FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

IMPLEMENTACIÓN DE UN PROTOTIPO CON RASPBERRY PI PARA EL RECONOCIMIENTO DE COLOR Y PESO MARCADO EN LOS CILINDROS DE GLP PARA UNA EMPRESA ENVASADORA

AUTOR

Renato Wladimir Albuja Rivilla

AÑO

2020

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

IMPLEMENTACIÓN DE UN PROTOTIPO CON RASPBERRY PI PARA EL RECONOCIMIENTO DE COLOR Y PESO MARCADO EN LOS CILINDROS DE GLP PARA UNA EMPRESA ENVASADORA

Trabajo de titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Redes y Telecomunicaciones

Profesor Guía

MSc. Carlos Carrión Betancourt

Autor

Renato Wladimir Albuja Rivilla

Año 2020

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido el trabajo, Implementación de un prototipo con raspberry pi para el reconocimiento de color y peso marcado en los cilindros de glp para una empresa envasadora, a través de reuniones periódicas con el estudiante Albuja Rivilla Renato Wladimir, en el semestre 202010, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación."

Carlos Carrión Betancourt

Magister en Redes y Telecomunicaciones

C.I. 1103738074

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, Implementación de un prototipo con raspberry pi para el reconocimiento de color y peso marcado en los cilindros de glp para una empresa envasadora, de Albuja Rivilla Renato Wladimir, en el semestre 202010, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

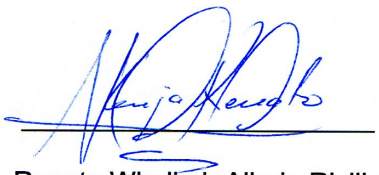
Milton Román Cañizares

Magister en Gerencia de Redes y Telecomunicaciones

C.I. 0502163447

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

"Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes."



Renato Wladimir Albuja Rivilla

C.I.: 2100181557

RESUMEN

La visión artificial es un instrumento futurista que con el avance de la tecnología ha logrado imponerse en el diario vivir y facilitar diferentes actividades que se automatizan permitiendo optimizar recursos a las personas. Es por lo que el proyecto de titulación se basa en la identificación de las características de un cilindro de gas con su color y peso marcado, para esto se utilizará una librería de visión artificial llamada open cv que permitirá procesar las imágenes capturadas y así identificar las características del cilindro. Contiene algoritmos que detectan, comparan y analizan imágenes captadas con una cámara. Para el proyecto se ha propuesto realizar el prototipo con ayuda de un raspberry pi 3 (Raspberry Pi, s.f.) y una cámara pi (Picamara, s.f.), esto permitirá integrar el software y hardware en un mini pc que corre un sistema operativo Raspbian Stretch (Raspian, s.f), se instalará Python como lenguaje de programación y todas las librerías para visión artificial que ofrece open cv. Una vez preparado el ambiente de programación se desarrollará el programa que tomará capturas de los cilindros de gas para luego detectar de forma automática. Para finalizar es importante mencionar que el proyecto servirá como base para dimensionar a gran escala y podrá implementarse en el negocio de GLP cumpliendo todos los requisitos de seguridad. El envasado de GLP es un proceso de alto riesgo que no da lugar a fallas en sus dispositivos que se encuentran dentro de la nave de envasado, cumpliendo normas de seguridad antiexplosivas por lo que no se puede introducir este prototipo en un ambiente real de envasado. El proyecto cumplirá con el objetivo basándose en una simulación de detección fuera de la nave de envasado, ya que se tiene que aprobar normas de seguridad y calidad industrial antes de introducir el proyecto en un ambiente de producción. Para esto se creará la zona de detección con cilindros reales, se trabajará en ello todo el desarrollo del proyecto. Después de obtener todo el programa de detección se tomará fotos de cilindros reales y se realizará las pruebas de detección de color y peso marcado.

ABSTRACT

Artificial vision is a futuristic instrument that, with the advancement of technology, has managed to impose itself on the daily life and facilitate different activities that are automated allowing people to optimize resources. That is why the titling project is based on the identification of the characteristics of a gas cylinder with its color and marked weight, for this an artificial vision library called open cv will be used that will allow to process the captured images and thus identify the cylinder characteristics It contains algorithms that detect, compare and analyze images captured with a camera. For the project it has been proposed to carry out the prototype with the help of a raspberry pi 3 (Raspberry Pi, s.f) and a pi camera (Picamara, s.f), this will allow to integrate the software and hardware into a mini pc that runs a Raspbian Stretch operating system (Raspian, s.f), Python will be installed as programming language and all libraries for artificial vision offered by open cv. Once the programming environment has been prepared, the program will be developed that will take captures of the gas cylinders and then automatically detect them. Finally, it is important to mention that the project will serve as a basis for large-scale sizing and can be implemented in the LPG business, complying with all security requirements. LPG packaging is a high-risk process that does not lead to failures in its devices that are inside the packaging ship, complying with anti-explosive safety standards so this prototype cannot be introduced in a real packaging environment. The project will meet the objective based on a detection simulation outside the packaging warehouse, since industrial safety and quality standards have to be approved before introducing the project into a production environment. For this, the detection zone will be created with real cylinders, the entire project development will be worked on. After obtaining the entire detection program, photos of real cylinders will be taken and color and weight detection tests will be performed.

INDICE

INDICE DE FIGURAS

INTRODUCCIÓN

Una empresa de envaso de GLP (Gas licuado de petróleo) es una entidad donde se realiza la comercialización y envasado de GLP doméstico e industrial. Cada empresa de GLP tiene asignado un color en su cilindro de gas para su identificación, la empresa tiene como medio de abastecimiento una nave de envasado donde se carga de GLP a cilindros de 15 kg y 45 kg, este proceso se realiza con mecanismos hidráulicos ya que dentro de esta nave de envasado no se puede tener mecanismos que ocupen una alimentación mayor a 24 voltios. Por otro lado, se debe tomar en cuenta que, por ser un área de alto riesgo, se debe utilizar cajas de protección de tipo antiexplosivas para cualquier conexión que se tenga dentro de la nave.

Aproximadamente se llenan 42000 cilindros diarios por planta. El proceso de descarga de cilindros lo realizan estibadores, quienes se encargan de contar y verificar el color y peso marcado en el cilindro. Dentro de este procedimiento un supervisor verifica toda la descarga siendo el quien apunta los datos en una guía de remisión para posterior enviar un pedido de envasado.

Cabe recalcar que este proceso requiere de mucha responsabilidad y concentración por parte de los estibadores y supervisor, por tal motivo, si no existe esta predisposición la empresa recibe sanciones que las impone la Dirección de la Agencia de Regulación y Control Hidrocarfurífero (Reglamento Actividades de Comercialización Gas Licuado de Petróleo, 2017), ya que el pasar por alto el color del cilindro de otras marcas y envasarlo sin la debida autorización por parte de la entidad de control es considerada una falta, de igual forma si no se especifica el peso marcado en el cilindro no se llenará de GLP correctamente y por lo tanto se entenderá que no se vende un peso ya marcado, causando así sanciones a la envasadora de GLP.

El proyecto contempla esta problemática como opción para mejorar el proceso de verificación de color y peso marcado en cada cilindro de GLP que es descargado en la nave de envasado. Este proyecto se realizará proponiendo un prototipo con un Raspberry PI 3 ya que es un dispositivo que brinda una amplia forma de integrar mecanismos electrónicos con un desarrollo personalizado, en

este caso se propone un desarrollo en lenguaje Python, el cual permitirá realizar una detección del color y peso marcado de un cilindro de gas, esto se hace por medio de visión artificial con una cámara, logrando así detectar lo antes mencionado.

Python y su librería OpenCV permitirá que el reconocimiento se pueda diseñar de manera correcta. Para detectar el color y peso marcado del cilindro se utilizará una cámara que nos permite captar varias imágenes para luego compararlas con el algoritmo desarrollado y así identificarlo.

El prototipo se usará solamente para la detección del objeto, por esta razón cabe indicar que el proyecto se enfoca en tomar fotos del cilindro de gas, para que se analice con un algoritmo de detección de imágenes el color y el peso marcado, y así validar un buen despacho de los cilindros.

Se encuentra una vulnerabilidad que puede afectar en costos a una empresa de GLP por lo que es muy importante fomentar la innovación para generar más oportunidades de implementación de proyectos tecnológicos que minimizan en este caso sanciones y mejoran el proceso operativo. Este proyecto es considerado en el área de GLP importante ya que cubre varias falencias que las empresas envasadoras actualmente no la tienen implementada y crea una apertura para nuevos proyectos de automatización. Este prototipo podría favorecer tanto a la entidad de control ARCH y a la empresa envasadora en cuestión de trazabilidad. Por lo tanto, en este proyecto como estudiante de la Udla de la carrera de Ing. Redes y Telecomunicaciones, se aporta con esta idea innovadora con miras al futuro.

1. MARCO TEÓRICO

1.1. Análisis General

Hoy en día la tecnología ha permitido crear proyectos de gran impacto, por tal razón es importante mencionar que cada vez la automatización con inteligencia artificial va tomando fuerza en el día a día. Existen maneras de controlar una actividad tan solo con instalar una cámara y un software personalizado que realiza la función de una persona que monitorea la misma.

Es por eso que en este proyecto se ha propuesto juntar varias herramientas tecnológicas que conducen a implementar un prototipo que detectará un cilindro de GLP con su color y peso marcado.

Para desarrollar este prototipo se necesita un Raspberry Pi 3, cámara Pi, adaptador de 5V, cable de red, monitor HDMI.

1.2. Análisis del escenario

La empresa envasadora que concierne al presente proyecto, cuenta con un área donde se realiza el envasado de GPL. Esta dispone de un carrusel donde se colocan los cilindros de gas y pasan por diferentes estaciones hasta llegar al llenado de GLP.

En la descarga de los cilindros se encuentran los estibadores y un supervisor, quienes realizan un conteo y selección del color del cilindro para luego colocarlos en el carrusel de GLP para continuar en el proceso. Es aquí donde el proyecto propone la automatización con un detector de objetos por medio de una cámara. Esta estará ubicada sobre la banda transportadora donde pasan todos los cilindros descargados por el estibador. Tiene que pasar el cilindro de GLP frente a la cámara para la detección del color y peso marcado.

De esta forma se logrará automatizar el proceso de manera precisa con un margen de error menor al del estibador. Se mejorará la recepción de los cilindros y evitará posibles fraudes por parte del personal de la envasadora. Logrando así optimizar tiempos al momento de detectar el color y peso marcado de la manera

más segura y precisa. Esto ayudará a las empresas a generar más desarrollos de automatización en procesos que requieren ser precisos.

A continuación, la figura 1 indica el proceso actual de la empresa envasadora.

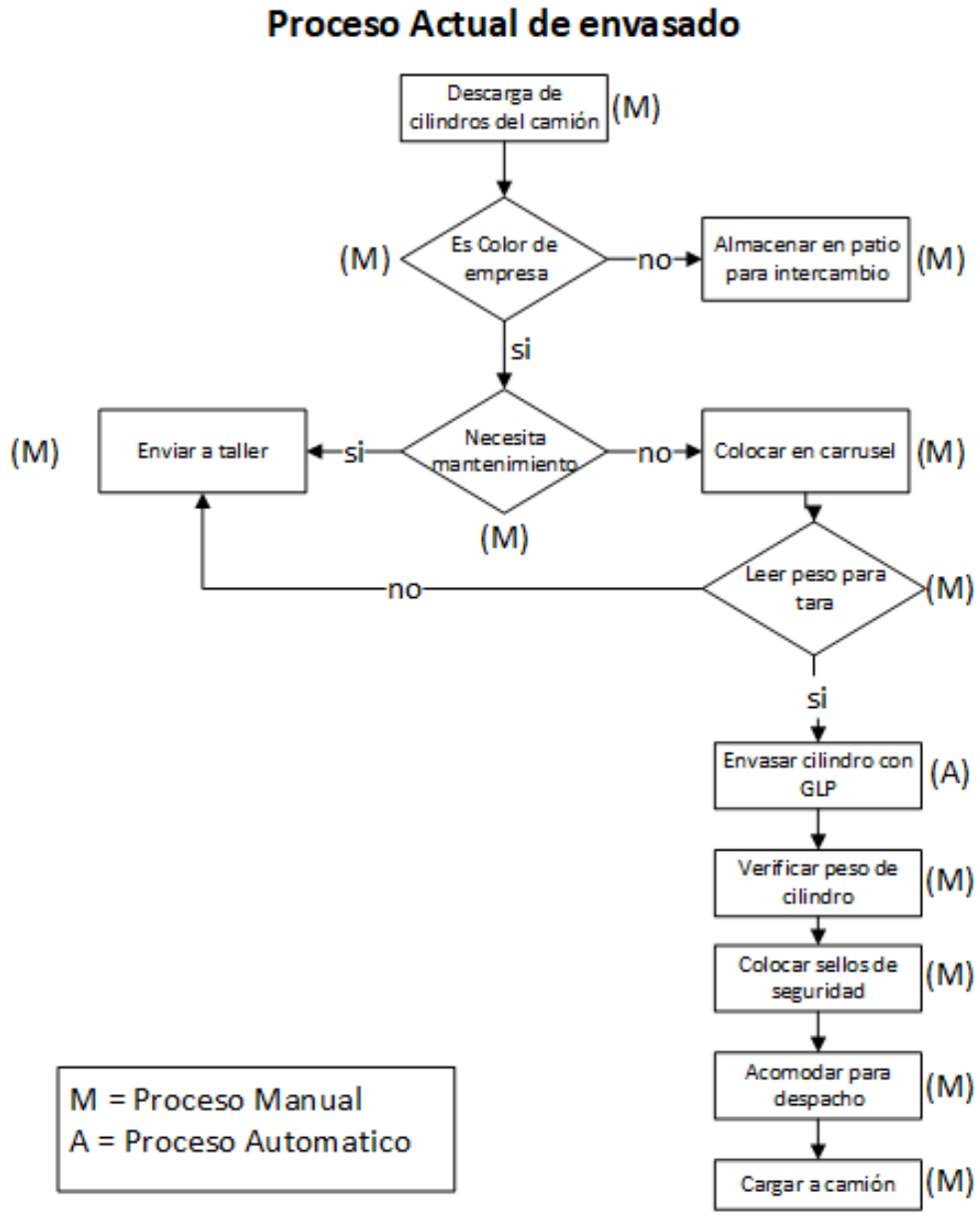


Figura 1. Proceso actual de envasado.

Figura 2. Área de descarga de cilindros de GLP.

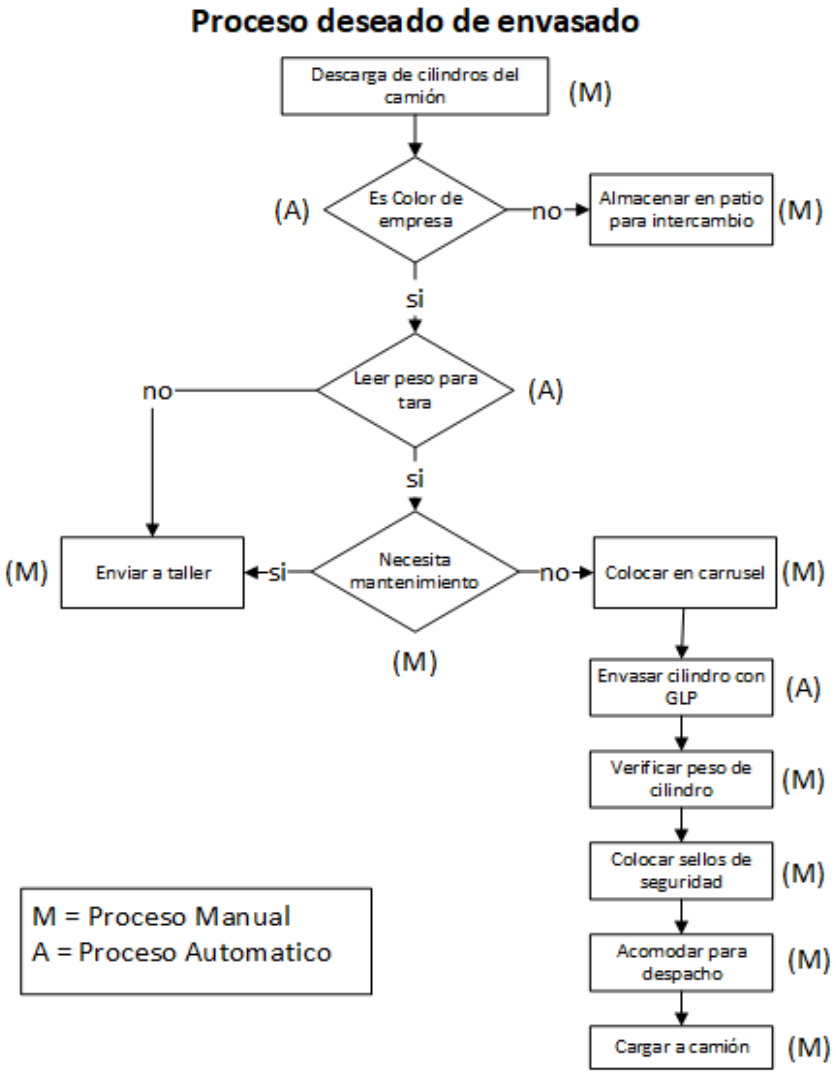


Figura 3. Proceso deseado de envasado.

