Smart Farming Technical Documentation

Team H

06.06.23

1 Team members

- Maxim Emile Speczyk
- Muhammad Umer Bin Yaqoob
- Stephanie Chinenye Okosa

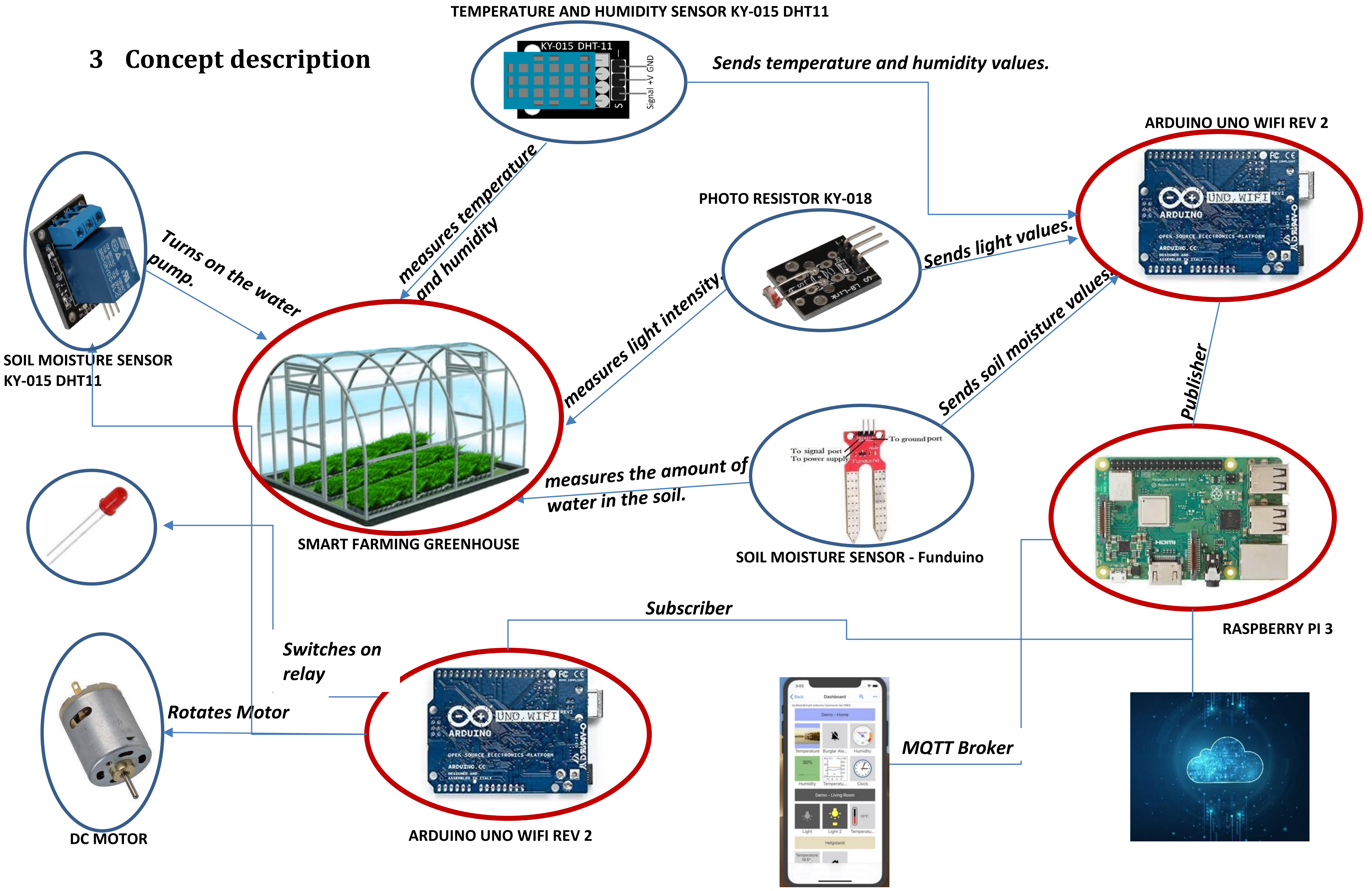
2 Introduction

Internet of things (IOT) refers to *"the worldwide network of interconnected objects uniquely addressable based on standard communication protocols [1]"*. These "objects" may vary based on different characteristics such as size of devices from small e.g., RFID to large devices e.g., cars. Wireless sensors in the context of IOT sensor, connect sensors to the internet via gateways. They are used for collecting data such as temperature, motion etc., and transmitting the sensed data through a centralized communication unit [2]. IOT systems can utilize sensor networks in the domain of agricultural smart farming by communicating with its router to gather data [*Lecture note*].

