Proyecto 2 Entrega 1: Detector de fugas de agua

Arantxa Espejo
Pontificia Universidad Javeriana
Facultad de Ingeniería
Bogotá, Colombia
arantxaespejo@javeriana.edu.co

Juan Manuel López
Pontificia Universidad Javeriana
Facultad de ingeniería
Bogotá, Colombia
jlopez@javeriana.edu.co

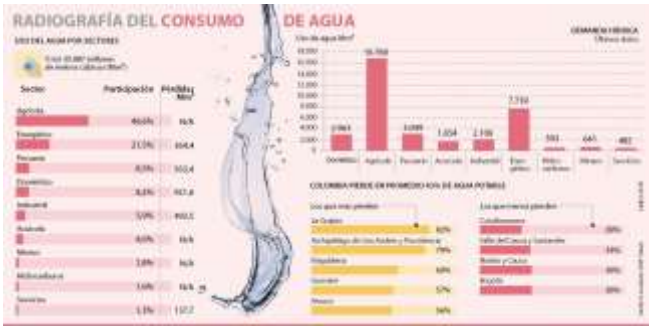
Karla Martínez
Pontificia Universidad Javeriana
Facultad de ingeniería
Bogotá, Colombia
karla.martinez@javeriana.edu.co

Resumen---Este proyecto presenta un sistema de detección de fugas de agua basado en el protocolo y una arquitectura IoT. Consta de nodos sensores inalámbricos (NodeMCU ESP8266) con sensores de agua, un broker MQTT central y una aplicación concentradora. Los nodos publican datos de detección de fugas en el broker. La aplicación concentradora recibe estos datos y activa alarmas publicando comandos en el broker. Los nodos suscritos accionan actuadores locales para alertar sobre fugas detectadas.

Palabras clave—Detector de fugas, MQTT, NodeMCU, ESP8266, IoT, Broker MQTT, monitoreo remoto, alarma de fugas.

A. Definición de la problemática

En Colombia, a pesar de ser uno de los países con mayores reservas de agua dulce, se enfrenta a una problemática preocupante: el desperdicio y las fugas de agua potable. Según datos del Departamento Nacional de Planeación, cuatro de cada 10 litros (aproximadamente el 43%) del agua potable producida en el país se pierde debido a fugas, rebosamientos, conexiones ilegales, errores de medición y otros factores. [1]

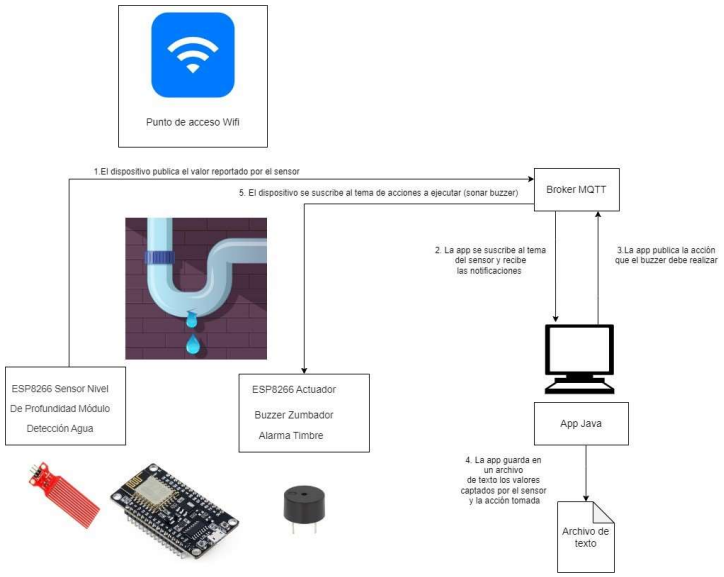


Fuente: Departamento Nacional de Planeación (DNP)

En este contexto, el proyecto propone desarrollar un sistema de detección y alerta de fugas de agua basado en el Internet de las Cosas (IoT). Este sistema utilizará sensores de agua estratégicamente ubicados en áreas propensas a fugas, como cerca de tuberías, electrodomésticos o sistemas de calefacción y refrigeración. Cada sensor estará conectado a un dispositivo individual que contará con un actuador, en este caso, un buzzer (alarma).

