Mamani, Mauricio Gabriel

Tecnologías de IoT, visión por computadora y robótica para la manipulación de objetos peligrosos

2022

Instituto: Instituto de Ingeniería y Agronomía

Carrera: Ingeniería en Informática

Informe final de la práctica profesional supervisada

Tecnologías de IoT, visión por computadora y robótica para la manipulación de objetos peligrosos.

INGENIERIA EN INFORMATICA - AÑO 2022

MAURICIO GABRIEL MAMANI

Universidad Nacional ARTURO JAURETCHE

DATOS DEL ESTUDIANTE

Apellido y Nombres: Mamani, Mauricio Gabriel

DNI: 38709581

Nº de Legajo: 8721

Correo electrónico: mauri.carp2027@gmail.com

Cantidad de materias aprobadas al comienzo de la PPS: 41

PPS enmarcada en el artículo (4 ó 7) de la Resolución (CS) 183/21.

DOCENTE SUPERVISOR

Apellido y Nombres: Prof. Ing. Salvatore Juan

Correo electrónico: jsalvatore@unaj.edu.ar

DOCENTE TUTOR DEL TALLER DE APOYO A LA PRODUCCIÓN DE TEXTOS ACADÉMICOS DE LA UNAJ

Apellido y Nombres: Prof. Kelly Carolina

Correo electrónico: kellygcarolina@gmail.com

DATOS DE LA ORGANIZACIÓN DONDE SE REALIZA LA PPS

Nombre o Razón Social: Universidad Nacional Arturo Jauretche

Dirección: Av. Calchaquí 6200, Florencio Varela, (1888) Buenos Aires, Argentina

Teléfono: +54 11 4275-6100

Sector: Programa Tecnologías de la Información y la Comunicación (TIC) en aplicaciones de interés social, Instituto de Ingeniería y Agronomía

TUTOR DE LA ORGANIZACIONAL

Apellido y Nombres: Prof. Mg. Osio, Jorge

Correo electrónico: josio@unaj.edu.ar

COORDINADOR DE LA CARRERA DE INGENIERÍA EN INFORMÁTICA

Nombre y apellido: Dr. Ing. Morales, Martín

Correo electrónico: martin.morales@unaj.edu.ar

Resumen

La presente práctica profesional supervisada se realizó con el propósito de introducir al alumno en temas relacionados a tecnologías de Internet de las cosas (IoT), la visión por computadora y robótica para la manipulación de objetos peligrosos a través de la investigación y el desarrollo de la misma práctica. Para ello se creará un sistema capaz de reconocer y procesar imágenes junto con el cual se podrá identificar de manera autónoma los objetos peligrosos y este podrá actuar. Por otro lado, el sistema deberá ser capaz de monitorear en tiempo real, tener acceso y control remoto del brazo robótico.

El desarrollo de la práctica se realizó mediante metodologías de trabajo rápidas o ágiles. en cual se probaban de manera rápida nuevas funcionalidades al sistema.

Se obtuvieron buenos resultados conformes a los objetivos planteados para esta práctica, se pudo abordar todos los temas de investigación y desarrollar el sistema final en funcionamiento.

Palabras claves: Internet de las cosas - visión por computadora - robótica - control remoto

Abstract

This supervised professional practice was carried out with the purpose of introducing the student to topics related to Internet of Things (IoT) technologies, computer vision and robotics for the manipulation of dangerous objects through

research and development of the same practice. For this, a system capable of recognizing and processing images will be created, together with which dangerous objects can be identified autonomously and it will be able to act. On the other hand, the system must be able to monitor in real time, have access and remote control of the robotic arm.

The development of the practice was carried out using fast or agile work methodologies. in which new functionalities to the system were quickly tested.

Good results were obtained in accordance with the objectives set for this practice, it was possible to address all the research topics and develop the final system in operation.

Keywords: Internet of things - computer vision - robotics - remote control

Dedicatoria y agradecimientos

Quiero agradecer a la Universidad Nacional Arturo Jauretche por permitirme ser parte de su comunidad y poder desarrollarme como estudiante y profesional. También quiero agradecer a mis tutores Jorge Osio, Juan Salvatore, Carolina Kelly de la práctica profesional supervisada (PPS) por la ayuda en el desarrollo y a Martín Morales por la organización de la misma. Además de agradecer a todos mis compañeros, amigos y profesores durante estos años de cursada y aprendizaje juntos.

Dedicar especialmente a mi familia y a aquellas personas cercanas que siempre me estuvieron acompañando y apoyando durante esta etapa de mi vida.

Contenido

4. Introducción

La presente práctica profesional supervisada (PPS) surge como proyecto en el marco de investigación propuesta por la Universidad Nacional Arturo Jauretche (UNAJ). Consistirá en el desarrollo e implementación de un sistema de control de un brazo robótico guiado mediante la visión artificial y control remoto.

4.1. Objetivos e implementación.

El objetivo de esta práctica es unir o hacer un acercamiento de la robótica y la visión por computadora, o digital, con la finalidad de implementar un sistema capaz de realizar determinadas tareas mediante un brazo robótico y de ser controlado remotamente. Por lo tanto, el sistema robótico deberá emplear la visión por computadora, a través de la cámara, para la detección de objetos fijos, que estén determinados como peligrosos o que requieran cierta precaución para manipularlos y permitir su manipulación mediante el control remoto. Este tendrá como parte de su tarea principal la detección, en la imagen, y posterior captura del objeto para luego trasladarse hacia la ubicación de destino. Al mismo tiempo, se desarrollará un sistema de control y monitoreo del robot.

La implementación de este proyecto se realizará mediante el uso de la librería para el procesamiento de imágenes, visión artificial y con funciones específicas necesarias, llamada *OpenCV*, para el sistema de control y monitoreo estará a cargo del kit *ESP32 CAM* (con wifi) el cual tendrá la función de ser el “cerebro” del robot para indicar acciones y movimientos realizados por el robot. Este módulo de desarrollo está conectado a una cámara, para obtener las imágenes en directo y además está conectado también al brazo robótico, el cual cuenta con una

estructura capaz de realizar los movimientos necesarios para mover objetos

4.2. Objetivos generales.

- Investigar el uso del procesamiento de imágenes y visión artificial en ciencias de la computación.

- Desarrollar un software o sistema que emplee técnicas de procesamiento de imágenes con el fin de proveer cierta autonomía a un brazo robótico en un entorno determinado.

- Desarrollar una aplicación que permita el monitoreo y control remoto en tiempo real del brazo para poder manipular los objetos.

4.3. Objetivos específicos

Para los objetivos específicos se determinaron y consideran los siguientes puntos:

- Evaluar las distintas alternativas de procesamiento de imágenes para visión artificial (librería *OpenCV, Deep Learning* para aplicaciones en visión por computadora, entre otras).

- Sistematizar la información referente a brazos robóticos y aplicaciones tecnológicas.

- Analizar las diferentes funciones y librerías disponibles para el control de un brazo robótico.

- Diseñar un sistema de control que interactúe con los sensores y manipule los mecanismos y motores del robot.

- Analizar las diferentes funciones de *OpenCV* que se van a utilizar para el procesamiento de imágenes.

- Desarrollar el software en lenguaje Python o C que sea capaz de procesar las imágenes y obtener los correspondientes datos a través de los cuales recibirá instrucciones de movimiento y dirección del brazo.

- Desarrollar una aplicación híbrida en un lenguaje de programación web o local que permita monitorear y controlar la dirección del brazo robótico de forma remota mediante el acceso a la cámara y a los comandos del robot para controlar sus cuatro movimientos básicos.

- Realizar distintas pruebas de funcionamiento del brazo y la aplicación de control y monitoreo para validar la propuesta realizada.

4.4. Tareas a ejecutar

1. investigar, analizar y estudiar conceptos de procesamiento de imágenes, sus diferentes técnicas y algoritmos. Para ello se estudiarán los diferentes conceptos y técnicas que se aplican en el procesamiento de imágenes para luego analizar las diferentes funciones que provee *OpenCV.* Se pondrá especial énfasis en aquellas que se utilizan para filtrar el ruido y/o colores, detectar bordes (cambios bruscos de intensidad lumínica), binarizar imágenes, etc.

2. Desarrollar un software que sea capaz de procesar las distintas imágenes obtenidas por una cámara de video y actuar en consecuencia. Se busca desarrollar un sistema que sea capaz de obtener imágenes en tiempo real, a través de una cámara web conectada al ESP32, procesarlas con *OpenCV* y obtener datos que ayuden al sistema a decidir que movimiento realizar. Dichas instrucciones serán entregadas al módulo de control para que accione y/o apague los correspondientes motores del brazo robótico.

3. Desarrollar una aplicación híbrida de monitoreo y control del robot. Se pretende desarrollar una aplicación híbrida que sea capaz de ejecutarse en cualquier dispositivo móvil o computadora. La misma debe permitir monitorear y controlar al robot de forma remota, ya sea para indicarle que se mueva de una posición a otra, o bien darle las instrucciones necesarias para realizar alguna acción que no está automatizada. Debe proveer acceso a la cámara en directo y a los comandos del mismo.

4. Probar los sistemas desarrollados. Se probará el sistema de visión artificial para la detección de objetos por separado. Este permitirá obtener la ubicación de un objeto dentro de la imágen. Por otro lado se realizarán diferentes pruebas para comprobar que el robot puede ser monitoreado y manejado remotamente desde la aplicación.

5. Obtención de resultados y redacción del informe final. Se pretende listar y catalogar todos los resultados obtenidos a lo largo del presente proyecto para su posterior exposición en un informe final que sintetice todo el conocimiento adquirido a lo largo de cada etapa del proyecto con sus progresos respectivos y datos correspondientes.

5. Herramientas de hardware y software

5.1. Espressif

Espressif System (688018.SH) es una empresa multinacional pública de semiconductores sin una fábrica establecida, con oficinas en China, República Checa, India, Singapur y Brasil. Es la creadora de populares series de chips, módulos y placas de desarrollo, como ESP8266, ESP32, ESP32-S, ESP32-C y ESP32-H. La empresa se ha comprometido y dedicado a la investigación y desarrollo de soluciones con mayor rendimiento, haciendo sus equipos más robustos, seguros, adaptables, con mayor conectividad y rentables. Esto trae como consecuencia que, gracias a su hardware y su código abierto, sean elegidos cada vez más para el desarrollo de soluciones de Internet of Things (IoT) y Artificial Intelligence of Things (AIoT).

5.2. ESP32

Es un microcontrolador (MCU) con Wifi integrado y conectividad Bluetooth con altas prestaciones y de bajo costo. Al poseer un diseño robusto es capaz de funcionar fiablemente en entornos industriales o desfavorables.

Posee un diseño robusto capaz de funcionar en la industria de manera fiable. Expuesto a condiciones extremas cuenta con la capacidad de resolver de forma dinámica efectos indeseados sobre el hardware y soporta temperaturas entre los -40 °C y los +125 ° C.

Debido a que fue diseñado para dispositivos móviles, electrónicos y portátiles con aplicaciones en Internet de las Cosas (IoT), posee un consumo de energía ultra bajo y cuenta con varios modos de funcionamiento en bajo consumo.

Esp32 está integrado por antenas incorporadas, balun de RF (dispositivo conductor que convierte líneas de transmisión desbalanceadas en líneas balanceadas), amplificadores de potencia, amplificador de recepción de bajo ruido y módulo para el suministro de energía. Además de lo mencionado anteriormente, cuenta con gran versatilidad ya que posee mínimos requisitos para su *Printed Circuit Board* (PCB). También puede interactuar con otros sistemas y aplicaciones, proporcionando funcionalidades de Wifi y bluetooth por medio de sus interfaces SPI/SDIO o I2C/UART.

ESP32-CAM

Es una placa de desarrollo basada en ESP32 pequeña y de bajo costo. La placa integra Wifi, Bluetooth tradicional y bluetooth de bajo consumo (BLE), con 2 CPU LX6 de 32 bits y alto rendimiento. Adopta una arquitectura de procesamiento con pipeline de 7 etapas, sensor en chip, sensor Hall, sensor de temperatura, etc. Además, su frecuencia de clock principal puede varíar de 80MHz a 240MHz.

Es totalmente compatible con los estándares Wifi 802.11b / g / n / e / i y Bluetooth 4.2, se puede utilizar en modo maestro para construir un controlador de red independiente o como esclavo de otras MCU host para agregar capacidades de red a los dispositivos existentes. En la Figura 1 se muestra el pinout del kit ESP32-CAM.

