FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

IMPLEMENTACIÓN DE UN PROTOTIPO PARA LA GESTIÓN DE PARQUEADEROS EN LA UNIVERSIDAD DE LAS AMÉRICAS

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de Información

Profesor Guía

Ing. Luis Alberto Morales Escobar

Autor

David Ramiro Pérez Alvear

Año

2016

DECLARACIÓN PROFESOR GUÍA

“Declaro haber dirigido este trabajo a través de reuniones periódicas con el (los) estudiante(s), orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”.

_____

Luis Alberto Morales Escobar
Ingeniero en Electrónica y Control
CI: 171514654-2

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.”

_____

David Ramiro Pérez Alvear

C.I. 040131929-8

AGRADECIMIENTOS

Agradezco en primer lugar a Dios que me ha dado la vida, y a mis padres quienes siempre me han brindado su apoyo para cumplir mis metas y objetivos.

También quiero brindar mi gratitud a mi tutor guía, el Ingeniero Luis Alberto Morales Escobar, por la orientación y apoyo en el desarrollo de este trabajo de titulación.

DEDICATORIA

Primeramente quiero dedicar este trabajo de titulación a Dios, mi Señor, y a mis padres que con todo su esfuerzo siempre han estado ahí para brindarme todo su apoyo.

RESUMEN

La Universidad de las Américas ofrece estacionamientos para sus estudiantes y docentes en las diferentes sedes que esta tiene. Tomando en cuenta la necesidad de los usuarios que utilizan dichos parqueaderos para poder estacionarse de manera más ágil y un control del uso del estacionamiento para el personal administrativo, se procede a implementar este prototipo para la gestión de parqueaderos en la Universidad.

Primeramente se realiza un análisis del esquema actual de la gestión de parqueaderos de la Universidad, verificando que en efecto es necesaria la implementación de un sistema que permita gestionar el uso del estacionamiento tanto para los clientes como para el personal administrativo.

La implementación del prototipo para la gestión de parqueaderos en la Universidad de las Américas, viene dada en dos etapas. La primera es todo lo relacionado con el hardware del sistema, es decir la parte tangible que incluye: maqueta, sensores infrarrojos, leds indicadores, placa, Arduino, Ethernet Shield, cables, etc. La detección de presencia de un vehículo en el estacionamiento viene dada por un sensor infrarrojo. La segunda etapa es todo lo relacionado con el software del sistema, es decir, la programación de la placa electrónica, la programación para la aplicación Web y Android y el desarrollo de la base de datos donde se almacenará la información de los estados de todos los parqueaderos y los registros suscitados en los mismos. El sistema utiliza peticiones HTTP con método POST para realizar la actualización de los estados y la inserción de registros; por lo cual es necesaria la conexión a una red local con acceso a Internet mediante un cable de red Ethernet con conector RJ45.

La aplicación en Android permitirá observar los estados de los estacionamientos, distinguidos por colores; verde para disponible y rojo para ocupado. En la aplicación Web también se tiene acceso a los estados de los

parqueaderos y adicionalmente se tienen opciones de administrador para el personal administrativo. Entre estas opciones se tiene el acceso a todos los registros de eventos dados en cada uno de los estacionamientos permitiendo conocer la hora exacta en la que cada estacionamiento fue ocupado o desocupado. También se tiene acceso a un registro general diario en el que se podrá observar el porcentaje de uso del estacionamiento por día. Además, el administrador podrá crear o eliminar parqueaderos, actualizar el estado de un estacionamiento. Finalmente el administrador podrá crear nuevos usuarios administradores y actualizar sus datos personales y su contraseña.

ABSTRACT

Universidad de las Americas offers parking for students and teachers in the different venues that this has. Considering the need of customers who use these parking lots to park more agile and control parking privileges to administrative staff, we proceed to implement this prototype to manage parking in the University. First an analysis of the current scheme of parking management at the University is made to verify that in fact the implementation of a system which manages the use of parking for both customers and for administrative staff is necessary. The prototype implementation of parking management at Universidad de las Americas, is given in two stages. The first is everything related to the system hardware, which is the tangible part including: model, infrared sensors, LED indicators, PCB, Arduino, Ethernet shield, cables, etc. Detecting presence of a vehicle in the parking lot is given by an infrared sensor. The second stage is everything related to the software system, including programming the Arduino, programming for Android and Web application and development of the database where information of the states and records of all parking will be stored. The system uses HTTP POST requests for updating the states and inserting records; so the connection to a local network with Internet access is required via a cable Ethernet with RJ45 connector.

The Android application will allow observing parking lot sates, distinguished by color; green for available and red for occupied. Web application also has access to the parking states and further options for administrative staff. One of these options is the access to all records of events given in each of the parks allowing the exact time that each parking was occupied or unoccupied. It also has access to a daily general log where you can see the parking percentage use per day. In addition, the administrator can create or delete parking lots and update the status of a parking lot. Finally, the administrator can create new administrators and update their personal information and password. At the stage of tests and results of the implemented prototype, possible errors are discarded in the operation.

ÍNDICE

INTRODUCCIÓN

El intervalo de tiempo que transcurre al buscar un parqueadero disponible en los diferentes estacionamientos de la Universidad de las Américas en muchas ocasiones es alto; por esta razón se ha planteado implementar un prototipo para la gestión de parqueaderos que permita ubicar un espacio disponible de manera rápida y ágil; esto mediante una aplicación en la que se podrá observar los estados de los estacionamientos.

Según NetMarketShare (2015) el sistema operativo Android lidera el mercado con un 53.54%; es por eso que la aplicación móvil se la desarrollará en la plataforma Android, además que el modelo de desarrollo para esta plataforma es de código abierto. Para usuarios que no dispongan de un celular con este sistema operativo; tendrán acceso al estado de los parqueaderos mediante la aplicación web desarrollada.

Hoy en día la gran mayoría de los estudiantes de la Universidad posee un celular con acceso a Internet; y por ende a la aplicación desarrollada mediante Android o vía Web. Por otro lado en el área administrativa es muy importante llevar un registro de cada estacionamiento con la fecha y hora en la que fue ocupado o desocupado, al igual que el tiempo en el que se lo utilizó; esto permitirá obtener reportes que permitirán conocer en que horario el estacionamiento permanece vacío o lleno; con el fin de dar un mejor servicio a los clientes que utilizan a diario las instalaciones de los estacionamientos de la Universidad.

OBJETIVOS:

General:

Implementar un sistema cliente-servidor para monitoreo y administración de parqueaderos de la Universidad de las Américas, a través de un prototipo electrónico y los sistemas informáticos web y móvil.

Específicos:

- Analizar el esquema actual mediante el cual se gestiona el parqueadero de la Universidad.
- Diseñar un prototipo que conste de sensores de emulación de los estacionamientos de la sede Queri, módulos de transmisión de datos; para enviar a un Servidor las señales de estado de cada uno de ellos.
- Desarrollar un sistema web, para monitoreo y administración del uso de parqueaderos del sistema.
- Desarrollar una aplicación móvil y web, que permita obtener la visualización de estacionamientos disponibles.

Justificación del proyecto

El aumento inesperado de las vías congestionadas en las vías cercanas a la Universidad es consecuencia del mal uso de las mismas, debido a que los ciudadanos al no encontrar un parqueadero disponible, estacionan sus vehículos en lugares no adecuados o prohibidos generando congestión en las vías.

Por tal motivo se implementará una aplicación móvil y web desarrollada en el sistema operativo Android, la cual recibirá la información de los estados de los parqueaderos, consultando a un servidor que recibe información de sensores electrónicos que se colocarán en dichos espacios, la aplicación creada mostrará cuantos y cuales parqueaderos se encuentran disponibles. Los usuarios buscarán alternativas que permitirán ganar tiempo para acudir a las diferentes actividades de la Universidad.

Adicionalmente mediante el desarrollo de la aplicación para el personal administrativo se podrá llevar un registro más detallado del número de veces e intervalos de tiempo que los usuarios utilizan los estacionamientos, y las horas

a las que el estacionamiento se encuentra totalmente lleno con el propósito de buscar soluciones para mejorar el servicio y que los clientes se sientan a gusto. Las pruebas del sistema implementado se realizarán en un prototipo con un modelo a escala del parqueadero de la sede Queri con sus respectivos estacionamientos, emulándolos debido a la dificultad que conlleva su implementación en un entorno real. Si se desea realizar la implementación en todos los parqueaderos de las distintas sedes de la Universidad se deberá instalar el sistema con los sensores en cada uno de los estacionamientos ya que estos dispositivos son los que envían información para que funcione el sistema. Cabe tomar en consideración que se debe analizar una alternativa de sensores robustos para aplicaciones de detección de presencia en exteriores.

Con la implementación de este sistema, en el futuro se podrá utilizar para cualquier tipo de parqueadero de acuerdo a la necesidad, por ejemplo: Universidades, Centros Comerciales, Estacionamientos Públicos.

Beneficios esperados

El prototipo para la gestión de parqueaderos de la Universidad de las Américas va a permitir obtener los siguientes beneficios concretos:

- Reducir el tiempo que un usuario necesita para estacionar su vehículo, encontrando un espacio disponible rápidamente observando los estados de todos los estacionamientos en la aplicación vía Web o mediante la aplicación en Android.
- Mayor control en cuanto a los registros de cada uno de los estacionamientos, permitiendo conocer la fecha y hora exacta en la que cada parqueadero fue ocupado o desocupado; y el tiempo en el que fue utilizado.
- Permitir tener un registro del porcentaje de uso del estacionamiento por hora durante el día, para así conocer en que horario el parqueadero

permanece lleno; y así buscar soluciones con el fin de brindar un mejor servicio a los usuarios.

Alcance del proyecto

El alcance de este trabajo de titulación es elaborar una aplicación móvil basada en el sistema operativo Android, que servirá para los conductores (cliente), que les facilitará la localización de parqueaderos disponibles, esto mediante un prototipo que contará con dispositivos electrónicos los cuales permitirán emitir una señal de disponible u ocupado. Otra de las fases del trabajo es desarrollar una aplicación web para usuarios que no posean un celular con sistema operativo Android, con la cual se podrá acceder a la misma información. Para el personal Administrativo del parqueadero; en la aplicación web tendrán la opción de llevar un registro del uso de dichos espacios.

Con la finalidad de realizar las pruebas del sistema implementado, se elaborará una maqueta con un modelo a escala del parqueadero de la sede Queri con sus respectivos estacionamientos que contengan los sensores electrónicos que permitirán conocer el estado del parqueadero y el número de estacionamientos disponibles, y de esta manera la información se actualizará en una base de datos en un servidor en un periodo de tiempo establecido. La aplicación que se desarrollará recibirá la información de la base de datos permitiendo conocer cuántos y cuáles parqueaderos están disponibles para su uso.

Se realizará la fase de diseño de la aplicación para el personal administrativo de los estacionamientos, que permitirá tener un mejor control de estos espacios, y también colaborará con reportes de un promedio de tiempo en que los usuarios utilizan el servicio de parqueo, con el fin de mejorar el servicio a los clientes.

Antecedentes

En la Universidad de las Américas, se ha realizado la implementación de parqueaderos públicos en las diferentes sedes existentes; esta implementación se ha realizado debido a la necesidad que presentan los estudiantes y docentes. Uno de los factores de mayor incidencia en la congestión vehicular a los alrededores de la universidad es el mal uso de las vías, que en su mayoría se encuentran ocupadas por vehículos estacionados en lugares prohibidos, reduciendo el espacio de circulación en las mismas.

Al no poder visualizar de una manera previa y simple los estacionamientos disponibles, los conductores se ven obligados a realizar una búsqueda tediosa hasta encontrar un parqueadero libre en el cual puedan estacionar su auto, causando pérdida de tiempo al conductor y de esta manera aumentando el problema de congestión en las vías cercanas a la Universidad.

Actualmente se pude observar sistemas de parqueaderos mediante sensores en los centros comerciales; los cuales muestran el estado del parqueadero a través de colores; un ejemplo es el color rojo mostrando un estado de ocupado y el color verde un estado de disponible; este modelo es el que permitirá dirigirse a un sector en donde haya parqueaderos disponibles de manera más eficiente.

La telefonía celular ha ido evolucionando a lo largo de la historia, el Ecuador tuvo la oportunidad de empezar a utilizar esta tecnología a partir de 1993; hoy en día el teléfono celular más que un lujo es una necesidad. Desde el año 2007 el sistema operativo Android fue anunciado y en la actualidad es uno de los sistemas más utilizados en los llamados ´Smartphone´ que cada día siguen abarcando todo el mercado.

En el mundo han sido desarrolladas millones de aplicaciones móviles y web; que han permitido un avance importante en la sociedad mediante los servicios que estas prestan. En Colombia se ha desarrollado una aplicación llamada "*Parkiando"*, la cual ayuda a los conductores a encontrar el parqueadero más apropiado, a la vez que ofrece un espacio para generar visibilidad de sus servicios.

1. Capítulo I: Marco Teórico

En el presente capítulo se realiza una descripción de los componentes del sistema a ser utilizados, con la finalidad de facilitar su entendimiento en lo concerniente a su fundamentación teórica. Dichos puntos abarcan los siguientes temas: sensores infrarrojos, sistemas embebidos, comunicación por SPI, modelo TCP/IP, Protocolo HTTP; métodos POST y GET, aplicaciones Android y aplicaciones Web.

Con el sustento teórico presentado, se pretende puntualizar los conceptos necesarios para el desarrollo del proyecto como tal, los cuales mediante su aplicación, permitirán dar solución al problema planteado.

1.1. Sensores infrarrojos

1.1.1. Descripción

“Los sensores infrarrojos son dispositivos electrónicos que nacieron en los década de los 90, los cuales detectan la radiación emitida por los materiales calientes, transformándola en una señal eléctrica” (Universidad Politécnica de Valencia, s.f.).

Los sensores de infrarrojos son dispositivos que detectan la radiación electromagnética infrarroja del ambiente. Si sólo se dispone del sensor fototransistor se llaman sensores pasivos usados para medir las radiaciones que provienen de los objetos, pero cuando se basa en la combinación entre un emisor y receptor en un mismo circuito se llaman sensores activos, en este caso el emisor es un led infrarrojo y como receptor un fototransistor. (Castro, 2014, p. 12).

1.1.2. Características

Para muchas aplicaciones se utilizan ópticas que reducen el campo visual, pero que involucran presencia de temperatura de conmutación, aquí es donde intervienen los sensores infrarrojos. El sensor infrarrojo requiere que exista una comunicación entre receptor y emisor de tipo lineal, para esto es importante la línea de vista para una efectiva transmisión. Este tipo de sensores puede detectar cualquier objeto ya sea en la más densa oscuridad. La transmisión de datos se realiza por medio de ondas de muy alta frecuencia (similar a las ondas de radio), como las infrarrojas, pero tienen limitaciones, como el ángulo y distancia, tienen que estar muy cerca y casi de frente para que exista transferencia de datos (Universidad Politécnica de Valencia, s.f.).

1.1.3. Clasificación

Es muy complicado realizar una clasificación única, debido a la gran cantidad de sensores infrarrojos que existen actualmente, la siguiente es una clasificación general y común.

- Reflectivo: Este tipo de sensor está conformado por un Led Infrarrojo que es el emisor de luz infrarroja y el fototransistor que es el encargado de medir la radiación proveniente del reflejo de la luz emitida por el Led emisor. Este Sensor es sensible a la luz que se presenta en el ambiente, pudiendo así perjudicar a las medidas realizadas. Otro factor importante que se debe tomar en cuenta es el coeficiente de reflectividad del objeto, ya que este depende del tipo de superficie. Su uso común son aplicaciones como seguidores de línea, de alineamiento, detección de material reflectante, como papel, tarjetas de IBM, cintas magnéticas.

- Modulado: El sensor infrarrojo modulado funciona de la misma manera que el sensor reflectivo, con la diferencia que utiliza la emisión de una

señal modulada, reduciendo así la influencia de la iluminación en el ambiente. Son sensores orientados a la detección de presencia.

- Sensores de Ranura: Este tipo de sensor detecta objetos que pasan entre el emisor y el receptor. Se utiliza comúnmente para control industrial, por ejemplo en los dispositivos *encoders ópticos* para controlar: radio de giro, velocidad y dirección del movimiento del eje del motor.

1.1.4. Aplicaciones

Principalmente los sensores infrarrojos se han diseñado para la detección de objetos; dependiendo de su forma, color y superficie, inclusive en ambientes con condiciones extremas. El Led emisor de este dispositivo electrónico tiene la apariencia de un led común y corriente, pero la diferencia está en que la luz que emite no es visible, y solo puede ser percibida por un dispositivo electrónico. Los sensores infrarrojos cada día son más utilizados, se puede citar algunos aparatos tales como: control de reproductores portátiles, apertura de puertas, aire acondicionado, entre otros. (Naranjo, Montoya, Tobón & Visbal, s.f.).

1.2. Sistemas embebidos

1.2.1. Descripción

Un sistema embebido es básicamente un sistema electrónico diseñado para realizar funciones específicas. Para esto ocupa uno o varios procesadores digitales ya sean microprocesadores o microcontroladores, los cuales son los encargados de la lógica del sistema a controlar. (Úbeda, 2009, p. 2).

1.2.2. Componentes de los Sistemas Embebidos

Hardware:

Por lo general consiste en un módulo electrónico que procesa la información generada por sensores u otros dispositivos electrónicos. Este módulo puede ser un:

- Microprocesador.
- Microcontrolador ya sea de 4, 8, 16 o 32 bits.

El desarrollo del sistema embebido puede tener varios requerimientos. Entre éstos, se pueden citar:

- Tamaño: Generalmente deberá ser reducido,
- Temperatura del ambiente de aplicación.
    - Cuando la carga en el circuito es alta la temperatura puede variar de 0ºC hasta 70ºC.
    - En campos como el industrial o automotriz puede llegar hasta 125ºC
- Robustez: Se debe considerar niveles de vibración, o posibles golpes para el diseño del sistema.
- Consumo de energía: Siempre analizando la solución más eficiente, buscando el menor consumo de energía posible. (Úbeda, 2009, p.p. 2-3)

Software:

Para el software del sistema, se tendrán en cuenta los requerimientos específicos de acuerdo a la aplicación. Por lo general, los recursos pueden ser limitados, entre estos: la cantidad de memoria, el nivel de procesamiento, espacio en almacenamiento, etc. La utilización de un sistema operativo va a depender del sistema y será analizado en el diseño del mismo. Si no se utiliza un sistema operativo, se procederá al uso de microcontroladores. Estas decisiones dependerán de los requerimientos del sistema, ya sean técnicos y económicos. (Úbeda, 2009, p.p. 3-4).

1.2.3. Arquitectura

La arquitectura de un sistema embebido se muestra en la Figura 1.

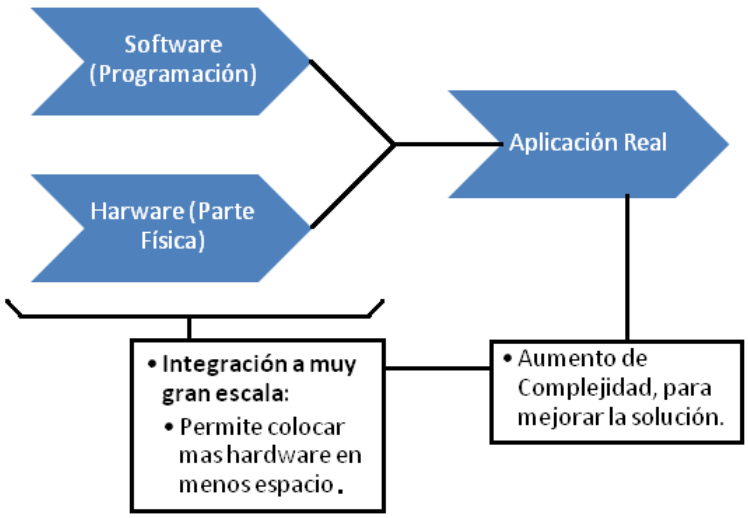


Figura 1. Arquitectura Sistema embebido.
Tomado de Carpintero, s.f., p. 7.

1.2.4. Aplicaciones

Por lo general los sistemas embebidos se aplican en los ámbitos: industrial, educativo, y para consumo. Entre los cuales se pueden enunciar algunos campos de aplicación:

- Electrodomésticos, multimedia, juguetes.
- Domótica.
- Equipos industriales para instrumentación, automatización y producción.
- Equipos para telecomunicaciones, etc.

Actualmente, los fabricantes de semiconductores ofrecen productos relacionados con las aplicaciones en las que se emplean. (Úbeda, 2009, pp. 4-6).

Algunas de las aplicaciones se pueden observar en la Figura 2.

Figura 2. Aplicaciones de Sistema embebido.
Tomado de Carpintero, s.f., p. 15.

1.3. Comunicación por SPI (Serial Peripheral Interface)

1.3.1. Descripción

SPI es un bus que contiene 3 líneas de trasmisión, sobre el cual se transmite paquetes de 8 bits. Cada línea lleva la información entre los dispositivos que se encuentran conectados. Los dispositivos pueden actuar como transmisor o receptor al mismo tiempo, siendo así una comunicación full dúplex. Una de estas líneas es de reloj y las otras dos son las líneas donde se transfieren los datos, una en cada dirección. El bus SPI utiliza un simple registro de desplazamiento para la transmisión de la información. (López, 2012)

1.3.2. Dispositivos:

Existen dos tipos de dispositivos que se conectan al bus, estos pueden ser:

- Maestros: Genera las señales de reloj y control e inicializa la transferencia de datos sobre el bus SPI.
- Esclavos: El dispositivo Maestro lo controla.

Cada dispositivo esclavo conectado al maestro es controlado por este a través de la línea selectora conocida como Select Slave (SS) o Chip Select (CS), cuando esta línea selectora ha sido seleccionada el dispositivo esclavo es activado. Normalmente cada dispositivo esclavo tiene su propia línea selectora. (López, 2012)

1.3.3. Especificaciones del Bus

Cada una de las líneas del bus SPI transmite la información en una misma dirección.

Las funciones de cada línea son:

- MOSI (Master Out Slave In), en esta línea se transportan datos desde el maestro hacia el esclavo.
- MISO (Master In Slave Out), en esta línea se transportan datos desde el esclavo hacia el maestro.
- SCLK *(Clock*), en esta línea se transporta la señal de la línea de reloj, generada por el dispositivo maestro y que sincroniza la transferencia de los datos.
- CS = Chip Select o SS Slave Select, esta línea permite la selección de un esclavo, activándolo. (López, 2012)

1.3.4. Funcionamiento del Reloj

La sincronización de la trasferencia de los datos viene dada por la línea de reloj (SCLK) del bus respectivo. En cada ciclo de reloj un BIT es transmitido.
Existen dos 2 bits de configuración para el bus SPI, los cuales son:

- **CPOL (Polaridad de Reloj),** La cual determina si el estado de reposo de la línea de reloj está en estado bajo (0) o en estado alto (1).
- **CPHA (Reloj de Fase),** La cual determina en cual filo de la señal de reloj los datos son desplazados ya sean hacia afuero o hacia dentro. Si es 0, los datos sobre la línea MISO son detectados en cada filo de

subida y los datos sobre la línea MOSI son detectados en cada filo de bajada. Si es 1, los datos sobre la línea MISO son detectados en cada filo de bajada y los datos sobre la línea MOSI son detectados cada filo de subida.

Se tienen 4 combinaciones diferentes. Para que dos dispositivos SPI puedan comunicarse entre sí, estos deben tener la misma Fase de Reloj y la misma Polaridad de Reloj. (López, 2012)

1.4. MODELO TCP/IP (Transfer Control Protocol – Internet Protocol)

1.4.4. Descripción

El modelo TCP/IP describe las funciones que se ejecutan en las capas respectivas del conjunto TCP/IP. Fue creado a principios de la década de los 60, se lo conoce como modelo de Internet. Este modelo define cuatro categorías de funciones que se deben cumplir para que las comunicaciones tengan éxito.

1.4.5. Arquitectura

Las capas del Modelo TCP/IP se ilustran en la Figura 3:

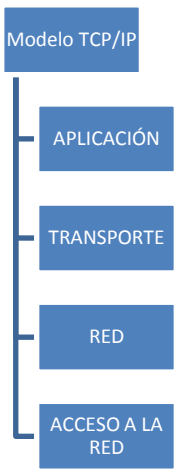


Figura 3. Capas Modelos TCP/IP.
Tomado de: Cisco CCNA, 2014.

- Aplicación: Datos para el usuario, dialogo y control de codificación.
- Trasporte: Permite la comunicación entre dispositivos de áreas distintas.
- Internet: Determina la mejor ruta en la red.
- Acceso a la Red: Controla tanto los dispositivos de acuerdo al hardware como los medios que conforman la red.

1.4.6. Unidad de Protocolo y Encapsulación

Mientras los datos de la aplicación se transmiten por los medios de la red, varios protocolos agregan información en cada nivel, a esto se conoce como encapsulación. La forma de los datos en cualquier capa se llama Unidad de Datos del Protocolo, conocida como PDU. De acuerdo al protocolo utilizado cada capa encapsula las PDU que recibe de la capa que le antecede. Cada capa tiene su propio PDU:

- Aplicación: Datos.
- Transporte: Segmento
- Red: Paquete
- Acceso a la Red: Trama

Los términos de la encapsulación de protocolos se establecen como se muestra en la Figura 4.

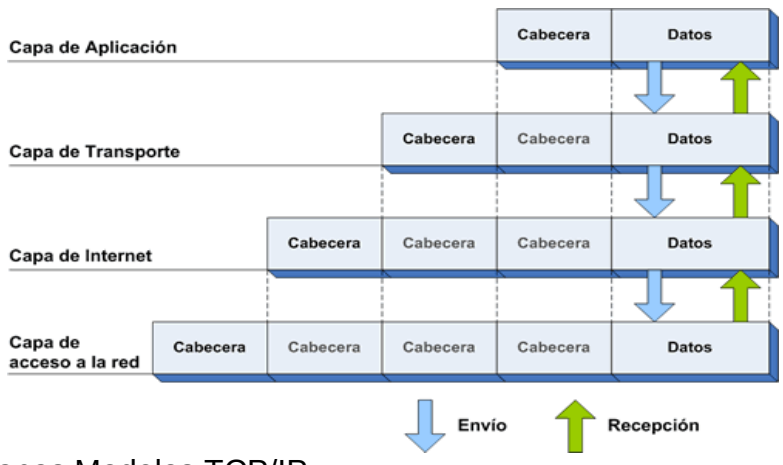


Figura 4. Capas Modelos TCP/IP. Tomado de Cisco CCNA, 2014.

1.4.7. Protocolos TCP/IP

En la Figura 5 se puede observar los protocolos TCP/IPA continuación una lista de protocolos.

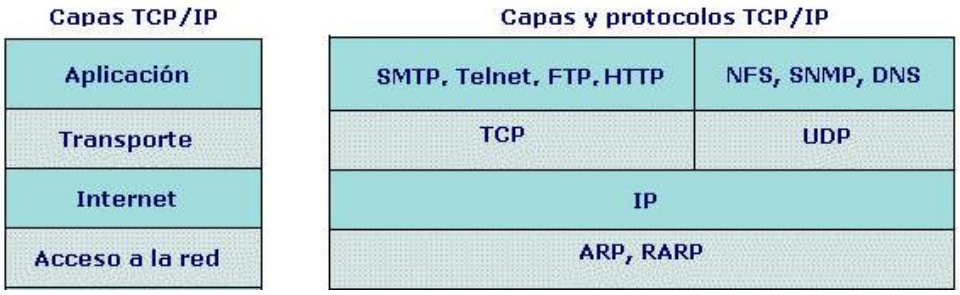


Figura 5. Capas Modelos TCP/IP.
Tomado de: Cisco CCNA, 2014.

1.5. HTTP: Métodos POST & GET

1.5.1. Descripción

El protocolo de transferencia de hipertexto, llamado HTTP, es el más utilizado en Internet. Permite la transferencia de mensajes con encabezados que indican su contenido. Este protocolo admite la transferencia de archivos con formato HTML entre un servidor web (url) y un cliente (navegador web). A continuación se explicará los métodos GET Y POST.

GET Y POST son métodos del protocolo HTTP el cual está compuesto por:

- Request: Petición al servidor
- Response: Respuesta de dicha petición.

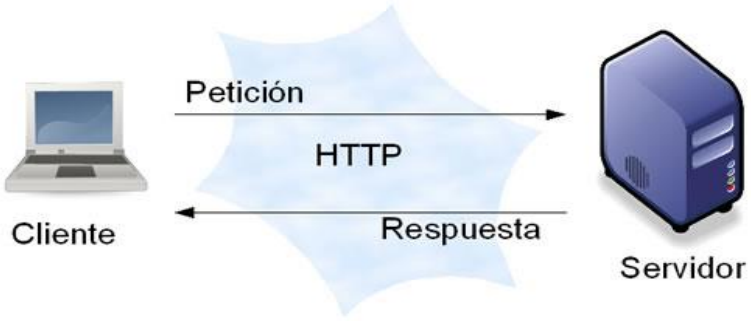


Figura 6. Capas Modelos TCP/IP.
Tomado de: CCM, s.f.

1.5.2. Definiciones

Mediante GET lo que se realiza es obtener información del servidor, esto puede ser un archivo o información de una base de datos. Para esto se realiza un Request al servidor que será procesado para luego retornar un Response. Mediante POST se realiza el envío de información por parte del cliente para que sea procesada y así poder agregar o actualizar información en el servidor.

Para esto se realiza un Request al servidor a través de un formulario que será procesado para luego retornar un Response. (Ardissone, 2011)

1.6. Sistema operativo Android

1.6.1. Descripción

El Sistema Operativo Android fue creado por la compañía Google, y está pensado y desarrollado desde la ideología *OpenSource* (Código Libre), por lo cual ha tenido un éxito enorme y gran aceptación en el poco tiempo de vida que tiene. (Cuturacion, 2014).

Este Sistema Operativo implementa una arquitectura en la que cualquier aplicación puede tener acceso a las capacidades del dispositivo móvil. Está desarrollado sobre el *Kernel* del Sistema Operativo Linux. Además, se utiliza una máquina personalizada virtual que fue diseñada para optimizarlos recursos de hardware y de memoria en el entorno móvil. (Molina, Sandoval & Toledo, 2012, p. 37).

1.6.2. Características

Las principales características del Sistema Operativo Android son:

- Software de código abierto.
- Utiliza SQLite como gestor de base de datos.
- Núcleo basado en el Kernel del Sistema Operativo Linux.
- Soporta HTML, HTML5, Adobe Flash Player.
- Tienda de aplicaciones pagadas o gratuitas (Google Play) (Androidos, 2012).

1.6.3. Arquitectura

La arquitectura de Android se muestra en la Figura 7.

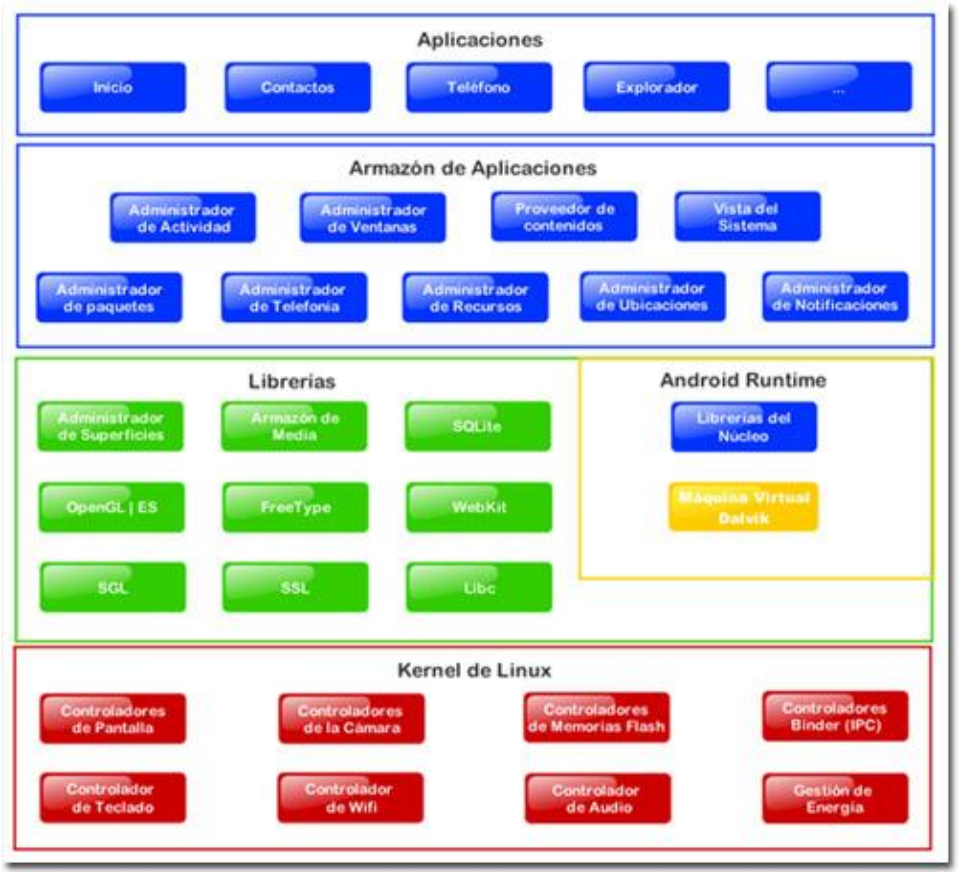


Figura 7. Arquitectura Android.

Tomado de: Androidos, s.f.

- **Aplicaciones:** Incluye correo electrónico, software para mensajes de texto, calendario, mapas, navegador web, contactos, entre otros.
- **Arquitectura de Aplicaciones:** La arquitectura permite la reutilización de componentes, es decir cualquier aplicación puede dar a conocer sus capacidades y cualquier otra puede hacer uso de las mismas.
- **Bibliotecas:** incluye bibliotecas de C/C++, las cuales son utilizadas por diferentes componentes del sistema.
- **Runtime:** Incluye bibliotecas base que brindan gran parte de funciones del lenguaje Java. Cada aplicación en Android ejecuta su proceso en una instancia de la máquina virtual, llamada *Dalvik*.
- **Núcleo Kernel Linux:** Android depende del sistema operativo Linux para los servicios base como: gestión de memoria, seguridad, pila de red, entre otros. (Androidos, 2012).

1.6.4. Versiones

A continuación se listan las plataformas lanzadas desde 2010 hasta la fecha:

- Gingerbread: Android 2.3 Nivel de API 9 (2010)
- Honeycomb: Android 3.0 Nivel de API 11 (2011)
- Android 3.1: Nivel de API 12 (2011)
- Android 3.2: Nivel de API 13 (2011)
- Ice CreamSandwich: Android 4.0 Nivel de API 14 (2011)
- JellyBean: Android 4.1 Nivel de API 16 (2012)
- KitKat: Android 4.4 Nivel de API 19 (2013) (Universidad Politécnica de Valencia, 2014).

1.6.5. Aplicaciones y Play Store

Android cuenta con su propio espacio de aplicaciones, que en un principio se llamó *Market*, actualmente tiene el nombre de *Play Store*. Desde esta se pueden descargar aplicaciones según la necesidad del usuario. El Play Store ofrece tanto aplicaciones gratuitas como pagadas. (Cuturacion, 2014).

1.7. Aplicaciones Web

1.7.1. Descripción

Una aplicación web es de tipo cliente-servidor, donde el cliente, el servidor y el protocolo mediante el que se comunican se encuentran estandarizados. El protocolo HTTP pertenece a los protocolos de comunicaciones TCP/IP, los cuales permiten la comunicación entre distintos computadores para la transmisión de información (Luján, 2002, p. 48).

1.7.2. Características

Entre las características de aplicaciones web, se destacan las siguientes:

- Compatibilidad multiplataforma.
- Actualización constante.
- Acceso inmediato y desde cualquier lugar.
- Seguridad en los datos. (Riyas, s.f.).

1.7.3. Estructura y Arquitectura

En las aplicaciones web por lo general se tienen tres niveles: el nivel superior que es el la interacción con el usuario (navegador web), el nivel inferior que provee la información necesaria que será consultada por el nivel superior (base de datos) y el nivel intermedio que se encarga de procesar los datos consultados (servidor web) (Luján, 2002, p. 47).

Por lo general las aplicaciones web se basan en la arquitectura cliente-servidor, como se muestra en la Figura 8:

- Cliente (el navegador web)
- Servidor (el servidor web) (Luján, 2002, p. 54).

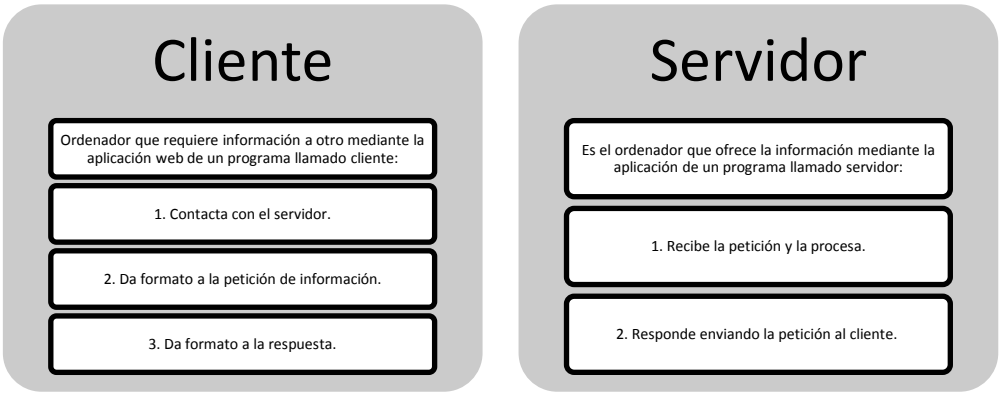


Figura 8. Estructura Cliente Servidor.
Tomado de: Carrera, 2008.

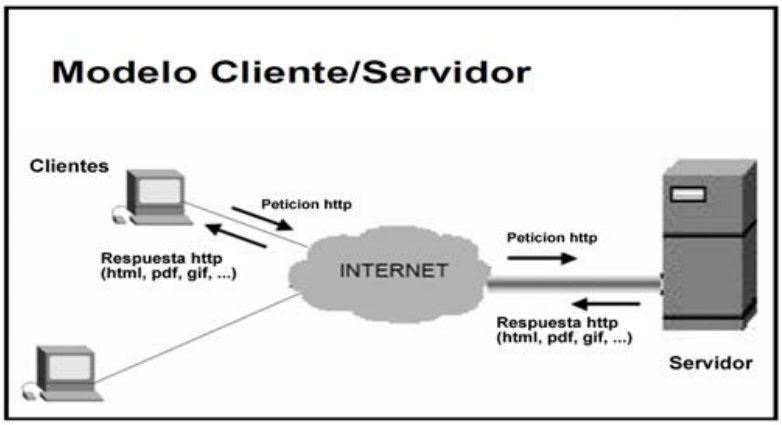


Figura 9. Arquitectura Cliente-Servidor.
Tomado de. Carrera, 2008.

La arquitectura cliente-servidor requiere una interfaz de usuario que se ejecuta en un ordenador, la cual envía solicitudes a un servidor web para ejecutar diferentes funciones.

1.7.4. Lenguajes de Programación

1.7.4.1. Lenguaje de programación PHP

“PHP (que significa Hypertext Pre-processor) es un lenguaje de programación de código abierto adecuado para el desarrollo de aplicaciones web, el cual puede ser incluido en lenguaje HTML” (PHP Group, 2014a).

PHP puede ser utilizado en sistemas operativos principales, como son: Microsoft, Linux, Windows, Mac OS, entre otros. En la actualidad PHP admite la mayoría de servidores web. (PHP Group, 2014b).

1.7.4.2. Lenguaje de programación Java

“Java es un lenguaje de programación orientado a objetos, su primera versión fue la JDK 1.0 lanzada en 1996. Hoy en día, es uno de los lenguajes más utilizados en todo el mundo” (Aprender a programar, 2014).

Como todos los tipos de lenguaje de programación tiene su propia estructura y sus propias reglas de sintaxis. Este lenguaje se deriva del lenguaje C, por lo que sus reglas de sintaxis son muy similares. La estructura de Java inicia con *paquetes*. Dentro de los paquetes se ubican las clases y dentro de las clases las variables, los métodos y las constantes. (Java, s.f.).

1.7.4.3. Lenguaje de programación Java script

JavaScript, es un lenguaje de programación desarrollado por Netscape Inc., el cual no forma parte de la plataforma Java y no permite la creación de aplicaciones independientes. JavaScript está incluido en documentos de tipo HTML y proporciona interactividad en páginas web que no se consigue con HTML simple (Java, s.f.).

1.7.4.4. Lenguaje de Programación C#

Fue creado por Anders Hejlsberg. C# es un lenguaje de programación orientado a objetos. Con este lenguaje se pretendió incorporar las ventajas de Java. Algunas de las características del lenguaje de programación C# son:

- Sincroniza la flexibilidad del C++ y la productividad de Visual Basic.
- Permite el ahorro de tiempo en la programación debido a su completa y muy bien diseñada librería de clases. (La revista informática, 2006).

2. Capítulo II: Análisis del Esquema Actual de Gestión de Parqueaderos en la Universidad de las Américas

Con el fin de determinar el esquema actual de gestión de parqueaderos en la UDLA, se ha procedido a realizar un diagnóstico exhaustivo de los mismos; dicho análisis se basó principalmente en el uso y/o aplicación de tres técnicas de investigación (observación, entrevista y encuesta). A continuación de detalla la información obtenida en cada una de las técnicas antes mencionadas:

2.1. Investigación de campo

Como técnicas de recopilación de información se utilizó tanto la observación como la entrevista y la encuesta.

2.1.1. Observación

La observación directa se aplicó para la investigación en sitio del problema, es decir los parqueaderos de la UDLA. La observación indirecta, en cambio se aplicó para la investigación documental o bibliográfica.

En la observación directa se pudo apreciar que muchos estudiantes al momento de buscar parqueadero deben dar varias vueltas para poder encontrarlo, perdiendo así tiempo y gasolina.

Además de que cuando el parqueadero está completamente lleno, se debe esperar la confirmación de una persona (trabajador), indicando que ha salido un vehículo para así permitir el ingreso de un nuevo vehículo; esto hace que se pierda tiempo.

Otra de las cosas que se pudo observar fue que no se lleva un registro por cada parqueadero, para saber cuándo ha sido ocupado o desocupado; al igual que un porcentaje de uso del estacionamiento en general por hora de servicio.

2.1.2. Entrevista a Jorge Dacier Alzate, Administrador de Parqueaderos UDLA

Se aplicó un cuestionario estructurado al responsable de la gestión de los parqueaderos en la UDLA. El formato de dicho cuestionario se adjunta en el Anexo 1.

A continuación, se muestran los resultados obtenidos de la entrevista como tal.

1. **En términos generales, ¿Cuál es el proceso actual a seguir para el uso de los parqueaderos en la UDLA?**
   Se tienen dos tipos de parqueaderos, uno para Administrativos y Docentes y otro para Estudiantes y visitantes. Mismos que en todas las sedes están sistematizados los que funcionan con tarjetas inteligentes que registran: Placas, Fotos de varios ángulos y de conductor, hora de ingreso entre otros.
2. **¿Quién está permitido usar el parqueadero?**
   El de Administrativos y Docentes acorde con la política de la Universidad es asignado a las personas que autoriza un Comité debidamente conformado sin que ello contenga un costo, teniendo restricciones de acuerdo a la capacidad del mismo. El de Estudiantes es tarifado para todos los usuarios en horarios de clases y los valores varían de acuerdo a la permanencia de cada alumno.
3. **¿Cuántos vehículos ingresan en promedio al día?**
   Sede Granados 800 aproximadamente entre docentes, funcionarios y estudiantes. Sede Queri 1200 aproximadamente entre docentes, funcionarios y estudiante. Sede UDLApark 400 aproximadamente entre docentes, funcionarios y estudiantes.
4. **¿Cuál es la capacidad máxima del parqueadero?**
   En Sede Granados 350 aproximadamente. En Sede Queri 330 aproximadamente. En Sede UDLApark 600 aproximadamente con la apertura de Subsuelos. Cabe recalcar que estos son compartidos entre estudiantes, docentes y funcionarios.

5. ***Actualmente, ¿Cuál es la manera y/o forma utilizada para el control de la entrada/salida de vehículos?***

   Para el control de entrada/salida de vehículos se dispone de un sistema sistematizados con tarjetas inteligentes y lectores correspondientes.

   Cada usuario que ingresa con su vehículo recibe una tarjeta inteligente, la cual es identificada para determinar la hora de entrada del vehículo. Esta tarjeta se encuentra en estado bloqueado hasta que se realice el pago, una vez realizado el pago correspondiente la tarjeta inteligente pasará a tener un estado de desbloqueado, para que al momento de ser leída por el lector de tarjetas en la salida, se permita la salida del vehículo.

6. **A su criterio ¿Cuáles son los subprocesos importantes a considerar para la realización del prototipo a proponer?**

Se podría considerar los siguientes subprocesos:

- *Control de espacios disponibles.*
- *Control de registros por parqueadero.*
- *Control de porcentajes de uso general del estacionamiento por horas.*

7. **Describa brevemente dichos subprocesos**

   - ***Control de espacios disponibles.***

     Controlar los espacios disponibles para que en horas pico se pueda permitir el ingreso de vehículos con mayor agilidad.

   - ***Control de registros por parqueadero.***

     Con registros por parqueadero, se podrá tener un control más detallado de las horas en las que cada parqueadero fue ocupado o desocupado, y el tiempo en el que estuvo siendo utilizado por el usuario.

   - ***Control de porcentajes de uso general del estacionamiento por horas.***

     Con este porcentaje se podrá saber a qué horas del día el parqueadero se encuentra totalmente lleno, o totalmente vacío;

de acuerdo al día. Y así buscar soluciones alternas para brindar un mejor servicio a los clientes.

8. **¿Cree conveniente implementar un sistema de gestión de parqueaderos aplicando tecnología móvil?**
   Es algo innovador y creo que sí.

2.1.3. Encuesta a Estudiantes de la Universidad de las Américas

Otro cuestionario estructurado fue aplicado a los clientes de los parqueaderos en la UDLA. En este sentido cabe mencionar que un promedio de 2400 son los autos que visitan diariamente los parqueados estudiados, los cuales incluyen Sede Granados, Queri y UDLApark. En base a esto, y usando la fórmula del muestro simple aleatorio para poblaciones finitas, se determinó la muestra a ser analizada.

$$M = \frac{Z^2 * p * q * N}{Z^2 * p * q + N * e^2} \qquad \textbf{(Ecuación 1)}$$

Donde:
M= Tamaño de la muestra
Z= Nivel de confiabilidad, 95%, 0.95/2 = 0.475, Z = 1.96
p= probabilidad de ocurrencia = 0.5
q= Probabilidad de no concurrencia= 1 - 0.5 = 0.5
N= Población= 1200
e= Error de muestreo = 0.05 (5%)

$$M = \frac{1.96^2 * 0.5 * 0.5 * 2400}{1.96^2 . * 0.5 * 0.5 + 2400 * 0.05^2}$$

$$M = 331$$

Es así que, la muestra resultante fue de 331 personas (clientes de los parqueaderos de la UDLA).

1. **¿Está satisfecho con el servicio prestado por los parqueaderos?**

Tabla 1. Porcentaje de respuestas pregunta 1.

Opciones	Respuestas	Porcentaje
Si	90	27,3%
No	241	72,7%
Total	331	100,0%

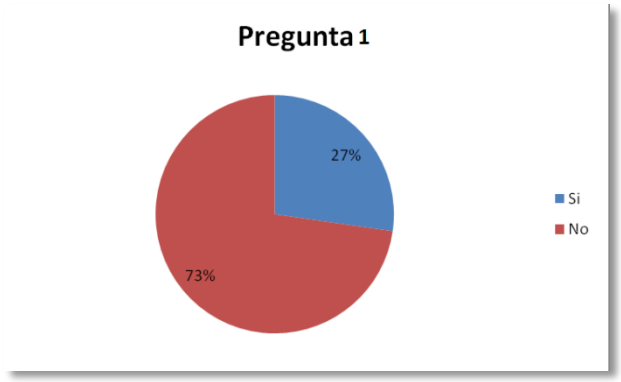


Figura 10. Gráfica porcentaje de respuestas pregunta 1.

Del total de los encuestados, el 73% No está satisfecho con el servicio prestado en los parqueaderos, mientras que tan solo un 27% si lo está.

2. ¿Qué es lo que más le disgusta de los parqueaderos actuales?

Tabla 2. Porcentaje de respuestas pregunta 2.

Opciones	Respuestas	Porcentaje
Servicio	38	11,4%
Infraestructura	38	11,4%
Organización	75	22,7%
Disponibilidad	143	43,2%
Otro	38	11,4%
Total	331	100,0%

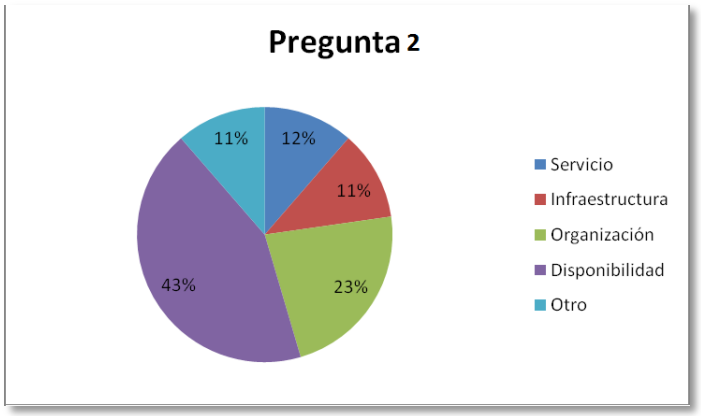


Figura 11. Gráfica porcentaje de respuestas pregunta 2.

De las respuestas obtenidas, se puede observar que al 43% de las personas encuestadas les disgusta la Disponibilidad en dichos parqueaderos, al 23% le disgusta la Organización, al 12% el Servicio, mientras que a un 11% le disgusta la Infraestructura, a otro 11% le disgusta algún otro aspecto.

3. ¿Cuánto paga mensualmente por el servicio de parqueadero?

Tabla 3. Porcentaje de respuestas pregunta 3.

Opciones	Respuestas	Porcentaje
Menos de $40	128	38,6%
Entre $40 y $60	158	47,7%
Más de $60	45	13,6%
Total	331	100,0%

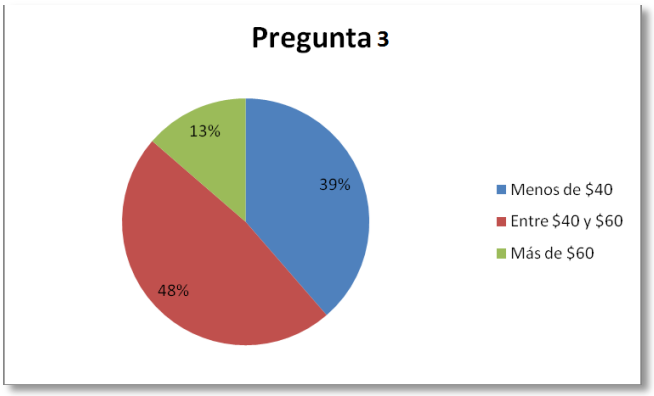


Figura 12. Gráfica porcentaje de respuestas pregunta 3.

En relación al pago mensual, el mayor porcentaje -48%- gasta entre $40 y $60 dólares, le sigue con el 39%, menos de $40 y por último se ubica más de $60 con el 13%.

4. ¿Con qué frecuencia utiliza el parqueadero?

Tabla 4. Porcentaje de respuestas pregunta 4.

Opciones	Respuestas	Porcentaje
1 vez a la semana	68	20,5%
Entre 2 y 3 veces a la semana	75	22,7%
Entre 4 y 5 veces a la semana	150	45,5%
Entre 6 y 7 veces a la semana	38	11,4%
Total	331	100,0%

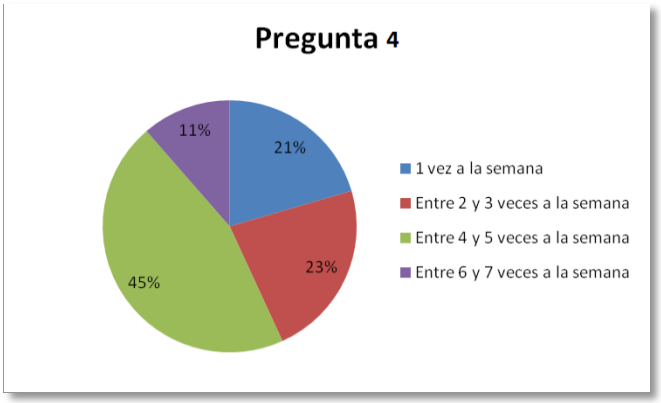


Figura 13. Gráfica porcentaje de respuestas pregunta 4.

Del 100% de los encuestados, el 45% utiliza los parqueaderos entre 4 y 5 veces por semana, el 23% lo hace entre 2 y 3 veces, el 21% 1 vez a la semana, y el 11% entre 6 y 7 veces a la semana.

5. ¿Qué tiempo le toma el estacionar su vehículo?

Tabla 5. Porcentaje de respuestas pregunta 5.

Opciones	Respuestas	Porcentaje
Menos de 5 minutos	38	11,4%
Entre 5 y 10 minutos	98	29,5%
Más de 10 minutos	196	59,1%
Total	331	100,0%

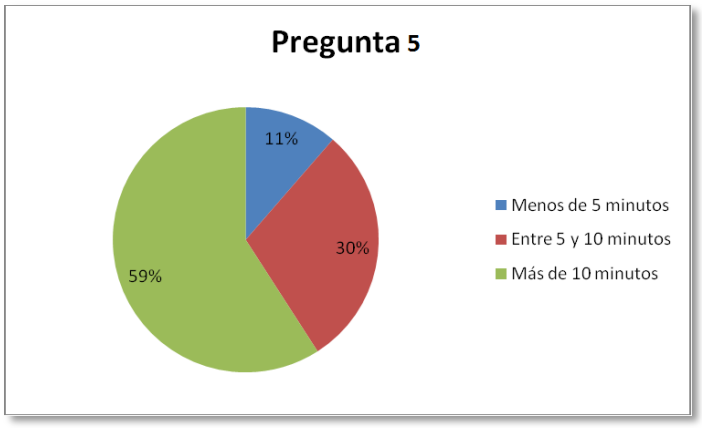


Figura 14. Gráfica porcentaje de respuestas pregunta 5.

Al 59% de los encuestados le toma más de 10 minutos estacionar su vehículo, al 30% en cambio le toma entre 5 y 10 minutos, mientras que al 11% le toma menos de 5 minutos.

6. **¿Estaría dispuesto a hacer uso de un parqueadero automatizado que ofrezca seguridad, rapidez y exactitud?**

Tabla 6. Porcentaje de respuestas pregunta 6.

Opciones	Respuestas	Porcentaje
Si	316	95,5%
No	15	4,5%
Total	331	100,0%

Figura 15. Gráfica porcentaje de respuestas pregunta 6.

Un 95% estaría dispuesto a hacer uso de un parqueadero automatizado que ofrezca seguridad, rapidez y exactitud; el 5% restante no está dispuesto a hacerlo.

7. **¿Le interesaría usar una "Aplicación móvil" que le permita conocer la disponibilidad de espacios disponibles en el parqueadero de la UDLA?**

.

Tabla 7. Porcentaje de respuestas pregunta 7.

Opciones	Respuestas	Porcentaje
Si	323	97,7%
No	8	2,3%
Total	331	100,0%

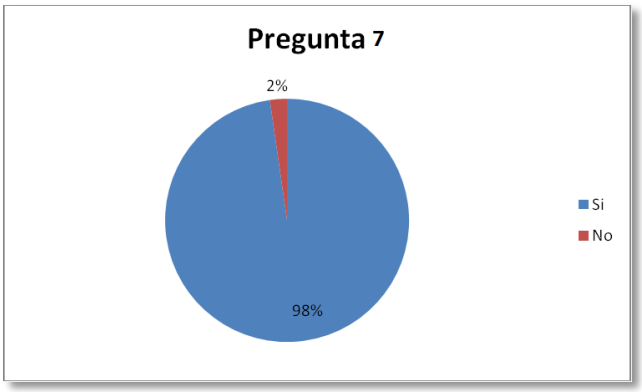


Figura 16. Gráfica porcentaje de respuestas pregunta 7.

Del total de las personas encuestadas, al 98% le interesaría usar una "Aplicación móvil" que le permita conocer la disponibilidad de espacios disponibles en el parqueadero de la UDLA, al 2% restante no le interesaría.

2.1.4. Análisis de resultados

De la investigación de campo aplicada, se puede concluir que existe un grado de insatisfacción referente al servicio actualmente prestado en los parqueaderos de la UDLA, asimismo se puede resaltar que existe gran

predisposición para el uso de un parqueadero automatizado y específicamente de un aplicación móvil que aporte sustancialmente a su gestión.

2.2. Procesos actuales

2.2.1. Sistema de ingreso

El proceso actualmente ejecutado para el ingreso de los vehículos a los parqueaderos de la UDLA se muestra en la Figura 17.

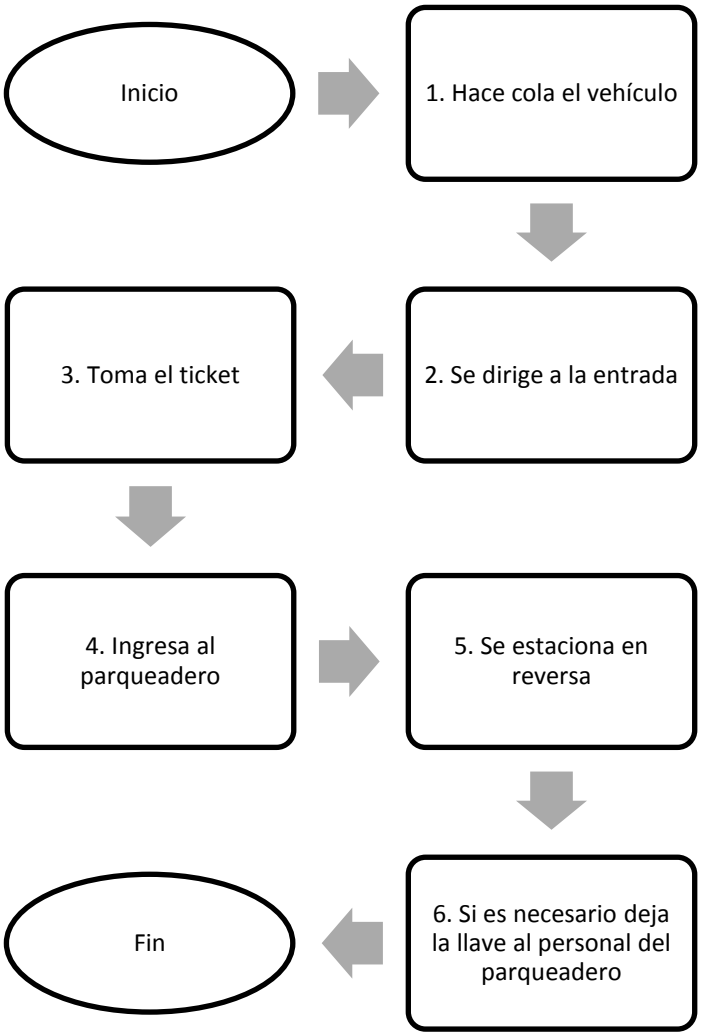


Figura 17. Sistema de ingreso.

2.2.2. Sistema de cobro

Para el cobro, el proceso actualmente ejecutado en los parqueaderos de la UDLA se muestra en la Figura 18.

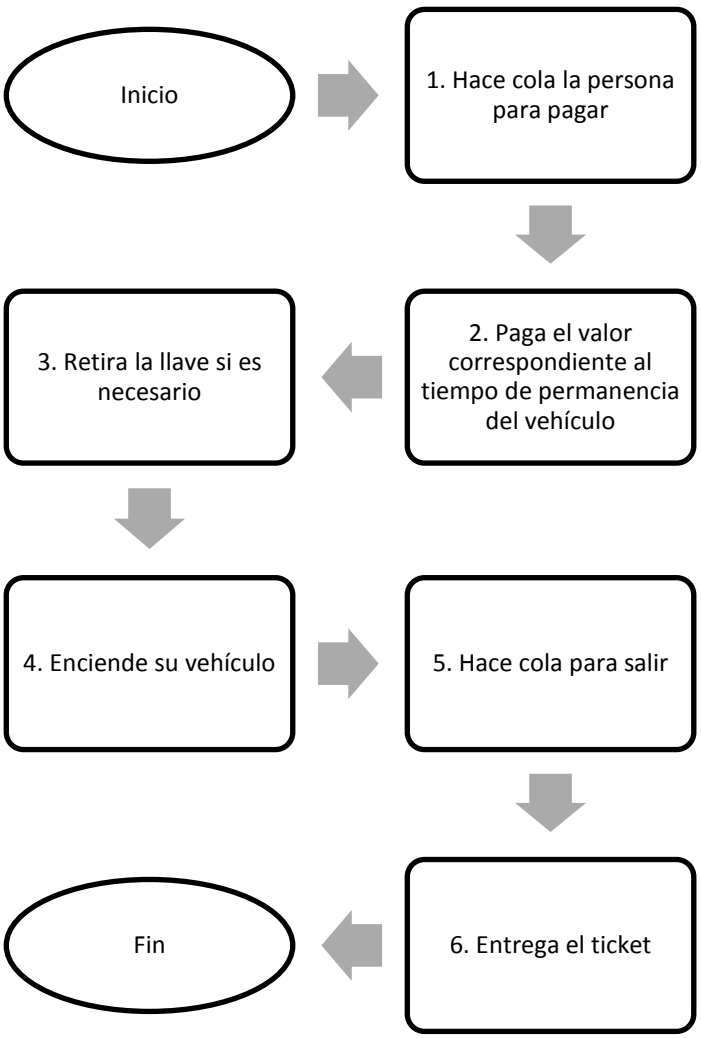


Figura 18. Sistema de cobro.

Cabe mencionar que para el cobro en la entrada se emite una tarjeta plástica, al momento de pagar se verifica la información que contiene, específicamente la hora de entrada, posteriormente luego de que el usuario para el valor correspondiente, esa tarjeta se desbloquea para que al momento de salir, y ser insertada, abra la barra para que salga el vehículo.

2.2.3. Sistema de control de congestión vehicular

El proceso ejecutado para el control de los vehículos a los parqueaderos de la UDLA se muestra en la Figura 19:

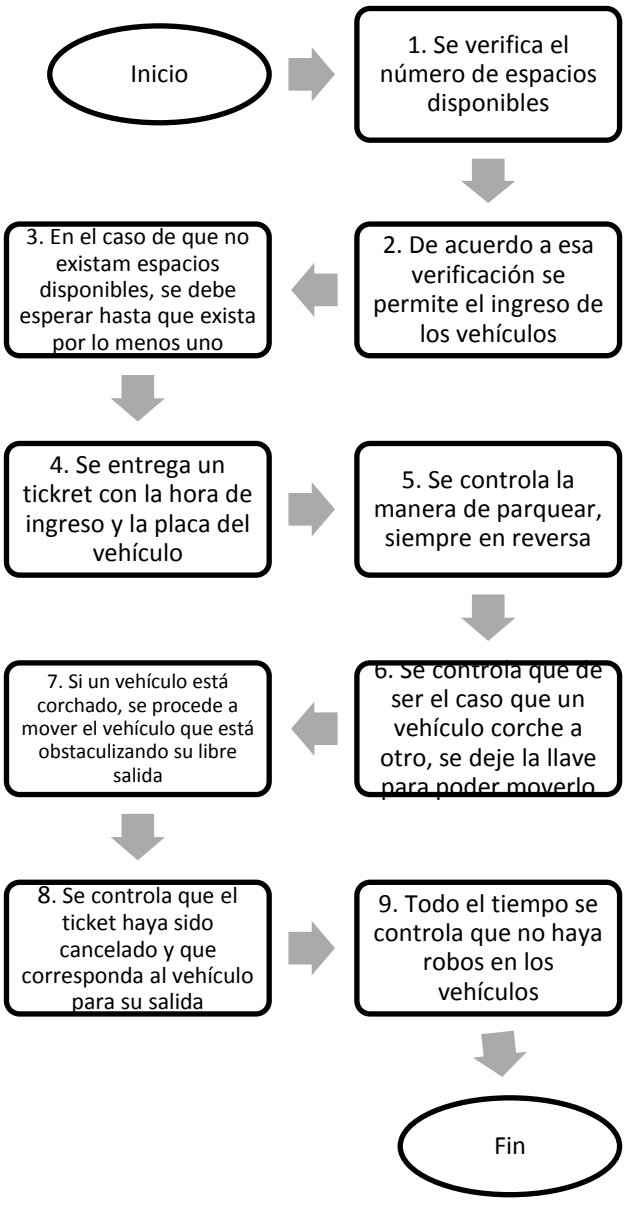


Figura 19. Sistema de control.

Se debe resaltar que la persona que cobra es una cajera, los espacios se verifican de manera visual en el parqueadero Queri, en el parqueadero de la Sede Granados tienen un sistema de luces, el cual indica con color rojo y verde los espacios ocupados y disponibles según corresponda. La seguridad se controla con cámaras e iluminación, además el personal se encuentra en constante vigila.

3. Capítulo III: Desarrollo del Hardware.

3. 1. Descripción de elementos y componentes utilizados.

El sistema se implementará en un prototipo de una maqueta a escala del parqueadero de estudiantes de la sede Queri de la Universidad de las Américas, en la cual se colocarán 10 sensores con sus respectivos Leds indicadores de estado, que simularán a 10 estacionamientos o muestras. Algunas de las razones por las cuales no se realiza la implementación en sitio son el tiempo que conllevaría realizarlo, el dinero con el que se debería contar, y los permisos necesarios. Cabe recalcar que se desarrollará el sistema de tal modo que cuando se realice la implementación en sitio presente la misma lógica.

Sensor Infrarrojo TCRT5000.

Este sensor permitirá detectar la presencia de un vehículo cuando está estacionado, debido a que es un sensor óptico reflectivo conformado por un emisor infrarrojo y un fototransistor. El sensor se muestra en la Figura 20.

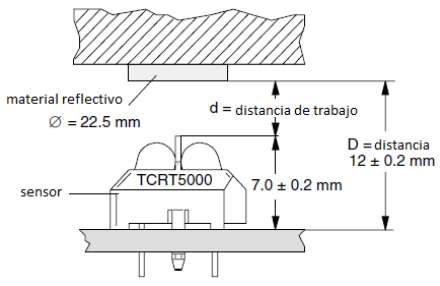


Figura 21. Funcionamiento Sensor
Tomado de: Datasheet, 2015.

Figura 20. Sensor Infrarrojo.
Tomado de: Nextia F., 2015.

Funcionamiento:

Como se muestra en la Figura 22, el emisor infrarrojo está emitiendo señales de luz, la cual necesita ser reflejada para que el fototransistor la detecte, mientras no exista un vehículo que produzca el reflejo de la luz, el fototransistor no detectará ninguna señal.

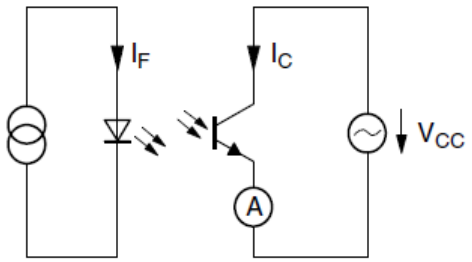


Figura 22. Diagrama Circuito TCRT5000.
Tomado de: Datasheet TCRT5000, 2015.

Este tipo de sensor fue seleccionado debido a su tamaño ya que el espacio en la maqueta es limitado. Dispone de un encapsulado que bloquea la luz visible y cuenta con 2 sujetadores en forma de clip para su sencillo montaje.

Conexión:

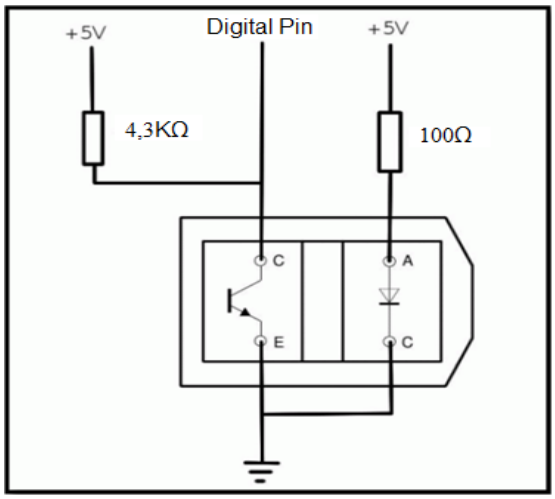


Figura 23. Diagrama circuito TCRT5000 con conexiones.
Tomado de Datasheet TCRT5000, 2015.

Como se muestra en la Figura 23, para conectarlo se utiliza una fuente de alimentación de 5V con una resistencia de 100Ω para el emisor y una resistencia de 4,3KΩ para el fototransistor y el pin de control va conectado al colector del fototransistor como se muestra en la figura. A continuación el detalle del cálculo de resistencias.

Detalle de cálculo de Resistencias:

Led Emisor:

El voltaje típico de acuerdo al *Datasheet* del sensor en el *Led* emisor de luz es 1.2V, con un máximo de 1.5V y la corriente de consumo de 40mA que es menor a la máxima de 60mA.

Tomando el valor típico de 1.2V. Se tienen los siguientes datos:

$Vin = 5V$; Voltaje de entrada.

$Vled = 1.2V$; Voltaje para funcionamiento de led emisor.

$I = 0{,}04\ A$; Corriente consumo led emisor.

$$R = \frac{(Vin - Vled)}{I}$$ **(Ecuación 2)**

$$R = \frac{(Vin - Vled)}{I} = 3.8/\ 0.04 = 95\ \Omega$$

Valor comercial de resistencia 100 Ω.

Fototransistor:

La Resistencia que será conectada al colector del fototransistor, viene dada por el siguiente cálculo:

Según el Datasheet del Sensor la corriente de colector es de 1mA. Se tienen los siguientes datos:

$Vin = 5V$; Voltaje de entrada.

$Veb\ (emisor - base)\ =\ 0{,}7V$; Voltaje para funcionamiento de led emisor de luz infrarroja.

$Veb\ (colector - emisor)\ \approx\ 0{,}3V$; Voltaje para funcionamiento de led emisor de luz infrarroja.

$I = 0{,}001\ A$; Corriente consumo colector.

$R = \frac{(Vin - Vce)}{I}$ **(Ecuación 3)**

$$R = \frac{(Vin - Vce)}{I} = \frac{(5 - 0.3)}{0.001} = 4700\ \Omega$$

Valor comercial de resistencia 4.3KΩ.

Especificaciones Técnicas (Resumen):

- Distancia de detección:**2 – 12 mm**
- Dimensiones (Largo x Ancho x Altura mm): **10.2 x 5.8 x 7**
- Método de sensado: **Reflectivo.**
- Voltaje LED: **1.25 V**

Arduino Mega 2560.

El Arduino Mega 2560 es un sistema embebido basado en el microcontrolador CMOS de 8bits *Atmega2560.* A este dispositivo llega la información que proviene de los sensores (estado), y a través de sus pines digitales se encenderán los Leds correspondientes para indicar el estado de cada estacionamiento; rojo para ocupado y verde para disponible.

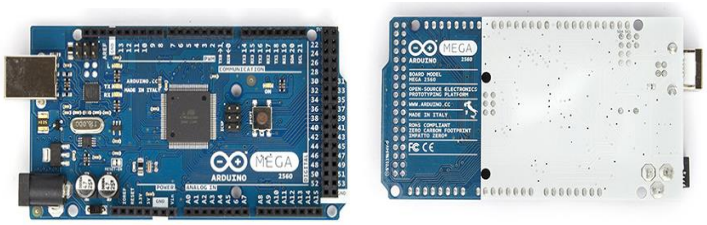


Figura 24. Arduino Mega 2560

Tomado de: Arduino, s.f.

Tipos de alimentación de Voltaje y Corriente para Arduino Mega:

El Arduino Mega 2560 puede ser alimentado de dos maneras una de ellas es a través de la conexión USB y a otra es mediante una fuente de alimentación externa. En este caso debido a que el Arduino no se encontrará cerca de una PC, deberá ser alimentado mediante una fuente externa. Cuando se utiliza los pines de 5V del Arduino como salida, el voltaje que ingresa por la fuente externa es regulado a 5V, si la fuente externa suministra menos de 7V, el pin de 5V puede suministrar menos de 5V, y si la fuente externa suministra más de 12 V, regulador puede sobrecalentarse y así dañarse. Es por eso que la fuente externa debe suministrar entre 7V a 12V.

Figura 25. Arduino Mega 2560, Alimentación Externa.
Tomado de: Arduino, s.f.

Conectando un sensor al Arduino:

Realizando la conexión del sensor infrarrojo TCRT5000 al Arduino Mega, se utilizarán los siguientes pines:

- 5V, Tierra y Pin Digital D44, en el cual se conectaría el control para determinar el estado del parqueadero de acuerdo al sensor. Este Pin será programado como de entrada recibiendo un 1 lógico o un 0 lógico por parte del sensor.

Las conexiones quedan como se muestra en la Figura 26.

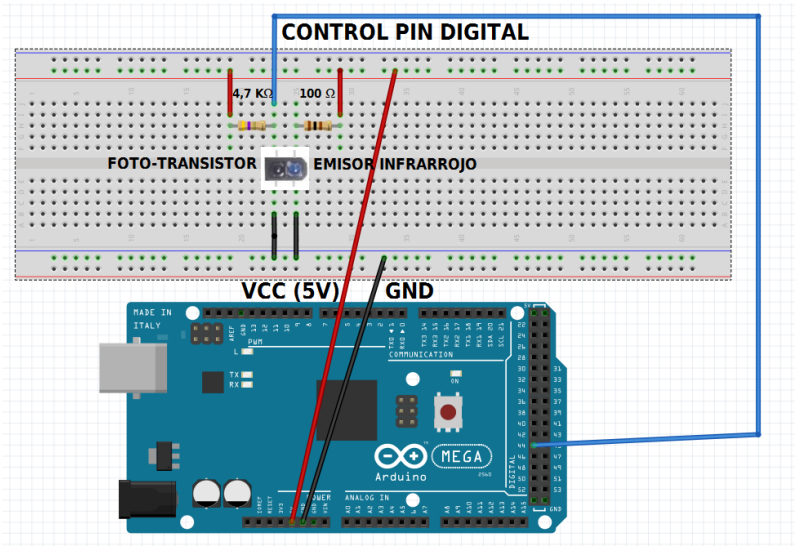


Figura 26. Conexión de un Sensor TCRT500 a Arduino Mega.

Valores Digitales:

La razón por la que se utiliza pines digitales en el control de cada sensor, es que solo existen dos estados, o bien sea 1 o 0 lógico, indicando que existe o no presencia de un vehículo. Cuando no existe un vehículo el valor de voltaje en el colector es de 4.79V lo cual corresponde a un 1 lógico. En el caso de que exista en elemento sensado a una distancia de 2.5mm el valor de voltaje en el colector es de 0.25V, que corresponde a 0 lógico.

Especificaciones Técnicas Arduino Mega 2560(Resumen):

- Posee Microcontrolador Atmega2560.
- La Tensión De Funcionamiento es de 5V.
- El Voltaje de entrada (recomendado) es de 7-12V.
- Posee 16 pines de entrada analógica.
- Posee 54 pines Digitales (de las cuales 15 proporcionan salida PWM)
- Resonador cerámico 16 MHz.
- Posee Conexión USB, conector de alimentación, botón de reinicio.

Leds

LED es una abreviatura que significa *light-emitting diode* en español ‘diodo emisor de luz’.
Para determinar la polaridad existen dos maneras:

- Por lo general la pata más larga viene a ser el ánodo (+) y la corta el cátodo (-).

- En el interior del LED, hay dos partes la plaqueta y el yunque, la diferencia entre estas es que el yunque es más grande. La plaqueta muestra el ánodo y el yunque, que muestra el cátodo, como se puede observar en la Figura 27.

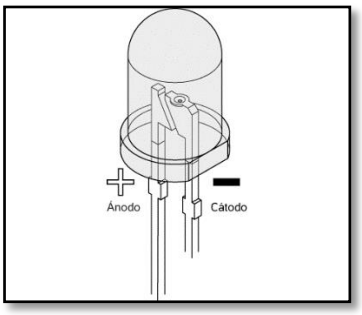


Figura 27. Led, ánodo y cátodo
Tomado de: Leds Magazine, s.f.

Detalle de cálculo de Resistencias para Leds:
Los Leds utilizados son de colores Rojo y Verde; de acuerdo al color se tienen los siguientes datos de acuerdo al fabricante:

- **Rojo:**

  $Vin = 5V$; Voltaje de entrada.

  $Vled = 1.9V$; Voltaje para funcionamiento led.

  $I = 10mA$; Corriente consumo led.

$$R = \frac{(Vin - Vled)}{I} = \frac{3.1}{0.01} = 310\Omega$$

  Valor comercial de resistencia 330 Ω.

- **Verde:**

  $Vin = 5V$; Voltaje de entrada.

  $Vled = 2.1V$; Voltaje para funcionamiento led.

  $I = 10mA$; Corriente consumo led.

$$R = \frac{(Vin - Vled)}{I} = \frac{2.9}{0.01} = 290\Omega$$

Valor comercial de resistencia 330 Ω.

Conexión de los Leds para indicar estado de un sensor:

Para la conexión de un Led se necesitan 5V de alimentación de voltaje, los cuales a través de una resistencia de 330 Ω (de acuerdo al cálculo anterior) se conectan al ánodo, el cátodo se conecta a Tierra. Añadiendo Leds al circuito previo, se puede observar que se utilizan pines digitales del Arduino como salidas para encenderlos o apagarlos según el estado del parqueadero.

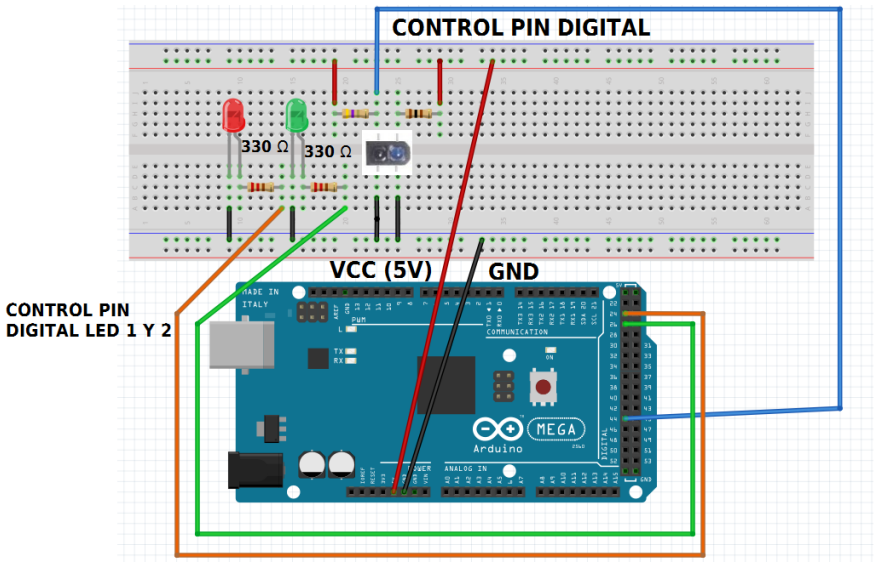


Figura 28. Conexión de un Sensor TCRT500 con sus respectivos Leds al Arduino Mega.

En la programación del Arduino se controla cuando prender los Leds correspondientes de acuerdo al valor digital del pin de control tomado en cada sensor infrarrojo del prototipo.

Shield Ethernet para Arduino Mega

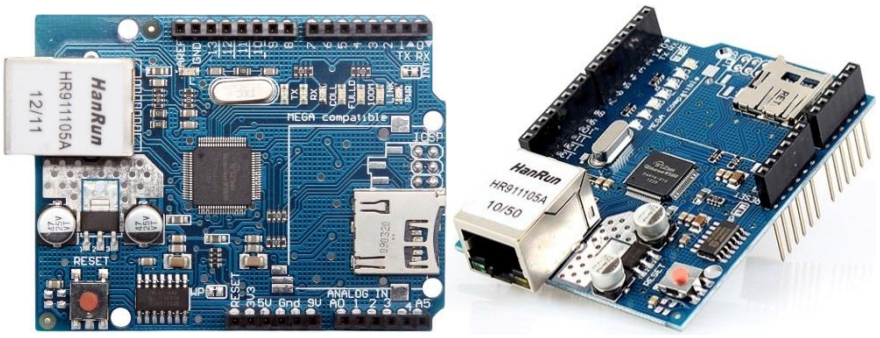


Figura 29. Shield Ethernet Rev. 2.
Tomado de: Arduino, s.f.

El Shield Ethernet para Arduino permite que este se conecte a una red local y al internet, con una IPv4 que puede ser dada mediante DHCP (Dynamic Host Configuration Protocol) o una dirección fija, además es necesaria la asignación de un MAC Address (dirección de control de acceso al medio), esta dirección viene dada en la caja del Shield adquirido. Se basa en el chip Ethernet Wiznet W5100.

Una vez conectado al Arduino utiliza la librería Ethernet la cual permite mediante programación la conexión a la red. El Arduino se comunica con este Shield mediante el integrado W5100 y con la tarjeta SD respectiva por protocolo de bus SPI a través del conector ICSP, que se muestra en la Figura 30.

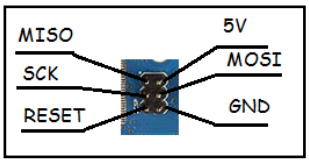


Figura 30. Componentes Conector ICSP.
Tomado de: Arduino, s.f.

Para esto el módulo se conecta a los terminales 50, 51, 52 y 53 en el Arduino Mega. Todos estos terminales no se pueden usar como pines de entrada/salida. Por lo tanto no se podrá utilizar estos pines para conectar ningún Led ni pin de control de estado de cada sensor.

Figura 31. Terminales (Pines) para comunicación por bus SPI.
Tomado de: Gammon Software Solutions, s.f.

Adicionalmente este Shield posee un conector Ethernet de estándar RJ45. Al presionar el botón de *Reset* en el Shield se realiza un reinicio del integrado W5100 y del Arduino al cual está conectado.

La alimentación del Shield se hace a través del Pin *Vin* del Arduino por lo que se recomienda tener conectada una fuente externa, ya que al momento de realizar las comunicaciones puede que la corriente que llega desde el puerto USB no sea suficiente.

Este Shield contiene varios Leds informativos:

- ON: Indica que el Arduino y el Shield están encendidos.
- LINK: Muestra la presencia de enlace de red y parpadea en el momento en que el Shield envía o recibe datos.
- 100M: Este Led muestra que existe una conexión de red de 100 Mb/s.
- RX: Titila cuando el Shield recibe datos.
- TX: Titila cuando el Shield envía datos.

Características a considerarse en el Shield Ethernet:

- Opera a 5V suministrados desde la placa de Arduino
- El controlador Ethernet es el W5100 con 16K de buffer interno. No consume memoria.
- El Shield se comunica con el microcontrolador por el bus SPI, por lo tanto para usarlo se deberá incluir la librería ***SPI.h*** la cual se encuentra en el siguiente link: http://arduino.cc/en/Reference/SPI
- Soporta hasta 4 conexiones simultaneas
- Usar la librería Ethernet para manejar el Shield, la cual se encuentra en el siguiente link: http://arduino.cc/en/Reference/Ethernet
- El Shield dispone de un lector de tarjetas micro-SD que es utilizado para guardar ficheros y servirlos sobre la red. Para ello es necesaria la librería SD, la cual se encuentra en el siguiente link: http://arduino.cc/en/Reference/SD
- Al trabajar con la SD, el pin 4 es usado como SS.

Integrado W5100:

Figura 32. Chip W5100.
Tomado de: Datasheet W5100, 2012.

Este integrado ha sido diseñado para facilitar la implementación de la conectividad a Internet sin la necesidad de un sistema operativo.

El W5100 incluye toda la estructura para poder contar con:

- Protocolo TCP/IP.
- Ethernet integrando MAC (dirección de control de acceso al medio) y PHY (Physical Layer).

Los protocolos soportados son los siguientes: TCP, UDP, IPv4, ICMP, ARP, IGMP and PPPoE. El integrado W5100 se conecta al Arduino mediante SPI. Este integrado implementa una pila TCP con todas las funciones del estándar IEEE 802.3 (Ethernet capa física y de enlace de datos) dentro del chip; esto hace que el chip Wiznet W5100 sea buena opción para integrar el sistema embebido en internet. El Wiznet W5100 actuará como un dispositivo esclavo SPI controlado por microcontrolador ATMega328 como el SPI Maestro.

Una vez explicado el funcionamiento del Shield, la conexión al Arduino Mega utilizado se realiza de la siguiente manera, como se muestra en la Figura 33.

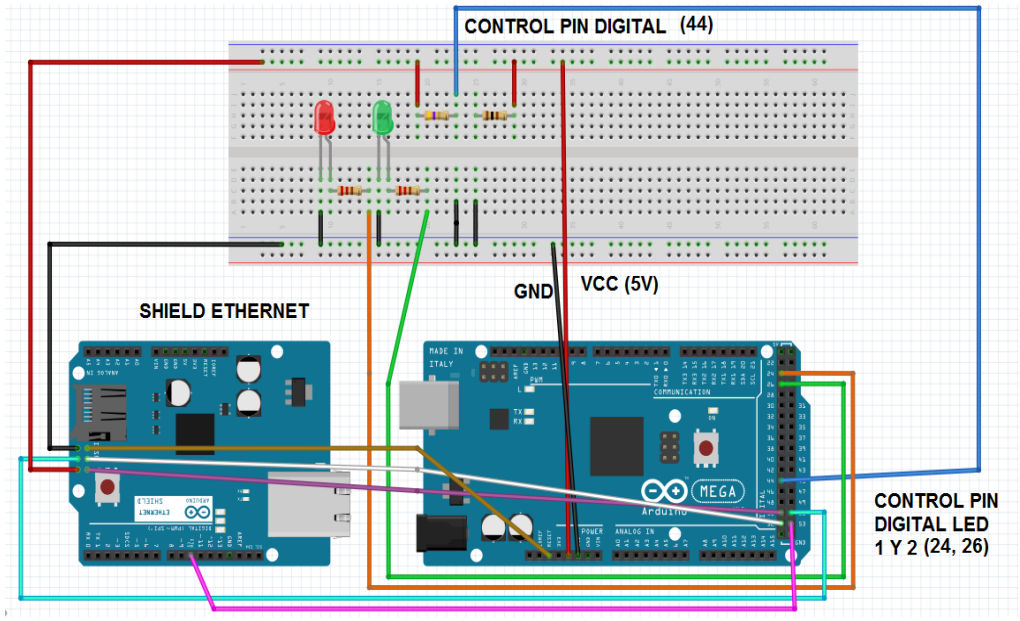


Figura 33. Conexión Shield Ethernet a Arduino Mega.

La conexión para los diez sensores y Leds indicadores se realiza como se ha detallado anteriormente para un solo parqueadero.

En el Anexo 3 se muestra el diseño de la PCB para 10 Sensores y 20 Leds, que serán conectados al Arduino Mega.

La distribución se muestra en la Tabla 8.

Tabla 8. Distribución de Terminales de Arduino Mega para Leds.

Led	Pin Digital Arduino
1	24
2	25
3	26
4	27
5	28
6	29
7	30
8	31
9	32
10	33
11	34
12	35
13	36
14	37
15	38
16	39
17	40
18	41
19	42
20	43

Tabla 9. Distribución de Terminales de Arduino Mega para Sensores.

Sensor	Pin Digital Arduino
1	44
2	45
3	46
4	47
5	48
6	48
7	2
8	3
9	5
10	6

Tabla 10. Distribución de Terminales de Arduino Mega para Leds y Sensores.

Sensor (Pin Digital)	Led (Pin Digital)
1(44)	13(36)
	15(38)
2(45)	1(24)
	3(26)
3(46)	5(28)
	7(30)
4(47)	6(29)
	8(31)
5(48)	10(33)

	12(35)
6(49)	9(32)
	11(34)
7(2)	2(25)
	4(27)
8(3)	17(40)
	19(42)
9(5)	14(37)
	16(39)
10(6)	20(43)
	18(41)

3. 2. Fuente de Alimentación

En el circuito presentado el consumo de corriente eléctrica por elemento:

- La corriente que consume cada Led es de ~10mA.
- Cada sensor consume alrededor de 20mA.
- El Shield Ethernet para Arduino tiene un consumo de ~150mA.

En la Tabla 11 se muestra el cálculo del consumo total de corriente del sistema.

Tabla 11. Cálculo consumo Total de Corriente del Sistema.

Dispositivo Electrónico	Cantidad	Consumo por Unidad	Total
Leds	10	10mA	100mA
Sensores	10	20mA	200mA
Shield Ethernet	1	150mA	150mA
		TOTAL:	**450mA**

Sumando la corriente se tiene un consumo total de alrededor de 450mA.

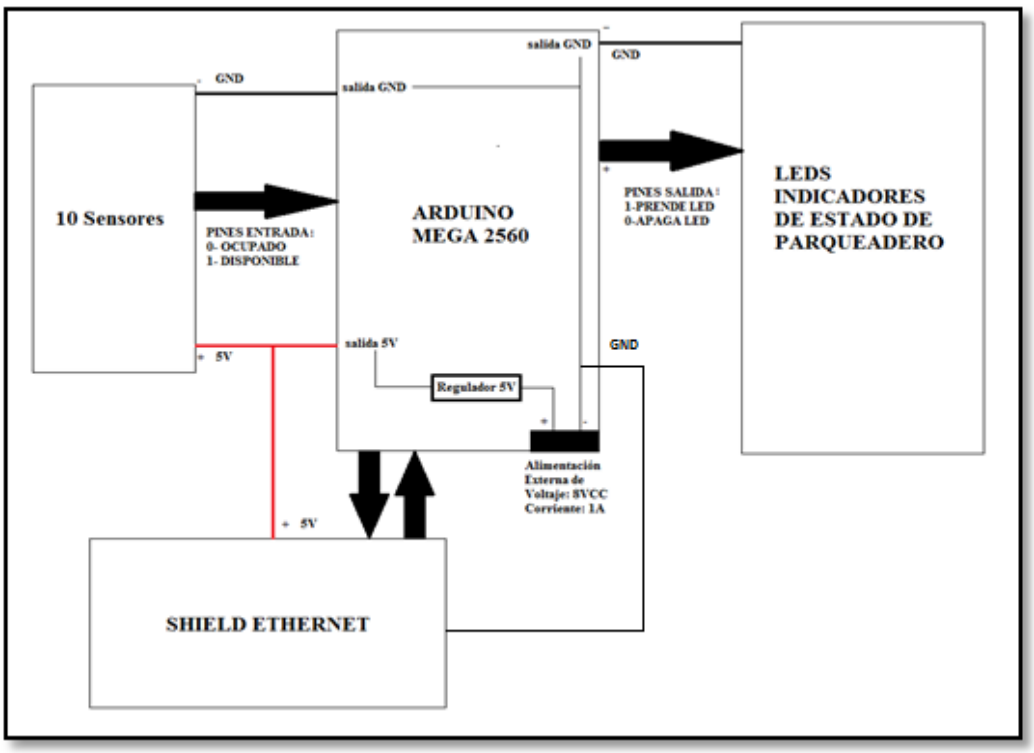


Figura 34. Diagrama electrónico del sistema

a. La comunicación entre los dispositivos electrónicos esta expresada por flechas negras.

Para poder suplir la corriente necesaria para el sistema se utiliza una fuente externa para el Arduino Mega de 8V y 1A, el Arduino con su regulador, regula el voltaje a 5V y mantiene la corriente de 1A. El 1 A de la Fuente es suficiente por lo que no se necesita el diseño de una fuente de más Amperaje.

Mientras más sensores se utilicen más Amperaje se necesitará y para esto es necesario el diseño de una Fuente de Alimentación capaz de suplir corriente a todos los elementos implicados. Cabe mencionar que en los Pines de Salida para controlar el prendido y apagado de Leds según el Datasheet del Arduino Mega, pueden suplir hasta 50mA, lo cual es suficiente en comparación con los 10mA que necesita cada Led.

3. 3. Diagrama del circuito completo:

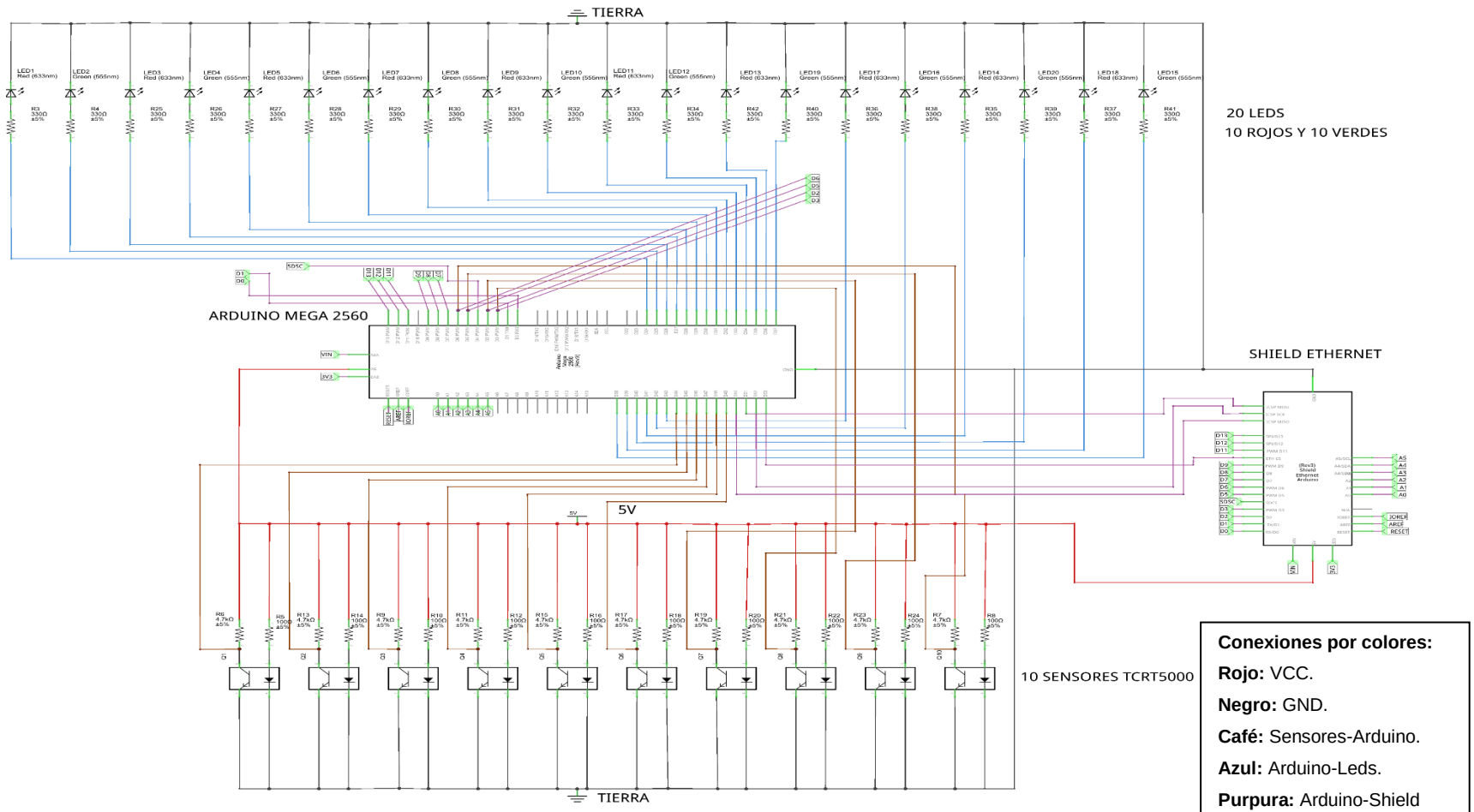


Figura 35. Diagrama del circuito completo del sistema.

Como se observa en la Figura 35, se tienen 10 sensores para 10 estacionamientos y 20 Leds que indicarán el estado de lo dichos parqueaderos (10 rojos y 10 verdes), los cuales son conectados al Arduino Mega. Finalmente se puede observar las conexiones del Shield Ethernet con el Arduino.

3. 4. Ensamblaje de sistema electrónico:

La maqueta en la que se implementará el sistema es un modelo a escala 150:1 del parqueadero de estudiantes de la Sede Query, como se muestra en la Figura 36.

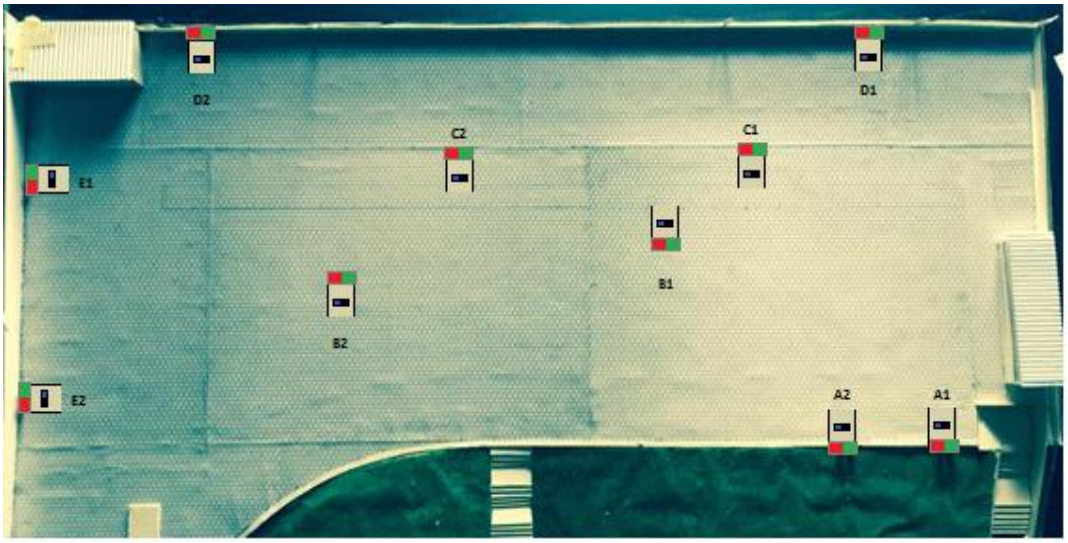


Figura 36. Maqueta Parqueaderos Estudiantes Sede Queri.

- Cada Sensor es ubicado en las posiciones de los estacionamientos señalados.
- Los Leds, rojo y verde, son ubicados en la parte inicial de cada parqueadero.

Ahora se coloca la placa (PCB) del sistema, junto a los cables con los sensores y Leds en la maqueta. Adicionalmente se colocan cables para conectar pines de 5V, Tierra y los pines de control al Arduino Mega; al cual se encuentra conectado el Shield Ethernet. A continuación como quedaría el modelo final:

Parte Superior:

Figura 37. Maqueta parte superior con ensamblaje electrónico (Sensores, Leds).

a. En total son 10 sensores cada uno con dos leds (rojo y verde).

Parte Inferior:

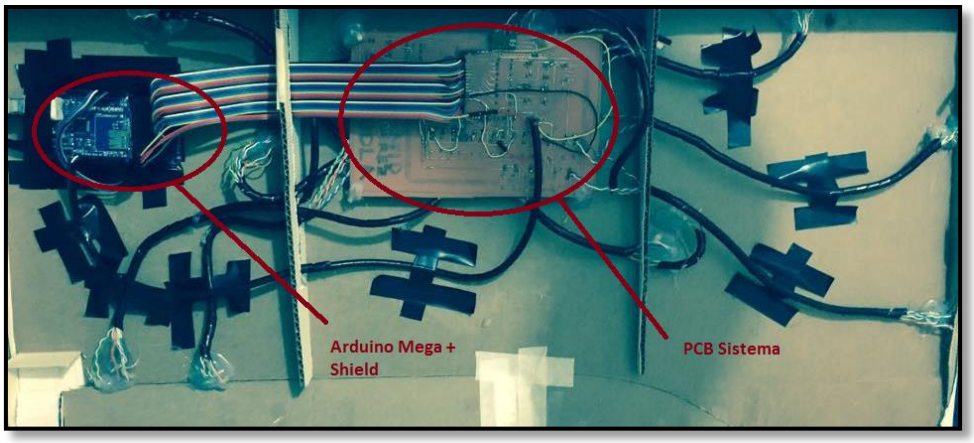


Figura 38. Maqueta parte inferior con ensamblaje electrónico (PCB, Arduino Mega, Shield Ethernet).

4. Capítulo IV: Desarrollo del Software.

4.1. Desarrollo base de datos MYSQL

Para el desarrollo de la Base de Datos, se utilizará la Sistema Gestor de Base de Datos MYSQL. Las Tablas a crear son las siguientes:

- Parqueadero
- Estado
- Registro
- Administrador

El diagrama Entidad-Relación se presenta en la Figura 39.

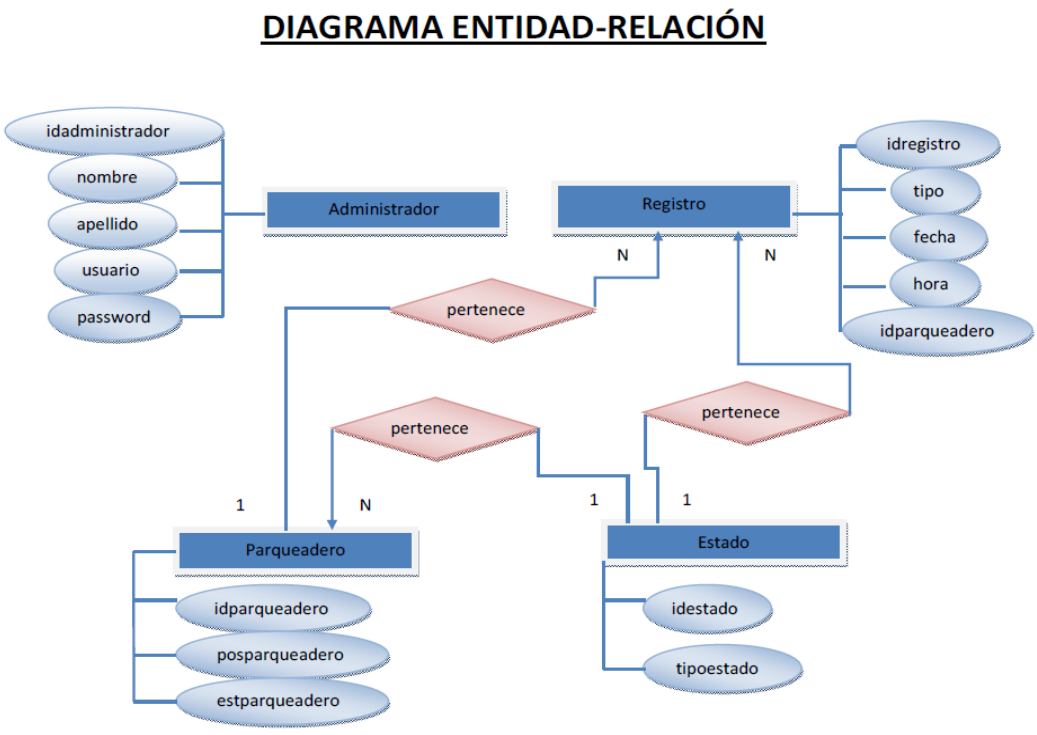


Figura 39. Diagrama Entidad-Relación.

En MySQL, el diagrama relacional de las *Tablas* creadas, se ilustra en la Figura 40.

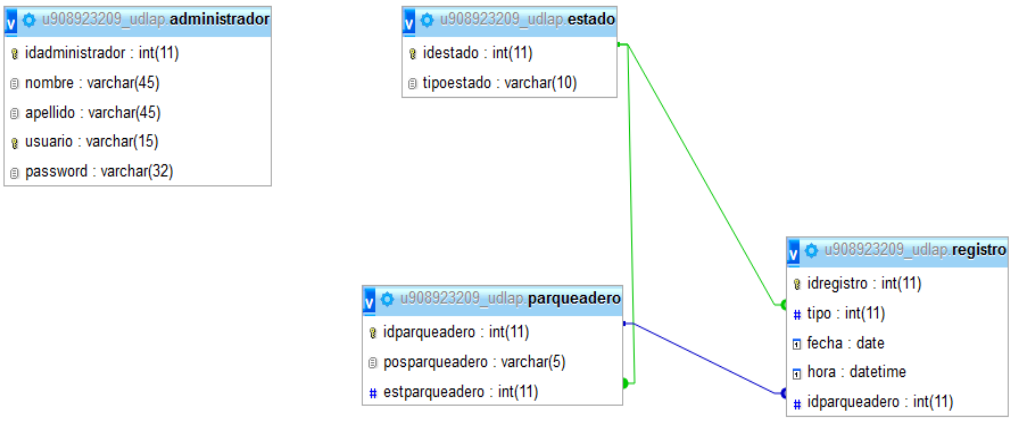


Figura 40. Diagrama Relacional MySQL.

Como se puede observar en la Figura 40 se tienen 3 relaciones, las cuales son:

- 1 Parqueadero pertenece a varios Registros.
- 1 Estado pertenece a varios Registros.
- 1 Estado pertenece a varios Parqueaderos.

Los campos de la Tabla Administrador:

- idadministrador, int (11).
- nombre, varchar (45).
- apellido, varchar (45).
- usuario, varchar (15).
- password, varchar (32).

Los campos de la Tabla Registro:

- idregistro, int (11).
- tipo, int (11).
- fecha, date.

- hora, datetime.
- idparqueadero, int (11).

Los campos de la Tabla Parqueadero:

- idparqueadero, int (11).
- posparqueadero, varchar (5).
- estparqueadero, int (11).

Los campos de la Tabla Estado:

- idestado, int (11).
- tipoestado, varchar (10).

La base de datos se llamará ***parqueaderosudla.***

4.1. Aplicación web

4.1.1. Introducción

El lenguaje de programación que se utilizará para el desarrollo de la Aplicación Web es PHP versión 5.5.23.

PHP (acrónimo recursivo de PHP: Hypertext Pre-processor), este lenguaje de programación es de lenguaje abierto, usado especialmente para el desarrollo de aplicaciones web y que puede ser incrustado en medio del código HTML. Lo que distingue a PHP de JavaScript es que el código se ejecuta en el lado del servidor; para después generar HTML y así enviarlo al cliente, para que este lo pueda visualizar en el navegador.

Adicionalmente se utilizara el Servidor de HTTP Apache versión 2.4.

Apache es un servidor web HTTP de código abierto, que implementa el protocolo HTTP/1.1. Con esta aplicación se puede montar un Servidor Web en cualquier equipo, con alta compatibilidad con sistemas operativos. Debido a

que soporta PHP como lenguaje de programación se utilizó este Servidor HTTP.

En la aplicación Web se incluirá el servicio que va a permitir consultar a la Base de Datos y así mostrar los datos de la tabla Parqueadero a través de un Objeto JSON, con el fin de obtener el estado de los parqueaderos.

Finalmente también se incluirá algunas funciones de JavaScript, para validaciones, para obtener los datos de los estados de los parqueaderos mediante Objetos JSON, y para actualizar el estado de los parqueaderos automáticamente.

4.1.2. Desarrollo

En el desarrollo se tomará en cuenta una arquitectura en capas:

- Base de Datos
- Procesamiento de Datos
- GUI

En la capa de Base de Datos se establecerán los parámetros (hostname, usuario, password, base de datos) para establecer la conexión a la Base de Datos del Servidor y para desconectarla. Además esto permitirá el rehúso del código para no escribirlo a cada momento para establecer una nueva conexión o cerrarla. La capa de Procesamiento va a permitir procesar los datos obtenidos en las consultas MYSQL, cálculos, gráficas, etc. En la capa GUI, que viene a ser la interfaz gráfica del usuario, se mostraran los datos procesados, de acuerdo a la necesidad.

La aplicación web como se mencionó anteriormente cuenta con el servicio para obtener los estados de los parqueaderos en la Base de Datos MYSQL a través de Objetos JSON.

El objeto JSON es un formato de texto para intercambio de datos que es independiente del lenguaje de programación y que puede ser interpretado por lenguajes como C, C++, C#, Java, JavaScript, entre otros. Es por eso que JSON es una herramienta ideal para el intercambio de información.

El Objeto JSON en su estructura contiene dos partes:

- Una sección donde se coloca el nombre o valor, que en varios lenguajes de programación es conocido como objeto.

- Un listado ordenado de los valores. En la mayoría de lenguajes de programación, esto se conoce como arreglos.

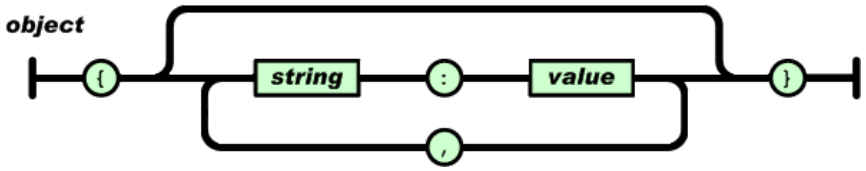


Figura 41. Estructura Objeto JSON.
Tomado de: Anieto2k, s.f.

El objeto JSON va a estar formado por un *Arreglo* de objetos *Parqueadero*, el cual es generado a partir de una consulta en MYSQL, para luego devolver el objeto como se muestra en la Figura 42.

{"parqueaderos":[{"idparqueadero":"1","posparqueadero":"A1","estparqueadero":"1"},
{"idparqueadero":"2","posparqueadero":"A2","estparqueadero":"1"},
{"idparqueadero":"3","posparqueadero":"B1","estparqueadero":"0"},
{"idparqueadero":"4","posparqueadero":"B2","estparqueadero":"0"},
{"idparqueadero":"5","posparqueadero":"C1","estparqueadero":"0"},
{"idparqueadero":"6","posparqueadero":"C2","estparqueadero":"0"},
{"idparqueadero":"7","posparqueadero":"D1","estparqueadero":"1"},
{"idparqueadero":"8","posparqueadero":"D2","estparqueadero":"0"},
{"idparqueadero":"9","posparqueadero":"E1","estparqueadero":"0"},
{"idparqueadero":"10","posparqueadero":"E2","estparqueadero":"0"}],"success":1}

Figura 42. Objeto JSON de la consulta de Tabla Parqueadero.

Por otra parte cuenta con un archivo ***update.php*** el cual va a permitir que el Arduino realice inserciones y actualizaciones a la Base de Datos mediante una petición HTTP de tipo POST. La aplicación web tiene dos finalidades:

- Observar el estado de los parqueaderos, esta opción está habilitada para cualquier persona que desee observar el estado de los estacionamientos.
- Administrar el uso de los parqueaderos, esta opción únicamente va a ser utilizada por los administradores del parqueadero, en la cual se tendrán las siguientes opciones:
    a) Observar los Registros de los parqueaderos.
    b) Crear, Actualizar, Eliminar algún parqueadero.
    c) Cambiar su contraseña, o modificar sus datos personales.
    d) Crear un nuevo Administrador.

Al momento de inicializar se presenta la siguiente *Pantalla Principal*:

Figura 43. Interfaz de la página principal del sistema web.

En la Figura 43 se puede observar las dos opciones mencionadas anteriormente.

En la primera que es Estado de Parqueaderos, se tiene la siguiente interfaz gráfica:

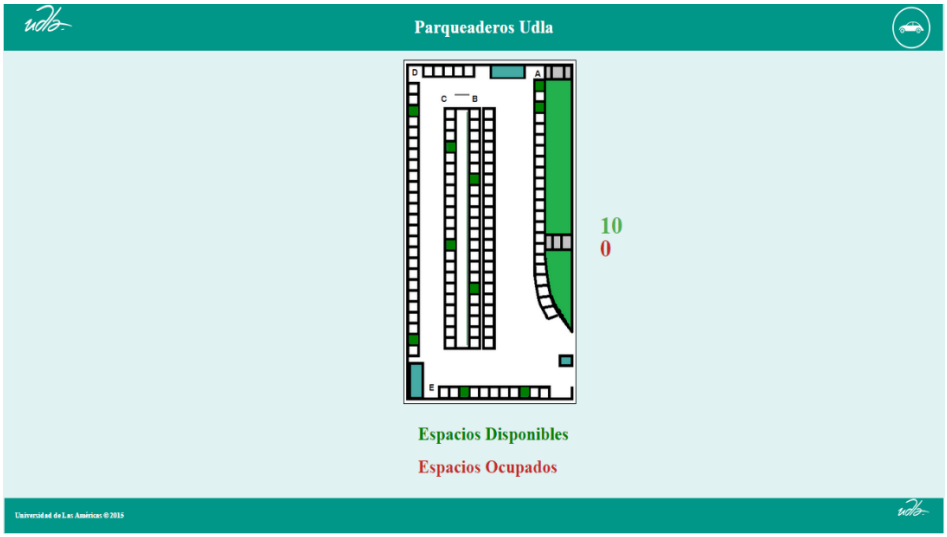


Figura 44. Interfaz de estados de parqueaderos.

En esta interfaz se puede observar el estado de los diez parqueaderos ubicados en la maqueta del prototipo, de color verde indica los estacionamientos disponibles y de color rojo los que se encuentran ocupados. Y en la parte derecha muestra el número total de espacios disponibles y espacios ocupados.

Los estados de los parqueaderos se actualizan automáticamente sin necesidad de actualizar la página, o de presionar algún botón, se actualiza en un intervalo de tiempo de 5 segundos.

Para obtener los datos de los parqueaderos se utilizará JavaScript, realizando una llamada al servicio de Objetos JSON, explicado anteriormente, mediante una petición HTTP con el método GET, para así obtener el objeto, y traducirlo mediante el método de la clase JSON llamado *parse*. Una vez interpretado el objeto, se procede a almacenar todos los estados de los parqueadero en un arreglo; para así de acuerdo al estacionamiento asignar un color de acuerdo al estado, rojo si está ocupado y verde si está disponible. Una vez obtenidos los

estados de los parqueaderos se procede a sumar todos los disponibles y todos los ocupados, para mostrar el valor total de espacios ya sea disponibles y ocupados. Finalmente se utiliza la función *setTimeout* para que esta actividad se repita cada cierto tiempo, en este caso cada 5 segundos.

Posteriormente para acceder a la segunda opción *Administración de Parqueaderos*, se debe realizar un inicio de sesión como se muestra en la Figura 45.

Figura 45. Interfaz de inicio de sesión para administradores.

En la parte superior se tendrá la opción de ir a la Página Principal.
Aquí se deberá ingresar un nombre de usuario y contraseña. El cuadro de texto para ingresar el usuario, se encuentra validado para permitir el ingreso de cualquier carácter con un tamaño máximo de 15, el cuadro de texto para la contraseña es de tipo password, con el fin de que al ingresarlo no se observe por seguridad.

Los dos cuadros de texto se encuentran validados para que no al momento de dar clic en Ingresar no puedan estar vacíos.

La contraseña ingresada es encriptada utilizando el algoritmo m50, el cual es representado por un número de 32 dígitos en hexadecimal, este tipo de encriptado es en una solo vía, ya que des-encriptarlo es prácticamente imposible.

La validación se la realiza comparando los datos ingresados con los datos de la Base de Datos. Si los datos no coinciden con los de la Base de Datos, aparecerá el mensaje, como se muestra en la Figura 46.

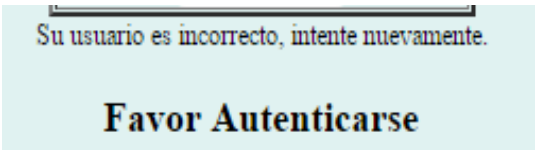


Figura 46. Mensaje en caso de ingreso erróneo de credenciales.

Caso contrario se validan los datos y se ingresará a la página donde se podrá realizar opciones de administrador, como se muestra a continuación en la Figura 47.

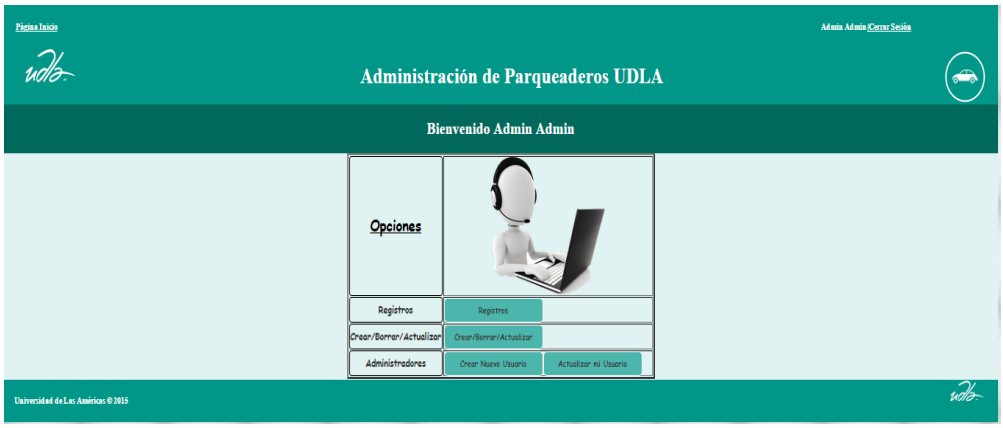


Figura 47. Interfaz de las opciones de administrador.

Como se puede observar en la parte superior se tienen las dos opciones de ir a la *Página de Inicio* y de *Cerrar Sesión*, además aparecerá el nombre y apellido del Administrador que ingresó, en este caso *Admin Admin.*

Para obtener los datos del Administrador se utiliza variables de tipo *SESSION* en donde se almacenará el nombre y apellido, a la vez de una variable que permitirá conocer si una sesión ha sido iniciada, para que cuando un Administrador elija en la pantalla de Inicio la opción *Administración de Parqueaderos* ya no se le solicite el Inicio de Sesión.

En el caso de cerrar la sesión todas las variables de tipo *SESSION* son eliminadas. Y si alguien desea acceder a las tareas de Administrador deberá iniciar sesión nuevamente. Entre las opciones que tiene un administrador se tienen las siguientes:

- Registros.
- Crear, Borrar, Actualizar.
- Crear nuevo usuario y Actualizar mi Usuario

Si se da clic en la opción Registros, se tiene la siguiente interfaz, como se muestra en la Figura 48.

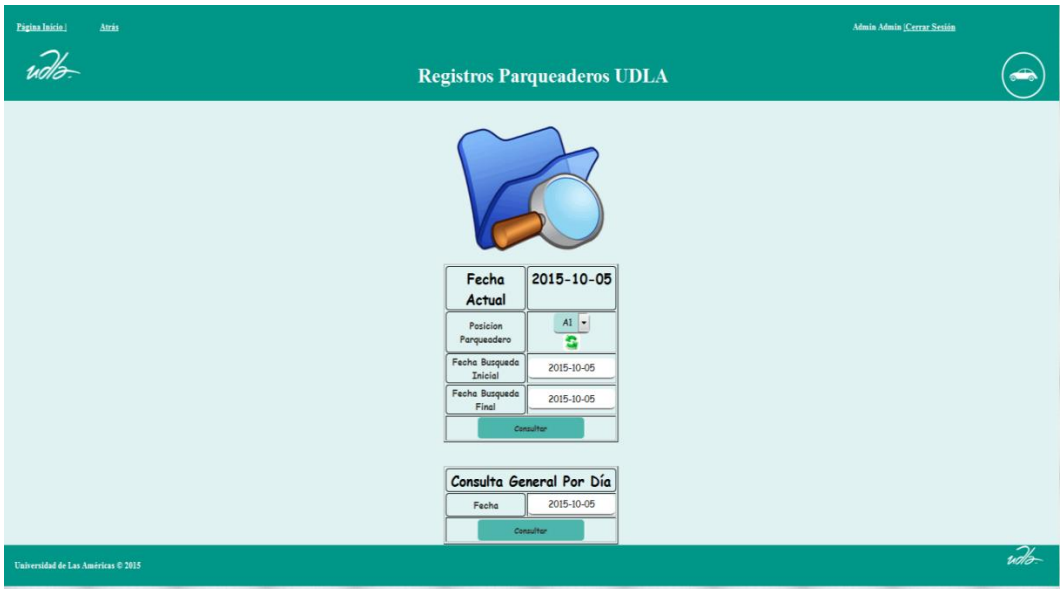


Figura 48. Interfaz de las opciones de administrador.

En la parte superior se tienen tres opciones de ir a la *Página de Inicio*, *Atrás* y de *Cerrar Sesión*. La opción de *Atrás* permite regresar al menú de las *Tareas de Administrador*.

Como se puede observar se podrá realizar dos tipos de consultas:

- Consulta por Parqueadero, en un rango de fechas escogidas.
- Consulta General de todos los parqueaderos por día.

Para realizar la primera, se debe escoger un parqueadero de acuerdo al identificador posición, aquí se colocó un cuadro combinado (combo box) para poder escoger la posición que se desee, los datos que aquí se muestran son datos consultados de la base de datos, para poder así obtener todas las posiciones del parqueadero, En la Figura 49 se muestran las posiciones de los parqueaderos que se encuentran en la Base de Datos.

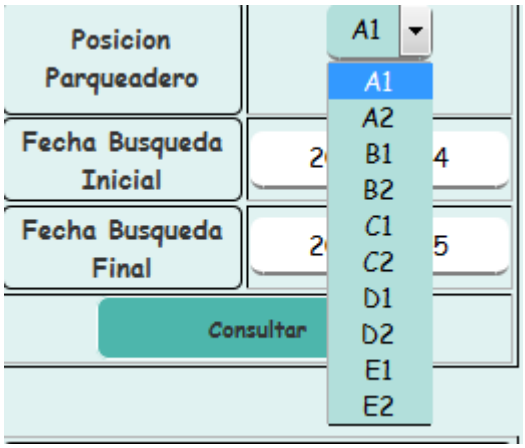
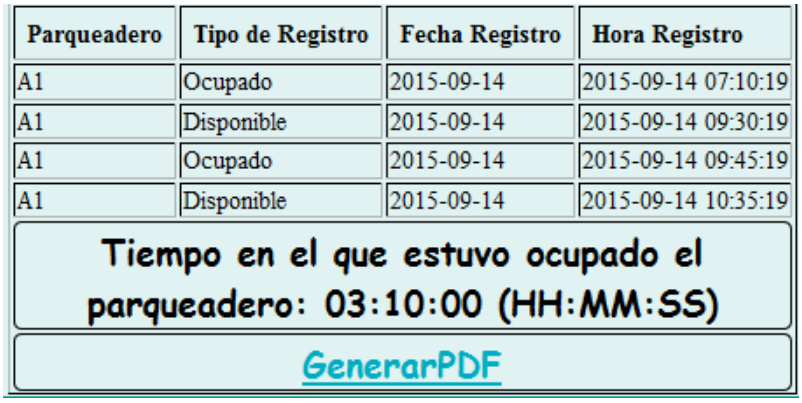


Figura 49. Posiciones de parqueaderos que constan en la Base de Datos.

Los dos siguientes parámetros para realizar la consulta son Fecha de Búsqueda Inicial y Final. El formato es YYYY:MM:DD (AÑO, MES, DIA).

Para el siguiente ejemplo se realizará una Búsqueda con el Parqueadero de posición A1, y de Fecha Inicial el 14 de Septiembre del 2015 y Fecha Final el mismo día. Estos parámetros son enviados en una sentencia MYSQL, de tipo *Select* con las 3 condiciones, si existen registros en esas fechas, se mostrarán en una Tabla, como se muestra en la Figura 50.



Parqueadero	Tipo de Registro	Fecha Registro	Hora Registro
A1	Ocupado	2015-09-14	2015-09-14 07:10:19
A1	Disponible	2015-09-14	2015-09-14 09:30:19
A1	Ocupado	2015-09-14	2015-09-14 09:45:19
A1	Disponible	2015-09-14	2015-09-14 10:35:19

Tiempo en el que estuvo ocupado el parqueadero: 03:10:00 (HH:MM:SS)

GenerarPDF

Figura 50. Ejemplo de resultado de consulta de registros por Parqueadero y rango de fechas establecidas.

Adicionalmente muestra el tiempo en el que estuvo ocupado el parqueadero en el formato (HH:MM:SS).

El algoritmo utilizado para calcular el tiempo en el que el que el parqueadero estuvo ocupado, es el siguiente:

- Se guardan los registros obtenidos en la consulta en un arreglo de objetos de la clase Registros, esta clase tiene los mismos atributos de la tabla *registro* en la base de datos.
- Se analiza cada uno de los objetos del arreglo de acuerdo al tipo (atributo) de registro, se tomará en cuenta los registros que sean de tipo Disponible.
- Si el registro anterior es de tipo Ocupado se efectúa la resta de las respectivas horas en los registros para encontrar el tiempo que estuvo ocupado en este evento.
- Así en un lazo *for* tomando en cuenta el tamaño del arreglo de registros, se efectuará la operación para encontrar el tiempo de todos los eventos que se produjeron entre las fechas antes mencionadas.
- Se procede a sumar el tiempo de todos los eventos y así se tiene el tiempo total en el que estuvo ocupado el parqueadero.

Estos datos pueden ser exportados a través de la generación de un archivo PDF, la Opción se muestra en la parte final de la Tabla. Esto se realizará con ***FPDF***, la cual es una clase PHP que permiten generar archivos PDF, puede ser descargada de http://www.fpdf.org/.

Al dar clic en esta opción aparece la siguiente ventana que va a permitir abrir o guardar el archivo:

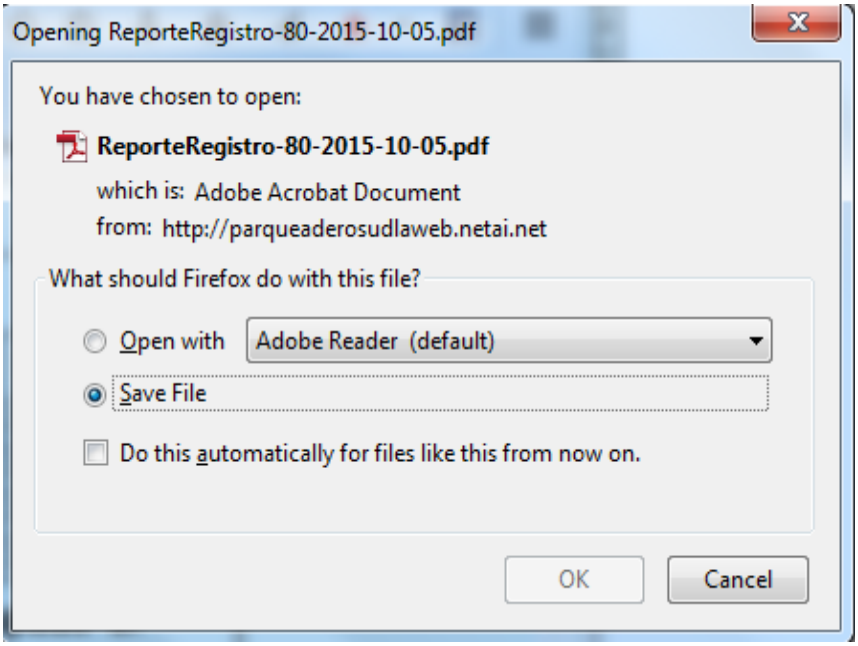


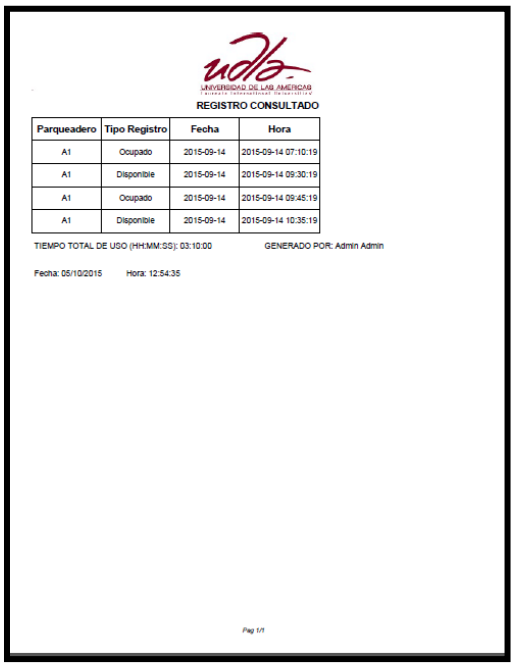
Figura 51. Ventana que aparece cuando se genera un archivo PDF de la consulta de registros por parqueadero en rango de fechas.

El formato del nombre del archivo es:

- “ReporteRegistro”.
- Numero Randómico entre 0 y 1000.
- Acompañado de la fecha actual.

Este formato va a permitir evitar la repetición del nombre con algún otro generado anteriormente. Adicionalmente en el archivo se incluirá la Fecha y Hora en la cual fue generado el PDF y el nombre del Administrador que lo realizó.

El archivo también mostrará la información previamente vista en la Tabla de la consulta, como se muestra a continuación en la Figura 52.



REGISTRO CONSULTADO

Parqueadero	Tipo Registro	Fecha	Hora
A1	Ocupado	2015-09-14	2015-09-14 07:10:19
A1	Disponible	2015-09-14	2015-09-14 09:30:19
A1	Ocupado	2015-09-14	2015-09-14 09:45:19
A1	Disponible	2015-09-14	2015-09-14 10:35:19

TIEMPO TOTAL DE USO (HH:MM:SS): 03:10:00 GENERADO POR: Admin Admin

Fecha: 05/10/2015 Hora: 12:54:35

Pag 1/1

Figura 52. Archivo PDF generado en la consulta de registros por parqueadero.

Para el segundo tipo de consulta, solo se necesitará la fecha en la cual se desea obtener los datos como parámetro, como se muestra en la Figura 53.

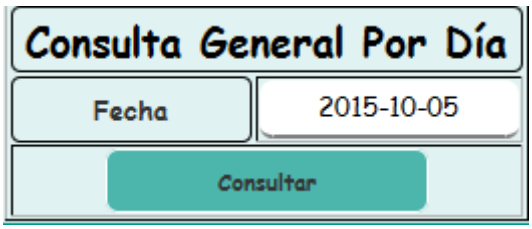


Figura 53. Recuadro de consulta general por día.

Los datos a obtener son el número de parqueaderos ocupado y disponibles, y el porcentaje del parqueadero lleno y vacío; esto por cada hora en el horario de 6 de la mañana a 11 de la noche. Para el ejemplo la consulta se realizará en la fecha 14 de Septiembre del 2015, el resultado se muestra en la siguiente tabla:

Tabla 12. Ejemplo de tabla del resultado de la consulta general por día.

Hora	Ocupados	Disponibles	%Lleno	%Vacio
6- 7	0	10	0 %	100 %
7- 8	5	5	50 %	50 %
8- 9	3	7	30 %	70 %
9- 10	3	7	30 %	70 %
10- 11	3	7	30 %	70 %
11- 12	1	9	10 %	90 %
12- 13	2	8	20 %	80 %
13- 14	1	9	10 %	90 %
14- 15	1	9	10 %	90 %
15- 16	1	9	10 %	90 %
16- 17	1	9	10 %	90 %
17- 18	0	10	0 %	100 %
18- 19	0	10	0 %	100 %
19- 20	1	9	10 %	90 %
20- 21	0	10	0 %	100 %
21- 22	0	10	0 %	100 %
22- 23	0	10	0 %	100 %
GenerarPDF				
Generar Gráficas				

El algoritmo utilizado para obtener el número de parqueaderos Ocupados, es el siguiente:

- Se crea una función que realizara los siguientes procedimientos:
  a) Obteniendo un arreglo de 17 valores por cada parqueadero, los 17 valores representan las 17 horas en las que el parqueadero está abierto.
  b) Este arreglo estará conformado por valores 1 o 0, dependiendo si el parqueadero estuvo ocupado o no en esa hora. Al final se devolverá el arreglo del parqueadero.
- Se obtendrán todos los arreglos dependiendo el número de parqueaderos, en este caso 1.
- Se sumarán los valores de todos los arreglos en cada posición, en este caso serían 17 posiciones.
- Y así se obtendrá un arreglo resultado de 17 valores, donde se mostrará el número de parqueaderos ocupados por hora.

El algoritmo utilizado para determinar si un parqueadero estuvo o no ocupado en cada hora es el siguiente:

- Se tomará en cuenta el número de registros por hora, el último estado registrado del parqueadero por cada hora y una variable que determinara si el parqueadero estuvo ocupado o disponible en esa hora.
- Existen 4 condicionales.
    - Si el número de registros es mayor a uno y el ultimo estado es 0, la variable determinante será igual a 1, esto indica que a pesar de que el parqueadero está vacío, existieron más registros anteriormente indicando que si fue ocupado en esta hora.
    - Si no hay registros en esa hora, la variable determinante será igual al último estado registrado.
    - Si no hay registros en la primera hora de servicio, 6 a 7 am, la variable determinante será igual a 0.
    - Finalmente si son más de las 7 de la mañana, solo existe un registro en esa hora, el último estado registrado es 0 y la variable determinante añadida al arreglo anteriormente es igual a 1, la variable determinante es igual a 1.

Retornando a la Tabla donde se generó el resultado de la consulta, en las dos últimas filas se tiene dos opciones, *Generar PDF*, y *Generar Gráficas*.

Al momento de dar clic en la opción Generar PDF, se genera un PDF con los datos obtenidos en la tabla, la hora y fecha, y el Administrador que lo realizó. Como se explicó anteriormente esto se realiza con ***FPDF***.

Aparecerá una ventana con dos opciones, abrir o guardar el archivo, como se muestra en la Figura 54.

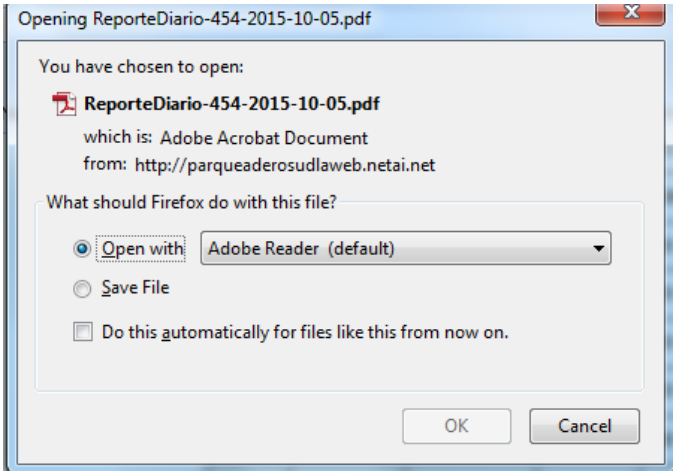
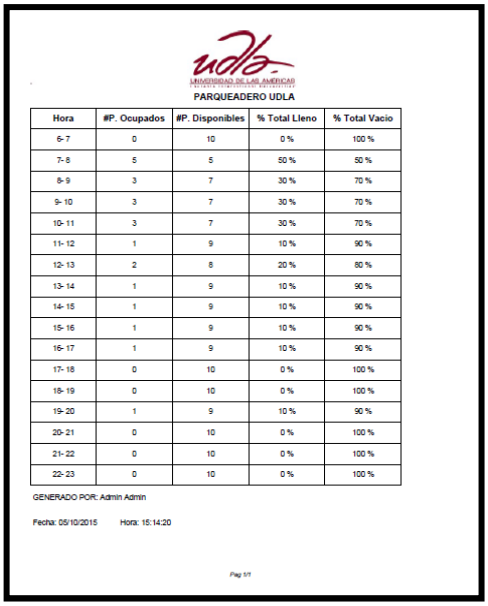


Figura 54. Ventana que aparece cuando se genera un archivo PDF de la consulta de reporte diario.

El formato del nombre del archivo es:

- “ReporteDiario”.
- Numero Randómico entre 0 y 1000.
- Acompañado de la fecha actual.

Este formato va a permitir evitar la repetición del nombre con algún otro generado anteriormente. El PDF del ejemplo de la consulta se muestra en la Figura 55.



PARQUEADERO UDLA

Hora	#P. Ocupados	#P. Disponibles	% Total Lleno	% Total Vacio
6- 7	0	10	0 %	100 %
7- 8	5	5	50 %	50 %
8- 9	3	7	30 %	70 %
9- 10	3	7	30 %	70 %
10- 11	3	7	30 %	70 %
11- 12	1	9	10 %	90 %
12- 13	2	8	20 %	80 %
13- 14	1	9	10 %	90 %
14- 15	1	9	10 %	90 %
15- 16	1	9	10 %	90 %
16- 17	1	9	10 %	90 %
17- 18	0	10	0 %	100 %
18- 19	0	10	0 %	100 %
19- 20	1	9	10 %	90 %
20- 21	0	10	0 %	100 %
21- 22	0	10	0 %	100 %
22- 23	0	10	0 %	100 %

GENERADO POR: Admin Admin

Fecha: 05/10/2015 Hora: 15:14:20

Figura 55. Archivo PDF generado en la consulta de reporte diario.

Si se escoge la opción Generar Gráficas, se mostrarán gráficas tipo pastel con porcentaje de uso por horas del estacionamiento en general, como se muestra en la figura 56.

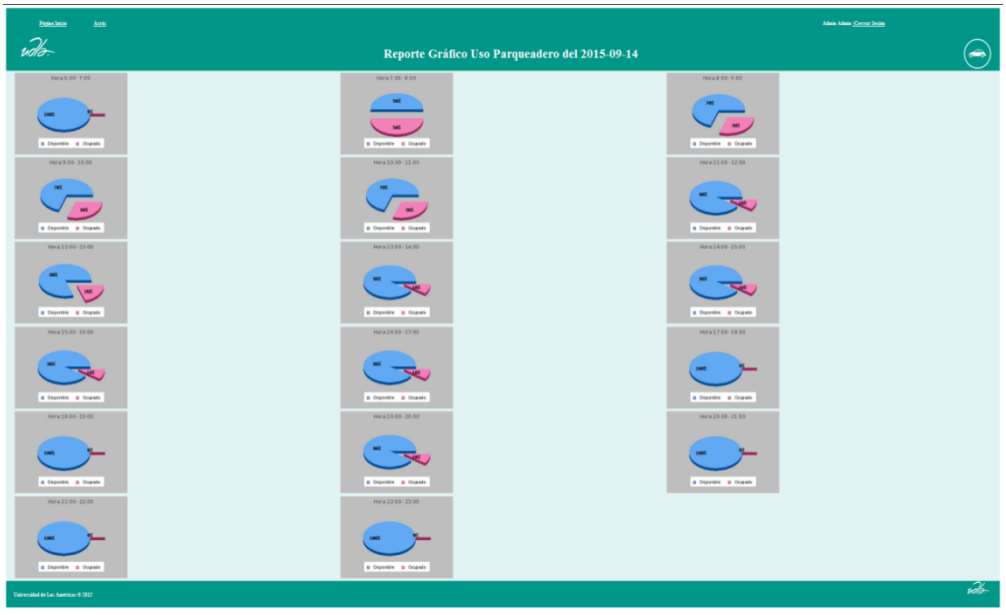


Figura 56. Gráficas de porcentaje de uso de parqueadero en general por horas.

En la parte superior se tienen tres opciones de ir a la *Página de Inicio*, *Atrás* y de *Cerrar Sesión*, además de que aparecerá el nombre y apellido del Administrador que ingresó, en este caso *Admin Admin.* La opción de *Atrás* permite regresar a la Interfaz de *Registros*.

Como se puede observar, aparece una gráfica tipo pastel por cada hora de servicio del Parqueadero. Color azul indica el porcentaje del estacionamiento que estuvo vacío y con rozado el porcentaje que estuvo lleno, utilizando los valores obtenidos en la tabla que se obtuvo anteriormente.

Estas gráficas se desarrollaron con ***JPGRAPH***, que es un creador de gráficas orientado a objetos compatible con versiones PHP desde la versión 5.1 a la 5.6. La Biblioteca puede ser descargada en el link: http://jpgraph.net/.
Si se da clic en la opción Crear, Actualizar, Eliminar en la página de *Administración de Parqueaderos*, aparecerá la interfaz como se muestra en la Figura 57.

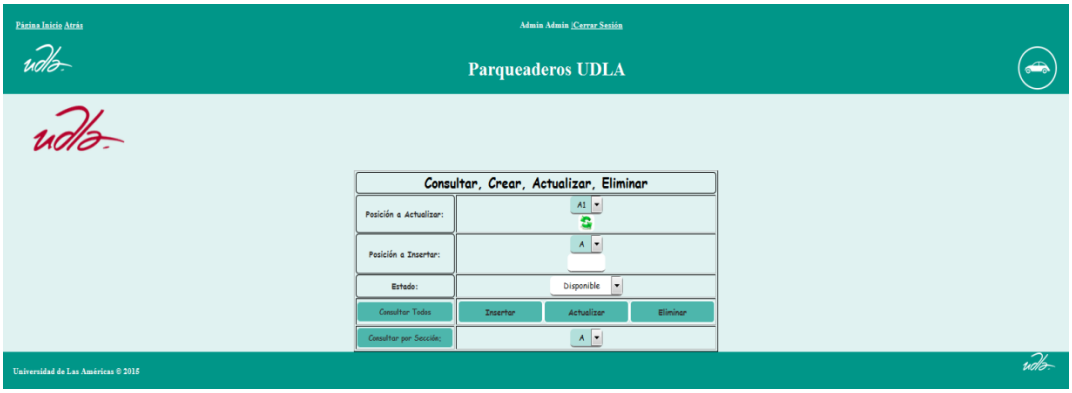


Figura 57. Interfaz para crear, actualizar y eliminar parqueaderos.

En la parte superior se tienen tres opciones: *Página de Inicio*, *Atrás* y de *Cerrar Sesión*, además de que aparecerá el nombre y apellido del Administrador que ingresó, en este caso *Admin Admin.* La opción de *Atrás* permite regresar al menú de las Tareas de Administrador.

En esta Opción, el Administrador puede realizar lo siguiente:

- Consultar los datos (Posición, Estado) de todos los parqueaderos, o filtrar sección.
- Insertar un nuevo parqueadero a la Tabla Parqueadero.
- Actualizar el estado de un parqueadero existente.
- Eliminar un Parqueadero existente, en esta opción hay que tener mucho cuidado debido a que si se elimina un parqueadero se eliminan sus registros.

  Las consultas, son realizadas a través de Sentencias SQL de tipo Select.

Para consultar por sección solo se añade la condición que la posición de los parqueaderos empiece con la letra seleccionada. La sección es una letra, en este caso para filtrar por sección se tendrá un cuadro combinado con todas las letras del abecedario, como se muestra en la Figura 58.

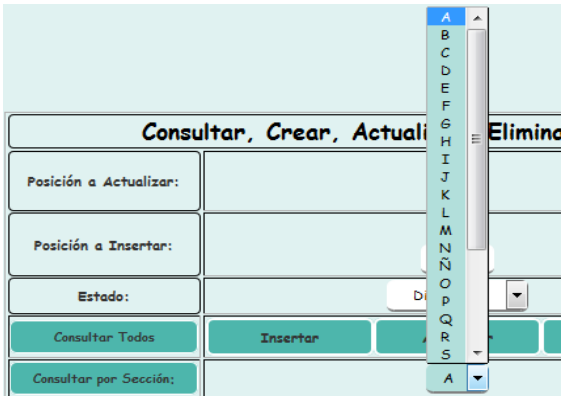


Figura 58. Listado de secciones a elegir.

Para insertar un nuevo parqueadero a la Base de Datos se utiliza sentencias MYSQL de tipo *Insert* con los campos requeridos. Para esto se debe escoger una posición que viene dada por una sección y un número de parqueadero, y el estado entre disponible u ocupado, el valor por defecto es disponible. La sección es escogida en el cuadro combinado en donde aparecen todas las letras del abecedario y adicionalmente se añade el número de parqueadero,

este cuadro de texto está validado para que solo se pueda ingresar números y que al momento de insertar no pueda estar vacío. Si se inserta un Parqueadero ya existente aparecerá el mensaje mostrado en la Figura 59.

Figura 59. Mensaje al consultar un parqueadero no existente.

Para Actualizar el estado de un parqueadero en la Base de Datos se utiliza sentencias MYSQL de tipo *Update*. Se debe escoger la posición del parqueadero en el cuadro combinado de las posiciones, posteriormente se escoge el estado al que se va a actualizar.

Finalmente para Eliminar un parqueadero en la Base de Datos se utiliza sentencias MYSQL de tipo *Delete*. Se escoge la posición del parqueadero en el cuadro combinado de las posiciones, esta opción solo se recomienda utilizar para eliminar un parqueadero mal ingresado, ya que si se elimina un parqueadero que ya contenga registros, los registros también serán eliminados.

La última Opción presenta dos alternativas:

- Crear un nuevo Administrador.
- Actualizar mi Usuario.

Figura 60. Opciones para crear un nuevo usuario o actualizar el usuario que inicio sesión.

En la primera alternativa se podrá crear un nuevo Usuario que será insertado a la tabla Administrador de la Base de Datos. Los campos para un nuevo Administrador son:

- Nombre
- Apellido

- Usuario
- Contraseña(Encriptada MD5)

Al dar clic en esta alternativa aparecerá la interfaz que se muestra en la Figura 61.

Figura 61. Interfaz para crear un nuevo usuario administrador.

En la parte superior se tienen tres opciones: *Página de Inicio*, *Atrás* y de *Cerrar Sesión*, además de que aparecerá el nombre y apellido del Administrador que ingresó, en este caso *Admin Admin.* La opción de *Atrás* permite regresar al menú de las Tareas de Administrador.

Validaciones:

- Los cuadros de textos están validados para que cuando se dé clic en Crear Usuario no puedan estar vacíos.
- Las contraseñas están validadas para que sean iguales caso contrario no se Insertará el nuevo administrador.
- Nombre y Apellido esta validado para que solo se pueda ingresar Letras.
- Nombre de Usuario esta validado para que se pueda ingresar números y letras, con una longitud máxima de 15.
- Si el Nombre de Usuario ya existe no permitirá el ingreso del nuevo Administrador.

Para insertar el nuevo administrador a la Base de Datos, se utiliza sentencias MYSQL de tipo *Insert*, donde los campos a insertar serán los ingresados por el Usuario. En la segunda alternativa que es *Actualizar mi Usuario*, aparecerá la siguiente interfaz:

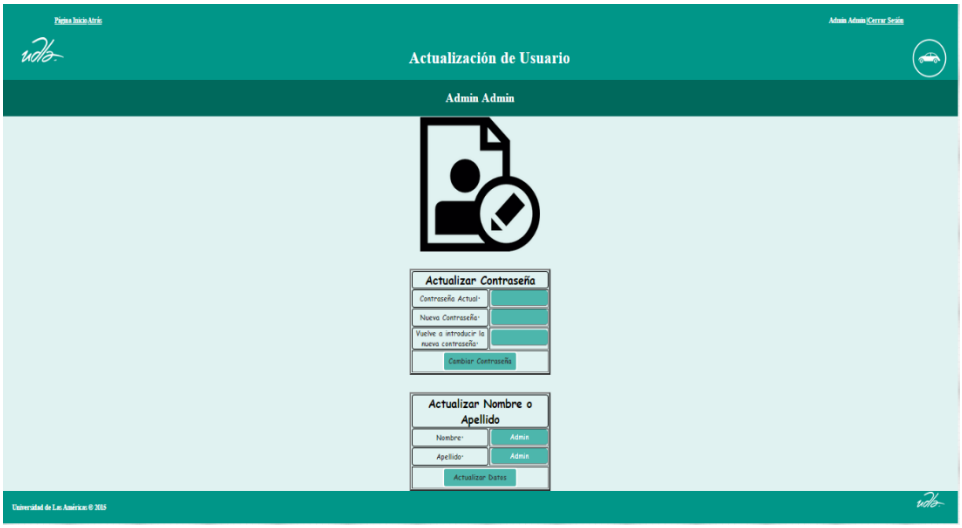


Figura 62. Interfaz para actualizar el usuario administrador que inicio sesión.

En la parte superior se tienen tres opciones: *Página de Inicio*, *Atrás* y de *Cerrar Sesión*, además de que aparecerá el nombre y apellido del Administrador que ingresó, en este caso *Admin Admin.* La opción de *Atrás* permite regresar al menú de las Tareas de Administrador.

Como se puede observar se puede actualizar la contraseña, y actualizar los datos personales Nombre y Apellido.

Validaciones:

- En el cambio de contraseña se solicitará la contraseña actual, si la contraseña actual es ingresada de manera errónea no se efectuará ningún cambio.

- Los cuadros de texto de las contraseñas son de tipo *password,* para que no puedan ser observados. También están validados para que al momento de dar clic en Cambiar Contraseña no puedan estar vacíos. Finalmente si las contraseñas no coinciden no se ejecutará el cambio.

- En los cuadros de texto Nombre y Apellido aparecerán los datos del Usuario que Inicio Sesión, estos podrán ser modificados, una vez ingresados los cambios se dará clic en Actualizar Datos para efectuar los cambios.

Para realizar las actualizaciones se utilizará sentencias MYSQL de tipo *Update*. En donde los campos a actualizar en el Usuario que inicio sesión son los datos ingresados por el usuario.

4.1.3. Seguridades

- Una de las seguridades del Sistema es la encriptación del password; en este caso se utilizó el algoritmo MD5.

- Si la aplicación va a estar en un servidor local se debe tener en cuenta los puertos habilitados para evitar algún tipo de ataque.
- Se recomienda la utilización de contraseñas complejas, es decir contraseñas que contengan al menos un carácter especial, y evitar uso de fechas de nacimiento, nombres y apellidos para evitar algún caso de hackeo de las cuentas de Usuarios Administradores. Ejemplo de contraseña compleja: *mju76yhn@.*

- Si se utiliza Hosting Web, por lo general los servidores de estas compañías presentan las seguridades necesarias para evitar algún tipo de ataque.

4.1.4. Hosting

Para cargar la aplicación al Internet, hay dos opciones:

- Tener un servidor local el cual cuente con la instalación de Apache 2.4, PHP 5.5.23 y MySQL Workbench Server Management, contratar una IP pública fija, y así asociarla a un dominio, con esto se podrá acceder a la aplicación en cualquier navegador, solo escribiendo la dirección de Dominio asociada al Servidor.

- La otra opción es contratar un servicio de *Hosting Web*; el cual cuente con Apache, PHP y MySQL, aquí se subirá la aplicación web y se deberá cargar el script de la Base de Datos. En cuanto a la conexión a la Base de Datos, se deberá cambiar el hostname, usuario, password y el nombre de la base de datos.

En este caso se ha buscado un Servicio Hosting Web gratuito, se encontró dos sitios en internet, los cuales son:

- www.hostinger.co
- http://www.000webhost.com/

Al haber escogido servicio gratuito presentan varias limitaciones pero que para la demostración del funcionamiento del sistema es suficiente.

Los dominios adquiridos son:

- http://parqueaderosudlaweb.netai.net
- http://parqueaderoudla.esy.es

4.1.5. Comunicación Arduino-Base de Datos mediante HTTP Request tipo POST.

Para esto se crea un archivo PHP, con variables tipo POST, las cuales son:

- Id del Parqueadero.
- Estado del Parqueadero.

Lo que se va a realizar es la actualización del estado del id del parqueadero recibido, y añadir un registro nuevo de acuerdo al id y estado recibido. Las variables son de tipo post, para que el Arduino al momento de realizar una petición http post apuntando a este archivo, reciba los datos de las variables (id y estado) y realice las operaciones en la Base de Datos.

4.1.1. Diagramas de flujo de la aplicación web

En la Figura 63, 64, 65 y 66 se muestran los diagramas de flujo de la aplicación web.

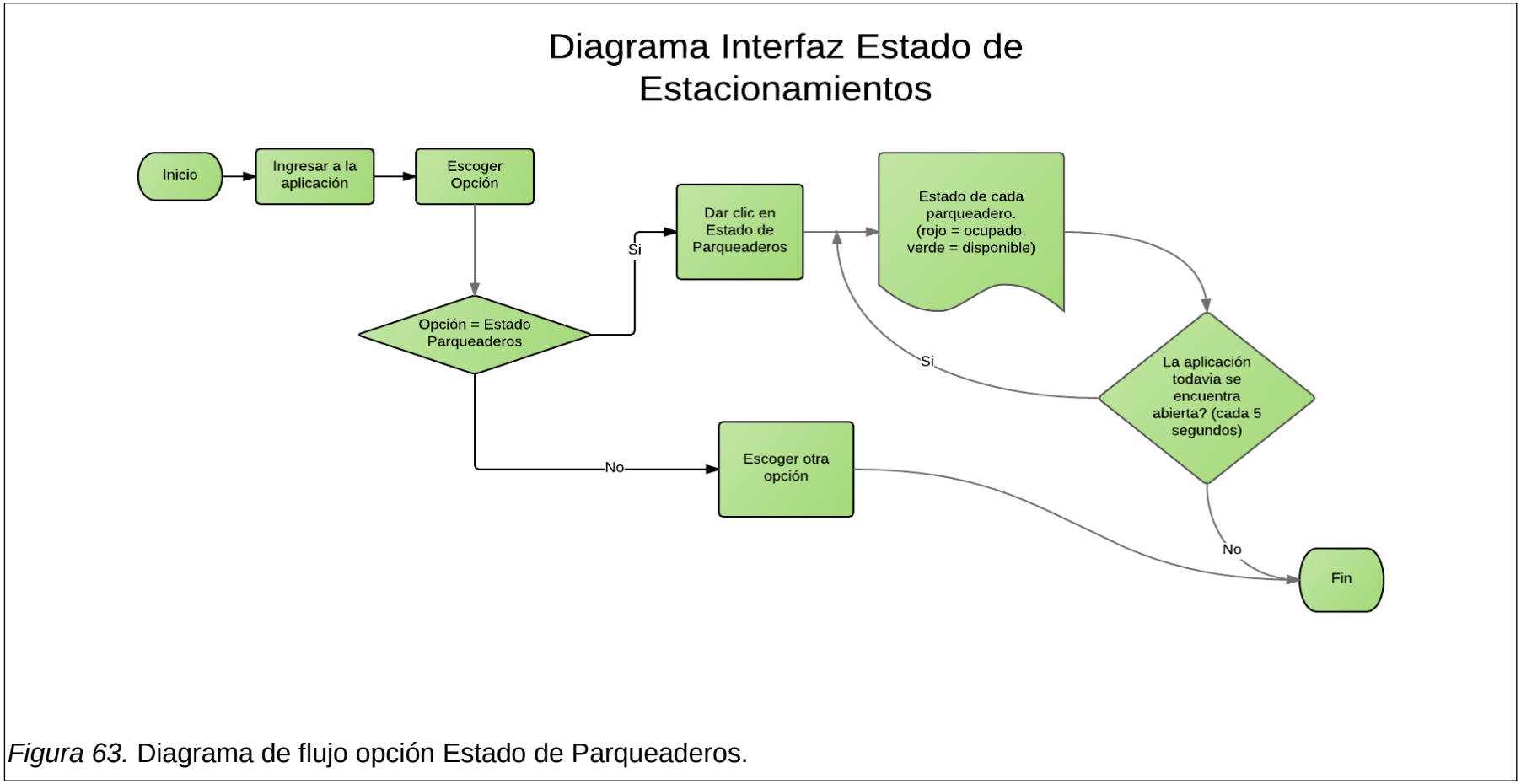


Figura 63. Diagrama de flujo opción Estado de Parqueaderos.

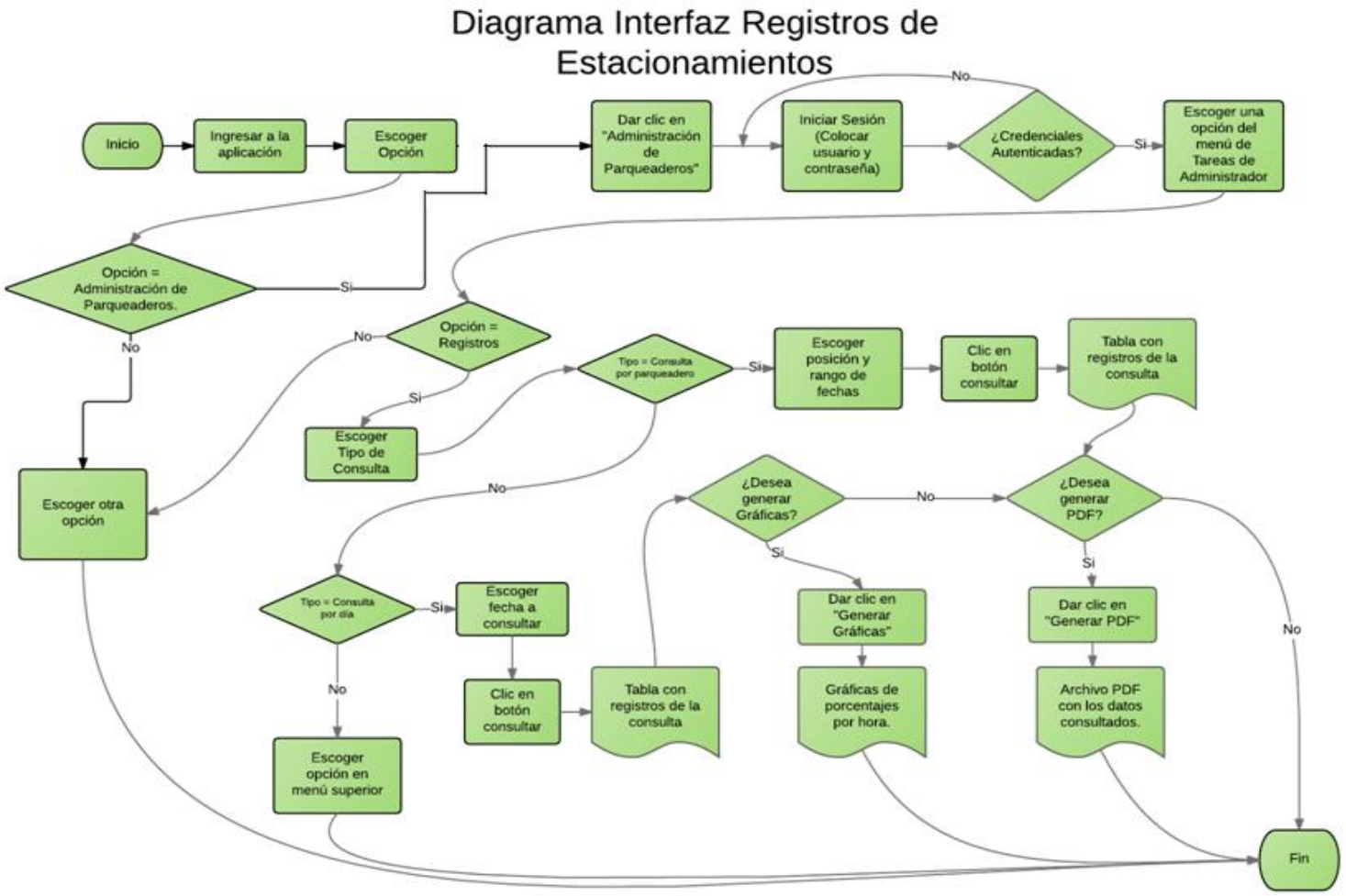


Figura 64. Diagrama de flujo opción Registro de Parqueaderos.

