UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA SUR

FACULTAD DE INGENIERÍA Y GESTIÓN

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

"DISEÑO, IMPLEMENTACIÓN Y TESTEO DE UNA RED DE INSTRUMENTOS INALÁMBRICOS CON CONEXIÓN DIRECTA A INTERNET Y DE BAJO COSTO PARA UN SISTEMA DE ALERTA TEMPRANA DE HUAICOS EN LA QUEBRADA DE JICAMARCA"

TESIS

Para optar el Título Profesional de

INGENIERO ELECTRÓNICO Y TELECOMUNICACIONES

PRESENTADO POR EL BACHILLER

MELÉNDEZ COVEÑAS, FRANK ENRIQUE

Villa El Salvador
2021

DEDICATORIA

A mi familia, por su incondicional apoyo.

AGRADECIMIENTOS

Al Programa Nacional de Innovación para la Competitividad y Productividad, Innóvate Perú, por los fondos asignados al proyecto 384-PNICP-PIAP-2014.

Al Radio Observatorio de Jicamarca, por brindarme la oportunidad y el apoyo para la realización del presente trabajo.

A Joaquín Verástegui por su enorme apoyo y paciencia como asesor en el Radio Observatorio de Jicamarca.

A Ricardo Palomares, por apoyarme como asesor de tesis en este largo camino y brindarme su tiempo para la redacción del documento.

A mi familia, Fernando Meléndez, Rufina Coveñas, Jonathan Meléndez, Nataly Meléndez, por ser parte de mi vida y la fuerza que me sostiene.

A todas las personas que de alguna u otra forma nos permiten, con sus enseñanzas, encontrar nuestro propio camino, en este regalo maravilloso que llamamos vida.

RESUMEN

La propuesta planteada en la tesis titulada Diseño, Implementación y Testeo de una Red de Instrumentos Inalámbricos con Conexión Directa a Internet y de Bajo Costo para un Sistema de Alerta Temprana de Huaicos en la Quebrada de Jicamarca consistió de dos Estaciones de Monitoreo que se conectaron a un Punto de Acceso. El Punto de Acceso se conectó a internet y envió los datos de prueba a un servidor en la nube. Se pudo acceder a los datos de forma remota.

Se usó un conjunto de protocolos de comunicación IoT. Se utilizó el protocolo CoAP en la transferencia de datos de las Estaciones de Monitoreo al Punto de Acceso. Se aprovechó la tecnología WiFi de largo alcance de los módulos ESP32 para lograr enlaces inalámbricos entre las Estaciones de Monitoreo y el Punto de Acceso. Se empleó el módulo SIM7000E para conectar el Punto de Acceso a la red de internet móvil a través de la tecnología CAT-M1. Se implementó el protocolo MQTT para acceder a los datos enviados a la nube.

Se realizaron cinco pruebas y se obtuvieron resultados exitosos en cada prueba. La primera prueba consistió en transmitir datos de prueba de las Estaciones de Monitoreo al Punto de Acceso, como resultado se obtuvo el envío de datos de las Estaciones de Monitoreo al Punto de Acceso. La segunda prueba consistió en transmitir datos a internet a través de la red móvil, como resultado se lograron transmitir datos de prueba del Punto de Acceso a un servidor en la nube a través de red móvil. La tercera prueba consistió en visualizar los datos de prueba en el servidor en la nube, como resultado se logró visualizar el flujo de datos enviados por el Punto de Acceso a la nube a través de una interfaz gráfica. La cuarta prueba consistió en el procesamiento de los datos de prueba y análisis de resultados, se logró obtener gráficos de latencia en el envío de los datos de prueba y la frecuencia de transmisión. La quinta prueba fue el presupuesto de la red de instrumentos inalámbricos, obteniéndose como resultado un prototipo de bajo costo. De esta manera, se logró cumplir con el objetivo propuesto en el proyecto 384-PNICP-PIAP-2014 financiado por el Programa Nacional de Innovación para la Competitividad y Productividad, Innóvate Perú, pues se obtuvo un prototipo de red de instrumentos inalámbricos de bajo costo con tecnología IoT con conexión directa a internet para el Radio Observatorio de Jicamarca que puede ser integrado a su Sistema de Alerta Temprana de Huaicos.

Palabras clave: Tecnología IoT (Internet of things), Cat-M1 (Categoría M1), CoAP (Constrained Application Protocol)

ABSTRACT

The proposal presented in the thesis entitled Design, Implementation and Testing of a Network of Wireless Instruments with Direct Internet Connection and Low Cost for an Early Warning System of Huaicos in the Quebrada de Jicamarca consisted of two Monitoring Stations that were connected to an Access Point. The Access Point was connected to the internet and sent the test data to a server in the cloud. Data could be accessed remotely.

A set of IoT communication protocols was used. The CoAP protocol was used to transfer data from the Monitoring Stations to the Access Point. The long-range WiFi technology of the ESP32 modules was used to achieve wireless links between the Monitoring Stations and the Access Point. The SIM7000E module was used to connect the Access Point to the mobile internet network through CAT-M1 technology. The MQTT protocol was implemented to access data sent to the cloud.

Five tests were conducted and successful results were obtained in each test. The first test consisted of transmitting test data from the Monitoring Stations to the Access Point, as a result, data was sent from the Monitoring Stations to the Access Point. The second test consisted of transmitting data to the internet through the mobile network, as a result it was possible to transmit test data from the Access Point to a server in the cloud through the mobile network. The third test consisted of viewing the test data in the cloud server, as a result it was possible to visualize the data flow sent by the Access Point to the cloud through a graphical interface. The fourth test consisted of the processing of the test data and analysis of the results, it was possible to obtain latency graphs in the sending of the test data and the transmission frequency. The fifth test was the budget of the wireless instrument network, resulting in a low-cost prototype. In this way, it was possible to meet the objective proposed in project 384-PNICP-PIAP-2014 financed by the National Innovation Program for Competitiveness and Productivity, Innóvate Perú, since a prototype of a low-cost wireless instrument network was obtained with IoT technology with direct connection to the internet for the Jicamarca Radio Observatory that can be integrated into its Huaicos Early Warning System.

Keywords: IoT technology (Internet of Things), Cat-M1 (Category M1), CoAP (Constrained Application Protocol)

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

Un sistema de alerta temprana de desastres naturales es muy importante para la población, pues permite que se tomen acciones antes de la llegada del fenómeno. Hemos sido testigos de innumerables tragedias en nuestro país ocasionados por este tipo de desastres y, a pesar de saber que estamos expuestos y propensos a ello, se hace muy poco por prevenir, a pesar de que existe tecnología para idear sistemas que permiten proteger a la población y mitigar los efectos que pueden ocasionar sismos, maremotos, fuertes lluvias o huaicos.

Bajo este contexto, el Radio Observatorio de Jicamarca (ROJ), sede del Instituto Geofísico del Perú (IGP), posee un sistema de alerta temprana de huaicos, que envía notificaciones con el fin de advertir a las autoridades competentes para que tomen las medidas que mejor convengan. Permite también alertar a Sedapal, con quien tienen un contrato, para que cierren las compuertas y evitar la contaminación del agua potable que abastece a la ciudad de Lima.

La tecnología que se usa para la tesis es la Internet de las Cosas, que proporciona herramientas para diseñar y crear redes que se pueden adaptar a necesidades simples como el envío de datos que requieren tráfico con bajas tasas de transmisión.

Internet de las Cosas permite crear redes privadas que pueden conectarse a las redes móviles.

Debido a problemas en cobertura, Internet de las Cosas presenta soluciones para crear redes privadas que permiten la transferencia de datos a través de un backhaul, es decir, un enlace que permita llevar los datos a internet, para luego poder acceder a ellos de forma remota.

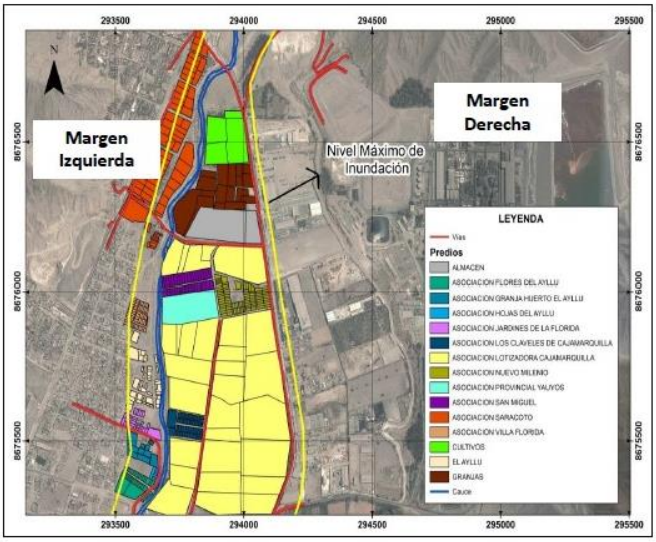
Lo que esta tesis propone es una red que consta de una parte que permite trasferir datos de una o más Estaciones de Monitoreo a un Punto de Acceso a través de enlaces privados en una zona inhóspita y de aquí enviarlos a internet usando la tecnología celular, siendo la base de esta red las tecnologías de Internet de las Cosas.

En el Capítulo I se elabora el planteamiento del problema, se presentan los objetivos, delimitaciones y justificaciones. En el Capítulo II se elabora el marco teórico sobre el que se sustenta el trabajo, se presentan los antecedentes y bases teóricas. En el Capítulo III se desarrolla la metodología, es decir, se expone el diseño de la investigación, se describe la implementación y las pruebas realizadas. Finalmente, el Capítulo IV presenta las conclusiones.

I. PLANTEAMIENTO DEL PROBLEMA

1.1 Descripción del problema

De acuerdo a la estimación de la vulnerabilidad ante eventos aluvionales hecho por el Instituto Geofísico del Perú el cual se puede ver en la figura 1, muestra que frente a eventos aluvionales en la quebrada de Jicamarca se ven expuestos un gran número de predios de varias localidades cercanas al evento natural como Saracoto, Villa Florida, San Miguel, Nuevo Milenio, entre otros, así como cultivos, granjas y almacenes. En la localidad de Nuevo Milenio, que es una de las zonas consideradas, de 90 predios 50 están expuestos.



	Asociación	Área (A) x ha.	% del (A) expuesta x ha.	N° de predios	N° predios expuestos
Margen Derecha	Saracoto	40	12	280	80
	Villa Florida	3	3	120	120
	Hojas del Ayllu	8	8	15	15
	El Ayllu			50	50
	Jardines de la Florida			40	40
	Huerto el Ayllu			30	30
	Flores del Ayllu			45	45
Margen Izquierda	cultivos, granjas y almacén	10	10	3	3
	San Miguel	5	5	60	60
	Nuevo Milenio			90	50
	Claveles de Cajamarquilla	2	2	40	40
	Lotizadora Cajamarquilla	37	26	40	27
	Total	105	66	798	545

Figura 1 Estimación de la vulnerabilidad ante eventos aluvionales
Fuente: Radio Observatorio de Jicamarca

Además, según el Reporte Nacional N° 23-2017 del Centro Nacional De Epidemiología, Prevención y Control de Enfermedades, el cual se puede consultar en la figura 2, el 31 de enero del 2017, el desborde de la quebrada Jicamarca (río Huaycoloro) y el río Rímac afectaron el distrito de San Juan de Lurigancho registrándose 18 familias damnificadas, 164 familias afectadas, 18 viviendas colapsadas y 164 viviendas afectadas.

PERÚ Ministerio de Salud | Centro Nacional de Epidemiología, Prevención y Control de Enfermedades

VIGILANCIA DE EVENTOS SANITARIOS EN MEDIOS DE COMUNICACIÓN

PARA DETECTAR BROTES, EPIZOOTIAS, DESASTRES Y OTRAS EMERGENCIAS SANITARIAS

REPORTE N° 23 - 2017 MIERCOLES 1 DE FEBRERO DEL 2017

REPORTE NACIONAL

BROTES, EPIZOOTIAS Y OTROS REPORTES DE SALUD

No se ha registrado ningún rumor o noticia de importancia para la salud pública nacional.

DESASTRES Y OTRAS EMERGENCIAS SANITARIAS

Hay más de 500 familias afectadas por el desborde del Huaycoloro

LIMA I El alcalde de San Juan de Lurigancho, Juan Navarro, informó que más de 500 familias de su jurisdicción resultaron afectadas por el desborde del río Huaycoloro, ocurrido ayer en la tarde.

Fuente:http://rpp.pe/lima/desastres-naturales/hay-mas-de-500-familias-afectadas-por-el-desborde-del-huaycoloro-noticia-1027747?ns_source=self&ns_mchannel=portada.home&ns_campaign=content.cronologico&ns_linkname=1

La noticia fue verificada por el Equipo Alerta Respuesta ante la instancia correspondiente concluyendo en:

NO CONCORDANTE: INDECI informa que el 31/01/17, aproximadamente a las 16:15 horas, a consecuencia de las intensas precipitaciones pluviales se produjo el incremento y posterior desborde de la quebrada Jicamarca (río Huaycoloro) y el río Rímac, afectando el distrito de San Juan de Lurigancho, de la provincia de Lima. Se registraron: 18 familias damnificadas, 164 familias afectadas, 18 viviendas colapsadas y 164 viviendas afectadas. Personal local continúa con la evaluación de los daños y el análisis de las necesidades.

Figura 2 Reporte Nacional N° 23-2017 del Centro Nacional De Epidemiología, Prevención y Control de Enfermedades
Fuente: http://www.dge.gob.pe/portal/docs/rumores/2017/Reporte%20027-2017.pdf

Este tipo de fenómenos naturales son comunes en estas zonas, por lo que El Radio Observatorio de Jicamarca, sede del Instituto Geofísico del Perú, cuenta con un Sistema de Alerta Temprana de Huaicos que monitorea la quebrada de Jicamarca. La información obtenida por los sensores es procesada y enviada a internet. notificando a las autoridades competentes para que tomen las medidas que mejor convengan y a Sedapal; con la que tienen un convenio, que les permite cerrar sus compuertas con una anticipación de una hora aproximadamente.

El sistema de alerta temprana de huaicos se encuentra en un lugar remoto y se conecta al internet del Radio Observatorio de Jicamarca. El internet del Radio Observatorio de Jicamarca depende de la conexión a internet del Instituto Geofísico del Perú, que se da por medio de un radioenlace. Si hay corte del servicio de internet, el sistema queda aislado. El corte del servicio se puede dar por distintas razones, por la desconexión de cables, corte del suministro eléctrico, los huaicos, fallas en la antena del radioenlace entre el Instituto Geofísico del Perú y el Radio Observatorio de Jicamarca. La existencia de varios intermediarios aumenta las posibilidades de cortes en el servicio de internet.

De acuerdo al Radio Observatorio de Jicamarca el sistema actual está presupuestado en cuarenta y nueve mil soles como se ve en la figura 3. Frente a la ocurrencia de huaicos

suele ocurrir daños al sistema como la pérdida de componentes que deben ser repuestos a la brevedad, generándose gastos elevados por mantenimiento. Los cables que conectan las estaciones y los sensores sufren daños debido al entorno en el que se encuentran por lo que deben ser cambiados cada cierto tiempo.

ANEXO 1: PRESUPUESTO

Item	Componente	Cantidad	Costo Unitario	Costo Total
1	Estación cliente (sensores, sistema de energía solar, comunicación inalámbrica)	2	12,400	24,800
2	Estación repetidora (sistema de energía solar y comunicación inalámbrica)	1	7,200	7,200
3	Estación central (servidor, ups, comunicación inalámbrica, sirena)	1	12,000	12,000
4	Servicios (instalación, buzones, malla proteccion)	1	5,000	5,000
Total S/.				49,000

Figura 3 Presupuesto Sistema de Alerta Temprana de Huaicos Actual del Radio Observatorio de Jicamarca
Fuente: Radio Observatorio de Jicamarca

1.2 Formulación del problema

1.2.1 Problema general

¿Cómo lograr que un sistema de alerta temprana de huaicos garantice la conexión permanente a internet y disminuya su costo de implementación y mantenimiento en la quebrada de Jicamarca?

1.2.2 Problemas específicos

- ¿Qué tecnología garantiza la conexión permanente a internet de un sistema de alerta temprana de huaicos en la quebrada de Jicamarca?
- ¿Cómo disminuir el costo de implementación y mantenimiento de un sistema de alerta temprana de huaicos en la quebrada de Jicamarca?

1.3 Objetivos de la investigación

1.3.1 Objetivo general

Diseñar, implementar y testear una red de instrumentos inalámbricos con conexión directa a internet y con tecnologías de bajo costo para un sistema de alerta temprana de huaicos en la quebrada de Jicamarca.

1.3.2 Objetivos específicos

- Diseñar, implementar y testear una red de instrumentos inalámbricos con conexión directa a internet.
- Diseñar e implementar una red de instrumentos inalámbricos con tecnologías de bajo costo.

1.4 Delimitación de la investigación

Se desarrolla, como prueba de concepto, un prototipo de red de instrumentos inalámbricos que pueda integrarse al sistema de alerta temprana que se encuentra en mejora continua, por lo cual el trabajo corresponde a un desarrollo tecnológico.

1.4.1 Delimitación espacial

El presente trabajo se realizó en la Quebrada de Jicamarca, en Cajamarquilla, San Juan de Lurigancho, Lima, Perú.

1.4.2 Delimitación temporal

Bajo el punto de vista temporal, el trabajo de tesis se puede separar en dos etapas. La primera fue la elaboración del prototipo el cual se realizó en las instalaciones del Radio Observatorio de Jicamarca gracias al Programa Nacional de Innovación para la Competitividad y Productividad, Innóvate Perú, por los fondos asignados al proyecto 384-PNICP-PIAP-2014, el cual se llevó a cabo durante el periodo del año 2018-2019. La segunda etapa consta de la documentación para la presentación del trabajo como parte del proceso de sustentación de tesis durante el año 2020.

1.4.3 Delimitación teórica

Se considera dentro de garantizar el seguimiento y alerta del desastre. Se centra en la elaboración de la arquitectura de la red.

1.5 Justificación del problema

Teórica

Permite adaptar nuevos conocimientos prácticos para implementar redes de monitoreo inalámbricas con tecnologías IoT para ser implementada en diversas partes del Perú con fines de monitoreo, prevención de desastres naturales, preservación de reservas naturales, etc.

Económica

Reduce el costo de un Sistema de Alerta Temprana de Huaicos en la quebrada de Jicamarca tanto en su implementación como en su posterior mantenimiento empleando tecnologías IoT de bajo costo y herramientas de desarrollo libre.

Social

Beneficia al Radio Observatorio de Jicamarca, sede del Instituto Geofísico del Perú, brindándole un prototipo de bajo costo, para brindar un mejor servicio a la población y a Sedapal, al contar con alternativas que permiten la disponibilidad constante del Sistema de Alerta Temprana de Huaicos.

Además, al ser de bajo costo favorece su implementación en otros lugares que requieren monitoreo ante este tipo desastres.

Ambiental

Al ser un sistema de bajo costo emplea componentes menos sofisticados y menos recursos lo que hace que tenga menos impacto ambiental.

II. MARCO TEÓRICO

2.1 Antecedentes de la investigación

A continuación, se presenta una relación de estudios e investigaciones relacionados al trabajo de tesis propuesto resaltando brevemente en cada uno de ellos aspectos relacionados a la tesis.

Tesis Nacionales

León, (2019), en su tesis de Pregrado de la Facultad de Ciencias e Ingeniería de la Pontificia Universidad Católica del Perú, titulada "Sistema Automático de Monitoreo de Mercurio en Tiempo Real en aguas aledañas a Exploraciones Mineras y Petroleras usando una plataforma IoT".

Plantea un sistema que consiste de un nodo sensor, un nodo Gateway, un módulo RF tanto en el nodo sensor como en el nodo Gateway para la transmisión de datos, y una topología tipo estrella, para agregar nodos en el caso que se desee. Finalmente, una solución en la nube donde son enviados los datos para ser mostrados en gráficas para poder ser analizados. Tanto el nodo sensor como el nodo Gateway usan microcontroladores para la parte de control, es decir, la gestión de datos. Este sistema considera que el nodo Gateway envía los datos a la nube a través de ethernet, por lo que debe estar en una central de monitoreo donde halla conexión eléctrica, no siendo completamente inalámbrico.

TTacca, (2017), en su tesis de Pregrado de la Escuela Profesional de Ingeniería Electrónica de la Universidad Nacional del Altiplano, titulada "Diseño de una Red Fog Basada en Internet de las Cosas para Monitorear la Contaminación en la Bahía del Lago Titicaca".

Emplea el protocolo MQTT para la transmisión de datos. Para manejar la recolección de datos adquiridos de los sensores utiliza la plataforma Arduino que conecta a la tarjeta de comunicación WiFiR232, esto le permite conectar un nodo de red de manera inalámbrica. Emplea un entorno de desarrollo libre para el diseño de una plataforma virtual para visualizar los parámetros de monitoreo (p. 65)

Presenta un prototipo de nodo capaz de comunicarse con un servidor de manera inalámbrica, además de poder visualizar los datos transportados por la red a través de un portal web.

Aguilar, (2019), en su tesis de Postgrado de Maestría de la Escuela de Posgrado de la Pontificia Universidad Católica del Perú, titulada “Diseño de un Sistema Basado en IoT para la Supervisión y Control de Estaciones Remotas de la Dirección de Hidrografía y Navegación de la Marina de Guerra del Perú”.

Emplea dispositivos de comunicación con tecnología LoRaWAN para transmitir información desde nodos finales instalados en las estaciones remotas hacia los Gateways ubicados en las estaciones de recepción. Emplea una conexión de internet, para retransmitir los datos de los Gateways hacia un servidor de red LoRaWAN. Posteriormente, procesa la información usando una plataforma IoT en una nube computacional en la que desarrolla soluciones de monitoreo y control para ser usados por los usuarios conectados mediante clientes web (p. 31)

Si bien logra cubrir kilómetros de distancia, LoRaWAN tiene limitaciones en cuanto a datos en tiempo real y un ancho de banda muy bajo.

Tesis Internacionales

Shi, (2018), en su tesis de Posgrado de Maestría del Departamento de Ciencias de la Computación de la Universidad de Saskatchewan en Canadá titulada “COAP Infrastructure for IOT”.

Realiza un estudio de la aplicación de Erlang, que es un programa orientado a la concurrencia de procesos (COP), al protocolo COAP. Usa dicho lenguaje para modelar una infraestructura CoAP (servidor y cliente) e implementar un prototipo. Evalúa su escalabilidad y confiabilidad tanto en un entorno con restricciones (niebla) como en un entorno sin restricciones (Cloud). Para ello usa dos servidores CoAP, uno para entornos restringidos y otro para entornos no restringidos. Considera dos parámetros a evaluar, la cantidad de información que se puede intercambiar por tiempo y la latencia. Considera un cliente virtual como un proceso de Erlang y satura los servidores con la concurrencia de clientes. Para el entorno no restringido emplea la nube como servidor, donde aloja ambos servidores CoAP y emplea diferentes benchmark con lo que logra 72000 solicitudes por segundo con el servidor CoAP para entornos no restringidos y 84000 peticiones por segundo con el servidor CoAP para entornos restringidos. Para el entorno restringido emplea un Rasberry Pi 3 donde aloja ambos servidores CoAP y una MacBook Pro como benchmark. De igual forma, satura los servidores con solicitudes de cliente de forma progresiva. Bajo estas condiciones logra 9000 solicitudes por segundo con el servidor CoAP para entornos

restringidos y 4500 solicitudes por segundo con el servidor CoAP para entornos no restringidos, por requerir de mayores recursos, de esta manera demuestra que la filosofía de Erlang se aplica al IoT emergente, y combinado con los sistemas web a gran escala conducen a servicios IoT escalables y confiables (p. 82)

Meneses, (2017), en su tesis de Pregrado de la Facultad de Ingeniería de la Universidad Piloto de Colombia, titulada "Sistema de monitoreo en la nube para medir los riesgos ambientales basados en sensores de bajo costo".

Plantea el desarrollo de un prototipo en el cual se lleva a cabo el procesamiento de los datos censados, es decir, el servidor web se encuentra en el sistema de monitoreo, para lo cual emplea hardware con mayores capacidades, pero no al nivel de un PC tradicional con el fin de mantener el bajo costo. Además, integra un módulo de tecnología celular para enviar mensajes de texto con los datos censados.

Chivata, (2017), en su tesis de Pregrado de la Facultad de Ingeniería de la Universidad Distrital Francisco José de Caldas, titulada "Modelo de red de estaciones meteorológicas modulares con plataforma IOT".

Emplea un microcontrolador de bajo consumo y RAM, dispositivos electrónicos Xbee para la interconexión y comunicación inalámbrica de los datos, la tarjeta de desarrollo Raspberry Pi 3 como Gateway. Emplea el protocolo MQTT para la transferencia de datos. Implementa la solución en la nube en un Web Service alojado en un servidor propio. Emplea una base de datos y un servidor de gestión. Consigue que el usuario final pueda tener control de la información de su interés de manera remota gracias al uso de tecnologías del Internet de las Cosas (p. 54)

Se revisaron papers relacionados al tema de la tesis. Sharevski (2018), en su investigación "Leveraging Cellular Internet-of-Things for Resilient and Robust Disaster Management" discute sobre las diferentes CIoT (Cellular Internet of Things) existentes y su infraestructura. Relaciona los desastres naturales con las CIoT y explica las ventajas que ofrece sobre las WSN como la cantidad de sensores que puede soportar por celda. Compara las tecnologías UAV/WSN con tecnologías CIoT, determinando que CIoT ofrece soporte completo de conectividad y administración centralizada para la mayoría de los tipos de desastres. Desarrolla un ejemplo práctico de gestión de desastres aplicado a una avalancha de nieve que consiste de cámaras para tomar fotografías, transceivers eMTC para la conexión con las celdas LTE, dispositivos UAV para el monitoreo aéreo, fCOW para restablecer el

servicio celular y un LiDAR para monitorear el nivel de nieve y sensores para el monitoreo de personas. Concluye que el estándar para infraestructura celular CIoT está ampliamente desplegado, debido a ser compatibles con las bandas LTE.

Chowdhury et al. (2017), desarrollan un prototipo de enjambre de robots con IoT para el monitoreo de desastres naturales, tomando como principal reto implementar estas tecnologías con un mínimo costo, máxima eficiencia, producción en masa y distribución como una alternativa a sistemas de elevado costo como el que emplea la NASA, que hace uso de un satélite para monitorear áreas afectadas y no es accesible económicamente para muchos. Para ello emplean una madre bot y dos pequeños sub bots. También módulos nRF transceiver para crear una red conectada para el envío y recepción de datos. Un sensor de ultrasonido con un módulo de cámara para obtener datos de mapeo, sensores, aplicaciones móviles para el control de los robots, y un portal web para visualizar el área del mapa afectado por el desastre, recibir mensajes de emergencia y gráficas en tiempo real del mapeo. Arduino para la lectura de los datos de los sensores y Raspberry Pi 3 para controlar los robots. A pesar que logran visualizar los datos en la web, las antenas nRF presentan un alcance de 250 m con línea de vista, siendo afectado si no hay línea de vista.