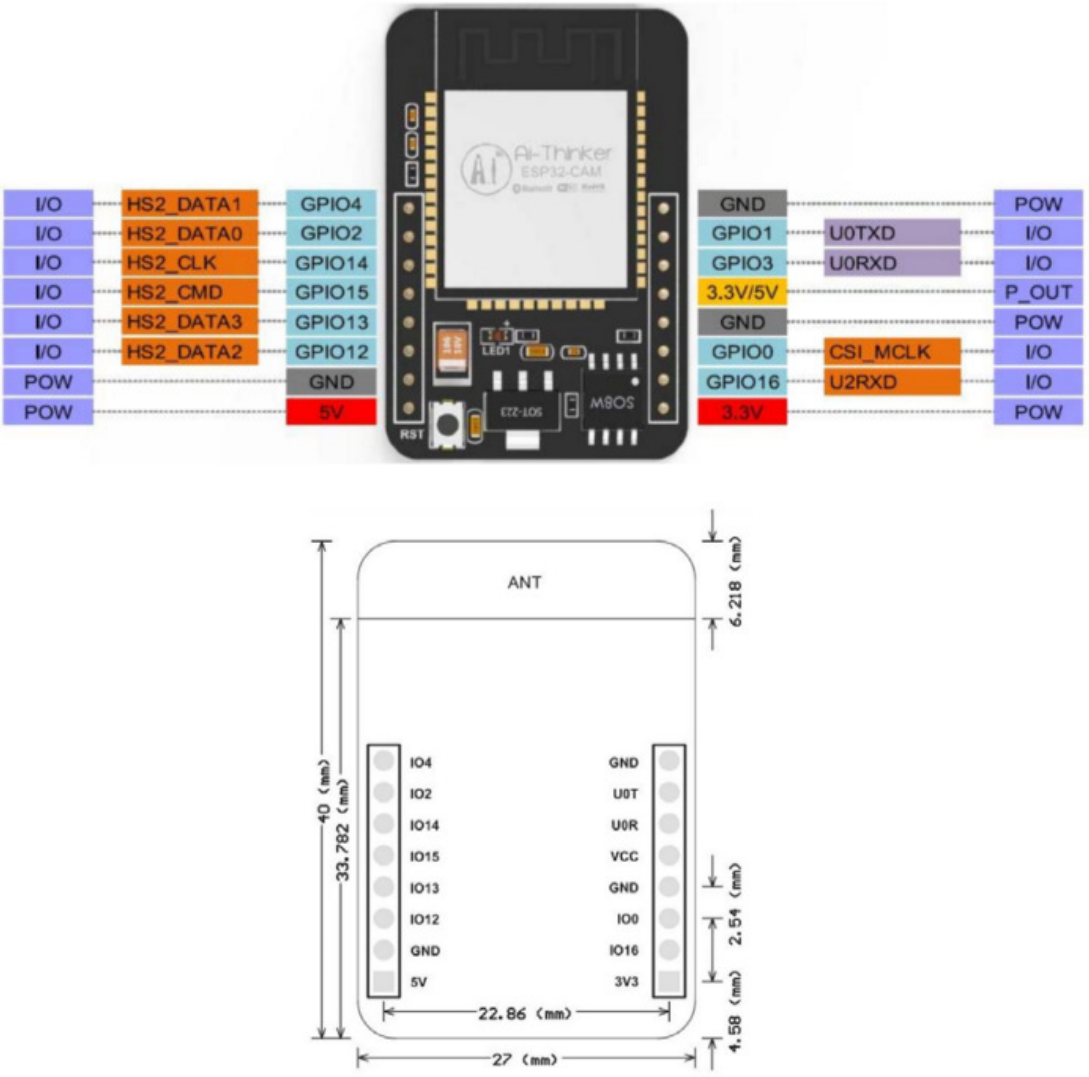


Figura 1. Imagen de pines de entrada/salida del módulo ESP32-CAM

5.3. Arduino IDE y MCU Arduino

Es una plataforma electrónica de código abierto (Open Source) que cuenta con hardware y software fáciles de usar, es decir, para poder realizar proyectos de manera sencilla.

Arduino fue creado el año 2005, en Italia, como un proyecto de estudiantes que buscaban una posibilidad más económica para realizar sus investigaciones y desarrollo en el instituto.

Gracias a la gran versatilidad y poder de adaptación que Arduino posee para crear proyectos, importar o bien adaptarlos como complementos de otros, podemos separarlos en dos grupos:

-Arduino como microcontrolador es utilizado cuando tiene un programa cargado desde la computadora y funciona de forma independiente. Este, de acuerdo al programa que se le cargue, controla y alimenta otros dispositivos que están conectados con él para interactuar con el medio físico.

-Arduino como interfaz, hace de mediador entre la computadora y otro dispositivo, para que este pueda interactuar con el mundo físico y ejecutar una tarea determinada.

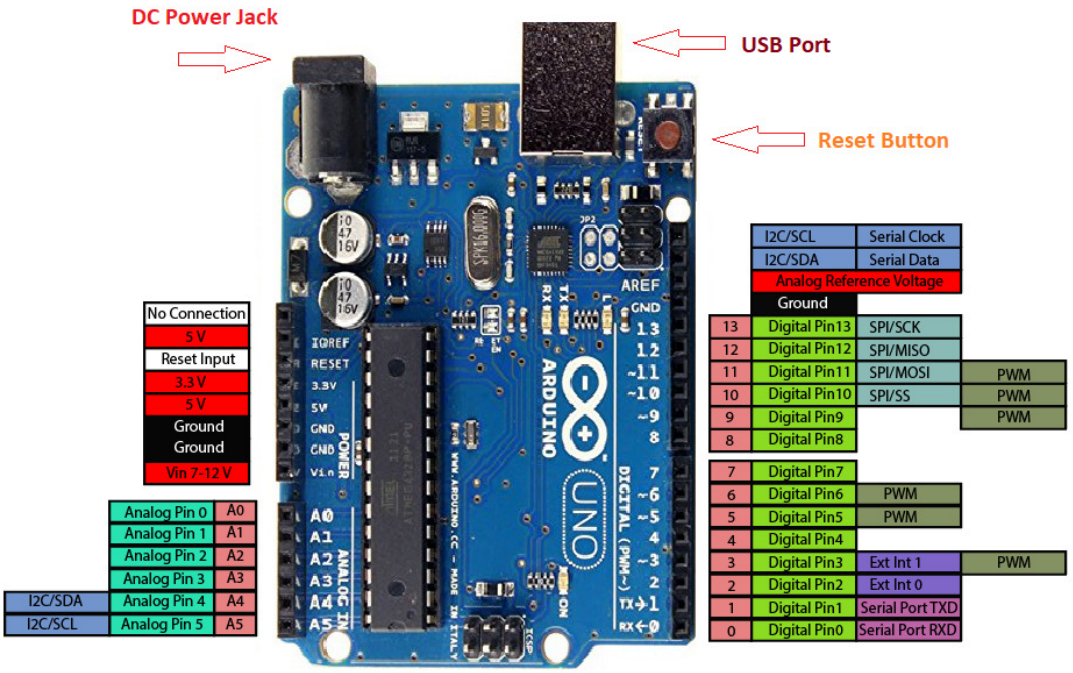


Figura 2. Pines de entrada y salida Arduino UNO R3

Básicamente, Arduino es una placa basada en un microcontrolador ATMEL. Los microcontroladores son circuitos integrados en los que se pueden grabar instrucciones, las cuales las escribes con el lenguaje de programación que puedes utilizar en el entorno Arduino IDE. Estas instrucciones permiten crear programas que interactúan con los circuitos de la placa.

El microcontrolador de Arduino posee lo que se llama una interfaz de entrada, que es una conexión en la que podemos conectar en la placa diferentes tipos de periféricos. La información de estos periféricos que conectes se trasladará al microcontrolador, el cual se encargará de procesar los datos que le lleguen a través de ellos.

El tipo de periféricos que puedas utilizar para enviar datos al microcontrolador depende en gran medida de qué uso le estés pensando dar. También, cuenta con una interfaz de salida, que es la que se encarga de llevar la información que se ha procesado en el Arduino a otros periféricos. Existen diferentes tipos de placas y cada una está diseñada para diferentes proyectos y alcance, con el cual cada uno decide cual es más conveniente utilizarla.

5.4. Python

Es un lenguaje de programación de alto nivel, multiplataforma, de código abierto con el cual se pueden desarrollar todo tipo de aplicaciones. Fue fundado por Guido Van Rossum a principio de los años 90 en Holanda. Su desarrollo se basó en el lenguaje de programación “ABC”, con el cual compartían la filosofía de que sea fácil de aprender, escribir y entender. Debido a su legibilidad en código, es uno de los más elegidos a nivel mundial y también a la hora de aprender. Posee una curva de aprendizaje muy rápida y una comunidad muy grande en todo el mundo que además de la documentación oficial aporta a su evolución continua. Gracias a su facilidad y flexibilidad para desarrollar aplicaciones en cualquier sistema operativo, se introdujo en temas que son muy demandados actualmente como Inteligencia

artificial, Big Data, Machine learning, Data Science, desarrollo web, desarrollo de juegos y muchos otros campos más.

5.5. OpenCV

Open Source Computer Vision (OpenCV) es la biblioteca de código abierto, libre y gratuita más grande en lo que refiere a funciones para visión por computadoras y machine learning del mundo. Comenzó como un proyecto de investigación de Intel, lanzando su primera versión 1.0 a comienzos del 2006. Actualmente, posee más de 2500 algoritmos implementados con soporte para múltiples lenguajes como lo son Python, Java, C++, etc. y múltiples plataformas, como Windows, Linux, Android, OS X y iOS.

5.6. Numpy

Es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas) y una variedad de rutinas para operaciones rápidas en matrices, es decir, que incluyen manipulación matemática, lógica, de formas, clasificación, selección, E/S., transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria, etc.

5.7. Brazo robótico.

Hiwonder Co. Ltd, fundada en 2015 en Shenzhen, China y es un proveedor de soluciones *integrales para Science*, Technology, Engineering and Mathematics (*STEAM) Education*. Ofrece productos y servicios integrales para el desarrollo de hardware, software y diseño curricular para todas las edades. Es poseedor de más de cuarenta patentes, hardware de código abierto, robot biónico inteligente, brazos robóticos, kits IoT, robótica de inteligencia artificial, y muchos otros desarrollos más. Actualmente, produce y brinda servicios de ventas en más de ochenta países.

LeArm es un brazo robótico inteligente de escritorio de alto rendimiento destinado a la educación. Posee 6 servos que le permiten agarrar en cualquier dirección. El brazo robótico también brinda soporte para programar aplicaciones en PC y para teléfonos móviles.

Está constituido por una construcción totalmente metálica: garra de metal, soporte de aluminio y una base más grande de metal en la parte inferior. El diseño estructural 6DOF puede hacer que el brazo robótico se mueva de manera flexible, para que pueda tomar objetos en cualquier dirección. Fue implementado con servos digitales de alta precisión, que hace que el control sea más preciso. Utiliza un servo controlador de 6 canales con un módulo bluetooth 4.0 incorporado. Cuenta con varios métodos de control, proporcionados por un mando inalámbrico (joystick PS3), un software de escritorio para pc y una aplicación Android/iOS gratuita para dispositivos móviles inteligentes. También puede controlarse y programarse, lo que se hizo en este proyecto, a través de los pines Tx y Rx con aplicaciones ejemplo llamadas software serial (implementa el protocolo serie por software) y hardware serial (implementa el protocolo serie mediante el módulo de HW).

Figura 3. Brazo Robótico LeArm 6DOF

6. Desarrollo de la práctica.

6.1. Investigación

Para realizar el proyecto es necesario concentrarnos en tres grandes bloques que conforman la sistematización para llevarlo a cabo. Un primer bloque de investigación a la parte visual, otro bloque en lo que hace referencia al hardware y, por último, el bloque de comunicación entre el ESP-CAM y el brazo. Antes de empezar con la práctica se debe investigar, analizar, comparar y buscar cualquier información relevante para lo que refiere a imágenes, imágenes digitales, procesamiento de imágenes y de visión por computadora. Es fundamental tener claro los conceptos previos para pensar en la forma y resolución de la problemática.

Una imagen según la Real academia española (RAE) podemos definirla como la representación, figura, semejanza y apariencia de algo, es decir, es una representación visual por medio de alguna técnica.

Una imagen digital no es más que una imagen en dos dimensiones, una matriz estándar de Numpy, que contiene píxeles de puntos de datos. Basados en el número de píxeles, cuanto mayor tenga una imagen, mejor es su resolución. Podemos pensar a los pixeles como pequeños bloques de información puestos en una cuadrícula 2D, donde la profundidad de un píxel hace referencia a la información del color presente en ella.

Una computadora solo puede procesar números binarios, para ello se debe transformar la imagen a una imagen binaria. Para determinar o calcular el color de una imagen se lo define en:

Número de colores o sombras =2^bpp (bpp=bits por pixel)

Teniendo en cuenta esta relación podemos determinar que cuantos más píxeles tengamos mayor es su representación en colores en la imagen.

Bits/pixel	Colores posibles
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
8	$2^8 = 256$
16	$2^{16} = 65000$

Imagen binaria

Una imagen binaria consta de un 1 bit/pixel, como la representación de una imagen en blanco y negro en forma de binaria donde '1' representa blanco puro mientras que '0' representa negro. La siguiente imagen está representada por 1 bit/píxel, lo que significa que la imagen sólo puede representarse con 2 colores posibles, ya que 2^1=2

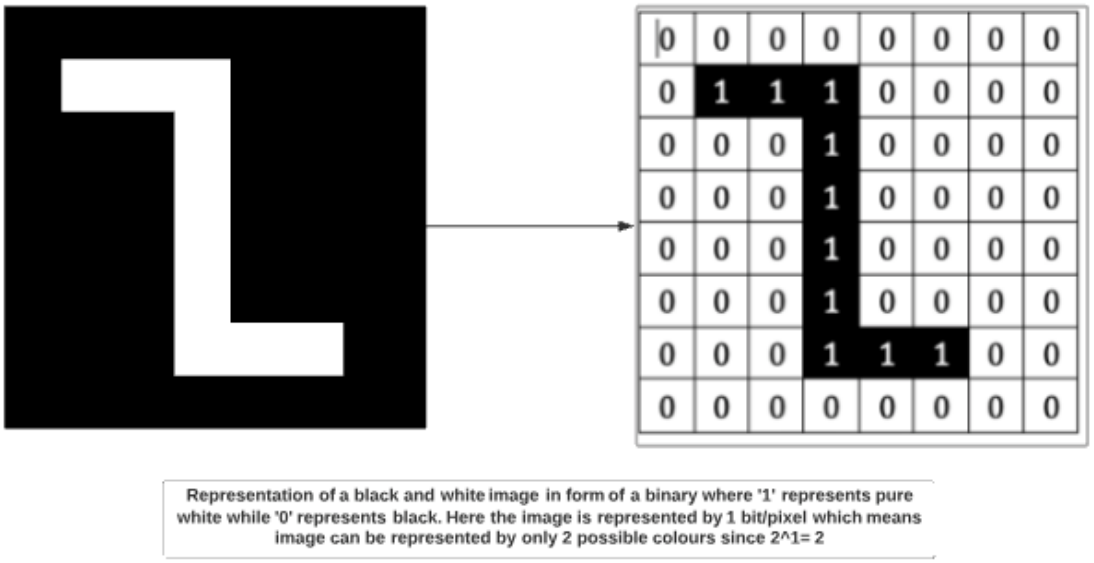


Figura 4. imagen de colores posibles de 1 bit/pixel

Una imagen puede estar definida en escala de grises, estas imágenes constan de 8 bits por pixel. De esta forma podemos decir que una imagen puede tener 256 grises o sombras diferentes, con cero (0) el negro y 255 el blanco.

Al igual que en la escala de grises una imagen a color también cuenta con 8 bits por pixel, es decir que en este caso podemos representar 256 colores diferentes. Los colores pueden ser representados por medio de la combinación correcta de rojo, verde y azul.

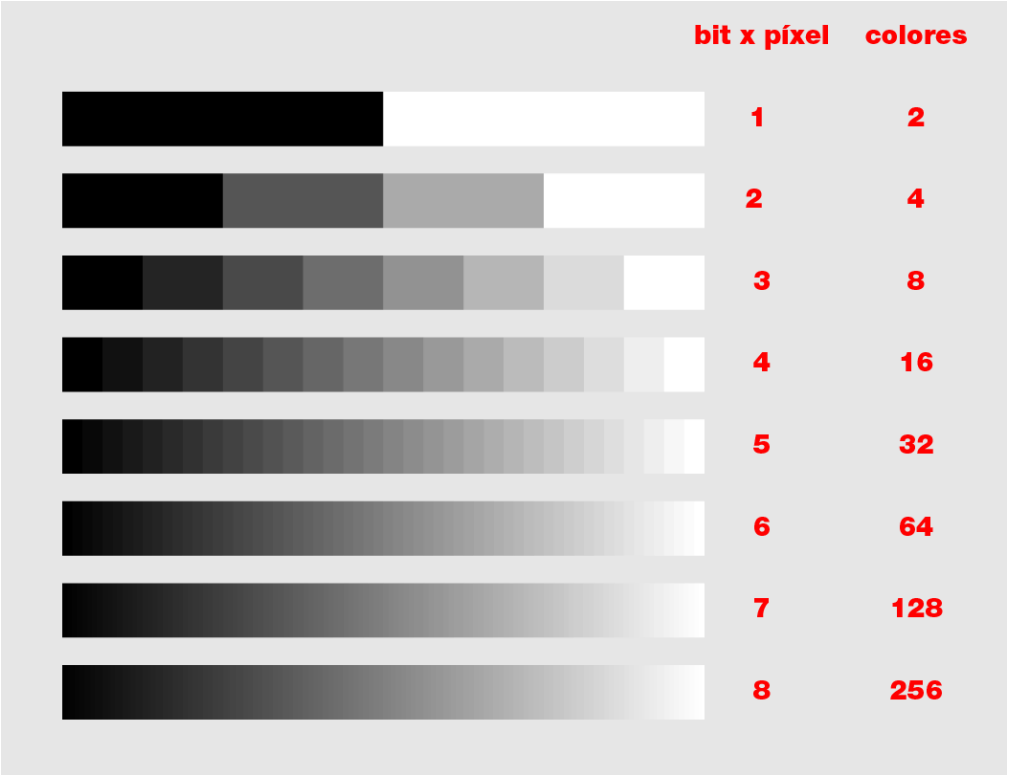


Figura 5. Escala de grises según la cantidad de bits. En color pasa lo mismo, depende de la cantidad de bits para la cantidad de colores.

Fuente: https://pixelnauta.com.ar/sitio_anterior/curso2011/photoshop1/clase1/index4.html

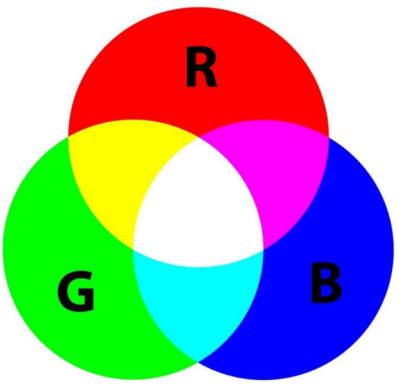


Figura 6. Imagen de combinación de colores red green blue RGB

Fuente: https://medium.com/@a20193062/color-digital-a14a08d18ed4

En el formato de RGB, los colores se generan a partir de una combinación de los colores principales rojo, verde y azul. Haciendo una combinación de estos colores obtenemos 8 bits por canal (color), o sea, que tendríamos un total de 24 bits en la imagen como muestra la siguiente figura (Figura 7).

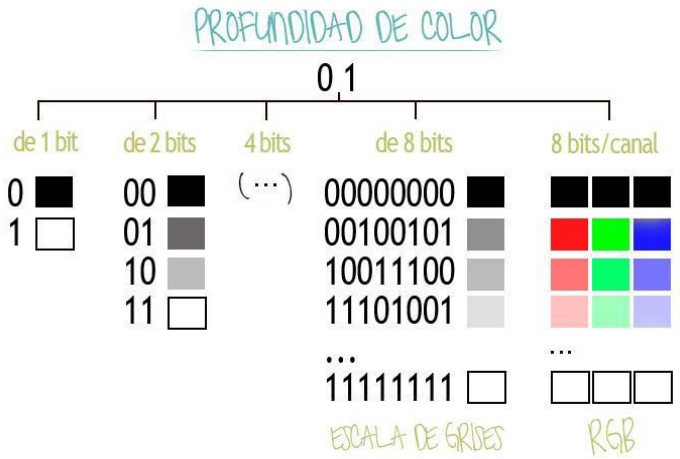


Figura 7. Mapa comparativo de bits por color.

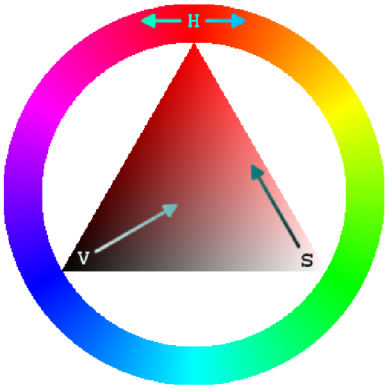
Fuente: https://adelossantos.files.wordpress.com/2014/05/3.jpg

En este trabajo se optó por trabajar con el formato de color HSV (*Hue,Saturation, Value*) debido a que OpenCV trabaja con este tipo de formato. De este modo, la interpretación de colores puede ser definida como rangos de colores dentro de un espectro y no como la combinación de estos como en RGB.

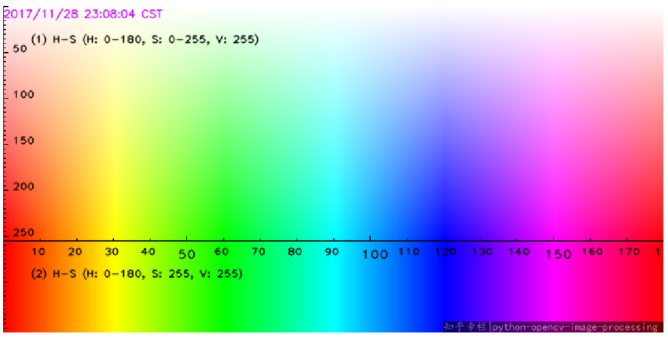
H (color en concreto). Valores de 0-360º. La gama cromática se representa en una rueda circular y este valor expresa su posición.

S (Saturación). Valores de 0-100%. De menos a más cantidad de color.

V (Value) o B (Brillo). Valores de 0-100%. De totalmente oscuro a la máxima luminosidad.



a. Representación en grados



b. REpresentación en ejes cartesianos

Figura 8. Imagen de representación de colores de hsv

Fuente a: https://es.wikipedia.org/wiki/Modelo_de_color_HSV

Fuente b: https://i.stack.imgur.com/TSKh8.png

Color en Hexadecimal

Otra forma de representar los colores en programas, desarrollo web y otras aplicaciones es representarlo en código hexadecimal, los colores son representados como tres pares de números hexadecimales para representar el mismo color que en RGB, HSV, CMYK o LAB. El código hexadecimal es representado con números y letras precedida por el carácter "#" que van desde 0 a 9 y de A a F. como ejemplo el negro sería #000000 y el blanco #FFFFFF.

#RRGGBB=#(RojoRojoVerdeVerdeAzulAzul)

6.2. Armado de brazo robótico

Gracias al aporte por parte de la Universidad, contamos con la disposición del brazo robótico LeArm 6DOF de LOBOT para los trabajos de investigación y desarrollo dentro de la misma. Este proyecto fue llevado a cabo con el brazo robótico mencionado y posee un conjunto de periféricos, con una aplicación de prueba, listos para el desarrollo de proyectos. Si bien, cuenta con un manual no muy específico en el box, trae consigo un joystick, para mando a distancia y pines Tx y Rx, para transmisión y recepción de comandos. Su armado completo se pudo realizar gracias a la ayuda de videos tutoriales por parte de la página oficial y de páginas no oficiales, ya que no quedaban del todo claros algunos detalles.

Al principio, no hubo grandes problemas con el armado general de las piezas y los servomotores, hasta que se realizaron pruebas de la aplicación. El brazo, al probar el software no realizaba los movimientos programados y hacía otros totalmente diferentes y bruscos. El problema radicaba en que los servos estaban en posición y orden incorrectos, también fue necesario calibrarlos para que se tuviera el rango y movimiento adecuados.

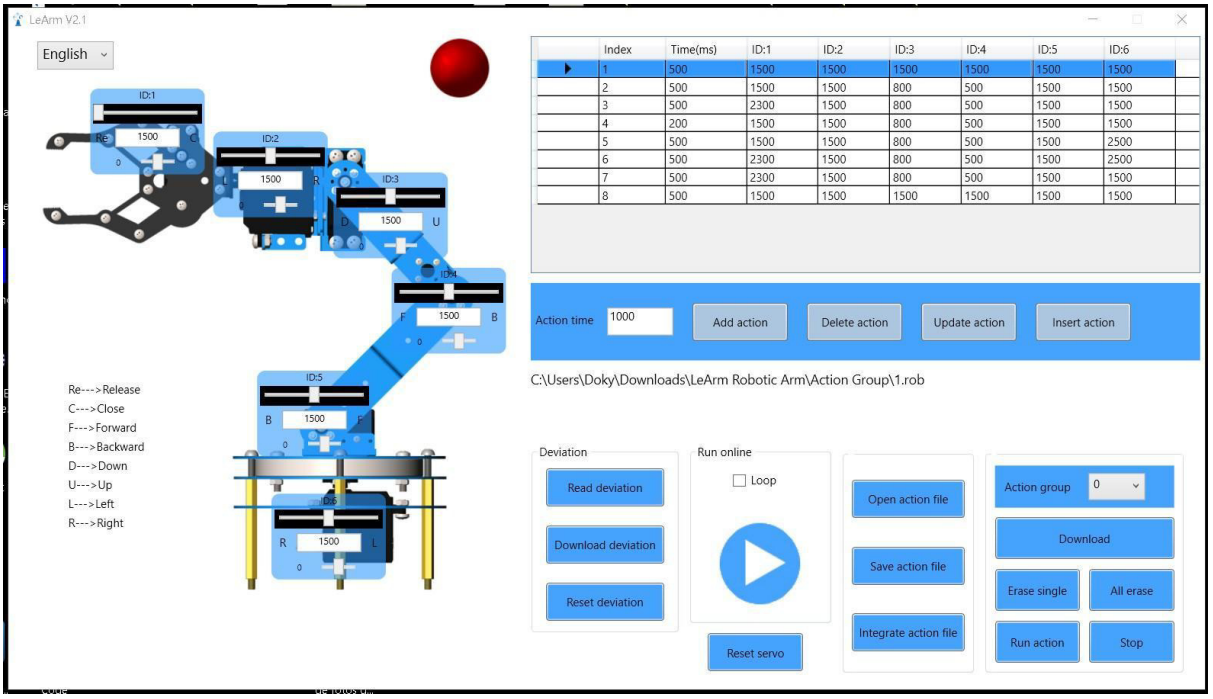


Figura 9. Aplicación LeArm para controlar el brazo.

Fuente: Hiwonder Co. Ltd

El software de la Figura 9 es una aplicación para realizar movimientos y conjuntos de movimientos predeterminados (action groups), se utilizó para realizar pruebas y calibrar el brazo. Podemos notar a simple vista que cada motor o “articulación” cuenta con un rango y dirección específica.

6.3. Armado con Arduino

La placa de Arduino UNO R3 con el microchip Atmega 328p, cuenta con pines de entradas y salidas analógicas como digitales, para la comunicación con sensores y otras placas. En este caso, el microcontrolador fue utilizado como programador para cargar código al ESP32-cam. Ya que al momento de empezar la práctica solo se contaba con ese material.

Instalación del entorno e interfaz de desarrollo

Primero se debe preparar el entorno de desarrollo para poder trabajar con Arduino y cargarle el código al ESP32-cam.

