UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio Politécnico

Prototipo De Sistema Embebido Para La Detección De Personas En Tiempo Real Para El Control De Espacios Mediante Técnicas De Aprendizaje Profundo

Germán Darío Picón Lescano

Ingeniería Electrónica

Trabajo de fin de carrera presentado como requisito
para la obtención del título de Ingeniero
Electrónico

Quito, 21 de mayo de 2021

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio Politécnico

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Prototipo De Sistema Embebido Para La Detección De Personas En Tiempo Real Para El Control De Espacios Mediante Técnicas De Aprendizaje Profundo

Germán Darío Picón Lescano

Nombre del profesor, Título académico	Diego Benítez, PhD. Ingeniería Eléctrica

Quito, 21 de mayo de 2021

© DERECHOS DE AUTOR

Nombres y apellidos: Germán Darío Picón Lescano

Código: 00136486

Cédula de identidad: 1717431801

Lugar y fecha: Quito, 21 de mayo de 2021

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

Este Proyecto se centra en el desarrollo de un detector de personas, con las siguientes características, que funcione en tiempo real, sea de bajo costo y pueda ser de software abierto. El detector se realizará con el entrenamiento de una red neuronal convolucional conocida como Tiny-YOLO y se lo implementará a un sistema embebido conformado por una Raspberry PI 3, un acelerador de Intel y un circuito de control formado por un relé, un transistor y los GPIO del Raspberry. Durante el proyecto se explican varios temas en especial como crear y entrenar esta red neuronal utilizando un set de datos existentes, así mismo el detector se calibró variando dos parámetros necesarios para obtener el nivel de confianza deseado al detectar una persona.

Palabras clave: Redes Neuronales Profundas, Stick de Cómputo Neuronal, OpenVino, Python, Raspberry Pi, Tiny-Yolo, IoU.

ABSTRACT

This Project focuses on the development of a people detector, with the following characteristics, that works in real time, is low cost and can be open software. The detector will execute with the training of a convolutional neural network known as Tiny-YOLO and it will be implemented in an embedded system made up of a Raspberry PI 3, an Intel accelerator and a control circuit made up of a relay, a transistor and the GPIOs. of the Raspberry. During the project several topics are explained, especially how to create and train this neural network using an existing data set, likewise the detector was calibrated by varying two parameters necessary to obtain the desired level of confidence when detecting a person.

Keywords: Deep Neural Networks Neural, Compute Stick, OpenVino, Python, Raspberry Pi, Tiny-Yolo, IoU.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El mundo se encuentra atravesando una revolución tecnológica total, el internet de las cosas, la realidad aumentada o la inteligencia artificial cada día van causando más revuelo en las noticias. La rutina y el diario vivir se tornan cada día más complejos, haciendo que busquemos formas de simplificarnos la vida, de solucionar nuestros problemas de la manera mas sencilla posible. Es aquí donde se ven que cada vez asoman más aplicaciones, más productos, más componentes electrónicos, diseñados por nosotros mismos para solucionar estas dificultades (Zapata & Rivera, 2016). Con este contexto aparecen los sistemas embebidos, cuyo uso es cada vez más común, debido a la evolución de la electrónica a lo largo del tiempo y esto ha permitido obtener dispositivos más pequeños, más baratos y mucho mejor equipados para solucionar una vida tan atareada.

No se podría hablar de solución de problemas si el sistema que se está implementando no podría aprender sobre la marcha, es aquí donde el aprendizaje profundo de redes neuronales ha demostrado ser la mejor técnica a la hora de enfrentarse a problemas muy complejos. En (Francisco Blanco Esquivel, 2020) se habla de cómo el principal desafío a la hora de implementar estas redes neuronales, es hacerlo en sistemas que cumplan en términos de rendimiento, sean resistentes a fallos y eficientes en consumo de potencia; este desafío aumenta a la hora de utilizar sistemas tan limitados como lo pueden ser los embebidos. Como vemos en (Alpaca Rendón, 2017), el Raspberry PI es un pequeño procesador capaz de trabajar con avanzados algoritmos de reconocimiento, almacenar información en bases de datos y conectarse a dispositivos externos de una fácil manera.

Los algoritmos basados en Deep Neural Networks (DNN), en especial los que se especializan en identificación y clasificación de objetos, consumen una gran cantidad de recursos y se ejecutan en grandes servidores (Tapia, 2020), pero entonces ¿Cómo un sistema tan pequeño, con varias limitaciones puede ejecutar semejantes programas? Pues por medio de la incorporación de una herramienta de desarrollo para inferencia de aprendizaje profundo, en este caso el neural compute stick de Intel (NCS), que disminuye en gran manera la carga

de recursos en los sistemas embebidos, permitiéndonos correr dichos algoritmos de manera precisa y veloz.

Se ha hablado de DNN y de su utilidad a la hora de resolver problemas complejos, por lo que fue necesario trabajar con un detector de objetos en tiempo real para administrar de manera correcta la energía en un lugar predeterminado con un sistema de control. Como sabemos en (Uijlings et al., 2018) la detección de objetos es uno de los problemas mas desafiantes en el campo de la visión por computadora. Aquí el objetivo es localizar, reconocer y contabilizar diferentes objetos en una escena y asignarles etiquetas, donde el problema reside en reutilizar clasificadores entrenados existentes para poder etiquetar los objetos en diferentes lugares. Por lo estudiado en (Huang et al., 2019), hay métodos como el You-Only-Look-Once (YOLO) o el Regional-based Convolutional Neural Networks (R-CNN), que logran un modelo eficiente y preciso con una alta precisión media (mAP), pero que sin embargo los cuadros por segundo (FPS) que se procesan en las computadoras no son aptos para su aplicación en tiempo real; por lo que se propuso un modelo reducido de YOLO, el tiny-YOLO.

El modelo tiny-YOLO, es una variante de la arquitectura YOLO, que es en promedio 442% mas veloz que su contraparte; y su frecuencia de actualización en una sola unidad de procesamiento grafico (GPU) alcanza los 244 FPS (Rosebrock, 2020). A su vez tiene las ventajas de ser un modelo pequeño, menor a 50 MB, y una velocidad de inferencia rápida, por lo que lo hace muy adecuado para ser implementado en una Raspberry Pi que junto al NCS nos permitirá realizar este prototipo de sistema embebido en tiempo real. Gracias a lo aprendido en (Tapia, 2020), una herramienta vital para poder trabajar con el NCS es utilizar el set de OpenVino que provee de varias librerías que optimizan la ejecución e inferencia de redes neuronales, este set de librerías soporta por defecto lenguajes como C++ y Python, siendo este último el elegido para este proyecto dado su facilidad de uso, con sintaxis fácil de entender y que es de carácter de código abierto, además de ser un lenguaje multiplataforma que es el recomendado de utilizar en Raspbian, el sistema operativo de Raspberry PI.

