UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE GUAYAQUIL
CARRERA DE INGENIERÍA ELECTRÓNICA

"DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA IDENTIFICACIÓN DE COLORES DE LAS LUCES DE LOS SEMÁFOROS VEHICULARES MEDIANTE LA HERRAMIENTA TEACHABLE MACHINE DE GOOGLE"

Trabajo de titulación previo a la obtención del
Título de Ingeniero Electrónico

AUTOR: EDGAR RAMIRO PINTO ARMIJOS

TUTOR: ING. VICTOR DAVID LARCO TORRES MSC.

GUAYAQUIL – ECUADOR
2023

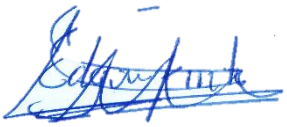
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA

Yo, **Edgar Ramiro Pinto Armijos** con documento de identificación N° 0931433130; manifestó que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Guayaquil, 15 de febrero del 2023

Atentamente,



Edgar Ramiro Pinto Armijos
0931433130

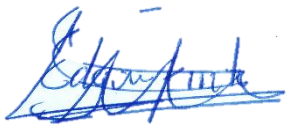
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR

Yo, **, Edgar Ramiro Pinto Armijos**, con documento de identificación N° 0931433130, expreso mi voluntad y por medio del presente documento de ceder a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales al ser el autor del trabajo de grado titulado: **"DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA IDENTIFICACIÓN DE COLORES DE LAS LUCES DE LOS SEMÁFOROS VEHICULARES MEDIANTE LA HERRAMIENTA TEACHABLE MACHINE DE GOOGLE"**, el cual ha sido desarrollado para optar por el título de: **INGENIERO ELECTRÓNICO**, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que realizo la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 15 de febrero del 2023

Atentamente,



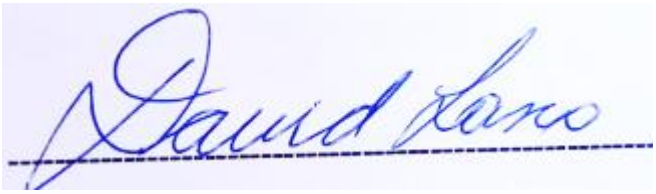
Edgar Ramiro Pinto Armijos
0931433130

CERTIFICADO DE DIRECCIÓN DE TRABAJO DE TITULACIÓN

Yo, **Víctor David Larco Torres.** con documento de identificación N° , docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **"DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA IDENTIFICACIÓN DE COLORES DE LAS LUCES DE LOS SEMÁFOROS VEHICULARES MEDIANTE LA HERRAMIENTA TEACHABLE MACHINE DE GOOGLE**", realizado por **Edgar Ramiro Pinto Armijos**, con documento de identificación N° 0931433130, obteniendo como resultado final el trabajo de titulación bajo la opción de trabajo de grado que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 15 de febrero del 2023

Atentamente,



Ing. Víctor David Larco Torres MSc.

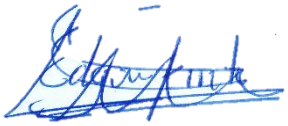
0912419611

DEDICATORIA

Mi título universitario se lo dedico

A Dios quien incentivo mi entrega y dedicación en el camino estudiantil.

A mis padres quienes fueron apoyo y parte fundamental de este grato proceso de formación cursado en este centro de educación superior.



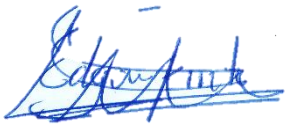
Edgar Ramiro Pinto Armijos
0931433130

AGRADECIMIENTO

La culminación del ciclo universitario embarga una serie de sentimientos encontrados tras tomar un momento y recordar todo lo vivido en cada una de las etapas cursadas a lo largo de la carrera.

A la planta docente quienes impartieron sus conocimientos y su guía, prestos a colaborar y despejar toda incógnita que surgía en el proceso de formación.
A la planta administrativa, personal capacitado con guía eficiente al atender y resolver todo tipo de tramites e inquietudes.

Compañeros de formación y carreras relacionadas a quienes tuve la oportunidad de conocer al cursar diversas materias, compartir la gestión, ejecución y desarrollo de proyectos en cada semestre mano a mano aportando de manera efectiva conocimientos conjuntos a fin de obtener los resultados deseados y más de una vez superando los mismos.



Edgar Ramiro Pinto Armijos

RESUMEN

AÑO	ALUMNOS	DIRECTOR DE PROYECTO TÉCNICO	TEMA DE PROYECTO TÉCNICO
2023	• Edgar Ramiro Pinto Armijos	Ing. Víctor David Larco Torres, MSc.	Diseño E Implementación De Un Prototipo Para La Identificación De Colores De Las Luces De Los Semáforos Vehiculares Mediante La Herramienta Teachable Machine De Google

El presente trabajo está en el desarrollo y diseño de un prototipo para la identificación de colores de semáforos vehiculares mediante la herramienta Teachable Machine de Google, es un proyecto que tiene como objetivo mejorar la seguridad en las carreteras. La herramienta Teachable Machine de Google permite entrenar a una red neuronal para identificar patrones específicos en una imagen. En este caso, se utiliza para identificar los colores de los semáforos vehiculares.

Este trabajo aporta a los estudiantes una herramienta de aprendizaje práctico ya que se implementa en una Raspberry Pi 4, un ordenador de bajo costo y bajo consumo, que cuenta con la potencia suficiente para ejecutar la herramienta Teachable Machine. Además, se utiliza la biblioteca de inteligencia artificial Keras, junto con otras bibliotecas como NumPy, Pygame y OpenCV, para crear un sistema capaz de identificar los colores de los semáforos vehiculares en tiempo real.

El presente documento en la parte de marco metodológico incluye los elementos que se tomaron en cuenta para la creación del proyecto, como instrumentación electrónica, diseño electrónico y programación en python.
A través del raspberry y teniendo como sistema operativo raspbian el prototipo resultante es un sistema capaz de identificar los colores de los semáforos vehiculares con una alta precisión. Además, al ser un sistema en tiempo real, puede ser utilizado para alertar a los conductores de los cambios en los semáforos, mejorando la seguridad en las carreteras.

El proyecto consta con una interfaz mediante una pantalla táctil instalada en el raspberry minimizando el tamaño del prototipo y siendo capaz de realizar lecturas a través de una cámara, permitiendo reproducir sonido acorde a la clasificación mediante el modelo entrenado en Teachable machine

PALABRAS CLAVES: PROTOTIPO, IDENTIFICACIÓN DE COLORES, SEMÁFOROS VEHICULARES, TEACHABLE MACHINE, RASPBERRY PI 4, BIBLIOTECA KERAS (IA), NUMPY, PYGAME, OPENCV, SISTEMA OPERATIVO RASPBIAN
.

ABSTRACT

YEAR	STUDENTS	DIRECTOR OF TECHNICAL PROJECT	TECHNICAL PROJECT THEME
2023	• Edgar Ramiro Pinto Armijos	Ing. Víctor David Larco Torres, MSc.	Design and Implementation of a Prototype for the Identification of Colors of Vehicle Traffic Light Using the Google Teachable Machine Tool

The present work is in the development and design of a prototype for the identification of colors of vehicular traffic lights using the Google Teachable Machine tool, a project aimed at improving road safety. The Google Teachable Machine tool allows for training a neural network to identify specific patterns in an image. In this case, it is used to identify the colors of traffic lights.

This work provides students with a practical learning tool as it is implemented on a Raspberry Pi 4, a low-cost, low-power computer that has sufficient power to run the Teachable Machine tool. Additionally, the artificial intelligence library Keras is used, along with other libraries such as NumPy, Pygame, and OpenCV, to create a system capable of identifying the colors of vehicular traffic lights in real-time.

In the present document, the methodological framework includes the elements that were taken into account for the creation of the project, such as electronic instrumentation, electronic design, and programming in python.

Through the raspberry and with the raspbian operating system, the resulting prototype is a system capable of accurately identifying the colors of vehicular traffic lights. Additionally, being a real-time system, it can be used to alert drivers of changes in the traffic lights, improving road safety.

The project has an interface through a touchscreen installed on the raspberry, minimizing the size of the prototype and being able to perform readings through a camera, allowing a sound to be played according to the classification through the model trained in Teachable Machine.

KEY WORDS: PROTOTYPE, COLOR IDENTIFICATION, VEHICLE TRAFFIC LIGHTS, TEACHABLE MACHINE, RASPBERRY PI 4, KERAS LIBRARY (AI), NUMPY, PYGAME, OPENCV, RASPBIAN OPERATING SYSTEM.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El proyecto está basado en la implementación y diseño de un prototipo para la identificación de colores en semáforos vehiculares utilizando la herramienta Teachable Machine de Google, teniendo como objetivo mejorar la seguridad en las carreteras al identificar los cambios en los semáforos en tiempo real y alertar a los conductores.

Teniendo como interlocutor para el entrenamiento basado en inteligencia artificial la herramienta Teachable Machine permite la creación de un modelo para identificar patrones específicos en una imagen y en este caso se utiliza para identificar los colores de los semáforos vehiculares

El proyecto está enfocado en los estudiantes de la carrera de Ingeniería en Electrónica de la Universidad Politécnica Salesiana Sede Guayaquil, por lo cual se utiliza una tarjeta Raspberry pi 4, que es un ordenador de bajo costo y consumo energético amigable a aplicaciones de prototipado que trabaja en el lenguaje Python usado en materias enfocadas a microcontroladores y sistemas embebidos. Sirviendo como base para la implementación de proyectos que deriven en otros sistemas de clasificación a objetos, personas, etc.

El objetivo principal de este proyecto es implementar un prototipo que integre las herramientas de inteligencia artificial de Google y los modelos resultantes con un sistema embebido permitiendo ejecutar bibliotecas como NumPy, Pygame y OpenCV para crear un sistema capaz de identificar los colores de los semáforos vehiculares en tiempo real.

Se propone una interfaz mediante una pantalla táctil instalada en la Raspberry Pi, lo que permite minimizar el tamaño del prototipo y realizar lecturas a través de una cámara. Además, se reproduce un sonido acorde a la clasificación mediante el modelo entrenado en Teachable Machine.

1. EL PROBLEMA

1.1. Antecedentes

Es muy bien conocido en la actualidad el índice de accidentes de tránsito dentro de ciudades con alta demanda poblacional, vehicular y la falta de cultura ante el respeto de las señales de tránsito, entre el caos, la contaminación visual y sonora un conductor o peatón puede ser el sujeto que ocasione un siniestro consecuente a tres factores principales, siendo estos: imprudencia del conductor por distracción, irrespeto a las señales de tránsito y el exceso de velocidad. (INEC, 2021), (Agencia Nacional de Tránsito, 2022). Por tal ha llevado proponer, diseñar y ensamblar un dispositivo capaz de obtener lectura de colores de semáforos vehiculares en ambiente urbano mediante la identificación de imágenes a través de machine learning como propuesta a su uso en medida preventiva dentro de vehículos operando como alerta ante la posible distracción del conductor.

1.2. Importancia y Alcances

La identificación de colores en semáforos vehiculares es una tarea importante para la seguridad vial. Los semáforos vehiculares tienen tres luces: roja, amarilla y verde, cada una de las cuales indica a los conductores qué deben hacer (detenerse, prepararse para detenerse o seguir avanzando). Sin embargo, a veces los semáforos pueden fallar o tener luces dañadas, lo que puede provocar confusión entre los conductores y aumentar el riesgo de accidentes.
Los beneficiarios de un sistema de identificación de colores en semáforos vehiculares mediante machine learning son:
Conductores: el sistema podría proporcionar información precisa y en tiempo real sobre el color de las luces de los semáforos vehiculares, lo que podría ayudar a los conductores a tomar decisiones de manera más segura y evitar accidentes.
Peatones: el sistema podría proporcionar información sobre el estado de los semáforos para peatones, lo que ayudaría a los peatones a tomar decisiones de manera más segura al cruzar la calle.
Autoridades de tránsito: el sistema podría proporcionar información sobre el funcionamiento de los semáforos vehiculares, lo que podría ayudar a las autoridades de tránsito a detectar y solucionar problemas con los semáforos de manera más eficiente.
Fabricantes de semáforos: el sistema podría proporcionar información sobre el rendimiento de los semáforos vehiculares, lo que podría ayudar a los fabricantes a mejorar la calidad y la durabilidad de sus productos.

1.3. Delimitación

1.3.1. Temporal

Se considera un periodo de seis meses distribuidos entre el estudio, recopilación de datos, desarrollo, pruebas de funcionamiento e implementación del prototipo a fin de brindar un producto terminado.

1.3.2. Espacial

Se enfoca hacia una sociedad a la cual se busca integrar, socializar y familiarizar con el prototipo recopilando datos de satisfacción y mejoras al producto a fin de brindar la mejor experiencia.

1.3.3. Académica

El proyecto contara con lo solicitado por la Universidad Politécnica Salesiana referente a los grados de investigación y desarrollo conjunto a los sustentos requeridos e inmersión de las asignaturas impartidas a lo largo de la trayectoria de grado.

1.4. Innovación

La innovación en este proyecto radica en la creación de un prototipo capaz de identificar los colores de los semáforos vehiculares en tiempo real mediante la herramienta Teachable Machine de Google y una raspberry pi.

2. Objetivos

2.1. Objetivo general

- Diseñar e Implementar un prototipo para identificación de colores en semáforos vehiculares.

2.2. Objetivos específicos

- Elaborar el universo de muestras para el entrenamiento a través de fotos de varios semáforos en diferentes estados.
- Elaborar las clases para el desarrollo de la clasificación mediante la herramienta de Teachable machine.
- Implementar un algoritmo de código que permita el uso de modelos para el machine learning
- Probar el modelo de machine learning en diferentes condiciones de iluminación y entornos de tráfico para evaluar su robustez.

3. FUNDAMENTOS TEÓRICOS

3.1. Inteligencia artificial

La inteligencia artificial (IA) es una rama de la informática que se centra en la creación de sistemas que pueden realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje, la resolución de problemas, la toma de decisiones y más (Norvig, 2010). La inteligencia artificial se logra a través de algoritmos y técnicas de aprendizaje automático que permiten que los sistemas mejoren su rendimiento con más datos y experiencia (Norvig, 2010).

Los sistemas de IA se dividen en tres amplias categorías: IA débil, IA fuerte e IA ultradensa (Y. LeCun, 2015). La IA débil es un tipo de IA que se limita a realizar ciertas tareas, mientras que la IA fuerte se refiere a la capacidad del sistema para realizar cualquier tarea intelectual que un ser humano pueda realizar (Negnevitsky, 2016). La IA súper intensiva es un tipo más avanzado de IA fuerte que involucra la capacidad de los sistemas para realizar tareas intelectuales de manera más eficiente que los humanos (Pearl, 2017).

La inteligencia artificial tiene una amplia gama de aplicaciones, desde la automatización de tareas hasta la mejora de la eficiencia y la productividad en diversas industrias (Alpaydin, 2011). Sin embargo, también plantea algunos desafíos éticos y morales, como la protección de datos y el impacto en el trabajo humano.

3.1.1. Usos de la inteligencia artificial en ingeniería

La inteligencia artificial se utiliza en una variedad de aplicaciones técnicas que incluyen:

3.1.1.1. Análisis y diagnóstico de datos

En ingeniería se puede utilizar la inteligencia artificial para analizar grandes cantidades de datos y descubrir patrones y tendencias que pueden no ser visibles a simple vista. Se puede utilizar en la industria aeronáutica, automotriz, de petróleo y gas, et La inteligencia artificial se utiliza en una variedad de aplicaciones técnicas

3.1.1.2. Control de procesos y optimización

La IA se utiliza para monitorear y controlar procesos industriales en tiempo real, permitiendo una optimización más eficiente y una mejora en la eficiencia energética.

3.1.1.3. Diseño y fabricación

La IA se utiliza en la generación automática de diseños y en la optimización de procesos de fabricación. Esto incluye la generación de prototipos virtuales y la simulación de procesos de fabricación.

3.1.1.4. Predicción de fallos

La IA se utiliza para predecir fallos en equipos y sistemas antes de que ocurran, lo que permite una planificación más eficiente y una reducción en el tiempo de inactividad.

3.1.1.5. Robótica

La IA se utiliza en la robótica para el desarrollo de robots autónomos y para mejorar la eficiencia y la precisión en tareas repetitivas.

3.2. Redes Neuronales

Ya en 1911, con los primeros estudios de Santiago Ramón y Cajal, las neuronas se definieron como el componente básico del sistema nervioso, incluyendo al cerebro, el cual está formado por un enorme número de ellas, existen numerosos estudios que muestran que estas neuronas están conectadas de forma masiva entre sí, y aunque esta cualidad está presente en todos los animales es mucho más acentuada y evidente en los seres humanos, característica que permite ser inteligentes y diferencia del resto de seres vivos (Hernández, 2010).

3.3. Sistemas embebidos

Un sistema embebido es un sistema electrónico diseñado para realizar una tarea específica e integrado en un dispositivo o producto, son básicamente pequeñas computadoras que se integran en otros dispositivos para proporcionar ciertas funciones (Furber, 2010).

Estos sistemas constan de hardware y software y pueden incluir microcontroladores, sensores, actuadores, memoria y dispositivos de entrada/salida (Liu, 2017).

El software que se ejecuta en un sistema integrado puede ser un sistema operativo o un conjunto de aplicaciones programadas para realizar tareas específicas (Jain, 2011).

Los sistemas integrados se utilizan en una amplia gama de aplicaciones, desde productos electrónicos de consumo hasta equipos industriales y médicos (Jain, 2011)..

Por ejemplo, los sistemas integrados se encuentran en televisores, reproductores de MP3, teléfonos móviles, automóviles, equipos de control industrial, sistemas de automatización de edificios, etc. (Jain, 2011).

La principal ventaja de los sistemas integrados es su capacidad para integrarse con otros productos y dispositivos para proporcionar una funcionalidad específica (Alur, 2013).

Esto los hace altamente eficientes y útiles en una amplia gama de aplicaciones, lo que les permite realizar tareas específicas de forma rápida y precisa (Gill, 2011).

3.4. Raspberry pi

La Raspberry Pi es una microcomputadora de bajo costo desarrollada en el Reino Unido por la Fundación Raspberry Pi. Lanzado en febrero de 2012, tiene como objetivo mejorar aún más la enseñanza de las ciencias de la computación y facilitar que los estudiantes aprendan a codificar (Foundation, Raspberry Pi Foundation. , 2023).(figura 1)

Figura 1: Raspberry pi (Foundation, Raspberry Pi Foundation. , 2023).

Raspberry Pi es un dispositivo pequeño y compacto que parece una placa base y se puede conectar a un monitor, teclado y mouse para actuar como una computadora completa (Upton, 2012).

Además, cuenta con una variedad de opciones de entrada y salida como puertos USB, HDMI, Ethernet y GPIO, lo que lo hace compatible con una amplia gama de periféricos y dispositivos externos (W. R. Stevens, 2014).

Raspberry Pi se ejecuta en un sistema operativo basado en Linux y se puede utilizar en una amplia variedad de aplicaciones, desde servidores de construcción hasta proyectos de robótica y automatización del hogar (W. R. Stevens, 2014). Además, también se puede utilizar para aprender a programar en lenguajes como Python y Scratch (Qazi., 2016).

En resumen, Raspberry Pi es una microcomputadora versátil y asequible que es muy popular entre los entusiastas de la tecnología y los educadores de todo el mundo (Cavanagh., 2015).

3.5. Tipos de Raspberry pi

3.5.1. Raspberry Pi Model B

Este es el modelo original de Raspberry Pi que se lanzó en 2012. Tiene un procesador ARM1176JZF-S de 700 MHz, 256 MB de RAM, conectividad Ethernet y un puerto USB (Foundation., Raspberry Pi Model B., 2012).

3.5.2. Raspberry Pi 2 Modelo B

Lanzado en febrero de 2015, este modelo ofrece más potencia de procesamiento gracias a su procesador ARM Cortex-A7 de cuatro núcleos a 900 MHz y 1 GB de RAM (Foundation, Raspberry Pi 2 Model B., 2015).

3.5.3. Raspberry Pi 3 Modelo B

Lanzado en febrero de 2016, este modelo tiene un procesador de cuatro núcleos ARM Cortex-A53 de 1,2 GHz, 1 GB de memoria y conectividad inalámbrica integrada. Raspberry Pi 3 Model B+: esta es una versión mejorada de Raspberry Pi 3 Model B, equipada con un procesador de cuatro núcleos ARM Cortex-A53 de 1,4 GHz para

una transferencia de datos más rápida y una mejor eficiencia energética (Foundation., Raspberry Pi 3 Model B. , 2016).

3.5.4. Raspberry Pi 4 Modelo B:

Es el último modelo de Raspberry Pi lanzado en junio de 2019. Ofrece una amplia gama de funciones, incluido un procesador de cuatro núcleos ARM Cortex-A72 de 1,5 GHz, hasta 8 GB de RAM y una GPU VideoCore VI (Foundation, Raspberry Pi 4 Model B, 2019).

3.6. Sistema Operativos y sus principales orígenes

Un sistema operativo (SO) es un software que administra los recursos de la computadora y proporciona una interfaz para que los usuarios y los programas interactúen con el hardware. Los sistemas operativos Raspberry Pi incluyen una variedad de opciones, cada una con sus propias características y especificaciones. Estos son algunos de los sistemas operativos Raspberry P más populares (M. Awan, 2018)

3.6.1. Raspbian

Es un sistema operativo basado en Debian optimizado para Raspberry Pi. Es una de las opciones de Raspberry Pi más populares y se usa ampliamente en proyectos educativos y de pasatiempos (A. I. Alsarawi, 2019).

3.6.2. Ubuntu

Es un sistema operativo de código abierto basado en Debian. Hay varias versiones de Raspberry Pi disponibles, incluidas Ubuntu Mate, Ubuntu Server y Xubuntu (A. I. Alsarawi, 2019)..

3.6.3. Windows 10 IoT Core

Es una versión reducida dew Windows 10 optimizada para internet de las cosas y dispoaitivos integrados como Rapsberry Pi (Kaur, 2019)

3.6.4. LibreELEC

Es un sistema operativo de centro de entretenimiento de código abierto basado en Kodi. Esta es una opción popular para los usuarios que desean convertir su Raspberry Pi en un centro multimedia (Kaushik, 2018).

3.6.5. OSMC

Es otro sistema basado en Kodi optimizado para Raspberry Pi. Ofrece amplias funciones de reproducción multimedia, incluida compatibilidad con varios formatos de audio y video. (Tyagi, 2018)

3.7. Lenguajes de Programación utilizados en Raspberry Pi

3.7.1. Python:

Uno de los lenguajes de programación más populares y ampliamente utilizados para la Raspberry Pi. Es fácil de aprender y cuenta con una gran cantidad de librerías y módulos para aplicaciones específicas como domótica, computación y visión artificial (H. Rahman, 2019).

3.7.2. C/C++:

Estos lenguajes de programación se utilizan en aplicaciones que consumen muchos recursos, como juegos, sistemas integrados y control de motores. Java: Es un lenguaje de programación orientado a objetos utilizado para desarrollar aplicaciones de software y dispositivos móviles (H. Rahman, 2019)..

3.7.3. Scratch:

Es un lenguaje de programación especialmente diseñado para niños y principiantes a la programación. Se utiliza para enseñar conceptos básicos de programación y resolución de proble
mas (H. Rahman, 2019)..

3.7.4. Ruby:

Es un lenguaje de programación orientado a objetos que se utiliza para desarrollar aplicaciones web y de servidor (H. Rahman, 2019)..

3.8. Protocolo de comunicación SPI

El protocolo SPI (Interfaz periférica en serie) es un estándar de comunicación en serie de alta velocidad para conectar dispositivos periféricos a un microcontrolador o sistema host (Sevcik, 2020).

Tal como se ilustra en la figura 2 SPI es un protocolo de comunicación full-duplex, lo que significa que permite que los datos se transfieran en ambas direcciones al mismo tiempo. Es un protocolo muy versátil con una amplia gama de aplicaciones, incluidos sistemas de control industrial, sistemas de automatización, dispositivos de almacenamiento de datos, sistemas de audio y video, y más (Patel, 2019).

SPI es conocido por ser un protocolo de comunicación rápido y confiable. Además, es fácil de implementar y requiere pocos componentes externos, lo que lo hace ideal para aplicaciones con requisitos de recursos limitados (Bednarek, 2012).

En el protocolo SPI, un dispositivo es el "maestro" y los otros dispositivos son los "esclavos". El dispositivo maestro controla el flujo de datos y coordina la comunicación entre los dispositivos esclavos (Chen, 2013).

Cada dispositivo esclavo tiene una dirección única y el dispositivo maestro puede seleccionar y comunicarse individualmente con cualquier dispositivo esclavo (Aggarwal, 2015).

