Documentation

Ghost of Uchiha

24.06.2022

Team members

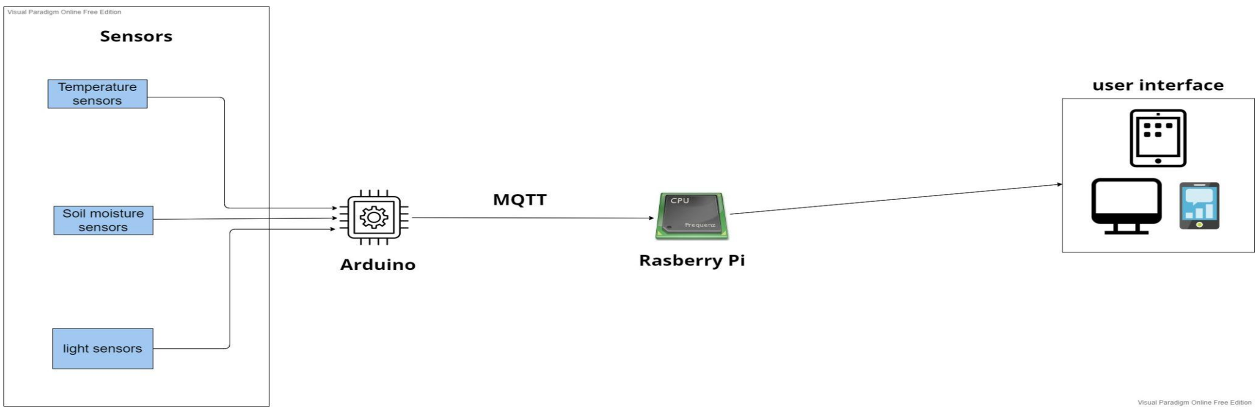
1. Jires voufo Donfack
2. Amit Chakma
3. Evrard Leuteu
4. Aditya Kumar

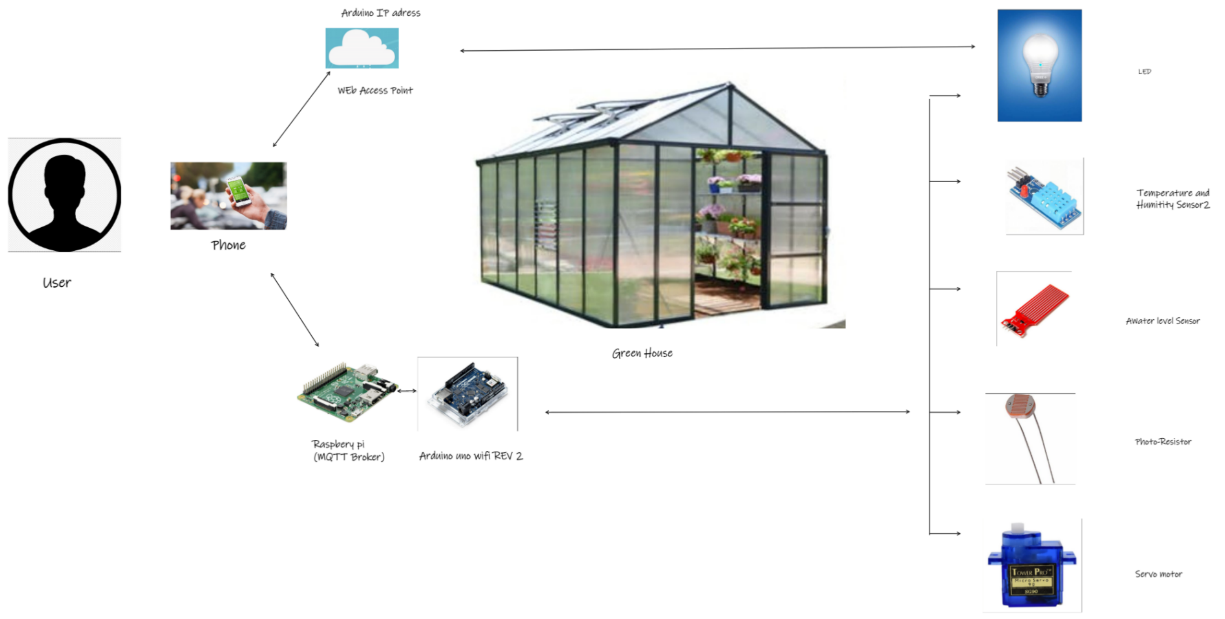
Introduction

This semester, we wanted to build a greenhouse for our project. The goal is to collect real-time data so that the system may be controlled properly. We will use the loT-WSN architecture to do this. The sensor layer, control layer, backhaul layer, application layer, and decision layer are all feasible architectural layers. The sensors that interact with the environment to acquire data are part of the sensing layer. These data are subsequently calculated and wirelessly sent to the control layer nodes. These control layers would be in charge of transmitting data to our system's back end and controlling the activities of our actuators. The backhaul merely acts as a link between these tiers. We will use a synchronous Message Queuing Telemetry Transport (MQTT) communication protocol for our project, which will be explored in further detail in the following sections. The application layer specifies the functions of our application and is in charge of the correct integration of data from these various sensors. Finally, the decision layer analyzes the data from these devices and regulates our system's overall functional need domain. These are simple to combine with IoT devices, such as a smartphone in our example. The overall implementation is covered later in the article.