El valor principal de este proyecto es diseñar un dispositivo de bajo costo, dado que se utilizan dispositivos de fácil adquisición, que sea reproducible por cualquier usuario y también se pueda modificar a futuro para otras aplicaciones, por eso la utilización de un lenguaje sencillo como Python y no C o C++, y que sea de poco consumo energético, por eso que funcione con una alimentación de 110 V que es el voltaje estándar utilizado en la mayor parte del continente americano. Y por último que detecte personas en tiempo real, de ahí que se haya utilizado una arquitectura reducida y eficiente como lo es el Tiny-YOLO.

METODOLOGIA

Lo que se logro en este proyecto fue crear un prototipo de sistema embebido utilizando una Raspberry PI 3, donde se incorporó un modelo pre entrenado de detección de objetos, optimizado por el procesador de Intel. Todo esto con el objetivo de que se realice la detección de personas en tiempo real de un espacio determinado, el nivel de confianza en la detección será evaluado y con este se implementara un circuito sencillo de control utilizando un relé que se activa con el GPIO 14 de la Raspberry PI y está conectado a una carga a 110 Voltios para simular un aparato electrónico ubicado en dicho espacio delimitado.

Raspberry PI

El Raspberry PI 3, es la tercera generación de este ordenador del tamaño de una tarjeta de crédito. Consiste en una placa base que soporta distintos componentes como un procesador ARM de hasta 1500 MHz, un chip grafico y una memoria RAM de hasta 8GB. Por medio de sus puertos y entradas, podemos conectar dispositivos periféricos como monitores, teclados, entre otros. La conexión a la red puede hacerse desde su puerto de Ethernet o a su vez por medio de WIFI utilizando una IP fija del router. Contiene un procesador Grafico VideoCoreIV, que permite la reproducción de video en alta definición y también consta de una ranura SD que nos permite instalar su sistema operativo, para este proyecto se instaló el Raspbian GNU/Linux versión 10.

Algunas consideraciones importantes que se deben tener en cuenta a la hora de trabajar con una Raspberry PI son que solo se debe conectar a una fuente de alimentación externa de 5 VCC y una corriente mínima de 2.5 A o a su vez utilizar un adaptador de potencia que reciba 110 a 240V y entregue el voltaje requerido. También este debe ser operado en un entorno con buena ventilación y sobre una superficie estable, plana y no conductiva. Y por último evitar manipular la placa del circuito minimizando así el riesgo de tener fallos mecánicos o eléctricos (Raspberry Pi Organization, n.d.).

La principal razón por la cual se utilizó una Raspberry PI 3, es por su precio tan

cómodo en el mercado. Otros procesadores para el desarrollo de sistemas embebidos con inteligencia artificial como el Google Coral o el NVIDIA Jetson Nano, tienen precios que oscilan entre los 150 y los 200 dólares incluyendo el kit completo, mientras que el Raspberry pi 3 cuesta entre 35 dólares, la placa base principal, hasta los 70 dólares, incluyendo adaptadores, tarjeta SD, cables HDMI, etc. En cuanto a rendimiento comparándolo con el NVIDIA Jetson Nano, ambos son muy similares, dado que poseen una memoria RAM similar de hasta 4-GB LPDDR4 cada uno, y su procesador en el Raspberry PI se ejecuta en una CPU ARM Cortex-A72 de 64 bits de cuatro núcleos a 1,5 GHz y el Jetson Nano se ejecuta en un ARM Cortex-A57 de cuatro núcleos de 64 bits a 1,42 GHz (All3DP, 2021), muy similares entre sí. Por lo que para poder tener un dispositivo a bajo costo y funcional se opto por la Raspberry PI en su versión 3.

Camera module V2

Como se esta realizando un dispositivo que detecte en tiempo real las personas en cualquier sito elegido, requerimos de una cámara que pueda realizar esta tarea y que a su vez cumpla con la alta exigencia de este. La Camera Module V2 (CMV2) es la indicada, ya que tiene un sensor Sony IMX219 de 8 megapíxeles (la cámara original del Raspberry PI es solo de 5 megapíxeles) que se utiliza tanto para realizar videos de alta resolución como para fotografías fijas. Es muy fácil de usar para gente principiante y a su vez ofrece varias cosas a usuarios mas experimentados (videos de lapso, cámara lenta, etc.). Comparado con la cámara original del kit, la CMV2 ofrece mejoras en calidad de imagen, fidelidad de color y detección con poca luz. se conecta mediante un cable plano de 15 cm al puerto CSI de la Raspberry Pi y funciona con todos los modelos de Raspberry Pi 1, 2, 3 y 4 (Raspberry, 2016). Existen varias bibliotecas creadas para su uso como la Picamera Python utilizada en este proyecto. Se puede adquirir esta excelente cámara por tan solo 23 dólares.

OpenVino y Neural Compute Stick

Como se explico previamente OpenVino es un set de herramientas de Intel que facilita la ejecución de redes neuronales, con él podemos desarrollar soluciones que emulan la visión humana. Soporta por defecto Python y C++ y cuenta con varios usos como el procesamiento

de múltiples flujos de video y el procesamiento asíncrono de video (Tapia, 2020).

Como se introdujo antes, el NCS es un dispositivo de computación neuronal, que ofrece un acceso fácil a las capacidades del aprendizaje profundo de alto rendimiento y de bajo consumo para aplicaciones IoT integradas que es precisamente lo que se realizó. Es un pequeño acelerador sin ventilador muy asequible para DNN y se basa en la unidad de procesamiento de visión (VPU) Intel Movidius y en su segunda versión la utilizada en el proyecto incorpora el Myriad X VPU, que posee un acelerador de hardware para inferencia DNN (Intel, n.d.). Durante la ejecución de un sistema utilizando el NCS 2 con un single shot multibox detector (SSD) se tiene un consumo del 10% y 15% de memoria RAM en promedio del CPU del Raspberry PI 3 donde el valor de referencia de mAP alcanza el 73% (Tapia, 2020). Tiene un bajo consumo de energía y es portable debido a su tamaño de memoria flash.

Gracias a OpenVino el NCS 2 es compatible con varios sistemas operativos, por lo que cualquiera puede utilizar una red entrenada en Raspbian y correr la aplicación en Windows, haciendo que el objetivo del proyecto de ser asequible para todos y reprogramable sea posible. Además, su uso permite reducir la carga del procesador del Raspberry Pi, permitiendo que este pueda realizar otras funciones simultaneas. Este acelerador cuesta aproximadamente 85 dólares en el mercado, un precio accesible que aun permite que nuestro proyecto se mantenga de bajo costo y como se explico con un bajo consumo de energía.

Figura 1. Raspberry PI, con el Neural Compute Stick y la camera module V2.

Tiny-YOLO

Para poder explicar el funcionamiento de ese detector en tiempo real, se debe entender como funciona el modelo YOLO, que es una red convolucional que predice simultáneamente

cuadros delimitadores, esta es la palabra clave para entender el modelo, y probabilidades de clase para esos cuadros, es por eso por lo que pertenece a los SSD.

Esta Red base corre a 45 FPS sin procesamiento por lotes hasta llegar a un máximo de 150 FPS en una Titan X, resultando en que la red YOLO obtenga el doble de mAP que otros detectores en tiempo real (Redmon et al., 2016). Para su entrenamiento YOLO únicamente necesita de imágenes completas de la clase u objeto que se desea detectar, ve la imagen completa durante el entrenamiento y el tiempo de prueba, por lo que codifica la información de la clase, así como también su apariencia, esto permite que la red generalice la representación del objeto. Claramente en términos de precisión esta por detrás de otros detectores porque, aunque pueda identificar de manera rápida un objeto en una imagen, batalla por localizar con precisión objetos pequeños.

Ahora Tiny-YOLO es un modelo de una Convolutional Neural Networks (CNN) que son parte de las DNN. Estas CNN son una pila de capas que se definen mediante la acción de varios filtros en la entrada, estos filtros se llaman “kernels”, por ejemplo la capa de agrupación máxima, devuelve el píxel con el valor máximo de un conjunto de píxeles dentro de un kernel, este a su vez barre a través de la entrada, submuestreando la capa. Tiny-YOLO consta de 9 capas convolucionales (CONV) y 3 capas completamente conectadas, para cada capa CONV el tamaño de la matriz kernel es de 3 y cada paso es de 1; específicamente para las primeras 6 capas CONV, cada una tiene una capa de agrupación máxima de 2 × 2 detrás, lo que reduce la resolución de las funciones y hace que la red sea robusta contra el ruido y distorsión (Ma et al., 2018).

Esto quiere decir que si a la entrada del modelo tenemos una imagen de 448x448, esta es dividida en una cuadricula de 7x7, cada celda de esta cuadricula predice 2 cuadros delimitadores y a su vez cada cuadricula es solo responsable de los objetos cuyo punto central este dentro de la misma.

Para representar un cuadro delimitador se requieren 5 parámetros obligatorios:

- Sus coordenadas en x

- Sus coordenadas en y
- Su ancho (w)
- Su alto (h)
- Su confianza

El parámetro de confianza indica si el cuadro delimitador actual contiene o no un objeto (medido por su probabilidad) y a su vez que tan precisa es la ubicación de este (medido por el IoU). Se calcula multiplicando estas dos métricas,

$$Confianza = Pr\,(Objeto) * IoU$$

y a su vez serán el tema principal de estudio de los resultados del proyecto.

En consecuencia, la salida se agrupa en segmentos de 7x7 y existe una correspondencia uno a uno entre una celda de la cuadricula y un segmento de salida. Cada segmento contiene 30 valores, 10 para dos cuadros delimitadores y 20 para probabilidades de clase. Estos últimos están indicados por pi que es una serie de probabilidades condicionales, por lo tanto para un cuadro delimitador, la probabilidad de la i-esima clase estará indicada por (Ma et al., 2018):

$$Pr\,(Clase_i \mid Objeto) * Pr(Objeto) * IoU = Pr\,(Clase_i) * IoU$$

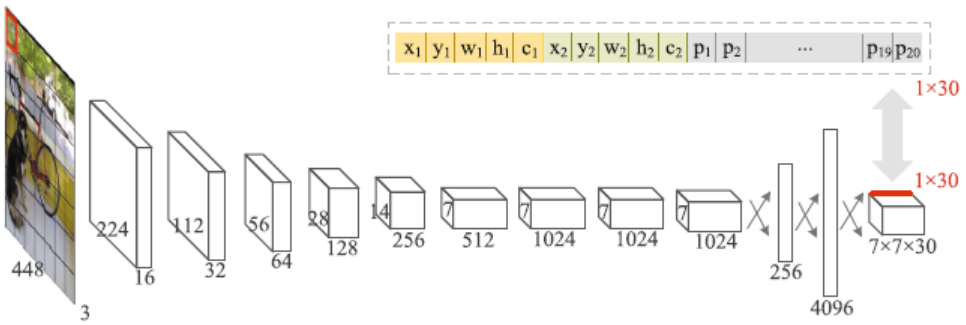

Figura 2. Estructura de la red tiny_YOLO.

Después de saber como funciona el modelo, es necesario entender algunos parámetros de la red, estos son:

- Tamaño de imagen que procesa la red: debe ser de 488 pixeles y esto debe ser fijo

para encajar con la red, es por eso que todas las imágenes que se le pasan son redimensionadas antes de entrar.

- Cantidad de cajas por imagen: se refiere a la máxima cantidad de objetos que se quieren detectar.
- Etiquetas: son las de los objetos que se desea detectar.
- Iteraciones: la cantidad de veces que la red entrenara sobre todo el dataset que se le mande.
- Tiempo de entrenamiento: este valor se refiera a la cantidad de veces que debe de entrenar una misma imagen.
- Nombre de pesos guardados: una vez entrenada la red, se guardan sus pesos en un archivo que se usa después para hacer las predicciones.

Esta red fue entrenada en el conjunto de datos Common objects in context (COCO), El COCO dataset contiene 91 categorías de objetos comunes, 82 de las cuales tienen más de 5,000 instancias etiquetadas. En total, el dataset tiene 2,500,000 instancias etiquetadas en 328,000 imágenes (Lin et al., 2014). En el caso de no haber utilizado este conjunto de datos, se hubiera tenido que crear uno con al menos 1000 imágenes etiquetadas con la clase deseada y de cada imagen se debe obtener un archivo .xml que contenga la clase y la posición del objeto en la imagen. Utilizar un dataset ya creado nos ahorra mucho tiempo y recursos, a parte de que es abierto al público en general por Microsoft, su creador.

Sistema de control

Una vez entrenada la red, probada y calibrada, esta detectara personas en un espacio delimitado, este podría ser una oficina, un restaurante, una tienda, entre otros. Lo que se quiera hacer con esta detección dependerá del sitio donde se implemente, como por ejemplo, en una oficina el encender las luces de los pasillos, el aire acondicionado, el proyecto en la sala de reuniones, etc. Por lo que un sistema de control que se conecte al dispositivo de detección es el siguiente paso, para esto haremos uso de los implementos del Raspberry PI, el

General Purpose Input Output (GPIO), un sistema de entrada y salida que consta de una serie de pines y conexiones que se usan para múltiples usos.