Smart farming ensures the optimized utilization of farm resources in modernizing agriculture using various technologies such as sensors at the right locations in farm with the sole aim of collecting and processing information to increase farm productivity and reduce manpower [3]. In this documentation, our focus in smart farming would be on the sub-domain of greenhouse monitoring with target applications in terms of soil monitoring, temperature monitoring, and water management.

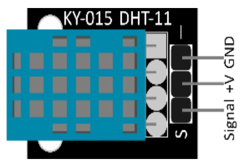
The idea is to create a smart greenhouse. A greenhouse that can regulate its internal environment to optimize plant growth. The greenhouse should maintain optimal temperature and humidity levels, adjust its lighting based on weather conditions and ensure proper irrigation. To achieve this, the Smart Greenhouse will be equipped with a set of sensors and actuators. It will be connected to a central server via MQTT protocol for real-time monitoring and control.

3 Concept description

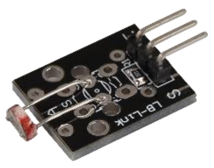


The main application of the smart greenhouse farming prototype is to:

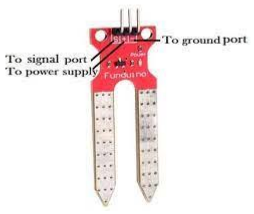
- Measure the amount of water in the soil and turn on the water pump when the water content is below threshold water fraction by volume.
- Measure the amount of light in the Greenhouse. When the light intensity is below minimum value as specified by the operator, the led bulb in the farmhouse is switched on.
- Measures the temperature and humidity of the greenhouse, to ensure that the greenhouse is within room conditions.



This is the KY015 Combi sensor which measure both temperature and humidity of the greenhouse. It is usually suitable for long term measurement as it has low sampling rates.



This is the KY-018 photoresistor. It measures the light intensity of the greenhouse. It contains an LDR resistor whose value decreases with increase in brightness of the greenhouse surroundings.



This is the ME110 soil moisture sensor for measuring the water content in the soil. The two probes of the device when exposed to water, increases conductivity which results in lower resistance increasing its output value.

This is the KY-019 5V relay module switch that normally closed by default. It is opened when the amount of water content in the soil is less than the minimum specified threshold, allowing water flow from the water pump into the soil.

This is a DC motor that propels the opening of the water pump.

4 Project/Team management

We used Scrum which is a subset of agile methodology. Every week we would discuss the project's progress and come up with to-do tasks. In the beginning, the team had planned meetings to decide what to work on during the upcoming week and the idea was discussed. We followed a collective way of solving different tasks in which each of us would work together at every step of the project.

Brainstorming and concept creation: All team members worked together to brainstorm and formulate the initial idea for the smart greenhouse. Each team member contributed their unique perspectives and ideas to create a viable project plan.

Block Diagram, Requirement Diagram, and Class Diagram: All three members contributed to the design and development of the block diagram, requirement diagram, and class diagram. Each member brought their own ideas, and through discussion and iteration, they came up with the final diagrams.

Coding and Implementation: The coding and implementation of the Arduino Uno Wi-Fi, MQTT connection, and other software and hardware components was a shared task. Each team member wrote sections of the code and helped troubleshoot issues. We worked together to establish the MQTT connections and ensure the proper functioning of the hardware.

5 Technologies

The team used various technological approaches to implement the project:
MQTT: The team used MQTT as a messaging protocol due to its low network usage and efficient distribution of data. It helped to maintain a constant communication link between the sensors, the Arduino, Raspberry Pi, and the server.

Arduino IDE: The team used the Arduino IDE to program the Arduino Uno Wi-Fi. The programming language used was C++.

Raspberry Pi OS: The team used the Raspberry Pi OS for the MQTT broker setup on the Raspberry Pi.

MQTTOOL Mobile Application: The team installed the mqtt service application which allows users to monitor and control the greenhouse from anywhere in the network. The app received published sensor data from the server, displays it in a user-friendly format, permits users to subscribe, and send commands to perform an action i.e., switching on the relay and allowing water to be replaced in the greenhouse, when there is low water content in the soil.

Server: To store data and retrieve it when required.

6 Implementation

Greenhouse: This is the physical structure in which the plants are grown. It contains various sensors and actuators.

Arduino Uno Wi-Fi REV 2: This is the microcontroller that interfaces with the sensors and actuators. It reads data from the sensors, controls the actuators based on this data, and communicates with the Raspberry Pi.

Sensors: These are devices that measure various environmental factors in the greenhouse. They include a soil moisture sensor, a temperature and humidity sensor, and a light sensor.

Actuators: These are devices that perform actions based on data from the sensors. They include a water pump for irrigation and servos to control windows and curtains.

Raspberry Pi: This is a small computer that acts as an MQTT broker. It receives data from the Arduino and forwards it to the mobile app.

MQTT Broker: This is a piece of software running on the Raspberry Pi that facilitates communication between the Arduino and the mobile app.