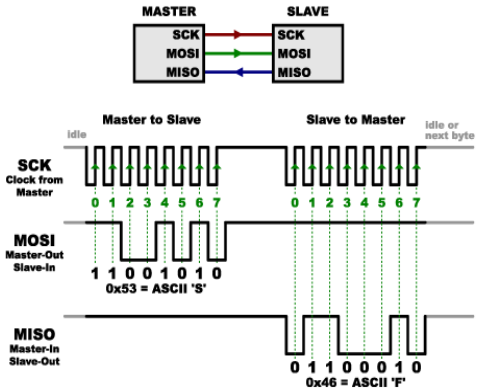


Figura 2: Protocolo Spi de comunicación (Sparkfun, 2018).

3.9. Protocolo de comunicación I2C

I2C (Circuito Interintegrado) es un protocolo de comunicación de baja velocidad diseñado para permitir que los dispositivos electrónicos se comuniquen entre sí (Teslabem, 2017). Desarrollado por Philips Semiconductors (ahora NXP Semiconductors) en 1982, se ha convertido en un estándar ampliamente utilizado para la comunicación entre microcontroladores, sensores, dispositivos de almacenamiento de datos y otros componentes electrónicos (Teslabem, 2017)..

El protocolo I2C utiliza dos líneas de comunicación: SDA (línea de datos en serie) y SCL (línea de reloj en serie) para transferir datos y sincronizar la transmisión. Cada dispositivo en una red I2C tiene una dirección única que permite que los dispositivos se comuniquen uno a uno o en grupos (Teslabem, 2017).. Además, el protocolo I2C admite una amplia gama de velocidades de transferencia de datos, lo que lo hace adecuado para una amplia gama de aplicaciones (Teslabem, 2017).

En pocas palabras, el protocolo I2C es una forma eficiente y efectiva de permitir que los dispositivos electrónicos se comuniquen entre sí, lo que lo hace ideal para una amplia variedad de aplicaciones, incluidos los sistemas de control industrial, los sistemas de automatización, los dispositivos de almacenamiento de datos y más.

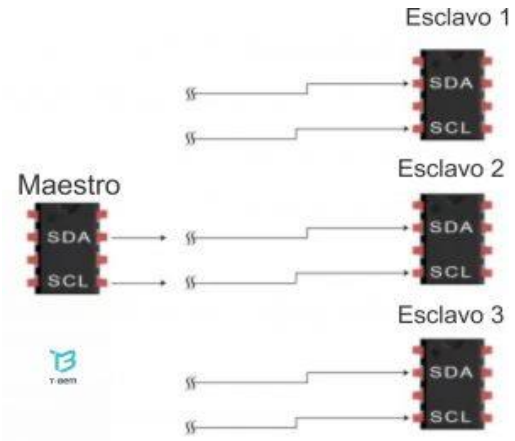


Figura 3: Protocolo i2c (Teslabem, 2017).

3.10. Cámara Raspberry

La cámara Raspberry Pi es un dispositivo de captura de imágenes y videos que se conecta a la placa de desarrollo de Raspberry Pi para ampliar sus capacidades de captura de imágenes y videos (Lai, 2020).

Es un componente externo que se conecta a la placa mediante un conector especial que le da acceso a la placa a los datos capturados por la cámara. Son compatibles con una variedad de sistemas operativos y software de visión artificial, lo que permite a los usuarios crear soluciones personalizadas para sus proyectos (Wang, 2018).

En resumen, la cámara Raspberry Pi es un componente valioso que amplía las capacidades de la placa Raspberry Pi y permite a los usuarios crear soluciones personalizadas de captura de imágenes y videos para sus proyectos (Xu, 2018).

Figura 4: Cámara de raspberry (Xu, 2018).

3.11. Patrón de colores

Un patrón de color es una combinación de colores que se repite sistemáticamente. Se pueden utilizar para objetos de diseño, páginas web, ilustraciones y más. Los patrones de color se pueden dividir en tres grandes categorías: patrones

monocromáticos que usan un solo color, patrones análogos como tonos similares del mismo color y patrones complementarios que son colores opuestos de la paleta (Zhang, 2018). La combinación de estas tres categorías puede crear patrones de color únicos y divertidos.

3.11.1. Patrón RGB

Es un modelo de color utilizado en tecnología y diseño para crear diferentes colores. El concepto se basa en el trabajo de Thomas Young en 1802, quien describió el uso de tres colores primarios de luz para crear color. Estos colores primarios son el rojo, el verde y el azul, y al combinar en diferentes proporciones permite crear una gama de colores (Elvins, 2009).

Figura 5: Patrón RGB (Elvins, 2009).

3.11.2. Patrón CMYK

Los modelos de color CMYK se utilizan en el diseño gráfico para crear impresiones de alta calidad. Consta de cuatro colores cian (C), magenta (M), amarillo (Y) y negro (K). Se llama "ciclo de cuatro colores" o "ciclo de tinta de cuatro colores" porque estos son los colores que se utilizan para imprimir. La mezcla de colores CMYK produce una amplia gama de colores, desde colores brillantes hasta pasteles suaves (Lam, 2015).

Figura 6: Patrón CMYK (Lam, 2015).

3.11.3. Patrón HSL

El modo HSL (tono, saturación, luminosidad) es una representación del color basada en términos matemáticos. El modelo HSL divide los colores en tres elementos principales: matiz, saturación y luminosidad. Estos tres elementos se combinan para formar colores visibles (Pugh, Finlayson, & Drew, 2015).

El tono se refiere a la percepción del color y se mide en el modelo HSL utilizando una escala de 0 a 360 grados (Jun, Lee, & Baik, 2016). Esta escala representa la luz visible en un espectro, lo que significa que 0 es rojo, 60 es amarillo, 120 es verde, 180 es cian, 240 es azul, 300 es magenta y 360 es rojo (Jun, Lee, & Baik, 2016)..

La saturación se refiere a qué tan "vibrante" es un color y se mide en una escala de 0 a 100%. Esto significa que 0% es gris claro y 100% es el color más oscuro.

Finalmente, el brillo se refiere al nivel de brillo, también medido en una escala de 0 a 100%. En esta escala, 0 % es negro, 50 % es brillo normal y 100 % es blanco.

En pocas palabras, los modelos HSL se utilizan para crear colores específicos utilizando tres elementos básicos: tono, saturación y luminosidad. Estos elementos se combinan para crear los colores que se observa en el ambiente (Jun, Lee, & Baik, 2016).

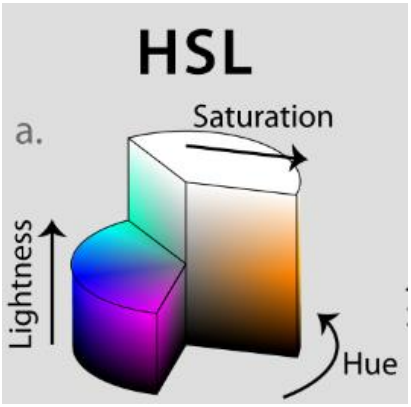


Figura 7: Patrón HSL (Jun, Lee, & Baik, 2016).

3.11.4. Patrón HSV

El esquema HSV (tono, saturación, valor) es un método de codificación de colores para mostrar y manipular imágenes. El modo HSV se basa en un modelo cilíndrico que utiliza tres variables cíclicas: matiz, saturación y luminosidad (Jun, Lee, & Baik, 2016).

La saturación es una medida de la pureza de un color. Cuanto mayor sea la saturación, más pura será la muestra. Este valor es una medida del brillo de un punto obtenido aplicando una curva logarítmica a la luz. Esto significa que cuanto mayor sea el valor, mayor será el brillo (Jun, Lee, & Baik, 2016)..

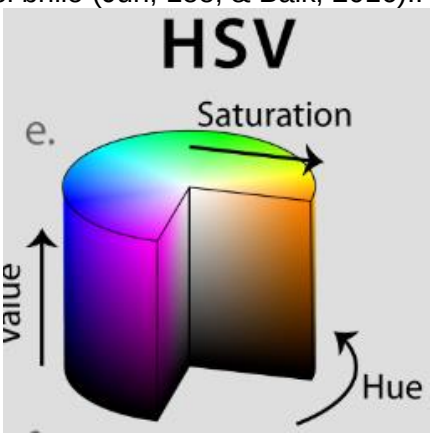


Figura 8: Patrón HSV (Jun, Lee, & Baik, 2016).

3.12. Google Teachable Machine

Google Teachable Machine es una herramienta basada en la web desarrollada por Google que permite que cualquier persona enseñe a su computadora a reconocer imágenes, sonidos y gestos (Machine, 2022).
La herramienta es fácil de descargar y usar, lo que permite a los usuarios crear y probar sus modelos aprendidos y exportarlos para usarlos en proyectos web, de aplicaciones y de hardware (Ahuja., 2022).

3.13. Tensor FLOW

TensorFlow es una biblioteca de código abierto de aprendizaje automático y Deep learning desarrollada por Google (Ortiz, 2019). Es una de las bibliotecas de aprendizaje profundo más populares y ampliamente utilizadas en la industria y la academia (Chilán Carrasco, 2020). TensorFlow permite a los desarrolladores y científicos de datos crear y entrenar modelos de aprendizaje automático en una amplia gama de aplicaciones, incluyendo visión por computadora, procesamiento del lenguaje natural y reconocimiento de patrones (Serrano Arenas, 2020). La biblioteca está diseñada para ser altamente escalable, lo que significa que puede ser utilizada en entornos de alta capacidad de procesamiento, incluyendo clúster de computadoras y dispositivos móviles (Jimenez Rubalcaba, 2016).

3.14. OpenCV

OpenCV es una biblioteca de visión por computadora de código abierto que permite la realización de tareas relacionadas con la visión artificial (Branque Rosales, 2022). Fue desarrollada por Intel y es ampliamente utilizada en aplicaciones de seguimiento de objetos, detección de rostros y reconocimiento de patrones, entre otras (García Chang, 2019). Con OpenCV, los desarrolladores pueden acceder a una amplia gama de algoritmos y herramientas para manipular y analizar imágenes y videos, lo que les permite crear aplicaciones sofisticadas de visión por computadora (Viera Maza, 2017).

4. MARCO METODOLÓGICO

4.1. Diseño del prototipo

En el desarrollo del proyecto de implementación se optó por el uso de un sistema embebido Raspberry pi 4 como etapa de procesamiento, con un sistema operativo raspbian. Teniendo diferentes etapas primero la etapa de adquisición donde se obtiene la imagen del mundo exterior que se conectará a la raspberry para su procesamiento posteriormente sigue la etapa de visualización donde se muestra la adquisición de imagen y una imagen del color de semáforo que este en el semáforo y al finalizar reproducirá en la etapa de sonido un archivo con el nombre del color que reconoció el programa. Todo el sistema será independiente ya que está conectado a un sistema de Power bank o fuente recargable como se detalla en la Figura 3-1.

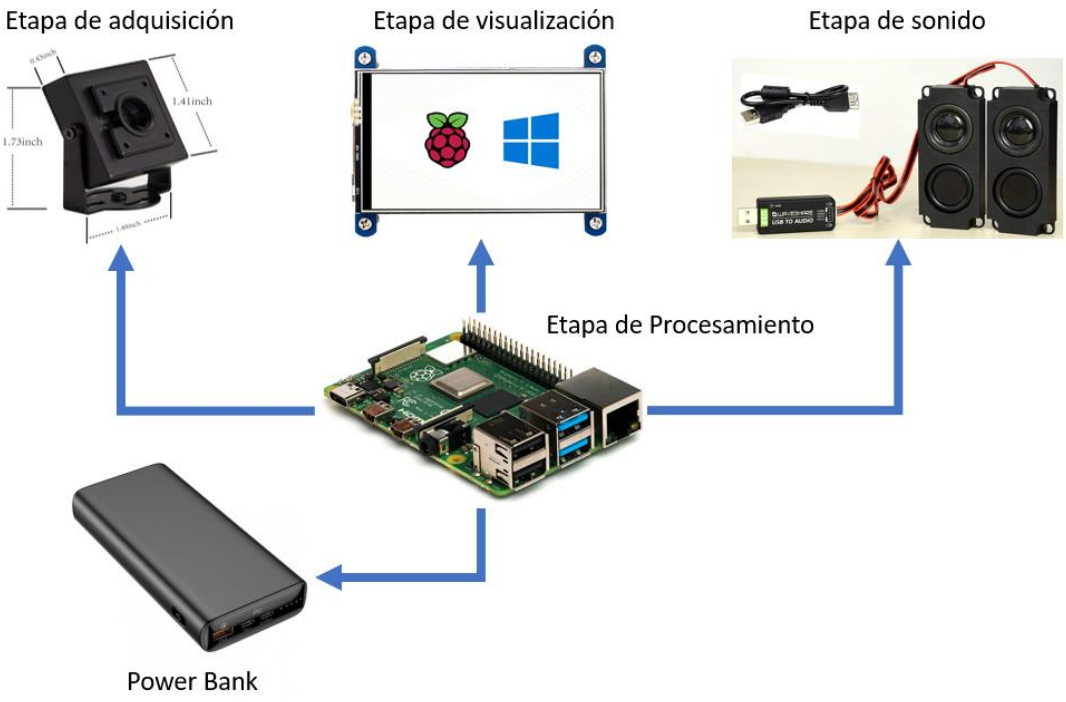


Figura 9: Diseño del prototipo

4.2. Diagrama de conexiones del proyecto

En la Figura 3-2 se detalla las conexiones del proyecto desde la parte central que es el sistema embebido raspberry pi, en conjunto con dos periféricos USB siendo uno el arduCam y el otro el módulo para sonido, también conectándose vía HMI paralelo a la pantalla táctil y todo siendo alimentado desde la raspberry que se conecta a la fuente recargable.

