"Responsabilidad con pensamiento positivo"

UNIVERSIDAD TECNOLÓGICA ISRAEL

TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE:

INGENIERO EN ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES

TEMA:

Desarrollo de un prototipo de una Red de Sensores Inalámbricos para monitorear por smartphone la disponibilidad de plazas de un estacionamiento utilizando tecnologías de IoT.

AUTOR:

Fausto Renán Bautista Palate

TUTOR:

Ing. Morales Arévalo Flavio David

Quito, Ecuador

Año 2018

DECLARACIÓN

Yo, Fausto Renán Bautista Palate, estudiante de la carrera de Electrónica Digital y Telecomunicaciones, perteneciente a la Universidad Tecnológica Israel, declaro que el contenido aquí descrito es de mi autoría, y de mi absoluta responsabilidad legal.

Quito D.M., agosto del 2018

Fausto Renán Bautista Palate
C.I.: 1714436647

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación certifico:

Que el trabajo de titulación "DESARROLLO DE UN PROTOTIPO DE UNA RED DE SENSORES INALÁMBRICOS PARA MONITOREAR POR SMARTPHONE LA DISPONIBILIDAD DE PLAZAS DE UN ESTACIONAMIENTO UTILIZANDO TECNOLOGÍAS DE IoT", presentado por el Sr. Fausto Renán Bautista Palate, estudiante de la carrera de Electrónica Digital y Telecomunicaciones, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

Quito D.M. agosto del 2018

TUTOR

……..

Ing. Flavio Morales Arévalo, Mg

AGRADECIMIENTO

Mi agradecimiento a Dios y a mi familia que por su apoyo incondicional ha hecho posible alcanzar mis metas y llegar a estas instancias. Este es el fruto de su sacrificio y mi manera de decirles que les estaré eternamente agradecido.

Fausto Renán Bautista Palate

DEDICATORIA

Este trabajo lo dedico a mi Dios que me cuida con salud y vida y también a mi madre que con su sacrificio y amor supo brindarme mi educación secundaria y apoyarme en mi educación superior. Porque la he visto luchar conmigo y velando que no nos falte nada y al ser el pilar más fuerte de mi familia, su buen ejemplo me ha inspirado a ser siempre mejor persona, pero sobre todo porque me ha enseñado que el amar a tus seres queridos consiste en demostrarlo y no solo que quede en palabras.

Fausto Renán Bautista Palate

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Tabla de Fórmulas

RESUMEN

El presente proyecto pretende mostrar el uso de una aplicación de tecnologías y plataformas utilizadas en Internet de las cosas, para subir información a la nube y ser aprovechada de diferentes maneras como por ejemplo servicios a la comunidad. La misma fue implementada como prototipo en los parqueaderos del Conjunto Habitacional Camino Real en el centro histórico de la Ciudad de Quito.

El llegar a conocer el status de las plazas del estacionamiento de manera remota se traduciría en no realizar viajes innecesarios a dicho sitio, con la consecuente pérdida de tiempo y por ende a no contribuir con embotellamientos. Si este sistema se aplica en estacionamientos municipales o privados, cuyas sedes se encuentren dentro de zonas críticas de alto índice de tráfico; resultaría muy provechoso para los conductores conocer de antemano si existen puestos libres evitando la entrada innecesaria de más vehículos a la zona de tráfico y ayudar a reducir congestiones.

El prototipo consta de sensores que monitorean cada estacionamiento y dicha información es transmitida inalámbricamente hacia una tarjeta receptora en donde el microcontrolador ATMEGA628 procesa la información y la misma es subida a la nube de internet con la ayuda del microcontrolador ESP8266 que utiliza el protocolo IEEE 802.11. Por medio de un servidor web levantado en la plataforma IoT conocida como ThingSpeak, se registra la información recolectada y se la presenta gráficamente para su administración. De igual manera dicha información es enviada por la nube hasta el smartphone donde también se presenta y consigue el monitoreo remoto.

Palabras claves: IoT, sensor ultrasónico, microcontrolador ATMEGA328, microcontrolador ESP8266, IEEE802.11, servidor ThingSpeak.

ABSTRACT

This project aims to show the use of an application of technologies and platforms used in the Internet of Things, to upload information to the cloud and be used in different ways such as community services. It was implemented as a prototype in the parking from the Conjunto Habitacional Camino Real in the historic center of Quito City.

Getting to know the status of parking spaces remotely would result in not making unnecessary trips to the site, with the consequent loss of time and therefore not contribute to traffic jams. If this system is applied in municipal or private parking, whose headquarters are located within critical zones of high traffic index; It would be very helpful for drivers to know beforehand if there are free positions preventing the unnecessary entry of more vehicles into the traffic area and help reduce congestion.

The prototype consists of sensors that monitor each parking lot and this information is transmitted wirelessly to a receiving card where the ATMEGA628 microcontroller processes the information and it is uploaded to the internet cloud with the help of the ESP8266 microcontroller that uses the IEEE 802.11 protocol. Through a web server built on the IoT platform known as ThingSpeak, the information collected is recorded and presented graphically for administration. In the same way, this information is sent by the cloud to the smartphone where it is also presented and gets remote monitoring.

Keywords: IoT, sensor, microcontroller ATMEGA328, ESP8266, IEEE802.11, ThingSpeak server.

INTRODUCCIÓN

ANTECEDENTES DE LA SITUACIÓN OBJETO DE ESTUDIO

En la actualidad en las zonas urbanas es conocido el tema de embotellamientos y el crecimiento del parque automotor. Esto hace que el encontrar disponible un sitio de estacionamiento se torne molestoso sobre todo en zonas de alto tráfico vehicular con calles angostas y de un solo sentido que hacen que el dar una vuelta a la manzana en busca de estacionamiento ocasionen pérdidas de tiempo que retrasan las actividades y las consecuentes molestias al conductor.

En la ciudad ya hay establecimientos de Centros Comerciales que disponen de formas de señalización en sus estacionamientos; sin embargo, el beneficio es solamente dentro de sus áreas de competencia, mientras que, en estacionamientos públicos, zonas azules u otras zonas de estacionamiento particular no se tiene manera de saber de su disponibilidad como para evitar estos inconvenientes.

PLANTEAMIENTO DEL PROBLEMA

Para un conductor que tiene pensado dejar su vehículo en un sitio de estacionamiento dentro de una zona de alto tráfico y no lo encuentra, le ocasiona molestias que retrasan sus actividades como embotellamientos, pérdidas de tiempo, desperdicio de combustible, además que contribuye con el aumento de contaminación al medio ambiente.

De igual manera cuando existen eventos especiales como partidos de fútbol, conciertos, desfiles en los cuales la demanda de estacionamientos en los sitios o sus alrededores aumentan notablemente.

Se requiere de un método que permita a un conductor el conocer previamente la disponibilidad de plazas de estacionamiento.

Actualmente existen tecnologías de comunicación que en conjunto con Redes Locales de Sensores hagan posible monitorear la ocupación de las plazas de un estacionamiento y esta

información pueda ser receptada por usuarios a través de su smartphone de donde se plantea la siguiente interrogante:

¿Se puede desarrollar un sistema de sensores inalámbricos para estacionamientos de zonas urbanas utilizando tecnologías para Internet de las cosas y que permita conocer previamente la disponibilidad de plazas a través de una aplicación móvil Android de un smartphone?

JUSTIFICACIÓN

Para poder solventar molestias en conductores que buscan estacionamientos y que a la vez contribuyen con los embotellamientos de las zonas de mayor carga vehicular se requiere un método para conocer previamente el status de las plazas de estacionamiento consistente en una red de sensores inalámbricos, pudiendo usarse además la nube de internet para que dicha información pueda ser accesible a cualquier persona desde un smartphone que posea conexión de datos, desde cualquier sitio y evitar las molestias por no poder encontrar un sitio con plazas disponibles.

Con ello se busca que los establecimientos de estacionamientos puedan enviar la información al público en general sobre la cantidad de puestos libres dispone como para que los conductores sepan cual es la mejor opción.

El estacionamiento del Conjunto Habitacional Camino Real, ofrece un buen escenario para la implementación de un prototipo del presente proyecto.

Objetivos

Objetivo general

Desarrollar un prototipo de una Red de Sensores Inalámbricos que sirva para monitorear la disponibilidad de plazas de un sitio de estacionamientos a través de una aplicación en un smartphone Android mediante el uso de tecnologías que se utilizan para IoT Internet de las cosas.

Objetivos específicos

- Desarrollar el prototipo para los Estacionamientos del Conjunto Habitacional Camino Real; el mismo que funcione con sensores inalámbricos y cuya disponibilidad sea visible en un smartphone.
- Establecer la red de sensores utilizando tecnologías de comunicación inalámbricas que son usadas en la actualidad en Aplicaciones de desarrollo de Internet de las cosas IoT como los protocolos IEEE 802.15.4 o IEEE 802.11
- Transmitir la información recolectada por el microcontrolador hacia el dispositivo móvil a través de la nube de Internet.
- Implementar una aplicación móvil para smartphone con sistema operativo Android, que permita visualizar el status de los puestos de parqueo en tiempo real y llevar un conteo del tiempo transcurrido en su ocupación.
- Implementar la tarjeta electrónica para un microcontrolador que procese la información recibida desde los sensores y desarrollar la programación del mismo.
- Realizar pruebas de funcionamiento, validación y análisis de resultados del prototipo.

Descripción de los capítulos

En el capítulo 1 se desarrolla brevemente la teoría en la que se basa el presente proyecto describiendo los diferentes elementos, tecnologías y plataformas IoT utilizados y esquematizado en fases de la siguiente manera:

Etapa de acceso: Microcontrolador Sensor ultrasónico, microcontrolador ATMEGA328, los módulos de radio frecuencia Tx/Rx 433Mhz.

Etapa de interface: Módulo wi-fi ESP826, programación IDE Arduino.

Etapa en la nube: ThingSpeak.

Etapa de aplicación en el Smartphone: Mit App Inventor.

Se menciona brevemente algunas aplicaciones existentes que dan a este servicio en otros países como PARKOOL, BLUEPARKING y las diferencias en el modo de funcionamiento con el propuesto en el presente proyecto.

En el capítulo 2 Se plantea la implementación de los dispositivos del capítulo 1 en donde el sensor ultrasónico generará información del status de cada estacionamiento y luego será transmitida por un módulo transmisor RF de 400Mhz hasta la placa de control, la misma que contará con el microcontrolador ATMEGA 328 para la parte del procesamiento.

Este microcontrolador procesa la información y por medio del módulo wi-fi ESP8266 comunica la información hacia la nube de internet mediante el protocolo IEEE802.11 y la salida de internet provisto por un ISP.

En la nube de Internet se propone utilizar la plataforma IoT denominada ThingSpeak para mostrar de manera gráfica la información de cada sensor de cada estacionamiento y el estatus si se encuentra libre u ocupado. Adicionalmente se pretende dar la posibilidad de reservar cualquiera de las plazas del sitio estacionamientos de prueba.

Finalmente se plantea la propuesta de una aplicación para móviles Android cuyo ejecutable tenga extensión .apk la cual es propia de archivos ejecutables para dispositivos con sistema operativo Android.

En el capítulo 3 se desarrolla de manera detallada el diseño de la placa del sensor y de la placa del microcontrolador ATMEGA328 a través de los diagramas de los circuitos diseñados. Se muestra las simulaciones realizadas en Proteus para validar los parámetros calculados para la puesta en marcha del hardware.

Para la parte lógica se muestra el desarrollo del programa donde consta los parámetros de red seteados para la conexión con el módulo ESP8266 y los direccionamientos IP asignados por DHCP por parte del ruteador del ISP.

Se muestra los resultados obtenidos después de las pruebas realizadas en el proto para validar los valores de voltaje y corriente de la placa, así como también se comprueba la llegada de la información hasta el Servidor de ThingSpeak a través del Internet.

Una vez que la información se encuentra disponible a través de la nube se muestra el desarrollo del ambiente de visualización de los estacionamientos para que sea comprensible para un usuario cualquiera y finalmente se muestra el diseño de la aplicación para conseguir el funcionamiento del sistema desde el Smartphone. Se verifica el funcionamiento de la aplicación tanto en la visualización como ejecución de instrucciones básicas desde la aplicación.

Finalmente se expone las conclusiones obtenidas una vez que los objetivos han sido alcanzados, así como también las recomendaciones que optimicen al producto final.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

1.1. Estado de Arte

En países como España existen aplicaciones como *Blue Parking*, en Argentina *Parkool*, en Mexico *Parkeo*, en donde las mismas están operativas y tienen como principio básico de funcionamiento el intercambio de información de la comunidad de usuarios de la aplicación.

Esto hace que la calidad del servicio de algunas aplicaciones dependa directamente de una constante y eficaz participación de los miembros de la comunidad el cual vendría a ser un parámetro del cual no se tiene control.

Son los usuarios los que potencian la información, los mapas y la navegación; cuantos más usuarios conduzcan con la aplicación abierta y de manera correcta, mejor será su funcionamiento.

Al poner sensores en cada estacionamiento que reemplacen el trabajo de la comunidad estamos asegurando que el 100% de los recursos aporten para un funcionamiento efectivo del sistema. Se requiere además que la información que generen los sensores sea subida de alguna manera a la nube de Internet para que esté disponible desde cualquier lugar y en cualquier momento. Adicional almacenar la información en una base de datos para llevar estadísticas del comportamiento en el transcurso del tiempo.

1.2. Marco Teórico

1.2.1 Sensor Ultrasónico

Figura 1.1. Sensor de distancia ultrasónico

Fuente: (Cosas de Ingeniería, 2015)

Los sensores ultrasónicos miden la distancia mediante el uso de ondas ultrasónicas en donde el cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna desde el objeto. Los sensores ultrasónicos miden la distancia al objeto contando el tiempo entre la emisión y la recepción. Un sensor óptico tiene un transmisor y receptor, mientras que un sensor ultrasónico utiliza un elemento ultrasónico único, tanto para la emisión como la recepción. (Keyence, 2018)

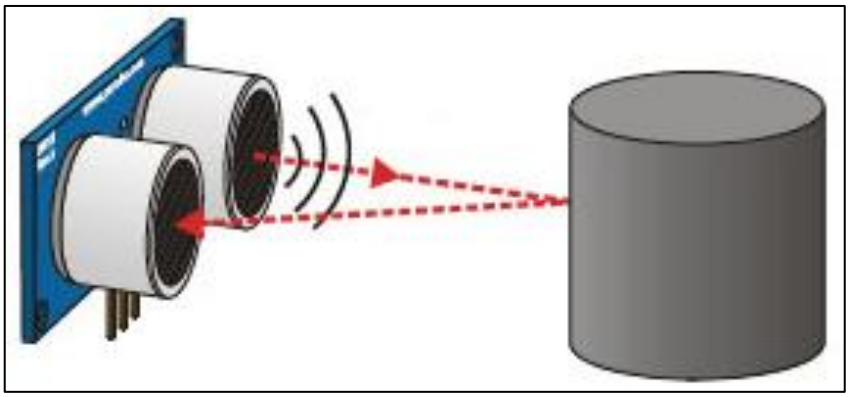


Figura 1.2. Modo de funcionamiento del Sensor Ultrasónico

Fuente: (CursoArduino, 2016)

1.2.2. Transmisor de Radio Frecuencia

Es un módulo de Radiofrecuencia a 433MHz que transmite datos en UHF y que sirve para montaje en circuitos PCB. Cuando trabaja con el receptor de 433MHz que es su complemento, conforman un sistema tx/rx, que facilita la implementación de enlaces de datos de radiofrecuencia, alcanzando distancias de hasta 80 metros dentro de edificaciones o 350 metros en campo abierto cuando opera con la fuente de 12V. (tecmikro, 2018)

Especificaciones técnicas transmisor de Radio Frecuencia:

- Señal de radiofrecuencia: Modulación ASK (Modulación por Desplazamiento de Amplitud)
- Fuente de alimentación: 12V (también disponible en versiones de 3V y 5V)
- Consumo de corriente: <16 mA
- Potencia de transmisión: 13 dBm
- Desviación de frecuencia: +- 75kHz
- Alcance útil hasta 350 metros (12V), 230 metros (5V), 160 metros (3V)
- Disponible en frecuencias de 433.92 MHz (433MHz) y 315.0 MHz
- Velocidades de transmisión hasta 20kbps (Dualtronica, 2018)

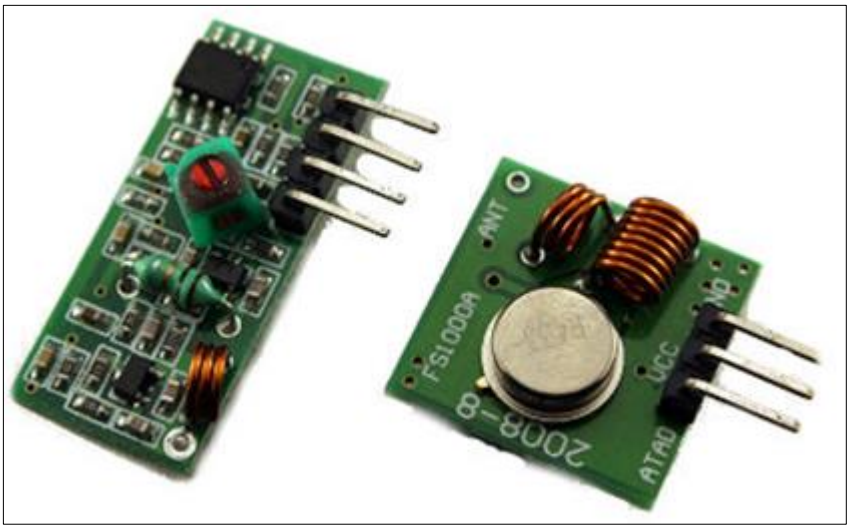


Figura 1.3. Módulo Tx y Rx de Radiofrecuencia

Fuente: (Etools, 2016)

1.2.3. Receptor de Radio Frecuencia

Este es un receptor de datos en UHF, para montaje en circuito impreso (PCB). Con el transmisor correspondiente (13dBm), permite implementar enlaces TX/RX inalámbricos

de datos a velocidades de hasta 4.8kbps con distancias de hasta 40 metros en interiores o 110 metros en campo abierto. (Dualtronica, 2018)

Características

- Velocidades de hasta 4.8kbps
- Alcance utilizable de hasta 110 metros
- Versiones disponibles en 433.92 MHz (433MHz) y 315.0MHz
  - Versiones disponibles: regulado y no regulado
  - Rápido tiempo de establecimiento de datos
  - Consumo de corriente: 2.2mA (Dualtronica, 2018)

1.2.4. Microcontrolador ATMEGA328

Es un microcontrolador RISC de 8 bits de alto rendimiento con memoria flash de 32 Kb capacidad de lectura y escritura, memoria EEPROM de 1 KB, SRAM de 2 KB, 23 líneas I/O de propósito general, 32 registros de propósito general, tres timers/counters con modos de comparación, interrupciones internas y externas, conversor A/D de 6 canales y 10 bits, watch dog programable con oscilador interno y cinco modos de ahorro de energía seleccionables por software. El dispositivo funciona entre 1.8 y 5.5 Voltios. (Geekfactory, 2018)

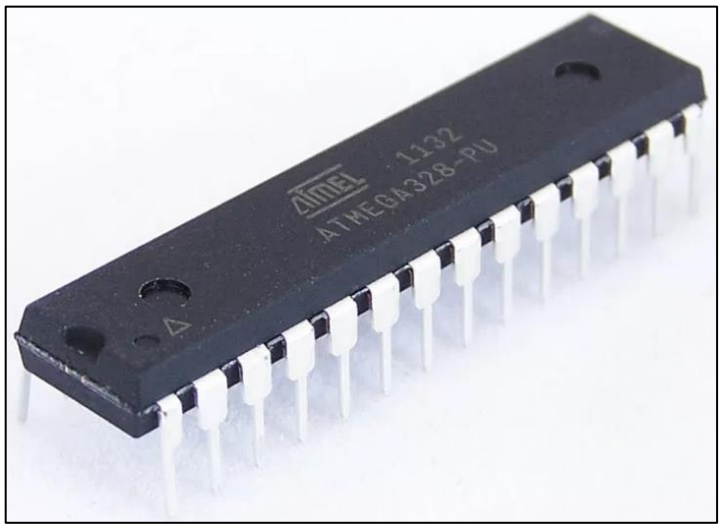


Figura 1.4. Microcontrolador ATMEGA328

Fuente: (Microchip, 2016)

1.2.5. Módulo wi-fi ESP8266

Este módulo es cada vez más utilizado en aplicaciones IoT debido a su bajo coste y fácil conexión a una red local o salida a la nube de Internet. Esto debido a que tiene integrado el stack de protocolos TCP/IP para la conexión fuera de la red local.
También incorpora el protocolo 802.11 b/g/n para la conexión inalámbrica local, eliminando además con esto, el problema de la conexión física por cable para otros dispositivos como por ejemplo placas Arduino. La conexión con un punto de acceso wi-fi se lleva a cabo mediante comandos de texto AT a través del puerto serie y una vez establecida, se comunicará con él. (Punto flotante, 2018)

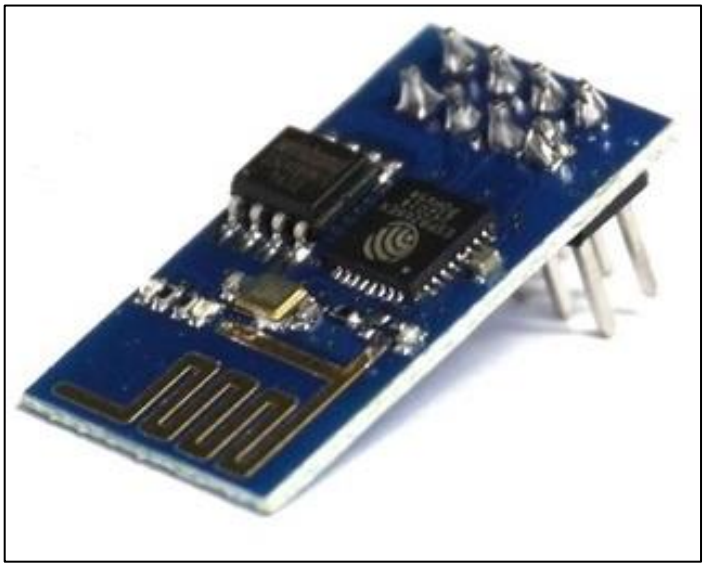


Figura 1.5. Módulo Wi-Fi ESP8266

Fuente: (Naylamp Mechatronics, 2016)

Entre las principales características se tiene:

- Trabaja con el protocolo 802.11 b/g/n
- Integra stack de protocolos TCP/IP
- 64KBytes de RAM de instrucciones
- 96KBytes de RAM de datos
- Alimentación 3.3 VDC (Naylamp Mechatronics, 2016)

1.2.6. Regulador de voltaje LM7805

Es un dispositivo electrónico que regula el voltaje positivo de 5V a 1A de corriente. En desarrollo de PCBs o tarjetas electrónicas con programadores Pic u otro tipo de microcontroladores, se está obligado a garantizar una fuente de tensión constante, eso disminuye la posibilidad de dañar la tarjeta de un circuito debido a oscilaciones en los niveles de tensión, la forma más práctica y simple de lograr esto es mediante el Regulador de voltaje 7805. El dispositivo cuenta con 3 pines y para la nomenclatura de las diferentes series; las primeras letras y dos números corresponden a la denominación, mientras que las dos últimas XX deben ser reemplazados por la tensión de salida requerida. (Veloso, 2016).

- Tensión de entrada
- Masa
- Tensión de salida (Veloso, 2016)

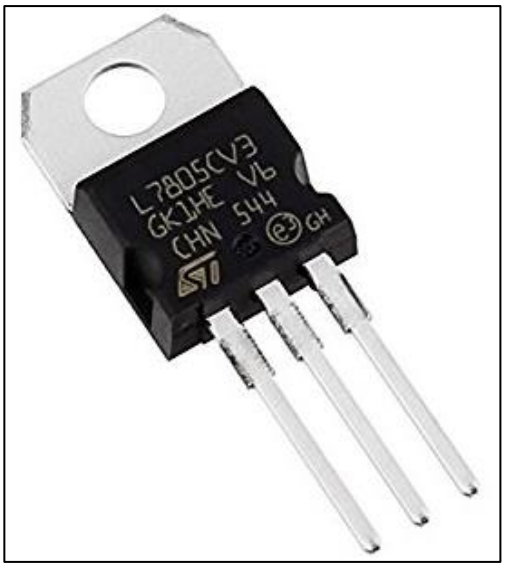


Figura 1.6. Regulador de voltaje LM7805

Fuente: (Eika Electronic, 2018)

1.2.7. IoT Internet of Things

El internet de las cosas optimiza dispositivos y sistemas que antes se conectaban mediante circuito cerrado como cámaras, sensores y les permite comunicarse globalmente a través de

las redes. Se podría decir que es una red que conecta objetos físicos por medio del uso del internet y que cuentan con algún tipo de inteligencia que realice eventos específicos en función de las tareas que sean requeridas remotamente. (Torres, 2014)

Los sistemas conectados a Internet de las cosas generalmente constan de 4 elementos principales:

- Sensor/actuador
- Dispositivo de comunicación
- Microcontrolador
- Fuente de alimentación (Vega, 2015)

Los sistemas embebidos dentro del campo del Internet de las cosas, a más de ser pequeños computadores empotrados en algo también basan su esencia en la comunicación con la red.

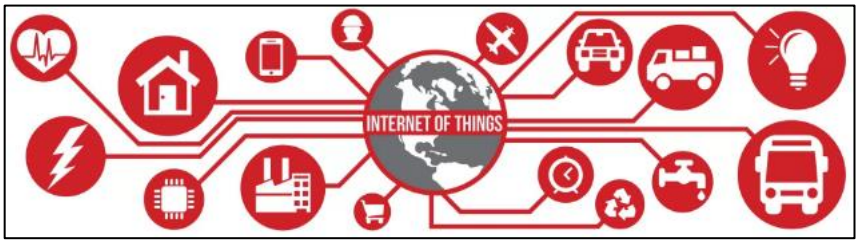


Figura 1.7. Internet of things

Fuente: (Hipertextual, 2014)

1.2.8. API's Interfaces de Programación de Aplicaciones

Son plataformas de código abierto cuyo objetivo es recoger, analizar, visualizar y manipular datos provenientes de la red. Ejemplos de este tipo de APIs serían: ThingSpeak, Xively, Cayenne, NodeRed.

Para el caso de ThingSpeak, se trata de una plataforma abierta para Internet de las Cosas que recopila, almacena, analiza, visualiza y actúa sobre la información recogida por sensores y dispositivos como hardware de código abierto como Arduino, Raspberry Pi o BeagleBone. ThingSpeak es una API conocida entre desarrolladores y dispone ya de una gran comunidad. (BBVA, 2016).

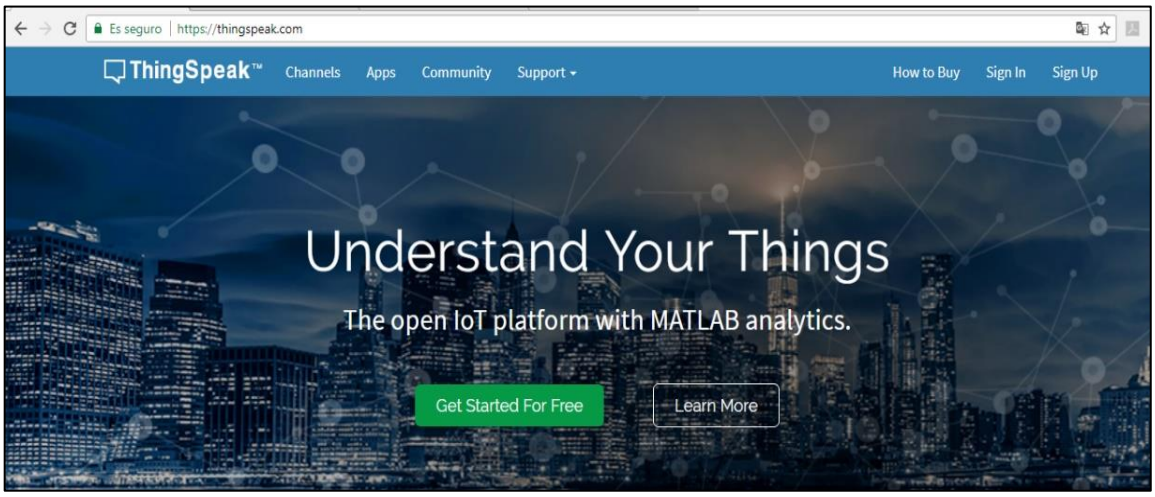


Figura 1.8. Plataforma ThingSpeak

Fuente: (ThingSpeak, 2018)

ThingSpeak API trabaja en base a canales los cuales contienen los campos de datos, ubicación y estado. (BBVA, 2016).

1.3. Fundamento teórico

1.3.1. Compartir la información de la red local con la nube de Internet

Un punto importante será buscar encontrar la mejor manera de subir información generada de la red local hacia la nube de internet para que se encuentre disponible a cualquier hora desde cualquier lugar. Existen varias maneras para conseguir esto; como son el módulo Raspberry, el módulo NodeMCU, módulos ethernet y Shield WiFi de Arduino, etc

.

El módulo NodeMCU tiene incorporado el microcontrolador ESP8266 el cual permite la conectividad inalámbrica a través del protocolo 802.11 y a la vez compatible con IPv4 ya que tiene incorporados los protocolos TCP/UDP/HTTP/FTP. Esto hace que con ayuda de un router pueda tener conectividad hacia la nube de internet.

El microcontrolador ESP8266 es el más económico y muy flexible en cuanto su programación ya que se puede descargar de firmware que permite programar en lenguajes como LUA, Python, Basic o JavaScript o inclusive es compatible con IDE de Arduino.

1.3.2. Plataforma IoT

Existen plataformas que son utilizadas en proyectos de Internet de las cosas como son: ThingSpeak (MathWorks Lab), Cayenne, Xively, Node-red (IBM), etc. Con algunas ventajas y limitantes propias de cada plataforma. También resulta importante determinar la plataforma IoT más óptima que también será clave para cumplir los objetivos propuestos en este documento.

Por ejemplo, ThingSpeak es una plataforma de Internet de las cosas de Código abierto la cual resulta amigable para conectar dispositivos Arduino, Raspberry Pi LoRa Wan. Permite llevar registros y bases de datos de sensores en HTTP, a través de Internet o en la misma red de área local. También permite la creación de aplicaciones de registro de dispositivos como sensores, aplicaciones de localización de seguimiento y una red social de cosas con frecuentes actualizaciones de estado. Al ser una plataforma abierta resulta económica y amigable para su programación e interpretación en su presentación de la información recolectada.

CAPÍTULO 2

PROPUESTA

De forma general se propone para el proyecto las siguientes etapas descritas a continuación:

- Tarjeta Sensor Transmisor (Tx)
- Tarjeta Receptor (Rx)
- Tarjeta de conexión a la nube de Internet
- Recopilación de Información en ThingSpeak
- Aplicación en MIT App Inventor

La tarjeta Sensor Transmisor tiene como componente el sensor Ultrasónico HC-SR04 que se encarga de detectar cuando hay o no presencia de un vehículo en el lugar de estacionamiento. Estos cambios de estado son procesados por el microcontrolador ATMEGA328 que pertenece a la misma tarjeta, acondicionando la señal con un 1 para el caso de Ocupado y un 0 para el caso de Libre. Posteriormente esta información es presentada en su puerto serial que está conectado con el módulo transmisor RF. Este a su vez hará el envío de la información inalámbricamente hasta la tarjeta Receptor Rx.

En la tarjeta Receptor Rx se recibe la información de todos los nodos, la cual es tomada por el microcontrolador ATMEGA328 perteneciente a esta tarjeta. Este realiza el análisis respectivo y en base a los resultados obtenidos, envía las instrucciones al Módulo de Conexión a Internet ESP8266 por medio de su puerto serial para que sea subido a la nube de Internet a través de su antena Wi-Fi y su stack de protocolos TCP-IP que vienen incorporados en el mismo.

El Módulo de Conexión a Internet ESP8266, envía la información hasta el servidor IoT ThingSpeak en la nube de Internet por medio de su conexión wi-fi y a través del stack de protocolos TCP/IP que tiene cargado en su microprocesador para su funcionamiento.

Para la recopilación de Información en ThingSpeak, el servidor permite recoger los datos provenientes de los Sensores Transmisores y ser configurados de forma gráfica para que cada Sensor tenga su respectivo diagrama donde se puede apreciar los cambios de estado de 1 a 0 que representan al estacionamiento como Libre u Ocupado, en el transcurso del tiempo.

Una vez que la información ya se encuentra disponible en internet, se ejecuta la aplicación para dispositivos móviles con sistema operativo Android que disponga de Datos para su conexión a Internet y desde el mismo se puede conocer el status del estacionamiento (Ocupado/Libre) o a su vez, ejecutar acciones como reservar o liberar una determinada plaza de estacionamiento.

El siguiente es un esquema donde se representa básicamente las tarjetas y etapas mencionadas en la descripción general anterior:

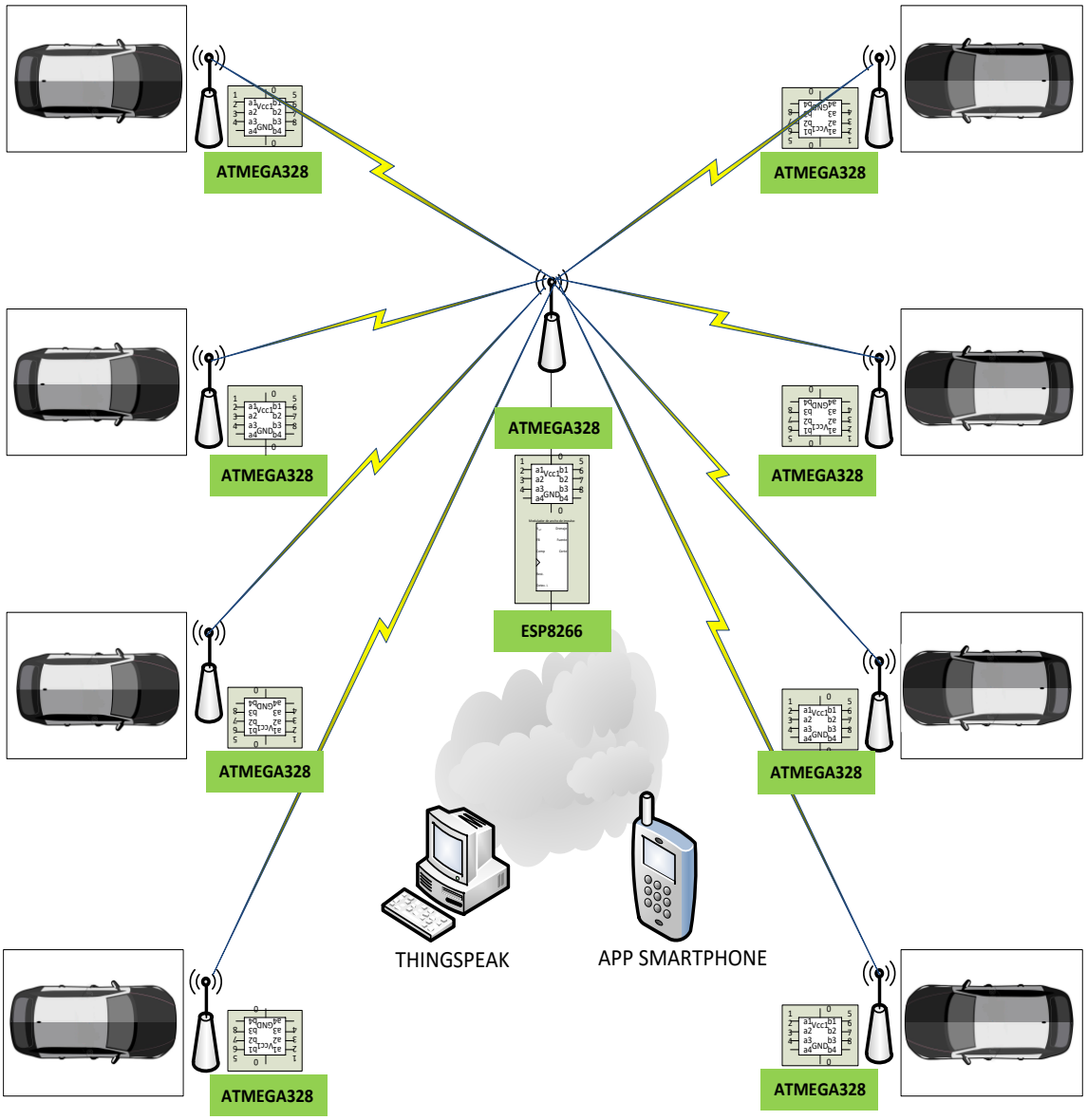


Figura 2.1. Diagrama esquemático de la red de sensores inalámbricos propuesta

Fuente: Elaborado por el autor

2.1. Propuesta de la Tarjeta Sensor Transmisor (Tx)

2.1.1. Etapa de alimentación de la tarjeta Sensor Transmisor (Tx)

Para la alimentación de la tarjeta se va a utilizar el regulador de voltaje LM7805 para acondicionar o limitar el excedente de voltaje que llegue de la fuente de alimentación hacia su microcontrolador, el mismo que necesita para su operación 5VDC.

Las características del regulador LM7805 se puede apreciar que cubre este requerimiento en el anexo B que corrresponde al Data Sheet de este dispositivo. La tabla lleva por nombre Electrical Characteristics (MC7805/LM7805)

Fuente: (All Data Sheet, 2018)

2.1.2. Desarrollo del monitoreo con Sensor de Ultra sonido HC-SR04

Se requiere detectar la presencia de un vehículo cuando llegue a ocupar uno de los estacionamientos, para lo cual se utilizará el sensor Ultrasónico HC-SR04, debido a sus características para medir distancias hasta un objeto. El resultado del cambio de estado del sensor, será enviado hasta un microcontrolador que estará en la misma tarjeta de sensores.

Su implementación en cuanto a la alimentación resulta sencilla, ya que el sensor ultrasónico opera con alimentación de 5VDC lo cual lo hace compatible con la tarjeta Sensor Transmisor cuya alimentación es de 5VDC. Se lo debe regular a una distancia aproximada de 2mts a partir del techo del estacionamiento hasta el techo del auto.

A continuación, los parámetros técnicos del dispositivo propuesto:

Tabla 2.1. Características técnicas Sensor Ultrasónico HC-SR04

Working Voltage	Voltaje de trabajo	DC 5V
Working Current	Corriente de trabajo	15 mA
Working Frequency	Frecuencia de trabajo	40 Hz
Max Range	Distancia máxima	4 m
Min Range	Distancia mínima	2 cm
Measuring Angle	Angulo eficaz	15 degree
Trigger Input Signal	Disparo de señal de entrada	10 us TTL pulse
Echo Output Signal	Eco señal de salida	Input TTL lever signal and the range in proportion
Dimension	Dimensiones	45*20*15 mm

Fuente: (Robotik Sistem, 2009)

Sus pines de conexión son los siguientes:

- Vcc = 5VDC
- Trig = Disparo de ultrasonido

- Echo = Recepción de ultrasonido
- GND = Tierra

Este dispositivo calcula la distancia en base a la siguiente fórmula matemática:

$$L = \frac{(Tiempo\ entre\ el\ Trig\ y\ el\ Echo) * (Velocidad\ del\ sonido)}{2}$$

Fórmula 2.1. Funcionamiento matemático sensor ultrasónico HCSR04

Fuente: (Llamas L, 2015)

y resumida de forma matemática resulta:

$$\text{Distancia } L = \frac{1}{2} * T * C$$

Donde: L= distancia,

T= tiempo entre la emisión y la recepción,

C= Velocidad del sonido.

2.1.3. Microcontrolador ATMEGA328

El microcontrolador Atmega328 es compatible con IDE de Arduino. Esto hace que resulte económico ya que el hardware y software para su programación son de código abierto. En la web existe disponibilidad de información para hacer uso de librerías que facilita la programación. Estos son puntos a favor para elegir este microcontrolador.

La información que va a manejar este microcontrolador en el Módulo Sensor Transmisor, son los cambios de estado del sensor del estacionamiento, es decir; libre/ocupado (1/0), lo que hace que un microcontrolador de 8 bits sea suficiente para la carga de procesamiento y a la vez reducir el consumo de energía.

Sus características técnicas hacen que resulte útil para el tratamiento de la señal ya que tiene la capacidad de ejecutar instrucciones mientras equilibra el consumo de energía y la velocidad de procesamiento. Generalmente son usados en áreas de automatización industrial, de viviendas y edificios que se asemeja bastante al presente proyecto.

Los pines que tiene este microcontrolador son los siguientes:

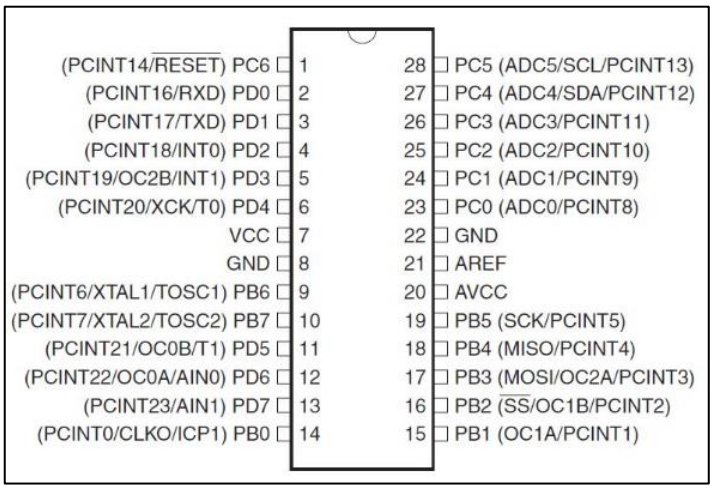


Figura 2.2. Distribución de pines Integrado Atmega328

Fuente: (Slide Share,2014)

Tabla 2.2. Especificaciones técnicas microcontrolador Atmega328

Voltaje de operación	5V
Número de Pines:	28
Memoria FLASH	32KB
Memoria RAM	2KB
EEPROM	1KB
Máxima frecuencia de funcionamiento	20Mhz
CPU	8-bit AVR
Pines de entrada/salida	23
Entradas Analógicas (ADC)	6
Formato DIP	

Fuente: (Naylamp Mechatronics, 2018)

También resultará útil para las pruebas en protoboard cuando se requiera, ya que su encapsulado PDIP permite ser montado directamente sobre el mismo.

2.1.4. Transmisor RF de Radiofrecuencia

Para la comunicación de los sensores con el nodo receptor se utilizarán módulos de radio frecuencia Tx/Rx 433Mhz cuya frecuencia es de banda libre. Resulta sencillo implementar enlaces de datos de radiofrecuencia con estos módulos ya que permiten alcanzar distancias de hasta 80 mts dentro de estructuras cerradas o 350 mts en lugares abiertos cuando opera con la fuente de 12V. Adicional el diseño de su placa sirve también para ser montado en un

circuito impreso PCB. Estas características lo hacen muy popular, económico y adecuados para el presente proyecto.

Especificaciones técnicas Transmisor:

Tabla 2.3. Transmisor RF de Radiofrecuencia

Señal de radiofrecuencia	Modulación ASK
Fuente de alimentación	12V, 5V, 3V
Consumo de corriente	<16 mA
Potencia de transmisión	13dBm
Desviación de frecuencia	+ - 75kHz
Alcance	230 metros (5V)
Frecuencia	433MHz y 315 MHz
Velocidades de Transmisión	Hasta 20kbps

Fuente: (Tecmikro, 2018)

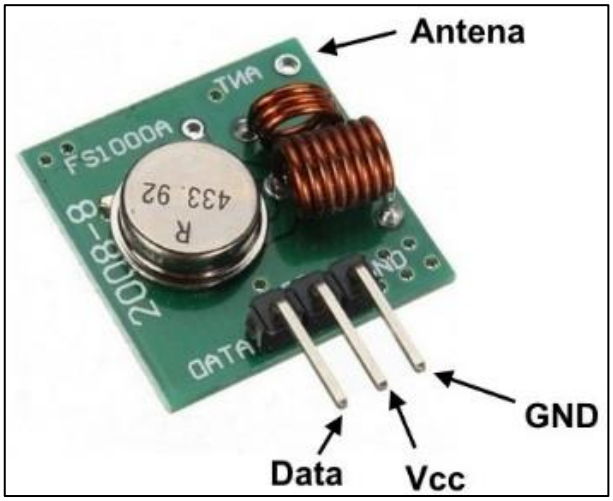


Figura 2.3. Transmisor RF de Radiofrecuencia 433Mhz

Fuente: (Luis Llamas, 2016)

El transmisor es compatible con aplicaciones inalámbricas de enlaces de datos uno-a-uno o de varios nodos.

2.1.5. Diagrama esquemático de la Tarjeta Sensor Transmisor (Tx)

A continuación, se muestra de forma esquemática cómo queda conformada la tarjeta con los dispositivos propuestos en los ítems anteriores y sus conexiones.

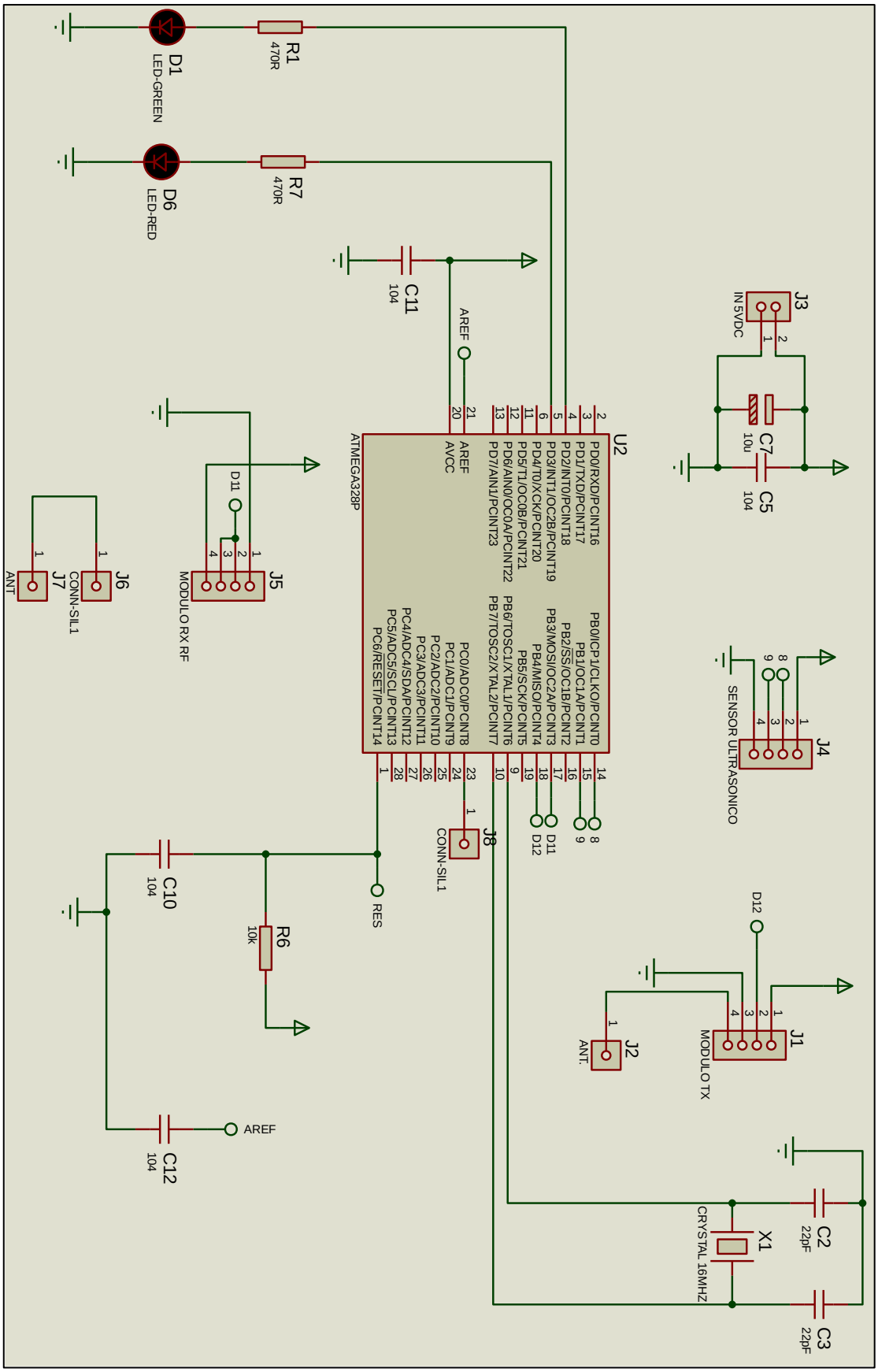


Figura 2.4. Diagrama esquemático de la Tarjeta Sensor Transmisor Tx

Fuente: Elaborado por el autor

2.1.6. Diseño de la placa PCB de la Tarjeta Sensor Transmisor (Tx)

Este diagrama se realiza en el módulo ARES del software Proteus 7 Professional que sirve para la fabricación de placas de circuito impreso ya que facilita la edición, ubicación, ruteo de pistas de cobre.

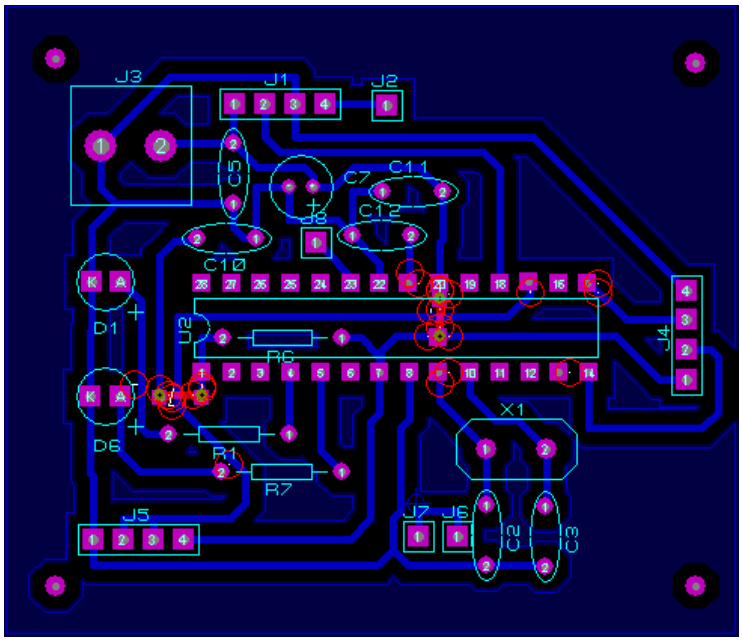


Figura 2.5. Diagrama PCB de la tarjeta Sensor Transmisor Tx

Fuente: Elaborado por el autor

2.2. Propuesta de la Tarjeta Receptor (Rx)

Para esta tarjeta, a más de otros elementos que se mencionará más adelante, se empleará también el microcontrolador ATMEGA328 que ya fue mencionado y analizado por lo que ya no es necesario detallarlos nuevamente.

Los demás elementos se indican a continuación:

2.2.1. Etapa de alimentación de la Tarjeta Receptor (Rx)

Para el desarrollo de esta tarjeta se va a utilizar adicionalmente el regulador de voltaje LM1117, el cual se va a encargar de limitar o absorber todo el excedente de voltaje que llegue de la fuente de alimentación hacia el módulo wi-fi ESP8266 (detallado más adelante), ya que el mismo necesita para su operación 3 a 3,6VDC. Con esto se consigue tener una

tensión constante de 3VDC que reducirá las posibilidades de quemar los circuitos de esta tarjeta debido a oscilaciones en los niveles de tensión.

A continuación, se muestra las características del regulador LM1117 en el que se puede apreciar que cubre este requerimiento:

Tabla 2.4. Características técnicas del regulador LM1117 (Anexo D)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V out	Output Voltage	LM1117-3.3 Iout = 10mA, Vin=5V Tj=25°C	3.267	3.300	3.333	V
		0 ≤ I out ≥ 800 mA, 4.75V ≤ Vin ≥ 10V	3.235	3.300	3.365	V

Fuente: (LM117 Data Sheet, 2006)

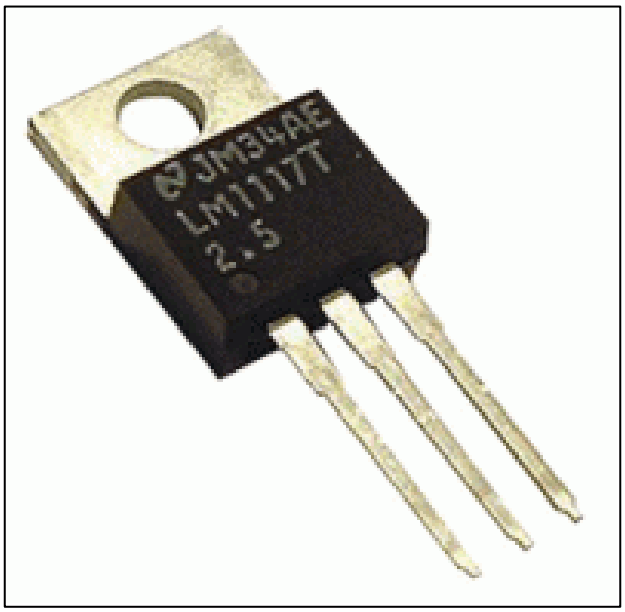


Figura 2.6. Regulador de voltaje LM1117

Fuente: (Micro JPM, 2018)

2.2.2. Etapa de recepción con módulos RF de Radiofrecuencia

Este dispositivo forma parte de la comunicación entre el Receptor y los sensores de tal manera que se pueda recibir los cambios de estado producidos en cada puesto de estacionamiento y enviados por los transmisores.

Los módulos de Radiofrecuencia vienen como pareja tanto Transmisor como Receptor y su frecuencia de operación igual es de 433Mhz de banda libre. Los módulos receptores pueden llegar a velocidades de hasta 4.8kbps y distancias de 40 mts dentro de edificios o 110 mts en campo abierto. Adicional el diseño de su placa sirve también para ser montado en un circuito impreso PCB.

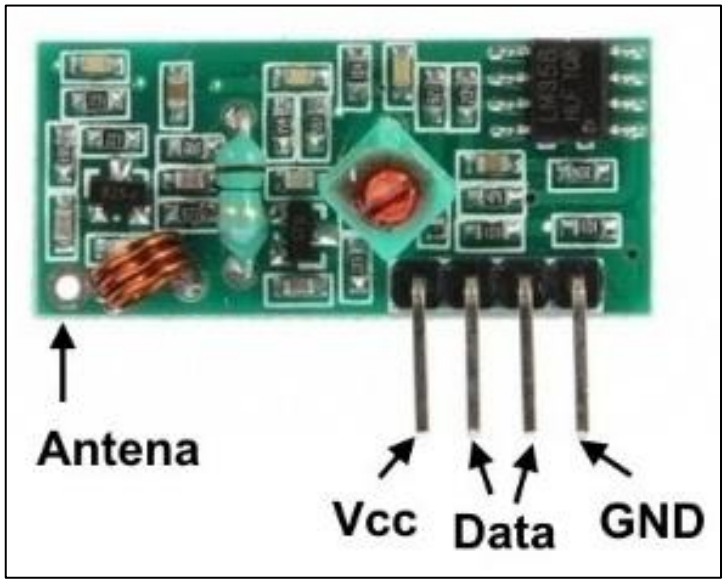


Figura 2.7. Receptor RF de Radiofrecuencia 433Mhz

Fuente: (Luis Llamas, 2016)

Especificaciones técnicas Receptor:

Tabla 2.5. Receptor RF de Radiofrecuencia

Fuente de alimentación	5V
Consumo de corriente	2.2 mA
Velocidades	Hasta 4.8kbps
Alcance utilizable	Hasta 110 mts
Frecuencia	433.92 MHz (433MHz) y 315.0MHz
Versiones disponibles	regulado y no regulado

Fuente: (Tecmikro, 2018)

2.2.3. Etapa de Conexión a la nube de Internet con el Módulo wi-fi ESP8266

El módulo ESP8266 es ideal para esta parte de la comunicación ya que, entre sus principales virtudes, está la de tener comunicación integrada 802.11 b/g/n que permite la salida a Internet a través de un proveedor de servicios ISP. Otra gran ventaja es que tiene incorporado la pila de protocolos TCP/IP lo que hace que se libere la mayor parte del trabajo de comunicación del procesador.

El voltaje de alimentación necesario es de 3.3VDC y el consumo de corriente dependiendo de la exigencia puede ir de 50mA a 200mA (al arrancar el dispositivo). Este voltaje será provisto por el Regulador de voltaje LM1117 el cual ya fue detallado anteriormente y estará presente en la misma placa del Receptor. En la siguiente tabla se muestran las especificaciones técnicas módulo ESP8266:

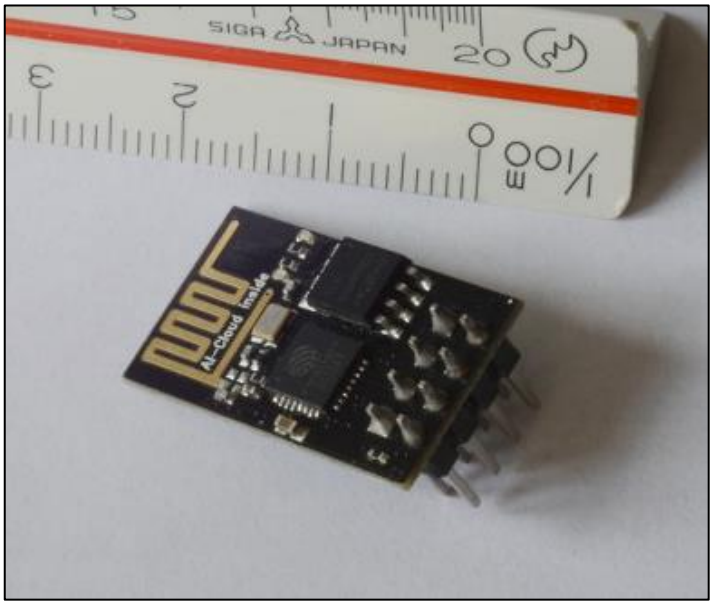


Figura 2.8. Módulo wi-fi ESP8266

Fuente: (Polaridad.es, 2016)

Tabla 2.6. Especificaciones técnicas módulo ESP8266

Parameters	Conditions	Min	Typical	Max	Unit
Storage Temperature Range		-40	Normal	125	°C
Maximun Soldering Temperature	IPC/JEDEC J-STD-020			260	°C
Working Voltage Value		3.0	3.3	3.6	V
I max				12	mA
Electrostatic Discharge (HBM)	TAMB=25°C			2	KV
Electrostatic Discharge (CDM)	TAMB=25°C			0.5	KV

Fuente: (ESP8266EX Datasheet, 2015 ANEXO A)

Tabla 2.7. Especificaciones técnicas módulo ESP8266

Protocolos soportados: 802.11 b/g/n
Wi-Fi Direct (P2p), Soft Access Point
Stack TCP/IP integrado
Potencia de salida: +19.5dBm en modo 802.11b
Sensor de temperatura integrado
Consumo en modo de baja energía: <10 Ua
64KBytes de RAM de instrucciones
96KBytes de RAM de datos
Integra núcleo de arquitectura RISC 32bits corre a 80Mhz

Fuente: (Geek Factory, 2015)

El microcontrolador ESP8266 resulta conveniente para este proyecto también debido a que es más económico y muy flexible para su programación, ya que se puede descargar algún firmware que permite programar por ejemplo en IDE de Arduino; que es precisamente el que se utilizará para poder acceder a su configuración por comandos AT.

2.2.4. Diagrama esquemático Tarjeta Receptor (Rx)

A continuación, se muestra cómo queda conformada la tarjeta con los dispositivos propuestos en los ítems anteriores y sus conexiones de forma esquemática:

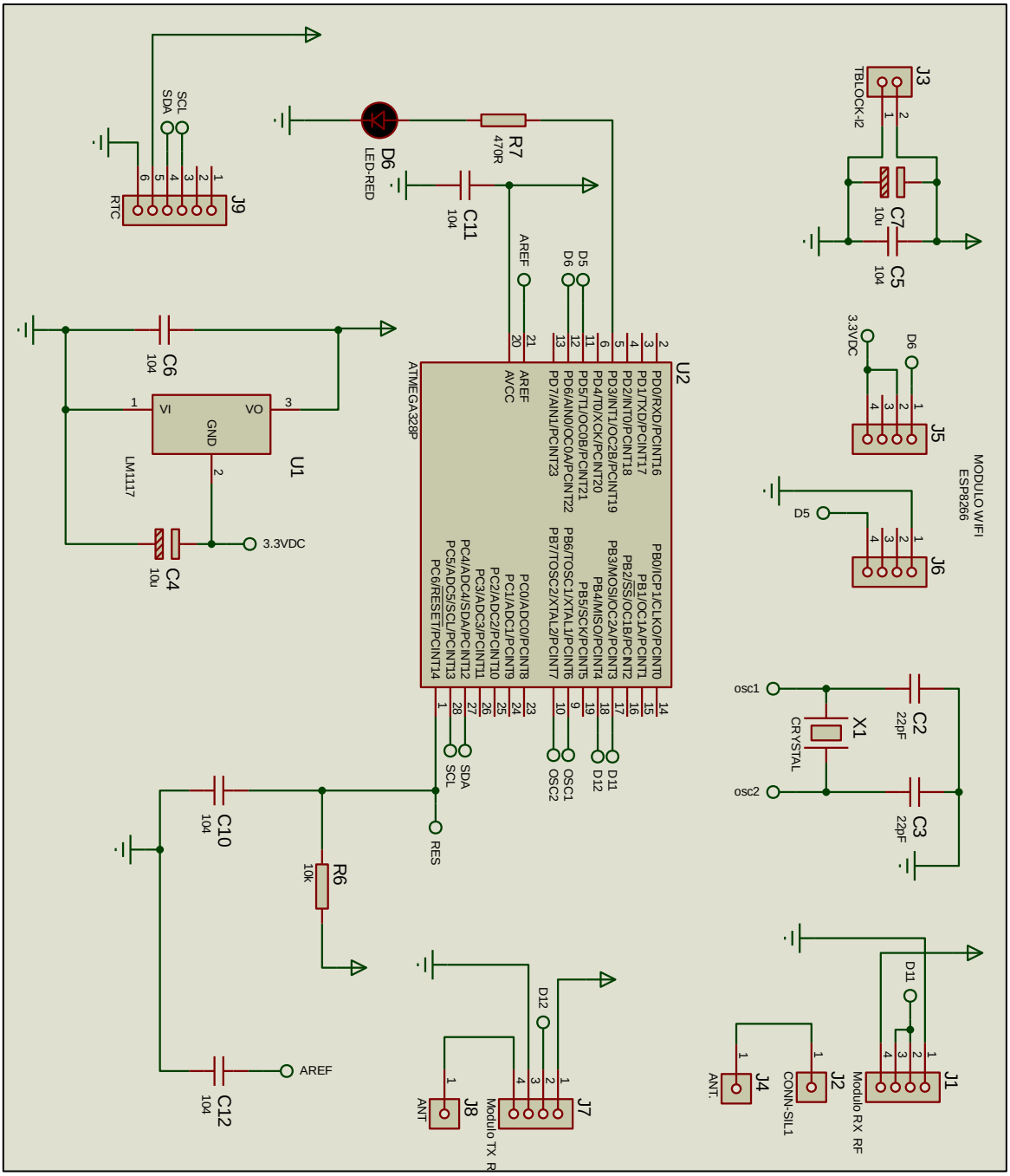


Figura 2.9. Diagrama esquemático de la Tarjeta Receptor (Rx)

Fuente: Elaborado por el autor

2.2.5. Diseño de la placa PCB de la Tarjeta Receptor (Rx)

De igual manera éste diagrama se realiza en el módulo ARES del software Proteus 7 Professional.

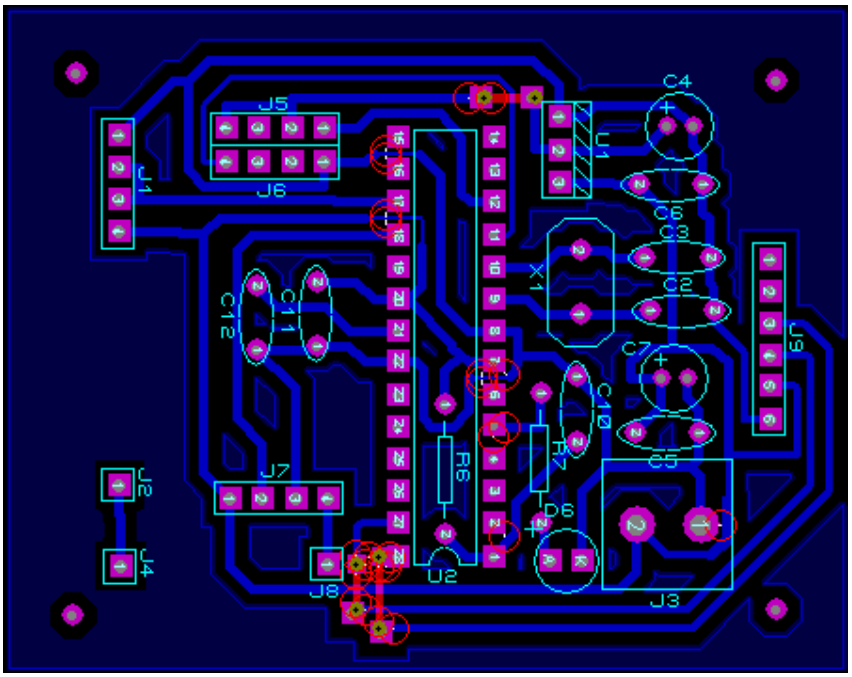


Figura 2.10. Placa PCB de la Tarjeta Receptor (Rx)

Fuente: Elaborado por el autor

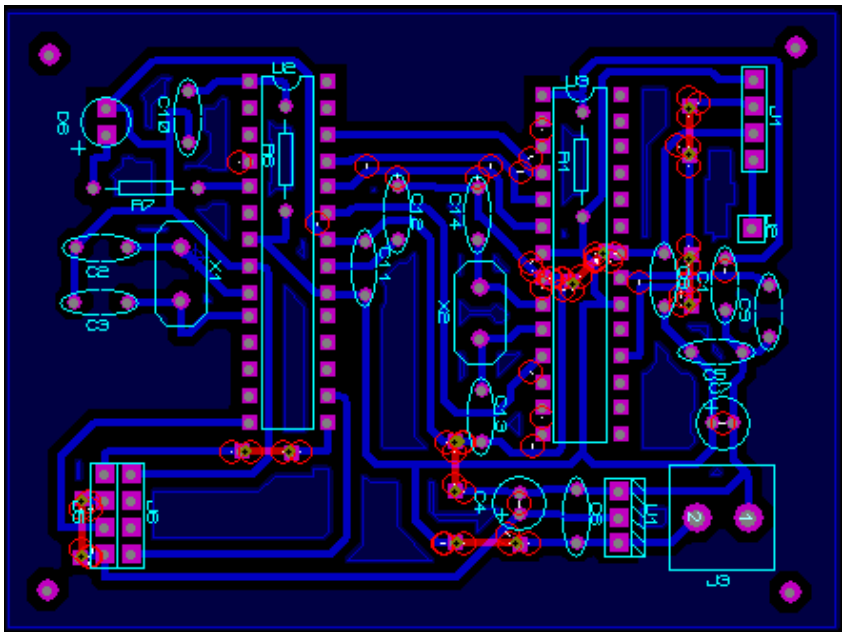


Figura 2.11. Placa PCB de Control de Estados

Fuente: Elaborado por el autor

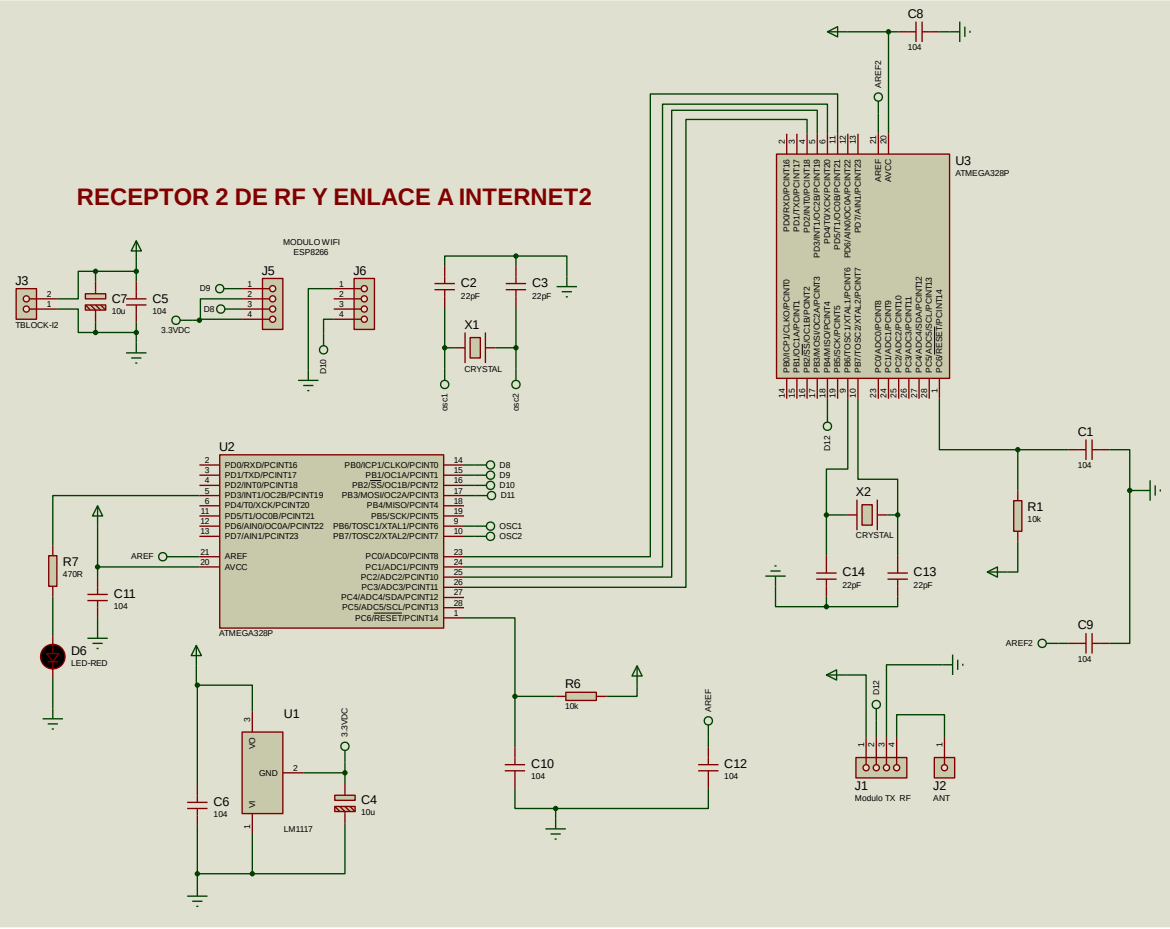


Figura 2.12. Diagrama esquemático del Control de Estados

Fuente: Elaborado por el autor

2.2.6. Programación en IDE (Entorno de Desarrollo Integrado de Arduino)

Para el desarrollo del programa del Atmega328 se va a utilizar el Entorno de Desarrollo Integrado IDE de Arduino, el mismo que se lo puede descargar gratuitamente de la página oficial en el siguiente link *https://www.arduino.cc/en/Main/Software* y se encuentra disponible para las diferentes versiones de sistemas operativos Windows, Mac OS y Linux.

Para establecer la comunicación entre los microcontroladores ATMEGA328 de las tarjetas Sensor Tx y Receptor Rx a nivel de software se va a utilizar la librería VirtualWire ya que se trata de un firmware de código abierto que lo hace ideal para incluirlo en lenguaje IDE de Arduino.

También lo podremos utilizar para configurar los demás dispositivos como son el ESP8266 y el sensor ultrasónico HC-SR04 con la diferencia que son líneas de comandos AT y a través del monitor serie.

El software de programación está basado en lenguaje C/C++ cuyo entorno es amigable, liviano y con las herramientas básicas para cargar, depurar y conectarse con la tarjeta diseñada para nuestro propósito.

A continuación, la interfaz gráfica del Entorno de Desarrollo Integrado IDE Arduino:

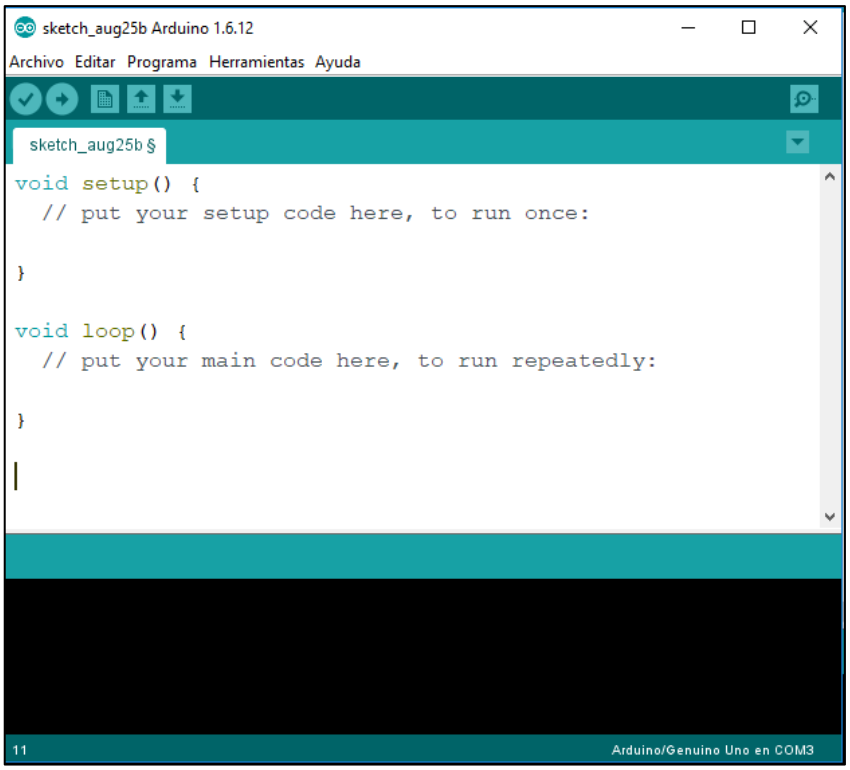
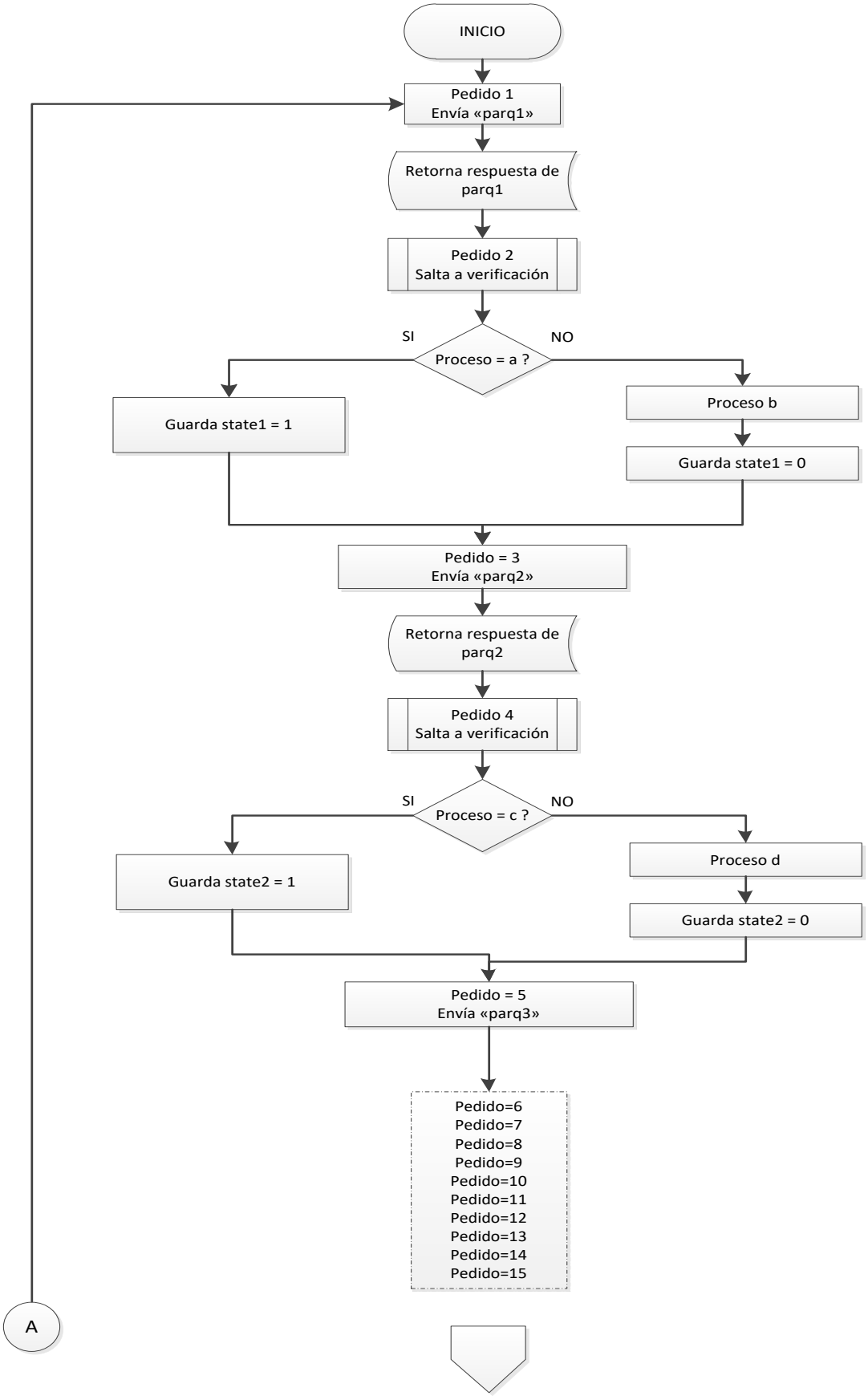


Figura 2.13. Entorno de Desarrollo Integrado IDE Arduino

Fuente: (Arduino, 2018)

2.2.7. Diagrama de flujo del programa de la Tarjeta Receptor (Rx)

El microcontrolador ATMEGA328 de esta tarjeta inicia pidiendo la información del status de cada estacionamiento, recopilar esta información y a su vez comunicarla hacia el módulo ESP8266 para su envío hacia el servidor ThingSpeak. El flujo de los procesos es el siguiente:


INICIO
Pedido 1
Envía «parq1»
Retorna respuesta de parq1
Pedido 2
Salta a verificación
SI
NO
Proceso = a ?
Guarda state1 = 1
Proceso b
Guarda state1 = 0
Pedido = 3
Envía «parq2»
Retorna respuesta de parq2
Pedido 4
Salta a verificación
SI
NO
Proceso = c ?
Guarda state2 = 1
Proceso d
Guarda state2 = 0
Pedido = 5
Envía «parq3»
Pedido=6
Pedido=7
Pedido=8
Pedido=9
Pedido=10
Pedido=11
Pedido=12
Pedido=13
Pedido=14
Pedido=15
A

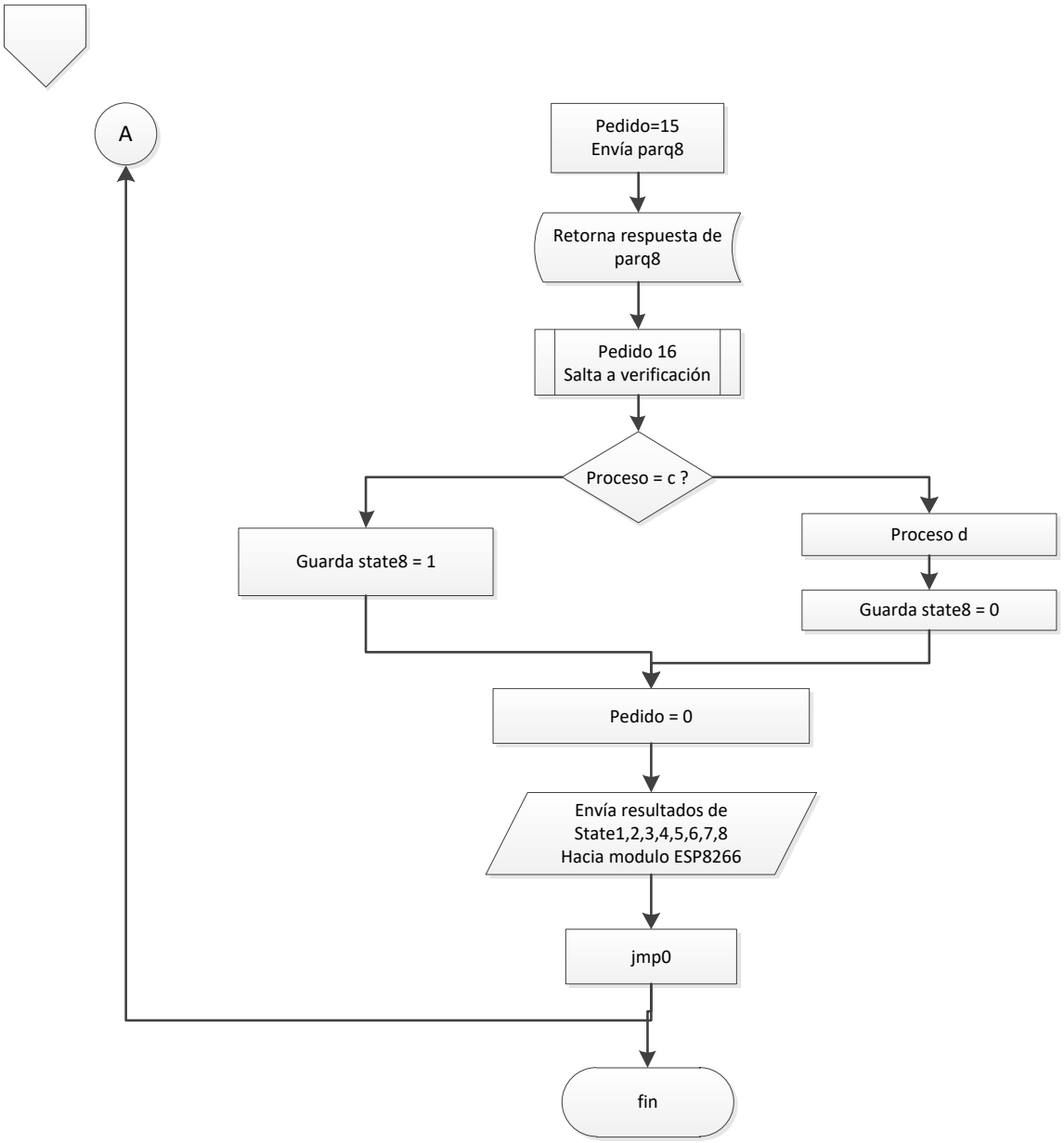


Figura 2.14. Diagrama de flujo del programa de la Tarjeta Receptor (Rx)

Fuente: Elaborado por el autor

2.2.8. Diagrama de flujo del programa de la Tarjeta Sensor Transmisor (Tx)

El flujo de los procesos es el siguiente:

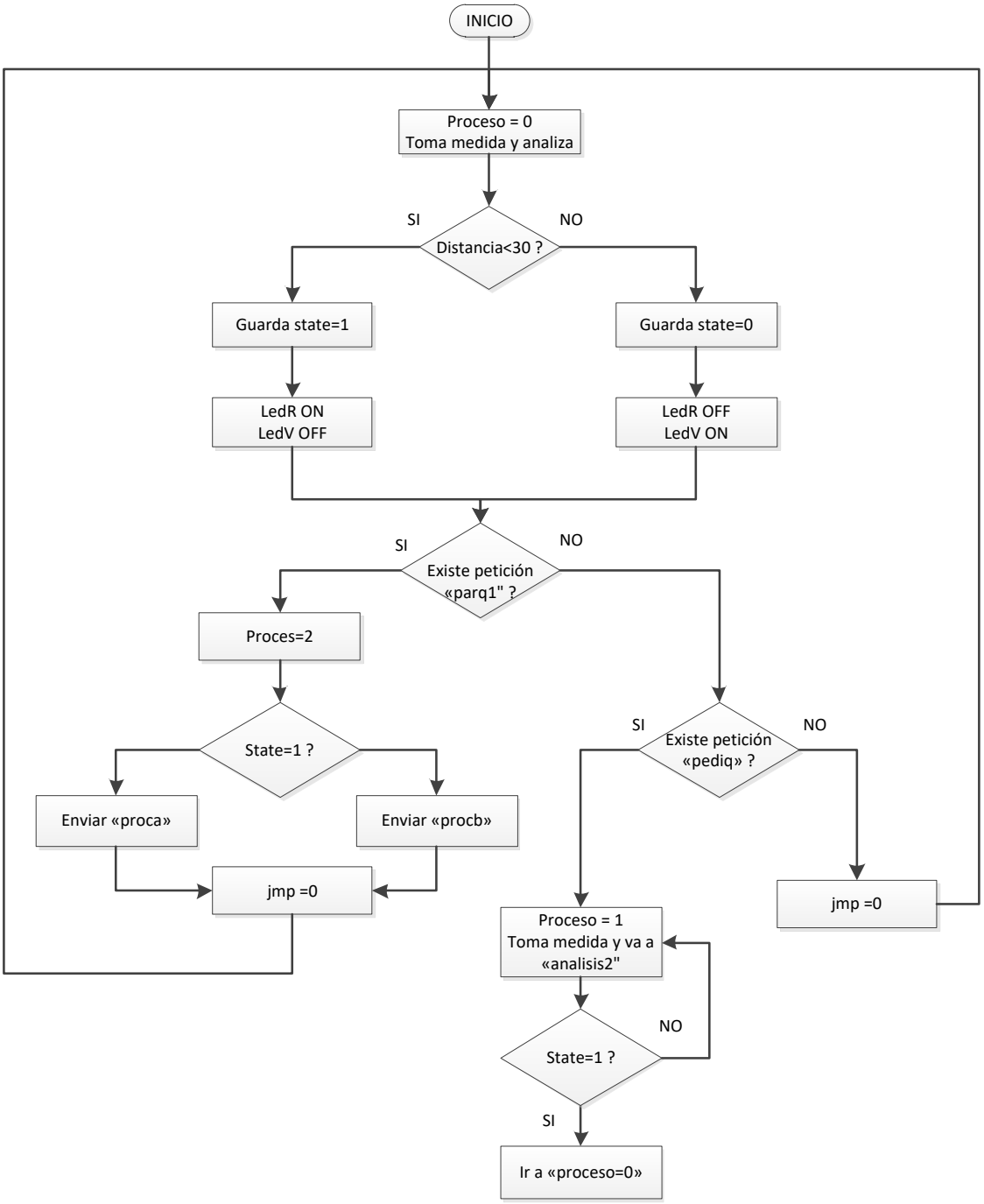


Figura 2.15. Diagrama de flujo del programa de la Tarjeta Sensor Transmisor (Tx)

Fuente: Elaborado por el autor

2.3. Establecimiento de la comunicación con la plataforma IoT.

2.3.1. Servidor IoT ThingSpeak

Las plataformas IoT brindan servicios para recopilar información recogida generalmente por sensores y microcontroladores, las cuales pueden ser desarrolladas por nosotros mismo o plataformas ya existentes con fines académicos o de lucro comercial.

ThingSpeak es una Interfaz de Programación de Aplicaciones API y viene a ser una plataforma de código abierto al que se puede acceder desde Internet y cuyo objetivo es recoger, almacenar, visualizar y manipular datos recogidos por sensores y dispositivos con hardware de código abierto como Arduino, Raspberry Pi o BeagleBone para el desarrollo de aplicaciones IoT.

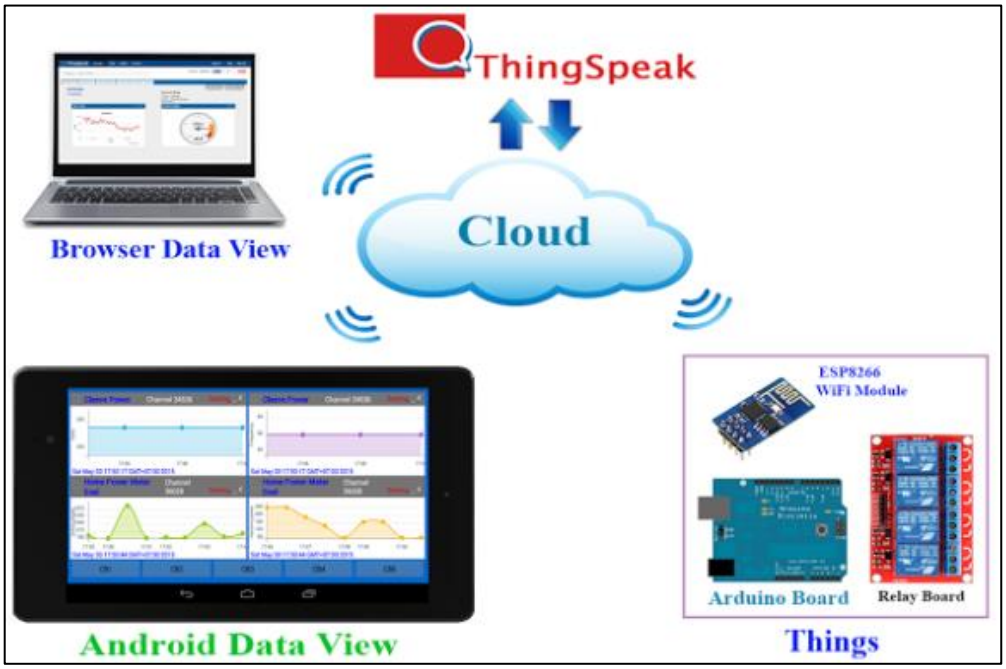


Figura 2.16. Esquema de conexiones de aplicaciones IoT con ThingSpeak

Fuente: (Aprendiendo Arduino, 2017)

2.3.2. Diseño de la aplicación en Mit App Inventor

Se ha seleccionado esta solución para nuestro diseño debido a muchas ventajas entre las cuales tenemos:

Mitt App Inventor es un servicio gratuito al que se accede a través de la nube y ofrece un entorno de desarrollo de aplicaciones para dispositivos móviles Android. Pese a que se trabaja en un navegador web permite almacenar el proyecto y también el seguimiento de los mismos además que permite generar archivos apk que pueden ser instalados como aplicación en cualquier dispositivo Android.

Para el acceso se lo realiza desde un navegador web y conectándose al siguiente link http://appinventor.mit.edu/explore/. Se ingresa con una cuenta normal de Google, Gmail.

Básicamente se trabaja con dos módulos: App Inventor Designer y App Inventor Blocks Editor.

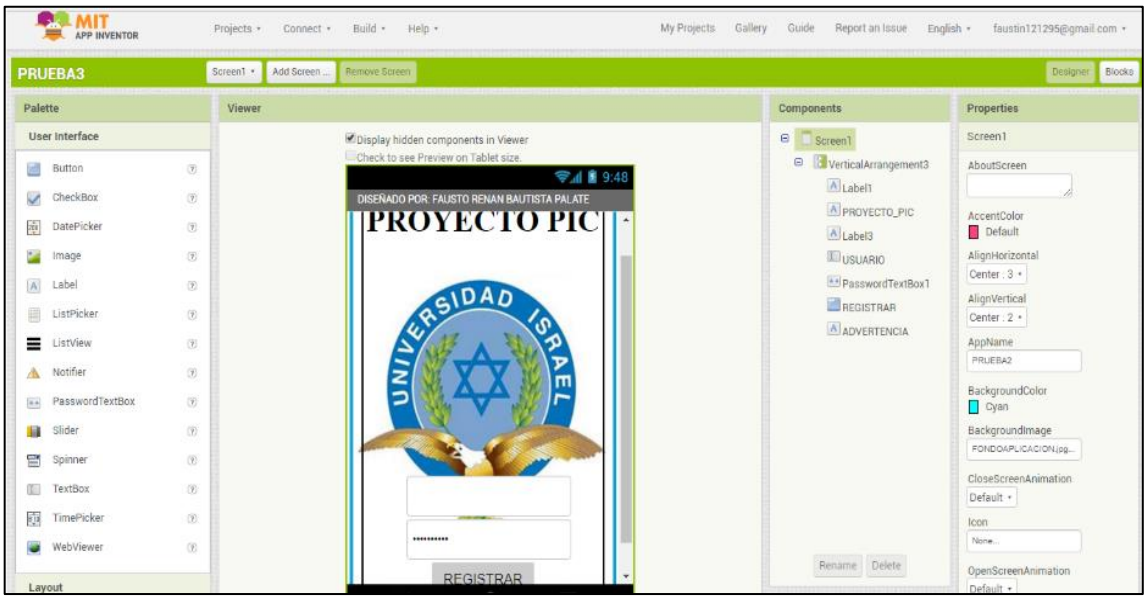


Figura 2.17. Pantalla Designer

Fuente: Elaborado por el autor

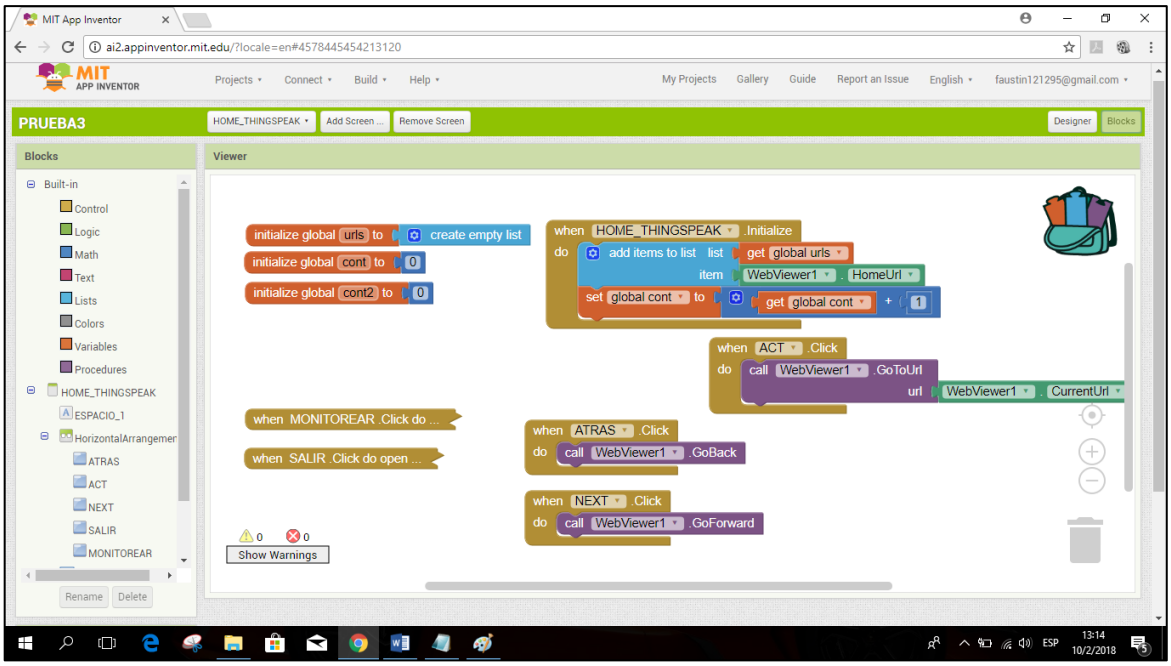


Figura 2.18. Pantalla Blocks Editor

Fuente: Elaborado por el autor

2.3.3. Diagrama de bloques

A continuación, se presenta la lógica que seguirán las diferentes etapas del proyecto.

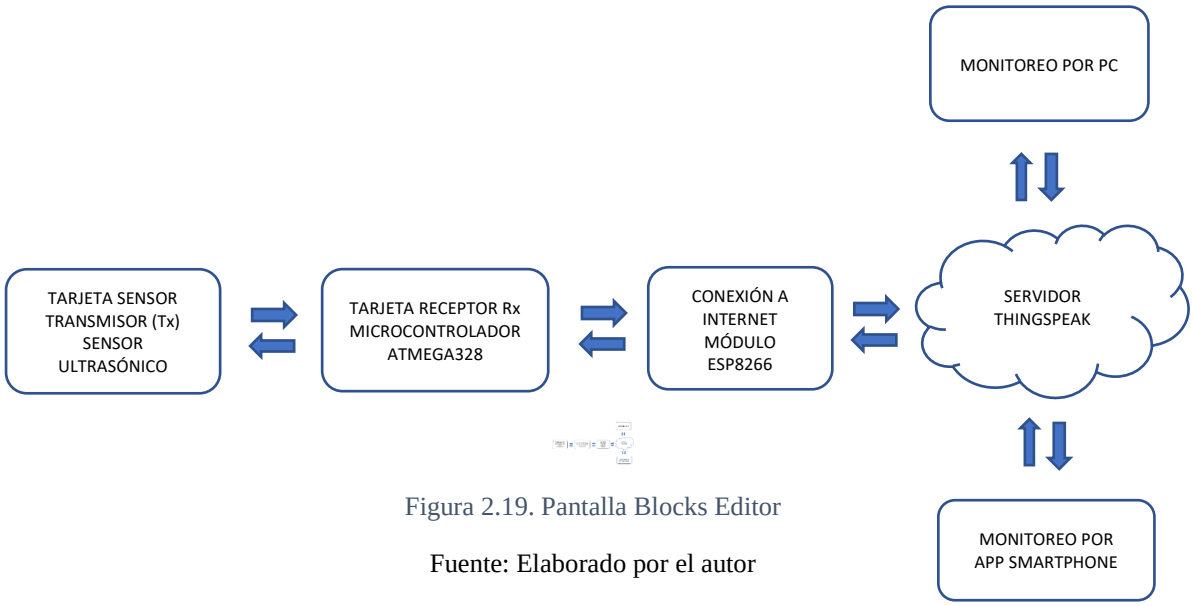


Figura 2.19. Pantalla Blocks Editor

Fuente: Elaborado por el autor

2.3.4. Diagrama del estacionamiento seleccionado para pruebas

En la siguiente ilustración se visualiza de manera esquemática la distribución de 8 de las plazas de estacionamiento del Conjunto Habitacional Camino Real ubicado en el centro histórico de la ciudad de Quito y la distribución de los sensores que como posición estratégica estaría en el techo del estacionamiento para detectar el auto desde arriba.

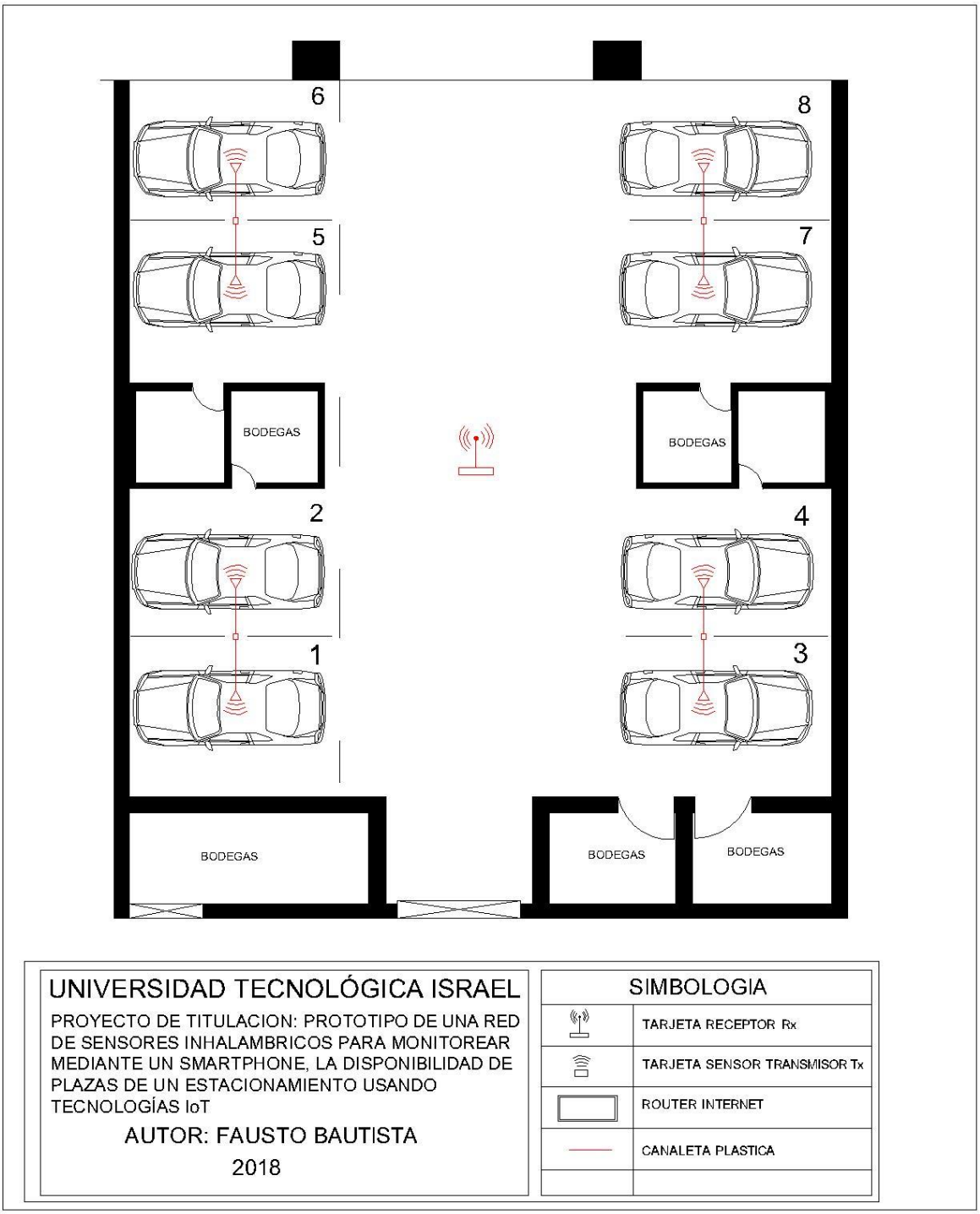


Figura 2.20. Diagrama del parqueadero del Conjunto Habitacional Camino Real

Fuente: Elaborado por el autor

CAPÍTULO 3

IMPLEMENTACIÓN

3.1. Fabricación de las placas de Sensores Tx y Tarjeta Receptor Rx

Esta placa se lo realizó en base a la impresión sobre el papel Transfer del diagrama PCB del capítulo 2, el mismo que sirvió para delinear las pistas en la baquelita también con la ayuda del marcador en los puntos donde hubo fallas en la adhesión del papel Transfer. Luego se procedió a aplicar la solución de cloruro férrico para definir las pistas de cobre y retirar el resto de material que no servía en la baquelita.

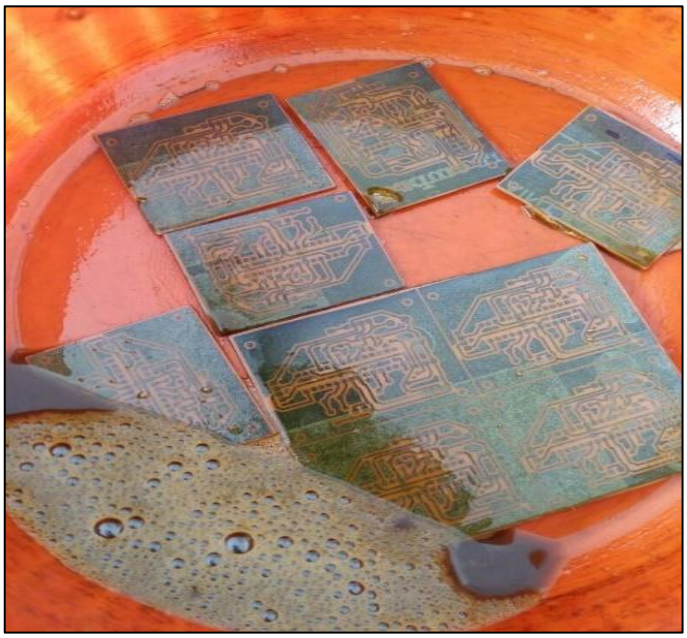


Figura 3.1. Sumersión de la baquelita en ácido

Fuente: Elaborado por el autor,

Luego se procedió a limpiar el papel y la tinta del marcador con una esponja metálica hasta verificar que las pistas de cobre tomen el brillo metálico.

Se procedió a realizar los orificios donde van a calzar los diferentes dispositivos con la ayuda de un taladro.

Figura 3.2. Realización de los orificios con el taladro

Fuente: Elaborado por el autor,

Después se procedió a soldar con el estaño todos los elementos en la tarjeta cuidando de no recalentar las pistas para que las mismas no se levanten.

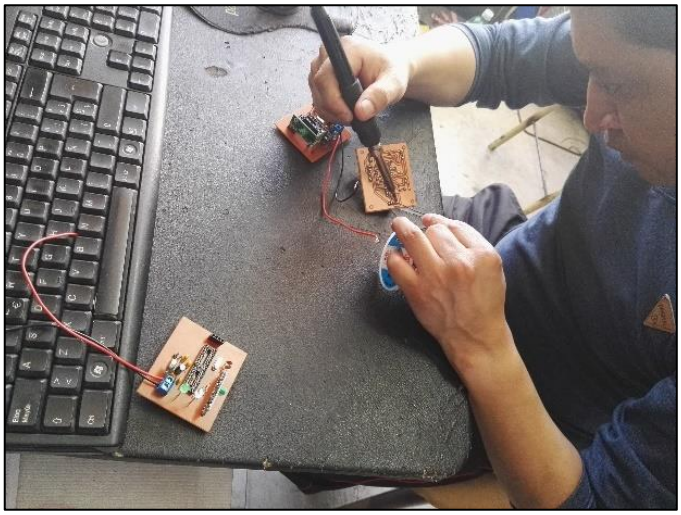


Figura 3.3. Suelda de elementos en la placa con estaño

Fuente: Elaborado por el autor

Una vez que se tiene listas las tarjetas, se puede adecuar cajas que contendrán a las mismas en su interior, facilitando su manipulación. Se utilizó cajas plásticas para proyectos de

19x10x6 cm que se encuentran en tiendas electrónicas y se las adecuó con aberturas para los sensores ultrasónicos y también a fijar los leds de alto brillo por la parte frontal (Rojo para ocupado y Verde para disponible) como se muestra a continuación:

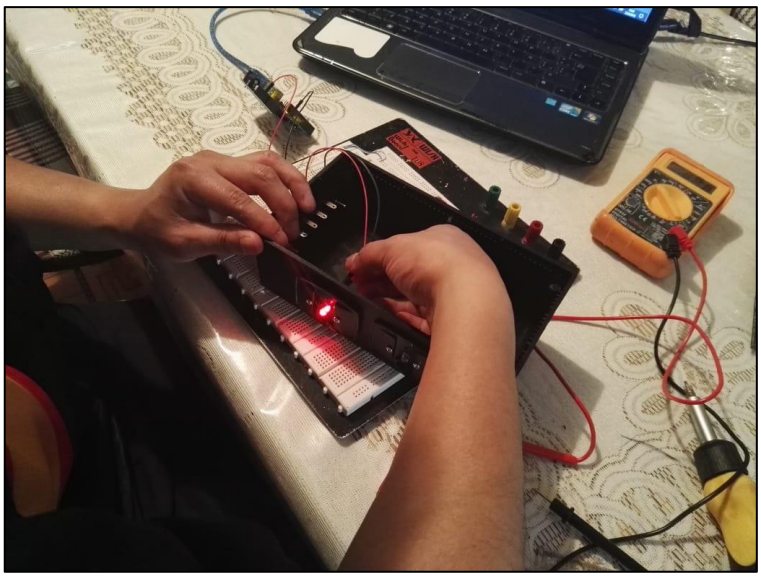


Figura 3.4.Colocación de luces en la caja de proyectos

Fuente: Elaborado por el autor

Luego se procedió a colocar lunas reflectivas en la parte frontal de la caja para amplificar la luz de los leds y sean más visibles.

Figura 3.5. Colocación de lunas reflectivas para visualización frontal de la señalización

Fuente: Elaborado por el autor

En la parte posterior de las cajas, se colocó terminales portajacks tipo hembra para la conexión de la alimentación de 110-120 VAC

Figura 3.6. Colocación de portajacks tipo hembra para alimentación

Fuente: Elaborado por el autor

Las tarjetas Sensor Transmisor (Tx) fueron sujetadas a la tapa superior por medio de postes metálicos que al mismo tiempo sirvieron para nivelar los sensores ultrasónicos en los orificios preparados en las cajas para aquello.

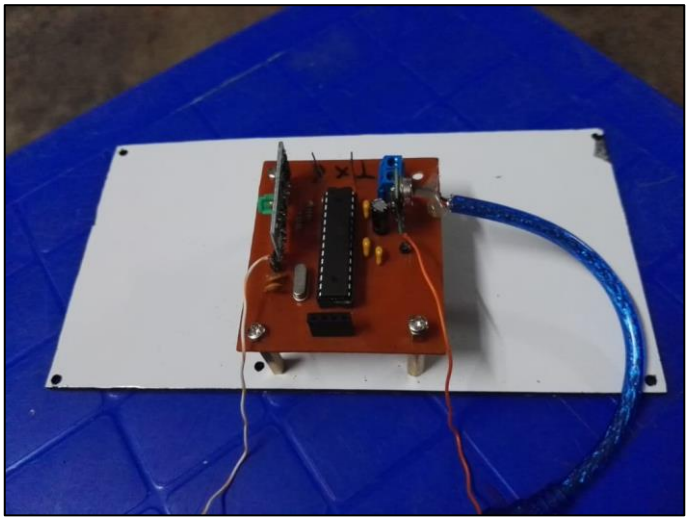


Figura 3.7. Sujeción de las tarjetas con postes metálicos

Fuente: Elaborado por el autor

Se procedió a sujetar la fuente de alimentación DC de la tarjeta a la caja por su interior y colocar mediante borneras el cable USB que sirvió para poder conectar a la fuente. A su vez la fuente fue soldada a los terminales portajacks y colocados para poder alojar la tarjeta junto con el sensor y la tapa por la parte superior.

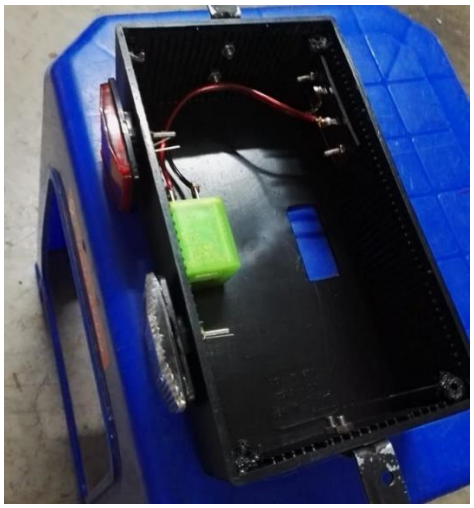


Figura 3.8. Disposición de los elementos dentro de la caja de proyectos

Fuente: Elaborado por el autor

3.2. Programación del módulo wi-fi ESP8266 de la Tarjeta Receptor (Rx)

Por otro lado, para la programación del ESP8266 se procedió a armar el siguiente circuito provisional para poder obtener 3VDC de alimentación el cual nos lo puede proveer una placa Arduino (únicamente para la parte de la programación) en uno de sus pines retirando previamente el microcontrolador.

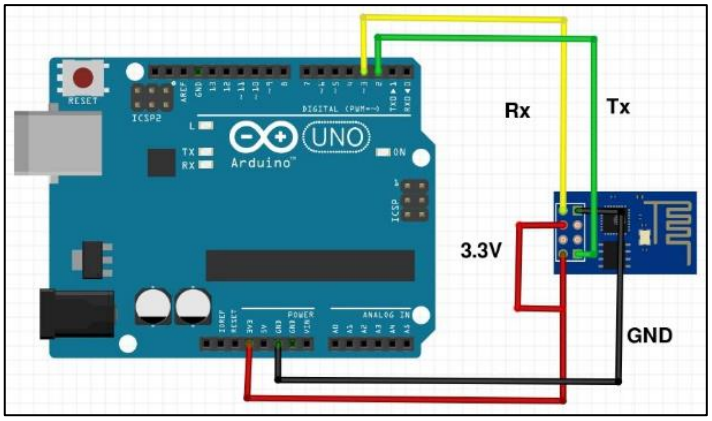


Figura 3.9. Conexiones para configurar el ESP8266

Fuente: (Prometec, 2017)

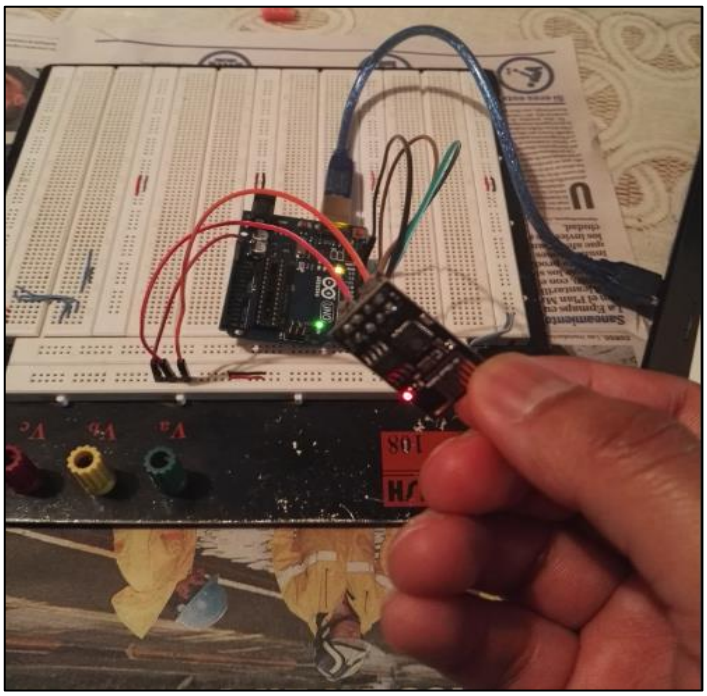


Figura 3.10. ESP8266 conectado para ser configurado con comandos AT

Fuente: Elaborado por el autor

Mediante comandos AT y a través del IDE de Arduino, se procedió a configurar los principales parámetros como el SSID de la red de la oficina de la administración del Conjunto, ya que el mismo se encuentra a la entrada de los estacionamientos que sirvieron de prueba. También se configuró el password, el puerto 80 como se muestra a continuación:

3.2.1 Configuración de la red WiFi

Los comandos AT ejecutados en la consola del IDE Arduino para conseguir obtener una IP del router que brinda Internet, son los siguientes:

Tabla 3.1. Configuración del ESP8266 con comandos AT

AT	Prueba la comunicación serial
AT+CWLAP	Busca las redes WiFi existentes a nuestro alrededor
AT+CWJAP="Camino Real","camino2016"	Configura el SSID y la contraseña de acceso a la red WiFi
AT+CIPSERVER=1,80	Abre el puerto 80

Autor: (Naylamp Mechatronics, 2016)

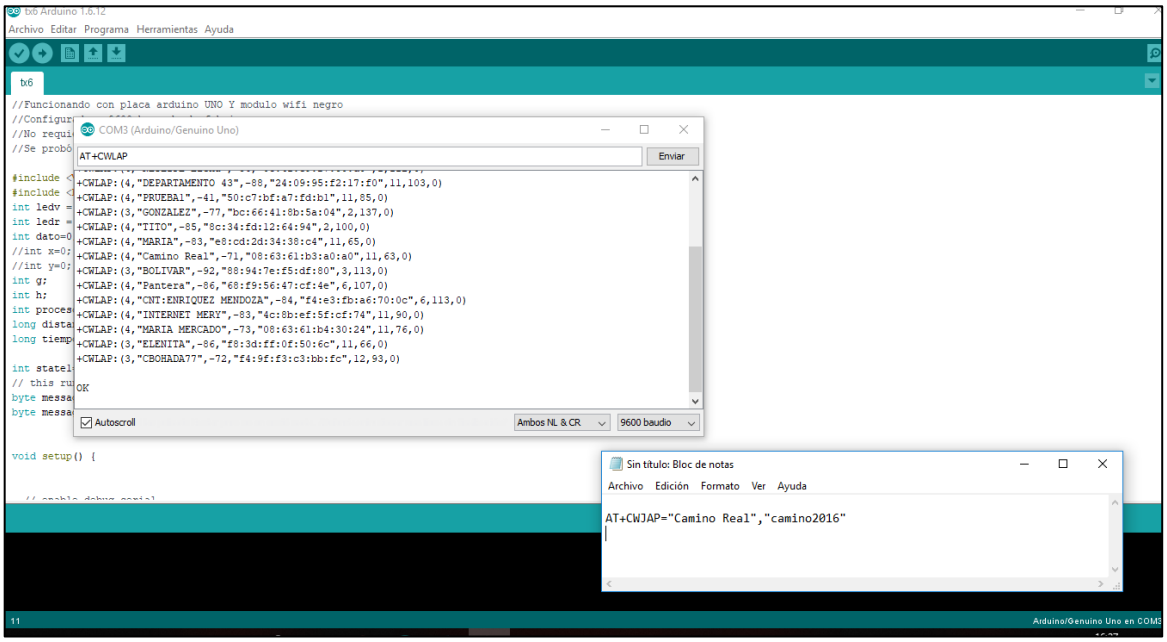


Figura 3.11. Búsqueda de las redes wi-fi disponibles

Fuente: Elaborado por el autor

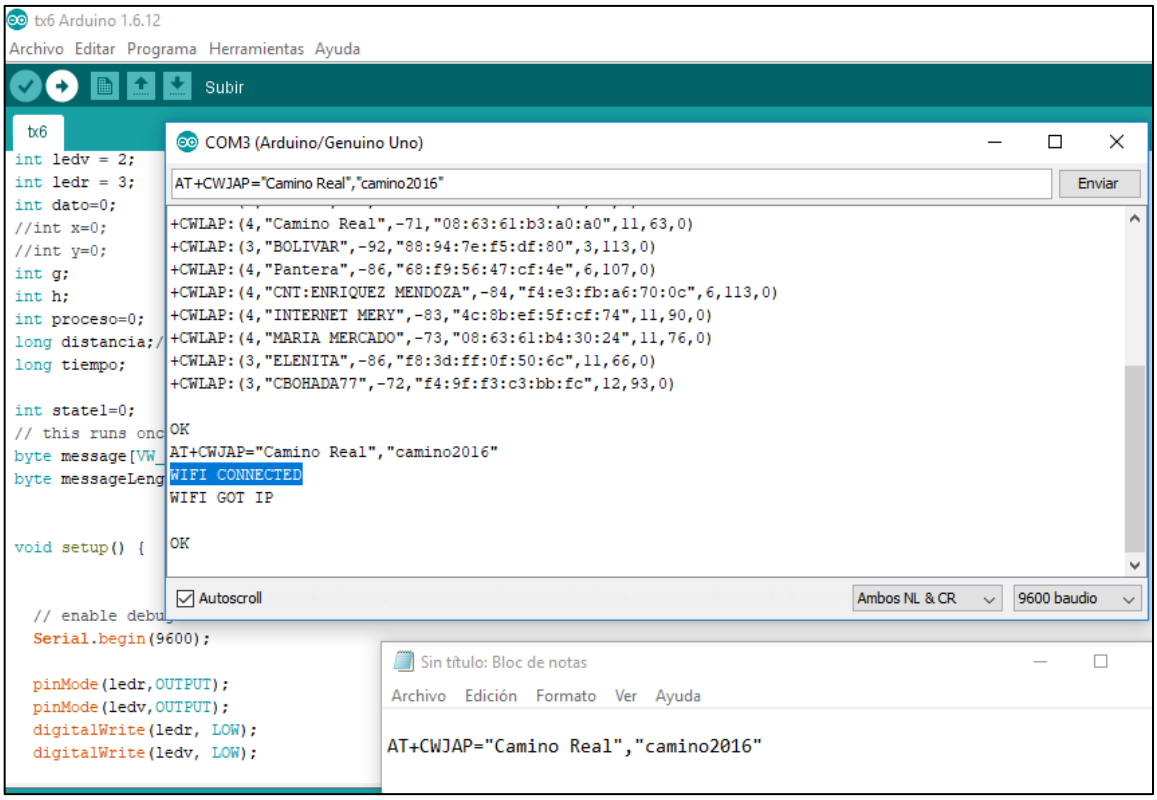


Figura 3.12. Conexión del módulo ESP8266 con la red WiFi del conjunto Camino Real

Fuente: Elaborado por el autor

3.2.2. Probando conectividad en la red WiFi

Podemos enviar un mensaje de prueba desde el navegador web apuntando a la IP asignada al ESP8266 y por el puerto 80. Los comandos AT y el socket completo sería el señalado a continuación:

Tabla 3.2. Verificando conectividad del ESP8266 con comandos AT

AT+CIFSR	Muestra entre otra información la IP asignada al dispositivo dentro de la red WiFi
AT+CIPSERVER=1,80	Abre el puerto 80

Autor: (Naylamp Mechatronics, 2016)

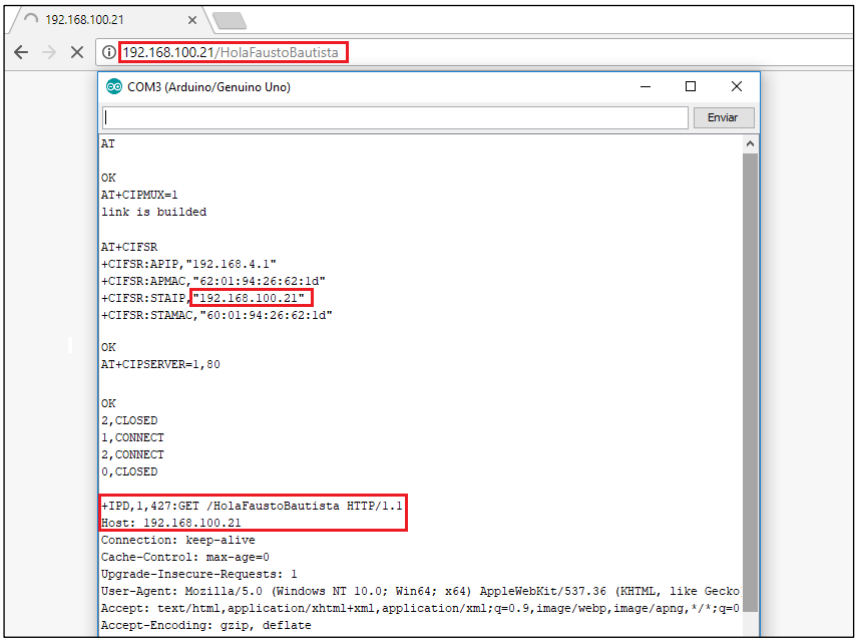


Figura 3.13. Pruebas de conectividad del ESP8266 con la red WiFi

Fuente: Elaborado por el autor

3.3. Programación de las tarjetas Sensor Transmisor Tx

Para la programación de los microcontroladores incluimos la librería VirtualWire para que se pueda establecer la comunicación entre los microcontroladores ATMEGA328 de las Tarjetas Sensor Transmisor Tx con la Tarjeta Receptor Rx. La librería será la encargada de tramitar las funciones de los radios 433MHz como lo son la recepción y envío de paquetes de datos, la comprobación de errores entre otras.

3.3.1 Programa para el sensor 1

Solo por motivos de programación del microcontrolador ATMEGA328 se utilizó la placa Arduino Uno para luego ser puesto en funcionamiento en la Tarjeta Sensor Transmisor (Tx) que fue diseñada anteriormente. En base a esto, elegimos en el IDE Arduino la tarjeta Arduino UNO.

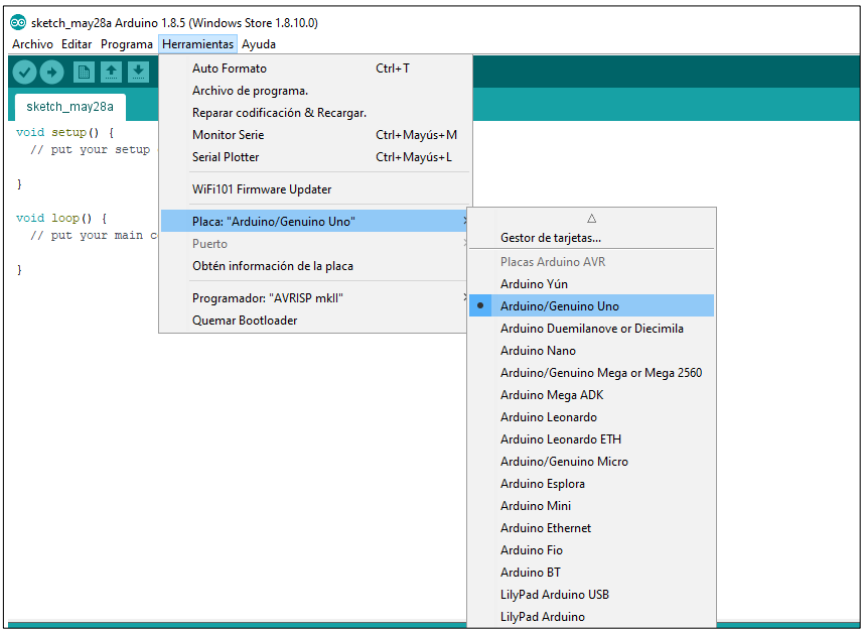


Figura 3.14. Configurando IDE Arduino para programar el microcontrolador ATMEGA328

Fuente: Elaborado por el autor

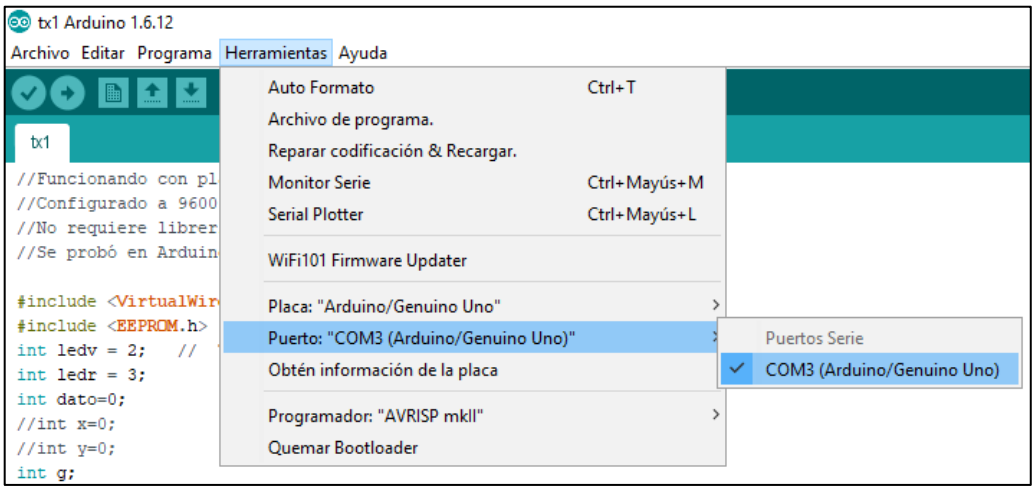


Figura 3.15. Selección del puerto com para la comunicación serial

Fuente: Elaborado por el autor

Al incluir la librería “VirtualWire.h” por defecto utiliza con los pines 11 y 12 de la tarjeta Sensor TX.

Tabla 3.3. Configuración del microcontrolador ATMEGA328

pinMode(11)	Activación del pin 11 para la Recepción
pinMode(12)	Activación del pin 12 para la Transmisión
vw_setup(2000);	Espera tiempo de 2 segundos e Inicia comunicación con el módulo RF
vw_rx_start();	Inicia la función de recepción para hacerlo bidireccional
proceso = EEPROM.read(10);	Lee el último estado guardado en la memoria al momento de reiniciar

Autor: (Naylamp Mechatronics, 2016)

Para establecer comunicación entre los pines de la tarjeta Sensor Transmisor y los sensores ultrasónicos tenemos:

pinMode(9, OUTPUT);	Activación del pin 9 como salida: para los sensores
pinMode(8, INPUT);	Activación del pin 8 como entrada: para los sensores

El código de programación a detalle se encuentra como Anexo A

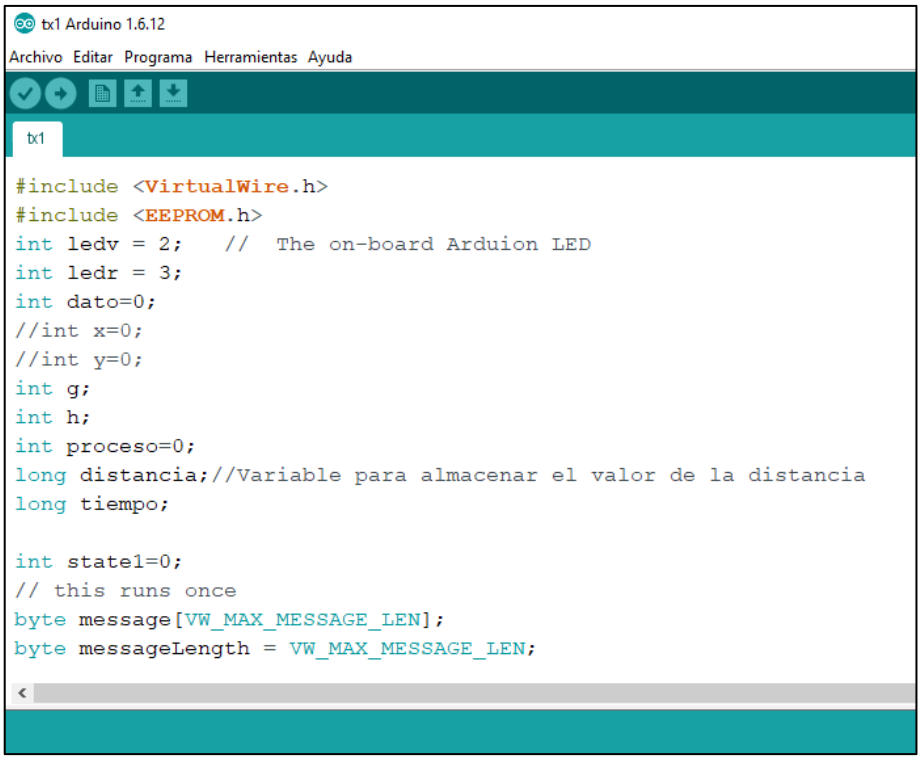


Figura 3.16. Carga del código en el microcontrolador

Fuente: Elaborado por el autor

Para el void loop la estructura está basada en la lógica seguida en el Diagrama de flujo del programa de la Tarjeta Sensor Transmisor (Tx) descrita en la sección 2.13. de este documento.

Se realizó bloques de instrucciones denominadas procesos (0,1 y 2) y por medio de subrutinas se realizan diferentes acciones; resumidas de la siguiente manera:

a) Proceso 0: Determina en función de las mediciones de su sensor ultrasónico y la subrutina "análisis" que realiza comparaciones; si existe presencia o ausencia de un auto y colocando dicho estado de manera visible en los pines que corresponden a los leds. (Verde para libre y rojo para ocupado). Este estado también se guarda en la variable llamada state (pudiendo ser state1=1 para ocupado o estate1=0 para libre).

b) Proceso 1: Entra a funcionar cuando le llega un pedido de la tarjeta de control de estados (código pediq) para hacer realizar un cambio de estado (a reservado); por lo que entra a analizar si se encuentra disponible u ocupado para poder ejecutar dicho cambio y por medio de la subrutina "análisis 2". Para hacer visible si pudo ejecutar el cambio de estado ejecuta en los pines de salida del led rojo un temporizador de manera que se vea un rojo parpadeante.

c) Proceso 2: Entra a funcionar cuando Rx hace la consulta al Tx sobre el estado del parqueadero y éste envía como respuesta un código, si está ocupado (proca) o si se encuentra libre (procb). Luego de enviar la respuesta nuevamente regresa a la subrutina Proceso 0 para mantener de forma permanente ejecutando este loop.

En el Anexo A, se muestra el código de programación del void loop donde se puede ver de manera detallada los procesos y subrutinas descritos en los numerales anteriores.

3.4. Programación Tarjeta Receptor Rx

Para el encabezado y void setup lo principal en resaltar serían las siguientes instrucciones: Se incluyó también la librería "VirtualWire.h" que por defecto utiliza los pines 11 y 12 de la Tarjeta Receptor Rx para la comunicación a nivel de Radio Frecuencia con las tarjetas Sensor Tx.

Tabla 3.4. Configuración de la comunicación serial

#include <SoftwareSerial.h>	Para la comunicación serial con el ESP8266 por los pines 6y5
SoftwareSerial Serial2x(6,5);	Se declaró el pin 5 rx y 6 tx
#include <stdlib.h>	Librería estándar para funciones básicas de comunicación con ThingSpeak
Serial2x.begin(9600);	Para comunicación del micro controlador y el módulo ESP8266

Autor: (Ballesteros; 2016)

La comunicación con ThingSpeak se puede realizar a partir de configurar el API Key gracias a la librería

String apiKey = "HG0ZPK7ADBT9FG8T"	Ingresa la API KEY que se generó al crear nuestro canal

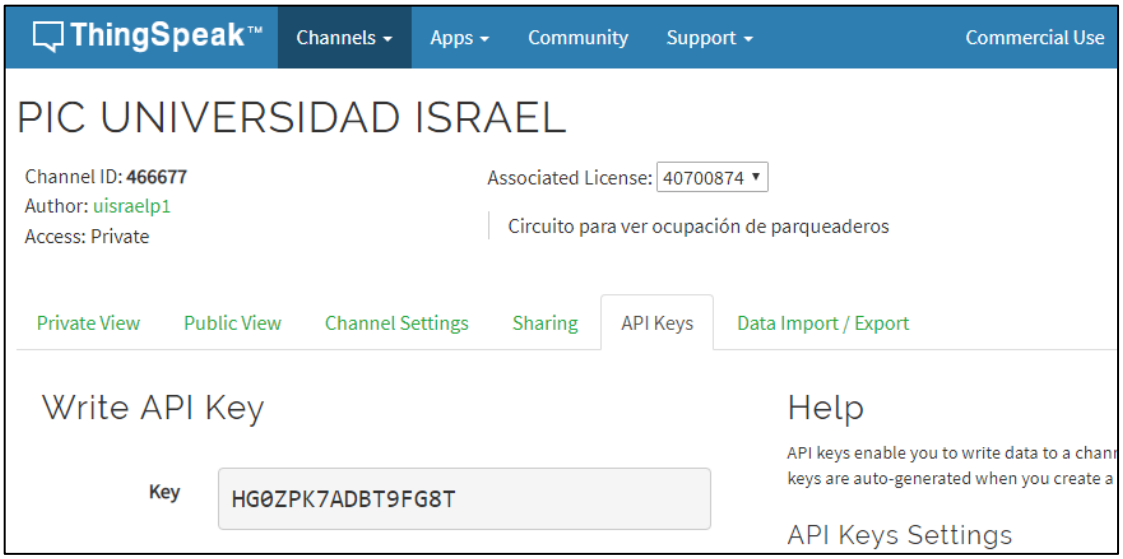


Figura 3.17. API de Thingpeak programado en el código de la Tarjeta Receptora Rx

Fuente: Elaborado por el autor

Se define como variables a los estados que le llegarán desde cada Sensor Tx de la red.

String state1;
String state2;
String state3;
String state4;
String state5;
String state6;
String state7;
String state8;

En resumen, para la parte del void loop en esencia se realiza la lógica detallada en el diagrama de flujo de la Tarjeta Receptor Rx; donde básicamente esta tarjeta realiza el pedido del estado en que se encuentra cada Sensor Tx por medio de códigos “parq1 hasta parq8”. Los códigos “parq1 hasta el 8” son enviados dentro de la subrutina ejecución de instrucciones que tienen por nombre “pedido=n” donde n va de 1 a 16.

Estos pasos se ejecutan uno por el envío del código y otro por el análisis del código recibido por respuesta a “parq1”. Nuevamente se ejecuta el proceso con el envío del código “parq2” para la consulta de estado del parqueadero 2.

Cuando se recibe la respuesta de cada Sensor Tx, este recibe por código “proca” (ocupado) o “procb” (libre) y según esto guarda los estados de los 8 parqueaderos con el nombre state1 hasta state8 que luego nos sirve para el envío hacia ThingSpeak.

Al final de ejecutar pedido=16 este pasa a convertirse en pedido=0 en donde se ejecuta el envío de los states por medio de comandos AT los cuales se encargan de la conexión TCP con la IP de ThingSpeak, abrir el puerto 80 y subir los states al servidor.

El código de programación de igual manera se encuentra incluido en el anexo E:

3.5. Programación del microcontrolador de la Tarjeta de Cambio de Estados

Se comunicó con el microcontrolador a través del puerto COM3 y se incluyó la librería ESPDUINO que contiene el firmware para establecer la comunicación con el módulo ESP8266 perteneciente a la tarjeta de cambio de estados. Se procedió a compilar el programa y a subirlo en el microcontrolador de la Tarjeta de Cambio de Estados como se muestra a continuación:

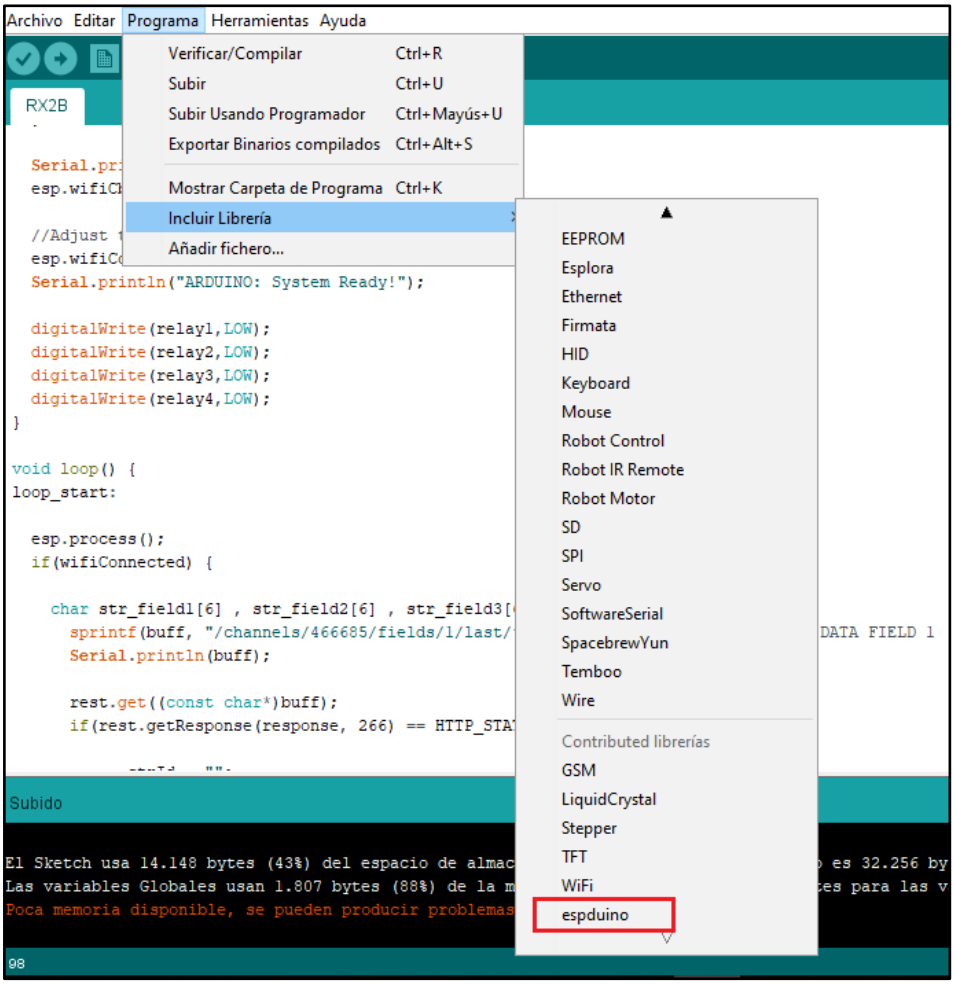


Figura 3.18. Inclusión de la librería ESPDUINO al programa

Fuente: Elaborado por el autor

Establecemos la conexión del módulo WiFi ESP8266 que pertenece a esta tarjeta con el SSID del router del ISP, el mismo que brinda el servicio de Internet al conjunto Camino Real junto con la contraseña de acceso y cargamos el programa que se encuentra incluido en el anexo A.

El router se encuentra ubicado en la oficina del administrador del conjunto que también se encuentra próximo a los estacionamientos del conjunto.

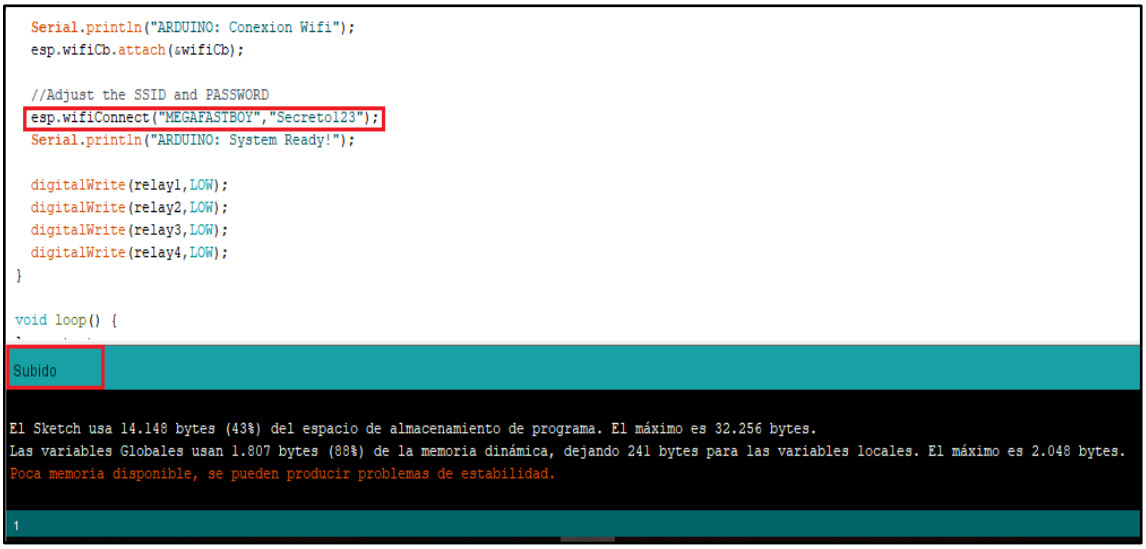


Figura 3.19. Configuración del SSID y contraseña del WiFi en el microcontrolador de cambios de estados

Fuente: Elaborado por el autor

Para permitir la programación se incluyó las siguientes librerías:

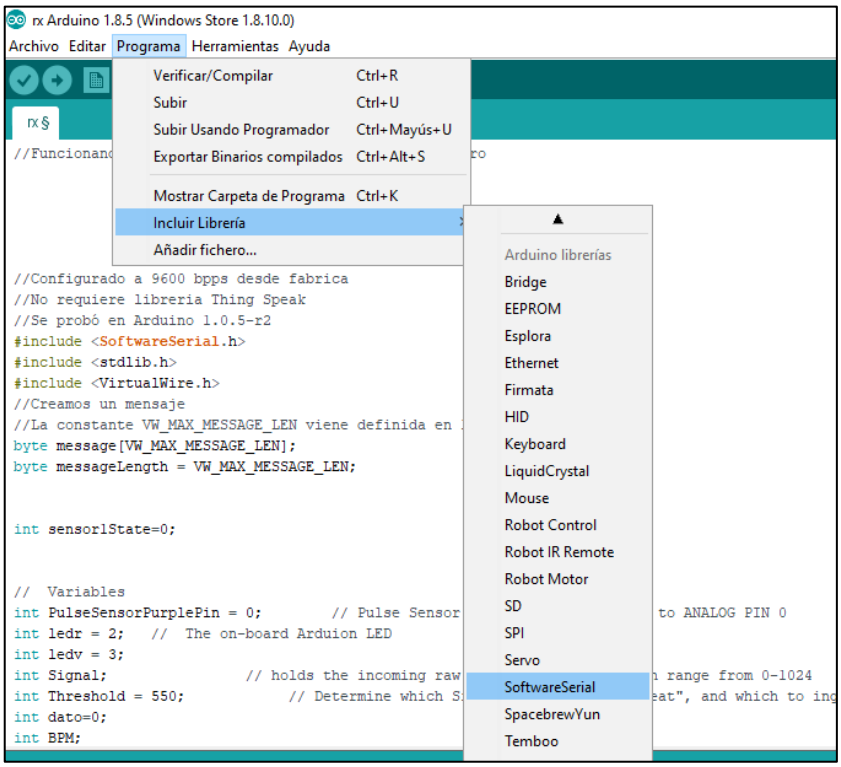


Figura 3.20. Librería utilizada para la comunicación serial

Fuente: Elaborado por el autor

El código de programación de igual manera se encuentra incluido en el anexo A:

3.6. Subir la información a ThingSpeak

3.6.1. Adquisición de licencia para servidor IoT denominado Thingspeak

ThingSpeak brinda la posibilidad de trabajar con licencia gratuita, pero con ciertas limitaciones. En la figura en cambio se muestran los diferentes tipos de licencia con los alcances y términos de uso correspondientes.

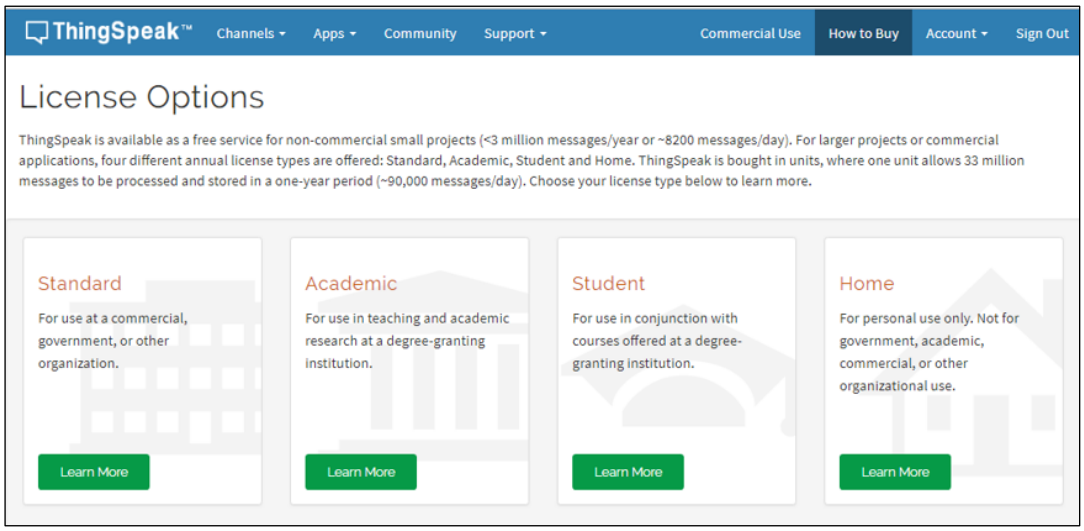


Figura 3.21. Tipos de licencias que ofrece ThingSpeak

Fuente: Elaborado por el autor

De manera explícita se muestra en el gráfico a continuación, las ventajas de utilizar la licencia "STUDENT" sobre la licencia gratuita.

	FREE For small non-commercial projects	STUDENT For students at degree-granting institutions[1]
Scalable for larger projects	✖ No. Annual usage is capped.	✔
Number of messages	3 million/year (~8,200/day)[2]	33 million/year per unit (~90,000/day per unit)[2]
Message update interval limit	Every 15 seconds	Every second
MATLAB Compute Timeout	20 seconds	20 seconds
Number of simultaneous MQTT subscriptions	Limited to 3	50 per unit
Private channel sharing	Limited to 3 shares	Unlimited
Technical Support	Forum	Forum

Figura 3.22. Ventajas de la licencia "Student" sobre la licencia "Free"

Fuente: Elaborado por el autor

Se llena la información requerida para la adquisición de la licencia

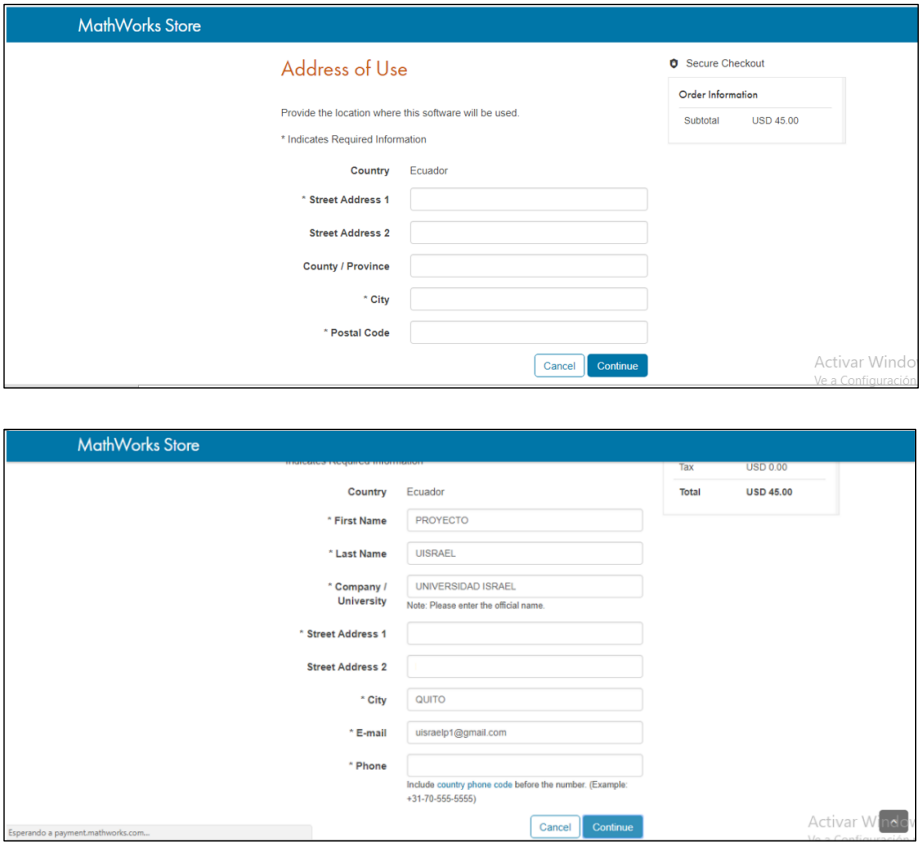


Figura 3.23. Llenando Datos para obtener la licencia "Student"

Fuente: Elaborado por el autor

Finalmente se obtiene la confirmación de que la licencia ya se encuentra vigente.

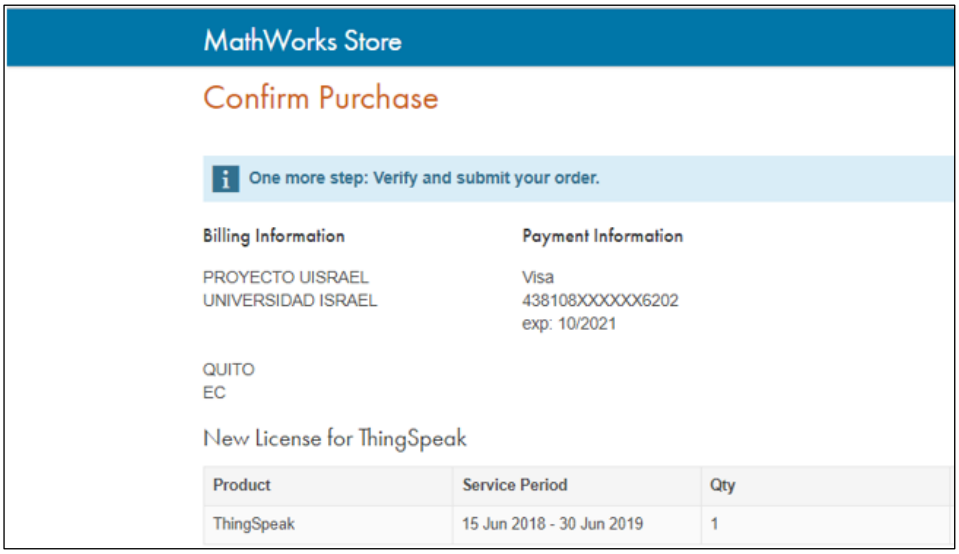


Figura 3.24. Confirmación de Activación de licencia

Fuente: Elaborado por el autor

3.6.2. Creación de fields para cada estacionamiento

Se ingresa en el link: https://thingspeak.com/login para crear nuestro canal llenando los siguientes campos

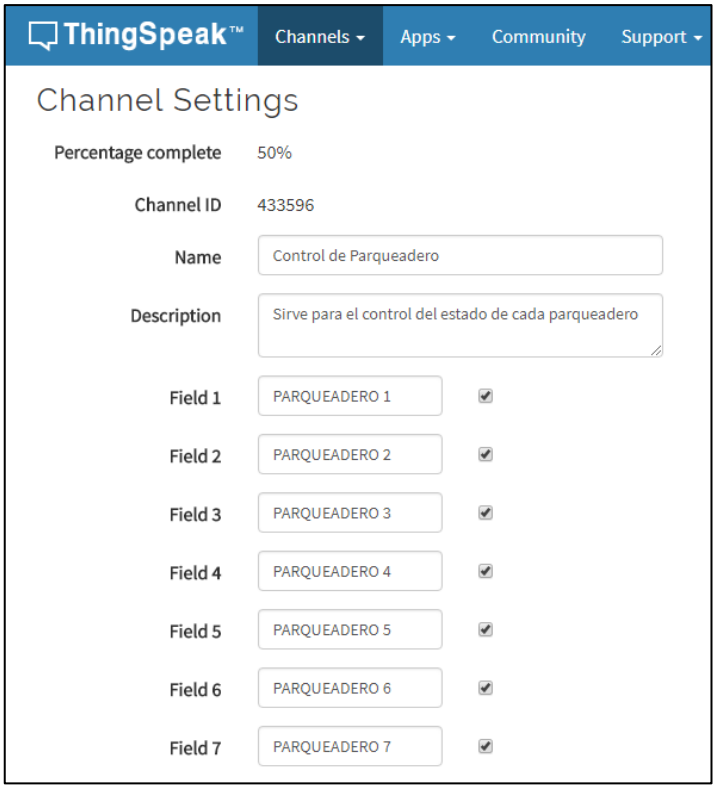


Figura 3.25. Creación del canal y de las gráficas de los sensores

Fuente: Elaborado por el autor

También se debe tener en cuenta el APIkey con el que el servidor establece conexión con el microcontrolador y poder subir la información al servidor pudiendo ser estos de lectura y escritura. En la gráfica a continuación se puede ver el API generado en ThingSpeak y la configuración en el programa de la Tarjeta Receptor (Rx)

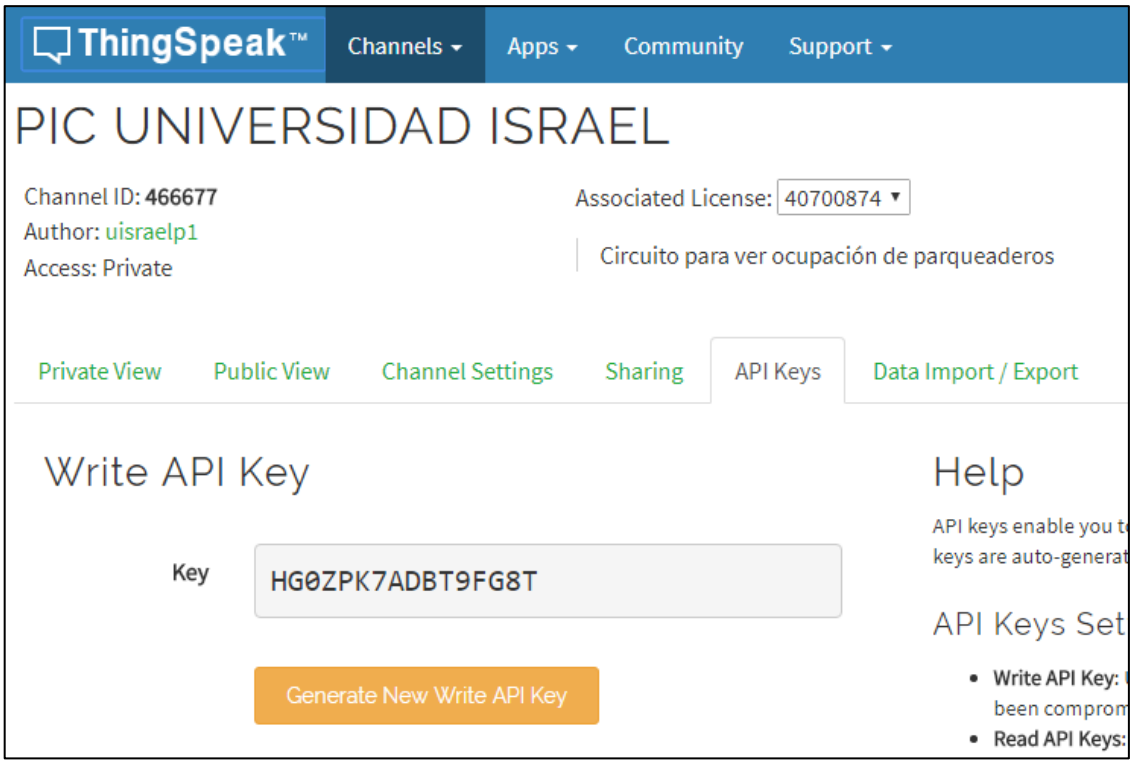
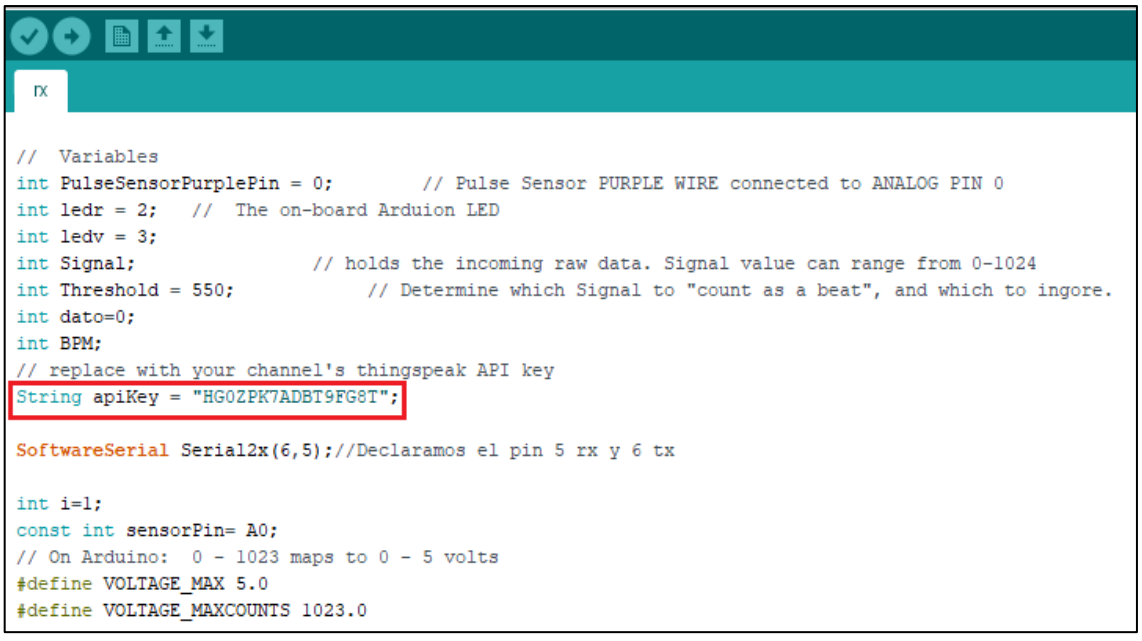


Figura 3.26. Generación del API key en ThingSpeak

Fuente: Elaborado por el autor



rx

```
//  Variables
int PulseSensorPurplePin = 0;        // Pulse Sensor PURPLE WIRE connected to ANALOG PIN 0
int ledr = 2;   //  The on-board Arduion LED
int ledv = 3;
int Signal;                // holds the incoming raw data. Signal value can range from 0-1024
int Threshold = 550;            // Determine which Signal to "count as a beat", and which to ingore.
int dato=0;
int BPM;
// replace with your channel's thingspeak API key
String apiKey = "HG0ZPK7ADBT9FG8T";

SoftwareSerial Serial2x(6,5);//Declaramos el pin 5 rx y 6 tx

int i=1;
const int sensorPin= A0;
// On Arduino:  0 - 1023 maps to 0 - 5 volts
#define VOLTAGE_MAX 5.0
#define VOLTAGE_MAXCOUNTS 1023.0
```

Figura 3.27. Configuración del API key en el programa de la Tarjeta Receptor (Rx)

Fuente: Elaborado por el autor

Una vez establecida la conexión con ThingSpeak se empiezan a generar las gráficas correspondientes a cada Field el cual fue asignado a los sensores.

En primera instancia se consiguió lecturas de los sensores 1 y 3 en donde se procedió a probar los sensores manualmente mediante obstrucción para simular la presencia de un auto.

A continuación, las gráficas obtenidas de las pruebas realizadas:

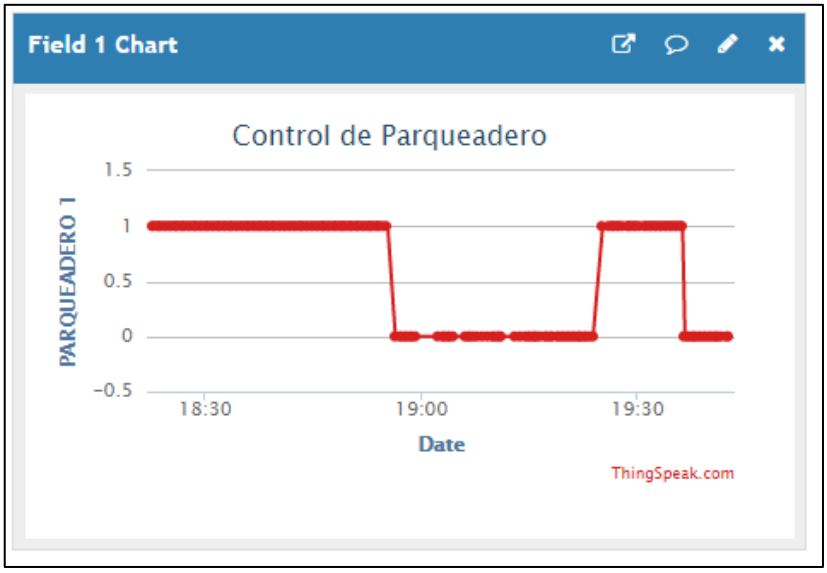


Figura 3.28. Lecturas del sensor 1 en el Field1 de ThingSpeak

Fuente: Elaborado por el autor

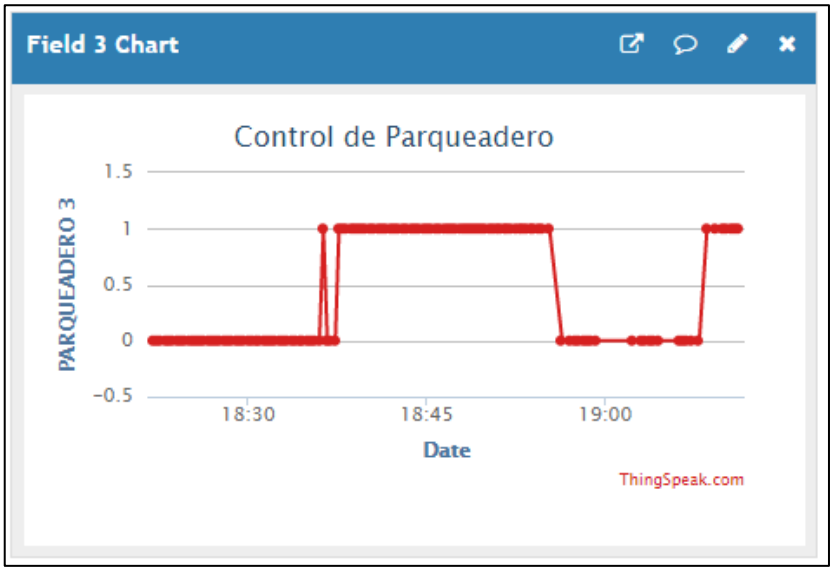


Figura 3.29. Lecturas del sensor 3 en el Field1 de ThingSpeak

Fuente: Elaborado por el autor

Se verifica que según la programación en el microcontrolador; cuando se coloca una obstrucción frente al sensor la gráfica se coloca en 1, mientras que al retirar el obstáculo la gráfica nuevamente regresa a 0.

Finalmente, y gracias al programa del microcontrolador en conjunto con las librerías VirtualWire.h se verifica que se ha establecido la transferencia de la información de los estados de cada sensor hacia el servidor IoT ThingSpeak. Se puede ya visualizar por medio de un 1 del servidor que se tiene un obstáculo en el sitio donde se encuentra el sensor y un 0 nos indica que no existe ningún tipo de obstáculo en el sitio del sensor.

A continuación, se muestra las gráficas generadas por la información proveniente de los 8 sensores armados y configurados:

Figura 3.30. Muestra de información proveniente de los sensores en el servidor IoT ThingSpeak (1)

Fuente: Elaborado por el autor

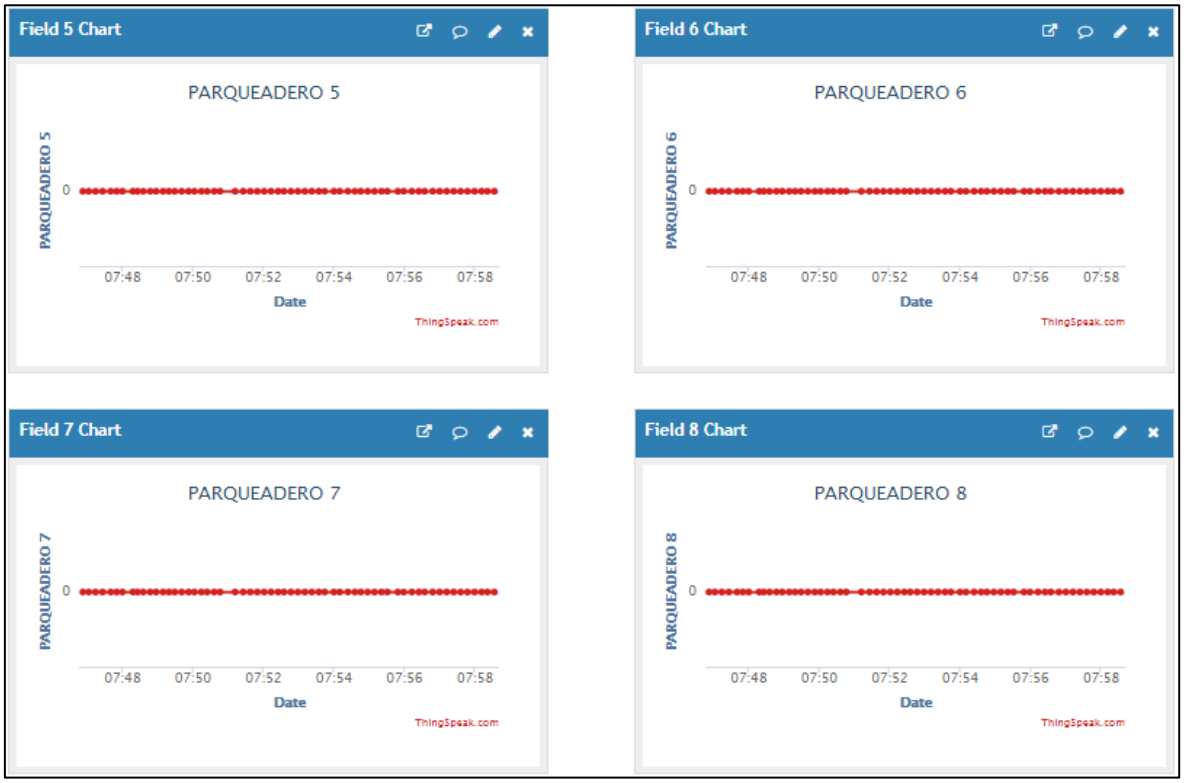


Figura 3.31. Muestra de información proveniente de los sensores en el servidor IoT ThingSpeak (2)

Fuente: Elaborado por el autor

Se crea el canal y queda registrada la fecha en el servidor

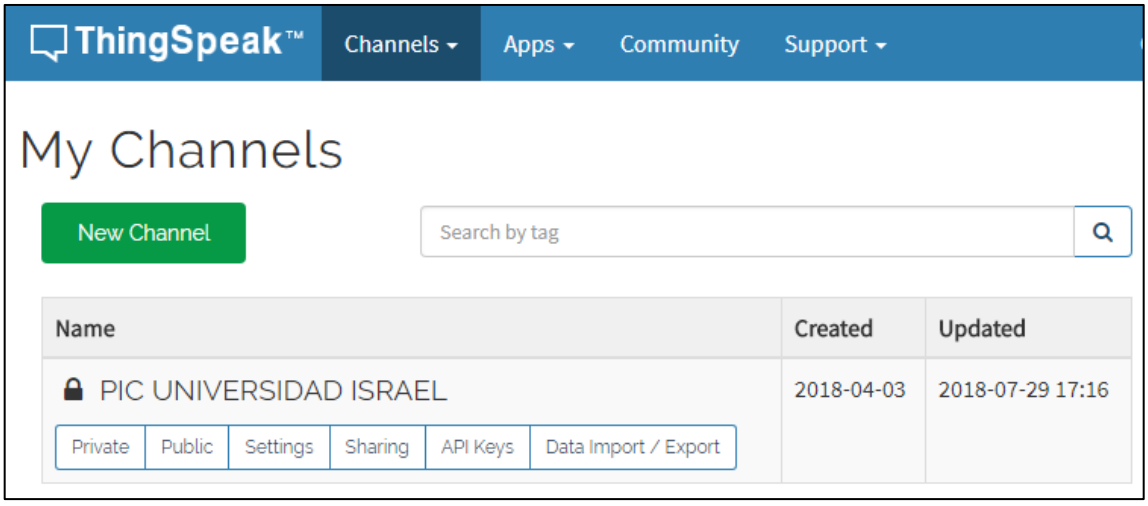


Figura 3.32. Registro de fecha de creación del canal en el servidor

Fuente: Elaborado por el autor

3.7. Diseño e implementación de la aplicación de control de administrador en App Inventor

3.7.1. Pantalla de presentación

Para la parte del diseño, la primera pantalla de presentación muestra el logo de la Universidad y el título del proyecto

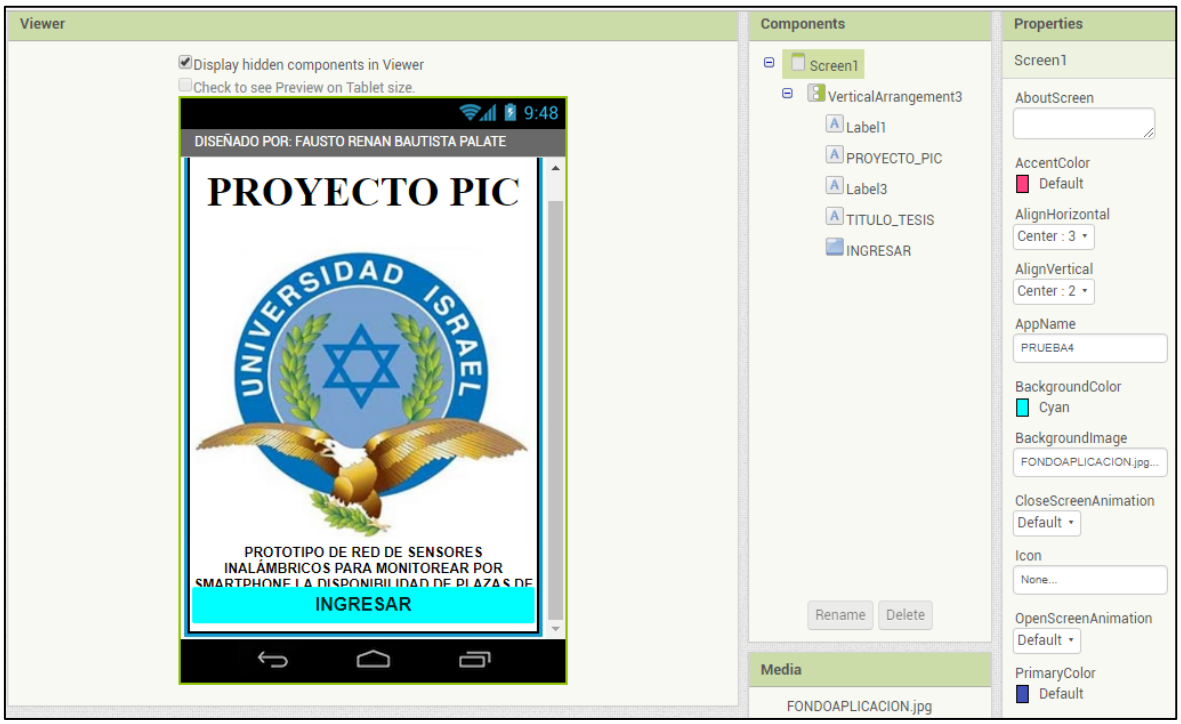


Figura 3.33. Pantalla de presentación

Fuente: Elaborado por el autor

3.7.2. Pantalla VISUALIZADOR_ESTACIONAMIENTOS

Se despliega la pantalla para visualizar los estacionamientos y los botones para acceder mediante credenciales a cuatro ambientes:

- Botón THINGSPEAK: Permite mediante credenciales de administrador, acceder a la pantalla HOME_THINGSPEAK que permite navegar en el servidor del mismo nombre y poder pasar por sus gráficas y bases de datos,
- Botón CONTROL DE RESERVAS: Permite mediante credenciales de administrador, acceder a la pantalla AREA_DE_RESERVAS para tener control de los estados de los sensores.

- Banner de loggeo: Permite mediante credenciales de administrador y de usuario acceder al visualizador de estacionamientos,
- Botón VISUALIZADOR DE ESTACIONAMIENTOS 2: Permite mediante credenciales de administrador y de usuario acceder de manera directa a la pantalla de visualización de estacionamientos. Al clickear este botón se llama a la función Activity Starter que nos dirige de manera directa al visualizador descrito en el ítem anterior.

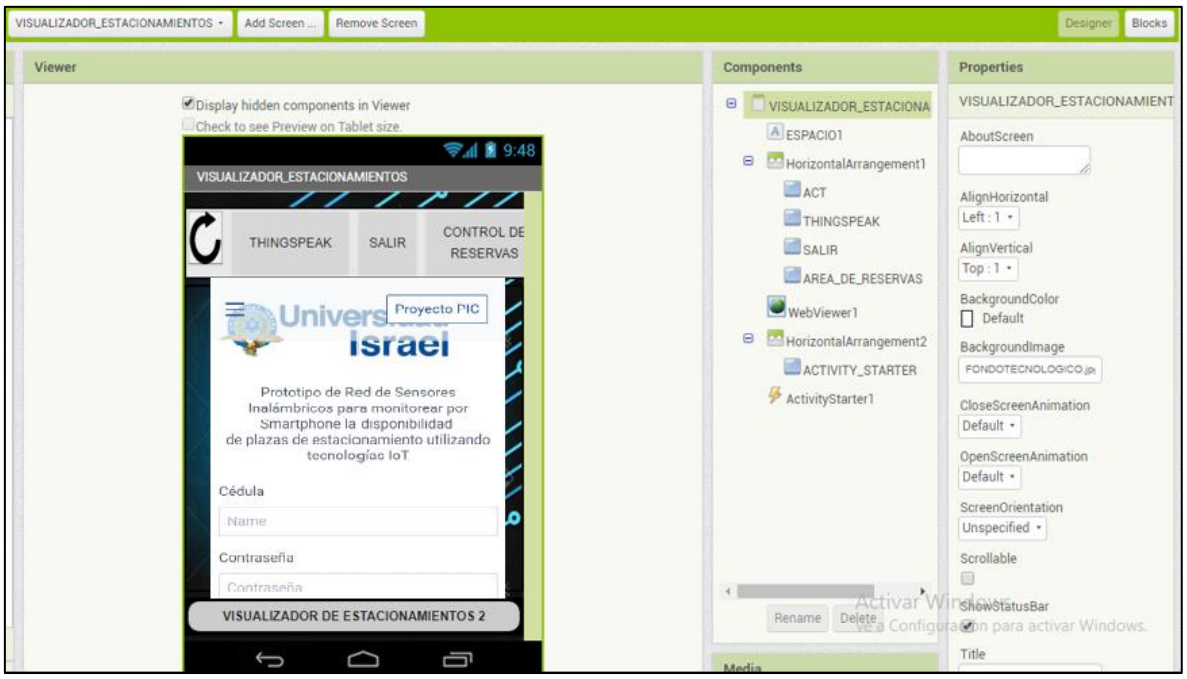


Figura 3.34. Pantalla Visualizador de Estacionamientos

Fuente: Elaborado por el autor

3.7.3. Pantalla PASSWORD_2 y PASSWORD_3:

Son las pantallas de seguridad para una vez autenticadas las credenciales ingresadas se pueda acceder a los ambientes ThingSpeak, Control de Reservas y Visualizador de Estacionamientos:

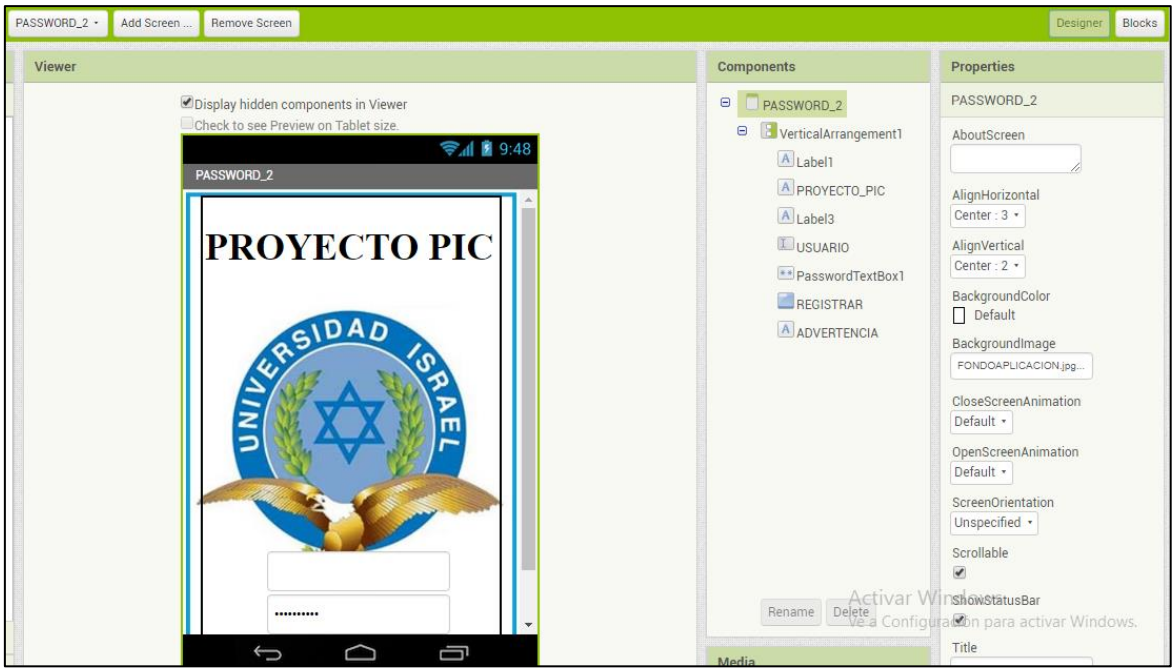


Figura 3.35. Designer Pantalla Password_2

Fuente: Elaborado por el autor

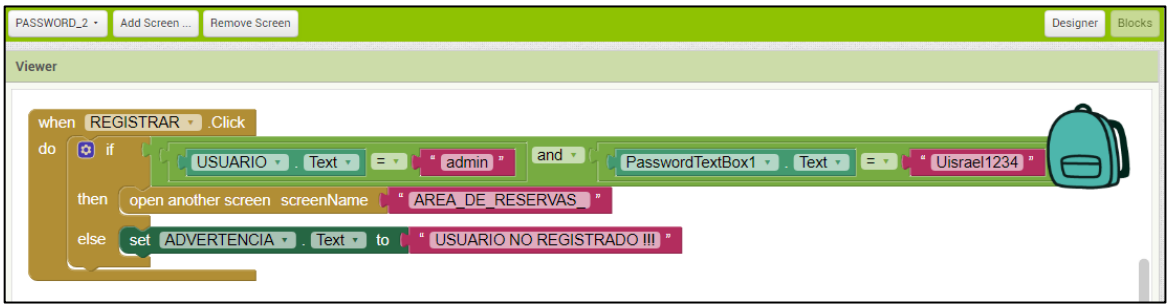


Figura 3.36. Block Diagram Pantalla Password_2

Fuente: Elaborado por el autor

3.7.4. Pantalla AREA_DE_RESERVAS:

Muestra los botones de control directo para reservar o liberar cada estacionamiento y seteo de los estados de los sensores y por medio de las API's (Application Programming Interfaces) de ThingSpeak que actúan desde esta aplicación hasta los sensores.

Cada API realiza un determinado cambio de estado y el mismo es identificado por un botón que lleva el nombre de la acción que realiza como Reservar Parqueadero, Liberar Parqueadero y Setear estado.

Designer:

Figura 3.37. Designer Pantalla AREA_RESERVAS

Fuente: Elaborado por el autor

Blocks:

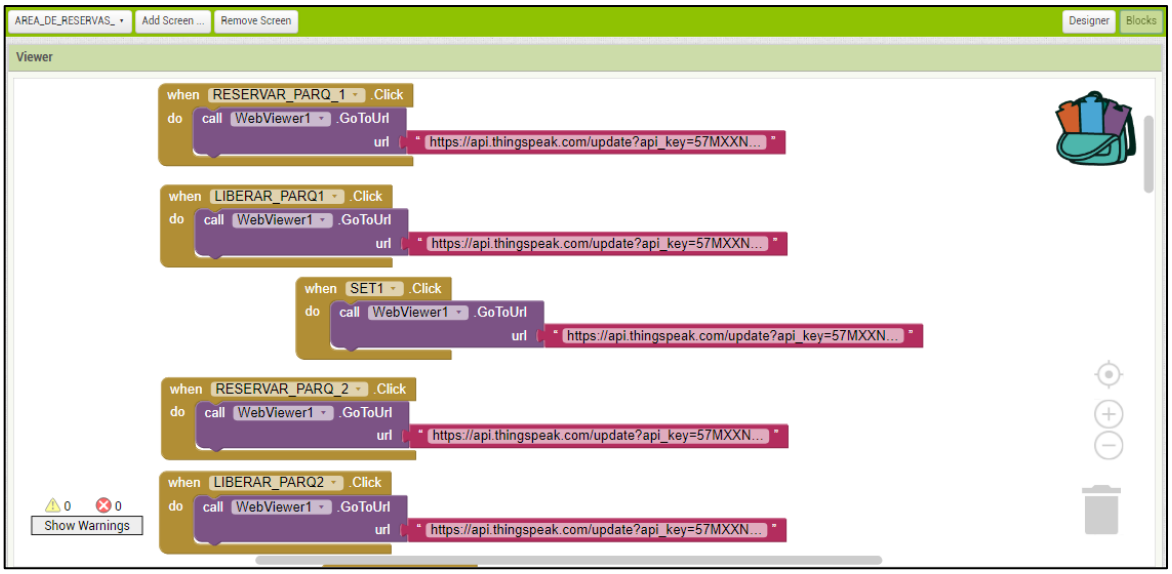


Figura 3.38. Block Diagram Pantalla AREA_RESERVAS

Fuente: Elaborado por el autor

3.7.5. Pantalla HOME THINGSPEAK:

Permite acceso a las gráficas originales de los estados de cada sensor (FIELDs) que representan los sitios de estacionamiento y control de las configuraciones permitidas por la licencia “40700874 STUDENT” en el servidor. Los botones configurados son:

- Control Reservas
- Salir (cerrar sesión)
- Adelante (>)Atrás (<)
- Actualizar página (flecha circular)

Designer

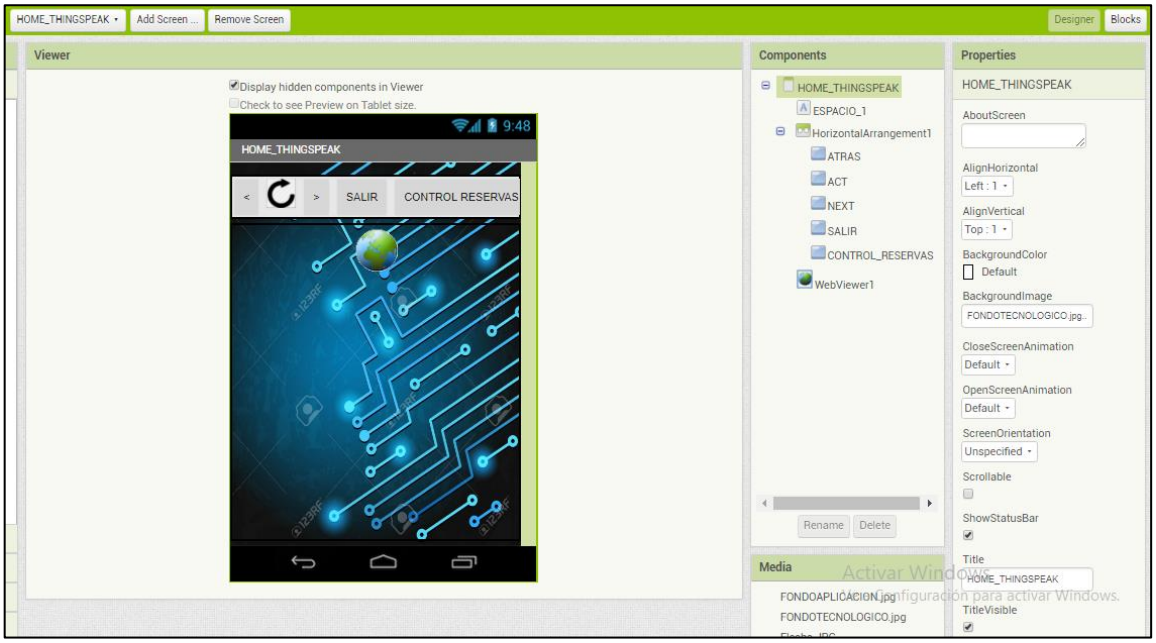


Figura 3.39. Designer Pantalla HOME_THINGSPEAK

Fuente: Elaborado por el autor

Blocks:

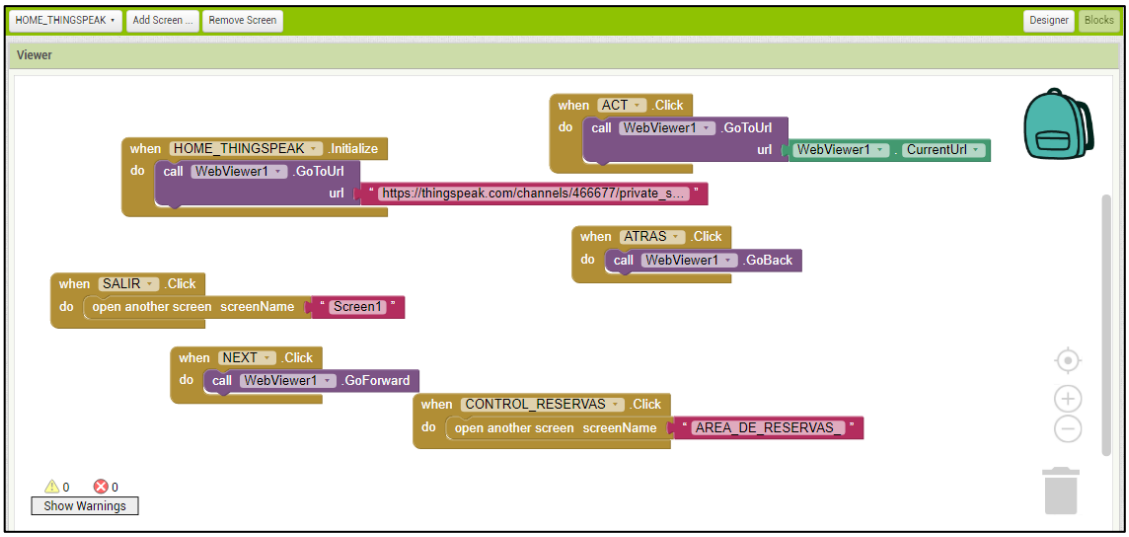


Figura 3.40. Block Diagram Pantalla HOME_THINGSPEAK

Fuente: Elaborado por el autor

3.7.6. Pantalla VISUALIZADOR_ESTACIONAMIENTOS:

En esta parte se muestra el estado de ocupación de los estacionamientos de una manera más comprensible para el usuario común. Es decir; si por ejemplo para la gráfica del Field 1 (Parqueadero 1) el estado “0” representa a estado libre; en esta pantalla en cambio aparecerá la palabra disponible en color verde y para la gráfica del Field 1 el estado “1” representa a estado ocupado o reservado en cambio en esta pantalla aparecerá un auto con el botón estadísticas en color rojo.

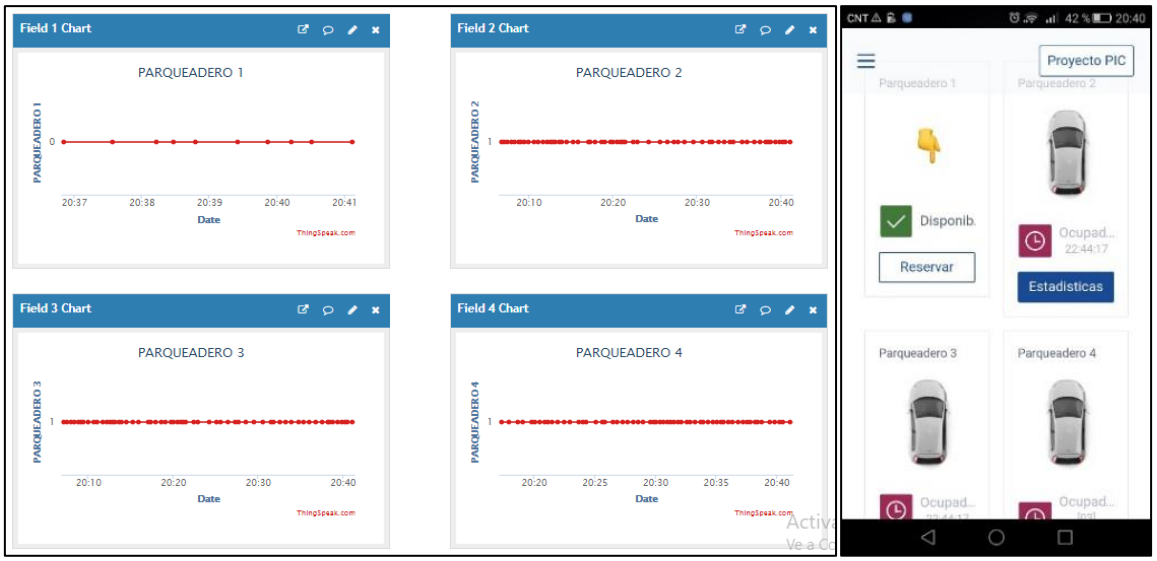


Figura 3.41. Pantalla Visualización de Estacionamientos

Fuente: Elaborado por el autor

Designer

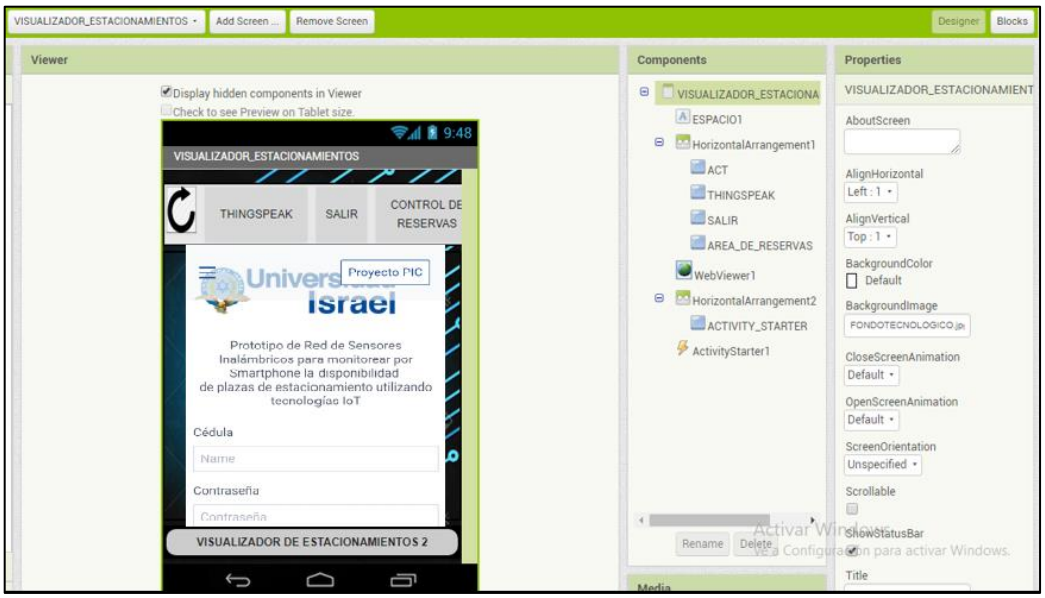


Figura 3.42. Designer Pantalla Visualizador de estacionamientos

Fuente: Elaborado por el autor

Blocks

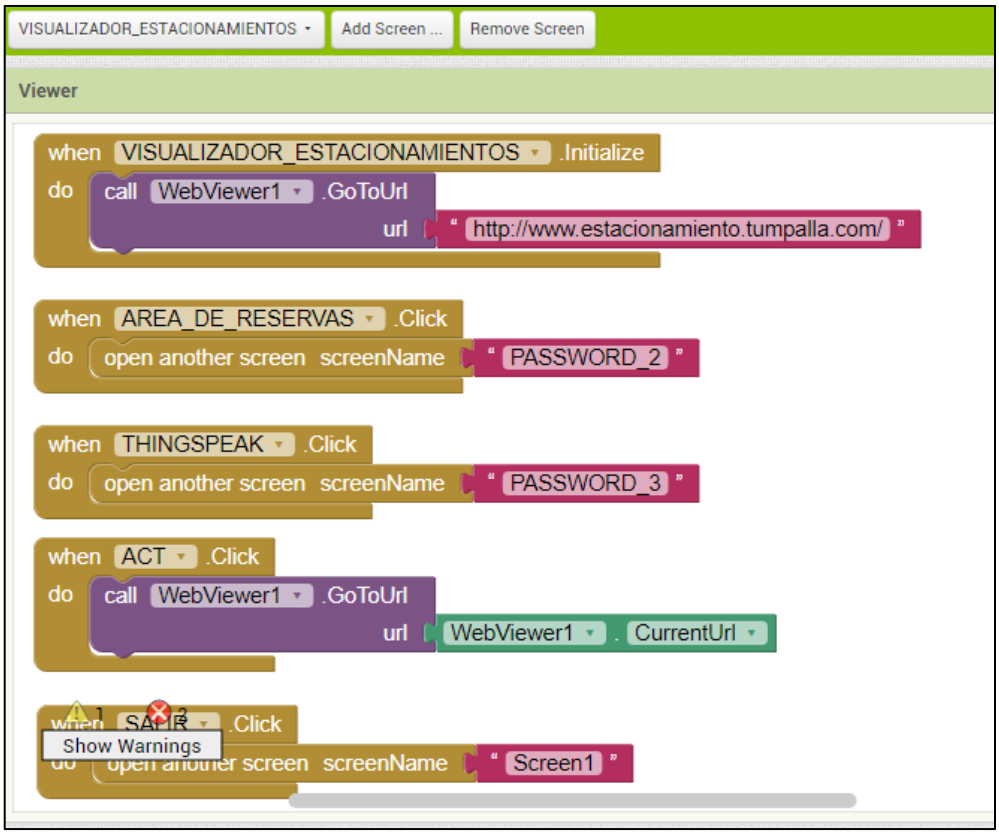


Figura 3.43. Block Diagram Visualizador de Estacionamientos

Fuente: Elaborado por el autor

3.8. Implementación del prototipo en el parqueadero de prueba

El funcionamiento del prototipo fue probado en las instalaciones del estacionamiento del conjunto habitacional “Camino Real” ubicado en el Centro Histórico de la ciudad de Quito. Las instalaciones se muestran en la siguiente figura:

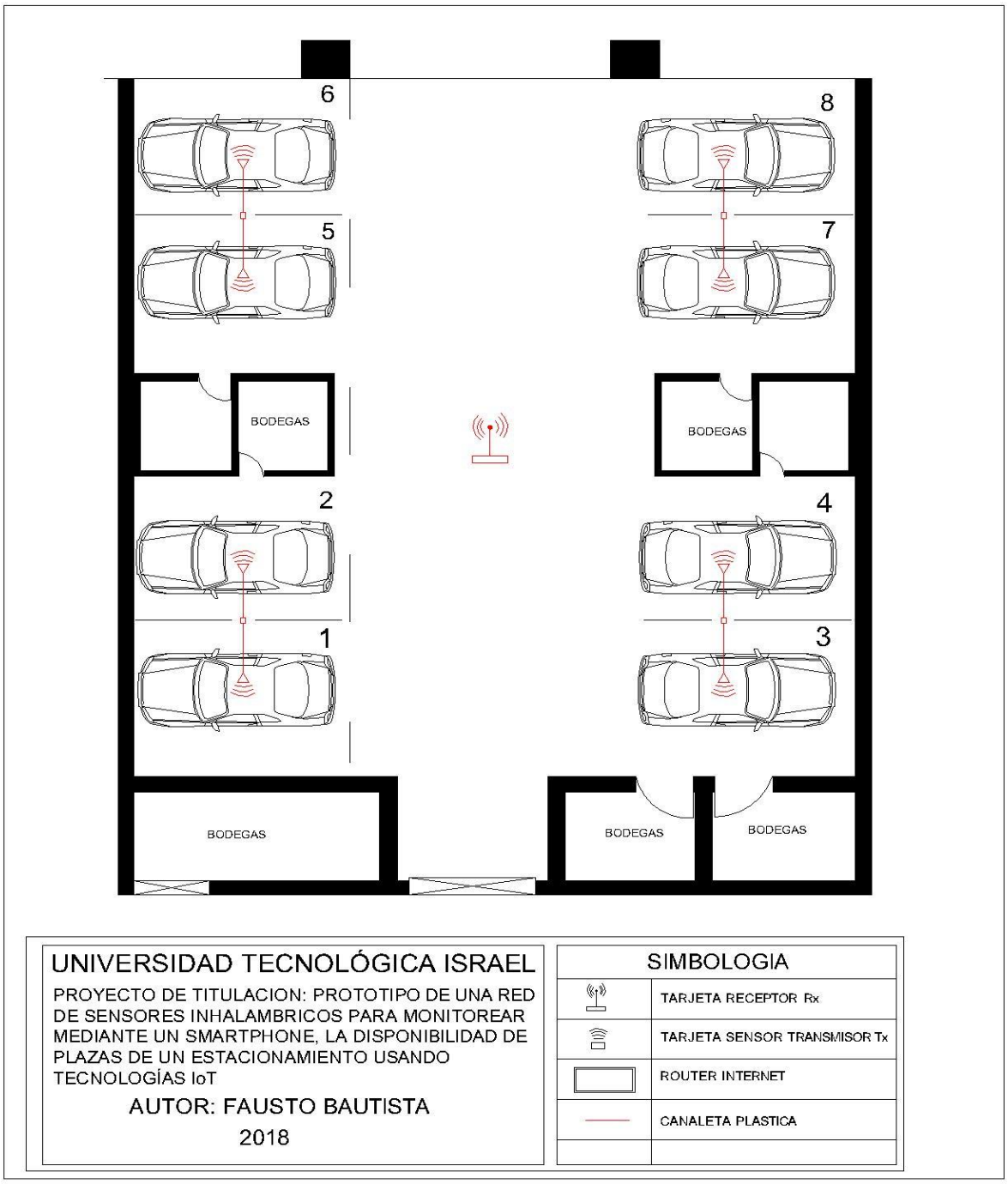


Figura 3.44. Plano del Parqueadero de Pruebas

Fuente: Elaborado por el autor

3.8.1. Instalación de las tomas de alimentación:

Para energizar a los sensores se utilizó el cableado ya existente en la misma infraestructura tomando únicamente el voltaje fase-neutro (110-120V).

Se ubicaron retornos para poder resetear a los sensores en caso de requerirlo. Para esto se pudo utilizar la misma tubería eléctrica ya existente para los sensores de iluminación y con la ayuda de la sonda plástica para guiar los retornos a través del tubo.

Desmontando momentáneamente los sensores de movimiento permitimos el paso de la sonda guía y el cable para retornos.

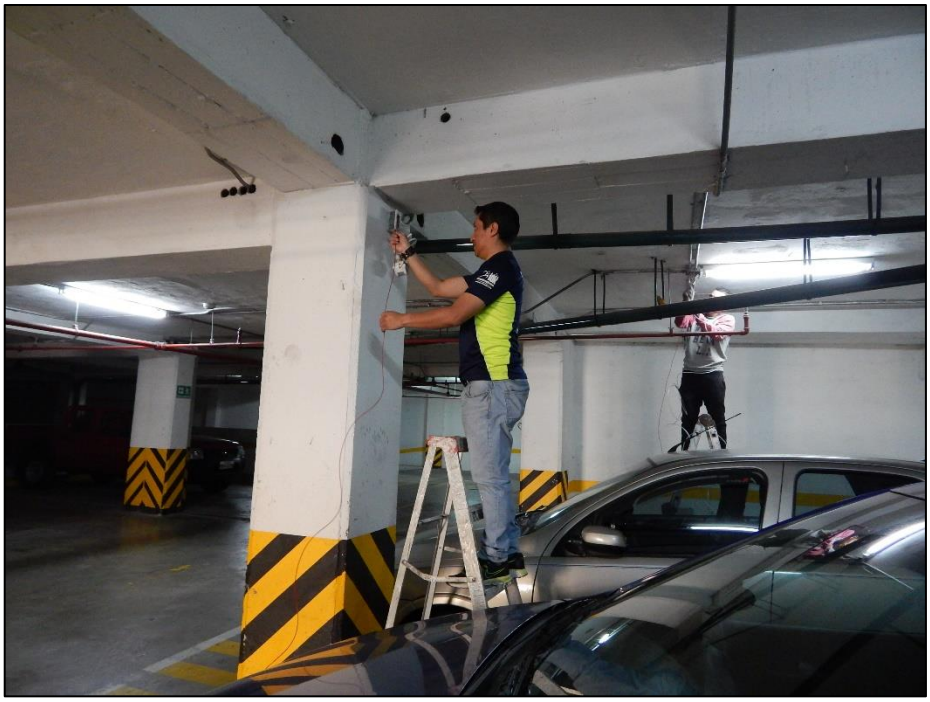


Figura 3.45. Pasando la sonda guía por la tubería de los sensores de movimiento

Fuente: Elaborado por el autor

3.8.2. Ubicación de la canaleta plástica 20x12

Se procedió a pegar la canaleta en la superficie del techo, para luego reforzarla con tornillos una longitud aproximada de 2mts a cada lado del cajetín central hasta llegar al centro de cada estacionamiento, luego se procedió a pasar los cables de alimentación y a colocar las tapas correspondientes como se indica en la siguiente figura:

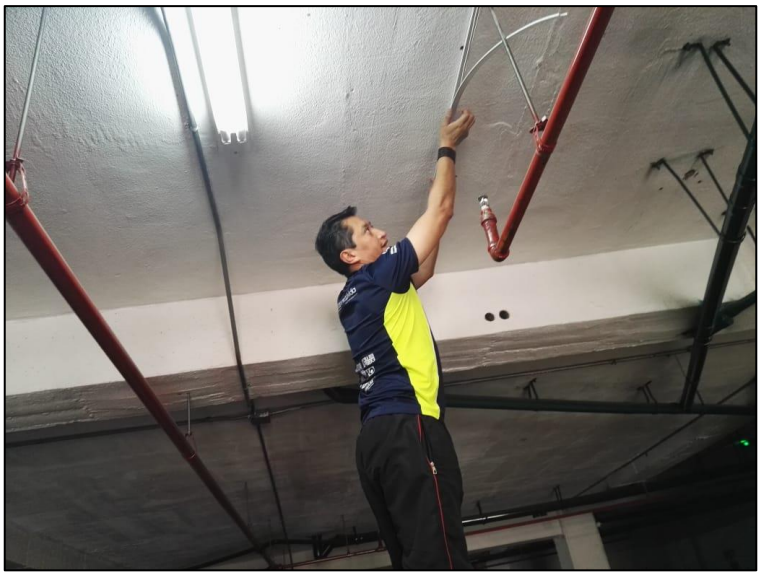


Figura 3.46. Instalación de canaleta plástica

Fuente: Elaborado por el autor

Se procedió a colocar los respectivos taco-fischer y se sujetó las cajas de sensores con tornillos bien sujetos al techo para luego proceder a conectar la alimentación de 110-120VAC quedando finalmente energizados.

Figura 3.47. Sujeción y alimentación de los Sensores Tx (A)

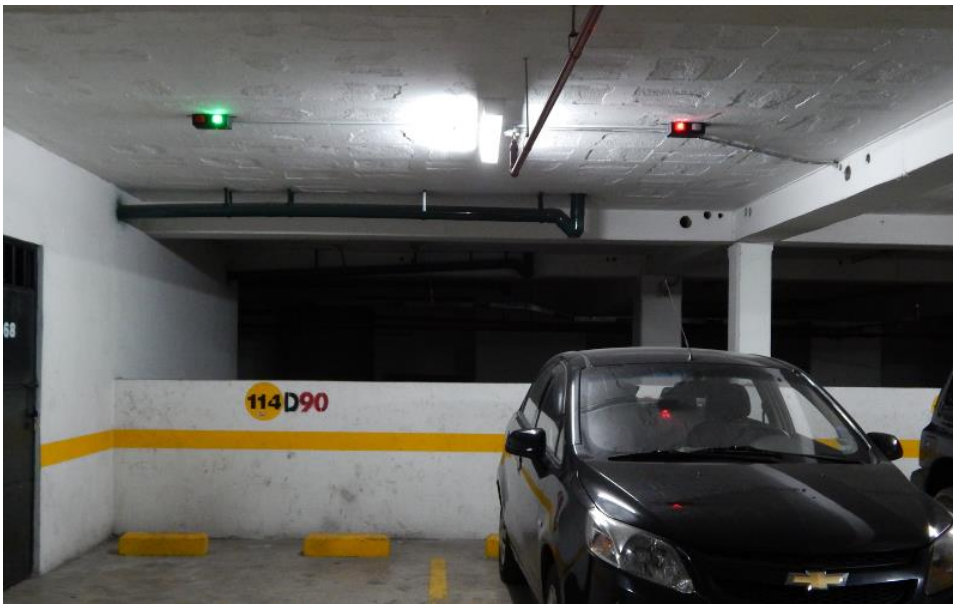


Figura 3.48. Sujeción y alimentación de los Sensores Tx (B)

Fuente: Elaborado por el autor

De igual manera se colocó la tarjeta Receptor Rx con la parte de control de Estados en una caja de proyectos sujetas de igual manera con postes metálicos a la base de la caja y todo el armazón con soportes necesarios para poder ser sujetado con tornillos como se indica a continuación.

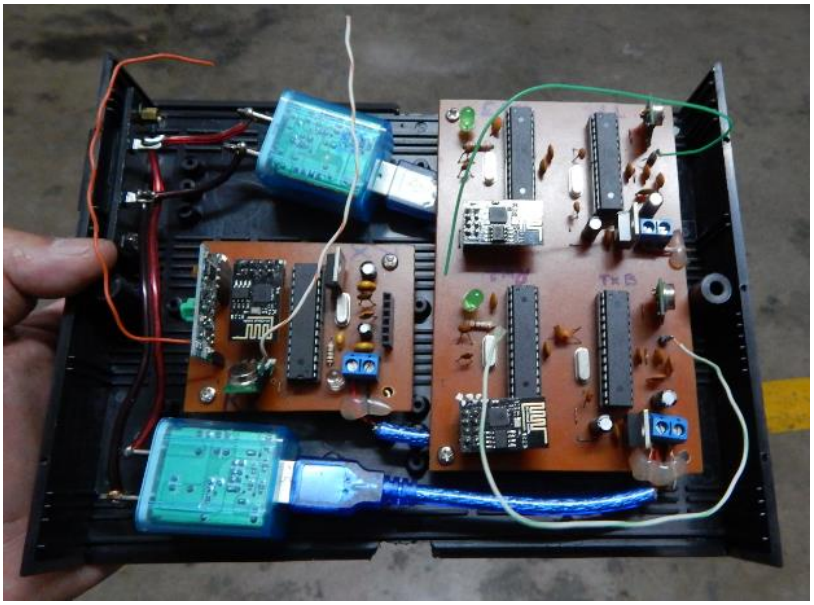


Figura 3.49. Encapsulamiento Tarjeta Receptor Rx en la caja de proyecto

Fuente: Elaborado por el autor

Figura 3.50. Colocación Tarjeta Receptor RX y control de estados en el techo

Fuente: Elaborado por el autor

Para determinar el mejor sitio de ubicación de la caja módulo Receptor Rx se probó en 3 sitios estratégicos como se indica a continuación:

a) En el centro de la red

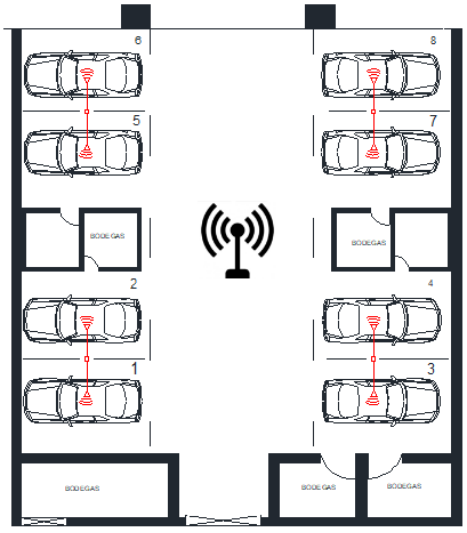


Figura 3.51. Pruebas con la Tarjeta Receptor RX en el centro de la red

Fuente: Elaborado por el autor

b) En el extremo superior de la red:

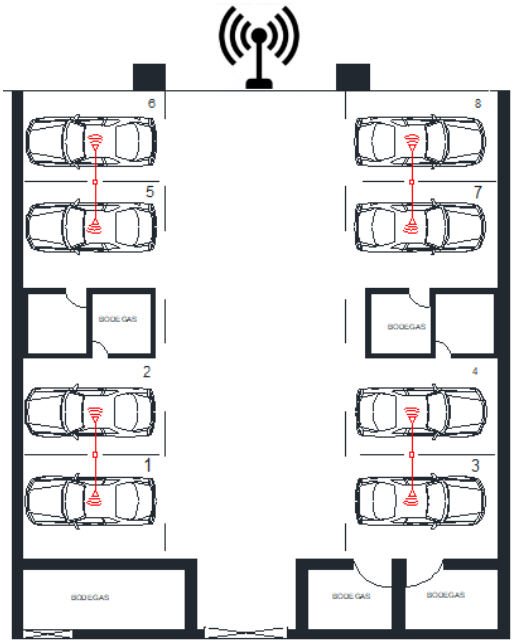


Figura 3.52. Pruebas con la Tarjeta Receptor RX en la parte superior de la red

Fuente: Elaborado por el autor

c) En el extremo inferior de la red

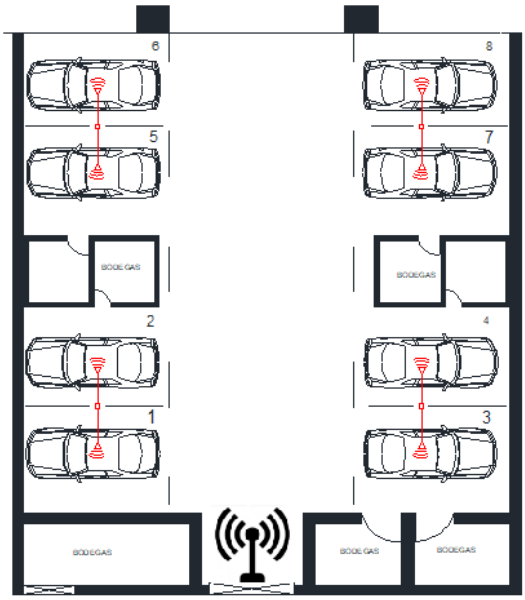


Figura 3.53. Pruebas con la Tarjeta Receptor RX en la parte inferior de la red

Fuente: Elaborado por el autor

Los mejores resultados se obtuvieron con la opción a; es decir, el módulo Receptor Rx en el centro de la red por lo que el dispositivo fue colocado en la viga central de los parqueaderos como se muestra en la siguiente figura:

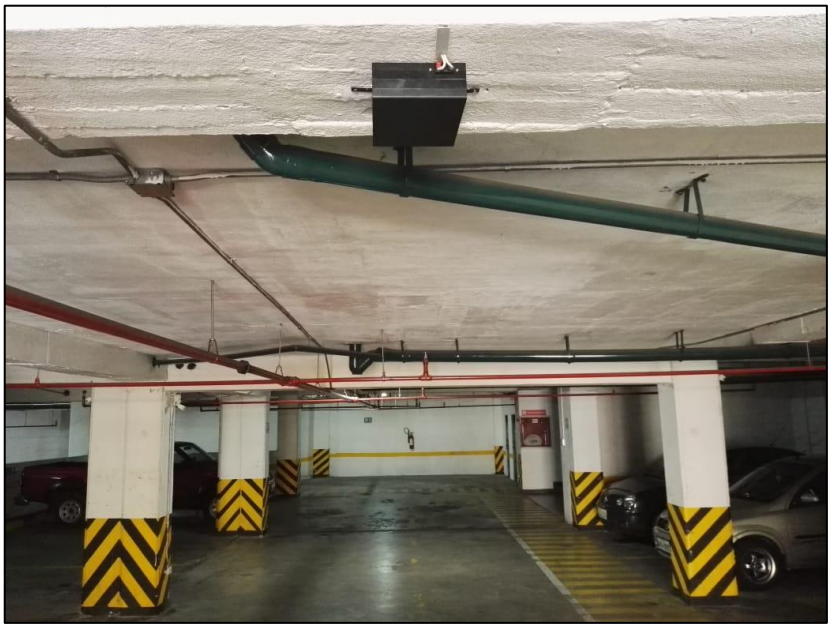


Figura 3.54. Colocación de la Tarjeta Receptor en el centro del parqueadero

Fuente: Elaborado por el autor

3.9. Presupuesto

a) Presupuesto para la parte del dispositivo

Tabla 3.5. Presupuesto Hardware de los dispositivos

PRESUPUESTO PIC PROTOTIPO DE RED INALÁMBRICA PARA MONITOREO POR SMARTPHONE				
ITEM	CANTIDAD	DESCRIPCION	V. UNITARIO	V. TOTAL
1	13	Integrado ATMEGA328	8,00	104
2	8	Sensor ultra sónico HC-SR04	5,00	40
3	9	Cristal 16MHz	0,74	6,66
4	18	Condensador 22pf/50V	0,15	2,7
5	20	Zócalo 14 pines	0,15	3
6	14	Condensador 104	0,15	2,1
7	9	Integrado LM7805	0,71	6,39
8	11	Integrado LM1117	1,42	15,62
9	20	Condensador 10uf 50V	0,19	3,8
10	11	Baquelita	0,90	9,9
11	9	Hoja de papel de transferencia	1,00	9
12	9	Cloruro férrico	0,80	7,2
13	10	Fuentes de poder	6,00	60
14	11	Módulo de RF 433 MHz	10,00	110
15	3	Integrado ESP8266	7,00	21
16	9	Cables USB	2,00	18
17	8	Cajas plásticas para proyectos 19x10x6 cm	5,00	40
18	1	Caja plástica para proyectos 20x20x6 cm	10,00	10
19	10	Portajacks tipo hembra	1,50	15
20	30	Postes metálicos con tornillos	1,00	30
				514,37

Fuente: Elaborado por el autor

b) Presupuesto para la parte de la implementación

Tabla 3.6. Presupuesto para la parte de la implementación

PRESUPUESTO PIC PROTOTIPO DE RED INALÁMBRICA PARA MONITOREO POR SMARTPHONE				
ITEM	CANTIDAD	DESCRIPCION	V. UNITARIO	V. TOTAL
1	30	TORNILLO COLEPATO	0,02	0,6
2	30	TACO FISHER #6	0,01	0,3
3	1	BROCA CONCRETO 1/4	1	1
4	4	SWITCH POWER VETO 110V SOBREPUESTO	1,5	6
5	11	CANALETA PLASTICA 20*12	0,9	9,9
6	70	CABLE SOLIDO #14	0,4	28
7	1	BREAKER 32 AMP	4,5	4,5
				50,3

Fuente: Elaborado por el autor

Conclusiones

- Se estableció la red inalámbrica con módulos RF 433MHz, los cuales transmiten la información proveniente de los sensores ultrasónicos hacia el módulo ESP8266 y por medio de éste se sube la información al servidor IoT Thingspeak para que la información sea almacenada en su base de datos y a la vez sea accesible para su monitoreo por smartphones con sistema operativo Android.

- Se realizó la implementación del prototipo descrito en este documento en los estacionamientos del Conjunto Habitacional "Camino Real" ubicado en el centro histórico de Quito, en donde se alcanzó el objetivo de poder observar desde la aplicación desarrollada, el status de ocupación de los 8 estacionamientos asignados para las pruebas en tiempo real.

- El subir la información local generada en la red inalámbrica del estacionamiento, fue posible gracias al protocolo 802.11 (WiFi) el cual está incorporado en el módulo ESP8266, motivo por el cual este dispositivo se hace muy útil para desarrollo de aplicaciones de Internet de las Cosa IoT y su uso vaya en aumento.

- La comunicación de la tarjeta Receptor Rx con el servidor ThingSpeak, se la realizó a través de la nube de Internet gracias a que el módulo ESP8266 tiene incorporado el stack de protocolos TCP/IP, que le permite a la Tarjeta Receptor Rx recopilar por un lado la información de los sensores de cada parqueadero hasta el microcontrolador ATMEGA328 y por otro lado subir la misma hacia la nube de Internet.

- Se desarrolló la aplicación Android mediante MIT APP INVENTOR para la parte de control de los estados, mediante los APIs de ThingSpeak a través de botones y pantallas que describen la función que realizan. También se agregó a la app una pantalla de visualización más amigable para cualquier usuario, la misma tiene las funciones básicas de monitoreo, reserva y registro de la base de datos.

- Se implementó la tarjeta electrónica que constituye el hardware con el que el microcontrolador ATMEGA328 se interconecta con los otros dispositivos propuestos y que lo hace compatible con el ambiente de desarrollo IDE de Arduino para facilitar el desarrollo del código de programación utilizado.

Recomendaciones

- Tener mucho cuidado en las conexiones auxiliares que se debe realizar para la configuración del módulo ESP8266, sobre todo en la alimentación de 3VDC de entrada, ya que al usar una tarjeta Arduino se puede conectar erróneamente con el pin de 5VDC y quemar al mismo.

- Se debe agregar a los módulos RF, antenas de longitud suficiente para mitigar afectaciones en la red por pérdidas de los enlaces o atenuaciones y que para este caso fueron de 15 cm para cubrir un espacio físico de 20 mts x 15 mts y de ser necesario colocar las antenas exteriormente, realizarlo procurando mantener la estética de los dispositivos fabricados

- El momento de crear usuarios, configurar credenciales robustas ya sea de cliente o de administrador para el acceso al servidor y a la aplicación, ya que al tener disponible esta información en la nube de Internet resulta ser una desventaja su mal uso ya sea por privacidad o seguridad.

- Se debe procurar utilizar la versión 1.6.12 del IDE Arduino para la carga del código de programación y no tener inconvenientes de compatibilidad de las librerías utilizadas con otras versiones de IDE. Los repositorios generalmente indican las versiones en la fueron probadas dichas librerías.

- Procurar que los estacionamientos seleccionados sean los de menor uso posible para poder facilitar los trabajos de implementación, pero sobre todo para dar facilidades en el momento de realizar las pruebas y no causar retrasos innecesarios por este motivo ni incomodar a sus ocupantes.

REFERENCIAS BIBLIOGRÁFICAS:

BBVAOPEN4U. (2016). *https://bbvaopen4u.com*. Obtenido de https://bbvaopen4u.com/es/actualidad/apis-para-el-internet-de-las-cosas-thingspeak-pachube-y-fitbit

Llamas, L. (2016). *https://www.luisllamas.es*. Obtenido de https://www.luisllamas.es/comunicacion-inalambrica-en-arduino-con-modulos-rf-433mhz/

MANOSALOSCABLES. (2016). *https://comprendiendoarduino.wordpress.com.* Obtenido de https://comprendiendoarduino.wordpress.com/2016/02/01/armar-un-arduino-uno-en-una-protoboard/

PROMETEC. (2017). *https://www.prometec.net.* Obtenido de https://www.prometec.net/esp8266/

Rico, E. (2018). *http://www.ermesh.com.* Obtenido de http://www.ermesh.com/anatomia-objetos-conectados-internet-de-las-cosas/#more-256

Scharler, H. (2018). *http://nothans.com.* Obtenido de http://nothans.com/measure-wi-fi-signal-levels-with-the-esp8266-and-thingspeak

TalosElectronics. (2018). *https://www.taloselectronics.com. Obtenido* de https://www.taloselectronics.com/products/sensor-ultrasonico-hc-sr04

Tecmikro. (2018). *http://programarpicenc.com*. Obtenido de http://programarpicenc.com/articulos/radiofrecuencia-sistema-tx-rx-a-433mhz/

Torres, J. (2014). *https://hipertextual.com*. Obtenido de https://hipertextual.com/archivo/2014/10/internet-cosas/

Vega, R. (2016). *https://ricveal.com.* Obtenido de https://ricveal.com/blog/arduino-esp8266/

Ventura, V. (2016). *https://polaridad.es.* Obtenido de https://polaridad.es/esp8266-modulo-wifi-elegir-caracteristicas/

ANEXOS

ANEXO A

CÓDIGO DE PROGRAMACIÓN TARJETA SENSOR Tx

```
#include <VirtualWire.h>
#include <EEPROM.h>
int ledv = 2;   //  The on-board Arduion LED
int ledr = 3;
int dato=0;
//int x=0;
//int y=0;
int g;
int h;
int proceso=0;
long distancia;//Variable para almacenar el valor de la distancia
long tiempo;

int state1=0;
// this runs once
byte message[VW_MAX_MESSAGE_LEN];
byte messageLength = VW_MAX_MESSAGE_LEN;


void setup() {


  // enable debug serial
  Serial.begin(9600);

  pinMode(ledr,OUTPUT);
  pinMode(ledv,OUTPUT);
  digitalWrite(ledr, LOW);
  digitalWrite(ledv, LOW);

  pinMode(8, OUTPUT); /*activación del pin 9 como salida: para el pulso ultrasónico*/
  pinMode(9, INPUT);

  Serial.write("Sistema encendido\n");
  //EEPROM.write(10, 0);
  vw_setup(2000);
  vw_rx_start();
  proceso = EEPROM.read(10);
}


// the loop
void loop() {

 if(proceso==0){
  medida1();
  analisis();
}
```

```
 if(proceso==1){
 digitalWrite(ledr, HIGH);
 digitalWrite(ledv, LOW);
 medida1();
 analisis2();
 digitalWrite(ledr, LOW);
 digitalWrite(ledv, LOW);
 delay(50);

 }

if(proceso==2){
  delay(200);
     if(state1==1){
       send("proca");

       asm volatile ("  jmp 0");
     }

       if(state1==0){

        send("procb");

        asm volatile ("  jmp 0");
       }
   proceso=h;

}


if (vw_get_message(message, &messageLength))
  {
     if(comparar("parq1") == 0){
      delay(10);
      h=proceso;
      proceso=2;
      for(g=0;g<1;g++){
      digitalWrite(ledv,HIGH);
      delay(50);
      digitalWrite(ledv,LOW);
      delay(50);
        }

     }


     else if(comparar("pediq") == 0){
      delay(10);
      proceso=1;
      for(g=0;g<1;g++){
      digitalWrite(ledv,HIGH);
      delay(50);
```

```
      digitalWrite(ledv,LOW);
      delay(50);
        }
     EEPROM.write(10, proceso);
     asm volatile ("  jmp 0");
     }

          else if(comparar("pedia") == 0){
      delay(10);
      proceso=0;
      for(g=0;g<1;g++){
      digitalWrite(ledv,HIGH);
      delay(50);
      digitalWrite(ledv,LOW);
      delay(50);
        }
     EEPROM.write(10, proceso);
     asm volatile ("  jmp 0");
     }

  }

}

void medida1(){
  digitalWrite(8,LOW); /* Por cuestión de estabilización del sensor*/
  delayMicroseconds(5);
  digitalWrite(8, HIGH); /* envío del pulso ultrasónico*/
  delayMicroseconds(10);//pausa de 10 microsegundos
  tiempo=pulseIn(9, HIGH); /* Función para medir la longitud del pulso entrante. Mide el
tiempo que transcurrido entre el envío
  del pulso ultrasónico y cuando el sensor recibe el rebote, es decir: desde que el pin 8 empieza
a recibir el rebote, HIGH, hasta que
  deja de hacerlo, LOW, la longitud del pulso entrante*/
  distancia= int(0.017*tiempo); /*fórmula para calcular la distancia obteniendo un valor
entero*/
  /*Monitorización en centímetros por el monitor serial*/
  //Serial.println("Distancia ");//Imprime en pantalla la palabra "Distancia"
  //Serial.println(distancia);//Imprime el valor de distancia en centímetros
  //Serial.println(" cm");
  delay(100);

}

void analisis(){
  if (distancia <= 130){
   //send("procesoa");
   //Serial.println("procesoa");
   state1=1;
   digitalWrite(ledr, HIGH);
   digitalWrite(ledv, LOW);
}
```

```
  if (distancia > 130){
  //send("procesob");
   //Serial.println("procesob");
    state1=0;
   digitalWrite(ledr, LOW);
   digitalWrite(ledv, HIGH);

}
}

void pausa2(){
delay (100);
}

void analisis2(){
  pausa2();
  state1=1;
  // send("procesoa");

  if (distancia <= 130){
  proceso=0;
  EEPROM.write(10, proceso);
  }

}

//Funcion para enviar el mensaje
void send (char *message)
{
  vw_send((uint8_t *)message, strlen(message)); //Envia el mensaje
  vw_wait_tx(); //Espera hasta que se haya acabado de transmitir todo

  Serial.println(message); //Muestra el mensaje por Serial
}

char comparar(char* cadena) {
  //Esta funcion compara el string cadena con el mensaje recibido.
  //Si son iguales, devuelve 1. Si no, devuelve 0.

  for(int i = 0; i<messageLength; i++)
  {
    if(message[i] != cadena[i])
    {
      return 1;
    }
  }

    return 0;
}
```

CÓDIGO DE PROGRAMACIÓN RECEPTOR Rx

```
#include <SoftwareSerial.h>
#include <stdlib.h>
#include <VirtualWire.h>
//Creamos un mensaje
//La constante VW_MAX_MESSAGE_LEN viene definida en la libreria
byte message[VW_MAX_MESSAGE_LEN];
byte messageLength = VW_MAX_MESSAGE_LEN;


int sensor1State=0;


//  Variables
int PulseSensorPurplePin = 0;       // Pulse Sensor PURPLE WIRE connected to ANALOG PIN 0
int ledr = 2;   //  The on-board Arduion LED
int ledv = 3;
int Signal;              // holds the incoming raw data. Signal value can range from 0-1024
int Threshold = 550;          // Determine which Signal to "count as a beat", and which to ingore.
int dato=0;
int BPM;
// replace with your channel's thingspeak API key
String apiKey = "HG0ZPK7ADBT9FG8T";

SoftwareSerial Serial2x(6,5);//Declaramos el pin 5 rx y 6 tx

int i=1;
const int sensorPin= A0;
// On Arduino:  0 - 1023 maps to 0 - 5 volts
#define VOLTAGE_MAX 5.0
#define VOLTAGE_MAXCOUNTS 1023.0



long distancia;//Variable para almacenar el valor de la distancia
long tiempo;

String state1;
String state2;
String state3;
String state4;
String state5;
String state6;
String state7;
String state8;
int g;
int pedido=1;

int h;
// this runs once
void setup() {
```

```
 // enable debug serial
 Serial.begin(9600);
 // enable software serial
 Serial2x.begin(9600);//Iniciamos el puerto serie del gps

 // reset ESP8266
 //Serial2.println("AT+RST");
 pinMode(ledr,OUTPUT);
 pinMode(ledv,OUTPUT);
 digitalWrite(ledr, LOW);
 digitalWrite(ledv, LOW);

 pinMode(9, OUTPUT); /*activación del pin 9 como salida: para el pulso ultrasónico*/
 pinMode(8, INPUT);

 Serial.write("Sistema encendido\n");
 vw_setup(2000);
 vw_rx_start();

 }


// the loop
void loop() {
  if(pedido==1){
    send("parq1");
    for(g=0;g<1;g++){
    digitalWrite(ledv,HIGH);
    delay(50);
    digitalWrite(ledv,LOW);
    delay(50);
  }
  pedido=2;

 }

 if(pedido==2){
  send("parq1");
  for(h=0;h<100;h++){
   if (vw_get_message(message, &messageLength))
  {
      if(comparar("proca") == 0){
       Serial.println("proca");
       state1="1";
       pausa();
       for(g=0;g<1;g++){
       digitalWrite(ledv,HIGH);
       delay(10);
       digitalWrite(ledv,LOW);
       delay(10);
```

```
      }
     pedido=3;
     h=1500;
     delay(300);
    }
    else if(comparar("procb") == 0)
    {
     Serial.println("procb");
     state1="0";
     pausa();
     for(g=0;g<1;g++){
     digitalWrite(ledv,HIGH);
     delay(10);
     digitalWrite(ledv,LOW);
     delay(10);
      }
     pedido=3;
     h=1500;
     delay(300);
    }
 }
 delay(10);
}

}



  if(pedido==3){
  send("parq2");
  for(g=0;g<1;g++){
  digitalWrite(ledv,HIGH);
  delay(50);
  digitalWrite(ledv,LOW);
  delay(50);
 }
 pedido=4;

}

if(pedido==4){
  send("parq2");
  for(h=0;h<200;h++){
  if (vw_get_message(message, &messageLength))
 {
    if(comparar("procc") == 0){
     Serial.println("procc");
     state2="1";
     pausa();
     for(g=0;g<1;g++){
     digitalWrite(ledv,HIGH);
```

```
     delay(10);
     digitalWrite(ledv,LOW);
     delay(10);
      }
     pedido=5;
     h=1500;
     delay(300);
    }
    else if(comparar("procd") == 0)
    {
     Serial.println("procd");
     state2="0";
     pausa();
     for(g=0;g<1;g++){
     digitalWrite(ledv,HIGH);
     delay(10);
     digitalWrite(ledv,LOW);
     delay(10);
      }
     pedido=5;
     h=1500;
     delay(300);
    }
 }
 delay(10);
}

}
```

```
  if(pedido==5){
  send("parq3");
  for(g=0;g<1;g++){
  digitalWrite(ledv,HIGH);
  delay(50);
  digitalWrite(ledv,LOW);
  delay(50);
 }
 pedido=6;

}

if(pedido==6){
  send("parq3");
  for(h=0;h<200;h++){
  if (vw_get_message(message, &messageLength))
 {
    if(comparar("proce") == 0){
```

```
     Serial.println("proce");
     state3="1";
     pausa();
     for(g=0;g<1;g++){
     digitalWrite(ledv,HIGH);
     delay(10);
     digitalWrite(ledv,LOW);
     delay(10);
      }
     pedido=7;
     h=1500;
     delay(300);
    }
    else if(comparar("procf") == 0)
    {
     Serial.println("procf");
     state3="0";
     pausa();
     for(g=0;g<1;g++){
     digitalWrite(ledv,HIGH);
     delay(10);
     digitalWrite(ledv,LOW);
     delay(10);
      }
     pedido=7;
     h=1500;
     delay(300);
    }
 }
 delay(10);
}

}
```

```
if(pedido==7){
   send("parq4");
   for(g=0;g<1;g++){
   digitalWrite(ledv,HIGH);
   delay(50);
   digitalWrite(ledv,LOW);
   delay(50);
 }
 pedido=8;

}
```

```
if(pedido==8){
  send("parq4");
  for(h=0;h<200;h++){
  if (vw_get_message(message, &messageLength))
 {
     if(comparar("procg") == 0){
      Serial.println("procg");
      state4="1";
      pausa();
      for(g=0;g<1;g++){
      digitalWrite(ledv,HIGH);
      delay(10);
      digitalWrite(ledv,LOW);
      delay(10);
       }
      pedido=9;
      h=1500;
      delay(300);
     }
     else if(comparar("proch") == 0)
     {
      Serial.println("proch");
      state4="0";
      pausa();
      for(g=0;g<1;g++){
      digitalWrite(ledv,HIGH);
      delay(10);
      digitalWrite(ledv,LOW);
      delay(10);
       }
      pedido=9;
      h=1500;
      delay(300);
     }
 }
 delay(10);
}

}
```

```
if(pedido==9){
   send("parq5");
   for(g=0;g<1;g++){
   digitalWrite(ledv,HIGH);
   delay(50);
   digitalWrite(ledv,LOW);
   delay(50);
```

```
 }
 pedido=10;

}

if(pedido==10){
  send("parq5");
  for(h=0;h<200;h++){
  if (vw_get_message(message, &messageLength))
 {
    if(comparar("proci") == 0){
     Serial.println("proci");
     state5="1";
     pausa();
     for(g=0;g<1;g++){
     digitalWrite(ledv,HIGH);
     delay(50);
     digitalWrite(ledv,LOW);
     delay(50);
      }
     pedido=11;
     h=1500;
     delay(300);
    }
    else if(comparar("procj") == 0)
    {
     Serial.println("procj");
     state5="0";
     pausa();
     for(g=0;g<1;g++){
     digitalWrite(ledv,HIGH);
     delay(50);
     digitalWrite(ledv,LOW);
     delay(50);
      }
     pedido=11;
     h=1500;
     delay(300);
    }
 }
 delay(10);
}

}



if(pedido==11){
   send("parq6");
   for(g=0;g<1;g++){
   digitalWrite(ledv,HIGH);
   delay(50);
```

```
   digitalWrite(ledv,LOW);
   delay(50);
 }
 pedido=12;

}

if(pedido==12){
  send("parq6");
  for(h=0;h<200;h++){
  if (vw_get_message(message, &messageLength))
 {
     if(comparar("prock") == 0){
      Serial.println("prock");
      state6="1";
      pausa();
      for(g=0;g<1;g++){
      digitalWrite(ledv,HIGH);
      delay(50);
      digitalWrite(ledv,LOW);
      delay(50);
       }
      pedido=13;
      h=1500;
      delay(300);
     }
     else if(comparar("procl") == 0)
     {
      Serial.println("procl");
      state6="0";
      pausa();
      for(g=0;g<1;g++){
      digitalWrite(ledv,HIGH);
      delay(50);
      digitalWrite(ledv,LOW);
      delay(50);
       }
      pedido=13;
      h=1500;
      delay(300);
     }
 }
 delay(10);
}

}
```

```
if(pedido==13){
  send("parq7");
  for(g=0;g<1;g++){
  digitalWrite(ledv,HIGH);
  delay(50);
  digitalWrite(ledv,LOW);
  delay(50);
 }
 pedido=14;

}

if(pedido==14){
  send("parq7");
  for(h=0;h<200;h++){
  if (vw_get_message(message, &messageLength))
 {
    if(comparar("procm") == 0){
     Serial.println("procm");
     state7="1";
     pausa();
     for(g=0;g<1;g++){
     digitalWrite(ledv,HIGH);
     delay(50);
     digitalWrite(ledv,LOW);
     delay(50);
      }
     pedido=15;
     h=1500;
     delay(300);
    }
    else if(comparar("procn") == 0)
    {
     Serial.println("procn");
     state7="0";
     pausa();
     for(g=0;g<1;g++){
     digitalWrite(ledv,HIGH);
     delay(50);
     digitalWrite(ledv,LOW);
     delay(50);
      }
     pedido=15;
     h=1500;
     delay(300);
    }
 }
 delay(10);
}

}
```

```
if(pedido==15){
  send("parq8");
  for(g=0;g<1;g++){
  digitalWrite(ledv,HIGH);
  delay(50);
  digitalWrite(ledv,LOW);
  delay(50);
 }
 pedido=16;

}

if(pedido==16){
  send("parq8");
  for(h=0;h<200;h++){
  if (vw_get_message(message, &messageLength))
 {
    if(comparar("proco") == 0){
     Serial.println("proco");
     state8="1";
     pausa();
     for(g=0;g<1;g++){
     digitalWrite(ledv,HIGH);
     delay(50);
     digitalWrite(ledv,LOW);
     delay(50);
      }
     pedido=0;
     h=1500;
     delay(300);
    }
    else if(comparar("procp") == 0)
    {
     Serial.println("procp");
     state8="0";
     pausa();
     for(g=0;g<1;g++){
     digitalWrite(ledv,HIGH);
     delay(50);
     digitalWrite(ledv,LOW);
     delay(50);
      }
     pedido=0;
     h=1500;
     delay(300);
    }
 }
 delay(10);
}

}
```

```
if(pedido==0){
// TCP connection
String cmd = "AT+CIPSTART=\"TCP\",\"";
cmd += "184.106.153.149"; // api.thingspeak.com
cmd += "\",80";
Serial2x.println(cmd);
Serial.println(cmd);

if(Serial2x.find("Error")){
  Serial.println("AT+CIPSTART error");
  return;
}

// prepare GET string
String getStr = "GET /update?api_key=";
getStr += apiKey;

getStr +="&field1=";
getStr += String(state1);
getStr +="&field2=";
getStr += String(state2);
getStr +="&field3=";
getStr += String(state3);
getStr +="&field4=";
getStr += String(state4);

getStr +="&field5=";
getStr += String(state5);
getStr +="&field6=";
getStr += String(state6);
getStr +="&field7=";
getStr += String(state7);
getStr +="&field8=";
getStr += String(state8);
getStr += "\r\n\r\n";

// send data length
cmd = "AT+CIPSEND=";
cmd += String(getStr.length());
Serial2x.println(cmd);
Serial.println(cmd);

if(Serial2x.find(">")){
  Serial2x.print(getStr);
  Serial.print(getStr);
}
else{
  Serial2x.println("AT+CIPCLOSE");
  // alert user
```

```
   Serial.println("AT+CIPCLOSE");
  }

  // thingspeak needs 15 sec delay between updates
  delay(2000);
  pedido=1;
  asm volatile ("  jmp 0");

}
}


void pausa(){
delay (5);
}


//Funcion para enviar el mensaje
void send (char *message)
{
  vw_send((uint8_t *)message, strlen(message)); //Envia el mensaje
  vw_wait_tx(); //Espera hasta que se haya acabado de transmitir todo

  Serial.println(message); //Muestra el mensaje por Serial
}



char comparar(char* cadena) {
  //Esta funcion compara el string cadena con el mensaje recibido.
  //Si son iguales, devuelve 1. Si no, devuelve 0.

  for(int i = 0; i<messageLength; i++)
  {
    if(message[i] != cadena[i])
    {
      return 1;
    }
  }

    return 0;
}
```

CÓDIGO DE PROGRAMACIÓN TARJETA CONTROLADORA DE ESTADOS

```
#include <Stepper.h>


//Probado en IDE 1.6.12
#include <SoftwareSerial.h>
#include <espduino.h>
#include <rest.h>


SoftwareSerial espPort(9, 10);
ESP esp(&espPort, &Serial, 8);
REST rest(&esp);
boolean wifiConnected = false;

const int relay1 =  17;
const int relay2 =  16;
const int relay3 =  15;
const int relay4 =  14;
int led =  3;


boolean data1 = false;
boolean data2 = false;
boolean data3 = false;
boolean data4 = false;
int loop_count = 0;


int g;

char response[266];
char buff[64];
String strId,strData,strCode;
String strData_Last1,strData_Last2,strData_Last3,strData_Last4;

void(* resetFunc) (void) = 0;

 void clearBuffer(void) {
      for (int i = 0;i<266;i++ ) {
        response[i]=0;
      }
}

void wifiCb(void* response)
{
  uint32_t status;
  RESPONSE res(response);

  if(res.getArgc() == 1) {
```

```
    res.popArgs((uint8_t*)&status, 4);
    if(status == STATION_GOT_IP) {
      Serial.println("TERHUBUNG KE WIFI");

      wifiConnected = true;
    } else {
      wifiConnected = false;
    }

  }
}

void setup() {

  pinMode(relay1, OUTPUT);
  pinMode(relay2, OUTPUT);
  pinMode(relay3, OUTPUT);
  pinMode(relay4, OUTPUT);

  pinMode(led, OUTPUT);
  digitalWrite(led,LOW);

  Serial.begin(115200);
  espPort.begin(19200);

  esp.enable();
  delay(500);
  esp.reset();
  delay(500);
  while(!esp.ready());

  Serial.println("ARDUINO: Setup client");
  if(!rest.begin("api.thingspeak.com")) {
    Serial.println("ARDUINO: Gagal Setup client");
    while(1);

  }

  Serial.println("ARDUINO: Conexion Wifi");
  esp.wifiCb.attach(&wifiCb);

  //Adjust the SSID and PASSWORD
  esp.wifiConnect("Claro_HERRERA0007716776","EDUARD81776856482223");
  Serial.println("ARDUINO: System Ready!");

  digitalWrite(relay1,LOW);
  digitalWrite(relay2,LOW);
  digitalWrite(relay3,LOW);
  digitalWrite(relay4,LOW);
}

void loop() {
```

```
loop_start:

  esp.process();
  if(wifiConnected) {

    char str_field1[6] , str_field2[6] , str_field3[6] , str_field4[6];
     sprintf(buff, "/channels/466685/fields/1/last/?key=47PNNYN89TBVWUTD"); //LAST DATA
FIELD 1
     Serial.println(buff);

     rest.get((const char*)buff);
     if(rest.getResponse(response, 266) == HTTP_STATUS_OK){

          strId = "";
          strData = "";
          strCode = "";
          getData();
        if (strId == "1" ){
          data1 = true;
          digitalWrite(led,HIGH);
          digitalWrite(relay1,HIGH);
          digitalWrite(relay2,LOW);
          digitalWrite(relay3,LOW);
          digitalWrite(relay4,LOW);
          delay(600);
          digitalWrite(led,LOW);
          digitalWrite(relay1,LOW);
          digitalWrite(relay2,LOW);
          digitalWrite(relay3,LOW);
          digitalWrite(relay4,LOW);
          strId="x";

        }
        if (strId == "0" ){
          digitalWrite(led,LOW);
          digitalWrite(relay1,LOW);
          digitalWrite(relay2,LOW);
          digitalWrite(relay3,LOW);
          digitalWrite(relay4,LOW);
          data1 = false;

        }

          if (strId == "2" ){
          data1 = true;
          digitalWrite(led,HIGH);
          digitalWrite(relay1,HIGH);
          digitalWrite(relay2,HIGH);
          digitalWrite(relay3,LOW);
          digitalWrite(relay4,LOW);
          delay(600);
          digitalWrite(led,LOW);
```

```
        digitalWrite(relay1,LOW);
        digitalWrite(relay2,LOW);
        digitalWrite(relay3,LOW);
        digitalWrite(relay4,LOW);
        strId="x";

       }
     }
    else{

    }

     delay(5000);
     sprintf(buff, "/channels/466685/fields/2/last/?key=47PNNYN89TBVWUTD"); //LAST DATA
FIELD 1
     Serial.println(buff);

     rest.get((const char*)buff);
     if(rest.getResponse(response, 266) == HTTP_STATUS_OK){

      strId = "";  strData = "";  strCode = "";
      getData();

     if (strId == "1" ){
      data2 = true;
      digitalWrite(relay1,LOW);
      digitalWrite(relay2,HIGH);
      digitalWrite(relay3,LOW);
      digitalWrite(relay4,LOW);
      digitalWrite(led,HIGH);
      delay(600);
      digitalWrite(relay1,LOW);
      digitalWrite(relay2,LOW);
      digitalWrite(relay3,LOW);
      digitalWrite(relay4,LOW);
      digitalWrite(led,LOW);
      strId="x";

     }
     if (strId == "0" ){
      data2 = false;
      digitalWrite(relay1,LOW);
      digitalWrite(relay2,LOW);
      digitalWrite(relay3,LOW);
      digitalWrite(relay4,LOW);
      digitalWrite(led,LOW);
     }

        if (strId == "2" ){
      data2 = true;
      digitalWrite(relay1,LOW);
      digitalWrite(relay2,HIGH);
```

```
  digitalWrite(relay3,HIGH);
  digitalWrite(relay4,LOW);
  digitalWrite(led,HIGH);
  delay(600);
  digitalWrite(relay1,LOW);
  digitalWrite(relay2,LOW);
  digitalWrite(relay3,LOW);
  digitalWrite(relay4,LOW);
  digitalWrite(led,LOW);
  strId="x";

 }
 }
 else{


}

 delay(5000);
 sprintf(buff, "/channels/466685/fields/3/last/?key=47PNNYN89TBVWUTD"); //LAST DATA
FIELD 1
 Serial.println(buff);

 rest.get((const char*)buff);
 if(rest.getResponse(response, 266) == HTTP_STATUS_OK){

  strId = "";  strData = "";  strCode = "";
  getData();

 if (strId == "1" ){
 data3 = true;
 digitalWrite(relay1,LOW);
 digitalWrite(relay2,LOW);
 digitalWrite(relay3,HIGH);
  digitalWrite(relay4,LOW);
 digitalWrite(led,HIGH);
 delay(600);
 digitalWrite(relay1,LOW);
 digitalWrite(relay2,LOW);
 digitalWrite(relay3,LOW);
 digitalWrite(relay4,LOW);
 digitalWrite(led,LOW);
 strId="x";

 }
 if (strId == "0" ){
 data3 = false;
 digitalWrite(relay1,LOW);
 digitalWrite(relay2,LOW);
 digitalWrite(relay3,LOW);
 digitalWrite(relay4,LOW);
 digitalWrite(led,LOW);
```

```
    }

      if (strId == "2" ){
    data3 = true;
    digitalWrite(relay1,LOW);
    digitalWrite(relay2,LOW);
    digitalWrite(relay3,HIGH);
     digitalWrite(relay4,HIGH);
    digitalWrite(led,HIGH);
    delay(600);
    digitalWrite(relay1,LOW);
    digitalWrite(relay2,LOW);
    digitalWrite(relay3,LOW);
    digitalWrite(relay4,LOW);
    digitalWrite(led,LOW);
    strId="x";

    }
    }
    else{


   }

   delay(5000);
   sprintf(buff, "/channels/466685/fields/4/last/?key=47PNNYN89TBVWUTD"); //LAST DATA
FIELD 1
    Serial.println(buff);

    rest.get((const char*)buff);
    if(rest.getResponse(response, 266) == HTTP_STATUS_OK){

      strId = "";  strData = "";  strCode = "";
      getData();                                  // GET DATA

    if (strId == "1" ){
    data4 = true;
   digitalWrite(relay1,LOW);
   digitalWrite(relay2,LOW);
   digitalWrite(relay3,LOW);
   digitalWrite(relay4,HIGH);
   digitalWrite(led,HIGH);
   delay(600);
   digitalWrite(relay1,LOW);
   digitalWrite(relay2,LOW);
   digitalWrite(relay3,LOW);
   digitalWrite(relay4,LOW);
   digitalWrite(led,LOW);
   strId="x";

    }
    if (strId == "0" ){
```

```
 data4 = false;
digitalWrite(relay1,LOW);
digitalWrite(relay2,LOW);
digitalWrite(relay3,LOW);
digitalWrite(relay4,LOW);
digitalWrite(led,LOW);
 }

    if (strId == "2" ){
 data4 = true;
digitalWrite(relay1,HIGH);
digitalWrite(relay2,LOW);
digitalWrite(relay3,LOW);
digitalWrite(relay4,HIGH);
digitalWrite(led,HIGH);
delay(600);
digitalWrite(relay1,LOW);
digitalWrite(relay2,LOW);
digitalWrite(relay3,LOW);
digitalWrite(relay4,LOW);
digitalWrite(led,LOW);
strId="x";

 }

 }
 else{


}

 delay(5000);
 loop_count++;
 Serial.println("LOOP : ");
 Serial.println(loop_count);

 if(loop_count == 6 ){
  loop_count = 0;

 if(data1) {
     dtostrf(1, 1, 1, str_field1);
 }else{
   dtostrf(0.1, 2, 1, str_field1);
 }
 if(data2) {
     dtostrf(1, 1, 1, str_field2);
 }else{
   dtostrf(0.1, 2, 1, str_field2);
 }
 if(data3) {
     dtostrf(1, 1, 1, str_field3);
 }else{
```

```
      dtostrf(0.1, 2, 1, str_field3);
    }
    if(data4) {
       dtostrf(1, 1, 1, str_field4);
    }else{
      dtostrf(0.1, 2, 1, str_field4);
    }

       sprintf(buff,
"//update?key=47PNNYN89TBVWUTD&field1=%s&field2=%s&field3=%s&field4=%s",str_field1
,str_field2,str_field3,str_field4);
       Serial.println(buff);

       rest.get((const char*)buff);
       Serial.println("ARDUINO: Mengirim data terbaru");

       if(rest.getResponse(response, 266) == HTTP_STATUS_OK){
         Serial.println("ARDUINO: Berhasil GET Data");
         strId = "";  strData = "";  strCode = "";
         getData();
       }
    delay(5000);
    }
 }

 else{
 }

}

void getData(){
  int i=0,j=0,k=0;

       for (i = 0; i < 10; i++){

       if((response[i] == '\r') || (response[i] == '\n')) {
       }
       else{
         strId += response[i];
       }

       if (response[i] == '\n'){
           i++;
           break;
       }
      }

      Serial.println("");
      Serial.print("ID : ");
      Serial.print(strId);

       for (j = i; j < (i+20); j++){
```

```
      if((response[j] == '\r') || (response[j] == '\n')) {
      }
      else{
        strData += response[j];
      }

      if (response[j] == '\n'){
          j++;
          break;
      }
     }

     Serial.println("");
     Serial.print("Data : ");
     Serial.print(strData);
      for (k = j; k < (j+10); k++){

      if((response[k] == '\r') || (response[k] == '\n')) {
      }
      else{
        strCode += response[k];
      }

      if (response[k] == '\n'){
          break;
      }
     }
     Serial.println("");
     Serial.print("Code : ");
     Serial.print(strCode);
     Serial.println("");
    }
 boolean hardReset() {
 String tmpData;
}
```

ANEXO B

Conjunto Habitacional "CAMINO REAL "

CARTA COMPROMISO Y ENTREGA

Yo, DIEGO ANDRÉS TAPIA FONSECA en mi condición de Administrador del Conjunto Habitacional "Camino Real" y después de haber solicitado se conceda el uso provisional de las instalaciones de los parqueaderos 123,122,121,120,115,114,113,112, con el objetivo de realizar la instalación y demostración del proyecto final de tesis de la carrera que cursa el Sr. Fausto Renan Bautista Palate con C.C. 1714436647 que será realizada con la presencia de representantes de su Universidad, emito este documento como carta de responsabilidad y compromiso sobre el mantenimiento del bien prestado.

Debo informar que este tema ya se conversó en reunión de directorio y fue aprobada por todos los miembros del mismo en donde se manifestaron las siguientes condiciones que se cumplirán al pie de la letra:

- Las instalaciones, así como cableado deberán ser sobrepuestos de manera de que no se atente con el estado de la pintura de las mismas.
- Las fechas en que se efectuarán estas instalaciones para esta actividad serán desde el 21 de julio hasta el 12 de agosto.
- En caso de afectarse la pintura o alguno de los componentes de cualquier parqueadero el Sr. Bautista asume el costo de la reparación de los mismos.

Mediante estas condiciones se realiza la entrega de los parqueaderos

ENTREGADO POR:

Ingo. Andrés Tapia
ADMINISTRADOR

CAMINO REAL Conjunto Habitacional ADMINISTRACIÓN

RECIBIDO POR:

FAUSTO BAUTISTA
ARRENDATARIO
PARQUEADERO DEP 74

Dirección: García Moreno N10-31 entre Galápagos y Oriente * Teléfono: 0984677604
E-mail: conjhab_caminoreal@hotmail.com

ANEXO C

"Responsabilidad con pensamiento positivo"

UNIVERSIDAD TECNOLÓGICA ISRAEL

MANUAL DE USUARIO

TEMA:

Desarrollo de un prototipo de una Red de Sensores Inalámbricos para monitorear por smartphone la disponibilidad de plazas de un estacionamiento utilizando tecnologías de IoT

AUTOR:

Fausto Renán Bautista Palate

INGENIERO EN ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES

Quito, Ecuador

Año 2018

CONTENIDO

MANUAL DE USUARIO

“Desarrollo de un prototipo de una Red de Sensores Inalámbricos para monitorear por smartphone la disponibilidad de plazas de estacionamientos utilizando tecnologías de IoT”

NOTAS PARA EL USUARIO:

Estimado Usuario, antes de utilizar el producto lea cuidadosamente las instrucciones de seguridad.

ADVERTENCIAS:

Riesgo eléctrico, tenga mucho cuidado con la manipulación de los componentes eléctricos del sistema, podría causar daños a su salud.

Mucha precaución en no derramar líquidos sobre los dispositivos electrónicos del sistema.

Utilice guantes o pulseras antiestáticas para realizar cualquier manipulación en los dispositivos y elementos.

FUNCIONAMIENTO:

El sistema permite el monitoreo del estado de ocupación de 8 plazas de estacionamiento a través de una aplicación móvil para smartphone con sistema operativo Android en tiempo real, desde cualquier lugar y en cualquier momento.

La aplicación desarrollada permite el acceso principalmente a tres pantallas.

a) THINGSPEAK

b) VISUALIZADOR DE ESTACIONAMIENTOS

c) CONTROL DE RESERVAS

*Figura **0**.1. MENU DE BOTONES DE LA APLICACION MOVIL*

Botón THINGSPEAK

Permite tener acceso directo al servidor IoT y visualizar los estados de los parqueaderos. Esto lo realiza a través de 8 gráficas propias del servidor IoT las cuales contienen las muestras realizadas en función del tiempo vs el estado de ocupación (0 para libre y 1 para ocupado) haciendo que se tenga un histórico visual de los cambios de estado a lo largo del tiempo y almacenando en su base de datos registros como cambios de estado, hora (h,mm,ss) y fecha.

Figura 102. PANTALLA THINGSPEAK

Botón VISUALIZADOR DE ESTACIONAMIENTOS

Para ingresar se escribir las credenciales de administrador o cliente.

Figura 03. Pantala de ingreso de credenciales

Este visualizador es otra representación de la misma información de ThingSpeak pero de manera mas amigable y de rápida interpretación ya que muestra el estado de los estacionamientos con la figura de la presencia de un AUTO para indicar el estado de ocupado y con la palabra “LIBRE” en color verde para el estado de desocupado.

Figura 10.4. VISUALIZADOR DE ESTACIONAMIENTOS

En esta pantalla se tiene los botones de reservar para cada estacionamiento. Presionando el mismo se realiza la solicitud al sistema para reservar la plaza lo que hará que el foco rojo del sensor empiece a parpadear hasta que el auto ubique su sitio y se retire levantando de estado parpadeante a estado en color verde nuevamente como indicativo de que queda libre

Se tiene el menú de opciones en el lado derecho de la pantala para ver el histórico de estacionamiento donde se tiene todos los registros de las plazas utilizadas junto con un cálculo de una tarifa ajustable en la aplicación

También se tiene un botón de seteo general para el caso de que se requiera reiniciar los procesos y contadores por parte del administrador.

También tiene el gestor de usuarios para ingresar los datos de usuarios y el rol de administrador o cliente

*Figura **0**5. Menú de opciones del visualizador de estacionamientos*

Botón CONTROL DE RESERVAS

Esta pantalla es para el administrador que adicional al visualizador de Estacionamientos también tiene botones para reservar, liberar y setear los cambios de estado

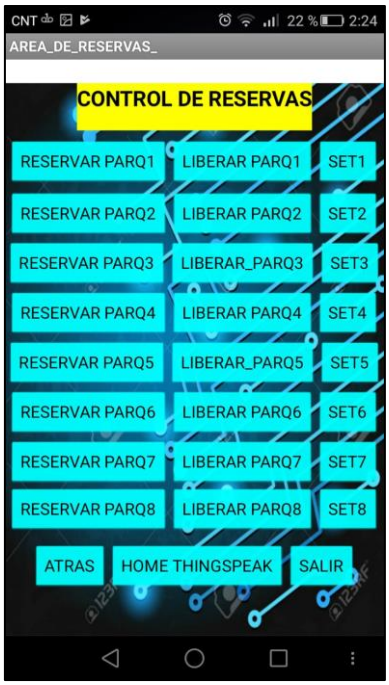


Figura 06. Pantalla de control de reservas

También tiene el botón para ir directamente a la pantalla ThingSpeak

ASISTENCIA TÉCNICA

Para temas de actualización o soporte por favor comuníquese al servicio técnico autorizado.

CONTACTOS:

Fausto Bautista

Tel: 0969081222

Mail: faustobestdj@yahoo.com

ANEXO D

"Responsabilidad con pensamiento positivo"

UNIVERSIDAD TECNOLÓGICA ISRAEL

MANUAL TÉCNICO

TEMA:

Desarrollo de un prototipo de una Red de Sensores Inalámbricos para monitorear por smartphone la disponibilidad de plazas de un estacionamiento utilizando tecnologías de IoT

AUTOR:

Fausto Renán Bautista Palate

INGENIERO EN ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES

Quito, Ecuador

Año 2018

CONTENIDO

MANUAL TÉCNICO

El presente documento presenta la información técnica de los dispositivos, códigos de programación y circuitos principales del proyecto denominado:

Desarrollo de un prototipo de una Red de Sensores Inalámbricos para monitorear por smartphone la disponibilidad de plazas de un estacionamiento utilizando tecnologías de IoT

1. FUNCIONAMIENTO:

El sistema permite el monitoreo del estado de ocupación de 8 plazas de estacionamiento a través de una aplicación móvil para smartphone con sistema operativo Android en tiempo real, desde cualquier lugar y en cualquier momento.

Etapa de acceso: Microcontrolador Sensor ultrasónico, microcontrolador ATMEGA328, los módulos de radio frecuencia Tx/Rx 433Mhz.

Etapa de interface: Módulo wi-fi ESP826, programación IDE Arduino.

Etapa en la nube: ThingSpeak.

Etapa de aplicación en el Smartphone: Mit App Inventor.

El sensor ultrasónico genera información del status de cada estacionamiento y luego es transmitida por un módulo transmisor RF de 400Mhz hasta la placa de control, la misma que cuenta con el microcontrolador ATMEGA 328 para la parte del procesamiento.

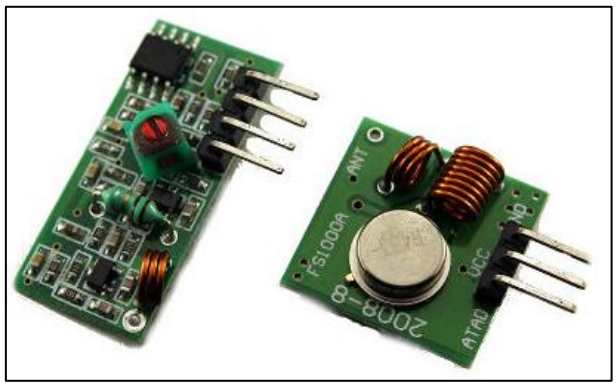
Este microcontrolador procesa la información y por medio del módulo wi-fi ESP8266 comunica la información hacia la nube de internet mediante el protocolo IEEE802.11 y la salida de internet provisto por un ISP.

En la nube de Internet utiliza la plataforma IoT denominada ThingSpeak para mostrar de manera gráfica la información de cada sensor de cada estacionamiento y el estatus si se encuentra libre u ocupado. Adicionalmente se da la posibilidad de reservar cualquiera de las plazas del sitio estacionamientos de prueba mediante las API’s que ThingSpeak brinda para estos efectos.

Finalmente la aplicación App para móviles Android cuyo ejecutable tenga extensión .apk la cual es propia de archivos ejecutables para dispositivos con sistema operativo Android.

Especificaciones técnicas transmisor de Radio Frecuencia:

- Señal de radiofrecuencia: Modulación ASK (Modulación por Desplazamiento de Amplitud)
- Fuente de alimentación: 12V (también disponible en versiones de 3V y 5V)
- Consumo de corriente: <16 mA
- Potencia de transmisión: 13 dBm
- Desviación de frecuencia: +- 75kHz
- Alcance útil hasta 350 metros (12V), 230 metros (5V), 160 metros (3V)
- Disponible en frecuencias de 433.92 MHz (433MHz) y 315.0 MHz
- Velocidades de transmisión hasta 20kbps



1.1. Receptor de Radio Frecuencia

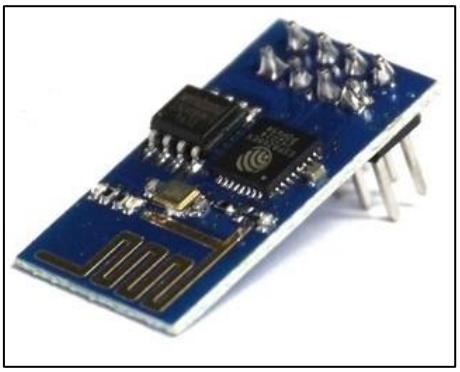
Este es un receptor de datos en UHF, para montaje en circuito impreso (PCB). Con el transmisor correspondiente (13dBm), permite implementar enlaces TX/RX inalámbricos de datos a velocidades de hasta 4.8kbps con distancias de hasta 40 metros en interiores o 110 metros en campo abierto.

Características

- Velocidades de hasta 4.8kbps

- Alcance utilizable de hasta 110 metros
- Versiones disponibles en 433.92 MHz (433MHz) y 315.0MHz
- Versiones disponibles: regulado y no regulado
- Rápido tiempo de establecimiento de datos
- Consumo de corriente: 2.2mA

1.2. Módulo wi-fi ESP8266



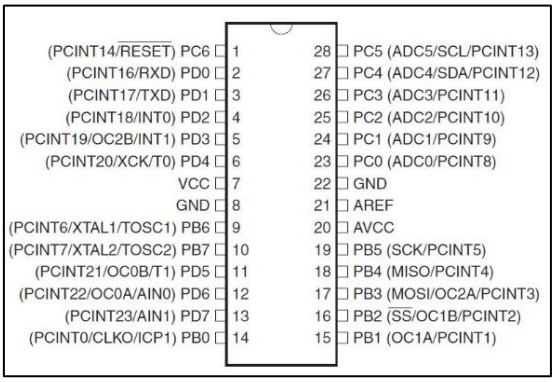
- Trabaja con el protocolo 802.11 b/g/n
- Integra stack de protocolos TCP/IP
- 64KBytes de RAM de instrucciones
- 96KBytes de RAM de datos
- Alimentación 3.3 VDC

1.3. Características técnicas Sensor Ultrasónico HC-SR04

Working Voltage	Voltaje de trabajo	DC 5V
Working Current	Corriente de trabajo	15 mA
Working Frequency	Frecuencia de trabajo	40 Hz
Max Range	Distancia máxima	4 m
Min Range	Distancia mínima	2 cm
Measuring Angle	Angulo eficaz	15 degree
Trigger Input Signal	Disparo de señal de entrada	10 us TTL pulse
Echo Output Signal	Eco señal de salida	Input TTL lever signal and the range in proportion
Dimension	Dimensiones	45*20*15 mm

1.4. Microcontrolador ATMEGA328

Voltaje de operación	5V
Número de Pines:	28
Memoria FLASH	32KB
Memoria RAM	2KB
EEPROM	1KB
Máxima frecuencia de funcionamiento	20Mhz
CPU	8-bit AVR
Pines de entrada/salida	23
Entradas Analógicas (ADC)	6
Formato DIP	



1.5. Diagrama esquemático de la Tarjeta Sensor Transmisor Tx

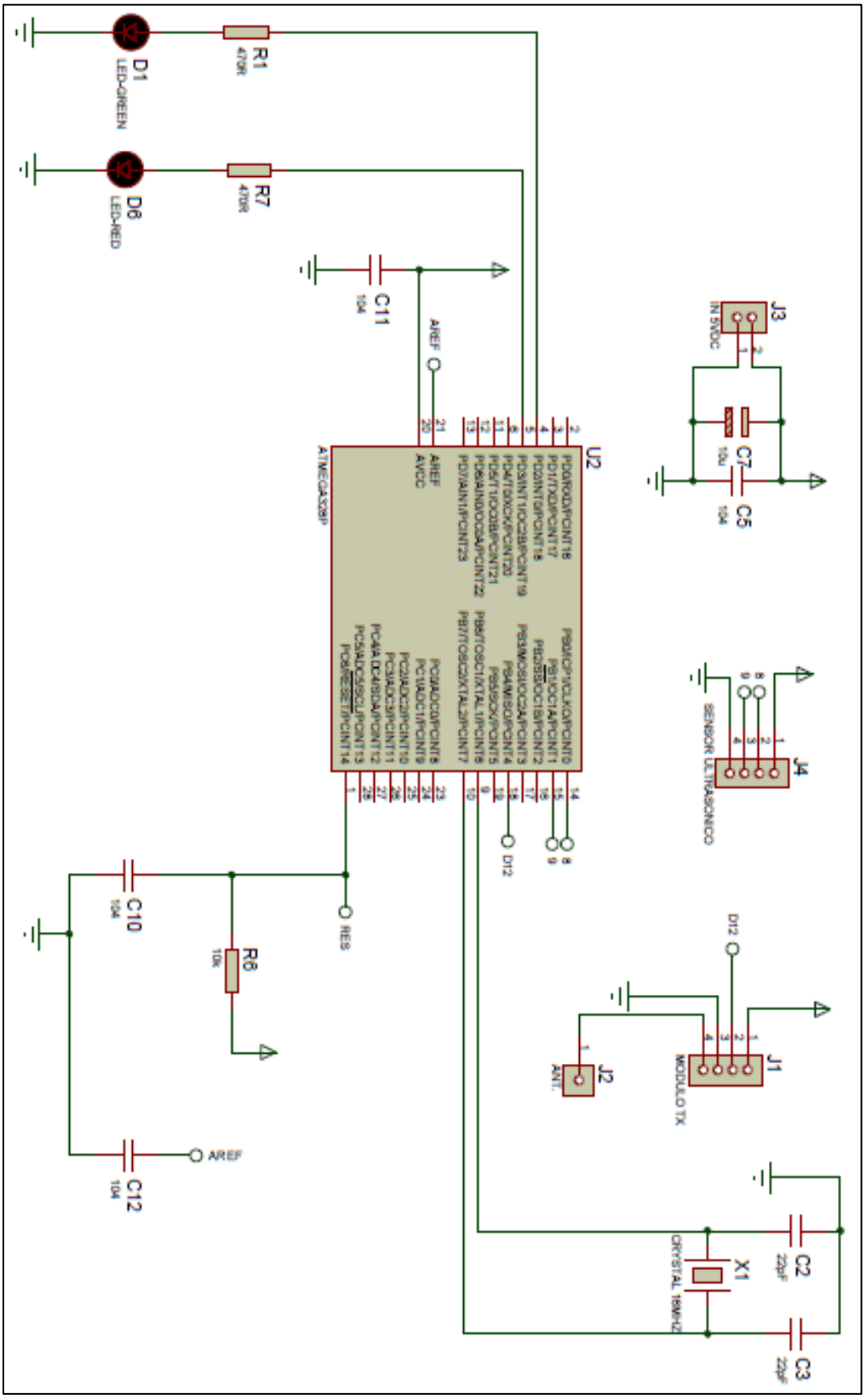


Diagrama esquemático de la Tarjeta Sensor Transmisor Tx

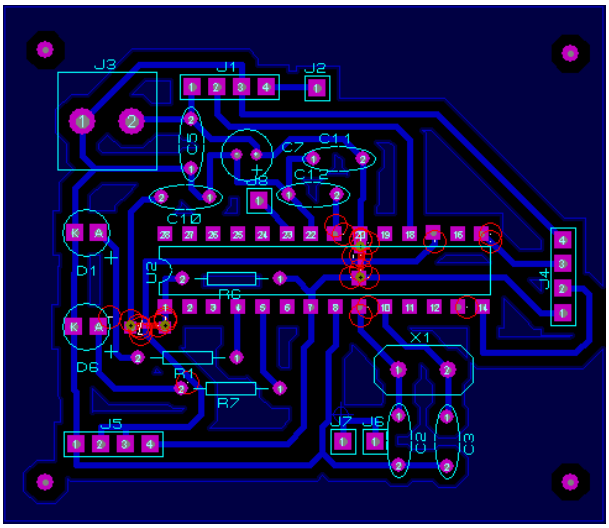
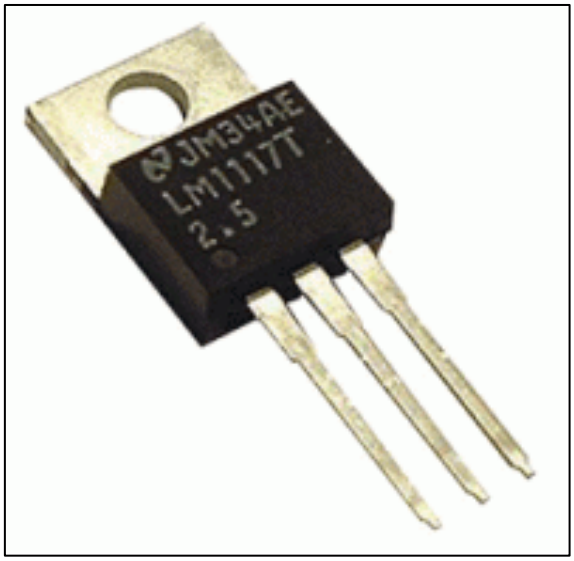


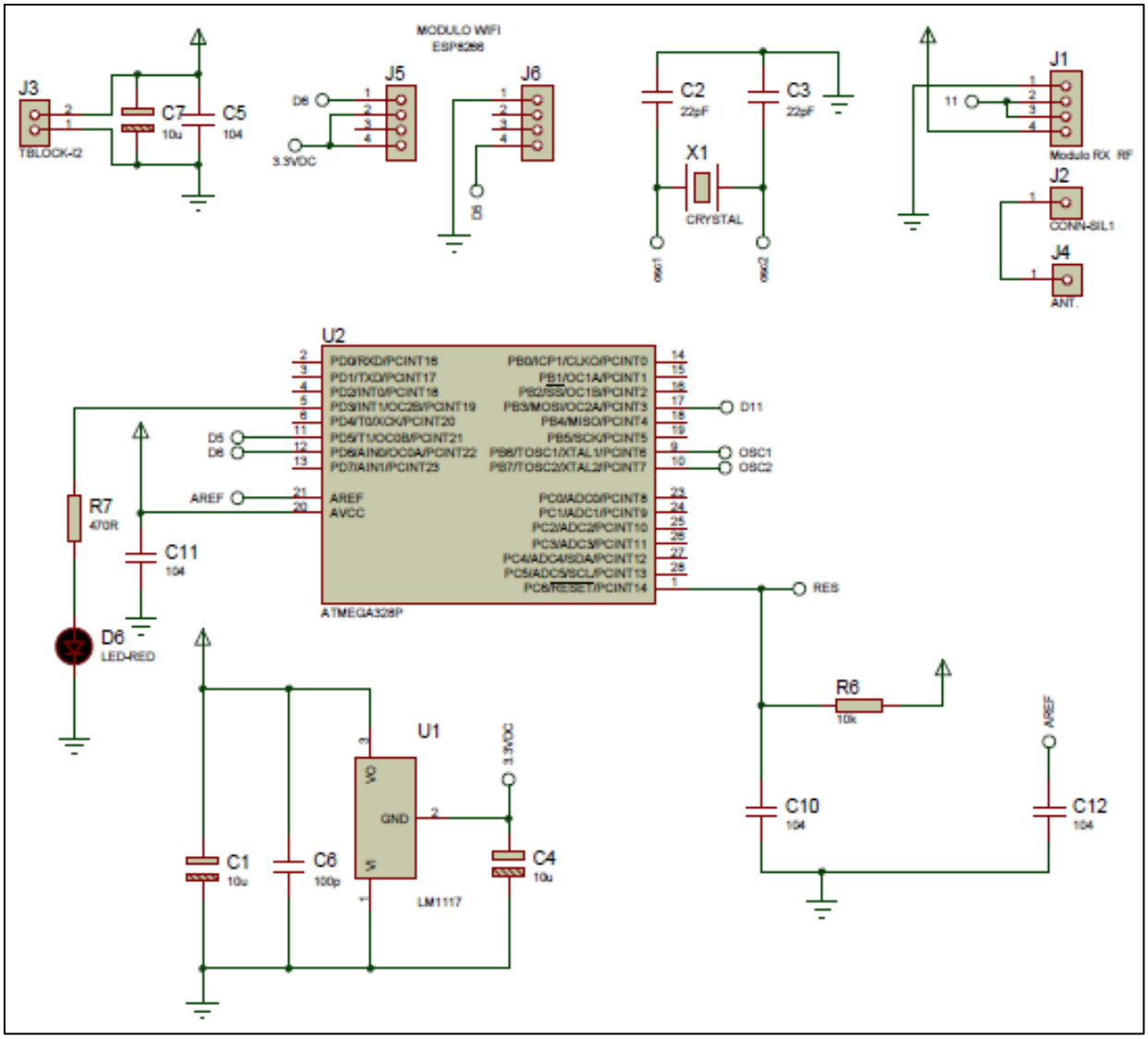
Diagrama PCB de la tarjeta Sensor Transmisor Tx

1.6. Características técnicas del regulador LM1117 (Anexo D)

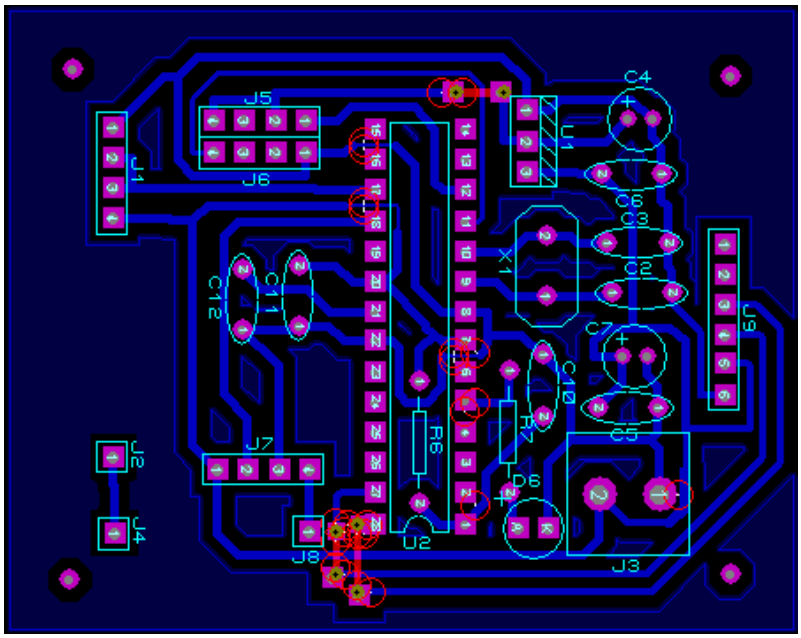
Symbol	Parameter	Conditions	Min	Typ	Max	Units
V out	Output Voltage	LM1117-3.3 Iout = 10mA, Vin=5V Tj=25°C	3.267	3.300	3.333	V
		0 ≤ I out ≥ 800 mA, 4.75V ≤ Vin ≥ 10V	3.235	3.300	3.365	V



Regulador de voltaje LM1117



Esquema Tarjeta Receptor (Rx)



Placa PCB de la Tarjeta Receptor (Rx)

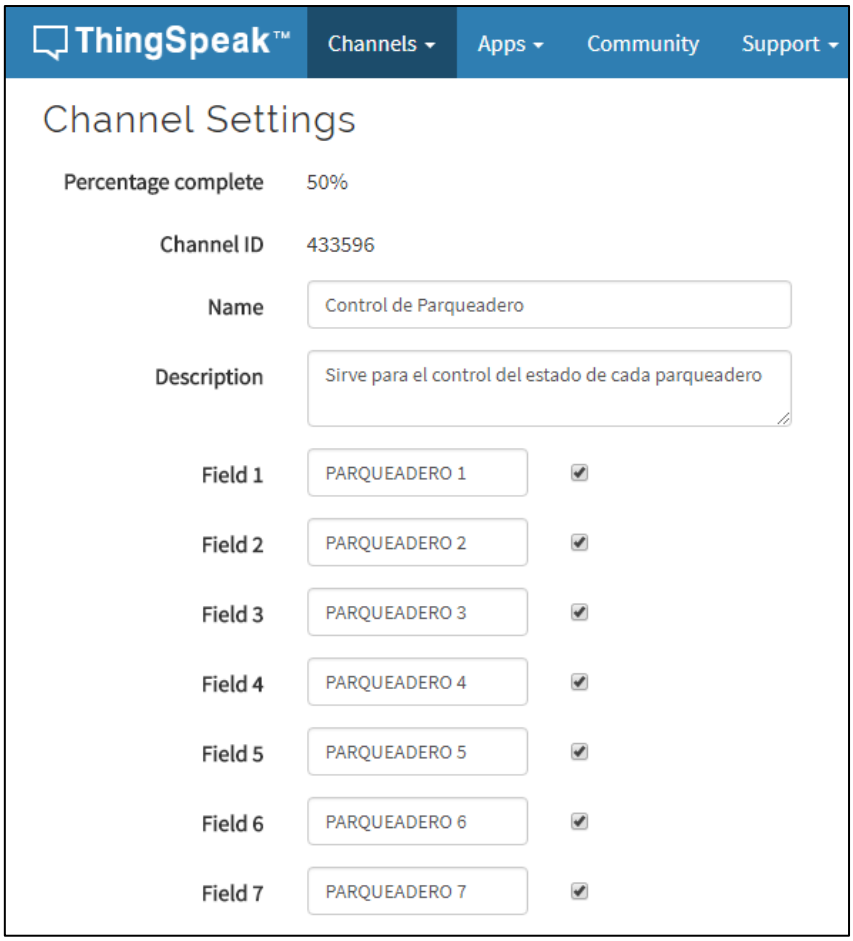
Programación Tarjeta Receptor Rx

#include <SoftwareSerial.h>	Para la comunicación serial con el ESP8266 por los pines 6y5
SoftwareSerial Serial2x(6,5);	Se declaró el pin 5 rx y 6 tx
#include <stdlib.h>	Librería estándar para funciones básicas de comunicación con ThingSpeak
Serial2x.begin(9600);	Para comunicación del micro controlador y el módulo ESP8266

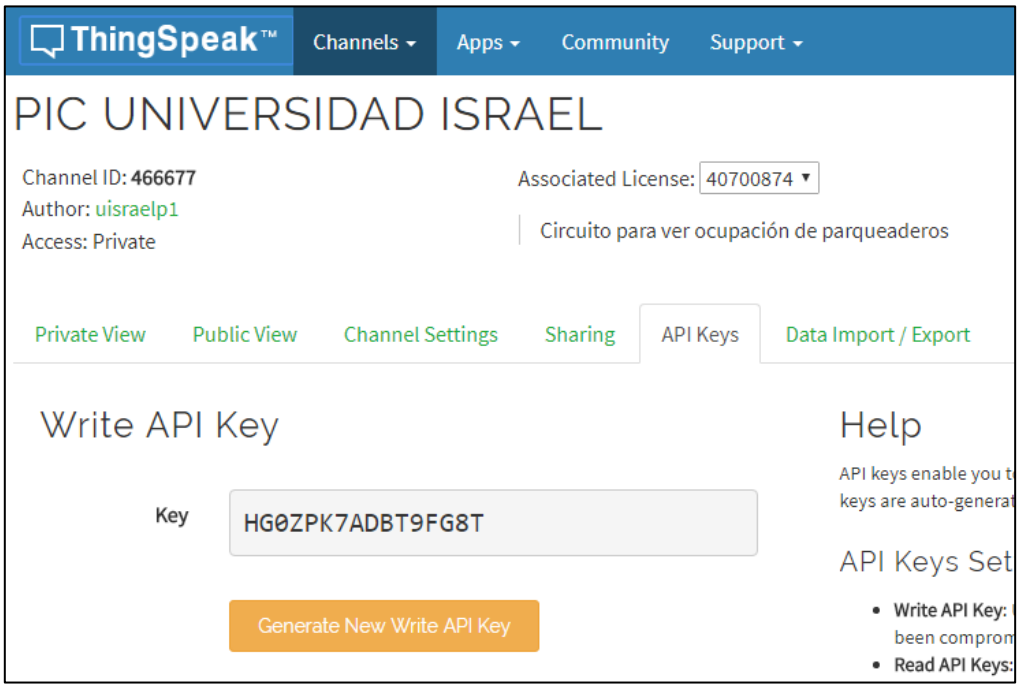
String apiKey = "HG0ZPK7ADBT9FG8T"	Ingresa la API KEY que se generó al crear nuestro canal

Creación de fields para cada estacionamiento en ThingSpeak

Se ingresa en el link: https://thingspeak.com/login para crear nuestro canal llenando los siguientes campos



verificación del canal y de las gráficas de los sensores



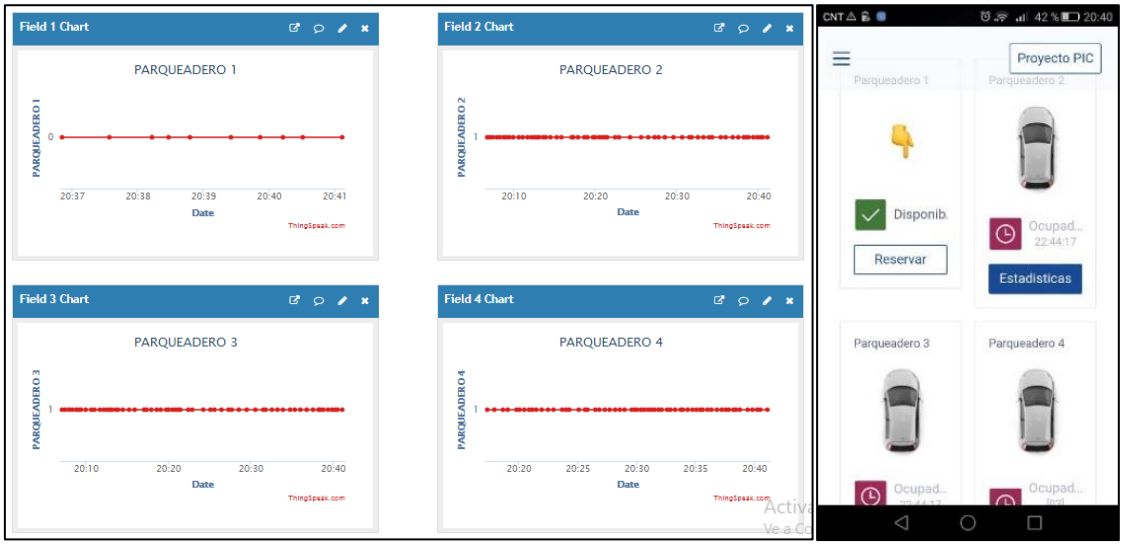
Generación del API key en ThingSpeak

```
rx

//  Variables
int PulseSensorPurplePin = 0;        // Pulse Sensor PURPLE WIRE connected to ANALOG PIN 0
int ledr = 2;   //  The on-board Arduion LED
int ledv = 3;
int Signal;                // holds the incoming raw data. Signal value can range from 0-1024
int Threshold = 550;            // Determine which Signal to "count as a beat", and which to ingore.
int dato=0;
int BPM;
// replace with your channel's thingspeak API key
String apiKey = "HG0ZPK7ADBT9FG8T";

SoftwareSerial Serial2x(6,5);//Declaramos el pin 5 rx y 6 tx

int i=1;
const int sensorPin= A0;
// On Arduino:  0 - 1023 maps to 0 - 5 volts
#define VOLTAGE_MAX 5.0
#define VOLTAGE_MAXCOUNTS 1023.0
```

Configuración del API key en el programa de la Tarjeta Receptor (Rx)

. Pantalla VISUALIZADOR_ESTACIONAMIENTOS

La gráfica del Field 1 (Parqueadero 1) el estado “0” representa a estado libre; en esta pantalla en cambio aparecerá la palabra disponible en color verde y para la gráfica del Field 1 el estado “1” representa a estado ocupado o reservado en cambio en esta pantalla aparecerá un auto con el botón estadísticas en color rojo.



Pantalla Visualización de Estacionamientos

ASISTENCIA TÉCNICA

Para temas de actualización o soporte por favor comuníquese al servicio técnico autorizado.

CONTACTOS:

Fausto Bautista

Tel: 0969081222

Mail: faustobestdj@yahoo.com

DECLARACIÓN Y AUTORIZACIÓN

Yo, Fausto Renán Bautista Palate, CI 1714436647 autor del trabajo de graduación:

Desarrollo de un prototipo de una Red de Sensores Inalámbricos para monitorear por smartphone la disponibilidad de plazas de un estacionamiento utilizando tecnologías de IoT, previo a la obtención del título de **Ingeniero en Electrónica Digital y Telecomunicaciones** en la UNIVERSIDAD TECNOLÓGICA ISRAEL.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Articulo 144 de la Ley Orgánica de Educación Superior, de difundir el respectivo trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de graduación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Quito, septiembre del 2018

Atentamente.

Fausto Renán Bautista Palate.
C.I. 1714436647

Plagiarism Checker X Originality Report

Similarity Found: 10%

Date: miércoles, agosto 15, 2018
Statistics: 2171 words Plagiarized / 12528 Total words
Remarks: Low Plagiarism Detected - Your Document needs Optional Improvement.

/ "Responsabilidad con pensamiento positivo" UNIVERSIDAD TECNOLOGICA ISRAEL TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE: INGENIERO EN ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES TEMA: Desarrollo de un prototipo de una Red de Sensores Inalámbricos para monitorear por smartphone la disponibilidad de plazas de un estacionamiento utilizando tecnologías de IoT. AUTOR: Fausto Renán Bautista Palate TUTOR: Ing.

Morales Arévalo Flavio David Quito, Ecuador Año 2018 DECLARACIÓN Yo, Fausto Renán Bautista Palate, estudiante de la carrera de Electrónica Digital y Telecomunicaciones, perteneciente a la Universidad Tecnológica Israel, declaro que el contenido aquí descrito es de mi autoría, y de mi absoluta responsabilidad legal. Quito D.M., agosto del 2018 Fausto Renán Bautista Palate C.I.:

1714436647

UNIVERSIDAD TECNOLÓGICA ISRAEL APROBACIÓN DEL TUTOR En mi calidad de tutor del trabajo de titulación certifico: Que el trabajo de titulación "DESARROLLO DE UN PROTOTIPO DE UNA RED DE SENSORES INALÁMBRICOS PARA MONITOREAR POR SMARTPHONE LA DISPONIBILIDAD DE PLAZAS DE UN ESTACIONAMIENTO UTILIZANDO TECNOLOGÍAS DE IoT", presentado por el Sr.

Fausto Renán Bautista Palate, estudiante de la carrera de Electrónica Digital y Telecomunicaciones, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación. Quito D.M. agosto del 2018 TUTOR ..... Ing. Flavio Morales Arévalo, Mg AGRADECIMIENTO Mi agradecimiento a Dios y a mi familia que siempre me están apoyando a cada momento por insignificante que parezca y ha hecho posible llegar a estas instancias. Este es el fruto de su sacrificio y mi manera de decirles gracias infinitas.

Fausto Renán Bautista Palate DEDICATORIA Este trabajo lo dedico a mi Dios que me cuida con salud y vida; y también a mi madre que con su sacrificio y amor supo brindarme mi educación secundaria. Por que la he visto sufrir velando que no nos falte nada y al ser el pilar más fuerte de mi familia, su buen ejemplo me ha inspirado a ser siempre mejor persona, pero sobre todo porque me ha enseñado que el amar a tus seres queridos consiste en demostrarlo y no quede solo en palabras.

Fausto Renán Bautista Palate TABLA DE CONTENIDO RESUMEN 8 CAPITULO 1. 15 FUNDAMENTACIÓN TEÓRICA 15 1.1. Estado de Arte 15 1.2. Marco Teórico 15 1.2.1. Sensor Ultrasónico 15 1.2.2. Transmisor de Radio Frecuencia 16 1.2.3. Receptor de Radio Frecuencia 17 1.2.4. Microcontrolador ATMEGA328 18 1.2.5. Módulo wi-fi ESP8266 18 1.2.7. API's Interfaces de Programación de Aplicaciones 20 1.2.8. Regulador de voltaje 7805 21 1.3. Fundamento teórico 21 1.3.1.

Compartir la información de la red local con la nube de Internet 21 1.3.2. Plataforma IoT 22 CAPITULO 2. 23 PROPUESTA 23 2.1. Propuesta de la tarjeta Sensor Transmisor (Tx) 25 2.1.1. Etapa de alimentación de la tarjeta Sensor Transmisor (Tx) 25 2.1.2. Desarrollo del monitoreo con Sensor de Ultra sonido HC-SR04 26 2.1.2.1. Microcontrolador ATMEGA328 27 2.1.2.2. Transmisor RF de Radiofrecuencia 28 2.1.5.

Diagrama esquemático de la Tarjeta Sensor Transmisor (Tx) 29 2.1.6. Diseño de la placa PCB de la Tarjeta Sensor Transmisor (Tx) 30 2.2. Propuesta de la Tarjeta Receptor (Rx) 31 2.2.1. Etapa de alimentación de la Tarjeta Receptor (Rx) 31 2.2.2.

RESUMEN El presente proyecto pretende mostrar el uso de una aplicación de tecnologías y plataformas utilizadas en Internet de las cosas, para subir información a la nube y ser aprovechada de diferentes maneras como por ejemplo servicios a la comunidad.

La misma fue implementada como prototipo en los parqueaderos del Conjunto Habitacional Camino Real en el centro histórico de la Ciudad de Quito. El llegar a conocer el status de las plazas del estacionamiento de manera remota, se traduciría en un traslado innecesario al sitio sin pérdida de tiempo y por ende a no contribuir a embotellamientos.

Si este sistema se aplica en estacionamientos municipales o privados, cuyas sedes se encuentren dentro de zonas críticas de alto índice de tráfico; resultaría muy provechoso para los conductores conocer de antemano si existen puestos libres evitando la entrada innecesaria de más vehículos a la zona de tráfico y ayudar a minorar congestiones.

El prototipo consta de sensores que monitorean cada estacionamiento y dicha información es transmitida inalámbricamente hacia una tarjeta receptora en donde el microcontrolador ATMEGA628 procesa la información y la misma es subida a la nube de internet con la ayuda del microcontrolador ESP8266 que utiliza el protocolo IEEE 802.11. Por medio de un servidor web levantado en la plataforma IoT conocida como ThingSpeak, se registra la información recolectada y se la presenta gráficamente para su administración. De igual manera dicha información es enviada por la nube hasta el smartphone donde también se presenta y consigue el monitoreo remoto.

Palabras claves: IoT, sensor ultrasónico, microcontrolador ATMEGA328, microcontrolador ESP8266, IEEE802.11, servidor web NOIP. ABSTRACT This project aims to show the use of an application of technologies and platforms used in the Internet of Things, to upload information to the cloud and be used in different ways such as community services. It was implemented as a prototype in the underground parking lots 1 of campus 1 of the Israel University.

Getting to know the status of the parking spaces remotely, would result in an unnecessary transfer to the site without loss of time and therefore not to contribute to traffic jams. If this system is applied in municipal or private parking, whose headquarters are located within critical zones of high traffic index; It would be very helpful for drivers to know beforehand if there are free positions avoiding the unnecessary entry of more vehicles into the traffic area and help reduce

congestion.

The prototype consists of sensors that monitor each parking lot and this information is transmitted wirelessly to a receiving card where the ATMEGA628 microcontroller processes the information and it is uploaded to the internet cloud with the help of the ESP8266 microcontroller that uses the IEEE 802.11 protocol. Through a web server built on the IoT platform known as Cayenne, the information collected is recorded and presented graphically for administration.

In the same way, this information is sent by the cloud to the smartphone where it is also presented and gets remote monitoring. Keywords: IoT, sensor, microcontroller ATMEGA328, ESP8266, IEEE802.11, NOIP web server. INTRODUCCIÓN ANTECEDENTES DE LA SITUACIÓN OBJETO DE ESTUDIO En la actualidad en las zonas urbanas es conocido el tema de embotellamientos y el crecimiento del parque automotor.

Esto hace que el encontrar disponible un sitio de estacionamiento se torne molestoso sobre todo en zonas de alto tráfico vehicular con calles angostas y de un solo sentido que hacen que el dar una vuelta a la manzana en busca de estacionamiento ocasionen pérdidas de tiempo que retrasan las actividades y las consecuentes molestias al conductor.

En la ciudad ya hay establecimientos de Centros Comerciales que disponen de formas de señalización en sus estacionamientos; sin embargo, el beneficio es solamente dentro de sus áreas de competencia, mientras que, en estacionamientos públicos, zonas azules u otras zonas de estacionamiento particular no se tiene manera de saber de su disponibilidad como para evitar estos inconvenientes.

PLANTEAMIENTO DEL PROBLEMA Para un conductor que tiene pensado dejar su vehículo en un sitio de estacionamiento dentro de una zona de alto tráfico y no lo encuentra, le ocasiona molestias que retrasan sus actividades como embotellamientos, pérdidas de tiempo, desperdicio de combustible, además que contribuye con el aumento de contaminación al medio ambiente.

De igual manera cuando existen eventos especiales como partidos de fútbol, conciertos, desfiles en los cuales la demanda de estacionamientos en los sitios o sus alrededores aumentan notablemente. Se requiere de un método que permita a un conductor el conocer previamente la disponibilidad de plazas de estacionamiento. Actualmente existen tecnologías de comunicación que en conjunto con Redes Locales de Sensores hagan posible monitorear la ocupación de

las plazas de un estacionamiento y esta información pueda ser receptada por usuarios a través de su smartphone de donde se plantea la siguiente interrogante: ¿Se puede desarrollar un sistema de sensores inalámbricos para estacionamientos de zonas urbanas utilizando tecnologías para Internet de las cosas y que permita conocer previamente la disponibilidad de plazas a través de una aplicación móvil Android de un smartphone? JUSTIFICACIÓN Para poder solventar molestias en conductores que buscan estacionamientos y que a la vez contribuyen con los embotellamientos de las zonas de mayor carga vehicular se requiere un método para conocer previamente el status de las plazas de estacionamiento consistente en una red de sensores inalámbricos, pudiendo usarse además la nube de internet para que dicha información pueda ser accesible a cualquier persona desde un smartphone que posea conexión de datos, desde cualquier sitio y evitar las molestias por no poder encontrar un sitio con plazas disponibles.

Con ello se busca que los establecimientos de estacionamientos puedan enviar la información al público en general sobre la cantidad de puestos libres dispone como para que los conductores sepan cual es la mejor opción. El estacionamiento del Conjunto Habitacional Camino Real, ofrece un buen escenario para la implementación de un prototipo del presente proyecto.

Objetivos Objetivo general Desarrollar un prototipo de una Red de Sensores Inalámbricos que sirva para monitorear la disponibilidad de plazas de un sitio de estacionamientos a través de una aplicación en un smartphone Android mediante el uso de tecnologías que se utilizan para IoT Internet de las cosas. Objetivos específicos Desarrollar el prototipo para los Estacionamientos del Conjunto Habitacional Camino Real; el mismo que funcione con sensores inalámbricos y cuya disponibilidad sea visible en un smartphone.

Establecer la red de sensores utilizando tecnologías de comunicación inalámbricas que son usadas en la actualidad en Aplicaciones de desarrollo de Internet de las cosas IoT como los protocolos IEEE 802.15.4 o IEEE 802.11 Transmitir la información recolectada por el microcontrolador hacia el dispositivo móvil a través de la nube de Internet.

Implementar una aplicación móvil para smartphone con sistema operativo Android, que permita visualizar el status de los puestos de parqueo en tiempo real y llevar un conteo del tiempo transcurrido en su ocupación. Implementar la tarjeta electrónica para un microcontrolador que procese la información recibida desde los sensores y desarrollar la programación del mismo.

Realizar pruebas de funcionamiento, validación y análisis de resultados del prototipo. Descripción de los capítulos En el capítulo 1 se desarrolla brevemente la teoría en la que se basa el presente proyecto describiendo los diferentes elementos, tecnologías y plataformas IoT utilizados y esquematizado en fases de la siguiente manera: __Etapa de acceso: Microcontrolador Sensor ultrasónico, microcontrolador ATMEGA328, los módulos de radio frecuencia Tx/Rx 433Mhz. Etapa de interface: Módulo wi-fi ESP826, programación IDE Arduino. Etapa en la nube: Thing___Speak.

Etapa de aplicación en el Smartphone: Mit App Inventor. Se menciona brevemente algunas aplicaciones existentes que dan a este servicio en otros países como PARKOOL, BLUEPARKING y las diferencias en el modo de funcionamiento con el propuesto en el presente proyecto.

En el capítulo 2 Se plantea la implementación de los dispositivos del capítulo 1 en donde el sensor ultrasónico generará información del status de cada estacionamiento y luego será transmitida por un módulo transmisor RF de 400Mhz hasta la placa de control, la misma que contará con el microcontrolador ATMEGA 328 para la parte del procesamiento.

Este microcontrolador procesa la información y por medio del módulo wi-fi ESP8266 comunica la información hacia la nube de internet mediante el protocolo IEEE802.11 y la salida de internet provisto por un ISP. En la nube de Internet se propone utilizar la plataforma IoT denominada ThingSpeak para mostrar de manera gráfica la información de cada sensor de cada estacionamiento y el estatus si se encuentra libre u ocupado. Adicionalmente se pretende dar la posibilidad de reservar cualquiera de las plazas del sitio estacionamientos de prueba.

Finalmente se plantea la propuesta de una aplicación para móviles Android cuyo ejecutable tenga extensión .apk la cual es propia de archivos ejecutables para dispositivos con sistema operativo Android. En el capítulo 3 se desarrolla de manera detallada el diseño de la placa del sensor y de la placa del microcontrolador ATMEGA328 a través de los diagramas de los circuitos diseñados. Se muestra las simulaciones realizadas en Proteus para validar los parámetros calculados para la puesta en marcha del hardware.

Para la parte lógica se muestra el desarrollo del programa donde consta los parámetros de red seteados para la conexión con el módulo ESP8266 y los direccionamientos IP asignados por DHCP por parte del ruteador del ISP. Se muestra los resultados obtenidos después de las pruebas realizadas en el proto

para validar los valores de voltaje y corriente de la placa, así como también se comprueba la llegada de la información hasta el Servidor de ThingSpeak a través del Internet.

Una vez que la información se encuentra disponible a través de la nube se muestra el desarrollo del ambiente de visualización de los estacionamientos para que sea comprensible para un usuario cualquiera y finalmente se muestra el diseño de la aplicación para conseguir el funcionamiento del sistema desde el Smartphone. Se verifica el funcionamiento de la aplicación tanto en la visualización como ejecución de instrucciones básicas desde la aplicación.

Finalmente se expone las conclusiones obtenidas una vez que los objetivos han sido alcanzados, así como también las recomendaciones que optimicen al producto final. CAPITULO 1. FUNDAMENTACIÓN TEÓRICA Estado de Arte En países como España existen aplicaciones como Blue Parking, en Argentina Parkool, en Mexico Parkeo, en donde las mismas están operativas y tienen como principio básico de funcionamiento el intercambio de información de la comunidad de usuarios de la aplicación.

Esto hace que la calidad del servicio de algunas aplicaciones dependa directamente de una constante y eficaz participación de los miembros de la comunidad el cual vendría a ser un parámetro del cual no se tiene control. Son los usuarios los que potencian la información, los mapas y la navegación; cuantos más usuarios conduzcan con la aplicación abierta y de manera correcta, mejor será su funcionamiento.

Al poner sensores en cada estacionamiento que reemplacen el trabajo de la comunidad estamos asegurando que el 100% de los recursos aporten para un funcionamiento efectivo del sistema. Se requiere además que la información que generen los sensores sea subida de alguna manera a la nube de Internet para que esté disponible desde cualquier lugar y en cualquier momento. Adicional almacenar la información en una base de datos para llevar estadísticas del comportamiento en el transcurso del tiempo.

Marco Teórico Sensor Ultrasónico / Figura 1.1. Sensor de distancia Ultrasónico Fuente: http://cosasdeingenieria.com/esp/item/496/117/sensor-ultrasonico-de-distancia-ping Los sensores ultrasónicos miden la distancia mediante el uso de ondas ultrasónicas en donde el cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna desde el objeto. Los sensores ultrasónicos miden la distancia

al objeto contando el tiempo entre la emisión y la recepción.

Un sensor óptico tiene un transmisor y receptor, mientras que un sensor ultrasónico utiliza un elemento ultrasónico único, tanto para la emisión como la recepción. / Figura 1.2. Modo de funcionamiento del Sensor Ultrasónico Fuente: http://cursoarduino.proserquisa.com/2016/10/05/tutorial-13-modulo-sensor-ultrasonico-haz-una-alarma/ Transmisor de Radio Frecuencia Es un módulo de Radiofrecuencia a 433MHz que transmite datos en UHF y que sirve para montaje en circuitos PCB.

Cuando trabaja con el receptor de 433MHz que es su complemento, conforman un sistema tx/rx, que facilita la implementación de enlaces de datos de radiofrecuencia, alcanzando distancias de hasta 80 metros dentro de edificaciones o 350 metros en campo abierto cuando opera con la fuente de 12V Especificaciones técnicas transmisor de Radio Frecuencia: Señal de radiofrecuencia: Modulación ASK (Modulación por Desplazamiento de Amplitud) Fuente de alimentación: 12V (también disponible en versiones de 3V y 5V) Consumo de corriente: <16 mA Potencia de transmisión: 13 dBm Desviación de frecuencia: +-75kHz Alcance útil hasta 350 metros (12V), 230 metros (5V), 160 metros (3V) Disponible en frecuencias de 433.92 MHz (433MHz) y 315.0 MHz Velocidades de transmisión hasta 20kbps / Figura 1.3.

Módulo Tx y Rx de Radiofrecuencia Fuente: http://www.electrontools.com/Home/WP/2016/06/04/modulo-de-radio-frecuencia-rf433-arduino/ Receptor de Radio Frecuencia Este es un receptor de datos en UHF, para montaje en circuito impreso (PCB). Con el transmisor correspondiente (13dBm), permite implementar enlaces TX/RX inalámbricos de datos a velocidades de hasta 4.8kbps con distancias de hasta 40 metros en interiores o 110 metros en campo abierto. Características Velocidades de hasta 4.8kbps Alcance utilizable de hasta 110 metros Versiones disponibles en 433.92 MHz (433MHz) y 315.0MHz Versiones disponibles: regulado y no regulado Rápido tiempo de establecimiento de datos Consumo de corriente: 2.2mA Microcontrolador ATMEGA328 Es un microcontrolador RISC de 8 bits de alto rendimiento con memoria flash de 32 Kb capacidad de lectura y escritura, memoria EEPROM de 1 KB, SRAM de 2 KB, 23 líneas I/O de propósito general, 32 registros de propósito general, tres timers/counters con modos de comparación, interrupciones internas y externas, conversor A/D de 6 canales y 10 bits, watch dog programable con oscilador interno y cinco modos de ahorro de energía seleccionables por software.

El dispositivo funciona entre 1.8 y 5.5 Voltios. / Figura 1.4. Microcontrolador

ATMEGA328 Fuente: (Microchip, 2016) Fuente: https://www.microchip.com/wwwproducts/en/ATmega328 Módulo wi-fi ESP8266 Este módulo es cada vez más utilizado en aplicaciones IoT debido a su bajo coste y fácil conexión a una red local o salida a la nube de Internet. Esto debido a que tiene integrado el stack de protocolos TCP/IP para la conexión fuera de la red local. También incorpora el protocolo 802.11 b/g/n para la conexión inalámbrica local, eliminando además con esto, el problema de la conexión física por cable para otros dispositivos como por ejemplo placas Arduino.

La conexión con un punto de acceso wi-fi se lleva a cabo mediante comandos de texto AT a través del puerto serie y una vez establecida, se comunicará con él. / Figura 1.5. Módulo Wi-Fi ESP8266 Fuente: http://www.naylampmechatronics.com/blog/21_Tutorial-ESP8266-Parte-I.html Entre las principales características se tiene: Trabaja con el protocolo 802.11 b/g/n Integra stack de protocolos TCP/IP 64KBytes de RAM de instrucciones 96KBytes de RAM de datos Alimentación 3.3

VDC IoT Internet of Things El internet de las cosas optimiza dispositivos y sistemas que antes se conectaban mediante circuito cerrado como cámaras, sensores y les permite comunicarse globalmente a través de las redes. Se podría decir que es una red que conecta objetos físicos por medio del uso del internet y que cuentan con algún tipo de inteligencia que realice eventos específicos en función de las tareas que sean requeridas remotamente.

Los sistemas conectados a Internet de las cosas generalmente constan de 4 elementos principales: Sensor/actuador Dispositivo de comunicación Microcontrolador Fuente de alimentación Los sistemas embebidos dentro del campo del Internet de las cosas, a más de ser pequeños computadores empotrados en algo también basan su esencia en la comunicación con la red. / Figura 1.6. Internet of things Fuente: https://hipertextual.com/archivo/2014/10/internet-cosas/ API's Interfaces de Programación de Aplicaciones Son plataformas de código abierto cuyo objetivo es recoger, analizar, visualizar y manipular datos provenientes de la red. Ejemplos de este tipo de APIs serían: ThingSpeak, Xively, Cayenne, NodeRed.

Para el caso de ThingSpeak, se trata de una plataforma abierta para Internet de las Cosas que recopila, almacena, analiza, visualiza y actúa sobre la información recogida por sensores y dispositivos como hardware de código abierto como Arduino, Raspberry Pi o BeagleBone. ThingSpeak es una API conocida entre desarrolladores y dispone ya de una gran comunidad. / Figura 1.7.

Plataforma ThingSpeak Fuente: (ThingSpeak, 2018) ThingSpeak API trabaja en base a canales los cuales contienen los campos de datos, ubicación y estado Regulador de voltaje 7805 Es un dispositivo electrónico que regula el voltaje positivo de 5V a 1A de corriente. En desarrollo de PCBs o tarjetas electrónicas con programadores Pic u otro tipo de microcontroladores, se está obligado a garantizar una fuente de tensión constante, eso disminuye la posibilidad de dañar la tarjeta de un circuito debido a oscilaciones en los niveles de tensión, la forma más práctica y simple de lograr esto es mediante el Regulador de voltaje 7805.

El dispositivo cuenta con 3 pines y para la nomenclatura de las diferentes series; las primeras letras y dos números corresponden a la denominación, mientras que las dos últimas XX deben ser reemplazados por la tensión de salida requerida. Tensión de entrada Masa Tensión de salida / Figura 1.8. Regulador de voltaje LM7805 Fuente: https://www.e-ika.com/regulador-de-voltaje-lm7805-5v-1a-5uds Fundamento teórico Compartir la información de la red local con la nube de Internet Un punto importante será buscar encontrar la mejor manera de subir información generada de la red local hacia la nube de internet para que se encuentre disponible a cualquier hora desde cualquier lugar.

Existen varias maneras para conseguir esto; como son el módulo Raspberry, el módulo NodeMCU, módulos ethernet y Shield WiFi de Arduino, etc. El módulo NodeMCU tiene incorporado el microcontrolador ESP8266 el cual permite la conectividad inalámbrica a través del protocolo 802.11 y a la vez compatible con IPv4 ya que tiene incorporados los protocolos TCP/UDP/HTTP/FTP.

Esto hace que con ayuda de un ___router pueda tener conectividad hacia la nube de inter__net. El microcontrolador ESP8266 es el mas económico y muy flexible en cuanto su programación ya que se puede descargar de firmware que permite programar en lenguajes como LUA, Python, Basic o JavaScript o inclusive es compatible con IDE de Arduino.

Plataforma IoT Existen plataformas que son utilizadas en proyectos de Internet de las cosas como son: ThingSpeak (MathWorks Lab), Cayenne, Xively, Node-red (IBM), etc. Con algunas ventajas y limitantes propias de cada plataforma. También resulta importante determinar la plataforma IoT más óptima que también será clave para cumplir los objetivos propuestos en este documento.

Por ejemplo, ThingSpeak es una plataforma de Internet de las cosas de Código abierto la cual resulta amigable para conectar dispositivos Arduino, Raspberry Pi

LoRa Wan. Permite llevar registros y bases de datos de sensores en HTTP, a través de Internet o en la misma red de área local. También permite la creación de aplicaciones de registro de dispositivos como sensores, aplicaciones de localización de seguimiento y una red social de cosas con frecuentes actualizaciones de estado.

Al ser una plataforma abierta resulta económica y amigable para su programación e interpretación en su presentación de la información recolectada. CAPITULO 2. PROPUESTA De forma general se propone para el proyecto las siguientes etapas descritas a continuación: Tarjeta Sensor Transmisor (Tx) Tarjeta Receptor (Rx) Tarjeta de conexión a la nube de Internet Recopilación de Información en ThingSpeak Aplicación en MIT App Inventor La tarjeta Sensor Transmisor tiene como componente el sensor Ultrasónico HC-SR04 que se encarga de detectar cuando hay o no presencia de un vehículo en el lugar de estacionamiento.

Estos cambios de estado son procesados por el microcontrolador ATMEGA328 que pertenece a la misma tarjeta, acondicionando la señal con un 1 para el caso de Ocupado y un 0 para el caso de Libre. Posteriormente esta información es presentada en su puerto serial que está conectado con el módulo transmisor RF. Este a su vez hará el envío de la información inalámbricamente hasta la tarjeta Receptor Rx.

En la tarjeta Receptor Rx se recibe la información de todos los nodos, la cual es tomada por el microcontrolador ATMEGA328 perteneciente a esta tarjeta. Este realiza el análisis respectivo y en base a los resultados obtenidos, envía las instrucciones al Módulo de Conexión a Internet ESP8266 por medio de su puerto serial para que sea subido a la nube de Internet a través de su antena Wi-Fi y su stack de protocolos TCP-IP que vienen incorporados en el mismo.

El Módulo de Conexión a Internet ESP8266, envía la información hasta el servidor IoT ThingSpeak en la nube de Internet por medio de su conexión wi-fi y a través del stack de protocolos TCP/IP que tiene cargado en su microprocesador para su funcionamiento. Para la recopilación de Información en ThingSpeak, el servidor permite recoger los datos provenientes de los Sensores Transmisores y ser configurados de forma gráfica para que cada Sensor tenga su respectivo diagrama donde se puede apreciar los cambios de estado de 1 a 0 que representan al estacionamiento como Libre u Ocupado, en el transcurso del tiempo.

Una vez que la información ya se encuentra disponible en internet, se ejecuta la aplicación para dispositivos móviles con sistema operativo Android que disponga de Datos para su conexión a Internet y desde el mismo se puede conocer el status

del estacionamiento (Ocupado/Libre) o a su vez, ejecutar acciones como reservar o liberar una determinada plaza de estacionamiento.

El siguiente es un esquema donde se representa básicamente las tarjetas y etapas mencionadas en la descripción general anterior: / Figura 2.1. Diagrama esquemático de la red de sensores inalámbricos propuesta Fuente: Elaborado por el autor 2.1. Propuesta de la tarjeta Sensor Transmisor (Tx) 2.1.1. Etapa de alimentación de la tarjeta Sensor Transmisor (Tx) Para la alimentación de la tarjeta se va a utilizar el regulador de voltaje LM7805 para acondicionar o limitar el excedente de voltaje que llegue de la fuente de alimentación hacia su microcontrolador, el mismo que necesita para su operación 5VDC.

A continuación, se muestra las características del regulador LM7805 en el que se puede apreciar que cubre este requerimiento en el anexo XXX Data sheet LM7805 FAIRCHILD SEMICONDUCTOR (ANEXO X) Fuente: http://pdf1.alldatasheet.com/datasheet-pdf/view/82833/FAIRCHILD/LM7805.html 2.1.2. Desarrollo del monitoreo con Sensor de Ultra sonido HC-SR04 Se requiere detectar la presencia de un vehículo cuando llegue a ocupar uno de los estacionamientos, para lo cual se utilizará el sensor Ultrasónico HC-SR04, debido a sus características para medir distancias hasta un objeto.

El resultado del cambio de estado del sensor, será enviado hasta un microcontrolador que estará en la misma tarjeta de sensores. Su implementación en cuanto a la alimentación resulta sencilla, ya que el sensor ultrasónico opera con alimentación de 5VDC lo cual lo hace compatible con la tarjeta Sensor Transmisor cuya alimentación es de 5VDC. Se lo debe regular a una distancia aproximada de 2mts a partir del techo del estacionamiento hasta el techo del auto.

A continuación, los parámetros técnicos del dispositivo propuesto: Tabla 1. Características técnicas Sensor Ultrasónico HC-SR04 Working Voltage _Voltaje de trabajo _DC 5V _ _Working Current _Corriente de trabajo _15 mA _ _Working Frequency _Frecuencia de trabajo _40 Hz _ _Max Range _Distancia máxima _4 m _ _Min Range _Distancia mínima _2 cm _ _Measuring Angle _Angulo eficaz _15 degree _ _Trigger Input Signal _Disparo de señal de entrada _10 us TTL pulse _ _Echo Output Signal _Eco señal de salida _Input TTL lever signal and the range in proportion _ _Dimension _Dimensiones _45*20*15 mm _ _https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/ Sus pines de conexión son los siguientes: Vcc = 5VDC Trig = Disparo de ultrasonido Echo = Recepción de ultrasonido GND = Tierra Este dispositivo calcula la distancia en base a la siguiente fórmula matemática: ??= ???????????? ??????????

???? ???????? ?? ???? ????h?? *(????????????????? ?????? ???????????) 2 Formula:1 Funcionamiento matemático sensor ultrasónico HCSR04 Fuente: (Llamas L, 2015) https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/ y resumida de forma matemática resulta: Distancia ??= 1 2 * ??* ?? Donde: L= distancia, T= tiempo entre la emisión y la recepción, C= Velocidad del sonido. 2.1.2.1.

Microcontrolador ATMEGA328 El microcontrolador Atmega328 es compatible con IDE de Arduino. Esto hace que resulte económico ya que el hardware y software para su programación son de código abierto. En la web existe disponibilidad de información para hacer uso de librerías que facilita la programación. Estos son puntos a favor para elegir este microcontrolador.

La información que va a manejar este microcontrolador en el Módulo Sensor Transmisor, son los cambios de estado del sensor del estacionamiento, es decir; libre/ocupado (1/0), lo que hace que un microcontrolador de 8 bits sea suficiente para la carga de procesamiento y a la vez reducir el consumo de energía. Sus características técnicas hacen que resulte útil para el tratamiento de la señal ya que tiene la capacidad de ejecutar instrucciones mientras equilibra el consumo de energía y la velocidad de procesamiento.

Generalmente son usados en áreas de automatización industrial, de viviendas y edificios que se asemeja bastante al presente proyecto. Los pines que tiene este microcontrolador son los siguientes: / Figura 2.2. Distribución de pines Integrado Atmega328 Fuente: https://es.slideshare.net/vicomg/microcontroladores-ver20 Especificaciones técnicas: Voltaje de operación _5V _ _Número de Pines: _28 _ _Memoria FLASH _32KB _ _Memoria RAM _2KB _ _EEPROM _1KB _ _Máxima frecuencia de funcionamiento _20Mhz _ _CPU _8-bit AVR _ _Pines de entrada/salida _23 _ _Entradas Analógicas (ADC) _6 _ _Formato DIP _ _ _Tabla 2.

Especificaciones técnicas microcontrolador Atmega328 Fuente: http://www.naylampmechatronics.com/atmel/111-atmega328.html También resultará útil para las pruebas en protoboard cuando se requiera, ya que su encapsulado PDIP permite ser montado directamente sobre el mismo. 2.1.2.2. Transmisor RF de Radiofrecuencia Para la comunicación de los sensores con el nodo receptor se utilizarán módulos de radio frecuencia Tx/Rx 433Mhz cuya frecuencia es de banda libre.

Resulta sencillo implementar enlaces de datos de radiofrecuencia con estos módulos ya que permiten alcanzar distancias de hasta 80 mts dentro de estructuras

cerradas o 350 mts en lugares abiertos cuando opera con la fuente de 12V. Adicional el diseño de su placa sirve también para ser montado en un circuito impreso PCB. Estas características lo hacen muy popular, económico y adecuados para el presente proyecto.

Especificaciones técnicas Transmisor: Señal de radiofrecuencia _Modulación ASK _ _Fuente de alimentación _12V, 5V, 3V _ _Consumo de corriente _<16 mA _ _Potencia de transmisión _13dBm _ _Desviación de frecuencia _+ - 75kHz _ _Alcance _230 metros (5V) _ _Frecuencia _433MHz y 315 MHz _ _Velocidades de Transmisión _Hasta 20kbps _ _Tabla 3. Transmisor RF de Radiofrecuencia Fuente: http://programarpicenc.com/articulos/radiofrecuencia-sistema-tx-rx-a-433mhz/ / Figura 2.3.

Transmisor RF de Radiofrecuencia 433Mhz Fuente: https://www.luisllamas.es/comunicacion-inalambrica-en-arduino-con-modulos-rf-433mhz/ El transmisor es compatible con aplicaciones inalámbricas de enlaces de datos uno-a-uno o de varios nodos. 2.1.5. Diagrama esquemático de la Tarjeta Sensor Transmisor (Tx) A continuación, se muestra de forma esquemática cómo queda conformada la tarjeta con los dispositivos propuestos en los ítems anteriores y sus conexiones. / Figura 2.4. Diagrama esquemático de la Tarjeta Sensor Transmisor Tx Fuente: Elaborado por el autor 2.1.6.

Diseño de la placa PCB de la Tarjeta Sensor Transmisor (Tx) Este diagrama se realiza en el módulo ARES del software Proteus 7 Professional que sirve para la fabricación de placas de circuito impreso ya que facilita la edición, ubicación, ruteo de pistas de cobre. / Figura 2.5. Diagrama PCB de la tarjeta Sensor Transmisor Tx Fuente: Elaborado por el autor 2.2.

Propuesta de la Tarjeta Receptor (Rx) Para esta tarjeta, a mas de otros elementos que se mencionará más adelante, se empleará también el microcontrolador ATMEGA328 que ya fue mencionado y analizado por lo que ya no es necesario detallarlos nuevamente. Los demás elementos se indican a continuación: 2.2.1. Etapa de alimentación de la Tarjeta Receptor (Rx) Para el desarrollo de esta tarjeta se va a utilizar adicionalmente el regulador de voltaje LM1117, el cual se va a encargar de limitar o absorber todo el excedente de voltaje que llegue de la fuente de alimentación hacia el módulo wi-fi ESP8266 (detallado más adelante), ya que el mismo necesita para su operación 3 a 3,6VDC.

Con esto se consigue tener una tensión constante de 3VDC que reducirá las posibilidades de quemar los circuitos de esta tarjeta debido a oscilaciones en los

niveles de tensión. A continuación, se muestra las características del regulador LM1117 en el que se puede apreciar que cubre este requerimiento: Symbol _Parameter _Conditions _Min _Typ _Max _Units _ _V out _Output Voltage _LM1117-3.3 Iout = 10mA, Vin=5V Tj=25°C 0 = I out = 800 mA, 4.75V = Vin = 10V _3.267 _3.300 _3.333 _V _ _ _ _ _ _ _ _ _ _ _ _3.235 _3.300 _3.365 _V _ _Fuente: Datasheet http://www.soloelectronica.net/PDF/LM1117.pdf / Figura 2.6. Regulador de voltaje LM1117 https://www.microjpm.com/products/lm1117-regulador-de-voltaje-bajo/ 2.2.2.

Receptor RF de Radiofrecuencia Este dispositivo forma parte de la comunicación entre el Receptor y los sensores de tal manera que se pueda recibir los cambios de estado producidos en cada puesto de estacionamiento y enviados por los transmisores. Los módulos de Radiofrecuencia vienen como pareja tanto Transmisor como Receptor y su frecuencia de operación igual es de 433Mhz de banda libre.

Los módulos receptores pueden llegar a velocidades de hasta 4.8kbps y distancias de 40 mts dentro de edificios o 110 mts en campo abierto. Adicional el diseño de su placa sirve también para ser montado en un circuito impreso PCB. / Figura 2.7. Receptor RF de Radiofrecuencia 433Mhz Fuente: https://www.luisllamas.es/comunicacion-inalambrica-en-arduino-con-modulos-rf-433mhz/ Especificaciones técnicas Receptor: Fuente de alimentación _5V _ _Consumo de corriente _2.2 mA _ _Velocidades _Hasta 4.8kbps _ _Alcance utilizable _Hasta 110 mts _ _Frecuencia _433.92 MHz (433MHz) y 315.0MHz _ _Versiones disponibles _regulado y no regulado _ _Tabla 4.

Receptor RF de Radiofrecuencia Fuente: http://programarpicenc.com/articulos/radiofrecuencia-sistema-tx-rx-a-433mhz/ 2.3. Tarjeta de Conexión a la nube de Internet. 2.3.1. Etapa de conexión a Internet con el Módulo wi-fi ESP8266 El módulo ESP8266 es ideal para esta parte de la comunicación ya que, entre sus principales virtudes, está la de tener comunicación integrada 802.11 b/g/n que permite la salida a Internet a través de un proveedor de servicios ISP.

Otra gran ventaja es que tiene incorporado la pila de protocolos TCP/IP lo que hace que se libere la mayor parte del trabajo de comunicación del procesador. / Figura 2.8. Módulo wi-fi ESP8266 Fuente: https://polaridad.es/esp8266-modulo-wifi-elegir-caracteristicas/ El voltaje de alimentación necesario es de 3.3VDC y el consumo de corriente dependiendo de la exigencia puede ir de 50mA a 200mA (al arrancar el dispositivo).

Este voltaje será provisto por el Regulador de voltaje LM1117 el cual ya fue detallado anteriormente y estará presente en la misma placa del Receptor. En la siguiente tabla se muestran las especificaciones técnicas módulo ESP8266: Parameters _Conditions _Min _Typical _Max _Unit _ _Storage Temperature Range _ _-40 _Normal _125 _°C _ _Maximun Soldering Temperature _IPC/JEDEC J-STD-020 _ _ _260 _°C _ _Working Voltage Value _ _3.0 _3.3 _3.6

_V _ _I max _ _ _ _12 _mA _ _Electrostatic Discharge (HBM) _TAMB=25°C _ _ _2 _KV _ _Electrostatic Discharge (CDM) _TAMB=25°C _ _ _0.5 _KV _ _Fuente: https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf Protocolos soportados: 802.11 b/g/n _ _Wi-Fi Direct (P2p), Soft Access Point _ _Stack TCP/IP integrado _ _Potencia de salida: +19.5dBm en modo 802.11b _ _Sensor de temperatura integrado _ _Consumo en modo de baja energía: <10 Ua _ _64KBytes de RAM de instrucciones _ _96KBytes de RAM de datos _ _Integra núcleo de arquitectura RISC 32bits corre a 80Mhz _ _Tabla 5. Especificaciones técnicas módulo ESP8266 Fuente: https://www.geekfactory.mx/tienda/radiofrecuencia/modulo-wifi-serial-esp8266-economico/ El microcontrolador ESP8266 resulta conveniente para este proyecto también debido a que es más económico y muy flexible para su programación, ya que se puede descargar algún firmware que permite programar por ejemplo en IDE de Arduino; que es precisamente el que se utilizará para poder acceder a su configuración por comandos AT.

2.3.2. Diagrama esquemático Tarjeta Receptor (Rx) A continuación, se muestra cómo queda conformada la tarjeta con los dispositivos propuestos en los ítems anteriores y sus conexiones de forma esquemática. / Figura 2.9. Esquema Tarjeta Receptor (Rx) Fuente: Elaborado por el autor 2.3.3.

Diseño de la placa PCB de la Tarjeta Receptor (Rx) De igual manera éste diagrama se realiza en el módulo ARES del software Proteus 7 Professional. / Figura 2.10. Placa PCB de la Tarjeta Receptor (Rx) Fuente: Elaborado por el autor 2.3.4. Programación en IDE (Entorno de Desarrollo Integrado de Arduino) Para el desarrollo del programa del Atmega328 se va a utilizar el Entorno de Desarrollo Integrado IDE de Arduino, el mismo que se lo puede descargar gratuitamente de la página oficial en el siguiente link https://www.arduino.cc/en/Main/Software y se encuentra disponible para las diferentes versiones de sistemas operativos Windows, Mac OS y Linux.

Para establecer la comunicación entre los microcontroladores ATMEGA328 de las

tarjetas Sensor Tx y Receptor Rx a nivel de software se va a utilizar la librería VirtualWire ya que se trata de un firmware de código abierto que lo hace ideal para incluirlo en lenguaje IDE de Arduino. También lo podremos utilizar para configurar los demás dispositivos como son el ESP8266 y el sensor ultrasónico HC-SR04 con la diferencia que son líneas de comandos AT y a través del monitor serie.

El software de programación está basado en lenguaje C/C++ cuyo entorno es amigable, liviano y con las herramientas básicas para cargar, depurar y conectarse con la tarjeta diseñada para nuestro propósito. A continuación, la interfaz gráfica del Entorno de Desarrollo Integrado IDE Arduino: / Figura 2.11. Entorno de Desarrollo Integrado IDE Arduino Fuente: Elaborado por el autor 2.3.5.

Diagrama de flujo del programa de la Tarjeta Receptor (Rx) El microcontrolador ATMEGA328 de esta tarjeta inicia pidiendo la información del status de cada estacionamiento, recopilar esta información y a su vez comunicarla hacia el módulo ESP8266 para su envío hacia el servidor ThingSpeak. El flujo de los procesos es el siguiente: _ _ Figura 2.12. Diagrama de flujo del programa de la Tarjeta Receptor (Rx) Fuente: Elaborado por el autor 2.3.6.

Diagrama de flujo del programa de la Tarjeta Sensor Transmisor(Tx) El flujo de los procesos es el siguiente: _ Figura 2.13. Diagrama de flujo del programa de la Tarjeta Sensor Transmisor (Tx) Fuente: Elaborado por el autor 2.4. Establecimiento de la comunicación con la plataforma IoT. 2.4.1.

Servidor IoT ThingSpeak Las plataformas IoT brindan servicios para recopilar información recogida generalmente por sensores y microcontroladores, las cuales pueden ser desarrolladas por nosotros mismo o plataformas ya existentes con fines académicos o de lucro comercial. ThingSpeak es una Interfaz de Programación de Aplicaciones API y viene a ser una plataforma de código abierto al que se puede acceder desde Internet y cuyo objetivo es recoger, almacenar, visualizar y manipular datos recogidos por sensores y dispositivos con hardware de código abierto como Arduino, Raspberry Pi o BeagleBone para el desarrollo de aplicaciones IoT. / Figura 2.14. Esquema de conexiones de aplicaciones IoT con ThingSpeak Fuente: https://aprendiendoarduino.wordpress.com/tag/thingspeak/ 2.4.2.

Diseño de la aplicación en Mit App Inventor Se ha seleccionado esta solución para nuestro diseño debido a muchas ventajas entre las cuales tenemos: Mitt App Inventor es un servicio gratuito al que se accede a través de la nube y ofrece un entorno de desarrollo de aplicaciones para dispositivos móviles Android. Pese a

que se trabaja en un navegador web permite almacenar el proyecto y también el seguimiento de los mismos además que permite generar archivos apk que pueden ser instalados como aplicación en cualquier dispositivo Android. Para el acceso se lo realiza desde un navegador web y conectándose al siguiente link http://appinventor.mit.edu/explore/.

Se ingresa con una cuenta normal de Google, Gmail. Básicamente se trabaja con dos módulos: App Inventor Designer y App Inventor Blocks Editor. / Figura 2.15. Pantalla Designer Fuente: Elaborado por el autor / Figura 2.16. Pantalla Blocks Editor Fuente: Elaborado por el autor 2.4.3. Diagrama de bloques A continuación, se presenta la lógica que seguirán las diferentes etapas del proyecto. / Figura 2.17.

Pantalla Blocks Editor Fuente: Elaborado por el autor 2.4.4. Diagrama del estacionamiento seleccionado para pruebas En la siguiente ilustración se visualiza de manera esquemática la distribución de 8 de las plazas de estacionamiento del Conjunto Habitacional Camino Real ubicado en las calles García Moreno Y Esmeraldas en pleno centro histórico de la ciudad de Quito y la distribución de los sensores que como posición estratégica estaría en el techo del estacionamiento para detectar el auto desde arriba. / Figura 2.18.

Diagrama parcial del parqueadero del Conjunto Habitacional Camino Real Fuente: Elaborado por el autor CAPITULO 3. IMPLEMENTACIÓN 3.1. Fabricación de las placas de Sensores Tx y nodo receptor Rx: Esta placa se lo realizó en base a la impresión sobre el papel Transfer del diagrama PCB del capítulo 2, el mismo que sirvió para delinear las pistas en la baquelita también con la ayuda del marcador en los puntos donde hubo fallas en la adhesión del papel Transfer.

Luego se procedió a aplicar la solución de cloruro férrico para definir las pistas de cobre y retirar el resto de material que no servía en la baquelita. / Figura 3.1. Sumersión de la baquelita en ácido Fuente: Elaborado por el autor, Luego se procedió a limpiar el papel y la tinta del marcador con una esponja metálica hasta verificar que las pistas de cobre tomen el brillo metálico. Se procedió a realizar los orificios donde van a calzar los diferentes dispositivos con la ayuda de un taladro.

/ Figura 3.2. Realización de los orificios con el taladro Fuente: Elaborado por el autor, Después se procedió a soldar con el estaño todos los elementos en la tarjeta cuidando de no recalentar las pistas para que las mismas no se levanten. / Figura 3.3.

Suelda de elementos en la placa con estaño Fuente: Elaborado por el autor Una vez

que se tiene listas las tarjetas, se puede adecuar cajas que contendrán a las mismas en su interior, facilitando su manipulación. Se utilizó cajas plásticas para proyectos de 19x10x6 cm que se encuentran en tiendas electrónicas y se las adecuó con aberturas para los sensores ultrasónicos y también a fijar los leds de alto brillo por la parte frontal (Rojo para ocupado y Verde para disponible) como se muestra a continuación: / Figura 3.4

Colocación de luces en la caja de proyectos Fuente: Elaborado por el autor Luego se procedió a colocar lunas reflectivas en la parte frontal de la caja para amplificar la luz de los leds y sean más visibles. / Figura 3.5. Colocación de lunas reflectivas para visualización frontal de la señalización Fuente: Elaborado por el autor En la parte posterior de las cajas, se colocó terminales portajacks tipo hembra para la conexión de la alimentación de 110-120 VAC / Figura 3.6.

Colocación de portajacks tipo hembra para alimentación Fuente: Elaborado por el autor Las tarjetas Sensor Transmisor (Tx) fueron sujetadas a la tapa superior por medio de postes metálicos que al mismo tiempo sirvieron para nivelar los sensores ultrasónicos en los orificios preparados en las cajas para aquello. / Gráfico 3.7. Sujeción de las tarjetas con postes metálicos Fuente: Elaborado por el autor Se procedió a sujetar la fuente de alimentación DC de la tarjeta a la caja por su interior y colocar mediante borneras el cable USB que sirvió para poder conectar a la fuente.

A su vez la fuente fue soldada a los terminales portajacks y colocados para poder alojar la tarjeta junto con el sensor y la tapa por la parte superior. / Gráfico 3.8. Disposición de los elementos dentro de la caja de proyectos Fuente: Elaborado por el autor 3.2. Programación del módulo wi-fi ESP8266 de la Tarjeta Receptor (Rx) Por otro lado, para la programación del ESP8266 se procedió a armar el siguiente circuito provisional para poder obtener 3VDC de alimentación el cual nos lo puede proveer una placa Arduino (únicamente para la parte de la programación) en uno de sus pines retirando previamente el microcontrolador. / Gráfico 3.9.. Conexiones para configurar el ESP8266 Fuente: https://www.prometec.net/arduino-wifi/# / Figura 3.10..

ESP8266 conectado para ser configurado con comandos AT Fuente: Elaborado por el autor Mediante comandos AT y a través del IDE de Arduino, se procedió a configurar los principales parámetros como el SSID de la red de la oficina de la administración del Conjunto, ya que el mismo se encuentra a la entrada de los estacionamientos que sirvieron de prueba. También se configuró el password, el puerto 80 como se muestra a continuación: 3.2.1

Configuración de la red WiFi Los comandos AT ejecutados en la consola del IDE Arduino para conseguir obtener una IP del router que brinda Internet, son los siguientes AT _Prueba la comunicación serial _ _AT+CWLAP _Busca las redes WiFi existentes a nuestro alrededor _ _AT+CWJAP="Camino Real","camino2016" _Configura el SSID y la contraseña de acceso a la red WiFi _ _AT+CIPSERVER=1,80 _Abre el puerto 80 _ _ / Figura 3.11.

Búsqueda de las redes wi-fi disponibles Fuente: Elaborado por el autor / Figura 3.12. Conexión del módulo ESP8266 con la red WiFi del conjunto Camino Real Fuente: Elaborado por el autor 3.2.2. Probando conectividad en la red WiFi Podemos enviar un mensaje de prueba desde el navegador web apuntando a la IP asignada al ESP8266 y por el puerto 80.

Los comandos AT y el socket completo sería el señalado a continuación: AT+CIFSR _Muestra entre otra información la IP asignada al dispositivo dentro de la red WiFi _ _AT+CIPSERVER=1,80 _Abre el puerto 80 _ _ / Figura 3.13. Pruebas de conectividad del ESP8266 con la red WiFi Fuente: Elaborado por el autor 3.3. Programación de las tarjetas Sensor Transmisor Tx Para la programación de los microcontroladores incluimos la librería VirtualWire para que se pueda establecer la comunicación entre los microcontroladores ATMEGA328 de las Tarjetas Sensor Transmisor Tx con la Tarjeta Receptor Rx.

La librería será la encargada de tramitar las funciones de los radios 433MHz como lo son la recepción y envío de paquetes de datos, la comprobación de errores entre otras. Programa para el sensor 1 Solo por motivos de programación del microcontrolador ATMEGA328 se utilizó la placa Arduino Uno para luego ser puesto en funcionamiento en la Tarjeta Sensor Transmisor (Tx) que fue diseñada anteriormente. En base a esto, elegimos en el IDE Arduino la tarjeta Arduino UNO. / Figura 3.14.

Configurando IDE Arduino para programar el microcontrolador ATMEGA328 Fuente: Elaborado por el autor / Figura 3.15. Selección del puerto com para la comunicación serial Fuente: Elaborado por el autor Al incluir la librería "VirtualWire.h" por defecto utiliza con los pines 11 y 12 de la tarjeta Sensor TX.

pinMode(11) _Activación del pin 11 para la Recepción _ _ pinMode(12) _Activación del pin 12 para la Transmisión _ _ vw_setup(2000); _Espera tiempo de 2 segundos e Inicia comunicación con el módulo RF _ _ vw_rx_start(); _Inicia la función de recepción para hacerlo bidireccional _ _ proceso = EEPROM.read(10); _Lee el último

estado guardado en la memoria al momento de reiniciar _ _ Para establecer comunicación entre los pines de la tarjeta Sensor Transmisor y los sensores ultrasónicos tenemos: pinMode(9, OUTPUT); _Activación del pin 9 como salida: para los sensores _ _ pinMode(8, INPUT); _Activación del pin 8 como entrada: para los sensores _ _ Con el programa ya a detalle tenemos para el encabezado y definición de variables, el void setup quedó de la siguiente manera: #include <VirtualWire.h> void setup() { // enable debug serial Serial.begin(9600); pinMode(ledr,OUTPUT); pinMode(ledv,OUTPUT); digitalWrite(ledr, LOW); digitalWrite(ledv, LOW); pinMode(8, OUTPUT); /*activación del pin 9 como salida: para el pulso ultrasónico*/ pinMode(9, INPUT); Serial.write("Sistema encendido\n"); vw_setup(2000); //Espera un tiempo de retardo de 2 segundos para permitir que el Rx esté listo para recibir datos vw_rx_start(); proceso = EEPROM.read(10); } / Figura 3.16.

Carga del código en el microcontrolador Fuente: Elaborado por el autor Para el void loop la estructura está basada en la lógica seguida en el Diagrama de flujo del programa de la Tarjeta Sensor Transmisor(Tx) descrita en la sección 2.3.5. de este documento. Se realizó bloques de instrucciones denominadas procesos (0,1 y 2) y por medio de subrutinas se realizan diferentes acciones; resumidas de la siguiente manera: Proceso 0: Determina en función de las mediciones de su sensor ultrasónico y la subrutina "análisis" que realiza comparaciones; si existe presencia o ausencia de un auto y colocando dicho estado de manera visible en los pines que corresponden a los leds. (Verde para libre y rojo para ocupado).

Este estado también se guarda en la variable llamada state (pudiendo ser state1=1 para ocupado o estate1=0 para libre). Proceso 1: Entra a funcionar cuando le llega un pedido de la tarjeta de control de estados (código pediq) para hacer realizar un cambio de estado (a reservado); por lo que entra a analizar si se encuentra disponible u ocupado para poder ejecutar dicho cambio y por medio de la subrutina "análisis 2".

Para hacer visible si pudo ejecutar el cambio de estado ejecuta en los pines de salida del led rojo un temporizador de manera que se vea un rojo parpadeante. Proceso 2: Entra a funcionar cuando Rx hace la consulta al Tx sobre el estado del parqueadero y éste envía como respuesta un código, si está ocupado (proca) o si se encuentra libre (procb). Luego de enviar la respuesta nuevamente regresa a la subrutina Proceso 0 para mantener de forma permanente ejecutando este loop.

A continuación, se muestra el código de programación del void loop donde se puede ver de manera detallada los procesos y subrutinas descritos en los

numerales anteriores. void loop() { if(proceso==0){ medida1(); analisis(); } if(proceso==1){ digitalWrite(ledr, HIGH); digitalWrite(ledv, LOW); medida1(); analisis2(); digitalWrite(ledr, LOW); digitalWrite(ledv, LOW); delay(50); } if(proceso==2){ delay(200); if(state1==1){ send("proca"); asm volatile (" jmp 0"); } if(state1==0){ send("procb"); asm volatile (" jmp 0"); } proceso=h; } if (vw_get_message(message, &messageLength)) { if(comparar("parq1") == 0){ delay(10); h=proceso; proceso=2; for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW); delay(50); } } else if(comparar("pediq") == 0){ delay(10); proceso=1; for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW); delay(50); } EEPROM.write(10, proceso); asm volatile (" jmp 0"); } else if(comparar("pedia") == 0){ delay(10); proceso=0; for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW); delay(50); } EEPROM.write(10, proceso); asm volatile (" jmp 0"); } } } void medida1(){ digitalWrite(8,LOW); /* Por cuestión de estabilización del sensor*/ delayMicroseconds(5); digitalWrite(8, HIGH); /* envío del pulso ultrasónico*/ delayMicroseconds(10);//pausa de 10 microsegundos tiempo=pulseIn(9, HIGH); /* Función para medir la longitud del pulso entrante.

Mide el tiempo que transcurrido entre el envío del pulso ultrasónico y cuando el sensor recibe el rebote, es decir: desde que el pin 8 empieza a recibir el rebote, HIGH, hasta que deja de hacerlo, LOW, la longitud del pulso entrante*/ distancia= int(0.017*tiempo); /*fórmula para calcular la distancia obteniendo un valor entero*/ /*Monitorización en centímetros por el monitor serial*/ //Serial.println("Distancia ");//Imprime en pantalla la palabra "Distancia" //Serial.println(distancia);//Imprime el valor de distancia en centímetros //Serial.println(" cm"); delay(100); } void analisis(){ if (distancia <= 130){ //send("procesoa"); //Serial.println("procesoa"); state1=1; digitalWrite(ledr, HIGH); digitalWrite(ledv, LOW); } if (distancia > 130){ //send("procesob"); //Serial.println("procesob"); state1=0; digitalWrite(ledr, LOW); digitalWrite(ledv, HIGH); } } void pausa2(){ delay (100); } void analisis2(){ pausa2(); state1=1; // send("procesoa"); if (distancia <= 130){ proceso=0; EEPROM.write(10, proceso); } } //Funcion para enviar el mensaje void send (char *message) { vw_send((uint8_t *)message, strlen(message)); //Envia el mensaje vw_wait_tx(); //Espera hasta que se haya acabado de transmitir todo Serial.println(message); //Muestra el mensaje por Serial } char comparar(char* cadena) { //Esta funcion compara el string cadena con el mensaje recibido. //Si son iguales, devuelve 1.

Si no, devuelve 0. for(int i = 0; i<messageLength; i++) { if(message[i] != cadena[i]) { return 1; } } return 0; } 3.4. Programación Tarjeta Receptor Rx Para el encabezado y void setup lo principal en resaltar serían las siguientes instrucciones: Se incluyó también la librería “VirtualWire.h” que por defecto utiliza los pines 11 y 12 de la

Tarjeta Receptor Rx para la comunicación a nivel de Radio Frecuencia con las tarjetas Sensor Tx. #include <SoftwareSerial.h> _Para la comunicación serial con el ESP8266 por los pines 6y5 _ _SoftwareSerial Serial2x(6,5); _Se declaró el pin 5 rx y 6 tx _ _#include <stdlib.h> _Librería estándar para funciones básicas de comunicación con ThingSpeak _ _Serial2x.begin(9600); _Para comunicación del micro controlador y el módulo ESP8266 _ _ La comunicación con ThingSpeak se puede realizar a partir de configurar el API Key gracias a la librería String apiKey = "HG0ZPK7ADBT9FG8T" _Ingresa la API KEY que se generó al crear nuestro canal _ _ / Figura 3.17.

API de Thingpeak programado en el código de la Tarjeta Receptora Rx Fuente: Elaborado por el autor Se define como variables a los estados que le llegarán desde cada Sensor Tx de la red. String state1; _ _String state2; _ _String state3; _ _String state4; _ _String state5; _ _String state6; _ _String state7; _ _String state8; _ _ El código de programación en lo que respecta al void setup quedaría de la siguiente manera: #include <SoftwareSerial.h> #include <stdlib.h> #include <VirtualWire.h> //La constante VW_MAX_MESSAGE_LEN viene definida en la libreria byte message[VW_MAX_MESSAGE_LEN]; byte messageLength = VW_MAX_MESSAGE_LEN; int sensor1State=0; // Variables int PulseSensorPurplePin = 0; // Pulse Sensor PURPLE WIRE connected to ANALOG PIN 0 int ledr = 2; // The on-board Arduion LED int ledv = 3; int Signal; // holds the incoming raw data. Signal value can range from 0-1024 int Threshold = 550; // Determine which Signal to "count as a beat", and which to ingore.

int dato=0; int BPM; // replace with your channel's thingspeak API key String apiKey = "HG0ZPK7ADBT9FG8T"; SoftwareSerial Serial2x(6,5);//Declaramos el pin 5 rx y 6 tx int i=1; const int sensorPin= A0; // On Arduino: 0 - 1023 maps to 0 - 5 volts #define VOLTAGE_MAX 5.0 #define VOLTAGE_MAXCOUNTS 1023.0 long distancia;//Variable para almacenar el valor de la distancia long tiempo; String state1; String state2; String state3; String state4; String state5; String state6; String state7; String state8; int g; int pedido=1; int h; // this runs once void setup() { // enable debug serial Serial.begin(9600); // enable software serial Serial2x.begin(9600);//Iniciamos el puerto serie del gps // reset ESP8266 //Serial2.println("AT+RST"); pinMode(ledr,OUTPUT); pinMode(ledv,OUTPUT); digitalWrite(ledr, LOW); digitalWrite(ledv, LOW); pinMode(9, OUTPUT); /*activación del pin 9 como salida: para el pulso ultrasónico*/ pinMode(8, INPUT); Serial.write("Sistema encendido\n"); vw_setup(2000); vw_rx_start(); } Para la parte del void loop en esencia se realiza la lógica detallada en el diagrama de flujo de la Tarjeta Receptor Rx; donde básicamente esta tarjeta realiza el pedido del estado en que se encuentra cada Sensor Tx por medio de códigos “parq1 hasta parq8”.

Los códigos “parq1 hasta el 8” son enviados dentro de la subrutina ejecución de instrucciones que tienen por nombre “pedido=n” donde n va de 1 a 16. Estos pasos se ejecutan uno por el envío del código y otro por el análisis del código recibido por respuesta a “parq1”. Nuevamente se ejecuta el proceso con el envío del código “parq2” para la consulta de estado del parqueadero 2.

Cuando se recibe la respuesta de cada Sensor Tx, este recibe por código proca (ocupado) o procb (libre) y según esto guarda los estados de los 8 parqueaderos con el nombre state1 hasta state8 que luego nos sirve para el envío hacia ThingSpeak. Al final de ejecutar pedido=16 este pasa a convertirse en pedido=0 en donde se ejecuta el envío de los states por medio de comandos AT los cuales se encargan de la conexión TCP con la IP de ThingSpeak, abrir el puerto 80 y subir los states al servidor.

El código de programación para la parte del void loop finalmente quedó de la siguiente manera: void loop() { if(pedido==1){ send("parq1"); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW); delay(50); } pedido=2; } if(pedido==2){ send("parq1"); for(h=0;h<100;h++){ if (vw_get_message(message, &messageLength)) { if(comparar("proca") == 0){ Serial.println("proca"); state1="1"; pausa(); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(10); digitalWrite(ledv,LOW); delay(10); } pedido=3; h=1500; delay(300); } else if(comparar("procb") == 0) { Serial.println("procb"); state1="0"; pausa(); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(10); digitalWrite(ledv,LOW); delay(10); } pedido=3; h=1500; delay(300); } } delay(10); } } if(pedido==3){ send("parq2"); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW); delay(50); } pedido=4; } if(pedido==4){ send("parq2"); for(h=0;h<200;h++){ if (vw_get_message(message, &messageLength)) { if(comparar("procc") == 0){ Serial.println("procc"); state2="1"; pausa(); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(10); digitalWrite(ledv,LOW); delay(10); } pedido=5; h=1500; delay(300); } else if(comparar("procd") == 0) { Serial.println("procd"); state2="0"; pausa(); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(10); digitalWrite(ledv,LOW); delay(10); } pedido=5; h=1500; delay(300); } } delay(10); } } if(pedido==5){ send("parq3"); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW); delay(50); } pedido=6; } if(pedido==6){ send("parq3"); for(h=0;h<200;h++){ if (vw_get_message(message, &messageLength)) { if(comparar("proce") == 0){ Serial.println("proce"); state3="1"; pausa(); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(10); digitalWrite(ledv,LOW); delay(10); } pedido=7; h=1500; delay(300); } else if(comparar("procf") == 0) { Serial.println("procf"); state3="0"; pausa(); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(10);

```
digitalWrite(ledv,LOW); delay(10); } pedido=7; h=1500; delay(300); } } delay(10); } }
if(pedido==7){ send("parq4"); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50);
digitalWrite(ledv,LOW); delay(50); } pedido=8; } if(pedido==8){ send("parq4");
for(h=0;h<200;h++){ if (vw_get_message(message, &messageLength)) {
if(comparar("procg") == 0){ Serial.println("procg"); state4="1"; pausa();
for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(10); digitalWrite(ledv,LOW);
delay(10); } pedido=9; h=1500; delay(300); } else if(comparar("proch") == 0) {
Serial.println("proch"); state4="0"; pausa(); for(g=0;g<1;g++){
digitalWrite(ledv,HIGH); delay(10); digitalWrite(ledv,LOW); delay(10); } pedido=9;
h=1500; delay(300); } } delay(10); } } if(pedido==9){ send("parq5");
for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW);
delay(50); } pedido=10; } if(pedido==10){ send("parq5"); for(h=0;h<200;h++){ if
(vw_get_message(message, &messageLength)) { if(comparar("proci") == 0){
Serial.println("proci"); state5="1"; pausa(); for(g=0;g<1;g++){
digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW); delay(50); } pedido=11;
h=1500; delay(300); } else if(comparar("procj") == 0) { Serial.println("procj");
state5="0"; pausa(); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50);
digitalWrite(ledv,LOW); delay(50); } pedido=11; h=1500; delay(300); } } delay(10); } }
if(pedido==11){ send("parq6"); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH);
delay(50); digitalWrite(ledv,LOW); delay(50); } pedido=12; } if(pedido==12){
send("parq6"); for(h=0;h<200;h++){ if (vw_get_message(message,
&messageLength)) { if(comparar("prock") == 0){ Serial.println("prock"); state6="1";
pausa(); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50);
digitalWrite(ledv,LOW); delay(50); } pedido=13; h=1500; delay(300); } else
if(comparar("procl") == 0) { Serial.println("procl"); state6="0"; pausa();
for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW);
delay(50); } pedido=13; h=1500; delay(300); } } delay(10); } } if(pedido==13){
send("parq7"); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50);
digitalWrite(ledv,LOW); delay(50); } pedido=14; } if(pedido==14){ send("parq7");
for(h=0;h<200;h++){ if (vw_get_message(message, &messageLength)) {
if(comparar("procm") == 0){ Serial.println("procm"); state7="1"; pausa();
for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW);
delay(50); } pedido=15; h=1500; delay(300); } else if(comparar("procn") == 0) {
Serial.println("procn"); state7="0"; pausa(); for(g=0;g<1;g++){
digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW); delay(50); } pedido=15;
h=1500; delay(300); } } delay(10); } } if(pedido==15){ send("parq8");
for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW);
delay(50); } pedido=16; } if(pedido==16){ send("parq8"); for(h=0;h<200;h++){ if
(vw_get_message(message, &messageLength)) { if(comparar("proco") == 0){
Serial.println("proco"); state8="1"; pausa(); for(g=0;g<1;g++){
```

digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW); delay(50); } pedido=0; h=1500; delay(300); } else if(comparar("procp") == 0) { Serial.println("procp"); state8="0"; pausa(); for(g=0;g<1;g++){ digitalWrite(ledv,HIGH); delay(50); digitalWrite(ledv,LOW); delay(50); } pedido=0; h=1500; delay(300); } } delay(10); } } if(pedido==0){ // TCP connection String cmd = "AT+CIPSTART=\"TCP\",\""; cmd += "184.106.153.149"; // api.thingspeak.com cmd += "\",80"; Serial2x.println(cmd); Serial.println(cmd); if(Serial2x.find("Error")){ Serial.println("AT+CIPSTART error"); return; } // prepare GET string String getStr = "GET /update?api_key="; getStr += apiKey; getStr +="&field1="; getStr += String(state1); getStr +="&field2="; getStr += String(state2); getStr +="&field3="; getStr += String(state3); getStr +="&field4="; getStr += String(state4); getStr +="&field5="; getStr += String(state5); getStr +="&field6="; getStr += String(state6); getStr +="&field7="; getStr += String(state7); getStr +="&field8="; getStr += String(state8); getStr += "\r\n\r\n"; // send data length cmd = "AT+CIPSEND="; cmd += String(getStr.length()); Serial2x.println(cmd); Serial.println(cmd); if(Serial2x.find(">")){ Serial2x.print(getStr); Serial.print(getStr); } else{ Serial2x.println("AT+CIPCLOSE"); // alert user Serial.println("AT+CIPCLOSE"); } // thingspeak needs 15 sec delay between updates delay(2000); pedido=1; asm volatile (" jmp 0"); } } void pausa(){ delay (5); } //Funcion para enviar el mensaje void send (char *message) { vw_send((uint8_t *)message, strlen(message)); //Envia el mensaje vw_wait_tx(); //Espera hasta que se haya acabado de transmitir todo Serial.println(message); //Muestra el mensaje por Serial } char comparar(char* cadena) { //Esta funcion compara el string cadena con el mensaje recibido.

//Si son iguales, devuelve 1. Si no, devuelve 0. for(int i = 0; i<messageLength; i++) { if(message[i] != cadena[i]) { return 1; } } return 0; } 3.5. Programación del microcontrolador de la Tarjeta de cambio de estados Se comunicó con el microcontrolador a través del puerto COM3 y se incluyó la librería ESPDUINO que contiene el firmware para establecer la comunicación con el módulo ESP8266 perteneciente a la tarjeta de cambio de estados.

Se procedió a compilar el programa y a subirlo en el microcontrolador de la tarjeta de cambio de estados como se muestra a continuación: / Figura 3.18.. Inclusión de la librería ESPDUINO al programa Fuente: Elaborado por el autor Establecemos la conexión del módulo WiFi ESP8266 que pertenece a esta tarjeta con el SSID del router del ISP, el mismo que brinda el servicio de Internet al conjunto Camino Real junto con la contraseña de acceso y cargamos el programa que se encuentra incluido en el Anexo "Programa".

El router se encuentra ubicado en la oficina del administrador del conjunto que

también se encuentra próximo a los estacionamientos del conjunto. / Figura 3.19. Configuración del SSID y contraseña del WiFi y subida del programa en el microcontrolador de cambios de estados Fuente: Elaborado por el autor Para permitir la programación se incluyó las siguientes librerías: / Figura 3.20. Librerías utilizadas Fuente: Elaborado por el autor 3.6. Subir la información a ThingSpeak 3.6.1.

Adquisición de licencia para servidor IoT denominado Thingspeak ThingSpeak brinda la posibilidad de trabajar con licencia gratuita, pero con ciertas limitaciones. En la figura en cambio se muestran los diferentes tipos de licencia con los alcances y términos de uso correspondientes. / Figura 3.21 Tipos de licencias que ofrece ThingSpeak Fuente: Elaborado por el autor De manera explícita se muestra en el gráfico a continuación, las ventajas de utilizar la licencia "STUDENT" sobre la licencia gratuita. / Figura 3.22 Ventajas de la licencia "Student" sobre la licencia "Free" Fuente: Elaborado por el autor Se llena la información requerida para la adquisición de la licencia / / Figura 3.23.

Llenando Datos para obtener la licencia "Student" Fuente: Elaborado por el autor Finalmente se obtiene la confirmación de que la licencia ya se encuentra vigente. / Figura 3.23 Confirmación de Activación de licencia Fuente: Elaborado por el autor _ 3.6.2. Creación de fields para cada estacionamiento Se ingresa en el link: https://thingspeak.com/login para crear nuestro canal llenando los siguientes campos / Figura 3.24.

Creación del canal y de las gráficas de los sensores Fuente: Elaborado por el autor También se debe tener en cuenta el APIkey con el que el servidor establece conexión con el microcontrolador y poder subir la información al servidor pudiendo ser estos de lectura y escritura. En la gráfica a continuación se puede ver el API generado en ThingSpeak y la configuración en el programa de la Tarjeta Receptor (Rx) / Figura 3.25.

Generación del API key en ThingSpeak Fuente: Elaborado por el autor / Figura 3.26. Configuración del API key en el programa de la Tarjeta Receptor (Rx) Fuente: Elaborado por el autor Una vez establecida la conexión con ThingSpeak se empiezan a generar las gráficas correspondientes a cada Field el cual fue asignado a los sensores.

En primera instancia se consiguió lecturas de los sensores 1 y 3 en donde se procedió a probar los sensores manualmente mediante obstrucción para simular la presencia de un auto. A continuación, las gráficas obtenidas de las pruebas

realizadas: / Figura 3.27. Lecturas del sensor 1 en el Field1 de ThingSpeak Fuente: Elaborado por el autor / Figura 3.28.

Lecturas del sensor 3 en el Field1 de ThingSpeak Fuente: Elaborado por el autor Se verifica que según la programación en el microcontrolador; cuando se coloca una obstrucción frente al sensor la gráfica se coloca en 1, mientras que al retirar el obstáculo la gráfica nuevamente regresa a 0. Finalmente, y gracias al programa del microcontrolador en conjunto con las librerías VirtualWire.h

se verifica que se ha establecido la transferencia de la información de los estados de cada sensor hacia el servidor IoT ThingSpeak. Se puede ya visualizar por medio de un 1 del servidor que se tiene un obstáculo en el sitio donde se encuentra el sensor y un 0 nos indica que no existe ningún tipo de obstáculo en el sitio del sensor.

A continuación, se muestra las gráficas generadas por la información proveniente de los 8 sensores armados y configurados: / / Figura 3.29 Muestra de la información proveniente de los sensores en el servidor IoT ThingSpeak Fuente: Elaborado por el autor Se crea el canal y queda registrada la fecha en el servidor / Figura 3.30 Registro de fecha de creación del canal en el servidor Fuente: Elaborado por el autor 3.7.

Diseño e implementación de la aplicación de control de administrador en App Inventor 3.7.1. Pantalla de presentación Para la parte del diseño, la primera pantalla de presentación muestra el logo de la Universidad y el título del proyecto / Figura 3.31. Pantalla de presentación Fuente: Elaborado por el autor 3.7.2.

Pantalla VISUALIZADOR_ESTACIONAMIENTOS Se despliega la pantalla para visualizar los estacionamientos y los botones para acceder mediante credenciales a cuatro ambientes: Botón THINGSPEAK: Permite mediante credenciales de administrador, acceder a la pantalla HOME_THINGSPEAK que permite navegar en el servidor del mismo nombre y poder pasar por sus gráficas y bases de datos, Botón CONTROL DE RESERVAS: Permite mediante credenciales de administrador, acceder a la pantalla AREA_DE_RESERVAS para tener control de los estados de los sensores.

Banner de loggeo: Permite mediante credenciales de administrador y de usuario acceder al visualizador de estacionamientos, Botón VISUALIZADOR DE ESTACIONAMIENTOS 2: Permite mediante credenciales de administrador y de usuario acceder de manera directa a la pantalla de visualización de estacionamientos. Al clickear este botón se llama a la función Activity Starter que

nos dirige de manera directa al visualizador descrito en el ítem anterior. / Figura 3.32.

Pantalla Visualizador de Estacionamientos Fuente: Elaborado por el autor 3.7.3. Pantalla PASSWORD_2 y PASSWORD_3: Son las pantallas de seguridad para una vez autenticadas las credenciales ingresadas se pueda acceder a los ambientes ThingSpeak, Control de Reservas y Visualizador de Estacionamientos: / Figura 3.33. Designer Pantalla Password_2 Fuente: Elaborado por el autor / Figura 3.34.

Block Diagram Pantalla Password_2 Fuente: Elaborado por el autor 3.7.4. Pantalla AREA_DE_RESERVAS: Muestra los botones de control directo para reservar o liberar cada estacionamiento y seteo de los estados de los sensores y por medio de las API's (Application Programming Interfaces) de ThingSpeak que actúan desde esta aplicación hasta los sensores.

Cada API realiza un determinado cambio de estado y el mismo es identificado por un botón que lleva el nombre de la acción que realiza como Reservar Parqueadero, Liberar Parqueadero y Setear estado. Designer: / Figura 3.35. Designer Pantalla AREA_RESERVAS Fuente: Elaborado por el autor Blocks / Figura 3.36.. Block Diagram Pantalla AREA_RESERVAS Fuente: Elaborado por el autor 3.7.5.

Pantalla HOME THINGSPEAK: Permite acceso a las gráficas originales de los estados de cada sensor (FIELDs) que representan los sitios de estacionamiento y control de las configuraciones permitidas por la licencia "40700874 STUDENT" en el servidor. Los botones configurados son: Control Reservas Salir (cerrar sesión) Adelante (>)Atrás (<) Actualizar página (flecha circular) Designer / Figura 3.37.

Designer Pantalla HOME_THINGSPEAK Fuente: Elaborado por el autor Blocks / Figura 3.38 Block Diagram Pantalla HOME_THINGSPEAK Fuente: Elaborado por el autor 3.7.6. Pantalla VISUALIZADOR_ESTACIONAMIENTOS: En esta parte se muestra el estado de ocupación de los estacionamientos de una manera más comprensible para el usuario común. Es decir; si por ejemplo para la gráfica del Field 1 (Parqueadero 1) el estado "0" representa a estado libre; en esta pantalla en cambio aparecerá la palabra disponible en color verde y para la gráfica del Field 1 el estado "1" representa a estado ocupado o reservado en cambio en esta pantalla aparecerá un auto con el botón estadísticas en color rojo. / / Figura 3.39. Pantalla Visualización de Estacionamientos Fuente: Elaborado por el autor Designer / Figura 3.40.

Designer Pantalla Visualizador de estacionamientos Fuente: Elaborado por el autor

Blocks / Figura 3.41. Block Diagram Visualizador de Estacionamientos Fuente: Elaborado por el autor 3.8. Implementación del prototipo en el parqueadero de prueba Para probar el funcionamiento del prototipo se pudo tener acceso a las instalaciones del estacionamiento del conjunto habitacional "Camino Real" ubicado en el Centro Histórico de la ciudad de Quito.

El gráfico de las instalaciones en conjunto con los del prototipo mostrados en archivo Autocad, se muestra en la siguiente figura: / Figura 3.42. Plano de Parqueaderos de Prueba Fuente: Elaborado por el autor 3.8.1. __Instalación de las tomas de alimentación: Para energizar a los sensores se utilizó el cableado ya existente en la misma infraestructura tomando únicamente el voltaje fase-neutro (110-120V). Se ubicaron retornos para poder resetear a los sensores en caso de requerirlo.

Para esto se pudo utilizar la misma tubería eléctrica ya existente para los sensores de iluminación y con la ayuda de la sonda plástica para guiar los retornos a través del tubo. Desmontando momentáneamente los sensores de movimiento permitimos el paso de la sonda guía y el cable para retornos. / Figura 3.43. Pasando la sonda guía por la tubería de los sensores de movimiento Fuente: Elaborado por el autor 3.8.2.

Ubicación de la canaleta plástica 20x12 Se procedió a pegar la canaleta en la superficie del techo, para luego reforzarla con tornillos una longitud aproximada de 2mts a cada lado del cajetín central hasta llegar al centro de cada estacionamiento, luego se procedió a pasar los cables de alimentación y a colocar las tapas correspondientes como se indica en la siguiente figura: / Figura 3.44.

Instalación de canaleta plástica Fuente: Elaborado por el autor Se procedió a colocar los respectivos taco-fischer y se sujetó las cajas de sensores con tornillos bien sujetos al techo para luego proceder a conectar la alimentación de 110-120VAC quedando finalmente energizados. / / Figura 3.45. Sujeción y alimentación de los Sensores Tx Fuente: Elaborado por el autor De igual manera se colocó la tarjeta Receptor Rx y la tarjeta controladora de Estados en una caja de proyectos sujetas de igual manera con postes metálicos a la base de la caja y todo el armazón con soportes necesarios para poder ser sujetado con tornillos.

Para determinar el mejor sitio de ubicación de la caja módulo Receptor RX se probó en 3 sitios estratégicos como se indica a continuación: En el centro de la red / Figura 3.46. Pruebas con la Tarjeta Receptor RX en el centro de la red Fuente: Elaborado por el autor En el extremo superior de la red: / Figura 3.47. Pruebas con

la Tarjeta Receptor RX en la parte superior de la red Fuente: Elaborado por el autor En el extremo inferior de la red / Figura 3.48.

Pruebas con la Tarjeta Receptor RX en la parte inferior de la red Fuente: Elaborado por el autor Los mejores resultados se obtuvieron con la opción a; es decir, el módulo Receptor Rx en el centro de la red por lo que el dispositivo fue colocado en la viga central de los parqueaderos como se muestra en la siguiente figura: / Figura 3.49. Colocación de la Tarjeta Receptor en el centro del parqueadero Fuente: Elaborado por el autor 3.9.

Presupuesto a) Presupuesto para la parte del dispositivo PRESUPUESTO PIC PROTOTIPO DE RED INALÁMBRICA PARA MONITOREO POR SMARTPHONE _ _ITEM _CANTIDAD _DESCRIPCION _V. UNITARIO _V. TOTAL _ _1 _13 _Integrado ATMEGA328 _8,00 _104 _ _2 _8 _Sensor ultra sónico HC-SR04 _5,00 _40 _ _3 _9 _Cristal 16MHz _0,74 _6,66 _ _4 _18 _Condensador 22pf/50V _0,15 _2,7 _ _5 _20 _Zócalo 14 pines _0,15 _3 _ _6 _14 _Condensador 104 _0,15 _2,1 _ _7 _9 _Integrado LM7805 _0,71 _6,39 _ _8 _11 _Integrado LM1117 _1,42 _15,62 _ _9 _20 _Condensador 10uf 50V _0,19 _3,8 _ _10 _11 _Baquelita _0,90 _9,9 _ _11 _9 _Hoja de papel de transferencia _1,00 _9 _ _12 _9 _Cloruro férrico _0,80 _7,2 _ _13 _10 _Fuentes de poder _6,00 _60 _ _14 _11 _Módulo de RF 433 MHz _10,00 _110 _ _15 _3 _Integrado ESP8266 _7,00 _21 _ _16 _9 _Cables USB _2,00 _18 _ _17 _8 _Cajas plásticas para proyectos 19x10x6 cm _5,00 _40 _ _18 _1 _Caja plástica para proyectos 20x20x6 cm _10,00 _10 _ _19 _10 _Portajacks tipo hembra _1,50 _15 _ _20 _30 _Postes metálicos con tornillos _1,00 _30 _ _ _ _ _ _514,37 _ _ b) Presupuesto para la parte de la implementación PRESUPUESTO PIC PROTOTIPO DE RED INALÁMBRICA PARA MONITOREO POR SMARTPHONE _ _ITEM _CANTIDAD _DESCRIPCION _V. UNITARIO _V.

TOTAL _ _1 _30 _TORNILLO COLEPATO _0,02 _0,6 _ _2 _30 _TACO FISHER #6 _0,01 _0,3 _ _3 _1 _BROCA CONCRETO 1/4 _1 _1 _ _4 _4 _SWITCH POWER VETO 110V SOBREPUESTO _1,5 _6 _ _5 _11 _CANALETA PLASTICA 20*12 _0,9 _9,9 _ _6 _70 _CABLE SOLIDO #14 _0,4 _28 _ _7 _1 _BREAKER 32 AMP _4,5 _4,5 _ _ _ _ _ _50,3 _ _ Tabla 6. Presupuesto del PIC Fuente: Elaborado por el autor Conclusiones Se estableció la red inalámbrica con módulos de RF cuya función es muestrear el estado de los estacionamientos y subir la información a través del módulo ESP8266 hasta el servidor IoT para que la información sea almacenada en su base de datos y a su vez sea accesible para su monitoreo mediante aplicaciones que operan con sistema operativo Android.

Se desarrolló la aplicación Android mediante MIT APP INVENTOR para la parte de

control de los estados, mediante los APIs de ThingSpeak a través de botones y pantallas que describen la función que realizan. También se agregó a la app una pantalla de visualización mas amigable para cualquier usuario donde se tiene las funciones básicas de monitoreo, reserva y registro de la base de datos.

Se implementó la tarjeta electrónica que constituye el hardware con el que el microcontrolador ATMEGA328 realiza el control de la etapa de comunicación por Radiofrecuencia y el control de la comunicación con el servidor IoT ThingSpeak. Se implementó la tarjeta que realiza el sensado del estado de los estacionamientos y la envía por Radio frecuencia hasta la tarjeta central que recopila la información para ser enviada al servidor IoT. Recomendaciones.

Se debe tener mucho cuidado en las conexiones auxiliares que se debe realizar para la configuración del módulo ESP8266 sobre todo en la alimentación de 3VDC de entrada para no quemar al mismo. Se debe procurar desarrollar la mayor parte del programa dentro de la sección void loop del IDE Arduino y tener la mínima cantidad de subrutinas fuera para optimizar el funcionamiento del programa.

Se debe mantener credenciales robustas de acceso al servidor y a la aplicación ya que al estar en la nube de Internet una desventaja es la vulnerabilidad del acceso si se tiene claves débiles. Se debe agregar antenas de longitud suficiente a los módulos RF para mitigar afectaciones en la red por pérdidas de enlace que para este caso fueron de 15 cm para cubrir un espacio físico de 20x15 mts Procurar que los estacionamientos seleccionados sean los de menor uso para poder facilitar los trabajos de implementación pero sobre todo para dar facilidades para el momento de realizar las pruebas y no causar retrasos innecesarios por este motivo.
Bibliografía: https://ricveal.com/blog/arduino-esp8266/
http://programarpicenc.com/articulos/radiofrecuencia-sistema-tx-rx-a-433mhz/
https://hipertextual.com/archivo/2014/10/internet-cosas/
https://bbvaopen4u.com/es/actualidad/apis-para-el-internet-de-las-cosas-thingspeak-pachube-y-fitbit
http://nothans.com/measure-wi-fi-signal-levels-with-the-esp8266-and-thingspeak
https://bbvaopen4u.com/es/actualidad/apis-para-el-internet-de-las-cosas-thingspeak-pachube-y-fitbit
https://www.minitronica.com/arduino-radiofrecuencia-433mhz-virtualwire/
https://www.taloselectronics.com/sensor-ultrasonico-hc-sr04/
https://hipertextual.com/archivo/2014/10/internet-cosas/

INTERNET SOURCES:

0% - http://repositorio.uisrael.edu.ec/bitstr
0% - https://www.scribd.com/document/36694530
0% - Empty
0% - https://www.scribd.com/document/36726489
0% - http://bibdigital.epn.edu.ec/browse?type
0% - http://coljuristas.org/documentos/libros
0% - https://www.scribd.com/document/29452544
0% - https://sarahelga.wordpress.com/oracione
0% - https://prezi.com/i6pt1dwggwgh/mi-educac
0% - https://www.scribd.com/document/33030871
0% - https://es.answers.yahoo.com/question/in
0% - https://www.bing.com/aclick?ld=d39SYuTyE
0% - https://docplayer.es/1172719-Aplicacion-
0% - https://www.scribd.com/document/21536996
0% - http://studylib.es/doc/4498864/dise%C3%B
0% - https://documentop.com/estudio-e-impleme
0% - https://www.scribd.com/document/37415847
0% - https://www.scribd.com/document/21314984
0% - https://www.scribd.com/document/24013066
0% - https://www.scribd.com/document/37948819
0% - https://es.scribd.com/doc/212541707/XBEE
0% - https://documentop.com/estudio-e-impleme
0% - https://www.scribd.com/document/33796535
0% - https://manualzz.com/doc/5378235/view-op
0% - https://es.scribd.com/document/358916671
0% - https://es.scribd.com/document/315008272
0% - https://es.scribd.com/document/298806734
0% - https://docplayer.es/65310367-Departamen
0% - http://docplayer.es/58192710-Ejercicios-
0% - http://bibdigital.epn.edu.ec/bitstream/1
0% - http://productforums.google.com/d/topic/
0% - http://geekswithblogs.net/gotchas/archiv
0% - https://es.scribd.com/document/282417103
0% - https://issuu.com/rodulfogonzalez/docs/t
0% - https://www.expoknews.com/el-futuro-son-
0% - http://gabinobarrerahabla.blogspot.com/
0% - https://en.wikipedia.org/wiki/Big_data
0% - https://www.psychologytoday.com/us/blog/
0% - https://doi.acm.org/10.1145/2594368.2594
0% - https://www.scribd.com/document/36084569

0% - https://engineering.tamu.edu/_files/_doc
0% - https://www.verizonwireless.com/support/
0% - http://pinegro.blogspot.com/2007/07/clas
0% - https://boe.vlex.es/vid/ley-5-2017-28-68
0% - https://issuu.com/la_hora/docs/quito5612
0% - https://www.es.paessler.com/monitoring_a
0% - https://www.ecologiaverde.com/como-afect
0% - http://elmandadito.blogspot.com/2012/12/
0% - http://www.coches-actu.com/actus,en-que-
0% - https://negociosycomunicaciones.wordpres
0% - https://www.scribd.com/document/34946061
0% - https://www.bing.com/aclick?ld=d3WV5qoD3
0% - https://www.bing.com/aclick?ld=d38P-R_Xw
0% - https://negociosyautos.wordpress.com/201
0% - http://monacoso17demarzoac.blogspot.com/
0% - https://es.wikihow.com/prevenir-accident
0% - https://issuu.com/comunicacionescolegioa
0% - http://docplayer.es/41789835-2013-volume
0% - https://www.bing.com/aclick?ld=d3zZRj0Ay
0% - https://elandroidelibre.elespanol.com/20
0% - https://www.bing.com/aclick?ld=d4A9e6903
0% - http://rgkit.blogspot.com/2014/09/estand
0% - https://issuu.com/innovar/docs/cat_innov
0% - https://www.scribd.com/document/36370915
0% - http://noticiasuruguayas.blogspot.com/20
0% - https://issuu.com/amuva/docs/memoria_pfc
0% - http://www.itierra.es/actividad-5-integr
0% - https://issuu.com/bibliotecapedagogica/d
0% - https://www.scribd.com/document/23556765
0% - http://www.investigacionsalud.gob.ec/ins
0% - https://www.scribd.com/document/28311533
0% - https://www.adslzone.net/postt276694.htm
0% - https://issuu.com/esc_arq_upr/docs/hiper
0% - https://issuu.com/jdelacruzpalacios/docs
0% - https://droidzon.com/lista-top-emulador-
0% - https://docplayer.es/29399460-Pontificia
0% - https://www.scribd.com/document/25679090
0% - https://www.scribd.com/document/23328920
0% - https://www.scribd.com/document/10230926
0% - http://soloelectronicos.com/2015/02/

0% - https://diarioelplanetablog.wordpress.co
0% - https://nocionporcomparirfreedownload.wo
0% - https://www.scribd.com/document/11520396
0% - https://es.scribd.com/doc/128282800/El-N
0% - https://issuu.com/atsmedellin/docs/manua
0% - https://es.scribd.com/doc/136459832/Libr
0% - http://www.revistajuridicaonline.com/wp-
0% - https://slidex.tips/download/conocimient
0% - https://noticias.caracoltv.com/acuerdo-f
0% - https://nemesisnoticias.wordpress.com/20
0% - https://www.wurth.es/seguridad-e-higiene
0% - https://www.bing.com/aclick?ld=d37hXli5r
0% - https://www.bing.com/aclick?ld=d3H0rh3Jw
0% - https://www.scribd.com/document/36206196
0% - http://docplayer.es/54763638-Universidad
0% - https://www.scribd.com/document/36658914
0% - http://a-electronics.com.mx/blog/page/2/
0% - https://www.scribd.com/document/37866935
0% - https://www.scribd.com/document/35215680
0% - http://programarpicenc.com/articulos/rad
0% - https://www.scribd.com/document/36574124
0% - http://mexico.newark.com/microchip/atmeg
0% - http://docplayer.es/15353321-Escuela-pol
0% - https://www.scribd.com/document/34454093
0% - https://es.scribd.com/document/345186747
0% - https://issuu.com/hernandezsanjuan/docs/
0% - http://bitacoraparaoscar.blogspot.com/p/
0% - https://www.scribd.com/document/25419263
0% - https://es.scribd.com/doc/179280520/trab
0% - https://www.bing.com/aclick?ld=d4ZTQppmh
0% - https://www.bing.com/aclick?ld=d3-2LuG4T
0% - http://nandoc92.blogspot.com/2013/03/mod
0% - http://pagina.jccm.es/contratacion/cgi-b
0% - https://mafiadoc.com/infraestructura-par
0% - http://jgcartru.blogspot.com/
0% - https://es.scribd.com/doc/55499717/Tesis
0% - http://docplayer.es/15268598-Una-aplicac
0% - https://www.bing.com/aclick?ld=d39Wk7moe
0% - http://docplayer.es/58192710-Ejercicios-
0% - http://soloelectronicos.com/2017/11/

0% - http://www.electrontools.com/Home/WP/201
0% - https://issuu.com/monolitic/docs/743rede
0% - http://www.mactronica.com.co/lm7805-1445
0% - https://www.arcaelectronica.com/integrad
0% - https://es.scribd.com/doc/279346422/Tesi
0% - https://www.e-ika.com/mini-fuente-de-ali
0% - https://es.scribd.com/doc/102719241/EDUC
0% - https://www.scribd.com/document/37730480
0% - http://oa.upm.es/cgi/exportview/subjects
0% - http://reyescarbajaladriel.blogspot.com/
0% - https://creaconlaura.blogspot.com/2013_0
0% - http://oa.upm.es/cgi/exportview/degree/G
0% - http://www.mailxmail.com/curso-desarroll
0% - http://oa.upm.es/cgi/search/simple/expor
0% - https://issuu.com/raymondcolle/docs/univ
0% - https://issuu.com/unigis_latina/docs/mer
0% - https://support.microsoft.com/es-mx/help
0% - https://issuu.com/indtec/docs/revista_sc
0% - https://www.scribd.com/document/26197553
0% - https://docplayer.es/32772298-Revista-de
0% - https://www.scribd.com/document/27525787
0% - http://www.academia.edu/8141706/Dise%C3%
0% - https://www.scribd.com/document/24404439
0% - https://issuu.com/luisbengochea/docs/com
0% - http://soloelectronicos.com/category/iot
0% - https://es.scribd.com/doc/29633032/Redes
0% - https://www.bing.com/aclick?ld=d3Vnggn87
0% - https://prezi.com/j788blbsldzk/la-moda-e
0% - https://winphonemetro.com/2013/10/dispon
0% - https://patentados.com/2015/sistema-y-me
0% - http://wwwaprendizesena.blogspot.com/201
0% - https://www.scribd.com/document/31716721
0% - http://docplayer.es/18258228-Pontificia-
0% - https://fr.scribd.com/doc/132681184/Sist
0% - https://manualzz.com/doc/5571034/jim%C3%
0% - https://es.slideshare.net/AlbertoSanchez
0% - https://pt.scribd.com/document/328085473
0% - http://informesdelaconstruccion.revistas
0% - https://www.bing.com/aclick?ld=d3Qlgw59y
0% - https://docplayer.es/58342400-Diseno-de-

0% - http://studylib.es/doc/1243904/cd-4890.p
0% - https://issuu.com/miguelperez/docs/_math
0% - https://es.scribd.com/doc/140408900/Tele
0% - http://arduino.cl/arduino-uno/
0% - http://electronica-pic.blogspot.com/2017
0% - http://docplayer.es/4108144-Robot-explor
0% - https://www.scribd.com/document/94960697
0% - https://issuu.com/atsmedellin/docs/manua
0% - https://es.wikihow.com/revisar-la-veloci
0% - https://frio-industrial.com/slides/slide
0% - https://www.scribd.com/document/18208450
0% - https://www.scribd.com/document/15176115
0% - https://es.scribd.com/document/339554020
0% - https://es.scribd.com/doc/33231196/Siste
0% - https://www.scribd.com/document/15122877
0% - https://aranibarvictorblog.wordpress.com
0% - https://www.scribd.com/document/36658914
0% - https://www.scribd.com/document/25067721
0% - http://studylib.es/doc/8759352/m---tesis
0% - https://documentop.com/estudio-e-impleme
0% - https://www.scribd.com/document/21539911
0% - http://docplayer.es/14520517-Universidad
0% - https://es.scribd.com/doc/237731860/SIST
0% - https://www.scribd.com/document/29351461
0% - https://docplayer.es/28120692-Tarjeta-de
0% - https://www.grainger.com.mx/producto/LIT
0% - http://studylib.net/doc/18731881/lm1117-
0% - https://en.wikipedia.org/wiki/2-2-2
0% - https://docplayer.es/8966605-Emisor-y-re
0% - http://venezueladx.blogspot.com/2011/
0% - https://es.scribd.com/document/337842215
0% - https://keytweak.softonic.com/
0% - https://vdocuments.site/documents/dcs-93
0% - https://issuu.com/bibliotecafredman/docs
0% - https://www.scribd.com/doc/36231191/Circ
0% - http://app-tek.net/producto/modulo-wifi-
0% - https://aprendiendoarduino.wordpress.com
0% - https://soloelectronicos.com/category/ar
0% - http://www.lan.com/upload/pdf/seccion-2-
0% - https://issuu.com/esvial/docs/libroatica

0% - https://es.scribd.com/doc/110359506/Tesi
0% - https://www.scribd.com/document/33993595
0% - https://issuu.com/teletronix/docs/teletr
0% - https://www.scribd.com/document/36876976
0% - http://librearduino.blogspot.com/feeds/p
0% - http://sistemas-clases.forosactivos.net/
0% - https://www.scribd.com/document/36475167
0% - http://soloelectronicos.com/page/64/
0% - https://www.bing.com/aclick?ld=d4YFe9f8W
0% - https://issuu.com/cvillenat/docs/pc30
0% - http://studylib.es/doc/1277448/cd-5884.p
0% - https://www.slideshare.net/JuanDavidNezL
0% - https://www.scribd.com/document/29078253
0% - http://docplayer.es/28120692-Tarjeta-de-
0% - https://www.scribd.com/document/26197553
0% - https://us.diablo3.com/en/blog/20057106/
0% - https://issuu.com/merce9/docs/el-futuro-
0% - http://sysarmy.com/planet/
0% - http://soloelectronicos.com/tag/datos-de
0% - https://us.diablo3.com/en/blog/20149714/
0% - https://www.scribd.com/document/36641213
0% - https://www.1and1.es/digitalguide/servid
0% - https://es.wikipedia.org/wiki/Bitcoin
0% - http://repositorio.utn.edu.ec/bitstream/
0% - https://es.scribd.com/document/357318328
0% - https://edoc.site/manual-de-gerencia-de-
0% - https://issuu.com/la_hora/docs/quito19ag
0% - https://issuu.com/eltiempovenezuela/docs
0% - https://es.scribd.com/doc/267581001/sist
0% - http://www.coacordoba.net/documentos/Cur
0% - http://www.buenastareas.com/materias/con
0% - https://www.scribd.com/document/21648460
0% - https://www.scribd.com/document/15326773
0% - https://elenadelarosa.wordpress.com/trab
0% - https://zonaarcadeblog.wordpress.com/cat
0% - https://www.scribd.com/document/20296720
0% - https://docplayer.es/81661640-Diseno-de-
0% - https://issuu.com/luisbengochea/docs/com
0% - http://www.icicm.com/files/Redes_Inalamb
0% - https://www.scribd.com/document/25452641

0% - https://es.scribd.com/document/283374191
0% - https://www.scribd.com/document/35441172
0% - https://www.scribd.com/document/37189216
0% - https://www.bing.com/aclick?ld=d3jNT6WV5
0% - http://docplayer.es/53713261-T-e-s-i-s-i
0% - https://www.bing.com/aclick?ld=d4iwzURGP
0% - https://www.scribd.com/document/35663472
0% - https://arduinoshields.wordpress.com/cat
0% - http://descubriendofenomenoselectricos.b
0% - https://github.com/AndriyHz/Teensy-3.2-a
0% - https://elcajondearduino.wordpress.com/a
0% - https://docplayer.es/11222576-Sensor-de-
0% - https://www.dspace.espol.edu.ec/bitstrea
0% - https://docplayer.es/14387476-Educabot-v
0% - https://www.scribd.com/document/34701755
0% - https://edoc.site/fundamentos-generales-
0% - http://docplayer.es/26491468-Escuela-pol
0% - https://www.scribd.com/document/72615103
0% - https://docplayer.es/68358695-Controlado
0% - https://github.com/jersagfast/RGBW-31/bl
0% - https://elcajondearduino.wordpress.com/c
0% - http://arduino123.blogspot.com/p/sonido.
0% - https://mundosmr.wordpress.com/2015/05/2
0% - http://www.electrontools.com/Home/WP/201
0% - https://www.scribd.com/document/32493858
0% - https://www.spainlabs.com/foros/tema-Pro
0% - https://pastebin.com/9B9RcGkN
0% - https://robologs.net/2015/02/10/tutorial
0% - https://robologs.net/2015/02/10/tutorial
0% - http://soloelectronicos.com/2012/12/02/
0% - https://www.scribd.com/document/37415847
0% - https://files.support.epson.com/pdf/tx56
0% - https://raulrprdesing.wordpress.com/auth
0% - https://www.calameo.com/books/0045818489
0% - https://arduino.stackexchange.com/questi
0% - https://elcajondearduino.wordpress.com/a
0% - https://docplayer.es/11222576-Sensor-de-
0% - https://forum.arduino.cc/index.php?topic
0% - https://github.com/techiesms/Monitoring-
0% - https://www.scribd.com/document/34725843

0% - http://www.electrontools.com/Home/WP/201
0% - https://www.minitronica.com/arduino-radi
0% - http://soloelectronicos.com/2012/12/page
0% - https://www.scribd.com/document/28240683
0% - https://edoc.site/pc-actual-260-pdf-free
0% - http://www.bing.com/videos/search?q=incl
0% - https://issuu.com/universidadnacionaldel
0% - http://docplayer.es/15004013-Efecto-del-
0% - https://juandomingofarnos.wordpress.com/
0% - https://www.upv.es/materiales/Fcm/Fcm03/
0% - https://vdocuments.mx/documents/micom-c2
0% - http://docplayer.es/1963314-La-version-d
0% - https://www.scribd.com/document/25909729
0% - https://es.scribd.com/document/342278877
0% - https://www.scribd.com/document/25932289
0% - https://docplayer.es/7982036-Editores-jo
0% - http://www.coches-actu.com/actus,diagram
0% - https://es.scribd.com/document/342278877
0% - https://losasociados.wordpress.com/2016/
0% - https://ardubasic.wordpress.com/tag/medi
0% - https://www.scribd.com/document/13085960
0% - http://repositorio.ug.edu.ec/bitstream/r
0% - https://issuu.com/separ/docs/el_libro_bl
0% - https://www.scribd.com/document/16930686
0% - https://es.scribd.com/doc/200248116/Libr
0% - https://es.scribd.com/doc/304487757/Clas
0% - https://www.scribd.com/document/13883510
0% - https://es.scribd.com/document/291851451
0% - http://docplayer.es/7819633-Universidad-
0% - https://www.scribd.com/document/34523501
0% - https://www.scribd.com/document/36694530
0% - https://www.scribd.com/document/30934645
0% - https://issuu.com/ucacue/docs/revista
0% - https://es.scribd.com/doc/64902948/bogot
0% - https://docplayer.es/24185450-Universida
0% - https://pt.scribd.com/doc/120683223/tesi
0% - https://www.scribd.com/document/27179024
0% - https://edoc.site/nec-nfpa-70-2008-spani
0% - https://pt.scribd.com/doc/101031043/Text
0% - https://www.scribd.com/document/26197553

0% - https://es.scribd.com/doc/236280623/Fisi
0% - https://es.scribd.com/doc/307548201/Memo
0% - https://www.scribd.com/document/30056191
0% - https://issuu.com/legissa/docs/rev_logis
0% - https://docplayer.es/80376377-Control-re
0% - https://docplayer.es/61090057-Actas-del-
0% - https://www.bing.com/aclick?ld=d3I7DNvta
0% - https://www.scribd.com/document/27671564
0% - https://administracionderequerimientos.w
0% - https://sites.google.com/site/conferenci
0% - http://www.un.org/es/databases/index.htm
0% - https://www.scribd.com/document/36973330
0% - https://polaridad.es/http-post-almacenar
0% - https://www.scribd.com/document/28311533
0% - https://www.scribd.com/document/32802194
0% - http://dietasparaadelgazarte.com/6-ejerc
0% - https://www.scribd.com/document/33354387
0% - http://rdsoporteymantenimientodepc.blogs
0% - http://tecnoemergentes.foroactivo.com/t8
0% - https://www.policialarevista.com/Nota/N7