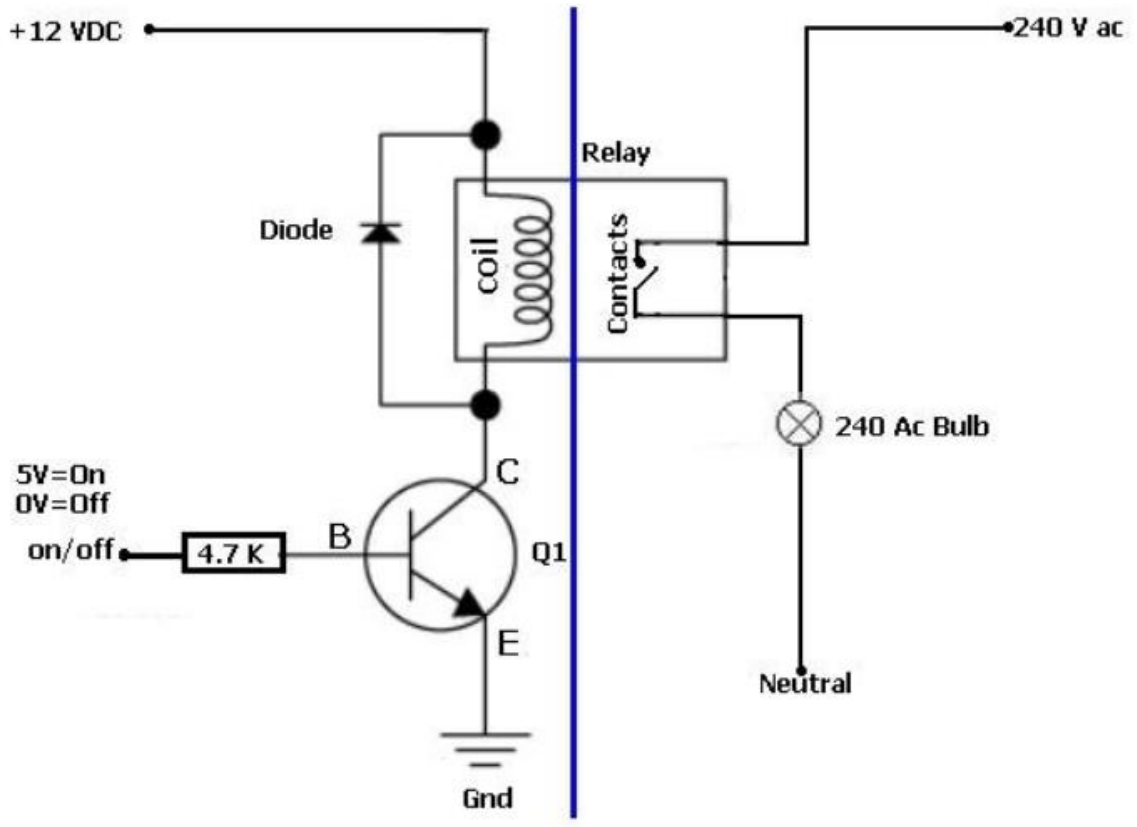

Figura 3. Circuito de activación de un relé por medio de un transistor.

El circuito realmente es simple, consiste en un relé que es alimentado con 5V por el GPIO 14 del Raspberry PI, este corresponde al Transmisor-Receptor Asíncrono Universal (UART), que es un dispositivo que controla los puertos y dispositivos serie de transmisión. Este relé esta protegido por un diodo conectado en paralelo inversamente polarizado, de manera que absorba picos de tensión causados cuando desactivamos el relé y la corriente que circulaba induzca una tensión elevada de polaridad opuesta. Este diodo que puede ser el 1N4007. Un transistor 2N3904 de juntura bipolar, esta conectado del modo como se ve en la figura tal, de modo que cierra el circuito poniendo a tierra el terminal de la bobina mientras que el otro terminal se encuentra conectado al positivo alimentado por la Raspberry PI. A la base de este esta conectado una resistencia de 4.7KΩ y después el control donde con 5V el relé se colocara en “on”, es decir con una salida alta, se supera la tensión de umbral de la base y esto permite la circulación de corriente entre base y tierra que lleva al transistor a estado de conducción cerrando el circuito de la bobina del relé y activándolo. y con 0V se colocara en

"off" es decir, cuando la salida del control es baja lo será también la base del transistor y por lo tanto no dejara pasar corriente entre emisor y colector para activar la bobina del relé. Los contactos del relé estarán conectados a una fuente de 110V para simular una acometida eléctrica cualquiera y también a un foco que simulara el aparato que deseamos activar.

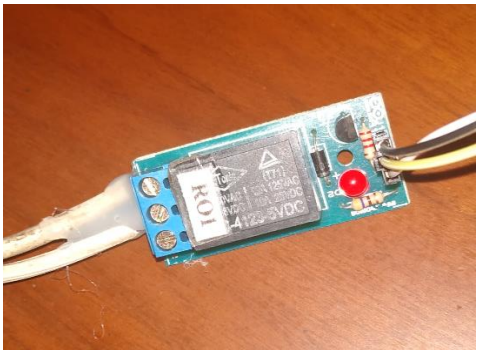

Figura 4. Circuito finalizado, soldado sobre una placa.

El control es el siguiente, hemos escogido una confianza general superior al 60%, esto con el fin de no activar el circuito si por error confunde algún objeto con una persona. cuando el sistema reconozca a la persona, enviara una señal de activación del GPIO14 que colocara en "alto" a la base y activara el relé, en el caso de que no detecte a nadie mandara una señal de falso al GPIO14 y este colocara a la base en "bajo", apagando el relé. Adicional al circuito antes mostrado se agrego una resistencia y un led a la alimentación de 5V para saber si mi detector se conecto al circuito de control.

RESULTADOS

Los resultados de este proyecto están divididos en 2 partes, la primera que se centra en la eficiencia del detector de objetos creado y la segunda que se centra en cómo después de reconocer y contar a la persona se activa el sistema de control

Eficiencia del detector

Ya que el proyecto se centra más en la detección de un tipo de clase, en este caso personas, y en la eficacia de esta detección, la confianza, se realizó varias pruebas con dos los parámetros específicos que se utilizan para medir la precisión de un detector de objetos en un conjunto de datos en particular y su confianza. Estas dos métricas se pueden variar en el código, de esa manera se puede ajustar el detector según el objetivo que se quiera para este.

El primer parámetro evaluado es el “Intersection over Union” (IoU) que es una buena métrica para medir la superposición entre dos cuadros delimitadores o máscaras. La evaluación del IoU se construye a partir de:

- Los cuadros delimitadores de verdad del terreno, es decir, los cuadros delimitados previamente y etiquetados a mano con el que se entrenó al modelo, en este cuadro esta especificado en qué parte de la imagen está el objeto. Como se utilizo un modelo pre entrenado estos cuadros de verdad se ubican en la carpeta del dataset con el que el modelo aprendió.

- Los cuadros delimitadores que son predichos durante las pruebas por el modelo entrenado.

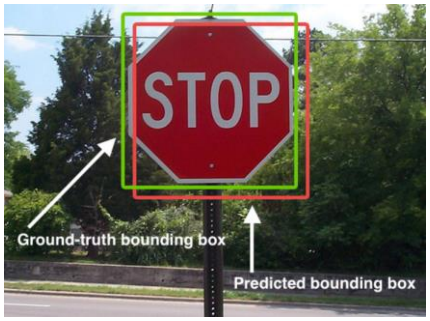

Figura 5. Un ejemplo de detección de una señal de pare en una imagen. El cuadro delimitador predicho se dibuja en rojo, mientras que el cuadro delimitador de verdad del terreno se dibuja en verde.

Este valor va de cero a uno, siendo uno una predicción completamente correcta, es decir un detector ideal, lo cual causaría problemas de rendimiento dado que el programa buscaría objetos que cuadren perfectamente con las imágenes con las que fue entrenado. Es por eso que se acostumbra a utilizar valores desde 0.5 a 0.1 para este parámetro, de esta manera el programa detectara cualquier objeto que pueda calzar con los cuadros delimitadores de verdad [7], con esto el detector se vuelve más eficiente para términos de detección en tiempo real.

Al momento de correr el código en el Raspberry PI, este busca en la carpeta de configuración que IoU deseamos para la prueba, con esto en mente, se planteo cinco valores diferentes para este parámetro (0.15, 0.25, 0.35, 0.45 y 0.55). Se probo el detector con un video base de personas comprando en un supermercado grabado por un celular con una calidad media. En cada prueba se imprimió el porcentaje de confianza, es decir cuanto reconoció de una persona en el video. Se eligió solamente imprimir cuando detectara en un 50% o mas a una persona, de esa manera se podría evaluar que IoU escogido reconoce de mejor manera a personas en movimiento.

Figura 6. Prueba con video base de personas circulando en un supermercado.

Se tomaron 100 muestras para cada IoU, con esto se calculo la tabla de frecuencia para cada IoU, con esto se puede ver que valor no solo reconoce más veces a una persona con un 60% o mas de confianza si no que también se puede ver con que valor de IoU se obtuvo más veces una detección superior a un 90%, lo cual es un valor de confianza excelente para detectores en tiempo real.

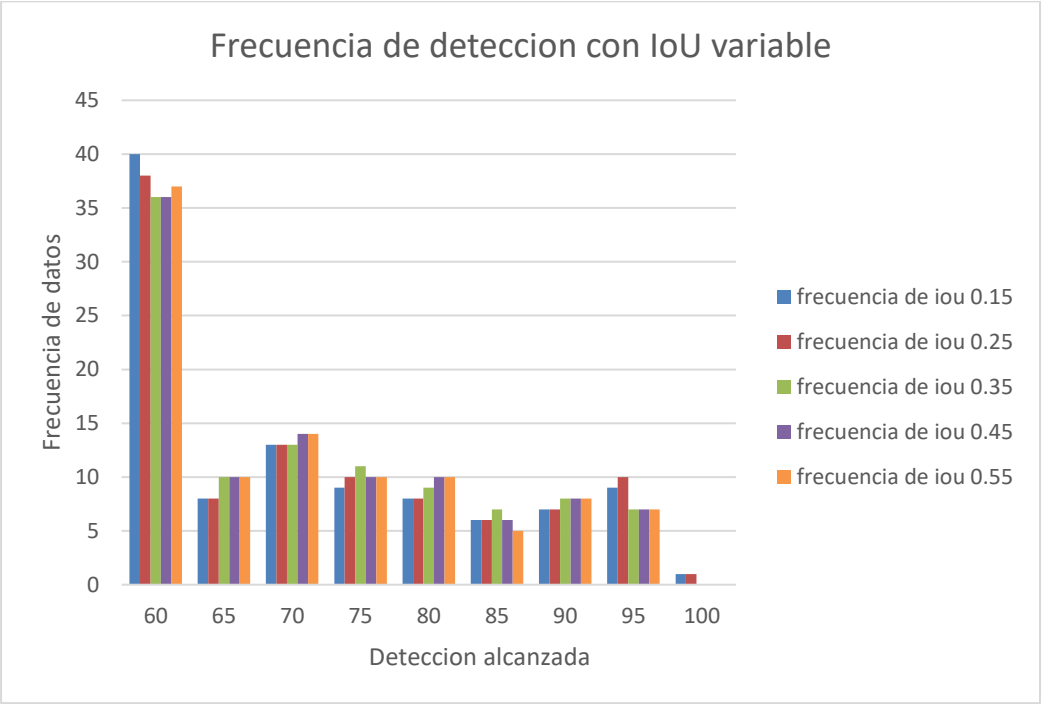

Figura 7. Frecuencia de detección alcanzada con cada valor de IoU para 100 muestras del video base.

Con los datos evaluados se determino que para el detector del proyecto un IoU de 0.25 es el que de mejor manera reconoce a personas con una confianza superior al 60% y además es de los que mas personas detecto con una confianza superior al 90%.

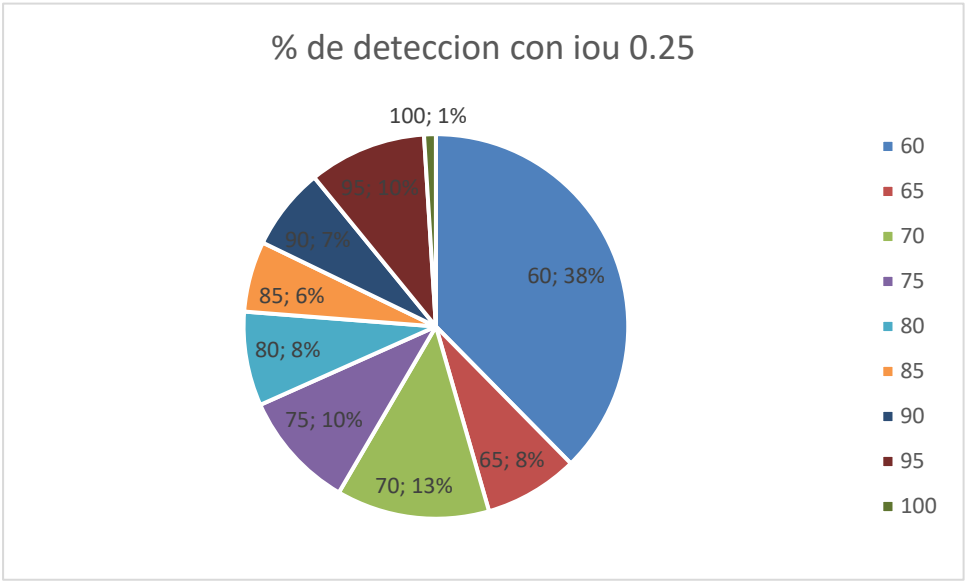

Figura 8. Porcentaje de detección obtenido al colocar un IoU de 0.25.

El segundo parámetro es el tamaño de probabilidad, donde se mide que tan seguido un evento se espera que ocurra. Este es la razón del tamaño del espacio de eventos, que es el número de resultados que interesan, con el tamaño del espacio muestral que es el número total de posibles resultados.