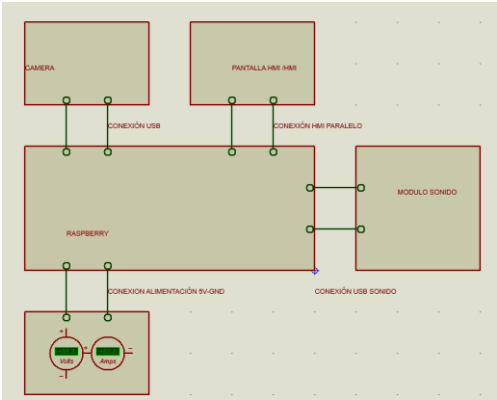


Figura 10: Conexiones del prototipo

4.3. Entorno de programación

En la Figura 3-2 se muestra el entorno de programación que es el **Thonny Python**, un componente que viene ya instalado en el sistema operativo raspbian el cual tiene una versión de Python 3.9.2.

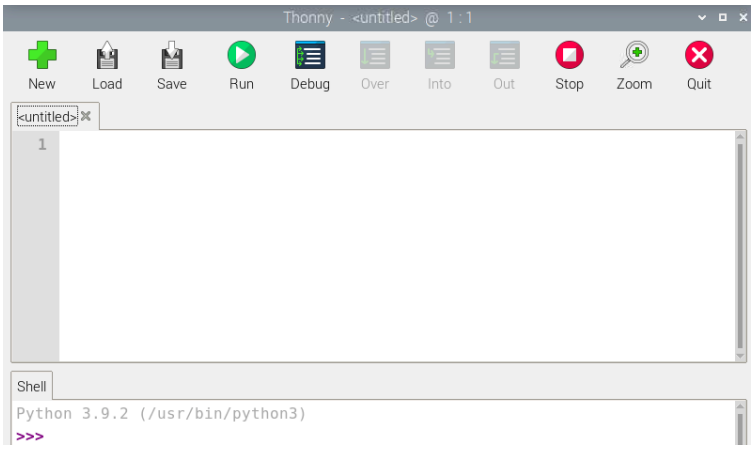


Figura 11: Entorno de programación Thonny Python

4.4. Etapa de adquisición

Para la etapa de adquisición es necesario la instalación de ciertos paquetes mediante la consola de la raspberry que permiten la toma de valores desde la cámara USB y su post procesamiento.
A continuación, se detallan los complementos a instalar para la ejecución del proyecto de implementación.

4.4.1. Instalación de Driver de camera

En las Figura 3-4 y 3-5 se detallan los tipos de conexiones de las cámaras a la raspberry teniendo como conexión primaria al sócalo integrado en la placa y como secundaria a un USB en la placa.

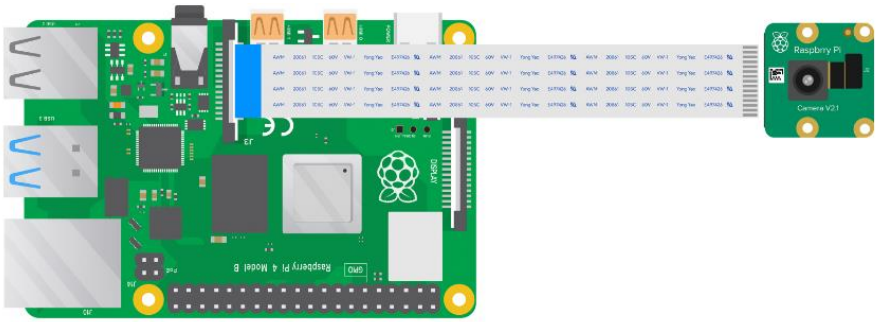


Figura 12: Conexiones de cámara con zócalo en placa (ArduCAM, 2022)

Figura 13: Conexiones en placa mediante USB para Cámara (pIHUT, 2021)

En la instalación de los controladores para la Cámara se procede a ejecutar los comandos de la Figura 3-6 en el terminal de la raspberry para la obtención del archivo de instalación del repositorio en GitHub (es necesario estar conectado a una red con acceso a internet).

```
wget -O install_pivariety_pkgs.sh https://github.com/ArduCAM/Arducam-Pivariety-V4L2-Driver
chmod +x install_pivariety_pkgs.sh
```

Figura 14: Comando primario para obtención de repositorio de GitHub (ArduCAM, 2022)

En la Figura 3-7 se procede a ejecutar en consola el comando para la instalación de libcamera comando que sirve para el llamar el objeto que ejecutara la adquisición de la imagen.

```
./install_pivariety_pkgs.sh -p libcamera_dev
```

Figura 15: Comando de instalación del objeto Libcamera (ArduCAM, 2022)

En la Figura 3-8 se procede a ejecutar en consola el comando para la instalación de los complementos de la librería libcamera .

```
./install_pivariety_pkgs.sh -p libcamera_apps
```

Figura 16: Comando de instalación del complemento Libcamera (ArduCAM, 2022)

En la Figura 3-9 se procede a ejecutar en consola el comando para la instalación de las librerías de todas las cámaras permitidas en la libcamera.

```
# PivarietyPlease se
./install_pivariety_pkgs.sh -p kernel_driver
```

Figura 17: Comando de instalación driver de cámaras para raspberry (ArduCAM, 2022)

4.4.2. Instalación de librería de procesamiento OPENCV

En la Figura 3-10 se verifica el prerrequisito para trabajar con las librerías de OpenCV el cual solo se ejecuta en sistemas de 64bits teniendo que ejecutar las siguientes líneas **cat /etc/os realese.**

Figura 18:: Verificación del sistema de 64bits para OpenCV (omes-va, 2019)

Posterior a la verificación se procede a ejecutar el comando **sudo apt update** y luego el comando **sudo apt upgrade** para la actualización de las librerías y dependencias en las raspberry, al finalizar estos comandos se ejecuta el **sudo apt install python3-opencv** como se muestra en la Figura 3-11 de la instalación del OpenCV.

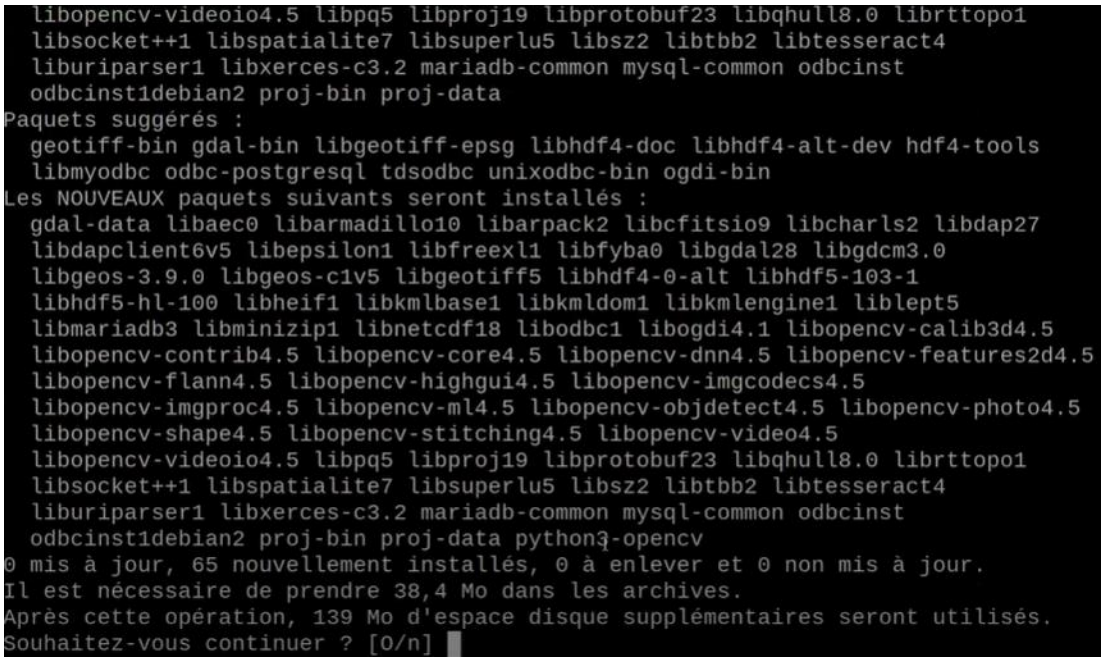


Figura 19: Instalación del OpenCV (omes-va, 2019)

4.4.3. Adquisición de video por camera

Para la adquisición de la imagen por la cámara se crea un archivo en el compilador Thonny con formato Py donde primero se llama a la instancia de la librería denominada cv2, se configura que el objeto tenga acceso a la cámara USB y se define el tamaño de la imagen a adquirir.

```
1 import cv2
2 # Inicia la captura de video desde la cámara
3 captura = cv2.VideoCapture(0)
4 # Configura el tamaño de la ventana de vídeo
5 captura.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
6 captura.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
```

Figura 20: Configuración de objeto cv2 y adquisición por Cámara USB

Se crea una instancia de repetición **while** , donde se procede a guardar los datos que estén en la cámara con la línea **captura.read()** y mostrarlos mediante el comando **cv2.inshow()**.

```
while True:
    # Lee el siguiente frame
    ret, frame = captura.read()
    # Muestra el frame en una ventana
    cv2.imshow("Captura de video", frame)
    # Si se presiona la tecla 'q', detiene la captura
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Libera los recursos de la cámara
captura.release()
# Cierra la ventana de vídeo
cv2.destroyAllWindows()
```

Figura 21: Ciclo while para obtener imagen de la cámara USB

Figura 22: Vista del video a través de librería OpenCV

4.5. Etapa de visualización

Para la etapa de visualización es necesario la instalación de ciertos paquetes mediante la consola de la raspberry que permiten el uso de una pantalla HMI o una pantalla TFT en modo bus paralelo.
A continuación, se detallan los complementos a instalar para la ejecución del proyecto de implementación.

4.5.1. Instalación de Driver de pantalla

En las Figura 3-15 y 3-16 se detallan los tipos de conexiones de las pantallas mediante el bus paralelo a la raspberry teniendo como conexión primaria al sócalo integrado en la placa y como secundaria a la conexión mediante el puerto HMI en la placa.

Figura 23: Conexiones de pantalla tft con zócalo en placa (ArduCAM, 2022)

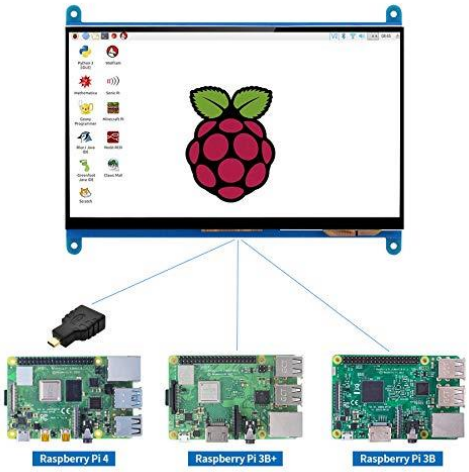


Figura 24: Conexiones en placa mediante HMI (pIHUT, 2021)

En la instalación de los controladores para la pantalla tft se procede a ejecutar los comandos de la Figura 3-17 en el terminal de la raspberry para la obtención del archivo de instalación del repositorio en GitHub (es necesario estar conectado a una red con acceso a internet).

```
git clone https://github.com/goodtft/LCD-show.git
```

Figura 25: Comando para descarga de repositorio (Altaruru, 2021)

En la Figura 3-18 se procede a ejecutar en consola el comando para la instalación de los complementos de la librería descargada del repositorio.

```
sudo apt install git
```

Figura 26: Comando para instalación de repositorio (Altaruru, 2021)

En la Figura 3-9 se procede a ejecutar en consola el comando para la instalación de los controladores de la pantalla paralela.

```
sudo chmod -R 755 LCD-show
```

Figura 27: Comando para instalación de controladores (Altaruru, 2021)

En la Figura 3-20 se procede a ejecutar en consola el comando para visualizar desde la pantalla paralela y que el puerto HMI entre como secundario.

```
cd LCD-show/
```

Figura 28: Comando para habilitar visualizador en pantalla tft (Altaruru, 2021)

En la Figura 3-21 se procede a ejecutar en consola el comando para la habilitación del controlador de acuerdo con el modelo de pantalla.

```
sudo ./LCD35-show
```

Figura 29: Habilitación de acuerdo el modelo: (Altaruru, 2021)

4.6. Etapa de sonido

Para la etapa de sonido se utilizan las librerías denominadas **Pygame**, ya que permiten la creación, carga de imágenes y sonidos que pueden ser puesto en marcha mediante comandos. Las librerías vienen incluidas en versiones superiores a Python 2.9.2.

