## **1. Resumen del Proyecto**

### **1.1. Descripción General**

El proyecto consiste en desarrollar un dispositivo IoT que mide distancias utilizando un sensor de ultrasonido. La información capturada se publica en AWS IoT mediante los tópicos "proximity" y "controlled" para establecer una comunicación bidireccional entre el dispositivo y una aplicación móvil desarrollada en Flutter.

**Objetivo del Proyecto**:

* La distancia medida por el sensor de ultrasonido, se publica en aws mediante el tema proximity donde es enviado a su vez a la aplicación móvil.
* La aplicación móvil enciende el led según el dato recibido y las reglas establecidas.
* La aplicación también envía un estado de vuelta al dispositivo mediante el tema controlled para controlar el Led que se debe encender.
* La información se debe enviar con un estándar de formato de datos ECMA 404, Json.

**Reglas:**

Crear una aplicación móvil en Flutter

1. Se debe suscribir al tema de proximidad
2. Debe publicar al tema controlled
3. Cuando la distancia esté entre 3 y 10 cm se enciende el led rojo de pantalla y publica 1
4. Cuando la distancia esté entre 11 y 15 cm se enciende el led amarillo de pantalla y publica 2
5. Cuando la distancia esté entre >15 cm se enciende el led verde de pantalla y publica 3

Complementar el cliente MQTT

1. Se debe suscribir al tema controlled
2. Cuando recibe 1 prende el led rojo
3. Cuando recibe 2 prende el led amarillo
4. Cuando recibe 3 prende el led verde

### **Metodología**

Se siguió el marco de trabajo Scrum, adoptando un modelo de desarrollo evolutivo, en el cual se adopta un enfoque iterativo e incremental, donde se genera valor en cada iteración realizada.

## **2. Iteraciones del Proyecto**

### **Iteración 1: Montaje y programación del dispositivo iot**

* **Objetivo**:
  + Montar dispositivo IoT ESP8266.
  + Configuración objeto en AWS que permita publicar y suscribir
* **Actividades**:
  + Montaje del ESP8266 con el sensor de ultrasonido y los LEDs.
  + Pruebas de conexión y funcionamiento del hardware.
  + Programación en Arduino para la suscripción y publicación del tema de proximidad en AWS IoT.
  + Pruebas unitarias en donde se muestra la transmisión de datos.

### **Iteración 2: Desarrollo Aplicación Web**

* **Objetivo**: Desarrollar una aplicación web que reciba datos del tema de proximidad en aws.
* **Actividades**:
  + Programación de la aplicación móvil en flutter
  + Pruebas de comunicación entre flutter y AWS IoT.
  + integración entre el dispositivo IoT, AWS y flutter.
  + Pruebas Unitarias tema proximidad en Flutter, dispositivo y AWS.

### **Iteración 3: Desarrollo tema Controlled Aws Flutter**

* **Objetivo**:
  + Configurar en aws el tema de controlled
  + Desarrollo tema controlled en flutter
* **Actividades**:
  + Realizar la configuración en Aws del tema Controlled
  + Desarrollo de la aplicación Flutter para recibir datos del sensor y controlar los LEDs.
  + Pruebas Unitarias.

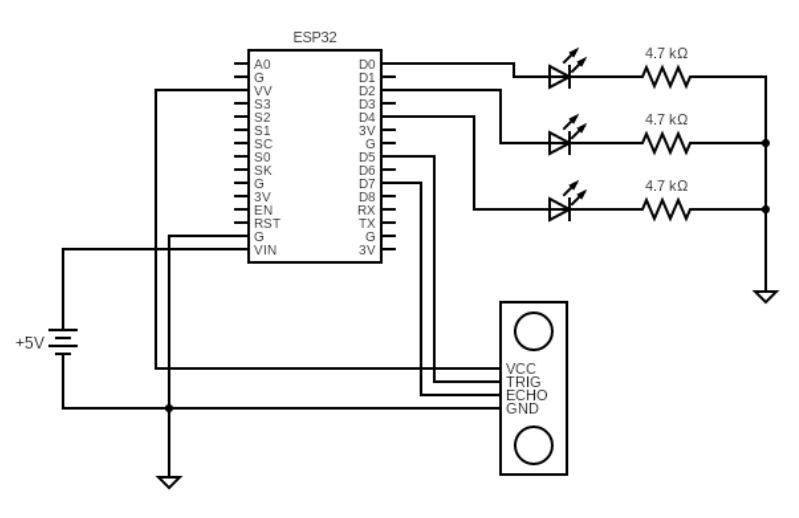
### **Iteración 4: Desarrollo tema Controlled Arduino**

* **Objetivo**: Desarrollo en arduino para integrar el tema de controlled
* **Actividades**:
  + Desarrollo de la aplicación Flutter para recibir datos del sensor y controlar los LEDs.
  + Integración y pruebas finales del sistema completo.