Concept description

Block diagram





The main application for our prototype

A Greenhouse is constructed and used to grow a variety of plants such as vegetables, fruits and flowers. They serve as a controlled and protected environment to grow healthier plants. Our prototype helps to make the process of using and maintaining these greenhouses with ease and efficiency by creating an IOT application around it. In our prototype, we will be using sensors to collect data such as soil moisture, temperature and light and use them to make decisions to perform activities such as irrigation and ventilation in the greenhouse. All this could be monitored using an application that would be accessible through smartphones, computers etc.

Devices, sensors, actuators, apps etc. used for our application

In this project we will be using the following Hardware and Software:

Control Unit

- Arduino Uno Wi-Fi(R2) to collect data coming from the sensors
- Raspberry Pi 3 takes data collected by Arduino and sends it via Wi-Fi to the internet
- A Smartphone to monitor the data from the internet

Sensors

- Temperature and Humidity combine to measure those two parameters from the environment
- Moisture sensor to measure the level of moisturization of the soil

Actuator

- A 3.6 V motor to pump water and irrigate the soil 4. Software

Arduino IDE to program the Arduino
Raspbian to program the Raspberry Pi

Project/Team management

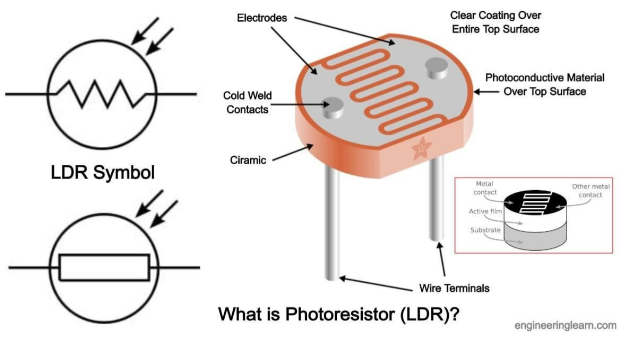
We distributed the task evenly among team members and we will be using the agile method for project management and would follow the double diamond approach for this project. The tasks were evenly divided between the group members. Following is a list of tasks performed by each member :

- **Aditya Kumar:** Wrote the code for the DTH11 Temperature/Moisture sensor as well as the project implementation and main application section of the documentation.
- **Amit Chakma:** wrote the code for the water level sensor as well as the block diagram and cases 1 and 2 in the Use Case section of the documentation.
- **Evrard Leuteu:** wrote the code for Servo motor and Mqtt implementation as well as the case 3 in the Use Case section of the documentation.
- **Jires voufo Donfack:** Wrote code for Light sensor and LED as well as the Technologie section of the documentation.

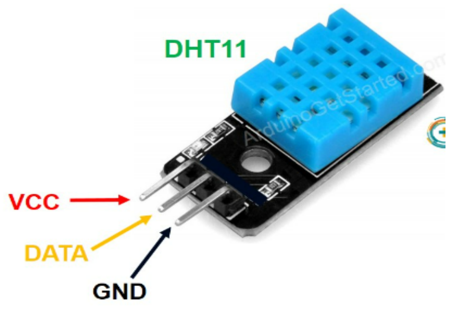
Technologies

Sensors

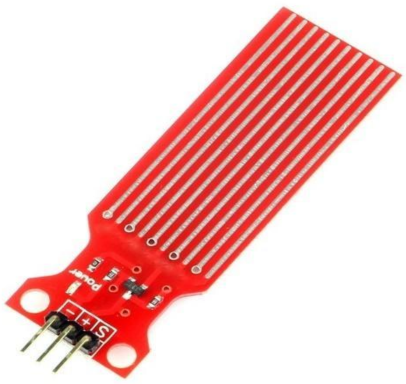
Photo-resistor: This component has a resistivity that varies depending on the amount of light that it perceives. When in the dark the resistance of this sensor gets quite high. Our sensor used in this project has values from 1KΩ under the light up to 10KΩ in the dark. It is made using a material. It works in a way that when photons come in contact with the top part of the sensor, electrons start travelling through the electrodes. Because this sensor only has resistance as output and Arduino takes voltage input, we use the voltage divider rule to connect it to our board. They can be used in devices like alarm clocks and streetlight.[1]



Temperature/Humidity sensor: There is a built-in NTC Thermistor and a capacitive sensor for respective temperature and humidity measurements .it has a measuring range of 0 to 50°C for temperature and 20 to 90% RH. This sensor has a digital output [2].