Se debe instalar el software de Arduino desde la página oficial (https://www.arduino.cc/), en este caso la versión 1.18.19.

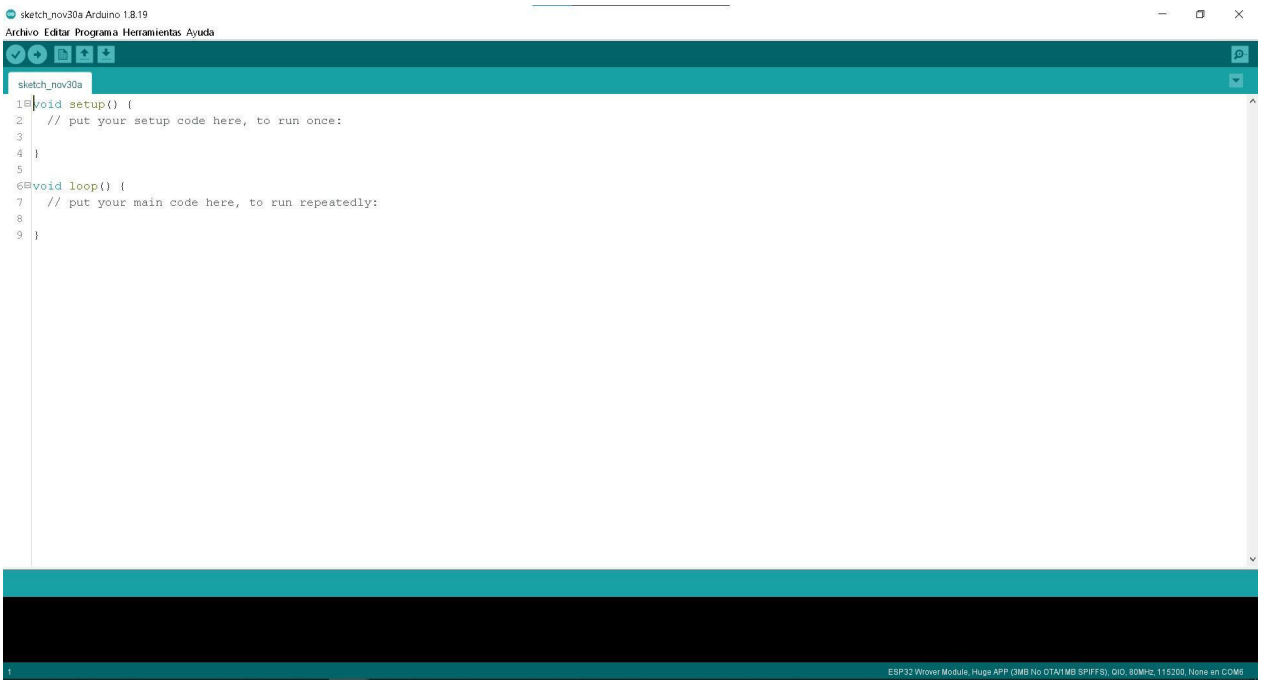


Figura 10. Interfaz de Arduino IDE

Antes de comenzar a desarrollar es necesario preparar el entorno y configurar la aplicación para que permita trabajar con el módulo ESP32-cam.

Se debe tener en cuenta que este programa es muy versátil y puede trabajar con muchas placas y módulos de desarrollo solo que no vienen preinstaladas, sino que es necesario descargar las librerías o de lo contrario no se podrá compilar ni reconocer la placa de trabajo.

Una vez instalado Arduino ide se debe ir a *Archivo>>Preferencias* y en el gestor de URLs colocamos el siguiente link:

https://dl.espressif.com/dl/package_esp32_index.json,http://arduino.esp8266.com/stable/package_esp8266com_index.json

Esas direcciones contienen las direcciones de archivos que nos permitirán trabajar con la placa. Seguido a eso, en *Herramientas>>Placa>>Gestor de tarjetas se* busca la tarjeta ESP32 y se instala *esp32 by Espressif System.*

Después de instalar la tarjeta se deben configurar los parámetros en: *Placa:"ESP32 Wrover Module", Upload speed: "115200", Flash Frecuency:"80MHz", Flash Mode: "QIO", Partition Scheme:"Huge APP (3MB No OTA/1MB SPIFFS)", Core Debug Level: "Ninguno" y el Puerto* depende del que se le asigne a la placa al momento de conectarla.

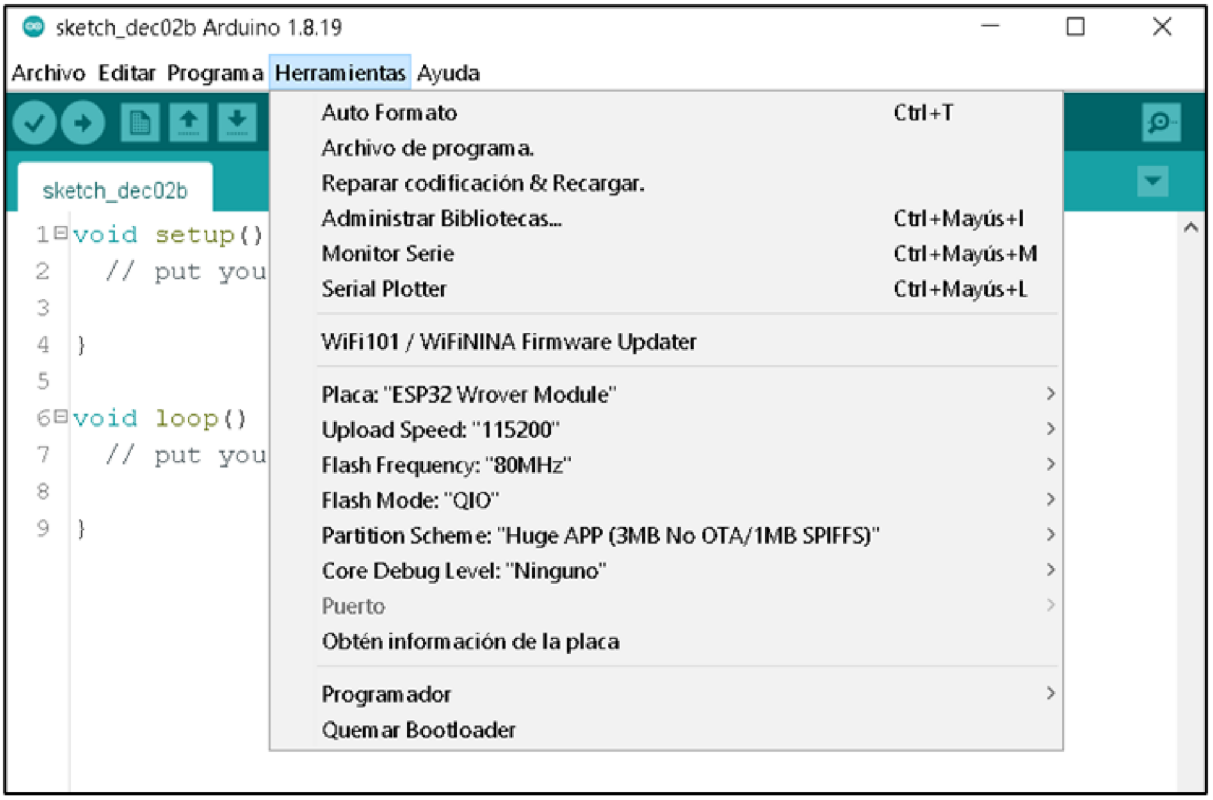


Figura 11. Configuración de la placa ESP32

Finalmente, una vez configurado todo se puede abrir un código de ejemplo para comprobar que esté todo bien configurado. En *archivo>>examples>>ESP32>>Camera>>CameraWebServer.* Dentro del código se debe descomentar solamente el *"#define CAMERA_MODEL_AI_THINKER"* y se debe agregar el SSID y contraseña de la red a la cual se va a conectar.

```
// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"

const char* ssid = "*********";
const char* password = "*********";
```

Figura 12. Configuración de proyecto de prueba

Para probar su funcionamiento, se debe conectar el módulo ESP32-cam con la placa de Arduino uno R3 y esta a su vez conectar a la computadora para cargar el código desde el ide de Arduino (ver figura 13).

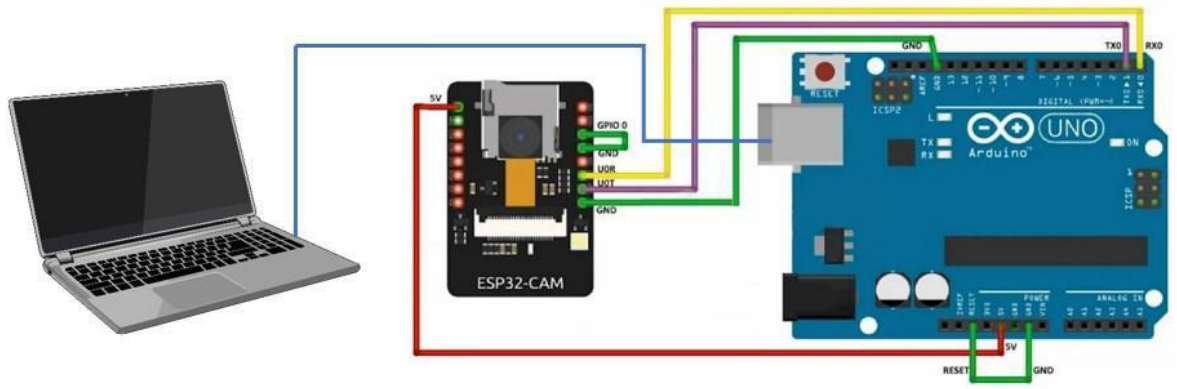


Figura 13. Conexión para programación del ESP

Con esta configuración no se deberían tener problemas, de todos modos se pueden producir ciertos inconvenientes y errores a la hora de correr el programa en la placa, para ello se detallaran los errores que han surgido a lo largo de la práctica y la forma en que fueron solucionados.

Error de conexión: se produce al momento de subir el código y cargarlo a la memoria de la placa. El problema que surge es que queda en un loop de *connecting* con la placa y finaliza la carga por conexión. El tiempo de carga (*time out*) llega a su fin y se produce el error.

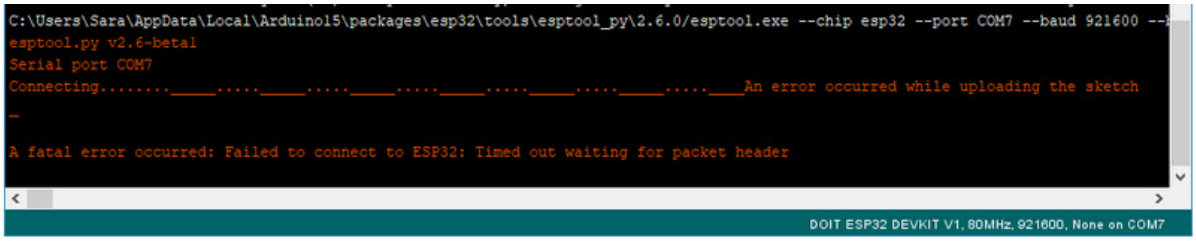


Figura 14. Error por Time out

Uno de los principales problemas que se tenía cuando se producía este error se debía en un 70% a un falso contacto entre los cables que conectan los pines del esp32 a los pines Arduino. Ya que al no estar soldados estos cables se van deteriorando y perdiendo firmeza en la conexión.

Error de energía: Producía el mismo error de conexión que mencionamos anteriormente y a su vez un error que indicaba que la placa no estaba conectada. Por ende, también se detenía la carga por time out.

Se probó como una posible solución soldar y conectar un capacitor (condensador electrolítico) de 10uF entre el GND (tierra) y el pin *EN.* Para obtener un voltaje parejo y evitar la caída de tensión al momento de correr el programa en la placa e iniciar la cámara.

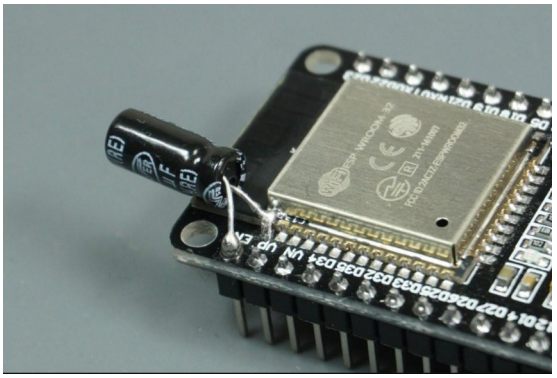


Figura 15. Capacitor para soportar caídas de tensión

Finalmente, se conectan los pines de 5v y GND a un USB para poder alimentarlo directamente de la pc y no desde el Arduino, ya que de esta manera el ESP puede recibir hasta 500mA desde la PC, mucha más corriente de la que puede entregar el kit de arduino.

Error de configuración: se produce al momento de validar el código, es necesario realizar la correcta configuración de los parámetros para que no se produzca un error en memoria que impida la carga del código en la placa.

Es necesario configurar la placa en *Wover* module y *Partition Scheme:"Huge APP (3MB No OTA/1MB SPIFFS)",* de lo contrario tendremos error de memoria.

Error de antena: una vez cargado el código y compilado al momento de hacer la prueba este no inicia el servidor ni la cámara. Debemos tener en cuenta el alcance de la antena wifi que viene incorporada. Tenemos que tener la placa de esp32-cam lo suficientemente cerca del router wifi para tener buena conexión. De lo contrario podremos tener error, señal intermitente, pérdida de fluidez y de imágenes.