## **3. Detalles Técnicos**

### **3.1. Hardware del Dispositivo IoT**

* **Componentes**:
  + ESP8266
  + Sensor de ultrasonido HC-SR04
  + LEDs (3 unidades)
  + Resistencias
  + Fuente
  + Protoboard
  + Cables
* **Esquema de Conexión**:
  + ESP8266 a protoboard.
  + HC-SR04 conectado a pines digitales del ESP8266.
  + LEDs conectados a pines digitales del ESP8266 con resistencias en serie.



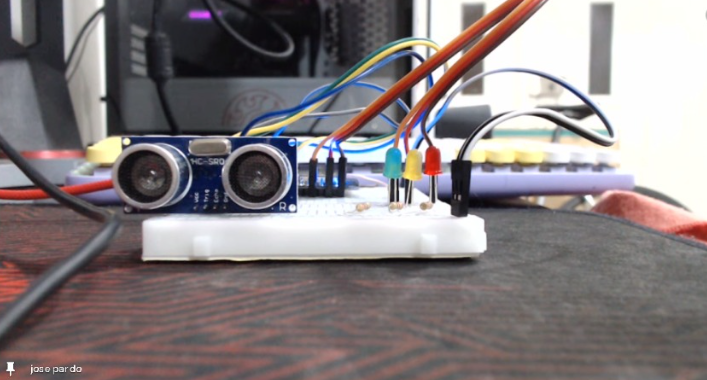


Imagen que contiene interior, tabla, computadora, escritorio

Descripción generada automáticamente

### **3.2. Código Arduino para ESP8266**

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <time.h>

#include <BearSSLHelpers.h>

#include "ca\_cert.h"

#include "client\_cert.h"

#include "private\_key.h"

// WiFi settings

const char\* ssid = "STARGO-VIRUS";

const char\* password = "M4g0d30z35";

// AWS IoT settings

const char\* aws\_endpoint = "a1mgq13vqqoshs-ats.iot.us-east-1.amazonaws.com";

const int port = 8883;

// NTP settings

const char\* ntpServer = "pool.ntp.org";

const long gmtOffset\_sec = 0;

const int daylightOffset\_sec = 0;

// Flag to track connection status

bool awsConnected = false;

// Create a secure client

BearSSL::WiFiClientSecure net;

PubSubClient client(net);

//Sensor

const int trigPin = 14; // TRIG connected to D5

const int echoPin = 13; // ECHO connected to D7

const int ledPin1 = 16; // LED connected to D0 LED ROJO

const int ledPin2 = 4; // LED connected to D2 LED AMARILLO

const int ledPin3 = 2; // LED connected to D4 LED VERDE

// Timer to track last message received time

unsigned long lastMsgTime = 0;

int proximity() {

// Clear the trigPin by setting it LOW

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Trigger the sensor by setting the trigPin HIGH for 10 microseconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Read the time for the echoPin to go HIGH

long duration = pulseIn(echoPin, HIGH);

// Calculate the distance

// Speed of sound wave divided by 2 (there and back)

int distance = duration \* 0.034 / 2;

// Print the distance on the serial monitor

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

return distance;

}

void setup\_wifi() {

delay(10);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

void callback(char\* topic, byte\* payload, unsigned int length) {

Serial.print("Message arrived [");

Serial.print(topic);

Serial.print("] ");

String message;

for (unsigned int i = 0; i < length; i++) {

Serial.print((char)payload[i]);

message += (char)payload[i];

}

Serial.println();

// Update the last message received time

lastMsgTime = millis();

// Process the message

if (String(topic) == "myhome/door/controlled") {

if (message == "1") {

digitalWrite(ledPin1, HIGH); // Turn on red LED

digitalWrite(ledPin2, LOW);

digitalWrite(ledPin3, LOW);

} else if (message == "2") {

digitalWrite(ledPin1, LOW);

digitalWrite(ledPin2, HIGH); // Turn on yellow LED

digitalWrite(ledPin3, LOW);

} else if (message == "3") {

digitalWrite(ledPin1, LOW);

digitalWrite(ledPin2, LOW);

digitalWrite(ledPin3, HIGH); // Turn on green LED

} else {

digitalWrite(ledPin1, LOW);

digitalWrite(ledPin2, LOW);

digitalWrite(ledPin3, LOW); // Turn off all LEDs if message is not 1, 2, or 3

}

}

}

void setup() {

Serial.begin(115200);

// Set the trigPin as an OUTPUT

pinMode(trigPin, OUTPUT);

// Set the echoPin as an INPUT

pinMode(echoPin, INPUT);

// Set the LED pins as OUTPUT

pinMode(ledPin1, OUTPUT);

pinMode(ledPin2, OUTPUT);

pinMode(ledPin3, OUTPUT);

setup\_wifi();

// Initialize NTP

configTime(gmtOffset\_sec, daylightOffset\_sec, ntpServer);

// Wait for time to be set

time\_t now = time(nullptr);

while (now < 8 \* 3600 \* 2) {

delay(500);

Serial.print(".");

now = time(nullptr);

}

Serial.println("");

Serial.println("Time synchronized");

// Load certificate and private key

BearSSL::X509List trustAnchor(ca\_cert);

BearSSL::X509List clientCert(client\_cert);

BearSSL::PrivateKey clientKey(private\_key);

net.setTrustAnchors(&trustAnchor);

net.setClientRSACert(&clientCert, &clientKey);

client.setServer(aws\_endpoint, port);

client.setCallback(callback);

client.setKeepAlive(60); // Increase keepalive interval to 60 seconds

// Check and connect to AWS IoT

if (client.connect("ESP8266Client")) {

if (!awsConnected) {

Serial.println("Connected to AWS IoT");

awsConnected = true; // Set the flag to true

}

client.subscribe("myhome/door/controlled"); // Subscribe to the topic

} else {

Serial.print("Connection to AWS IoT failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 5 seconds");

// Print detailed SSL error information

int ssl\_error = net.getLastSSLError();

Serial.print("Last SSL error code: ");

Serial.println(ssl\_error);

}

}

void loop() {

if (!client.connected()) {

// Reconnect if connection is lost

if (client.connect("ESP8266Client")) {

if (!awsConnected) {

Serial.println("Connected to AWS IoT");

awsConnected = true; // Set the flag to true

}

client.subscribe("myhome/door/controlled"); // Subscribe to the topic

}

}

client.loop();

// Check if 2 seconds have passed since the last message was received

if (millis() - lastMsgTime > 2000) {

digitalWrite(ledPin1, LOW); // Turn off red LED

digitalWrite(ledPin2, LOW); // Turn off yellow LED

digitalWrite(ledPin3, LOW); // Turn off green LED

}

// Read proximity sensor

int distance = proximity();

// Convert distance to string

char distanceStr[10];

itoa(distance, distanceStr, 10);

// Publish the distance

client.publish("myhome/door/proximity", distanceStr);

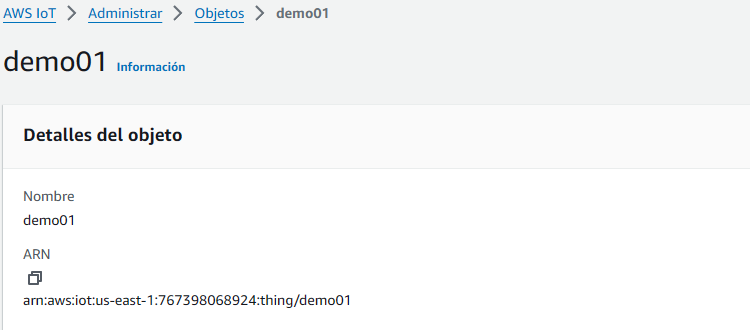
delay(1000); // Wait for 1 second before publishing again

}

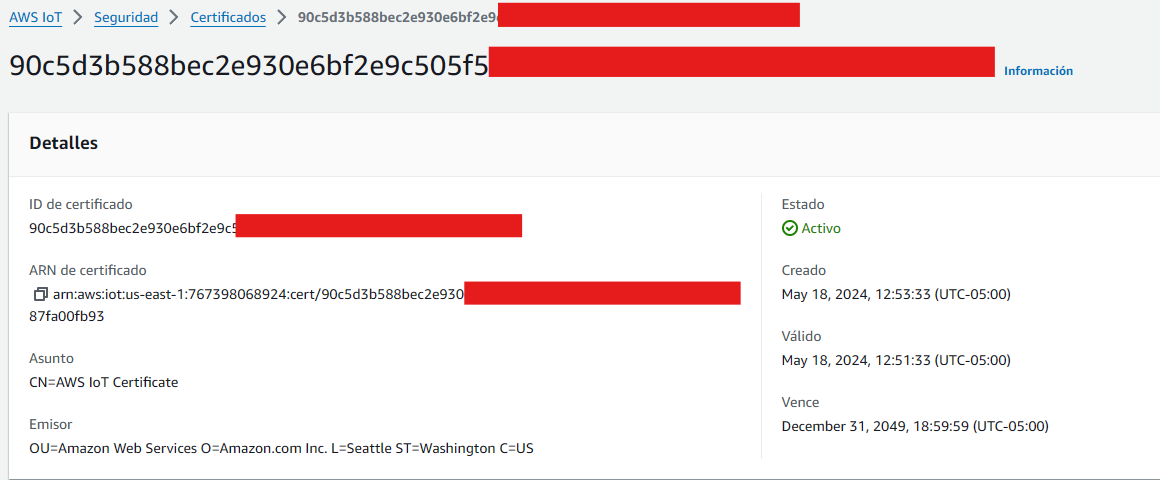
El código correspondiente se encuentra en el repositorio de github

