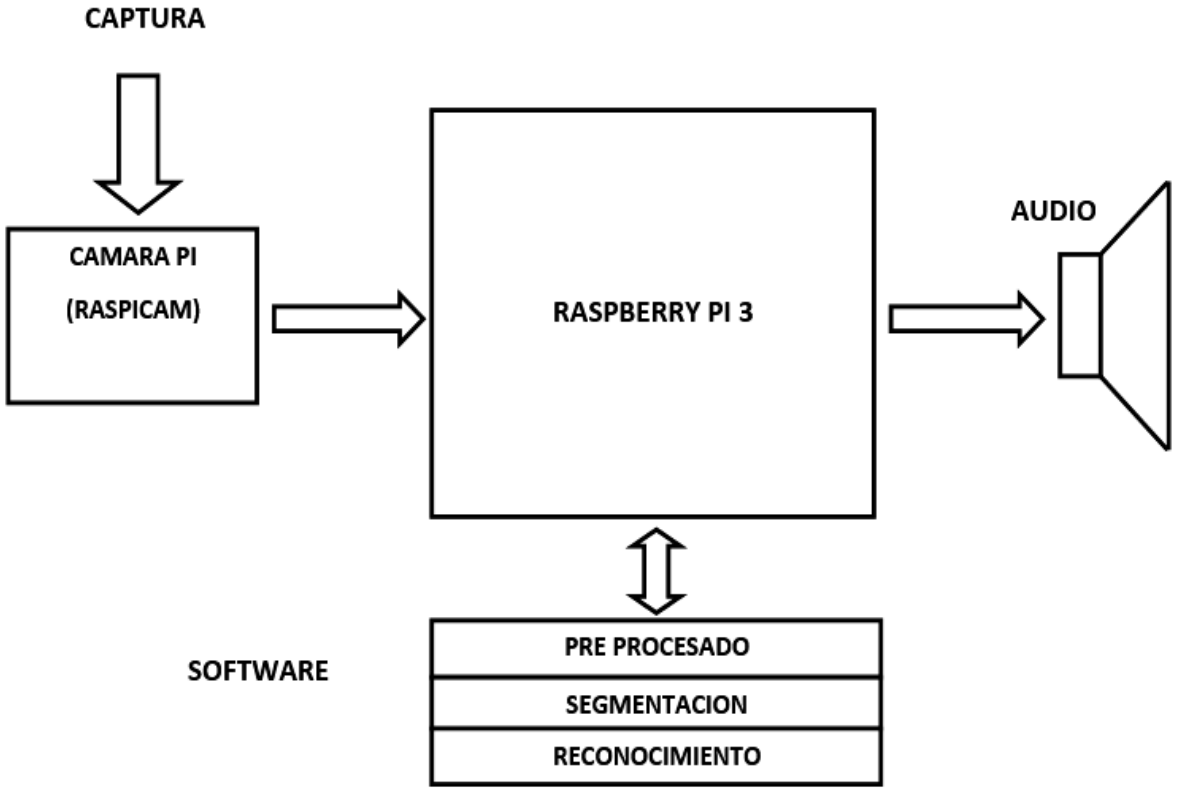

Figura 3. 1.Partes del prototipo implementado global.

Conexión de la cámara *Raspicam*

La cámara *Raspicam* se conectó directamente al conector *CSI* de la *Raspberry pi,* Figura 3.2, mediante el bus de datos, el cual está formado por varias líneas de 15 *cm* y se encuentra fijo a las ranuras de la cámara en su puerto serie, por este, circula tanto el voltaje de 5V que alimenta a la misma y los datos generados por esta hacia la tarjeta *Raspberry pi.*

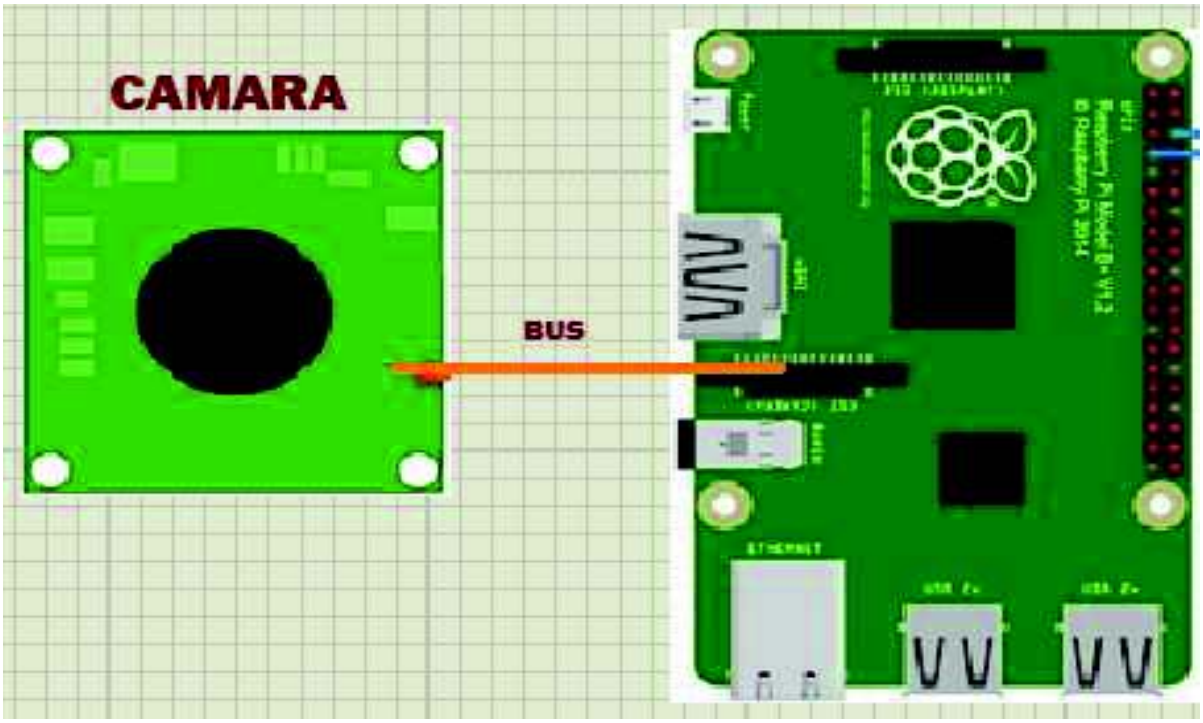

Figura 3. 2. Conexión de la cámara.

Conexión de los parlantes *USB*

La conexión de los parlantes se detalla a continuación:

Se insertó el plug macho del parlante en el *Jack* de audio de la *Raspberry pi*, a través de ello, se obtiene la salida de audio y para alimentar los parlantes se conectó el conector *USB* en el puerto *USB* de la *Rasberry pi*, como se observa en la Figura 3.3.

Conexión de la fuente constante de iluminación

Finalmente, los 6 *leds* se conectaron en paralelo, en los pines GPIO, la *Raspberry pi* cuenta con 40 pines, pero en este caso se utilizó el pin GPIO 6 como tierra (GND) y el pin GPIO 4 como positivo (Vcc).

La tarjeta Raspberry pi se alimenta mediante una fuente de corriente de 5V a 2.5 A, a la salida de los puertos GPIO hay 5V y la corriente estándar que consumen los diodos *led* azules es 25mA.

En base a esto, el valor de resistencia para protección de los *leds* sería: 5V/25mA=200 *ohmios* (*ohm*).

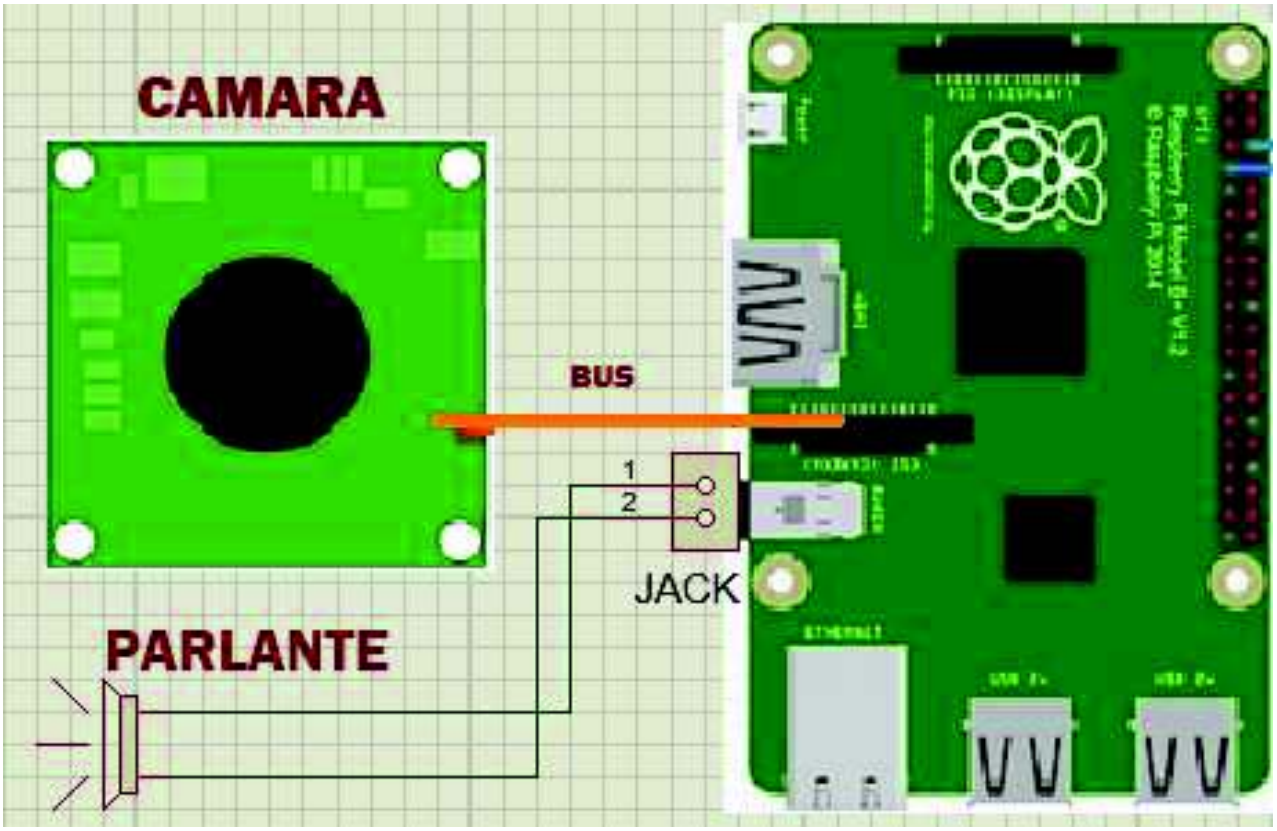

Figura 3. 3. Conexión de parlante.

Un valor de resistor físico real cercano a 200 *ohm* obtenido es: 220 *ohm*, Figura 3.4.