Water level sensor: It has 10 parallel conductors that act like a potentiometer with resistance changes based on the level of water. The higher the level of water, the lower the resistance. Based on the value of the resistance a certain output voltage is sent. It can be used to check leakage of water, rainfall, or maybe if a tank is overflowing [4].

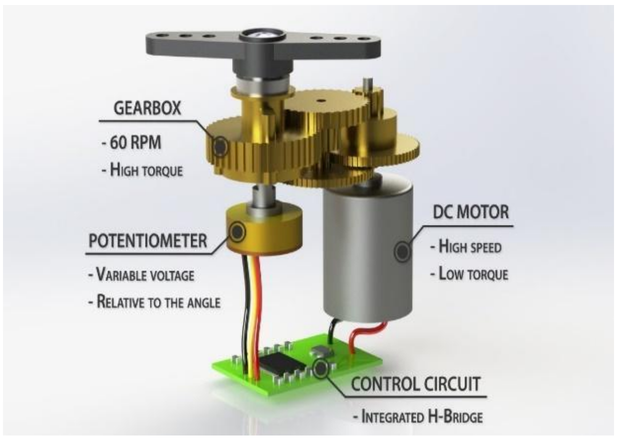


Actuators

LED: is used in this situation to showcase a situation whereby when it gets dark then the light turns ON automatically

Resistors: we used one 220-ohm (use to limit the amount of current flowing) resistor for the LED and one 10K -ohm for the photo-resistor

Servomotor (SG90): It can turn up to 90 degrees in both directions. The controlling of the position happens via PWM (pulse width modulation) with a frequency of 50Hz and a period of 20ms. It can be used in situations where position control is required. The main components of a servomotor are a potentiometer, gearbox, DC motor and a control circuit [5].



Servomotor Features	
Torque	1-5kg/cm
Voltage	4.8V
Operating Temperature	0 to 55°C
Weight	9g
Speed	0.3 sec/60 Grad

Communication Protocols

MQTT: message queuing telemetry transport is a lightweight publisher/subscriber messaging protocol designed for machine-to-machine telemetry in low bandwidth environments. It is designed for the internet of things. Some of its advantages are quick to implement, fast and efficient message delivery, and reduction in network bandwidth usage. Some of the issues that it faces are related to safety [3]

WI-FI: It is a protocol that makes it possible for devices like cell phones and many other devices to connect and exchange information via order with the internet. This is done through a wireless router

Software

Arduino was used to writing the code for sensors and actuators. The Command-line in Raspberry Terminal was used for MQTT. Raspbian: an operating system for the raspberry. VNC: was used during the set-up of the Raspberry

Programming Languages

C-language was used on the Arduino side to write the code for all sensors and the stepper motor.

Controller Units

Arduino Uno Wi-Fi(R2)

This component is WIFI capable. It acts as a sensor node to collect all data from all our sensors and send it via WIFI to the host which is Raspberry. It also drives our output which is the servo motor.