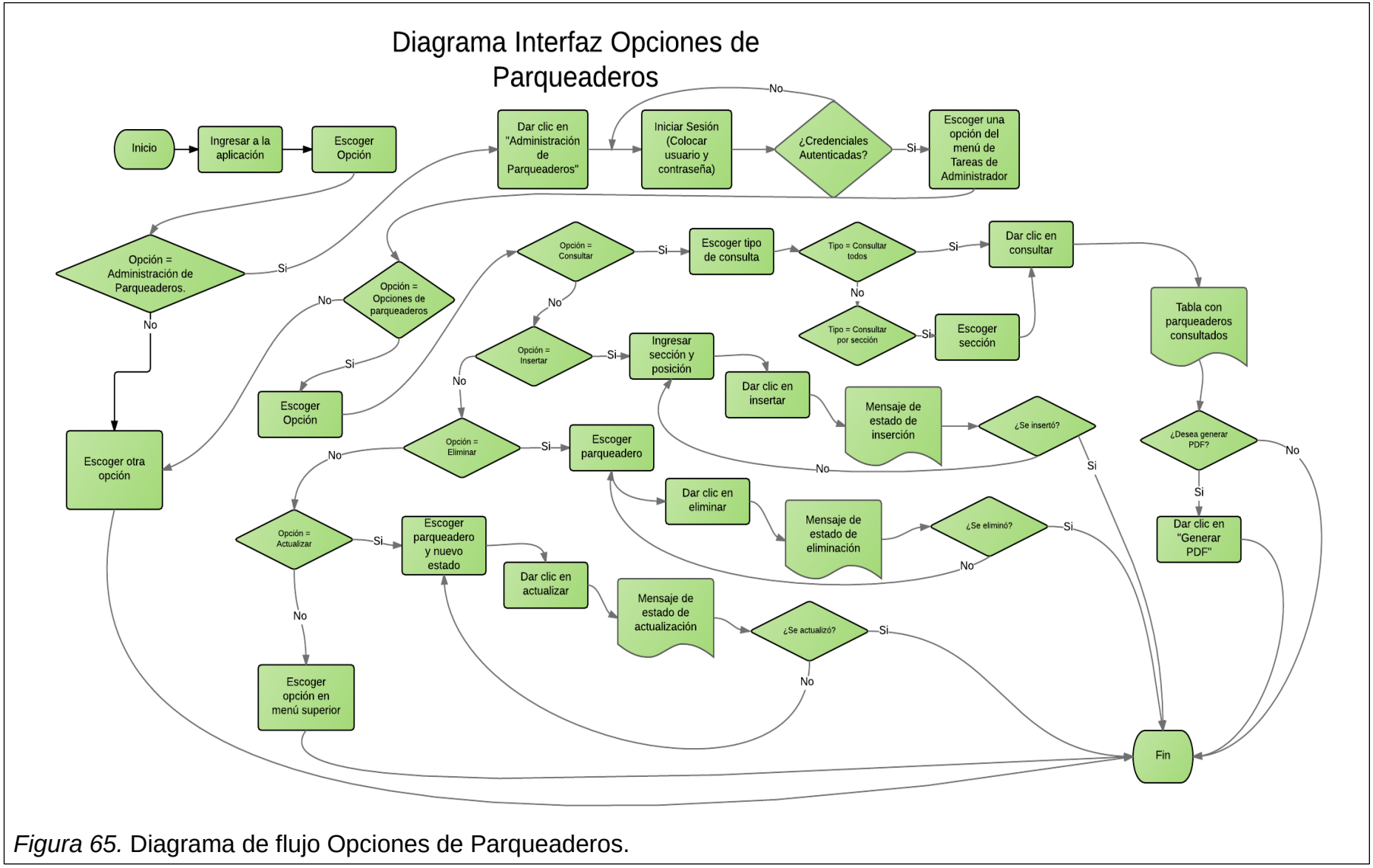


Figura 65. Diagrama de flujo Opciones de Parqueaderos.

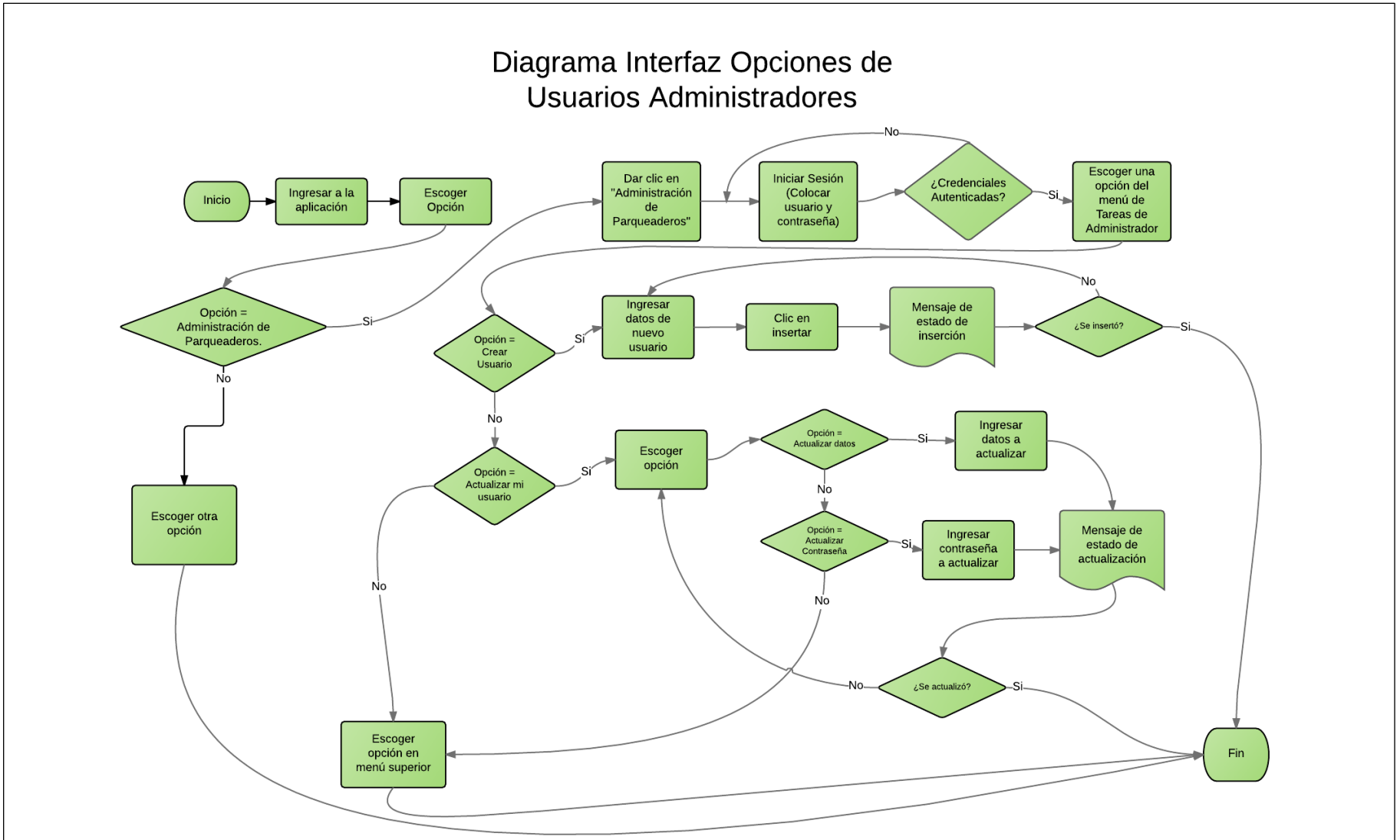


Figura 66. Diagrama de flujo Opciones de Usuarios Administradores.

4.2. Arduino Mega Programación

4.2.1. *Introducción*

La programación del Arduino se realiza a través del uso de varios lenguajes de programación y sean Java, C#, entre otros. En este caso se utilizará el lenguaje propio de código abierto basado en ***Processing,*** el cual es muy parecido a C++.

4.2.2. Desarrollo

Incluyendo Librerías

Se importan las librerías Ethernet y SPI, las cuales van a permitir controlar el Shield Ethernet, y así configurar direcciones MAC, IP y controlar la comunicación con el Arduino Mega mediante bus SPI.

Declaración de Variables

- Las variables para cada valor digital leído en los pines de control.
- La variable que almacenara la dirección MAC del Shield Ethernet.
- La variable que almacena la dirección IP fija asignada, en caso de que DHCP falle.
- La variable que almacenara la dirección del servidor, a donde se enviará la información.
- La variable para almacenar la información que será enviada al servidor mediante el método POST del protocolo HTTP.
- Las variables que permitirán que solo se ejecute una vez el cambio de estado en los sensores.

Configuración

En la configuración se procederá a asignar los pines respectivos del Arduino ya sean como entrada o salida.

- Pines de salida, para los Leds que indicaran el estado del parqueadero.
- Pines de entrada, aquellos que permiten obtener la información de los sensores.

Adicionalmente se realiza una lectura de los pines de entrada que brindan información de los sensores, y de acuerdo al valor leído se asignará un valor a las variables que permitirán que se ejecute una sola vez el cambio de estado.

Finalmente se inicia la comunicación serial, la cual va a permitir conocer el estado del proceso y así poder corregir errores en la programación.

Bucle

En el bucle se programa lo que se desea repetir constantemente. Mediante condicionales se verifica si el estado del parqueadero es 0 o 1 y adicionalmente se lo compara con la variable que permitirá realizar el cambio de estado una sola vez.

Dentro del condicional se coloca la acción que se desea realizar: existen dos casos:

- Si el estado del parqueadero es 0 de acuerdo al valor medido en el pin de control de entrada, y la variable declarada es 1, se realizará lo siguiente:

    a) Se procederá a apagar el Led de color verde y a encender el Led de color rojo, de acuerdo a los pines a los que estén conectados, previamente configurados como salidas.
    b) Se asigna la data en la variable declarada al inicio para enviar la información al servidor, es decir se le asigna los valores a las variables de tipo POST para poder realizar el siguiente paso (estado = 1 y el id correspondiente).

c) Posteriormente se procede a realizar el método POST del protocolo HTTP, apuntando a la dirección URL del servidor (un archivo ***.php***, que requiere variables de tipo POST) y utilizando la data asignada en el paso anterior. El archivo php al que se apunta debe actualizar en la Base de Datos el estado del respectivo parqueadero a 1 (Ocupado) y a la vez añadir un registro nuevo de tipo 1 (Ocupado), mediante sentencias SQL.

- Si el estado del parqueadero es 1 de acuerdo al valor medido en el pin de control de entrada y la variable declarada es 0, se realizará lo siguiente:

a) Se procederá a apagar el Led de color rojo y a encender el Led de color verde, de acuerdo a los pines a los que estén conectados, previamente configurados como salidas.
b) Se asigna la data en la variable declarada al inicio para enviar la información al servidor, es decir se le asigna los valores a las variables de tipo POST para poder realizar el siguiente paso (estado = 0 y el id correspondiente).
c) Posteriormente se procede a realizar el método POST del protocolo HTTP, apuntando a la dirección URL del servidor (un archivo ***.php***, que requiere variables de tipo POST) y utilizando la data asignada en el paso anterior. El archivo php al que se apunta debe actualizar en la Base de Datos el estado del respectivo parqueadero a 0 (Disponible) y a la vez añadir un registro nuevo de tipo 0 (Disponible), mediante sentencias SQL.

En este caso el número de parqueaderos para el prototipo es de diez, teniendo así un total de veinte condicionales, dos para cada estacionamiento. Al final se coloca un Delay de 100 milisegundos.

4.1.1. Diagrama de flujo de la programación en Arduino

En la Figura 67 se muestra el diagrama de flujo de la programación en el Arduino.

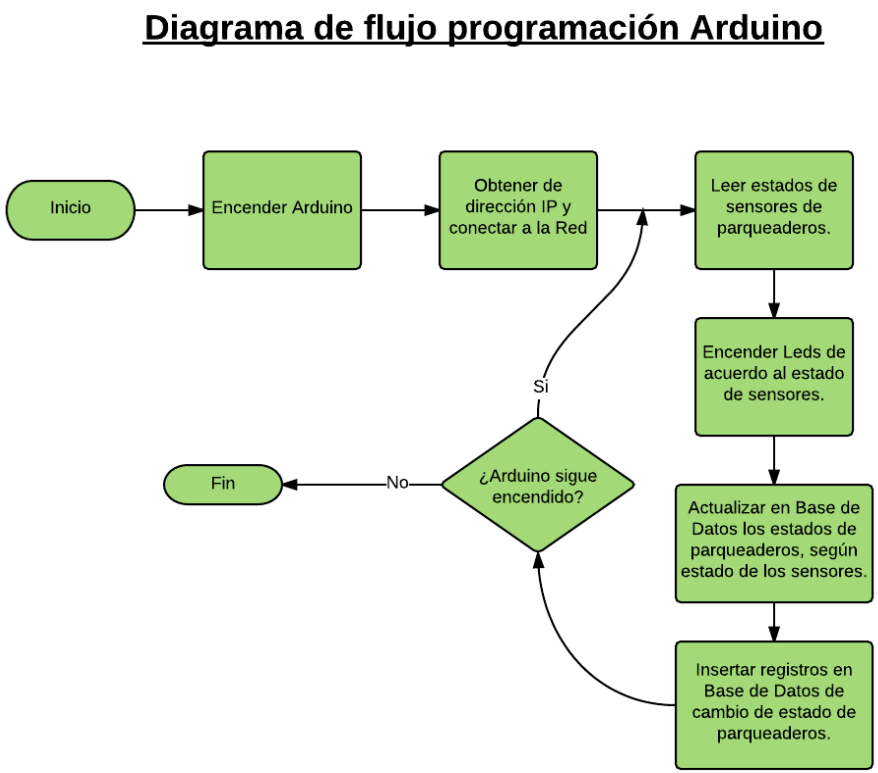


Figura 67. Diagrama de flujo programación Arduino.

4.2. Aplicación móvil en Android

El Software para el desarrollo de la Aplicación Móvil fue Android Studio V1.0, la cual reemplazó a eclipse como el IDE oficial para el desarrollo de aplicaciones para Android.

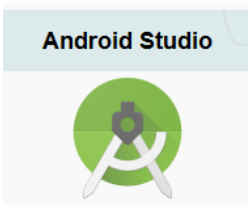


Figura 68. Logo Android Studio.

La versión SDK mínima configurada en el inicio para el desarrollo de la aplicación es Android 2.3 (Gingerbread), para así poder abarcar el 99.5 % de todos los dispositivos Android en el Mercado. Para el Desarrollo de la aplicación se necesitarán 3 clases:

- Parqueadero
- JSONParser
- MainActivity

Para conectar la aplicación Android a la Base de Datos MySQL, se realizará a través de objetos JSON, de igual manera como se realizó en la aplicación web. Como se explicó anteriormente el objeto JSON va a estar formado por un arreglo de objetos Parqueadero, el cual es generado por la Aplicación Web, a través de un archivo php. Como se muestra en la Figura 69.

{"parqueaderos":[{"idparqueadero":"1","posparqueadero":"A1","estparqueadero":"1"},
{"idparqueadero":"2","posparqueadero":"A2","estparqueadero":"1"},
{"idparqueadero":"3","posparqueadero":"B1","estparqueadero":"0"},
{"idparqueadero":"4","posparqueadero":"B2","estparqueadero":"0"},
{"idparqueadero":"5","posparqueadero":"C1","estparqueadero":"0"},
{"idparqueadero":"6","posparqueadero":"C2","estparqueadero":"0"},
{"idparqueadero":"7","posparqueadero":"D1","estparqueadero":"1"},
{"idparqueadero":"8","posparqueadero":"D2","estparqueadero":"0"},
{"idparqueadero":"9","posparqueadero":"E1","estparqueadero":"0"},
{"idparqueadero":"10","posparqueadero":"E2","estparqueadero":"0"}],"success":1}

Figura 69. Objeto JSON de la consulta de Tabla Parqueadero.

Cada objeto del arreglo contiene los datos obtenidos en una consulta MySQL de la tabla *Parqueadero* de la Base de Datos. Estos datos son requeridos a través del método GET del protocolo HTTP a la URL donde se encuentra el archivo ***.php***.

Los datos son guardados en una variable de tipo *String*. Para poder leer esta variable, es necesario importar la clase ***JSONObject***. Se crea una instancia de la clase y se incluye de atributo la variable de tipo *String* previamente asignada, y así se logra guardar la información en un Objeto JSON.

Los datos de los diez parqueaderos son almacenados en un *ArrayList* de Objetos Parqueaderos. De acuerdo al estado de cada parqueadero almacenado en *el ArrayList* se pintará el estacionamiento en el mapa del parqueadero. Si el estado es 0 el parqueadero se colocará de color verde y si el estado es 1, el parqueadero se colocará de color rojo.

La interfaz gráfica de aplicación fue realizada sobre un *RelativeLayout* para que la aplicación pueda ser vista en diferentes dispositivos de acuerdo al tamaño de su pantalla.

Una vez finalizada la programación de la aplicación se procede a generar el ***APK*** que será instalado en cualquier celular que tengas instalado la versión Android 2.3.3 en Adelante.

En la Figura 70 se puede observar el icono y la interfaz de la aplicación:

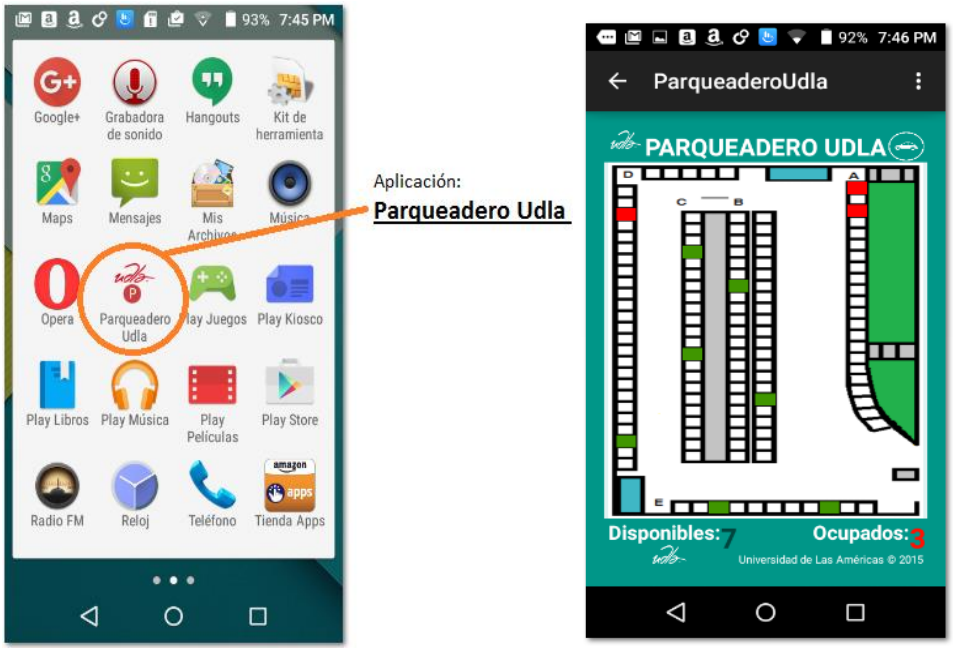


Figura 70. Ícono e Interfaz de la Aplicación Parqueadero Udla en Android.

Como se puede observar de color rojo están los parqueaderos ocupados y de color verde los disponibles. Para el prototipo se implementaron 10 parqueaderos, y en la figura, en la parte inferior muestra el número total de espacios disponibles (7) y ocupados (3).

4.2.1. Diagrama de flujo de la aplicación en Android

En la Figura 71 se muestra el diagrama de flujo de la aplicación en Android.

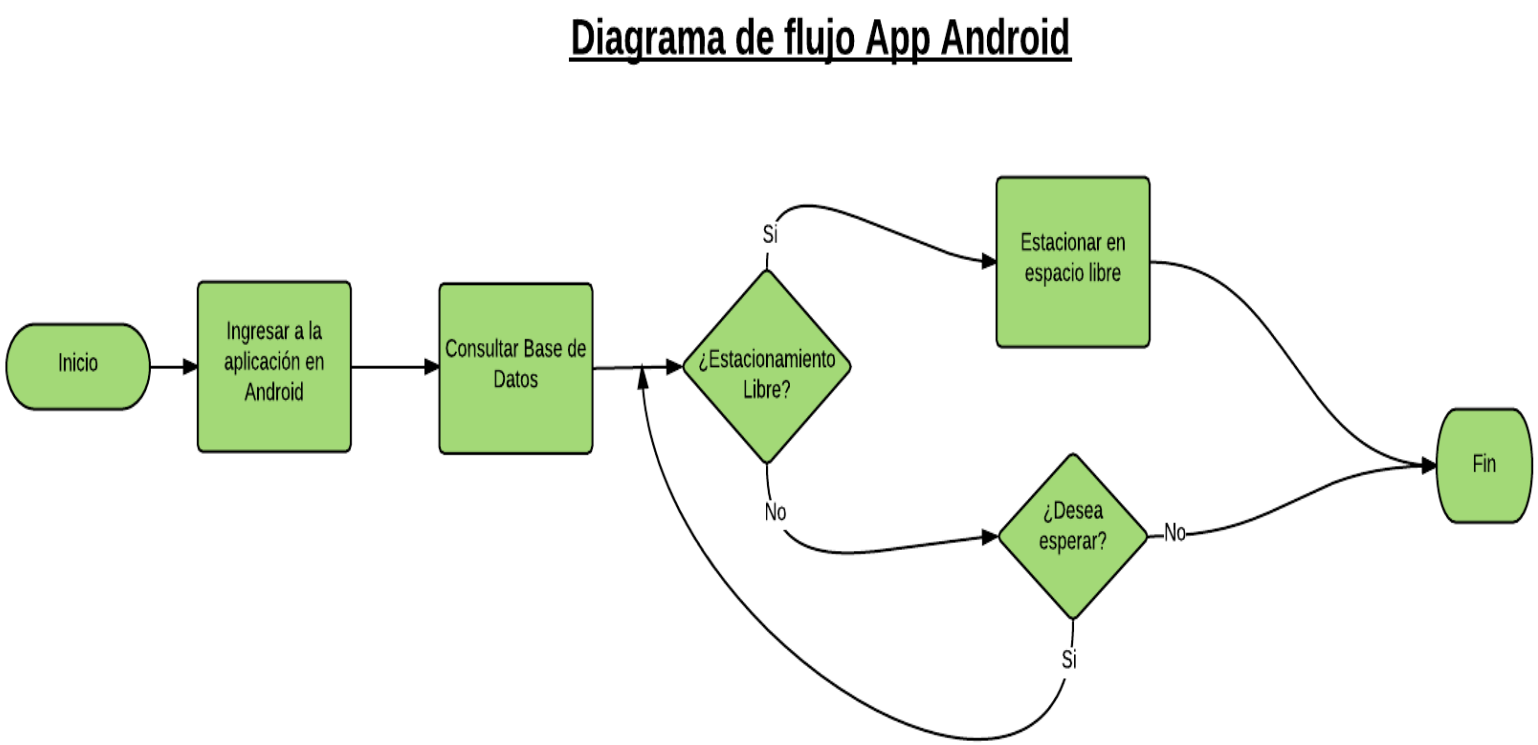


Figura 71. Diagrama de flujo programación Android.

5. Capítulo V: Pruebas y Resultados

5.1 Pruebas Sensores Infrarrojo TCRT5000:

Antes de soldar los sensores, es necesario probarlos uno a uno para comprobar que su funcionamiento sea correcto. Esto se realizará en Protoboard y se utilizará la conexión de la Figura 67.

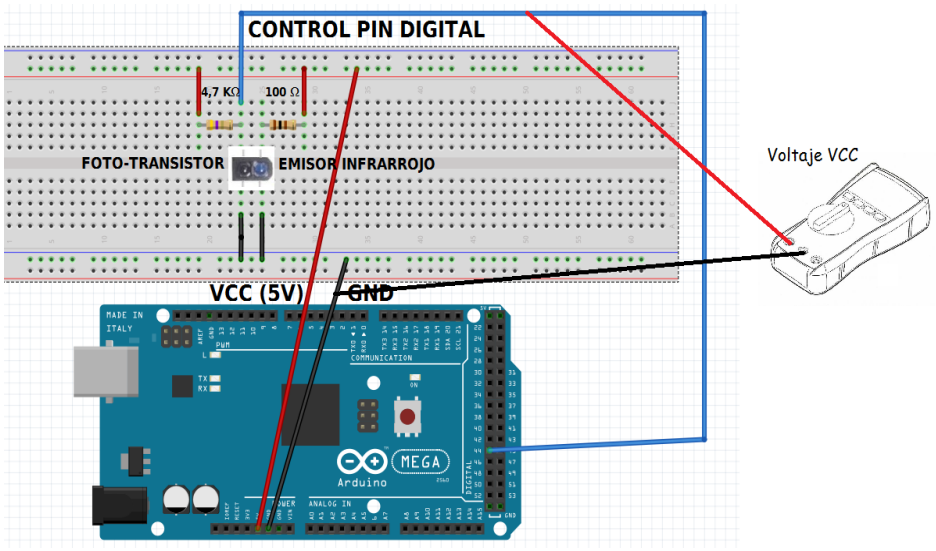


Figura 72. Conexión en Protoboard para las pruebas de sensores TCRT5000.

5.1.1 Distancia de Detección:

Las pruebas determinaron que para obtener un 0 lógico en la salida debe existir una distancia de máxima detección de 2.5 mm.

Si la distancia es mayor, el valor de voltaje es mayor a 0.8V y la salida no muestra un 0 lógico.

Si no existe ningún objeto donde se refleje la luz, el voltaje medido en el pin de control es de 4.8V.

5.1.2 Medición Voltaje Fototransistor, cuando la luz infrarroja es reflejada por un objeto.

Para cada sensor se procede a medir voltaje en el colector del fototransistor del Sensor.
Valor Voltaje Medido cuando existe reflexión: _0.25_V
Valor Voltaje Medido sin reflexión: _4.7_V

5.2 Pruebas Con Leds:

5.2.1 Encendido de Leds de acuerdo al estado del Sensor.

Una vez probados los 10 sensores TCRT5000, se realizan las pruebas de los Leds igualmente en Protoboard antes de soldarlos, para verificar que estén en correcto funcionamiento.

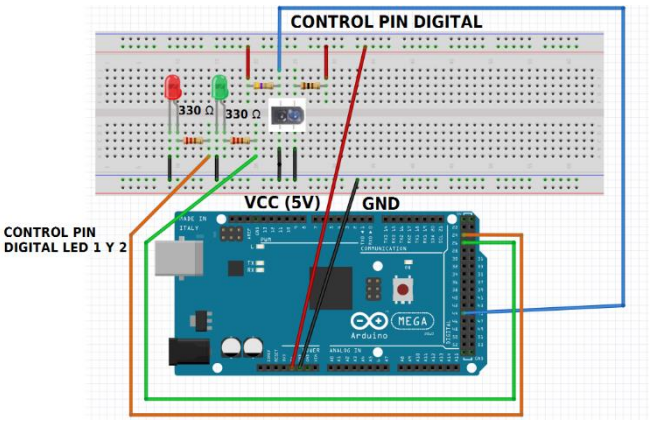


Figura 73. Conexión en Protoboard para las pruebas de encendido de Leds.

Se realiza la configuración para que cuando el Pin de control se encuentre en estado bajo, se envíe una señal de estado alto al pin donde está conectado el control del Led Rojo y estado bajo al control del Led Verde.

Y cuando el Pin de control del sensor, se encuentre en estado alto, se envíe una señal de estado alto al pin donde está conectado el control del Led Verde y estado bajo al pin de control del Led Rojo.

5.3 Pruebas Con Shield Ethernet:

5.3.1 Conexión a la Red, asignación de dirección IP:

Se procederá a observar que es lo que está sucediendo con el proceso de conexión del Ethernet Shield a la Red Local, a través de impresiones Seriales.

Una vez que una dirección IP sea asignada al Ethernet Shield, en la impresión serial aparecerá impreso la dirección IP correspondiente.

Como se muestra a en la Figura 74.

```
connecting...
Mi direccion IP es: 192.168.0.104.
```

Figura 74. Impresión serial que muestra que ha sido asignada una dirección IP al Shield Ethernet.

Esto demuestra que el Arduino está conectado a la Red, a través del Shield Ethernet.

5.3.2 HTTP Request, con método POST, de acuerdo al estado de cada Sensor y verificación en interfaz de estado de parqueaderos en aplicación Web y Android:

Los archivos .php de los servicios Hosting se encuentran en las siguientes direcciones:

- http://parqueaderosudlaweb.netai.net/ab/update.php
- http://parqueaderoudla.esy.es/ab/update.php

A estas direcciones se realizarán las peticiones HTTP de tipo POST, enviando el valor del id y el estado del parqueadero correspondiente.

El archivo *Update.php* como se explicó anteriormente va a realizar la actualización del estado del parqueadero y la inserción de un nuevo registro.

Para comprobar el funcionamiento se lo realizará a través de impresiones Seriales, como se lo realizó anteriormente. Cada vez que se dé un evento en el que cambió el estado del parqueadero, imprimirá la acción ejecutada.

A continuación en la Figura 75 se muestra el funcionamiento.

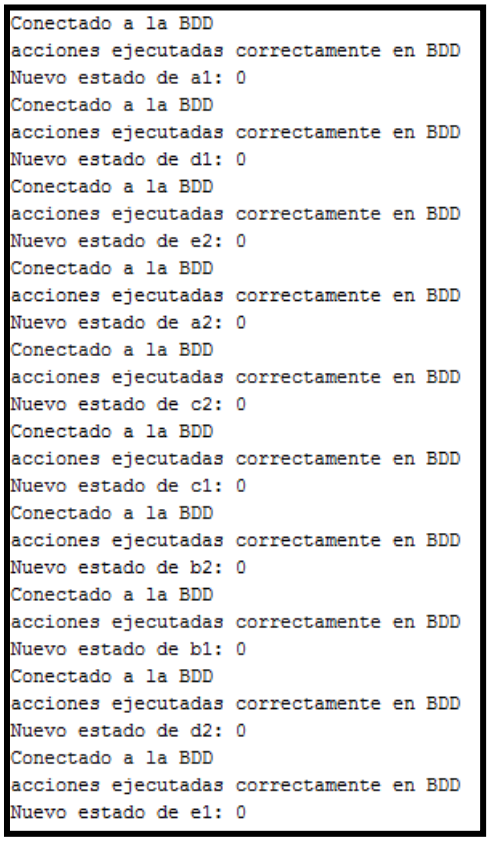


Figura 75. Impresión serial que muestra conexión a la Base de Datos y que se efectuaron los cambios en la base de datos.

Ahora se verificará que en la Base de Datos se hayan efectuado los cambios, tanto en la tabla *Registro* como en la Tabla *Parqueadero*.

Tabla 13. Registros del ejemplo añadidos en la Tabla Registros.

647	0	2015-10-06	2015-10-06 21:54:37	1
648	0	2015-10-06	2015-10-06 21:54:38	7
649	0	2015-10-06	2015-10-06 21:54:38	10
650	0	2015-10-06	2015-10-06 21:54:39	2
651	0	2015-10-06	2015-10-06 21:54:40	6
652	0	2015-10-06	2015-10-06 21:54:40	5
653	0	2015-10-06	2015-10-06 21:54:41	4
654	0	2015-10-06	2015-10-06 21:54:41	3
655	0	2015-10-06	2015-10-06 21:54:42	8
656	0	2015-10-06	2015-10-06 21:54:42	9

Como se puede observar en la tabla 13 se insertaron nuevos 10 registros, correspondientes a los diez eventos sucitados anteriormente.

Tabla 14. Datos en tabla *Parqueaderos* después de los cambios efectuados.

Index	Pos	Estado
1	A1	0
2	A2	0
3	B1	0
4	B2	0
5	C1	0
6	C2	0
7	D1	0
8	D2	0
9	E1	0
10	E2	0

Y en la Tabla *Parqueadero*, se puede observar que todos los estados estan en 0.

- Ahora se colocará un vehiculo en el parqueadero A1, como se observa en la Figura 76.

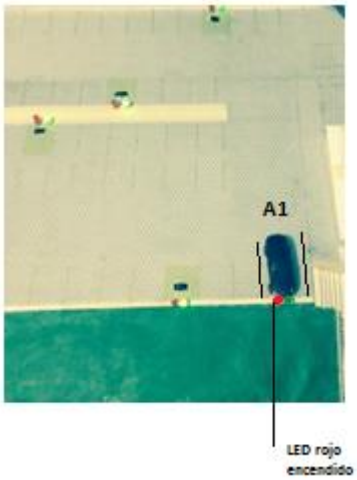


Figura 76. Estacionamiento A1 ocupado.
a. LED rojo encendido

En la impresión Serial del Arduino se comprueba que conste el movimiento, como se muestra en la Fiura 77.

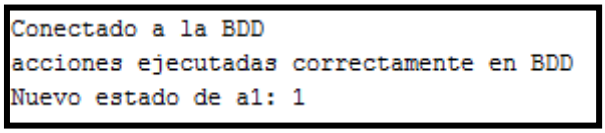
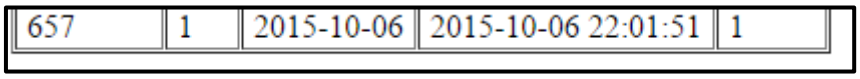


Figura 77. Verificación impresión serial que muestra conexión a la Base de Datos y que se efectuó el cambio de estado de parqueadero A1 a ocupado (1).

Ahora se verifica que se hayan efectuado los cambios en la Base de Datos, tanto en la Tabla *Registro* como en la Tabla *Parqueadero.*



657	1	2015-10-06	2015-10-06 22:01:51	1

*Figura 78.*Captura verificación de inserción de nuevo registro en tabla Registro de parqueadero A1 (id=1).

Como se puede observar en la figura 78 se añadio un registro en la Tabla *Registro*.

Tabla 15. Tabla de verificación de cambio de estado en tabla Parqueadero de parqueadero A1.

Index	Pos	Estado
1	A1	1
2	A2	0
3	B1	0

En la tabla *Parqueadero* se cambió el estado de este parqueadero, como se muestra en la tabla 15. Al acceder a la aplicación web, en la opción de estados de parqueaderos, se muestra que en efecto el parqueadero se encuentra ocupado, se puede observar en la Figura 79.

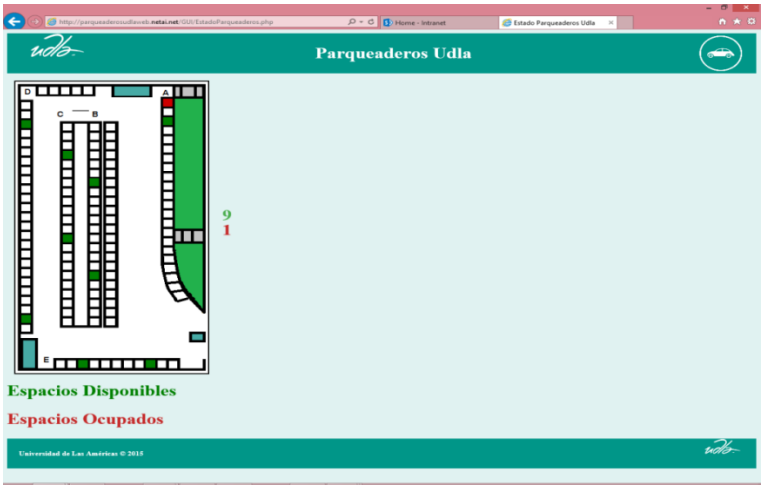


Figura 79. Verificación de cambio de estado en parqueadero A1 en Interfaz Web de estado de parqueaderos.

Se accede desde la Aplicación en Android y se obtiene el mismo resultado que en la aplicación web, como se muestra en la Figura 80.

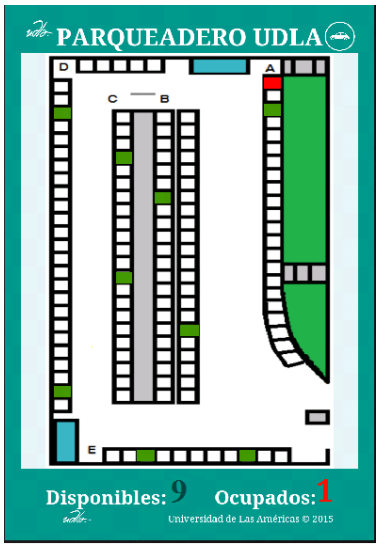


Figura 80. Verificación de cambio de estado de parqueadero A1 en Interfaz Android de estado de parqueaderos.

- Finalmente se retirará el vehiculo del parqueadero A1.

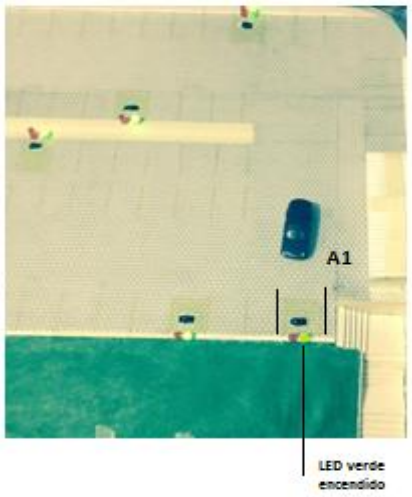


Figura 81. Estacionamiento A1 desocupado.
a. LED verde encendido

Se realiza la misma verificación:

Primeramente se comprueba que en la impresión Serial del Arduino conste el movimiento, como se muestra en la Figura 82.

```
Conectado a la BDD
acciones ejecutadas correctamente en BDD
Nuevo estado de a1: 0
```

Figura 82. Verificación impresión serial que muestra conexión a la Base de Datos y que se efectuó el cambio de estado de parqueadero A1 a disponible (0).

Y se comprueba que tanto en la Tabla *Registro* se haya añadido el evento y en la Tabla *Parqueadero* se haya haya actualizado el estado, se puede observar esta verificación en las figura 83 y tabla 16.

658	0	2015-10-06	2015-10-06 22:06:50	1

Figura 83. Captura verificación de inserción de nuevo registro en tabla Registro de parqueadero A1.

Tabla 16. Verificación de cambio de estado en tabla Parqueadero de parqueadero A1.

Index	Pos	Estado
1	A1	0
2	A2	0
3	B1	0

Al acceder a la aplicación web, en la opción de estados de parqueaderos, se muestra que en efecto el parqueadero paso a estar disponible. Esto se puede observar en la Figura 84.

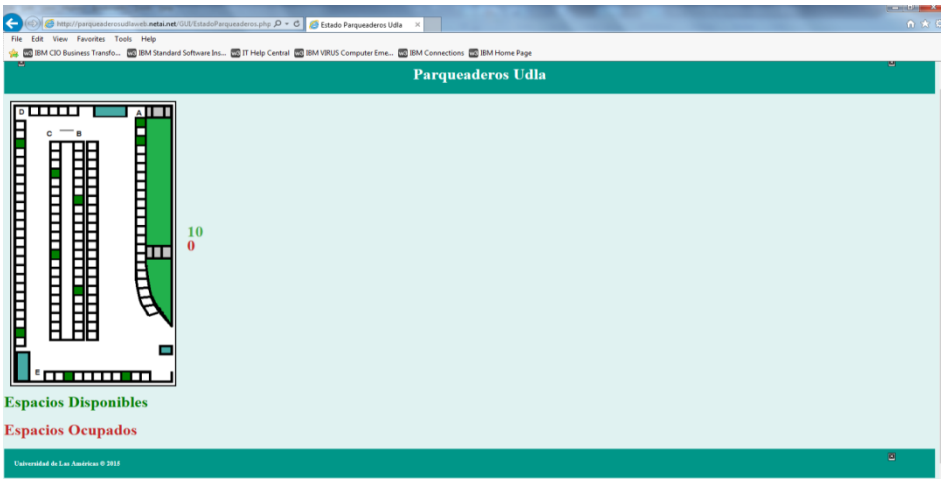


Figura 84. Verificación de cambio de estado de parqueadero A1 en Interfaz Web de estado de parqueaderos.

Se accede desde la Aplicación en Android y se obtiene el mismo resultado que en la aplicación web, como se muestra en la Figura 85.

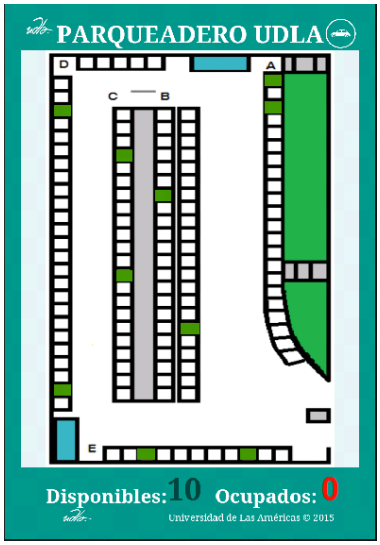


Figura 85. Verificación de cambio de estado en Interfaz Android de estado de parqueaderos.

5.4 Prueba de funciones en aplicación web:

5.4.1 Prueba de Inicio de Sesión:

- **Validaciones espacios en blanco:**

Si no se llena los cuadros de texto en el inicio de sesión, es decir usuario y contraseña, la aplicación mostrará el mensaje respectivo indicando que esos campos son requeridos para la autenticación. En las Figuras 86, 87 y 88 se muestran las verificaciones de las validaciones.

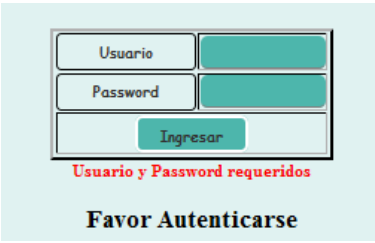


Figura 86. Verificación cuando no se llena ningún cuadro de texto en el inicio de sesión.

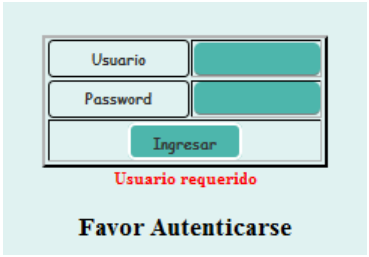


Figura 87. Verificación cuando no se llena el cuadro de texto de Usuario en el inicio de sesión.

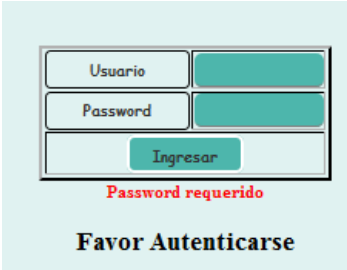


Figura 88. Verificación cuando no se llena el cuadro de texto de Password en el inicio de sesión.

- **Credenciales Falsas:**

Si se ingresa un usuario y contraseña erróneos, la aplicación no permitirá el acceso a las opciones de administrador, mostrando el mensaje que se muestra en la Figura 89, correspondiente a la verificación de credenciales falsas.

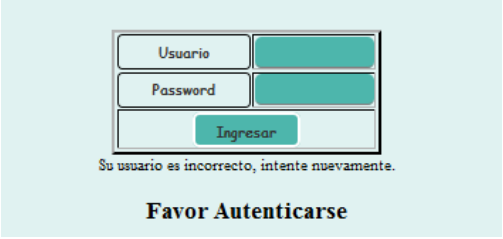


Figura 89. Verificación de credenciales falsas.

- **Credenciales Correctas:**

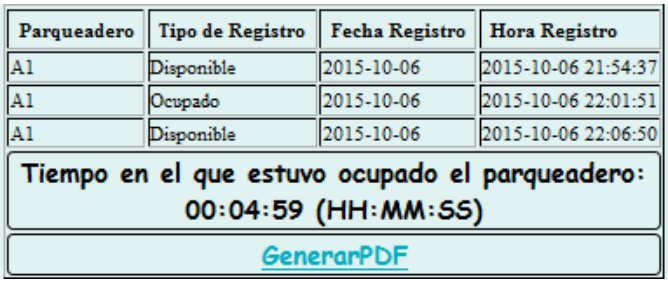
Si se ingresa las credenciales correctas, se podrá tener acceso a las opciones de administrador, que me muestran en la Figura 90.

Figura 90. Verificación de que se ingresó credenciales correctas en el inicio de sesión.

5.4.2 Prueba de Consultas de Registros.

- **Consulta por parqueadero y en rango de fechas:**

Utilizando los movimientos realizados previamente en el parqueadero A1, se puede observar los registros obtenidos en la consulta, mostrados en la Figura 91.



Parqueadero	Tipo de Registro	Fecha Registro	Hora Registro
A1	Disponible	2015-10-06	2015-10-06 21:54:37
A1	Ocupado	2015-10-06	2015-10-06 22:01:51
A1	Disponible	2015-10-06	2015-10-06 22:06:50

Tiempo en el que estuvo ocupado el parqueadero: 00:04:59 (HH:MM:SS)

GenerarPDF

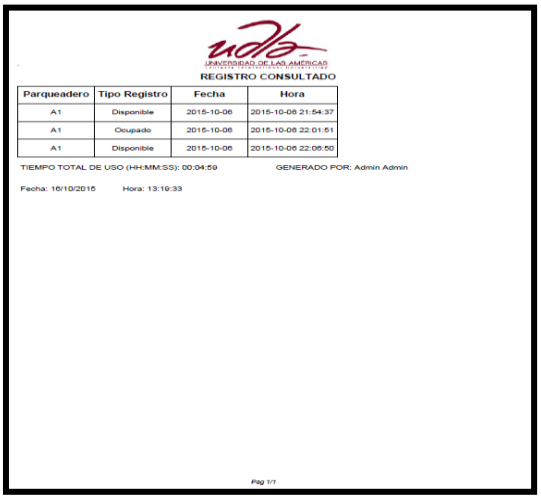
Figura 91. Verificación de consulta por parqueadero en rango de fechas establecidas.

De acuerdo a la verificación mostrada en la Figura 91, se puede deducir:

- En efecto existieron 3 registros:

1. Cuando se encendió el sistema y se actualizó el estado del parqueadero como disponible.
2. Cuando un vehículo procedió a ocupar el estacionamiento.
3. Cuando el mismo vehículo desocupo el estacionamiento.

- El cálculo del tiempo en el que estuvo es correcto; tomando la hora en la que fue ocupado (22:01:51) y en la que paso a estar disponible (22:06:50), lo cual da como resultado un uso de 4 minutos con 59 segundos.
- Se puede observar la opción de generar PDF. Al dar clic se generará el archivo respectivo. En la Figura 92, se observa el archivo generado a partir de la consulta realizada.


UNIVERSIDAD DE LAS AMÉRICAS

REGISTRO CONSULTADO

Parqueadero	Tipo Registro	Fecha	Hora
A1	Disponible	2015-10-06	2015-10-06 21:54:37
A1	Ocupado	2015-10-06	2015-10-06 22:01:51
A1	Disponible	2015-10-06	2015-10-06 22:06:50

TIEMPO TOTAL DE USO (HH:MM:SS): 00:04:59 GENERADO POR: Admin Admin

Fecha: 16/10/2015 Hora: 13:19:33

Pag 1/1

Figura 92. Verificación de generación de archivo PDF.

- **Consulta general por día:**

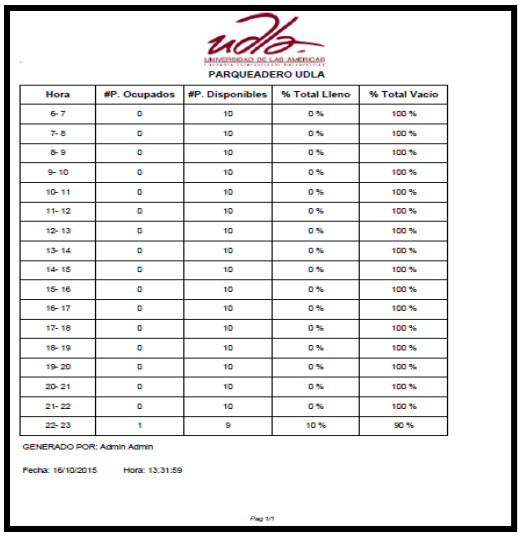
Para la verificación se utiliza igualmente los movimientos realizados previamente en el parqueadero A1, se puede observar los registros obtenidos en la consulta, mostrados en la tabla 17.

Tabla 17. Verificación de ocupación parqueadero en consulta diaria.

Hora	Ocupados	Disponibles	%Lleno	%Vacio
6- 7	0	10	0 %	100 %
7- 8	0	10	0 %	100 %
8- 9	0	10	0 %	100 %
9- 10	0	10	0 %	100 %
10- 11	0	10	0 %	100 %
11- 12	0	10	0 %	100 %
12- 13	0	10	0 %	100 %
13- 14	0	10	0 %	100 %
14- 15	0	10	0 %	100 %
15- 16	0	10	0 %	100 %
16- 17	0	10	0 %	100 %
17- 18	0	10	0 %	100 %
18- 19	0	10	0 %	100 %
19- 20	0	10	0 %	100 %
20- 21	0	10	0 %	100 %
21- 22	0	10	0 %	100 %
22- 23	1	9	10 %	90 %
GenerarPDF				
Generar Gráficas				

De acuerdo a la verificación mostrada en la tabla 17, se puede deducir:

- En efecto solo existió un parqueadero utilizado entre las 22 y 23 horas, el cual fue el A1. En las otras horas no se ocupó ningún estacionamiento.
- Se puede observar la opción de generar PDF. Al dar clic se generará el archivo respectivo. En la Figura 93, se observa el archivo generado a partir de la consulta realizada.



PARQUEADERO UDLA

Hora	#P. Ocupados	#P. Disponibles	% Total Lleno	% Total Vacio
6- 7	0	10	0 %	100 %
7- 8	0	10	0 %	100 %
8- 9	0	10	0 %	100 %
9- 10	0	10	0 %	100 %
10- 11	0	10	0 %	100 %
11- 12	0	10	0 %	100 %
12- 13	0	10	0 %	100 %
13- 14	0	10	0 %	100 %
14- 15	0	10	0 %	100 %
15- 16	0	10	0 %	100 %
16- 17	0	10	0 %	100 %
17- 18	0	10	0 %	100 %
18- 19	0	10	0 %	100 %
19- 20	0	10	0 %	100 %
20- 21	0	10	0 %	100 %
21- 22	0	10	0 %	100 %
22- 23	1	9	10 %	90 %

GENERADO POR: Admin Admin

Fecha: 16/10/2015 Hora: 13:31:59

Figura 93. Verificación de generación de archivo PDF.

- Se puede observar la opción de generar Gráficas. Al dar clic se generarán las gráficas respectivas. En la Figura 94, se observa el resultado de la generación de gráficas.

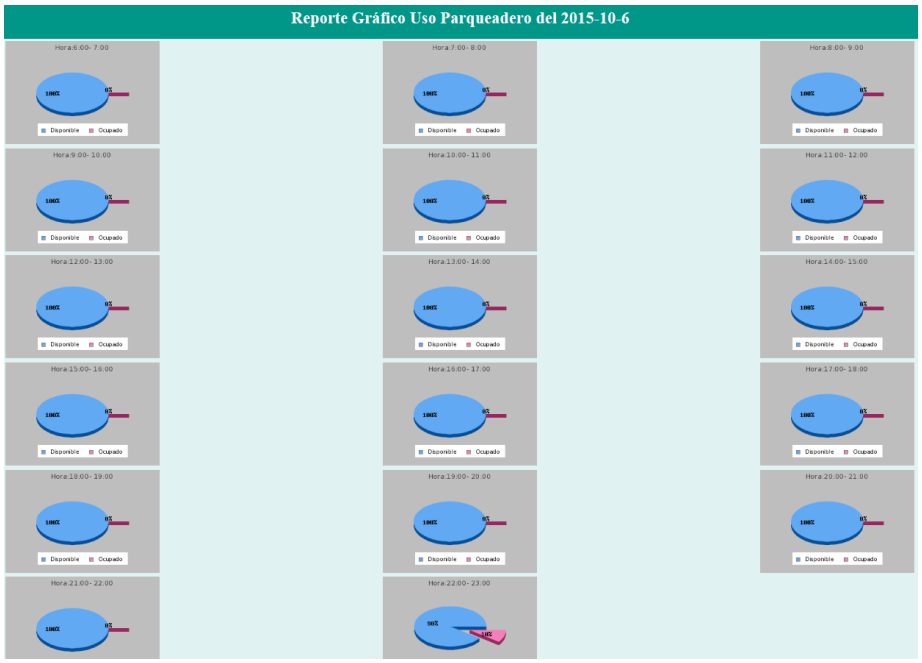


Figura 94. Verificación de generación de Gráficas porcentuales por día.

5.4.3 Prueba consulta, inserción, actualización y eliminación de parqueaderos.

- **Consulta de parqueaderos:**

Se verificará los dos tipos de consulta:

a) Consultar todos:

Como se muestra en la tabla 18 se muestran todos los parqueaderos existentes en la Base de Datos, con su estado respectivo. En la parte inferior se tiene la opción de generar PDF de la consulta realizada, la verificación de la generación del archivo PDF se puede observar en la Figura 95.

Tabla 18. Tabla verificación de consulta de todos los parqueaderos existentes.

Parqueadero	Estado
A1	Disponible
A2	Disponible
B1	Disponible
B2	Disponible
C1	Disponible
C2	Disponible
D1	Disponible
D2	Disponible
E1	Disponible
E2	Disponible

GenerarPDF

LISTA DE PARQUEADEROS

Id	Pos	Estado
1	A1	Disponible
2	A2	Disponible
3	B1	Disponible
4	B2	Disponible
5	C1	Disponible
6	C2	Disponible
7	D1	Disponible
8	D2	Disponible
9	E1	Disponible
10	E2	Disponible

GENERADO POR: Admin Admin

Fecha: 16/10/2015 Hora: 14:08:58

Figura 95. Verificación archivo de generación PDF consultar todos.

b) Consultar por sección:

Como se observa en la Figura 96, para realizar la consulta se debe escoger la sección a consultar, en este caso “A”. Una vez escogida la sección y realizada la consulta aparecen todos los parqueaderos de dicha sección. En la parte inferior se tiene la opción de generar PDF de la consulta realizada, la verificación de la generación del archivo PDF se puede observar en la Figura 97.

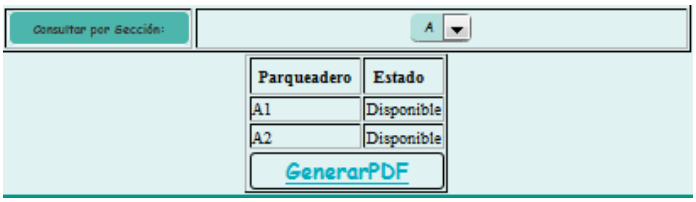


Figura 96. Verificación de consulta de parqueadero por sección

.

Figura 97. Verificación de generación de PDF consulta por sección.

- **Inserción de parqueaderos:**

Primeramente se verificará que la posición en la sección no pueda estar vacía al momento de realizar la inserción, mostrando el mensaje que se requiere el número en sección, como se muestra en la Figura 98.

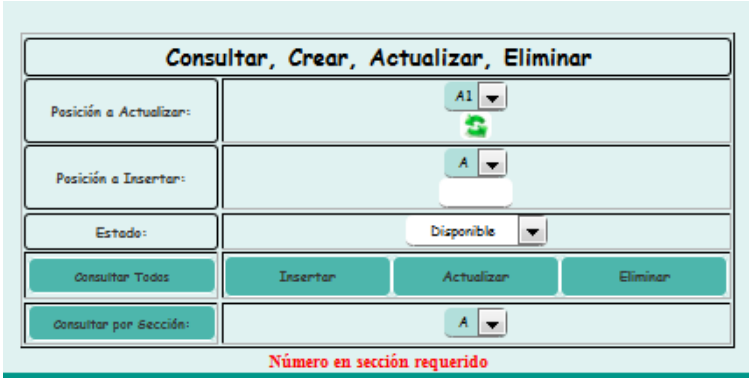


Figura 98. Verificación cuadro de texto posición vacío.

Ahora se verificará que no se pueda insertar un parqueadero ya existente, mostrando un mensaje de error, como se muestra en la en la Figura 99.

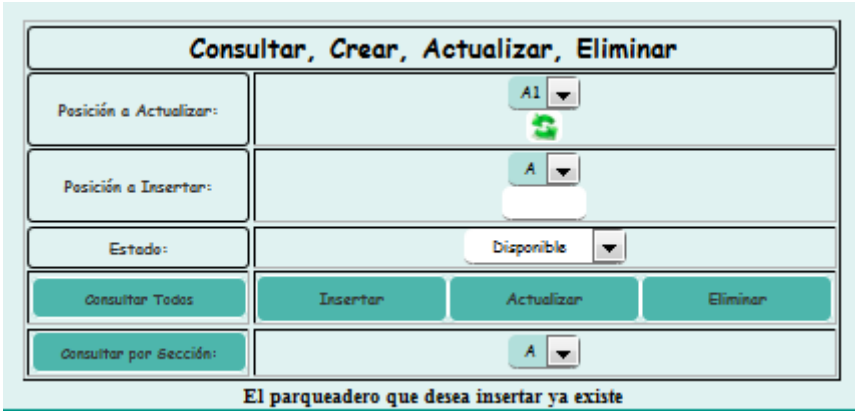


Figura 99. Verificación inserción parqueadero existente.

Finalmente se insertará un nuevo parqueadero no existente, con la posición G12, mostrando el mensaje de inserción correcta, como se muestra en la Figura 100.

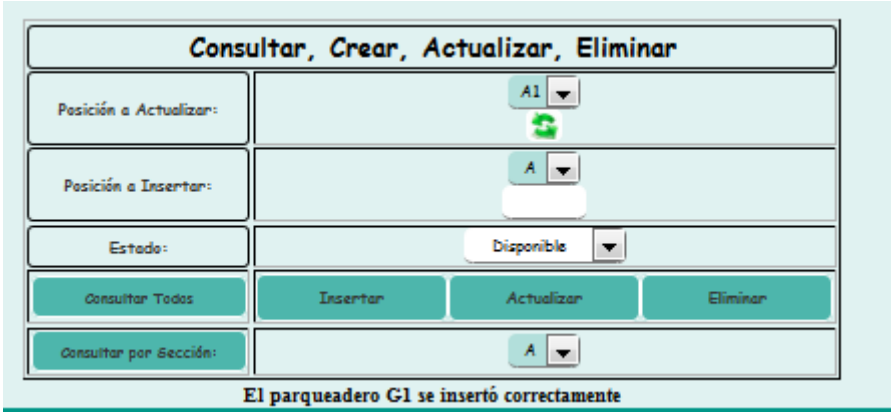


Figura 100. Verificación inserción parqueadero NO existente.

- **Actualización de estado de parqueaderos:**

Para la actualización la única validación que se ocupa es que el nuevo estado del parqueadero a actualizar sea diferente al que tiene previamente.

El estado del parqueadero A1 actualmente es *Disponible*, ahora se lo actualizará al mismo estado Disponible.

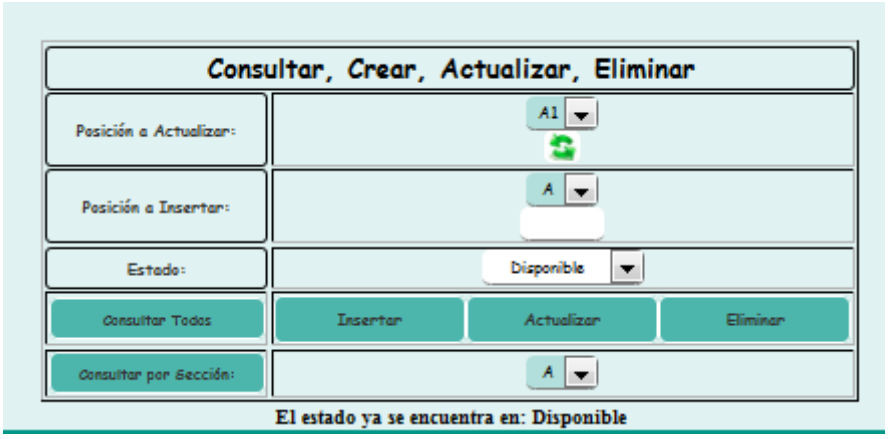


Figura 101. Verificación actualización de estado de parqueadero igual al actual.

Como se muestra en la Figura 101 se verifica que se muestre el mensaje de que el estado del parqueadero A1 ya se encuentra en estado *Disponible*.

Ahora se actualizará el estado del parqueadero A1 a un estado diferente al actual, en este caso el estado actual es *Disponible*, y se actualizará a un nuevo estado *Ocupado*.

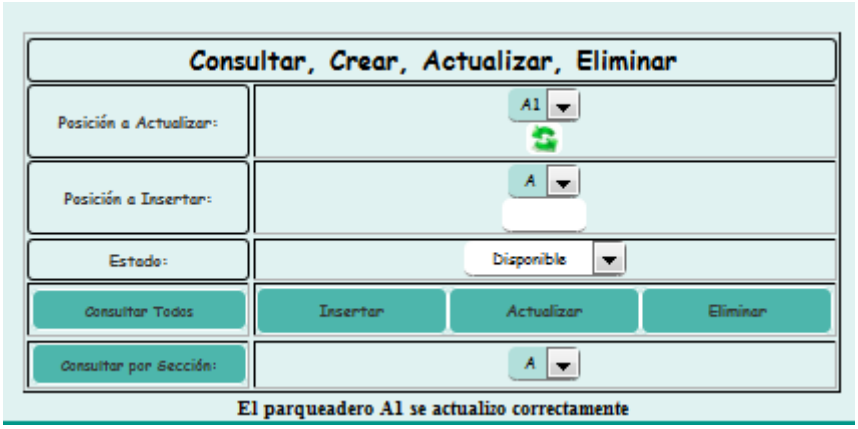


Figura 102. Verificación actualización de estado de parqueadero diferente al actual.

Como se observa en la Figura 102 se verifica que el estado del parqueadero haya sido actualizado correctamente, mostrando el mensaje correspondiente.

- **Eliminación de parqueaderos:**

Como se explicó anteriormente para la eliminación se deberá elegir un parqueadero del cuadro combinado. Se procede a eliminar el parqueadero creado anteriormente “G1”.

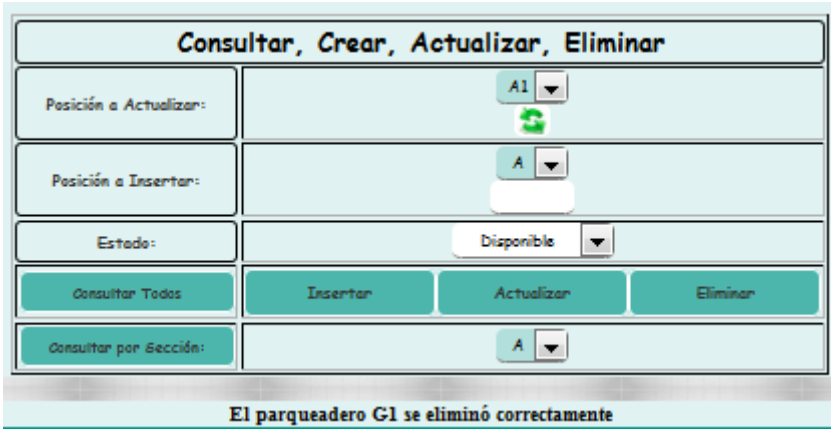


Figura 103. Verificación eliminación parqueadero

Como se muestra en la Figura 103 se verifica que el parqueadero haya sido eliminado con el mensaje de confirmación correspondiente.

5.4.4 Prueba Insertar Nuevo Usuario

Primeramente se verificará que los campos a insertar no puedan estar vacíos al momento de realizar la inserción, mostrando el mensaje correspondiente, como se muestra en las Figuras 104, 105, 106, 107, 108 y 109.

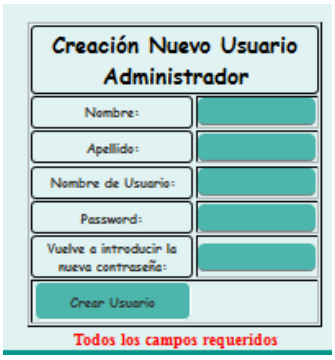


Figura 104. Verificación cuando no se ingresa ningún campo.

