```
//MAIN FILE(ESP32_MAIN)



#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <time.h>
#include "Secrets.h" // Importing secrets
// IR Sensor Configuration
#define IR_SENSOR_PIN 34 // Analog pin for IR sensor
float distance_cm;
unsigned long lastMillis = 0;
const long interval = 5000; // Publish interval
#define AWS_IOT_PUBLISH_TOPIC "esp32/ir_sensor/data"
#define AWS_IOT_SUBSCRIBE_TOPIC "esp32/commands"
WiFiClientSecure net;
PubSubClient client(net);
time_t now;
time_t nowish = 1510592825;
// Function to synchronize time with NTP
void NTPConnect(void) {
Serial.print("Setting time using SNTP");
configTime(TIME_ZONE * 3600, 0, "pool.ntp.org", "time.nist.gov");
now = time(nullptr);
while (now < nowish) {
delay(500);
Serial.print(".");
now = time(nullptr);
}
Serial.println("done!");
struct tm timeinfo;
gmtime_r(&now, &timeinfo);
Serial.print("Current time: ");
Serial.print(asctime(&timeinfo));
}
// MQTT Message Received Callback
void messageReceived(char *topic, byte *payload, unsigned int
length) {
Serial.print("Received [");
Serial.print(topic);
Serial.print("]: ");
for (int i = 0; i < length; i++) {
Serial.print((char)payload[i]);
}
Serial.println();
}
// Connect to AWS IoT
void connectAWS() {
delay(3000);
WiFi.mode(WIFI_STA);
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.println(String("Attempting to connect to SSID: ") +
```

```
String(WIFI_SSID));
while (WiFi.status() != WL_CONNECTED) {
Serial.print(".");
delay(1000);
}
NTPConnect();
net.setCACert(cacert);
net.setCertificate(client_cert);
net.setPrivateKey(privkey);
client.setServer(MQTT_HOST, 8883);
client.setCallback(messageReceived);
Serial.println("Connecting to AWS IoT");
while (!client.connect(THINGNAME)) {
Serial.print(".");
delay(1000);
}
if (!client.connected()) {
Serial.println("AWS IoT Timeout!");
return;
}
// Subscribe to a topic
client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC);
Serial.println("AWS IoT Connected!");
}
// Function to Read and Convert IR Sensor Data
float getDistance() {
int sensorValue = analogRead(IR_SENSOR_PIN);
// Convert analog value to distance (for Sharp GP2Y0A21YK0F)
if (sensorValue > 20) {
return 4800.0 / (sensorValue - 20);
}
return -1; // Invalid reading
}
// Publish MQTT Message
void publishMessage() {
StaticJsonDocument<200> doc;
doc["time"] = millis();
doc["distance_cm"] = getDistance();
char jsonBuffer[512];
serializeJson(doc, jsonBuffer); // Convert JSON object to string
client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
}
void setup() {
Serial.begin(115200);
connectAWS();
}
void loop() {
distance_cm = getDistance();
if (distance_cm < 0) {
Serial.println(F("Invalid IR Sensor Reading!"));
return;
}
Serial.print(F("Distance: "));
Serial.print(distance_cm);
```

```
Serial.println(F(" cm"));
delay(2000);
now = time(nullptr);
if (!client.connected()) {
connectAWS();
} else {
client.loop();
if (millis() - lastMillis > interval) {
lastMillis = millis();
publishMessage();
}
}
}
```

```
//SUB FILE(ESP32_SCERET.h)
```

```
#include <pgmspace.h>

#define SECRET

const char WIFI_SSID[] = "HOME_2.4G";
const char WIFI_PASSWORD[] = "Panaganti";

#define THINGNAME "ESP32_DHT11"

int8_t TIME_ZONE = 0; // Adjust as per your region

const char MQTT_HOST[] = "a10vxnmzceu0lb-ats.iot.ap-
south-1.amazonaws.com";

// AWS Certificates
static const char cacert[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExBBbWF6
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFowOTEL
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1hem9uIFJv
b3QgQ0EgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHHKeNXj
ca9HgFB0fW7Y14h29Jlo91ghYPl0hAEvrAIthtOgQ3pOsqTQNroBvo3bSMgHFzZM
9O6II8c+6zf1tRn4SWiw3te5djgdYZ6k/oI2peVKVuRF4fn9tBb6dNqcmzU5L/qw
IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwhmahRWa6
VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08p8kDi1L
93FcXmn/6pUCyziKrlA4b9v7LWIbxcceVOF34GfID5yHI9Y/QCB/IIDEgEw+OyQm
jgSubJrIqg0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMC
AYYwHQYDVR0OBBYEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3DQEBCwUA
```

```
A4IBAQCY8jdaQZChGsV2USggNiMOruYou6r4lK5IpDB/G/wkjUu0yKGX9rbxenDI
U5PMCCjjmCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxhlbI1Bjjt/msv0tadQ1wUs
N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWlJbYK8U90vv
o/ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFo6IQ6XU
5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXeGADbkpy
rqXRfboQnoZsG4q5WTP468SQvvG5
-----END CERTIFICATE-----
)EOF";

// Copy contents from AWS Thing's certificate
static const char client_cert[] PROGMEM = R"KEY(
-----BEGIN CERTIFICATE-----
MIIDWjCCAkKgAwIBAgIVAPty6mpPmTiP9iP1BwB6V8J138elMA0GCSqGSIb3DQEB
CwUAME0xSzBJBgNVBAsMQkFtYXpvbiBXZWIgU2VydmljZXMgTz1BbWF6b24uY29t
IEluYy4gTD1TZWF0dGxlIFNUPVdhc2hpbmd0b24gQz1VUzAeFw0yNTAzMDYxOTEx
MjhaFw00OTEyMzEyMzU5NTlaMB4xHDAaBgNVBAMME0FXUyBJb1QgQ2VydGlmaWNh
dGUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCpISTJ9PNstTRsSneD
WrWDsDLWGJkUXDDxaJl/P+6fc3D8jFprJ+f0FKSCMjwQZzkxcTpiV7xK0bkybDSv
VUrguV/owPXLDYZoNKviWw7IdIQSWFQiW8W0xzWTbwkKlDPE9XDyMUq9GoIs5KGv
Cgxbf1KNxUbteshsJYl6Sx+loZk8qXk9G4nE8FS4m+DyJpi7ECV7itTWrQBldska
x+PVjeOtqUYYroBGS2NbkQaz+qZTbNfRd0ca9LU8FbASrn40Y/UIJmB5+deznNBm
ycKk/CbDQ7abAFCweftY6AOqinQlZ0wZWt0IHZyA2HAhSUhtJDxs4BTqobY6cz/r
+fxXAgMBAAGjYDBeMB8GA1UdIwQYMBaAFI4m2d8eaH1kkPvCpeYeShvKUtbQMB0G
A1UdDgQWBBRefw8EV378hpaXoYdIXMuXuLuq4jAMBgNVHRMBAf8EAjAAMA4GA1Ud
DwEB/wQEAwIHgDANBgkqhkiG9w0BAQsFAAOCAQEASDVtT3yyH6ETOdTYEurUTh8K
n0f2CRQSKt6KDXIA/LGwIvLNc1gLKk6V69lSo5+eLuMEOCYMoidDcO6K9CesHJev
6I/FWBhY7Vts4bg3JCdVXFjfTATLboGHBc5F97ehXSaTm75HYO+DsiW6iHYD1PkB
T6U81duf/NwaLz4FjuTZuzdrqI6194XowuKY7vdqn6dXuXvnNY3BNPd7wgGrsCuC
D1sai/Y6WTz9/sfnKBnuqpQdWwUCW+fV8urSfIgwC1/tjumAX1neTEaMpwluiCrs
Bd9zvNaNhYW/moQ/acmRqbvZMR7BVu/CHzeZY0nZjexNWnDoF2M+JUCkohZ4lg==
-----END CERTIFICATE-----

)KEY";

// Copy contents from AWS Thing's private key
static const char privkey[] PROGMEM = R"KEY(
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAqSEkyfTzbLU0bEp3g1q1g7Ay1hiZFFww8WiZfz/un3Nw/Ixa
ayfn9BSkgjI8EGc5MXE6Yle8StG5Mmw0r1VK4Llf6MD1yw2GaDSr4lsOyHSEElhU
IlvFtMc1k28JCpQzxPVw8jFKvRqCLOShrwoMW39SjcVG7XrIbCWJeksfpaGZPKl5
PRuJxPBUuJvg8iaYuxAle4rU1q0AZXbJGsfj1Y3jralGGK6ARktjW5EGs/qmU2zX
0XdHGvS1PBWwEq5+NGP1CCZgefnXs5zQZsnCpPwmw0O2mwBQsHn7WOgDqop0JWdM
GVrdCB2cgNhwIUlIbSQ8bOAU6qG2OnM/6/n8VwIDAQABAoIBAG9ACvgBdWHra/lT
nnu73iwqabYk5gTbg6I4QMfQYQFnqUmmdQ+JZ21JatDthE7x51Abf+ESfxvkCmMd
hCZU/3UE2w66ZrPTSLcg9nNlnDrFUd7TVTxoi4sXgqlhX1wjEk6beFkHjXcIidpH
5gbCoFBLVx1ESByMKGvi4SSdZ8rFiGQXKOPJQxROCUyL4HUeSLFpIpwStGzWO12G
CF3pjAOc3AhR0Ko/YWfKB/4dePisOt+qCfIb8KAWNIwxIqFgFgRwYPNqcslH7TIi
bhuJoRJTna7DOK+bNKAw+HiE8h+LKoqwp9z3n6RgThsGr5yda1SFGHH+qYePPqih
Io92P6ECgYEA28S9RfpJnsJ6+gcGASmP2gk/IzypFz8IZDt/mbHOnI8qz0ztpACC
9qHJBmZW0xdX7WAy6XeZI1Ajaaqn+B5MqC/XZrkAtG+kxGPKCU4PUEpBEBz8yIKz
eCeW7F/svebCDDt+UhxRhj2WZ+5wH/7I6m4GNEYEozKFx+l348JM6TECgYEAxQMz
DsmXhS4NQX/YFO03+XshYqem2J7zBp4mEFivZJF8OBcuLgP0ktWU9Si9kjdZNpic
uyqzDVbAeUkl2MRy5RGOzF32D/lVEREUfzfMvUktvGCK55fmZQMRE9JFuAd8NxzN
IL6xIDqRpGuc5+ziFBlZzWoIOpFPB6T+gBSyXAcCgYAsO+o41orQ6KYmAxnjTAFH
```

```
SospxYDnS+mdexhQ0Cgc/PgJ55sBYpT5nVC/+ANL06v//FfjBA6PvAaUsyPB3h9h
ztdVNrF8n+gTG17EBrEwYPG9k4gRt7T130oSY7Bb/MFcAvGHB2olUeofGy/4UKkR
/nlDNkCitH2W4lgeQLKIUQKBgAOiaHm4XFQ3okT4D8trCOJQVDBWuWw7mm4f0sHw
cOPhDzKVsuNkbUmcKRBQvRaKcNnh/NozEnSHiitynwEdtqTlKwY4IYbv6ZLUBtXF
+L/xLqfP/CADnFt2sAT2lxSrBq6ZmGsnOWtHyVIILi/VHeKTCSYR867o6VVRxOQl
/ZhXAoGATGLyTOOAe73c2Pter8A05s1ugYUlVmxNFl0u78pp6UrsJeJGuCgMGEGY
cntI0YAQMvgjRj9QbQyim9Mhj2shCdQWExX/TWo/4vdLE7+M/mYO2Dv3VaqS54O3
uMPSfS5pRZVBFsJe0Ko6NaVrhOdDFkwJshNTCOCzhpf3JRzQzCc=
-----END RSA PRIVATE KEY-----

)KEY";
```