UNIVERSIDAD IBEROAMERICANA
LEÓN

INTERNET DE LAS COSAS

Reporte Internet de las Cosas

2do Parcial

Diego Mares Rodríguez
Luis Pablo López Iracheta
María José Gamba Santibáñez

03/11/2024

ÍNDICE:

INTRODUCCIÓN: **3**
MARCO TEÓRICO: **3**
PROTOCOLO MQTT 3
MQTT 3
Modelo Publicador/Suscriptor 3
Broker 4
HiveMQ Cloud: 4
NODE-RED 5
Partes del proyecto: **6**
Código: 6
Secciones de Screenshot: 9
HiveMQ: 9
NodeRed: 9
CONCLUSIONES: **11**
Diego Mares Rodriguez: 11
María José Gamba: 12
Luis Pablo López Iracheta: 12
Bibliografía Consultada (APA): **12**

INTRODUCCIÓN:

En este segundo parcial, nosotros realizamos la conexión de varios sensores (lluvia, ruido, aire, humedad, etc.) en nuestro caso en el ESP32, también sin olvidar la conexión que nosotros realizamos con una cámara que estuviera relacionada con el modelo de ESP32. Esto lo realizamos en esfuerzo para irnos acostumbrando al uso del ESP32, su camará y cómo esto va a ayudarnos a prepararnos para el proyecto final, consistiendo en una cámara que permita agilizar el uso del estacionamiento de la universidad, tomando en cuenta también los factores externos que se detecta con el uso de los sensores ya antes mencionados.

El proyecto en sí detecta la calidad del aire, la humedad que hay cuando llueve, la temperatura presente en el ambiente y el nivel de humedad. La cámara cumplirá la función principal de monitoreo a través del ESP32 cam.

MARCO TEÓRICO:

MQTT (Message Queuing Telemetry Transport) y Node-RED son dos herramientas fundamentales en este campo, ya que permiten a los desarrolladores integrar, procesar y visualizar datos de forma efectiva en entornos IoT. En este caso, utilizaremos las herramientas de Hivemq y Node-red.

PROTOCOLO MQTT

MQTT

MQTT es un protocolo de mensajería liviano, diseñado específicamente para dispositivos con capacidades limitadas y para redes de baja calidad o con ancho de banda restringido. Su popularidad ha crecido en el mundo de IoT gracias a su eficiencia en la transmisión de datos, facilitando la comunicación en tiempo real entre dispositivos y aplicaciones. MQTT sigue un modelo publicador/suscriptor y generalmente opera en la nube.

Modelo Publicador/Suscriptor

A diferencia de un modelo de comunicación directa, MQTT utiliza un enfoque de publicador/suscriptor. Esto significa que los dispositivos no se comunican directamente, sino que publican o se suscriben a "temas" específicos. Cada tema representa una categoría o canal de datos. Los dispositivos que publican envían sus datos a un tema, mientras que los suscriptores reciben los mensajes de los temas a los que están suscritos. Este modelo permite una comunicación eficiente y flexible.

Publicador: Es el dispositivo que envía mensajes a un tema determinado. Por ejemplo, un sensor de temperatura podría enviar datos al tema “casa/temperatura”.

Suscriptor: Es el dispositivo o aplicación que se suscribe a un tema para recibir actualizaciones. Por ejemplo, una aplicación de monitoreo se suscribirá al tema “casa/temperatura” para recibir los datos del sensor.

Broker

El broker es un componente clave en el modelo MQTT. Es el servidor que actúa como intermediario entre publicadores y suscriptores, gestionando y enviando los mensajes a los suscriptores adecuados. En este proyecto, se utilizó el broker HiveMQ Cloud.

HiveMQ Cloud:

HiveMQ Cloud es una plataforma basada en la nube que opera mediante el protocolo MQTT y funciona como un broker para el intercambio de mensajes en proyectos de Internet de las Cosas (IoT). Su objetivo principal es facilitar la creación de prototipos y la implementación de soluciones IoT escalables, proporcionando a los usuarios una interfaz sencilla y un entorno seguro para publicar y suscribirse a mensajes.

Temas

Los temas son los canales de comunicación en MQTT. Funcionan como una estructura jerárquica de texto (por ejemplo, “casa/temperatura/sala”), permitiendo organizar los datos por categorías o ubicaciones específicas. Esto ayuda a filtrar los mensajes y recibir sólo la información relevante en cada suscripción.

