ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

CONSTRUCCIÓN DE UN PROTOTIPO DE DISPENSADOR AUTOMÁTICO DE FLUIDOS UTILIZANDO VISIÓN ARTIFICIAL

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN TECNOLOGÍA ELECTROMECÁNICA

CARLOS LUIS MORALES AGUAGALLO

carlos.morales @epn.edu.ec

DIRECTOR: ING. FANNY FLORES, MSc.

fanny.flores@epn.edu.ec

CODIRECTOR: ING. VÍCTOR HUGO HIDALGO, PhD

victor.hidalgo@epn.edu.ec

Quito, Enero 2019

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por CARLOS LUIS MORALES AGUAGALLO, bajo nuestra supervisión.

Ing. Fanny Flores Estévez, MSc.

DIRECTORA DE PROYECTO

Ing. Víctor Hugo Hidalgo, PhD.

CODIRECTOR DE PROYECTO

DECLARACIÓN

Yo, CARLOS LUIS MORALES AGUAGALLO, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada 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 mi derecho 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.

_____

Carlos Luis Morales Aguagallo

DEDICATORIA

Este trabajo va dedicado para mis seres queridos que han estado ahí.

Mi madre Rosa María Aguagallo por haberme dado un cuerpo fuerte y guiado mi camino con el cariño más puro, se lo dedico.

Mi padre José Gregorio Morales por haberme enseñando a no huir y enfrentarme a los caminos más duros con honor y sacrificio, esto va por usted.

Mi hermano, por ser mi apoyo, mi rival, mi amigo aquel niño que me encuentra virtudes por cada 10 defectos, esto te lo dedico.

A mi Tarzan más que una mascota parte de mí y de mi familia te fuiste hace mucho, jamás olvidaré los momentos con el balón viejo y entre quebradas, hasta el fin de los tiempos hermano mío, te quiero y esto va por ti.

AGRADECIMIENTOS

El agradecimiento sentimiento puro y sincero, que nace de lo más profundo y que ha regado sus semillas en los capítulos de cada ser.

Y como no podría ser de otra manera; sin reglas, ni formatos, y en fragmentos de epopeya inspirados en las personas que en el trascurso de la vida me brindaron la oportunidad de sentir, de vivir, de reír.

Gracias padre, por su valor y su sacrificio, gracias madre por su calor incondicional, por haber estado ahí desde el día 16 el inicio, por haber soportado a un noctámbulo que llegaba tarde al amanecer.

A mi hermano, por su confianza, por enseñarme acerca del trabajo, la humildad y la constancia. Por apoyarme en momentos donde golpeaba más fuerte la desesperanza.

Gracias familia por llenarme de momentos para remembranzas, por sus ánimos, por sus mimos, mil veces gracias.

Aquellos los puedo llamar amigos, pues me han matado el hambre, las ansias y el vicio. Es muy grato compartir un pan, una bebida, un instante cuando te lo brinda un hermano al cual no te une un vínculo de sangre.

A mis maestros.

Lic. Eduardo Suarez por enseñarme que lo más importante es lo que se lleva por dentro, y su frase:" No renuncies, que tú puedes, que no pares, que no quedes, ¿Qué No?, lo que te hace fuerte de verdad se llama sentimientos."

Gracias Ing. Fanny Flores, por su guía, su compresión, su atención y su paciencia. En los senderos de la vida hay faros que nos encaminan, es este caso con un tema de tesista y la pluma de un poeta descubrí lo que quiero hacer parte de mi vida.

Y finalmente gracias, como podría no mencionar tu melodía, por hacerme levitar, por ser aquella droga que logra calmarme, dulce aroma de vida que sabe a muerte en el mismo instante. Gracias, por resurgir y conquistarme, porque al lado del silencio me haces temblar, me curas, tensa mis músculos, me ilusionas y evitas que tanto odio me ametralle. Gracias Hip Hop.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En el siguiente proyecto se ha desarrollado un sistema de dispensación de fluidos mediante el software Open CV de procesamiento de imágenes y lenguaje de programación Python.

El prototipo de dispensador consta de un sistema de rieles de aluminio y rieles telescópicas que, con un acople de banda a los motores, se encarga del movimiento general 3D del dispositivo dispensador. Utiliza para las pruebas un carro de juguete al cual se le ha acondicionado en una de sus caras laterales una superficie que posee los rasgos de una circunferencia (orificio) dentro de un cuadrado. También, cuenta con una pantalla en donde se visualizan los procesos del sistema y que con una interfaz humana máquina controlada por un teclado se encarga de la selección de fluido, así como de su cantidad.

El prototipo consta de un sistema de sensores de visión artificial, que por medio de una cámara web, una Pi camera V2 y programación, se encarga de evaluar fotogramas hasta que los sensores ópticos detecten ya sea el nivel de un fluido por medio de reconocimiento de color de un flotador en un tanque o figuras geométricas como contornos cuadrados y circunferencias. Además, cuenta con un sensor ultrasónico usado para sensar proximidad del carro hacia el carrete trasportador.

Los sensores extraen información del entorno físico, y de acuerdo con la misma, entran en acción los actuadores. Todo esto es controlado por un algoritmo de control diseñado por el autor.

Al finalizar el proceso, dependiendo de las instrucciones de inicio se habrá dispensado la cantidad requerida o se habrá detenido el proceso y los actuadores regresarán a las posiciones iniciales para comenzar de nuevo con otro proceso de dispensación.

ABSTRACT

Through the following project an, automatic fluid distribution system has been developed. The system is based on Python programming language and Open CV, an image processing software. 3D movements over x, y and z axes, are developed with the aid of a band motor, aluminum and telescopic railing systems.

Moreover, the system prototype processes artificial vision. In order to fulfill this task, a Pi Camera V2 and Web Camera are used. Each photogram is evaluated until detecting the fluid level, by color or shapes recognition.

Else, an ultrasonic sensor is used to measure the distance between the target and the distribution system. Consequently, a mobile entity reaches the target, based on the measured distance.

1. INTRODUCCIÓN.

Los problemas que se podrían presentar al utilizar métodos tradicionales para manejar fluidos, abordan algunos campos dependiendo del tipo de industria, proceso o fluido. Se los puede clasificar como problemas que afectan a la salud del ser humano y problemas de riesgo de accidente.

En primer lugar, la manipulación directa de ciertos fluidos, cuyas características físicas o químicas resultan nocivas y afectan a corto, mediano o largo plazo la salud del individuo en cuestión, ya sea por contacto directo o inhalación de vapores tóxicos.

Así también, el inadecuado uso de una instalación surtidora de fluidos peligrosos o una falla eventual como un cortocircuito o un cerrillo encendido, pueden ser detonantes para un accidente.

De acuerdo con el estudio matemático de “La teoría de las colas”, que establece una base teórica de servicio que se puede esperar para obtener un determinado recurso y que para este caso, los elementos del modelo matemático son de naturaleza variable, generando así periodos de buen y mal servicio. [1]

De acuerdo a los problemas anteriormente descritos, se plantea una alternativa de manejo de fluidos, basada en visión artificial.

En estos días, con el incesante crecimiento del mundo industrial, para un producto final ya sea de servicio o físico, no solo se toma en cuenta la calidad sino que también se busca seguridad, rapidez y bajo costo de producción. Esto ha encaminado a que las industrias busquen soluciones para resolver dichas problemáticas, dando nacimiento a métodos innovadores de automatización a otro nivel. [2]

Estos nuevos métodos de automatización pretenden reducir el error causado por el factor humano. Además, pueden trabajar en ambientes hostiles que afectan el desempeño ideal de un trabajador sin que el nivel de producción se afecte de manera significativa.

Por esto, con este proyecto se pretende dar a conocer la versatilidad y adaptabilidad que posee un módulo Raspberry Pi para el control de motores y nivel de fluidos [3]

1.1 MARCO TEÓRICO

- **Visión artificial**

La visión artificial es una parte de la inteligencia artificial donde por medio de programación, señala características relevantes capturadas en una imagen con el fin de producir información para ser tratados por un procesador. En la **Figura 1.1** se muestra los componentes que forman parte de un sistema de visión artificial.

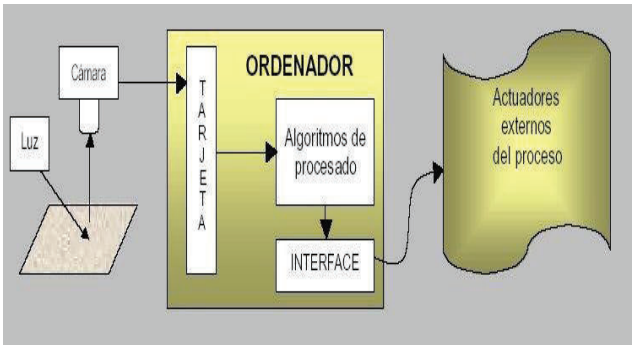

Figura 1. 1. Sistema de visión artificial. [4].

- **Aspectos importantes en un sistema de visón Artificial**

➢ **Iluminación**

Para analizar un objeto por medio de una captura de cámara, es necesario proporcionar ha dicho objeto de una iluminación adecuada con la finalidad de resaltar los rasgos más relevantes del objeto a analizar. La iluminación es la parte más importante para los sistemas de visión artificial.

➢ **Adquisición**

Esta tarea es realizada por cámaras, rayos x y ultrasónicos, cuya función es capturar los rayos reflejados por los objetos y convertirlos en información digital o eléctrica para poder ser procesados.

➢ **Procesamiento**

Este es un conjunto de tareas que realiza un procesador con los datos adquiridos y que tras aplicar varias técnicas y sub procesos, permite la extracción de las características necesarias y requeridas para su interpretación. [4]

➢ **Actuadores**

Generalmente los sistemas con visión artificial forman parte de grandes procesos pertenecientes a grandes sistemas; los resultados son entregados a actuadores (motores o bombas) que realizan la acción mecánica en el sistema. [5]

- **Rasberry Pi**

La Raspberry Pi versión 3B es una computadora de bajo costo de tamaño reducido, que utiliza un teclado, mouse y se puede conectar a una pantalla LCD, TV o a otro computador. Este pequeño dispositivo permite incursionar en sistemas operativos tipo Linux como Raspbian. Es capaz de cumplir todas las actividades que una computadora de escritorio realiza, desde procesar textos hasta ser una consola de juegos [6]. En el **Anexo A** se presenta teoría complementaria acerca de los modelos de Raspberry Pi y sus características.

- **Open CV**

Open CV u "Open Source Computer Vision Library" es una biblioteca de visión por computador de código abierto compatible con Mac OS X, Windows, Linux y que posee desarrollo para lenguajes como Python, Ruby, Matlab y otros. [7]

Open CV tiene por objetivo poner a disposición estructuras de visión por computadora fáciles, útiles y eficientes para programadores. En la **Figura 1.2** se muestran algunas tareas que se pueden realizar mediante Open CV. En el **Anexo A. 2** se presenta como instalar las extensiones necesarias para trabajar con Open CV.

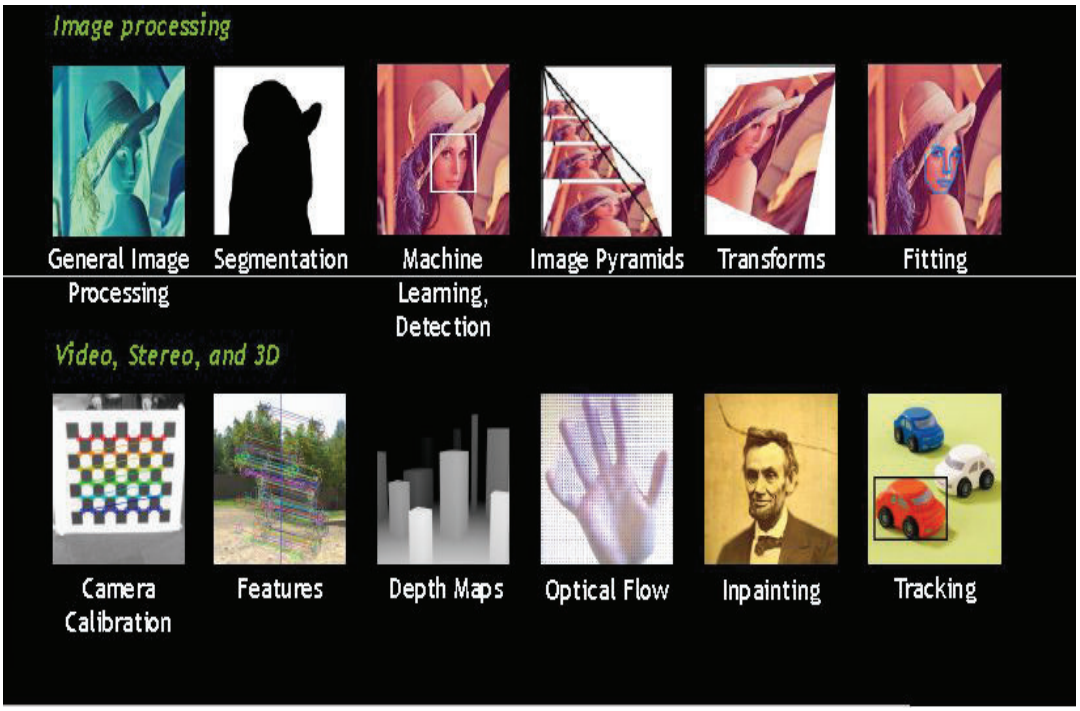

Figura 1. 2.Tareas con Open CV. [7]

- **Modelo de color RGB**

Este es un modelo de suma de colores que permite, en programación, caracterizar los colores en forma de arreglos, arrays o vectores. **Figura 1.3.**

Las siglas RGB corresponden a los colores Red (Rojo), Green (Verde), Blue (Azul).

En la Tabla 1 se muestra el código RGB para los colores primarios y secundarios.

Tabla 1. Códigos RGB de colores primarios y secundarios. [8]

Colores	**Código RGB**
Negro	(0,0,0)
Blanco	(255,255,255)
Azul	(0,0,255)
Rojo	(255,0,0)
Verde	(0,255,0)
Morado	(255,0,255)
Amarrillo	(255,255,0)
Celeste	(0,255,255)

El grado de porción de los colores se codifican en 3 cuentas de manera que para cada color se tendrá 3 valores de 0 a 255; es decir, cuanto mayor sea su valor, su color primario aportará mayor intensidad en la mezcla.

Figura 1. 3. Colores primarios y secundarios RGB. [8]

En la Tabla 2 se presenta la diferencia de orden de los valores de las cuentas de los códigos de los colores primarios en RGB y BGR.

Tabla 2. Códigos de colores primarios RGB a BGR. [8]

Color	**Código RGB**	**Código BGR Open CV**
Rojo	(255,0,0)	(0,0,255)
Azul	(0,0,255)	(255,0,0)
Verde	(0,255,0)	(0,255,0)

- **Modelo de color HSV**

El modelo de procesamiento de imágenes HSV (**Figura 1.4)**, en coordenadas polares maneja los argumentos de *Hue* (Tono), *Saturation* (Saturación) y *Value* (Valor de Luminancia).

Como se muestra en la **Figura 1.5** para los parámetros "saturación" y "valor" de esta función en Open CV, se toma valores de 0 a 255 cada uno. El parámetro del "tono" toma valores de 0 a 179 para algunas versiones de Open CV.

Tono: Es el argumento que indica el color evaluado; es decir, la longitud de onda del color captado por la cámara.

Saturación: Aprecia la cantidad de luz blanca presente en un tono.

Valor: Este argumento está relacionado con la luminancia e indica el brillo de la luz emitida o reflejada por una superficie.

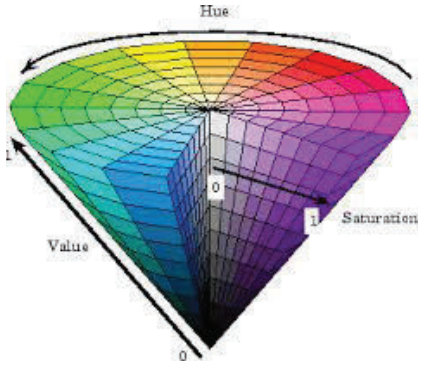

Figura 1. 4. Modelo de color HSV. [9]

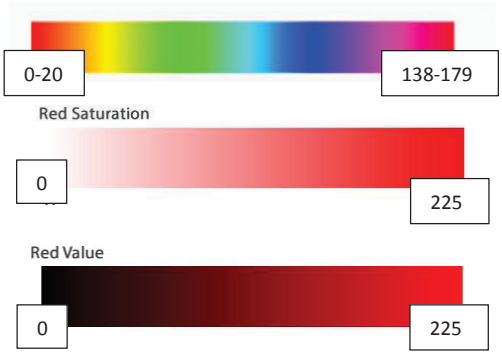

Figura 1. 5. Escalas de tono, Saturación y Valor de luminancia del modelo de color HSV para rojo. [9]

En la tabla 3 se muestra el código de colores en formato HSV en rangos alto y bajo de intensidad para los colores primarios.