Mobile App: This is a software application that runs on the user's mobile device. It allows the user to monitor the sensor data and control the actuators.

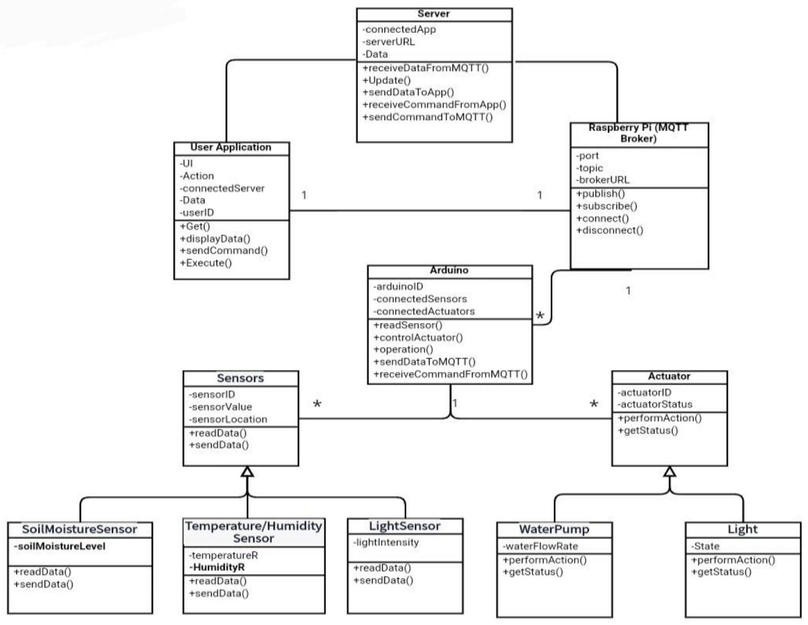


Figure 1: Class Diagram Smart Greenhouse System

.

Sensor classes: Would be used to continually monitor the environmental conditions of the greenhouse such as soil moisture levels, temperature and humidity, and light intensity.

Actuator classes: Would be used to carry out certain actions based on the readings from the sensors. For example, if the soil moisture level is too low, a water pump (an instance of the water pump class) could be activated to water the plants and if the light is dim.

Arduino Class: Would serve as the main controller, reading the data from the sensors, deciding what action to take, and controlling the actuators accordingly.

MQTT Class: Would be used by the Arduino to communicate the sensor readings and any actions taken to a remote server, and to receive commands from the server.

Server Class: Would manage this data, store it for further analysis, and provide a way for the user to view the data and send commands.

APP CLASS: Would be a user-friendly way for the greenhouse owner to monitor the state of the greenhouse, view historical data, and send commands (e.g., manually start the water pump).

There are various use case applications of smart green-house system. However, our focus is on irrigation in the greenhouse where the temperature, light intensity and soil water content are monitored. In cases where any of the above features are not within the proper room condition to yield healthy crops, the actuators are wirelessly activated in order to regulate and adjust the features back to their stable state.