2.2 Bases teóricas

2.2.1 Red de instrumentos inalámbricos

Para Fernández et al. (2009) una red de instrumentos inalámbricos consta de una red de sensores con dispositivos que permiten que se comuniquen de forma inalámbrica. Para ello, hacen uso del espectro electromagnético, empleando una antena y energía adicional es posible transmitir las señales a gran distancia (p. 17).

2.2.2 Tecnología IoT

IoT es muy amplio, incluso en definiciones, pero se considera, según Cirani (2018), que es un mundo de millones de objetos que poseen inteligencia integrada, sistemas de comunicación y pueden detectar o actuar sobre algún parámetro. Para que todo ello se dé, los objetos deben conectarse a treves de internet (p. 1).

2.2.3 Arquitectura IoT

Para construir una arquitectura de red basada en tecnología IoT se considera una arquitectura simplificada propuesta por Barton, Hanes y Salgueiro (2017) que consta de dos bloques mostrados en la figura 4.

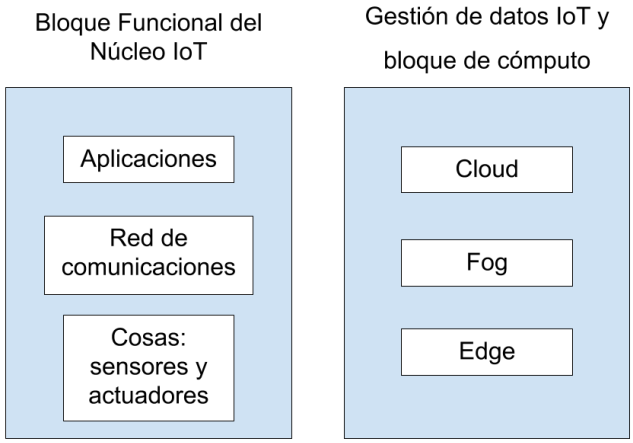


Figura 4 Arquitectura IoT simplificada
Fuente: IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things

Dentro del bloque funcional del núcleo IoT se encuentran la capa de sensores y actuadores, considerados como cosas, la capa de red de comunicaciones y la capa de aplicaciones. Mientras que el bloque de cómputo y gestión de datos IoT consta de la capa Cloud (nube), Fog (niebla) y Edge (borde).

Bloque Funcional del Núcleo IoT

Capa 1: de sensores y actuadores (cosas)

Se debe tener en cuenta parámetros como si las cosas van a ser alimentadas por batería o se conectarán por corriente, serán móviles o estáticos, la frecuencia de reporte será alta o baja, la densidad de datos intercambiados en cada ciclo de informe, la distancia a la cual se encuentra la puerta de enlace, la cantidad de objetos en una misma celda conectados a la misma puerta de enlace.

Capa 2: de red de comunicaciones

Una vez determinado lo anterior, se puede conectar el objeto para comunicarlo.

Dentro de esta capa, se consideran cuatro subcapas, que son subcapa de acceso a red, puerta de enlace y subcapa backhaul, subcapa de transporte de red y subcapa de gestión de red IoT.

a. Subcapa de acceso a red

Para elegir la tecnología IoT que mejor se ajuste a los requerimientos que se planteen, se debe considerar el tipo de topología que esta permita, siendo la distancia entre el objeto inteligente y el recopilador de información un factor importante.

La figura 5 muestra algunas tecnologías de acceso IoT y las distancias de transmisión.

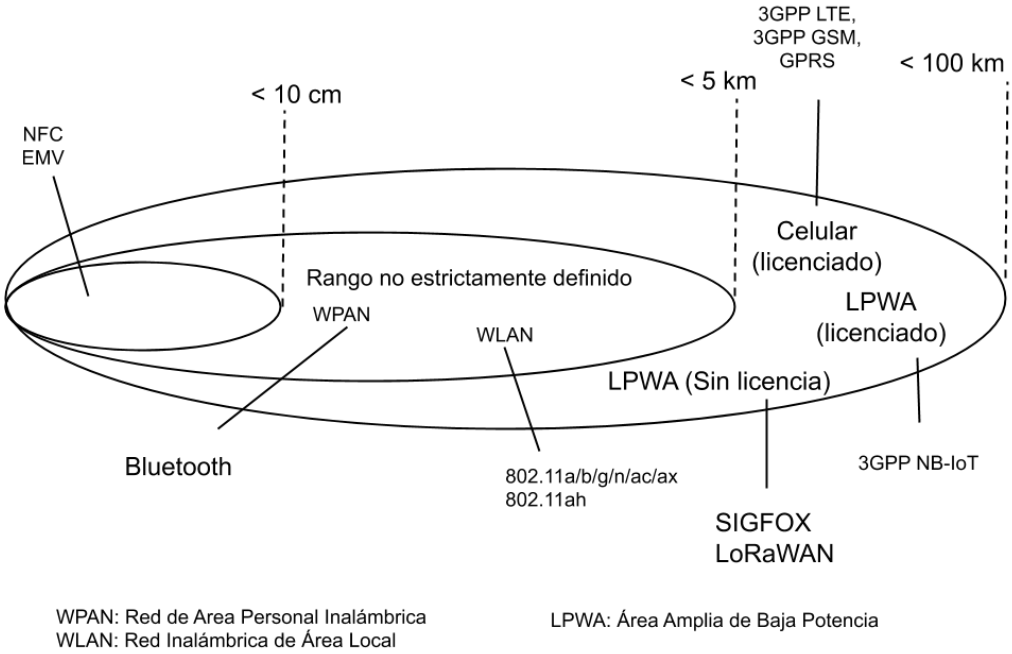


Figura 5 Tecnologías de acceso y distancias de transmisión estimadas.
Fuente: IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things

Una de las tecnologías que aquí destacan, según LoRa Alliance es LoRaWAN que aprovecha las características de largo alcance de la capa física LoRa para comunicación inalámbrica que permite un enlace de un solo salto entre el dispositivo final y una o varias puertas de enlace, sin embargo, presenta limitaciones en cuanto a transmisión de datos en tiempo real y un ancho de banda muy bajo (“Limitations of LoRaWAN”, 2020).

Algunas topologías ofrecen una estructura de conectividad flexible permitiéndoles ampliar las posibilidades de comunicación, como la topología punto a punto o la topología punto a multipunto.

b. Puerta de enlace y subcapa de backhaul

Los datos que obtiene el sensor deben enviarse a otro medio, a través del backhaul y ser transportados a la estación central. El Backhaul, es la tecnología que permite transportar los datos recogidos desde la puerta de enlace hacia la estación central (servidor). Esta comunicación entre medios está a cargo del Gateway.

Elegir una tecnología de backhaul depende de la distancia de comunicación y la densidad que se deben enviar.

Tabla 1 Backhaul

Tecnología	Tipo y Rango	Características Arquitectónicas
Ethernet	Cableado	Requiere un cable por sensor/grupo de sensores. Se adapta a la posición del sensor en un ambiente estable, el rango es limitado y el enlace es muy confiable
Wi-Fi (2.4 GHz, 5 GHz)	Inalámbrico	Puede conectar múltiples clientes a un solo punto de acceso, el rango es limitado. Se adapta a casos donde la alimentación del cliente no es un problema, gran ancho de banda disponible, pero puede haber interferencias de otros sistemas. El punto de acceso necesita un cable
Celular	Inalámbrica, varios kilómetros	Puede conectar un gran número de clientes; gran ancho de banda disponible; espectro licenciado, libre de interferencias

Fuente: IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things

WiFi y la tecnología celular ofrecen conexiones inalámbricas, mientras que Ethernet ofrece un medio alámbrico. La tecnología celular cuenta con un conjunto de características denominado Cellular Internet of Things (CIoT), introducidas en la Release 13 de la 3GPP. Se encuentran Extended Coverage GSM IoT (EC-GSM-IoT), enhanced Machine Type Communication (eMTC) y Narrowband IoT (NB-IoT) (Ericsson AB, 2016). También, en la Release 13 se define una nueva categoría, nombrada Category-M1 (Cat-M1) que logra mejorar la cobertura con un ancho de banda tan reducido como 1.4 MHz, además reduce el costo y la complejidad de los dispositivos eMTC, teniendo como objetivo mejorar la cobertura en al menos 15dB en comparación a la Categoría (Cat-1), como muestra la figura 6 (3GPP, 2013).

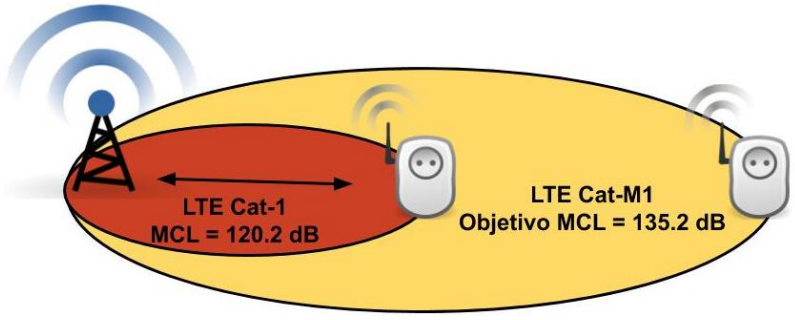


Figura 6 Objetivo de la mejora en la cobertura de LTE Cat-M1
Nota 2 Donde MCL (Maximum Coupling Loss) indica la medida de la cobertura.

Según la GSM Association, (2018), se prevé que un subconjunto de las bandas compatibles con 3GPP Release 13 sean usadas para Cat-M1, estas son las bandas 1, 2, 3, 4, 5, 12, 13, 20, 25, 26, 28 (p. 10).

En cuanto a EC-GSM-IoT se extiende a áreas donde no hay cobertura LTE, se despliega sobre las bandas GSM, proporciona un ancho de banda de 200 kHz.

En el caso de NB-IoT existen 3 modos de operación (figura 7), el modo Autónomo que usa una portadora GSM como portadora NB-IoT, habilitando su uso en 900 MHz o en 1800 MHz. El modo dentro de Banda en el que una banda de frecuencia portadora LTE se asigna para ser usada como una frecuencia NB-IoT. En este caso, es el proveedor de servicios quien realiza esta asignación. El otro modo es Banda de Guarda, en este caso una portadora NB-IoT se coloca entre bandas WCDMA o LTE, por lo que se da la coexistencia entre bandas LTE y NB-IoT (Barton. et al., 2017).

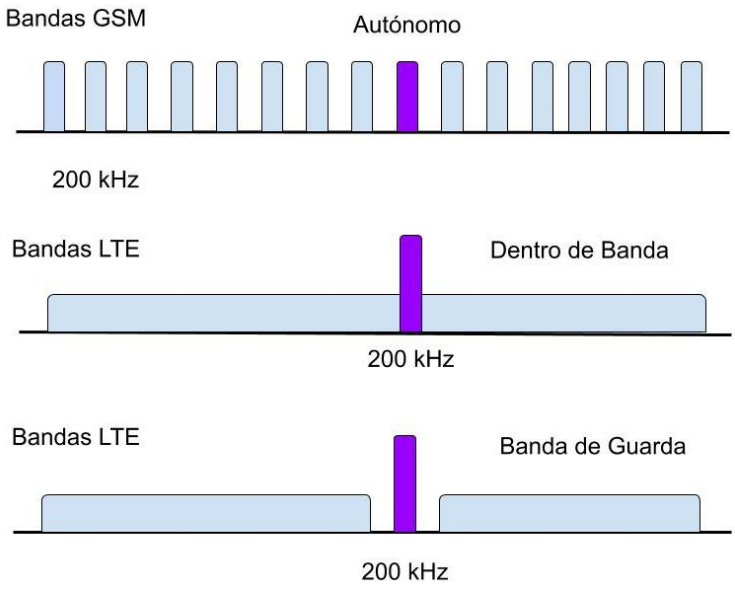


Figura 7 Modos de Operación NB-IoT
Fuente: IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things

c. Subcapa transporte de red

Comúnmente las implementaciones son flexibles y con múltiples vías de comunicación debido al uso del protocolo IP, que permite intercambiar información entre medios diferentes. Esto lleva a que los protocolos de transporte creadas por encima de IP puedan ser aprovechados para determinar si la red debe controlar la entrega de los paquetes de datos usando TCP o si es mejor usar UDP para la tarea de control.

d. Subcapa de gestión de red IoT

Los protocolos IP, TCP y UDP pueden llevar la conectividad a las redes IoT, mientras los protocolos de capa superior pueden transmitir los datos entre los objetos inteligentes y otros sistemas.

En este punto se tocan protocolos de transmisión.

Naik, (2017), considera en su investigación cuatro protocolos IoT como los más importantes: AMQP, HTTP, CoAP y MQTT (p. 1). AMQP es un protocolo binario M2M para mensajería corporativa que soporta la arquitectura solicitud/respuesta y publicación/suscripción. Usa TCP como protocolo de transporte. Requiere un encabezado fijo de 8-bytes con pequeñas cargas cuyo tamaño máximo depende del intermediario/servidor o la tecnología programada. En su funcionamiento requiere que el editor o consumidor cree un "intercambio" con un nombre dado y luego transmite ese nombre. Los editores y consumidores usan ese nombre para descubrirse entre sí. Luego, el consumidor crea una cola conectándola al intercambio. Un proceso llamado "vinculante" verifica la que los mensajes recibidos por el intercambio coincidan con la cola AMQP.

En el caso de HTTP es un protocolo de mensaje predominante en la web, soporta la arquitectura web RESTful de solicitud/respuesta, usa Identificador de Recursos Uniforme (URI). El servidor envía datos a través del URI y el cliente recibe estos datos a través de un URI particular. Está basado en texto, el tamaño de las cargas útiles del encabezado y mensaje dependen del servidor web o la tecnología de programación. Usa TCP como protocolo de transporte.

CoAP

Bormann, Hartke y Shelby (2014), definen CoAP como un protocolo para aplicación restringida para el internet de las cosas. Controla el intercambio de mensajes sobre UDP basándose en la arquitectura REST usando métodos básicos tales como GET, POST, PUT y DELETE para acceder a diversos recursos de internet a través de servicios web. Emplea el método cliente/servidor y la comunicación solicitud/respuesta.

Dentro de sus principales características, destacan:

- Intercambio asíncrono de mensajes.

- Encabezado overhead reducido.

El formato del mensaje CoAP contiene en la cabecera la versión del protocolo CoAP, el tipo del mensaje y la longitud del campo del token, como se ve en la figura 8. Los mensajes son identificados por un Message ID, para detectar posibles mensajes duplicados y garantizar la confiabilidad en la comunicación.

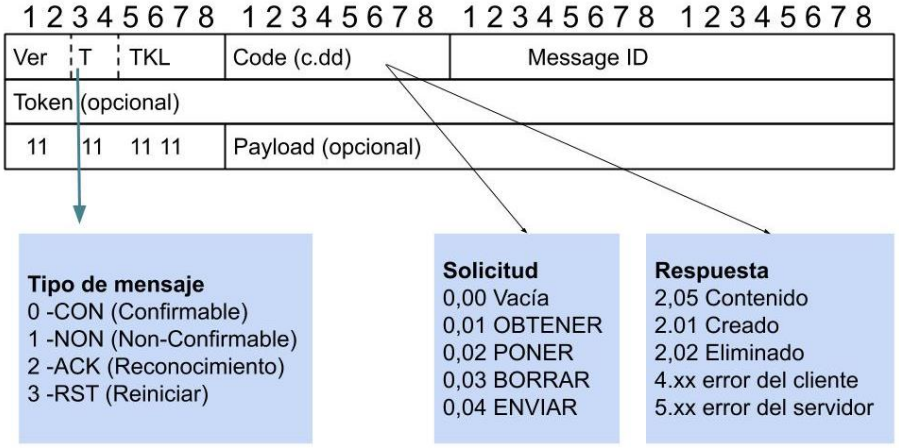


Figura 8 Formato del mensaje CoAP
Fuente: https://tools.ietf.org/pdf/rfc7252

Franekova & Peniak (2016) encontraron que CoAP cumple con altas expectativas permitiendo la integración general entre sistemas embebidos y sistemas en la nube al permitir la traducción directa entre HTTP/REST y las instancias de CoAP empleando una puerta de enlace IoT que permite equilibrar las diferentes restricciones de rendimiento de los sistemas integrados (p. 204).

MQTT

Es un protocolo ligero de la capa de aplicación para dispositivos IoT. Ha sido estandarizado por la Organization for the Advancement of Structured Information Standards (OASIS). Se basa en el modelo Publicar/Suscribir. Los mensajes se publican en el broker y los receptores pueden suscribirse al broker para recibir los mensajes. No es necesario que los publicadores y suscriptores se conozcan, pues el broker se encarga de gestionar los datos.

La figura 9 muestra la arquitectura de MQTT, donde el broker es un servidor MQTT que ofrece una interfaz a los clientes. El broker debe ser accesible para todos los clientes y tener suficientes recursos como almacenamiento, ancho de banda, etc. Los clientes pueden tanto publicar como suscribirse. El cliente y el broker usan los temas para identificar un recurso en particular.

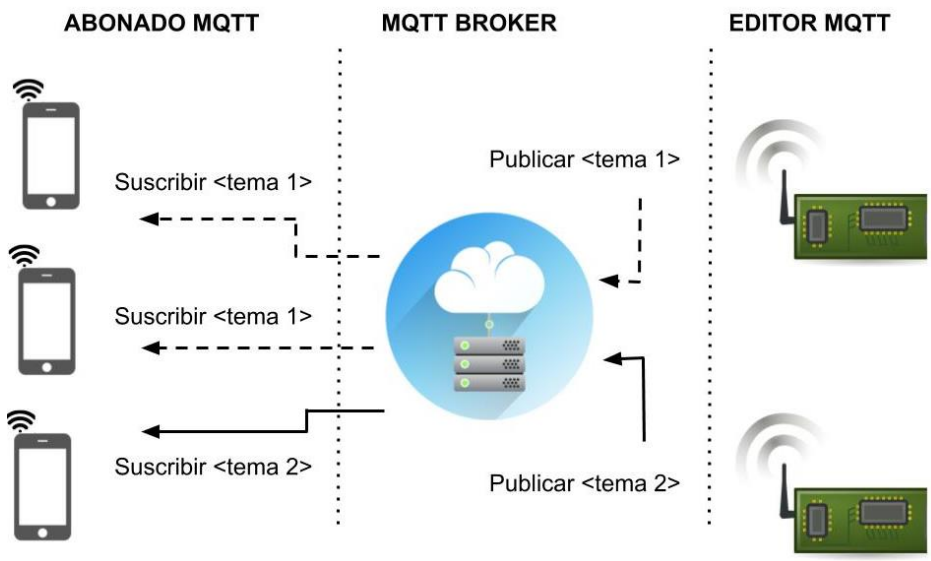


Figura 9 Arquitectura MQTT
Fuente: http://docs.oasisopen.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

La transmisión de los mensajes tiene tres niveles de seguridad QoS (Quality of Service). Nivel QoS 0, que envía a lo más una vez el mensaje y no requiere confirmación de recepción, nivel QoS 1, que envía el mensaje al menos una vez y nivel QoS 2 que envía el mensaje exactamente una vez.

Capa 3: de aplicaciones

Dentro de esta capa se encuentran aplicaciones analíticas que recopilan información de los objetos inteligentes, la procesan y muestran la información y aplicaciones que controlan el comportamiento del objeto inteligente. Además, se analiza la red, pues puede ser afectada por pérdidas o degradación en la conectividad de los objetos inteligentes conectados (Barton et al. 2017, “Applications and Analytics Layer”).

Gestión de datos IoT y bloque de cómputo

También se debe considerar la cantidad de datos que se van a analizar y su sensibilidad temporal. Esto permite determinar si la computación en la nube es suficiente o si la computación en la niebla o en el borde mejorarían la eficiencia del sistema. Diversos problemas como lo limitado del ancho de banda en la última milla, la alta latencia, la confiabilidad y disponibilidad del backhaul o el volumen de los datos pueden ser tratados si se identifica a qué nivel de las siguientes capas trabajar.

1) Computación en el borde

En este nivel los datos se procesan en los sensores y dispositivos de IoT.

2) Computación en la niebla

En este nivel, los datos se procesan previamente correlacionándose con otras entradas para producir información relevante.

3) Computación en la nube

El lugar donde se procesan los datos se ubica fuera de los objetos inteligentes, comúnmente en la nube, centralizándose su procesamiento.

Siendo la latencia menor en el procesamiento de los datos si se trabaja en la capa de borde y mayor si tienen que ser enviados a la nube para su procesamiento como se ve en la figura 10.

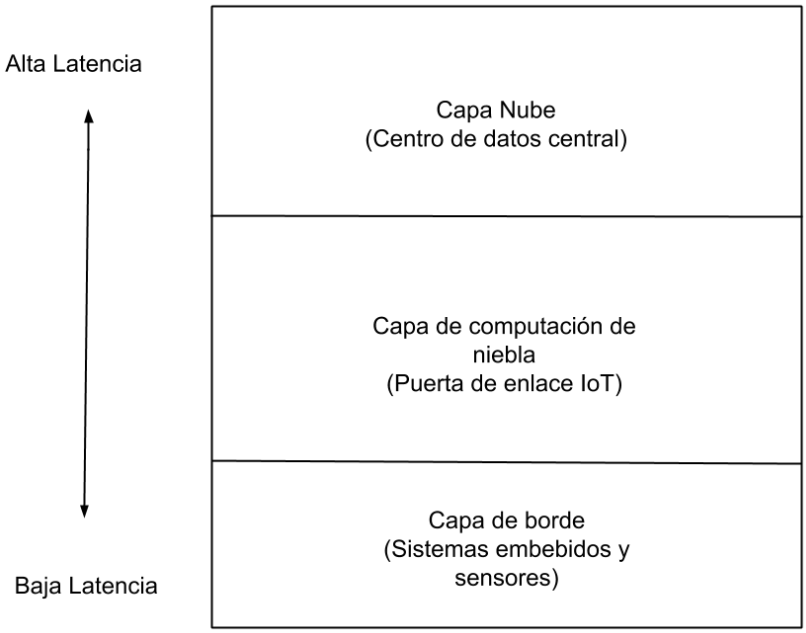


Figura 10 Jerarquización de capas de un sistema IoT
Fuente: IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things

III. METODOLOGÍA

3.1 Diseño de investigación

La metodología propuesta a desarrollar en la presente tesis para el logro de los objetivos fue el siguiente:

- Se identificó una necesidad (el problema).
- Se definió un objetivo general y objetivos específicos para permitir solucionar el problema planteado.
- Se investigó antecedentes relacionados al problema.
- Se analizó las potenciales soluciones tecnológicas al problema general y se seleccionaron las herramientas más adecuadas.
- Se diseñó la solución tecnológica.
- Se implementó la solución tecnológica en un prototipo.
- Se testeó el prototipo.
- Se establecieron conclusiones respecto a la solución planteada.

3.2 Descripción de la metodología

La figura 11 muestra las ubicaciones actuales de los puntos de monitoreo del sistema de alerta temprana de huaicos del Radio Observatorio de Jicamarca.

Figura 11 Sistema de alerta temprana de huaicos del Radio Observatorio de Jicamarca
Fuente: Radio Observatorio de Jicamarca

La estación Huaycoloro cuenta con cinco sensores (uno de nivel, dos de presencia, uno de vibraciones y una cámara), mientras que la estación Rio Seco cuenta con cuatro sensores (uno de nivel, dos de presencia y uno de vibración). Los sensores se encuentran alejados de las estaciones y conectados por medio de cables. Se necesita que cada sensor tenga su propia conexión inalámbrica, pues las conexiones alámbricas suelen deteriorarse con el paso del tiempo y las exigencias del ambiente.

Se propuso una red local que consistió de instrumentos de red inalámbricos que se conectaron a una estación central que tuvo conexión directa a internet. Esta propuesta surgió porque actualmente el sistema de alerta temprana de huaicos se conecta a internet por medio de las instalaciones del Radio Observatorio de Jicamarca. Por esta razón, frente a la ocurrencia de un huaico es muy probable que ocurra un corte de energía en las instalaciones del Radio Observatorio de Jicamarca, lo cual dejaría a las estaciones de monitoreo sin conexión a internet.

El Instituto Geofísico del Perú (IGP) proyecta extender el Sistema de Alerta Temprana de Huaicos (SATH) instalado en la Quebrada de Jicamarca, así como replicar el mismo SATH en diferentes zonas del país afectadas por los huaicos. Estos lugares suelen tener una geografía accidentada y requieren de sensores instalados muy próximos al cauce del huaico. Esto plantea la necesidad de que los enlaces entre los sensores y la estación central tengan cierta distancia, de manera que la estación central no pueda ser afectada por el paso del huaico y que los componentes deberían ser de bajo costo, por las condiciones hostiles del clima que podrían dañarlos, de manera que no representen un alto costo en pérdidas.

3.2.1 Implementación de la investigación

De acuerdo a lo visto anteriormente, la red a implementar debe cumplir con los siguientes requerimientos.

- Poder conectarse a internet sin depender de la conexión a la red del Radio Observatorio de Jicamarca.
- Los enlaces deben tener un alcance no menor a 700 m.
- La red de instrumentos inalámbricas debe ser de bajo costo.

3.2.1.1 Cálculos y selección de tecnologías IoT

A continuación, se discute y analiza diferentes tecnologías con el fin de realizar la selección.

Como se vio en las referencias, la tabla 2 muestra protocolos IoT para transferencia de datos. De todos los protocolos mostrados en la tabla, solo CoAP ha sido diseñado para aplicaciones IoT, los demás, HTTP, AMQP y MQTT se fuerzan o adaptan para aplicaciones en este campo, no obstante, MQTT se ha adaptado bien a estas necesidades por lo que es conveniente emplearlo junto con CoAP.

Tabla 2 Selección de Protocolos para IoT

Criterio	**HTTP**	**AMQP**	**CoAP**	**MQTT**
Modelo de interacción	Cliente/Servidor	solicitud/respuesta y publicación/suscripción	Similar al modelo Cliente/Servidor	Publicación/Suscripción
Protocolo de transporte	TCP	TCP	UDP	TCP
Adecuado para entornos restringidos	Fue diseñado para su uso en páginas web	Protocolo binario M2M para mensajería corporativa	Cabecera reducida, diseñado para aplicaciones IoT	Protocolo ligero, requiere de un bróker cuya implementación es compleja

Fuente: Elaboración propia

Los protocolos de acceso, permiten conectar y comunicar los nodos.

Se necesitan protocolos de acceso para la red local y para la conexión a internet. La tabla 3 muestra protocolos para el acceso a la red local. Se encuentra Zigbee, cuyo alcance es de 80 metros, también Bluetooth Low Energy, cuyo alcance es de 100 m, por lo que quedan descartados al no cumplir los requerimientos. También se consideró Lora que, a pesar de tener un alcance de 10 a 20 km, requiere de un Gateway, lo que hace que se eleva el costo. Lora opera en una porción del espectro licenciado para el Perú, además presenta limitaciones para la transferencia de datos en tiempo real y tiene un ancho de banda muy bajo, por lo que se termina descartando. Se considera también WiFi IEEE 802.11 LR, que es un protocolo exclusivo de la compañía de semiconductores ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. que distribuye la serie de microcontroladores ESP32. La distancia que cubre es de 1.5 km, trabaja en la frecuencia de 2.4 GHz, por lo que se adecúa a las necesidades del sistema.

