FACULTAD DE INGENIERÍA

TRABAJO DE TITULACION PREVIO A LA OBTENCIÓN DEL TÍTULO DE MÁSTER EN TECNOLOGÍAS DE LA INFORMACIÓN MENCIÓN EN REDES DE COMUNICACIONES

Tema: Propuesta de una solución IoT (Internet of things) piloto, para mejorar la movilidad y servicio en las líneas de autobuses del transporte público en la ciudad de Quito.

AUTOR: JESSICA GABRIELA MEJÍA PLACENCIA

QUITO, 2020 – AGOSTO

ÍNDICE DE CONTENIDOS

Planteamiento del Problema de Investigación 1

INTRODUCCIÓN 2

ANTECEDENTES 2

JUSTIFICACIÓN 3

OBJETIVOS 4

Objetivo General 4

Objetivos Específicos 4

CAPÍTULO 1 5

1. Marco Teórico 5

1.1. Definición IoT (Internet of things) 5

1.2. Elementos de un Sistema IoT 6

1.2.1. Hardware (actuadores y sensores): 6

1.2.2. Plataforma de Middleware: 6

1.2.3. Herramientas: 6

1.3. Facultades de un sistema IoT 6

1.4. LPWAN (Redes de área amplia y de baja potencia) 7

1.4.1. Tecnologías LPWAN 8

1.4.2. Tecnología LoRa 8

1.4.2.1. LoRaWAN 9

1.4.2.1.1. Arquitectura LoRaWAN 9

1.4.2.1.2. Tipos de módulos LoRaWAN 10

1.4.2.1.3. Seguridad LoRaWAN 11

1.4.2.1.4. Gateway LoRaWAN 12

1.5. Arduino 12

1.5.1. Modelos de Tarjetas de Arduino 12

1.5.1.1. Arduino UNO 13

1.5.1.2. Arduino MEGA/2560 13

1.6. Sistemas de Transporte Inteligente 14

1.6.1. Características de los Sistemas de transporte inteligente 15

1.6.2. Estandarización para Sistemas de Transporte Inteligente 16

1.7. Geolocalización 16

1.8. Plataformas en la nube para IoT 16

1.8.1. Kaa IoT 16

1.8.2. OpenIoT 17

1.8.3. ThingSpeak 18

1.8.4 M2MLight 18

CAPÍTULO 2 20

2. Análisis de la situación actual del transporte en la ciudad de Quito y propuesta de despliegue de tecnología LoRa. 20

2.1. Organización 20

2.1.1. Sistema Metrobus – Q 20

2.1.2. Sistema Convencional 22

2.2. Datos actualizados 24

2.3. Deficiencias en el Sistema de Transporte de la Ciudad de Quito 25

2.3.1. Desarticulación y solapamiento de la red 25

2.3.2. Desorganización, Ineficiencia y Pérdida de Tiempo 25

2.3.3. Congestión y Saturación de la Infraestructura vial 26

2.4. Perspectivas de diseño de un Sistema de Transporte Inteligente 26

2.4.1. Perspectiva de servicio 26

2.4.2. Perspectiva Operacional 27

2.4.3. Perspectiva Tecnológica 27

2.4.4. Perspectiva Institucional 28

2.4.5. Perspectiva Económica 29

2.5. Perspectivas de mejora utilizando tecnología LoRa 30

2.5.1. Propuesta de despliegue de tecnología LoRa 30

CAPÍTULO 3 33

3. Diseño de Piloto IoT 33

3.1 Diseño de comunicación Lora 33

3.2 Diseño de nodo autobús 35

3.2.1 Dispositivos para diseño nodo autobús 35

3.2.1.1. LoRa GPS Shield v95+Arduino UNO 35

3.2.1.2. Especificaciones técnicas Lora GPS Shield v95 35

3.2.1.3. Especificaciones técnicas de GPS 36

3.2.2 Conexiones entre dispositivos para nodo autobús 37

3.3 Diseño de nodo Gateway 40

3.3.1 Dispositivos para diseño nodo Gateway 40

3.3.1.1. Especificaciones técnicas Gateway LoRa LG01-P 40

3.4 Diseño de nodo Parada 43

3.4.1 Dispositivos para diseño nodo parada 43

3.4.1.1. LoRa Shield v95+ Arduino Mega UNO 43

3.4.1.2. Especificaciones Técnicas LoRa Shield v95: 43

3.4.1.3. LCD 1602 44

3.4.1.4. Controlador I2C para LCD 16X2 45

3.4.1.3.1. Especificaciones técnicas controlador I2C 45

3.5 Diseño de Nodo Servidor 46

3.6 Diagrama de despliegue de Piloto 48

3.7 Costos de Implementación del prototipo 49

CAPÍTULO 4 51

4. Implementación de Piloto 51

4.1. Implementación Comunicación LoRa entre nodos 52

4.2. Implementación nodo Autobús 54

4.3. Implementación nodo Gateway 54

4.3.1. Protocolo MQTT 57

4.3.1.1. Funcionamiento MQTT 58

4.3.1.2. Estructura de un mensaje MQTT 60

4.3.1.3. Estructura de red para reenvío de mensajes MQTT 61

4.3.1.4. Configuración conexión MQTT 61

4.3.1.5. Definición de Canales MQTT 63

4.3.2. Configuración de Sistema Linux de Gateway 63

4.3.3. Envió de mensajes MQTT de Gateway al Servidor M2Mlight 66

4.3.4. Recepción de mensajes de Servidor M2Mlight a Gateway 67

4.4. Implementación de procesos y mensajes en Servidor M2Mlight 67

4.4.1. Menú de sensores 70

4.5. Implementación nodo Parada 70

4.6. Pruebas de implementación 72

4.7. Análisis de Resultados y Evaluación Final 81

CONCLUSIONES 86

RECOMENDACIONES 87

BIBLIOGRAFÍA 88

ÍNDICE DE TABLAS

Tabla 1. Factor de dispersión 9

Tabla 2. Características de Sistemas de Transporte Inteligente 15

Tabla 3. Configuraciones para dispositivos radio Lora 33

Tabla 4. Distribución de Pines LoRa GPS Shield v95+Arduino 37

Tabla 5. Conexiones entre Arduino y display LCD1602 44

Tabla 6. Detalle de costos Dragino LoRa IoT Kit de desarrollo V2 915MHZ 49

Tabla 7. Detalle de costos Dragino Lora GPS Shield 915Mhz 49

Tabla 8 Detalle de costos de Arduino UNO y LCD 1602+I2C. 50

Tabla 9. Detalle de Gastos Totales de la Implementación 50

ÍNDICE DE FIGURAS

Figura 1. Internet of Things 6

Figura 2. Capacidades de un sistema IoT 7

Figura 3. Arquitectura de red LoRa WAN..... 10

Figura 4. Tipos de dispositivos LoRaWAN..... 11

Figura 5. Características de Arduino UNO..... 13

Figura 6. Características Arduino MEGA 14

Figura 7. Sistemas de transporte inteligente 14

Figura 8. Características de Kaa IoT..... 17

Figura 9. Diagrama de funcionamiento de OpenIoT 17

Figura 10. Estructura básica de ThingSpeak..... 18

Figura 11. Servidor M2MLight..... 19

Figura 12. Sistema Metrobus-Q..... 21

Figura 13. Esquema de Sistema de Buses Convencionales de la ciudad de Quitos 23

Figura 14. Propuesta de despliegue piloto con tecnología LoRa..... 31

Figura 15. Lora Shield para Arduino..... 36

Figura 16. Conexión de GPS y Arduino UNO 38

Figura 17. Arquitectura Gateway LG01-P..... 41

Figura 18. Controlador I2C..... 45

Figura 19. Comunicación LoRa y configuración de parámetros 53

Figura 20. Dispositivo Lora GPS Shield ubicado en vehículo para simulación de piloto..... 54

Figura 21. Ingreso a Gateway Lora..... 55

Figura 22. Ingreso de parámetros de configuración en Gateway..... 56

Figura 23. Elección de servidor 56

Figura 24. Elección de modo de conexión 57

Figura 25. Funcionamiento Protocolo MQTT 58

Figura 26. Inicio de conexión TCP/IP con Bróker 59

Figura 27. Envío de mensajes MQTT 59

Figura 28. Subscripción para envío de mensajes MQTT 60

Figura 29´. Estructura de un mensaje MQTT 60

Figura 30. Topología para conexión MQTT 61

Figura 31. Configuración MQTT Server 62

Figura 32. Definición de canales en Gateway 63

Figura 33. Inicio de sesión con Putty SSH puerto 22 64

Figura 34 Configuración en parte Linux de Gateway por SSH 65

Figura 35 .MQTT Explorer 66

Figura 366. Procesos y mensajes de piloto 68

Figura 37 Registro de usuario en Plataforma M2MLight 69

Figura 38 Menú de sensores en plataforma M2Mlight 70

Figura 39 Implementación nodo parada 71

Figura 40 Panel Map de servidor M2Mlight 72

Figura 41 Puntos de prueba para piloto 73

Figura 42 Escenario de nodos parada en plataforma M2MLight 74

Figura 43 Instalación de LoRa GPS Shield en automóvil 74

Figura 44 Transcurso de autobús de Parada 0 a Parada 1, 24Km/h 75

Figura 45 Tiempo de arribo de parada 0 a parada 1 76

Figura 46 Arribo próximo de autobús a parada 1, 5,22 Km/h 76

Figura 47. Autobús paso de parada 1 y continua a parada 2..... 77

Figura 48. Autobús llegará próximamente a Parada Final 2, 2,2Km/h..... 77

Figura 49. Autobús llegará a la parada 2 en un tiempo de 33s 78

Figura 50. Autobús pasa de parada 2 78

Figura 51. Autobús finaliza la ruta fijada para el piloto 79

Figura 52. Recibe información de posición de dispositivo LoRa GPS Shield 79

Figura 53. Muestra mensajes del tiempo de arribo a las paradas..... 80

Figura 54. Muestra mensajes de Paso de autobús de las paradas 80

Figura 55. Enlace de Radio de Nodo Gateway hacia Parada inicial..... 83

Figura 56. Enlace de Radio de Nodo Gateway hacia Parada Final 83

Figura 57. Niveles de cobertura alcanzada por tecnología LoRa, evaluado en 20km de alcance máximo 84

ÍNDICE DE DIAGRAMAS

Diagrama 1. Diseño comunicación Lora, basado en (Palacios Ibarra & Zúñiga Pérez, 2019)..... 34

Diagrama 2. Diseño Nodo Autobús 39

Diagrama 3. Diseño Nodo Gateway 42

Diagrama 4. Diseño Nodo Parada..... 46

Diagrama 5. Diseño Nodo Servidor..... 47

Diagrama 6. Despliegue Piloto 48

Planteamiento del Problema de Investigación

¿Es posible mejorar la movilidad y el servicio de las líneas de autobuses de transporte público en la ciudad de Quito?

¿Es posible reducir los tiempos de espera de los usuarios de transporte en la ciudad de Quito, ofreciendo información a tiempo real sobre la ubicación y velocidad de las unidades?

¿Aplicando soluciones IoT es posible mejorar el problema de calidad del servicio al usuario en la ciudad de Quito?

¿Es posible evaluar el sistema de unidades de transporte público, empleando tecnología LoRa?

INTRODUCCIÓN

ANTECEDENTES

La situación de movilidad en el transporte público de autobuses en la ciudad de Quito presenta grandes inconvenientes tales como: falta de cultura vial, afectación al medio ambiente y congestión. Además, no dispone de un diseño tecnológico que permita un mejor control de movilidad de las unidades y servicio a los habitantes.

Las líneas de autobuses de transporte público de la ciudad de Quito no cuentan con infraestructura necesaria que permita brindar información sobre el tiempo real de su ubicación y disponibilidad, lo que ocasiona que los tiempos de espera de los usuarios sean extensos y desconocidos.

Uno de los problemas más grandes que presenta la ciudad de Quito, es el congestionamiento vehicular, ya que cada vez son más frecuentes los embotellamientos de tráfico en ciertas avenidas y calles, por lo que los ciudadanos tardan horas en llegar a sus destinos.

En el servicio de autobuses de transporte público, no es posible realizar evaluaciones de tipo tecnológico, debido a que la ciudad de Quito se encuentra actualmente en proceso de desarrollo al transporte inteligente.

JUSTIFICACIÓN

Gracias al gran avance de la tecnología, integrando una solución IoT, es posible mejorar la calidad de vida de los usuarios que utilizan el transporte público en la ciudad de Quito, incrementando el desarrollo tecnológico y logístico de la capital del Ecuador.

A través de la aplicación de una tecnología IoT, se permitirá al usuario conocer la ubicación y tiempo de arribo de las unidades; optimizando de esta manera su tiempo al momento de desplazarse de un lugar a otro.

Utilizando la tecnología LoRa se realizará una propuesta piloto de diseño que aporte con el desarrollo tecnológico y mejore el servicio de transporte en la ciudad de Quito.

Con el uso de la tecnología LoRa, se evaluará el aporte significativo que tienen las soluciones tecnológicas hoy en día para mejorar los servicios y la calidad de vida de los usuarios.

OBJETIVOS

Objetivo General

Proponer una solución IoT con tecnología LoRa para mejorar la movilidad y servicio de las líneas de autobuses de transporte público en la ciudad de Quito para el 2020.

Objetivos Específicos

Aplicar tecnología IoT para mejorar la calidad y servicio de transporte en la ciudad de Quito.

Crear una solución con tecnología LoRa, que permita conocer en tiempo real la espera del usuario en las paradas, además de obtener la geolocalización y velocidad de las unidades.

Identificar los dispositivos LoRa idóneos, que permitan solucionar la propuesta de diseño y así mejorar la calidad del servicio de transporte público de la ciudad de Quito.

Evaluar si la solución de movilidad inteligente con tecnología LoRa es una alternativa de mejora para el servicio en las líneas de autobuses del transporte público en la ciudad de Quito.

CAPÍTULO 1

1. Marco Teórico

Este capítulo muestra los conceptos de las herramientas tecnológicas que serán utilizadas para el desarrollo del piloto propuesto.

1.1. Definición IoT (Internet of things)

El concepto Internet de las cosas, se refiere a un tipo de tecnología fundamentada en la "conexión de objetos cotidianos a Internet que intercambian, agregan y procesan información sobre su entorno físico para proporcionar servicios de valor añadido a los usuarios finales"(Barrio Andrés, 2018) Figura 1.

El término Internet de las Cosas fue establecido por Kevin Ashton, en una presentación realizada en el año 1.999 para la multinacional Procter & Gamble, "donde representaba un sistema en el cual los objetos en el mundo físico podrían conectarse a Internet a través de sensores para automatizar la recogida de datos, propugnando su aplicación en la cadena de suministro, añadiéndoles etiquetas RFID"(Barrio Andrés, 2018).

Por tanto, el objetivo principal de IoT es conectar los objetos que no tienen conexión a una red de Internet, creando la posibilidad de interactuar con personas y varios objetos.

IoT, hoy en día es considerado como la "Tercera Ola de Internet", ya que ha logrado generar un gran impacto económico en todas las industrias a nivel mundial. Debido a su gran crecimiento exponencial, ha conseguido brindar "beneficios económicos, mejoras de procesos y mantenimientos preventivo"(Pisano, 2018).

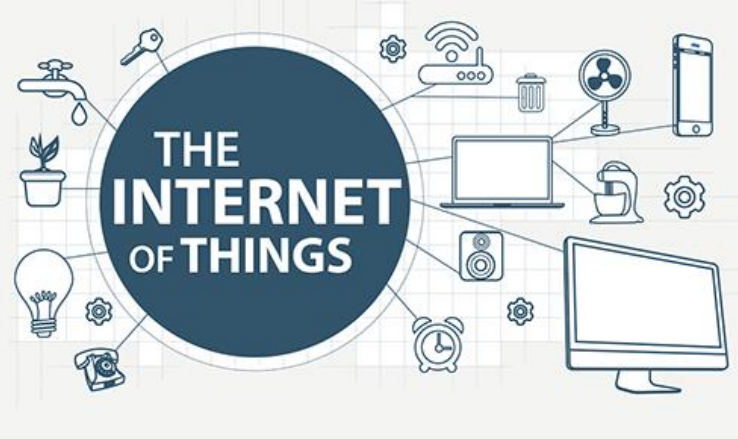

Figura 1. Internet of Things

Tomado de (Casanova, 2017)

1.2. Elementos de un Sistema IoT

1.2.1. Hardware (actuadores y sensores): son dispositivos controladores encargados de recopilar los datos y transmitirlos a su entorno, siendo esta información siempre en tiempo real.

1.2.2. Plataforma de Middleware: es responsable del intercambio y análisis de información entre las aplicaciones.

1.2.3. Herramientas: permiten la visualización e interpretación de la información, por lo que deben ser diseñadas para ser accesadas por diferentes aplicaciones y dispositivos. (Tavizon et al., 2016)

1.3. Facultades de un sistema IoT

Las facultades que tiene un sistema IoT están comprendidas dentro de los elementos que se presentan a continuación en la Figura 2:

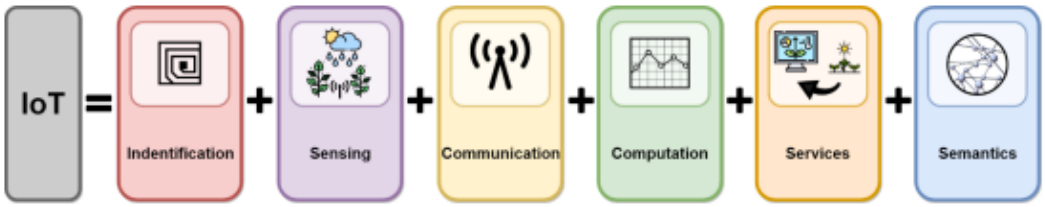

Figura 2. Capacidades de un sistema IoT

Tomado de (Ortiz Monet, 2019)

- **Identificación:** ofrece un ID único a cada elemento de una red.
- **Captación:** recopila la información de los objetos que se encuentran dentro de la red.
- **Comunicación:** su objetivo es la conexión inalámbrica entre los nodos que conforman el sistema.(Ortiz Monet, 2019)
- **Computación:** hace mención al procesamiento de los datos, el cual puede ser ejecutado desde diferentes tipos de plataformas.(Ortiz Monet, 2019)
- **Servicios:** se procesan los datos ofreciendo varios servicios a los interesados.
- **Semántica:** brinda los servicios de la aplicación al usuario final, gracias a los datos que han sido procesados y analizados.(Ortiz Monet, 2019)

1.4. LPWAN (Redes de área amplia y de baja potencia)

LPWAN es un protocolo de transporte inalámbrico que se caracteriza por operar en las bandas de Sub-GHz sin licencia y cumple requerimientos para trabajar en entornos de red M2M (máquina a máquina) y de IoT, esta tecnología tiene la capacidad de permitir comunicaciones de largo alcance de 0.3 kb/s a 50 Kb/s a bajo consumo de energía y costo a una gran cantidad de terminales.

Cabe mencionar que LPWAN utiliza banda ISM en Europa, siendo un espectro de frecuencias sin licencia que trabaja en el rango de 867 y 869 MHz, mientras que en Estados Unidos se usa la

franja entre 902 y 928 MHz, por lo que depende de la ubicación donde vaya a ser utilizada. (Dopazo González, 2019)

1.4.1. Tecnologías LPWAN

En la actualidad existen 5 tipos de plataformas disponibles dentro de las redes LPWAN: LoRa, Sigfox, NB-IoT, Weightless y RPMA; pero, para el desarrollo de este trabajo de titulación, solo se hará mención a la tecnología utilizada.

1.4.2. Tecnología LoRa

LoRa es la modulación inalámbrica utilizada para enlaces de largo alcance, trabaja con parámetros de configuración como son: canal, spreading factor (SF), coding rate (CR) y bandwidth, que permiten ajustar parámetros en la tasa de transmisión, corrección de errores y rango de transmisión. “Lora utiliza un tipo de modulación en radiofrecuencia patentado por Semtech llamado Chirp Spread Pectrum” (Cárdenas et al., 2018) que trabaja en las bandas de 433 MHz, 868 MHz (16 canales) y 915 MHz (72 canales) según el país donde se realice la aplicación.

- **Canal:** define la frecuencia central de trabajo.
- **Spreading Factor(SF):** el factor de dispersión, define el número de bits para codificar un símbolo. “Sus valores típicos se encuentran entre 7 y 12, cuanto más alto sea su valor, más robusto será el enlace de comunicación, y entre más bajo sea su valor aumenta la tasa de transmisión” (Ballesta Viñas, 2018).Tabla 1.

Tabla 1. Factor de dispersión

SF	Tasa de bits equivalente (kb/s)	Sensibilidad (dBm)
12	0.293	-137
11	0.537	-134.5
10	0.976	-132
9	1.757	-129
8	3.125	-126
7	5.468	-123

Tomado de (Moya Quimbita, 2018)

- **Coding Rate (CR):** esta codificación puede obtener valores de 4/5, 4/6, 4/7 y 4/8 para optimizar la localización y corrección de errores cíclica.
- **Bandwidth:** especifica el ancho de frecuencia que vamos a emplear.

1.4.2.1. LoRaWAN

El protocolo MAC llamado LoRaWAN fue desarrollado por Lora Alliance, integrada por empresas multinacionales de telecomunicaciones como: Cisco, IBM y Orange. "LoRaWAN define el protocolo de comunicación y la arquitectura del sistema para la red, mientras que la capa física LoRa habilita el enlace de comunicación de largo alcance"(Ballesta Viñas, 2018).

1.4.2.1.1. Arquitectura LoRaWAN

La arquitectura de red LoRaWAN muestra habitualmente una topología tipo estrella, que está comprendida por las llamadas "pasarelas (Gateway), nodos finales (LoRa thing), servidor de red, servidor de aplicación" (Moya Quimbita, 2018).

En la red LoRaWAN los nodos no se relacionan a un Gateway determinado, sino los datos transmitidos por un nodo son admitidos por múltiples Gateway. El trabajo del servidor de red, como se indica en la Figura 3, es “negociar la red, filtrar los paquetes redundantes recibidos, ejecutar pruebas de seguridad, demostrar confirmaciones a través del Gateway óptimo y efectuar adaptación a la velocidad de datos”(Moya Quimbita, 2018)

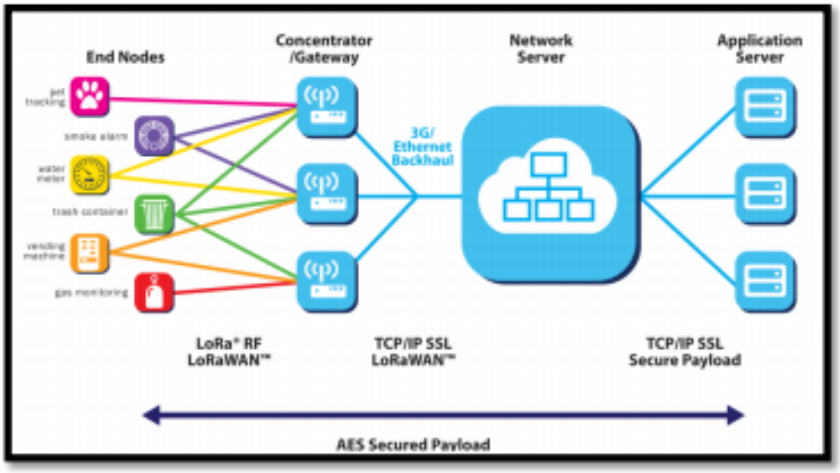

Figura 3. Arquitectura de red LoRa WAN

Tomado de (Moya Quimbita, 2018)

1.4.2.1.2. Tipos de módulos LoRaWAN

LoRaWAN maneja diferentes tipos de módulos que intercambian la latencia de comunicación del enlace descendente de la red frente a la duración de la batería, tratando de perfeccionar los perfiles de aplicaciones finales. Como se puede apreciar en la Figura 4.

- **Módulos de Clase A:** Admite comunicaciones bidireccionales y “ofrece el menor consumo de energía, pero solo logra recibir un enlace descendente después de enviar un mensaje de enlace ascendente hacia el Gateway” (Casanova González, 2020).

- **Módulos de Clase B:** “admite más espacios de mensajes de enlace descendente que para los de clase A, logrando disminuir la latencia de los mensajes, pero al mismo tiempo hace que sea menos eficiente en el uso de la energía” (Casanova González, 2020).
- **Módulos de Clase C:** “tiene ventanas de recepción continua que solo se cierran cuando el dispositivo está enviando un mensaje de enlace ascendente”(Casanova González, 2020), por tanto este tipo de clase ofrece el menor ahorro de energía y es recomendable usarlo en dispositivos que cuenten con una fuente de alimentación externa.

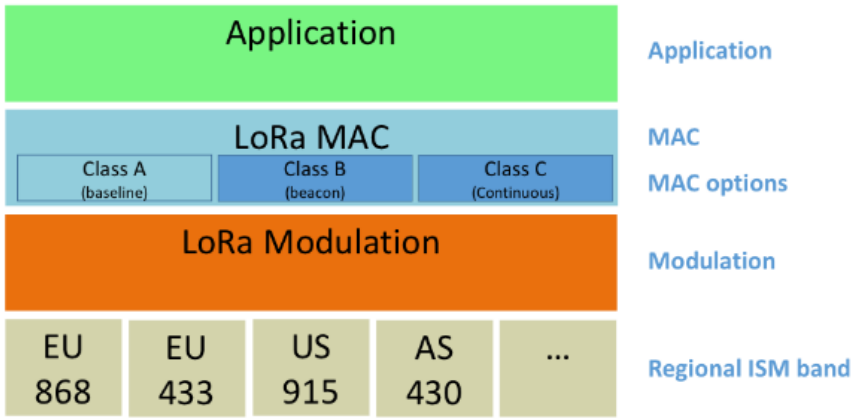

Figura 4. Tipos de dispositivos LoRaWAN

Tomado de (Sabas, 2017)

1.4.2.1.3. Seguridad LoRaWAN

LoRaWAN hace uso del algoritmo cifrado AES-128 para resguardar las comunicaciones de datos.

- **Network Session Key:** Clave de 128 bits que garantiza seguridad a nivel de red.(Casanova González, 2020)
- **Application Session Key:** Clave de 128 bits que garantiza seguridad extremo-extremo.(Casanova González, 2020)

- **Application Key:** clave de 128 bits que se utiliza para despliegues OTAA.(Casanova González, 2020)

1.4.2.1.4. Gateway LoRaWAN

"Es un dispositivo informático, que permite la interconexión de redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación. Su propósito es traducir la información del protocolo utilizado en una red al protocolo usado en la red de destino" (Torres Rodríguez & Patiño López, 2019).

1.5. Arduino

Es una multi-plataforma electrónica de código abierto que reduce el proceso de trabajar con microcontroladores, basada en hardware y software libre, accesible y fácil de utilizar en cualquier tipo de aplicación. Puede ser programada en Windows, MacOS y GNU/Linux. La placa opera con 5V o .3.3V dependiendo del modelo.

El hardware de Arduino está comprendido por un microcontrolador AVR del fabricante ATMEL, capaz de "ejecutar operaciones en sincronismo con una señal de reloj, a la velocidad de 8 a 32 MHz, dependiendo del modelo, que le provee un cristal de cuarzo"(Céspedes Machicao, 2017), su ambiente de programación es de tipo open-source desarrollado en Processing y Java.

1.5.1. Modelos de Tarjetas de Arduino

Los diferentes modelos de Arduino han sido desarrollados gracias a la gran demanda de usuarios e investigaciones tecnológicas. Dos de los modelos más utilizados en las aplicaciones se describen a continuación.

1.5.1.1. Arduino UNO

La placa Arduino Uno, se basa en un microcontrolador ATmega328 que cumple funciones de programación y comunicación a la vez. Está compuesto por 14 pines digitales de E/S, de los cuales 6 son salidas PWM, 6 pines de entradas analógicas, memoria flash de 32KB, SRAM de 2KB y EEPROM de 1KB. Trabaja en un voltaje de operación de 5v, como se aprecia en la Figura 5.

Figura 5. Características de Arduino UNO

***Tomado de* (Guerrero, 2014)**

1.5.1.2. Arduino MEGA/2560

Se trata de un tablero electrónico más robusto y potente, basado en un microcontrolador Atmega 2560. Está integrado por 54 pines digitales de E/S, de los cuales 15 son salidas PWM, 16 pines de entradas analógicas, memoria flash de 256KB, SRAM de 8KB y EEPROM de 4KB. Trabaja en un voltaje de operación de 7 a 12V. Tal como se observa en la Figura 6.

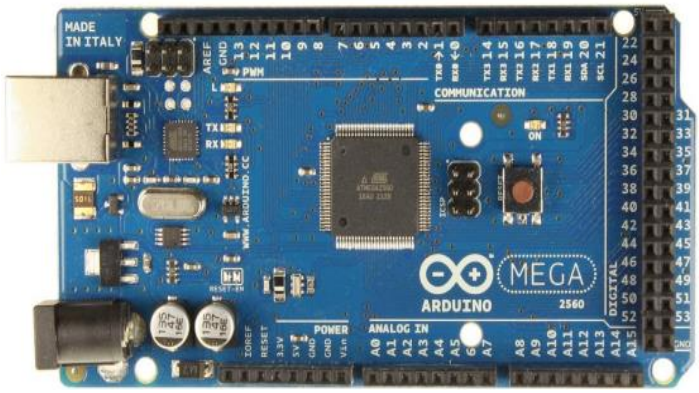

Figura 6. Características Arduino MEGA

Tomado de (Tapia Ayala & Manzano Yupa, 2013)

1.6. Sistemas de Transporte Inteligente

Los sistemas de transporte inteligente (SIT) inician en los años 90 como una opción sostenible al problema ocasionado por la creciente demanda de movilidad. Están constituidos por tecnologías informáticas y de telecomunicaciones, diseñadas para reunir la información, procesarla, analizarla y usarla para mejorar y garantizar la estabilidad de las personas, la movilidad, el medio ambiente, el costo-beneficio, la seguridad vial, la productividad y la economía. Como se visualiza en la Figura 7.

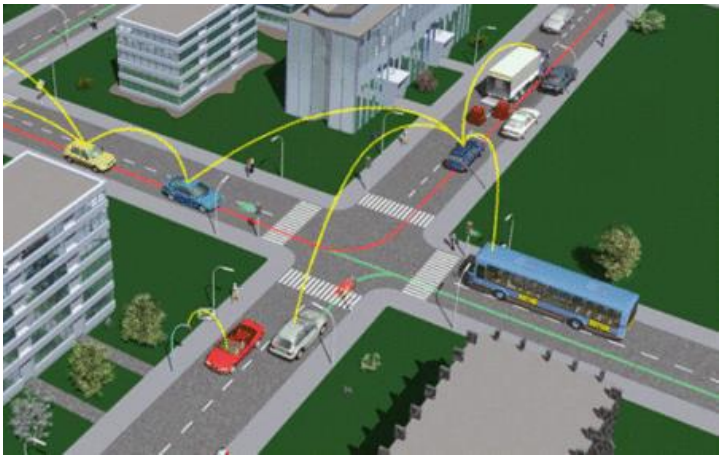

Figura 7. Sistemas de transporte inteligente

Tomado de (Toyota España, 2013)

1.6.1. Características de los Sistemas de transporte inteligente

Tabla 2. Características de Sistemas de Transporte Inteligente

Sistemas de Información Avanzados de Viajeros	Provisión de información de tráfico en tiempo real
	Guía de ruta/sistemas de navegación
	Información de estacionamiento
	Sistemas de información meteorológica
Sistemas Avanzados de Administración del Transporte	Centros de operación del tráfico
	Control adaptable de señales de tránsito
	Señales de mensajes dinámicos
Sistemas de Tarifas de Transporte Habilitados	Peajes electrónicos
	Pago de tarifa o precio electrónico
	Líneas de expreso
	Tarifas de uso de vehículos por kilómetro recorrido
	Variables de las tarifas de estacionamiento
Sistemas de Transporte Público Avanzados	Información en tiempo real del estado del sistema de transporte público.
	Localización automática de vehículos
	Pago de tarifa electrónica
Vehículo a Infraestructura de Integración y Vehículo a Vehículo de Integración	Sistema de anticolisión en intersecciones
	Adaptación inteligente de la velocidad

Adaptado de (Ezell, 2010)

1.6.2. Estandarización para Sistemas de Transporte Inteligente

Se trata de un proceso en el cual se garantiza compatibilidad, reproducción, expansión, seguridad y calidad de elementos que se desarrollan independientemente dentro de un sistema, generando calidad, reducción de costos y tiempo.(GSD+, 2018b)

1.7. Geolocalización

Es una tecnología que utiliza datos obtenidos de un dispositivo para identificar o descubrir la ubicación real de una cosa o de un individuo en un entorno físico o virtual. Es apropiado conocer los problemas relativos a la seguridad y a la privacidad para utilizar esta herramienta de manera responsable.

Los datos de geolocalización pueden ser adquiridos en modo activo, conocido como geolocalización, basado en dispositivo de usuario; o, un modo pasivo conocido como geolocalización, basada en servidores por correlación de datos. (Ingenima, 2020).

1.8. Plataformas en la nube para IoT

Permiten el acceso general, almacenamiento y volumen de los datos obtenidos por los dispositivos, ofreciendo escalabilidad y elasticidad.

A continuación, se realiza una breve explicación de 4 plataformas Open Source:

1.8.1. Kaa IoT

Como se puede apreciar en la Figura 8, se trata de una tecnología middleware de código abierto que cuenta con una arquitectura de microservicios, permite implementar aplicaciones en cualquier

lenguaje de programación y brinda una gestión flexible en lo que al ciclo de vida de las credenciales se refiere.(Ortiz Monet, 2019)

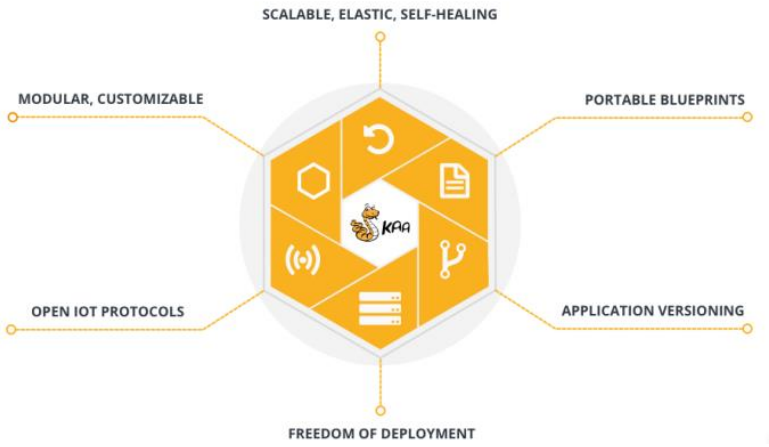

Figura 8. Características de Kaa IoT

Tomado de (Ortiz Monet, 2019)

1.8.2. OpenIoT

Es una plataforma de middleware de código abierto que agrupa IoT con los servicios de computación en la nube. Ofrece el registro, implementación y descubrimiento de sensores según su tipo y ubicación. Puede desplegar aplicaciones sencillas prácticamente sin programar. (Ortiz Monet, 2019). Figura 9.

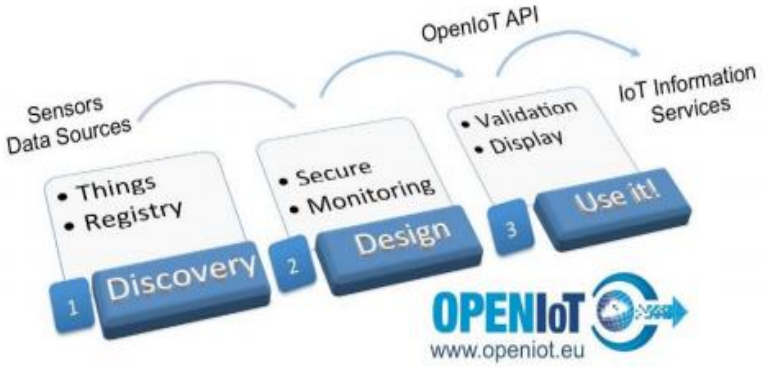

Figura 9. Diagrama de funcionamiento de OpenIoT

Tomado de (Ortiz Monet, 2019)

1.8.3. ThingSpeak

ThingSpeak es una plataforma de IoT de código abierto que admite la recolección de los datos identificados por los sensores, almacenamiento en la nube y su posterior visualización y análisis. Además, permite el desarrollo de prototipos y construcción de sistemas IoT sin tener que configurar servidores o desarrollar software web. (Ortiz Monet, 2019). Figura 10.

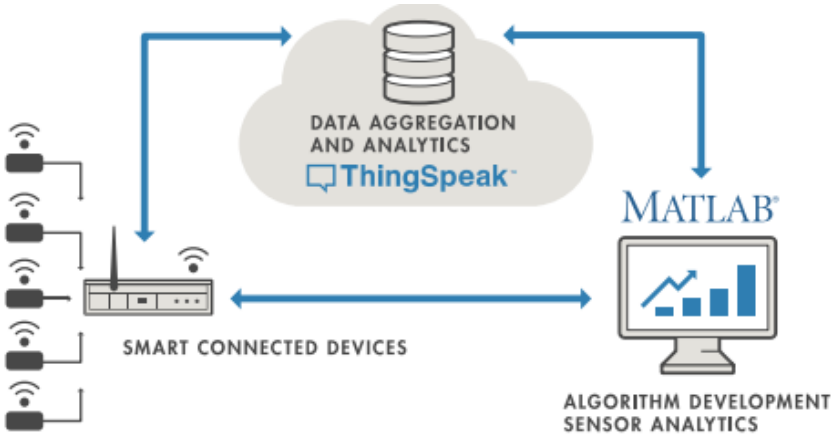

Figura 10. Estructura básica de ThingSpeak

Tomado de (Ortiz Monet, 2019)

1.8.4 M2MLight

Esta plataforma está diseñada para personas que tienen un conocimiento básico de los dispositivos de Internet of Thing (IoT). M2mlight.com es un servicio de plataforma básico gratuito para dispositivos IoT. Puede almacenar sus cámaras, sensores, actuadores y datos de alertas en la nube, como se aprecia en la Figura 11. Luego, usando su computadora de escritorio o teléfono inteligente a través de Internet, puede ver los videos, gráficos, mapas y estadísticas de estos dispositivos.(Albuja Sáenz, 2020)

Figura 11. Servidor M2MLight

Tomado de (Albuja Sáenz, 2020)

CAPÍTULO 2

2. Análisis de la situación actual del transporte en la ciudad de Quito y propuesta de despliegue de tecnología LoRa.

En este capítulo se presentará cuál es la situación actual del transporte en la ciudad de Quito y la propuesta de despliegue piloto con tecnología LoRa.

2.1. Organización

La situación actual del Sistema Integrado de Transporte Público del Distrito Metropolitano de Quito (DMQ) está conformada por dos subsistemas: El subsistema convencional, de rutas y frecuencias, y el subsistema convencional Metrobus-Q. Su diferencia está en sus capacidades de control y servicio.

2.1.1. Sistema Metrobus – Q

El sistema Metrobus-Q, se encuentra constituido por los servicios de transporte: Trolebús, Ecovía, Corredor Central Norte, y Corredor Suroriental, cada uno a su vez con su conjunto troncal y servicio de alimentadores. Como se aprecia en la Figura 12, es parte del Sistema Integrado de Transporte Público Metropolitano (SITPM), que se encarga de administrar los sistemas masivos de transporte de la ciudad, tanto públicos como privados.(Agencia de Ecología Urbana de Barcelona, 2017)

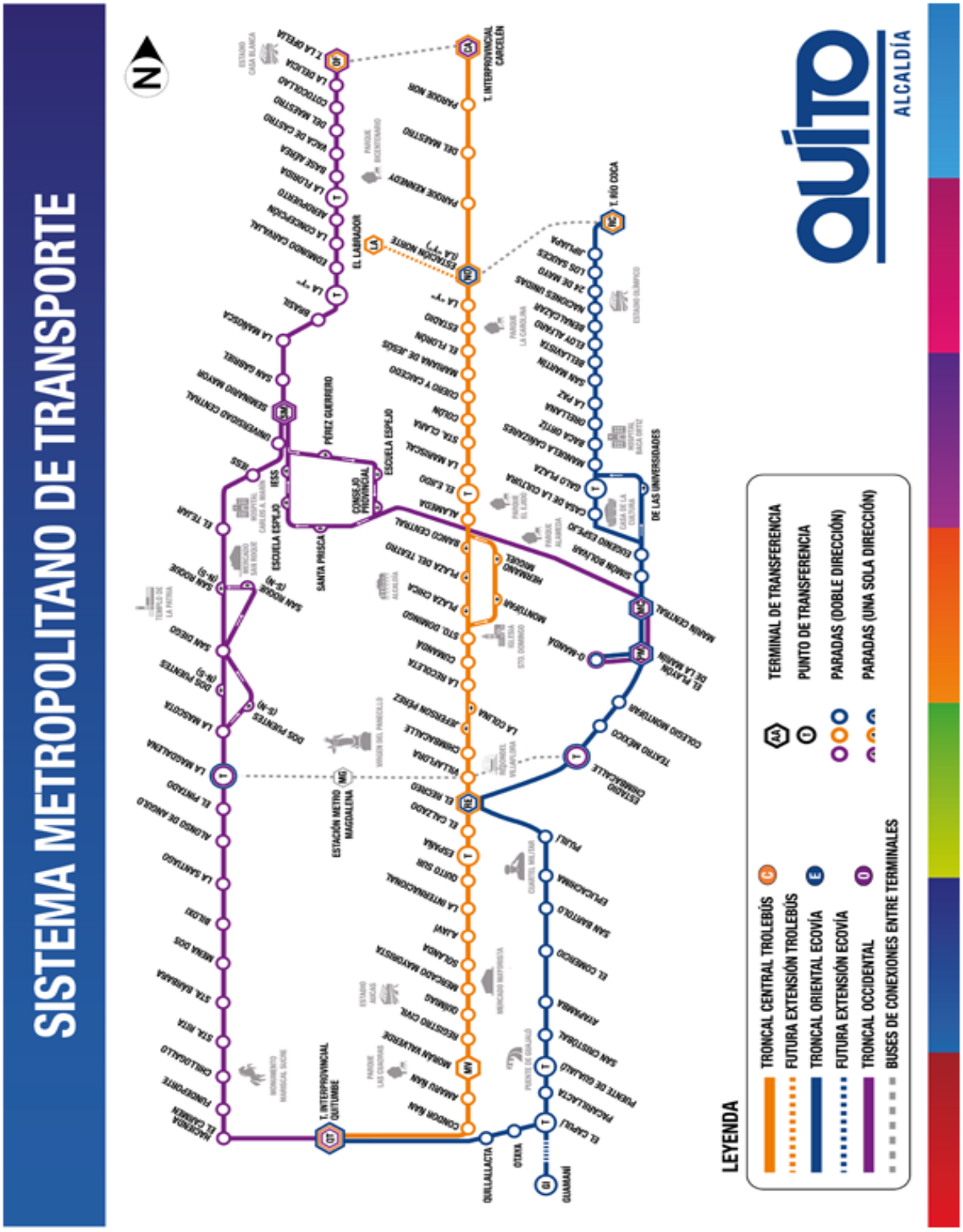

Figura 12. Sistema Metrobus-Q

Tomado de (Quito, 2019b)

2.1.2. Sistema Convencional

El Transporte Convencional, como se indica en la Figura 13, se caracteriza por ser desarticulado, unimodal e ineficiente, sujeto a grandes requerimientos funcionales por las prohibiciones de accesibilidad. Es gestionado por medio de contratos de operación, firmados con los distintos operadores legalmente establecidos en el Distrito Metropolitano de Quito; en dichos contratos se instituye el número de rutas, flota asignada, horarios y frecuencias de operación.

A continuación, se detallan las principales características de la red actual de buses convencionales:

- Existe redundancia de rutas en las principales avenidas que conectan la ciudad.
- Las demandas locales (barrios y parroquias) impiden que se conforme una red lógica y eficiente.
- La red actual no es legible, ni entendible para el ciudadano. Cada usuario únicamente conoce “su” ruta/s.(Agencia de Ecología Urbana de Barcelona, 2017)

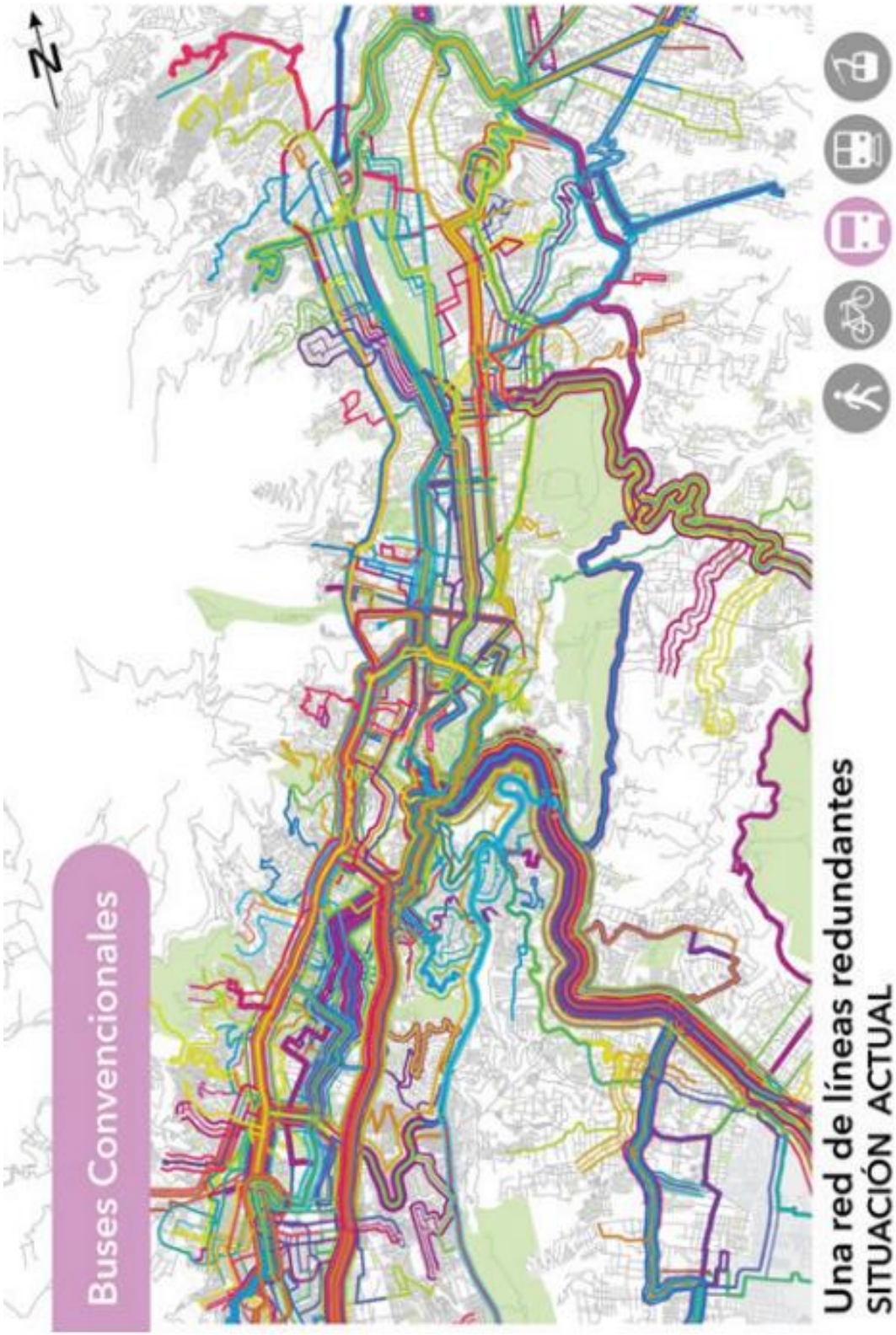

Figura 13. Esquema de Sistema de Buses Convencionales de la ciudad de Quitos

Tomado de (Agencia de Ecología Urbana de Barcelona, 2017)

2.2. Datos actualizados

En la ciudad de Quito, el sistema de transporte actualmente conforma un sistema desintegrado, de baja calidad de servicio, que atiende a los segmentos de demanda bajo diseños tradicionales ineficientes. Las diferentes líneas de autobuses urbanos no se complementan adecuadamente y su limitada integración causa muchos inconvenientes al momento de viajar y realizar transferencias y transbordos. Por tanto, los usuarios, son mal atendidos y no cuentan con una red que mejore y proporcione su integración al sistema.(Agencia de Ecología Urbana de Barcelona, 2017)

En la ciudad de Quito, se han realizado diferentes estudios de movilidad que han revelado que el sistema de transporte atraviesa una situación muy problemática y que necesita una reorganización integral.

Con respecto a los avances tecnológicos, hoy en día la ciudad de Quito cuenta con un proyecto de ciudad inteligente al 2022 llamado plan “Orquestador de la Movilidad”, lo cual implica articular un proceso de transformación en la planificación de la ciudad, buscando situar a la capital a la vanguardia tecnológica en América Latina.

El plan mencionado anteriormente, se basa fundamentalmente en la innovación tecnológica y busca un replanteo de la movilidad en todo el Distrito Metropolitano de Quito. Se trata de un sistema integral autosuficiente que contará con la ayuda de elementos como la inteligencia artificial, big data y machine learning que actuarán sobre la base de tres ejes primordiales: seguridad vial, movilidad inteligente y sostenibilidad.(Quito, 2019a)

El plan ‘Orquestador de la Movilidad’ buscará introducirse como una herramienta tecnológica de apoyo para la toma de decisiones que convertirá positivamente la convivencia vial. Así mismo, permitirá una movilidad inteligente que disminuirá los tiempos en los traslados de los ciudadanos; en definitiva, contribuirá a mejorar el estilo de vida de todos los quiteños.(Quito, 2019a)

2.3. Deficiencias en el Sistema de Transporte de la Ciudad de Quito

2.3.1. Desarticulación y solapamiento de la red

- Discrepancia de los subsistemas de transporte público para funcionar como un sistema integrado.
- Competitividad entre los sub sistemas urbanos Metrobus-Q y buses convencionales, creando una situación insostenible de solapamiento de la red y sobredimensionamiento de la flota.
- La competencia por pasajeros conduce a la redundancia de trayectos, resultando en ineficiencia e incremento de congestión.
- Gran redundancia de rutas en los principales Corredores de Transporte
- La individualidad de cada subsistema crea un impacto perjudicial que se refleja en la ineficiencia operativa, la mala calidad del servicio, y la inseguridad vial en general.(Agencia de Ecología Urbana de Barcelona, 2017)

2.3.2. Desorganización, Ineficiencia y Pérdida de Tiempo

- Ineficacia y redundancia del sistema, sumada a la congestión de las vías, lleva a largos tiempos de espera.
- Disparidades en la distribución del servicio: altas frecuencias y sobreoferta en algunas áreas y bajos niveles de servicio en otras.
- Baja conectividad y longitudes y tiempos de viaje desproporcionados.
- Calidad del servicio afectada gracias a la informalidad e insuficiencia de paradas formales sobre el territorio.

- Ausencia de mecanismos tecnológicos de auxilio al servicio.
- Dificultad en la lectura de la red.(Agencia de Ecología Urbana de Barcelona, 2017)

2.3.3. Congestión y Saturación de la Infraestructura vial

- Los grupos más vulnerables, peatones y no motorizados, son poco atendidos y no cuentan con una red que promueva y facilite su articulación al sistema.
- Existe deficiente activación del espacio público, que repercute en un círculo vicioso que aumenta el caos y congestión vehicular, empujando a la sociedad a depender del automóvil.
- El sistema actual de transporte público no es ni competitivo ni atractivo frente al vehículo particular.(Agencia de Ecología Urbana de Barcelona, 2017)

2.4. Perspectivas de diseño de un Sistema de Transporte Inteligente

2.4.1. Perspectiva de servicio

Esta perspectiva es el punto de partida para diseñar un sistema de transporte inteligente debido a que se enfoca en las vivencias del usuario al interactuar con el sistema, permitiendo de esta manera identificar como el SIT responde a sus necesidades. Adicionalmente, en este punto se establece las métricas de medición necesarias para evaluar la calidad del servicio del sistema.

Desde esta perspectiva de servicio es imprescindible que se ejecuten las siguientes labores:

- Identificar las necesidades del usuario que deben ser atendidas por el SIT.
- Identificar las funciones específicas que debe incorporar el piloto SIT para satisfacer las necesidades del usuario.

- Aprobar las funciones específicas del usuario, para garantizar la alineación entre las necesidades y las funciones para satisfacerlas.
- Detallar las funciones aprobadas para el cumplimiento de las necesidades del usuario.
- Identificar y definir todas las interfaces con el usuario. (GSD+, 2018b)

2.4.2. Perspectiva Operacional

Esta perspectiva contempla los procesos operacionales que se deben realizar para proporcionar adecuadamente los servicios identificados en la perspectiva de servicio.

Los aspectos operacionales de un SIT deben ser definidos con especial cuidado, debido a que afectan considerablemente al cumplimiento de los niveles del servicio del sistema:

- Se deben diseñar e implementar programas para socializar los beneficios y proporcionar capacitación a los usuarios sobre el uso del sistema.
- Identificar los cambios que afecten la operación y mantener los lineamientos de los requerimientos propuestos.(GSD+, 2018b)

2.4.3. Perspectiva Tecnológica

Hace referencia a la tecnología necesaria para proveer los servicios a los usuarios y ejecutar los procesos operacionales. Por tanto, está perspectiva debe incorporar la caracterización de los componentes que constituyen la arquitectura tecnológica y las interfaces entre componentes. La perspectiva también debe incorporar la descripción de requerimientos funcionales y no funcionales de la plataforma tecnológica.

Dado lo expuesto, los aspectos principales a ser considerados en un SIT, son los siguientes:

- Utilización de estándares técnicos que favorezcan la flexibilidad y escalabilidad del sistema.
- Seleccionar la tecnología adecuada que permita satisfacer los requerimientos del sistema, considerando el posible cambio en las tendencias del mercado.
- Definir los requerimientos de seguridad del sistema. (GSD+, 2018b)

2.4.4. Perspectiva Institucional

Esta perspectiva se encarga de definir la manera en que se distribuyen las tareas entre entidades gubernamentales y privadas para llevar a cabo la concepción, diseño, implementación, operación y financiamiento del sistema.

La gestión de interesados (stakeholders) consiste en la identificación de los actores que serán responsables de realizar el diseño, implementación, operación y financiamiento del sistema. Por lo cual, es necesario definir los roles y responsabilidades según su campo de acción empresarial.

La perspectiva institucional debe garantizar a los interesados:

- Conseguir que todos los actores sean identificados al iniciar del proyecto, debido a que la incorporación tardía de requerimientos o restricciones de los actores tendrá impactos directos en el tiempo y costo de la implementación y la operación del sistema.
- Identificar elementos que permitan a los actores definir sus requerimientos junto a un procedimiento para realizar su validación y establecer una metodología para la jerarquización o priorización de las necesidades de los diferentes actores.(GSD+, 2018a)

La gestión de adquisiciones implica la elección de la estrategia y del modelo de contratación de los sistemas SIT, en función a los requerimientos del sistema, la complejidad del sistema, los

tiempos de implementación, entre otros. Para ello, se deben tomar en consideración los puntos que se describen a continuación:

- Realizar una definición previa de cómo se desea efectuar la asignación de riesgos entre las partes involucradas en el contrato.
- Seleccionar un modelo de contratación, tomando en cuenta la capacidad de la entidad contratante, la complejidad del sistema y los tiempos requeridos de implementación.
- Incorporar en los contratos de los actores, niveles de servicio que sean fácilmente medibles. Esto facilita la fiscalización de la operación de los actores y la generación oportuna de multas. (GSD+, 2018b)

2.4.5. Perspectiva Económica

Esta perspectiva comprende el proyecto SIT desde el análisis socioeconómico, el análisis financiero y análisis comercial. Dado que los SIT son sistemas sociales, que involucran personas e interacciones entre éstas para su funcionamiento, es oportuno diseñar el sistema teniendo en cuenta las consecuencias que trae la implementación del sistema en la sociedad. Específicamente esta perspectiva debe responder las preguntas de los cuatro quién: ¿Quién se beneficia?, ¿Quién paga?, ¿Quién provee? y ¿Quién pierde?

Un análisis económico del SIT debe ser capaz de descubrir los ganadores con su implementación y ganancia, en términos no monetarios. Parte de este análisis es desarrollado en conjunto con la perspectiva de servicio, ya que el sistema debe estar orientado a que sean los usuarios del sistema quienes se beneficien.

Desde la perspectiva económica es primordial tener en cuenta los siguientes aspectos:

- Contar con pleno conocimiento de la legislación local en materia de fondeo de proyectos SIT.
- Identificar múltiples fuentes de financiamiento y generar fondos de contingencia para costos inesperados.
- Realizar análisis de sensibilidad y estimaciones sobre las variables con incertidumbre.
- Construir fórmulas de remuneración a los actores que incorporen el cumplimento de los niveles de servicio agregados en sus contratos. (GSD+, 2018b)

2.5. Perspectivas de mejora utilizando tecnología LoRa

Al implementar tecnología LoRa se busca la profesionalización y modernización, que permitirá incrementar la confiabilidad, cobertura y calidad del servicio. Todo esto se puede lograr gracias a que LoRa ofrece grandes ventajas, además de ser un protocolo de red de largo alcance y robusto frente a interferencias, esta tecnología se caracteriza principalmente por su bajo consumo, lo que aporta la gran posibilidad de aplicar al transporte inteligente en la ciudad de Quito. Cabe mencionar que LoRa ha sido implementada ya en más de 100 países en el mundo con resultados satisfactorios.

2.5.1. Propuesta de despliegue de tecnología LoRa

En la actualidad, muchos países de gran desarrollo tecnológico han ido adaptando el concepto de smart mobility a sus ciudades, lo que les ha permitido realizar algunas implementaciones con tecnologías como lo son LoRa y Sigfox; siendo LoRa una tecnología dominante, por sus características de largo alcance, ancho de banda, bajo consumo de energía, seguridad integrada y sobre todo bajo coste de sus dispositivos, convirtiéndola en una solución tecnológica muy económica, atractiva y fácil de implementar.(Cárdenas et al., 2018)

Por tanto, y en vista de los problemas existentes en la movilidad y calidad de servicio del transporte en la ciudad de Quito, basados en los principios de la tecnología LoRa, se plantea desplegar una red piloto de comunicaciones inalámbricas que permitan mostrar la ubicación y velocidad de los autobuses en tiempo real y brindar información acerca del tiempo de llegada, como se muestra en el esquema de la Figura 14, lo que mejorará la información que el usuario reciba y pueda optimizar de mejor manera su tiempo y recursos.

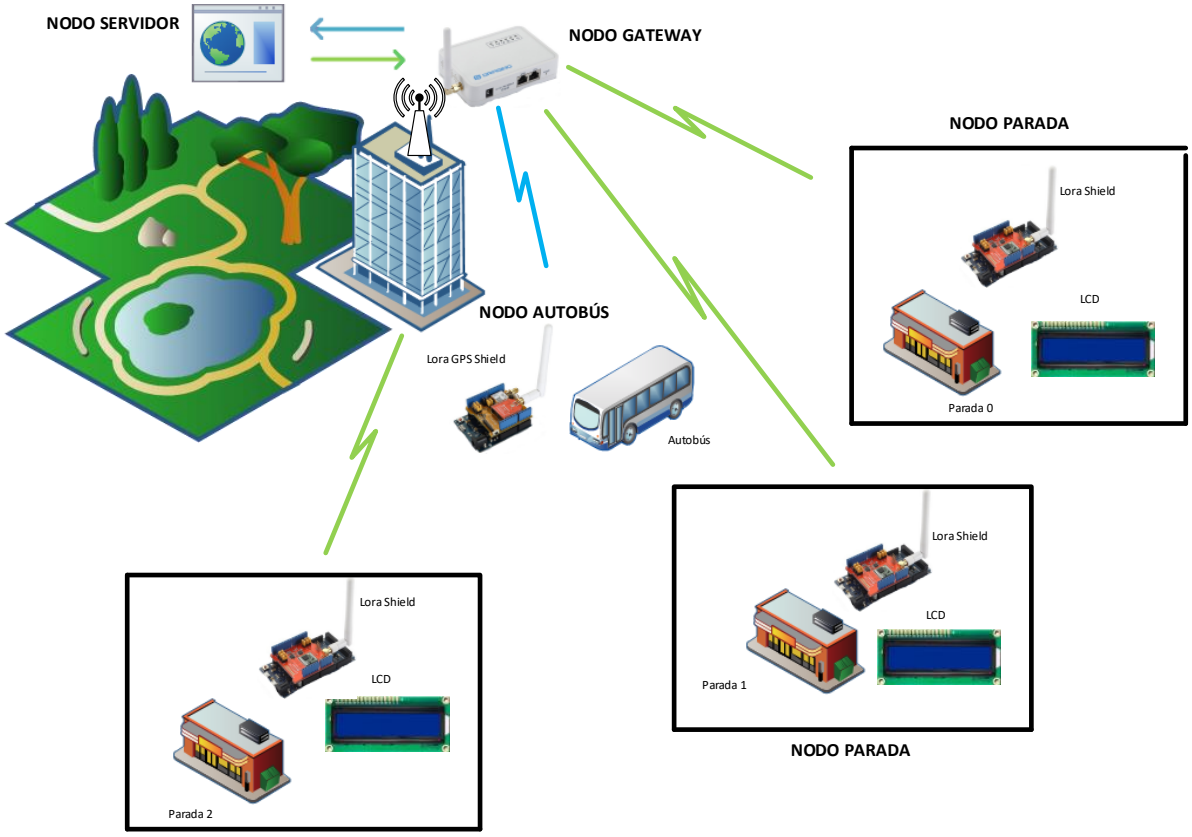

Figura 14. Propuesta de despliegue piloto con tecnología LoRa

Como puede observarse en el esquema, se propone utilizar 4 tipos de nodos que se describen a continuación:

- **Nodo autobús:** recibirá los datos de ubicación y velocidad por GPS y enviará la información por canal de radio Lora al nodo Gateway.
- **Nodo parada:** recibirá la información del nodo Gateway, enviada por el nodo servidor, mediante canal de radio LoRa y procesará los datos recogidos para presentar la información del tiempo de arribo del autobús en LCD.
- **Nodo Gateway:** se encargará de recoger los paquetes de datos enviados por el nodo autobús a través de tecnología LoRa y transmitirlos por internet a un servidor en la nube, para luego enviar la información de arribo en tiempo real en cada nodo parada.
- **Nodo Servidor:** recibirá los datos del nodo autobús a través del nodo Gateway, para luego calcular el tiempo de arribo del autobús a las paradas y enviarlo hacia el nodo Gateway donde se indicará la información en cada parada por medio de un dispositivo LCD. Además, se podrá visualizar la posición y velocidad del autobús en página web en tiempo real.

CAPÍTULO 3

3. Diseño de Piloto IoT

El diseño piloto está conformado por 4 nodos: nodo autobús, instalado en un vehículo particular; nodo Gateway, instalado en un lugar alto que proporcione un área de cobertura admisible; nodo parada, instalado en paradas definidas por rutas de buses existentes; y, nodo servidor, ubicado en la nube.

3.1 Diseño de comunicación Lora

En este primer punto se especifica la configuración de parámetros BW, SF y CR que permitirán definir, el rango de cobertura, velocidad de transmisión de datos y su robustez frente a interferencias. Cabe mencionar que la frecuencia de operación debe ser seleccionada tomando en consideración la regulación de frecuencias adoptadas en la región en donde será realizado el piloto.

Los parámetros de configuración deben ser elegidos de la Tabla 3 desarrolla por Dragino que se muestra a continuación:

Tabla 3. Configuraciones para dispositivos radio Lora

	Parámetros			
Configuración	**BW**	**SF**	**CR**	**Descripción**
1	125 KHz	7	4/5	Medio+ Medio alcance
2	500 KHz	7	4/5	Rápido + Corto alcance
3	32 KHz	9	4/8	Lento + Largo alcance
4	125 KHz	12	4/8	Lento + Largo alcance

Tomado de (Palacios Ibarra & Zúñiga Pérez, 2019)

Por lo expuesto se muestra el Diagrama 1 que permitirá la conexión entre dispositivos LoRa, siendo para esto indispensable la activación del puerto serial y cada uno de los módulos para la transmisión y recepción de datos.

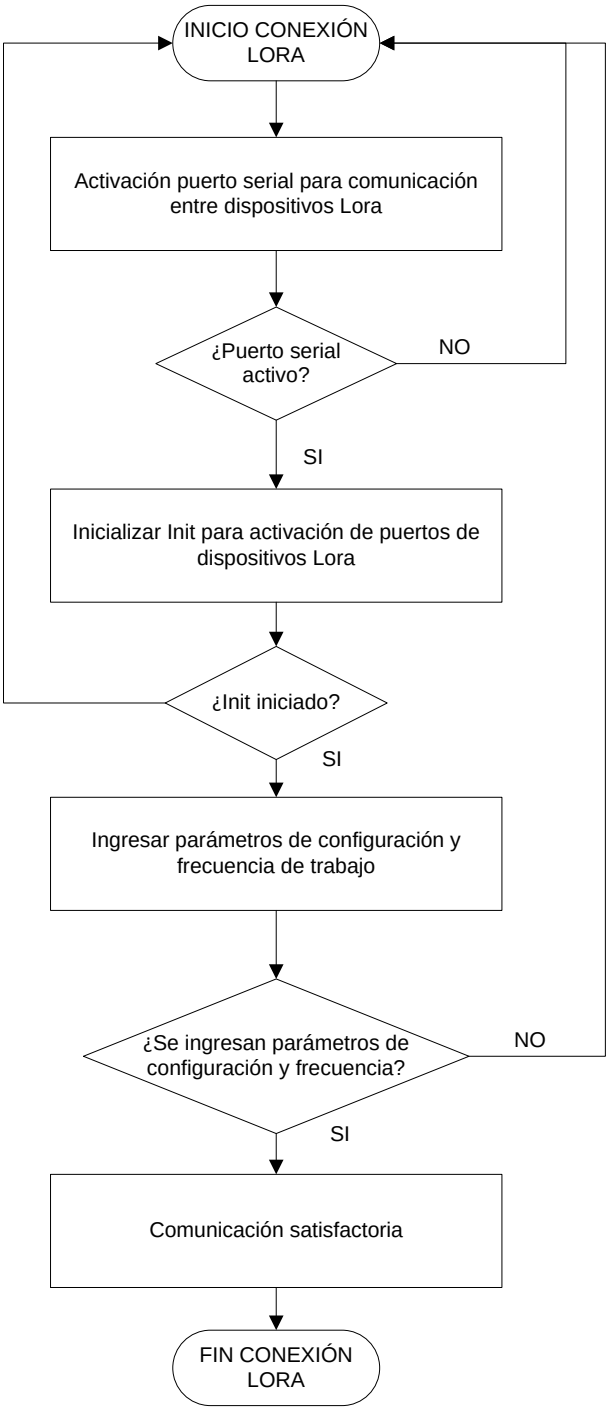

Diagrama 1. Diseño comunicación Lora, basado en (Palacios Ibarra & Zúñiga Pérez, 2019)

3.2 Diseño de nodo autobús

Este módulo será el responsable de captar los datos de geolocalización y velocidad emitidos por un dispositivo GPS, y a su vez enviará los datos válidos identificados al nodo Gateway por medio del canal de radio LoRa dispuesto para esta función.

Para realizar este diseño se utilizará LoRa GPS Shield v95+Arduino UNO, dispositivo que será expuesto a continuación con la finalidad de puntualizar y definir los puertos que serán utilizados para no tener inconvenientes en la comunicación.

3.2.1 Dispositivos para diseño nodo autobús

3.2.1.1. LoRa GPS Shield v95+Arduino UNO

El dispositivo clase C de LoRa, que consta en la Figura 15, se basa en el chip semtech SX1276 / SX1278, desarrollado por Dragino, es un transceptor de largo alcance que trabaja para el sistema Arduino y se fundamenta en la biblioteca de código abierto. Facilita comunicación de largo alcance de espectro extendido y alta inmunidad a interferencias.(Dragino, 2020a)

3.2.1.2. Especificaciones técnicas Lora GPS Shield v95

- Velocidad de bits hasta 300 kbps.
- Sensibilidad: hasta -148 dBm.
- Inmunidad de bloqueo.
- Modulación FSK, GFSK, MSK, GMSK, LoRaTM y OOK.
- Rango dinámico RSSI: 127 Db.
- Detección automática de RF y CAD con AFC.
- Sensor de temperatura incorporado e indicador de batería baja.(Dragino, 2020a)

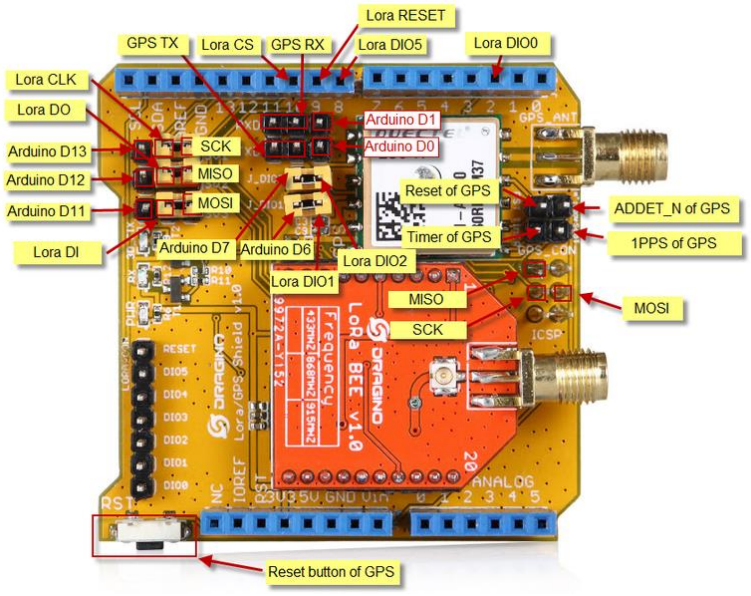

Figura 15. Lora Shield para Arduino

Tomado de (Dragino, 2020a)

La parte de GPS está basada en MYK MT3339, diseñado para aplicaciones que requieren información de ubicación o sincronización, su conexión con Arduino es a través del puerto serie.

3.2.1.3. Especificaciones técnicas de GPS

- Cumple con GPS, SBAS.
- Velocidad de bits hasta 300 kbps.
- Interfaces seriales UART: 4800 ~ 115200 bps ajustables, Predeterminado: 9600bps.
- Voltaje de E / S: 2.7V ~ 2.9V.
- Protocolos: NMEA 0183, PMTK.
- Sensibilidad -148dBm, seguimiento -165dBm, readquisición -160dBm.
- Temperatura funcionamiento: -40° C a 85° C.
- Temperatura de almacenamiento: -45° C a 125° C.
- Altitud de rendimiento dinámico: Máx.18000m.

- Velocidad máxima: Máx.515m / s.
- Receptor de banda L1 (1575.42MHz) Canal 22 (Seguimiento) / 66 (Adquisición).(Dragino, 2020a)

3.2.2 Conexiones entre dispositivos para nodo autobús

La comunicación entre dispositivos será realizada por medio de SPI, que es un protocolo de comunicación serial síncrono full dúplex, que permite recibir y transmitir información al mismo tiempo.

Para realizar las conexiones, se procede a identificar la disponibilidad de pines de cada uno de los dispositivos con ayuda de la Tabla 4, logrando el diseño circuital del nodo autobús.

Tabla 4. Distribución de Pines LoRa GPS Shield v95+Arduino

LoRa GPS Shield v95+Arduino	Disponibilidad
SLC	Utilizable
SDA	Utilizable
REF	Utilizable
GND	Utilizable con condición
D13	Digital Utilizable con condición
D12	Digital Utilizable con condición
D11	Digital Utilizable con condición
D10	Digital Utilizable con condición
D9	Digital No Utilizable
D8	Digital Utilizable con condición
D7	Digital Utilizable con condición
D6	Digital Utilizable con condición
D5	Digital Utilizable
D4	Digital Utilizable
D3	Digital Utilizable
D2	Digital No Utilizable
D1	Digital Utilizable
D0	Utilizable
A0	Analógico Utilizable
A2	Analógico Utilizable
A3	Analógico Utilizable
A4	Analógico Utilizable
A5	Analógico Utilizable

Tomado de* (*Dragino, 2020a)

Para conseguir que el GPS reciba los datos y Arduino pueda procesarlos, se realiza la conexión que se muestra en la Figura 16, en donde, se puede visualizar que GPS_RXD se conecta con el pin D4 y GPS_TXD a D3.

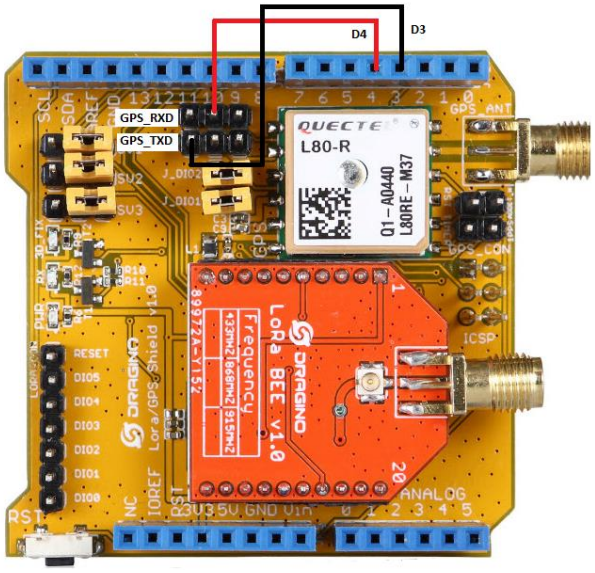

Figura 16. Conexión de GPS y Arduino UNO

Tomado de (Dragino, 2020a)

Para desarrollar el software de este módulo se debe realizar la comunicación de radio con la finalidad de obtener la ubicación y velocidad actual del autobús a través del dispositivo GPS, una vez obtenidos los datos, serán enviados al nodo Gateway por el canal radio LoRa, configurado en la frecuencia de 915MHz. Diagrama 2.

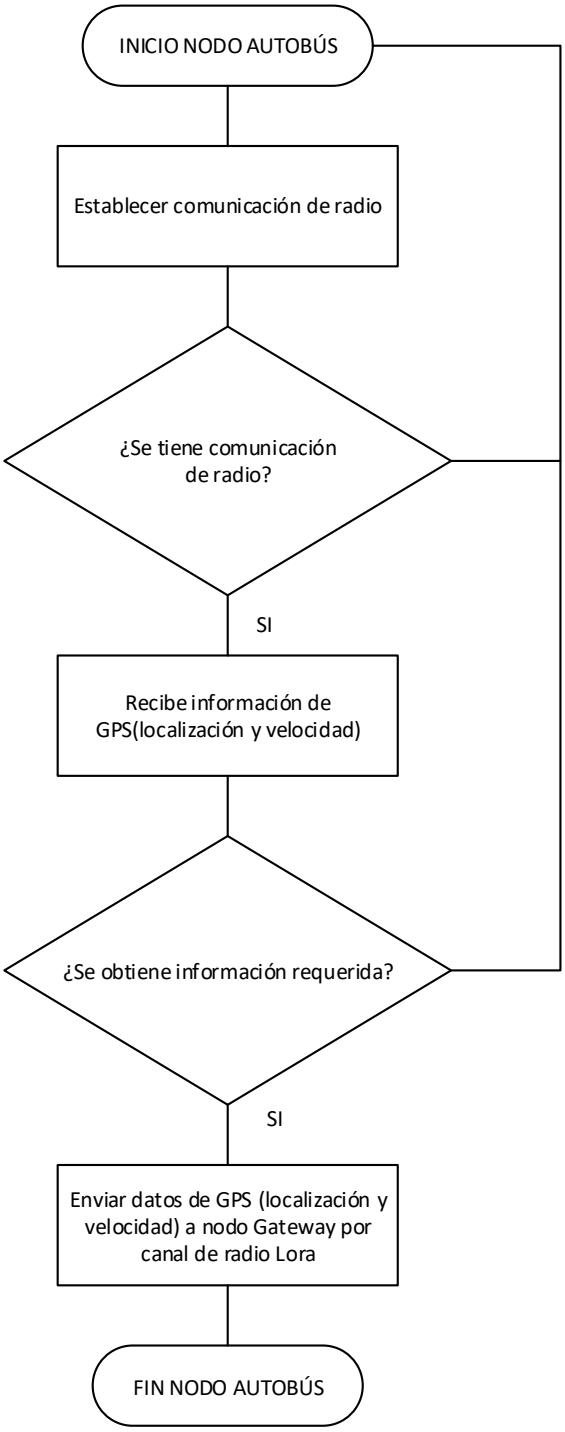

Diagrama 2. Diseño Nodo Autobús

3.3 Diseño de nodo Gateway

Este nodo será el responsable de receptar y transmitir toda la información que viene del nodo autobús por medio del canal de radio LoRa hacia el nodo servidor de red centralizado por medio de Ethernet y a su vez este responderá al nodo Gateway con información de llegada del autobús en tiempo real a los nodos parada. Diagrama 3.

3.3.1 Dispositivos para diseño nodo Gateway

Para la implementación de este nodo receptor y transmisor de información solamente se requiere un Gateway LoRa LG01-P que se describe a continuación.

El LG01-P es un dispositivo LoRa Gateway de un solo canal de código abierto, que admite conectar la red inalámbrica LoRa a una red IP a través de WiFi, Ethernet o celular 3G / 4G a través del módulo LTE opcional. Además, ejecuta el sistema Open Source OpenWrt, donde el usuario puede cambiar el archivo fuente o compilar el sistema para permitir sus aplicaciones personalizadas. (Dragino Technology Co., 2019) Figura 17.

3.3.1.1. Especificaciones técnicas Gateway LoRa LG01-P

Linux Side:

- Procesador: 400MHz, 24K MIPS
- Memoria Flash: 16MB
- Memoria RAM: 64MB

MCU/LoRa Side:

- Microcontrolador: ATMega328P
- Memoria Flash:32KB.
- Memoria RAM:2KB

- Chip LoRa: SX2176/78

Interfaces:

- 2 Puertos RJ45:10M/100M
- Wi-Fi: 802.11 b/g/n
- Wireless LoRa
- Voltaje de entrada: 12V DC
- 1 conector USB 2.0 (Dragino Technology Co., 2019)

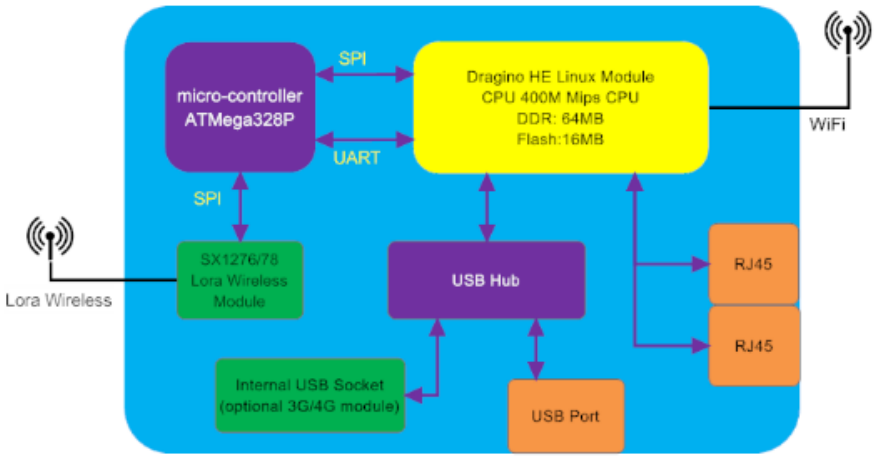

Figura 17. Arquitectura Gateway LG01-P

Tomado de (Dragino Technology Co., 2019)

Después de identificar las características y la arquitectura, se procede a realizar la conexión de la red por browser con la IP: 10.130.1.1 que permite el ingreso a la interfaz de DRAGINO que permitirá realizar las configuraciones necesarias para tener comunicación con los dispositivos.

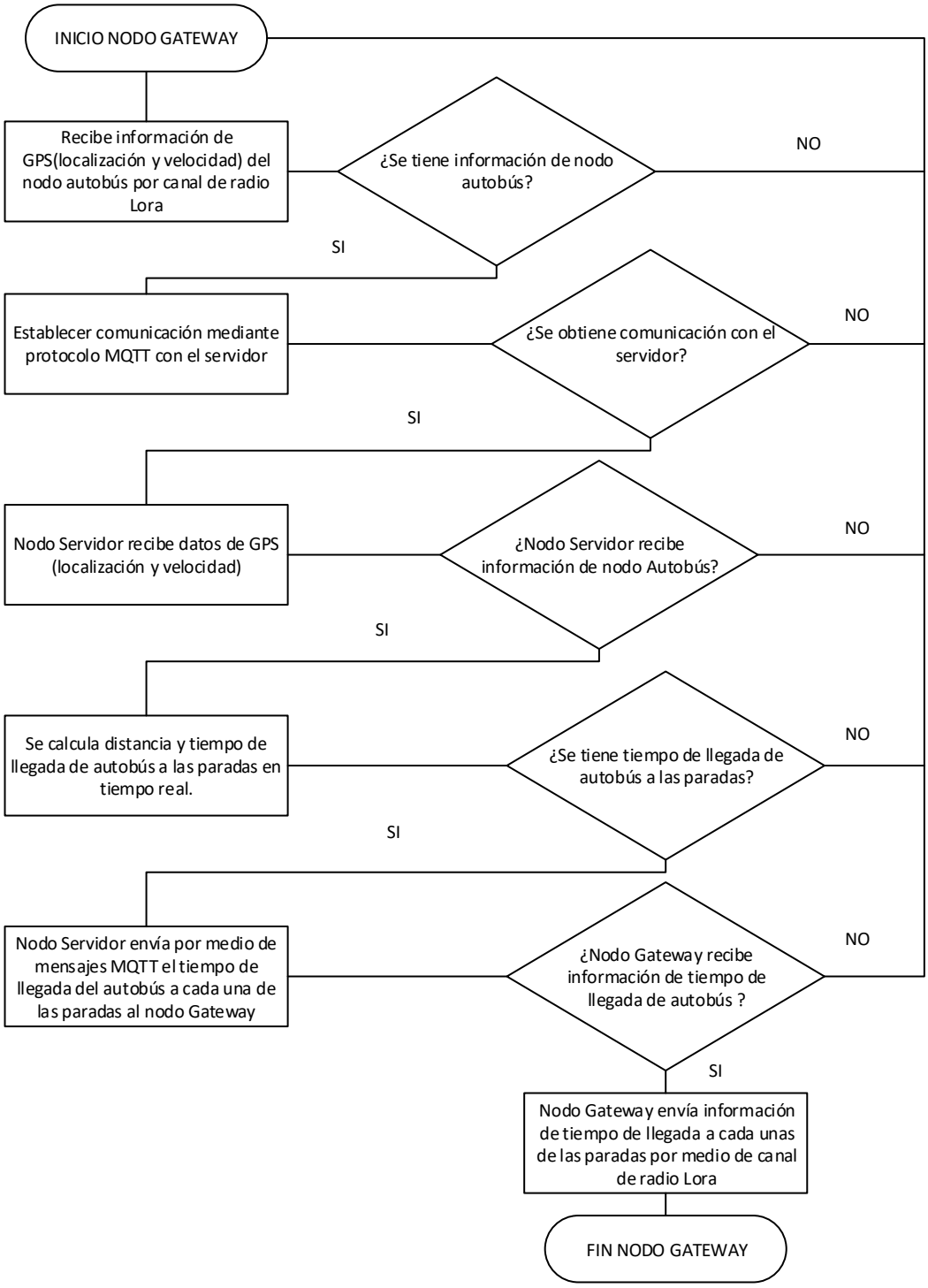

Diagrama 3. Diseño Nodo Gateway

3.4 Diseño de nodo Parada

El nodo parada se encargará de recibir la información del tiempo de llegada del autobús a través del nodo Gateway y la mostrará en un dispositivo LCD al usuario, como se muestra en el Diagrama 4. Cabe mencionar que el piloto solo dispondrá de tres nodos parada en línea recta, por temas de emergencia sanitaría COVID-19 se tuvo que reducir el área de cobertura.

3.4.1 Dispositivos para diseño nodo parada

Para este nodo será necesario un módulo radio LoRa Shield v95+ Arduino UNO y un LCD 1602+Adaptador I2C.

3.4.1.1. LoRa Shield v95+ Arduino Mega UNO

Este dispositivo se basa en el chip Semtech SX1276 / SX1278 y en una biblioteca de código abierto. Admite al usuario enviar datos LoRa y obtener largas distancias a baja data-rate.

3.4.1.2. Especificaciones Técnicas LoRa Shield v95:

- Velocidad hasta 300 kbps.
- Sensibilidad: hasta-148 dBm.
- Bloqueo a inmunidad.
- Modulación: FSK, GFSK, MSK, GMSK, LoRaTM y OOK.
- Rango Dinámico de RSSI: 127dB.
- Sensor de temperatura incorporado e indicador de batería baja.(AliExpress, 2020)

3.4.1.3. LCD 1602

Es un módulo matriz de puntos que despliega letras, números y caracteres, etc. Se compone de posiciones de matriz de puntos de 5x7 o 5x11; cada posición puede mostrar un personaje. Hay un punto entre dos caracteres y un espacio entre líneas, separando así los caracteres y las líneas. El modelo 1602 significa que muestra 2 líneas de 16 caracteres.(Sunfouder, 2017)

Tabla 5. Conexiones entre Arduino y display LCD1602

LCD1602	Conexión Arduino MEGA, UNO, NANO
VSS	GND
VDD	5V
VEE	Potenciómetro
RS	D8
RW	GND
EN	D9
D4	D4
D5	D5
D6	D6
D7	D7
Led+	VCC
Led-	GND

Tomado de (Naylamp Mechatronics, 2016)

3.4.1.4. Controlador I2C para LCD 16X2

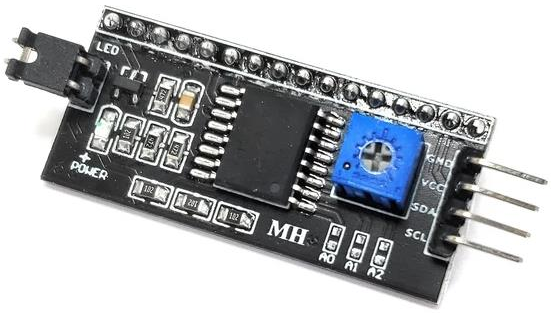

Figura 18. Controlador I2C

Tomado de (330ohms, 2018)

Como se observa en la Figura 18, el módulo I2C permite reducir los pines dedicados a la pantalla LCD y desplegar la información por solo dos pines.

3.4.1.3.1. Especificaciones técnicas controlador I2C

- Voltaje de alimentación: 5V
- Retroiluminación y contraste se ajustan mediante el potenciómetro.
- 2 Interfaces IIC
- Compatible para LCD 16x2 y LCD 20x4.

Para realizar el diseño de este nodo, se debe verificar si el nodo Gateway recibe la información del Nodo Servidor por protocolo MQTT; si recibe correctamente, el nodo Gateway envía la información del tiempo de arribo del autobús a cada una de las paradas por canal de radio LoRa y se mostrará por LDC.

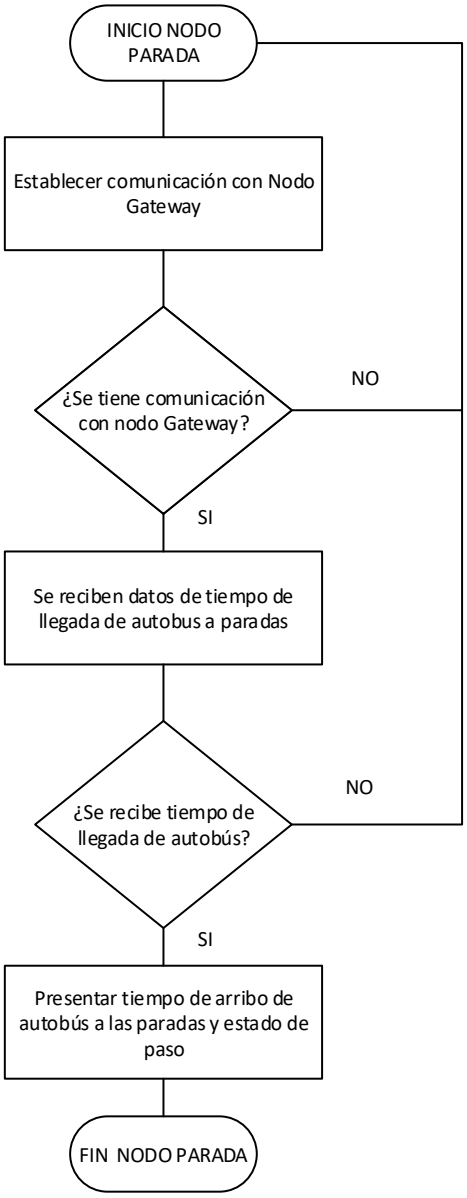

Diagrama 4. Diseño Nodo Parada

3.5 Diseño de Nodo Servidor

Como se indica en el Diagrama 5, el nodo servidor tendrá la función de recibir y trasmitir la información por medio del nodo Gateway mediante mensajes MQTT. Además, será responsable de presentar la información de ubicación del autobús por medio de página web sobre mapa de ruta, cálculo de la distancia y tiempo de llegada actual del autobús a cada parada.

Para dar solución a este diseño se han seleccionado el servidor M2MLIGHT que fue descrito en el capítulo 1 de fundamentos teóricos.

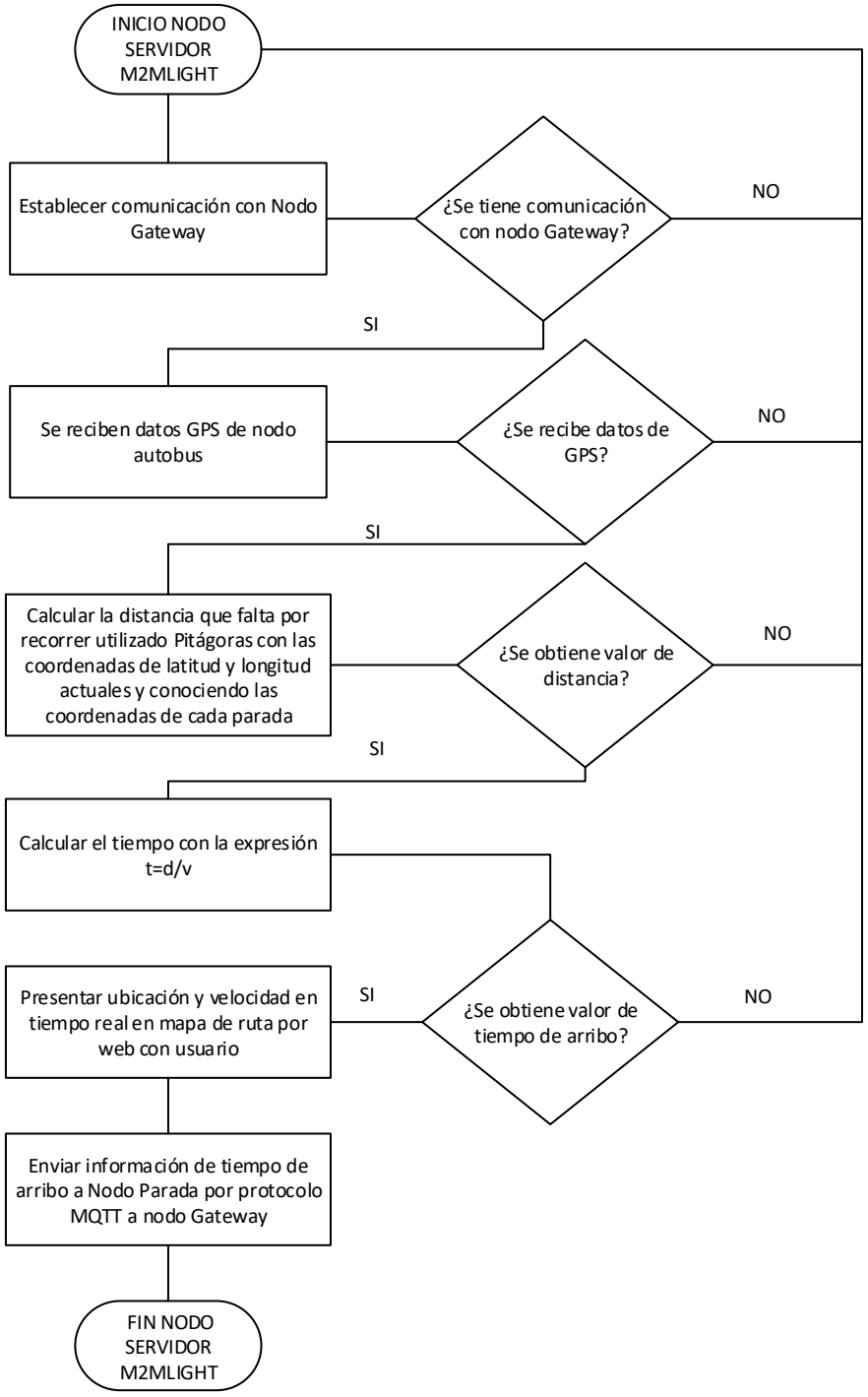

Diagrama 5. Diseño Nodo Servidor

3.6 Diagrama de despliegue de Piloto

A continuación, en el Diagrama 6 de despliegue del piloto, se indica la información del tiempo de arribo del autobús a cada una de las paradas, con la opción adicional de poder visualizar por la web la posición y velocidad en tiempo real del autobús.

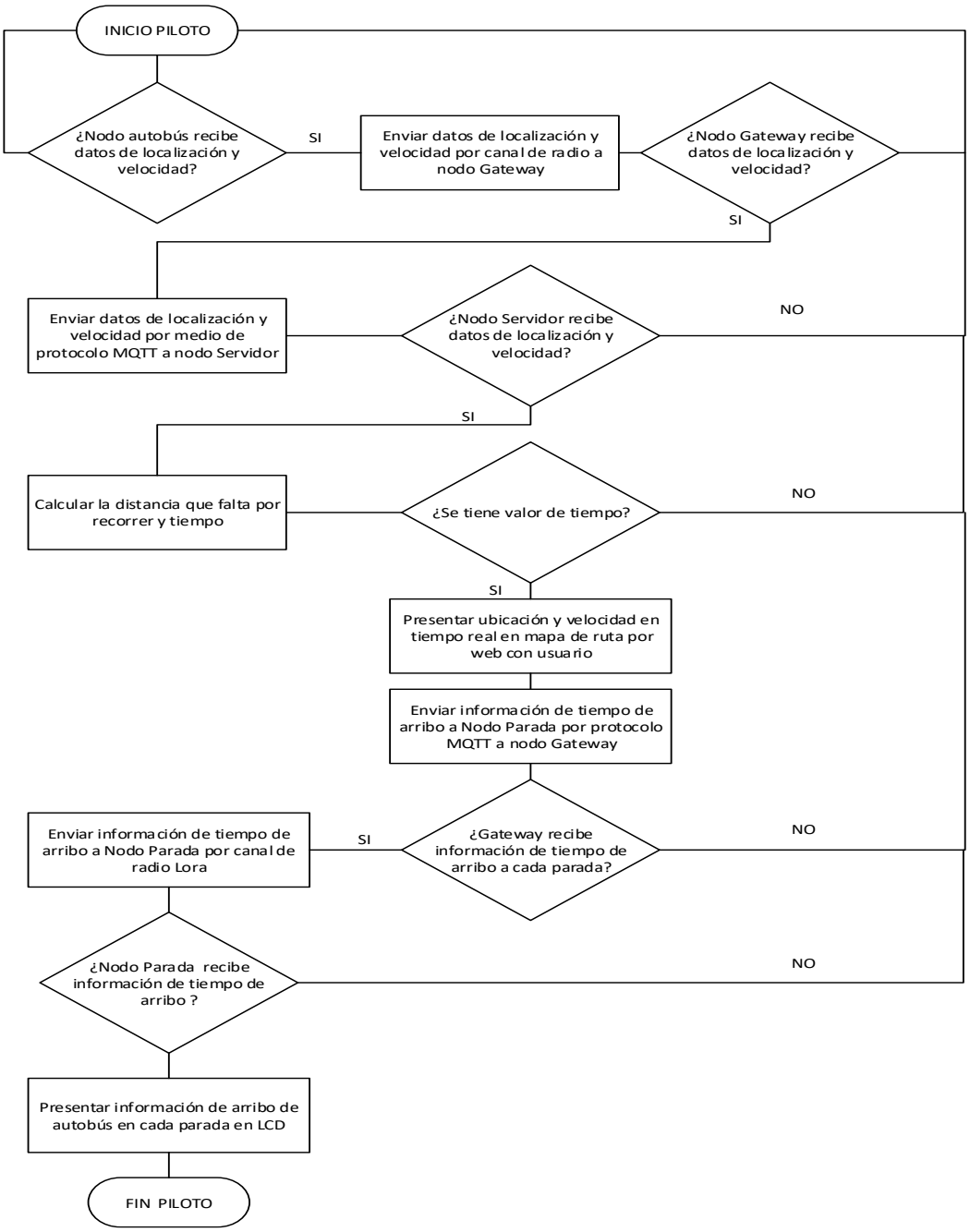

Diagrama 6. Despliegue Piloto

3.7 Costos de Implementación del prototipo

Ante la imposibilidad de conseguir los dispositivos LoRa, utilizados en el piloto propuesto, en el mercado ecuatoriano, éstos fueron adquiridos en Amazon. El desglose se encuentra detallado en las Tablas 6 y 7.

Tabla 6. Detalle de costos Dragino LoRa IoT Kit de desarrollo V2 915MHZ

Detalle	Costo
Precio	$167,99
Envío de AmazonGlobal	$27,89
Depósito de derechos de importación estimado	$96,94
Total	**$292,82**

Tomado de (Amazon, 2020b)

Tabla 7. Detalle de costos Dragino Lora GPS Shield 915Mhz

Detalle	Costo
Precio	$27,99
Envío de AmazonGlobal	$17,79
Depósito de derechos de importación estimado	$16,48
Total	**$62,26**

Tomado de (Amazon, 2020a)

Los elementos que se describen a continuación, en la Tabla 8, fueron adquiridos en TECmikro, ubicado en la ciudad de Quito.

Tabla 8 Detalle de costos de Arduino UNO y LCD 1602+I2C.

Detalle	Cantidad	Costo incluido IVA	Total
Arduino UNO	1	$10,60	$10,60
LCD 1602+I2C	2	$7,45	$14,90
			$25,50

Tomado de (Ecuador, 2020)

A continuación, en la Tabla 9, se presenta el detalle de gastos totales generados para la implementación de este proyecto piloto, por un total de USD $ 380,58.

Tabla 9. Detalle de Gastos Totales de la Implementación

Detalle	Cantidad	Costo	Total
Arduino UNO	1	$10,60	$10,60
LCD 1602+I2C	2	$7,45	$14,90
Dragino LoRa IoT Kit de desarrollo V2 915MHZ	1	$292,82	$292,82
Dragino Lora GPS Shield 915Mhz	1	$62,26	$62,26
			$380,58

CAPÍTULO 4

4. Implementación de Piloto

La implementación del piloto se basa en componentes de hardware, software y el proceso de comunicación entre nodos, tomando en consideración la lógica de conexión y funcionamiento correspondiente.

Para el desarrollo del software es necesario utilizar las librerías y clases que se mencionan a continuación:

- **Librería SoftwareSerial.h:** es una librería desarrollada para permitir la comunicación serial en otros pines digitales del Arduino, usando software para replicar la funcionalidad. Se pueden tener varios puertos seriales de software con velocidades de hasta 115200 bps.(Arduino, 2015)
- **Librería TinyGPS.h:** facilitará la identificación de la velocidad, longitud, latitud y otros parámetros sin tener que recurrir a algoritmos complejos para lograr obtenerlas.
- **Librería SPI.h:** es una librería diseñada específicamente para negociar la transmisión de datos en serie entre una placa Arduino y un terminal cualquiera.(jlquijado, 2016b)
- **Driver RH-RF95.h:** se trata de un controlador de la librería Radiohead, que admite el envío y recepción de mensajes en paquetes a través de una variedad de radios de datos comunes. (Doxigen, 2017)
- **Librería LiquidCrystal_I2C.h:** esta librería permite crear un objeto que representa al display LCD y que contiene todas las operaciones “de bajo nivel” para que la programación de este dispositivo resulte más fácil.(jlquijado, 2016a)

- **Librería Bridge.h:** se trata de una biblioteca puente para dispositivos Yún, la cual simplifica la comunicación entre ATmega32U4 y AR9331 actuando como la interfaz para la línea de comandos de Linux.(Arduino, 2020b)
- **Clase Process.h:** es una clase de la biblioteca Bridge que permite ejecutar procesos Linux en el AR9331, esto con el propósito de transferir datos desde un servidor web y obtener información sobre el procesador Linux.(Arduino, 2020a)
- **Clase FileIO.h:** es una clase que permite escribir y leer una tarjeta SD montada en un dispositivo Yún. Es parte de la biblioteca Bridge. No es llamada directamente, sino es invocada cada vez que utiliza una función que se basa en la misma. (Arduino, 2020c)

4.1. Implementación Comunicación LoRa entre nodos

Como se indica en la Figura 19, en este punto es necesario el uso de las librerías <SoftwareSerial.h> y el driver <RH_RF95> como se muestra en las líneas de código realizadas en Software Arduino.

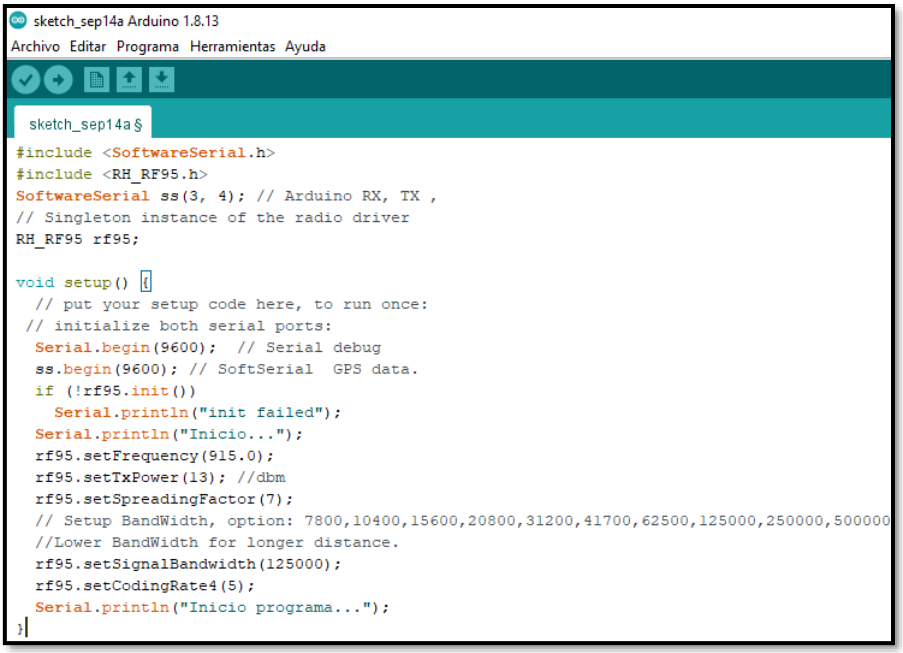

Figura 19. Comunicación LoRa y configuración de parámetros

En la figura 19, se observa que se definen en primera instancia los pines de Rx y Tx para la recepción y envió de la información. A continuación, se inicializa la comunicación serial con el computador en 9600 bps. Luego se establecen los valores de los parámetros de configuración de frecuencia=915 MHz, potencia de transmisión= 13dbm, factor de propagación=7, ancho de banda= 125KHz y la tasa de codificación=4/5, con el objetivo de obtener la comunicación entre nodos con los valores establecidos.

4.2. Implementación nodo Autobús

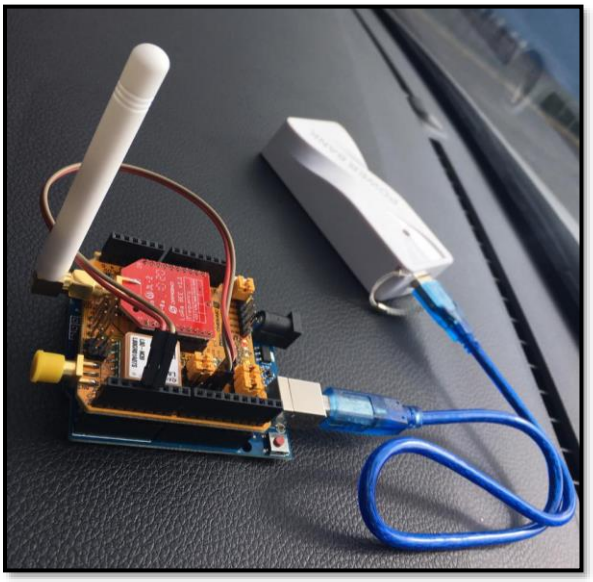

Figura 20. Dispositivo Lora GPS Shield ubicado en vehículo para simulación de piloto

Tal como se indica en la Figura 20, una vez inicializada la comunicación LoRa y asignados los parámetros de configuración, se procede a utilizar la librería **<TinyGPS.h>** que permite extraer los datos GPS en entero flotante, los datos de longitud, latitud y velocidad obtenidos son concatenados para ser transmitidos por canal de radio LoRa al nodo Gateway. Código fuente de este nodo se encuentra en Anexo 1.

4.3. Implementación nodo Gateway

Para esta ejecución es necesario configurar el Gateway siguiendo los pasos que se muestran a continuación:

La conexión con el dispositivo Gateway se lo realiza por medio de LAN por lo que se conecta un cable ethernet para ingresar por browser al quipo y luego se activará la opción de WAN en la configuración para la conexión con los dispositivos LoRa.

El ingreso se lo realiza ubicando la ip: 10.130.1.1 en browser, a continuación, se debe ingresar el **usuario:** root y la **clave:** dragino como se muestra en la Figura 21:

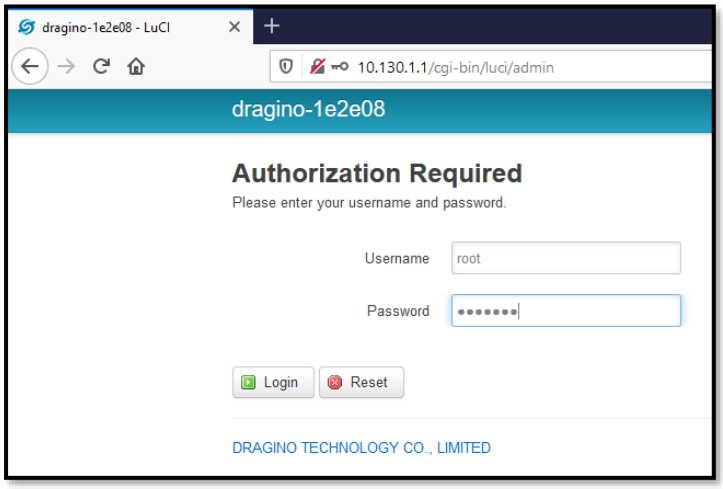

Figura 21. Ingreso a Gateway Lora

El siguiente paso, como se muestra en la Figura 22, es configurar los parámetros de radio en LG01, con el objetivo de recibir la información de radio LoRa de los nodos asociados al Gateway.

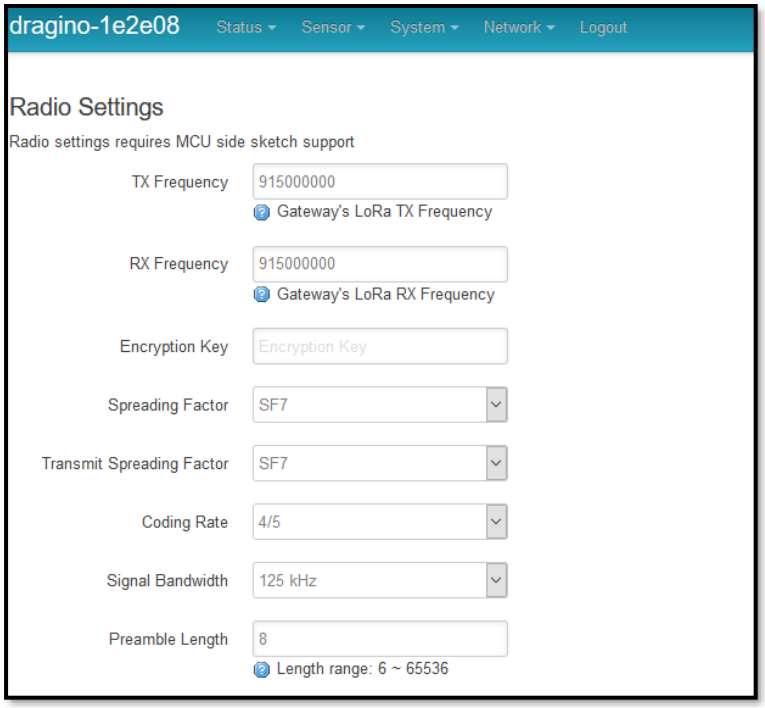

Figura 22. Ingreso de parámetros de configuración en Gateway

Es necesario también elegir el servidor IoT en el que se va a trabajar, como se muestra en la Figura 23.

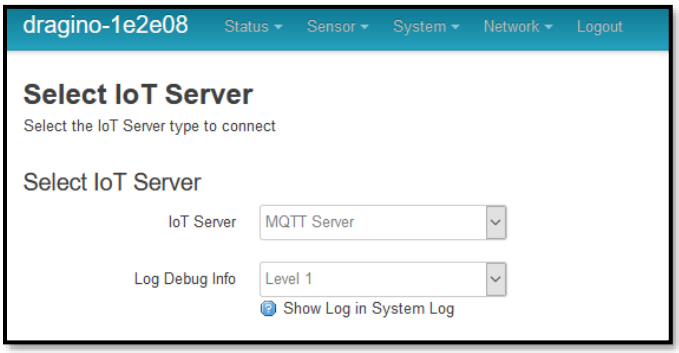

Figura 23. Elección de servidor

A continuación nos dirigimos hacia la parte de Network para elegir el modo de conexión, ubicando en **Access Internet Vía:** WAN Port y en **Way to Get IP :** DHCP, como se muestra en la Figura 24:

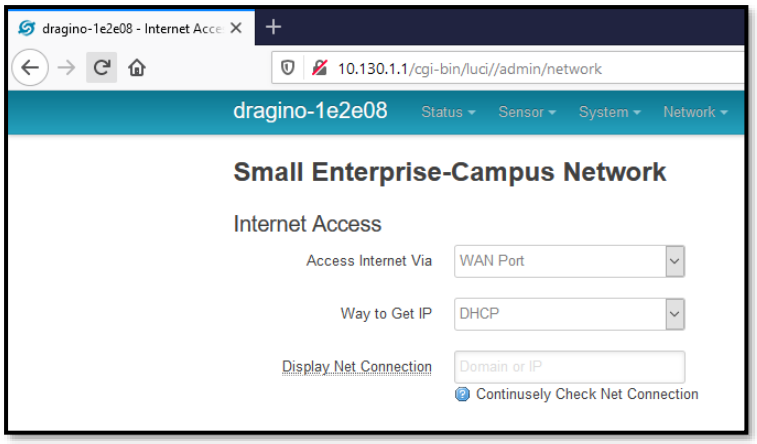

Figura 24. Elección de modo de conexión

4.3.1. Protocolo MQTT

MQTT es un protocolo de comunicación M2M de publicación-subscripción para transferir mensajes entre dispositivos, se basa en el protocolo TCP para la transmisión de datos. Está orientado al envío de datos en aplicaciones donde se requiera utilizra un reducido ancho de banda. Además, sus características le permiten entregar un consumo realmente bajo, así como optimizar recursos para su funcionamiento.

- **Broker MQTT:** se trata de un servidor que maneja todos los mensajes de los clientes y los redirecciona a sus destinos apropiados.
- **Cliente (remitente y receptor):** cualquier dispositivo que ejecute la biblioteca MQTT y se conecte al corredor a través de una red. Los clientes pueden ser editores, suscriptores o ambos.

- **Tópico:** es la información organizada en una jerarquía de tópicos. Los clientes pueden publicar y/o suscribirse a cualquier tema de acuerdo con las reglas de ACL (Lista de control de acceso).

4.3.1.1. Funcionamiento MQTT

El principal componente de la arquitectura MQTT es el bróker, el cual es el punto central al cual se conectan los demás nodos o clientes.

Tal como se aprecia en la Figura 25, el intercambio de mensajes en este protocolo se realiza a través de tópicos jerárquicos que permiten filtrar los mensajes que le corresponden a cada cliente. Es decir, cuando un cliente se subscribe a un tópico determinado podrá recibir mediante un bróker los mensajes que haya publicado otro cliente con el mismo tópico.

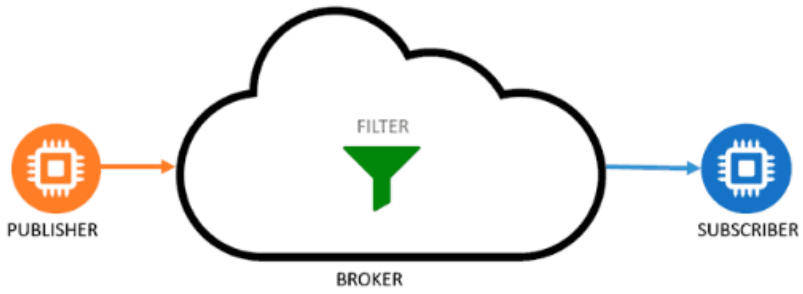

Figura 25. Funcionamiento Protocolo MQTT

Tomado de (Llamas, 2019)

El cliente inicia una conexión TCP/IP con el bróker utilizando el puerto definido por el operador del bróker, el cual conserva un registro de los clientes conectados. Esta conexión se mantiene accesible hasta que el cliente la finaliza. MQTT emplea puertos estándar: puerto 1883 para comunicación no cifrada y el 8883 para comunicación cifrada.

Para esto, el cliente envía un mensaje CONNECT, que contiene información de nombre de usuario, contraseña, client-id, etc., hacia el bróker. En respuesta el bróker envía un mensaje CONNACK al Cliente, que contiene el resultado de respuesta de la conexión.(Llamas, 2019). Figura 26.

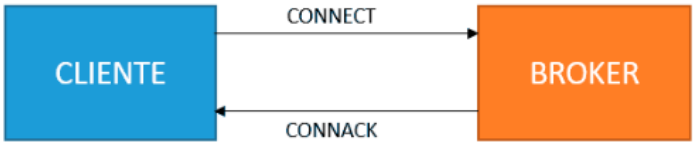

Figura 26. Inicio de conexión TCP/IP con Bróker

Tomado de (Llamas, 2019)

Para remitir los mensajes al bróker, como se indica en la Figura 27, el cliente utiliza mensajes tipo PUBLISH, que contienen el tópico y la carga útil limitada a 256 megabytes.

Figura 27. Envío de mensajes MQTT

Tomado de (Llamas, 2019)

Para que el cliente pueda suscribirse a los tópicos se utilizan paquetes SUBSCRIBE y SUBACK y para la cancelación de la subscripción UNSUSCRIBE/UNSUBACK. Figura 28.

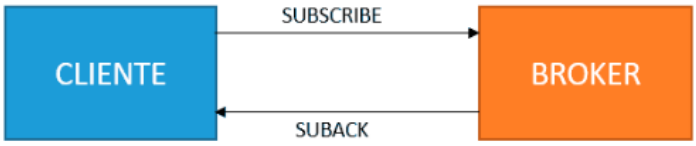

Figura 28. Subscripción para envío de mensajes MQTT

Tomado de (Llamas, 2019)

4.3.1.2. Estructura de un mensaje MQTT

Como podemos visualizar en la Figura 29, los elementos más importantes del protocolo MQTT son: la definición y tipología de los mensajes, donde cada mensaje consta de 3 partes:

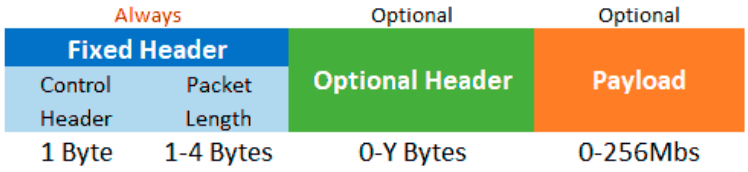

Figura 29´. Estructura de un mensaje MQTT

Tomado de (Llamas, 2019)

- **Cabecera fija:** Utiliza 2 a 5 bytes y es de carácter obligatorio. Se encuentra conformada por un código de control, que se encarga de identificar el tipo de mensaje enviado, y de la longitud del mensaje.
- **Cabecera variable:** es de carácter opcional, contiene información adicional necesaria en ciertos mensajes o situaciones.
- **Carga útil:** Es el contenido real del mensaje. Puede tener un máximo de 256 Mb .(Llamas, 2019)

4.3.1.3. Estructura de red para reenvío de mensajes MQTT

- **Uplink:** El sensor envía datos a LoRa Gateway a través de LoRa Wireless, la puerta de enlace procesará estos datos y reenviará a MQTT Broker remoto a través de Internet.
- **Downlink:** La puerta de enlace suscribir un tema es el intermediario MQTT, cuando haya actualización sobre el tema, la puerta de enlace conocerá y difundirá los datos a la red LoRa local. Figura 30.(Dragino,2020b)

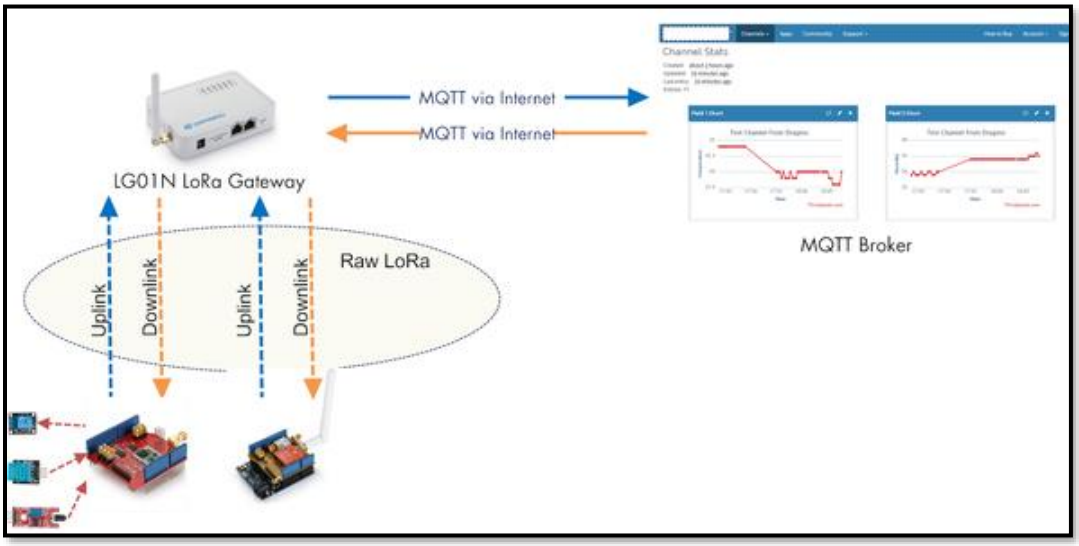

Figura 30. Topología para conexión MQTT

Tomado de (Dragino, 2020b)

4.3.1.4. Configuración conexión MQTT

Para la recepción y envió de mansajes por protocolo MQTT se realiza la siguiente configuración, ingresado en la parte de Sensor>MQTT> MQTT Client en el dispositivo LoRa Gateway:

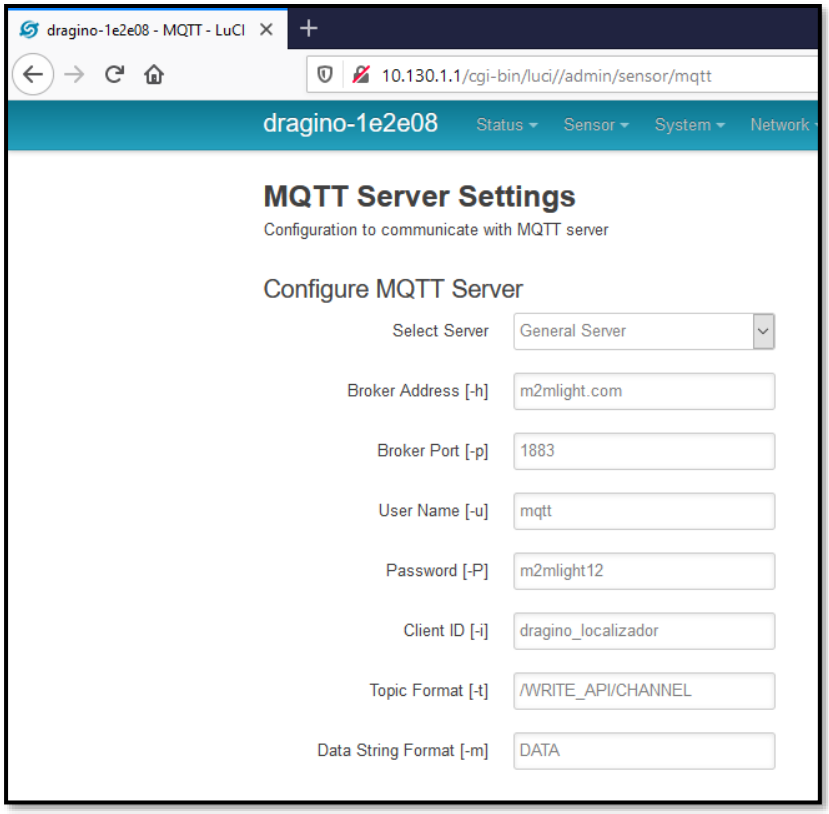

Figura 31. Configuración MQTT Server

En la Figura 31, se definen los parámetros de configuración que permitirán generar una carga ascendente flexible para la publicación MQTT, en donde la dirección del Broker estará dirigida al servidor m2mlight.com por el puerto 1883 y el formato del tópico estará determinado de la siguiente manera: /WRITE_API/CHANNEL, en donde WRITE_API hace referencia al API de escritura del canal remoto y CHANNEL al ID de canal remoto. Además, el formato de cadena de datos será DATA, que contendrá la carga útil de datos del mensaje para la publicación.

4.3.1.5. Definición de Canales MQTT

El Gateway mantiene una tabla de definición de canal que asigna los identificadores de nodo final (ID local) a los identificadores de publicación (ID remota) en la plataforma de IoT del host. La puerta de enlace ejecuta un script llamado 'mqtt_process.sh' que escanea el directorio '/var/iot/canales' buscando archivos nuevos; en este caso, nuestro archivo tendrá este identificador 4536. Cuando es detectado un nuevo archivo en '/var/iot/channels', mqtt_process busca las definiciones de canal para encontrar una entrada que coincida con el nuevo nombre de archivo, extrae el valor de ID remoto correspondiente y lo usa como el nombre del canal. Como se indica en la Figura 32.

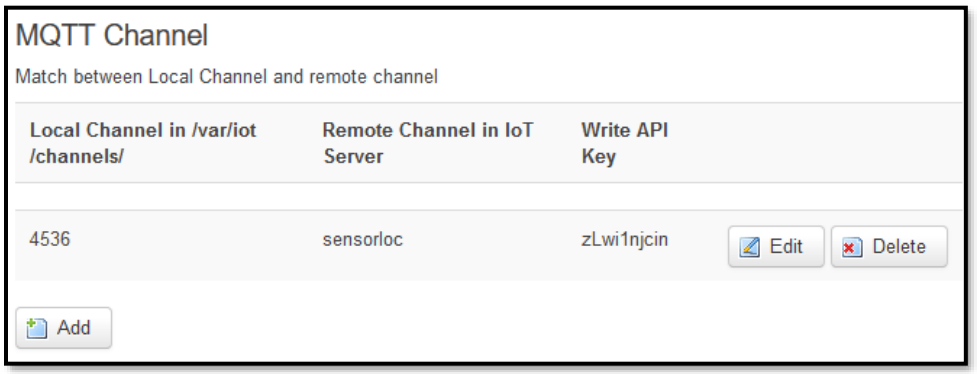

Figura 32. Definición de canales en Gateway

4.3.2. Configuración de Sistema Linux de Gateway

Mosquito es un servidor de mensajes de código abierto que implementa las versiones 3.1 y 3.1.1 del protocolo MQTT, permite una mensajería ligera, utilizando el modelo de publicación/suscripción.

Para la configuración es necesario realizar una sesión en putty con la ip del Gateway: 10.130.1.1 en el puerto 22 y con las credenciales user name: root y passsword: dragino, como se muestra en la Figura 33.

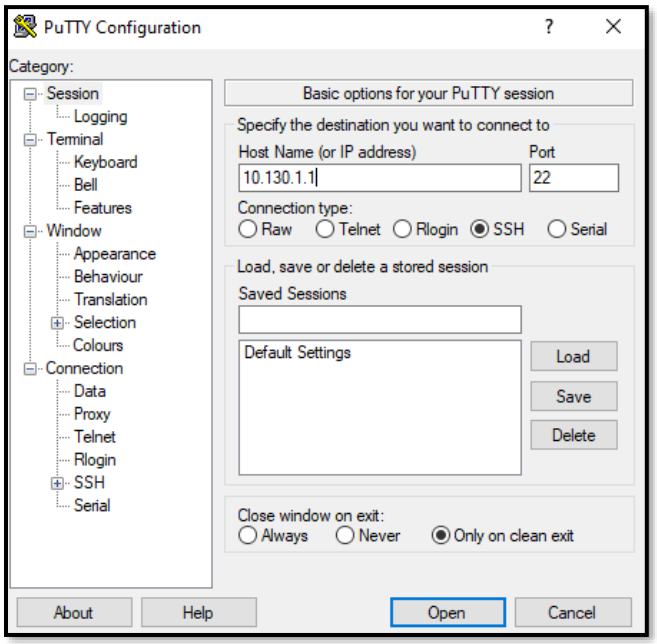

Figura 33. Inicio de sesión con Putty SSH puerto 22

En esta parte de Linux del Gateway se tiene el objetivo de hacer una subscripción; es decir, observar los mensajes depositados en un tópico, será posible con el uso del comando "mosquitto_sub", como se puede apreciar en las Figuras 34. Para comprobar la recepción de los mensajes se utilizará MQTT Explorer, Figura 35.

MQTT Explorer es un cliente MQTT integral que proporciona una descripción general estructurada de sus temas MQTT y hace que trabajar con dispositivos y servicios en su corredor sea muy simple. Entre sus características principales se mencionan las siguientes:

- Visualiza tópicos y actividad temática
- Eliminar tópicos retenidos
- Busca y filtra tópicos
- Eliminar tópicos de forma recursiva
- Diferencia mensajes recibidos actuales y anteriores
- Publica tópicos
- Conserva historial de cada tópico

Figura 34 Configuración en parte Linux de Gateway por SSH

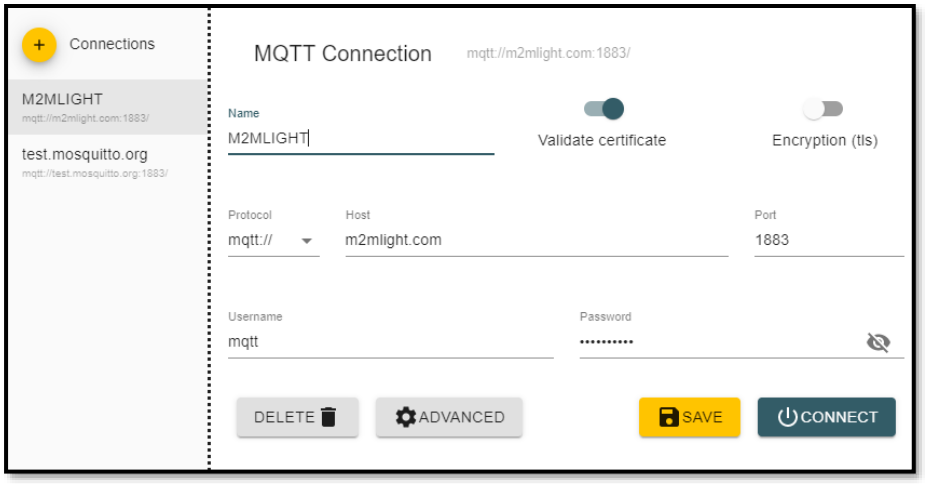

Figura 35 .MQTT Explorer

4.3.3. Envió de mensajes MQTT de Gateway al Servidor M2Mlight

El dispositivo LoRa GPS del nodo Autobús, envía al Gateway por medio de tecnología Lora, un mensaje con la velocidad y ubicación geográfica del bus, cada cierto intervalo de tiempo. Por ejemplo, 6 segundos. El Gateway envía inmediatamente esta información de la posición del Bus, a m2mlight a través de un mensaje MQTT.

El mensaje MQTT está compuesto por:

- **Tópico:** user_api_key/sensorloc
- **Mensaje:** sensor_api_key&latitude&longitude&speed

Donde:

- sensor_api_key: es el identificador o api_key del sensor (GPS del bus)
- latitude: latitud en grados. Ej. -0.174729
- longitude: longitud en grados. Ej. -78.480549
- speed: velocidad en km/h. Ej. 30

Por ejemplo: ZhFr1njcp1&-0.174729&-78.480549&30

4.3.4. Recepción de mensajes de Servidor M2Mlight a Gateway

En el Gateway se crea un proceso para recibir los mensajes de m2mlight. El código fuente de este proceso se encuentra en: /mnt/gps_lora/zLwi1njcin.sh. El mensaje recibido se escribe en un archivo texto, para luego ser leído por un programa escrito en la parte de arduino del Gateway. El código fuente de este programa se encuentra en el mismo Gateway. La programación de esta configuración se encuentra en el anexo 2.

4.4. Implementación de procesos y mensajes en Servidor M2Mlight

Tanto las paradas como los buses se comunican con el Gateway o Pasarela, a través de tecnología LoRa. Y este Gateway se comunica con la plataforma m2mlight vía Internet usando una red Wifi o una red celular. El Gateway se comunica en los dos sentidos con m2mlight a través del protocolo MQTT que permite que los mensajes sean pequeños y controlados. En la Figura 36, se resume los procesos y mensajes entre m2mlight y el Gateway.

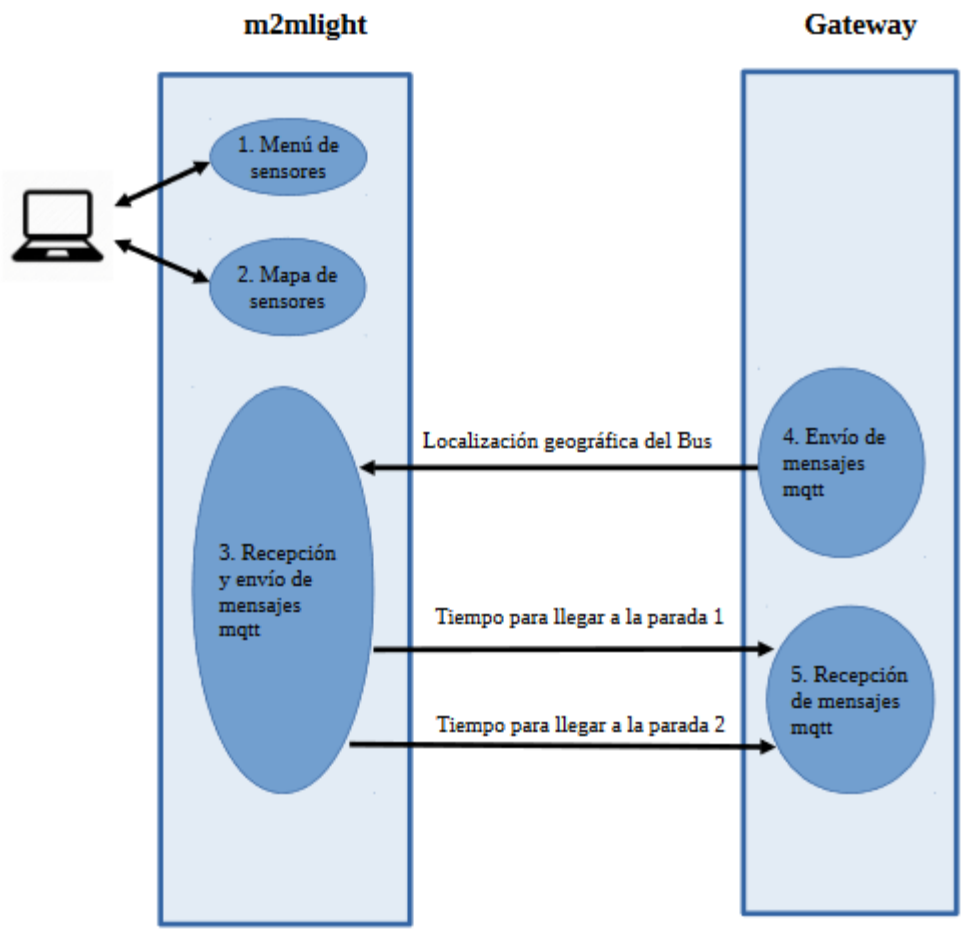

Figura 366. Procesos y mensajes de piloto

Para tener acceso a la plataforma es necesario crear un usuario y contraseña, respetando los siguientes pasos:

Se ingresa por browser a la dirección m2mlight.com y escogemos la opción de register user en donde se registran campos que se muestran a continuación en la Figura 37:

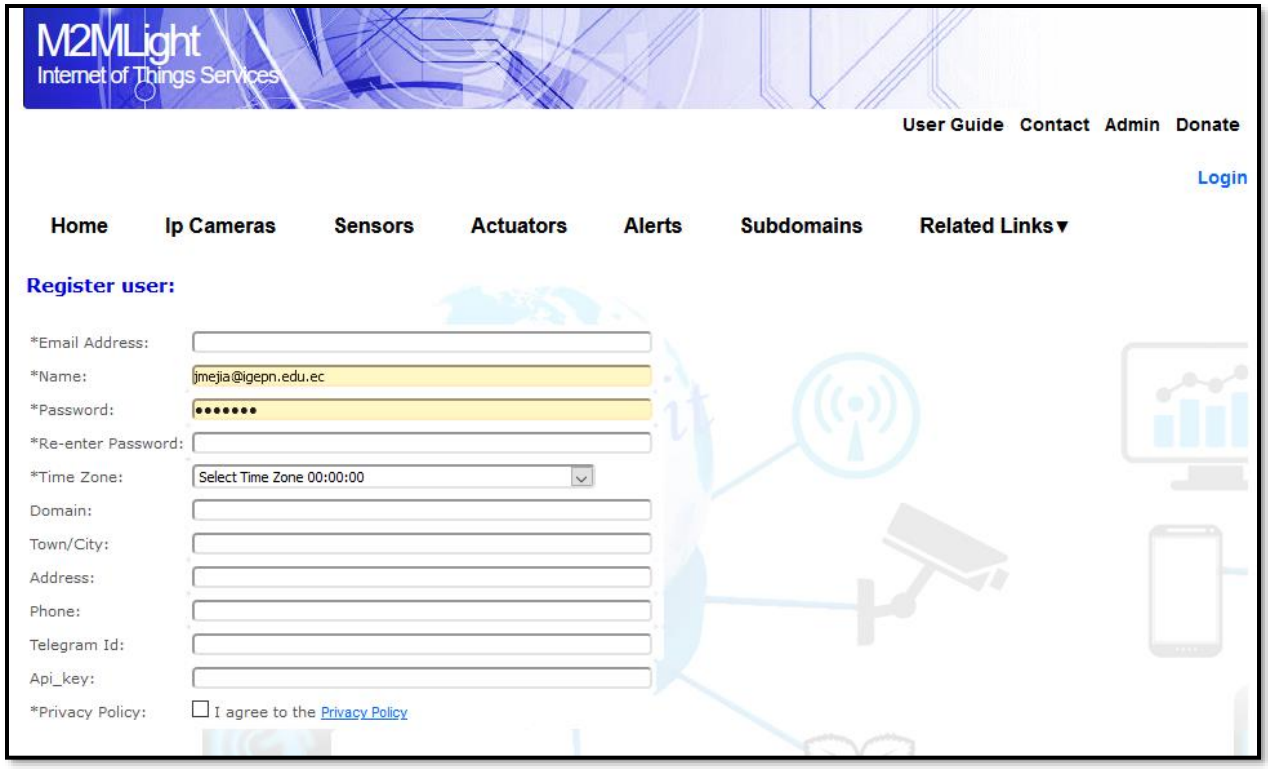

Figura 37 Registro de usuario en Plataforma M2MLight

- **Dirección de correo electrónico:** obligatorio. La dirección de correo electrónico será el identificador de usuario en m2mlight.com. Los mensajes de cámaras, sensores, actuadores y alertas se enviarán a esta dirección.
- **Nombre:** obligatorio. Nombre de usuario.
- **Contraseña:** requerida. Contraseña de al menos 7 caracteres.
- **Zona horaria:** obligatorio. Es importante almacenar los datos con el tiempo apropiado.
- **Dominio:** opcional. Este campo no es obligatorio, pero es necesario si desea utilizar la opción de subdominios.
- **Pueblo / Ciudad, Dirección y Teléfono:** opcional.
- Los otros campos también son opcionales.

4.4.1. Menú de sensores

Una vez ingresado a la plataforma m2mlight, en la opción “Sensors” se definen como sensores el autobús y las 3 paradas de bus. Para las paradas, y debido a que no cuentan con equipo GPS, se ingresa manualmente desde la línea de comandos en linux, su ubicación geográfica con mensajes MQTT, como se puede apreciar en Figura 38. Por ejemplo, para la “PARADA UNO”: #php mqtt_pub.php zLwi1njcin/sensorloc "1vPt1njcp7&-0.173145&-78.480256". El código de esta implementación se detalla en el Anexo 3.

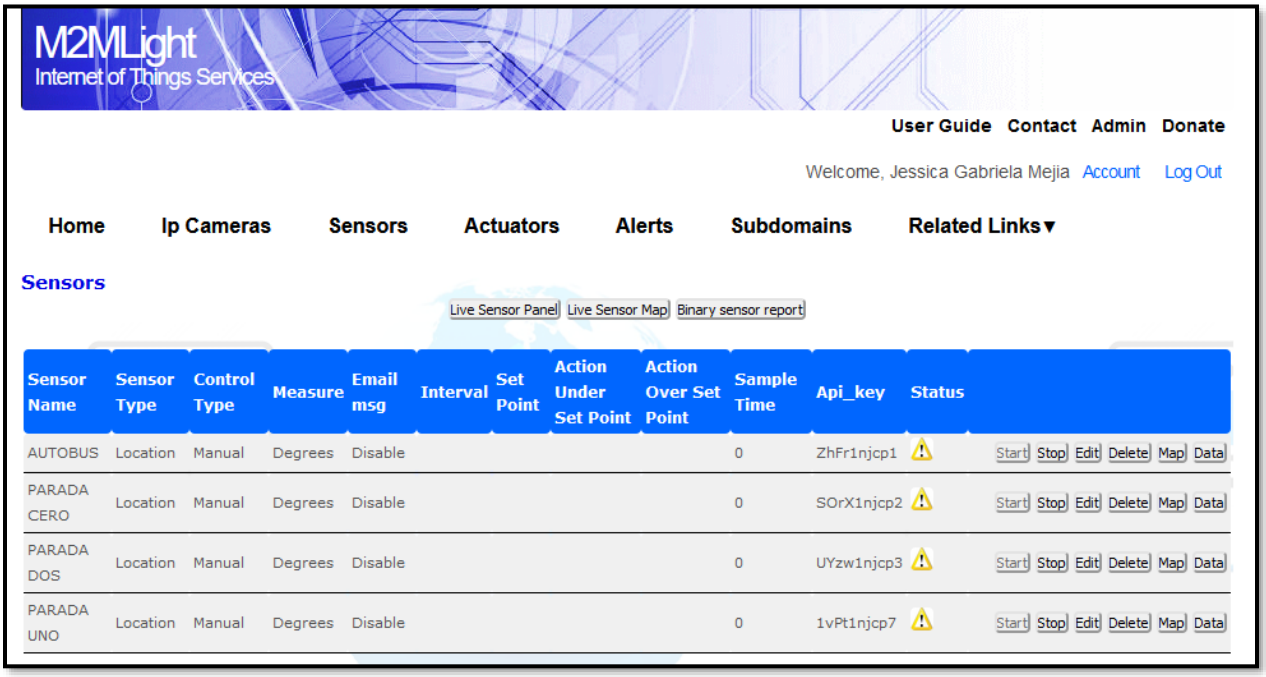

Figura 38 Menú de sensores en plataforma M2Mlight

4.5. Implementación nodo Parada

En la implementación de este nodo, como se puede apreciar en la Figura 39, los dispositivos de radio LoRa más Arduino son ensamblados y configurados en IDE Arduino para conseguir la recepción de datos. Configuración se muestra en anexo 4.

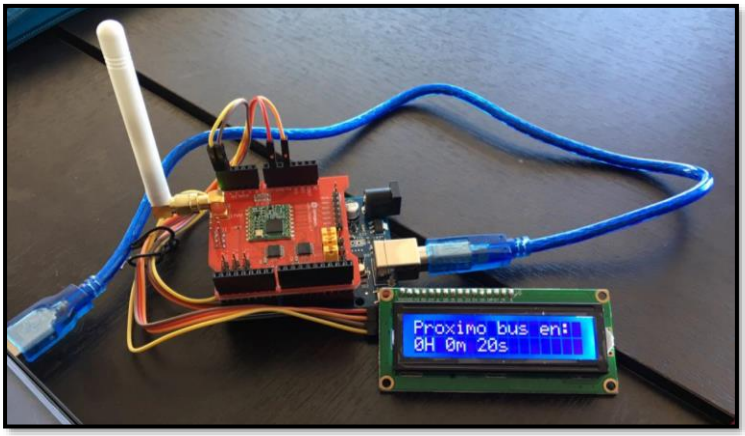

Figura 39 Implementación nodo parada

Por tanto, la función principal de este nodo es recibir la información de tiempo de llegada y si paso o no el autobús en las 2 paradas de 3 establecidas; solo dos paradas cuentan con una pantalla de LCD ya que una de ellas solo se toma como punto de referencia para inicio de la ruta.

Cabe mencionar que el piloto despliega la información de posición y velocidad en tiempo real en la plataforma M2Mlight, facilitando la visualización a través de mapas como se muestra en la Figura 40.

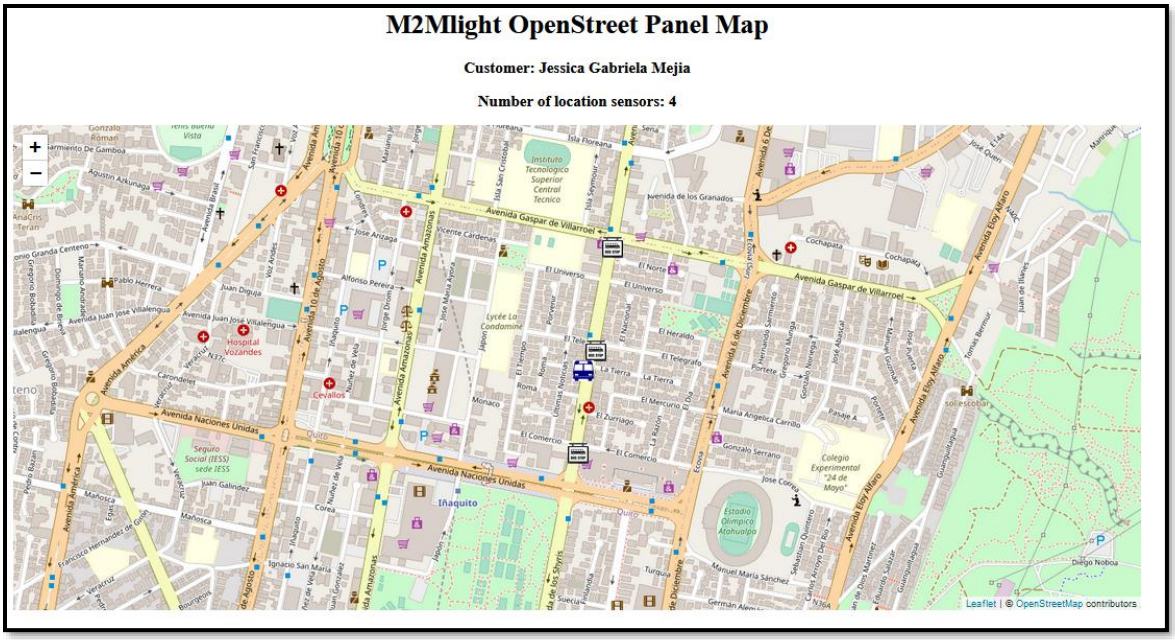

Figura 40 Panel Map de servidor M2Mlight

4.6. Pruebas de implementación

Para realizar las pruebas, se fija una ruta de bus existente en la ciudad de Quito, se toman 3 paradas de la ruta de buses que van por la Av. de los Shyris. Se inicia en la parada de los Shyris y El Comercio hasta la parada de la Shyris y Gaspar de Villarroel, como se puede apreciar en la Figura 41.

Figura 41 Puntos de prueba para piloto

Como se aprecia en las Figuras 42 y 43, una vez fijados los puntos de prueba se procede a ubicar el dispositivo LoRa Shield GPS en el automóvil para iniciar con el recorrido. Para esto ya se tienen conectados los 3 nodos parada, el nodo Gateway y abierta la plataforma para visualizar en tiempo real la ubicación y velocidad de nodo autobús y el cliente MQTT Explorer para detectar los mensajes que llegan.

Figura 42 Escenario de nodos parada en plataforma M2MLight

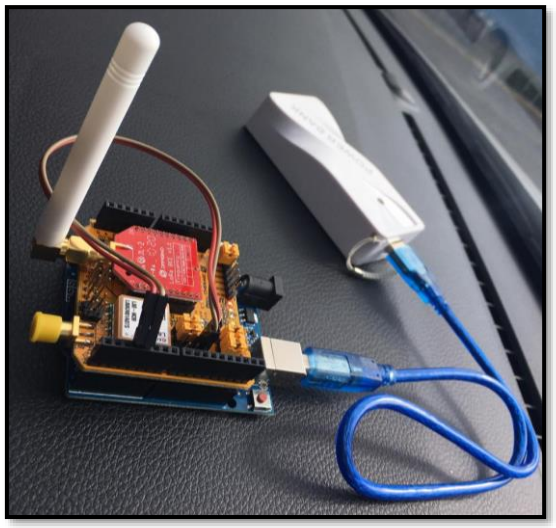

Figura 43 Instalación de LoRa GPS Shield en automóvil

A fin de conseguir una buena comunicación entre nodos es necesario ubicar el nodo Gateway en un buen sitio para que logre su mejor desempeño, por lo cual el dispositivo es ubicado en el

Edificio Rubio, 9no piso, en donde la antena del dispositivo, es instalada en una de las ventanas del piso y el Gateway en una mesa conectado a la red. Cabe mencionar que las antenas de los dispositivos son de 915 MHz y 3dBi de ganancia.

Para iniciar con las pruebas también se prepararán dos computadores, con los que se visualizarán los mensajes recibidos en cliente MQTT y la ubicación y velocidad en tiempo real en la plataforma M2MLight. Por la emergencia sanitaria COVID-19, por seguridad, los dispositivos LoRa Shield de los nodos parada, no fueron trasladados a las paradas, pero fueron visualizados en la sala de monitoreo del piloto.

El vehículo empieza a movilizarse desde la parada 0 a la parada 1 y de la parada 1 a la parada final 2 y en su trayecto se puede verificar su velocidad y posición. Al pasar por cada nodo parada se visualiza por medio de LCD el tiempo de arribo a cada parada y la información de que, si pasó o no, como se puede observar detalladamente en las Figuras 44, 45, 46, 47, 48, 49, 50 y 51.

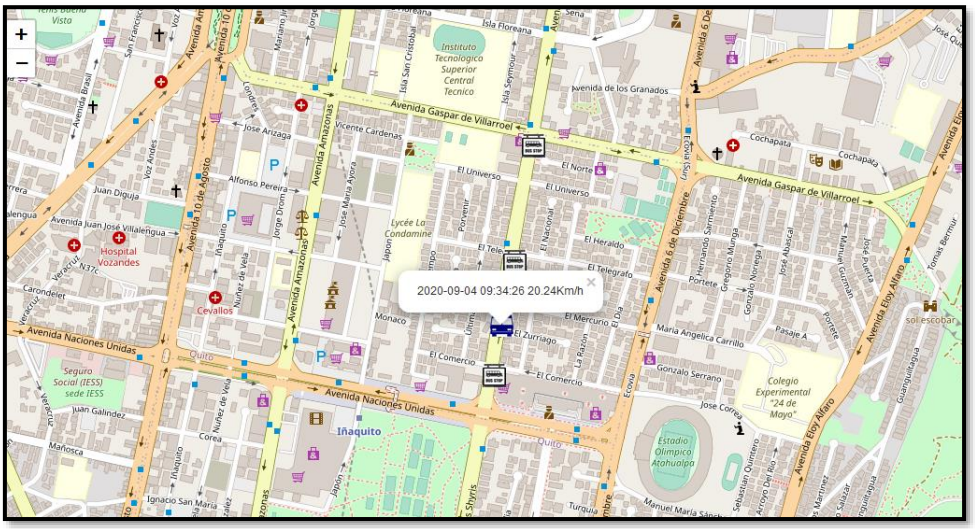

Figura 44 Transcurso de autobús de Parada 0 a Parada 1, 24Km/h

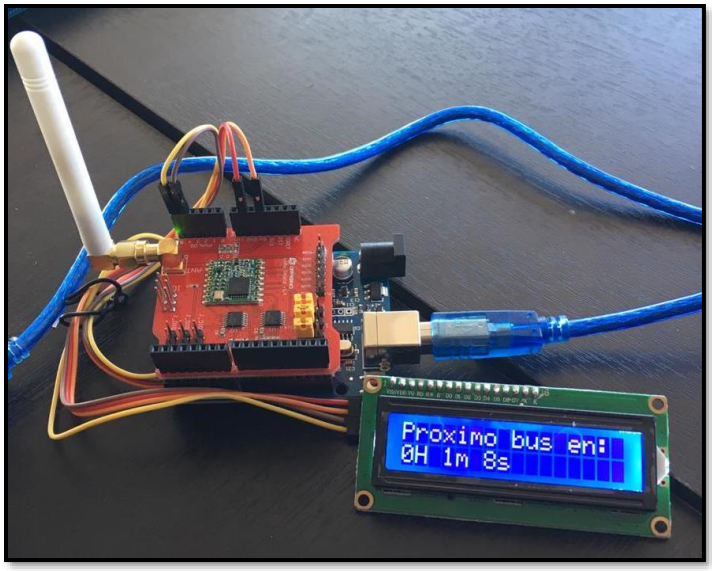

Figura 45 Tiempo de arribo de parada 0 a parada 1

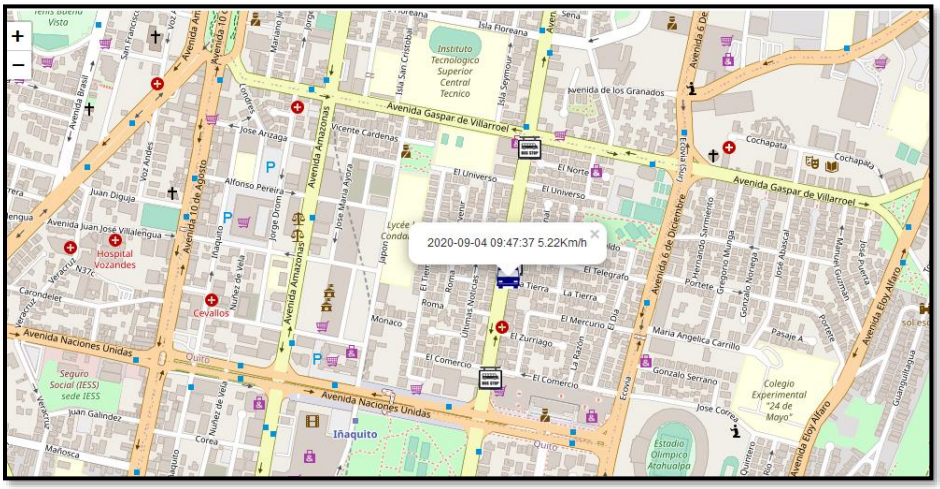

Figura 46 Arribo próximo de autobús a parada 1, 5,22 Km/h

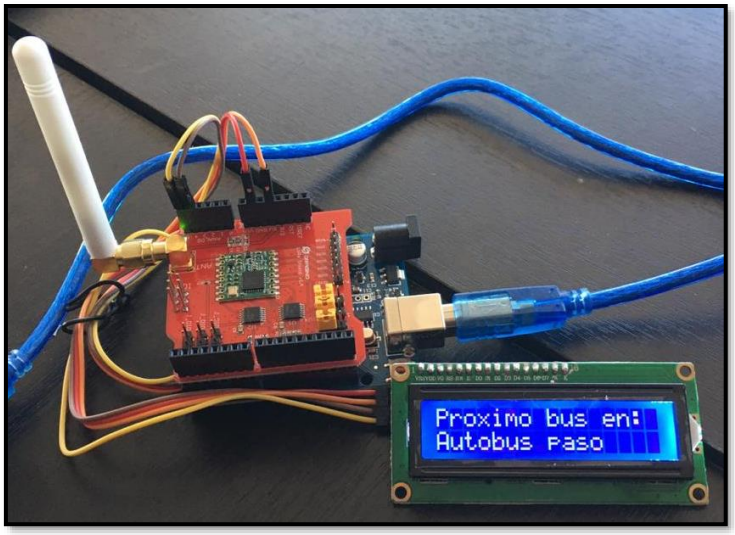

Figura 47. Autobús paso de parada 1 y continua a parada 2

Figura 48. Autobús llegará próximamente a Parada Final 2, 2,2Km/h

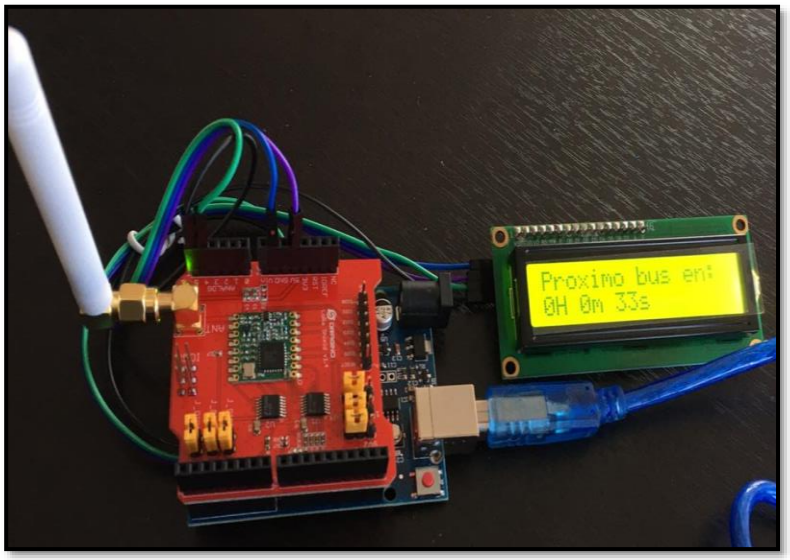

Figura 49. Autobús llegará a la parada 2 en un tiempo de 33s

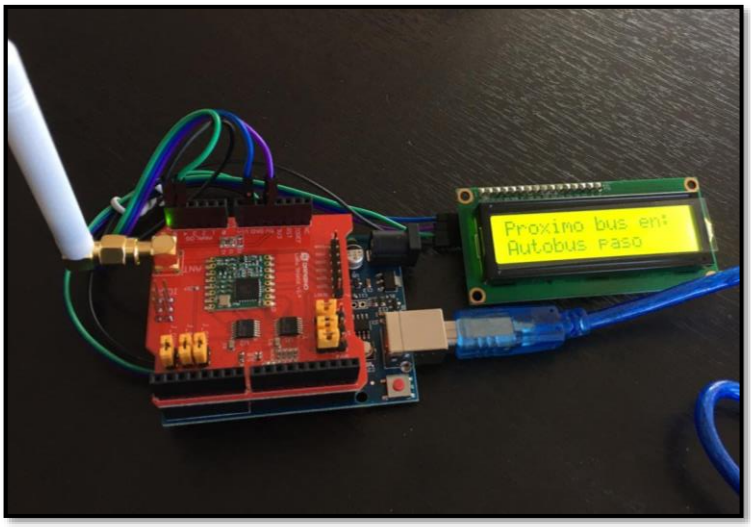

Figura 50. Autobús pasa de parada 2

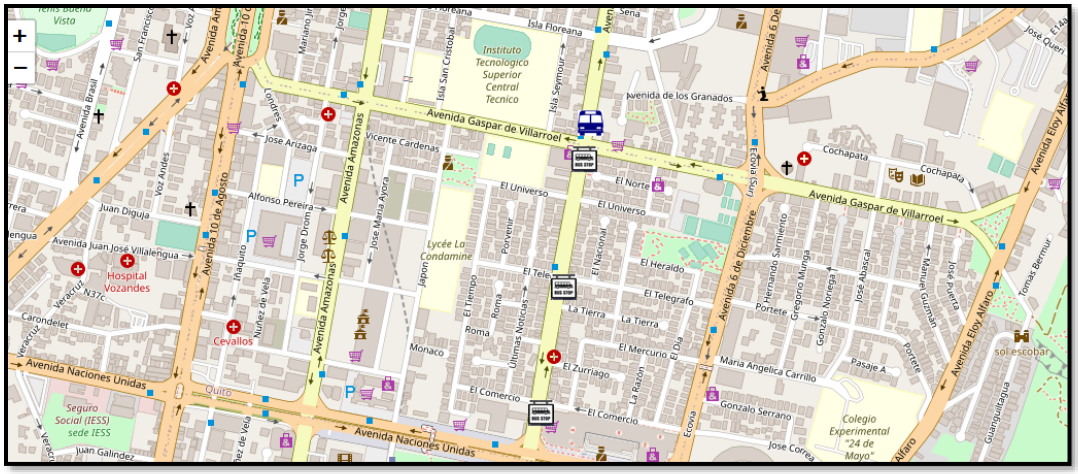

Figura 51. Autobús finaliza la ruta fijada para el piloto

A continuación, se muestra en Figuras 52,53 y 54 la recepción de mensajes en MQTT Explorer.

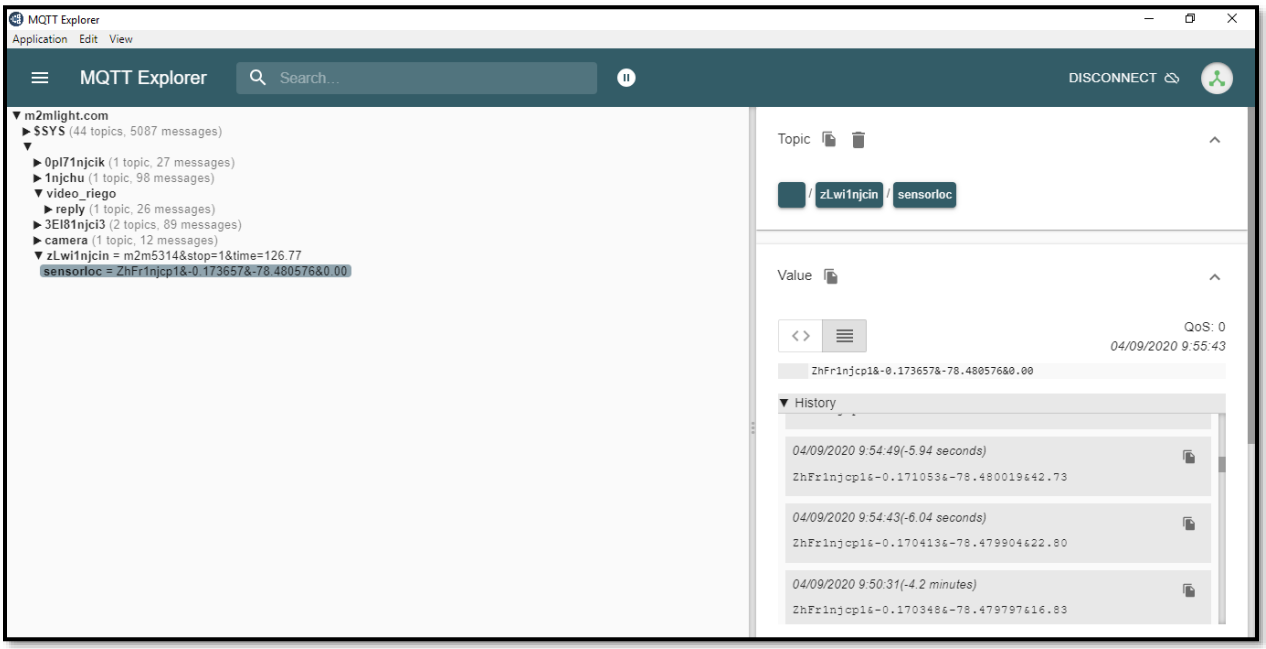

Figura 52. Recibe información de posición de dispositivo LoRa GPS Shield

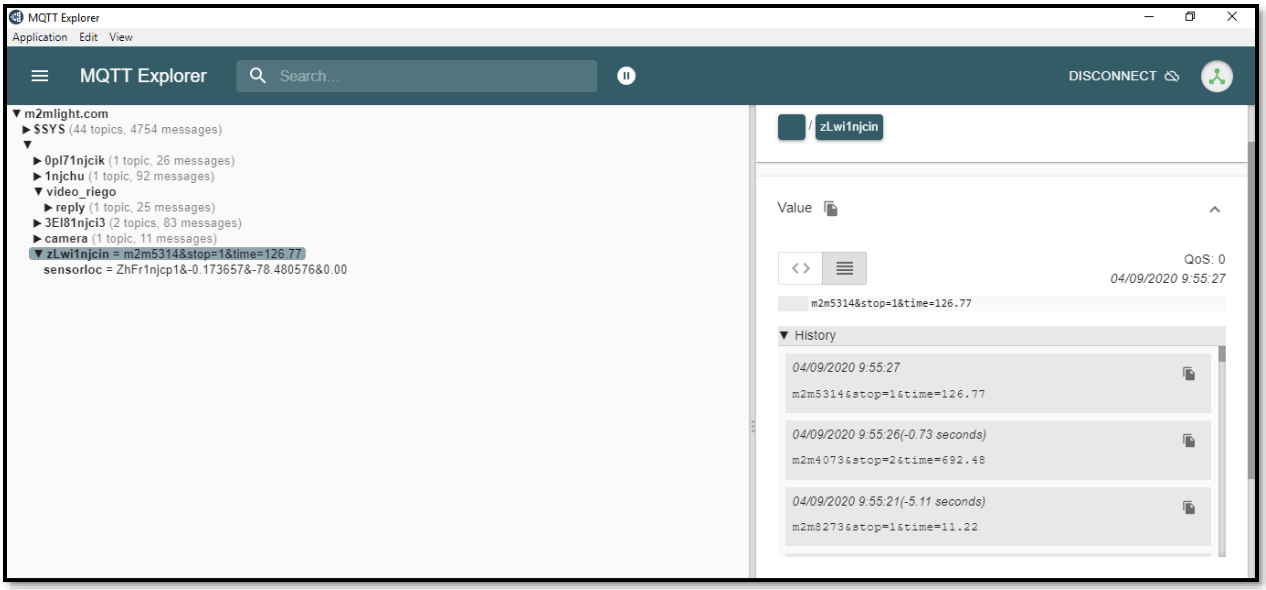

Figura 53. Muestra mensajes del tiempo de arribo a las paradas

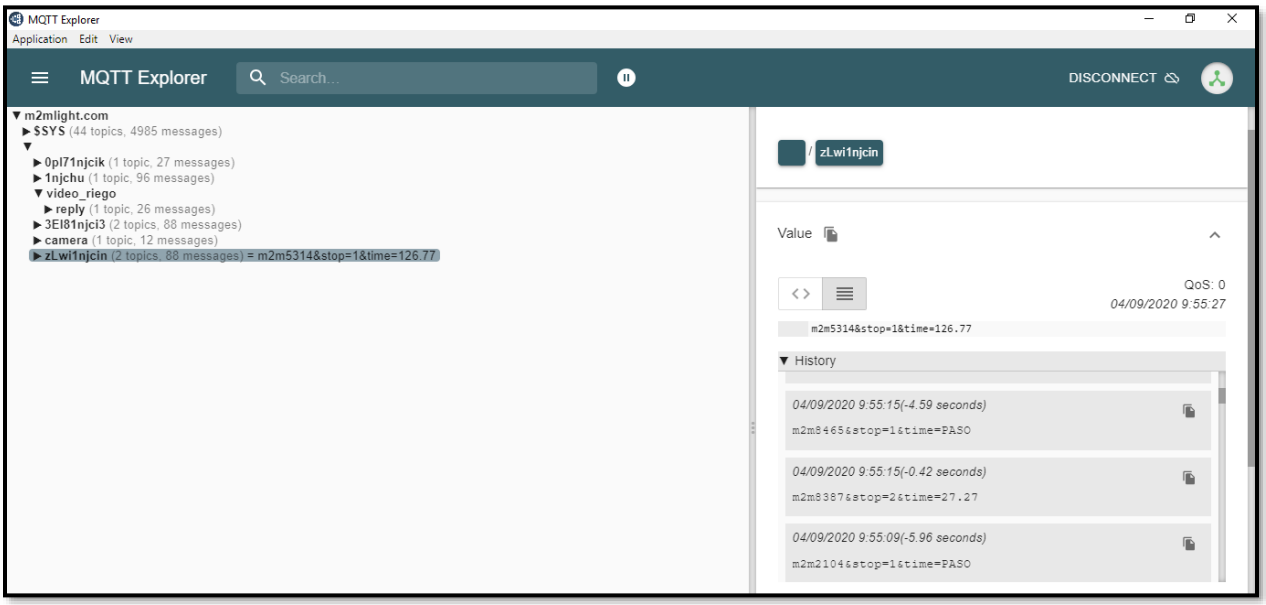

Figura 54. Muestra mensajes de Paso de autobús de las paradas

4.7. Análisis de Resultados y Evaluación Final

IoT ha causado gran revolución tecnológica a nivel mundial, debido a que posibilita la creación de sistemas inteligentes para diferentes aplicaciones; por lo cual, en principio se analizan tecnologías de comunicación para IoT como son: NB-IoT, LoRa y Sigfox con la finalidad de encontrar una mejor solución tecnológica. La tecnología LoRa fue escogida en esta ocasión debido a que NB-IoT, a pesar de que presenta mejores características que LoRa, utiliza espectro licenciado y no está ampliamente disponible en el mercado. Sigfox, en cambio, tiene un presupuesto de enlace restringido y no se ha desplegado aún en redes significativas. En este contexto, está claro que utilizar tecnología LoRa en este proyecto, permite sustituir las soluciones de comunicación actual, generando un aporte significativo a la mejora del sistema de transporte inteligente. De esta manera se consiguió obtener una eficiencia reveladora relacionada con el tiempo de espera de los usuarios en las paradas, obteniendo un promedio considerable de 5 minutos; a diferencia de lo que ocurre hoy en día que se tiene que esperar en un rango de tiempo de 20 a 30 minutos sin tener conocimiento de arribo y ubicación de los autobuses.

En la actualidad, la ciudad de Quito cuenta con un sistema de transporte público, que a pesar del tiempo transcurrido y los diferentes paliativos insertados no ha logrado solucionar los problemas de movilidad y mejorar la calidad en el servicio al usuario; sobre todo los aspectos relacionados a los largos tiempos de espera en las paradas, inconveniente que no ha sido considerado aún en las aplicaciones tecnológicas sugeridas en las propuestas de mejoras establecidas en los diferentes proyectos desarrollados para el sistema de transporte; por tanto, evaluando esta necesidad, he desarrollado una aplicación tecnológica basada en un proyecto de titulación de la Universidad del Cauca, Popayán, que tiene la capacidad de presentar la información del tiempo de llegada en las paradas de unidades de transporte; además de poder, visualizar en una

plataforma, la posición y velocidad de los autobuses, para información y beneficio del usuario, por ejemplo:

Si una persona esperaba la unidad de autobús en una parada, sin un tiempo determinado; con esta aplicación, a más de saber el tiempo exacto de llegada de la unidad, podrá conocer la posición y velocidad en la que ésta se está desplazando.

Al realizar las pruebas de campo en este piloto, se pudo evidenciar que la tecnología LoRa alcanza gran cobertura y alta tolerancia a interferencias, permitiendo sobrepasar el área de cobertura establecida, que en este caso fue reducida por motivos de pandemia, pero con gran presencia de edificios que no fueron impedimento para obtener respuesta de los nodos hasta una distancia significativa, garantizando la transmisión ininterrumpida de la información al usuario. Cabe mencionar que para obtener estos resultados fue muy importante la ubicación de la antena del Gateway, haciendo posible que todos los puntos transitados por el vehículo sean visibles e hizo posible la lectura de datos de velocidad y posición, lo que permitió dar respuesta al usuario tanto en las paradas como en la plataforma.

Los dispositivos LoRa GPS Shield, LoRa Shield y Gateway, utilizados en este proyecto de titulación, permitieron encontrar la mejor configuración para obtener una amplia distancia de transmisión sin comprometer la veracidad de los datos; es decir, que los datos recibidos y transmitidos fueron adquiridos de forma correcta sin afectar su consumo energético, entregando para la comunidad un sistema significativamente sostenible.

Para comprobar el alcance y funcionamiento de la red LoRa implementada, se realiza una simulación en el software Radio Mobile de los enlaces desde el Nodo Gateway hacia cada nodo de las paradas en donde llega el autobús, como se muestra en las figuras 55 y 56:

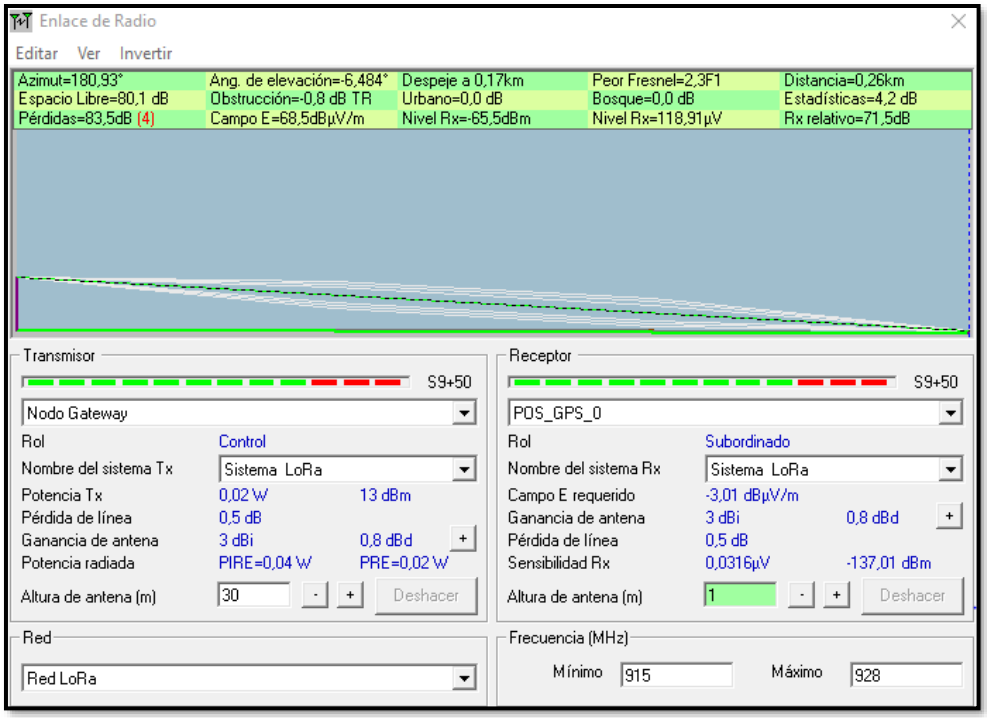

Figura 55. Enlace de Radio de Nodo Gateway hacia Parada inicial

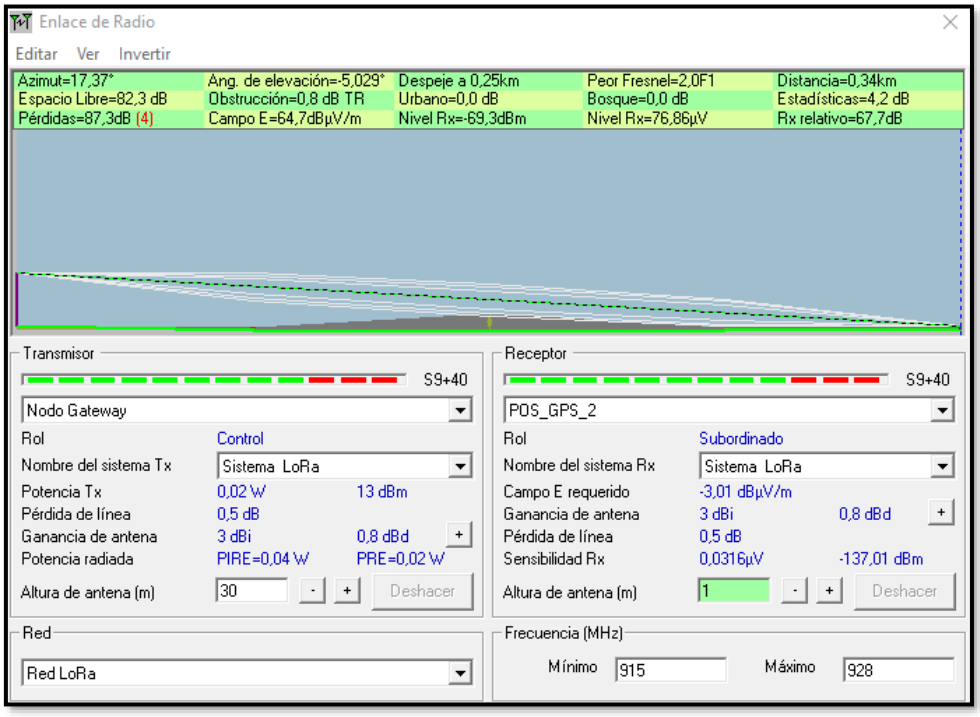

Figura 56. Enlace de Radio de Nodo Gateway hacia Parada Final

Las simulaciones de los puntos de enlace muestran niveles de recepción óptimos, esto quiere decir menor a -80dBm. Se puede observar también que existe suficiente despeje en la primera zona de Fresnel con un valor de 1,5F1, indicando que no hay ninguna obstrucción en la línea de vista, lo que permite que los dispositivos se comuniquen sin ningún inconveniente con una potencia de transmisión de 13dBm.

Utilizando tecnología LoRa fue posible evaluar la disponibilidad de acceso a la información en tiempo real y de manera confiable, además de evaluar la integridad de los datos recibidos en cada una de las paradas y en la plataforma, con lo que se obtuvo buenos niveles de cobertura en las condiciones establecidas, como se aprecia en la figura 57. Cabe mencionar que el análisis de cobertura fue realizado con el alcance máximo de 20Km, pero se logró una comunicación sin perdida hasta los 6km.

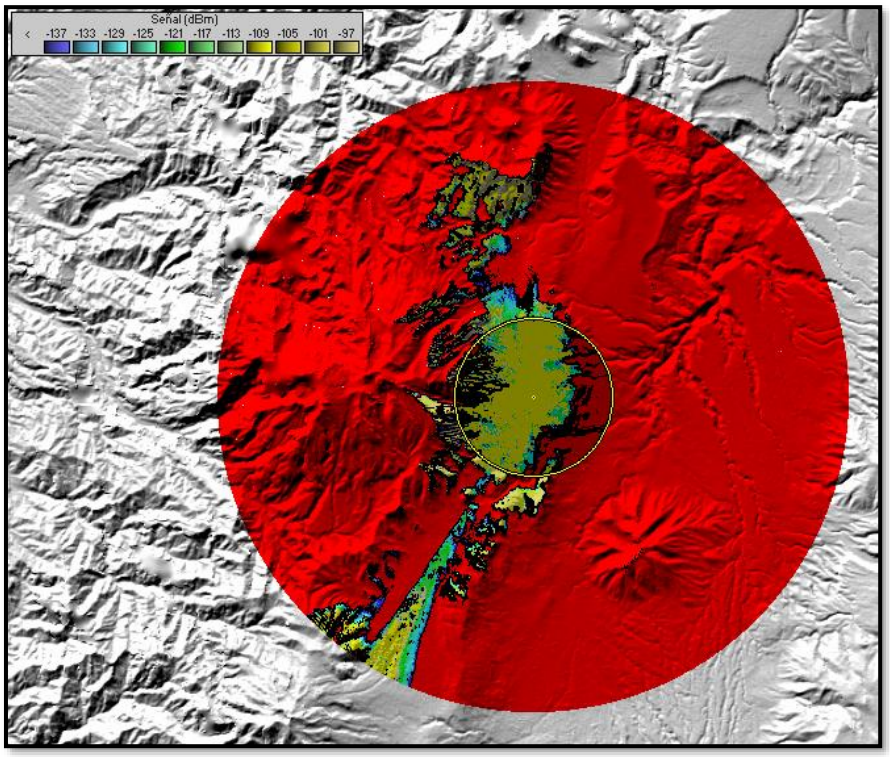

Figura 57. Niveles de cobertura alcanzada por tecnología LoRa, evaluado en 20km de alcance máximo

Por lo expuesto, se puede establecer que la tecnología LoRa es una solución que permite optimizar recursos públicos, tomando en cuenta el crecimiento futuro de la red, además el consumo de energía de los dispositivos es bajo, lo que permitiría llevar a la ciudad de Quito a un modelo de negocio colaborativo; esto quiere decir, que podemos ir más allá de los niveles de innovación establecidos actualmente, creando interconexión de miles de dispositivos, o bien desarrollando nuevas propuestas de soluciones para aprovechar la infraestructura disponible, sin perjudicar a los servicios que actualmente se ofrecen y aportando de esta manera significativamente al progreso tecnológico del sistema de transporte inteligente en la ciudad de Quito.

CONCLUSIONES

Actualmente la ciudad de Quito cuenta con un transporte constituido por un sistema integrado de baja calidad en el servicio al usuario. Por tal motivo, y a fin de ayudar a solventar estos inconvenientes, se decide utilizar una solución IoT con tecnología LoRa como un modelo que pretende alcanzar un impacto en el desarrollo tecnológico y mejorar la calidad de vida de los habitantes.

En este trabajo final de titulación, se diseñó, implemento y evaluó la tecnología LoRa con el propósito de obtener mejoras en la calidad de servicio de transporte público de la ciudad de Quito. Lo cual, permite obtener una solución de información al usuario que optimice y ayude a reducir su tiempo al momento de desplazarse de un lugar a otro.

Los nodos y gateway escogidos para esta solución son capaces de crear una red de bajo coste y amplia cobertura, siendo más efectiva en zonas donde no existió obstrucción en la línea de vista. Los dispositivos se conectaron en topología estrella, consiguiendo interactuar de forma individual y directa entre ellos.

Una vez realizada la evaluación de la implementación de la tecnología LoRa en este proyecto, se verificó que se cumplió con las perspectivas de servicio, operación, tecnología y economía, debido a que fue posible mejorar significativamente el servicio de transporte y responder efectivamente a las necesidades del usuario.

RECOMENDACIONES

IoT se integra cada vez más a la vida diaria de las personas, por lo cual, es importante que se sigan desarrollando ideas tecnológicas que permitan mejorar su calidad de vida.

Se recomienda a futuro implementar una aplicación Android, con las mismas prestaciones que tiene la plataforma M2MLight, con la finalidad de ofrecer comodidad al usuario al momento de explorar la información requerida.

Para evitar inconvenientes en la comunicación entre los nodos, es recomendable ubicar los dispositivos Gateway en lugares altos cuando se trabaja en zonas urbanas.

BIBLIOGRAFÍA

330ohms. (2018). *Controlador I2C para LCD 16x02 y 20x04*. 330ohms. https://www.330ohms.com/products/interfaz-i2c-para-lcd-16x2

Agencia de Ecología Urbana de Barcelona. (2017). Reestructuración de la Red de Transporte Público de Pasajeros del Distrito Metropolitano de Quito. In *DMQ*. http://www7.quito.gob.ec/mdmq_ordenanzas/Comisiones del Concejo/Movilidad/2017/Presentaciones/2017-09-13/PRESENTACIÓN Sept2017 - PARTE I & II- SITP DMQ.pdf

Albuja Sáenz, M. F. (2020). *Internet of Things Platform, Home Automation - M2mlight*. http://m2mlight.com/

AliExpress. (2020). *De larga distancia inalámbrico 433_868_915 Mhz Lora Shield para Arduino Leonardo, UNO, Mega2560, Duemilanove, debido_shielded smd power inductor_shield suppliesshield fabric - AliExpress*. https://es.aliexpress.com/item/32591527766.html

Amazon. (2020a). *Dragino Lora GPS Shield 915Mhz*. Amazon. https://www.amazon.com/-/es/Dragino-compatible-Arduino-Leonardo-consumo/dp/B07HD1MH3J/ref=sr_1_1?__mk_es_US=ÅMÅŽÕÑ&dchild=1&keywords=lora+dragino+LoRa+Shield+for+915+mhz&qid=1602122247&sr=8-1

Amazon. (2020b). *Dragino LoRa IoT Kit de desarrollo V2 915MHZ*. Amazon. https://www.amazon.com/-/es/Dragino-desarrollo-Gateway-sistemas-alcance/dp/B07WMFFHPG/ref=sr_1_1?__mk_es_US=ÅMÅŽÕÑ&dchild=1&keywords=lora+dragino+kit+for+915+mhz&qid=1602121284&sr=8-1

Arduino. (2015). *Arduino - SoftwareSerial*. Arduino. http://arduino.cc/en/Reference/SoftwareSerial

Arduino. (2020a). *Arduino - Process*. Arduino. https://www.arduino.cc/en/Tutorial/Process

Arduino. (2020b). *Arduino - YunBridgeLibrary*. Arduino. https://www.arduino.cc/en/Reference/YunBridgeLibrary

Arduino. (2020c). *Arduino - YunFileIOConstructor*. Arduino. https://www.arduino.cc/en/Reference/YunFileIOConstructor

Ballesta Viñas, J. (2018). Evaluación de tecnologías LPWAN para escenarios de Smart Cities. *Universidad Politécnica de Cartajena*, 1–36. http://repositorio.upct.es/handle/10317/7296

Barrio Andrés, M. (2018). *Internet de las Cosas* (REUS S.A(2). https://books.google.com.ec/books?id=jF-LDwAAQBAJ&printsec=frontcover&dq=iot+que+es&hl=es&sa=X&ved=0ahUKEwivltOYlsvoAhWtnOAKHWAHAncQ6AEIOTAC#v=onepage&q=iot que es&f=false

Cárdenas, M., Gonzáles, D., & Retamal, C. (2018). *Protocolo LoRa para implementación IoT en Smart Cities*. 1–5. http://profesores.elo.utfsm.cl/~agv/elo323.ipd438/2s18/projects/reports/RetamalCardenasGonzalez/Informe_Lora.pdf

Casanova, A. (2017). *Seguridad en redes LoraWAN (Parte I)*. https://digimodes.wordpress.com/2017/02/18/seguridad-en-redes-lorawan-parte-i/

Casanova González, M. Á. (2020). *El protocolo LoRaWAN*. AlfaIOT. https://alfaiot.com//en_US/blog/ultimas-noticias-2/post/el-protocolo-lorawan-6

Céspedes Machicao, M. (2017). Características de las placas arduino. *Universidad Autónoma Juan Misael Saracho, (1ª ed.)*, 2–7. http://www.uajms.edu.bo/revistas/wp-content/uploads/2017/12/Art1-bit@bitdic2017.pdf

Dopazo González, L. (2019). *Test y despliegue de tecnología de comunicaciones LoRa para*

aplicaciones de Internet of Things [Universidad Politécnica de Madrid]. http://oa.upm.es/54465/1/TFG_LAURA_DOPAZO_GONZALEZ.pdf

Doxigen. (2017). *RadioHead: RH_RF95 Class Reference*. https://www.airspayce.com/mikem/arduino/RadioHead/classRH__RF95.html

Dragino. (2020a). *Lora_GPS Shield - Wiki for Dragino Project*. Wiki Dragino. http://wiki.dragino.com/index.php?title=Lora/GPS_Shield#Pin_Definition

Dragino. (2020b). *MQTT Forward Instruction - Wiki for Dragino Project*. Wiki Dragino. http://wiki.dragino.com/index.php?title=MQTT_Forward_Instruction

Dragino Technology Co., L. (2019). *LG01-P IoT Gateway featuring LoRa® technology*.

Ecuador, Tec. (2020). *Arduino UNO y LCD 1602+I2C*. TECmikro Ecuador. https://tecmikro.com/

Ezell, S. (2010). Intelligent transportation systems. *ITIF*, *5*(4), 58. https://doi.org/10.4018/978-1-5225-5643-5.ch076

GSD+. (2018a). *Esquema de implantación de tecnologías inteligentes de transporte en América Latina: estado actual y avances en el ámbito urbano* (CAF (ed.)). http://scioteca.caf.com/handle/123456789/1395

GSD+. (2018b). *Esquemas de implantación de tecnologías inteligentes de transporte en América Latina: estado actual y avances en el ámbito urbano*.

Guerrero, J. (2014). *Arduino Uno: Especificaciones y características | PlusElectric*. PlusElectric. https://pluselectric.wordpress.com/2014/09/21/arduino-uno-especificaciones-y-caracteristicas/

Ingenima. (2020). *Qué es la geolocalización y cómo funciona*. EvaluandoSoftware.Com. https://www.evaluandosoftware.com/la-geolocalizacion-funciona/

jlquijado. (2016a). *La Librería LiquidCrystal*. HTTP Masters.

https://eldesvandejose.com/2016/04/02/la-libreria-liquidcrystal/

jlquijado. (2016b). *La Librería SPI*. HTTP Masters. https://eldesvandejose.com/2016/05/10/la-libreria-spi/

Llamas, L. (2019). *¿Qué es MQTT_ Su importancia como protocolo IoT*. https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

Moya Quimbita, M. A. (2018). *Evaluación de pasarela LoRa / LoRaWAN en entornos urbanos* [Universidad Politécnica de Valencia]. https://riunet.upv.es/bitstream/handle/10251/109791/Moya - Evaluación de pasarela LoRa/LoRaWAN en entornos urbanos.pdf?sequence=1&isAllowed=y

Naylamp Mechatronics. (2016). *Tutorial LCD, conectando tu arduino a un LCD1602 y LCD2004*. https://www.naylampmechatronics.com/blog/34_Tutorial-LCD-conectando-tu-arduino-a-un-LCD1.html

Ortiz Monet, M. (2019). *IMPLEMENTACIÓN Y EVALUACIÓN DE PLATAFORMAS EN LA NUBE PARA SERVICIOS DE IoT*. Universidad Politécnica de Valencia.

Palacios Ibarra, M. A., & Zúñiga Pérez, S. L. (2019). Análisis de Tecnologías de Radio para la Transmisión de Información al Usuario de un Sistema de Control de Flota [Universidad del Cauca]. In *Uiversidad del Cauca*. http://dtm.unicauca.edu.co/pregrado/conmutacion/transp/1-Introduccion.pdf

Pisano, A. (2018). *Internet de la Cosas* [Univesidad de San Andrés]. https://www.owasp.org/images/3/36/IoT_CyberSecurity.pdf

Quito, M. del D. M. de. (2019a). *QUITO SE PROYECTA AL 2020 COMO UNA "SMART CITY."* DMQ. http://www.quito.gob.ec/index.php/municipio/245-sistema-metropolitano-de-transporte

Quito, M. del D. M. de. (2019b). *Sistema Metropolitano de Transporte | Municipio del Distrito Metropolitano de Quito*. Dmq. http://www.quito.gob.ec/index.php/municipio/245-sistema-metropolitano-de-transporte

Sabas. (2017). *Haciendo IoT con LoRa_ Capitulo 2 Tipos y Clases de Nodos*. Beelan. https://medium.com/beelan/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be

Sunfouder. (2017). *Lcd1602-2*. Sunfouder. http://wiki.sunfounder.cc/index.php?title=File:Lcd1602-2.jpg

Tapia Ayala, C. H., & Manzano Yupa, H. M. (2013). *Evaluacion De La Plataforma Arduino E Implementación De Un Sistema De Control De Posicion Horizontal* [Universidad Politécnica Salesina Sede Guayaquil]. https://dspace.ups.edu.ec/bitstream/123456789/5522/1/UPS-GT000511.pdf

Tavizon, A., Laines, C., & Guajardo, T. (2016). IOT, el internet de las cosas y la innovación de sus aplicaciones. *ResearchGate*, *May*. https://www.researchgate.net/publication/326129401_IOT_el_internet_de_las_cosas_y_la_innovacion_de_sus_aplicaciones

Torres Rodríguez, J. A., & Patiño López, L. F. (2019). Diseño e Implementación de un Sistema Embebido Basado en IoT para la Gestión del Transporte Público. In *Journal of Chemical Information and Modeling* (Vol. 0, Issue 0). https://doi.org/10.1017/CBO9781107415324.004

Toyota España. (2013). *Todo sobre los ITS, los Sistemas de Transporte Inteligentes*. Www.Motorpasion.Com. http://www.motorpasion.com/espaciotoyota/todo-sobre-los-its-los-sistemas-de-transporte-inteligentes

ANEXO 1

PROGRAMACIÓN EN SOFTWAR E ARDUINO IDE DE NODO AUTOBÚS

gps_lora Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

Subir

gps_lora §

```
#include <SoftwareSerial.h>
#include <TinyGPS.h>
#include <SPI.h>
#include <RH_RF95.h>

TinyGPS gps;
SoftwareSerial ss(3, 4); // Arduino RX, TX ,
// Singleton instance of the radio driver
RH_RF95 rf95;
String message="";
static void smartdelay(unsigned long ms);
void setup() {
  // put your setup code here, to run once:
 // initialize both serial ports:
  Serial.begin(9600);  // Serial debug
  ss.begin(9600); // SoftSerial  GPS data.
  if (!rf95.init())
    Serial.println("init failed");
  Serial.println("Inicio...");
  rf95.setFrequency(915.0);
  rf95.setTxPower(13); //dbm
  rf95.setSpreadingFactor(7);
  // Setup BandWidth, option: 7800,10400,15600,20800,31200,41700,62500,125000,250000,500000
  //Lower BandWidth for longer distance.
  rf95.setSignalBandwidth(125000);
  rf95.setCodingRate4(5);
  Serial.println("Inicio programa...");
```

gps_lora Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

gps_lora §

```
void loop() {
  // put your main code here, to run repeatedly:
float flat, flon;
unsigned long age;

smartdelay(1000);
gps.f_get_position(&flat, &flon, &age);
float velocidad = gps.f_speed_kmph();
//<localchannel>&latitude&longitude&speed
if((flat != TinyGPS::GPS_INVALID_F_ANGLE ) || (flon != TinyGPS::GPS_INVALID_F_ANGLE ) || (velocidad != TinyGPS::GPS_INVALID_F_SPEED )  ){
    //valores correctos
      message="<4536>ZhFrlnjcp1&";
      message.concat(String(flat,6));
      message.concat("&");
      message.concat(String(flon,6));
      message.concat("&");
      message.concat(String(velocidad,2));
      Serial.println(message);
      //Serial.println(message);

      uint8_t dataArray[message.length()+1];
      message.toCharArray((char*)dataArray, message.length()+1); //conversion
      rf95.send(dataArray, sizeof(dataArray)); // envio mensaje lora
      rf95.waitPacketSent();
  }else{
    Serial.println("DATOS NO VALIDOS");
  }
```

gps_lora Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

gps_lora §

```
      message.concat(String(flon,6));
      message.concat("&");
      message.concat(String(velocidad,2));
      Serial.println(message);
      //Serial.println(message);

      uint8_t dataArray[message.length()+1];
      message.toCharArray((char*)dataArray, message.length()+1); //conversion
      rf95.send(dataArray, sizeof(dataArray)); // envio mensaje lora
      rf95.waitPacketSent();
  }else{
    Serial.println("DATOS NO VALIDOS");
  }

}
static void smartdelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
    while (ss.available())
    {
      //ss.print(Serial.read());
      gps.encode(ss.read());
    }
  } while (millis() - start < ms);
}
```

ANEXO 2

PROGRAMACIÓN EN SOFTWARE ARDUINO IDE DE NODO GATEWAY

programa_gatw_1 Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

programa_gatw_1 §

```
#define BAUDRATE 115200
#include <Console.h>
#include <Bridge.h>
#include <SPI.h>
#include <RH_RF95.h>
#include <Process.h>
#include <FileIO.h>

RH_RF95 rf95;  // Singleton instance of the radio driver
static unsigned long newtime;
static uint8_t packet[64];
int data_pos;
boolean data_format=false;

int index=0;
char ch; //char acumulador
char ch_a[40]; //mqtt msg
int size_ch=0;

void setup()
{
      Bridge.begin(BAUDRATE);
   // while (!Console) ; // Wait for console port to be available
    delay(1000);
   if (!rf95.init())
        Console.println("init failed");
        rf95.setFrequency(915.0);
```


programa_gatw_1 Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda
programa_gatw_1 §
delay(1000);
if (!rf95.init())
Console.println("init failed");
rf95.setFrequency(915.0);
rf95.setTxPower(13);
rf95.setSpreadingFactor(7);
rf95.setSignalBandwidth(125000);
rf95.setCodingRate4(5);
Console.print(F("Inicio"));
}
void loop()
{
if (rf95.available())
{
// Should be a message for us now
uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);
if (rf95.recv(buf, &len))
{
//Console.print("mensaje: ");
//Console.println((char*)buf);
uint8_t data[64] = {'\0'};
char id[8] = {'\0'};
int i;
if (buf[0] == 0x3C) // Only process if data is start with <
{
for (i = 0; i < len; i++)

programa_gatw_1 Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

programa_gatw_1 §

```
        {
            id[i]= buf[i];
            if (buf[i] == 0x3E)
            {
              data_pos = i+1;
              if (data_pos < len)
              data_format=true;
              break; //Get ASCII >, so break;
              data_format==false;
              break; // Data Format mismatch, quit process.
            }
        }
       int h = 0;
       char id1[8] =  {'\0'};
        for (int j = 1; j< i;j++)
        {
          id1[h++] = id[j];
        }
         for (int k=data_pos;k<len;k++)
        {
          data[k-(i+1)]=buf[k];
        }

    //Console.println(id1);
    Console.println((char*)data);
    Process p;
    p.begin("store_data");
```

programa_gatw_1 Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

Verificar

programa_gatw_1 §

```
      p.begin("store_data");
      p.addParameter(idl);
      p.addParameter(data);
      p.run();
    }
    }else{
      Console.println("recv failed");
    }

}

//read uncoming mqtt messages
File  file=FileSystem.open("/mnt/gps_lora/ZhFrlnjcpl.txt",FILE_READ);

if(file){
  unsigned long tam_arch = file.size();
    if(tam_arch>0 ){
      index=0;
      memset(ch_a,0,sizeof(ch_a));
      while (file.available()){ //read all txt file
      ch = file.read();
        ch_a[index]=ch;
      index++;
    }
    ch_a[index-1] = '\0';
     //file.flush();
     file.close();
```

programa_gatw_1 Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

programa_gatw_1 §

```
    File file=FileSystem.open("/mnt/gps_lora/ZhFrlnjcpl.txt",FILE_WRITE); //empty file
    file.close(); //close file
   }else{
     memset(ch_a,0,sizeof(ch_a));
     file.close();
     if(tam_arch != 0){
     File file=FileSystem.open("/mnt/gps_lora/ZhFrlnjcpl.txt",FILE_WRITE); //empty file
     file.close(); //close

     }
   }
  }else{
 // Console.println("Error Opening file");
    delay(1);
   }
   size_ch = strlen(ch_a);
   if(size_ch>0){
     //envio mensaje lora a parada
     Console.println(ch_a);
     rf95.send(ch_a, sizeof(ch_a));
     rf95.waitPacketSent();
     memset(ch_a,0,sizeof(ch_a));

   }
 feeddog();
}
```

programa_gatw_1 Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

programa_gatw_1 §

```
      rf95.waitPacketSent();
      memset(ch_a,0,sizeof(ch_a));

    }
 feeddog();
}

void feeddog()
{
    int i = 0;

    memset(packet, 0, sizeof(packet));

    Process p;     // Create a process
    p.begin("date");
    p.addParameter("+%s");
    p.run();
    while (p.available() > 0 && i < 32) {
        packet[i] = p.read();
        i++;
    }
    newtime = atol(packet);

    File dog = FileSystem.open("/var/iot/dog", FILE_WRITE);
    dog.println(newtime);
    dog.close();
}
```

ANEXO 3

PROGRAMACIÓN EN SOFTWARE DE NODO SERVIDOR EN LINUX

```
<?php
/* mqqt sub to listen messages for zLwi1njcin api_key user
   Stop buses system
   Calculate distance of two points and time to reach stop bus 2
   Length in meters of 1° of latitude = always 111.32 km
   Length in meters of 1° of longitude = 40075 km * cos (latitude ) / 360
*/
require("config_pg.php");
$client = new Mosquitto\Client();
$client->onConnect('connect');
$client->onDisconnect('disconnect');
$client->onSubscribe('subscribe');
$client->onMessage('message');
$client->setCredentials ('mqtt', 'm2mlight12');
$client->connect ("local host", 1883, 60);
$client->subscribe("/zLwi1njcin/#", 1);
//$client->loopForever($timeout = 30000);
$client->loopForever ();
$client->disconnect();
unset($client);
```

```
function connect($r)
//        echo "I got code {$r}\n";
}
function subscribe () {
//        echo "Subscribed to a topic\n";
}
function message($message) {
//   printf ("Got a message on topic %s with payload: %s\n",
//                $message->topic, $message->payload);
   $api_key = substr($message->topic, 1);
   $value = $message->payload;
echo "api_key= “. $api_key."  value= “. $value."\n";
   // For location sensor values, Topic: user_api_key/sensorloc/
   // In $value (payload):  sensor_api_key&latitude&longitude&speed
   $resul = strpos ($api_key, "sensorloc");
   if (¡$resul! == false) {
      $user_api_key = substr ($api_key,0, $resul-1);
echo "user_api_key= “. $user_api_key;
      $pos_sep = strpos($value,"&"); //first &
      if ($pos_sep! == false) {
         $api_key = substr ($value,0, $pos_sep);
            $sensor_api_key = $api_key;
         $pos_sep2 = strpos ($value,"&”, $pos_sep+1);
```

```
//echo " ak= “. $api_key;
            if (¡$pos_sep2! == false) {
               $latitude = substr ($value, $pos_sep + 1, $pos_sep2 - $pos_sep -1);
//echo " lat= “. $latitude;
            $pos_sep3 = strpos ($value,"&”, $pos_sep2+1);
            if ($pos_sep3! == false)
               $longitude = substr ($value, $pos_sep2 + 1, $pos_sep3 -$pos_sep2 -1);
//echo " lon= “. $longitude;
               $speed = substr ($value, $pos_sep3 + 1);
                }else{
               $longitude = substr ($value, $pos_sep2 + 1);
               $speed = 0;
            }
//echo " vel= “. $speed;
            if (is_numeric($latitude) and is_numeric($longitude) and is_numeric($speed)) {
               exec ("php send_sensor_location_mqtt.php “. $api_key. " " . $latitude. " " .
                     $longitude. " " . $speed);
                  sleep (1); // wait a second
               send_time_reach_stop2($user_api_key, $sensor_api_key);
                }
         }
      }
   } else {
```

```
    // For single sensor values, Topic: user_api_key/sensor/
    // In $value (payload):  sensor_api_key&value. Example: Uy4rF45ty&23.25
    $resul = strpos ($api_key, "sensor");
    if ($resul! == false) {
      $pos_sep = strpos($value,"&");
      if ($pos_sep! == false) {
        $api_key = substr ($value,0, $pos_sep);
        $value = substr ($value, $pos_sep + 1);
        if (is_numeric($value))
          exec ("php send_sensor_value_mqtt.php “. $api_key. " " . $value. " > /dev/null &");
      }
    } else {
      $resul = strpos ($api_key, "alert");
      if ($resul! == false) { // The format to receive alerts: user_api_key/alert/
        // the alert_pi_key is in the payload
        $alert_api_key = $value;
echo "alert_api_key=". $alert_api_key;
        $command="php
/var/www/vhosts/m2mlight.com/httpdocs/iot/mqtt_send_email_alert.php ".
                    $alert_api_key;
        exec ($command, $output);
       }
    } //sensor
```

```
  } //sensorloc
}
function disconnect () {
//       echo "Disconnected cleanly\n";
}
// send a mqtt message to GW, with the time to reach the second stop
function send_time_reach_stop2($user_api_key, $sensor_api_key) {
  global $con;
  $sql="select sensor_id from sensor
          where api_key = '$sensor_api_key'";
  $result=pg_query($sql);
  $value3 = pg_fetch_array($result);
  $sensor_id = $value3["sensor_id"];
  // Getting last 3 speeds and positions within an interval of 30 secs.
  $sql="select latitud, longitud, velo_gps from sensor_location
       where sensor_id='$sensor_id'
       and tiempo > (extract (epoch from now () AT TIME ZONE 'COT') - 30)
       order by tiempo desc limit 3";
  $result = pg_query($sql);
  if (! $result) {
    echo pg_last_error ();
    exit;
  } else {
```

```
    $numRows = pg_num_rows($result);
    if ($numRows==0) {
      echo "Error: there are no data in sensor location";
      exit;
    }
  }
  $i = 0;
  $tot_velo = 0;
  $row = pg_fetch_assoc($result);
  for ($i=0; $i < $numRows; $i++) {
    if ($i == 0) { // bus current position
      $bus_lat = $latitud=$row["latitud"];
      $bus_lon = $longitud=$row["longitud"];
    }
    $velo_gps=$row["velo_gps"];
    $tot_velo = $tot_velo + $velo_gps;
  }
  // speed average
  $speed_avg = $tot_velo / $numRows;
echo " speed_avg= “. $speed_avg;
  //Stop 2
  $stop_lat = -0.170473;
  $stop_lon = -78.479821;
```

```
   //Stop 1
   $stop_lat1 = -0.173145;
   $stop_lon1 = -78.480256;
   // delta latitude & longitude
   $d_lat = $stop_lat - $bus_lat;
   $d_lon = $stop_lon - $bus_lon;
   $d_lat1 = $stop_lat1 - $bus_lat;
   $d_lon1 = $stop_lon1 - $bus_lon;
   // dy and dx in km
   $dy = $d_lat * 111.32;
   $m_lat = ($stop_lat + $bus_lat) / 2; //latitude media
   $dx = $d_lon * 40075 * cos($m_lat) / 360;
   $dy1 = $d_lat1 * 111.32;
   $m_lat1 = ($stop_lat1 + $bus_lat) / 2; //latitude media
   $dx1 = $d_lon1 * 40075 * cos($m_lat1) / 360;
//echo " dx= “. $dx;
//echo " dy= “. $dy;
   $dist = sqrt ($dx * $dx + $dy * $dy);
   $dist1 = sqrt ($dx1 * $dx1 + $dy1 * $dy1);
    echo " dist= “. $dist;
   // Time calculation
   if ($speed_avg > 0) {
```

```
      // Stop 2
      $dtime = $dist / $speed_avg; // in hours
      $dtime = $dtime * 3600; // in seconds
      $dtime = number_format($dtime,2);
      if ($bus_lat > $stop_lat) // se paso la parada 2
        $dtime = "PASO";
echo " dtime= “. $dtime;
      // Stop 1
      $dtime1 = $dist1 / $speed_avg; // in hours
      $dtime1 = $dtime1 * 3600; // in seconds
      $dtime1 = number_format($dtime1,2);
      if ($bus_lat > $stop_lat1) // se paso la parada 1
        $dtime1 = "PASO";
     // Send mqtt message about Stop 2
     $id_mess = str_pad (rand (1,9999), 4, "0", STR_PAD_LEFT);
     $message = 'm2m’. $id_mess.'\&stop=2\&time= ‘. $dtime;
     $command="php/var/www/vhosts/m2mlight.com/httpdocs/iot/mqtt_pub.php“. $user_api_key.
              " “. $message;
     //echo $command;
     exec ($command, $output);
     sleep (1);
     // Send mqtt message about Stop 1
     $id_mess = str_pad (rand (1,9999), 4, "0", STR_PAD_LEFT);
```

```
    $message = 'm2m’. $id_mess.'\&stop=1\&time= ‘. $dtime1;
    $command="php/var/www/vhosts/m2mlight.com/httpdocs/iot/mqtt_pub.php“. $user_api_key.
            " “. $message;
    exec ($command, $output);
  }
}
?>
```

ANEXO 4

PROGRAMACIÓN EN SOFTWARE ARDUINO IDE DE NODO PARADA

parada Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

parada

```
#include <LiquidCrystal_I2C.h>
#include <RH_RF95.h>
#include <SPI.h>

RH_RF95 rf95;  // Singleton instance of the radio driver
LiquidCrystal_I2C lcd(0x27,16,2); // si no te sale con esta direccion  puedes usar (0x3f,16,2) || (0x27,16,2)  ||(0x20,16,2)


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  if (!rf95.init())
       Serial.println("Init Error");
   rf95.setFrequency(915.0);
   rf95.setTxPower(13);
   rf95.setSpreadingFactor(7);
   rf95.setSignalBandwidth(125000);
   rf95.setCodingRate4(5);
   Serial.println("Inicio...");
//Para LCD
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Proximo bus en:");
  lcd.setCursor (0,1);
  lcd.display();
```

parada Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

parada

```
void loop() {
  // put your main code here, to run repeatedly:
if (rf95.available())
  {
  // Should be a message for us now
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (rf95.recv(buf, &len))
    {
     //Serial.print("mensaje: ");
      //Serial.println((char*)buf);
      uint8_t aux[4];
      for(int j=0;j<3;j++){
        aux[j]=buf[j];
      }
      aux[3]='\0';
      if(strcmp(aux,"m2m")==0){
        // Serial.print("mensaje: ");
        // Serial.println((char*)buf);
        char *token;
        const char s[2] = "&";
        const char parametro_1[6] = "stop=";
        const char parametro_2[6] = "time=";
        const char parada[2] = "2d<d";
        char stp[2],ti_me[5];

        token = strtok(buf, s);
```

parada Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

parada

```
while( token != NULL ) {
  if(strncmp(token, parametro_1,5)== 0){ //stop
    //Serial.println("entro stop");
    //Serial.println((char*)token);
    for(int i=5; i<=strlen(token)+1;i++){
    stp[i-5]=token[i];
    //Serial.println((char*)stp);
    }
  }else if((strncmp(token, parametro_2,5)== 0 )&& (strcmp(stp,parada)== 0)){
    //Serial.println("entro time");
    //Serial.println((char*)token);
    for(int i=5; i<=strlen(token)+1;i++){
    ti_me[i-5]=token[i];
    //Serial.println((char*)ti_me);
    lcd.clear();
    delay(200);
    lcd.print("Proximo bus en:");
    lcd.setCursor (0,1);
    lcd.print((char*)ti_me);
    }

  }else{
  delay(1);
  }

  token = strtok(NULL, s);
}
```