Se evaluó que para tener mayor alcance podía conectarse y soldarse una antena wifi externa al esp32-cam, resolviendo de este modo los problemas de alcance con la red a la cual se conecta.

Error de conexión: Cabe destacar que como el desarrollo no se realizaba en un solo lugar era necesario conectarlo al wifi local de trabajo. Por tal motivo a veces el wifi cargado por código y el de la computadora de desarrollo difieren, produciendo el no inicio del programa y cámara. Por lo tanto, se debe tener en cuenta que la conexión de wifi del código que cargamos en la placa y del wifi de la computadora de desarrollo deben ser la misma.

6.4. Armado con programador

Luego de trabajar por un tiempo con Arduino como programador se pudo obtener gracias a la ayuda de los tutores un módulo FTDI, adaptador serie USB a de TTL. Debido a que la placa ESP32-cam no cuenta con conexión USB es necesario este tipo de adaptador para poder subir nuestro código, anteriormente se trabajaba con Arduino como programador.

El adaptador FTDI es una placa simple, sencilla y práctica para cargar código a placas. El modelo que se obtuvo es el adaptador FTDI con chipset CP2102, que cuenta con cinco pines para conectar y USB para conectar a la computadora. Tiene pines de *5V, 3V, TX, RX y GND.*

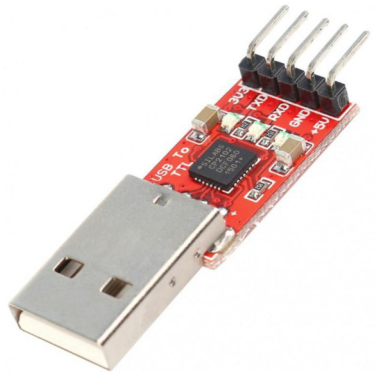


Figura 16. Programador USB para ESP32-CAM

Tabla 1: de pines conectados entre el ESP32-cam y el adaptador.

ESP32-cam	**FTDI**
5V	5V
GND	GND
RX	OUT
TX	OUR
3V	-

Con este tipo de adaptador se hace más simple y robusta la conexión con el ESP32-cam en comparación con el uso de Arduino, por el armado de cables y el USB directo a la computadora.

Luego de preparar el entorno físico del hardware, se procedió a realizar la misma configuración que se realizó con el Arduino, ya que utilizan los mismos parámetros. Al igual que con Arduino se pueden producir los mismos errores y solucionarlos de la misma forma.

6.5. Brazo y cámara

La cámara y el brazo deben estar conectados al ESP32-cam para integrar un sistema en conjunto. Por tal motivo es necesario integrar el brazo al módulo que

ya cuenta con la cámara principalmente porque se deben transmitir instrucciones de los movimientos por señales a través los pines tx y rx de la placa. Para la comunicación entre brazo y ESP32 se editó el proyecto *software serial* y se lo adaptó al nuevo hardware.

El módulo del ESP32-cam ya cuenta con pines conectados a la cámara y pines disponibles para poder integrar, como en este caso, otros dispositivos.

Tabla 2. Pines de ESP32-cam que utiliza la cámara

OV2640	ESP32-cam	Variable en el código
D0	GPIO 5	Y2_GPIO_NUM
D1	GPIO 18	Y3_GPIO_NUM
D2	GPIO 19	Y4_GPIO_NUM
D3	GPIO 21	Y5_GPIO_NUM

D4	GPIO 36	Y6_GPIO_NUM
D5	GPIO 39	Y7_GPIO_NUM
D6	GPIO 34	Y8_GPIO_NUM
D7	GPIO 35	Y9_GPIO_NUM
XCLK	GPIO 0	XCLK_GPIO_NUM
PCLK	GPIO 22	PCLK_GPIO_NUM
VSYNC	GPIO 25	VSYNC_GPIO_NUM
HREF	GPIO 23	HREF_GPIO_NUM
SDA	GPIO 26	SIOD_GPIO_NUM
SCL	GPIO 27	SIOC_GPIO_NUM

POWER PIN	GPIO 32	PWDN_GPIO_NUM

Tabla 3. Pines disponibles

ESP32-cam	Descripción
5VDC	Entrada 5 volts
3.3VDC	Entrada 3.3 volts
GND	Negativo
GND	Negativo
GND	Negativo
GPIO 0	JUMP con GND para carga
GPIO 1	Trasmisor TX

GPIO 3	Receptor RX
VCC OUT	Salida de voltaje 5v/3.3v
GPIO 16	Entrada/Salida
GPIO 2	Entrada/Salida
GPIO 4	Entrada/Salida
GPIO 12	TX Brazo robótico
GPIO 13	Entrada/Salida
GPIO 14	Entrada/Salida
GPIO 15	Entrada/Salida

Luego de analizar la placa del ESP32-cam y verificar cuales son los pines libres y su funcionalidad, se procedió a analizar la placa del brazo robótico. Las características que posee este controlador son:

- Conexión Bluetooth, para manejarlo desde una aplicación celular.
- Pines para conectar hasta 6 (seis) servomotores.
- Conector USB, para conectarlo y controlarlo desde el software *LeArm*.
- Led de señal y led de encendido.
- Botón de *Run* para correr las instrucciones que se le programen.
- Alarma de voltaje bajo.
- Buzzer para indicar alerta.
- Pines para conectar el joystick Wireless y controlarlo desde el mando.
- Jack para conectarlo a DC (corriente continua o directa)
- Switch de fuente de alimentación.
- Entrada para conectar a corriente continua (vcc+ ,vcc-), batería.
- Pines para comunicarse con otras placas de desarrollo. Posee pines de 5v, GND, TX y RX.

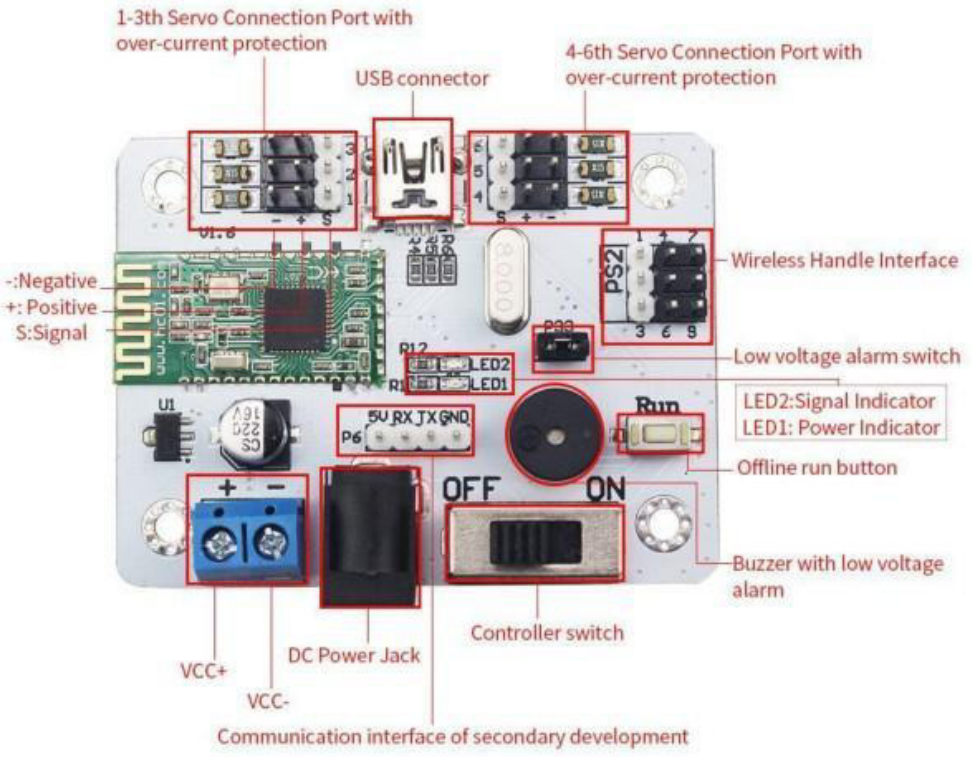


Figura 17. controlador del brazo robótico. (fuente manual de usuario oficial)

Como se puede observar, ambas placas pueden ser conectadas por medio de pines específicos para la comunicación y el desarrollo a través de la comunicación *Serial.*

Cuando se habla de Serial, se hace referencia al tipo de comunicación que se implementa entre los dispositivos. Para la comunicación de dos dispositivos digitales se utilizan dos puertos (pines) o más, en este caso para transmitir (TX) y recibir (RX) datos.

La comunicación se puede realizar mediante el proyecto Hardware Serial, en donde se utiliza el módulo serie del kit ESP32-CAM, en donde se utiliza un hardware dedicado al que se le proveen los datos a transmitir. En Software Serial

se utilizan pines digitales genéricos GPIO, si bien están configurados el pin transmisor (pin de salida) y el receptor (pin de entrada) también se implementa el protocolo serie por software para el envío bit a bit de los datos (*Serial.print, Serial.begin* o *Seria.read*).

7. Visión digital

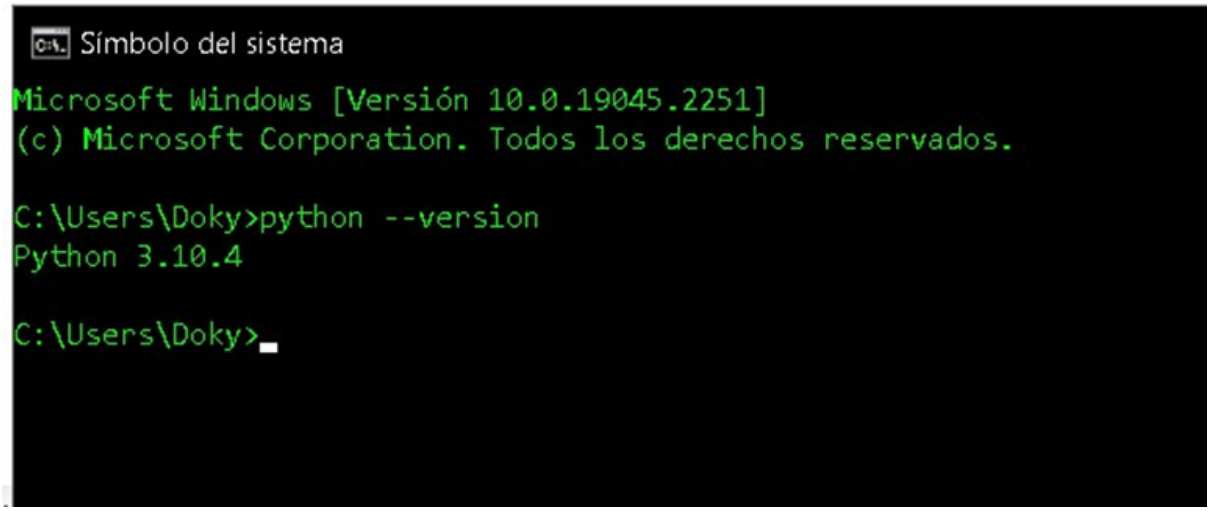
7.1. Entornos virtuales

Para el desarrollo de la visión digital en este proyecto se utilizaron entornos virtuales en Python. Un entorno virtual podemos definirlo como un entorno parcialmente aislado de Python el cual permite instalar paquetes en particular para una aplicación. Supongamos que tenemos nuestras aplicaciones como Python, OpenCV, Numpy y otras. No hace mucho tiempo, para trabajar en diferentes proyectos o simplemente probar código hecho en otras versiones era necesario instalar todos los paquetes con su versión específica y al momento de probar otro proyecto debíamos desinstalar todo lo anterior para instalar lo que requería el nuevo proyecto. Esto se vuelve demasiado tedioso para el desarrollo y gracias a la creación de entornos virtuales se puede trabajar con diferentes versiones de esos programas sin necesidad de estar instalando y desinstalando todos los paquetes. Básicamente lo que hace es instalar una versión estándar y en cada proyecto que se cree un entorno virtual podemos trabajar e instalar los paquetes con sus diferentes versiones sin necesidad de desinstalar nada.

7.2. Instalación del entorno

Primero que nada debemos instalar Python, podemos hacerlo desde su página oficial (https://www.python.org/downloads), en este caso se instaló la versión 3.10.4 para Windows 64 bits. En la instalación es necesario instalar pip para usarlo posteriormente. También debemos asegurarnos que estén agregadas las variables de entorno para que el sistema pueda reconocerlo. Desde la consola de Python o de

Windows con el comando >>*Python –version* podemos ver que se haya instalado correctamente.



Si bien se puede hacer desde la pagina oficial (https://opencv.org/) tambien se puede instalar una versión no oficial (https://pypi.org/project/opencv-contrib-python/) esto ayudará y ahorrará mucho tiempo a la hora de desarrollar. Las diferentes posibilidades para instalar OpenCV según lo requerimientos son:

pip install opencv-python, para instalar los módulos principales.

pip install opencv-contrib-python, para instalar módulos principales y extras (contrib).

pip install opencv-python-headless, instala módulos principales sin funcionalidad GUI.

pip install opencv-contrib-python-headless, instala módulos principales y

extras (contrib) sin funcionalidad GUI.