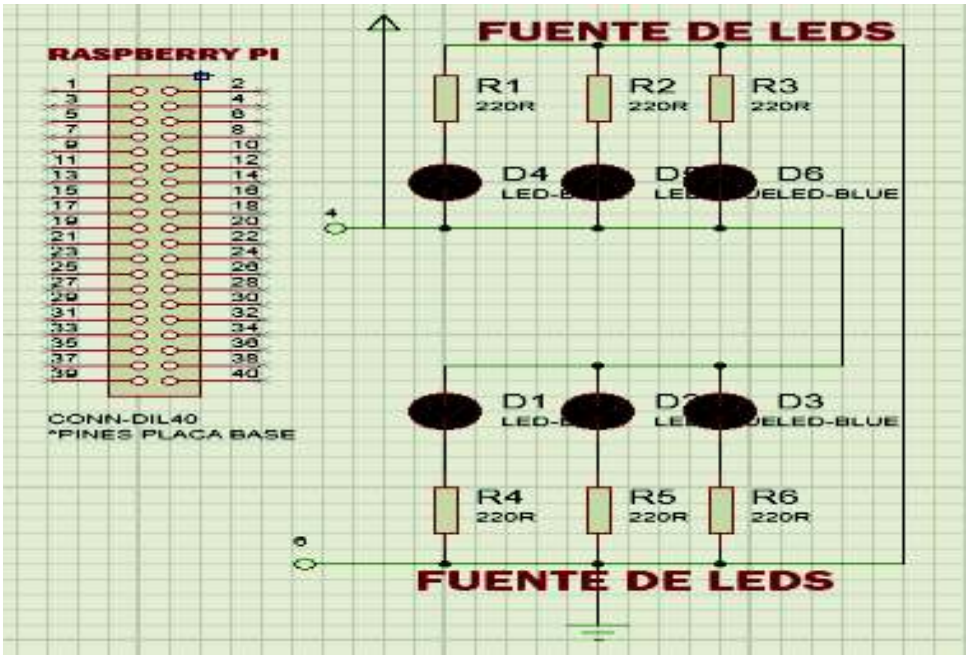

Figura 3. 4. Conexión de fuente de iluminación.

Interacción entre *hardware* y *software*

Para preparar la imagen se debe pre-procesarla, segmentarla, analizarla y guardarla, esto se logró gracias a las librerías que el *Raspberry pi* maneja, las cuales se detallan a continuación:

De manera inicial, fue necesario utilizar la librería "*cv2"*, esta librería conjuntamente con la memoria RAM procesan las imágenes digitales descomponiéndolas en matrices. Una vez obtenidas las matrices, se utilizaron las librerías "*imutils"* y "*numpy"* conjuntamente, para trabajar entre ellas y finalmente, se usó la librería "*matplotlib"* para realizar cálculos matemáticos requeridos para comparar las imágenes.

Todas las librerías antes mencionadas, son complementarias entre ellas para conseguir el procesamiento de imágenes en *visión artificial*.

A continuación, para conseguir la parte de la voz, se utilizó la librería *"os"* que contiene los codecs Mp3 para reproducir el audio.

Además, se utilizó la librería "*time",* que permite colocar retardos, para especificar los tiempos entre la toma de una muestra y otra.

Por otra parte, la librería "*glob"*, sirvió para direccionar a una ruta seleccionada y ahí buscar archivos o documentos donde se encuentran las imágenes y audios.

Estas librerías, tanto "*os"* y "*glob"*, se utilizan en conjunto, ya que el reproductor de la *Rasberry pi* reproducirá el audio que se encuentra indexado en una ruta especificada.

Las partes internas de la *Raspberry pi*, como: *microprocesador*, memoria *RAM*, disco duro (*micro SD*), son esenciales para trabajar en conjunto con las librerías y poder ejecutar las tareas, instrucciones lógicas y operaciones numéricas, Figura 3.5.

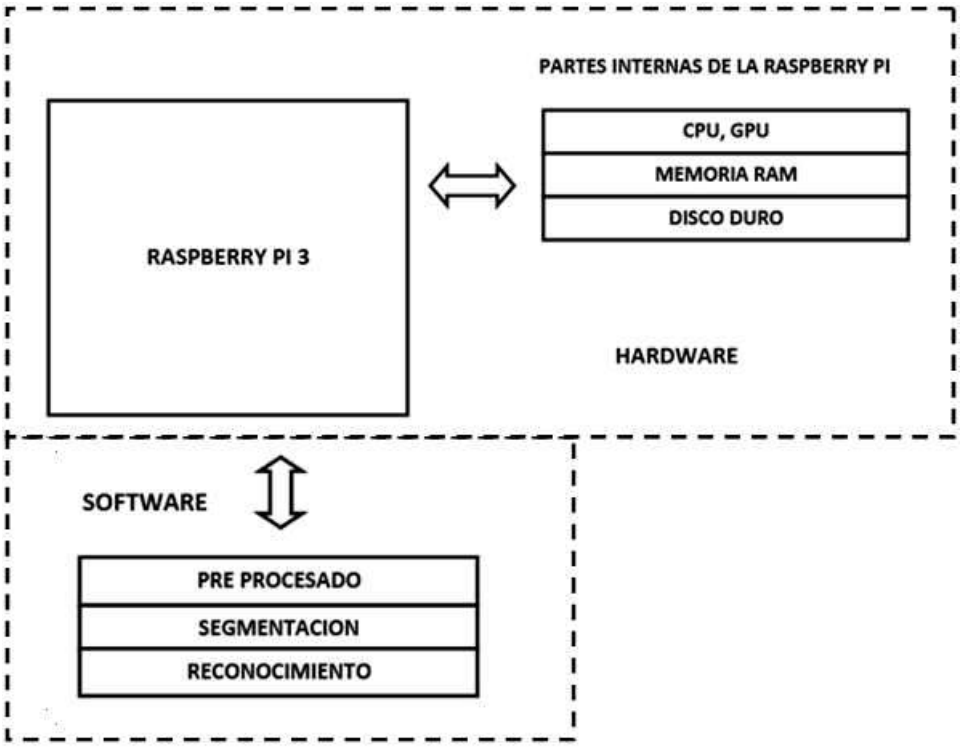

Figura 3. 5. Interacción entre *hardware* y *software*.

3.4. Diseño Físico

Diagrama final de conexiones del prototipo

En la Figura 3.6, se observa el digrama final de conexiones diseñado, y a continuación, se explica el funcionamiento técnico de todo el circuito.

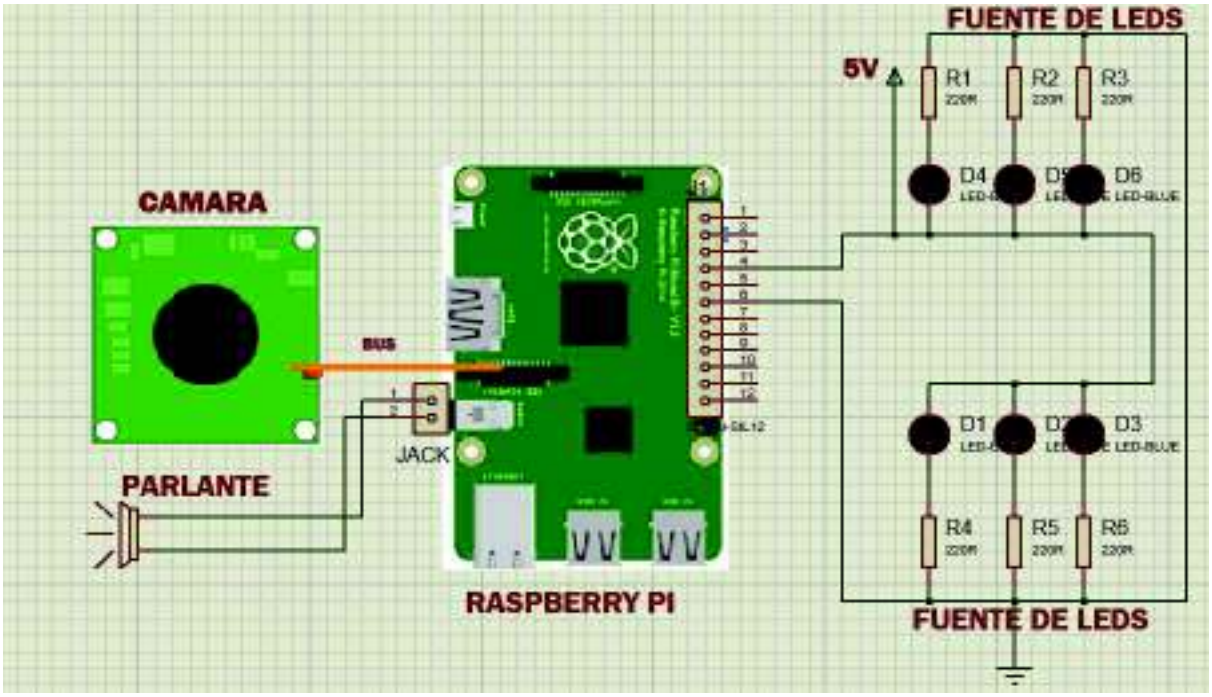

Figura 3. 6.Diagrama final de conexiones del prototipo

La información captada por la cámara se envía como ceros 0 y unos 1 hacia la *Raspberry pi*, a través del puerto serial *CSI*, para ello utiliza el bus de 12 hilos, de los cuales se dividen en buses de direcciones, buses de control y buses de datos.

Los buses de datos transportan en sí la información (ceros y unos) que va a ser procesada, la cual va a ser almacenada en forma de grupos de bits, estos se alojan de manera temporal en celdas que forman la memoria *RAM*, mientras se realiza el procesamiento.

Los buses de direcciones transportan señales hacia el microprocesador, el cual se encarga de ejecutar instrucciones programadas, entre ellas: realizar operaciones aritméticas como: suma, resta, multiplicación. Además, si necesita datos los solicita a la memoria *RAM.*

Para la captura de muestras; cuando se toma una imagen, se envía una instrucción al procesador de imágenes (*GPU)* desde el microprocesador para extraer toda la información que está en la memoria *RAM* a través de los buses de control, para: pre-procesarla, segmentarla y realizar operaciones matemáticas de las matrices.

Con la imagen preparada se la almacena en la *micro SD* para posteriormente utilizarla para la comparación con nuevas imágenes captadas.

Cuando se introduce el billete a ser consultado dentro del prototipo, la cámara captura la imagen y envía una señal a la *Raspberry*, esta acciona el *microprocesador* el cual gestiona los procesos y tareas en conjunto con la memoria *RAM* para tratar a la imagen, despúes, se compara esta imagen con las muestras almacenadas en la *micro SD*.

La imagen del billete a ser consultado no se almacena en la *micro SD*, ya que una vez que se obtenga la información de la denominación del billete, este proceso se repite para el resto de billetes a ser consultados, entonces, la información de ese billete solo se queda almacenada temporalmente en la memoria *RAM*.

Los LEDS, entrarán en funcionamiento inmediatamente cuando se conecte la fuente de corriente al prototipo, ya que estos servirán de iluminación extra para la cámara y de esta manera obtener una mejor visión del billete a ser analizado.

3.5. Software

Inicialmente, se deben configurar las librerías necesarias para trabajar con *visión artificial*, para conseguir el audio y definir tiempos.

Siguiente, en base a los requerimientos se realizán las configuraciónes respectivas de la cámara, en la cual se definen parámetros como: resolución y cantidad de imágenes tomadas por segundo.

Una vez que las muestras base han sido tomadas, procesadas y almacenadas en la *micro SD*, se procede, con la captura de la imagen del billete a ser consultado mediante la cámara.

Esta convierte la imagen analógica en imagen digital para luego enviarla a la *Raspberry pi* mediante el bus de datos, esta será la encargada de pre-procesar la imagen, es decir, prepara la imagen eliminando las zonas no útiles y realzando las zonas de interés, posteriormente, segmenta la imagen, es decir, realiza matrices de la imagen para su posterior interpretación.

A continuación, se realiza el reconocimiento de la imagen, analizando las diferentes características de las matrices en busca de patrones y caracteres que se asemejen a los patrones y caracteres de la plantilla de imágenes guardadas en la carpeta “imágenes” que se encuentra en la *micro SD*.

Una vez que se ha encontrado la similitud entre las características de sus matrices se informa al usuario la denominación del billete, presentando un mensaje de voz y para ello se utiliza los codecs de *Mp3*.

En la Figura 3.7, se observa el flujograma que indica el funcionamiento global del prototipo de identificación de denominación de billetes. Para cumplir con cada una de las etapas del proceso de funcionamiento del prototipo, se explicarán las instrucciones lógicas utilizadas en cada una de ellas.

