TEC | Tecnológico de Costa Rica

Tecnológico de Costa Rica

Escuela Ingeniería Electrónica

Proyecto de Graduación

“Diseño de un prototipo electrónico para el control automático de la luz alta de un vehículo mediante detección inteligente de otros automóviles.”

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura

Ronald Miranda Arce

San Carlos, 2018

INSTITUTO TECNOLOGICO DE COSTA RICA

ESCUELA DE INGENIERIA ELECTRONICA

PROYECTO DE GRADUACIÓN

TRIBUNAL EVALUADOR

ACTA DE EVALUACIÓN

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Estudiante: Ronald Miranda Arce

Carné: 2013025663

Nombre del Proyecto:

Diseño de un prototipo electrónico para el control automático de la luz alta de un vehículo mediante detección inteligente de otros automóviles.

Miembros del Tribunal

Ing. Leonardo Cardinale Villalobos.

Profesor lector

Ing. Javier Rivera Alvarado.

Profesor lector

Ing. Luis Diego Gómez Rodríguez.

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Nota Final del Proyecto de Graduación: 100

San Carlos, 19 de febrero, 2019

Declaración de autenticidad

Yo, Ronald Miranda Arce, en calidad de estudiante de la carrera de ingeniería Electrónica, declaro que los contenidos de este informe de proyecto de graduación son absolutamente originales, auténticos y de exclusiva responsabilidad legal y académica del autor.

Ronald Fernando Miranda Arce

Santa Clara, San Carlos, Costa Rica

Cédula: 207320032

Resumen

Se presenta el diseño y prueba de un prototipo electrónico para el control automático de la luz alta de un vehículo mediante la detección inteligente de otros automóviles, de esta forma mejorar la experiencia de manejo y seguridad al conducir en la noche, disminuyendo el riesgo de accidentes por deslumbramiento en las carreteras.

Se expone el uso de *machine learning* (máquina de aprendizaje automático) para el entrenamiento de una red neuronal, que permita el reconocimiento en profundidad de patrones, en este caso identificar el patrón que describe un vehículo cuando se acerca durante la noche.

Se entrenó exitosamente una red neuronal capaz de identificar correctamente hasta un 89% de los casos ocurridos en las pruebas realizadas, generando las señales para controlar de forma automática los cambios de luz.

Summary

The design and testing of an electronic prototype for the automatic control of the vehicle's high light through the intelligent detection of other automobiles is presented.

This improves driving experience and driving safety at night, reducing the risk of accidents due to glare on the roads.

It exposes the use of machine learning for the training of a neural network, that allows deep recognition of patterns, in this case identify the pattern that describes a vehicle when it approaches.

A neural network capable of correctly identifying up to 89% of the cases occurred in the tests performed was successfully trained, generating the signals to automatically control the high beam and low beam.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1: Introducción

El proyecto se desarrolló para la empresa TransYamil, la cual cuenta con una flotilla de tres busetas las cuales realizan giras turísticas a Panamá y Nicaragua, por lo que los choferes manejan por periodos de siete a diez horas seguidas, debido a esto la empresa desea mejorar las condiciones de trabajo de sus conductores y la seguridad de sus clientes, por medio de un sistema que permita hacer más cómoda y segura la conducción durante la noche.

Se propone un sistema que mejore la experiencia de manejo, debido a que se olvidarán por completo de estar realizando el cambio manual de luces cada vez que se acerca otro vehículo, de esta forma se espera que se mejore la experiencia al conducir, y que esta se vuelva más placentera y segura.

Se pretende desarrollar un prototipo de un dispositivo que controle de forma automática la luz alta de un vehículo mediante detección inteligente de otros automóviles.

El algoritmo de detección de las luces se llevó a cabo mediante la implementación de *machine learning* (aprendizaje automático) utilizando las herramientas de OpenCV y TensorFlow las cuales permiten el reconocimiento de patrones.

1.1 Problema existente e importancia de su solución

El cambio inoportuno de las luces altas del vehículo por parte de conductores desatentos puede provocar accidentes en la vía, debido a encandilamientos. Los conductores se enfrentan a un problema cuando se conduce de noche, debido al riesgo de accidente por deslumbramiento producto de las luces de gran intensidad (luz larga) que incide directamente sobre los ojos del conductor.

Existe un efecto en la salud del conductor asociado con esta situación llamado el efecto Troxler, el cual provoca un tipo de ceguera temporal, por lo que el tiempo de reacción del conductor se incrementa hasta en un segundo y medio, lo que a cien kilómetros por hora supone 42 metros recorridos por el vehículo, la duración del efecto puede aumentar según, la edad, cansancio y tiempo al volante. [1]

Una reacción tardía debido al deslumbramiento del conductor según la situación podría causar un accidente mortal como el choque entre carros o el atropello de un peatón.

El mecanismo manual para el cambio de las luces del vehículo es susceptible a fallas por parte de los conductores, debido a que realizar la tarea repetitiva de mover la manilla de la luz puede ser tedioso, no activar las luces por miedo de cegar a los conductores que se aproximan o simplemente son poco atentos y olvidan una tarea tan importante, que sumado a la oscuridad de las rutas rurales pueden provocar accidentes. [2]

Se propone un sistema que se encargue de reconocer cuando un vehículo se acerca, para realizar el cambio de luz automáticamente. Al resolver este problema se espera que se mejore la experiencia de manejo y disminuya el riesgo de accidentes, debido al encandilamiento de otros conductores. Al utilizar luces altas con mayor frecuencia, el sistema permite la detección temprana de peatones y obstáculos, además, mejora el rango de visión del conductor y se optimiza el uso de las luces altas. [3]

1.2 Solución seleccionada

La empresa TransYamil se preocupa por mejorar la comodidad de sus choferes, ya que manejan alrededor de diez horas seguidas en las giras turísticas que realizan, la mayor parte del tiempo durante la noche.

De la misma forma TransYamil busca hacer una inversión para mejorar la seguridad en carretera de sus choferes y clientes, disminuyendo el riesgo de accidentes por deslumbramiento en las carreteras. Por otro lado, se prevé un posible negocio, debido al apogeo de los carros autónomos y a las posibles regulaciones futuras que obliguen la implementación de medidas similares de seguridad y confort al conducir.

Se diseñó e implementó un prototipo para el cambio automático de las luces de un vehículo, con el objetivo de aumentar el confort de los conductores durante la noche, debido a que se espera que estos dejen de preocuparse por estar realizando continuamente el cambio manual de luces cada vez que se acerca otro vehículo, de esta manera evitar los accidentes causados por encandilamientos por parte de choferes desatentos.

Esta solución se plantea como el desarrollo de un algoritmo que mediante una red neuronal procese video y sea capaz de detectar un vehículo para controlar los cambios de luz automáticamente. El prototipo cuenta con un computador embebido conectado a una cámara y a un faro delantero de prueba, como se muestra en la figura 1.

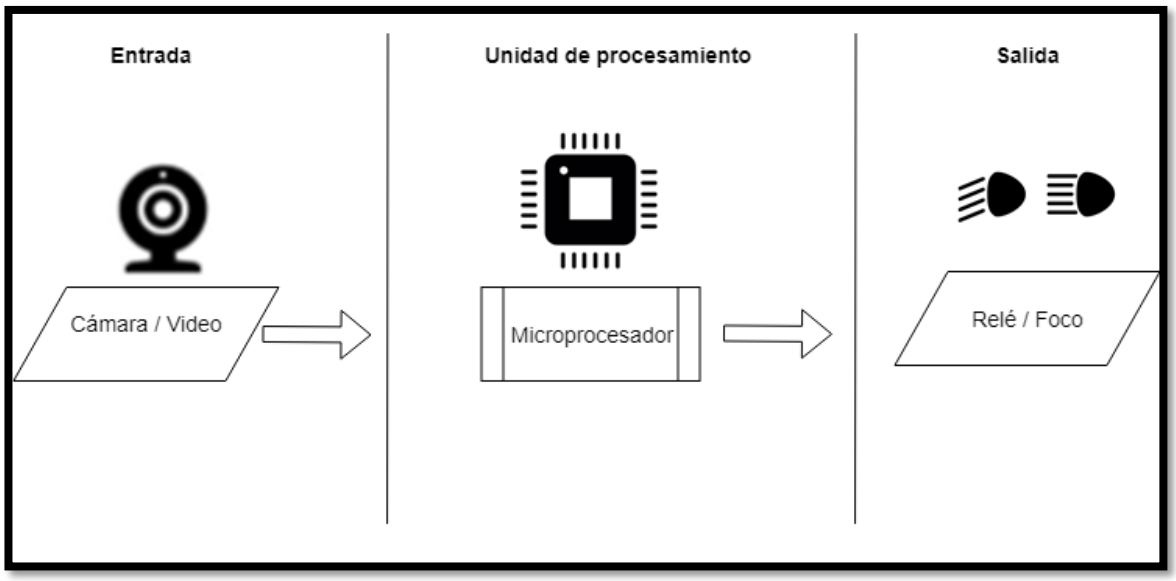


Figura 1. Solución seleccionada

Fuente: Propia

Además, el prototipo debe soportar las variaciones de voltaje que se puedan dar en un vehículo, por lo que se implementó un regulador de tensión.

El algoritmo para el procesamiento de video se desarrolló en Python y la red neuronal se entrenó utilizando las librerías de TensorFlow, herramienta que permite el aprendizaje automático y en profundidad de patrones.

Capítulo 2: Meta y objetivos

2.1 Meta

Mejorar la experiencia de manejo y seguridad al conducir de noche, disminuyendo el riesgo de accidentes por deslumbramiento en las carreteras.

2.2 Objetivo general

Desarrollar un prototipo electrónico que controle de forma automática la luz alta de un vehículo mediante detección inteligente de otros automóviles.

- **Indicador**: Se realiza el cambio automático de la intensidad de la luz según la detección inteligente de otros vehículos.

2.3 Objetivos específicos

- Desarrollar una red neuronal que detecte un vehículo durante la noche con una precisión de al menos un 70%.
  - **Indicador**: La red neuronal detecta de forma correcta más de un 70% de las imágenes de prueba.

- Desarrollar un algoritmo que junto con la red neuronal reconozca en tiempo real los vehículos.
  - **Indicador**: El porcentaje de eventos identificados correctamente debe superar el 80%.

- Implementar el algoritmo en un computador versátil y de bajo costo que controle la luz corta y larga de un foco de prueba.
  - **Indicador**: Tiempo de respuesta del algoritmo en el sistema embebido.

Capítulo 3: Marco teórico

Se dan a conocer los diversos componentes utilizados en este proyecto, además de algunos temas referentes al aprendizaje automático considerados en el diseño de la solución. Se presentan las herramientas computacionales y los diferentes elementos electrónicos involucrados, sus funciones y la justificación del por qué estos fueron utilizados.

3.1.1 Aprendizaje automático

El aprendizaje de máquina es un método de análisis de datos, que automatiza el desarrollo de modelos analíticos. Es una rama de la inteligencia artificial basada en la idea de que los sistemas pueden aprender de los datos, en este caso de imágenes para identificar patrones, como se muestra en la figura 2 y tomar decisiones con la mínima intervención humana. [4]

Se emplea una biblioteca de software libre para realizar cálculos numéricos mediante diagramas de flujo de datos donde los nodos de los diagramas representan operaciones matemáticas y las aristas reflejan las matrices de datos multidimensionales denominados tensores.

Figura 2. Uso del aprendizaje automático

Fuente: Propia

3.1.2 Redes neuronales

Una red neuronal es un conjunto de nodos o capas conectados que modelan una red de neuronas, es decir un cerebro biológico. Las conexiones entre las neuronas artificiales se llaman *"edges"* (bordes). Las neuronas artificiales y los bordes suelen tener un *"weight" (peso)* que se ajusta a medida que avanza el aprendizaje. En la figura 3, se muestra la representación lógica de una red neuronal.

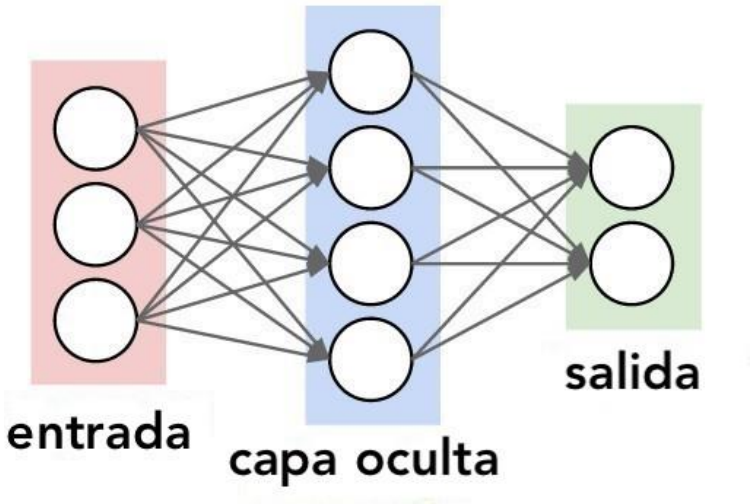


Figura 3. Red Neuronal

Fuente: [17]

3.1.3 Aprendizaje por transferencia

Los modelos modernos de reconocimiento de imágenes tienen millones de parámetros que se pueden modificar para obtener un resultado deseado. Sin embargo, desarrollar un modelo desde cero requiere gran cantidad de datos para entrenarlos y etiquetarlos correctamente, además de una gran cantidad de potencia de cálculo (horas de GPU). [5]

El aprendizaje de transferencia es una técnica que permite tomar una pieza de un modelo que ya ha sido entrenado en una tarea y reutilizarlo en un nuevo modelo para una tarea similar. Existe un banco de redes neuronales disponibles creadas para identificar distintos tipos de objetos, entre las más destacadas se encuentran “Mobilenet” e “Inception” y sus distintas versiones.

3.1.4 Vector de características de imagen o “Bottleneck”

Para entrenar una red neuronal primero se deben de crear los *bottleneck* (cuello de botell*a)*, lo cual consiste en analizar las imágenes en el disco para calcular el vector de características de la imagen y guardar en cache el vector correspondiente de cada imagen. [5]

El bottleneck es una de las capas internas de la red neuronal, la cual genera un conjunto de valores que son lo suficientemente confiables para distinguir entre todas las clases (objetos) que se le ha pedido que reconozca.

Eso significa que es un resumen significativo y compacto de las imágenes, sin embargo, el vector de características que ya fue calculado para estas imágenes puede ser reutilizado para reentrenar una nueva red neuronal y distinguir entre nuevos tipos de objetos. [5]

3.1.5 Reentrenamiento de una red neuronal

Cuando el cálculo del vector de características esta completado se puede empezar a reentrenar la capa final de la red. En cada proceso de instrucción se obtiene el valor de la precisión de entrenamiento, precisión de validación y la función de pérdida o entropía cruzada.

Una verdadera medida del rendimiento de la red es medir su rendimiento en un conjunto de datos diferentes a las imágenes utilizadas en el entrenamiento, esto se mide por medio de la precisión de validación. [5]

La precisión del entrenamiento muestra qué porcentaje de las imágenes utilizadas en el conjunto de imágenes del entrenamiento actual se etiquetaron con la clase correcta.

