ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO PARA EL CONTROL Y MONITOREO DE REGISTRO DE TIEMPOS DE TRANSPORTE URBANO MEDIANTE EL USO DE TECNOLOGÍA RFID, ARDUINO MEGA Y RASPBERRY PI 2

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES

WILLIAM ANDRÉS HERRERA LLORI

andresh2910@hotmail.com

PABLO RIGOBERTO VELOSO LLUMIGUSIN

pablorveloso123@hotmail.com

DIRECTORA: ING. MÓNICA VINUEZA RHOR

monivinueza@yahoo.com.ar

Quito, Agosto 2017

DECLARACIÓN

Nosotros, William Andrés Herrera Llori y Pablo Rigoberto Veloso Llumigusin declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

William Andrés Herrera Llori

Pablo Rigoberto Veloso Llumigusin

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por William Andrés Herrera Llori y Pablo Rigoberto Veloso Llumigusin, bajo mi supervisión.

ING. MÓNICA VINUEZA RHOR

DIRECTORA DE PROYECTO

AGRADECIMIENTO

En primer lugar gracias Dios padre todopoderoso por todas las bendiciones derramadas sobre mí, gracias a ello soy mejor persona.

A la persona que ha estado conmigo apoyándome, dándome su mano y acompañándome de manera incondicional en este largo camino de la vida, a ti mi esposa Alexandra un millón de gracias por ser la otra mitad de mi vida. Te amo.

A mi madre Liliana, no tengo más que decirle un gran Dios le pague por todo lo que me ha enseñado y me ha dado, la amo con todas mis fuerzas, es mi mayor ejemplo y por usted soy la persona que soy.

A mi querido hijo Alan, lo mejor que me ha pasado, te amo, tú eres mi orgullo y el mejor regalo que la vida pudo darme.

A mis hermanos queridos Juan, José, Ronald, Adrián, Carlos y Micaela; a mi padre José y a todas las personas que siempre supieron apoyarme con una palabra de aliento, muchísimas gracias por todo su apoyo.

Un agradecimiento especial a mi compañero de trabajo y amigo Roberto Toapanta quien supo ayudarme en el presente proyecto.

Y por último a ti Pablo, por ser mi gran amigo, acompañarme desde el inicio de mi carrera universitaria y culminar juntos este trabajo.

Andrés Herrera Llori

AGRADECIMIENTO

Agradezco a Dios por darme la oportunidad de vivir este momento muy especial en mi vida. Por llenarme de fuerza, guiarme y no dejarme caer en cada paso que doy.

A mis padres Arturo y Carmen por ser un gran ejemplo a seguir, por apoyarme en todo momento, por su sacrificio diario y entrega total para ayudarme a culminar con éxito esta etapa de mi vida. Por su paciencia y amor incondicional que me impulsa a seguir alcanzando nuevas metas.

A mis hermanos Consuelo, Angel y Margoth; cuñados y sobrinos por su apoyo moral, por sus consejos, por siempre estar pendientes y por las palabras de aliento que me impulsaron a nunca rendirme.

A la Ing. Mónica Vinueza por su colaboración y su acertada dirigencia para la culminación de este proyecto de titulación.

A mis amigos, profesores y especialmente a Andrés por brindarme su amistad sincera, conocimiento y ayuda durante mi vida estudiantil.

Pablo Veloso

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto tiene como finalidad la implementación de un prototipo para el control y monitoreo de registro de tiempos de transporte urbano, utilizando tecnología RFID. Para lo cual se utilizó varios dispositivos electrónicos, tales como el Módulo Arduino Mega 2560 y el Módulo Raspberry Pi 2B, los cuales permiten leer los datos de identificación de varias etiquetas RFID a través de un lector RFID.

Los datos obtenidos, se logran almacenar en una base de datos, para posteriormente ser presentados en una interfaz web con el fin de que sea accesible para las personas que lo requieran. Con el uso de estos dispositivos se logra obtener los tiempos reales en que las etiquetas pasan por el lugar donde se sitúa el lector.

Este proyecto expone los fundamentos teóricos, la descripción en forma general de las características y clasificación de los dispositivos utilizados en la construcción del prototipo; detalla las herramientas del software y hardware utilizados para la elaboración del prototipo, así como también para el desarrollo de la interfaz gráfica.

Presenta las pruebas de funcionamiento del prototipo, su presentación final en una maqueta, y por último se obtienen todas las conclusiones y recomendaciones presentadas durante el desarrollo del prototipo.

ABSTRACT

The purpose of this project is the implementation of a prototype for the control and monitoring of urban transport times, using RFID technology. Several electronic devices, such as the Arduino Mega 2560 Module and the Raspberry Pi 2B module, were used to read the identification data of several RFID tags through an RFID reader.

These data are stored in a database and later presented in a web interface in order to be accessible to those who require it. With the use of these devices is possible to obtain the actual times in which the labels pass through the place where the reader is placed.

This project exposes the theoretical foundations, the general description of the characteristics and classification of the devices used in the construction of the prototype. It details the tools of the software and hardware used in the elaboration of the prototype, as well as the development of the graphical interface.

It presents the tests of operation of the prototype and its final presentation in a scale model, and finally, the conclusions and recommendations of the prototype are presented.

1. INTRODUCCIÓN

En la actualidad, el control y registro de tiempos de circulación de las unidades de transporte urbano se lo realiza con un reloj por impresión de tarjeta, cuando un bus pasa por un sitio establecido como punto de control. Este proceso es ineficiente, inseguro y no refleja los tiempos reales que registran las unidades, ya que la información puede ser adulterada por las personas responsables de llevar a cabo esta labor.

Estas circunstancias por lo general se presentan cuando hay incumplimiento en el recorrido de las rutas establecidas, atrasos injustificados, pérdida de las tarjetas o tráfico en la ciudad que les imposibilita llegar a tiempo a los puntos de control. Además esta tarea es peligrosa ya que los “controladores de los buses”, quienes son los encargados de realizar este trabajo, pueden sufrir algún tipo de accidente que afecte su integridad.

El presente proyecto surgió de la necesidad de mejorar, supervisar y automatizar este proceso, utilizando dispositivos y elementos electrónicos que existen en el mercado actual y permiten resolver los problemas antes mencionados. Este proyecto comprende el diseño y construcción hasta el nivel de la presentación de un prototipo que refleja las ventajas y funcionalidades que posee el sistema desarrollado.

Se investigó una alternativa práctica y de bajo costo para una posible solución, siendo ésta la utilización de dispositivos RFID (Identificación por Radio Frecuencia), que permiten la lectura de etiquetas o tags de forma inalámbrica y sin necesidad de tener línea de vista.

En este prototipo, se utiliza un lector RFID MRFC522 y varios tags para el prototipo, conjuntamente con el módulo electrónico Arduino Mega 2560 que permite la lectura del código de las etiquetas. Además se empleó el módulo Raspberry PI modelo 2B con el objetivo de procesar la información adquirida desde el Arduino y almacenarla en una base de datos desarrollada en MySQL, la cual se muestra en una interfaz web que es accesible para los propietarios o las personas encargadas del control de las unidades de transporte, cuando ellos lo requieran.

1.1. Tecnología RFID

RFID (Identificación por radio frecuencia) es un tipo de tecnología que permite el reconocimiento de personas, vehículos, productos u objetos de forma inalámbrica y sin la necesidad de tener línea de vista. [1]

Un sistema que utiliza tecnología RFID presenta dos elementos principales:

- **Etiqueta o tag RFID:** es un pequeño transmisor que puede ser colocado en una persona o un objeto con el fin de identificarlo.
- **Lector RFID:** es el encargado de trasmitir energía al tag para poder recibir los datos que ésta posee y enviarlos a un sistema de procesamiento de información.

La principal característica de la comunicación RFID es la utilización de ondas electromagnéticas para la transmisión de la información que posee una etiqueta, la cual es recibida y procesada por un lector.

Las bandas de frecuencia que son utilizadas en la implementación de sistemas con tecnología RFID se visualizan en la siguiente tabla:

Tabla 1 Frecuencias de operación de los sistemas RFID. [2]

Frecuencia	***Rango de Operación***	***Alcance***	***Transferencia de Datos***	***Longitud de Onda***
LF	*30KHz a 300KHz*	*10 cm*	*Alta*	*10Km a 1Km*
MF	*300KHz a 3MHz*	*20 cm*	*Alta*	*1Km a 100m*
HF	*3MHz a 30MHz*	*1 metro*	*Alta*	*100m a 10m*
UHF	*300MHz a 3GHz*	*Hasta 3 metros*	*Rápida*	*1m a 1dm*
SHF	*3GHz a 30GHz*	*Hasta 10 metros*	*Súper rápida*	*1dm a 1cm*

Comunicación RFID

La comunicación por radiofrecuencia es un aspecto principal en la tecnología RFID, estas ondas electromagnéticas se propagan en forma de onda, por lo tanto pueden ser alteradas en frecuencia y amplitud. Existen dos formas de comunicación electromagnética [3]:

- **Campo Lejano:** basado en campos electromagnéticos utilizados en comunicaciones a larga distancia o para altas frecuencias; se usan normalmente con etiquetas UHF, como se aprecia en la figura 1.

  Emplea la ***propagación de ondas electromagnéticas*** para energizar a la etiqueta, que captura estas ondas propagadas por el lector, para ello tanto la antena del tag como la del lector deben ser dipolo. El tag recibe la energía como un voltaje alterno mediante los extremos del dipolo, la cual permite alimentar los circuitos integrados del tag.

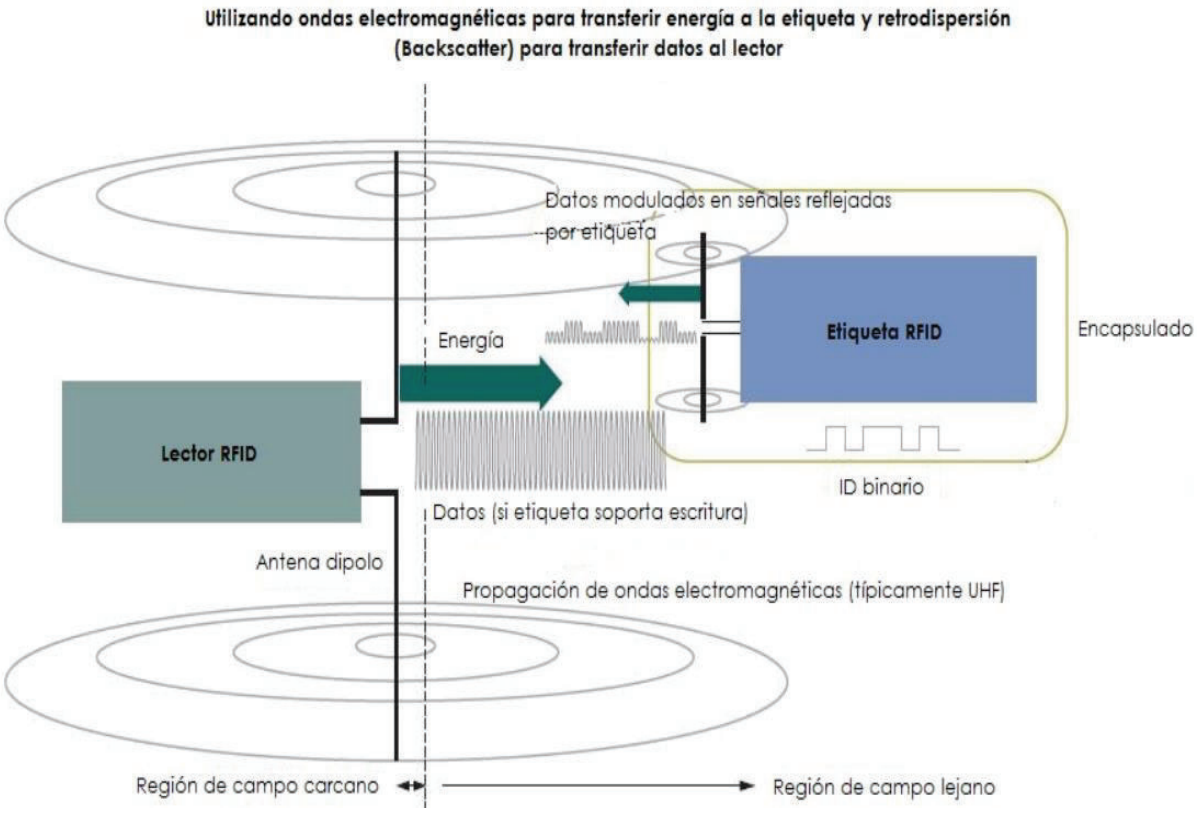


Figura 1 Comunicación entre tag y lector por campo lejano. [4]

- **Campo Cercano**: basado en campos electromagnéticos en comunicaciones de corta distancia o bajas frecuencias, es utilizada por tags pasivos HF, mismos que se utilizan en el proyecto. Estos campos son generados por una bobina que posee el lector, los cuales atraviesan la sección de la antena del tag, de este modo generan un voltaje por inducción el cual alimenta al microchip de la etiqueta permitiendo que envíe los datos contenidos en ella, como se muestra en la figura 2. A este proceso se lo denomina ***acoplamiento inductivo.***

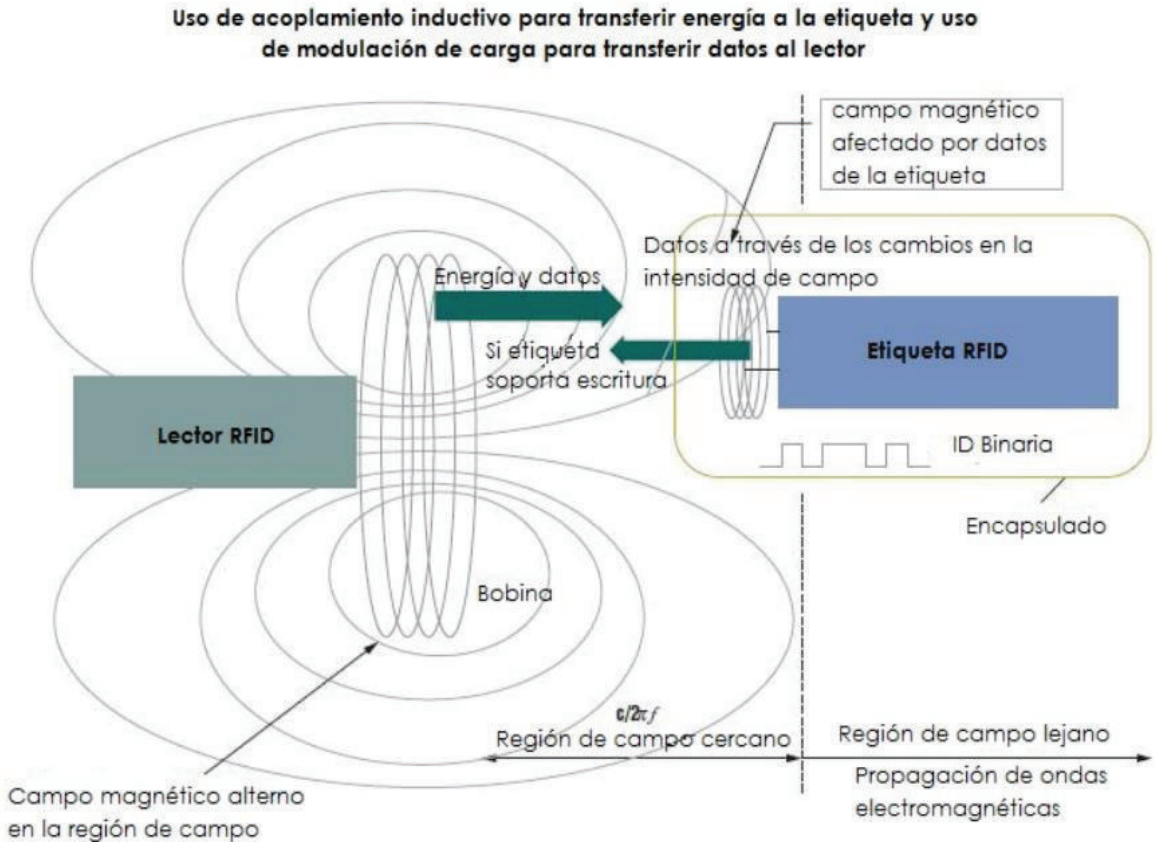


Figura 2 Comunicación entre tag y lector por campo cercano. [5]

Funcionamiento de un Sistema RFID.

Un sistema RFID presenta 3 pasos básicos para su funcionamiento: [6]

- El lector emite señales de radiofrecuencia a la etiqueta, las cuales son captadas por la antena que posee el tag.

- Estas ondas alimentan a un microchip dentro del tag, el cual envía la información registrada en su memoria de vuelta al lector.
- El lector recibe esta información del tag y la envía a un ordenador para su procesamiento.

En la siguiente figura se puede observar el funcionamiento básico de un sistema RFID.

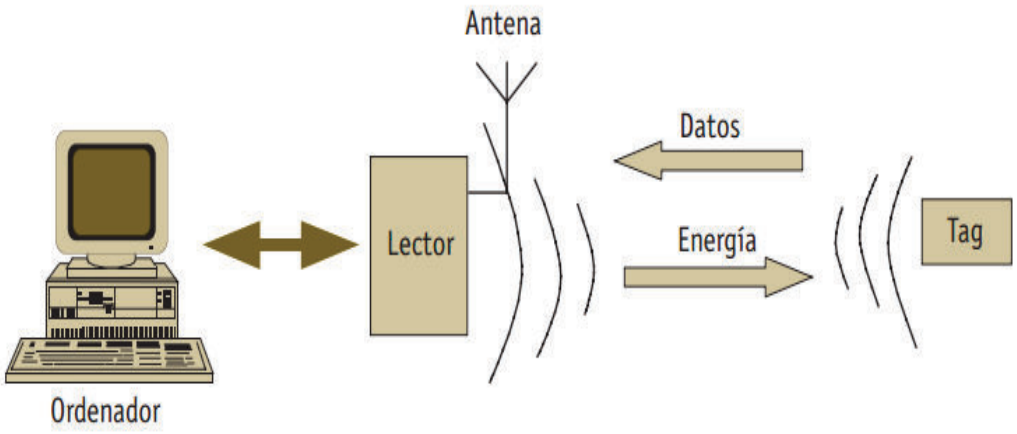


Figura 3 Funcionamiento de un sistema RFID. [7]

Para un óptimo funcionamiento de cualquier sistema RFID se debe tomar en cuenta los siguientes factores principales:

- El rango de lectura entre el lector y la etiqueta.
- La velocidad de transmisión de los datos de la etiqueta.
- El tamaño de los tags.
- El tipo de alimentación del sistema.
- El tipo de tag (si es activo o pasivo).
- El rango de operación.

Ventajas y Desventajas de la Tecnología RFID

La tecnología RFID posee ventajas y desventajas que son importantes al momento de utilizarla como método de autenticación:

Ventajas:

- Los tags no necesitan estar visibles en el exterior, ya que pueden ser detectados si están en el rango de lectura del lector.
- Elimina el escaneo y búsqueda manual del objeto o sujeto a identificar.
- Es una tecnología inalámbrica, que permite reducir costos y mejorar los procesos manuales.
- Son eficientes y tienen una vida útil de larga duración.
- Se logra obtener datos en tiempo real de los objetos que están siendo monitoreados o identificados.

Desventajas:

- Puede existir interferencia en la comunicación en ambientes donde hay presencia de líquidos, metales, entre otros.
- El rango de alcance entre lector y etiqueta es limitado y depende de muchos factores, como es el caso de la frecuencia.
- El costo de implementar este tipo de tecnología en monitoreo e identificación de objetos a largo alcance es relativamente elevado, aunque en la actualidad debido al avance tecnológico estos costos se están reduciendo cada vez más.

1.2. Etiquetas RFID

Las etiquetas o tags RFID son unos pequeños dispositivos fáciles de adherir a cualquier objeto, poseen en su interior una antena de radiofrecuencia y un microchip digital, que les permiten responder a las señales de un lector RFID, como se aprecia en la figura 4. [8]

La antena permite que el chip pueda enviar los datos que posee hasta el lector, el cual convierte las ondas de radio reflejadas por los tags en información digital hacia un ordenador para su procesamiento; para la transmisión de datos se necesita pequeñas cantidades de energía en el orden de los micro watts.

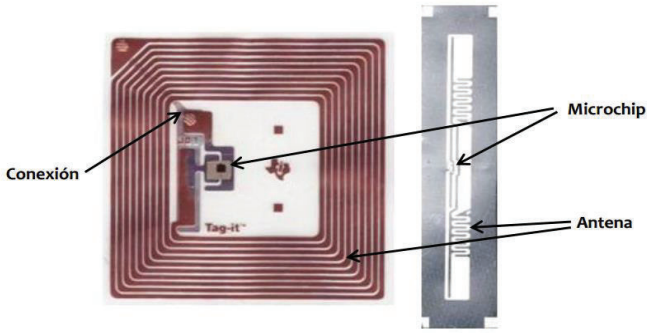


Figura 4 Esquema interno de una etiqueta RFID. [9]

Tipos de Etiquetas RFID

Existen dos tipos de etiquetas: [10]

a) **Por su capacidad de programación:** los tags RFID pueden ser:

- **Sólo lectura**: vienen programados de fábrica.
- **Una escritura y múltiples lecturas:** admiten reprogramarlos por una sola vez.
- **Lectura y escritura:** pueden ser reprogramados en varias ocasiones.

b) **Por su modo de alimentación:** se clasifican en:

- **Activos:** poseen una batería interna que permite que puedan alcanzar distancias de lectura muy grandes para la transmisión de los datos. Estas se utilizan para la localización de personas u objetos y en aplicaciones donde se incorporan otros dispositivos como sensores de velocidad, temperatura, etc., en las que la información se requiere en tiempo real, ya que todo el tiempo están enviando señales de su ubicación.

- **Pasivos:** no tienen una batería incluida y necesitan energía del lector para la transmisión de información. Tiene corto rango de alcance. Los tags pasivos trabajan con el siguiente rango de frecuencias:

Tabla 2 Alcance de las Etiquetas pasivas. [11]

FRECUENCIA	ALCANCE	VELOCIDAD
LF (125 a 134 KHz)	0 a 10 cm	Baja
HF (13.56 MHz)	0 a 60 mm	Alta
UHF (860-960 MHz)	10 cm a 10 m	Muy Alta

Tecnología NFC

La tecnología NFC (Near Field Communication) es una evolución de la tecnología RFID, la cual trabaja a una frecuencia de 13,56 MHz; permite la conectividad inalámbrica segura entre dos dispositivos, con el correspondiente intercambio de datos.

Tags Pasivos NFC

Los tags pasivos NFC (comunicación de campo cercano), son un subtipo de la tecnología RFID, que se diferencia únicamente en su rango de alcance, el cual es de 0 a 60 mm. Estos son ampliamente utilizados en sistemas de control de acceso, control de productos, inventarios, etc. Se construyen en forma de tarjeta o llavero. Ver características técnicas en el Anexo B del informe. En la figura 5 se puede observar el esquema de un tag NFC.

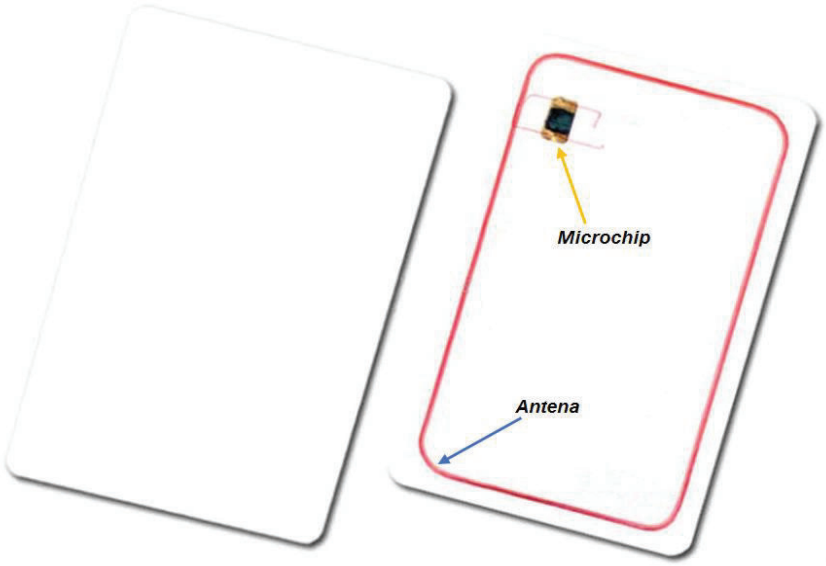


Figura 5 Tag NFC. [13]

1.3. Lector RFID RC522

El lector RFID-RC522 es un dispositivo electrónico mediante el cual el campo electromagnético que genera induce a una comunicación con los tags RFID NFC, cuando están cerca del él, sin necesidad de contacto [14]. Este dispositivo y la identificación de sus pines se muestran en la figura 6.

El lector RFID está constituido por una antena, un transceptor y un decodificador; estos emiten ondas de RF a una frecuencia establecida de 13.56MHz, con la finalidad de obtener la información de identificación única almacenada en el microchip de la etiqueta.

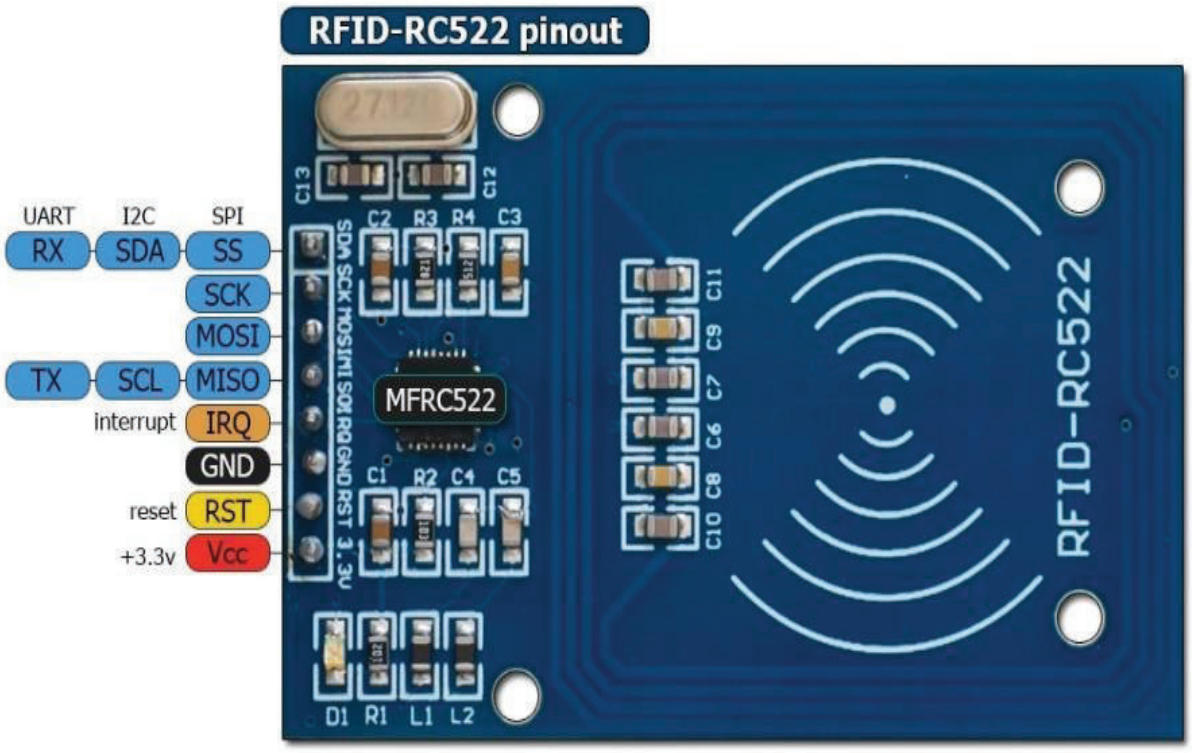


Figura 6 Partes de un lector RFID-RC522. [15]

Su funcionamiento radica en el envío de señales periódicas para determinar si existe alguna respuesta por parte de un dispositivo que se encuentre dentro de su rango de lectura.

Cuando recibe alguna respuesta de una etiqueta RFID, el lector toma la información que contiene la etiqueta, la codifica y envía a los dispositivos de control o procesamiento. Ver características técnicas en el Anexo B del informe.

El módulo lector RFID en este proyecto es utilizado para identificar y leer la información almacenada en los tags, el cual se puede ubicar a una distancia máxima de 6cm para que las tarjetas sean identificadas sin ningún problema.

1.4. Arduino MEGA 2560

El Arduino Mega, el cual se puede observar en la figura 7, es un módulo electrónico que permite implementar proyectos de hardware y software; está compuesto por varios circuitos impresos, los cuales integran un microcontrolador, adicional posee su propio editor de código fuente, en donde se puede desarrollar todos los programas para su utilización con cualquier módulo o sensor electrónico adicional compatible con este. Además, todas las herramientas de software necesarias para la elaboración de los programas son libres y gratuitos. [17]

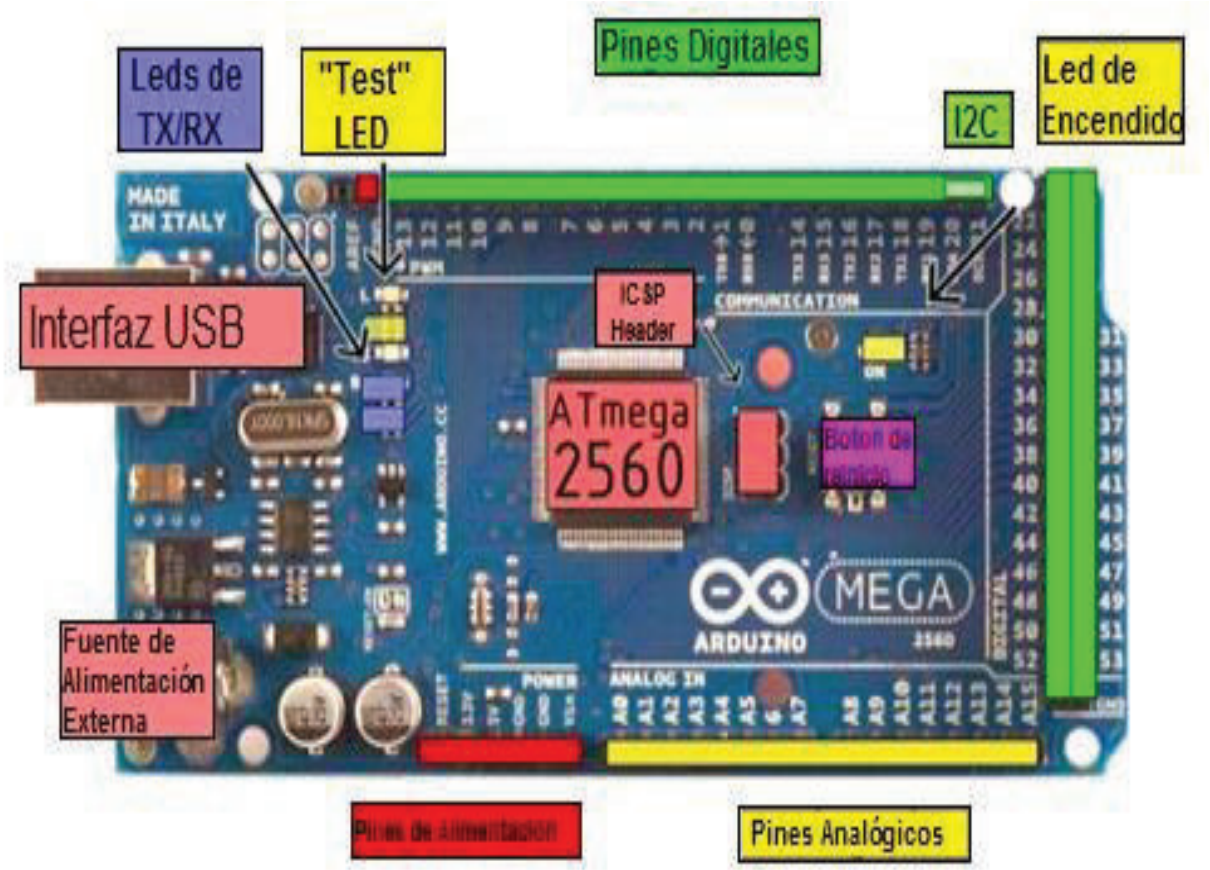


Figura 7 Partes de un Arduino Mega 2560. [18]

Está compuesto por un microcontrolador ATMEGA 2560, el cual tiene 54 pines digitales de entrada/salida, 16 entradas analógicas, un cristal de 16MHz, conexión USB, un puerto de alimentación y un botón de *reset* que permite reiniciar el Arduino a su estado de fábrica. Ver características técnicas en el Anexo B del informe.

El módulo Arduino Mega 2560 cumple la función de procesar los datos obtenidos por el lector RFID-RC522, para luego enviarlos al módulo Raspberry PI2.

El Arduino Mega presenta varias ventajas en cuanto a su utilización: [20]

- Usar un módulo Arduino simplifica el proceso de trabajar con microcontroladores.
- Las placas Arduino son más económicas en comparación con otro tipo de plataformas de microcontroladores.
- El software de Arduino es multiplataforma, funciona en los sistemas operativos Windows, Macintosh OSX y Linux.
- El ambiente en el cual se programa el Arduino es fácil de utilizar para cualquier usuario.
- El software Arduino es de licencia libre, es decir de código abierto, puede ampliarse a través de librerías de C++.

1.5. Raspberry PI 2B

El Raspberry Pi 2B, el cual se muestra en la figura 8 y 9, es un mini computador de bajo costo y tamaño pequeño (similar a un documento de identificación), que presenta todas las características de un PC, convirtiéndolo en la opción perfecta para estimular la programación y automatizar muchas tareas. [21]

El Raspberry Pi usa un sistema operativo basado en Linux denominado RASPBIAN JESSIE, que contiene un ambiente gráfico amigable con los usuarios, siendo de esta manera fácil su utilización y además presenta todas las aplicaciones básicas que posee un computador. Ver características técnicas en el Anexo B del informe.

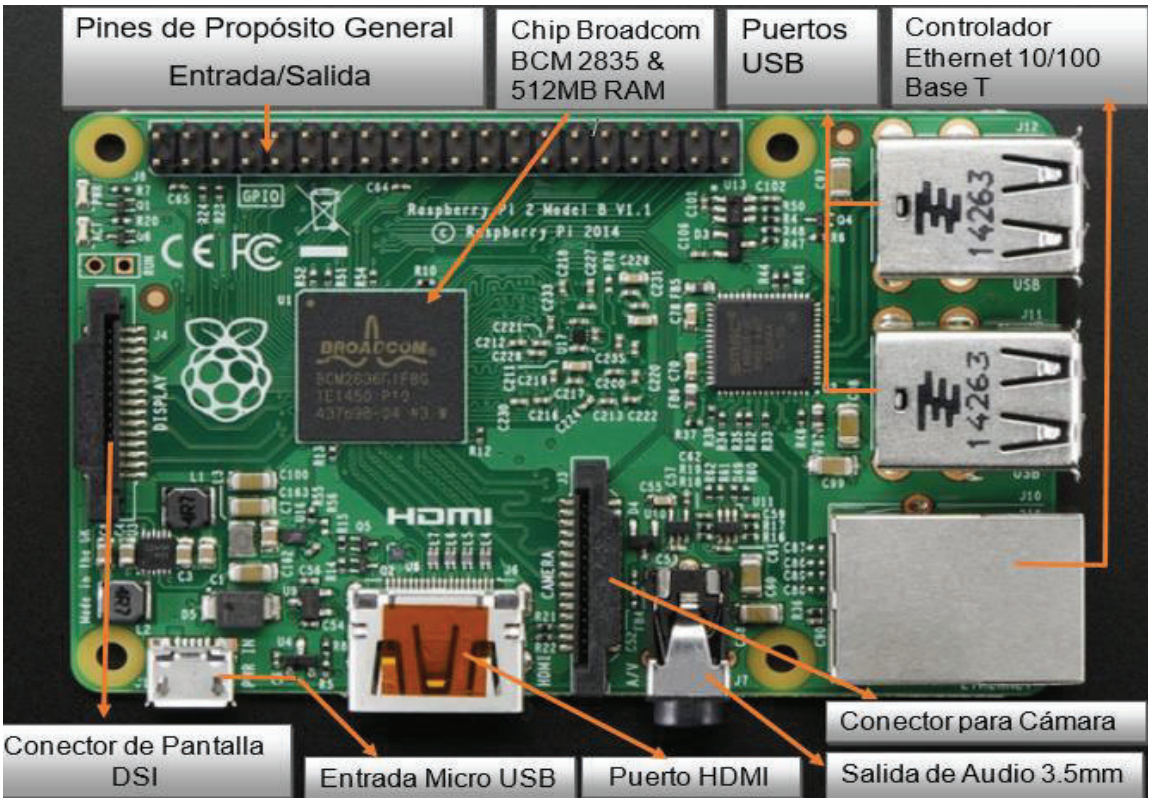


Figura 8 Raspberry Pi, Módulo B, Vista superior. [22]

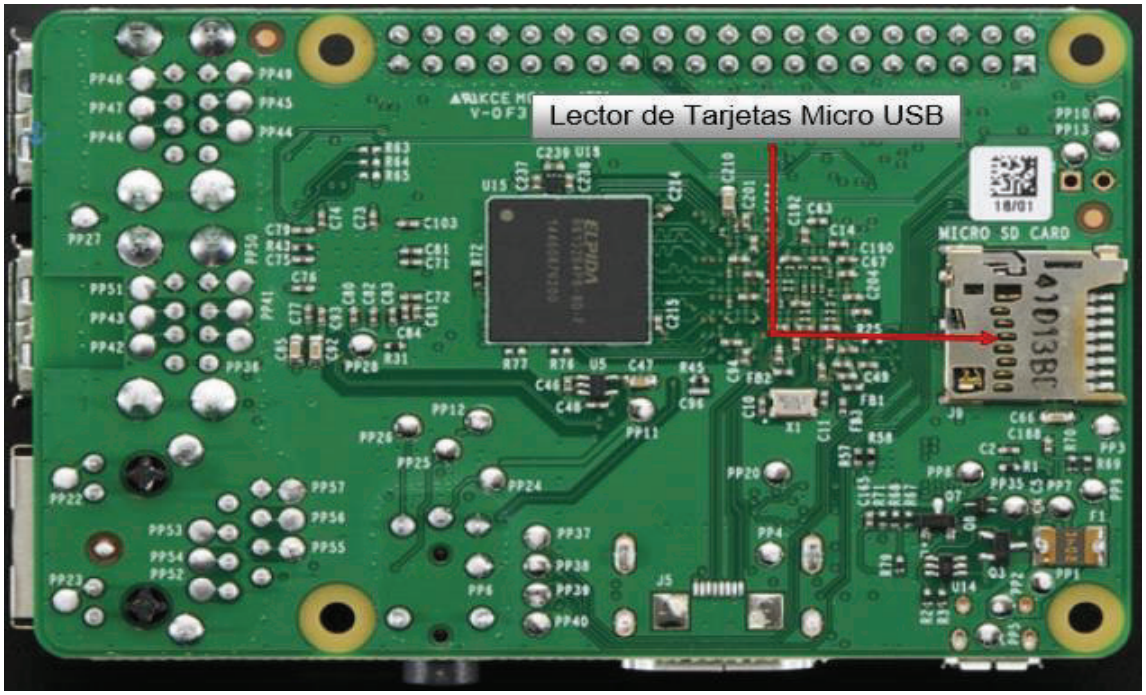


Figura 9 Raspberry Pi, Módulo B, Vista posterior. [23]

Este modelo presenta los siguientes componentes:

- Dos puertos USB 2.0.
- 40 pines de propósito general de entrada/salida.
- Un controlador Ethernet 10/100 Base T.
- Un procesador Broadcom BCM2836 con 512MB de RAM.
- Un lector de tarjetas Micro SD.
- Una entrada micro USB
- Un puerto HDMI.
- Un conector para cámara, una salida de video RCA y de audio 3.5mm

El Raspberry presenta varias ventajas para su utilización, entre las principales están:

- Bajo consumo de energía (0.5W en reposo)
- Fácil de usar y aprender.
- Bajo costo.
- Compatibilidad con Arduino.

Además, el Raspberry presenta una distribución de pines denominados de propósito general de entrada/salida (GPIO - *General Purpose Input/Output)*, los cuales se detallan en el Anexo B del informe.

El Raspberry PI 2B es el centro de almacenamiento y procesamiento del proyecto porque en él se almacena la base de datos; esta base se podrá visualizar desde un navegador web con el objetivo de que esta información esté disponible para los usuarios en cualquier momento.

1.6. Servidor Apache

Apache es un servidor web de software libre que permite manejar varios sitios web, para el prototipo es ideal ya que es compatible con el sistema operativo del Raspberry. [25]

Apache se encarga de escuchar las peticiones que se envía mediante cualquier navegador web, las gestiona y envía nuevamente con el resultado correspondiente.

Esta aplicación presenta varias características principales:

- Es multiplataforma (Windows, Linux, Mac)

- Utiliza el protocolo HTTP para presentar archivos o datos de una página web.
- Tiene soporte de seguridad SSL y TLS.
- Puede soportar diferentes lenguajes: Perl, PHP, Python.

Además, presenta varias ventajas para el manejo de bases de datos: [26]

- Permite visualizar y comprobar la presentación final a los usuarios de una página web.
- El servidor Apache es completamente gratuito y puede ser descargado por cualquier persona en el mundo, permitiendo desarrollar la base de datos sin pagar por ningún tipo de licencia o permiso.

1.7. PHP

PHP es una herramienta que permite procesar el código que se ejecuta cuando el servidor recibe una petición de una página web. Resuelve lo que necesita ser mostrado en la interfaz web y a continuación, envía la página al navegador. [27]

Mediante PHP se puede mostrar no solo contenido estático sino también se puede crear páginas dinámicas, ideal para aplicaciones donde se necesita visualizar datos que llegan en tiempo real. Ver características en el Anexo B del informe.

Ventajas de PHP. [28]

- Permite realizar programación estructurada.
- Compatibilidad con otros programas.
- Tiene herramientas visuales para su configuración y administración

1.8. MYSQL

MySQL es un programa que permite la gestión de una base de datos relacional y multihilo, que permite usar varios usuarios; además es software libre [29]. Está desarrollado en su mayor parte en ANSI C. Ver características en el Anexo B del informe.

Ventajas de MySQL.

- Velocidad rápida para ejecutar instrucciones.
- Presenta costos bajos en cuanto a los requerimientos para la creación de base de datos. Su licencia comercial cuesta 200 USD, pero existe una versión gratuita para uso no comercial.
- Su instalación y configuración inicial es fácil
- Se puede implementar en varios sistemas operativos.
- Conectividad y seguridad.

Para la instalación de MySQL, el computador debe cumplir algunos requerimientos:

- Debe tener 512 MB de memoria RAM.
- Poseer 1GB de espacio en el disco duro.
- Tener instalado uno de los siguientes sistemas operativos: Windows, Linux y Unix.
- La arquitectura del sistema puede ser de 32 o 64 bits.
- Debe soportar el protocolo de red TCP/IP.

1.9. PhpMyAdmin

PhpMyAdmin es un programa desarrollado en PHP, el cual permite administrar la base de datos creada en MySQL a través de cualquier navegador de Internet. Con ella se puede modificar la base datos, mostrar los resultados almacenados, gestionar usuarios de MySQL, etc. [30]

Ventajas de PhpMyAdmin:

- Posee una interfaz web intuitiva.
- Exporta datos en diferentes formatos
- Permite la creación de gráficos en PDF del diseño de la base de datos

Para utilizar PhpMyAdmin es necesario cumplir con algunos requisitos: [32]

- Servidor Web: puede ser Apache, nginx, IIS, etc., en el que se instalan los archivos de PhpMyAdmin.
- PHP: Es necesario PHP 5.5.0 o posterior.

- Base de Datos: MySQL 5.5 o posterior; MaríaDB 5.5 o posterior
- Navegador Web: Es necesario un navegador con cookies y JavaScript activado.

1.10. Python

Python es un lenguaje de programación potente y fácil de usa, el cual mediante el desarrollo de códigos, permite interconectar un proyecto con Raspberry Pi para presentarlo al mundo real. [33]

La sintaxis de Python es muy limpia, con énfasis en la legibilidad y utiliza palabras clave en inglés estándar.

Ventajas de Python:

- Al poseer muchas librerías, estas nos ayudan a realizar múltiples tareas sin necesidad de programarlas desde cero.
- Facilidad para crear programas.
- Lenguaje multiplataforma.
- Con la utilización del pseudocódigo, se concentra más en la solución de un problema que en su sintaxis.

1.11. SQL Workbench

Es un programa utilizado para el diseño de base de datos, el cual agrupa el desarrollo de software, la administración, creación y mantenimiento para el sistema de base de datos en MySQL. [35]

Ventajas:

- Brinda libertad a los usuarios para crear y modificar bases de datos.
- El usuario puede usar, copiar, modificar y redistribuir en cualquier computador la base de datos diseñada en el programa.
- Existe ahorro en una adquisición de licencia para su utilización.
- Es muy eficiente para desarrollar bases de datos.

1.12. Comunicación Serie

La comunicación serie es una interfaz de comunicación de información digital que permite establecer transferencia de datos entre varios dispositivos.

El puerto serie es la principal forma de comunicación entre un módulo Arduino y un Raspberry. Este puerto permite el envío de datos mediante una secuencia de bits, para lo cual se necesita mínimo de dos conectores; uno de trasmisión (TX) y uno de recepción (RX).

El protocolo de comunicación utilizado en el proyecto es el denominado SPI (*Serial Peripherical Interface*), que es un protocolo síncrono full dúplex que permite recibir y transmitir información, con el objetivo de que dos dispositivos pueden comunicarse entre sí al mismo tiempo. En este protocolo se define un maestro, el cual está encargado de transmitir información a sus esclavos; y el esclavo, el cual es el dispositivo que recibe y envía la información al dispositivo maestro, tal como se puede apreciar en la figura 10. [36]

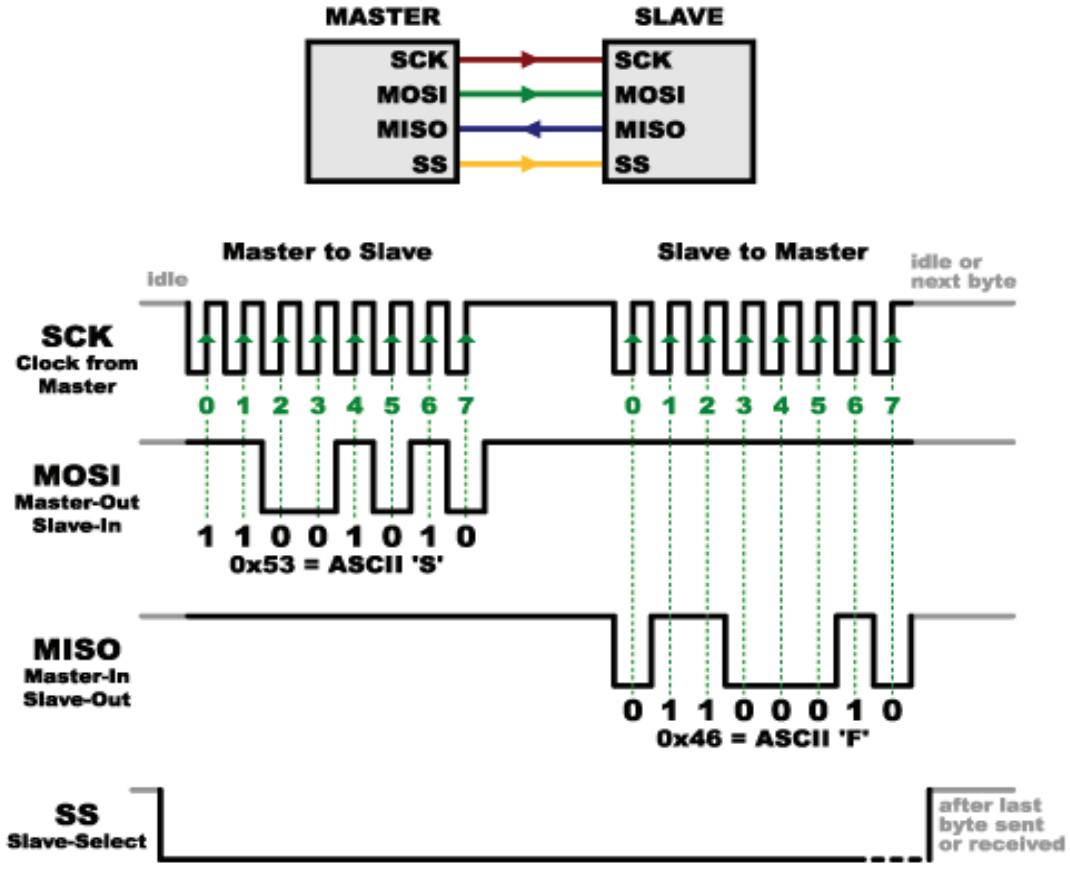


Figura 10 Funcionamiento del protocolo SPI. [37]

Existen cuatro líneas lógicas que realizan este proceso, las cuales están definidas en el Arduino por los siguientes pines:

- MOSI (Pin50) (*Master Out Slave In):* Esta línea es empleada para llevar los datos que vienen del maestro hacia el esclavo.
- MISO (Pin 51) (*Master In Slave Out*): Esta línea es empleada para llevar los datos que vienen del esclavo hacia el maestro.
- CLK (Pin 52) (*Clock*): Esta línea proviene del maestro y es la encargada de enviar la señal de reloj que permite sincronizar los dispositivos.
- SS (Pin 53) (*Slave Select*): Esta línea es la encargada de escoger y habilitar a un esclavo.

1.13. Acceso Remoto

El módulo Raspberry es un microordenador que no posee dispositivos de entrada y salida, por lo que, para su utilización es necesario emplear una aplicación que permita acceder a la interfaz del Raspberry; para ello se utiliza la herramienta TeamViewer.

TeamViewer es un software informático que permite la conexión de forma remota a otro equipo, permitiendo compartir y controlar escritorios, reuniones en línea, transferencia de archivos entre ordenadores, etc. [38]

En la actualidad se ha desarrollado TeamViewer Beta para Raspberry Pi, el cual ofrece acceso no supervisado a dispositivos Raspberry Pi con el objetivo de llevar a cabo su mantenimiento, control y administración. [39]

2. METODOLOGÍA

El presente proyecto se desarrolló empleando dos tipos de investigación: exploratorio y aplicado. La investigación se ajusta a un estudio exploratorio, ya que el tema del prototipo no ha sido abordado en el campo del transporte público en el país, si bien existen soluciones similares a otro tipo de necesidades, no se ha realizado la automatización del proceso que abarcó este proyecto.

A su vez, la investigación es aplicada ya que mediante los conocimientos adquiridos en clase se busca diseñar e implementar una solución con el objetivo de controlar y monitorear de forma automatizada el recorrido que los buses de transporte público realizan en sus diferentes rutas.

En la investigación realizada, se empleó la técnica de recolección de información. Se realizó un estudio literario de varios autores y documentos digitales disponibles en Internet con el fin de adquirir los conocimientos necesarios para entender el funcionamiento de los dispositivos electrónicos y aplicaciones informáticas.

Para la implementación del prototipo fue necesario fijar qué tipo de tecnología inalámbrica para la lectura de datos de los tags era la más adecuada.

Se realizó un estudio de los dispositivos electrónicos a utilizarse en el proyecto tales como: módulo Arduino, módulo lector RFID, tags, módulo Raspberry Pi, módulo Wifi 802.11n, de los cuales se eligió los que de mejor manera satisfacen los requerimientos del prototipo.

El prototipo, será alimentado por un cargador conectado a un tomacorriente; éste se conecta en el Raspberry que a su vez es el encargado de alimentar a todos los dispositivos.

Luego, se procedió al desarrollo de un código para la lectura del identificador único de los tags, a través del lector RFID que está conectado a la placa Arduino, mediante el programa de desarrollo de códigos de Arduino denominado IDE.

Para establecer la comunicación serial entre el módulo Arduino Mega 2560 con el módulo Raspberry PI 2B, se implementó un código en Python que permite leer los datos que son enviados a través del Arduino y almacenarlos en la base de datos.

La conexión física entre los módulos se realiza a través de un cable serial USB.

Después de leer los tags se procedió a la implementación de una base de datos en la que se incluye conceptos y aplicaciones informáticas como: Servidor Apache, PHP, MYSQL, PhpMyAdmin, Phyton, SQL Workbench, las cuales permiten crear el programa necesario para visualizar la información requerida por los administradores de los buses en una interfaz web y desde el Internet.

Finalmente, se realizó las pruebas para comprobar el funcionamiento del prototipo con el objetivo de corregir los errores que se presenten y solventarlos de la mejor manera; para una mejor visualización del funcionamiento del prototipo, se lo presenta en una maqueta.

2.1. Requerimientos para el diseño del prototipo

El cumplimiento del tiempo de las rutas que el transporte público realiza en muchas ocasiones no es el real, ya que se ven expuestos a diferentes factores los cuales les afectan directamente, por tal motivo se vio la necesidad de automatizar, monitorear y mejorar este proceso en el presente proyecto.

Este prototipo está diseñado para controlar y monitorear los tiempos exactos en que los buses de transporte público pasan por un sitio de control, además que los datos que se registren podrán visualizarse a través de una interfaz web en el momento que el dueño o administrador del bus así lo requiera, tomando en cuenta que tanto el Raspberry PI 2B y el computador o celular desde el cual se requiera acceder deberán estar conectados a Internet para que el proceso se realice.

El proyecto está diseñado en dos partes:

La primera parte consiste en el diseño para la lectura de los Tags, la cual fue desarrollada para cumplir con los siguientes requerimientos:

- Identificar el código único de las tarjetas RFID NFC, el cual se asocia a cada unidad de transporte.
- Utilizar un módulo Arduino que permita recibir y enviar los datos obtenidos por el lector hacia el Raspberry.

La segunda etapa consiste en la recepción, procesamiento y almacenamiento de la información, el cual cumple los siguientes requerimientos:

- Recepción de datos por parte del Raspberry PI.
- Implementar una base de datos que recoja y almacene la información de forma ordenada.
- Desarrollar una interfaz web que permita la presentación de los datos en un entorno amigable con el usuario.

3. RESULTADOS Y DISCUSIÓN

3.1. Diseño del Prototipo

De acuerdo a los requerimientos que se mencionan en la metodología, se diseñó el proyecto en dos partes: la parte de lectura de los datos y la parte de procesamiento y almacenamiento.

Todo el prototipo es alimentado con un voltaje de +5V para su correcto funcionamiento, mismo que es proporcionado por el módulo Raspberry, ya que a este se le conecta un cargador de 120VAC/5VDC.

La parte de lectura de datos contiene un módulo Arduino MEGA 2560 el cual se conecta con el lector RFID; el lector funciona a +3.3V, voltaje que es regulado automáticamente por el Arduino. El lector y el Arduino son los encargados de identificar, recibir y enviar los datos proporcionados por las etiquetas RFID NFC hacia el módulo Raspberry.

La parte de procesamiento y almacenamiento contiene un módulo Raspberry PI 2B, el cual necesita una tarjeta MicroSD con el objetivo de almacenar su sistema operativo, llamado *Raspbian Jessie Pixel*, y la base de datos, la cantidad de elementos seleccionados se la puede observar en la Tabla 3.

Además se incluye un módulo Wifi 802.11n que permite que el prototipo se conecte a Internet y de esta manera poder acceder a información de manera remota a través de la aplicación TeamViewer. En la figura 11 se visualiza el diagrama total del prototipo.

Lectura de Datos

Para la elaboración de la primera etapa del prototipo, la cual permite la lectura de los tags, se tomó en cuenta la utilización de los siguientes dispositivos:

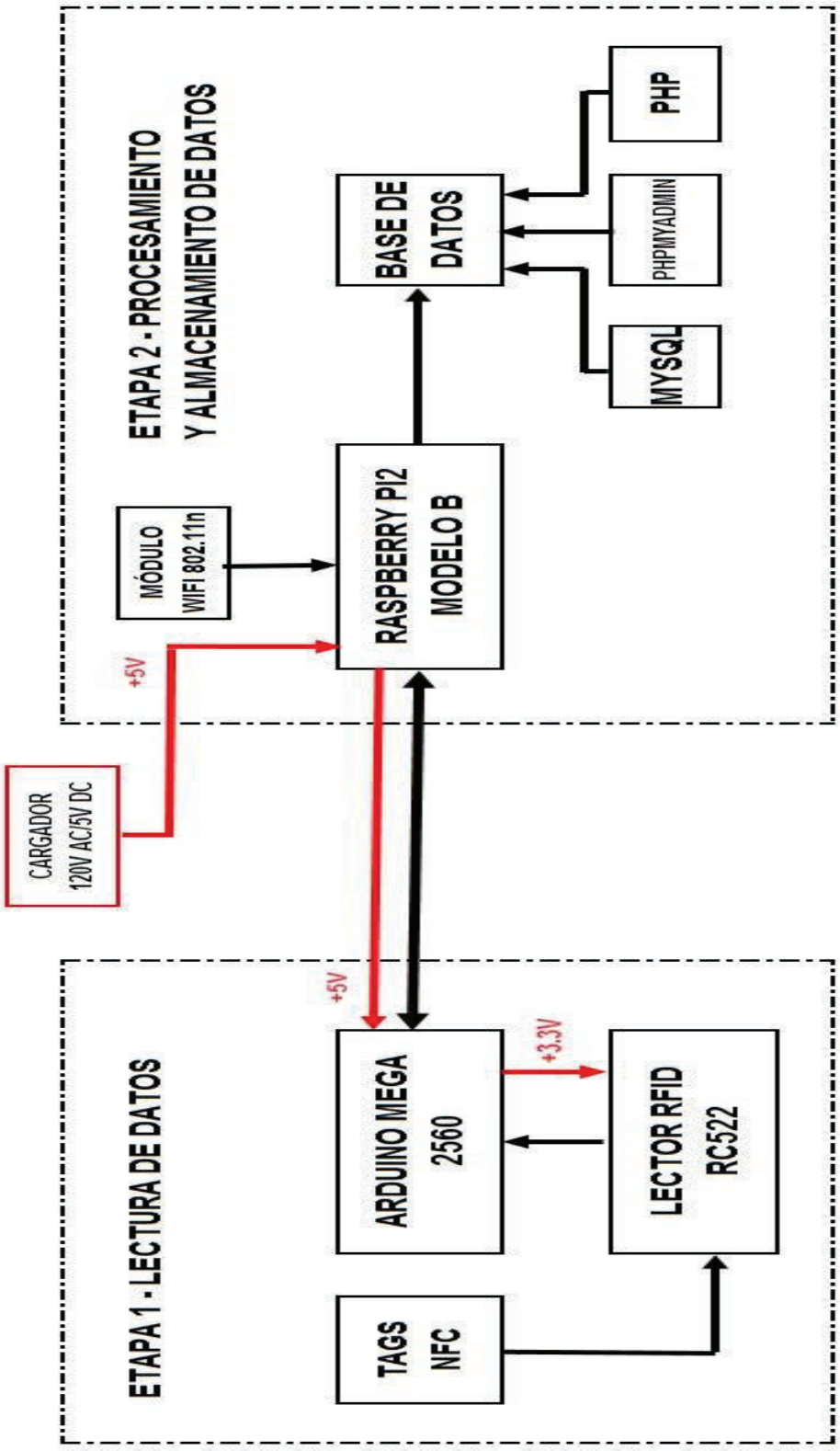


Figura 11 Diagrama de bloques del prototipo.

Tabla 3 Elementos seleccionados para la etapa 1 del prototipo

CANTIDAD	DISPOSITIVO
1	Módulo Arduino Mega 2560
1	Lector RFID MFRC522
3	Tags RFID tipo Tarjeta
Varios	Cables para conexión de dispositivos

Conexión lector RFID-RC522 con Arduino Mega 2650

El lector RFID-RC522 permite la lectura de los datos de los tags, para posteriormente enviarlos al Módulo Arduino Mega 2650. La conexión se la realiza mediante la distribución de pines que se detalla en la siguiente tabla y se la visualiza en la figura 12.

Tabla 4 Distribución de pines para interconexión del Arduino con el lector. [43]

Lector RC522	Arduino Mega 2560
SDA (SS)	SDA (pin 53)
SCK (CLK)	SCK (pin 52)
MOSI	MOSI (pin 51)
MISO	MISO (pin 50)
GND (tierra)	GND
RST(reset)	RST (pin 5)
Alimentación (3.3V)	3.3V

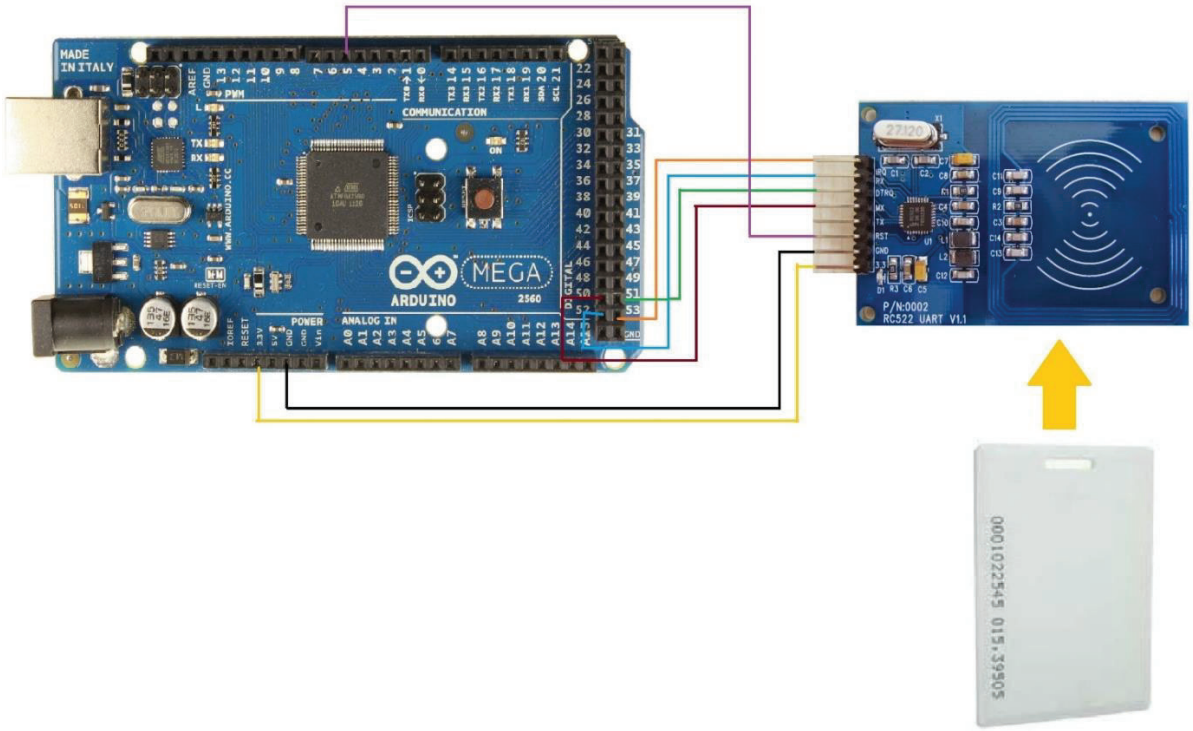


Figura 12 Conexión física del Arduino con el lector RFID.

3.2. Procesamiento y Almacenamiento de Datos

En el desarrollo de la segunda etapa del proyecto, la cual permite que los datos recolectados por el Arduino sean enviados al Raspberry para procesarlos y almacenarlos en la base de datos, se utilizan los siguientes elementos:

Tabla 5 Elementos seleccionados para la etapa 2 del prototipo

CANTIDAD	DISPOSITIVO
1	Módulo Raspberry PI 2B
1	Cargador 120V AC/5V DC
1	Cable serial USB
1	Tarjeta de Memoria externa de 16GB para el sistema operativo del Raspberry
1	Módulo Wifi 802.11n

Es necesario, primero colocar una tarjeta de memoria externa en el Raspberry, ya que allí se almacenará el sistema operativo y la base de datos. Por lo general se recomienda instalar una tarjeta con capacidad superior a 8GB.

Conexión Módulo Arduino Mega 2650 con el Raspberry PI 2B

Para establecer la comunicación entre el Raspberry y el Arduino, físicamente estos se comunican mediante un cable USB, estos se comunican serialmente entre sí para el envío y recepción de los datos, como se puede apreciar en la figura 13.

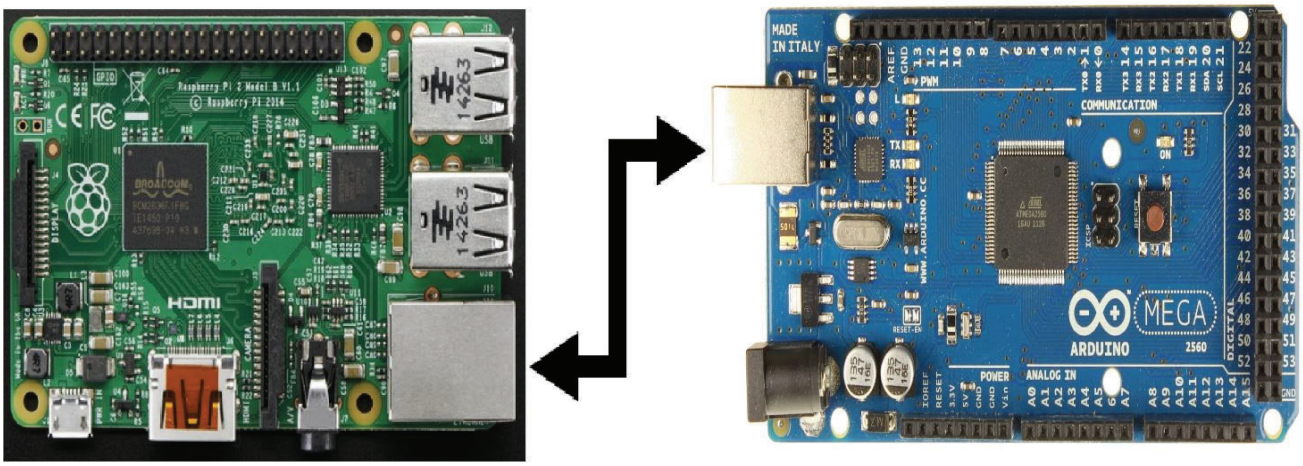


Figura 13 Conexión del Arduino Mega con el Raspberry PI 2B.

3.3. Desarrollo de los Programas del Prototipo

Una vez establecidos los componentes del prototipo, se procede con el desarrollo de los códigos en el Arduino y en el Raspberry, con el objetivo de presentar los datos en una interfaz web.

Lectura de datos de los tags en Arduino.

En primer lugar, mediante la utilización de programación en lenguaje C/C++, y con el fin de comprobar la comunicación entre el lector y el Arduino, se emplea el software de código abierto para Arduino (IDE – Entorno de Desarrollo Integrado).

En la figura 14 se muestra el diagrama de flujo del programa para la lectura de los tags en Arduino.

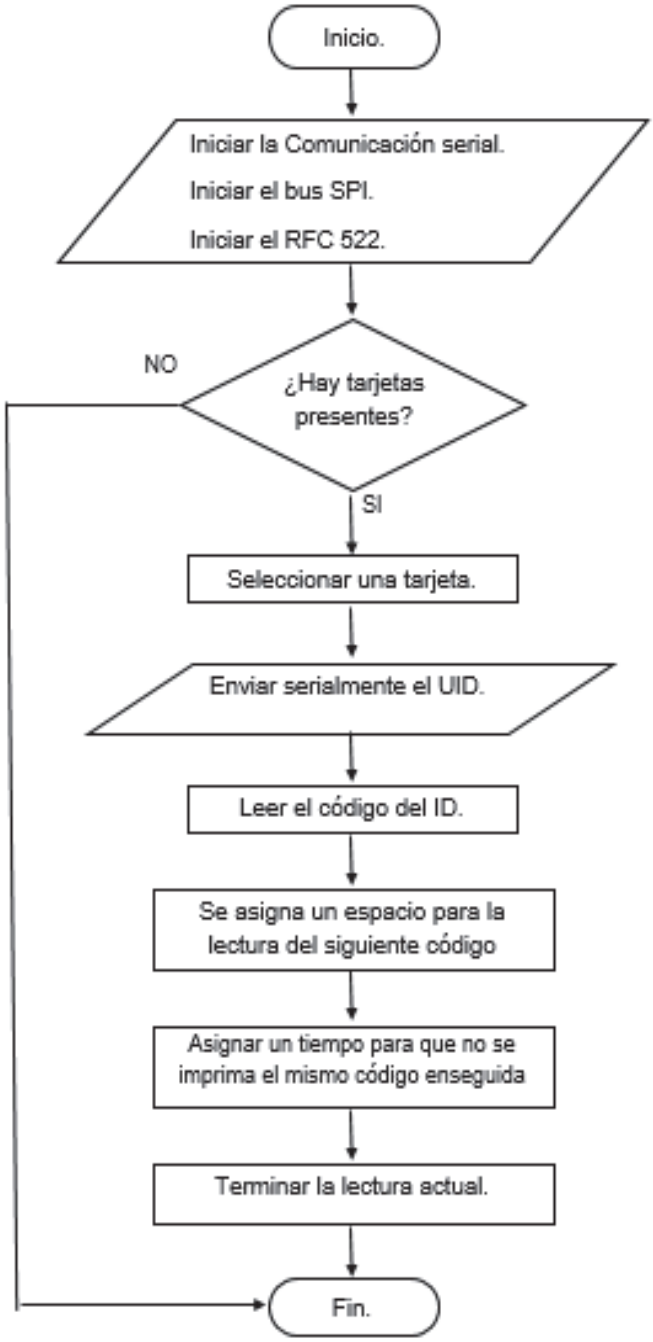


Figura 14 Diagrama de Flujo para la lectura de los tags en Arduino.

Se emplean las librerías que proporcionan los fabricantes de Arduino (las cuales se detallan a continuación en el código de lectura de los tags), de este modo se obtienen los identificadores únicos de cada tag, los cuales serán asignados a cada una de las unidades de transporte.

Código para la lectura de los datos de los tags.

```
#include <SPI.h>                                    //Incluye la librería para usar el protocolo SPI
#include <MFRC522.h>                                //Incluye la librería para el lector RC522
#define RST_PIN  5                                  //Define el Pin 5 para el reseteo del RC522
#define SS_PIN  53                          //Define el Pin 53 para activar el SS (SDA) del RC522
MFRC522 mfrc522 (SS_PIN, RST_PIN);                  //Se crea el objeto para el RC522


void setup() {
  Serial.begin (9600);                                          //Inicia la comunicación serial
  SPI.begin ();                                                 //Inicia el Bus SPI
  mfrc522.PCD_Init ();                                          // Inicia el MFRC522
//Serial.println("Numero de tags RFID");
}
void loop() {

  if ( mfrc522.PICC_IsNewCardPresent())             // Revisa si hay nuevas tarjetas presentes
            {

              if ( mfrc522.PICC_ReadCardSerial())               //Selecciona una tarjeta
                {

//Serial.print(("Numero HEX del Tag:"));                        // Envía serialmente el UID
for (byte i = 0; i < mfrc522.uid.size; i++)   //mfrc522.uid.size---obtiene tamaño del código de ID
                      {
Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");      //mfrc522.uid.uidByte---lee el código de ID
Serial.print(mfrc522.uid.uidByte[i], HEX);              }
Serial.println();                                       //espacio para la lectura del siguiente código
delay(1000);                  //tiempo para que no pueda imprimirse el mismo número enseguida

                    mfrc522.PICC_HaltA ();                  // Termina la lectura de la tarjeta actual
                }
            }
        }
```

Lectura de la información del Arduino en el Raspberry.

Luego de haber verificado que el Arduino pueda leer el identificador de las etiquetas, es necesario implementar un código que permita leer los datos que llegan del Arduino y almacenarlos en la base de datos; para ello se emplea Python. En la figura 15 se visualiza el diagrama de flujo para la leer la información del Arduino en el Raspberry.

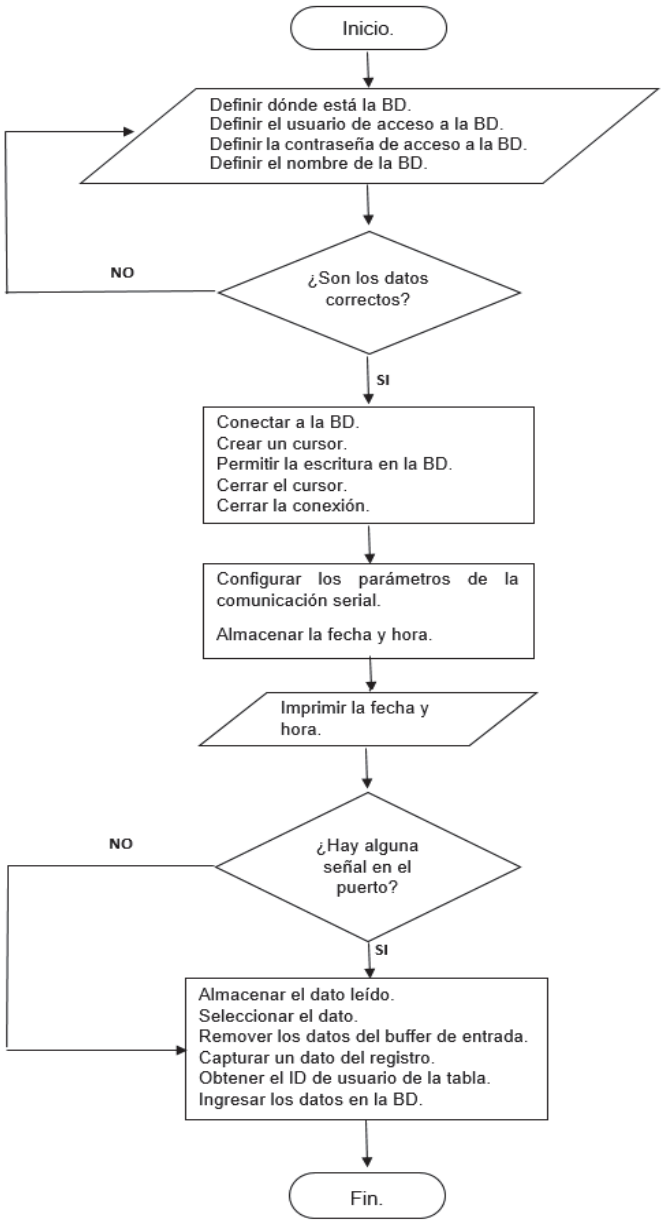


Figura 15 Diagrama de Flujo para leer la información del Arduino en el Raspberry.

Código para la lectura y almacenamiento de los datos en el Raspberry utilizando Python.

```
#Importa los módulos que se utilizan para conexión con la base de datos

import serial
import time
import commands
import os
import MySQLdb
import socket

# Código para escanear automáticamente el puerto USB al que se conecta el Arduino en el Raspberry PI2, de esta manera se asegura que el Arduino se pueda conectar a cualquier puerto del Raspberry para el envío de la información

port_ttyacm = commands.getoutput("ls -a /dev/ttyACM*")
print "Arduino port: " + port_ttyacm
print "port initialized correctly"
#print "port not initialized correctly"
if port_ttyacm == "/dev/ttyACM0":
    print "Port ttyACM0"
    os.system("sudo chmod 777 /dev/ttyACM0")
    os.system("sudo chown pi:pi /dev/ttyACM0")
else:
    if port_ttyacm == "/dev/ttyACM1":
        print "Port ttyACM1"
        os.system("sudo chmod 777 /dev/ttyACM1")
        os.system("sudo chown pi:pi /dev/ttyACM0")
    else:
        if port_ttyacm == "/dev/ttyACM2":
            print "Port ttyACM2"


# Código que permite la conexión con la base de datos:
```

```
DB_HOST = 'localhost'                    #define el servidor o host donde está la base de datos
DB_USER = 'traspduino_user'              #define el usuario de acceso a la base de datos
DB_PASS = 'admin12345'                   # define la contraseña de acceso a la base de datos
DB_NAME = 'traspduino'                   #define el nombre de la base de datos

def run_query(query=''):
    datos_conexion = [DB_HOST, DB_USER, DB_PASS, DB_NAME]
    conn = MySQLdb.connect(*datos_conexion)          # Conecta a la base de datos
    cursor = conn.cursor()                           # Crea un cursor
    cursor.execute (query)

if query.upper().startswith('SELECT'):
        data = cursor.fetchall()                     # Trae los resultados de un select
    else:
        conn.commit()                                # Permite la escritura de datos en la base
        data = None
    cursor.close()                                   # Cierra el cursor
    conn.close()                                     # Cierra la conexión
    return data


#Código para almacenamiento de datos en la base de datos

arduino = serial.Serial(port_ttyacm, baudrate = 9600, timeout = 3.0)   # configura los
parámetros de la comunicación serial

hora=time.strftime('%Y-%m-%d %H:%M:%S')          # almacena la fecha y hora
print hora                                       # imprime la fecha y hora

while True:

    if arduino.inWaiting() > 0:                  # espera a que haya algún dato en el puerto
        arduinoData = arduino.readline()         # almacena el dato leído
        aData = arduinoData[1:12]                # selecciona el dato
        print aData
        arduino.flushInput()                     # remueve los datos del buffer de entrada
        # Seleccionar solo registros coincidentes


        #idunidades = "E5 7A 1D 00"              # esta variable puede ser cargada desde el php
        idunidades = aData                       # esta variable puede ser cargada desde el php
        query = "SELECT * FROM unidades WHERE idunidades = '%s'" % idunidades
```

```
result = run_query(query)
print result
for registro in result:                                  # Captura un dato del registro
    unidades_socios_idsocios = registro[4]      # Obtiene el idsuarios de la tabla usuarios
    print unidades_socios_idsocios


query = "INSERT INTO registro_tiempos (fecha_registro, hora_registro,
lugar_control_idlugar, lugar_control_rutas_idrutas, unidades_idunidades,
unidades_socios_idsocios) VALUES (CURRENT_DATE(), NOW(), 1, 2,
'%s','%s')" %(idunidades, unidades_socios_idsocios)
run_query(query)                                        #Ingresa los datos en la base
```

Creación de la base de datos

Luego de establecer la comunicación entre el Raspberry y el Arduino, es necesario implementar la base de datos dentro del Raspberry, donde se almacena la información de los registros de cada tag.

En primer lugar se configura el Raspberry con una dirección IP estática cualquiera ya que éste actuará como el servidor web; para acceder a él se utiliza un cable Ethernet el cual se conecta a cualquier ordenador, permitiendo visualizar la interfaz gráfica de Raspberry.

En el prototipo se utilizó una dirección IP estática cualquiera, como se puede apreciar en la figura 16, que es la 192.168.1.50. También se puede ingresar al Raspberry mediante un terminal SSH.

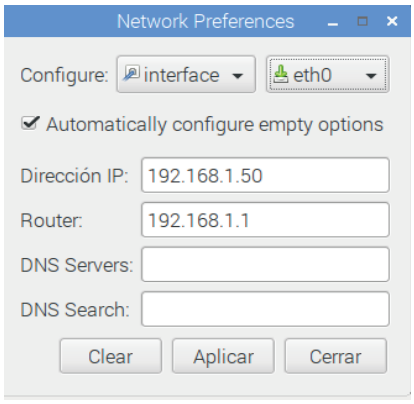


Figura 16 Configuración de la dirección IP estática del Raspberry.

Luego de ello, se instalan varias herramientas en el Raspberry para el desarrollo de la base y la interfaz web:

a) Servidor Apache
b) PHP
c) PhpMyAdmin
d) MySQL

Instalación del Servidor Apache

El servidor Apache será el encargado de escuchar las peticiones que se envíen mediante cualquier navegador web de la base de datos que se implemente.

Para descargar Apache, se abre una terminal de consola en el Raspberry y se ejecuta el siguiente comando:

sudo apt-get install apache2 –y

Una vez instalado Apache, se puede observar la página web por defecto cuando se navega en la dirección http://localhost/ (en el Raspberry), o accediendo a la dirección IP estática del Raspberry desde otro ordenador que esté conectado a la misma red.

Instalación de PHP

Una vez instalado el servidor Apache en el Raspberry, se procede a instalar PHP utilizando el comando:

sudo apt-get install php5 libapache2-mod-php5 -y

Mediante PHP se puede mostrar contenido estático y se puede crear páginas dinámicas con la información de los tags, ideal para el prototipo donde se necesita que se pueda visualizar los datos que llegan desde el Arduino en tiempo real.

Instalación de MySQL

El siguiente paso es instalar MYSQL para almacenar y administrar en una base de datos la información enviada por los tags. Para instalar MYSQL se introduce el siguiente comando:

sudo apt-get install mysql-server

Además se ejecuta el comando *sudo apt-get install php5-mysql* que se encarga de actualizar las librerías PHP encargadas de lidiar con los datos de MySQL.

Instalación de PhpMyAdmin

PhpMyAdmin permitirá administrar la base de datos creada en MySQL a través de cualquier navegador de Internet. Con ella se puede modificar la base datos, mostrar los resultados almacenados, gestionar usuarios de MySQL, etc., de forma que se pueda crear la base desde una interfaz más amigable con el usuario.

Para su instalación se debe ejecutar el comando en el terminal del Raspberry:

sudo apt-get install phpmyadmin

Para acceder a PhpMyAdmin desde el navegador, se debe configurar el servidor Apache mediante el siguiente comando:

sudo nano /etc/apache2/apache2.conf

En el fichero de configuración se digita la línea *Include /etc/phpmyadmin/apache.conf ;se* guarda los cambios y se reinicia el Raspberry, con lo que se completa la instalación.

Desde un navegador web cualquiera se puede visualizar la interfaz web de PhpMyAdmin la dirección IP estática del Raspberry, seguido de PhpMyAdmin: http//192.168.1.50/PhpMyAdmin. Al tener una interfaz web, se puede realizar consultas desde una computadora con acceso a Internet.

Diseño de la base de datos con MySQL Workbench

Para el manejo de gran cantidad de información que será recolectada de cada bus de servicio público, al momento de cruzar por el punto de control, es necesario crear una base de datos capaz de almacenar cada uno de los registros y estar relacionado cada registro.

Por lo cual, se debe tomar en cuenta factores como: tamaño de la información, velocidad de acceso, facilidad de extraer la información requerida, entre otros.

A través de MySQL Workbench, se facilita la administración de base de datos en una interfaz amigable para el usuario donde se puede realizar el diseño y la estructura de la misma.

La base de datos que se denomina *transdb*, está compuesta por 7 tablas relacionadas de acuerdo a las reglas de negocio [40], como se muestra en la figura 17.

Las tablas que están asociadas con el registro de las unidades de transportes son: tabla unidades, socios, rutas, registro_tiempos, lugar_control, las cuales están relacionadas entre sí.

Mientras que la tabla login_attempts, members es asociada al registro de los usuarios que ingresan a la aplicación web para tener una página segura.

Las relaciones entre las tablas son las siguientes:

- Relación entre la tabla unidades y la tabla registro_unidades es de uno a muchos.
- Relación entre la tabla unidades y la tabla socios es de uno a muchos.
- Relación entre la tabla rutas y la tabla lugar_control es de uno a muchos.
- Relación entre la tabla lugar_control y la tabla registro_tiempos es de uno a muchos.

Una vez establecidas las relaciones que permitirán conectar la base de datos, el programa compila la información y genera el código que permitirá crear la base de datos con los requerimientos establecidos; este código se inserta dentro de PhpMyAdmin y automáticamente se generan todas las tablas y variables necesarias de la base de datos.

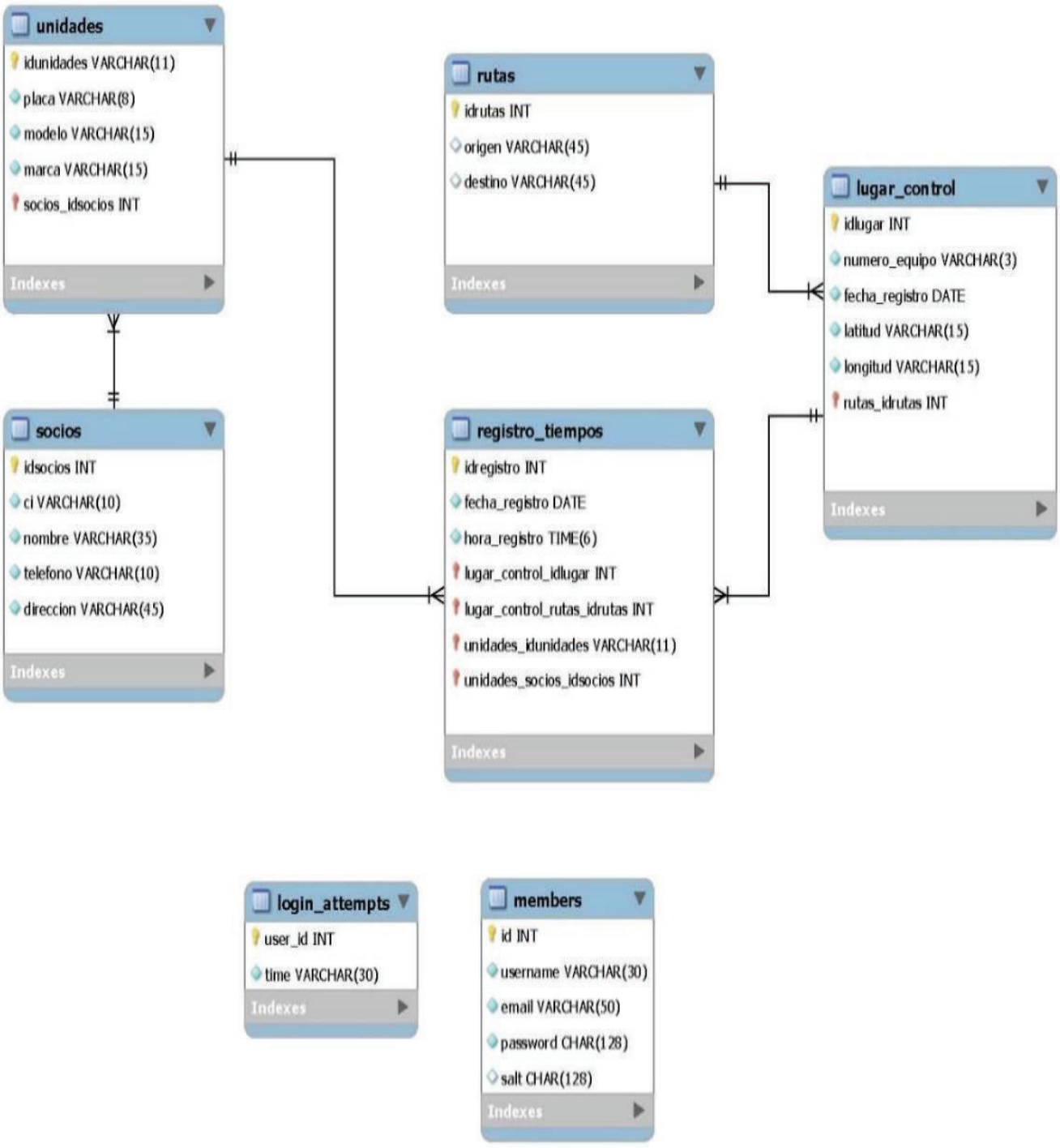


Figura 17 Tablas de relaciones de la base de datos en MySQL.

3.4. Implementación de la Interfaz Web.

Una vez terminada la base de datos en MySQL, se debe crear los scripts para obtener y modificar los datos en una interfaz web, utilizando HTML y PHP. Para ello, es necesario crear archivos con la extensión *.php* para las diferentes páginas del prototipo, las cuales se enlazan con la base de datos con el fin de obtener los datos en una página web, como se puede observar en la figura 18.

Estos se crean en el directorio */var/www/html* del Raspberry, los cuales son:

- *index.php*: Página del inicio de sesión.
- *register.php*: Página para el registro de un usuario nuevo.
- *error.php:* Página cuando se produce error de inicio de sesión.
- *register_success.php:* Página para el registro de un usuario nuevo
- *transportation_owners.php:* Página para la visualización de la información de los socios.
- *unit_registration.php:* Página para ver el registro de los tiempos de las unidades.
- *protected_page.php:* Página para ver la información de las unidades.

Figura 18 Captura de pantalla de la interfaz web final disponible para el usuario final

En cada uno de los archivos se introduce un código que permite enlazar la base de datos con la interfaz web, mostrando al usuario la información que desea visualizar. En el prototipo se utilizó una plantilla PHP prediseñada[1], para que cada una de éstas se pueda observar en un entorno amigable con el usuario. Los códigos de la base de datos se pueden observar en el Anexo D.

3.5. Pruebas de Funcionamiento

En este apartado se realizan las pruebas de funcionamiento con el propósito de verificar los resultados finales del proyecto.

En primer lugar se realizaron pruebas para obtener la lectura del identificador de los tags, como se puede apreciar en la figura 16, dando como resultado un bucle infinito de la lectura de un solo tag sin poder detenerla. Se verificó todas las líneas del código y se encontró que el error fue que faltaba la sentencia que permita terminar la lectura del tag.

Observación de los datos de los tags en Arduino

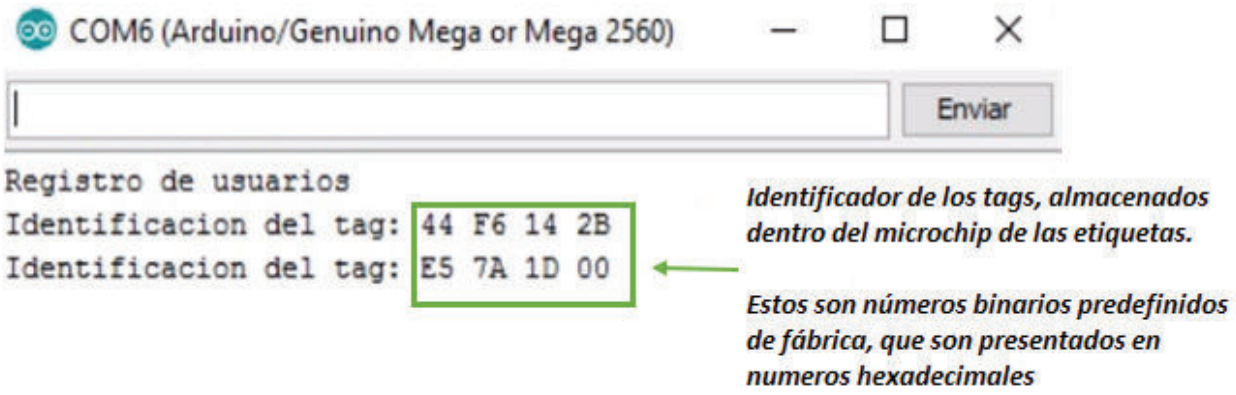


Figura 19 Lectura de los identificadores de los tags en el monitor serie del IDE de Arduino.

Se tuvo que ingresar varias sentencias en el código de la lectura de los tags en Arduino para solucionar el problema; en primer lugar la sentencia *SerialPrint (),* para dar un espacio para la

[1] Diseñada por Roberto Toapanta y Ángel Toapanta, modificada por Andrés Herrera y Pablo Veloso.

lectura del siguiente identificador del tag, luego se ingresó la sentencia *delay (1000)* que permite dar el tiempo necesario para que no se imprima el mismo número de manera seguida, por último se ingresó la sentencia *mfrc522.PICC_HaltA();* que permite terminar la lectura de la tarjeta utilizada en ese momento.

Finalizado este proceso, se compiló de nuevo el programa y se pudo observar que se obtenía la lectura de cada uno de los tags sin ningún tipo de error.

Datos enviados desde Arduino al Raspberry

Para presentar los datos se utilizó, en primer lugar, un módulo *Shield Ethernet* para Arduino, con el objetivo de importar los datos de los registros directamente en una extensión .CSV de Excel, pero no se obtuvieron resultados satisfactorios ya que la programación del *Shield Ethernet* resultó bastante compleja, por lo que se optó por utilizar un Raspberry.

Luego se intentó enviar los datos en una hoja de datos Excel directamente al Raspberry, pero no se pudo obtener resultados de ningún tipo de información de consulta o antecedentes de un trabajo similar en Internet, por lo que se optó en la utilización de una base de datos para el almacenamiento de la información.

Para ello, en primer lugar se creó una base de datos en MySQL directamente; es decir, se le dio un nombre, usuario y contraseña de ingreso, que es la parte inicial de la base. Una vez terminado este paso, fue necesario implementar un código que permita leer los datos enviados del Arduino al Raspberry y almacenarlos en la base.

En el código desarrollado en Python existieron varios problemas, cada vez que se conectaba el Arduino a un puerto diferente del Raspberry, los datos no llegaban, ya que si se utilizaba un puerto diferente al cual se le asignaba la entrada de datos, éste no funcionaba.

Dicho inconveniente se solucionó implementando líneas de código que permiten identificar automáticamente el puerto de conexión para la recepción de los datos, de esta manera se pudo visualizar en el monitor serie de Python que los datos llegaban al Raspberry por

cualquiera de los puertos al cual se le haya conectado el Arduino para su almacenamiento en la base, como se puede apreciar en la figura 20.

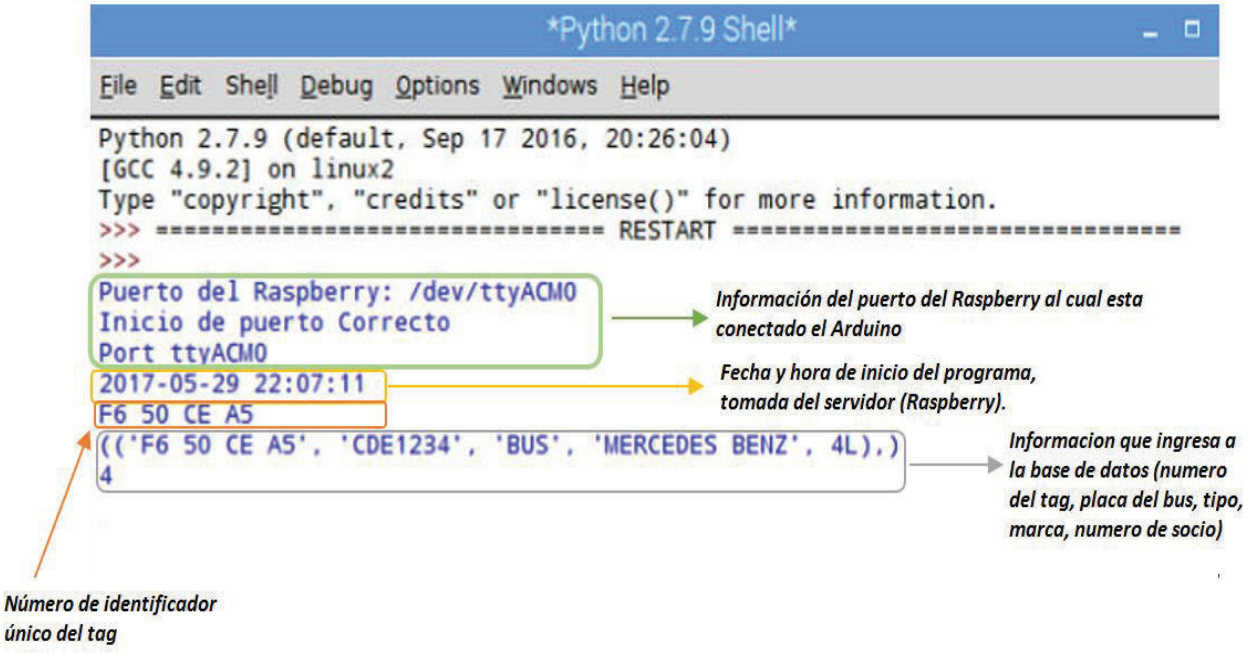


Figura 20 Lectura de la interfaz de Python para la lectura de los tags en el Raspberry.

Obtención de la hora y fecha de los tags en tiempo real

En primera instancia, se tuvo previsto insertar el módulo de reloj Tiny RC DS1307 en el proyecto, este es un módulo de reloj compatible con Arduino y Raspberry que permite almacenar la hora del servidor en su memoria interna mediante un script en el Raspberry con el objetivo de respaldar a cualquiera de estos dispositivos la fecha y hora en el momento que se desactualicen.

El problema suscitado en el prototipo fue que a medida que se desarrollaba la base de datos, los repositorios y librerías propias del Raspberry se actualizaban constantemente, provocando que el módulo de reloj no sea compatible con dichas actualizaciones, que son necesarias para el desarrollo de la base de datos.

Por lo que, la solución que se utilizó para poder enviar el tiempo real del momento que pasa el tag por el lector, fue tomar en consideración la hora que proporciona el servidor, de este

modo se mantienen actualizados los registros de las unidades con la fecha y hora real del servidor, misma que es proporcionada por la red.

Pruebas de la base de datos

Para desarrollar la base de datos se presentaron muchos inconvenientes, debido a que no se disponía de los conocimientos necesarios para poder implementarla. Este inconveniente se solucionó investigando en Internet, participando en foros y video tutoriales sobre cómo desarrollar la base de datos en un servidor web, utilizando las herramientas descritas en el proyecto. Una vez que se tuvo una idea clara de cómo implementar y qué herramientas se necesitan para la base de datos, el problema que se tuvo fue el relacionar cada una de las tablas que componen la base.

Para ello, se recurrió al programa *SQL Workbench*, donde se establecieron “reglas de negocio”, estas reglas se consideran como cualquier necesidad, requerimiento, o actividad especial que se comprueba al momento de intentar grabar información, borrar, actualizar o consultar la ya existente; las cuales se imponen por los usuarios o los administradores de la base de datos. [41]. De este modo se pudo relacionar todos los datos de los socios unidades de transporte, etc., creados en la base con cada una de las etiquetas RFID.

El otro inconveniente que se tuvo fue el tipo de datos, ya que estos deben tener el mismo formato tanto en la entrada de la información, mediante en script de Python, como en las tablas de relación de *SQL Workbench* y la base de datos. Todos los datos se definieron en formato *int*, pero solo se permitía el ingreso de números enteros; este inconveniente se solucionó optando por asignar en todos los códigos tipos de datos en formato *varchart*, de este modo se aseguró que se pueda ingresar cualquier dato (numérico, alfabético o alfanumérico) con una longitud de hasta 255 caracteres.

Visualización de los datos en la interfaz web

Para poder visualizar la información en una interfaz web, fue necesario instalar un servidor web que contenga las bases de datos y la interfaz en PHP. Se escogió la implementación de

un servidor Apache, debido a las ventajas que tiene en el sistema operativo del Raspberry, como se especifica en un capítulo anterior.

Surgieron inconvenientes en la instalación del mismo, ya que las actualizaciones del programa provocaron la incompatibilidad del mismo con el Raspberry; dicho problema se solucionó realizando una actualización total a los repositorios tanto del servidor Apache como del Raspberry.

Otro problema que se presentó fue relacionar todos los archivos y plantillas PHP con las tablas de la base de datos, ya que no se disponía del conocimiento necesario para realizar este proceso. Luego de realizar varias investigaciones y recurrir a charlas con profesionales sobre el tema, el inconveniente se solucionó integrando a cada uno de los archivos un código que realice la petición de la información a la base de datos de MySQL cada vez que se genere un nuevo dato, para presentarlo al mismo tiempo en la interfaz web.

Conexión a la base de datos vía Internet

Una vez terminada la base de datos y la aplicación web, ésta debe estar disponible en el Internet para que cualquier persona pueda acceder a ella; aquí se presentaron problemas ya que la única solución viable para poder visualizar la interfaz era configurar al Raspberry con una dirección IP pública, la cual es un poco difícil de obtener ya que se debe contratar una con un proveedor de Internet, representando costos adicionales a la inversión en el prototipo.

La solución a este problema fue la utilización de una aplicación que permita acceder al Raspberry remotamente desde cualquier parte sin la necesidad de que los equipos se encuentren en la misma red o de la IP pública.

Es por ello que se utilizó la aplicación TeamViewer en su versión para Raspberry, la cual fue lanzada en marzo del presente año; dicha aplicación nos permite acceder al dispositivo con el único requerimiento de que el Raspberry y el dispositivo con el cual se va acceder a él (teléfonos, tabletas, PC’s, etc.) se encuentren conectados a cualquier red de Internet.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Se verificó que no existe en la ciudad un sistema que realice el control de tiempo de las unidades de transporte en tiempo real cuando éstas pasan por un punto de monitoreo, ya que este procedimiento se lo realiza de forma manual.

- El prototipo es alimentado por un cargador que convierte la energía de 120VAC en 5VDC, por lo que el consumo de electricidad es muy bajo; además el Raspberry es el encargado de proporcionar energía a los demás dispositivos.

- Se pudo comprobar que el lector RFID MFCR522 permite obtener los identificadores de los tags, pero a una distancia de 60cm, ya que este dispositivo solo es utilizado con Arduino para la realización de proyectos a pequeña escala.

- El Arduino Mega 2560 es una alternativa económica y fácil de utilizar, ya que permite la conexión con el Raspberry de forma rápida y su programación no es complicada. Además, permite integrar módulos adicionales, tal como el lector RFID, para la lectura de cualquier tipo de datos.

- Para la lectura de las etiquetas, se diseñó un código utilizando el programa IDE de Arduino, mediante el cual se obtuvo los datos del tag con el objetivo de almacenarlos en la memoria del Arduino para su posterior envío al Raspberry.

- Para la conexión entre el Arduino y el Raspberry, se utilizan los puertos serie de cada uno de ellos, de esta manera la información llega al Raspberry para ser almacenada en una base de datos.

- El Raspberry PI 2B constituye una gran opción para la implementación de la base de datos, ya que es un minicomputador de bajo costo que ofrece todas las aplicaciones al igual que un computador de escritorio o una laptop basado en Linux.

- Para el desarrollo de la base de datos se integraron las aplicaciones PHP, MySQL y PhpMyAdmin, las cuales permitieron crear una base de datos visible en una interfaz web, permitiendo que se pueda administrar totalmente la base de datos de forma sencilla y amigable.

- Para definir las relaciones entre las tablas que pertenecen a la base de datos, se utilizó el programa MySQL Workbench, el cual permitió relacionar los datos ingresados de las etiquetas con los datos de los socios de las unidades de transporte y de los registros de los tiempos. De esta manera se puede observar los datos de forma clara y ordenada.

- La presentación de la base de datos en la web se desarrolló utilizando una plantilla prediseñada en PHP almacenada dentro de los directorios del sistema operativo del Raspberry. A esta interfaz se accede digitando en un navegador web la dirección estática o dinámica que se asigna al Raspberry.

- La información de los registros de tiempos de las unidades de transporte se presenta en una interfaz gráfica amigable y fácil de usar, la cual está almacenada en el Raspberry Pi; se puede acceder a ella desde cualquier dispositivo, utilizando la aplicación TeamViewer. Para ello el Raspberry debe estar conectado a una red de forma alámbrica o inalámbrica; para el caso del prototipo se utilizó el módulo inalámbrico Wifi 802.11n.

- La descarga de datos de la información se la puede realizar solo desde la interfaz de PhpMyAdmin a la cual ingresa únicamente el administrador, en varios formatos como PDF, CSV, etc.

- El diseño del presente prototipo servirá como base para el desarrollo de este tipo de sistemas en la vida real, ya que el mayor limitante del proyecto es el rango de lectura de las tarjetas, por lo que se podría reemplazar el lector RFID y los tags por dispositivos de mayor alcance.

4.2. Recomendaciones

- Para que una base de datos sea confiable, se recomienda crear respaldos continuos de la información, ya que en casos de pérdida o eliminación, ayudará a la reconstrucción de la base de datos.

- Para prevenir fallos de operación del sistema a causa de interrupciones por falta de servicio eléctrico, se recomienda instalar una fuente de energía alternativa, de ese modo se asegura que el prototipo por lo general no deje de funcionar.

- En un futuro, se podría implementar una opción de descarga de los datos en la página web, de esta manera los usuarios podrían utilizar los datos en la forma que a bien tuvieren.

- Se debe leer las especificaciones técnicas de cada dispositivo antes de trabajar con los mismos, con el fin de evitar daños o fallas en su utilización.

- El administrador de la base de datos debe tener conocimientos básicos de PhpMyAdmin para poder manejar la base, ya que será la persona responsable de toda la información que esta contenga.

- Para la implementación del sistema en la vida real, se pueden utilizar varios lectores RFID, como por ejemplo el lector de largo alcance modelo UHF AYU-900 de Rosslare, este posee un rango de lectura de hasta 12 metros, es compatible con el Raspberry ya que se puede establecer la comunicación directa entre ellos utilizando un cable Ethernet.

5. BIBLIOGRAFÍA

1. Gidekel, A. (2006). Introducción la identificación por Radio Frecuencia. Argentina.

2. Urbina, R. (2011). Tutorial sobre circuitos RFID. Tesis Licenciatura. Ingeniería en Electrónica y Computadoras. Departamento de Computación, Electrónica y Mecatrónica, Escuela de Ingeniería, Universidad de las Américas Puebla. Recuperado de http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/urbina_r_rd/capitulo2.pdf

3. Báez R., Chacano J. (2013). Tecnología RFID para el reconocimiento y asignación de pallets y rollos de cartulina en bodega. Facultad de Ciencias de la Ingeniería, Escuela de Ingeniería Civil Electrónica, Universidad Austral de Chile. Recuperado de http://cybertesis.uach.cl/tesis/uach/2013/bmfcib142t/doc/bmfcib142t.pdf

4. Báez R., Chacano J. (2013). Tecnología RFID para el reconocimiento y asignación de pallets y rollos de cartulina en bodega. Facultad de Ciencias de la Ingeniería, Escuela de Ingeniería Civil Electrónica, Universidad Austral de Chile. Recuperado de http://cybertesis.uach.cl/tesis/uach/2013/bmfcib142t/doc/bmfcib142t.pdf

5. Báez R., Chacano J. (2013). Tecnología RFID para el reconocimiento y asignación de pallets y rollos de cartulina en bodega. Facultad de Ciencias de la Ingeniería, Escuela de Ingeniería Civil Electrónica, Universidad Austral de Chile. Recuperado de http://cybertesis.uach.cl/tesis/uach/2013/bmfcib142t/doc/bmfcib142t.pdf

6. Portillo, J. Bermejo, A. Bernardos, A. (2008). Tecnología de identificación por Radiofrecuencia (RFID): Aplicaciones en el Ámbito de la Salud. Recuperado de http://www.madrimasd.org/informacionIDI/biblioteca/Publicacion/Vigilancia-tecnologica/descargar_documentos/fichero.asp?id=VT13_RFID.pdf

7. Portillo J., Bermejo A., Bernardos A. (2008). Tecnología de Identificación por Radiofrecuencia (RFID): Aplicaciones en el Ambito de la Salud. Madrid-España.

8. Minitrónica. (2017). Uso del módulo lector RFID-RC522 con Arduino. Recuperado de http://www.minitronica.com/blog/lector-rfid-rc522-arduino/

9. Los Autores.

10. Urbina, R. (2011). Tutorial sobre circuitos RFID. Tesis Licenciatura. Ingeniería en Electrónica y Computadoras. Departamento de Computación, Electrónica y Mecatrónica, Escuela de Ingeniería, Universidad de las Américas Puebla. Recuperado de http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/urbina_r_rd/capitulo3.pdf

11. RFID, “Tecnología RFID: Introducción,” RFID Magazine, p. 21, 2005.ShopNFC. (2015).

12. Los Autores.

13. Prometec. (2017). Los RFID. Recuperado de http://www.prometec.net/arduino-rfid/

14. Prometec. RFID: Identificación por RF. Recuperado de http://www.prometec.net/arduino-rfid/

15. NXP Semiconductors. (Abril, 2016). Datasheet MFRC522. Recuperado de https://www.nxp.com/documents/data_sheet/MFRC522.pdf

16. Arduino. (Enero, 2017). Arduino Mega. Recuperado de http://arduino.cl/arduino-mega-2560/

17. Los Autores.

18. Arduino. (2017). ATmega2560 – Arduino Pin Mapping. Recuperado de. https://www.arduino.cc/en/Hacking/PinMapping2560

19. Acamica. (Febrero, 2016). Ventajas y desventajas de Arduino. Recuperado de https://edgardosilvi.wordpress.com/2016/02/29/acamica-ventajas-y-desventajas-de-arduino/

20. RaspberryPi. (2017). Raspberry PI 2 Model B. Recuperado de https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

21. Los Autores.

22. Los Autores.

23. RaspberryPi. (2017). Raspberry PI Pinout. Recuperado de https://es.pinout.xyz/

24. EcuRed. (Mayo, 2017). Servidor HTTP Apache. Recuperado de https://www.ecured.cu/Servidor_HTTP_Apache

25. Prezi. (Diciembre, 2015). Servidor Apache. Recuperado de https://prezi.com/zg2vtfpdbje4/servidor-apache/

26. PHP. (Febrero, 2017). Que es PHP. Recuperado de http://php.net/manual/es/intro-whatis.php

27. Prezi. (Septiembre, 2014). Lenguaje PHP y SQL. Recuperado de https://prezi.com/3esgjofbtf_j/lenguaje-php-y-sql/

28. Prezi. (Febrero, 2015). MySQL. Recuperado de https://prezi.com/0h1ddup36its/mysql/

29. phpMyAdmin. (2017). Bringing MySQL to the web. Recuperado de https://www.phpmyadmin.net/

30. Prezi. (Junio, 2014). PHPMYADMIN. Recuperado de https://prezi.com/yanq2ucf_5yz/phpmyadmin/

31. Requisitos. (2016). Servidor web. Recuperado de https://docs.phpmyadmin.net/es/latest/require.html

32. RaspberryPi. (2017). Python. Recuperado de https://www.raspberrypi.org/documentation/usage/python/

33. Prezi. (Marzo, 2014). Presentación Python. Recuperado de https://prezi.com/nmtiwfi7guh2/presentacion-python/

34. Prezi. (Julio, 2015). MySQL WORKBENCH. Recuperado dehttps://prezi.com/_vllrwpudkcz/mysql-workbench/

35. PanamaHitek. (Octubre, 2014). Cómo funciona el protocolo SPI. Recuperado de http://panamahitek.com/como-funciona-el-protocolo-spi/

36. Sparkfun. (2013). Serial Peripheral Interface (SPI). Recuperado de https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

37. Wikipedia. (Abril, 2017). TeamViewer. Recuperado de https://es.wikipedia.org/wiki/TeamViewer

38. GitHub. (Abril, 2017). Control Remoto para Pi. Recuperado de https://github.com/aikoncwd/aikoncwd-rpi-mediacenter/issues/65

39. aulaClic. (Julio, 2001). Lenguaje de definición de datos. Recuperado de http://www.aulaclic.es/sql/t_8_2.html

40. Reglas de Negocio, Normalización, SQL (DDL - DML - TCL - DCL). Recuperado de http://unefabasededatos2009.blogspot.com/2009/04/reglas-de-negocio-normalizacion-sql-ddl.html

ANEXOS

ANEXO A: Manual de Usuario Prototipo Traspduino V1

ANEXO B: Características Técnicas de los Dispositivos y Programas utilizados en el Prototipo.

ANEXO C: Maqueta del Prototipo

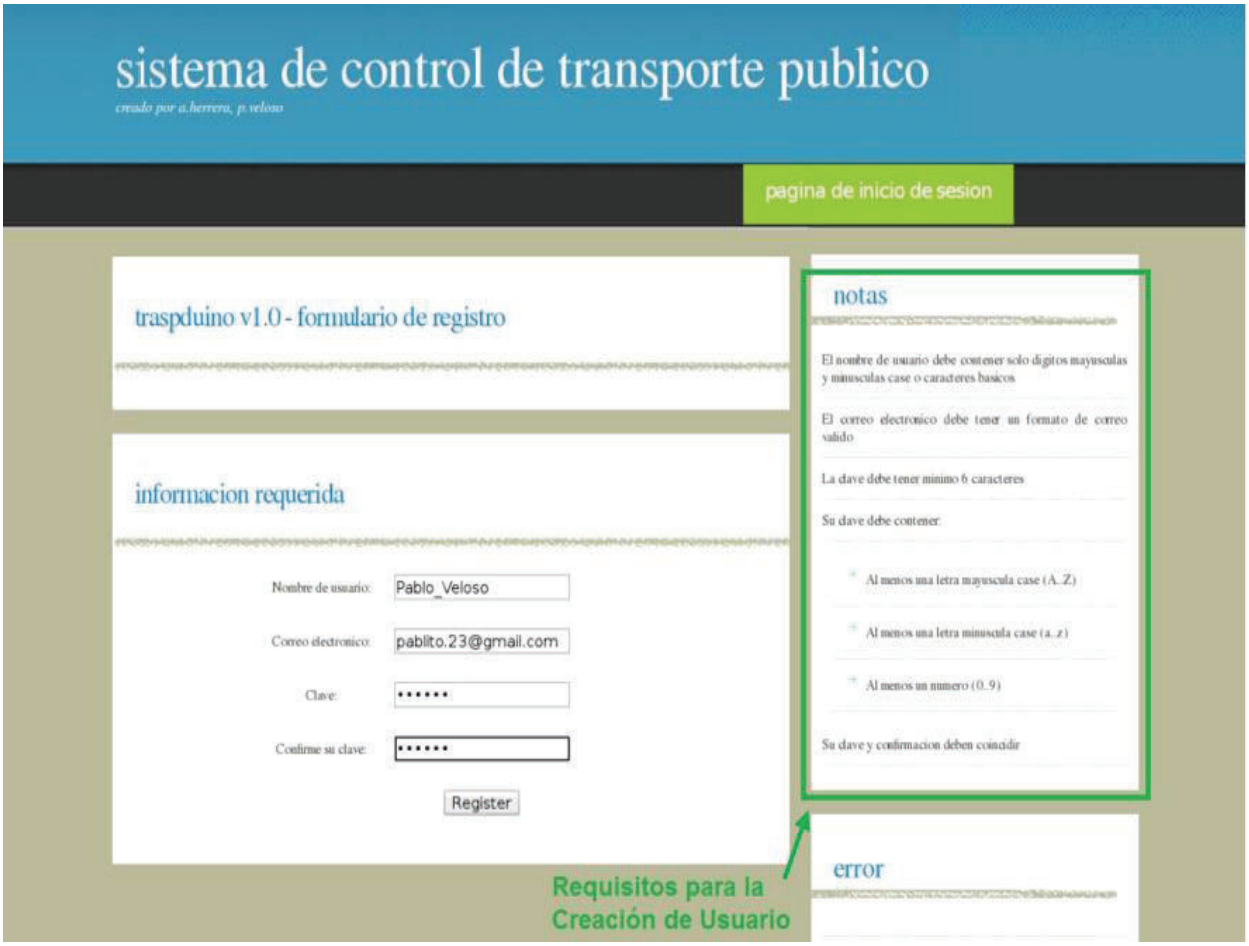
ANEXO D: Códigos de programación del prototipo

ANEXO A: Manual de Usuario Prototipo Traspduino V1.0

Para el ingreso a la página web es necesario abrir un navegador web y digitar la dirección IP que le asigne estáticamente al Raspberry (usuario) o dinámicamente (la red). Una vez dentro de la página, se debe realizar la creación de un usuario para de esta manera autentificar el ingreso.

1.- Una vez en la interfaz Web, hacer clic en “regístrese”.

2.- Para el ingreso, se debe llenar los campos: Nombre del Usuario, Correo electrónico, Clave y luego confirmarla, siguiendo los parámetros que se describen en la interfaz web, de esta forma se tiene acceso a los datos de las unidades de transporte.



Los requisitos necesarios para la creación de usuario son:

- El nombre del usuario debe contener solo dígitos, mayúsculas y minúsculas o caracteres básicos.
- El correo electrónico debe tener un formato de correo electrónico válido.
    - La clave debe tener minino 6 caracteres.
    - Al menos una letra mayúscula (A….Z).
    - Al menos una letra minúscula (a…z).
    - Al menos un número (0…9).
- La clave y la confirmación deben coincidir.

3.- Una vez completados todos los campos necesarios para el registro, aparecerá un mensaje indicando que el registro ha sido completado con éxito y se puede volver a la página inicial para ingresar utilizando el usuario creado.

Acceso a la página web con un usuario registrado

Luego de registrar la información del usuario, iniciar la sesión en la interfaz web con el correo y clave que se registró anteriormente.

Después de realizar este proceso se podrá visualizar la información que se presenta en la interfaz, es decir los datos de las unidades de transporte, de los socios o dueños de las unidades y el tiempo que marcan al pasar por el punto de control.

1.- Ingresar el correo y clave registrados.

En la sección de error se nos desplegará un mensaje cuando los datos ingresados al iniciar la sesión sean incorrectos.

2.- Una vez que se ingresa se puede visualizar la información de las Unidades de Transporte, Socios y Registros del tiempo.

Esta imagen muestra la pestaña con la información de las unidades de transporte registradas, en la cual se puede visualizar los datos de los buses, tales como: la placa, el modelo, la marca y el socio.

En la siguiente pestaña se muestra la información de los socios en la que se incluye un número identificador, número de cédula, nombre, teléfono y dirección.

En la última pestaña se muestra la información del tiempo de las unidades de transporte cuando pasan por el punto de control.

ANEXO B: Características Técnicas de los Dispositivos y Programas utilizados en el Prototipo.

1.- Características técnicas de un tag pasivo NFC. [12]

ETIQUETA NFC RFID	
Memoria de almacenamiento	EEPROM de 1KB
Rango de lectura	0 a 60 mm
Duración (vida útil)	10 años máx.
Frecuencia de operación	13.56Mhz
Distancia de lectura	0 a 60mm
Identificador único	4 Bytes
Protocolo de funcionamiento	ISO14443A
Dimensiones	32 x 14 x 4 mm
Materiales de construcción	PVC - ABS

2.- Características técnicas lector RFID [24]

MÓDULO LECTOR RFID RC522	
Microcontrolador	MFRC522
Corriente de operación	13-26mA a 3.3V
Corriente de *Stand by*	0-13mA a 3.3V
Corriente de *sleep mode*	<80uA
Corriente máxima	30mA
Frecuencia de operación	13.56Mhz
Distancia de lectura	0 a 60mm
Protocolo de Comunicación	SPI
Velocidad máxima de lectura de datos	10Mbit/s
Dimensiones	40 x 60 mm

3.- Características técnicas Arduino Mega 2560

ARDUINO MEGA 2560	
Microcontrolador	Atmega2560
Voltaje de Funcionamiento	5V
Voltaje de entrada (Recomendado)	7-12V
Voltaje de Entrada (limite)	6-20V
Pines Digitales E/S	54 (15 con salida PWM)
Pines de entrada Analógica	16
Corriente Continua para pines de E/S	20mA
Corriente CC para Pin 3.3V	50mA
Memoria Flash	256KB, 8 KB utilizado por el gestor de arranque
SRAM	8KB
EEPROM	4KB
Velocidad de Reloj	16 MHz
Medidas	101,52 mm x 53,3 mm
Peso	37g

4.- Identificación de Pines Arduino Mega 2560. [19]

Los pines digitales (*DIGITAL PINS*) que van desde 0 hasta 54, se puede utilizar como una entrada o salida, utilizando las funciones pinMode (), *digitalWrite* (), y *digitalRead* (). Estos operan a 5 voltios. Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia de *pull-up* (desconectada por defecto) de 20-50 *kOhms.*

Los pines de entradas analógicas (*ANALOG PINS*) van desde A0 hasta A15, son entradas que pueden variar su voltaje en un intervalo de –Vcc y + Vcc; este tipo de entradas son más escasas, más lentas y más caras que las entradas digitales.

Los pines de entradas analógicas (*ANALOG PINS*) van desde A0 hasta A15, son entradas que pueden variar su voltaje en un intervalo de –Vcc y + Vcc; este tipo de entradas son más escasas, más lentas y más caras que las entradas digitales.

Los pines SPI (*SERIAL PERIPHERAL INTERFACE* - INTERFAZ PERIFÉRICA SERIAL) (Pines 50, 51, 52 y 53), se utilizan para la comunicación entre dos microcontroladores.

PIN	**FUNCION**
0	Serial / Digital / RX Datos TTL en serie
1	Serial / Digital / TX Datos TTL en serie
2	Digital / PWM / Interrupción Externa
3	Digital / PWM / Interrupción Externa
4	Digital / PWM
5	Digital / PWM / Pin Analógico Relacionado
6 al 13	Digital / PWM
14	Digital / Serial / TX Datos TTL en serie
15	Digital / Serial / RX Datos TTL en serie
16	Digital / Serial / TX Datos TTL en serie
17	Digital / Serial / RX Datos TTL en serie
18	Digital / Serial / TX Datos TTL en serie / Interrupción Externa
19	Digital / Serial / RX Datos TTL en serie / Interrupción Externa
20	Digital / Serial / Interrupción Externa
21	Digital / Serial / Interrupción Externa
22 al 43	Digital
44-45-46	Digital / PWM
47-48-49	Digital
50	Digital / MISO
51	Digital / MOSI
52	Digital / CLK
53	Digital / SS
A0 al A15	Analógico

5.- Identificación de Pines de Propósito General de Entrada/Salida del Raspberry PI 2B [16]

PINES GPIO RASPBERRY PI 2-B	
PIN	**FUNCIÓN**
1	Alimentación 3.3V
2	Alimentación 5V
3	BCM 2 - SDA
4	Alimentación 5V
5	BCM 3 - SCL
6	GND
7	BCM 4 - GPCLK0
8	BCM 14 - TXD
9	GND
10	BCM 15- RXD
11	BCM 17
12	BCM 18 - PWM0
13	BCM 27
14	GND
15	BCM 22
16	BCM 23
17	Alimentación 3.3V
18	BCM 24
19	BCM 10 - MOSI
20	GND

PINES GPIO RASPBERRY PI 2-B	
PIN	**FUNCIÓN**
21	BCM 9 - MISO
22	BCM 25
23	BCM 11 - SCLK
24	BCM 8 - CE0
25	GND
26	BCM 7 - CE1
27	BCM 0 - ID_SD
28	BCM 1 - ID_SC
29	BCM 5
30	GND
31	BCM 6
32	BCM 12 - PWM0
33	BCM 13 - PWM 1
34	GND
35	BCM 19 - MISO
36	BCM 16
37	BCM 26
38	BCM 20 - MOSI
39	GND
40	BCM 21 - SCLK

6.- Características Técnicas Raspberry PI 2B

RASPBERRY PI, MODELO B, GENERACIÓN 2	
Fabricante	Fundación Raspberry Pi
Modelo	Raspberry Pi, Modelo B, Generación 2
Fecha de Lanzamiento	Febrero, 2015
Sistema Operativo	Raspbian Jessie con Pixel
Potencias Nominales	4,0 W 800 mA
Fuente de Alimentación	5 V, vía MicroUSB
CPU	ARM Cortex-A7 a 900 MHz
Memoria (SDRAM)	1 GB
Salidas de Audio	Analógica 3,5 mm
Almacenamiento	MicroSDHC slot
Conector de Red	10/100 Mbit/s Ethernet
Tamaño	85,60 mm x 56,5 mm
Peso	45 g

7**.- Características de PHP.**

- Estabilidad, tiene un sistema robusto y estable.
- Seguridad, maneja distintos niveles de seguridad.
- Simplicidad al momento de su utilización.
- Es multiplataforma (Windows, Linux, Mac)

8.- Características de MySQL

- MySQL es un sistema de administración de base de datos. Una base de datos es una colección estructurada de tablas que contienen datos.
- Es un sistema de administración relacional de bases de datos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Brinda flexibilidad y robustez.
- Disponibilidad en gran cantidad de plataformas y sistemas.

- Conectividad segura.
- Escrito en una combinación de lenguaje C y C++.

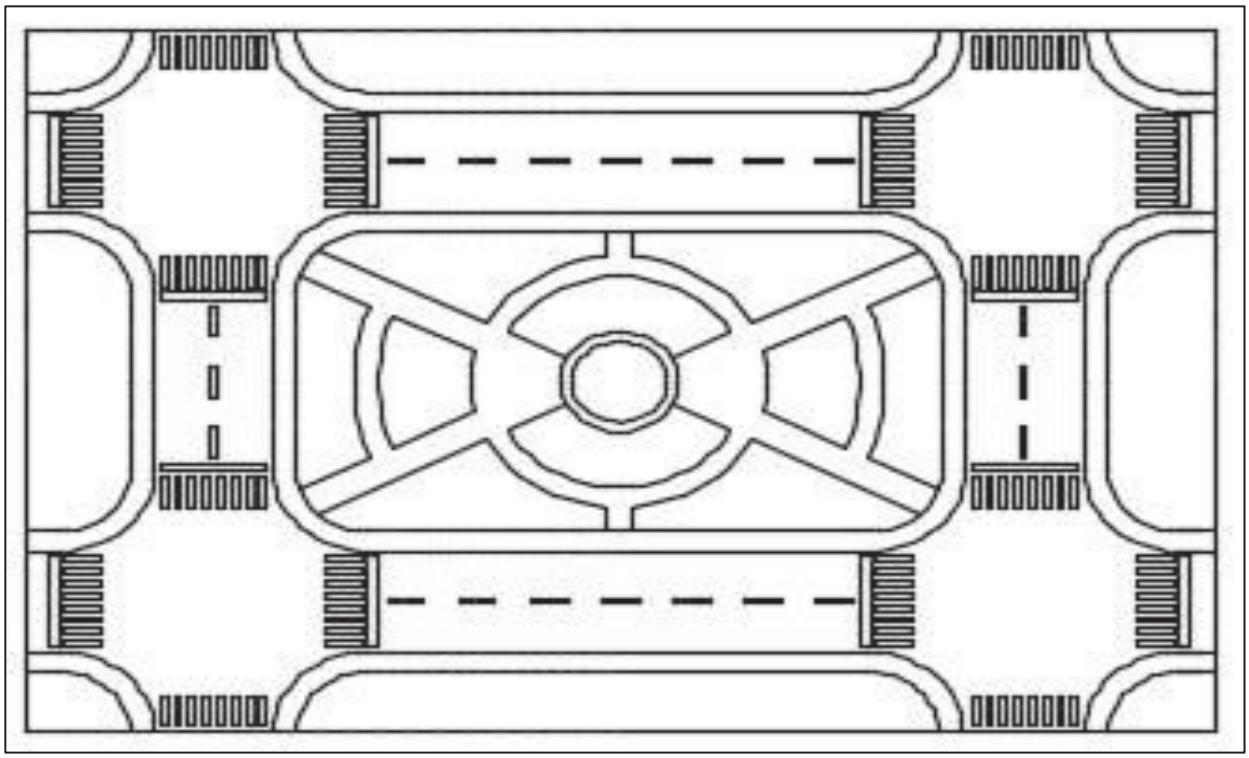
9.- Características de PhpMyAdmin. [31]

- Administración completa de tablas y de las Bases de Datos.
- Ejecuta sentencias SQL.
- Administra usuarios y privilegios de MySQL

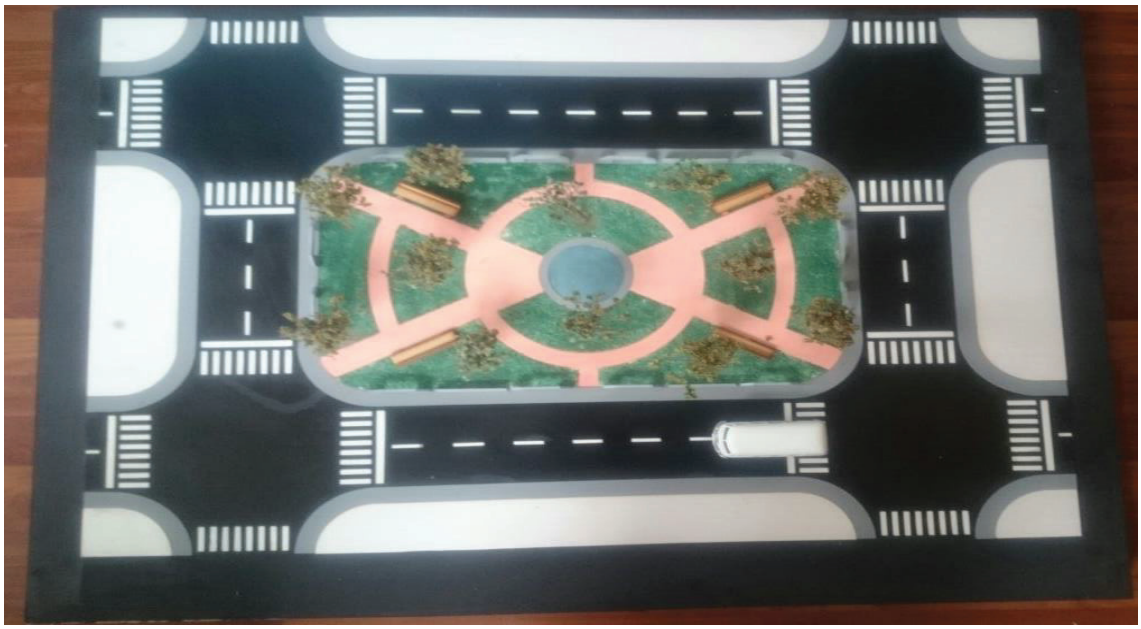
10.- Características de Python. [34]

- Simple y sencillo de aprender.
- Libre y de fuente abierta "*Open Source*".
- Programación orientada a objetos.
- Librerías extendidas.

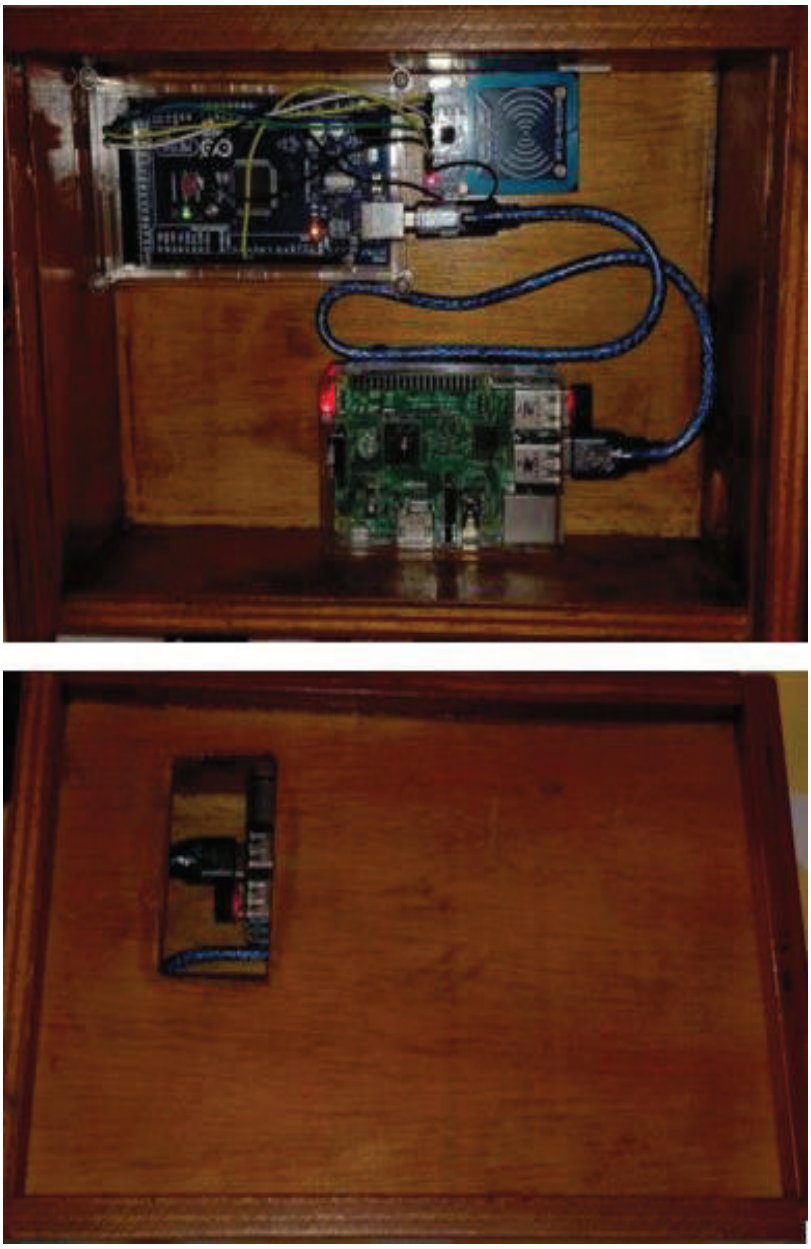
ANEXO C: Maqueta del Prototipo



Plano de la Maqueta



Maqueta del Prototipo



Caja de almacenamiento de los dispositivos del Prototipo

ANEXO D: Código de la estructura de la base de datos en SQL.

```
-- phpMyAdmin SQL Dump
-- version 4.2.12deb2+deb8u2
-- http://www.phpmyadmin.net
--
-- Servidor: localhost
-- Versión del servidor: 5.5.54-0+deb8u1
-- Versión de PHP: 7.0.15-1~bpo8+1

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Base de datos: `traspduino`
--
-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `login_attempts`
--

CREATE TABLE IF NOT EXISTS `login_attempts` (
`user_id` int(11) NOT NULL,
  `time` varchar(30) COLLATE utf8_spanish_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;

-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `lugar_control`
--

CREATE TABLE IF NOT EXISTS `lugar_control` (
`idlugar` int(11) NOT NULL,
  `numero_equipo` varchar(3) COLLATE utf8_spanish_ci NOT NULL,
  `fecha_registro` date NOT NULL,
  `latitud` varchar(15) COLLATE utf8_spanish_ci NOT NULL,
  `longitud` varchar(15) COLLATE utf8_spanish_ci NOT NULL,
  `rutas_idrutas` int(11) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;

--
-- Volcado de datos para la tabla `lugar_control`
--

INSERT INTO `lugar_control` (`idlugar`, `numero_equipo`, `fecha_registro`, `latitud`, `longitud`,
`rutas_idrutas`) VALUES
(1, 'EQ1', '2017-01-12', '-0.3383977', '-78.5503268', 2);
```

```
-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `members`
--

CREATE TABLE IF NOT EXISTS `members` (
`id` int(11) NOT NULL,
  `username` varchar(30) COLLATE utf8_spanish_ci NOT NULL,
  `email` varchar(50) COLLATE utf8_spanish_ci NOT NULL,
  `password` char(128) COLLATE utf8_spanish_ci NOT NULL,
  `salt` char(128) COLLATE utf8_spanish_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `registro_tiempos`
--

CREATE TABLE IF NOT EXISTS `registro_tiempos` (
`idregistro` int(11) NOT NULL,
  `fecha_registro` date NOT NULL,
  `hora_registro` time NOT NULL,
  `lugar_control_idlugar` int(11) NOT NULL,
  `lugar_control_rutas_idrutas` int(11) NOT NULL,
  `unidades_idunidades` varchar(11) COLLATE utf8_spanish_ci NOT NULL,
  `unidades_socios_idsocios` int(11) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=44 DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
--
-- Volcado de datos para la tabla `registro_tiempos`
--

INSERT INTO `registro_tiempos` (`idregistro`, `fecha_registro`, `hora_registro`, `lugar_control_idlugar`,
`lugar_control_rutas_idrutas`, `unidades_idunidades`, `unidades_socios_idsocios`) VALUES

-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `rutas`
--

CREATE TABLE IF NOT EXISTS `rutas` (
`idrutas` int(11) NOT NULL,
  `origen` varchar(45) COLLATE utf8_spanish_ci DEFAULT NULL,
  `destino` varchar(45) COLLATE utf8_spanish_ci DEFAULT NULL
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;

--
-- Volcado de datos para la tabla `rutas`
--
INSERT INTO `rutas` (`idrutas`, `origen`, `destino`) VALUES
(1, 'CDLA. EJERCITO', 'MARÍN'),
(2, 'GUAMANÍ', 'MARÍN');

-- --------------------------------------------------------
```

```
--
-- Estructura de tabla para la tabla `socios`
--

CREATE TABLE IF NOT EXISTS `socios` (
`idsocios` int(11) NOT NULL,
  `ci` varchar(10) COLLATE utf8_spanish_ci NOT NULL,
  `nombre` varchar(35) COLLATE utf8_spanish_ci NOT NULL,
  `telefono` varchar(10) COLLATE utf8_spanish_ci NOT NULL,
  `direccion` varchar(45) COLLATE utf8_spanish_ci NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;

--
-- Volcado de datos para la tabla `socios`
--

INSERT INTO `socios` (`idsocios`, `ci`, `nombre`, `telefono`, `direccion`) VALUES
(1, '1716315047', 'ROBERTO TOAPANTA', '0999098696', 'CDLA. EJERCITO'),
(2, '1726354876', 'ANDRES HERRERA', '0938472635', 'MENA'),
(3, '043847593', 'PABLO VELOSO', '0987676567', 'BARRIONUEVO'),
(4, '1635478576', 'MANUEL CARRASCO', '0986567876', 'GUAMANI');
-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `unidades`
--

CREATE TABLE IF NOT EXISTS `unidades` (
  `idunidades` varchar(11) COLLATE utf8_spanish_ci NOT NULL,
  `placa` varchar(8) COLLATE utf8_spanish_ci NOT NULL,
  `modelo` varchar(15) COLLATE utf8_spanish_ci NOT NULL,
  `marca` varchar(15) COLLATE utf8_spanish_ci NOT NULL,
  `socios_idsocios` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;

--
-- Volcado de datos para la tabla `unidades`
--

INSERT INTO `unidades` (`idunidades`, `placa`, `modelo`, `marca`, `socios_idsocios`) VALUES
('44 F6 14 2B', 'CRD1768', 'B12', 'VOLVO', 2),
('A4 AE CF A5', 'GEV1245', 'AXOR', 'MERCEDES BENZ', 3),
('E5 7A 1D 00', 'PGV0123', 'JEEP', 'MAZDA', 1),
('F6 50 CE A5', 'CDE1234', 'AXOR', 'MERCEDES BENZ', 4);

--
-- Índices para tablas volcadas
--
--
-- Indices de la tabla `login_attempts`
--
ALTER TABLE `login_attempts`
 ADD PRIMARY KEY (`user_id`);
```

```
--
-- Indices de la tabla `lugar_control`
--
ALTER TABLE `lugar_control`
 ADD PRIMARY KEY (`idlugar`,`rutas_idrutas`), ADD KEY `fk_lugar_control_rutas1_idx`
(`rutas_idrutas`);

--
-- Indices de la tabla `members`
--
ALTER TABLE `members`
 ADD PRIMARY KEY (`id`);

--
-- Indices de la tabla `registro_tiempos`
--
ALTER TABLE `registro_tiempos`
 ADD PRIMARY KEY
(`idregistro`,`lugar_control_idlugar`,`lugar_control_rutas_idrutas`,`unidades_idunidades`,`unidades_so
cios_idsocios`), ADD KEY `fk_registro_tiempos_lugar_control1_idx`
(`lugar_control_idlugar`,`lugar_control_rutas_idrutas`), ADD KEY `fk_registro_tiempos_unidades1_idx`
(`unidades_idunidades`,`unidades_socios_idsocios`);

--
-- Indices de la tabla `rutas`
--
ALTER TABLE `rutas`
 ADD PRIMARY KEY (`idrutas`);

--
-- Indices de la tabla `socios`
--
ALTER TABLE `socios`
 ADD PRIMARY KEY (`idsocios`);

--
-- Indices de la tabla `unidades`
--
ALTER TABLE `unidades`
 ADD PRIMARY KEY (`idunidades`,`socios_idsocios`), ADD KEY `fk_unidades_socios1_idx`
(`socios_idsocios`);

--
-- AUTO_INCREMENT de las tablas volcadas
--

--
-- AUTO_INCREMENT de la tabla `login_attempts`
--
ALTER TABLE `login_attempts`
MODIFY `user_id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT de la tabla `lugar_control`
--
```

```
ALTER TABLE `lugar_control`
MODIFY `idlugar` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=2;
--
-- AUTO_INCREMENT de la tabla `members`
--
ALTER TABLE `members`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT de la tabla `registro_tiempos`
--
ALTER TABLE `registro_tiempos`
MODIFY `idregistro` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=44;
--
-- AUTO_INCREMENT de la tabla `rutas`
--
ALTER TABLE `rutas`
MODIFY `idrutas` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=3;
--
-- AUTO_INCREMENT de la tabla `socios`
--
ALTER TABLE `socios`
MODIFY `idsocios` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=5;
--
-- Restricciones para tablas volcadas
--
--
-- Filtros para la tabla `lugar_control`
--
ALTER TABLE `lugar_control`
ADD CONSTRAINT `fk_lugar_control_rutas1` FOREIGN KEY (`rutas_idrutas`) REFERENCES `rutas`
(`idrutas`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--
-- Filtros para la tabla `registro_tiempos`
--
ALTER TABLE `registro_tiempos`
ADD CONSTRAINT `fk_registro_tiempos_lugar_control1` FOREIGN KEY (`lugar_control_idlugar`,
`lugar_control_rutas_idrutas`) REFERENCES `lugar_control` (`idlugar`, `rutas_idrutas`) ON DELETE
NO ACTION ON UPDATE NO ACTION,
ADD CONSTRAINT `fk_registro_tiempos_unidades1` FOREIGN KEY (`unidades_idunidades`,
`unidades_socios_idsocios`) REFERENCES `unidades` (`idunidades`, `socios_idsocios`) ON DELETE
NO ACTION ON UPDATE NO ACTION;

--
-- Filtros para la tabla `unidades`
--
ALTER TABLE `unidades`
ADD CONSTRAINT `fk_unidades_socios1` FOREIGN KEY (`socios_idsocios`) REFERENCES
`socios` (`idsocios`) ON DELETE NO ACTION ON UPDATE NO ACTION;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```