1.3. Metodología

Se utilizará el método exploratorio y experimental debido a la innovación que se tendrá durante este proyecto. Se debe investigar sobre cómo integrar toda esta tecnología en el prototipo y después realizar las debidas pruebas previo a su implementación final.

1.4. Técnica de Detección de imagen

1.4.1. Haar Cascade Classifier

Es un algoritmo que permite la detección rápida de objetos con aprendizaje automático. Es utilizado para detectar mediante video o imagen. Fue propuesto por (Viola, Paul, Jones y Michael, 2001).

A través de este método se entrena al algoritmo por medio de un modelo cascada, este se entrena con varios ejemplos de imágenes del objeto que se requiere identificar. Para esto se considera imágenes positivas a las que a pesar de su resolución son el objeto a identificar y las imágenes negativas son cualquier objeto diferente al que se quiere identificar. Con esto se generará una base de comparación. Esta debe tener imágenes del objeto a detectar del mismo tamaño y con diferentes posiciones para tener diferentes ángulos de comparación.

El algoritmo contempla 4 etapas:

1.4.1.1. Funciones Haar

Este método de clasificación tiene varias etapas que se aplican a cada región de la imagen que se desea detectar, para que esto se genere un clasificador básico solo con 1 de 4 técnicas llamadas *Discrete Adaboost, Real Adaboost, Gentle Adaboost, longitboost*. (Viola et al, 2001). Por ejemplo, para armar un árbol de clasificación se toman estas regiones de las imágenes.

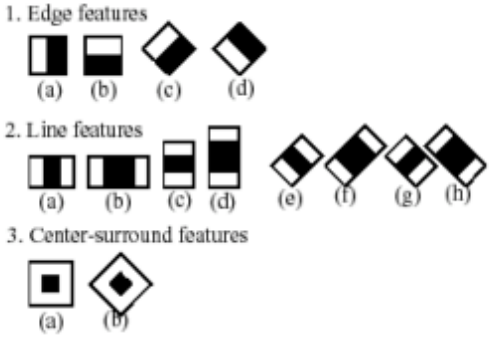


Figura 4. Regiones de características de HAAR

Tomado de (Viola et al, 2001)

1.4.1.2. Crear Imágenes Integrales

Para crear una imagen integral no es más que la transformación de la imagen original que nos dará como resultado la misma imagen del mismo tamaño donde el valor de cada pixel será la suma de todos los pixeles de la imagen original situados a la izquierda y arriba. Esto nos permite calcular rápidamente la suma de todos los pixeles de cualquier rectángulo de la imagen. (Imagen Integral, s.f)

$$II\ (x,y) = \Sigma\, I(x',y') \quad \text{Ecuación 1}$$

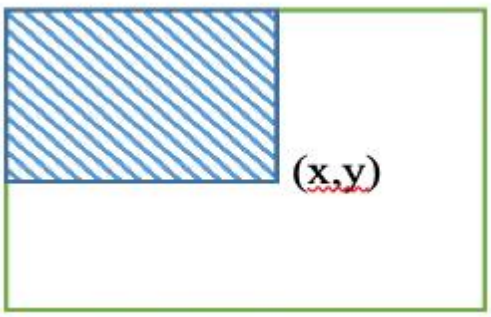


Figura 5. Suma de pixel de imagen

Lo que se obtiene es un cálculo constante de la suma de la intensidad de los pixeles de cualquier parte de la imagen, así se obtiene una detección sin importar la escala.

Por ejemplo:

Se tiene una matriz de 5x5 que representa a los pixeles de una imagen.

```
I = 5×5

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

Figura 6. Matriz 5x5

Tomado de (Integralimage, s.f).

En una imagen integral, cada píxel es la suma acumulada del píxel directamente encima de él y a su izquierda. Por ejemplo, en esta matriz trivial, el píxel de la matriz original en la fila 1, columna 1 (valor 17) no se modificará en una imagen integral porque está agregando 0s al valor. (La función añade una fila de 0s arriba y a la izquierda de la matriz original). La imagen integral calcula el valor de la imagen para el píxel (1, 2) en la matriz original, se agrega el píxel por encima de él (0) y el píxel a su izquierda (17): 24 + 17 + 0 = 41. Calculando el valor de la imagen integral para pixel (1, 3) en la matriz original, agrega el píxel por encima de él (0) y el píxel a su izquierda, que ya se ha sumado, 41. Por lo tanto, el valor en el píxel (2, 4) en la imagen integral es 1 + 41 + 0 = 42. Este proceso continúa para cada píxel de la imagen original. Obtenido de documentación Mathworks. (Integralimage, s.f).

```
J = 6×6

     0     0     0     0     0     0
     0    17    41    42    50    65
     0    40    69    77    99   130
     0    44    79   100   142   195
     0    54   101   141   204   260
     0    65   130   195   260   325
```

Figura 7. Matriz resultante

Tomado de (Integralimage, s.f).

1.4.1.3. Entrenamiento con *Adaboost*

Adaboost genera un meta-algoritmo robusto en la cual fija las mejores características y entrena los clasificadores *adaboost,* tiene 4 técnicas que son *Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost.* (Viola et al, 2001).

En este proceso se escanea toda la imagen por cada trama y se encuentra el objeto dentro de la imagen realizando la interacción de forma horizontal y vertical para así encontrar clasificadores débiles que son posibles pixeles que contengan parte del objeto, de esta manera se crea un identificador robusto y crea más posibles características del objeto que pueden ser comparadas n veces, para luego pasar al clasificador en cascada.

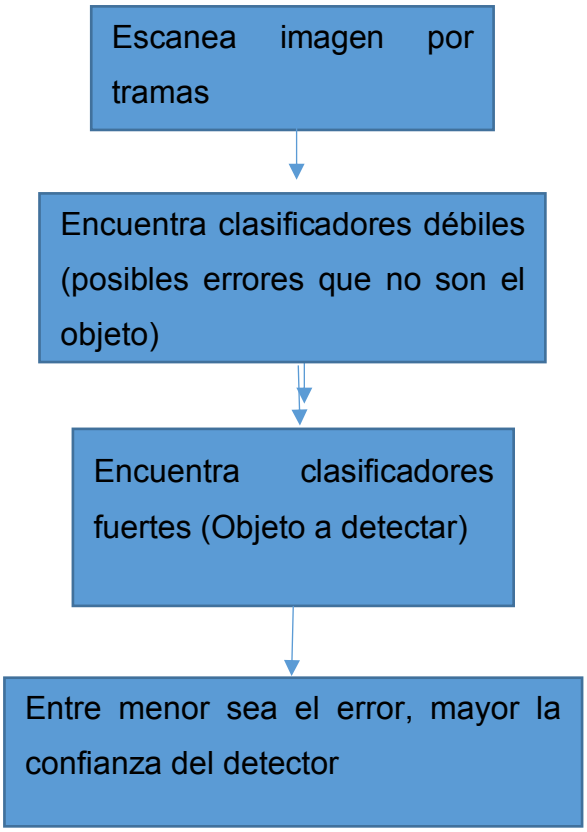


Figura 8. Proceso Algoritmo Adaboost

1.4.1.4. Clasificador en cascada

Se basa en una sucesión de pasos que contiene clasificadores simples llamados *Boosting*, creando así un clasificador más robusto tomando en cuenta un promedio de todas las imágenes ya entrenadas.

1.5. Análisis de configuración y variables del prototipo

1.5.1. Configuración de prototipo

Para realizar la detección de los cilindros de gas se tendrá un prototipo que por medio de la cámara tomará fotos de un cilindro de gas, para luego ser analizadas por un algoritmo que los compara e identifica el color y peso marcado.

Para obtener una detección precisa del cilindro de gas se genera una base de comparación de imágenes positivas y negativas, esto significa que el algoritmo clasificará cada imagen capturada y determinará si detecto el objeto.

Flujo de proceso de detección.

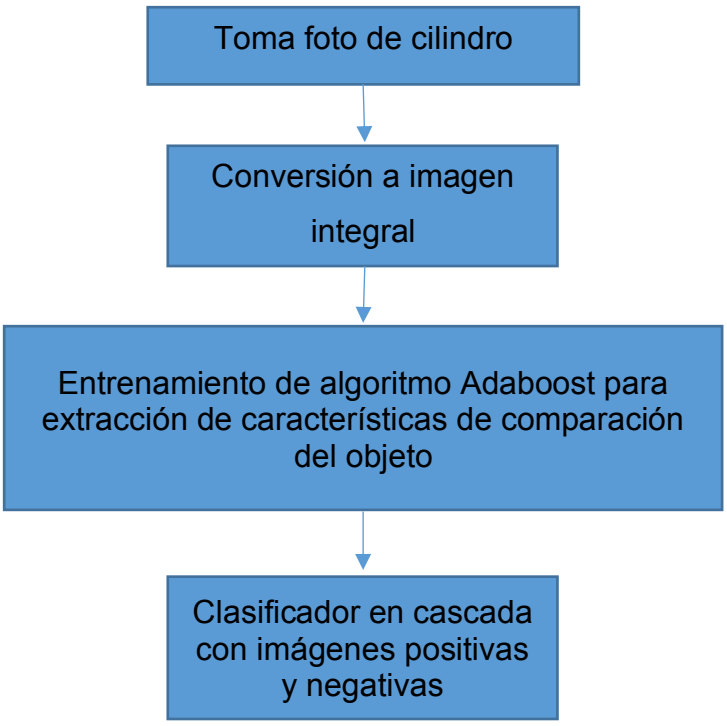


Figura 9. Fase 1 Aprendizaje de Objeto

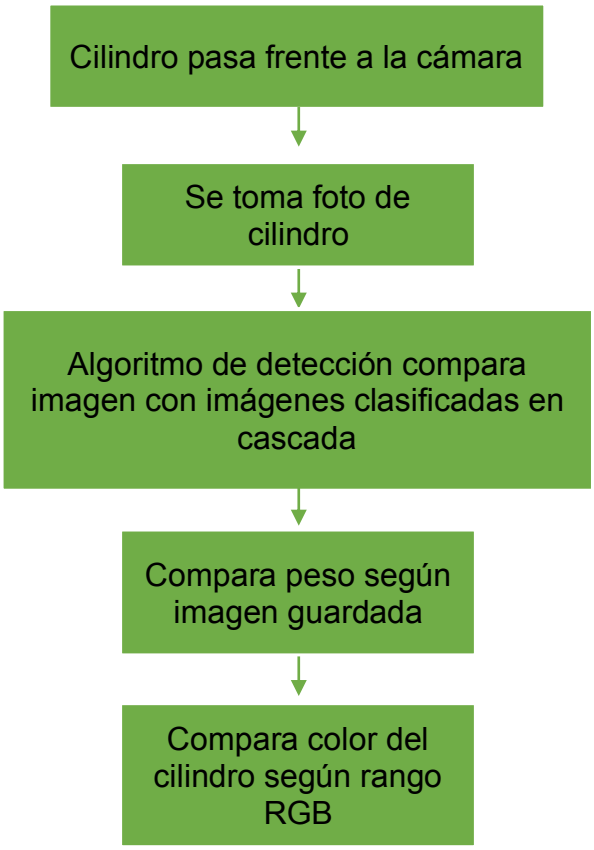


Figura 10. Fase 2 Detección

1.5.2. Las variables a considerar.

1.5.2.1. Color

De esta forma se identificará de que planta envasadora es cada cilindro, así el estibador podrá enviar a envasar solo los que son de su empresa. El color se lo configurará en base a un rango de color claro y oscuro ya que con la variación de iluminación ambiente puede tener un margen de error, esto se entenderá mejor en el capítulo de pruebas.

Para el prototipo se asumirá tener tres empresas a las cuales se las llamará Empresa A con color azul, Empresa B color amarillo y Empresa C color tomate, con el fin de identificarlas para su detección.

Color azul en rango RGB desde el mínimo al máximo representa en este caso a empresa A.

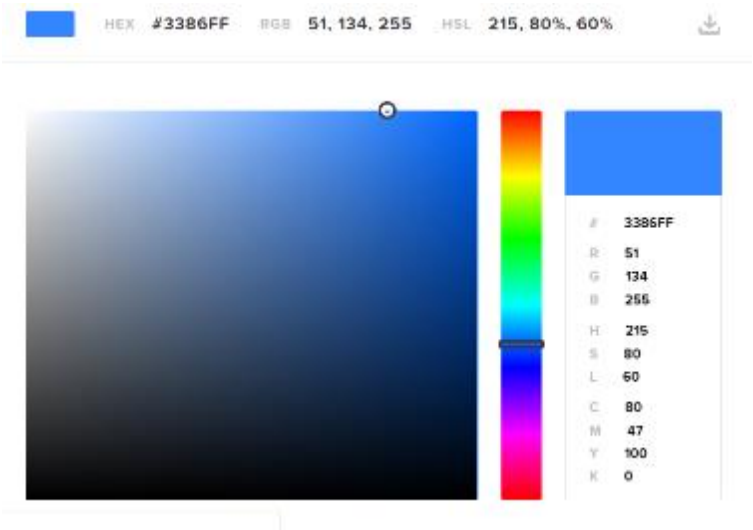


Figura 11. RGB mínimo en azul

Tomado de (Web HTML Color, s.f)

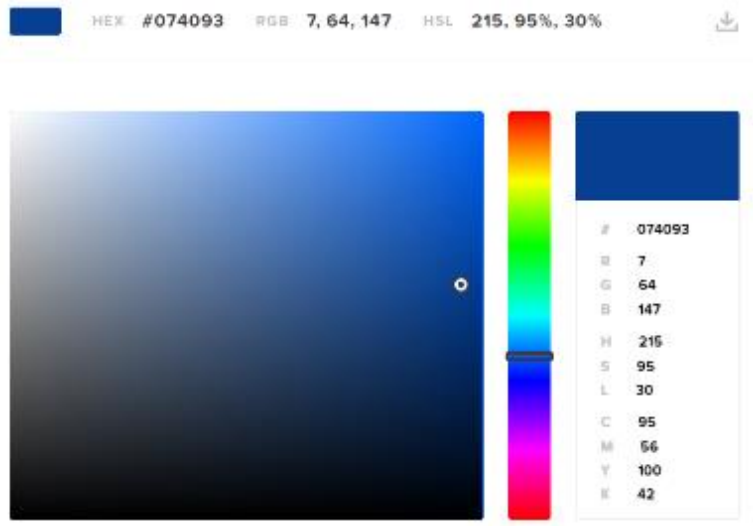


Figura 12. RGB máximo en azul

Tomado de (Web HTML Color, s.f)

Color amarillo en rango RGB desde el mínimo al máximo representa en este caso a empresa B.

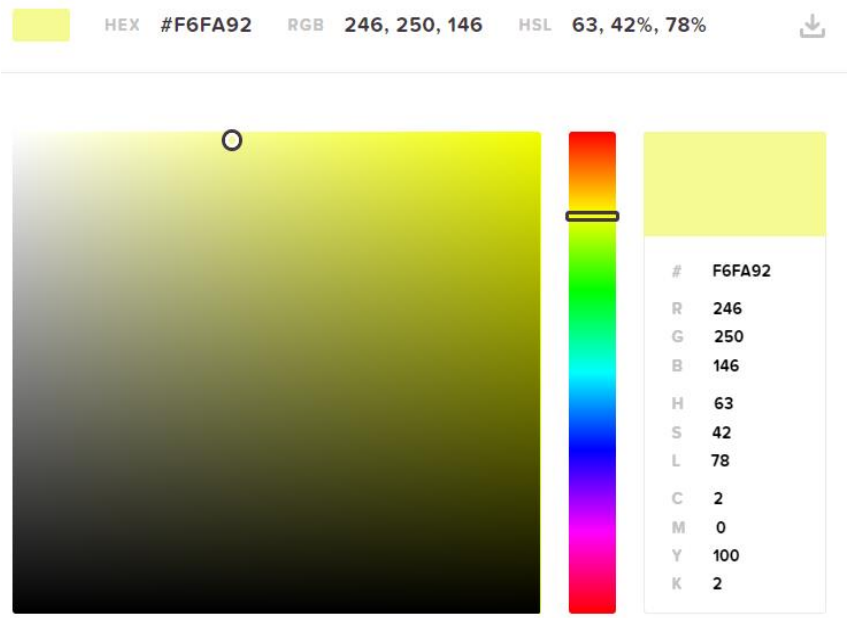


Figura 13. Color Amarillo RGB mínimo

Tomado de (Web HTML Color, s.f)

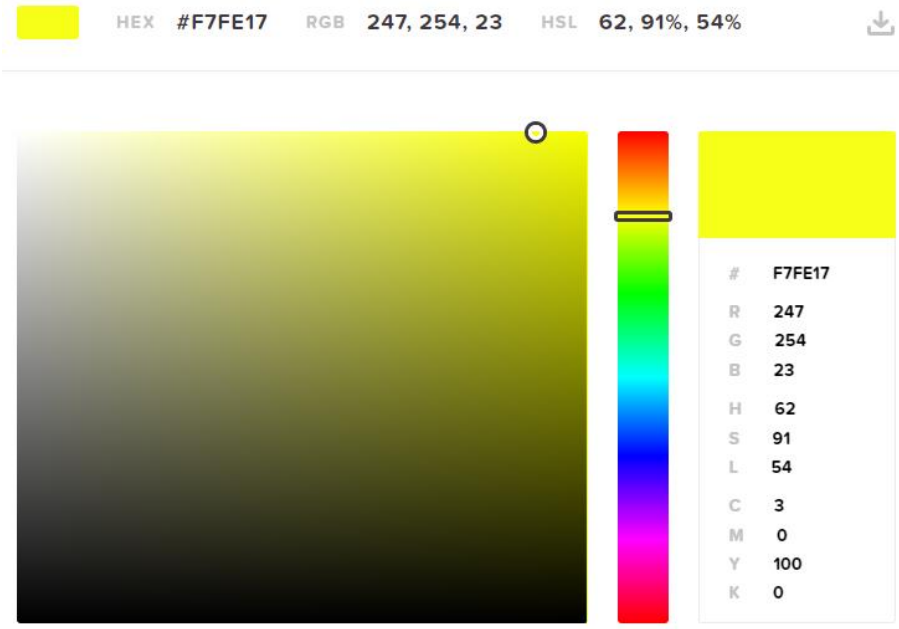


Figura 14. Color Amarillo RGB máximo

Tomado de (Web HTML Color, s.f)

Color amarillo en rango RGB desde el mínimo al máximo representa en este caso a empresa C.

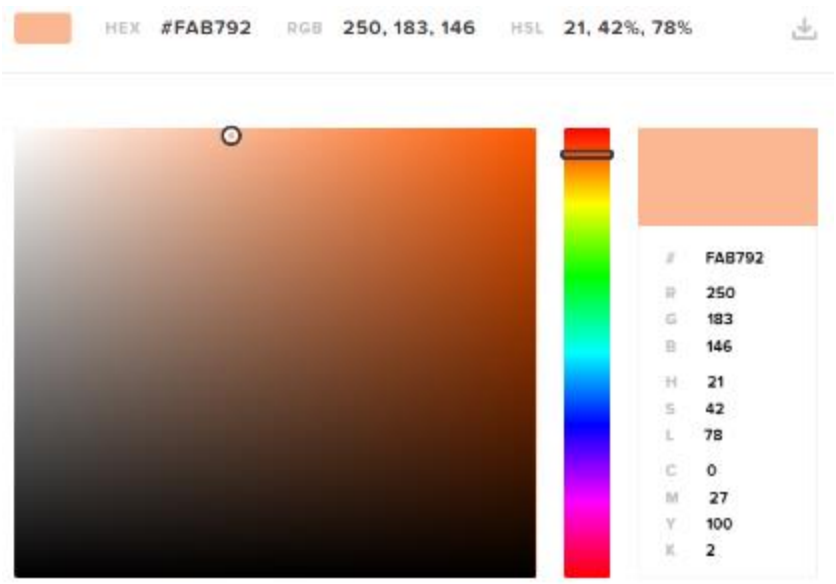


Figura 15. Color tomate RGB mínimo

Tomado de (Web HTML Color, s.f)

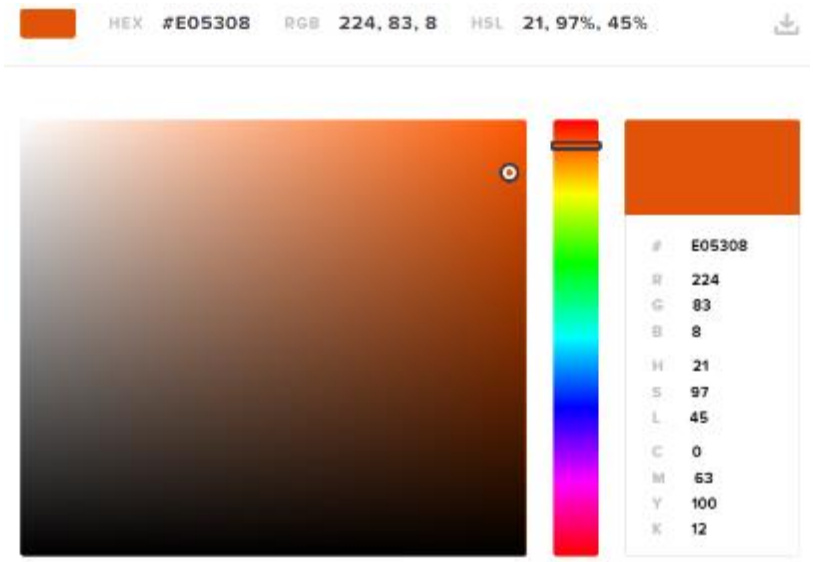


Figura 16. Color tomate RGB máximo

Tomado de (Web HTML Color, s.f)

1.5.2.2. Peso

Para la lectura del peso marcado se creará una base de fotos de diferentes pesos para luego ser analizados por el programa.

El peso marcado se encuentra en la parte superior del cilindro de gas.

Figura 17. Peso marcado en cilindro de gas

1.6. Conceptos Básicos

1.6.1. Raspberry Pi 3

Mini computadora que permite crear ambientes de programación unificando el hardware y software, posee un procesador 1.4 GHz de 64 *bits quad core*, memoria ram de 1 Gb, *dual band Wireless LAN*, *Extended* 40-*pin GPIO header*, salida HDMI, puertos USB y conecto micro USB para alimentación de 5v con 3A. (Raspberry Pi, s.f)

Figura 18. Raspberry Pi 3

Tomado de (Raspberry Pi, s.f)

1.6.2. Cámara Pi

Posee una resolución de 1080p, captura fotos en 5 MP y se conecta al raspberry por medio de un bus de datos.

Figura 19. Cámara PI

Tomado de (Raspberry Pi, s.f)

1.6.3. Cilindro de Gas

Tanque en el cual se realiza la distribución de GLP (Gas licuado de petróleo), el cual está compuesto de propano y butano.

Figura 20. Cilindro de Gas

2. ANALISIS DE TECNOLOGIA

En este capítulo se estudiará las herramientas, métodos de detección de objetos y librerías, mismos que permiten por medio de visión artificial detectar objetos.

2.1. Herramientas

2.1.1. Sistema operativo Raspbian

Es un sistema operativo basado en la distribución Debian orientado al dispositivo Raspberry Pi. Contiene programas básicos y utilidades que permiten comparar con las funciones de un PC, pero la utilidad se basa en la preinstalación de paquetes que permiten compilar programas. Esto quiere decir, que está directamente enfocado en el desarrollo de aplicaciones que fusionan el hardware y software fácilmente.

Existen actualmente comunidades que permiten que estas librerías se actualicen constantemente y hacen que se vea optimizado cada día el rendimiento en Raspberry Pi.

Cabe indicar que este sistema operativo no fue desarrollado propiamente por Raspberry Pi, fue creado por un grupo de programadores fanáticos del hardware y con el objetivo principal del proyecto Debian. Logrando desarrollar un sistema operativo liviano que fácilmente se instala en un micro SD y es totalmente personalizable.

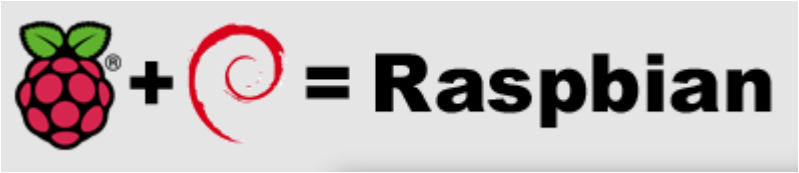


Figura 21. Logo de sistema operativo.

Tomado de (Raspberry Pi, s.f)

2.1.2. Librería Open CV

Open CV es una librería open source que permite la visión por computadora, se lo utiliza para fines académicos y proyectos, se lo puede integrar en interfaces Python, C++ y Java. Tiene versiones compatibles con Windows, Mac, Linux OS, Android y IOS. Fue desarrollada para eficiencia computacional enfocada en aplicaciones en producción, por ser escrita en C / C ++ permite aprovechar el procesamiento multi-core. Actualmente tiene una aceptación en todo el mundo, posee una comunidad de alrededor 47 mil usuarios y sus descargar bordea los

14 millones. Se utiliza fuertemente en la robótica avanzada y otras aplicaciones de are interactivo. (Open CV, s.f)

Se decide utilizar esta librería ya que es de uso libre y conjuntamente con Python se integran perfectamente.

Tiene la finalidad de extraer datos del mundo físico por medio de imágenes digitales para luego procesarlas con un computador.

Estos son algunos de los campos que se utiliza la visión artificial.

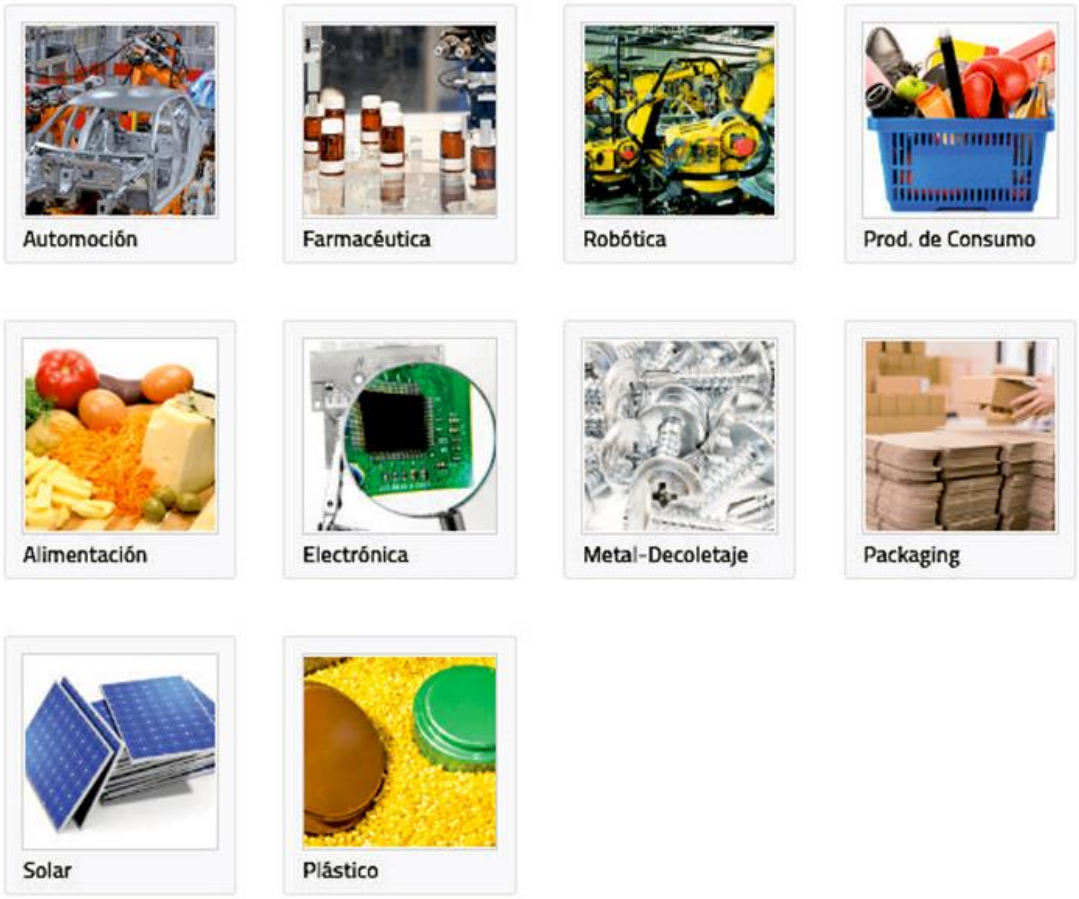


Figura 22. Industrias que aplican visión artificial.

Tomado de (Nolasco Valenzuela, 2018)

2.1.3. Python

Lenguaje de programación de interpretado, permite la programación orientada a objetos ya que es multiparadigma generando un código reducido y de fácil integración. Posee un repositorio propio siendo este de fácil acceso a los paquetes que se requiera. (Python, s.f.)

Para este proyecto se trabajará con la versión de Python 3.5 que es estable y la mayoría de los paquetes son compatibles.

Python es un lenguaje interpretado esto significa que no ejecuta el código fuente para su ejecución (Nolasco Valenzuela, 2018).

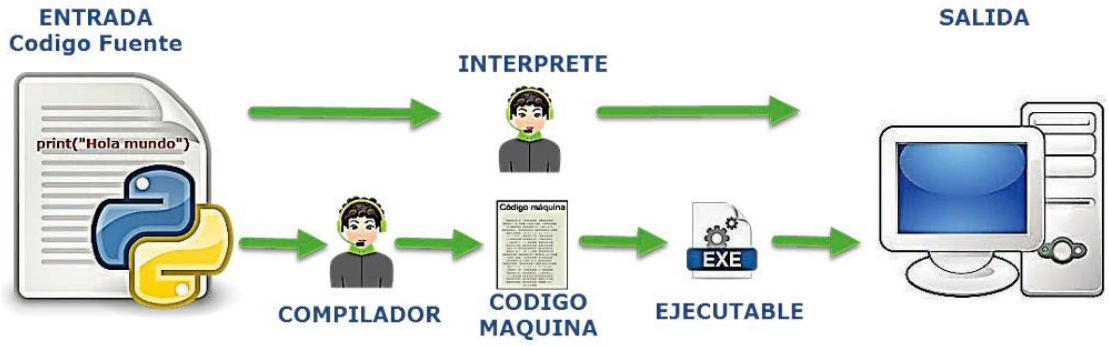


Figura 23. Proceso de compilación en Python

Tomado de (Nolasco Valenzuela, 2018)

En el aspecto comercial las empresas como Dropbox y Instagram utilizan Python en la actualidad, esta es una muestra que el lenguaje de programación está orientado a grande escala.

Python también es muy usualmente utilizado en *Data Sciencia* y *Machine Learning.* Entre los más comunes se tiene: Pandas, Scikit-Learn y Tensorflow.

A continuación, una lista de lenguajes en donde Python puede ser implementado:

- CPython – Escrito en C.
- IronPython – usado en .NET.

- Pypy – Python con mejor rendimiento con compilador JIT.
- Jython – usado en JVM.

Lista de Lenguajes más conocidos:

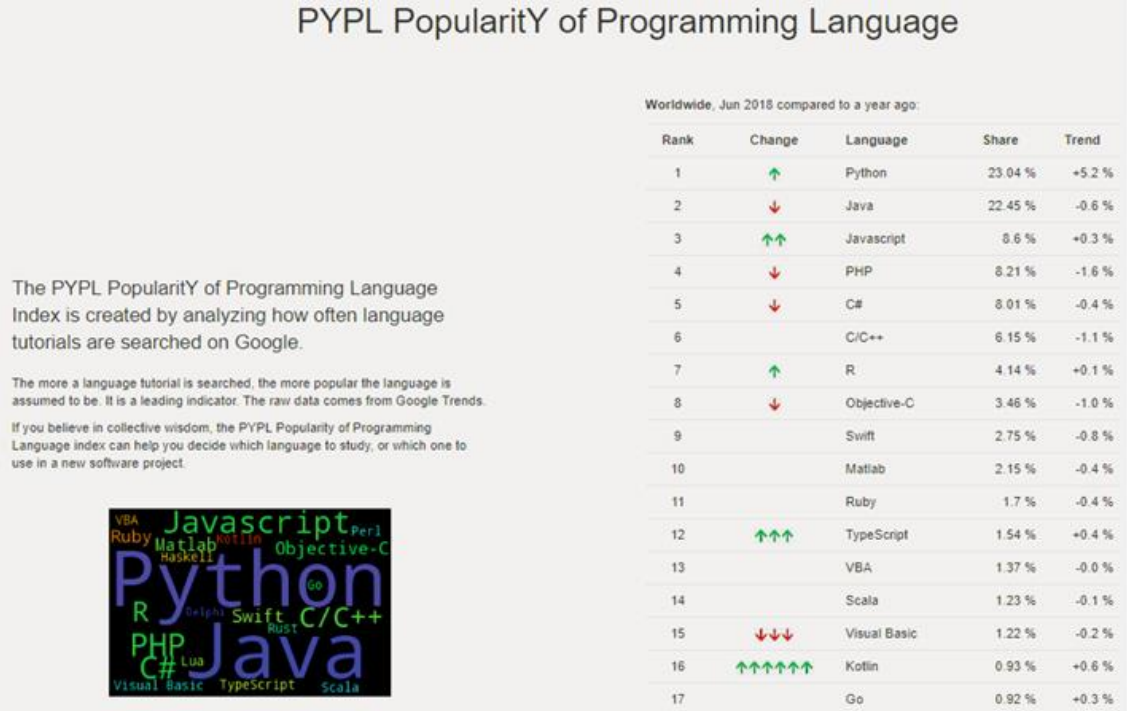


Figura 24. Lenguajes de programación.

Tomado de (Pypl, s.f.)

2.1.4. Librerías Matemáticas.

Existen librerías que conjuntamente logran filtrar imágenes y crear una muestra de comparación de objetos, se detalla las usadas en este proyecto.

2.1.4.1. Scipy

Colección de algoritmos matemáticos basados en la extensión Numpy de Python, esta librería brinda una potente serie de comandos y clases que le permiten al programa manipular y visualizar datos. Es una librería para procesamiento de datos que se compara como MATLAB, IDL, Octave, R-Lab y SciLab. Prácticamente es una librería que a nivel científico permite procesar datos a gran escala y con comandos simples.

En el caso de este proyecto se utiliza arreglos que permiten comparar cada trama de la imagen que se desea detectar.

2.1.4.2. Numpy

Es un paquete que posee objetos de matriz N-dimensional, funciones de difusión, algebra lineal, transformada de Fourier. Se utiliza con fines científicos ya que es un contenedor de datos multidimensional de datos genéricos. (Numpy, s.f.) En el proyecto nos permite almacenar todas las tramas de cada captura. Por su manejo de vectores y matrices facilita el cálculo de funciones matemáticas que se utilizan en el aprendizaje y entrenamiento de ciertos algoritmos de visión artificial.

2.1.4.3. Matplotlib

Librería que se utiliza para trazado de Python, permite tener gráficos interactivos, desarrollo de interfaces de usuario y servidores de aplicación web. (Pypi Org, s.f.) En el proyecto se utiliza para presentar en una ventana la captura de video.

Figura 25. Representación gráfica en 2D con Matplotlib.

Tomado de (Nolasco Valenzuela, 2018)

2.1.5. Librerías Esenciales

2.1.5.1. Picamara

Este paquete proporciona la interfaz para el módulo de la cámara del Raspberry pi para la versión de Python 2.7 o Python 3.7 (Picamara, s.f.)

2.1.5.2. Cv2

Es el paquete de opencv que contiene todas las clases y métodos de detección de objetos, para el proyecto se utiliza la versión 3.1.0 (Open CV, s.f.)

2.1.5.3. Glob

Librería que permite examinar el contenido que se encuentra en un directorio, permite la extracción de archivos, la utilizamos para crear la carpeta que contiene las capturas de aprendizaje para el reconocimiento. (Pypi Org, s.f.)

2.1.5.4. OS

Librería que permite manipular y obtener de los directorios sus ubicaciones, crearlos y eliminarlos. (Pypi Org, s.f.) En el proyecto me permite manipular archivos y directorios.

2.2. Machine Learning

Es una forma de encontrar relaciones entre cada dato analizado para poder realizar una determinada función en base a algoritmos. La mayor parte de desarrollos la tienen las empresas grandes como Google y Facebook que están implementándola cada día.

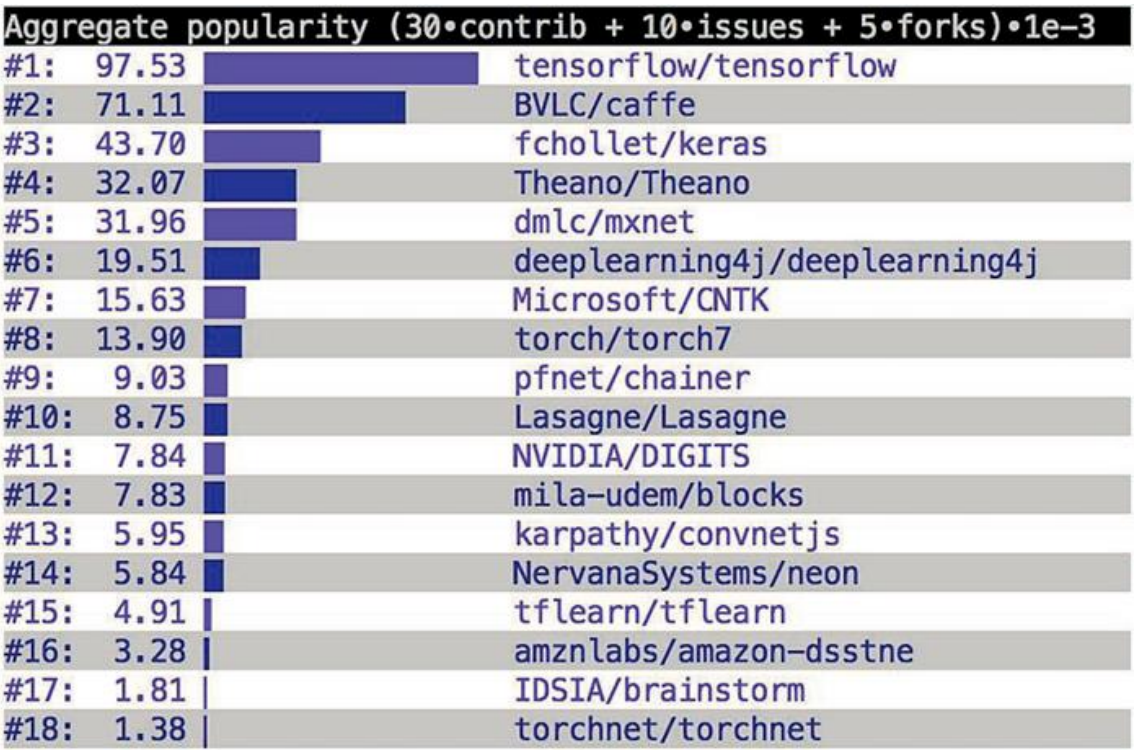


Figura 26. Lista de librerías más utilizadas.

Tomado de (Nolasco Valenzuela, 2018)

2.2.1. Librerias mas utilizadas.

Tensorflow

Librería Open source de *machine learning* que contiene nodos que trabajan con las operaciones matemáticas y arreglos de datos que son los tensores. Tiene una herramienta visual llamada *TensorBoard* que ayuda a visualizar los grafos cuando el código se encuentra en ejecución.

Ventajas:

- Utiliza Python y Numpy.
- Visualización de los grafos en ejecución.
- GPU múltiple.
- Sus modelos y datos son paralelos.

Desventajas:

- Su lentitud es comparable a otros.

- Sin licencia de comercialización.
- Existe mucho Python en su código por lo que le hace lento.
- Se debe entrenar individualmente cada batch.

Caffe

Este lenguaje trabaja con redes convencionales lo que hace que no trate como texto, sonido o series de datos. Sus ventajas son que presta atención a la velocidad, modularidad y expresión.

Ventajas:

- Buen procesamiento de imágenes.
- Entrena modelos.
- Su interfaz en amigable para el usuario.

Desventajas:

- No es comercial.
- No se destaca en redes concurrentes.
- Desarrollo muy lento.

Keras

Open source que utiliza redes neuronales escritos en Python, se puede integrar con otros lenguajes como Tensorflow y otros.

Ventajas:

- Permite extender y modular.
- Crecimiento acelerado.

Desventajas:

- Dependiente de otro Framework
- Es recomendable analizar los datos con otros Framework.

Theano

Es una librería de Python que utilizando arreglos multidimensionales optimiza y analiza expresiones.

Ventajas:

- Utiliza Python y Numpy.
- Trabaja con redes concurrentes.
- Permite evaluar de manera unitaria y tiene auto verificación de errores.

Desventajas:

- Es comparado como leguaje de bajo nivel.
- Contiene modelos muy grandes extensos de ejecución.

Mxnet

Framework de *Machine Learning* que contiene API en Pyhon, R, Julia y scala, estos son utilizados en varios servicios de AWS:

Ventajas:

- Gran diversidad de APIs
- Escalablemente fácil.

Desventajas:

- No es bueno en redes neuronales recursivas.
- No dispone de compilación.

2.3. Método de detección de imágenes

Para tener una referencia de detección de objetos y conocer sobre reconocimiento de imágenes se investiga el método clasificador cascada el cual nos da una idea de cómo debería ser desarrollado nuestro programa para la detección de cilindros.

2.3.1. Funciones Importantes

Para esto se debe tener en cuenta ciertas funciones que interactúan con las imágenes.

IMREAD – Permite la lectura de imágenes.

IMREAD_COLOR – Carga imagen a color

IMREAD_GRAYSCALE – Carga imagen en escala de grises.

IMREAD_UNCANGED – Carga la imagen sin alteraciones

IMSHOW – Muestra la imagen en una ventana.

IMWRITE – Utilizada para guardar la imagen.

2.3.2. Entrenamiento de Clasificador de Cascada

Para este método de clasificación se divide en dos etapas, una para el entrenamiento y otra para la detección, el modelo de detección utiliza modelos HAAR o LBP (Local Binary Patterns).

Este modelo de clasificación tiene ya preestablecido algunos archivos XML que contienen el entrenamiento en base a diferentes imágenes de un objeto en específico, por ejemplo, el de detección de rostros. Sin embargo, este modelo permite la creación de otros archivos que contengan los parámetros y características de cualquier otro objeto que se requiera. En el caso de este proyecto se creará en base a una serie de imágenes de cilindros de gas para que así se obtenga el archivo XML para la detección.

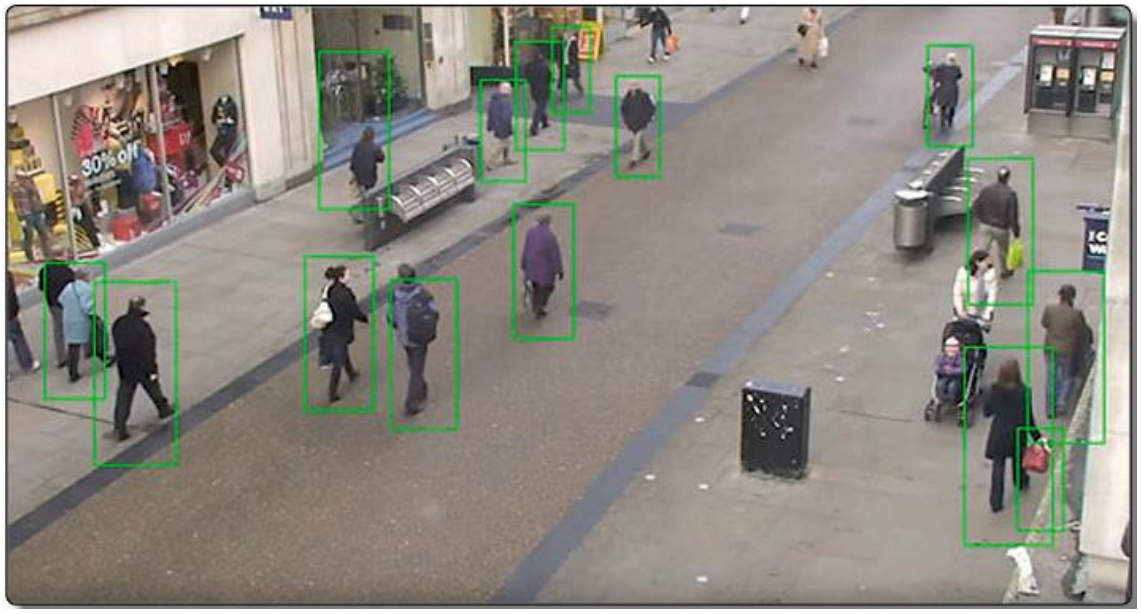


Figura 27. Detección de personas con entrenamiento cascada.

Tomado de (Nolasco Valenzuela, 2018)

2.3.3. Clasificación con Algoritmo AdaBoost

Este es un modelo predictivo que permite crear clasificadores débiles positivos y negativos, lo que quiere decir que un clasificador es una etapa o región en la imagen a detectar, logrando obtener una predicción exacta basándose en varios modelos anteriores. Fue desarrollado para la clasificación binario logrando ser el mejor para boosting, la mayoría de los métodos hacen referencia a AdaBoost. (Jason Brownlee, 2016).

3. HARDWARE Y SOFTWARE

Este capítulo se encargará de explicar el diseño del modelo de entrenamiento de detección y diseño de modelo de detección. El hardware del prototipo son todos los dispositivos conectados al Raspberry Pi y por el lado del software la preparación previa del ambiente de programación.

3.1. Preparación de ambiente de programación

Para llevar a cabo el desarrollo del algoritmo previamente se deben instalar librerías, las cuales permitirán correr el programa sin ningún problema. Es por eso que se detalla a continuación una explicación de la instalación de las librerías.

3.1.1. Instalación de librería Open cv

Es la principal librería que se requiere para lograr el objetivo de detección, ya que contiene toda la lógica de visión artificial para configurar la detección del objeto, es por eso que se debe instalar la versión más estable que se encuentra publicada en la página web oficial de open cv.

Desde la consola se ingresa el siguiente código para la descarga de todos los archivos de open cv.

Código:

```
cd ~ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip
unzip opencv.zip wget -O opencv_contrib.zip
```

https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip unzip opencv_contrib.zip

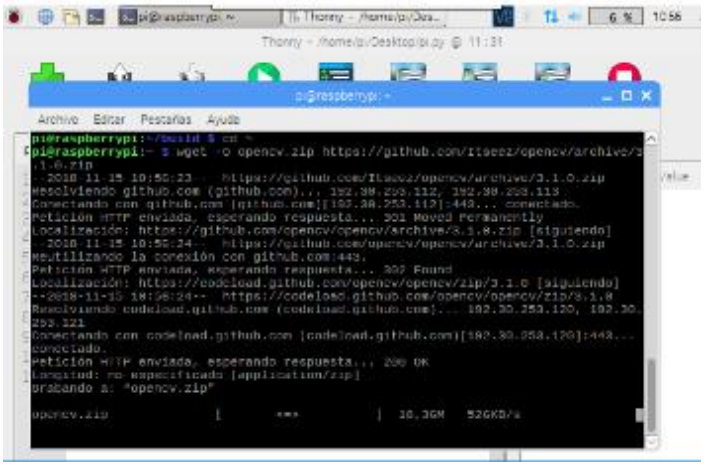


Figura 28. Instalación de paquetes open cv en Raspberry Pi.

Se verifica el tamaño en disco antes de desempaquetar los archivos open cv.

```
pi@raspberrypi:~ $ df -h
S.ficheros     Tamaño Usados  Disp Uso% Montado en
/dev/root         15G   4,2G  9,8G  30% /
devtmpfs         434M      0  434M   0% /dev
tmpfs            438M      0  438M   0% /dev/shm
tmpfs            438M    12M  427M   3% /run
tmpfs            5,0M   4,0K  5,0M   1% /run/lock
tmpfs            438M      0  438M   0% /sys/fs/cgroup
/dev/mmcblk0p1    44M    22M   22M  51% /boot
tmpfs             88M      0   88M   0% /run/user/1000
```

Figura 29. Verificación de tamaño en memoria SD.

Se libera espacio borrando aplicaciones que no se utilizará, esto se requiere ya que se estará trabajando desde una memoria SD de un tamaño de 32 GB, se logrará así optimizar el almacenamiento.

```
pi@raspberrypi:~ $ sudo apt-get purge wolfram-engine
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
```

Figura 30. Comando para purgar espacio en memoria SD

Siempre se debe actualizar todos los paquetes, previo a cualquier instalación con el comando: sudo apt-get update.

```
pi@raspberrypi:~ $  sudo apt-get update
Obj:1 http://archive.raspberrypi.org/debian stretch InRelease
Des:2 http://raspbian.raspberrypi.org/raspbian stretch InRelease [15,0 kB]
Des:3 http://raspbian.raspberrypi.org/raspbian stretch/main armhf Packages [11,7
 MB]
Descargados 11,7 MB en 19s (586 kB/s)
Leyendo lista de paquetes... Hecho
```

Figura 31. Actualización de paquetes.

Luego se instalará la dependencia cmake que permitirá construir el ambiente de programación en conjunto con las librerías de open cv.

```
pi@raspberrypi:~ $ sudo apt-get install build-essential cmake pkg-config
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
build-essential ya está en su versión más reciente (12.3).
cmake ya está en su versión más reciente (3.7.2-1).
pkg-config ya está en su versión más reciente (0.29-4).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
```

Figura 32. Instalación de dependencia CMAKE.

Las librerías que también se instalará son las de reconocimiento de imágenes que se instalarán por separado, ejecutando el comando:

sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev l

Figura 33. Librería para reconocimiento de imágenes.

También se instalará librerías de códecs de video que sirven para capturar las imágenes por medio de la cámara.

Figura 34. Librería que contiene los códec de video.

Se instalará la versión de Python 2.7 y Python 3 para el desarrollo del algoritmo de programación, este será el lenguaje en el que se escribirá el código.

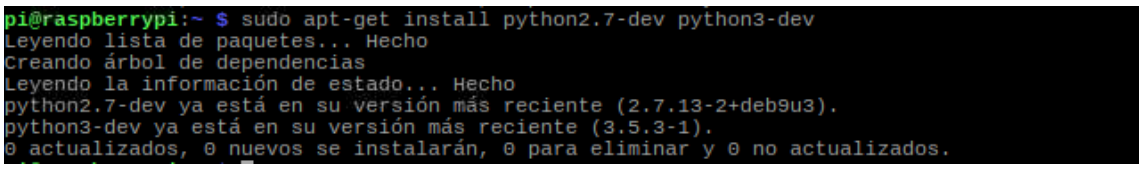


Figura 35. Instalación de Python 2.7 y Python 3

Se descargará la versión 3.1.0 de open cv y se instalará.

Figura 36. Instalación de open cv 3.1.0

Se creará un ambiente virtual para la ejecución del programa, logrando la optimización de recursos de hardware y un mejor desempeño de la deteccion en el programa python.

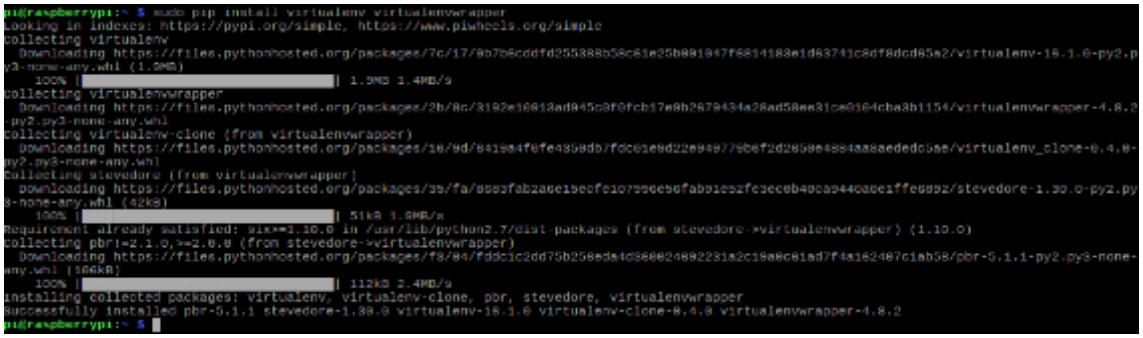


Figura 37. Creación de ambiente virtual

3.2. Diseño de Detector de Cilindro de gas

3.2.1. Diagrama de Flujo de Aprendizaje

Esta fase se debe realizar siempre que existan cambios en los objetos, por ejemplo, ingreso de nuevos colores de cilindros o nuevos pesos específicos.

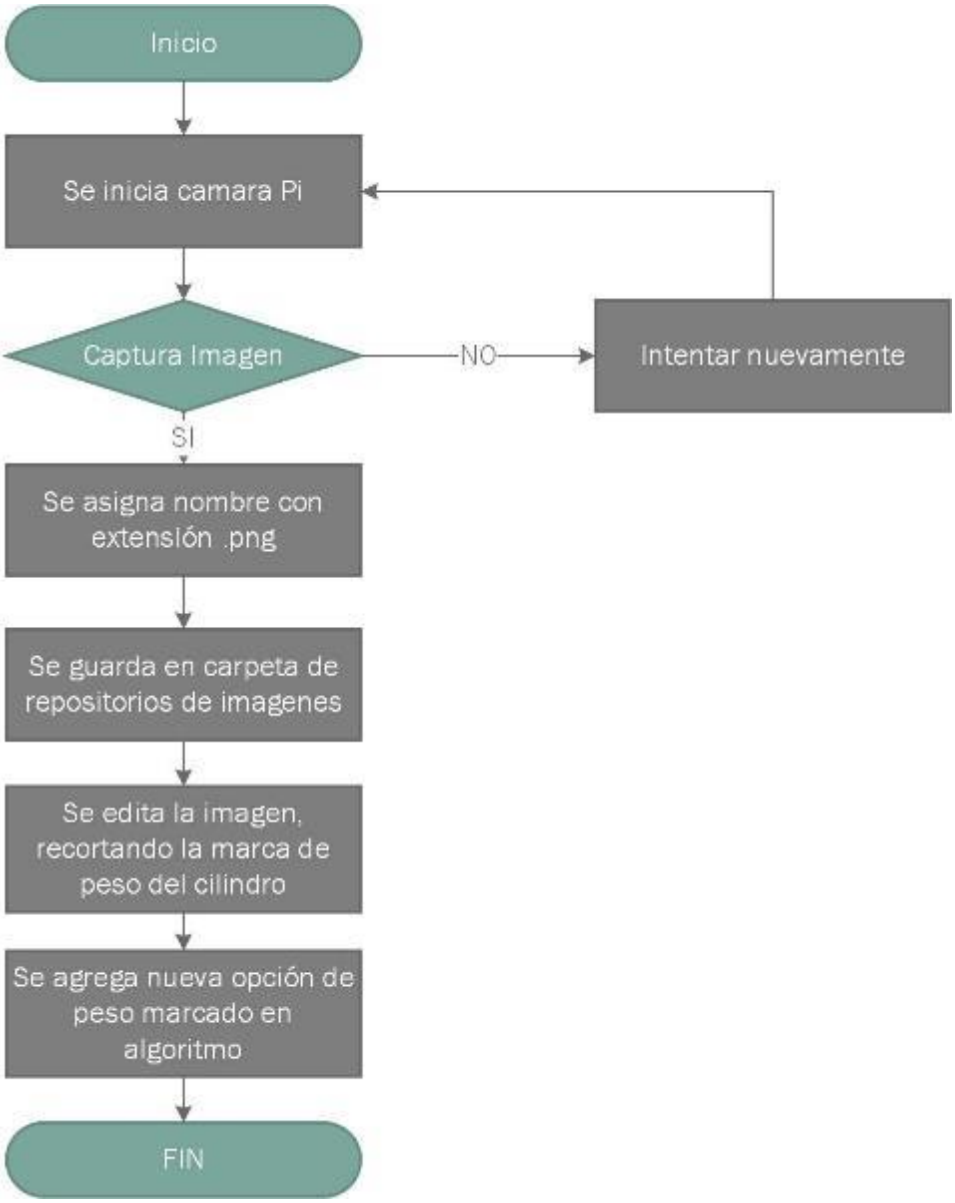


Figura 38. Fase aprendizaje

3.2.2. Diagrama de Flujo de Reconocimiento

Este es el proceso que realizará el programa para detectar el color y peso marcado.

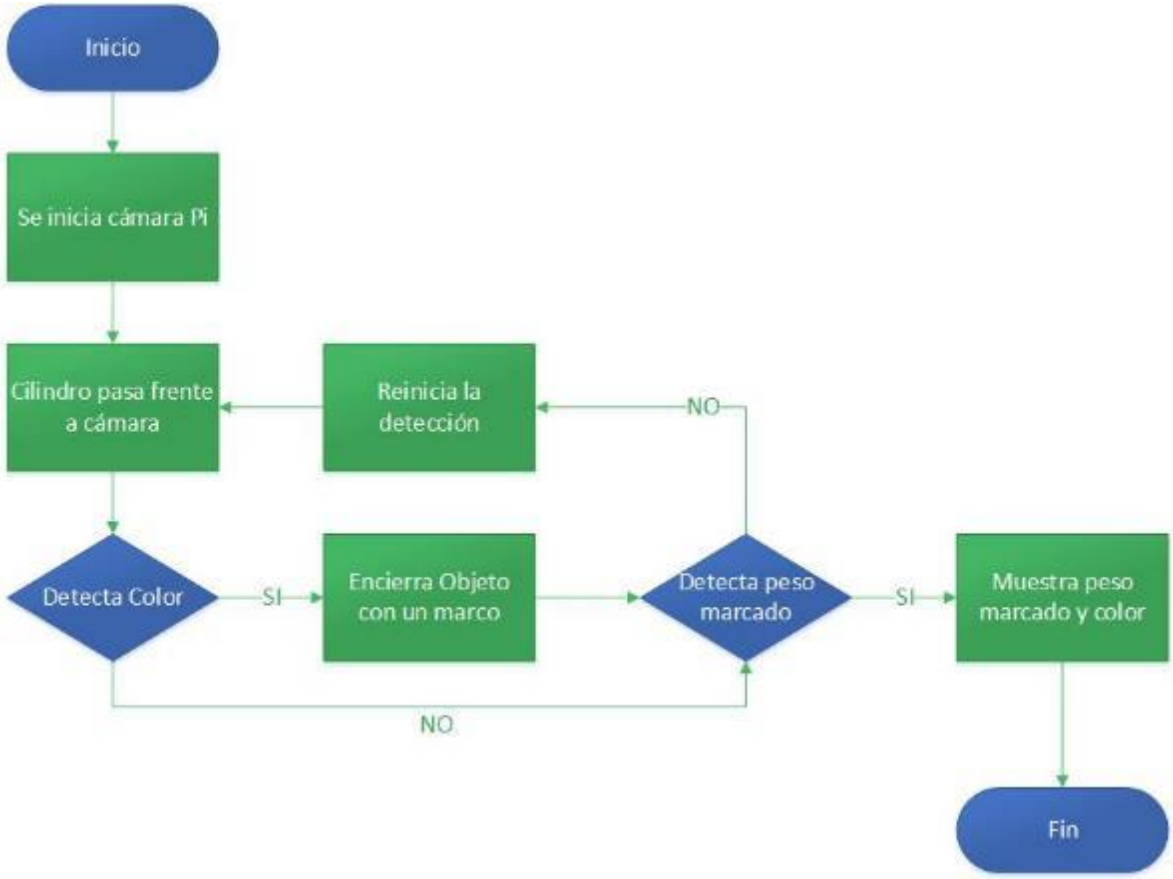


Figura 39. Diagrama de Detección

3.2.3. Color

La definición de color se realizará analizando el color RGB máximo y mínimo que tiene el cilindro al momento que lo detecta la cámara, con esa captura se extraerá el código de colores en tonos claros y oscuros, en las siguientes imágenes se indicará el procedimiento de extracción de color RGB.

Figura 40. Se obtiene color claro de cilindro de gas

Tomado de (HTML Color Picker, s.f.)

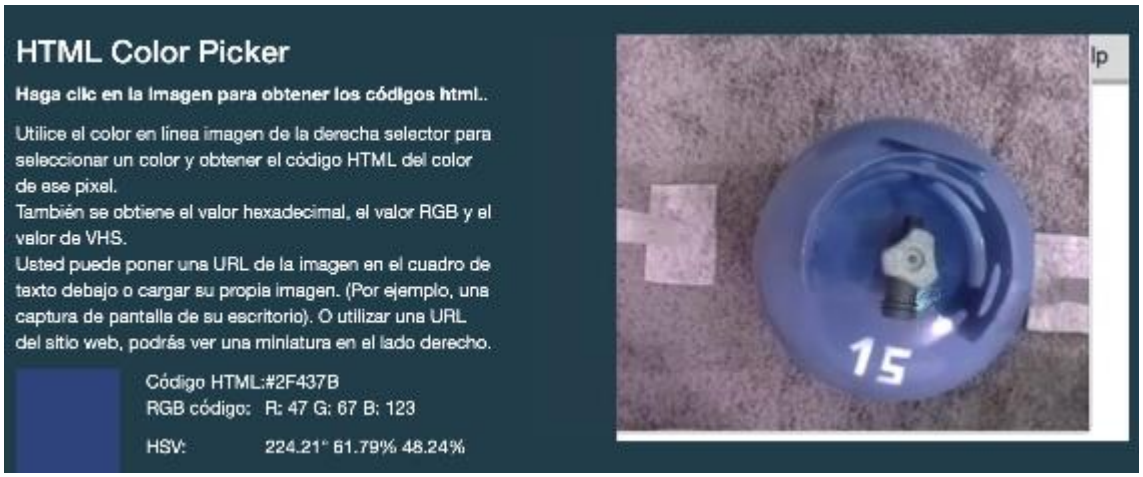


Figura 41. Se obtiene el color oscuro de cilindro de gas

Tomado de (HTML Color Picker, s.f.)

3.2.4. Peso Marcado

Para el peso se capturará nuevamente una imagen del cilindro de gas y se recortará el número que indica el peso marcado, para la detección de diferentes pesos se repite este proceso de igual forma.

Se debe generar diferentes opciones de ubicación del número marcado, es por eso que se debe rotar el número en diferentes ángulos para así detectar de cualquier forma que se encuentre.

3.3. Diseño fisico de prototipo

Para la construcción del prototipo se creará un brazo que sujetará la cámara en la parte superior de esta forma el cilindro será detectado solo en el espacio que cubre el cilindro de gas.

Figura 42. Prototipo de reconocimiento de cilindros de gas

Figura 43. Prototipo de reconocimiento de cilindros de gas

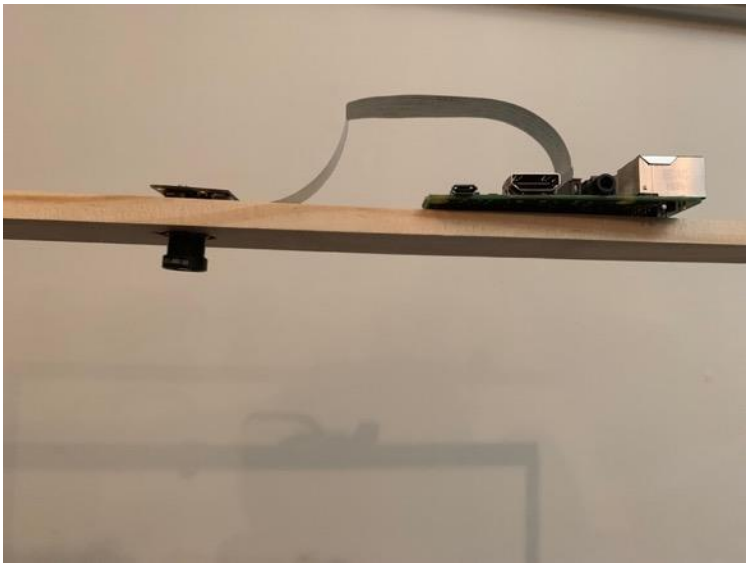


Figura 44. Raspberry Pi con cámara Pi

4. DESARROLLO DE PROTOTIPO

En este capítulo se implementará la lógica de programación que se explicó en el diseño del prototipo.

El algoritmo de programación quedará de la siguiente forma para lograr la detección de color y peso marcado.

4.1. Declaración de librerias

Se declaran las librerías a utilizar, la mas sobresaliente es cv2 que hace referencia a OpenCV permitiendo la detección de objetos.

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

Captura y lectura de la camara PI

Se realiza la creación de instancia a objeto PiCamera la cual permitirá captar el video, se define la resolución de cámara y tramas óptimas para la reproducción del video. También, se establece el camino donde se guardará las imágenes que servirán como repositorio base para la detección.

```
camera = PiCamera()
camera.resolution = (320, 240)
camera.framerate = 10
rawCapture = PiRGBArray(camera, size=(320, 240))
kernel = np.ones((5,5),np.uint8)
imagePaths = sorted(glob.glob("imagenes" + "/*.png"))
```

Rango RGB de los colores a detectar

Los colores se definen por medio del código RGB (Red, Green and Blue) de cada color, se requiere que se establezca el color más oscuro y claro para tener un rango de colores independiente de la luminosidad y ruido.

```
Azul_bajo = np.array([112, 61, 45], dtype = "uint8") # OSCURO
Azul_Alto = np.array([139, 96, 93], dtype = "uint8")#  CLARO

Amarillo_bajo = np.array([252, 193, 2], dtype = "uint8")# OSCURO
Amarillo_Alto = np.array([246, 190, 7], dtype = "uint8")#  CLARO

Tomate_bajo = np.array([254, 80, 115], dtype = "uint8") # OSCURO
Tomate_Alto = np.array([233, 79, 19], dtype = "uint8")#  CLARO

Gris_bajo = np.array([112, 105, 118], dtype = "uint8") # OSCURO
Gris_Alto = np.array([123, 113, 126], dtype = "uint8")#  CLARO
```

4.2. Detección de Color

Es la variable que indica si esta en modo aprendizaje o de reconocimiento.

```
VerificarColor = 1
```

```
time.sleep(0.1)
```

Captura los frames de la cámara.

```
for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
  time.sleep(0.1)
  image = frame.array
  imagen_color = image
```

Se muestra en ventana el video

```
  cv2.imshow("Deteccion de Cilindro", image)
```

Ingreso a modo detección cuando la variable VerificarColor = 1

```
 if (VerificarColor==1):
```

Se utiliza la librería cv2 para detectar objeto según el color configurado, se lo configura para cada color, realizando la comparación de cada pixel en base al color RGB.

Para color Gris

```
c_gris = cv2.inRange(imagen_color, Gris_bajo, Gris_Alto)
    c_gris = cv2.GaussianBlur(c_gris, (3, 3), 0)
    (_,    cnt_gris,    _)    =    cv2.findContours(c_gris.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)


    if len(cnt_gris) > 0:
      print 'COLOR GRIS DETECTADO\n'
      mascara = cv2.inRange(imagen_color, Gris_bajo, Gris_Alto)
      opening = cv2.morphologyEx(mascara, cv2.MORPH_OPEN, kernel)
      x,y,w,h = cv2.boundingRect(opening)
      cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),3)
      cv2.imshow("Deteccion de Cilindro", imagen_color)
```

Para color Tomate

```
    c_tmt = cv2.inRange(imagen_color, Tomate_bajo, Tomate_Alto)
    c_tmt = cv2.GaussianBlur(c_tmt, (3, 3), 0)
```

```
    (_,   cnt_tomate,   _)   =   cv2.findContours(c_tmt.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
     if len(cnt_tomate) > 0:

      print 'COLOR TOMATE DETECTADO\n'
      mascara    =    cv2.inRange(imagen_color,    Tomate_bajo,
Tomate_Alto)
      opening = cv2.morphologyEx(mascara, cv2.MORPH_OPEN, kernel)
      x,y,w,h = cv2.boundingRect(opening)
      cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),3)
      cv2.imshow("Deteccion de Cilindro", imagen_color)
```

Para color Amarillo

```
    c_amarl    =    cv2.inRange(imagen_color,    Amarillo_bajo,
Amarillo_Alto)
    c_amarl = cv2.GaussianBlur(c_amarl, (3, 3), 0)
    # find contours in the image
    (_,   cnt_amarl,   _)   =   cv2.findContours(c_amarl.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    if len(cnt_amarl) > 0:
      print 'COLOR AMARILLO DETECTADO\n'
      mascara    =    cv2.inRange(imagen_color,    Amarillo_bajo,
Amarillo_Alto)
      opening = cv2.morphologyEx(mascara, cv2.MORPH_OPEN, kernel)
      x,y,w,h = cv2.boundingRect(opening)
      cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),3)
      cv2.imshow("Deteccion de Cilindro", imagen_color)
```

Para color Azul

```
    c_azul = cv2.inRange(imagen_color, Azul_bajo, Azul_Alto)
    c_azul = cv2.GaussianBlur(c_azul, (3, 3), 0)
    (_,    cnt_azul,    _)    =    cv2.findContours(c_azul.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    if len(cnt_azul) > 0:
      print 'COLOR AZUL DETECTADO\n'
```

```
        mascara = cv2.inRange(imagen_color, Azul_bajo, Azul_Alto)
        opening = cv2.morphologyEx(mascara, cv2.MORPH_OPEN, kernel)
        x,y,w,h = cv2.boundingRect(opening)
        cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),3)
        cv2.imshow("Deteccion de Cilindro", imagen_color)

MiImagenGris = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Una vez que se conoce el código de colores mínimos y máximos este reconocerá el color del cilindro de gas.

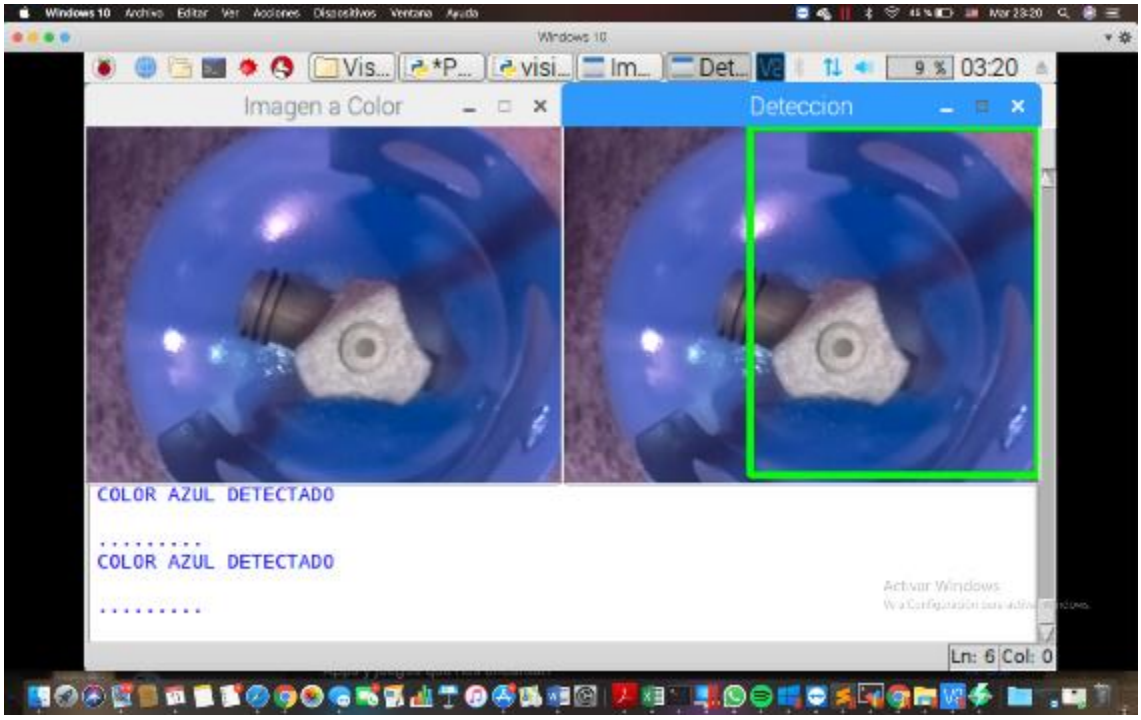


Figura 45. Detección de color en cilindro de gas

4.3. Ejecución de programa

4.3.1. Detección de Peso y objeto

Se comparan las imágenes guardadas en la carpeta imágenes, tanto el modelo físico del cilindro como los números marcados, cada imagen tiene un ID con el cual el programa logra reconocer. Las fotos tomadas están en escala de grises.

```
  index = 0
  for (imagePath) in zip(imagePaths):
    DirDestino= imagePaths[index]
imagenes\Cargador.png
    NombreObjetoLeido = DirDestino.strip(".png")
    NombreObjetoLeido = NombreObjetoLeido.strip("imagenes")
    NombreObjetoLeido                                          =
NombreObjetoLeido[1:len(NombreObjetoLeido)]
    ImagenLeidaFolder = cv2.imread(DirDestino,0)
    w, h = ImagenLeidaFolder.shape[::-1]
    resBBtt                                                    =
cv2.matchTemplate(MiImagenGris,ImagenLeidaFolder,cv2.TM_CCOEFF_N
ORMED)
    threshold = 0.8
    locBBtt = np.where( resBBtt >= threshold)


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


      if NombreObjetoLeido =="141":
        print 'Cilindro 14 Kg'
        break


      if NombreObjetoLeido =="142":
        print 'Cilindro 14 Kg'
        break


      if NombreObjetoLeido =="143":
        print 'Cilindro 14 Kg'
        break


      if NombreObjetoLeido =="144":
        print 'Cilindro 14 Kg'
        break
```

```
      if NombreObjetoLeido =="151":
        print 'Cilindro 15 Kg'
        break

      if NombreObjetoLeido =="152":
        print 'Cilindro 15 Kg'
        break
      if NombreObjetoLeido =="153":
        print 'Cilindro 15 Kg'
        break
      if NombreObjetoLeido =="154":
        print 'Cilindro 15 Kg'
        break


    else:
      print '.........'

  rawCapture.truncate(0)

  if cv2.waitKey(1) == 27:
    break
cv2.destroyAllWindows()
```

El directorio imágenes contendrá el peso del cilindro de gas en diferentes posiciones para la respectiva comparación dentro del algoritmo.

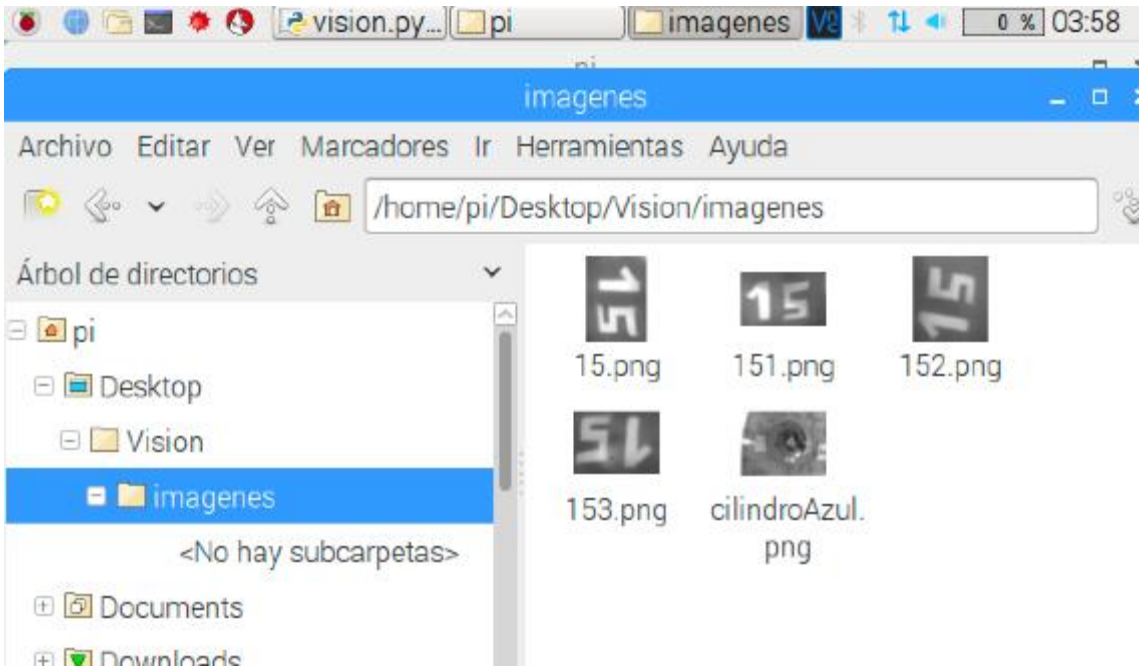


Figura 46. Etiqueta de peso marcado en diferentes posiciones

En el código se colocarán las diferentes opciones que tendrá el programa para comparar la posición y número detectado.

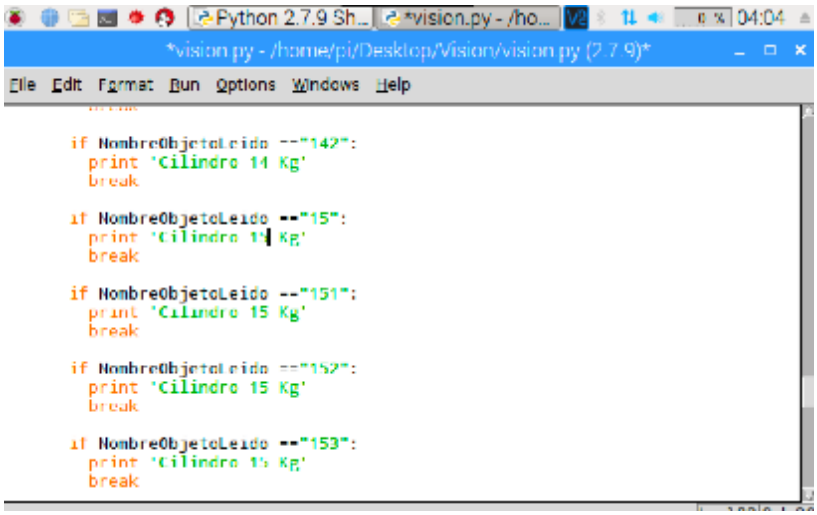


Figura 47. Configuración de opciones en algoritmo de detección

4.3.2. Script de entrenamiento

Script para fase de entrenamiento se activa cuando la variable VerificarColor = 0, una vez dentro lo que hace es capturar fotos, convertirlas en escala de grises y guardar en la carpeta “imágenes”.

```
TeclaPresionada = cv2.waitKey(1)


  if TeclaPresionada==105:    #TECLA (i)
    #print 'Presiono la tecla (i) para proceder con el
almacenamiento de la imagen'
    cv2.imshow("IMAGEN QUE SERA GUARDADA", MiImagenGris)


    TeclaPresionada = cv2.waitKey(3)
    cadena1 = raw_input('Ingrese el nombre del objeto seguido de
enter(Sino desea guardar solo presione enter):')
    if cadena1!="":
      cadena1 = 'imagenes/' + cadena1 + '.png'
      print 'LA IMAGEN SE GUARDO con el nombre:\n',cadena1
      cv2.imwrite(cadena1,MiImagenGris)
    else:
      print 'NO SE GUARDO LA IMAGEN\n'
```

4.4. Compilación de Programa

Una vez se obtiene el paquete ejecutable del programa se procede a generar el aprendizaje del algoritmo para la detección.

4.4.1. Fase de Entrenamiento de Software

Se ejecutará primero el programa de entrenamiento que capturará diferentes posiciones del cilindro original. Creará archivos de imágenes que serán guardados en el directorio imágenes y estarán identificados con un Id.

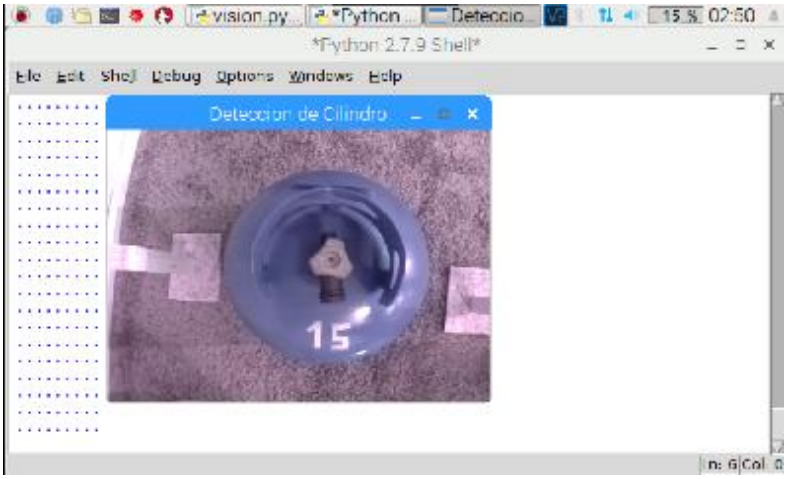


Figura 48. Captura de cilindro con Script de detección

Se presionará la tecla “i” para capturar cilindro de gas y convertir en escala de grises para posteriormente comparar en el algoritmo de detección.

Figura 49. Conversión a escala de gris.

Se asignará un nombre y se guardará en una carpeta llamada imágenes, es aquí donde el programa buscará las imágenes a comparar.

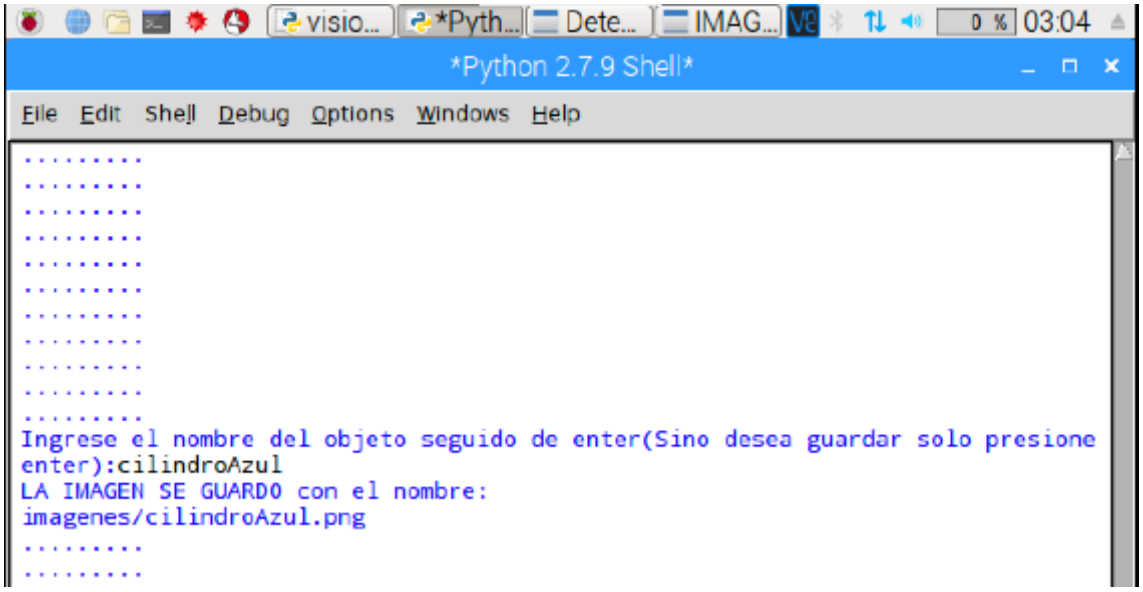


Figura 50. Programa guarda imagen

Con esto se tiene la referencia positiva de detección del cilindro de gas donde se validará su forma y se tomará la etiqueta del peso para ser identificada, este proceso el programa lo realizará automáticamente una vez obtenida una muestra del cilindro de gas, luego se creará las diferentes posiciones que puede tener el cilindro de gas al momento de pasar frente a la cámara.

Con esta configuración se validará que la detección de color y peso marcado se realizará correctamente, en este caso se realiza con un cilindro color azul para la explicación funcional del programa. En el capitulo de pruebas se evidenciará el reconocimiento con otros colores y pesos marcados en los cilindros de gas.

.

Figura 51. Detección de peso marcado y color de cilindro de gas

5. PRUEBAS

Se realizaron pruebas a partir de un prototipo a escala mientras se desarrollaba el prototipo para entender los conceptos de detección de objetos y reconocimiento de características como el color y etiqueta de peso marcado en cilindro.

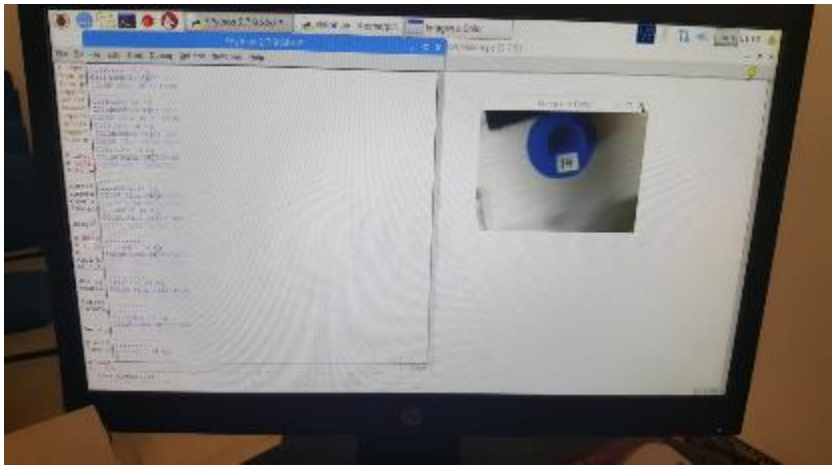


Figura 52. Reconocimiento de cilindro a escala color azul

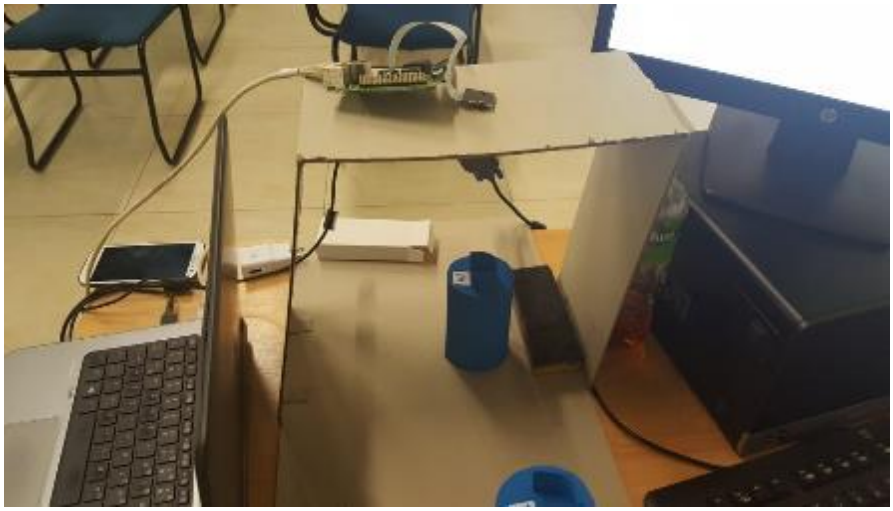


Figura 53. Raspberry Detectando Cilindro a escala color azul

Para comprobar la diferencia de color se coloca un cilindro color tomate, se verifica que detecta el color y el peso.

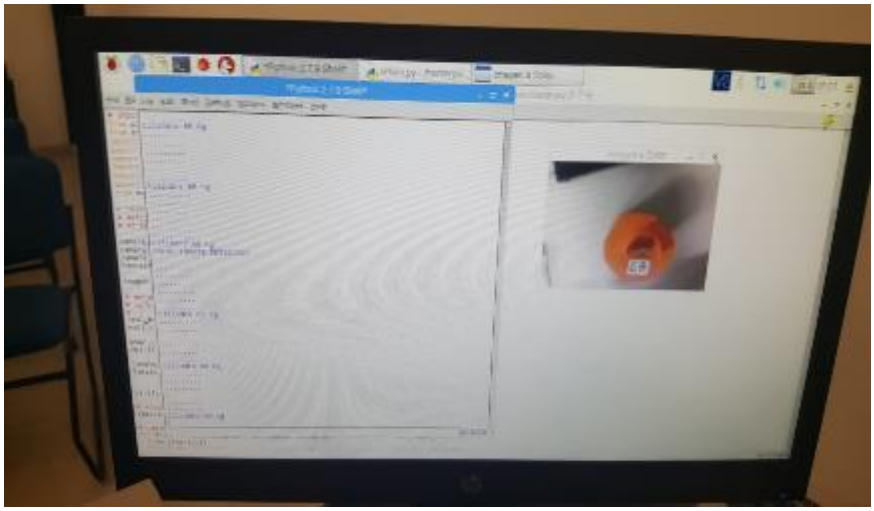


Figura 54. Reconocimiento de cilindro a escala color tomate

Luego de conseguir la parametrización de los códigos de colores RGB, se procederá a realizar pruebas con cilindros de gas reales.

Figura 55. Detección de Cilindro azul con programa

Figura 56. Reconocimiento de Cilindro gris con programa

5.1. Pruebas de luminosidad.

Cuando no existe luz suficiente en el objeto el programa no lo detecta.

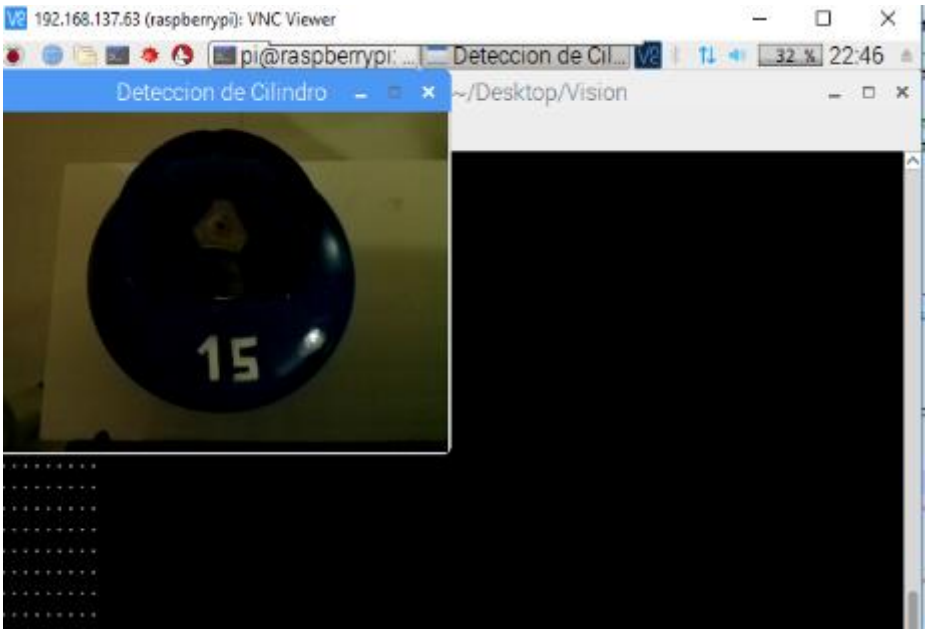


Figura 57. Cilindro sin iluminación

Se coloca luz blanca sobre el objeto y simplemente se logra detectar el color del cilindro de gas más no el peso. Esto ocurro ya que la etiqueta del peso no coincide con ninguna de las imágenes creadas en el repositorio de comparación, es por eso que parte importante de la prueba es crear más muestras de la etiqueta del peso con diferentes tamaños con el entrenamiento en cascada

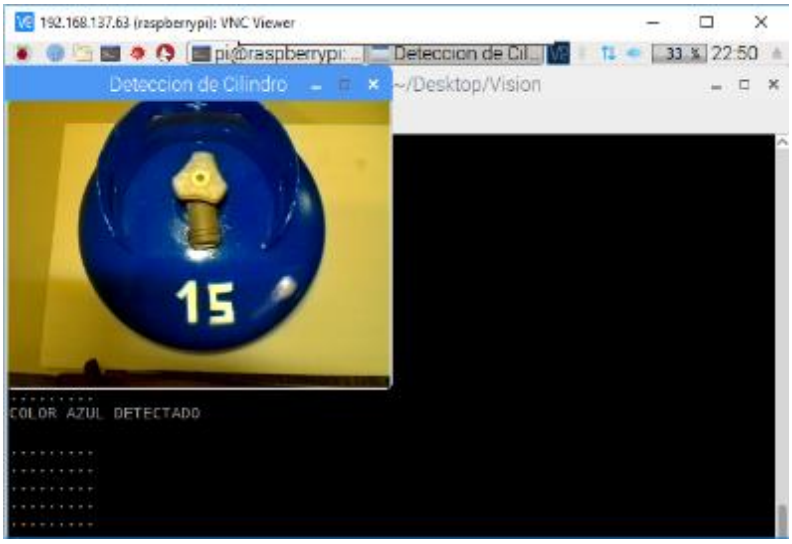


Figura 58. Detección de color en cilindro de gas

Se realiza una prueba después de crear más imágenes de comparación de la etiqueta del peso marcado y se logra obtener la detección del cilindro con su peso y color.

Figura 59. Detección de color y peso marcado en cilindro de gas

Finalmente, se comprobará con estas pruebas parámetros a controlar para que el programa de detección sea más preciso al momento de la detección del color y peso marcado en el cilindro de gas.

Uniendo todas estas correcciones se obtendrá la siguiente detección

Figura 60. Detección de Color y Peso Marcado en Cilindro de Gas

Como se puede observar se logra cumplir el objetivo del proyecto corrigiendo y controlando la iluminación y control del contraste del ambiente.

A continuación, se puede verificar que el programa detecta el peso marcado en diferentes posiciones que se encuentre.

Figura 61. Reconocimiento de Color y Peso.

5.2. Prueba con cilindros de diferente color.

Se realiza pruebas con el prototipo incluyendo diferentes colores de cilindros para validar que la detección sea solo en el espacio mientras pasa frente a la cámara, evitando el error de detección. Se configurará en el algoritmo para que solo detecte al cilindro de gas cuando este directamente enfocado por la cámara, reduciendo el error.

Figura 62. Reconocimiento de cilindro color amarillo

Figura 63. Reconocimiento de cilindro azul

Como resultado se obtendrá que los cilindros de gas pueden estar separados a una distancia mínima, pero siempre el programa detectara el cilindro de gas que este en las posiciones indicadas dentro del marco de captura de la cámara.

5.3. Errores

Existieron errores en el prototipo cuando no se lograba controlar el ambiente de detección, esto sucedía cuando el programa tomaba todo el marco de la imagen para detectar el objeto. La solución al problema fue asignar coordenadas fijas al marco de detección reduciendo el tamaño hasta cubrir un área de un cilindro de gas.

Figura 64. Programa sin control de posición.

Figura 65. Programa con control de posición.

5.4. Condiciones de funcionamiento

El prototipo logró cumplir las expectativas de funcionamiento en cuestión de luminosidad y ruido. No se requiere crear un ambiente aislado para su detección, se controló determinando posiciones fijas para detectar el objeto así se minimizó el margen de error. El prototipo puede estar bajo luz ambiente y reconocer el color sin ningún inconveniente ya que tiene configurado tonos bajos y altos de los respectivos colores a detectar.

5.4.1. Resultados

Se ha obtenido estos datos con el prototipo implementado.

Tabla 1.

Tiempos de detección de color.

Tiempos de detección de color		
Color	Tiempo	Ruido
Azul	1 segundo	no
Amarillo	1 segundo	no
Tomate	1 segundo	no

Tabla 2.

Tiempos de detección de Peso.

Tiempos de detección de peso		
Peso	Tiempo	Ruido
14.6 kg	1 segundo	no
14.6 kg	1 segundo	no
14.7 kg	1 segundo	no

Como conclusión se puede verificar que el algoritmo cumple con la detección en base a los tiempos de captura que están configurados en el programa. Este tipo de reconocimiento se base en relación con el tiempo de captura que tiene la cámara, por tal motivo este cumple su objetivo cada segundo.

6. CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

Al finalizar el proyecto propuesto, se implementó un prototipo que permite la detección de color y peso marcado en el cilindro de GLP, basándose en herramientas Open Source como Opencv y Python como lenguaje de programación, y utilizando como hardware principal el dispositivo Raspberry Pi.

La visión artificial es una herramienta que permitió controlar diferentes actividades que actualmente son supervisadas por el hombre, como se observó en el proyecto se puede detectar cilindros de gas con su color y peso marcado.

En el desarrollo del proyecto fue muy importante el conocimiento del lenguaje de programación Python y sus librerías para adaptarlo a la necesidad requerida.

La librería Opencv permitió conseguir la detección basándose en su método de detección por entrenamiento en cascada, creando diferentes opciones de comparación para alcanzar el reconocimiento.

Para poder utilizar la librería Opencv en un Raspberry Pi se necesitó una instalación específica para la versión de Python 2 y 3, esto se lo puede encontrar en la parte de anexos del proyecto, teniendo en cuenta que esta es la base principal para que el programa se ejecute sin ningún inconveniente.

Se verificó que sin importar la luminosidad y ruido se puede crear una detección visual precisa, ya que la librería Opencv permite controlar las características de la imagen en función a su luz y contraste.

Para la detección del cilindro de gas se entrenó el algoritmo con imágenes positivas y negativas que permitirán identificar si es un cilindro de gas u otro objeto.

6.2. RECOMENDACIONES

Como recomendación general es importante conocer el lenguaje de programación Python para poder crear modificaciones en base a lo requerido, con esto se dice que las librerías y lógica que exista pueden ser adaptadas para diferentes escenarios.

La librería open cv tiene varios métodos de reconocimiento de imágenes. Para este proyecto se utilizó la técnica *haar cascade*, por tal razón se aconseja leer bien las funciones y algoritmos existentes para entender bien la lógica de entrenamiento de imágenes.

Para obtener una mejor y más amplia detección, se aconseja utilizar servidores en la nube que prestan su servicio de procesamiento de imágenes tales como Microsoft, Amazon y Google.

Existen métodos en open cv que le permiten controlar diferentes escenarios de luminosidad y ruido ya que es el principal error al momento de realizar las pruebas con objetos en luz ambiente.

Para mejorar la resolución de la imagen se puede adaptar en el mismo programa el uso de una cámara web con mejor resolución.

El proyecto utilizo la última versión de Raspbian para asegurar que el prototipo se encuentre actualizado y así la compatibilidad con las librerías de open cv no tengan inconveniente al momento de ser ejecutadas.

En caso de que no se detecte el número del peso marcado en el cilindro de gas, se debe realizar nuevamente el paso de parametrización de peso, recortando la imagen en la posición no detectada y agregar en el codigo. Este proceso se realizará al 100 % con todas las opciones que existan con el algoritmo de fase de entrenamiento que lo realizará automaticamente.

REFERENCIAS

Cascade Classification (2019). Recuperado el 10 de Octubre de 2019 de https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html

Html Color Codes. (s.f.). Recuperado el 5 de Noviembre de 2019 de https://htmlcolorcodes.com/es/

Imagen Integral (s.f). Recuperado el 10 de Octubre de 2019 de https://es.coursera.org/lecture/deteccion-objetos/l5-3-imagen-integral-CROQ4

Imaginghub. (s.f.). Recuperado el 10 de Octubre de 2019 de https://imaginghub.com/projects/144-installing-opencv-3-on-raspberry-pi-3?gclid=CjwKCAiAz7TfBRAKEiwAz8fKOIEYcwcdm-47J052VtGen7zRCyndbij1rz4O4eDY5iuaCQes6vaOChoCgtoQAvD_BwE#documentation

Integralimage (s.f) Recuperado el 10 de Octubre de 2019 de https://es.mathworks.com/help/images/ref/integralimage.html

Jason Brownlee. (2016) Boosting and AdaBoost for Machine Learning. Recuperado el 8 de Octubre de 2019 de https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/

Nolasco Valenzuela, J. S. (2018). *Python: Aplicaciones practicas.* España: RA-MA Editorial.

Numpy. (s.f.). Recuperado el 27 de DIciembre de 2019 de http://www.numpy.org/

Open CV. (s.f). Recuperado el 27 de DIciembre de 2019 de https://opencv.org/

Opencv documentation. (s.f.). Recuperado el 9 de Agosto de 2019 de https://docs.opencv.org/master/dc/d88/tutorial_traincascade.html

Picamara. (s.f.). Recuperado el 27 de DIciembre de 2019 de https://picamera.readthedocs.io/en/release-1.13/

Pypi Org. (s.f.). Recuperado el 27 de DIciembre de 2019 de https://pypi.org/project/glob2/

Python. (s.f.). Recuperado el 28 de DIciembre de 2019 de https://www.python.org/

Raspberry Pi. (s.f.). Recuperado el 27 de DIciembre de 2019 de https://www.raspberrypi.org/documentation/remote-access/vnc/

Raspberry Pi. (s.f.). Recuperado el 27 de DIciembre de 2019 de https://www.raspberrypi.org

Reglamento Actividades de Comercialización Gas Licuado de Petróleo. (2017). Recuperado el 27 de DIciembre de 2019 de https://www.controlhidrocarburos.gob.ec/biblioteca/

RS. (s.f.). Recuperado el 12 de DIciembre de 2019 de https://uk.rs-online.com/web/generalDisplay.html?id=raspberrypi&src=raspberrypi

Viola, Paul & Jones, Michael. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE Conf Comput Vis Pattern Recognit. 1. I-511. 10.1109/CVPR.2001.990517.

ANEXOS

ANEXO A

Instalación de Sistema Operativo en Raspberry Pi.

Es necesario instalar un sistema operativo en el raspberry por lo que para este caso se escoge Debian, ya que es propio de raspberry y trae preinstalados programas de desarrollo con son Python.

A continuación, una breve explicación de la instalación.

Se descarga imagen iso de Debian Strech desde la página oficial de raspberry.

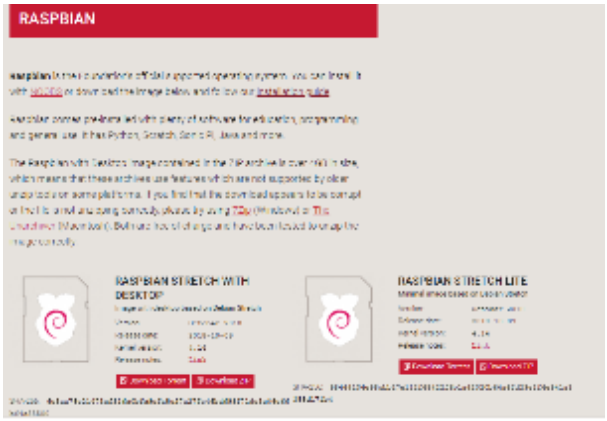


Figura 66. Tipos de Sistemas Raspbian

Tomado de (Raspberry Pi, s.f.)

Para cargar la imagen ISO en la tarjeta de memoria SD es necesario formatear con el programa SD Card Formatter.

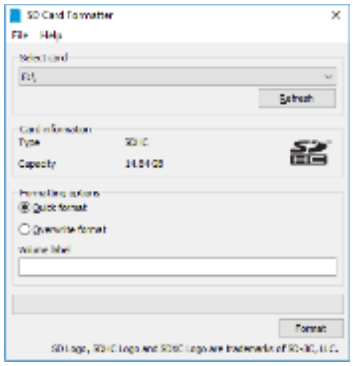


Figura 67. Programa SD Card Formatter

En el programa Win32 Disk Imager se carga la imagen ISO de Debian, de esta forma la memoria SD quedara con un formato booteable.

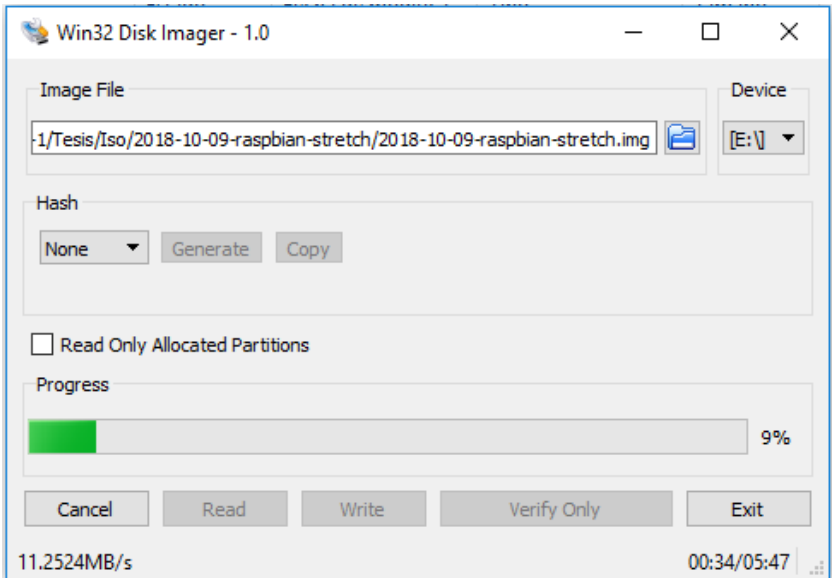


Figura 68. Programa Win32 disk imager

Una vez se termine de copiar los archivos de la imagen se inserta la memoria en el Raspberry pi y se conecta el Raspberry a la fuente de alimentación de 5V.

También se debe conectar el Raspberry al cable de red del router, con esto unimos al dispositivo a la red local.

Luego conectar por medio de SSH, para esto se realiza un escaneo de la red con la herramienta IP advance.

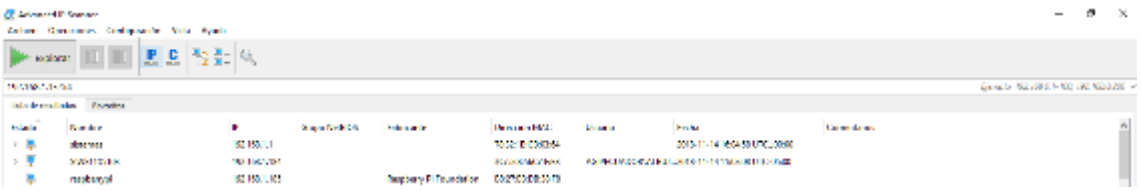


Figura 69. Herramienta IP Advance

Con el programa Putty se coloca la IP que refleja en el escaneado de la red, esta nos permite ingresar al Raspberry por SSH.

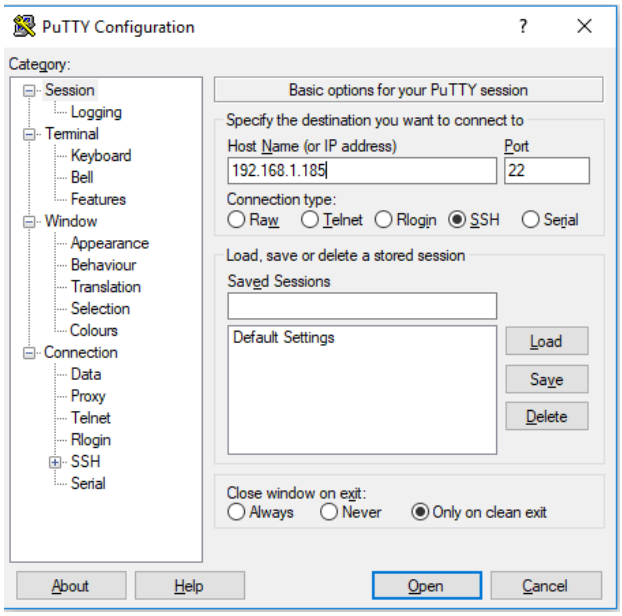


Figura 70. Programa Putty

Se ingresa en campor de usuario : pi y contraseña: raspberry, con esto ya se tiene acceso al raspberry por consola.

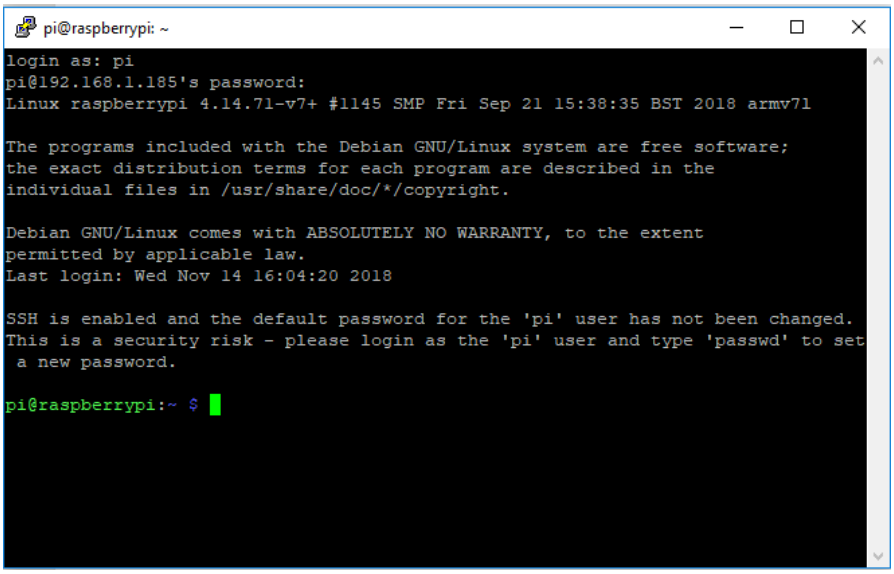


Figura 71. Acceso por consola a raspberry PI.

ANEXO B

Instalación de VNC server

Es necesario acceder al Raspberry Pi en modo gráfico de manera remota para así visualizar el video de la cámara desde la PC. Para ello se debe instalar el paquete VNC server.

Primero se realiza una actualización de todos los paquetes con el comando sudo apt-get update.

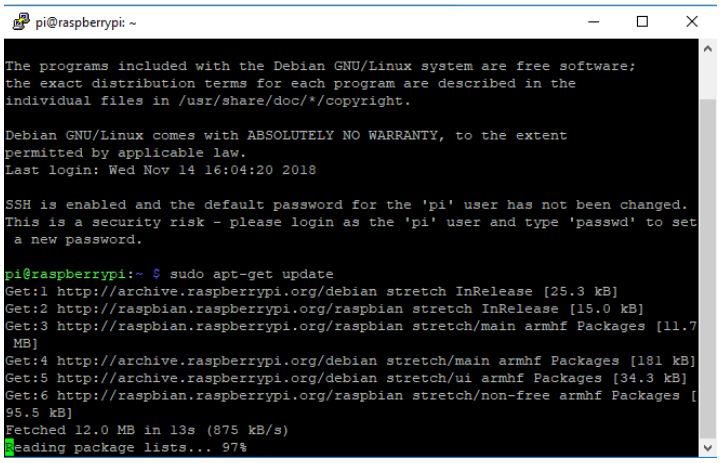


Figura 72. Actualización de paquetes en Debian.

Una vez actualizados los paquetes se instala el paquete VNC server con el comando: sudo apt-get install realvnc-vnc-server realvnc-vnc-viewer

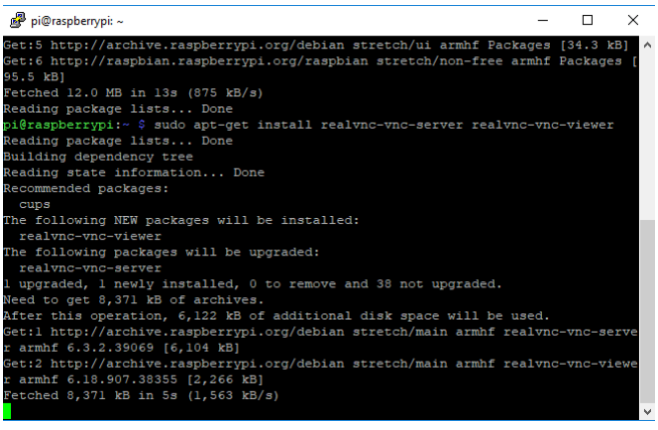


Figura 73. Instalación de paquete VNC server

Para activar VNC se ingresa el comando sudo raspi-config, de esta forma se ingresa a la configuración del Raspberry y se configura las opciones de interfaz.

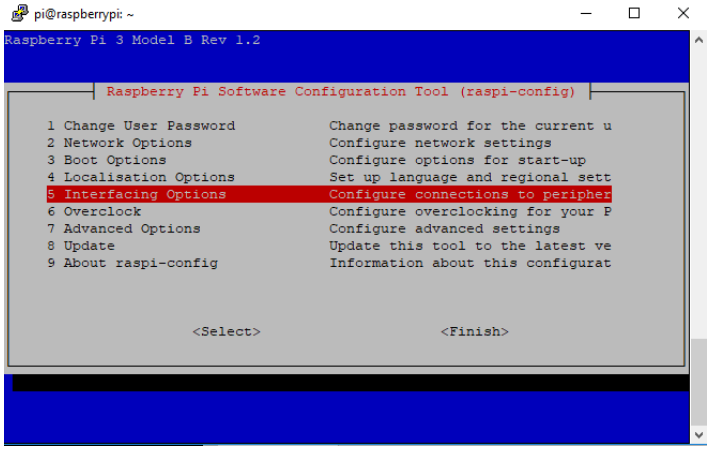


Figura 74. Pantalla de configuración de herramientas de Raspberry Pi.

Se activa para opción de VNC para obtener la conexión sin problemas.

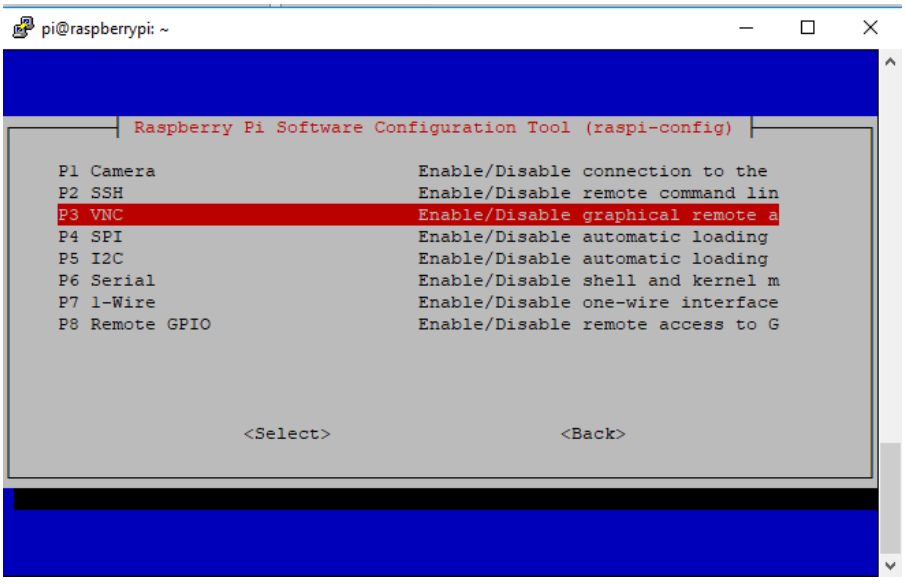


Figura 75. Activación de VNC.

Finalmente, despues de la configuración previa en el rapberry pi, se debe instalar en la PC el programa VNC viewer para realizar la conexión con el dispositivo.

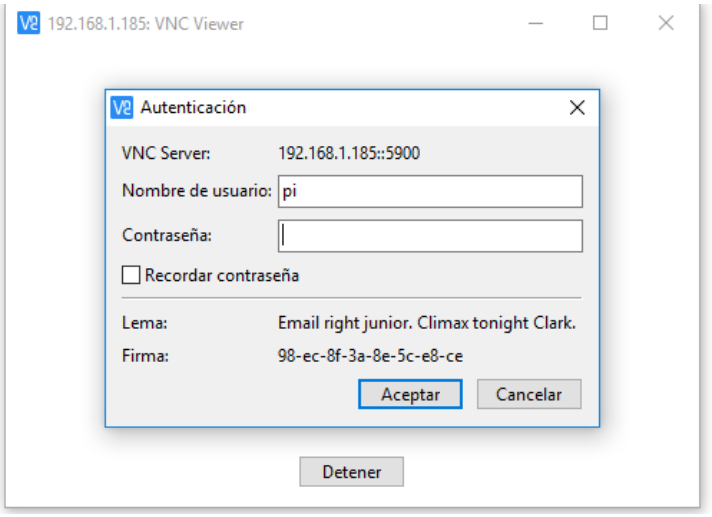


Figura 76. Ventana de conexión desde PC.

De esta forma se obtiene acceso remoto de forma gráfica al Raspberry Pi.

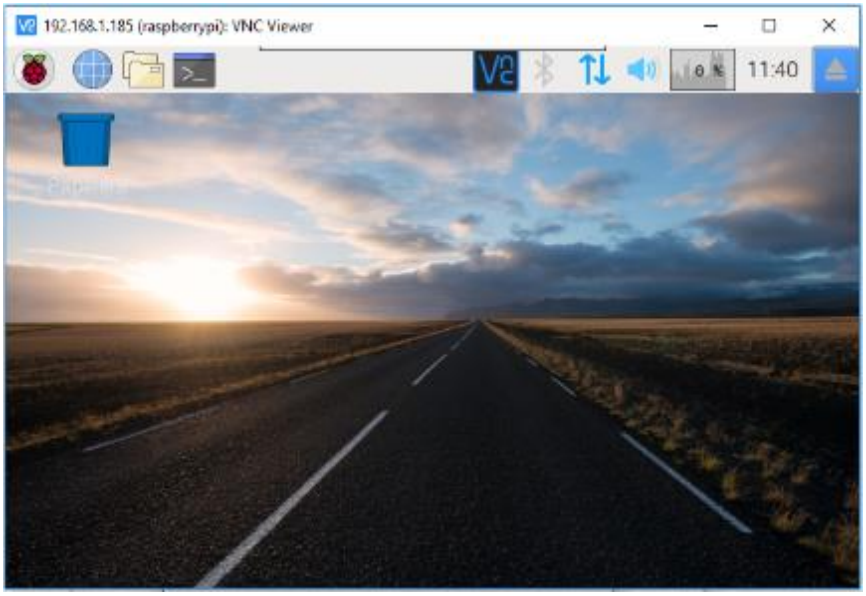


Figura 77. Conexión remota a Raspberry Pi.

!
!
!