La entropía cruzada es una función de pérdida que permite identificar qué tan bien está progresando el proceso de aprendizaje. El objetivo del entrenamiento es hacer que la pérdida sea lo más pequeña posible, de modo que se pueda conocer si el aprendizaje está funcionando y cuándo es adecuado parar el entrenamiento, es decir que la pérdida sigue una tendencia descendente

Existen varios tipos de redes neuronales base, entre las más relevantes están las redes tipo “Inception”, las cuales son redes convolucionales profundas con modelos de gran tamaño, con varias capas y un costo computacional elevado, pero con ganancias en la precisión. Por otro lado, las redes tipo “Mobilenet” presentan modelos eficientes de menor tamaño para aplicaciones móviles y equipos con menor capacidad de procesamiento. [6]

El valor aceptado de pérdida cambia según la red neuronal base, para las redes tipo “Inception” se debe de procurar obtener un valor estable menor a 0.1, por otro lado, para las redes neuronales “Mobilenet” el valor de pérdida debe de ser menor a dos. [6]

En cada paso del entrenamiento se eligen diez imágenes al azar, se asocia con su vector de características y se introducen en la capa final para obtener las predicciones. Estas predicciones son comparadas contra las etiquetas reales (nombre correcto) para actualizar los pesos de la capa final.

Los problemas más comunes provienen de la calidad de datos (en este caso imágenes) que se suministran como entrada para entrenar la red.

Las imágenes deben ser una representación probable del escenario, que el usuario final de la aplicación encontrará.

Conociendo los conceptos, herramientas y la forma de cómo se va a hacer la detección inteligente de los vehículos cabe mencionar el hardware que es capaz de manipular y captar este tipo de información. Se mencionan las especificaciones de tres placas de desarrollo, las cuales se contemplan como alternativas para realizar la tarea de procesamiento de imagen.

3.1.6 Microprocesador Raspberry Pi 3

Es un dispositivo con las tres unidades principales de un computador, memoria, unidad central de procesamiento (CPU) y periféricos de entrada y salida, el cual puede ser programado para realizar distintas tareas. En la figura 4, se muestra la Raspberry Pi 3. Especificaciones de la Raspberry Pi 3:

- Procesador: 1,2 GHz de cuatro núcleos ARM Cortex-A53
- GPU Dual Core VideoCore IV Multimedia
- RAM: 1GB LPDDR2.
- Conector GPIO:
    - Proporciona 27 pines GPIO, así como 3,3 V, +5 V y GND líneas de suministro

Figura 4. Raspberry Pi 3

Fuente: [15]

3.1.7 Placa de desarrollo Nvidia Drive AGX

Es una plataforma escalable de computo que permite el desarrollo de aplicaciones para vehículos autónomos. El equipo es capaz de procesar en tiempo real datos provenientes de cámaras, radar y lidar. En la figura 5, se muestra la plataforma Nvidia AGX.

Especificaciones NVIDIA Drive AGX:

- Procesador: 8-core CPUs basados en ARM
- GPU: Volta-class GPU
- Proporciona 16 interfaces CAN de entrada y salida.
- Ancho de banda de la memoria GPU 750 Gb/s

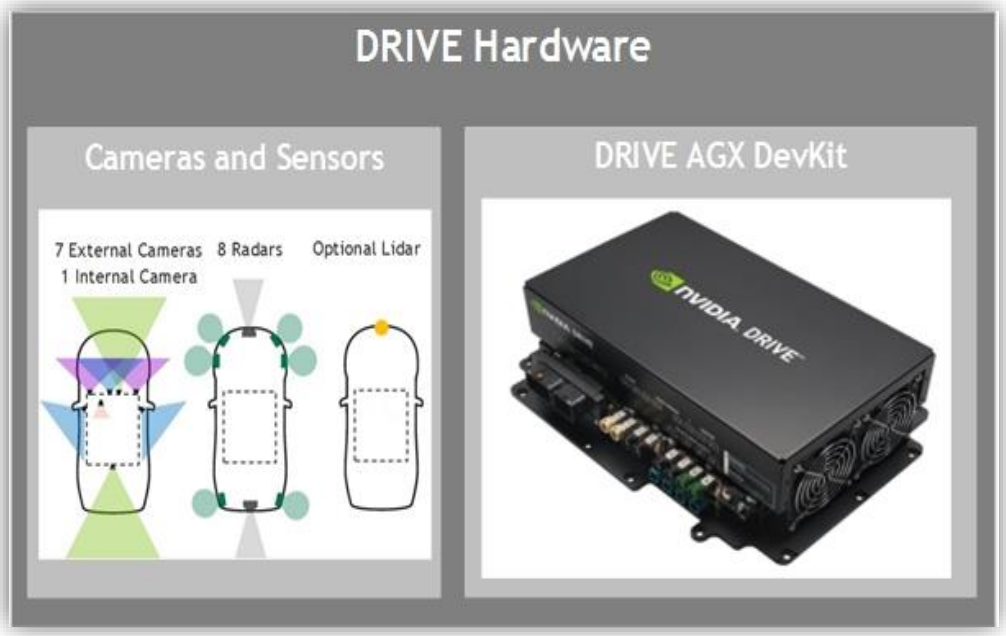


Figura 5. Nvidia AGX

Fuente: [16]

3.1.8 Asus Tinker S

El Tinker S es una plataforma de placa simple, cuenta con una distribución basada en Debian está optimizada específicamente para placas SBC (computador de placa simple). También cuenta con el soporte *"plug & play"* (conecta y reproduce) que permite un fácil acceso a unidades flash, discos duros y dispositivos de entrada y salida como cámaras requeridas para el desarrollo del proyecto. Especificaciones Asus Tinker board S:

- Procesador: 1,8 GHz de cuatro núcleos ARM Cortex-A17
- GPU 600 MHz Mali
- RAM: 2GB LPDDR3.
- Conector GPIO:
    - Proporciona 40 pines GPIO, así como 3,3 V, +5 V y GND líneas de suministro
- Almacenamiento: 16Gb eMMC y ranura para Micro SD.

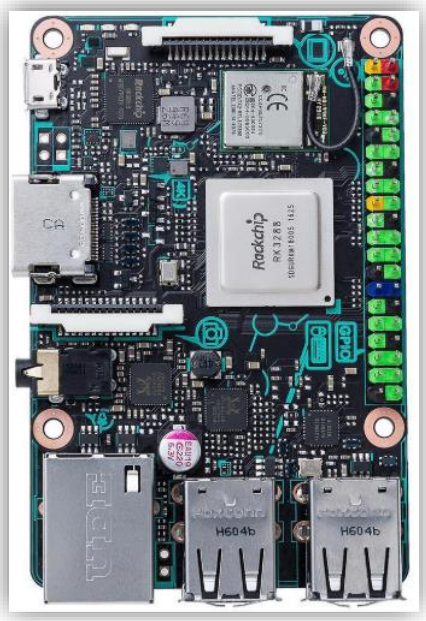


Figura 6. Tinkerboard S

Fuente: [13]

3.1.9 Cámara

Se necesita un dispositivo capaz de registrar imágenes estáticas o en movimiento, en el mercado existen varias opciones: ELP-KRL156, Sony IMX322, ELP-DL36 y Arducam 5. Debido a su característica de compatibilidad y fácil instalación, la cantidad de cuadros por segundo que puede grabar, resolución de imagen y la documentación disponible se optó por la cámara ELP-KRL156.En la figura 7, se muestra el lente de la cámara ELP- KRL156.

Algunas de sus características son:

- Conexión: Plug and Play cámara de alta resolución.
- Sensor: Omnivision OV5647 de 2 megapíxeles.
- Resolución de imagen: 1920x1080
- Máxima resolución de video: 1080p.

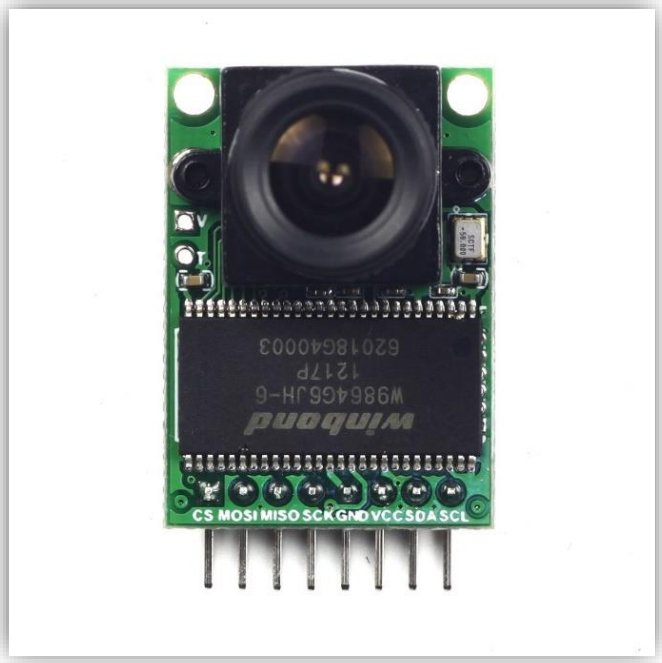


Figura 7. Cámara modelo Arducam 5

Fuente: [7]

:

3.1.10 Módulos transmisores ESP286

En muchas aplicaciones se necesita transmitir una señal de control e información para realizar una acción determinada, estos dispositivos también permiten eliminar el uso de cables. El transmisor ESP286 es un chip integrado con conexión WiFi y compatible con el protocolo TCP/IP, por lo que el objetivo principal es dar acceso a cualquier microcontrolador a una red. En la figura 8, se muestra el móduloESP286.

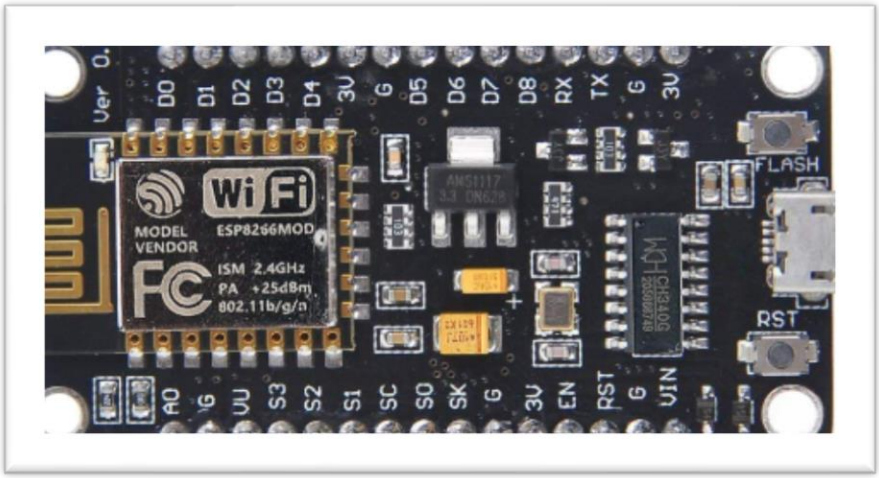


Figura 8. Modulo ESP286

Fuente: Propia

3.1.11 Arduino Micro

Este tipo de placa es un microcontrolador, el cual cuenta con 20 pines de entrada y salida digital (de los cuales 7 se pueden usar como salidas PWM y 12 como entradas analógicas), un oscilador de cristal de 16 MHz, una conexión micro USB, un encabezado ICSP y un botón de reinicio. Su forma y tamaño permite colocarlo fácilmente en cualquier lugar.

3.1.12 Relé

Este es un módulo de dos canales de 5V, puede ser controlado directamente por una amplia gama de microcontroladores como Arduino, Raspberry entre otros. Los puertos "NC" significa "normalmente conectados " y los puertos "NO" significa "normalmente abierto ". Este módulo también está equipado con 2 LED para mostrar el estado de los relés.

3.1.13 Encapsulado LM350

Una vez ya se conoce como se va a tratar la señal de control, es necesario tomar en cuenta que entre la batería del vehículo y la unidad que genera las señales de control debe de ir un regulador de voltaje, por lo que se toma en cuenta utilizar el encapsulado LM350.

Por medio del encapsulado LM350 se puede diseñar un regulador de voltaje de 12 V de entrada y 5 V de salida. Se busca un regulador que cumpla con los requerimientos para alimentar la Raspberry 3 o la Tinkerboard, es decir 5V y 3A. En la tabla 1 se muestran las características eléctricas más relevantes del encapsulado

Tabla 1 Características eléctricas del encapsulado LM350

Parámetro	Condición	Valor	Unidades
Voltaje de referencia	3V ≤ (vin − vout) ≤ 35V	1.270	V
Corriente de carga mínima	Vin − Vout = 35v	10	mA
Límite de corriente 1	Vin − Vout ≤ 10v	3.0	A
Límite de corriente 2	Vin − Vout = 30v	0.3	A
Estabilidad de la temperatura	Tmin ≤ Tj ≤ Tmax	1	%

3.2 Antecedentes y estado de la cuestión

Se investigan los antecedentes donde se desarrollaron aplicaciones similares al sistema propuesto, para comparar y facilitar el desarrollo de este. Se necesita conocer el funcionamiento, equipo, costo y técnicas utilizadas en otros diseños.

La tendencia para resolver este tipo de problemas es utilizar el procesamiento de imagen, haciendo uso de diferentes enfoques, ya sea reconocer el patrón de las luces, la forma del vehículo o las condiciones lumínicas. [3]

3.2.1 Controlador inteligente de la luz alta IHC

El concepto de este sistema se puede dividir en dos partes el sistema de detección y el control de encendido/apagado de los focos, en la figura 9 se muestra el diagrama de flujo de cómo funciona el controlador. El enfoque es desarrollar un sistema de control de bajo costo, por lo que se utiliza una foto resistencia (LDR) y un amplificador operacional como comparador, para detectar el tráfico entrante y las luces de ambiente en la carretera. [3]

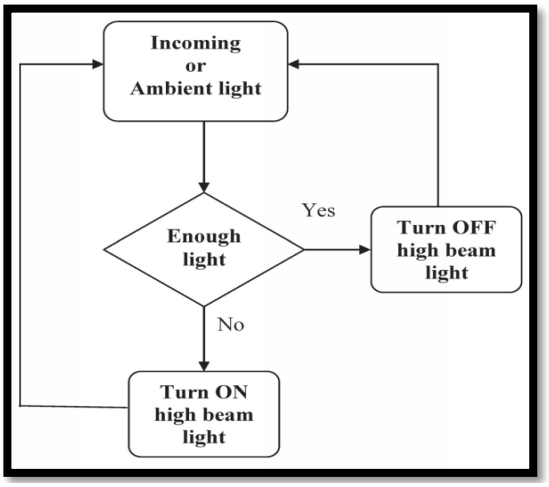


Figura 9. Diagrama de flujo controlador IHC

Fuente: [3]

3.2.2 Asistente de iluminación SmartBeam

El SmartBeam utiliza una minicámara integrada con un microprocesador dentro del espejo retrovisor, combinado con un algoritmo de toma de decisiones para operar automáticamente las luces del vehículo. Sin embargo, se advierte que no es utilizable en zonas geográficas con muchas pendientes o curvas, además depende de las condiciones del tiempo, ya que la cámara sería incapaz de captar una imagen clara, debido a la presencia de la neblina o las fuertes lluvias.

3.2.3 Focos LED adaptables

Se utilizan focos móviles los cuales por medio de un una cámara detectan la situación del tráfico durante la noche. De esta manera el sistema adapta la distribución de la luz según la situación. Es decir, cuando se detecta tráfico de frente se rota el foco de forma que el ángulo de incidencia de la luz cambie y no encandile al conductor que viene de frente.

3.2.4 Sensor de seguridad AHB

Diseñado para activarse a velocidades superiores a 25 mph, AHB cuenta con una cámara en el vehículo para ayudar a detectar el patrón que describen los faros de los vehículos que se aproximan, para cambiar automáticamente entre las luces altas y bajas en consecuencia. Este sistema utiliza procesamiento de imagen y focos fijos. [8]

Capítulo 4: Procedimiento Metodológico

En este capítulo se describen las etapas del método de diseño en ingeniería seguidas para llegar a la solución del problema. Estas etapas hacen referencia en términos generales acerca del reconocimiento y definición del problema, obtención y análisis de información, así como la evaluación de las alternativas, síntesis de una solución e implementación de la misma.

4.1 Reconocimiento y definición del problema

Se identifica un claro aumento de los accidentes durante la noche como se observa en la tabla 2, producida por el departamento de estadísticas del consejo de seguridad vial de Costa Ria (COSEVI). Durante la noche aumenta la proporción de accidentes mortales, crece de forma notable entre otras cosas porque la agudeza visual se reduce en un setenta por ciento, y el sentido de profundidad es siete veces menor, es decir se reduce la capacidad para detectar objetos y su distancia. [9]

Tabla 2 Cantidad de muertos por año según franja horaria

Costa Rica: cantidad de muertos en sitio por año según franja horaria, periodo 2012-2016

Franja horaria	Año				
	2012	2013	2014	2015	2016
00-06	81	83	102	113	129
06-12	56	59	72	100	96
12-18	93	60	78	72	102
18-24	102	96	113	140	158
Total	332	298	365	425	485

Fuente: [10] Cosevi, Área de investigación y estadística

Además, según los estudios realizados por Mace (2001) se ha encontrado que los conductores perciben el deslumbramiento producido por los faros como el principal peligro durante la conducción nocturna y una de las principales causas para abstenerse de conducir durante la noche. [11]

Es por esta razón que se plantea el desarrollo de un dispositivo para mejorar la experiencia de manejo y aumentar la seguridad de los conductores y sus acompañantes. Es importante tener en cuenta que las muertes por milla recorrida son de tres a cuatro veces más altas durante la noche que durante el día. [12]

4.2 Obtención y análisis de información

Se utilizaron los libros “Aprendizaje Automático, un enfoque práctico” y “Hands-On Machine Learning with TensorFlow” como lecturas base para el entendimiento en estas áreas, asimismo se realizó una investigación en internet sobre temas relacionados con redes neuronales, aprendizaje de transferencia, reentrenamiento de una red neuronal y visión por computador, para proveer de fundamento a la sección del marco teórico del informe. Como complemento se acudió al “Simposio Internacional sobre Aplicaciones del Aprendizaje Automático“ en el campus central del TEC por parte del grupo PARMA (Pattern Recognition and Machine Learning Group), donde se expone temas acerca del área transdisciplinar del aprendizaje automático, comprende paradigmas de alto interés tanto académico como industrial, como pueden ser reconocimiento de patrones, inteligencia artificial, recuperación y análisis de información, temas afines al proyecto a desarrollar.

Se entrevistó al ingeniero y profesor del Instituto Tecnológico de Costa Rica, Rogelio Gonzalez, acerca de un proyecto con una solución similar en el campo de la visión por computador y aprendizaje automático, como método de análisis para evaluar la información obtenida.

Además, el ingeniero proporcionó información acerca del proceso y experiencia obtenida en el desarrollo de un proyecto de este tipo, por lo que se utilizó como referencia para seguir con el planteamiento de una posible solución.

4.3 Evaluación de las alternativas y síntesis de una solución

Se planteó una solución conformada en forma general por tres partes acorde con los objetivos específicos, cada etapa está conformada por una fase de desarrollo y otra de evaluación y correcciones de errores. Se realizó una primera descripción del software y el hardware a utilizar en el prototipo meta. La primera tarea fue desarrollar una red neuronal que detecte un vehículo durante la noche, posteriormente se desarrolló un algoritmo que junto con la red neuronal reconozca de forma inteligente y en tiempo real los vehículos, y por último se implementó el algoritmo en un computador versátil y de bajo costo que se pueda conectar a los focos delanteros de un automóvil.

Se debe elegir entre diferentes tecnologías para el desarrollo del prototipo, por medio de diferentes criterios técnicos, ya sea para el desarrollo del algoritmo, como para la elección correcta del hardware que mejor cumpla con los requisitos y genere los mejores resultados.

A partir de la investigación realizada y según la sección 1.3 se pretende entrenar una red neuronal capaz de identificar cuando un vehículo se acerca, esto es posible realizarlo por medio de dos bibliotecas de código abierto para el aprendizaje automático, TensorFlow y YOLO (*You Only Look Once*). La cantidad y calidad de la documentación disponible y la existencia de otros proyectos desarrollados con TensorFlow, son las razones por las cuales se eligió utilizar esta librería. [5]

Para la segunda fase de desarrollo, según la entrevista realizada al ingeniero Rogelio Gonzalez, este recomendó utilizar la librería OpenCV para visión por computador, debido a que es de código libre, ampliamente documentada y existe una gran cantidad de proyectos desarrollados con la misma. De la misma forma, la experiencia adquirida anteriormente en cursos de la carrera con el uso de esta biblioteca serían los criterios utilizados para llevar acabo la segunda etapa.

Para el tercer objetivo se debe de elegir el uso de un computador de placa simple y una cámara que sean de bajo costo, pero que pueda cumplir con un rendimiento aceptable en aplicaciones de reconocimiento en tiempo real. Para cumplir con el requisito de bajo costo, se presentan tres opciones para el microprocesador; Asus TinkerBoard, Raspberry PI y Arduino, siendo la opción más económica el Arduino, seguido por la Raspberry y por último el TinkerBoard. Según la sección 3.5 del marco teórico, se conocen las especificaciones de los microcontroladores, de acuerdo con sus prestaciones mayor capacidad de procesamiento, cantidad de memoria RAM y disponibilidad de una tarjeta de video integrada se elige la Asus Tinker.

El estándar en la empresa automotriz es el uso de baterías de doce voltios por lo que el sistema planteado se debe alimentar a una batería de este tipo. Por esta razón se necesita un módulo capaz de utilizar este tipo de baterías y alimentar el hardware utilizado, por lo que se plantea el desarrollo de un regulador de voltaje que utilice una entrada de doce voltios y tenga una salida hasta de tres amperios.

Se debe implementar un circuito regulador de tensión capaz de mantener constante el valor de voltaje de salida en 5 V, en un rango válido de operación (de 8V a 40V), por medio de la configuración de componentes pasivos como resistencias y capacitores, además del uso del encapsulado LM350. [14]

De la misma manera se puede optar por un módulo ya desarrollado, impreso y confiable, que cumpla esta función como lo es el regulador LTC3780.

Debido a la facilidad de implementación de un regulador de voltaje y a que el regulador propuesto equivale a una quinta parte del costo del regulador LTC3780, se escogió implementar el regulador de voltaje y no la compra de uno.

4.4 Implementación de la solución

Para realizar la implementación de la solución se subdivide este apartado en etapas, como se muestra en la figura 10, cada etapa describe las actividades realizadas para llevar a cabo los objetivos específicos planteados.

Diagrama de bloques para la implementación de la solución

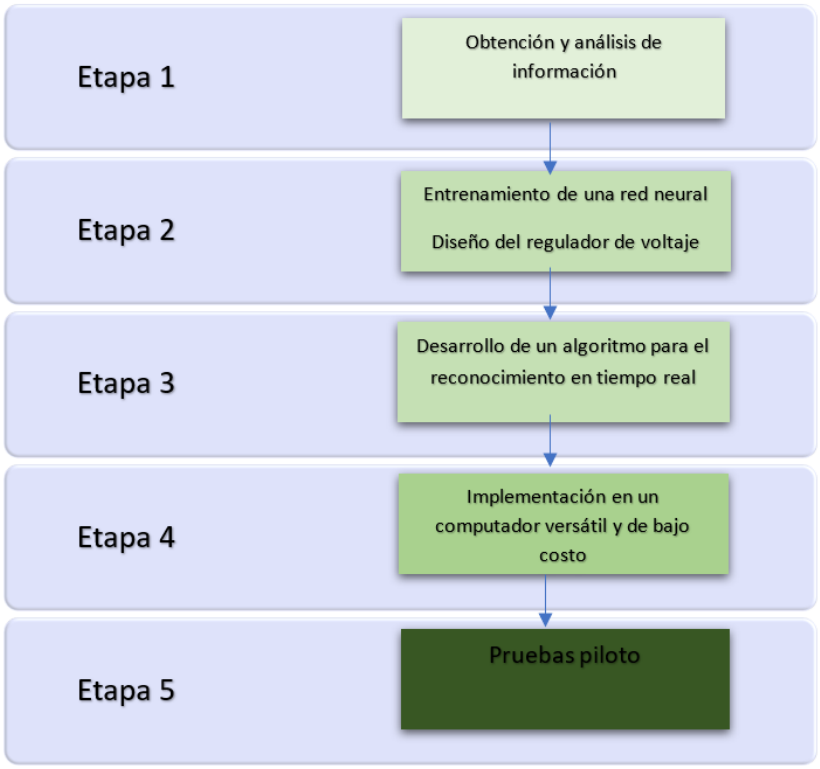


Figura 10. Diagrama de bloques de la solución

Fuente: Propia

4.4.1 Etapa 1

En esta etapa se propone conocer los conceptos y técnicas más relevantes acerca del aprendizaje automático y la visión por computador. Para esta fase de la solución se realizó las siguientes actividades:

1. Lectura de apartados relevantes de los libros “Aprendizaje Automático, un enfoque práctico” y “Hands-On Machine Learning with TensorFlow”, además de reportes relacionados con la visión por computador.
2. Realizar el curso online aprendizaje automático para principiantes.
3. Estudiar el uso de las bibliotecas de aprendizaje automático TensorFlow y visión por computador OpenCV.
4. Realizar de taller exploratorio “TensorFlow for Poets” conociendo acerca del entrenamiento de redes neuronales

4.4.2 Etapa 2

En esta etapa corresponde poner en práctica los conceptos y técnicas aprendidas para poder entrenar una red neuronal que detecte un vehículo durante la noche, para que sea posible se debe de cumplir las siguientes tareas:

1. Recopilar imágenes de vehículos durante la noche, que cumplan con los parámetros adecuados de resolución, tamaño y perspectiva para poder elaborar un set de datos confiable.
2. Instalar el ambiente de programación para el desarrollo de aplicaciones con TensorFlow.
    a. Sistema operativo Ubuntu/Linux.
    b. Paquete para la computación en paralelo CUDA (Arquitectura Unificada de Dispositivos de Cómputo) de Nvidia.
    c. Interfaz de programación de aplicaciones para construir y ejecutar gráficos de TensorFlow.

3. Realizar el proceso de entrenamiento de la red neuronal, es decir:
    a. Modificación de parámetros (orientación, posición, color, saturación).
    b. Etiquetado de imágenes (alrededor de 200), generación de archivos descriptivos (.xml).
    c. Generación de registros de tensores (tfrecords).
    d. Creación de archivo de configuración para la red.
    e. Definir el número de pasos del entrenamiento.
4. Realizar el número de iteraciones necesarias para obtener valores aceptables de precisión de entrenamiento, precisión de la validación y entropía cruzada.
5. Ajustar y corregir errores.

4.4.3 Etapa 3

Para el desarrollo del algoritmo que incorpora la red neuronal que realiza el reconocimiento inteligente y en tiempo real de otros vehículos, se necesita cumplir con los siguientes puntos:

1. Instalar el ambiente de programación para trabajar con OpenCV.
2. Desarrollar el algoritmo para el reconocimiento en tiempo real.
3. Programar la rutina para incorporar la red neuronal al algoritmo de reconocimiento.
4. Crear una rutina para el análisis de video, facilitando la evaluación y análisis del funcionamiento del algoritmo.
5. Probar y depurar errores.

4.4.4 Etapa 4

Esta etapa consiste en implementar el algoritmo ya desarrollado en un computador versátil y de bajo costo que se pueda conectar a los focos de un vehículo, para ello se necesita:

1. Instalar el ambiente de desarrollo para utilizar TensorFlow y OpenCV
2. Instalar al interfaz de programación de TensorFlow para dispositivos móviles.
3. Instalar las librerías de OpenCV
4. Implementar el algoritmo desarrollado anteriormente que cuenta con la red neuronal y la detección en tiempo real.
5. Conectar el prototipo desarrollado a un faro delantero de prueba.

4.4.5 Etapa 5

En esta etapa se propone hacer una serie de pruebas en el prototipo desarrollado para poder obtener información que pueda ser analizada para evaluar su funcionamiento.

1. Verificar el etiquetado correcto de una serie de imágenes de vehículos en la noche.
2. Grabar distintos videos en diferentes vías para posterior análisis.
3. Analizar los videos con el algoritmo de análisis de video y reconocimiento.
4. Probar el cambio de luz con el algoritmo de análisis de video.
5. Probar en tiempo real el cambio correcto de los focos.

4.5 Reevaluación y rediseño

El sistema para desarrollar se plantea hasta una fase de prototipo, por lo que se pretende evaluar su funcionamiento en este nivel, obteniendo los datos de su rendimiento para analizar posibles mejoras.

Se debe tomar en cuenta que el prototipo del proyecto es capaz de hacer el reconocimiento de un vehículo en la noche, generando correctamente las señales de control para cambiar el tipo de luz de unos focos delanteros de prueba según se requiera. Además, se alimenta de una batería de automóvil y soporta las variaciones de voltaje producidas en los vehículos.

Sin embargo, no es una implementación final que se instale en un auto, debido a que se debe crear un sistema que interactúe con la computadora o mecanismo ya presente en el vehículo encargado de realizar el cambio de luz.

El área del aprendizaje automático avanza cada vez más, por lo que posibles progresos en las técnicas de reconocimiento de patrones pueden ser implementados en el sistema.

Capítulo 5: Explicación detallada de la solución

En el apartado 4.3 “evaluación de las alternativas para la solución” se escogió entre las opciones cuales componentes de software y hardware cumplían los requisitos de costo y desempeño, por lo que en esta sección se plantean una serie de propuestas alrededor de la configuración de estos componentes, sus conexiones y las funciones que cada uno desempeña

5.1 Análisis de las soluciones y selección final

A continuación, se presentan y analizan las diferentes soluciones planteadas, sus ventajas y desventajas, así como las razones por las que fueron descartadas. Además de los criterios que llevaron a la selección de la solución final.