Para este caso el evento que interesa es la clase “persona” por lo que para esta prueba se

incluyo todos los eventos donde se detectara a una persona sin importar la confianza con la que se detecto a diferencia del parámetro anterior. Para verificar esta métrica se probaron tres valores de tamaño de probabilidad (0.2, 0.5 y 0.8).

Primero se realizó un mapeo general con 50 pruebas con cada probabilidad de donde ya se percibió que el valor de probabilidad que presentaba una mejor detección de personas era el de 0.8, pero igual era necesario analizar la frecuencia de detección de los otros umbrales para así determinar si alguno también era útil para el modelo.

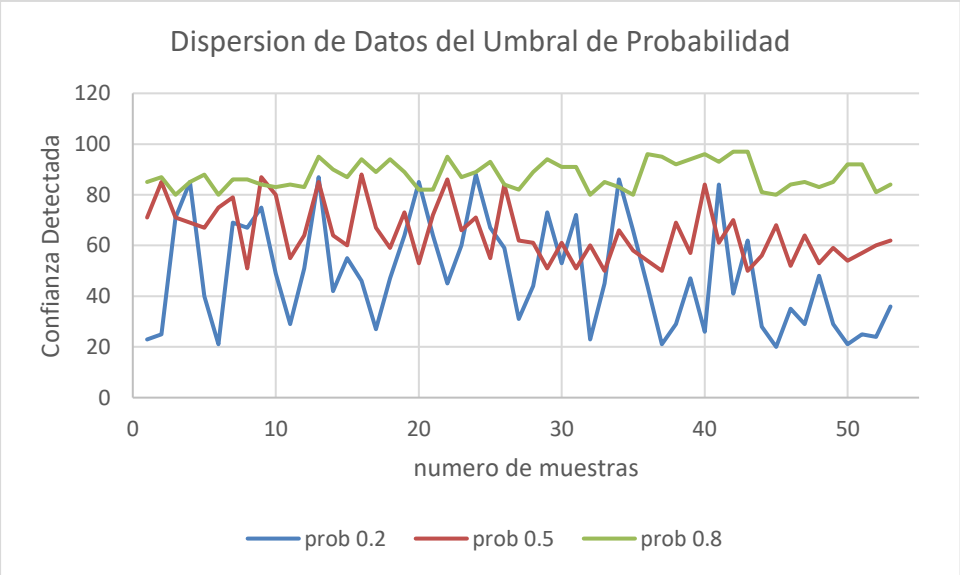

Figura 9. Dispersión de Datos de 50 muestras con los 3 diferentes valores de probabilidad.

Por lo observado en la figura 9, para los dos primeros casos fueron necesarias tomar 100 muestras ya que los valores oscilaban entre el 20% y el 90% de confianza, lo cual dificultaba el análisis de frecuencia. Pero para el ultimo valor fueron necesarias solo 53 muestras puesto que todos los valores de confianza se encontraban entre el 80% y el 97%, por lo que si se tomaban mas datos no hubieran representado una diferencia en los resultados.

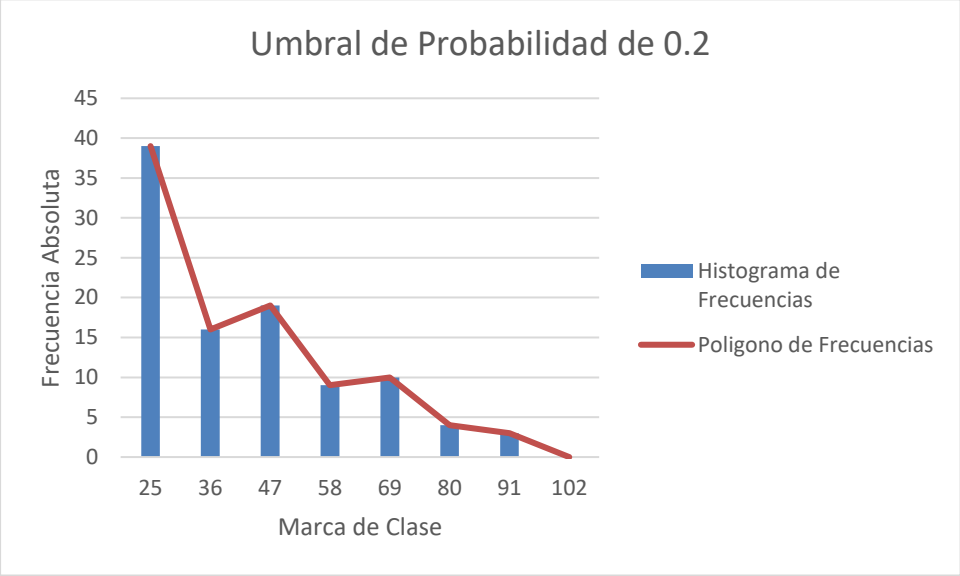

Figura 10. Histograma de frecuencias con polígono de frecuencia para una probabilidad de 0.2.

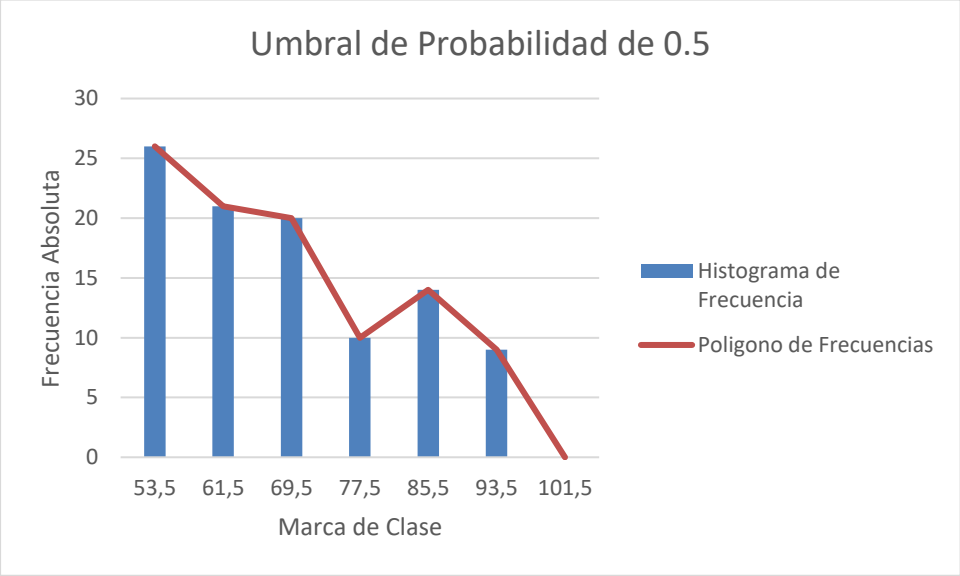

Figura 11. Histograma de frecuencias con polígono de frecuencia para una probabilidad de 0.5.

