UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE CONTEO VEHICULAR MEDIANTE INTELIGENCIA ARTIFICIAL PARA REALIZAR EL CONTROL ADAPTATIVO DE REGULADORES DE TRÁNSITO LC2

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Electrónica

AUTORAS: CÓRDOVA MARTILLO JOCELYNE THAIS
MENDIETA GARCÉS KEVYN ALEXANDER
TUTOR: ING. GEOVANNY GARCÍA,MSc

Guayaquil – Ecuador
2023

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Córdova Martillo Jocelyne Thais con documento de identificación N° 0926322900 y Mendieta Garcés Kevyn Alexander con documento de identificación N° 0953625100, manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación

Guayaquil, 18 de enero del año 2024.

Atentamente,

Jocelyne Thaís Córdova Martillo
0989639888

Kevyn Alexander Mendieta Garcés
0991262373

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Córdova Martillo Jocelyne Thaís con documento de identificación N° 0926322900 y Mendieta Garcés Kevyn Alexander con documento de identificación N° 0953625100, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE CONTEO VEHICULAR MEDIANTE INTELIGENCIA ARTIFICIAL PARA REALIZAR EL CONTROL ADAPTATIVO DE REGULADORES DE TRÁNSITO LC2", el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica**,** quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

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

Guayaquil, 18 de enero del año 2024.

Atentamente,

Jocelyne Thaís Córdova Martillo
0989639888

Kevyn Alexander Mendieta Garcés
0991262373

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Geovanny García con documento de identificación N° 0922357702, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación**:** DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE CONTEO VEHICULAR MEDIANTE INTELIGENCIA ARTIFICIAL PARA REALIZAR EL CONTROL ADAPTATIVO DE REGULADORES DE TRÁNSITO LC2, realizado por Córdova Martillo Jocelyne Thaís con documento de identificación N° 0926322900 y Mendieta Garcés Kevyn Alexander con documento de identificación N° 0953625100, obteniendo como resultado final el trabajo de titulación bajo la opción de Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 18 de enero del año 2024.

Atentamente,

Ing. Geovanny García

0922357702

DEDICATORIA

A los que siempre han estado conmigo a lo largo de mi vida siguiendo mis pasos cuidándome, guiándome, queriéndome y que siempre han querido lo mejor para mí. A mi papi, a mi mami y a mi hermano. Este logro profesional es de cada uno de nosotros.

Jocelyne Córdova M.

A mi madre, la mujer más importante de mi vida, Miriam Gioconda Garces Aguirre, quien me enseñó a nunca dejar de soñar, a nunca rendirme, aunque todo este en contra, a siempre tender la mano a quien sea que lo necesite. Todo lo que soy es gracias a ti.

Kevyn Mendieta G.

\

AGRADECIMIENTO

A Dios por guiarme en el camino.

A mis padres y a mi hermano, que han creado a la persona en la que me he convertido, gracias por estar para mi cuando los necesito, ayudándome, apoyándome, soportándome y siendo mi ejemplo de superación. A los Vera por alegrarse de cada uno de mis triunfos. A Edu, Miguel y Mendieta quienes han formado parte de este progreso y demostrado ser unos muy buenos amigos. A Boris que siempre estuvo pendiente de mí, ayudándome con lo que ha podido.

Jocelyne Córdova M.

A Dios por brindarme una familia increíble.

A mi padre, Carlos, que ha sido parte importante en todas las etapas de mis estudios brindándome su apoyo incondicional y compañía en todo el proceso.

A mis hermanas, Karina, Diana y Anahis por soportarme y ayudar a madurar.

A mis tíos, Henry y Carmen por ser un apoyo increíble en todos los problemas vividos.

A mi familia por estar siempre presente y unida. A mis amigos, Allan y Karelys que me demuestran que, aunque pasen los años los amigos siempre están juntos.

A mi amiga y compañera de tesis, Jocelyne, quien admiro y sin querer se convirtió en mi compañía durante todo lo vivido en la universidad.

Kevyn Mendieta G.

Resumen

Con este documento se quiere mostrar como el uso de la inteligencia artificial puede ser usado para realizar procedimientos de una forma eficiente, que posteriormente sirvan a la comunidad como es el caso de los semáforos. Por eso se estudia cómo se puede obtener la cantidad de vehículos que pasan en una intersección mediante la IA, usando un sistema embebido donde se instala Python para poder programar. Con el fin de enviar la cantidad de vehículos que pasan por esa intersección a un microcontrolador de tránsito (LC2), que con esa información realiza las fórmulas necesarias para tener un tiempo óptimo en las luces de los semáforos. Siendo posible el envío y recepción de datos desde Python hasta el LC2 mediante dos módulos ESP32. Estos módulos emplean la comunicación serial y el protocolo de comunicación ESP-now, este protocolo requiere que un módulo trabaje como maestro y otro esclavo.

*Palabras Clave***:** Inteligencia artificial, Python, ESP32, ESP-now, LC2.

Abstract

This document aims to show how the use of artificial intelligence can be used to carry out procedures in an efficient way, which subsequently serve the community as in the case of traffic lights. Therefore, we study how to obtain the number of vehicles passing through an intersection by means of AI, using an embedded system where Python is installed to be able to program. In order to send the number of vehicles passing through that intersection to a traffic microcontroller (LC2), which with that information performs the necessary formulas to have an optimal time in the traffic lights. It is possible to send and receive data from Python to LC2 by means of two ESP32 modules. These modules use serial communication and the ESP-now communication protocol, this protocol requires that one module works as a master and the other as a slave.

Keywords: Artificial intelligence, Python, ESP32, ESP-now, LC2.

ÍNDICE DE CONTENIDO

Índice de Figuras

Índice de Tablas

I INTRODUCCIÓN

Con este primer capítulo, se quiere explicar los pasos que se realizaron con el fin de que el lector comprenda el presente trabajo, proporcionando los antecedentes debidos. Mostrando la programación adecuada en Python, que permita desarrollar la inteligencia artificial con el fin de poder contabilizar los vehículos que pasan por una intersección vehicular.

En el proyecto se investiga cual es el mejor algoritmo destinado a la detección y seguimiento de objetos que entrega una mayor facilidad, menor índice de errores y una mejoría en el rendimiento de la misma. De igual forma se considera importante descubrir cómo se puede realizar la comunicación entre el sistema embebido y el controlador LC2.

Este trabajo se lleva a cabo con el uso de Yolo, comparándolo con el resto de sus versiones, similares a la hora de ejecutarse, pero presentando diferentes rendimientos. Se hace uso del lenguaje de programación Python para configurar y modelar nuestro algoritmo Yolo a la vez que realiza el conteo de los vehículos.

A su vez, se utiliza un entorno de desarrollo integrado utilizado en programación informática. Este software es Pycharm, el cual en este proyecto permite una mayor facilidad al programar.

II PROBLEMA

La mayoría de los semáforos que existen son programados para tener distintos planes a lo largo del día, en donde su ciclo corresponde a un tiempo designado de forma fija para las luces rojas, amarillas y verdes. Por lo cual, cuando se presentan flujos inesperados de los vehículos, estos siguen con su programación habitual.

Al contrario, los semáforos adaptativos pueden designar los tiempos óptimos de luces dependiendo del flujo vehicular que se encuentre en ese momento con el objetivo de reducirlo de una mejor forma.

En la intersección de las calles Av. Domingo Comín y Padre Ignacio Rueda Latasa que corresponde a la ruta de salida para Estudiantes y Docentes de la Universidad Politécnica Salesiana, el tráfico en las horas picos ocasiona retrasos de entre 20 y 30 minutos para la entrada y salida de dicha Universidad. Esto es debido a la carencia de un sistema de semaforización adaptativo que permita regular los tiempos entre cada cambio de luz del semáforo mediante inteligencia artificial en esas calles.

Por lo cual se considera que este tema es importante porque plantea mejorar el flujo vehicular existente en la intersección dando la oportunidad a los estudiantes y docentes llegar a tiempo a sus actividades académicas y retornar a sus hogares en un menor tiempo.

III OBJETIVOS

3.1 Objetivo general

Diseñar e implementar un prototipo de conteo vehicular mediante inteligencia artificial para realizar el control adaptativo de reguladores de tránsito LC2.

3.2 Objetivos específicos

• Desarrollar mediante un sistema embebido el modelamiento de la inteligencia artificial para reconocer y contabilizar a los vehículos.

• Diseñar una tarjeta de interfaz electrónica para la comunicación entre el sistema embebido y el controlador LC2.

• Identificar el correcto funcionamiento al visualizar el cambio de los tiempos en las luces de los semáforos.

IV FUNDAMENTO TEÓRICO

4.1 Sistemas embebidos

Los sistemas embebidos son sistemas computacionales compuestos de circuitos electrónicos digitales, con los cuales se pueden realizar diversas tareas de control que sean específicas en tiempo real. Por ejemplo, Raspberry Pi, Odroid, Panda, entre otros. Estos sistemas en su arquitectura tienen un microprocesador quien es el encargado de analizar la programación y realizar las tareas planeadas.

Para elaborar la programación se pueden utilizar diversos lenguajes como el ANSI C que es universal para algunos sistemas embebidos, contiene librerías que facilita la programación y también se programa por funciones. El lenguaje C++ es una programación orientada a objetos, haciendo arreglos de datos para facilitar la programación (Salas, 2015).

Otro lenguaje de programación que está orientado a objetos es Python, el cual cuenta con una sintaxis e instrucciones que permiten reducir el tiempo en la elaboración de líneas de código (Trejos, Muñoz, 2021).

4.2 Inteligencia artificial

La inteligencia artificial, con el acrónimo en español IA, fue incluida por el matemático John Mc Carthy como una ciencia de la ingeniería que logra que las computadoras razonen y aprendan, desarrollando programas inteligentes a través de algoritmos. Siendo los algoritmos una serie de pasos que se realizan en busca de resolver un problema. Pero si se necesita almacenar y analizar una mayor cantidad de datos se puede utilizar el aprendizaje automático, conocido en inglés como Machine Learning (Garrido, 2020).

4.3 Visión artificial

Con la visión artificial se quiere distinguir una imagen para entender que sucede dentro de la misma. Teniendo un proceso para la captación de la imagen, segmentación, localización y reconocimientos de los vehículos que transcurren por una calle para poder contabilizarlos. Dos de los componentes imprescindibles son un sensor óptico o video cámara que permita la detección de la imagen.

El otro es un computador que guarda las imágenes procesándolas para continuar con la segmentación, que es donde se dividen las imágenes en subregiones resaltando la región que son de interés, con el objetivo de que sea más fácil el reconocimiento de los vehículos (Loaiza, Manzano & Múnera, 2012).

4.4 Tipos de imágenes

Existen las imágenes de intensidad que, con dispositivos ópticos como las cámaras fotográficas o video cámaras, son capaces de captar la luz. También se encuentran las imágenes de alcance que no usan la luz para conseguir la imagen sino otros dispositivos ópticos como el láser, el radar y el sonar. Por último, se puede disponer de cámaras térmicas que arrojan otro tipo de imágenes.

Se pueden usar las imágenes de alcance para medir distancias entre objetos, imágenes térmicas para obtener distintos valores de temperatura y las imágenes de intensidad, que serán las utilizadas en este proyecto para localizar a los vehículos en las imágenes captadas (Loaiza, Manzano & Múnera, 2012).

4.5 Regiones de interés

Las regiones de interés son aquellas regiones con gran porcentaje de contener algún objeto en imágenes o videos en tiempo real. Este proceso ayuda a disminuir drásticamente el tiempo de cómputo y a mejorar la eficiencia de los resultados al momento de detectar objetos. (Calero, Quinga, Onofa & Gallardo, 2019).

4.6 Red neuronal convolucional

Una red neuronal convolucional es una red neuronal artificial creada por software y hardware. Estas neuronas artificiales se conectan a campos receptivos de manera similar a las neuronas de un cerebro biológico. Desde el punto de vista de Massiris, Delriux y Fernández (2018) el propósito de la red neuronal convolucional es extraer todas las características de una imagen y luego usar dichas características para detectar o clasificar los objetos en una imagen.

Los parámetros de los filtros que se pueden aprender en estas capas; se ajustarán y optimizarán junto con los componentes de clasificación para minimizar el error de clasificación total.

Desde que las redes neuronales fueron creadas, el campo de detección de objetos ha avanzado a pasos agigantados permitiendo a los algoritmos desarrollados en los últimos años destacar en este campo entregando cada vez un mejor resultado para diferentes tareas asignadas en tiempo real.

4.7 Yolo

El algoritmo conocido como Yolo cuyas siglas significa “You Only Look One” (Solo Mira una Vez) es un sistema de codigo abierto que utiliza una red neuronal convolucional única para detectar objetos en imágenes o videos en tiempo real con alta precisión y rendimiento computacional eficiente. Este algoritmo fue creado y desarrollado en el año 2016 por Joseph Redmon y Santosh Divvala.

Yolo a lo largo de los años ha presentado un total de 8 versiones la cual hasta la fecha Yolov8 es la versión que presenta mejor velocidad, rendimiento, estabilidad y precisión.

Yolov8 es desarrollada por la empresa Ultralytics, mezclando todas las características de las versiones anteriores de Yolo junto con información contextual con el fin de mejorar la precisión en la detección de objetos. (Terven & Cordova, 2019).

Esta versión para realizar la detección de objetos, hace el uso de tres capas de detección de escala para acomodar objetos de diferentes escalas, las cuales son el segmento de entrada, la columna vertebral y el segmento de salida. (Wang, Gao, Jia & Li, 2023).

4.8 Dataset

EL Dataset son conjuntos de datos, tambien llamados conjuntos de entramiento o conjuntos de datos de entrada, el cual se puede dividir en diferentes clases y estas en subclases teniendo cada una ellas una etiqueta de identificación. Generalmente se identifican con la terminación YAML. Su uso se implementa en proyectos relacionados con IA ya que estas necesitan de un previo entrenamiento. (Jesús Bobadilla, 2020).

4.9 DeepSORT

El algoritmo DeepSort es una extensión de SORT (Simple Real Time Tracker), es de los mejores algoritmos de seguimiento más empleados en proyectos de inteligencia artificial. Este algoritmo añade información de apariencia para mejorar el rendimiento de SORT lo que da la opción de rastrear objetos durante periodos más largos. Los algoritmos SORT y DeepSORT son los más eficientes en el rastreo de objetos, teniendo en cuenta el balance entre la precisión y la velocidad (Ameijeiras, González & Hernández, 2020).

4.10 Comunicación serial

La comunicación serial es un método de transferencia de datos secuencial, bit por bit, esto a través de un canal de comunicación. Este método de fácil comprensión tiene características propias como la velocidad de transmisión, protocolos de comunicación citando de ejemplo RS-232, SPI, I2C, sincronización de datos y la confianza en lugares donde lo importante es la transferencia de datos eficiente y precisa.

4.11 Comunicación ESP32

El ESP32, fue desarrollado por Espressif Systems, siendo este un sistema de bajo costo y de un menor consumo de energía compuesto de una serie de chips (SoC) con la capacidad de conectarse mediante Wi-Fi y Bluetooth de modo dual. Los interruptores de antenas, el balun RF, el amplificador de potencia, el amplificador de recepción de bajo ruido, los filtros y los módulos de administración de energía se integran muy bien con esta tarjeta electrónica. Está pensado para que funcione en aplicaciones vinculadas con el IoT y dispositivos móviles (Fernández & Uquillas, 2020)

4.12 Protocolo de comunicación ESP-NOW

El protocolo de comunicación ESP-NOW fue creado por Espressif, empresa que desarrolló el microcontrolador ESP32. Entre sus características de funcionamiento está que es una comunicación eficiente, rápida e inalámbrica, con la cual se puede enviar un máximo de 250 bytes, con un alcance de 220 metros en condiciones ideales, variando en condiciones reales, opera en la banda de 2,4 GHz y estas pueden ser susceptible a interferencias de otros dispositivos.

También es posible la comunicación entre dos dispositivos ESP32 sin el uso de una red wifi, ya que todos los dispositivos cuentan con una única dirección MAC, que facilita identificar la placa que trabaja como receptora (Arduino, 2024).

Es posible configurar de dos formas su comunicación, dependiendo de lo que se necesite, existe la comunicación unidireccional y la comunicación bidireccional. Como su nombre lo indica, la primera es capaz de enviar la información desde un dispositivo que trabaje como emisor para ser recibida por un dispositivo que sea receptor, pero no consta de una retroalimentación del receptor al emisor.

En cambio, con la segunda comunicación teniendo dos dispositivos, estos pueden enviar y recibir información de forma bidireccional. Es decir, que cada uno trabajará como emisor y receptor en un determinado momento (Arduino, 2024).

4.13 Controlador de tránsito LC2

El controlador de tránsito es un equipo electrónico capaz de controlar varios semáforos que se encuentran en una intersección con la finalidad de aligerar el flujo de vehículos, permitir el paso seguro de los peatones reduciendo de esa manera accidentes vehiculares.

La empresa CORMAR S.A. entre sus diversos controladores a comercializarcuenta con el LC2 el cual posee las siguientes características:

- Microcontrolador de gama alta.
- Display de Cristal Líquido LCD-20x4.
- Teclado numérico 3x4.
- Conector para entrada y salida de sincronismo.
- Conector para señales de entrada (pulsadores).
- Programación de planes de trabajo.
- Programación de flasheo.
- Reloj interno de precisión para control de tiempos.
- Posibilidad de programación inclusive de semáforo peatonales.
- Conectores desmontables para reemplazo de tarjeta electrónica.
- Todas las salidas son de relé de estado sólido.
- Leds indicadores de estados de fase semafóricas.
- Corriente de salida para fase de fuerza de hasta 1 Amp. por salida.

En la siguiente figura 1 se muestra al controlador LC2, con cada uno de sus componentes, entre ellos la tarjeta principal que consta de un microcontrolador 18F4550, periféricos externos como teclado matricial de 4x3, Display LCD 20x4, módulo de reloj de tiempo real, convertidores de serie a paralelo, borneras, etc.

También se puede apreciar los dispositivos de protección como lo son breakers y fusileras, fuente de poder de 110v a 5v y los módulos de interface que permiten accionar los semáforos con tan solo 5v de control.

Figura 1

Controlador LC2

 .

Nota: También se observa en la parte inferior de la imagen las borneras donde se conectan las luces de cada uno de los semáforos. Fuente: Diseño e implementación de una Red inalámbrica para el Control de Reguladores de tránsito programables autosustentables, mediante el uso de Módulos XBEE Pro y Paneles Solares, por L. Córdova, 2022, Universidad Politécnica Salesiana.

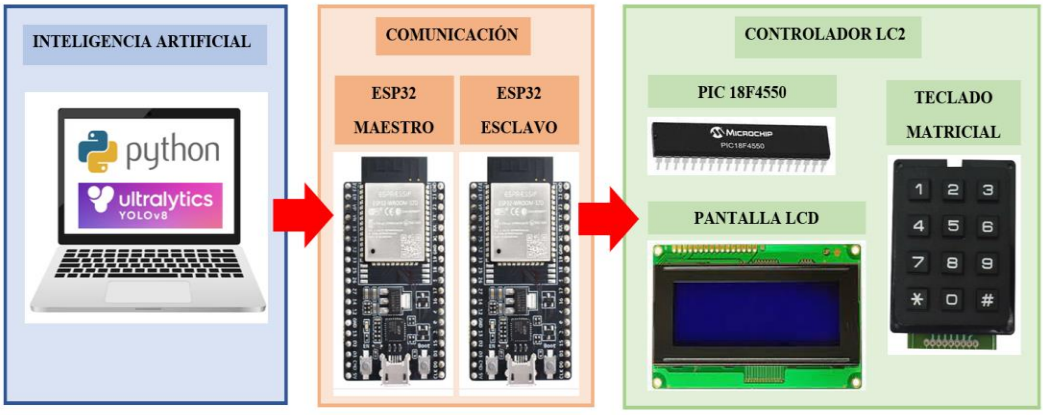
V MARCO METODOLÓGICO

Para empezar, se quiere situar al lector concretamente en las fases que se realizaron para poder llevar a cabo el proyecto de titulación. Por consiguiente, con la ayuda de software Python y Yolov8 se consigue la programación de la inteligencia artificial para el conteo vehicular, captado en la intersección que corresponde a las calles Av. Domingo Comín y Padre Ignacio Rueda Latasa.

A su vez, el Python envía mediante comunicación serial la cantidad de vehículos a un módulo ESP32 configurado como maestro, el cual tiene el fin de comunicase con otro módulo ESP32 configurado como esclavo mediante el protocolo de comunicación ESP-now. Por último, el módulo ESP32 esclavo envía estos datos al microcontrolador del semáforo, el cual con un modelamiento matemático genera el tiempo óptimo para las luces de los semáforos. En la figura 2 que se presenta a continuación se puede observar gráficamente lo que se detalla.

Figura 2

Diagrama unifilar del sistema inteligente



Nota: Se concidera cada recuadro como los pasos a seguir para este proyecto. Fuente:Los autores.

5.1 Ejecutar Yolov8 en Python

Para lograr el desarrollo del modelamiento de la inteligencia artificial en un sistema embebido para reconocer y contabilizar a los vehículos, que corresponde al primer objetivo específico, se realizan los siguientes pasos:

5.1.1 Acceder al repositorio de GitHub

Para este proyecto se toma como punto de partida un repositorio de GitHub el cual se encuentra público para su uso, se seguirá las instrucciones que este proporciona para acceder, descargar y ejecutar Yolov8 con DeepSORT en Pyhcarm.

Antes de ejecutar Yolov8 con DeepSORT en Pycharm se debe acceder a su contenido, para ello se debe clonar el repositorio. Para clonar el repositorio desde Pycharm, se usa la siguiente línea de comando:

git clone https://github.com/MuhammadMoinFaisal/YOLOv8-DeepSORT-Object-Tracking.git

Este comando permite clonar el repositorio al que se apunta y descargar de manera local los archivos en un nuevo directorio.

Culminada la clonación del repositorio de manera local, Pycharm notifica un aviso desde la terminal indicando que la descarga ha culminado.

Al culminar la descarga del repositorio, se debe acceder a la carpeta clonada para instalar las bibliotecas que necesita Yolov8 con DeepSORT para su correcto funcionamiento. Para acceder a la carpeta clonada desde pycharm se usa la siguiente línea de comando:

cd YOLOv8-DeepSORT-Object-Tracking

Este comando permite moverse entre directorios del sistema y da la oportunidad de especificar la ruta a la que se desea acceder.

Al especificar la ruta con el comando, Pycharm notifica con un mensaje desde la terminal la ruta donde se trabaja.

Para el correcto funcionamiento de Yolov8 con DeepSORT este necesita de librerías específicas. Para acceder e instalar las bibliotecas que se usan en el proyecto se usa el siguiente comando:

pip install -e '.[dev]'

Como se muestra en la figura 3 este comando permite instalar de manera sencilla una librería o en su defecto un paquete de librerías.

Figura 3

Instalación de las bibliotecas

Nota: Terminal de pycharm con el comando de instalacion para las librerias. Fuente: Los autores.

Terminada la descarga de las bibliotecas se debe configurar el directorio para el proyecto, aquí es donde se encuentra la codificación necesaria para que Yolo con DeepSORT funcione. Para configurar el directorio para el proyecto se usa la siguiente línea de comando:

cd ultralytics/yolo/v8/detect

Para verificar el correcto funcionamiento de Yolov8 con DeepSORT se necesita de una fuente de video, para ello se obtiene un video desde la red. Para descargar un video de muestra desde Google drive con Pycharm se usa la siguiente línea de comando:

gdown "https://drive.google.com/uc?id=1rjBn8Fl1E_9d0EMVtL24S9aNQOJAveR5&confirm=t"

Este comando permite bajar archivos o carpetas desde los servidores de Google para poder ser usados desde Pycharm.

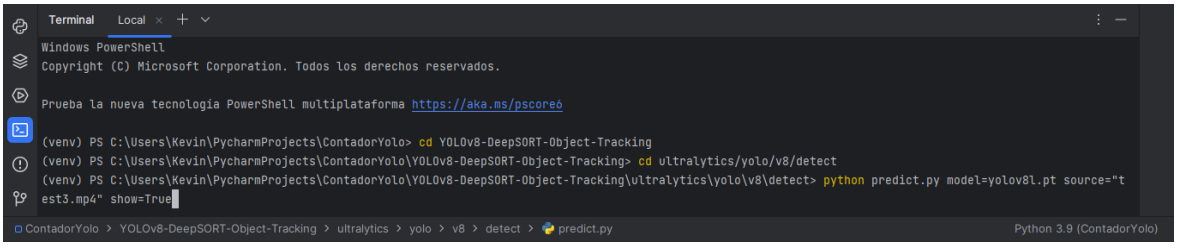
En la figura 4 se puede observar terminada la descarga del video de ejemplo pycharm indica el proceso, el peso y la ubicación de este.

Se procede a validar el funcionamiento de Yolov8 con DeepSORT. Para esto se debe especificar el archivo.py, el modelo de Yolo, la fuente de análisis y activar la vista de ventana. Para ejecutar Yolov8 con DeepSORT sobre el video se usa la siguiente línea de comando, como se muestra en la figura 4.

python predict.py model=yolov8l.pt source="test3.mp4" show=True

Figura 4

Especificaciones para Yolov8 con DeepSORT

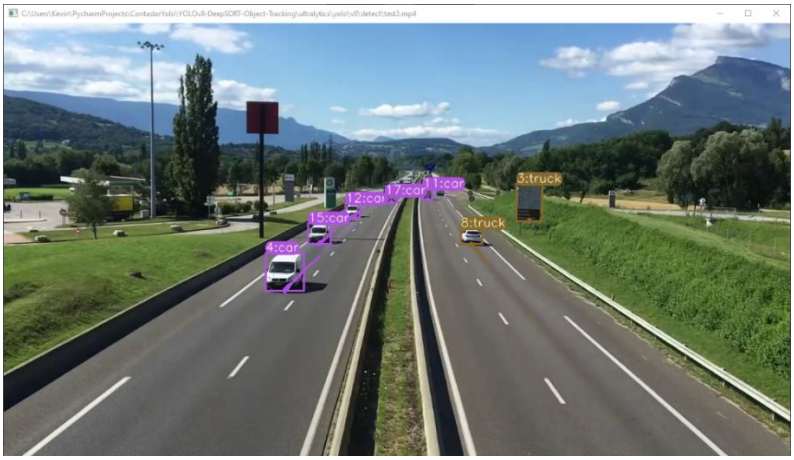


Nota: Terminal de pycharm con el comando para las especificaciones de Yolov8. Fuente: Los autores.

En la figura 5 se observa el correcto funcionamiento de Yolov8 con DeepSORT. Yolo detecta los automóviles en tiempo real en todo momento mientras que DeepSORT se encarga de seguir el movimiento del objeto.

Figura 5

Yolov8 con DeepSORT sobre el video de prueba

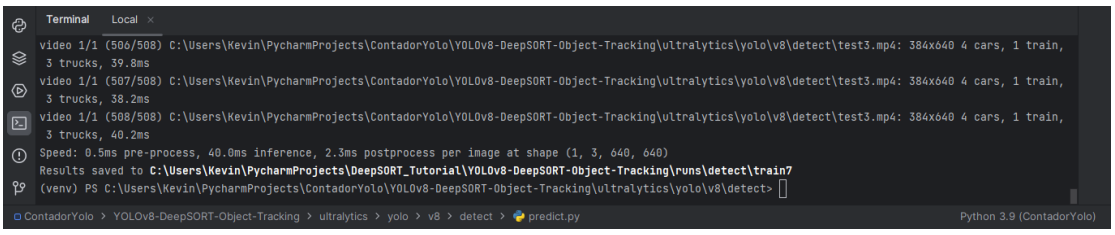


Nota: Yolov8 detectando los autos en el video de prueba. Fuente:Los autores.

Como se contempla en la figura 6, al finalizar el video de prueba la ventana de presentación se cierra automáticamente y Pycharm entrega un mensaje en la terminal donde se describe la resolución del video, los objetos que logro detectar, los tiempos de procesamiento y la ruta donde se guardan los resultados.

Figura 6

Descripción del video de prueba

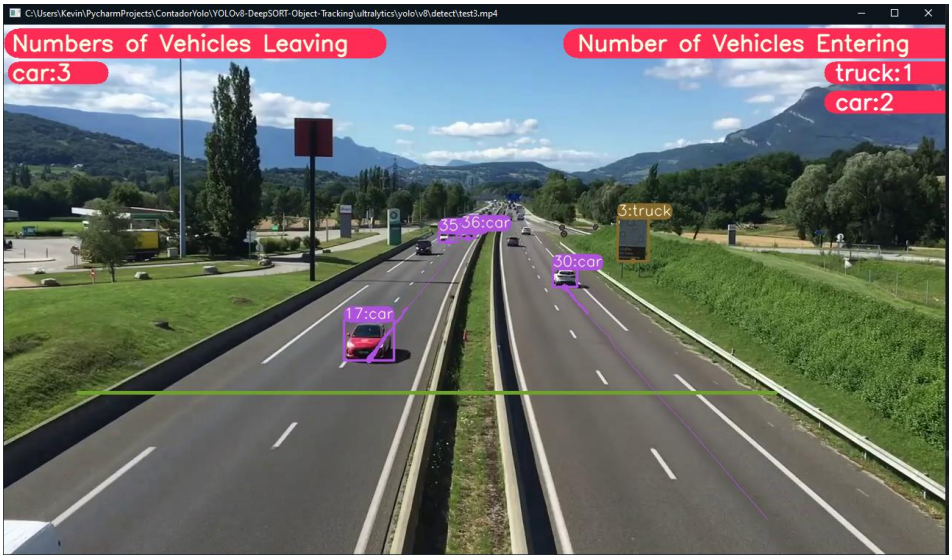


Nota: Terminal de pycharm con la ruta de guardado del video. Fuente: Los autores.

El código cuenta con una versión con contador como se ve en la figura 7, el cual será la base para iniciar con el proceso de modelar según lo planteado anteriormente en el objetivo uno.

Figura 7

Yolov8 con DeepSORT y contador sobre el video de prueba



Nota: Acceso al repositorio de Github completo. Fuente: Los autores.

5.1.2 Escoger la fuente de imagen donde trabajara la inteligencia artificial

A continuación, se escoge la fuente de imagen o video al cual se le implementara la inteligencia artificial, por motivos prácticos se trabaja con un video de la intersección de las calles Av. Domingo Comín y Padre Ignacio Rueda Latasa donde se planteó la problemática.

Para ejecutar la inteligencia artificial sobre el video escogido se usa el siguiente comando en la terminal de Pycharm:

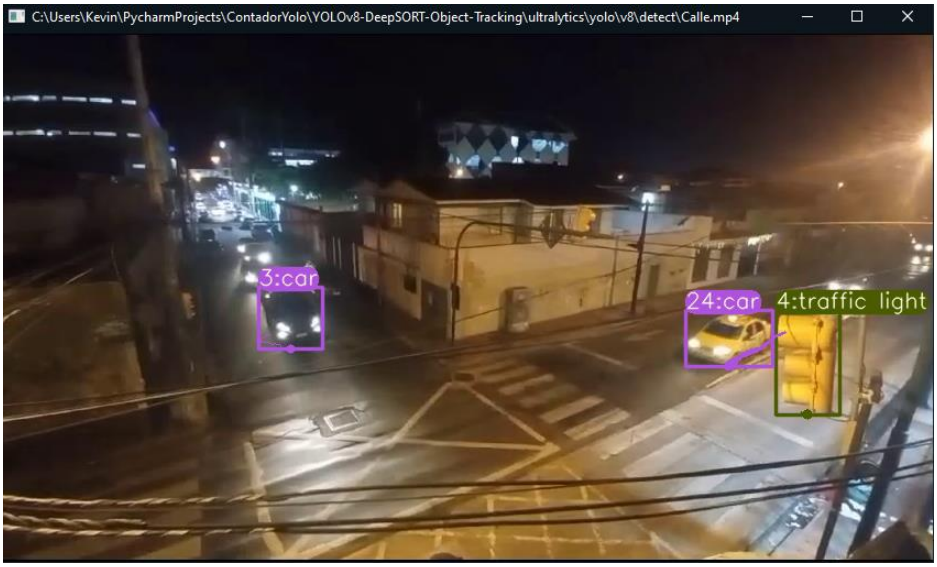
python predict.py model=yolov8l.pt source="Calle.mp4" show=True

El cambio se realiza en la categoría de source donde se coloca la el nombre de la imagen o video junto a su respectiva extensión.

Al ejecutar el comando se levanta nuevamente la ventana de video donde Yolov8 con Deepsort se encuentran trabajando, como se puede ver en la figura 8.

Figura 8

Yolov8 con DeepSORT sobre el video de la intersección



Nota:Intersección propuesta en la problemática. Fuente:Los autores.

5.1.3 Ajustar los parámetros respectivos para el correcto funcionamiento

Yolov8 al ser un modelo para la detección de objetos en tiempo real, necesita de datasets que son una colección de datos alojadas dentro del equipo en el que se ejecuta, en este caso se trabaja con el dataset COCO128 el cual incorpora un gran apartado visual de imágenes de todo tipo, para un mejor orden se las clasifican en diferentes clases.

Para el presente trabajo se utiliza la clase número 2 la cual está asociada con autos, esto con la finalidad de que Yolov8 solo detecte autos y ningún otro objeto en la imagen o video.

Otro parámetro que se debe tomar en cuenta y modificar es el área de conteo dentro del video, para esto se cambia de posición la línea original de la programación la cual servirá para el conteo de la calle principal y se crea otra línea para el conteo de la calle secundaria como se puede apreciar en la figura 9.

Figura 9

Yolov8 con DeepSORT y contador sobre el video de la intersección

Nota: Implementación de la IA completa. Fuente: Los autores.

5.1.4 Crear registro tipo base de datos

Adicionalmente, esta base de datos servirá como un registro, para esto se usa la base de datos PosgreSQL la cual contara con una tabla llamada contador_datos donde se almacenará los datos de los contadores cada vez que se actualice el valor del contador.

Para realizar la conexión con la base de datos desde Python se utiliza la librería psycopg2 la cual permite crear, eliminar, editar e insertar datos desde Pycharm, de igual forma permite realizar consultas tipo SQL.

En este caso la tabla ya se encuentra creada, la cual se puede observar en la figura 10.

Figura 10

Tabla en la base de datos

	id [PK] integer	objeto_nombre character varying (255)	objeto_tipo character varying (255)	contador_secundaria integer	contador_principal integer
1	2342785	car	Calle Principal	[null]	8
2	2341351	car	Calle Secundaria	2	[null]

Nota: Tabla generada por la base de datos PostgreSQL. Fuente: Los autores.

Para acceder a la base de datos se necesita del host, nombre de la base de datos, nombre de usuario y contraseña, estas credenciales se encuentran especificadas en el código de programación al momento de hacer la conexión con la base de datos.

5.2 Diseño de una tarjeta de interfaz electrónica para la comunicación entre el sistema embebido y el controlador LC2

Para cumplir con el segundo objetivo específico, el cual consiste en el diseño de una tarjeta de interfaz electrónica para la comunicación entre el sistema embebido y el controlador LC2, se realizan los siguientes pasos.

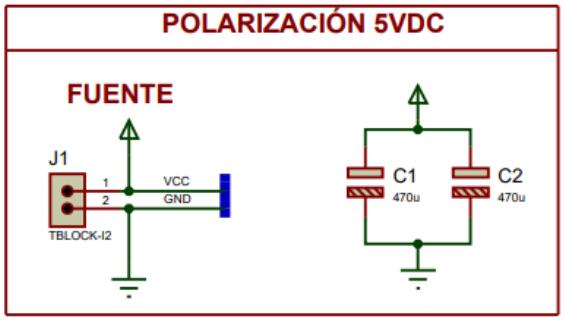
5.2.1 Realizar el circuito esquemático de la tarjeta electrónica

Dicho circuito sirve para realizar la simulación de la comunicación viendo los resultados en una pantalla LCD. Asegurando de esta forma el correcto funcionamiento de la tarjeta que se creó, con los componentes que tiene el controlador LC2.

El circuito consta de una entrada de alimentación externa de 5v a través de la bornera J1. Este voltaje alimenta a los dispositivos que requieren 5v tales como: el microcontrolador, el LCD de 20x4 y módulo de tiempo real DS1307. La etapa de la fuente de poder posee dos condensadores de 470µf, esto con la finalidad de ayudar a tener un voltaje de 5v continuos sin rizados de voltajes que afecten el sistema, estas indicaciones se aprecian en la figura 11.

Figura 11

Fuente de polarización 5 VDC

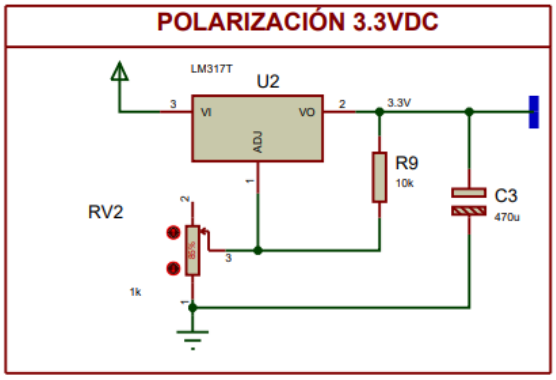


Nota: Conexiones necesarias para el funcionamiento del circuito utilizado para la simulación de comunicación serial y para realizar la tarjeta PCB. Fuente: Los autores.

Adicional se encuentra un regulador de voltaje ajustable llamado LM317T (U2), que permite obtener un voltaje en su salida desde 1,2v hasta 37v mediante un potenciómetro (RV2). Por lo cual, se lo usa para suministrar los 3.3v que necesita el módulo ESP32 para su alimentación. El condensador de 470µf también cumple la función de evitar los rizos de voltaje indeseados, como se muestra en la siguiente figura 12.

Figura 12

Fuente de polarización 3.3 VDC

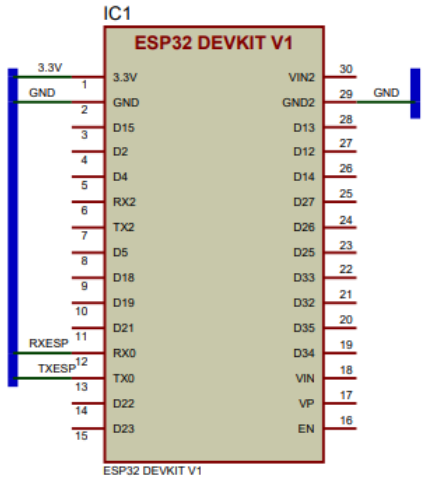


Nota: Conexiones necesarias para el funcionamiento del circuito. Fuente: Los autores.

Mediante los pines TX y RX del ESP32 (esclavo), se comunicará de manera cruzada con los pines del micro 18f5450, como se aprecia en la figura 13. Se busca que el ESP32 programado como esclavo se enlace con el protocolo de comunicación ESP-now con otro ESP32 programado como maestro. Este último recibe los datos del contador por comunicación serial desde el Yolo v8 que se encuentra ejecutándose en la programación de Python.

Figura 13

Módulo ESP32



Nota:Conexiones necesarias para el funcionamiento del circuito utilizado para la simulación de comunicación serial y para realizar la tarjeta PCB. Fuente: Los autores.

En el software de Proteus no existe la librería para el módulo ESP32 que se necesita para la realización de este proyecto, por tal motivo se realizó la librería del mencionado elemento en el componente ISIS del Proteus, tal como se puede apreciar en la fig. 13.

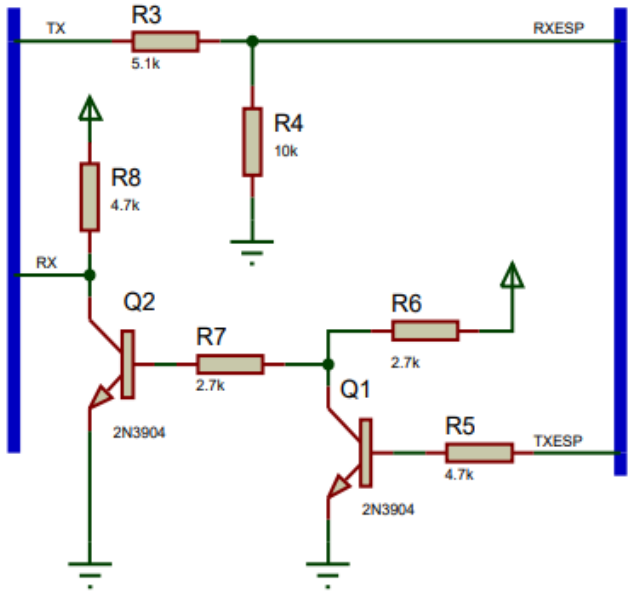
Dado que el micro se polariza con 5v y el módulo ESP32 se polariza con 3,3v es necesario implementar un circuito de interfase para poder comunicar a ambos dispositivos.

El micro a través de su pin TX envía 0v para el “0” lógico y 5v para el “1” lógico, los cuales deben reducirse para ingresar al pin RX del módulo ESP32 a 0v para el “0” lógico y 3,3v para el “1” lógico, esto se hace con el objetivo de evitar daños en el módulo wifi. Para esto se utilizó un divisor de voltaje con dos resistencias (R3 y R4) con valores de 5,1kΩ y 10kΩ, respectivamente como se muestra en la figura 14.

En cambio, para obtener una compatibilidad lógica entre la señal TX del ESP32 y la señal RX del micro se necesita elevar el voltaje de 3,3v correspondiente al “1” lógico del módulo wifi a 5v correspondiente al “1” lógico del micro, esto se consigue mediante dos transistores (Q1 y Q2) NPN 2N3904 y cuatro resistencias (R5, R6, R7 y R8) con valores descritos en la figura 14.

Figura 14

Circuito de adaptación de voltaje para comunicación serial



Nota: Conexiones necesarias para el funcionamiento del circuito utilizado para la simulación de comunicación serial y para realizar la tarjeta PCB. Fuente: Los autores.

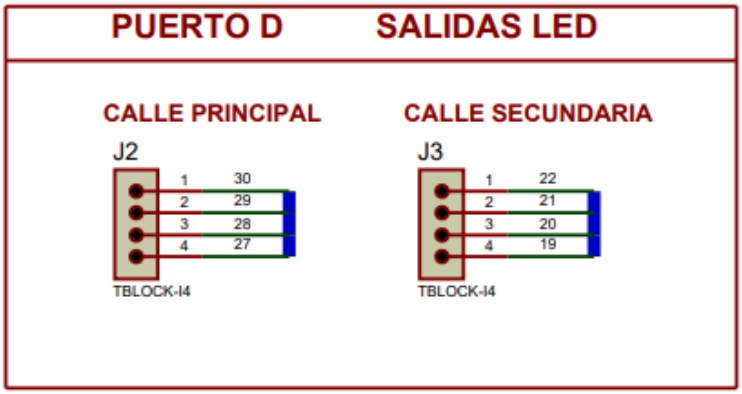
El circuito de la figura anterior funciona de la siguiente manera, al tener 0v en la salida del pin Tx del ESP32 se induce al transistor Q1a trabajar en la zona de Corte, con lo cual no hay corrientes de base colector ni emisor; con lo que R6 y R7 entrarán a saturar el transistor Q2 obteniendo 0V en el colector y por ende “0” lógico en el pin Rx.

De la misma manera cuando en la salida del pin Tx del ESP32 se envía 3,3v se induce al transistor Q1a trabajar en la zona de saturación, obteniendo 0v en su colector y con esto se lleva a zona de corte con corrientes nulas en el transistor Q2, por lo que al pin Rx del micro se llegará 5v a través de la resistencia R8 que se encuentra conectada en configuración de PULL UP.

En las borneras J2 y J3 conectaremos las salidas opto acopladas las cuales permiten el funcionamiento de las luces del semáforo LC2. Estas salidas provienen del puerto D del microcontrolador.

Figura 15

Salidas opto acopladas

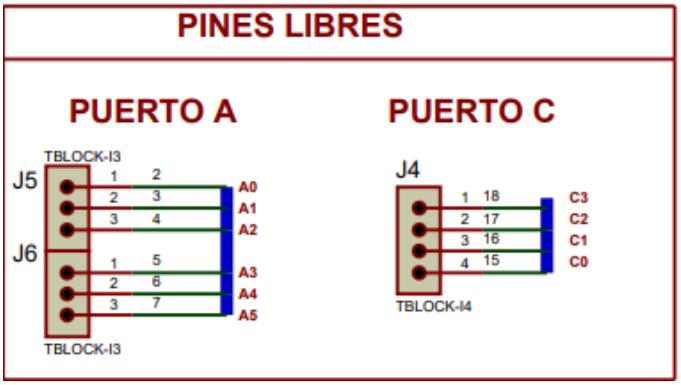


Nota: Conexiones necesarias para el funcionamiento del circuito utilizado para la simulación de comunicación serial y para realizar la tarjeta PCB. Fuente: Los autores.

En las borneras J4 del puerto C, los pines del C0 al C3, y en las borneras J5 y J6 correspondientes al puerto A se tiene pines disponibles para cualquier otra actividad que se quiera hacer dependiendo de las necesidades y la programación del microcontrolador.

Figura 16

Pines disponibles

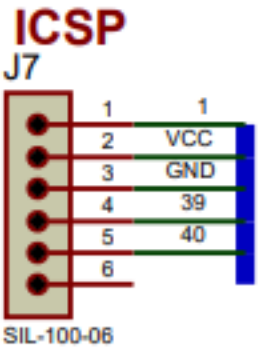


Nota: Conexiones necesarias para el funcionamiento del circuito. Fuente: Los autores.

La bornera J7 permite la programación de micro sin necesidad de sacarlo de la tarjeta mediante protocolo ICSP.

Figura 17

Bornera para el protocolo ICSP

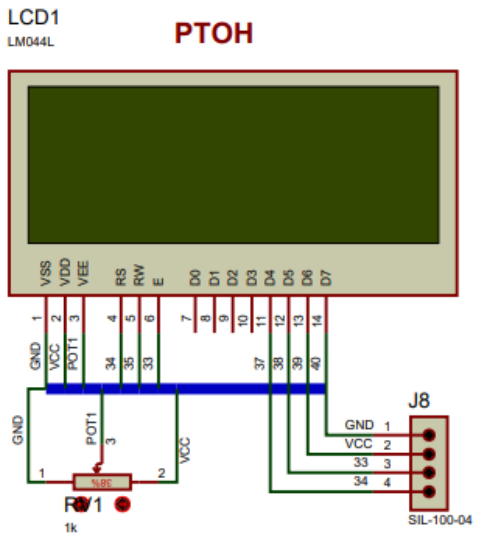


Nota: Conexiones necesarias para el funcionamiento del circuito. Fuente: Los autores.

La bornera J8 tiene la función de conectar el micro con el LCD si este tuviera un módulo de comunicación serie. De no tener este módulo se deberá soldar los 16 pines del LCD a la tarjeta principal. A su vez, podemos observar un potenciómetro para regular el contraste en el caso de que el LCD tenga comunicación paralela con el micro.

Figura 18

Conexiones para la pantalla LCD



Nota: Conexiones necesarias para el funcionamiento del circuito. Fuente: Los autores.

El DS1307 es un módulo de reloj de tiempo real que se comunica con el microcontrolador a través de un protocolo de comunicación serie sincrónico llamado I2C, que utiliza dos pines uno de datos y otro de clock.

Este módulo es necesario para obtener la hora en el equipo y poder trabajar en los diferentes planes que se configuran en el controlador de tránsito LC2.

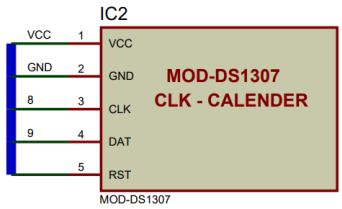
Este módulo cuenta además con una batería incorporada para mantener la hora y la fecha, aunque el suministro de energía exterior se haya interrumpido, el módulo posee internamente un circuito de transferencia automática de energía de tal manera que la pila sólo es utilizada cuando

la fuente externa se ve suspendida, garantizando de esta manera una vida útil más larga para esta batería de tan solo 3v.

A continuación, se muestra en la figura 19 las conexiones necesarias para el funcionamiento del DS1307.

Figura 19

Módulo clock.



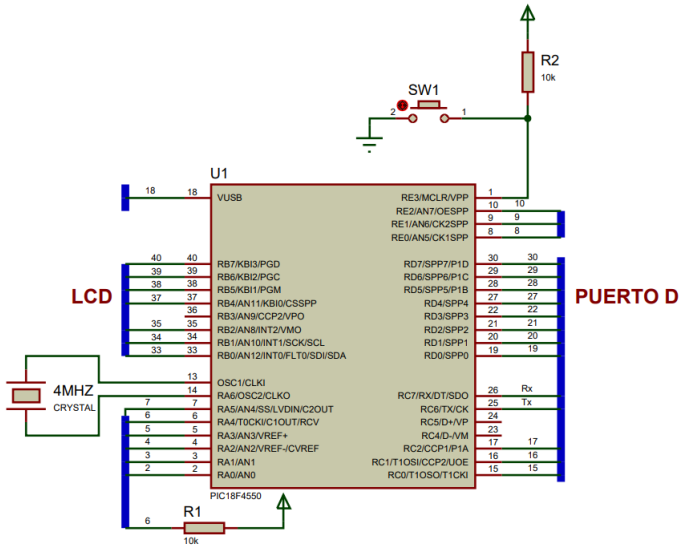
Nota: Conexiones necesarias para el funcionamiento del circuito. Fuente: Los autores.

El microprocesador usado para este trabajo de titulación es el 18F4550, que se muestra en la siguiente figura 20, el cual posee cinco puertos de entradas y salidas numerados del puerto A al puerto E. El micro tiene un cristal de cuarzo de 20MHZ para generar los pulsos de reloj necesarios para su correcto funcionamiento.

En este trabajo en el puerto A se encuentran pines libres, el puerto B se utiliza para las conexiones de la pantalla matricial LCD, el puerto C se encuentra los pines para la comunicación serial en RC7 el pin RX y en RC6 el pin TX, el puerto D se usa para las salidas led que representa a los semáforos tanto para la calle principal como la calle secundaria y en el puerto E se localiza el módulo DS1307. Este microprocesador.

Figura 20

Conexiones para micro 18F4550



Nota: Conexiones necesarias para el funcionamiento del circuito. Fuente: Los autores.

5.2.2 Realizar el circuito PCB de la tarjeta electrónica.

En el componente ARES de Proteus tampoco existe la librería para el módulo ESP32 que se necesita para la realización de este proyecto, por tal motivo se realizó la librería del mencionado elemento, tal como se puede apreciar en la figura 21.

En este circuito solo se encuentra las conexiones que necesita el ESP32 para su funcionamiento tanto para el que está programado en modo esclavo como el que es maestro.

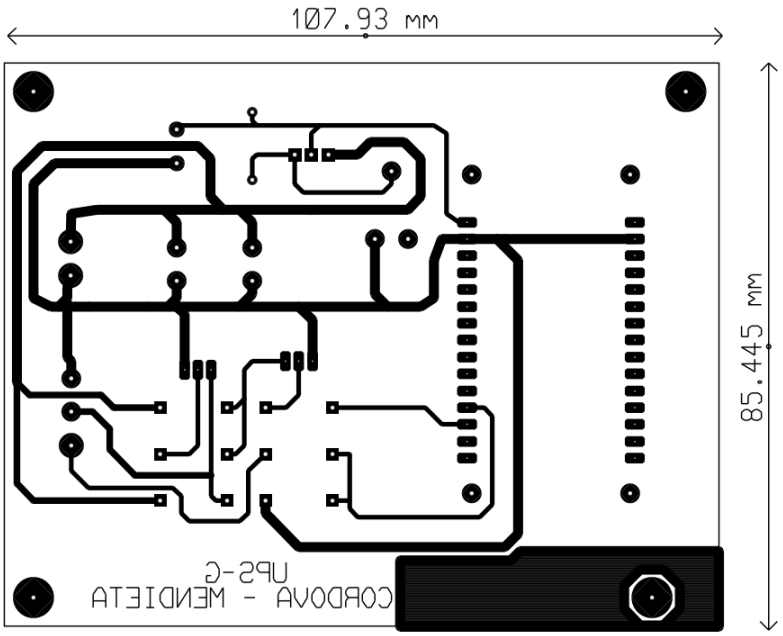
Las conexiones que necesita son para su alimentación de 3.3v, GND y comunicación serie con RX y TX, previamente explicado.

La dimensión de la tarjeta PCB es de 107,93mm de ancho y 85,44mm de largo, es una tarjeta de una sola capa, con elementos through hole, con mascara antisolder, el ancho de las

pistas para las señales de control es de 30 mils, mientras que para las pistas de poder su ancho es de 60 mils, como se puede reconocer en la figura 21.

Figura 21

Características de la tarjeta PCB



Nota: Vista de la capa de cobre y de la serigrafía de los elementos. Fuente:Los autores.

La tarjeta consta de una bornera de dos polos para ingresar el voltaje de polarización que es de 5vdc. Una segunda bornera de tres polos en donde se ingresa la entrada y salida de la comunicación llamada RX y TX y la señal GND del controlador, recordando que todas las GND deben estar referenciadas al mismo punto.

También contiene el regulador ajustable LM317T al cual ingresan los 5v y regula a 3,3v con ayuda de un potenciómetro y una resistencia de acuerdo a la siguiente fórmula:

$$V_{salida} = V_{out} = \left(1 + \frac{R2}{R1}\right)$$

(Electrónica Unicrom, s.f.)

Dada que la ecuación de voltaje de salida involucra dos variables (R1 y R2), se debe asumir un valor para R1, en este caso será de 1kΩ y despejar de la fórmula el valor de R2, para conseguir voltaje de 3,3v.

$$3{,}3v = \left(1 + \frac{R2}{1\text{k}\Omega}\right)$$

$$R2 = (3{,}3v - 1) * 1k\Omega = 2{,}3k\Omega$$

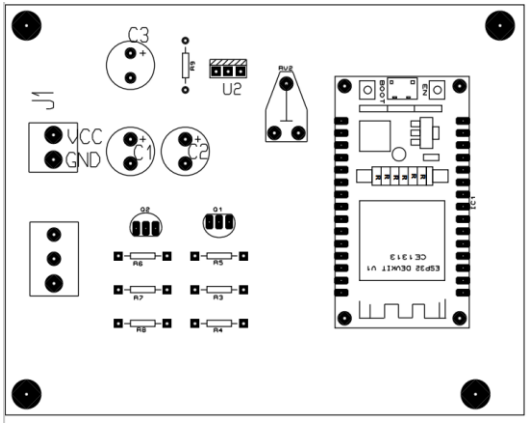
Ya que este valor no es comercial se opta por utilizar un potenciómetro de 5kΩ y ajustarlo a los 2,3kΩ que es lo que se requiere.

Para mantener un voltaje constante libre de rizados y fluctuaciones en la fuente de 5v se adjuntan dos condensadores polarizados (C1 y C2) de 470uf a 16v y para la fuente de 3,3v otro condensador (C3) de 470uf a 16v para el mismo propósito.

Como se mencionó anteriormente, para obtener una compatibilidad lógica entre el micro 18f4550 y el ESP32 se utilizan las resistencias (R3, R4, R5, R6, R7 y R8) y los dos transistores (Q1 y Q2), como se identifican en la siguiente figura 22.

Figura 22

Vista de componentes de la tarjeta PCB



Nota Vista de superior de componentes. Fuente: Los autores.

5.3 Visualizar el cambio de los tiempos en las luces de los semáforos.

5.3.1 Conexión de la comunicación serial entre Python con el ESP32 Maestro

Para habilitar la comunicación serial se utiliza la librería pyserial, esta librería permite abrir un puerto serial en Python con la extensión serial.Serial en la que se debe especificar el puerto por el cual se enviarán los datos y la velocidad de comunicación.

Esta comunicación ayudará a enviar los datos tipo string del contador a un EspS32 que servirá como maestro en el protocolo de comunicación ESP NOW.

5.3.2 Conexión del ESP32 maestro con el ESP32 esclavo mediante protocolo de comunicación ESP-NOW.

En este trabajo de titulación se utiliza la comunicación half dúplex, debido a que el ESP32 programado como maestro es capaz de recibir información del conteo vehicular y enviarla al ESP32 programado como esclavo, pero esta comunicación no ocurre simultáneamente. Así mismo, el ESP32 esclavo recibe esta información y la envía al microcontrolador del semáforo.

El código necesario para el funcionamiento del ESP32 como maestro es el siguiente:

Al inicio es necesario incluir las librerías del protocolo ESP-now, las librerías para el wifi en el ESP32 y definir el canal que se va a usar.

En la función void ScanForSlave() se buscan dispositivos que en su SSID (Service Set Identifier), empiecen con la cadena “slave”, para procesarlos a continuación. Extrayendo la información que se encuentra en el BSSID (Basic Service Set Identifier), útil para la conexión y comunicación cuando se usa el protocolo ESP-NOW. Para así encontrar la dirección MAC (Media Access Control) única asociada al esclavo.

En el bool manageSlave () se chequea que el esclavo esté emparejado al maestro, caso contrario usa esp_now_add_peer() para emparejarlo. Si la condición DELETEBEFOREPAIR es verdadero se elimina el enlace que existe antes de enlazar nuevamente.

En la función void onDataReceived() para la recepción de datos por comunicación serial, se comprueba si existen datos en el puerto serial con Serial.available(), en el caso de que existan datos con la ayuda de Serial.read() se puede leer un byte de los mismos. Almacenándolos en la variable receivedData.

En la función loop() se busca constantemente un esclavo para emparejarse y llama a la función que controla los datos recibidos por el puerto serial.

Con la función OnDataSent() se puede observar la información que se envió al ESP32 esclavo. Finalmente, con el comando esp_now_send() en la función sendDataToSlave() se envía el dato al ESP32 esclavo.

Seguidamente, se explica la programación necesaria en el ESP32 esclavo.

Se deben incluir las mismas librerías que en el ESP32 maestro y definir el mismo canal. Con la función InitESPNow() se inicializa la comunicación, que cuando presente fallos se va a reiniciar mediante el comando ESP.restart().

En la función cofigDeviceAP () con el comando WiFi.softAP se crea un punto de acceso AP, se define un SSID que en su cadena empiece con “slave” y una contraseña.

Se inicializa el puerto serial con Serial.begin(9600), esta velocidad en baulios se usa para ambos ESP32. Con el comando esp_now_register_recv_cb () se recoge los datos que reciba la función OnDataRecv.

Cuando se recibe una trama de datos del ESP32 maestro, es cuando la función OnDataRecv se ejecuta. En esta función es donde se encuentra un led que representa a la

adquisición de datos. También se almacena en el array llamado valor cuando los datos corresponden a valores numéricos, los cuales se envían como una cadena de datos por puerto serial con la cabecera CM y a continuación los tres dígitos recibidos. Por ejemplo, enviará por puerto serial “CM005”, “CM012”, entre otros.

5.3.3 Fórmula para el tiempo óptimo de luz verde.

Para identificar el tiempo óptimo de luz verde en el ciclo se puede utilizar el método de Webster en donde el tiempo de ciclo óptimo va a ser igual a 1,5 por el tiempo total perdido por ciclo más 5 en donde esto va a ser dividido por 1 menos la sumatoria del flujo de saturación de la intersección (Alba & Hernández, 2013).

Expresado en la siguiente fórmula:

$$Tco = \frac{1{,}5L + 5}{1 - \sum_{i=1}^{\Phi} Si}$$

Donde:

Tco: Tiempo óptimo de ciclo (s).

L: Tiempo total perdido por ciclo (s).

Si: Índice de saturación.

Φ: Número de fases.

El tiempo total perdido por ciclo (L) es igual a la suma de tiempo en que se realiza un cambio en amarillo, a este tiempo también se considera como tiempos Ámbar con una duración de tres a cuatro segundos, y cuando el semáforo se encuentra en rojo, llamado tiempo de despeje con una duración de un segundo, esto será multiplicado por el número de fases. (Alba & Hernández, 2013). Por lo tanto:

$$L = (Tiempo\ Ámbar + Tiempo\ de\ despeje) * Número\ de\ fase$$

$$L = (3seg + 1seg) * 2$$

$$L = 8seg$$

Para encontrar el índice de saturación (Si) es necesario contar con el flujo de carros que no giran por hora (q). El valor de q es igual a la cantidad de carros que circulan por esa calle por el factor de equivalencia que será dividido por el número de carriles que tiene la calle.

$$q = \frac{Cantidad\ de\ carros * Factor\ de\ equivalencia}{Número\ de\ carriles}$$

Donde:

Factor de equivalencia: en carros es igual a uno.

El valor de q se dividirá por el flujo de saturación, que representa la máxima cantidad de tráfico que fluye por el o los carriles cuando el semáforo está en verde durante un intervalo de una hora, este valor corresponde a 1800. El índice de saturación se debe calcular para cada movimiento que se realiza (Alba & Hernández, 2013).

$$Si = \frac{q}{Flujo\ de\ saturación}$$

Según la problemática de este trabajo de titulación, el resultado del tiempo óptimo de verde para cada intersección debe encontrarse de una forma más rápida. Si se obtiene la cantidad de vehículos en un periodo de una hora, al momento de desarrollar las fórmulas con esa información ya no va existir el tráfico que se quiere mejorar.

Por lo tanto:

$$Si = \frac{q}{\frac{Flujo\ de\ saturación}{6}}$$

$$Si = \frac{q}{\frac{1800}{6}}$$

$$Si = \frac{q}{300}$$

Donde:

Flujo de saturación es dividido por seis para coincidir con los datos que envía el Python dentro de un tiempo de 10 min.

Por lo tanto:

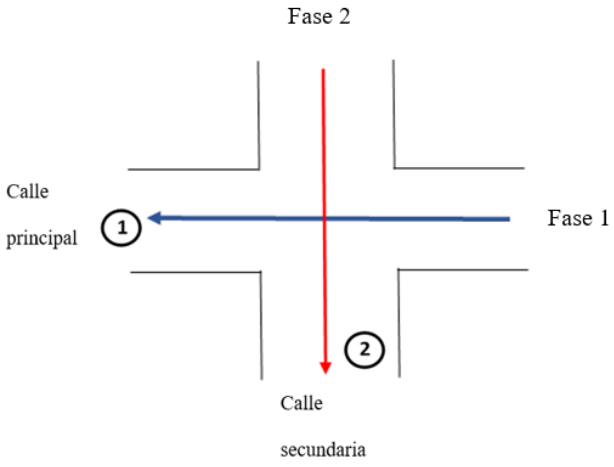
$$Si1 = \frac{q1}{300}$$

$$Si2 = \frac{q2}{300}$$

El índice de saturación se debe calcular para todos los movimientos que se encuentran en esa fase, en este caso el número de fase corresponde al número de movimientos, ya que en la fase uno se realiza el movimiento uno y en la fase dos se realiza el movimiento dos como se muestra en la figura 23.

Figura 23

Movimiento y fases de la intersección



Nota: Sentido de las vías involucradas en la problemática. Fuente: Los autores.

La suma entre Si1 y Si2 dará S que corresponde a la sumatoria del índice de saturación necesaria para encontrar el tiempo óptimo de ciclo. Con los datos encontrados se realiza la fórmula para obtener el tiempo de luz verde óptimo.

$$Tv1 = \left(\frac{Si1}{S}\right) * (Tco - L)$$

$$Tv2 = \left(\frac{Si2}{S}\right) * (Tco - L)$$

5.3.4 Ingresar en el microcontrolador las fórmulas del método Webster con sus respectivas variables para modificar el tiempo de luz verde.

A continuación, se detallará las funciones principales, que se llaman en el void main dentro de la programación del pic, que permite el funcionamiento del controlador hasta la comunicación serial con el ESP32.

Con la primera función void presentación (void), se visualiza en el LCD la presentación del equipo con el que se va a trabajar, en este caso es el controlador de tránsito programable CSP/LC2.

En la función void configuración (void), se determina un tiempo para decidir si se arranca con el funcionamiento del controlador o se ingresa a un sub-menú de configuración.

En la siguiente función void configura2(void), se presentan varias acciones a seguir para configurar el correcto funcionamiento del controlador. Las opciones presentadas son:

1. Reloj presionando en el teclado matricial1.
2. Tiempos presionando en el teclado matricial 2.
3. Planes presionando en el teclado matricial 3.
4. Salir presionando en el teclado matricial 4 u otros valores.

La función int teclado (int), permite el ingreso de valores numéricos, mediante el teclado matricial, para diferentes funciones que requieren datos como fecha, hora, planes, entre otros.

Con la función void reloj (void), le permite al técnico operador ingresar fecha y hora para un adecuado funcionamiento del LC2.

A través de la función void planes (void), se permite determinar la cantidad de planes de trabajaos según la hora del día, en este proyecto se tienen dos planes trabajando, el normal y el de flasheo. Aquí se llama a la función ing_hor para determinar la hora de inicio de cada uno.

En el void tiempos (void), está la función que permite ingresar valores de:

1. Tiempo de ciclo del semáforo de las dos fases involucradas.
2. Tiempo del semáforo amarillo.
3. Tiempo asignado al rojo en ambas fases, también llamado tiempo de despeje.
4. Tiempo de encendido de la luz verde de la fase principal.
5. Tiempo de encendido de la luz verde de la fase secundaria.

En la función int valor (int a, int b), esta función recibe dos valores enteros correspondientes a dos códigos ASCII numéricos provenientes de la interrupción por comunicación serial asíncrona, los transforma a valores reales de 00 hasta 99 para poder analizarlas y presentarlas.

Con la función void retardo (void), se recibe un valor entero que representa al tiempo que durará cada estado del semáforo, además permite visualizar en el LCD el tiempo asignado y el tiempo transcurrido.

USART que tiene el micro controlador, es decir, se utiliza interrupción para la comunicación o también la llamada comunicación por Hardware, la cual consiste en utilizar los pines asignados por el fabricante para Tx y para Rx estos son los pines C6 y C7 respectivamente.

La captura de los datos que vienen de manera serial desde el ESP32 esclavo se la realizó mediante la instrucción scanf, la cual permite obtener una cadena de caracteres hasta que se termine con un Enter, esta cadena se guarda en una variable tipo arreglo de tamaño 5 donde el primer y el último dato son letras que identifican si la intersección es la principal o la secundaria dependiendo si las letras son AB o CD.

El formato de recepción entonces es el siguiente:

AxxxB para la intersección principal y CxxxD para la intersección secundaria, entendiéndose que xxx corresponde a la cantidad de vehículos que detecto la inteligencia artificial.

Luego de recibir la información se procede a preguntar si las letras recibidas en la primera y última posición corresponden a AB, de ser así los números que se encuentran en tipo string almacenados dentro de la variable data se guardan en un arreglo val1 para transformarlo a un número entero. Igual procedimiento se realiza si el dato que llega se encuentra entre las cabeceras CD, guardándose en otro arreglo llamado val2.

Estos dos valores corresponden a la cantidad de vehículos que circulan por dicha intersección y que son ingresados en la fórmula para determinar la cantidad de tiempo de cada luz verde.

VI RESULTADOS

6.1 Identificar los vehículos y contabilizarlos mediante la inteligencia artificial.

Los datos que se envían por comunicación serial son los contadores de 3 dígitos de cada calle, los cuales al momento de ser enviados se les agrega un encabezado como se ve en la figura 24 que pertenece al contador de la calle principal y la figura 25 que pertenece al contador de la calle secundaria para ser identificados de mejor manera por el PIC luego de pasar por los ESP32 mediante el protocolo de comunicación ESP NOW.

Figura 24

Contador de la calle principal enviado por comunicación serial

```
video 1/1 (26/1514) C:\Users\Kevin\PycharmProjects\(
Dato enviado por comunicacion serial:A001B
video 1/1 (27/1514) C:\Users\Kevin\PycharmProjects\(
video 1/1 (28/1514) C:\Users\Kevin\PycharmProjects\(
```

Nota: Fotogramas del video analizados por la IA. Fuente: Los autores.

Figura 25

Contador de la calle secundaria enviado por comunicación serial

```
video 1/1 (471/1514) C:\Users\Kevin\PycharmProjects\(
Dato enviado por comunicacion serial:C001D
video 1/1 (472/1514) C:\Users\Kevin\PycharmProjects\(
video 1/1 (473/1514) C:\Users\Kevin\PycharmProjects\(
```

Nota: Fotogramas del video analizados por la IA. Fuente: Los autores.

6.2 Reconocer la correcta comunicación entre la tarjeta de interfase y el LC2

En la siguiente figura número 26 se muestra la tarjeta de interfase que se realizó para la comunicación con el LC2, donde se obtienen los correctos voltajes de 0 y 5v en la bornera que se conecta con el LC2 también 0 y 3,3v para el lado del ESP32.

Con dicha tarjeta se realizaron las pruebas de funcionamiento en donde se logra obtener los datos del programa Hércules, que simula los datos enviados desde Python, mostrándolos en la pantalla LCD del microcontrolador LC2.

Figura 26

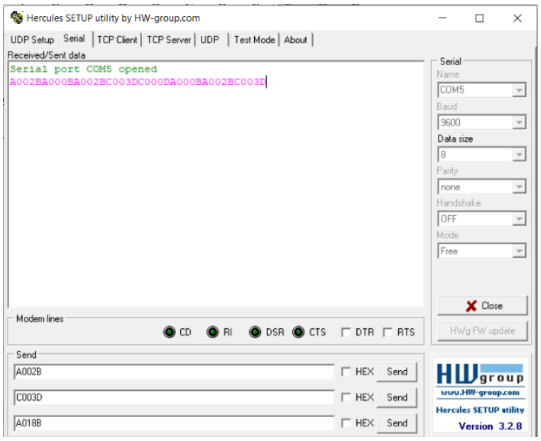
Envío de datos desde el Hércules al LC2

Nota:Trama enviada hacia el LC2 desde el Hércules. Fuente:Los autores.

Desde el Hércules se le envía la cadena de datos A002B y luego se envía la cadena de datos C003D como se muestra en la siguiente figura 27.

Figura 27

Envío de cadena de datos desde el Hércules.



Nota:Pantalla correspondiente al software Hércules. Fuente:Los autores.

El dato que le llega se muestra en la pantalla LCD como un aumento del período dependiendo de la cabecera que se envíe, siendo A002B para la calle principal, como se muestra en la figura 28.

Figura 28

Recepción de cadena de datos calle principal

Nota:Comunicación completa entre los ESP32, el LC2 y Hércules. Fuente:Los autores.

Cuando se envíe desde el Hércules la cadena C003D que corresponde a la calle secundaria, se cambiará el período de dicha calle.

Figura 29

Recepción de cadena de datos calle secundaria

.

Nota:Comunicación completa entre los ESP32, el LC2 y Hércules. Fuente:Los autores.

6.3 Identificar el correcto funcionamiento al visualizar el cambio de los tiempos en las luces de los semáforos

Para cumplir con el tercer objetivo de identificar el correcto funcionamiento al visualizar el cambio de los tiempos en las luces de los semáforos se realizaron simulaciones con el ide Pychram de Python, con el ide de Arduino para los esp32 que se comunican con el protocolo ESP-now configurados como esclavo/maestro y con el software de proteus (ISIS).

Para estas pruebas fue necesaria que la conexión de la comunicación serial entre el ESP32 esclavo con el Pic 18F4550 estuviera funcionando correctamente.

La cantidad de vehículos se ingresan manualmente a una hoja de Excel en la columna de #carros para encontrar el resultado de tiempo óptimo de verde para ambas calles. Esta tabla se genera con el objetivo de realizar una comparación entre los datos de la fórmula obtenidos en dicho Excel y el cambio de tiempo que va a realizar el LC2. Esos resultados se pueden observar en la siguiente tabla 1.

Tabla 1

Resultados en Excel de fórmulas del método Webster.

Fase	Mov.	#Carros	#Carriles	Feq	q	Si	S	L	Tco	Tv1	Tv2
1	1	125	1	1	125	0,42	0.583	8	40,8	23,43	9,37
2	2	50	1	1	50	0,16					

En la figura 30 se puede apreciar la simulación que se realizó entre la recepción de datos por comunicación serial y el desarrollo de las fórmulas del método Webster. Mediante la pantalla negra del virtual terminal se simula el envío de los valores de los automóviles que circulan por los sentidos de ambas calles con el respectivo formato que es A125B correspondiente a 125

vehículos que circulan por la calle principal en el transcurso de 10 min. y C050D correspondiente a 50 vehículos que han pasado por la calle secundaria.

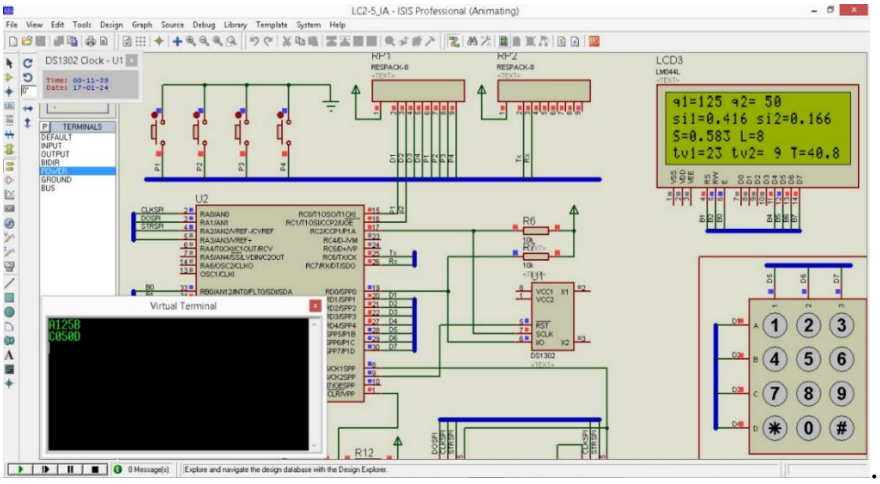
En la pantalla del LCD se puede visualizar cada uno de los valores intermedios que se obtienen hasta encontrar los valores finales de los tiempos de las luces verdes para la calle principal y para la calle secundaria.

Como se menciona en la metodología es necesario encontrar el tiempo perdido por ciclo (L), además el flujo de carros que no giran (q), el cual se necesita para calcular el índice de saturación para cada calle (Si1 y Si2), para al sumarlos (S), poder ingresarlo a la fórmula de tiempo óptimo de ciclo (Tco), que por motivos de espacio de la pantalla LCD se denomina (T).

Una vez tenemos el tiempo óptimo de ciclo se puede identificar cual será el tiempo óptimo de luz verdad para cada intersección (Tv1 y Tv2). Todas las variables que se muestran en el LCD coinciden con el resultado de dichas fórmulas obtenidas en el Excel tal como se muestra en la figura 30.

Figura 30

Funcionamiento del método Webster.



Nota: En la imagen se encuentra el mismo resultado que salió en la tabla 1. Fuente:Los autores.

En la tabla 2 se muestra los resultados correspondientes del Excel de nuevos valores que se envían por Hércules.

Tabla 2

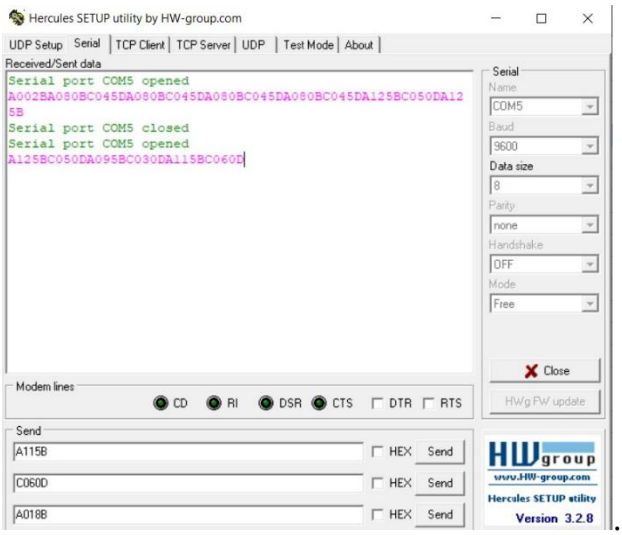
Resultados en Excel de fórmulas del método Webster.

Fase	Mov.	#Carros	#Carriles	Feq	q	Si	S	L	Tco	Tv1	Tv2
1	1	115	1	1	115	0,38	0.583	8	40,8	21,55	11,25
2	2	60	1	1	60	0,2					

A continuación, se muestra en la figura 31 el envío de las cadenas de datos que corresponden a A115B de la calle principal y C060D de la calle secundaria desde el Hércules, que simula el Python enviando estos datos por la comunicación serial al ESP32 maestro.

Figura 31

Envío de cadena de datos desde el Hércules.



Nota: Abriendo el puerto serial se logra enviar al ESP32 maestro la cadena de datos. Fuente:Los autores

En la siguiente figura 32 se aprecia que la comunicación a llegado al LC2 y las fórmulas del método Webster han sido correctamente desarrolladas.

Figura 32

Cambio de tiempo en la luz verde.

Nota: Los valor presentados entre corchetes corresponden a los tiempos prestablecidos que fueron ingresados por teclado, mismos que se encuentran grabados en la memoria emprom. Fuente:Los autores

El valor máximo que puede tener Tv1 considerando que es la calle principal es de 70seg, siendo el mínimo que puede tener 20seg. A su vez, el valor máximo que puede tener Tv2 considerando que es una calle secundaria es de 50seg y el valor mínimo es de 10seg. Estos valores son prestablecidos en la memoria eemprom.

En cambio, cuando los valores de tiempo verde Tv1 y Tv2 son menores a 10 seg, se asignará el valor mínimo de asignado a cada uno de los tiempos, que por defecto son 20 para Tv1 y 10 para Tv2.

Tabla 3

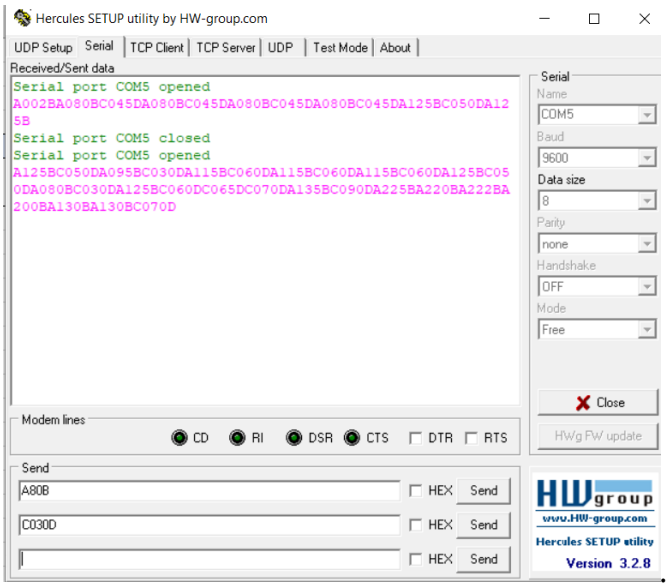
Resultados en Excel de fórmulas del método Webster.

Fase	Mov.	#Carros	#Carriles	Feq	q	Si	S	L	Tco	Tv1	Tv2
1	1	80	1	1	80	0,26	0.36	8	26,84	13,7	5,13
2	2	30	1	1	30	0,1					

En la siguiente figura 33 se muestra el envío de la cadena de datos de A080B y C030D para observar el funcionamiento en el LC2.

Figura 33

Envío de cadena de datos desde el Hércules.



Nota: Enviando al ESP32 maestro la cadena de datos a probar. Fuente:Los autores.

Como en la tabla 3 el valor de Tv1 es igual a 13,7 y el valor de Tv2 es igual a 5,13, se designa el valor mínimo que puede tener el tiempo de verde que es de 20 y 10 respectivamente.

Esto se debe a que normalmente se considera que el vehículo tarda un máximo de 5seg en salir cuando se encuentra detenido.

Como se puede observar en la figura 34 en la pantalla LCD, se encuentra el valor mínimo de Tv1 con el nombre de F1 y el valor mínimo de Tv2 en F2.

Figura 34

Cambio de tiempo en la luz verde.

Nota: Estos valores de tiempo de verde coinciden con el tiempo prestablecido. Fuente:Los autores.

De igual manera si los valores de tiempo verde Tv1 y Tv2 son mayores a los valores máximos establecidos por defecto que son 70 para Tv1 y 50 para Tv2, entonces Tv1 o Tv2

tomarán los valores por defecto. En la tabla 4 se muestra un ejemplo de lo mencionado anteriormente.

Tabla 4

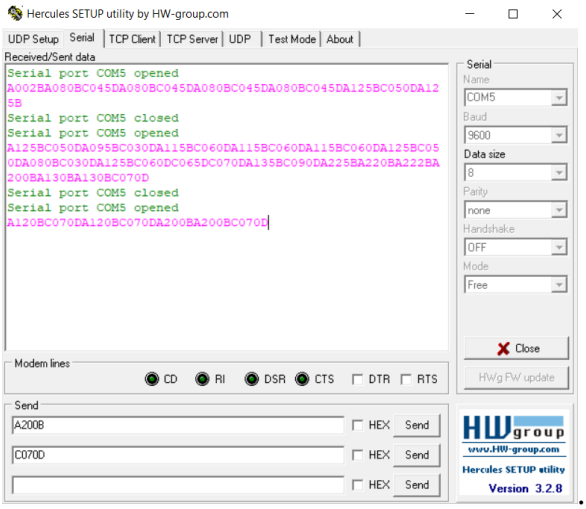
Resultados en Excel de fórmulas del método Webster.

Fase	Mov.	#Carros	#Carriles	Feq	q	Si	S	L	Tco	Tv1	Tv2
1	1	200	1	1	200	0,67	0.9	8	170	120	42
2	2	70	1	1	70	0,23					

En la siguiente figura 35 se muestra el envío de la cadena de datos de A200B y C070D para observar el funcionamiento en el LC2.

Figura 35

Envío de cadena de datos desde el Hércules.



Nota: Enviando al ESP32 maestro la cadena de datos a probar. Fuente:Los autores.

Como el valor de Tv1 es 120 que supera el valor máximo designado, en la pantalla del LCD se aplica el valor máximo de 70. Mientras que el valor de Tv2 se encuentra en el rango prestablecido, por lo cual, se visualiza el mismo valor que se encuentra en la tabla 4.

Figura 36

Cambio de tiempo en la luz verde.

Nota: El valor de F1 que corresponde a Tv1 coincide con el máximo valor de tiempo de verde prestablecido. Fuente:Los autores

VII CRONOGRAMA

En la tabla 5 se muestra el cronograma de todas las actividades que se realizará a lo largo del proceso que se llevará a cabo desde el 23 de octubre hasta el 24 de enero, se comenzará desde el levantamiento de la información para realizar varias propuestas de modelos para la inteligencia artificial, luego se proseguirá a realizar el montaje del controlador LC2 dando pie al diseño de la tarjeta para la comunicación del sistema embebido con el LC2. Por último, se realizará pruebas de funcionamiento y se cerrará este proceso con el documento de la tesis.

Tabla 5

Cronograma de actividades para el desarrollo del Proyecto de Titulación

ACTIVIDADES	octubre		noviembre				diciembre				enero
	3	4	1	2	3	4	1	2	3	4	1
Levantamiento de información											
Propuestas de modelas para la inteligencia artificial											
Montaje para la inteligencia artificial											
Diseño de la tarjeta para la comunicación de sistema embebido con el LC2											
Pruebas de funcionamiento											
Realizar el documento de la tesis											

VIII PRESUPUESTO

En la tabla 6 se muestra el presupuesto que se invertirá en el proyecto. Considerando que la empresa CORMAR S.A. facilitará los elementos necesarios como: el controlador LC2, dos semáforos de 200 mm correspondientes a los semáforos vehiculares que se encuentran dentro de las ciudades, una cámara de vigilancia y la fabricación de la tarjeta PCB con el diseño previamente realizado para la creación del prototipo de un semáforo adaptativo.

Tabla 6

Presupuesto estimado para el desarrollo del Proyecto de Titulación

Cant.	Detalle	Valor unit	Valor total
45	Horas de ingeniería	2,81	$126,45
	Valor total		$126,45

IX CONCLUSIONES

Según los resultados obtenidos al final del proyecto se logró el primer objetivo por que se logró ejecutar la IA que realiza la detección y conteo de autos junto con él envió de datos del contador mediante comunicación serial en pycharm usando el lenguaje de programación Python en su versión 3,9.

De la misma forma se cumplió con el segundo objetivo ya que se construyó la placa PCB que permite la comunicación entre el ESP32 y el PIC, apoyándose con el protocolo de comunicación ESP NOW mediante los ESP32 maestro/esclavo para el envío de datos desde el sistema embebido mostrando los datos en la pantalla LCD.

En el tercer objetivo gracias a la implementación de la fórmula del método de WEBSTER con los datos obtenidos de la IA del conteo vehicular, se pudo obtener el tiempo de ciclo óptimo necesario para calcular el tiempo óptimo para el correcto cambio de la luz verde en los semáforos, estas fórmulas se encuentran dentro de la programación del Pic 18F4550.

X RECOMENDACIONES

En este proyecto se ha hecho uso de varios entornos de desarrollo como lo son Pycharm, Arduino para la programación que se relaciona con la IA y la comunicación, de igual forma se usó software de simulación de circuitos eléctricos como lo son Proteus para la creación de placa PCB por la facilidad de acceder a ellos y el conocimiento base que se tiene, sin embargo, se sugiere tener un dominio alto en los lenguajes de programación Python y el de Arduino similar al C++ y la correcta instalación de las diferentes dependencias o librerías.

Al momento de realizar la comunicación entre dispositivos es muy importante verificar la correcta velocidad de estos, ya que al no ser la misma no se lograrán comunicar entre ellos, dando conflictos y enviando datos que son incorrectos.

Es posible ejecutar esta programación en cualquier equipo, sin embargo, se recomienda que sea un equipo que cuente con tarjea grafica ya que al ejecutar una IA necesita realizar varios procesos en muy poco tiempo y está al ser levantada en un equipo que no cuenta con tarjeta gráfica su rendimiento será mermado notablemente.

XI REFERENCIAS

Salas Arriarán, S. (2015). Todo sobre sistemas embebidos: arquitectura y diseño de aplicaciones prácticas con el PIC18F. Universidad Peruana de ciencias Aplicadas https://bibliotecas.ups.edu.ec:3488/es/ereader/bibliotecaups/41261?page=43

Trejos Buriticá, O, y Muñoz Guerrero, L. (Il.) (2021). Introducción a la Programación con Python. 1. Madrid, RA-MA Editorial. Recuperado de https://bibliotecas.ups.edu.ec:3488/es/ereader/bibliotecaups/230298?page=10.

Garrido, Á. (2020). Los avances de la inteligencia artificial. Madrid, Dykinson. Recuperado de https://bibliotecas.ups.edu.ec:3488/es/ereader/bibliotecaups/129597?page=195.

Loaiza Quintana, A, Manzano Herrera, D y Múnera Salazar, L. (2012). Sistema de visión artificial para conteo de objetos en movimiento. Universidad Autónoma de Occidente. Recuperado de https://red.uao.edu.co/bitstream/handle/10614/10870/A0114.pdf?sequence=4&isAllowed=y

Flores Calero, M, Quinga, B, Onofa, N, y Gallardo, F, (2019). Generación de regiones de interés con potencial de contener peatones mediante búsqueda focalizada usando visión monocular. https://cienciamerica.edu.ec/index.php/uti/article/view/178/291

Ultralytics. (s. f.). *Inicio*. Documentación Ultralytics YOLOv8. https://docs.ultralytics.com/es/

Terven Juan , R, Cordova Esperanza Diana, M, (2024). A comprehensive review of yolo architectures in computer vision: from yolov1 to yolov8 and yolo-na https://arxiv.org/pdf/2304.00501.pdf

Wang, X., Gao, H., Jia, Z., & Li, Z. (2023). BL-YOLOv8: An improved road defect detection model based on YOLOv8. *Sensors (Basel, Switzerland)*, *23*(20), 8361. https://doi.org/10.3390/s23208361

Wang, G., Chen, Y., An, P., Hong, H., Hu, J., & Huang, T. (2023). UAV-YOLOv8: A small-object-detection model based on improved YOLOv8 for UAV aerial photography scenarios. *Sensors (Basel, Switzerland)*, *23*(16), 7190 . https://doi.org/10.3390/s23167190

Jesús Bobadilla (2020) Machine Learning y Deep Learning: Usando Python, Scikit y Keras. https://books.google.es/books?hl=es&lr=&id=iAAyEAAAQBAJ&oi=fnd&pg=PA11&dq=dataset+que+es&ots=Qiv7uZkP4v&sig=ljKj2-Y5qeBjHnUsweNnvqEdYPw#v=onepage&q&f=false

Ameijeiras Sánchez, D, Gonzales Diez, H, Hernández Heredia, Y. (2020). Revisión de algoritmos de detección y seguimiento de objetos con redes profundas para videovigilancia inteligente. http://scielo.sld.cu/pdf/rcci/v14n3/2227-1899-rcci-14-03-165.pdf

Arévalo Calderón y Ortega Ulloa. (2016). Desarrollo de una interfaz para la visualización y Adquisición de datos provenientes del ecu a través de OBD-H mediante un dispositivo de comunicación serial y del analizador de gases Qrotech 6000. https://dspace.ups.edu.ec/bitstream/123456789/12029/1/UPS-CT005836.pdf

Fernández, Maritza Paola, y Jhon Fabricio Uquillas. (2020). Análisis De Desempeño Del Estándar Lorawan Para Soluciones De Smart Campus, Implementando Un Sistema De Monitoreo Iot En La Universidad De Las Fuerzas Armadas - Espe. Recuperado de https://dspace.ups.edu.ec/bitstream/123456789/24087/1/TTS1121.pdf

Arduino. (2024). Comunicación de dispositivo a dispositivo con ESP-NOW. Recuperado de https://docs.arduino.cc/tutorials/nano-esp32/esp-now

Córdova Martillo, L. S. (2022). Diseño e implementación de una Red inalámbrica para el Control de Reguladores de tránsito programables autosustentables, mediante el uso de Módulos XBEE Pro y Paneles Solares. Recuperado de https://dspace.ups.edu.ec/bitstream/123456789/22856/1/UPS- GT003870.pdf

Schmidt, D. (2022). Raspberry Pi: configuración y programación con Python: (2 ed.). RA-MA Editorial. https://bibliotecas.ups.edu.ec:3488/es/lc/bibliotecaups/titulos/222674

Muñoz Guerrero, L. E. (Il.) & Trejos Buriticá, O. I. (2021). Introducción a la programación con Python: (1 ed.). RA-MA Editorial. https://bibliotecas.ups.edu.ec:3488/es/lc/bibliotecaups/titulos/230298

Raspberrypi (s. f.). Accesorios y complementos raspberry pi. Recuperado de https://raspberrypi.cl/

Electrónica Unicrom, (s.f.). Fuente voltaje programable con LM317. Recuperado de https://unicrom.com/fuente-voltaje-programable-con-lm317/

María Alba y Oisy Hernández. (2013). Comparación de dos métodos de diseño para ciclos de semáforos. Recuperado de https://rci.cujae.edu.cu/index.php/rci/article/view/123/pdf

XII ANEXO

Código de la IA

```
# Librerias a usar en el proyecto
import threading
import psycopg2
from psycopg2 import sql
import hydra
import torch
import argparse
import time
from pathlib import Path
import cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random
from ultralytics.yolo.engine.predictor import BasePredictor
from ultralytics.yolo.utils import DEFAULT_CONFIG, ROOT, ops
from ultralytics.yolo.utils.checks import check_imgsz
from ultralytics.yolo.utils.plotting import Annotator, colors, save_one_box
import cv2
from deep_sort_pytorch.utils.parser import get_config
from deep_sort_pytorch.deep_sort import DeepSort
from collections import deque
import numpy as np
import serial

# Conexcion a la base de datos y variables globales

db_config = {
    'host': 'localhost',
    'database': 'postgres',
    'user': 'postgres',
    'password': '12345'
}
conn = psycopg2.connect(**db_config)
cursor = conn.cursor()

palette = (2 ** 11 - 1, 2 ** 15 - 1, 2 ** 20 - 1)
data_deque = {}

deepsort = None

object_counter = {}

prev_counter_principal = None

object_counter1 = {}

prev_counter_secundaria= None
```

```
line = [(500, 200), (800, 280)]
line1 = [(100, 290), (390, 250)]

puerto_serial = serial.Serial('COM3', 115200)

# Parametros del deepsort
def init_tracker():
    global deepsort
    cfg_deep = get_config()
    cfg_deep.merge_from_file("deep_sort_pytorch/configs/deep_sort.yaml")

    deepsort = DeepSort(cfg_deep.DEEPSORT.REID_CKPT,
                        max_dist=cfg_deep.DEEPSORT.MAX_DIST,
min_confidence=cfg_deep.DEEPSORT.MIN_CONFIDENCE,
                        nms_max_overlap=cfg_deep.DEEPSORT.NMS_MAX_OVERLAP,
                        max_iou_distance=cfg_deep.DEEPSORT.MAX_IOU_DISTANCE,
                        max_age=cfg_deep.DEEPSORT.MAX_AGE,
n_init=cfg_deep.DEEPSORT.N_INIT,
                        nn_budget=cfg_deep.DEEPSORT.NN_BUDGET,
                        use_cuda=True)

# Declaracion de funciones a llamar mienstras se ejecuta la IA
def xyxy_to_xywh(*xyxy):
    bbox_left = min([xyxy[0].item(), xyxy[2].item()])
    bbox_top = min([xyxy[1].item(), xyxy[3].item()])
    bbox_w = abs(xyxy[0].item() - xyxy[2].item())
    bbox_h = abs(xyxy[1].item() - xyxy[3].item())
    x_c = (bbox_left + bbox_w / 2)
    y_c = (bbox_top + bbox_h / 2)
    w = bbox_w
    h = bbox_h
    return x_c, y_c, w, h


def xyxy_to_tlwh(bbox_xyxy):
    tlwh_bboxs = []
    for i, box in enumerate(bbox_xyxy):
        x1, y1, x2, y2 = [int(i) for i in box]
        top = x1
        left = y1
        w = int(x2 - x1)
        h = int(y2 - y1)
        tlwh_obj = [top, left, w, h]
        tlwh_bboxs.append(tlwh_obj)
    return tlwh_bboxs


def compute_color_for_labels(label):
    if label == 2:  # Car
        color = (222, 82, 175)
    else:
        color = [int((p * (label ** 2 - label + 1)) % 255) for p in palette]
    return tuple(color)
```

```
def draw_border(img, pt1, pt2, color, thickness, r, d):
    x1, y1 = pt1
    x2, y2 = pt2
    # Parte superior izquierda
    cv2.line(img, (x1 + r, y1), (x1 + r + d, y1), color, thickness)
    cv2.line(img, (x1, y1 + r), (x1, y1 + r + d), color, thickness)
    cv2.ellipse(img, (x1 + r, y1 + r), (r, r), 180, 0, 90, color, thickness)
    # Parte superior derecha
    cv2.line(img, (x2 - r, y1), (x2 - r - d, y1), color, thickness)
    cv2.line(img, (x2, y1 + r), (x2, y1 + r + d), color, thickness)
    cv2.ellipse(img, (x2 - r, y1 + r), (r, r), 270, 0, 90, color, thickness)
    # Parte inferior izquierda
    cv2.line(img, (x1 + r, y2), (x1 + r + d, y2), color, thickness)
    cv2.line(img, (x1, y2 - r), (x1, y2 - r - d), color, thickness)
    cv2.ellipse(img, (x1 + r, y2 - r), (r, r), 90, 0, 90, color, thickness)
    # Parte inferior derecha
    cv2.line(img, (x2 - r, y2), (x2 - r - d, y2), color, thickness)
    cv2.line(img, (x2, y2 - r), (x2, y2 - r - d), color, thickness)
    cv2.ellipse(img, (x2 - r, y2 - r), (r, r), 0, 0, 90, color, thickness)

    cv2.rectangle(img, (x1 + r, y1), (x2 - r, y2), color, -1, cv2.LINE_AA)
    cv2.rectangle(img, (x1, y1 + r), (x2, y2 - r - d), color, -1, cv2.LINE_AA)

    cv2.circle(img, (x1 + r, y1 + r), 2, color, 12)
    cv2.circle(img, (x2 - r, y1 + r), 2, color, 12)
    cv2.circle(img, (x1 + r, y2 - r), 2, color, 12)
    cv2.circle(img, (x2 - r, y2 - r), 2, color, 12)

    return img


def UI_box(x, img, color=None, label=None, line_thickness=None):
    # Traza un cuadro delimitador en la imagen img
    tl = line_thickness or round(0.002 * (img.shape[0] + img.shape[1]) / 2) +
1  # grosor de línea/fuente
    color = color or [random.randint(0, 255) for _ in range(3)]
    c1, c2 = (int(x[0]), int(x[1])), (int(x[2]), int(x[3]))
    cv2.rectangle(img, c1, c2, color, thickness=tl, lineType=cv2.LINE_AA)
    if label:
        tf = max(tl - 1, 1) # grosor de fuente
        t_size = cv2.getTextSize(label, 0, fontScale=tl / 3, thickness=tf)[0]

        img = draw_border(img, (c1[0], c1[1] - t_size[1] - 3), (c1[0] +
t_size[0], c1[1] + 3), color, 1, 8, 2)

        cv2.putText(img, label, (c1[0], c1[1] - 2), 0, tl / 3, [225, 255,
255], thickness=tf, lineType=cv2.LINE_AA)
def intersect(A, B, C, D):
    return ccw(A, C, D) != ccw(B, C, D) and ccw(A, B, C) != ccw(A, B, D)
def ccw(A, B, C):
    return (C[1] - A[1]) * (B[0] - A[0]) > (B[1] - A[1]) * (C[0] - A[0])
def get_direction(point1, point2):
    direction_str = ""
```

```
    # calcular la dirección del eje y
    if point1[1] > point2[1]:
        direction_str += "South"
    elif point1[1] < point2[1]:
        direction_str += "North"
    else:
        direction_str += ""

    # calcular la dirección del eje x
    if point1[0] > point2[0]:
        direction_str += "East"
    elif point1[0] < point2[0]:
        direction_str += "West"
    else:
        direction_str += ""

    return direction_str

def reset_counters():
    global object_counter, object_counter1
    object_counter = {}
    object_counter1 = {}
    threading.Timer(60.0, reset_counters).start()
def draw_boxes(img, bbox, names, object_id, identities=None, offset=(0, 0)):
    global prev_counter_principal, prev_counter_secundaria
    cv2.line(img, line[0], line[1], (46, 162, 112), 3)
    cv2.line(img, line1[0], line1[1], (46, 162, 112), 3)
    height, width, _ = img.shape

    # eliminar el punto rastreado del buffer si el objeto se pierde
    for key in list(data_deque):
        if key not in identities:
            data_deque.pop(key)

    for i, box in enumerate(bbox):
        x1, y1, x2, y2 = [int(i) for i in box]
        x1 += offset[0]
        x2 += offset[0]
        y1 += offset[1]
        y2 += offset[1]

        # código para encontrar el centro del borde inferior
        center = (int((x2 + x1) / 2), y2)

        # obtener ID del objeto
        id = int(identities[i]) if identities is not None else 0

        # crear un nuevo buffer para un nuevo objeto
        if id not in data_deque:
            data_deque[id] = deque(maxlen=64)
        color = compute_color_for_labels(object_id[i])
        obj_name = names[object_id[i]]
        label = '{}{:d}'.format("", id) + ":" + '%s' % (obj_name)
```

```
        # Dibuja solo si el objeto detectado es un automóvil (clase 2)
        if object_id[i] == 2:
            UI_box(box, img, label=label, color=color, line_thickness=2)
            # dibujar rastro
            for i in range(1, len(data_deque[id])):
                # comprobar si el valor del buffer es ninguno
                if data_deque[id][i - 1] is None or data_deque[id][i] is None:
                    continue
                # generar espesor dinámico de senderos
                thickness = int(np.sqrt(64 / float(i + i)) * 1.5)
                # dibujar senderos
                cv2.line(img, data_deque[id][i - 1], data_deque[id][i], color,
thickness)

        # agregar centro al búfer
        data_deque[id].appendleft(center)
        if len(data_deque[id]) >= 2:
            direction = get_direction(data_deque[id][0], data_deque[id][1])
            if intersect(data_deque[id][0], data_deque[id][1], line[0],
line[1]):
                cv2.line(img, line[0], line[1], (255, 255, 255), 3)
                if "West" in direction:
                    if obj_name not in object_counter:
                        object_counter[obj_name] = 1
                    else:
                        object_counter[obj_name] += 1

            if intersect(data_deque[id][0], data_deque[id][1], line1[0],
line1[1]):
                cv2.line(img, line1[0], line1[1], (255, 255, 255), 3)
                if "North" in direction:
                    if obj_name not in object_counter1:
                        object_counter1[obj_name] = 1
                    else:
                        object_counter1[obj_name] += 1
        # Mostrar recuento en la esquina superior izquierda
        for idx, (key, value) in enumerate(object_counter.items()):
            cnt_str1 = str(key) + ":" + str(value).rjust(3, '0')
            cv2.line(img, (20, 25), (260, 25), [85, 45, 255], 40)
            cv2.putText(img, f'C.Calle Principal', (11, 35), 0, 1, [225, 255,
255], thickness=2, lineType=cv2.LINE_AA)
            cv2.line(img, (20, 65 + (idx * 40)), (150, 65 + (idx * 40)), [85,
45, 255], 30)
            cv2.putText(img, cnt_str1, (11, 75 + (idx * 40)), 0, 1, [225, 255,
255], thickness=2, lineType=cv2.LINE_AA)

            objeto_nombre = str(key)
            objeto_tipo = "Calle Principal"
            contador_principal = value

            #Envio de datos a la base de datos
            tabla = 'contador_datos'
            condicion = "objeto_tipo = 'Calle Principal' AND objeto_nombre =
```

```
'"
            condicion += str(key)
            condicion += "'"
            consulta = sql.SQL("DELETE FROM {} WHERE
{};").format(sql.Identifier(tabla), sql.SQL(condicion))
            cursor.execute(consulta)
            conn.commit()

            insert_query = sql.SQL("""
                    INSERT INTO contador_datos (objeto_nombre, objeto_tipo,
contador_principal)
                    VALUES (%s, %s, %s)
                    """)
            cursor.execute(insert_query, (objeto_nombre, objeto_tipo,
contador_principal))
            conn.commit()
            #Envio de datos por comunicacion serial
            if prev_counter_principal is None or prev_counter_principal !=
value:
                mensaje_principal = f"A{str(value).zfill(3)}B"
                print(f"Dato enviado por comunicacion serial:"+
mensaje_principal)
                puerto_serial.write(mensaje_principal.encode())
                prev_counter_principal = value

            # Mostrar recuento en la esquina superior derecha
        for idx, (key, value) in enumerate(object_counter1.items()):
            cnt_str = str(key) + ":" + str(value).rjust(3, '0')
            cv2.line(img, (width - 500, 25), (width, 25), [85, 45, 255], 40)
            cv2.putText(img, f'C.Calle Secundaria', (width - 500, 35), 0, 1,
[225, 255, 255], thickness=2, lineType=cv2.LINE_AA)
            cv2.line(img, (width - 150, 65 + (idx * 40)), (width, 65 + (idx *
40)), [85, 45, 255], 30)
            cv2.putText(img, cnt_str, (width - 150, 75 + (idx * 40)), 0, 1,
[255, 255, 255], thickness=2, lineType=cv2.LINE_AA)

            objeto_nombre = str(key)
            objeto_tipo = "Calle Secundaria"
            contador_secundaria = value

            #Envio de datos a la base de datos
            tabla = 'contador_datos'
            condicion = "objeto_tipo = 'Calle Secundaria' AND objeto_nombre =
'"
            condicion += str(key)
            condicion += "'"
            consulta = sql.SQL("DELETE FROM {} WHERE
{};").format(sql.Identifier(tabla), sql.SQL(condicion))
            cursor.execute(consulta)
            conn.commit()
            # Insertar datos en la tabla
            insert_query = sql.SQL("""
                    INSERT INTO contador_datos (objeto_nombre, objeto_tipo,
contador_secundaria)
```

```
                    VALUES (%s, %s, %s)
                    """)
            cursor.execute(insert_query, (objeto_nombre, objeto_tipo,
contador_secundaria))
            conn.commit()
            # Envio de datos por comunicacion serial
            if prev_counter_secundaria is None or prev_counter_secundaria !=
value:
                mensaje_secundaria = f"C{str(value).zfill(3)}D"
                print(f"Dato enviado por comunicacion serial:" +
mensaje_secundaria)
                puerto_serial.write(mensaje_secundaria.encode())
                prev_counter_secundaria = value

    return img


reset_counters()


class DetectionPredictor(BasePredictor):

    def get_annotator(self, img):
        return Annotator(img, line_width=self.args.line_thickness,
example=str(self.model.names))

    def preprocess(self, img):
        img = torch.from_numpy(img).to(self.model.device)
        img = img.half() if self.model.fp16 else img.float()  # uint8 to
fp16/32
        img /= 255  # 0 - 255 to 0.0 - 1.0
        return img

    def postprocess(self, preds, img, orig_img):
        preds = ops.non_max_suppression(preds,
                                        self.args.conf,
                                        self.args.iou,
                                        agnostic=self.args.agnostic_nms,
                                        max_det=self.args.max_det)

        for i, pred in enumerate(preds):
            shape = orig_img[i].shape if self.webcam else orig_img.shape
            pred[:, :4] = ops.scale_boxes(img.shape[2:], pred[:, :4],
shape).round()

        return preds

    def write_results(self, idx, preds, batch):
        p, im, im0 = batch
        all_outputs = []
        log_string = ""
        if len(im.shape) == 3:
            im = im[None]  # expandir para atenuar lotes
        self.seen += 1
```

```
        im0 = im0.copy()
        if self.webcam:  # tamaño_lote >= 1
            log_string += f'{idx}: '
            frame = self.dataset.count
        else:
            frame = getattr(self.dataset, 'frame', 0)

        self.data_path = p
        save_path = str(self.save_dir / p.name)  # im.jpg
        self.txt_path = str(self.save_dir / 'labels' / p.stem) + ('' if
self.dataset.mode == 'image' else f'_{frame}')
        log_string += '%gx%g ' % im.shape[2:]  # cadena de impresión
        self.annotator = self.get_annotator(im0)

        det = preds[idx]
        all_outputs.append(det)
        if len(det) == 0:
            return log_string
        for c in det[:, 5].unique():
            n = (det[:, 5] == c).sum()  # # detecciones por clase
            log_string += f"{n} {self.model.names[int(c)]}{'s' * (n > 1)}, "
        # escribir
        gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # ganancia de
normalización whwh
        xywh_bboxs = []
        confs = []
        oids = []
        outputs = []
        for *xyxy, conf, cls in reversed(det):
            x_c, y_c, bbox_w, bbox_h = xyxy_to_xywh(*xyxy)
            xywh_obj = [x_c, y_c, bbox_w, bbox_h]
            xywh_bboxs.append(xywh_obj)
            confs.append([conf.item()])
            oids.append(int(cls))
        xywhs = torch.Tensor(xywh_bboxs)
        confss = torch.Tensor(confs)

        outputs = deepsort.update(xywhs, confss, oids, im0)
        if len(outputs) > 0:
            bbox_xyxy = outputs[:, :4]
            identities = outputs[:, -2]
            object_id = outputs[:, -1]

            draw_boxes(im0, bbox_xyxy, self.model.names, object_id,
identities)

        return log_string


@hydra.main(version_base=None, config_path=str(DEFAULT_CONFIG.parent),
config_name=DEFAULT_CONFIG.name)
def predict(cfg):
    init_tracker()
    cfg.model = cfg.model or "yolov8n.pt"
```

```
    cfg.imgsz = check_imgsz(cfg.imgsz, min_dim=2)  # check image size
    cfg.source = cfg.source if cfg.source is not None else ROOT / "assets"
    predictor = DetectionPredictor(cfg)
    predictor()


if __name__ == "__main__":
    predict()
    cursor.close()
    conn.close()
puerto_serial.close()
```

