MGS 602 IoT Project

~ Avinash Vishala Nagaraja

50482713

The corresponding report will contain the steps involved in setting up an IoT device that could sense the Temperature and Humidity of the room and report it back on the webpage.

Hardware Requirements:

1. ESP32 Microcontroller
2. DHT22 Sensor
3. Breadboard and Wire Kit

Software Requirements:

Arduino IDE

Introduction

ESP32 Microcontroller is a feature rich MCU with integrated Wi-Fi and Bluetooth connection for a variety of applications. It has a durable design that can withstand temperatures between -40C to 125C. With built-in antenna switches, RF baluns, power amplifiers, low noise receive amplifiers, filters, and power management modules, the ESP32 is highly integrated. ESP32 achieves extremely low power consumption and is designed for mobile devices, wearable electronics, and Internet of Things applications. It can operate as a fully independent system or as a slave device to a host MCU. Through its SPI/SDIO or I2C/UART interfaces, ESP32 may connect to other systems to provide Wi-Fi and Bluetooth capability.

DHT22 is a straightforward, inexpensive digital temperature and humidity sensor. It measures the humidity in the air using a thermistor and a capacitive humidity sensor, and it outputs a digital signal on the data pin (no analog input pins needed). Although reasonably easy to operate, data collection requires precise timing. The sensor's only significant drawback is that it can only provide fresh data once every two seconds, which means that when utilizing the library, sensor values could be as much as two seconds outdated.

Steps Involved in connection:

Task 1:

For a simpler connection between the controller and the DHT sensor, the microcontroller is fastened to the breadboard. The microcontroller's 3V3 (3 Volts Pin) is connected to the sensor's positive pin, and the ground pin is connected to the sensor's negative pin. The sensor's data pin is connected to the data input pin D4, which facilitates the data transfer between the two.

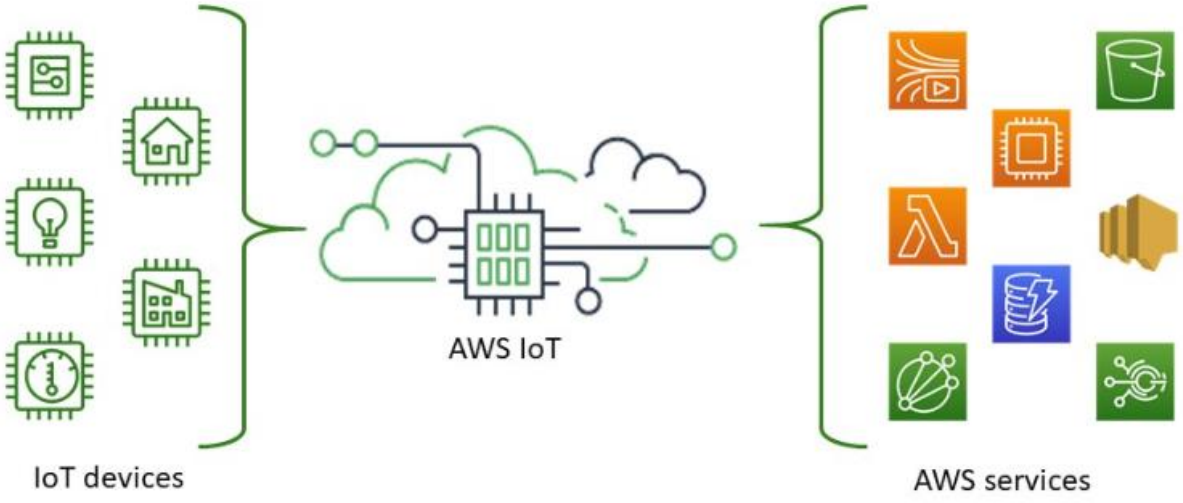
Once all the physical connections are complete, we would use the Arduino IDE program to upload the code to the microcontroller. To compile the code, we would need to download all the necessary libraries. And by rendering the HTML code on the controller and accessing the webpage using the appropriate IP for the device, we would enable a connection to the microcontroller over our WIFI routers, allowing us to get real-time readings through a webpage on your computers.

Task 2:

Once the device is operational, we will assign ports to our home routers using port triggering, which can be done using the mobile application of the service provider. The system is set up so that the device responds to requests through port 80 internally and answers through port 60200 externally. The public IP address can then be used to obtain the sensor's reading: **98.11.173.152:60200**

Task 3:

Multiple IoT devices might be challenging to manage, therefore we can use the AWS IoT services to make it simple. A managed cloud service called AWS IoT Core enables connected devices to quickly and securely communicate with other gadgets and cloud-based software.



You can choose the most suitable and cutting-edge technology for your solution using AWS IoT. AWS IoT Core supports the following protocols to assist you in managing and supporting your IoT devices in the field:

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

It's easy to get started with AWS; simply establish an account and take advantage of Amazon's free trial. Once the account is created, you'll have access to a variety of Amazon services that you can use to finish this project.

We would use AWS IoT Core to connect our IoT device. To do this, we would first create a thing in the IoT core console and then describe the thing's parameters, such as its name.

Once completed, we must provide the device certificates that will enable the device to authenticate itself to the cloud. After that, we need to assign each of the required policies for our project. For this project, we'll use the following three policies:

- Publish
- Subscribe
- Connect
- Receive

Once done assign the policy to the device and then download all the necessary files,

- device certificate: For the device identification
- private key: to communicate through encryption
- Root CA1 certificate: for authentication

Then download all the necessary libraries for the code to compile and upload to the Microcontroller. When the controller is enabled and connects to the Aws IoT the data is uploaded to the AWS. Once you subscribe to “esp32/pub” we would be able to view the data. Immediately you can see the message sent to the Serial Monitor.

Network Diagram