Universidad de Cuenca

Facultad de Ingeniería

Carrera de Electrónica y Telecomunicaciones

Sistema de detección de plazas disponibles para los parqueaderos del campus central de la Universidad de Cuenca.

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones.

Autores :

Jairo Miguel Lara Serrano — C.I. 140077388-1
miguel.laras@ucuenca.edu.ec

Josué David Pazmiño Moreira — C.I. 070416089-4
josue.pazmino@ucuenca.edu.ec

Director :

Ing. Juan Manuel Andrade Rodas, PhD — C.I. 010330554-6

Co-Director :

Ing. Remigio Clemente Guevara Baculima, Mg. — C.I. 010278291-8

Cuenca - Ecuador

16 - 04 - 2021

Resumen

El presente trabajo experimental plantea una solución a la gestión espacios de parqueo dentro de los predios de la Universidad de Cuenca. Se detalla la implementación de un prototipo de sistema autónomo conformado por Redes Inalámbricas de Sensores (WSN) y Procesamiento de Imágenes, que permita detectar las plazas disponibles en un conjunto de ocho parqueaderos del estacionamiento de la Facultad de Ingeniería.

El prototipo implementado consta de dos sistemas. El primero, compuesto por dos módulos de sensores, y el segundo, basado en procesamiento de imágenes. La adquisición, acondicionamiento y procesamiento de la información se realiza en una Raspberry Pi y la visualización de los espacios disponibles se realiza por medio de módulos LED de alto brillo instalados en cada parqueo y un Panel LED para todo el sistema. La comunicación entre los módulos del sistema utiliza tecnología wifi. Los sistemas trabajan de forma independiente permitiéndonos realizar un análisis comparativo entre los dos sistemas, diferenciando costos y eficiencia de cada uno, y a su vez complementario, con el objetivo de aumentar la fiabilidad de todo el sistema general.

El sistema basado en sensores consta de dos módulos alimentados desde la red eléctrica. El primer módulo (Módulo Comparador) está conformado por tres sensores LDR de referencia, ubicados, uno en luz directa, otro semi iluminado y otro en sombra para ser utilizados en el algoritmo de detección. El sistema además incluye un conjunto de 8 resistencias de diferente valor, con el objetivo de obtener el comportamiento de cada LDR en base a la iluminación existente; cuenta con un dispositivo wemos NodeMCU v3, el cual realiza la adquisición de los datos de cada sensor LDR y los envía al Raspberry Pi. El segundo módulo (Módulo Sensor), usa un multiplexor para el grupo de ocho sensores LDR asignados a cada parqueadero y otro multiplexor para el grupo de ocho resistencias, permitiendo seleccionar la entrada que procesará el microcontrolador.

El sistema basado en visión artificial, consta de una cámara web ubicada en el tercer piso de la Facultad de Ingeniería, con el objetivo de obtener una vista panorámica de todo el estacionamiento, incluyendo los parqueos de interés. La información obtenida por la cámara será procesada en la Raspberry Pi, usando el método de Background Subtraction y Análisis de mapas de transitoriedad.

Una vez realizado el procesamiento de la información de ambos sistemas, la respuesta es enviada al Módulo Sensor y al Panel LED con el fin de indicar el parqueadero disponible más cercano.

Palabras clave : Parqueadero. Red inalámbrica de sensores. WSN. Cámara. Procesamiento de imágenes, LDR. NodeMCU. Raspberry pi. LED

Abstract

The present experimental work proposes a solution to the management of parking spaces within the Universidad de Cuenca. The implementation of an autonomous-prototype system using Wireless Sensor Networks (WSN) and image processing presented in detail, which allows detecting the availability of eight parking spaces in the School of Engineering parking lot.

The implemented prototype consists of two systems. The first, comprising two sensor modules, and the second based on image processing. The acquisition, conditioning and processing of the information is carried out by a Raspberry Pi and the visualization of the available spaces uses high brightness LED modules installed in each of the parking spaces and a single LED Panel for the whole system. Communication between the modules is done using a Wi-Fi network. Both systems work independently, allowing us to carry out a comparative analysis between the two systems, differentiating costs and efficiency of each one, and in turn complementary, with the aim of increasing the reliability of the whole system.

Both systems can be complementary, the sensor-based system consists of two modules powered from the power grid. The first module (Comparator Module) is made up of three reference LDR sensors. The sensors are located, one in direct light, another semi-illuminated and the last one in shadow to be used in the detection algorithm. The system also includes a set of eight resistors of different value, in order to obtain the behavior of each LDR based on the existing lighting; It has a wemos NodeMCU v3 device, which acquires the data from each LDR sensor and sends it to the Raspberry Pi. The second module (Sensor Module) uses a multiplexer for the group of eight LDR sensors assigned to each parking lot and another multiplexer for the group of eight resistors, allowing the input selection for the microcontroller to process.

The system based on artificial vision consists of a web camera located on the third-floor lot of the School of Engineering, in order to obtain a panoramic view of the entire parking lot, including the parking spaces of interest. The information obtained by the camera will is processed on the Raspberry Pi, using the Background Subtraction and Transience Map Analysis method.

Once the information processing of both systems has been carried out, the response is sent to the Sensor Module and the LED Panel in order to indicate the closest available parking space.

Keywords: Parking. Wireless sensor network. WSN. Camera. Image processing. Ldr. NodeMCU. Raspberry pi. Led

Índice general

Índice de figuras

Índice de tablas

Cláusula de Propiedad Intelectual

Yo Jairo Miguel Lara Serrano, autor del trabajo de titulación "Sistema de detección de plazas disponibles para los parqueaderos del campus central de la Universidad de Cuenca", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor

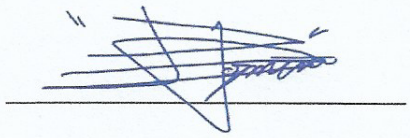
Cuenca, 15 de abril del 2021

Jairo Miguel Lara Serrano
C.I: 140077388-1

Cláusula de Propiedad Intelectual

Yo Josue David Pazmiño Moreira, autor del trabajo de titulación "Sistema de detección de plazas disponibles para los parqueaderos del campus central de la Universidad de Cuenca", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor

Cuenca, 15 de abril del 2021



Josue David Pazmiño Moreira
C.I: 070416089-4

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo Jairo Miguel Lara Serrano en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Sistema de detección de plazas disponibles para los parqueaderos del campus central de la Universidad de Cuenca", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 16 de abril del 2021

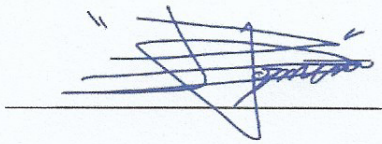
Jairo Miguel Lara Serrano
C.I: 140077388-1

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo Josue David Pazmiño Moreira en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Sistema de detección de plazas disponibles para los parqueaderos del campus central de la Universidad de Cuenca", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 16 de abril del 2021



Josue David Pazmiño Moreira
C.I:070416089-4

Dedicatoria

Mi tesis se la dedico a mi mamá Mariana Serrano y a mi papá Miguel Lara por el apoyo incondicional otorgado durante todo el transcurso de mi carrera.

A mi esposa Débora Romero por su sacrificio, entrega y motivación en mis momentos de flaqueza y por inspirarme a seguir adelante con mis planes de vida.

A mis hijas Heily y Hellen por ser mi inspiración para superarme cada día y entregarme esa energía que emiten al verme llegar a casa todas las noches después de las largas jornadas de trabajo, para ustedes mi tesis y el resto de mi vida.

Jairo Miguel Lara Serrano

Dedicatoria

A mi Dios, por darme el privilegio tan grande de tener a mis Padres Wilfrido Pazmiño y Eva Moreira, que han sabido guiarme e inculcarme valores y principios para ser una persona de bien. Los cuales no me han fallado ni un instante de mi vida y siempre han estado para alentarme y brindarme su apoyo con mucho amor.

A mi esposa Abigail Coronel y a mi hija Sophia Pazmiño, quienes se han convertido en mi apoyo, fuerza y motivación para alcanzar todas las metas que me he propuesto. Los amores de mi vida, que han complementado mi felicidad total en el transcurso de mi carrera.

A mis amigos, tutores y colegas, que han sabido ser un apoyo y me han brindado una mano en toda mi carrera Universitaria.

Josué David Pazmiño Moreira

Agradecimientos

Un agradecimiento a todas las personas que han hecho posible la culminación de mi carrera y de este trabajo de titulación.

Primero a mis padres por brindarme el apoyo y financiamiento que me permitió iniciar esta carrera.

A los Ingenieros Remigio Guevara y Juan Andrade, directores de este trabajo de titulación por la guía que nos han brindado.

A los demás miembros de la red sísmica del Austro donde nos brindaron la acogida para realizar este proyecto y por la colaboración y asesoría en los temas donde ustedes tienen mayor experiencia.

A todas las personas con las que he tenido la dicha de encontrarme y compartir este camino, porque de una forma u otra fueron parte de mi formación no solo como profesional sino como persona. A mi familia, compañeros, amigos, profesores y tutores que de forma directa o indirecta aportaron en la culminación de esta tesis.

Jairo Miguel Lara Serrano

Agradecimientos

Agradezco a Dios por haberme brindado salud y bienestar durante toda mi vida y por haberme permitido rodearme de personas que han sido muy importantes para mí y han apoyado mi proceso de educación Universitaria.

Agradezco a los Ingenieros Remigio Guevara y Juan Manuel Andrade, quienes, con su sabiduría supieron guiarnos en la elaboración de este trabajo de titulación.

Agradezco a los ingenieros que conforman la red sísmica del Austro, quienes junto con el trabajo del Ingeniero Remigio Guevara, nos brindaron la ayuda y asesoría oportunas en la instalación y composición del prototipo presentado.

De manera especial, agradezco a mis Padres, Esposa e Hija, por todo su apoyo, animo, paciencia y sacrificio, en todo tipo de circunstancias, al pasar por momentos tanto felices como dificultosos tanto en mi vida cotidiana como en mi carrera Universitaria.

Josué David Pazmiño Moreira

Abreviaciones y Acrónimos

BBM Break-Before-Make. 36, 59, 69, 70
BLE Bluetooth Low Energy. 38
BS Background Subtraction. 22, 24, 45, 46, 49, 50, 97, 98, 108, 109, 123, 124, 128

CMOS Complementary Metal Oxide Semiconductor. 36–38, 70, 80
CNN Convolutional Neural networks. 22, 48, 50, 128
CSI Camera Serial Interface. 38

DC Direct Current. 58, 59, 69
DiP Dual in Line Package. 33, 86
DSI Display Serial Interface. 38

EDL Eclipse Distribution License. 30
EMC Electromagnetic Compatibility. 41, 46, 60, 70, 83, 108, 127
EMI ElectroMagnetic Interference. 41, 46, 60, 64, 70, 76, 83, 108, 127
EPL Eclipse Public License. 30
ESD Electro Static Discharge. 36, 38, 46, 59, 60, 69, 70

GSM Global System for Mobile communications. 49

HAT Hardware Attached on TOP. 38
HBM Human-Body Model. 38, 46, 60, 70
HDMI High Definition Multimedia Interface. 38
HOG Histogram of Oriented Gradients. 49

I2C Inter Integrated Circuits. 31, 79, 82, 84, 112
IoT Internet of Things. 26, 31
IP Internet Protocol. 28, 62, 74, 84

JSON JavaScript Object Notation. 53, 63, 68, 74

LAN Local Area Network. 38
LDR Light Dependent Resistor. 22, 32, 46, 49–51, 53–57, 59, 63, 64, 68, 69, 74–77, 108, 111, 113, 114, 122, 125, 127–129
LED Light Emitting Diode. 21, 23, 33, 34, 51, 68, 69, 72, 73, 76, 77, 79–82, 84–86, 88, 103–105, 108, 111–116, 118, 119, 122, 127–130

M2M Machine to Machine. 26, 28

MIPI Mobile Industry Processor Interface. 38
MISO Master In Slave Out. 80
MM Machine Model. 38, 46, 60, 70
MoG Mixture of Gaussians. 22, 45, 49, 50, 108, 128
MOSI Master Out Slave In. 80
MQTT Message Queue Telemetry Transport. 23, 26–28, 46, 51, 53, 62, 68, 74, 84, 87, 88, 103, 129

NMOS Negative-channel Metal-Oxide Semiconductor. 37
NodeMCU Node Micro Controller Unit. 31, 46, 53, 59–64, 68–70, 74, 75, 79, 80, 82–84

PCB Printed Circuit Board. 41, 46, 60, 64, 70, 72, 75, 82, 83, 85
PDI Procesamiento Digital de Imágenes. 22, 48, 50, 51, 108, 110, 114–116, 118, 121, 123, 124, 126, 128, 129
PIR Passive Infrared. 49, 50, 127
PoE Power over Ethernet. 38
PVC PolyVinyl Chloride. 78

RFID Radio Frequency Identification. 49
ROI Región de Interés. 22, 46, 49, 50, 128
Ron Resistance On Channel. 35, 53
RPi Raspberry Pi 3. 23, 38, 46, 49–51, 53, 62, 68, 74, 84, 87, 128–130

SCK Serial Clock. 80
SCL Serial Clock. 79, 80
SDA Serial Data. 79, 84
SDK Software Development Kit. 31
SE Structuring Element. 42–44
SIFT Scale-Invariant Feature Transform. 49
SoC System On a Chip. 31, 38, 60, 70, 83
SPI Serial Peripheral Interface. 31, 34, 80, 82, 84
SSL Secure Sockets Layer. 28
SVM Support Vector Machine. 22, 48, 50, 128

TA Transience Analysis. 22, 24, 50, 108, 109, 116, 124, 128
TCP Transmission Control Protocol. 28
TLS Transport Layer Security. 28
TTL Transistor-Transistor Logic. 36, 37, 80
TWI Two-Wire Interface. 79

UART Universal Asynchronous Receiver Transmitter. 31
URL Uniform Resource Locator. 61
USB Universal Serial Bus. 31, 38

Vis Voltage Input Signal. 35, 36, 53
Vpp Peak to Peak Voltage. 36

WiFi Wireless Fidelity. 31, 60, 70
WSN Wireless Sensor Network. 22, 25, 46–48, 50, 51, 58, 108, 110, 111, 113, 114, 118, 121, 122, 124–130

Capítulo 1

Introducción

En este capítulo se expone el problema y justificación del proyecto de investigación, así como el alcance y los objetivos generales y específicos a cumplir durante su desarrollo.

1.1. Identificación del problema

El aumento de vehículos dentro de la zona urbana, se ha convertido en un factor que ha influenciado directamente al desarrollo de algunos problemas como: aumento de tráfico vehicular, desperdicio de combustible, exposición a daños materiales, contaminación, entre otros; pero una consecuencia directa en la interacción diaria de la población es la poca disponibilidad de espacios de parqueo, y de los espacios existentes, la pobre gestión para la optimización de tiempos de estacionamiento.

En el casco urbano de la ciudad, los sistemas de estacionamiento son tradicionales, donde una persona se encarga de ayudar al conductor a encontrar un espacio libre para estacionarse. De igual manera se puede apreciar que en parqueaderos pertenecientes a instituciones privadas o públicas, el espacio es ocupado rápidamente y, como consecuencia, el largo tiempo de búsqueda del espacio vacante, el tránsito del vehículo hacia el espacio adecuado y la carga de ticket manualmente, hace que el proceso sea agobiante.[1]

Estos problemas se hacen visibles en los espacios de estacionamiento del Campus Central de la Universidad de Cuenca, de tal manera que suele ser necesario recorrer dos o tres zonas de parqueo para encontrar un espacio libre, llegando incluso no tener más alternativa que apegar el automóvil a un parterre, con la consiguiente obstaculización del tránsito de los demás vehículos. En este contexto y con la finalidad de resolver todos estos inconvenientes, se propone un Sistema de Estacionamiento Inteligente, es decir, un sistema autónomo que permita detectar de plazas disponibles de los parqueaderos del campus Central de la Universidad de Cuenca y visualizar por medio de un panel Light Emitting Diode (LED) y módulos LED de alto brillo instalados en el asfalto, tomando como piloto parte del parqueadero ubicado frente a la Facultad de Ingeniería.

1.2. Justificación

Una de las técnicas más comunes en los sistemas de parqueo autónomos es el uso de cámaras y procesamiento de imágenes; sin embargo, aunque son sistemas eficientes, los problemas que acarrean este tipo de sistema son un alto consumo energético tanto como precios elevados [2]. Otra de las opciones es Wireless Sensor Network (WSN), que es una tecnología basada en el despliegue de varios sensores en un área determinada para detectar diferentes parámetros físicos. Estos sensores se comunican de manera inalámbrica entre ellos y con una base central, encargada del procesamiento de toda la información recogida por los sensores [3]. La implementación de los dos sistemas genera una solución robusta, generando un acceso rápido y fácilmente visible, en donde el sistema de WSN está compuesto por sensores Light Dependent Resistor (LDR) y el sistema de procesamiento de imágenes está compuesto por una cámara web.

El sensor LDR cambia su resistencia de forma inversamente proporcional a la cantidad de luz que incide sobre él: a medida que incrementa la energía luminosa en el sensor, su resistencia comenzará a disminuir de varios mega ohmios a unos pocos ohmios de forma exponencial [4]. En [5] se realiza una comparación entre varios sensores para su uso dentro de un sistema de estacionamiento autónomo; aunque los resultados muestran que la precisión del sensor LDR se ve afectado por el cambio de luz a lo largo del día y las condiciones climáticas, el trabajo propuesto establece diferentes valores de umbral sobre la base del cambio de intensidad luminosa para realizar comparaciones. De esta manera, los resultados tendrán mayor precisión cuando no haya una luz fuerte o exista sombra sobre la foto resistencia.

Para los sistemas basados en visión artificial existen dos métodos para la detección de espacios de parqueadero disponibles, una basada en los autos y otra basada en los espacios. En la primera se extrae características de los autos y en la segunda se extraen características de los parqueaderos vacíos [6]. Sistemas de detección utilizando Convolutional Neural networks (CNN) logran alcanzar una precisión del 99 % en la detección de parqueos disponibles [6], [7] . En [8] se realiza un sistema para detección en ambientes abiertos y se utiliza un histograma de gradientes orientados para alimentar una Support Vector Machine (SVM) en la fase de entrenamiento. Para incrementar la precisión de la detección, se elimina el fondo restringiendo el análisis a una Región de Interés (ROI) y se elimina de la detección objetos extraños como peatones bloqueando esta región de interés. Con estas consideraciones se obtiene una precisión del 91.1 % hasta 98.8 % dependiendo de las condiciones ambientales. En [9] se realiza un análisis temporal de los fotogramas de video para detectar la variación de parqueaderos libres y ocupados. Se combinan las técnicas de Background Subtraction (BS) utilizando Mixture of Gaussians (MoG) para detectar y rastrear vehículos y un análisis de transitoriedad Transience Analysis (TA) para detectar el estacionamiento y la salida de vehículos. El método propuesto está diseñado para abordar los principales desafíos de los escenarios urbanos (múltiples objetos en movimiento, condiciones de iluminación exterior y oclusiones entre vehículos) sin entrenamiento.

1.3. Alcance

El presente trabajo experimental de titulación detalla la implementación de un prototipo de sistema autónomo conformado por un sistema de WSN y otro sistema de Procesamiento Digital de Imágenes (PDI), que permita detectar las plazas disponibles en un conjunto de ocho parqueaderos del estacionamiento de la Facultad de Ingeniería.

El sistema basado en sensores, contempla el diseño de placas electrónicas para el Módulo Comparador y para el Módulo Sensor, con criterios compatibilidad electromagnética para cada dispositivo y componente dentro del sistema. Estos criterios están regulados por normas que deben cumplir los equipos y dispositivos mencionadas en [10]; estas normas serán tomadas en cuenta al momento de diseñar las placas de adquisición de datos para el diseño del sistema de parqueo de la Universidad de Cuenca.

El procesamiento de la información se realizará en una Raspberry Pi 3 (RPi), mientras que los espacios disponibles son visualizados por medio de módulos LED de alto brillo instalados en el suelo y un Panel LED. La comunicación entre todos los módulos del sistema general se realiza por medio de Protocolo Message Queue Telemetry Transport (MQTT), en donde la red es generada por un router dedicado para el proyecto. En la Figura 1.1 se puede observar la composición del sistema general, en donde los sistemas trabajan de forma independiente permitiendo realizar un análisis comparativo, diferenciando costos y eficiencia de cada uno, y a su vez complementario, con el objetivo de aumentar la fiabilidad del sistema.

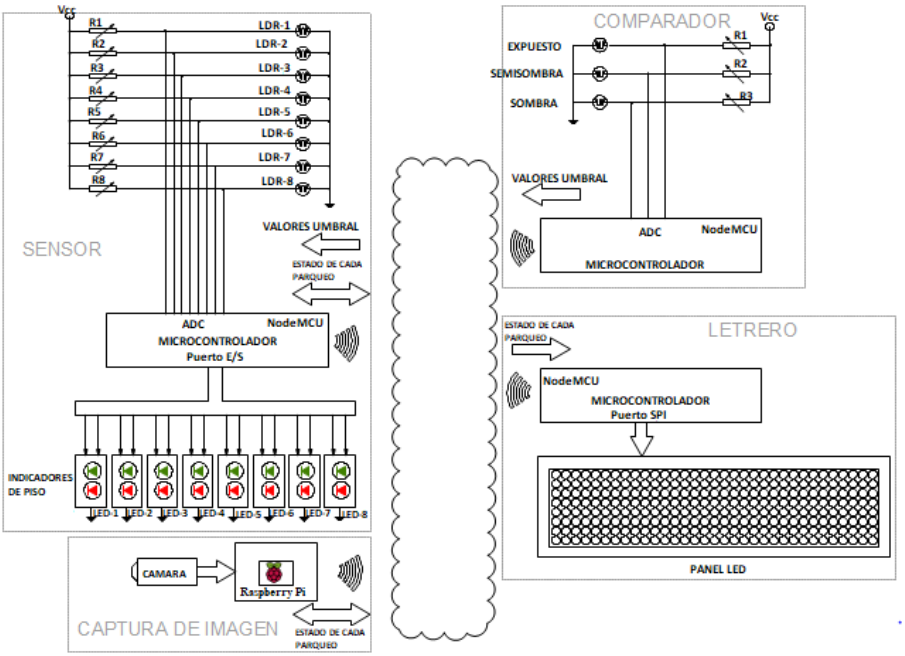


Figura 1.1: Esquema General del Sistema

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar e implementar un sistema de parqueo inteligente dentro de la Universidad de Cuenca combinando un subsistema compuesto de sensores LDR y la placa de desarrollo wemos NodeMCU v3 y otro subsistema basado en visión por computador.

1.4.2. Objetivos específicos

- Revisar el estado del arte y trabajos relacionados.
- Diseñar e Implementar placas electrónicas basadas en compatibilidad electromagnética
- Configurar la placa de desarrollo wemos NodeMCU v3 y procesar los datos recogidos por los sensores.
- Desarrollar el Algoritmo de Procesamiento de Imágenes, usando BS y TA
- Realizar un análisis comparativo entre los dos sistemas, diferenciando costos y eficiencia de cada uno.
- Integrar el funcionamiento de los dos sistemas, en donde el sistema de procesamiento de imágenes tendrá protagonismo en caso que el sistema de sensores falle, aumentando la eficiencia del sistema general.

CAPÍTULO 2

Marco teórico

2.1. Redes de Sensores Inalámbricos (WSN)

El aumento de avances tecnológicos en los campos donde los sistemas necesitan ser monitorizados y controlados continuamente, ha permitido el desarrollo de nodos de sensores que integran capacidades de detección, procesamiento y comunicación, asequibles y con bajo consumo de potencia. Estas propiedades facilitan la implementaciones de nodos en áreas extensas, formando una Red Inalámbrica de Sensores (WSN).[3]

A continuación se muestran algunas características principales de las WSN [11][12]:

- **Escalabilidad**: Según la tipología y algoritmos, se puede desplegar redes de miles de nodos y sensores.
- **Cooperación de Nodos Sensores**: Antes de transmitir la información, se realiza un reconocimiento de nodos vecinos.
- **Comunicación**: Debido al despliegue amplio de los nodos sensores, la comunicación *multi-hop* (salto múltiple de uno a otro) consigue un menor consumo de potencia.

En la Figura 2.1 se puede observar los elementos básicos de una WSN, los cuales son [11]:

- **Sensores**: Miden magnitudes físicas del medio y las transforman en señales eléctricas.
- **Nodo Sensor**: Son procesadores, los cuales convierten las señales eléctricas a datos y las envían como información a la estación base.
- **Gateway**: Son nodos que actúan como interfaz o puente para la comunicación con más WSNs

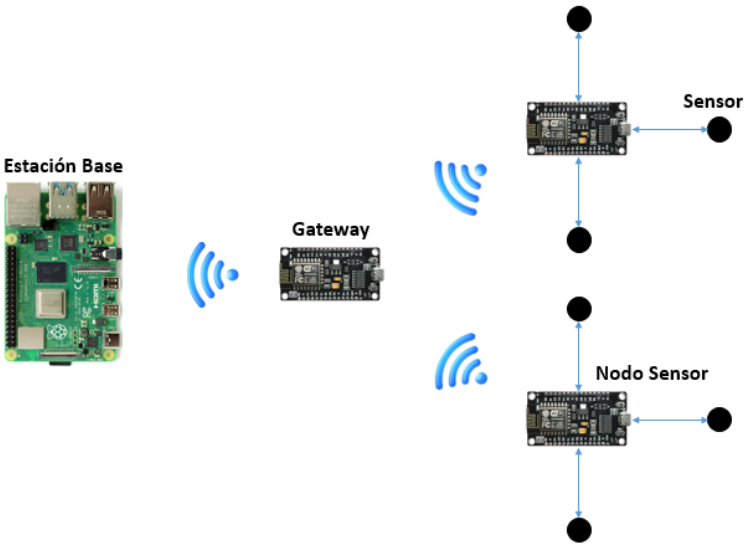


Figura 2.1: Elementos de una WSN básica

2.2. Protocolo MQTT

MQTT, es un protocolo usado para la comunicación Machine to Machine (M2M) en Internet of Things (IoT), el cual está orientado a la comunicación de sensores, debido a que consume muy poco ancho de banda y puede ser utilizado en la mayoría de los dispositivos embebidos con pocos recursos.

La arquitectura de MQTT sigue una topología de estrella, como se puede ver en la Figura 2.2, con un nodo central que hace de servidor o *Broker* con una capacidad de hasta 10000 clientes.

El *Broker* es el encargado de gestionar la red y de transmitir los mensajes, para mantener activo el canal, los clientes mandan periódicamente un paquete (PINGREQ) y esperan la respuesta del Broker (PINGRESP). La comunicación puede ser cifrada entre otras muchas opciones .[13]

La comunicación se basa en *Topics*, que el cliente que publica el mensaje crea, y los nodos que deseen recibirlo deben suscribirse a él. La comunicación puede ser de uno a uno, o de uno a muchos. Un *Topic* se representa mediante una cadena y tiene una estructura jerárquica. Cada jerarquía se separa con '/'.

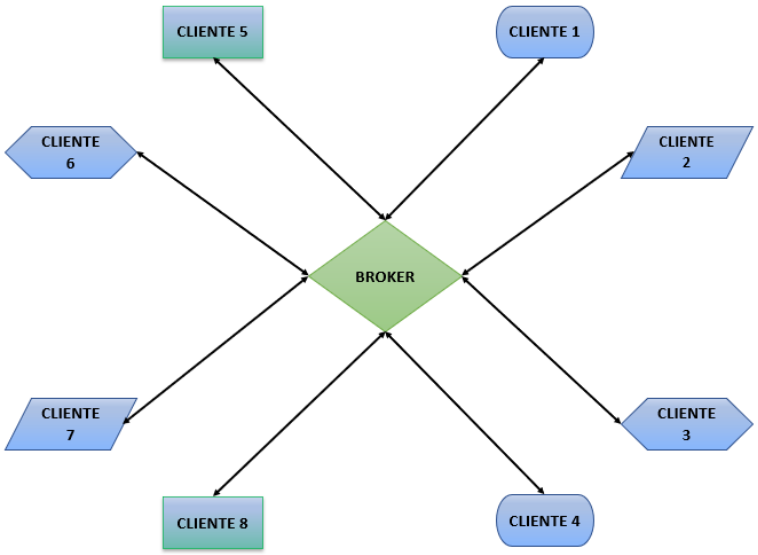


Figura 2.2: Arquitectura centralizada en el Broker

Se puede acceder a un *topic* usando su dirección, como por ejemplo: *edificio1/planta5/sala1/raspberry2/temperatura* o */edificio3/planta0/sala3/arduino4/ruido*. De esta forma se pueden crear jerarquías de clientes que publican y reciben datos, como se puede observar en la Figura 2.3. De esta forma un nodo puede suscribirse a un *Topic* concreto (edificio1/planta2/sala0/arduino0/temperatura) o a varios (edificio1/planta2/#).

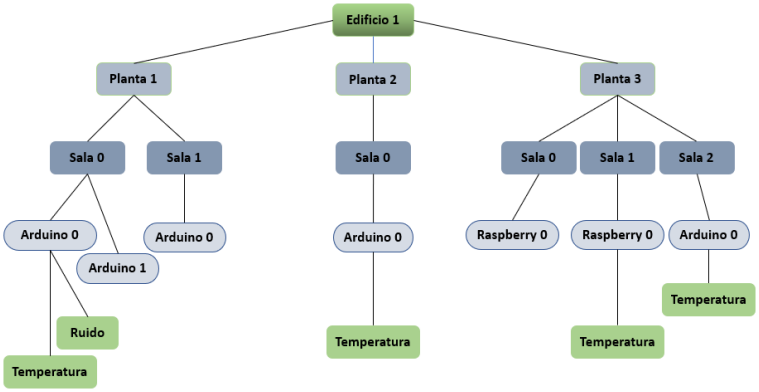


Figura 2.3: Jerarquías entre Topics.

Uno de los componentes más importantes del protocolo MQTT es la definición y topología de los mensajes, ya que son una de las bases de la agilidad en la que radica su fortaleza. Como muestra la Figura 2.4, cada mensaje consta de 3 partes.[14]

- **Cabecera fija**. Ocupa 2 a 5 bytes, obligatorio. Consta de un código de control, que identifica el tipo de mensaje enviado, y de la longitud del mensaje. La longitud se codifica en 1 a 4 bytes, de los cuales se emplean los 7 primeros bits, y el último es un bit de continuidad.

- **Cabecera variable**. Opcional, contiene información adicional que es necesaria en ciertos mensajes o situaciones.

- **Contenido o Payload**. Es el contenido real del mensaje. Puede tener un máximo de 256 Mb aunque en implementaciones reales el máximo es de 2 a 4 kB.

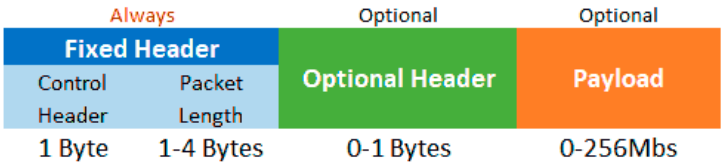


Figura 2.4: Composición de un mensaje MQTT [14]

La seguridad siempre debe ser un factor importante a considerar en cualquier sistema de comunicación M2M. El protocolo MQTT dispone de distintas medidas de seguridad que se puede adoptar para proteger las comunicaciones.

Esto incluye seguridad Secure Sockets Layer (SSL)/Transport Layer Security (TLS), que indica a los visitantes que pueden transmitir información confidencial de manera segura hacia y desde el servidor y autenticación por usuario y contraseña o mediante certificado. Sin embargo, hay que tener en cuenta que muchos de los dispositivos IoT disponen de escasa capacidad, por lo que SSL/TLS puede suponer una carga de proceso importante.

MQTT aporta una serie de características como su sencillez y ligereza. Esto lo hace adecuado para aplicaciones IoT, donde frecuentemente se emplean dispositivos de escasa potencia, conjuntamente, requiere un ancho de banda mínimo, lo cual es importante en redes inalámbricas, o conexiones con posibles problemas de calidad.

2.2.1. Cliente/Broker

Para filtrar los mensajes que son enviados a cada cliente los mensajes se disponen en *Topics* organizados jerárquicamente. Un cliente puede publicar un mensaje en un determinado *Topic*. Otros clientes pueden suscribirse a este *Topic*, y el *Broker* le hará llegar los mensajes.

Los clientes inician una conexión Transmission Control Protocol (TCP)/Internet Protocol (IP) con el *Broker*, el cual mantiene un registro de los clientes conectados. Esta conexión se mantiene abierta hasta que el cliente la finaliza. Por defecto, MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS.

Para ello el cliente envía un mensaje *CONNECT* que contiene información necesaria (nombre de usuario, contraseña, ID del Cliente). El *Broker* responde con un mensaje *CONNACK*, que contiene el resultado de la conexión (aceptada, rechazada, etc), como se puede observar en la Figura 2.5.

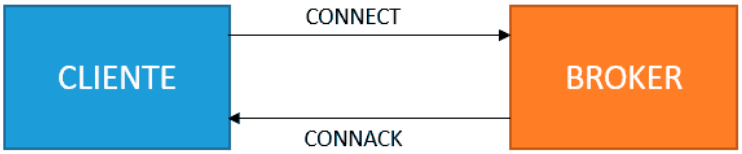


Figura 2.5: Establecimiento de Comunicación Cliente/Broker [14]

Según la Figura 2.6 para enviar los mensajes el cliente emplea mensajes *PUBLISH*, que contienen el *Topic* y el *payload*.

Figura 2.6: Publicación Cliente/Broker [14]

Según la Figura 2.7, para suscribirse y de-suscribirse se emplean mensajes SUBSCRIBE y UNSUSCRIBE, que el servidor responde con SUBACK y UNSUBACK.

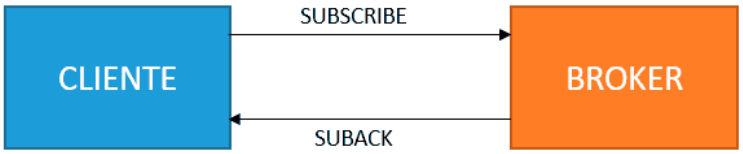


Figura 2.7: Suscripción Cliente/Broker [14]

Por otro lado, para asegurar que la conexión está activa los clientes mandan periódicamente un mensaje PINGREQ que es respondido por el servidor con un PINGRESP. Finalmente, el cliente se desconecta enviando un mensaje de *DISCONNECT*.[14]

2.2.2. Broker

Una de las claves principales de la comunicación MQTT es el Broker, que es el que se encarga de receptar los mensajes enviados por los clientes y distribuirlos entre sí en un sistema publicación -suscripción, como se puede observar en la 2.8

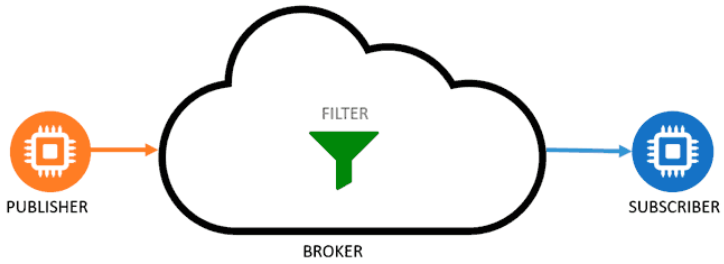


Figura 2.8: Conexión Cliente/Broker [14]

Existe una gran cantidad de *Brokers MQTT* disponibles cada uno con sus características, ventajas e inconvenientes. Elegir el *Broker* más adecuado para nuestro proyecto condiciona el buen funcionamiento y éxito del mismo. Entre algunos de los principales MQTT disponibles se tiene.

- **Mosquitto**. Es un *Broker Open Source* desarrollado por la fundación Eclipse y distribuido bajo licencia EPL/EDL. Está programado en C, y es multiplataforma. Es un *Broker* liviano y adecuado para uso en servidores de baja potencia.

- **Mosca**. Es un Broker MQTT Open Source para Node.js, desarrollado en Javascript por Matteo Collina. Puede ser empleado como aplicación independiente o embebido en cualquier proyecto de Node.js.

- **Aedes**. Del mismo autor que Mosca, Aedes es un servidor Broker MQTT Open Source para Node.js diseñado para ser un reemplazo de Mosca.

- **EMQTT**. *Erlan MQTT Broker* es *Open Source*, desarrollado en Erlang/OTP, está diseñado para aplicaciones con grandes exigencias en escalabilidad.

- **RabbitMQ**. Es un popular Broker de mensajería AMQP Open Source, que también permite emplear el protocolo MQTT a través de un Adaptador.

- **Moquette**. Un Broker MQTT Open Source escrito en Java desarrollado por Eclipse, que destaca por bajo peso.

2.2.3. Servidor - Eclipse Mosquito

Es un servidor de mensajes de código abierto con licencia Eclipse Public License (EPL)/Eclipse Distribution License (EDL) que implementa las versiones 5.0, 3.1.1 y 3.1 del protocolo MQTT. Esto lo hace adecuado para el IoT de mensajería como con sensores de baja potencia o dispositivos móviles como teléfonos, ordenadores embebidos o microcontroladores, como se ve en la Figura 2.9.

Mosquitto es un *Broker MQTT Open Source* ampliamente utilizado debido a su ligereza lo que nos permite, fácilmente, emplearlo en gran número de ambientes, incluso si éstos son de pocos recursos.[15]

Figura 2.9: Mosquito Broker RPi [15]

2.3. NodeMCU

Node Micro Controller Unit (NodeMCU) es un entorno de código abierto, construido como un System On a Chip (SoC), ESP8266, con el objetivo de realizar desarrollo conjunto de hardware y software. El módulo ESP8266 fue desarrollado por Espressif Systems, el cual consta de CPU, RAM, módulo Wi-Fi, sistema operativo y Software Development Kit (SDK). El modelo Lolin NodeMCU V3, tiene un tamaño de 58mm × 32mm, lo que lo convierte en un elemento versátil y robusto para trabajos de IoT de todo tipo.

En la tabla 2.1 se pueden observar las especificaciones técnicas, en donde se detalla el convertidor y tipo de conector Universal Serial Bus (USB), junto con los protocolos de comunicaciones soportados como Universal Asynchronous Receiver Transmitter (UART), Serial Peripheral Interface (SPI), Inter Integrated Circuits (I2C), el estándar Wireless Fidelity (WiFi) soportado, entre otros, y en la Figura 2.10 se puede observar la distribución de pines de la tarjeta de desarrollo.[16]

Tabla 2.1: Especificaciones Técnicas - Lolin NodeMCU V3

Microcontrolador	ESP-8266 32 bits
Modelo NodeMCU	Clonar Lolin
Tamaño	58 mm x 32 mm
Espaciado entre Pines	27,94mm
Velocidad de Reloj	80 MHz
USB a Serie	CH340G
Conector USB	Micro USB
Tensión de Funcionamiento	3,3 V
Voltaje de Entrada	4,5V - 10V
Memoria Flash/SRAM	4 MB / 64 KB
Pines E/S Digitales	11
Pines E/Analógica	1
Rango ADC	0 - 3,3V
UART / SPI / I2C	1 / 1 / 1
WiFi	802.11 b/g/n
Rango de Temperatura	-40C - 125C

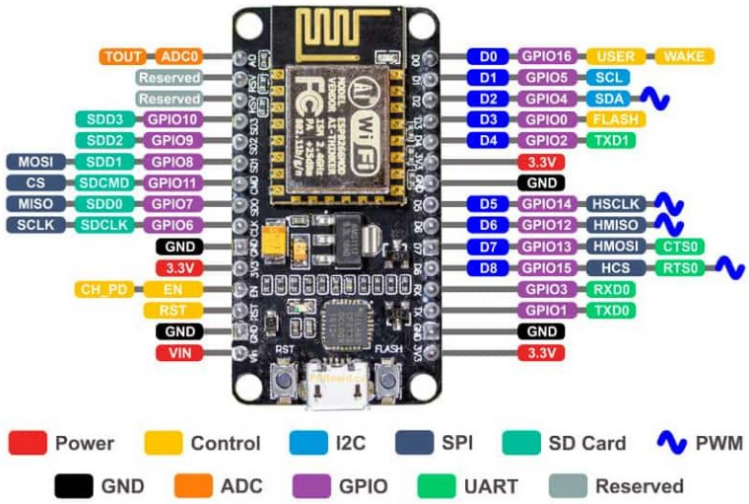


Figura 2.10: NodeMCU Pinout [16]

2.4. Light Dependent Resistor (LDR)

Una Resistencia Dependiente de Luz o LDR, consiste en una célula de Sulfuro de Cadmio, encapsulada con una resina transparente, resistente a la humedad, donde su resistencia aumenta cuando el nivel de luminosidad disminuye.

En la Figura 2.11 se puede observar la composición de un LDR, mientras que en la Figura 2.12 se puede observar la resistencia de un sensor LDR a diferentes niveles de luz. Un Lux equivale a un lumen/m^2, es decir, cien lúmenes, concentrados sobre un metro cuadrado, iluminan esa superficie con cien lux, sin embargo, los mismos cien lúmenes, distribuidos sobre diez metros cuadrados, producen una luminancia de sólo 10 lux en esa área.