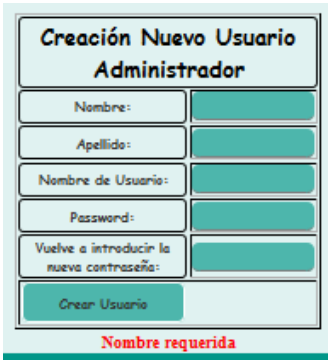


Figura 105. Verificación cuando no se ingresa el nombre.

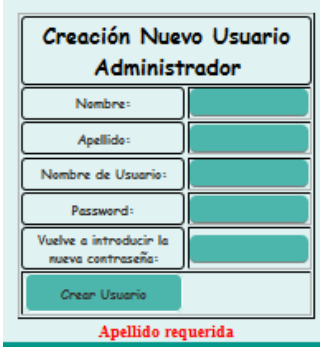


Figura 106. Verificación cuando no se ingresa el apellido.

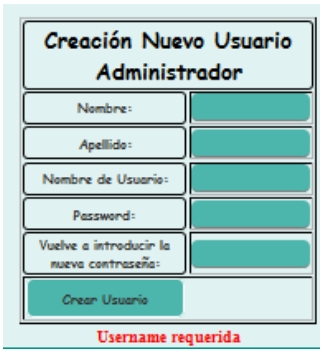


Figura 107. Verificación cuando no se ingresa nombre de usuario.

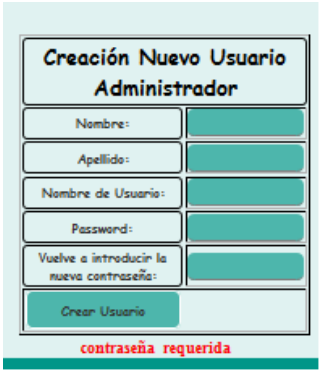


Figura 108. Verificación cuando no se ingresa contraseña.

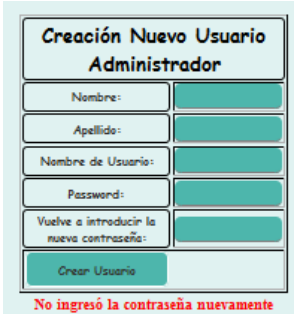


Figura 109. Verificación cuando no se ingresa la contraseña nuevamente.

Ahora se realizará la verificación en el caso de que las contraseñas ingresadas no coincidan mostrando el mensaje correspondiente como se muestra en la Figura 110.

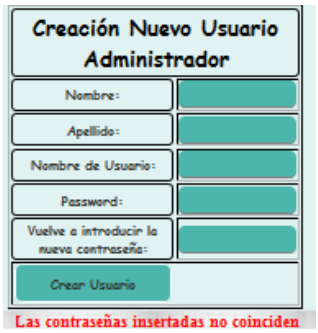


Figura 110. Verificación cuando las contraseñas no coinciden.

Adicionalmente se verificará que el nombre de usuario a ingresar no se repita con ningún usuario creado anteriormente. Se ingresa un nuevo usuario con un usuario ya existente y se comprueba que se muestre el mensaje correspondiente, como se observa en la Figura 111.

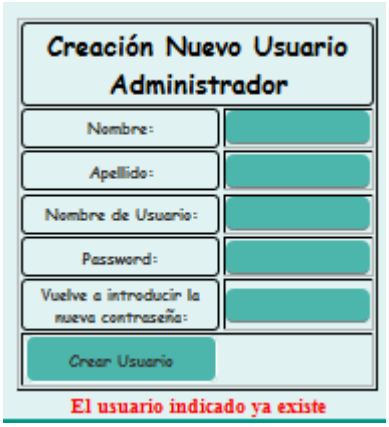


Figura 111. Verificación cuando el nombre de usuario ingresado ya existe.

Finalmente se insertará un nuevo usuario, llenando los datos correctamente y con un nombre de usuario que no haya sido creado anteriormente. Se verifica que el mensaje confirme la inserción como se muestra en la Figura 112.

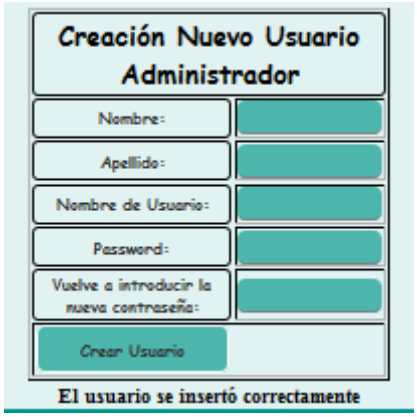


Figura 112. Verificación de inserción correcta de nuevo usuario.

5.4.5 Prueba Actualizar mi Usuario

Como se explicó anteriormente en esta interfaz se podrá actualizar la contraseña, nombre y apellido.

- **Actualización de contraseña**

Primeramente se verifica que los campos no puedan estar vacíos, como se muestra en las Figuras 113, 114, 115, 116.

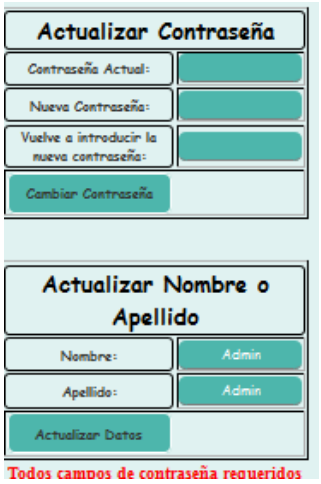


Figura 113. Verificación cuando no se ingresa ningún campo de contraseña.

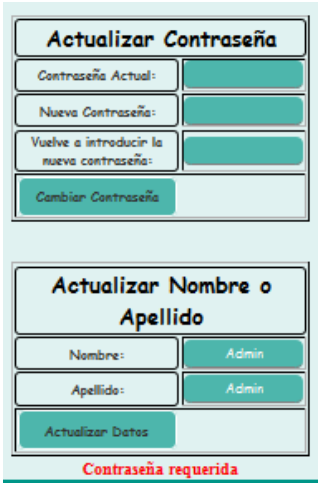


Figura 115. Verificación cuando no se ingresa la nueva contraseña.

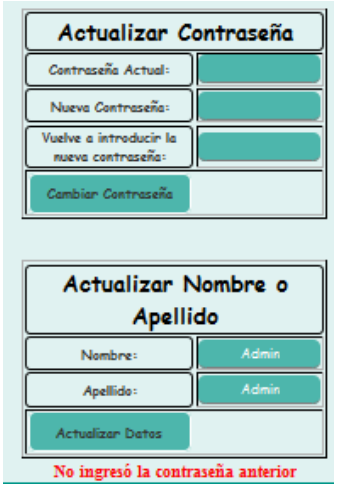


Figura 114. Verificación cuando no se ingresa la contraseña anterior.

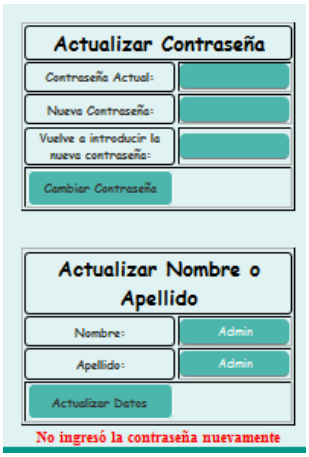


Figura 116. Verificación cuando no se vuelve a ingresar la nueva contraseña.

Adicionalmente se verifica que la contraseña actual corresponda a la del usuario que inicio sesión como se muestra. Se insertará una contraseña errónea, y se comprueba que aparezca el mensaje correspondiente, como se observa en la Figura 117.

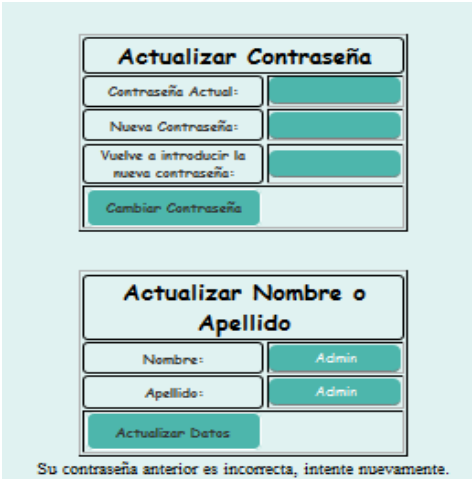


Figura 117. Verificación cuando se ingresa la contraseña actual errónea.

También se verifica la validación de que las nueva contraseñas coincidan; para esto se ingresará dos contraseñas diferentes, se mostrará el mensaje correspondiente como se muestra en la Figura 118.

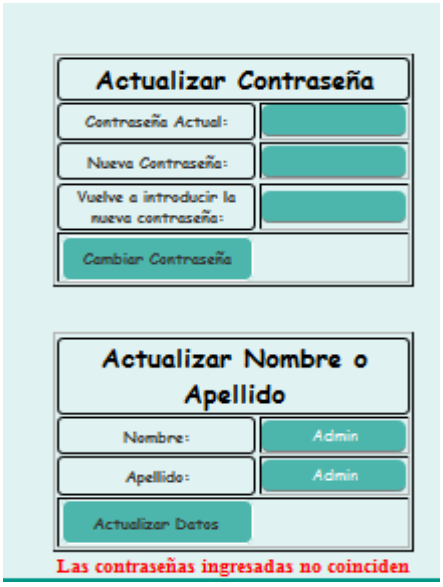


Figura 118. Verificación cuando no coinciden los dos campos de nueva contraseña.

Finalmente se realizará una actualización de contraseña ingresando los campos correctos. Se verifica que aparezca el mensaje correspondiente, como se muestra en la Figura 119.

Figura 119. Verificación de cambio correcto de contraseña.

5.8.1.1 Actualización de nombre y apellido

Para la verificación de la función de actualizar nombre y apellido, primeramente se comprobará que en los cuadros de texto aparezcan los datos actuales del usuario que inicio sesión. Como se muestra en la Figura 120 se comprueba que en efecto el usuario es Admin Admin.

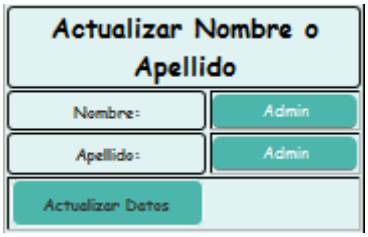


Figura 120. Verificación de cuadros de texto con datos de usuario.

Adicionalmente se verificará que los campos no puedan estar vacíos, como se muestra en las Figuras 121 y 122.

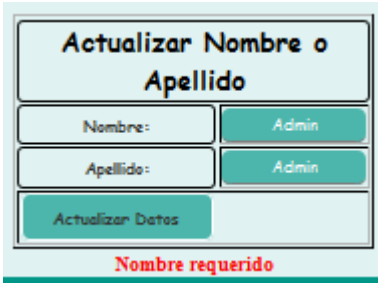


Figura 121. Verificación cuando no se ingresa nombre.

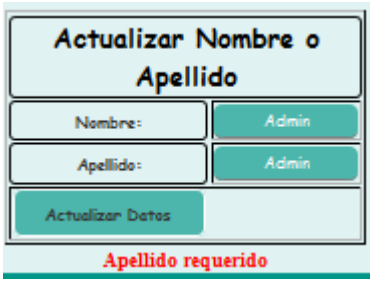


Figura 122. Verificación cuando no se ingresa apellido

Cuando no se llena un campo ya sea nombre o apellido y se realiza la actualización aparecerá el mensaje indicando que el campo no ingresado es requerido.

Finalmente se realizará la actualización de los datos del usuario Admin, se verifica que se haya efectuado el cambio con el mansaje correspondiente, como se muestra en la Figura 131.

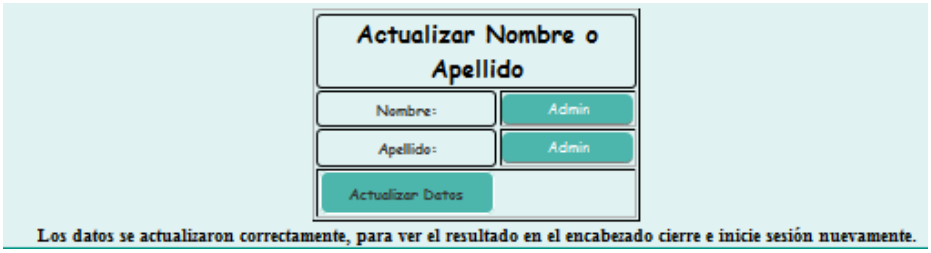


Figura 123. Verificación actualización de nombre y apellido de usuario.

5.5 Pruebas de Alimentación de Voltaje y Corriente al circuito:

Como se mencionó anteriormente la Fuente a utilizar es una Fuente externa que suple ~8V y una corriente 1A.

La fuente va a alimentar al Arduino Mega, y este alimentará a todos los sensores, Leds y al Shield Ethernet.

5.5.1 Medición Voltaje Entrada

Al medir el Voltaje de Entrada se comprobó que en efecto sea de ~8V, a continuación el valor obtenido:
_7,66 _V

5.5.2 Medición de Voltaje Salida Arduino

Al medir el Voltaje de Salida en el Arduino Mega, se constató que este regule el Voltaje de Entrada a 5V. A continuación el valor obtenido:
_4.97_V

5.5.3 Medición de Consumo Corriente Por Sensor

Los valores obtenidos son:

- Led emisor de Luz: V = 1.22V, R = 100Ω; I = 12.2Ma
- Fototransistor (Pin control en Colector) sin objeto reflejado: V = 4.70V,R = 4,3KΩ; I = 1mA
- Fototransistor (Pin control en Colector) con objeto reflejado: V = 0.25V,R = 4,3KΩ; I = 0.05mA

Total Corriente Mínima= _12.25_mA
Total Corriente Máxima= _13.2_mA
Total de Corriente Máxima por 10 sensores: 132 mA.

5.5.4 Medición de Consumo Corriente Por Led

Los valores obtenidos son:
V = 1.88V
R = 220Ω
I = _8.5_Ma
Total de Corriente Máxima por 10 Leds encendidos al tiempo: 85 mA.

Consumo Total del Circuito: ~132mA (Corriente por los 10 Sensores) + ~85mA (Corriente por 10 Leds) + ~150mA (Corriente Shield Ethernet) = **~367mA.**

CONCLUSIONES Y RECOMENDACIONES

Conclusiones:

- Antes de soldar los *Sensores* y *Leds*, se debe comprobar su correcto funcionamiento, ya que si se presentan errores, se descartará que el error se encuentre ya sea en el *Sensor* o *Led*.

- Si un *Led* no se enciende, se debe revisar las conexiones a la PCB, ya que quizás alguno de los cables (Tierra, Control-VCC) se encuentre roto, se deba cambiarlo o volver a soldarlo a la placa.

- Si un *Sensor* no cambia de estado, se debe revisar las conexiones a la PCB, ya que quizás alguno de los cables ya sea del *Led emisor de Luz* o el *Fototransistor* este roto, y se deba cambiarlo o volver a soldarlo.

- Se debe contar con una buena velocidad de conexión a Internet, esto permitirá que las peticiones HTTP Post se realicen rápidamente y el cambio de estado en cada parqueadero sea instantáneo; se recomienda una Velocidad mínima de 3Mb.

- Si no existe conexión a Internet, y el Arduino apunta al Servidor de Hosting, los datos no se van a actualizar. Ya que no se va a poder realizar la petición HTTP de tipo Post.

- Los Pines digitales que no se pueden utilizar son 50, 51, 52, 53, 4 y 10; debido a que estos son utilizados para la comunicación SPI entre Arduino Mega y Shield Ethernet.

- Los Pines de control de los sensores son conectados a entradas digitales, esto debido a que se manejarán solo dos estados (ocupado o disponible), esto va a permitir que se pueda incrementar el número de

parqueaderos utilizando codificadores para los sensores y decodificadores para los Leds respectivos.

- Este sistema de gestión para parqueaderos, se puede implementar en cualquier estacionamiento público. Los requerimientos son un punto de red con conexión a Internet.

- Otra solución válida, es la de conectar el Arduino al Internet a través de un Shield GSM, siguiendo el mismo principio de que para actualizar el estado de un parqueadero en la Base de Datos se debe realizar una petición HTTP de tipo POST o GET, según la programación.

Recomendaciones:

- Se recomienda utilizar *Software de Diseño Electrónico* para la elaboración de la *PCB*, para evitar errores en la construcción.

- Para la utilización del sistema como *Administrador*, se recomienda dar una explicación según lo indicado en el capítulo del *Desarrollo del Software*; con el fin de que cada Administrador conozca las funciones de cada *Botón* en la aplicación y así evitar errores en la operación del sistema.

- El punto de red a conectar debe estar a menos de 100 metros de distancia del Shield Ethernet, debido a lo establecido por el *Estándar IEEE 802.3*.

- Para la detección de errores se recomienda utilizar el método del descarte, ir analizando por partes el *Sistema* y así ir descartando el error en diferentes etapas, hasta encontrar donde se encuentra la falla.

- Se recomienda utilizar el Sensor Ultrasónico HC-SR04 para la implementación en sitio, mediante este sensor se podrá detectar presencia en un rango de 3cm a 3m. La distancia promedio entre el suelo y la parte de baja de un vehículo es de 40cm. Con programación se determinará que el sensor detecte presencia cuando un objeto se encuentre en un rango de 20cm a 40cm de distancia.

- Este sistema puede ser aplicado en cualquier parqueadero público, privado; se recomienda utilizar Shield Ethernet en lugares donde se pueda contar con un punto de Red. Si se da el caso de que no se disponga un punto de Red, se deberá optar por la opción de conectar el sistema al Internet mediante un Shield GSM utilizando la Red Celular, se debe considerar que se deberá contar con una SIM de un Operador que tenga mayor cobertura en el sitio.

REFERENCIAS

Alegsa (s.f).*Definición de Python (lenguaje de programación).* Recuperado el 1 de diciembre de 2014, de http://www.alegsa.com.ar/Dic/python.php

Alegsa (s.f.).*Definición de Visual basic*. Recuperado el 1 de diciembre de 2014, de http://www.alegsa.com.ar/Dic/visual%20basic.php

Androidos (s.f.). *Android OS Características*. Recuperado el 8 de diciembre de 2014, de http://androidos.readthedocs.org/en/latest/data/caracteristicas/

Aprender a programar (s.f.). *¿Qué es Java?.* Recuperado el 1 de diciembre de 2014, de http://www.aprenderaprogramar.com/index.php?option=com_content&id=368:ique-es-java-concepto-de-programacion-orientada-a-objetos-vs-programacion-estructurada-cu00603b&Itemid=188

Bueno Saber (s.f.). *Tipos de sensores de infrarrojos*. Recuperado el 1 de junio de 2014, de http://bueno-saber.com/aficiones-juegos-y-juguetes/ciencia-y-naturaleza/tipos-de-sensores-de-infrarrojos.php

Carpintero, G. (2010). *Sistemas embebidos*. Recuperado el 12 de diciembre de 2014, de http://ocw.uc3m.es/tecnologia-electronica/sistemas-embebidos-basados-en-fpgas-para-instrumentacion/material-de-clase-1/introduccion-a-sistemas-de-instrumentacion-embebidos

Carrera, A. (2008). *Arquitectura aplicación Web*. Recuperado el 1 de diciembre de 2014, de http://daw-fiec.pbworks.com/w/page/16963465/Arquitectura%20aplicaci%C3%B3n%20Web

Centro De Artigo (s.f.). *Frecuencias de Radio*. Recuperado el 17 de diciembre de 2014, de http://centrodeartigo.com/articulos-enciclopedicos/article_81531.html

Cross PCX (s.f.). *Definición de C*++. Recuperado el 1 de diciembre de 2014, de http://crosspcx.foroactivo.com.es/t36-definicion-de-c

Cuturacion (s.f). *Android: Principales características del sistema operativo de Google*. Recuperado el 17 de diciembre de 2014, de http://culturacion.com/android-principales-caracteristicas-del-sistema-operativo-de-google/

Garavito, O. (2009). *Características de Radio Frecuencia*. Recuperado el 13 de diciembre de 2014, de http://omar-gj.blogspot.com/2010/10/caracteristicas-de-radio-frecuencia.html

Java (s.f.). *¿Qué es Java?.* Recuperado el 1 de diciembre de 2014, de https://www.java.com/es/download/whatis_java.jsp

Java (s.f.). *Diferencias entre JavaScript y Java*. Recuperado el 15 de diciembre de 2014, de https://www.java.com/es/download/faq/java_javascript.xml

La Revista Informática (s.f.). *Lenguaje de Programación C#.* Recuperado el 18 de diciembre de 2014, de http://www.larevistainformatica.com/C1.htm

Lizarazo, T. (2013). *Universidades aportan al país con proyectos fuera de serie.* Recuperado el 10 de Septiembre de 2014, de http://www.eltiempzawo.com/archivo/documento/CMS-12623744

Molina, Y., Sandoval, J. & Toledo, S. (2012). *Sistema operativo Android: características y funcionalidad para dispositivos móviles*. Recuperado el 10 de diciembre de 2014, http://repositorio.utp.edu.co/dspace/bitstream/11059/2687/1/0053M722.pdf

Naranjo, K., Montoya, J., Tobón, M. & Visbal, D. (s.f.). *Sensor infrarrojo*. Recuperado el 1 de Junio de 2015, de http://server-die.alc.upv.es/asignaturas/PAEEES/2008-09/Sensor%20Infrarrojo%20-%20Grupo%20Naranja.pdf

National Instruments (s.f.). *Comunicación Serial: Conceptos Generales*. Recuperado el 8 de diciembre de 2014, de http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1#Serial

Nocedal, J. (2014). *RF Jamming*. Recuperado de Universidad de las Américas Puebla, el 13 de diciembre de 2014, de http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/nocedal_d_jm/capitulo_1.html#

Pergamino virtual (s.f.). *PEARL*. Recuperado el 1 de diciembre de 2014, dehttp://www.pergaminovirtual.com.ar/definicion/PERL.html

PHP Group (s.f.). *¿Qué es PHP?.* Recuperado el 1 de diciembre de 2014, de http://php.net/manual/es/intro-whatis.php

PHP Group (s.f.). *¿Qué puede hacer PHP?.* Recuperado el 1 de diciembre de 2014, de http://php.net/manual/es/intro-whatcando.php

Reyes, C. (2008). *Microcontroladores PIC*. Ecuador: Rispergraf.

Riyas, I. (s.f.). *Características de aplicaciones web*. Recuperado el 1 de diciembre de 2014, de http://estudiantealdeaunare3irmadj.blogspot.com/p/caracteristicas-de-aplicaciones-web.html

Ruby (s.f.). *Acerca de Ruby*. Recuperado el 1 de diciembre de 2014, de https://www.ruby-lang.org/es/about/

The last lab Project (s.f.). *Aplicaciones de los sensores*. Recuperado el 1 de diciembre de 2014, de http://thelastlabproject.blogspot.com/2010/12/aplicaciones-de-los-sensores.html

Úbeda, B. (2009). *Apuntes de: Sistemas embebidos*. Recuperado el 3 de diciembre de 2014, de http://ocw.um.es/ingenierias/sistemas-embebidos/material-de-clase-1/ssee-t01.pdf

Universidad Politécnica de Valencia (s.f.). *Las versiones de Android y niveles de API.* Recuperado el 10 de diciembre de 2014, de http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/146-las-versiones-de-android-y-niveles-de-api

Universidad Politécnica de Valencia (s.f.). *Sensor infrarrojo*. Recuperado el 3 de diciembre de 2014, de http://server-die.alc.upv.es/asignaturas/PAEEES/2008-09/Sensor%20Infrarrojo%20-%20Grupo%20Naranja.pdf

Xataka ciencia (s.f.). *Sensores infrarrojos*. Recuperado el 8 de diciembre de 2014, de http://www.xatakaciencia.com/tecnologia/sensores-infrarrojos

Zator Systems (s.f.). *Comunicación serie*. Recuperado el 8 de diciembre de 2014, de http://www.zator.com/Hardware/H2_5_1.htm

ANEXOS

ANEXO 1: Cuestionario Entrevista

ENTREVISTA AL ENCARGADO Y/O RESPONSABLE DE LA GESTIÓN DE LOS PARQUEADEROS EN LA UDLA

Nombre:

Cargo:

1. En términos generales, ¿Cuál es el proceso actual a seguir para el uso de los parqueaderos en la UDLA?

2. ¿Quién está permitido usar el parqueadero?

3. ¿Cuál es el total de personas habilitadas para usarlo?

4. ¿Cuántos vehículos ingresan en promedio al día?

5. ¿Cuál es la capacidad máxima del parqueadero?

6. ¿Cuáles son los ingresos mensuales receptados por el servicio de parqueadero?

7. Actualmente, ¿Cuál es la manera y/o forma utilizada para el control de la entrada/salida de vehículos?

8. A su criterio ¿Cuáles son los subprocesos importantes a considerar para la realización del prototipo a proponer?

9. Describa brevemente dichos subprocesos (considerar los Subsistemas propuestos –ingreso, cobro y control)

10. ¿Cree conveniente implementar un sistema de control e identificación vehicular aplicando tecnología móvil?

ANEXO 2: Cuestionario Encuesta

ENCUESTA DIRIGIDA A LOS USUARIOS DE LOS PARQUEADEROS DE LA UDLA

1. ¿Está satisfecho con el servicio prestado por los parqueaderos?

Si ()

No ()

2. ¿Qué es lo que más le disgusta de los parqueaderos actuales?

Servicio ()

Infraestructura ()

Organización ()

Disponibilidad ()

Otro ()

3. ¿Cuánto paga mensualmente por el servicio de parqueadero?

Menos de $40 ()

Entre $40 y $60 ()

Más de $60 ()

4. ¿Con qué frecuencia utiliza el parqueadero?

1 vez a la semana ()

Entre 2 y 3 veces a la semana ()

Entre 4 y 5 veces a la semana ()

Entre 6 y 7 veces a la semana ()

5. ¿Qué tiempo le toma el estacionar su vehículo?

Menos de 5 minutos ()

Entre 6 y 10 minutos ()

Más de 10 minutos ()

6. ¿Estaría dispuesto a hacer uso de un parqueadero automatizado que ofrezca seguridad, rapidez y exactitud?

Si ()

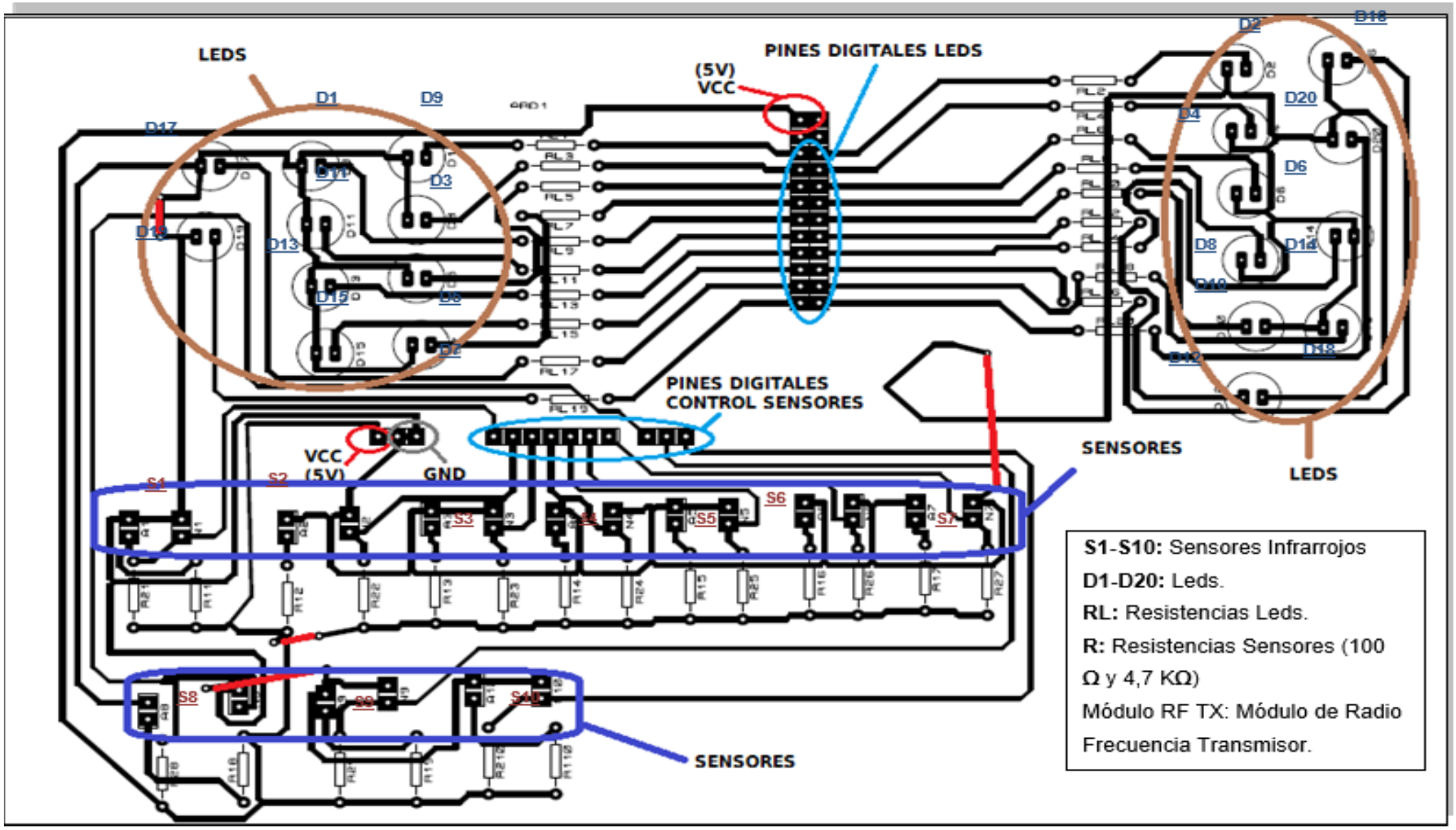
No ()

7. ¿Le interesaría usar una “Aplicación móvil” que le permita conocer la disponibilidad de espacios disponibles en el parqueadero de la UDLA?

Si ()

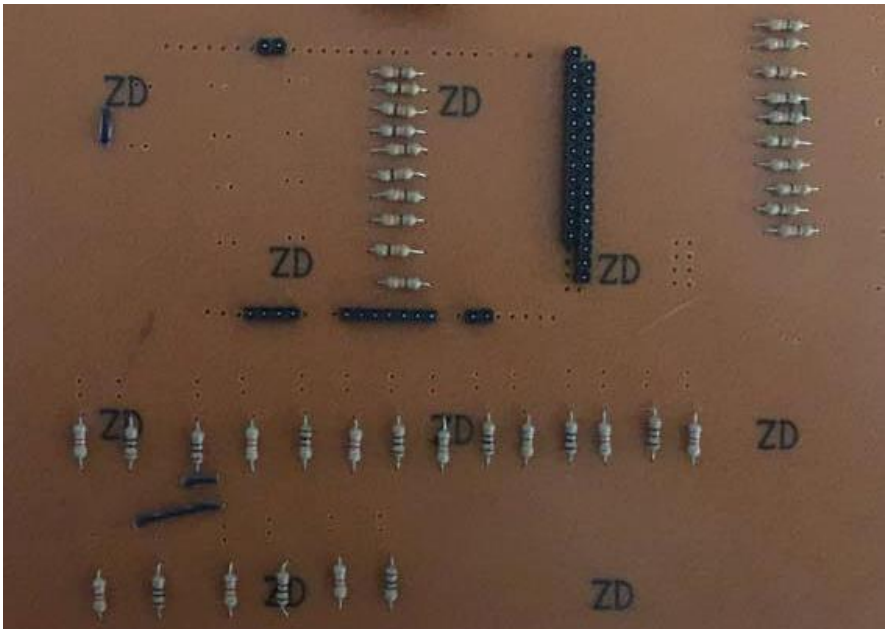
No ()

ANEXO 3: Diseño PCB para el sistema.



ANEXO 4: PCB del sistema.

ANEXO 5: PCB del sistema con componentes soldados.



ANEXO 6: PCB del sistema con componentes soldados y cables para Sensores y Leds.