Tabla 3 Selección de protocolos de acceso

Criterio	Zigbee	Bluetooth Low Energy	WiFi IEEE 802.11 LR	Lora
Espectro	2.4 GHz	2.4 GHz	2.4 GHz	433 MHz, 779 - 787 MHz, 863-870 MHz y 902-928MHz
Escenario de uso	Objetos inteligentes y sensores que tienen bajo ancho de banda y requieren baja potencia	Para aplicaciones de baja potencia, indoor	Modulación diferente al WiFi tradicional que sacrifica ancho de banda para conseguir un mayor alcance	Conexiones a grandes distancias y para redes de IoT, requiere Gateway costoso
Alcance	80 m	100 m	1.5 km (Antena externa) y 300 m (Antena PCB)	10 a 20 km
Velocidad máxima de transferencia	250 kbps	1,4 Mbps	1/2 Mbits y 1/4 Mbits	50 kbps
Consumo de potencia	100 mW	0.01 W–0.50 W	90 mW	100 mW

Fuente: Elaboración propia

Para el siguiente paso, se requiere contar con la conexión a internet.

Para tal fin, la tabla 4 presenta algunas tecnologías pertenecientes al grupo de Cellular Internet of Things.

En el caso de las tecnologías GPRS y EDGE, las cuales dependen de la cobertura 2G, permiten a dispositivos inalámbricos conectarse a internet a través de las redes GSM, proporcionando una velocidad máxima de transferencia de 50 kbps para GPRS y 150 kbps para EDGE.

En cuanto a la tecnología NB-IoT presenta tres modos de operación, lo que significa que su disponibilidad depende del operador móvil, pudiendo implementarse en redes 4G. Por otro lado, proporciona una velocidad de transferencia de 60 kbps.

En el caso de la tecnología Cat-M1, opera en redes 4G. Cat-M1 proporciona una velocidad de transferencia de 1 Mbps.

Se opta por la tecnología Cat-M1 considerando que a futuro se requiera mayor transferencia de datos y es esta tecnología la que proporciona mayor transferencia de datos. Además, al ser una tecnología pensada para casos de usos móviles, si bien el Punto de Acceso debe permanecer en una ubicación fija, deja abierta la posibilidad de que el sistema pueda adaptarse otros sistemas como vehículos aéreos no tripulados sin afectar el rendimiento.

Tabla 4 Selección de backhaul

Criterio	**NB-IOT**	**CAT-M1**	**GPRS Y EDGE**
Espectro	Autónomo Dentro de banda Bandas de Guarda	Opera en redes ya existentes dentro de la red 4G	Opera en redes 2G
Escenario de uso	Dispositivos que generan un tráfico de datos bajo	Casos de uso móvil	Dispositivos inalámbricos que quieran conectarse a internet
Alcance	Mayor a 4G	Mayor a 4G	Depende de la cobertura 2G
Velocidad máxima de transferencia	60 kbps	1 Mbps	50 kbps/150 kbps

Fuente: Elaboración propia

Se usó la aplicación Network Cell Info Lite disponible para el sistema operativo Android que escaneó y mostró la cobertura de la red 4G disponible en la Quebrada de Jicamarca. La figura 12 mostró la cobertura de Entel en la banda 1.

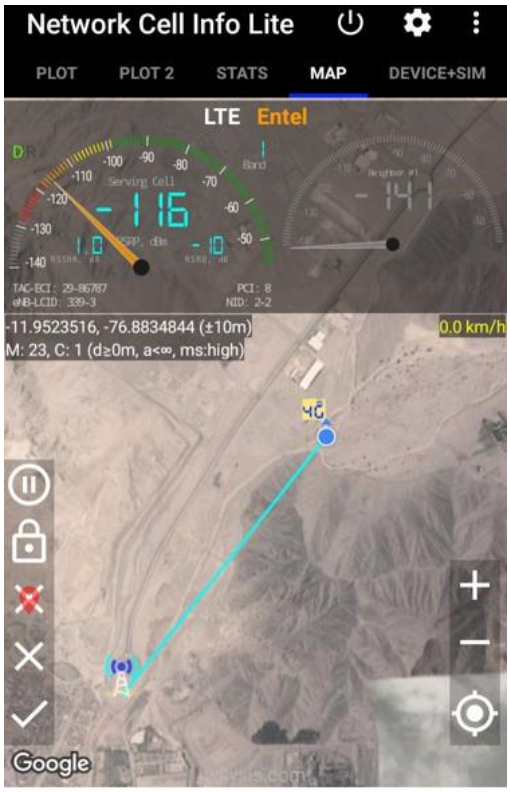


Figura 12 Cobertura 4G LTE ENTEL en la Quebrada de Jicamarca
Fuente: Elaboración propia

La figura 13 mostró la cobertura de Claro en la banda 28.

Figura 13 Cobertura 4G LTE CLARO en la Quebrada de Jicamarca
Fuente: Elaboración propia

La figura 14 mostró la cobertura de Movistar en la red 2G.

Figura 14 Cobertura 2G MOVISTAR en la Quebrada de Jicamarca
Fuente: Elaboración propia

Se seleccionó la operadora Claro por tener cobertura con la tecnología que soporta el protocolo Cat-M1, es decir, la banda 28 y ser operativo durante el desarrollo de la tesis.

Hasta este punto, se cuenta con el protocolo para la red local, con la tecnología de acceso tanto para la red local como para la conexión a internet. Ahora se selecciona el hardware para el desarrollo de la red. La tabla 5 muestra las plataformas de desarrollo que se han considerado. El Arduino Yun es el sistema embebido que se emplea actualmente en el Sistema de Alerta Temprana de Huaicos. Tiene un sistema operativo que es Linux y su costo es de 22.93 dólares. Muy por el contrario, se propone el ESP8266-DevKitC, una plataforma de desarrollo diseñado para aplicaciones IoT, pero su hardware es limitado en muchos aspectos, como que cuenta con el protocolo WiFi tradicional. Su costo es de 8 dólares. Por otro lado, como ubicándose entre las prestaciones que brinda el Arduino Yun y lo simple que es el ESP8266-DevKitC, se propone el ESP32-DevKitC, que cuenta con el protocolo WiFi IEEE 802.11 LR, que es exclusivo de este microcontrolador, brinda la posibilidad de crear enlaces de 1 km. Además, cuenta con dos cores, uno dedicado a la parte de procesamiento y el otro dedicado al WiFi. Su costo es de 10 dólares, por lo que se justifica el bajo costo.

Tabla 5 Selección de Microcontrolador

Criterio	ARDUINO YUN	ESP8266-DevKitC	ESP32-DevKitC
Unidad de procesamiento	Atheros AR9331 (Linux)	Tensilica L106 32 bit	Xtensa Dual-Core 32-bit LX6
Entorno de desarrollo	Arduino IDLE	Arduino IDLE	Arduino IDLE
Arquitectura	MIPS	32 bit L106 Rick	Dual Core 32 bits
Voltaje de operación	5 V	3.3 V	3.3 V
Alimentación	5 V	5 V	5 V
Entradas digitales	20	13	34
Ethernet	IEEE 802.3 (10/100Mbit /s)	-	Ethernet MAC
WiFi	IEEE 802.11b/g/n (2.4GHz)	802.11b/g/n Wi-Fi Direct (P2P) Soft-AP	802.11 b/g/n (hasta +20dBm) WEP, WAP MODO LR
Cores	1	1	2
Precio	$22.93	$8	$10

Fuente: Elaboración propia

Adicionalmente, cuenta con una presentación con pigtail, que permite conectar una antena externa, como se observa en la figura 15.

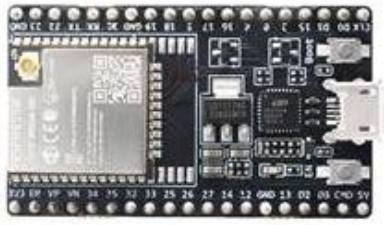


Figura 15 Módulo ESP32 con pigtail para antena externa
Fuente: www.google.com.pe

Para tener conexión a internet, se consideran los módulos inalámbricos de telefonía móvil que se muestran en la tabla 6. La SIM800L permite conectarse a 2G por medio de las bandas 850/1900 MHz, que son bandas para redes 2G disponibles en el Perú, pero queda descartado por solo contar con tecnología 2G. El módulo SIM7000E, el cual está diseñado para conexión a protocolos IoT de 4G, que son Narrow Band y Cat-M1, que es lo que se pretende usar. También permite conectarse a 2G por medio de las bandas 900/1800 MHz, pero no están disponibles en el Perú. El módulo GSM 4 CLICK permite conectarse a 2G por medio de las bandas 850/1900/900/1800 MHz, pero su velocidad de transferencia es menor al de los otros módulos en el uplink. El EVK-R412M basado en el módulo SARA-R412M-02B opera en 2G Quad-band y bandas LTE, pero su costo es muy elevado.

Tabla 6 Selección de módulo inalámbrico de telefonía móvil

Criterio	SIM800L	SIM7000E	GSM 4 CLICK	EVK-R412M
2G	SÍ (Quad-band 850/900/1800/1900MHz)	Sí (GPRS/EDGE 900MHz/1800MHz)	Sí GSM Quad-band 850/1900/ 900/1800 MHz	Sí Quad-band 850/900/ 1800/1900 MHz
3G	No	No	No	No
4G	No	FDD-LTE B3/B8/B20/B28	No	LTE FDD Bands 2, 3, 4, 5, 8, 12, 13, 20, 28
Velocidad de transferencia	Max. 85.6 kbps downlink/uplink	Uplink up to 375kbps, Downlink up to 300kbps (cat m1)	42.8 for the uplink and 85.6 kbps for downlink	Up to 375 kb/s UL up to 300 kb/s DL (cat m1)
Suministro voltaje	4V	3.8 V	3.8V	9-18V
Costo	$13.90	$39.90	$49.90	$399

Fuente: Elaboración propia

El módulo SIM7000E se puede observar en la figura 16.

Figura 16 Módulo SIM7000E
Fuente: www.google.com

Se seleccionaron los protocolos CoAP y MQTT para la transferencia de datos, el protocolo WiFi IEEE 802.11 LR como protocolo de acceso, el protocolo Cat-M1 como backhaul, el ESP32 como microcontrolador y el módulo SIM7000E para la conexión a internet.

Para la comunicación con la red móvil, se emplearon comandos AT. La tabla 7 muestra una lista de comandos a tener en cuenta.

Tabla 7 Comandos AT

Comando AT	Función
AT+CGDCONT	Define el contexto del PDP
AT+CRRCSTATE	Consulta estado RRC
AT+CGREG?	Estado del registro de la red
AT+CPSI?	Solicitar información del sistema EU
AT+COPS?	Selección de operador
AT+HTTPTERM	Termina el servicio http
AT+HTTPINIT	Inicializa el servicio http
AT+HTTPPARA	Establecer el valor de los parámetros http
AT+HTTPACTION	Método de acción de http
AT+HTTPREAD	Lee la respuesta del servidor http
AT+CSQ	Informe de calidad de señal
AT+CFUN	Establecer la funcionalidad del teléfono
AT+CPIN	Ingrese su pin
AT+CPMS	Almacenamiento preferido de mensajes
AT+GSN	Solicitar IMEI
AT+CGNSPWR	Control de potencia GNSS
AT+CREG	Registro de red
AT+CNSMOD=1	Informe automático del modo del sistema de la red
AT+CNBP	Establece el estado de la preferencia de la banda
AT+CMNB?	Selección entre cat-m1 y NB
AT+CBANDCFG	Configura la banda Cat-M1 o NB
AT+CNMP?	Selección de modo preferido
AT+CAPNMODE	Selecciona el modo de aplicación configurar APN
AT+CGACT	PDP contexto activar o desactivar
AT+SAPBR	Configuración del portador para aplicaciones basadas en ip
AT+CGNAPN	Obtener APN de la red en Cat-M1 o NB
AT+CGPADDR	Mostrar dirección PDP
AT+CBC	Batería cargada
AT+CGNSINF	Información de navegación GNSS analizada a partir de sentencias NMEA
AT+CIPSTATUS	Consultar el estado actual de la conexión
AT+CIPSHUT	Desactivar el contexto PDP GPRS
AT+CIPMUX	Iniciar la conexión IP múltiple
AT+CIPRXGET	Obtener datos de la red manualmente
AT+CIPSTART	Iniciar la conexión TCP o UDP
AT+CIPSEND	Enviar datos a través de una conexión TCP o UDP

Fuente: SIM7000 Series_AT Command Manual_V1.04

La simulación del radioenlace entres el Punto de Acceso y las Estaciones se realiza para tener la referencia de que los radioenlaces son viables.

En base a que Espressif Systems declara que el módulo ESP32 cuenta con el modo LR patentado por ellos, el cual puede alcanzar un rango de línea de visión de 1km, esto debido a que presenta una mejor sensibilidad de recepción (4 dB de ganancia), una capacidad anti interferente y una distancia de transmisión más larga que el modo 802.11b tradicional (“ESP-IDF Programming Guide”, 2020).

Se empleó el software Radio Mobile. Radio Mobile utiliza el modelo de propagación ITM (Irregular Terrain Model) el cual se basa en la teoría electromagnética y el análisis estadístico tanto de las características del terreno como de las mediciones de radio. Predice la atenuación media de una señal de radio en función de la distancia y la variabilidad de la señal en el tiempo y el espacio (Hufford, Kissick y Longley, 1982).

Este modelo tiene las siguientes limitaciones:

-Frecuencia: 20 MHz – 20 GHz

-Distancia: 1 – 2000 km

-Altura de la antena: 0,5 – 3000 m

De acuerdo a la bibliografía revisada, el modelo hace predicciones de área para aplicaciones de diseños preliminares. Además, el modelo no proporciona predicciones para distancias menores a 1 km, pero Radio Mobile proporciona una herramienta que permite visualizar los niveles de cobertura sobre el mapa por lo que distancias menores a 1 km pueden ser puntos de ubicación para las estaciones.

Para la simulación se consideraron los siguientes valores.

Tabla 8 Propiedades del Punto de Acceso

Punto de acceso	**Valores**
Potencia de Tx	20 dBm
Sensibilidad	-102 dBm
Tipo de antena	Dipolo
Ganancia de la antena	3 dBi
Pérdida de línea	-1.5 dB
Altura de la antena	5 m
Latitud	-11,9505972
Longitud	-76,87633056

Fuente: Elaboración propia

Tabla 9 Propiedades Estación Río Seco

Estación Río Seco	**Valores**
Potencia de Tx	20 dBm
Sensibilidad	-102 dBm
Tipo de antena	Dipolo
Ganancia de la antena	3 dBi
Pérdida de línea	-1.5 dB
Altura de la antena	2 m
Latitud	-11,94684167
Longitud	-76,86747778

Fuente: Elaboración propia

Tabla 10 Propiedades Estación Huaycoloro

Estación Huaycoloro	**Valores**
Potencia de Tx	20 dBm
Sensibilidad	-102 dBm
Tipo de antena	Dipolo
Ganancia de la antena	3 dBi
Pérdida de línea	-1.5 dB
Altura de la antena	6 m
Latitud	-11,95315833
Longitud	-76,88516389

Fuente: Elaboración propia

La figura 17 muestra la ubicación del Radio Observatorio de Jicamarca georreferenciado.

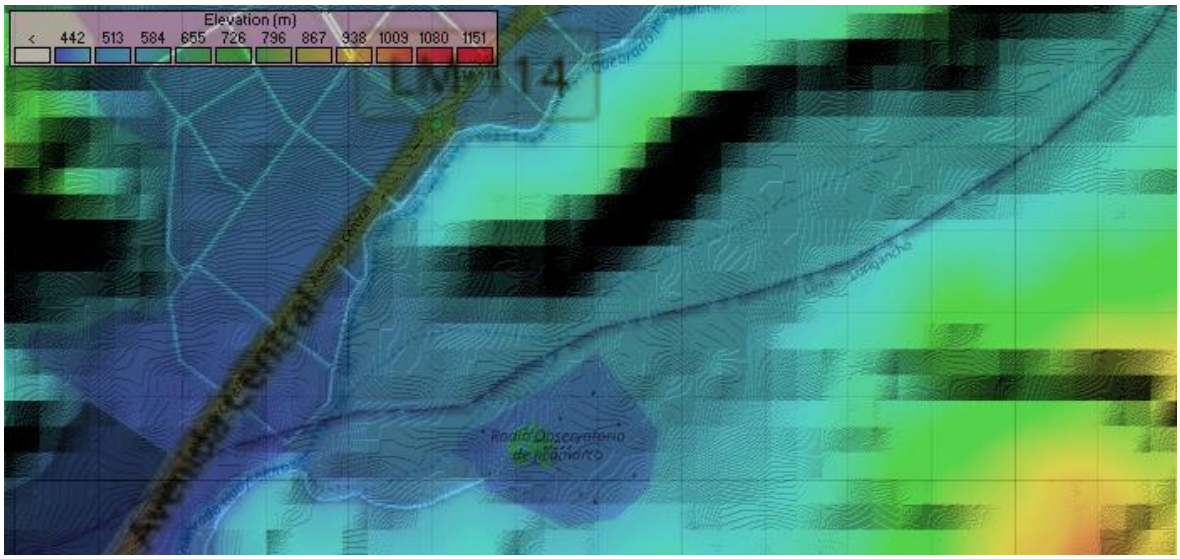


Figura 17 Mapa ubicación Radio Observatorio de Jicamarca
Fuente: Elaboración propia

En la figura 18 se puede observar los niveles de cobertura alrededor del Punto de Acceso, mostrando gran parte del área tiene niveles de entre -87 dBm a -91 dBm.

Figura 18 Enlaces Punto de Acceso – Estaciones
Fuente: Elaboración propia

Finalmente, las figuras 19 y 20 muestran que tanto la Estación Río Seco como la Estación Huaycoloro tienen un nivel de recepción relativo (Rx Relative) es positivo, con lo cual el nivel de recepción (Rx level) está por encima del umbral.

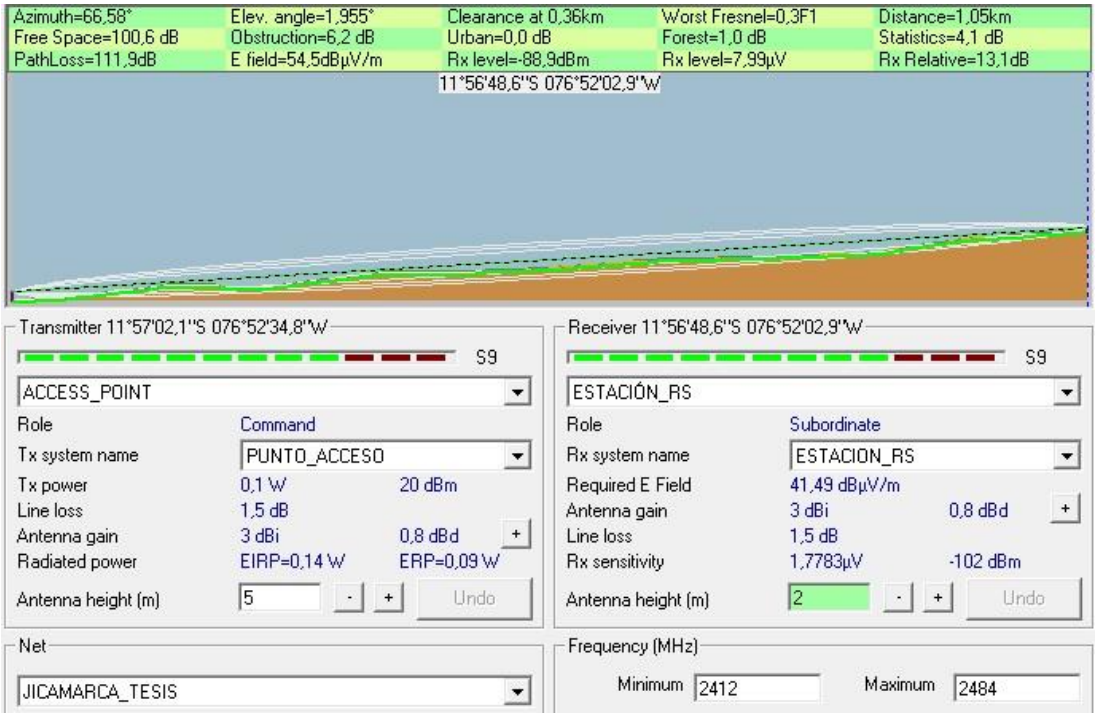


Figura 19 Línea de vista Punto de Acceso - Estación Río Seco
Fuente: Elaboración propia

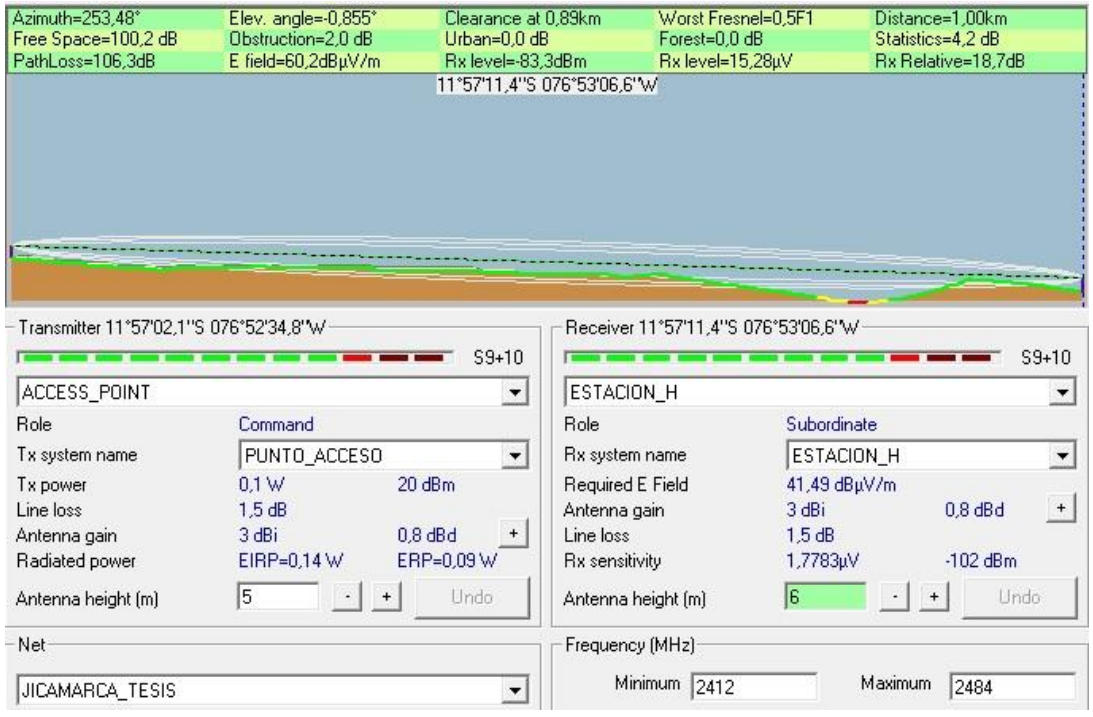


Figura 20 Línea de vista Punto de Acceso - Estación Huaycoloro
Fuente: Elaboración propia

3.2.1.2 Diagrama de bloques

En este apartado, se encuentran los diagramas de bloques del Punto de Acceso y de la Estación de Monitoreo.

El esquema general propuesto consiste en una red local que se conecta a la red de telefonía móvil. La red local consiste de la Estación Huaycoloro y Estación Río seco. Ambas se conectan al Punto de Acceso de forma inalámbrica en el que se implementa un servidor y una puerta de enlace para comunicar la red local con internet como se ve en la figura 21.

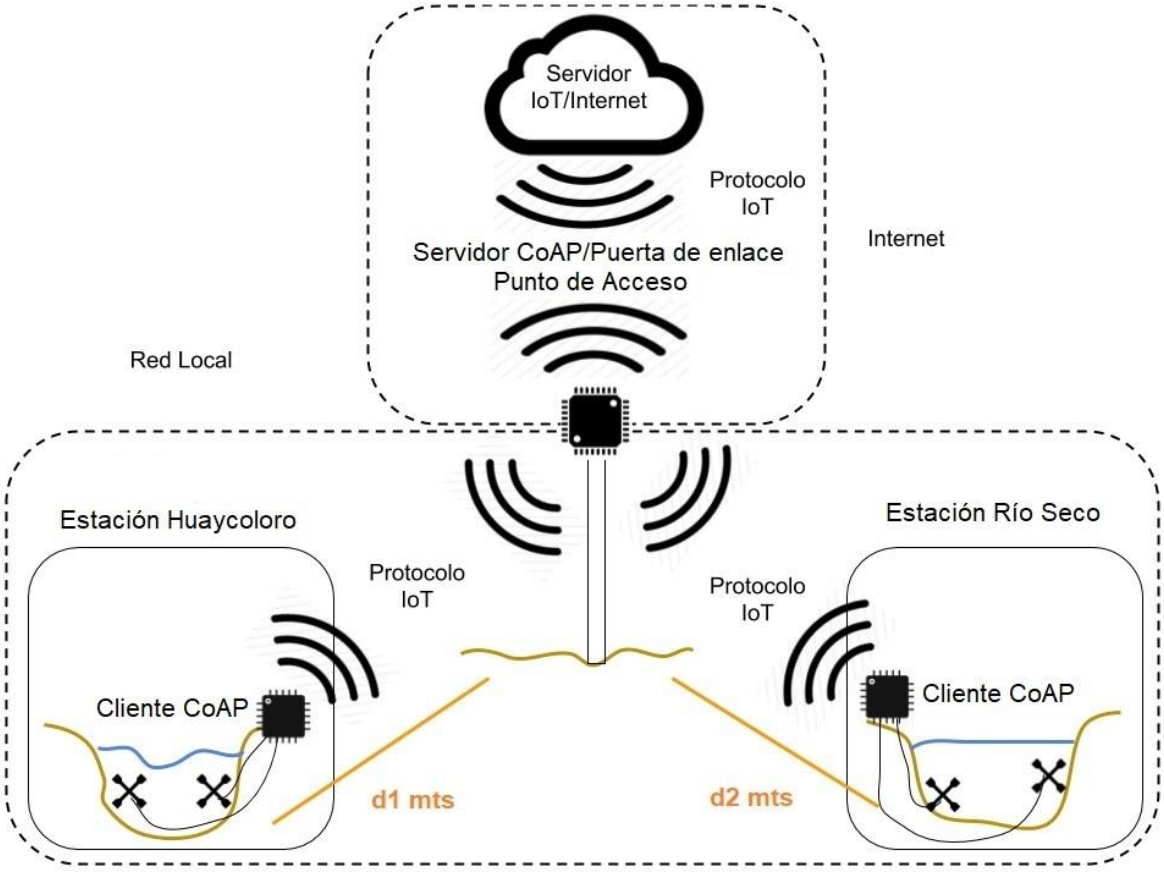


Figura 21 Esquema de la red propuesta
Fuente: Elaboración propia

I. Diagrama de Bloques de la Estación de Monitoreo

El punto de monitoreo, consiste de un nodo que está conformado por un módulo de desarrollo ESP32, en el que se implementa el módulo WiFi como tecnología de acceso, habilitando el modo de largo alcance WiFi IEEE 802.11 LR. Se implementa el cliente CoAP como protocolo de gestión de red IoT para la red privada local. Al módulo de desarrollo que hace la adquisición de datos se conecta una pantalla, que permite visualizar el flujo de datos. Finalmente se conecta también la antena WiFi de 2.4 GHz y un proveedor de energía. Esto se puede ver en el diagrama de bloques de la figura 22.

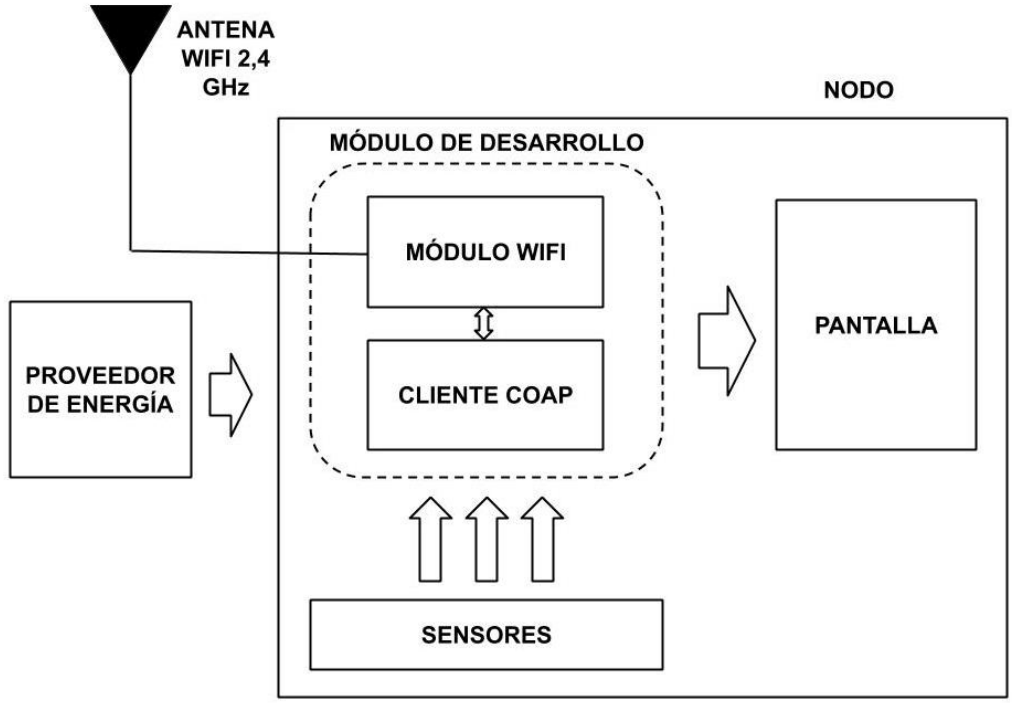


Figura 22 Diagrama de bloques Estación de Monitoreo
Fuente: Elaboración propia

II. Diagrama de Bloques del Punto de Acceso

Consiste del módulo de desarrollo ESP32 en el que se implementa el módulo WiFi como tecnología de acceso, habilitando el modo de largo alcance WiFi IEEE 802.11 LR para establecer conexión con las estaciones de monitoreo. Se implementa el servidor CoAP como protocolo de gestión de red IoT para la red privada local que recibe los datos enviados por las estaciones de monitoreo. Se implementa un cliente MQTT, que se comunica con el servidor CoAP, el cual le proporciona los datos que envía al bróker. El módulo de desarrollo cuenta con una antena WiFi 2,4 GHz para la comunicación inalámbrica con los nodos. El módulo de desarrollo se comunica con el módulo LTE. Al módulo LTE se le conecta una antena 4G que le permite acceder a la tecnología de acceso de radio móvil. El módulo de tecnología celular permite trabajar con el protocolo Cat-M1. De esta forma se conecta a internet a través de la red móvil, y envía los datos del Punto de Acceso a la nube.

Adicionalmente, se conecta una pantalla al módulo de desarrollo que permite visualizar los datos y un proveedor de energía para energizar el Punto de Acceso como muestra la figura 23.

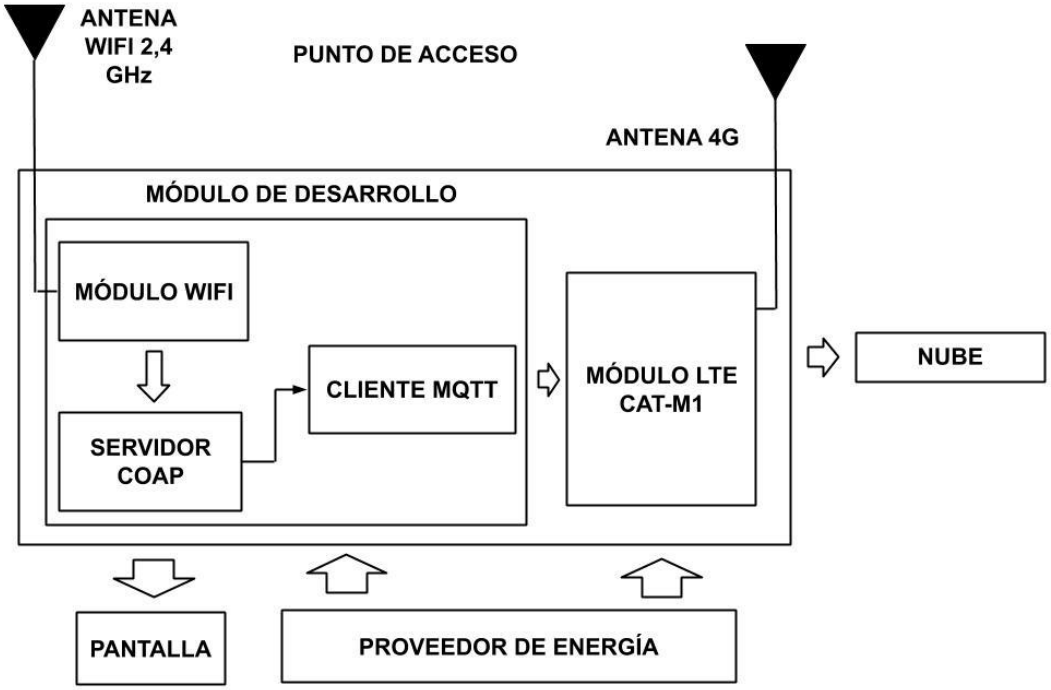


Figura 23 Diagrama de bloques Punto de Acceso
Fuente: Elaboración propia

3.2.1.3 Diagrama de flujo

1. Diagrama de flujo de la Estación de Monitoreo

Esta etapa comienza con la configuración de los parámetros de WiFi, los parámetros para cliente CoAP. Se habilita el modo de largo alcance para dar inicio al enlace WiFi y al protocolo de transferencia CoAP. Si la Estación de Monitoreo establece el enlace wifi, toma el dato de entrada, se genera un método de transferencia de datos Request Put de CoAP que contiene el dato de entrada y lo envía al Punto de Acceso. De lo contrario, mientras el estado de WiFi sea no conectado, realiza hasta cinco intentos de reconexión (tries), de no lograr la reconexión, retorna a realizar los intentos. Esto se puede ver en la figura 24.

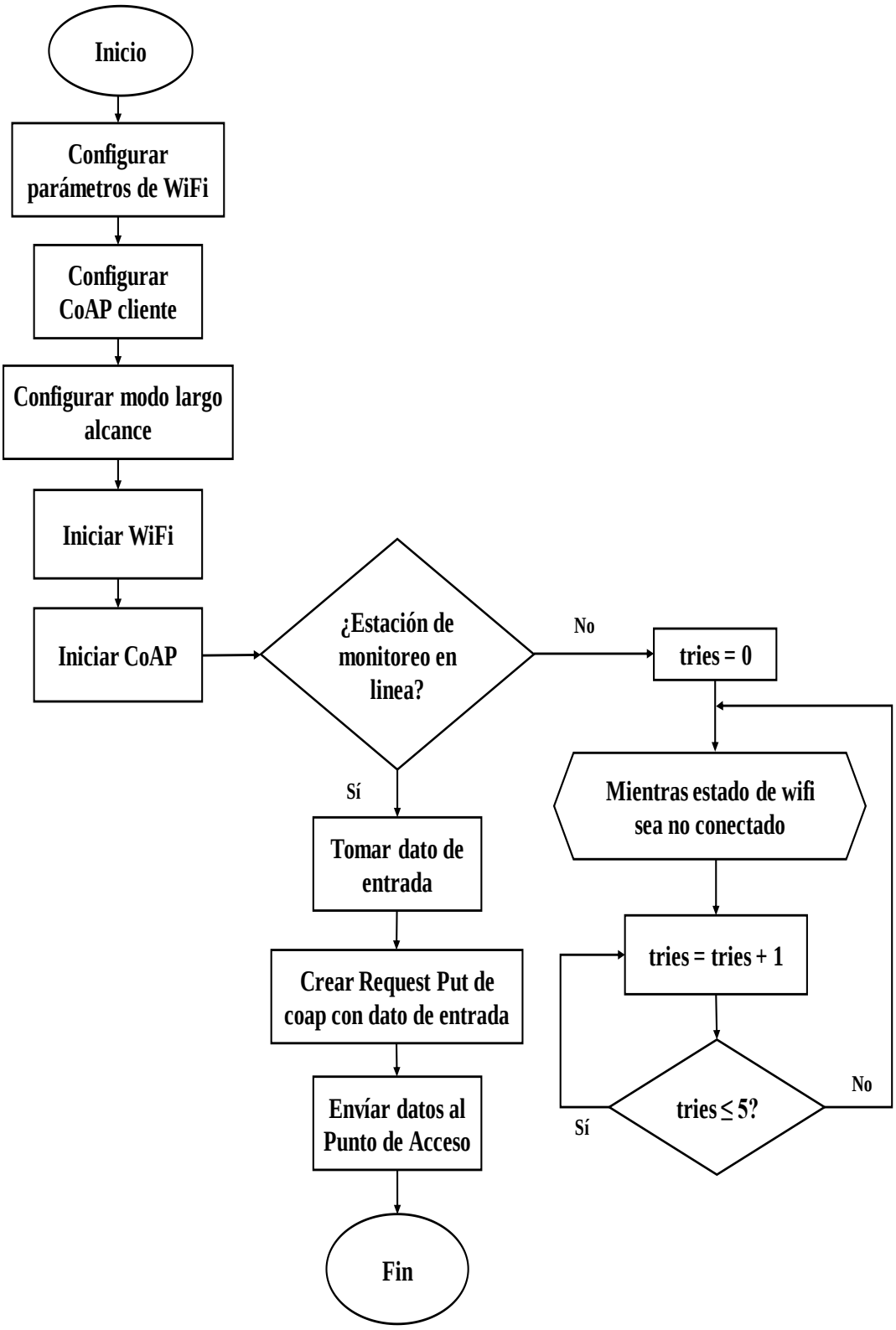


Figura 24 Diagrama de flujo de la Estación de Monitoreo
Fuente: Elaboración propia

2. Diagrama de flujo del Punto de Acceso

Se comienza por establecer los tópicos, que son las rutas en la nube donde se alojaran los datos que se envían. Se establecen los parámetros de WiFi, los parámetros para el servidor CoAP, se crean dos variables para controlar el tiempo referente a la recepción de datos de la red local (coap_timeout) y al envío de datos a la nube (mqtt_timeout). Se configura el modo de largo alance, se inicia el enlace WiFi. El Punto de Acceso recibe el dato de la Estación de monitoreo. Se enciende el módulo SIM7000E, el módulo ESP32 y el módulo de SIM7000E se comunican por su puerto serial. En caso no se obtenga el estado de la red móvil, se reintenta hasta lograr la conexión, luego se establece la conexión a través de Cat-M1por medio de comandos AT. Coap_timeout y mqtt_timeout se actualizan con el tiempo del sistema. La variable coap_timeout se actualiza cada vez que el servidor CoAP recibe un dato, si no recibe el dato en menos de dos segundos, se actualiza con el tiempo del sistema y permanece en estado de recepción. Los datos recibidos por el servidor CoAP son almacenados en un buffer. Cuando se establece la conexión a la red móvil, el Punto de Acceso envía los datos provenientes del servidor CoAP a la nube periódicamente. Esto se puede ver en la figura 25.

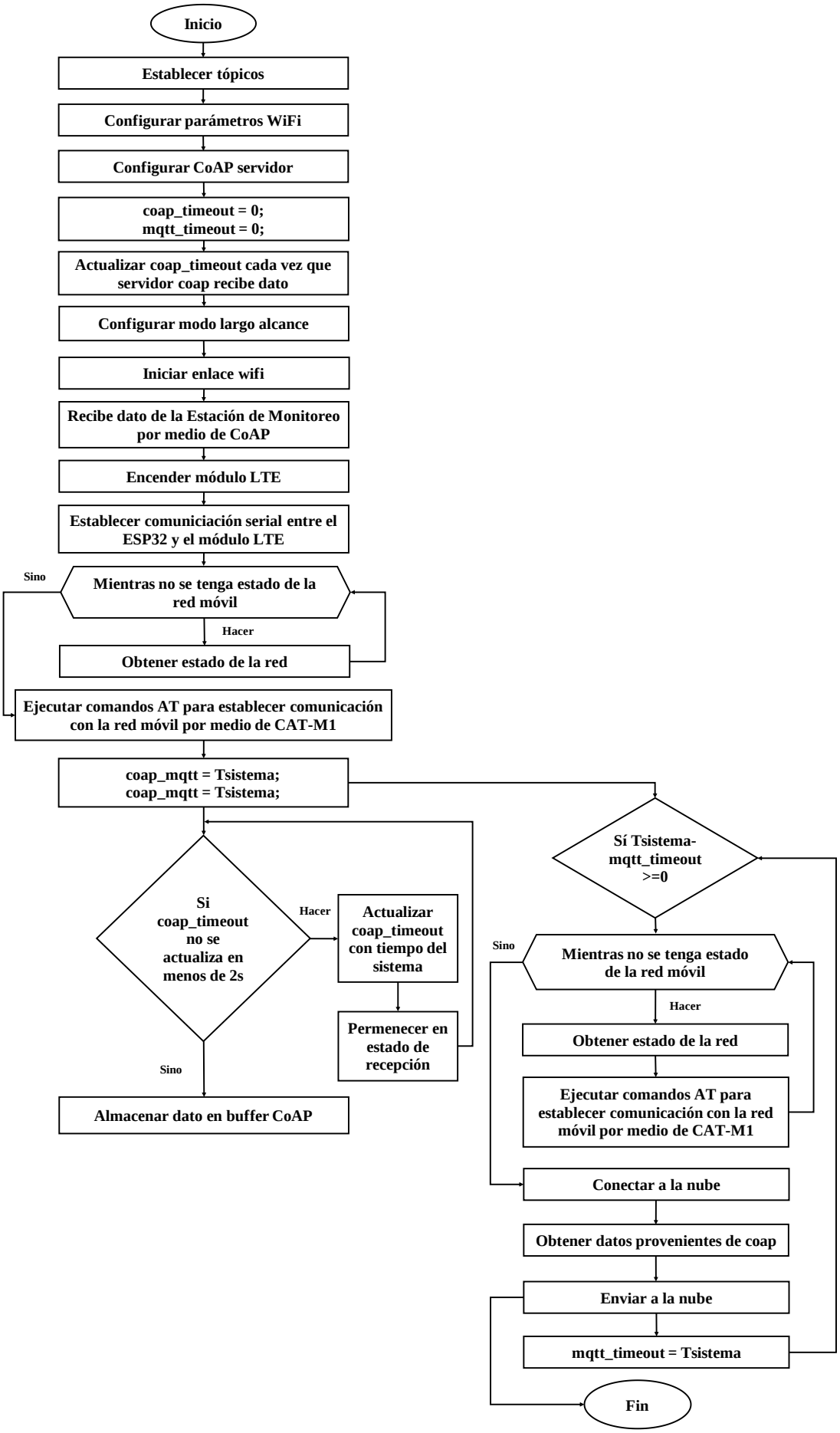

Inicio
Establecer tópicos
Configurar parámetros WiFi
Configurar CoAP servidor
coap_timeout = 0;
mqtt_timeout = 0;
Actualizar coap_timeout cada vez que servidor coap recibe dato
Configurar modo largo alcance
Iniciar enlace wifi
Recibe dato de la Estación de Monitoreo por medio de CoAP
Encender módulo LTE
Establecer comuniciación serial entre el ESP32 y el módulo LTE
Sino
Mientras no se tenga estado de la red móvil
Hacer
Obtener estado de la red
Ejecutar comandos AT para establecer comunicación con la red móvil por medio de CAT-M1
coap_mqtt = Tsistema;
coap_mqtt = Tsistema;
Si coap_timeout no se actualiza en menos de 2s
Hacer
Actualizar coap_timeout con tiempo del sistema
Permenecer en estado de recepción
Sino
Almacenar dato en buffer CoAP
Sí Tsistema-mqtt_timeout >=0
Sino
Mientras no se tenga estado de la red móvil
Hacer
Obtener estado de la red
Ejecutar comandos AT para establecer comunicación con la red móvil por medio de CAT-M1
Conectar a la nube
Obtener datos provenientes de coap
Enviar a la nube
mqtt_timeout = Tsistema
Fin

Figura 25 Diagrama de flujo del Punto de Acceso
Fuente: Elaboración propia

3.2.2 Pruebas realizadas

En esta etapa se desarrolló como prueba de concepto un prototipo con la finalidad de observar el comportamiento de los datos en la red.

3.2.2.1 Código

En esta sección se muestra parte del código para configurar los instrumentos de la red inalámbrica.

La figura 26 muestra los parámetros para la configuración de WiFi.

```
const char* ssid = "kkkkk";         //  AP ssid
const char* password = "12345678"; //  AP password
```

Figura 26 Configuración WiFi
Fuente: Elaboración propia

La función WIFI_STA estableció el WiFi en modo estación mientras que el modo LR se habilitó por medio de la función esp_wifi_set_protocol como se ve en la figura 27.

```
WiFi.mode( WIFI_STA );                                          //  para modo estación
int a= esp_wifi_set_protocol( WIFI_IF_STA, WIFI_PROTOCOL_LR ); //  configura modo LR
```

Figura 27 Fracción código modo largo alcance
Fuente: Elaboración propia

El cliente CoAP se configuró a través de la función callback_response que permite declarar la dirección IP a la cual debe enviar el dato y el puerto. Además, se declara el protocolo de transporte UDP como se observa en la figura 28.

```
void callback_response(CoapPacket &packet, IPAddress ip, int port); //  CoAP client response callback

WiFiUDP udp;                                                        //  Crea la clase WiFiUDP para
                                                                    //  enviar y recibir mensajes UDP
Coap coap(udp);                                                     //  Crea la clase coap usando udp
```

Figura 28 Configuración de cliente CoAP
Fuente: Elaboración propia

El dato a enviar, que en este caso de experimentación fue el RSSI, se almacena en una variable “message” como se aprecia en la figura 29.

```
String message = String(WiFi.RSSI());                  //  mensaje tipo string
message = "RSSI: " + message;                           //  concatena trama del mensaje
char message_char[100];                                 //  almacena valor actual message_char[100]
message.toCharArray(message_char,message.length()+1);   //  Copia los caracteres de la cadena en el
                                                        //  búfer suministrado.
```

Figura 29 Código de referencia para dato a enviar
Fuente: Elaboración propia

La función coap.put permite enviar el dato a la dirección IP a través del puerto para CoAP como se observa en la figura 30.

```
Serial.println("Send Request");                                          //  imprime en monitor serial
int msgid = coap.put(IPAddress(192, 168, 4, 1), 5683, "rssi", message_char); //  prepara mensaje a enviar por coap
Serial.println(message_char);                                           //  imprime en pantalla lcd
```

Figura 30 Código de referencia para transmisión de datos a través de CoAP
Fuente: Elaboración propia

Para el Punto de Acceso se estable el servidor en la nube al que serán enviados los datos. La figura 31 muestra el código de referencia para la configuración del broker MQTT para el prototipo.

```
/************************* CONFIGURA MQTT *********************************/
#define AIO_SERVER      "85.119.83.194"                 //  Servidor MQTT
#define AIO_SERVERPORT  1883                            //  Puerto
```

Figura 31 Configuración servidor en la nube MQTT
Fuente: Elaboración propia

Se declara la ruta en la nube a la cual serán enviados los datos, en este caso de experimentación se le denominó “JRO_SATH/DATA” como muestra la figura 32.

```
Adafruit_MQTT_Publish feed_data = Adafruit_MQTT_Publish(&mqtt, "JRO_SATH/DATA"); //  Tópico para datos CoAP
```

Figura 32 Código de referencia para establecer tópicos en el bróker MQTT
Fuente: Elaboración propia

Para el Punto de Acceso, se establece el wifi en modo Access Point. De igual manera se habilita el modo LR como se observa en la figura 33.

```
WiFi.mode( WIFI_AP_STA );                                   //  para modo access point
int a= esp_wifi_set_protocol( WIFI_IF_AP, WIFI_PROTOCOL_LR ); //  habilita modo LR
```

Figura 33 Fragmento de código para configurar wifi en modo LR
Fuente: Elaboración propia

Se establece el servidor CoAP que permite recibir los datos enviados por la Estación que en este caso de experimentación fue el RSSI. La función “callback”, permite al cliente devolver el estado del recurso solicitado según muestra la figura 34.

```
coap.server(callback_rssi, "rssi");       //  configura callback en el server coap
```

Figura 34 Código de referencia para establecer servidor CoAP
Fuente: Elaboración propia

Los datos recibidos por el servidor CoAP se envían a la nube en pequeñas tramas a través de la función “mqtt_publish_checkSuccess” como muestra la figura 35.

```
MQTT_publish_checkSuccess(feed_data, buffer_coap_char, strlen(buffer_coap_char)); //  publica datos del coap
```

Figura 35 Código de referencia para la publicación de datos en la nube
Fuente: Elaboración propia

3.2.2.2 Pruebas de funcionamiento

- Prueba 1: Transmisión de datos de las Estaciones al Punto de Acceso

El prototipo para el Punto de Acceso que se muestra en la figura 36 constó del módulo SIM7000E, el módulo ESP32, la antena de telefonía móvil, la antena WiFi, la antena GPS, la pantalla LCD para visualizar los datos y una batería portátil.

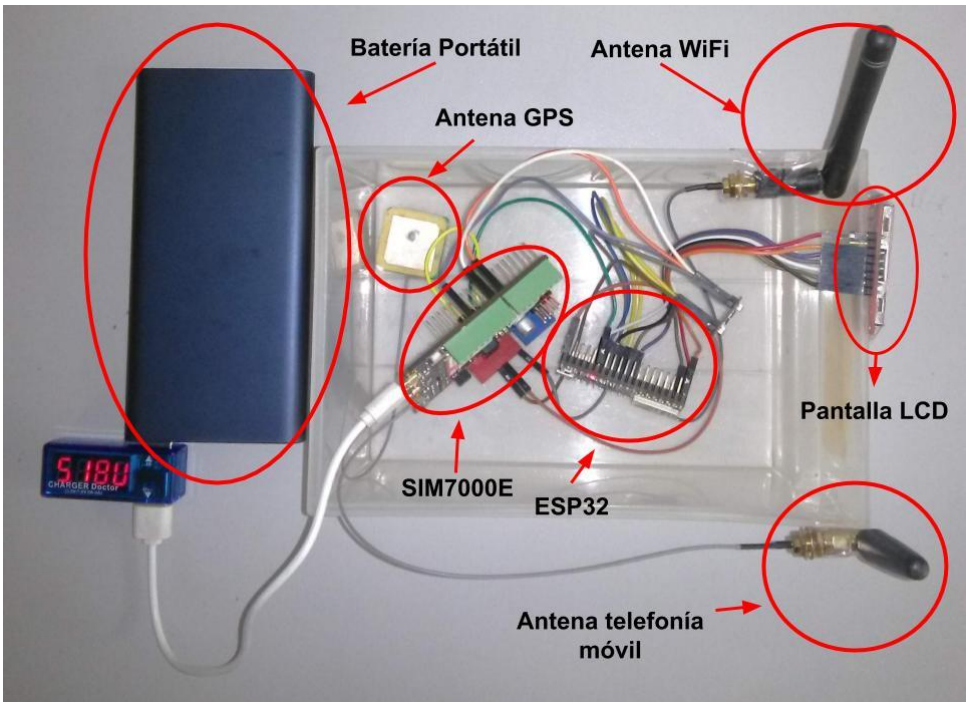


Figura 36 Punto de Acceso
Fuente: Elaboración propia

El prototipo para las Estaciones constó de un módulo ESP32, una antena WiFi, pantalla LCD para visualizar los datos y una batería portátil. En la figura 37 se puede apreciar una Estación.

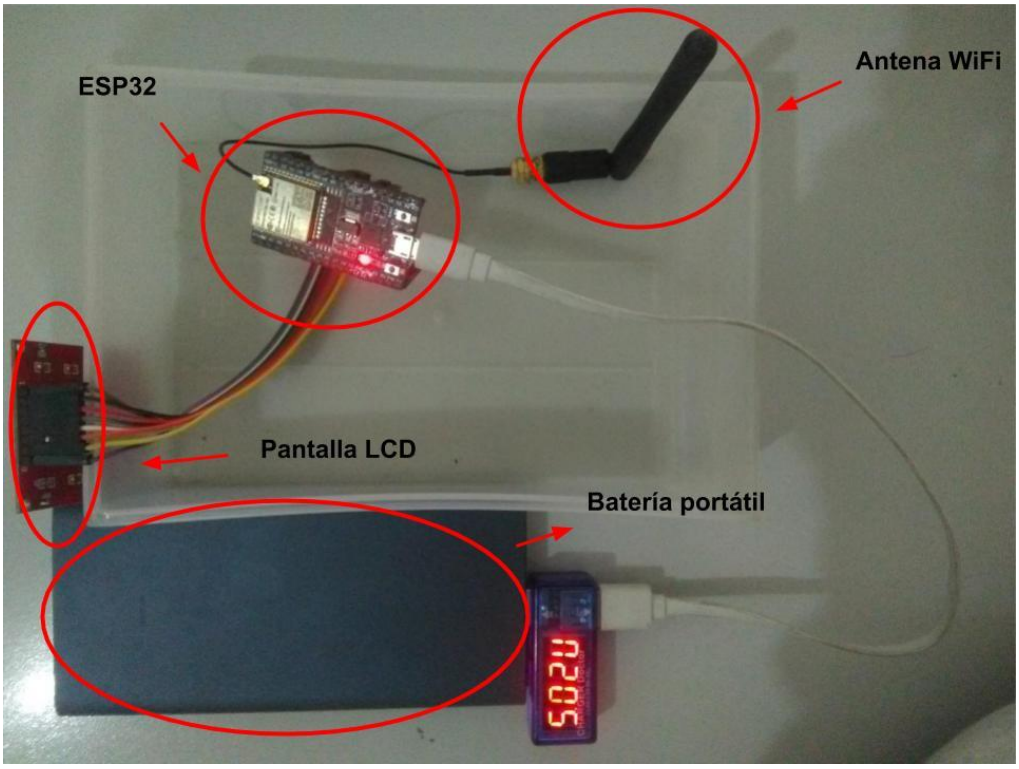


Figura 37 Estación
Fuente: Elaboración propia

Las estaciones emitieron datos (como ya se mencionó, el RSSI) a través del protocolo de transferencia CoAP al Punto de Acceso.

Resultados de la prueba 1

La figura 38 muestra el Punto de Acceso colocado en campo. Se puede observar que una de las Estaciones está en línea y ha transmitido con éxito un dato que en ese instante fue -76 dBm.

Figura 38 Punto de Acceso en campo
Fuente: Elaboración propia

Las figuras 39 y 40 muestran las Estaciones ubicadas en campo emitiendo datos con éxito.

Figura 39 Estación 1 emitiendo datos en campo
Fuente: Elaboración propia

Figura 40 Estación 2 emitiendo datos en campo
Fuente: Elaboración propia

- Prueba 2: Transferencia de datos a internet a través de la red móvil

Los datos que recibió el Punto de Acceso fueron transferidos a internet a través de la red móvil empleando la tecnología Cat-M1.

Resultados de la prueba 2

Para ello se inicializó el módulo SIM7000E estableciendo la funcionalidad del modem y verificando el código PIN con los comandos CFUN y CPIN, como se ve en la figura 41 a través del monitor serial de Arduino. Se solicitó el IMEI del equipo con el comando AT+GSN y el RSSI del módulo celular con el comando AT+CSQ. Mostró un valor de 99, 99 que quiere decir no reconocido o no detectable por lo que reintentó conectarse a la red. Verificó el estado de la red con el comando AT+CGREG? Como resultado se obtuvo “no registrado (buscando)”.

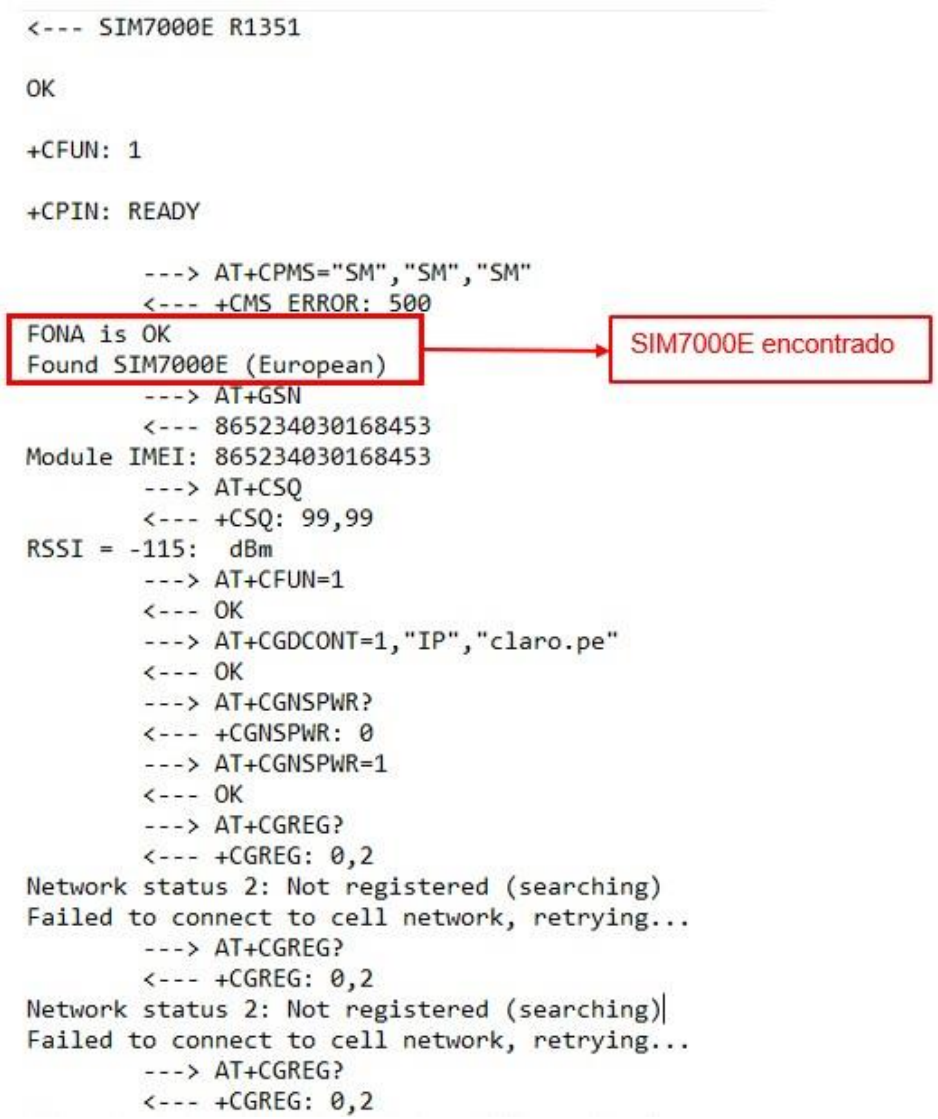


Figura 41 Monitor Serial Inicialización del Módulo SIM7000E
Fuente: Elaboración propia

Luego de un momento se logró conectar mostrando el estado “Registrado (home)”. Se estableció la selección entre Cat-m1 y NB con el comando AT+CMNB? Se configuró la banda Cat-m1 con el comando AT+CBANDCFG. Se conectó a la banda 28 LTE (700 MHz) de la operadora Claro como muestra la figura 42. Se comprobó el APN con el comando AT+CGNAPN, lo que devolvió el valor “1” que significa que la red envió el parámetro APN al equipo del usuario y el APN “claro.pe”.

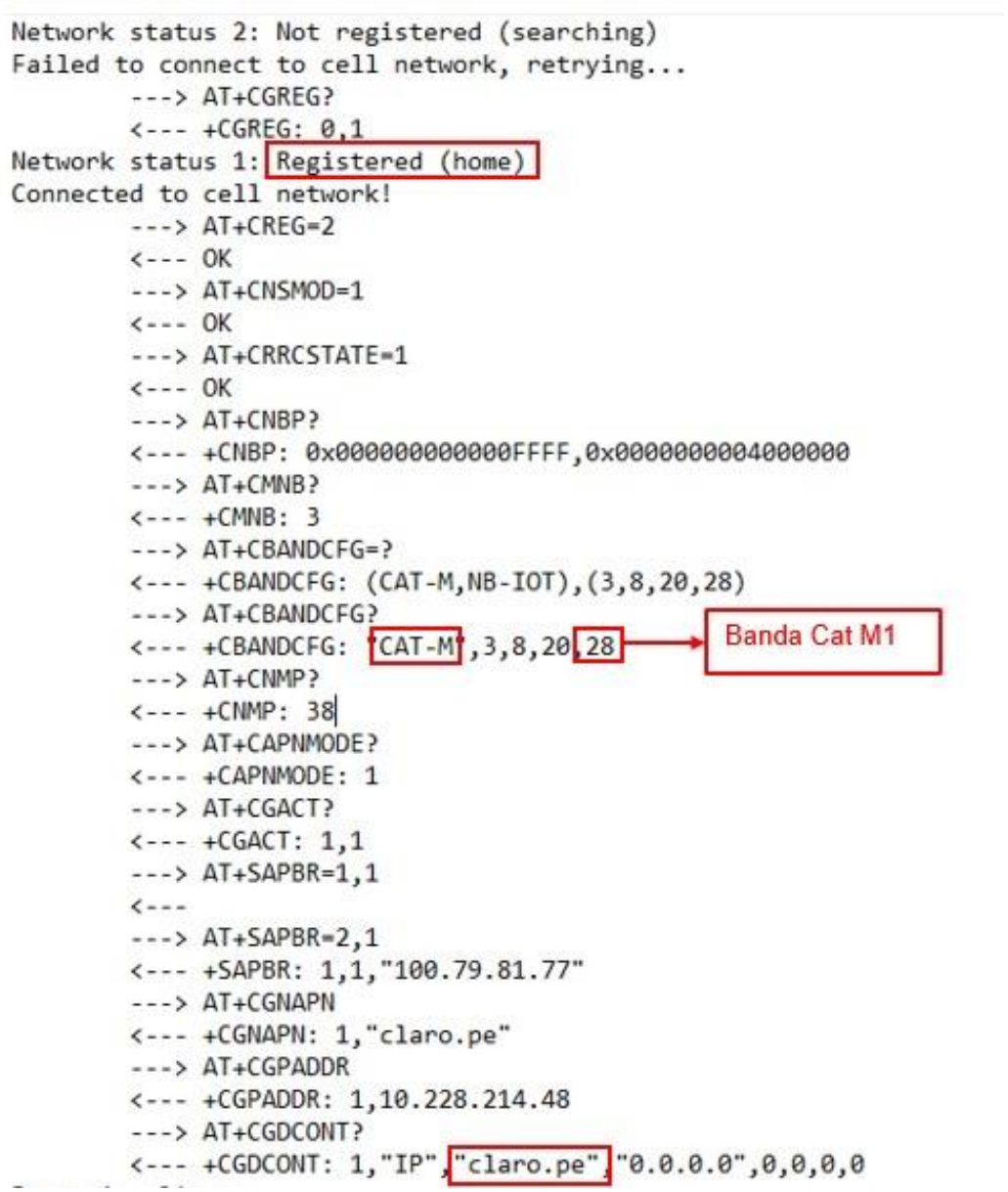


Figura 42 Monitor serial Arduino módem registrado
Fuente: Elaboración propia

La figura 43 muestra parte del flujo de valores enviados en la transferencia de datos a través de CoAP. Debido a que los clientes estaban siendo trasladados, aún estaban cerca del Punto de Acceso por lo que los valores del RSSI fueron altos. Luego verificó el registro en la red con el comando AT+CGREG? y mostró los datos de la batería y del GPS con los comandos AT+CBC y AT+CGNSINF respectivamente.

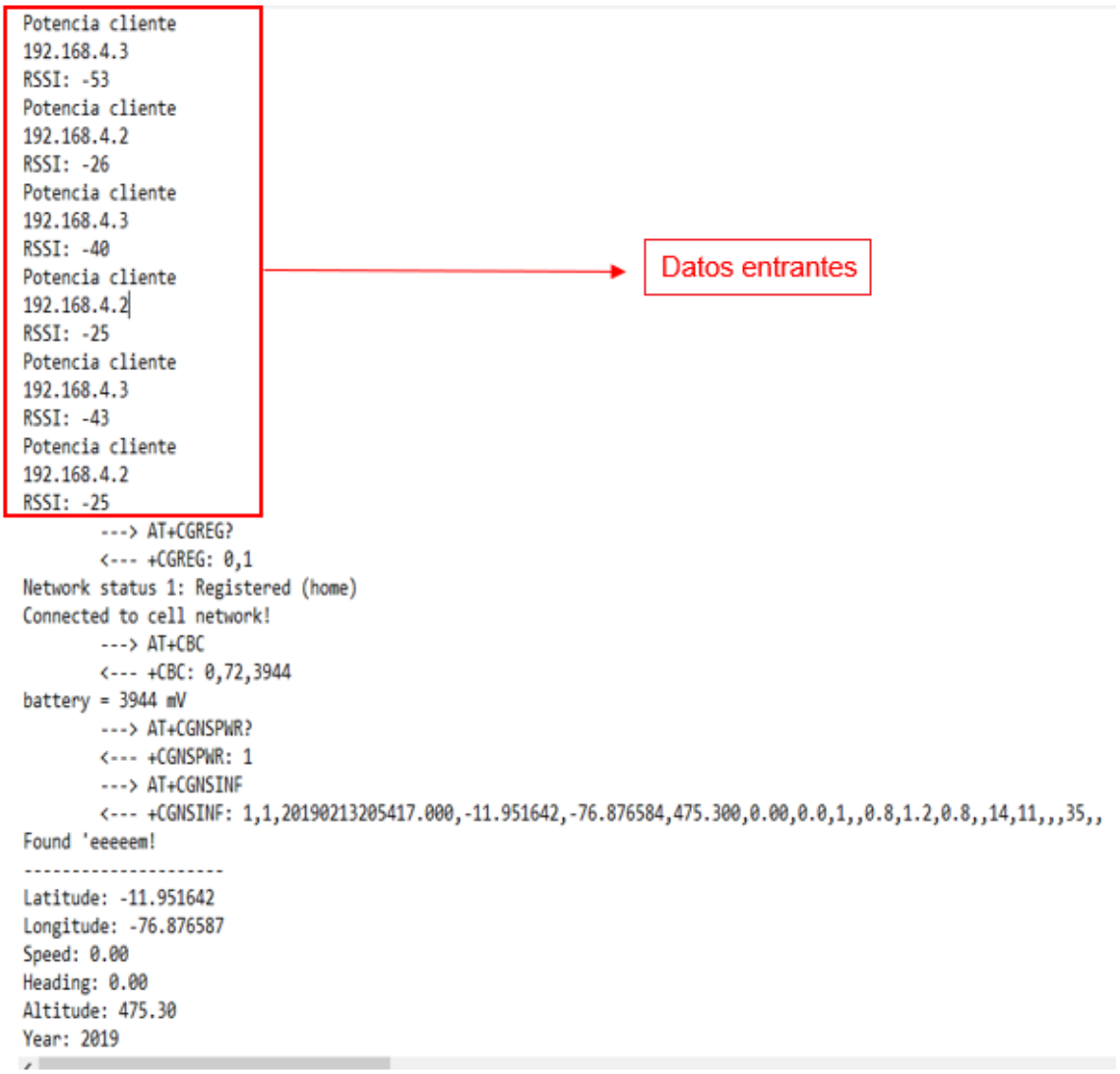


Figura 43 Monitor serial Arduino datos provenientes de las Estaciones
Fuente: Elaboración propia

Mostró la fecha, verificó la potencia del módulo de telefonía celular con el comando AT+CSQ que mostró el valor de -83 dBm (figura 44). Se consultó el estado de la conexión con el comando AT+CIPSTATUS.

Se inició la conexión TCP con el comando AT+CIPSTART al servidor MQTT “Mosquitto”.

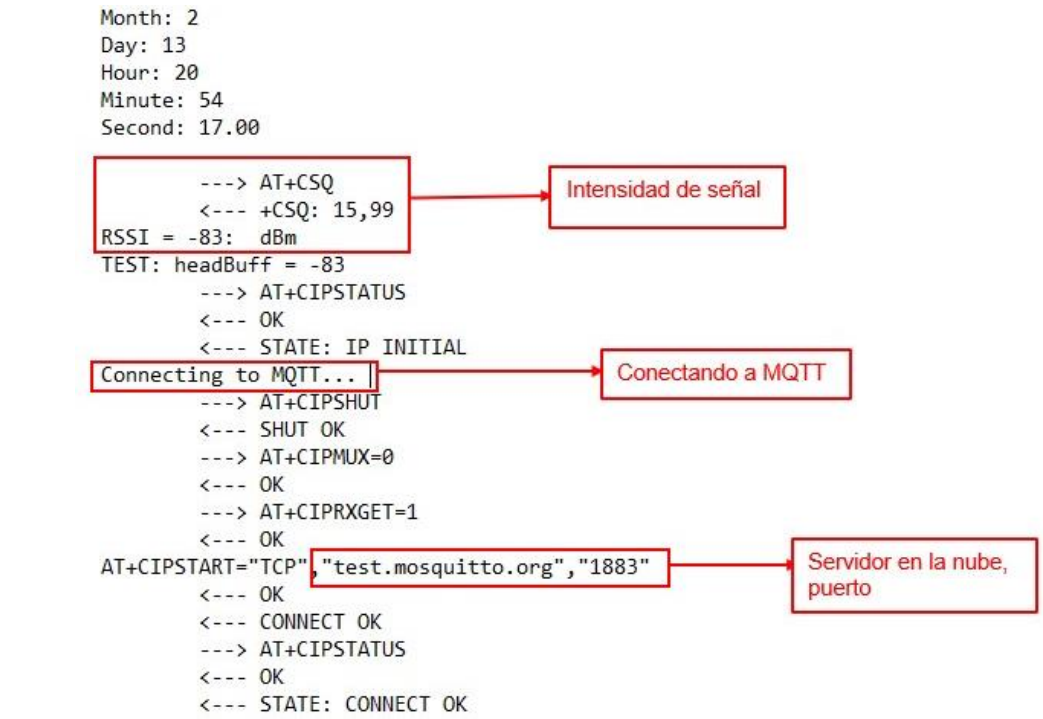


Figura 44 Monitor serial conexión a Mosquitto
Fuente: Elaboración propia

Conectado al servidor Mosquitto, figura 45, almacenó los datos en el buffer.

Luego se verificó el estado de la conexión con el comando AT+CIPSTATUS, que devolvió el estado CONNECT OK.

Se enviaron los datos y los publicó en el servidor Mosquitto. Los datos son enviados usando el comando "AT+CIPSEND=" que permite enviar datos a través de una conexión TCP o UDP

Figura 45 Monitor serial envío de datos al servidor Mosquitto
Fuente: Elaboración propia

- Prueba 3: Visualización de datos en la nube

Al ser un diseño de prototipo, se empleó un bróker MQTT Mosquitto de versión gratuita que se instaló en una PC, además se elaboró una interfaz gráfica en el software libre Processing para visualizar la emisión de datos a la nube.

Resultados de la prueba 3

La figura 46 muestra una parte de los datos emitidos del Punto de Acceso al tópico suscrito JRO_SATH/DATA mientras las Estaciones estaban siendo trasladadas.

La nomenclatura de suscripción -h indica el host que almacenó los datos. Este host es el bróker Mosquitto y su dirección IP fue 85.119.83.194, con calidad (-q) 1 que garantiza la entrega del mensaje.

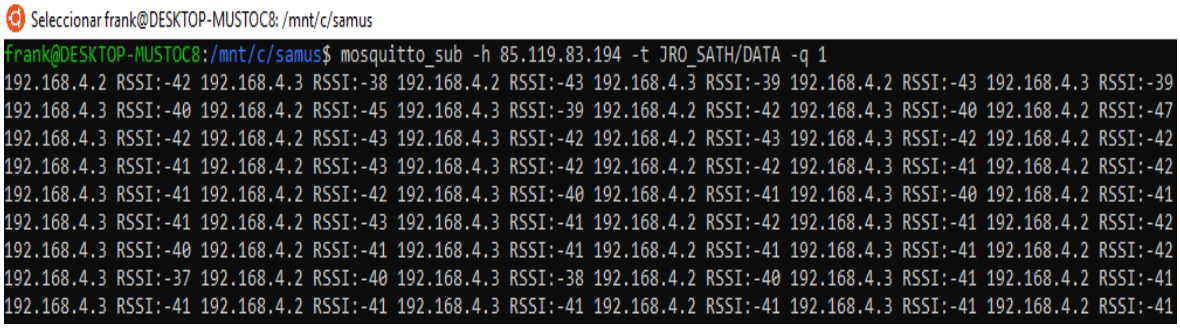


Figura 46 Bróker MQTT Mosquitto
Fuente: elaboración propia

Como ya se mencionó, con el fin de observar el comportamiento de los datos en la red, la interfaz gráfica permitió visualizar de manera más amigable los datos enviados a la nube. De acuerdo a la prueba realizada, los ítems seleccionados fueron la dirección IP de las Estaciones, el "RSSI" enviado por las Estaciones, además del "RSSI" del módulo SIM7000E publicado en otro tópico.

En este instante el valor enviado por la Estación 1 en campo fue -76 dBm, el valor enviado por la Estación 2 en campo fue -88 dBm, el RSSI del módulo SIM7000E fue -83 dBm como muestra la figura 47.

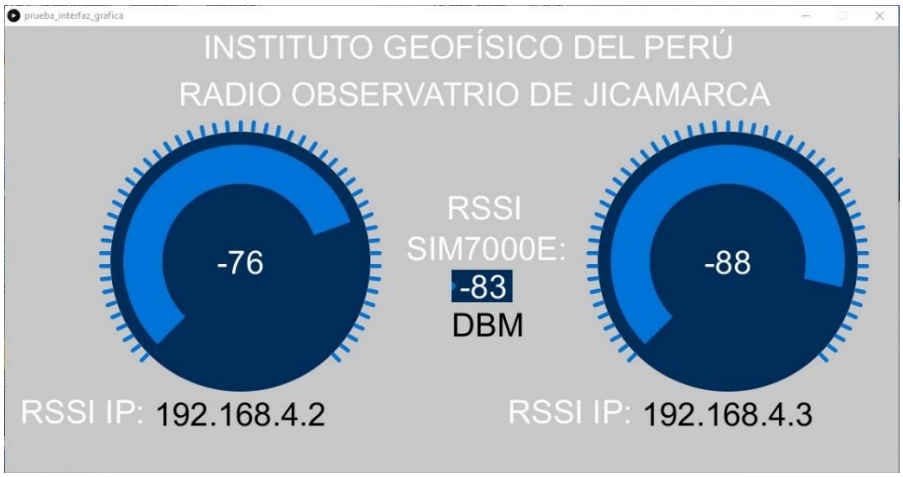


Figura 47 Interfaz gráfica
Fuente: Elaboración propia

- Prueba 4: Procesamiento y análisis de resultados

Posteriormente, en coordinación con el Radio Observatorio de Jicamarca se realizó el procesamiento de los datos que se enviaban al broker MQTT como parte del proceso de observación del comportamiento de los datos en la red con el prototipo. Consistió en determinar la frecuencia con la que estos eran publicados a internet. Para ello se modificó el programa elaborado en Processing para que genere archivos en formato .CSV (valores separados por coma). Uno de ellos con los valores del RSSI (datos de prueba), relacionados con la dirección IP de la Estación de la que procedía dicha información y el tiempo en el que fue enviado. El segundo, con los valores del RSSI del modem SIM7000E cuando transmitió datos al broker y el tiempo en el que fue enviado.

Se elaboró un programa en Octave que permitió importar los archivos .CSV para generar gráficos en los que se pudo observar la latencia de los paquetes que fueron enviados a internet por el módulo SIM7000E y los niveles del RSSI del módem SIM7000E.

Se obtuvo por un lapso de 16 horas el registro de los valores del RSSI (datos de prueba), relacionados con la dirección IP de la que procedió dicha información y el tiempo en el que fue enviado (log_data_06_05_2019), el registro del RSSI del modem SIM7000E cuando transmitió datos al broker y el tiempo en el que fue enviado (log_rssi_06_05_2019), la gráfica de la latencia de los paquetes que fueron enviados a internet por el módulo SIM7000E y la gráfica con los niveles del RSSI del módulo SIM7000E.

Resultados de la prueba 4

En esta sección, se analiza el comportamiento de los datos en la red de acuerdo a las pruebas realizadas con el prototipo.

De acuerdo a la figura 48 se observa que el Punto de Acceso envía un conjunto de datos a la nube cada 10 segundos aproximadamente, por lo que en un intervalo de alrededor de 1 minuto se está recibiendo 6 paquetes con datos de ambas Estaciones.

Day	Month	Year	Hour	Minute	Second	IP	RSSI
6	5	2019	12	35	13	client conne	254
6	5	2019	12	35	22	192.168.4.2	-78
6	5	2019	12	35	22	192.168.4.3	-87
6	5	2019	12	35	22	192.168.4.2	-77
6	5	2019	12	35	22	192.168.4.3	-78
6	5	2019	12	35	22	192.168.4.2	-77
6	5	2019	12	35	32	192.168.4.2	-76
6	5	2019	12	35	32	192.168.4.2	-76
6	5	2019	12	35	32	192.168.4.2	-77
6	5	2019	12	35	32	192.168.4.3	-87
6	5	2019	12	35	32	192.168.4.2	-77
6	5	2019	12	35	42	192.168.4.2	-76
6	5	2019	12	35	42	192.168.4.3	-87
6	5	2019	12	35	42	192.168.4.2	-77
6	5	2019	12	35	42	192.168.4.3	-89
6	5	2019	12	35	42	192.168.4.2	-77
6	5	2019	12	35	51	192.168.4.2	-77
6	5	2019	12	35	51	192.168.4.3	-87
6	5	2019	12	35	51	192.168.4.2	-75
6	5	2019	12	35	51	192.168.4.3	-83
6	5	2019	12	35	51	192.168.4.2	-77
6	5	2019	12	36	1	192.168.4.2	-75

log_data_06_05_2019

Figura 48 Registro de datos enviados por las Estaciones
Fuente: Elaboración propia

Esto se puede apreciar mejor en la figura 49 en la que el Punto de Acceso envía un solo valor cada 10 segundos aproximadamente, por lo que en un intervalo de 3 minutos se está recibiendo 18 paquetes.

Day	Month	Year	Hour	Minute	Second	RSSI
6	5	2019	12	35	13	254
6	5	2019	12	35	18	-83
6	5	2019	12	35	28	-83
6	5	2019	12	35	37	-83
6	5	2019	12	35	46	-83
6	5	2019	12	35	56	-83
6	5	2019	12	36	6	-83
6	5	2019	12	36	16	-83
6	5	2019	12	36	25	-83
6	5	2019	12	36	35	-83
6	5	2019	12	36	45	-83
6	5	2019	12	36	54	-83
6	5	2019	12	37	4	-83
6	5	2019	12	37	14	-83
6	5	2019	12	37	23	-83
6	5	2019	12	37	32	-115
6	5	2019	12	37	42	-85
6	5	2019	12	37	52	-85
6	5	2019	12	38	2	-85
6	5	2019	12	38	11	-85
6	5	2019	12	38	21	-85
6	5	2019	12	38	31	-85

log_rssi_06_05_2019

Figura 49 Registro del RSSI del módulo SIM7000E
Fuente: Elaboración propia

Generalizando, la latencia proporciona el lapso de tiempo desde que el Punto de Acceso envía los datos hasta que estos llegan a la nube. Esto se puede apreciar de la figura 50.

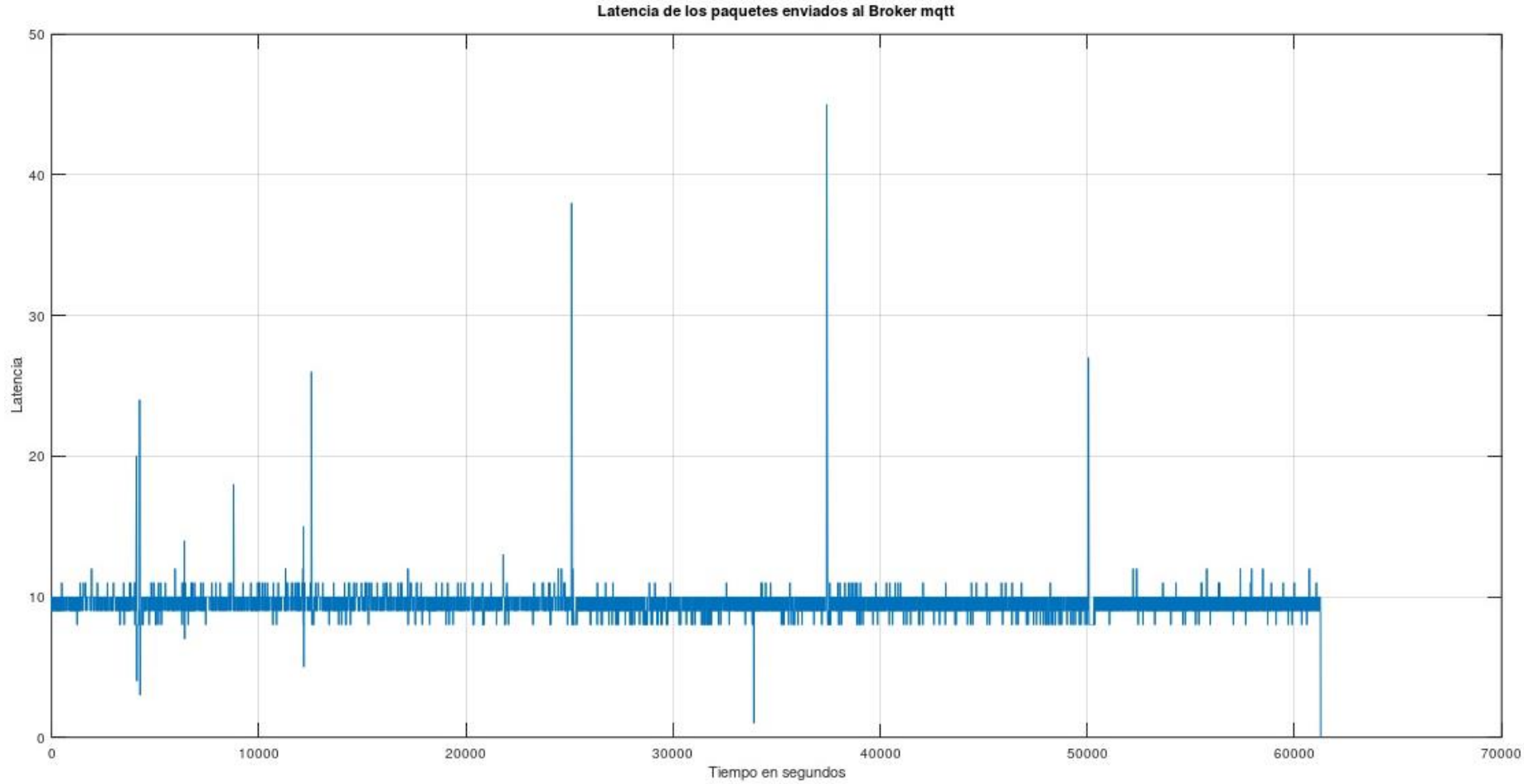


Figura 50 Latencia de los paquetes enviados a internet desde el módulo SIM7000E del archivo log_data_06_05_2019
Fuente: Elaboración propia

Con la ayuda del programa elaborado en Octave, se observa que, en promedio, la latencia en el envío de los paquetes del módem SIM7000E al bróker fue de 9.54s, como muestra la figura 51, durante las 16 horas que se hizo el registro de la emisión de los datos.

```
>> mean(latencia)
ans =  9.5493
```

Figura 51 Promedio de la latencia en el envío de paquetes del módulo SIM7000E del archivo log_data_06_05_2019
Fuente: Elaboración propia

No obstante, se presentaron varios picos fuera de este rango. La latencia tomó valores superiores a 10 s un número de 175 veces, como muestra la figura 52.

```
>> size(latencia(find(latencia>10)))
ans =

   175     1
```

Figura 52 Latencias superiores al promedio
Fuente: Elaboración propia

Siendo su máximo valor de 45 s, presentándose solo en una ocasión como muestra las figuras 53 y 54.

```
>> mean(max(latencia))
ans =  45
```

Figura 53 Latencia máxima en el envío de datos del Punto de Acceso a la nube
Fuente: Elaboración propia

```
>> size(max(latencia))
ans =

   1   1
```

Figura 54 Número de ocasiones de máxima latencia en el envío de datos del Punto de Acceso a la nube
Fuente: Elaboración propia

Lo cual puede deberse a diversas razones, como problemas en la red móvil, el clima, inconvenientes en la transmisión de las estaciones, inconvenientes con el servidor en la nube.

La figura 55 muestra los niveles del RSSI del módem SIM7000E lo cual proporciona una referencia de los niveles adecuados que debe percibir el módulo para enviar datos a la nube.

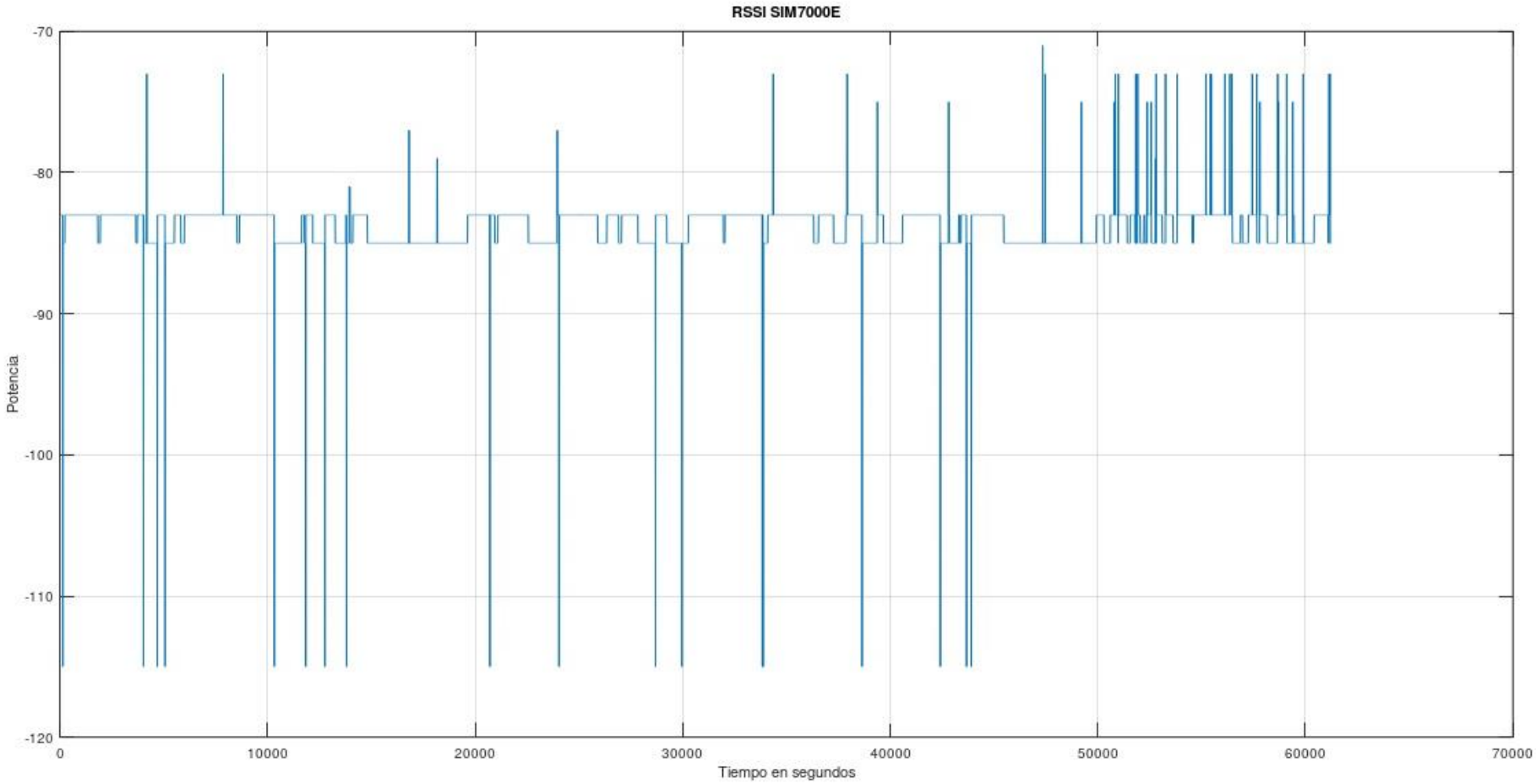


Figura 55 RSSI del módulo SIM7000E
Fuente: Elaboración propia

Nuevamente, con la ayuda del programa elaborado en Octave, se obtuvo el promedio de la potencia del módulo SIM7000E al estar transmitiendo al broker, que fue de -83,87 dBm, como muestra la figura 56.

```
>> mean(rssi(3:end))
ans = -83.870
```

Figura 56 Promedio de la potencia del módem SIM7000E
Fuente: Elaboración propia

Sin embargo, se registraron valores por debajo de -115 dBm un total de 18 veces, como muestra la figura 57. En dichos puntos es probable que se presente desconexiones a la red móvil por problemas de cobertura debido a inconvenientes con la red móvil o el clima.

```
>> size(rssi(find(rssi<-85)))
ans =

   18    1
```

Figura 57 Valores muy bajos del RSSI del módem SIM7000E
Fuente: Elaboración propia

- Prueba 5: Costos

Los elementos de la red, tanto los instrumentos como las tecnologías empleadas seleccionadas para la elaboración de esta tesis han sido sometidos a rigurosos filtros con el fin de garantizar su correcto funcionamiento al darse las condiciones necesarias, como la cobertura móvil, la disponibilidad de la red móvil. El módulo de desarrollo ESP32 ha sido diseñado exclusivamente para aplicaciones IoT al igual que el módem SIM7000E. Ambos dispositivos son fabricados por empresas reconocidas en el campo del IoT. La tecnología Cat-M1 es parte de la Cellular Internet of Things (CIoT), introducidas en la Release 13 de la 3GPP. Las tecnologías CoAP es reconocida por la Internet Engineering Task Force (IETF) y MQTT ha sido estandarizado por la Organization for the Advancement of Structured Information Standards (OASIS).

Resultados de la prueba 5

Los costos asociados al trabajo de tesis se detallan en esta sección. De acuerdo al esquema planteado, se propone dos puntos de monitoreo, por lo que algunos elementos están multiplicados.

Tabla 11 Tabla de costos

	Componente	Marca y modelo	Proveedor	Cantidad	Precio unitario (USD)	Total (USD)
Equipos	Microcontrolador	ESP32	Mouser Electronics	3	10	30
	Módulo de transmisión de RF	ESP32	Mouser Electronics	-	-	0
	Módulo de transmisión de RF de tecnología celular	SIM7000E	DFRobot	1	39.90	39.90
	Antena WiFi	Genérico	Mouser Electronics	3	4.59	13.77
	Antena LTE	50mm portátil 700MHz Mini 4G LTE antena dipolo	Alibaba	1	7	7
Software	Arduino IDE	Arduino	Gratuito	-	0	0
Total						90,67

Fuente: Elaboración propia

El costo para acceder a la tecnología Cat-M1 a través de la red móvil no está incluido en el análisis ya que el sistema es un diseño de prototipo. Al tratarse de una tecnología nueva y propiedad de una empresa privada, está sujeta a cambios en la red y depende de la tarificación que establezca la operadora móvil para un mejor servicio, por lo cual depende de las negociaciones entre las áreas comerciales de ambas instituciones. Sin embargo, en el momento de las pruebas se pudo acceder por medio del paquete de megas que se adquirió temporalmente.

El costo del servicio en la nube no está incluido en el análisis ya que el sistema es un diseño de prototipo y es posible implementar un servidor propio con entornos de desarrollo libre, sin embargo, es posible contratar un servicio en la nube para gestionar los datos. En el caso del servidor Mosquitto empleado en el prototipo es un servicio gratuito, es estable a pesar de estar hecho para fines de pruebas, pero ofrece servicios para que la conexión sea más segura brindando la posibilidad de donar un monto de dinero. Existen otras soluciones como Google Cloud IoT en las que se tarifica por MB de datos intercambiados con el servicio desde dispositivos del Internet de las Cosas después de haber utilizado el nivel gratuito de 250 MB al mes.

El costo de mantenimiento de las Estaciones es cubierto por el Radio Observatorio de Jicamarca ya que dentro de su plan de actividades se contempla el mantenimiento de la

instrumentación y equipos que este posee y cuenta con personal técnico propio para cumplir dichas actividades por lo que no es un gasto adicional. Sin embargo, la ventaja que ofrece la propuesta planteada de acuerdo a la descripción del problema en la que se menciona que la pérdida de los equipos es común durante los huaicos es que al emplear instrumentos de redes inalámbricos más económicos, que utilizan entornos de desarrollo libre y que son diseñados para entornos limitados reemplazarlos es menos costoso.

El sistema de alerta temprana de huaicos actual del Radio Observatorio de Jicamarca está presupuestado, siendo comparado a nivel de red con el sistema propuesto en la presente tesis, en lo que se muestra en la tabla 12.

Se consideró dos estaciones clientes y un repetidor.

Tabla 12 Tabla costos elementos de red del Sistema de Alerta Temprana de Huaicos

Elemento	**Unidades**	**Precio ($)**	**Total**
Access Point Para Exterior Tp-Link / CPE220	3	44,65	133,95
Inyector POE pasivo Tp-Link	4	4,52	18,08
Switch Tp-Link	3	9,05	27,15
Cable STP	4	93,51	374,04
Conectores RJ 45 metálicos para exteriores		21,12	21,12
Arduino Yun	2	49,77	99,54
Total $			673,88

Fuente: Elaboración propia

Finalmente, la red de instrumentos inalámbricos propuesta en la Tesis tiene un costo total de $90.67, mientras que la red de instrumentos inalámbricos en uso tiene un costo total de $673.88, esto representa una reducción del 86.55%.

IV. CONCLUSIONES

- Se diseñó, implementó y testeó un prototipo de red de instrumentos inalámbricos con conexión directa a internet y con tecnologías de bajo costo para un sistema de alerta temprana de huaicos en la quebrada de Jicamarca obteniéndose resultados exitosos en las pruebas.
- Se diseñó, implementó y testeó un prototipo de red de instrumentos inalámbricos con conexión directa a internet. La prueba 1 sirvió para demostrar la transferencia de datos de los instrumentos de comunicación inalámbricos en una red local, la prueba 2 permitió demostrar la transferencia del flujo de datos adquiridos por la red local a internet a través de la red móvil por medio de la tecnología Cat-M1. La prueba 3 permitió la visualización de los datos transferidos por la red por medio de un servidor alojado en la nube, la prueba 4 permitió procesar los datos que se enviaron a la nube y verificar la disponibilidad de la red.
- Se diseñó e implementó un prototipo red de instrumentos inalámbricos con tecnologías de bajo costo. Se logró disminuir el costo de la red inalámbrica empleando tecnologías IoT. En la prueba 5 se demostró que el presupuesto representa una disminución del 86.55% del costo de la red de instrumentos inalámbricos, empleando instrumentos de red inalámbricos de fabricantes reconocidos en el sector de Internet of Things. Las tecnologías de radiofrecuencia que se usaron están estandarizadas para un óptimo uso de recursos en transferencia de datos en entornos limitados.
- Se logró cumplir con el objetivo propuesto en el proyecto 384-PNICP-PIAP-2014 financiado por el Programa Nacional de Innovación para la Competitividad y Productividad, Innóvate Perú, al desarrollar un prototipo de red de instrumentos inalámbricos de bajo costo y con conexión directa a internet para el Radio Observatorio de Jicamarca que puede ser integrado a su Sistema de Alerta Temprana de Huaicos como parte del proceso de mejora continua.

V. REFERENCIAS BIBLIOGRÁFICAS

Aguilar, S. (2019). *Diseño de un sistema basado en IoT para la supervisión y control de estaciones remotas de la dirección de hidrografía y navegación de la Marina de Guerra del Perú* (Tesis de Postgrado, Pontificia Universidad Católica del Perú, Lima, Perú). Recuperada de http://hdl.handle.net/20.500.12404/14211

Barton, R., Hanes, D., Salgueiro, G., (2017). *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things.* Indiana 46240 USA: Cisco Press.

Bormann, C., Hartke, K., Shelby, Z. (junio, 2014). *RFC 7252 – The Constrained Application Protocol (CoAP)*, de Internet Engineering Task Force (IETF). Recuperado de: https://tools.ietf.org/pdf/rfc7252

Cirani, S., Ferrari, G., Picone, M., Veltri, Luca., (2018). *Internet of Things Architectures, Protocols and Standards*. Hoboken, NJ: Wiley.

Chivata, J. R. (2017). *Modelo de red de estaciones meteorológicas modulares con plataforma IOT* (Tesis de Pregrado, Universidad Distrital Francisco José de Caldas, Bogotá D.C., Colombia). Recuperada de http://hdl.handle.net/11349/7429

Chowdhury, F. H., Nahian, R., Uddin, T., Rezwan, S., Khan, M., Sufian, A., . . . Hassan, N. N. (2017). Design, control & performance analysis of forecast junction IoT and swarm robotics based system for natural disaster monitoring. 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), doi: 10.1109/ICCCNT.2017.8204148

Dávila, D., (2016). *Sistemas de alerta temprana ante inundaciones en América Latina* [versión PDF]. Recuperado de: https://solucionespracticas.org.pe/Sistemas-de-alerta-temprana-ante-inundaciones-en-America-Latina

Ericsson AB (2016). Cellular networks for Massive IoT, Ericsson. (vol.1, no. January). Recuperado de: https://www.gsma.com/membership/wp-content/uploads/2016/03/wp_iot.pdf

Fernández, R., Ordieres, J., Martínez, F., González, A., Alba, F., Lostado, R., Verónica Pernía, A., (2009). *Redes inalámbricas de sensores: teoría y aplicación práctica*. Recuperado de:

https://publicaciones.unirioja.es/catalogo/monografias/mdi26.shtml

Franekova, M. & Peniak, P. (2016). Model of integration of embedded systems via CoAP protocol of Internet of Things. 2016 International Conference on Applied Electronics (AE), doi: 10.1109/AE.2016.7577273

GSM Associaton. (2018). LTE-M Deployment Guide to Basic Feature set Requirements. (Versión 2.0). Recuperado

de: https://www.gsma.com/newsroom/wp-content/uploads//CLP.29-v2.0.pdf

Leon, F. (2019). Sistema Automático de Monitoreo de Mercurio en Tiempo Real en aguas aledañas a Explotaciones Mineras y Petroleras usando una Plataforma IoT (Tesis de Pregrado, Pontificia Universidad Católica del Perú, Lima, Perú). Recuperada de http://hdl.handle.net/20.500.12404/13641

LoRa Alliance. *What is the LoRaWAN Specification?* Recuperado de: https://lora-alliance.org/about-lorawan

The Things Network (Agosto, 2020). *Limitations of LoRaWAN*. Recuperado de: https://www.thethingsnetwork.org/docs/lorawan/limitations.html

Meneses, C. (2017). Sistema de Monitoreo en la Nube para medir los Riesgos Ambientales basados en Sensores de Bajo Costo (Tesis de Pregrado, Universidad Piloto de Colombia, Bogotá, Colombia). Recuperada de http://polux.unipiloto.edu.co:8080/00003994.pdf

Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. 2017 IEEE International Systems Engineering Symposium (ISSE), doi: 10.1109/SysEng.2017.8088251

OASIS Standard (diciembre, 2015). *MQTT version 3.1.1 Plus Errata 01*, Recuperado de http://docs.oasisopen.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

Raghavendra, C. S., Sivalingam K. M., & Znaty, T., *Wireless Sensor Networks*.

USA: Springer.

Shanghai SIMCom Wireless Solutions Ltd. (2018) *SIM7000 Series_AT Command Manual_V1.04*. Recuperado de: http://www.microchip.ua/simcom/LTE/SIM7000/SIM7000%20Series_AT%20Command%20Manual_V1.04.pdf

Sharevski, F. (2018). Leveraging Cellular Intemet-of-Things for Resilient and Robust Disaster Management. 2018 IEEE Global Communications Conference (GLOBECOM), doi: 10.1109/GLOCOM.2018.8647495

Shi, H. (2018). *CoAP infrastructure for IoT* (Tesis de Postgrado, University of Saskatchewan, Saskatchewan, Canadá). Recuperada de http://hdl.handle.net/10388/9710

TTacca, E. M. (2017). *Diseño de una red FOG basado en internet de las cosas para monitorear la contaminación en la bahía del Lago Titicaca* (Tesis de Pregrado, Universidad Nacional del Altiplano, Puno, Perú). Recuperada de: http://repositorio.unap.edu.pe/handle/UNAP/6344

UN/ISDR. (marzo, 2006). *Desarrollo de Sistemas de Alerta Temprana: Lista de Comprobación.* Trabajo presentado en la EWC III Tercera Conferencia Internacional sobre Alerta Temprana, Bonn, Alemania. Recuperado de: https://www.unisdr.org/files/608_spanish.pdf

Hufford, G. A., Kissick, W. A., & Longley, A. G. (abril, 1982). *A Guide to the Use of the ITS Irregular Terrain Model in the Area Prediction Mode* (No. 82-100). U.S. Department of Commerce. Recuperado de: https://www.its.bldrdoc.gov/publications/details.aspx?pub=82-100

Wi-Fi Driver - ESP32. *ESP-IDF Programming Guide latest documentation.* Recuperado de https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/wifi.html

3GPP (2013). *Study on provision of low-cost Machine-Type Communications (MTC) User Equipments (UEs) based on LTE*, de 3GPP TR 36.888 V2.1.1 Recuperado de: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2578

ANEXOS

ANEXO 1: CONVENIO N°384 "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALERTA TEMPRANA BASADO EN UN ESTUDIO GEOLÓGICO Y MODELAMIENTO COMPUTACIONAL DE FLUJOS ALUVIONALES PARA LA PREVENCIÓN DE DESASTRES"

PERÚ Ministerio de la Producción

Innóvate Perú

TERCERA ADENDA AL CONVENIO DE ADJUDICACIÓN DE RECURSOS NO REEMBOLSABLES (RNR) QUE OTORGA EL PROGRAMA NACIONAL DE INNOVACIÓN PARA LA COMPETITIVIDAD Y PRODUCTIVIDAD PARA LA EJECUCIÓN DEL PROYECTO DE INVESTIGACIÓN APLICADA: "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALERTA TEMPRANA BASADO EN UN ESTUDIO GEOLÓGICO Y MODELAMIENTO COMPUTACIONAL DE FLUJOS ALUVIONALES PARA LA PREVENCIÓN DE DESASTRES".

Tercera adenda al Convenio N° 384-PNICP-PIAP-2014.

Consta en este documento la tercera adenda al convenio de adjudicación de recursos no reembolsables para la ejecución del **PROYECTO DE INVESTIGACIÓN APLICADA,** que se celebra entre el **PROGRAMA NACIONAL DE INNOVACIÓN PARA LA COMPETITIVIDAD Y PRODUCTIVIDAD** del **MINISTERIO DE LA PRODUCCIÓN**, con RUC N° 20565526694, representado por su Coordinador Ejecutivo, Ingeniero **ALEJANDRO AFUSO HIGA**, identificado con DNI N° 10283313, facultado con Resolución Ministerial N° 300-2014-PRODUCE, domiciliado en Calle Manuel Gonzales Olaechea N° 435 - San Isidro, Lima; en adelante "**INNÓVATE PERÚ**" y el **INSTITUTO GEOFÍSICO DEL PERÚ**, con RUC N° 20131367008, representada por su Presidente Ejecutivo **HERNANDO JHONNY TAVERA HUARACHE**, identificado con DNI N° 10831063, facultado con Resolución Suprema N° 002-2017- MINAM, de fecha 31 de mayo de 2017, con domicilio para estos efectos en Calle Badajoz 169, Urb. Mayorazgo IV-Etapa - Ate - Lima, en adelante "**ENTIDAD EJECUTORA**"; declarando que convienen en lo siguiente:

ANTECEDENTES:

1) Las partes suscribieron el Convenio N° 384-PNICP-PIAP-2014, para la ejecución del proyecto: "**DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ALERTA TEMPRANA BASADO EN UN ESTUDIO GEOLOGICO Y MODELAMIENTO COMPUTACIONAL DE FLUJOS ALUVIONALES PARA LA PREVENCIÓN DE DESASTRES**".
2) Mediante solicitud vía sistema electrónico el jefe de la Unidad de Monitoreo requiere la tramitación de una adenda que comprenda la ampliación del plazo de vigencia del convenio hasta el 22.12.2017, en atención a la solicitud de la entidad ejecutora ingresada con Registro N° 201706821; y a la evaluación realizada por el ejecutivo del proyecto, quien sustenta su solicitud del plazo adicional con el fin de que la **ENTIDAD EJECUTORA** cumpla con los objetivos y fines señalados en el convenio; en consideración que la "**ENTIDAD EJECUTORA** informó que la ampliación del plazo de duración del convenio permitirá el registro y sustento de gastos realizados y la ejecución del último desembolso que se realizará al proyecto, el cual implica el proceso de transferencia financiera a la cuenta de la "**ENTIDAD EJECUTORA**".

Estando a los antecedentes expuestos:

QUINTA CLÁUSULA ADICIONAL: Las partes acuerdan modificar la Cláusula Segunda: Duración del Convenio N° 384-PNICP-PIAP-2014, siendo el nuevo texto el siguiente:

"CLÁUSULA SEGUNDA: DURACIÓN

Página 1 de 2

El presente convenio tiene una duración de ***treinta y seis (36) meses, el mismo que vencerá el día 22 de diciembre del 2017.*** *La fecha de inicio del convenio y/o proyecto es la que corresponde al primer desembolso a la cuenta corriente del proyecto. Sólo en caso de fuerza mayor y debidamente sustentado, la Unidad de Monitoreo podrá autorizar la prórroga del término del presente convenio por el plazo que determine".*

Encontrándose conformes con el contenido del presente documento, en dos ejemplares de igual valor y tenor, se firma el 26 OCT. 2017

POR INNÓVATE PERÚ

ALEJANDRO AFUSO HIGA
Coordinador Ejecutivo

POR LA ENTIDAD EJECUTORA

HERNANDO JHONNY TAVERA HUARACHE
Representante Legal

Radio Observatorio de Jicamarca

ANEXO 2: HOJA DE DATOS DE LOS DISPOSITIVOS UTILIZADOS EN EL PROYECTO

MÓDEM SIM7000E

SIM7000E

eMTC/NB-IoT/EDGE Module

Product Description

The SIM7000E is Quad-Band LTE-FDD and Dual-Band GPRS/EDGE module solution in a SMT type which supports LTE CAT-M1(eMTC) and NB-IoT up to 375kbps data transfer.

It has strong extension capability with rich interfaces including UART, USB2.0, GPIO etc. The module provides much flexibility and ease of integration for customer's application.

The package of SIM7000E is compatible with SIM900 and SIM800F. AT commands of SIM7000E are mostly compatible with SIM800F. This also minimize the investments of customers, and enables a short time-to-market.

It is designed for applications that need low latency, medium throughput data communication in a variety of radio propagation conditions. Due to the unique combination of performance, security and flexibility, this module is ideally suited for M2M applications, such as metering, telematics, asset tracking, remote monitoring , E-health , mobile pos terminals and sharing bike.

Key Benefits

- With Power Save Mode(PSM) and Extended Discontinuous Reception(eDRX), SIM7000E can extend battery life to 10 years
- SIM7000E provides deeper coverage enhancement compared to GSM
- The package and AT commands of SIM7000E mostly are compatible with SIM900 and SIM800F.

General Features

Quad-Band FDD-LTE B3/B8/B20/B28
GPRS/EDGE 900/1800Mhz
Output power
- GSM900: 2W
- DCS1800: 1W

Control Via AT Commands
Supply voltage range: 3.0V~ 4.3V, Typ: 3.8V
Operation temperature: -40℃ to +85℃
Dimensions: 24 X 24 X 2.6mm
Weight: 3.0g
GNSS (GPS,GLONASS and BeiDou/Compass, Galileo, QZSS)

Data

LTE CAT-M1(eMTC)
- Uplink up to 375kbps, Downlink up to 300kbps

NB-IoT
- Uplink up to 66kbps, Downlink up to 34kbps

EDGE Class
- Uplink up to 236.8Kbps, Downlink up to 236.8Kbps

GPRS
- Uplink up to 85.6Kbps, Downlink up to 85.6Kbps

Specifications for SMS

Point to point MO and MT
Text and PDU mode

Other Features

USB Driver for Microsoft Windows XP/Vista/7/8
USB Driver for Windows CE/Mobile
USB Driver for Linux/Android
Firmware update via USB
TCP/IP
UDP
PPP
SMS
EMAIL
FTP/HTTP/SSL
VoLTE*
FOTA *

Interfaces

USB2.0	x1	(high-speed)
UART	x2	(7-wire UART and 2-wire UART interface multiplex from GPIO)
SIM card	x1	(1.8V and 3V)
I2C	x1	
GPIO	x5	
ADC	x1	

Consumption

Power off: 7uA
PSM: 9uA
Sleep: 1mA
Idle: 11mA

Certifications

CE / RoHS / REACH

1. Overview

ESP32-WROOM-32D and ESP32-WROOM-32U are powerful, generic Wi-Fi+BT+BLE MCU modules that target a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding.

ESP32-WROOM-32U is different from ESP32-WROOM-32D in that ESP32-WROOM-32U integrates a U.FL connector. For detailed information of the U.FL connector please see Chapter 10. Note that the information in this data sheet is applicable to both modules. Any differences between them will be clearly specified in the course of this document. Table 1 lists the difference between ESP32-WROOM-32D and ESP32-WROOM-32U.

Table 1: ESP32-WROOM-32D vs. ESP32-WROOM-32U

Module	ESP32-WROOM-32D	ESP32-WROOM-32U
Core	ESP32-D0WD	ESP32-D0WD
SPI flash	32 Mbits, 3.3 V	32 Mbits, 3.3 V
Crystal	40 MHz	40 MHz
Antenna	onboard antenna	U.FL connector (which needs to be connected to an external IPEX antenna)
Dimensions (Unit: mm)	(18.00±0.10) × (25.50±0.10) × (3.10±0.10) (See Figure 6 for details)	(18.00±0.10) × (19.20±0.10) × (3.20±0.10) (See Figure 7 for details)
Schematics	See Figure 3 for details.	See Figure 4 for details.

At the core of the two modules is the ESP32-D0WD chip that belongs to the ESP32 series* of chips. The chip embedded is designed to be scalable and adaptive. There are two CPU cores that can be individually controlled, and the CPU clock frequency is adjustable from 80 MHz to 240 MHz. The user may also power off the CPU and make use of the low-power co-processor to constantly monitor the peripherals for changes or crossing of thresholds. ESP32 integrates a rich set of peripherals, ranging from capacitive touch sensors, Hall sensors, SD card interface, Ethernet, high-speed SPI, UART, I²S and I²C.

Note:
* For details on the part numbers of the ESP32 family of chips, please refer to the document *ESP32 Datasheet*.

The integration of Bluetooth, Bluetooth LE and Wi-Fi ensures that a wide range of applications can be targeted, and that the module is all-around: using Wi-Fi allows a large physical range and direct connection to the Internet through a Wi-Fi router, while using Bluetooth allows the user to conveniently connect to the phone or broadcast low energy beacons for its detection. The sleep current of the ESP32 chip is less than 5 μA, making it suitable for battery powered and wearable electronics applications. The module supports a data rate of up to 150 Mbps, and 20 dBm output power at the antenna to ensure the widest physical range. As such the module does offer industry-leading specifications and the best performance for electronic integration, range, power consumption, and connectivity.

The operating system chosen for ESP32 is freeRTOS with LwIP; TLS 1.2 with hardware acceleration is built in as well. Secure (encrypted) over the air (OTA) upgrade is also supported, so that users can upgrade their products even after their release, at minimum cost and effort.

Table 2 provides the specifications of ESP32-WROOM-32D and ESP32-WROOM-32U.

Table 2: ESP32-WROOM-32D and ESP32-WROOM-32U Specifications

Categories	Items	Specifications
Certification	RF Certification	FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC
	Wi-Fi Certification	Wi-Fi Alliance
	Bluetooth certification	BQB
	Green Certification	REACH/RoHS
Test	Reliability	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Frequency range	2.4 GHz ~ 2.5 GHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with –97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH
	Audio	CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I^2C, LED PWM, Motor PWM, I^2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC
	On-chip sensor	Hall sensor
	Integrated crystal	40 MHz crystal
	Integrated SPI flash [1]	4 MB
	Operating voltage/Power supply	2.7 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Recommended operating temperature range [2]	–40 °C ~ +85 °C

Notice:

1. ESP32-WROOM-32D and ESP32-WROOM-32U with 8 MB flash or 16 MB flash are available for custom order.
2. ESP32-WROOM-32D and ESP32-WROOM-32U with high temperature range (–40 °C ~ +105 °C) option are available for custom order. 4 MB SPI flash is supported on the high temperature range version.
3. For detailed ordering information, please see *Espressif Product Ordering Information*.

5. Electrical Characteristics

5.1 Absolute Maximum Ratings

Stresses beyond the absolute maximum ratings listed in Table 5 below may cause permanent damage to the device. These are stress ratings only, and do not refer to the functional operation of the device that should follow the recommended operating conditions.

Table 5: Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
VDD33	Power supply voltage	–0.3	3.6	V
I_{output} [1]	Cumulative IO output current	-	1,100	mA
T_{store}	Storage temperature	–40	150	°C

1. The module worked properly after a 24-hour test in ambient temperature at 25 °C, and the IOs in three domains (VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO) output high logic level to ground. Please note that pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.
2. Please see Appendix IO_MUX of ESP32 Datasheet for IO's power domain.

5.2 Recommended Operating Conditions

Table 6: Recommended Operating Conditions

Symbol	Parameter	Min	Typical	Max	Unit
VDD33	Power supply voltage	2.7	3.3	3.6	V
$I_{V\ DD}$	Current delivered by external power supply	0.5	-	-	A
T	Operating temperature	–40	-	85	°C

5.3 DC Characteristics (3.3 V, 25 °C)

Table 7: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter		Min	Typ	Max	Unit
C_{IN}	Pin capacitance		-	2	-	pF
V_{IH}	High-level input voltage		0.75×VDD[1]	-	VDD[1]+0.3	V
V_{IL}	Low-level input voltage		–0.3	-	0.25×VDD[1]	V
I_{IH}	High-level input current		-	-	50	nA
I_{IL}	Low-level input current		-	-	50	nA
V_{OH}	High-level output voltage		0.8×VDD[1]	-	-	V
V_{OL}	Low-level output voltage		-	-	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	-	40	-	mA
		VDD3P3_RTC power domain [1, 2]	-	40	-	mA
		VDD_SDIO power domain [1, 3]	-	20	-	mA

Symbol	Parameter	Min	Typ	Max	Unit
I_{OL}	Low-level sink current (VDD[1] = 3.3 V, V_{OL} = 0.495 V, output drive strength set to the maximum)	-	28	-	mA
R_{PU}	Resistance of internal pull-up resistor	-	45	-	kΩ
R_{PD}	Resistance of internal pull-down resistor	-	45	-	kΩ
V_{IL_nRST}	Low-level input voltage of CHIP_PU to power off the chip	-	-	0.6	V

Notes:

1. Please see Appendix IO_MUX of ESP32 Datasheet for IO's power domain. VDD is the I/O voltage for a particular power domain of pins.
2. For VDD3P3_CPU and VDD3P3_RTC power domain, per-pin current sourced in the same domain is gradually reduced from around 40 mA to around 29 mA, V_{OH} >=2.64 V, as the number of current-source pins increases.
3. Pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.

5.4 Wi-Fi Radio

Table 8: Wi-Fi Radio Characteristics

Parameter	Condition	Min	Typical	Max	Unit
Input frequency	-	2412	-	2484	MHz
Output impedance*	-	-	*	-	Ω
TX power	11n, MCS7	12	13	14	dBm
	11b mode	17.5	18.5	20	dBm
Sensitivity	11b, 1 Mbps	-	–98	-	dBm
	11b, 11 Mbps	-	–89	-	dBm
	11g, 6 Mbps	-	–92	-	dBm
	11g, 54 Mbps	-	–74	-	dBm
	11n, HT20, MCS0	-	–91	-	dBm
	11n, HT20, MCS7	-	–71	-	dBm
	11n, HT40, MCS0	-	–89	-	dBm
	11n, HT40, MCS7	-	–69	-	dBm
Adjacent channel rejection	11g, 6 Mbps	-	31	-	dB
	11g, 54 Mbps	-	14	-	dB
	11n, HT20, MCS0	-	31	-	dB
	11n, HT20, MCS7	-	13	-	dB

* For the modules that use IPEX antennas, the output impedance is 50 Ω. For other modules without IPEX antennas, users do not need to concern about the output impedance.

ANEXO 3: CÓDIGO FUENTE DEL SERVIDOR

```
#include <Arduino.h>                                    //  Inluye librería arduino
#include <WiFi.h>                                       //  Incluye librería WiFi
#include <WiFiUdp.h>                                    //  Incluye librería UDP
#include <coap.h>                                       //  Incluye librería coap
#include <esp_wifi.h>           //  Incluye librería para configurar el wifi en el esp
#include <U8g2lib.h>            //  Incluye librería para el el display

#ifdef U8X8_HAVE_HW_SPI     //  Si se va a usar SPI para manejar el display
#include <SPI.h>                //  Incluye librería SPI
#endif                          //  finaliza
#ifdef U8X8_HAVE_HW_I2C     //  Si se va a emplear I2C para manejar el display
#include <Wire.h>               //  Incluye la librería Wire
#endif                          //  finaliza

U8G2_PCD8544_84X48_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 5, /* dc=*/ 2, /*
reset=*/ 4);                    //  Nokia 5110 Display

#include "Adafruit_MQTT.h"      //  Incluye librería Adafruit MQTT
#include "Adafruit_MQTT_FONA.h"   //  Incluye librería Adafruit MQTT FONA
#define SIMCOM_7000               //  SIM7000

// DEFINICIÓN DE PINES
// PARA SIM7000
#define FONA_PWRKEY 0           //
#define FONA_RST 9              //
#define FONA_TX 16              //  RX del microcontrolador
#define FONA_RX 17              //  TX del microcontrolador

#define LED 13                  //  Solo para probar si es necesario

#define samplingRate 1     //  El tiempo que queremos retrazar despues de cada publicación
(en segundos)

#define fonaSS Serial2      //  Comunicación serial

// Para módulos LTE CAT-M/NB-IoT
// No se incluye el pin de reinicio porque está reservado para emergencias en el módulo
LTE
  Adafruit_FONA_LTE fona = Adafruit_FONA_LTE();      //  Declara librería LTE

// CONFIGURA MQTT
#define AIO_SERVER      "85.119.83.194"    //  Servidor MQTT
#define AIO_SERVERPORT  1883               //  Puerto

Adafruit_MQTT_FONA mqtt(&fona, AIO_SERVER, AIO_SERVERPORT);  //
Configure la clase MQTT FONA, el servidor MQTT y los detalles de inicio de sesión

uint8_t txfailures = 0;  //  Cuantas transmisiones fallidas seguidas se permiten antes de
reiniciar
```

```
// MQTT FEEDS
Adafruit_MQTT_Publish feed_location = Adafruit_MQTT_Publish(&mqtt,
"JRO_SATH_HUAYCOLORO/RSSI");   //  Tópico para ubicación
Adafruit_MQTT_Publish feed_data = Adafruit_MQTT_Publish(&mqtt,
"JRO_SATH/DATA");                    //  Tópico para datos CoAP
Adafruit_MQTT_Publish feed_dbm = Adafruit_MQTT_Publish(&mqtt,
"JRO_SATH/dbm");                     //  Tópico para RSSI del módulo de telefonía

// PARÁMETROS
char imei[16] = {0};       //  Para identificar el dispositivo
uint8_t type;              //  Declara entero de un byte para el tipo de módulo
uint16_t battLevel = 0;     //  Nivel de batería (porcentaje)
float latitude, longitude, speed_kph, heading, altitude, second;       //  Parámetros GPS
uint16_t year;                                                         //  variable año
uint8_t month, day, hour, minute;                                      //  Configura fecha
uint8_t counter = 0;                                                   //

char latBuff[100], longBuff[100], locBuff[100], speedBuff[100],    // almacena variables
tipo char
    headBuff[100], altBuff[100], battBuff[100],
    timeBuff[100], yearBuff[100], monthBuff[100], dayBuff[100],
    hourBuff[100], minuteBuff[100] , secondBuff[100];
int rssi = 99;                  //  almacena rssi en un dato de 2 bytes
int dbm = -115;                 //  almacena dbm en un dato de 2 bytes

//configuración servidor
const char* ssid = "kkkkk";                 //  AP ssid
const char* password = "12345678";          //  AP password

IPAddress local_ip(192, 168, 4, 1);         //  ip fija del servidor
IPAddress gateway(192, 168, 4, 1);          //  ip del gateway
IPAddress subnet(255, 255, 255, 0);         //  máscara
IPAddress local_dns(192, 168, 4, 1);        //  dns local

void callback_response(CoapPacket &packet, IPAddress ip, int port); //  CoAP client
response callback

void callback_rssi(CoapPacket &packet, IPAddress ip, int port); //  CoAP server endpoint
url callback

// UDP and CoAP class
WiFiUDP udp;   //  Crea la clase WiFiUDP para enviar y recibir mensajes UDP
Coap coap(udp); //  Crea la clase coap usando udp
unsigned long coap_timeout = 0;   //  Almacena variable coap_timeout en variable de 4
bytes
unsigned long mqtt_timeout = 0;  //  Almacena variable mqtt_timeout en variable de 4
bytes
int sample_count = 0;         //
String modemInfo;             //  variable información del modem como string
```

```
char message_char[100];    //  almacena el mensaje de coap
String buffer_coap = "";    //  buffer coap tipo string

// CoAP server endpoint URL
void callback_rssi(CoapPacket &packet, IPAddress ip, int port)  //  prepara la clase coap
{
  IPAddress ip_client = udp.remoteIP();  //  asigna ip de clientes
  u8g2.clearDisplay();                    //  limpia pantalla lcd
  u8g2.setCursor(0, 7);                   //  posición cursor en pantalla lcd
  u8g2.print("SERVER UP  ");              //  imprime en pantalla
  u8g2.setCursor(0, 14);                  //  posición cursor en pantalla lcd
  u8g2.print(modemInfo);                  //  imprime en pantalla
  u8g2.setCursor(0, 21);                  //  posición cursor en pantalla lcd
  u8g2.print(WiFi.softAPIP());            //  imprime ip del access point
  u8g2.setCursor(0, 28);                  //  posición cursor en pantalla lcd
  u8g2.print("CLIENT ONLINE");            //  imprime en pantalla
  u8g2.setCursor(0, 35);                  //  posición cursor en pantalla lcd
  u8g2.print(ip_client);                  //  imprime en la pantalla lcd la ip del cliente
  u8g2.setCursor(0, 42);                  //  posición cursor en pantalla lcd
  u8g2.print("MESSAGE:");                 //  imprime en pantalla lcd
  Serial.println(F("Potencia cliente"));  //  imprime en el monitor serial
  Serial.println(udp.remoteIP());         //  imprime en el monitor serial

 // send response
  char q[packet.payloadlen + 1];        //  almacena carácter "q" en un tipo de dato char
  memcpy(q, packet.payload, packet.payloadlen);  //  copia payload al carácter "q"
  q[packet.payloadlen] = NULL;          //  almacena en carácter "q"

  String message(q);                    //  establece variable q como string
  message.reserve(100);                 //
  message.remove(message.indexOf(" Sensor:"));  //  remueve dato del sensor
  message.remove(0,6);                          //  ubica a partir de dónde retirar
  buffer_coap.concat(ip_client[0]);             //  concatena ip
  buffer_coap.concat(".");                //  concatena ip
  buffer_coap.concat(ip_client[1]);       //  concatena ip
  buffer_coap.concat(".");                //  concatena ip
  buffer_coap.concat(ip_client[2]);       //  concatena ip
  buffer_coap.concat(".");                //  concatena ip
  buffer_coap.concat(ip_client[3]);       //  concatena ip
  buffer_coap.concat(" RSSI:");           //  concatena texto "RSSI" estaba comentado
  buffer_coap.concat(message);            //  concatena mensaje coap
  buffer_coap.concat(" ");                //  concatena espacio
  char message_char[100];                 //  cambia mensaje a tipo de dato char?
  Serial.print(F("RSSI: "));              //  imprime en el monitor serial
  Serial.println(message);                //  imprime en el monitor serial
  u8g2.setCursor(0, 49);                  //  posición cursor en pantalla lcd
  u8g2.print("RSSI: ");                   //  imprime en pantalla lcd
  u8g2.print(message);                    //  imprime en pantalla lcd
  u8g2.sendBuffer();                      //  imprime en pantalla lcd?
  message.toCharArray(message_char,message.length());  //  cambia tipo de dato a arreglo
```

```
  coap.sendResponse(ip, port, packet.messageid, message_char);  //  coap envia respuesta
  coap_timeout = millis();   //  reinicio de variable
  sample_count++;            //  no se usa coap_timeout
}

// CoAP client response callback
void callback_response(CoapPacket &packet, IPAddress ip, int port) { //  prepara clase
callback response y prepara ip y puerto
  Serial.println(F("[Coap Response got]"));  //  imprime en pantalla

  char q[packet.payloadlen + 1];     //  almacena carácter "q" en un tipo de dato char
  memcpy(q, packet.payload, packet.payloadlen);  //  copia payload al carácter "q"
  q[packet.payloadlen] = NULL;  //  almacena en carácter "q"

  Serial.println(q);   //  imprime en monitor serial "q" que es la variable donde coap
almacena el dato
}

void setup() {                    //  inicia el setup

  modemInfo.reserve(20);      //  reserva información para mostrar información del modem
  buffer_coap.reserve(2000); //  reserva memoria en buffer
  Serial.begin(2000000);        //  velocidad de baudios
  Serial.println(F("Server" )); //  imprime en el monitor serial
  pinMode(2, OUTPUT);       //  para depuración

  u8g2.begin();                   //  inicia pantalla lcd
  u8g2.clearBuffer();             //  limpia carácteres en pantalla
  u8g2.setFont(u8g2_font_5x7_mr);  //  tipo de fuente
  u8g2.setCursor(0, 7);                    //  ubica cursor en pantalla
  u8g2.print("SERVER DOWN");    //  imprime en pantalla
  u8g2.sendBuffer();                       //  imprime en pantalla lcd

  WiFi.mode( WIFI_AP_STA );       //  para modo access point
  int a= esp_wifi_set_protocol( WIFI_IF_AP, WIFI_PROTOCOL_LR ); //  habilita modo
LR
  if (a==0)                        //  si variable "a" toma valor "0"
  {
    Serial.println(F(" "));         //  imprime espacio
    Serial.print(F("Error = "));  //  imprime en monitor serial
    Serial.print(a);                //  imprime en monitor serial
    Serial.println(F(" , Mode LR OK!")); //  imprime en monitor serial

    u8g2.setCursor(0, 14);            //  ubica cursor en pantalla lcd
    u8g2.print("Mode LR OK!");  //  imprime en pantalla
    u8g2.sendBuffer();              //  imprime en pantalla
  }
  Else                              //  if hay algún error en la configuración LR
  {
    Serial.println(F(" "));          //  imprime espacio
```

```
    Serial.print(F("Error = "));    //  imprime error
    Serial.print(a);                                    //  imprime en monitor serial
    Serial.println(F(" , Error in Mode LR!"));          //  imprime en monitor serial
    u8g2.setCursor(0, 14);                              //  ubica cursor en pantalla lcd
    u8g2.print("Mode LR ERROR!");                       //  imprime en pantalla
    u8g2.sendBuffer();                                  //  imprime en pantalla
  }
  WiFi.softAPConfig(local_ip, gateway, subnet);       //  configura parámetros del wifi
  WiFi.softAP(ssid, password);                         //  configura id y contraseña de wifi
  u8g2.setCursor(0, 21);                               //  ubica cursor en pantalla lcd
  u8g2.print(WiFi.softAPIP());                         //  muestra ip en pantalla lcd
  u8g2.sendBuffer();                                   //  imprime en pantalla
  Serial.println( WiFi.softAPIP() );                   //  imprime ip en el monitor serial
  Serial.println(F("#"));                              //  para la depuración-for debug
  delay( 1000 );                                       //  retraso

// añade servidores url endpoints.
  Serial.println(F("Setup Callback rssi"));     //  imprime en monitor serial
  coap.server(callback_rssi, "rssi");           //  configura callback en el server coap
  // cliente responde callback.
  // este endpoint es un único callback.
  Serial.println(F("Setup Response Callback"));   //  imprime en monitor serial
  coap.response(callback_response);               //  coap responde al callback

  // start coap server/client
  coap.start();                                   //  inicia servidor coap

  u8g2.setCursor(0, 7);                           //  imprime en pantalla
  u8g2.print("SERVER UP  ");                      //  imprime en pantalla
  u8g2.sendBuffer();                              //  imprime en pantalla

  Serial.println(F("*** SIMCom Module MQTT ***")); //  imprime en monitor serial

  pinMode(FONA_RST, OUTPUT);                      //  define pin como salida
  digitalWrite(FONA_RST, HIGH);                   //  estado predeterminado Default state

  pinMode(FONA_PWRKEY, OUTPUT);                   //  estado predeterminado
  powerOn();                                      //  Encender el módulo
  delay(2000);                                    //  retraso de 2s
  moduleSetup();       //  Establece por primera vez el serial de comunicaciones e
                       //  imprime el imei, establece comunicación serial por primera vez
                       //  y imprime IMEI
  // Establecer el módem en plena funcionalidad
  fona.setFunctionality(1);               //  ejecuta comando AT+CFUN=1

  fona.setNetworkSettings(F("claro.pe")); //  Para Claro (Perú) SIM card - CAT-M1 (Band
28)

  if (!fona.enableGPS(true)) {                        //  si no habilita gps
    Serial.println(F("Failed to turn on GPS"));       //  imprime error
```

```
    delay(2000);                                                    //  vuelve a intentar cada 2 s
  }

  while (!netStatus()) {                                            //  establece condicional
    Serial.println(F("Failed to connect to cell network, retrying...")); //  imprime en monitor
serial
    delay(2000);                                                    //  Reintentar cada 2s
  }
  Serial.println(F("Connected to cell network!"));                  //  imprime en monitor serial

  delay(1000);                                                      //  retraso 1s

  fona.getReply("AT+CREG=2");                                       //  estado del registro de la red
  fona.getReply("AT+CNSMOD=1");                                     //  modo del sistema de la red
  fona.getReply("AT+CRRCSTATE=1");                                  //  estado rrc
  fona.getReply("AT+CNBP?");                                        //  estado de la banda
  fona.getReply("AT+CMNB?");                                        //  selección cat-m1 o nb
  fona.getReply("AT+CBANDCFG=?");                                   //  configuración cat-m1 o nb
  fona.getReply("AT+CBANDCFG?");                                    //
  fona.getReply("AT+CNMP?");                                        //  selección de modo preferido
  fona.getReply("AT+CAPNMODE?");                                    //  configura apn
  fona.getReply("AT+CGACT?");                       //  pdp contexto activar o desactivar
  fona.getReply("AT+SAPBR=1,1");  //  configura portador para aolicaciones basadas en ip
  delay(3000);                                                      //  retraso 3 seg.
  fona.getReply("AT+SAPBR=2,1");   //  configura portador para aplicaciones basadas en
ip
  delay(5000);                                                      //  retraso 5 seg.
  fona.getReply("AT+CGNAPN");              //  obtener apn de la red cat-m1 o nb
  fona.getReply("AT+CGPADDR");             //  mostrar dirección pdp
  fona.getReply("AT+CGDCONT?");                                     //  define el contexto del pdp

  delay(2000);                                                      //  retraso 2s
  coap_timeout = millis();                          //  inicializa variable coap_timeout
  mqtt_timeout = millis();                          //  inicializa variable mqtt_timeout
  buffer_coap = "";                                 //  establece memoria buffer para coap

}

void loop() {                                                       //  inicia loop

  coap.loop()                                                       //  registra datos coap

  if ((millis()-coap_timeout)>=2000)  //  Entra al "if" si no recibe datos por más de 2
segundos
  {
    coap_timeout = millis();    //  escribe el tiempo actual en milis en la variable
coap_timeout
    u8g2.clearDisplay();                                            //  borra carácteres en el display
    u8g2.setCursor(0, 7);                                           //  ubica cursor en pantalla lcd
    u8g2.print("SERVER UP");                                        //  imprime en pantalla lcd
```

```
  u8g2.setCursor(0, 14);                                  //  ubica cursor en pantalla lcd
  u8g2.print(modemInfo);                                  //  imprime en pantalla lcd?
  u8g2.setCursor(0, 21);                                  //  ubica cursor en pantalla lcd
  u8g2.print(WiFi.softAPIP());                            //  imprime en pantalla ip
  u8g2.setCursor(0, 28);                                  //  ubica cursor en pantalla lcd
  u8g2.print("Listening...");                             //  imprime en pantalla
  u8g2.sendBuffer();                                      //  imprime en pantalla
  sample_count = 0;                                       //  no se usa
 }

 if ((millis()-mqtt_timeout) >= 0)   //  Entra al "if" si no recibe datos por más de 0 segundos
 {
  // Conecta a la red celular y verifica la conexión
  // Si no tiene éxito, If unsuccessful, vuelva a intentarlo cada 2s hasta que se realice una conexión
  int registered = 1;                 //  registro toma valor "1"
  while (!netStatus()) {              //  mientras el estado de la red sea errada
   Serial.println(F("Failed to connect to cell network, retrying..."));  //  imprime en monitor serial
   delay(2000);                                                          //  reintenta cada 2s
  registered = 0;                                                        //  hasta que registre
  }
  Serial.println(F("Connected to cell network!"));          //  imprime en monitor serial

  if (registered == 0)                                      //  condicional, si registró
  {
   delay(1000);                                             //  retraso 1s
   fona.getReply("AT+CREG=2");                              //  estado del registro de la red
   fona.getReply("AT+CNSMOD=1");                            //  modo del sistema de la red
   fona.getReply("AT+CRRCSTATE=1");                         //  estado rrc
   fona.getReply("AT+CNBP?");                               //  estado de la banda
   fona.getReply("AT+CMNB?");                               //  selección cat-m1 o nb
   fona.getReply("AT+CBANDCFG=?");                          //  configuración cat-m1 o nb
   fona.getReply("AT+CBANDCFG?");                           //  obtener respuesta
   fona.getReply("AT+CNMP?");                               //  selección de modo preferido
   fona.getReply("AT+CAPNMODE?");                           //  configura apn
   fona.getReply("AT+CGACT?");           //  pdp contexto activar o desactivar
   fona.getReply("AT+SAPBR=1,1"); //  configura portador para aplicaciones basadas en ip
   delay(3000);                                             //  retraso 3s
   fona.getReply("AT+SAPBR=2,1"); //  configura portador para aplicaciones basadas en ip
   delay(5000);                                             //  retraso 5s
   fona.getReply("AT+CGNAPN");     //  obtener apn de la red cat-m1 o nb
   fona.getReply("AT+CGPADDR");   //  mostrar dirección pdp
   fona.getReply("AT+CGDCONT?"); //  define el contexto del pdp
   delay(1000);                                             //  retraso 1s
  }
```

```
  battLevel = readVcc();                                    // obtener voltaje en mV

  if (!fona.enableGPS(true)) {          //  condicional, si GPS está desactivado
    Serial.println(F("Failed to turn on GPS")); //  imprimir en monitor serial
  }

  if (!fona.getGPS(&latitude, &longitude, &speed_kph, &heading, &altitude, &year,
&month, &day, &hour, &minute, &second)) { //  condición, obtener ubicación, tiempo
UTC
    Serial.println(F("Failed to get GPS location."));          //  imprimir en monitor serial
  }

  Serial.println(F("Found 'eeeeem!"));                        //  imprimir en monitor serial
  Serial.println(F("----------------------"));                  //  imprimir en monitor serial
  Serial.print(F("Latitude: ")); Serial.println(latitude, 6);     //  imprimir en monitor serial
  Serial.print(F("Longitude: ")); Serial.println(longitude, 6); //  imprimir en monitor serial
  Serial.print(F("Speed: ")); Serial.println(speed_kph);       //  imprimir en monitor serial
  Serial.print(F("Heading: ")); Serial.println(heading);       //  imprimir en monitor serial
  Serial.print(F("Altitude: ")); Serial.println(altitude);       //  imprimir en monitor serial

  Serial.print(F("Year: ")); Serial.println(year);              //  imprimir en monitor serial
  Serial.print(F("Month: ")); Serial.println(month);            //  imprimir en monitor serial
  Serial.print(F("Day: ")); Serial.println(day);                //  imprimir en monitor serial
  Serial.print(F("Hour: ")); Serial.println(hour);              //  imprimir en monitor serial
  Serial.print(F("Minute: ")); Serial.println(minute);          //  imprimir en monitor serial
  Serial.print(F("Second: ")); Serial.println(second);          //  imprimir en monitor serial

  Serial.println(F("----------------------"));                  //  imprimir en monitor serial

  rssi = fona.getRSSI();          //  obtener rssi del módulo de telefonía
  dbm = -115;                     //  establece limite de dbm

  if (rssi == 0) dbm = -115;                                                  //  rango
  if (rssi == 1) dbm = -111;                                                  //  rango
  if (rssi == 31) dbm = -52;                                                  //  rango
  if ((rssi >= 2) && (rssi <= 30)) {                                          //  condicional
    dbm = map(rssi, 2, 30, -109, -53); //  establece rangos para el rssi con respecto a los
dbm
  }
  Serial.print(F("RSSI = ")); Serial.print(dbm); Serial.print(F(": ")); Serial.println(F("
dBm"));              //  imprime en monitor serial
                       //  Formato de los números punto flotante
  dtostrf(latitude, 1, 6, latBuff);                           //  latitud
  dtostrf(longitude, 1, 6, longBuff);                         //  longitud
  dtostrf(speed_kph, 1, 0, speedBuff);                        //  velocidad
  dtostrf(dbm, 1, 0, headBuff);                               //  rssi módem
  Serial.print(F("TEST: headBuff = "));                       //  imprime en monitor serial
  Serial.println(headBuff);                                   //  muestra rssi módem
  dtostrf(altitude, 1, 1, altBuff);                           //  altitud
  dtostrf(battLevel, 1, 0, battBuff);                         //  nivel de batería
```

```
    dtostrf(year, 1, 0, yearBuff);                                  //  año
    dtostrf(month, 1, 0, monthBuff);                                //  mes
    dtostrf(day, 1, 0, dayBuff);                                    //  día
    dtostrf(hour, 1, 0, hourBuff);                                  //  hora
    dtostrf(minute, 1, 0, minuteBuff);                              //  minuto
    dtostrf(second, 1, 0, secondBuff);                              //  segundo
//  construir matriz de ubicación separada por comas
    sprintf(locBuff, "%s,%s,%s,%s (speed,latitude,longitude,altitude)", speedBuff, latBuff,
longBuff, altBuff);                                                 //  ubicación
    sprintf(timeBuff,"%s-%s-%s %s:%s:%s UTC
time",dayBuff,monthBuff,yearBuff,hourBuff,minuteBuff,secondBuff);       //  tiempo

    MQTT_connect();                                                 //  Mqtt conecta automáticamente
  Serial.println(F("Buffer test:"));                                //  imprime en monitor serial
  Serial.println(buffer_coap);          //  imprime en monitor serial datos recibidos de coap

  char *buffer_coap_char = new char[buffer_coap.length() + 1]; //  cambia de variable char
  strcpy(buffer_coap_char, buffer_coap.c_str());                    //  copia string
  buffer_coap = "";                   //  almacena datos actuales de coap

  Serial.println(buffer_coap_char);    //  imprime en monitor serial contenido coap

    MQTT_publish_checkSuccess(feed_location, locBuff, strlen(locBuff)); //  publica
ubicación
    delay(1000);                                                    //  retraso 1s
    MQTT_publish_checkSuccess(feed_dbm, headBuff, strlen(headBuff));  //  publica rssi
    delay(1000);                                                    //  retraso 1s
    MQTT_publish_checkSuccess(feed_location, timeBuff, strlen(timeBuff)); //  publica
hora
    delay(1000);                                                    //  retraso 1s
    MQTT_publish_checkSuccess(feed_data, buffer_coap_char, strlen(buffer_coap_char));
                                                                    //  publica datos del coap
    delay(1000);                                                    //  retraso 1s

    mqtt_timeout = millis();   //  escribe el tiempo actual en milis en la variable
mqtt_timeout
  }
}

void powerOn() {                                                    //  encender el módulo
  digitalWrite(FONA_PWRKEY, LOW);                                   //  estado bajo
  #if defined(SIMCOM_7000)                                          //  para sim7000
    delay(2000);                                                    //  retrasar 2s
  #endif                                                            //  finaliza función

  digitalWrite(FONA_PWRKEY, HIGH);                                  //  estado alto
}

void moduleSetup() {                                                //  configuración del módulo
```

```
fonaSS.begin(9600);          //  tasa de baudios por defecto del sim7000

Serial.println(F("Configuring to 115200 baud"));  //  imprime en monitor serial
fonaSS.println("AT+IPR=115200");  //  establece velocidad de transmisión
delay(100);                       //  breve pausa para dejar correr el comando
fonaSS.begin(115200);                                   //  inicia comunicación serial
while (! fona.begin(fonaSS)) {                          //  condición
  Serial.println(F("Couldn't find FONA"));              //  imprime en el monitor serial
delay(1000);                                            //  retraso 1s
}

type = fona.type();                  //  configura type para seleccionar el tipo de módulo
Serial.println(F("FONA is OK"));                        //  imprime en el monitor serial
Serial.print(F("Found "));                              //  imprime en el monitor serial
switch (type) {                                         //  selecciona el tipo de módulo
  case SIM800L:                                         //  para sim800L
    Serial.println(F("SIM800L")); break;                //  interrumpir
  case SIM800H:                                         //  para SIM800H
    Serial.println(F("SIM800H")); break;                //  interrumpir
  case SIM808_V1:                                       //  SIM808_V1
    Serial.println(F("SIM808 (v1)")); break;            //  interrumpir
  case SIM808_V2:                                       //  para SIM808_V2
    Serial.println(F("SIM808 (v2)")); break;            //  interrumpir
  case SIM5320A:                                        //  para SIM5320A
    Serial.println(F("SIM5320A (American)")); break;    //  interrumpir
  case SIM5320E:                                        //  para SIM5320E
    Serial.println(F("SIM5320E (European)")); break;    //  interrumpir
  case SIM7000A:                                        //  para SIM7000A
    Serial.println(F("SIM7000A (American)")); break;    //  interrumpir
  case SIM7000C:                                        //  para SIM7000C
    Serial.println(F("SIM7000C (Chinese)")); break;     //  interrumpir
  case SIM7000E:                                        //  para SIM7000E
    Serial.println(F("SIM7000E (European)")); break;    //  interrumpir
  case SIM7000G:                                        //  para SIM7000G
    Serial.println(F("SIM7000G (Global)")); break;      //  interrumpir
  case SIM7500A:                                        //  para SIM7500A
    Serial.println(F("SIM7500A (American)")); break;    //  interrumpir
  case SIM7500E:                                        //  para SIM7500E
    Serial.println(F("SIM7500E (European)")); break;    //  interrumpir
  default:                                              //  defecto
    Serial.println(F("???")); break;                    //  interrumpir
}

// Print module IMEI number.
uint8_t imeiLen = fona.getIMEI(imei);    //  obtener número imei del módulo
if (imeiLen > 0) {                                      //  condicional
  Serial.print(F("Module IMEI: ")); Serial.println(imei);   //  imprime en el monitor serial
}
else                                                    //  sino
{
```

```
    Serial.print(F("Invalid IMEI"));                    //  imprime en el monitor serial
  }

  rssi = fona.getRSSI();                                //  obtener rssi del módulo de telefonía
  dbm = -115;                                           //  asigna a dbm un valor límite

  Serial.print(F("RSSI = ")); Serial.print(dbm); Serial.print(F(": ")); Serial.println(F("
dBm"));                                                 //  Imprime en el monitor serial
  if (rssi == 0) dbm = -115;                            //  rango
  if (rssi == 1) dbm = -111;                            //  rango
  if (rssi == 31) dbm = -52;                            //  rango
  if ((rssi >= 2) && (rssi <= 30)) {                    //  condicional
    dbm = map(rssi, 2, 30, -109, -53);      //  establece rango de rssi respecto a dbm
  }
}

float readVcc() {                       //  lee la tensión de alimentación del módulo
  if (!fona.getBattVoltage(&battLevel)) Serial.println(F("Failed to read batt"));
                                                        //  leer voltaje de batería
  else Serial.print(F("battery = ")); Serial.print(battLevel); Serial.println(F(" mV")); //
  return battLevel;                                     //  retorna nivel de voltaje
}

bool netStatus() {                              //  estado de la red
  int n = fona.getNetworkStatus();              //  variable para el estado de la red

  Serial.print(F("Network status ")); Serial.print(n); Serial.print(F(": "));     //  imprime en
pantalla
  if (n == 0) Serial.println(F("Not registered"));                                 //  caso 1
  if (n == 1) Serial.println(F("Registered (home)"));                              //  caso 2
  if (n == 2) Serial.println(F("Not registered (searching)"));                     //  caso 3
  if (n == 3) Serial.println(F("Denied"));                                        //  caso 4
  if (n == 4) Serial.println(F("Unknown"));                                       //  caso 5
  if (n == 5) Serial.println(F("Registered roaming"));                            //  caso 6

  if (!(n == 1 || n == 5)) return false;                //  si "n" no toma ningun valor
  else return true;                                     //  retorna verdad
}

void MQTT_connect() {           //  función para conectar y reconectar según sea necesario al
servidor MQTT
  int8_t ret;                   //  almacena variable "ret" en variable de 1 byte

  if (mqtt.connected()) {                               //  parar si ya está conectado
    return;                                             //  termina función
  }

  Serial.println(F("Connecting to MQTT... "));          //  imprimir en pantalla

  while (ret = mqtt.connect() != 0) {           //  connect devolverá cero para conectado
```

```
   Serial.println(mqtt.connectErrorString(ret));                  //  imprimir en el monitor serial
   Serial.println(F("Retrying MQTT connection in 3 seconds..."));  //  imprimir en el
monitor serial
   mqtt.disconnect();                                             //  mqtt desconectado
   delay(3000);                                                   //  espera 5s
  }
  Serial.println(F("MQTT Connected!"));                           //  imprimir en el monitor serial
}

void MQTT_publish_checkSuccess(Adafruit_MQTT_Publish &feed, const char
*feedContent, uint16_t bLen) {                                    //  publica en el servidor mqtt
  Serial.println(F("Sending data..."));                           //  imprimir en el monitor serial
  if (! feed.publish((uint8_t *)feedContent,bLen+1)) {            //  si hay error en publicar
    Serial.println(F("Publish failed"));                          //  imprimir en el monitor serial
    txfailures++;                                                 //  incrementa transmisión fallida en 1
  }
  else {                                                          //  sino
    Serial.println(F("Publish OK!"));                             //  imprimir en el monitor serial
    txfailures = 0;                                               //  transmisiones fallidas igual a 0
  }
}
```

ANEXO 4: CÓDIGO FUENTE DEL CLIENTE

```
#include <Arduino.h>                                    //  incluye librería arduino
#include <WiFi.h>                                       //  incluye librería wifi
#include <WiFiUdp.h>                                    //  incluye librería UDP
#include <coap.h>                                       //  incluye librería coap
#include <esp_wifi.h>                                   //  incluye librería wifi esp
#include <U8g2lib.h>                          //  incluye librería Nokia 5110 Display

#ifdef U8X8_HAVE_HW_SPI                       //  si comunicación serial es por SPI
#include <SPI.h>                                        //  incluir librería SPI
#endif                                                  //  condicional
#ifdef U8X8_HAVE_HW_I2C                       //  si comunicación serial es por I2C
#include <Wire.h>                                       //  incluir librería Wire
#endif                                                  //  condicional

U8G2_PCD8544_84X48_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 5, /* dc=*/ 2, /*
reset=*/ 4);                                            // Nokia 5110 Display

const char* ssid = "kkkkk";                             //  AP ssid
const char* password = "12345678";                      //  AP password

//IPAddress local_ip(192, 168, 4, 2);                   //ip cliente 1
IPAddress local_ip(192, 168, 4, 3);                     //  ip cliente 2
IPAddress gateway(192, 168, 4, 1);                      //  gateway
IPAddress subnet(255, 255, 255, 0);                     //  máscara
IPAddress local_dns(192, 168, 4, 1);                    //  dns
const char *toStr( wl_status_t status ) {               //  estado conexión
   switch( status ) {                                   //  casos
   case WL_NO_SHIELD: return "No shield";               //  caso 1
   case WL_IDLE_STATUS: return "Idle status";           //  caso 2
   case WL_NO_SSID_AVAIL: return "No SSID avail";   //  caso 3
   case WL_SCAN_COMPLETED: return "Scan compleded";       //  caso 4
   case WL_CONNECTED: return "CONNECTED";           //  caso 5
   case WL_CONNECT_FAILED: return "Failed";         //  caso 6
   case WL_CONNECTION_LOST: return "Connection lost";       //  caso 7
   case WL_DISCONNECTED: return "Disconnected";     //  caso 8
   }
   return "Unknown";                    //  retorna Unknown en caso no se de ningún caso
}

extern "C" int rom_phy_get_vdd33();             //  declara variable para obtener bateria

void callback_response(CoapPacket &packet, IPAddress ip, int port);        //  CoAP client
response callback

WiFiUDP udp;                    //  Crea la clase WiFiUDP para enviar y recibir mensajes UDP
Coap coap(udp);                 //  Crea la clase coap usando udp

void callback_response(CoapPacket &packet, IPAddress ip, int port)         // CoAP client
response callback
```

```
{
  Serial.print("Coap response: ");                          //  imprimir en monitor serial

  char q[packet.payloadlen + 1];              //  almacena carácter "q" en un tipo de dato char
  memcpy(q, packet.payload, packet.payloadlen);             //  copia payload al carácter "q"
  q[packet.payloadlen] = NULL;                              //  almacena en carácter "q"

  Serial.println(q);                                        //  imprime datos de coap?
}

void setup() {                                              //  inicia setup

  Serial.begin(2000000);                  //  valocidad de transmisión en microcontrolador
  u8g2.begin();                                             //  inicia pantalla lcd
  u8g2.clearBuffer();                                       //  imprime en pantalla lcd
  u8g2.setFont(u8g2_font_profont11_mf);            //  limpia carácteres en pantalla lcd
  u8g2.setCursor(0, 7);                                     //  ubica cursor en pantalla lcd
  u8g2.print("CLIENT OFFLINE");                             //  imprime en pantalla lcd
  u8g2.sendBuffer();                                        //  imprime en pantalla lcd
  Serial.println( "Client" );                               //  imprime en monitor serial


  WiFi.mode( WIFI_STA );                                    //  para modo estación
  int a= esp_wifi_set_protocol( WIFI_IF_STA, WIFI_PROTOCOL_LR );  //  configura
modo LR
    if (a==0)                                               //  si variable "a" toma valor "0"
    {
      Serial.println(" ");                                  //  imprimir en monitor serial
      Serial.print("Error = ");                             //  imprimir en monitor serial
      Serial.print(a);                                      //  imprimir en monitor serial
      Serial.println(" , Mode LR OK!");                     //  imprimir en monitor serial
      u8g2.setCursor(0, 16);                                //  imprime en pantalla lcd
      u8g2.print("Mode LR OK!");                            //  imprime en pantalla lcd
    }
    Else                                                    //  sino
    {
      Serial.println(" ");                                  //  imprimir en monitor serial
      Serial.print("Error = ");                             //  imprimir en monitor serial
      Serial.print(a);                                      //  imprimir en monitor serial
      Serial.println(" , Error in Mode LR!");               //  imprimir en monitor serial
      u8g2.setCursor(0, 16);                                //  imprime en pantalla lcd
      u8g2.print("Error in Mode LR!");                      //  imprime en pantalla lcd
    }
    WiFi.config(local_ip, gateway, subnet, local_dns, local_dns);     //  configura wifi
    WiFi.begin(ssid, password);                             //  este ssid no es visible

    u8g2.sendBuffer();                                      //  imprime en pantalla lcd
    int temp_count = 1;                          //  asigna a variable temp_count el valor 1
    u8g2.setCursor(0, 25);                                  //  posición en pantalla lcd
    u8g2.print("Connecting");                               //  imprimir en pantalla lcd
```

```
  while (WiFi.status() != WL_CONNECTED)  //  mientras el estado wifi sea no conectado
  {
    delay(1000);                                    //  retraso 1s
    Serial.print(".");                              //  imprime en monitor serial
    u8g2.print(".");                                //  imprime en pantalla lcd
    u8g2.sendBuffer();                              //  imprime en pantalla lcd
    if (temp_count==5)                        //  si la cantidad de intentos es igual a 5
    {
      temp_count = 1;                               //  primer intento
      u8g2.setCursor(0, 25);                     //  ubicar cursor en monitor serial
      u8g2.print("Connecting       ");             //  imprimir en pantalla lcd
      u8g2.sendBuffer();                            //  imprime en pantalla lcd
      u8g2.setCursor(0, 25);                     //  ubicar cursor en monitor serial
      u8g2.print("Connecting");                     //  imprimir en pantalla lcd
      u8g2.sendBuffer();                            //  imprime en pantalla lcd
    }
    Else                                            //  sino
    {
      temp_count++;                             //  variable temp_count incrementa en 1
    }
  }
  u8g2.clearDisplay();                          //  borra caracteres de pantalla lcd
  Serial.println("WiFi ONLINE");                    //  imprime en serial monitor
  Serial.print("IP address: ");                     //  imprime en serial monitor
  Serial.println(WiFi.localIP());                   //  imprime en serial monitor
  u8g2.setCursor(0, 7);                             //  ubica cursor en pantalla lcd
  u8g2.print("CLIENT ONLINE");                      //  imprime en pantalla lcd
  u8g2.setCursor(0, 16);                            //  ubica cursor en pantalla lcd
  u8g2.print(WiFi.localIP());                       //  imprime en pantalla lcd
  u8g2.sendBuffer();                                //  imprime en pantalla lcd
  Serial.println("Setup Response Callback");        //  imprime en el monitor serial
                                                  //  cliente responde al callback.
                                                 //  este endpoint es un solo callback.
  Serial.println("Setup Response Callback");        //  imprime en el monitor serial
  coap.response(callback_response);             //  respuesta a la devolución de llamada


 coap.start();                                      //  comienza coap
}

void loop() {                                       //  comienza loop
 if ( WiFi.status() != WL_CONNECTED )          //  si el estado de wifi es no conectado
 {
  Serial.println( "DisONLINE." );                   //  imprime en monitor serial
  u8g2.setFont(u8g2_font_profont11_mf);             //  imprime en pantalla lcd
  u8g2.clearDisplay();                          //  limpia caracteres en pantalla lcd
  u8g2.setCursor(0, 7);                             //  ubica cursor en pantalla lcd
  u8g2.print("CLIENT OFFLINE");                     //  imprime en monitor serial
  u8g2.setCursor(0, 16);                            //  ubica cursor en pantalla lcd
  u8g2.print("Connecting");                         //  imprime en pantalla lcd
```

```
  u8g2.sendBuffer();                                    //  imprime en pantalla lcd

  int tries = 0;                                        //  variable intentos igual a 0
  WiFi.begin( ssid, password );                         //  inicia wifi
  int x = 1;            //  variable x almacena el número de intentos, valor inicial igual a 1
  while( WiFi.status() != WL_CONNECTED ) {              //  condicional
    delay( 1000 );                                      //  retraso
    u8g2.print(".");                                    //  imprime en pantalla lcd
    u8g2.sendBuffer();                                  //  imprime en pantalla lcd
    if (x==5)                                           //  intentos igual a 5
    {
      u8g2.setCursor(0, 25);                            //  ubica cursor en pantalla lcd
      u8g2.print("Connecting       ");                  //  imprime en pantalla lcd
      u8g2.sendBuffer();                                //  imprime en pantalla lcd
      u8g2.setCursor(0, 25);                            //  ubica cursor en pantalla lcd
      u8g2.print("Connecting");                         //  imprime en pantalla lcd
      u8g2.sendBuffer();                                //  imprime en pantalla lcd
    }
    Else                                                //  sino
    {
      x++;                                              //  intentos aumenta en 1
    }
    tries++;                                            //  intentos aumenta en uno
    if ( tries == 5 )                                 //  concional, si intentos igual a 5
       return;                                          //  finaliza función
    Serial.println( toStr( WiFi.status() ) );           //  imprime en monitor serial
  }
  Serial.print( "ONLINE " );                            //  imprime en monitor serial
  Serial.println( WiFi.localIP() );                     //  imprime en monitor serial
  u8g2.clearDisplay();                                  //  borra caracteres en pantalla lcd
  u8g2.sendBuffer();                                    //  imprime en monitor serial
 }
 Else                                                   //  sino
 {
  u8g2.setFont(u8g2_font_profont11_mf);                 //  fuente
  u8g2.setCursor(0, 7);                                 //  ubica cursor en pantalla lcd
  u8g2.print("CLIENT ONLINE");                          //  imprime en pantalla lcd
  u8g2.setCursor(0, 16);                                //  ubica cursor en pantalla lcd
  u8g2.print(WiFi.localIP());                           //  imprime en pantalla lcd
  u8g2.setCursor(0, 40);                                //  ubica cursor en pantalla lcd
  u8g2.setFont(u8g2_font_10x20_mf);                     //  fuente
  u8g2.print("RSSI:");                                  //  imprime rssi en pantalla lcd
  u8g2.print(WiFi.RSSI());                              //  imprime dirección ip
  u8g2.print("      ");                                 //  imprime en patalla lcd
  u8g2.sendBuffer();                                    //  imprime en pantalla lcd
  String message = String(WiFi.RSSI());                 //  mensaje tipo string
  float sensor = 50.28;                                 //  dato aleatorio sensor
  message = "RSSI: " + message + " Sensor: " + String(sensor);     //  concatena trama del
mensaje
  char message_char[100];                       //  almacena valor actual message_char[100]
```

```
   message.toCharArray(message_char,message.length()+1);          //  Copia los caracteres de la cadena en el búfer suministrado.

   Serial.println("Send Request");                                 //  imprime en monitor serial
   int msgid = coap.put(IPAddress(192, 168, 4, 1), 5683, "rssi", message_char);  //  prepara mensaje a enviar por coap
   Serial.println(message_char);                                  //  imprime en pantalla lcd
   delay(500);                                                    //  retrazo de medio segundo
   coap.loop();                                                   //  inicia coap loop
  }
}
```

ANEXO 5: MATRIZ DE CONSISTENCIA

Tesista: Frank Enrique Meléndez Coveñas

Tesis: Diseño, implementación y testeo de una red de instrumentos inalámbricos con conexión directa a internet y de bajo costo para un sistema de alerta temprana de huaicos en la quebrada de Jicamarca

Diseño, implementación y testeo de una red de instrumentos inalámbricos con conexión directa a internet y de bajo costo para un sistema de alerta temprana de huaicos en la quebrada de Jicamarca			
Planteamiento del problema	Objetivos	Variables	Metodología
Problema general	Objetivo general	Operacionalización de las variables	Tipo de investigación
¿Cómo lograr que un sistema de alerta temprana de huaicos garantice la conexión permanente a internet y disminuya su costo de implementación y mantenimiento en la quebrada de Jicamarca?	Diseñar, implementar y testear una red de instrumentos inalámbricos con conexión directa a internet y con tecnologías de bajo costo en la quebrada de Jicamarca	- Red de instrumentos inalámbrica - Conexión directa a internet	Investigación aplicada o tecnológica
Problemas Específicos	Objetivos específicos	Indicadores de evaluación de resultados	Nivel de investigación
¿Qué tecnología garantiza la conexión permanente a internet de un sistema de alerta temprana de huaicos en la quebrada de Jicamarca? ¿De qué manera se disminuye el costo de implementación y mantenimiento de un sistema de alerta temprana de huaicos en la quebrada de Jicamarca?	- Diseñar, implementar y testear una red de instrumentos inalámbricos con conexión directa a internet. - Diseñar e implementar una red de instrumentos inalámbricos con tecnologías de bajo costo.	● Red de instrumentos inalámbrica: - Cobertura - Precio de componentes ● Conexión directa a internet - Latencia	Nivel experimental