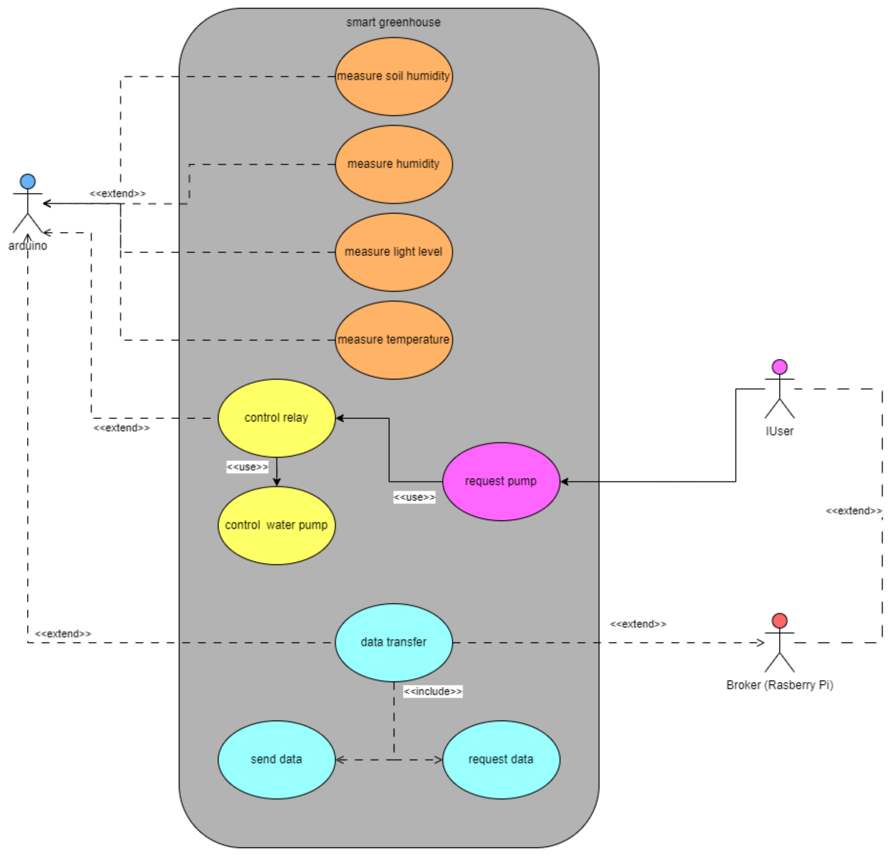
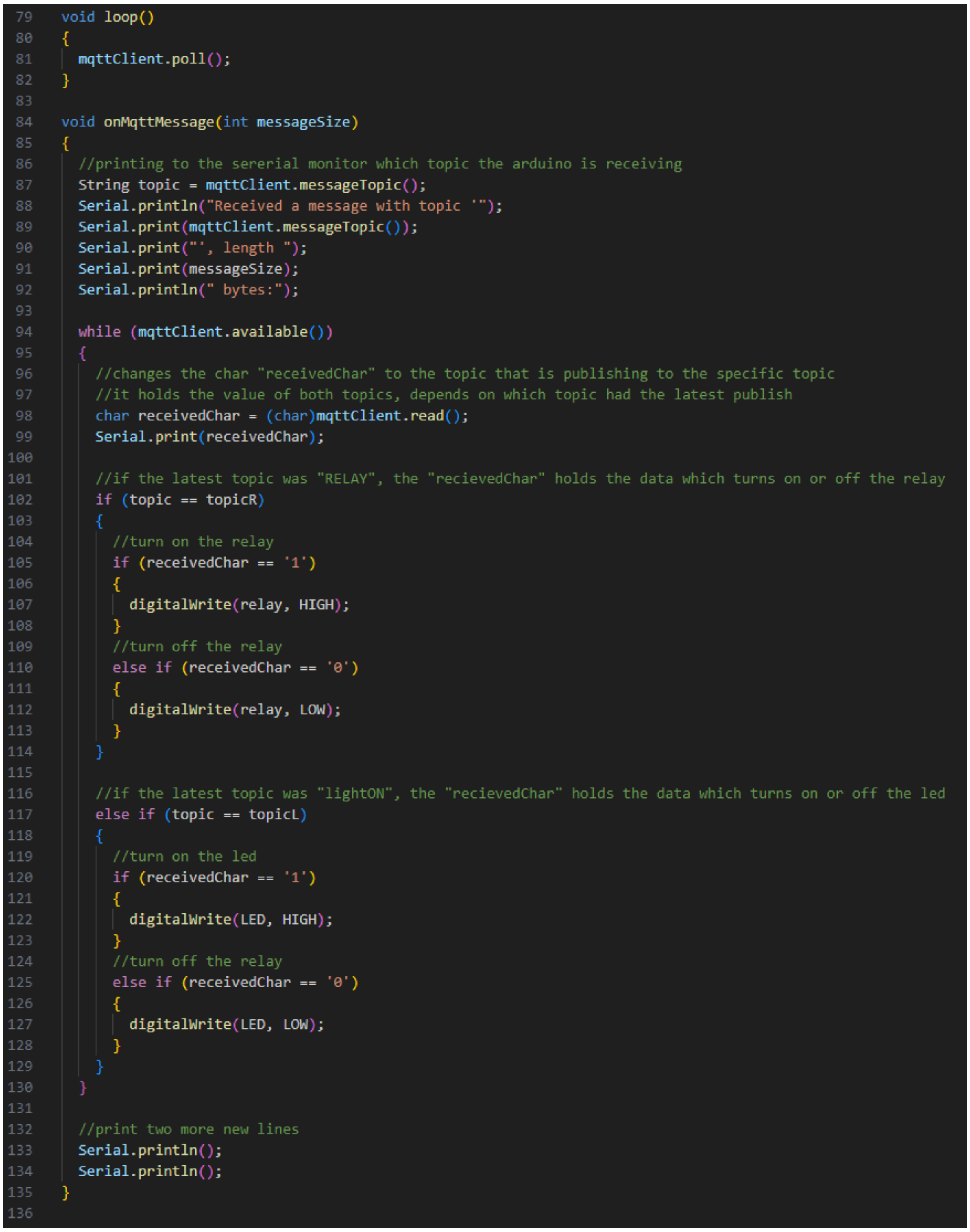


Figure 2: Use Case Diagram Smart Greenhouse System

Below is the implementation procedure of the smart greenhouse system within the above shown use case in Figure 2.

Mqtt subscriber code: For this task the following libraries are used. The first one is the ArduinoMqttClient.h which is responsible for the Mqtt support. The second library is the

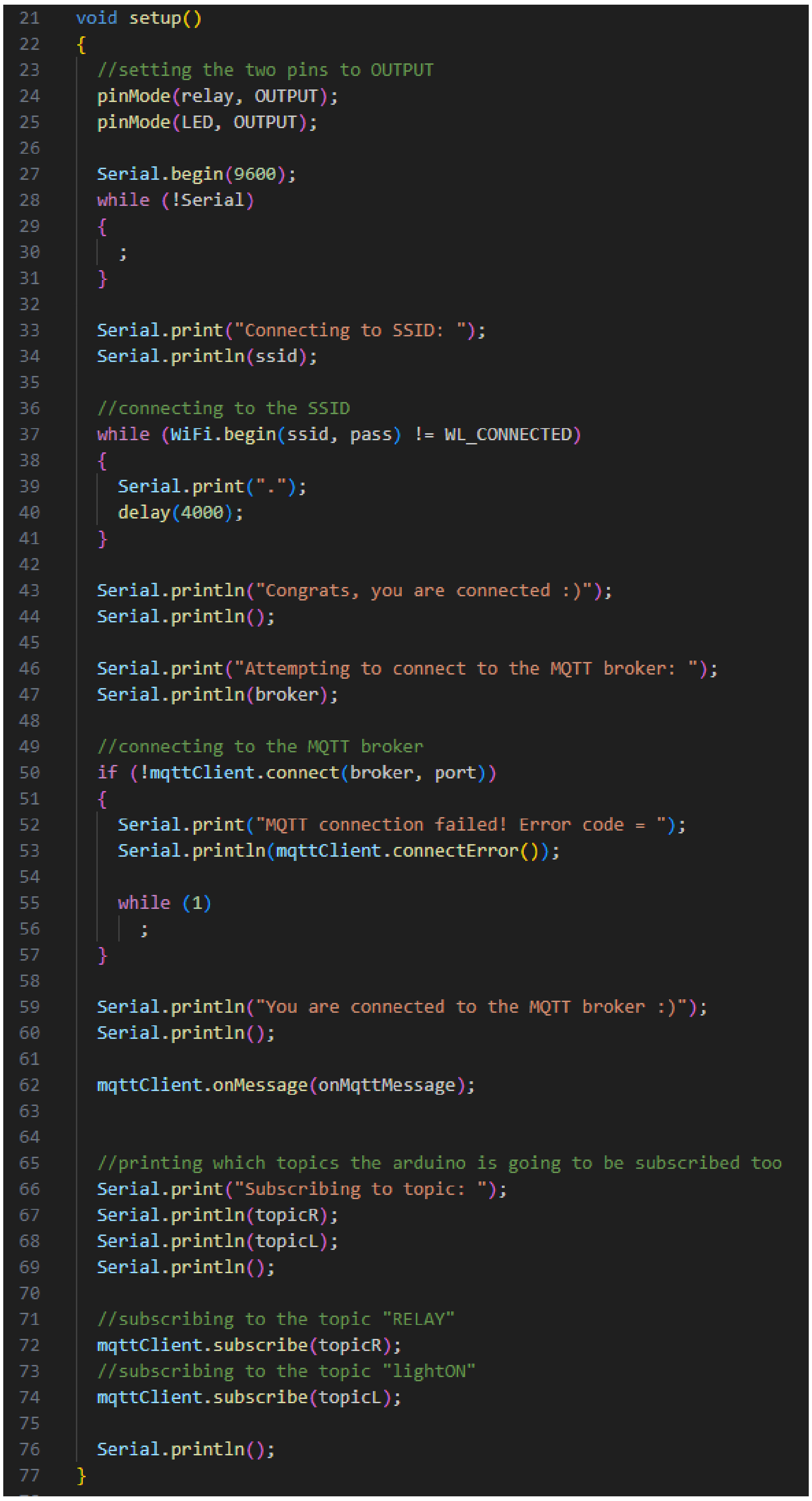
WiFiNINA.h which brings the support for the Wi-Fi module, which is implemented in the Arduino. Below is the Arduino code for the Mqtt subscriber which controls the relay, which is attached to the water pump, and the led. The relay will open when the topic “RELAY” is set to 1, as well as close when it is set to 0. The same process happens for the led, the main difference is that this is controlled by the topic “lightON”.



```
void loop()
{
  mqttClient.poll();
}

void onMqttMessage(int messageSize)
{
  //printing to the sererial monitor which topic the arduino is receiving
  String topic = mqttClient.messageTopic();
  Serial.println("Received a message with topic '");
  Serial.print(mqttClient.messageTopic());
  Serial.print("', length ");
  Serial.print(messageSize);
  Serial.println(" bytes:");

  while (mqttClient.available())
  {
    //changes the char "receivedChar" to the topic that is publishing to the specific topic
    //it holds the value of both topics, depends on which topic had the latest publish
    char receivedChar = (char)mqttClient.read();
    Serial.print(receivedChar);

    //if the latest topic was "RELAY", the "recievedChar" holds the data which turns on or off the relay
    if (topic == topicR)
    {
      //turn on the relay
      if (receivedChar == '1')
      {
        digitalWrite(relay, HIGH);
      }
      //turn off the relay
      else if (receivedChar == '0')
      {
        digitalWrite(relay, LOW);
      }
    }

    //if the latest topic was "lightON", the "recievedChar" holds the data which turns on or off the led
    else if (topic == topicL)
    {
      //turn on the led
      if (receivedChar == '1')
      {
        digitalWrite(LED, HIGH);
      }
      //turn off the relay
      else if (receivedChar == '0')
      {
        digitalWrite(LED, LOW);
      }
    }
  }

  //print two more new lines
  Serial.println();
  Serial.println();
}
```