Ahora debemos dirigirnos a la consola de Windows e instalar el paquete que se requiera. En este caso se optó por *instalar pip install opencv-contrib-python.*

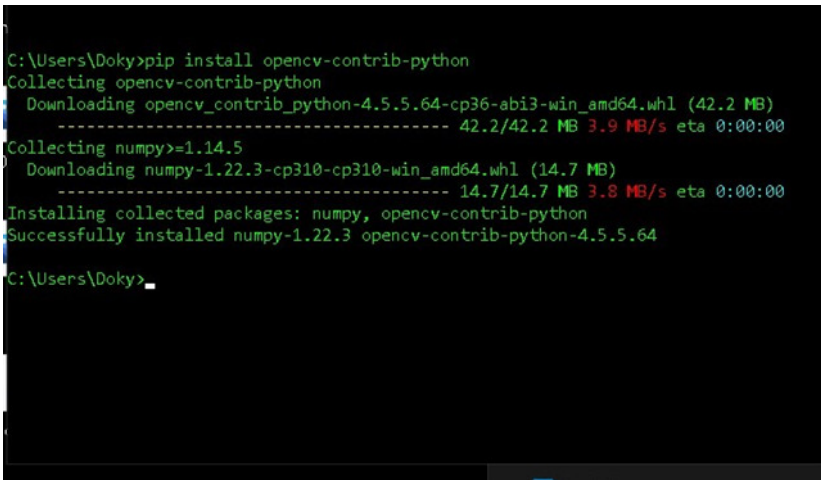


Figura 19. Imagen de instalación de OpenCV correcta

Para probar se pueden usar los siguientes comandos

>>Python

>>import cv2

>>cv2.__version__

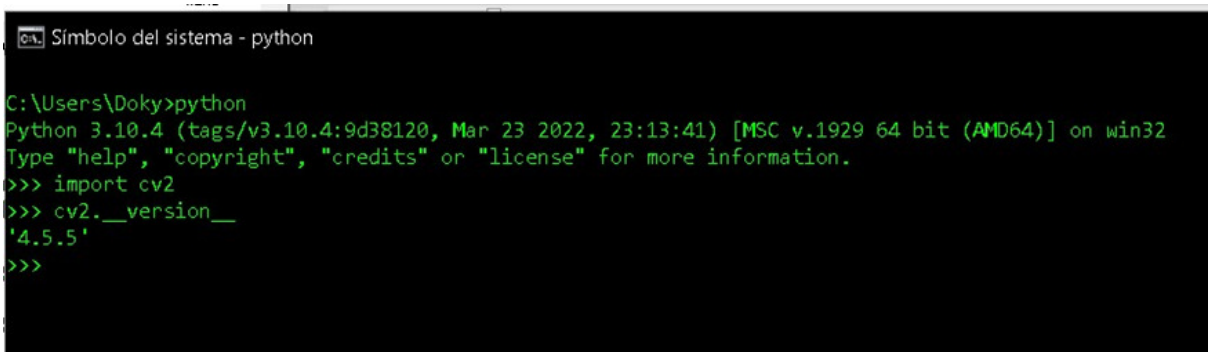


Figura 20. Prueba opencv

Seguido a esto, se procede a la instalación de Numpy. Es una librería necesaria para el desarrollo, con soporte para crear vectores y matrices grandes multidimensionales, además de otras funciones matemáticas de alto nivel.

Para llegar al objetivo fue necesario un avance paulatino con respecto al desarrollo de práctica. Los primeros pasos consistieron en trabajar en diferentes proyectos que hacían un aporte hacia el objetivo final. Antes es necesario para cada proyecto crear entornos virtuales para así poder aislar cada caso y trabajarlo por separado. Pero que en una sumatoria de estos nos darán las herramientas para poder unificar lo aprendido y sumarlo al proyecto definitivo.

La creación del o de los entornos virtuales se realiza de la siguiente manera:

Desde la consola cmd se instala virtualenv para crear entornos virtuales

>>pip install virtualenv

Se apunta a una carpeta donde estará alojado el proyecto. Una vez en la carpeta del proyecto se crea el entorno virtual

>>cd proyecto1

>>virtualenv nombre_del_entorno_virtual

Aparecerá un mensaje indicando que se creó el entorno virtual. Luego se procede a activarlo

>>.\nombre_del_entorno_virtual\Scripts\activate

Está activo si aparece el nombre del entorno virtual en la parte superior izquierda al principio del path.

Una vez verificado, ahí mismo se pueden instalar los diferentes paquetes para cada proyecto.

Si se desea saber qué paquetes están instalados y que versiones de cada uno en cada entorno, desde la consola se debe dirigir a la carpeta del proyecto y poner el siguiente comando

>>pip freeze

Y para desactivarlo o salir se hace lo siguiente

>>desactivate

7.3. Reconocimiento de imágenes

En principio se inicia con reconocimiento de imágenes. Es muy importante para comenzar a explorar lo que es el mundo de OpenCV. Entonces en esta primera prueba se realizó una visualización y se mostró una imagen.

1. Es necesario importar la librería de OpenCV (“cv2”).
2. Importar Numpy (extensión de Python) para trabajar con matrices y vectores.
3. Con la función de cv2.imread('C:\\Users\\Doky\\Downloads\\OpenCV-y-Python--master\\pelotaRoja.png') se puede leer una imagen en formato PNG llamada “pelotaRoja”.
4. En la línea 4 transformar la imagen a escala de grises
5. En la siguiente, línea 5, umbralizar y así obtener una imagen binarizada.
6. Especificar la imagen binaria *cv2.findcontourns*
7. Luego dibujar los contornos con *cv2.drawContours.*
8. Hay que especificar en donde se van a dibujar los contornos, en este caso en la imagen
9. Y por último, mostrarlos en una ventana.

```python
import cv2
import numpy as np
imagen = cv2.imread('C:\\Users\\Doky\\Downloads\\OpenCV-y-Python--master\\pelotaRoja.png')
gray = cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
_,th = cv2.threshold(gray,100,255,cv2.THRESH_BINARY)

contornos1,hierarchy1 = cv2.findContours(th, cv2.RETR_EXTERNAL,
            cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(imagen, contornos1, -1, (0,255,0), 3)
print ('len(contornos1[2])=',len(contornos1[2]))
cv2.imshow('imagen',imagen)
cv2.imshow('th',th)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figura 21. Código detectar contorno en imagen

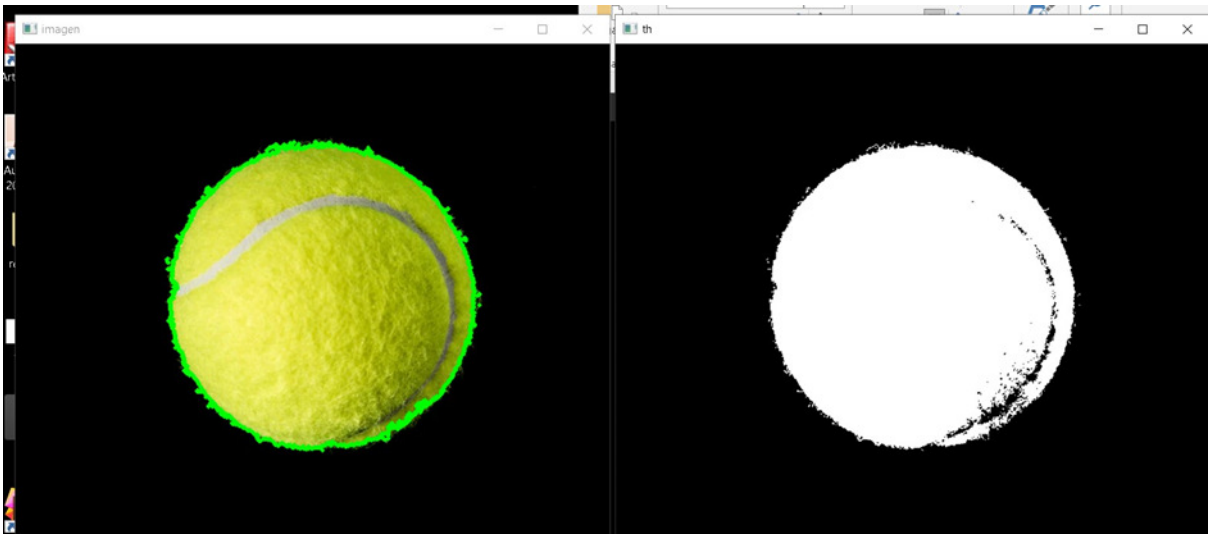


Figura 22. Resultado del procesamiento y obtención del contorno

7.4. Obtener stream

Una vez realizada la prueba de puesta en funcionamiento de Python con OpenCV y sus paquetes, se procede a obtener la imagen de un video. Esto quiere decir que ya

no es una imagen fija, nítida y estable, por lo que se probó primero leyendo un video descargado localmente para determinar la forma en que se mostraba. Una vez finalizado correctamente se pudo pasar a obtener la imagen desde la cámara de la computadora, en mi caso desde la webcam de la notebook.

Para obtener la imagen desde la webcam del sistema se debe configurar en cero(0) para la computadora

```
>>cam = cv2.VideoCapture(0)
```

ret detecta imágenes .frame (imágenes leídas en milisegundos)

```
>>ret,frame = cam.read()
```

En este ejemplo se detectan los colores verdes en un rango

```
>>rangomax = np.array([255,251,50])#detectamos objetos verdes maximo 255,51,50

>>rangomin = np.array([0,0,0])
```

Cuando un pixel esté en el rango lo pone en blanco y cuando este fuera de este lo pone en negro 51,1,0

```
>>mascara = cv2.inRange(frame, rangomin, rangomax)
```

Como la imagen en video es más difícil de definir es necesario eliminar el ruido de la imagen.

```
>>opening = cv2.morphologyEx(mascara, cv2.MORPH_OPEN, kernel)
```

Ahora para identificar qué es lo que se está detectando se crea un rectángulo alrededor y un punto en el centro, esto ayudará a saber si es correcto el objeto.

```
>>cv2.rectangle(frame,(x,y),(x+w,y+h),(255,255,0),4)
```

>>cv2.circle(frame,(x+w//2,y+h//2),6,(0,0,100),-1)

Finalmente se muestra en una ventana y se da la opción de finalizar el video con la tecla ”Esc”

>>cv2.imshow('camara' ,frame)

>>k = cv2.waitKey(1) & 0xFF#exit con Esc

>>if k==27:

>> break

```
import cv2
import numpy as np

#numpy (estenson python )trabaja con matrices y vectores

cam = cv2.VideoCapture(0) #obtener webcam del sistema 0 para el de la compu
#cam = cv2.VideoCapture('http://192.168.247.91:8081') #obtener webcam del sistema  para el de la camara ip

kernel = np.ones((5,5),np.uint8)#unos de 5,5 y 8 bit

while (True):
    ret,frame = cam.read() #ret detecta imagenes .frame imagenes leidas en milisegundos
    rangomax = np.array([255,251,50])#detectamos objetos verdes maximo 255,51,50
    rangomin = np.array([0,0,0])#detectamos objetos verdes minimo
    mascara = cv2.inRange(frame, rangomin, rangomax)#cuando un pixel este en el rango lo pone en blanco y
    #cuando este en fuera lo pone en negro 51,1,0
    opening = cv2.morphologyEx(mascara, cv2.MORPH_OPEN, kernel) #eliminammis ruido de la imagen
    x,y,w,h = cv2.boundingRect(opening)# x , y puntos inicio. h,w punto final del rectangulo
    cv2.rectangle(frame,(x,y),(x+w,y+h),(255,255,0),4)
    #rectangulo---(imagen donde se muestra(coord inicial),(coord final),(color), grosor )
    cv2.circle(frame,(x+w//2,y+h//2),6,(0,0,100),-1)
    #cv2.circle(frame,(x+w//2,y+h//12),6,(0,0,100),-1)
    cv2.imshow('camara' ,frame)
    k = cv2.waitKey(1) & 0xFF#exit con Esc
    if k==27:
        break
    #cap.release()
```

Figura 23. Código de detección de color verde

7.5. Detectar color y determinar coordenadas

Lo primero que se debe hacer en este caso es pasar los colores de RGB a HSV, ya que de esta forma es mucho más fácil trabajar con ellos e identificarlos.

Pasar los colores de la imagen de video a HSV:

```
>>frameHSV = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
```

Igualmente se debe determinar el rango en el que trabajan, en este caso en los colores del espectro azul

```
>>azulBajo = np.array([100,100,20],np.uint8) #identificar azul
```

```
>>azulAlto = np.array([125,255,255],np.uint8)
```

Luego de detectar que estén en el rango se procede a ponerlos en una variable mask binarizada.

```
>> mask = cv2.inRange(frameHSV,azulBajo,azulAlto)
```

Con esto ya es posible ver en blanco los colores dentro del rango que se detectaron en el video.

El proceso sería el siguiente:

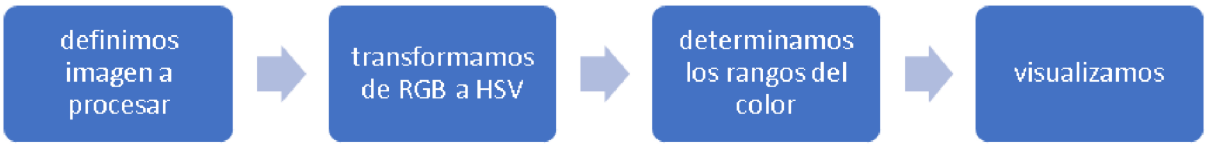


Figura 24. Proceso de detección de color

Una vez comprobado se procede a seguir con la mejora, en este caso agregando las coordenadas en la imagen. Lo que se observa a simple vista es que si bien se

detecta el color azul de los objetos también hay muchas áreas detectadas que no corresponden a un objeto en particular y el cual se desea eliminar para determinar la coordenada en concreto del objeto que interesa analizar.

Continuando con el código anterior se desea mejorar la precisión por lo tanto se recorren todos los contornos detectados con:

```
>>for c in contornos:
```

La forma en que se compara es determinada por su área definida en píxeles.

```
>>area = cv2.contourArea(c)
```

A medida que se van analizando uno por uno, se determina que como rango para ser establecido como un objeto debía superar los 3000 pixeles de área.

```
>>if area > 3000:

>>  M = cv2.moments(c)

>>if (M["m00"]==0): M["m00"]=1

>> x = int(M["m10"]/M["m00"])

>>y = int(M['m01']/M['m00'])
```

Ya teniendo el o los objetos de interés que en este caso superan los 3000 pixeles de área, se procede a determinar su centro y calcular su coordenada de la siguiente manera:

```
>>cv2.circle(frame, (x,y), 7, (0,255,0), -1)

>>font = cv2.FONT_HERSHEY_SIMPLEX

>>cv2.putText(frame, '{},{}'.format(x,y),(x+10,y), font,

0.75,(0,255,0),1,cv2.LINE_AA)
```

>>nuevoContorno = cv2.convexHull(c)

>>cv2.drawContours(frame, [nuevoContorno], 0, (255,0,0), 3)

Finalmente se muestra en una nueva ventana.

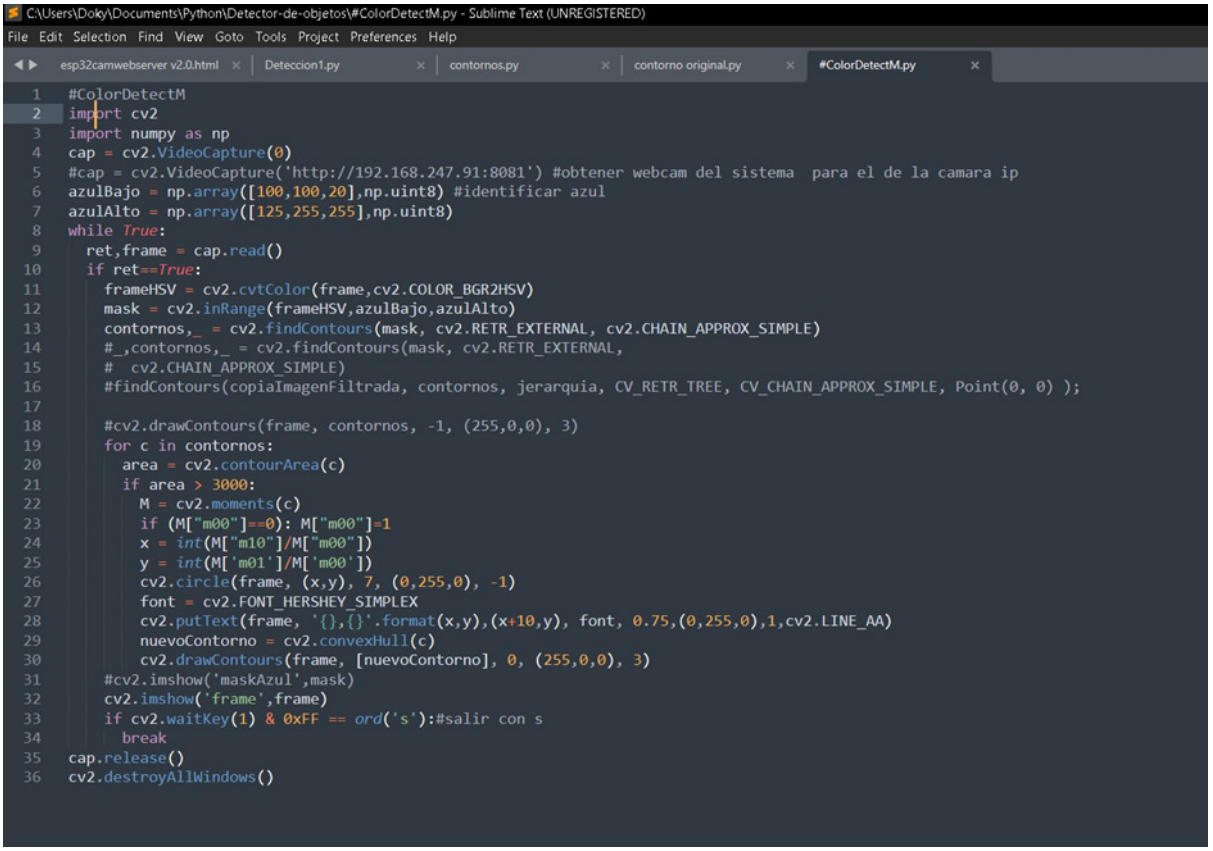


Figura 25. Código detección de colores azul y coordenadas

De esta manera se pueden obtener objetos de diferentes tamaños, colores, saber su ubicación, su contorno y seleccionar por su área.

Ahora se puede tomar el video desde la ip que transmite la ESP32-CAM para detectar objetos, colores y determinar su ubicación con la siguiente modificación en el código.

>>cap = cv2.VideoCapture('http://192.168.247.91:8081')

Cabe resaltar que la ip depende del lugar en que se encuentre y de la red a la que esté conectada.

El proceso que de ahora en más se llevará a cabo para obtener las coordenadas es el que se observa en la figura 26.

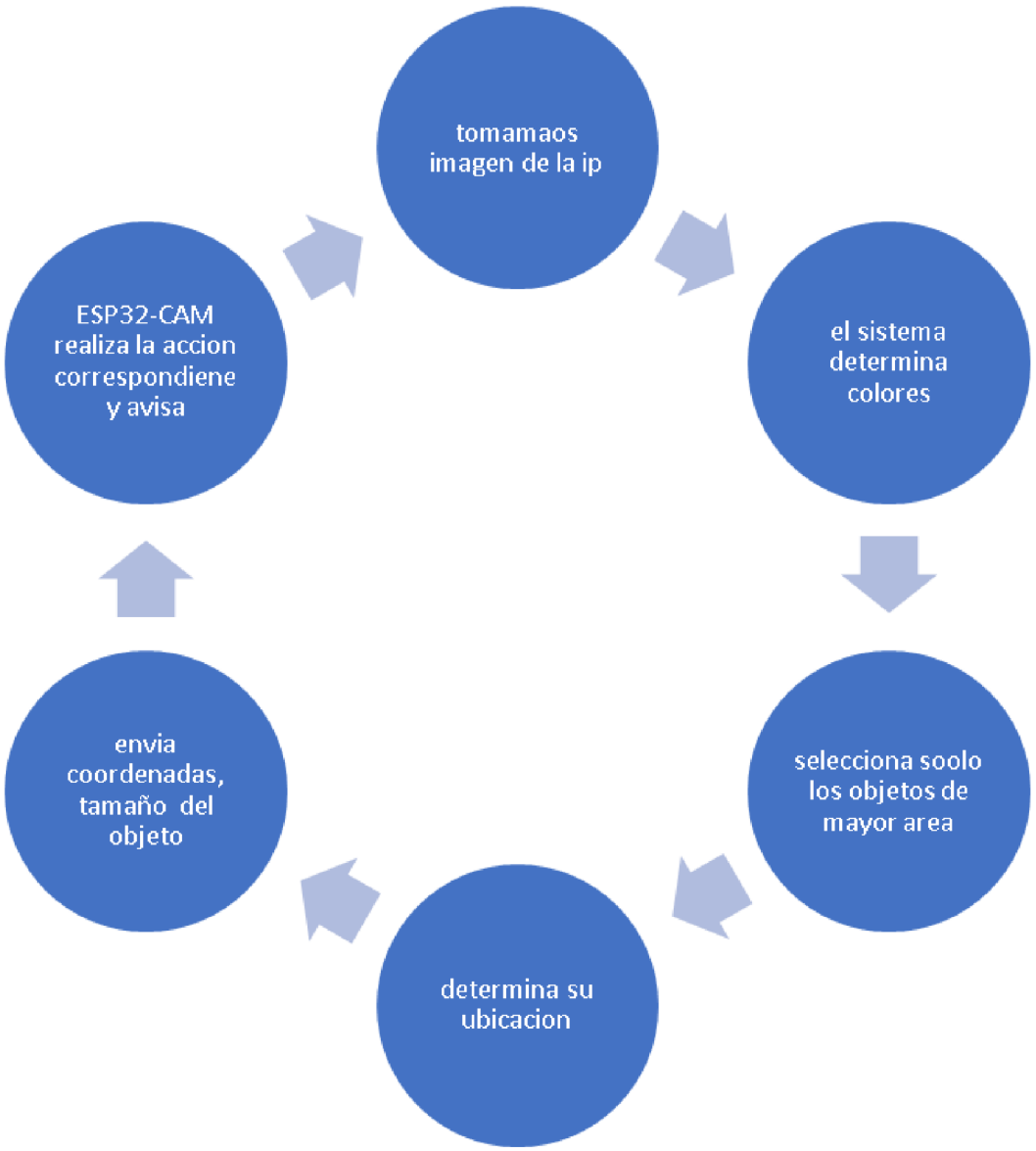


Figura 26. Gráfica de secuencia de detección de objetos

Desde el lado del brazo es necesario que la cámara esté en una posición fija ya sea por encima del brazo robótico o que no pertenezca a su estructura.

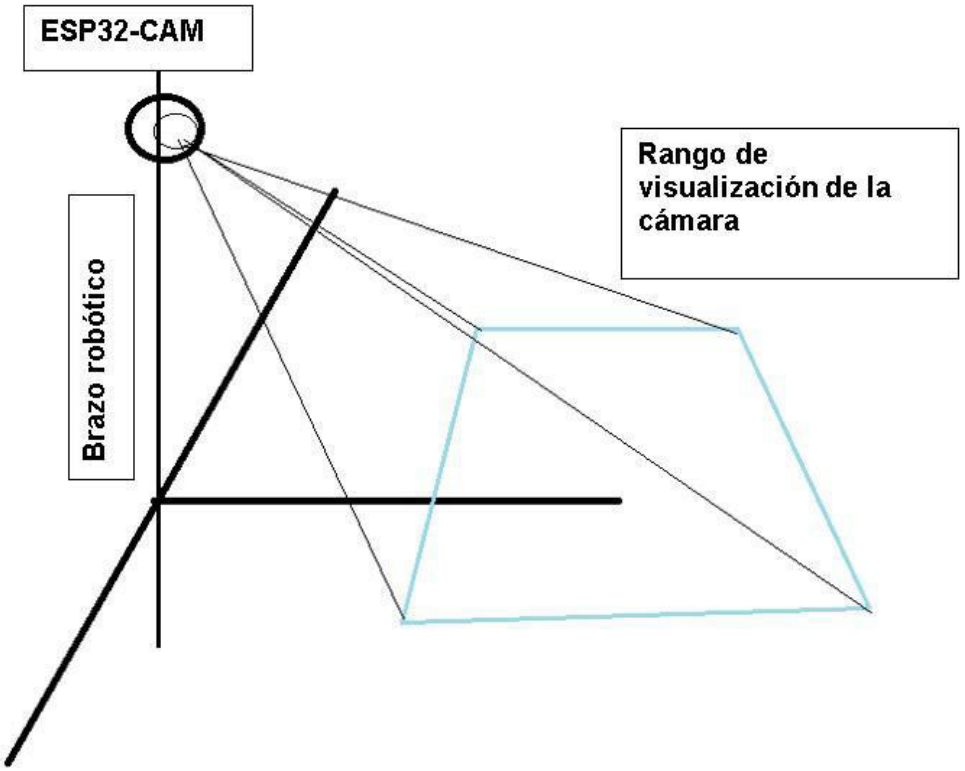


Figura 27. Imagen de posición del brazo robótico, cámara e imagen que toma

Para finalizar con el proceso de desarrollo se determina que las coordenadas de los objetos que detecte la cámara, dentro del ángulo de la visión mencionado anteriormente, serán enviadas para el cálculo de su distancia. Entonces cuando el objeto esté dentro de un rango determinado se enviará inmediatamente la información al brazo robótico para que realice el movimiento adecuado.

Si se toma como referencia una hoja A4 y la posicionamos en la cámara, sabemos que sus dimensiones son de 29,7 cm x 21 cm.

Se puede calcular de la siguiente fórmula:

relación de aspecto= a(alto) x b(ancho)

pixeles= a x b

Entonces:

Relación aspecto = 29,5cm/21cm= 1.40:1

Ancho=29,5cm=720 Píxeles

Alto=720/1.4=514,28 Pixeles

entonces 21cm equivale a 514 píxeles

29cmx21cm

Ya sabiendo cuantos pixeles representan 1cm (24,47 pixeles) se puede poner en el código de la detección de objetos que cuando un objeto esté entre el rango superior no enviará ninguna señal. Ahora mientras esté dentro del rango se enviará automáticamente la información y la instrucción.

Rango superior

Rango inferior

Rango superior:

Altura desde 244 píxeles (10cm) a 514,28 pixeles (21cm)

Largo desde 0 píxeles(0cm) a 720 pixeles (29,5cm)

Rango inferior:

Altura desde 0 píxeles (0cm) a 243 píxeles (10cm)

Largo desde 0 píxeles(0cm) a 720 pixeles(29,5cm

7.6. Aplicación final.

Como resultado final después del desarrollo y la investigación por parte de todas las etapas propuestas tenemos una aplicación funcional, la cual detecta objetos de manera autónoma y manda por medio de la interfaz serie la información para que pueda ser procesada. De este modo, de manera local quedaría cubierta la autonomía del proyecto para realizar las acciones según corresponda por parte del desarrollo en Python y OpenCV.

Respecto al monitoreo y control remoto, el mismo se obtiene mediante la aplicación web, a la que se debe acceder desde cualquier cliente (navegador web) y permite visualizar la aplicación tanto en computadoras, tablets o celulares. Se debe destacar que la aplicación web se encuentra embebida en la memoria del procesador ESP-32 por lo que soporta una cantidad limitada de clientes, de cualquier forma este tipo de sistemas está diseñado para ser usado por un cliente a la vez.

La interfaz que el usuario posee es muy sencilla e intuitiva de tal forma que sea fácilmente manipulable y de acceso inmediato para cualquier emergencia. Contiene tres botones superiores para monitoreo y cuatro inferiores para el control del brazo robótico. También posee un menú desplegable para ajustar parámetros de la cámara y/o adaptarla al ambiente o luz del ambiente donde se la coloque.

Get still para obtener una captura

Enroll face para detección de rostros

Start Stream para comenzar video stream

► para mover el el brazo a la izquierda

◄ para mover el brazo a la derecha

▲ para mover el brazo hacia arriba

▼ para mover el brazo hacia abajo

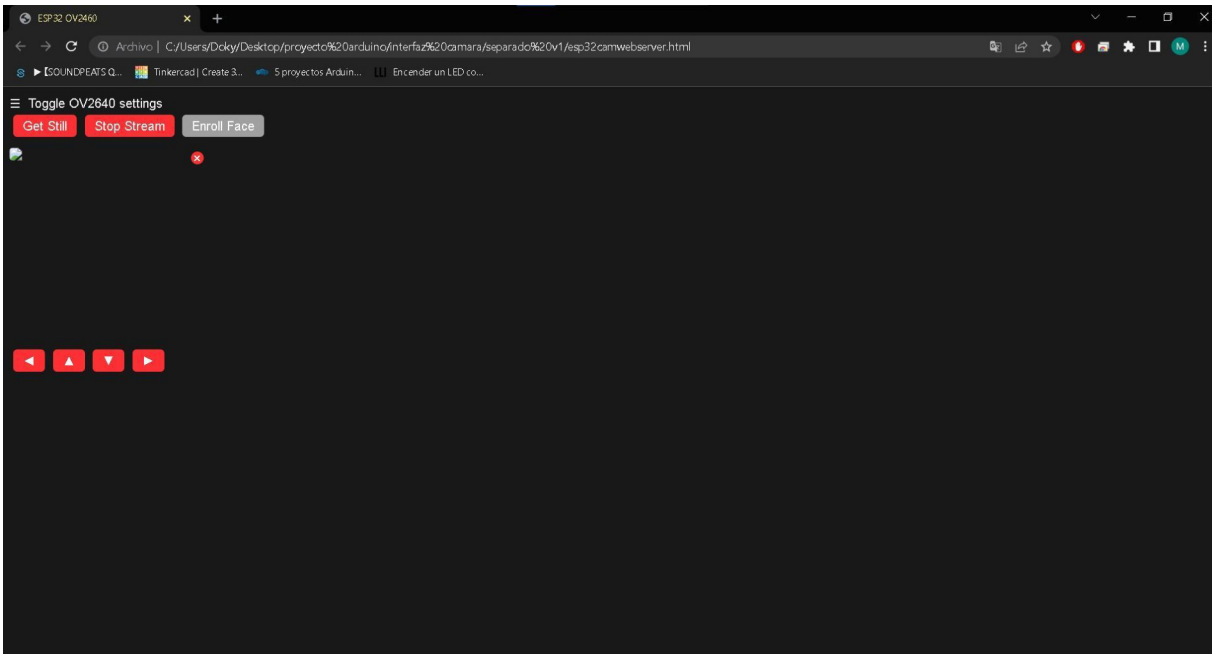


Figura 28. Imagen de interfaz de usuario desde una notebook

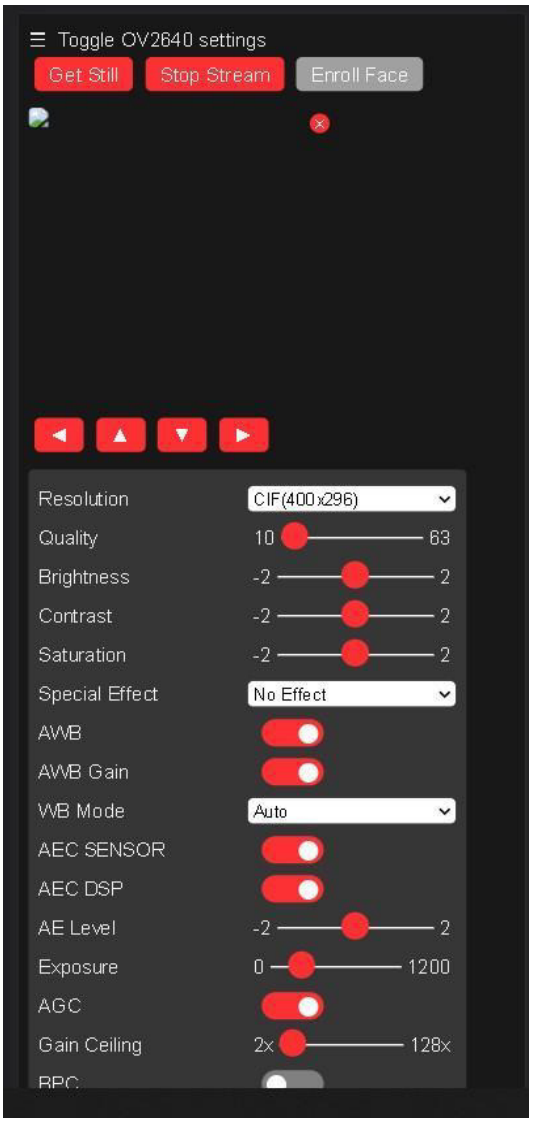


Figura 29. Imagen de interfaz vista desde un smartphone

8. Mejoras y actualizaciones

8.1. Contexto actual

Hoy en día la tecnología está incrustada en la vida cotidiana del ser humano, que lo hace parecer dependiente de ella. Sin duda alguna es muy importante a diario y clave tanto para el desarrollo como la subsistencia del hombre. Lo que habitualmente se utiliza como los aparatos electrónicos, maquinarias entre otros, nos trae consigo el avance en la comunicación entre individuos y dispositivos. Sin embargo, lo que hace posible dicha comunicación en su gran mayoría son las redes. Internet hace posible la comunicación remota, estar conectados transmitiendo y recibiendo información en diferentes formatos. Ahora, a lo que se llegó con el avance de internet es mantener informados a los usuarios a través de dispositivos terminales, acerca del medio en el que se encuentran. A este concepto se lo llama *Internet de las cosas IOT (internet of things),* estos dispositivos terminales cuentan con sensores para tomar medidas de diferentes recursos o estímulos que los rodean. Gracias a que poseen conexión con internet la información obtenida es procesada y mostrada a los usuarios para saber por ejemplo probabilidades de lluvia, humedad, nivel de río, cloacas, actividad sísmica, etc. Otro lugar en donde se puede ver esta tecnología es en las casas inteligentes.

