Neiva, 16 de Febrero 2016

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

Johan Sebastián Pérez Gutiérrez con C.C. No.1.075.252.368, Maidy Xiomara Rojas Sánchez, con C.C. No. 1.075.257.252, autor(es) de la tesis y/o trabajo de grado titulado DISEÑO DE UN PROTOTIPO DE SISTEMA DE MONITOREO PARA DETECTAR LA DISPONIBILIDAD DE LOS PARQUEADEROS DEL CENTRO COMERCIAL SANTA LUCÍA PLAZA, presentado y aprobado en el año 2016 como requisito para optar al título de Ingeniero Electrónico; autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

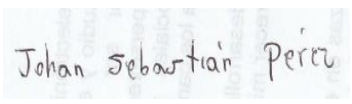
□□Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

• Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

• Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE:

Firma: 

EL AUTOR/ESTUDIANTE:

Firma: 

TÍTULO COMPLETO DEL TRABAJO: DISEÑO DE UN PROTOTIPO DE SISTEMA DE MONITOREO PARA DETECTAR LA DISPONIBILIDAD DE LOS PARQUEADEROS DEL CENTRO COMERCIAL SANTA LUCÍA PLAZA

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
Pérez Gutiérrez Rojas Sánchez	Johan Sebastián Maidy Xiomara

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Molina Mosquera	Johan Julián

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre
Martínez Mosquera Cerquera	Germán Vladimir

PARA OPTAR AL TÍTULO DE: Ingeniero Electrónico

FACULTAD: Ingeniería

PROGRAMA O POSGRADO: Electrónica

CIUDAD: Neiva **AÑO DE PRESENTACIÓN:** 2016 **NÚMERO DE PÁGINAS:** 101

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas___ Fotografías X Grabaciones en discos___ Ilustraciones en general___ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros__

SOFTWARE requerido y/o especializado para la lectura del documento:

MATERIAL ANEXO:

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	Inglés
1.	**Sensor**	**Sensor**
2.	**Supervisión**	**Monitoring**
3.	**Disponibilidad**	**Availability**
4.	**Arduino**	**Arduino**

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

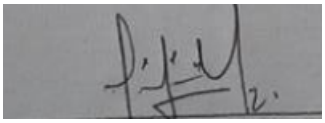
Este articulo presenta el diseño de un modelo de sistema de monitoreo, para detectar la disponibilidad de los parqueaderos del centro comercial Santa Lucía Plaza, mediante la implementación de un modelo arquitectónico con 20 plazas de parqueo y una interfaz web para que los usuarios puedan visualizar los estados de cada una y llevar un registro del tiempo de ocupación de manera remota. La lectura del estado de las plazas es realizada a través de sensores ultrasónicos, de esta manera se obtienen los datos que luego son procesados por medio de la tarjeta programable Arduino Mega ADK. Se establece una comunicación serial USB con un equipo de cómputo para que la información pueda ser transmitida en tiempo real a la página web mediante una aplicación en Java. Adicionalmente se implementa en el modelo arquitectónico una entrada automatizada en donde se realiza un conteo de los automóviles que ingresan al centro comercial, para ello se realiza una identificación con sensores infrarrojos evasores de obstáculos. Finalmente se ubica una talanquera móvil que a través de servomotores permite el paso de los vehículos.

ABSTRACT: (Máximo 250 palabras)

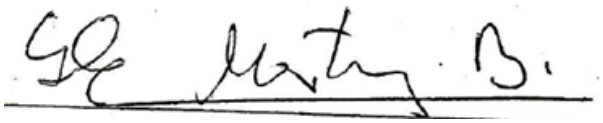
This article present the design of a monitoring system model, to detect a vacant parking in the mall Santa Lucia Plaza by implementing an architectural model with twenty parking spaces and a web interface to allow users to visualize the status of each and keep a record of the occupation on time remotely. The information of the vacant parking is made by ultrasonic sensors to obtain the data which will be processed by Arduino Mega ADK programmable card. The serial communication will be established by a USB with a computer to allow transmit the information into the web though a java app on real time. Additionally is implemented in an architectural model with an automated input where a count will made of cars who enter to the mall, for that is performed an identification of the obstacles with an infrared sensors. Finally it places an automatic car park barrier to allow the traffic flow.

APROBACION DE LA TESIS

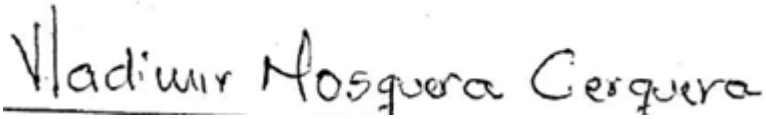
Nombre Presidente Jurado: Johan Julián Molina Mosquera

Firma: 

Nombre Jurado: Germán Martínez

Firma: 

Nombre Jurado: Vladimir Mosquera Cerquera

Firma: 

DISEÑO DE UN PROTOTIPO DE SISTEMA DE MONITOREO PARA DETECTAR LA DISPONIBILIDAD DE LOS PARQUEADEROS DEL CENTRO COMERCIAL SANTA LUCÍA PLAZA.

Autores:
JOHAN SEBASTIÁN PÉREZ GUTIÉRREZ
MAIDY XIOMARA ROJAS SÁNCHEZ

Universidad Surcolombiana
Facultad de Ingeniería
Proyecto Curricular de Ingeniería Electrónica
Neiva, Huila
2015

DISEÑO DE UN PROTOTIPO DE SISTEMA DE MONITOREO PARA DETECTAR LA DISPONIBILIDAD DE LOS PARQUEADEROS DEL CENTRO COMERCIAL SANTA LUCÍA PLAZA.

Autores:
JOHAN SEBASTIÁN PÉREZ GUTIÉRREZ
MAIDY XIOMARA ROJAS SÁNCHEZ

Trabajo de grado para optar el título de:
INGENIERO ELECTRÓNICO

Director:
ING. JOHAN JULIÁN MOLINA MOSQUERA

Universidad Surcolombiana
Facultad de Ingeniería
Proyecto Curricular de Ingeniería Electrónica
Neiva, Huila
2015

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

Firma del director de Proyecto de Grado

_____

Firma del jurado

_____

Firma del jurado

DEDICATORIA

“Este trabajo de grado se lo dedico a Dios por darme esta vida tan maravillosa y llenarme de fortaleza para sobrepasar todos los obstáculos que se me presentaron, a mis padres, Daniel Pérez Alarcón y Clara Victoria Gutiérrez,porque me brindaron durante todo este camino su apoyo económico y moral, porque en los momentos más difíciles me ofrecieron su compresión y cariño, me animaron a seguir luchando para cumplir mis objetivos e hicieron de mí una mejor persona.

A mis hermanos Jiro Daniel y Carolina, a mis compañeros de estudio que aportaron su granito de arena para mi desarrollo académico y humano, a mi novia Ana María que durante este último año fue mi mayor motivación para trabajar duro por mis sueños, su amor y apoyo me permitieron culminar satisfactoriamente este proceso.”

JOHAN SEBASTIÁN PÉREZ GUTIÉRREZ

A Dios quien me dio fe, fortaleza, salud, sabiduría y esperanza para culminar este proyecto. A mi querida madre Deicy Sánchez porque creyó en mi e hizo el esfuerzo para sacarme adelante, dándome ejemplos dignos de superación y entrega, porque en gran parte gracias a ella hoy puedo ver alcanzada mi meta, ya que siempre estuvo impulsándome en los momentos más difíciles de mi carrera, y porque el orgullo que siente por mí fue lo que me hizo seguir hasta el final.

A mis familiares, amigos y compañeros de estudio porque siempre estuvieron en momentos difíciles y tuvieron la voz de aliento dándome fuerzas para no decaer y por el contrario continuar con el gran propósito.

MAIDY XIOMARA ROJAS SÁNCHEZ

AGRADECIMIENTOS

Agradecemos de manera especial a nuestros padres que han sido el motor de nuestras vidas y un apoyo incondicional para la culminación de este proyecto, al programa de Ingeniería Electrónica de la Universidad Surcolombiana, por el apoyo académico que nos brindaron en todo este proceso. También agradecemos a cada uno de nuestros compañeros que nos acompañaron, aconsejaron y ofrecieron su ayuda durante estos años de estudio.

Ofrecemos un agradecimiento sincero al Ingeniero Johan Julián Molina Mosquera, director de nuestro proyecto de grado por las asesorías brindadas para cumplir este objetivo, a los Ingenieros German Martínez y Vladimir Mosquera por la colaboración que nos prestaron a la hora de presentar el anteproyecto.

Por último, queremos agradecer a la constructora Santa Lucía, por hacernos participe de su proyecto de construcción del centro comercial Santa Lucía Plaza y por todos los aportes que nos brindaron.

TABLA DE CONTENIDO

LISTA DE FIGURAS.

LISTA DE TABLAS

LISTA DE ANEXOS

RESUMEN

Este proyecto de grado presenta el diseño de un modelo de sistema de monitoreo, para detectar la disponibilidad de los parqueaderos del centro comercial Santa Lucía Plaza, mediante la implementación de un modelo arquitectónico con 20 plazas de parqueo y una interfaz web para que los usuarios puedan visualizar los estados de cada una.

El documento se divide principalmente en 3 secciones, la primera explica detalladamente el funcionamiento del sistema de detección de disponibilidad, datos de los sensores utilizados, especificaciones y programación de la tarjeta utilizada para el procesamiento de los datos.

La segunda sección hace un análisis del protocolo de comunicación implementado, sus características y la forma como se transmiten los datos desde la tarjeta al equipo de cómputo y de este a la interfaz web. Se tienen en cuenta dos formas de transmisión, una física y otra virtual.

En la tercera sección se explica todo lo relacionado con la configuración del servidor web y la base de datos utilizada para guardar la información requerida, el diseño de la interfaz web y el desempeño completo de la página creada con cada una de sus funciones.

1 GENERALIDADES

No es un secreto que la composición del parque automotor de la ciudad de Neiva ha aumentado notablemente. Como ciudadanos podemos observar que cada vez es más grande la cantidad de vehículos que se transportan por la ciudad, de esta manera se hace mucho más complicado para el dueño del vehículo encontrar parqueaderos disponibles, especialmente en los centros comerciales.

Este problema no solo se ve reflejado dentro de las instalaciones de dichos lugares, sino también en la movilidad de las calles aledañas, provocando un embotellamiento vehicular y afectando la calidad de vida de las personas. Esta dificultad se puede encontrar en diferentes puntos de la ciudad de Neiva, al igual que en algunos centros comerciales, de manera más común en el San Pedro Plaza, donde la espera para ubicarse en un parqueadero sobre todo en las horas de mayor flujo vehicular es agobiante.

Observando los antecedentes anteriormente mencionados, se propone en este proyecto de grado un modelo de monitoreo para un centro comercial que está en etapa de construcción, como lo es el Santa Lucía Plaza, con el fin de evitar todos estos inconvenientes y de esta manera no solo se mejora la movilidad del lugar, sino también en la ciudad en forma general. Actualmente es común observar en las grandes ciudades del país, tecnologías implementadas que permiten el monitoreo y control de los parqueaderos, por lo general los datos obtenidos solo quedan para la propiedad del establecimiento y no es compartido con los usuarios o ciudadanos en general. La página web diseñada en este proyecto de grado, es una gran alternativa para que los habitantes desde cualquier parte de la ciudad estén enterados del flujo vehicular del parqueadero y de la disponibilidad de las plazas en tiempo real.

Con este proyecto se desea comunicar los datos de cada una de las plazas a los usuarios que accedan a la página, a través de una red de sensores y un protocolo de comunicación, de esta manera se brinda al cliente comodidad y facilidad para que pueda acceder al centro comercial de la mejor manera.

1.1 Planteamiento del problema

En la actualidad podemos observar cómo los centros comerciales son más importantes en la vida diaria de los ciudadanos, el crecimiento reflejado en la oferta de servicios en estos establecimientos se ha dado como respuesta a la prominente demanda de los mismos, procedente de los consumidores que día a día requieren que los Centros Comerciales les ofrezcan todo en un mismo lugar.

Las personas buscan cada día localizar toda la oferta de servicios lo más cerca posible de sus hogares, para dedicar más tiempo al desarrollo de sus vidas y cada vez menos tiempo a transportarse. La rutina de visitar los Centros Comerciales, ya no es cosa de solo un fin de semana, sino una costumbre diaria debido a que allí se encuentra desde lo más común a lo más excepcional, partiendo de algo básico como: alimentos y servicios de belleza, hasta las formas de entretenimiento más avanzadas que existen en la actualidad, siendo éstos los rubros que con mayor fuerza y rapidez han crecido en el sector.

En los últimos años el aumento de vehículos en las calles colombianas ha incrementado notablemente, llegando a tal punto que encontrar un lugar para estacionar un auto es todo un problema, sobre todo en las plazas de aparcamiento de centros comerciales. Debido al crecimiento del parque automotor, el problema del tráfico es una realidad que se enfrenta día a día y con ella la falta de espacio dónde estacionar dicho vehículo; razón por la cual, el sector público y privado se ha visto en la obligación de acoger un conjunto de acciones dentro de las cuales se aprecian medidas frente al problema de falta de parqueaderos y mejoramiento de la eficiencia de los ya existentes.

La inadecuada organización de los parqueaderos en los centros comerciales impide el cumplimiento de un objetivo básico, como lo es brindar comodidad a las personas para ingresar a las instalaciones y poder disfrutar de sus beneficios; aunque de ante mano, el ingresar a dichos espacios produce un estancamiento en la movilidad vehicular, el problema se empeora cuando los usuarios luego de una larga espera al momento de llegar a la entrada, se encuentran con la noticia de que el parqueadero está lleno, uno de los factores que más contribuye en dicho aumento de tiempo es el problema de que los conductores de los diferentes medios de transporte motorizados, como los automóviles y las motocicletas, no encuentran un lugar fácil para situarse y los usuarios deben emplear cantidad de tiempo buscando un sitio disponible para poder parquear sus automóviles. Otro factor importante que debemos tener en cuenta es el consumo de combustible, el cual por cada periodo

de espera aumenta; si buscamos que los ciudadanos recorran distancias menores, podemos generar un ahorro de dinero a los usuarios de automóviles y a su vez evitar un mayor desgaste de éstos.

Por lo tanto, con el propósito de facilitar dicho proceso, es necesario el desarrollo de un plan estratégico utilizando herramientas electrónicas, las cuales brindarán comodidad y eficiencia a la hora de hacer el proceso de parqueo para que todos los clientes queden satisfechos a la hora de visitar el establecimiento comercial.

2. JUSTIFICACIÓN

Las herramientas electrónicas han entrado de lleno a nuestra vida cotidiana, facilitando el accionar de las personas que las utilizan, lo que a su vez ha llevado a que se conviertan en un pilar básico para el desarrollo de la sociedad.

La comodidad y calidad es algo que buscamos en todos los servicios que utilizamos, por esta razón es importante el desarrollo de diferentes proyectos que atiendan a las necesidades y ayuden a las labores cotidianas de las personas, como es el caso de este proyecto, el cual por medio de herramientas de hardware y software interconectadas entre sí, pueden ayudar a la población en general a una actividad, que cada vez se hace más complicada como lo es encontrar de manera fácil un lugar para aparcar los vehículos en un centro comercial en tiempo real, informando la cantidad de plazas de aparcamiento disponibles y el lugar en donde estas se encuentran, logrando disminuir notablemente el tiempo de búsqueda en el parqueadero y ahorrar dinero en combustible.

El proyecto pretende generar un beneficio muy grande de movilidad para la ciudad, debido a que se pueden evitar los trancones que se originan en las calles aledañas a los centros comerciales a consecuencia de los largos tiempos de espera que tienen que soportar los conductores para poder ingresar a dichos establecimientos por el caos vehicular que se genera dentro de los parqueaderos. Cabe destacar que la solución planteada en el proyecto, lleva a una disminución notable del estrés en algunas personas que se desesperan realizando la labor de parqueo, y si logramos más tranquilidad en los clientes originamos también un método de prevención para futuros accidentes en la disputa por ocupar una plaza libre.

En el contexto ambiental, si reducimos los tiempos en el que los carros circulan por el parqueadero estaremos aportando una gran ayuda en la disminución de las emisiones de gases de efecto invernadero, que tanto están afectan a la atmosfera y estará brindando un aire menos contaminado a las personas que circulan en dicho lugar. Es importante resaltar, que no solo se disminuye la contaminación ambiental sino que también los índices de contaminación auditiva por el ruido de motores y silbatos, se verán reducidos evitando efectos nocivos fisiológicos y psicológicos en los usuarios.

3. OBJETIVOS

3.1 Objetivo General

- Diseñar el modelo de un sistema de monitoreo para detectar la disponibilidad de los parqueaderos del centro comercial Santa Lucía Plaza, con el fin de facilitar el acceso y disminuir los tiempos de búsqueda de las plazas de parqueo.

3.2 Objetivos Específicos

- Diseñar el modelo arquitectónico a escala de 1:25, con 20 puestos de parqueo.

- Diseñar un modelo a base de sensores ultrasónicos, un microcontrolador y un protocolo de comunicación, con el fin de recolectar el número de plazas libres y ocupadas en el parqueadero del centro comercial Santa Lucía Plaza.

- Identificar los protocolos de comunicación para el monitoreo en tiempo real.

- Establecer un servidor de base de datos con el fin de organizar y recolectar toda la información del parqueadero obtenida por el dispositivo.

- Implementar una interfaz donde se pueda visualizar la cantidad de parqueaderos libres y determinar fácilmente su ubicación.

- Identificar y evaluar el desempeño del modelo electrónico de control y monitoreo del sistema de parqueo.

4. ALCANCES Y LIMITACIONES

4.1 Alcances

Con la ejecución de este proyecto que se desea desarrollar, se podrá cubrir gran parte de las necesidades de los usuarios del centro comercial Santa Lucía Plaza, por factores de ingreso y movilidad en el parqueadero; igualmente se podrá obtener un sistema de monitoreo en las plazas de parqueo a través de sensores y un sistema de comunicación que nos informará la cantidad y ubicación de los lugares disponibles.

Se permitirá el almacenamiento de toda la información proveniente de los sensores, en un servidor de base de datos, lo que hará posible un mejor registro y control de esta para el administrador del parqueadero.

El software y la interfaz que se desarrollará, permitirá a los clientes visualizar el área del parqueadero y de esta manera identificar los lugares de los que dispone el lugar para ubicar los vehículos, disminuyendo los tiempos de espera y el gasto de combustible.

Al finalizar el modelo, se dispondrá de una gran herramienta que ofrecerá comodidad y sofisticación al parqueadero del Centro Comercial, convirtiéndose en el primero en la ciudad de Neiva en implementar este tipo de tecnología.

4.2 Limitaciones

La limitación de lo planteado es una opción para clarificar y poder tomar la decisión si se desarrolla o no el proyecto, teniendo en cuenta que si se lleva a cabo existen cambios del modelo presentado a la realidad.

5. DETECCIÓN DE LA DISPONIBILIDAD DE LAS PLAZAS DE PARQUEO

Para empezar con el desarrollo del sistema que se encarga de la detección de disponibilidad, se deben tener en cuenta ciertos elementos básicos y necesarios que se van a describir a continuación, que contribuyen a que el modelo tenga una correcta implementación y su funcionamiento sea el más adecuado, con el fin de satisfacer todas las necesidades requeridas.

La construcción del hardware se ha distribuido principalmente en tres etapas:

- Etapa de sensado
- Etapa de procesamiento de datos
- Etapa de comunicación 1

Cabe aclarar que el proceso de comunicación del sistema en general se ha dividido en dos partes: la primera que es la que nombramos en este apartado como comunicación 1, hace referencia al enlace físico que se realiza entre el controlador y el equipo de cómputo, por medio del cable USB. La segunda parte es la transmisión de datos, que se hace desde un aplicativo en java hacia la interfaz web donde se va a visualizar el sistema de parqueo.

A continuación, se mostrará un diagrama de bloques donde se explica de manera general y clara el funcionamiento del sistema y la interacción de los diferentes componentes físicos que hacen posible cada etapa.

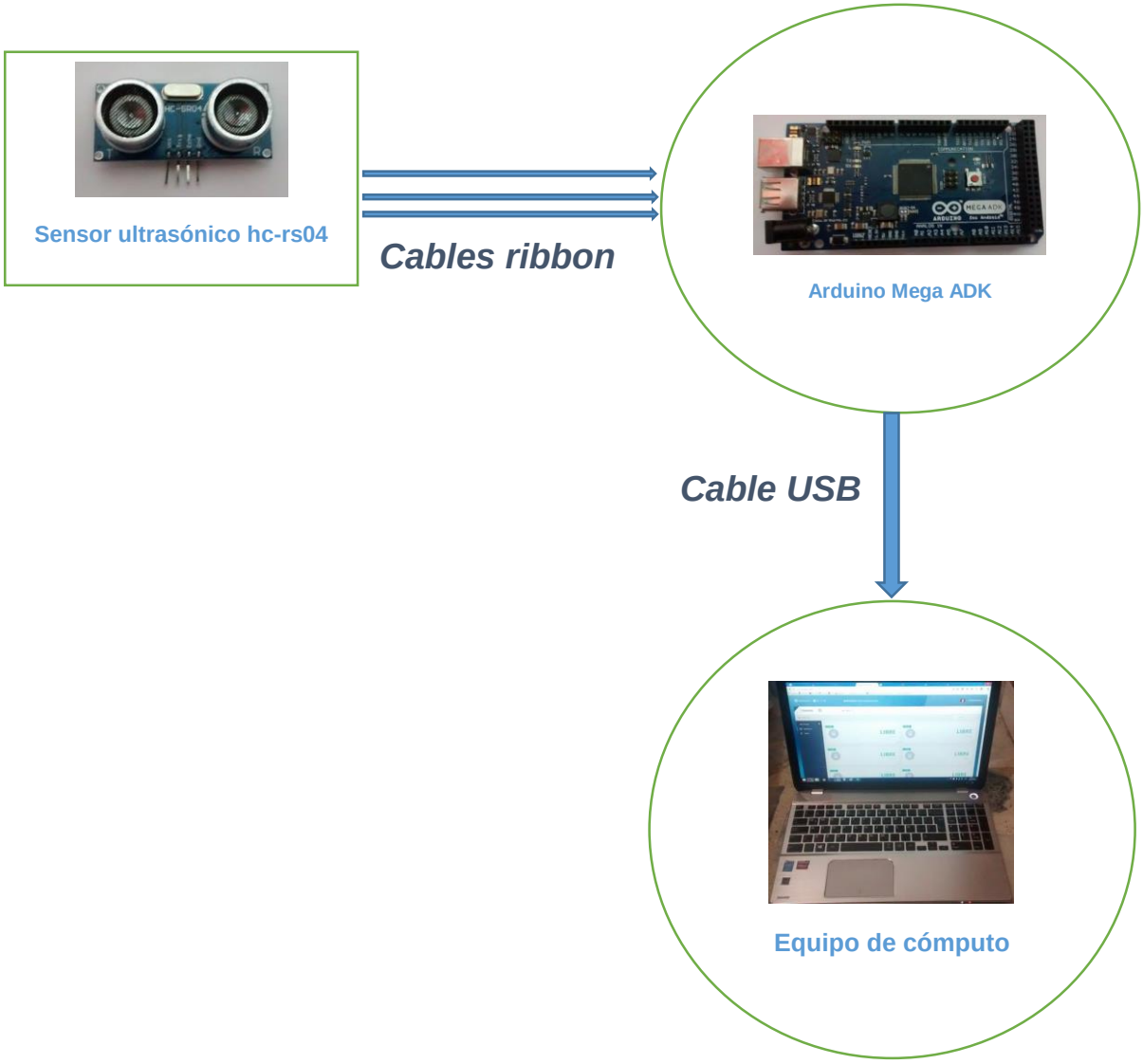


Figura 1.Sistema diagrama de detección Sensor ultrasónico HC-RS04

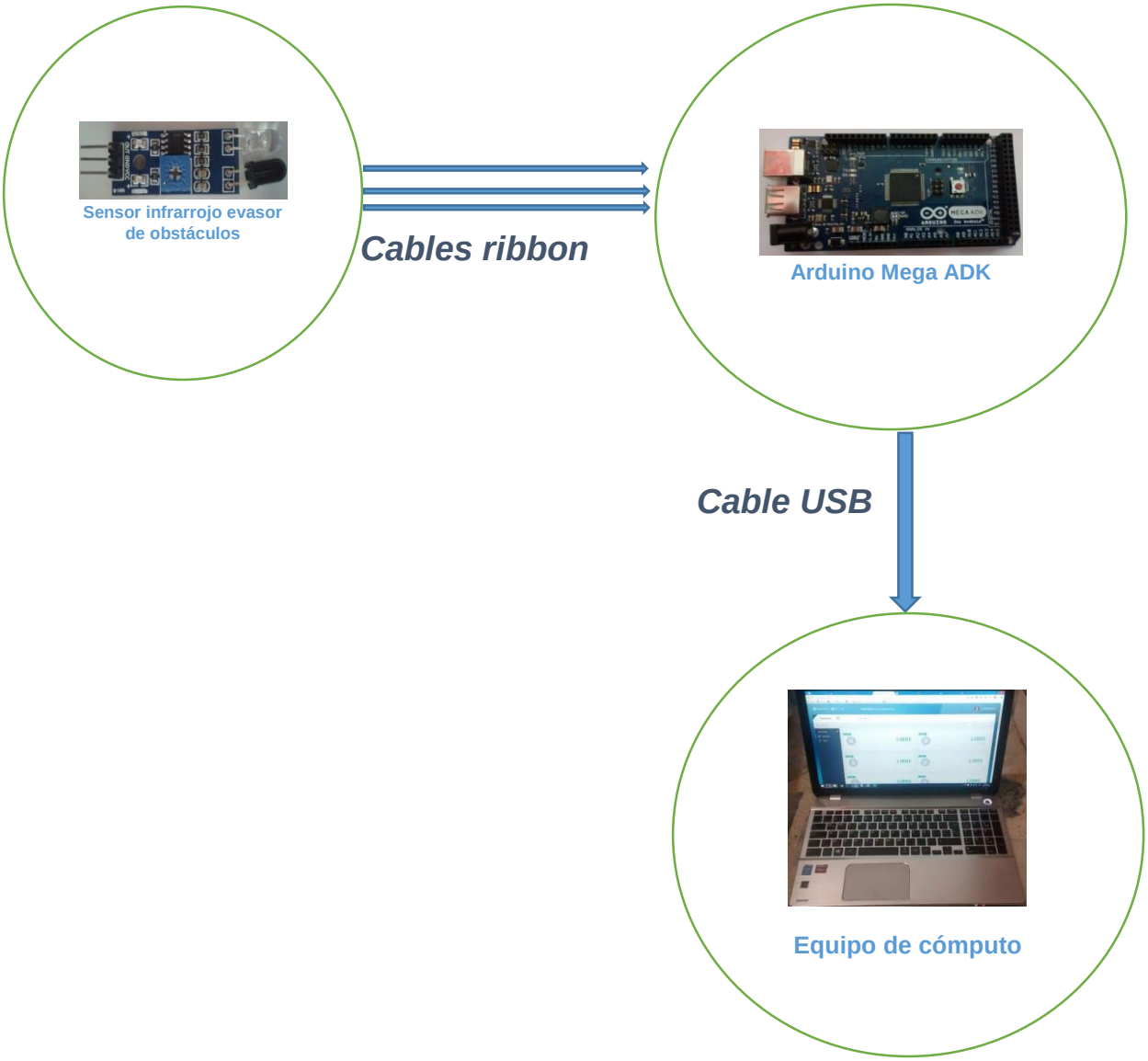


Figura 2.Sistema diagrama de detección Sensor infrarrojo evasor de obstáculos

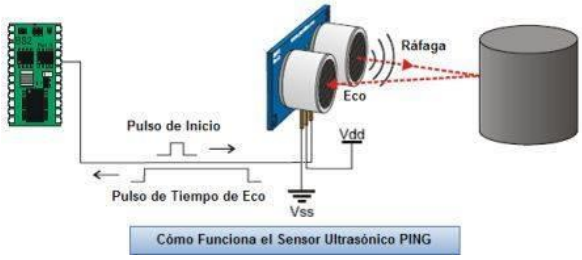
Para realizar el modelo de detección, se tuvo en cuenta que este cumpliera con algunas características primordiales para asegurar su buen funcionamiento. Un factor muy importante, era la fiabilidad bajo las condiciones ambientales críticas de un parqueadero y así evitar la menor perdida de datos. Otro factor, era manejar la menor cantidad de elementos para ahorrar costos, pero que a su vez cumplieran perfectamente con la labor de detección y comunicación en tiempo real.

Como se puede observar en la Figura 1 y Figura 2, el primer proceso es el de sensado, que se lleva a cabo por medio de sensores ultrasónicos hc-rs04 y sensores infrarrojos evasor de obstáculos, distribuidos en 20 plazas de parqueo, los datos obtenidos son transportados por medio de cables ribbon hacia la tarjeta programable Arduino Mega ADK, para que sean procesados y acondicionados a los requerimientos del sistema, esto se logra a través del microcontrolador ATMEGA 2560, teniendo en cuenta que en el parqueadero real se va a utilizar cable coaxial, Por último se realiza la comunicación de la tarjeta programable al equipo de cómputo por medio de un bus serie universal (USB).

5.2 Sensor ultrasónico HC-SR04

Para detectar los estados de cada una de las plazas de parqueo presentes en el modelo arquitectónico, se implementaron sensores de proximidad ultrasónicos de referencia hc-rs04. Son sensores que se basan primordialmente en la emisión de ondas ultrasónicas, que ayudan a determinar la distancia a un objeto, presentando una analogía con algunos animales de la naturaleza como lo son los murciélagos y delfines.

Figura 3.Sensor ultrasónico HC-SR04



[1]

5.2.1 Funcionamiento.

El sensor hc-rs04 cuenta con 4 pines:

Pines	Descripción
VCC	Alimentación del sensor
TRIG	Entrada del sensor (TTL)
ECHO	Salida del sensor (TTL)
GND	Tierra del sensor

Tabla 1.*Pines sensor hc-rs04*

Para que el sensor hc-rs04 se active y empiece a realizar la medición, el TRIG debe recibir un pulso en alto equivalente a 5V durante un tiempo mínimo de 10us, esto produce que el sensor transmita 8 pulsos a 40 KHZ, a lo que se le denomina el ultrasonido. En el instante en que el sensor transmite el ultrasonido, su salida ECHO se pone en alto a la espera de que la onda sea reflejada por el obstáculo. Cuando el sensor vuelve a recibir la onda reflejada, inmediatamente el estado de su pin ECHO se pone en bajo y finaliza el conteo de tiempo.

En el momento de realizar los cálculos para determinar la distancia, se debe tener en cuenta primero que la distancia que recorre la onda es el doble de la del objeto y segundo se debe tener presente la velocidad del sonido que es equivalente aproximadamente a 340m/s.

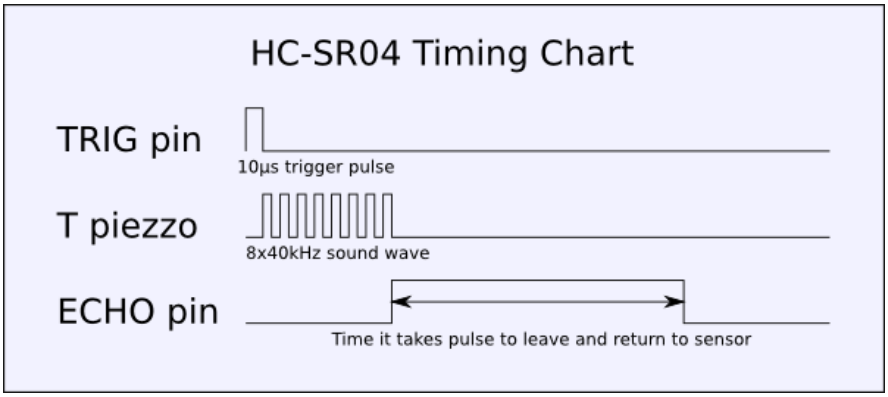


Figura 4.Funcionamiento de la pines [1]

5.2.3 Inconvenientes y limitaciones.

En el momento que el sensor envía una señal de ultrasonido desde el transmisor, se debe tener en cuenta que el campo que se genera es similar a la figura de un cono. El eco que llega al receptor como respuesta a la reflexión de la onda, señala primordialmente el obstáculo más cercano que se encuentra dentro del cono acústico, pero en realidad no señala en ningún instante en que lugar se encuentre el mismo.

Es cierto que por lo general el sensor va a detectar el objeto que se ubique en el medio del campo generado, pero existe la probabilidad que en algún momento la reflexión sea producida por un objeto que se encuentra en la periferia del campo de acción, produciendo una lectura errónea. Es primordial tener en cuenta este aspecto para tratarlo cuidadosamente.

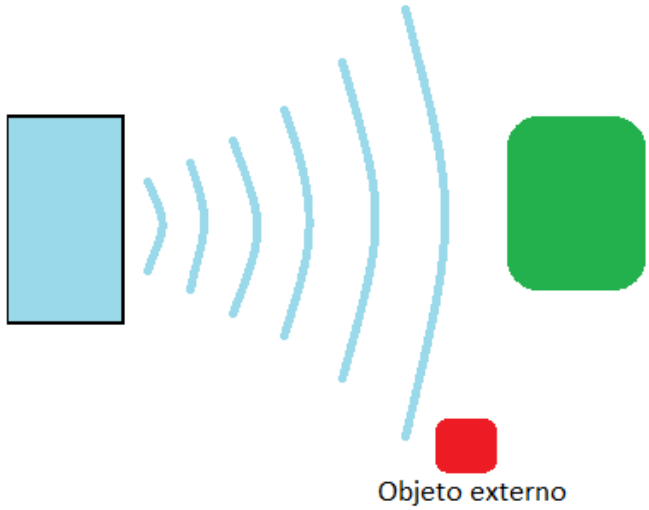


Figura 5. Problema por presencia de objeto en la periferia

Cuando se trabaja con sensores ultrasonidos en espacios con una distribución muy reducida como es el caso de este proyecto, se presenta un problema con la reflexión de las ondas, es decir se produce un fenómeno de ecos falsos. Unas de las razones por la que se genera este acontecimiento, es por la presencia de muchas superficies rodeando el sensor, entonces la señal emitida por el transmisor golpea en varias superficies antes de retornar al receptor. El problema con esta anormalidad, se ve

reflejado en la lectura de los datos, debido a que el sensor siempre va a medir el tiempo en que la onda se demoró en llegar y esta al realizar la reflexión en varias superficies, va a arrojar una distancia mucho mayor a la que realmente se encuentra el objeto más cercano, que fue el que probablemente produjo la reflexión.

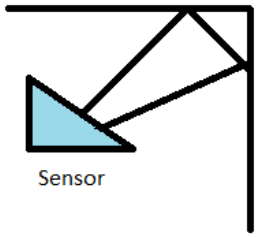


Figura 6. Ecos reflejados por superficies.

Un inconveniente muy frecuente presente en el modelo arquitectónico, es que los sensores están en forma contigua uno del otro, formando una serie de sensores. Al ocurrir esto en distancias cortas, se produce otro fenómeno de reflexiones falsas, debido a que en el instante en que un sensor emite una señal, esta puede ser recibida por el sensor de al lado, que a su vez puede estar esperando su señal enviada. Se debe prestar mucha atención a esto, ubicando los sensores a cierta distancia en la que uno no interfiera con el otro.

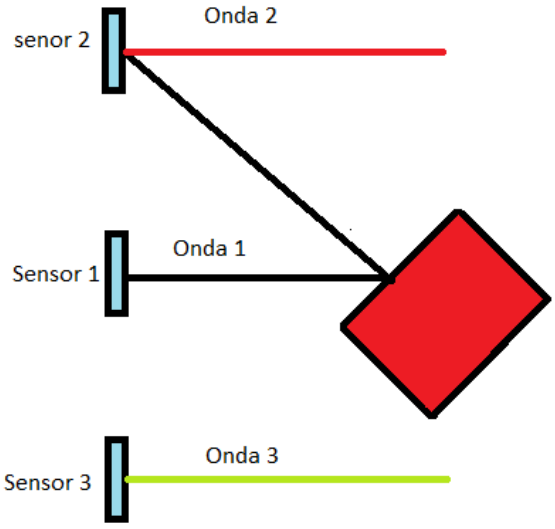


Figura 7.*Ecos de otros sensores*

Debido a los inconvenientes anteriormente mencionados, se presentó un conflicto en el momento de implementar 20 sensores ultrasonido en un espacio tan reducido del modelo arquitectónico del parqueadero, por tal razón se decidió utilizar sensores

infrarrojos evasor de obstáculos como garantía que funcionara el sistema a escala; cabe destacar que en el modelo arquitectónico se van a utilizar dos sensores ultrasonidos para demostrar su correcto funcionamiento en el parqueadero real.

5.3 Sensor infrarrojo evasor de obstáculos

Este módulo sensor infrarrojo emisor y receptor el cual se puede adaptar a luz ambiente y distancia de detección a través de un potenciómetro que viene incluido en la board, dicha distancia se encuentra comprendida entre 2cm - 30cm, con un ángulo de detección de 35°. Estos infrarrojos emiten señales a cierta frecuencia cuando se encuentra detección de algún obstáculo (superficie de reflexión). La señal captada por estos sensores es acondicionada mediante un circuito comparador, esto se ve reflejado mediante un LED indicador de color verde, en donde dependiendo de la configuración del usuario, podrá establecer niveles altos (1 lógico) y bajos (0 lógico) de tensión. [2]

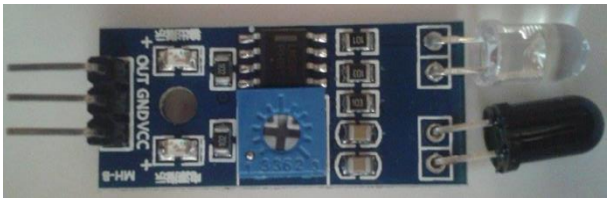
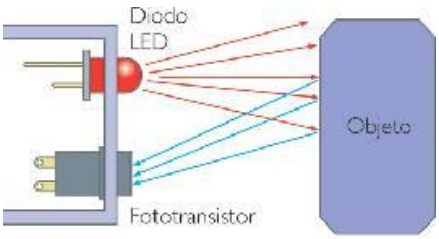
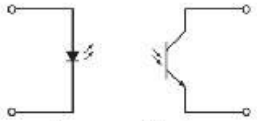


Figura 8. Sensor infrarrojo evasor de obstáculos



Disposición física del sensor óptico.



Esquema eléctrico. [3]

5.3.1 Funcionamiento

- Diodo emisor de luz infrarroja (LED IR)

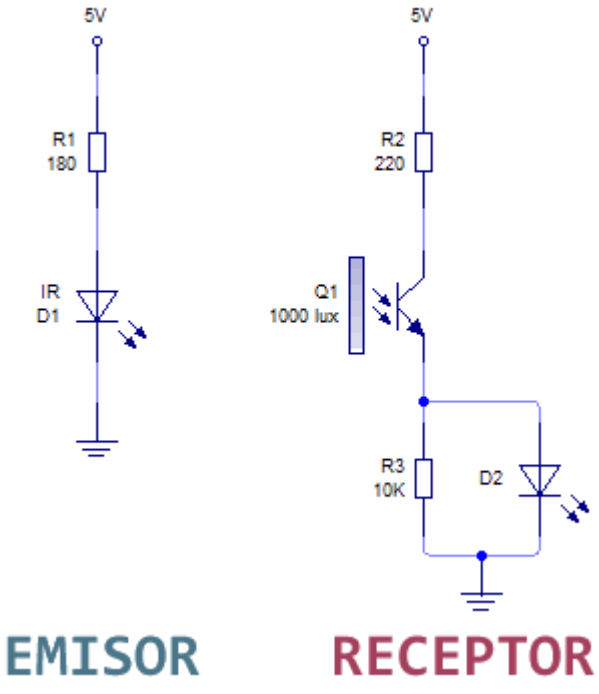
Este LED emite un tipo de radiación electromagnética llamada infrarroja, que es invisible para el ojo humano porque su longitud de onda es mayor a la del espectro visible. [3]

- Fototransistor

Este dispositivo se diferencia de un transistor común porque su base ha sido sustituida por un cristal fotosensible que regula el flujo de corriente colector – emisor de acuerdo a la luz incidente sobre él (en nuestro caso luz infrarroja). [3]

El fototransistor, aunque con la apariencia de un LED común, debe conectarse con la patilla larga a masa y la corta a voltaje.

Con los términos básicos aclarados, en el siguiente esquema se puede observar el emisor y el receptor:



[3]

5.4Arduino Mega ADK

La tarjeta programable Arduino ADK, es una herramienta muy poderosa para el procesamiento de las señales obtenidas por los sensores. Su interfaz de programación es muy fácil de usar y soporta el manejo de gran cantidad de datos. La escogencia de esta tarjeta se debió a la gran cantidad de entradas y salidas que contiene su microcontrolador, algo muy útil para el manejo de los 20 sensores de proximidad con los que se trabajó, proporcionando un beneficio de ahorrar costos y evitar la compra de más controladores.

5.3.1 Concepto

El Arduino Mega ADK, es una placa electrónica basada en el Atmega 2560. Cuenta con una interfaz de host USB para conectar con los teléfonos que tienen implementados el sistema operativo móvil Android, está basado en el MAX3421E IC, consta de 54 pines digitales de entrada / salida (de los cuales 15 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (hardware puerto serial), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reinicio. [4]

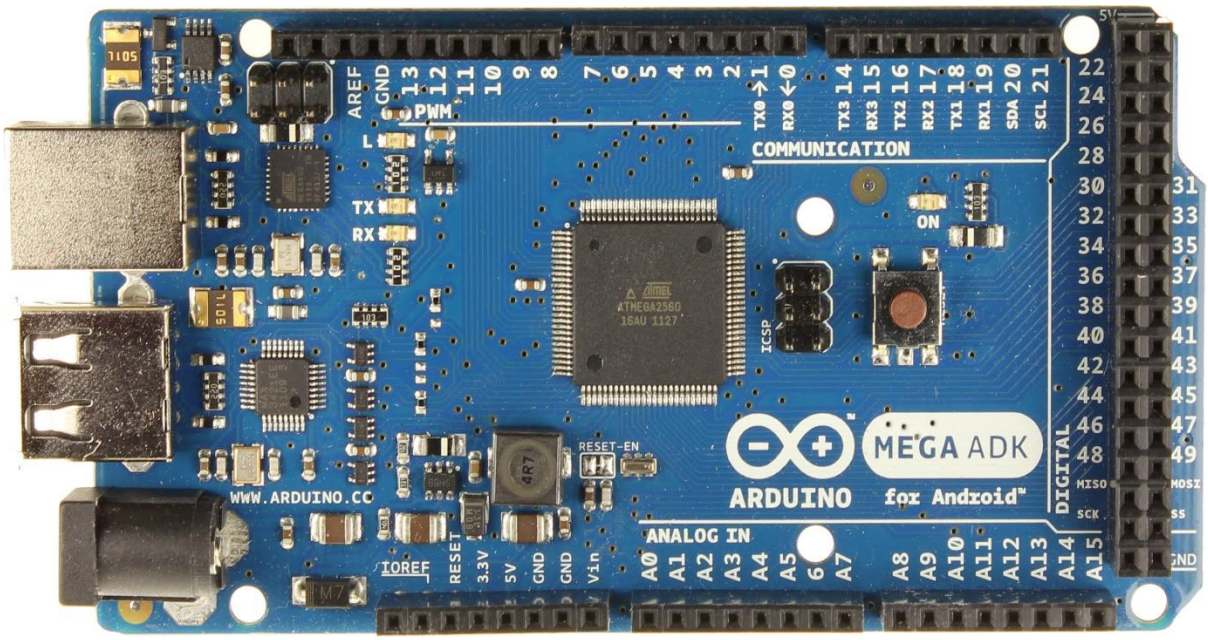


Figura 9*. Arduino Mega ADK [5]*

5.3.2 Características

- Microprocesador ATmega2560.
- Controlador USB host compatible con Google Android.
- Microprocesador Atmega8U2 programado como convertidor USB-serial.
- Tensión operativa: 5 V.
- Tensión de alimentación (recomendado): 7-12 V.
- Tensión de alimentación (límites): 6-20 V.
- 54 entradas/salidas digitales (14 de estas se pueden utilizar para salidas PWM para control de la velocidad de un motor CC por modulación de pulsos).
- 16 entradas analógicas para sensores.
- Corriente continua máxima para las entradas/salidas: 40 mA.
- Corriente continua máxima para los pins de 3,3V: 50 mA.
- Flash memory 256 KB (el bootloader usa 8 KB).
- SRAM 8 KB; EEPROM 4 KB.
- Velocidad del reloj: 16 MHz. [6]

La placa tiene la capacidad de ofrecer 50 pines digitales, con la particularidad de que cada uno se le puede asignar un uso como entrada o salida. Cada uno de los pines trabaja con una alimentación de 5V y son capaces de proporcionar o recibir una corriente máxima de 40 mA. Otra característica importante es que cada uno tiene una resistencia de pull-up de 20-50 kOhms que genera un desconectado por defecto.

Es importante tener en cuenta la capacidad de memoria que tiene un dispositivo para guardar la programación lo suficientemente grande y almacenar extensas cantidades de líneas de código. En el caso de esta, se tiene 256KB de memoria flash para almacenamiento del código. Para el cargador de arranque de la tarjeta se toman 8 KB que trabaja como RAM estática o también llamada SRAM.

5.4 Programación para el procesamiento de datos

En esta sección se hará una explicación detallada del desarrollo de la programación de la tarjeta Arduino Mega ADK, para el correcto procesamiento de los datos obtenidos por los sensores.

El Arduino Mega ADK se puede programar con el software de Arduino, que está disponible de manera gratuita en su página oficial. El Atmega2560 que es el microcontrolador que maneja el Mega ADK, viene precargado con un gestor de arranque que facilita poder cargar un nuevo código sin tener la necesidad de utilizar un programador de hardware externo. La comunicación se realiza utilizando el protocolo originalSTK500v2.La plataforma de Arduino se programa con un software propio, basado en C++, que es un lenguaje de programación de alto nivel.

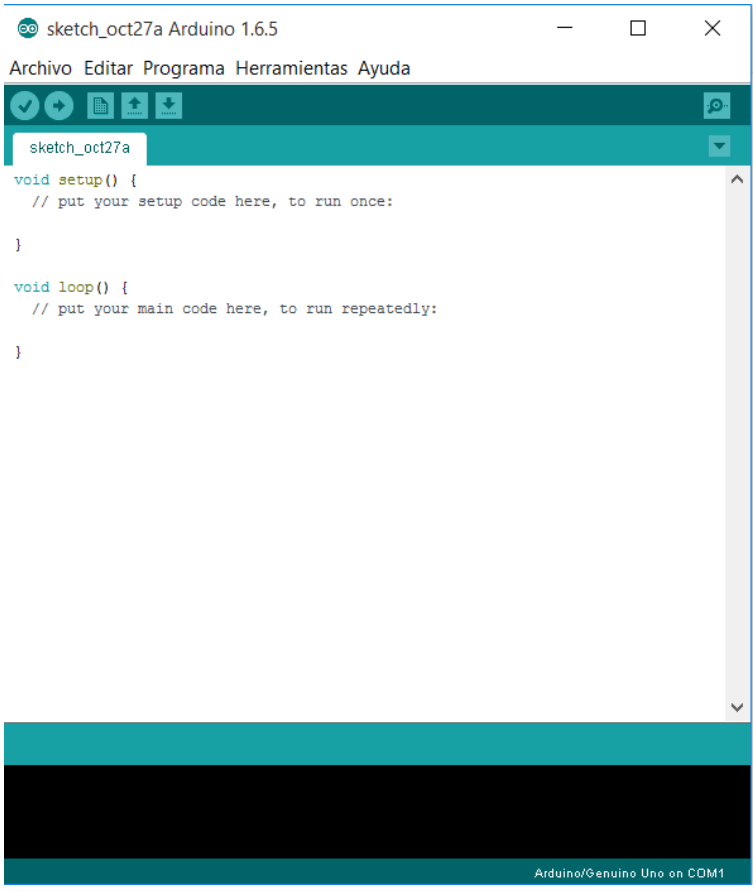


Figura 7.*Interfaz de programación de Arduino*

El primer paso es enlazar la librería del sensor ultrasónico en la programación, para leer el fichero de cabecera y de esta manera usar las funciones y variables referentes al sensor. La librería se añade de la siguiente forma:

```
#include <Ultrasonic.h>
```

El siguiente paso, consiste en determinar los pines que se van a programar en la tarjeta correspondientes a cada uno de los 20 sensores. Para lo anterior se realiza las siguientes líneas de código:

```
Ultrasonic sensor4(14,15); // (Trig PIN,Echo PIN)
Ultrasonic sensor8(7,6);   // (Trig PIN,Echo PIN)
```

A continuación se establecen las variables, en este caso se trabajan como variables booleanas, las cuales se pueden representar con valores de lógica binaria, se utilizan dos valores que generalmente se toman como verdadero o falso.

```
boolean maxi1,med1,mini1 = false;
```

Como se puede observar en la línea de código, los 3 diferentes rangos van a estar con un valor inicial en falso, es decir no se está detectando ningún objeto en ninguna de las 3 distancias.

```
int ir1;
```

Se establece cada sensor infrarrojo como variable entera

```
void setup() { Serial.begin(9600);
pinMode(21,INPUT);
pinMode(19,INPUT);
pinMode(17,INPUT);
pinMode(41,INPUT);
pinMode(2,INPUT);
pinMode(4,INPUT);
pinMode(9,INPUT);
pinMode(11,INPUT);
pinMode(40,INPUT);
```

```
pinMode(62,INPUT);
pinMode(57,INPUT);
pinMode(12,INPUT);
pinMode(68,INPUT);
pinMode(67,INPUT);
pinMode(65,INPUT);
pinMode(60,INPUT);
pinMode(59,INPUT);
pinMode(54,INPUT);
delay(1000); }
```

Se establecen los pines de la tarjeta que utiliza cada sensor infrarrojo y se utiliza un retardo de 1 segundo cuando empiezan las lecturas.

Por último, se realiza una lógica para determinar en el momento la activación del sensor infrarrojo e informar que una plaza está ocupada y en cuales se desactiva e informa que la plaza se desocupó.

```
if(ir1 > 0 && maxi1==false) {
maxi1=true;
}
if(ir1 > 0 && med1==true) {
Serial.println("sensor1:OUT");
med1=false;
delay(500);
 }
if(ir1 < 1 && maxi1==true) {
Serial.println("sensor1:IN");
maxi1=false;
med1=true;
delay(500);
}
```

El sensor realiza lecturas digitales, por lo tanto en el momento que envié un cero se va a imprimir en el puerto serial “sensor:IN” y en el momento que envié un uno se va a imprimir en el puerto serial “sensor:OUT”, todo esto se realiza con un retardo de 500ms.

En el caso de los dos sensores ultrasónicos utilizados se usa la siguiente lógica:

```
if((sensor4.Ranging(CM) > 10) && (sensor4.Ranging(CM) < 13) && maxi4==false) {
maxi4=true;
}
if((sensor4.Ranging(CM) >= 11) && med4==true) {
Serial.println("sensor4:OUT");
```

```
med4=false;
delay(500);
}
if((sensor4.Ranging(CM) > 6) && (sensor4.Ranging(CM) < 9) && maxi4==true) {
Serial.println("sensor4:IN");
maxi4=false;
med4=true;
delay(1000);
}
```

Con la lógica anterior el sensor ultrasonido va a imprimir en el puerto serial “sensor:OUT” en el instante que tome mediciones mayores a 11cms e imprime “sensor:IN” en el instante que tome mediciones mayores a 6cms y menores a 9cms, indicando que el automóvil se encuentra en la plaza de parqueo.

6. CONTADOR Y TALANQUERA MOVIL

Para simular una de las entradas del parqueadero en el modelo arquitectónico se utilizó un contador de automóviles y la talanquera móvil; para realizar el funcionamiento de estos elementos se utilizaron sensores infrarrojos, una pantalla LCD, un arduino uno y servomotores.

Por medio de los sensores infrarrojos conectados al arduino, se realiza la identificación de carros que van ingresando al parqueadero y a través de la pantalla LCD se visualiza la cantidad de automóviles que se encuentran dentro del parqueadero habiendo sido totalizados por la tarjeta que maneja este subsistema.

En el momento que el sensor detecta un automóvil la talanquera realiza un desplazamiento de 90° por medio de servomotores y así permitir el ingreso de los vehículos al parqueadero.

6.1 Servomotores

Es un dispositivo con un eje de rendimiento controlado ya que puede ser llevado a posiciones angulares específicas al enviar una señal codificada. Con tal de que exista una señal codificada en la línea de entrada, el servo mantendrá la posición angular del engranaje. Cuando la señal codificada cambia, la posición angular de los piñones cambia. En la práctica, se usan servos para posicionar elementos de control como palancas, pequeños ascensores y timones. [7]

Uno (rojo) es para alimentación, Vcc (~ +5volts); otro (negro) para conexión a tierra (GND); el último (blanco o amarillo) es la línea de control por la que se le envía la señal codificada para comunicar el ángulo en el que se debe posicionar.

6.2 Configuración tarjeta programable

A continuación se muestra el código que se utilizó para programar el funcionamiento de los servomotores.

```
#include <Servo.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
int contador = 0; // Se crea variable para realizar el conteo de carros.
int ir1; // Se crea  variable para identificar el sensor de la entrada.
int ir2; // Se crea  variable para identificar el sensor de la salida.
Servo myservo1; // Se crea variable para controlar servomotor de la talanquera entrada.
Servo myservo2; // Se crea variable para controlar servomotor de la talanquera Salida.
void setup()
{
   lcd.begin(16, 2); //Se configuran el tipo de LCD a utilizar.
   lcd.setCursor(0,0); //Se resetea el cursor
   lcd.print("  PARQUEADERO"); //Se escriben datos fijos de la pantalla.
   lcd.setCursor(0,1); // Para posicionar el cursor.
   lcd.print("# de Carros ="); //Se escriben datos fijos de la pantalla.
   Serial.begin(9600); //Se configura el puente de comunicación.

   myservo1.attach(3);// Se configura el pin PWM para el servomotor de la entrada.
   myservo2.attach(11); // Se configura el pin PWM para el servomotor de la salida.
   pinMode(2,INPUT); //Se configura el pin como entrada digital para detectar el carro entrando.
   pinMode(12,INPUT); //Se configura el pin como entrada digital para detectar el carro saliendo.
```

```
    delay(400);


}
void loop ()
{
ir1 = digitalRead(2); // Asigno la lectura del sensor a las variables.
ir2 = digitalRead(12); // Asigno la lectura del sensor a las variables.


if(ir1 < 1 && contador <= 19) // Realizo la comparación para detectar el cambio de
estado
   {                    // del sensor infrarrojo y condiciono el número máximo de
      contador++;          // carros a contar.
      delay(700);          // Cuando se cumpla la condición se incrementa la variable
     myservo1.write(90);     // "contador" y se posiciona el servomotor para dar paso
     delay(3000);          // al vehiculo que esta ingresando dejando un pequeño
retardo.
   }
   else if(ir2 < 1 && contador >= 1) // Realizo la comparación para detectar el
cambio de estado
   {                        // del sensor infrarrojo y condiciono el número máximo de
   contador--;                 // carros a descontar.
   delay(700);                 // Cuando se cumpla la condición se disminuye la
variable
   myservo2.write(90);            // "contador" y se posiciona el servomotor para dar
paso
     delay(3000);                // al vehículo que va saliendo dejando un pequeño
retardo.
   }
   else if(ir1 == 1 && ir2 == 1) // Si los sensores no detectan el automóvil, la
variable no
   {                       // incremente ni disminuye y los servomotores mantienen
```

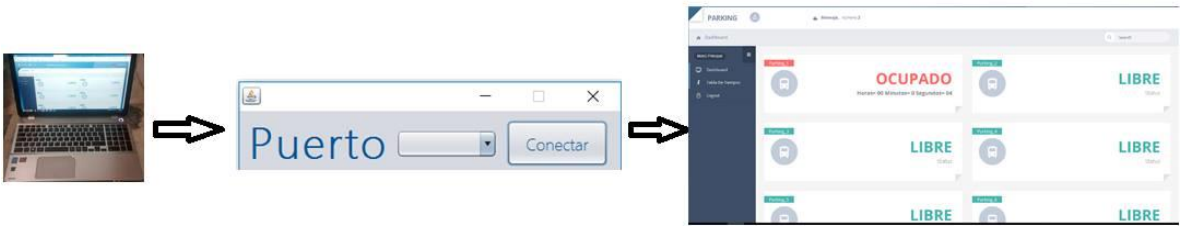
```
    contador + 0;                // posición impidiendo el paso.
    myservo2.write(0);
    myservo1.write(0);
    }
  lcd.setCursor(14,1); //Determino la posición donde deseo escribir datos en la LCD.
  lcd.print(contador); //Se escribe constantemente el valor de la variable contador en la LCD.
```

7. COMUNICACIÓN ENTRE EL MODELO DE DETECCION Y LA INTERFAZ WEB.

En esta sección se explicará el protocolo de comunicación utilizado para el envío de datos entre el modelo de detección y la interfaz web. El protocolo de comunicación se utiliza para establecer unas reglas y un estándar que permita la transmisión de datos entre dos entidades y a su vez establece el método de detectar y corregir errores cuando se requiera utilizar. Para el caso de este proyecto se usó el protocolo de comunicación serial dividido en dos partes:

1- Comunicación entre tarjeta programable y equipo de cómputo, por medio de (USB).

2- Comunicación entre el equipo de cómputo y la interfaz web, por medio de una conexión http Get en java.



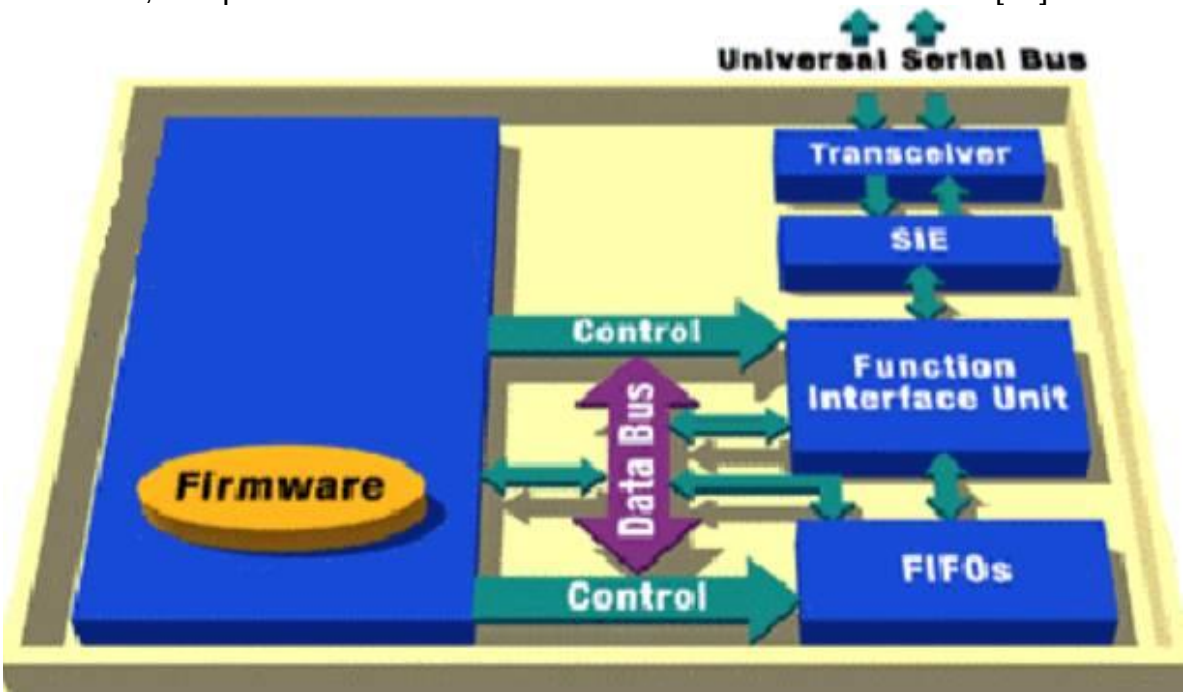
7.1 Comunicación USB

El USB es un bus punto a punto: dado que el lugar de partida es el host (PC o hub), el destino es un periférico u otro hub. No hay más que un único host (PC) en una arquitectura USB. Los PC estándar tienen dos tomas USB, lo que implica que para permitir más de dos periférico simultáneamente, es necesario un hub. Algunos periféricos incluyen un hub integrado, por ejemplo, el teclado USB, al que se le puede conectar un Mouse USB. Los periféricos comparten la banda de paso del USB. El protocolo se basa en el llamado paso de testi (token). Los ordenados proporciona el testigo al periférico seleccionado y seguidamente éste le devuelve el testigo en su respuesta. Este bus permite la conexión y la des-conexión en cualquier momento sin necesidad de apagar el equipo. [8]

7.1.1Estructura USB

El Bus Serial Universal está dado esencialmente por un cable especialmente diseñado para la transmisión de datos entre la computadora y diferentes periféricos, que pueden acceder simultáneamente al mismo con el fin de recibir o transmitir datos. Todos los dispositivos conectados acceden al canal o medio para transmitir sus datos de acuerdo a las normas de administración del host regido por un protocolo que consecutivamente va dando la posibilidad de transmitir a cada periférico, el protocolo USB se parece de cierta forma al protocolo Token Ring. [9]

La arquitectura del bus garantiza la posibilidad de que los periféricos sean conectados y desconectados del host mientras este y otros periféricos están operando normalmente, característica a la que se denomina Conectar y Desconectar dinámico o simplemente En Caliente, sin perjuicio para ningún dispositivo en funcionamiento. Todos los dispositivos USB responden también a un mismo patrón estandarizado, que más allá de las características propias de cada fabricante, comprende los mismos elementos funcionales. Estos son: [10]



[10]

7.1.1.1 Transceiver.

El cable USB está compuesto por solo cuatro cables: Vbus, D+, D-y GND. La información y los datos se mueven por los cables D+ y D-, con dos velocidades: 12Mbps o 1.5Mbps. Fabricado dentro del mismo chip controlador de periférico, y puede verse como la interfaz misma de un dispositivo externo contra el resto del sistema. [10]

7.1.1.2 Serial interface engine (SIE):

El SIE tiene la función de seriar y agrupar las transmisiones, además maneja los protocolos de comunicación, las secuencias de paquetes, el control CRC y la codificación NRZI. [10]

7.1.1.3Function interface unit (FIU):

Este elemento administra los datos que sontransmitidos y recibidos por el cable USB. Se basa y apoya en el contenido y estado de los FIFOs (a continuación). Monitorea los estados de las transacciones, los buffer FIFO, ysolicita atención para diversas acciones através de interrupciones contra el CPU delhost. [11]

7.1.1.4 FIFOs(First In, Firts Out):

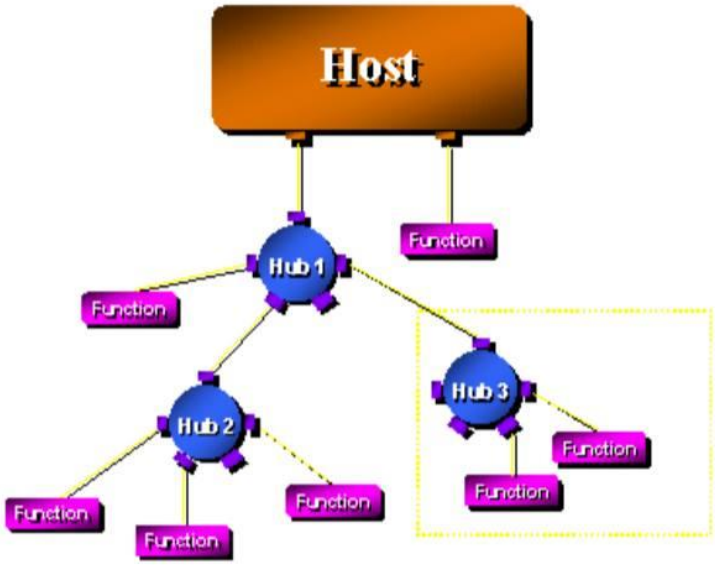
Un controlador de USB típico tiene dependiendo de su velocidad máxima soportada entre 3 y 32 buffers tipo FIFO, una mitad destinada a la transmisión y la otra destinada a la recepción de datos. Tanto para la transmisión como para la recepción, los buffers soportan las diferentes formas de transmisión que USB permite generalmente impuestas por restricciones de memoria en cada una. [11]

Por lo general las FIFOs se asocian en pares de transmisión-recepción identificándose con un mismo número, pero no hay que perder de vista que pertenecen a espacios de memoria separadas. Lo antes dicho tiene una excepción, esta es la FIFO0 que es bidireccional y ocupa una única porción de memoria. Esta generalmente, tiene reservado en el buffer en espacio de 8 a 16 bytes, y se dedica

a almacenar información de control relacionada a las transferencias. El resto de las FIFOs, tienen asignado un espacio de memoria que puede variar entre 16 y 1024 bytes dependiendo para que tipo de transmisión estén preparadas, estas FIFOs se emplean para el control de interrupciones y transmisiones tanto isocrónicas como las Bulk (Bulk significa voluminoso o de gran volumen, traducción poco ilustrativa, por lo cual es preferible entender el término por su nombre en inglés. [10]

7.1.2 Topología del bus

La forma física en la que los elementos se interconectan dentro del sistema USB, puede asemejarse a la topología estrella estratificada piramidalmente. El centro de cada estrella es un hub, un dispositivo que por un lado se conecta al computador o a otro hub y por otro lado, permite conectar al mismo varios dispositivos o en su defecto nuevos hubs. Muchos dispositivos USB han de traer conectores USB adicionales incorporados, por ejemplo un monitor puede tener 3 ó 4 conectores USB donde pueden ir el teclado, el ratón, y algún otro dispositivo. Por su parte el teclado puede tener otros más, y así sucesivamente hasta tener más de 127 dispositivos, todos funcionando simultáneamente. [12]



[12]

7.1.3 Modelo lógico funcional USB

El flujo de datos USB se da partir de tres niveles lógicos: entre el Software Cliente y la Función, el Controlador USB y el dispositivo, y finalmente la capa física, donde la transmisión realmente sucede. [11]

Es importante entender que este modelo es muy parecido al OSI, el estándar de redes, y su comprensión radica en el hecho de que si bien existe un solo canal físico, pero los datos son manejados en cada punto por unidades homólogas o idénticas, tal como si estuviesen sosteniendo una comunicación directa. Por esta razón se las denomina Capas Lógicas. El nivel superior lógico es el agente de transporte de datos que mueve la información entre el Software Cliente y el dispositivo. Existe un Software Cliente en el host, y un Software de atención al mismo en cada una de las funciones o periféricos USB. A este nivel, el host se comunica con cada uno de los periféricos en alguna de las varias formas posibles de transmisión que soporta USB. El Software Cliente solicita a los dispositivos diversas tareas y recibe respuestas de ellos a través de esta capa. [11]

La capa lógica intermedia es administrada por el Software de Sistema USB, y tiene la función de facilitarles las tareas particulares de comunicación a la capa superior, administrando la parte del periférico con la que la capa superior desea comunicarse, maneja la información de control y comando del dispositivo, etc. Su objetivo es permitir a la capa superior concentrarse en las tareas específicas tendientes a satisfacer las necesidades del usuario, adicionalmente gestiona el control interno de los periféricos. [12]

El acceso al bus es bajo la modalidad de Ficha o Token, lo que involucra siempre complejidad de protocolos, especialmente si agregamos dos velocidades posibles: 12Mbps ó 1.5Mbps. Todos estos algoritmos y procesos son administrados por el Host-USB, reduciendo la complejidad del periférico, y lo más importante, el costo final de los dispositivos USB. La capa física del modelo lógico USB comprende los puertos físicos, el cable, los voltajes y señales, el hardware y funcionamiento del hardware. Esta capa tiene el objetivo de liberar a las capas superiores de todos los problemas relacionados a la modulación, voltajes de transmisión, saltos de fase, frecuencias y características netamente físicas de la transmisión. [12]

7.1.4 Protocolo del bus

La transacción o envío de datos provenientes de la tarjeta Arduino, se realiza por medio del- bus serial universal, en donde cada transferencia se lleva acabo dependiendo de lo que el controlador del host decida, enviando un paquete al dispositivo que tiene el turno de usar el bus.

Cada aparato tiene un número de identificación que ha sido dado por el controlador en el instante en el que el computador arranca o cuando se incluye un nuevo elemento al sistema de comunicación, lo cual es primordial para que cada dispositivo que se encuentre conectado pueda identificar si un paquete de información es para él o no. Los paquetes de datos que circulan por el sistema de comunicación USB, son denominados Token Packet. En el momento de las transmisiones de información, cada elemento debe recibir el permiso del controlador para ejecutar la acción correspondiente a su función, y cuando ya no quedan más datos para enviar, el mismo elemento envía un paquete el host confirmando el fin de su tarea y dando paso a que otros dispositivos realicen el proceso.

Al realizar la transmisiones de datos, es muy común que durante la comunicación se presenten errores, por lo tanto el protocolo USB utiliza un sistema de recuperación de errores muy confiable como lo es el CRC (código de redundancia cíclica). Este modelo de detección de errores, consiste en que el emisor representa las filas de datos como polinomios por medio de operaciones matemáticas, en el momento en que los paquetes lleguen al receptor, este realiza una división entre el polinomio que llega y otro ya preestablecido o polinomio generador, si el resto de la división es 0 quiere decir que los datos llegaron correctamente, en caso contrario significará que se produjo un error y se solicita el reenvío del mensaje.

7.1.5 Comunicación de la tarjeta programable por USB

Como se ha nombrado anteriormente, se utilizó el cable USB para realizar la comunicación entre el Arduino y el equipo de cómputo, con el fin de enviar la información procesada recogida por los sensores, a este proceso se le llama conexión serie, debido a que la transferencia de bits se realiza uno detrás del otro que en este caso seria los 1 y 0 de los estado de cada plaza de parqueo.

Es importante tener en cuenta que sobre el cable USB, los estados bajos y altos se emiten por voltajes o tensión eléctrica. Todos los datos que se mandan o reciben

entre el equipo de cómputo y la tarjeta Arduino, son en esencia los cambios de estados altos y bajos (1 y 0), por eso si se pudiera observar estas transmisiones gráficamente se verían una fila de unos y ceros. Se puede observar que al realizar la transmisión en un extremo se enciende y se apaga un pin (Arduino) y en el otro extremo se va a leer el voltaje de la red que representa la información. Para que la comunicación se realice correctamente, se debe asegurar que la velocidad con la que se envían los caracteres que son los baudios (medida utilizada en telecomunicaciones), sea convenida por ambas partes y de esta manera los elementos puedan entenderse.

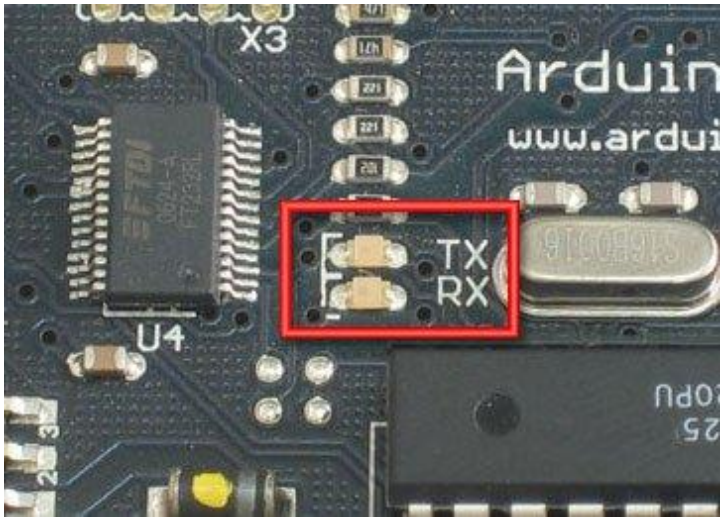


Figura 8*. Pines de transmisión y recepción de datos. [13]*

Cuando se hace uso del Arduino, se puede identificar el proceso de comunicación en serie en el momento que se envían sketches a la tarjeta. Cuando se realiza la compilación del código, en realidad se inicia un proceso de conversión de la programación que se ha desarrollado a datos binarios 1 y 0. Cuando el código es cargado al Arduino, cada uno de los bits es transferido a través del cable USB al chip principal de la placa, es en ese momento donde se puede observar claramente que los Leds próximos al conector USB empiezan a emitir una luz parpadeante, Dando a entender que ha empezado el envío de datos. El Led RX se enciende en el momento en que se está recibiendo información y el led RT parpadea en el momento en que se realiza la transferencia de datos.

7.1. 5 .1 Declaración“begin”

En el momento en que se desea establecer una comunicación serial, el primer paso que se debe tener en cuenta es indicarle al Arduino que se va a realizar un vínculo con él, y a su vez indicar el protocolo de comunicación que se va a utilizar para realizar el enlace. Para desarrollar el proceso anterior, se usa la función “setup”, en esta función se señala todos los servicios y configuraciones de la tarjeta que se van a tener en cuenta a la hora de realizar la programación. Luego se incluye la declaración “begin” de la siguiente manera:

Serial.begin(velocidad)

Al insertar la declaración “*Serial.begin*”, se señala que se va a manejar la comunicación serial. Después de la declaración, se puede observar que entre paréntesis se encuentra la palabra velocidad, en ese espacio va expresado el número de baudios, que representan el número de bits que se pueden enviar por segundo. Existen una gran cantidad de velocidades de transmisión posibles, dependiendo de la velocidad de conexión que tenga cada elemento. Debido al alcance tecnológico al que hemos llegado, se pueden observar en la actualidad dispositivos con una gran velocidad de conexión, de ahí la importancia de establecer la velocidad más adecuada.

La placa Arduino tiene la capacidad de soportar velocidades de 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 o 115200. En el caso de la comunicación de la tarjeta con el equipo de cómputo, se utilizó una velocidad media de 9600 baudios. En el instante en que se crea la declaración “begin” en la función setup, se estable un “buffer” de información que se puede enviar de un lado para otro, y de esta manera se indica que se va a trabajar con una comunicación serial a una determinada velocidad de baudios, en el caso de este proyecto sería igual a:

Serial.begin(9600)

7.1.5.2 Declaraciones “print” y “println”

En el momento en el que se aplica la declaración de que se va a realizar una comunicación serial entre la placa Arduino y el equipo de cómputo, se procede a enviar por parte de la tarjeta un buffer con la información al ordenador, y por medio de una pantalla, se puedan visualizar para beneficio del programador y del que vaya a utilizar el programa. Para el anterior proceso se utilizan dos mandatos:

“print”,
“printnl”,

Estos dos mandatos tienen casi las mismas funciones, que es cargar los datos en el buffer con un formato anteriormente preestablecido y enviarlos al equipo de cómputo, donde se va a observar por medio de la pantalla. Se puede ver que el segundo mandato agrega un “nl” en el final de la palabra, esta pequeña diferencia indica que finaliza los datos al enviar y de esta manera se evita que se acumule datos enviados en la pantalla y así controlar el momento en que se desea cambiar de línea de texto.

Es muy útil establecer el tipo de configuración mencionado en el párrafo anterior, en el momento en que se quiera que la información enviada aparezca una al lado de la otra, se debe utilizar el mandato “print”, pero si lo que se quiere es que el texto se muestre en una línea distinta se envía el mandato “printnl”.

Serial.print(val,format)
Serial.printnl(val, format)

Donde val, identifica el tipo de valor numérico que se va a enviar y format indica que lo que se está enviando sean datos imprimibles.

7.2 Comunicación por Get de java.

Los datos que llegan al equipo de cómputo proveniente de la tarjeta programable, deben transmitirse en tiempo real a la URL de la página web para que puedan ser visualizados en la interfaz diseñada, de esta manera el usuario pueda estar al tanto del comportamiento del parqueadero y la disponibilidad de las plazas. Para realizar el proceso anterior se utiliza una aplicación básica en java que permite la subida de datos a la red de una manera instantánea.

7.2.1Comunicación con Arduino

En el momento de realizar una conexión entre Java y Arduino, es necesario usar una librería personal de Java llamada RXTX. Esta librería es exigida por Java, para realizar una transmisión completa de información a través del puerto serie. El permiso de distribución que utiliza es GNU LGPL. Está basado en la definición para

la API de comunicaciones de Java, pero usan un paquete propio, genuino, el cual es compatible a la hora de utilizar la especificación.

En la actualidad la librería RXTX ha sustituido a las extensiones APIque se encontraban anteriormente, por lo tanto es apta para ser utilizada sin ningún problema.

Otro aspecto que se debe tener en cuenta, es que se tiene que ajustar la dirección del puerto COM de la tarjeta Arduino que en ese momento se esté utilizando, para el caso de este proyecto, se especifica el puerto "COM 7".

EL primer paso que se debe realizar a la hora de establecer la conexión, es adjuntarla librería RXTX en el entorno de programación, en este caso IDE (Netbeans). Cuando ya se han descargado las librerías y los programas, se procede a descomprimir el archivo “rxtx librerías para 32 bits” y el archivo rxtxSerial.dll se copia en la carpeta donde se va a ejecutar el programa.

Posteriormente, cuando todos los archivos se encuentren disponibles, se abre Netbeans, se ejecuta un nuevo proyecto, allí se ingresa a propiedades y se da clic en librerías, finalmente se selecciona add/jar/folder y se escoge la librería RXTXcomm.

A continuación se empezará la configuración del IDE de Java y se determinan los puertos que se van a utilizar en la comunicación serial entre Arduino y Java. Se emplea un código muy básico de programación para la explicación del proceso.

Importamos las librerías:

```
import gnu.io.PortInUseException;
import gnu.io.SerialPort;
import gnu.io.UnsupportedCommOperationException;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Enumeration;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.UIManager;
import javax.swing.UIManager.LookAndFeelInfo;
import javax.swing.UnsupportedLookAndFeelException;
```

Se procede a virtualizar los puertos COM:

```
public class EscrituraSerial1 {
public static void main(String[] args) {
```

Se buscan todos los puertos disponibles y se guardan en el objeto puertos:

```
Enumeration puertos;
puertos=CommPortIdentifier.getPortIdentifiers();
```

Identificar cada uno de los puertos COM disponibles:

```
CommPortIdentifier portId;
```

A través de la siguiente clase se abren los puertos:

```
SerialPort serialport;
```

Buscar puerto por puerto y señalar con cual se desea trabajar:

```
while (puertos.hasMoreElements()) {
portId = (CommPortIdentifier) puertos.nextElement();
System.out.println(portId.getName());
if (portId.getName().equalsIgnoreCase("COM7")) {
try {
```

Finalmente se realiza la escritura del puerto serial:

```
serialport= (SerialPort)portId.open("EscrituraSerial1", 500
serialport.close();
} catch (Exception e)
```

Para realizar la escritura serial por print, se debe declarar el objeto de salida de tipo printstream y con el ciclo for recorrer los datos txt. Al finalizar enviamos al puerto serial.

```
salida = new PrintStream(serialport.getOutputStream());
for (int i = 0; i <vec.length; i++) {
String c=extraer.readLine();
vec[i]=c;
salida.print(vec[i]+"\n");
}
```

En el caso que se requieran hacer lecturas seriales, se utiliza la siguiente variable de tipo Inputstream para obtener el dato y posteriormente imprimirlo:

```
entrada = serialport.getInputStream
t.start();
} catch (Exception e) {
} } }
}

public static class LeerSerial implements Runnable {
int aux;
public void run () {
while(true){
try {
aux = entrada.read();
Thread.sleep(100);
if (aux>0) {
System.out.println((char)aux);
}
} catch (Exception e) {
} } }
}public static void main(String[] args) {
new LecturaSerial();
}}
```

7.2.2. Aplic

ación Java

Las librerías, clases y código general mencionado anteriormente, hacen parte de la estructura del aplicativo realizado en Java para la comunicación con la página web, el código completo se encuentra en los anexos de este documento. Al ejecutar toda la programación se obtiene la siguiente interfaz:

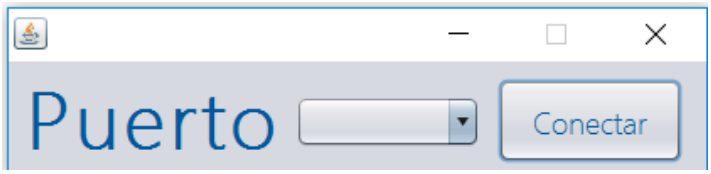


Figura 9.*Interfaz del aplicativo java*

Se puede observar que es una interfaz básica y fácil de usar, en el momento en el que se realiza la conexión USB, aparece el puerto COM que se está utilizando, y finalmente solo se necesita presionar el botón conectar para que se inicie la comunicación a la página web.

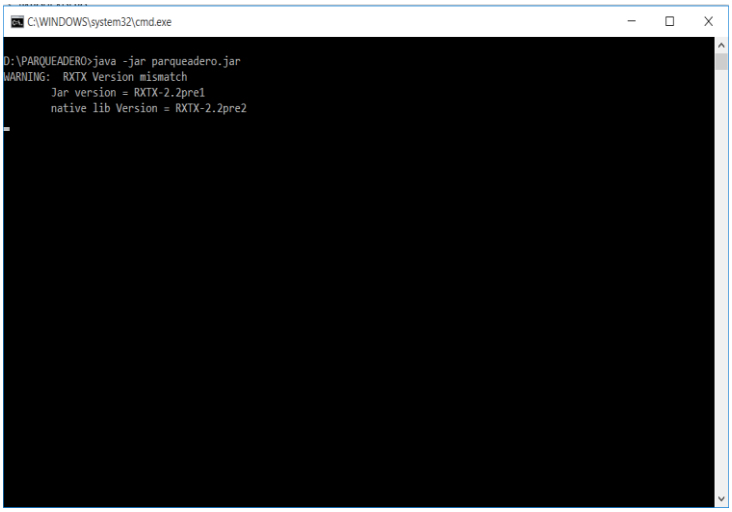


Figura 10.*Ventana de visualización de la escritura serial.*

Se tiene la opción de que junto a la aplicación se abra una ventana cmd, en la cual se puede observar todas las impresiones realizadas por el código y de esta manera, estar al tanto de cómo está funcionando la aplicación y que datos se están enviando.

7.2.3. Transmisión de datos a la URL

Para realizar la conexión, se utilizó el método Get, con el fin de enviar los datos procedentes de la tarjeta Arduino a la URL de la página web a través de la aplicación en Java. Para realizar el procedimiento de conexión con la página web, se debe tener en cuenta que se está trabajando con un Get y se utiliza la clase URLconnection, que es requerida para realizar transmisiones de datos a sitios HTTP desde una aplicación java.

La URLconnection se debe tomar como un elemento que se implementa cuando es necesario sostener un enlace en tiempo real entre una aplicación y una URL ya especificada.

Se utiliza la siguiente demostración para entender el uso de la clase mencionada:

```
1URL url = newURL("http://www.parqueadero.work/WS/setparqueadero/");
2URLConnection con = url.openConnection();
```

Se ejecutan las siguientes líneas partiendo de la URL especificada:

```
importjava.io.BufferedReader;
importjava.io.IOException;
importjava.io.InputStreamReader;
importjava.net.MalformedURLException;
importjava.net.URL;
importjava.net.URLConnection;
private void sendGet(String sensor1)
    throws Exception
  {
    String url = "http://www.parqueadero.work/WS/setparqueadero/" + sensor1;

    URL obj = new URL(url);
    HttpURLConnection con = (HttpURLConnection)obj.openConnection();

    con.setRequestMethod("GET");
```

```
    con.setRequestProperty("User-Agent", "iexplore");

    int responseCode = con.getResponseCode();
    System.out.println("Enviado a URL : " + url);

    BufferedReader in = new BufferedReader(new
InputStreamReader(con.getInputStream()));

    StringBuffer response = new StringBuffer();
    String inputLine;
    while ((inputLine = in.readLine()) != null) {
response.append(inputLine);
    }
    in.close();
```

La aplicación Java envía constantemente datos en tiempo real a la interfaz web provenientes del Arduino, con esto se asegura una comunicación segura, rápida, confiable y con la menor pérdida de datos.

En la programación de Arduino se establecieron variables tipo booleanas, para identificar los estados de cada plaza, se fijó en el código Java que para el valor 0 el sensor está detectando un objeto en la espacio de parqueo y con el valor 1 el sensor no está detectando nada y la plaza se encuentra libre. Para que la página web interprete el número de plaza y el estado se estableció el siguiente formato:

1_0. 2_1. 3_0

El primer número indica el número de la plaza de parqueo, se separa con un guion abajo del siguiente número que representa el estado de la plaza, y por último se asigna un punto para separar los datos de los siguientes sensores.

8. REQUERIMIENTOS PARA EL DESARROLLO DE LA PÁGINA WEB.

En este capítulo se presentan todos los requerimientos que se necesitan para el desarrollo de la página web, comenzando con la configuración de framework, empleado para definir el patrón de arquitectura de software establecido. Este proceso se lleva a cabo con el fin separar los datos de la lógica. También se va a identificar el tipo de servidor y base de datos que se utilizaron.

8.1 Frameworks

Un framework Web, se puede definir como un conjunto de componentes (por ejemplo clases en java y descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web. [14].

El framework que se utilizó para realizar la página Web fue el **Codeigniter**, es un conjunto de herramientas para aplicaciones web usando php, su objetivo principal es permitir desarrollar proyectos mucho más rápido de lo que se podría si se escribieran desde cero, admite creativamente enfocarse en el proyecto minimizando la cantidad de código necesario para una tarea dada.

8.1.1. Diagrama de flujo del Framework Codeigniter

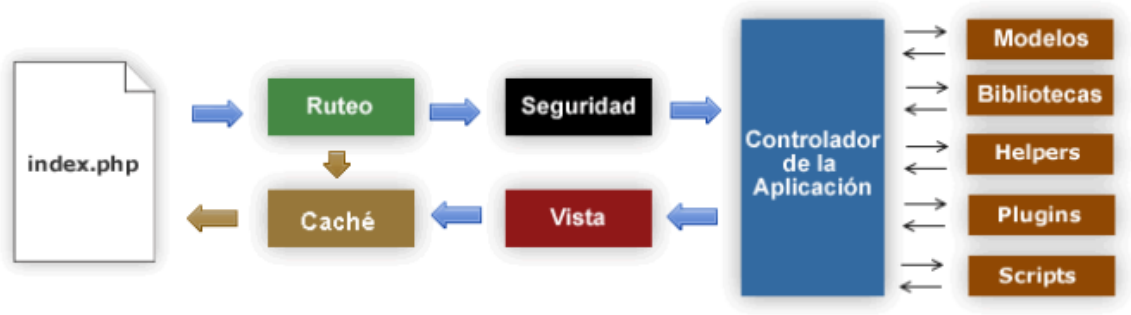


Figura 11. *Diagrama de flujo del Framework Codeigniter [15]*

- El index.php, se utiliza como el controlador frontal, inicializando los recursos básicos que necesita Codeigniter para ejecutar.

- El Ruteador explora la solicitud HTTP para determinar que debería hacer con ella.
- El archivo caché se envía directamente al navegador sin pasar por la ejecución normal del sistema.
- En Seguridad, antes que se cargue el controlador de la aplicación, por razones de seguridad se filtran la solicitud HTTP y cualquier otro dato enviado por el usuario.
- El controlador carga el modelo, las bibliotecas del núcleo, helpers y cualquier otro recurso requerido para procesar una solicitud específica.
- La Vista terminada se procesa y se envía al navegador para que se pueda ver. Si el Caché está habilitado, la Vista se cachea primero para las siguientes solicitudes que la necesiten puedan ser servidas. [15]

8.1.2. Configuración

Primero se procede a cargar las funciones locales del Framework:

require(BASEPATH.'core/Common.php');

Luego se cargan las constantes del Framework:

if (defined('ENVIRONMENT') AND
file_exists(APPPATH.'config/'.ENVIRONMENT.'/constants.php'))
{ require(APPPATH.'config/'.ENVIRONMENT.'/constants.php'); } else
{ require(APPPATH.'config/constants.php'); }

Después se define un controlador de error personalizado para que se pueda registrar los errores de php:

set_error_handler('_exception_handler'); if (! is_php('5.3'))

{ @set_magic_quotes_runtime(0); // Kill magic quotes }

Se inserta el prefijo subclase:

if (isset($assign_to_config['subclass_prefix']) AND $assign_to_config['subclass_prefix'] != '')

{ get_config(array('subclass_prefix' => $assign_to_config['subclass_prefix'])); }

Después se establece un tiempo de ejecución del script:

if (function_exists("set_time_limit") == TRUE AND @ini_get("safe_mode") == 0) {
@set_time_limit(300); }

Se inicia el temporizador:

$BM =& load_class('Benchmark', 'core'); $BM->mark('total_execution_time_start'); $BM->mark('loading_time:_base_classes_start');

Se carga el controlador de aplicación y el controlador local

require BASEPATH.'core/Controller.php'; function &get_instance() { return CI_Controller::get_instance(); } if (file_exists(APPPATH.'core/'.$CFG->config['subclass_prefix'].'Controller.php')) { require APPPATH.'core/'.$CFG->config['subclass_prefix'].'Controller.php'; }

8.1.3. Modelo vista controlador

El modelo vista controlador, se puede definir como una arquitectura utilizada en la aplicación web diseñada para este proyecto, con el fin de separar los datos de la página con la lógica de programación. El hecho de implementar esta configuración, permite construir las piezas de un programa por separado para después enlazarlas en el momento de ejecución, con el fin de separar los componentes defectuosos sin necesidad de afectar las demás piezas que intervienen en el sistema.

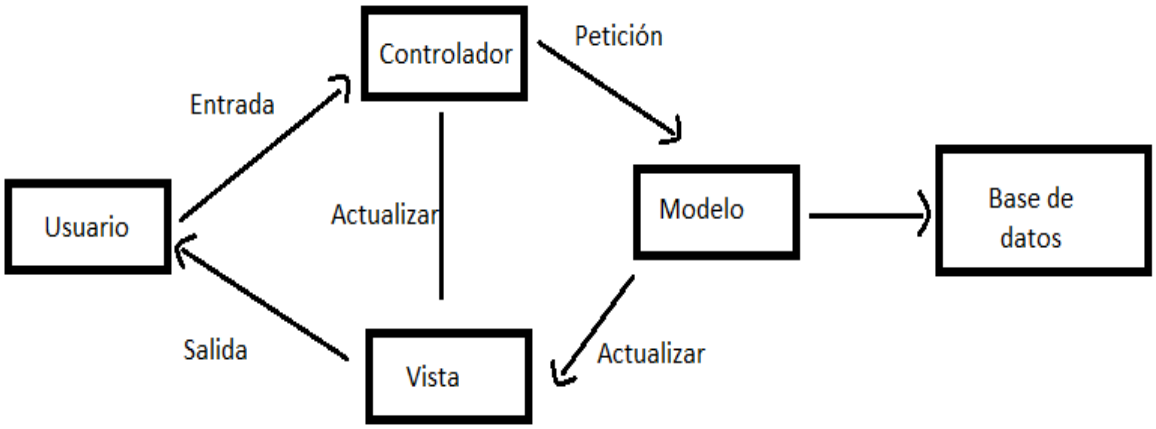


Figura 12.*Relación entre modelo vista controlador*

8.1.3.1 Modelo.

En este primer paso de la configuración MVC, se representan todos los datos almacenados en el programa. Gestiona los datos y verifica las modificaciones realizadas durante la configuración. Este objeto no presenta ningún conocimiento de las vistas o los controladores, pero si se le encomienda, el deber de mantener un enlace constante con las vistas y notificarlas en el momento en que el modelo cambie.

El modelo tiene la capacidad de acceder a la base de datos del sistema, con lo que le da la facilidad de definir los protocolos o reglas que se van a utilizar. El enlace que mantiene con la vista permite que ante cualquier cambio producido por un agente externo, estos se puedan visualizar de manera rápida y correcta.

```
28  class CI_Model {
29
30      /**
31       * Constructor
32       *
33       * @access public
34       */
35      function __construct()
36      {
37          log_message('debug', "Model Class Initialized");
38      }
39
40      /**
41       * __get
42       *
43       * Allows models to access CI's loaded classes using the same
44       * syntax as controllers.
45       *
46       * @param       string
47       * @access private
48       */
49      function __get($key)
50      {
51          $CI =& get_instance();
52          return $CI->$key;
53      }
54  }
```

Figura 13. *Clase modelo de la página web.*

8.1.3.2 Vista.

La vista es la responsable de la representación visual de los datos manejados por el modelo. Es la encargada de crear una visualización del modelo y mostrar al usuario todos los datos almacenados. Por lo general este objeto realiza una interacción constante con el controlador, pero en ocasiones se presenta la posibilidad de que se comunique directamente con el modelo a través de una petición de este mismo.

La labor principal de la vista es obtener todos los datos procesados por el controlador o el modelo, para luego ofrecerlos de manera visual a cada uno de los usuarios, por lo tanto se presenta una función de actualización, que puede ser invocada por los otros dos objetos en el momento de que se genere algún cambio.

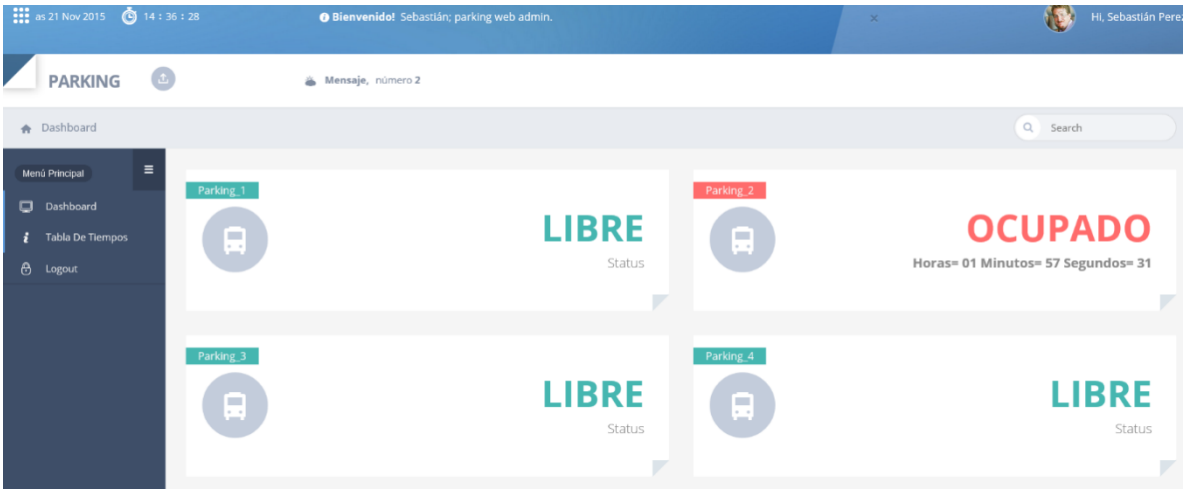


Figura 14*. Vista de la página web*

8.1.3.3 Controlador

La tarea del controlador consiste en darle un propósito a las órdenes provenientes del usuario, ejerciendo su labor principalmente en los datos manejados por el modelo. Realiza un constante enlace con el modelo y la vista, ya que en el momento en que se realiza un cambio, tiene que ejecutar sus funciones ya sea por una orden de la vista o por un cambio en los datos del modelo.

El controlador se encarga de tomar las solicitudes de entrada. Contiene un número de reglas que se encargan de configurar los eventos, ya sean por peticiones del modelo o la vista. Si recibe una petición del modelo, generalmente es para el manejo de datos y si por el contrario recibe una petición de la vista, es para generar un proceso de actualización.

```php
*/
class CI_Controller {

	private static $instance;

	/**
	 * Constructor
	 */
	public function __construct()
	{
		self::$instance =& $this;

		// Assign all the class objects that were instantiated by the
		// bootstrap file (CodeIgniter.php) to local class variables
		// so that CI can run as one big super object.
		foreach (is_loaded() as $var => $class)
		{
			$this->$var =& load_class($class);
		}

		$this->load =& load_class('Loader', 'core');

		$this->load->initialize();

		log_message('debug', "Controller Class Initialized");
	}

	public static function &get_instance()
	{
		return self::$instance;
	}
}
```

Figura 15. *Clase controlador de la página web.*

8.1.3.4 Operación del modelo.

En general la implementación de la configuración MVC actúa de la siguiente manera:

- En el primer paso, el cliente realiza un contacto con la interfaz de usuario, ya sea presionando un botón o llenado algún espacio.

- En el segundo paso, el controlador toma la orden ejercida por el usuario a través de la vista, y de esta manera realizar un manejo de la orden que llegó a través de un gestor de eventos.

- En el tercer paso, el controlador realiza una conexión con el modelo, y así ejecutar una actualización referente a la acción realizada con el usuario en la interfaz.

- En el cuarto paso, el controlador le encarga a los objetos de la vista la labor de efectuar la interfaz de usuario. En ese momento, la vista recibe sus datos del modelo para crear la interfaz más adecuada para el usuario, dependiendo de los requerimientos que este hizo en el primer paso.

- Por último, la interfaz manejada por el usuario se dedica a la tarea de esperar más órdenes, para comenzar de nuevo con el ciclo anterior.

8.2 Servidor XAMPP

En el momento de escoger los servicios requeridos para el desarrollo de la página web, se tuvieron en cuenta las funciones prestadas por el servidor XAMMP.

8.2.1 Concepto

Es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor Web Apache y los intérpretes para lenguajes de script: PHP y Perl.El nombre proviene del acrónimo de X(para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl. El programa está liberado bajo la licencia GNU y actúa como un servidor Web libre, fácil de usar y capaz de interpretar páginas dinámicas. [16]

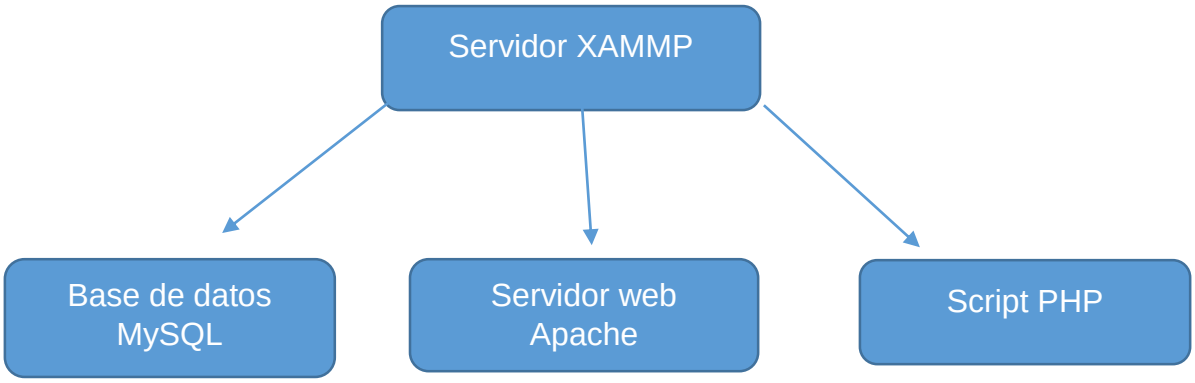


Figura 16.*Composición del servidor XAMPP.*

8.2.2. Características y requisitos

XAMPP solamente requiere descargar y ejecutar un archivo zip, tar, o exe, con unas pequeñas configuraciones en alguno de sus componentes que el servidor Web necesitará. [16]

XAMPP se actualiza regularmente para incorporar las últimas versiones de Apache/MySQL/PHP y Perl. También incluye otros módulos como:OpenSSLy phpMyAdmin. Para instalar XAMPP se requiere solamente una pequeña fracción del tiempo necesario para descargar y configurar los programas por separado. [16]

8.2.3 Base de datos MySQL

MySQL es un sistema gestor de bases de datos, DBMS (Date Base Management System), muy conocido y ampliamente usado por su simplicidad y notable rendimiento. [17]

Para el desarrollo de la página web se utilizó una base de datos MySQL con el nombre de **Parking**, en donde se crearon las tablas para organizar la información que se fue recopilando y así aplicarla a la página.

8.2.3.1 Configuración

La instalación y configuración de la base de datos, se realizó por medio de la plataforma Linux. Para ejecutar la instalación de la base de datos mysql, se debe tener en cuenta los paquetes mysql-server, mysql-common y mysql-client mediante apt-get.

sudo apt-get install mysql-server mysql-common mysql-client

La base de datos mysql, presenta un script para el arranque y para detenerse por medio de la siguiente línea.

sudo /etc/init.d/mysql restart
sudo /etc/init.d/mysql stop

El archivo encargado de la configuración de la base de datos mysql es:

/etc/mysql/my.cnf

Por medio del anterior, se pueden configurar gran variedad de aspectos, como el lugar de almacenamiento de los datos, el puerto que se va a usar y algunas otras configuraciones generales.

Es importante establecer una contraseña root para la modificación de la base de dato, entonces se inicia el MySQL

sudo /etc/init.d/mysql start

Luego se debe iniciar el cliente de mysql como usuario root y en el momento en que se observe el prompt de mysql (mysql>), se realiza una orden grant para crear la contraseña de root:

```
mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 4.0.20-log
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> grant all privileges on to root@localhost identified by 'secreta' with grant
option;
Query OK, 0 rows affected (0.00 sec)
mysql> quit
```

Con lo anterior, se ha logrado establecer como contraseña de root la palabra 'secreta'. En el momento en que se desee entrar de nuevo, se agrega la opción –p para que exija la contraseña root.

```
mysql -u root –p
```

En la siguiente imagen se observa claramente los aspectos de la configuración de la base de datos MySQL para realizar la página web.

```php
<?php  if ( ! defined('BASEPATH')) exit('No direct script access allowed');

$active_group = 'default';
$active_record = TRUE;

$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'root';
$db['default']['password'] = '';
$db['default']['database'] = 'parking';
$db['default']['dbdriver'] = 'mysqli';
$db['default']['dbprefix'] = '';
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = '';
$db['default']['char_set'] = 'utf8';
$db['default']['dbcollat'] = 'utf8_general_ci';
$db['default']['swap_pre'] = '';
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;
```

8.2.3.3 Modelo entidad relación

A continuación se muestra el modelo entidad relación de las diferentes tablas que contiene la base de datos; con este diseño se puede entender de una manera más clara los datos que contiene cada tabla y la relación que hay entre ellos.

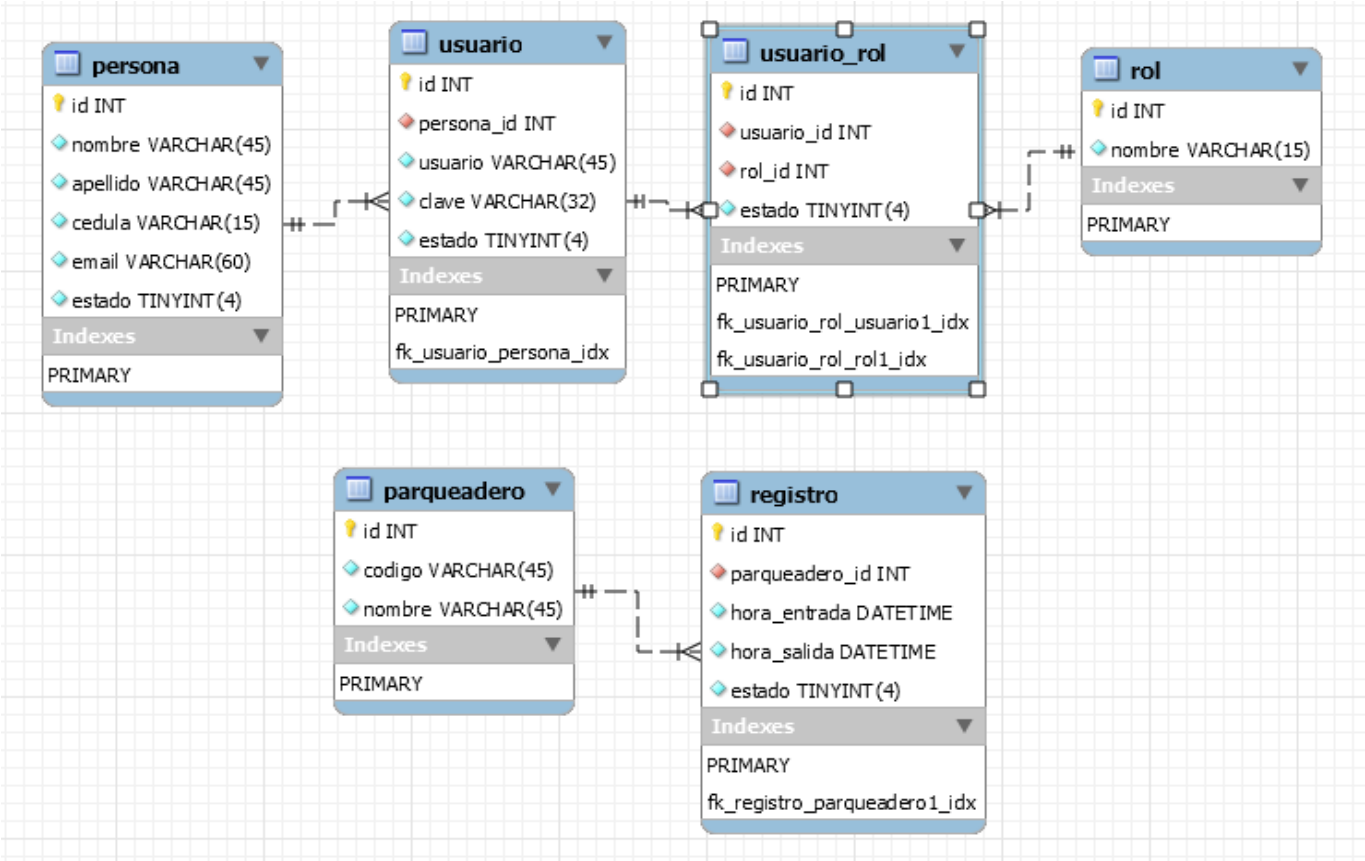


Figura 17. Modelo entidad relación

La primera tabla tiene como entidad **persona**, allí encontramos los diferentes atributos que hacen relación al contenido de cada entidad, observando los siguientes campos alfanuméricos de tipo Varchar (longitud variable): nombre, apellido, cedula y email, y el campo numérico de tipo Tinyint (un byte): estado

La segunda tabla tiene como entidad **usuario**, donde ya se encuentra una relación con la anterior entidad (**persona**), en el cual se ingresan los datos de usuario y clave para poder acceder a la página web; encontramos los diferentes campos alfanuméricos de tipo Varchar: usuario y clave, y el campo numérico de tipo Tinyint: estado.

La tercera tabla tiene como entidad **usuario_rol** donde ya se encuentra la relación con las dos anteriores entidades (persona y usuario), y también posee el campo numérico de tipo Tinyint: estado; es donde se ingresan los tipos de usuario, que puede ser usuario 1 como cliente o usuario 2 como administrador.

La cuarta tabla tiene como entidad **rol,** donde solo encontramos el campo alfanumérico de tipo Varchar: nombre; donde se especifica el tipo de usuario, ya sea usuario 1 o usuario 2.

La quinta tabla tiene como entidad **parqueadero**, donde encontramos los campos alfanuméricos de tipo Varchar: código y nombre, en el cual el nombre es el número del parqueadero y el código es la cantidad de plazas.

La sexta tabla tiene como entidad **registro**, allí encontramos los diferentes campos alfanuméricos de tipo Datetime: hora de entrada y hora de salida y el campo numérico de tipo Tinyint: estado.

8.2.4 Servidor Apache

8.2.4.1 Concepto

El servidor HTTP Apache es un servidor webHTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web. [18]

8.2.4.2 Características

Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos HTTP. Presenta las siguientes características:

- Multiplataforma.
- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.
- Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor. [19]

8.2.4.3 Configuración.

Con la siguiente línea, se instalan los archivos correspondientes para que el servidor Apache funcione correctamente.

 # apt-get install apache2

Para realizar la correcta configuración de apache2, se busca la carpeta /etc/apache2. Luego se procede a detectar el archivo principal llamado /etc/apache2/apache2.conf. Es importante tener en cuenta, que se debe realizar una copia de seguridad al modificar las configuraciones.

Se pueden hacer cambios de infinidad de parámetros. En el momento de colocar en marcha el servidor, se edita el archivo apache2.conf e inserta el siguiente parámetro:

ServerNamewww.islapaloma.com.

Si en tal caso, no se cuenta con un servidor DNS, se puede editar el archivo /etc/hosts y añadir la dirección IP del servidor e indicar el nombre. Como ejemplo:

192.168.1.230www.islapaloma.com.

Finalmente se establece el scritp de arranque y parada:

/etc/init.d/apache2 restart
/etc/init.d/apache stop

8.2.5 PHP

8.2.5.1 Concepto.

PHP (acrónimo recursivo de *PHP: Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Lo que distingue al PHP, que esta de lado del cliente como Javascript, es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga. [20]

8.2.5.2 Características.

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una Base de Datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial, utilizando la enorme cantidad de módulos.
- Es libre, por lo que se presenta como una alternativa de fácil acceso sin licencia.
- Permite aplicar técnicas de programación Orientada a Objetos.

- No requiere definición de tipos de variables, aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.

- Tiene manejo de excepciones (desde PHP5). [21]

8.2.5.3 Configuración.

El primer paso, es instalar PHP en el servidor Apache establecido, por medio de la apt-get.

Sudo apt-get install libapache2-mod-php5

Posteriormente, se procede a conectar la base de datos MySQL por medio de PHP, instalando el módulo php5-mysql.

Sudo apt-get install php5-mysql

Para realizar la configuración de PHP, se usaría el archivo:

/etc/php5/apache2/php.ini

Por medio del archivo anterior, se pueden configurar una cantidad de parámetros tales como: modo seguro, mostrar errores, tiempo de ejecución de script, tamaño de datos, tamaño de archivos y activación al acceso de datos.

Después de realizar la configuración, se reactiva el apache y se efectúa la verificación de que el servidor y PHP estén trabajando correctamente, creando una página PHP a través de la función phpinfo().

9. INTERFAZ Y FUNCIONAMIENTO DE LA PÁGINA WEB.

Para el desarrollo de la interfaz web se utilizó la herramienta HTML 5, definiendo la estructura básica del contenido como son: textos, imágenes, videos entre otros. Por medio del entorno de programación Netbeans, se realizó todo el código necesario.

9.2 Dominio

Para empezar a identificar el contenido de la página de web, se debe tener claro el concepto de un dominio y para qué se utiliza. La dirección de internet o dominio, es un grupo de caracteres que se utilizan para reconocer la ubicación concreta de la página en la red, mediante esta dirección se establece en qué lugar del servidor web de encuentra.

Es primordial entender que cada dominio es exclusivo, es decir no es posible que dos personas o usuarios que accedan a este servicio, tengan el mismo nombre de dominio.

En realidad, las máquinas en Internet se identifican mediante un número único que recibe el nombre de dirección IP(siglas de 'Internet Protocol', Protocolo de Internet). Para hacernos una idea, un ejemplo de dirección IP puede ser 223.142.1.90. De esta forma, el navegador localiza el servidor en el que se encuentra la página que buscamos. Dado que las direcciones IP no resultan fáciles de recordar para los usuarios, se creó un sistema de nombres, que asocia a estas direcciones unos caracteres de texto, normalmente dos palabras separadas por un punto. La palabra de la izquierda es más específica y la de la derecha más genérica (".com", ".es", etc.). Los dominios fueron creados para sustituir las direcciones IP por palabras, que siempre resultan más fáciles de recordar. Por eso, se dice que un dominio es una forma básica de localizar un ordenador en Internet. [22]

En el desarrollo de la página web se utilizó el dominio " www.parqueadero.work", el .work hace referencia a que es una página trabajo o para la realización de un proyecto como en este caso.

9.3 Login

En el momento en que se realiza el proceso de ejecución del dominio en el navegador web, se invoca la página diseñada para el proceso de visualización y la primera imagen que se observa es la interfaz del login. En un ambiente de seguridad informática, el login es utilizado para obtener un control a la hora del acceso individual de la página web mediante una identificación de usuario y contraseña.

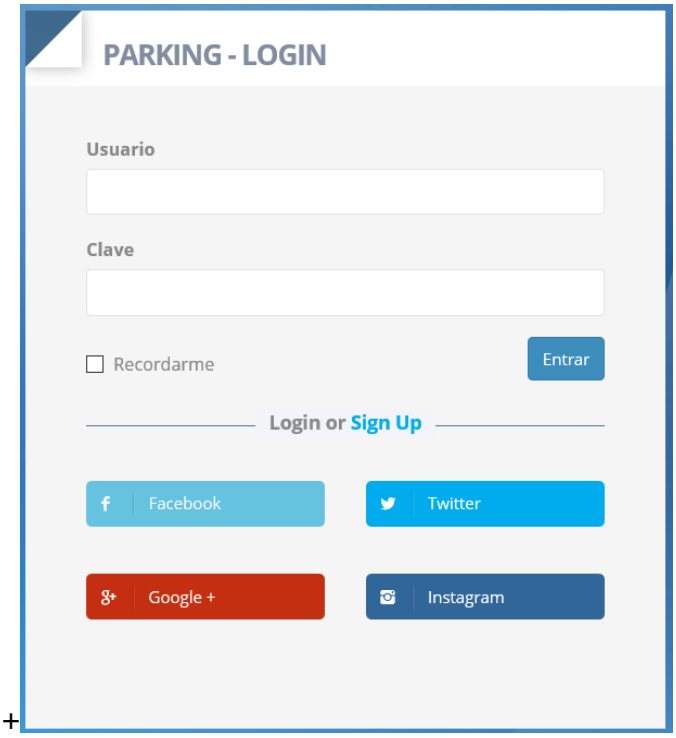


Figura 18. *Login de la página web*

La página web permite el acceso de dos formas: como administrador y como cliente. Al ingresar como administrador, se tiene permiso a todos los privilegios y opciones que ofrece la página, que en este caso sería el registro que se lleva de los tiempos en el que los automóviles permanecen en las plazas de parqueo.

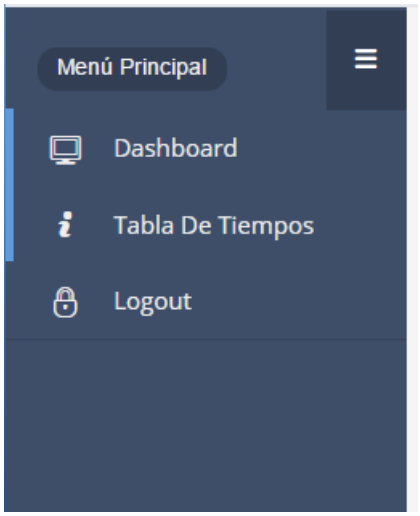


Figura*19. Vista del menú principal por el Administrador*

En el caso de los clientes, solo van a tener acceso a la interfaz de las plazas de parqueo, de esta manera se reserva el registro de tiempos, solo a la persona encargada de la administración, brindando seguridad a los demás clientes.

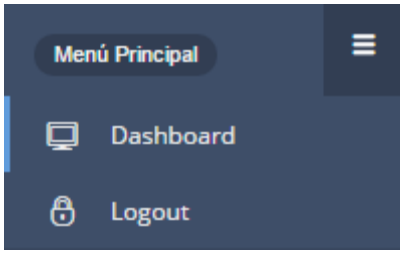


Figura 20. *Vista del menú principal por el Cliente*

Durante el proceso de login se pueden presentar dos advertencias:

1. “Los campos no pueden estar vacíos”, esto se presenta en el momento en el que el individuo que desea ingresar a la página, no rellena con ningún carácter los espacios de usuario y contraseña.

2. “Usuario y/o clave inválidos”, esto se presenta en el momento en el que el individuo que desea acceder a la página, ingreso mal los datos en los espacios requeridos.

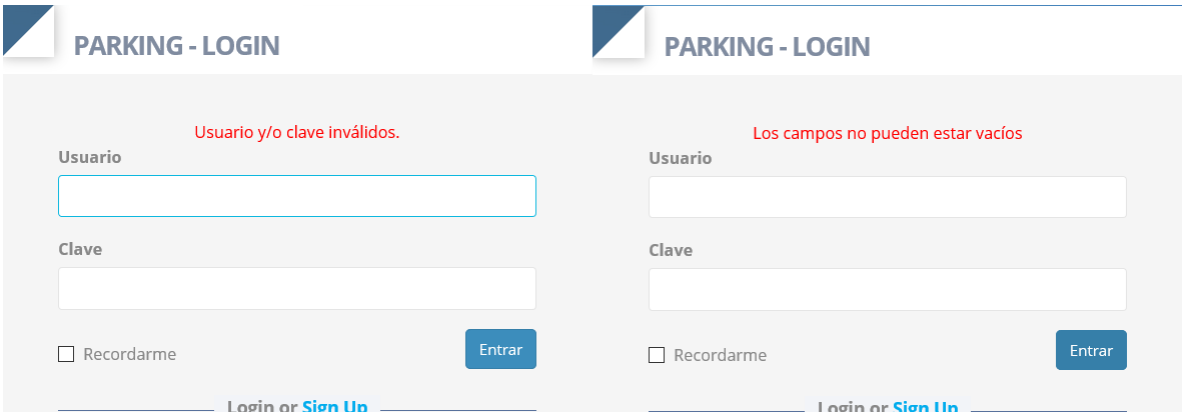


Figura 21.*Advertencias en el acceso a la página*

9.4 Interfaz de la página web

La explicación de la página web se realiza por medio de la interfaz de administración, debido a que esta tiene las funciones completas que la página puede ofrecer.

9.4.1 Menú principal

El menú principal de la página es muy sencillo y consta de 3 elementos:

1. El dashboard, que se encarga de devolver al usuario a la interfaz principal, en este caso las plazas de parqueo.

2. Tabla de tiempos, se encarga de enviar al administrador de la página, a la interfaz de la tabla de registro de entrada y salida de cada uno de los usuarios.

3. Logout, consiste en cerrar el acceso personal a la página web, enviando al individuo de regreso al login.

9.4.2 Plazas de parqueo

El primer análisis se hará con respecto a la representación de las plazas de parqueo, que se mostrará a continuación.

Figura 22. *Interfaz de las plazas de parqueo libres*

La distribución de las plazas de parqueo van en el orden mostrado en la figura anterior, cada uno de los recuadros contienen el estado de la plaza, que se representa de color verde cuando están libres, y en la esquina superior izquierda se observa el número del parqueadero para su identificación en el modelo arquitectónico.

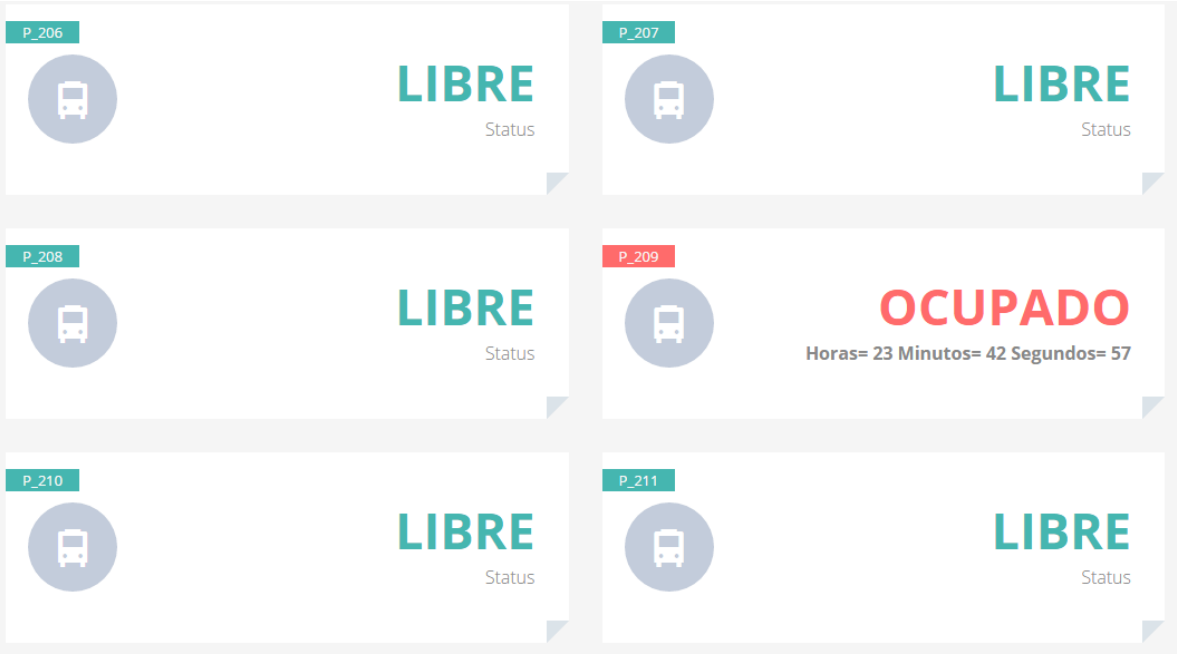


Figura 23.*Interfaz con plaza de parqueo ocupada*

En el instante en el que un objeto se ubique en una plaza de parqueo, inmediatamente su estado se cambia a ocupado. Esta representación se realiza por medio un color rojo. Un poco más abajo de la palabra que indica el estado del parqueadero, se genera un cronometro que va a estar funcionando durante todo el momento en el que la plaza se encuentre en esa condición, de esta manera el usuario va a llevar un seguimiento de cuánto tiempo lleva allí. Esto es de gran utilidad en el momento en que la administración decida hacer cobro del servicio, ya que el usuario va a ser consciente del tiempo exacto en el que ha estado allí, a la hora de evitar inconvenientes.

9.4.3 Tabla de tiempos

Es una función extra que se añadió al sistema de monitoreo, con el fin de llevar un registro de entrada y salida de cada uno de los vehículos. En el momento en que la administración quiera cobrar el servicio de parqueo, este registro va a ser de una gran ayuda.

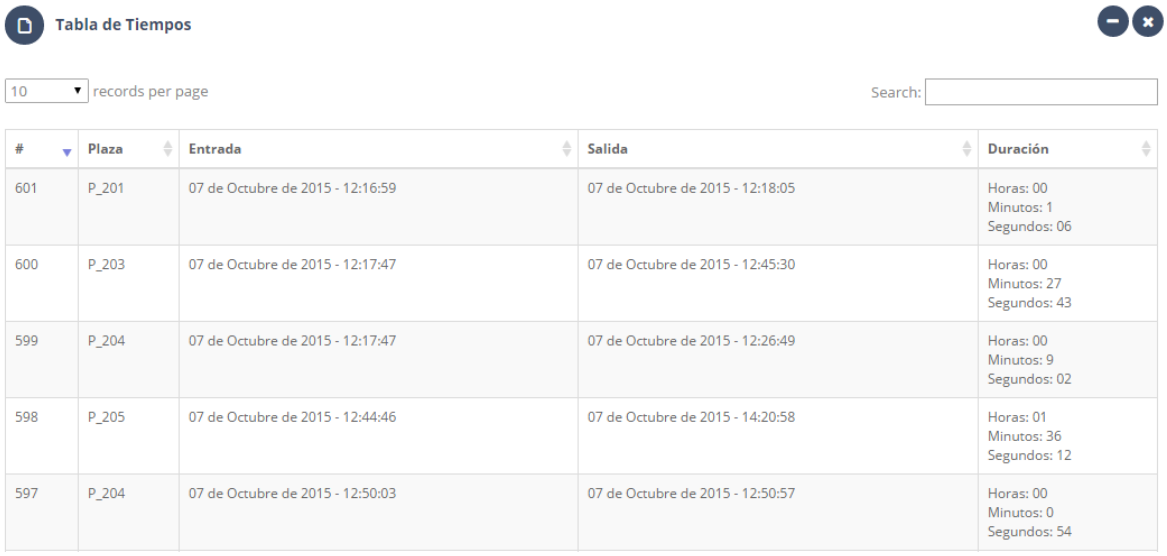


Figura 24.*Tabla de tiempos de la página web*

La tabla anterior se resume en 5 columnas.

- El primer espacio hace referencia al número de registro, de esta forma se lleva un conteo de todos los datos almacenados.
- La segunda columna muestra el nombre de la plaza, identificadas con números del 200 al 219, con el fin de determinar la ubicación en el modelo arquitectónico.
- El tercer espacio representa el año, fecha, mes y hora en el que el vehículo ingresó a la plaza de parqueo,
- La cuarta columna informa en el mismo formato de la tercera, el momento en el que el vehículo abandona el espacio de parqueo, con la particularidad de que si el usuario sigue allí, simplemente se va a visualizar la palabra ocupado.
- El último espacio de la tabla, va a registrar el tiempo exacto en el que el vehículo duró ocupando la plaza correspondiente.

Figura 25.*Filtrado en la tabla de tiempos de la página web.*

La tabla de tiempos incluye una función muy útil a la hora de organizar todos los datos del registro. Como se puede observar en la figura 28, en la casilla search se realiza un filtrado de todos los datos almacenados. En el momento en el que la tabla maneje grandes cantidades de información, se le saca provecho a esta herramienta, visualizando datos de una plaza de parqueo en especial, o el movimiento que tuvo el parqueadero en ciertas horas del día y en general ordenar la información de la manera que el administrador lo requiera.

Aparte de lo mencionado anteriormente, la página web brinda la fecha y hora exacta del día que está en transcurso y a su vez ofrece un espacio con la posibilidad de observar mensajes dinámicos informando la cantidad de plazas libres u ocupadas para los clientes.

Figura 26. *Información de fecha, hora y ocupación*

10. CONSTRUCCIÓN Y EVALUACIÓN DELOS SISTEMAS DE MONITOREO Y VISUALIZACIÓN.

En esta sección, se muestra de manera detallada la construcción del modelo arquitectónico, configuración y acople del sistema en general. Se explicará primero el diseño del parqueadero a escala de 1:25, la ubicación respectiva de los sensores y el montaje de la tarjeta programable. Posteriormente se evalúa la integración entre hardware y software con el protocolo de comunicación y los sistemas que se utilizaron para que se pudiera visualizar de forma precisa y en tiempo real lo que ocurre en el modelo arquitectónico construido.

10.1 Construcción del parqueadero a escala.

Para cumplir los objetivos propuestos en este proyecto de grado, se debió construir un parqueadero a escala con 20 plazas de parqueo. A continuación se presenta el plano arquitectónico real del parqueadero donde se obtuvo la porción de plazas que se implementaron en el modelo desarrollado.

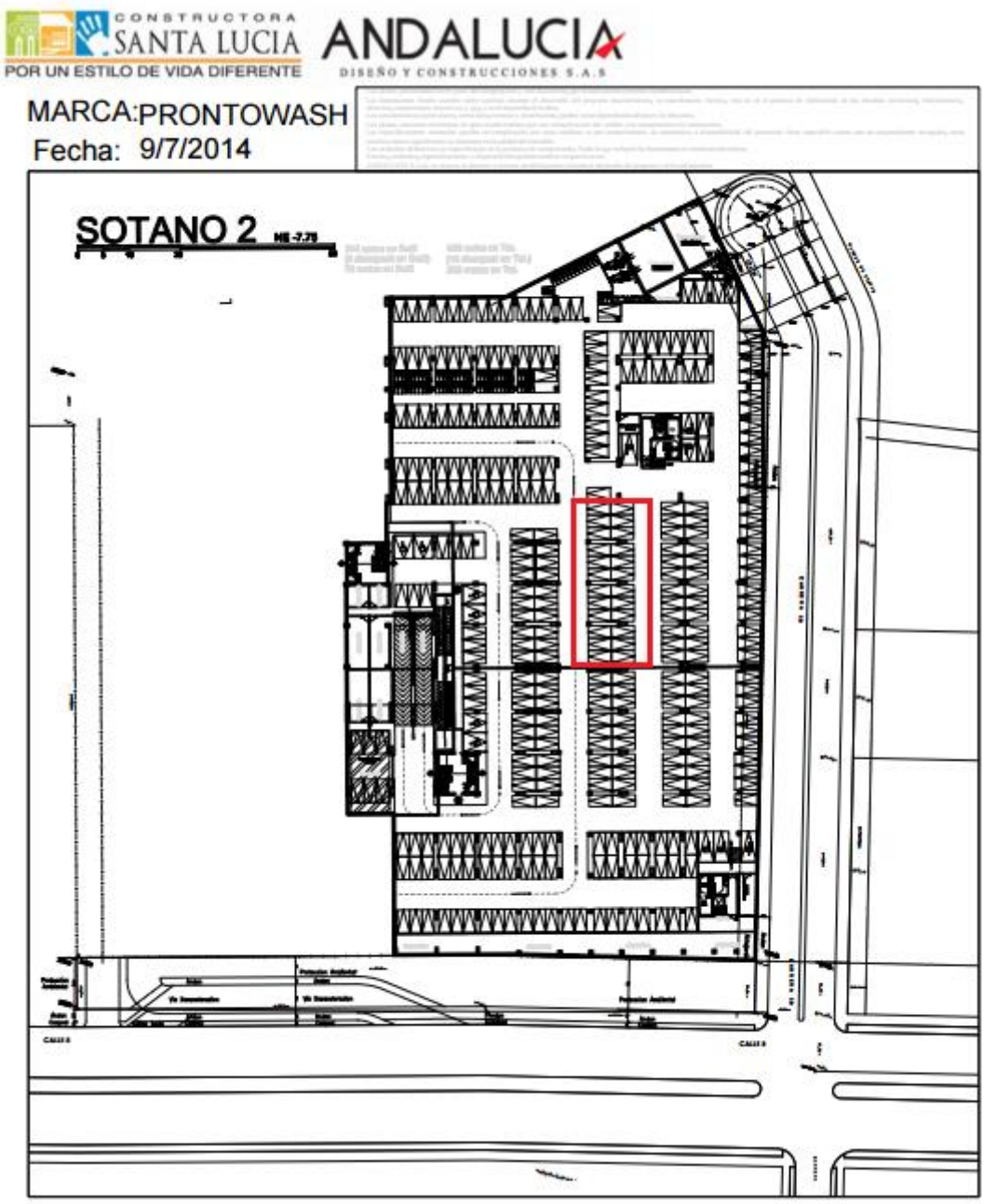


Figura 27.*Plano real del parqueadero Santa Lucía Plaza*

Del plano real tomamos una sección de 24 plazas de parqueo, ubicadas en el Sótano 2, de las cuales utilizamos 20, numeradas de la 200 a la 223 con el fin de identificar que parqueadero corresponde a cada recuadro de la interfaz web.

Para la construcción del modelo arquitectónico utilizamos una placa de madera MDF con dimensiones de 1.80 cms de largo por 86 cms de ancho y 8 mm de grosor.

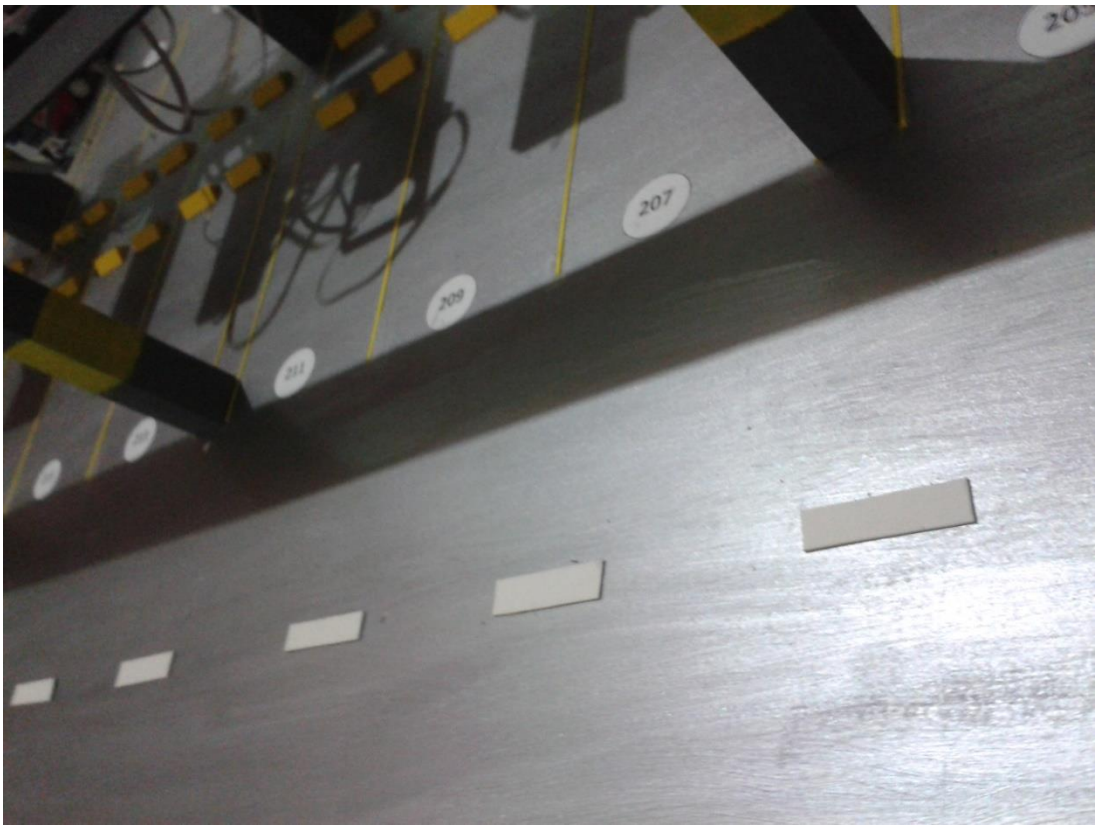


Figura 28.*Numeración plazas de parqueo*

Como se puede observar en la figura anterior, la distribución de la numeración de las plazas de parqueo permite identificar que por un carril están las plazas pares y por el otro las impares.

En la placa base del modelo arquitectónico se ubicaron las plazas una detrás de otra, como se puede observar en la figura del plano real del parqueadero. Cada plaza de parqueo tiene un tamaño real de 5mts de largo por 2.5mts de ancho y 3mts de alto; dependiendo del tamaño del automóvil que se utilizó para realizar las pruebas se determinó que la escala adecuada para realizar el modelo arquitectónico es de 1:25, de esta manera, en base a las medidas reales se realizó la conversión para obtener una plaza de parqueo a escala de 20cms de largo por 10cms de ancho y 12cms de alto.

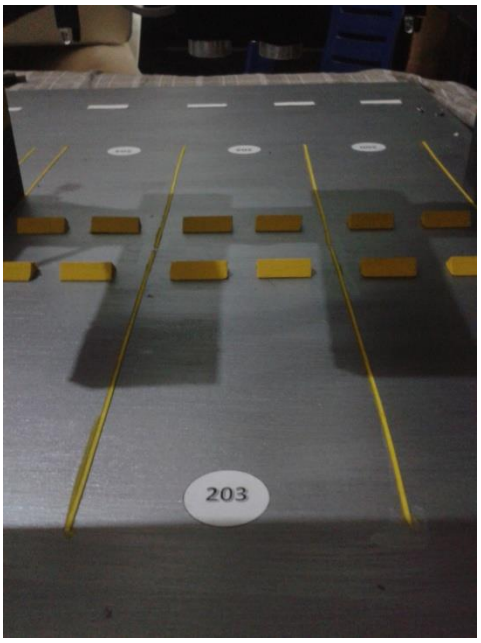


Figura 29.*Representación plaza de parqueo a escala*

Después de establecer la distribución de las plazas de parqueo se procedió a realizar la ubicación de cada uno de los sensores. Por funcionamiento y comodidad se determinó situar los sensores en la parte superior sobre pequeñas cajas de 20cms de largo por 5cms de ancho, ajustadas a la medida de cada sensor.

Figura 30. *Cajas de la ubicación de los sensores*

En el centro de la distribución de las plazas de parqueo, se ubicó la tarjeta programable con el fin de que sea equidistante a cada uno de los sensores. La conexión se realiza por cable ribbon, y de esta manera se logra un ensamble mucho más fácil y sin el caos de una gran cantidad de cables sueltos. Al utilizar este tipo de cable, se debió realizar previamente una extensión shield para el Arduino Mega, a través de una placa de pruebas universal y así acoplar los conectores a los pines de la tarjeta que le corresponden.

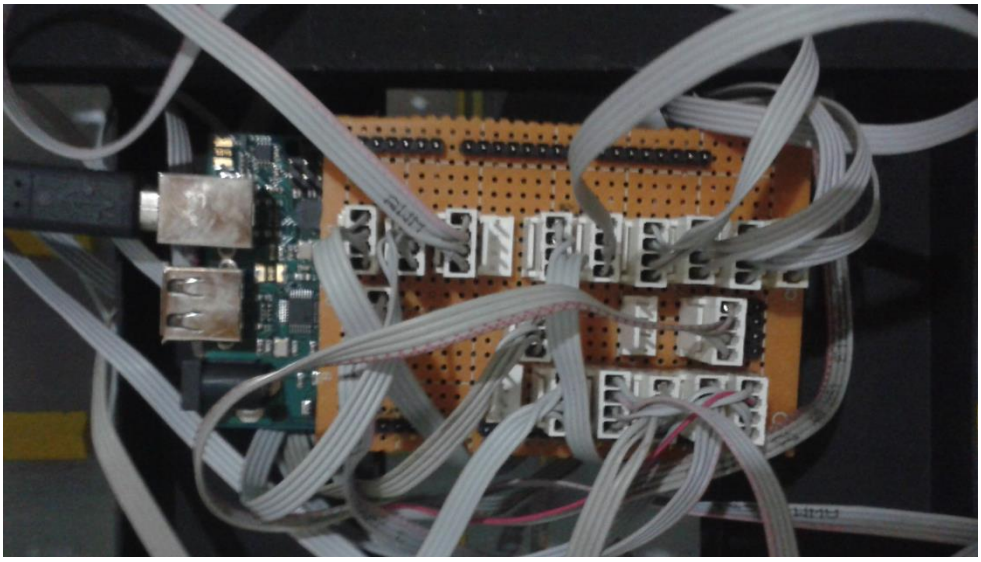


Figura 31.*Conexiones de la tarjeta programable*

A un costado de la placa de madera se simulo una de las entradas del parqueadero real, cada acceso con 10cms de ancho en base al automóvil que se utilizó para las pruebas. Esta entrada consta de dos cajas, en la cual encontraremos primero la pantalla LCD donde se visualiza el contador de carros, seguido de la talanquera móvil en donde observamos los sensores infrarrojos y servomotores, luego se encontraran los conos para la señalización y separación de carriles.

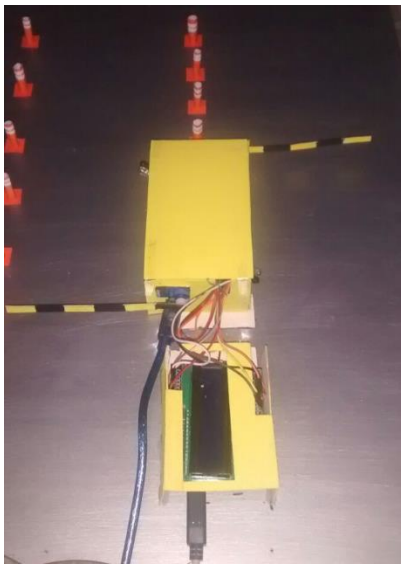


Figura 32. *Contador y talanquera móvil*

Después de recopilar y unir cada una de las partes anteriormente mencionadas, se obtiene el modelo arquitectónico del parqueadero con 20 plazas, listo para ser acoplado con el sistema de visualización.

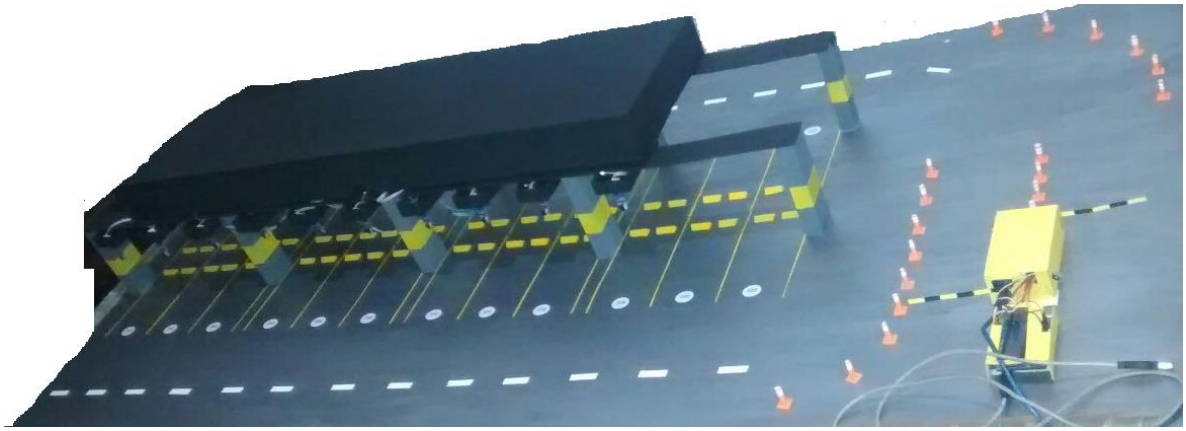


Figura 33.*Modelo arquitectónico del parqueadero*

La alimentación de este sistema, se hace a través de la tarjeta programable por medio de un adaptador de 12 voltios conectado a la red domiciliaria eléctrica.

10.2 Evaluación del Acople entre hardware y software.

Para una correcta evaluación del sistema, se debe asegurar que lo que ocurra en el modelo arquitectónico se va a poder visualizar en la interfaz web diseñada en tiempo real. Como se mencionó en la sección 3, este sistema cuenta con un protocolo de comunicación serial implementado de dos formas, una física por medio de la conexión USB y otra virtual a través del aplicativo en Java. Estos dos métodos permiten la comunicación entre el dispositivo de detección y la página web.

El primero paso en la comunicación, es establecer la conexión física entre la tarjeta programable y el equipo de cómputo a través del cable USB.

Es oportuno aclarar tres puntos:

En el caso de las pruebas, se utiliza un computador portátil como equipo de cómputo, pero esta no es la única posibilidad para realizar la comunicación. Se señala como equipo de cómputo, cualquier mecanismo o material de computación con puerto USB y que tenga la capacidad de ejecutar un aplicativo en java. Así que desde una tarjeta raspberry PI se puede realizar esta función.

Gracias a que la interfaz de visualización se encuentra alojada en la red de internet, a través de un servidor web, se puede obtener un monitoreo totalmente remoto, razón por la cual no es necesario implementar un puesto de control para hacer dicha labor. Cada usuario puede realizar un seguimiento desde cualquier parte del mundo sin necesidad de una conexión física, solo con el acceso a internet.

En el caso de que se manejen más plazas de parqueo, se pueden conectar muchas más tarjetas programables a un solo equipo de cómputo a través de un hubs USB, lo que a su vez permite que la comunicación se haga con un mayor alcance.

Lo anterior se definió para comprobar, que la comunicación física establecida no es una limitante en el sistema diseñado y que este tipo de protocolo puede cumplir

perfectamente con las labores requeridas, permitiendo un ahorro económico con una menor cantidad de dispositivos.

Después de establecer el sincronismo con el equipo de cómputo, se ejecuta la etapa virtual de comunicación por medio de la aplicación en java.

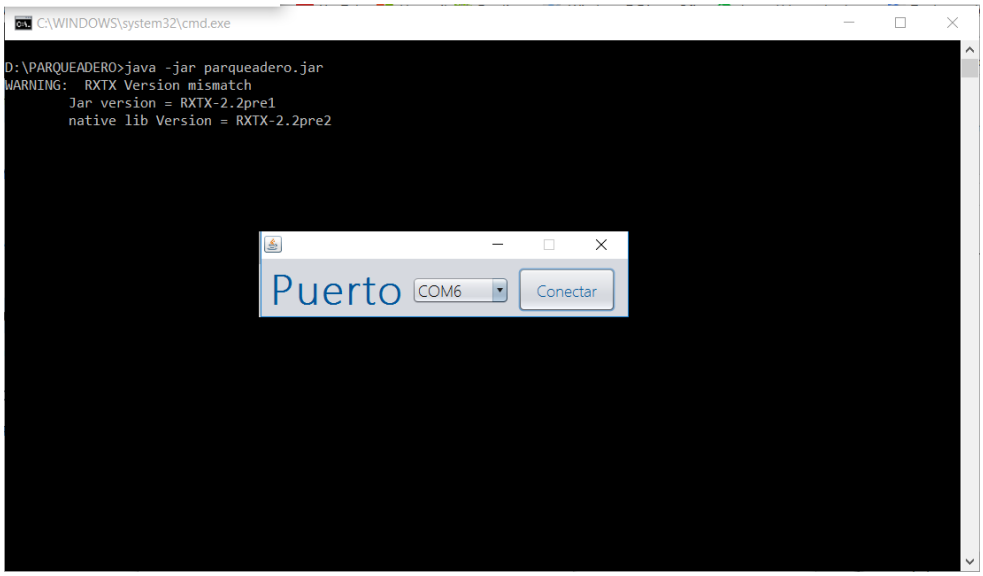


Figura 344. *Ejecución de la aplicación java.*

En el momento en que se genera el puerto COM, se procede a realizar la conexión URL presionando el botón “conectar”.

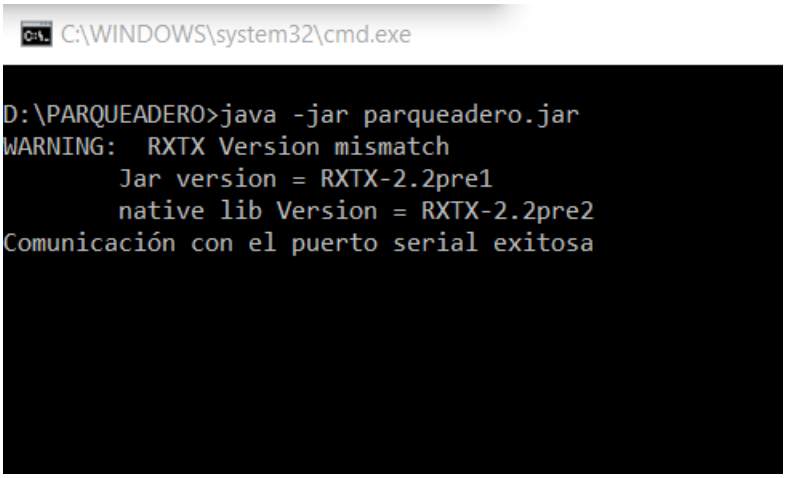


Figura 35.*Cuadro de visualización de flujo de datos.*

En este momento se ha establecido el acople y la comunicación entre los dos sistemas. En el instante en que los sensores detecten algo, se va a poder visualizar inmediatamente en la interfaz web.

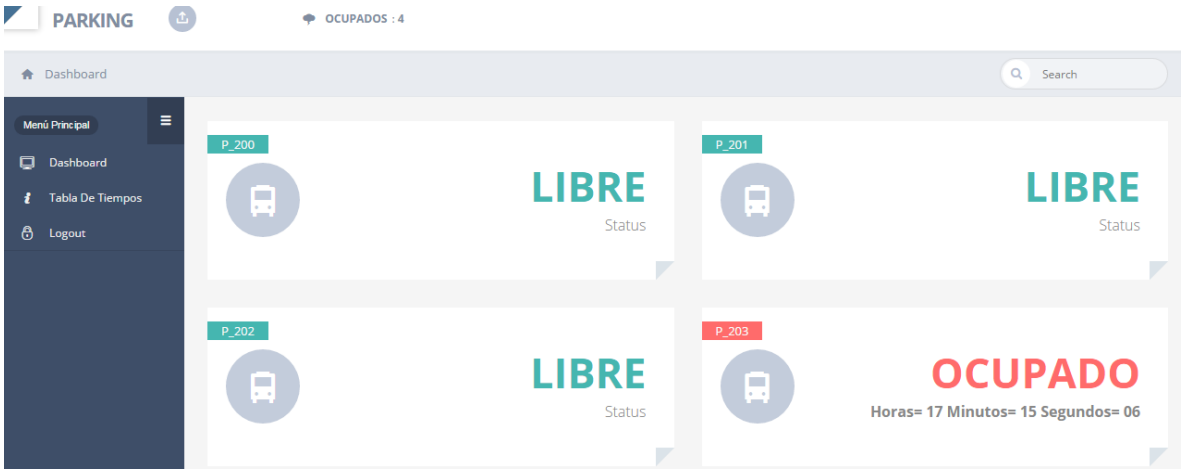


Figura 36.*Visualización de plaza de parqueo ocupada.*

Anexo a este documento, se puede observar un video que permite examinar de una manera más detalla y verídica el desempeño del sistema.

11. CONCLUSIONES

La utilización de los sensores ultrasonidos brindó una gran ventaja en el momento de obtener los datos satisfactoriamente, debido a su alta resolución y sus pocas zonas ciegas. Fue una herramienta muy versátil, ya que realiza la detección sin contacto físico y a diferencia de otro tipo de sensores, no presentó ningún problema en condiciones ambientales críticas, porque independientemente de la luz, material del obstáculo, color o transparencia, siempre ofreció una toma de datos correcta.

Se obtuvo que al implementar una red de sensores, se deba tener en cuenta el espacio disponible, ya que existe la posibilidad de que se produzcan problemas cuando los sensores están muy cerca uno del otro, en el caso de este proyecto se presentó este inconveniente debido a la cantidad de sensores ubicados en un espacio reducido, por tal razón se utilizaron los sensores infrarrojos evasor de obstáculos, debido que su tamaño fue más adecuado para implementarlo en el modelo arquitectónico, además al ser sensores digitales fue mucho más practico realizar la detección del automóvil.

Se comprobó que las placas electrónicas programables como Arduino, son una inmensa ventaja en el momento de diseñar e implementar este tipo de proyectos. Producen unas excelentes prestaciones; por su gran diversidad como la inmunidad al ruido y ofreciendo una variedad de funciones en un solo dispositivo, lo que hace que la reducción de costos a la hora de la implementación sea notable. En una sola tarjeta, se pudo ubicar los 20 sensores sin ningún inconveniente y a su vez brindó la posibilidad de realizar una comunicación segura con el equipo de cómputo, reduciendo apreciablemente la densidad de elementos en el sistema.

Java fue una herramienta muy útil a la hora de enlazar los datos con la página web, sobre todo porque es un mecanismo multiplataforma que permite su utilización en cualquier tipo de sistema operativo y de una manera muy eficiente y rápida. Esto permite tener un abanico de muchas posibilidades a la hora de elegir el equipo de cómputo que se va a conectar con el sistema.

.

El uso de una página web, ofreció la mejor ventaja del sistema de detección, porque permitió que el monitoreo del modelo, se realizara de una manera totalmente remota y además que fuera asequible para todos los usuarios interesados.

Se comprueba como Ajax es unas de las mejores técnicas de desarrollo web, debido a que le brinda a la página una ventaja destacable con respecto a las páginas convencionales, ejecutando una actualización asíncrona constante de la interfaz web, sin necesidad de que sea recargada por un operador.

12. TRABAJOS FUTUROS

Implementar sensores inalámbricos, con el fin de reducir el cableado en el modelo, de esta manera asegurar un mayor alcance de transmisión y que la vida útil del sistema aumente.

Se podría implementar no solo una herramienta de detección, sino también una de seguridad a través de un sistema por visión de computador en la entrada del parqueadero, con el propósito de tomar imágenes detallada de cada vehiculó que ingresa, obteniendo número de placa, color y forma del automóvil, de esta manera se controla la entrada de los usuarios.

Para mejora de este proyecto, se puede desarrollar un sistema en la entrada del parqueadero que verifique si existen plazas disponibles e imprima un ticket con la información de la plaza asignada y la hora de ingreso del vehículo. A su vez que brinde una opción en la página web en el caso de que el usuario desee reservar un puesto con un costo agregado.

13. REFERENCIAS BIBLIOGRÁFICAS

[1] K. Soria, «HC-SR04 Sensor ultrasonico,» 2013.

[2] Vistronica, «Reseña sensor infrarrojo evasor de obstaculos,» [En línea]. Available: http://www.vistronica.com/sensores/proximidad-y-distancia/modulo-sensor-infrarrojo-evasor-de-obstaculos-detail.html?product_rewrite=modulo-sensor-infrarrojo-evasor-de-obstaculos.

[3] C. electronicos, «Sensor de infrarrojos (emisor y receptor),» [En línea]. Available: http://www.circuitoselectronicos.org/2010/05/sensor-de-infrarrojos-emisor-y-receptor.html.

[4] Arduino, «Arduino Mega ADK,» 2014.

[5] Arduino, «Arduino MEga ADK,» 2014.

[6] RO-BOTICA, «Placa controladora Arduino Mega ADK,» Barcelona, 2015.

[7] «Servomotores,» [En línea]. Available: http://platea.pntic.mec.es/vgonzale/cyr_0204/ctrl_rob/robotica/sistema/motores_servo.htm.

[8] E. Lopez Pérez, «EL protocolo USB,» México D.F, 2010.

[9] O. J. Flores, Interfases o estándares Interfases o estándares USB para las comunicaciones en serie..

[10] Universidad Nacional de Cordoba, «Universal Serial Bus, The Easy Way to Plug & Play».

[11] S. I. Torres Rodriguez, Universal Serial Bus, Canarias.

[12] A. Lopez Barragan y E. Gonzales Suarez , «Puertos USB - Bus Serie Universal y descripción de la norma IEEE 1394,» Valladolid, 2003.

[13] ROBOTIX, «serial comunication».

[14] J. J. Gutiérrez, «¿ Qué es un framework web?».

[15] CodeIgniter, «Guia de usuario CodeIgniter,» 2012. [En línea]. Available: http://www.areadepruebas.com.ar/downloads/CodeIgniter_Guia_Usuario_2.1.3.pdf.

[16] «XAMPP».

[17] L. A. Casillas Santillán, M. G. Ginesta y O. Pérez Mora, «Base de datos en MySQL,» Catalunya.

[18] «Hypertext Transfer Protocol».

[19] Opensuce, «Apache».

[20] PHP, «¿Qué es PHP?».

[21] «Programación en PHP».

[22] j. L. Colvée, «Los dominios de internet,» Anetcom, 2014.

[23] D. PÉREZ, «Sensores de distancia por ultrasonido,» 2013.

[24] A. J. Cardenas , «Sensor Ultrasonico,» 2015.

[25] A. Tomar y E. Lim, «Introduction Superspeed USB 3.0 Protocol,» 2011.

[26] Wikimedia, «USB icon».

[27] ModDIY, «USB 2.0 / 3.0 Connectors & Pinouts».

[28] J. Martinez Ladron de Guevara, «Fundamentos de programacion en Java,» Madrid.

[29] B. Alonso, «Entorno de desarrollo Java,» 2014.

[30 Universidad de Murcia, «Manuel Básico de creación de páginas web,» Murcia.
]

[31 A. Ruiz, «¿Cuáles son las características principales del HTML?,» 2014.
]

[32 E. Mandado Pérez y A. Roldan Murillo, SENSORES DE ULTRASONIDO,
] Vigo.

[33 Pablo, Tutorial Arduino – Módulo 3G, Ultrasonidos, Keypad & Led RGB, 2012.
]

[34 J. A. Polania, Estructura de la placa Arduino, Neiva.
]

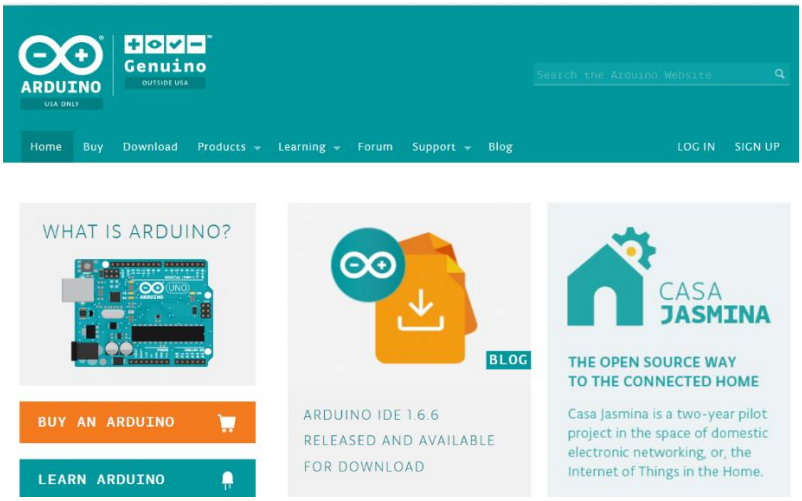
[35 MSrobotics, Arduino Micro, 2015.
]

14. ANEXOS

Anexo 1. Documentación de software utilizado**.**

- **Arduino**

Se usa para el procesamiento de la información este programa, tiene una plataforma de Hardware libre de la misma forma que su entorno de desarrollo, de esta manera Arduino permite la descarga gratuita desde su página principal (www.arduino.cc).



Se utiliza la opción Download para descargar el ejecutable.

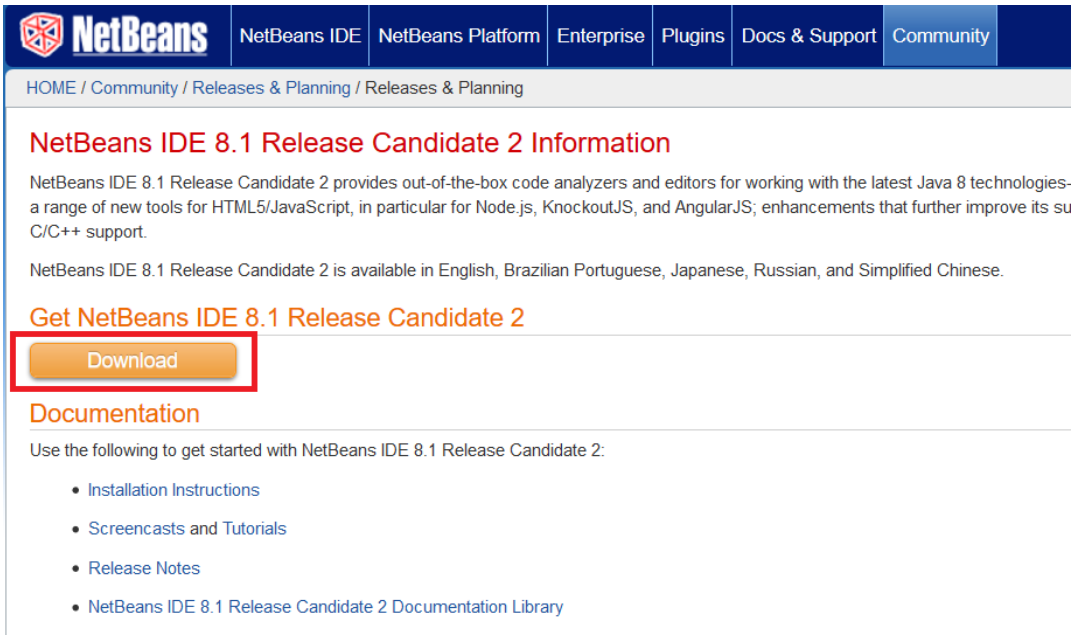
El gestor de descarga del software Arduino nos permite instalarlo a través de su página principal, específicamente en: www.arduino.cc/en/Main/Software dependiendo el sistema operativo que lleves instalado en el equipo se escoge la pestaña de instalación.

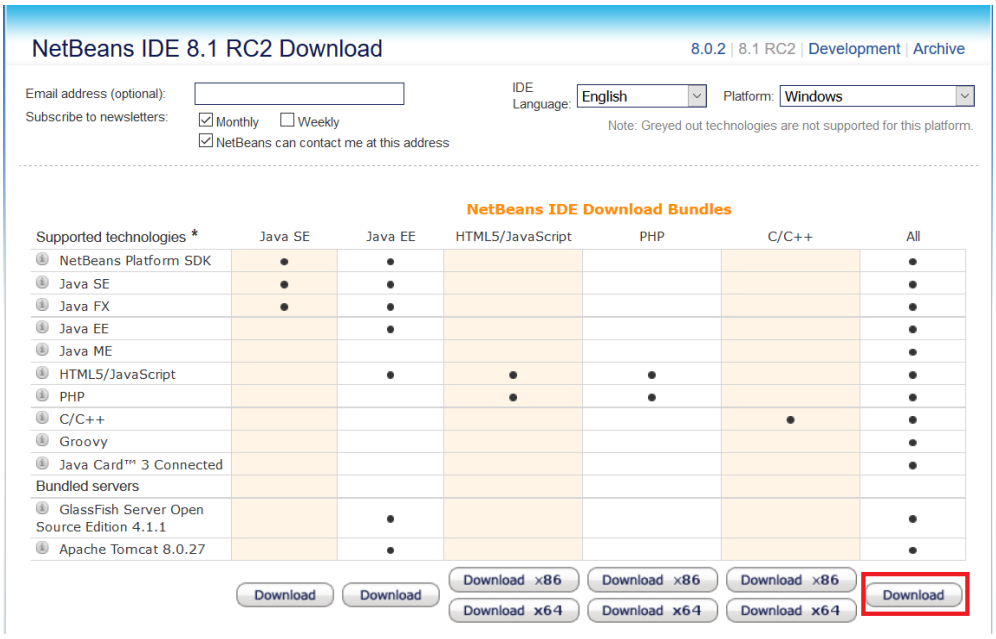
Posteriormente antes de realizar la descarga, la página pide una contribución económica voluntaria. Se da click en “Just download” en el caso de que solo se quiera descargar.

Finalmente genera el archivo ejecutable para guardar o ejecutar.

- **Netbeans**

Este software es totalmente gratuito y es utilizado para realizar la aplicación java y la programación de la página web. A continuación se presenta el proceso de descarga. La página oficial es: https://netbeans.org/.



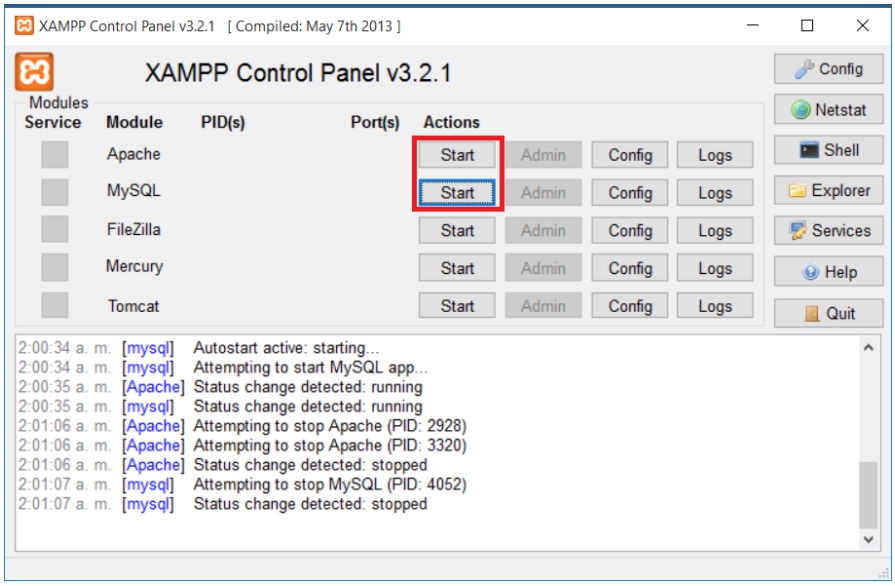


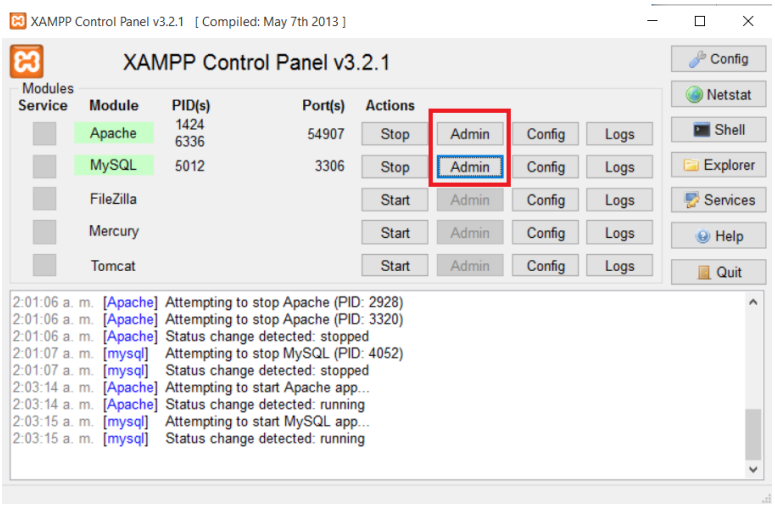
Se escoge la última opción para obtener el programa con la totalidad de funciones.

- **Xampp:**

Distribución de apache totalmente gratuita usada para el manejo del servidor apache y la base de datos mysql. Se descarga a través de la siguiente página web: https://www.apachefriends.org/es/index.html.

Genera el ejecutable, posteriormente se instala y aparece el siguiente recuadro:

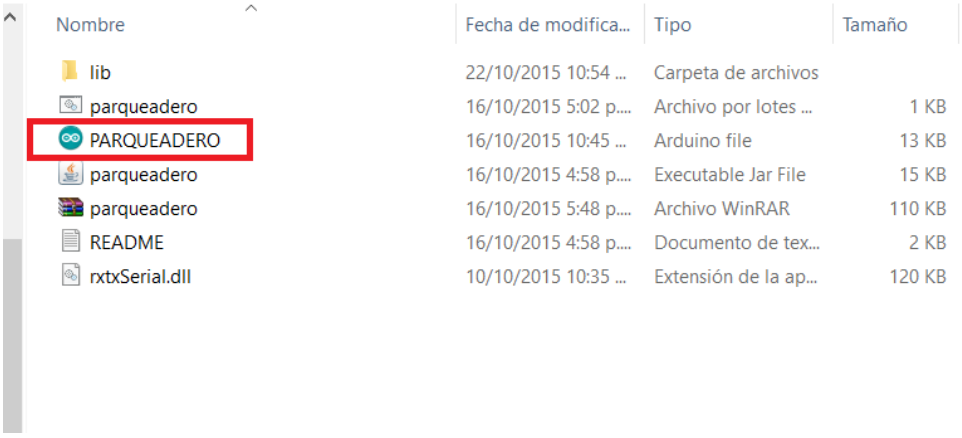




Para administrar la base de datos y el servidor.

Anexo 2. Código de programación de Arduino**.**

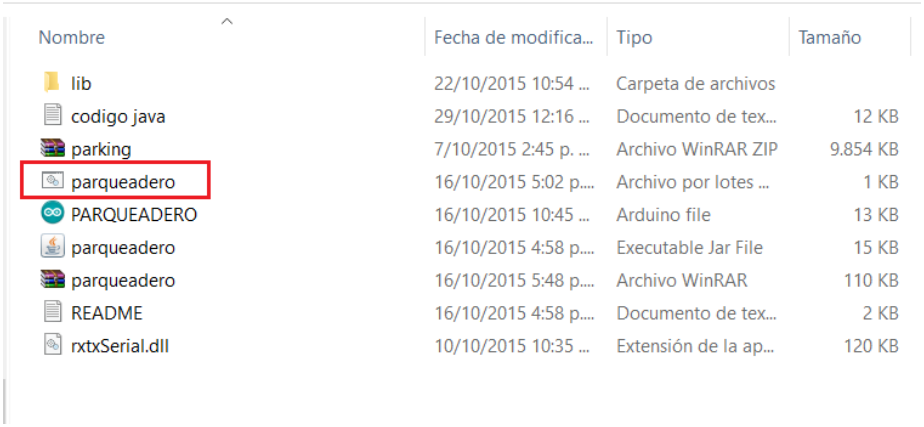
Adjunto a este documento se encuentra una carpeta llamada “PARQUEADERO”, en el momento de abrirla aparecerá lo siguiente:



Al ejecutar ese archivo, aparecerá el código completo con el que se programó la tarjeta.

Anexo 3. Ejecución de la aplicación Java

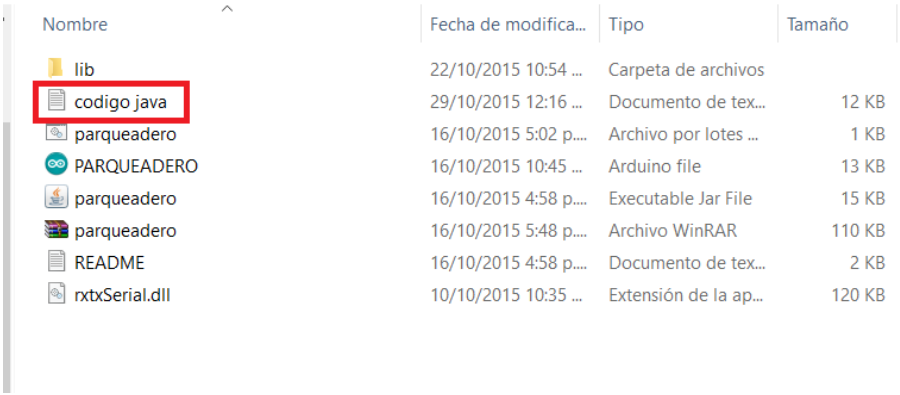
Adjunto a este documento se encuentra una carpeta llamada “PARQUEADERO”, en el momento de abrirla aparecerá lo siguiente:



Al ejecutar el archivo señalado, se abre la aplicación java junto con un recuadro cmd para visualizar el flujo de datos.

Anexo 4. Código de programación para la aplicación Java

Adjunto a este documento se encuentra una carpeta llamada “PARQUEADERO”, en el momento de abrirla aparecerá lo siguiente:



Al ejecutar ese archivo, aparecerá el código completo con el que se programó la aplicación.

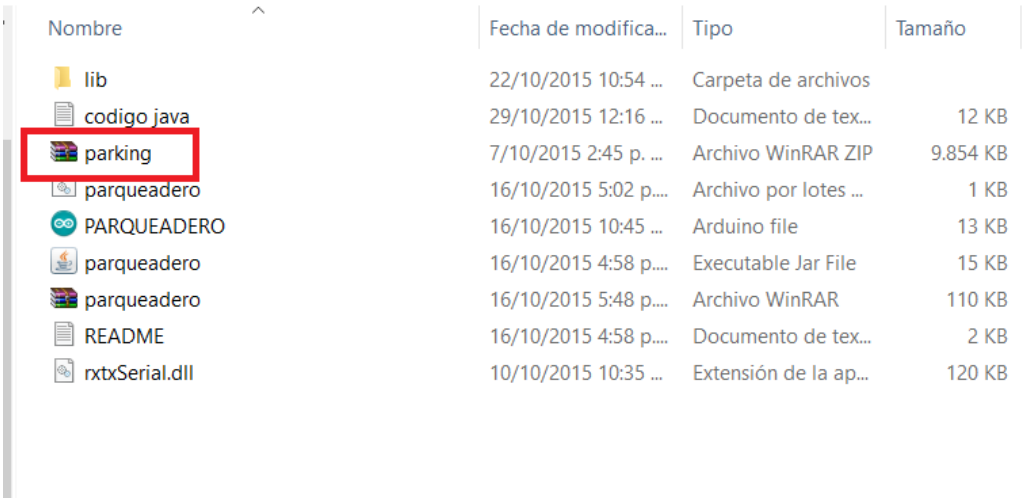
Anexo 5. Ingreso a la página web

Para acceder como administrador a la página web, se deben ingresar el usuario: sebastian y la contraseña: 12345.

Para acceder como cliente a la página web, se debe ingresar el usuario: cliente y la contraseña: cliente.

Anexo 6. Programación de la página web

Adjunto a este documento, se encuentra una carpeta llamada “PARQUEADERO”, en el momento de abrirla aparecerá lo siguiente:



Al ejecutar esa carpeta comprimida, se puede observar toda la programación realizada para obtener la página web.