Figure 3: Relay and led controlling code

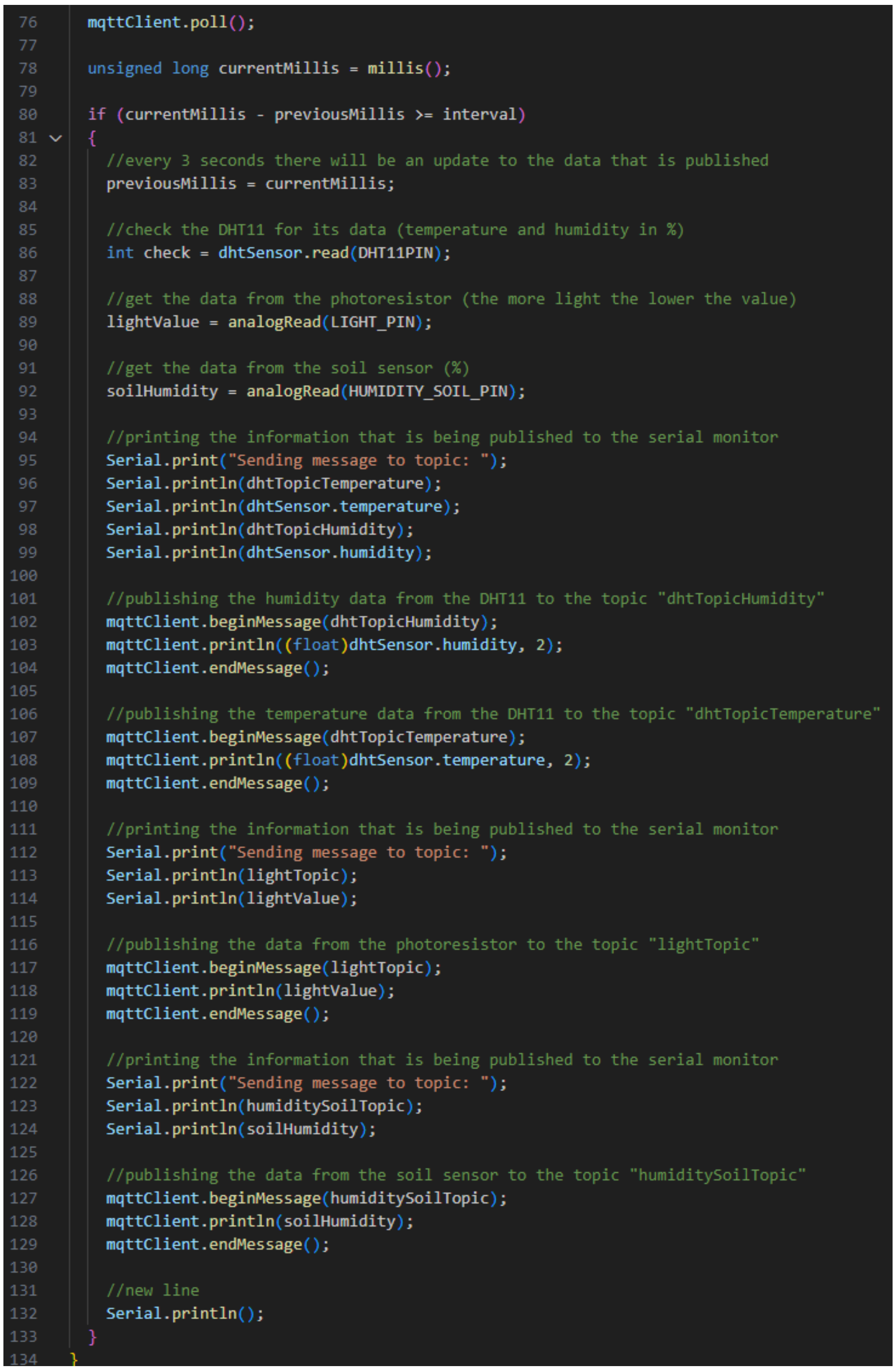
Below is the code for connecting the Arduino to the mqtt broker (Raspberry Pi), to the SSID, as well as subscribing to the topics “RELAY” and “LightON”.



```
void setup()
{
  //setting the two pins to OUTPUT
  pinMode(relay, OUTPUT);
  pinMode(LED, OUTPUT);

  Serial.begin(9600);
  while (!Serial)
  {
    ;
  }

  Serial.print("Connecting to SSID: ");
  Serial.println(ssid);

  //connecting to the SSID
  while (WiFi.begin(ssid, pass) != WL_CONNECTED)
  {
    Serial.print(".");
    delay(4000);
  }

  Serial.println("Congrats, you are connected :)");
  Serial.println();

  Serial.print("Attempting to connect to the MQTT broker: ");
  Serial.println(broker);

  //connecting to the MQTT broker
  if (!mqttClient.connect(broker, port))
  {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());

    while (1)
      ;
  }

  Serial.println("You are connected to the MQTT broker :)");
  Serial.println();

  mqttClient.onMessage(onMqttMessage);


  //printing which topics the arduino is going to be subscribed too
  Serial.print("Subscribing to topic: ");
  Serial.println(topicR);
  Serial.println(topicL);
  Serial.println();

  //subscribing to the topic "RELAY"
  mqttClient.subscribe(topicR);
  //subscribing to the topic "lightON"
  mqttClient.subscribe(topicL);

  Serial.println();
}
```

Figure 4: Connecting to the SSID, broker and subscribing to topics

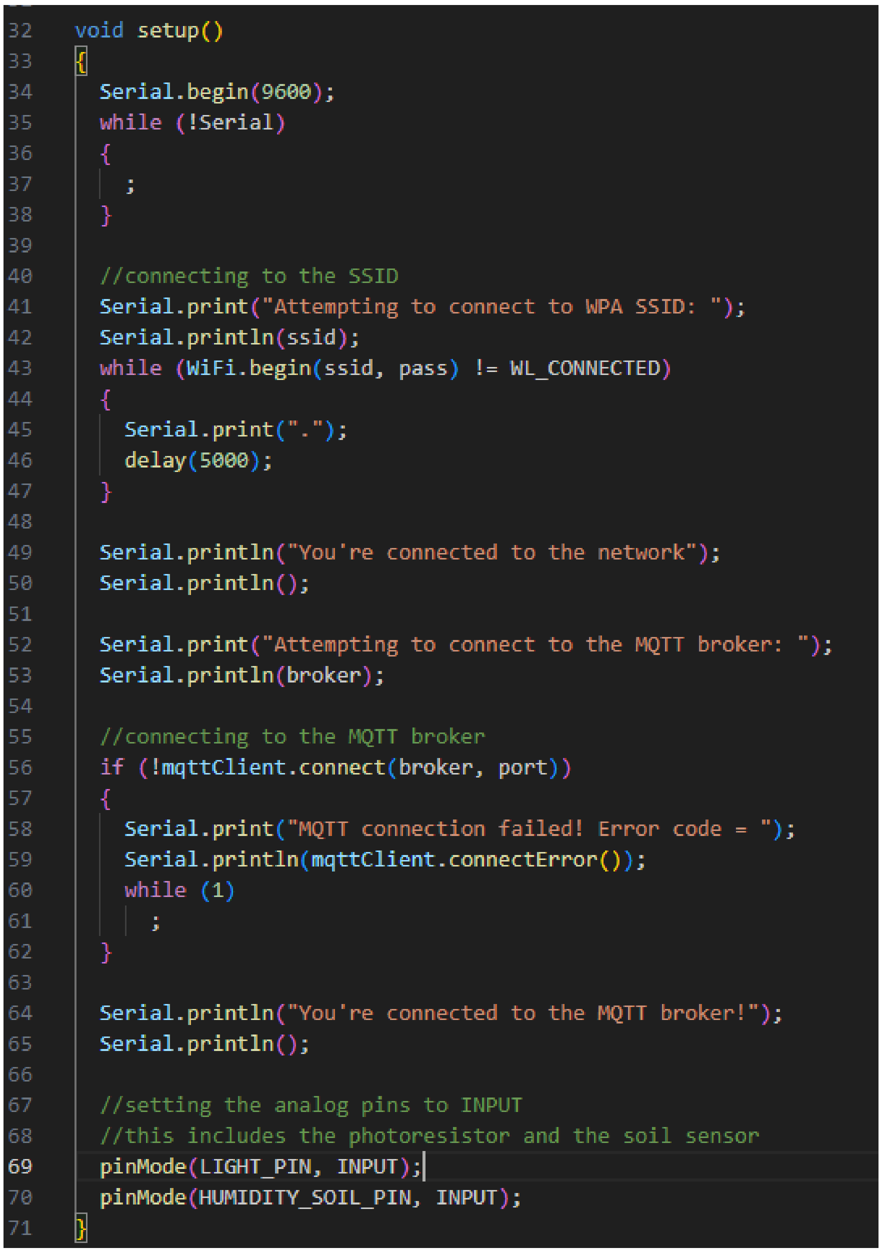
Mqtt publisher code: For this task the same libraries are in use as in the Mqtt subscriber code. Additionally, the library dht11.h is used for the DHT11 sensor which is responsible for collecting the data of the air temperature and humidity. Below is the Arduino code for the Mqtt publisher which is responsible for collecting the sensor data and publishing it to the corresponding topics. The topics are the following. The topic “dhtTopicHumidity” includes the humidity of the air. The topic “dhtTopicTemperature” contains the temperature of the air. The topic “lightTopic” includes the data of the photoresistor and the topic “humiditySoilTopic” contains the humidity of the soil.



```
  mqttClient.poll();

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval)
  {
    //every 3 seconds there will be an update to the data that is published
    previousMillis = currentMillis;

    //check the DHT11 for its data (temperature and humidity in %)
    int check = dhtSensor.read(DHT11PIN);

    //get the data from the photoresistor (the more light the lower the value)
    lightValue = analogRead(LIGHT_PIN);

    //get the data from the soil sensor (%)
    soilHumidity = analogRead(HUMIDITY_SOIL_PIN);

    //printing the information that is being published to the serial monitor
    Serial.print("Sending message to topic: ");
    Serial.println(dhtTopicTemperature);
    Serial.println(dhtSensor.temperature);
    Serial.println(dhtTopicHumidity);
    Serial.println(dhtSensor.humidity);

    //publishing the humidity data from the DHT11 to the topic "dhtTopicHumidity"
    mqttClient.beginMessage(dhtTopicHumidity);
    mqttClient.println((float)dhtSensor.humidity, 2);
    mqttClient.endMessage();

    //publishing the temperature data from the DHT11 to the topic "dhtTopicTemperature"
    mqttClient.beginMessage(dhtTopicTemperature);
    mqttClient.println((float)dhtSensor.temperature, 2);
    mqttClient.endMessage();

    //printing the information that is being published to the serial monitor
    Serial.print("Sending message to topic: ");
    Serial.println(lightTopic);
    Serial.println(lightValue);

    //publishing the data from the photoresistor to the topic "lightTopic"
    mqttClient.beginMessage(lightTopic);
    mqttClient.println(lightValue);
    mqttClient.endMessage();

    //printing the information that is being published to the serial monitor
    Serial.print("Sending message to topic: ");
    Serial.println(humiditySoilTopic);
    Serial.println(soilHumidity);

    //publishing the data from the soil sensor to the topic "humiditySoilTopic"
    mqttClient.beginMessage(humiditySoilTopic);
    mqttClient.println(soilHumidity);
    mqttClient.endMessage();

    //new line
    Serial.println();
  }
}
```

Figure 5: Publishing the sensor data

Below is the code for connecting the Arduino to the Mqtt broker, as well as to the SSID.



```
void setup()
{
  Serial.begin(9600);
  while (!Serial)
  {
    ;
  }

  //connecting to the SSID
  Serial.print("Attempting to connect to WPA SSID: ");
  Serial.println(ssid);
  while (WiFi.begin(ssid, pass) != WL_CONNECTED)
  {
    Serial.print(".");
    delay(5000);
  }

  Serial.println("You're connected to the network");
  Serial.println();

  Serial.print("Attempting to connect to the MQTT broker: ");
  Serial.println(broker);

  //connecting to the MQTT broker
  if (!mqttClient.connect(broker, port))
  {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());
    while (1)
      ;
  }

  Serial.println("You're connected to the MQTT broker!");
  Serial.println();

  //setting the analog pins to INPUT
  //this includes the photoresistor and the soil sensor
  pinMode(LIGHT_PIN, INPUT);
  pinMode(HUMIDITY_SOIL_PIN, INPUT);
}
```

Figure 6: Connecting to the SSID and broker (publisher)

7 Sources/References

[1] Tushar Semwal, Faiz Iqbal (2022): Cyber-Physical Systems (9781000562644). Available online at https://learning.oreilly.com/library/view/cyber-physicalsystems/9781000562644/., checked on 06.06.23.

[2] Khalil, Nacer; Abid, Mohamed Riduan; Benhaddou, Driss; Gerndt, Michael (2014): Wireless Sensors Networks for Internet of Things, checked on 06.06.23.

[3] Govind Singh Patel, Amrita Rai, Nripendra Narayan Das, R.P. Singh (2021): Smart Agriculture: CRC Press. Available online at https://learning.oreilly.com/library/view/smart-agriculture/9781000327892/xhtml/ch07.xhtml, checked on 06.06.23.