En la Figura 3-22 se detallan las líneas de comandos necesarias para la carga de un archivo en formato mp3 y luego la reproducción de sonido de este mediante consola.

```
1  import pygame
2  # Inicializa pygame
3  pygame.init()
4  while True:
5      pygame.mixer.music.load("sonido.mp3")
6      pygame.mixer.music.play()
```

Figura 30: Ejemplo de carga de archivo y reproducción de sonido (Altaruru, 2021)

4.7. Etapa de procesamiento

En la etapa de procesamiento es necesario realizar previamente el entrenamiento del modelo en formato Keras con las herramientas de Google Teachable machine, a continuación, se detallan los pasos para el entrenamiento y posterior código para realizar la clasificación con el modelo entrenado y lo que este captando la cámara de video.

4.7.1. Entrenamiento mediante Teachable machine

En la etapa de procesamiento es necesario realizar previamente tener una cuenta en **https://teachablemachine.withgoogle.com/**, seleccionar en la barra superior izquierda la opción de nuevo proyecto y se muestra las opciones de la Figura 3-23 donde se configura el tipo de clasificación.

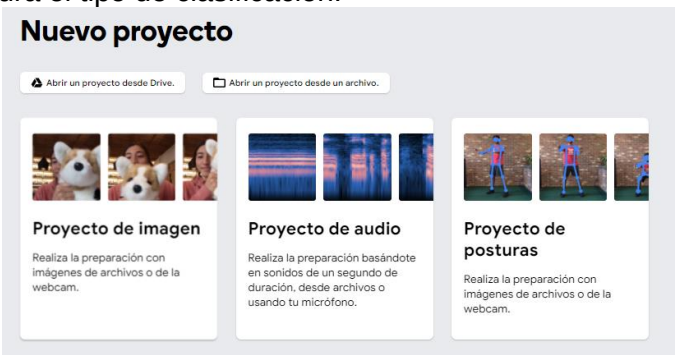


Figura 31: Selección del tipo de proyecto en Teachable Google machine

Para el entrenamiento es necesario tener una muestra de los tipos de datos a clasificar teniendo como base el semáforo en verde que muestra en la Figura 3-24.

Figura 32: Muestras de semáforo en verde

Para el entrenamiento del color amarillo es necesario tener una muestra de los tipos de datos a clasificar teniendo como base el semáforo en amarillo que muestra en la Figura 3-25.

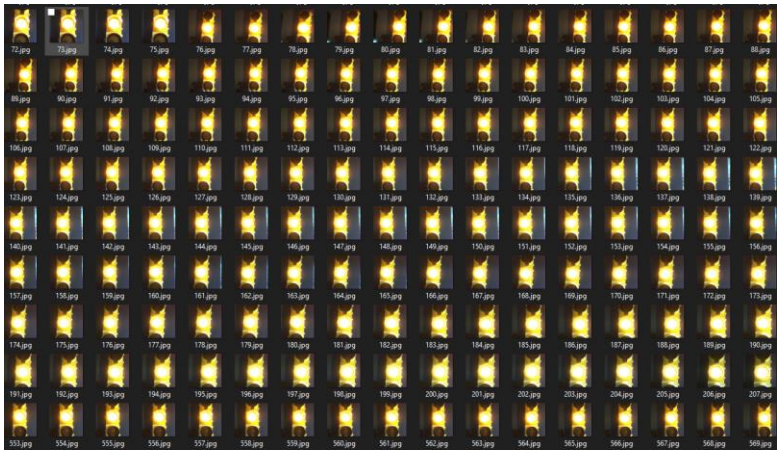


Figura 33: Muestras de semáforo en amarillo

Para el entrenamiento del color rojo es necesario tener una muestra de los tipos de datos a clasificar teniendo como base el semáforo en rojo que muestra en la Figura 3-26.

Figura 34: Muestras de semáforo en rojo

Para el caso donde no esté presente ningún color es necesario tener una muestra de los tipos de datos a clasificar teniendo como base el semáforo en estado de reposo o con las luces apagadas que muestra en la Figura 3-27.

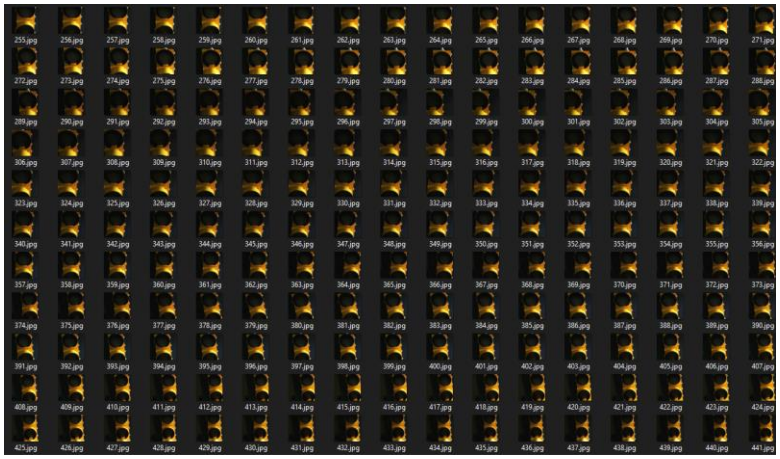


Figura 35: Muestras de semáforo apagado

En la página se procede a realizar la carga de las muestras y asignar los nombres de las clases de cada muestra como se muestra en la Figura 3-28.

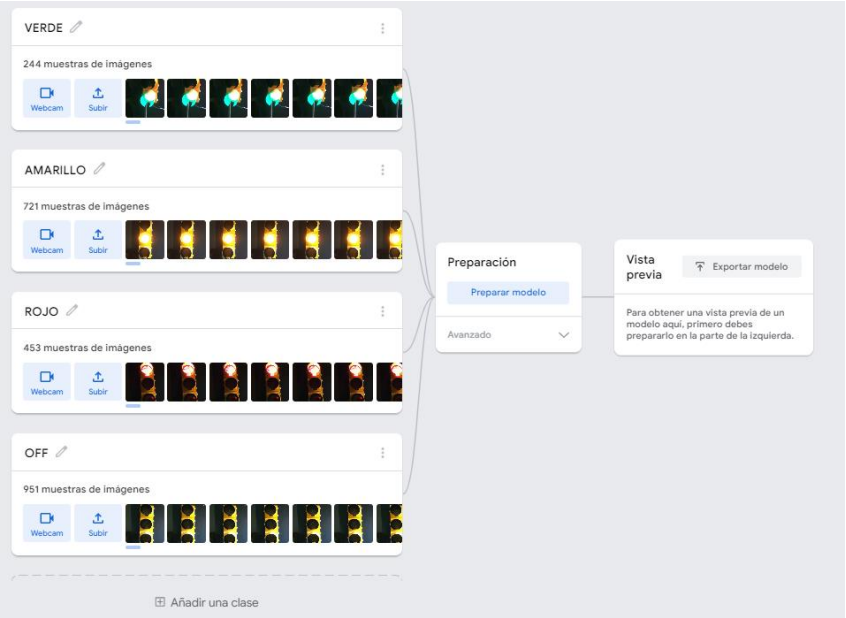


Figura 36: Creación de clases y carga de muestras en pagina

Posterior a la carga de las muestras de cada clase se procede a realizar la creación del modelo de acuerdo con los parámetros que se muestran en la Figura 3-29.

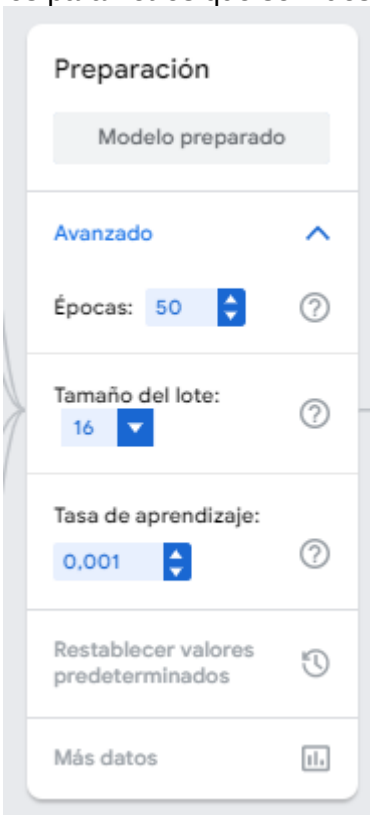


Figura 37: Parámetros para entrenamiento del modelo en Teachable

Para la obtener el modelo en un archivo en formato Keras es necesario seleccionar la opción de TensorFlow mostrada en la Figura 3-30.

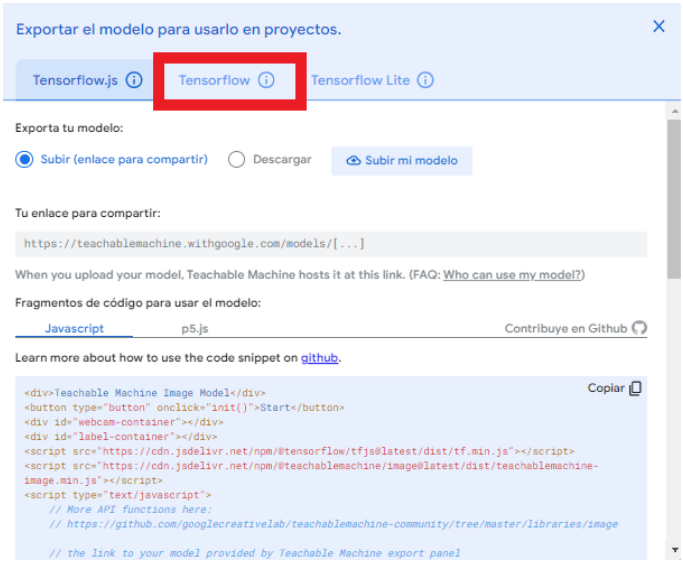


Figura 38: Parámetros para entrenamiento del modelo en Teachable

4.7.2. Instalación de TensorFlow

En la etapa de procesamiento para el manejo de los archivos en formato Keras es necesario tener instalado las librerías TensorFlow en la raspberry, proceso que se detalla a continuación.

En la instalación del TensorFlow es necesario ejecutar los comandos de la Figura 3-31 en el terminal de la raspberry.

```
$ sudo apt upgrade
```

Figura 39: Comando primario previo a instalación TensorFlow

En la Figura 3-32 se procede a ejecutar en consola el comando para la instalación de *libatlas-base-dev* que son instancias previas para poder instalar el TensorFlow.

```
$ sudo apt install libatlas-base-dev
```

Figura 40: *Comando para instalación libatlas-base-dev*

En la Figura 3-33 se procede a ejecutar en consola el comando para la instalación de todos los controladores necesarios para TensorFlow.

```
$ pip3 install --upgrade https://storage.googleapis.com/tensorflow/mac/cpu/tensorflow-1.9.0-py3-none-any.whl
```

Figura 41: Comando de instalación para controladores TensorFlow

4.7.3. Adquisición de predicción mediante TensorFlow

En el manejo de la librería Keras para el uso de TensorFlow es necesario el designar un objeto y cargar el archivo con extensión h5, realizar la carga de un archivo en formato JPG y posterior realizar la predicción del modelo establecido comparando con la imagen actual, teniendo como resultado el porcentaje de igualdad a la clase del modelo y su respectivo vector como se muestra en la Figura 3-34.

```python
import tensorflow.keras as keras

# Carga el modelo guardado previamente
model = keras.models.load_model("modelo_entrenado.h5")

# Realiza una predicción con una imagen de prueba
imagen = keras.preprocessing.image.load_img("imagen_prueba.jpg", target
imagen = keras.preprocessing.image.img_to_array(imagen)
imagen = imagen / 255.0
prediccion = model.predict(imagen[np.newaxis, ...])

# Imprime la clase con mayor probabilidad
clase_predicha = np.argmax(prediccion[0])
print("La clase predicha es: ", clase_predicha)
```

Figura 42: Código de ejemplo para el uso de Keras y TensorFlow

5. RESULTADOS

En este capítulo se detallan los datos posteriores al procesamiento y lo que se visualiza en la pantalla de la raspberry, al terminar la predicción se debe observar el color que reconoció del semáforo a manera que sean 4 opciones posibles que son el color rojo, amarillo, verde y el sin color o apagado como se demuestran muestra en la Figura 4-1.

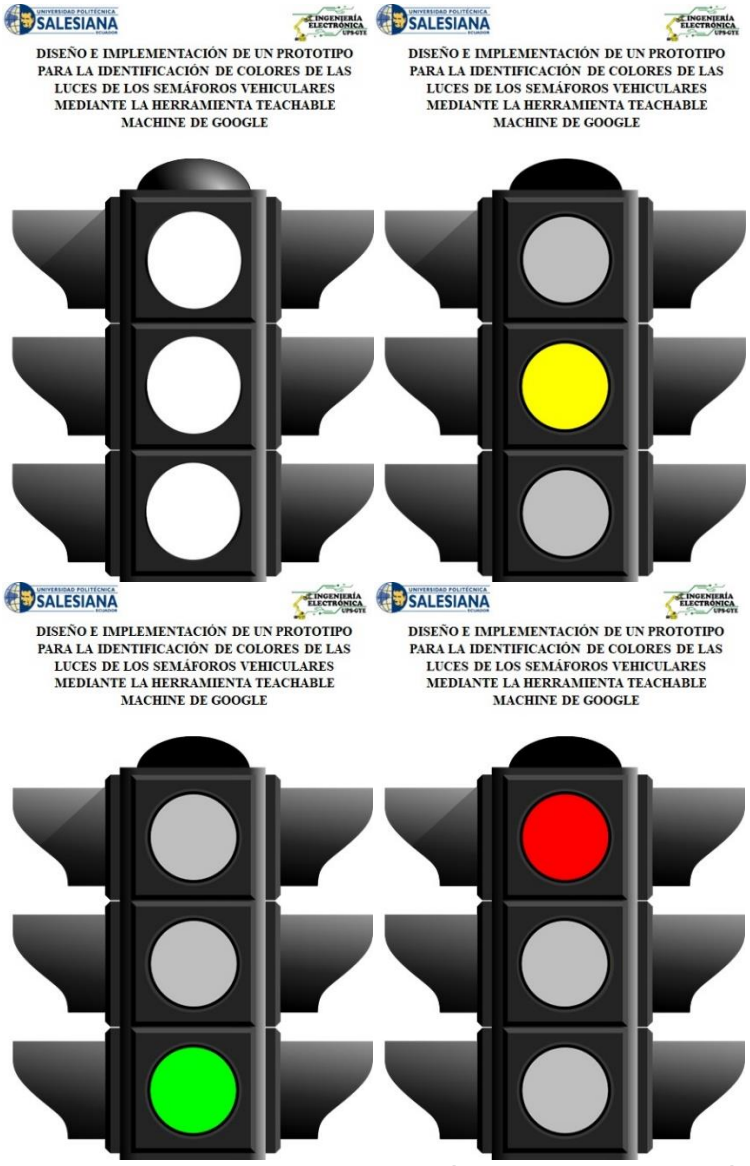


Figura 43: Resultado de acuerdo con la predicción de los colores según TensorFlow

5.1. Resultados en Google Teachable Machine

Para las pruebas se realizó una evaluación en el Teachable machine mediante el uso de una cámara USB conectada al computador que posterior al entrenamiento como se muestra en las Figuras 4-3, 4-4 y 4-5 para la comparación de los colores rojo, amarillo y verde.

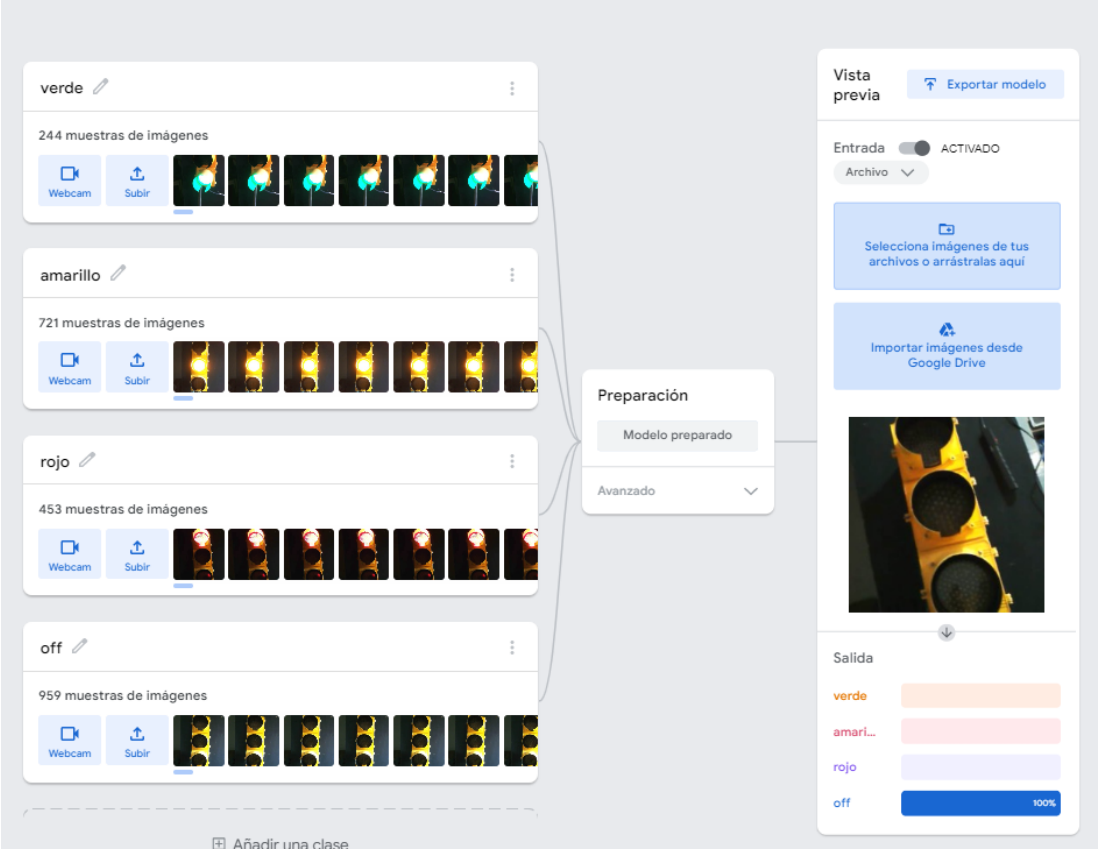


Figura 44: Comprobación de clases mediante Teachable machine del estado inactivo

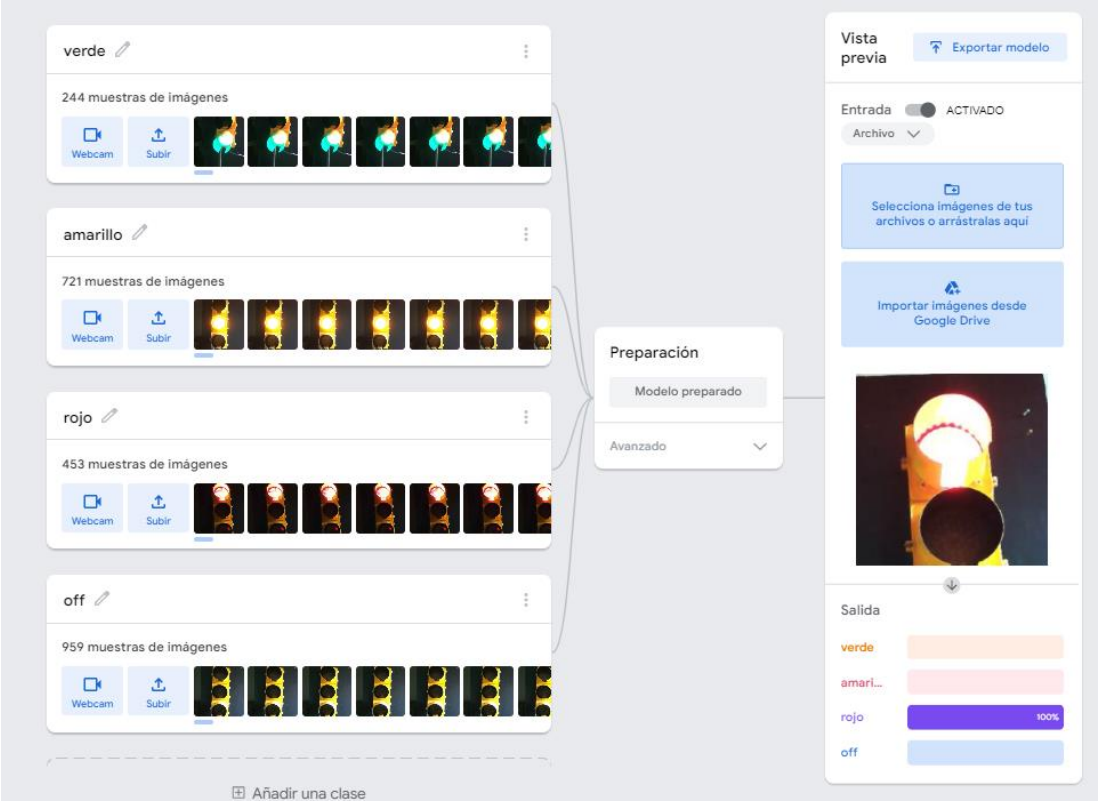


Figura 45: Comprobación de clases mediante Teachable machine del color rojo

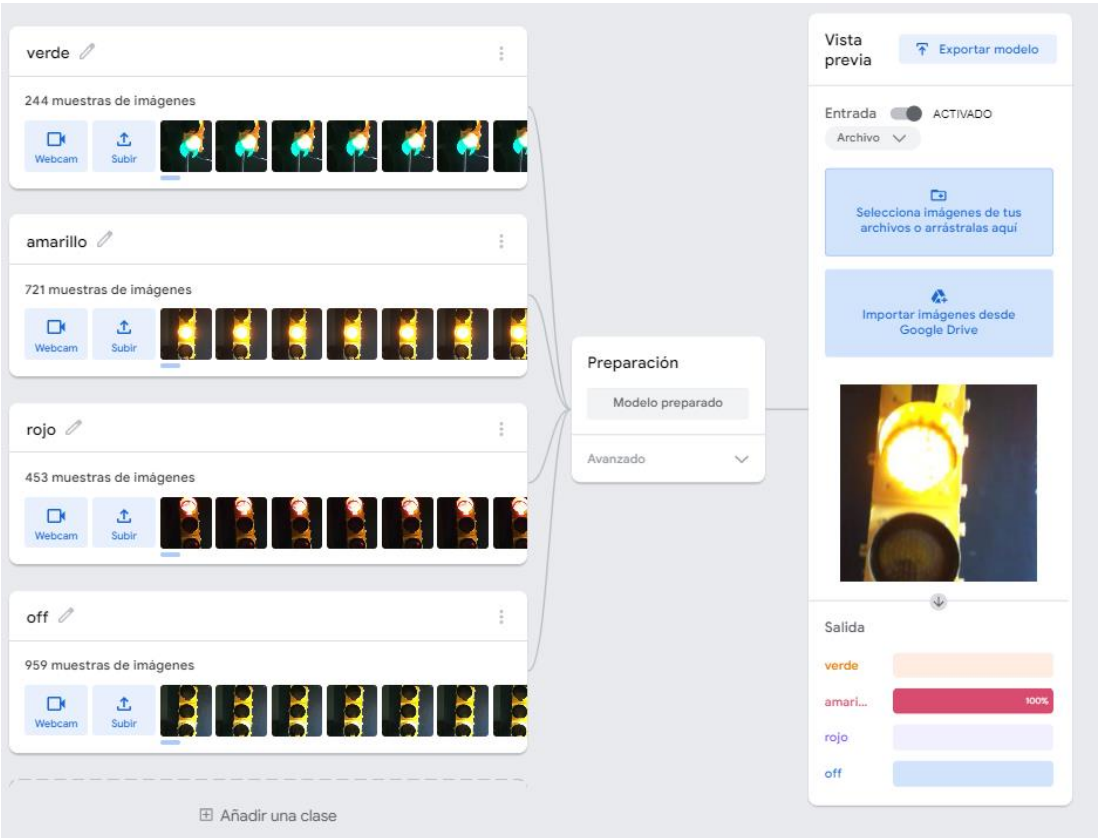


Figura 46: Comprobación de clases mediante Teachable machine del color amarillo

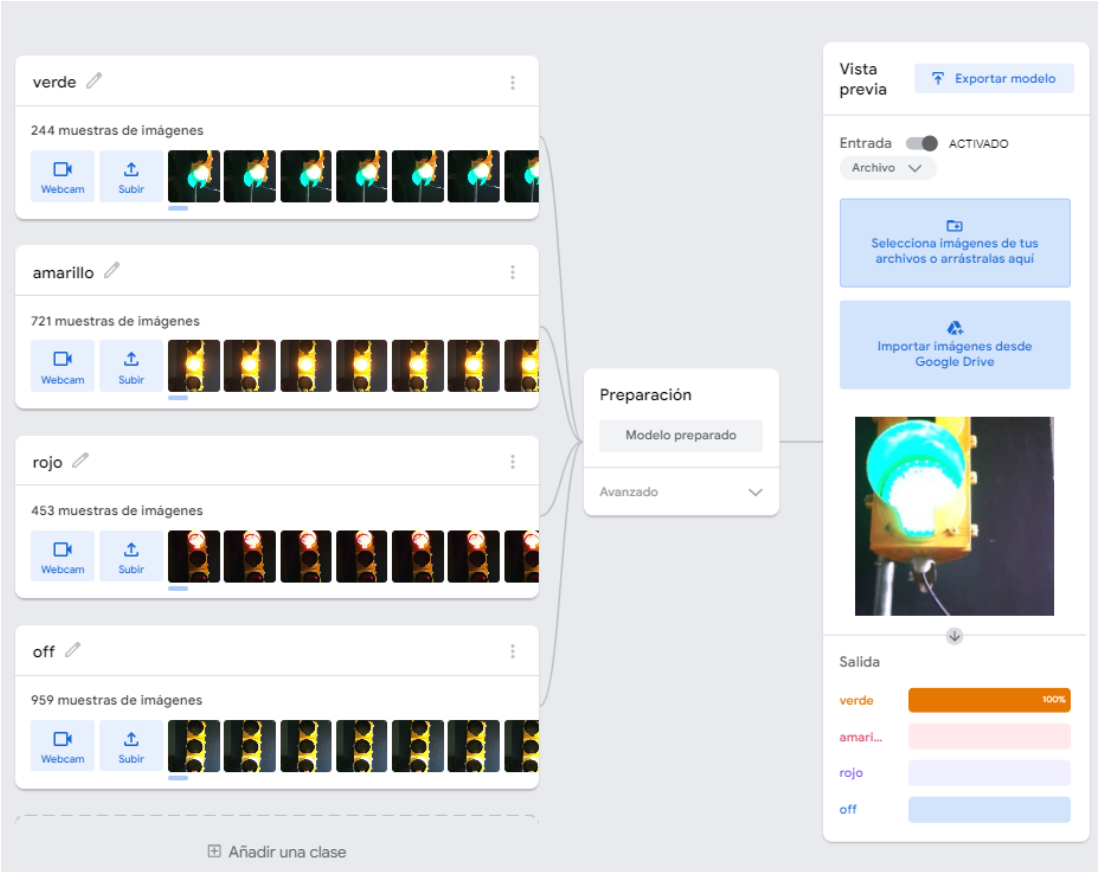


Figura 47: Comprobación de clases mediante Teachable machine del color verde

5.2. Resultados en Raspberry pi

5.2.1. Pruebas con luminosidad reducida

Se realizaron pruebas ejecutando en el ide de Thonny Python en la raspberry que se detalla en anexos donde al ejecutarse se muestran las detecciones para los colores una imagen en jpg y lo que capta la cámara como se detallan en la Figura 4-6, 4-7 y 4-8 con poca luminosidad.

Figura 48: Predicción mediante raspberry usando modelo Keras para color rojo con luminosidad reducida

Figura 49: Predicción mediante raspberry usando modelo Keras para color amarillo con luminosidad reducida

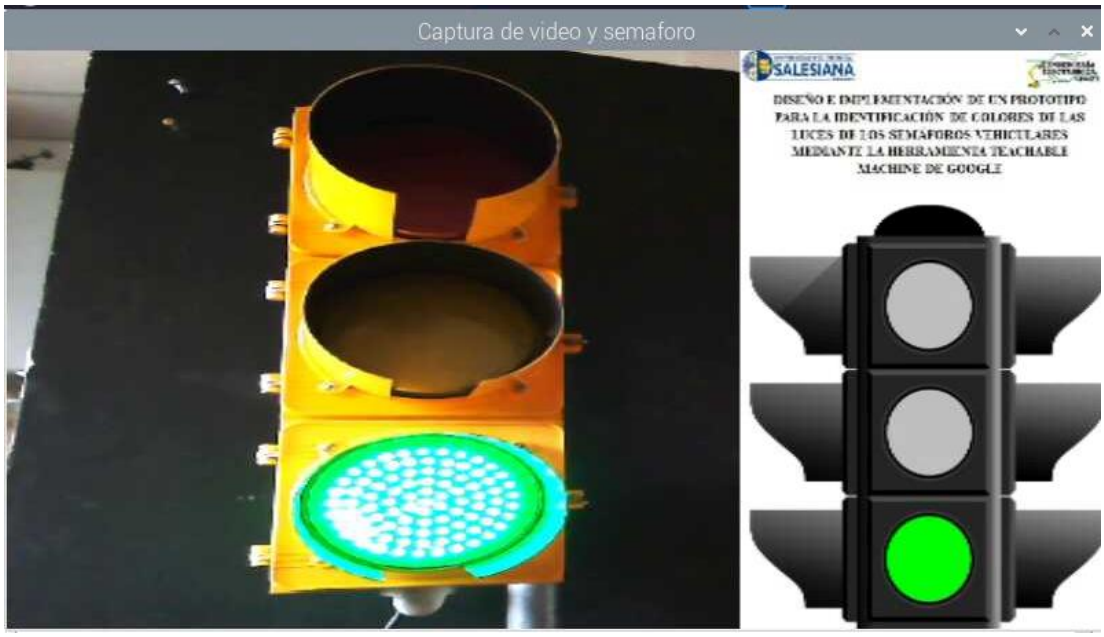


Figura 50: Predicción mediante raspberry usando modelo Keras para color verde con luminosidad reducida

5.2.2. Pruebas con luminosidad normal

Se realizaron pruebas ejecutando en el ide de Thonny Python en la raspberry que se detalla en anexos donde al ejecutarse se muestran las detecciones para los colores una imagen en jpg y lo que capta la cámara como se detallan en la Figura 4-9, 4-10 y 4-11 con luminosidad normal.

Figura 51: Predicción mediante raspberry usando modelo Keras para color rojo con luminosidad normal

Figura 52: Predicción mediante raspberry usando modelo Keras para color amarillo con luminosidad normal

Figura 53: Predicción mediante raspberry usando modelo Keras para color verde con luminosidad normal

CONCLUSIONES

- Para la creación del universo de muestras de los diferentes colores es necesario tomar una cantidad de muestras superior a 150 muestras y en diferentes ángulos para aumentar la precisión de las predicciones.
- En la creación del modelo Keras en Google Teachable machine es necesario tener clasificada previamente las imágenes para entrenamiento revisando la calidad de estas.
- Para el manejo de las librerías TensorFlow que dan acceso al uso de modelos Keras es necesario realizar la correcta instalación de librerías y complementos para trabajar en la raspberry pi.
- En las pruebas de luminosidad teniendo poca o alta es necesario realizar el entrenamiento de las condiciones de cada luz para un óptimo desarrollo del procesamiento.
- El sistema de la raspberry deberá ser raspbian con 64 bits ya que las librerías trabajan solo en esta arquitectura.

RECOMENDACIONES

- Realizar las muestras de los universos y recortar las zonas de interés para la clasificación.
- En el desarrollo del modelo Teachable machine es necesario tener una meta de predicción al modelo menor a 0.0001 para una óptima clasificación.
- En el desarrollo del programa es necesario tener en cuenta las directivas a usar para el modelo Keras en TensorFlow.
- Es necesario que el entrenamiento de acuerdo a los entornos donde tenga poca o alta luminosidad se tomen las muestras de cada color para el entrenamiento.

REFERENCIAS BIBLIOGRAFICAS

A. I. Alsarawi, A. A.-M.-F. (2019). *An IoT-based Health Monitoring System Using Raspberry Pi and Wireless Body Area Network.* nternational Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, Malaysia, 2019, pp. 1-5.

Agencia Nacional de Tránsito. (2022). *Agencia Nacional de Tránsito*. Recuperado el 21 de diciembre de 2022, de https://www.ant.gob.ec/estadisticas-siniestros-de-transito-prueba/

Aggarwal, K. J. (2015). *Serial Peripheral Interface (SPI) Bus: Protocol and Its Applications.* Journal of Emerging Technologies in Computing and Information Sciences, vol. 5, no. 2.

Agüero, M. B. (2017). Obtenido de Sistema de adquisición de datos: https://materias.df.uba.ar/mta2019c1/files/2014/08/Adquisicion-de-datos-Aguero.pdf

Ahuja., J. (2022). *Google Teachable Machine: A Comprehensive Guide to Introducing ML Concepts in the Easiest Wa.*

Alpaydin, E. (2011). *Introduction to Machine Learning.* MIT Press.

Altaruru. (2021). *www.altaruru.com*. Obtenido de https://www.altaruru.com/raspberry-pi-como-instalar-una-pantalla-lcd-de-3-5/: https://www.altaruru.com/raspberry-pi-como-instalar-una-pantalla-lcd-de-3-5/

Alur, A. G. (2013). *Embedded Systems Design: A Unified Hardware/Software Introduction.* Morgan Kaufmann Publishers.

ArduCAM. (2022). *ArduCAM.* Obtenido de https://docs.arducam.com/Raspberry-Pi-Camera/Pivariety-Camera/Quick-Start-Guide/

Arrow . (2015). Obtenido de DAS o DAQ: https://www.arrow.com/es-mx/categories/data-acquisition/data-converters/data-acquisition-systems

Astrom, K. J. (1995). *PID Controllers: Theory, Design and Tuning.* ISA; Edición: Second Edition.

Åström, K. J. (1995). *PID Controllers: Theory, Design, and Tuning. Second Edition. Research Triangle Park.* Tore: Instrument Society of America.

Báez, P. G. (2016). Introducción a las Redes Neuronales y su aplicación a la Investigación Astrofísica. *Investigación Astrofísica*.

Bednarek, M. (2012). *Serial Peripheral Interface (SPI) - Communication Interface of Microcontrollers.* Advanced Microcontroller Systems, InTech.

Benavides, K. R. (2017). Locomoción de Robot Móvil.

Branque Rosales, N. B. (2022). *Implementación de un sistema automatizado, mediante el uso de visión artificial para la clasificación del maracuyá, según su color de madurez y el uso de un sistema SCADA para el monitoreo de la productividad.* Bachelor's thesis, La Libertad: Universidad Estatal Península de Santa Elena, 2022).

Cavanagh., J. K. (2015). *Raspberry Pi: An educational platform for teaching computer science.* Computer Science Education, 25(3), pp. 169-181.

Chen, J. (2013). *Serial Peripheral Interface (SPI) in Embedded Systems.* International Journal of Computer Applications, vol. 78, no. 11.

Chilán Carrasco, M. F. (2020). *onstrucción y entrenamiento de Redes Neuronales Artificiales en la Biblioteca de Tensorflow para el reconocimiento de caracteres numéricos.* Universidad de Guayaquil. Facultad de Ingeniería Industrial.

Digilent. (2015). *Pmod GYRO: giroscopio digital de 3 ejes*. Obtenido de https://store.digilentinc.com/pmod-gyro-3-axis-digital-gyroscope/

Digilent. (2017). *Pmod BLE: interfaz Bluetooth de baja energía*. Obtenido de Digilent: https://store.digilentinc.com/pmod-ble-bluetooth-low-energy-interface/

Digilent. (2020). *Digilent*. Obtenido de Pmod ALS: Ambient Light Sensor: https://store.digilentinc.com/pmod-als-ambient-light-sensor/

Elvins, A. (2009). *Un análisis de la detección de Patrones de Color RGB (CRGPD).* Estudio de Ingeniería de la Universidad de Florida, Gainesville.

Foundation, R. P. (2015). *Raspberry Pi 2 Model B.* Obtenido de Raspberry Pi 2 Model B.: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

Foundation, R. P. (2019). *Raspberry Pi 4 Model B.* Obtenido de Raspberry Pi 4 Model B: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

Foundation, R. P. (9 de febrero de 2023). *Raspberry Pi Foundation. .* Obtenido de https://www.raspberrypi.org/

Foundation., R. P. (2012). *Raspberry Pi Model B.* Obtenido de Raspberry Pi Model B.: https://www.raspberrypi.org/products/model-b/

Foundation., R. P. (2016). *Raspberry Pi 3 Model B. .* Obtenido de Raspberry Pi 3 Model B. : https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Furber, S. (2010). *Embedded Systems: Architecture, Programming, and Design.* CRC Press.

García Chang, M. E. (2019). *Diseño e implementación de una herramienta de detección facial.*

Gill, S. S. (2011). *Embedded Systems: An Introduction.* Journal of Emerging Technologies in Computing and Information Sciences, vol. 1, no. 1.

H. Rahman, N. I. (2019). *Development of a smart irrigation system using Raspberry Pi.* 2019 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT), Dhaka, Bangladesh, 2019, pp. 1-6, doi: 10.1109/ICEEICT.2019.8765224.

Hernández, A. I. (2010). REDES NEURONALES MULTIMODELO APLICADAS AL CONTROL DE SISTEMAS. *Sabadell*, 79. Universidad Autonoma de Barcelona.

hetpro-store. (2017). Obtenido de https://hetpro-store.com/TUTORIALES/wp-content/uploads/2016/05/lm35-img.png?x18372

INEC. (2021). *Ecuador en cifras.* Recuperado el 21 de diciembre de 2022, de https://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_Economicas/Estadistica%20de%20Transporte/2021/2021_SINIESTROS_PPT.pdf

Jain, K. (2011). *Embedded Systems: A Contemporary Design Tool.* International Journal of Computer Applications, vol. 38, no. 9.

Jimenez Rubalcaba, Y. (2016). *BIBLIOTECA PARA EL PRE-PROCESAMIENTO PARALELO DEL MODELO BDAM USANDO UN CLUSTER DE COMPUTADORAS .* Bachelor's thesis, Universidad de las Ciencias Informáticas. Facultad 6.

Jun, J., Lee, J., & Baik, J. (2016). *Color image segmentation based on HSL and HSV color spaces.* Expert Syst. Appl. 30 (3), 573–586. .

Kang. (2000). MOSFET devices with polysilicon on single-layer. (págs. 35-38). IEEE Technical Digest. IEDM.

Kaur, A. S. (2019). *Implementation of Raspberry Pi Based Electronic Voting System.* International Conference on Computer, Communication and Computing Technologies (ICCCCT), Mohali, India, 2019, pp. 1-5.

Kaushik, S. S. (2018). *IoT Based Home Automation System Using Raspberry Pi.* Dehradu: International Conference on Emerging Technologies and Innovations in Engineering (ICETIE), pp. 1-5.

Lai, X. T. (2020). *Una arquitectura de alto rendimiento de detección de objetos basada en cámaras de Raspberry Pi.* IEEE Access, vol. 8, pp. 77463-77473.

Lam, L. H. (2015). *La aplicación de patrones de color CMYK para la identificación de imágenes.* IEEE Transactions on Circuits and Systems for Video Technology, 25(5), 866-875.

Liu, Y. (2017). *An Overview of Embedded Systems.* Journal of Computer Science, vol. 3, no. 5.

M. Awan, A. A. (2018). *Performance Analysis of Raspberry Pi 3 for IoT based Home Automation System.* IEEE International Conference on Computing, Communication and Automation (ICCCA), Noida, 2018, pp. 1-5.

Machine, M. L. (2022). *Pratik Shukla.*

Manuel Baltieri, C. L. (Marzo de 2018). *http://dx.doi.org/10.1101/284562.* Obtenido de University of Sussex: https://www.biorxiv.org/content/biorxiv/early/2018/03/19/284562.full.pdf

Moreno Armendariz, A. S. (2001). Sistema de Información Científica. *Red de Revistas Científicas de América Latina, el Caribe, España y Portugal*, 13-32.

Negnevitsky, M. (2016). *Artificial Intelligence: A Guide to Intelligent Systems.* Pearson Education.

Ni. (2019). *USB-6009*. Obtenido de https://www.ni.com/es-cr/support/model.usb-6009.html

Norvig, S. R. (2010). *Artificial Intelligence: A Modern Approach.* Pearson Education.

omes-va. (2019). *omes-va.COM.* Obtenido de omes-va.COM: https://omes-va.com/como-instalar-opencv-en-raspberry-pi/

Ortiz, G. B. (2019). *APLICACION DE DEEP LEARNING USANDO TENSORFLOW PARA ANALISIS DE LA CALIDAD DE SOFTWARE DESARROLLADO EN IBM RPG.*

Patel, M. R. (2019). *Serial Peripheral Interface (SPI) Bus in Embedded Systems Design.* Cengage Learning.

Paul, R. (2008). Temperature effects. *Neutron Physics*. Francia: EDP Sciences.

Pearl, J. (2017). *Causality: Models, Reasoning, and Inference.* Cambridge University Press,.

pIHUT. (2021). *pIHUT.* Obtenido de pIHUT: https://thepihut.com/products/wide-angle-1080p-uvc-compliant-usb-camera-module-with-metal-case

Pinto, E. (2023).

Pugh, N., Finlayson, G., & Drew, M. (2015). *A survey of approaches to color image contrast enhancement.* Color Res. Appl. 30 (1), 1–20. .

Pulido, M. Á. (2000). *Convertidores de frecuencia, controladores de motores y SSR.* Marcombo.

Qazi., M. A. (2016). *Raspberry Pi: A Comprehensive Study.* Journal of Emerging Technologies in Web Intelligence, 8(2), pp. 93-101.

Sánchez, L. F. (2016). Control cinemático y dinámico. *Laboratorio de Robótica Móvil y Sistemas Automatizados*.

Serrano Arenas, J. S. (2020). *Prototipo de aplicación móvil para la identificación de mazorcas de cacao enfermas haciendo uso de visión por computadora y aprendizaje de máquina.*

Sevcik, S. H. (2020). *Serial Peripheral Interface (SPI) in Fundamentals of Embedded Systems.* Prentice Hall.

Sparkfun. (2018). *serial-peripheral-interface-spi*. Obtenido de Sparkfun: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all

Teslabem. (2017). *Teslabem*. Obtenido de Fundamentos I2C – Aprende.: https://teslabem.com/nivel-intermedio/fundamentos/

Texas Instruments. (2017). *LM35 Precision Centigrade Temperature Sensors.* Obtenido de Texas Instruments: https://www.ti.com/lit/ds/symlink/lm35.pdf

Tyagi, A. M. (2018). *IoT based Wireless Home Automation System Using Raspberry Pi.* Kolkata: 3rd International Conference on Computers and Devices for Communication (CODEC).

Upton, E. (2012). *The Raspberry Pi: An Introduction.* IEEE Micro, 32(2), pp. 56-60.

Veloso, C. (2016). *FUNCIONAMIENTO DE UN SENSOR DE TEMPERATURA.* Obtenido de electrontools: https://www.electrontools.com/Home/WP/funcionamiento-de-un-sensor-de-temperatura/

Viera Maza, G. I. (2017). *Procesamiento de imágenes usando OpenCV aplicado en Raspberry Pi para la clasificación del cacao.*

Vizcaíno, J. R. (2011). LabView. En *Entorno gráfico de programación* (págs. 3-10). Marcombo.

W. R. Stevens, A. R. (2014). *Advanced Programming in the UNIX Environment (3rd ed.).* Addison-Wesley Professional.

Wang, X. y. (2018). *Detección de objetos usando Raspberry Pi y OpenCV.* EEE 11th Global Conference on Consumer Electronics (GCCE).

Xu, S. e. (2018). *Reconocimiento de números de estacionamiento basado en Raspberry Pi y cámaras de visión.* IEEE International Conference on Industrial Technology (ICIT), 2018, pp. 1-5.

Y. LeCun, Y. B. (2015). *Deep learning.* Nature, vol. 521, no. 7553, pp. 436-444.

Zator. (2016). *Tecnología del PC.* Obtenido de Tecnología del PC: https://www.zator.com/Hardware/H2_5_1_1.htm

Zhang, Y. L. (2018). *Color-pattern recognition based on fuzzy neural network and its applications to image segmentation.* IEEE Access, 6, 8169-8179.

ANEXOS

Código opencv

```
import cv2

# Inicia la captura de video desde la cámara predeterminada (puede ser 0 o 1
dependiendo de tu configuración)
captura = cv2.VideoCapture(0)

# Configura el tamaño de la ventana de vídeo
captura.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
captura.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

while True:
    # Lee el siguiente frame
    ret, frame = captura.read()

    # Muestra el frame en una ventana
    cv2.imshow("Captura de video", frame)

    # Si se presiona la tecla 'q', detiene la captura
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Libera los recursos de la cámara
captura.release()

# Cierra la ventana de vídeo
cv2.destroyAllWindows()
```

Código keras

```
import tensorflow.keras as keras

# Carga el modelo guardado previamente
model = keras.models.load_model("modelo_entrenado.h5")

# Realiza una predicción con una imagen de prueba
imagen = keras.preprocessing.image.load_img("imagen_prueba.jpg",
target_size=(224, 224))
imagen = keras.preprocessing.image.img_to_array(imagen)
imagen = imagen / 255.0
prediccion = model.predict(imagen[np.newaxis, ...])

# Imprime la clase con mayor probabilidad
clase_predicha = np.argmax(prediccion[0])
print("La clase predicha es: ", clase_predicha)
```

Código final

```
import cv2
import numpy as np
import pygame
from keras.models import load_model
import time

# Inicializar pygame mixer
pygame.init()

# Cargar los sonidos
sonido_rojo = pygame.mixer.Sound("sonido_rojo.mp3")
sonido_verde = pygame.mixer.Sound("sonido_verde.mp3")
sonido_amarillo = pygame.mixer.Sound("sonido_amarillo.mp3")

# Carga el modelo entrenado en Teachable Machine
model = load_model("modelo_entrenado.h5")

# Inicia la captura de video desde la cámara predeterminada (puede ser 0 o 1
dependiendo de tu configuración)
captura = cv2.VideoCapture(0)

# Configura el tamaño de la ventana de vídeo
captura.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
captura.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

while True:
    # Lee el siguiente frame
    ret, frame = captura.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.resize(gray, (224, 224))
    gray = np.expand_dims(gray, axis=-1)
    gray = np.expand_dims(gray, axis=0)

    # Realizar la predicción
    prediccion = model.predict(gray)
    clase_predicha = np.argmax(prediccion[0])
    probabilidad = prediccion[0][clase_predicha]*100

    # Seleccionar color solo cuando la probabilidad sea mayor al 80%

    if probabilidad > 80:
        if clase_predicha == 0:
            color = (0,0,255) # Rojo
            text = "Rojo"
            sonido_rojo.play()
            semaforo = cv2.imread("semaforo_rojo.jpg")
        elif clase_predicha == 1:
            color = (0,255,0) # Verde
            text = "Verde"
             sonido_verde.play()
            semaforo = cv2.imread("semaforo_verde.jpg")
        else:
```

```
            color = (0,255,255) # Amarillo
            text = "Amarillo"
            sonido_amarillo.play()
            semaforo = cv2.imread("semaforo_amarillo.jpg")
    else:
        color = (0,0,0)
        text = "No se detecto"
        semaforo = cv2.imread("semaforo_off.jpg")

    # Dibujar un rectangulo alrededor del objeto
    cv2.rectangle(frame, (0, 0), (50, 50), color, 2)
    cv2.putText(frame, text, (10, 25), cv2.FONT_HERSHEY_SIMPLEX, 0.7, color, 2)

    # Redimensiona la imagen de semaforo para que tenga el mismo alto que el
frame
    semaforo = cv2.resize(semaforo, (frame.shape[1] // 2, frame.shape[0]))

    # Concatenar la imagen del semaforo y el frame
    result = cv2.hconcat([frame, semaforo])

    # Muestra el resultado en una ventana
    cv2.imshow("Captura de video y semaforo", result)

    # Espera 2 segundos antes de continuar con la siguiente iteración
    time.sleep(2)

    # Si se presiona 'q', se detiene el bucle
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Detener todos los sonidos
pygame.mixer.stop()

# Libera los recursos de la cámara
captura.release()

# Cierra la ventana de vídeo
cv2.destroyAllWindows()

# Finalizar pygame mixer
pygame.quit()
```