Código ESP-NOW maestro

```
/TESIS

#include <esp_now.h>

#include <WiFi.h>

#include <esp_wifi.h> // only for esp_wifi_set_channel()

// Global copy of slave

esp_now_peer_info_t slave;

#define CHANNEL 1

#define PRINTSCANRESULTS 0

#define DELETEBEFOREPAIR 0

uint8_t lastReceivedData = 0;

// Init ESP Now with fallback

void InitESPNow() {

  WiFi.disconnect();

  if (esp_now_init() == ESP_OK) {

    //Serial.println("ESPNow Init Success");

  }

  else {

        ESP.restart();

  }

}

// Scan for slaves in AP mode

void ScanForSlave() {

  int16_t scanResults = WiFi.scanNetworks(false, false, false, 300, CHANNEL); // Scan

only on one channel

  // reset on each scan
```

```
  bool slaveFound = 0;
  memset(&slave, 0, sizeof(slave));
  //Serial.println("");
  if (scanResults == 0) {
    //Serial.println("No WiFi devices in AP Mode found");
  } else {
    //Serial.print("Found "); Serial.print(scanResults); Serial.println(" devices ");
    for (int i = 0; i < scanResults; ++i) {
      // Print SSID and RSSI for each device found
      String SSID = WiFi.SSID(i);
      int32_t RSSI = WiFi.RSSI(i);
      String BSSIDstr = WiFi.BSSIDstr(i);
      delay(10);
      // Check if the current device starts with `Slave`
      if (SSID.indexOf("Slave") == 0) {
        Serial.print(BSSIDstr); Serial.print("]"); Serial.print(" (");
Serial.print(RSSI); Serial.print(")"); Serial.println("");
        // Get BSSID => Mac Address of the Slave
        int mac[6];
        if ( 6 == sscanf(BSSIDstr.c_str(), "%x:%x:%x:%x:%x:%x",  &mac[0], &mac[1],
&mac[2], &mac[3], &mac[4], &mac[5] ) ) {
          for (int ii = 0; ii < 6; ++ii ) {
            slave.peer_addr[ii] = (uint8_t) mac[ii];
          }
        }
        slave.channel = CHANNEL; // pick a channel
        slave.encrypt = 0; // no encryption
```

```
        slaveFound = 1;
        break;
      }
    }
  }
  // clean up ram
  WiFi.scanDelete();
}
// Check if the slave is already paired with the master.
// If not, pair the slave with master
bool manageSlave() {
  if (slave.channel == CHANNEL) {
    if (DELETEBEFOREPAIR) {
      deletePeer();
    }
    // check if the peer exists
    bool exists = esp_now_is_peer_exist(slave.peer_addr);
    if ( exists) {
      // Slave already paired.
      //Serial.println("Already Paired");
      return true;
    } else {
      // Slave not paired, attempt pair
      esp_err_t addStatus = esp_now_add_peer(&slave);
      if (addStatus == ESP_OK) {
        return true;
```

```
      } else if (addStatus == ESP_ERR_ESPNOW_NOT_INIT) {
        return false;
      } else if (addStatus == ESP_ERR_ESPNOW_ARG) {
        return false;
      } else if (addStatus == ESP_ERR_ESPNOW_FULL) {
        return false;
      } else if (addStatus == ESP_ERR_ESPNOW_NO_MEM) {
        return false;
      } else if (addStatus == ESP_ERR_ESPNOW_EXIST) {
        return true;
      } else {
        return false;
      }
    }
  } else {
    return false;
  }
}
void deletePeer() {
  esp_err_t delStatus = esp_now_del_peer(slave.peer_addr);
  //Serial.print("Slave Delete Status: ");
  if (delStatus == ESP_OK) {
  } else if (delStatus == ESP_ERR_ESPNOW_NOT_INIT) {
  } else if (delStatus == ESP_ERR_ESPNOW_ARG) {

  } else if (delStatus == ESP_ERR_ESPNOW_NOT_FOUND) {
  } else {
```

```
  }
}
void onDataReceived();
void sendDataToSlave(uint8_t data);
// Callback cuando se recibe un dato por el puerto serie
void onDataReceived() {
  while (Serial.available()) {
    uint8_t receivedData = Serial.read();
    // Realizar la comparación con el último dato recibido
    //if (receivedData != lastReceivedData) {
      // Si hay un cambio, enviar el nuevo dato al esclavo
      sendDataToSlave(receivedData);
      //lastReceivedData = receivedData; // Actualizar el último dato recibido
    //} else {
      // Si no hay cambios, no enviar el dato
      //Serial.println("El dato es igual al último recibido. No se enviará.");
    }
}
void sendDataToSlave(uint8_t data) {
  if (slave.channel == CHANNEL) {
    const uint8_t *peer_addr = slave.peer_addr;
    esp_err_t result = esp_now_send(peer_addr, &data, sizeof(data));
    // Manejo del resultado del envío...
  } else {


  }
}
```

```
uint8_t data = 0;

// send data

void sendData() {

  data++;

  const uint8_t *peer_addr = slave.peer_addr;

  esp_err_t result = esp_now_send(peer_addr, &data, sizeof(data));

  if (result == ESP_OK) {

    //Serial.println("Success");

  } else if (result == ESP_ERR_ESPNOW_NOT_INIT) {

    // How did we get so far!!

    //Serial.println("ESPNOW not Init.");

  } else if (result == ESP_ERR_ESPNOW_ARG) {

    //Serial.println("Invalid Argument");

  } else if (result == ESP_ERR_ESPNOW_INTERNAL) {

    //Serial.println("Internal Error");

  } else if (result == ESP_ERR_ESPNOW_NO_MEM) {

    //Serial.println("ESP_ERR_ESPNOW_NO_MEM");

  } else if (result == ESP_ERR_ESPNOW_NOT_FOUND) {

    //Serial.println("Peer not found.");

  } else {

    //Serial.println("Not sure what happened");

  }

}

// callback when data is sent from Master to Slave

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {

  char macStr[18];

  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
```

```
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
mac_addr[5]);
}
void setup() {
  Serial.begin(9600);
  Serial.setTimeout(50);
  Serial.setRxBufferSize(64);
  WiFi.mode(WIFI_STA);
  esp_wifi_set_channel(CHANNEL, WIFI_SECOND_CHAN_NONE);
  InitESPNow();
  esp_now_register_send_cb(OnDataSent);
}
void loop() {
  ScanForSlave();
  if (slave.channel == CHANNEL) {
    bool isPaired = manageSlave();
    if (isPaired) {
      onDataReceived(); // Llama a la función que maneja los datos recibidos por
Serial
      // Resto del código existente...
    } else {
      //Serial.println("Falló el emparejamiento con el esclavo.");
    }
  }
  delay(500);
}
```

Código ESP-NOW esclavo

```
/**
   Tesis
*/
#include <esp_now.h>
#include <WiFi.h>
#define CHANNEL 1
int cont=0;
char valor[3];
// Init ESP Now with fallback
void InitESPNow() {
  WiFi.disconnect();
  if (esp_now_init() == ESP_OK) {
  //  Serial.println("ESPNow Init Success");
  }
  else {
  //  Serial.println("ESPNow Init Failed");
    // Retry InitESPNow, add a counte and then restart?
    // InitESPNow();
    // or Simply Restart
    ESP.restart();
  }
}

// config AP SSID
void configDeviceAP() {
```

```
  const char *SSID = "Slave_1";
  bool result = WiFi.softAP(SSID, "Slave_1_Password", CHANNEL, 0);
  if (!result) {
  //  Serial.println("AP Config failed.");
  } else {
   // Serial.println("AP Config Success. Broadcasting with AP: " + String(SSID));
  //  Serial.print("AP CHANNEL "); Serial.println(WiFi.channel());
  }
}
// callback when data is recv from Master
void OnDataRecv(const esp_now_recv_info_t * info, const uint8_t * data, int data_len)
{
  //if (data[0] > 47 && data[0] < 58)
  {
    char macStr[5];
    digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  //snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
  //         info->src_addr[0], info->src_addr[1], info->src_addr[2], info-
>src_addr[3], info->src_addr[4], info->src_addr[5]);
  //Serial.print("Last Packet Recv from: "); Serial.println(macStr);
  //Serial.write("CM");
  //Serial.write(*data);
    valor[cont]=data[0];
    cont++;
    //if (cont >= 3){
      macStr[0]='C';
      macStr[1]='M';
```

```
      macStr[2]=valor[0];
      macStr[3]=valor[1];
      macStr[4]=valor[2];
      //Serial.write(macStr);
      Serial.write(* data);
      digitalWrite(LED_BUILTIN, LOW);
      cont = 0;
   // }
  }
  //  delay(25);
 // Serial.println();
 //Serial.println("");
}
void setup() {
  Serial.begin(9600);   //115200
 // Serial.println("ESPNow/Basic/Slave Example");
  //Set device in AP mode to begin with
  pinMode(LED_BUILTIN, OUTPUT);
  WiFi.mode(WIFI_AP);
  // configure device AP mode
  configDeviceAP();
  // This is the mac address of the Slave in AP Mode
  //Serial.print("AP MAC: ");
  //Serial.println(WiFi.softAPmacAddress());
  // Init ESPNow with a fallback logic
  InitESPNow();
  // Once ESPNow is successfully Init, we will register for recv CB to
```

```
  // get recv packer info.
  esp_now_register_recv_cb(OnDataRecv);
}
void loop() {
  // Chill
}
```

Código de la comunicación para el PIC

```
#INT_RDA

void RDA_isr()

{

char st1[5];          // @123/567#

char st2[5];

  data="     ";

  scanf("%s",data);

  printf(lcd_putc,"\f");

  if(data[0]==65 && data[4]==66)       // Pregunta por A y B )

  {

//   cont++;

   st1[0]=data[1];

   st1[1]=data[2];

   st1[2]=data[3];

   val1 = atoi(st1);

   tf1=t_fas1+val1;

  }

  if(data[0]==67 && data[4]==68)       // Pregunta por C y D )

  {

//   cont++;

   st2[0]=data[1];

   st2[1]=data[2];

   st2[2]=data[3];

   val2 = atoi(st2);

   val2=val2/10;
```

```
   tf2=t_fas2+val2;

  }

//  printf(lcd_putc,"Cont=%d",cont);

//  lcd_gotoxy(1,3);

//  printf(lcd_putc,"[%s]",data);

   lcd_gotoxy(1,2);

   printf(lcd_putc,"F1[%2d]=%2d  F2[%2d]=%2d",t_fas1,tf1,t_fas2,tf2);

}
```

Código con la implementación con la fórmula de Webster

```
#include<18f4550.h> //micropic

#FUSES XT, NOWDT, NOWRT, NOPROTECT, NOPUT, NOLVP

#use delay (clock=1000000) // o 4Mhz

#use rs232(baud=9600, xmit=pin_C6,rcv=pin_C7,TIMEOUT=1000)

#include <lcd420.c>

#include <stdlib.h>

#INCLUDE <math.h>

#byte portb=0xf81

#byte portd=0xf83


int t_ci,t_ai, t_ri;       //  LOS TIEMPOS COMIENZAN DESDE LA DIRECCION 150 DE LA

EEPROM

int t_fas1,t_fas2;        // 1 secuencias y 2 fases

int t_v1,t_v2;

int cant_fas, v1min, v1max, v2min, v2max;

int cant_carril_p, cant_carril_s;


int16 val1;     //valor de cantidad de carros principal

int16 val2;     //valor de cantidad de carros secundaria

int tf1,tf2;


int fact_equ;

int16 fluj_sat;

int16 q1,q2;        //flujo de autos directos calle principal q1 y secundaria q2
```

```
float S,si1,si2;    // Índice de saturación
int L;
float tco;           //tiempo óptimo de ciclo



void main(void)
{
set_tris_b(0x00);   // Aqui programo al puerto b como salidas
 set_tris_d(0x1E);
 set_tris_e(0x00);
 portb=0; portd=0;
t_ai=3; t_ri=1;
cant_carril_p=1;
cant_carril_s=1;
cant_fas=2;
t_v1=0; t_v2=0;
fact_equ=1;          //factor de equivalencia
fluj_sat=300;        //flujo de saturación 1800/hora  300/10min
val1=125;
val2=50;
tf1=0;
tf2=0;

 lcd_init();

while(true)
 {
```

```
    q1=(val1*fact_equ)/cant_carril_p;
    q2=(val2*fact_equ)/cant_carril_s;
    si1=(float)(q1*1)/fluj_sat;
    si2=(float)(q2*1)/fluj_sat;
    S=si1+si2;
    L=cant_fas*(t_ai + t_ri);
    tco=((L/2)+L)+5;
    tco=tco/(1-S);
    t_v1=(int)((si1/S)*(tco-L));
    t_v2=(int)((si2/S)*(tco-L));

      lcd_gotoxy(1,1);
      printf(lcd_putc,"q1=%3ld q2=%3ld",q1,q2);
      lcd_gotoxy(1,2);
      printf(lcd_putc,"si1=%1.3f si2=%1.3f",si1,si2);
      lcd_gotoxy(1,3);
      printf(lcd_putc,"S=%1.3f L=%1d",S,L);
      lcd_gotoxy(1,4);
      printf(lcd_putc,"tv1=%2d tv2=%2d T=%1.3f",t_v1,t_v2,tco);

   }
}
```