Calidad de Servicio (QoS)

MQTT permite especificar un nivel de calidad de servicio (QoS) para garantizar la entrega de los mensajes, dependiendo de la confiabilidad requerida:

- **QoS 0 (al menos una vez):** No garantiza la entrega; el mensaje se envía solo una vez, sin confirmar si el receptor lo ha recibido.
- **QoS 1 (al menos una vez):** Asegura que el mensaje llegará al menos una vez, *aunque puede entregarse varias veces.*
- **QoS 2 (exactamente una vez):** Garantiza que el mensaje se entregará una vez y solo una vez, lo cual requiere más confirmaciones.

NODE-RED

Para monitorear y visualizar los datos enviados por MQTT, se usó Node-RED. Esta herramienta permite conectar y visualizar datos en tiempo real mediante una interfaz gráfica de programación. Node-RED es ideal para quienes desean integrar y analizar datos IoT sin escribir mucho código.

Entre sus principales ventajas está:

Interfaz gráfica intuitiva
Node-RED permite construir flujos de datos mediante un sistema de arrastrar y soltar nodos, lo que facilita la creación de aplicaciones sin necesidad de conocimientos profundos en programación. Esto reduce el tiempo de configuración y permite crear prototipos rápidamente.

Nodos preconfigurados para MQTT
Node-RED ya incluye nodos para el protocolo MQTT, lo que simplifica el proceso de conexión con HiveMQ. Basta con ingresar las credenciales del broker (en este caso, HiveMQ Cloud) para que los datos comiencen a fluir en el sistema. Esto permite configurar y supervisar datos sin una programación compleja, lo que lo convierte en una herramienta accesible y poderosa.

Dashboards integrados
Una característica destacada de Node-RED es su complemento de dashboard. Este permite crear interfaces visuales para ver datos en gráficos, medidores, y otros elementos visuales sin salir de la plataforma. En el contexto de este proyecto, el dashboard permite que Node-RED se suscriba a temas de HiveMQ Cloud y visualice los datos de los dispositivos IoT en tiempo real.

MQTT y Node-RED son componentes esenciales para la comunicación en IoT. Mientras MQTT facilita el intercambio de datos entre dispositivos mediante un modelo publicador/suscriptor y con niveles de calidad de servicio ajustables, Node-RED permite visualizar estos datos de forma accesible y efectiva. Esta combinación de herramientas permite crear y gestionar soluciones IoT de manera escalable, optimizando la conectividad y la visualización de datos en tiempo real.

Partes del proyecto:

Código:

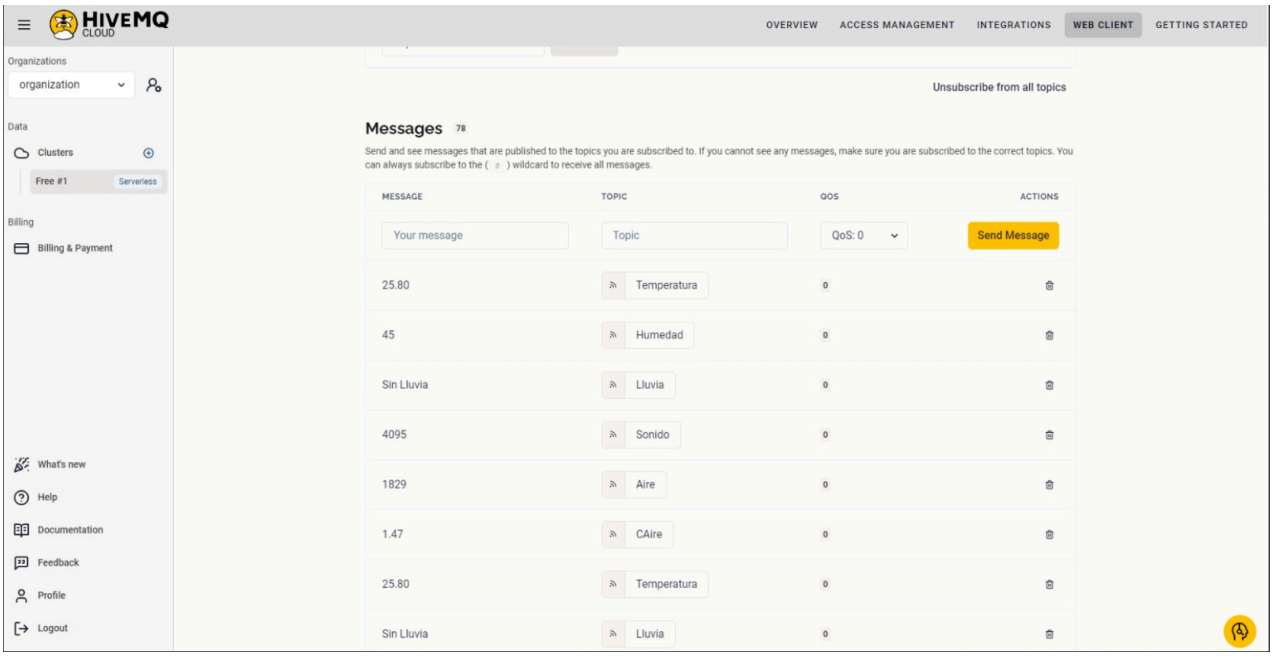
```
// Autores: Luis Pablo López Iracheta,  María José Gamba Santibáñez, Diego Mares
// Incluimos las librerias necesarias
#include <WiFi.h>
#include <PubSubClient.h>
#include <WiFiClientSecure.h>
#include <DHT.h>

//---- WiFi settings
const char* ssid = "INFINITUM38A4_2.4";
const char* password = "sY4mX2MZkG";
//---- MQTT Broker settings
const  char*  mqtt_server  =  "4d58dc4d4da64daca6f0003289196edb.s1.eu.hivemq.cloud";
// Cluster url
const  char*  mqtt_username  =  "MAMALU";    //  Broker  usuario  que  usamos  para
HiveMQCloud
const  char*  mqtt_password  =  "Cont54%jl03#";          //  Broker  contraseña  del
HiveMQCloud
const int mqtt_port = 8883;
const char* mqtt_topic = "led/control";

//----------------------------SENSORES PUERTOS ------------------------------
// Definir el pin y modelo del sensor DHT
const byte DHTPIN = 13;
const byte LLUVIA_PIN = 34; // Sensor de lluvia
const byte SONIDO_PIN = 35; // Sensor de sonido
const byte ledPin1 = 26;    // LED
const byte AIRE_PIN = 32;    // LED

#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

WiFiClientSecure espClient;
PubSubClient client(espClient);

// Declaramos los tópicos a utilizar con sus nombres con lo que aparecerán
const char* Valor_Temperatura_Topic = "Temperatura";
const char* Valor_Humedad_Topic = "Humedad";
const char* Valor_Lluvia_Topic = "Lluvia";
const char* Valor_Sonido_Topic = "Sonido";
const char* Valor_AIRE_Topic = "Aire";
const char* Valor_CAIRE_Topic = "CAire";

static const char *root_ca PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw
TzELMAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmV0IFNlY3VyaXR5IFJlc2Vh
cmNoIEdyb3VwMRUwEwYDVQQDEwxJU1JHIFJvb3QgWDEwHhcNMTUwNjA0MTEwNDM4
WhcNMzUwNjA0MTEwNDM4WjBPMQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJu
ZXQgU2VjdXJpdHkgUmVzZWFyY2ggR3JvdXAxFTATBgNVBAMTDElTUkcgUm9vdCBY
MTCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAK3oJHP0FDfzm54rVygc
h77ct984kIxuPOZXoHj3dcKi/vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+
0TM8ukj13Xnfs7j/EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U
A5/TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW
T8KOEUt+zwvo/7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/ELNKjD+Jo2FR3qyH
B5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC
B5iPNgiV5+I3lg02dZ77DnKxHZu8A/lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv
KBds0pjBqAlkd25HN7rOrFleaJ1/ctaJxQZBKT5ZPt0m9STJEadao0xAH0ahmbWn
OlFuhjuefXKnEgV4We0+UXgVCwOPjdAvBbI+e0ocS3MFEvzG6uBQE3xDk3SzynTn
jh8BCNAw1FtxNrQHusEwMFxIt4I7mKZ9YIqioymCzLq9gwQbooMDQaHWBfEbwrbw
```

```
qHyGO0aoSCqI3Haadr8faqU9GY/rOPNk3sgrDQoo//fb4hVC1CLQJ13hef4Y53CI
rU7m2Ys6xt0nUW7/vGT1M0NPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNV
HRMBAf8EBTADAQH/MB0GA1UdDgQWBBR5tFnme7bl5AFzgAiIyBpY9umbbjANBgkq
hkiG9w0BAQsFAAOCAgEAVR9YqbyyqFDQDLHYGmkgJykIrGF1XIpu+ILlaS/V9lZL
ubhzEFnTIZd+50xx+7LSYK05qAvqFyFWhfFQDlnrzuBZ6brJFe+GnY+EgPbk6ZGQ
3BebYhtF8GaV0nxvwuo77x/Py9auJ/GpsMiu/X1+mvoiBOv/2X/qkSsisRcOj/KK
NFtY2PwByVS5uCbMiogziUwthDyC3+6WVwW6LLv3xLfHTjuCvjHIInNzktHCgKQ5
ORAzI4JMPJ+GslWYHb4phowim57iaztXOoJwTdwJx4nLCgdNbOhdjsnvzqvHu7Ur
TkXWStAmzOVyyghqpZXjFaH3pO3JLF+l+/+sKAIuvtd7u+Nxe5AW0wdeRlN8NwdC
jNPElpzVmbUq4JUagEiuTDkHzsxHpFKVK7q4+63SM1N95R1NbdWhscdCb+ZAJzVc
oyi3B43njTOQ5yOf+1CceWxG1bQVs5ZufpsMljq4Ui0/1lvh+wjChP4kqKOJ2qxq
4RgqsahDYVvTH9w7jXbyLeiNdd8XM2w9U/t7y0Ff/9yi0GE44Za4rF2LN9d11TPA
mRGunUHBcnWEvgJBQl9nJEiU0Zsnvgc/ubhPgXRR4Xq37Z0j4r7g1SgEEzwxA57d
emyPxgcYxn/eR44/KJ4EBs+lVDR3veyJm+kXQ99b21/+jh5Xos1AnX5iItreGCc=
-----END CERTIFICATE-----
)EOF";

// Función de callback MQTT (para recibir mensajes)
void callback(char* topic, byte* payload, unsigned int length) {
  String incommingMessage = "";
  for (int i = 0; i < length; i++) {
    incommingMessage += (char)payload[i];
  }

  Serial.print("Mensaje recibido en tópico [");
  Serial.print(topic);
  Serial.print("]: ");
  Serial.println(incommingMessage);

  // Control del LED en función del mensaje recibido
  if (incommingMessage == "ON") {
    digitalWrite(ledPin1, HIGH);  // Enciende LED en pin 26
    Serial.println("LED encendido");
  } else if (incommingMessage == "OFF") {
    digitalWrite(ledPin1, LOW);   // Apaga LED en pin 26
    Serial.println("LED apagado");
  }
}

// Función de reconexión a MQTT utilizando las credenciales dadas
void reconnect() {
  while (!client.connected()) {
    Serial.print("Intentando conexión MQTT...");
    String clientId = "ESP32Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str(), mqtt_username, mqtt_password)) {
      Serial.println("Conectado al broker MQTT");
      client.subscribe(mqtt_topic);
    } else {
      Serial.print("Error, rc=");
      Serial.print(client.state());
      Serial.println(" Intentando de nuevo en 3 segundos...");
      delay(3000);
    }
  }
}

// Código de inicio
void setup() {
  dht.begin();
  delay(2000);
// Iniciamos el monitor serial
  Serial.begin(9600);
  pinMode(SONIDO_PIN, INPUT);  // Declaramos el PIN del sonido
  pinMode(ledPin1, OUTPUT); // Declaramos el PIN del led
  Serial.print("\nConectando a ");
  Serial.println(ssid);
```

```
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  randomSeed(micros());
  Serial.println("\nWiFi connected\nIP address: ");
  Serial.println(WiFi.localIP());
  espClient.setCACert(root_ca);
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
 // -------- Tomamos el valores de los sensores---
  float temperatura = dht.readTemperature();
  int humedad = dht.readHumidity();
  int valorLluvia = analogRead(LLUVIA_PIN);
  int valorSonido = analogRead(SONIDO_PIN);
  int valorAire = analogRead(AIRE_PIN);
  Serial.println(valorAire);
  float valorCalidadAire = (valorAire / 4095.0) * 3.3;
 // ------------------------------------------

 // Mediante IF, hacemos una escala para la lluvia
  String estadoLluvia;
  if (valorLluvia <= 1023) {
    estadoLluvia = "Aguacero";
  } else if (valorLluvia <= 2047) {
    estadoLluvia = "Lluvia moderada";
  } else if (valorLluvia <= 3071) {
    estadoLluvia = "Lluvia ligera";
  } else {
    estadoLluvia = "Sin Lluvia";
  }
 // Publicamos los valores para verlos en las pantalla

  publishMessage(Valor_Temperatura_Topic, String(temperatura), true);
  publishMessage(Valor_Humedad_Topic, String(humedad), true);
  client.publish(Valor_Lluvia_Topic, estadoLluvia.c_str());
  client.publish(Valor_Sonido_Topic, String(valorSonido).c_str());
  client.publish(Valor_AIRE_Topic, String(valorAire).c_str());
  client.publish(Valor_CAIRE_Topic, String(valorCalidadAire).c_str());
  delay(2000);
}
 // Función para el mensaje de temperatura y humedad
void publishMessage(const char* topic, String payload, boolean retained){
  if (client.publish(topic, payload.c_str(), retained))
    Serial.println("Message published [" + String(topic) + "]: " + payload);
}
```

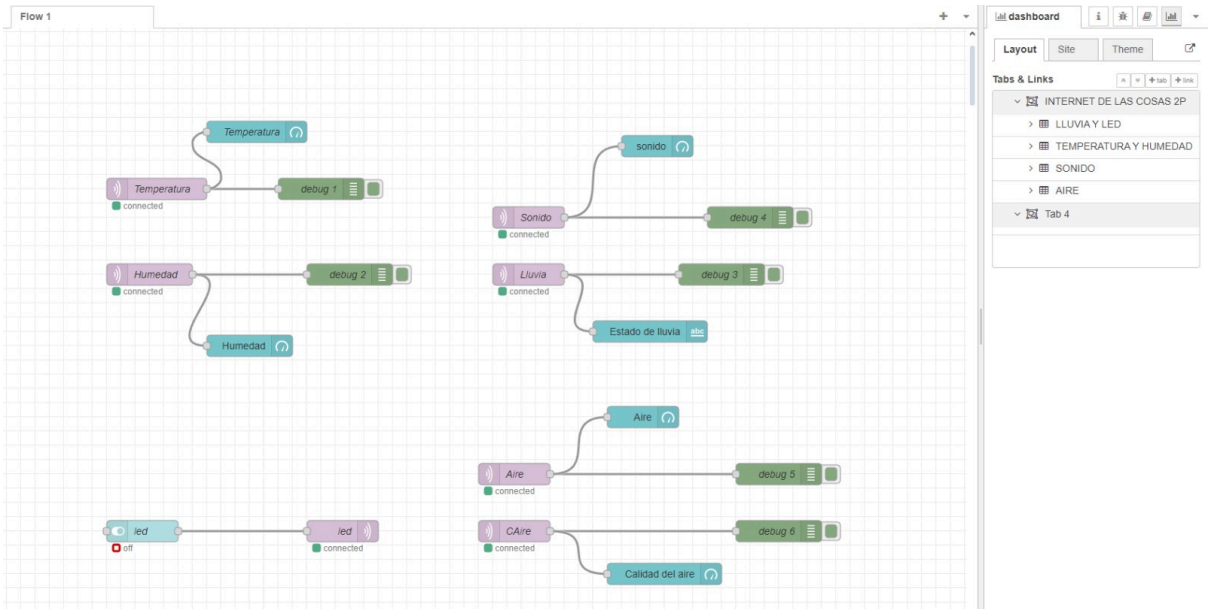
Secciones de Screenshot:

HiveMQ:



En la aplicación de HiveMQ pudimos desarrollar un Cluster donde almacenar información que recibía la ESP32, de manera que podíamos suscribir nuestro usuario a los datos y verlos en pantalla. Esta parte era fundamental para poder completar el proyecto y poder analizar la información de los sensores.

NodeRed:



Esta es la parte del Dashboard en Node Red una vez ya se inicializó de forma correcta en la terminal. Lo único que se está haciendo aquí es más que nada crear la conexión y suscribirse a HiveMQCloud con los permisos que se deben y con el servidor/broker que ya habías creado anteriormente. Lo único que se debe de revisar es, en base al cluster que ya habíamos creado, revisar si ya recibe una señal. Si lo hace como nosotros lo configuramos es que ya se conectó/suscribió de manera correcta. Lo que se ve es cada una de las funciones que nosotros hicimos de los sensores. Si se observa están todos, hasta el del LED que agregamos.

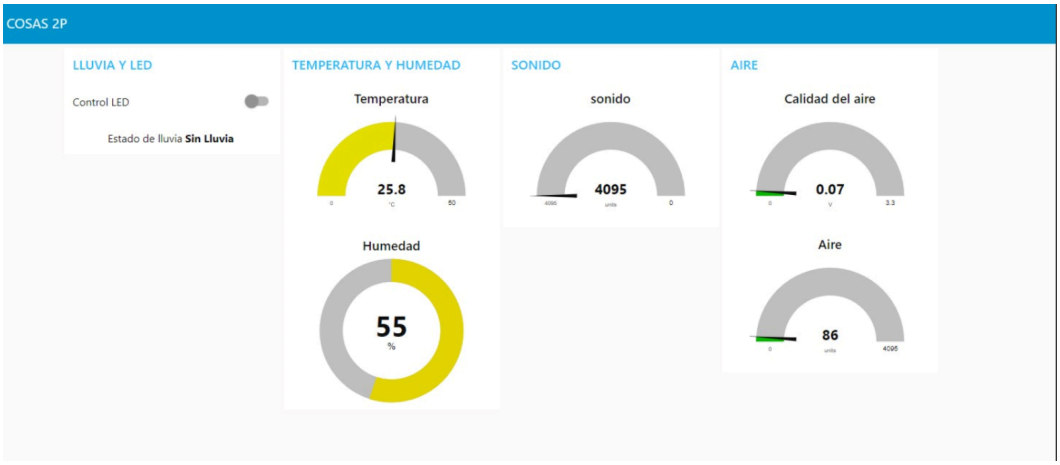
Explicación del Flujo de Node-RED

Temperatura y Humedad:
Hay un nodo llamado "Temperatura" y otro llamado "Humedad". Ambos nodos están conectados y envían datos a sus respectivos indicadores en el dashboard.
El nodo "Temperatura" envía sus datos a un nodo de debug ("debug 1"), lo que permite monitorear el valor que se está transmitiendo.
De manera similar, el nodo "Humedad" está conectado a un nodo de debug ("debug 2") para verificar los valores de humedad.

Sonido y Lluvia:
Hay dos nodos adicionales: "Sonido" y "Lluvia", que están configurados para monitorear las condiciones de sonido y si hay presencia de lluvia.
Ambos nodos están conectados a nodos de debug ("debug 4" y "debug 3") para visualizar los datos de cada sensor.
El nodo "Lluvia" también está conectado a un nodo de texto en el dashboard, llamado "Estado de lluvia", que muestra en el panel el estado de la lluvia en tiempo real.

Calidad del Aire:
Se incluye un sensor de "Aire" que monitorea la calidad del aire.
Este nodo está conectado a dos nodos de debug ("debug 5" y "debug 6") para visualizar los datos de calidad del aire.
El nodo "CAire" representa una descripción de la calidad del aire en un cuadro de texto dentro del dashboard.

Control de LED:
En la parte inferior, hay un nodo llamado "led" que probablemente se utiliza para controlar un LED físico, encendiéndolo o apagándolo.
Este nodo muestra un indicador de conexión, que en este caso está "off", lo que indica que actualmente no está conectado o no está recibiendo señales.



Lo que se logra apreciar en esta screenshot sería la parte del resultado final de las conexiones de los dispositivos con el NodeRed. Aquí se podría decir que es la “interfaz” de lo que detectan los sensores de humedad, aire, temperatura, calidad del aire y sonido. En base al nivel de sensibilidad que pusimos, va mostrando un medidor que indica el nivel de cada uno (alto, medio, bajo) en base al color y el nivel de porcentaje. Además de que se incluye el interruptor para encender y apagar el led.

Esta interfaz nos ayuda a presentar de mejor manera los datos obtenidos, además de que nos permite visualizarlos de manera remota e interactuar con los componentes.

CONCLUSIONES:

Diego Mares Rodriguez:

Mis conclusiones personales con respecto al proyecto se refiere es que me ha sido de gran utilidad en el sentido del uso del ESP32 con las conexiones que se pueden utilizar en la nube con otras “plataformas” y como estás se pueden manipular a través de un dashboard virtual que estén conectados a los dispositivos. Se me hizo curioso e interesante como a través de las conexiones a nuestro propio servidor pudimos crear una interfaz que nos permitiera ver en tiempo real cómo es que los sensores (en base al nivel de sensibilidad que pusimos) muestran los resultados que detectaba en ese momento.

Creo que con las bases que llevamos hasta ahora podemos empezar a trabajar con más confianza en cuanto al proyecto se refiere del estacionamiento y que, tal vez en un futuro, podamos seguir teniendo ideas similares para proyectos futuros en base a servidores en la nube y conexiones de los dispositivos/sensores también el la nube.

María José Gamba:

La realización de este proyecto ha sido una experiencia muy enriquecedora que me permitió adentrarme en el mundo del Internet de las Cosas (IoT). Trabajar con el protocolo MQTT y Node-RED me hizo ver cómo se puede establecer una comunicación eficiente entre dispositivos, algo que es clave para desarrollar soluciones tecnológicas hoy en día.

Aprender a usar MQTT me mostró por qué es tan importante tener un protocolo ligero y adaptable, especialmente para ambientes con poca conectividad, algo que sigue siendo un reto en muchos lugares. Por otro lado, ver los datos en tiempo real a través de Node-RED fue súper útil, ya que no solo hizo más fácil entender la información, sino que también me hizo valorar la importancia de tener interfaces intuitivas y fáciles de usar.

A lo largo del proyecto, también reflexioné sobre el impacto que estas tecnologías pueden tener en la automatización y el monitoreo en tiempo real. Me dejó con muchas ganas de seguir explorando estas herramientas y su aplicación en futuros proyectos, ya que creo que tienen el potencial de mejorar procesos y facilitar la vida en muchas áreas.

Esta experiencia no solo me ayudó a mejorar mis habilidades técnicas, sino que también despertó más mi curiosidad por las nuevas tendencias tecnológicas. Sin duda, me ha preparado para poder contribuir al avance de la innovación en México y otros lugares en el futuro.

Luis Pablo López Iracheta:

Este proyecto fue un gran paso en mi aprendizaje sobre IoT. Al integrar varios sensores con la ESP32 y conectarlos a Node-RED, entendí mejor cómo los datos del ambiente pueden ayudarnos a tomar decisiones en tiempo real. Fue interesante ver cómo cada componente juega su papel y cómo se pueden coordinar para lograr un sistema funcional.

Este trabajo me enseñó a ser más paciente y detallista, debido a que cada ajuste o configuración cuenta para obtener buenos resultados. Además, me abrió los ojos a todo lo que se puede hacer con herramientas simples y me motivó a seguir explorando proyectos de automatización y control. En general, fue una experiencia enriquecedora que me acerca a mis metas en tecnología.

Bibliografía Consultada (APA):

- Node-RED. (2024). *Nodered: Low-code programming for event-driven applications.* https://nodered.org/

- HiveMQ Cloud. (2024) *HiveMQ Documentation*. https://docs.hivemq.com/hivemq-cloud/index.html

- AWS. (2022) *¿Qué es MQTT?* https://aws.amazon.com/es/what-is/mqtt/

- Pascual, C. (2022). *ESP32 CAM introducción y primeros pasos.* Programarfacil Arduino y Home Assistant. https://programarfacil.com/esp32/esp32-cam/

- Hunkeler, U., & Truong, H. L. (2010). "M2M Communication: The Future of Smart Grid." In *Smart Grid and Internet of Things* (pp. 17-35). Springer.Este libro aborda cómo la comunicación máquina a máquina (M2M) puede integrarse con tecnologías emergentes como IoT, proporcionando una base teórica sólida sobre MQTT.