En cada una de las alternativas se desarrollará un software capaz de detectar los vehículos durante la noche, haciendo uso de las librerías de TensorFlow y OpenCV debido a las ventajas mencionadas en la sección 4.3 “evaluación de las alternativas y síntesis de una solución’, por lo que los criterios para seleccionar una de las soluciones se basan en términos de desempeño y costo.

Además, cada propuesta tendrá en común una unidad que funcionará como acople entre la alimentación del automotor (batería 12V) y el sistema propuesto. Se utilizarán componentes de bajo costo como resistencias, capacitores y el encapsulado para LM350 para la implementación del regulador de voltaje.

5.1.1 Solución I

La propuesta consta de dos encapsulados, el primero se encarga del reconocimiento de las luces frontales y traseras de un vehículo, posteriormente este encapsulado transmitirá la señal de control al segundo encapsulado, donde este se encargará de la adquisición del dato y el cambio de luz correspondiente, como se muestra en la figura 11.

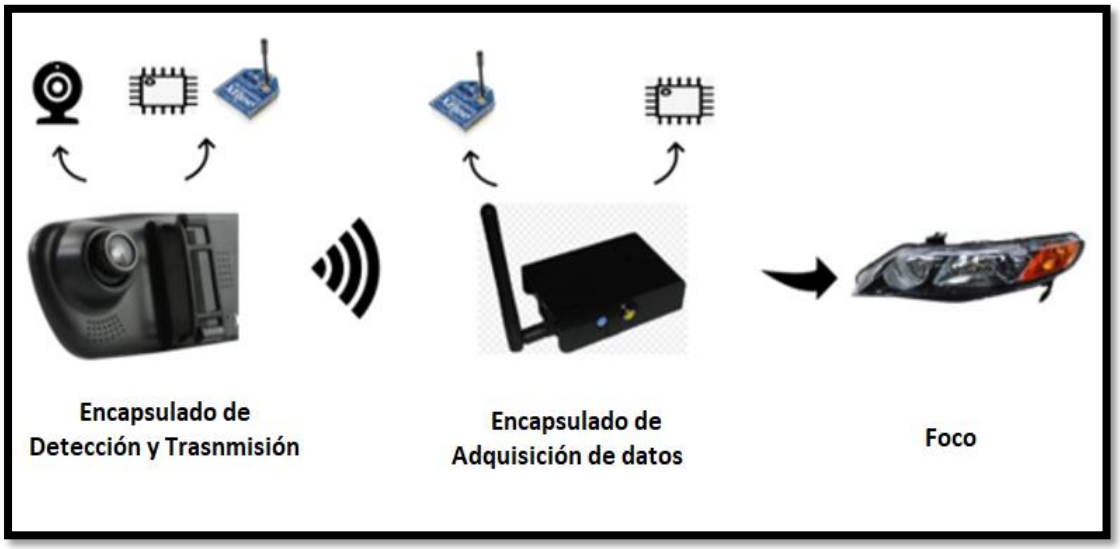


Figura 11. Solución Propuesta I

Fuente: Propia

El encapsulado del reconocimiento contará con una cámara, un microcontrolador Raspberry Pi y un módulo transmisor Esp8266 que funcionará como coordinador de la red punto a punto. Se podría ubicar en el retrovisor del vehículo, por lo que se necesitaría de un encapsulado capaz de transmitir una señal de control cuando se detecta un vehículo, y de otro módulo que controle los focos, capaz de recibir la señal y hacer el cambio de las luces. El módulo de adquisición de datos contará con un microcontrolador Arduino y también con un módulo de recepción de datos Esp8266.

Este tipo de solución cuenta con la ventaja de aislar el módulo de reconocimiento de la humedad, el calor y la vibración, ya que se podría instalar dentro de la cabina del vehículo, además de eliminar el uso excesivo de cables para la interconexión de los módulos, asimismo cuenta con la ventaja de poseer dos microcontroladores para dividir el trabajo en tareas más simples.

Sin embargo, con un mejor posicionamiento del módulo de reconocimiento se puede eliminar el uso de un microcontrolador y los submódulos de transmisión y recepción, por lo que el costo general de proyecto disminuye considerablemente, razón por la que se descartó esta alternativa de solución.

5.1.2 Solución II

Esta alternativa cuenta únicamente con un módulo, en lugar de dos como la propuesta anterior, esto se puede si el encapsulado de reconocimiento se ubica cerca de los faros delanteros, por lo que elimina la necesidad de contar con un encapsulado para la recepción de datos.

Figura 12. Propuesta Solución II

Fuente: Propia

Esta solución se plantea como el desarrollo de un prototipo que se encargue del reconocimiento de un automotor, contará con un computador de placa simple y una cámara, capaz de conectarse a los faros delanteros de un vehículo para cambiar la luz según corresponda. El prototipo se podrá conectar a una batería de 12 V, debido a que es el estándar en la mayoría de los vehículos.

De esta manera el costo total del hardware que se necesita para resolver el problema disminuye considerablemente, ya que se elimina un módulo innecesario, sin afectar el rendimiento general del sistema y cumplir con el objetivo final.

5.1.3 Solución III

Como alternativa se plantea utilizar una tarjeta de desarrollo más robusta que cuente con la capacidad de procesamiento paralelo como la Nvidia Drive, la cual se utiliza de forma frecuente en vehículos autónomos y aplicaciones en inteligencia artificial

Se usaría el mismo esquema de solución anterior, es decir, un dispositivo que se encargue del reconocimiento de un vehículo, el cual contará esta vez con la tarjeta de desarrollo Nvidia drive y una cámara. Este dispositivo se podrá conectar a los faros delanteros de un vehículo, para que una vez haga la detección cambie la luz en consecuencia, de la misma forma el prototipo se alimentara de una batería de 12 V.

Sin embargo, debido al alto costo de esta tarjeta y a la dificultad para obtener el componente se descarta la opción.

5.2 Descripción del hardware

Se describe la estructura y funcionamiento de cada uno de los módulos que conforman el sistema desarrollado para resolver el problema planteado.

Por consiguiente, se divide el sistema en las unidades funcionales que lo componen, incluyendo una descripción en la forma de implementar cada etapa del hardware.

El sistema se subdivide en dos unidades principales, la unidad de alimentación y la unidad de reconocimiento de vehículos, la cual se encarga de la recolección, procesamiento y análisis de video.

En la figura 13 y 14 se muestra la estructura de las unidades del sistema, es decir el hardware, utilizado sus conexiones, entradas y salidas.

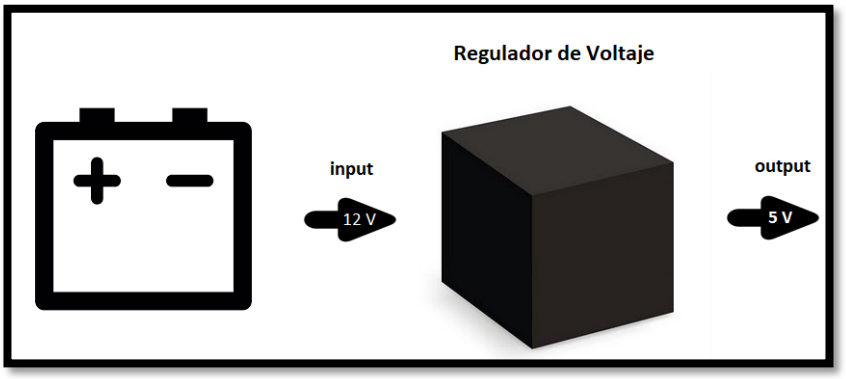


Figura 13. Regulador de voltaje

Fuente: Propia

La primera unidad se conecta a una batería de doce voltios, en donde esta mantiene constante una salida de cinco voltios y hasta tres amperios para garantizar el correcto funcionamiento de la unidad siguiente.

La segunda unidad se conecta al regulador de cinco voltios y tiene como entrada de datos una cámara de video, la salida de este módulo es una señal de control, la cual se conecta a un relé conectado a un foco delantero de prueba.

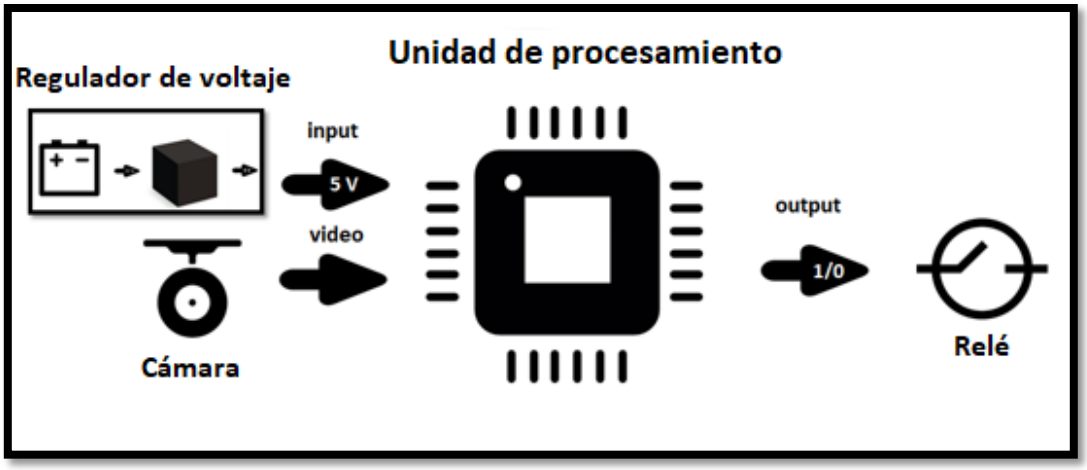


Figura 14. Unidad de procesamiento

Fuente: Propia

En seguida se hace una descripción detallada de cada una de las unidades funcionales del dispositivo.

5.2.1 Unidad de Alimentación

Se decidió implementar un circuito por medio del uso del encapsulado LM350, debido a que la alimentación de la mayoría de los vehículos cuenta con una batería de doce voltios, por lo que se debe obtener 5V y una corriente de hasta 3A para energizar la TinkerBoard según el fabricante. [13]

La unidad de alimentación permite obtener según la configuración de las resistencias del circuito un voltaje de salida constante y un rango de corriente de salida de hasta tres amperios. Según la hoja de datos si el regulador se encuentra a más de veinte centímetros de la batería o fuente de alimentación se debe utilizar un capacitor en la entrada del circuito. [14]

De la misma forma se utiliza un capacitor en la salida del regulador para obtener una mejor respuesta transitoria. En la figura 15, se puede observar el circuito propuesto para obtener el voltaje de salida deseado de cinco voltios, el valor de las resistencias “R1” es de 240 Ω y para la resistencia” R2” se utiliza un potenciómetro, ya que permite ajustar el divisor de voltaje para obtener un valor más preciso en el voltaje de salida.

Según el fabricante se debe utilizar un capacitor de 47 µF en paralelo a las resistencias “R1” y “R2” para obtener una mejor respuesta transitoria. [14]

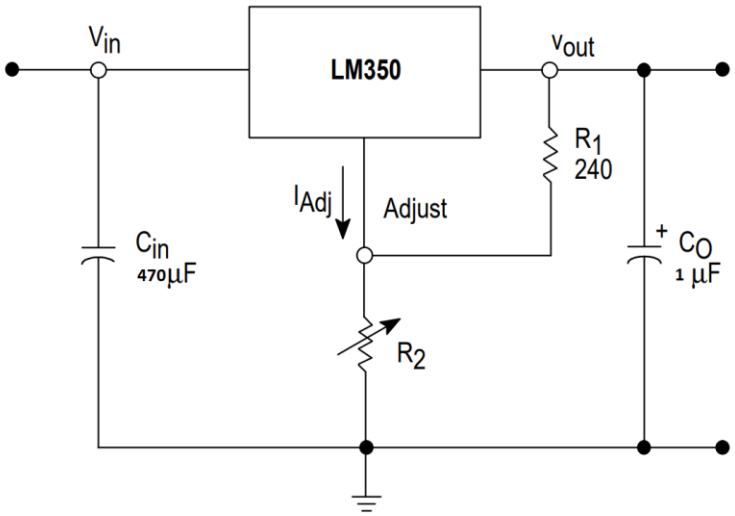


Figura 15. Circuito regulador de voltaje implementado

Fuente: [14]

En el apéndice A, se documenta la imagen real del regulador de voltaje implementado.

5.2.2 Unidad de reconocimiento de vehículos

Para poder hacer el análisis de video, se utiliza como unidad de procesamiento la Tinkerboard. Se conecta una cámara en uno de los puertos USB disponibles. En la figura 16, se muestran los pines de entrada y salida de la Tinkerboard.

Una vez que el algoritmo determina o no la cercanía de un vehículo, la Tinkerboard enciende o apaga una de las salidas digitales disponibles (en este caso los pines 13 y 15).

Los pines 13 y 15 se conectan a los canales de un relé, para accionar las luces del faro. De la misma forma se alimenta el relé por medio de los pines de alimentación de 5 voltios y “GND” (neutro).

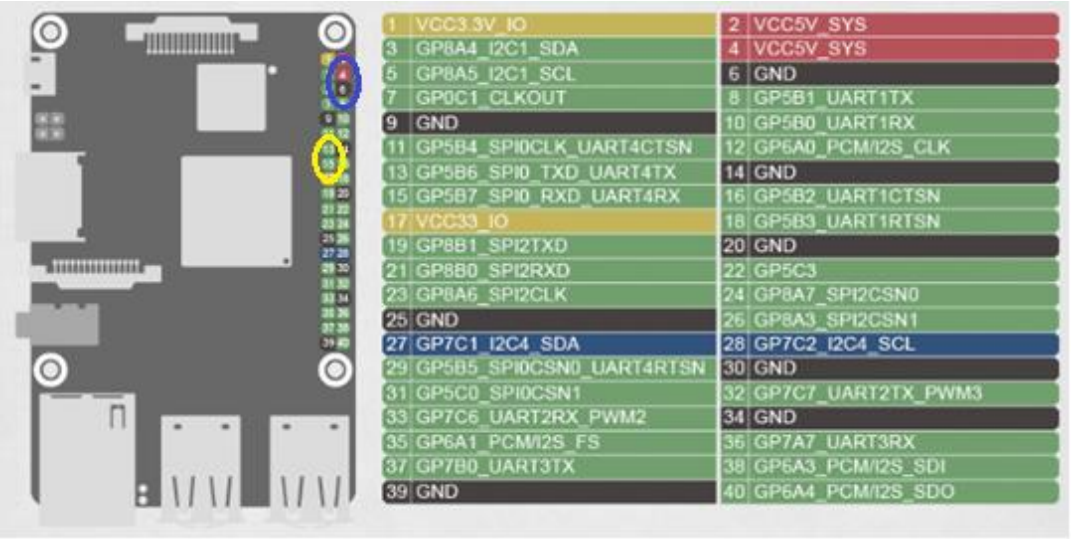


Figura 16. Pines Tinkerboard S

Fuente: [13]

Es necesario conocer la forma como se conectan las unidades funcionales del prototipo. En la figura 17, se muestran cuales pines son los que se interconectan entre los componentes. En el apéndice B, se documenta la foto real del prototipo.

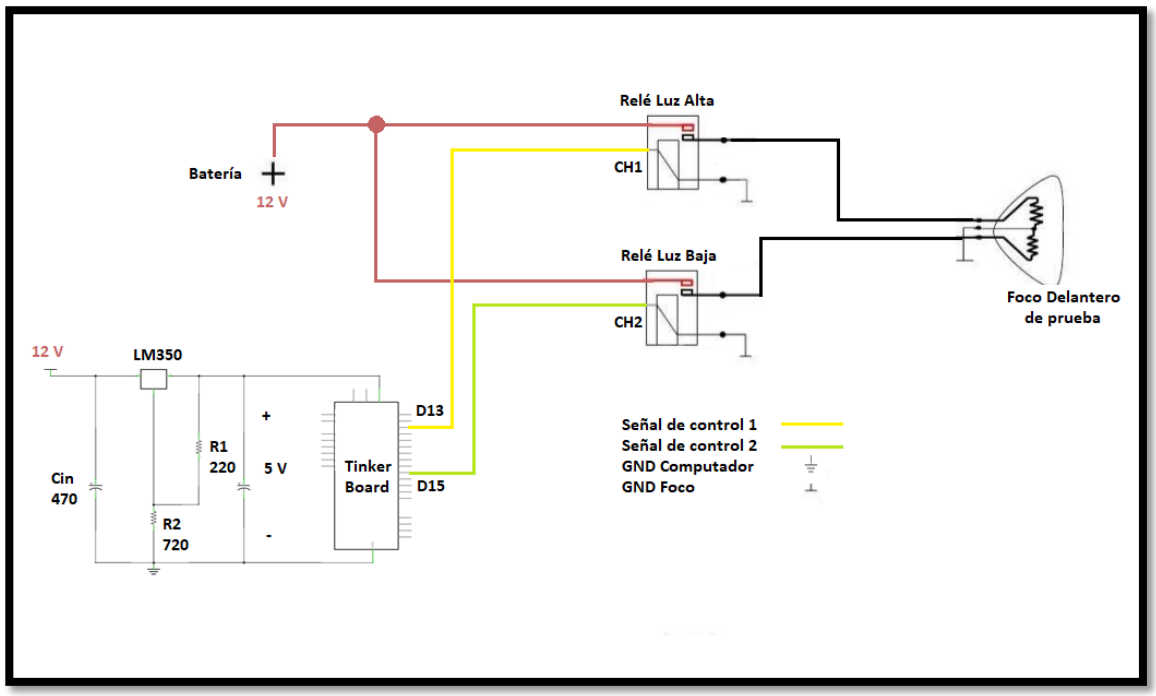


Figura 17. Esquemático del prototipo

Fuente: Propia

5.2.1 Hardware utilizado para el entrenamiento

Es importante mencionar que para el entrenamiento de la red neuronal se utilizó una computadora con una tarjeta GPU (unidad de procesamiento gráfico) GTX 1070, procesador Intel i7 6700 HQ y 16 gigabytes de memoria RAM para obtener tiempos de entrenamiento más cortos.

Se utilizó la laptop mencionada para realizar las pruebas iniciales, donde se comprobó el correcto funcionamiento del algoritmo que incorpora la red neuronal y que además es capaz de realizar la detección de vehículos en tiempo real o por medio de video.

Asimismo, se conectó con un Arduino Micro por medio de los puertos USB (laptop) y micro USB (Arduino) permitiendo obtener pines digitales de salida, con los cuales se puede transmitir la señal de control por medio de comunicación serial entre la computadora y el microcontrolador. En la figura 18 se muestra una representación del sistema descrito.

La señal de control se transmite a un relé electrónico, el cual da paso o no de corriente entre el foco y la batería, para activar la luz corta o larga del reflector, según lo determine el algoritmo de detección inteligente de vehículos.

El microcontrolador y el relé se conectan por medio del cable hembra – hembra entre dos de los pines digitales del Arduino y las entradas “CH1” y “CH2” del relé, además se enlazan los pines de alimentación de cinco voltios y neutro.

En la figura 19, se muestra de forma específica como se conecta el hardware que conforma el equipo de entrenamiento y pruebas.

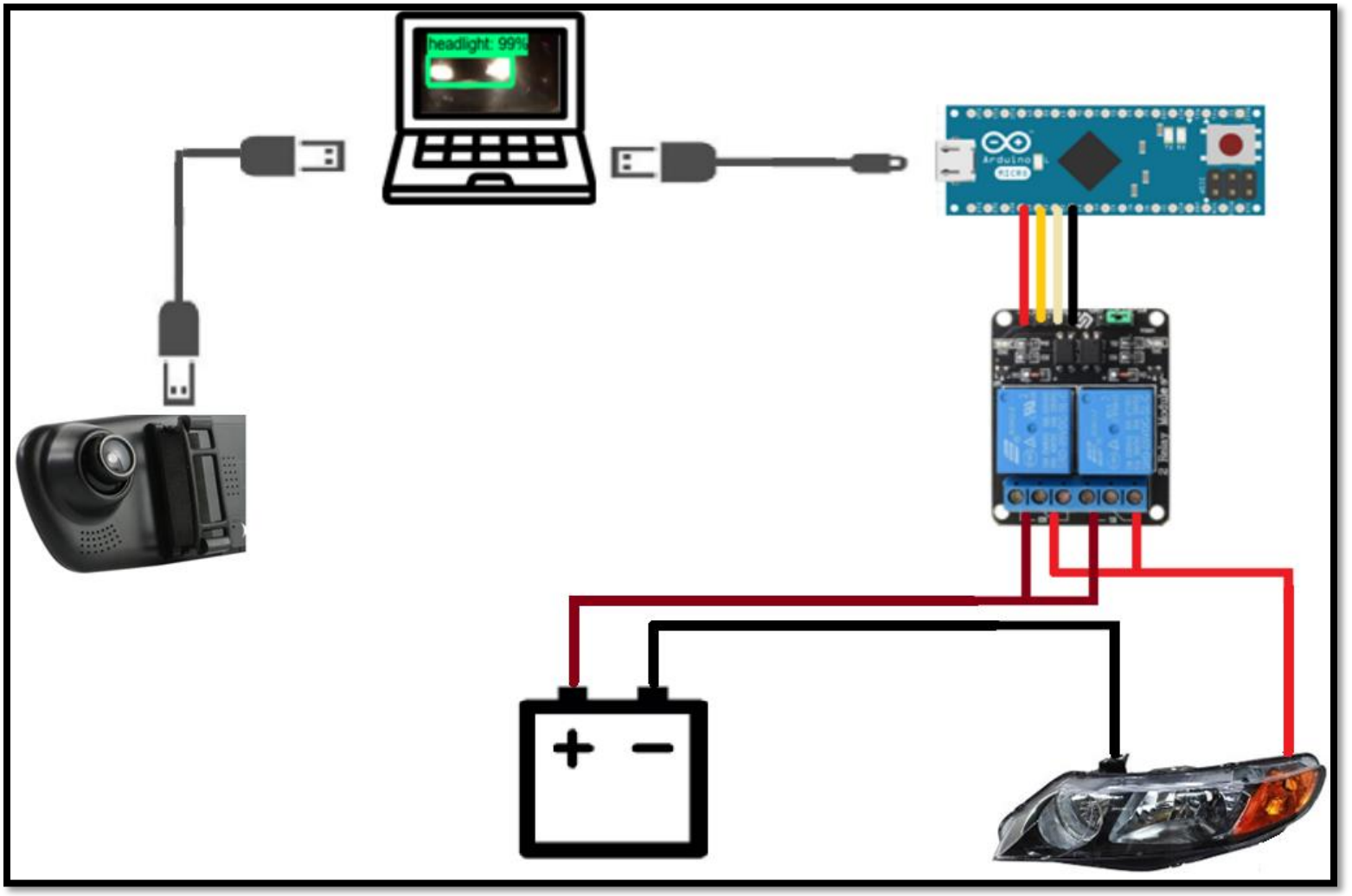


Figura 18. Equipo de entrenamiento y pruebas

Fuente: Propia

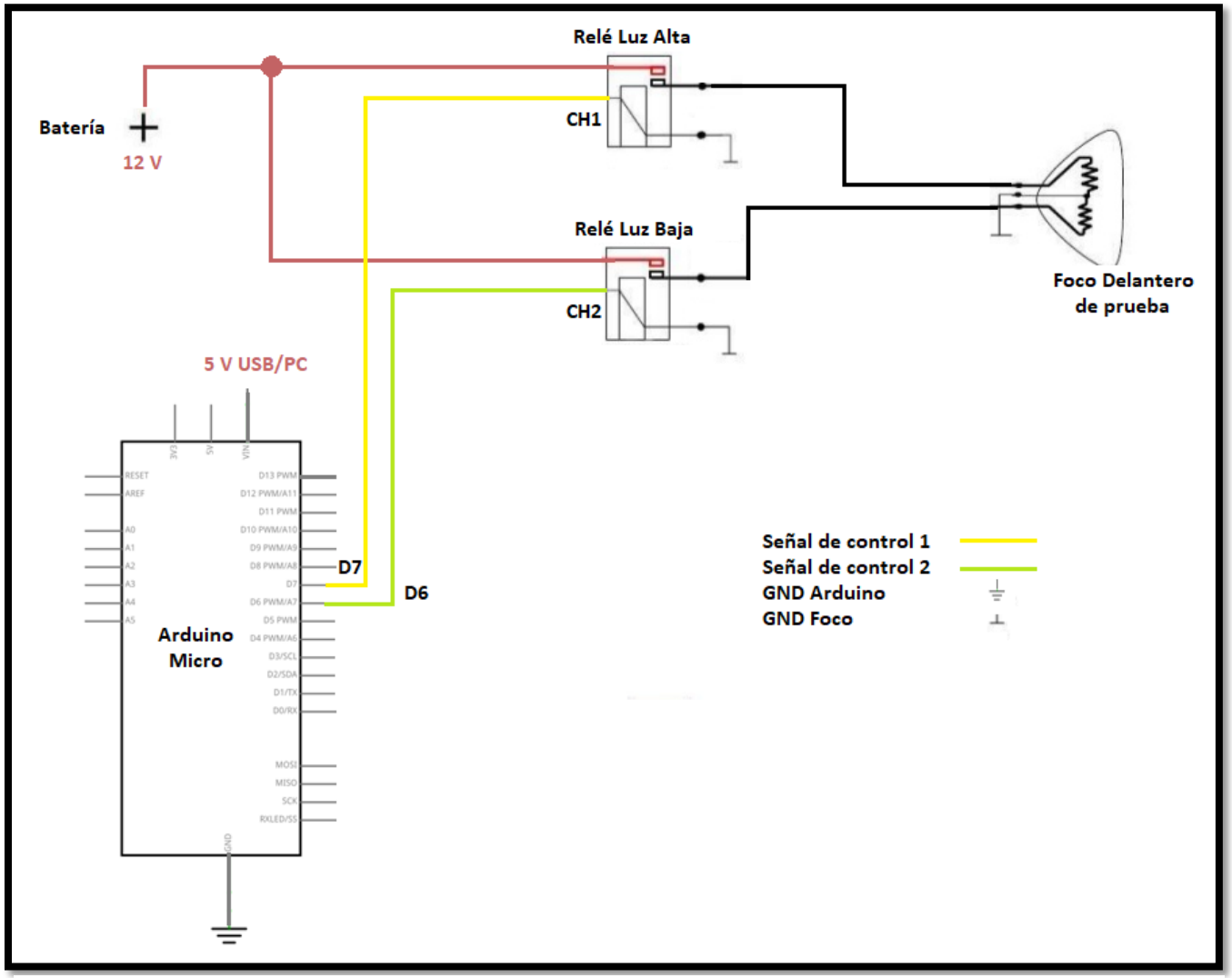


Figura 19. Esquemático del hardware de pruebas

Fuente: Propia

5.3 Descripción del software

En esta sección se describe el software utilizado para desarrollar y manejar el sistema, por lo tanto, se indican las principales rutinas realizadas para llegar a la solución del problema.

Para describir el software utilizado, primero se debe mencionar cómo es el entrenamiento de una red neuronal, este es un proceso de prueba y error, en donde se necesita proporcionar una serie de condiciones y modificar ciertos parámetros dependiendo de la aplicación que se le vaya a dar. Y de esta manera obtener un resultado que cumpla con el objetivo para el que fue ideada la red neuronal.

Para entrenar una red neuronal correctamente, primero se debe crear un conjunto de datos, en este caso imágenes en donde se muestre el contexto y objeto que se quiere identificar, además de su correspondiente etiqueta. Cada imagen debe de cumplir con unos requisitos para que puedan ser utilizadas y no creen incertidumbre en el producto final, entre estos parámetros se encuentra, la resolución, tamaño, orientación y color de imagen.

Se estableció como parámetro un tamaño de imagen de 240 x 200, para poder ajustar el tamaño de todas las imágenes recolectadas automáticamente, se creó una rutina con OpenCV como se muestra a en la figura 20.

```
with open('imagen_name', 'r+b') as f:
    with Image.open(f) as image:
        cover = resizeimage.resize_cover(image, [240, 200])
        cover.save('i.jpeg', image.format)
```

Figura 20. Rutina para ajuste del tamaño

Fuente: Propia

Se necesita etiquetar cada una de las imágenes con su respectiva clase, altura, ancho, coordenadas “x” y “y” donde se encuentra el objeto de interés en la imagen, para esto se hace uso de un script de fuente abierta llamado labelimg.py, el cual genera un archivo “.xml” (lengua de etiquetado) con toda la información necesaria.

Figura 21. Etiquetado de imágenes con ayuda de Labelimg

Fuente: Propia

Una vez se obtienen los archivos “.xml” se necesitan convertir a formato “.csv” (valores separados por coma) para poder crear registros de TensorFlow con la información de cada imagen y empezar el entrenamiento. La rutina que hace la conversión de formato se muestra en la figura 22.

```
def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     member[0].text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
                     int(member[4][3].text)
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df
```

Figura 22. Rutina xml – cvs

Fuente: Propia

Es necesario hacer una manipulación previa de las imágenes mediante código para poder obtener un formato de archivo con el cual poder iniciar el entrenamiento.

Se recomienda hacer uso de la transferencia de conocimiento donde se aprovecha el hecho que la red ya fue creada, por lo tanto, se planea utilizar como base una red neuronal entrenada para reconocer objetos comunes, existen dos posibilidades “SSD-Mobilnet” e “Inception v2”.

La red neuronal se origina de una de estas por medio del uso de la API de TensorFlow, la cual consiste en un conjunto de rutinas, estas permiten reentrenar una red a partir de la creación de un archivo de configuración para la red nueva, donde se indica la dirección de las carpetas donde se ubican las imágenes, el nuevo número de clases y dirección de los registros del nuevo conjunto de datos.

A continuación, se muestra en la figura 23 un ejemplo como se utiliza la API de TensorFlow.

```
./train \
    --logtostderr \
    --train_dir=path/to/train_dir \
    --model_config_path=model_config.pbtxt \
    --train_config_path=train_config.pbtxt \
    --input_config_path=train_input_config.pbtxt
```

Figura 23. Uso API Tensorflow

Fuente: Propia

También se pueden variar algunos parámetros como el brillo (--random_brightness), escalado (--random_scale), numero de pasos de entrenamiento (--training_steps), cambio de orientación aleatoria (--flip_left_right), todos estos parámetros se cambian para obtener un mejor o peor proceso de entrenamiento. Este proceso se repite una y otra vez para un mismo conjunto de datos hasta alcanzar el resultado deseado.

Una vez terminado el proceso de entrenamiento se obtiene la red neuronal en un archivo ".pb", se procede a implementar la red neuronal en un código en Python, el cual pueda hacer el reconocimiento de un vehículo en tiempo real o mediante el análisis de video.

La rutina que se muestra en la figura 24 incorpora el modelo de la red neuronal entrenada. Para mejorar el rendimiento del código se carga el modelo una única vez en la memoria al inicio del programa, para más tarde poder obtener los valores de las predicciones.

```
# carga el modelo en memoria
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

    sess = tf.Session(graph=detection_graph)
```

Figura 24. Rutina para cargar el modelo en memoria

Fuente: Propia

En la figura 25, se muestra la función que realiza la detección actual corriendo el modelo con el video como entrada, además se encierra en un recuadro el objeto, muestra la clase y el valor del porcentaje de similitud.

Se puede variar cuando imprimir en pantalla la detección, bajando el valor de “min_score_thresh”, como se observa en la figura 25 es de 0.8, es decir solo las predicciones con una similitud mayor al 80 % se muestran en pantalla.

```
# realiza la detección actual corriendo el modelo con el video como entrada
(boxes, scores, classes, num) = sess.run(
    [detection_boxes, detection_scores, detection_classes, num_detections],
    feed_dict={image_tensor: frame_expanded})

# encierra en un recuadro el objeto, muestra la clase y valor de similitud
vis_util.visualize_boxes_and_labels_on_image_array(
    frame,
    np.squeeze(boxes),
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates=True,
    line_thickness=8,
    min_score_thresh=0.8)
```

Figura 25. Código para impresión de los valores en pantalla.

Fuente: Propia

La forma de generar la señal de control cambia según el equipo que se utiliza, es decir, si se utiliza la Tinkerboard únicamente se deben configurar dos pines como salidas digitales y conectarlo a las dos entradas del relé. Por otro lado, si se utiliza la computadora se debe hacer una comunicación serial con el Arduino Micro que está conectado al relé.

Sin importar el equipo, ambas rutinas siguen la misma lógica, ya que primero se determina si se identifica un vehículo en un 70%, si es así, se genera una señal de control, para que mediante el relé se cierre el circuito de la luz corta y abra el circuito de la luz larga, es decir bajar la luz, si el porcentaje de acierto es menor al 70%, se genera otra señal de control, pero para hacer el caso contrario.

Antes de cambiar de intensidad de luz, se creó una función de suavizado o filtrado para evitar que se den cambios bruscos entre un tipo de luz y otro, por lo que se crea una lista, donde solo se cambia a luz larga, si en diez predicciones no se ha detectado un automotor.

```
# generacion de la senales de control
if scores[0][0] > 0.7:
    print('Vehiculo')
    l.append(0)
    l.popleft()
    print(l)
    ser.write('a'.encode('utf-8'))

else:
    print('Scanning')
    l.append(1)
    l.popleft()
    print(l)
    cnt = Counter(l)
    sc = cnt[1]
    if sc == 6:
        ser.write('b.'.encode('utf-8'))
        print('Luz larga')
```

Figura 26. Rutina de Filtrado y Suavizado

Fuente: Propia

También se implementó una subrutina para obtener la cantidad de cuadros por segundo (FPS) a los que se analiza video, como medida para comprobar el rendimiento general del sistema. Los FPS son una medida de la velocidad a la cual un dispositivo puede mostrar imágenes ya procesadas.

En la figura 27, se muestra el diagrama de flujo que sigue la lógica general de cómo se maneja el sistema.

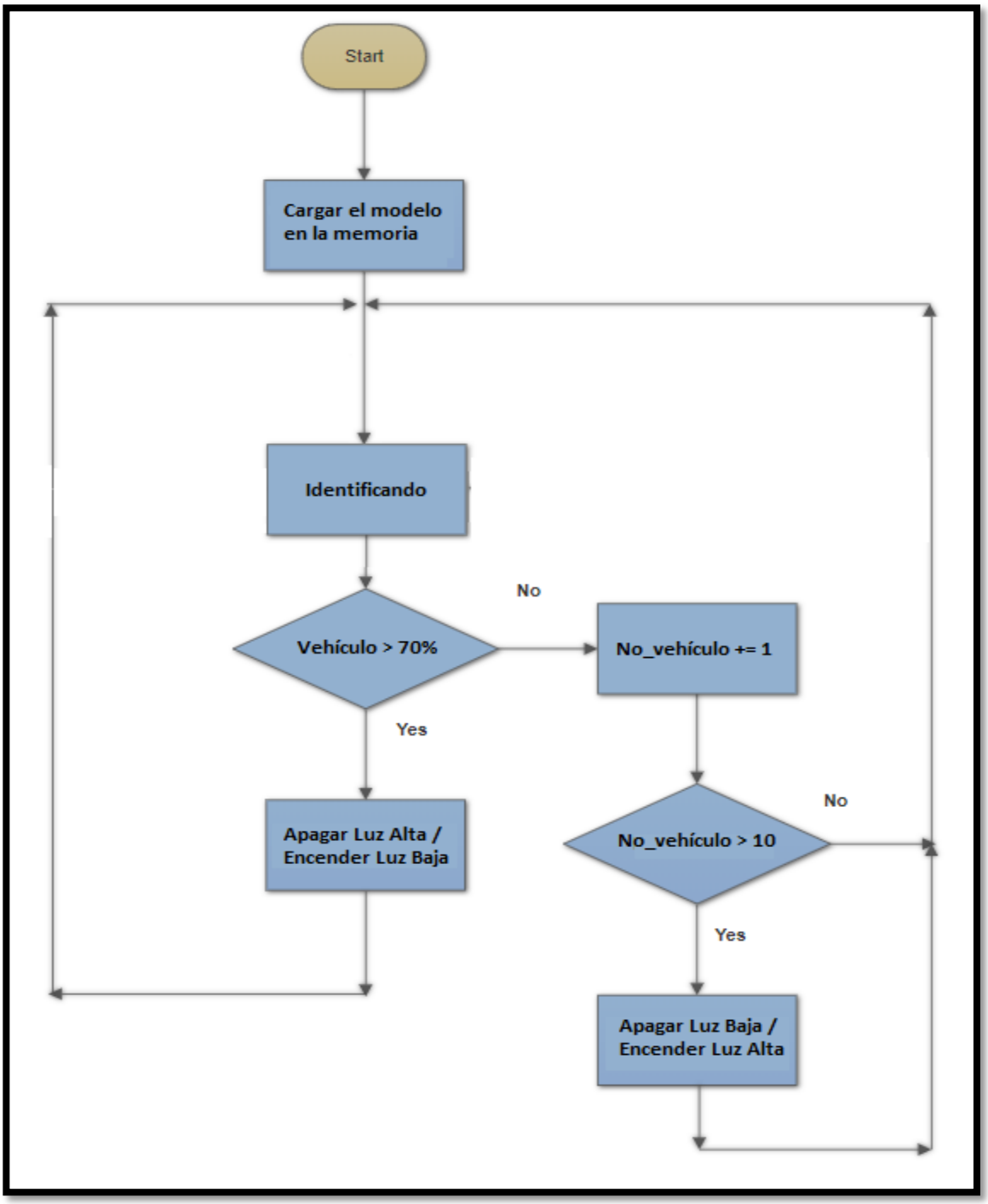


Figura 27. Diagrama de flujo lógica del sistema

Fuente: Propia

Capítulo 6: Análisis de Resultados

Los resultados del proyecto se exponen según los objetivos planteados, es decir, se analizan los resultados del proceso de entrenamiento, el desempeño del modelo entrenado, el funcionamiento del algoritmo que incorpora la red neuronal y hace el análisis de video, la generación correcta de la señal de control y los resultados de la implementación en un computador de bajo costo.

6.1 Resultados del entrenamiento

Se repitió el proceso de entrenamiento obteniendo diferentes modelos, ya que se cambiaron varios parámetros como el número de imágenes, número de pasos de entrenamiento, orientación, saturación y brillo de la imagen. Además, se utilizaron los modelos de las redes “Inception v2” y “SSDlite-Mobilenet”, como base para el entrenamiento de dos redes neuronales nuevas.

Se entrenaron las dos redes bajo las mismas condiciones, es decir, se utilizó el mismo conjunto de imágenes y hasta 30000 pasos de entrenamiento, para comparar su rendimiento posteriormente.

El conjunto de datos para el entrenamiento cuenta con 200 imágenes, cada imagen tiene como características establecidas: un tamaño de 240 de ancho por 240 de alto y una resolución de 96 dpi (pixeles por pulgada).

En la figura 28, se muestra el gráfico de la función de pérdida del entrenamiento de la **"Red neuronal 1"** basada en el modelo "SSDlite-Mobilenet"

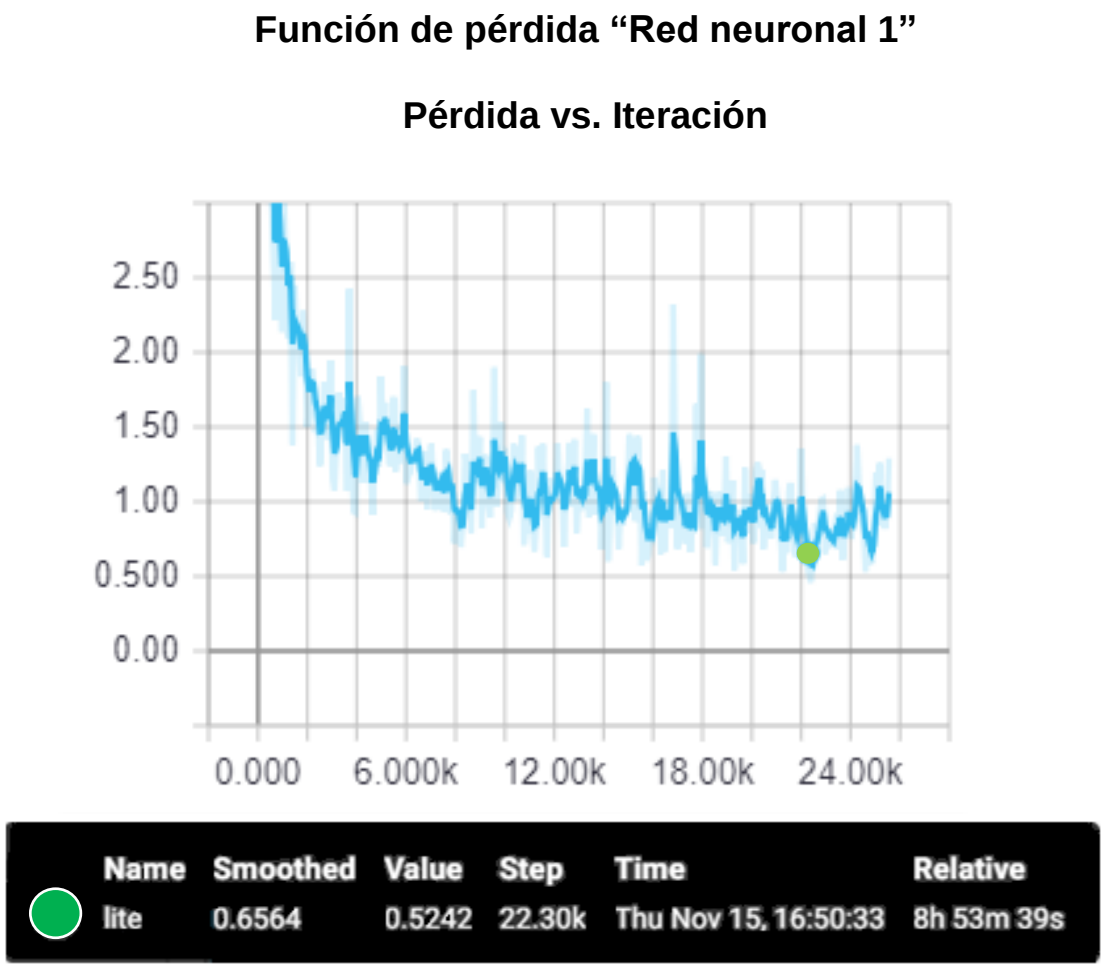


Figura 28. Función de pérdida de la "Red neuronal 1"

Fuente: Propia

Como se puede observar a partir del paso 18000, se obtiene un valor por debajo de 1, lo que indica que el entrenamiento se realizó de forma correcta y se puede parar el entrenamiento, de acuerdo con la sección 3.1.5 del marco teórico. Según el gráfico de la figura 28, las características escogidas para el conjunto de imágenes permiten un entrenamiento exitoso El valor mínimo en el gráfico indica que en el paso 22300 se alcanzó una pérdida de 0.5242 a las 8 horas y 53 minutos de empezado el entrenamiento.

En la figura 29, se muestra el gráfico de la función de pérdida del entrenamiento de la **“Red neuronal 2”** basada en el modelo “Inception v2”.

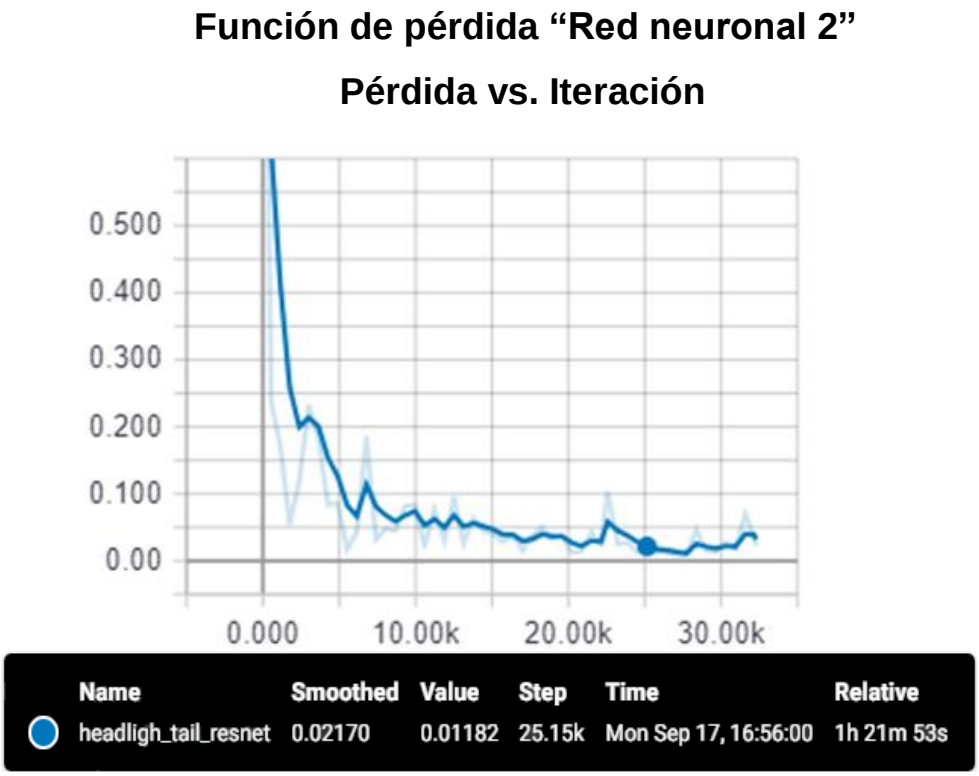


Figura 29. Función de pérdida de la "Red neuronal 2"

Fuente: Propia

A partir de la iteración 25000 se alcanza un valor constante menor a 0.1, por lo que el entrenamiento se realizó exitosamente según la sección 3.1.5 del marco teórico. El valor de pérdida en el paso 25150 es de 0.01 y se alcanzó en 1 hora y 21 minutos de iniciado el entrenamiento.

Según el gráfico de la figura 29, los parámetros utilizados y las características del conjunto de datos garantizan un entrenamiento exitoso.

En la tabla 3, se muestra los tiempos de duración del entrenamiento de una red neuronal según el número de pasos de instrucción y el tipo de procesamiento utilizado. La prueba se realizó por medio del equipo de entrenamiento de la sección 5.2.1 y la plataforma de monitoreo Tensorboard.

Tabla 3 Tiempos de entrenamiento según el tipo de procesamiento.

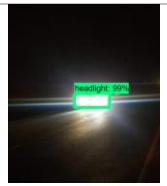
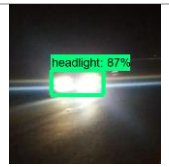
Pasos de entrenamiento	Tiempo GPU	Tiempo Procesador
500	12 min	50 min
1000	25 min	2 h
1500	37 min	2 h 53 min
2000	48 min	3 h 35 min

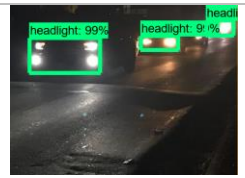
Según la tabla 3, el procesamiento en paralelo o por GPU acorta los tiempos de entrenamiento, por lo que se utiliza este tipo de procesamiento para hacer los entrenamientos.

6.2 Análisis de imágenes

Si bien se determinó que el entrenamiento se realizó exitosamente hay que comprobarlo, es decir, se debe probar el funcionamiento de las redes neuronales nuevas, para verificar si reconocen de forma correcta los objetos de interés. En la tabla 4, se clasifican diez imágenes de posibles escenarios a los que se puede enfrentar el sistema.

Tabla 4 Clasificación de imágenes.

Número	Imagen	Red neuronal 1	Red neuronal 2	Predicción Correcta Red Neuronal 1	Predicción Correcta Red Neuronal 2
1				✓	✓
2				✗	✓
3				✓	✓
4				✓	✓

5				✓	✓
6				✓	✓
7				✓	✗
8				✓	✓
9				✓	✓
10				✗	✗

Los resultados que ofrece esta prueba son muy interesantes, debido a que permite observar claramente el comportamiento de los modelos, por ejemplo, en las imágenes 2 ,4 y 7 queda en evidencia que la red 2 es más precisa.

En la imagen 10, si bien no se detectan todas las fuentes de luz, se le da un valor bajo del 30% a una de las luces, por lo que el sistema en cierto grado es susceptible a luces externas como las que se encuentran en una ciudad.

Además, en la imagen 9 se nota como ambas redes pueden detectar más de un vehículo en una misma imagen, lo cual facilitaría hacer el control del sistema.

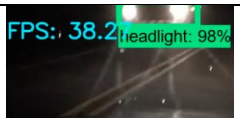
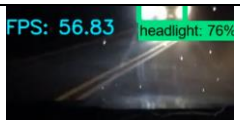
6.3 Análisis de video

Se analizan diferentes videos bajo condiciones normales de manejo, clima despejado, velocidad entre 30 – 80 km/h. La duración de los videos es aproximadamente de 2 minutos.

Por medio de las siguientes tablas se hace el análisis de dos videos, para determinar el funcionamiento del algoritmo desarrollado.

En la primera columna se especifica el minuto en el cual se registra un evento, en la segunda columna la detección realizada por un humano, en la tercera columna se indica la detección realizada por el algoritmo y en la última columna se comprueba si el algoritmo acertó correctamente las predicciones.

Tabla 5 Pruebas del video 1 con el modelo 1.

Evento (mm:ss)	Detección Humana	Detección Algoritmo	Evaluación
00:00	Vehículo	 Vehículo	Correcto
00: 26	Vehículo	 No hay Vehículo	Falso negativo
00:55	Vehículo	 Vehículo	Correcto
01:07	Vehículo	 Vehículo	Correcto
01:12	Vehículo	 Vehículo	Correcto
01:17	Vehículo	 Vehículo	Correcto
01:18	Vehículo	 Vehículo	Correcto

01:20	Vehículo	 Vehículo	Correcto
01:30	Vehículo	 Vehículo	Correcto

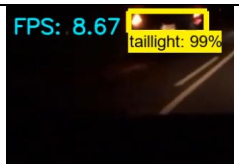
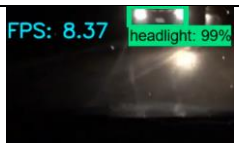
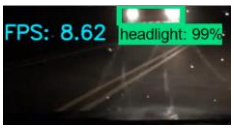
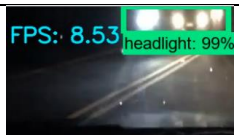
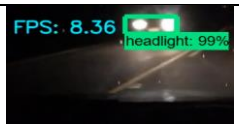
Según la tabla 5, de nueve detecciones que debió de realizar el algoritmo, se detectaron ocho vehículos, es decir, se identificó correctamente el 89% de los vehículos que se acercaron. En el evento del segundo 00:26 se obtuvo un falso negativo.

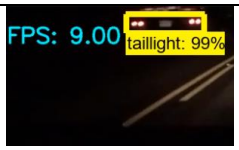
La duración del análisis fue de 1:09 minutos, menor que la duración del video, la tasa de FPS se mantuvo entre los 45-60 FPS, como se observa en la figura 30, por lo que el tiempo de la respuesta del sistema es rápido, ya que se analizan entre 45 – 60 imágenes por segundo.

Figura 30. Rendimiento del algoritmo en FPS

Fuente: Propia

Tabla 6 Pruebas de video 1 con el modelo 2.

Evento (mm:ss)	Detección Humana	Detección Algoritmo	Evaluación
00:00	Vehículo	 Vehículo	Correcto
00: 26	Vehículo	 Vehículo	Correcto
00:55	Vehículo	 Vehículo	Correcto
01:07	Vehículo	 Vehículo	Correcto
01:12	Vehículo	 Vehículo	Correcto
01:17	Vehículo	 Vehículo	Correcto
01:18	Vehículo	 Vehículo	Correcto

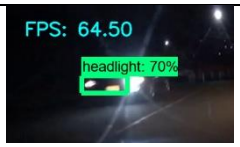
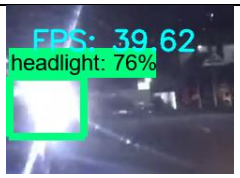
01:20	Vehículo	 Vehículo	Correcto
01:30	Vehículo	 Vehículo	Correcto

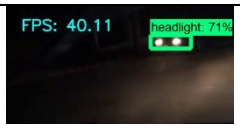
La red neuronal 2 detectó el 100% de los carros, por lo que se comprueba que es más preciso que el otro modelo, sin embargo, el análisis del video duró 4:50 minutos, dos veces más que la duración del video y la tasa de FPS se mantuvo en el rango de 8 a 10 FPS, como se observa en la figura 31.

Figura 31. Rendimiento del algoritmo en FPS

Fuente: Propia

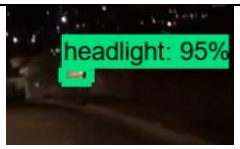
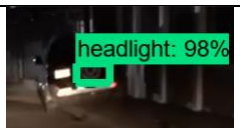
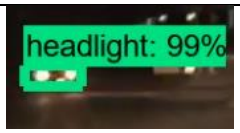
Tabla 7 Pruebas de video 2 con el modelo 1.

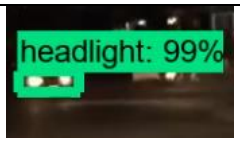
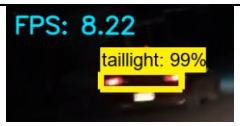
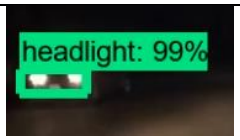
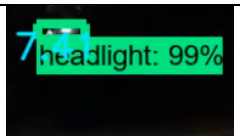
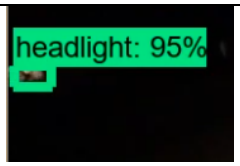
Evento (mm:ss)	Detección Humana	Detección Algoritmo	Evaluación
00:03	Vehículo	 Vehículo	Correcto
00: 14	Vehículo	 Vehículo	Correcto
00:28	Vehículo	 Vehículo	Correcto
00:37	Vehículo	 Vehículo	Correcto
00:40	Vehículo	 Vehículo	Correcto
00:46	Vehículo	 Vehículo	Correcto

01:07	Vehículo	 Vehículo	Correcto
01:13	Vehículo	 Vehículo	Correcto
01:16	Vehículo	 No hay Vehículo	Falso positivo

Como se puede ver en la tabla 7, se detectaron el total de los vehículos que aparecen, sin embargo, se obtuvo una predicción de vehículo cuando no lo había, es decir un falso positivo. El análisis del video duró 60 segundos**.**

Tabla 8 Pruebas del video 2 con el modelo 2.

Evento (mm:ss)	Detección Humana	Detección Algoritmo	Evaluación
00:03	Vehículo	 Vehículo	Correcto
00: 14	Vehículo	 Vehículo	Correcto
00:21	No hay Vehículo	 no hay Vehículo	Falso positivo
00:28	Vehículo	 Vehículo	Correcto
00:34	No hay Vehículo	 no hay Vehículo	Falso positivo
00:37	Vehículo	 Vehículo	Correcto
00:40	Vehículo	 Vehículo	Correcto

00:48	No hay Vehículo	 Vehículo	Falso positivo
00:46	Vehículo	 Vehículo	Correcto
01:07	Vehículo	 Vehículo	Correcto
01:13	Vehículo	 Vehículo	Correcto
01:17	No hay Vehículo	 No hay Vehículo	Falso positivo
01:20	No hay Vehículo	 Vehículo	Falso positivo
01:25	No hay Vehículo	 Vehículo	Falso positivo

De la misma forma cuando se utiliza el algoritmo con la segunda red neuronal, se detectaron todos los vehículos que aparecen, sin embargo, se obtuvieron 6 falsos positivos durante la prueba. El análisis del video duró 4: 30 minutos.

Observaciones:

- La primera red neuronal presenta un mejor tiempo de respuesta, se puede observar en la duración de las pruebas y en la cantidad de cuadros por segundo que analiza.
- La segunda red neuronal es más precisa, debido a que en las dos pruebas detectó el total de los carros, sin embargo, es susceptible a generar falsos positivos.
- Es necesario filtrar las predicciones para eliminar el número de falsos positivos.
- Los falsos negativos representan un peligro para el conductor del otro vehículo, debido a que no se hace el cambio de luz a luz corta, por lo que se debe tomar en cuenta a la hora de generar las señales de control.

6.4 Pruebas de filtrado

Antes de emitir la señal de control se hace un filtrado para reducir el número predicciones incorrectas.

Se observa como en la figura 32 se obtuvo un falso positivo, debido a la iluminación de dos bombillos de una casa, sin embargo, por medio del filtrado de las predicciones con valores menores a un 85%, se puede obtener como resultado la reducción de falsos positivos, como se observa en la figura 33.

Figura 32. Se detecta un falso positivo

Fuente: Propia

Figura 33. Se elimina el falso positivo por medio de la etapa de filtrado

Fuente: Propia

Se implementó la etapa de filtrado antes de generar las señales de control, para optimizar el uso de la luz larga, ya que no se cambiará a luz corta debido a falsos positivos con predicciones con un valor de acierto menor al 85%. Las predicciones con valor bajo se deben a la identificación de formas similares al patron que se desea identificar.

6.5 Pruebas de suavizado

Cuando se aproxima una fila de vehículos a diferente distancia uno de otro, el algoritmo identifica el primer vehículo, una vez que pasa, el sistema ya no lo identifica más y activa la luz alta, encandilando al segundo conductor de la fila, de la misma forma con los demás conductores que vienen, por lo que se implementó una etapa para evitar los cambios intermitentes de luz y no deslumbrar a los conductores.

En la figura 34, se muestra en consola las señales de control generadas para realizar el cambio de luz cuando se presenta la situación mencionada anteriormente. En la figura 35 se implementó una función para evitar los cambios intermitentes de luz.

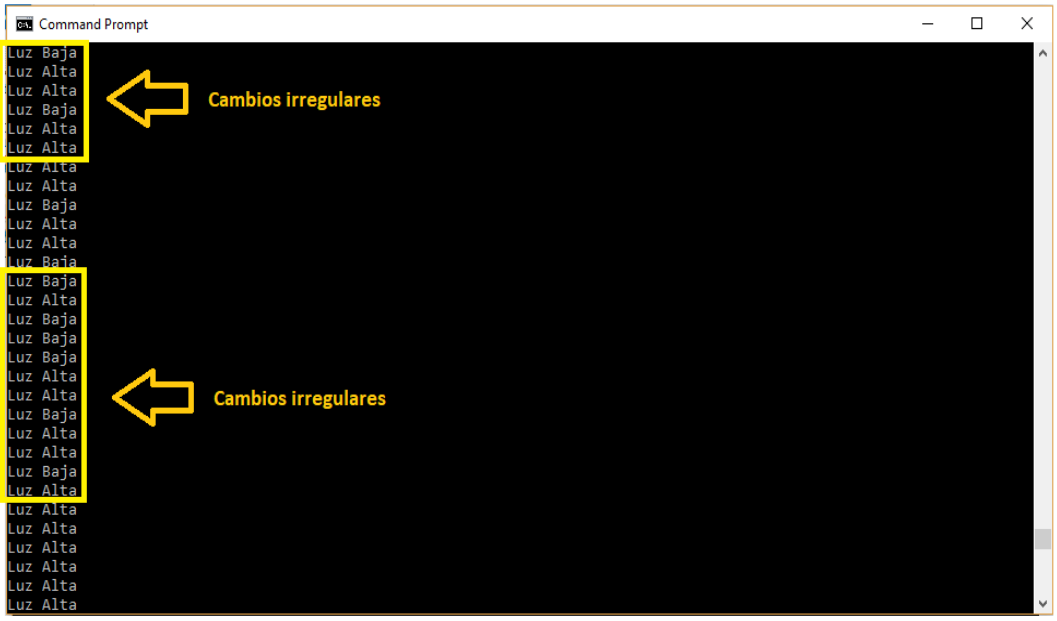


Figura 34. La señal de control genera cambios de luz intermitentes

Fuente: Propia

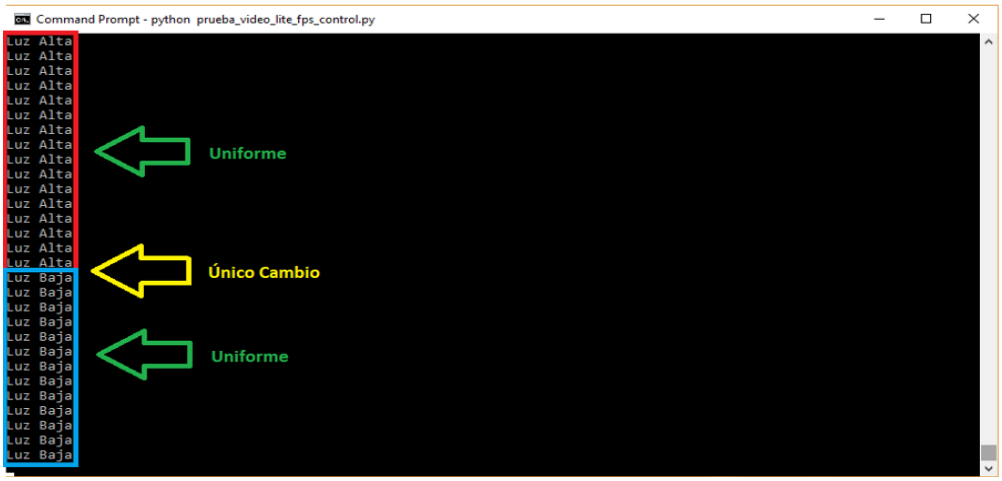


Figura 35. Etapa de suavizado

Fuente: Propia.

Como se observa en la figura 35, la salida es más uniforme, solo se da un único cambio de luz al acercarse a la fila de autos y se mantiene baja, por lo que se elimina el parpadeo y por consecuente posibles encandilamientos.

6.6 Pruebas regulador de voltaje

En la tabla 9, se muestra el voltaje de salida del regulador de voltaje para diferentes valores de tensión de entrada.

Tabla 9 Valores de entrada y salida del regulador.

Tensión de entrada (V)	Tensión de salida (V)
5	3.565
6	4.547
10	4.955
12	**4.957**
15	4.958

Según la tabla 9 para valores de tensión mayores a 10 V la salida se mantiene constante, en aproximadamente 4.9 V. Por lo que se cumplen con los requisitos para poder energizar adecuadamente el microprocesador utilizado.

6.7 Implementación en un computador de bajo costo

El sistema se implementó en las placas de computador simple Raspberry Pi 3 y Asus Tinkerboard S, se pudo implementar en ambas plataformas con el entorno de desarrollo necesario para permitir el funcionamiento del algoritmo desarrollado.

La rutina para obtener la cantidad de cuadros por segundo a los que se analiza el video permite medir el rendimiento general del sistema en cada una de las plataformas. En la tabla 10, se muestra la plataforma utilizada, el rango de valores de FPS obtenidos durante las pruebas y el tiempo de respuesta de cada una de las unidades de procesamiento.

Tabla 10 Rendimiento de los diferentes computadores utilizando la red neuronal 1.

Unidad de procesamiento	Velocidad de procesamiento GHz	Memoria RAM GB	FPS	Tiempo de respuesta (s)
Raspberry Pi 3	Quad Core a 1.2	1	0.5 - 1	8- 10
Tinkerboard S	Quad Core a 1.8	2	3 - 4	1-2
Equipo entrenamiento	Quad Core a 2.6	12	40 - 60	Inmediato

No se puede utilizar las Raspberry Pi 3, debido que se obtiene un tiempo de respuesta lento de 10 segundos. Para una aplicación en donde los objetos de interés se mueven a cierta velocidad, este tipo de placa no es funcional. Por otro lado, para aplicaciones más estáticas, como por ejemplo el conteo de un producto de interés en un estante, se podría utilizar esta opción**.**

En la Tinkerboard se obtiene entre 3 - 5 FPS, una mejora considerable en comparación a la Raspberry Pi 3 y un tiempo de respuesta entre 1-2 segundos.

En la figura 36, se muestra la cantidad de recursos que utiliza un equipo para realizar la tarea de detección en tiempo real, con un tiempo de respuesta cercano a 0 segundos.

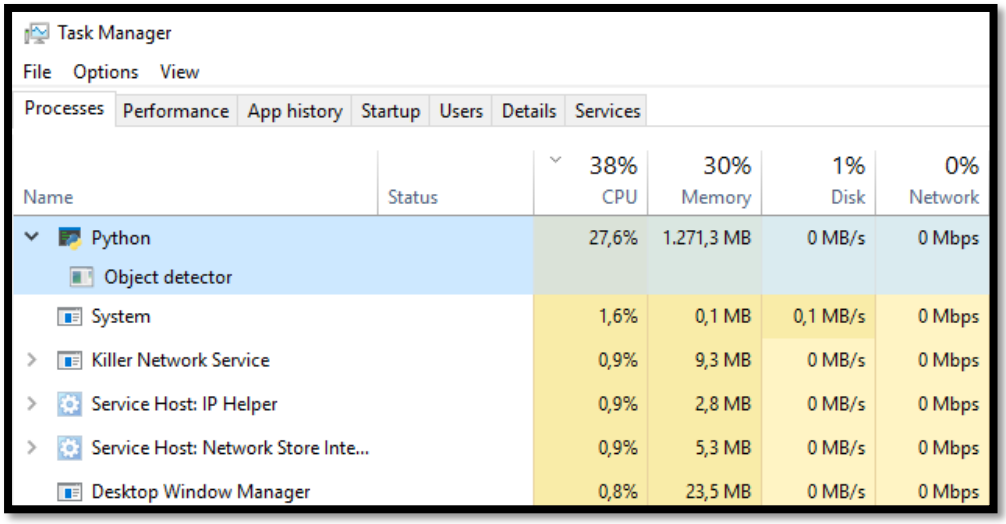


Figura 36. Utilización de recursos detección en tiempo real.

Fuente: Propia.

Conociendo la velocidad de procesamiento de las diferentes plataformas utilizadas y por medio de las pruebas realizadas, se puede conocer una aproximación de las especificaciones técnicas que necesita un computador para garantizar un tiempo de respuesta cercano a los cero segundos.

Se necesita 1,5 GB de memoria RAM y un procesador de 4 núcleos con una velocidad de procesamiento igual o superior a los 2.6 GHz. Se da una aproximación, debido a que el rendimiento de la placa que se escoja como unidad de procesamiento, dependerá de múltiples factores, como la arquitectura del computador, el sistema operativo, entre otros.

Capítulo 7: Conclusiones y recomendaciones

7.1 Conclusiones

- Para obtener un conjunto de datos válidos, es necesario tomar una imagen de cada posible escenario con el que se vaya a enfrentar la red neuronal.

- La aleatoriedad en el proceso de entrenamiento produce resultados diferentes para un mismo conjunto de imágenes.

- Utilizar una GPU que permita procesamiento en paralelo reduce considerablemente la duración de los procesos de entrenamiento.

- Utilizar imágenes con una resolución de 96 dpi y un tamaño de 240 de ancho por 240 de alto, permitió entrenar correctamente la red neuronal para reconocer el patrón que describe un vehículo durante la noche.

- Según las funciones de pérdida para obtener un valor de pérdida aceptable, se debe de entrenar una red neuronal por más de 20 000 pasos.

- El conjunto de imágenes recopilado cumplió con los requisitos necesarios para realizar un entrenamiento adecuado.

- Las redes entrenadas permiten detectar satisfactoriamente cuando otro vehículo se está aproximando durante la noche, cada una con precisión y tiempo de respuesta diferente, ya que estos resultados varían según el modelo que se utilice como base.

- La “red neuronal 1” presenta mejores características de precisión y tiempo de respuesta para satisfacer las necesidades del proyecto.

- Establecer que las predicciones reales fueran solamente las que tuvieran 85% de acierto disminuyó la cantidad de falsos positivos.

- El algoritmo desarrollado para detectar un vehículo en tiempo real o en video durante la noche, permite generar las señales de control para subir o bajar la luz correctamente.

- Se aproximaron las especificaciones de memoria (1.5 GB) y de procesamiento (Quad Core de 2.6 MHz) que garantizan un tiempo de respuesta cercano a cero.

- El circuito implementado para regular el voltaje garantiza una salida constante de 5 V para voltajes de entrada entre 8V - 40V, por lo que soporta las variaciones presentes en un vehículo.

7.2 Recomendaciones

- Aumentar el número de imágenes para mejorar la precisión del modelo.

- Implementar sensores adicionales que midan el nivel de luz, para que contribuyan con la generación de una señal de control más eficiente.

- Implementar el regulador de voltaje en un circuito impreso para compactar los componentes y de esta forma disminuir el espacio que ocupa.

- Se recomienda implementar un ventilador a la unidad de procesamiento y al regulador de voltaje, para evitar sobrecalentamientos.

- Se recomienda realizar el acople con el mecanismo mecánico o electrónico encargado del cambio manual de las luces y el sistema propuesto.

- Se sugiere considerar utilizar un equipo con mayor capacidad de procesamiento para la ejecución de posibles trabajos posteriores relacionados con el proyecto,

- Utilizar una carcasa para los módulos resistente a las condiciones de vibración y calor producidas por el motor del vehículo.

Bibliografía

[1] Consumer, «Espejos anti deslumbramiento,» Consumer, 14 Junio 2004. [En línea]. Available: http://www.consumer.es/web/es/motor/mantenimiento_automovil/2004/06/14/104185.php. [Último acceso: 12 Diciembre 2018].

[2] Y. Campos, Interviewee, *Definición del problema.* [Entrevista]. 10 Agosto 2018.

[3] S. A. M.Alsumady, «Intelligent Automatic High Beam Light Controller,» Old City Publishing, Jordan, 2013.

[4] F. S. Caparrini, «Introducción al Aprendizaje Automático,» 23 Septiembre 2017. [En línea]. Available: http://www.cs.us.es/~fsancho/?e=75. [Último acceso: 5 Agosto 2018].

[5] TensorFlow, «TensorFlow,» 3 Marzo 2018. [En línea]. Available: https://www.tensorflow.org/hub/tutorials/image_retraining?hl=es. [Último acceso: 5 Agosto 2018].

[6] H. Kinsley, «pythonprogramming,» 25 Agosto 2017. [En línea]. Available: https://www.youtube.com/watch?list=PLQVvvaa0QuDcNK5GeCQnxYnSSaar2tpku&v=srPndLNMMpk. [Último acceso: 1 Noviembre 2018].

[7] arducam, «arducam,» [En línea]. Available: http://www.arducam.com/category/camera-module-demonstration/. [Último acceso: 30 agosto 2018].

[8] Toyota, «How Automatic High Beams help make your night driving safer,» Toyota, 17 Marzo 2017. [En línea]. Available: https://www.toyota.ca/toyota/en/connect/402/how-automatic-high-beams-help-make-your-night-driving-safer. [Último acceso: 15 Agosto 2018].

[9] F. CEA, «seguridad-en-la-circulacion,» 15 Agosto 2018. [En línea]. Available: https://www.seguridad-vial.net/conduccion/seguridad-en-la-circulacion/129-la-vision-en-la-conduccion.

[10] Cosevi, «cvs.go,» 1 octubre 2018. [En línea]. Available: https://www.csv.go.cr/documents/. [Último acceso: 3 septiembre 2018].

[11] D. G. P. P. R. S. R. y. A. W. Mace, «Countermeasures for reducing the effects of headlight glare,» AAA Foundation for Traffic Safety., 2001.

[12] J. M. &. F. M. J. Sullivan, «The role of ambient light level in fatal crashes: inferencesfrom daylight saving time transitions,» de *Accident Analysis & Prevention*, 2002, pp. 487-498.

[13] Asus, «tinker board S,» 21 marzo 2018. [En línea]. Available: https://www.asus.com/us/Single-Board-Computer/Tinker-Board-S/. [Último acceso: 23 agosto 2018].

[14] T. Instruments, «ti,» Marzo 2013. [En línea]. Available: http://www.ti.com/lit/ds/symlink/lm350-n.pdf. [Último acceso: 27 Noviembre 2018].

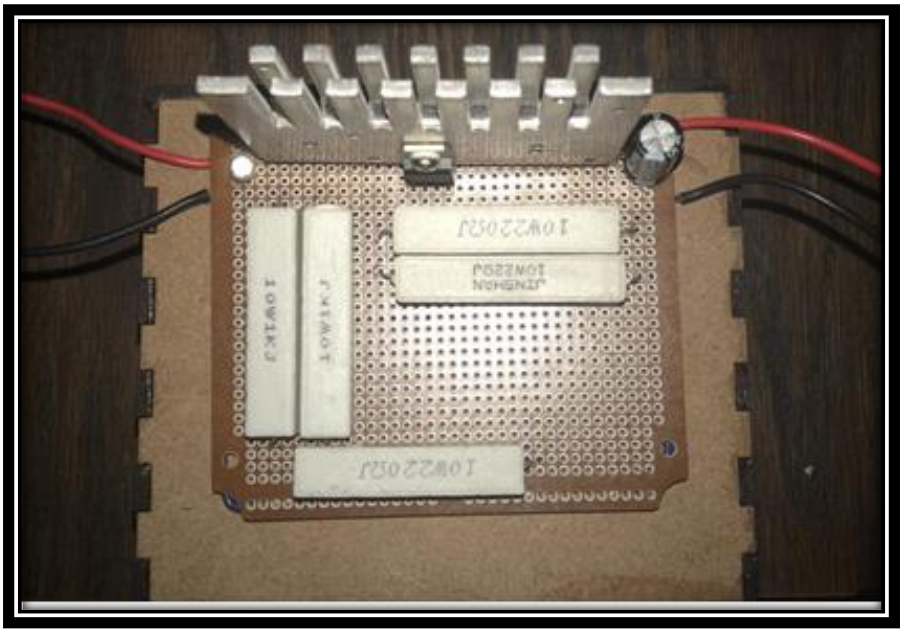
[15] Raspberry, «raspberry,» raspberry, 10 enero 2018. [En línea]. Available: https://www.raspberrypi.org/products/. [Último acceso: 25 agosto 2018].

[16] Nvidea, «nvidia,» nvidia, 22 agosto 2017. [En línea]. Available: https://www.nvidia.com/en-us/self-driving-cars/drive-platform/. [Último acceso: 26 agosto 2018].

[17] A. Chris, «Aprendizaje automático,» 1 diciembre 2018. [En línea]. Available: https://www.inteldig.com. [Último acceso: 3 diciembre 2018].

Apéndice A:

Foto del circuito regulador de voltaje implementado.



Apéndice B:

Foto del prototipo desarrollado.