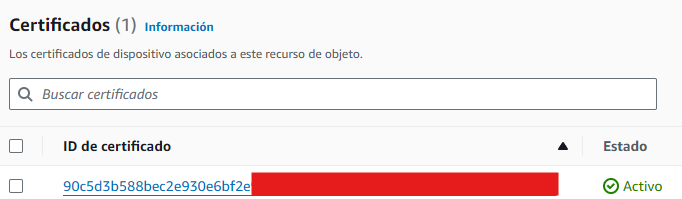
### **3.3. Configuración de AWS IoT**

* **Creación de un "Objeto"** en AWS IoT.

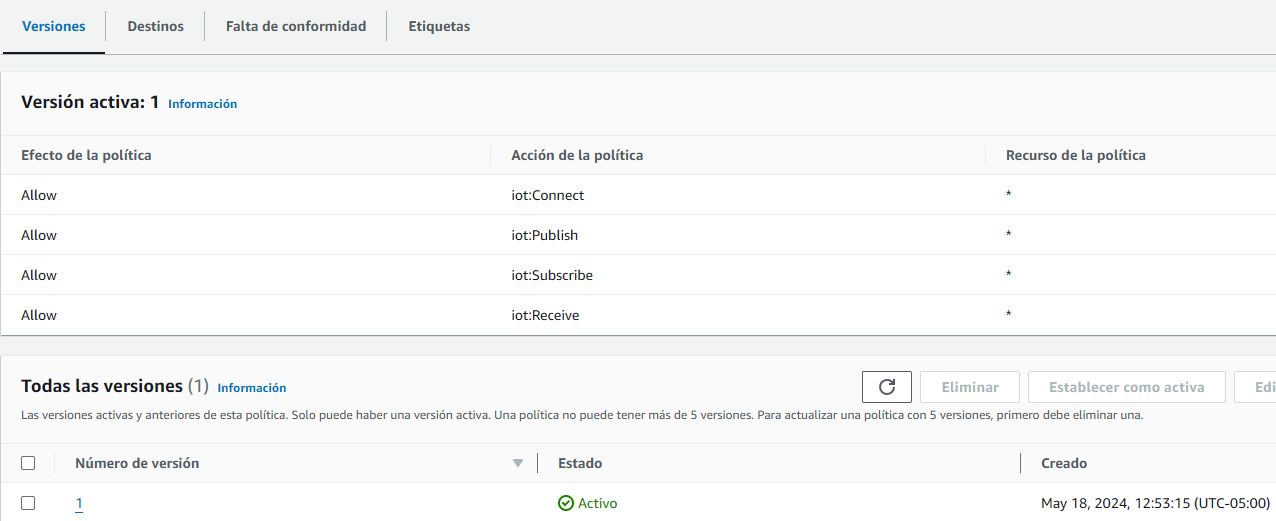


* **Creación de Certificados**: Descargar y asociar certificados al objeto.

