

PROTOTIPO DE SISTEMA DE GESTION DE APARCAMIENTO EN EL
CONTEXTO DE CIUDADES INTELIGENTES

ALEJANDRO ESCOBAR HENAO

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
PROGRAMA ACADÉMICO DE INGENIERÍA ELECTRÓNICA
SANTIAGO DE CALI
2018

PROTOTIPO DE SISTEMA DE GESTION DE APARCAMIENTO EN EL
CONTEXTO DE CIUDADES INTELIGENTES

ALEJANDRO ESCOBAR HENAO

Trabajo de grado de investigación
Grupo SISTEL-UV
Ingeniería Electrónica
Universidad del Valle

Director
FABIO GERMAN GUERRERO

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
PROGRAMA ACADÉMICO DE INGENIERÍA ELECTRÓNICA
SANTIAGO DE CALI
2018

Agradecimientos

Expreso mi más profundo y sincero agradecimiento a todas aquellas personas que con su ayuda han colaborado en la realización del presente trabajo, en particular al Profesor Fabio German Guerrero, director de tesis, por la orientación, el seguimiento y la supervisión de la misma, pero sobre todo por la motivación y el apoyo recibido.

Un agradecimiento muy sincero por la comprensión, paciencia y el ánimo recibidos de mi familia y amigos.

Tabla de contenido

Capítulo 1.....	13
Marco Referencial.....	13
Antecedentes.....	16
Marco Teórico.....	17
Conceptos de Smart City aplicado hacia los sistemas de aparcamiento.....	17
El papel de las TIC en las ciudades inteligentes.....	18
Protocolos de comunicación empleados en el contexto de Smart City	20
Recomendaciones de la Unión Internacional de las Telecomunicaciones (ITU)	20
Estrategias existentes para el desarrollo de un sistema de aparcamiento	22
Tecnología RFID	22
Tecnología Video Cámaras	24
Tecnología Vanet	25
Tecnología GPS	27
Tecnología WSN	28
Clasificación del sistema a implementar	29
Clasificar algunas estrategias y contextualización hacia Smart City	29
Capítulo 2.....	32
Requerimientos del sistema	32
Requerimientos funcionales.....	32
Requerimientos no funcionales	33
Diseño del sistema	34
Diseño del prototipo de sistema.....	34
Webcam	35
Raspberry PI3 Model B.....	35
Alojamiento (Hosting)	36
Tablas de registro en la base de datos	37
Aplicación Android.....	38
Diseño metodológico del sistema	41
Implementación del sistema	42
Parqueadero prototipo físico	42

Instalación de las webcam en Rasbian	43
Procesamiento de la imagen	44
Alojamiento web publico (Hosting)	53
Aplicación Android	55
Capítulo 3.....	59
Resultados y Discusión	59
Objetivos para realizar pruebas	59
Metodología a seguir	59
Aspectos a examinar	60
Resultados esperados	60
Pruebas del sistema	61
Limitaciones	67
Bajo nivel de luz	67
Fuente de luz directa	68
Conclusiones	70
Anexos.....	72
Bibliografía	78

Lista de tablas

Tabla 1. Ámbitos que ofrecen las ciudades inteligentes.....	13
Tabla 2. Ejemplos de tecnologías relacionadas con las TIC.....	18
Tabla 3. Ventajas, sensores, dispositivos y software empleados en la tecnología RFID. [18][19][20].	23
Tabla 4. Ventajas, sensores, dispositivos y funcionamiento de las video cámaras. [21], [22].	25
Tabla 5. Ventajas, sensores, dispositivos y funcionamiento de la tecnología Vanet. [24], [25], [26].....	26
Tabla 6. Ventajas, sensores, dispositivos y funcionamiento de la tecnología GPS. [27]....	28
Tabla 7. Ventaja, sensores, dispositivos y funcionamiento de la tecnología WSN. [29] [30].	29
Tabla 8. Contextualización de tecnologías hacia Smart City, para un parqueadero de 16 plazas.....	30
Tabla 9. Datos obtenidos al ejecutar algoritmo presentado en la Ilustración 25.....	51
Tabla 10 Datos para obtener el valor máximo y mínimo implementado en el Diagrama de la Ilustración 25.....	52

Lista de figuras

Ilustración 1. Diagrama de campos para ser aplicados a la conformación de Ciudades inteligentes.	14
Ilustración 2. Red de tecnología RFID.	23
Ilustración 3. Red de tecnología Video Cámara.	24
Ilustración 4. Red de tecnología Vanet.	26
Ilustración 5. Red de tecnología GPS.	27
Ilustración 6. Red de tecnología WSN.	28
Ilustración 7. Idea de prototipo de parqueadero.	34
Ilustración 8 WebCam MICROSOTF MODELO 1407, usada para toma de imágenes en el prototipo de parqueadero. [29]	35
Ilustración 9 WebCam GENIUS FASECAM 310, usada para toma de imágenes en el prototipo de parqueadero. [28]	35
Ilustración 10 Tarjeta Raspberry Pi 3 Model b para realizar el procesamiento de imágenes.	36
Ilustración 11 Característica de la tabla "Usuarios"	37
Ilustración 12 Característica de la tabla "Estadoparqueadero"	38
Ilustración 13 Diagrama de flujo que debe seguir la plantilla 1 de la aplicación Android.	39
Ilustración 14 Diagrama de flujo que debe seguir la plantilla 2 de la aplicación Android.	40
Ilustración 15 Diagrama de flujo que debe seguir la plantilla 3 de la aplicación Android.	41
Ilustración 16 Diseño de estructura para gestión de aparcamiento.	42
Ilustración 17 Diseño de prototipo de parqueadero completo.	43
Ilustración 18 Diseño de prototipo de parqueadero por módulos.	43
Ilustración 19 Diagrama de procesos para adquisición y manipulación de la imagen.	44
Ilustración 20 Imagen del parqueadero a escala de grises.	45
Ilustración 21.. Imagen con el Filtro Gaussiano.	46
Ilustración 22. Imagen con detección de bordes.	46
Ilustración 23 Diagrama de flujo para obtener los bordes.	48
Ilustración 24. Bordes guardados de las imágenes segmentadas.	49
Ilustración 25 Diagrama de Flujo para determinar el estado del parqueadero.	50
Ilustración 26 Implementación de la GUI de la plantilla 1.	56
Ilustración 27. Segunda plantilla de la aplicación Android.	57
Ilustración 28. Tercera plantilla de la aplicación Android.	58
Ilustración 29. Resultado del parqueadero cuando se coloca un vehículo en la plaza 1 en el módulo 1.	61
Ilustración 30 registro general del estado de parqueadero.	61
Ilustración 31. Consulta del parqueadero a través de la aplicación Android cuando el ultimo Id es 164.	62
Ilustración 32 Resultado del parqueadero cuando se colocan vehículos en las plazas 1 y 2 en el módulo 1.	62
Ilustración 33 Consulta del parqueadero a través de la aplicación Android cuando el ultimo Id es 165.	63

Ilustración 34. Resultado del parqueadero cuando se colocan vehículos en las plazas 1 y 2 en el módulo 1	63
Ilustración 35.Consulta del parqueadero a traves de la aplicación Android cuando el ultimo Id es 166.....	64
Ilustración 36.Resultado del parqueadero cuando se colocan vehículos en las plazas 1,2 y 3 en el módulo 1.....	64
Ilustración 37.Consulta del parqueadero a través de la aplicación Android cuando el ultimo Id es 167.....	65
Ilustración 38. Resultado del parqueadero cuando se colocan vehículos en las plazas 1,2,3 y 4 en el módulo 1	65
Ilustración 39.Consulta del parqueadero a través de la aplicación Android cuando el ultimo Id es 169.....	66
Ilustración 40.Resultado del parqueadero cuando se colocan vehículos en las plazas 1, 2,3 ,4 y 6 en el módulo 1 y plazas 2, 5 y 7 en el módulo 2.....	66
Ilustración 41.Consulta del parqueadero a través de la aplicación Android cuando el ultimo Id es 170.....	66
Ilustración 42 Consulta en la base de datos	67
Ilustración 43. Captura del parqueadero con bajo nivel de iluminación.	68
Ilustración 44.Resultado de la detección de bordes del parqueadero con bajo nivel de iluminación.....	68
Ilustración 45. Captura del parqueadero con fuente de luz directa.....	68
Ilustración 46. Resultado de la detección de bordes con fuente directa.	69
Ilustración 47. Plazas que presentan falla con fuente de luz directa.	69

Resumen

A medida que pasa el tiempo, las necesidades del ser humano van en aumento; desde la invención de la computadora, el primer celular y los diferentes artefactos tecnológicos han sido parte de la solución a dichas necesidades que surgen en el diario vivir. Hoy por hoy, el sector urbano cuenta con la prestación de múltiples servicios, pero sin tener en consideración factores como: la gestión de recursos, costo, estructuración y calidad de los mismos.

Uno de estos servicios es el sistema de aparcamiento donde la deficiencia del monitoreo y control de las plazas libres genera pérdida de tiempo para los usuarios y ocasiona congestión de tráfico. Es por eso que en este trabajo se propone la construcción de un parqueadero prototipo de área plana en el cual se garantiza la gestión y control de plazas libres bajo los lineamientos instaurados por la Unión Internacional de las Telecomunicaciones enfocados hacia las Ciudades inteligentes; lo anterior se realiza por medio de procesamiento de imágenes en tarjeta de procesamiento Raspberry Pi 3 Model B, teniendo como sensor una WebCam reduciendo costos en la construcción e implementación. Los datos generados en el procesamiento son guardados en una Base de Datos permitiendo al usuario el monitoreo del parqueadero a través de una aplicación Android. Este proyecto pretende contribuir al fortalecimiento e instauración de uno de los campos de las Ciudades inteligentes, esperando que en un futuro no muy lejano sea aplicado en la ciudad.

Palabras Claves: Ciudades Inteligentes, Unión Internacional de las Telecomunicaciones (ITU), Raspberry Pi 3 Model B, Parqueadero prototipo, control de plazas libres.

ABSTRACT

As the time happens, the needs of the human being are increasing, from the invention of the computer, the first cellular phone one and the different technological appliances have been a part of the solution to the above mentioned needs, that arise in the dialy life. Now days today the urban sector possesses the presentation of multiple services, but without having in consideration factors as: the resource management, cost, structure and quality of the same one.

One of these services is the parking system where the deficiency of the monitoring and control of the free spaces, generating loss of time for the users and causing congestion of traffic inside the same one. That is the reason because the construction of a parking prototype which guarantees the management and control of free spaces under the

guidelines established by the International Union of the Telecommunications focused towards the intelligent Cities. The items are done using image processing in card of Raspberry Pi 3 Model B processing, having as sensor a WebCam, reducing costs in the construction and implementation. The information generated in the processing are stored in a Database allowing the monitoring of the parking to the user across an application Android. This project tries to contribute to the strengthening and restoration of one of fields of the Smart cities hoping that in a not very distant future it is applied in the City.

Key words: Intelligent Cities, International Union of the Telecommunications (ITU), Raspberry Pi 3 Model B, Parking prototype, control of free spaces.

INTRODUCCIÓN

Los sistemas de aparcamiento son un terreno o área específica que pueden ser de una pequeña planta o área plana o un edificio completo, incluyendo no solo un diseño estructural, sino también un diseño mecánico y electrónico, que en conjunto tratan de proporcionar un servicio de parqueadero lo más funcional posible.

En alguno de estos sistemas no se les da demasiada importancia y a medida que su flujo incrementa, la desorganización vehicular se hace evidente y pasan a ser zonas de inseguridad tanto para personas como para automóviles.

La mayoría de parqueaderos sólo cuentan con un control básico de las entradas y salidas con detectores de vehículos pero no cuentan con un sistema de gestión de plazas libres a pesar de los avances tecnológicos con los que se cuenta hasta el momento [1]. Se debe tener en cuenta que las tecnologías con mayor eficiencia y versatilidad para estos casos son las “tecnologías inalámbricas” como base para la construcción de redes inalámbricas.

Este tipo de redes han tomado gran importancia y han despertado el interés de muchos en cuanto a cómo poder comunicar diferentes equipos sin necesidad de utilizar medio cableado dando mayor comodidad al usuario al permitir el movimiento del sistema tecnológico que la conforma y facilita la conexión entre sistemas diferentes; sin embargo, presenta una desventaja que a medida que avanza el tiempo se hace menos notoria y es en cuanto a sus velocidades de transmisión de información y la seguridad de datos, ya que el medio de propagación es el ambiente y se encuentran expuestos a muchas interferencias.

La movilidad que permite este tipo de redes depende mucho del sistema empleado, pero, a pesar de todo, son muy útiles en lugares donde no es posible emplear cableado. A medida que se adapta la tecnología a un sistema, este logra mayor eficiencia y automatización en los procesos y servicios que brinda al usuario, de modo que el sistema se puede caracterizar por ser inteligente.

Este libro, en su capítulo 1, muestra las especificaciones y lineamientos que debe tener un sistema de aparcamiento para poder trascender o ser aplicado en una ciudad inteligente; además, se realiza un análisis entre los métodos utilizados en dichos sistemas presentando el funcionamiento, tecnologías o sensores empleados, ventajas y desventajas de cada uno de los métodos. En el capítulo 2 se procede a implementar uno de los métodos estudiados teniendo en cuenta los lineamientos de la Unión Internacional de las Telecomunicaciones. Luego, en el capítulo 3 se presentan los resultados y conclusiones obtenidas en el desarrollo de este proyecto, y se proponen trabajos futuros que podrían complementarlo.

OBJETIVO GENERAL

Diseñar un sistema prototipo de gestión a través de Internet para un parqueadero de 16 plazas basado en el concepto de parqueaderos inteligentes de la Unión Internacional de las Comunicaciones (ITU).

OBJETIVOS ESPECÍFICOS

1. Construir un prototipo de parqueadero a escala (maqueta), de 16 bahías o plazas usando las tecnologías de comunicación Wi-Fi / Ethernet y un método de instrumentación y sensado establecido de bajo coste.
2. Diseñar y desarrollar el servicio de gestión en la nube del prototipo de parqueadero inteligente, para acceder a la información desde dispositivos móviles.
3. Desarrollar una aplicación para equipos móviles que permita consultar y verificar el servicio de gestión de parqueo aplicado al prototipo de parqueadero.

CAPITULO 1.

1.1. Marco Referencial

A continuación se plantean cuatro conceptos que son fundamentales para la conceptualización del proyecto: Ciudades inteligentes, sistemas inteligentes, campos de aplicación y la gestión de aparcamiento.

CIUDADES INTELIGENTES

Las ciudades inteligentes, citado por la Unión Internacional de las telecomunicaciones (ITU), son:

“... aquellas ciudades que utiliza tecnologías de la información y la comunicación (TIC) y otros medios para mejorar la calidad de vida, la eficiencia de la operación y los servicios urbanos, y la competitividad, garantizando al mismo tiempo que cumple con las necesidades de las generaciones presentes y futuras con respecto a los aspectos económicos, sociales y medioambientales”.[2]

CAMPOS DE APLICACIÓN

Esta definición se puede aplicar en varios campos como población, habitabilidad, movilidad, medio ambiente, gobierno y economía, desencadenando varias actividades (ver tabla 1). Las ciudades inteligentes se basan en la implementación de arquitecturas tecnológicas e incorporación de sistemas inteligentes en diversos contextos para mejorar la calidad de vida y la eficacia de los procesos instaurados o empleados en la ciudad, llevando a la creación de ciudades sostenibles.[3]

Tabla 1. Ámbitos que ofrecen las ciudades inteligentes

Campos	Actividades relacionadas
Población	<ul style="list-style-type: none">• Formación.• Creatividad.• Participación en la vida pública.
Habitabilidad	<ul style="list-style-type: none">• Condiciones socio- sanitarias.• Oferta cultural.• Seguridad Ciudadana.• Calidad de vivienda.• Facilidades educativas.• Turismo.
Movilidad	<ul style="list-style-type: none">• Transporte sostenible.• Control de tráfico inteligente.• Infraestructura TIC.• Gestión del tráfico en tiempo real.

	<ul style="list-style-type: none"> • Gestión de aparcamiento.
Medio Ambiente	<ul style="list-style-type: none"> • Gestión de recursos sostenibles. • Segregación de residuos innovadora. • Gestión de parques y jardines. • Medición de parámetros ambientales. • Reducción de la contaminación. • Predicción meteorológica.
Gobierno	<ul style="list-style-type: none"> • Transparencia. • Participación ciudadana. • Estrategias políticas Claras.
Economía	<ul style="list-style-type: none"> • Productividad. • Flexibilidad laboral.

Cada una de estas actividades mencionadas en la tabla anterior representa la base o fundamento de varios sistemas inteligentes que trabajan en conjunto para prestar servicios que puedan ser utilizados o consumidos por muchas personas o entidades, de manera que estén interconectados para llegar a conformar una ciudad inteligente (ver ilustración 1) [4].

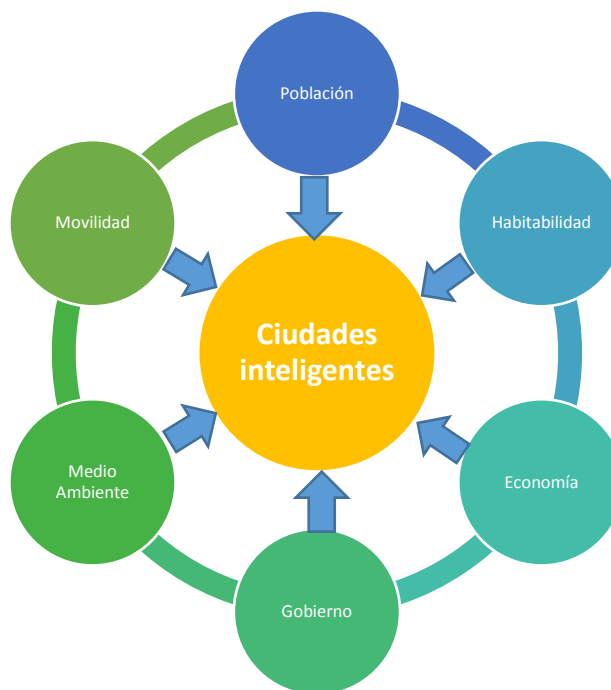


Ilustración 1. Diagrama de campos para ser aplicados a la conformación de Ciudades inteligentes.

EJEMPLOS DE CIUDADES INTELIGENTES EN EL MUNDO

En el artículo titulado “*Cinco casos exitosos de Ciudades inteligentes*” publicado en la página web www.Enter.com [5], se muestra cinco ciudades que se han destacado por implementar tecnologías verdes y por utilizar mejor los recursos.

“Tokio: es considerada como ciudad inteligente ya que la ciudad cuenta con un servicio que brinda a los usuarios alertas tempranas para prepararse ante la presencia de un sismo, dicha alarma se envía a través de las redes por medio de un mensaje de texto, también por radio o tv.

Presenta otro elemento tecnológico como la NFC, todo el transporte público se puede pagar con NFC y varias tiendas reciben pagos por este medio; en este mismo sentido se está implantando tecnología verde.”

“Paris: tiene los primeros puestos en la categoría Urbes verdes, se caracteriza por la implementación de alquiler de bicicletas para no usar tanto el automóvil, presenta otro servicio como trazar una ruta y reservar uno de los vehículos que están ubicados por toda la ciudad y se puede acceder al servicio con la ayuda de aplicaciones IOS y Android.”

“Nueva York: puso en marcha una plataforma interactiva colocando pantallas inteligentes en casi toda la ciudad para informar sobre noticias. La ciudad se alió con IBM que está dedicado a encontrar la forma de ayudar a las empresas. Otra forma es prevenir incendios, devolución de impuestos”.

“Santander: implementó señales de tránsito inteligentes y si alguno de los sensores está fallando se puede avisar al administrador por medio de un dispositivo móvil.”

“Toronto: Desarrollo una de iniciativa para sanear las redes del metro utilizando gas proveniente de los vertederos empleado por las furgonetas y transporte de mayor consumo.”

PARQUEADEROS INTELIGENTES

Partiendo de esto, se puede decir que, en el campo de movilidad, la gestión de aparcamiento contribuye al diseño de las ciudades inteligentes, ya que responden a la demanda de disponibilidad de espacios en los parqueaderos. Así pues, y haciendo referencia a la definición propuesta por la ITU anteriormente citada, esta gestión mejora la prestación de un servicio necesario en las generaciones presentes, reduciendo el tiempo en la ubicación de dichos espacios y mejorando la congestión de tráfico en el interior del parqueadero.

En el artículo “*Internet of Things for Smart Cities*” [6] se hace referencia a un aparcamiento inteligente como un servicio que: “... se basa en la ubicación de un conjunto de sensores y pantallas inteligentes que los automovilistas puedan ver información a lo largo del camino hacia el estacionamiento”. En complemento, un

sistema inteligente de gestión debe presentar detectores de vehículos o plazas libres como elemento esencial.

Algunos de estos detectores son los denominados bucles inductivos que presentan problemas en la instalación y mantenimiento; en la práctica es uno de los más usados hoy en día [7]. También se puede emplear tecnologías de comunicación de corto alcance como son los identificadores de radio frecuencia (RFID) o de campo cercano (NFC), en el que es posible realizar un sistema de verificación electrónica en las ranuras reservadas para personas residentes o personas con discapacidad [6].

Otro mecanismo para la detección de vehículo son las tecnologías inalámbricas, tales como WSN (Wireless Sensor Network), que presentan facilidad para acoplarse y no necesitan de cableado [7], [8].

La tecnología inalámbrica es una red de sensores que contiene un gran número de nodos que se encuentran desplegados en la zonas de detección y otros nodos en los que se puede recopilar, procesar y transmitir información de manera automatizada y rentable [1].

1.2. Antecedentes

El desarrollo de las tecnologías ha permitido minimizar el tráfico en el aparcamiento, aplicando varios sistemas que utilizan como eje central el monitoreo dinámico. Estas soluciones se centran en los estacionamientos actuales, empleando unas aplicaciones de red y sensores de comunicación inalámbrica.

Lee, Yoon y Ghosh hicieron la propuesta de un enfoque híbrido [9] para un sistema de aparcamiento inteligente utilizando una combinación de ultrasonido y sensores magnéticos. Ellos demostraron que a partir de los híbridos se generaban soluciones prácticas y precisas, el alcance de su trabajo se limitó a la detección de vehículos utilizando WSN (Wireless Sensor Network) que proporciona de manera inteligente la solución de gestión de aparcamiento.

D-Systems Project [10] presentó varios problemas al emplear un Sistema de red WSN, a causa de los sensores magnéticos que se utilizaron. En este trabajo se implementa un sistema de gestión de aparcamiento por medio de una arquitectura por niveles y se proponen varias recomendaciones sobre el uso de enrutamientos dinámicos y robustos.

Iris-Net [7] propuso una red de sensores en toda el área de arquitectura utilizando cámaras de video, micrófonos y detectores de movimiento. Los dispositivos de detección se utilizan para detectar la disponibilidad de plazas de aparcamiento. A través de sus aplicaciones web, los clientes pudieron adquirir información del estacionamiento en tiempo real, pero, debido a la gran cantidad de datos; se necesitaba una mayor potencia y ancho de banda de comunicación de la que se había empleado, siendo esto una limitación de la red WSN.

Jatuporn Chinrungrueng [8] propuso la WSN óptico como un sistema de conteo de vehículos en aparcamientos inteligentes, Para lograr dicho conteo utilizó un sensor óptico para el nodo del sensor inalámbrico. También consiguió diferenciar el tamaño y la identificación de diferentes vehículos.

S S. V. Srikanth, Pramod P. J, Dileep K. P, Tapas S, Mahesh U. Patil, Sarathandra Babu N [1] proponen un Sistema de gestión de aparcamiento inteligente basado en la tecnología de red inalámbrica que proporcionan características avanzadas como la supervisión a distancias de aparcamiento, orientación automática y el mecanismo de reserva de aparcamiento. Los usuarios tienen la posibilidad de encontrar una vacante en el estacionamiento en tiempo real.

Tang [11] describe un sistema de gestión de aparcamiento inteligente basado en sensores inalámbricos, estos sensores son desplegados por todo el parqueadero y son utilizados para detectar y controlar la ocupación de la zona de aparcamiento. Los sensores envían información periódicamente a una base de datos y en ella se puede acceder para realizar diversas funciones de gestión tales como encontrar plazas vacías, gestión de seguridad e informe estadístico.

Pala y Inanç [12] utilizan RFID para el registro de entrada y salida, la autorización de entrada depende de los registros del vehículo.

1.3. MARCO TEÓRICO

1.3.1. Conceptos de Smart City aplicado hacia los sistemas de aparcamiento

Para poder comprender el concepto de Smart City, se puede hacer referencia a varias definiciones, la primera está dada por la International Business Machines Corp (IBM) donde afirma que *“Ciudad inteligente es aquella que hace un uso óptimo de toda la información interconectada que se encuentra disponible en la actualidad para poder comprender y controlar sus operaciones y optimizar el uso de recursos.”*

La segunda definición está dada por la Unión internacional de las Telecomunicaciones en la que dice que *“... aquellas ciudades que utilizan tecnologías de la información y la comunicación (TIC) y otros medios para mejorar la calidad de vida, la eficiencia de la operación y los servicios urbanos y la competitividad, garantizando al mismo tiempo que cumple con las necesidades de las generaciones presentes y futuras con respecto a los aspectos económicos, sociales y medioambientales.”*

Haciendo una comparación de las dos definiciones, el concepto se centra en el análisis y uso de las tecnologías modernas, sean de nuevas creación o ya existentes que se utilizan en el diario vivir, basado en una plataforma que genera información y datos denominada (TIC).

Las Tecnologías de la Información y Comunicación son aquellas tecnologías que nos permiten acceder, producir, guardar, prestar y transferir información, facilitando la comprensión del funcionamiento de la ciudad en términos de eficiencia, consumo, recursos y servicios que en la mayoría de casos se encuentran presentes en toda la ciudad. [13]

Los dispositivos y servicios (TIC) son facilitadores o proveedores que permiten que las soluciones para garantizar un servicio sean realizables a través de un sistema tecnológico, esto no quiere decir que sólo por tener un PC o un teléfono inteligente implica que el sistema implicado para la solución sea inteligente.

1.3.2. El papel de las TIC en las ciudades inteligentes

Las TIC juegan un papel importante proporcionando una plataforma virtual para poder crear una red de información y de conocimiento. La información obtenida por dichas plataformas sirve como referencia para tomar decisiones y orientar la solución a diversos fines para mejorar la calidad de vida de los ciudadanos y de la sociedad en conjunto.

Dentro de una ciudad se pueden encontrar múltiples sistemas (elementos de infraestructura) y son considerados como subredes de una red más amplia denominada “sistemas de sistemas” o “red de redes”, cuando ellos son integrados a las TIC son considerados como “la internet de las cosas (IoT)” para Ciudades. [14]

Las tecnologías de información (IT) o una red de comunicaciones de datos, donde la corriente principal de los procesos está basados en las (TIC) y puede emplearse con algunas modificaciones, lo anterior se asemeja a tener varios sub-sistemas de un sistema en común que se comportan como una red y presentan interconectividad entre sus nodos para llegar a una característica final.

La infraestructura de las TIC incluye el hardware y componentes de software (infraestructura de red, computación en la nube, software de aplicaciones y dispositivos de acceso). A manera de ejemplo se puede listar algunas tecnologías relacionadas con las comunicaciones (*ver tabla 2*).

Tabla 2. Ejemplos de tecnologías relacionadas con las TIC.

INFRAESTRUCTURA	TECNOLOGIAS RELACIONADAS
<i>Bienes inmuebles y edificios</i>	<ul style="list-style-type: none"> • Las sinergias entre la eficiencia energética, el confort y la seguridad • Construcción como una red: La integración de múltiples tecnologías (HVAC, iluminación, cargas de enchufe, fuego, seguridad, movilidad, renovable, almacenamiento, materiales, calidad del aire interior, etc.)

	<ul style="list-style-type: none"> • Software: Eficiencia, automatización y control, análisis de datos.
<i>Industrial y de fabricación</i>	<ul style="list-style-type: none"> • interoperabilidad de datos • La producción sostenible y las emisiones cero • Los sensores conectados en red y la computación en nube • Fábricas del Futuro
<i>Energía y servicios Públicos</i>	<ul style="list-style-type: none"> • Red inteligente y medición inteligente: Generación / distribución / medición • Comunicaciones inalámbricas • Análisis y políticas • El equilibrio de carga, la descentralización y la cogeneración
<i>Aire, agua y gestión de residuos</i>	<ul style="list-style-type: none"> • Sistemas de información del Agua (WIS) • Agua, ahorro de energía y residuos esquema de optimización integrada • Las redes de sensores para los sistemas de agua y aire
<i>Seguridad y protección</i>	<ul style="list-style-type: none"> • Análisis de vigilancia de vídeo. • Comunicación fluida durante los desastres naturales y artificiales
<i>Cuidado de la salud</i>	<ul style="list-style-type: none"> • Hospitales inteligentes • Cuidado de la salud en tiempo real incluyendo la analítica • Inicio y la atención sanitaria a distancia, incluida la vigilancia • Gestión de documentos electrónicos
<i>Educación</i>	<ul style="list-style-type: none"> • El aprendizaje flexible en un entorno de aprendizaje interactivo • El acceso a los contenidos digitales de clase mundial en línea utilizando las tecnologías de colaboración • mooc (MOOC)
<i>Movilidad y transporte</i>	<ul style="list-style-type: none"> • Tecnologías de transporte inteligentes en la edad de las ciudades inteligentes: <ul style="list-style-type: none"> • Gestión del tráfico: Seguimiento y enrutamiento • Gestión de aparcamiento: Seguimiento y enrutamiento • En tiempo real vinculación con las emisiones, patrones de tráfico, menor consumo de combustible.

Se evidencia que la gestión de aparcamiento contiene algunas Tecnologías de la Información y Comunicación y garantizan un papel importante dentro de las ciudades inteligentes.

1.3.3. Protocolos de comunicación empleados en el contexto de Smart City

En el momento de diseñar es necesario tener en cuenta que varios sistemas se deben comunicar entre sí. Esta comunicación se realiza a través de la infraestructura de las TIC, sin importar cuál sea el medio de transporte de los datos se debe garantizar el envío de los mismos, por ejemplo los datos voz, video (multimedia), VoIP (voz sobre IP), PoE (Power over Ethernet) son enviados desde un punto inicial hasta un punto de llegada. Y esto se logra porque las TIC es la base principal que forma la columna vertebral para que los diferentes sensores y actuadores se puedan comunicar utilizando el Internet de las Cosas (IoT).

A medida que se avanza tecnológicamente, la adopción del internet de las cosas está creciendo de manera descontrolada. Se dice que, en años futuros, las cosas estarán interconectadas y cada elemento tendrá una única dirección de protocolo (IP), gracias al protocolo de internet versión 6 (IPV6). Los datos de los sensores se recogerán, se procesarán y se analizarán en tiempo real reduciendo los costos, lo que permite que la Internet de las Cosas (IoT) sea denominada como una infraestructura global.

Al combinar las TIC y la IoT se tiene la capacidad y el potencial de abordar y resolver problemas puesto que proporciona una información confiable proveniente de diversas fuentes para poder agregar, crear, guardar, integrar la información en tiempo real, buscar datos en servidores sociales, bases de datos a través de Internet y ofrecer un conjunto de información y servicios para cada usuario final.

1.3.4. Recomendaciones de la Unión Internacional de las Telecomunicaciones con relación a las Smart City

La ITU, en su informe denominado “UIT-T Y.2221” [15], ha preparado una serie de recomendaciones que se deben tener en cuenta en el momento de crear aplicaciones y brindar servicios.

Independientemente de los tipos de redes, se puede utilizar una conectividad de red NGN o IP, teniendo en cuenta las siguientes conexiones.

- 1) Las redes basadas en conexión IP pueden conectarse a otras redes basadas en conexión IP.
- 2) Las redes de sensores no basadas en IP deben estar conectadas a otras redes utilizando un Gateway para la conversión de protocolos.
- 3) La escalabilidad se debe tener en cuenta para que se pueda soportar el sensor a redes de grandes escalas y así evitar problemas.

Es importante que un área inteligente tenga una red de datos que converja de una manera segura, presentando los siguientes atributos:

- 1) Debe integrar voz, video y datos;
- 2) En la seguridad de datos, cada trama se debe incluir en un cifrado, tanto para la admisión y la detección de intrusiones.
- 3) Data Quality of Service (QoS).
- 4) Gestión de ancho de banda.
- 5) Gestión de dispositivos TIC.

Dentro de las tecnologías típicas que forman parte de una infraestructura de datos / TIC se pueden incluir:

- 1) Infraestructura de red LAN: debe presentar un enfoque modular y jerárquico, simplificando la implementación y administración de la red, permitiendo que varios elementos se puedan replicar, de manera que, si se encuentran fallos, sean fáciles de solucionarlos.
- 2) La conmutación de enrutamiento de núcleo: realiza la función de interconexión de los conmutadores de acceso y la disponibilidad de conectividad de usuario a los servidores y también a Internet.
- 3) La conmutación de acceso: se agrega el tráfico del usuario final, se debe tener en cuenta la capacidad de usuarios que desea tener por medio de portátiles, ordenadores, telefonía IP.
- 4) Redes inalámbricas: se debe facilitar el acceso a wifi.
- 5) Firewalls y routers de la red.

En la seguridad se debe:

- 1) Proteger los servidores (por ejemplo, servidores de correo electrónico, servidores web) de los ataques maliciosos.
- 2) Proteger la red desde la negación externa maliciosa de los ataques del servicio a través de paquetes IP incorrectos o mal formados.
- 3) Proteger a los usuarios contra ataques maliciosos externos a través de datos HTTP incorrectos o mal formados.

Para realizar la conectividad con el mundo externo debe ser necesario el uso de routers para comunicarse con los proveedores de servicio de Internet y para realizar una conectividad interna o local se debe realizar un enrutamiento básico de red privada.

El artículo denominado ITU-T FG-SSC [16], hace énfasis en varios sistemas pertenecientes a las ciudades inteligentes y de manera particular se establece una guía

para un sistema de gestión de aparcamiento donde establece que dicho sistema proporciona a los conductores la ventaja de estacionar sus vehículos de forma rápida, segura y sencilla y al mismo tiempo permite mantener vigilados los estacionamientos. Normalmente se incluye los siguientes aspectos, considerando que cada uno de ellos se logra teniendo en cuenta las recomendaciones anteriores (conectividad, seguridad e infraestructura de datos/TIC):

- 1) El software para atar todos los componentes del sistema de gestión de estacionamiento.
- 2) Sistema de control de acceso automatizado - puertas automáticas, controles de barrera, sistemas de billeteaje.
- 3) Seguridad de video-vigilancia bajo un barrido de vehículos presentes en el sistema.
- 4) Sistemas automatizados.
- 5) El recuento de vehículos en tiempo real y la guía de estacionamiento en tiempo real que muestra las plazas vacías en todos los niveles.

De las anteriores recomendaciones se puede inferir que un sistema perteneciente a las ciudades inteligentes debe proporcionar una interconexión (pública o privada) con varios sistemas garantizando la manipulación y almacenamiento de la información de manera segura y debe responder en tiempo real.

1.4. Estrategias existentes para el desarrollo de un sistema de gestión de aparcamiento.

Un sistema de gestión de aparcamiento puede ser desarrollado por diferentes métodos y cada método depende de la tecnología utilizada. Por lo tanto se consideró realizar tablas donde se especifique las ventajas, los sensores y dispositivos utilizados en cada método. (Ver Tablas 3, 4, 5, 6 y 7)

1.4.1. Tecnología RFID:

Definición: “RFID (siglas de Radio Frequency IDentification, en español IDentificación por RadioFrecuencia) es un sistema de almacenamiento y recuperación de datos remoto que usa dispositivos denominados etiquetas, tarjetas, transpondedores o tags RFID. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio. Las tecnologías RFID se agrupan dentro de las denominadas Auto ID (Automatic IDentification, o Identificación Automática). Las etiquetas RFID son unos dispositivos pequeños, similares a una pegatina, que pueden ser adheridas o incorporadas a un producto, un animal o una persona. Contienen antenas para permitirles recibir y responder a peticiones por radiofrecuencia desde un emisor-receptor RFID. Las etiquetas pasivas no necesitan alimentación eléctrica interna, mientras que las activas sí lo requieren. Una de las ventajas del uso de radiofrecuencia (en lugar, por ejemplo, de infrarrojos) es que no se requiere visión directa entre emisor y receptor, puede decirse que es una

transferencia de datos sin contacto físico". [17] Cuando la etiqueta se aproxima a la antena, se produce inducción entre ellas permitiendo que se transmita la información.[17], [18], [19]

Al utilizar esta tecnología para un sistema de aparcamiento, cada vehículo debe contener una etiqueta RFID, de manera que se pueda registrar cada evento y estado del mismo, esta información se almacena en una base de datos y se adquiere por un usuario a través de un dispositivo móvil. Este proceso se muestra en la ilustración 2.

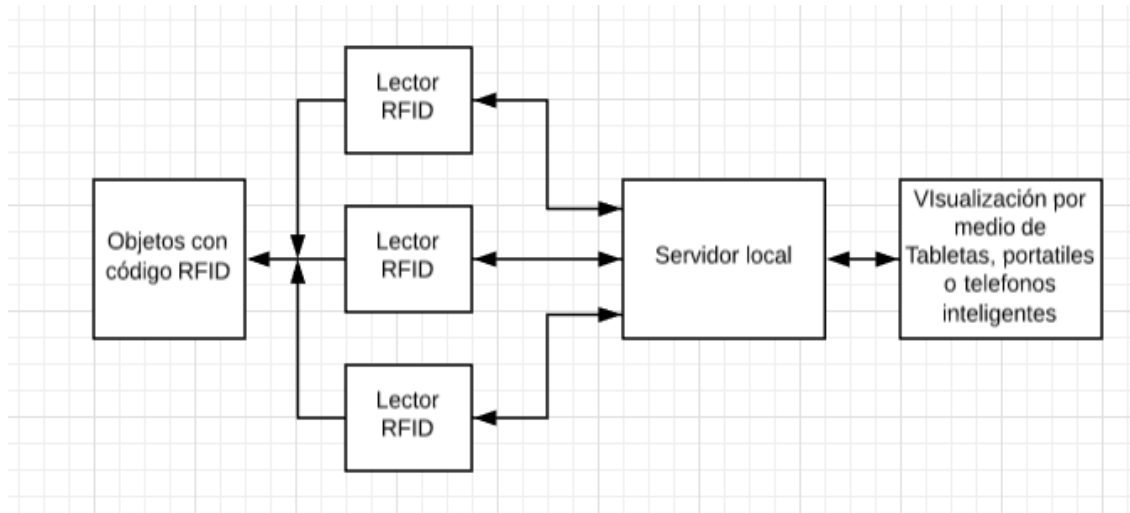


Ilustración 2. Red de tecnología RFID.

Tabla 3. Ventajas, sensores, dispositivos y software empleados en la tecnología RFID. [18][19][20].

Tecnología RFID	
Ventajas	Sensores, dispositivos y software utilizados
<ol style="list-style-type: none"> 1. Registro completo del estado e información del objeto. 2. Permite actualizar información detallada sobre cada uno de los objetos que contiene los Tags 3. Permite identificar objetos de una manera fácil (buscar, rastrear y seguir). 4. No necesita de fuente de alimentación para las etiquetas RFID. 	<ul style="list-style-type: none"> ➤ Etiquetas RFID o Tags RFID. ➤ PDA (Asistentes personal digital) o agenda de bolsillo o PC. ➤ Visual Basic es un lenguaje de programación orientado a objetos. ➤ ASP.Net sitio web de control. ➤ My SQL para emplear base de datos.

1.4.2. Tecnología Video Cámaras:

Una video cámara (transductor óptico) es un dispositivo que permite registrar imágenes y sonidos convirtiéndolos en señales eléctricas que pueden ser reproducidas en un aparato determinado. En su proceso interno las cámaras realizan la descomposición de la luz en tres componentes (rojo, verde y azul) mediante un prisma de espejos dicroicos y del otro lado del prisma se encuentran los captadores, que reconstruyen la imagen y la reenvían a los circuitos preamplificadores.

La imagen pasa por un proceso de corrección de parámetros y finalmente es grabada o emitida hacia la red que conforma el entorno de trabajo. Al implementar esta tecnología en un sistema de aparcamiento se emplea el proceso descrito anteriormente y se almacena la imagen del parqueadero [21], [22].

Cuando se digitaliza la señal, ésta es enviada a un dispositivo para realizar el procesamiento mediante un software a través de algoritmos de detección de puntos o bordes y determinar algunos objetos o parámetros propios contenidos en la imagen. Los parámetros encontrados son enviados a una base de datos para ser visualizados por medio de un dispositivo móvil. Este proceso se muestra en la ilustración 3.

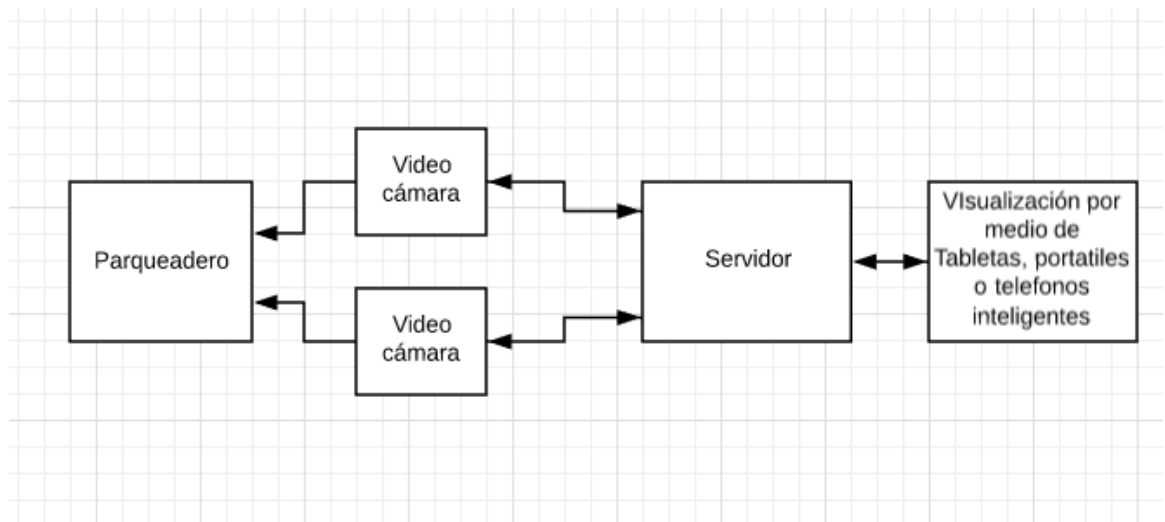


Ilustración 3. Red de tecnología Video Cámara.

Tabla 4. Ventajas, sensores, dispositivos y funcionamiento de las videocámaras. [21], [22].

Tecnología (cámara digital)	
Ventajas	Sensores y dispositivos utilizados
Obtener gran abarque de la zona en cuestión.	✓ Cámaras de video con dispositivo AV (NTSC O PAL) para el procesamiento de imágenes de video.
tiene un gran potencial para sistemas de cómputo en un futuro ser incorporado a la vida cotidiana	✓ FPGA
	✓ Antena para el envío de la imagen
	✓ Fuente de alimentación.
	✓ Software de procesamiento de imagen.
	✓ Convertidor analógico digital.

1.4.3. Tecnología Vanet:

Definición: “Las redes vehiculares se consideran una especialización de las Redes Móviles Ad hoc (MANET, Mobile Ad hoc Networks) y su principal diferencia es que los nodos están restringidos para moverse únicamente a lo largo y ancho de las vías. En una VANET, cada vehículo se define como un nodo de la red y está equipado con una unidad de comunicación a bordo denominada OBU (On-Board Unit) y una unidad de aplicación llamada AU (Application Unit). (ver ilustración 4).

La función de la OBU es intercambiar información con otros vehículos o con puntos de acceso estacionarios ubicados alrededor de las carreteras y permiten el acceso a internet, denominados RSU (Road-Side Unit); mientras que las AU hacen referencia a los dispositivos que muestran información al usuario.” [23]

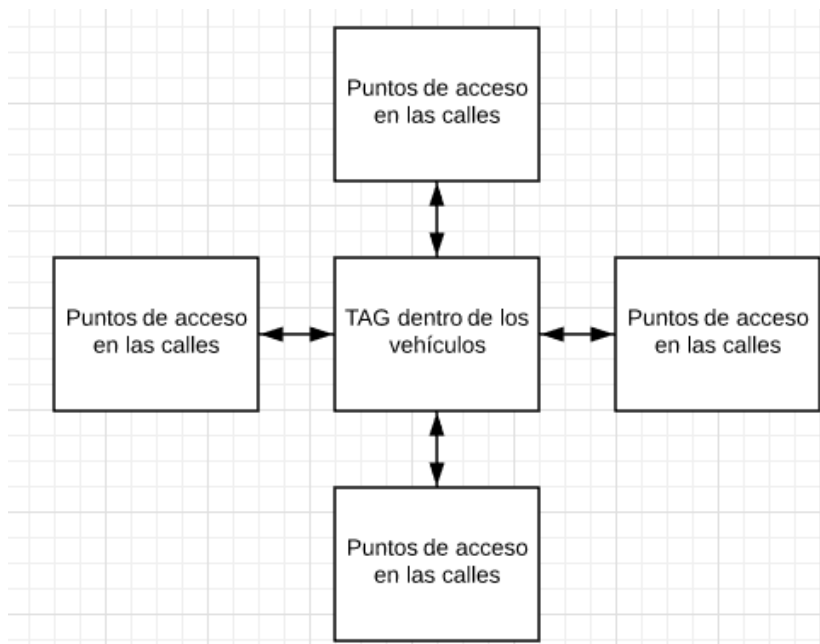


Ilustración 4. Red de tecnología Vanet.

Tabla 5. Ventajas, sensores, dispositivos y funcionamiento de la tecnología Vanet. [24], [25], [26]

Tecnología (VANET)	
Ventajas	Sensores y dispositivos utilizados
<ul style="list-style-type: none"> ✓ Controla la seguridad y la gestión de tráfico. ✓ Los nodos se mueven a velocidades altas y se protegen de ataques y amenazas. ✓ Presenta cambios rápidos en la topología de la red. ✓ No hay restricciones de energía ya que los vehículos son capaces de proporcionar la energía. ✓ Tamaño de la red ilimitado. ✓ Comunicación inalámbrica. ✓ Soporta el estacionamiento en tiempo real. ✓ Son vigilados por las RSU (postes). 	<ul style="list-style-type: none"> ✓ Canales de comunicación radiación electromagnética. ✓ Los TAGs están situados dentro de los vehículos.

1.4.4. Tecnología GPS:

La tecnología GPS (Global Positioning System) fue desarrollada por el departamento de defensa de EEUU como un recurso global para navegación y posicionamiento de uso militar y civil. El sistema contiene 24 satélites en órbita a una distancia de más de veinte mil kilómetros. Estos satélites funcionan como puntos de referencia, con los cuales un receptor en tierra puede "triangular" su propia posición, ya que sus órbitas son monitoreadas con gran precisión desde estaciones en tierra.

Esta tecnología en un sistema de aparcamiento (*ver ilustración 5*), debe incluir un GPS en el interior del vehículo y tener las coordenadas espaciales de cada una de las plazas del parqueadero almacenadas en una base de datos. El proceso de detección de plazas libres se logra al comparar la distancia geoespacial de cada vehículo con respecto a las coordenadas de las plazas almacenadas. Cuando esta tiende a cero garantiza que la plaza dentro del parqueadero se encuentra ocupada, el usuario puede verificar la gestión desde un dispositivo móvil.

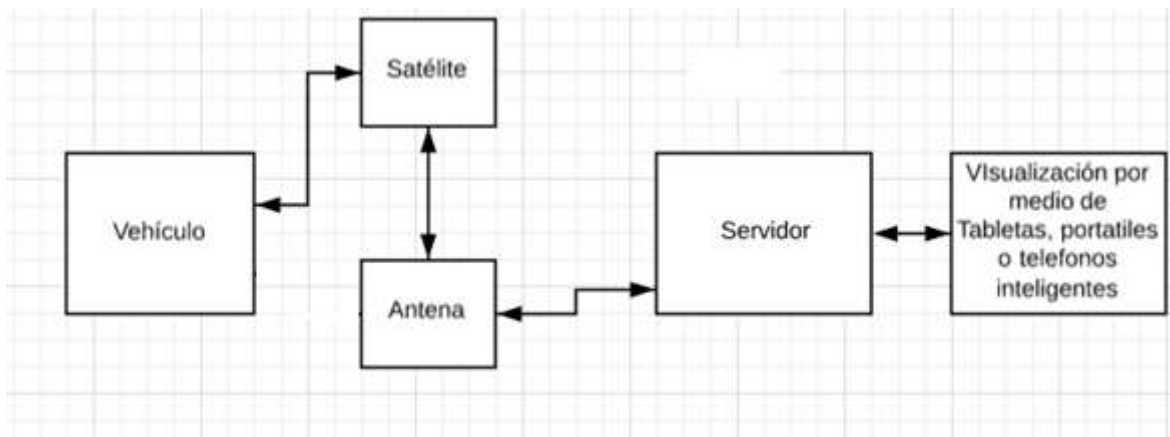


Ilustración 5. Red de tecnología GPS

Tabla 6. Ventajas, sensores, dispositivos y funcionamiento de la tecnología GPS. [27].

Tecnología (GPS)	
Ventajas	Sensores, dispositivos y software utilizados
<ul style="list-style-type: none"> ✓ No necesita infra estructura. ✓ Es fácil de usar. ✓ Visualización en tiempo real. ✓ Se resalta la ubicación en el GPS del usuario lo que ayuda con la búsqueda de coordenadas. ✓ Ayuda a encontrar el parqueadero más cercano. 	<ul style="list-style-type: none"> ✓ Utiliza las aplicaciones de google. ✓ GPS ✓ Tablet, móviles.

1.4.5. Tecnología WSN:

Definición: “Una red de sensores inalámbricos (WSN) es una red inalámbrica que consiste en dispositivos distribuidos, espaciados y autónomos que utiliza sensores para monitorear las condiciones físicas o ambientales. Un sistema WSN incorpora un gateway que provee conectividad inalámbrica de regreso al mundo de cables y nodos distribuidos (ver ilustración 6).”[28]

En un sistema de aparcamiento, esta tecnología permite que los nodos de sensores puedan recolectar, procesar y transmitir una información a través de Wi-fi al Gateway. El Gateway se encuentra conectado a un servidor de datos donde se almacena la información en tiempo real, posterior a ello el usuario puede realizar un análisis y visualizar los datos a través de un dispositivo móvil.

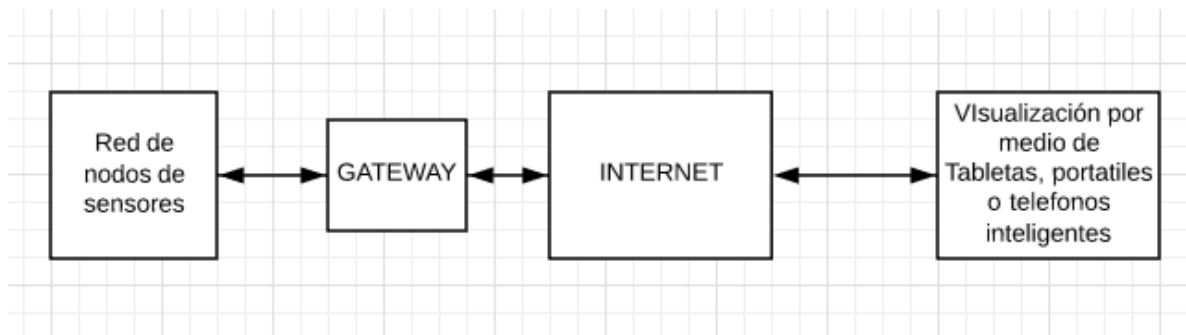


Ilustración 6. Red de tecnología WSN.

Tabla 7. Ventaja, sensores, dispositivos y funcionamiento de la tecnología WSN. [29] [30].

Tecnología WSN	
Ventajas	Sensores, dispositivos y software utilizados
<ol style="list-style-type: none"> 1) Se puede conectar con diversos nodos de sensores midiendo los mismos parámetros en un área de interés. 2) Envío periódico de la información recogida. 3) Son de bajo costo. 4) Con esta arquitectura se puede desacoplar la capa superior de aplicación, debido a que cualquier modificación que se presente en la red de sensores inalámbricos no afecta la capa superior de aplicación. 5) La capa de aplicación solo se encarga de administrar la información almacenada en la base de datos. 	<ul style="list-style-type: none"> ➤ Procesador y un chip de radio o procesador de placas de radio (MPR 2400). Las tarjetas MPR o Xbee funcionan de manera compatible con IEEE802.15.4 y se puede conectar a tableros de diversos sensores. ➤ Sistema operativo TinyOS proporciona eventos de bajo nivel para gestión de tareas. ➤ Xmesh crossbow pila de red ➤ Tarjetas de adquisición de datos MTS310, RN131 Se puede equipar con sensores de luz, temperatura y acústica. ➤ la tarjeta MIB510, Xbee pro. Proporciona una puerta de enlace que permite la adquisición de datos sensoriales sobre un PC y permite un control del sistema de aplicación de capa superior. ➤ El protocolo de red utilizado en el sistema es IEEE 802.15.4. ➤ Redes wi-fi. IEEE 802.11 ➤ Intel Enhanced 8051 núcleo.

1.5 Clasificación del sistema a implementar.

1.5.1 Clasificación de estrategias y contextualización hacia la Smart City.

Se realiza un cuadro comparativo empleando las tablas descritas en cada tecnología, y se aplica a un parqueadero de 16 plazas. Donde se analiza la conectividad, infraestructura de los datos, seguridad y cantidad de sensores (ver tabla 8)

Tabla 8. Contextualización de tecnologías hacia Smart City, para un parqueadero de 16 plazas.

Análisis para un parqueadero de 16 plazas					
	Tecnologías				
	RFID	Video Cámaras	Vanet	GPS	WSN
Conectividad	Transmite mediante ondas de radio frecuencia (no maneja conexión IP)	Presenta conexión IP y permite la conexión a otras redes. También presenta una conexión inalámbrica.	Presenta conexión TCP/IP	El sistema debe tener conexión IP para el envío y recepción de información.	Permite la conexión inalámbrica
Infraestructura de Datos	Maneja estándares, políticas, tecnología, servidores y aplicaciones, pero todo es de manera local.	Maneja estándares, políticas, tecnología, servidores web y aplicaciones web.	Maneja una unidad de aplicación denominada AU. La OBU permite el intercambio de información	(No necesita infraestructura para encontrar la ubicación). Pero puede manejar servidores para guardar datos	Maneja estándares, políticas, tecnología, servidores web y aplicaciones web.
Seguridad	Es más segura, porque no hay conexión externa, es menos propensa a ataque maliciosos.	Depende de las políticas del servidor donde alojemos nuestra base de datos.	La comunicación es de vehículo a vehículo o de vehículo a infraestructura de comunicación, esto requiere un número elevado de transmisiones lo cual garantiza un porcentaje alto de pérdidas de paquetes	Depende de las políticas del servidor donde alojemos nuestra base de datos.	Depende de las políticas del servidor donde alojemos nuestra base de datos.
Cantidad de sensores	Una etiqueta y un lector para cada vehículo	Un sensor por cada 8 plazas	Cada vehículo debe tener una unidad de comunicación	Cada vehículo debe contener un GPS	Cada plaza debe tener un sensor inalámbrico.

Al realizar el análisis de las implementaciones descritas anteriormente, se establece que la tecnología WSN y la tecnología de video cámara son las que más se asemejan a la definición de Smart City, las cuales nos dan solución a la gestión del sistema de aparcamiento; ya que presentan conectividad, infraestructura de datos, seguridad y cumplen con las recomendaciones mencionadas en la sección 1.3.4 de este mismo capítulo.

En la tecnología de video cámara se emplean menos sensores, ya que esta abarca varias plazas con un solo sensor para ofrecer el servicio de gestión, esto hace que el sistema sea menos costoso en cuanto a su implementación

En este sistema de video cámara se tiene como variable o señal de medición las coordenadas del parqueadero vacío a partir de imágenes del mismo, luego se utiliza el procesamiento de imágenes y el resultado de dicho procesamiento se envía, haciendo uso del protocolo IPV6, por medio de un módulo Wifi a un servidor web, seguido se enlaza el servidor web con una aplicación Android, garantizando así el servicio de gestión de aparcamiento.

CAPITULO 2.

2.1. Requerimientos del sistema.

2.1.1. Requerimientos funcionales.

En un sistema de aparcamiento que implementa la tecnología de video cámara se debe tener en cuenta algunas condiciones básicas para su desarrollo, estas son:

- **Iluminación:** los sistemas de iluminación varían en cuanto a si son parqueaderos de garaje (espacios cerrado) o los parkings (espacios abiertos), aunque ambos tienen el mismo propósito, se debe procurar que no queden zonas oscuras o sin iluminar, esto tiene que ver con la seguridad de los usuarios y la captura clara de las imágenes por medio de las cámaras.

En los parqueaderos cerrados, las luces permanecen encendidas las 24 horas del día, por lo tanto se requiere que las lámparas tengan una alta durabilidad y ofrezcan una luz lo generalmente potente, por lo cual se recomienda utilizar un sistema de iluminación LED.

En los parqueadero abiertos se emplean elementos de iluminación exterior como proyectores led situados a una altura elevada para facilitar una distribución uniforme de la luz y debe contar con una seguridad que garantice la resistencia a la intemperie, esta iluminación solo se utiliza en horas nocturnas o de baja luminosidad.

- **Suelo del parqueadero:** en un parqueadero es importante identificar el material con el cual está construido el suelo, para garantizar que la iluminación no se refleje de manera directa y afecte cada una de las imágenes obtenidas por las cámaras.
- **Ubicación de las cámaras:** se establece que cada cámara capture 8 plazas del parqueadero, por lo tanto es importante ubicarlas en una parte alta que proporcione la seguridad y el cuidado de las mismas con relación a la intemperie, esta seguridad depende si el parqueadero es un lugar abierto o cerrado.
- **Señalización en el interior:** se debe realizar la señalización de direccionamiento o ruta a seguir por cada vehículo, para disminuir la congestión en el interior. La señalización deben ser de color amarillo o blanco para generar un contraste con el color oscuro del suelo y lograr identificar cada una de las plazas.
- **Procesamiento de la señal:** para lograr un buen procesamiento se requiere de un procesador de 4 núcleos a 1,2 GHz con un soporte de 64-bit, un mínimo de 1GB de RAM, capacidad de almacenamiento mayor a 16 Gb.

- **Módulo Wifi:** se debe contar con un módulo Wifi para el envío de información a la base de datos pública.
- **Base de datos:** se debe utilizar una base de datos que incluyan programas no visibles que administren y gestionen la información contenida, y así garantizar la seguridad, integridad y consistencia de la información.
- **Aplicación Android:** Debe ser amigable con el usuario, es decir de fácil uso, garantizar la conexión con la base de datos para el ingreso de información y contener botones que permitan la consulta del estado del parqueadero en tiempo real.

La primera plantilla contiene la presentación del parqueadero, en la segunda plantilla están los campos para el registro de información del usuario como nombre, cedula y placa y en la tercer plantilla es donde el usuario realiza la consulta del estado del parqueadero a partir de la identificación del ultimo id de la tabla en la base de datos.

2.1.2. Requerimientos no funcionales.

En este sistema los requerimientos no funcionales se categorizan en 2 campos (requerimientos del producto y requerimientos organizacionales)

- **Requerimientos del producto:** El sistema en funcionamiento debe ser capaz de actualizar la base de datos en un tiempo menor a 3 segundos y realizar la consulta desde la aplicación Android hasta la base de datos en un tiempo máximo de 1 segundo, lo anterior se cumple cuando hay una conexión Wifi.

El sistema debe ser capaz de operar adecuadamente hasta con 100 usuarios en conexiones simultáneas a la base de datos, La cantidad de disco duro de alojamiento de datos debe ser mayor o igual a 100GB.

Los permisos de acceso al sistema solo podrá ser cambiados por el administrador de la base de datos, además el sistema no debe ser vulnerable a los virus y al malware

La tasa de errores del sistema debe ser menor al 10% en condiciones de iluminación adecuado y el tiempo de disponibilidad del sistema debe ser del 99% de las veces que el usuario intente acceder a la consulta de gestión de aparcamiento.

El tiempo para iniciar el sistema o reiniciar el sistema no podrá ser mayor a 5 minutos.

- **Requerimientos organizacionales:** la aplicación Android será desarrollada bajo el lenguaje de programación java, C, C++ y para su verificación será efectuada al cargar la app en un dispositivo móvil o un simulador de Android.

El procesamiento de la señal será realizado en un lenguaje de programación de código abierto y orientado a objetos que facilite el uso de librerías, para la solución de diversos problemas.

En la base de datos se debe crear una tabla para el registro de nombre, cedula y número de placa y otra para el registro del estado de las plazas del parqueadero.

La resolución de la cámara debe ser de 640x480 pixeles o mayor para lograr una buena claridad de la imagen. La conexión debe ser por puerto USB y tener compatibilidad con versiones del sistema operativo GNU/Linux y versiones avanzadas de Windows xp.

El código para la consulta y escritura en la base de datos será escrito en PHP, ya que es el adecuado para el desarrollo web.

2.2. Diseño del sistema

2.2.1. Diseño del prototipo de parqueadero

Se plantea construir un prototipo de parqueadero con 16 bahías, el cual se subdivide en dos módulos y cada módulo tiene 8 bahías, tal como se muestra en la ilustración 7.

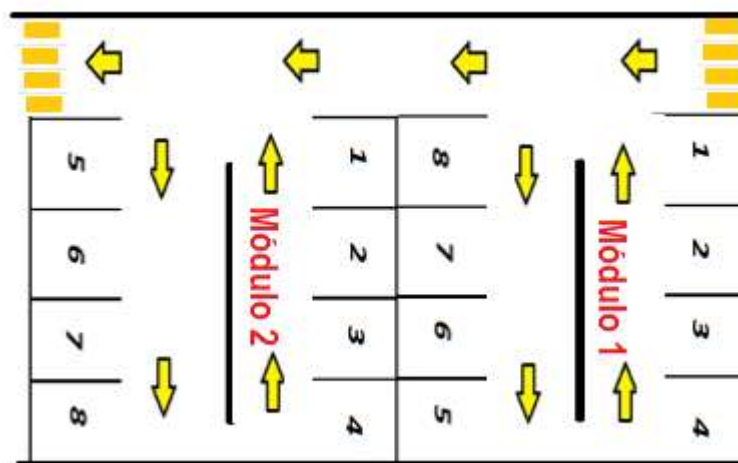


Ilustración 7. Idea de prototipo de parqueadero

2.2.2. WebCam:

Para tomar datos del parqueadero, se utilizan dos web cam, lo cual permite que sea de bajo costo para su implementación, garantizando, así, los requerimientos establecidos. Cada una de estas webcam cuenta con *una resolución máxima de 640x480(VGA), una interfaz USB2.0 compatible con USB1.1 un sensor de tipo COMS con una velocidad de cuadros máxima de 30 hz* [29], [30].



Ilustración 9 WebCam GENIUS FASECAM 310, usada para toma de imágenes en el prototipo de parqueadero. [31]



Ilustración 8 WebCam MICROSOFT MODELO 1407, usada para toma de imágenes en el prototipo de parqueadero. [32]

Se recomienda utilizar dos web-cam que sea del mismo modelo y marca, para que las imágenes almacenadas tengan los mismos parámetros de contraste, iluminación, etc. Para la implementación de este sistema solo se cuenta con estas dos web-cam, aunque cada una de ellas cumple con los requerimientos establecidos, cabe resaltar que las imágenes tendrán distintos parámetros de contraste, iluminación, etc.

2.2.3. Raspberry Pi3 Model B.

El sistema embebido que se emplea para el procesamiento de imagen es una tarjeta Raspberry Pi 3 Model B (ver ilustración 10) y cuenta con unas especificaciones que pueden observarse en la página oficial www.raspberrypi.org :

“Un procesador a 1,2 GHz de 64 bits con cuatro núcleos ARMv8, 802.11n Wireless LAN, Bluetooth 4.1, Bluetooth Low Energy (BLE), 4 puertos USB, 40 pines GPIO, puerto Full HDMI, Puerto Ethernet, conector combo compuesto de audio y vídeo de 3,5 mm, interfaz de la cámara (CSI), interfaz de pantalla (DSI), ranura para tarjetas microSD

(ahora push-pull en lugar de push-push), núcleo de gráficos VideoCore IV 3D, sistema operativo GNU/Linux ARM(Debian, Fedora, Arch Linux), RISC OS, dimensiones de placa de 8.5 por 5.3 cm.”



Ilustración 10 Tarjeta Raspberry Pi 3 Model b para realizar el procesamiento de imágenes.

2.2.4. Alojamiento(Hosting).

Se realiza una breve descripción de las ventajas que presenta un *hosting* público y uno privado, para escoger el que más se ajuste a la definición instaurada para las Smart City.

Hosting privado:

Un alojamiento web privado no tiene límites de espacio de almacenamiento, no hay limitaciones en cuanto a publicación de cualquier contenido, porque todo se hace de forma local.

Todo el contenido se guardaría de forma local y estaría disponible para que los usuarios conectados a la misma red puedan tener acceso a la información, lo anterior implica gasto de ancho de banda de subida para poder mostrar la página web.

Se debe tener encendido 24 horas el ordenador y protegido contra malware y otros ataques.

- **Componentes hardware:** Como mínimo, se necesita, un router con conexión a Internet, un ordenador que haga las veces de servidor web y cableado de red RJ45 para conectar el router con el ordenador.

- **Componentes software:** Sistema Operativo Windows o GNU/Linux, programa servidor como podría ser Apache Server, un programa para la creación y edición de páginas Web.

Hosting público:

- Tiene hardware avanzado, lo que permite altas velocidades y buena capacidad de almacenamiento, lo anterior depende del tipo de contrato que se haya establecido con la empresa prestadora de servicio.

Teniendo en cuenta las descripciones de los servicios, se dispone de un servidor privado para la realización de pruebas y toma de datos iniciales y así proceder a alquilar un dominio público que cumpla con los requerimientos y las recomendaciones de la Unión Internacional de las Telecomunicaciones (ITU), presentadas en el capítulo 1.

Para almacenar los datos en la nube se accede a un *hosting público* que ofrece la empresa GoDaddy, esta empresa permite 200 conexiones simultaneas a la base de datos, brinda 100GB de almacenamiento, no limita el ancho de banda, un dominio publico, seguridad contra malwares y otros virus de la web, establece una conexión segura por medio del certificado SSL y soporta PHP 7.0, 7.1, lo que garantiza el cumplimiento de los requerimientos.

2.2.5. Tablas de registro en la base de datos

En la tabla Usuarios se crea una fila ID auto incrementable de tipo entero, la segunda fila “nombres” es varchar o caracter, la tercera, denominada “cedula”, es tipo entero, y la última fila “placa” es de tipo varchar. El usuario puede consultar sus datos sólo colocando su nombre en la aplicación Android sólo si con anterioridad se ha registrado, (ver ilustración 11)

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(50)			No	Ninguna	AUTO_INCREMENT
2	nombres	varchar(30)	utf8_spanish2_ci		No	Ninguna	
3	cedula	int(12)			No	Ninguna	
4	placa	varchar(10)	utf8_spanish2_ci		No	Ninguna	

Ilustración 11 Característica de la tabla “Usuarios”

En la tabla “estadoparqueadero” se crea una fila ID auto incrementable, estableciendo que, a medida que se registren los resultados provenientes de procesamiento, el ID aumente de uno en uno, también se debe incluir 16 filas denominadas zona1, zona2, zona3, zona4, zona5, zona6, zona7, zona8, zona9, zona10, zona11, zona12, zona13, zona14, zona15, zona16, y cada una de tipo varchar, (ver ilustración 12).

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
☐ 1	id	int(100)			No	Ninguna	AUTO_INCREMENT
☐ 2	zona1	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 3	zona2	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 4	zona3	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 5	zona4	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 6	zona5	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 7	zona6	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 8	zona7	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 9	zona8	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 10	zona9	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 11	zona10	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 12	zona11	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 13	zona12	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 14	zona13	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 15	zona14	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 16	zona15	varchar(10)	utf8_spanish2_ci		Sí	NULL	
☐ 17	zona16	varchar(10)	utf8_spanish2_ci		Sí	NULL	

Ilustración 12 Característica de la tabla "Estadoparqueadero"

2.2.6. Aplicación Android

Para la creación de la aplicación Android se utiliza un software libre denominado Android Studio, el cual contiene librerías para la comunicación con otros sitios web, entorno gráfico y emulación rápida en cualquier dispositivo android.[33]

Dentro de la aplicación Android se plantea diseñar tres plantillas, obedeciendo a lo estipulado en los requerimientos, de manera que sea de fácil manejo para el usuario, se establece que la primer plantilla debe incluir una imagen del parqueadero prototipo para familiarizar al usuario con el entorno, además se debe incluir un botón que permita la transición de la primer plantilla a la segunda, el diagrama de flujo para la primer plantilla se muestra en la ilustración 13.

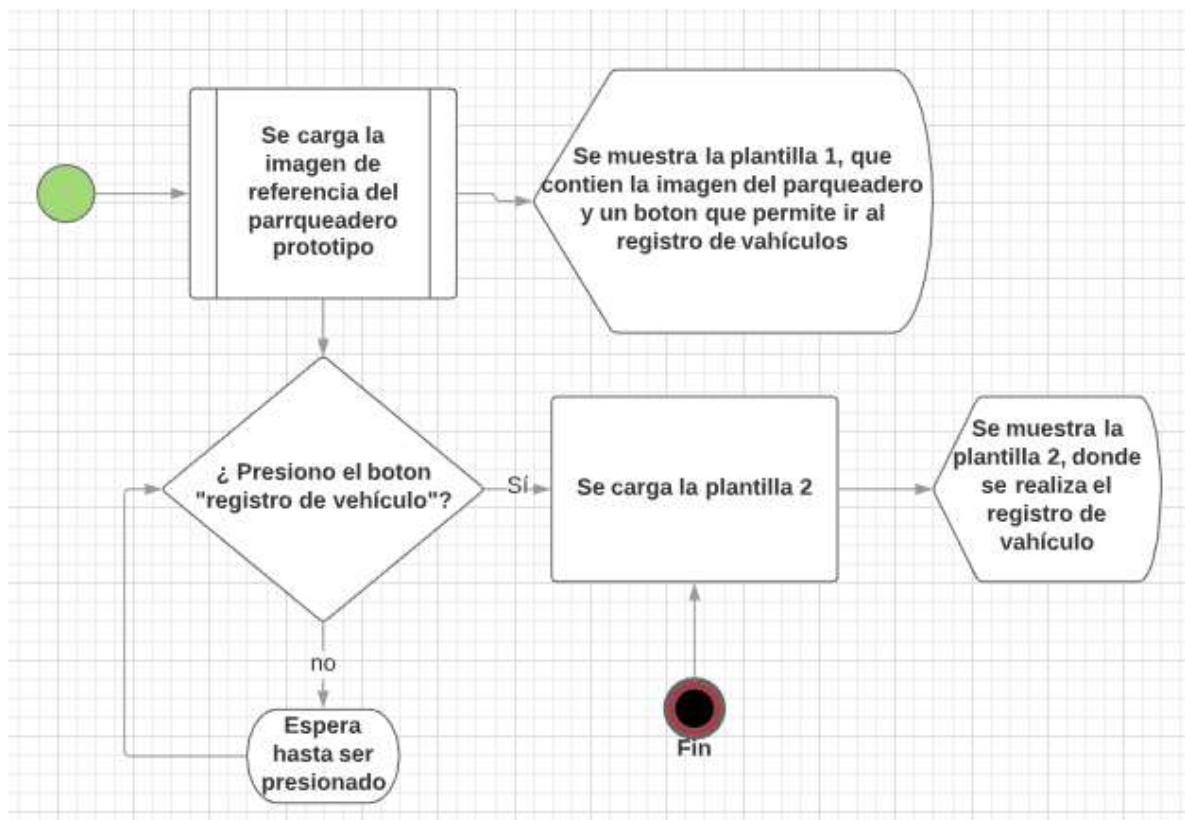


Ilustración 13 Diagrama de flujo que debe seguir la plantilla 1 de la aplicación Android

En el diseño de la plantilla 2, se establece que debe tener líneas de edición de texto para que el usuario pueda ingresar el nombre, la cedula y el número de placa del vehículo, esto es necesario para tener un control de los usuarios que ingresan al parqueadero, además se debe incluir un botón que permita enviar la información suministrada a la base de datos y alojarla en la tabla usuarios descrita anteriormente, un botón que permita la consulta a la base de datos, teniendo como referencia el nombre de usuario, es decir, el usuario desea consultar sus datos, por ende el ingresa su nombre en la línea de edición de texto nombre y presiona el botón consultar, internamente se busca nombre del usuario en la base de datos y se carga toda la información relacionada con dicho nombre.

Cuando se presione el botón guardar se debe mostrar un texto informativo de los datos que se guardaron correctamente. Se debe incluir otro botón que permita la transición entre la plantilla 2 y la plantilla 3, este proceso se muestra en la ilustración 14

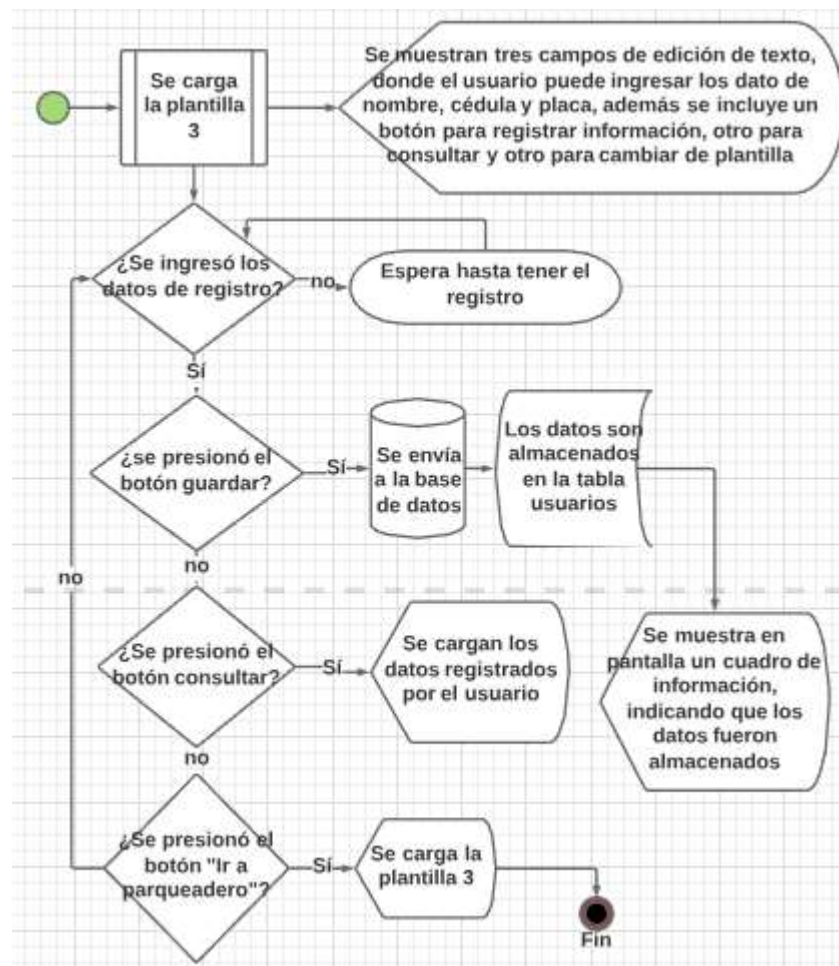


Ilustración 14 Diagrama de flujo que debe seguir la plantilla 2 de la aplicación Android

En el diseño de la plantilla 3, se establece que para la visualización del estado del parqueadero se utiliza líneas de edición de texto donde solo se muestren la información al presionar el botón consultar, se debe incluir un botón que permita consultar el último ID de la tabla estadoparqueadero mostrada en la ilustración 12, otro botón que permita la consulta de la plazas libres del parqueadero y un último botón que permita la transición con la plantilla 2. La secuencia para realizar la consulta del estado es el siguiente:

- se presiona el botón #_ID, que permite la extracción del ultimo ID de la tabla.
- se presiona el botón consultar, que permite la consulta por medio del ID encontrado en el paso anterior, los datos almacenados se cargan en las líneas de edición de texto.

Cabe resaltar que en las líneas de edición de texto solo aparecerá un cero o un uno, donde cero indica que la plaza se encuentra libre y uno que la plaza se encuentra ocupada. El proceso mencionado se muestra en la ilustración 15.

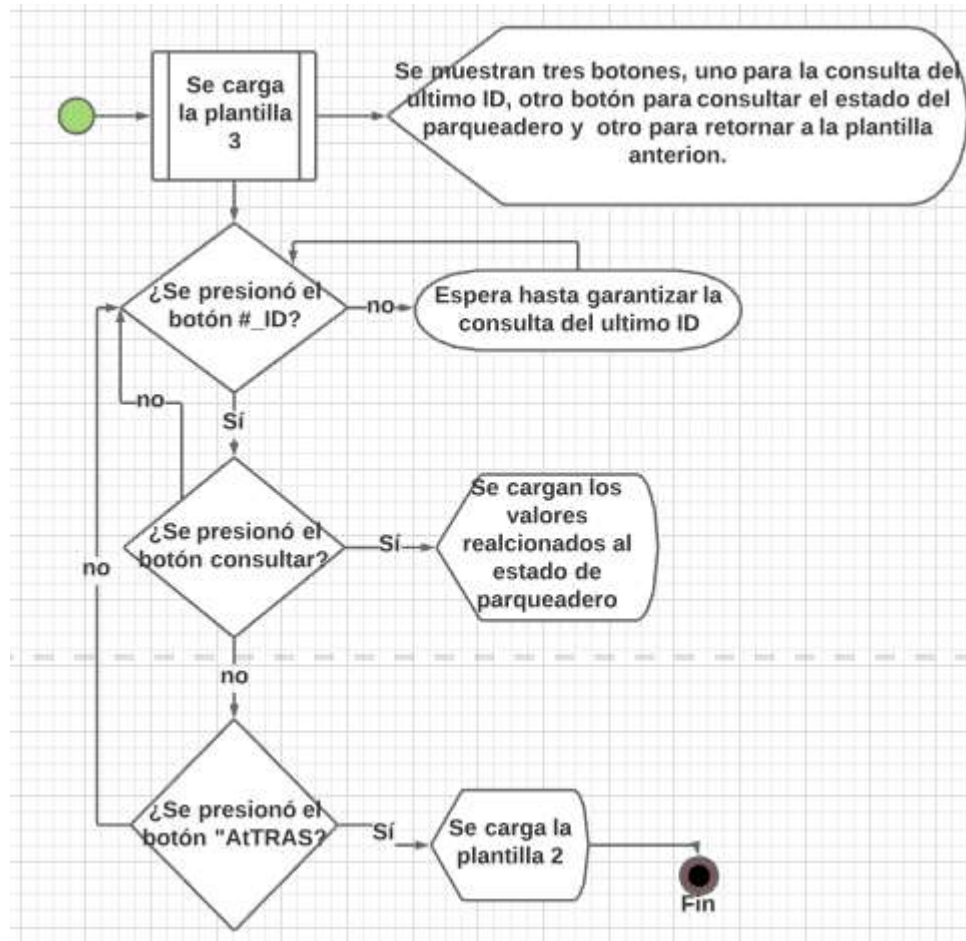


Ilustración 15 Diagrama de flujo que debe seguir la plantilla 3 de la aplicación Android

2.2.7. Diseño metodológico del sistema

El sistema de aparcamiento cuenta con 16 plazas, en las 16 plazas se realiza el procesamiento. Para ello se fijan las dos webcam y cada una tendrá la función de sensar y tomar control de 8 plazas del prototipo. La imagen tomada se almacena en la tarjeta Raspberry Pi 3, posterior a ello se procesa a través de Python y la herramienta Opencv. Al finalizar el procesamiento, los datos obtenidos se envían al hosting público adquirido

y, por medio de la aplicación Android, se visualiza si hay o no plazas libres. Teniendo como esquema general para la solución a lo planteado se muestra la ilustración 16. Este diseño se plantea considerando cada uno de los requerimientos citados en la sección 2.1 del capítulo 2.

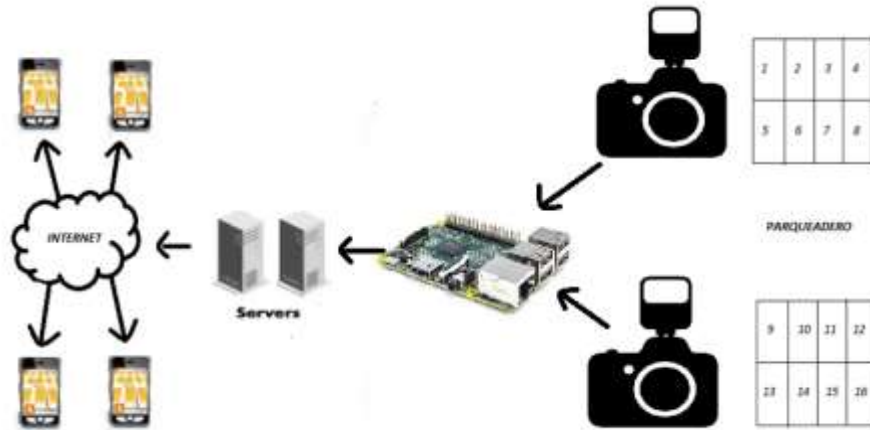


Ilustración 16 Diseño de estructura para gestión de aparcamiento.

2.3. Implementación del sistema

2.3.1. Parqueadero prototipo físico

El parqueadero cuenta con una entrada y una salida, el sistema contiene dos webcam denominadas webcam1, establecida como el sensor del módulo 1, y la Webcam2, establecida como el sensor del módulo 2. Dentro del parqueadero hay señalización con el fin de que los vehículos no se desplacen en contravía. Se diseña una torre donde se fijan los sensores para obtener la señal y ejecutar el procesamiento de la imagen. (Ver ilustraciones 17 y 18)



Ilustración 17 Diseño de prototipo de parqueadero completo



Ilustración 18 Diseño de prototipo de parqueadero por módulos

2.3.2. Instalación de las Webcam en Rasbian

En la tarjeta Raspberry se carga el sistema operativo Rasbian, que tiene instalado por defecto Python 2.7, pero antes se debe reconocer los puertos y dispositivos de video asociados a cada web-cam. Por lo tanto se conecta cada una de las webcam USB a la Raspberry pi3 y, se ejecuta desde la consola los comandos de instalación de la webcam como sigue:

```
sudo apt-get update
sudo apt-get install libjpeg8-dev imagemagick subversion
cd /usr/src/
```

```
svn co https://svn.code.sf.net/p/mjpg-streamer/code/ mjpg-streamer
cd mjpg-streamer
make
```

Al verificar si ya se instaló la webcam, por medio de consola se obtiene lo siguiente:

```
MJPEG Streamer Version: svn rev: 3:172
i: Using V4L2 device.: /dev/video0
i: Desired Resolution: 640 x 480
i: Frames Per Second.: 5
i: Format.....: YUV
i: JPEG Quality.....: 80
o: www-folder-path...: ./ home/pi/Desktop/base/
o: HTTP TCP port.....: 8080
o: username:password.: disabled
o: commands.....: disabled
```

Donde se garantiza que las imágenes tomadas se guardan en la carpeta /home/pi/Desktop/base/. Lo anterior se realiza para cada una de las Webcam.

2.3.3. Procesamiento de la imagen

Para realizar el procesamiento de la variable, se establece el Diagrama de bloques de la ilustración 19, que será ejecutado en Python 2.7

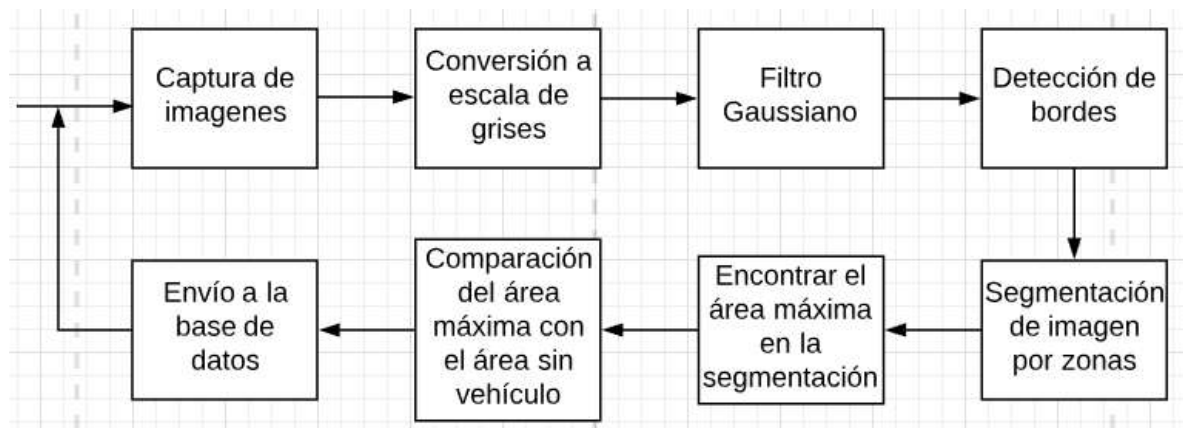


Ilustración 19 Diagrama de procesos para adquisición y manipulación de la imagen.

Se cargan las librerías para capturar las imágenes con Opencv, Numpy y lograr la conexión con la base de datos mysql.

```
# -*- coding: utf-8 -*-
#include <iostream>
#include <opencv2/opencv.hpp>

import mysql.connector
import numpy as np
import cv2
import cv2 as cv
import time
```

Con el comando `cv2.Capture ( )` se selecciona el dispositivo de cámara que se va a utilizar. Para la activación de las webcam y la captura de la imagen se utiliza el comando `video1.read( )` y luego se guardan en un lugar de la tarjeta SD de la Raspberry pi3. Cuando se almacena la imagen, se desactiva el dispositivo para reducir el consumo de corriente.

```
video1 = cv2.VideoCapture(0)
video2 = cv2.VideoCapture(1)
check, frame = video1.read()
check1, frame1 = video2.read()
cv2.imwrite('/home/pi/Desktop/base/camara1.jpg',frame)
cv2.imwrite('/home/pi/Desktop/base/camara2.jpg',frame1)
video1.release()
video2.release()
```

Posteriormente se carga la imagen almacenada con el comando `cv2.imread( )` y se procede a convertir a escalas de grises con `cv2.cvtColor(imagen, tipo de conversión)`, este método se aplica para las imágenes capturadas por las dos cámaras (ver *ilustración 20*). Las imágenes son diferentes debido a que los parámetros de fábrica de contraste y brillo son distintos.

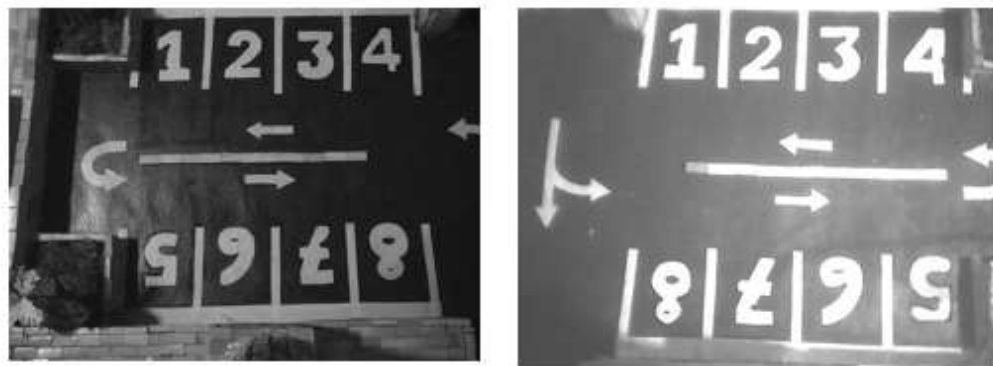


Ilustración 20 Imagen del parqueadero a escala de grises.

Con el comando `cv2.GaussianBlur` se aplica un filtro que se utiliza para reducir ruido y suavizar la imagen en los cambios de umbrales, tal y como se muestra en la ilustración 21.



Ilustración 21.. Imagen con el Filtro Gaussiano.

Luego se detectan los bordes utilizando el comando `cv2.Canny` (ver ilustración 22), donde se debe especificar el umbral que se desea visualizar en la imagen; en este caso, se encontró el umbral de manera empírica, para poder identificar aquellos pixeles que pertenecen al borde, se escoge un umbral mínimo de 100 y máximo de 230 porque se desea que el contorno sea completamente cerrado, es decir que no haya discontinuidades, teniendo en cuenta que entre mayor sea el rango hay menores discontinuidades y esto implica que el borde presente mayor grosor.

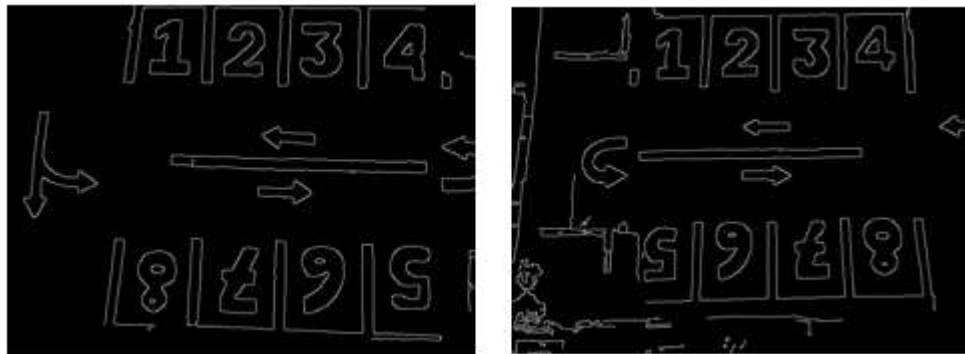


Ilustración 22.Imagen con detección de bordes

A continuación se presenta el código con los comandos mencionados:

```
# Cargamos la imagen
vacía1 = cv2.imread('/home/pi/Desktop/base/camara1.jpg')
vacía2 = cv2.imread('/home/pi/Desktop/base/camara2.jpg')
# Convertimos a escala de grises
```

```

grisvacia1 = cv2.cvtColor(vacia1, cv2.COLOR_BGR2GRAY)
grisvacia2 = cv2.cvtColor(vacia2, cv2.COLOR_BGR2GRAY)

# Aplicar suavizado Gaussiano
gauss1 = cv2.GaussianBlur(grisvacia1, (5,5), 0)
gauss2 = cv2.GaussianBlur(grisvacia2, (5,5), 0)

# Detectamos los bordes con Canny
canny1 = cv2.Canny(gauss1, 100, 230)
canny2 = cv2.Canny(gauss2, 100, 230)

```

Posteriormente se procede a realizar la segmentación en cada una de las imágenes (ver ilustración 24) teniendo en cuenta las plazas en el parqueadero y luego se guardan y se cuentan aquellos bordes que cumplen con el umbral establecido para cada imagen segmentada. Este procedimiento se realiza para cada una de las plazas o bahías en cada uno de los módulos. A continuación se muestra el código que realiza este proceso descrito y es replicable para cada una de las plazas

```

#####plaza1#####
plaza1=canny1[2:134, 187:274]
for j in range(86):
    plaza1[0][j]=255
cv2.imwrite('/home/pi/Desktop/base/cam1/plaza1.jpg',plaza1)
#cv2.imshow('canny1', plaza1)
ret1, thresh1 =cv2.threshold (plaza1,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

#adquirir contornos  contornos1,jerarquia1 =
cv2.findContours(thresh1,cv2.RETR_LIST,cv2.CHAIN_APPROX_NONE)

lon_contornos1=len(contornos1)

```

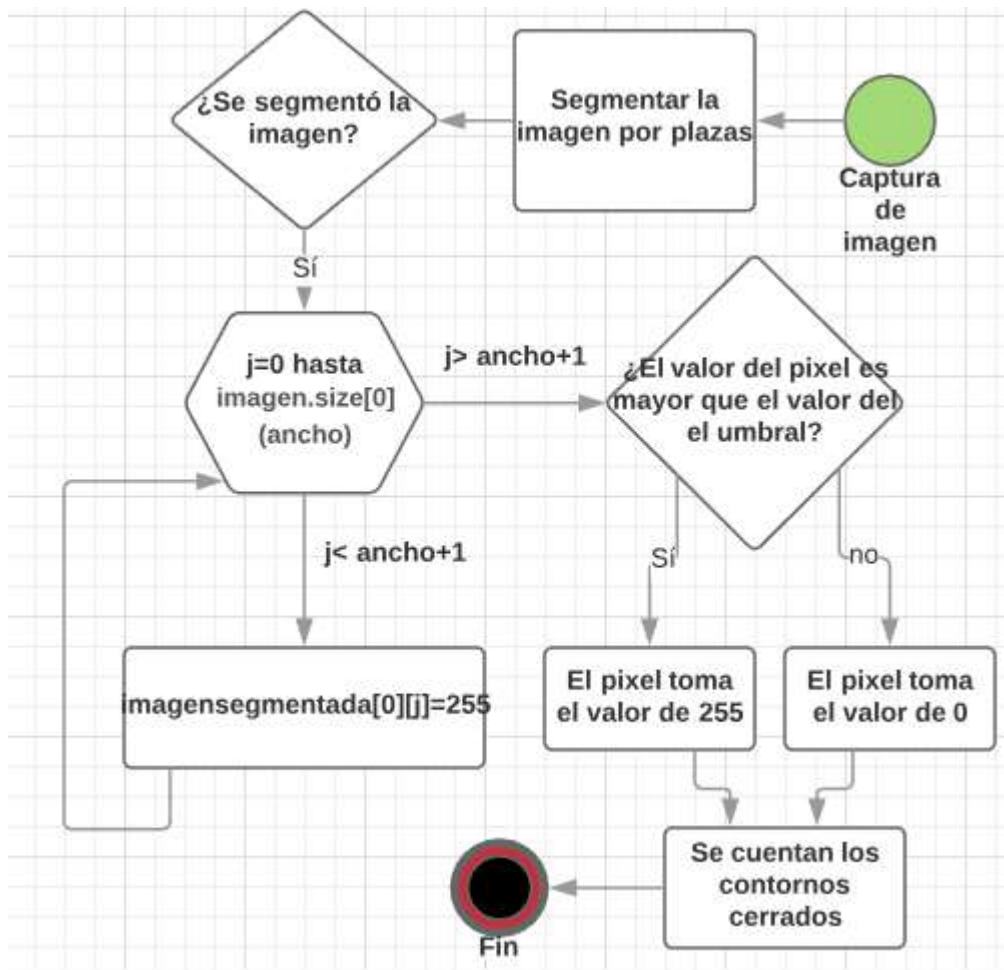


Ilustración 23 Diagrama de flujo para obtener los bordes

Al obtener la segmentación de la imagen se ejecuta un ciclo For para cambiar el valor de los pixeles de la parte superior, para crear un contorno abierto y verificar si se realiza el conteo exacto de los bordes cerrado, este ciclo es opcional, ya que no altera el producto final.

Luego se compara el valor de umbral con el valor de cada pixel de la imagen, si el valor del pixel es mayor que el valor de umbral se debe asigna un valor de 255 a dicho pixel, en caso contrario se le asigna un valor de 0, seguido se cuenta la cantidad de bordes cerrados y se almacenan.

Al ejecutar el anterior algoritmo se obtiene cada una de las plazas del parqueadero segmentadas y definidas por el borde o contorno cerrado (ver ilustración 24)

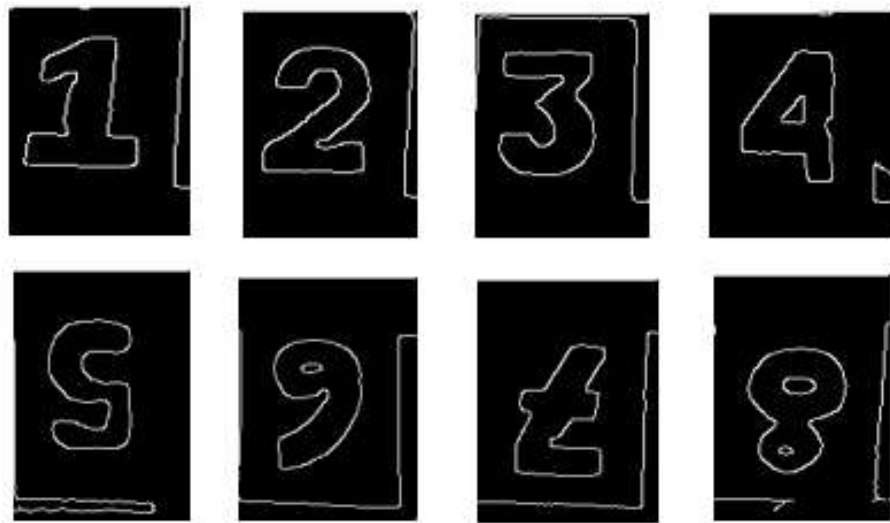


Ilustración 24. Bordes guardados de las imágenes segmentadas

Es posible que en cada una de las imágenes segmentadas se detecten varios contornos cerrados, esto implica que se debe escoger el contorno correcto que representa cada uno de los números de cada plaza. Y para ello se crean dos listas, una para almacenar todos aquellos contornos hallados y la otra es para almacenar el área que determina cada contorno.

Cuando se tienen almacenados los contornos se aplica un ciclo For, para recorrer la lista, dentro del ciclo se aplica la función `cv2.contourArea` para encontrar el área comprendida por el contorno cerrado y se almacena en la lista de área.

Se crea otro ciclo For para recorrer la lista que contiene las áreas y por medio de un algoritmo se identifica el área máxima de toda la lista, cabe resaltar que este proceso se aplica para cada una de las segmentaciones de imágenes, es decir para cada plaza del parqueadero (ver ilustración 25). En el siguiente código se muestra todo lo descrito en los últimos tres párrafos.

```
#####area1#####
area1=[]
for i in range(lon_contornos1):
    cnt1=contornos1[i]
    M1= cv2.moments(cnt1)
    area1.append(cv2.contourArea(cnt1))
print area1
#escoger el area mayor
area_max1=0
for i in range(lon_contornos1):
    if(area1[i] >area_max1):
```

```

    area_max1=area1[i]
    print ("area plaza1",area_max1)
    if ((area_max1>2000)and(area_max1<2900)):
        estado1=0
    else:
        estado1=1

```

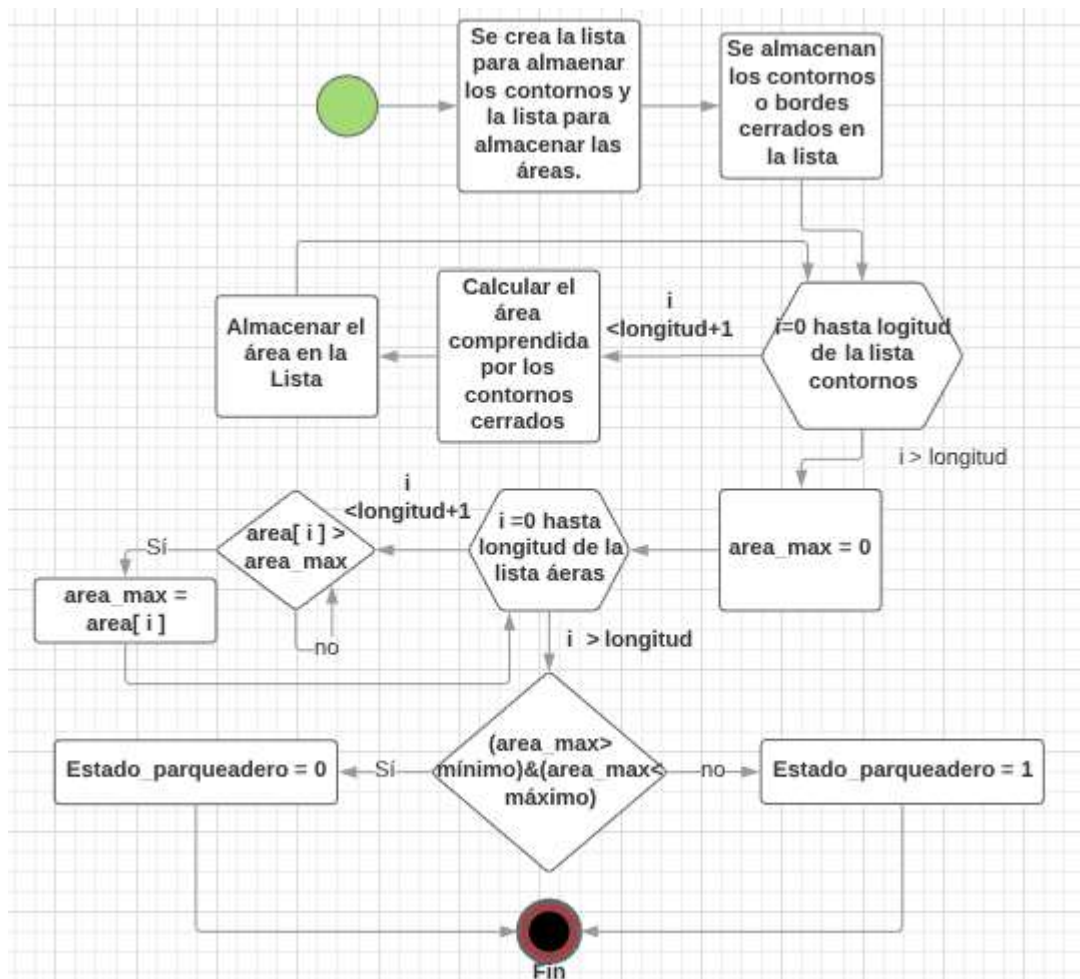


Ilustración 25 Diagrama de Flujo para determinar el estado del parqueadero

En el algoritmo de escoger el área máxima contenida en la lista, se crea una variable inicializada en cero, denominada $\text{área_max} = 0$, se procede a comparar cada uno de los valores de la lista con la variable y en caso de ser mayor el valor de la lista, la variable adquiere su valor, por ejemplo, si $\text{área_max} = 1234$ y $\text{área}[i] = 2000$, área_max toma el valor de $\text{área}[i]$, por lo tanto $\text{área_max} = 2000$. La comparación continúa hasta que se compare el último valor de la lista $\text{área}[i]$.

Para determinar el estado del parqueadero se utilizan dos valores constantes denominado en la ilustración 20 como mínimo y máximo, estos valores se obtienen de ejecutar 10 veces el algoritmo de extracción de área máxima, dichos valores se registran en la tabla 9. Entre mayor sea el número de ejecuciones, más exactos son los valores de mínimo y máximo.

Tabla 9. Datos obtenidos al ejecutar algoritmo presentado en la Ilustración 25.

ÁREAS DE PLAZAS VACIAS											MEDIA
plaza1	2269	2242	2254	2223	2263	2475	2259	2344	2319	2331	2297,9
plaza2	2333	2276	2306	2219	2303	2426	2382	2524	2508	2508	2378,5
plaza3	2506	2481	2477	2442	2485	2699	2539	2623	2609	2612	2547,3
plaza4	2693	2676	2679	2678	2686	2765	2714	2710	2724	2717	2704,2
plaza5	3624	3618	3605	3596	3616	3740	3740	3627	3612	3622	3640
plaza6	3409	3380	3393	3348	3388	3555	3341	3428	3434	3414	3409
plaza7	2827	2814	2829	2777	2813	3016	2775	2864	2877	2850	2844,2
plaza8	3288	3274	3287	3250	3276	3692	3472	3310	3325	3295	3346,9
plaza9	1772	1783	1766	1744	1765	1855	1843	1852	1886	1797	1806,3
plaza10	1977	1974	1981	1935	1955	1966	1965	1957	2012	2012	1973,4
plaza11	2053	2055	2060	2027	2044	2043	2050	2041	2083	2087	2054,3
plaza12	1682	1686	1695	1659	1673	1683	1694	1681	1715	1716	1688,4
plaza13	2330	2336	2330	2289	2320	2324	2329	2323	2351	2350	2328,2
plaza14	2667	2680	2680	2633	2661	2673	2676	2664	2703	2701	2673,8
plaza15	2351	2359	2362	2309	2328	2391	2390	2376	2389	2386	2364,1
plaza16	2922	2925	2928	2894	2908	2922	2924	2913	2946	2943	2922,5

Al tener los 10 valores de área cada plaza del parqueadero, se establece que todos los valores de las áreas máximas de cada plaza de las siguientes ejecuciones de código, tienen un valor cercano al promedio. Por lo tanto es necesario establecer un valor mínimo y un valor máximo que abarque todas las áreas de las plazas.

Para determinar estos valores se obtiene la diferencia entre el dato mayor y el dato menor de cada plaza, se escoge la diferencia más grande y se aproxima, que en este caso es de 442 y su aproximación es a 450, luego se suma el promedio con 450 para obtener el máximo y se resta el promedio con 450 para obtener el mínimo. Los datos obtenidos se registran en la tabla 10.


```
com.commit()
com.close()
time.sleep(5)
```

2.3.4. Alojamiento web público (*Hosting*)

Se adquiere un *hosting* público desde la página de GoDaddy denominado www.parqueaderoprototipo.com o dirección IP 160.153.128.35 y por medio de la herramienta cPanel incluida, se puede hacer uso de My SQL Database, Remote MySQL y phpMyAdmin.

En la configuración de usuario se puede habilitar los dispositivos que pueden acceder a nuestra base de datos de manera remota por medio de la opción Remote MySQL. Siguiendo las especificaciones se habilita el acceso a todos los usuarios, es decir a los 200 que permite el sitio.

Usando la herramienta de phpMyAdmin se crea la tabla denominada “estadoparqueadero” donde se almacenan los resultados de cada uno de los estados de las plazas del parqueadero, según los requerimientos y el diseño establecido en este mismo capítulo. Esta tabla sólo tiene permisos de escritura desde Python y lectura desde cualquier dispositivo móvil que contenga la aplicación Android diseñada.

La segunda tabla, denominada “usuarios”, almacena los datos de los usuarios tales como: nombre, número de Cedula y el número de placa del vehículo que se dejará en el parqueadero, siguiendo los requerimientos y la forma de diseño. Esta tabla tiene permisos de lectura y escritura desde la aplicación Android, es decir que el usuario puede registrar sus datos en el momento que desee.

En el cPanel se encuentra otra herramienta que es la carpeta de archivos privados, donde se alojan todos los archivos .php para el registro y la consulta de la base de datos. Se cuenta con un archivo principal denominado función.php, el cual contiene las instrucciones de conexión a la base de datos. Al cargar el archivo consulta.php y registro.php cada uno llama a la función.php para establecer la conexión y proceder a la consulta o registro de datos.

Función.php

```
<?php
header( 'Content-Type: text/html;charset=utf-8' );
function ejecutarSQLCommand($commando){
    $mysqli = new mysqli("160.153.128.35", "", "", "parqueadero");

    /* check connection */
    if ($mysqli->connect_errno) {
        printf("Connect failed: %s\n", $mysqli->connect_error);
```

```

        exit();
    }

    if ( $mysqli->multi_query($commando)) {
        if ($resultset = $mysqli->store_result()) {
            while ($row = $resultset->fetch_array(MYSQLI_BOTH)) {
                $resultset->free();

            }

        }
    }
    $mysqli->close();
}

function getSQLResultSet($commando){
    $mysqli = new mysqli("160.153.128.35", "", "", "parquadero");
    /* check connection */
    if ($mysqli->connect_errno) {
        printf("Connect failed: %s\n", $mysqli->connect_error);
        exit();
    }
    if ( $mysqli->multi_query($commando)) {
        return $mysqli->store_result();
    }
    $mysqli->close();
}
?>

```

Sólo se tiene presente un archivo de registro que accede solamente a la tabla Usuarios, para ingresar todos los datos suministrados por el usuario, como nombre, placa de vehículo y número de cedula, y dejar esta información almacenada en la base de datos.

registro.php

```

<?php include ('functions.php');
$nombres=$_GET['nombres'];
$cedula=$_GET['cedula'];
$placa=$_GET['placa'];
ejecutarSQLCommand("INSERT INTO `usuarios` (nombres, cedula, placa)
VALUES (
'$nombres' ,
'$cedula' ,
'$placa')
ON DUPLICATE KEY UPDATE `nombres`= '$nombres', `cedula`= '$cedula',
`placa`='$placa';");
?>

```

Se tienen dos archivos de consulta, uno para los datos del usuario y el otro para consultar el estado del parqueadero por medio del último Id auto incrementable, según lo establecido en la etapa de diseño.

consulta.php

```
<?php
include('functions.php');
$nombres=$_GET["nombres"];
if($resultset=getSQLResultSet("SELECT * FROM `usuarios` WHERE nombres='$nombres'")){
    while ($row = $resultset->fetch_array(MYSQLI_NUM)){
        echo json_encode($row);
    }
}
?>
```

consulta.php

```
<?php
include('functions.php');
$id=$_GET["id"];
if($resultset=getSQLResultSet("SELECT * FROM `estadoparqueadero` WHERE id='$id'")){
    while ($row = $resultset->fetch_array(MYSQLI_NUM)){
        echo json_encode($row);
    }
}
?>
```

2.3.5. Aplicación Android

Para el desarrollo de la aplicación móvil se ha utilizado el software Android Studio, el cual ha sido nombrado por Google como el IDE oficial para desarrollar aplicaciones para Android.

Para la emulación se instala un software denominado Genymotion el cual aprovecha la arquitectura x86 y ejecuta de una manera más fluida.

Archivo activity_main.xml (plantilla 1)

Siguiendo los requerimientos y diseño establecido en este mismo capítulo se procede a la implementación de la plantilla 1 sobre el software (*Ver ilustración 26*).

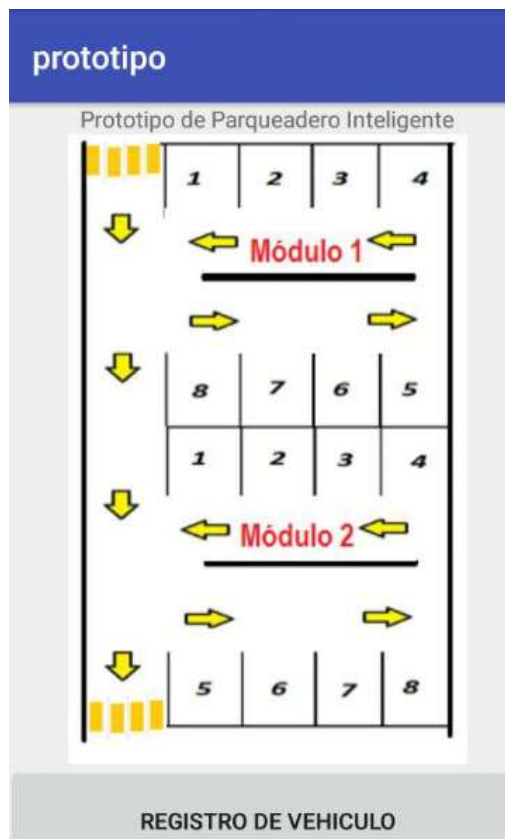


Ilustración 26 Implementación de la GUI de la plantilla 1.

Al visualizar el entorno gráfico se puede apreciar la imagen general del parqueadero prototipo y el botón establecido para comunicar con la siguiente plantilla de la aplicación; para ello se crea el archivo MainActivity.java que es donde se colocan cada una de las funciones que realizan los objetos creados en el Main_activity.xml o entorno gráfico, dicho código se encuentra en los Anexos, pero se direcciona a partir de la ilustración 13, presentada en la etapa de diseño.

Archivo activity_main2.xml

En esta parte se puede realizar el registro del usuario que está utilizando el parqueadero, ingresando los datos solicitados y oprimiendo el botón guardar; la aplicación llama al archivo registro.php e inserta los datos en la tabla “Usuarios” alojada en el hosting público.

Para la realizar la consulta de los datos del Usuario, sólo se escribe el nombre y se oprime el botón consultar y la aplicación utiliza los archivos consulta.php incluidos en la base de datos y se carga la información en la aplicación (ver ilustracion 27).

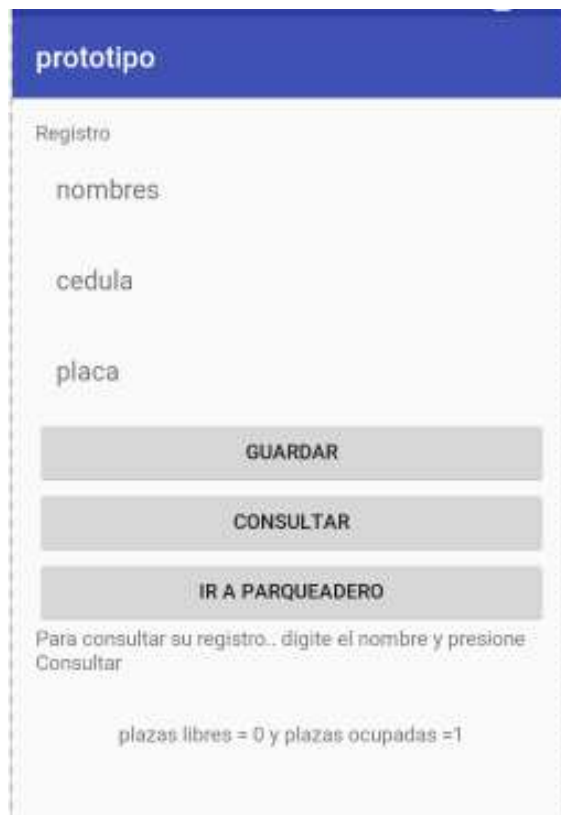


Ilustración 27.Segunda plantilla de la aplicación Android

Al visualizar el entorno gráfico para la segunda plantilla de la aplicación, se observa que hay tres EditText para que el usuario ingrese la información y los botones que permiten registrar y consultar de la base de datos y un botón adicional que permite ir a la plantilla 3 de la aplicación. Todas estas funciones se encuentran en el archivo MainActivity2.java que se muestra en los Anexos, estas funciones se dirigen a partir del diagrama mostrado en la ilustración 14 de la etapa de diseño.

Archivo activity_main3.xml

En esta parte se visualiza la consulta del estado de las plazas del parqueadero. Con el primer botón se consulta el último Id que se ha registrado en la tabla estadoparqueadero y lo carga en el EditText id, seguido se debe presionar el botón “consultar” para extraer los valores almacenados, asumiendo que 0 indica que el espacio o zona está libre y 1 indica que el espacio o zona está ocupada.

Como el parqueadero prototipo se encuentra dividido en dos módulos, donde cada módulo tiene 8 plazas, en el entorno gráfico se muestra una pequeña imagen del módulo 1 y frente a esta imagen las plazas de dicho módulo, de manera similar para el módulo 2.(ver ilustración 28).

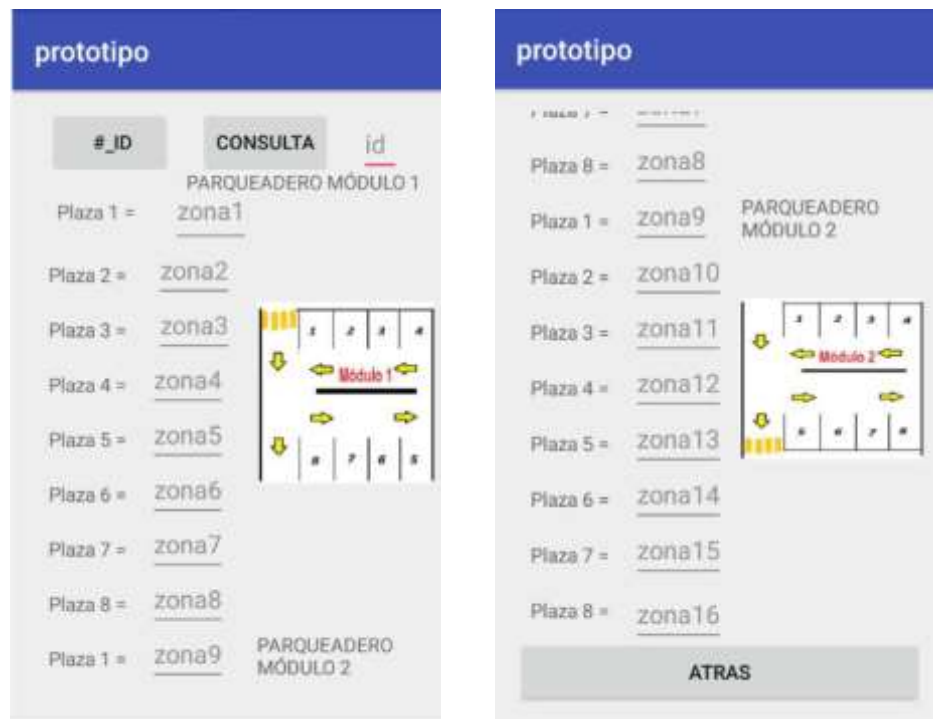


Ilustración 28. Tercera plantilla de la aplicación Android.

Las funciones implementadas en esta plantilla 3, se basan en el diagrama presentado en la ilustración 15 de la etapa de diseño. El código para se encuentra en los Anexos y tiene como nombre MainActivity3.java.

CAPITULO 3.

3.1. Resultados y Discusión.

En este trabajo se han implementado los lineamientos instaurados por la Unión Internacional de las Telecomunicaciones (ITU) descritos en el capítulo 1, El cual tiene como eje central, el mejoramiento en la prestación de un servicio utilizando tecnologías de la información y comunicación garantizando una conexión remota por medio de wifi, seguridad e infraestructura de datos/TIC, gestionando al usuario el aparcamiento en un tiempo cercano al real.

3.1.1. Objetivos para realizar pruebas

Objetivo general:

Visualizar el estado del parqueadero a través de la aplicación Android.

Objetivos específicos:

- Comprobar el almacenamiento de la captura con sus parámetros.
- Comprobar y visualizar el estado de las zonas del parqueadero cuando están libres.
- Comprobar y visualizar el estado de las zonas del parqueadero cuando están ocupadas.

3.1.2. Metodología a seguir en las pruebas

Se realiza la conexión de todas las partes del sistema para obtener el sistema de aparcamiento que brinda la gestión, para verificar el funcionamiento del sistema se plantea la siguiente metodología.

- Llevar a cabo pruebas funcionales, en la que se hace la verificación dinámica del comportamiento del sistema y para ello se planifican los aspectos que se deben examinar dentro del sistema.

- Al encontrar errores en el sistema, se llega a un planteamiento coherente, justificando la causa del error.

3.1.3. Aspectos a examinar

Para una correcta verificación se debe evaluar los siguientes aspectos:

- El sistema es capaz de realizar una correcta adquisición de la imagen y almacenarla en un disco interno, esta imagen debe cumplir con los parámetros establecidos en los requerimientos.
- El valor de área máxima de cada una de las plazas libres se sitúa dentro del rango establecido como máximo y mínimo, lo que implica que el estado de cada plaza le corresponde el valor cero, garantizando que está libre.
- Al ingresar un vehículo en una de las plazas, el valor del área máxima debe salirse del rango establecido, lo que implica que el estado en dicha plaza le corresponde el valor uno, garantizando que la plaza se encuentra ocupada. Se puede aumentar la cantidad de vehículos y se repite el proceso.
- Los valores establecidos para cada una de las plazas deben quedar registrados en la tabla “estadoparqueadero” de la base de datos.
- En la aplicación Android se debe cumplir las transiciones entre las 3 plantillas por medio del botón correspondiente.
- En la plantilla 2 de la aplicación se ingresa la información correspondiente al registro y se procede a verificar los datos almacenados en la tabla “usuarios” de la base de datos.
- Se realiza la consulta del estado del parqueadero por medio de la aplicación Android.

3.1.4. Resultados esperados.

- Imagen con los parámetros establecidos, tales como: Tamaño y formato.
- Estado de zonas con el valor cero, si en ella no se encuentra un vehículo.
- Estado de zonas con el valor uno, si en ella se encuentra un vehículo.
- Visualización del estado del parqueadero por medio de la aplicación Android.

3.1.5. Pruebas del sistema

Prueba 1: Cuando se ejecuta el programa y al colocar un vehículo en la primera plaza del módulo 1, se obtiene la *ilustración 29*.

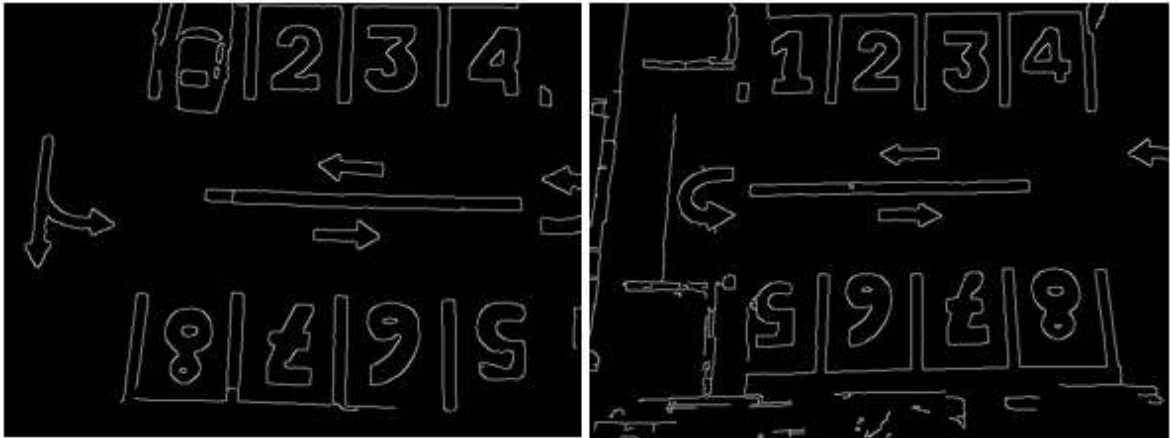


Ilustración 29. Resultado del parqueadero cuando se coloca un vehículo en la plaza 1 en el módulo 1

Se detectan los bordes por medio del procesamiento mostrado en el capítulo 2, donde se imprime en pantalla el estado del parqueadero dando como resultado el resgistro [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], aclarando que el registro de manera general almacena los estados del parqueadero por plazas, teniendo en cuenta que los ocho primeros valores de arreglos corresponden a las plazas del módulo1 y los últimos ocho valores de arreglos corresponde a las plazas del módulo 2, como se muestra en la ilustración 30.

[plaza1, plaza2, plaza3, plaza4, plaza5, plaza6, plaza7, plaza8, plaza1, plaza2, plaza3, plaza4, plaza5, plaza6, plaza7, plaza8]

Ocho plazas del módulo 1 Ocho plazas del módulo 2

Ilustración 30 registro general del estado de parqueadero

Al verificar la base de datos (*ver ilustración 42*), se evidencia que se e almacena el registro [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] para el ID 164.

Al presionar el botón de consulta de la aplicación Android (*ver ilustración 31*), se cargan los datos en la aplicación, teniendo en cuenta que el cero indica espacios libres y 1 indica los espacios vacíos.



Ilustración 31. Consulta del parqueadero a través de la aplicación Android cuando el ultimo Id es 164.

El anterior resultado indica que en procesamiento implementado el valor de área máxima del contorno cerrado de la plaza es de 8756, lo cual indica este valor se sale del rango establecido para la zona1 citado en el capítulo 2, este resultado es reflejado en la aplicación Android que debe tener el usuario y pueda así dirigirse directamente a las zonas que se encuentran libres para disminuir el tráfico dentro del parqueadero.

Prueba 2: Al colocar otro vehículo en la plaza 2 del módulo 1, se obtiene la ilustración 32 con el detector de bordes

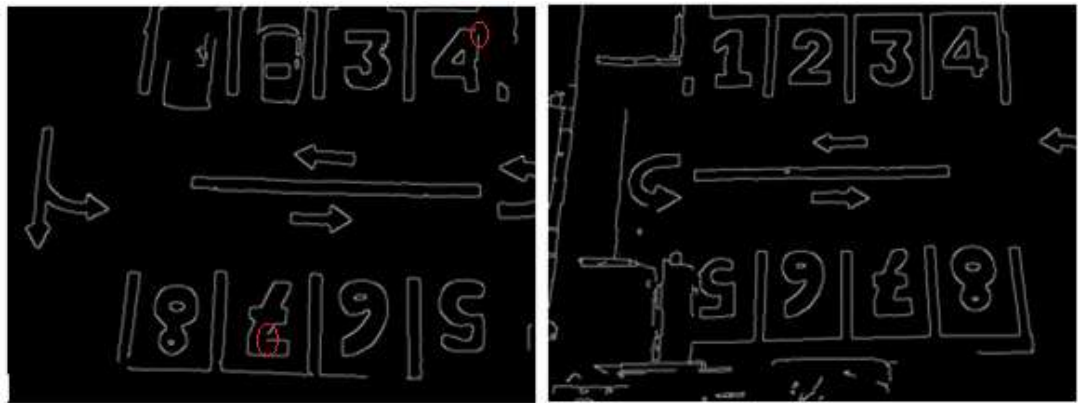


Ilustración 32 Resultado del parqueadero cuando se colocan vehículos en las plazas 1 y 2 en el módulo 1

Al verificar la base de datos (ver ilustración 42), se evidencia que se almacena el registro [1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0] para el ID 165.

Cuando se realiza el procesamiento de imagen se evidencia que en la plaza 4 y plaza 7 del módulo 1 no se detectan contornos cerrados, lo que implica que el área máxima

sea cero y se encuentre por fuera del rango establecido en el capítulo 2. Por lo tanto el estado en estas plazas corresponde al valor 1. Este error se explica en el campo 3.2 de este mismo capítulo. Al presionar el botón de consulta de la aplicación Android (ver *ilustración 33*), se cargan los datos en la aplicación, teniendo en cuenta que el cero (0) indica espacios libres y uno (1) indica los espacios ocupados.



Ilustración 33 Consulta del parqueadero a través de la aplicación Android cuando el ultimo Id es 165

Prueba 3: Se realiza de nuevo la prueba y se obtiene la ilustración 34.

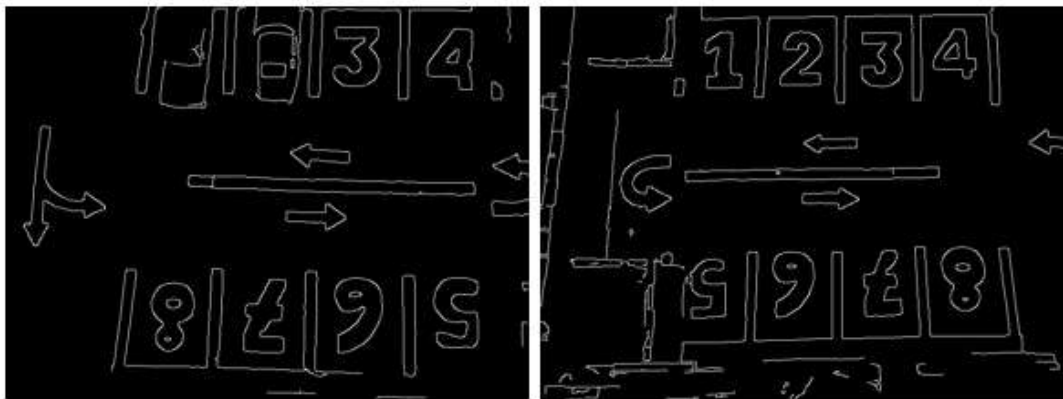


Ilustración 34. Resultado del procesamiento de imagen del parqueadero cuando se colocan vehículos en las plazas 1 y 2 en el módulo 1

Con el procesamiento de imagen se obtiene los siguientes datos correspondientes a la gestión de plazas en el parqueadero prototipo para el Id 166: [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]. Se puede verificar que los resultados son correspondientes a lo que se encuentra registrado en la base de datos (ver *ilustración 42*). Al presionar el botón de

consulta de la aplicación Android, se cargan los datos en la aplicación, recordando que cero indica espacios libres y uno indica espacios ocupados.



Ilustración 35.Consulta del parqueadero a través de la aplicación Android cuando el ultimo Id es 166

En este caso, cuando se incluye un vehículo en la plaza 2, el área máxima del contorno cerrado es 7986 y se encuentra por fuera del rango establecido en el capítulo 2 .

Prueba 4: Se prueba con más vehículos y se obtiene la ilustracion 36.

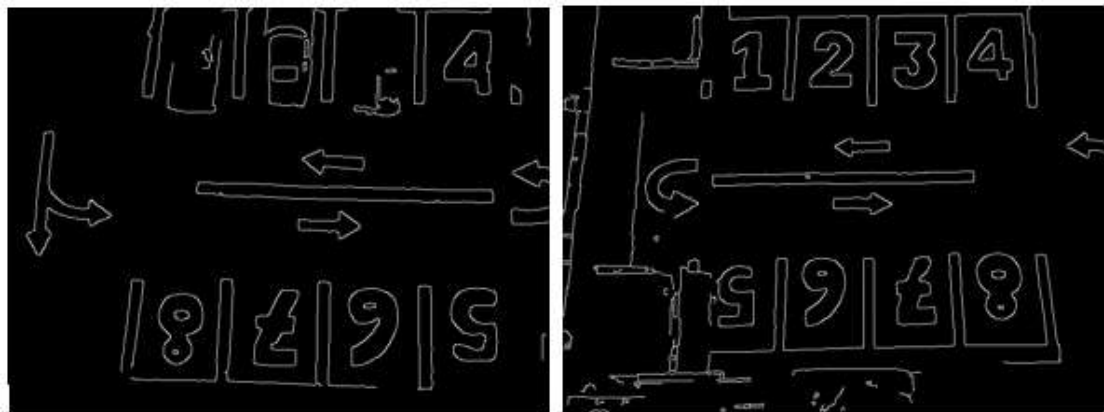


Ilustración 36.Resultado del parqueadero cuando se colocan vehículos en las plazas 1,2 y 3 en el módulo 1.

Se obtienen los siguientes registros y cada uno es almacenado en la base de datos con un correspondiente ID autoIncrementable: para el ID 167, [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], para el ID 169, [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] y, para el ID 170, [1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0], Se puede verificar que los resultados son correspondientes a lo que se encuentra registrado en la base de datos (ver *ilustración42*), El área Máxima de la zona 3 es de 600 y se ubica por fuera del rango por lo tanto le corresponde el valor uno.

Se obtienen las ilustraciones 37,38, 39, 40, 41 Las imágenes correspondientes al procesamiento y verificación de la aplicación Android.



Ilustración 37. Consulta del parqueadero a través de la aplicación Android cuando el ultimo Id es 167

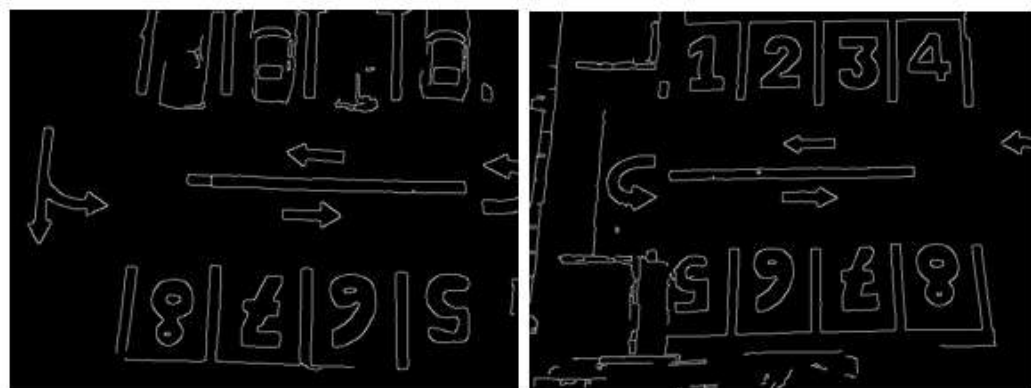


Ilustración 38. Resultado del parqueadero cuando se colocan vehículos en las plazas 1,2,3 y 4 en el módulo 1



Ilustración 39.Consulta del parqueadero a través de la aplicación Android cuando el ultimo Id es 169.

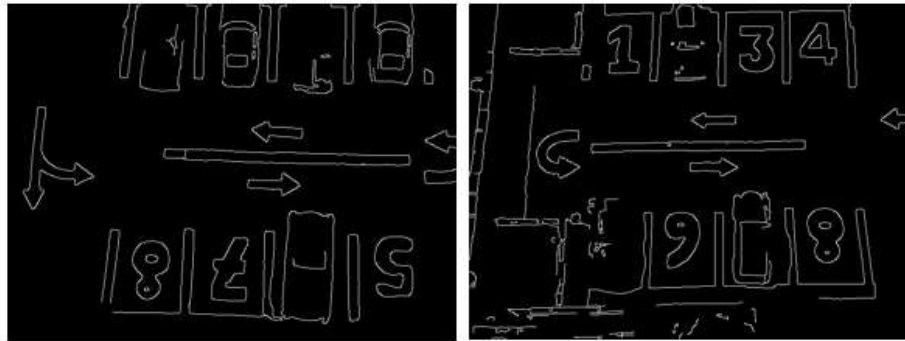


Ilustración 40.Resultado del parqueadero cuando se colocan vehículos en las plazas 1, 2,3 ,4 y 6 en el módulo 1 y plazas 2, 5 y 7 en el módulo 2

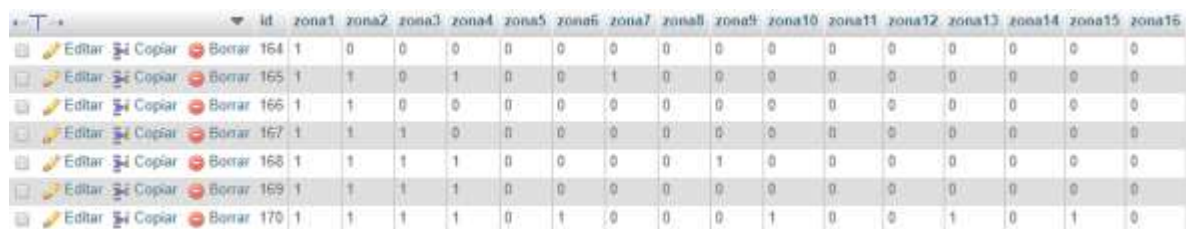


Ilustración 41.Consulta del parqueadero a través de la aplicación Android cuando el ultimo Id es 170.

Siempre que se incluya un vehículo en cada una de las plazas libres, el área máxima de los contornos cerrados formados cambiará de forma drástica; es decir, si en la imagen no se encuentra un contorno cerrado, dicha área tiende a cero. En caso contrario, si hay un vehículo y forma un contorno cerrado, el área será muy grande en comparación con la establecida para cada número. De manera que si hay un objeto muy grande que obstruya el número, automáticamente cambiará el área máxima permitida; por lo tanto, hay un cambio de estado de libre a ocupado, tal y como se observa en cada uno de los resultados.

El área máxima para las zonas del módulo 1 son: zona1 = 100, zona 2=8556, zona 3 = 600, zona4 = 9742, zona 6 = 10760 y El área máxima para las zonas del módulo 2 son: zona2 = 800, zona 5 = 70, zona7 = 4975.

El tiempo de actualización de los resultados en la base de datos se establece con el tiempo de espera presente al final del código de Python; se recomienda una actualización de los datos de cada 1 segundo o 2 segundos para no saturar el procesador de la raspberry pi3.



id	zona1	zona2	zona3	zona4	zona5	zona6	zona7	zona8	zona9	zona10	zona11	zona12	zona13	zona14	zona15	zona16
164	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
165	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0
166	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
167	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
168	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0
169	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
170	1	1	1	1	0	1	0	0	0	1	0	0	1	0	1	0

Ilustración 42 Consulta en la base de datos

3.2. Limitaciones.

3.2.1. Bajo nivel de luz.

El prototipo de parqueadero presenta una limitación cuando el nivel de luz es bajo, se procede a reducir el nivel de luz y realizar otra prueba.



Ilustración 43. Captura del parqueadero con bajo nivel de iluminación.

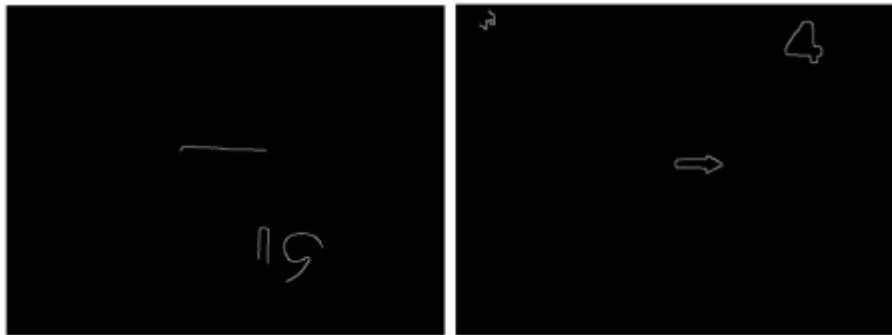


Ilustración 44. Resultado de la detección de bordes del parqueadero con bajo nivel de iluminación

En la ilustración 43 se puede observar el parqueadero con poco nivel de iluminación y en la ilustración 44, se observa el procesamiento de dicha imagen, en la cual no se logra detectar los bordes correspondientes de cada uno de los números de las plazas, esto genera que se envíe una lista de datos erróneos a la base de datos. Los datos registrados en la base de datos son [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1], lo cual deja claro que del módulo 2 se logró la detección de la plaza 4.

3.2.2. Fuente de luz directa.

Al colocar una fuente de luz directa, el parqueadero presenta una limitación (ver ilustración 45).



Ilustración 45. Captura del parqueadero con fuente de luz directa

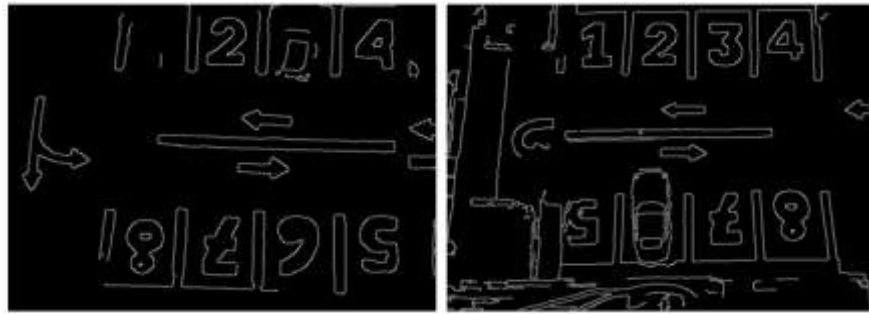


Ilustración 46. Resultado de la detección de bordes con fuente directa.

Una fuente de luz directa no permite que en la captura de la imagen se identifiquen con claridad los bordes, debido a que la fuente de luz se refleja directamente sobre algunas plazas, esto ocasiona que algunos de los contornos no aparezcan totalmente cerrados (ver ilustración 46) y, por ende, el área máxima tienda a cero. En este caso, las plazas afectadas son la plaza 5 del módulo 1 y las plazas 5 y 7 del módulo 2. Al analizar con detalle las plazas, encontramos que los contornos no están cerrados, esto es lo ocurrido en la prueba 2. (Ver ilustración 47).



Ilustración 47. Plazas que presentan falla con fuente de luz directa.

Lo anterior hace que se envíe el listado [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0] a la base de datos, lo cual es un dato erróneo.

3.3. Conclusiones

- El sistema de aparcamiento prototipo presenta un funcionamiento similar al esperado cuando se garantiza una iluminación adecuada (de forma no directa), debido a que el suelo de la maqueta está recubierto de un material que refleja las ondas de luz, lo cual ocasiona que al sensor el parqueadero se obtenga una imagen con un destello de luz. Este destello no deja captar los bordes de los números ubicados en las plazas.
- Al escoger la tecnología de video cámara para la gestión de aparcamiento presentada en el capítulo 1, se establece una conexión IP entre la tarjeta Raspberry, la base de datos y la aplicación Android. La seguridad de los datos se tiene en cuenta en el alojamiento web público (Hosting) y una tecnología de datos TIC en los estándares y políticas de la tecnología implementada. Por lo tanto se da cumplimiento a las recomendaciones de las TIC para ciudades inteligentes, cabe aclarar que estas recomendaciones se emplean en el momento de crear y brindar un servicio.
- Debido a la constante interacción del Backend con la base de datos, se establece el uso de conexiones seguras (HTTPS) y bases de datos encriptadas. Lo anterior se logra desde el alojamiento web público (Hosting) utilizado y mencionado en el capítulo 2, el cual brinda confidencialidad por medio de la certificación SSL, que incluye una encriptación SHA-2 y de 2048 bits, creando un túnel seguro y evitando que se espíe cualquier información ingresada o solicitada. Esta certificación tiene un costo anual.
- Se desarrolla una aplicación Android que permite visualizar el estado actual del parqueadero en el momento en que el usuario realice la consulta en la base de datos pública.
- Es importante garantizar un nivel de luz dentro del parqueadero homogéneo en cada una de las plazas, para que el detector de bordes pueda identificar el cambio de umbral establecido o, de lo contrario, ajustar el valor de umbral establecido en el código creado en Python, ya que hay muchos factores que afectan directamente, tales como el cambio del día y el estado del tiempo.
- El tiempo de captura y el procesamiento de imágenes es ajustable por el propietario del parqueadero ya que depende del espacio de almacenamiento de la imagen y de qué tan frecuentado sea el parqueadero; lo ideal es que este tiempo sea lo más pequeño posible y así poder gestionar la información en tiempo real, así como lo establecen los servicios de la ciudades Inteligentes, citadas por la Unión Internacional de las Telecomunicaciones (ITU).

- Cualquier usuario puede acceder a la gestión de aparcamiento a través de la aplicación Android diseñada usando una conexión IP o inalámbrica.
- Este trabajo de grado se puede aplicar en parqueaderos reales teniendo en cuenta las conclusiones anteriores y además sirve de base para generar muchos trabajos de grado futuros, tales como:
 - Generar una red neuronal para el reconocimiento de vehículos en el parqueadero.
 - Desarrollar aplicaciones para la seguridad y control del registro de vehículos.
 - Desarrollo de un sistema de iluminación óptimo aplicable a un parqueadero, para disminuir los factores que afectan los niveles de luminosidad en el parqueadero
 - Implementar múltiples cámaras para un parqueadero con más de 20 plazas.
 - Desarrollar múltiples aplicaciones Android para reservar lugares dentro del parqueadero, con un tiempo mínimo de espera y forma de pago en línea por uso del servicio.

3.4. Anexos

Código de las funciones de la plantilla 1, denominado MainActivity.java

```
package com.example.alejandro.prototipo;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
    Button btnRegistro;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnRegistro =(Button)findViewById(R.id.btnRegistro);
        btnRegistro.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent =new Intent(MainActivity.this, Main2Activity.class);
                startActivity(intent);
            } });
    } }
```

Código de las funciones de la plantilla 2, denominado MainActivity2.java.

```
android:id="@+id/btnParqueadero"/>
package com.example.alejandro.prototipo;
import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import org.json.JSONArray;
import org.json.JSONException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
public class Main2Activity extends AppCompatActivity {
```

```

Button btnconsultar, btnGuardar, btnParqueadero;
EditText etCedula, etNombres, etPlaca;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main2);
    btnParqueadero=(Button)findViewById(R.id.btnParqueadero);
    btnParqueadero.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent =new Intent(Main2Activity.this, Main3Activity.class);
            startActivity(intent);
        }
    });
    btnconsultar = (Button)findViewById(R.id.btnConsultar);
    btnGuardar = (Button)findViewById(R.id.btnGuardar);
    etCedula = (EditText)findViewById(R.id.etCedula);
    etNombres = (EditText)findViewById(R.id.etNombres);
    etPlaca = (EditText)findViewById(R.id.etPlaca);
    btnconsultar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            new
ConsultarDatos().execute("http://parqueaderoprototipo.com/sitios/si/consulta.php?nombres="+etNo
mbres.getText().toString());
        }
    });
    btnGuardar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            new
CargarDatos().execute("http://parqueaderoprototipo.com/sitios/si/registro.php?nombres="+etNombr
es.getText().toString()+"&cedula="+etCedula.getText().toString()+"&placa="+etPlaca.getText().toStr
ing());
        }
    });
}

private class CargarDatos extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {
        // params comes from the execute() call: params[0] is the url.
        try {
            return downloadUrl(urls[0]);
        } catch (IOException e) {
            return "Unable to retrieve web page. URL may be invalid.";
        }
    }
    // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(String result) {
        Toast.makeText(getApplicationContext(), "Se almacenaron los datos correctamente",
Toast.LENGTH_LONG).show();
    }
}

private class ConsultarDatos extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {

```

```

        // params comes from the execute() call: params[0] is the url.
        try {
            return downloadUrl(urls[0]);
        } catch (IOException e) {
            return "Unable to retrieve web page. URL may be invalid.";
        }
    }
    // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(String result) {
        JSONArray ja = null;
        try {
            ja = new JSONArray(result);
            etNombres.setText(ja.getString(1));
            etCedula.setText(ja.getString(2));
            etPlaca.setText(ja.getString(3));
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
    private String downloadUrl(String myurl) throws IOException {
        Log.i("URL", ""+myurl);
        myurl = myurl.replace(" ", "%20");
        InputStream is = null;
        // Only display the first 500 characters of the retrieved
        // web page content.
        int len = 500;
        try {
            URL url = new URL(myurl);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setReadTimeout(10000 /* milliseconds */);
            conn.setConnectTimeout(15000 /* milliseconds */);
            conn.setRequestMethod("GET");
            conn.setDoInput(true);
            // Starts the query
            conn.connect();
            int response = conn.getResponseCode();
            Log.d("respuesta", "The response is: " + response);
            is = conn.getInputStream();
            // Convert the InputStream into a string
            String contentAsString = readIt(is, len);
            return contentAsString;
            // Makes sure that the InputStream is closed after the app is
            // finished using it.
        } finally {
            if (is != null) {
                is.close();
            }
        }
    }
    public String readIt(InputStream stream, int len) throws IOException,
    UnsupportedEncodingException {
        Reader reader = null;
        reader = new InputStreamReader(stream, "UTF-8");
        char[] buffer = new char[len];

```

```

        reader.read(buffer);
        return new String(buffer);
    }
</RelativeLayout>

```

Código de las funciones de la plantilla 3, denominado MainActivity2.java.

```

    android:layout_alignParentEnd="true" />
package com.example.alejandro.prototipo;
import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import org.json.JSONArray;
import org.json.JSONException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
public class Main3Activity extends AppCompatActivity {
    Button btnAtras,btnConsulta_id, btnConsulta;
    EditText etId,etZona1, etZona2, etZona3, etZona4, etZona5, etZona6, etZona7, etZona8,
etZona9, etZona10, etZona11, etZona12, etZona13, etZona14, etZona15, etZona16;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);
        btnAtras=(Button)findViewById(R.id.btnAtras);
        btnAtras.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent =new Intent(Main3Activity.this, Main2Activity.class);
                startActivity(intent);            }
        });
        btnConsulta = (Button)findViewById(R.id.btnConsulta);
        btnConsulta_id=(Button)findViewById(R.id.btnConsulta_id) ;
        etId=(EditText)findViewById(R.id.etId);
        etZona1 = (EditText)findViewById(R.id.etZona1);
        etZona2 = (EditText)findViewById(R.id.etZona2);
        etZona3 = (EditText)findViewById(R.id.etZona3);
        etZona4 = (EditText)findViewById(R.id.etZona4);
        etZona5 = (EditText)findViewById(R.id.etZona5);

```

```

etZona6 = (EditText)findViewById(R.id.etZona6);
etZona7 = (EditText)findViewById(R.id.etZona7);
etZona8 = (EditText)findViewById(R.id.etZona8);
etZona9 = (EditText)findViewById(R.id.etZona9);
etZona10 = (EditText)findViewById(R.id.etZona10);
etZona11 = (EditText)findViewById(R.id.etZona11);
etZona12 = (EditText)findViewById(R.id.etZona12);
etZona13 = (EditText)findViewById(R.id.etZona13);
etZona14 = (EditText)findViewById(R.id.etZona14);
etZona15 = (EditText)findViewById(R.id.etZona15);
etZona16 = (EditText)findViewById(R.id.etZona16);
btnConsulta_id.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        new
Main3Activity.ConsultarDatos().execute("http://parqueaderoprototipo.com/sitios/consultar_id/consult
a.php"); } });
btnConsulta.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        new
Main3Activity.ConsultarDatos().execute("http://parqueaderoprototipo.com/sitios/parqueadero/consul
ta.php?id=" + etId.getText().toString());
    }
}); }
private class ConsultarDatos extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {
        // params comes from the execute() call: params[0] is the url.
        try {
            return downloadUrl(urls[0]);
        } catch (IOException e) {
            return "Unable to retrieve web page. URL may be invalid.";
        }
    }
    // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(String result) {
        JSONArray ja = null;
        try {
            ja = new JSONArray(result);
            etId.setText(ja.getString(0));
            etZona1.setText(ja.getString(1));
            etZona2.setText(ja.getString(2));
            etZona3.setText(ja.getString(3));
            etZona4.setText(ja.getString(4));
            etZona5.setText(ja.getString(5));
            etZona6.setText(ja.getString(6));
            etZona7.setText(ja.getString(7));
            etZona8.setText(ja.getString(8));
            etZona9.setText(ja.getString(9));
            etZona10.setText(ja.getString(10));

```

```

        etZona11.setText(ja.getString(11));
        etZona12.setText(ja.getString(12));
        etZona13.setText(ja.getString(13));
        etZona14.setText(ja.getString(14));
        etZona15.setText(ja.getString(15));
        etZona16.setText(ja.getString(16));
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

private String downloadUrl(String myurl) throws IOException {
    Log.i("URL", ""+myurl);
    myurl = myurl.replace(" ", "%20");
    InputStream is = null;
    // Only display the first 500 characters of the retrieved
    // web page content.
    int len = 500;
    try {
        URL url = new URL(myurl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(10000 /* milliseconds */);
        conn.setConnectTimeout(15000 /* milliseconds */);
        conn.setRequestMethod("GET");
        conn.setDoInput(true);
        // Starts the query
        conn.connect();
        int response = conn.getResponseCode();
        Log.d("respuesta", "The response is: " + response);
        is = conn.getInputStream();
        // Convert the InputStream into a string
        String contentAsString = readIt(is, len);
        return contentAsString;
        // Makes sure that the InputStream is closed after the app is
        // finished using it.
    } finally {
        if (is != null) {
            is.close();
        }
    }
}

public String readIt(InputStream stream, int len) throws IOException,
    UnsupportedEncodingException {
    Reader reader = null;
    reader = new InputStreamReader(stream, "UTF-8");
    char[] buffer = new char[len];
    reader.read(buffer);
    return new String(buffer);
}
}

```

3.5. Bibliografía

- [1] S. V. Srikanth, Pramod P. J, Dileep K. P, Tapas S, Mahesh U. Patil, Sarat Chandra Babu N, 'Design and Implementation of a prototype Smart PARKing (SPARK) System using Wireless Sensor' 2009.
- [2] International telecommunication Union, ITU-T Focus Group on Smart SustainableCities (technical report), ITU-2015
- [3] Isabel Cebrián, Rocío Ingelmo, Ernst & Young Francisco Javier Martínez, Ernst & Young Tomas Pastor, Ernst & Young Carlos Plasencia, Ferrovial Servicios Sergio Serna, Luis Valero, Libro blanco Smart cities, (Pagina 6, 14,19 y 22), Ernst and Young, Ferrovial and Madrid Network, 2012.
- [4] Luis Felipe Herrera-Quintero, Walid David Jalil-Naser, Klaus Banse, J. Javier Samper- Zapater, Smart cities approach for Colombian Context. Learning from ITS experiences and linking with government organization, Smart Cities Symposium Prague (SCSP), Prague, Czech Republic, 2015.
- [5]Proyecto Enter.co, (2012),cinco casos exitosos de ciudades inteligentes, [Online]. Available:<http://www.enter.co/cultura-digital/ciudadinteligente/5-casos-exitosos-de-ciudades-inteligentes/>
- [6] Andrea Zanella,, Nicola Bui, Angelo Castellani, Lorenzo Vangelista and Michele Zorzi, (2014) Internet of Things for Smart Cities - In IEEE Internet Of Things Journal, Vol. 1, No.1.
- [7] IrisNet: Internet-scale Resource-Intensive Sensor Network Service, [Online]. Available: <http://www.intel-iris.net>
- [8] J.Chinrungrueng, Udomporn Sunantachaikul, Satien Triamlumlerd Smart Parking: an Application of optical Wireless Sensor Network, Proceedings of the 2007 International Symposium on Applications and the Internet Workshops.
- [9] Sangwon Lee, Dukhee Yoon and Amitabha Ghosh, Intelligent Parking Lot Application Using Wireless Sensor Networks, Proceedings of IEEE conference, 978-1-4244-2249-4/08
- [10] Barton, J.,J. Buckley, B. O'Flynn, S.C. O'Mathuna, J.P. Benson, T.O'Donovan, U. Roedig, and C. Sreenan, The D-Systems Project - Wireless Sensor Networks for Car- Park Management, Proceedings of the 65th IEEE Vehicular Technology Conference, Dublin, Ireland, pp.170-173, Apr 2007.

- [11] V. Tang, Y. Zheng, J. Cao: 'An intelligent car park management system based on wireless sensor networks'. Proc. Int. Symp. Pervasive Computing and Applications, pp. 65–70. Urumqui, 2006.
- [12] Z. Pala, N. Inanc: 'Smart parking applications using RFID technology'. Proc. First Annual – RFID Eurasia, 2007
- [13] <http://www.itu.int/en/ITU-T/ssc/Pages/info-ssc.aspx>
- [14] ITU-T, FG-SSC, TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, ITU-T Focus Group on Smart Sustainable Cities, An overview of smart sustainable cities and the role of information and communication technologies, (10/2014).
- [15] UIT-T Y.2221, TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, series y: global information infrastructure, internet protocol aspects and next-generation networks, (01/2010).
- [16] ITU-T, FG-SSC, TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, ITU-T Focus Group on Smart Sustainable Cities, Intelligent sustainable buildings for smart sustainable cities, (05/2015)
- [17] Luis ROC, Optimotion, RFID, [Online]. Available: <http://optimotion.tech/identificacion-radiofrecuencia>.
- [18] R. S. Chen, C. C. Chen, K. C. Yeh, Y. C. Chen and C. W. Kuo, "Using RFID technology in food produce traceability", WSEAS Transactions on information science and applications, vol. 5, no. 11, (2008), pp. 1551-1560.
- [19] M. S. Jian, C.L. Lee, K. S. Yang, and N. Y. Huang, "Context and Location Aware Public/Personal Information Service based on RFID System Integration" Proc. Of WSEAS Inter. Conf. on Sys., July 2008.
- [20] Prof. Gayatri Bhandari, Mrinal Bari, Shweta Borse, Ashwini Gaikwad, Reshma Kadam in their research "Parking Navigation System Based on RFID and IR Sensor" (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (2), 2015, 1468-1472
- [21] H. Al-Kharusi, I. Al-Bahadly, "Intelligent Parking Management System Based on Image Processing", School of Engineering and Advanced Technology, Massey University, Palmerston North, New Zealand, 14 April 2014.
- [22] D. B. L. Bong, K. C. Ting and K. C. Lai, "Integrated Approach in the Design of Car Park Occupancy Information System (COINS)", IAENG International Journal of Computer Science, IJCS, vol. 35, no. 1, (2008).
- [23] Orozco Sarasti, G. Llano Ramírez. (2014). Aplicaciones para redes VANET Estimación del exponente de Hurst y dimensión fractal para el análisis de series de tiempo de absorbancia UV-Vis. Ciencia e Ingeniería Neogranadina, 24, febrero 2014.

- [24] Divya Chadha, Reena, Vehicular Ad hoc Network (VANETs): A Review, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3, Issue 3, March 2015.
- [25] Rongxing Lu, Xiaodong Lin, Haojin Zhu, Xuemin, An Intelligent Secure and Privacy-Preserving Parking Scheme Through Vehicular Communications, IEEE Transactions on Vehicular Technology (Volume: 59, Issue: 6, July 2010)
- [26] MS Rupa Rani, Prof. Sapna Khapre, Prof. Nishant M. Borkar, "A Survey On Providing Privacy Of Navigation In Vehicular Network Adhoc Network (Vanets)", International Journal of Emerging Trends & Technology in Computer Science, volume 4, Issue 1, January-February 2015.
- [27] M. Gahlan, V. Malik, D. Kaushik, "GPS Based Parking System", COMPUSOFT, An international journal of advanced computer technology, (Volume-V, Issue-I) January – 2016.
- [28] National instruments, <http://www.ni.com/white-paper/7142/es/>
- [29] Vanessa W.S. Tang, Y. Zheng, J. Cao, "An Intelligent Car Park Management System based on Wireless Sensor Networks", 1st International Symposium on Pervasive Computing and Applications, 2006, pp. No. 65 – 70. IEEE 2006.
- [30] Jihoon Yang, Jorge Portilla and Teresa Riesgo, Smart Parking Service based on Wireless Sensor Networks Yang, Jihoon, J. Portilla, and T. Riesgo. "Smart parking service based on wireless sensor networks." IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society. IEEE, 2012.
- [31] Cnet, Catalogo products, genius-facecam-310, [Online]. <https://www.cnet.com/products/genius-facecam-310-web-camera/specs/>
- [32] Pcel, Catalogo Cámaras, Microsoft-JSD-00008-81999, Monterrey
- [33] Digital Learning SL, Academia Android (marzo 2018), [Online]. Available: <https://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse>