

Hanoi University of Science and Technology
School of Information and Communication Technology
— o0o —



Graduation Research 1

Final Report

Advisor : Dr. Truong Thi Dieu Linh

Student : Nguyen Tieu Phuong - 20210692
Global ICT Program

Class : ICT-01, Cohort 66

Hanoi - 2023

Project Overview

This project aims to provide an introduction to Arduino and basic IoT development. The primary objectives are as follows:

- Construct a circuit to locally measure humidity and temperature by acquiring proficiency in Arduino programming and circuit wiring to interface with the sensor.
- Establish connectivity of the circuit to the cloud using the MQTT protocol to publish the acquired data.
- Implement remote data retrieval from the cloud-based system.

The research will focus on practical implementation and theoretical understanding of basic IoT principles, sensor interfacing, and cloud-based communication protocols. For full source codes and documentation, please visit this [GitHub repository](#).

Requirements

The following hardware and software components will be essential for the successful implementation of the project:

Hardware

- NodeMCU 1.0 (ESP-12E Module)
- Arduino Uno
- DHT11 Sensor
- Breadboard
- Jumpers
- Resistors
- Adapters

Software

- Arduino IDE 2.0
- CP2102 driver (for NodeMCU)
- Arduino libraries: DHT, ESP8266WiFi, Ticker, AsyncMqttClient
- Python library: Paho-MQTT

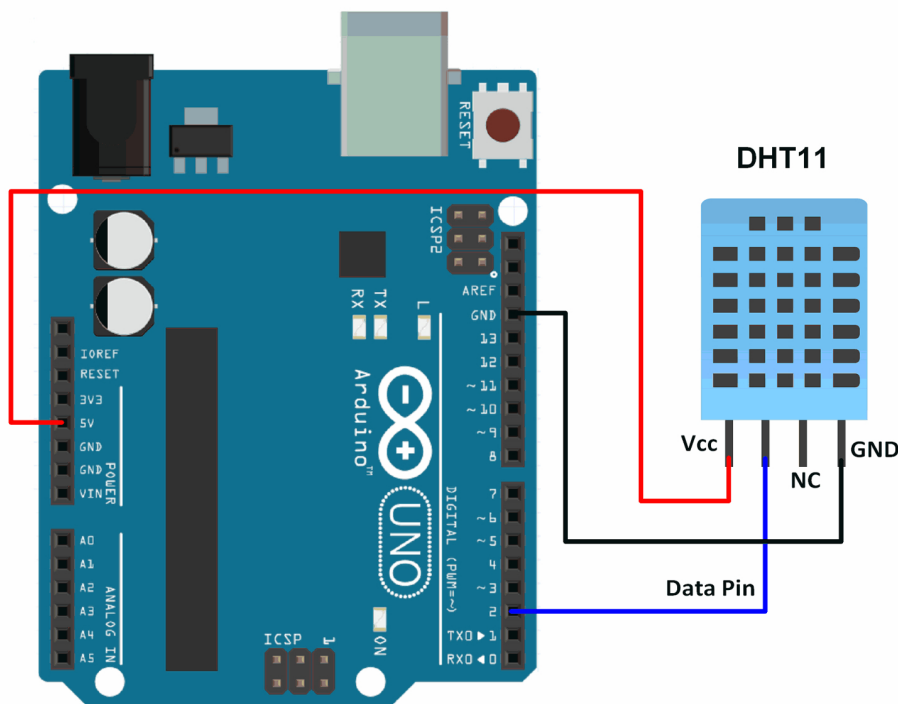
Reading Temperature and Humidity Locally

1. Building the Circuit

Reading data from a DHT11 sensor is fairly simple. Connect the components according to the schematic diagram.

The DHT11 sensor is shipped in two versions: 3-pin type and 4-pin type. These types are identical in function, the 3-pin type is actually the 4-pin type with the additional pin hidden and acts as a pulled up resistor. We do not need to connect this pin in the 4-pin sensor.

Lastly, navigate to **Sketch > Include Library > Manage Libraries** and install the **DHT sensor library** by Adafruit in Arduino IDE.



2. Write the Code

Copy and paste the following code into the Arduino IDE:

```
#include "DHT.h"
const int DHTPIN = 2;    // depends on your wiring
const int DHTTYPE = DHT11;

DHT dht(DHTPIN, DHTTYPE);
```

```
void setup() {  
  Serial.begin(9600); // Start serial communication  
  dht.begin();        // Initialize the DHT sensor  
}  
  
void loop() {  
  float h = dht.readHumidity();  
  float t = dht.readTemperature();  
  
  if (isnan(h) || isnan(t)) {  
    Serial.println("Failed to read from DHT sensor!");  
    return;  
  }  
  
  Serial.print("Humidity: ");  
  Serial.print(h);  
  Serial.print("%  Temperature: ");  
  Serial.print(t);  
  Serial.println("°C");  
  
  delay(2000); // Wait 2 seconds before next reading  
}
```

Connect the Arduino to your computer using the USB cable. Select the correct board and port in the Arduino IDE. Click the "Upload" button to transfer the code to the Arduino.

3. Final Results

Open the serial monitor by going to **Tools > Serial Monitor**. We should now see the temperature and humidity readings displayed in the serial monitor, updating every 2 seconds.

Using MQTT to Send and Receive Data Remotely

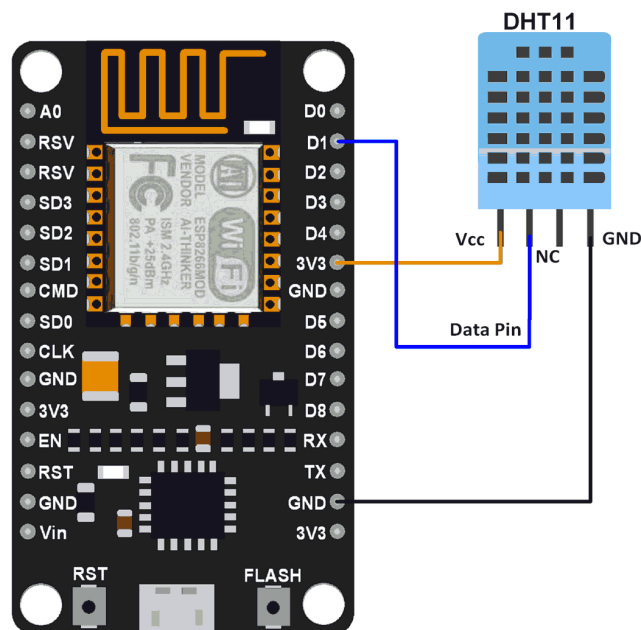
About MQTT

The **Message Queuing Telemetry Transport (MQTT)** protocol is a lightweight messaging protocol designed for communication between resource-constrained devices in the Internet of Things (IoT). It employs a pub-sub architecture, making it ideal for efficient data exchange in low-bandwidth networks.

1. **Devices (clients):** These are sensors, actuators, or other IoT devices that publish or subscribe to data.
2. **Broker:** This is the central server that acts as a message intermediary between clients.
3. **Topics:** These are channels by which messages are published and subscribed to.

1. Building the Circuit

Since Arduino Uno does not support WiFi, we need another module that supports Internet connectivity. For this project, NodeMCU 1.0 is chosen. The circuit building is similar to that of Arduino in the previous section. Before using, the CP2102 driver for NodeMCU needs to be installed.



2. Setup the MQTT Broker

The MQTT broker can be either a self-hosted or paid solution from various MQTT providers. For the scope of this project, the Free Public MQTT Broker by HiveMQ is used. This broker is available under `broker.hivemq.com`, at TCP port 1883.

3. Setup the MQTT Client

A client can be a publisher (push data to the broker), or a subscriber (fetch data from the broker), or both. We will set up our circuit to be the publisher, which publish data under two topics `esp/dht/temperature` and `esp/dht/humidity` and our local machine to be the subscriber to these topics.

Publisher

We will connect our circuit to the Internet via WiFi, and then the MQTT broker. Once the connection is established, we create the topics and begin the process of pushing data to the cloud.

The full source code for the publisher is available [here](#).

```
void setup() {
  Serial.begin(115200);
  Serial.println();

  dht.begin();

  wifiConnectHandler = WiFi.onStationModeGotIP(onWifiConnect);
  wifiDisconnectHandler = WiFi.onStationModeDisconnected(onWifiDisconnect);

  mqttClient.onConnect(onMqttConnect);
  mqttClient.onDisconnect(onMqttDisconnect);
  mqttClient.onPublish(onMqttPublish);
  mqttClient.setServer(MQTT_HOST, MQTT_PORT);

  connectToWifi();
}

void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    hum = dht.readHumidity();
    temp = dht.readTemperature();
  }
}
```

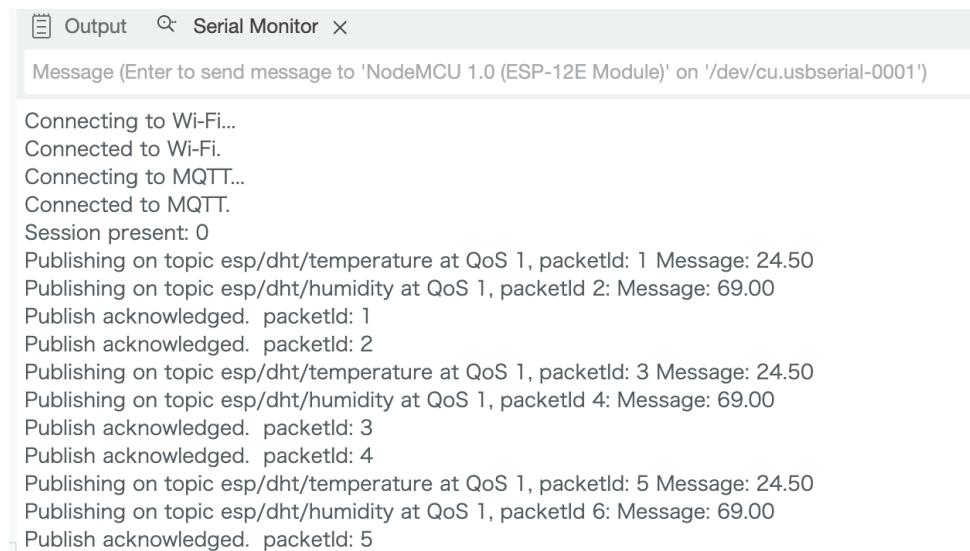
```

// Publish an MQTT message on topic esp/dht/temperature
uint16_t packetIdPub1 = mqttClient.publish(MQTT_PUB_TEMP, 1, true,
                                           String(temp).c_str());
Serial.printf("Publishing on topic %s at QoS 1, packetId: %i ",
              MQTT_PUB_TEMP, packetIdPub1);
Serial.printf("Message: %.2f \n", temp);

// Publish an MQTT message on topic esp/dht/humidity
uint16_t packetIdPub2 = mqttClient.publish(MQTT_PUB_HUM, 1, true,
                                           String(hum).c_str());
Serial.printf("Publishing on topic %s at QoS 1, packetId %i: ",
              MQTT_PUB_HUM, packetIdPub2);
Serial.printf("Message: %.2f \n", hum);
}
}

```

From the **Serial Monitor**, we could observe how the data is being sent to our MQTT broker of choice. To verify, we shall implement another client that subscribes to these topics.



The screenshot shows the Serial Monitor interface with the following text:

```

Output Serial Monitor x
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on '/dev/cu.usbserial-0001')

Connecting to Wi-Fi...
Connected to Wi-Fi.
Connecting to MQTT...
Connected to MQTT.
Session present: 0
Publishing on topic esp/dht/temperature at QoS 1, packetId: 1 Message: 24.50
Publishing on topic esp/dht/humidity at QoS 1, packetId 2: Message: 69.00
Publish acknowledged. packetId: 1
Publish acknowledged. packetId: 2
Publishing on topic esp/dht/temperature at QoS 1, packetId: 3 Message: 24.50
Publishing on topic esp/dht/humidity at QoS 1, packetId 4: Message: 69.00
Publish acknowledged. packetId: 3
Publish acknowledged. packetId: 4
Publishing on topic esp/dht/temperature at QoS 1, packetId: 5 Message: 24.50
Publishing on topic esp/dht/humidity at QoS 1, packetId 6: Message: 69.00
Publish acknowledged. packetId: 5

```

Subscriber

We will write a simple Python script that fetches and prints out data to the terminal. There is the **PAHO MQTT** library that provides great support for the task.

```
import paho.mqtt.client as mqtt
```

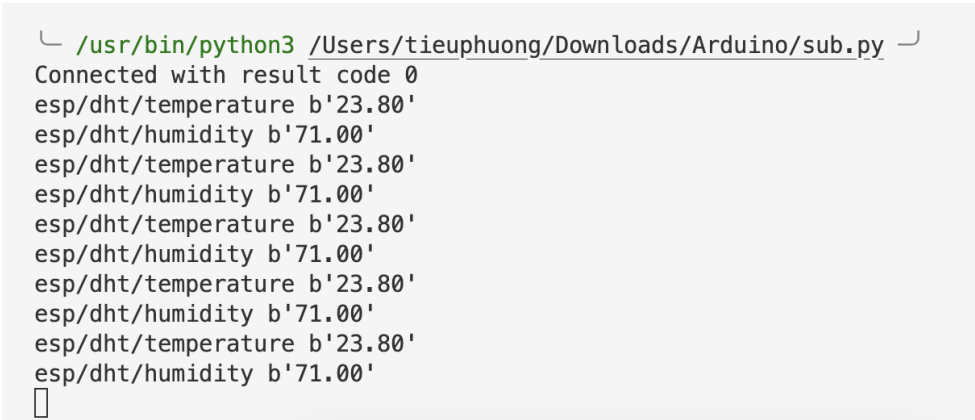
```
# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("esp/dht/temperature")
    client.subscribe("esp/dht/humidity")

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect("broker.hivemq.com", 1883, 60)
client.loop_forever()
```

There should be data fetched from the broker under the topics that we've published to earlier.

A terminal window showing the execution of a Python script. The prompt is `/usr/bin/python3`. The script is `/Users/tieuphuong/Downloads/Arduino/sub.py`. The output shows a successful connection with result code 0, followed by alternating messages for `esp/dht/temperature` (value 23.80) and `esp/dht/humidity` (value 71.00).

```
└─ /usr/bin/python3 /Users/tieuphuong/Downloads/Arduino/sub.py ─┐
Connected with result code 0
esp/dht/temperature b'23.80'
esp/dht/humidity b'71.00'
esp/dht/temperature b'23.80'
esp/dht/humidity b'71.00'
esp/dht/temperature b'23.80'
esp/dht/humidity b'71.00'
esp/dht/temperature b'23.80'
esp/dht/humidity b'71.00'
esp/dht/temperature b'23.80'
esp/dht/humidity b'71.00'
```

Conclusion

In conclusion, this project has successfully achieved its learning objectives by familiarizing us with Arduino programming, sensor interfacing, and IoT principles. The practical implementation of building a circuit to measure humidity and temperature locally, connecting it to the cloud, and remotely retrieving the data has provided valuable hands-on experience in the field of IoT.