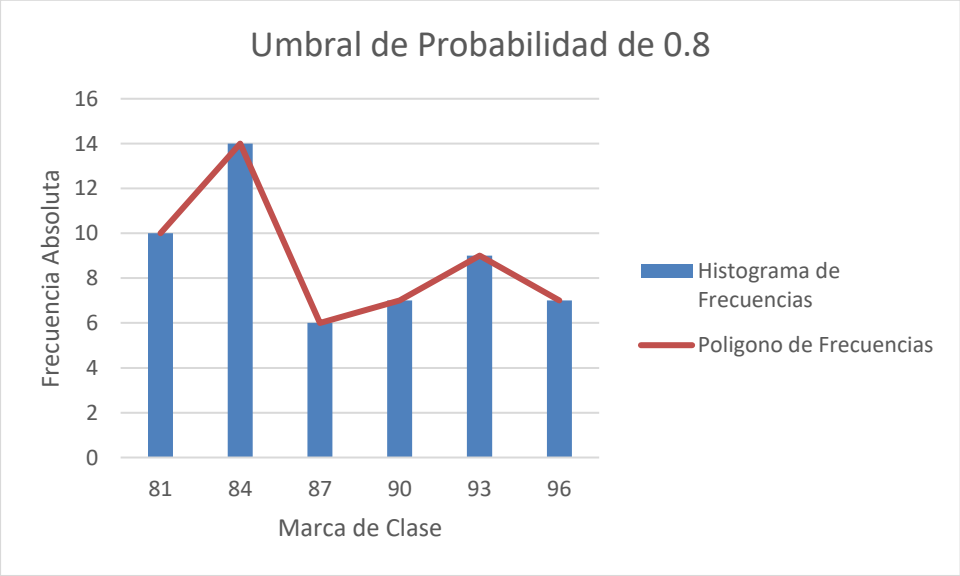

Figura 12. Histograma de frecuencias con polígono de frecuencia para una probabilidad de 0.8.

Con el análisis de cada umbral testeado, se corroboro la idea planteada al momento de

realizar un mapeo general donde se observo en la figura 4, que al colocar un tamaño de probabilidad de 0.8 en el detector, con el que se obtienen detecciones con un alto grado de confianza.

Definidos ya los dos valores ideales para el detector de este proyecto se realizaron tres pruebas con la cámara grabando en tiempo real. Para cada prueba se tomaron 25 muestras, es decir el sujeto de prueba paso 25 veces en frente de la cámara y se registro la confianza con la que se lo detecto. Las pruebas fueron caminando rápido y con poca iluminación (prueba 1), caminando rápido con buena iluminación (prueba 2) y por ultimo caminando lento con buena iluminación (prueba 3).

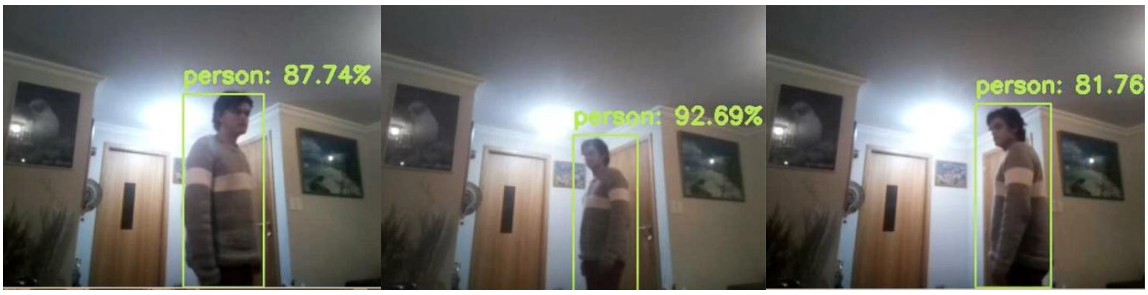

Figura 13. Capturas de pantalla de la prueba 3.

Como se muestra en la figura 13, al momento de tener una buena iluminación y caminar a un paso relajado enfrente de la cámara, el detector capta de mejor manera a la persona obteniendo 7 veces valores superiores al 94% de confianza de las 25 muestras. Mientras que con poca luz y caminando más rápido el rango de detección cae hasta obtener 8 muestras con un 82% de confianza y 10 muestras con un 85% de confianza; esto no significa que el detector este mal, al contrario, indica que aun en condiciones exageradas como personas caminando rápido y con baja iluminación del entorno, el modelo responde de buena manera.

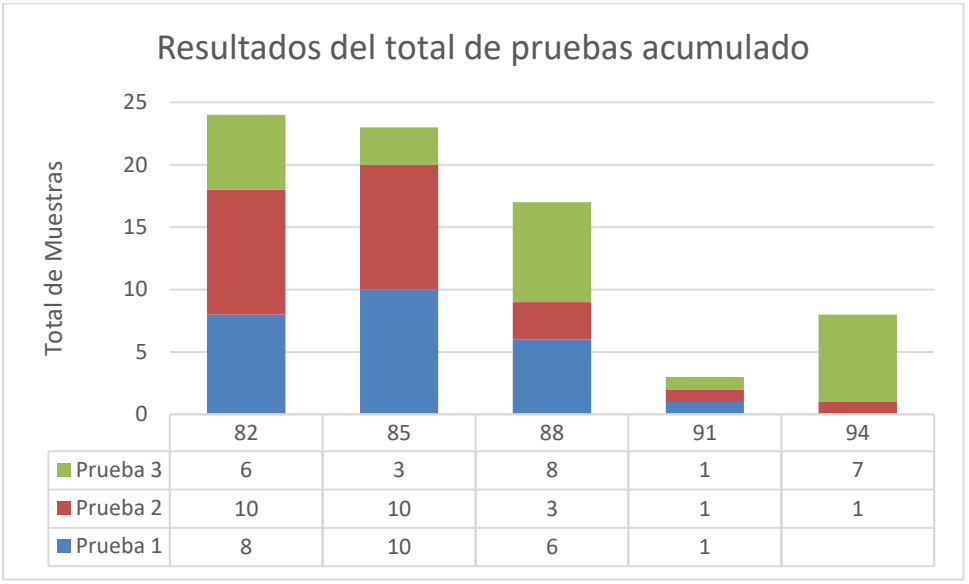

Figura 14. Resultado del total de pruebas realizadas en tiempo real.

Se intento hacer una prueba con el sujeto corriendo tanto con buena iluminación como con mala iluminación, pero al parecer el modelo si necesita de un tiempo mínimo de exposición del objeto para predecir el cuadro de detección. Pero en vista que este prototipo se planea implementar en espacios cerrados, variables como personas corriendo serán casi cero.

Sistema de Control Funcionando

Como se planifico en el comento que el Sistema reconoce a una persona con un nivel de confianza igual o superior al 60%, el GPIO 14 se pone con valor verdadero, lo que indica se pone a 5V, activando así el relé y por consiguiente el foco a 110V.

Figura 15. Foco encendido a 110V por medio del relé al momento de detectar una persona con confianza igual o mayor al 60%.