Tabla 3. Valores de colores primarios en el modelo HSV. [9]

Colores en HSV	Código de rango de valores para colores primarios
rojo-bajos1	[0,60,70]
rojo-altos1	[13, 250, 253]
rojo-bajos2	[138,75,70]
rojo-altos2	[179, 255, 255]
verde-bajos	[44,49,49])
verde-altos	[86, 255, 255]
azul-bajos	[90,60,60]
azul-altos	[125, 255, 255]

- **Coordenadas por píxel**

Este sistema de coordenadas es usado para la localización de un pixel en una imagen o fotograma. Es muy parecido al sistema de coordenadas cartesianas, con los valores positivos ordenada (y) hacia abajo y se limita a las dimensiones del fotograma normalmente de 640x480 pixeles. **Figura 1.6.**

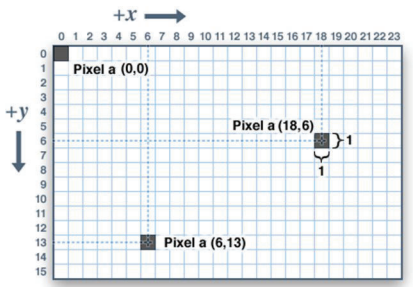

Figura 1. 6. Sistema de coordenadas por pixeles. [10]

- **Funciones de dibujo Open CV- Python**

➢ **Línea**

Función: cv2.line(objeto,(x_1,y_1),(x_2,y_2),color BGR, grosor)

Función de escritura que se utiliza para dibujar un segmento línea, se utiliza la función *line* que usa los siguientes parámetros:

- ✓ Objeto que es la imagen o fotograma de entrada.
- ✓ Coordenada del punto 1 en pixeles (x_1,y_1).
- ✓ Coordenada del punto 2 en pixeles (x_2,y_2).
- ✓ Color BGR en forma de vector (b,g,r).
- ✓ Grosor del segmento de línea.

➢ Circunferencia

Función: cv2.circle(objeto, (x, y), r, color BGR, grosor)

Función de escritura que se utiliza para dibujar una circunferencia se utiliza la función *circle* que usa los siguientes parámetros:

- ✓ Objeto que es la imagen o fotograma de entrada.
- ✓ Coordenada del centro en pixeles (x,y).
- ✓ Radio de la circunferencia.
- ✓ Color BGR en forma de vector (b,g,r).
- ✓ Grosor de la circunferencia.

➢ Rectángulo

Función: cv2.rectangle(objeto, (x_1,y_1), (x_2,y_2),color BGR, grosor)

Función de escritura que se utiliza para dibujar un rectángulo se utiliza la función *rectangle* que usa los siguientes parámetros:

- ✓ Objeto que es la imagen o fotograma de entrada.
- ✓ Coordenada del vértice superior izquierdo en pixeles (x_1,y_1).
- ✓ Coordenada del vértice inferior derecho en pixeles (x_2,y_2).
- ✓ Color BGR en forma de vector (B,G,R).
- ✓ Grosor del segmento del contorno del rectángulo.

➢ Texto

Función: cv2.putText (objeto, "texto", (x,y),

cv2.FONT_HERSHEY_SIMPLEX, tamaño, color BGR, Grosor)

Función de escritura que se utiliza para escribir un texto sobre una imagen se hace uso de la función *putTex* tal como se muestra y usa los siguientes parámetros:

- ✓ Objeto que es la imagen o fotograma de entrada.
- ✓ Texto.
- ✓ Coordenada de inicio del texto en pixeles (x,y).
- ✓ Tipo de letra cv2.FONT_HERSHEY_SIMPLEX .
- ✓ Tamaño de la letra 0-10.
- ✓ Color BGR en forma de vector (b,g,r).
- ✓ Grosor de la letra.

➢ Texto de tiempo real

Función de escritura que se utiliza para escribir datos en un fotograma como hora y fecha, se debe incluir al algoritmo las bibliotecas datetime.

Los objetos Fecha y Hora se crean mediante los siguientes algoritmos.

Fecha: D=date.today()

Hora: H=datetime.now().hour, M=datetime.now().minute, S=datetime.now ().second

➢ Operación de imagen por bits

O también llamada operaciones *bitwise;* en programación digital permiten operar los objetos bit a bit e incluyen operadores lógicos como *and, or, not y xor.*

Estos operadores permiten sumar, restar, invertir y discriminar los pixeles de un fotograma cuya información está en bits.

Los parámetros que estas funciones manejan son:

- ✓ Objeto 1 de entrada
- ✓ Objeto 2 de entrada
- ✓ Objeto de salida Resultados.

• Modulación por ancho de pulsos o PWM

Es un sistema de control DC en el que se varía la señal de voltaje con la finalidad de activar o controlar actuadores. Esta señal consta de 2 valores de tensión uno en alto (1) y otro en bajo (0). En robótica es un método muy utilizado para regular la velocidad y la fuerza de un motor DC.

• Motores a paso Bipolares

Este tipo de motores tienen 4 cables de salida y necesita de la manipulación en su alimentación de voltaje. Por el cambio de dirección de corriente en las bobinas de sus devanados por medio de un puente H. Su conexión se aprecia en la **Figura 1.7.**

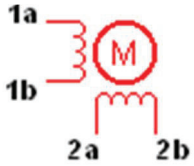

Figura 1. 7. Conexión motor a pasos 4 hilos. [11]

Para su control, se sigue la siguiente configuración lógica, presentada en la Tabla 4.

Tabla 4. Configuración lógica para motores a paso bipolares. [11]

Paso	1A	1B	2A	2B
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	1	0

- **Puente H**

Para que un motor a pasos trabaje, necesita un amplificador ya que la potencia que proporciona un controlador es relativamente pequeña e insuficiente para mover cargas. En el **Anexo A. 5** se presenta el driver L298N utilizado en el proyecto que contiene 2 puentes H.

- **Puerto GPIO**

Protocolo de entrada y salida de datos bit a bit. Existen 2 maneras para establecer el modo de numeración del puerto GPIO.

- Por medio del nombre propio del pin.

Esta numeración se establece mediante el algoritmo:

Función: GPIO.setmode(GPIO.BCM)

- Por medio de la posición del pin.

Esta numeración se establece mediante el comando:

Función: GPIO.setmode(GPIO.BOARD)

En la **Figura 1.8**, se muestra en detalle el nombre y numeración de cada pin de una Raspberry Pi.

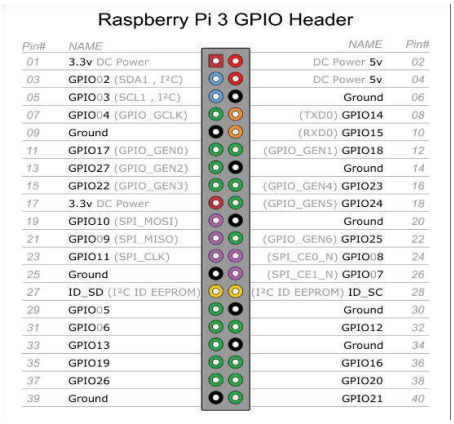

Figura 1. 8. Raspberry Pi 3 GPIO Header. [14]

- **Configuración del puerto GPIO**

➢ **Entrada**

Un pin se configura como entrada mediante el siguiente algoritmo:

Función: GPIO.setup(número o nombre del pin, GPIO.IN)

➢ **Salida**

Un Pin se configura como salida mediante el siguiente algoritmo:

Función: GPIO.setup(número o nombre del pin, GPIO.OUT)

Una vez establecido el pin como salida, se le da un valor de 1 o 0 mediante los siguientes algoritmos:

GPIO.output(número o nombre del pin ,1)

GPIO.output(número o nombre del pin ,0)

El valor de 1 lógico también se puede representar mediante *True* o *GPIO.HIGH*; de igual manera para el valor de 0 lógico se puede representar con *False* o *GPIO.LOW*.

- **Visión artificial con una Web Cam.**

A este proceso se le llama “Captura de imágenes en forma de frames”.

Para la extracción de datos con una Web Cam se procede a trabajar por fotos, fotogramas o imágenes que en conjunto forman un video, para lo cual se hace uso de bucles *while*. [15]

Funciones necesarias:

➢ **import cv2**

Función que llama a la biblioteca de Open Cv.

➢ **objeto= cv2.VideoCapture(“# del puerto web cam”)**

Para capturar un video en vivo con cámara web se debe crear un objeto donde su argumento es el número del puerto USB que usa el dispositivo. Cuando solo hay un dispositivo conectado, su número es cero para cualquier puerto del módulo Raspberry Pi.

- **ret,objeto=objeto.read()**

Esta función evalúa el funcionamiento de la cámara web, devolviendo valores booleanos. 1 si la cámara proporciona datos y 0 si no proporciona datos. En conjunto con el bucle infinito *while,* las fotografías que la cámara proporciona forman un video sin sonido durante el ciclo infinito.

- **cv2.imshow("Nombre de la ventana",objeto)**

Esta función despliega las imágenes obtenidas por la cámara web en una ventana donde sus argumentos son el nombre de la ventana y los datos obtenidos por la cámara web.

- **cv2.waitKey(T)&0xFF==ord("TECLA"):**

Esta función ejecuta una orden cuando una tecla es presionada, donde T es el parámetro tiempo en milisegundos de ejecución y "TECLA" es el parámetro que indica qué tecla se usa para ejecutar la acción.

0xFF se usa para no tener errores en sistemas operativos de 64 bits.

- **objeto.release()**

Esta función apaga la cámara.

- **cv2destroyAllWindows()**

Cierra todas las ventanas abiertas por proceso de Open CV.

En la **Figura 1.9**, se presenta el código para extraer imágenes de un entorno físico en forma de frames con una webcam.

```
import cv2

cam= cv2.VideoCapture(0)

while(1):
    ret,imagen=cam.read()

    cv2.imshow("INTERFAZ",imagen)

    if cv2.waitKey(1)&0xFF==ord("q"):
            break
            cam.release()

            cv2destroyAllWindows()
```

Figura 1. 9. Código Python para el uso de una Web Cam. [15]

- **Visión artificial con una Pi Camera V2.**

La principal diferencia con una Web Cam común es que para esta cámara es necesario un algoritmo más desarrollado, como se muestra a continuación. [16]

Funciones necesarias:

- **PiRGBArray**

  Extensión de Python que hace un llamado de inclusión a la biblioteca *picamera.array*; permite trabajar las imágenes obtenidas como matrices, utilizando el modelo de color RGB.

- **PiCamera**

  Extensión de Python que hace un llamado de inclusión a la biblioteca PiCamera; trabaja con una Pi Camera.

- **Numpy as np**

  Extensión de Python que hace un llamado de inclusión a la biblioteca *numpy*. Proporciona un soporte matemático para trabajar datos en forma de matrices y vectores.

- **Itertools**

  Extensión de Python que hace un llamado de inclusión a la biblioteca *intertools*. Proporciona un soporte eficiente de herramientas básicas de memoria y filtros.

- **cam.capture_continuous(rawCapture,format="bgr",use_video_port=True)**

  Esta función crea un objeto que evalúa de forma booleana si la PiCamera entrega o no datos en el modelo de colores RGB. Necesita de un lazo *for* en conjunto con la sentencia *rawCapture.truncate(0).*

- **objeto=frame.array**

  Esta función toma los datos de un frame o fotograma y refleja sus datos RGB en forma de matriz. La imagen tratada tiene características iguales que en el algoritmo *ret,objeto=objeto.read()*, usado para una web cam común.

En la **Figura 1.10**, se presenta el código para extraer imágenes en forma de frames con una Pi Camera.

```python
#-PROGRAMA COMUN-#
# -*-coding:utf-8 -*-
#--------------------------Bibliotecas-----------------------------#
from picamera.array import PiRGBArray
from picamera import PiCamera
import cv2
import numpy as np
import itertools
#------------Inicio de camara  por frames como video ----------------#
cam=PiCamera()
cam.resolution=(520,540)                #Resolución de la imagen.
cam.framerate=25
rawCapture=PiRGBArray(cam,size=(520,540)) #Tamaño de la ventana.
time.sleep(0.01)

#-----------PROGRAMA PRINCIPAL PARA INTEGRAR FUNCIONES----------------#
def inicio():
    for frame in cam.capture_continuous(rawCapture,format="bgr",use_video_po
rt=True): #Lazo for para  arrancar el video.
        ima=frame.array
        #Frames transformados en forma de matriz RBG.
        salida(ima)                                    #Funcion Salida#
        rawCapture.truncate(0)
        if cv2.waitKey(1) & 0xFF == ord("q"):           #Apagar Camara y ve
ntanas opencv con la tecla "q"#
            break
            cv2.destroyAllWindows()
            cam.close()
def salida(ima):
    cv2.imshow("SENSOR VISUAL",ima)

#----------------------LLAMADA DEL PROGRAMA-------------------------#
inicio()
```

Figura 1. 10. Código Python para el uso de una Pi Camera. [16]

- **Transformación a escala de grises**

Función: cv2.cvtColor(objeto, cv2.COLOR_BGR2GRAY)

Esta función transforma una imagen a escala de grises. Sus parámetros son:

- ✓ El objeto o imagen de entrada.
- ✓ El código de tipo de conversión cv2 .COLOR_BGR2GRAY.

- **Filtro Bilateral**

Función: cv2.bilateralFilter(objeto, diámetro, desviación estándar 1, desviación estándar 2)

La función permite el suavizado de una imagen o fotograma y además mantiene afilado los contornos. [16] Sus argumentos son:

- ✓ Imagen de entrada.
- ✓ Diámetro de la vecindad de pixeles.
- ✓ Desviación estándar o cercanía de los pixeles en el espacio del color, pueden ser entre 1 y 10 pixeles.
- ✓ Desviación estándar en el espacio de las coordenadas del color.

- **Detección de Filos o bordes**

Función: cv2.Canny(objeto, umbral mínimo, umbral máximo)

Esta función permite detectar los filos o bordes de los objetos presentes en una imagen. Utiliza un umbral de histéresis**,** donde clasifica cuál pixel es borde y cuál no y toman valores entre 0 y 255.

- **Detector de circunferencias**

Función: cv2.HoughCircles (objeto, cv2.cv.CV_HOUGH_GRADIENT, acumulador, mínima distancia de centros, param1, param2, minRadius, maxRadius). [18]

Esta función encuentra circunferencias en un objeto o fotograma en escala de grises ya filtrada y trabajada con la función para detectar bordes. Los argumentos que maneja son:

- ✓ Imagen u objeto de entrada.
- ✓ Método de detección único HOUGH_GRADIENT.
- ✓ Acumulador inverso de la resolución de la imagen puede ser 1 o 2. Distancia mínima entre los centros de los círculos encontrados.
- ✓ Parámetro 1 umbral del contorno circular.
- ✓ Parámetro 2 representa el umbral del acumulador para los centros de los círculos detectados.
- ✓ Radio mínimo del círculo.
- ✓ Radio máximo del círculo.

Nota: La inclusión del siguiente algoritmo sirve para filtrar posibles circunferencias que se extienden más allá del tamaño de la imagen o fotograma.

np.round(circles[0, :]).astype("int")

- **Extracción de contornos**

Función: cv2.findContours(objeto,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

Esta función encuentra contornos cerrados y los aproxima a figuras geométricas definidas; además, almacena los datos en forma de vector. Esta función cuenta con tres argumentos: [19]

- ✓ Imagen u objeto de entrada previamente tratada.
- ✓ *cv2.RETR_TREE* es el método de extracción o jerarquía del contorno.
- ✓ *cv2.CHAIN_APPROX_SIMPLE)* es el método de aproximación del contorno.

- **Área de un contorno**

Función: cv2.contourArea(“Área en pixeles”)

Este algoritmo cuantifica el área en pixeles cuadrados de un contorno cerrado.

- **Determinación de contorno cerrado o curva**

Función: cv2.arcLength(cont, True)

Este algoritmo evalúa con 1(True) para un contorno cerrado o con 0 (False) para un segmento de curva. [19]

- **Aproximación a una figura geométrica mediante vértices**

Función: cv2.approxPolyDP(cont, 0.1 * arc_len, True)

Este algoritmo evalúa la cantidad de vértices posibles de un contorno, proporcionada por la función extracción de contornos.

Para el correcto uso de este algoritmo es necesario considerar un correcto valor porcentual del segundo argumento, normalmente debe ser bajo 10% (0.1).

Una vez obtenido el número de vértices, se puede pre-clasificar la figura geométrica del contorno analizado mediante la inclusión del siguiente algoritmo.

if (len(approx) == # de vertices).

- **Extracción del centro de masa de un contorno**

Función: cv2.moments(cont)

Esta función permite calcular rasgos de interés de un objeto inmerso en un fotograma como son el centro de masa o el área. Además, con una inclusión apropiada de esta función, se puede discriminar áreas inferiores. Este proceso se llama eliminación del ruido. [20]

Centro X = int(M["m10"] / M["m00"])

Centro Y = int(M["m01"] / M["m00"])

- **Dibujo de un contorno**

Función: cv2.drawContours(objeto, [approx], índice de contornos,color BGR,grosor)

Esta función dibuja el contorno individual de un objeto en una imagen y contiene 5 argumentos que son: [21]

- ✓ Imagen u objeto de entrada establecida.
- ✓ Cantidad de vértices evaluados por la función *approxPolyDP* como una lista de Python.
- ✓ Índice de contornos para dibujar todos los contornos es -1.
- ✓ Color BGR en forma de vector (B,G,R).
- ✓ Grosor de la letra.

- **Conversión del modelo de color BRG a HSV**

Función: cv2.cvtColor(objeto, cv2.COLOR_BGR2HSV)

Para esta conversión la función utiliza los argumentos:

- ✓ Imagen u objeto de entrada.
- ✓ El código de tipo de conversión *cv2 .COLOR_BGR2HSV.*

- **Máscara de color**

Una máscara de color es una imagen que contiene solo 2 colores: blanco para el color que se requiere caracterizar y negro para lo demás**.** [21]

Función: cv2.inRange(objeto HSV, Código HSV color bajo, Código HSV color alto)

Esta función sirve para reconocer los pixeles de una imagen que están dentro de un rango. La función maneja 3 argumentos que son:

- ✓ Imagen u objeto de entrada en formato HSV.
- ✓ Color HSV en bajo.
- ✓ Color HSV en alto.

2. METODOLOGÍA

Las metodologías utilizadas para el alcance de los objetivos establecidos, han tenido un enfoque experimental-deductivo-analítico. Por consiguiente, se enfatizan los siguientes métodos de investigación.

2.1 Método deductivo

Este método ha tomado parte importante en el comienzo del proyecto, aportando tras una investigación, un conjunto de ideas y datos generales que se han ido correlacionando y han establecido un camino de cómo desarrollar el proyecto.

2.2 Método analítico

El método analítico ha sido utilizado en el estudio minucioso de las partes necesarias y esenciales de la investigación, que aporta el método deductivo. Gracias a este método se ha podido entender la naturaleza estructural de los algoritmos en Python para el tratamiento de imágenes e incorporar nuevas ideas con la finalidad de realizar el sistema.

2.3 Método experimental

Este método ha sido utilizado en la realización de pruebas de ensayos de acierto y error, como para algoritmos y estructuras tras el control deliberado de las variables físicas y no físicas en las pruebas de campo de cada uno de las partes del sistema.

2.4 Descripción de la metodología usada.

Se investigó sobre los fundamentos de los elementos que intervinieron en el desarrollo del prototipo de dispensador automático de fluido. Así, se empezó con el lenguaje de programación Phyton y el módulo Raspberry. Se realizó una maqueta hecha de aluminio, acero negro (tol).También, se acondicionó un carro de juguete para que porte el objetivo a detectar siendo este una circunferencia dentro de un cuadrado. Se desarrolló algoritmos de visión artificial con el uso de una cámara web y una Pi camera para detectar contornos cuadrados, circulares colores rojos y azules. Además, se desarrolló algoritmos para el control de motores DC y de un sensor ultrasónico. Se procedió a la calibración de cada una de las partes del sistema, mismos que tras ser probados han sido montados sobre estructura correspondiente. Una vez montado el sistema, se procedió con las pruebas pertinentes con la finalidad de encontrar, evaluar y reducir los posibles errores del sistema.

3. RESULTADOS Y DISCUSIÓN

En la **Figura 3.1** se presenta el proyecto finalizado y el carro portador del objetivo a localizar que es una circunferencia dentro de un cuadrado.

Figura 3. 1. Prototipo de dispensador de fluidos (Exterior) y carro.

En la **Figura 3.2** se presenta el interior del Prototipo de dispensador de fluidos.

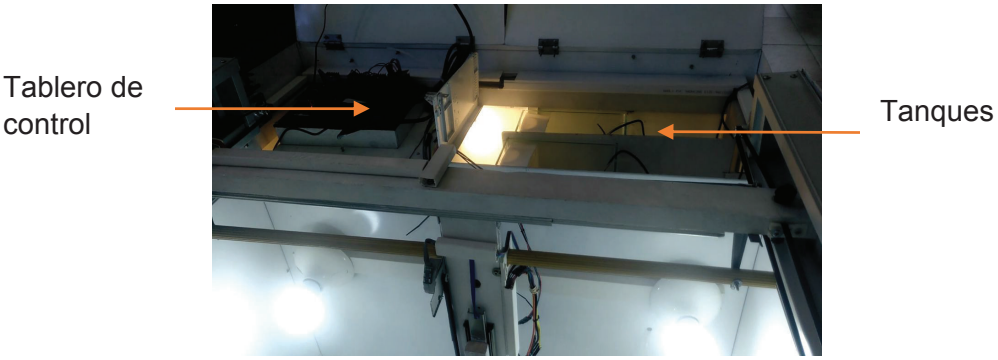

Figura 3. 2. Prototipo de dispensador de fluidos (Interior).

En la **Figura 3.3** se presenta el tablero de control y el carrete trasportador de mangueras, cámara web y sensor ultrasónico.

Figura 3. 3. Carrete transportador y tablero de control.

3.1 Identificación de requerimientos del prototipo.

Para el desarrollo del prototipo, ha sido necesario seleccionar y fabricar los componentes, dependiendo de factores como: disponibilidad en el mercado, flexibilidad y compatibilidad. Además, se ha requerido del desarrollo de los siguientes programas:

- ✓ Detector de circunferencias.
- ✓ Detector de contornos cuadrados.
- ✓ Detector de colores.
- ✓ Medidor de distancia.
- ✓ Control PWM.

En la Tabla 5, se muestra la distribución de pines GPIO para controlar los actuadores.

Tabla 5. Distribución de pines del puerto GPIO en el proyecto.

Actuador	Descripción	Nombre del pin	Número del pin
Motor a paso (KS). Mov X.	Pines que controlan un motor a paso para el movimiento X del sistema de rieles de movimiento.	GPIO 4 GPIO 17 GPIO 27 GPIO 22	7 11 13 15
Motor a paso (KS). Mov Z.	Pines que controlan dos motores a paso para el movimiento Z del sistema de rieles de movimiento.	GPIO 5 GPIO 6 GPIO 13 GPIO 19	29 31 33 35
Motor a paso (Mitsuki) Mov. Y.	Pines que controlan un motor a paso para el movimiento Y del sistema de rieles de movimiento.	GPIO 18 GPIO 23 GPIO 24 GPIO 25	12 16 18 22
Motor DC.	Pin que controla un motor DC para el movimiento de la extensión que porta las mangueras.	GPIO 26	37
Bomba sumergible. Tanque 1.	Pin que controla la bomba 1.	GPIO 12	32
Bomba sumergible. tanque 2	Pin que controla la bomba 2.	GPIO 16	36
Ultrasónico	Pines que controlan el sensor ultrasónico.	GPIO 20 GPIO 21	38 40

3.2 Diseño del prototipo

- **Detector de circunferencias**

Para la detección de circunferencias y consecuentemente la extracción de sus rasgos de interés como posición y radio, se sigue el siguiente proceso.

- Se procede a capturar las imágenes en forma de frames por medio de una cámara web y se transforman a escala de grises, como se muestra en la **Figura 3.4.**

Figura 3. 4. Trasformación de BGR a Escala de grises.

- En este punto, se trabaja las imágenes por medio de los filtros “Bilateral” y “Gausiano”, como se muestran en la **Figuras 3.5** y la **Figura 3.6** respectivamente, con la finalidad de eliminar el ruido de imagen y mantener afilados los contornos.

Figura 3. 5. Imagen escala de grises filtro bilateral.

Figura 3. 6. Imagen escala de grises filtro gausiano.

- Se transforma las imágenes a manera de resaltar los filos y bordes de cada contorno presente, mediante la función *canny* .**Figura 3.7.**

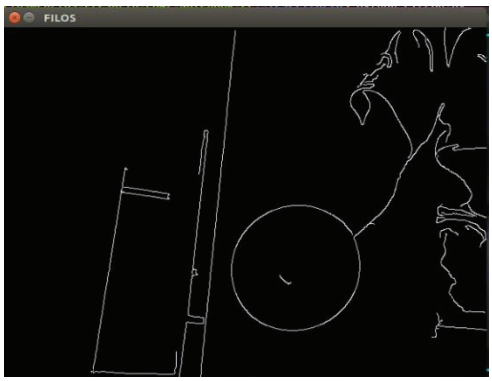

Figura 3. 7. Bordes de una imagen.

- Finalmente, se utiliza la función cv2.HoughCircles para encontrar las circunferencias dentro de cada imagen y extraer sus rasgos de interés. Los rasgos de interés como radio y posición se visualizan sobre la imagen de entrada por medio de la función de escritura *putTex*. Además, se ha establecido un radio mínimo de 55 pixeles y un máximo de 150 pixeles para que posibles circunferencias fuera de este rango, no se tomen en cuenta. El programa deja de funcionar al presionar la letra “q”.

  El resultado se presenta en la **Figura 3.8.**

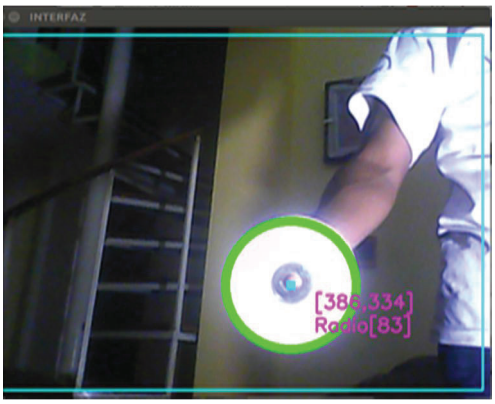

Figura 3. 8. Detector de circunferencias.

En la **Figura 3.9** se presenta el código del sensor detector de circunferencias.

```
1
2  # -*-coding:utf-8 -*-
3  #--------------------------Bibliotecas-------------------------------#
4
5  import cv2
6  import time
7  import numpy as np
8
9  #----------PROGRAMA PRINCIPAL PARA INTEGRAR FUNCIONES----------------#
10 def circunferencia():
11     cam= cv2.VideoCapture(0)
12     while(1):
13         ret,imagen=cam.read()
14         cv2.imshow("INTERFAZ",imagen)
15         circulo1(imagen)
16         if cv2.waitKey(1)&0xFF==ord("q"):
17             break
18             cap.release()
19             cv2destroyAllWindows()
20
21 #-------------------------------FUNCIONES-----------------------------#
22 def circulo1(imagen):
23     global x,r,y
24     gris = cv2.cvtColor( imagen, cv2.COLOR_BGR2GRAY )
25     cv2.imshow("MUNDO 1",gris)
26     gris = cv2.bilateralFilter(gris,1,10,120)
27     cv2.imshow("MUNDO 2",gris)
28     gris= cv2.GaussianBlur(gris, (5,5),0)
29     cv2.imshow("MUNDO 3",gris)
30     filos  = cv2.Canny( gris,50,250)
31     cv2.imshow("MUNDO OBSCUROS4",filos)
32     circles = cv2.HoughCircles(filos, cv2.cv.CV_HOUGH_GRADIENT, 1, 200, para
   m1=30, param2=45, minRadius=0, maxRadius=0)
33     if circles is not None:
34         circles = np.round(circles[0, :]).astype("int")
35         for (x, y, r) in circles:
36             if (r>55) and (r<150):
37                 mensaje(imagen)
38
39 def mensaje(imagen):
40     global x,r,y
41     cv2.circle(imagen, (x, y), r, (0, 255, 0), 10)
42     cv2.rectangle(imagen, (x - 5, y - 5), (x + 5, y + 5), (255,255,0), -1)
43     cv2.rectangle(imagen, (10,10), (630,470), (255,255,0), 2)
44     cv2.putText (imagen, "[" + str (x) + "," + str (y) + "]", (x+ 30, y+ 30)
   ,
45     cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 255),2)
46     cv2.putText (imagen, "Radio"+"[" + str (r) + "]", (x+ 30, y+ 60),
47     cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 255),2)
48     cv2.imshow( 'INTERFAZ', imagen )
49
50 #-----------------------LLAMADA DEL PROGRAMA-------------------------#
51 circunferencia()
52
```

Figura 3. 9. Código Python para la detección de circunferencias.

- **Detector de contornos cuadrados**

Para la detección de contornos cuadrados y la extracción de sus rasgos de interés como posición y centro se sigue el siguiente proceso. [21]

- Capturar las imágenes en forma de frames del medio físico por medio de la cámara web. **Figura 3.10**.

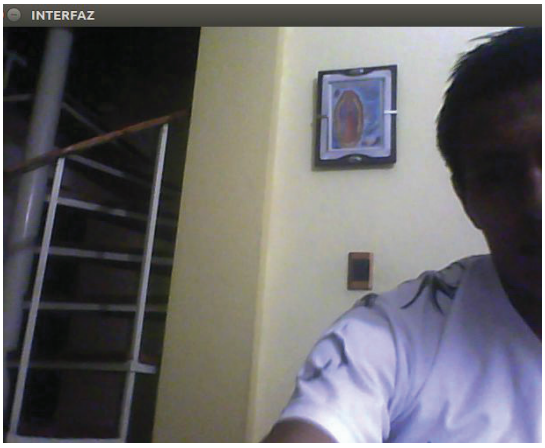

Figura 3. 10. Imagen inicial.

- Conversión a escala de grises mediante la función *cvtColor.***Figura 3.11.**

Figura 3. 11. Imagen en escala de grises.

- Filtrado de las imágenes en escala de gises mediante las funciones gaussiano y bilateral.

- Afilado de filos y bordes mediante la función *canny*. **Figura 3.12.**

Figura 3. 12. Imagen de contornos.

En este punto se procede a discriminar los contornos por su número de vértices. Puesto que el sensor es de contornos cuadrados, su número de vértices es de 4, número que utiliza la función *approxPoly para determinar contornos de forma cuadrada*. También, con la finalidad de discriminar áreas muy pequeñas o muy grandes pertenecientes a contornos de forma cuadrada, se ha estimado un valor mínimo de 9000 pixeles y máximo de 50000 pixeles. Además, mediante la función *moments*, se extrae el centro de masa del contorno cuadrado. El contorno y su centro se presentan resaltados sobre la imagen de entrada por medio de las funciones de escritura. En la **Figura 3.13** se presenta el resultado del sensor detector de contornos.

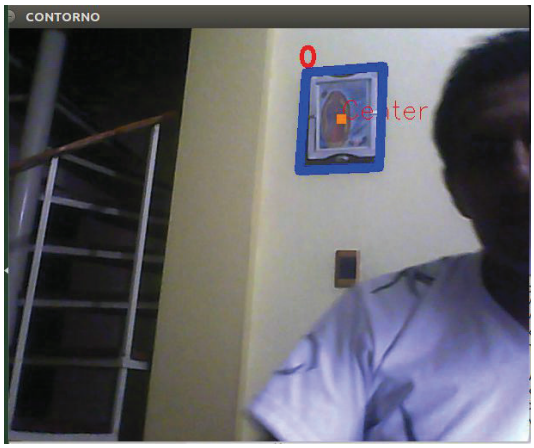

Figura 3. 13. Detector de contornos cuadrados.

En la **Figura 3.14** se presenta el código del detector de contornos cuadrados.

```
# -*-coding:utf-8 -*-
#--------------------------Bibliotecas-------------------------------#
import cv2
import time
import numpy as np
#----------PROGRAMA PRINCIPAL PARA INTEGRAR FUNCIONES----------------#
def contorno():

    cam= cv2.VideoCapture(0)
    while(1):
        ret,imagen=cam.read()
        cv2.imshow("INTERFAZ",imagen)
        bordes1(imagen)

        if cv2.waitKey(1)&0xFF==ord("q"):
            break
            cap.release()
            cv2destroyAllWindows()

def bordes1(imagen):

    gris = cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
    gris= cv2.GaussianBlur (gris, (5,5),0)
    gris = cv2.bilateralFilter( gris, 1,7, 100)
    gris= cv2.GaussianBlur(gris, (1,1), 100)
    edges  = cv2.Canny( gris, 4, 90)
    cv2.imshow("MUNDO OBSCURO",edges)
    contours, hierarchy = cv2.findContours(edges,cv2.RETR_TREE,cv2.CHAIN_APP
ROX_SIMPLE)
    for cont in contours:
        if cv2.contourArea(cont)>9000 and cv2.contourArea(cont)<50000:
            arc_len = cv2.arcLength( cont, True )
            approx = cv2.approxPolyDP( cont, 0.1 * arc_len, True )
            if ( len( approx ) == 4 ):

                M = cv2.moments( cont )
                cX = int(M["m10"] / M["m00"])
                cY = int(M["m01"] / M["m00"])
                cv2.putText(imagen, "Center", (cX, cY), cv2.FONT_HERSHEY_SIM
PLEX, 1.0, (0, 0, 255), 1)
                cv2.rectangle(imagen, (cX - 5, cY - 5), (cX + 5, cY + 5), (0
, 128, 255), -1)
                x,y,w,h=cv2.boundingRect(cont)
                cv2.putText(imagen,"0",(x,y),cv2.FONT_HERSHEY_SIMPLEX,1.0,(0
,0,255),3)
                cv2.drawContours( imagen, [approx], -1, ( 255, 0, 0 ), 10)
                cv2.imshow('CONTORNO',imagen)
#-----------------------LLAMADA DEL PROGRAMA-------------------------#
contorno()

```

Figura 3. 14. Código Python para la detección de contornos cuadrado.

- **Detector de colores**

Para la detección de colores en una imagen o conjunto de imágenes mediante Open CV, se procesa la imagen de la siguiente manera. [21]

- Capturar las imágenes en forma de frames.
- Convertir el o los fotogramas del modelo de color BGR al modelo de color HSV.
- Establecer un rango de color.
- Crear una máscara de color.
- Aplicar una operación lógica de imágenes en función de sus bits.

- **Detector de colores azules**

- Se procede a extraer las imágenes en forma de frames con una webcam. **Figura 3.15.**

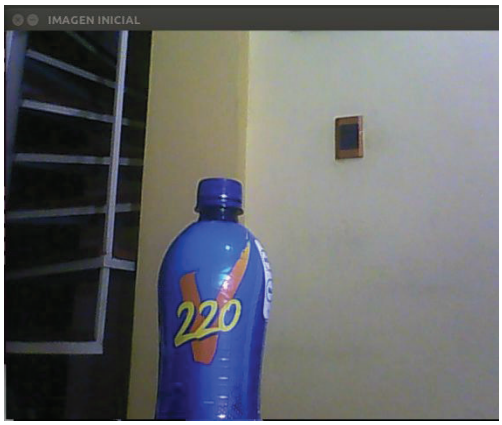

Figura 3. 15. Imagen inicial del detector de colores azules.

- Se convierten las imágenes del modelo de color BRG a HSV. **Figura 3.16**.

Figura 3. 16. Imagen HSV del detector de colores azules.

- Se procede a crear una máscara de color.

  Aquí, se ha establecido 2 arreglos de valores que son color en alto y color en bajo. Estos arreglos limitan el rango del color a detectar.

  Azul bajo = [80, 50, 50])

  Azul alto = [150, 250, 250])

  Para eliminar ruidos y discriminar la presencia de colores azules pequeños dentro de la imagen o fotograma, se hace una discriminación por medio del tamaño de un área. De modo que para detectar áreas grandes se ha previsto un valor de cuenta de 2000000.

  La máscara de color se muestra en la **Figura 3.17.**

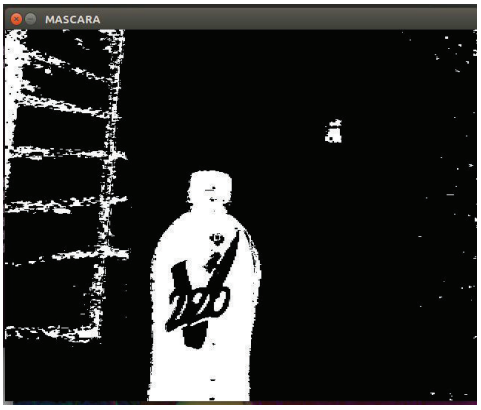

Figura 3. 17. Máscara del detector de colores azules.

- Se crea de manera opcional una máscara de color, con el color establecido a detectar con la ayuda de la función *bitwise_and*. **Figura 3.18.**

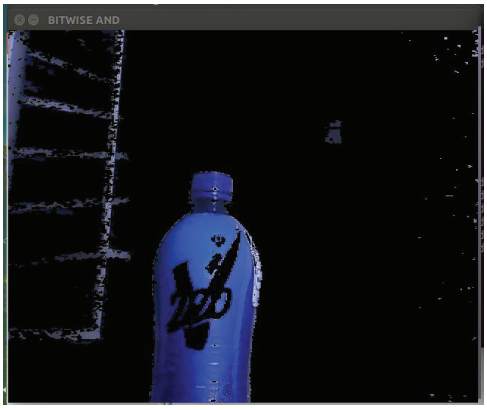

Figura 3. 18. Máscara de color del detector de colores azules.

- Finalmente, mediante la función *moments* se extrae el centro de masa de la máscara de color y por medio de las funciones de escritura se dibuja sobre la imagen inicial la ubicación del centro de masa. El resultado del sensor se muestra en la **Figura 3.19.**

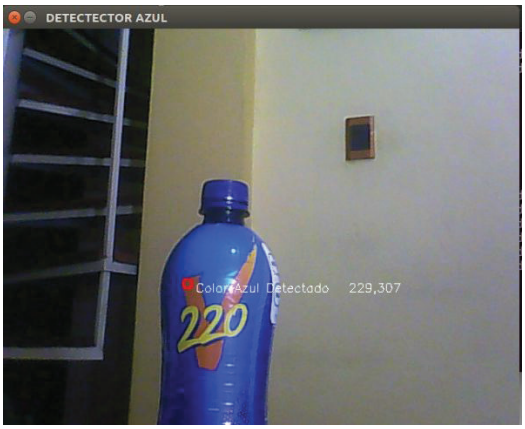

Figura 3. 19. Detector de presencia del color azul.

En la **Figura 3.20** se muestra el código del detector de presencia del color azul.

```
# -*-coding:utf-8 -*-
#----------------------------Bibliotecas-------------------------------#
import cv2
import time
import numpy as np
#-----------PROGRAMA PRINCIPAL PARA INTEGRAR FUNCIONES-----------------#
def inicio():
    cam= cv2.VideoCapture(0)
    while(1):
        ret,imagen=cam.read()
        cv2.imshow("IMAGEN INICIAL",imagen)
        azul(imagen)
        if cv2.waitKey(1)&0xFF==ord("q"):
            break
            cap.release()
            cv2destroyAllWindows()
#----------------------------FUNCIONES---------------------------------#
def azul(imagen):
    hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
    cv2.imshow("HSV",hsv)
    lower_blue = np.array([110,50,50])
    upper_blue = np.array([130,255,255])
    mask = cv2.inRange(hsv, lower_blue, upper_blue)
    cv2.imshow("MASCARA",mask)
    res = cv2.bitwise_and(imagen,imagen, mask= mask)
    cv2.imshow("BITWISE AND",res)
    moments = cv2.moments(mask)
    area = moments['m00']

    if(area > 2000000):


        mx = int(moments['m10']/moments['m00'])
        my = int(moments['m01']/moments['m00'])
        cv2.rectangle(imagen, (mx-5, my-5), (mx+5, my+5),(0,0,255), 2)
        cv2.putText(imagen, "Color Azul Detectado  "+ str(mx)+","+str(my), (
mx+10,my+10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255), 1)
        cv2.imshow("DETECTECTOR AZUL",imagen)
#-----------------------LLAMADA DEL PROGRAMA-------------------------#
inicio()

```

Figura 3. 20. Código del detector de presencia del color azul.

- **Detector de colores rojos**

➢ Se procede a extraer las imágenes en forma de frames con una webcam. La imagen inicial se presenta en la **Figura 3.21.**

Figura 3. 21. Imagen inicial del sensor de colores rojos.

➢ Se convierten las imágenes del modelo de color BRG a HSV. **Figura 3.22.**

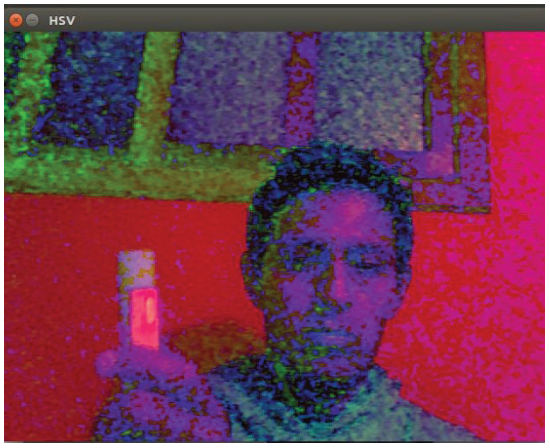

Figura 3. 22. Imagen HSV del sensor de colores rojos.

➢ Se construye una máscara de color donde se ha estimado 2 arreglos que limitan el rango del color a detectar; además para eliminar ruidos y discriminar presencia de colores rojos dentro de la imagen se hace una discriminación por medio del tamaño de un área en unidades de pixeles con un valor de 200000.

Rojo bajo = [170, 50, 50])
Rojo alto = [179, 250, 250])

En la **Figura 3.23** se muestra la máscara de color.

Figura 3. 23. Máscara del sensor de colores rojos.

- De manera opcional, con la ayuda de la función *bitwise_and* se puede crear una máscara de color con el color establecido como se muestra en la **Figura 3.24**, en este caso el color rojo.

Figura 3. 24. Máscara de color del sensor de colores rojos.

Nota: Se debe tomar en cuenta que la tonalidad del color rojo en la escala del modelo de color HSV toma valores de 0 a 20 o de 139 a 179 aproximadamente.

- Finalmente, mediante la función *moments* se extrae el centro de masa de la máscara de color y por medio de las funciones de escritura se dibuja sobre la imagen inicial la ubicación del centro de masa. El resultado del detector se muestra en la **Figura 3.25**.

Figura 3. 25. Detector de presencia de colores rojos

En la **Figura 3.26** se muestra el código del detector de presencia de colores rojos.

```
#Programa detector de colores rojos
# -*-coding:utf-8 -*-
#----------------------------Bibliotecas-------------------------------#
import cv2
import time
import numpy as np
#-----------PROGRAMA PRINCIPAL PARA INTEGRAR FUNCIONES-----------------#
def inicio():

    cam= cv2.VideoCapture(0)
    while(1):
        ret,imagen=cam.read()
        cv2.imshow("IMAGEN INICIAL",imagen)
        rojo(imagen)

        if cv2.waitKey(1)&0xFF==ord("q"):
            break
            cap.release()
            cv2destroyAllWindows()

def rojo(imagen):

    hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
    cv2.imshow("HSV",hsv)
    rojoup=np.array([179,255,255],np.uint8)
    rojodown=np.array([170,70,70],np.uint8)
    mask = cv2.inRange(hsv, rojodown, rojoup)
    cv2.imshow("MASCARA",mask)
    res = cv2.bitwise_and(imagen,imagen, mask= mask)
    cv2.imshow("BITWISE AND",res)

    moments = cv2.moments(mask)
    area = moments['m00']

    if(area > 200000):


        mx = int(moments['m10']/moments['m00'])
        my = int(moments['m01']/moments['m00'])
        cv2.rectangle(imagen, (mx-5, my-5), (mx+5, my+5),(0,0,255), 2)
        cv2.putText(imagen, "Color Rojo Detectado   "+ str(mx)+","+str(my),
(mx+10,my+10),            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255), 1)
        cv2.imshow("DETECTECTOR ROJO",imagen)

inicio()
```

Figura 3. 26. Código del detector de presencia del color rojo.

- **Control de un motor DC**

Este control es el más sencillo y consiste en alimentar a las borneras de un motor DC con una señal amplificada por un puente H con valores de 1 y 0 lógico que salen del puerto GPIO.

Se ha establecido el pin 39 como GND o cero lógico y el pin 40 como uno lógico. Cuando el motor arranca, se muestra el mensaje *MOTOR ON* y cuando se detiene se muestra el mensaje *MOTOR OFF.* Además, cada ciclo de encendido y apagado dura un segundo con la finalidad de poder apreciar el movimiento del motor DC.

Se ha considerado la utilización de un contador, de modo que cuando su valor llegue a 4, el programa se detenga y muestre el mensaje *END*.

En la **Figura 3.27** se muestra el algoritmo de control.

```
1  #-ARRANQUE Y PARO DE UN MOTRO DC  4 CICLOS-#
2  # -*-coding:utf-8 -*-
3  #--------------------------Bibliotecas-----------------------------#
4  import RPi.GPIO as GPIO
5  import time
6  #----------------------------PROGRAMA-----------------------------#
7  GPIO.setmode(GPIO.BOARD)
8  GPIO.setup(40,GPIO.OUT) #PIN 40
9  GPIO.setup(39,GPIO.OUT) #PIN 39
10 contador=0
11 while True:
12     print "MOTOR ON"
13     GPIO.output(40,1)
14     GPIO.output(39,0)
15     time.sleep(1)
16     print "MOTOR OFF"
17     GPIO.output(40,0)
18     GPIO.output(39,0)
19     time.sleep(0)
20     contador =contador +1
21     if (contador ==4):
22         break
23         print "END"
24
```

Figura 3. 27. Código para el arranque y paro de un motor DC 4 ciclos.

- **Control de un motor a pasos 4 hilos**

Para el control de este tipo de motores es necesario realizar el cálculo del número de pasos en una revolución de un motor para incorporar estos datos al algoritmo de control. El ángulo de paso del motor viene dado en los datos de placa del motor o su *datasheet*.

$$\textbf{Número de pasos}(\textbf{NP}) = \frac{\textbf{360 grados}}{\textbf{ángulo de paso del motor}} \qquad \text{(Ec.1)}$$

NP $_{\text{motor KS}}$= 360/1.8= **200 pasos x vuelta.**

NP $_{\text{motor Mitsuki}}$ =360/7.8=**48 pasos x vuelta.**

Este número, llamado constante de giro por vuelta, se dividirá para el número de pines físicos utilizados, normalmente 4.

$$\textbf{Constante de giro por vuelta}(\textbf{CG}) = \frac{\textbf{Número de pasos}}{\textbf{4}} \qquad \text{(Ec.2)}$$

CG $_{\text{motor KS}}$= 200/4=**50**

CG $_{\text{motor Mitsuki}}$=48/4=**12**

Como pines de control se ha establecido los pines 7, 11, 13, 15 para el motor KS y 12, 16, 18, 22 para el motor Mitsuki.

Para el cambio de giro de este tipo de motor, solo se invierte el orden de los vectores o arrays de la secuencia para el control de motores bipolares:

- Giro: [[1,0,1,0],[1,0,0,1],[0,1,0,1],[0,1,1,0]]
- Giro Invertido: [[0,1,1,0],[0,1,0,1],[1,0,0,1],[1,0,1,0]]

La constate de giro por vueltas se la puede modificar para aumentar o reducir el ángulo de giro de un motor según lo necesitado, si se desea 2 vueltas se duplica este valor.

Cada pulso en el control PWM se ha establecido mediante una serie de pruebas, estimando tiempos de 0.001 segundos para el motor KS y 0.01 segundos para el motor Mitsuki.

En las **Figuras 3.28 y 3.29** se presentan los algoritmos de control de los motores KS y Mitsuki respectivamente.

```
#-----------------Programa pwm motor bipolar motor KS-----------------#
# -*-coding:utf-8 -*-
#----------------------------Bibliotecas-------------------------------#
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)# Modo de # de los pines.
ControlPin=[7,11,13,15] #  Designo mis pines de trabajo.
for pin in ControlPin:# Para cada pin en la matriz Controlpin.
    GPIO.setup(pin,GPIO.OUT) # Para cada pin le instalo como salida.
    GPIO.output(pin,0)# Para cada pin le limpio con 0.
    #Secuecia  para un motor  a paso bipolar.
    seq=[[1,0,1,0],[1,0,0,1],[0,1,0,1],[0,1,1,0]]
    # un paso=1.8 grados
    # gaget resuction 360/1.8=200
    # 200 pasos una  vuelta
    # 200/4= 50= constante de giro por vuelta .
for i in range(50):
    for k in range(4):
        #50*4=200 Una vuelta.
        GPIO.output(ControlPin,seq[k])
        time.sleep(0.001)# Tiempo de trabajo por pulso
GPIO.cleanup()
```

Figura 3. 28. Código de control de una vuelta del motor a pasos bipolar KS.

```
#----- Programa pwm motor bipolar mitsuki (Avanzo)
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)

ControlPin=[12,16,18,22] #  designo mis pines de trabajo
for pin in ControlPin:#para cada pin en la matriz Controlpin
    GPIO.setup(pin,GPIO.OUT) # para cada pin le instalo como salida
    GPIO.output(pin,0)# para cada pin le limpio con 0
# un paso=7.8grados
# gaget resuction 360/7.8=48
    seq=[[0,1,1,0],[0,1,0,1],[1,0,0,1],[1,0,1,0]]

for i in range(12): #48pasos una revolucion
    #time.sleep(0.01)# retraso de 0.02s maximo
    for pin in range(4):
        GPIO.output(ControlPin,seq[pin])
        time.sleep(0.01)# tiempo velocidad
GPIO.cleanup()
```

Figura 3. 29. Código de control de una vuelta del motor a pasos Mitsuki.

- **Sensor de nivel de líquidos**

Para la visualización e ingreso de datos se ha diseñado una interfaz donde se ha considerado la facilidad de entendimiento visual con el usuario.

Para este sensor se ha utilizado la Pi Camera V2.

En la **Figura 3.30** se presenta un esquema de la interfaz de nivel de fluidos.

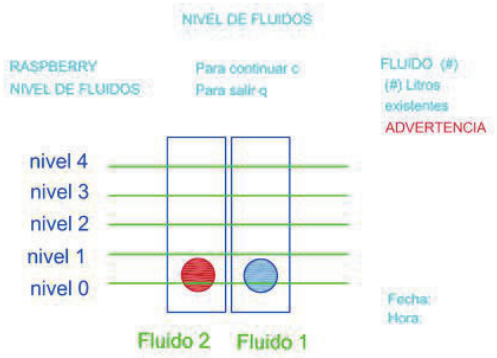

Figura 3. 30. Esquema de la interfaz de nivel de fluidos.

Como se muestra en el diseño, esta interfaz es sencilla y especifica la interacción con el usuario por medio de las teclas “c” para continuar y “q” para abandonar el proceso con un teclado; además incluye funciones como fecha, hora y mensajes de estado.

Como sensor de nivel, se ha colocado un flotador de color en cada tanque, de modo que el sensor de visión artificial evalúe la posición del centro de masa del flotador con respecto a niveles establecidos en un fotograma. Dependiendo de la cantidad de agua presente en los tanques y de la cantidad requerida, los actuadores como las bombas desempeñan la tarea correspondiente del proceso.

En la **Figura 3.31** se presenta un diseño de cada tanque, con su respectivo flotador.

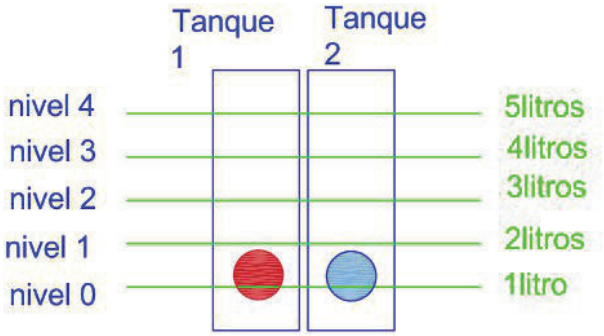

Figura 3. 31. Bosquejo de posición de los tanques.

Cada nivel representa la cantidad de 1 litro y el proceso de dispensación se efectúa si y solo si el nivel del fluido es mayor al nivel 1 y la cantidad del fluido requerido está contenida en el tanque. El nivel 0 es el nivel de referencia y es el nivel de fluido mínimo que debe existir para que la bomba permanezca sumergida y de este modo se conserve la presión en las mangueras.

El nivel de los flotadores es evaluado en escala por pixeles (Y) con un margen de tolerancia de +- 5% para cada nivel, como se muestra en la tabla 6.

Tabla 6. Niveles de fluido y correspondencia para cada fotograma.

Y pixel	Litros de fluido	Nivel
80 +- 5	4 litros dispensables	4 máx.
160 +-5	3 litros dispensables	3
240 +-5	2 litros dispensables	2
320 +-5	1 litros dispensables	1 min.
390 +-5	0 litros dispensables	0

Cada flotador es evaluado por su centro de masa y su localización en cada fotograma mediante la inclusión del sensor detector de colores rojos y azules. Para su calibración, se ha estimado una línea de color blanco que es sobrepuesta con las bases de los tanques.

Mediante el uso de las funciones de escritura de Open CV y programación Python se presenta la interfaz de nivel de fluidos. **Figura 3.32.**

Figura 3. 32. Interfaz de nivel de fluidos.

- **Interfaz de objetivo**

De igual manera, esta interfaz se ha diseñado para brindar facilidad de operación al usuario. También indica el estado del proceso, así como la fecha y hora. Esta interfaz recibe fotogramas de una Web Cam.

En la **Figura 3.33** se presenta el diseño de la interfaz para detectar el objetivo.

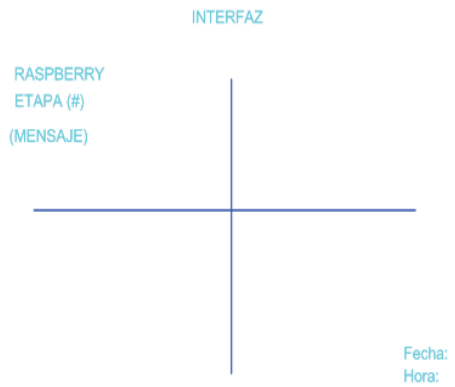

Figura 3. 33. Diseño de la interfaz del objetivo.

En esta interfaz se visualiza el reconocimiento de la carcasa del carro, el reconocimiento del objetivo (circunferencia inscrita en un cuadrado) y la ubicación del centro de masa del objetivo en el fotograma, como se aprecia en la **Figura 3.34.**

Esta interfaz también muestra líneas que se entrecruzan en el centro de cada fotograma "Coordenadas (320,240)", este centro actúa como referencia.

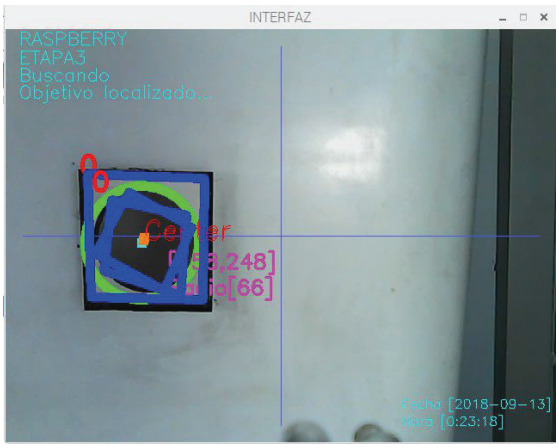

Figura 3. 34. Interfaz de objetivo.

La diferencia de posición entre el centro de masa del objeto y el centro del fotograma proporciona datos con los que en conjunto con la programación previamente establecida permite operar los actuadores tales como son los motores a paso con la finalidad de colocar el objetivo (circunferencia inscrita en un cuadrado) en el centro de la imagen.

- **Menú-selector de fluidos y cantidad**

Para el ingreso al sistema, se ha diseñado y construido un menú con fotogramas de manera que simule un selector, tanto para el tipo de fluido como para su cantidad.

Para la cantidad, se han considerado 2 niveles: 1 litro y 2 litros.

Se presenta los bosquejos del menú 1 y menú 2 en las **Figuras 3.35 y 3.36,** respectivamente.

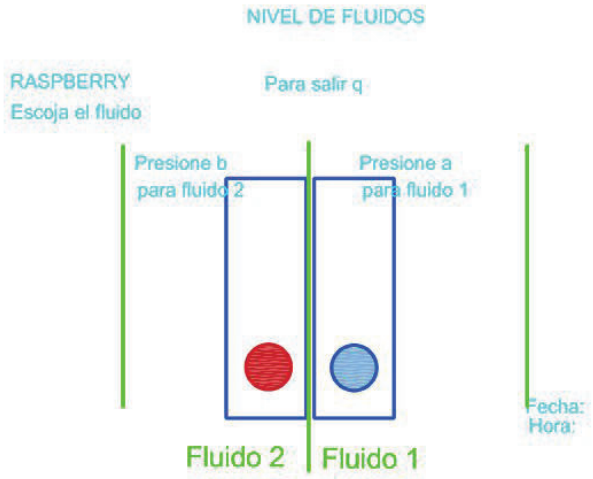

Figura 3. 35. Diseño Menú 1 selector.

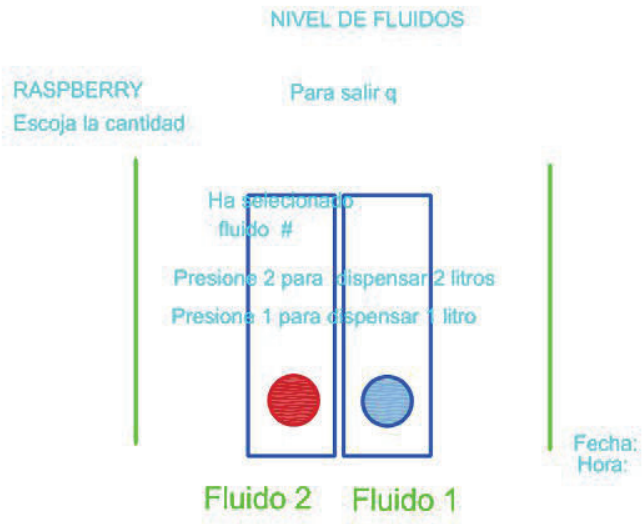

Figura 3. 36. Diseño Menú 2 selector.

Mediante las funciones de escritura de Open CV y programación Python, se presenta en las **Figuras 3.37 y 3.38,** los menús 1 y 2, respectivamente.

Menú 1.

Selector de inicio del sistema, “a” para fluido 1 y “b” para fluido 2.

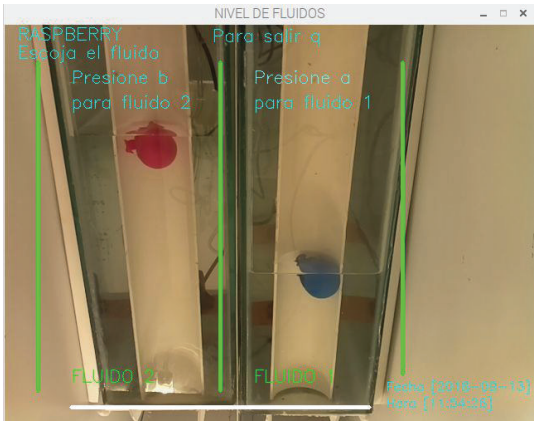

Figura 3. 37. Menú 1 selector.

Menú 2.

Selector del sistema, “1” para dispensar 1 litro y “2” para dispensar 2 litros del fluido pre-seleccionado.

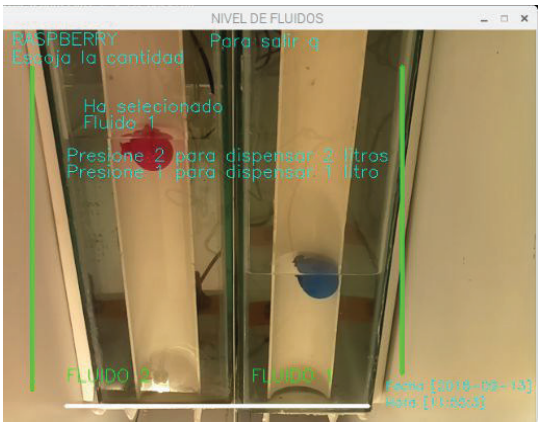

Figura 3. 38. Menú 2 selector.

- **Algoritmo de control para un ultrasónico**

Para evaluar la distancia desde el soporte que lleva las mangueras hasta la superficie perteneciente del carro que posee el objetivo (circunferencia inscrita en un cuadrado), se adapta la expresión de evaluación de distancia con un sensor ultrasónico, teniendo en cuenta que la velocidad del sonido en el aire es 340 m/s. En el **Anexo A. 4** se presenta las características del Sensor Ultrasónico HC-SR04.

$$\text{Distancia} = \text{Velocidad del sonido}/2 * (t2 - t1)$$

Se han utilizado los terminales 40 para eco y 38 para trigger del módulo Raspberry Pi.

A continuación, se muestra en la **Figura 3.39** los datos evaluados por el sensor ultrasónico.

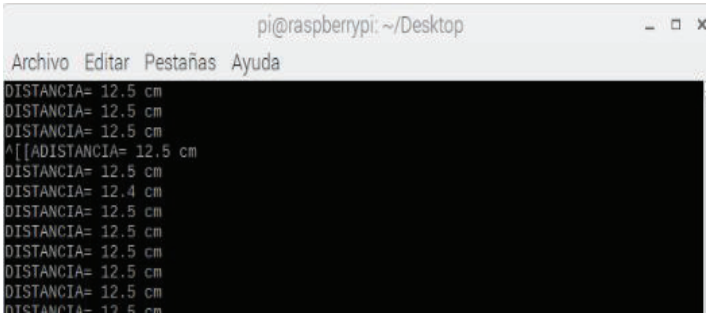

Figura 3. 39. Datos del ultrasónico en una terminal.

En la **Figura 3.40** se presenta el código de control del sensor ultrasónico.

```
#----Programa de control de sensor ultrasonico ciclo infinito----------#
# -*-coding:utf-8 -*-
#----------------------------Bibliotecas-------------------------------#
import RPi.GPIO as GPIO
import time
#----------PROGRAMA PRINCIPLA PARA INTEGRAR FUNCIONES-----------------#
def main():
    while (1):
        ultrasonico.
#-----------------------------FUNCIONES-------------------------------#
def ultrasonico():
    global z
    GPIO.setmode(GPIO.BOARD)
    GPIO.setmode(GPIO.BOARD)
    trig=40
    eco=38
    GPIO.setup(trig,GPIO.OUT)
    GPIO.output(trig,0)
    GPIO.setup(eco,GPIO.IN)
    time.sleep(0.1)
    GPIO.output(trig,1)
    time.sleep(0.1)
    GPIO.output(trig,0)

    while GPIO.input(eco)==0:
        pass
    start=time.time()

    while GPIO.input(eco)==1:
        pass
    stop=time.time()
    z=((stop-start)*17000)
    print "DISTANCIA=",round(z,1),"cm"
    GPIO.cleanup()

```

Figura 3. 40. Código de control del sensor ultrasónico.

- **Diagrama de flujo del proceso del prototipo dispensador de fluidos.**

Como se puede apreciar en la **Figura 3.41**, el programa comienza con el menú 1, mismo que es visible en la pantalla del proyecto. El menú 1 proporciona al usuario la selección de 2 opciones de fluidos, “a” para el fluido 1 y “b” para el fluido 2.

Luego, el programa despliega un segundo menú que proporciona al usuario la selección de la cantidad de fluido, “1” para un litro y “2” para dos litros. Además, el programa evalúa el nivel de fluido presente en los tanques. El programa asigna a la Pi Camera V2 para la evaluación del nivel del fluido por medio de detección de los colores de los flotadores azul y rojo.

El programa continúa con el reconocimiento de la carcasa del auto. Aquí, el programa utiliza un sensor detector del color azul para detectar la carcasa del auto. Para esto se utiliza una Cámara Web.

Luego, el prototipo realiza un escaneo buscando un contorno circular, para lo cual se usa un sensor de circunferencias. Al haberse detectado circunferencias, el programa evalúa si está inmersa dentro de un contorno cuadrado.

Luego, el prototipo realiza un proceso de centrado en los ejes axiales x, y. Luego, el programa realiza un acercamiento que termina cuando el brazo dispensador se encuentra a menos de 10 centímetros.

El prototipo realiza un centrado y un acercamiento final que termina cuando el brazo dispensador está a una distancia menor a 2,5 centímetros. El programa extiende el brazo dispensador dentro del orificio, ejecutándose el movimiento en el eje z.

El prototipo continúa con la dispensación del fluido previamente seleccionado.

Una vez finalizada la dispensación, el prototipo realiza una acción de alejamiento de 7 centímetros y despliega interfaces con mensajes. Finalmente, el programa evalúa la distancia del auto al sensor ultrasónico. Si la distancia se incrementa de 7 centímetros a una distancia mayor de 25 centímetros por abandono del auto, los actuadores regresan a su posición y se reinicia el proceso.

El programa proporciona vías para el abandono del proceso, digitando la letra “q” o digitando el comando **CTRL c** en la terminal. El ingreso de los datos como tipo de fluido, cantidad y la orden de abandono, se realiza a través de un teclado.

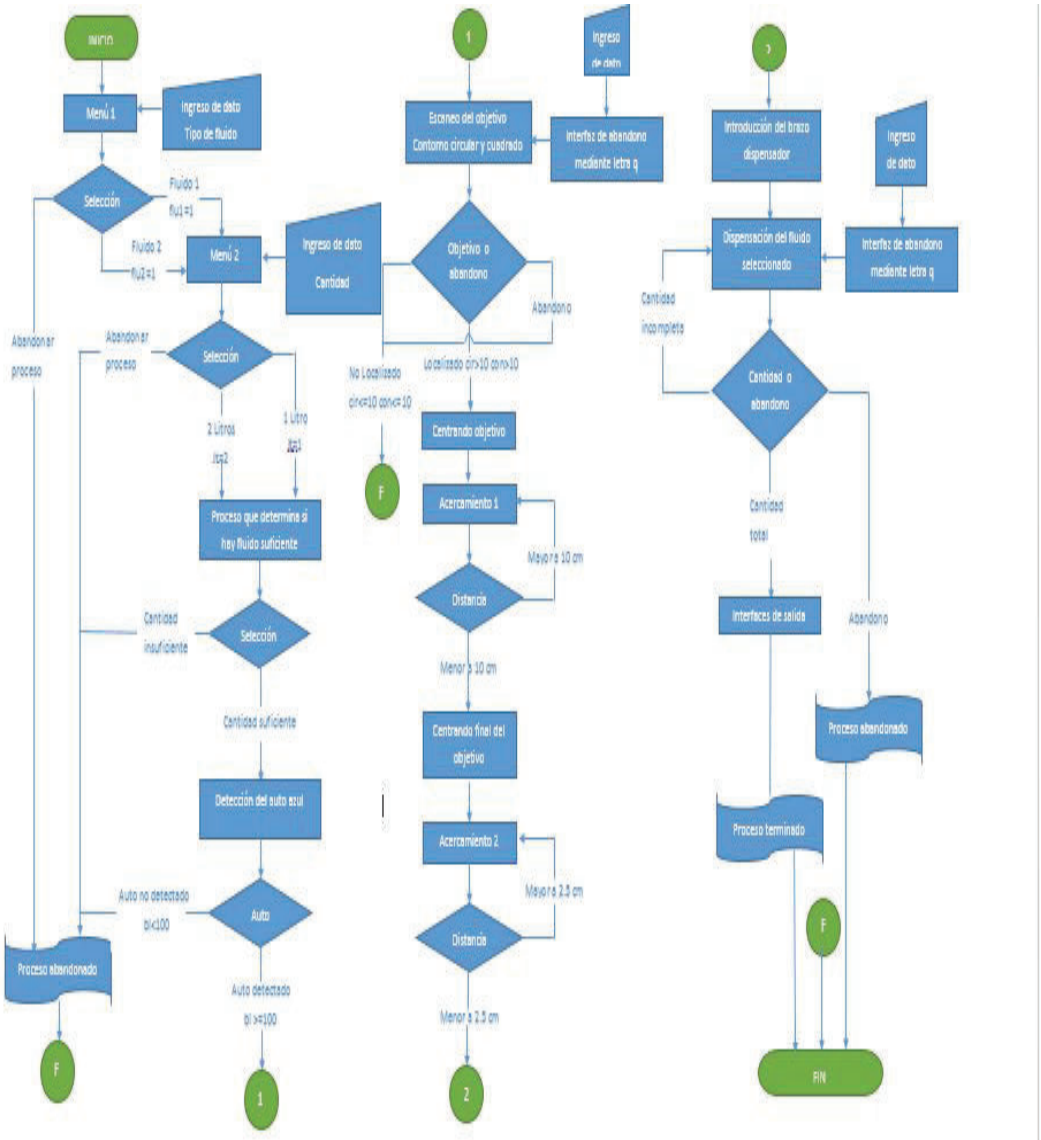

Figura 3. 41.Diagrama de flujo del proceso del prototipo dispensador de fluidos.

- **Estructuras**

Para el proyecto se han diseñado dos estructuras cuyas formas se presentan en las **Figuras 3.42 y 3.43**. Sus dimensiones se muestran en el **Anexo C** (Láminas C1 y C2 respectivamente)

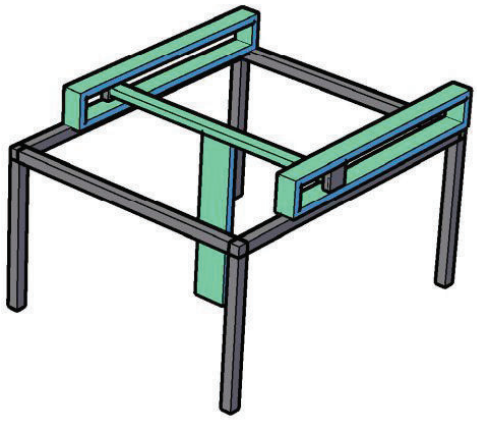

Figura 3. 42. Estructura de soporte de rieles móviles.

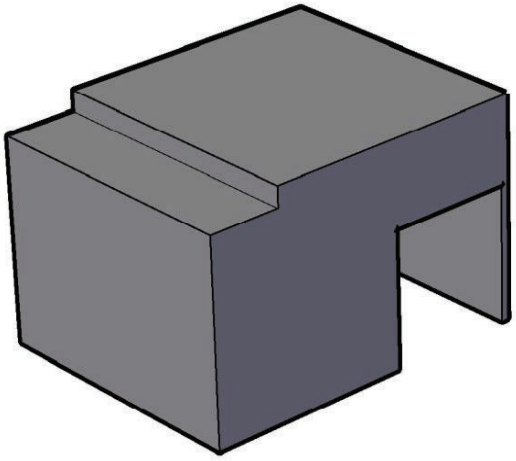

Figura 3. 43. Estructura de cubierta.

3.3 Montaje del Sistema de dispensador automático

- **Montaje de las estructuras**

Para el montaje físico del prototipo se ha seguido una serie de procesos y subprocesos guiados por el diagrama de flujo que se muestra en la **Figura 3.44**.

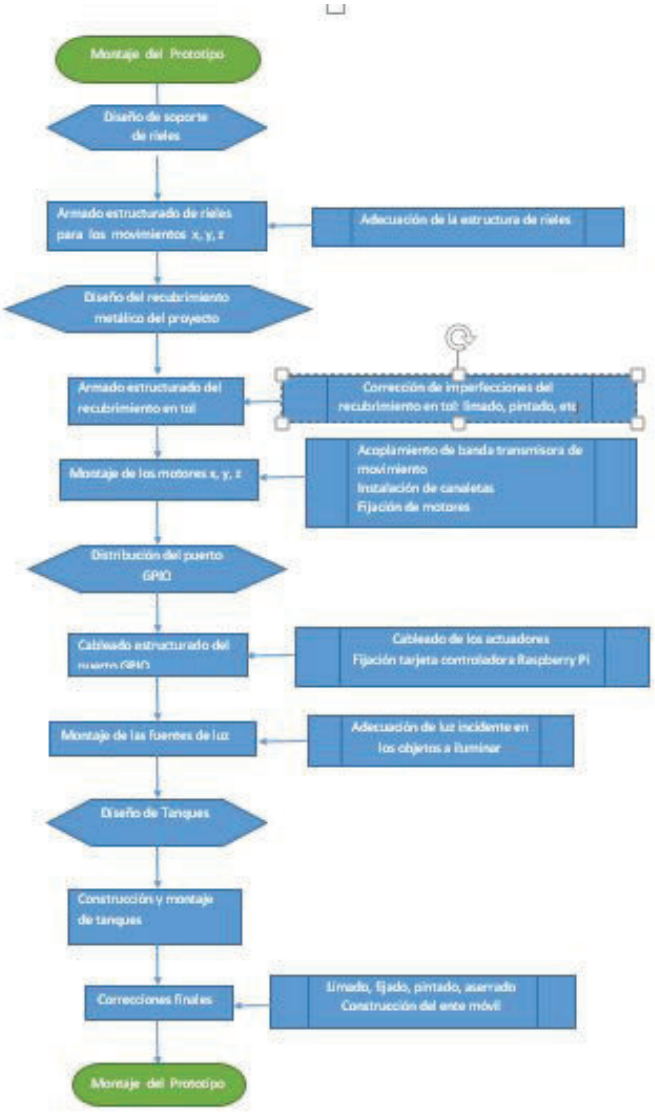

Figura 3. 44. Diagrama de flujo montaje físico del proyecto.

3.4 Pruebas de campo del prototipo

- **Prueba 1: Dispensado de fluido 1, 1 litro**

Objetivo: Mediante el prototipo dispensador de fluido surtir en el ente móvil un litro de agua del tanque 1 y estimar el tiempo de operación.

El proceso se ha iniciado a 11:55:3 como se puede apreciar en las **Figuras 3.45 y 3.46**, que muestran el menú 1 y menú 2, respectivamente.

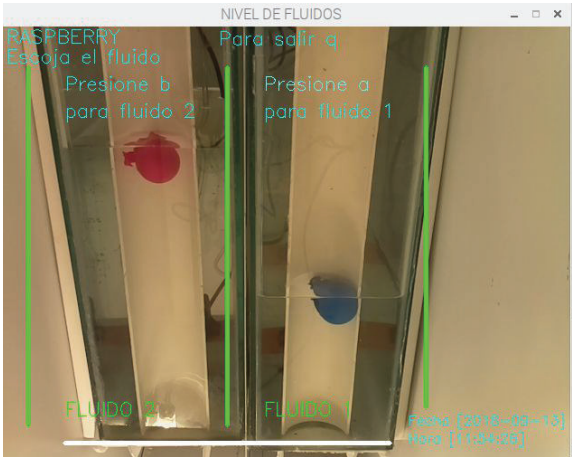

Figura 3. 45. Menú 1 prueba 1.

Figura 3. 46. Menú 2 prueba 1.

La **Figura 3.47** muestra que el tanque 1 tiene el nivel de fluido necesario para proveer de un litro del mismo al ente móvil. Además, el sistema de rieles que dispensa se encuentra en la posición inicial.

Figura 3. 47. Inicio del proceso y posición inicial del sistema de rieles. Prueba 1

Como se muestra en la **Figura 3.48**, el módulo comienza reconociendo el ente móvil presente en la instalación. Se ha diseñado el sensor que reconoce el color de la carcasa y las dimensiones de la parte frontal del auto o ente móvil.

Figura 3. 48. Detector de ente móvil. Prueba 1.

La **Figura 3.49** muestra que el prototipo detecta el posible objetivo reconociendo círculos de cierto radio, de manera que otros posibles de menor radio no sean detectados por el sistema.

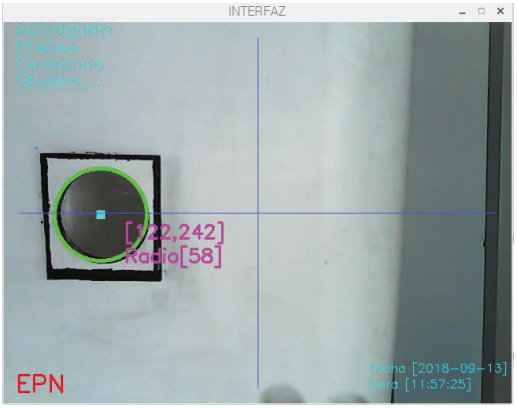

Figura 3. 49. Detección posible objetivo. Prueba 1.

Una vez encontrado el objetivo por medio del sensor de circunferencia y cuadrados (**Figura 3.50)**, el sistema comienza con el proceso de centrado. **Figura 3.51.**

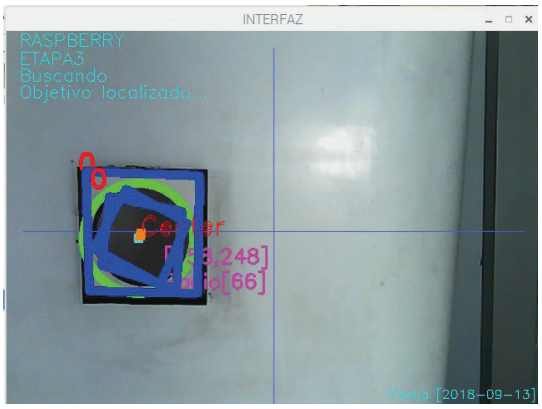

Figura 3. 50. Detección del objetivo. Prueba 1.

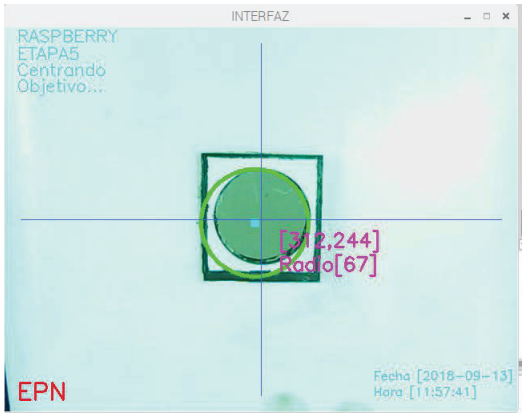

Figura 3. 51. Centrado del objetivo. Prueba 1.

En el proceso de centrado actúa el sensor ultrasónico, en conjunto con el sensor de circunferencias. El estado del sensor ultrasónico se visualiza en una terminal como se muestra en la **Figura 3.52.**

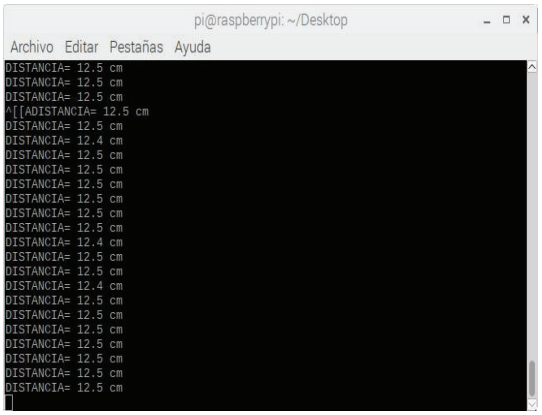

Figura 3. 52. Resultados del sensor ultrasónico. Prueba 1.

Se realiza un segundo centrado y acercamiento final como muestra la **Figura 3.53.**

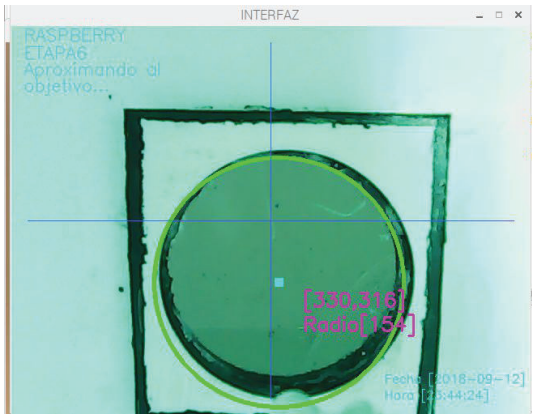

Figura 3. 53. Acercamiento y segundo centrado final. Prueba 1.

De este modo, la manguera dispensadora se ha introducido en el carro. Esto se puede visualizar en la **Figura 3.54**.

Figura 3. 54. Introducción del brazo móvil. Prueba 1.

En esta parte del proceso entra en acción el sensor de nivel del tanque desde el nivel 1 hasta el nivel 0. La **Figura 3.55** muestra el flotador en el nivel superior de descarga.

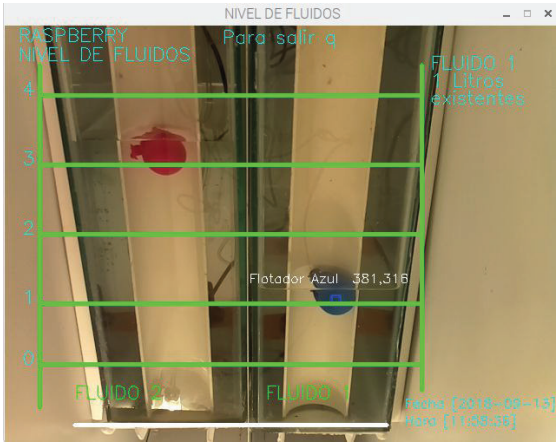

Figura 3. 55. Interfaz de fluidos de nivel superior. Prueba 1.

Se puede visualizar la dispensación del fluido en el auto. **Figura 3.56.**

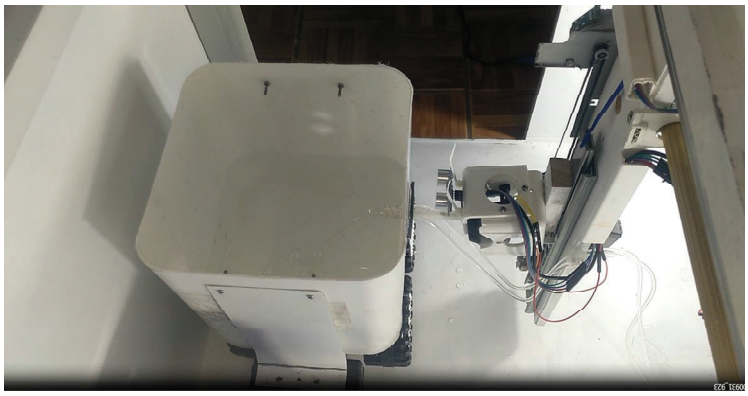

Figura 3. 56. Dispensación de fluido. Prueba 1.

En esta parte del proceso, la **Figura 3.57** muestra que el proceso ha finalizado a las 12:1:41. Tomando en cuenta el tiempo inicial 11:55:3, se ha estimado un tiempo de operación para dispensar un litro de agua del tanque 1 de 5 minutos con 38 segundos.

Figura 3. 57. Interfaz de fluidos nivel inferior. Prueba 1.

Una vez terminado el proceso, entra el sensor ultrasónico en acción evaluando la presencia del ente móvil. Cuando el ente móvil abandona la instalación, el proceso comienza de nuevo.

En la **Figura 3.58** se puede apreciar la finalización del proceso.

Figura 3. 58. Interfaz de objetivo, final de proceso. Prueba 1.

Se realizó varias pruebas adicionales con la misma cantidad de fluido para los 2 taques de donde se obtuvieron varios tiempos de operación y se presentan en la tabla 7.

- **Prueba 2: Dispensado de fluido 2, 2 litros**

Objetivo: Mediante el prototipo dispensador de fluido surtir en el ente móvil dos litros de agua del tanque 2 y medir el tiempo de operación.

La **Figura 3.59** muestra el inicio del proceso.

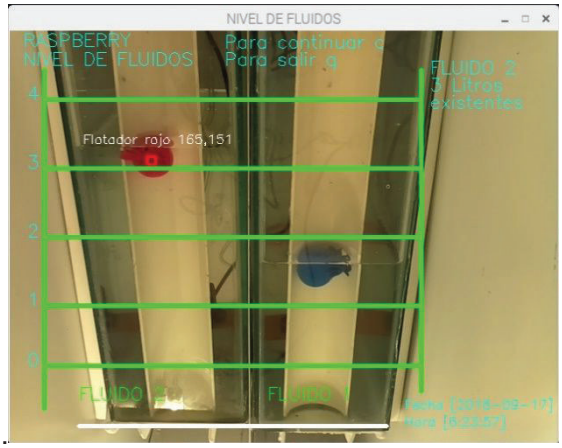

Figura 3. 59. Nivel inicial. Prueba 2.

En esta parte, la interfaz que muestra la **Figura 3.60** indica que el proceso ha finalizado a las 6:22:45. Tomando en cuenta el tiempo inicial 6:32:2, se ha determinado un tiempo de operación para dispensar dos litros de agua del tanque 2 de 9 minutos con 17 segundos.

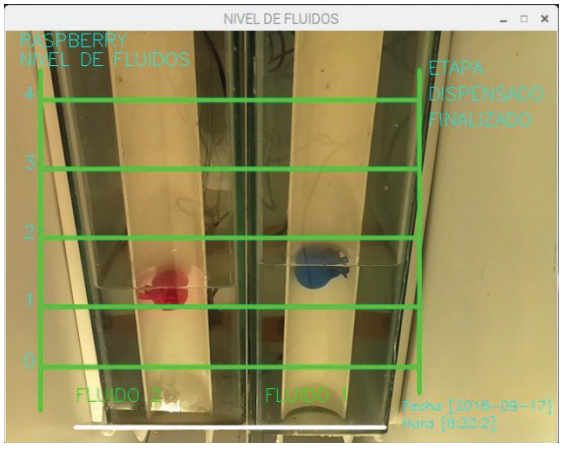

Figura 3. 60. Nivel final. Prueba 2.

Se realizó varias pruebas adicionales con la misma cantidad de fluido para los 2 tanques de donde se obtuvieron varios tiempos de operación y se presentan en la tabla 8.

3.5 Evaluación de Resultados

- **Evaluación de tiempo de operación**

Esta evaluación está orientada a hacer un análisis estadístico por medio de una media aritmética del tiempo aproximado de dispensación para 1 litro y 2 litros de fluido en este módulo.

Para la evaluación de resultados se ha tomado en cuenta los tiempos de ejecución de los procesos de las pruebas de campo.

Datos para 1 litro

Tabla 7. Datos de pruebas para surtir un litro.

Pruebas	Tanque 1(h:m:s)	Tanque 2(h:m:s)	Tiempo estimado
1	0:05:38	0:05:13	0:05:26
2	0:05:47	0:05:25	0:05:36
3	0:05:56	0:05:37	0:05:46
4	0:05:05	0:05:19	0:05:12
5	0:05:14	0:05:01	0:05:07
6	0:05:23	0:05:13	0:05:18
7	0:05:32	0:05:25	0:05:28
8	0:05:41	0:05:37	0:05:39
	Tiempo promedio de operación		0:05:27

Datos para 2 litros.

Tabla 8. Datos de pruebas para surtir dos litros.

Pruebas	Tanque 1(h:m:s)	Tanque 2(h:m:s)	Tiempo estimado
1	0:09:24	0:09:10	0:09:17
2	0:09:17	0:10:25	0:09:51
3	0:10:16	0:10:37	0:10:27
4	0:10:15	0:10:19	0:10:17
5	0:09:54	0:10:01	0:09:57
6	0:09:43	0:10:13	0:09:58
7	0:10:12	0:10:25	0:10:18
8	0:10:11	0:10:37	0:10:24
	Tiempo promedio de operación		0:10:04

Para dispensar 1 litro de un fluido, se requiere un tiempo de ejecución de aproximadamente 5 minutos con 27 segundos y para dispensar dos litros de fluido toma un tiempo de ejecución aproximado de 10 minutos con 4 segundos.

- **Evaluación eléctrica**

En la tabla 9 se presenta la corriente que consume los componentes eléctricos del sistema y se hace una comparación con la fuente de 12 voltios que alimenta cada uno.

Tabla 9. Evaluación eléctrica.

Equipo eléctrico	Cantidad	Corriente	Fuente
Motor KS	3	2.4 (A)	Durante el proceso actúan como mucho 2 motores KS a la ves siendo así la cantidad de consumo máximo de 4,8 amperios de una **fuente 1** de 5 amperios
Motor mitzuki	1	1,6(A)	
Motor DC	1	1(A)	
Bombas DC	2	1,5(A)	
Rapsberry PI	1	1.8(A)	La Raspberry pi controlando los 4 drivers L298N, la cámara web o la pi Camera consume consume 1.8 amperios de una **fuente 2** de 2.5 amperios
Pantalla landzo	1	1 (A)	La pantalla Landzo de 5 pulgadas consume 1.2 amperios de una **fuente 3** de 1.5 amperios

- **Evaluación del algoritmo**

En la Tabla 10 se presenta presenta la prueba de calidad del algoritmo con respecto a las características físicas y de interfaz del prototipo dispensador de fluidos. [25]

Tabla 10. Evaluación del algoritmo.

Pruebas del sistema	Atributos	Validación
Funcionalidad	Características y capacidades del programa	✓
	Generalidad de las funciones	✓
	Seguridad de sistemas	✓
Facilidad de uso	Factores Humanos	✓
	Factores estéticos	✓
	Consistencia de la interfaz	✓
Confiabilidad	Tiempo medio de fallos	✓
	Exactitud de las salidas	✓
	Capacidad de recuperación ante fallas	✓
Rendimiento	Velocidad del procesamiento	✓
	Tiempo de respuesta	✓
	Eficacia	✓
Capacidad de soporte	Adaptabilidad	✓
	Capacidad de pruebas	✓
	Capacidad de configuración	✓

3.6 Mejoras al diseño

Para esta sección se ha hecho un estudio de los posibles materiales y/o equipos que pudieron ser añadidos y que podrían ser reemplazados en el diseño físico del proyecto y que no pudieron ser incluidos por causas como el precio, viabilidad regional de obtención o tamaño pero son una opción de inclusión en posibles proyectos y estudios con módulos Raspberry Pi.

- **Driver A4988**

A diferencia del driver l298n, que solo es un módulo con 2 puentes h, este driver fue especialmente construido para manejar motores a paso y por tanto posee una electrónica más compleja y son altamente precisos y eficaces pero delicados. [26]

En la **Figura 3.61** se muestra la distribución de los pines del driver A4988.

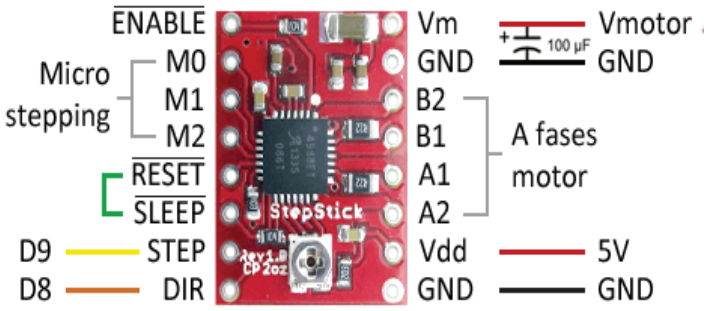

Figura 3. 61. Driver A4988.[26]

Para manejar este driver solo es necesario 2 salidas digitales, una para controlar el avance del motor y otra para indicar el sentido de giro. Además, se puede incrementar su apreciación de hasta 1/16 de paso de motor. Para su calibración, cuenta con un potenciómetro (R_s) que debe ser ajustado dependiendo de un voltaje de referencia (V_{ref}) y las fórmulas establecidas.

$$I_{max} = v_{ref}/(8*R_s) \quad \text{(Ec.3)}$$

$$V_{ref} = I_{max}*8 \quad \text{(Ec.4)}$$

Disponen de protecciones contra sobre intensidad, cortocircuito, sobretensión y sobre temperatura. La corriente máxima es de 4(A). La corriente nominal es de 2(A). Voltaje máximo 35 (V). Rs típico 0.05, 0.1 o 0.2 Ω.

- **Módulo mesa tornillo THS.**

Este kit de materiales es exclusivamente hecho para proyectos de robótica y es utilizado para permitir el movimiento del mismo. [26]

En la **Figura 3.62** se muestra los componentes de este kit.

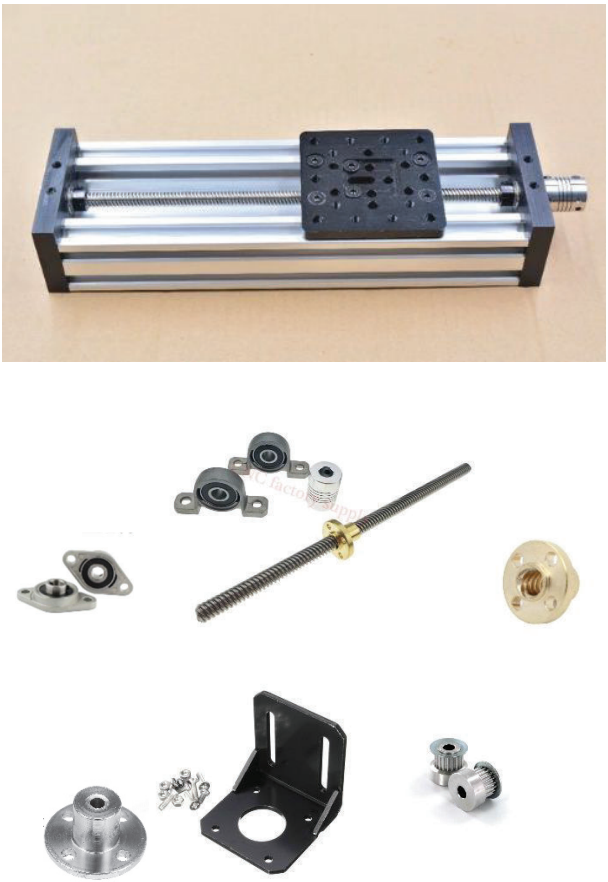

Figura 3. 62. Kit módulo mesa tornillo THSL. [26]

Este módulo tiene un juego de varillas cilíndricas, además de soportes de plomo, rodamientos, cojinetes, poleas y acople para motores. Proporciona una gran estabilidad y precisión de movimiento. Su costo es elevado y sus dimensiones están estandarizadas.

4. CONCLUSIONES Y RECOMENDACIONES.

4.1 Conclusiones

La construcción de un prototipo de dispensador automático de fluidos que utiliza visión artificial demuestra que es una alternativa viable y referente, enfocado en el desarrollo de la automatización.

Durante el desarrollo del proyecto fue posible la detección y seguimiento de objetos, demostrando así que es factible detectar y ubicar un objetivo en el espacio tridimensional en base a sus rasgos de interés y la consecuente acción de sus actuadores.

El algoritmo que controla el proyecto fue desarrollado de manera modular, por etapas con la finalidad de que sus partes puedan ser reutilizables. Además que con esto se tiene la posibilidad de que de ser necesario se pueda mejorar y modificar la estructura del mismo sin que este se vea afectado de manera significativa.

La correcta iluminación al objeto de interés resultó un aspecto de importancia para el sistema de visión artificial puesto que una correcta iluminación ayuda a clarificar las características más relevantes y proporciona imágenes de calidad para ser procesadas y analizadas.

Teniendo en cuenta las necesidades y limitaciones del desplazamiento físico de los actuadores para el movimiento x, y, z durante cada parte del proceso, se ha utilizado un contador para realizar el control de cada motor a pasos, incluido en sus respectivos algoritmos de control. De esta manera, dichos contadores sirven como sensores de fin de carrera; además, aportan con la información de posicionamiento que ayuda a que los actuadores regresen a sus posiciones iniciales.

De acuerdo con el sistema realizado, el módulo Raspberry Pi ha demostrado ser un equipo con una alta adaptabilidad y versatilidad con componentes externos ya que ha permitido realizar distintas tareas que se proliferan se en el campo del control y de la robótica.

Los movimientos rápidos en un sistema de visión artificial tienden a distorsionar y desenfocar los fotogramas o imágenes debido a que la velocidad del sistema de visón artificial depende de la velocidad característica de la cámara y de la velocidad del procesador.

4.2 Recomendaciones

De acuerdo al proyecto realizado los sensores que trabajan con el ultrasónico y con visión artificial están anclados cada uno a su propia velocidad de respuesta, por esto no pueden trabajar de forma eficiente si estos pretenden realizar una tarea conjunta al mismo tiempo, por lo tanto se recomienda evitar esta configuración.

Para cada sensor de color, el objetivo a detectar debe estar en un entorno físico que tenga poca o nula presencia de dicho color para que se desarrolle el sistema, además se debe pre seleccionar los objetivos tomando en cuenta el tamaño de la presencia del color y forma.

Para evitar errores por ruidos y desenfoque en la captura de las imágenes o fotogramas, es de vital importancia un conocimiento básico de los tipos de iluminación y el grado de reflexión de luz de los materiales que están compuestos los objetos a analizar.

En el diseño y construcción de sensores de detección de objetos por medio de visión artificial, se debe tomar en cuenta que el error se incrementa cuanto más pequeño sea el objetivo, y se reduce si el objetivo a analizar tiene presencia relevante en el área de la imagen o fotograma.

Para sistemas que utilicen visón artificial se recomienda que la velocidad del objeto en proceso sea moderada y acorde a la velocidad de la cámara para evitar pérdida de datos, desenfoques y sombras en las imágenes o fotogramas capturadas.

Se debe tener extrema precaución al manejar los puertos GPIO de los controladores Raspberry Pi 3 B. En el momento del encendido, se ejecuta el puerto 7 con 1 lógico de modo que al estar conectado a un amplificador y consecuentemente un actuador, este entra en operación pudiendo dañar el equipo.

Las funciones más complejas de Open CV, dependiendo de la versión, suelen cambiar ligeramente; por esto se recomienda realizar un estudio de la misma antes de diseñar algoritmos ya que esto puede ocasionar errores de compilación.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] Teoría de las colas, Víctor Manuel Quesada & Bargüen Martinez Ferreira, Noviembre 10,2014. [En línea]. Available: http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/garduno_a_f/capitulo2.pdf . [Último acceso: 06 2017].

[2] Revista internacional De Ciencia y Tecnologia, Estudio del Sistema Automático de Abastecimiento de Vehículos con Robótica Brazo controlado mediante PLC,Abril 9, 2016. [En línea]. Available: http://www.sersc.org/journals/IJAST/vol89/4.pdf . [Último acceso: 06 2017].

[3] Aplicación práctica de la visión artificial en el control de procesos industriales, Secretaria de estado de Educación y Formación Profesional, Abril 5, 2016. [En línea]. Available: http://visionartificial.fpcat.cat/wp-content/uploads/UD_1_didac_Conceptos_previos.pdf. [Último acceso: 06 2017].

[4] Fundamentos de la Visión Artificial, Prof. Dr. Francisco Gómez Rodríguez, Marzo 03,2016, Departamento de Arquitectura y tecnología, Universidad de Sevilla. [En línea]. Available: http://www.monografias.com/trabajos18/teoria-colas/teoria-colas.shtml . [Último acceso: 07 2017].

[5] Aplicación práctica de la visión artificial en el control de procesos industriales, Secretaria de estado de Educación y Formación Profesional, Abril 5, 2011. [En línea]. Available: http://visionartificial.fpcat.cat/wp-content/uploads/UD_1_didac_Conceptos_previos.pdf. [Último acceso: 07 2017].

[6] Raspberry Pi Foundation, «Raspberry Pi,» [En línea]. Available: https://www.raspberrypi.org/. [Último acceso: 07 2017].

[7] Packaging Machinery Handbook. John R. Henry. November 17, 2017. The complete guide to automated packaging machinery including packaging line design.

[8] Learn Opne Cv, «Blob Detenction using Open CV,» [En línea]. Available: https://www.learnopencv.com/blob-detection-using-opencv-python-c/ [Último acceso: 17 2015]. [Último acceso: 07 2017].

[9] Open Source Computer Vision, « Object Detection» [En línea]. Available: https://translate.google.com.ec/translate?hl=es&sl=en&tl=es&u=https%3A%2F%2Fdocs.opencv.org%2F3.1.0%2Fdd%2Fd49%2Ftutorial_py_contour_features.html&anno=2&sandbox=1

[10] Open CV 3.0.0, «Smoothing Images,» [En línea]. Available: https://docs.opencv.org/3.0-beta/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html#smoothing.

[11] Technology in action, Learn C,C Motors,PWM. Pete Membrey and David Hows. December 21 ,2015.

[12] Technology in action, Learn Raspberry Pi Programming with Python .Wolfram Donat. April 26 ,2016.

[13] Technology in action, Beginning Sensors Networks with Arduino and Raspberry Pi, Charles Bell.May 30 ,2015.

[14] Raspberry Pi Foundation, «Raspberry Pi Products,» [En línea]. Available: https://www.raspberrypi.org/products/. [Último acceso: 10 2017].

[15] Learn Opencv, « Install OpenCV on Ubuntu » [En línea]. Available: https://www.learnopencv.com/install-opencv-4-on-ubuntu-18-04/ [Último acceso: 07 2017].

[16] Pyimagesearch , « Taking screenshots with OpenCV and Python» [En línea]. Available: https://www.pyimagesearch.com/2018/01/01/taking-screenshots-with-opencv-and-python/. [Último acceso: 07 2017].

[17] Open CV 3.0.0, « More Morphology Transformations,» [En línea]. Available: https://docs.opencv.org/3.0-beta/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html#hough-circle. /. [Último acceso: 08 2017].

[18] Open CV 3.0.0, « Hough Circle Transform» [En línea]. Available: https://docs.opencv.org/3.0-beta/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html#canny-detector. /. [Último acceso: 08 2017].

[19] Open CV 3.0.0, « Finding contours in your image » [En línea]. Available: https://docs.opencv.org/3.0-beta/doc/tutorials/imgproc/shapedescriptors/find_contours/find_contours.html#find-contours . /. [Último acceso: 08 2017].

[20] Open CV 3.0.0, « Image Moments» [En línea]. Available: https://docs.opencv.org/3.0-beta/doc/tutorials/imgproc/shapedescriptors/moments/moments.html#moments . /. [Último acceso: 08 2017].

[21] Open Source Computer Vision, « Object Detection» [En línea]. Available: https://translate.google.com.ec/translate?hl=es&sl=en&tl=es&u=https%3A%2F%2Fdocs.opencv.org%2F3.1.0%2Fdd%2Fd49%2Ftutorial_py_contour_features.html&anno=2&sandbox=1

[22] Open Source Computer Vision, « Changing Colorspaces» [En línea]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html /. [Último acceso: 09 2017].

[23] Broadcom Corporation, «VideoCore IV Architecture Guide,» 2016. [En línea]. Available: https://docs.broadcom.com/docs/12358545. /. [Último acceso: 09 2017].

[24] EcuRed Conocimiento para todos. Web Cam. 22 de enero de 2017. [En línea]. Available: https://www.ecured.cu/C%C3%A1mara_Web /. [Último acceso: 09 2017].

[25] Analizando sistemas de calidad/ pruebas/ Desempeño de un Sistema. El Modelo FURPS+ Available: https://analisistem.wordpress.com/2010/02/23/el-modelo-furps/ . /. [Último acceso: 09 2017].

[26] Código 21 tecnologías alternativas, Material de robotica 3D y chromebooks.Guia completa de materiales de robotica para proyectos. [En línea]. Available: http://codigo21.educacion.navarra.es/prestamo-de-material-de-robotica/descripcion-del-material-de-robotica-disponible-para-prestamos/ [Último acceso: 09 2018]

[27] Raspberry Noods « Instalar el sistema operativo en la Raspberry Pi» [En línea]. Available: https://raspberryparatorpes.net/instalacion/noobs-paso-a-paso-instalar-el-sistema-operativo-en-la-raspberry-pi/ [Último acceso: 09 2018].

[28] Raspberry Pi Foundation, «Installing operating system images,» 2018. [En línea]. Available: https://docs.opencv.org/3.1.0/d4/da8/group__imgcodecs.htmlhttps://www.raspberrypi.org/documentation/installation/installing-images/README.md.

[29] Raspberry Pi Foundation, «RASPBIAN,» [En línea]. Available: https://www.raspberrypi.org/documentation/raspbian/. [Último acceso: 09 2017].

[30] B. Corporation, «BCM2835 ARM Peripherals,» 2014. [En línea]. Available: https://www.raspberrypi.org/documentation/hardw are/raspberrypi/bcm2835/BCM2835-ARM- Peripherals.pdf. [Último acceso: 10 2017].

[31] GL4R3,Utramandiga, Ultrasonic Sensor,Arduino & Rspberry pi ,September 27,2015. [En línea]. Available: https://projects.robotics.ec/en/projects2896/Ultrasonic. /. [Último acceso: 10 2017].

[32] Amplificadores, Driver L289N. Guia Circutos de Potencia. November 1 ,2015. [En línea]. Available: https://www.inventable.eu/2010/12/08/driver-para-motor-paso-a-paso//. [Último acceso: 10 2017].

ANEXOS

ANEXO A: TEORÍA COMPLEMENTARIA

ANEXO B: COSTOS DEL PROYECTO

ANEXO C: DIMENSIONES DE LAS ESTRUCTURAS

ANEXO D: CÓDIGO DEL PROTOTIPO

ANEXO A: TEORÍA COMPLEMENTARIA

ANEXO A1. INSTALACIÓN DE RASPBIAN NOODS

Noods es una aplicación para Raspberry Pi que se instala en una tarjeta de memoria y permite arrancar la Rasperry pi para instalar en la misma un sistema operativo por medio de una interfaz sencilla. [27]

Descargar Noods.

Esta aplicación se encuentra disponible para todo el público en la página oficial de Raspberry pi. Se muestra la página oficial de Rasberry en la **Figura A. 1.**

Web: https://www.raspberrypi.org/downloads/noobs/

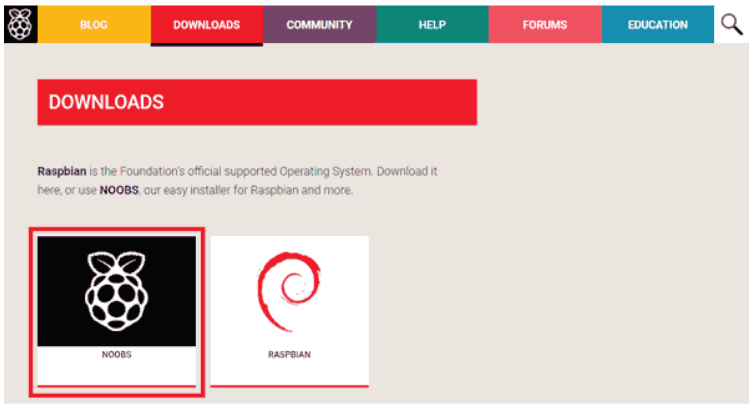

Figura A. 1.Página oficial de Raspbian [27].

Existen 2 versiones de esta aplicación disponibles como muestra la **Figura A. 2**, una que viene con el sistema operativo y otra que no. Esta última hace la instalación descargando el sistema operativo directamente desde internet.

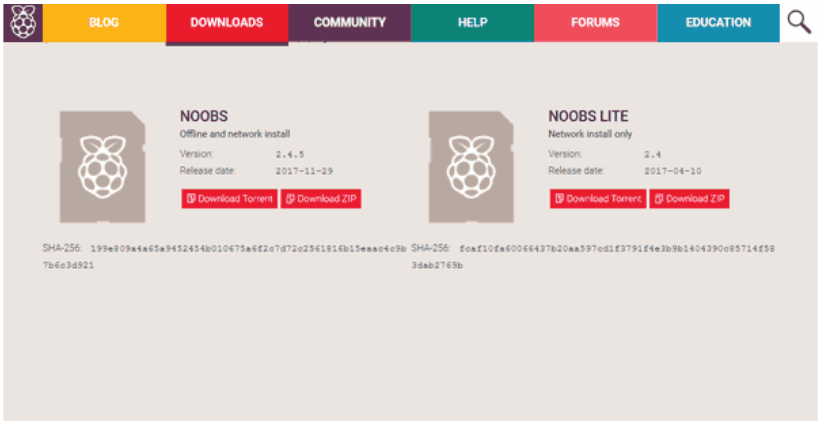

Figura A. 2.Página de descargas de Raspberry [27].

La instalación de nodos únicamente consiste en descomprimir el contenido del fichero zip que se ha descargado en un micro sd o tarjeta de memoria, además se debe formatear adecuadamente la tarjeta de memoria de manera que se pueda utilizar su recurso de manera eficiente.

Una vez instalado noods, se coloca la tarjeta de memoria en la Raspberry y se enciende. Es necesario que la raspberry esté conectada a una pantalla monitor y contar con un

mouse y teclado USB. En esta instancia, aparecerá un menú con distintos sistemas operativos **(Figura A.3).** Aquí se selecciona instalar Raspbian.

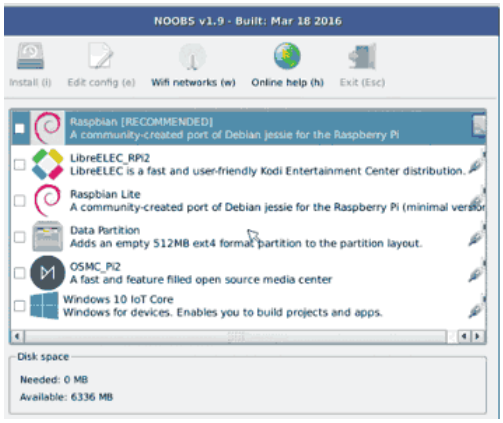

Figura A. 3.Interfaz Noods [27].

Se espera que la aplicación noods instale el sistema operativo. La ventana del proceso de instalación se presenta en la **Figura A. 4.**

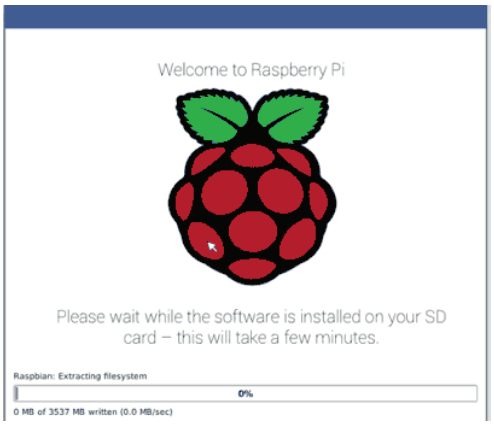

Figura A. 4.Interfaz de instalación noods [27].

Una vez finalizado, la Raspberry Pi se reiniciará y se ejecutará con el sistema operativo instalado. Se muestra el escritorio de Raspbian en la **Figura A. 5.**

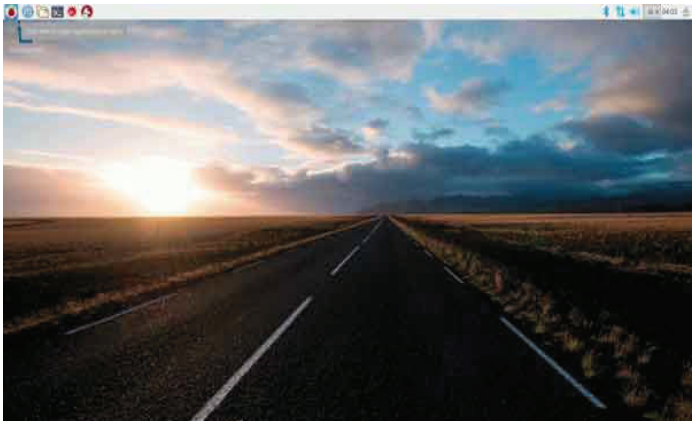

Figura A. 5.Escritorio de Raspbian [27].

ANEXO A2. INSTALACIÓN DE EXTENSIONES O LIBRERÍAS NECESARIAS EN RASPBIAN

Antes de trabajar con el módulo Raspberry Pi, se lo debe configurar y preparar por medio de la instalación de las librerías necesarias para lo se abre una terminal **(Figura A. 6)** y se escribe los siguientes comandos. [28]

- *sudo apt-get upgrade*: Descarga las librerías o extensiones de Raspbian o Ubuntu que han sido actualizadas.
- *sudo apt-get update :* Actualiza las librerías o extensiones de Raspbian o Ubuntu.
- *sudo apt-get install Python-numpy*: Es una extensión de Python que le agrega soporte para vectores y matrices usada en una matemática de alto nivel.
- *sudo apt-get install Python-scipy:* Es una extensión de Python que sirve como soporte para procesamiento de señales e imágenes.
- *sudo apt-get install libopencv-**: Es la librería o extensión de visión por computadora que utiliza lenguaje de programación Python, -* hace que se instale todas sus extensiones.
- *sudo apt-get install Scrot:* Esta librería o extensión sirve para poder tomar capturas de pantalla de una raspberry pi a través de una terminal.
- *sudo apt-get install Python-picamera*: Si se desea trabajar con rapsberry pi camera se debe instalar su respectiva librería para el control de la misma. Esto no es necesario para la Web Cam.

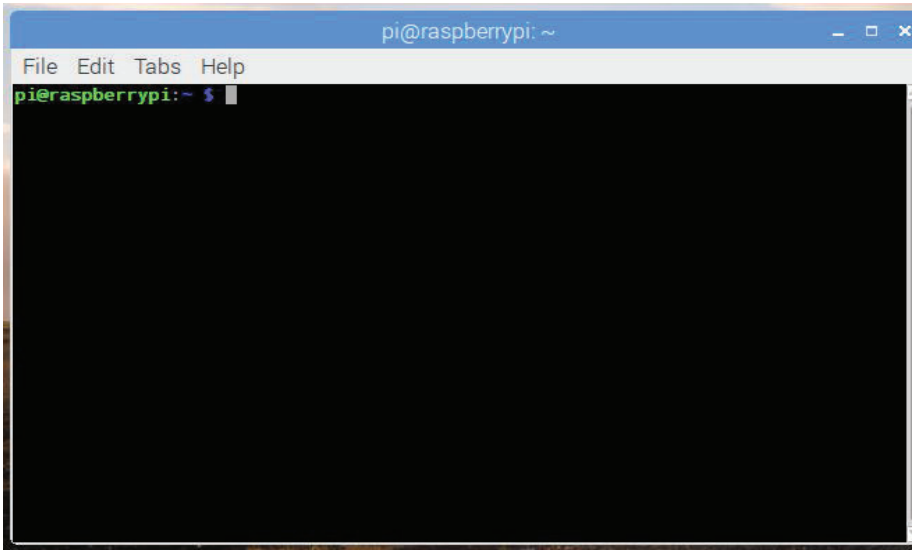

Figura A. 6.Terminal Raspbian [28].

ANEXO A3. RASPBERRY PI

En la actualidad existen 4 versiones básicas de Raspberry, con un total de 6 modelos disponibles.

RPi 1: Esta es la versión básica, basada en el System on Chip BCM2835, dispone de un procesador ARM 11, single-core a 700 MHz. Disponible en 2 modelos A+ y B+ con pequeñas diferencias entre ambos.

RPi 2: Esta versión salió en el año 2014, está basada en el SoC BCM2836, cuya principal diferencia es el procesador que cambia a un ARM quad-core Cortex A7 a 900 MHz. Disponible solo en el modelo B.

RPi 3: Tercera generación de Raspberry pi, la única diferencia con su predecesor es el cambio del SoC al modelo BCM2837, es decir la sustitución del procesador por un ARMv8 quad-core de 64-bit a 1.2GHz. [29]

ARQUITECTURA ARM

La arquitectura ARM o también llamada RISC (ordenador con conjunto reducido de instrucciones) es una arquitectura computacional para el diseño de microprocesadores.

Las instrucciones en este tipo de arquitectura son simples y cortas de modo que consumen menos ciclos de reloj, procesan más rápido, consumen menos energía y disipa menos calor que los procesadores con arquitectura x86 (Intel,AMD o Cyrix).

SoC BCM

También llamados sistemas en chip de Broadcom es una tendencia de diseño de microcontroladores establecido por la empresa Broadcom para ARM, donde se busca la integración en un solo chip de varios módulos de un sistema tales como: [30]

- Temporizador.
- Entradas/salidas digitales de propósito genérico, GPIO *(General Purpose Input-Output)*.
- Puerto USB.
- Controlador de acceso directo a memoria, (DMA).
- Maestro y esclavo de bus I2C *(Inter-Integrated Circuit).*
- Maestros y esclavo de bus SPI *(Serial Peripheral Interface).*
- Puertos serie, UART.
- Interfaz para memorias eMMC, SD, SDIO.
- Controlador de interrupciones.
- Interfaz HDMI
- Audio PCM a través de bus I2S *(Integrated Interchip Sound).*
- Módulo para generación de pulsos de anchura variable, PWM.

RAM

O también llamada memoria de acceso aleatorio, es la memoria de un CPU donde residen los programas y datos sobre los cuales se realizan operaciones de lectura y escritura. Es considerada una memoria temporal.

HDMI

Interfaz multimedia de alta definición y hace referencia a la norma de un periférico de conexión que permite trasmitir audio y video.

En la Tabla A se presenta las características de cada versión de Raspberry.

Tabla A. Características de los modelos de Raspberry.[29]

	CM	A+	B+	2B	Zero	3B
SoC (BCMxxxx)	2835	2835	2835	2836	2835	2837
Frecuencia	700MHz	700MHz	700MHz	900MHz	1GHz	1.2GHz
Arquitectura	ARM6	ARM6	ARM6	ARM7	ARM6	ARM8
Núcleos	1	1	1	4	1	4
Memoria RAM	512MB	256MB	512MB	1GB	512MB	1GB
Memoria Flash	4GB	microSD	microSD	microSD	microSD	microSD
GPIO pins	54	26	26	26	26	26
Puertos USB	1	1	4	4	1	4
Ethernet 10/100	☐	☐	☑	☑	☐	☑
WiFi	☐	☐	☐	☐	☐	☑
Bluetooth	☐	☐	☐	☐	☐	☑
HDMI	☑	☑	☑	☑	☑	☑

ANEXO A4: SENSOR ULTRASÓNICO HC-SR04

Como muestra la **Figura A. 7,** este sensor consta de un terminal emisor (echo) que emite un ultrasónico de alta frecuencia imperceptible al oído humano y de un terminal receptor (trigger) que capta tras un determinado tiempo la señal que regresa cuando esta ha chocado contra una superficie. El movimiento del sonido es rectilíneo uniforme ya que depende de la velocidad del sonido en el aire, dato importante a la hora de realizar diseños de control para este instrumento.Tiene un rango de distancias de 3 cm hasta 3 m con una precisión de 3 mm. [31]

Figura A. 7.Sensor ultrasónico hc-sr04 [31].

Fórmula para calcular la distancia.

$$\text{Distancia} = \text{Vsa}/2 * (\text{t2} - \text{t1}) \quad \text{(Ec.5)}$$

V_{sa} = velocidad del sonido en el aire (340 m/s)

t2=tiempo final de la llegada de la señal.

t1= tiempo de salida de la señal.

La fórmula divide para 2 ya que el recorrido evalúa la distancia de ida y de vuelta de la señal.

ANEXO A5. DRIVER L289N

Usado en robótica para el manejo de motores. Este módulo posee 2 puentes o canales h, con lo que permite controlar la velocidad y el sentido de giro de 2 motores DC o un motor a pasos unipolar /bipolar. Se puede apreciar cada una de sus partes en la **Figura A. 8.**

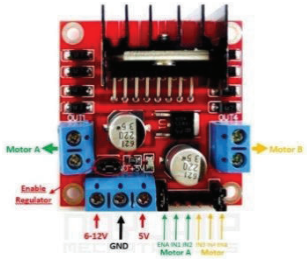

Figura A. 8.Driver L298N [32].

Tiene 2 formas de ser alimentado:

Utilizando una sola fuente conectada al pin 6-12V y con el *Enable Regulator* aclarando que aquí el circuito lógico del controlador está conectado eléctricamente a la fuente que alimenta el motor por medio de un divisor de voltaje. Utilizando dos fuentes, una que va conectada al pin 5V que alimenta al circuito lógico y otra que va conectada al pin 6-12V con la que trabaja el motor. Para esto se retira el *Enable Regulador*. [32]

Los pines IN1, IN2, IN3, IN4 controlan las salidas OUT1, OUT2, OUT3 y OUT4, respectivamente. El pin ENA, sirve para habilitar o deshabilitar las salidas OUT1 y OUT2 y el ENB habilita o deseablita las salidas OUT3 y OUT4.Corriente máxima 4 Amperios. Corriente nominal 2 Amperios. Bajo voltaje de saturación en los transistores de salida. Corte de operación por sobrecalentamiento. Voltaje máximo 46 volts.

ANEXO B. COSTO DEL PROYECTO.

En la Tabla B se presenta los costos del proyecto.

Tabla B. Costos del proyecto.

Elemento	Descripción	Cantidad	Valor unitario	Valor Total
1	Raspberry pi B+	1	$ 65,00	$ 65,00
2	Cámara web genius	1	$ 20,00	$ 20,00
3	Pi camera v2	1	$ 30,00	$ 30,00
4	Estructura metálica	1	$ 180,00	$ 180,00
5	Estructura de aluminio	1	$ 100,00	$ 100,00
6	Ultrasónico	1	$ 5,00	$ 5,00
7	Motor Ks	3	$ 20,00	$ 60,00
8	Tanques de vidrio	2	$ 8,00	$ 16,00
9	Motores Mitsuki	1	$ 5,00	$ 5,00
10	Poleas GT	3	$ 5,00	$ 15,00
11	Pantalla 5 pulgadas.	1	$ 60,00	$ 60,00
12	Driver L298N	5	$ 5,00	$ 25,00
13	Motor Dc	1	$ 5,00	$ 5,00
14	Mini-Bomba Sumergible	2	$ 20,00	$ 40,00
15	Rieles telescópicas	4	$ 4,00	$ 16,00
16	Teclado	1	$ 10,00	$ 10,00
17	Otros	1	$ 50,00	$ 50,00
TOTAL	PROYECTO	1		$ 702,00

ANEXO C: DIMENSIONES DE LAS ESTRUCTURAS

ANEXO D: CÓDIGO DEL PROTOTIPO