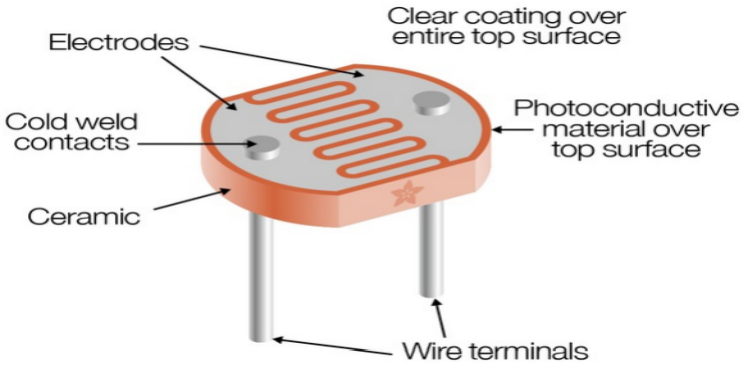


Figura 2.11: LDR [17]

Debido a factores como construcción, tamaño, iluminación, entre otros, las curvas observadas pueden variar. Generalmente, los LDR de Sulfuro de Cadmio comunes, en particular, tienden a ser sensibles a la luz entre 700nm (rojo) y 500nm (verde), teniendo una respuesta espectral es similar a la del ojo humano. [17]

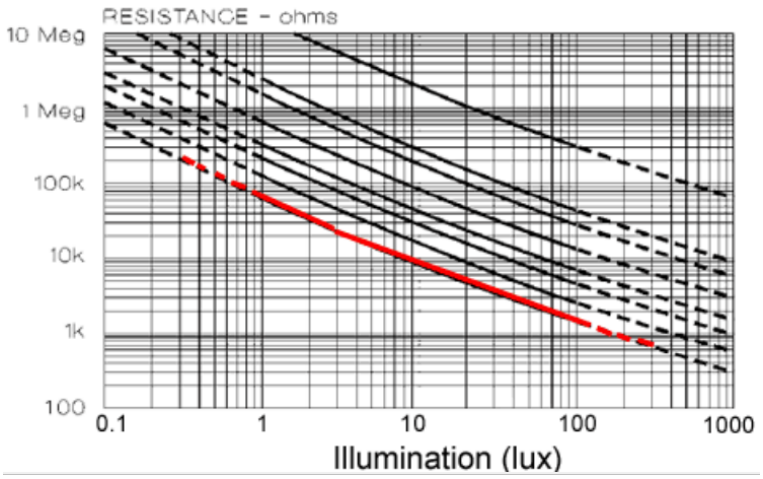
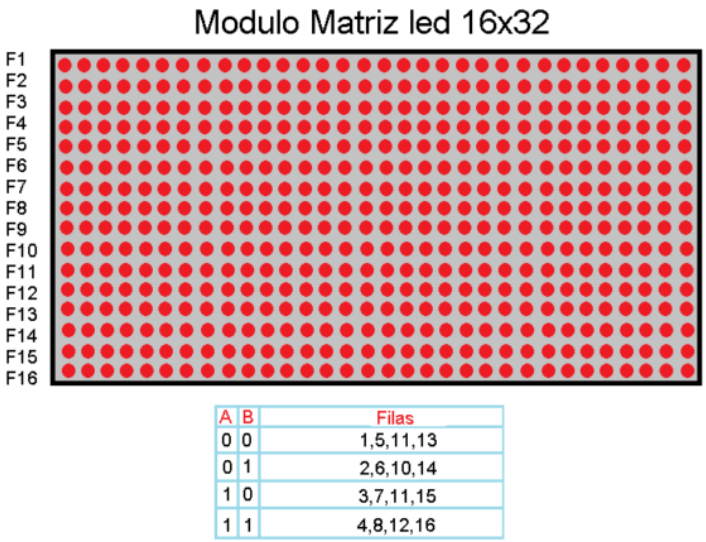


Figura 2.12: Comportamiento Resistencia vs. Iluminación [17]

2.5. Panel LED P10

Un panel LED estándar es una matriz un conjunto de 512 diodos LED conformada por 16 filas y 32 columnas de estos, como se puede observar en la Figura 2.13 . La facilidad de instalación, resistencia a ambientes interiores y exteriores y la poca demanda de energía hace que un panel LED pueda ser usado en cualquier tipo de sector donde se necesite un medio de visualización, además de su estructura modular que permite apilar varios de estos módulos obteniendo un panel de mayor dimensión.[18]



A	B	Filas
0	0	1,5,11,13
0	1	2,6,10,14
1	0	3,7,11,15
1	1	4,8,12,16

Figura 2.13: Panel LED P10 estándar [18]

Las principales características son:

- Diodos LED tipo Dual in Line Package (DiP) mejoran la dispersión lumínica.

- Voltaje de alimentación 5V
- Resistencia al Agua
- Arreglo de LEDs de 16X32 píxeles, separados a 10mm.
- Puerto de comunicación HUB12

El puerto HUB12 permite el intercambio de información de manera que la entrada y salida de información sea en serie; como se puede observar en la Figura 2.14 los pines A y B selecciona las filas que se desean controlar, el pin CLK pertenece al reloj, el pin SCLK sirve para escribir los datos en los registros, los datos ingresan por medio del pin R y el pin OE controla el brillo de los LEDs.

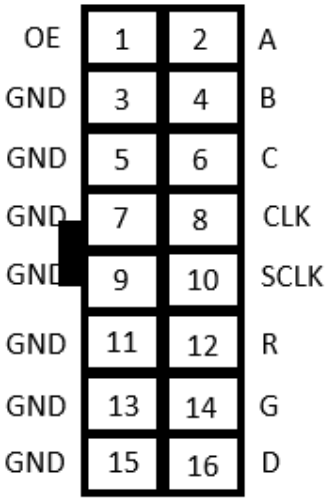


Figura 2.14: Puerto HUB12

El control del panel se lo realiza por medio del protocolo SPI, por lo que está constituido por 16 registros de desplazamiento 74HC595 en donde cada registro controla 8 LEDs diferentes. Adicionalmente cuentan con un multiplexor 74HC138 que se encarga de activar las filas correspondientes al momento de mostrar la información, mientras que los transistores mosfet de canal P controlan la corriente de los LEDs.[18]

2.6. Circuitos Integrados

Un circuito integrado, es una estructura de pequeñas dimensiones formado de transistores, diodos, resistencias, condensadores, entre otros, interconectados en una pastilla de silicio; el cual realiza una función en específico.

2.6.1. Multiplexor CD4051BE

Funciona como un interruptor analógico de 8 canales controlado por estados lógicos de 3 bits; según la combinación de los 3 bits se puede seleccionar un canal para conectarlo con la salida, en la Figura 2.15 se puede observar la distribución de los pines para el Multiplexor CD4051B y a su vez en la tabla 2.2 se puede observar la

descripción de cada uno de los pines.[19]

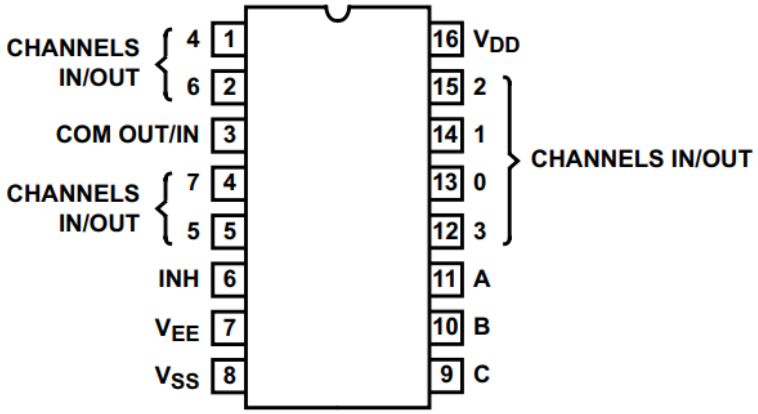


Figura 2.15: Distribución de pines del multiplexor CD4051B [19]

Tabla 2.2: Función de Cada Pin - CD4051B

N° Pin	Nombre	Función	Descripción
1	CH4	E/S	Canal 4 Entrada/Salida
2	CH6	E/S	Canal 6 Entrada/Salida
3	COM	E/S	Entrada/Salida COMÚN
4	CH7	E/S	Canal 7 Entrada/Salida
5	CH5	E/S	Canal 5 Entrada/Salida
6	INH	E	Habilita o Deshabilita todos los Canales
7	VEE	-	Entrada de Voltaje Negativo
8	VSS	-	Tierra
9	C	E	Selector de Canal (Bit Mas Importante)
10	B	E	Selector de Canal
11	A	E	Selector de Canal (Bit Menos Importante)
12	CH3	E/S	Canal 3 Entrada/Salida
13	CH0	E/S	Canal 0 Entrada/Salida
14	CH1	E/S	Canal 1 Entrada/Salida
15	CH2	E/S	Canal 2 Entrada/Salida
16	VDD	-	Entrada de Voltaje Positivo

El control de las señales analógicas entrantes depende de la alimentación de VDD, VEE y VSS, siendo 20v la tensión de alimentación máxima, tanto para VDD como para cada canal. Por ejemplo, si VDD = +4.5V, VSS = 0V o Tierra, y VEE = –13.5V, las señales analógicas entrantes, entre –13.5V a +4.5V se pueden controlar mediante las entradas digitales de 0V a 5V. Ahora bien, si VDD = +3.3V, VSS = 0V o Tierra, y VEE = 0V, entonces VDD–VSS = VDD – VEE = 3.3V, por lo que es posible controlar señales analógicas entrantes de hasta 3.3V mediante las entradas digitales de 0V a 3.3V.

En la Figura 2.16 se puede observar el comportamiento de la Resistance On Channel (Ron) conforme varía el Voltage Input Signal (Vis), de manera que, mientras menor es la diferencia VDD - VEE la Ron es mayor.

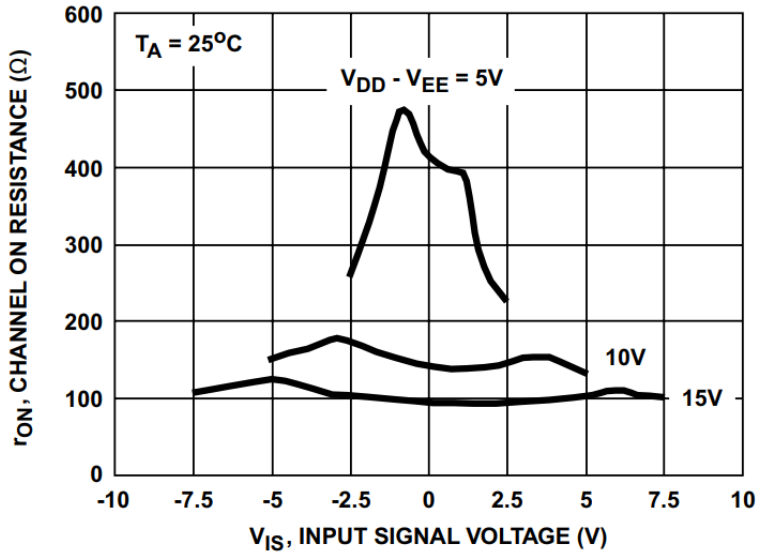


Figura 2.16: Comportamiento Resistencia de Canal vs. Voltaje de Entrada de la Señal [19]

Este multiplexor disipa una cantidad extremadamente baja de energía en reposo sobre todos los rangos de suministro de voltaje VDD–VSS y VDD–VEE, independientemente del estado lógico de las señales de control. Cuando el Pin INH tiene estado lógico "1", todos los canales son desactivados. A continuación, se detallan algunas características del multiplexor: [19]

- Rango de 3V a 20V de Vis Digital.
- Rango ≤ 20V Peak to Peak Voltage (Vpp) de Vis Analógica.
- Conversión de Señales digitales entre 3V a 20V (VDD-VSS = 3V a 20V) a señales analógicas ≤ 20V Vpp (VDD-VEE = 20V).
- En reposo, la disipación de energía es muy baja en todas las condiciones de suministro y entrada de control digital, 0.2 µW (típico) para VDD-VSS = VDD-VEE = 10V.
- La conmutación Break-Before-Make (BBM) elimina la superposición entre canales.

2.6.2. Driver UNL2803

El Circuito Integrado ULN2803A es una matriz de 8 pares transistores Darlington de 50V y 500mA, lo que le permite que sus salidas soporten de alto voltaje, como se puede observar en la Figura 2.17. En la salida cuenta con un circuito de sujeción por diodos en cátodo común para conmutar cargas inductivas y proteger el circuito de sobretensión y Electro Static Discharge (ESD) .

El rango de corriente del colector de cada par Darlington es hasta 500 mA. Los pares Darlington se pueden conectar en paralelo para una mayor capacidad de corriente.[20]

El dispositivo ULN2803A tiene una resistencia de base de 2.7KΩ para cada par Darlington para operar directamente con dispositivos Transistor-Transistor Logic (TTL) o Complementary Metal Oxide Semiconductor (CMOS).

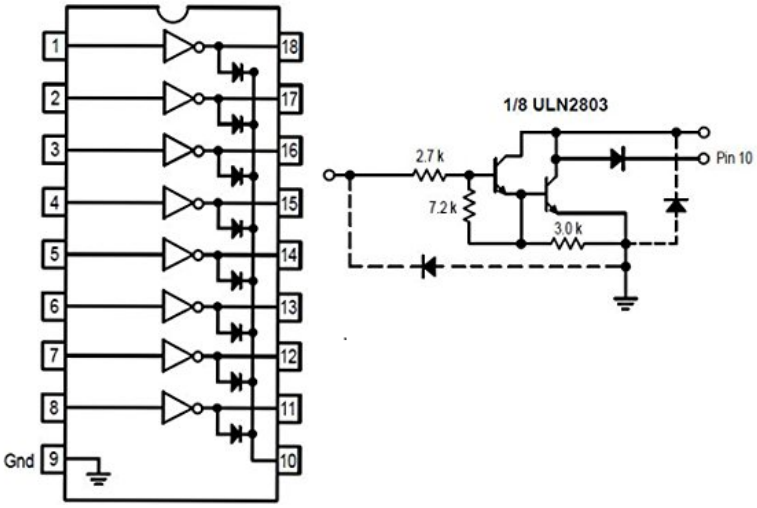


Figura 2.17: Circuito Integrado UNL2803 [20]

2.6.3. Registro 74LS595

El circuito integrado 74LS595 consta de un registro de desplazamiento de 8 bits y un circuito *Latch* tipo D de 8 bits con salidas paralelas de tres estados, en la Figura 2.18 se puede observar la distribución de pines, junto con el diagrama lógico compuesto del registro de desplazamiento y el circuito *Latch* tipo D, mientras que en la tabla 2.3 se describe la función de cada uno de los pines.[21]

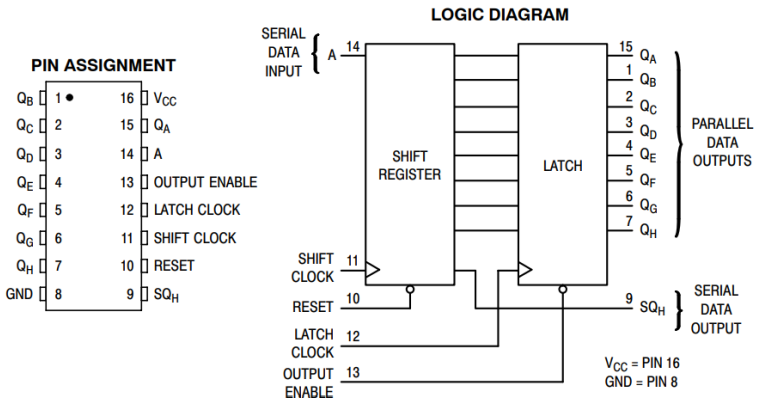


Figura 2.18: Circuito Integrado 74LS595 [21]

El registro de desplazamiento y el circuito Latch tienen entradas de reloj independientes. Este dispositivo también tiene un reinicio asíncrono para el registro de desplazamiento.

Entre las características más importantes del circuito integrado se tienen las siguientes: [21]

- Salidas de interfaz directa a CMOS, Negative-channel Metal-Oxide Semiconductor (NMOS) y TTL.
- Rango de voltaje de funcionamiento: 2,0 a 6,0 V.
- Corriente de entrada baja: 1.0 μA.

Tabla 2.3: Función de Cada Pin - 74LS595

Nombre	Descripción
A (Pin 14)	Entrada de datos en serie.
Shift Clock (Pin 11)	Entrada de reloj del registro de desplazamiento.
Reset (Pin 10)	Al Activar en Bajo restablece la parte del registro de desplazamiento sin afectar el circuito Latch.
Latch Clock (Pin 12)	Entrada de reloj del circuito Latch.
Output Enable (Pin 13)	Al Activarlo en Bajo hace que los datos de en el circuito Latch se presenten en las salidas. Al Activarlo en Alto, fuerza a las salidas (QA - QH) al estado de alta impedancia.
OUTPUTS QA - QH (Pines 15; 1-7)	Salidas no invertibles de 3 estados.
SQH (Pin 9)	Salida de datos en serie no invertible, permite el acoplamiento con otro registro de desplazamiento de 8 bits.

- Alta inmunidad al ruido de los dispositivos CMOS.
- Alto Rendimiento a ESD: Human-Body Model (HBM) >2000V; Machine Model (MM) >200V. [22][23].
- Estos son dispositivos libres de Plomo.

2.7. Raspberry Pi 3

El RPi es una placa de desarrollo tipo SoC, como se puede ver en la Figura 2.19, es considerado un minicomputador diseñado para realizar trabajos de desarrollo de software y hardware en entornos con espacios reducidos y limitaciones económicas, permitiendo procesar grandes cantidades de información de manera remota.

Cuenta con procesador de cuatro núcleos de 64 bits que funciona a 1.4GHz, Local Area Network (LAN) inalámbrica de banda dual de 2.4GHz y 5GHz, Bluetooth 4.2/Bluetooth Low Energy (BLE), conector Ethernet y capacidad Power over Ethernet (PoE) Hardware Attached on TOP (HAT) lo que permite energizar el equipo por medio de redes habilitadas para PoE, sin embargo, la red a la que está conectado el equipo debe tener su propio suministro de energía.[24]

Adicionalmente cuenta con puertos Mobile Industry Processor Interface (MIPI) Display Serial Interface (DSI) para conectar un módulo *display* y puertos MIPI Camera Serial Interface (CSI) para conectar una mini cámara, conjuntamente con puertos de Audio y video, dos puertos USB y un puerto High Definition Multimedia Interface (HDMI), los que le permiten conectar dispositivos como auriculares, pantalla, teclado, mouse, entre otros, para controlar el RPi mediante la interfaz gráfica provista por el Sistema Operativo Raspbian.

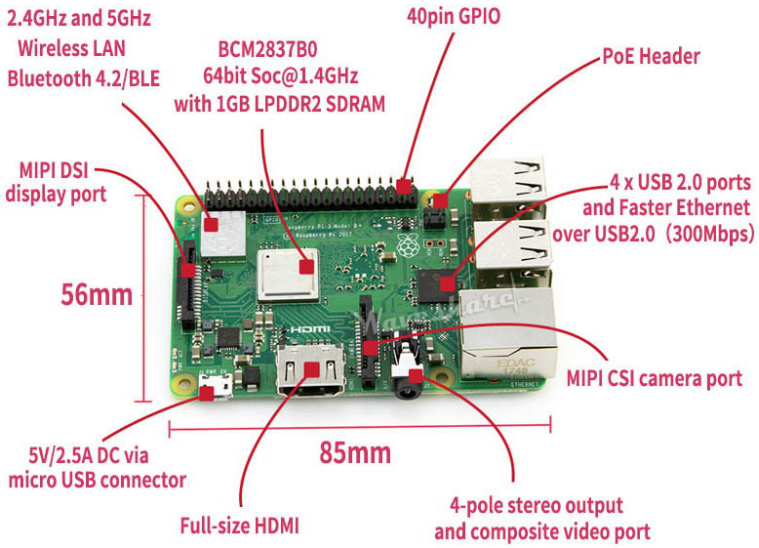


Figura 2.19: Raspberry Pi [24]

2.8. Cámara Web

La cámara web *Genius Slim 1300* es una cámara diseñada con un sensor de alta calidad de 1.3 megapíxeles, permitiendo obtener imágenes de 640X480 con un ángulo de visión de 90^o. Utiliza tecnología de conexión *plug & play* que facilita su uso directo sin necesidad de instalar ningún tipo de software para su funcionamiento. La Figura 2.20 muestra la cubierta de la cámara, que le brinda de protección ante golpes y una estructura que le permite ser instalada en cualquier lugar. [25]

Figura 2.20: Cámara Web Genius Slim 1300 [25]

2.9. Estándar IP (Ingress Protection)

Debido a la susceptibilidad de los circuitos electrónicos a diferentes tipos de daños, se han implementado normas que se utilizan para definir la efectividad de protecciones de equipos electrónicos por medio de códigos alfanuméricos, estas normas son conocidas como estándar IP (*Ingress Protection*).

El código alfanumérico está compuesto por las siglas IP seguidas por dos dígitos, como se puede observar en la Figura 2.21, en donde, el primer dígito describe el nivel de protección frente a sólidos, mientras que el segundo dígito describe el nivel de protección frente a líquidos.

En las tablas 2.4 y 2.5 se puede observar las descripciones correspondientes a cada uno de los valores otorgados a los dígitos, de esta forma se tiene una idea del nivel de protección de un circuito electrónico.[26]

Figura 2.21: Estándar IP [26]

Tabla 2.4: Descripción Primer Dígito

Nivel	Efectivo contra
0	Sin Protección
1	Protegido contra elementos con tamaño superior a 50mm
2	Protegido contra elementos con tamaño superior a 12mm
3	Protegido contra elementos con tamaño superior a 2.5mm
4	Protegido contra elementos con tamaño superior a 1mm
5	Protegido parcialmente contra la penetración de polvo
6	Protegido totalmente contra la penetración de polvo

Tabla 2.5: Descripción Segundo Dígito

Nivel	Efectivo contra
0	Sin Protección
1	Protección contra caída vertical de gotas de agua
2	Protección contra caída de gotas de agua a una inclinación máxima de 15^o
3	Protección contra caída de lluvia fina a una inclinación máxima de 60^o
4	Protección parcialmente contra chorros de agua desde cualquier ángulo
5	Protección totalmente contra chorros de agua desde cualquier ángulo
6	Protección contra fuertes chorros de agua desde cualquier ángulo
7	Protección contra inmersión completa a 1 metro durante 30 minutos
8	Protección contra inmersión completa a profundidad y duración que especifique el fabricante

2.10. Compatibilidad Electromagnética

La Electromagnetic Compatibility (EMC) es una rama de las ciencias eléctricas que se dedica a estudiar, eliminar y prevenir los efectos en un circuito eléctrico o electrónico producidos por la existencia de energía electromagnética, basándose en normas y regulaciones para asegurar la confiabilidad y eficiencia de los sistemas electrónicos bajo un ambiente en específico.

En la Figura 2.22 se puede observar la necesidad de lidiar con módulos o elementos electrónicos que son fuentes de ElectroMagnetic Interference (EMI)s, los cuales afectan a demás componentes, disminuyendo la confiabilidad del sistema. [27][28]

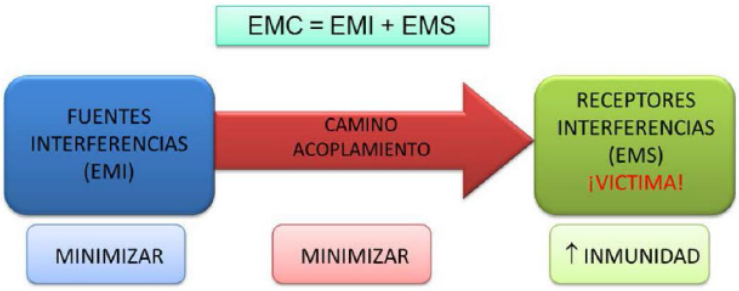


Figura 2.22: Elementos de problemas en EMC [27]

A continuación se detallan algunas consideraciones de diseño:

1. **Minimizar EMI:**
   - *a*) Elegir componentes electrónicos adecuados.
   - *b*) Definir adecuadamente la topología del circuito.
   - *c*) Proteger los contactos.
2. **Maximizar inmunidad del receptor:**
   - *a*) Apantallado del circuito.
   - *b*) Definir topologías de zonas según su funcionamiento.
3. **Minimizar el camino de acoplamiento del ruido:**
   - *a*) Distribución adecuada de alimentaciones.
   - *b*) Evitar antenas y lazos de masa.
   - *c*) Distribución adecuada de pistas.

Debido a que las principales fuentes de interferencia son los campos eléctricos, magnéticos y electromagnéticos generados por las antenas creadas por los caminos entre componentes electrónicos, el diseño inicial de una Printed Circuit Board (PCB) debe relacionarse íntimamente con técnicas de EMC para garantizar que el producto pueda formar parte de una sistema donde se encuentren diferentes módulos electrónicos sin sufrir interferencias ni producirlas en los demás módulos, como se puede observar en la Figura 2.23, recursos como tiempo y dinero se ven afectados al momento de diseñar un circuito electrónico .[27] [28] [10]

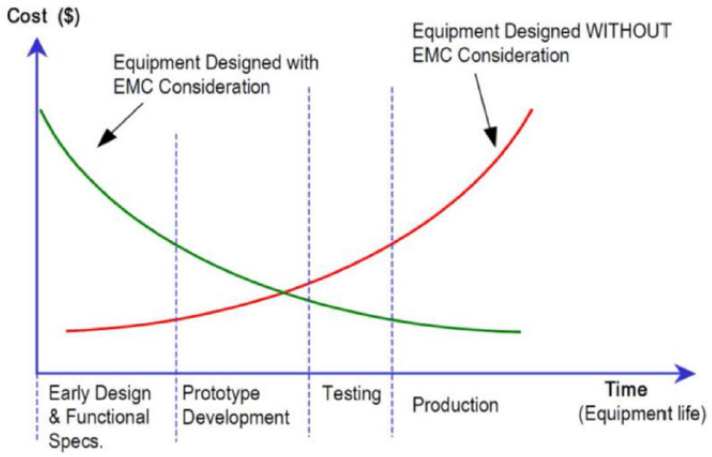


Figura 2.23: Costes de Desarrollo de un Circuito Electrónico [27]

2.11. Algoritmos de procesamiento de imágenes

El sistema visual de un ser vivo tiene la capacidad de detectar y analizar una gran cantidad de imágenes en tiempo real con un gran poder de procesamiento. Para recrear el procesamiento del sistema visual humano se utilizan un conjunto de técnicas aplicadas a imágenes con el objetivo de aislar las características de interés, facilitando la búsqueda de información en las imágenes; algunas de las técnicas se describen a continuación.

2.11.1. Operaciones morfológicas

La morfología matemática consiste en un conjunto de técnicas matemáticas que permiten la manipulación de formas en una imagen. El lenguaje utilizado en las operaciones morfológicas proviene de la teoría de conjuntos, de manera que en una imagen binaria cada conjunto representa la forma de un objeto. Así, si el píxel es blanco su valor es 1 y si el píxel es negro su valor es 0 .[29] [30]

De esta forma, cada elemento de un conjunto A o B, es un píxel o punto de coordenadas (x,y) en el plano bidimensional de la imagen. El proceso de transformaciones morfológicas necesita de dos entradas, la primera es la imagen original, y la segunda es el Structuring Element (SE) o kernel que decide la naturaleza de la operación. El SE se desplaza sobre la imagen situando su centro en cada píxel de la imagen original, aplicando una regla definida por la operación morfológica sobre los píxeles situados bajo el SE. De manera, que el valor de cada píxel de la imagen de salida se basa en una comparación del píxel correspondiente en la imagen de entrada con sus vecinos.

Las operaciones morfológicas más comunes son la dilatación y la erosión. La dilatación aumenta la cantidad de píxeles blancos del conjunto u objeto, mientras que la erosión aumenta la cantidad de píxeles negros en base al kernel utilizado. A partir de estas operaciones se tienen otras variantes como Apertura y Cierre.

- **Dilatación**. Hace que los objetos sean más visibles, rellenando los pequeños agujeros dentro de los objetos

y aumentando el contorno de los mismos, lo cual es útil para recuperar el tamaño inicial de un objeto o para unir partes rotas de mismo. La respuesta del proceso de dilatación es el conjunto de píxeles barridos por el centro del SE mientras algún punto del SE coincide con alguno de la imagen. Es decir, un píxel resultante es "1"si al menos un píxel de la imagen que cae dentro del SE o kernel es "1".[29]

El proceso de dilatación sigue la siguiente función matemática, donde A es una imagen y B es el SE, de manera que A_b es la traslación de A por b, lo que también se puede entender como el desplazamiento de todos los píxeles de B sobre A. La imagen resultante se verá afectada por la forma del SE o kernel, en la Figura 2.24 se puede observar el proceso de dilatación en una imagen binaria, donde los píxeles de color beige son los que se han añadido formando un nuevo objeto más grande.[30]

$$A \oplus B = \bigcup_{b \in B} A_b \tag{2.1}$$

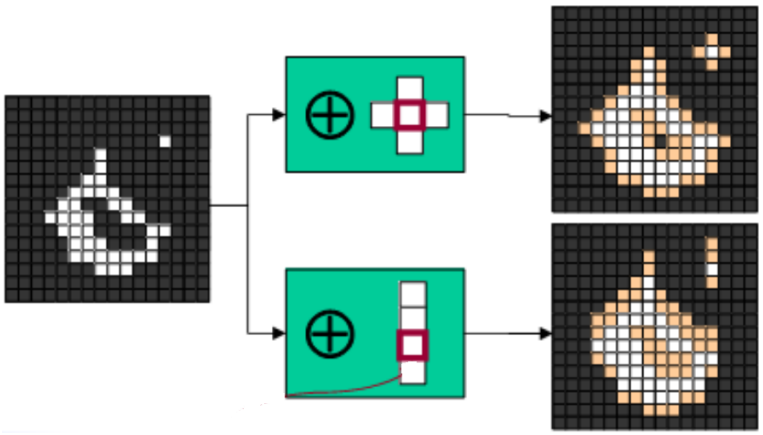


Figura 2.24: Proceso de dilatación de un objeto en una imagen binaria [30]

- **Erosión**. Elimina la presencia de ruido en la imagen y contrae los bordes de los objetos en una imagen, permitiendo visualizar de una mejor manera la forma o esqueleto de cada objeto. La respuesta del proceso de erosión es el conjunto de píxeles barridos por el centro del SE siempre que se cumpla, que todos los puntos del SE estaban contenidos en la imagen. Es decir, un píxel resultante será "1"si todos los píxeles que caen dentro del kernel son "1", de lo contrario su valor será "0".[29]

  El proceso de erosión sigue la siguiente función matemática, donde A es una imagen y B es el SE, de manera que A_{-b} es la traslación de A por -b, lo que también se puede entender como el desplazamiento del centro de B sobre cada píxel de A. La imagen resultante se verá afectada por la forma del SE o kernel, en la Figura 2.25 se puede observar el proceso de erosión en una imagen binaria, donde los píxeles de color rojo son los que se han eliminado y los de color blanco representan al nuevo objeto.[30]

$$A \ominus B = \bigcap_{b \in B} A_{-b} \tag{2.2}$$

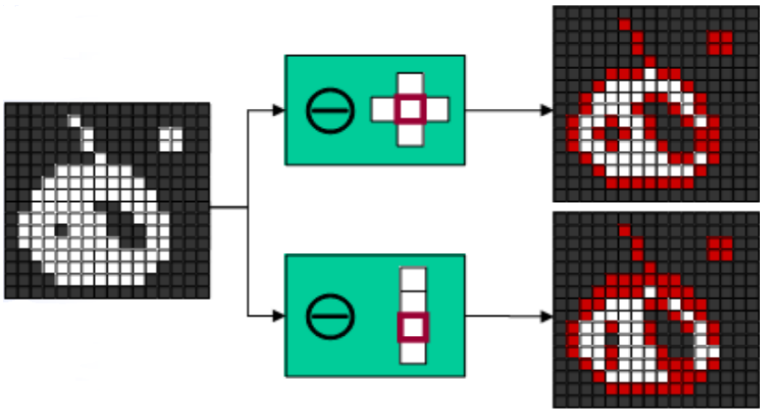


Figura 2.25: Proceso de Erosión de un Objeto en una imagen binaria [30]

- **Apertura**, Elimina la presencia de ruido y de objetos pequeños en la imagen mientras conserva la forma y el tamaño de los objetos más grandes de la imagen. La operación morfológica de apertura, primero erosiona una imagen eliminando todo tipo de ruido y luego la dilata utilizando el mismo SE para ambas operaciones.

  La imagen resultante se verá afectada por la forma del SE o kernel, en la Figura 2.26 se puede observar el proceso de apertura en una imagen binaria, donde los píxeles de color rojo son los que se han eliminado, los de color beige son los que se han añadido manteniendo el tamaño del objeto original.[30]

$$A \circ B = (A \ominus B) \oplus B \tag{2.3}$$

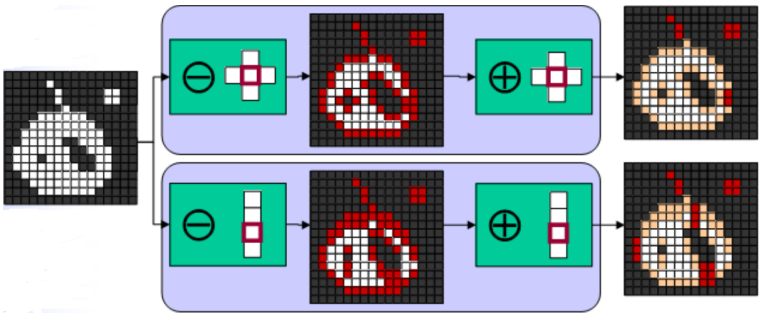


Figura 2.26: Proceso de Apertura en una imagen binaria [30]

- **Cierre**, Rellena los pequeños agujeros presentes dentro de los objetos en la imagen mientras conserva su forma y tamaño. La operación morfológica de cierre, primero dilata una imagen cerrando todos los espacios vacíos dentro de un objeto y uniendo los objetos pequeños en una imagen, y luego la erosiona utilizando el mismo SE para ambas operaciones.

  La imagen resultante se verá afectada por la forma del SE o kernel, en la Figura 2.27 se puede observar el proceso de cierre en una imagen binaria, donde los píxeles de color beige son los que se han añadido para eliminar los agujeros dentro del objeto y los píxeles de color rojo son los que se han eliminado manteniendo el tamaño del objeto original.[30]

$$A \bullet B = (A \oplus B) \ominus B \tag{2.4}$$

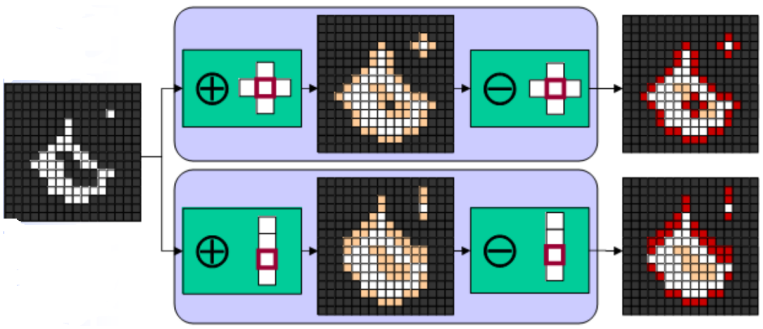


Figura 2.27: Proceso de Cierre en una imagen binaria [30]

2.11.2. Background subtraction

La técnica BS es ampliamente utilizada en la detección de objetos en movimiento de una escena, como se puede observar en la Figura 2.28, la técnica genera una imagen binaria que contiene los píxeles que pertenecen a los objetos en movimiento. BS calcula la máscara de primer plano o imagen binara por medio de una resta entre el fotograma actual y un modelo de fondo, el cual contiene generalmente la parte estática de la escena .[31]

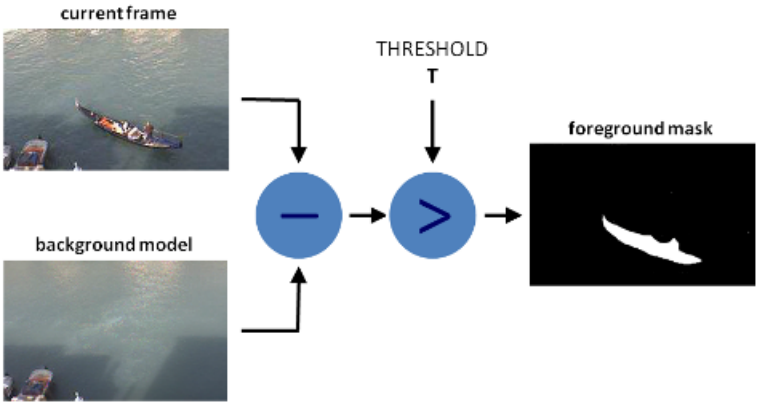


Figura 2.28: Funcionamiento de Background Subtraction [31]

El proceso de BS se realiza de manera que, primero se calcula un modelo inicial del fondo, y después, actualiza ese modelo para adaptarse a posibles cambios en la escena.

2.11.3. Mixture of gaussians

Dentro del procesamiento de imágenes para la detección de objetos por medio de BS, se tiene la técnica MoG, que es un algoritmo introducido por P. Kaew TraKul Pong y R. Bowden en 2001 [32], el cual modela cada píxel de fondo mediante una mezcla de K distribuciones gaussianas (K= 3 a 5).

La probabilidad de que un píxel tenga un valor X_N en el tiempo N es escrita como:

$$p(X_N) = \sum_{j=1}^{k}(w_j)\eta(X_N;\theta_j) \tag{2.5}$$

En donde w_k es el peso del k-ésimo componente Gaussiano y $\eta * (X_N;\theta_j)$ es la distribución normal del k-ésimo componente representada por [32]:

$$\eta(X;\theta_k) = \frac{1}{(2\pi)^{\frac{D}{2}}|\sigma_k^2|^{\frac{1}{2}}} e^{-\frac{(X-\mu_k)^T(X-\mu_k)}{2\sigma_k^2}} \tag{2.6}$$

De esa manera la probabilidad de un píxel se define como:

$$p(X_N) = \sum_{j=1}^{k}\frac{(w_j)}{(2\pi)^{\frac{D}{2}}|\sigma_j^2|^{\frac{1}{2}}} e^{-\frac{(X_N-\mu_j)^T(X_N-\mu_j)}{2\sigma_j^2}} \tag{2.7}$$

Así, las K distribuciones están ordenadas en base a la al valor otorgado por $\frac{w_k}{\sigma_k}$, en donde las primeras B distribuciones son usadas como un modelo de fondo del escenario, siendo B estimado de la siguiente manera:

$$B = arg\min_b(\sum_{j=1}^{b} W_j > T) \tag{2.8}$$

El umbral T es la probabilidad previa mínima de que el fondo esté en la escena. BS se realiza marcando un píxel de primer plano que esté a más de 2,5 desviaciones estándar de cualquiera de las distribuciones B.

De esta manera, los pesos de la mezcla representan las proporciones de tiempo que esos colores permanecen en la escena, así, los probables colores de fondo son los que permanecen más tiempo y más estáticos.[33] [32]

Sin embargo, si el primer valor de un píxel dado es un objeto en primer plano, solo existirá un gaussiano donde su peso es igual a la unidad. Por lo tanto, se necesitará un tiempo de adaptación hasta que el fondo genuino pueda considerarse como un fondo, incluyendo que la situación puede empeorar en entornos concurridos donde un fondo limpio es raro. Para tener un análisis de un entorno en específico se utilizan algoritmos complementarios para identificar identificar un fondo genuino, junto con la limitación de ROIs.[32]

2.12. Conclusiones

El uso del protocolo MQTT dentro de las WSN permite tener un sistema de sensores robusto y fácilmente escalable. De una manera complementaria, el uso de sensores de bajo consumo como LDRs junto con módulos programables como NodeMCU y RPi permiten la existencia de equilibrio entre el consumo de energía y el procesamiento de la información recolectada.

El uso de normas y criterios de EMC es un factor primordial durante el diseño y desarrollo de PCBs, entre los cuales se encuentra la selección adecuada de dispositivos robustos basados en los modelos HBM y MM, que ayudan a minimizar el factor de riesgo ante ESD y EMI , como los mencionados en la Sección 2.6.

Adicionalmente, el uso de una cámara web permite obtener una visión adecuada de un área de interés, prescindiendo del uso de demás módulos para digitalizar la imagen, y gracias a su bajo consumo se reduce el gasto energético. Dado que la implementación del procesamiento de imágenes se la realiza en un computador, el uso del algoritmo adaptativo mencionado en la Sección 2.11, permite obtener resultados eficientes limitando su consumo computacional, los cuales, junto con el trabajo complementario de la WSN, da como resultado un sistema robusto para la detección de plazas disponibles dentro de un estacionamiento.

Finalmente, debido a que el sistema opera de manera expuesta, se hace uso del estándar IP mencionado en la Sección 2.9 para el desarrollo de protecciones que le permitirán al sistema en general ser impermeable y resistente a polvo, garantizando su funcionamiento de manera continua.

CAPÍTULO 3

Estado del Arte

En el capítulo 2 se ha realizado una introducción teórica, enfocada en las características y funcionamiento, de cada sección del sistema compuesto WSN y Visión Artificial.

En este capítulo se presentan algunos trabajos relacionados, referentes a la tecnología usada en la implementación de WSN conjuntamente con algoritmos de PDI, enfocados a la detección de espacios de estacionamiento. El objetivo principal es analizar el rendimiento, cobertura y escalabilidad en base a los recursos utilizados, los cuales servirán de guía para el proceso de implementación del sistema planteado en el capítulo 1.

3.1. Trabajos Relacionados con WSN y Visión Artificial

Existen varias alternativas para solucionar el problema de acceso a parqueaderos de estacionamiento, siendo los sistemas de visión artificial los más escogidos debido a la facilidad de instalación, sin embargo, representan una inversión elevada de precios, consumo energético y procesamiento computacional.

En [2] se realiza un conteo del número de vehículos estacionados y se identifican los puestos disponibles. El sistema define una imagen de un automóvil como imagen de referencia, las imágenes capturadas de los demás parqueaderos se comparan secuencialmente con las imágenes de referencia. El procesamiento de las imágenes se desarrolla utilizando el algoritmo de detección de bordes Prewitt y, de acuerdo con la información coincidente, se muestra la información de espacios disponibles al conductor.

En [6] se implementa un control de espacios de parqueo en un estacionamiento al aire libre dentro de un campus universitario, usando CNN, en donde el sistema es entrenado con una secuencia de 3000 imágenes recopiladas, logrando una precisión de detección superior al 99 %. Se puede visualizar la ubicación exacta y la cantidad de espacios de estacionamiento vacíos dentro de un estacionamiento en tiempo real, usando una aplicación celular.

En [8] se realiza un sistema para detección en ambientes abiertos, el cual utiliza un histograma de gradientes orientados para alimentar una SVM en la fase de entrenamiento. Para incrementar la precisión de la detección se

realiza una exclusión de los instantes donde ocurre movimiento, se elimina el fondo restringiendo el análisis a una ROI y se elimina de la detección objetos extraños como peatones bloqueando esta región de interés. Con estas consideraciones se obtiene una precisión entre 91.1 % hasta 98.8 % dependiendo de las condiciones ambientales.

En [34] se utiliza un sistema basado en las características de los vehículos. Se aplica una clusterización de dichas características para ubicar los espacios de parqueadero. Para clasificar los espacios ocupados de los vacíos se analiza la variabilidad de las escalas de grises, en un espacio vacío los grises varían poco en comparación con la variación de los espacios ocupados, esto debido a que los espacios vacíos solo consisten de asfalto o pavimento.

En [35] se implementa un sistema de gestión de estacionamiento mediante las imágenes obtenidas en tiempo real por cuatro cámaras ubicadas en diferentes lugares alrededor del mismo. Se realiza el preprocesamiento en todas las imágenes de entrada, diferenciando los autos del asfalto por medio de la gestión del color. Luego, se genera un modelo de fondo de estacionamiento adaptable usando BS. El color adecuado de cada plaza de aparcamiento se averigua mediante el método estadístico en secuencias de imágenes en color capturadas por una cámara, y se extrae el primer plano en función de la información de color. El resultado se modificará aún más mediante la detección de sombras basada en el análisis de luminancia. El sistema puede gestionar grandes áreas en base a la cantidad de cámaras y ajuste de posición.

En [36] se describe un sistema que utiliza BS, por medio del modelo MoG. El fondo se procesa utilizando el descriptor Histogram of Oriented Gradients (HOG), el modelo de detección de esquinas Scale-Invariant Feature Transform (SIFT) y los espacios de color *HSV, YUV, YCbCr* para resolver varios problemas de detección. Estos algoritmos combinados mejoran la precisión de la detección de ocupación del estacionamiento lo que ha dado como resultado una alta tasa de precisión durante un largo período de tiempo. Los resultados con una tasa de precisión más baja son ocasionados por la sombra y el camuflaje, ya que la sombra se identifica como un objeto. En un período de tiempo prolongado la tasa de precisión es superior al 93 %.

En relación con implementaciones usando módulos de sensores, en [37] se implementa un sistema de detección de parqueos libres usando Radio Frequency Identification (RFID) para permitir el control de acceso de vehículos y módulo Global System for Mobile communications (GSM) para informar al usuario sobre la disponibilidad de espacios dentro de los estacionamientos. Sin embargo, se llega a la conclusión que el magnetómetro es el sensor ideal para la detección de parqueaderos disponibles, aunque su precio por unidad es elevado; adicionalmente, se plantea una propuesta de gestión de iluminación, que pretende un ahorro significativo de presupuesto.

En [38] se expone un sistema de estacionamiento inteligente integrado en la nube basado en IoT, el cual consiste en una implementación de sensores Passive Infrared (PIR) y sensores ultrasónicos que están conectados de forma inalámbrica a una RPi mediante el chip ESP8266, con el fin de determinar si un espacio de estacionamiento está vacío o no. La visualización de los estacionamientos vacíos se la realiza a través de luminarias y por medio de una aplicación móvil que permite al usuario final verificar la disponibilidad de espacio de estacionamiento y reservar un espacio de estacionamiento en consecuencia.

En [5] se realiza una comparación completa entre el LDR que funciona en el principio de detección de

sombras y el sensor PIR que funciona en el mecanismo de detección de objetos. Se presenta el análisis de desempeño de la precisión para la detección de espacios de estacionamiento vacantes y detección de vehículos en diferentes condiciones. Se concluye que la precisión del sensor LDR se ve muy afectada por el cambio de intensidad luminosa a lo largo del día, por lo que es necesario calcular diferentes valores de umbral sobre la base del cambio en la intensidad luminosa, lo que aumenta la complejidad del código, a diferencia del sensor PIR que tiene la capacidad de identificar diferentes tipos de vehículos sin ser afectado por una condición ambiental, sin embargo, su consumo excesivo de energía lo hace menos rentable en comparación con el sensor LDR.

3.2. Conclusiones

En las investigaciones realizadas sobre métodos de implementación de WSN, descritas en [37], [38] y [5], se usan diferentes tipos de sensores. Mediante un análisis de resultados se obtiene que el magnetómetro es el sensor ideal para la detección de espacios de estacionamiento, sin embargo, es un módulo que requiere costos elevados de instalación y mantenimiento; de forma similar, con el uso del sensor PIR se obtienen resultados eficientes al ser un módulo diseñado para detección de objetos, pero su consumo energético elevado disminuye su rentabilidad. Finalmente, el sensor LDR es un dispositivo pequeño, económico y resistente, que requiere una inversión de instalación y mantenimiento baja, aunque es afectado por el cambio de luminosidad, siendo necesario calcular diferentes valores de umbral, la complejidad de código puede ser solventada gracias a las capacidades de procesamiento de una RPi. De esta forma, el sensor LDR se convierte en un dispositivo fiable y rentable para ser usado en sistemas de detección de parqueaderos disponibles, tanto al aire libre como en ambientes cerrados.

En lo que concierne a los sistemas basados en PDI, en [6] se utiliza una CNN entrenada, logrando alcanzar una precisión del 99 % [7]. En [8] se utiliza un histograma de gradientes orientados para alimentar una SVM en la fase de entrenamiento. Se incrementa la precisión de la detección mediante la definición de ROIs, obteniendo una precisión entre 91.1 % hasta 98.8 %. Aunque los resultados de aplicaciones con CNN y SVM son satisfactorios, se requiere un dispositivo dedicado al procesamiento de imágenes, debido a la demanda computacional. En [2] se define una imagen de un automóvil como imagen de referencia y se la compara con las imágenes capturadas de los demás parqueaderos, el procesamiento se basa en el algoritmo de detección de bordes Prewitt, aunque permite obtener resultados eficientes sin ser ocupar demasiado espacio computacional, necesita más de una cámara para cumplir su función, limitando su escalabilidad. Finalmente, en [9] se combinan las técnicas de BS utilizando MoG para detectar y rastrear vehículos, y análisis de transitoriedad TA para detectar el estacionamiento y la salida de vehículos. Este sistema aborda los principales desafíos de los escenarios urbanos sin entrenamiento, lo que lo hace ideal para trabajar complementariamente con un sistema de WSN en una micro-computadora.

El presente documento proporciona un análisis detallado del comportamiento del sensor LDR en diferentes ambientes, por medio de curvas que representan varios escenarios. Conjuntamente se propone un sistema electrónico compuesto por sensores LDR, cuya deteccion es adaptativa en base a la luminosidad del ambiente. Se realiza la comparación entre el sistema de sensores y un sistema de procesamiento de imágenes basado en BS en base a la precisión de deteccion y costos de implementación, teniendo en cuenta las limitaciones mencionadas posteriormente. Finalmente, se realiza un análisis económico en el que se detalla el costo de implementación de cada uno de los módulos desarrollados y instalacion del prototipo del sistema general, junto con una proyección aproximada de costos para la implementación en todo el campus central de la Universidad de Cuenca.

CAPÍTULO 4

Diseño e Implementación

En el presente capitulo se detalla la implementación de prototipo de sistema autónomo conformado por Redes Inalámbricas de Sensores WSN y Procesamiento de Imágenes PDI, que permite detectar las plazas disponibles en un conjunto de ocho parqueaderos del estacionamiento de la Facultad de Ingeniería y visualizarlas por medio de módulos LED de alto brillo instalados en cada parqueo y un Panel LED para todo el sistema.

La adquisición, acondicionamiento y procesamiento de la información del sistema de sensores se la realiza en una Raspberry Pi, mientras que la comunicación entre los módulos del sistema se la realiza usando Wi-Fi, a través del protocolo MQTT.

En la Figura 1.1 se puede observar la composición del sistema general, en donde los sistemas trabajan de forma independiente permitiendo realizar un análisis comparativo, diferenciando costos y eficiencia de cada uno, y a su vez complementario, con el objetivo de aumentar la fiabilidad del sistema.

4.1. Sistema basado en WSN

El sistema basado en sensores consta de dos módulos alimentados desde la red eléctrica cuyo funcionamiento se describe en el diagrama de flujo de la Figura 4.1. El primer módulo (Módulo Comparador), descrito a detalle en la Sub Sección 4.1.1, está conformado por tres sensores LDR de referencia que sometidos a multiplexación con 8 resistencias de diferente valor óhmico con el objetivo de obtener el comportamiento de cada LDR en base a la iluminación existente en 3 escenarios diferentes. El segundo módulo (Módulo Sensor), descrito en la Sub Sección 4.1.2, consta de ocho sensores LDR que serán usados para detectar la presencia de un automóvil en ocho parqueaderos. Los datos obtenidos por los módulos son publicados en un tópico usando el protocolo MQTT y serán procesados en la RPi. En base a los datos recolectados por el módulo comparador el algoritmo de detección desarrollado en la Sub Sección 4.1.4 definirá el estado de cada parqueadero, lo cual será enviado al controlador del panel LED y al módulo sensor para visualización.

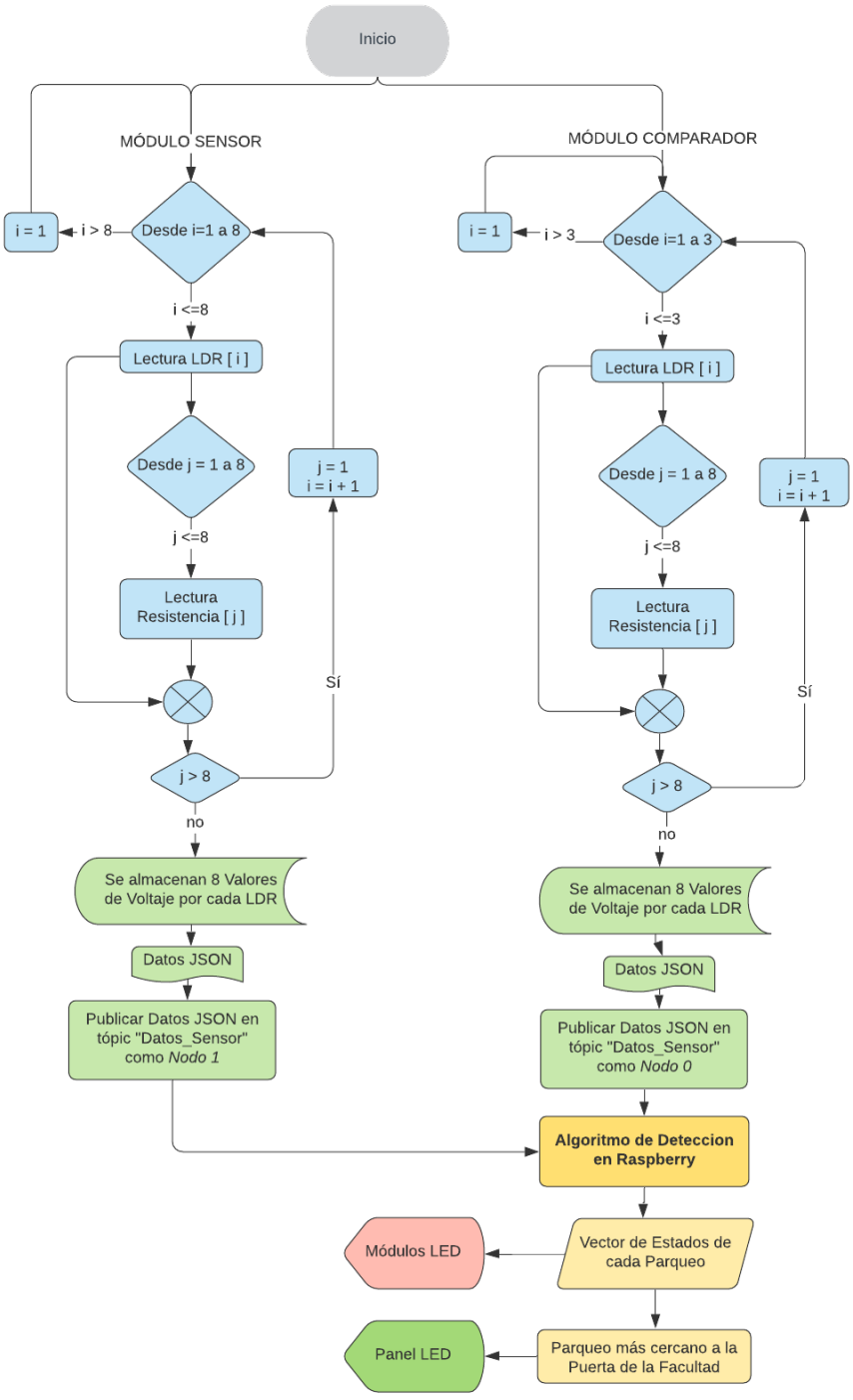


Figura 4.1: Diagrama de Flujo WSN

4.1.1. Módulo comparador

El módulo comparador está conformado por tres sensores LDR cuyos comportamientos servirán como referencias, ubicados, uno en luz directa, otro semi iluminado y otro en sombra, con el objetivo de simular los escenarios más importantes que se dan en la detección de parqueos disponibles, los cuales son: libre, ocupado y libre con sombra. También cuenta con un conjunto de 8 resistencias, las cuales sirven para obtener una curva de comportamiento de cada sensor LDR en base a la iluminación presente en el ambiente, el dimensionamiento de las resistencias según sus valores óhmicos se detalla en la siguiente Sub sección.

Por medio de un dispositivo NodeMCU se controlan dos multiplexores CD4051, cuyo funcionamiento se menciona en la Sección 2.6, los cuales permiten seleccionar el sensor LDR y la resistencia de interés. Primeramente, se selecciona al sensor LDR ubicado en luz directa y se multiplexan cada una de las ocho resistencias, de menor a mayor valor óhmico, obteniendo ocho valores diferentes de voltaje por medio del pin de entrada analógica (A0) del NodeMCU. Los valores de voltaje obtenidos se los almacena en un archivo JavaScript Object Notation (JSON), el cual es un estándar basado en texto plano para el intercambio de información, y son enviados a una RPi por medio del protocolo MQTT, con el objetivo que sirvan de referencia para el algoritmo de detección.

El mismo procedimiento se realiza con los sensores LDR ubicados en los escenarios semi iluminado y sombra.

Dimensionamiento de resistencias

El dimensionamiento de los valores óhmicos de las resistencias, servirán para asignar valores específicos a las 8 resistencias que pertenecerán al módulo comparador y que a su vez servirán para caracterizar la curva de comportamiento de cada sensor LDR de referencia. Para ello, se procede a armar un módulo sensor temporal que servirá como herramienta para obtener diferentes valores de voltaje por medio de un partidor de tensión usando un sensor LDR y una caja sustituta de resistencias, en diferentes horas del día y en diferentes escenarios. Por medio de la caja sustituta de resistencias, se establecen 40 valores de resistencias, desde los 33Ω hasta los 100KΩ.

Como se puede observar en la Figura 4.2, el módulo sensor temporal consta de:

- *Arduino Uno*, el cual servirá como interfaz con el computador, para la recolección y visualización de los datos obtenidos.
- *LDR*, el cual permitirá obtener una señal de voltaje conforme a la iluminación y variación de valores de resistencias.
- *Caja sustituta de resistencias*, es un dispositivo mecánico que permite variar a voluntad los valores de resistencias entre 1Ω y 10MΩ obteniendo una precisión del 1 %.[39]
- *Multiplexor CD4051BE*, debido a que la señal de voltaje que llega al pin analógico (A0) del NodeMCU pasa a través del multiplexor, esta es afectada por la resistencia interna del circuito integrado. En la Figura 2.16 se observa el comportamiento de la Ron del multiplexor según la Vis, de modo que al usar 3.3V como referencia para la señal de entrada y para la alimentación, se estima que la resistencia interna es cercana a 200Ω.

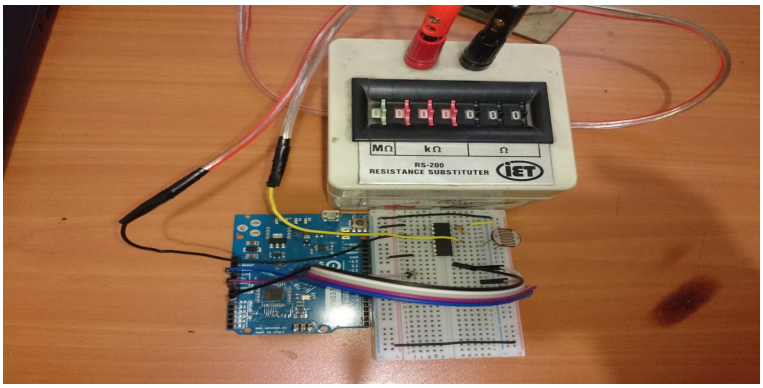


Figura 4.2: Módulo sensor temporal

Se procede a realizar el censado de la iluminación en tres ambientes diferentes, los cuales son:

- Soleado
- Nublado y lluvioso
- Penumbra y Noche

En cada uno de los ambientes especificados, se varían los escenarios, con el objetivo de obtener una mejor apreciación del comportamiento del LDR. En el primer ambiente, soleado, se realizan las mediciones en cinco escenarios diferentes tomando en cuenta la exposición del sensor LDR bajo luz directa y bajo diferentes tipos de sombra.

En la Figura 4.3 se observan cinco curvas de comportamiento diferentes, en donde, el eje de las abscisas (eje X) corresponde los 40 valores de resistencias entre 33Ω hasta los 100KΩ mientras que el eje de las ordenadas (eje Y) corresponde a los valores de voltaje obtenidos entre 0V y 3.3V; los valores que componen cada una de las curvas fueron definidos ubicando el sensor LDR en los siguientes escenarios:

- Debajo de una camioneta bajo sombra de un edificio, que simula un parqueadero ocupado.
- Debajo de una camioneta bajo luz directa de sol, que simula un parqueadero ocupado.
- Bajo luz directa de sol, que simula un parqueadero libre.
- Bajo sombra de un edificio, que simula un parqueadero libre, pero con sombra.
- Bajo sombra de un árbol, que simula un parqueadero libre, pero con sombra.

En los valores de voltaje obtenidos, se encuentra incluida la resistencia interna del multiplexor. Se puede observar que, en ambiente soleado, los valores pertenecientes a los parqueaderos libres se encuentran por debajo de los 2V y los valores pertenecientes a los parqueaderos ocupados se encuentran por encima de los 3V, existiendo así, una distancia mayor a 1V entre los dos posibles casos (libre u ocupado). Esto muestra que, para el ambiente soleado, se puede asignar cualquier valor de resistencia entre 33Ω hasta los 100KΩ a las 8 resistencias que se usaran en el prototipo del módulo comparador.

Además de las curvas de comportamiento de cada uno de los escenarios, se puede observar una curva de color negro, la cual es denominada ”UMBRAL” que ha sido definida con el propósito de tomar la decisión sobre el estado del parqueadero.

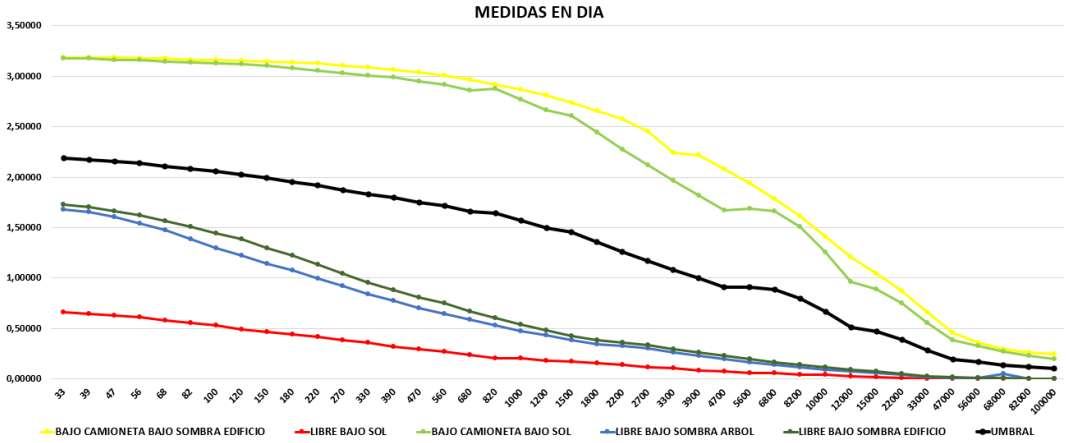


Figura 4.3: Curvas de Comportamiento del sensor en día Soleado

En el segundo ambiente, nublado, se realizan las mediciones en cuatro escenarios diferentes tomando en cuenta la exposición del sensor LDR bajo luz directa y bajo diferentes tipos de sombra. Como se puede observar en la Figura 4.4 las curvas de comportamiento fueron definidas en los siguientes escenarios:

- Debajo de una camioneta con sombra total bajo cielo nublado, que simula un parqueadero ocupado.
- Debajo de una camioneta con sombra parcial bajo cielo nublado, que simula un parqueadero ocupado.
- Debajo de un auto pequeño bajo cielo nublado, que simula un parqueadero ocupado.
- Bajo cielo nublado, que simula un parqueadero libre, pero con sombra.

Se nota que en un ambiente nublado, la diferencia entre zonas iluminadas y con sombra es muy tenue, por lo que los valores pertenecientes a los parqueaderos libres se encuentran por debajo de los 2.5V y los valores pertenecientes a los parqueaderos ocupados se encuentran por encima de los 2.8V, existiendo así, una pequeña diferencia no mayor a 1V entre los dos posibles casos (libre u ocupado).

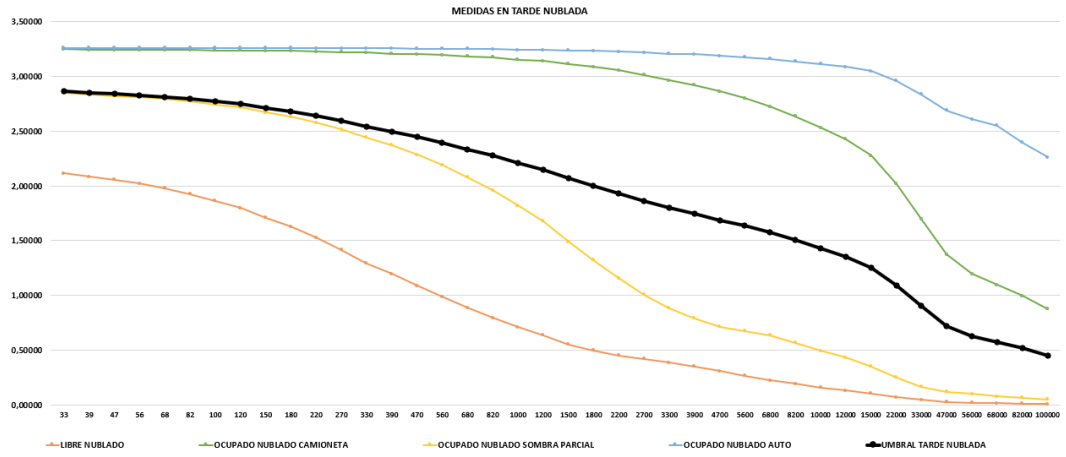


Figura 4.4: Curvas de Comportamiento del sensor en el día nublado y lluvioso

Se puede observar que para los 10 primeros valores de la curva de umbral, esta se encuentra muy cerca a la curva que describe el escenario con el sensor LDR debajo de una camioneta con sombra parcial, de color amarillo; esto se debe a que este escenario se lo ha definido como un suceso especial, en caso de existir un auto

que se encuentre mal estacionado ocupando la mitad de un parqueadero o menos y su sombra cubre cualquiera de los dos sensores adyacentes.

De manera que, a diferencia del primer ambiente soleado, en el que se puede escoger cualquier valor de resistencia entre 33Ω hasta los 100KΩ, en este ambiente nublado existe una mejor diferenciación de los diferentes escenarios para valores de resistencias superior a 1KΩ.

Finalmente, el tercer ambiente, penumbra y noche, es caracterizado por siete escenarios diferentes ubicando el sensor LDR bajo la luz directa de las luminarias del estacionamiento y bajo diferentes tipos de sombra. Como se puede observar en la Figura 4.5 las curvas de comportamiento fueron definidas en los siguientes escenarios:

- Debajo de un auto con sombra total, que simula un parqueadero ocupado.
- Debajo de un auto con sombra parcial, que simula un parqueadero ocupado.
- Bajo cielo con penumbra, que simula un parqueadero libre.
- Bajo luz directa de un poste de iluminación, que simula un parqueadero libre en la noche.
- Bajo luz directa entre postes de iluminación, que simula un parqueadero libre en la noche.
- Bajo sombra densa, que simula un parqueadero libre, pero con sombra.
- Bajo sombra parcial o semi densa, que simula un parqueadero libre, pero con sombra.

En este caso se puede evidenciar que debido a la poca luz ambiental los 14 valores iniciales en todos los escenarios se encuentran sobre los 3V, estando estrechamente relacionados y dificultando diferenciar el estado de los parqueos. Sin embargo, al aumentar la resistencia por encima de los 1KΩ, aumenta la distancia entre las curvas de comportamiento de cada escenario, lo que permite detectar de una mejor manera el estado de cada parqueadero.

Se pueden observar dos escenarios especiales, los cuales son: libre bajo sombra densa y libre bajo sombra parcial o semi densa. Estos casos se dan cuando un auto estacionado se encuentra alejado de poste de iluminación y a su otro lado está un parqueo vacío cubierto por la sombra del auto, en donde la sombra puede ser densa o semi densa, generando errores de detección en el sensor LDR del parqueo libre.

Para el escenario de un estacionamiento libre bajo sombra densa, todos los valores que definen la curva correspondiente a este escenario se ubican sobre la curva umbral, denotando que en este caso especial el parqueo vacío aparenta estar ocupado. Para el escenario de un estacionamiento libre bajo sombra parcial o semi densa, a partir de una resistencia de 33KΩ los valores de que definen la curva correspondiente a este escenario se ubican debajo la curva umbral.

De manera que, a diferencia del primer ambiente, soleado, en el que se puede escoger cualquier valor de resistencia entre 33Ω hasta los 100KΩ, y del segundo ambiente, nublado, en el que existe una mejor diferenciación entre los escenarios para valores de resistencias superior a 1KΩ, para el ambiente actual, penumbra y noche, se puede diferenciar de una manera fiable a partir de valores de resistencias superiores a 33KΩ.

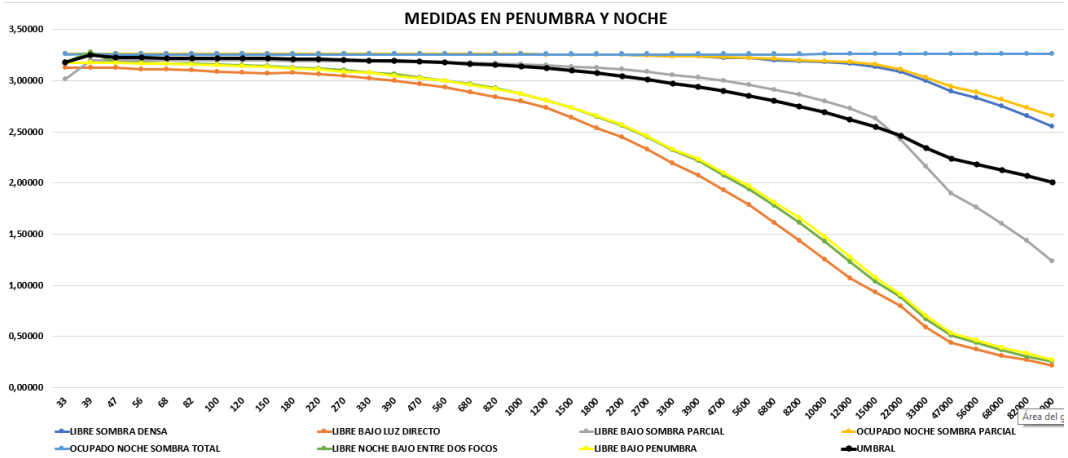


Figura 4.5: Curvas de Comportamiento del sensor en la noche

Debido a que el módulo comparador, consta de tres sensores LDR ubicados cada uno en un escenario diferente, los valores de umbral se los calcula por medio de un promedio de los valores obtenidos de cada sensor. El umbral de decisión se define en base al comportamiento de la iluminación en los ambientes de penumbra y noche, puesto que es cuando se tienen más escenarios que pueden ocasionar fallas en la detección. Inicialmente se realiza un promedio normal entre los valores obtenidos de cada sensor como se muestra en la ecuación 4.1, sin embargo, como se puede observar en la Figura 4.6 el escenario de un parqueo libre pero con sombra parcial es susceptible a ser detectado como ocupado, siendo posible tener una detección coherente para valores muy altos de resistencia.

$$V_{Umbral} = \frac{V_{ocupado} + V_{libreconsombra} + V_{libre}}{3} \tag{4.1}$$

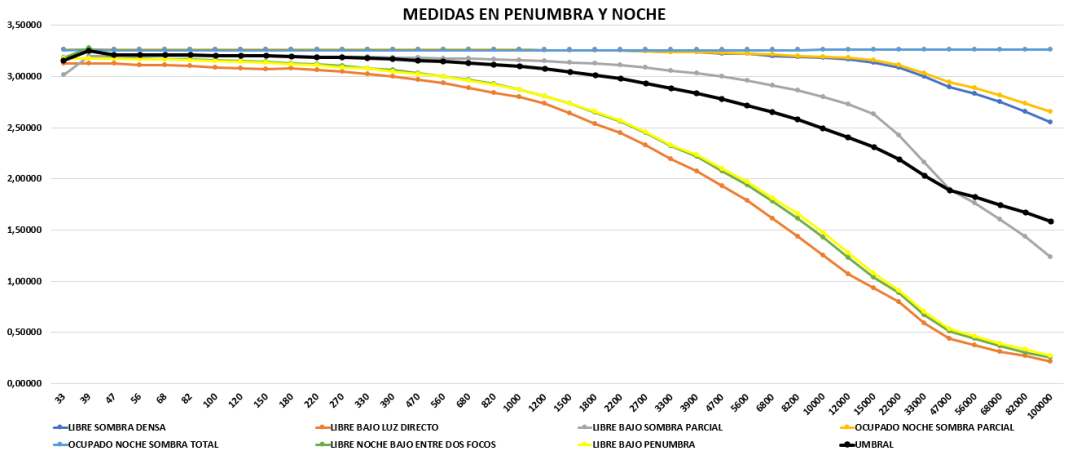


Figura 4.6: Umbral de Decisión con promedio normal

A partir de los resultados de la ecuación 4.1 se otorga un peso tripe para el sensor que simula un parqueo ocupado como se observa en la ecuación 4.2. El comportamiento del umbral calculado se lo evidencia en la Figura 4.7, el cual muestra inestabilidad en la detección del estado del parqueo en el escenario el escenario de un parqueo libre pero con sombra parcial.

$$V_{Umbral} = \frac{3 * V_{ocupado} + V_{libreconsombra} + V_{libre}}{5} \tag{4.2}$$

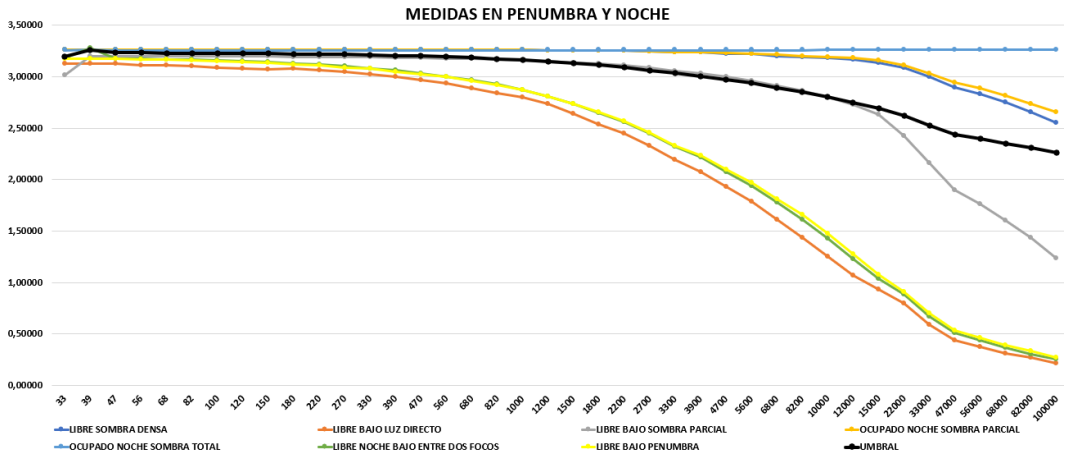


Figura 4.7: Umbral de Decisión con promedio con triple peso en ocupado

En base al comportamiento de los umbrales calculados por medio de las ecuaciones 4.1 y 4.2 se asigna solamente doble peso al sensor que simula un parqueo ocupado como se muestra en la ecuación 4.3, logrando una detección equilibrada en todos los escenarios evidenciada en las figuras 4.3. 4.4 y 4.5.

$$V_{Umbral} = \frac{2 * V_{ocupado} + V_{libreconsombra} + V_{libre}}{4} \tag{4.3}$$

La asignación de un doble peso al valor que simula el estado ocupado, se la realiza con el objetivo de aumentar el valor de umbral, evitando inconsistencias al decidir sobre el estado del parqueadero en los escenarios nublado y nocturno. El desarrollo completo del algoritmo de detección se lo aborda en la Sección 4.1.4.

Finalmente, basado en las curvas de comportamiento de los tres ambientes analizados, se han elegido valores de resistencias que permitan realizar una caracterización adecuada de la cantidad de luz presente en cada uno de los ambientes, para detectar de una manera precisa el estado de cada parqueo, las cuales son: 470Ω, 1KΩ, 3.3KΩ, 10KΩ, 33KΩ, 56KΩ, 82KΩ y 100KΩ.

Diseño esquemático y PCB del módulo comparador

Como se mencionó en la parte introductoria de la Sub Sección 4.1.1 y como se puede observar en la Figura 4.8, el prototipo del módulo comparador se lo ha realizado tomando en cuenta los dispositivos que lo componen, para lo cual se ha usado el software de desarrollo de placas electrónicas Altium Designer.[40]

La alimentación inicial de cada uno de los módulos que conforman el sistema de WSN es de 12V Direct Current (DC), sin embargo, tomando en cuenta los niveles de voltaje tanto para la alimentación como para las señales, se han implementado dos reguladores al inicio de cada módulo.

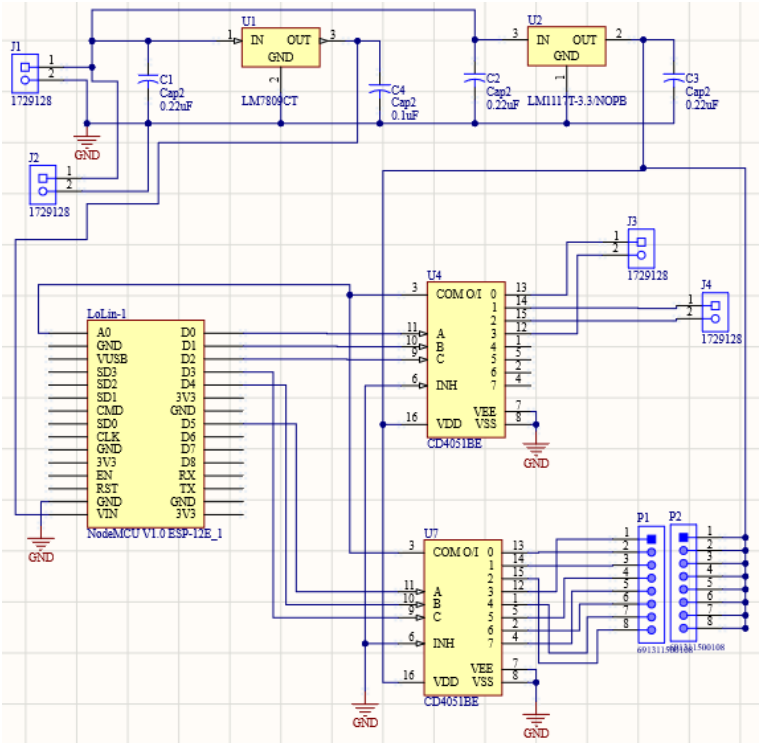


Figura 4.8: Diseño esquemático del módulo comparador

El primer regulador es denominado LM7809 que puede ser energizado hasta con 35V DC generando una salida de 9V y corriente de hasta 1A [41], este regulador alimenta únicamente el dispositivo NodeMCU el cual, según la Tabla 2.1, soporta hasta 10V DC como voltaje de entrada. Dado que la tensión de funcionamiento del NodeMCU es de 3.3V DC, el segundo regulador utilizado se denomina LM1117, que puede ser alimentado hasta con 15V DC generando una salida de 3.3V y corriente de hasta 500mA [42]; este regulador energiza los demás circuitos integrados presentes en el módulo al mismo nivel de voltaje con el que funcional el NodeMCU, eliminando los errores de comunicación por las diferencias de voltaje entre las señales a procesar.

Cada uno de los reguladores cuenta con un par de condensadores electrolíticos, cuyos valores son sugeridos en la propia hoja de información de cada regulador, los cuales eliminan cualquier señal indeseable que puede afectar al circuito, estabilizando la alimentación otorgada a cada dispositivo presente.

Los circuitos integrados que se utilizan en el módulo comparador son los multiplexores CD4051BE mencionados en la Sección 2.6, su conmutación BBM permite seleccionar cualquier canal sin que exista riesgo de interferencia entre las señales, lo que también protege al circuito de ESD.

En base a la Tabla 2.2 los dos multiplexores están conectados, por un lado al NodeMCU el cual selecciona un canal por medio de una combinación de 3 bits en los pines 9,10 y 11 y recibirá la información a través de su pin analógico (A0) y por otro lado, uno está conectado a las salidas para los tres sensores LDR mediante borneras y el otro a las ocho resistencias definidas anteriormente, formando así un partidor de tensión entre las resistencias y los sensores LDR. Los pines número 3 (COM) de ambos multiplexores, se conectan al pin analógico (A0) del NodeMCU, obteniendo los niveles de voltaje del partidor de tensión diseñado.

Una vez realizado el diseño esquemático del módulo comparador, se procede a generar el diseño de la PCB, como se observa en la Figura 4.9 el diseño se lo ha realizado teniendo en cuenta diferentes reglas y sugerencias de EMC mencionadas en [10] y [27], las cuales son:

1. **Minimizar EMI:**
   - *a*) Se han seleccionado componentes resistentes a ESD, que a su vez han sido desarrollados en base a pruebas con los modelos HBM y MM.
   - *b*) La topología del circuito muestra zonas diferentes y definidas por sus componentes. Se observa que los reguladores de voltaje se encuentran separados del NodeMCU y los multiplexores.
   - *c*) Las pistas de los contactos mantienen una distancia de 1mm, lo que disminuye la existencia de interferencias entre los pines de alimentación y de señales.
2. **Maximizar inmunidad del receptor:**
   - *a*) Gracias a la existencia de un solo plano de masa, y que este rodea completamente a los dispositivos que componen el módulo comparador, no existen extensiones de pistas que pueden actuar como antenas. A su vez, gracias a que el NodeMCU es un dispositivo SoC, este cuenta con su propio blindaje contra EMI, lo que, conjuntamente con el plano de masa, la señal inalámbrica WiFi no afecta a los demás dispositivos.
   - *b*) Se han definido topologías de zonas según su funcionamiento, los circuitos integrados se encuentran en la parte central de la placa, mientras que las borneras y resistencias están en la periferia, lo que disminuye cualquier daño físico o eléctrico a los multiplexores y NodeMCU
3. **Minimizar el camino de acoplamiento del ruido:**
   - *a*) Debido a que las fuentes de alimentación son susceptibles a generar ruido, los reguladores se encuentran en la periferia del módulo comparador y cuentan con condensadores que filtran las perturbaciones eléctricas existentes. Debido a que el módulo no trabaja a frecuencias altas, y el NodeMCU cuenta con su propia protección, no es necesario la presencia de condensadores destinados para desacoplamiento.
   - *b*) Distribución adecuada de las pistas permite que la mayoría de estas se encuentren en una sola capa, existiendo una única pista en la capa superior de la placa, que es de alimentación de 3.3V. La ubicación de los dispositivos permite que las pistas tengan la menor longitud posible sin que exista interferencia entre los mismos, manteniendo una distancia de 1mm entre pistas.

     Al cambiar de dirección en cada pista, estas forman un ángulo de 135^o lo que evita que se creen esquinas que pueden convertirse en puntos de radiación. Finalmente, la separación entre las pistas y el plano plano de masa es de 1mm, impidiendo que existan puntos de cortocircuito.

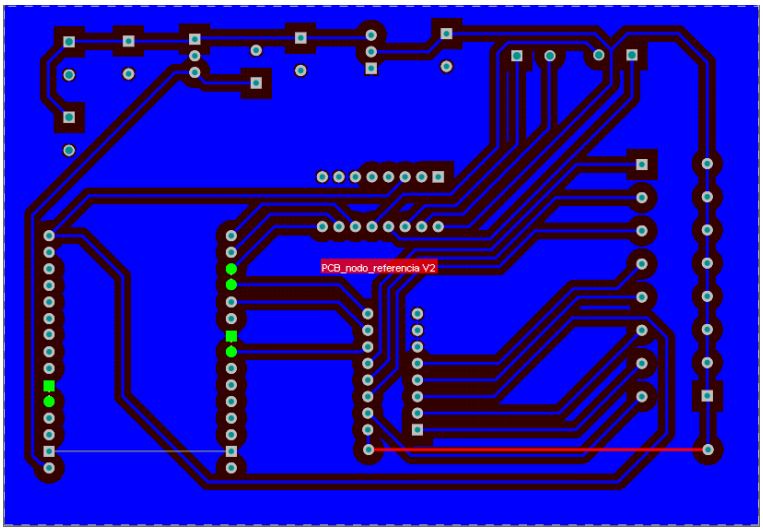


Figura 4.9: Diseño en PCB del Módulo Comparador

Finalmente, en la Figura 4.10 se puede observar cómo quedará el módulo comparador listo para su funcionamiento. Los dos bloques de 8 terminales serán reemplazados por las 8 resistencias definidas en la Sub Sección 4.1.1.

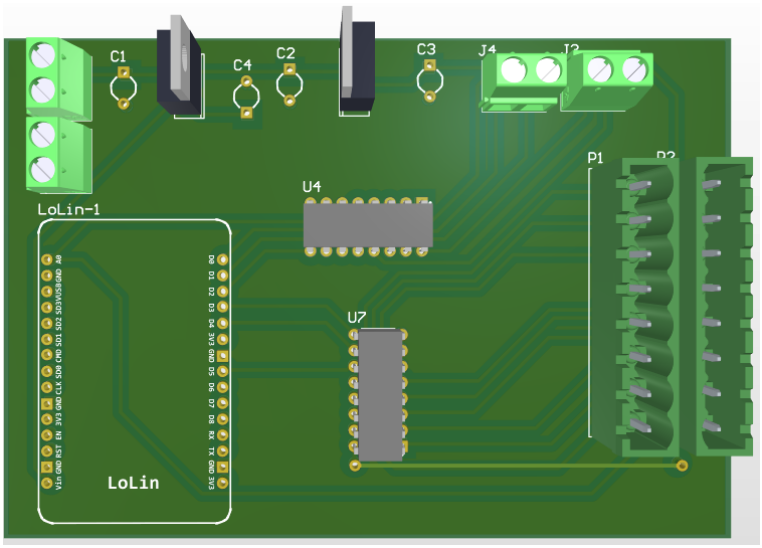


Figura 4.10: Representación en 3D del Módulo Comparador

Programación de módulo NodeMCU

En base al funcionamiento descrito en la introducción de la Sub Sección 4.1.1, se realiza la programación del NodeMCU, para ello se utiliza el software de programación Arduino IDE [43]. Dado que la tarjeta de desarrollo NodeMCU no es nativa de Arduino, para poder programar este dispositivo, se procede a agregar la siguiente Uniform Resource Locator (URL) `http://arduino.esp8266.com/stable/package_esp8266com_index.json` en la ventana "Preferencias"dentro de la sección *Gestor de URLs Adicionales de Tarjetas* como se puede observar en la Figura 4.11.

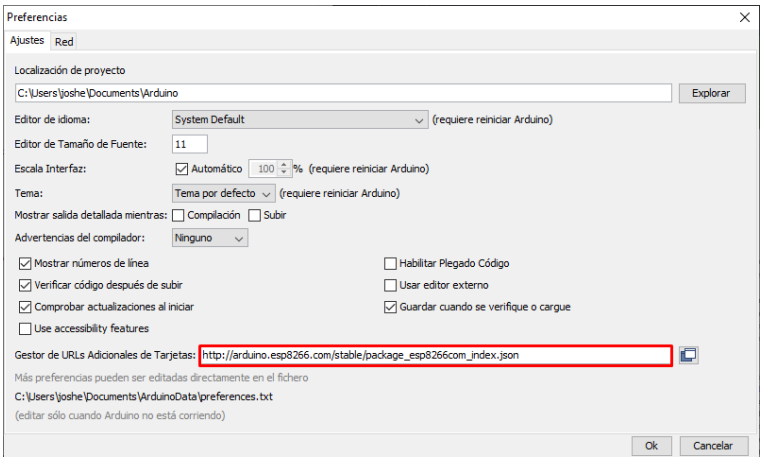


Figura 4.11: Gestor de Tarjetas - Arduino IDE

A continuación, se describen las partes más importantes del código implementado. El código completo se lo encuentra en el apéndice A.

- Se importan las librerías necesarias para la compilación adecuada del código,

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

```

- Se definen las variables iniciales y los pines con los que se controlaran los multiplexores.

```
int sensorPin = A0;
uint16_t sensorValue = 0;
const int node_id = 0; // Nodo 0 es el nodo de referencia
// Nombre del Topic en donde se publicará la información
const String data_out_topic = "datos_sensor";
Pines para controlar qué LDR leer (Bit 0 - 7)
uint8_t pin_a_ldrs = 16; // LSB //
uint8_t pin_b_ldrs = 5;
uint8_t pin_c_ldrs = 4; // MSB
// Pines para controlar con qué resistencia leer (Bit 0 - 7)
uint8_t pin_a_res = 14;//LSB
uint8_t pin_b_res = 2;
uint8_t pin_c_res = 0; // MSB

int choosen_ldr = 0;
int counter = 0 ;
int index_ldr = 3;
int interval = 5000;
uint8_t current_value = 0;
uint8_t var = 0;

```

- Se define el nombre de la Red, contraseña y dirección IP del Brooker MQTT

```
const char* ssid = "Linksys01263";
const char* password = "123456789";
const char* mqtt_server = "192.168.1.100";

```

- Se realiza la conexión entre el NodeMCU y la RPi

```
1   void setup_wifi()
2   {
3     delay(10);
4     // We start by connecting to a WiFi network
5     WiFi.begin(ssid, password);
6     while (WiFi.status() != WL_CONNECTED)
7     {
8       delay(500);
9     }
10  }
11
```

- Se realiza un censado cada 5 segundos, escogiendo inicialmente el sensor LDR con índice 0 y se obtienen los valores de voltaje en base a un barrido de las 8 resistencias, se agrupan los 8 valores de voltaje en formato JSON y se los publica en el *topic* "datos_sensor", antepuesto del identificador del nodo o módulo y del índice del sensor. Se realiza el mismo procedimiento con los dos sensores restantes.

  El índice 0 pertenece al escenario que simula un parqueo ocupado, el índice 1 pertenece al escenario que simula un parqueo libre con semi sombra, y el índice 2 pertenece al escenario que simula un parqueo libre e iluminado.

```
1   if (now - lastMeasure > 5000)
2   {
3     const size_t capacity=JSON_ARRAY_SIZE(8)+JSON_OBJECT_SIZE(4);
4     double valor = 0;
5     lastMeasure = now;
6
7     DynamicJsonDocument doc(capacity);
8     doc.clear();
9     doc["node"] = node_id;
10    // indice del LDR muestreado
11    doc["parking_index"] = choosen_ldr;
12    // El valor que se publica a los Leds(en decimal)
13    doc["current_value"] = current_value;
14    JsonArray data = doc.createNestedArray("data");
15
16    // Barrido en resistencias de 0 - 7
17    for (int i = 0; i <= 7; i++)
18    {
19      // Con esto barremos las resistencias no los LDRs
20      choose_res(i);
21      valor = read_ldr(choosen_ldr) * 3.3 / 1024;
22      data.add(valor);
23      delay(100);
24    }
25    String output = "";
26    serializeJson(doc, output); // Hacia el String
27    int tam = output.length() + 1;
28    char buf[tam];
29    output.toCharArray(buf, tam);
30    client.publish("datos_sensor", buf);
31    choosen_ldr++;
32  }
33
```

- Finalmente, se definen las funciones *read_ldr* y *choose_res*, las cuales reciben los valores binarios y del bucle anterior y los envían a las salidas del NodeMCU para controlar los multiplexores.

```
1   //valor : índice del ldr a leer  0 <= valor <= 7 en bits
2   int read_ldr (uint8_t valor)
3   {
4     digitalWrite(pin_a_ldrs, bitRead(valor, 0));
```

```
5       digitalWrite(pin_b_ldrs, bitRead(valor, 1));
6       digitalWrite(pin_c_ldrs, bitRead(valor, 2));
7       delay(10);
8       return analogRead(sensorPin);
9     }
10
11    //valor: índice de la resistencia a usar 0 <= valor <= 7 en bits
12    void choose_res (uint8_t valor)
13    {
14      digitalWrite(pin_a_res, bitRead(valor, 0));
15      digitalWrite(pin_b_res, bitRead(valor, 1));
16      digitalWrite(pin_c_res, bitRead(valor, 2));
17    }
18
```

Implementación del módulo comparador

Una vez que se ha realizado la impresión de la PCB y se ha implementado el código desarrollado en el NodeMCU se procede a soldar los componentes y probar el funcionamiento del código y del módulo general.

En la Figura 4.12 se puede observar la composición de los datos enviados, en donde primero se identifica el nodo, es decir es el número identificador del módulo comparador, en este caso 0. A continuación se encuentra el índice correspondiente a cada uno de los 3 sensores LDR y, finalmente, se encuentran los valores de voltaje obtenidos por el barrido de las 8 resistencias, desde la resistencia de menor valor óhmico hasta la de mayor valor óhmico.


```
WiFi connected - ESP IP address: 192.168.1.116
Attempting MQTT connection...connected
Datos bufer: {"node":0,"parking_index":0,"current_value":0,"data":[0,0,0,0,0,0.599414,0.341602,0.602637]}

Datos bufer: {"node":0,"parking_index":1,"current_value":0,"data":[0,0,0,0,0,0.979688,0.750879,0.676758,0.686426]}

Datos bufer: {"node":0,"parking_index":2,"current_value":0,"data":[0.077344,0.003223,0.103125,0.067676,0.125684,0.167578,0.212695,0.444727]}

Datos bufer: {"node":0,"parking_index":0,"current_value":0,"data":[0.244922,0.270703,0.296484,0.296484,0.277148,0.299707,0.293262,0.309375]}

Datos bufer: {"node":0,"parking_index":1,"current_value":0,"data":[0.032227,0.25459,0.560742,0.435059,0.209473,0.141797,0.183691,0.319043]}

Datos bufer: {"node":0,"parking_index":2,"current_value":0,"data":[0.299707,0.373828,0.818555,0.944238,0.708984,0.979688,0.76377,1.028027]}
```


Figura 4.12: Datos enviados desde el módulo comparador

Finalmente, la Figura 4.13 se puede observar la tarjeta el módulo comparador completamente armada, en donde se evidencia la localización de cada elemento en base a su función, manteniendo un distanciamiento adecuado para disminuir la existencia de EMI, mientras que en la Figura 4.14 se encuentra completamente armado el módulo comparador, junto con pequeñas maquetas en donde se encuentran los sensores LDR que sirven para simular los tres ambientes que servirán de referencia, listo para realizar las pruebas de laboratorio que serán abordadas en la Sección 5.1,

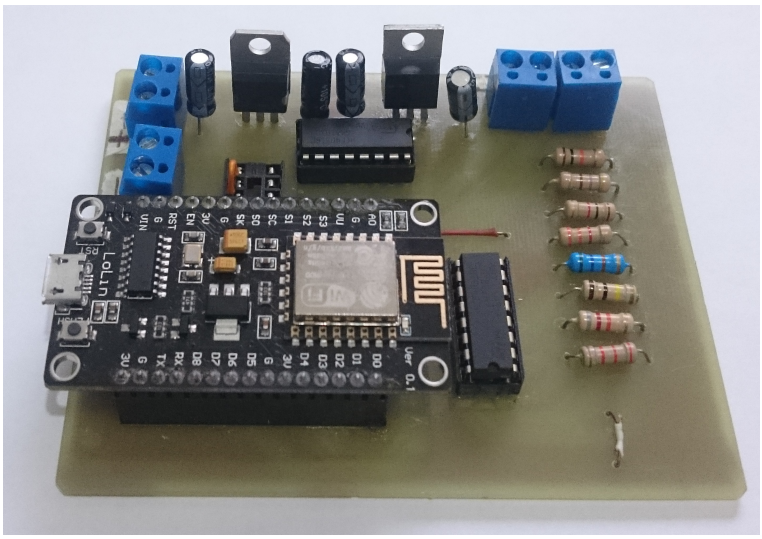


Figura 4.13: Módulo comparador impreso y armado

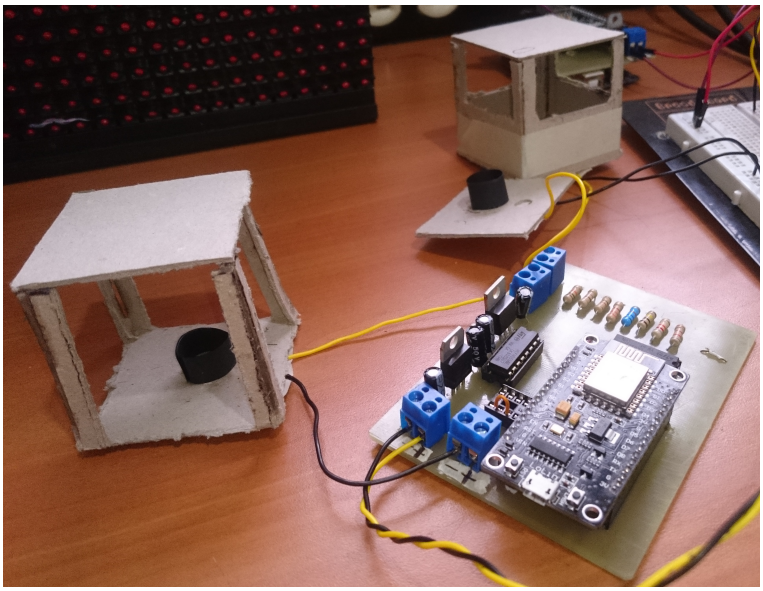


Figura 4.14: Módulo comparador para pruebas

Modelado y construcción de equipo protector

El objetivo del módulo comparador, es que sirva de referencia para tomar decisiones en base a los valores censados por el módulo sensor, para ello, es necesario que se lo ubique en un lugar en donde sus mediciones no sufran perturbaciones o interferencias. Cerca de la cima de un poste de iluminación es un lugar propicio, puesto que podrá censar la luz presente en el ambiente, tanto de día como de noche, sin ser expuesto a obstáculos que generen sombras sobre el módulo. Debido a que el módulo comparador estará ubicado en la intemperie, es necesario que tenga la protección adecuada, para ello se ha diseñado una carcasa por medio del software *Fusion 360*. [44]

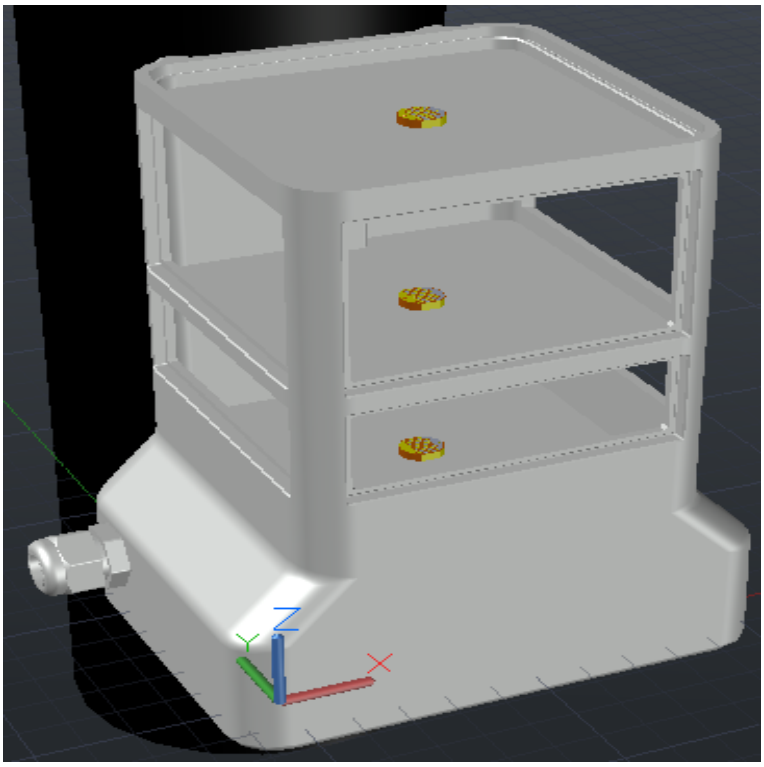


Figura 4.15: Diseño 3D del Módulo Comparador

Como se puede observar en la Figura 4.15, la carcasa cuenta de 4 niveles, en el primer nivel se encuentra encapsulada la tarjeta del módulo comparador, el segundo nivel simula el escenario de parqueo ocupado, dejando un espacio reducido a la entrada de luz; el tercer nivel simula el escenario de parqueo libre con sombra, mientras que el ultimo nivel simula el escenario del parqueo libre bajo luz directa.

La Figura 4.16, muestra el primer nivel que contiene la tarjeta del módulo comparador completamente protegida.

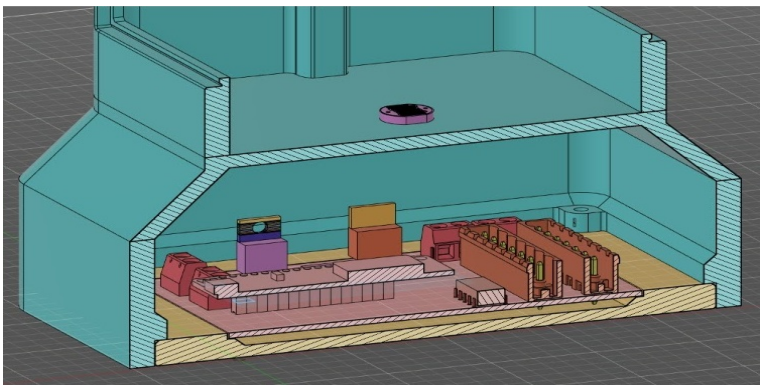


Figura 4.16: Tarjeta del Módulo Comparador encapsulada en la carcasa

La carcasa cual fue impresa en 3D con filamento *PETG* [45], este filamento permite impermeabilizar y robustecer la carcasa, siendo capaz de soportar diferentes cambios ambientales y golpes. En la Figura 4.17 se

puede observar el módulo comparador completamente armado y listo para ser instalado, cuenta con un con un conector de cable o prensaestopa PG9 para la entrada de cables de alimentación, el cual es utilizado para impermeabilizar y proteger las entradas y salidas de los cables.[46]

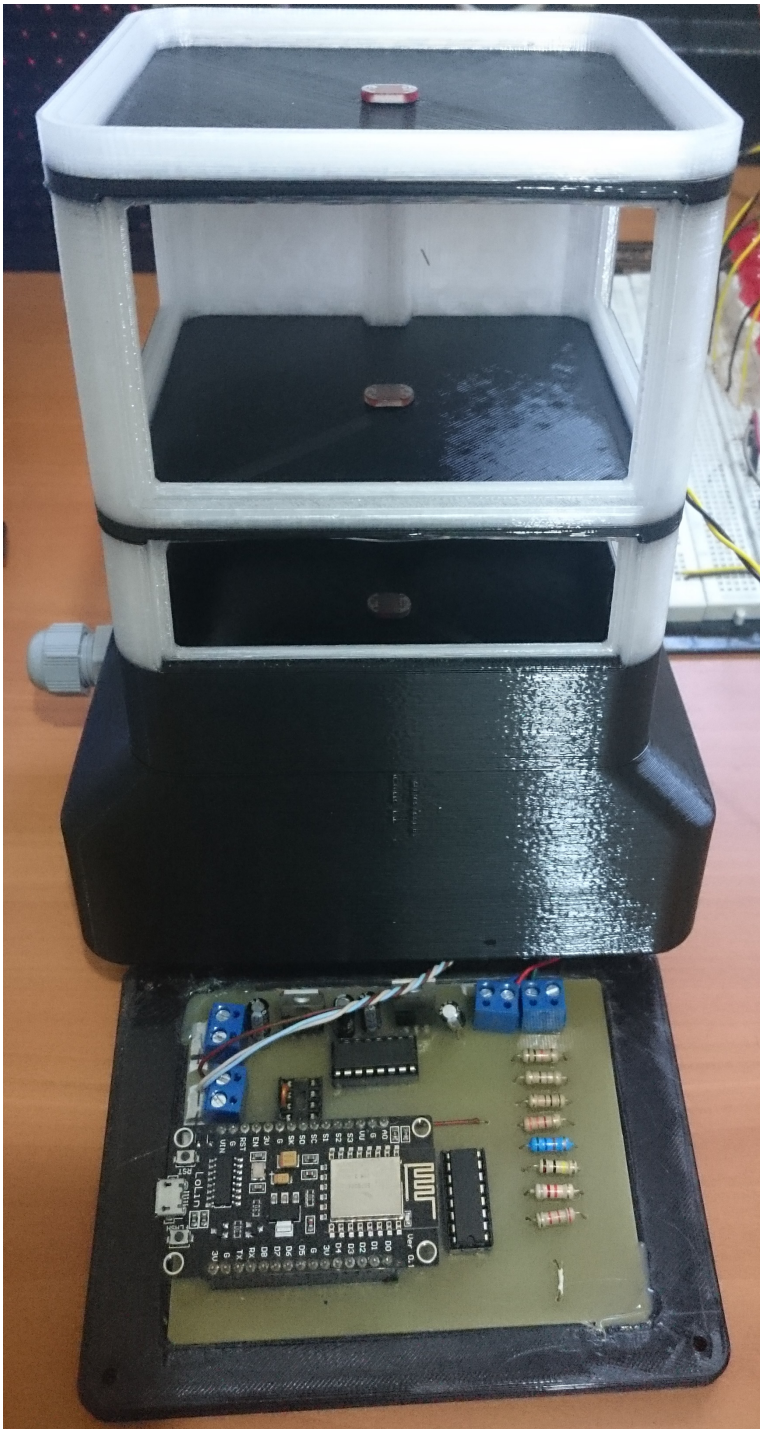


Figura 4.17: Prototipo del Módulo Comparador finalizado

Finalmente, en las ventanas de la protección se han ubicado pequeñas láminas de acrílico de 3mm de grosor

y se han sellado los bordes con silicona, así, se puede garantizar a durabilidad y eficiencia de la carcasa, gracias a que los materiales cumplen con el estándar IP mencionado en la Sección 2.9,

4.1.2. Módulo sensor

El Módulo Sensor está conformado por ocho sensores LDR, dos multiplexores CD4051BE, dos registros de desplazamiento 74LS595, dos controladores UNL2803, un dispositivo NodeMCU y 8 resistencias para la caracterización del comportamiento de cada sensor LDR. Cada uno de los sensores LDR se encuentra ubicado en el centro de cada parqueadero, con el objetivo de detectar si el parqueo se encuentra ocupado o libre, mientras que las 8 resistencias, descritas en la Sub Sección 4.1.1, sirven para obtener una curva de comportamiento de cada sensor LDR en base a la iluminación presente en cada ambiente.

Por medio de un dispositivo NodeMCU se controlan dos multiplexores CD4051, cuyo funcionamiento se menciona en la Sección 2.6, los cuales permiten seleccionar el sensor LDR y la resistencia de interés. Primero, se selecciona uno de los ocho sensores LDRs y se multiplexa con cada una de las ocho resistencias, de menor a mayor valor óhmico, obteniendo ocho valores diferentes de voltaje, que son leídos por medio del pin de entrada analógica (A0) del NodeMCU. Los valores de voltaje obtenidos se los almacena en un archivo JSON y son publicados en un *topic* por medio del protocolo MQTT al que la RPi tendrá acceso; el procedimiento se repite con los demás sensores LDR.

Los datos son procesados en la RPi por medio del algoritmo de detección implementado en la Sub Sección 4.1.4, el cual indica los parqueaderos libres mediante un vector de 8 elementos, configurando en cada elemento el valor 0 si está disponible y 1 si está ocupado. Este vector es publicado en un *topic* por medio del protocolo MQTT al cual tendrán acceso el módulo sensor y el panel LED.

El vector contiene la información del estado de cada uno de los parqueaderos, por lo que es procesado por medio del NodeMCU del módulo comparador. Se realiza la operación NOT sobre el vector y este nuevo vector se lo concatena con el original. El tren de 16 bits resultante se lo envía de forma serial a los registros de desplazamiento 74LS595, los cuales transforman la entrada serial en una salida en paralelo como se menciona en la Sección 2.6, así, cada salida en paralelo de cada registro de desplazamiento tendrá un valor de 0 o 1. De esta forma, se tienen dos valores para cada parqueadero los cuales activarán un módulo LED de color verde si está disponible o activarán un módulo LED de color rojo si el parqueadero está ocupado.

Para que los módulos LED sean visibles a distancias mayores a 5 metros, es necesario alimentarlos con 12V, sin embargo, dado que los valores lógicos obtenidos en las salidas de los registros de desplazamiento están entre 0V y 3.3V, es necesario usar dos controladores UNL2803, que son alimentados directamente con 12V permitiendo aumentar la corriente y voltaje, y así, controlar eficazmente los módulos LED.

Diseño esquemático y PCB del módulo sensor

Como se mencionó en la parte introductoria de la Sub Sección 4.1.2 y como se puede observar en la Figura 4.18, El módulo sensor cumple una función similar a la del módulo comparador, con la diferencia que en éste también se realiza parte de la visualización de los estados de los parqueaderos, exigiendo el uso de

otros componentes, para lo cual se ha usado el software de desarrollo de placas electrónicas *Altium Designer*. [40]

Tal como se menciona en la Sub Sección 4.1.1, la alimentación inicial del módulo comparador es de 12V DC, a lo que se siguen los reguladores LM7809 [41], que alimenta al dispositivo NodeMCU, y LM1117 [42] que alimenta los demás circuitos integrados presentes en el módulo al mismo nivel de voltaje con el que funciona el NodeMCU, eliminando los errores de comunicación por las diferencias de voltaje entre las señales a procesar. Cada uno de los reguladores cuenta con un par de condensadores electrolíticos, cuyos valores son sugeridos en la propia hoja de información de cada regulador, los cuales eliminan cualquier señal indeseable que puede afectar al circuito, estabilizando la alimentación otorgada a cada dispositivo presente.

La conmutación BBM realizada por los multiplexores CD4051 permite seleccionar cualquier canal sin que exista riesgo de interferencia entre las señales, lo que también protege al circuito de ESD. En base a la Tabla 2.2 los dos multiplexores están conectados, por un lado al NodeMCU el cual selecciona un canal por medio de una combinación de 3 bits en los pines 9,10 y 11 y recibirá la información a través de su pin analógico (A0) y por otro lado, uno está conectado a las salidas para los ocho sensores LDR mediante borneras y el otro a las ocho resistencias definidas anteriormente, formando así un partidor de tensión entre las resistencias y los sensores LDR. Los pines número 3 (COM) de ambos multiplexores, se conectan al pin analógico (A0) del NodeMCU, obteniendo los niveles de voltaje del partidor de tensión diseñado.

Como se puede observar en la Figura 4.18, los circuitos integrados 74LS595 mencionados en la Sección 2.6 convierten una entrada de datos serial en una salida de datos en paralelo. En la Tabla 2.3 se observa que el pin 9 permite la conexión en cascada de más registros de desplazamiento, lo cual le permite al primer registro recibir el tren de 16 bits desde el NodeMCU y junto con el segundo registro, convertirlo en salidas en paralelo.

Finalmente, las salidas de los circuitos integrados 74LS595 servirán como señales piloto para los controladores UNL2803, los cuales, al estar alimentados directamente con 12V facilitan el control directo de los módulos LED.

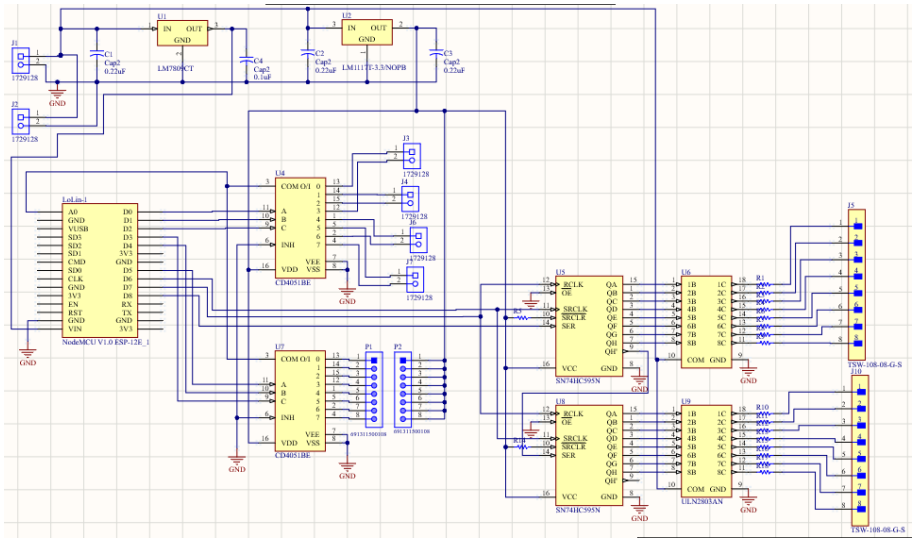


Figura 4.18: Diagrama esquemático del módulo sensor

Una vez realizado el diseño esquemático del módulo sensor, se procede a generar el diseño de la PCB, como se observa en las figuras 4.19 y 4.20 el diseño se lo ha realizado teniendo en cuenta diferentes reglas y sugerencias de EMC mencionadas en [10] y [27], las cuales son:

1. **Minimizar EMI:**
   - *a*) Como se menciona en la Sección 2.6, se han seleccionado componentes resistentes a ESD
     - El circuito integrado CD4051 trabaja con multiplexación BBM lo que disminuye el riesgo de un cortocircuito interno o interferencias, siendo resistente ante ESD.
     - El circuito integrado 74LS595 cuenta con una alta inmunidad al ruido de los dispositivos CMOS, conjuntamente con un alto rendimiento a ESD en base a los modelos HBM y MM.
     - El circuito integrado UNL2803 cuenta con un circuito de sujeción por diodos en cátodo común, lo que lo hace apto para conmutar cargas inductivas y proteger al circuito de sobretensión y ESD como se puede observar en la Figura 2.17.
   - *b*) Las pistas de los contactos mantienen una distancia de 1mm, lo que disminuye la existencia de interferencias entre los pines de alimentación y de señales.
2. **Maximizar inmunidad del receptor:**
   - *a*) Como se observa en las figuras 4.19 y 4.20, los diseños PCB de la capa superior y la capa inferior cuentan con un plano de masa que rodea todos los componentes, permitiendo el retorno de las señales espurias que se pueden producir por los dispositivos electrónicos. A su vez, gracias a que el NodeMCU es un dispositivo SoC, este cuenta con su propio blindaje contra EMI, lo que conjuntamente con el plano de masa, la señal inalámbrica WiFi no afecta a los demás dispositivos.

   - *b*) Se han definido topologías de zonas según su funcionamiento, los circuitos integrados se encuentran en la parte central de la placa, mientras que las borneras y resistencias están en la periferia, lo que disminuye cualquier daño físico o eléctrico a los multiplexores y NodeMCU
3. **Minimizar el camino de acoplamiento del ruido:**
   - *a*) Debido a que las fuentes de alimentación son susceptibles a generar ruido, los reguladores se encuentran en la periferia del módulo comparador y cuentan con condensadores que filtran las perturbaciones eléctricas existentes. Debido a que el módulo no trabaja a frecuencias altas, y el NodeMCU cuenta con su propia protección, no es necesario la presencia de condensadores destinados para desacoplamiento.

   - *b*) Distribución adecuada de las pistas permite que la mayoría de estas se encuentren en la capa inferior y en la capa superior se encuentran las pistas de alimentación de 12V y 3.3V rodeadas por el plano de masa. La ubicación de los dispositivos permite que las pistas tengan la menor longitud posible sin que exista interferencia entre los mismos, manteniendo una distancia de 1mm entre pistas.

     Al cambiar de dirección en cada pista, estas forman un ángulo de 135^o lo que evita que se creen esquinas que pueden convertirse en puntos de radiación. Finalmente, la separación entre las pistas y el plano de masa es de 1mm, impidiendo que existan puntos de cortocircuito.

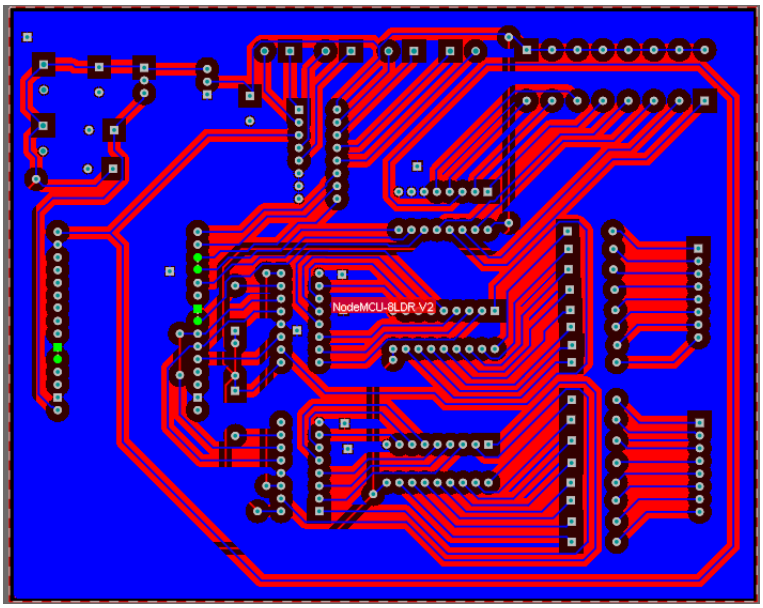


Figura 4.19: Diseño en PCB del Módulo Sensor - Capa Inferior

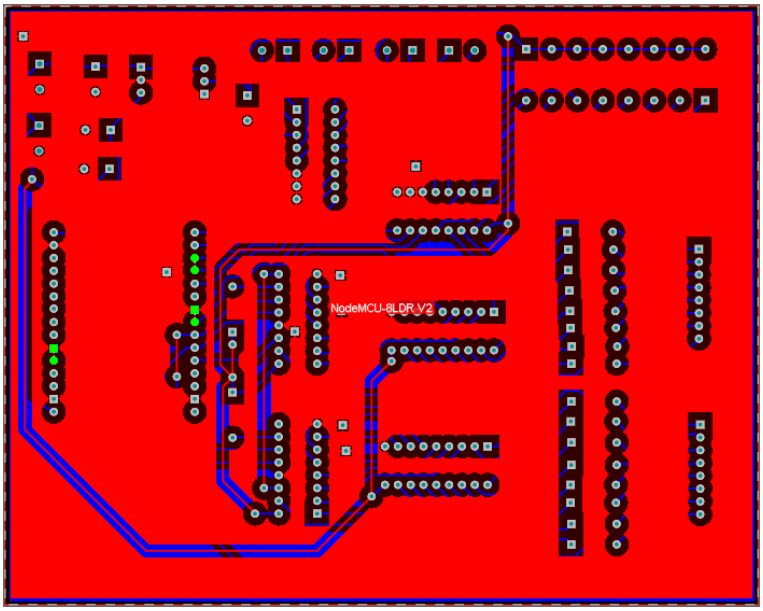


Figura 4.20: Diseño en PCB del Módulo Sensor - Capa Superior

Finalmente, en la Figura 4.21 se puede observar cómo quedará el módulo comparador listo para su funcionamiento. Los dos bloques de 8 terminales serán reemplazados por las 8 resistencias definidas en la Sub Sección 4.1.1. También se puede observar dos conectores de 8 pines que servirán como interfaz para conectar la tarjeta del módulo sensor con la placa de potencia.

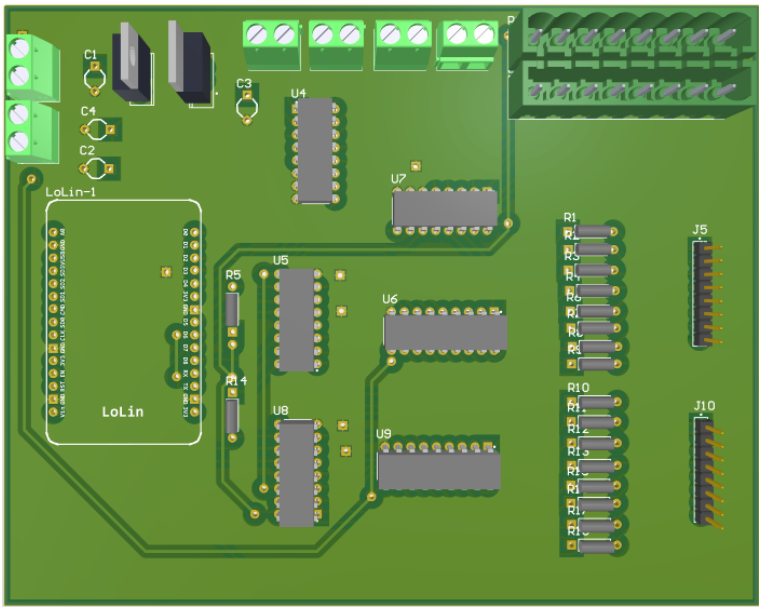


Figura 4.21: Representación en 3D del Módulo Sensor

Diseño esquemático y PCB de la placa de potencia

Como se puede observar en la Figura 4.22, el diseño esquemático de la placa de potencia consta de dos conectores de 8 pines, el primer conector sirve para recibir las señales que activaran o desactivaran los módulos LED de color verde, mientras que el segundo conector sirve para recibir las señales que activaran o desactivaran los módulos LED de color rojo. Cada uno de los pines se encuentra conectado con un terminal de bornera, permitiendo acoplar cada módulo con su señal respectiva.

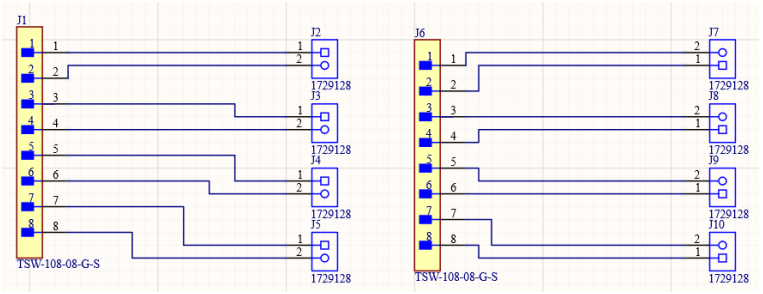


Figura 4.22: Diseño Esquemático de la Placa de Potencia

Debido a que la placa de potencia es una interfaz entre la tarjeta del módulo sensor y los módulos LED, el diseño PCB de ésta tiene un plano de tierra con el objetivo de evitar cualquier tipo de interferencia entre los conectores, como se observa en la Figura 4.23.

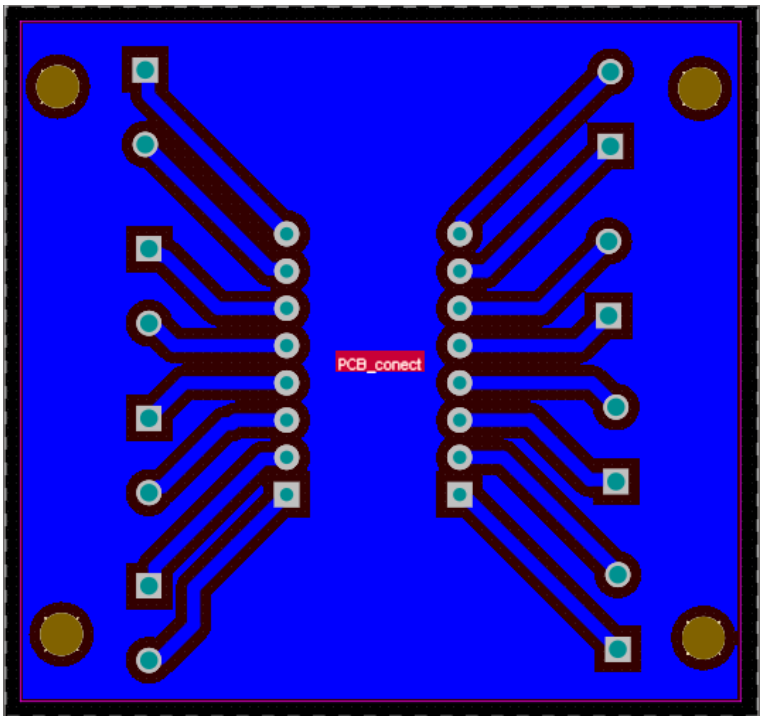


Figura 4.23: Diseño PCB de la Placa de Potencia

Finalmente, en la Figura 4.24 se puede observar cómo quedará la placa de control de los módulos LED listo para su funcionamiento.

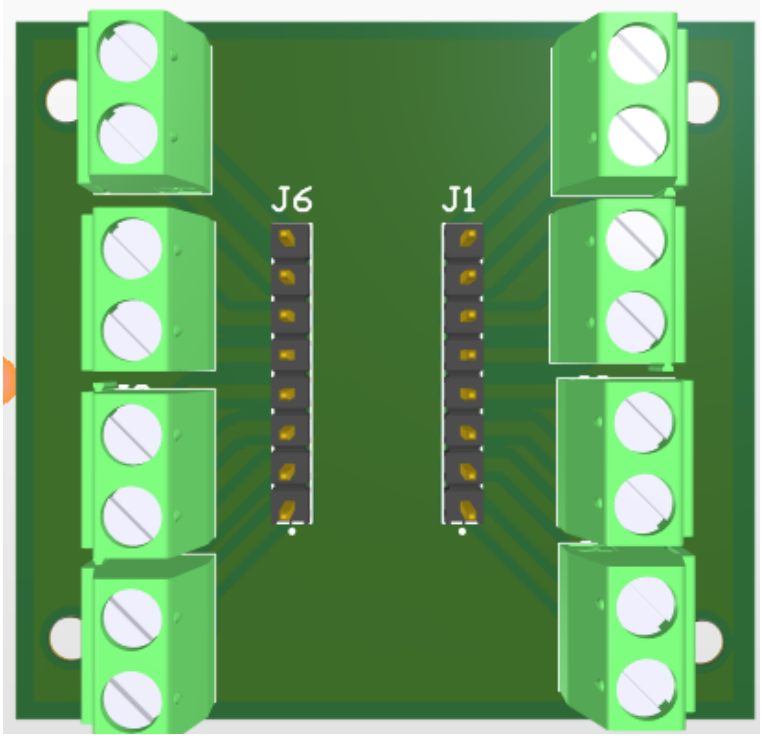


Figura 4.24: Representación en 3D de la Placa de Potencia

Programación del NodeMCU

A continuación, se describen las partes más importantes del código implementado. El código completo se lo encuentra en el apéndice A. El procedimiento de importación de las librerías necesarias para la compilación adecuada del código, definición del nombre de la Red, contraseña y dirección IP del *Brooker* MQTT, y, conexión del NodeMCU con la RPi es el mismo que el del código implementado en la Sub Sección 4.1.1.

De igual manera, el barrido de la información de cada sensor LDR se lo realiza cada 5 segundos, escogiendo inicialmente el sensor LDR con índice 0 se obtienen los valores de voltaje en base a las 8 resistencias, se agrupan los 8 valores de voltaje en formato JSON y se los publica en el *topic* "datos_sensor", antepuesto del identificador del nodo o módulo y del índice del sensor. Se realiza el mismo procedimiento con los ocho sensores restantes, Los índices 0 al 7, representan los parqueaderos del 1 al 8 respectivamente.

- Se definen las variables iniciales y los pines con los que se controlaran los multiplexores.

```
int sensorPin = A0;
uint16_t sensorValue = 0;


//TODO: revisar estos valores para cada nodo sensor
const int node_id = 1;
const String data_out_topic = "datos_sensor"; // este no cambiar

// pines para controlar los leds 0 - 7
uint8_t pin_clock_leds    = 12; // D6  Clock
uint8_t pin_data_leds     = 15; // D8  Latch
uint8_t pin_latch_leds    = 13; // D7  DATA_IN

// Pines para controlar qué ldr leer 0 - 7
uint8_t pin_a_ldrs = 16; // LSB
uint8_t pin_b_ldrs = 5;
uint8_t pin_c_ldrs = 4; // MSB

// Pines para controlar con qué resistencia leer el ldr 0 - 7
uint8_t pin_a_res = 14;//LSB
uint8_t pin_b_res = 2;
uint8_t pin_c_res = 0; // MSB

int counter = 0 ;
int index_ldr = 3;
int interval = 5000;
int choosen_ldr = 0;
uint8_t current_value = 0;

uint8_t var = 0;

```

- Se define la función *callback*, la cual accede al *topic* "nodo1"que contiene el estado de cada parqueadero en forma de un número decimal, lo transforma en un vector binario en donde cada dígito corresponde a un parqueadero en específico y finalmente lo envía a la función *putLEDs*.

```
void callback(String topic, byte* message, unsigned int length)
{
  String messageTemp;

  for (int i = 0; i < length; i++)
  {
    messageTemp += (char)message[i];
```

```
8       }
9       uint8_t val = messageTemp.toInt();
10
11      // manda a los leds
12      putLeds(val);
13      current_value = val;
14    }
15
```

- Finalmente, se definen las funciones *putLeds*, *read_ldr* y *choose_res*, las cuales reciben los valores binarios y los envían a las salidas del NodeMCU para controlar los multiplexores. La función *putLEDs* obtiene el vector recibido con la información de los estados de cada parqueadero por medio de la función *callback* y se realiza la operación NOT sobre el vector y este nuevo vector se lo concatena con el original. El tren de 16 bits resultante se lo envía de forma serial a los registros de desplazamiento 74LS595 por medio del pin D8 del NodeMCU.

```
1     void putLeds(uint8_t salidaLeds)
2     {
3       digitalWrite(pin_latch_leds, HIGH);
4       uint16_t valor_salida = ((~salidaLeds) << 8) | salidaLeds;
5       for (int i = 0; i < 16; i++)
6       {
7         digitalWrite(pin_clock_leds, LOW);
8         digitalWrite(pin_data_leds, bitRead(valor_salida, i));
9         digitalWrite(pin_clock_leds, HIGH);
10      }
11      digitalWrite(pin_latch_leds, LOW);
12    }
13    /*
14   valor : index del ldr a leer  0 <= valor <= 7
15    */
16    int read_ldr (uint8_t valor)
17    {
18      digitalWrite(pin_a_ldrs, bitRead(valor, 0));
19      digitalWrite(pin_b_ldrs, bitRead(valor, 1));
20      digitalWrite(pin_c_ldrs, bitRead(valor, 2));
21      delay(10);
22      return analogRead(sensorPin);
23    }
24
25    /*
26       valor: index de la resistencia a usar  0 <= valor <= 7
27    */
28    void choose_res (uint8_t valor)
29    {
30      digitalWrite(pin_a_res, bitRead(valor, 0));
31      digitalWrite(pin_b_res, bitRead(valor, 1));
32      digitalWrite(pin_c_res, bitRead(valor, 2));
33    }
34
```

Implementación del módulo sensor

Una vez que se ha realizado la impresión de la PCB y se ha implementado el código desarrollado en el NodeMCU se procede a soldar los componentes y probar el funcionamiento del código y del módulo general.

En la Figura 4.25 se puede observar la composición de los datos enviados, en donde primero se identifica el nodo, es decir, es el número identificador del módulo comparador, en este caso 1. A continuación se encuentra el índice correspondiente a cada uno de los 8 sensores LDR y, finalmente, se encuentran los valores de voltaje

obtenidos por el barrido de las 8 resistencias, desde la resistencia de menor valor óhmico hasta la de mayor valor óhmico.

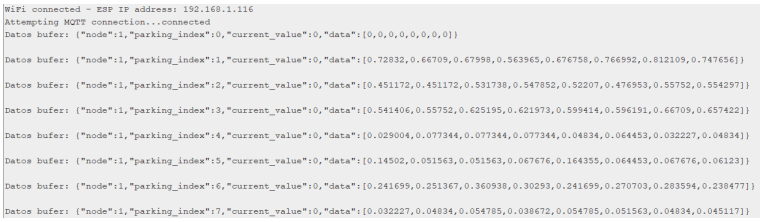


Figura 4.25: Datos enviados desde el Módulo Comparador

Finalmente, en la Figura 4.26 se puede observar la tarjeta el módulo sensor completamente armada, en donde se evidencia la localización de cada elemento en base a su función, el apantallamiento para pistas y un distanciamiento adecuado para disminuir la existencia de EMI, listo para realizar las pruebas de laboratorio que serán abordadas en la Sección 5.1,

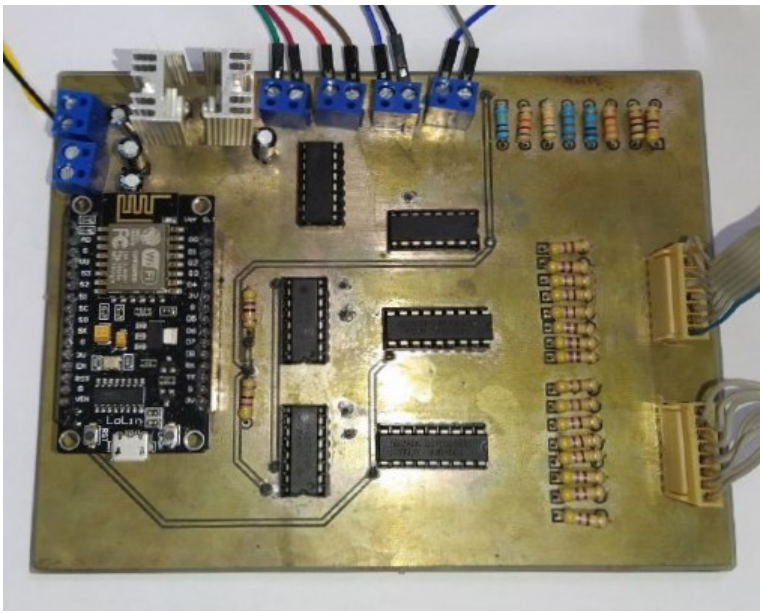


Figura 4.26: Módulo Sensor impreso y armado

Protecciones

Debido a las conexiones cableadas de los sensores LDR y los módulos LED con la tarjeta principal del módulo sensor, es necesario que éste se encuentre en un lugar cercano. De igual forma, puesto a que los módulos LED y los sensores LDR se encontraran instalados en el suelo, es necesario proveer de una protección adecuada para todos los elementos que componen el módulo sensor para que no sean afectados por los cambios ambientales como sol fuerte y lluvias, de igual manera, para que no sean dañados por algún golpe o caída. A continuación, se muestran las protecciones implementadas para los elementos del módulo sensor:

- Primero, se procede a realizar la protección adecuada para los sensores LDR dado que estos se encontrarán distribuidos en el suelo, en el centro de cada parqueadero. Como se puede observar en la Figura 4.27, se ha diseñado una estructura que permita sostener el sensor LDR y al mismo tiempo que lo proteja de caídas o aplastamientos, la cual va a ser generada por medio de impresión en 3D.

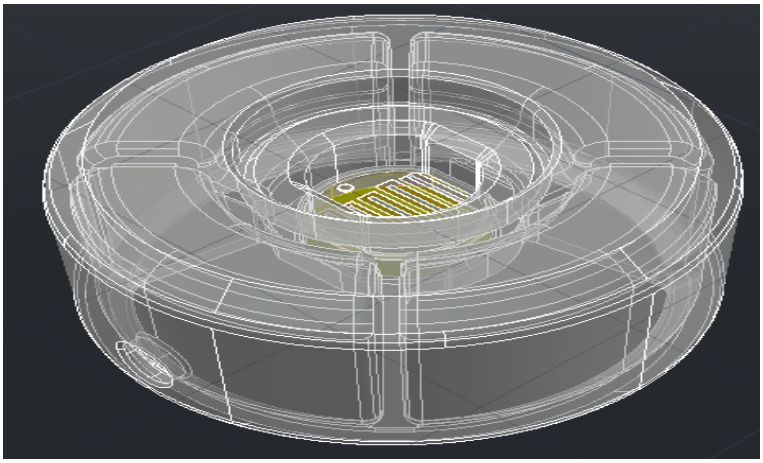


Figura 4.27: Diseño en 3D de la protección para el sensor LDR

En la Figura 4.28 se puede observar el prototipo completamente impreso, se lo ha realizado con filamento *PETG* [45] que permite impermeabilizar y robustecer la carcasa, siendo capaz de soportar diferentes cambios ambientales y golpes.

El material con el que se ha rellenado la estructura es resina de poliuretano que se caracteriza por su resistencia notable ante la tracción, compresión e impactos, se solidifica rápidamente y se puede adherir a distintas superficies, incluyendo el hormigón.[47][48]

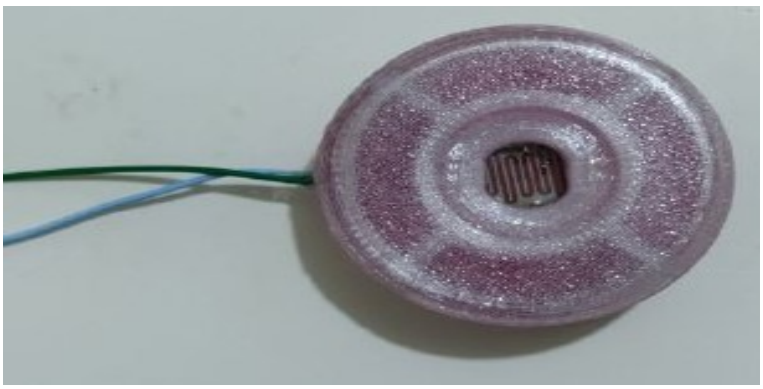


Figura 4.28: Protección del sensor LDR armada

- La protección para los módulos LED se la ha realizado de manera similar a la protección de los sensores LDR, en la Figura 4.29 se puede observar el diseño de una estructura que permite sostener el módulo LED mientras le brinda protección e impermeabilidad.

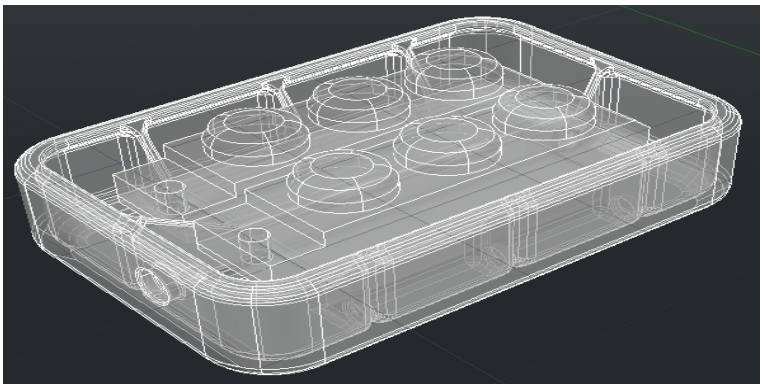


Figura 4.29: Diseño en 3D de la protección para el módulo LED

De igual forma, en la Figura 4.30 se puede observar el proceso de rellenado de la estructura con resina de poliuretano, lo que le da durabilidad, resistencia e impermeabilidad

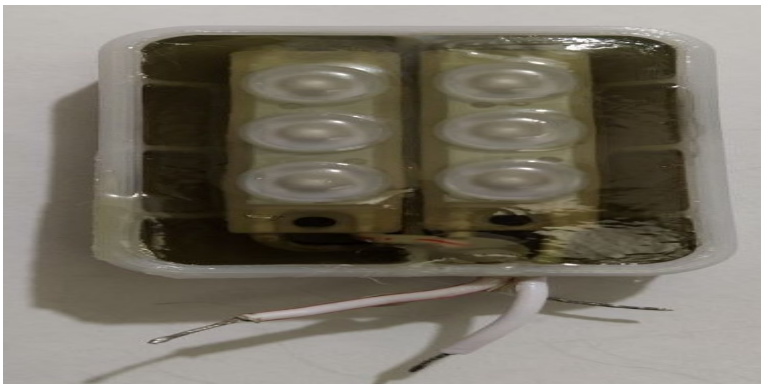


Figura 4.30: Protección del módulo LED armada

Con el objetivo de resguardar todos los elementos del módulo sensor, se procede a utilizar una caja de paso o caja estanca que es apta para todo tipo de ambientes. Es elaborada de material PolyVinyl Chloride (PVC) cumpliendo con el estándar IP65, mencionado en la Sección 2.9, lo que garantiza su alta resistencia al agua, polvo, agentes químicos, atmosféricos e impactos accidentales.

Dado que el módulo sensor y el módulo comparador son alimentados con 12V, se ubica una fuente conmutada dentro de la caja de paso, la cual cuenta con conectores de cable o prensaestopas PG9 para la entrada y salida de cables de alimentación y de señales que son utilizados para impermeabilizar y proteger las entradas y salidas de los cables [46]. En la Figura 4.31 se puede observar la distribución de los elementos del módulo sensor dentro de la caja estanca.

Figura 4.31: Protección del Módulo Sensor

4.1.3. Panel LED

Un punto importante en la resolución del problema de detección de parqueaderos libres, es la visibilidad. En esta Sub Sección de detalla la configuración del panel LED, el cual junto con los módulos LED, es una herramienta usada para informar acerca del estado de los parqueos censados. Este panel será controlado por medio de una configuración maestro/esclavo usando los dispositivos, NodeMCU como maestro y Arduino UNO como esclavo.

Al actuar como maestro, el NodeMCU accede al *topic panel* que contiene la identificación del parqueadero libre más cercano a la puerta principal de la facultad de Ingeniería. Una vez que obtiene la información de interés, la transmite al *Arduino UNO* que actúa como esclavo, el cual enviara la información al panel LED para que sea visualizada. En el *Arduino* se encuentra la codificación de la fuente de letra que se usara para visualizar en el panel LED.

La configuración maestro/esclavo pertenece al protocolo I2C, el cual se basa en una conexión de interfaz de bus en serie. También es denominado Two-Wire Interface (TWI) debido a que solo usa dos cables para establecer la comunicación, que son Serial Data (SDA) y Serial Clock (SCL). El transmisor comprueba que el receptor recibió los datos correctamente por medio de un acuse de recibo.[49]

De esta manera, el puerto SDA se utiliza para el intercambio de información entre los dispositivos, mientras

que el puerto SCL se utiliza para sincronizar el reloj entre los dispositivos. El dispositivo maestro requiere la dirección del esclavo para poder iniciar el envío de información.

Configuración del panel LED y Arduino Uno

Como se menciona en la Sección 2.5, cada panel estándar está compuesto por 512 diodos LED, sin embargo, en este trabajo experimental se trabaja con 3 módulos en cascada ubicados horizontalmente, lo que correspondería a un panel de 16X96 píxeles.

A pesar que el panel LED puede trabajar con valores CMOS y TTL, no se puede conectar directamente el NodeMCU debido a su limitación de corriente en cada uno de sus pines de salida, es por ello que se procede a utilizar un Arduino UNO como interfaz entre el panel y el NodeMCU; en la Figura 4.32 se puede observar la distribución de pines del Arduino Uno.

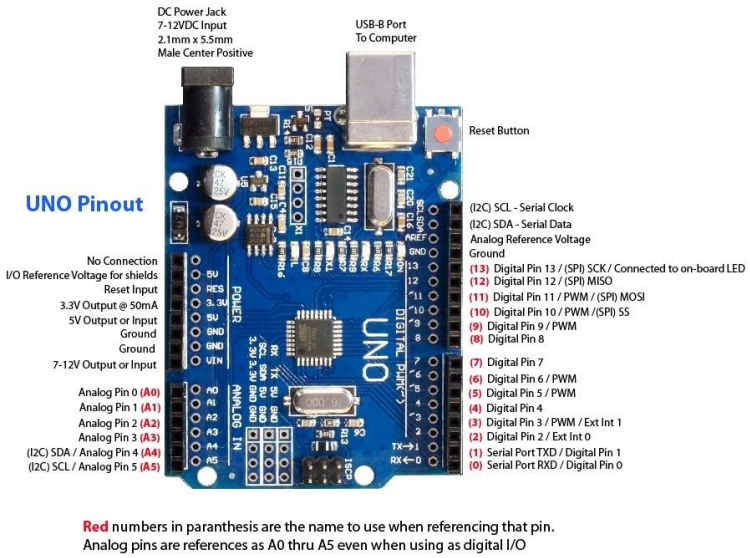


Figura 4.32: Distribución de pines del Arduino Uno [50]

Dado que la comunicación entre el Arduino UNO y el panel es por medio del protocolo SPI, es necesario destinar los pines correspondientes a Serial Clock (SCK), Master In Slave Out (MISO) y Master Out Slave In (MOSI) del Arduino Uno para la conexión con el puerto HUB12 del panel LED, según se detalla en la Figura 4.33. [51]

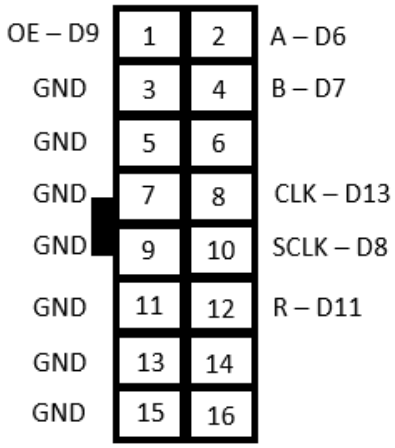


Figura 4.33: Conexión del Puerto HUB12 con pines de Arduino Uno

En base a la cantidad de caracteres que se busca mostrar a través del panel LED se ha desarrollado una fuente exclusiva para la visualización del mensaje. El software *GLCD Font Creator V2*, permite realizar el diseño de cada uno de los caracteres en base a los píxeles del panel que se deseen utilizar, tomando en cuenta que un píxel corresponde a un LED dentro del panel. En la Figura 4.34 se puede observar el diseño de los número y letras mayúsculas que son los caracteres que más se utilizarán.[52]

Figura 4.34: Caracteres diseñados para la visualización en el Panel LED

Finalmente, como se puede observar en la Figura 4.35, se presenta una cadena de caracteres en el panel LED por medio de un código de prueba.

Figura 4.35: Prueba de Comunicación y Fuente diseñada

Diseño esquemático y PCB del módulo controlador de panel LED

La configuración maestro/esclavo con la que se controlada el panel LED consta de los dispositivos NodeMCU y Arduino Uno. La comunicación entre estos dispositivos es por medio del protocolo I2C, mientras que la comunicación entre el Arduino Uno y el panel LED es por medio del protocolo SPI.

Con el objetivo de acoplar el módulo controlador con respecto al espacio existente dentro de la estructura del panel LED, se han sintetizado las conexiones entre los dispositivos en una sola placa de PCB. Como se puede observar en la Figura 4.36, la placa inicialmente será alimentada con 12V, a continuación cuenta con un regulador de LM7809 que alimentará el NodeMCU y el Arduino Uno; en el diseño esquemático se incluye un socket de 16 pines que se conectara con el puerto HUB12 del panel LED mediante un bus de cable plano

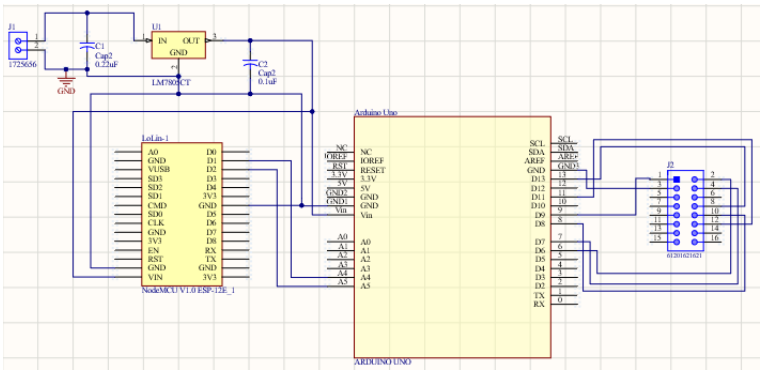


Figura 4.36: Diseño esquemático de la placa de conexión maestro/esclavo

El diseño de la PCB se lo realiza en base a las dimensiones del Arduino Uno, con el objetivo que la placa sirva como interfaz o *shield* entre el NodeMCU y el Arduino Uno, optimizando el espacio existente dentro del panel LED, como se lo puede observar en las figuras 4.37 y 4.38.

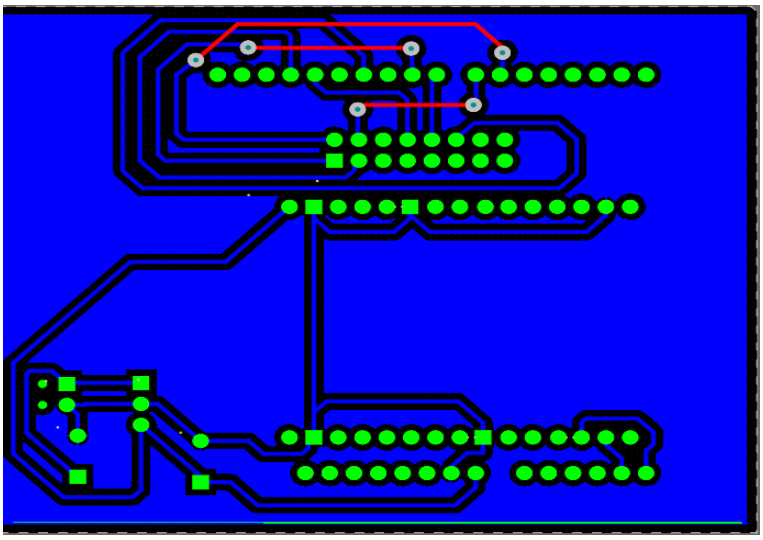


Figura 4.37: Diseño en PCB de la placa de conexión maestro/esclavo

Como se detalla en las Sub Secciones 4.1.1 y 4.1.2, el diseño en PCB se lo ha realizado en base a diferentes criterios de EMC, en donde se puede observar que:

- Los dispositivos se encuentran ubicados y separados en base a su función.
- Tanto el NodeMCU como el Arduino Uno son dispositivos SoC, por lo que cada uno cuenta con el blindaje adecuado para resistir diferentes fuentes de EMI,
- la presencia de un plano de masa entre los dos despectivos disminuye la existencia de efectos capacitivos o interferencias entre los dos dispositivos electrónicos, lo que a su vez protege los demás elementos como conectores de ruidos.
- Las pistas y pines mantienen una distancia de 1mm entre sí y con respecto al plano de masa

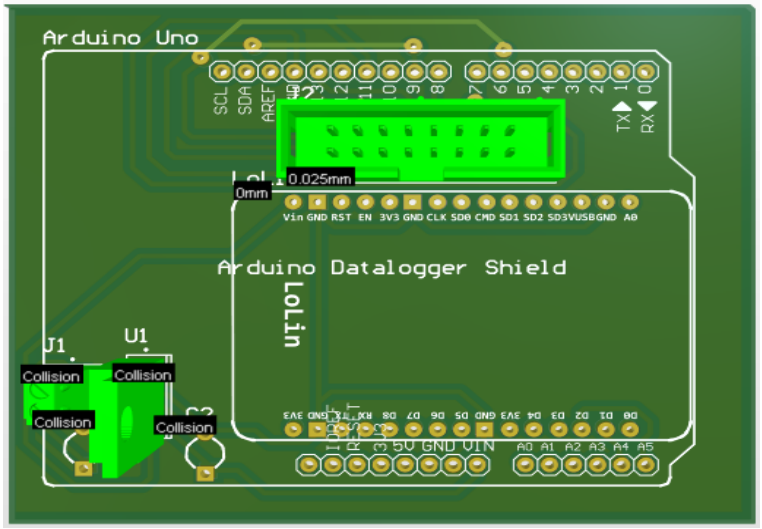


Figura 4.38: Representación en 3D de la placa de conexión maestro/esclavo

Programación de NodeMCU y Arduino Uno

La implementación de código para el control del panel LED se divide en dos partes que son la programación del maestro y la programación del esclavo, a continuación, se describen las funciones más importantes de cada uno; los códigos completos se los encuentran en el apéndice A.

- Inicialmente se procede a realizar la programación del NodeMCU. El procedimiento de definición del nombre de la Red, contraseña y dirección IP del *Brooker* MQTT, y, conexión del NodeMCU con la RPi es el mismo que en las Sub Secciones 4.1.1 y 4.1.2.

  - Primeramente se importan las librerías necesarias para la compilación del código, en donde, la librería "Wire.h"permite la comunicación entre el NodeMCU y el Arduino por medio del protocolo I2C.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <Wire.h>

```

  - La funcion *callback* se ve modificada, de manera que al obtener el mensaje publicado en el topic "panel", lo convierte en una cadena de caracteres y lo envía al Arduino por medio del puerto SDA

```
void callback(String topic, byte* message, unsigned int length)
{
  String messageTemp;

  for (int i = 0; i < length; i++)
  {
    messageTemp += (char)message[i];
  }
  Wire.beginTransmission(8); /* begin with device address 8 */
  Wire.write(messageTemp.c_str());
  Wire.endTransmission();    /* stop transmitting */
  delay(500);
}

```

- A continuación se detalla la implementación del código en el dispositivo esclavo, el cual contiene la fuente de letra con la que se mostrara el mensaje, mostrada en la Figura 4.34.

  - Inicialmente se importan las librerías necesarias para la compilación del código. La librería "DMD2.h"permite realizar la comunicación entre el Arduino y el Panel LED incluyendo la fuente desarrollada en el archivo "Tesis1.h".

    Adicionalmente se importa la librería "SPI.h"que permite la comunicación serial con el puerto HUB12 en base a la conexión de pines, como se muestra en la Figura 2.14.

```
#include <Wire.h>
#include <SPI.h>
#include <DMD2.h>
#include <fonts/Tesis1.h>

```

  - Se procede a implementar la función *receiveEvent* la cual lee la información proveniente del dispositivo maestro por medio del puerto SDA y la envía al panel LED por medio de la librería "*DMD*" usando el protocolo SPI.

```
1         void receiveEvent(int howMany)
2         {
3           String v;
4           while (0 <Wire.available())
5           {
6             char c = Wire.read();      /* receive byte as a character */
7             v += c;
8           }
9           const char *MESSAGE = v.c_str();
10          int s = strlen(MESSAGE);
11
12          //int x = s*10;
13          dmd.clearScreen();// clear screen
14          dmd.drawString(0, 0,MESSAGE);
15          //dmd.drawString(x, 0," LIBRE");
16          //dmd.drawString(25, 0," LIBRE");
17          //delay(1000); // letters speed changing
18        }
19
```

Implementación del módulo controlador de panel LED

Una vez que se ha realizado la implementación del código en los dispositivos maestro y esclavo, y se tiene la placa PCB impresa, se procede se procede a soldar los componentes y probar el funcionamiento del código y del módulo general.

En las figuras 4.25 y 4.40 se puede observar el prototipo del controlador del panel LED completamente armado, evidenciando el trabajo conjunto que realizan los dispositivos maestro y esclavo.

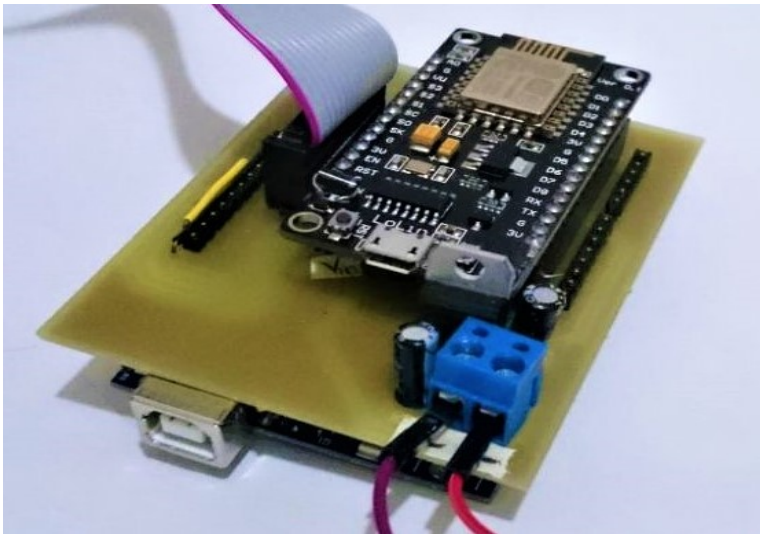


Figura 4.39: Módulo de control maestro/esclavo armado

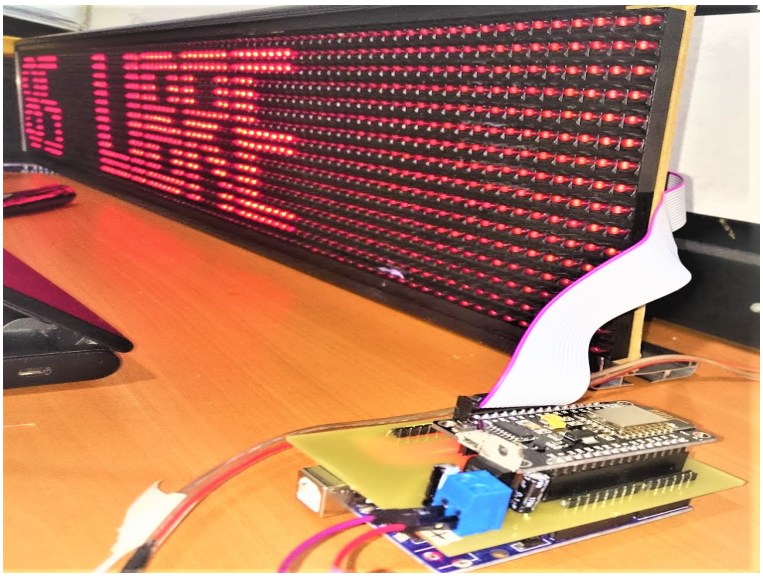


Figura 4.40: Pruebas de funcionamiento del módulo de control maestro/esclavo

Equipo Protector

En base a lo mencionado en la Sección 2.5, el panel LED con el que se desarrolla el presente trabajo de titulación , está confirmado por una matriz de 16X96 diodos LED tipo DiP, el cual cuenta con una estructura de metal que sostiene el sistema electrónico y lo protege de exposiciones a cambios ambientales y golpes.

Dado que se encontrará instalado en la intemperie, se lo ha robustecido con dos láminas de acrílico de 3mm instaladas en la parte frontal y posterior de la estructura lo cual impermeabiliza el sistema electrónico. Finalmente, el módulo controlador del panel es ubicado en el interior de la estructura con el objetivo de aprovechar el espacio y protegerlo de cualquier daño causado por la exposición, como se puede observar en la Figura 4.41.

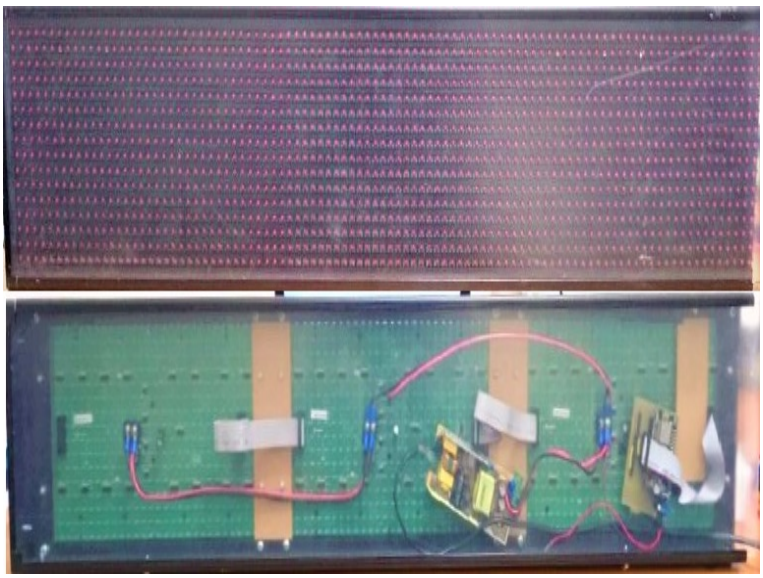


Figura 4.41: Protección del panel Led

4.1.4. Raspberry Pi

Para la implementación de esta tesis se utiliza una RPi con el sistema operativo Raspberry Pi OS, el cual se configurará con un servidor Mosquitto MQTT y servirá como concentrador de los datos.

Configuración como Servidor MQTT

Para la configuración del RPi como un *Broker MQTT mosquitto* se ingresa las siguientes líneas en la terminal.

- Se realiza la instalación del servidor Mosquitto en la terminal de comandos de la RPi

```
sudo apt update
sudo apt install -y mosquitto mosquitto-clients

```

- Se confirma la instalación tecleando *Y* y *enter*. Para que inicie el servidor Mosquito al iniciar el RPi se usa el siguiente comando.

```
sudo systemctl enable mosquitto.service

```

- El siguiente comando permite comprobar que la instalación fue satisfactoria.

```
 mosquitto -v

```

- Finalmente, como se observa en la Figura 4.42, se obtiene la versión, puerto y estado del servidor MQTT.

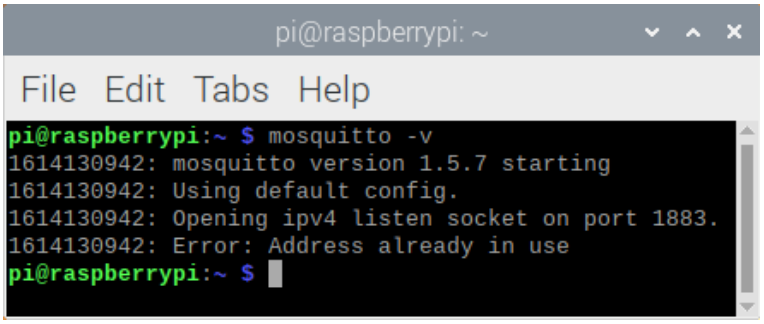


Figura 4.42: Respuesta al comando mosquitto -v cuando se completó la instalación correctamente

Implementación de Algoritmo de Detección

- Se definen las librerías necesarias para correr el programa.

```
import json
import time
import paho.mqtt.client as mqtt

```

- En las siguientes líneas de código se definen las funciones que se ejecutarán como parte del paquete MQTT. Dentro de estas funciones están primero, la configuración del cliente MQTT, luego la implementación de la función *on_connect* que se ejecuta al momento en que el script se conecta con el servidor MQTT. La función *on_message* se ejecuta cuando llega un mensaje al servidor, aquí es donde ejecutaremos el algoritmo principal de detección. Por último, se configura la dirección donde está corriendo el servidor y se declara el inicio del loop.

```
mqttc = mqtt.Client()
mqttc.on_connect = on_connect
mqttc.on_message = on_message
mqttc.connect("localhost", 1883, 60)
mqttc.loop_start()

```

- Se definen las variables que se usarán para almacenar los datos que arriban desde los sensores y los datos que son necesarios almacenar para el procesamiento futuro.

```
valor = 0
ref_libre = []
ref_sombra_media = []
ref_sombra_total = []
data_values = []
max_distance_index = 0
current_value = 0


```

- La siguiente función se ejecutará al momento de conectarse el programa al servidor MQTT, en este caso el módulo se subscribe al *topic datos_sensor* que utilizarán los módulos para publicar las lecturas de los sensores.

```
def on_connect(client, userdata, flags, rc):
    # el mismo topic 'datos_sensor' para todos los nodos
    client.subscribe('datos_sensor')

```

- Con motivo de optimizar el uso de la red se trabajó sobre un valor decimal siendo este modificado a nivel binario en donde reside la información pues se toma los índices de los valores binarios como los índices de los parqueaderos. Por este motivo se implementó la función *set_bit* para modificar el valor binario a nivel de bit. Este valor será enviado al módulo sensor para que pueda activar o desactivar los módulos LED de cada espacio de parqueadero dependiendo del valor del bit correspondiente.

```
def set_bit(value_to_modify, index, bit_to_set):
    mask = 1 << index
    value_to_modify &= ~mask
    if bit_to_set:
        value_to_modify |= mask
    return value_to_modify

```

- Al almacenar solamente el valor en decimal se implementó la función *binarizar*, con el objetivo de obtener los valores en binario y poder detectar en qué índice se encuentran disponibles los parqueaderos y poder publicar hacia el panel LED, siendo cero en binario para el parqueadero ocupado y un uno para uno libre.

```
def binarizar(decimal):
    binario = ''
    while decimal // 2 != 0:
        binario = str(decimal % 2) + binario
        decimal = decimal // 2
    return str(decimal) + binario

```

- La función *main* del programa de python se ejecutará vacía debido a que toda la lógica se encuentra en la función *on_message*

```
def main():
```

```
        while True:
            # time.sleep(2)
            _ = 0
if __name__ == "__main__":
    main()
```

- La función *get_max_distance_index* calcula el índice de los *arrays* de mediciones en donde tenemos la mayor apertura entre las medidas de voltaje en ambiente libre y ocupadas, de esta forma aseguramos ocupar el valor de resistencia que nos ayuda a diferenciar de mejor manera las dos curvas. Para lograr esto se inicia comprobando que hayan llegado los valores de las tres referencias. Luego se procede a calcular y comparar las distancias como se puede observar a en la Figura 4.43 para luego almacenar el índice y distancia de la mayor para utilizar en la siguiente repetición. Por último, se procede a retornar un 1 si el cálculo fue exitoso y un -1 si el cálculo falló por alguna razón.

```
def get_max_distance_index():
  global ref_libre
  global ref_sombra_media
  global max_distance_index
  global ref_sombra_total
  global data_values
  # inicializar variables primero desde nodo referencia
  if ((ref_libre != []) & (ref_sombra_total != []) & (ref_sombra_media != [])):
      distancia_previa = 0
      for index in range(len(data_values)):
          current_distance = abs(ref_sombra_total[index] - ref_libre[index])
          if current_distance > distancia_previa:
              max_distance_index = index
              distancia_previa = current_distance
      return 1
  else:
      return -1
```

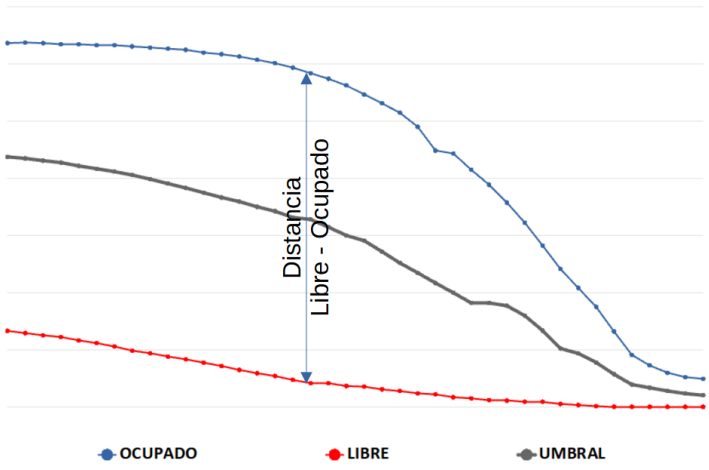


Figura 4.43: Imagen de las referencias

- La función *on_message* a continuación se ejecuta cuando llega un mensaje al nodo por medio del *topic* al que se ha suscrito. Como se observa en las líneas de código siguientes lo primero que hacemos es declarar las variables globales que se utilizan para manejar los datos. Luego obtenemos los datos que llegan en el *payload* del mensaje y se interpretan como un *frame* de datos. Luego identificamos si el *frame* llega del

nodo cero que corresponde al nodo de referencias, entonces se almacenan los datos en su correspondiente variable.

En caso de que los datos que llegan no sean del nodo cero, esto significa que llegan de un nodo sensor, para lo cual se recupera el valor decimal actual (*current_value*) que el nodo envía en la trama, este valor convertido a binario contiene la información sobre espacios de parqueadero ocupados y libres que actualmente está manejando el nodo. Este valor se va a modificar a nivel de bit dependiendo del resultado obtenido en el cálculo. Los cálculos siguientes se realizan utilizando el índice donde existe una mayor apertura entre los valores, este índice se calcula con la función *get_max_distance_index*.

En caso de que la función regrese un valor 1 se procede con el cálculo para detectar si los valores representan un parqueo libre u ocupado. Para tomar la decisión de si el parqueadero está libre u ocupado se compara el valor que llega en la trama con una media entre las referencias en donde se otorga el doble del peso a los valores de la referencia que se encuentra en sombra simulando el parqueadero ocupado.

En caso de que la medida supere el umbral, se toma como ocupado y se modifica el valor a publicar utilizando la función *set_bit* teniendo como base el valor *current_value*, de esta forma solo se cambia el bit en el índice que hace falta.

Cuando el valor no supera el umbral, el parqueadero se considera disponible, entonces se modifica el bit y se procede a publicar el valor final en el *topic* del nodo al que pertenece el mensaje.

Finalmente, una vez obtenido el valor se publica en el *topic* "*panel*" cuál de todos los parqueaderos está libre, para lo cual se usa la función *binarizar* que convierte el valor decimal en una cadena de bits, siendo posible obtener el índice del primer bit marcado en 1 que simboliza un parqueadero libre.

```
1    def on_message(client, userdata, message):
2      global ref_libre
3      global ref_sombra_media
4      global ref_sombra_total
5      global data_values
6      global max_distance_index
7
8      data_frame = json.loads(message.payload)
9
10     if (data_frame["node"] == 0):
11         if (data_frame["parking_index"] == 0):
12             ref_sombra_total = data_frame["data"]
13         if (data_frame["parking_index"] == 1):
14             ref_sombra_media = data_frame["data"]
15         if (data_frame["parking_index"] == 2):
16             ref_libre = data_frame["data"]
17     else:
18         value_to_publish = data_frame["current_value"]
19         data_values = data_frame["data"]
20         tmp = get_max_distance_index()
21         if tmp != -1:
22             i = max_distance_index
23             umbral = (ref_libre[i] + 2*ref_sombra_total[i] + ref_sombra_media[i])/4
24             if data_values[i] > umbral:
25                 # ocupado es binario 0
26                 value_to_publish = set_bit(value_to_publish, data_frame['parking_index'], 0)
```

```
27                else:
28                    # libre es binario 1
29                    value_to_publish = set_bit(value_to_publish, data_frame['parking_index'], 1)
30                topic = "nodo" + str(data_frame["node"])
31                value_to_publish_bin = binarizar(value_to_publish)
32                valor = '0'*(8 - len(str(value_to_publish_bin))) + value_to_publish_bin
33                parking_libre = 7 - valor.index('0')
34                client.publish(topic, value_to_publish)
35                client.publish('panel', 'A'+ parking_libre + ' Libre')
36
```

4.2. Sistema basado en visión artificial

Para la implementación del algoritmo principal de visión artificial que se muestra en el diagrama 4.44, se ubicó la cámara de tal manera que se obtuvo una visibilidad completa del parqueadero. Los autos se estacionan realizando un movimiento paralelo a la horizontal de la cámara con lo que es fácil identificar en qué parqueadero se encuentra maniobrando un automóvil detectando las diferencias en las posiciones en y respecto de puntos fijos en los estacionamientos.

Con esta información en mente se realiza una extracción de los contornos de los objetos en movimiento. Se realiza el seguimiento de todos los objetos utilizando la clase Object.

El algoritmo observa cuando un objeto pasa por la franja azul de la Figura 4.49. Cuando un objeto pasa por la franja, se almacenan las coordenadas y se ejecuta el método *parking_*, cuyo diagrama se muestra en la Figura 4.46, el cual calcula si el auto se está parqueando o está saliendo para cada *frame* y almacena el resultado en un *array*. Cuando el automóvil sale de la franja, se calcula cuál de los estados tuvo mayor incidencia y se calcula el nivel de confianza. En la Sección 4.2.2 se explica con mayor detalle la implementación del algoritmo de detección.

4.2.1. Posicionamiento de la cámara

La ubicación de la cámara debe cumplir dos condiciones importantes. Primero, la imagen capturada debe tener buena resolución para poder obtener las características de los objetos que se mueven en ella y segundo, debe capturar toda el área de interés, la cual incluye toda el área del parqueadero de la facultad de ingeniería.

Para cumplir con el primer requisito se consiguió una cámara web de 480x480 de resolución lo que nos permitió enfocar adecuadamente el área de interés. Para el segundo requisito se ubicó la cámara en la fachada de la facultad de ingeniería a la altura del techo del segundo piso con objetivo de obtener un ángulo para toda el área de interés como se observa en la Figura 4.47 obteniendo como resultado una visibilidad del área como se muestra en la Figura 4.48. A pesar de haber obtenido una amplia visibilidad, el lugar está lejos de ser el ideal debido a que el ángulo de inclinación genera oclusión entre vehículos. En la Figura 4.48 se muestra además el área de interés de los parqueaderos, en donde se va a calcular la disponibilidad de los mismos.

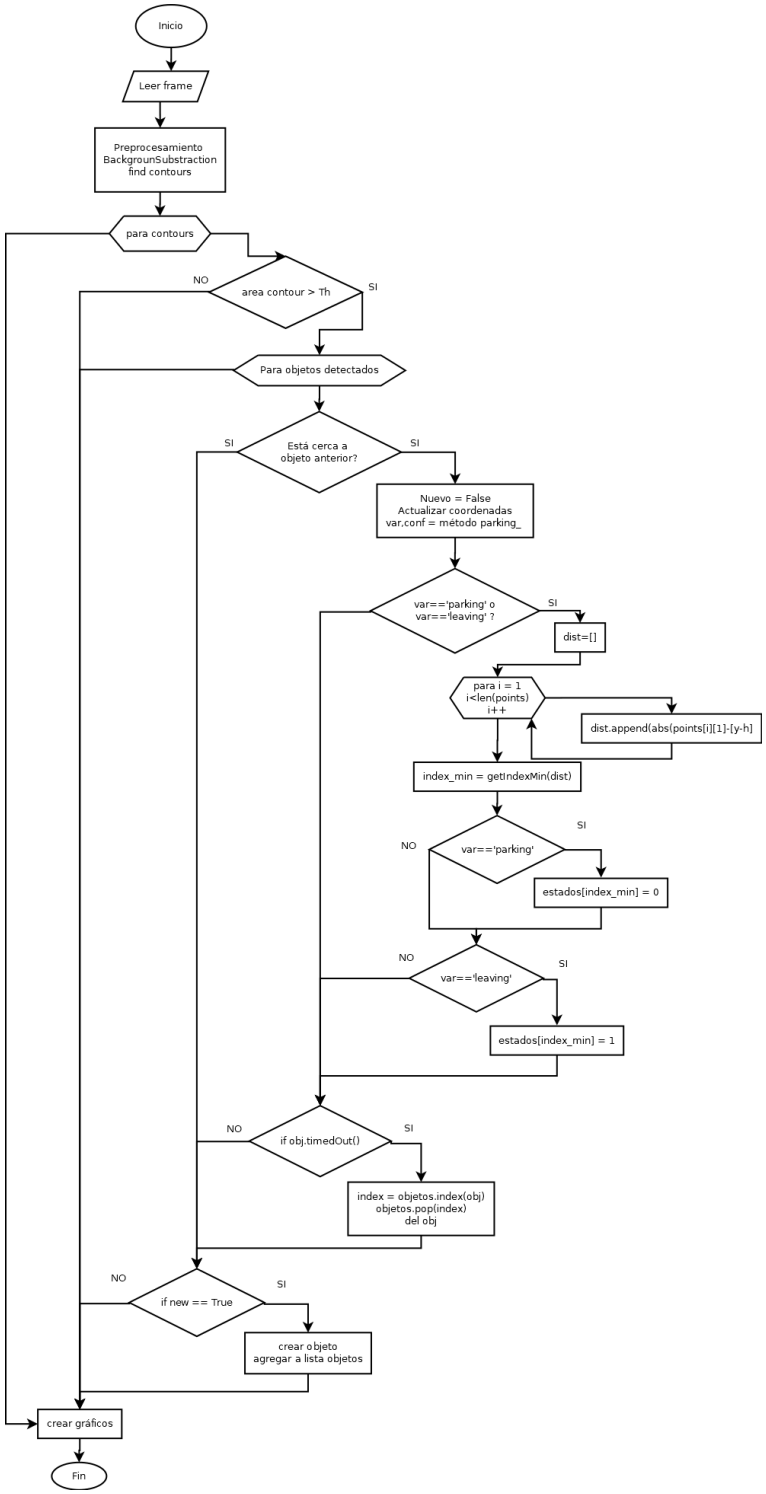


Figura 4.44: Diagrama de Flujo del programa principal

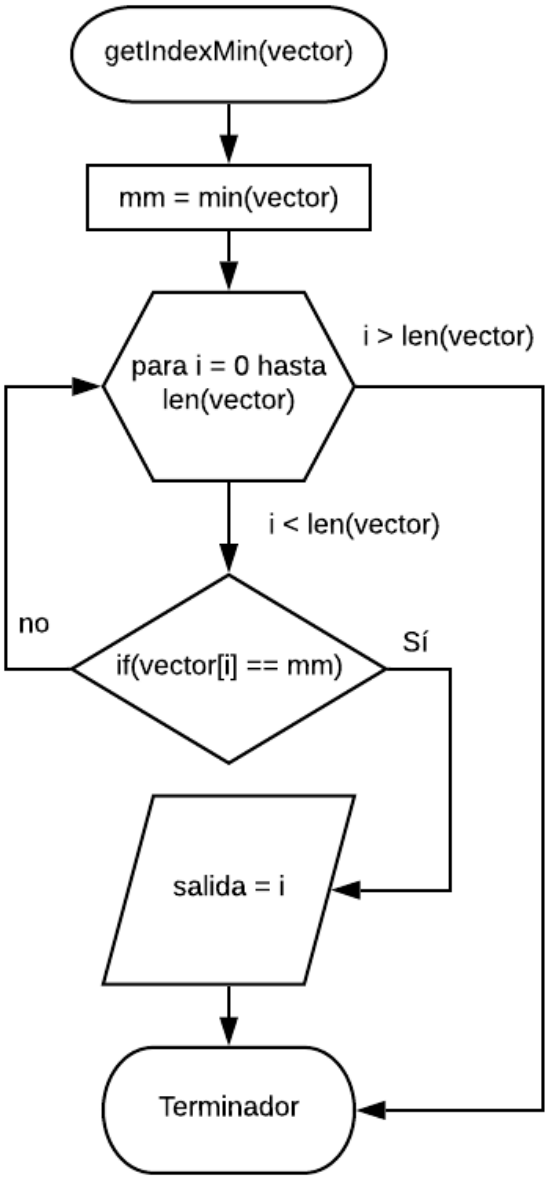


Figura 4.45: Diagrama de flujo de algoritmo para obtener el índice del valor mínimo

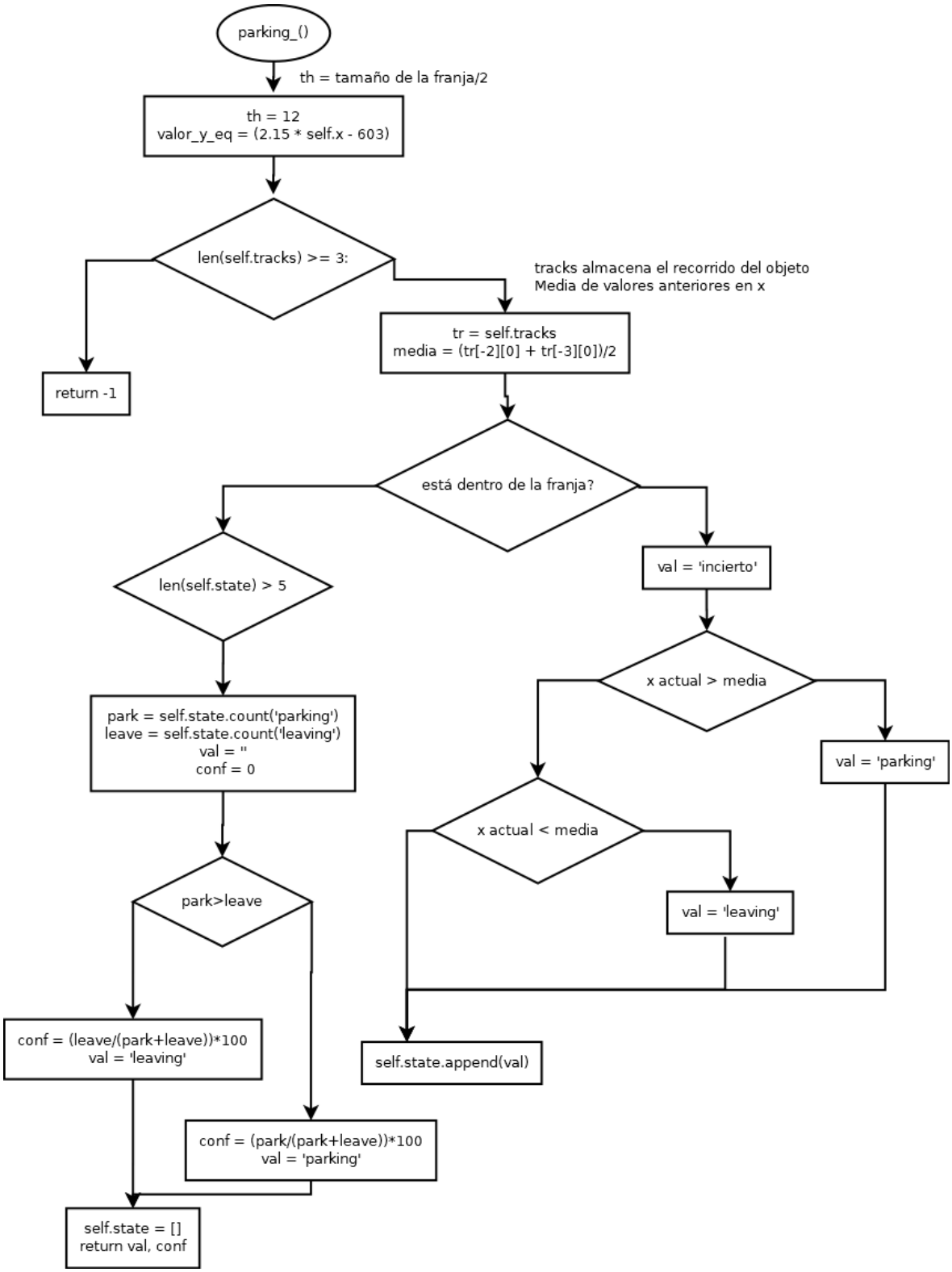


Figura 4.46: Diagrama de flujo del algoritmo de detección de estacionamiento o salida

Figura 4.47: Ubicación de la cámara respecto del parqueadero

Figura 4.48: Imagen del área de interés obtenida

4.2.2. Implementación de algoritmos de procesamiento de imágenes

El algoritmo de procesamiento de imágenes funciona como un complemento para el sistema de sensores en el cual detectamos la disponibilidad de los parqueaderos seleccionados usando herramientas disponibles en el paquete *OpenCV* de *Python*.

- Para este código se utilizaron los paquetes mostrados a continuación, de los cuales el paquete *Object* es una clase que se utilizará para llevar el registro de los objetos detectados. Los paquetes *numpy*, *math* y *cv2* contienen herramientas para trabajar con arrays, operaciones matemáticas y procesamiento de imágenes respectivamente.

```
import math
import numpy as np
from cv2 import cv2
import Object

```

- A continuación, en el código seguimos con la definición de la función *getIndexMin*, cuyo diagrama se muestra en la Figura 4.45, la cual nos ayuda a encontrar el índice del valor mínimo de un *array*.

```
def getIndexMin(array):
    mm = min(array)
    for i in range(len(array)):
        if array[i] == mm:
            return i

```

- Luego se definen variables a partir de la resolución del video para lo cual se lee un frame y se obtiene su ancho y alto. De esta forma podemos encontrar su área para a partir de esta encontrar un área umbral que deben ocupar como mínimo los objetos para ser detectados como automóviles

```
VIDEO_SOURCE = 'video2.mp4'
cap = cv2.VideoCapture(VIDEO_SOURCE)

ret, frame = cap.read()
h = frame.shape[0]
w = frame.shape[1]

frameArea = h*w
areaTH = frameArea/350

```

- Continuando con el código también se define un número máximo de edad que deben tener los objetos para eliminarlos, esta edad se calcula como la cantidad de *frames* que se han procesado. Se define también la variable donde se almacenarán los objetos, por último, se definen el kernel que se ocupará para realizar los procesamientos morfológicos sobre las imágenes del video.

```
max_p_age = 7
objetos = []
kernelOp = np.ones((3, 3), np.uint8)

```

- A continuación, se define un array de puntos los cuales serán útiles para detectar en qué estacionamiento se está ubicando el auto que entra o de cuál está saliendo. Cada par de valores representa las coordenadas en x y y de los puntos verdes indicados en la Figura 4.49

```
points = np.array([
    [420, 350],
    [401, 301],
    [385, 261],
    [368, 231],
    [359, 206],
    [349, 183],
    [341, 164],
    [332, 147],
])

```

Figura 4.49: Puntos marcados en el array

- A continuación, se definen e inicializan algunas variables entre ellas la variable que almacena el estado ocupado o libre de cada uno de los parqueaderos, se define una fuente a utilizar, un id de objeto que nos servirá para identificar a cada objeto como único y se define un color para utilizar en el texto.

```
    # (1 para libres 0 para ocupados)
    estados = [1, 1, 1, 1, 1, 1, 1, 1]  # 8 espacios inicializados en libres
    font = cv2.FONT_HERSHEY_SIMPLEX
    oid = 1
    text_color = (255, 0, 0)

```

- Se define el objeto que nos ayudará a calcular el BS. Para esto es necesario definir valores que permitan obtener una buena extracción del fondo para nuestra aplicación, se define un *history* de 500, un total de 3 *mezclas gaussianas*, un *backgroundRatio* de 0,5 para eliminar los movimientos lentos como de hojas de árboles manteniendo los movimientos de velocidades de tráfico normal. Por último, se configura un *noiseSigma* de 0 para obtener un valor automático.

```
    fgbg = cv2.bgsegm.createBackgroundSubtractorMOG(
        history=500, nmixtures=3, backgroundRatio=0.5, noiseSigma=0)

```

- A continuación, se implementa el *loop* principal, el cual se repite para cada imagen que se tome de la cámara. Inicialmente, se toma una lectura de la cámara y se realiza una copia para trabajar sobre ella. Se dibuja una línea que delimita la zona tránsito de los vehículos con la zona de parqueo como se muestra en la Figura 4.49, a continuación, se realiza un filtrado de ruido de la imagen usando el filtro Gaussiano con, kernel de 3X3 y σ=3, antes de aplicar el algoritmo de BS sobre la imagen para evitar el ruido. Las figuras 4.51a y 4.51b muestran dos imágenes una antes y otra después de aplicar el filtro gaussiano.

(a) Antes de aplicar el filtro gaussiano

(b) Después de aplicar el filtro gaussiano

Figura 4.50: Filtrado gaussiano

```
1     while True:
2         ret, frame = cap.read()
3         frameOriginal = np.copy(frame)
4
5         cv2.line(frameOriginal, (290, 70),
6                  (points[0][0], points[0][1]), (255, 0, 0), 5)
7
8         newFrame = cv2.GaussianBlur(
9             frameOriginal, (3, 3), sigmaX=3, sigmaY=3)
10        fgmask = fgbg.apply(newFrame)
11
```

(a) Antes de aplicar el algoritmo BS

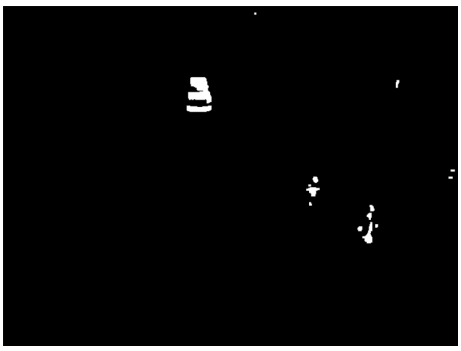
(b) Después de aplicar el algoritmo BS

Figura 4.51: Resultado del algoritmo Background Substraction

- Una vez aplicado el algoritmo de BS se realiza operaciones morfológicas para eliminar aún más el ruido generado debido a movimientos de la cámara o debido a los árboles que se mueven y generan pequeños puntos en la detección de objetos. En la primera operación morfológica se aplica un *MORPH_OPEN* con la ayuda de la función *morphologyEx* del paquete OpenCV con el objetivo de eliminar el ruido en la imagen resultante, el resultado se presenta en las figuras 4.52a y 4.52b. Luego se hace un *MORPH_CLOSE* para eliminar las manchas negras dentro de los objetos detectados, para esto se usa la función *getStructuringElement* del paquete *OpenCV* para general un kernel de 30x30 de tal manera que se puedan juntar las partes del mismo objeto como se muestra en las figuras 4.53a y 4.53b.

```
1     fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernelOp)
2     fgmask = cv2.morphologyEx(
3         fgmask, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (30, 30)))
4
```

(a) Antes de aplicar MORPH_OPEN



(b) Después de aplicar MORPH_OPEN

Figura 4.52: Resultado del Algoritmo MORPH_OPEN

(a) Antes de aplicar MORPH_CLOSE

(b) Después de aplicar MORPH_CLOSE

Figura 4.53: Resultado del Algoritmo MORPH_CLOSE

- A continuación se encuentran los contornos de los objetos detectados con la ayuda de la función *findContours*.

```
    contours, hierarchy = cv2.findContours(
        fgmask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

```

- Para cada uno de los contornos detectados se obtiene su área para filtrar los objetos pequeños, es decir se tomarán solamente los objetos que superen el área de umbral calculada anteriormente. Se dibujan los contornos de estas áreas, así como se dibuja también un punto rojo en su centro y se obtiene y dibuja un rectángulo que encierra el área del objeto.

```
    for i in range(len(contours)):
        area = cv2.contourArea(contours[i])
        if area > areaTH:
            color_contours = (0, 0, 255)
            # cv2.drawContours(drawing, contours, i,
            #                  color_contours, 1, 8, hierarchy)
            cv2.drawContours(
                frameOriginal, contours[i], -1, (255, 255, 255), cv2.FILLED)

            M = cv2.moments(contours[i])
            cx = int(M['m10']/M['m00'])
            cy = int(M['m01']/M['m00'])
            x, y, w, h = cv2.boundingRect(contours[i])
            cv2.circle(frameOriginal, (cx, cy), 5, (0, 0, 255), -1)
```

```
15
16              img = cv2.rectangle(frameOriginal, (x, y),
17                                  (x+w, y+h), (0, 255, 0), 2)
18
```

- A continuación, se realiza el análisis de transitoriedad para cada uno de los objetos detectados. En este punto se detectan los objetos y se comparan con los objetos detectados anteriormente para seguir el rastro para el mismo objeto o para crear uno nuevo cuando alguno entre en la escena.

  Como se muestra en las líneas de código siguientes, para cada objeto se comprueba si está cerca de un objeto detectado anteriormente en cuyo caso se toma como el mismo objeto. Luego se llama a la función *parking* ubicada en la clase *Object* para comprobar si el objeto está entrando a un parqueadero, saliendo de un parqueadero o solo está pasando. A continuación, se comprueba que el objeto se está estacionando o saliendo del parqueadero, en cuyo caso se modifica la variable *estados* que contiene los valores de espacios libres u ocupados. Luego se comprueba si la edad del objeto supera la edad límite, de ser así se elimina de la lista de objetos. En caso de que no se haya cambiado la bandera de objeto existente se crea un nuevo objeto y se agrega a la lista de objetos.

```
1               new = True
2               for obj in objetos:
3                   obj.age_one()
4                   if abs(x-obj.getX()) <= w + 5 and abs(y-obj.getY()) <= h + 5:
5                       new = False
6                       obj.updateCoords(cx, cy)
7                       var = obj.parking_()
8                       if(var == 'parking' or var == 'leaving'):
9                           dist = []
10                          for tempVal in range(len(points)):
11                              dist.append(abs(points[tempVal][1] - [y+h]))
12                          index_min = getIndexMin(dist)
13                          if var == 'parking':
14                              estados[index_min] = 0  # marca ocupado
15                          elif var == 'leaving':
16                              estados[index_min] = 1  # marca libre
17                          print(var + ' at ' + str(index_min))
18                          print(estados)
19
20                  cv2.putText(frameOriginal, str(obj.getId()), (obj.getX(), obj.getY()),
21                              font, 0.3, text_color, 1, cv2.LINE_AA)
22                  if obj.timedOut():
23                      # sacar obj de la lista objetos
24                      index = objetos.index(obj)
25                      objetos.pop(index)
26                      del obj  # liberar la memoria de obj
27              if new == True:
28                  p = Object.MyObject(oid, cx, cy, max_p_age)
29                  objetos.append(p)
30                  oid += 1
31
```

4.2.3. Clase diseñada para el análisis transitorio

Para realizar el análisis transitorio de los objetos detectados se diseñó la clase *Object* la cual se describe a continuación.

- Primero empezamos definiendo la clase y su inicializador con los parámetros que se ocuparán para caracterizar un objeto detectado, entre ellos necesitamos un identificador único de objeto, las coordenadas

del centro del objeto, un vector de los puntos por los cuales ha pasado el objeto, la propiedad *done* para denotar que el objeto cumplió su ciclo de vida, el vector de estado para almacenar los cálculos de entrada (*parking*) y salida (*leaving*), la edad del objeto y por último la edad máxima que puede tener el objeto antes de ser desechado.

```
class MyObject:

    def __init__(self, i, xi, yi, max_age):
        self.i = i
        self.x = xi
        self.y = yi
        self.tracks = []
        self.done = False
        self.state = []
        self.age = 0
        self.max_age = max_age

```

- Luego se definen las funciones que permitirán obtener la identificación y posición.

```
def getId(self):
    return self.i

def getX(self):
    return self.x

def getY(self):
    return self.y

```

- A continuación, se define la función *updateCoords* la cual obtiene como entradas las coordenadas del punto central del objeto, se utiliza estos valores para obtener la media móvil con el valor anteriormente almacenado con objetivo de filtrar el ruido y lo guarda al final del vector *tracks*. Se define también el *getter "timedOut"* que devuelve el valor de la variable *done* la que nos permite saber si el objeto aún se encuentra en la escena. Por último, se define la función *age_one* la que aumenta la edad del objeto y comprueba si se ha superado la edad máxima para marcar en *True* la variable *done*.

```
def updateCoords(self, xn, yn):
    self.age = 0
    if len(self.tracks) > 0:
        x_media = round((self.x + self.tracks[-1][0]) / 2)
        y_media = round((self.y + self.tracks[-1][1]) / 2)
    else:
        x_media = xn
        y_media = yn
    self.tracks.append([x_media, y_media])
    self.x = xn
    self.y = yn

def timedOut(self):
    return self.done

def age_one(self):
    self.age += 1
    if self.age > self.max_age:
        self.done = True
    return True

```

- Por último, para la clase Objeto se define el método *parking_* el cual nos ayuda a calcular el estado de un auto que se está moviendo. Los posibles estados son *parking*, *leaving* e incierto, existe un estado extra *´1´*, el cual se devuelve cuando se encuentra un error en los datos o en los cálculos y no se logró obtener un

estado definido.

Para detectar los estados primero se define una función que marca la recta entre el punto inicial y el punto final de la región de interés del lado donde entran y salen los vehículos representada con la línea azul de la Figura 4.49.

En caso de que se tengan más de tres valores almacenados en la variable *tracks* se procede a obtener una media de los dos penúltimos valores con objetivo de reducir el ruido aleatorio generado por el centro de los contornos. Este valor se compara con el último valor del vector *tracks* en caso de que este sea mayor a la media obtenida significa que el objeto está moviéndose hacia la derecha por lo que se está estacionando, caso contrario se está moviendo hacia la izquierda por lo que debe estar saliendo del parqueadero. Este resultado se almacena al final del vector *state*. Una vez que el objeto sale de la franja se utiliza este vector para decidir si el objeto salió o entró al parqueadero.

En caso de que el objeto detectado no se encuentre dentro de la franja se comprueba si el vector de estado tiene más de 5 valores, esto nos puede indicar que ya pasó por la franja lo que permite disparar el algoritmo de decisión.

Para el algoritmo de decisión se cuentan los *parking* y *leaving* almacenados en el vector *state* y se retorna el estado que tenga el mayor número de incidencias. Por último, se calcula la razón entre la instancia con mayor número de incidencias y el total para estimar con qué nivel de confianza podemos decir que se llegó a la respuesta correcta.

```
1     def parking_(self):
2         th = 12
3         valor_y_eq = (2.15 * self.x - 603)
4         if len(self.tracks) >= 3:
5             tr = self.tracks
6             media = (tr[-2][0] + tr[-3][0])/2
7             # dentro de la franja de la linea
8             if (self.y >= valor_y_eq - th) & (self.y <= valor_y_eq + th):
9                 if self.tracks[-1][0] > media:  # va hacia la derecha
10                    val = 'parking'
11                elif self.tracks[-1][0] < media:  # va hacia la izquierda
12                    val = 'leaving'
13                else:
14                    val = 'incierto'
15                self.state.append(val)
16            elif(len(self.state) > 5):  # posiblemente fuera de la franja de la line
17                park = self.state.count('parking')
18                leave = self.state.count('leaving')
19                val = ''
20                conf = 0
21                if (park > leave):
22                    conf = (park/(park+leave))*100
23                    val = 'parking'
24                else:
25                    conf = (leave/(park+leave))*100
26                    val = 'leaving'
27                self.state = []
28                return val, conf
29        return '-1'
30
31
```

4.2.4. Integración de Subsistemas de WSN y PDI

Los dos subsistemas se diseñaron para trabajar conjuntamente. Durante el funcionamiento de los algoritmos se determinó que se publicará el valor del parqueadero libre calculado por el algoritmo de sensores. Si por alguna razón cualquiera de los módulos deja de funcionar se publicarán los valores calculados por el algoritmo de procesamiento de imágenes como se detalla en el código a continuación.

Iniciamos importando y configurando la computadora como cliente MQTT en el mismo script de python que se utilizó para el análisis de video así como también definimos variables que nos servirán de banderas para publicar en el panel LED y una variable para controlar el valor a publicar.

```
mqttc = mqtt.Client()
mqttc.on_connect = on_connect
mqttc.on_message = on_message
mqttc.connect("192.168.1.100", 1883, 60)
mqttc.loop_start()

is_published_1 = False
is_published_2 = False
parqueo_libre = 0
```

La variable *parqueo_libre* es configurada con el valor correspondiente al menor índice calculado durante la detección del lugar de parqueo, mientras que la función *on_connect* ejecuta el código de subscripción hacia el tópico necesario para recibir los datos de los módulos de sensores y referencia.

```
def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    # el mismo topic 'datos_sensor' para todos los nodos
    client.subscribe('datos_sensor')
```

Cada que llega un mensaje desde los sensores se ejecuta la función *on_message*, entonces se actualiza el tiempo de llegada del último mensaje para poder comprobar si se está recibiendo los paquetes.

```
def on_message(client, userdata, message):
    global ref_node_epoch
    global sensor_node_epoch

    data_frame = json.loads(message.payload)

    if (data_frame["node"] == 0):
        ref_node_epoch = time.time()
    elif(data_frame["node"] == 1):
        sensor_node_epoch = time.time()
```

Para coordinar si se mandará o no paquetes hacia el panel desde el algoritmo de procesamiento de video se comprueba si están llegando paquetes desde los sensores. En caso de que no se reciban paquetes durante más de diez segundos significa que no están funcionando los módulos por lo tanto se procede a publicar valores en el panel. Si el valor a publicar es de cero, es decir no hay espacios disponibles, se envía la hora hacia el panel, caso contrario se publica el valor almacenado en la variable *parqueo_libre* que es donde se almacena el valor del último parqueadero que se desocupó.

```
        now = time.time()
        val_ref = (now - ref_node_epoch)
```

```
        val_sensor = (now - sensor_node_epoch)

        if ((val_ref > 10) \| (val_sensor > 10)):
            # publicar el valor al panel.
            if(value_to_publish == 0):
                if(round(val_ref) % 5 == 0 | round(val_sensor) % 5 == 0):
                    if(not is_published_1):
                        is_published_1 = publish_date(is_published_1)
                else:
                    is_published_1 = False
            else:
                if(round(val_ref) % 5 == 0 | round(val_sensor) % 5 == 0):
                    if(not is_published_2):
                        print('D' + str(parqueo_libre) + ' LIBRE')
                        mqttc.publish(
                             'panel', 'D' + str(parqueo_libre) + ' LIBRE')
                        is_published_2 = True
                else:
                    is_published_2 = False
```

4.3. Análisis económico

En esta sección se presenta un valor aproximado de los elementos que conforman cada uno de los módulos, es decir, el precio aproximado de la implementación. Los precios de la impresión de las placas es un valor aproximado, debido a que son una colaboración de la Red Sísmica del Austro.

Inicialmente, en la Tabla 4.1 se puede observar el precio de cada uno de los elementos que componen el módulo de control del panel LED. La lista de materiales se la ha detallado en base a los elementos descritos en la Figura 4.36, a cuál tiene un valor aproximado de $40.

Tabla 4.1: Lista de materiales y precios del módulo controlador del panel LED

Materiales	**Designación**	**Cantidad**	**Precio Unitario**	**Subtotal**
Capacitor 0.1uf	C2	1	0,10	0,10
Capacitor 0.22uf	C1	1	0,10	0,10
Terminal Block Alimentacion	J1	1	0,20	0,20
Conector Macho 2x8 - HUB12	J2	1	0,30	0,30
Peineta Hembra	LoLin-1	2	0,75	1,50
LM7809	U1	1	0,50	0,50
ARDUINO UNO	ARDUINO UNO	1	15,00	15,00
NodeMCU	NodeMCU	1	15,00	15,00
Peineta Macho	Shield Arduino	1	1,00	1,00
Placa impresa	Placa impresa	1	5,00	5,00
			TOTAL	38,70

A continuación, en la Tabla 4.2 se muestran los precios de cada uno de los elementos que componen el módulo comparador. La lista de materiales se la ha detallado en base a los elementos descritos en la Figura 4.8, la cual alcanza un valor aproximado de $30.

Tabla 4.2: Lista de materiales y precios del módulo comparador

Materiales	Designación	Cantidad	Precio Unitario	Subtotal
NodeMCU	NodeMCU	1	15,00	15,00
Capacitor 0.33uf	C1	1	0,10	0,10
Capacitor 0.1uf	C2, C4	2	0,10	0,20
Capacitor 0.22uf	C3	1	0,10	0,10
Terminal Block Alimentación	J1, J2, J3, J4	4	0,20	0,80
Peineta Hembra	LoLin-1, P1, P2	2	0,75	1,50
LM7809	U1	1	0,75	0,75
LM1117	U2	1	1,00	1,00
LM741	U3	1	0,35	0,35
Socket 8 Pines	U3	1	0,10	0,10
Multiplexor CD4051BE	U4, U7	2	0,65	1,30
Socket 16 Pines	U4, U7	2	0,10	0,20
Placa impresa	Placa impresa	1	8,00	8,00
			TOTAL	29,40

Por consiguiente, se puede observar en la Tabla 4.3 los precios de cada uno de los elementos que componen el módulo sensor y la placa de control de los módulos LED. La lista de materiales alcanza un valor aproximado de $65 y se la ha detallado en base a los elementos descritos en las figuras 4.18 y 4.22.

Finalmente, en la Tabla 4.4 se especifican distintos valores que pertenecen a los diferentes elementos necesarios para la instalación de todo el prototipo, los cuales son protecciones, herramientas, elementos para la adquisición de video y procesamiento de la información, lo que tiene un valor cercano a $350.

Tabla 4.3: Lista de materiales y precios del módulo sensor

Materiales	Designación	Cantidad	Precio Unitario	Subtotal
Capacitor 0.33uf	C1	1	0,10	0,10
Capacitor 0.1uf	C2, C4	2	0,10	0,20
Capacitor 0.22uf	C3	1	0,10	0,10
Terminal Block Alimentación	J1, J2	2	0,20	0,40
Terminal Block LDRs	J3, J4, J6, J7	4	0,20	0,80
Mólex Macho 8 Pines	J5, J10	2	0,45	0,90
Peineta Hembra	LoLin-1, P1, P2	2	0,75	1,50
Resistencia 1/4 W - 470 OHM.	R1 - R18	18	0,03	0,54
LM7809	U1	1	0,50	0,50
LM1117	U2	1	0,75	0,75
LM741 - 8 Pines	U3	1	0,35	0,35
Socket - 8 Pines	U3	1	0,10	0,10
Multiplexor CD4051BE	U4, U7	2	0,65	1,30
Socket 16 Pines	U4, U5, U7, U8	4	0,10	0,40
Registro 74LS595 o 74HCT595	U5, U8	2	1,00	2,00
UNL2803	U6, U9	2	0,80	1,60
Socket 18 Pines	U6, U9	2	0,10	0,20
NodeMCU	NodeMCU	1	15,00	15,00
Módulos LED	Módulo Led	20	0,65	13,00
Sensores LDR	LDR	10	1,00	10,00
Placa impresa	Placa impresa	1	15,00	15,00
			TOTAL	64,74
Placa de Control de Módulos LED				
Placa impresa	Placa impresa	1	2,00	2,00
Mólex Macho 8 Pines	J1, J6	2	0,45	0,90
Terminal Block 1x2 LEDs	J2, J3, J4, J5, J7, J8, J9, J10	8	0,20	1,60
			TOTAL	4,50

Tabla 4.4: Lista de materiales y precios para la instalación

Materiales	Designación	Cantidad	Precio Unitario	Subtotal
Cámara	Cámara	1	40,00	40,00
Raspberry Pi 3	Raspberry Pi 3	1	60,00	60,00
Panel Led P10- Rojo	Panel Led	1	70,00	70,00
Cable de 3 Pares (100m)	Multipar	1	31,00	31,00
Cable de Timbre	Cable de Timbre (m)	15	0,20	3,00
Silicona para Asfalto	Tubos Silicona	3	1,25	3,75
Extensión 110V	Extensión (m)	40	0,65	26,00
Canaletas Rectangulares (2m)	Canaleta Rectangular	10	0,75	7,50
Canaleta de Alfombra (2m)	Canaleta Alfombra	1	4,20	4,20
Laminas Protección Panel LED 18cm X 98cm	Acrílicos Panel	2	10,00	20,00
Protecciones Módulos LED	Protecciones LED	6	2,16	12,96
Protecciones LDRs	Protecciones LDR	6	2,16	12,96
Caja de Paso (40cm X 35cm)	Caja de Paso	1	14,50	14,50
Prensaestopas plásticos PG21	Prensaestopas	3	1,00	3,00
Protección Módulo Comparador	Protección Módulo C.	1	40,00	40,00
Amarraderas de Plástico	Abrazaderas	1	2,05	2,05
			TOTAL	350,92

4.4. Conclusiones

El desarrollo de cada uno de los módulos pertenecientes al sistema WSN ha evidenciado un proceso detallado y paulatino, debido a que el funcionamiento de todo el sistema se encuentra distribuido e interconectado.

Como se puede observar en la Sub Sección 4.1.1, la parte principal de módulo comparador es el dimensionamiento de los valores de resistencias puesto que es necesario modelar estadísticamente los diferentes escenarios posibles para obtener una detección adecuada. Después de un análisis minucioso de cada una de las curvas de comportamiento de los sensores LDR descritas en las figuras 4.3, 4.4 y 4.5 se fijaron los ocho valores de resistencia, los cuales son 470Ω, 1KΩ, 3.3KΩ, 10KΩ, 33KΩ, 56KΩ, 82KΩ y 100KΩ. Las resistencias con valores entre 470Ω y 3.3KΩ permitirán una detección adecuada en un ambiente soleado, mientras que los valores entre 1Ω y 56KΩ permitirán una detección adecuada en un ambiente nublado y los valores superiores a 33KΩ permitirán una detección adecuada en la noche.

Además, el diseño basado en los diferentes criterios de EMC mencionados en las Sub Secciones 4.1.1 y 4.1.2 ha permitido reducir considerablemente a susceptibilidad de los circuitos electrónicos con respecto a cualquier tipo de fuente de EMIs, siendo las características más importantes la selección adecuada de los circuitos integrados cuyas características son detalladas en 2.6 y la presencia de planos de masa, puesto que sirven para el retorno de cualquier tipo de señal no deseada. De la misma forma, en las figuras 4.10, 4.21, 4.24 y 4.38 se puede evidenciar la ubicación de los elementos electrónicos en base a su funcionamiento, mostrando un distanciamiento adecuado de los mismos con el objetivo que las pistas tengan la menor distancia posible evitando generar esquinas que pueden convertirse en puntos de radiación.

Dado que el sistema de WSN será instalado en la intemperie, es necesario protegerlo adecuadamente contra los cambios climáticos y golpes, para ello se han diseñado y seleccionado protecciones que cumplan con los estándares de *Ingress Protection* mencionados en la Sección 2.9. Para proteger el módulo comparador se diseñó e implementó la carcasa que se muestra en la Figura 4.17 la cual fue impresa con filamento PETG y adecuada con un conector prensaestopa, que impermeabilizan el módulo entero. De la misma forma, puesto que los módulos LED y sensores LDR estarán instalados en el asfalto, las protecciones para estos que se observan en las figuras 4.28 y 4.30 fueron impresas con filamento PETG y rellenadas con resina de poliuretano con el objetivo de resistir alto tráfico y cambios climáticos, en especial la exposición al agua y peso de las llantas de los autos. La protección del panel LED se la puede observar en la Figura 4.41 en donde, las láminas de acrílico le dan la impermeabilidad necesaria a la estructura general, manteniendo selladas las uniones con silicona. Finalmente, los elementos que componen y alimentan al módulo sensor se encuentran ubicados en una caja estanca la cual cumple con el estándar IP65, como se puede observar en la Figura 4.31.

El sistema de procesamiento de imágenes consta de una WebCam, la cual se encuentra ubicada en el tercer piso de la facultad de ingeniería con el objetivo de abarcar la mayor parte del estacionamiento de la facultad, incluida el área de interés a detectar, como se puede observar en las figuras 4.47 y 4.48. El algoritmo de PDI implementado se lo detalla en la Sección 4.2.2 el cual ha sido desarrollado como un algoritmo adaptativo en base a BS y TA, primero se procede a difuminar la imagen por medio de un filtro gaussiano eliminando el ruido presente en la imagen, a continuación se utiliza el algoritmo de BS mediante a implementación de la función MoG que proporciona una imagen en mapa de bits, en donde los objetos en movimiento tendrán color blanco y

los el resto de la imagen está en color negro. Se realizan las operaciones morfológicas de ”apertura” y ”cierre”, descritas en la Sección 2.11, en la imagen obtenida por BS con el objetivo de eliminar la información irrelevante y rellenar de color blanco las áreas en negro que están dentro de los objetos detectados, creando un solo cuerpo para su respectiva segmentación.

Una vez realizada la adecuación de la imagen, se realiza el proceso de TA en donde se genera un rectángulo alrededor del área del objeto, facilitando el seguimiento del mismo. Se implementa una línea umbral en la imagen, el cual será usado como indicador para detectar la dirección de movimiento del objeto y así definir si un auto está estacionándose o saliendo del parqueadero. Se define en una matriz, los píxeles pertenecientes al límite de cada una de las líneas amarillas dentro la región de interés, que se observa en la Figura 4.48, la cual servirá para calcular la distancia entre el área del objeto y el píxel definido; el estacionamiento ocupado corresponderá el vector con la menor distancia de todos.

Capítulo 5

Pruebas y resultados

En el presente capítulo se detalla el proceso de pruebas de funcionamiento dentro del laboratorio y la respectiva instalación de cada uno de los módulos desarrollados en sus puestos de trabajo. Se muestran los resultados de las pruebas de funcionamiento de los subsistemas de WSN y PDI por separado, en base a diferentes ambientes y escenarios; estos resultados son comparados con el objetivo de obtener una valoración de la eficacia de cada uno de los subsistemas.

5.1. Pruebas de laboratorio

Una vez que se ha realizado el desarrollo físico de los módulos que componen el prototipo del sub sistema WSN se realizan las pruebas pertinentes para evidenciar el funcionamiento esperado de cada uno de los módulos y realizar las correcciones necesarias en caso de existir. En la Figura 5.1 se evidencia su correcto funcionamiento.

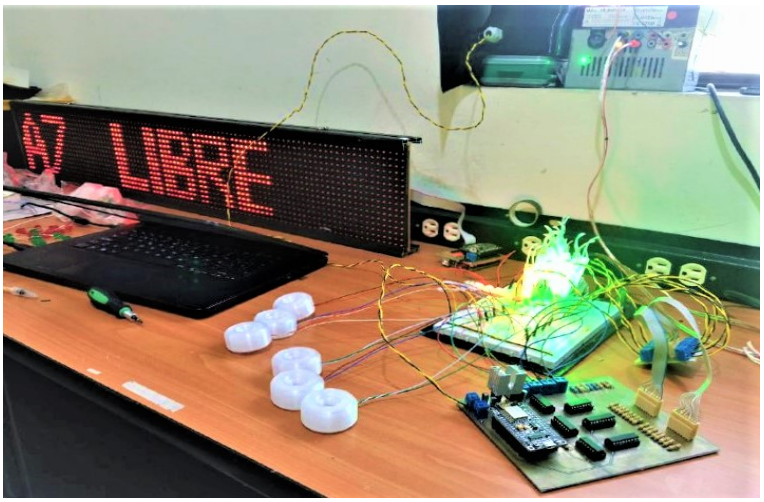


Figura 5.1: Pruebas en laboratorio de sub sistema WSN

Se puede observar la existencia de cada uno de los módulos que componen el sub sistema de WSN y su funcionamiento esperado. El módulo comparador de la Figura 4.17 está ubicado en un lugar que cuenta con luz natural, al mismo tiempo se utilizan los sensores LDR con sus respectivas protecciones mostradas en la Figura 4.28 y mediante el proceso de censado que realiza el módulo sensor los LEDs indican que todos los supuestos puestos de parqueo se encuentran libres, mientras que el panel LED muestra el nombre de cada uno de ellos.

Para las pruebas pertinentes del sub sistema de procesamiento de imágenes, se trabaja con un video capturado por medio de la cámara ubicada conforme se muestra en la Figura 4.47. Se utiliza un vector de ocho elementos que indica el estado de cada parqueo en base a su índice respectivo, así, para la prueba inicial se han establecido en cero todos los elementos, es decir, como si todos los parqueos estuvieran ocupados. Como se menciona en la sección 4.2.4, si todos los parqueaderos se encuentran ocupados el subsistema de procesamiento de imágenes mostrará la fecha y hora actual a través del panel LED como se puede apreciar en la Figura 5.2.

Figura 5.2: Pruebas de laboratorio de sub sistema PDI - Fecha y Hora

A continuación, en la Figura 5.3 se puede observar la línea dibujada que servirá para el análisis transitorio mencionado en la sub sección 4.2.3, mediante la cual se reconoce si un auto se está estacionado o está desocupando el parqueo. Adicionalmente, se puede observar los puntos definidos de cada estacionamiento con color verde que sirven para identificar el parqueo que está siendo ocupado o desocupado.

El vector de 8 elementos que inicialmente se encontraba en cero, después que un auto deja su estacionamiento, el vector indica que la posición 6 se encuentra desocupada por medio del valor ”1” junto con el porcentaje de confianza de la detección.

Figura 5.3: Pruebas de laboratorio de sub sistema PDI

Finalmente, en base a la implementación detallada en la sub sección 4.2.4, el Raspberry Pi identifica el nombre del parqueo libre, reporta el parqueadero libre hacia el módulo del panel LED para que este se comunique mediante I2C con el Arduino UNO el que a su ves controla el panel LED para mostrar los resultados tal como se puede evidenciar en la Figura 5.4.

Figura 5.4: Pruebas de laboratorio de sub sistema PDI - Detección del Parqueo

5.2. Instalación del Prototipo de Sistema Autónomo de Detección de parqueos libres

El módulo sensor contempla una sección de instalación de cableado desde la placa donde se encuentra el NodeMCU hasta el lugar de cada uno de los parqueaderos como se muestra en la imagen 5.5. Este módulo realiza un barrido de lecturas sobre los LDRs de cada parqueadero con cada una de las resistencias que luego se empaquetan y envían mediante Wi-Fi a la Raspberry PI para realizar su procesamiento.

El Raspberry Pi corre el algoritmo de decisión en base a los datos recibidos desde el módulo de referencias. Una ves decidido si el parqueadero de donde recibió los datos está libre u ocupado el Raspberry reporta por Wi-Fi los resultados al módulo de sensores para que este pueda variar los valores de los módulos LED instalados de con cable desde el módulo sensor ubicado en el poste hacia las esquinas de cada uno de los parqueaderos.

En base a lo mencionado en la sección 1.3, el prototipo del sub sistema de WSN diseñado para el presente proyecto experimental tiene como objetivo ser instalado en un espacio de ocho parqueaderos, en los que se instalarán los sensores LDR y módulos LED de las figuras 4.28 y 4.30 por medio de cortes realizados en el asfalto. En la Figura 5.5 se puede observar los parqueaderos asignados y las dimensiones de los cortes necesarios para la instalación de los componentes de la red de sensores, al igual que la ubicación de los mismos.

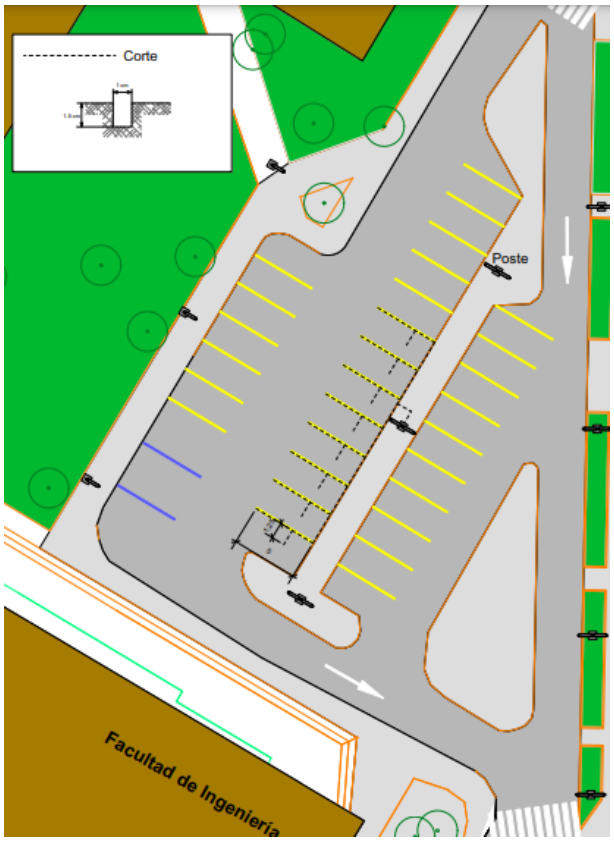


Figura 5.5: Planificación de cortes en el asfalto

Debido a que los cortes en el asfalto del parqueadero están sujetos a los permisos pertinentes por parte de la Universidad, se realiza la intervención de los cortes para la instalación del prototipo en sólo cuatro parqueos, mostrándose en la Figura 5.6 el proceso de instalación del cableado.

Figura 5.6: Instalación de subsistema de Sensores

Finalmente, en la Figura 5.7 se puede observar el subsistema de sensores completamente instalado con todos sus componentes. El módulo comparador está ubicado encima del panel LED que indica el siguiente parqueo libre para ser ocupado, y debajo de éste, se encuentra el módulo sensor dentro de la caja de paso al cual se conectan los sensores LDR y módulos LED instalados en cada parqueadero.

Figura 5.7: Subsistema de sensores instalado

Por seguridad, el raspberry y el router utilizados para la comunicación y procesamiento de los datos se los ubican cerca de la cámara conforme se muestra en la Figura 4.47.

5.2.1. Pruebas de funcionamiento del sistema autónomo de detección de parqueos libres

Una vez realizada la instalación de los subsistemas, se comprueba su funcionamiento en los ambientes soleado, nublado, lluvioso y nocturno mencionados en la sub sección 4.1.1. La visualización de la detección por parte de la WSN se basa en activar cada módulo LED con color verde si el parqueo esta libre y con rojo si se encuentra ocupado, mientras que el subsistema de PDI muestra el parqueo a ocupar por medio del panel LED siempre y cuando no se encuentre activo el subsistema WSN.

Como se menciona en las sub secciones 4.1.4 y 4.2.4 los subsistemas indicarán el nombre del parqueadero libre con el objetivo de canalizar ágilmente el tráfico en el estacionamiento. En la Figura 5.8 se puede observar

la notación utilizada para indicar el parqueo disponible,

Las pruebas de funcionamiento reales en base al subsistema de PDI se las realiza hasta las 18h30, dado que el personal administrativo se retira de las instalaciones a partir de las 18h00.

Figura 5.8: Parqueo libre desde el panel LED

Adicionalmente, en el caso que todos los parqueos se encuentren ocupados no se mostrará nada más que la fecha y hora actuales tal como se puede observar en la Figura 5.9. A continuación, se presentan los resultados de las pruebas realizadas.

Figura 5.9: Fecha y hora desde el panel LED

Pruebas en ambiente soleado

El análisis del funcionamiento de los subsistemas en un ambiente soleado se lo presenta con el objetivo de determinar la visibilidad de la señalización de los parqueos y su exactitud de detección. Como se mencionó en la sección 4.1.2 el censado de cada parqueadero se lo realiza cada 5 segundos, garantizando el estado detectado de cada parqueo.

- **Subsistema de Sensores**
  En la Figura 5.10 se puede observar que el parqueadero 4 se encuentra ocupado, siendo evidentes los LEDs de color verde indicando la disponibilidad de los parqueos 1 al 3.

Figura 5.10: Parqueo 4 ocupado

De la misma forma, en la Figura 5.11 se puede observar que los parqueaderos 2 y 4 se encuentran disponibles. Así, se comprueba que el funcionamiento es el esperado en un día soleado. Los módulos LED están ubicados en el límite de las líneas de parqueo facilitando su visualización.

Figura 5.11: Parqueos 1 y 4 ocupados

- **Subsistema de visión artificial**
  El subsistema de PDI se basa en la detección del movimiento por medio de TA, el cual permite identificar el parqueadero que está siendo ocupado o desocupado. En la Figura 5.12 se puede observar que el parqueo 1 está siendo desocupado, de manera que su píxel correspondiente se torna en color verde, mientras que los demás parqueos se encuentran con etiqueta de color rojo. Adicionalmente, se puede observar el vector que denota el estado de cada parqueo en base a su índice correspondiente, junto con el porcentaje de confianza de la detección, en este caso es del 100 %.

  En las siguientes figuras se pueden apreciar 10 etiquetas, en donde, las que se encuentran ubicadas en los extremos sirven como píxeles pilotos para evitar la existencia de falsos positivos o negativos en los parqueos 1 y 8, generados por el movimiento de los autos cercanos a estos.

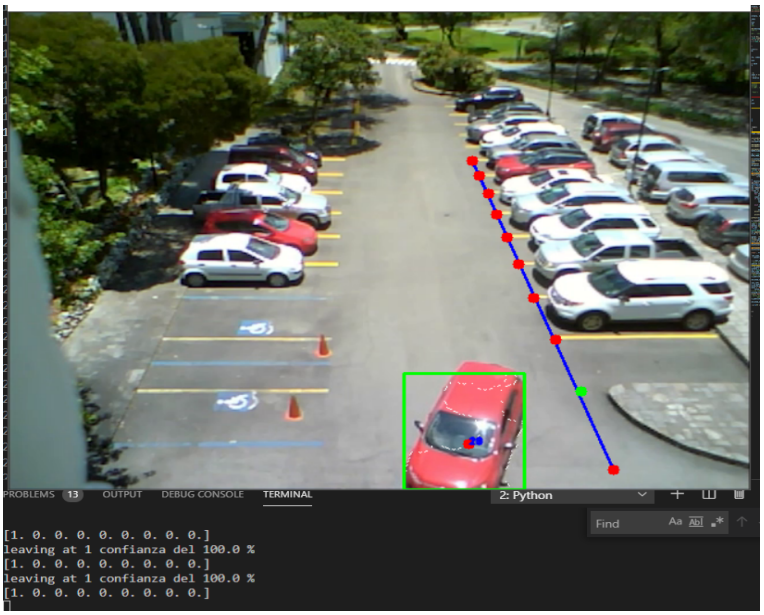


Figura 5.12: Parqueo 1 desocupado

En la Figura 5.13 se puede observar que el parqueo 2 está siendo desocupado, mostrando su etiqueta con color verde y evidenciando un porcentaje de confianza en la detección del 72 %. También se observa un falso positivo en el parqueo 5, el cual se genera debido a que la identificación de los parqueos es susceptible a la velocidad con la que se mueve el automóvil al estacionarse o salir del parqueo.

Figura 5.13: Parqueos 2 y 5 desocupados

Finalmente, en la Figura 5.14 se puede observar el estado de los ocho parqueaderos de interés, mostrando que los parqueos 1, 2, 5 y 8 se encuentran libres. Sin embargo, se observa un falso negativo en el parqueo 2, que en este caso se da debido a que el auto que desocupó aquel espacio salió generando una curva

cerrada desde su arranque, pasando por encima del parqueo libre a su costado y generando errores en la detección del parqueo que realmente está siendo desocupado.

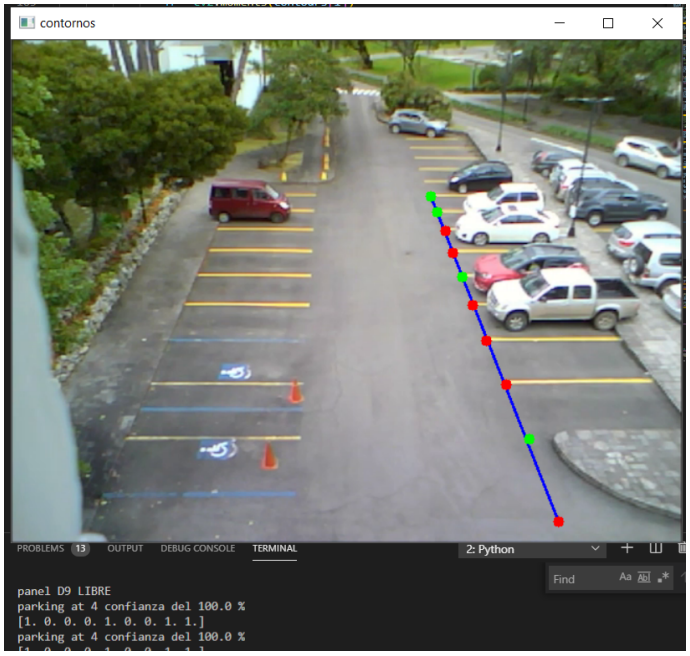


Figura 5.14: Parqueos 1, 2, 5 y 8 desocupados

Pruebas en ambiente nublado

Las pruebas realizadas en el ambiente nublado se las realizan con el objetivo de determinar la resistencia, visibilidad de la señalización y exactitud de detección del subsistema WSN debido al cambio de luminosidad en el ambiente. De igual forma para denotar la eficacia del subsistema de PDI ante un ambiente que presenta dificultades de detección aparentes.

- Subsistema de sensores
  En la Figura 5.15 se puede observar que los cuatro parqueos se encuentran ocupados, de manera que los módulos LED se activan con color rojo. La presencia de un auto junto a otro, permite limitar la luminosidad debajo del cada uno de los autos, mejorando significativamente la detección del estado de cada parqueadero.

Figura 5.15: Parqueos 1 al 4 ocupados

En la Figura 5.16 se puede observar que los cuatro parqueaderos se encuentran libres, de manera que los módulos LED se activan con color verde.

Figura 5.16: Parqueos 1 al 4 libres

Finalmente, en la Figura 5.17 se observa que los parqueaderos 1 y 4 se encuentran ocupados, siendo evidente que los módulos LED de los parqueaderos 2 y 3 se están de color verde, indicando su disponibilidad.

Figura 5.17: Parqueos 2 y 3 libres

- **Subsistema de Visión artificial**
  En la Figura 5.18 se puede observar la detección del movimiento de un automóvil que está desocupando el parqueo 5. La detección tiene una confianza del 100 %, evidenciándose que los parqueaderos 1 y 5 se encuentran libres por medio de etiquetas de color verde.

Figura 5.18: Parqueos 1 y 6 libres

En la Figura 5.19 se puede observar que el parqueadero 6 está siendo desocupado y su porcentaje de confianza es del 100 %. De la misma forma que se observa que los parqueos 1, 4 y 6 se encuentran libres por medio de etiquetas de color verde, aunque, se encuentra un falso positivo correspondiente al parqueo 8.

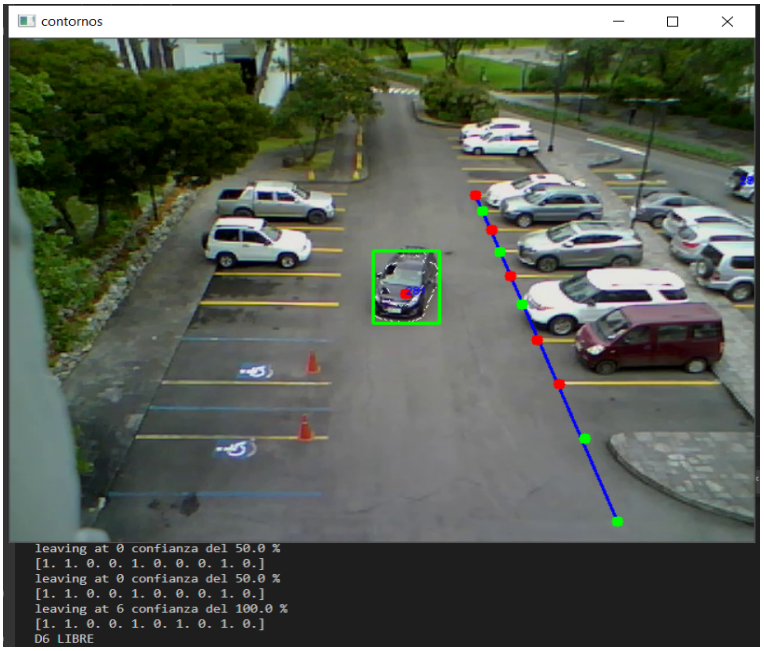


Figura 5.19: Parqueos 1, 4 y 6 libres

Finalmente, en la Figura 5.20 se puede observar que el parqueadero 7 está siendo desocupado, con un

porcentaje de confianza de detección del 100 %. De la misma forma, se muestra que los parqueos 1, 5, 7 y 8 se encuentran libres, y no se evidencia ningún falso positivo ni negativo.

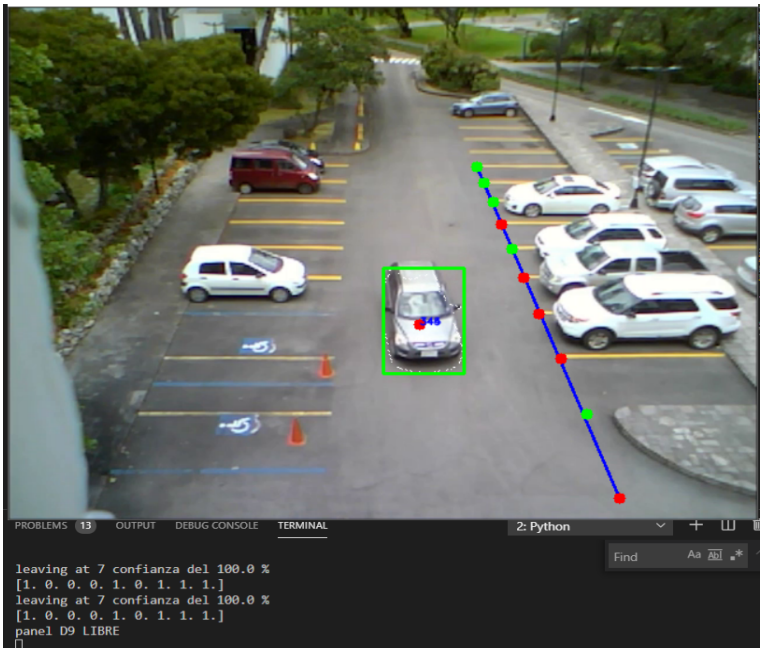


Figura 5.20: Parqueos 1, 5, 7 y 8 libres

Pruebas en ambiente nocturno

Las pruebas realizadas en el ambiente nocturno se las realizan con el objetivo de determinar la exactitud de detección del subsistema WSN debido al cambio de luminosidad en el ambiente. En el horario de penumbra pasado las 18h00 se tienen dos fuentes de luz, la primera es la luz solar aún presente en el ambiente y la segunda es la luz generada por los postes de iluminación; dándose un caso particular en el que, a medida que se pone el sol la variación de luz es más evidente, generando mayor diferencia entre los datos del módulo sensor y el módulo comparador, lo que produce falsos positivos.

En este ambiente nocturno se pueden dar algunos casos particulares, puede ser que se dañen las luminarias de los postes o que llueva. El análisis en los diferentes posibles escenarios sirve para denotar la eficacia del subsistema de PDI y WSN ante un ambiente que presenta dificultades de detección aparentes.

- Subsistema de Sensores
  En la Figura 5.21 se puede observar un escenario particular suscitado en la penumbra. Debido a la existencia de dos fuentes luminosas, se generan diferentes tipos de sombras las cuales pueden llegar a afectar la detección del estado de cada parqueadero. En la Figura se observa que la detección es acertada, denotando la disponibilidad de los parqueaderos 2 y 4.

Figura 5.21: Parqueos 1 y 4 ocupados

En la Figura 5.22 se puede observar la existencia de solo una fuente lumínica, siendo esta, los postes de iluminación. Debido a la luz ambiental, la visibilidad de los módulos LED es mayor, indicando que los parqueaderos 1 y 3 se encuentran disponibles; conjuntamente, el panel LED indica qué parqueadero está listo para ser ocupado.

Figura 5.22: Parqueo 3 ocupado

En la Figura 5.23 se puede evidenciar el funcionamiento del subsistema WSN en un escenario lluvioso, el cual puede llegar a afectar el censado del estado de los parqueaderos debido al reflejo de la luz sobre el asfalto mojado. Gracias al diseño mostrado en la Figura 4.28, los sensores LDR trabajan adecuadamente ante la exposición a lluvia.

Figura 5.23: Parqueos 4 ocupado

Finalmente, en base al análisis realizado en la sub sección 4.1.1 en la Figura 4.5 donde se plantea un escenario de un parqueo libre tapado por la sombra de un auto, se procede a ubicar un sensor LDR en medio de dos autos con el objetivo de censar el estado de su ubicación como se muestra en la Figura 5.24.

El módulo LED correspondiente al sensor LDR se activa con color verde, denotando un estado de disponibilidad, esto gracias a la detección realizada por medio de la fórmula de umbral descrita en la sub sección 4.1.1.

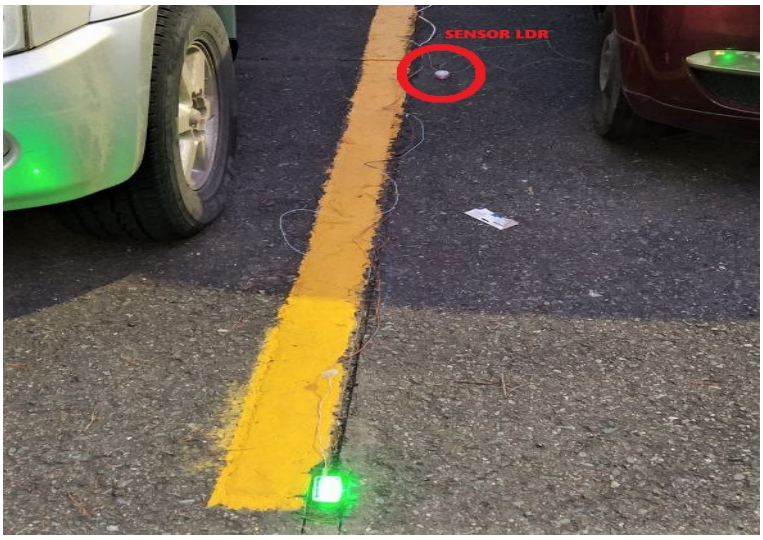


Figura 5.24: Prueba de parqueo libre entre sombras de autos

- **Subsistema de Visión artificial**
  Uno de los escenarios particulares que dificulta la detección de estados de parqueo por medio de PDI es en la lluvia y penumbra. En este escenario se tiene el reflejo de las luminarias de los postes y de los faros de los autos sobre el asfalto mojado, de modo que el área en movimiento procesada por el algoritmo de BS se ve distorsionada por el reflejo.

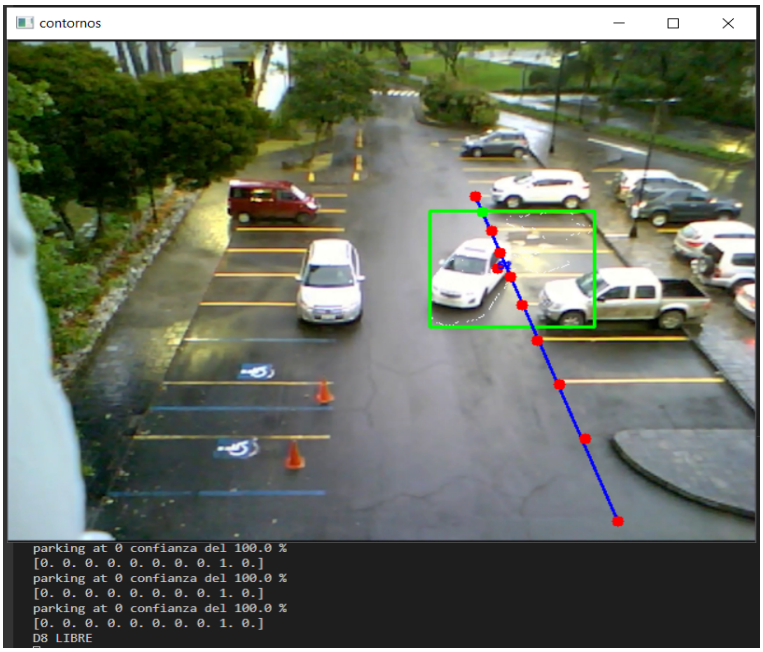


Figura 5.25: Detección de parqueo desocupado en lluvia

En la Figura 5.25 se puede observar que el área reconocida por el movimiento del auto se expande

sobre tres parqueaderos, así el auto haya desocupado el parqueo 6. Esta distorsión produce errores de detección, puesto que la comparación realizada en el proceso de TA se ve afectada por el desplazamiento del centroide del rectángulo circunscrito.

En la Figura 5.26 se puede observar la normalización del área reconocida por el movimiento del auto, al igual que el tamaño del rectángulo circunscrito. Sin embargo, la etiqueta del parqueo 3 se establece en color verde debido a distorsión del área observada en la Figura 5.25.

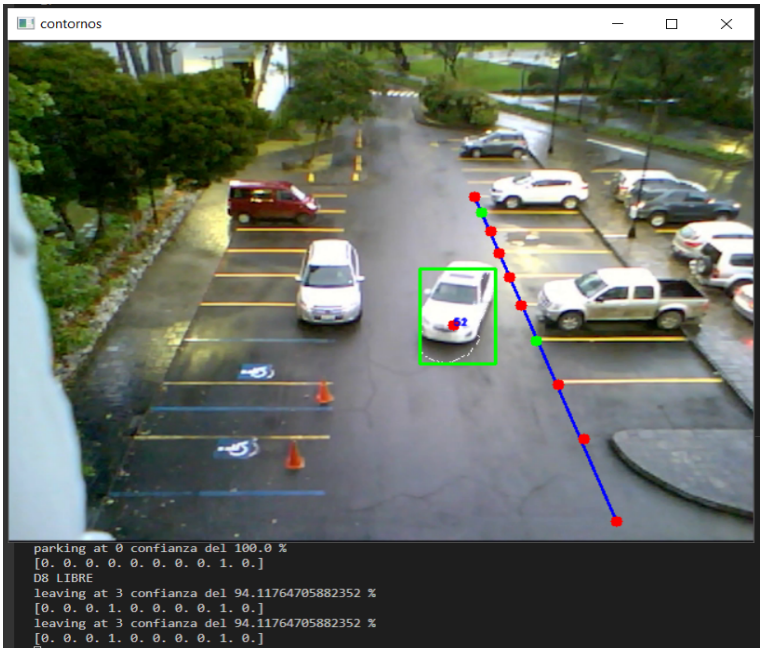


Figura 5.26: Detección de parqueo desocupado en lluvia

5.2.2. Comparación Subsistema de Sensores WSN vs. Subsistema de Visión Artificial

La comparación realizada en la presente sub sección está basada en los resultados obtenidos y evidenciados en la sección anterior. En el presente enlace se puede acceder a las grabaciones correspondientes al comportamiento de los sub sistemas de WSN y PDI. https://drive.google.com/drive/folders/1k-7XwnkpY7C2He6P3gYiap9OCD3ufGNb?usp=sharing

El análisis del subsistema de PDI se ha dividido en dos casos, uno cuando un auto está entrando al parqueadero y otro cuando está saliendo. Además, se dividió en dos ambientes, uno en día y otro en noche debido a la baja diferencia que existe entre los ambientes soleado y nublado. Cabe mencionar que en un ambiente con lluvia los errores aumentan considerablemente debido al reflejo de los autos lo que incrementa el tamaño de la región detectada por el BS.

En la Tabla 5.1 se muestra los resultados obtenidos al correr el algoritmo sobre las grabaciones realizadas. Aquí se puede notar una diferencia entre la detección de los vehículos que entran respecto de los que salen del parqueadero. Esto se debe a que al momento de desocupar un parqueo existe un historial de fondo de donde se encuentra el vehículo, por lo que al salir, el espacio vacío se detecta como objeto hasta que recorra la cantidad

de *frames* necesarios para que el espacio vuelva a formar parte del fondo.

Se detecta errores cuando existen peatones cerca de los vehículos que salen, en este caso los peatones forman un solo objeto con el vehículo por lo que aumenta el área detectada por el algoritmo ocasionando un fallo en la identificación del parqueadero de donde sale el vehículo.

Tabla 5.1: Resultados de observaciones durante el día

	Entrada	Salida
Total	23	32
Errores	4	20
Correctos	19	22
Porcentaje de correctos	82 %	68.75 %

Para el análisis del escenario nocturno la cantidad de datos disponibles disminuye considerablemente debido a que la mayor parte de personal administrativo se retira a partir de las 18h00 debido al horario vigente en el período de la pandemia. Sin embargo se logró recolectar los datos disponibles en la Tabla 5.2.

La mayor parte de datos se originaron en la salida de los vehículos. En este caso los vehículos proyectan una luz sobre el pavimento la cual se detecta como parte del mismo, esto genera la detección de un objeto combinado entre el auto y la luz generada al salir del parqueadero. Esto genera un salto del punto de referencia hacia fuera de la región de interés lo que provoca fallos en la detección de la salida.

Tabla 5.2: Resultados de observaciones durante la noche

	Entrada	Salida
Total	8	19
Errores	4	12
Correctos	4	7
Porcentaje de correctos	50 %	36.84 %

En cuanto a la velocidad de la detección para el sistema de visión artificial se puede aseverar que la detección se realiza en tiempo real a medida que el vehículo sale o entra al parqueadero, mientras que en el sistema de detección con WSN se realiza la detección con un retraso de entre 0 y 4 segundos debido a que los nodos envían datos al concentrador cada 500ms y este calcula la disponibilidad al momento de la llegada de los datos. Este retraso se considera aceptable para efectos prácticos debido a que el cambio de libre a ocupado y viceversa se realiza mientras el auto se encuentra entrando o saliendo del espacio.

Para el caso de los sensores la detección de lugares libres u ocupados es del 100 % exceptuando en momentos específicos del día. Estas fallas son momentáneas en horarios específicos, durante el amanecer y el atardecer, puesto que el sensor LDR que simula un parqueadero ocupado del nodo de referencia recibe luz debido al ángulo de inclinación que tienen los rayos solares y a la existencia de luz por parte de los postes de iluminación. Este

fenómeno provoca un comportamiento errático del umbral lo que origina errores de detección que se corrigen en la siguiente iteración de lectura de los sensores.

5.3. Conclusiones

La comprobación de comportamiento del subsistema de WSN se basa en la precisión de detección del estado de cada uno de los parqueaderos. Dado que la instalación de este subsistema en los predios de la Universidad está sujeta al consentimiento de las autoridades competentes, solo se procedió a instalar en cuatros parqueos. Como se puede observar en la sub sección 5.2.1, las pruebas realizadas en los ambientes soleado, nublado y nocturno han evidenciado que la detección del estado de cada uno de los parqueos es exacta, teniendo una confianza mayor al 99 %. Esta eficiencia es posible gracias a las protecciones implementadas para cada componente del subsistema.

El comportamiento del subsistema de PDI se puede evidenciar en la sub sección 5.2.1, en donde se denota que el subsistema es susceptible a la velocidad con la que un auto se estaciona o desocupa un parqueadero, de igual forma, es susceptible a la presencia de más cuerpos en movimiento en el área de interés.

La presencia de peatones cerca de los autos que se encuentran en movimiento, hace que estos sean reconocidos como un solo cuerpo con el auto por un corto tiempo, lo que genera variación del área reconocida y a su vez fallas en la detección del parqueo utilizado. Un escenario lluvioso es un factor importante para el procesamiento de las imágenes, dado que la presencia del reflejo de la luz sobre el asfalto mojado hace que el área reconocida por el movimiento del auto se distorsione según este avanza.

En la Tabla 5.1 se puede observar que en el día el subsistema de PDI se tiene un 82 % de confianza al detectar un auto en estacionamiento y un 68.75 % de confianza al detectar que un auto desocupa un parqueo, En la Tabla 5.2 se observa que la confianza de detección al estacionar un auto en un parqueo disminuye a 50 % y al desocupar un parqueo se tiene 36.84 %, lo cual se evidencia con las diferentes figuras mostradas en la sub sección 5.2.1.

Capítulo 6

Conclusiones y recomendaciones

6.1. Conclusiones

En las investigaciones realizadas y mencionadas en el capítulo 3 sobre métodos de implementación de WSN, descritas en [37], [38] y [5], se usan diferentes tipos de sensores. El magnetómetro y sensor PIR brindan una detección adecuada del estado de cada parqueo, debido a que son módulos diseñados para la detección de objetos, sin embargo, los costos elevados de instalación, mantenimiento y su elevado consumo energético disminuye su rentabilidad. El sensor LDR es el dispositivo que se ha utilizado para el desarrollo del prototipo del subsistema de WSN gracias a su tamaño pequeño, precio económico y resistencia a golpes y cambios climáticos. Su sensibilidad a la luminosidad, permite calcular diferentes valores de umbral, facilitando la adaptación del subsistema a cualquier escenario presente en diferentes tipos de ambientes, lo que lo convierte en un dispositivo fiable y rentable para ser usado en sistemas de detección de parqueaderos disponibles, tanto al aire libre como en ambientes cerrados.

En el desarrollo del prototipo evidenciado en el capítulo 4 se observa que la parte principal de módulo comparador es el dimensionamiento de los valores de resistencias puesto que es necesario modelar estadísticamente los diferentes escenarios posibles para obtener una detección adecuada. Conjuntamente, el diseño basado en los diferentes criterios de EMC mencionados en las Sub Secciones 4.1.1 y 4.1.2 ha permitido reducir considerablemente a susceptibilidad de los circuitos electrónicos con respecto a cualquier tipo de fuente de EMIs,

Dado que el subsistema de WSN funciona en la intemperie, las protecciones de cada uno de sus elementos cumplen con los estándares de *Ingress Protection* mencionados en la Sección 2.9. En la Figura 4.17 se observa la protección del módulo comparador, la cual fue impresa con filamento PETG, adecuada con un conector prensaestopa y recubierta con silicona en aerosol con el objetivo de impermeabilizar el módulo entero. De la misma forma, los módulos LED y sensores LDR cuentan con protecciones mostradas en las figuras 4.28 y 4.30, las cuales fueron impresas con filamento PETG y rellenadas con resina de poliuretano con el objetivo de resistir alto trafico y cambios climáticos.

Dados los resultados presentados en el capítulo 5 se puede observar que la que la efectividad del subsistema

de WSN es cercana al 100 %, dando fallos únicamente en ocasiones donde existe doble incidencia de luz sobre los sensores. El caso particular en donde se generan falsos positivos, es cuando se tiene una fuente de luz originada por las luminarias del parqueadero y otra de origen solar que se comporta de forma errática, especialmente en la penumbra, donde debido al ángulo de inclinación de la luz solar, incide luz sobre el sensor de referencia de sombra perteneciente al módulo comparador, provocando un comportamiento errático del umbral. Sin embargo, el error se corrige en la siguiente iteración de censado del LDR de sombra, corrigiendo el error en menos de 4 segundos.

En lo que concierne al subsistema basado en PDI, el algoritmo implementado combina las técnicas de BS utilizando el método MoG para detectar y rastrear vehículos, y análisis de transitoriedad TA para determinar el estacionamiento y la salida de vehículos, permitiendo que su trabajo llegue a ser complementario en la detección de parqueos libres en caso falle el subsistema de WSN.

Finalmente, el comportamiento del subsistema de PDI presenta una menor efectividad resumida en las Tablas 5.1 y 5.2, en donde la detección está sujeta a muchas restricciones debido a la ubicación y resolución de la cámara, y la capacidad de procesamiento del ordenador dedicado al procesamiento del video. Las condiciones ideales para ubicar la cámara son a una altura considerable ubicada directamente sobre el parqueadero de tal manera que se logre obtener un ángulo de elevación cercano a 90 grados, de esta forma se evitaría el problema de la oclusión que ocurre entre los vehículos y las fallas de detección debido a errores en los cálculos de distancias debido a la perspectiva generada por el ángulo de elevación. Debido a la alta taza de errores obtenidos en el sistema basado en PDI se implementó la integración de los sistemas de tal manera que se prioriza el funcionamiento del sistema de sensores sobre el sistema de imagen.

El procesamiento de video en la RPi es evidentemente limitado, existiendo saltos en los *frames* de los videos, lo que genera un aumento de errores en la detección de parqueos libres por medio de visión artificial. En base a esto, se ha utilizado una computadora con un procesador Intel Core i7 4470 de 3.4GHz para correr el algoritmo de video, el cual también accede a los datos del procesamiento de sensores publicados por la RPi. Así, basado en la frecuencia de llegada de los datos se decide si hace falta que el subsistema de procesamiento de imágenes intervenga en lo que se muestra o no en el panel LED. De manera que se enviará datos desde el sistema de PDI hacia el panel LED sólo cuando el nodo de sensores o el nodo de referencia fallen.

6.2. Recomendaciones

- Dadas las observaciones realizadas durante el procesamiento de imágenes se pudo evidenciar que el RPi no soporta la carga pesada de procesamiento necesario para el procesamiento de imágenes, esto lleva a escoger una computadora de mesa para realzar el procesamiento de video. Teniendo en cuenta la capacidad computacional del nuevo ordenador se debería contemplar la posibilidad de realizar el desarrollo de la detección de video utilizando redes neuronales y *deep learning*. Basado en [6] una CNN entrenada, puede alcanzar una precisión del 99 % , mientras que según [8] una SVM incrementa su precisión de detección mediante la definición de ROIs, obteniendo una precisión entre 91.1 % hasta 98.8 %.

- En caso de considerar la instalación del sistema de sensores en todos los lugares de parqueadero o en el

mejor de los casos para toda la universidad, se recomienda realizar el estudio de la implementación de un prototipo que funcione de forma autónoma e independiente, es decir que implemente un dispositivo de comunicación integrado en el sensor para evitar el excesivo cableado que se podría generar al instalar los sensores en un número elevado de lugares de parqueo generando una red de sensores inalámbricos donde cada parqueo sería un nodo autónomo con su propia energía proporcionado por baterías.

- Una de las limitaciones del proyecto que generaron inconvenientes en el desarrollo del sistema basado en PDI fue la resolución de la cámara y la ubicación de la misma. Por consiguiente se recomienda que en un futuro proyecto se considere utilizar mayor financiamiento para asegurar mejores resultados con la adquisición de equipos con mejores prestaciones y poder además ubicar la cámara a una altura considerable sobre la mitad del parqueadero para obtener un mejor ángulo sobre la mayor cantidad de espacios de parqueadero posibles para favorecer la definición de áreas de interés y obtener mejores resultados en la detección de espacios disponibles.

6.3. Trabajos futuros

- **Mejora del Prototipo**
  El prototipo experimental del sub sistema de WSN diseñado en el capítulo 4, se lo ha realizado con el objetivo de censar 8 parqueos por medio de un multiplexor CD4051B y controlarlos por medio de los registros de desplazamiento 74LS595 mencionados en la Sección 2.6, lo cual hace necesario la existencia de cableado desde el módulo sensor hasta cada uno de los sensores LDR y módulos LED.

  Se propone independizar de cada sensor LDR con su propio módulo esp8266, el cual le permita enviar los datos censados directamente al RPi para su respectivo procesamiento; el cambio propuesto permite prescindir de la tarjeta electrónica del módulo sensor y el cableado del mismo con los demás elementos del subsistema.

  Cada parqueo tendría su propio sub sistema de censado, manteniendo la comunicación con el RPi por medio del protocolo MQTT, lo cual facilitará la detección y localización de errores y daños de cada sensor LDR. Conjuntamente, cada módulo de censado controlará sus módulos LED correspondientes, siendo necesarios sólo los cables de alimentación y manteniendo el diseño de cortes mostrado en la Figura 5.5 o considerar la alimentación por medio de baterías y celdas solares en cuyo caso se incrementaría la inversión inicial en favor de la escalabilidad del sistema.

- **Instalación en todos los parqueos del campus central de la Universidad de Cuenca**
  En base a lo mencionado en la Sección 1.3 y en los resultados evidenciados en la Sección 5.3 se comprueba que el sistema general funciona de manera eficiente. El comportamiento del proyecto experimental desarrollado es el esperado, en especial el sub sistema de WSN que cuenta con eficiencia del 100 %, junto con la resistencia evidenciada ante cambios climáticos, golpes y tensiones.

  El sub sistema de PDI puede ser mejorado de muchas maneras dado a que está sujeto a diferentes factores como resolución y ubicación de la cámara, acceso al sistema de video de la Universidad y capacidad de

procesamiento computacional del ordenador dedicado al procesamiento del video.

Como un factor de escalabilidad, se ha pensado en la instalación del prototipo en todo el campus central de la Universidad de cuenca, el cual se muestra en la Figura 6.1.

Figura 6.1: Campus Central de la Universidad de Cuenca

Mediante un análisis detallado de la Figura 6.1, se llegan a contar 496 parqueos existentes en el campus central, los cuales se encuentran repartidos en 16 bloques diferentes. En la Tabla 6.1 se detalla la cantidad parqueaderos en cada bloque junto con la cantidad de módulos sensores y comparadores necesarios en cada bloque de parqueaderos.

Se ha destinado 1 módulo comparador para actuar como referencia de hasta 2 módulos sensores y cada módulo sensor puede censar y controlar 8 parqueaderos diferentes, debido a los diferentes escenarios que se dan en cada uno de los bloques a causa de la infraestructura del campus central.

Así, se puede observar que para un total de 496 parqueaderos se necesitan 62 módulos sensores y 34 módulos comparadores, conjuntamente se necesitan 16 paneles LED, a ser ubicados en la entrada de cada bloque de parqueaderos para la visualización del parqueo a ser ocupado y 13 RPi para procesar los datos de los parqueos de cada facultad.

En la Tabla 6.2 se puede observar un proyección aproximada a los costos de instalación del prototipo de WSN en el campus central. Los precios que la componen son tomados de las Tablas 4.2, 4.3, 4.1 y 4.4, las cuales están detalladas en la Sección 4.3.

Para el proceso de instalación se ha aproximado una cantidad de cable de 3 pares necesario, sin embargo, una medición presencial permitirá una proyección más específica de la cantidad de los materiales necesarios.

Tabla 6.1: Detalle de parqueos en el campus central

Ubicación	N° Parqueos	N° Módulos Sensor	N° Módulos Comparador
Facultad de Ingeniería	33	4	2
Teatro Carlos Cueva Tamariz	28	4	3
Edificio Administrativo	14	2	1
Facultad Arquitectura	40	5	2
Dispensario Médico	27	3	2
Facultar Ciencias Químicas	29	4	2
Facultad Filosofía	52	7	3
Facultad de Jurisprudencia	26	3	2
Facultad de Psicología	46	6	3
Facultad Ciencias Económicas B1	20	3	2
Facultad Ciencias Económicas B2	8	1	1
Facultad Ciencias Económicas B3	5	1	1
Facultad Ciencias Económicas B4	18	2	1
Facultad Arquitectura Maestrías	55	7	3
Coliseo y Canchas	48	6	3
Facultad de Educación Física	47	6	3
Total	**496**	**62**	**34**

Tabla 6.2: Proyección de costos de instalación

Descripción	Cantidad	P. Unitario	P. Total
Tarjeta Electrónica Módulo Comparador	34	52,24	1776,16
Tarjeta Electrónica Modulo Sensor	62	20,41	1265,42
Panel LED	16	93,7	1499,2
Protecciones LDR	496	6	2976
Protecciones LEDs	496	6	2976
Caja de Paso para Módulo Sensor	62	14,5	899
Protección para Módulo Comparador	34	40	1360
Raspberry	13	60	780
Cable 3 pares (100m)	24,8	31	768,8
Total Aproximado			**14300,58**

ANEXO A

Códigos de los Módulos del Subsistema de Sensores WSN

A.1. Código de Modulo Comparador

```
// Se cargan las librerias para el modulo ESP8266
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

int sensorPin = A0;
uint16_t sensorValue = 0;

// Nodo 0 es el nodo de referencia
const int node_id = 0;
const String data_out_topic = "datos_sensor"; // este no cambiar

// Pines para controlar qué ldr leer 0 - 7
uint8_t pin_a_ldrs = 16; // LSB
uint8_t pin_b_ldrs = 5;
uint8_t pin_c_ldrs = 4; // MSB

// Pines para controlar con qué resistencia leer el ldr 0 - 7
uint8_t pin_a_res = 14;//LSB
uint8_t pin_b_res = 2;
uint8_t pin_c_res = 0; // MSB

int choosen_ldr = 0;
int counter = 0 ;
int index_ldr = 3;
int interval = 5000;
uint8_t current_value = 0;

uint8_t var = 0;

// Datos de la red
//const char* ssid = "RaspberryPiFi";
//const char* password = "0123456789A";

// Direcci'on del broker en raspberry
```

```
36 //const char* mqtt_server = "192.168.42.1";
37
38 const char* ssid = "Linksys01263";
39 const char* password = "123456789";
40
41 // Direccion IP del broker en raspberry
42 const char* mqtt_server = "192.168.1.100";
43
44 // Initializa el espClient de mqtt
45 WiFiClient espClient;
46 PubSubClient client(espClient);
47
48 // Timers auxiliar variables
49 long now = millis();
50 long lastMeasure = 0;
51
52
53 void setup_wifi()
54 {
55   delay(10);
56   // We start by connecting to a WiFi network
57   WiFi.begin(ssid, password);
58   while (WiFi.status() != WL_CONNECTED)
59   {
60     delay(500);
61   }
62 }
63
64
65 void callback(String topic, byte* message, unsigned int length)
66 {
67   // en este nodo no llegan datos
68 }
69
70 // This functions reconnects your ESP8266 to your MQTT broker
71 // Change the function below if you want to subscribe to more topics with your ESP8266
72 void reconnect()
73 {
74   // Loop until we're reconnected
75   while (!client.connected())
76   {
77     if (client.connect("ESP8266Client " + node_id))
78     { // Este String debe ser unico para cada cliente
79       Serial.println("connected");
80     }
81     else
82     {// Wait 5 seconds before retrying
83       delay(5000);
84     }
85   }
86 }
87
88 void setup()
89 {
90
91   // LDRs en BCD
92   pinMode(pin_a_ldrs, OUTPUT);
93   pinMode(pin_b_ldrs, OUTPUT);
94   pinMode(pin_c_ldrs, OUTPUT);
95
96   // Resistencias en BCD
97   pinMode(pin_a_res, OUTPUT);
98   pinMode(pin_b_res, OUTPUT);
99   pinMode(pin_c_res, OUTPUT);
100
```

```
101   // Inicializa las salidas en LOW
102   //  display_leds(0, false);
103   choose_res(0); // inicializa salidas res en LOW
104   read_ldr(0);   // Inicializa salidas ldr en LOW
105
106   Serial.begin(115200);
107   setup_wifi();
108   client.setServer(mqtt_server, 1883);
109   client.setCallback(callback);
110 }
111
112 /*
113  * En el loop nos encargamos de que se conecte, arme y publique el Json de datos
114  */
115
116 void loop()
117 {
118   if (!client.connected())
119   {
120     reconnect();
121   }
122   if (!client.loop())
123     client.connect("ESP8266Client " + node_id);
124   now = millis();
125
126   // La publicacion se la realiza cada 5 segundos
127   if (now - lastMeasure > 5000)
128   {
129     const size_t capacity = JSON_ARRAY_SIZE(8) + JSON_OBJECT_SIZE(4);
130     double valor = 0;
131     lastMeasure = now;
132
133     DynamicJsonDocument doc(capacity);
134     doc.clear();
135     doc["node"] = node_id;
136
137     doc["parking_index"] = choosen_ldr; // indice del LDR muestreado
138     doc["current_value"] = current_value; // El valor que se publica a los Leds(en decimal)
139     JsonArray data = doc.createNestedArray("data");
140
141     // Barrido en resistencias
142     for (int i = 0; i <= 7; i++)
143     {// desde cero hasta siete
144       choose_res(i); // Con esto barremos las resistencias no los LDRs
145       valor = read_ldr(choosen_ldr) * 3.3 / 1024;
146       data.add(valor);
147       delay(100);
148     }
149
150     String output = "";
151     serializeJson(doc, output); // Hacia el String
152     int tam = output.length() + 1;
153     char buf[tam];
154     output.toCharArray(buf, tam);
155     client.publish("datos_sensor", buf);
156     choosen_ldr++;
157   }
158
159   if(choosen_ldr > 2) // son 3 referencias 0, 1, 2
160   {
161     choosen_ldr = 0;
162   }
163 }
164
165
```

```
/*
   valor : index del ldr a leer  0 <= valor <= 7
*/
int read_ldr (uint8_t valor)
{
  digitalWrite(pin_a_ldrs, bitRead(valor, 0));
  digitalWrite(pin_b_ldrs, bitRead(valor, 1));
  digitalWrite(pin_c_ldrs, bitRead(valor, 2));
  delay(10);
  return analogRead(sensorPin);
}

/*
   valor: index de la resistencia a usar  0 <= valor <= 7
*/
void choose_res (uint8_t valor)
{
  digitalWrite(pin_a_res, bitRead(valor, 0));
  digitalWrite(pin_b_res, bitRead(valor, 1));
  digitalWrite(pin_c_res, bitRead(valor, 2));
}
```

A.2. Código de Modulo Sensor

```

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

int sensorPin = A0;
uint16_t sensorValue = 0;


//TODO: revisar estos valores para cada nodo sensor
const int node_id = 1;
const String data_out_topic = "datos_sensor";

// pines para controlar los leds 0 - 7
uint8_t pin_clock_leds    = 12; // D6  Clock
uint8_t pin_data_leds     = 15; // D8  Latch
uint8_t pin_latch_leds    = 13; // D7  DATA_IN

// Pines para controlar qué ldr leer 0 - 7
uint8_t pin_a_ldrs = 16; // LSB
uint8_t pin_b_ldrs = 5;
uint8_t pin_c_ldrs = 4; // MSB

// Pines para controlar con qué resistencia leer el ldr 0 - 7
uint8_t pin_a_res = 14;//LSB
uint8_t pin_b_res = 2;
uint8_t pin_c_res = 0; // MSB

int counter = 0 ;
int index_ldr = 3;
int interval = 5000;
int choosen_ldr = 0;
uint8_t current_value = 0;

uint8_t var = 0;

// Datos de la red
const char* ssid = "Linksys01263";
```

```
39 const char* password = "123456789";
40 const char* mqtt_server = "192.168.1.100";
41
42 // Initializa el espClient de mqtt
43 WiFiClient espClient;
44 PubSubClient client(espClient);
45
46 // Timers auxiliar variables
47 long now = millis();
48 long lastMeasure = 0;
49
50
51 void setup_wifi()
52 {
53   delay(10);
54   // We start by connecting to a WiFi network
55   WiFi.begin(ssid, password);
56   while (WiFi.status() != WL_CONNECTED)
57   {
58     delay(500);
59   }
60 }
61 /*
62  * Esta función se ejecuta al recibir un mensaje (MQTT)
63  * Recibimos un número en decimal (en binario indica que leds deben prenderse)
64  */
65 void callback(String topic, byte* message, unsigned int length)
66 {
67   String messageTemp;
68   for (int i = 0; i < length; i++)
69   {
70     messageTemp += (char)message[i];
71   }
72   Serial.println();
73   uint8_t val = messageTemp.toInt();
74   Serial.println(val);
75   // manda a los leds
76   putLeds(val);
77   current_value = val;
78 }
79
80 void reconnect()
81 {
82   while (!client.connected())
83   {
84     Serial.print("Attempting MQTT connection...");
85     if (client.connect("ESP8266Client " + node_id))
86     { // Este String debe ser unico para cada cliente
87       Serial.println("connected");
88       client.subscribe("nodo1");
89     }
90     else
91     {
92       Serial.print("failed, rc = ");
93       Serial.print(client.state());
94       Serial.println(" try again in 5 seconds");
95       // Wait 5 seconds before retrying
96       delay(5000);
97     }
98   }
99 }
100
101 void setup()
102 {
103   pinMode(pin_clock_leds, OUTPUT);
```

```
104   pinMode(pin_data_leds, OUTPUT);
105   pinMode(pin_latch_leds, OUTPUT);
106
107   // LDRs en BCD
108   pinMode(pin_a_ldrs, OUTPUT);
109   pinMode(pin_b_ldrs, OUTPUT);
110   pinMode(pin_c_ldrs, OUTPUT);
111
112   // Resistencias en BCD
113   pinMode(pin_a_res, OUTPUT);
114   pinMode(pin_b_res, OUTPUT);
115   pinMode(pin_c_res, OUTPUT);
116
117   // Inicializa las salidas en LOW
118   choose_res(0); // inicializa salidas res en LOW
119   read_ldr(0);   // Inicializa salidas ldr en LOW
120
121   Serial.begin(115200);
122   setup_wifi();
123   client.setServer(mqtt_server, 1883);
124   client.setCallback(callback);
125 }
126
127 /*
128  * En el loop nos encargamos de que se conecte, arme y publique el Json de datos
129  */
130 void loop()
131 {
132   if (!client.connected())
133   {
134     reconnect();
135   }
136   if (!client.loop())
137     client.connect("ESP8266Client " + node_id);
138   now = millis();
139
140   // Publishes new array of values to the raspberry every (interval/1000) seconds
141   if (now - lastMeasure > 5000)
142   {
143     double valor = 0;
144     lastMeasure = now;
145     const size_t capacity = JSON_ARRAY_SIZE(8) + JSON_OBJECT_SIZE(4);
146     DynamicJsonDocument doc(capacity);
147     doc.clear();
148     doc["node"] = node_id;
149     doc["parking_index"] = choosen_ldr; // indice del LDR muestreado
150     doc["current_value"] = current_value; // El valor que se publica a los Leds(en decimal)
151     JsonArray data = doc.createNestedArray("data");
152
153     // Barrido en resistencias
154     for (int i = 0; i <= 7; i++)
155     {// desde cero hasta siete
156       choose_res(i); // Con esto barremos las resistencias no los LDRs
157       valor = read_ldr(choosen_ldr) * 3.3 / 1024;
158       data.add(valor);
159     }
160
161     String output = "";
162     serializeJson(doc, output); // Hacia el String
163     int tam = output.length() + 1;
164     char buf[tam];
165
166     output.toCharArray(buf, tam);
167     Serial.print("Datos bufer: ");
168     Serial.println(buf);
```

```
169     client.publish("datos_sensor", buf);
170
171     choosen_ldr++;
172   }
173   if(choosen_ldr >=8)
174   { // del 0 al 7
175     choosen_ldr = 0;
176   }
177 }
178
179
180 void putLeds(uint8_t salidaLeds)
181 {
182   digitalWrite(pin_latch_leds, HIGH);
183   uint16_t valor_salida = ((~salidaLeds) << 8) | salidaLeds;
184   for (int i = 0; i < 16; i++)
185   {
186     digitalWrite(pin_clock_leds, LOW);
187     //digitalWrite(pin_data_leds,bitRead(salidaLeds, i));
188     digitalWrite(pin_data_leds, bitRead(valor_salida, i));
189     digitalWrite(pin_clock_leds, HIGH);
190   }
191   digitalWrite(pin_latch_leds, LOW);
192 }
193
194 /*
195    valor : index del ldr a leer  0 <= valor <= 7
196 */
197 int read_ldr (uint8_t valor) {
198   digitalWrite(pin_a_ldrs, bitRead(valor, 0));
199   digitalWrite(pin_b_ldrs, bitRead(valor, 1));
200   digitalWrite(pin_c_ldrs, bitRead(valor, 2));
201   delay(10);
202   return analogRead(sensorPin);
203 }
204
205 /*
206    valor: index de la resistencia a usar  0 <= valor <= 7
207 */
208 void choose_res (uint8_t valor) {
209   digitalWrite(pin_a_res, bitRead(valor, 0));
210   digitalWrite(pin_b_res, bitRead(valor, 1));
211   digitalWrite(pin_c_res, bitRead(valor, 2));
212 }
```

A.3. Código de Modulo Panel LED

A.3.1. Código Maestro - NodeMCU

```
1  #include <ESP8266WiFi.h>
2  #include <PubSubClient.h>
3  #include <ArduinoJson.h>
4  #include <Wire.h>
5
6  // DATOS DE LA RED //
7
8  const char* ssid = "Linksys01263";
9  const char* password = "123456789";
10 const char* mqtt_server = "192.168.1.100";
11
12
```

```
13 WiFiClient espClient; // Initializa el espClient de mqtt
14 PubSubClient client(espClient);
15
16 long now = millis(); // Timers auxiliar variables
17
18 void setup()
19 {
20   Serial.begin(9600); /* begin serial for debug */
21   Wire.begin(D1, D2); /* join i2c bus with SDA=D1 and SCL=D2 of NodeMCU */
22   setup_wifi();
23   client.setServer(mqtt_server, 1883);
24   client.setCallback(callback);
25 }
26
27 void loop()
28 {
29   if (!client.connected())
30   {
31     reconnect();
32   }
33   if (!client.loop())
34       client.connect("CLIENTE PANEL");
35   now = millis();
36 }
37
38 void setup_wifi()
39 {
40   delay(10);
41   Serial.println();
42   Serial.print("Conectandose a: ");
43   Serial.println(ssid);
44   WiFi.begin(ssid, password);
45
46   while (WiFi.status() != WL_CONNECTED)
47   {
48     delay(500);
49     Serial.print("Reintentando conectarse a: ");
50     Serial.println(ssid);
51   }
52   Serial.print("WiFi connectado - Direccion ESP IP: ");
53   Serial.println(WiFi.localIP());
54 }
55
56 void callback(String topic, byte* message, unsigned int length)
57 {
58   Serial.print("Mensaje Recibido en el topic: ");
59   Serial.print(topic);
60   Serial.print(". Mensaje: ");
61   String messageTemp;
62
63   for (int i = 0; i < length; i++)
64   {
65     Serial.print((char)message[i]);
66     messageTemp += (char)message[i];
67   }
68   Serial.println();
69   Wire.beginTransmission(8); /* begin with device address 8 */
70   Wire.write(messageTemp.c_str());
71   Wire.endTransmission();    /* stop transmitting */
72   Serial.println(messageTemp.c_str());
73   Serial.println();
74   delay(500);
75 }
76
77 void reconnect()
```

```
{
  while (!client.connected())
  {
    Serial.print("Conectando con el Broker MQTT...");
    if (client.connect("CLIENTE PANEL"))
    {
      Serial.println("PANEL LED Conectado");
      client.subscribe("panel");
      Serial.println("Suscrito a panel");
    }
    else
    {
      Serial.print("FALLO, rc = ");
      Serial.print(client.state());
      Serial.println(" reintentando en 5 segundos");
      delay(5000);
    }
  }
}
```

A.3.2. Código Esclavo - Arduino Uno

```
#include <Wire.h>
#include <SPI.h>
#include <DMD2.h>
#include <fonts/Tesis1.h>


const uint8_t *FONT = Tesis1;
SoftDMD dmd(3,1);   // Number of P10 panels used X, Y

DMD_TextBox box(dmd, 0, 0, 96, 16); // x, y change text posistion [ Set Box (dmd, x, y, Height, Width)]

void setup()
{
  Wire.begin(8);                /* join i2c bus with address 8 */
  Wire.onReceive(receiveEvent); /* register receive event */
  //Wire.onRequest(requestEvent); /* register request event */
  Serial.begin(9600);           /* start serial for debug */

  dmd.setBrightness(10); // Set brightness 0 - 150
  dmd.selectFont(FONT); // Font used
  dmd.begin();     // Start DMD
}

void loop()
{
  delay(100);
}

// function that executes whenever data is received from master

void receiveEvent(int howMany)
{
  String v;
  while (0 <Wire.available())
  {
    char c = Wire.read();      /* receive byte as a character */
    //Serial.println(c);           /* print the character */
    v += c;
  }
  Serial.println(v);
  const char *MESSAGE = v.c_str();
```

```
42   Serial.println(MESSAGE);
43   int s = strlen(MESSAGE);
44   Serial.println(s);
45
46   dmd.clearScreen();// clear screen
47   dmd.drawString(0, 0,MESSAGE);
48
49   Serial.println();             /* to newline */
50 }
```

A.4. Código de Algoritmo de Detección Implementado en la Raspberry Pi

```
1  from cv2 import cv2
2  import csv
3  import numpy as np
4  import Object
5  import math
6
7  VIDEO_SOURCE = 'video2.mp4'
8
9  cap = cv2.VideoCapture(VIDEO_SOURCE)
10
11 fgbg = cv2.bgsegm.createBackgroundSubtractorMOG(
12 history=500, nmixtures=3, backgroundRatio=0.5, noiseSigma=0)
13
14 class spots:
15     loc = 0
16
17 def callback(foo):
18     pass
19
20 def distance(inicio,fin):
21     distancia = math.sqrt((fin[0]-inicio[0])**2+(fin[1]-inicio[1])**2)
22     return distancia
23 def getIndexMin(vector):
24     mm = min(vector)
25     for i in range(len(vector)):
26         if vector[i] == mm:
27             return i
28
29
30 ret, frame = cap.read()
31 h = frame.shape[0]
32 w = frame.shape[1]
33
34 objetos = []
35
36 frameArea = h*w
37 areaTH = frameArea/350
38 print('Area Threshold',areaTH)
39 max_p_age = 7
40
41 kernel = np.ones((3, 3), np.uint8)
42
43 kernelOp = np.ones((3, 3), np.uint8)
44 kernelOp2 = np.ones((5, 5), np.uint8)
45 kernelCl = np.ones((11, 11), np.uint8)
46
47 points = np.array([
48     [420, 350],
```

```
    [401, 301],
    [385, 261],
    [368, 231],
    [359, 206],
    [349, 183],
    [341, 164],
    [332, 147],
])

font = cv2.FONT_HERSHEY_SIMPLEX

pid = 1

while True:

    spots.loc = 0
    ret, frame = cap.read()
    frameOringinal = np.copy(frame)

    cv2.line(frameOringinal,(290,70),(points[0][0],points[0][1]),(255,0,0),5)

    for i in objetos:
        i.age_one()

    frame = cv2.GaussianBlur(frameOringinal, (3, 3), sigmaX=3, sigmaY=3)
    fgmask = fgbg.apply(frame)

    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernelOp)
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_CLOSE, kernelCl, iterations=2)

    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel)
    fgmask = cv2.dilate(fgmask, kernel, iterations=0)
    fgmask = cv2.morphologyEx(
        fgmask, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (30, 30)))

    contours, hierarchy = cv2.findContours(
        fgmask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)


    x = range(332,418)
    y=[]
    for i in range(len(x)):
        y.append( 3 * x[i] - 903 )

    hull = []
    for c in contours:
        area = cv2.contourArea(c)
        rect = cv2.minAreaRect(c)

        if area > areaTH and area < 7000:
            cv2.fillPoly(frameOringinal, pts=c, color=(255, 255, 255))
            cv2.drawContours(frameOringinal, [
                             c], 0, (0, 255, 0), 2, cv2.LINE_AA)
            (x, y, w, h) = cv2.boundingRect(c)
            cv2.rectangle(frameOringinal, (x, y), (x + w, y + h),
                          (0, 255, 0), 1, cv2.LINE_AA)
        if area < areaTH:
            cv2.fillPoly(frameOringinal, pts=[c], color=0)
        hull.append(cv2.convexHull(c, False))

    drawing = np.zeros((frame.shape[0], frame.shape[1], 3), np.uint8)

    for i in range(len(contours)):
        hull.append(cv2.convexHull(contours[i], False))
```

```
114         area = cv2.contourArea(contours[i])
115         if area > areaTH:
116             color_contours = (0, 0, 255)
117             color = (255, 0, 0)
118             cv2.drawContours(drawing, contours, i, color_contours, 1, 8, hierarchy)
119 
120             cv2.drawContours(drawing, hull, i, color, -1, 8)
121 
122             M = cv2.moments(contours[i])
123             cx = int(M['m10']/M['m00'])
124             cy = int(M['m01']/M['m00'])
125             x, y, w, h = cv2.boundingRect(contours[i])
126 
127             new = True
128             for obj in objetos:
129                 cv2.putText(frameOringinal, str(obj.getId()), (obj.getX(), obj.getY()),
130                     font, 0.3, obj.getRGB(), 1, cv2.LINE_AA)
131                 if abs(x-obj.getX()) <= w + 5 and abs(y-obj.getY()) <= h + 5:
132                     new = False
133                     obj.updateCoords(cx, cy)
134                     var = obj.parking_()
135                     if(var == 'parking' or var == 'leaving'):
136                         dist = []
137                         for tempVal in range(len(points)):
138                             dist.append(abs(points[tempVal][1] - [y+h]))
139                         index_min = getIndexMin(dist)
140                         print(var + ' at ' + str(index_min))
141 
142                 if obj.timedOut():
143                     # sacar obj de la lista objetos
144                     index = objetos.index(obj)
145                     objetos.pop(index)
146                     del obj  # liberar la memoria de i
147             if new == True:
148                 p = Object.MyObject(pid, cx, cy, max_p_age)
149                 objetos.append(p)
150                 pid += 1
151 
152             cv2.circle(frameOringinal, (cx, cy), 5, (0, 0, 255), -1)
153             img = cv2.rectangle(frameOringinal, (x, y), (x+w, y+h), (0, 255, 0), 2)
154             cv2.drawContours(frameOringinal, contours[i], -1, (255, 255, 255), cv2.FILLED)
155 
156             cv2.circle(drawing, (cx, cy), 5, (0, 0, 255), -1)
157             img = cv2.rectangle(drawing, (x, y), (x+w, y+h), (0, 255, 0), 2)
158             cv2.drawContours(drawing, contours[i], -1, (255, 255, 255), cv2.FILLED)
159 
160     cv2.imshow('contornos', frameOringinal)
161     cv2.imshow('hulls', drawing)
162 
163     font = cv2.FONT_HERSHEY_SIMPLEX
164     cv2.putText(frame, 'PARQUEOS LIBRES: ' + str(spots.loc),
165                 (10, 30), font, 1, (0, 255, 0), 3)
166 
167     # cv2.imshow('frame', fgmask)
168 
169     if cv2.waitKey(1) & 0xFF == ord('q'):
170         break
171 
172 
173 cap.release()
174 cv2.destroyAllWindows()
```

ANEXO B

Código de Algoritmo de Procesamiento de Imágenes en Python

B.1. Código del objeto utilizado para el algoritmo de Análisis de transitoriedad

```
import time
from random import randint

class MyObject:

    def __init__(self, i, xi, yi, max_age):
        self.i = i
        self.x = xi
        self.y = yi
        self.tracks = []
        self.done = False
        self.state = []
        self.age = 0
        self.max_age = max_age
        # self.dir = None

    def getId(self):
        return self.i

    def getX(self):
        return self.x

    def getY(self):
        return self.y

    def updateCoords(self, xn, yn):
        self.age = 0
        if len(self.tracks) > 0:
            x_media = round((self.x + self.tracks[-1][0]) / 2)
            y_media = round((self.y + self.tracks[-1][1]) / 2)
        else:
            x_media = xn
            y_media = yn
```

```
34          self.tracks.append([x_media, y_media])
35          self.x = xn
36          self.y = yn
37
38      def timedOut(self):
39          return self.done
40
41      def age_one(self):
42          self.age += 1
43          if self.age > self.max_age:
44              self.done = True
45          return True
46
47      def parking_(self):
48          th = 12
49          valor_y_eq = (2.15 * self.x - 603)
50          if len(self.tracks) >= 5:
51              tr = self.tracks
52              media4 = (tr[-2][0] + tr[-3][0])/2
53              # dentro de la franja de la linea
54              if (self.y >= valor_y_eq - th) & (self.y <= valor_y_eq + th):
55                  if self.tracks[-1][0] > media4:  # va hacia la derecha
56                      val = 'parking'
57                  elif self.tracks[-1][0] < media4:  # va hacia la izquierda
58                      val = 'leaving'
59                  else:
60                      val = 'incierto'
61                  self.state.append(val)
62              elif(len(self.state) > 5):  # posiblemente fuera de la franja de la linea
63                  if (self.state.count('parking') > self.state.count('leaving')):
64                      self.state = []
65                      return 'parking'
66                  else:
67                      self.state = []
68                      return 'leaving'
69          return '-1'
```

B.2. Código de Algoritmo de procesamiento de imágenes

```
1  import math
2
3  import numpy as np
4  from cv2 import cv2
5
6  import Object
7
8  def getIndexMin(vector):
9      mm = min(vector)
10     for i in range(len(vector)):
11         if vector[i] == mm:
12             return i
13
14
15 VIDEO_SOURCE = 'video2.mp4'
16 cap = cv2.VideoCapture(VIDEO_SOURCE)
17
18 ret, frame = cap.read()
19 h = frame.shape[0]
20 w = frame.shape[1]
21
22 frameArea = h*w
23 areaTH = frameArea/350
```

```
24
25 max_p_age = 7
26 objetos = []
27 kernelOp = np.ones((3, 3), np.uint8)
28
29 points = np.array([
30     [420, 350],
31     [401, 301],
32     [385, 261],
33     [368, 231],
34     [359, 206],
35     [349, 183],
36     [341, 164],
37     [332, 147],
38 ])
39
40
41 # (1 para libres 0 para ocupados)
42 estados = [1, 1, 1, 1, 1, 1, 1, 1]  # 8 espacios inicializados en libres
43 font = cv2.FONT_HERSHEY_SIMPLEX
44 oid = 1
45 text_color = (255, 0, 0)
46
47 fgbg = cv2.bgsegm.createBackgroundSubtractorMOG(
48     history=500, nmixtures=3, backgroundRatio=0.5, noiseSigma=0)
49
50 while True:
51
52     ret, frame = cap.read()
53     frameOriginal = np.copy(frame)
54
55     cv2.line(frameOriginal, (290, 70),
56              (points[0][0], points[0][1]), (255, 0, 0), 5)
57
58     newFrame = cv2.GaussianBlur(
59         frameOriginal, (3, 3), sigmaX=3, sigmaY=3)
60     fgmask = fgbg.apply(newFrame)
61
62     fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernelOp)
63     fgmask = cv2.morphologyEx(
64         fgmask, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (30, 30)))
65
66     contours, hierarchy = cv2.findContours(
67         fgmask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
68
69     for i in range(len(contours)):
70         area = cv2.contourArea(contours[i])
71         if area > areaTH:
72             color_contours = (0, 0, 255)
73             cv2.drawContours(
74                 frameOriginal, contours[i], -1, (255, 255, 255), cv2.FILLED)
75
76             M = cv2.moments(contours[i])
77             cx = int(M['m10']/M['m00'])
78             cy = int(M['m01']/M['m00'])
79             x, y, w, h = cv2.boundingRect(contours[i])
80             cv2.circle(frameOriginal, (cx, cy), 5, (0, 0, 255), -1)
81
82             img = cv2.rectangle(frameOriginal, (x, y),
83                                 (x+w, y+h), (0, 255, 0), 2)
84
85             new = True
86             for obj in objetos:
87                 obj.age_one()
88                 if abs(x-obj.getX()) <= w + 5 and abs(y-obj.getY()) <= h + 5:
```

```
89                      new = False
90                      obj.updateCoords(cx, cy)
91                      var = obj.parking_()
92                      if(var == 'parking' or var == 'leaving'):
93                          dist = []
94                          for tempVal in range(len(points)):
95                              dist.append(abs(points[tempVal][1] - [y+h]))
96                          index_min = getIndexMin(dist)
97                          if var == 'parking':
98                              estados[index_min] = 0  # marca ocupado
99                          elif var == 'leaving':
100                             estados[index_min] = 1  # marca libre
101                         print(var + ' at ' + str(index_min))
102                         print(estados)
103
104                 cv2.putText(frameOriginal, str(obj.getId()), (obj.getX(), obj.getY()),
105                             font, 0.3, text_color, 1, cv2.LINE_AA)
106                 if obj.timedOut():
107                     # sacar obj de la lista objetos
108                     index = objetos.index(obj)
109                     objetos.pop(index)
110                     del obj  # liberar la memoria de obj
111             if new == True:
112                 p = Object.MyObject(oid, cx, cy, max_p_age)
113                 objetos.append(p)
114                 oid += 1
115
116     cv2.imshow('contornos', frameOriginal)
117
118     if cv2.waitKey(1) & 0xFF == ord('q'):
119         break
120
121 cap.release()
122 cv2.destroyAllWindows()
```

Bibliografía

[1] M. Vicente, "Análisis y solución del problema de estacionamiento en el centro de las ciudades," *Informes de la Construcción*, vol. 34, 04 2012.

[2] S. Banerjee, P. Choudekar, y M. K. Muju, "Real time car parking system using image processing," *2011 3rd International Conference on Electronics Computer Technology*, vol. 2, pp. 99–103, 2011.

[3] J. Yick, B. Mukherjee, y D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, num. 12, pp. 2292–2330, 08 2008. [En línea]. Disponible: https://linkinghub.elsevier.com/retrieve/pii/S1389128608001254

[4] J. A. Márquez Domínguez y E. M. Flórez Barragán, "Desarrollo de un prototipo funcional para el monitoreo de la señal y conectividad de dispositivos de conexión inalámbrica en la corporación universitaria del caribe-CECAR," 2017, [Accedido: 2021-01-28]. [En línea]. Disponible: https://repositorio.cecar.edu.co/jspui/handle/123456789/54

[5] M. Bachani, U. M. Qureshi, y F. K. Shaikh, "Performance analysis of proximity and light sensors for smart parking," *Procedia Computer Science*, vol. 83, pp. 385–392, 01 2016, [Accedido: 2021-01-28]. [En línea]. Disponible: http://www.sciencedirect.com/science/article/pii/S1877050916302332

[6] C.-K. Ng, S.-N. Cheong, E. Hajimohammadhosseinmemar, y W.-J. Yap, "Mobile outdoor parking space detection application," *2017 IEEE 8th Control and System Graduate Research Colloquium (ICSGRC)*, 2017.

[7] T. Jensen, H. Schmidt, N. Bodin, K. Nasrollahi, y T. Moeslund, "Parking space occupancy verification - improving robustness using a convolutional neural network," in *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP, (VISIGRAPP 2017)*, 01 2017, pp. 311–318.

[8] O. Bulan, R. Loce, W. Wu, Y. Wang, E. Bernal, y Z. Fan, "Video-based real-time on-street parking occupancy detection system," *Journal of Electronic Imaging*, vol. 22, p. 041109, 10 2013.

[9] C. G. Postigo, J. Torres, y J. M. Menéndez, "Vacant parking area estimation through background subtraction and transience map analysis," *IET Intelligent Transport Systems*, vol. 9, num. 9, pp. 835–841, 11 2015, [Accedido: 2021-01-27]. [En línea]. Disponible: https://onlinelibrary.wiley.com/doi/10.1049/iet-its.2014.0090

[10] (2013, 06) Compatibilidad electromagnética y seguridad funcional en sistemas electrónicos - marcombo, s.a. (ediciones técnicas). [En línea]. Disponible: https://www.marcombo.com/compatibilidad-electromagnetica-y-seguridad-funcional-en-sistemas-electronicos-9788426716439/

[11] C. D. Ramirez, R. S. Bocanegra, y M. S. Sierra, "INTEGRACIÓN DE SENSORES INALÁMBRICOS y DOMÓTICA," p. 71, 2011.

[12] (2018) Diseño e implementación de un protocolo de comunicación domótico para interacción entre sensores y actuadores. [Accedido: 2021-01-29]. [En línea]. Disponible: https://1library.co/document/y6enkl5z-diseno-implementacion-protocolo-comunicacion-domotico-interaccion-sensores-actuadores.html

[13] (2018, 02) Guía de introducción a MQTT con ESP8266 y raspberry pi. [Accedido: 2021-01-29]. [En línea]. Disponible: https://programarfacil.com/esp8266/mqtt-esp8266-raspberry-pi/

[14] (2019) ¿qué es MQTT? su importancia como protocolo IoT. [Accedido: 2021-01-29]. [En línea]. Disponible: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

[15] jecrespom. (2018, 11) Broker MQTT. [Accedido: 2021-01-29]. [En línea]. Disponible: https://aprendiendoarduino.wordpress.com/category/broker-mqtt/

[16] (2018) NodeMCU ESP8266 specifications, overview and setting up. [Accedido: 2021-01-29]. [En línea]. Disponible: https://www.make-it.ca/nodemcu-arduino/nodemcu-details-specifications/

[17] L. Ada. (2018, 08) photocells-932884.pdf. [Accedido: 2021-02-0]3. [En línea]. Disponible: https://www.mouser.com/datasheet/2/737/photocells-932884.pdf

[18] Cómo controlar módulos p10. [Accedido: 2021-02-25]. [En línea]. Disponible: https://www.proyectoselectronicoss.com/modulos-led-p10-16x32/

[19] T. Instruments. (2000, 03) cd4051-ti.pdf. [Accedido: 2021-02-03]. [En línea]. Disponible: https://global.oup.com/us/companion.websites/fdscontent/uscompanion/us/pdf/microcircuits/students/logic/cd4051-ti.pdf

[20] (2017, 02) uln2803a.pdf. [Accedido: 2021-01-30]. [En línea]. Disponible: https://www.ti.com/lit/ds/symlink/uln2803a.pdf?ts=1612000681966

[21] (2015) MC74hc595-d.pdf. [Accedido: 2021-01-30]. [En línea]. Disponible: https://www.onsemi.com/pub/Collateral/MC74HC595-D.PDF

[22] (2012) User guide of ansi/esda/jedec js-001 human body model testing of integrated circuits. [Accedido: 2021-03-03]. [En línea]. Disponible: https://www.jedec.org/sites/default/files/JTR001-01-12%20Final.pdf

[23] (2019, 06) Detalles del modelo de máquina (MM). [Accedido: 2021-01-31]. [En línea]. Disponible: http://www.esdunlimited.com/mm.html

[24] (2017) 200206 raspberry pi 3 model b plus product brief PRINT&DIGITAL.pdf. [Accedido: 2021-01-31]. [En línea]. Disponible: https://static.raspberrypi.org/files/product-briefs/200206+Raspberry+Pi+3+Model+B+plus+Product+Brief+PRINT&DIGITAL.pdf

[25] (2007) Cámara web genius iSlim 1300, 1.3 mpx, micrófono, USB - 32200163101. [Accedido: 2021-03-02]. [En línea]. Disponible: https://intercompras.com/p/camara-web-genius-islim-mpx-microfono-usb-44453

[26] (2019) IP ratings | IEC. [Accedido: 2021-03-03]. [En línea]. Disponible: https://www.iec.ch/ip-ratings

[27] (2020, 05) Compatibilidad electromagnética (EMC). [Accedido: 2021-02-01]. [En línea]. Disponible: https://www.4-layers.com/blog/compatibilidad-electromagnetica-emc/

[28] (2018) Tecnicas de diseño en EMC 2018.pdf. [Accedido: 2021-02-01]. [En línea]. Disponible: http://materias.fi.uba.ar/6610/Apuntes/Tecnicas%20de%20disenio%20en%20EMC%202018.pdf

[29] (2017, 07) Transformaciones morfológicas. [Accedido: 2021-03-02]. [En línea]. Disponible: https://unipython.com/transformaciones-morfologicas/

[30] F. J. G. Fernández, "RECONOCIMIENTO DE OBJETOS EN UNA COCINA CON UNA WEBCAM," p. 126, 11 2009.

[31] OpenCV: Cómo utilizar métodos de resta en segundo plano. [Accedido: 2021-02-02]. [En línea]. Disponible: https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html

[32] P. KaewTraKulPong y R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-Based Surveillance Systems*, P. Remagnino, G. A. Jones, N. Paragios, y C. S. Regazzoni, Eds. Springer US, 05 2002, pp. 135–144, [Accedido: 2021-02-02]. [En línea]. Disponible: http://link.springer.com/10.1007/978-1-4615-0913-4_11

[33] (2014, 11) Background subtraction — OpenCV 3.0.0-dev documentation. [Accedido: 2021-02-02]. [En línea]. Disponible: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html

[34] X. Ling, J. Sheng, O. Baiocchi, X. Liu, y M. E. Tolentino, "Identifying parking spaces detecting occupancy using vision-based IoT devices," 06 2017, pp. 1–6.

[35] S.-F. Lin, Y.-Y. Chen, y S.-C. Liu, "A vision-based parking lot management system." IEEE, 10 2006, pp. 2897–2902, [Accedido: 2021-03-29]. [En línea]. Disponible: http://ieeexplore.ieee.org/document/4274321/

[36] Tătulea, Călin, Brad, Brâncovean, y Greavu, "An image feature-based method for parking lot occupancy," *Future Internet*, vol. 11, num. 8, p. 169, 08 2019, [Accedido: 2021-03-29]. [En línea]. Disponible: https://www.mdpi.com/1999-5903/11/8/169

[37] N. S. Narayanan, F. Shaji, J. Praseedom, y P. Thiyagarajan, "Intelligent parking and power management system using sensors," 03 2019, pp. 1–7.

[38] A. Khanna y R. Anand, "IoT based smart parking system," 01 2016.

[39] I. LABS. (2018, 04) RS-200. [Accedido: 2021-02-17]. [En línea]. Disponible: https://es.ietlabs.com/pdf/Datasheets/RS-CS-LS.pdf

[40] (2021) Software y herramientas de diseño de PCB para construir la próxima generación en electrónica. [Accedido: 2021-02-19]. [En línea]. Disponible: https://www.altium.com/es

[41] (2005) LM7809 datasheet - 3-terminal 1a positive voltage regulator. [Accedido: 2021-02-1]9. [En línea]. Disponible: https://www.digchip.com/datasheets/parts/datasheet/670/LM7809.php

[42] T. Instruments. (2020) lm1117.pdf. [Accedido: 2021-02-19]. [En línea]. Disponible: https://www.ti.com/lit/ds/symlink/lm1117.pdf?ts=1613712009968&ref_url=https%253A%252F%252Fwww.google.com%252F

[43] (2021) Software. [Accedido: 2021-02-19]. [En línea]. Disponible: https://www.arduino.cc/en/software

[44] (2021) Fusion 360 | 3d CAD, CAM, CAE & PCB cloud-based software | autodesk. [Accedido: 2021-02-20]. [En línea]. Disponible: https://www.autodesk.com/products/fusion-360/overview,https://www.autodesk.com/products/fusion-360/overview

[45] (2020, 06) Filamento PETG: todo sobre el filamento 3d. [Accedido: 2021-02-20]. [En línea]. Disponible: https://all3dp.com/es/1/filamento-petg-impresion-3d/

[46] (2002) Prensaestopa plastica. [Accedido: 2021-02-21]. [En línea]. Disponible: https://www.grainger.com.mx/static/ft/20028931_TD.PDF

[47] (2021) Plástico líquido -resina de poliuretano - $20,00. [Accedido: 2021-02-24]. [En línea]. Disponible: https://articulo.mercadolibre.com.ec/MEC-429503389-plastico-liquido-resina-de-poliuretano-_JM

[48] admin_wp. (2014, 06) ¿qué es la resina de poliuretano? [Accedido: 2021-02-24]. [En línea]. Disponible: http://rubept.com/es/que-es-la-resina-de-poliuretano/

[49] (2017, 11) NodeMCU i2c with arduino IDE | NodeMCU. [Accedido: 2021-02-25]. [En línea]. Disponible: https://www.electronicwings.com/nodemcu/nodemcu-i2c-with-arduino-ide

[50] (2014) Uno r3 SMD with CH340 USB. [Accedido: 2021-04-15]. [En línea]. Disponible: https://protosupplies.com/product/uno-r3-smd/

[51] sfeelectronicsFollow. (2017) Display text at p10 LED display using arduino. [Accedido: 2021-02-25]. [En línea]. Disponible: https://www.instructables.com/Display-Text-at-P10-LED-Display-Using-Arduino/

[52] (2021) GLCD font creator. [Accedido: 2021-02-25]. [En línea]. Disponible: http://www.mikroe.com/glcd-font-creator