La manera de comprobar que el sistema esta listo para funcionar lo indica el led incorporado al circuito que se enciende, mostrando que el Raspberry PI alimenta al relé con 5V. este indicador podría colocarse también a la salida de los contactos del relé para observar el cambio de nivel en la base que enciende y apaga el mismo.

Figura 16. Led encendido indicando que la Raspberry PI está alimentando al relé con 5V.

CONCLUSIONES

Al inicio del Proyecto se planteó lo siguiente, realizar un prototipo de bajo costo que pudiera detectar personas en tiempo real, que sea de software libre y reproducible, con el que se activara un circuito de control para la administración de aparatos electrónicos.

El costo de fabricación de este prototipo no supera los 200 dólares, dado que se utilizaron los productos más asequibles pero que también cumplan la función de entrenar una red neuronal, como lo son el Raspberry PI y el NCS 2, la cámara propuesta en el proyecto puede ser incluso opcional, dado que la cámara que viene incluida en el kit de Raspberry PI también ofrecerá resultados satisfactorios, de esta manera se puede reducir aún mas el costo del prototipo. El software utilizado en el prototipo es de libre adquisición, el sistema operativo Raspbian GNU/Linux viene incluido en los paquetes con la compra de la Raspberry PI o también puede ser descargado desde la web oficial del mismo.

El lenguaje utilizado en el código fue Python, que según la IEEE Spectrum fue el lenguaje más utilizado en el 2019, así mismo lo definen como un lenguaje de programación multiparadigma, dinámico y multipropósito, que es fácil de aprender, usar y comprender. Está desarrollado bajo una licencia de código abierto, por lo que es de libre uso y distribución, incluso para uso comercial, entonces cualquier persona puede acceder al código y modificarlo para otras aplicaciones.

La detección en tiempo real con tiny-YOLO resulto ser la esperada, al ser pre entrenada con el COCO dataset, se ahorro tiempo de entrenamiento y de implementación, el dataset también es de acceso libre, y al poseer mas de 91 tipos de clases, el código puede modificarse para detectar cualquier objeto. El detector funciona bien para condiciones especiales como movimientos acelerados de la persona detectada o la falta de iluminación de la habitación. En términos de confianza al colocar un IoU de 0.25 y una probabilidad de 0.8, se logro que el detector pudiera reconocer personas casi siempre con una confianza superior al 80%. Una recomendación a esto seria en el caso de querer detectar varias clases a la vez reducir la

probabilidad y mantener el IoU en valores bajos como 0.1 a 0.3, de esta manera red buscara mas objetos en una misma imagen. Dado que el objetivo del prototipo era detectar únicamente personas, la probabilidad alta resulta la mejor opción.

En cuanto al circuito electrónico para el control mediante la detección de personas, funciona con 110V, lo que se obtiene en cualquier lugar del país. En el caso de querer implementar este proyecto en otro país o en otro continente donde haya por ejemplo 240V, esto es sencillo de lograr. Como la tensión de control es independiente de la tensión con la que se alimenta al relé se puede trabajar con tensiones separadas, esto provee de libertad a la hora de elegir la tensión deseada y de conectar distintos tipos de relé sin necesidad de modificar la parte de control.

En conclusión, el prototipo realizado cumple todos los puntos deseados. Y tiene crecimiento a futuro para aplicaciones en internet of things (IoT), si reemplazamos el control por cable a una conexión por internet, y también otros sistemas de automatización de espacios para la gestión de la energía, como por ejemplo apagar todos los aparatos y dispositivos de un lugar donde no se detecte a nadie ocupándolos. El límite estará dado solo lo pondrá la siguiente persona que modifique y mejore este prototipo.

REFERENCIAS BIBLIOGRÁFICAS

All3DP. (2021). *Raspberry Pi vs Jetson Nano: The Differences in 2021*. https://all3dp.com/2/raspberry-pi-vs-jetson-nano-differences/

Alpaca Rendón, J. (2017). *Reconocimiento de Patrones de Movimiento Basado en Raspberry Pi y Cámaras Megapíxel Para Mejorar la Atención de Pacientes Hospitalizados*. UNIVERSIDAD CATÓLICA DE SANTA MARÍA.

Francisco Blanco Esquivel. (2020). *Implementación de redes neuronales en sistemas empotrados de altas prestaciones*. UNIVERSIDAD AUTÓNOMA DE MADRID.

Huang, R., Pedoeem, J., & Chen, C. (2019). YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers. *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, 2503–2510. https://doi.org/10.1109/BigData.2018.8621865

Intel. (n.d.). *Dispositivo de computación neuronal: aprendizaje más profundo y rápido del desarrollo IoT*. Retrieved May 11, 2021, from http://www.interempresas.net/Informatica_Industrial/FeriaVirtual/Producto-Dispositivo-de-computacion-neuronal-Intel-Neural-Compute-Stick-2-173317.html

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8693 LNCS*(PART 5), 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

Ma, J., Chen, L., & Gao, Z. (2018). Hardware implementation and optimization of tiny-YOLO network. *Communications in Computer and Information Science*, *815*, 224–234. https://doi.org/10.1007/978-981-10-8108-8_21

Raspberry. (2016). *Camera Module V2*. https://www.raspberrypi.org/products/camera-module-v2/

Raspberry Pi Organization. (n.d.). *Raspberry Pi 3 Model A+ – Raspberry Pi*. https://www.raspberrypi.org/products/raspberry-pi-3-model-a-plus/

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Rosebrock, A. (2020). *YOLO and Tiny-YOLO object detection on the Raspberry Pi and Movidius NCS*. https://www.pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-ncs/

Tapia, D. (2020). Rendimiento de Raspberry Pi 3B+ con Intel NCS ejecutando métodos de aprendizaje profundo neuronal en clasificación de tipo de vehículos en tiempo continuo. [Universidad del Azuay]. In *Universidad del Azuay*. http://dspace.uazuay.edu.ec/bitstream/datos/6819/1/07260.pdf

Uijlings, J. R. R., Van De Sande, K. E. A., Gevers, T., Smeulders, A. W. M., Girshick, R., Ren, S., He, K., Girshick, R., Sun, J., Johnson, J., Karpathy, A., Redmon, J., Divvala, S., Girshick, R., Farhadi, A., Networks, R. F. C., Dai, J., Chen, X., Gupta, A., … Berg, A. C. (2018). Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2017-Janua*(1), 2–9. http://ieeexplore.ieee.org/document/6909475/%0Ahttp://arxiv.org/abs/1811.04533%0Ahttps:/

/doi.org/10.1016/j.patcog.2017.10.013%0Ahttp://openaccess.thecvf.com/content_cvpr_2017/html/Redmon_YOLO9000_Better_Faster_CVPR_2017_paper.html%0Ahttp://arxiv.org/abs/1

Zapata, O., & Rivera, D. (2016). *Sistema de detección de movimiento para uso residencial, con notificación a móviles, utilizando el microcomputador Raspberry PI.* UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA.