UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
Facultad de Ingeniería, Arquitectura y Diseño
Programa de Maestría y Doctorado en Ciencias e Ingeniería

DISEÑO Y CONSTRUCCIÓN DE UN CONTROLADOR DE SEMÁFORO INTELIGENTE CON COMUNICACIÓN INALÁMBRICA

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN INGENIERÍA

presenta:

ALFONSO NAVARRO ESPINOZA

Director de tesis

Dr. Everardo Inzunza González

Ensenada, Baja California, México. Junio de 2022.

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Ingeniería, Arquitectura y Diseño

DISEÑO Y CONSTRUCCIÓN DE UN CONTROLADOR DE SEMÁFORO INTELIGENTE CON COMUNICACIÓN INALÁMBRICA

TESIS

que para obtener el grado de MAESTRO en INGENIERÍA presenta:

ALFONSO NAVARRO ESPINOZA

Y aprobada por el siguiente comité:

Dr. Everardo Inzunza González
Director de Tesis

Dr. Oscar Roberto López Bonilla
Co-Director de Tesis

Dr. Enrique Efrén García Guerrero
Miembro del Comité

Dr. Ulises Jesús Tamayo Perez
Miembro de Comité

Dra. Eloisa del Carmen García Canseco
Miembro del Comité

Junio de 2022

RESUMEN de la tesis de **Alfonso Navarro Espinoza**, presentada como requisito parcial para obtener el grado de MAESTRO EN INGENIERÍA, del programa de Maestría y Doctorado en Ciencias e Ingeniería de la UABC. Ensenada, B. C. México, Junio de 2022.

DISEÑO Y CONSTRUCCIÓN DE UN CONTROLADOR DE SEMÁFORO INTELIGENTE CON COMUNICACIÓN INALÁMBRICA

Resumen aprobado por:

Dr. Everardo Inzunza González
Director de Tesis

Dr. Oscar Roberto López Bonilla
Co-director de Tesis

En este trabajo de tesis de maestría, se presenta el diseño y desarrollo de un prototipo de controlador de semáforo inteligente basado en un microcontrolador STM32F103. El sistema cuenta con una interfaz gráfica de usuario para la programación y el control de los semáforos, está dotado de módulos de comunicación inalámbrica para la transmisión remota de instrucciones e información. Se proponen y evalúan múltiples algoritmos de aprendizaje automático (ML) para la predicción del flujo vehicular en una intersección, sentando así las bases para el control adaptativo del tráfico, ya sea mediante el control remoto de los semáforos o mediante la aplicación de un algoritmo que ajuste los tiempos en función del flujo previsto. Dos conjuntos de datos públicos son usados para entrenar, validar y probar los modelos ML.
Como parte complementaria se propone un contador de vehículos basado en sensores ultrasónicos y comunicación inalámbrica con una PC para la generación de una base de datos.

Palabras clave: Sistema de transporte inteligente, Semáforo inteligente, Predicción del flujo de tráfico, Comunicación inalámbrica, Aprendizaje automático.

ABSTRACT of the thesis of **Alfonso Navarro Espinoza**, presented as a partial requirement to obtain the degree of MASTER IN ENGINEERING, of the program of MSc and PhD in Sciences and Engineering of UABC. Ensenada, B. C., Mexico, June 2022.

DESIGN AND CONSTRUCTION OF A SMART TRAFFIC LIGHT CONTROLLER WITH WIRELESS COMMUNICATION

Abstract approved by:

Dr. Everardo Inzunza González
Thesis Supervisor

Dr. Oscar Roberto López Bonilla
Thesis Co-Supervisor

In this master's thesis, the design and development of a prototype of an intelligent traffic light controller based on a STM32F103 microcontroller is presented. The system has a graphical user interface for programming and control of traffic lights, is equipped with wireless communication modules for remote transmission of instructions and information. Machine-learning (ML) algorithms are proposed and evaluated for predicting traffic flow at an intersection, thus laying the groundwork for adaptive traffic control, either by remote control of traffic lights or by applying an algorithm that adjusts the timing according to the predicted flow. Two public datasets are used to train, validate and test the proposed ML models.
As a complementary part, a vehicle counter based on ultrasonic sensors and wireless communication with a PC for the generation of a database is proposed.

Keywords: Intelligent transportation system, Smart traffic light, Traffic flow prediction, Wireless communication, Machine Learning.

A mi familia

Agradecimientos

*Al **Dr. Everardo Inzunza González**, por haberme brindado su dirección y apoyo en las distintas etapas del programa de maestría, por sus valiosos consejos, paciencia y dedicación.*

*Al **Dr. Oscar Roberto López Bonilla**, por todo el apoyo que me ha dado en la realización del presente trabajo de tesis y por los conocimientos compartidos.*

*Al **Dr. Enrique Efrén García Guerrero, Dra. Eloisa del Carmen García Canseco, Dr. Ulises Jesús Tamayo Pérez**, por sus aportaciones y oportunos comentarios en las revisiones y presentaciones dentro del proceso del posgrado.*

*A mis padres **Marcos Navarro Pérez y Gloria Espinoza Ascencio**, por todo su apoyo, motivación y consejos no solo en esta etapa de mi vida, sino en todo momento. Los quiero mucho!*

*A mis hermanos **Viviana y Carlos**, por estar presentes y ser una parte fundamental en esta etapa tan importante en mi vida. Los quiero mucho hermanos!*

*A mis compañeros de posgrado **Eduardo Enrique Contreras Lujan y Edgar Rene Ramos Acosta**, por las experiencias y la amistad brindada a lo largo de estos años de estudio.*

*Al **CONACyT**, por apoyarme financieramente en mis estudios de maestría.*

*A nuestra alma mater: **Universidad Autónoma de Baja California**, que tanto me ha dado académicamente y profesionalmente.*

A todas las personas que de alguna u otra manera estuvieron involucradas conmigo a lo largo de mis estudios de maestría.

Ensenada, B. C. México. Alfonso Navarro Espinoza
Junio de 2022.

Tabla de Contenido

Página

Tabla de Contenido **i**
Agradecimientos **vi**
Lista de figuras **ix**
Lista de tablas **xi**
I Introducción **1**
I.1 Motivación . . . 2
I.2 Planteamiento del problema . . . 3
I.2.1 Propuesta de solución . . . 3
I.3 Objetivos del trabajo de tesis . . . 4
I.3.1 Objetivo General . . . 4
I.3.2 Objetivos específicos . . . 4
I.4 Organización del manuscrito . . . 5
II Marco Teórico **6**
II.1 Semáforo . . . 6
II.1.1 Historia del semáforo . . . 7
II.1.2 Elementos e indicaciones del semáforo . . . 8
II.1.3 Distribución de los tiempos del semáforo . . . 10
II.2 Sistemas de transporte inteligente . . . 11
II.2.1 Sensores en semáforos inteligentes . . . 12
II.2.2 Predicción de flujo de tráfico . . . 13
II.2.3 Comunicaciones inalámbricas . . . 13
II.2.4 Revisión del estado del arte . . . 15
II.3 Resumen . . . 18
III Desarrollo de sistema para control remoto de semáforos inteligentes **19**
III.1 Descripción general . . . 19
III.2 Microcontrolador STM32F103C8T6 . . . 20
III.2.1 Sistema embebido . . . 22
III.2.2 Reloj en tiempo real . . . 23
III.3 Módulos de comunicación inalámbrica XBee 3 . . . 28
III.3.1 Enlace de XBee con computadora central . . . 32
III.3.2 Enlace de XBee con sistema embebido . . . 32
III.4 Diodo Emisor de Luz (LED) . . . 36
III.4.1 Modos de funcionamiento del semáforo inteligente . . . 37
III.5 Resumen . . . 43
IV Contador ultrasónico de vehículos e interfaz gráfica **44**
IV.1 Contador ultrasónico de vehículos . . . 44
IV.1.1 Sensor ultrasónico GY-US42 V2 . . . 47

Tabla de Contenido (Continuación)

Página

IV.1.2 Comunicación inalámbrica y algoritmo del contador 48
IV.2 Desarrollo interfaz gráfica de usuario 52
IV.3 Resumen 56
V Algoritmos Machine Learning para predicción del flujo vehicular 57
V.1 Base de datos 57
V.2 Preprocesamiento de los datos 57
V.3 Redes Neuronales Recurrentes 60
V.3.1 Diseño 60
V.3.2 Entrenamiento 61
V.4 Modelos de regresión 61
V.4.1 Regresor Perceptrón Multicapa 62
V.4.2 Regresión Lineal 63
V.4.3 Métodos de ensamble 64
V.4.4 Regresor Potenciación de Gradiente 65
V.4.5 Regresor Gradiente Estocástico 66
V.4.6 Regresor Bosques Aleatorios 66
V.5 Resumen 67
VI Resultados 69
VI.1 Prototipo electrónico 69
VI.2 Métricas de rendimiento algoritmos ML 74
VI.3 Resumen 79
VII Conclusiones generales 80
VII.1 Trabajo futuro 81
A Publicaciones derivadas del trabajo de tesis 90

Lista de figuras

Figura		Página
1	Escenario de solución propuesto.	4
2	Tipos de soporte de los semáforos, poste y ménsula. (Fuente: SCT. Manual de Señalización Vial y Dispositivos de Seguridad. México, 2014.)	9
3	Diagrama del sistema para control de semáforos.	20
4	Microcontrolador STM32F103C8T6.	21
5	Configuración periféricos del Microcontrolador.	22
6	Placa de desarrollo Blue Pill.	23
7	Programador ST-LINK V2.	23
8	Configuración RCC.	24
9	Configuración RTC.	24
10	Interrupción Alarma.	24
11	Configuración de Reloj.	25
12	XBee 3 antena PCB.	30
13	Diagrama de conexión básica del XBee 3.	30
14	Adaptador Ft232rl.	32
15	Configuración serial.	33
16	Configuración Acceso Directo a Memoria.	33
17	Configuración de Interrupción Global.	33
18	(**a**) Símbolo LED, (**b**) LED	37
19	Fases intersección tipo 2.	42
20	Medición de distancia utilizando sensor ultrasónico (Prasetyo *et al.*, 2018).	44
21	Sensores ultrasónicos en estacionamientos inteligentes.	45
22	Sensores ultrasónicos en autos modernos.	46
23	Funcionamiento del contador ultrasónico de vehículos.	46
24	Ejemplificación de las mediciones.	47
25	Sensor ultrasónico GY-US42V2.	47
26	Medidas semáforo.	48
27	Diagrama de conexión I2C de múltiples sensores.	49
28	Comandos para cambio de dirección.	49
29	Diagrama de algoritmo para conteo vehicular.	50
30	Interfaz gráfica.	53
31	Cuadro combinado desplegable.	53
32	Archivo CSV con información de las intersecciones.	53
33	Ventana para añadir intersección.	54
34	Ventana para eliminar intersección.	54
35	Cadena de texto enviada.	55

Lista de figuras (Continuación)

Figura		Página
36	Ventana modo de emergencia.	55
37	Archivo CSV generado.	56
38	Diferencias entre la sustitución de ceros aplicando una media general y una media móvil. (**a**) Media General, y (**b**) Media Móvil.	59
39	Arquitectura de las redes neuronales recurrentes. (**a**) LSTM-NN, (**b**) GRU-NN.	60
40	Rendimiento del entrenamiento de ambas redes neuronales. (**a**) LSTM NN, (**b**) GRU NN.	61
41	Arquitectura de Perceptrón Multicapa (Guerillot y Bruyelle, 2017)	62
42	Regresión lineal (Tran, 2019)	64
43	Representación esquemática de Regresor Potenciación de Gradiente (Baturynska y Martinsen, 2021)	65
44	Diagrama de flujo de Regresor Bosques Aleatorios (Rodriguez-Galiano *et al.*, 2016)	67
45	Diagrama de flujo del método propuesto.	68
46	Prototipos a nivel protoboard.	69
47	Estado de luz verde. (a) Dirección Norte, (b) Dirección Sur, (c) Dirección Este, (d) Dirección Oeste.	70
48	Estado de luz amarillo. (a) Dirección Norte, (b) Dirección Sur, (c) Dirección Este, (d) Dirección Oeste.	70
49	Estado de luz verde. (a) Dirección Norte-Sur, (b) Dirección Este-Oeste.	71
50	Estado de luz amarillo. (a) Dirección Norte-Sur, (b) Dirección Este-Oeste.	71
51	Modo nocturno.	72
52	Control de emergencia de las intersecciones.	73
53	Prototipo de contador ultrasónico.	73
54	Comparación de los tiempos de entrenamiento.	77
55	Comparación de los modelos de predicción del flujo de tráfico para un día de prueba. Predicción del flujo de tráfico del carril 1 (**a**), carril 2 (**b**), carril 3 (**c**) y carril 4 (**d**).	78
56	Comparación de los modelos de predicción del flujo de tráfico durante toda una semana. Predicción del flujo de tráfico del carril 1 (**a**), carril 2 (**b**), carril 3 (**c**) y carril 4 (**d**).	79

Lista de tablas

Tabla		Página
I	Características del microcontrolador STM32F103C8T6.	21
II	Características del módulo XBee 3.	29
III	Características técnicas del sensor ultrasónico GY-US42 V2.	48
IV	Comparación de las métricas de rendimiento utilizando el primer conjunto de datos (Axenie y Bortoli, 2020).	75
V	Métricas de rendimiento utilizando el segundo conjunto de datos (PeMS) Fu *et al.* (2017).	76

Capítulo I

Introducción

El presente trabajo de tesis de maestría está enfocado en el desarrollo de un prototipo de controlador de semáforo inteligente capaz de comunicarse de forma inalámbrica con otros semáforos, ser monitoreado y configurado desde una computadora central. Esto debido a una necesidad actual del municipio de Ensenada para reducir los congestionamientos en horas pico, para lo cual se dota al controlador de un sistema embebido y un módulo de comunicación inalámbrica, y se desarrolla una interfaz gráfica de usuario que permite programar los tiempos de los semáforos y controlarlos en situaciones de emergencia de manera remota; la aplicación utiliza una base de datos con la información de las intersecciones como el nombre, tipo y la dirección del dispositivo de comunicación inalámbrica, la base de datos puede ser modificada desde la misma interfaz al agregar y eliminar intersecciones. Adicionalmente, se presenta el diseño de un sistema para detectar la cantidad de vehículos que cruzan por las distintas vías de una intersección mediante el uso de sensores ultrasónicos, dichas mediciones se envían de manera inalámbrica a una central en la cual se almacenen las mediciones en un archivo de valores separados por coma (*.csv) etiquetado con fecha y hora de la medición. Finalmente, se proponen y evalúan distintos algoritmos Machine Learning (ML) para la predicción del flujo vehicular basados en las lecturas de flujos anteriores, esto con el fin de adaptar los tiempos de manera anticipada con base en las predicciones.

I.1 Motivación

En la actualidad, debido al acelerado crecimiento de la población y, como consecuencia, la cantidad de automóviles en las ciudades aunada a las limitaciones tecnológicas de las señales de control de tráfico han hecho del congestionamiento vehicular uno de los grandes problemas de la vida moderna. Esta problemática se da principalmente en las avenidas principales en los horarios tanto de entrada como de salida de las escuelas y los trabajos. Los principales afectados son los conductores y usuarios del transporte público, que pierden gran cantidad de tiempo en el tráfico y son propensos a sufrir accidentes, sin embargo, se tienen consecuencias negativas para todos los habitantes de la ciudad. Los impactos negativos se ven reflejados en el medio ambiente a manera de contaminación del aire por la emisión de gases de los vehículos y contaminación acústica provocada por el ruido generado por los vehículos, en la salud provocados principalmente por el estrés que trae consigo el estar detenido en el tránsito y en la economía al representar tiempo perdido para los usuarios y un mayor gasto en los costos de operación del vehículo, particularmente combustible (Bull y Thomson, 2002). La optimización en el control del tránsito permite a los habitantes gozar de una mayor calidad de vida al perder menos tiempo cuando se transportan de un lugar a otro.

Para un control más eficiente del tráfico es necesario conocer el estado del mismo en tiempo real, así como la capacidad de llevar a cabo acciones en consecuencia. Aquí es donde entran los Sistemas de Transporte Inteligente (ITS, por sus siglas en inglés), como un componente crítico de la infraestructura en las ciudades inteligentes (Chen *et al.*, 2021). Utilizando la información del big data y las tecnologías de comunicación, los ITS pueden proporcionar un análisis de la infraestructura vial en tiempo real y un control del tráfico más eficiente (Boukerche y Wang, 2020b).

I.2 Planteamiento del problema

En los últimos años, se ha registrado un aumento considerable en el tráfico de Ensenada, sin embargo, el último estudio para optimizar el tránsito y que la población llegue de un punto a otro sin una demora excesiva del que se tiene conocimiento data del año 2009 (Garcia, 2021). Existen problemas de congestión evidentes en la avenida Reforma, en vías secundarias y primarias, lo que representa una seria problemática, ya que esta es una de las arterias principales del municipio. Cabe mencionar que la mayoría de los semáforos de la ciudad de Ensenada tienen 20 años de edad y su acceso a configuración y programación de tiempos de estados se realiza de forma muy austera, un operador, tiene que ir físicamente a cada semáforo para actualizar los tiempos de los estados, en ciertas temporadas del año, lo cual es un proceso lento y tedioso para los operadores. Los tiempos de los estados programados son incapaces de adaptarse a las situaciones del tráfico en tiempo real, como las obras que se llevan a cabo actualmente, lo que provoca que en los horarios pico del día, exista congestionamiento vehicular. También, la falta de sincronización de los semáforos es un factor determinante en el aumento de tráfico en la ciudad. Por tal motivo, las autoridades municipales, se acercaron a la Universidad Autónoma de Baja California (UABC) para solicitar el servicio de modernización y sincronización de los semáforos de Ensenada.

I.2.1 Propuesta de solución

Para dar solución a este problema, se propone el diseño y construcción de un prototipo de controlador de semáforo inteligente capaz de ser programado y controlado de manera remota, además de una red inalámbrica de sensores ultrasónicos para contabilizar el número de vehículos que transitan en cada uno de los sentidos de las intersecciones, asimismo se plantea el uso de algoritmos de aprendizaje automático para la predicción del flujo vehicular, los cuales basados en las lecturas obtenidas por medio de los sensores

permiten anticipar el comportamiento del tráfico y llevar a cabo las acciones necesarias para reducir congestionamientos. En la figura 1 se aprecia el escenario de solución propuesto.

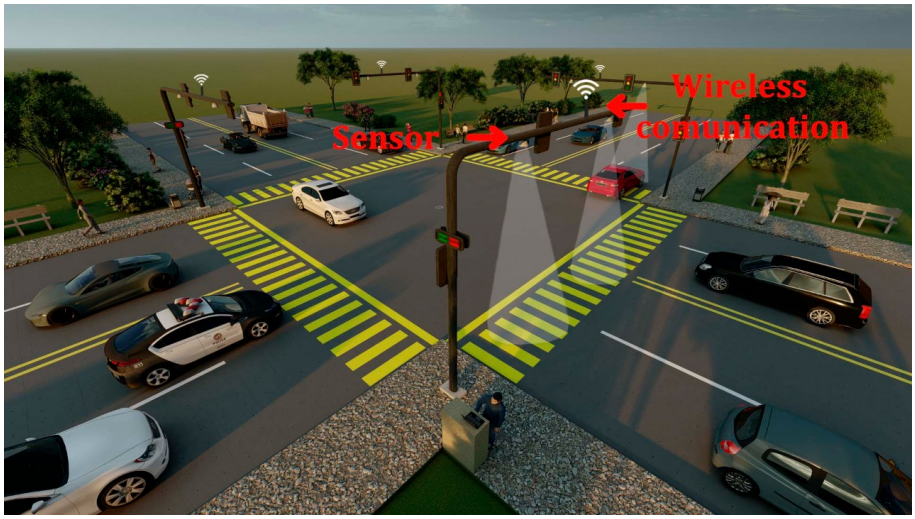


Figura 1: Escenario de solución propuesto.

I.3 Objetivos del trabajo de tesis

I.3.1 Objetivo General

El objetivo general de este trabajo de tesis es: Diseñar y construir un controlador de semáforo con capacidad de comunicarse con una computadora central y otros semáforos de manera inalámbrica para una mejor sincronización y, de esta manera, controlar con una mayor eficiencia el flujo vehicular, así mismo se contempla diseñar un contador vehicular y proponer algoritmos machine learning para predecir el flujo de tráfico en una intersección.

I.3.2 Objetivos específicos

- Caracterizar los sensores.
- Evaluar distintos sistemas embebidos.

- Seleccionar el sistema embebido.
- Examinar y definir el protocolo de comunicación inalámbrica.
- Seleccionar bases de datos para la predicción de flujo vehicular.
- Seleccionar y evaluar múltiples algoritmos ML.
- Diseñar el algoritmo del controlador principal.
- Desarrollar el software embebido del controlador principal.
- Sincronizar la comunicación de los semáforos con una computadora central.
- Evaluar el funcionamiento y desempeño de los semáforos.

I.4 Organización del manuscrito

El presente trabajo de tesis está compuesto por siete capítulos. En el **capítulo II** se presenta el marco teórico referente a los semáforos, los sistemas de transporte inteligente y los elementos que los conforman. En el **capítulo III** se describen los elementos que conforman el sistema para el control de semáforos. En el **capítulo IV** se describen el contador ultrasónico de vehículos y la interfaz gráfica de usuario desarrollada para el control de las intersecciones. En el **capítulo V** se presentan múltiples algoritmos ML para la predicción del flujo de tráfico. En el **capítulo VI** se presentan los resultados obtenidos del diseño del sistema y las métricas de rendimiento de los algoritmos. Finalmente, en el **capítulo VII** se presentan las conclusiones generales y los trabajos a futuro.

Capítulo II

Marco Teórico

II.1 Semáforo

La definición etimológica de semáforo según la RAE proviene del griego sema que significa señal y de foros que significa portador, por lo que un semáforo es "lo que lleva las señales". En castellano, hace siglos, se llamaba semáforos a las torres de señales, desde las que por medio de señales ópticas se comunicaban las noticias importantes, más rápido que con un caballo al galope. También se conocía así a las estaciones desde las que se trasmitían las señales del telégrafo óptico, que por lo general eran establecidas en las costas y puertos, y tenían como propósito dar a conocer las llegadas y maniobras de los buques o bien darles a conocer avisos urgentes. El semáforo consistía en un elevado mástil en el cual los vigías efectuaban las señales por medio de travesaños con bolas o banderas y, si era de noche, con linternas (Salvat, 1914).

Actualmente, el semáforo, también conocido como señal de control de tráfico, se define como un dispositivo eléctrico de señales luminosas que es normalmente ubicado en las intersecciones viales para regular el tránsito de vehículos y peatones, mediante indicaciones visuales de luces de colores universalmente aceptados, como lo son el verde, el amarillo y el rojo. Su finalidad principal es la de permitir el paso, alternadamente, a las corrientes de tránsito que se cruzan, permitiendo el uso ordenado y seguro del espacio disponible (Rafael *et al.*, 2018).

II.1.1 Historia del semáforo

El primer semáforo se instaló en Londres en el año de 1868, fue diseñado por John Peake Knight e imitaba a las señales de ferrocarril y únicamente usaba las luces de gas rojas y verdes por la noche. Un par de zumbidos señalaban que el tráfico que podía avanzar era el de la avenida y un solo zumbido indicaba que era el tráfico de la calle. Este primer semáforo fue todo un éxito, sin embargo, este duró poco debido a una explosión provocada por una filtración de gas en uno de los focos, costándole la vida al policía encargado de su custodia y funcionamiento.

La necesidad de controlar el tránsito en los Estados Unidos surge como consecuencia al rápido crecimiento del tráfico vehicular que se dio tras la producción en serie del modelo T de Henry Ford en el año de 1913 (McShane, 1999). Como solución a esa creciente problemática en el año de 1914 en la ciudad de Cleveland se colocó el primer semáforo eléctrico. Este mantenía, al igual su antecesor londinense, los dos colores clásicos del semáforo y contaba también con un zumbido que, en lugar de respaldar la indicación luminosa, señalaba el cambio del verde al rojo. Años más tarde, en 1920, el zumbido se sustituyó por la luz de color ámbar gracias a un nuevo diseño de William Potts. Este nuevo diseño permitía advertir de una mejor manera al conductor sobre el inminente cambio a la luz roja. Los primeros semáforos debían ser controlados de manera manual por un oficial de tránsito que tenía que estar en una cabina sobre la acera, hasta que a partir del año de 1924 gracias a los avances de la ingeniería se implementaron semáforos automáticos con temporizadores, ese mismo año, en México, se instalaron los primeros semáforos mecánicos constituidos por un tubo con dos letreros en forma de cruz, que decían Adelante y Alto, y en el año 1932 fueron puestos al servicio los primeros semáforos eléctricos (Cantú, 1976). En 1961 se implementa un nuevo elemento, el muñeco que representaba al peatón.

Los semáforos han ido evolucionando con el paso del tiempo y actualmente se están

utilizando lámparas led para la señalización luminosa, puesto que las lámparas de LED utilizan solo 10 % de la energía consumida por las lámparas incandescentes, tienen una vida estimada 50 veces superior, y por tanto, generan un importante ahorro de energía y de mantenimiento, satisfaciendo el objetivo de conseguir una mayor fiabilidad y seguridad pública.

II.1.2 Elementos e indicaciones del semáforo

El semáforo consta de una serie de elementos físicos, los cuales (Rafael *et al.*, 2018) define como se enumeran a continuación :

- **Cabeza:** Es la armadura que contiene las partes visibles del semáforo.
- **Soporte:** Son las estructuras utilizadas para sujetar la cabeza del semáforo y tienen como función situar los elementos luminosos del semáforo en la posición en donde el conductor y el peatón tengan la mejor visibilidad y puedan observar sus indicaciones. Según el Manual de Señalización Vial y Dispositivos de Seguridad de la Secretaría de Comunicaciones y Transporte SCT de México, en su sexta edición del año 2014, existen dos tipos de soporte para los semáforos: de soporte tipo poste y de soporte tipo ménsula, con las ubicaciones transversales y alturas ilustradas en la figura 2.
- **Lente:** Es la parte de la unidad óptica que por refracción dirige la luz proveniente de la lámpara y de su reflector en la dirección deseada.
- **Cara:** Es el conjunto de unidades ópticas (lente, reflector, lámpara y portalámpara) que están orientadas en la misma dirección. En cada cara del semáforo existirán como mínimo dos, usualmente tres, o más unidades ópticas para regular uno o más movimientos de circulación.

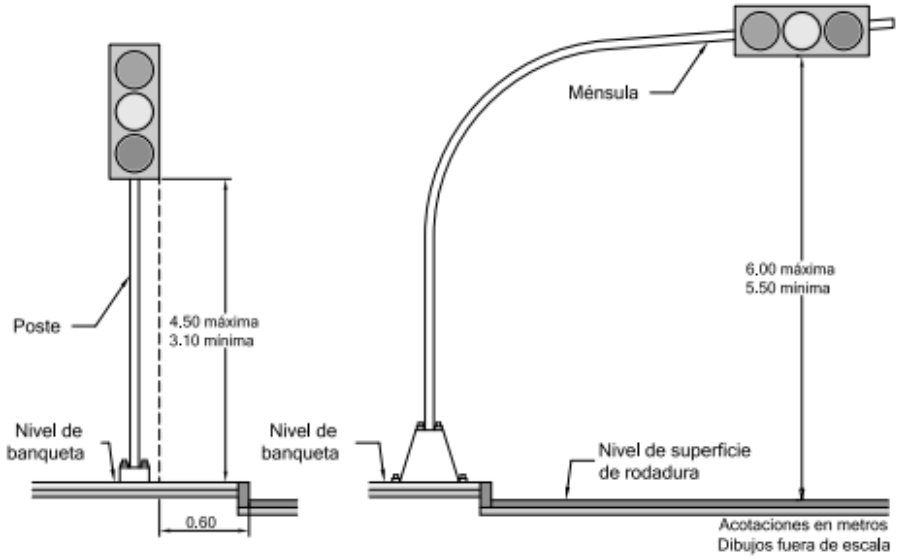


Figura 2: Tipos de soporte de los semáforos, poste y ménsula. (Fuente: SCT. Manual de Señalización Vial y Dispositivos de Seguridad. México, 2014.)

(Rafael *et al.*, 2018) describió las indicaciones de los colores de los semáforos de la siguiente manera:

1. **Rojo fijo:** Los conductores se detendrán antes de la línea de parada. Los peatones no cruzarán la vía, a menos que algún semáforo les dé la indicación de paso.

2. **Amarillo fijo:** Advierte a los conductores que está a punto de aparecer la luz roja y que el flujo vehicular que regula la luz verde debe detenerse. Además, avisa a los peatones que no disponen de tiempo suficiente para cruzar, excepto cuando exista algún semáforo indicándoles que pueden realizar el cruce. Sirve para despejar el tránsito en una intersección y para evitar frenadas bruscas.

3. **Verde fijo:** Los conductores podrán seguir de frente o dar vuelta a la derecha o a la izquierda, a menos que una señal prohíba dichas vueltas. Los peatones que avancen hacia el semáforo podrán cruzar, a menos que algún otro semáforo les indique lo contrario.

4. **Rojo intermitente:** Los conductores harán alto obligatorio y se detendrán antes de la línea de parada. Se empleará en el acceso a una vía principal.

5. **Amarillo intermitente:** Los conductores ejecutarán el cruce con precaución. Se empleará en la vía que tenga la preferencia.

6. **Verde intermitente:** Advierte a los conductores el final del tiempo de luz verde.

II.1.3 Distribución de los tiempos del semáforo

En el análisis del control de intersecciones con semáforos y en los requisitos para la distribución de sus tiempos, es necesario precisar algunos términos básicos o parámetros de tiempo, (Rafael *et al.*, 2018) los define de la siguiente manera:

- **Indicación de señal:** es el encendido de una de las luces del semáforo o una combinación de varias luces de manera simultánea.

- **Ciclo o longitud de ciclo:** tiempo requerido para una secuencia completa de todas las indicaciones de señal del semáforo.

- **Intervalo:** cualquiera de las diversas divisiones del ciclo, durante la cual las indicaciones de señal del semáforo no cambian.

- **Fase:** parte del ciclo asignada a cualquier combinación de uno o más movimientos que reciben simultáneamente el derecho de paso, durante uno o más intervalos. Es la selección y ordenamiento de movimientos simultáneos. Una fase puede significar un solo movimiento vehicular, un solo movimiento peatonal, o una combinación de movimientos vehiculares y peatonales. Una fase comienza con la pérdida del derecho de paso de los movimientos que entran en conflicto con los que lo ganan. Un movimiento pierde el derecho de paso en el momento de aparecer la indicación amarilla.

- **Secuencia de fases:** orden predeterminado en que ocurren las fases del ciclo.
- **Intervalo de despeje o todo rojo:** tiempo de exposición de una indicación roja para todo el tránsito en la intersección. Es utilizado en la fase que recibe el derecho de paso después del amarillo de la fase que lo pierde, con el fin de dar un tiempo adicional que permita a los vehículos, que pierden el derecho de paso, despejar la intersección antes de que los vehículos, que lo ganan, reciban el verde. Se aplica sobre todo en aquellas intersecciones que sean demasiado anchas. También puede usarse para crear una fase exclusiva para peatones.

II.2 Sistemas de transporte inteligente

Gracias al avance de las tecnologías como lo son el Internet de las cosas (IoT, por sus siglas en inglés) y la inteligencia artificial (IA) surgen nuevos conceptos como las ciudades inteligentes dentro la cual encontramos los ITS.

Los ITS consisten en una red de sensores que se encuentran a lo largo de las calles, recolectan datos y, en la mayoría de los casos, los envían a un servidor remoto. El servidor remoto se encarga de procesar los datos y de distribuirlos. Estos datos pueden ser utilizados para el análisis y manejo del tráfico. En el año del 2010 durante los juegos asiáticos celebrados en Guangzhou, China se da un gran avance en el ámbito de los ITS (Xiong *et al.*, 2010) empleando la tecnología para el control y la gestión del tráfico paralelo.

Dentro de las soluciones que ofrecen los ITS nos encontramos con el creciente concepto de semáforos inteligentes; en la actualidad la mayoría de los semáforos utilizados en las ciudades funcionan con ciclos estáticos de tiempo para cada una de los estados (verde, amarillo y rojo) lo que los hace poco eficientes en múltiples situaciones, ya que no se pueden adaptar a las distintas condiciones que se presentan a lo largo del

tiempo. El ejemplo más claro es estar detenido esperando la luz verde cuando no hay ningún automóvil en la calle que la tiene.

II.2.1 Sensores en semáforos inteligentes

Los semáforos inteligentes utilizan sensores para detectar la cantidad de vehículos presentes en las calles y adaptan sus estados de manera automática utilizando distintos métodos para reducir los tiempos de espera, produciendo además beneficios económicos e inclusive ambientales al reducir las emisiones de gases contaminantes a la atmósfera.

Los sensores utilizados puedes ser tanto intrusivos como no intrusivos. Dentro de los sensores intrusivos se encuentran las espiras magnéticas, sensores piezo-eléctricos, tubos neumáticos, sensores de fibra óptica y sensores geomagnéticos, los sensores no intrusivos son los radares de microondas, sensores láser, infrarrojos, ultrasónicos, acústicos y las cámaras de vídeo que por lo regular se colocan a los costados o por encima de las vías.

A continuación se muestran algunos tipos de detectores utilizados para registrar el paso de vehículos en intersecciones:

- **Presión:** Se colocan debajo del pavimento con la parte superior al nivel del mismo. Con la presión ejercida por las llantas del vehículo se cierra un circuito, que registra la presencia del vehículo.

- **Radar:** Consiste de un transmisor que emite un haz cónico de microondas. Al pasar bajo él un vehículo, parte de las ondas son reflejadas hacia la antena receptora localizada en la misma unidad indicando la presencia del vehículo.

- **Inducción:** Son probablemente, los de uso más extendido en las grandes ciudades. Por lo general, se trata de un alambre en forma de lazo rectangular o hexagonal y un amplificador. El alambre se inserta bajo el pavimento a través de una ranura. Al pasar el vehículo sobre el lazo, el campo magnético del mismo

registra la presencia de la masa metálica.

- **Vídeo:** Las imágenes de los vehículos son captadas por cámaras de vídeo sobre una amplia área de detección, las cuales son procesadas en tiempo real.

II.2.2 Predicción de flujo de tráfico

Los ITS se basan en las predicciones de tráfico como un componente crítico (Yuan y Li, 2021). El objetivo de la previsión del tráfico es predecir las condiciones futuras del tráfico en una red de transporte basándose en observaciones históricas (Li y Shahabi, 2018). Estos datos pueden ser útiles en aplicaciones de ITS, como el control de la congestión del tráfico y el control de los semáforos (Boukerche y Wang, 2020a). Por ejemplo, puede calcular la probabilidad de congestión en el segmento de carretera correspondiente y prepararse para ello con antelación (Boukerche *et al.*, 2020). La predicción del tráfico puede dividirse en dos tipos de técnicas: paramétricas, que incluyen métodos estocásticos y temporales, y no paramétricas, como los modelos de machine learning (ML), utilizados recientemente para resolver problemas de tráfico complejos. La revisión realizada en (George y Santra, 2020) encontró que los algoritmos no paramétricos superan a los algoritmos paramétricos debido a su capacidad para tratar un gran número de parámetros en datos masivos.

II.2.3 Comunicaciones inalámbricas

Se entiende por comunicación inalámbrica a aquella comunicación en la cual el emisor y receptor intercambian información utilizando el espectro electromagnético (Blázquez, 2015). La clasificación de las comunicaciones inalámbricas puede ser diferente dependiendo la documentación consultada, según el alcance se pueden establecer tres grandes grupos: **Redes de área personal inalámbrica** (WPAN: wireless personal area networks), **Redes de área local inalámbrica** (WLAN: wireless local area

networks) y **Redes de área extendida inalámbrica** (WWAN: wireless wide area networks).

Las comunicaciones inalámbricas son fundamentales para los ITS, al permitir la intercomunicación entre los elementos que conforman el sistema a manera de sincronizar y optimizar su funcionamiento, asimismo, permite reunir de manera remota los datos recolectados. A continuación se muestran algunas de las tecnologías inalámbricas más ampliamente utilizadas en los ITS:

- **Bluetooth:** Bluetooth es una especificación regulada por el grupo de trabajo IEEE 802.15.1, que permite la transmisión de voz y datos entre diferentes dispositivos mediante un enlace de radiofrecuencia en la banda ISM de 2,4 GHz.

- **NFC:** La tecnología near field communication (NFC) permite la transmisión de datos de una manera simple entre diferentes dispositivos mediante un enlace de radiofrecuencia en la banda ISM de 13,56 MHz.

- **Zigbee:** Zigbee es un estándar de comunicaciones inalámbricas, regulado por el grupo de trabajo IEEE 802.15.4 en el 2004, que permite habilitar redes inalámbricas con capacidades de control, y monitorizar que sean seguras, de bajo consumo energético y de bajo coste de procesador, de manera bidireccional.

- **IEEE 802.11:** El IEEE 802.11 es una familia de estándares para redes locales inalámbricas desarrollada por el IEEE, que fue definida en 1997 (en el año 1999 se definieron los estándares 802.11a y 802.11b). El estándar garantiza la interoperabilidad entre diferentes fabricantes.

- **GSM:** Acrónimo de Global System for Mobile Communication (o Sistema Global para las Comunicaciones Móviles), el GSM consiste en un sistema estándar de telefonía móvil digital. Esta tecnología permite enviar y recibir emails, navegar

por Internet o acceder a otros programas disponibles en otro dispositivo. El Group Special Mobile fue el organismo que se encargó de la configuración técnica de una normativa de transmisión y recepción para la telefonía móvil.

II.2.4 Revisión del estado del arte

En la literatura actual, múltiples trabajos utilizan sensores para agilizar el tráfico. En el trabajo de (Serrano *et al.*, 2005) desarrollan un semáforo inteligente capaz de capturar, con ayuda de la visión artificial, la presencia o ausencia de vehículos y peatones, priorizando la seguridad de los peatones. Los autores (Chavan *et al.*, 2009) usan una red de sensores infrarrojos para controlar los semáforos de múltiples intersecciones, informar a las personas el estado del tránsito y controlar el tráfico en situaciones de emergencia utilizando la tecnología GSM. Los autores (Bhaskar *et al.*, 2016) reportan el uso de sensores inductivos para medir la densidad del tráfico, radiotransmisores para detectar vehículos de emergencia y sensores IR para detectar autos que obstruyan los cruces peatonales. En el trabajo de (Ghazal *et al.*, 2016) se implementa el diseño de un sistema empleando un microcontrolador PIC 16F877A que evalúa la densidad del tráfico utilizando sensores Infrarrojos, además de usar módulos de radiofrecuencia XBee para comunicar los vehículos de emergencia con el controlador. En (Li *et al.*, 2017), los autores proponen un diseño basado en sensores ultrasónicos para detectar la densidad del tráfico en cada vía de una intersección y cambiar los tiempos de los semáforos basado en las lecturas. Sensores infrarrojos y cámaras son utilizados por (Saleh *et al.*, 2018) para detectar la cantidad de vehículos y controlar el tráfico, además de implementar sensores magnéticos para detectar carros que se pasen el semáforo en rojo y capturar una imagen del infractor con las cámaras. Los autores (Diaz *et al.*, 2018) describen el diseño y la implementación de un semáforo inteligente basado en IoT empleando una Raspberry Pi y un sensor PIR el cual detecta la presencia de vehículos en cada

vía, en caso de no haber vehículos en una calle y en la otra si, producir el cambio de luz. En (Nguyen-Ly *et al.*, 2019), proponen un semáforo inteligente a bajo costo utilizando transreceptores ópticos (láser y fotorresistencia), este sistema es diseñado principalmente para motocicletas, ya que estas son el principal medio de transporte en el país de Vietnam. Los autores (Manasi *et al.*, 2020) proponen una nueva estrategia para las señales de tránsito inteligente, utilizando un par de sensores de imagen en cada vía, se centran en la eficiencia y eficacia de la gestión al tomar el tiempo de recorrido de los vehículos a través de una región determinada. Los autores (Alaidi *et al.*, 2020) llevan a cabo el diseño de un semáforo inteligente basado en Arduino y sensores infrarrojos, para que una luz cambie a verde se deben alcanzar la cantidad de 30 carros esperando en esa vía; si se presenta esa cantidad en más de una se utiliza la prioridad FIFO. Este método redujo el tiempo de espera a más de la mitad en una intersección en Kut ciudad de Irak. Un sistema de adquisición de datos del tráfico basado en sensores ultrasónicos, utilizando un montaje vertical o sobre la vía para limitar interferencias de humanos u otros objetos no vehiculares es propuesto por (Appiah *et al.*, 2020).

Con respecto a los algoritmos ML para la predicción del flujo de tráfico, en el trabajo de (Alam *et al.*, 2019) se evalúan cinco algoritmos ML para predecir el volumen total del flujo de tráfico en la ciudad de Oporto; los algoritmos son: Regresión lineal, Regresión de optimización mínima secuencial, Perceptrón multicapa, árbol de regresión M5P y Bosque aleatorio. Los resultados experimentales muestran que el árbol de regresión M5P supera a los demás modelos de regresión. Los autores en (Li *et al.*, 2021) informan de algunos métodos ML multimodelos para la estimación del flujo de tráfico a partir de datos de coches. En particular, evaluan la capacidad del regresor de proceso gaussiano (GPR) para abordar esta cuestión.

El Deep Learning (DL) como subconjunto del ML utiliza redes neuronales multicapa expuestas a muchos datos para entrenarse. Esta capacidad de los modelos de DL para

extraer conocimientos de sistemas complejos los ha convertido en una solución robusta y viable en el campo de los ITS (Haghighat *et al.*, 2020). Una Red Neural Perceptrón Multicapa (MLP-NN) es presentada en (Ferreira *et al.*, 2019) y (Hosseini *et al.*, 2014), esta última con una técnica de información mutua para predecir el flujo de tráfico. Las simulaciones muestran una disminución del error de previsión en comparación con los resultados de los modelos de media y media móvil integrada autorregresiva (ARIMA) que utilizan datos de tráfico de períodos anteriores. La red neuronal de retropropagación (BPNN) es una de las arquitecturas más típicas de las redes neuronales y se utiliza ampliamente en muchas tareas de predicción y clasificación. Los autores (Jiang *et al.*, 2021) proponen un sistema de control de señales de tráfico urbano basado en la predicción del flujo de tráfico empleando BPNN. También (Chen *et al.*, 2016) utiliza BPNN para predecir los volúmenes de tráfico futuros en el diseño de un sistema de control de semáforos junto con un algoritmo genético para la optimización del tiempo. Con este método, la tasa de espera media se reduce en casi un 30% en comparación con el sistema de control de semáforos de tiempo fijo. Siguiendo este método, la combinación de un algoritmo genético y una red neuronal da lugar a la denominada Red Neuronal Genética (Wang, 2021). (Lawe y Wang, 2016) presentan un método de red neuronal de aprendizaje profundo para optimizar el flujo de tráfico y reducir la congestión en las intersecciones clave mediante el uso de datos históricos de todos los movimientos de una intersección, series de tiempo y variables ambientales como las características de entrada. La salida se introduce en una ecuación de retardo que genera los mejores tiempos verdes para gestionar el retraso del tráfico. Las redes neuronales recurrentes (RNN) tienen un estado interno que puede representar la información del contexto, mantienen la información sobre las entradas pasadas durante un período de tiempo y se utilizan normalmente para capturar secuencias dinámicas de datos, las RNN basadas en los métodos DL Long Short-Term Memory (LSTM) y Gated Recurrent Units (GRU)

en (Fu *et al.*, 2017) superan el modelo ARIMA; además, los autores informan que este es el primer uso de GRU en la predicción del flujo de tráfico. Actualmente, los modelos GRU siguen utilizándose para el desarrollo de la predicción inteligente del flujo de tráfico (Hussain *et al.*, 2021).

II.3 Resumen

En este capítulo se presentan las definiciones y los elementos básicos del semáforo, así como un panorama general de los sistemas de transporte inteligente, y como la sinergia de estos dos da lugar a los semáforos inteligentes. Se mostró el papel que juegan los distintos tipos de sensores, algoritmos para la predicción del flujo de tráfico y comunicaciones inalámbricas en el control óptimo del tránsito vehicular. Además, se presentó una revisión del estado del arte sobre trabajos relacionados con este tema de tesis, desde diferentes enfoques como lo son el hardware y los algoritmos de aprendizaje automático.

Capítulo III

Desarrollo de sistema para control remoto de semáforos inteligentes

III.1 Descripción general

El sistema para el control de los semáforos desarrollado en este trabajo de tesis se subdivide en 3 partes: La primera parte es la central, en este lugar un operador podrá configurar los tiempos de los semáforos y controlarlos en caso de emergencia.

- En la computadora central se ejecuta la aplicación con la interfaz gráfica en la que el usuario puede programar los tiempos de los semáforos.
- El módulo de comunicación inalámbrica se utiliza para enviar y recibir información de las intersecciones y la red inalámbrica de sensores.
- Con los datos recibidos de las intersecciones se crea una base de datos que puede ser utilizada para entrenar un algoritmo de Machine Learning.

La segunda parte es el controlador de semáforos, está compuesto por un módulo de comunicación inalámbrica y un sistema embebido. Este se encarga de controlar los tiempos de los semáforos con lo que recibe de la central, comunicarse con cada semáforo para obtener las lecturas de los sensores y enviar las lecturas a la central.

La tercera parte es una red inalámbrica de sensores, se espera que cada semáforo cuente con un módulo XBee y sensores ultrasónicos igual al número de carriles para contabilizar los vehículos. El módulo controla los sensores ultrasónicos mediante comandos I2C para realizar mediciones de distancia y aplicar un algoritmo para el

conteo de vehículos, al recibir una señal del controlador de semáforos cada cierto tiempo envía las lecturas y reinicia el conteo.

La representación del sistema desarrollado en forma de diagrama se puede observar en la figura 3.

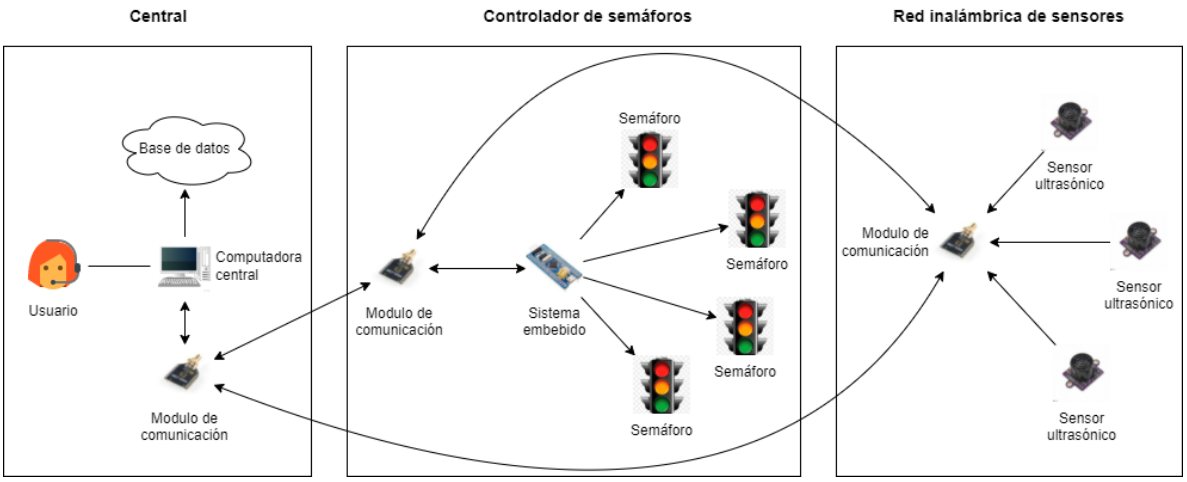


Figura 3: Diagrama del sistema para control de semáforos.

III.2 Microcontrolador STM32F103C8T6

El microcontrolador STM32F103C8T6 es el elemento principal en el controlador de semáforos, ya que es el encargado de llevar a cabo funciones primordiales del dispositivo, como lo son:

- Controlar el tiempo de encendido de las luces del semáforo.
- Establecer la comunicación con el módulo XBee.
- Solicitar las mediciones de los contadores ultrasónicos de vehículos.
- Recibir y procesar los comandos de la central.

El microcontrolador se muestra en la figura 4 y algunas de sus características se indican en la tabla I.

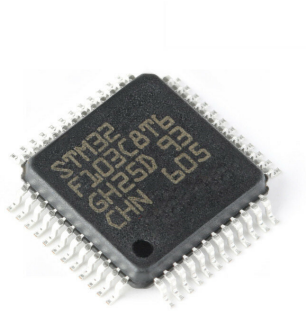


Figura 4: Microcontrolador STM32F103C8T6.

Tabla I: Características del microcontrolador STM32F103C8T6.

Propiedades	**STM32F103C8T6**
Voltaje de operación	$2 - 3.6$ V
Frecuencia de reloj	72 Mhz
Número de entradas y salidas	37
Puertos I2C	2
Puertos SPI	2
Puertos USART	3
RTC	1
Resolución ADC	12 bit
Canales ADC	10
Memoria Flash	64kb
SRAM	20kb
Resolución PWM	16bit
Debugging	Serial, JTAG
Temperatura de operación	-40 a 85°

La programación del μC se realiza en lenguaje C. La compilación, depuración y programación del código se lleva a cabo en STM32CubeIDE que es una herramienta de desarrollo que además permite configurar las entradas y salidas del microcontrolador STM32, el reloj del sistema e incluso permite conocer detalles del consumo energético según los periféricos seleccionados, estas características lo hacen una opción más robusta al momento de desarrollar prototipos. En la figura 5 se muestran los periféricos

configurados en la aplicación y se listan a continuación: pines para el programador, 12 salidas digitales para las luces de cada semáforo nombradas según es el caso, un puerto UART para la conexión de un módulo XBee de transmisión inalámbrica de datos y un RTC.

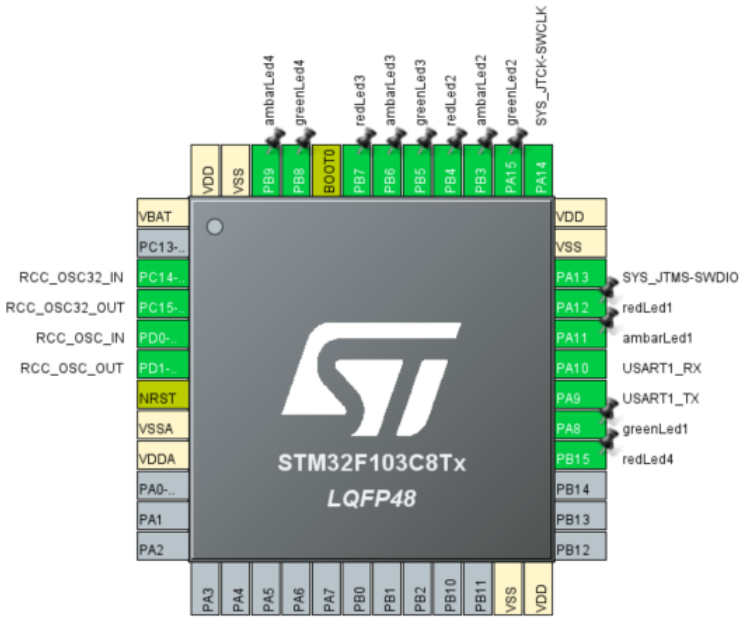


Figura 5: Configuración periféricos del Microcontrolador.

III.2.1 Sistema embebido

El sistema embebido seleccionado es conocido como Blue Pill y se muestra en la figura 6. La elección de esta placa de desarrollo es debido a que, además de contar con el microcontrolador seleccionado, tiene un bajo costo, su pequeño tamaño (similar al de un Arduino Nano), cuenta con un regulador que nos permite alimentar el microcontrolador con 5 V, se puede montar en una placa de pruebas y es posible programarlo tanto de manera serial utilizando el puerto USB micro, como con el JTAG, el cual permite depurar el código en tiempo de ejecución utilizando un debugger, en este trabajo se utilizó el programador ST-LINK V2 mostrado en la figura 7.

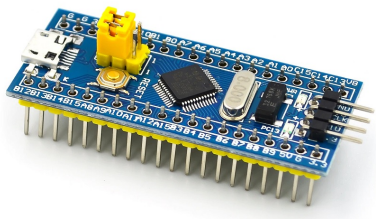


Figura 6: Placa de desarrollo Blue Pill.

Figura 7: Programador ST-LINK V2.

III.2.2 Reloj en tiempo real

La mayoría de los dispositivos STM32, como el utilizado en este trabajo, tienen RTC (Real Time Clock) incorporado que puede mantener el seguimiento de la hora y la fecha actuales, esto es posible gracias a un oscilador de cristal con una frecuencia de 32,768 kHz, la misma frecuencia utilizada en los relojes de cuarzo. El diapasón de cuarzo de estos cristales no cambia mucho de tamaño con la temperatura, por lo que esta no modifica mucho su frecuencia. El RTC puede utilizarse para cronómetros, despertadores, relojes, pequeñas agendas electrónicas y muchos otros dispositivos. Dentro del sistema desarrollado se aprovecha esta virtud del microcontrolador a manera de programar alarmas cada cierto tiempo para solicitar a la red de sensores inalámbrica sus mediciones. De esta manera, los controladores actúan como intermediarios entre la red de sensores de cada intersección y la central. Para configurar el RTC desde la aplicación STM32CubeIDE se realiza el siguiente procedimiento:

- En System Core se selecciona RCC (Reset and Clock Control) y se selecciona Crystal/Ceramic Resonator para el reloj de alta velocidad (HSE) y el reloj de

baja velocidad (LSE) como se muestra en la figura 8.

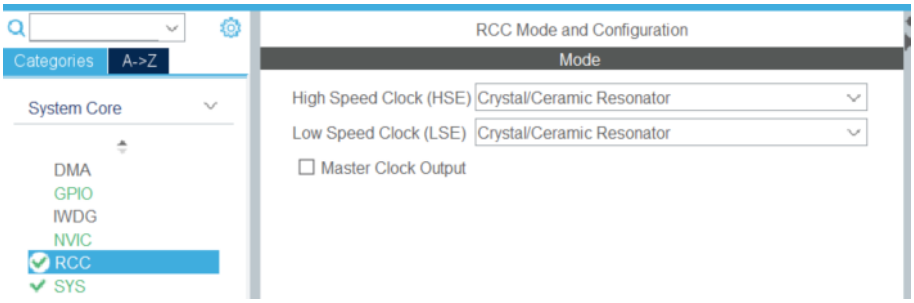


Figura 8: Configuración RCC.

- En Timers se selecciona RTC y se activan las casillas activar fuente de reloj y activar calendario (véase figura 9).

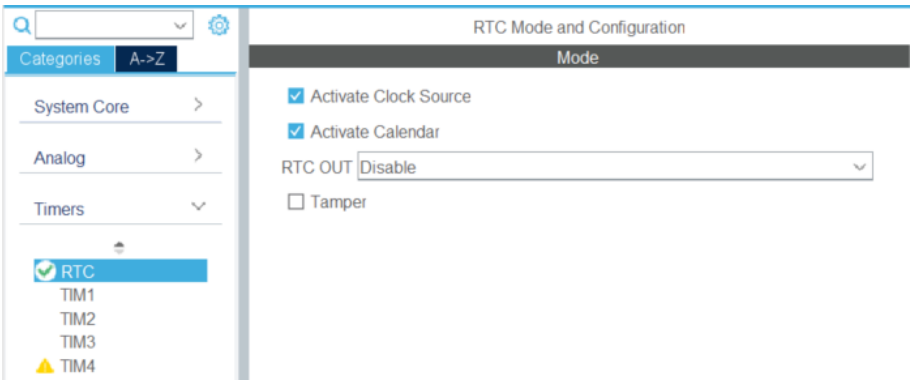


Figura 9: Configuración RTC.

- En configuración y Ajustes de NVIC (Nested Vectored Interrupt Controller) se configura una interrupción para la alarma (véase figura 10).

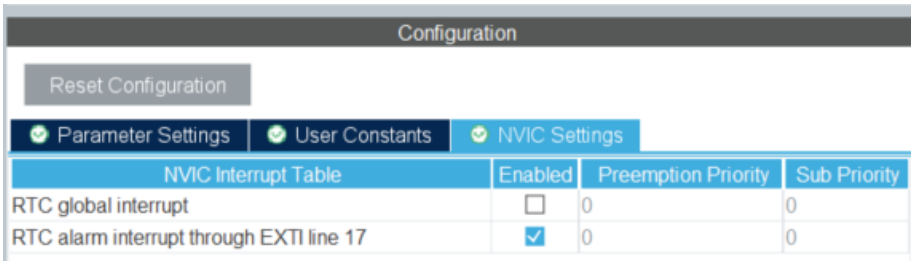


Figura 10: Interrupción Alarma.

- Se configura el reloj como se muestra en la figura 11.

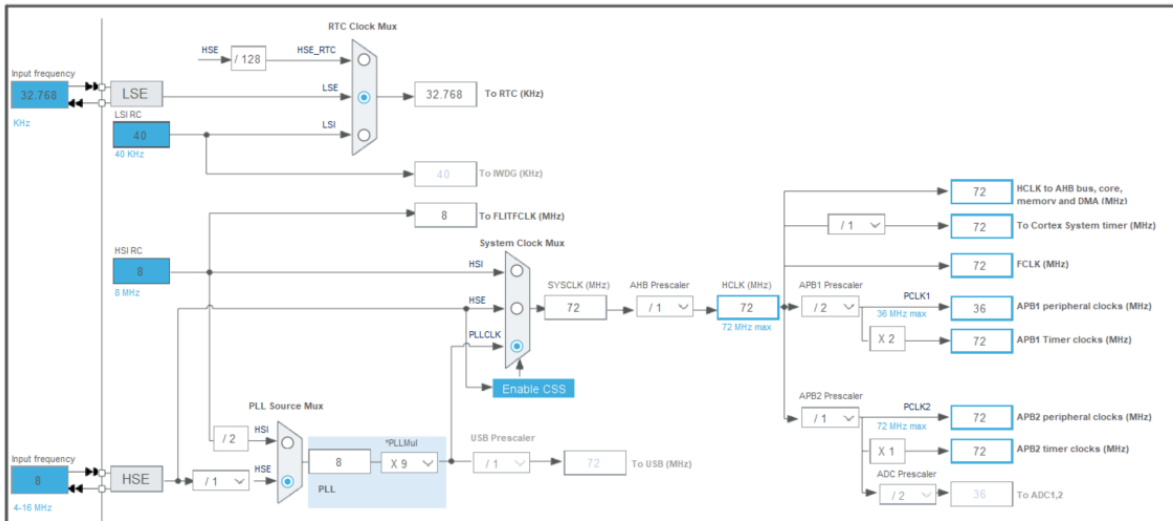


Figura 11: Configuración de Reloj.

- **Nota: Una vez configurado se recomienda no utilizar el LED integrado en la placa, ya que utiliza la misma línea que el RTC.**

El código en C básico para trabajar con el RTC y la interrupción por alarma se muestra a continuación:

- Se definen las variables para almacenar la hora y la fecha, así como una bandera que será utilizada para la alarma.

```
/* USER CODE BEGIN PV */
// Variables RTC
char time[10];
char date[10];
volatile uint8_t band = 0; //Bandera utilizada en alarma
/* USER CODE END PV */
```

- Se crea una función para convertir Byte a BCD (Binary-Coded Decimal), en otras placas STM32 esta función ya se incluye.

```
/**
* @brief  Converts a 2 digit decimal to BCD format.
* @param  Value: Byte to be converted
```

```
* @retval Converted byte
*/
static uint8_t RTC_ByteToBcd2(uint8_t Value)
{
    uint32_t bcdhigh = 0;

    while(Value >= 10){
        bcdhigh++;
        Value -= 10;
    }
    return  ((uint8_t)(bcdhigh << 4) | Value);
}
```

- Se crea una función en la que se define la hora y la fecha, además se define un registro de respaldo en caso de reinicio del sistema.

```
void set_time (void)
{
  RTC_TimeTypeDef sTime;
  RTC_DateTypeDef sDate;
  sTime.Hours = 0x10; // set hours
  sTime.Minutes = 0x20; // set minutes
  sTime.Seconds = 0x30; // set seconds
  if (HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BCD) != HAL_OK)
  {
    Error_Handler();
  }
  sDate.WeekDay = RTC_WEEKDAY_MONDAY; // day
  sDate.Month = RTC_MONTH_AUGUST; // month
  sDate.Date = 0x9; // date
  sDate.Year = 0x21; // year
  if (HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BCD) != HAL_OK)
  {
    Error_Handler();
  }
  HAL_RTCEx_BKUPWrite(&hrtc, RTC_BKP_DR1, 0x32F2); // backup
   register
}
```

- Se crea una función para obtener la fecha y la hora.

```
void get_time(void)
{
    RTC_DateTypeDef gDate;
    RTC_TimeTypeDef gTime;
    /* Get the RTC current Time */
    HAL_RTC_GetTime(&hrtc, &gTime, RTC_FORMAT_BIN);
    /* Get the RTC current Date */
    HAL_RTC_GetDate(&hrtc, &gDate, RTC_FORMAT_BIN);
```

```
    /* Display time Format: hh:mm:ss */
    sprintf((char*)time,"%02d:%02d:%02d",gTime.Hours, gTime.
   Minutes, gTime.Seconds);
    /* Display date Format: dd-mm-yy */
    sprintf((char*)date,"%02d-%02d-%2d",gDate.Date, gDate.Month,
   2000 + gDate.Year);
}
```

- La función set_alarm define una alarma cada cierto tiempo (en este caso cada minuto).

```
void set_alarm (void)
{
    RTC_TimeTypeDef sTime;
    HAL_RTC_GetTime(&hrtc, &sTime, RTC_FORMAT_BIN);

    sTime.Minutes+=1;
    if(sTime.Minutes==60){
        sTime.Hours += 1;
        if(sTime.Hours == 24){
          sTime.Hours = 0;
        }
        sTime.Minutes=0;
    }

    RTC_AlarmTypeDef sAlarm;
    sAlarm.AlarmTime.Hours = RTC_ByteToBcd2(sTime.Hours);
    sAlarm.AlarmTime.Minutes = RTC_ByteToBcd2(sTime.Minutes);
    sAlarm.AlarmTime.Seconds = 0x00;
    sAlarm.Alarm = RTC_ALARM_A;
    if (HAL_RTC_SetAlarm_IT(&hrtc, &sAlarm, RTC_FORMAT_BCD) !=
   HAL_OK){
        Error_Handler();
    }
}
```

- Se redefine el Callback de la alarma, se manda la bandera a 1 y se programa la siguiente alarma.

```
void HAL_RTC_AlarmAEventCallback(RTC_HandleTypeDef *hrtc)
{
  band = 1;
  /* Next Alarm*/
  set_alarm();
}
```

- Dentro del user code begin 2 se revisa el registro para saber si el microcontrolador

fue reiniciado y no reiniciar el reloj cada que esto ocurra, además se programa la alarma por pirmera vez.

```
if(HAL_RTCEx_BKUPRead(&hrtc, RTC_BKP_DR1) != 0x32F2){
    set_time();
}

set_alarm();
```

- Dentro del ciclo infinito se revisa el tiempo y el estado de la bandera, se programan las acciones a realizar en caso que la bandera se active.

```
get_time();

if(band == 1){
  band = 0;
  //Acciones a realizar
}
```

III.3 Módulos de comunicación inalámbrica XBee 3

El Xbee 3 es un módulo de bajo costo que permite realizar conexiones inalámbricas entre dispositivos electrónicos. Trabaja con una frecuencia de 2.4 GHz y permite crear redes de conexión punto a punto, punto a multipunto, broadcast y mesh.

La serie 3 de XBee, a diferencia de sus antecesores, permite configurar el firmware en función a la conectividad que se necesite: DigiMesh, Zigbee y 802.15.4, además de la posibilidad de configurar la funcionalidad de BLE (Bluetooth Low-Energy). Esta versión incluye soporte de MicroPython, lo que permite realizar operaciones y procesar información directamente desde el XBee. Algunas de sus especificaciones son:

- Tasa de transferencia: RF 250 Kbps, Serial hasta 1 Mbps
- Alcance en interior/urbano hasta 60 m
- Alcance exterior/RF línea vista hasta 1200 m
- Comunicación serial UART, SPI, I2C

- Configuración por comando AT o API, local o via OTA
- Montaje en formato Micro, Through-Hole, SMD
- Entradas Analógicas: 4 ADC de 10-bits
- Puertos Digitales: 15 puertos de entrada/ salida digital
- Memoria 1 MB / 128 Kb Ram (32 Kb disponibles para MicroPython)
- CPU: HCS08 hasta 50.33 MHz
- Protocolo: Zigbee 3.0
- Encriptado: 128/250 bits AES

En este trabajo de tesis, con el fin de llevar a cabo la comunicación entre una computadora central y múltiples semáforos, además de con una red de sensores inalámbrica, se utiliza el módulo XBee 3 con antena PCB que se muestra en la figura 12; el protocolo de comunicación utilizado es Zigbee 3. Las principales características técnicas del módulo se muestran en la tabla II.

Tabla II: Características del módulo XBee 3.

Característica	**Valor**
Potencia de transmisión	+8 dBm
Sensibilidad de Receptor	-103 dBm
Frecuencia	ISM 2.4Ghz
Voltaje de trabajo	2.1 – 3.6V
Corriente trasmitiendo	40mA @ 8 dBm
Corriente recibiendo	17mA

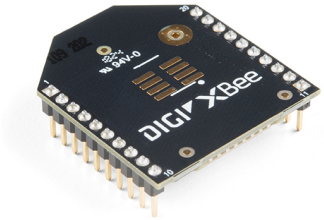


Figura 12: XBee 3 antena PCB.

En la figura 13 se presenta la conexión básica del módulo XBee, la cual consiste en los pines de comunicación serial (Tx y Rx), un voltaje de alimentación de 3.3 V y la conexión a tierra (GND).

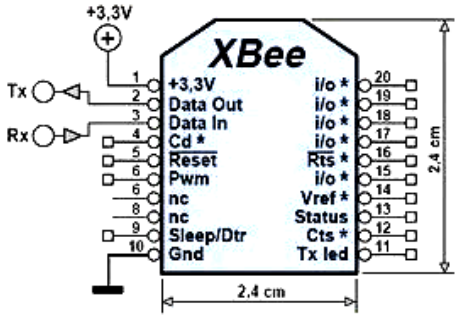


Figura 13: Diagrama de conexión básica del XBee 3.

El módulo XBee 3 cuenta con distintos modos de operación:

- **Modo transparente:** Este modo está destinado principalmente a la comunicación punto a punto, donde no es necesario ningún tipo de control. También se usa para reemplazar alguna conexión serial por cable, ya que es la configuración más sencilla posible y no requiere una mayor configuración. En este modo todo lo que ingresa por el pin 3 (Data in), es guardado en el buffer de entrada y luego transmitido, mientras que, todo lo que ingresa como paquete RF, es guardado en el buffer de salida y luego enviado por el pin 2 (Data out) (Oyarce *et al.*, 2010).

- **Modo API (Application Programming Interface):** Este modo es más complejo, pero permite el uso de frames con cabeceras que aseguran la entrega de los datos. El módulo así envía paquetes de datos contenidos en frames a otros módulos de destino, con información a sus respectivas aplicaciones, conteniendo paquetes de estado, así como el origen, RSSI (potencia de la señal de recepción) e información de la carga útil de los paquetes recibidos. Entre las opciones que permite la API, se tienen: Transmitir información a múltiples destinatarios sin entrar al modo de Comandos, recibir estado de éxito/falla de cada paquete RF transmitido e identificar la dirección de origen de cada paquete recibido. Este modo se subdivide a su vez en dos:
  - **Modo API 1 (sin escapes):** El dispositivo empaqueta todos los datos de entrada y salida de la UART en formato API, sin secuencias de escape. Se basa únicamente en el delimitador de inicio y los bytes de longitud para diferenciar las tramas API. Si se pierden los bytes de un paquete, el recuento de longitud estará desactivado, y la siguiente trama API (paquete) también se perderá.
  - **Modo API 2 (con escapes):** El dispositivo está en modo API e inserta secuencias de escape para permitir los caracteres de control. El dispositivo pasa XON (0x11), XOFF (0x13), Escape (0x7D) y el delimitador de inicio 0x7E como datos, esto para mejorar la fiabilidad, especialmente en entornos de RF ruidosos.
- **MicroPython REPL:** Este modo nos permite programar y ejecutar scripts en MicroPython utilizando los módulos XBee. REPL (Read Evaluate Print Loop) es el nombre dado al prompt interactivo de MicroPython. Usar el REPL es, con mucho, la forma más fácil de probar el código Python y ejecutar comandos.

III.3.1 Enlace de XBee con computadora central

Para la conexión del módulo XBee 3 con la computadora central se utiliza un módulo adaptador USB a serial Ft232rl el cual se muestra en la figura 14. El adaptador se conecta a un puerto USB de la computadora utilizando un cable con conector de USB a mini USB. El módulo XBee es configurado bajo el esquema de modo API 1 como coordinador.

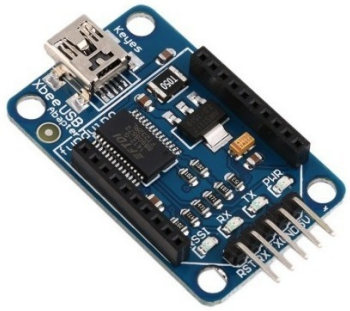


Figura 14: Adaptador Ft232rl.

III.3.2 Enlace de XBee con sistema embebido

Los módulos XBee se comunican con el microcontrolador vía puerto UART, por lo que se debe configurar esta comunicación en el microcontrolador, con ayuda de STM32CubeIDE en conectividad se selecciona USART1, modo Asíncrono y el Baud Rate deseado el cual debe ser igual al del XBee, la configuración se muestra en la figura 15, posteriormente se agrega acceso directo a memoria (DMA) tanto para envío como para recepción y una interrupción global (véase figuras 16 y 17).

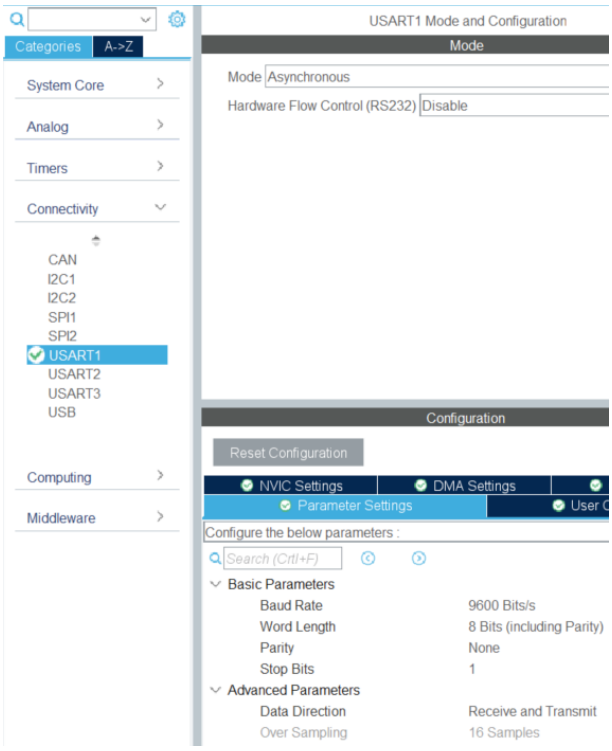


Figura 15: Configuración serial.

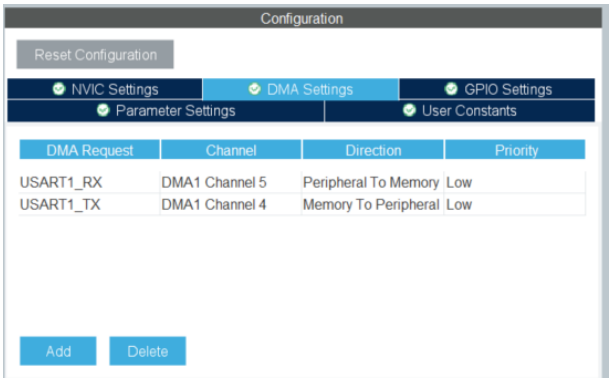


Figura 16: Configuración Acceso Directo a Memoria.

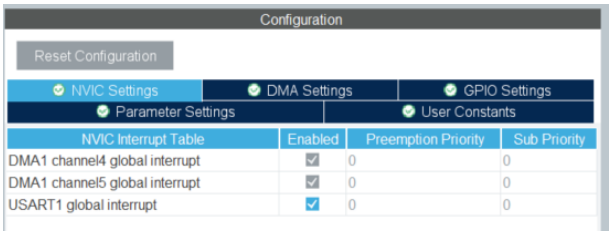


Figura 17: Configuración de Interrupción Global.

La conexión entre el módulo XBee 3 y el microcontrolador es la siguiente: Para la alimentación del módulo XBee el pin 1 se conecta a los 3.3 V y el pin 10 a GND, el pin 2 del XBee que corresponde a Data Out (Tx) se conecta al USART1_RX (PA10) del microcontrolador y el pin 3 del XBee que corresponde a Data In (Rx) se conecta al USART1_TX (PA9) del microcontrolador, los pines RX y TX del microcontrolador variarán según el puerto serial configurado, en este caso se hace mención a los pines del puerto USART1 configurado previamente. A continuación se describe el código en lenguaje C cargado en el microcontrolador para trabajar con el módulo XBee:

- Se importan las librerías necesarias:

```
/* USER CODE BEGIN Includes */
#include "string.h"
#include "stdio.h"
#include "stdlib.h"
/* USER CODE END Includes */
```

- Se definen las variables necesarias para la recepción y el envío de datos:

```
/* USER CODE BEGIN PV */
// Variables comunicacion
uint8_t newData = 0; //Bandera recepcion de datos via UART
uint8_t RxData[24];
#define RxBuf_SIZE 32
uint8_t RxBuf[RxBuf_SIZE];
#define packet_SIZE 18
uint8_t packet[packet_SIZE];
/* USER CODE END PV */
```

- Se define una función callback para el evento de recepción utilizando la interrupción por DMA, en la función se activa una bandera y se guardan los datos recibidos:

```
/* USER CODE BEGIN 0 */
void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart,
   uint16_t Size){
/* Revisa que se haya recibido informacion por el UART1, de esta
   forma se
*  pueden utilizar diferentes conexiones UART. */
  if (huart->Instance == USART1){

    newData = 1; // Activa bandera
```

```
    /* Recibe una cantidad de datos hasta que ocurre un evento
   IDLE, es decir,
     *cuando se dejan de recibir datos en un determinado periodo.
    Por lo que
     *no es necesario saber la cantidad de datos que se esperan
   recibir.*/
    HAL_UARTEx_ReceiveToIdle_DMA(&huart1, RxBuf, RxBuf_SIZE);

    /* Deshabilita interrupcion que sucede cuando la mitad de los
    datos han
     * sido transferidos.*/
    __HAL_DMA_DISABLE_IT(&hdma_usart1_rx, DMA_IT_HT);

    memcpy(RxData, &RxBuf[0], sizeof(RxBuf)); // Copia los datos
   recibidos.
    memset(RxBuf, '\0', sizeof(RxBuf)); // Reinicia el buffer
    parseData(); // Funcion para dividir datos recibidos
  }
}
/* USER CODE END 0 */
```

- Dentro del ciclo infinito se pueden llevar a cabo las operaciones tanto de transmisión como de recepción. En este caso se muestra el código en el que, una vez activada la bandera de recepción, se manda nuevamente a cero y envía un paquete al XBee.

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    // Recepcion via puerto UART
    if(newData == 1){
        newData = 0; //Reinicia bandera
        HAL_UART_Transmit(&huart1, packet, sizeof(packet), 1000);
    }
/* USER CODE END WHILE */
}
```

En el caso de módulo XBee es configurado en el modo de operación MicroPython REPL para facilitar el manejo de los paquetes de envío y recepción. El código desarrollado es el siguiente:

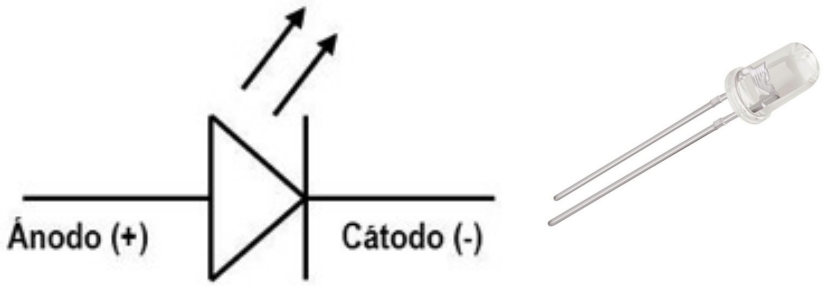
```
import xbee, time, sys

while True:
```

```
    p = xbee.receive()
    data = sys.stdin.read()
    if p:
        print(p['payload'].decode())
    else:
        time.sleep(1)
    if data:
        #Accion a realizar al recibir por UART
        xbee.transmit(xbee.ADDR_COORDINATOR, data)
```

III.4 Diodo Emisor de Luz (LED)

Un diodo emisor de luz (también conocido por la sigla LED, del inglés light-emitting diode) es una fuente de luz constituida por un material semiconductor dotado de dos terminales (como se puede observar en la figura 18). Se trata de un diodo de unión p-n, que emite luz cuando está activado. Si se aplica una tensión adecuada a los terminales, los electrones se reordenan con los huecos en la región de la unión p-n del dispositivo, liberando energía en forma de fotones. Este efecto se denomina electroluminiscencia, y el color de la luz generada (que depende de la energía de los fotones emitidos) viene determinado por el ancho de la banda prohibida del semiconductor.

El objetivo del diodo emisor de luz en el sistema desarrollado, es de representar las salidas luminosas que tiene un semáforo, por lo que se utilizan los característicos colores verde, ámbar y rojo. Es muy útil para realizar el prototipo, ya que se enciende al recibir señales altas por un pin.



(a) (b)

Figura 18: (**a**) Símbolo LED, (**b**) LED

III.4.1 Modos de funcionamiento del semáforo inteligente

En el microcontrolador se programan 4 modos de funcionamiento para el control de las luces del semáforo dependiendo la cadena de texto recibida de manera inalámbrica. Una vez recibidos los datos nuevos se llama a una función para dividir la cadena de texto por comas, en la cual el primer elemento corresponde al modo de funcionamiento.

Si el valor del modo de funcionamiento es igual a 0 o 1, corresponde a los ciclos normales del semáforo y la cadena de texto continuara con el tiempo que se mantendrán encendidas cada una de las luces, por lo que la cadena se sigue dividiendo y guardando los valores convertidos a milisegundos en un arreglo; se utilizan valores predefinidos para el tiempo de parpadeo de luz verde y el tiempo de transición de rojo a verde.

Si el valor es 2 se trata del estado de emergencia, por lo que habrá un control directo sobre el estado de las luces, la cadena de texto continua con el estado indicado para cada luz (1 para encendido y 0 para apagado), por lo que la cadena se sigue dividiendo y guardando los valores en un arreglo, los tiempos se mandan a 0 para pasar inmediatamente a la función.

El valor 3 corresponde al modo de funcionamiento nocturno y este es el único elemento de la cadena.

El código de la función descrita se muestra a continuación:

```
/* USER CODE BEGIN 4 */
void parseData(){
  /* Divide los datos recibidos separados por comas */
  char * strtokIndx; // Utilizado por strtok() como indice
  strcpy(tempChars, (char *)RxData); /* Crea una copia temporal para
   proteger los
  *datos originales */
  strtokIndx = strtok(tempChars,","); // Toma el primer valor
  lightTimes[0] = atoi(strtokIndx); // Este valor define el modo de
   funcionamiento

  if (lightTimes[0]==3){ //Modo nocturno
    //Espera a que termine el ciclo actual
  }
  else if (lightTimes[0]==2){ //Emergencia
    for(int i=1;i<=9;i++){
      strtokIndx = strtok(NULL, ","); // Continua donde termina el
   valor anterior
      emergencies[i] = atoi(strtokIndx);     // asigna el valor a
   emergencies[i]
      lightTimes[i] = 0; //Hace cero los tiempos para pasar
   inmediatamente
    }
    blinkTime = 0; //Tiempo total parpadeo luz verde en milisegundos
    blinkingTime = 0; //Tiempo entre parpadeos
    numBlink = 0; //Cantidad de parpadeos de luz verde
    allRed = 0; //Tiempo todos a rojo en milisegundos
  }
  else{ //Ciclos normales de semaforo
    for(int i=1;i<=9;i++){
      strtokIndx = strtok(NULL, ","); // Continua donde termina el
   valor anterior
      lightTimes[i] = atoi(strtokIndx);     // asigna el valor a
   lightTimes[i]
      lightTimes[i] = lightTimes[i]*1000; //De segundos a milisegundos
    }
    blinkTime = 2000; //Tiempo total parpadeo luz verde en
   milisegundos
    blinkingTime = 200; //Tiempo entre parpadeos
    numBlink = 5; //Cantidad de parpadeos de luz verde
    allRed = 1000; //Tiempo todos a rojo en milisegundos
  }
}
/* USER CODE END 4 */
```

Una vez dividida la cadena de texto dentro del ciclo infinito se llama a una función para mandar todas las luces a cero y así evitar que se activen múltiples señales al mismo tiempo, posteriormente se selecciona el modo de funcionamiento dependiendo el primer

valor del arreglo.

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    // Recepcion via puerto UART
    if(newData == 1){
        newData = 0;
        reset();
        if (lightTimes[0]==3){
          nightMode();
        }
        else if (lightTimes[0]==2){ //Emergencia
          emergency(emergencies);
        }
        else if (lightTimes[0]==0){ //Ciclo normal del semaforo
          interType1(lightTimes);
        }
        else if (lightTimes[0]==1){
          interType2(lightTimes);
        }
    }
/* USER CODE END WHILE */
}
```

A continuación se muestran cada uno de los modos de funcionamiento con el respectivo código de su función.

- **Intersección tipo 1:** El modo de funcionamiento para intersección de tipo 1 corresponde a las intersecciones en los que el permiso de pase se da a cada vía de manera individual, por lo que cada semáforo tiene sus propios tiempos.

```
void interType1 (int lightTime[9]) {
  HAL_GPIO_WritePin(redLightPorts[1],redLightPins[1], 1);
  HAL_GPIO_WritePin(redLightPorts[2],redLightPins[2], 1);
  HAL_GPIO_WritePin(redLightPorts[3],redLightPins[3], 1);
  HAL_GPIO_WritePin(greenLightPorts[0],greenLightPins[0], 1);
  int secuencia[] = {0,2,1,3};
  int actual;
  int anterior;
  while (newData == 0){
    for (int i=0;i<4;i++){
      get_time();
      if(band == 1){ //Revisa si se activa la alarma
        band = 0;
        HAL_UART_Transmit(&huart1, packet, sizeof(packet), 1000);
      }
```

```
      actual=secuencia[i];
      anterior=secuencia[(i+3)%4];
      HAL_GPIO_WritePin(amberLightPorts[anterior],amberLightPins[
   anterior], 0); //Apaga amarillo anterior
      HAL_GPIO_WritePin(redLightPorts[anterior],redLightPins[
   anterior], 1); //Enciende rojo anterior
      HAL_Delay(allRed);
      HAL_GPIO_WritePin(redLightPorts[actual],redLightPins[actual
   ], 0); //Apaga rojo actual
      HAL_GPIO_WritePin(greenLightPorts[actual],greenLightPins[
   actual], 1); //Enciende verde actual
      HAL_Delay(lightTime[((actual*2)+1)]-blinkTime); //Tiempo
   semaforo verde -2 segundos de parpadeo
      //Parpadeo luz verde.
      for (int k=0;k<(numBlink*2);k++){
        HAL_GPIO_TogglePin(greenLightPorts[actual],
   greenLightPins[actual]);
        HAL_Delay(blinkingTime);
      }
      HAL_GPIO_WritePin(greenLightPorts[actual],greenLightPins[
   actual], 0); //Apaga verde actual
      HAL_GPIO_WritePin(amberLightPorts[actual],amberLightPins[
   actual], 1); //Enciende amarillo actual
      HAL_Delay(lightTime[((actual*2)+2)]); //Tiempo semaforo
   amarillo
    }
  }
}
```

- **Intersección tipo 2:** El modo de funcionamiento para intersección de tipo 2 corresponde a las intersecciones en los que el permiso de pase se da a dos vías de manera simultánea al no entrar en conflicto, por lo que los semáforos de dichas vías tendrán los mismos tiempos y solo se tendrán las fases mostradas en la figura 19.

```
void interType2 (int lightTime[5]) {
  int tmp_aux;
  HAL_GPIO_WritePin(redLightPorts[1],redLightPins[1], 1);
  HAL_GPIO_WritePin(redLightPorts[3],redLightPins[3], 1);
  HAL_GPIO_WritePin(greenLightPorts[0],greenLightPins[0], 1);
  HAL_GPIO_WritePin(greenLightPorts[2],greenLightPins[2], 1);
  while (newData == 0){
    tmp_count=1; //Reinicia el contador al terminar cada ciclo
    for(int i=0;i<2;i++){
      tmp_aux = (i+1)%2;
      HAL_GPIO_WritePin(amberLightPorts[tmp_aux],amberLightPins[
   tmp_aux], 0);
```

```
      HAL_GPIO_WritePin(amberLightPorts[tmp_aux+2],amberLightPins
   [tmp_aux+2], 0);
      HAL_GPIO_WritePin(redLightPorts[tmp_aux],redLightPins[
   tmp_aux], 1);
      HAL_GPIO_WritePin(redLightPorts[tmp_aux+2],redLightPins[
   tmp_aux+2], 1);
      HAL_Delay(allRed);
      HAL_GPIO_WritePin(redLightPorts[i],redLightPins[i], 0);
      HAL_GPIO_WritePin(redLightPorts[i+2],redLightPins[i+2], 0);
      HAL_GPIO_WritePin(greenLightPorts[i],greenLightPins[i], 1);
      HAL_GPIO_WritePin(greenLightPorts[i+2],greenLightPins[i+2],
    1);
      HAL_Delay(lightTime[tmp_count]-blinkTime);
      tmp_count++;
      //Parpadeo luz verde.
      for (int j=0;j<(numBlink*2);j++){
        HAL_GPIO_TogglePin(greenLightPorts[i], greenLightPins[i])
   ;
        HAL_GPIO_TogglePin(greenLightPorts[i+2], greenLightPins[i
   +2]);
        HAL_Delay(blinkingTime);
      }
      HAL_GPIO_WritePin(greenLightPorts[i],greenLightPins[i], 0);
      HAL_GPIO_WritePin(greenLightPorts[i+2],greenLightPins[i+2],
    0);
      HAL_GPIO_WritePin(amberLightPorts[i],amberLightPins[i], 1);
      HAL_GPIO_WritePin(amberLightPorts[i+2],amberLightPins[i+2],
    1);
      HAL_Delay(lightTime[tmp_count]);
      tmp_count++;
    }
  }
}
```

- **Modo de emergencia:** El modo de emergencia, al ocurrir inmediatamente después de ser recibido por el controlador, inicia con una etapa de aviso para los conductores con una duración de 5 segundos, posteriormente enciende las luces según la indicación recibida.

```
void emergency (int emergencies[9]) {
  int j=1;
  //Etapa aviso estado de emergencia (5 segundos de parpadeo)
  for (int i=0;i<24;i++){ //i DEBE terminar en numero par para
   que terminen apagados
    HAL_GPIO_TogglePin(amberLightPorts[0],amberLightPins[0]);
    HAL_GPIO_TogglePin(redLightPorts[1],redLightPins[1]);
    HAL_GPIO_TogglePin(amberLightPorts[2],amberLightPins[2]);
    HAL_GPIO_TogglePin(redLightPorts[3],redLightPins[3]);
```

```
    HAL_Delay(200);
  }
  // Instrucciones emergencia
  for (int i=0;i<4;i++){
    HAL_GPIO_WritePin(greenLightPorts[i],greenLightPins[i],
   emergencies[j]);
    j++;
    HAL_GPIO_WritePin(redLightPorts[i],redLightPins[i],
   emergencies[j]);
    j++;
  }
}
```

- **Modo Nocturno:** Este modo de funcionamiento hace parpadear las luces amarillas de los semáforos norte y sur, y las rojas de los semáforos este y oeste.

```
void nightMode () {
  while (newData == 0){
    HAL_GPIO_TogglePin(amberLightPorts[0],amberLightPins[0]);
    HAL_GPIO_TogglePin(redLightPorts[1],redLightPins[1]);
    HAL_GPIO_TogglePin(amberLightPorts[2],amberLightPins[2]);
    HAL_GPIO_TogglePin(redLightPorts[3],redLightPins[3]);
    HAL_Delay(blinkingTime);
  }
}
```

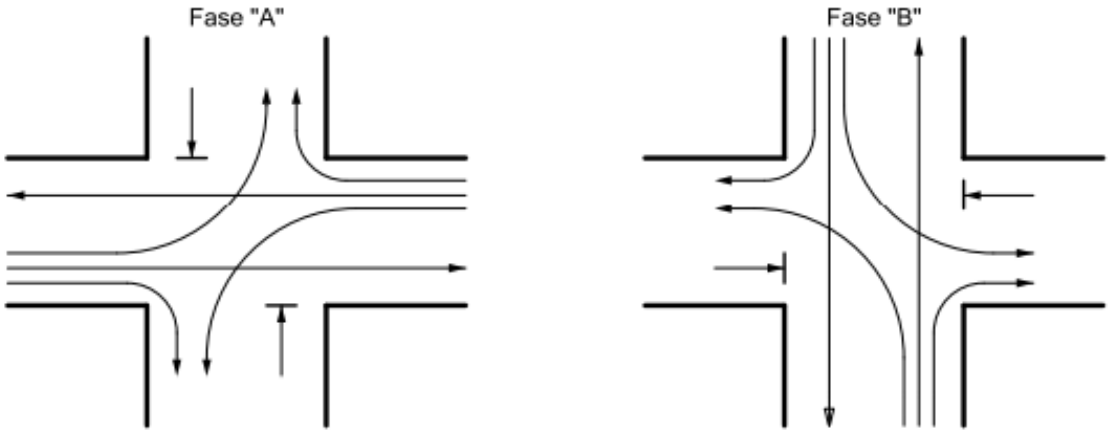


Figura 19: Fases intersección tipo 2.

III.5 Resumen

En este capítulo se describió el sistema para el control remoto de semáforos inteligentes desarrollado en el presente trabajo de tesis, se describen cada uno de los componentes que conforman a la central y el controlador, haciendo énfasis en sus principales características y la manera en que son configurados. Asimismo se muestra la programación realizada para cumplir con las funciones deseadas de comunicación inalámbrica y control de las señales.

Capítulo IV

Contador ultrasónico de vehículos e interfaz gráfica

IV.1 Contador ultrasónico de vehículos

Como se menciona en el capítulo II uno de los principales componentes de los ITS son los sistemas de adquisición de información conformados por redes de sensores. Al contabilizar los vehículos que cruzan por las vías cada cierto tiempo, se puede obtener el comportamiento del tráfico vehicular según los distintos horarios. De esta manera, se puede utilizar esta información para tener un control dinámico de los tiempos con los que se programan los dispositivos para manejo de tráfico, ya sea por la toma de decisiones de un operador o de manera automática utilizando un algoritmo como el que se muestra en el capítulo V.

El sistema para contabilizar vehículos propuesto en este trabajo utiliza la tecnología ultrasónica para llevar a cabo el conteo. Los sensores ultrasónicos son comúnmente usados para medir distancias al generar pulsos de ultrasonidos que rebotan contra el objeto de interés como se muestra en la figura 20.

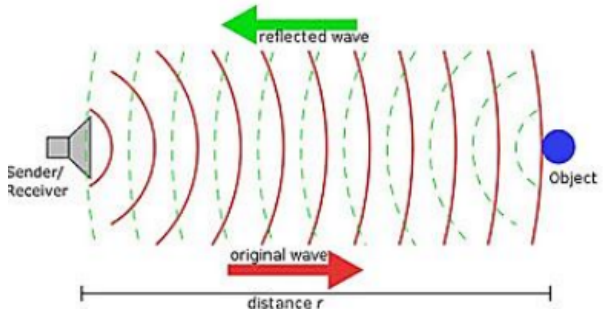


Figura 20: Medición de distancia utilizando sensor ultrasónico (Prasetyo *et al.*, 2018).

Se mide el tiempo que la onda tarda en regresar al sensor y como se sabe que la distancia es igual a la velocidad por el tiempo, donde la velocidad ultrasónica está dada por 1 y el tiempo de viaje se debe dividir a la mitad dado que este representa el tiempo tanto de ida como de regreso de la onda. Lo que nos da la fórmula 2 para el cálculo de la distancia basado en la medición del sensor.

$$Vel.ultrasónica = 331.5m/s + (0.61 * temperatura) \tag{1}$$

$$d = Vel.ultrasónica * (t/2) \tag{2}$$

El uso de este tipo de sensores no es nuevo en la industria del transporte, se ha utilizado en estacionamientos inteligentes (véase figura 21) y es muy común encontrar este tipo de sensores en la parte trasera de los autos modernos para evitar colisiones (véase figura 22).

Figura 21: Sensores ultrasónicos en estacionamientos inteligentes.

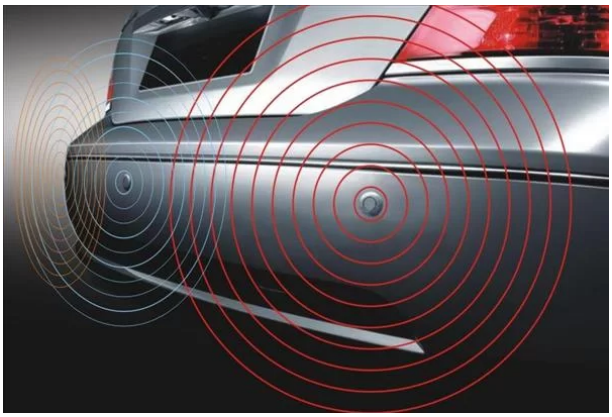


Figura 22: Sensores ultrasónicos en autos modernos.

El sistema propuesto consiste en colocar los sensores por encima de la vía como se muestra en la figura 23, siguiendo la premisa de (Appiah *et al.*, 2020), que consiste en que el sensor estará continuamente midiendo una distancia fija (la que se tiene entre él y el pavimento) y, al circular un vehículo por debajo del sensor, la distancia que este mida será menor. De esta manera, se tiene una salida similar a la ejemplificada en la gráfica de la figura 24 y la tarea se reduce a contabilizar los flancos.

Figura 23: Funcionamiento del contador ultrasónico de vehículos.

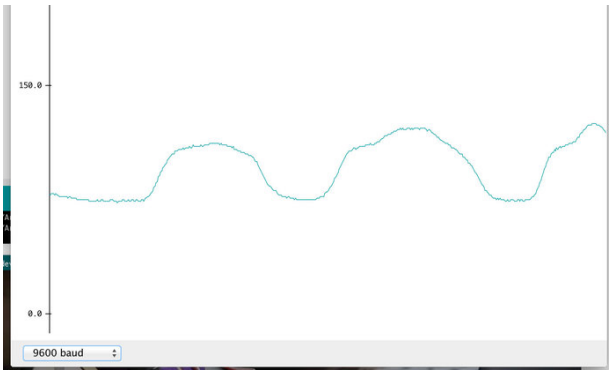


Figura 24: Ejemplificación de las mediciones.

Se espera que el sistema conforme una red inalámbrica de sensores, al implementar contadores de vehículos en cada vía de las distintas intersecciones. Para ello los sensores ultrasónicos, igual al número de carriles que conforma la vía, se conectan a un módulo de comunicación inalámbrica.

IV.1.1 Sensor ultrasónico GY-US42 V2

Para el desarrollo del contador se utiliza el sensor ultrasónico GY-US42 V2. Este sensor se puede comunicar con el microcontrolador utilizando los pines Echo-Trig al igual que el ampliamente utilizado HC-SR04, además, puede comunicarse mediante I2C, lo que representa una ventaja al momento de conectar múltiples sensores al microcontrolador, por lo cual se eligió este tipo de comunicación para el contador. En la figura 25, se muestran distintas vistas del sensor GY-US42 V2 y sus principales características técnicas se muestran en la tabla III.

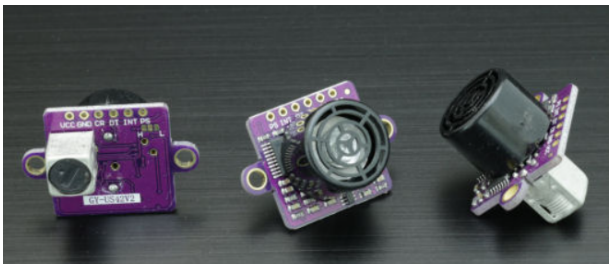


Figura 25: Sensor ultrasónico GY-US42V2.

Tabla III: Características técnicas del sensor ultrasónico GY-US42 V2.

Parámetro	**Valor**
Alcance mínimo	20 cm
Alcance máximo	720 cm
Frecuencia de trabajo	15 Hz (rango completo)
Voltaje de operación	3-5 V
Consumo de corriente	9 mA

Además de las opciones de comunicación con un microcontrolador, otro de los motivos de la elección de este sensor es el alcance máximo de aproximadamente 7 metros a comparación con otros sensores de su tipo que tienen un alcance promedio de 4 metros el cual, como se muestra en la figura 26, puede no ser suficiente para el funcionamiento requerido.

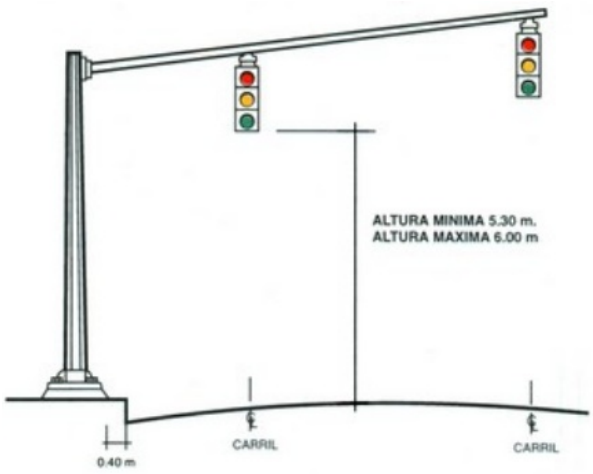


Figura 26: Medidas semáforo.

IV.1.2 Comunicación inalámbrica y algoritmo del contador

La comunicación inalámbrica del contador con el controlador de semáforos y la central se lleva a cabo utilizando un módulo XBee 3 visto en la sección III.3. La conexión I2C de múltiples sensores se muestra en el diagrama de la figura 27 y se describe a continuación: el pin CR del sensor se conecta al pin 19 del XBee (I2C SCL) y el pin DT

del sensor se conecta al pin 7 del XBee (I2C SDA), las resistencias pull-up mostradas en el diagrama no son necesarias. La alimentación del módulo XBee es con 3.3 V y la de los sensores con 5 V para maximizar el alcance.

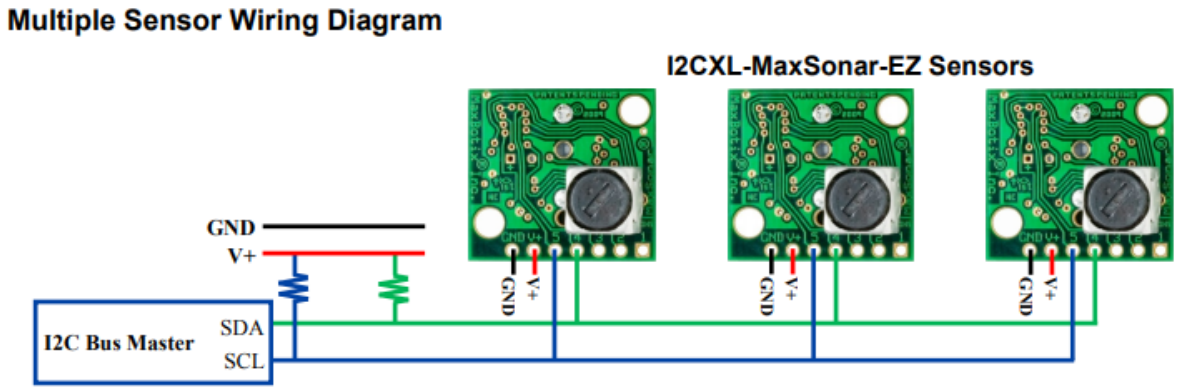


Figura 27: Diagrama de conexión I2C de múltiples sensores.

Dado que la dirección por defecto de los sensores es la misma, se deben cambiar de manera individual previo a conectarlos juntos, siguiendo los comandos de la hoja de datos mostrados en la figura 28.

Change the sensor address	**1.** Initiate a write at the sensor address	224 (default)	1110 0000	The sensor will only accept even address values. If an odd numbered address is sent the sensor will be set to the next lowest even number. If the sensor is told to change to one of the invalid addresses below the sensor will ignore this command and stay at its current address. **Invalid Address Values:** 0, 80, 164, 170
	2a. Write three bytes to the sensor starting with the addr_unlock_1 command	**170**	1010 1010	
	2b. Write the addr_unlock_2 command	**165**	1010 0101	
	2c. Write the new sensor address	*(User Value)*	#### ###0	

Figura 28: Comandos para cambio de dirección.

El diagrama de la figura 29 muestra el algoritmo para el conteo vehicular.

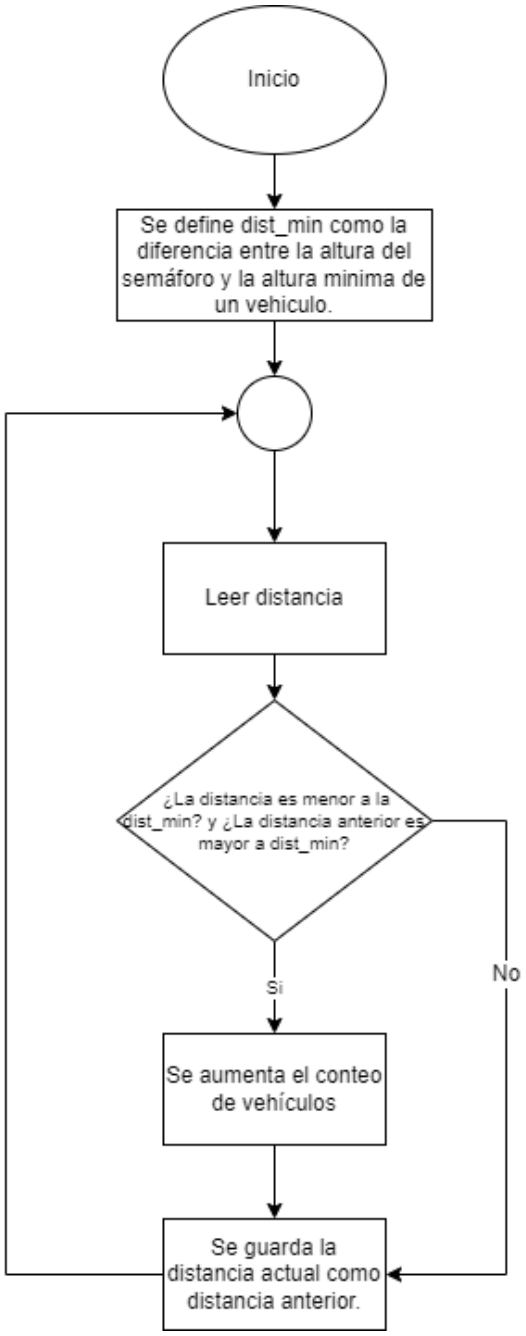


Figura 29: Diagrama de algoritmo para conteo vehicular.

El módulo XBee se configura en el modo de operación MicroPython REPL para el manejo de los sensores y mensajes. El código cargado se muestra a continuación:

- Se importan las librerías y variables necesarias.

```
from machine import I2C
from time import sleep
import xbee

rng_cmd = b'\x51' #Comando para medir distancia
lane_dist = list()
# Variables para conteo vehicular
tl_height = 530 # Altura del semaforo
min_veh_height = 100 # altura minima de vehiculo de 1 metro
dist_min = tl_height - min_veh_height # Distancia a partir de la
    cual se considera
#la presencia de un vehiculo
n_veh = list() #Numero de vehiculos por carril
old_dist_lane = list() #Utilizada para comparar la distancia
    anterior
```

- Se inicializa la comunicación I2C a 100 kHz, se buscan los dispositivos y se imprimen las direcciones de 7 bits (Es importante haber configurado previamente diferentes direcciones para cada sensor).

```
i2c = I2C(1, freq=100000) # create I2C peripheral at frequency of
    100kHz
devices = i2c.scan()
print(devices)
```

- Se rellenan las listas con ceros igual al numero de sensores.

```
lane_dist = [0] * len(devices)
n_veh = [0] * len(devices)
old_dist_lane = [0] * len(devices)
```

- En el ciclo infinito se revisa si se recibe un mensaje, en caso de que esto ocurra se envían las lecturas al coordinador y se reinicia el conteo.

```
while True:
    p = xbee.receive()
    if p:
        # Enviar y reiniciar conteo de vehiculos
        message = ""
        i=0
        for device in devices:
        message = message + str(n_veh[i])+","
            n_veh[i]=0
            i+=1
        xbee.transmit(xbee.ADDR_COORDINATOR, message)
        print(message)
```

- De igual manera en el ciclo infinito se leen las distancias de cada sensor, la distancia en centímetros está dada por el primer byte multiplicado por 256 más el segundo byte. En caso de que se detecte un flanco de bajada entre lecturas se cuenta un vehículo.

```
i = 0
for device in devices:
    #Lecutra de distancias
    i2c.writeto(device, rng_cmd) # write range command
    sleep(.1)
    val = i2c.readfrom(device, 2) # read 2 bytes
    lane_dist[i] = (val[0]*256) + val[1]
    print(lane_dist[i])
    #Conteo de vehiculos
    if (lane_dist[i])<=dist_min and (old_dist_lane[i])>dist_min:
   #Flanco de bajada
        n_veh[i]+=1
    old_dist_lane[i] = lane_dist[i]
```

IV.2 Desarrollo interfaz gráfica de usuario

Se desarrolla una interfaz gráfica utilizando tkinter en Python para la programación y control de manera remota de las intersecciones (véase figura 30), con el fin de permitir a un usuario de tránsito llevar a cabo estas acciones desde una central, en la que una computadora se conecte a un módulo XBee y este comience una red con los otros módulos, para enviar las indicaciones a cada intersección.

Figura 30: Interfaz gráfica.

En la sección de intersección se tiene un cuadro combinado desplegable mostrado en la figura 31, este elemento carga una base de datos hecha en un archivo de valores separados por comas (CSV) el cual contiene el nombre de las intersecciones, la dirección con la que se identificara a cada una al comunicarse de manera inalámbrica y el tipo (obsérvese figura 32); el tipo 0 representa una intersección en la que cada semáforo tiene sus propios tiempos y el tipo 1 representa intersecciones en las que los semáforos de norte-sur y este-oeste funcionan de manera simultánea, por lo que sus tiempos son iguales. El tipo de intersección se envía como modo de funcionamiento al controlador.

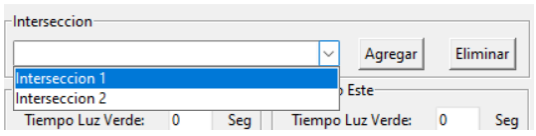


Figura 31: Cuadro combinado desplegable.

	A	B	C	D
1	Interseccion	Direccion	Tipo	
2	Interseccion 1	REMOTE	1	
3	Interseccion 2	Sensor	0	
4				

Figura 32: Archivo CSV con información de las intersecciones.

En la misma sección de intersección se pueden agregar nuevas intersecciones a la base de datos, al pulsar el botón de agregar se abre una nueva ventana (mostrada en la figura 33) en la que se deben ingresar el nombre, la dirección (Node ID del XBee) y el tipo de la nueva intersección. Asimismo, se pueden eliminar intersecciones existentes, pulsando el botón eliminar, se abre una ventana en la que se debe seleccionar la intersección y pulsar el botón eliminar (véase figura 34).

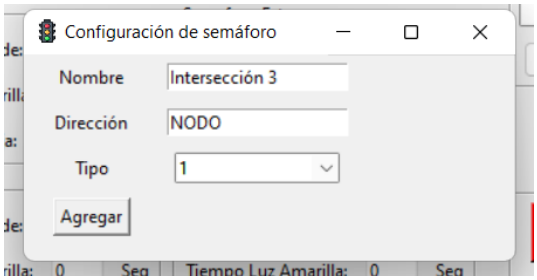


Figura 33: Ventana para añadir intersección.

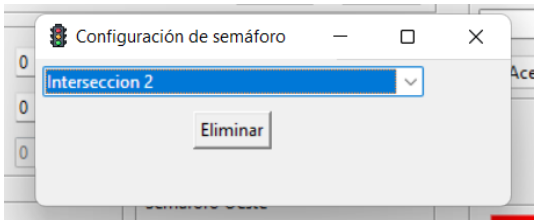


Figura 34: Ventana para eliminar intersección.

En el programa se capturan los tiempos de luz verde y amarilla para cada semáforo, en la sección control se tiene un cuadro combinado desplegable en el que se cargan los puertos serie encontrados, al seleccionar el puerto con el módulo XBee y pulsar el botón aceptar se envía una cadena de texto con el modo de funcionamiento y los tiempos vía puerto serie para, de esta manera, comunicar el programa con el módulo XBee y este, a su vez, enviar la trama de datos a la dirección especificada. La cadena de texto se muestra en la figura 35, el primer elemento corresponde al modo de funcionamiento y los siguientes a los tiempos verdes y ámbar de cada semáforo.

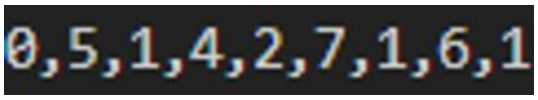


Figura 35: Cadena de texto enviada.

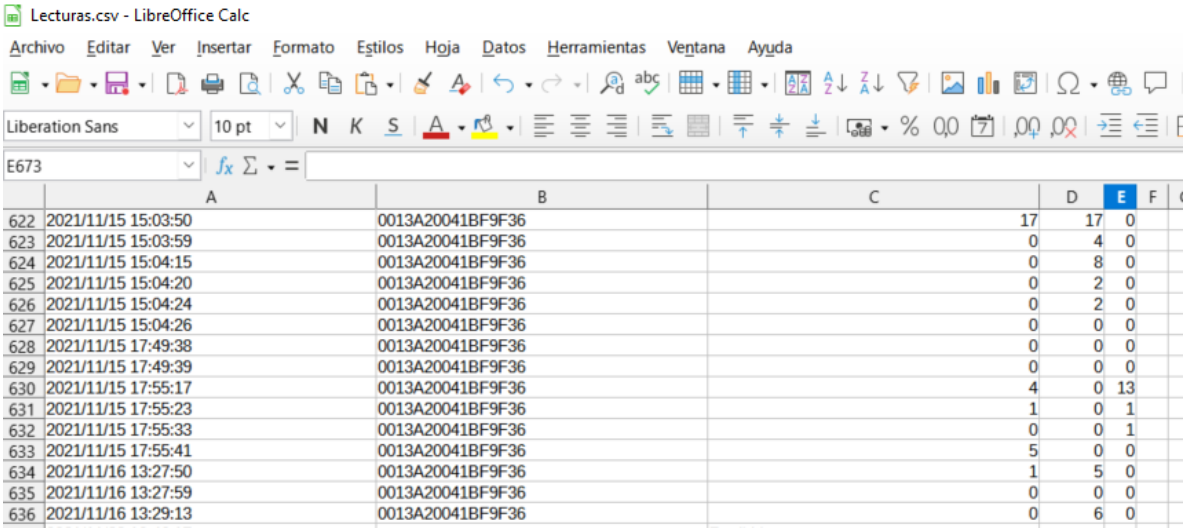
Cada vez que el usuario modifica los cuadros de texto se valida que no se dejen vacíos y que el valor ingresado sea numérico, de igual manera, al pulsar el botón aceptar para enviar la cadena de texto, se valida que los valores ingresados sean mayores a cero, que el usuario haya seleccionado la intersección de destino y el puerto serie, de lo contrario arrojará una ventana de error.

El modo de operación nocturno hace que los semáforos norte y sur parpadeen en amarillo, y los semáforos este y oeste parpadeen en rojo, al pulsar el botón se envía el modo de funcionamiento 3 a todas las intersecciones. El botón de emergencia abre una nueva ventana en la que se muestran las intersecciones en forma de lista, nos permite seleccionar la intersección que queramos controlar y la acción a realizar como se muestra en la figura 36, al pulsar el botón enviar se manda una cadena a las intersecciones seleccionadas con el modo de funcionamiento 2 y el estado indicado para cada luz (1 para encendido y 0 para apagado). Se tiene una restricción con respecto a las luces verdes que se permiten encender al mismo tiempo con el fin de evitar accidentes, así como un periodo de 5 segundos para advertir a los conductores.

Figura 36: Ventana modo de emergencia.

En el momento que se selecciona un puerto del cuadro combinado desplegable se

crea un hilo en el que se revisa si se recibe nueva información por el puerto seleccionado y se genera un archivo CSV con la fecha y hora, la dirección MAC del XBee emisor y los datos recibidos como se muestra en la figura 37.


Lecturas.csv - LibreOffice Calc

Archivo Editar Ver Insertar Formato Estilos Hoja Datos Herramientas Ventana Ayuda

Liberation Sans 10 pt

E673

	A	B	C	D	E	F
622	2021/11/15 15:03:50	0013A20041BF9F36	17	17	0	
623	2021/11/15 15:03:59	0013A20041BF9F36	0	4	0	
624	2021/11/15 15:04:15	0013A20041BF9F36	0	8	0	
625	2021/11/15 15:04:20	0013A20041BF9F36	0	2	0	
626	2021/11/15 15:04:24	0013A20041BF9F36	0	2	0	
627	2021/11/15 15:04:26	0013A20041BF9F36	0	0	0	
628	2021/11/15 17:49:38	0013A20041BF9F36	0	0	0	
629	2021/11/15 17:49:39	0013A20041BF9F36	0	0	0	
630	2021/11/15 17:55:17	0013A20041BF9F36	4	0	13	
631	2021/11/15 17:55:23	0013A20041BF9F36	1	0	1	
632	2021/11/15 17:55:33	0013A20041BF9F36	0	0	1	
633	2021/11/15 17:55:41	0013A20041BF9F36	5	0	0	
634	2021/11/16 13:27:50	0013A20041BF9F36	1	5	0	
635	2021/11/16 13:27:59	0013A20041BF9F36	0	0	0	
636	2021/11/16 13:29:13	0013A20041BF9F36	0	6	0	

Figura 37: Archivo CSV generado.

IV.3 Resumen

En este capítulo se describió el contador ultrasónico de vehículos propuesto, se presenta el funcionamiento y la configuración de los sensores, el algoritmo para el conteo de vehículos y la comunicación inalámbrica de los sensores. Se realza la importancia de los módulos XBee 3, debido a que además de permitir llevar a cabo la comunicación inalámbrica, son los que controlan los sensores y ejecutan el algoritmo de conteo, permitiendo así, prescindir de microcontroladores en cada uno de los nodos de la red inalámbrica de sensores. De igual manera, se muestra el diseño y funcionamiento de la interfaz gráfica de usuario desarrollada con la herramienta tkinter de Python para la programación de los tiempos, el control de las intersecciones y la generación de una base de datos tras la recepción de las lecturas de los sensores.

Capítulo V

Algoritmos Machine Learning para predicción del flujo vehicular

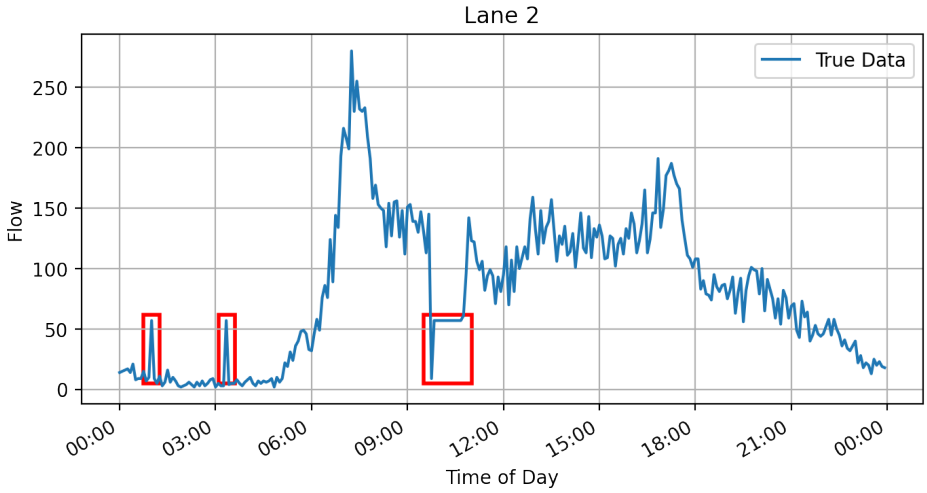
V.1 Base de datos

En este trabajo, se utiliza un conjunto de datos de predicción del tráfico del Centro de Investigación de Huawei en Múnich, que es un conjunto de datos público para la predicción del tráfico derivado de una variedad de sensores de tráfico, por ejemplo, bucles de inducción (Axenie y Bortoli, 2020), es importante señalar que, en la actualidad, hay pocos conjuntos de datos públicos (Hou *et al.*, 2021). Los datos pueden usarse para prever los patrones de tráfico y modificar los parámetros de control de los semáforos. La base de datos contiene datos registrados de seis cruces de la zona urbana durante 56 días, en forma de series temporales de flujo, que representan el número de vehículos que pasan cada cinco minutos durante todo un día, lo que se recomienda para las predicciones a corto plazo (Chen *et al.*, 2012). Para este trabajo, se utilizan cuatro de las seis intersecciones a manera de simular cuatro carriles de una intersección.

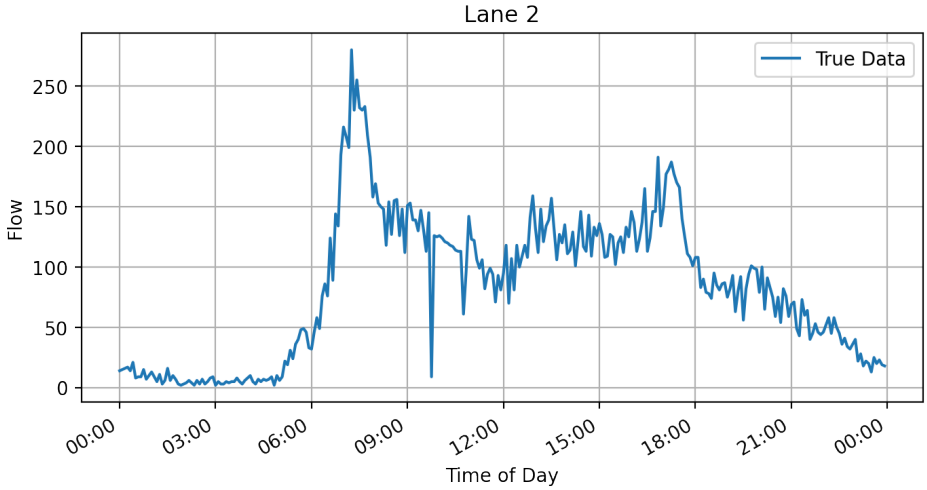
V.2 Preprocesamiento de los datos

Es habitual encontrar valores faltantes en las bases de datos representados por ceros, probablemente debido a fallos en los sensores. En el artículo de (Fu *et al.*, 2017) se reemplazan los datos que faltan utilizando el valor medio histórico, mientras que (Lawe y Wang, 2016) informan de la sustitución de estos valores por la media de toda la columna que contiene el valor que falta. Aunque estos datos sustituidos no son valores

realistas, los investigadores han determinado que son mejores a la ausencia de datos. Sin embargo, al realizar las predicciones, este procedimiento genera picos en los valores reales aumentando el error, ya que, en algunas ocasiones, la tendencia muestra que los valores de cero eran reales. Estos picos están resaltados en rectángulos rojos en la Figura 38a; por ello se aplica una media móvil utilizando las 12 lecturas anteriores. La figura 38b muestra cómo se evitan los cambios bruscos causados por la media general. Una vez aplicada la medía móvil, la base de datos se divide en un 75% de los datos (42 días) para el entrenamiento y un 25% (14 días) para las pruebas.



(a)



(b)

Figura 38: Diferencias entre la sustitución de ceros aplicando una media general y una media móvil. (**a**) Media General, y (**b**) Media Móvil.

Posteriormente, los datos se escalan en un rango de 0 a 1, siguiendo la distribución normal estándar empleando MinMaxScaler de la librería scikit-learn (Pedregosa *et al.*, 2011). Para este experimento, se usa el flujo de tráfico de la hora anterior, que es una secuencia de tiempo de 12 puntos de datos, para predecir el flujo de tráfico que viene en los próximos cinco minutos. Para ello, se crean listas agrupadas en 13 lecturas; estas listas se utilizan para el entrenamiento y la prueba. Las listas generadas se convierten

en matrices y se modifica la secuencia de entrenamiento. Una vez hecho esto, la última columna de los arreglos se toma como la salida 'Y', y el resto de columnas como las entradas 'X'.

V.3 Redes Neuronales Recurrentes

V.3.1 Diseño

Se diseñan dos redes neuronales recurrentes: GRU y LSTM. Se utiliza la librería Keras (Chollet *et al.*, 2015) para crear los modelos. La arquitectura es la misma para ambas, tal y como se muestra en la Figura 39 y se explica a continuación: La capa de entrada con forma igual al número de pasos de tiempo por el número de carriles. A continuación, dos capas recurrentes con 64 neuronas, 20% de dropout, y finalmente, una capa de salida con neuronas iguales al número de carriles y función de activación sigmoidea.

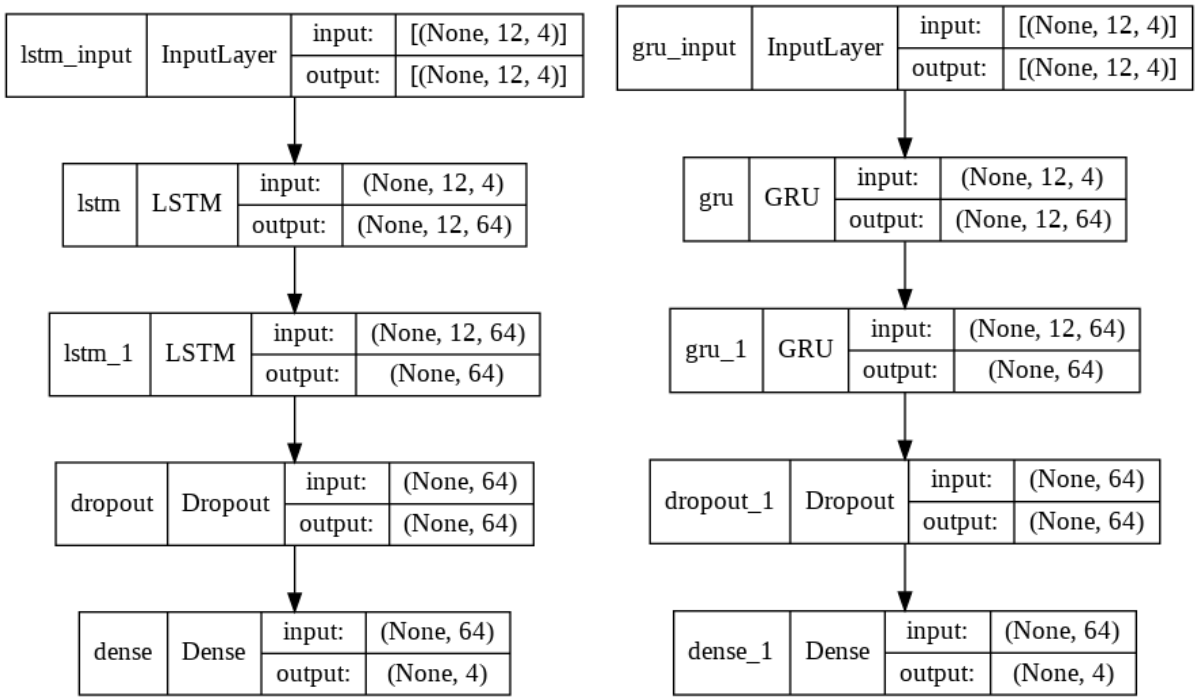
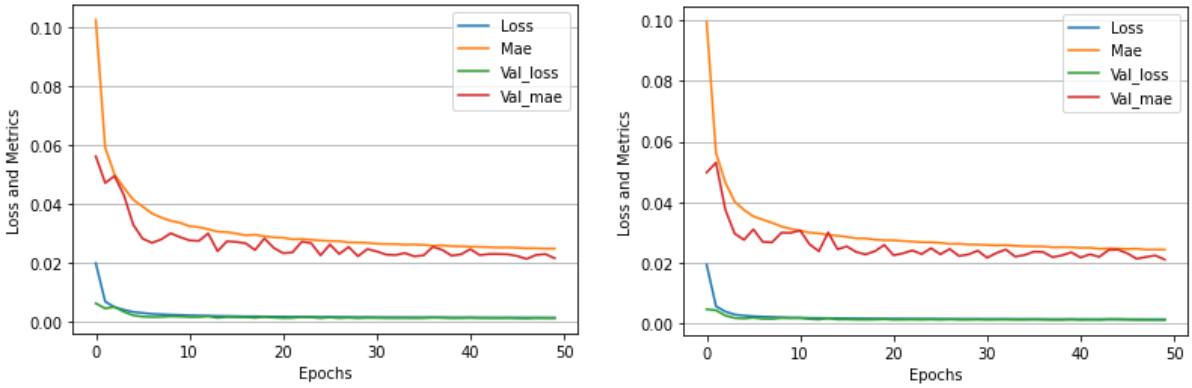


Figura 39: Arquitectura de las redes neuronales recurrentes. (**a**) LSTM-NN, (**b**) GRU-NN.

V.3.2 Entrenamiento

En la compilación del modelo, se usa el error cuadrático medio (MSE) como función de pérdida, el optimizador es RMSprop de la librería Keras (Chollet *et al.*, 2015) con parámetros por defecto y el error medio absoluto (MAE) es la función métrica. Para el entrenamiento, se utiliza un tamaño de lote de 128 y 50 épocas, el cinco por ciento de los datos de entrenamiento se utiliza para la validación. Los experimentos se realizan en Google Colaboratory (Bisong, 2019) junto con Weights & Biases (Biewald, 2020) para su seguimiento. La figura 40 muestra el rendimiento de entrenamiento de las dos arquitecturas, donde se pueden observar las métricas de pérdida y evaluación en el entrenamiento y la validación.



(a) (b)

Figura 40: Rendimiento del entrenamiento de ambas redes neuronales. (**a**) LSTM NN, (**b**) GRU NN.

V.4 Modelos de regresión

Se utilizan cinco modelos de regresión de la biblioteca scikit-learn (Pedregosa *et al.*, 2011) en Python: Regresión lineal, Regresor Potenciación del Gradiente, Regresor Perceptrón Multicapa, Regresor Descenso de Gradiente Estocástico y Regresor de Bosques Aleatorios, todos ellos con parámetros por defecto y un estado aleatorio igual a cero

para la reproducibilidad del experimento. Se cambia la forma del arreglo de 'X' tanto para el entrenamiento como para la prueba, ya que estos modelos requieren una matriz 2D en lugar de la 3D empleada en las RNN; después, los modelos se alimentan con la división de entrenamiento.

V.4.1 Regresor Perceptrón Multicapa

El perceptrón multicapa (MLP, por sus siglas en inglés) es una de las redes neuronales artificiales (ANN, por sus siglas en inglés) más populares. Aplica un procedimiento de entrenamiento supervisado utilizando ejemplos de datos con salidas conocidas. Este procedimiento genera un modelo de función no lineal que permite predecir los datos de salida a partir de unos datos de entrada dados (Taud y Mas, 2018).

El MLP se compone de una capa de entrada, una o más capas ocultas y una capa final de salida, cada capa, excepto la de salida, incluye una neurona de sesgo (bias), la arquitectura se muestra en la figura 41. El algoritmo de retro-propagación es empleado para su entrenamiento.

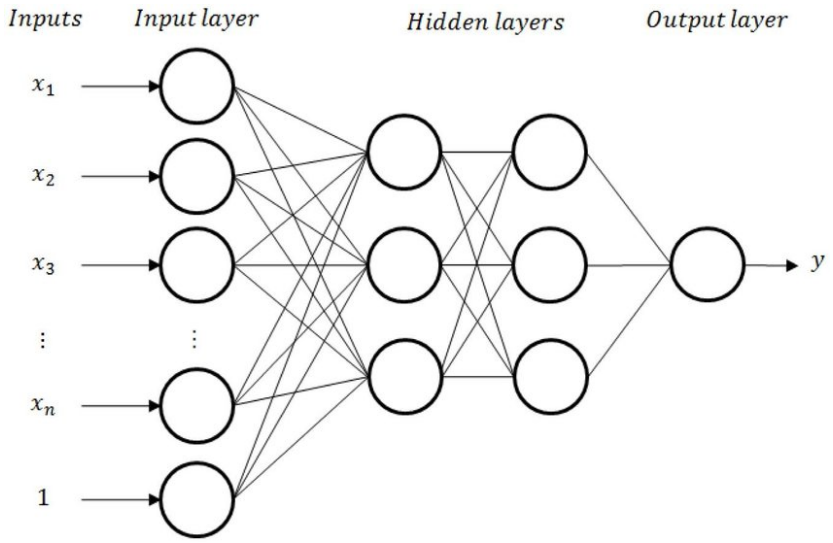


Figura 41: Arquitectura de Perceptrón Multicapa (Guerillot y Bruyelle, 2017)

Particularmente al crear un MLP para regresión, no es necesario usar una función

de activación en la capa de salida, lo que también puede considerarse como el uso de la función de identidad como función de activación. Por lo tanto, utiliza el error cuadrado como función de pérdida, y la salida es un conjunto de valores continuos.

V.4.2 Regresión Lineal

La regresión lineal intenta modelar la relación entre dos (o más) variables ajustando una línea recta a los datos (véase figura 42). Partiendo de la ecuación de una relación lineal: $y = m(x) + b$, en el caso del ML, m se representa como el peso y b como el sesgo o bias. Esta relación se denomina regresión lineal univariante porque solo hay una variable independiente. En muchos casos, los modelos no podrán ser predichos por una única variable independiente. En estos casos, habrá múltiples variables independientes que influyan en la variable dependiente. Esto a menudo puede modelarse como se muestra a continuación:

$$y = m_1x_1 + b_1 + m_2x_2 + b_2 + ... \tag{3}$$

Donde el peso y el sesgo de cada variable independiente influyen en la variable dependiente resultante.

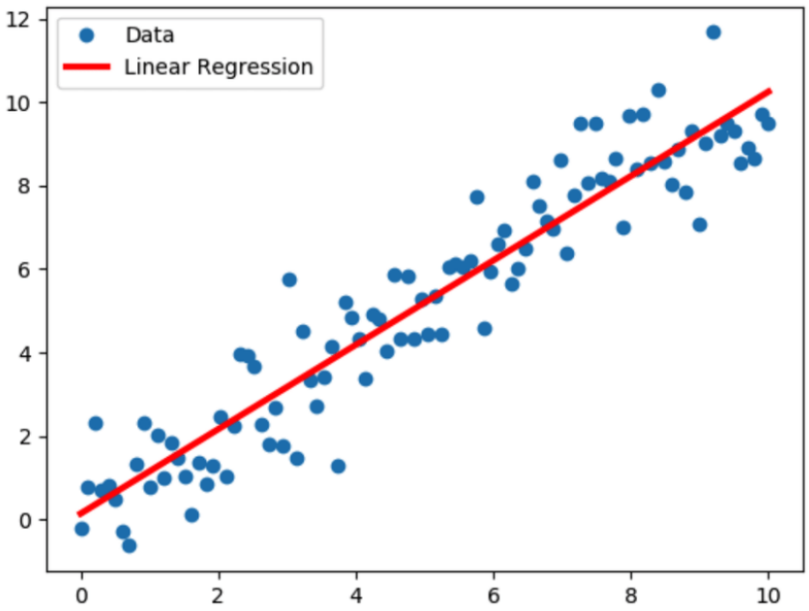


Figura 42: Regresión lineal (Tran, 2019)

V.4.3 Métodos de ensamble

Todos los modelos de aprendizaje estadístico y ML sufren el problema de equilibrio entre bias y varianza. El término bias o sesgo se refiere a cuánto se alejan en promedio las predicciones de un modelo respecto a los valores reales, es decir, refleja que tan capaz es el modelo de aprender la relación real que existe entre los predictores y la variable respuesta, mientras que el término varianza hace referencia a cuánto cambia el modelo dependiendo de los datos utilizados en su entrenamiento (Rodrigo, 2020). Los métodos de ensamble combinan múltiples modelos en uno nuevo con el objetivo de lograr un equilibro entre bias y varianza, consiguiendo así mejores predicciones que cualquiera de los modelos originales de manera individual. A continuación se definen dos de los tipos de ensamble más utilizados según (Rodrigo, 2020):

- **Bagging:** se ajustan múltiples modelos, cada uno con un subconjunto distinto de los datos de entrenamiento. Para predecir, todos los modelos que forman el agregado participan aportando su predicción. Como valor final, se toma la media

de todas las predicciones (variables continuas) o la clase más frecuente (variables categóricas). Los modelos de Bosques Aleatorios están dentro de esta categoría.

- **Boosting:** se ajustan de manera secuencial múltiples modelos sencillos, llamados aprendices débiles (weak learners), de forma que cada modelo aprende de los errores del anterior. En el caso de Gradient Boosting Trees, los aprendices débiles se consiguen utilizando árboles con una o pocas ramificaciones. El valor final se obtiene de la misma manera que en el caso de bagging, la media o la clase más frecuente según sea el caso.

V.4.4 Regresor Potenciación de Gradiente

Un modelo Potenciación de Gradiente está formado por un conjunto de árboles de decisión individuales, entrenados secuencialmente, de forma que cada nuevo árbol trata de mejorar los errores de los árboles anteriores. La predicción de una nueva observación se obtiene agregando las predicciones de todos los árboles individuales que forman el modelo (Rodrigo, 2020).

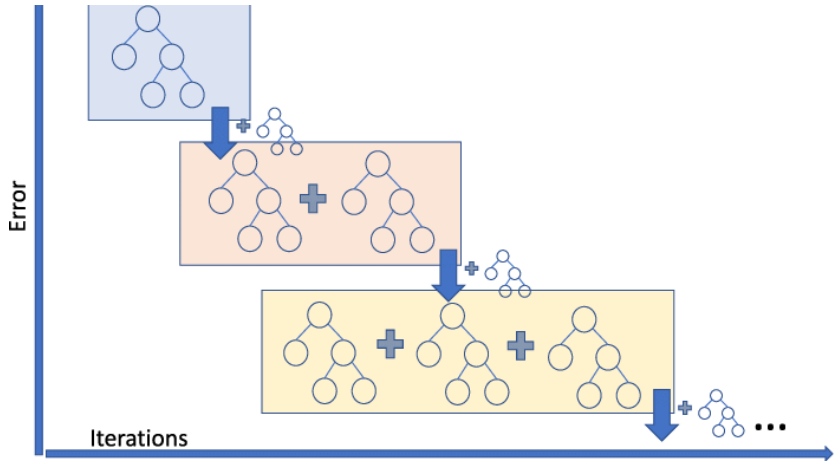


Figura 43: Representación esquemática de Regresor Potenciación de Gradiente (Baturynska y Martinsen, 2021)

V.4.5 Regresor Gradiente Estocástico

Tiempo después de la publicación de algoritmo de Potenciación de Gradiente, se le incorporó una de las propiedades de bagging, el muestreo aleatorio de observaciones de entrenamiento. En concreto, en cada iteración del algoritmo, el aprendiz débil se ajusta empleando únicamente una fracción del conjunto de entrenamiento, extraída de manera aleatoria y sin reemplazo. Al resultado de esta modificación se le conoce como Potenciación de Gradiente Estocástico (Rodrigo, 2020).

V.4.6 Regresor Bosques Aleatorios

Un bosque aleatorio es una colección de árboles de predicción $h(x;\theta_k), k = 1, ..., K$ donde x representa el vector de entrada observado de longitud p con un vector aleatorio asociado X y θ_k son vectores aleatorios independientes e idénticamente distribuidos. Para la regresión, la predicción del bosque aleatorio es la media no ponderada de la colección (Segal, 2004). En la figura 44 se muestra un diagrama con la estructura del modelo.

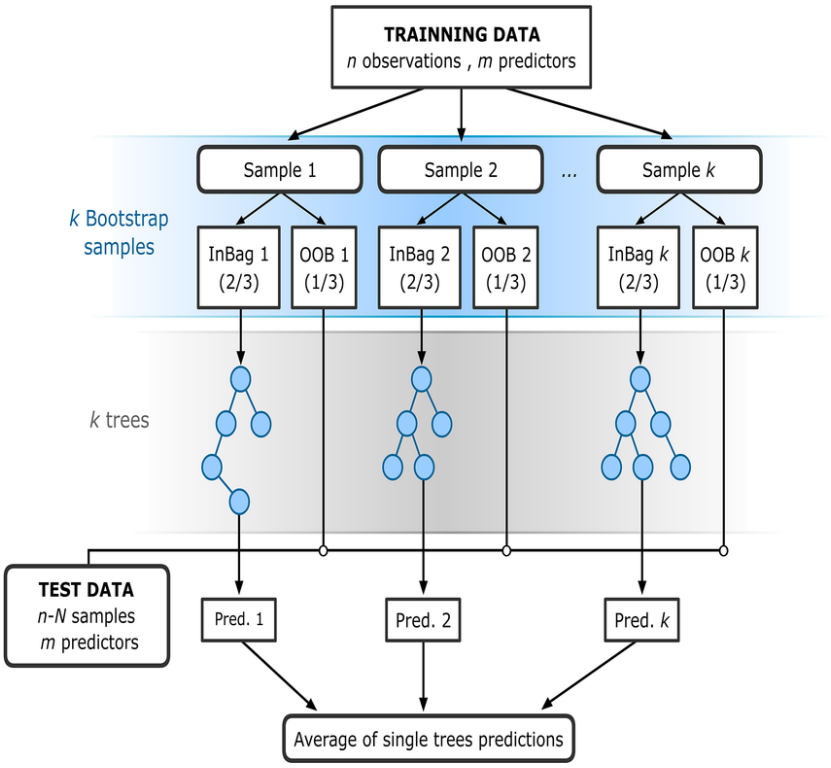


Figura 44: Diagrama de flujo de Regresor Bosques Aleatorios (Rodriguez-Galiano *et al.*, 2016)

V.5 Resumen

En este capítulo se describió la metodología llevada a cabo para entrenar los algoritmos ML, se menciona la base de datos utilizada, así como el preprocesamiento llevado a cabo sobre la misma, en el caso de las redes neuronales diseñadas se muestran sus arquitecturas y, finalmente el proceso de entrenamiento tanto de las redes neuronales diseñadas como de los modelos de regresión. El resumen de la metodología presentado en forma de diagrama de flujo se muestra en la Figura 45.

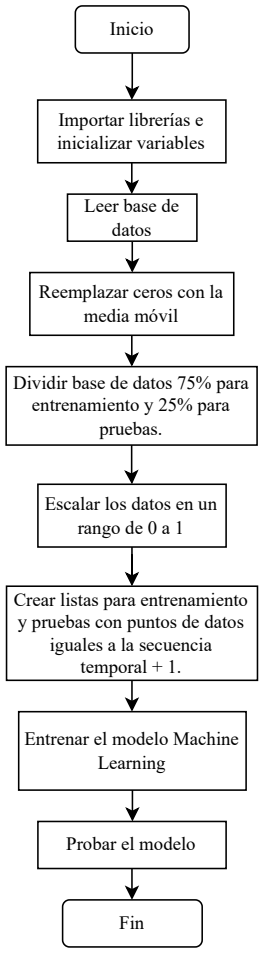


Figura 45: Diagrama de flujo del método propuesto.

Capítulo VI

Resultados

VI.1 Prototipo electrónico

Se desarrollaron dos prototipos a nivel protoboard mostrados en la figura 46, a manera de representar a los dos tipos de intersección programados, cada uno de ellos cuenta con los elementos mencionados en la sección III, además de una fuente para protoboard y una batería para alimentar el circuito.

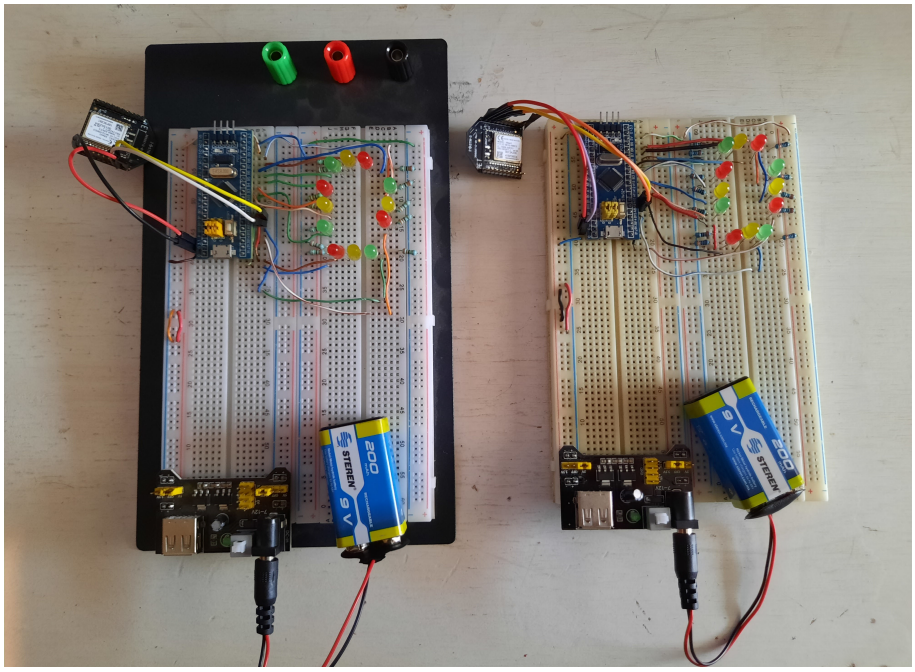
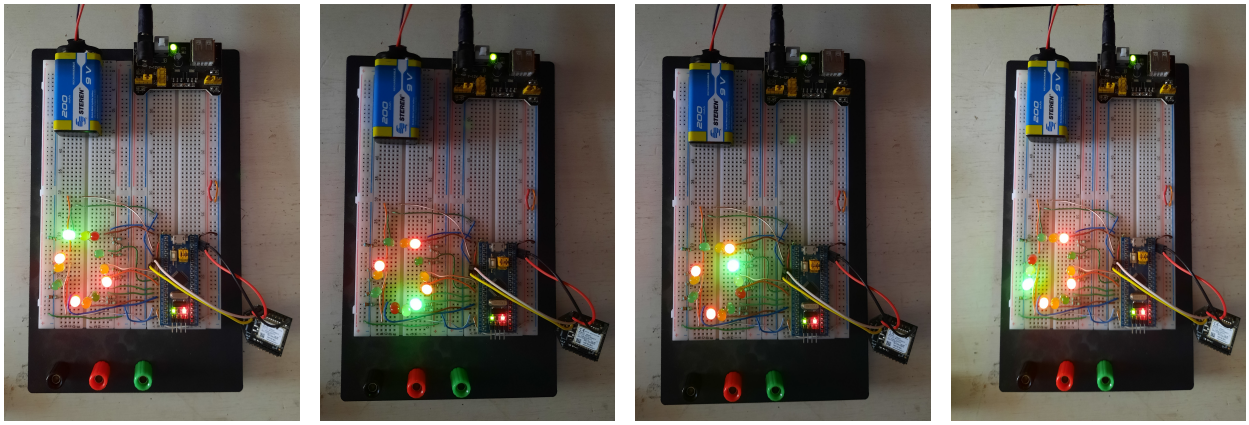


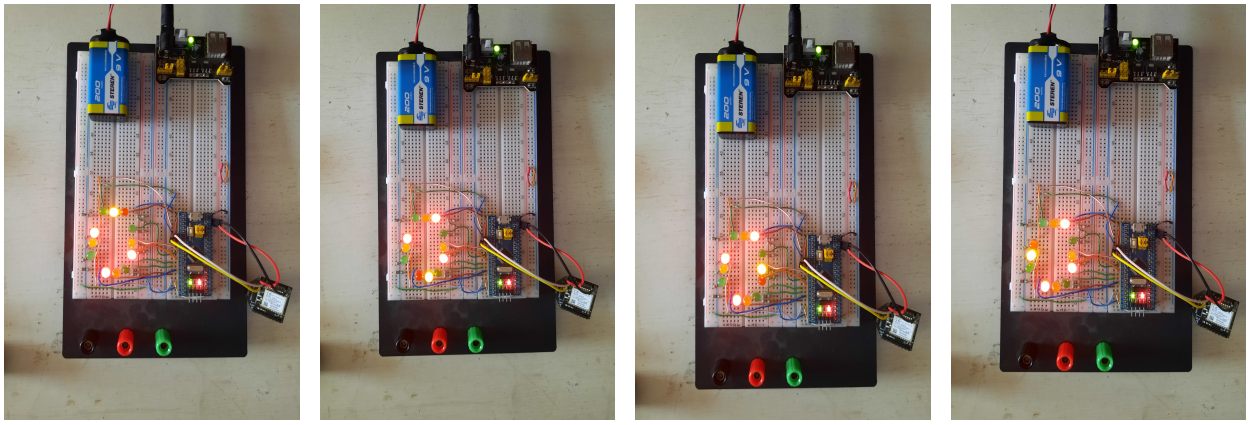
Figura 46: Prototipos a nivel protoboard.

En las figuras 47 y 48 se muestran los estados de luz verde y amarilla respectivamente, esto para el primer tipo de intersección en el que el derecho de paso se otorga a una dirección a la vez.



(a) (b) (c) (d)

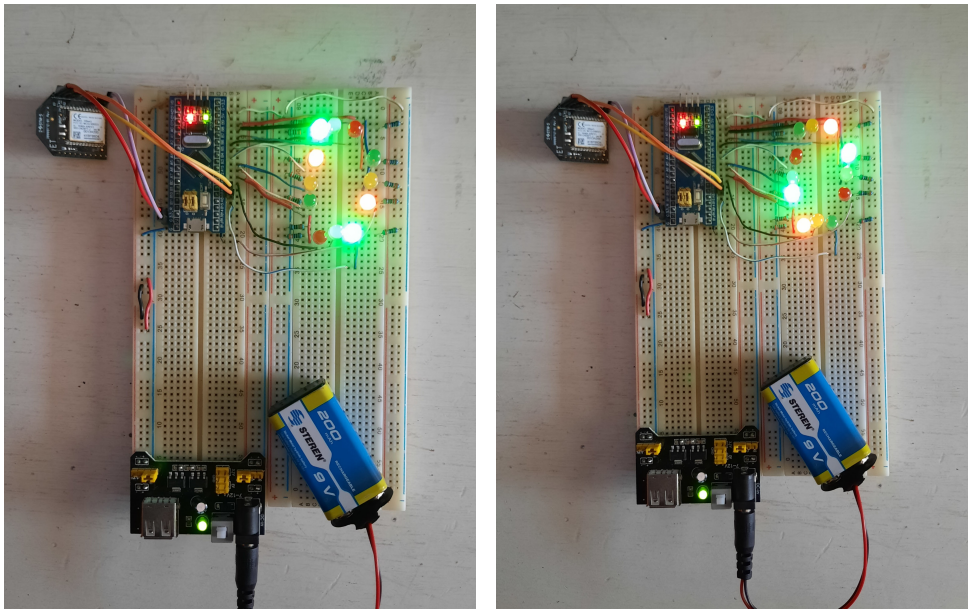
Figura 47: Estado de luz verde. (a) Dirección Norte, (b) Dirección Sur, (c) Dirección Este, (d) Dirección Oeste.



(a) (b) (c) (d)

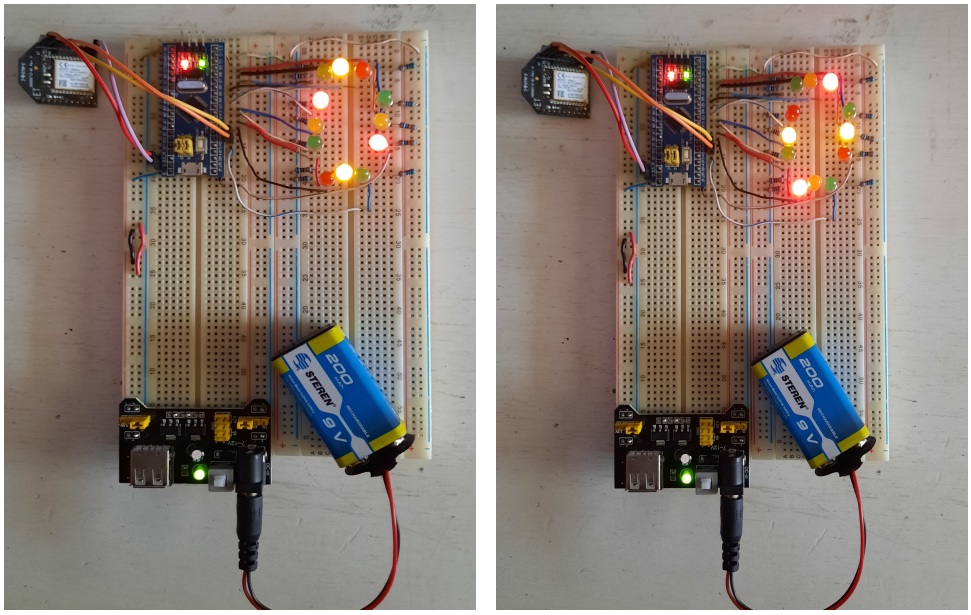
Figura 48: Estado de luz amarillo. (a) Dirección Norte, (b) Dirección Sur, (c) Dirección Este, (d) Dirección Oeste.

En las figuras 49 y 50 se muestran los estados de luz verde y amarilla para el segundo tipo de intersección, en el que se les otorga el derecho de paso a las direcciones que no causan un conflicto directo entre sí (Norte y Sur, Este y Oeste).



(a) (b)

Figura 49: Estado de luz verde. (a) Dirección Norte-Sur, (b) Dirección Este-Oeste.



(a) (b)

Figura 50: Estado de luz amarillo. (a) Dirección Norte-Sur, (b) Dirección Este-Oeste.

El modo nocturno se muestra en la figura 51, en la cual se aprecia que en las

direcciones norte y sur de ambas intersecciones parpadean las luces amarillas, mientras que en las direcciones este y oeste parpadean las luces rojas. Ejemplificando así, el caso de la avenida principal y las calles que la cruzan.

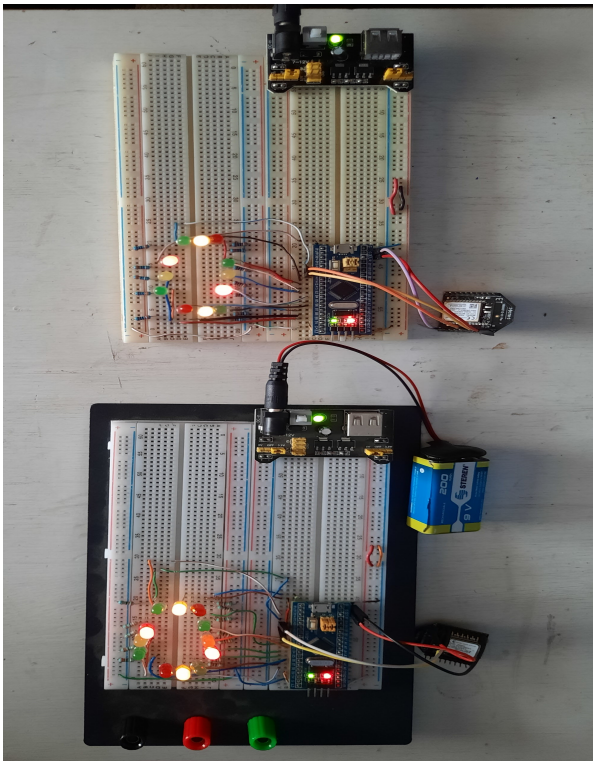


Figura 51: Modo nocturno.

En la figura 52 se muestran un ejemplo de control de las intersecciones en estado de emergencia, en él se ponen las luces con dirección norte en verde para, por ejemplo, permitir el paso de una ambulancia.

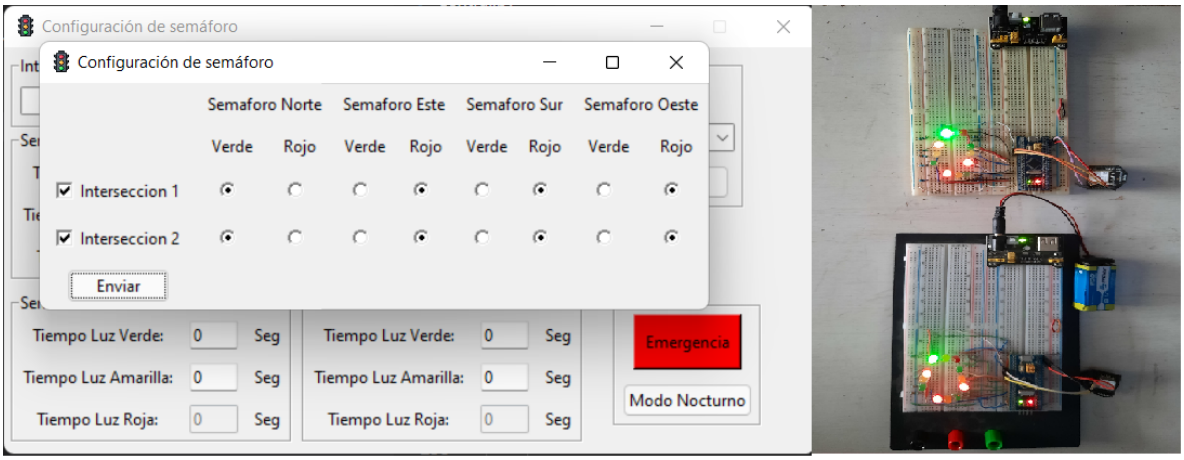


Figura 52: Control de emergencia de las intersecciones.

El prototipo del contador ultrasónico se muestra en la figura 53, consiste en tres sensores ultrasónicos que permitirían contabilizar el número de vehículos que cruzan por tres carriles de una vía, además del modulo de comunicación inalámbrica que transmite las lecturas a la computadora central, todo esto alimentado por una batería de 9 V conectada a una fuente de protoboard, la cual permite alimentar con 3.3 V el módulo XBee y con 5 V los sensores para el rango máximo de funcionamiento.

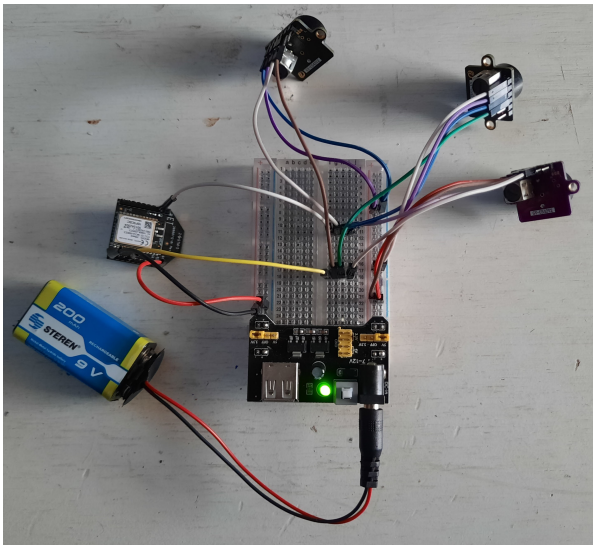


Figura 53: Prototipo de contador ultrasónico.

VI.2 Métricas de rendimiento algoritmos ML

Para evaluar el rendimiento de los algoritmos ML y DL, primero se aplica un escalado inverso a los valores de prueba 'y'. A continuación, nos apoyamos en las métricas de la biblioteca scikit-learn (Pedregosa *et al.*, 2011), el error medio absoluto (MAE) (Chen *et al.*, 2020; Cai *et al.*, 2020), el error cuadrático medio (RMSE) (Li y Shahabi, 2018; Luo *et al.*, 2019), error medio porcentual absoluto (MAPE) (Ferreira *et al.*, 2019; Haghighat *et al.*, 2020), R-cuadrado (R^2) (Boukerche y Wang, 2020b; Boukerche *et al.*, 2020) y varianza explicada (EV). Se definen como:

$$MAE(y, \widehat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} |y_i - \widehat{y}_i| \tag{4}$$

$$MAPE(y, \widehat{y}) = \frac{100\%}{n} \sum_{i=0}^{n-1} \frac{|y_i - \widehat{y}_i|}{y_i} \tag{5}$$

$$RMSE(y, \widehat{y}) = \left[\frac{1}{n} \sum_{i=0}^{n-1} (y_i - \widehat{y}_i)^2 \right]^{\frac{1}{2}} \tag{6}$$

$$R^2(y, \widehat{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2} \tag{7}$$

$$EV(y, \widehat{y}) = 1 - \frac{Var\,[y - \widehat{y}]}{Var\,[y]} \tag{8}$$

donde n es el número de muestras, y es el flujo de tráfico observado, $\widehat{y}$ es el flujo de tráfico predicho y $\overline{y}$ es la media.

MAE y RMSE miden los errores de predicción absolutos, y MAPE mide los errores de predicción relativos. Los números más pequeños indican un mayor rendimiento de predicción para estas tres métricas (Cai *et al.*, 2020). Los valores de R^2 y EV oscilan entre cero y uno, y cuanto más cerca del valor de uno, mejor se ajusta el modelo de regresión.

La tabla IV enumera las métricas de rendimiento de cada uno de los modelos ML y DL, Perceptrón Multicapa y Potenciación del Gradiente obtuvieron R-Cuadrado y Varianza Explicada por encima de 0,93, MAE de 10,8, MAPE de 21% y RMSE de 15,4. En cambio, Bosques Aleatorios tuvo un R-cuadrado y una varianza explicada ligeramente inferiores a 0,93, un MAE de 10,88, un MAPE de 21% y un RMSE de 15,5. Mientras que GRU y LSTM obtuvieron un R-cuadrado y una varianza explicada cercanos a 0,92, un MAE de 10,88, un MAPE de 22% y un RMSE de 15,6, el R-cuadrado y la varianza explicada de la Regresión Lineal fueron de 0,926, un MAE de 11,2, un MAPE de 24% y un RMSE de 15,85; finalmente, el Gradiente Estocástico tuvo un R-cuadrado y una varianza explicada de 0,9, un MAE de 12,8, un MAPE de 29% y un RMSE de 18.

Para los resultados mostrados, las RNN se entrenan iterativamente diez veces, y se calcula la media de cada métrica; en el caso de los modelos ML(scikit-learn), el estado aleatorio nos permite tener los mismos resultados.

Tabla IV: Comparación de las métricas de rendimiento utilizando el primer conjunto de datos (Axenie y Bortoli, 2020).

Modelo ML / DL	**MAE**	**MAPE**	**RMSE**	$\boldsymbol{R^2}$	**EV Score**
MLP-NN	10.8281	21.1593%	15.4202	0.9304	0.9307
Potenciación del Gradiente	10.8508	21.9493%	15.4121	0.9305	0.9306
Bosques Aleatorios	10.8827	21.8392%	15.5481	0.9296	0.9297
GRU	10.8843	22.8492%	15.6191	0.9278	0.9295
LSTM	10.8806	22.3244%	15.6771	0.9267	0.9287
Regresión Lineal	11.2010	24.3238%	15.8545	0.9263	0.9264
Gradiente Estocástico	12.8230	29.0075%	18.3727	0.9003	0.9004

Además, se han realizado pruebas de robustez utilizando un conjunto de datos diferente (Fu *et al.*, 2017) al utilizado inicialmente para el entrenamiento y la validación. Estos nuevos datos proceden del conjunto de datos PeMS, que cuenta con más de 15.000 sensores desplegados en todo el estado de California, concretamente en el cuarto distrito, que se encuentra en el área de la bahía, Alameda, Oakland de EE.UU. Para

la robustez, R^2 y la puntuación EV son buenos parámetros porque estas métricas son adimensionales, funcionan para diferentes conjuntos de datos de diferentes escalas y están normalizadas. Las métricas obtenidas con el conjunto de datos externo se enumeran en la tabla V. Comparando los resultados de R^2 y la puntuación EV que aparecen en las tablas IV y V, se puede observar que en ambos casos R^2 y la puntuación EV están entre los valores de 0,9 y 0,95. Por lo tanto, dado que R^2 y EV se mantienen dentro del mismo rango independientemente del conjunto de datos, podemos confirmar que los modelos propuestos son robustos para la predicción del flujo de tráfico.

Tabla V: Métricas de rendimiento utilizando el segundo conjunto de datos (PeMS) Fu *et al.* (2017).

Modelo ML / DL	**MAE**	**MAPE**	**RMSE**	$\boldsymbol{R^2}$	**EV Score**
MLP-NN	7.2427	18.2176	9.8096	0.9393	0.9395
Potenciación del Gradiente	7.12151	17.6224	9.6648	0.941	0.941
Bosques Aleatorios	7.05046	17.3788	9.5799	0.9421	0.9421
GRU	7.64266	18.5307	10.2406	0.9338	0.9381
LSTM	7.32852	19.0923	9.8816	0.9384	0.9388
Regresión Lineal	7.51693	20.3822	10.1914	0.9344	0.9344
Gradiente Estocástico	8.39243	23.7443	11.3199	0.9191	0.9194

Por otro lado, para analizar el coste-beneficio de la implementación de los modelos, se obtuvo el tiempo medio de entrenamiento de cada uno de ellos, los experimentos se realizaron en el entorno de ejecución de Google Colaboratory (Bisong, 2019), y además se utilizó Weights & Biases (Biewald, 2020) para llevar un control de los tiempos.

La figura 54 representa el tiempo de entrenamiento de los siete modelos ML probados en este estudio. Es importante señalar que el entrenamiento de los modelos de scikit learn lleva menos tiempo que el de las RNN diseñadas. Los modelos LSTM y GRU son los que tienen un mayor tiempo de entrenamiento 321 y 250 segundos respectivamente, entre los modelos de scikit learn Bosques Aleatorios, Potenciación del Gradiente, Regresión Lineal, Gradiente Estocástico y MLP-NN 46, 28, 20, 20 y 18 segundos siendo MLP-NN el más rápido y es el modelo que también tiene mejores métricas

de rendimiento, como se muestra en la Tabla IV.

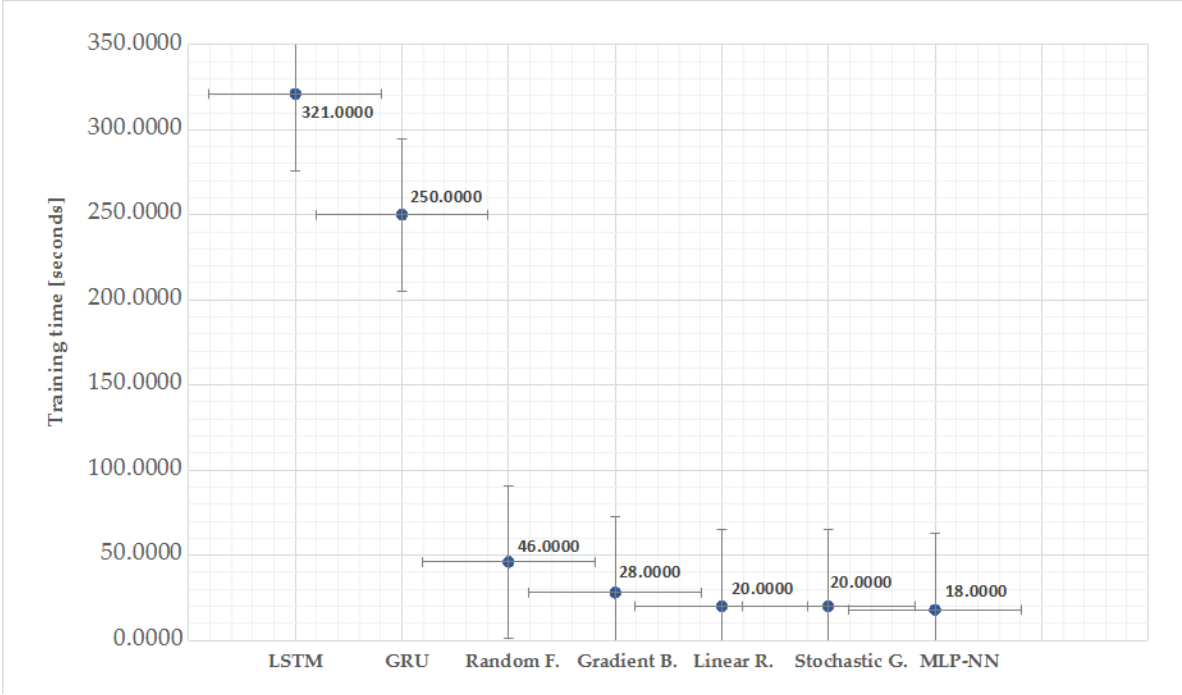


Figura 54: Comparación de los tiempos de entrenamiento.

Las predicciones realizadas sobre la división de la prueba para un día en los cuatro carriles se muestran en la Figura 55 y para toda una semana se representan en la Figura 56.

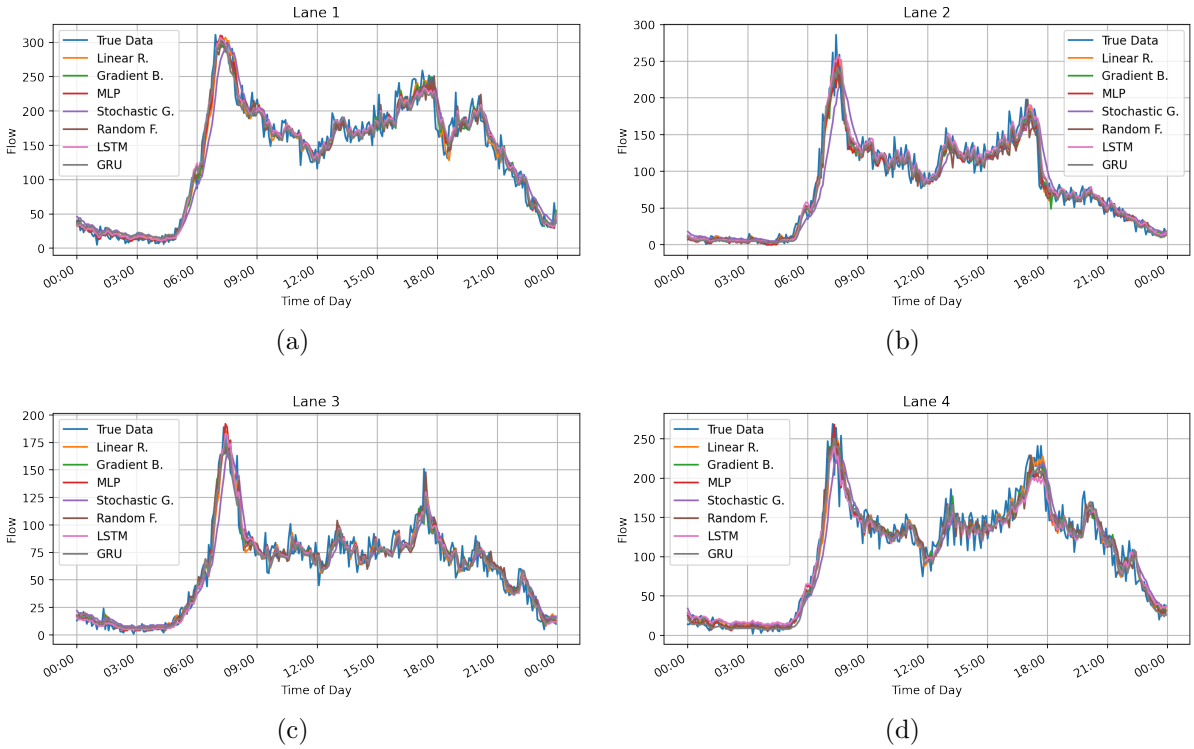


Figura 55: Comparación de los modelos de predicción del flujo de tráfico para un día de prueba. Predicción del flujo de tráfico del carril 1 (**a**), carril 2 (**b**), carril 3 (**c**) y carril 4 (**d**).

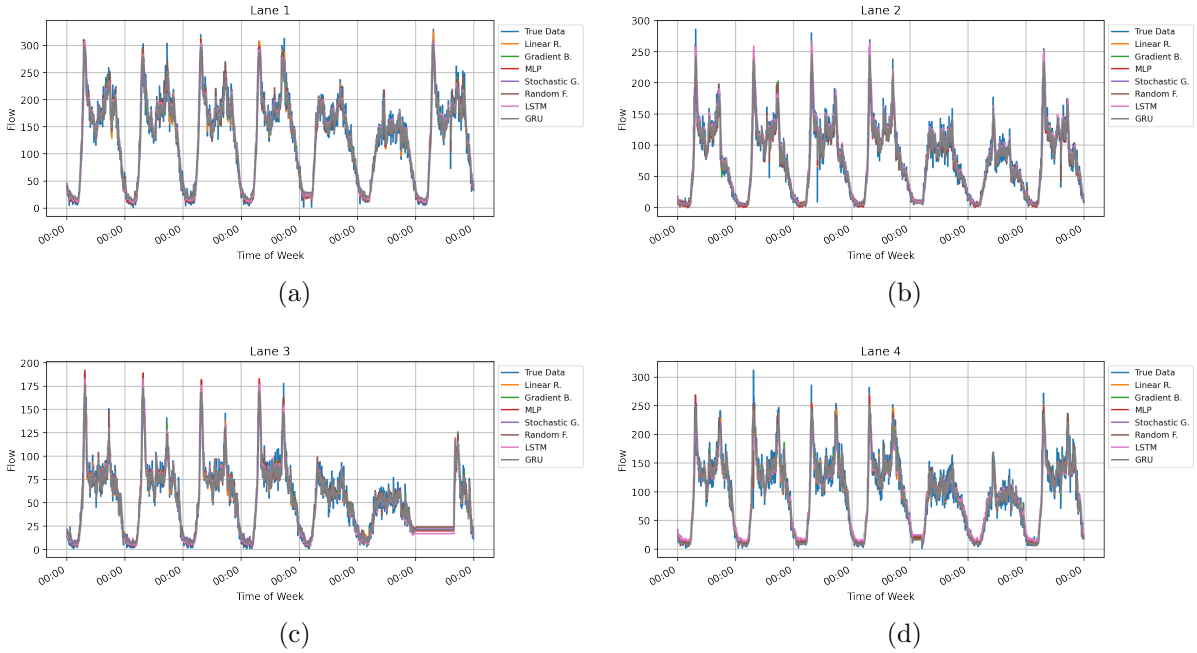


Figura 56: Comparación de los modelos de predicción del flujo de tráfico durante toda una semana. Predicción del flujo de tráfico del carril 1 (**a**), carril 2 (**b**), carril 3 (**c**) y carril 4 (**d**).

VI.3 Resumen

En este capítulo se mostraron los resultados de los prototipos desarrollados a lo largo de este trabajo de tesis, asimismo, se presentaron y analizaron las métricas de rendimiento de los algoritmos machine learning propuestos obteniendo buenos resultados al tener puntuaciones cercanas a los valores ideales. Se demostró la robustez de los modelos al evaluarlos con dos bases de datos distintas y mantener valores similares entre sí. Por último, se analizó el costo-beneficio de la implementación de cada uno de los modelos al obtener los tiempos de entrenamiento y compararlos, donde los modelos scikit tienen la ventaja de tomar menos de un minuto en entrenarse.

Capítulo VII

Conclusiones generales

En este trabajo de tesis, se presentó el desarrollo de un prototipo de controlador de semáforo inteligente basado en el microcontrolador STM32F103 dotado con comunicación inalámbrica mediante el uso de un módulo XBee 3. Se realizó una interfaz gráfica de usuario que permite llevar a cabo el control y la programación de los semáforos de manera remota desde una computadora central. De esta manera, se aporta una alternativa para agilizar el tráfico de la ciudad al hacer factible modificar los tiempos de los semáforos en distintos horarios siguiendo la dinámica del flujo vehicular, así como el control en situaciones de emergencia.

Como parte complementaria, se propuso y desarrolló a nivel prototipo un contador de vehículos basado en sensores ultrasónicos con comunicación inalámbrica que resulta factible de utilizar en la ciudad debido a su naturaleza no intrusiva, brindando una alternativa de solución a la falta de estudios para optimizar el tránsito que se tiene en la localidad.

Por último, se evaluaron diversos modelos ML para la predicción del flujo de tráfico vehicular, sentando así las bases para el control adaptativo del tráfico. Se utilizaron dos conjuntos de datos públicos para entrenar, validar y probar los modelos propuestos. Los resultados experimentales mostraron que el regresor perceptrón multicapa tiene un mejor rendimiento y requiere menos tiempo de procesamiento para entrenar (18 s). El regresor potenciación de gradiente tiene un rendimiento similar, pero requiere más tiempo de procesamiento (28 s). Tanto las RNN como el regresor de bosques aleatorios tienen una puntuación similar. Sin embargo, las RNN tardan más tiempo en entrenar

(entre 250 y 321 s). Por último, la regresión lineal y el regresor de gradiente estocástico tienen un buen tiempo de procesamiento (20 s), pero son los que peor rendimiento tienen entre estos modelos. Todos los modelos ML consiguen una puntuación de varianza explicada (EV Score) y R-cuadrado (R^2) superior a 0,90, MAE cercano a 10; el RMSE es cercano a 15 y el MAPE está entre el 20 y el 30 por ciento. En realidad, el rendimiento de estos siete algoritmos no difiere significativamente. En conclusión, los resultados fueron satisfactorios para predecir el flujo de tráfico en los cuatro carriles de una intersección, demostrando la viabilidad de implementar estos modelos en controladores de semáforos inteligentes.

VII.1 Trabajo futuro

Debido a las amplias posibilidades que se tienen con respecto al control del tráfico gracias a las tecnologías del IoT y los algoritmos de IA, se cuenta con un amplio campo de desarrollo, a continuación se enlistan posibles trabajos a futuro:

- Desarrollar la tarjeta electrónica del controlador e implementarla en una intersección.
- Agregar nuevos tipos de intersecciones e incluir los cruces peatonales.
- Implementar sensores de corriente para detectar lamparas fundidas del semáforo.
- Desarrollar e implementar una red inalámbrica de sensores para monitorear intersecciones de la ciudad y crear una base de datos propia para el entrenamiento de uno de los algoritmos ML.
- Implementar un algoritmo de optimización para la asignación de los tiempos de los estados basado en las predicciones del modelo ML.
- Implementar los modelos entrenados y optimizados en Edge Computing Device para predicción en tiempo real.

- Predicción del tráfico utilizando visión artificial.

Bibliografía

Alaidi, A. H. M., Aljazaery, I. A., AlRikabi, H. T. S., Mahmood, I. N., y Abed, F. T. (2020). Design and implementation of a smart traffic light management system controlled wirelessly by arduino. *International Journal of Interactive Mobile Technologies*, **14**(7): 32–40.

Alam, I., Md. Farid, D., y Rossetti, R. J. (2019). The prediction of traffic flow with regression analysis. En A. Abraham, P. Dutta, J. K. Mandal, A. Bhattacharya, y S. Dutta, editores, *Advances in Intelligent Systems and Computing*, Vol. 813, páginas 661–671, Singapore. Springer Singapore. ISBN 9789811314971.

Appiah, O., Quayson, E., y Opoku, E. (2020). Ultrasonic sensor based traffic information acquisition system; a cheaper alternative for ITS application in developing countries. *Scientific African*, **9**: e00487.

Axenie, C. y Bortoli, S. (2020). Road traffic prediction dataset.

Baturynska, I. y Martinsen, K. (2021). Prediction of geometry deviations in additive manufactured parts: comparison of linear regression with machine learning algorithms. *Journal of Intelligent Manufacturing*, **32**.

Bhaskar, L., Sahai, A., Sinha, D., Varshney, G., y Jain, T. (2016). Intelligent traffic light controller using inductive loops for vehicle detection. En *Proceedings on 2015 1st International Conference on Next Generation Computing Technologies, NGCT 2015*, páginas 518–522. ISBN 9781467368094.

Biewald, L. (2020). Experiment Tracking with Weights & Biases. *Software available from wandb.com*, páginas 1–5.

Bisong, E. (2019). Google Colaboratory. En E. Bisong, editor, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, páginas 59–64. Apress, Berkeley, CA. ISBN 978-1-4842-4470-8.

Blázquez, J. P. (2015). Introducción a los sistemas de comunicación inalámbricos. *Universitat Oberta de Catalunya*.

Boukerche, A. y Wang, J. (2020a). Machine Learning-based traffic prediction models for Intelligent Transportation Systems. *Computer Networks*, **181**: 107530.

Boukerche, A. y Wang, J. (2020b). A performance modeling and analysis of a novel vehicular traffic flow prediction system using a hybrid machine learning-based model. *Ad Hoc Networks*, **106**: 102224.

Boukerche, A., Tao, Y., y Sun, P. (2020). Artificial intelligence-based vehicular traffic flow prediction methods for supporting intelligent transportation systems. *Computer Networks*, **182**: 107484.

Bull, A. y Thomson, I. (2002). Urban traffic congestion: its economic and social causes and consequences. *Cepal review*.

Cai, L., Janowicz, K., Mai, G., Yan, B., y Zhu, R. (2020). Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS*, **24**(3): 736–755.

Cantú, R. A. (1976). *Los semáforos y el control dinámico del tránsito*. Representaciones y Servicios de Ingeniería.

Chavan, S. S., Deshpande, R. S., y Rana, J. G. (2009). Design of intelligent traffic light controller using embedded system. En *2009 2nd International Conference on Emerging Trends in Engineering and Technology, ICETET 2009*, páginas 1086–1091. ISBN 9780769538846.

Chen, C., Wang, Y., Li, L., Hu, J., y Zhang, Z. (2012). The retrieval of intra-day trend and its influence on traffic prediction. *Transportation Research Part C: Emerging Technologies*, **22**: 103–118.

Chen, C., Liu, B., Wan, S., Qiao, P., y Pei, Q. (2021). An Edge Traffic Flow Detection

Scheme Based on Deep Learning in an Intelligent Transportation System. *IEEE Transactions on Intelligent Transportation Systems*, **22**(3): 1840–1852.

Chen, Q., Song, Y., y Zhao, J. (2020). Short-term traffic flow prediction based on improved wavelet neural network. *Neural Computing and Applications*, **33**(14): 8181–8190.

Chen, Y. R., Chen, K. P., y Hsiung, P. A. (2016). Dynamic traffic light optimization and control system using model-predictive control method. En *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, páginas 2366–2371. ISBN 9781509018895.

Chollet, F. *et al.* (2015). Keras. Documentation at keras.io.

Diaz, N., Guerra, J., y Nicola, J. (2018). Smart Traffic Light Control System. En *2018 IEEE 3rd Ecuador Technical Chapters Meeting, ETCM 2018*, páginas 1–4. ISBN 9781538666579.

Ferreira, Y. M., Frank, L. R., Julio, E. P., Ferreira, F. H. C., Dembogurski, B. J., y Silva, E. F. (2019). Applying a Multilayer Perceptron for Traffic Flow Prediction to Empower a Smart Ecosystem. En *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 11619 LNCS, páginas 633–648. Springer Nature Switzerland. ISBN 9783030242886.

Fu, R., Zhang, Z., y Li, L. (2017). Using LSTM and GRU neural network methods for traffic flow prediction. En *Proceedings - 2016 31st Youth Academic Annual Conference of Chinese Association of Automation, YAC 2016*, páginas 324–328. ISBN 9781509044238.

Garcia, J. (2021). No se puede solucionar el tráfico de ensenada sin un estudio. Reuperado de `https://www.elimparcial.com/tijuana/ensenada/No-se-puede-solucionar-el-trafico-de-Ensenada-sin-un-estudio---20211028-0020.html`.

George, S. y Santra, A. K. (2020). Traffic Prediction Using Multifaceted Techniques: A Survey. *Wireless Personal Communications*, **115**(2): 1047–1106.

Ghazal, B., Elkhatib, K., Chahine, K., y Kherfan, M. (2016). Smart traffic light control system. En *2016 3rd International Conference on Electrical, Electronics, Computer Engineering and their Applications, EECEA 2016*, páginas 140–145. ISBN 9781467369428.

Guerillot, D. y Bruyelle, J. (2017). Uncertainty assessment in production forecast with an optimal artificial neural network.

Haghighat, A. K., Ravichandra-Mouli, V., Chakraborty, P., Esfandiari, Y., Arabi, S., y Sharma, A. (2020). Applications of Deep Learning in Intelligent Transportation Systems. *Journal of Big Data Analytics in Transportation*, **2**(2): 115–145.

Hosseini, S. H., Moshiri, B., Rahimi-Kian, A., y Araabi, B. N. (2014). Traffic flow prediction using mi algorithm and considering noisy and data loss conditions: An application to minnesota traffic flow prediction. *Promet - Traffic - Traffico*, **26**(5): 393–403.

Hou, Y., Chen, J., y Wen, S. (2021). The effect of the dataset on evaluating urban traffic prediction. *Alexandria Engineering Journal*, **60**(1): 597–613.

Hussain, B., Afzal, M. K., Ahmad, S., y Mostafa, A. M. (2021). Intelligent Traffic Flow Prediction Using Optimized GRU Model. *IEEE Access*, **9**: 100736–100746.

Jiang, C. Y., Hu, X. M., y Chen, W. N. (2021). An Urban Traffic Signal Control System Based on Traffic Flow Prediction. En *2021 13th International Conference on Advanced Computational Intelligence (ICACI)*, páginas 259–265. ISBN 9781665412544.

Lawe, S. y Wang, R. (2016). Optimization of traffic signals using deep learning neural networks. En B. H. Kang y Q. Bai, editores, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in*

Bioinformatics), Vol. 9992 LNAI, páginas 403–415, Cham. Springer International Publishing. ISBN 9783319501260.

Li, J., Boonaert, J., Doniec, A., y Lozenguez, G. (2021). Multi-models machine learning methods for traffic flow estimation from Floating Car Data. *Transportation Research Part C: Emerging Technologies*, **132**: 103389.

Li, Y. y Shahabi, C. (2018). A brief overview of machine learning methods for short-term traffic forecasting and future directions. *SIGSPATIAL Special*, **10**(1): 3–9.

Li, Z., Li, C., Zhang, Y., y Hu, X. (2017). Intelligent traffic light control system based on real time traffic flows. En *2017 14th IEEE Annual Consumer Communications and Networking Conference, CCNC 2017*, páginas 624–625. ISBN 9781509061969.

Luo, C., Huang, C., Cao, J., Lu, J., Huang, W., Guo, J., y Wei, Y. (2019). Short-Term Traffic Flow Prediction Based on Least Square Support Vector Machine with Hybrid Optimization Algorithm. *Neural Processing Letters*, **50**(3): 2305–2322.

Manasi, P. S., Nishitha, N., Pratyusha, V., y Ramesh, T. K. (2020). Smart Traffic Light Signaling Strategy. En *Proceedings of the 2020 IEEE International Conference on Communication and Signal Processing, ICCSP 2020*, páginas 1200–1203. ISBN 9781728149882.

McShane, C. (1999). The origins and globalization of traffic control signals. *Journal of Urban History*, **25**(3): 379–404.

Nguyen-Ly, T. T., Tran, L., y Huynh, T. V. (2019). Low-cost, high-efficiency hardware implementation of smart traffic light system. En *Proceedings - 2019 International Symposium on Electrical and Electronics Engineering, ISEE 2019*, páginas 28–32. ISBN 9781728153537.

Oyarce, A., Aguayo, P., y Martin, E. (2010). Guía del usuario xbee series 1. *Ingeniería MCI Ltda*, **40**.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., y Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Documentation at scikit-learn.org.

Prasetyo, M. A., Latuconsina, R., y Purboyo, T. W. (2018). A proposed design of traffic congestion prediction using ultrasonic sensors. *International Journal of Applied Engineering Research*, **13**(1): 434–441.

Rafael, C. A. L., Mayor, R., y James, C. G. (2018). *Ingeniería de tránsito, Fundamentos y aplicaciones*. Alfaomega, 9ªedición. México.

Rodrigo, J. A. (2020). Gradient boosting con python.

Rodriguez-Galiano, V., Sánchez Castillo, M., Dash, J., Atkinson, P., y Ojeda-Zujar, J. (2016). Modelling interannual variation in the spring and autumn land surface phenology of the european forest. *Biogeosciences*, **13**: 3305–3317.

Saleh, A., Adeshina, S. A., Galadima, A., y Ugweje, O. (2018). An intelligent traffic control system. En *2017 13th International Conference on Electronics, Computer and Computation, ICECCO 2017*, Vol. 2018-January, páginas 1–6. ISBN 9781538625019.

Salvat (1914). *Diccionario Salvat enciclopedico popular ilustrado: (inventario Del Saber Humano)*, **(1906-1914)**.

Segal, M. R. (2004). Machine learning benchmarks and random forest regression.

Serrano, Á., Conde, C., Rodríguez-Aragón, L. J., Montes, R., y Cabello, E. (2005). Computer vision application: Real time smart traffic light. En *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 3643 LNCS, páginas 525–530. ISBN 3540290028.

Taud, H. y Mas, J. (2018). Multilayer perceptron (mlp). En *Geomatic approaches for modeling land change scenarios*, páginas 451–455. Springer.

Tran, H. (2019). Survey of machine learning and data mining techniques used in multimedia system.

Wang, R. (2021). Optimal method of intelligent traffic signal light timing based on genetic neural network. *Advances in Transportation Studies*, **2021**(Special issue 1): 3–12.

Xiong, G., Wang, K., Zhu, F., Cheng, C., An, X., y Xie, Z. (2010). Parallel traffic management for the 2010 Asian Games. *IEEE Intelligent Systems*, **25**(3): 81–85.

Yuan, H. y Li, G. (2021). A Survey of Traffic Prediction: from Spatio-Temporal Data to Intelligent Transportation. *Data Science and Engineering*, **6**(1): 63–85.

Apéndice A

Publicaciones derivadas del trabajo de tesis

Publicación en revista JCR

A. Navarro-Espinoza, O. R. López-Bonilla, E. E. García-Guerrero, E. Tlelo-Cuautle, D. López-Mancilla, C. Hernández-Mejía, E. Inzunza-González. 2022. *Traffic Flow Prediction for Smart Traffic Lights using Machine Learning Algorithms.* MDPI Technologies Vol. 10, Issue 1, 5. DOI: `https://doi.org/10.3390/technologies10010005`

Presentación en Congreso Internacional

A. Navarro-Espinoza, O. R. López-Bonilla, E. E. García-Guerrero, E. Tlelo-Cuautle, D. López-Mancilla, C. Hernández-Mejía, E. Inzunza-González, 2021. *Traffic flow prediction for smart traffic lights.* 9$^{\text{th}}$ International Workshop on Numerical and Evolutionary Optimization.

Publicaciones en colaboración

A. Navarro-Espinoza, E. Inzunza-González, O. R. López-Bonilla, E. E. García-Guerrero, 2020. *Diseño y Construcción de un Controlador de Semáforo Inteligente con Comunicación Inalámbrica.* 7mo Simposio Internacional de Investigación "Perspectivas científico tecnológicas sobre investigación, desarrollo e innovación", 102-110, ISBN: 978-0-9998657-4-3.

Presentaciones digitales colaborativas

A. Navarro-Espinoza, E. E. Contreras Lujan, E. R. Ramos Acosta, E. Inzunza-González, E. E. García-Guerrero, O. R. López-Bonilla, 2021. *Machine learning for kids.* XXVIII Jornadas de Ingeniería, Arquitectura y Diseño. `https://youtu.be/JkNhe0fkGtI`

A. Navarro-Espinoza, E. Inzunza-González, E. E. García-Guerrero, O. R. López-Bonilla, 2020. *Aplicaciones del internet de las cosas en ciudades inteligentes.* Expociencia y tecnología 2020. `https://www.youtube.com/watch?v=n5k6OpnhY1M`

E. E. Contreras Lujan, **A. Navarro-Espinoza**, E. R. Ramos Acosta, E. Inzunza-González, E. E. García-Guerrero, O. R. López-Bonilla, 2021. *Machine learning aplicado al diagnostico de enfermedades.* XXVIII Jornadas de Ingeniería, Arquitectura y Diseño. `https://www.youtube.com/watch?v=lzkowEpeYis`

E. R. Ramos Acosta, **A. Navarro-Espinoza**, E. E. Contreras Lujan, E. Inzunza-González, E. E. García-Guerrero, O. R. López-Bonilla, 2021. *Aprendizaje Profundo Aplicado en la Industria.* XXVIII Jornadas de Ingeniería, Arquitectura y Diseño. `https://www.youtube.com/watch?v=zdOKr_dT3e8`