Los dispositivos con sensores estarán equipados con un módulo WiFi (NodeMCU con ESP8266) que les permitirá conectarse a una red de datos y enviar información sobre la detección de fugas de agua a una aplicación central desarrollada en Java, esta actuará como un concentrador de datos, recibiendo los mensajes de los sensores y determinando cuándo activar las alarmas correspondientes en cada dispositivo.
Mediante la implementación de este sistema de detección y alerta de fugas de agua basado en IoT, se busca minimizar el desperdicio de agua, reducir los daños causados por fugas no detectadas y facilitar una respuesta rápida y eficiente ante estas situaciones.

B. Diagrama esquemático de los dos dispositivos



Ver anexo 1

C. Presupuesto

- NodeMCU con microcontrolador ESP8266
  Precio: 12.000 COP enlace artículo

- Sensor agua Control De Nivel De Agua Para Arduino
  Precio: 4.950 COP enlace artículo

- Jumpers (Cables Arduino)
  Precio: 8.000 COP enlace artículo

- Buzzer (alarma)
  Precio: 15.000 COP enlace articulo

Total: 39.950 COP

D. Documentación del protocolo MQTT

Protocolo MQTT

MQTT es un protocolo de mensajería ligero y eficiente, diseñado para comunicaciones máquina a máquina (M2M) y para el Internet de las Cosas (IoT). Fue desarrollado por IBM y posteriormente estandarizado por OASIS. MQTT es ampliamente utilizado en aplicaciones donde se requiere una comunicación confiable y eficiente con un bajo consumo de recursos y ancho de banda.

Características clave de MQTT

- **Modelo de publicación/suscripción**: MQTT utiliza un modelo de comunicación de publicación/suscripción basado en temas (topics). Los clientes publican mensajes en temas específicos, y otros clientes se suscriben a los temas que les interesan para recibir los mensajes correspondientes.
- **Arquitectura descentralizada**: MQTT emplea un intermediario central llamado broker, que actúa como un punto de comunicación para todos los clientes. Los clientes no se comunican directamente entre sí, sino que envían y reciben mensajes a través del broker.
- **Eficiencia y ligereza**: MQTT está diseñado para ser un protocolo liviano y eficiente, lo que lo hace adecuado para dispositivos con recursos limitados, como sensores y dispositivos IoT.
- **Calidad de Servicio (QoS)**: MQTT ofrece tres niveles de Calidad de Servicio (QoS) para garantizar la entrega de mensajes: QoS 0 (entrega máxima una vez), QoS 1 (entrega al menos una vez) y QoS 2 (entrega exactamente una vez).
- **Seguridad**: MQTT puede implementar mecanismos de autenticación y cifrado para proteger la comunicación y evitar accesos no autorizados.

Para entender mas fácilmente este protocolo imaginemos un carro, este tiene varios sensores que miden cosas como presión de llantas, temperatura del motor, nivel de gasolina, etc. Supongamos que un dispositivo dentro del carro recolecta estos datos de los sensores y los envía a una computadora en la nube utilizando internet. Tú puedes ver esta información desde una aplicación en tu celular o computadora, y además controlar algunas funciones del carro de forma remota, como

bloquear/desbloquear puertas o encender el motor.

En nuestro detector de fugas de agua, tenemos dispositivos similares a los sensores del carro, que son los nodos sensores (NodeMCU + ESP8266). Estos nodos están distribuidos en diferentes zonas y tienen un sensor de agua conectado para detectar fugas.

Cuando un nodo sensor detecta una fuga de agua, envía esa información (como un mensaje) a un broker MQTT, que vendría siendo como la "gran computadora en la nube" del ejemplo del carro.

Nuestra aplicación concentradora (Java) está "suscrita" al broker MQTT, por lo que recibe todos los mensajes de los nodos sensores que indican que se detectó una fuga.

La aplicación concentradora procesa esta información y determina si es necesario activar alguna alarma. En ese caso, envía un mensaje de "activar alarma" al broker MQTT.

Los nodos sensores correspondientes a la zona donde se detectó la fuga también están "suscritos" al tema de "activar alarma" del broker, por lo que reciben ese mensaje y accionan un actuador (buzzer) para alertar sobre la fuga detectada.

Es como si los nodos sensores fueran tus "carros" esparcidos por toda la casa, el broker MQTT fuera la "gran computadora en la nube", y la aplicación concentradora fuera tu aplicación en el celular que "platica" con los nodos a través del broker, recibiendo información de fugas y enviando comandos de activar alarmas.

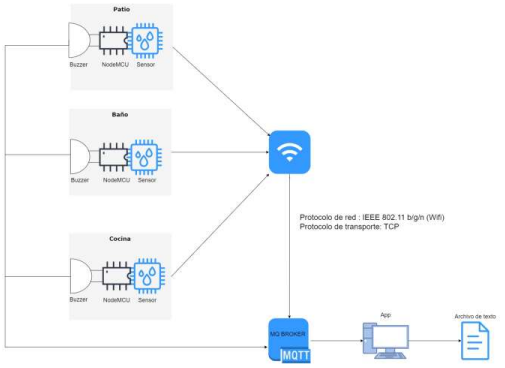
El protocolo MQTT permite esta comunicación eficiente y descentralizada entre los nodos sensores, el broker central y la aplicación concentradora, de manera similar a como funciona en el ejemplo del carro IoT.

Componentes de MQTT en nuestro proyecto

En concreto los componentes principales que utilizan MQTT son:

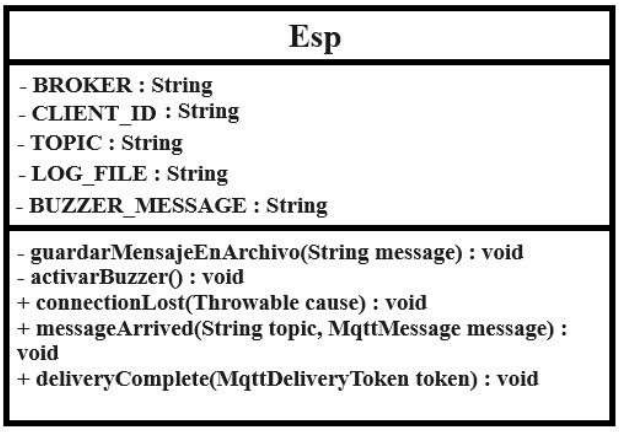
- **Nodos sensores (NodeMCU + ESP8266)**: Estos dispositivos actuarán como clientes MQTT. Publicarán datos de detección de fugas en temas específicos del broker MQTT.
- **Broker MQTT (Mosquitto)**: El broker MQTT actuará como intermediario central de mensajes. Recibirá los datos publicados por los nodos sensores y los reenviará a los clientes suscritos a esos temas.
- **Aplicación concentradora (Java)**: Esta aplicación se suscribirá a los temas relevantes en el broker MQTT para recibir los datos de detección de fugas. También publicará mensajes de activación de alarmas en temas específico

E. TOPOLOGÍA DETALLADA DE LA RED

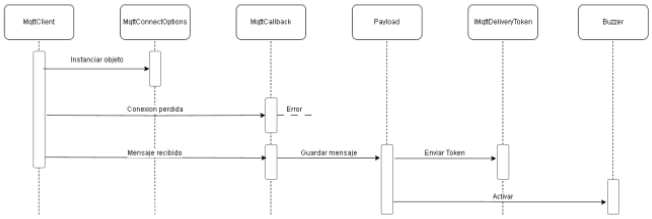


Ver Anexo 2

F. DIAGRAMA DE CLASES



G. DIAGRAMA DE SECUENCIA



VER ANEXO 3

H. PROTOTIPO

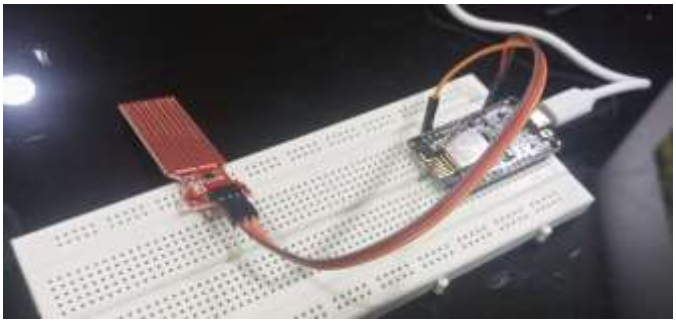


Imagen 1. Sensor de agua conectado a NodeMCU

Imagen 2. Buzzer (actuador) conectado a NodeMCU

Video demostrativo:
https://youtu.be/qrwAir7Wo4A?feature=shared

I. PROTOCOLOS DE PRUEBA

- Pruebas de conectividad: Se realizaron con el fin de confirmar la conexión entre todos los elementos presentes en la topología, así como su conexión a internet.
- Resultados:

Imagen 3 y 4. Prueba de funcionamiento de Mosquitto

- Pruebas del receptor: Se realizaron con el fin de verificar el buen funcionamiento del receptor (detector de agua) y que se estuviese activando cuando entraba en contacto con el agua
- Resultados:

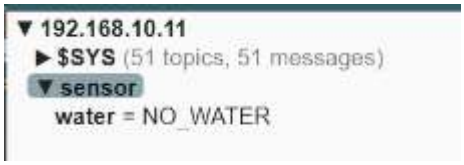


Imagen 5. Sensor conectado a Mosquitto

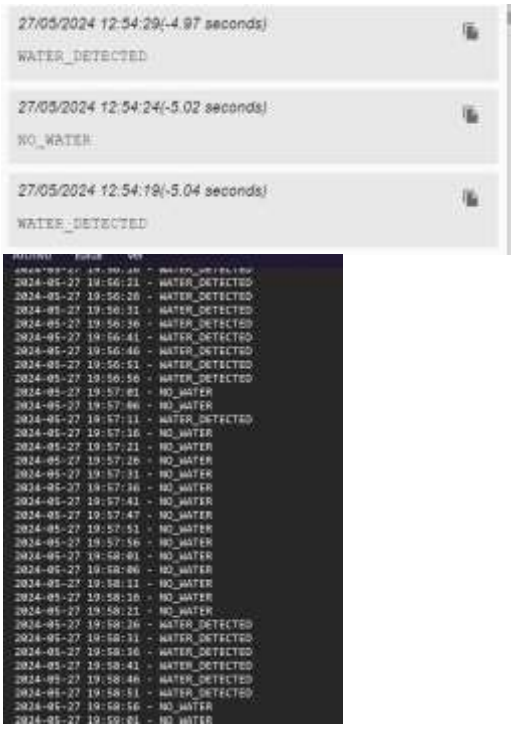


Imagen 6 y 7. Correcta detección de agua por parte del sensor

- Prueba del actuador: Comprobar el correcto funcionamiento del actuador(buzzer) respecto a los datos obtenidos y a los parámetros establecidos para la activación de este.
- Resultado:

https://youtube.com/shorts/FPgd4-GyegE?si=AIDEKNZZ2H3hxB_b

REFERENCIAS

¿Qué es el MQTT? - Explicación del protocolo MQTT - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/mqtt/

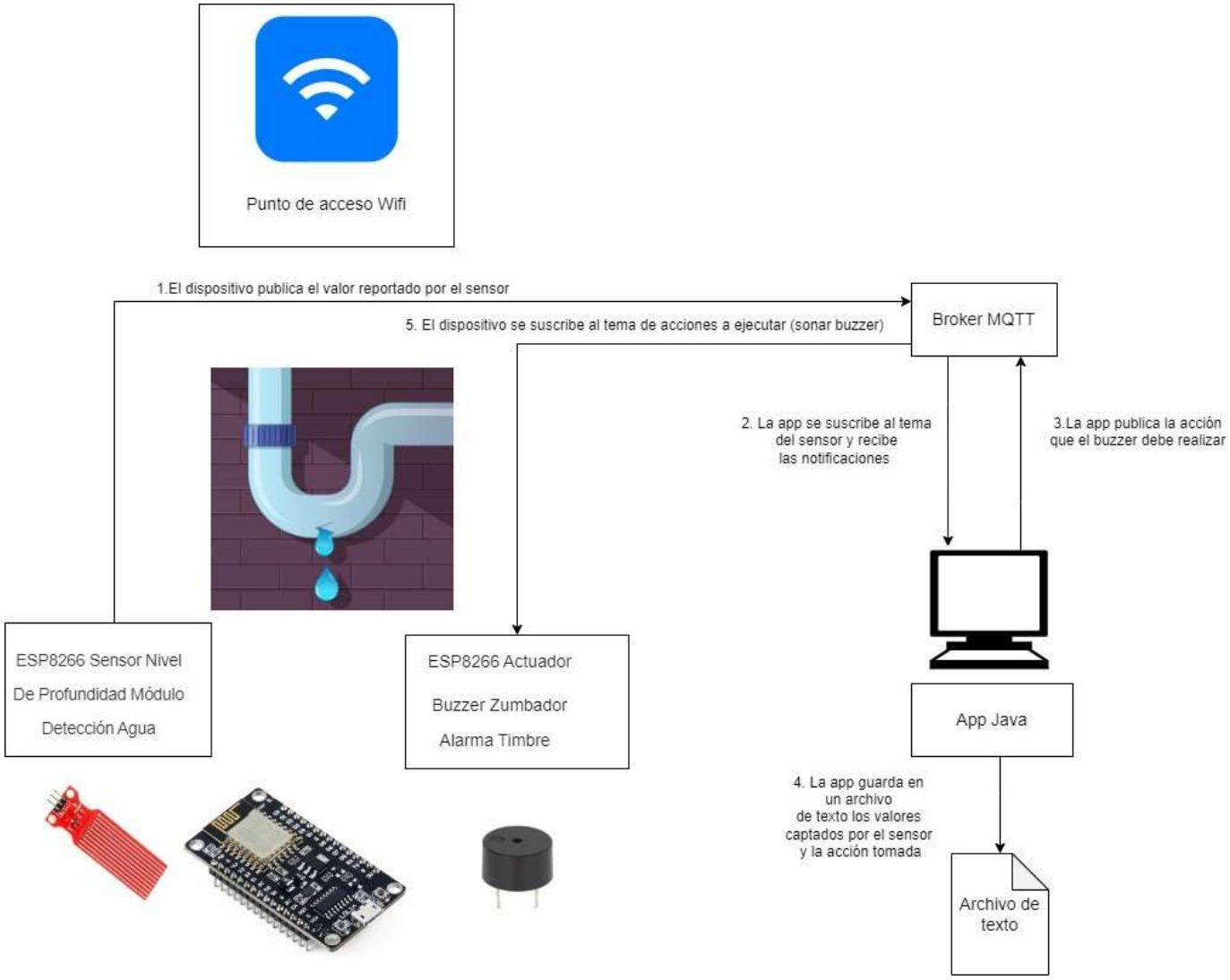
IBM Maximo Monitor Version 8.8.0 and later, and SaaS. (s. f.). https://www.ibm.com/docs/es/mas-cd/maximo-monitor/continuous-delivery?topic=messaging-standards-requirements

MQTT - The Standard for IoT Messaging. (s. f.). https://mqtt.org/

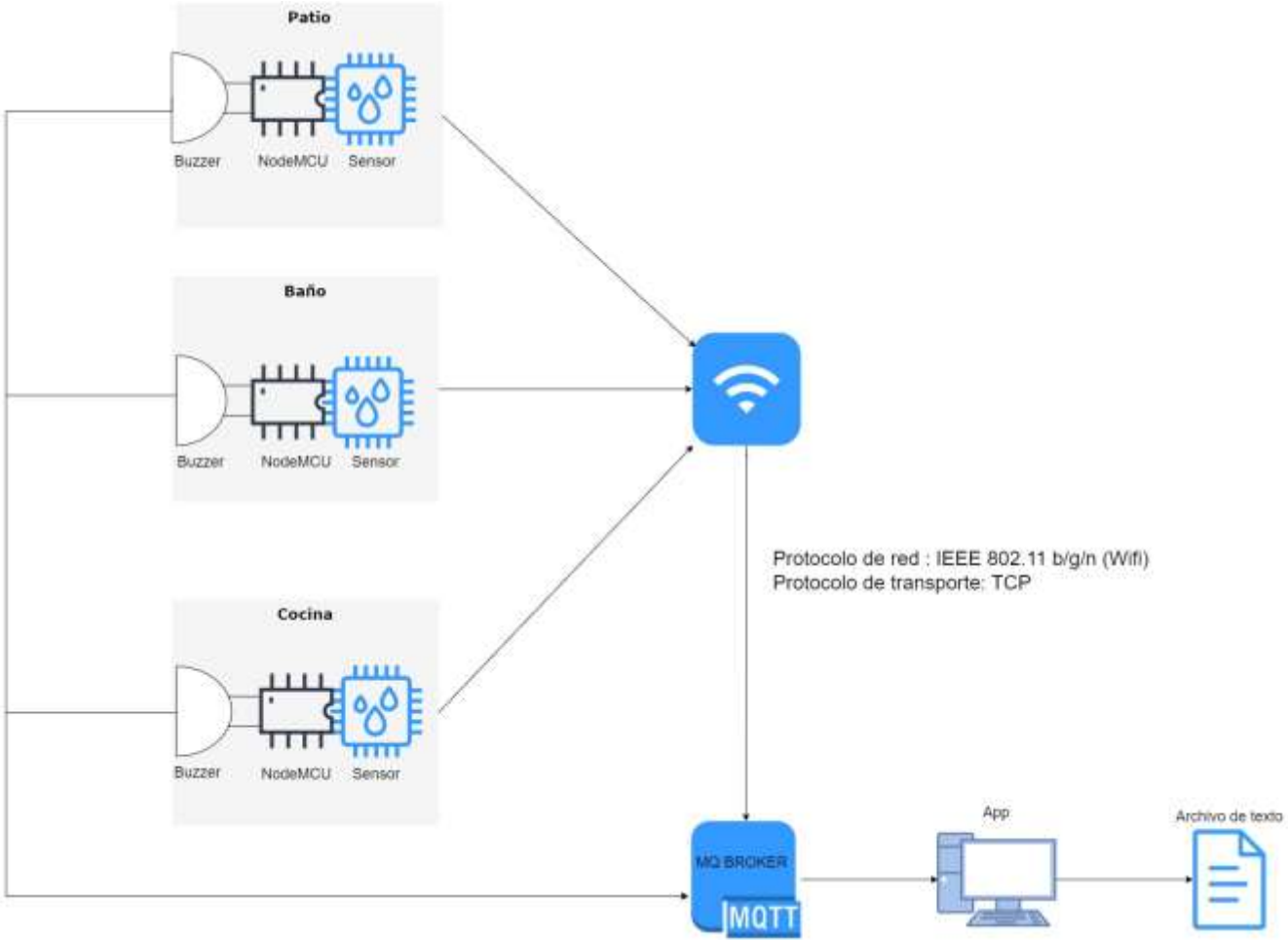
Departamento Nacional de Planeación. (2015). Proyecto de Ley Plan Nacional de Desarrollo 2014-2018: Todos por un nuevo pais.

Espinosa, J. C. G., & Benavides-Muñoz, H. M. (2019). Valor de ajuste del índice de fugas de agua en infraestructuras. Dyna, 86(208), 316-320.

ANEXO 1


Punto de acceso Wifi
1.El dispositivo publica el valor reportado por el sensor
5. El dispositivo se suscribe al tema de acciones a ejecutar (sonar buzzer)
Broker MQTT
2. La app se suscribe al tema del sensor y recibe las notificaciones
3.La app publica la acción que el buzzer debe realizar
ESP8266 Sensor Nivel De Profundidad Módulo Detección Agua
ESP8266 Actuador Buzzer Zumbador Alarma Timbre
App Java
4. La app guarda en un archivo de texto los valores captados por el sensor y la acción tomada
Archivo de texto

ANEXO 2


Patio
Buzzer
NodeMCU
Sensor
Baño
Buzzer
NodeMCU
Sensor
Cocina
Buzzer
NodeMCU
Sensor
Protocolo de red : IEEE 802.11 b/g/n (Wifi)
Protocolo de transporte: TCP
MQ BROKER
MQTT
App
Archivo de texto

ANEXO 3

Diagrama de clase

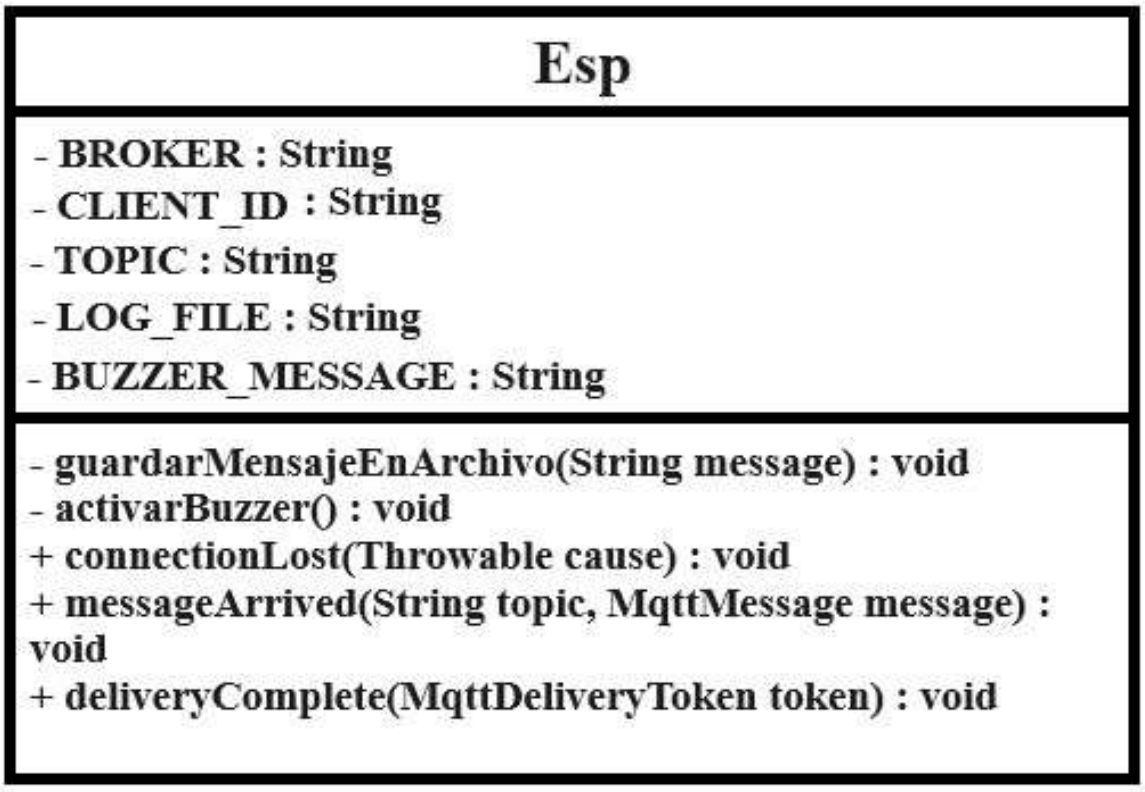


Diagrama de secuencia