Hoy en día la mayoría de los aparatos electrónicos que se utilizan de manera habitual posibilitan la interacción con ellos o conocer su estado por medio de internet junto con una aplicación móvil o sitio web al que estén vinculados. Por citar algunos de los ejemplos más comunes como prender el aire acondicionado, ver cámaras de seguridad y hablar, encender la lavadora, saber que falta en la heladera, conocer el estado de las plantas, dejar limpiando la casa con la

aspiradora, prender o apagar luces y muchas otras cosas más. En base a este desarrollo podemos ver muchas personas, aficionados, investigadores, alumnos, profesores y universidades dedicando tiempo al desarrollo de nuevos proyectos.

8.2. Módulos a agregar

En cuanto a la continuación del proyecto se podrían aplicar ciertas mejoras que fueron consideradas para obtener un mayor rendimiento en cuanto a lo que brinda la arquitectura y el hardware. Además, se debe tener en cuenta que el algoritmo de detección de objetos y cálculo de las coordenadas dentro de una imagen todavía no fue integrado al sistema completo (brazo - cámara). Lo que significa que ambas partes fueron implementadas y probadas por separado, logrando un funcionamiento aceptable.

Se debe tener en cuenta que son mejoras con respecto al objetivo que se proponga. En este caso, tanto la cámara como el brazo robótico están fijos en un determinado lugar lo que lo hace dependiente del ambiente.

Evaluando diferentes escenarios y posibles aplicaciones del mismo se consideraron las siguientes sugerencias:

Si fuese necesario mover la estructura principal del brazo por determinado motivo, se requeriría agregar un módulo de movilidad. Esto puede ser una posibilidad para darle mayor autonomía al proyecto. Básicamente se puede fijar la base del brazo a un robot móvil (plataforma con ruedas que contienen servomotores para realizar la tracción). El ESP32-CAM cuenta con pines adicionales que están disponibles para interactuar con el robot móvil, por ejemplo, el pin GPIO16 y el GPIO15. De realizar esa mejora sería necesario agregar una batería, un microcontrolador para que controle los motores y tener en cuenta que tendrá una autonomía limitada por la carga de la batería.

Por ejemplo:

Figura 30. Plataforma de robot móvil

Fuente: https://http2.mlstatic.com/D_NQ_NP_755363-MLA41756510883_052020-W.jpg

El sistema móvil también se puede trabajar en conjunto con una cámara panorámica desde el techo y así determinar las coordenadas del brazo y movilizarse de manera autónoma o controlada. Este esquema puede ser muy favorable, pero requiere otra complejidad en sincronización y desarrollo.

9. Conclusión

Según los objetivos planteados al comienzo del proyecto se logró investigar lo que refiere al procesamiento de imágenes y visión artificial de manera satisfactoria. Se comprendieron los conceptos básicos y fundamentales para poder realizar un desarrollo aplicable al control automático de objetos.

Como resultado se obtuvo una aplicación híbrida, bastante robusta, intuitiva y simple de usar en donde el sistema puede realizar el procesamiento de imágenes obtenidas mediante vídeo en tiempo real y puede decidir por sí mismo. En el que cuenta con reglas específicas la forma en que selecciona los objetos y la manera de actuar. Si bien se puede mejorar su adaptación dependerá del ámbito en el que se lo utilice.

Por otro lado, se pudo realizar el desarrollo del sistema que cuenta con interfaz web para poder tener un control desde cualquier lugar con conexión a internet y un browser para el acceso.

Se debe destacar que si bien surgieron problemas y contratiempos producto del mismo desconocimiento de las tecnologías y herramientas utilizadas al principio, pudieron ser resueltos a medida que se avanzaba en el proyecto.

Para finalizar, el desarrollo en este tipo de tecnologías nos brinda un gran abanico de posibilidades de crecimiento en cualquier proceso del desarrollo de software y hardware. Es muy amplia la variedad de aplicaciones en que se puede utilizar esta combinación de tecnologías posibilitando la adaptación para el ámbito que lo requiera. Por lo tanto siempre puede aplicarse y adaptarse a nuevos desafíos tanto como a viejas problemáticas. En fin, para aquellas personas que quieran empezar en la investigación, el desarrollo y las mejoras de tecnologías IoT, Visión por computadora y robótica les espera un mundo de posibilidades para formarse y tener una excelente experiencia.

10.Bibliografía.

- **Fabricante de los procesadores ESP32. https://www.espressif.com/(20/06/2022)**
- **Bibliotecas de funciones open sourse para procesamiento de imágenes en python. https://docs.opencv.org/(20/06/2022)**
- **Lenguaje de programación de alto nivel. https://www.python.org/(20/06/2022)**
- **Entorno de desarrollo integrado "Arduino IDE". https://www.arduino.cc/ (20/06/2022)**
- **A public platform building the definitive collection of coding questions & answers. https://stackoverflow.com/(20/06/2022)**
- **"Qué es Arduino, cómo funciona y qué puedes hacer con uno" https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno (20/06/2022)**
- **¿Qué es OpenCV? Instalación en Python y ejemplos básicos https://www.inesem.es/revistadigital/informatica-y-tics/opencv/ (20/06/2022)**
- Modos o modelos de color HSB (o HSV) y códigos hexadecimales: qué son y usos específicos. https://www.comunicacion-multimedia.info/2010/05/modos-o-modelos-de-color-hsb-o-hsv-y.html **(20/06/2022)**
- **Documentación oficial OpenCV. "Thresholding Operations using inRange"https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html**
- **Documentación oficial OpenCV. "Canny Edge Detector". https://docs.opencv.org/3.1.0/da/d5c/tutorial_canny_detector.html**
- **Valverde Rebaza, Jorge. "Detección de bordes mediante el algoritmo de Canny". Universidad Nacional de Trujillo.**
- **Curso "Visión por computador con OpenCV y Python: Control de robots"**

https://www.udemy.com/course/control-de-robots-con-opencv-python-y arduino

- **Detección de objetos por colores en imágenes con Python y OpenCV https://medium.com/@gastonace1/detecci%C3%B3n-de-objetos-porcolores-en-im%C3%A1genes-con-python-y-opencv-c8d9b6768ff**
- Arduino en español.”Biblioteca de funciones software serial

  http://manueldelgadocrespo.blogspot.com/p/biblioteca.html **(13/12/2022)**

- **Proyectos de Ingeniería. “A platform for engineers & technical professionals to share their engineering projects, solutions & experiences with TEP Community & support open source” https://www.theengineeringprojects.com/(13/12/2022)**
- **The fundamental package for scientific computing with Python. https://numpy.org/ (13/12/2022)**
- **HIWonder. “Compañía de robots educativos”. https://www.hiwonder.hk/(13/12/2022)**

-