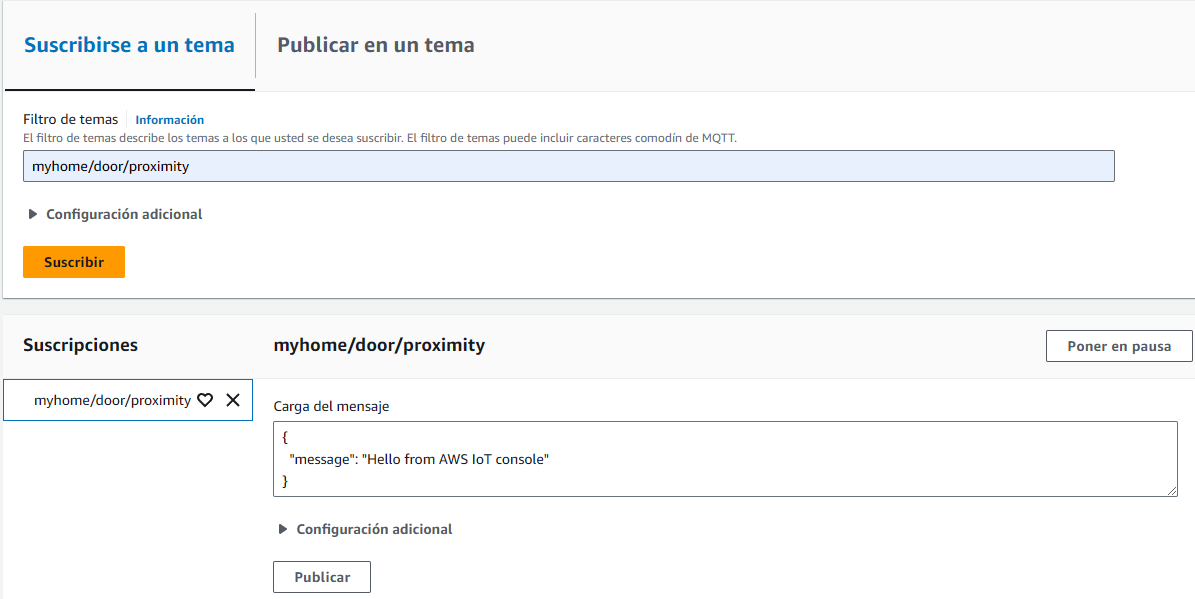


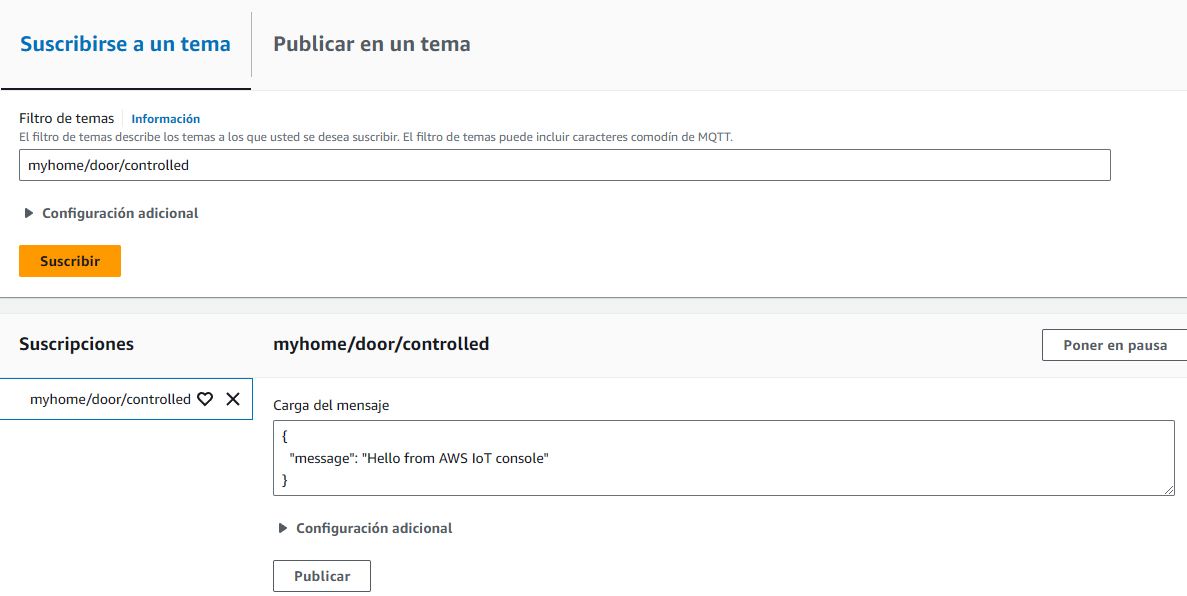


* **Definición de Políticas**: crear y asociar las políticas permiten suscripción y publicación en los tópicos proximity y controlled.



* **Configuración de Temas**: Configurar myhome/door/proximity para recibir datos del ESP8266 y myhome/door/controlled para recibir datos desde la aplicación Flutter.





### **3.4. Aplicación Flutter**

* **Dependencias en pubspec.yaml**
* **Código Flutter**

El código correspondiente se encuentra en el repositorio de github

**Certificados**: Los bytes de los certificados deben estar incluidos en la aplicación para la conexión segura.

### **4. Repositorios y Control de Versiones**

#### **4.1. Repositorio de Código Fuente**

* https://github.com/johandelevopment/IngenieriaSoftwareUdistrital