Electronics News &

EE Times

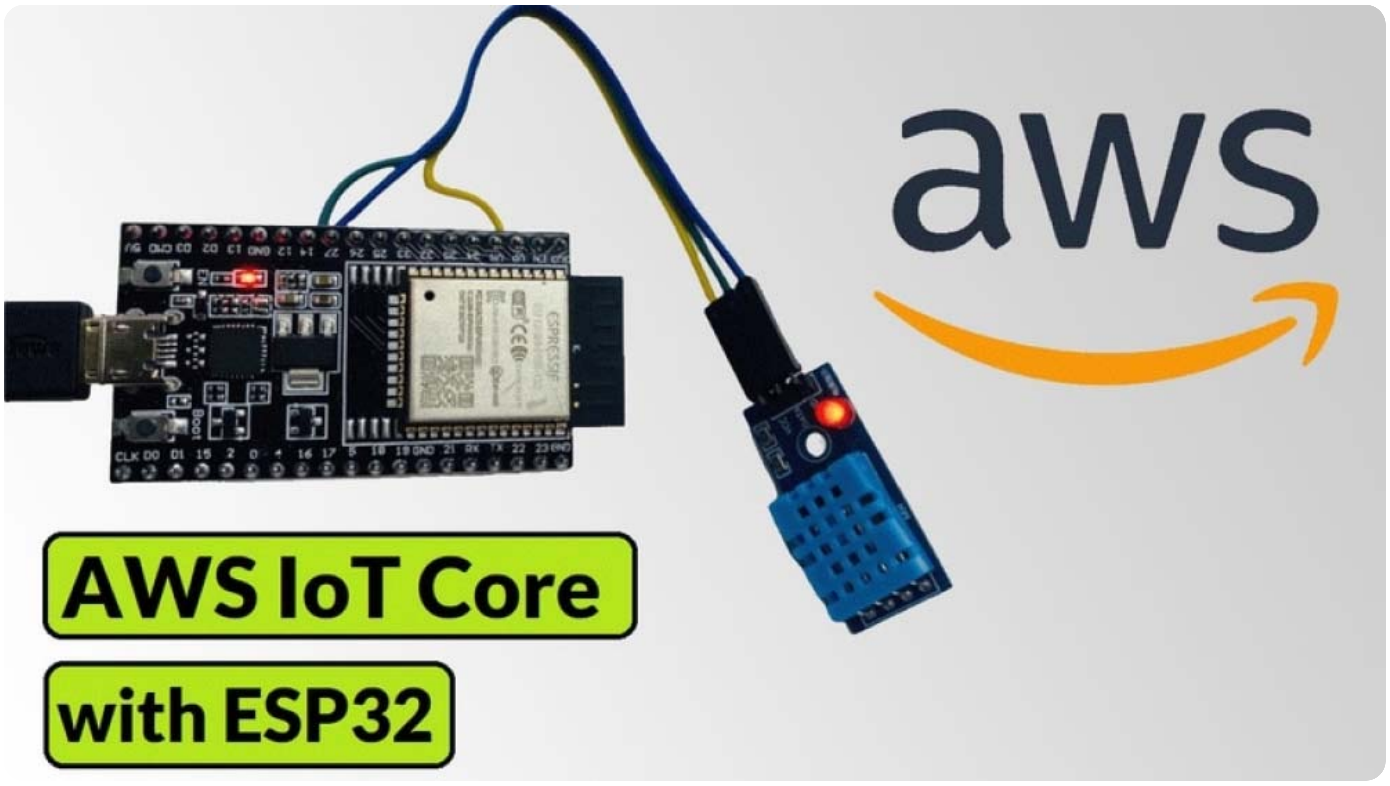
Home » Connecting ESP32 to Amazon AWS IoT Core using MQTT

ESP32 PROJECTS IOT PROJECTS

Connecting ESP32 to Amazon AWS IoT Core using MQTT

By **Admin** — Updated: August 20, 2022 10 Comments 9 Mins Read

Share



Overview

This is a getting started tutorial about **Amazon Web Services**, i.e **AWS IoT Core** with **ESP32**. The AWS IoT Core is a managed **cloud service** that lets connected devices easily and securely interact with **cloud applications** and other devices.

In this tutorial, we will learn how you can connect the **ESP32 with AWS IoT Core** & publish sensor reading to **AWS MQTT**. For a demo, we will use the **DHT11 Sensor** and read the humidity temperature data. The ESP32 will connect to the local WiFi network and will post the DHT11 Sensor data to **AWS IoT Cloud**. Not only posting data, but we can also receive the data from AWS Dashboard. Earlier we posted the sensor Data to AWS Dashboard using **AWS IoT Core & ESP8266**.

The tutorial comprises multiple sections

- Creating **Policy** and attaching to Thing
- Generating **Certificates**
- Modifying Arduino Sketch according to Thing Data & Credentials
- **Subscribe** & **Publish** Data to and from AWS Dashboard

This tutorial is for beginners who want to learn about the **Amazon AWS IoT Core** for **IoT Applications**. Earlier we learned about the IoT platform like **Google Firebase** and **Arduino IoT Cloud**. But with AWS IoT Core, you can build and manage devices for commercial applications.

Hardware Setup

The hardware required for this project is an ESP32 Wifi Module. And for the sensor part, we will use DHT11 Humidity and Temperature Sensor.

Connect the DHT11 Sensor to ESP32 Board as per circuit diagram here.

You can use a breadboard for connection or simply use a male-to-female connector wire.

AWS offers Internet of Things (IoT) services and solutions to connect and manage billions of devices. These cloud services connect your IoT devices to other devices and AWS cloud services. AWS IoT provides device software that can help you integrate your IoT devices into AWS IoT-based solutions. If your devices can connect to AWS IoT, AWS IoT can connect them to the cloud services that AWS provides.

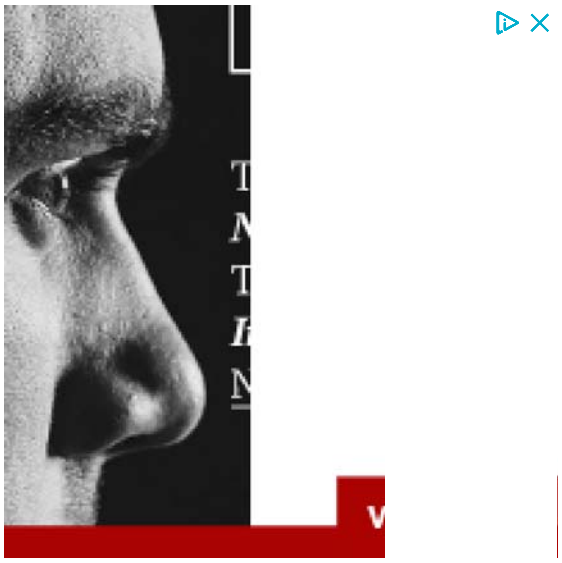
AWS IoT lets you select the most appropriate and up-to-date technologies for your solution. To help you manage and support your IoT devices in the field, AWS IoT Core supports these protocols:

- MQTT (Message Queuing and Telemetry Transport)
- MQTT over WSS (Websockets Secure)
- HTTPS (Hypertext Transfer Protocol – Secure)
- LoRaWAN (Long Range Wide Area Network)

Signing in

Go to your web browser and search visit following link: **aws.amazon.com/iot-core/**.

Basically we need to set up the AWS Account now. Therefore create an account using the email ID and password. The account also requires your bank Credit card information. There will be no charges but AWS just needs a verification using your bank account. It will also ask for phone number verification. Hence the account will be successfully created.



After successfully signing in, the AWS Management Console window will open. In the services search tab at the top write 'IoT core' & hit enter.

You can click on IoT Core, so an AWS IoT Dashboard will appear now.

On the left side of the dashboard, there are so many options. But we need to work with two options here. One is the **manage** option and the other one is the **secure** option.

Creating a Thing

Now we need to create a thing associated with our project. For this, follow the following steps:

- Specifying thing properties
- Configuring device certificate
- Attaching policies to certificate

Under the manage option click on Thing. Now we need to create a Thing here. So, click on Create

You can select whether create a single thing or create many things. But for our applications, select create a single thing. Then click on Next.

Under additional configurations, there is no need to make any changes.

Under the device shadow option, select the first option as No shadow. Then click on Next.

EE|Times
The Electronics News You Want.
The Reliable Insights You Need.

Generate Device Certificate

But the AWS recommendation is to select the Auto Generate New Certificate. Then click on Next.

Create & Attach Policy

Now we need to attach a policy to the Things we created. But no policies are here right now. So we need to create a policy first.

So click on create policy. Here give any name to the policy. For example, I will give it a name as "***ESP32_Policy***".

Now the add statement part is very important. Under the action, type IoT. So multiple options will pop up. From here we will only need to publish, Subscribe, Connect and Receive.

Now click on create to create the policy. So the policy has been created successfully.

Downloading Certificates and Keys

Now we need to download the required certificates from this list.

First, download the **device certificate** and then rename it as a device certificate for identification.

Also, download the **public key** and rename it as a public key. Then download the **private key** and rename it as a private key.

So we have downloaded all the certificates that we need for our project.

EE|Times
The Electronics News You Want.
The Reliable Insights You Need.

Installing Necessary Arduino Libraries

1. ArduinoJSON Library

So first go to the library manager and search for “JSON” & install the library as shown in the figure below.

2. PubSubClient Library

Again go to the library manager and search for “PubSubClient” & install the library from Nick O’Leary.

3. DHT11 Sensor Library

Search for “dht11” & install the library as shown below.

Source Code/Program for connecting AWS IoT Core with ESP32

The code/program that **interfaces ESP32 with DHT11 Sensor** & connects to the **Amazon AWS IoT Core** is written in Arduino IDE. The code is divided into two sections. One is the main ino file and other the header file.

Main .ino File

Open a new sketch in Arduino IDE & paste the following code and save it.

```

#include "DHT.h"
#define DHTPIN 14     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11

#define AWS_IOT_PUBLISH_TOPIC   "esp32/pub"
#define AWS_IOT_SUBSCRIBE_TOPIC "esp32/sub"

float h ;
float t;

DHT dht(DHTPIN, DHTTYPE);

WiFiClientSecure net = WiFiClientSecure();
PubSubClient client(net);

void connectAWS()
{
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.println("Connecting to Wi-Fi");

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }

  // Configure WiFiClientSecure to use the AWS IoT device credentials
  net.setCACert(AWS_CERT_CA);
  net.setCertificate(AWS_CERT_CRT);
  net.setPrivateKey(AWS_CERT_PRIVATE);

  // Connect to the MQTT broker on the AWS endpoint we defined earlier
  client.setServer(AWS_IOT_ENDPOINT, 8883);

  // Create a message handler
  client.setCallback(messageHandler);
```

```
    Serial.print(".");
    delay(100);
  }

  if (!client.connected())
  {
    Serial.println("AWS IoT Timeout!");
    return;
  }

  // Subscribe to a topic
  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC);

  Serial.println("AWS IoT Connected!");
}

void publishMessage()
{
  StaticJsonDocument<200> doc;
  doc["humidity"] = h;
  doc["temperature"] = t;
  char jsonBuffer[512];
  serializeJson(doc, jsonBuffer); // print to client

  client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
}

void messageHandler(char* topic, byte* payload, unsigned int length)
{
  Serial.print("incoming: ");
  Serial.println(topic);

  StaticJsonDocument<200> doc;
  deserializeJson(doc, payload);
  const char* message = doc["message"];
  Serial.println(message);
}

void setup()

void loop()
{
  h = dht.readHumidity();
  t = dht.readTemperature();


  if (isnan(h) || isnan(t) )  // Check if any reads failed and exit early (to
  {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.println(F("°C "));

  publishMessage();
  client.loop();
  delay(1000);
}
```

Fastweb Mobile

150Giga con 5G incluso, 100 SMS e Chiamate Illimitate a 7,95€ al Me

Ad Fastweb

Secrets.h

```
#define SECRET
#define THINGNAME "***************"                         //change this

const char WIFI_SSID[] = "***************";               //change this
const char WIFI_PASSWORD[] = "***************";           //change this
const char AWS_IOT_ENDPOINT[] = "***************";       //change this

// Amazon Root CA 1
static const char AWS_CERT_CA[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----
)EOF";

// Device Certificate                                          //change tl
static const char AWS_CERT_CRT[] PROGMEM = R"KEY(
-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----


)KEY";

// Device Private Key                                          //change tl
static const char AWS_CERT_PRIVATE[] PROGMEM = R"KEY(
-----BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----


)KEY";
```

Modifying Arduino Sketch according to the Thing

Now it’s time to modify the Arduino Sketch File. Go to ***secrets.h*** tab and begin the modification.

Paste the thing name to the following line of code.

```
#define THINGNAME "***************"
```

Under the WiFi SSID and password, enter the WiFi SSID and Password of your local network.

```
const char WIFI_SSID[] = "***************";
const char WIFI_PASSWORD[] = "***************";
```

Now, we need to insert the AWS IoT Endpoint here. To get the endpoint, go to the settings part of AWS Dashboard. Yow will get the endpoint.

Click on the copy icon to copy the endpoint. Go back to Arduino IDE and paste it on the following line.

```
const char AWS_IOT_ENDPOINT[] = "***************";
```

You need to insert the Amazon Root CA1 in between the following line.

```
// Amazon Root CA 1
static const char AWS_CERT_CA[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----
)EOF";
```

So for this, we need to go back to the certificate that we downloaded earlier. Open this file with Notepad++ and copy all the text.

Then go back to Arduino IDE and insert the copied text between begin certificate and the end the begin certificate and end certificate section.

```
// Device Certificate
static const char AWS_CERT_CRT[] PROGMEM = R"KEY(
-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----
```

Under the "**Device Private Key**", we need to insert the device's private key. So go to the downloaded folder again and open the device's private key file using Notepad++. Again copy the text and paste it between begin & end parts.

```
// Device Private Key
static const char AWS_CERT_PRIVATE[] PROGMEM = R"KEY(
-----BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----
```

So all the modification of the Arduino ESP32 Sketch related to AWS IoT Core is done now.

EE|Times
The *Electronics News* You Want.
The *Reliable Insights* You Need.

upload option to upload the code to the ESP32 board.

Once the code uploading is done, open the Serial Monitor. The ESP32 will try connecting to the WiFi Network. Once it gets connected to the WiFi Network, it will try connecting to the AWS IoT Server.

Subscribing Sensor Data to AWS Dshboard

The same thing should also be posted to the AWS Server. To check that, go to the test section of AWS Dashboard. Under the test section, we have an option for subscribe and publish.

Now to see the data, you need to subscribe to a topic. For that type ***“esp32/pub“*** under the topic filter section. In the additional configuration, you can make changes if you want.

Then click on subscribe. When you hit the subscribe button, immediately the data from ESP32 will be uploaded to AWS Dashboard. Thus, you have successfully sent the DHT11 Sensor data to Amazon AWS IoT Core using ESP32.

The data is updated here after an interval of every one second. This is really amazing as we are able to receive the data to AWS IoT Core Dashboardsent from ESP32 via MQTT protocol. This is how we read the subscribed data.

EE|Times
The *Electronics News* You Want
The *Reliable Insights* You Need.

Publishing Data to Serial Monitor

Now let’s see if we are able to publish the data from AWS IoT core to ESP32 or not.

Now to see the data, you need to publish to a topic. For that type ***“esp32/sub“*** under the topic filter section. Under additional configuration do nothing. Then click on publish.

Immediately you can see the message sent to the Serial Monitor. This is amazing again. You can use this method to **Control an LED** using publish method.

This is how you can **send** or **receive** data from **Amazon AWS IoT Core** using **ESP32**. Using **AWS MQTT**, we can subscribe to sensor readings topics published by various **IoT nodes**. This is a basic beginner’s tutorial for the users who want to get started with **Amazon Web Services** for their **IoT devices**.

Video Tutorial & Guide

Getting Started with Amazon AWS IoT Core using ESP32 || Creating Thing, Policy & Certificates

Watch this video on YouTube.

RELATED POSTS

IoT Smart Exhaust Fan: ESP32 Based Monitoring & Control

IoT Solar Panel Monitoring System with ESP8266 & MQTT
Updated: December 5, 2023

Alexa & ESP32 Based Smart & Manual Home Automation System
Updated: November 21, 2023

Smart Density Based Traffic Light Control System with ESP32 & Blynk

Smart & Manual Home Automation with Alexa & ESP8266

Updated: November 20, 2023

LoRa-Enabled IoT Geo-Fencing using GPS, Arduino & ESP8266

Updated: October 23, 2023 2

VIEW 10 COMMENTS

AM EX
SPESE PERSONALI O PROFESSIONALI?
AMERICAN EXPRESS
ANDREA TESTA
GIANCARLO BAGNETTI
BAGNETTI SRL
American Express ha la Carta ch
Scopri le promozioni
Messaggio Pubblicitario con finalità promozionale. Consult
contrattuale e precontrattuale su americanexpress.it/ te

Latest Posts

Rain Detection using Raindrop Sensor Module & Arduino
December 24, 2023

How to use HC-SR501 PIR Motion Sensor with Arduino
December 23, 2023

IoT Smart Exhaust Fan: ESP32 Based Monitoring & Control
December 8, 2023

IoT Solar Panel Monitoring System with ESP8266 & MQTT
December 5, 2023

How to use IR Sensor Module with Arduino – Simple Guide

How to use LDR Sensor Module with Arduino
November 27, 2023

Interfacing MPU6050 Accelerometer & Gyroscope with Arduino
November 27, 2023

IoT Applications for Enhanced Living & Working Environments
November 21, 2023

Top Posts & Pages

ECG Graph Monitoring with AD8232 ECG Sensor & Arduino

Pulse Rate (BPM) Monitor using Arduino & Pulse Sensor

Measure Soil Nutrient using Arduino & Soil NPK Sensor

ESP32 CAM Object Detection & Identification with OpenCV

Designing of MPPT Solar Charge Controller using Arduino

Interfacing SX1278 (Ra-02) LORA Module with Arduino

IoT Based Electricity Energy Meter using ESP32 & Blynk

Temperature Based Fan Speed Controller using Arduino

Categories

Arduino Projects	(184)
Articles	(50)
Learn Electronics	(19)
Product Review	(13)
Tech Articles	(20)

Power Electronics	(13)
IoT Projects	(172)
ESP32 MicroPython	(7)
ESP32 Projects	(54)
ESP32-CAM Projects	(13)
ESP8266 Projects	(73)
LoRa/LoRaWAN Projects	(22)
Microcontrollers	(36)
AMB82-Mini IoT AI Camera	(3)
BLE Projects	(17)
STM32 Projects	(19)
Raspberry Pi	(82)
Raspberry Pi Pico Projects	(49)
Raspberry Pi Pico W Projects	(11)
Raspberry Pi Projects	(22)

J@LC JLCPCB

FOLLOW US

Facebook

Twitter

Pinterest

Instagram

YouTube

ABOUT US

"'How to Electronics' is a vibrant community for electronics enthusiasts and professionals. We deliver latest insights in areas such as Embedded Systems, Power Electronics, AI, IoT, and Robotics. Our goal is to stimulate innovation and provide practical solutions for students, organizations, and industries. Join us to transform learning into a joyful journey of discovery and innovation.