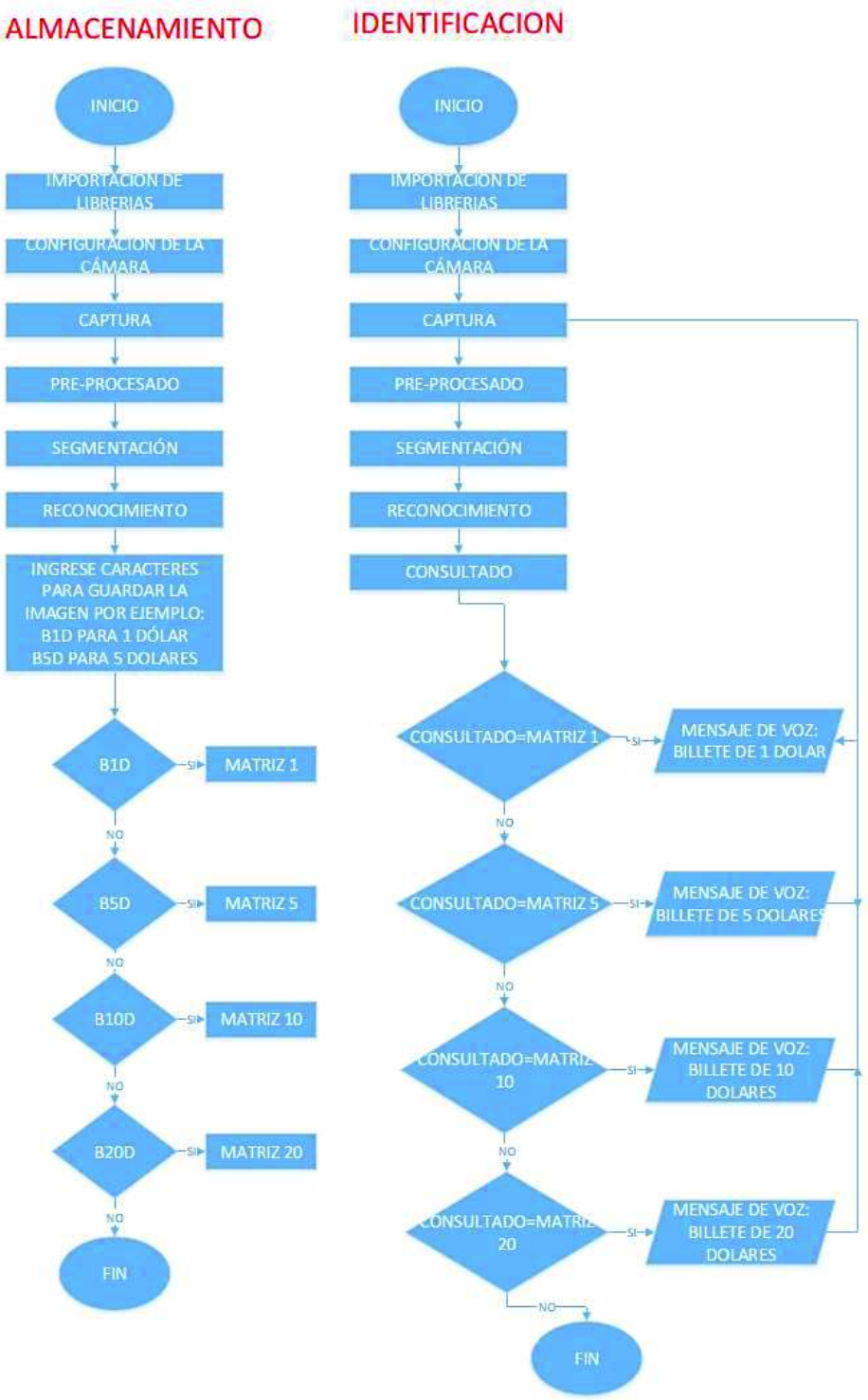

Figura 3. 7. Diagrama de flujo del funcionamiento global del prototipo

Utilización de las librerías de *OpenCV* en *Raspberry pi*

Antes de realizar el almacenamiento de muestras e identificación de la denominación, es esencial importar las siguientes librerías que se observan en la Figura 3.8, siendo necesarias para el correcto funcionamiento del programa, a continuación, se detallan para qué sirve cada una de ellas:

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import imutils
import cv2
import numpy as np
import os
import glob
from matplotlib import pyplot as plt
```

Figura 3. 8. Importación de librerías necesarias

Con las dos primeras líneas de programación se especifica que se va a utilizar la cámara propia de la Raspberry pi llamada “picamera”, a continuación, se importa la librería de tiempo, la cual se utiliza para colocar retardos.

Luego, se importa la librería que permite trabajar con matrices “arrays”, despues, la librería de visión artificial para el procesamiento de imágenes digitales, la librería que permite trabajar entre matrices y consiguiente la librería que permite realizar cálculos matemáticos dentro de las matrices.

Adicional, se importa la librería que permite indexar en la ruta seleccionada para examinar los archivos y documentos que contiene, además de, verificar si un archivo o carpeta existe en la ruta

Finalmente, se importa la librería que permite interactuar con el sistema operativo, es decir permite utilizar recursos del sistema operativo como es la parte de los codecs de audio para este proyecto.

Configuración inicial de la cámara

Como se observa en la Figura 3.9, en la primera línea de comandos ***camera = PiCamera()*** , con este comando se crea una variable, en la cual se almacenará todo lo que la cámara captura o esté observando en ese momento.

A continuación, con ***camera.resolution =(x, y),*** se define la resolución de la cámara, en este caso se colocó 432 x 240 *pixeles*.

Como se pretende enfocar una parte específica del billete, no es necesario utilizar la máxima resolución de la misma.

Por otra parte, entre más pequeña sea la resolución de la cámara, es mejor, debido a que los arrays también son más pequeños y de esta manera no tardará mucho tiempo el programa en analizar las matrices de esa imagen.

```
camera = PiCamera()
camera.resolution = (432, 240)
camera.framerate = 10
rawCapture = PiRGBArray(camera, size=(432, 240))

imagePaths = sorted(glob.glob("imagenes" + "/*.png"))

# allow the camera to warmup
time.sleep(0.1)
```

Figura 3. 9. Configuración e inicialización de la cámara

Ahora, ***Camera.framerate = 10*** con esta línea se define un “framerate” de 10, es decir la cantidad de imágenes que se analiza por minuto.

Luego, ***rawCapture = PIRGBArray (camera, size =(432, 240))*** se transforma la imagen que capturo, a escala de grises, esto es para reducir los arrays a pocos valores.

Finalmente, ***imagePaths = sorted(glob.glob("imágenes" + "/*.png))*** se declara un puntero "imagePaths" que direccione a todas las imágenes capturadas a la ruta de la carpeta "imágenes".

Con el comando ***sorted*** y ***global***, lo que se realiza es un índice, en el cual las muestras almacenadas serán ordenadas de manera alfabética, esto es clave al momento que el programa esté comparando coincidencias entre la matriz de el billete consultado "imagen de entrada" con la matriz de cada una de las imágenes guardadas inicialmente en la etapa: toma y almacenamiento de muestras.

Captura de imágenes y almacenamiento de muestras

Antes de que el prototipo opere, fue necesario tomar muestras de billetes base para su posterior comparación.

Una vez importadas las librerías necesarias y configurada e inicializada la cámara, el programa estará constantemente tomando muestras cada 10 segundos, se definió este tiempo debido a que la *Raspberry pi* solo posee una memoria *RAM* de 1GB, y un procesador de video cuya velocidad de procesamiento no es muy alta.

Entonces, se tomaron muestras a diferentes tiempos, con el objetivo de determinar el delay adecuado, de tal manera que el tiempo entre la toma de una muestra y otra, no sature la memoria *RAM* y de igual manera que el tiempo de respuesta no fuese demasiado alto, en base a las pruebas realizadas, con 10 segundos como *delay,* se obtuvieron los mejores resultados.

En esta fase, mediante ***TeclaPresionada = cv2.waitkey(1)*** se definió una variable llamada "Tecla presionada" la cual va a capturar una tecla, definida en código ASCII, es decir, espera unos milisegundos para cualquier evento de teclado, entonces, se creó un "lazo for" con ***if TeclaPresionada == 105:*** en el cual se analiza si la tecla presionada es la "i = 105 en código ASCII" se procede a realizar una captura de la imagen, la procesa y almacena temporalmente en la variable "image".

Posteriormente, con ***cv2.imshow ("IMAGEN QUE SERA GUARDADA, MiImagenGris")*** el programa muestra al usuario mediante una ventana la imagen resultante, entonces, el usuario decide, si la imagen generada es la que se desea guardar, se procede a ingresar una cadena de caracteres, es decir, el nombre de la imagen ***cadena 1 = raw_input(´ingrese el nombre del objeto seguido de un enter')*** estos caracteres se almacenarán en la "cadena 1".

A continuación, se crea un bucle anidado mediante ***TeclaPresionada = cv2.waitkey(3)*** donde la variable llamada "Tecla presionada" va a capturar una tecla definida en código ASCII, en este caso va a capturar la tecla "enter" entonces, si se ingresa caracteres, el programa ingresa a este lazo ***if cadena 1! = "":*** donde una vez ingresada la cadena de caracteres el programa vuelve a mostrar al usuario mediante una ventana la muestra generada junto con el nombre que se le acaba de asignar ***print 'LA IMAGEN SE GUARDO con el nombre:\n', cadena 1***.

Dicha cadena 1, tiene como dirección la ruta de la carpeta "imágenes /+ nombre de la cadena + .png" estas muestras se guardarán con el formato ".png"; caso contrario, el usuario presionó la tecla "enter" para salir y el programa imprimirá el mensaje LA IMAGEN NO SE GUARDO ***print 'LA IMAGEN NO SE GUARDO:\n'***.

Figura 3. 10. Captura de imagenes y almacenamiento de muestras para el funcionamiento del prototipo

Una vez con las muestras guardadas en la carpeta “imágenes”, utilizando la aplicación “*MobaXterm*” se conectó a la Raspberry pi mediante una “conexión cliente *ssh*”, luego, se arrastró las muestras de la carpeta “imágenes” al escritorio de la *laptop* y con la ayuda del programa editor de imágenes *Paint* se recortó las muestras reduciéndolas a las zonas de interés, una vez recortadas se arrastró estas muestras a la carpeta “imágenes” nuevamente, esta carpeta está ubicada en el escritorio de la *Raspberry pi*.

En la Figura 3.10, se presenta la línea de comandos empleados para la toma de muestras y almacenamiento de las mismas para el funcionamiento del prototipo.

Pre-procesado de imágenes

En la Figura 3.11, se presenta la linea de comandos utilizados para cumplir con esta etapa: primero, ***for frame in camera.capture_continuos(rawCapture, format=”bgr”, use_video_port=True):*** el programa analiza si es posible tomar *frames* desde esta cámara, en el caso de que si sea posible el programa ingresa aquí ***image = frame.array*** y crea de esa imagen tomada un ***Array***, es decir, una matriz de *pixeles* de esa imagen y la guarda en la variable llamada “image”, luego, a esta la transformará en escala de grises y la guarda en la variable “MiImagenGris”, después de guardarla, con el comando ***cv2.inshow(“Escala Grises”, MiImagenGris)*** el programa muestra la imagen guardada y transformada a escala de grises.

```
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    time.sleep(0.1)
    # grab the raw NumPy array representing the image, then initialize the timestamp
    # and occupied/unoccupied text
    image = frame.array

    # show the frame
    # cv2.imshow("Frame", image)
    #cambia a escala de grises
    MiImagenGris = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    cv2.imshow("Escala Grises", MiImagenGris)
```

Figura 3. 11. Pre-procesado de imagen

Segmentación de imágenes

En la línea de comandos de la Figura 3.12, se puede observar la utilización de: ***index= 0*** es decir, se declara una variable con un valor inicial de cero, luego, ***for (imagePath) in zip (imagePaths):*** se realiza la declaración de ***Arrays*** los cuales son grupos de variables almacenadas en una sola línea de código, a los cuales se puede acceder a través de índices y verificar si un archivo, en este caso una imagen existe o no en esa ruta indicada.

Con ***DirDestino = imagePaths[index]*** en esta variable se va almacenando los caracteres o ***Arrays*** de cada una de las imágenes "muestras almacenadas "que están dentro de este puntero "imagePaths" ya que la variable "index" va reduciendo su valor (-1) como se observa en la Figura 3.14 y por ende recorrerá todos los elementos que se encuentren dentro de éste.

Con ***NombreObjetoLeido= DirDestino.strip***, con el comando "***strip***" se analiza los caracteres de la matriz de inicio a fin, es decir, se divide una imagen digital en regiones homogéneas con respecto a ciertas características definidas por el usuario como es la longitud de los caracteres.

En este caso de 1 ***(1:len***) y el contorno de la matriz que será de ***"W, H"***, entonces, el programa va eliminando los caracteres que estén fuera de los parámetros définidos, así, centrándose en analizar automáticamente solo la zona de interés.

```
index = 0
for (imagePath) in zip(imagePaths):
  #print imagePath
  DirDestino= imagePaths[index]
  #- print DirDestino #Formato direccion leida => imagenes\Cargador.png
  NombreObjetoLeido = DirDestino.strip(".png")
  NombreObjetoLeido = NombreObjetoLeido.strip("imagenes")
  NombreObjetoLeido = NombreObjetoLeido[1:len(NombreObjetoLeido)]
```

Figura 3. 12. Segmentación de imagen

Límites de la matriz

En la Figura 3.13, una vez con la matriz de la imagen de muestra de tamaño **(WXH)** lista y almacenada en “NombreObjetoLeido”, ahora, se procede a leer la imagen del billete a ser consultado que es el que está capturando en ese momento la cámara, esto con el comando ***cv2.imread*** y lo almacenará temporalmente en la variable “ImagenLeidaFolder”.A esta imagen la procesa, es decir, la divide en regiones homogéneas con respecto a ciertas características definidas por el usuario como es la longitud de los caracteres en este caso de 1 ***(1:len***) y el contorno de la matriz que será de ***“w, h”.***

Reconocimiento de la denominación de billetes

Con***resBBtt=cv2.matchTemplate(MiImagenGris,ImagenLeidaFolder,cv2.TM_CCOEFF_NOMRMED)*,** como ambas imágenes ahora tienen matrices de ***Arrays*** del mismo tamaño gracias a los parámetros definidos por el usuario, se realizá operaciones básicas entre pixeles en la cual se multiplican las matrices de ambas imágenes**(WxH),(wxh)** y como resultado se obtiene la resta **(W-w+1, H-h+1)** de matrices.

El resultado se almacenará en la variable **resBBtt**, adicional, en la línea de comandos mediante ***treshold=0.8*** se define una precisión de 0.8, Figura 3.13.

```
ImagenLeidaFolder = cv2.imread(DirDestino,0)#lee la imagen
w, h = ImagenLeidaFolder.shape[::-1]
resBBtt = cv2.matchTemplate(MiImagenGris,ImagenLeidaFolder,cv2.TM_CCOEFF_NORMED)
threshold = 0.8
```

Figura 3. 13. Operación matemática entre matrices

Condicional para la toma de decisión

Finalmente, con ***locBBtt= np.where (resBBtt >= treshold)*** el programa compara, si el resultado de la resta de matrices es mayor o igual a **0.8**, el valor de la precisión,

entonces, ingresa al ***lazo for***, donde va a analizar imagen por imagen en la carpeta de imágenes base, en busca de coincidencias entre patrones de las matrices.

```
locBBtt = np.where( resBBtt >= threshold)

if zip(*locBBtt[::-1]):
```

Figura 3. 14. Condicional para toma de decisión

Reconocimiento billete de un dólar

Una vez que se detectó la matriz que se asemeja o tiene coincidencias con el valor de 0.8, el programa ingresa a la subrutina correspondiente para identificar e indicar al usuario la denominación del billete consultado mediante un mensaje de voz.

Para ello, éste analiza con cúal de las muestras guardadas es el billete consultado igual, luego, se declara la ruta del audio *Mp3* perteneciente a dicho billete y se utiliza el reproductor “omxplayer” propio de *Raspbian* para reproducirlo, también, se imprime el valor del billete a través del *hyperterminal* de *Python*, Figura 3.15.

```
if NombreObjetoLeido =="b1d4":
  print 'Billete de 04 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b1d.mp3")
  break

if NombreObjetoLeido =="b1d5":
  print 'Billete de 05 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b1d.mp3")
  break

if NombreObjetoLeido =="b1d6":
  print 'Billete de 06 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b1d.mp3")
  break

if NombreObjetoLeido =="b1d7":
  print 'Billete de 07 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b1d.mp3")
  break

if NombreObjetoLeido =="b1d8":
  print 'Billete de 08 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b1d.mp3")
  break
```

Figura 3. 15. Subrutinas para billete de un dólar

Para el resto de billetes a ser consultados, se realizaron las mismas subrutinas, pero se cambia el valor (nombre de la respectiva muestra) y la ruta del audio *Mp3* perteneciente a cada uno de los billetes, estas subrutinas se adjuntan en el Anexo A.

En la Figura 3.19, ya sea que se haya identificado o no un billete, con la línea de código ***cv2.destroyAllWindows ()*** se destruyen todas las ventanas que estén abiertas y de esta manera el programa está listo para realizar la consulta de un nuevo billete.

```
# clear the stream in preparation for the next frame
rawCapture.truncate(0)

# if the 'q' key was pressed, break from the loop
if cv2.waitKey(1) == 27:
    break

cv2.destroyAllWindows()
```

Figura 3. 16. Limpieza de registro y salir del programa

Plug and Play

Uno de los requerimientos del prototipo fue que debe ser *plug and play*, es decir, que al conectarlo a la alimentación éste entraría en funcionamiento de inmediato, para ello se debe seguir los siguientes pasos:

En la Figura 3.17, se tiene en el escritorio la carpeta llamada “VisionBilletes” y adentro de esta el archivo con nombre “*visión.py*”, se utiliza el comando ***ls –l*** para ver qué archivos se encuentran en esta carpeta.

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Oct 23 20:29:25 2018
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ cd VisionBilletes
pi@raspberrypi:~/Desktop/VisionBilletes $ ls -1
imagenes
imutils.py
imutils.pyc
mp3
vision.py
pi@raspberrypi:~/Desktop/VisionBilletes $
```

Figura 3. 17.Hyperterminal de Raspbian

Desde el hyperterminal de Raspbian, Figura 3.17, se direccionó hasta la carpeta que contiene el fichero con el comando ***cd*** *(change directory)***.**

Creación de *Script*

Una vez que se ha cambiado de directorio a la carpeta *VisionBilletes*, se creó un archivo con extension ***.sh*** con nombre ***apli_vision***, es decir, se creó un *script*, para despúes ejecutarlo automáticamente, Figura 3.18.

Figura 3. 18. Creación de archivo *apli_vision.sh*

Luego, se escribe la ruta que es ***bin/ sh***, se colocó el nombre del archivo ***MiArchivo.sh***, se guarda y se retorna al directorio raíz (*root*) ***cd /***. Figura 3.19.

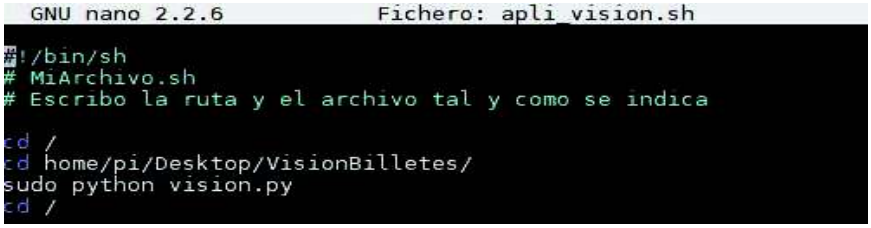

Figura 3. 19. Almacenamiento de archivo en carpeta VisionBilletes

Asignación de permisos en ficheros, en *Linux*

Posterior, se asignó permisos al archivo con el comando ***sudo chmod 755 apli_vision.sh,*** a continuacion, en la Figura 3.20, se explica como se está asignando permisos en formato octal y se interpreta de la siguiente manera:

Figura 3. 20. Permisos de un fichero en formato octal

7: el usuario tendrá todos los permisos de escritura, lectura y ejecución en ese fichero.

5: el grupo tendrá solo permisos de lectura y ejecución, pero no de escritura.

5: otros usuarios al igual que grupo solo tiene permisos de lectura y ejecución.

Una vez que se haya entendido cómo se asignan permisos en los ficheros, se procedió a la asignación de permisos en el código de programación, Figura 3.21.

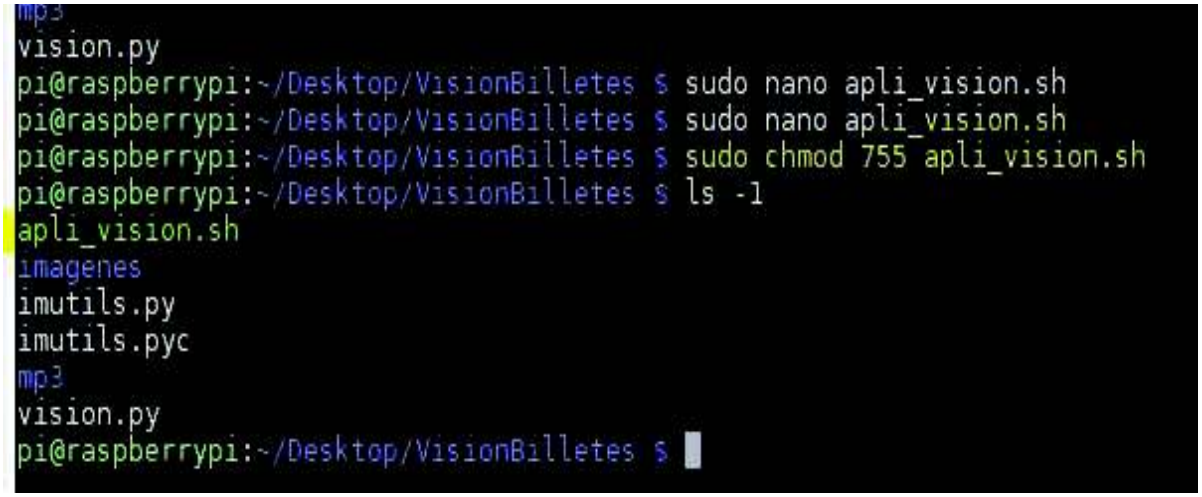

Figura 3. 21. Asignación de permisos en el código de programación

Se verificó el funcionamiento, para esto se debe dirigir hasta el directorio principal y ejecutar el archivo ***.sh*** creado anteriormente: ***sh apli_vision.sh.***

Luego, se regresa al directorio raíz ingresando dos veces el comando ***cd "cambiar de directorio"*** y se creó un directorio con nombre "***logins***" con el comando ***mkdir,*** quedando como se muestra en la Figura 3.22.

Finalmente, se ejecutó la línea **sudo crontab –e** y se adicionó al final lo siguiente:

*@rebootsh/home/pi/Desktop/**VisionBilletes/apli_vision.sh**>/home/pi/**logins/cronlog** 2>&1*

Esta línea de código se puede interpretar que al reiniciar el equipo y volver a iniciar, éste se dirigirá automáticamente a la carpeta VisiónBilletes y ejecutará el ***script*** llamado **apli_vision.sh** en donde está contenido el archivo **vision.py**.

Figura 3. 22. Creación de directorio *logins*

3.6. Implementación

Montaje de componentes

Luego de probar el funcionamiento de cada uno de los componentes electrónicos conectados con la *Raspberry pi*, se procedió a realizar una caja de madera de 28 *cm* de alto x 38 *cm* de ancho y dentro de esta caja se colocaron los componentes según las características determinadas en la metodología:

Ubicación de la Cámara

La cámara se colocó de manera fija en la caja, mediante 4 tornillos empotrados a la parte superior de la caja y a una altura de 23 *cm* tomando como referencia la superficie donde se coloca el billete, como se muestra en la Figura 3.23, dado que, esto ayuda

al ángulo focal de la cámara al momento de capturar las zonas de interés en este caso, zonas de los billetes.

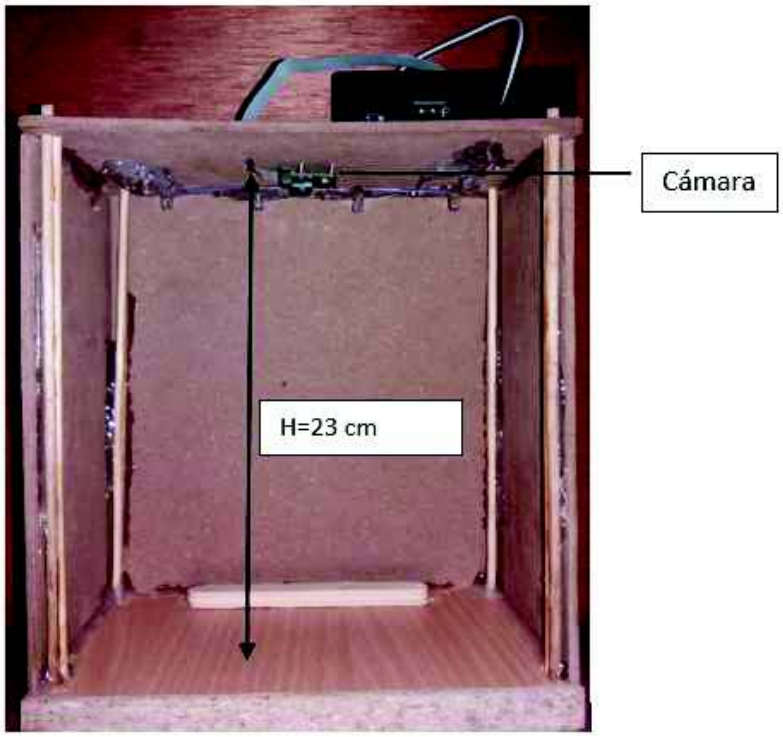

Figura 3. 23. Ubicación de cámara

Ubicación de fuente de iluminación

En la Figura 3.24, se puede observar la fuente constante de iluminación, es decir, los seis (6) *leds* de color azul se ubicaron alrededor de la cámara y se conectaron en paralelo.

Cada uno de estos con una resistencia de 220 *ohmios* para limitar la corriente que circula por los leds, estos ayudan como iluminación extra al momento que la cámara capture, enfoque las imágenes de los billetes.

Los conectores que alimentan a los leds salen por el destaje que se observa en la Figura 3.25 y se conectan al puerto *GPIO* 6 como tierra y al puerto *GPIO* 4 como

positivo (5V), de esta manera se energiza al prototipo y se encenderán desde el instante que la *Raspberry pi* se conecte a la alimentación.

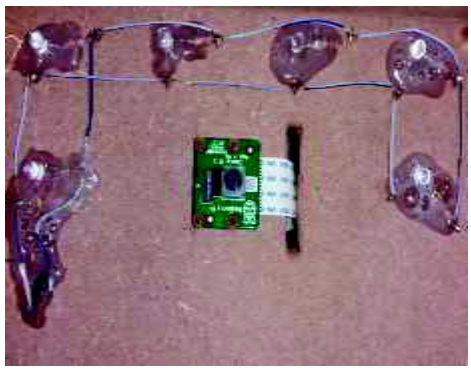

Figura 3. 24. Ubicación de la fuente de iluminación alrededor de la cámara

Ubicación de la *Raspberry pi*

Como se observa en la Figura 3.25, sobre la parte superior de la caja se colocó la *Raspberry pi* unida a la caja mediante cinta doble faz, luego, en la parte superior de la caja se realizó un destaje por donde sale el bus de datos de la cámara para su conexión a la *Raspberry pi*.

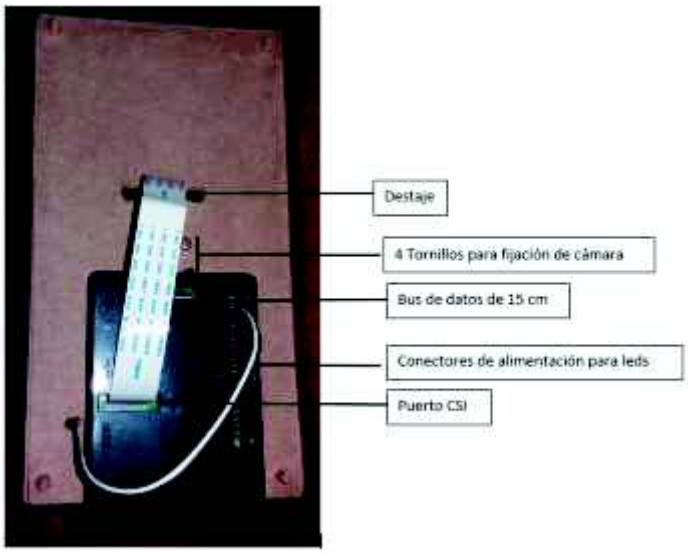

Figura 3. 25. Parte posterior de la caja

Ubicación de los parlantes *USB*

Los parlantes se distribuyeron a los costados de la cámara, como se observa en la Figura 3.26.

Luego, se conectaron a la *Raspberry pi* mediante el plug a la salida de audio de la misma y el cable de alimentación en los puertos *USB* de la *Raspberry pi*.

Adicional, se organizó el cableado de los mismos de manera que no dañase la presentación del prototipo.

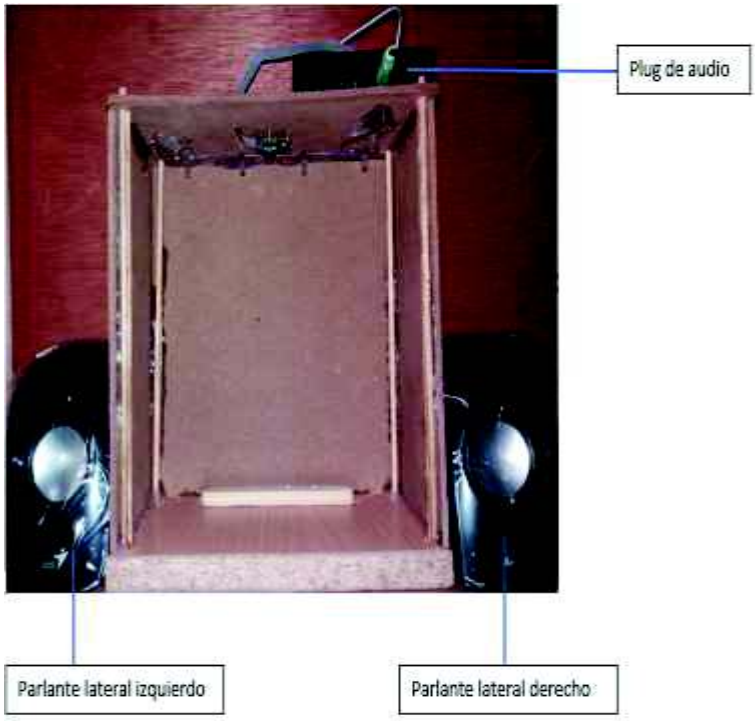

Figura 3. 26. Conexión de parlantes

Prototipo final

Al final de colocar todos los componentes y conectarlos a la *Raspberry pi*, se procede a ensamblar la caja y cerrarla por completo, dejando unicamente el espacio para que ingrese una mano y pueda colocar el billete a ser consultado sobre la superficie, una

vez realizado aquello, se procede a conectar el cargador de la *Raspberry* a la fuente de corriente para que entre en funcionamiento y se obtuvo como resultado el prototipo de la Figura 3.27.

Figura 3. 27.Prototipo de identificación de billetes terminado

3.7. Pruebas de funcionamiento

Finalmente, una vez que se ha implementado el dispositivo se realizarón las pruebas de funcionamiento respectivas:

- Pruebas con billetes de 1 dólar, 5 dólares, 10 dólares y 20 dólares, de cada uno de ellos al menos se realizo 2 ejercicios de identificación, el primer ejercicio consiste en colocar un billete en diferentes orientaciones, anverso y reverso.

El segundo ejercicio consiste en colocar billetes de diferente denominación a los mencionados anteriormente, con el fin de identificar posibles factores que pueden afectar el dispositivo.

El resto de resultados obtenidos de las pruebas se adjuntan en el Anexo B.

Pruebas con billete de 1 dólar

Pruebas no exitosas

Con estas muestras almacenadas, Figura 3.28, al introducir un billete a ser consultado dentro del prototipo, este tardaba más de 10 segundos aproximadamente en informar la denominación del billete en cuestión y más del 80 por ciento de las pruebas fallaron, ya que, el prototipo no lograba reconocer, tampoco informar la denominación del mismo, Figura 3.29.

Sin embargo, estas muestras no fueron descartadas y también se almacenaron en la *micro SD,* dado que, mientras más muestras, el prototipo puede encontrar más coincidencias al momento de comparar.

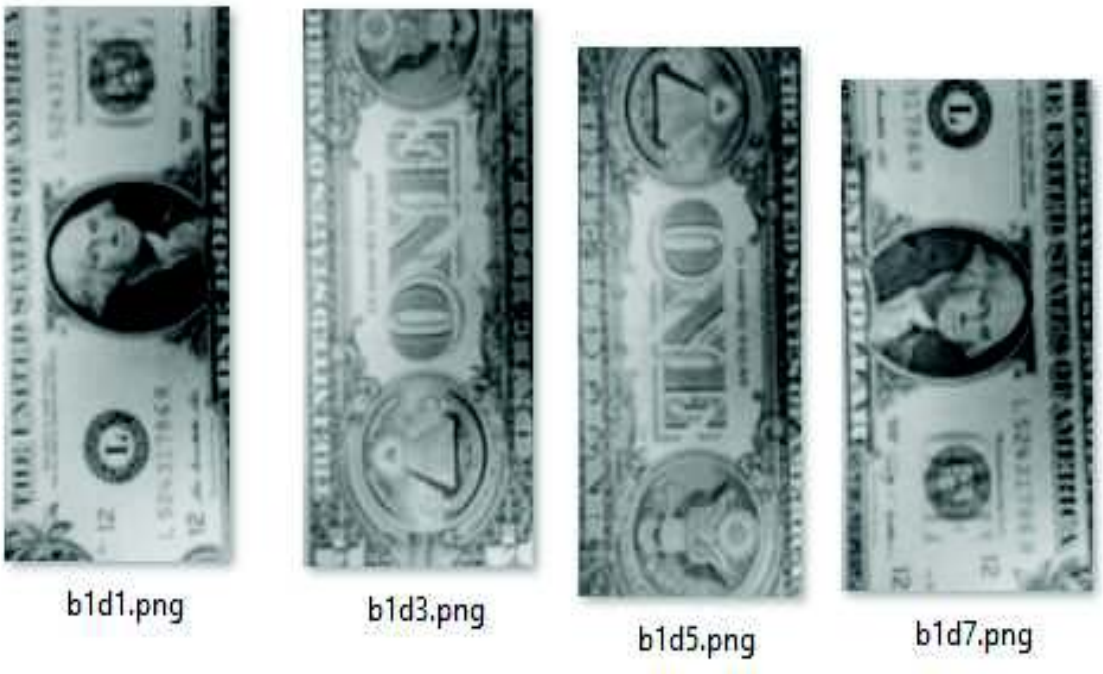

Figura 3. 28. Muestras no exitosas

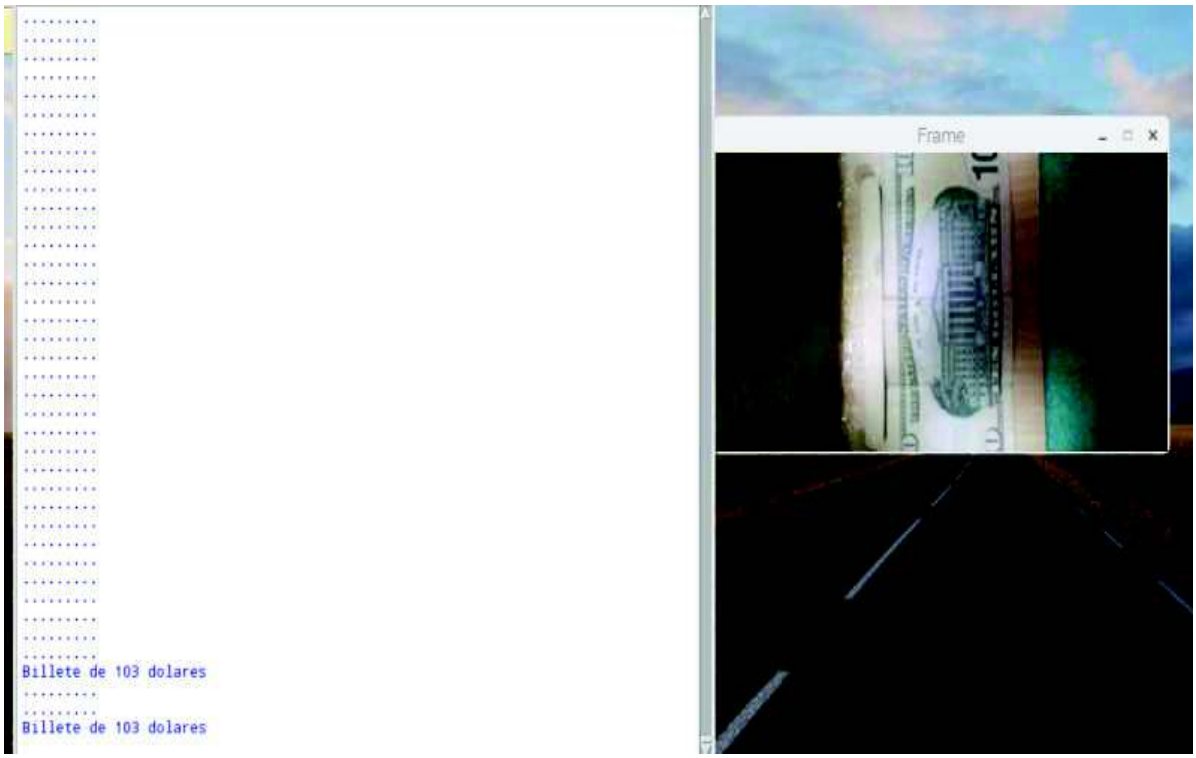

Figura 3. 29. Tiempo de reconocimiento aproximadamente 10 segundos y funcionamiento aleatorio

Pruebas exitosas

En base a las pruebas no exitosas, se analizó las posible soluciones, donde se probó: enfocando partes más específicas del billete, dado que al enfocarse en una parte precisa, se obtendría la suficiente información del billete para su respectiva comparación e identificación.

Entonces, se almacenaron las muestras Figura 3.30, obteniendo resultados satisfactorios como: que el tiempo de identificación en comparación a las pruebas no exitosas no sobrepasa los 5 segundos aproximadamente, Figura 3.31, y al colocar billetes en diferentes condiciones el prototipo logró identificar e informar la denominación, razón por la cual lo hizo confiable.

Figura 3. 30. Muestras exitosas

Otra solución que se dio fue colocar un tope en el lugar donde se va a colocar el billete, Figura 3.31, ya que de esta manera se asegura que el billete estará colocado en cualquiera de sus caras, pero siempre en la misma posición, lo cual ayudaría aún más a la cámara al momento de capturar y analizar patrones referenciales.

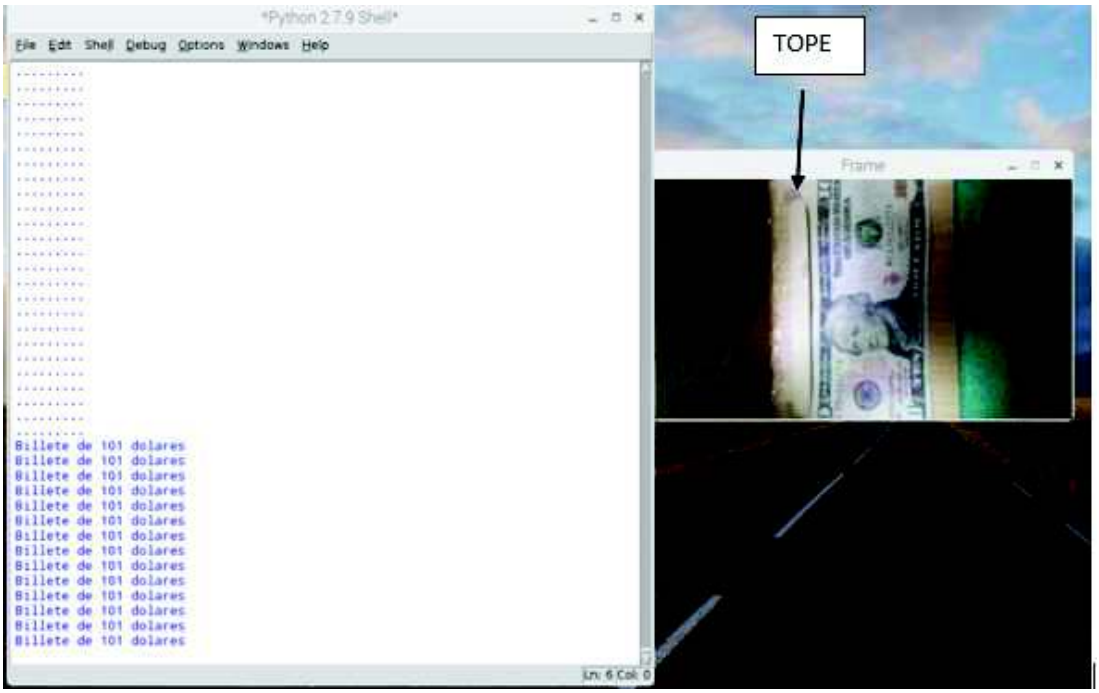

Figura 3. 31. Tiempo de identificación aproximadamente de 5 segundos

Pruebas con billetes de diferente denominación a las muestras almacenadas

Finalmente, se realizaron pruebas con billetes de diferente denominación a las mencionadas anteriormente y con billetes de otros países, con el objeto de verificar que el prototipo sea confiable, no identificando billetes diferentes a las muestras base, Figura 3.32. El resto de muestras se adjuntan en el anexo C.

Como se observa en la Figura 3.32, al colocar dentro del prototipo billetes de diferente denominación a las muestras base e inclusive al colocar billetes que no pertenecen al país, pero que sean del mismo valor, el programa no los reconoce, haciéndolo sumamente confiable, ya que caso contrario los hubiese identificado como un falso positivo, pero no fue así.

Figura 3. 32.Pruebas con billetes de diferente denominación y diferente país

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

Luego de realizar una breve descripción de las técnicas utilizadas en *visión artificial* para el procesamiento de imágenes digitales, se eligió la técnica de comprobar el histograma de la imagen, porque esta técnica divide el numero de *pixeles* de la imagen y analiza las zonas de interés, es decir, ira descartando los pixeles q no se asemejen a la imagen almacenada, hasta conseguir una coincidencia, esta fue la que mejor se acopló al funcionamiento del proyecto.

Para el diseño físico y técnico del prototipo se tuvo que visitar la fundación para determinar la necesidad a resolver, además, analizar el lugar donde iba a ser colocado el prototipo, las posibles dimensiones que tendría y observar los posibles factores que pudieran afectarlo y de esta manera se procedió a investigar los elementos necesarios para su implementación.

El dispositivo identificador de billetes utiliza una cámara, con un *software* que permite dar facilidades a través del uso de librerías y de esta manera reducir las líneas de la programación para el procesamiento de imágenes digitales, y, además, reducir los errores comunes en la programación cuando se trabaja con *visión artificial*.

Para identificar objetos mediante *visión artificial*, se necesita crear un ambiente controlado entre la cámara de la *Raspberry pi*, la distancia de la misma hacia el punto donde se va a colocar el objeto, el tope y la fuente de iluminación constante para obtener un mejor enfoque del objeto y una captura de imagen clara.

Este proyecto se basa en buscar coincidencias entre un objeto capturado en comparación a una base de imágenes guardadas, para ello se ayuda de operaciones básicas con matrices, pre-procesado de imagen y segmentación de las mismas, por lo que es importante la utilización de librerías como:

cv2 para procesamiento de imágenes, *numpy* para trabajar entre matrices y *matplotlib* para realizar operaciones numéricas entre ellas.

Python es un lenguaje de programación multiplataforma que nos permite manejar las librerías de Vision Artificial, con lo cual se minimiza el tiempo de desarrollo de una aplicación.

El prototipo se basa en que el usuario ingresa el billete a ser consultado dentro de la caja, la *Raspberry* realiza las tareas necesarias para identifcar la denominación, este proceso de identificación de la denominación tiene un tiempo de respuesta de 5 segundos aproximadamente, este tiempo puede variar dependiendo del aspecto físico del billete a ser consultado.

Debido a que la ceguera afecta la capacidad de las personas de realizar varios trabajos con normalidad, el dispositivo es *plug and play*, es decir, que ellos al conectar el prototipo a la toma corriente, éste entrará a funcionar de inmediato. Una vez instalado el equipo, se colocó una toma corriente con una muesca de manera que estas personas puedan guiarse para enchufar el equipo, adicional, mediante la ranura y el tope que posee el prototipo para colocar los billetes, podrán verificar que el prototipo esta colocado de manera correcta y no este de cabeza.

Aplicando el conocimiento adquirido durante la carrera Universitaria y en conjunto con el trabajo de investigación se logro desarrollar un dispositivo que identifique la denominación de billetes mediante *visión artificial*, este dispositivo permite a los discapacitados visuales de la fundación determinar la denominación de los billetes percibidos al momento de cobrar el servicio de Cyber, mejorando la calidad de vida de estas personas, la identificación ya no es un problema para estas personas e inclusive han mejorado los tiempos de atención a los clientes.

4.2. Recomendaciones

En caso de que se pretenda realizar modificaciones al presente proyecto para trabajos futuros, se recomienda que además de la identificación de la denominación del billete, éste identifique la veracidad del mismo.

Para obtener un prototipo con tiempos de respuestas mucho más rápido que el actual proyecto, se sugiere investigar acerca de tarjetas que tengan mejores prestaciones en memoria *RAM* y procesador gráfico en comparación a la tarjeta *Raspberry pi*, para de esta manera alcanzar un procesamiento mas rápido de imágenes y por ende obtener un prototipo con tiempos de respuesta inmediatos.

Para que el prototipo informe el valor del billete consultado en un tiempo no mayor a los 5 segundos se sugiere colocar los billetes a ser consultados lo mas planos, planchados posibles y bien pegados al tope de la superficie.

Para el caso de billetes arrugados, en donde el prototipo supere los 5 segundos de respuesta y no logre informarnos la denominación, se sugiere presionar con la yema de los dedos el billete hasta obtener una posición casi plana del mismo y la cámara logre identificar alguna coincidencia en base a las muestras almacenadas.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] EP, «20 MINUTOS EDITORA, S.L.,» 28 10 2010. [En línea]. Available: https://www.20minutos.es/noticia/856381/0/ciegos/tacto/desarrollo/.

[2] AXA, «Axa Computacion,» 13 11 2018. [En línea]. Available: https://axa.com.ar/webaxa/parlantes-20-21-51-usb/1001-parlante-genius-sp-u115-bk-usb.html.

[3] U. P. d. Valencia, «Historia de la informatica,» 18 12 2013. [En línea]. Available: https://histinf.blogs.upv.es/2013/12/18/raspberry-pi/. [Último acceso: 22 12 2018].

[4] M. Á. Abellán, «PROGRAMO ERGO SUM,» 16 8 2017. [En línea]. Available: https://www.programoergosum.com/cursos-online/raspberry-pi/232-curso-de-introduccion-a-raspberry-pi/instalar-raspbian.

[5] J. PASTOR, «Xataka,» 25 4 2018. [En línea]. Available: https://www.xataka.com/ordenadores/raspberry-pi-3-model-b-analisis-mas-potencia-y-mejor-wifi-para-un-minipc-que-sigue-asombrando.

[6] R. p. 3, «Prototipo pi,» 2 5 2016. [En línea]. Available: https://prototipopi.wordpress.com/author/prototipopi/. [Último acceso: 1 1 2019].

[7] M. A. Albellan, «PROGRAMO ERGO SUM,» 17 8 2017. [En línea]. Available: https://www.programoergosum.com/cursos-online/raspberry-pi/232-curso-de-introduccion-a-raspberry-pi/instalar-raspbian. [Último acceso: 22 12 2018].

[8] EcuRed, «OpenCV,» 24 2 2016. [En línea]. Available: https://www.ecured.cu/OpenCV. [Último acceso: 22 12 2018].

[9] RS, «RS Components Ltd,» 23 12 2017. [En línea]. Available: https://es.rs-online.com/web/p/modulos-de-video/9132664/.

[10] D. S. Montemayor, «Repositorio Universidad de Oviedo,» 12 2 2015. [En línea]. Available: http://digibuo.uniovi.es/dspace/bitstream/10651/30550/8/TFMDSMMemoriaRuo.pdf. [Último acceso: 23 12 2018].

[11] J. F. V. Serrano, «VISION POR COMPUTADOR,» 18 9 2016. [En línea]. Available: http://www.visionporcomputador.es/libroVision/libro.html.

[12] A. R. Bazaga, «Oficina de Software Libre,» 18 8 2015. [En línea]. Available: https://osl.ull.es/software-libre/opencv-libreria-vision-computador/.

[13] J. A. Luna, «hipertextual,» 15 4 2017. [En línea]. Available: https://hipertextual.com/2015/04/megapixeles.

[14] I. Taboola, «LINUXADICTOS,» 16 11 2017. [En línea]. Available: https://www.linuxadictos.com/conseguir-ver-los-permisos-fichero-formato-octal.html.

[15] S. Velasco, «Aula SUN UCM,» 5 5 2008. [En línea]. Available: https://webs.ucm.es/info/aulasun/archivos/SCRIPTS.pdf.

6. ANEXOS

ANEXO A: SUBRUTINAS DE LOS BILLETES

Una vez que se detectó con qué matriz se asemeja o tiene coincidencias el billete consultado, a continuación, el programa ingresa a la subrutina correspondiente para indicar al usuario la denominación del billete consultado mediante un mensaje de voz.

Para ello se declara la ruta del audio *Mp3* perteneciente a dicho billete y se utiliza el reproductor "*omxplayer*" propio de *Raspbian* para reproducirlo, también, se imprime el valor del billete a través del *hyperterminal* de *Python,* Figura 6.1 y Figura 6. 2.

```
if NombreObjetoLeido =="b5d3":
  print 'Billete de 53 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b5d.mp3")
  break

if NombreObjetoLeido =="b5d4":
  print 'Billete de 54 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b5d.mp3")
  break

if NombreObjetoLeido =="b5d5":
  print 'Billete de 55 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b5d.mp3")
  break

if NombreObjetoLeido =="b5d6":
  print 'Billete de 56 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b5d.mp3")
  break

if NombreObjetoLeido =="b5d7":
  print 'Billete de 57 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b5d.mp3")
  break
```

Figura 6. 1.Subrutinas billete de cinco dólares

```
if NombreObjetoLeido =="b10d2":
  print 'Billete de 102 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b10d.mp3")
  break

if NombreObjetoLeido =="b10d3":
  print 'Billete de 103 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b10d.mp3")
  break

if NombreObjetoLeido =="b10d4":
  print 'Billete de 104 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b10d.mp3")
  break

if NombreObjetoLeido =="b10d5":
  print 'Billete de 105 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b10d.mp3")
  break

if NombreObjetoLeido =="b10d6":
  print 'Billete de 106 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b10d.mp3")
  break

if NombreObjetoLeido =="b10d7":
  print 'Billete de 107 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b10d.mp3")
  break

if NombreObjetoLeido =="b10d8":
  print 'Billete de 108 dolares'
  os.system("omxplayer -b /home/pi/Desktop/VisionBilletes/mp3/b10d.mp3")
  break
```

Figura 6. 2.Subrutinas billete de diez dólares

ANEXO B: TOMA DE MUESTRAS

En este caso fue necesario tomar 2 muestras por cada uno de los lados del billete, también en sentido inverso, obteniendo en total 8 muestras por cada uno de los billetes, estas muestras tanto exitosas y no exitosas fueron almacenadas, las cuales se muestran a continuación:

Por cada lado se toman dos muestras, Figura 6.3, una enfocada a una parte específica del billete y la otra muestra enfocando casi a todo el billete, se tomarón las muestras de esta manera, ya que así el programa tendrá más opción de identificar la denominación en el menor tiempo posible, dado que encuentra más coincidencias con mayor cantidad de información almacenada.

Los nombres de las muestras fueron asignados de la siguiente manera:

Por ejemplo: **b5d3**, este hace referencia al billete de cinco dólares, colocando la inicial de bilete “b” el valor “5” dólar “d” y el número al final hace relación al número de muestra tomada.

A continuación, se presentan las muestras obtenidas de los billetes de cinco, diez y veinte dólares, cabe recalcar que el proceso mencionado en los párrafos anteriores se repitió para cada uno de los billetes y se obtuvieron los siguientes resultados: Figura 6.4 y Figura 6.5.

MUESTRAS BILLETE DE 5 DÓLAR

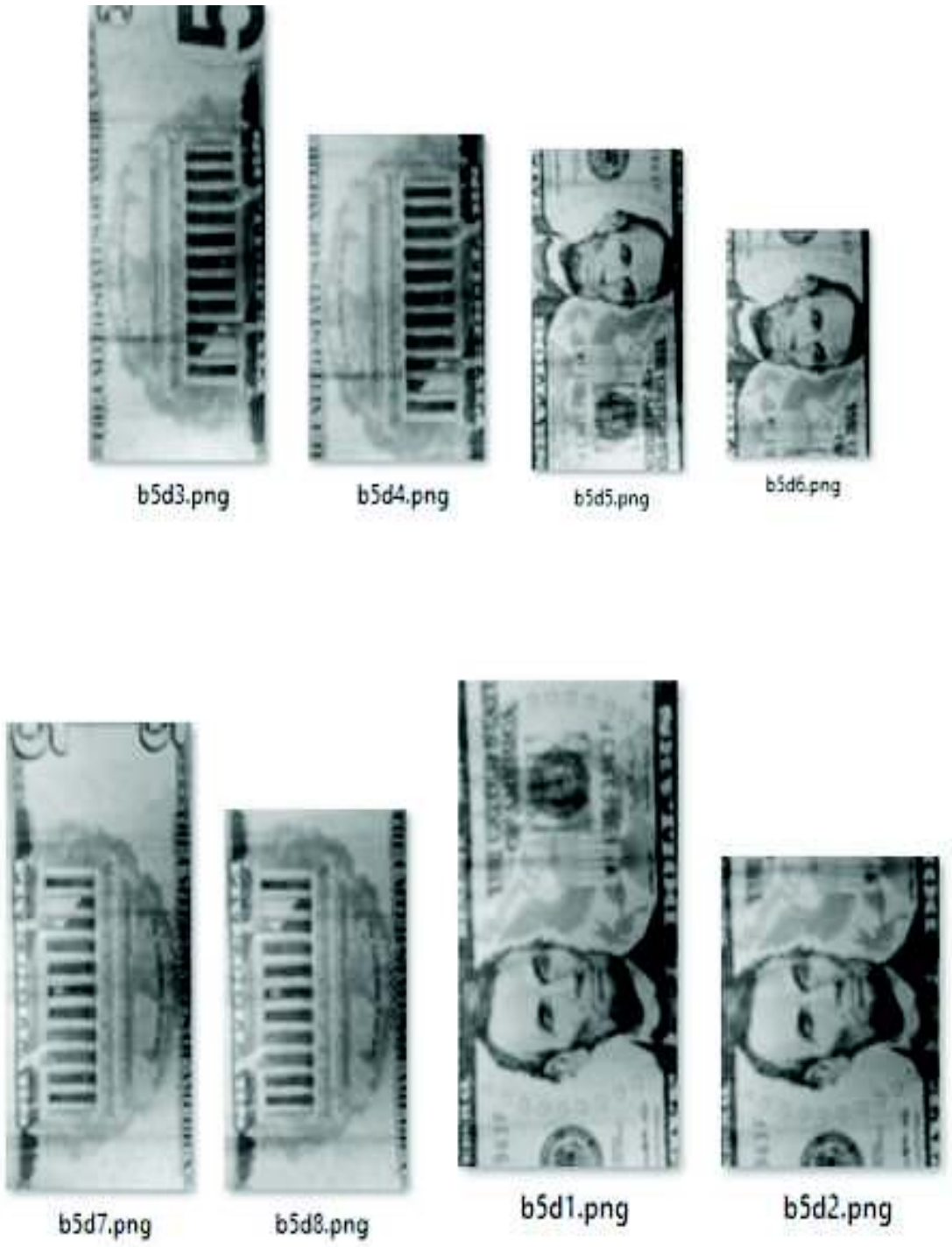

Figura 6. 3.Muestras billete de 5 dólares

MUESTRAS BILLETE DE 10 DÓLAR

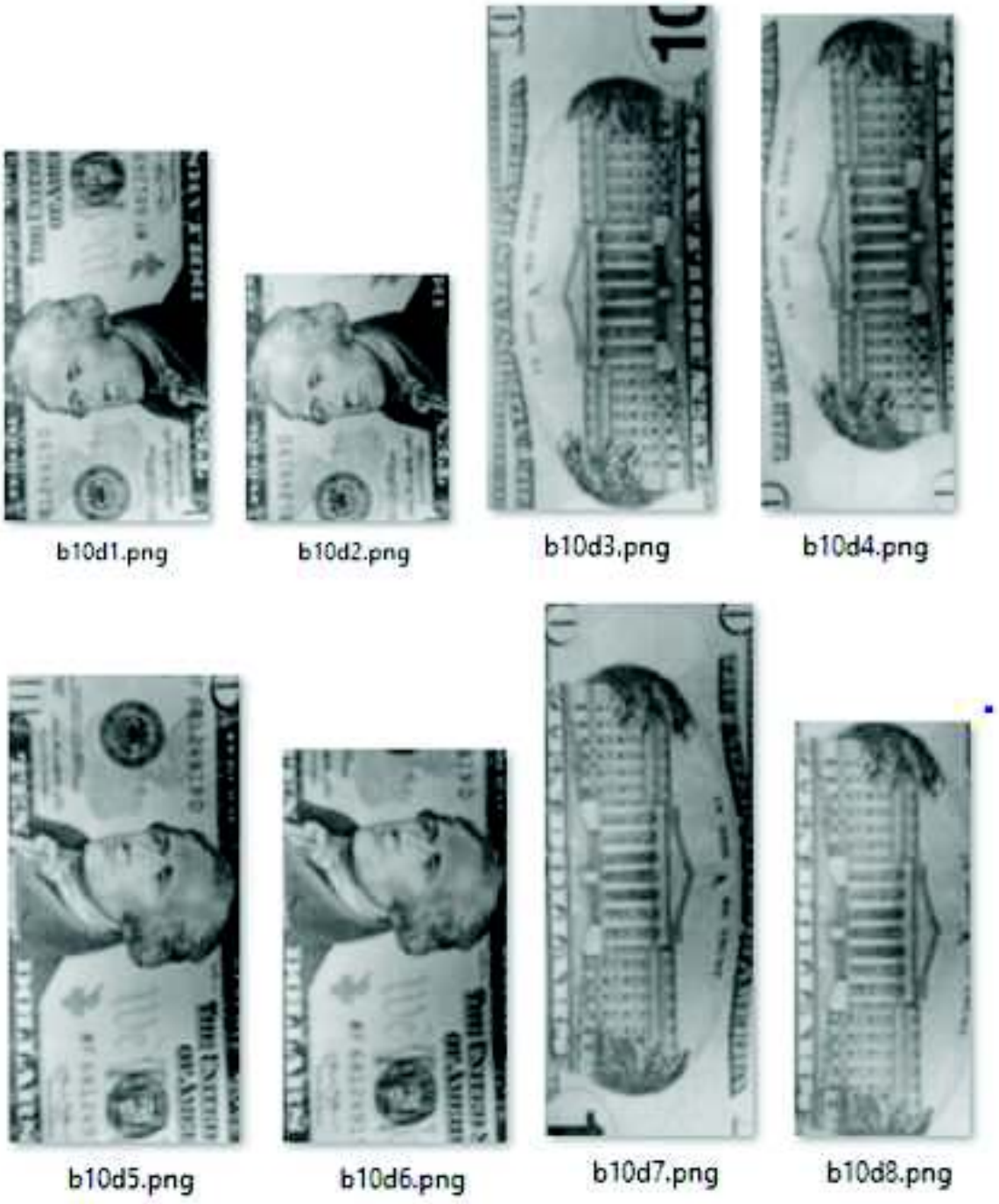

Figura 6. 4. Muestras billete de 10 dólares

MUESTRAS BILLETE DE 20 DÓLAR

Figura 6. 5. Muestras billete de 20 dólares

ANEXO C: PRUEBAS DEL PROGRAMA CON BILLETES SEGÚN LOS REQUERIMIENTOS

Como se observa en la Figura 6.6, el programa está constantemente tomando capturas de billete a ser consultado, luego la *Raspberry* procesa las capturas, para posterior, comparar los resultados de esa captura con las imágenes base y finalmente imprimir la denominación del billete "**Billete de 51 dolares**".

Quiere decir que la parte que capturó del billete se asemeja a la muestra número uno del billete de cinco dólares inicialmente almacenado, esto se puede corroborar con las muestras que se encuentran en el Anexo B. y de igual manera informa al usuario mediante un mensaje de voz.

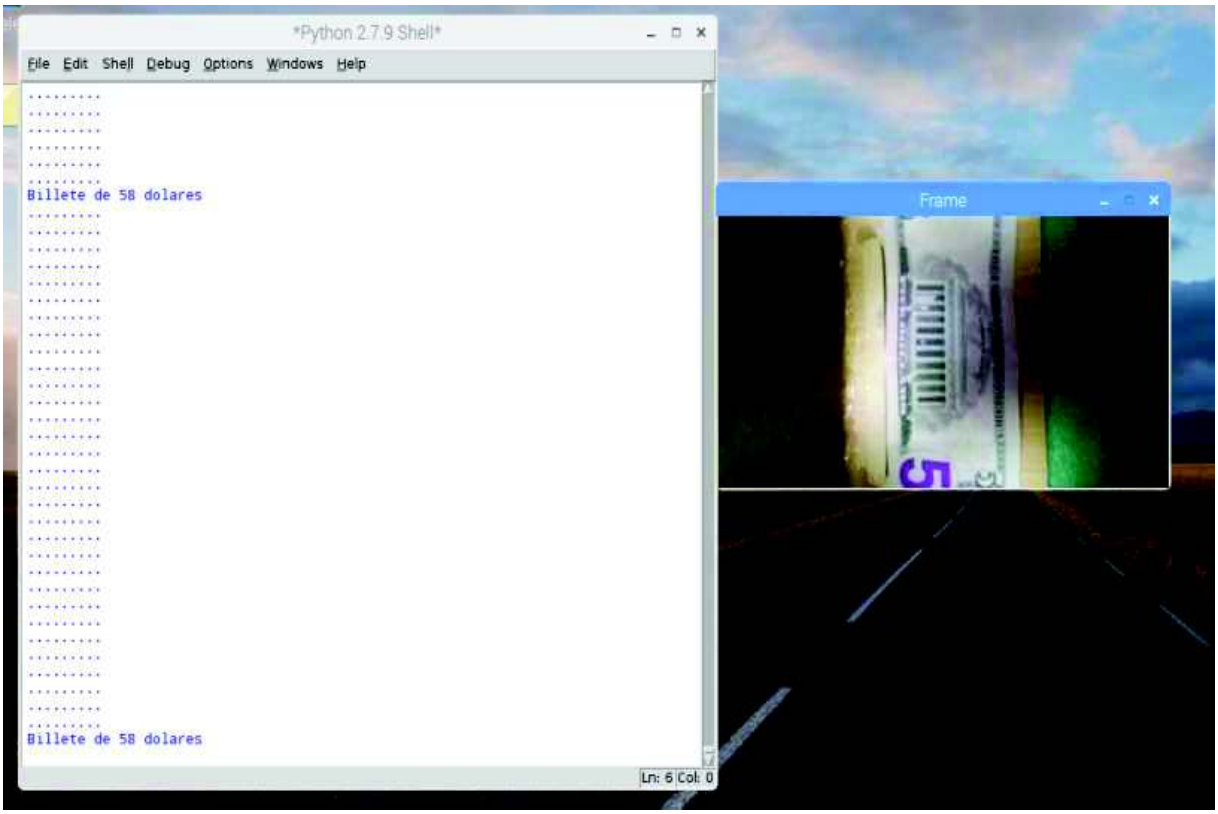

Figura 6. 6. Identificación de la denominación del billete de 5 dólares

Como se observa en la Figura 6.7, el programa está constantemente tomando capturas de billete a ser consultado, luego la *Raspberry* procesa las capturas, para posterior, comparar los resultados de esa captura con las imágenes base y finalmente imprimir la denominación del billete "**Billete de 103 dolares**".

Quiere decir que la parte que capturó del billete se asemeja a la muestra número tres del billete de diez dólares inicialmente almacenado, esto se puede corroborar con las muestras que se encuentran en el Anexo B. y de igual manera informa al usuario mediante un mensaje de voz.

Ademas, se observa que el tiempo de respuesta es menor a 5 segundos.

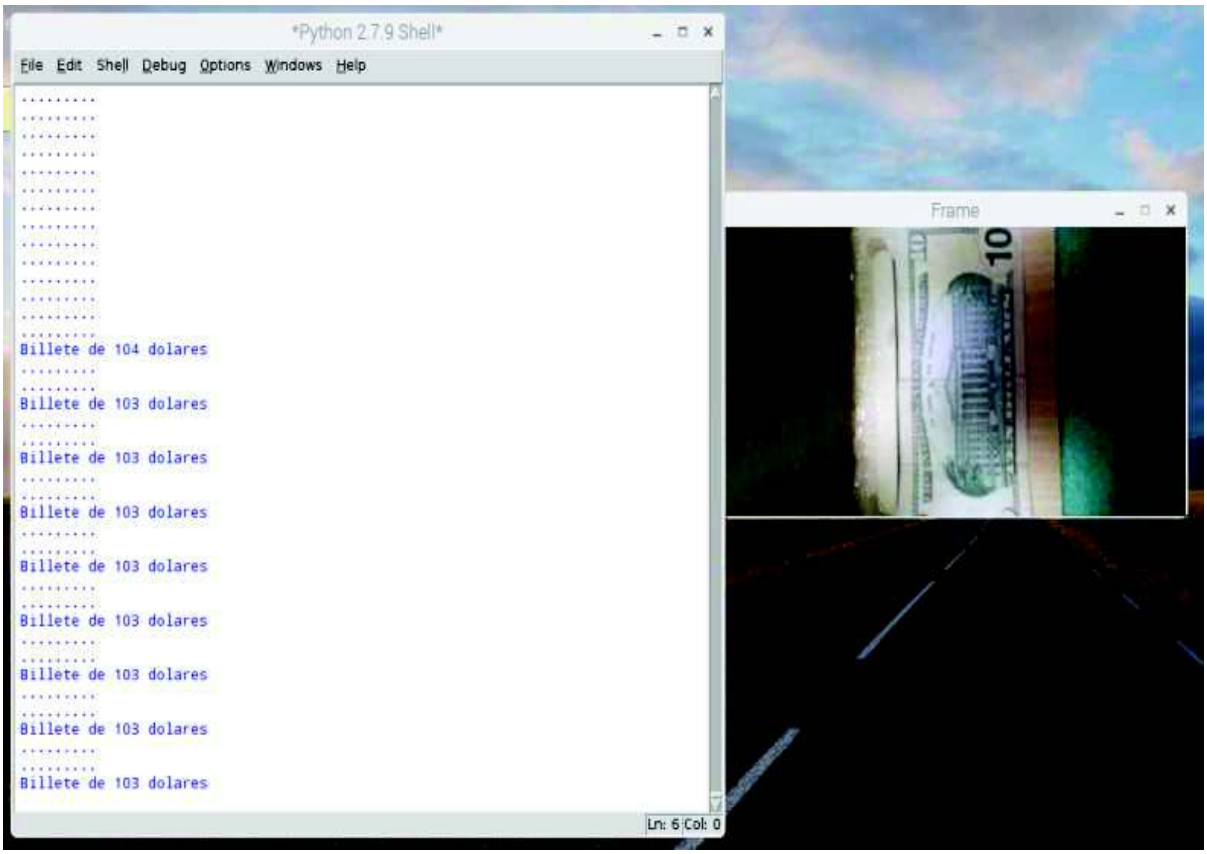

Figura 6. 7. Identificación de la denominación del billete de 10 dólares

Como se observa en la Figura 6.8, el programa está constantemente tomando capturas de billete a ser consultado, luego la *Raspberry* procesa las capturas, para posterior, comparar los resultados de esa captura con las imágenes base y finalmente imprimir la denominación del billete "**Billete de 207 dolares**".

Quiere decir que la parte que capturó del billete se asemeja a la muestra número siete del billete de veinte dólares inicialmente almacenado, esto se puede corroborar con las muestras que se encuentran en el Anexo B. y de igual manera informa al usuario mediante un mensaje de voz.

Ademas, se observa que el tiempo de respuesta es mayor a 10 segundos, esto puede ser debido a que la cámara capturo todo el billete y se tarda mayor tiempo en procesar esa información.

Figura 6. 8. Identificación de la denominación del billete de 20 dólares

ANEXO D: PRUEBAS CON BILLETES DE OTROS PAÍSES

Como se observa en la Figura 6.9, el programa está constantemente tomando capturas de billete a ser consultado, luego la *Raspberry* procesa las capturas, para posterior, comparar los resultados de esa imagen con las imágenes base y finalmente imprimir la denominación del billete "**……..**", quiere decir que la parte que capturó la imagen no se asemeja a ninguna muestra almacenada.

Figura 6. 9. Muestras con billetes de diferente país

Como se observa en la Figura 6.10, el programa está constantemente tomando capturas de billete a ser consultado, luego la *Raspberry* procesa las capturas, para posterior, comparar los resultados de esa imagen con las imágenes base y finalmente imprimir la denominación del billete "**……..**", quiere decir que la parte que capturó la imagen no se asemeja a ninguna muestra almacenada.

Figura 6. 10. Muestras con billetes de diferente país