Raspberry Pi 3 Model B

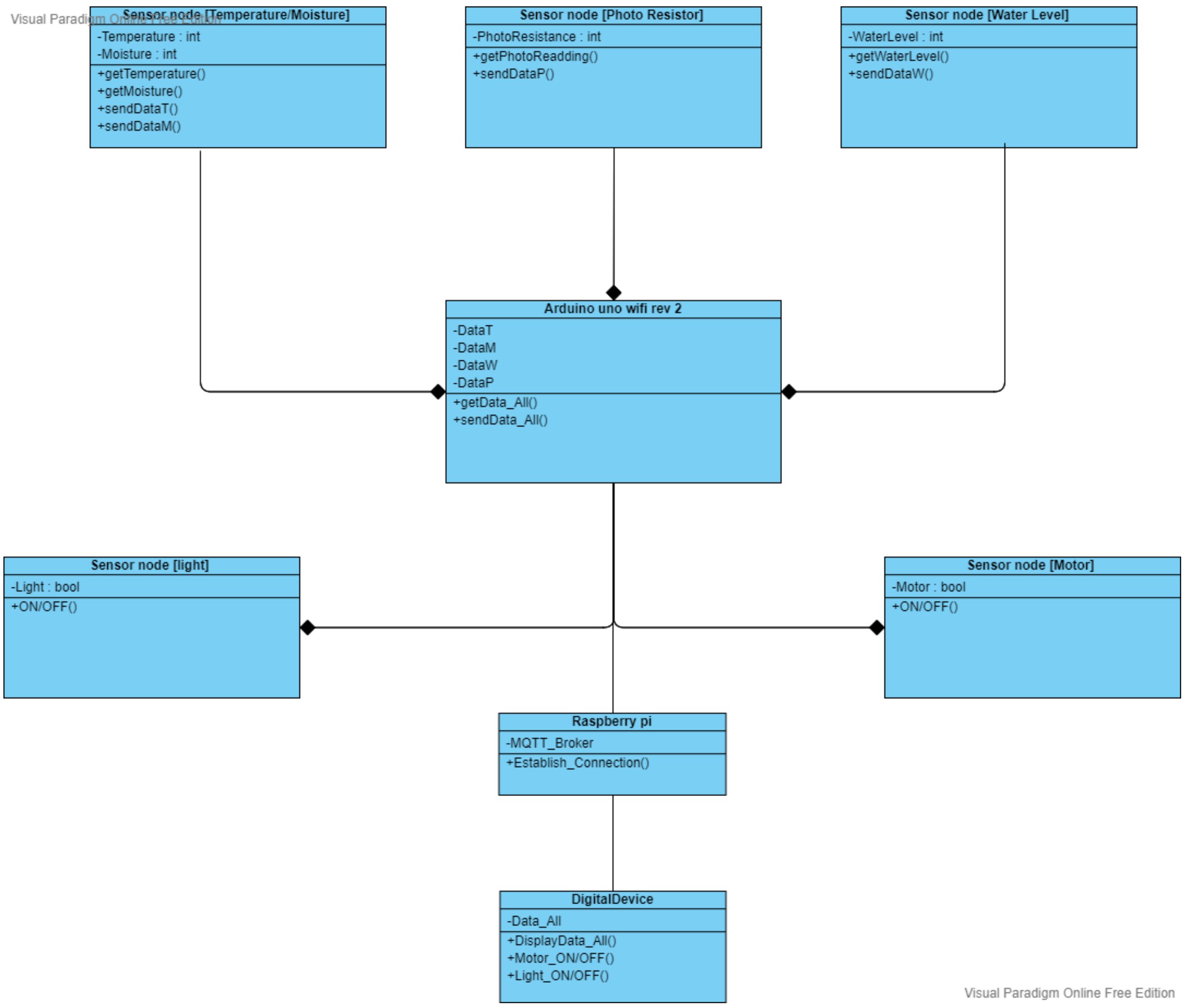
Is the host computer and must communicate with the Arduino to collect the data and store it. The data is then published to the web server via the MQTT protocol.

Implementation

To describe the implementation of the project, first, we will explain the structure of our project. It is constructed around 4 key elements :

- **Sensor node [Publisher]:** A node that consists of a Sensor such as a humidity sensor that is connected to an Arduino wifi rev 2. This node sends information to the server.
- **Sensor node [Subscriber]:** A node that consists of a Sensor such as a Motor sensor that is connected to an Arduino wifi rev 2. This node receives information for the Server.
- **Raspberry PI [MQTT broker]:** A raspberry pi is used as the MQTT broker that serves the connection between the publisher and the subscriber.
- **Digital device [Subscriber/Publisher]:** A digital device such as a mobile phone or a laptop that accesses the web services. This device would be used to receive the information from the Sensor nodes and will let the user perform some action based on that information.

The following is the class diagram for our Implementation.



We started our implementation by working on the Sensor nodes. The team decided that it would be best to work on an individual sensor node that could publish the data and then integrate the other nodes into our project.

We started with installing an Arduino IDE and set up the Arduino wifi rev 2 modules. Once the setup was complete we used the DHT11 sensor to send data to a web server. we used several libraries such as WiFiNINA, SPI, and DHT. We also made "`Arduino_secrets.h`" that would store the SSID and PASS for the network, thus making it more modular. At this point of development, we were not using the Raspberry pi. The implemented code can be found in the appendix.

The next step of our project was to set up the Rasberry pi as the MQTT broker. For this, we started with installing a VNC viewer. VNC is a free remote desktop application that helps you access a remote machine (in our case Raspberry pi) and control it with the existing hardware of your system. Once we set up our environment, we started with our development process by installing MOSQUITTO Broker on the Raspberry pi to support the MQTT protocol. Mosquitto is a message broker. Once this setup is complete, one can use the following line of code to Publish or Subscribe. To publish : “`mosquito -d mosquitto_pub -d -t testTopic -m`” and to subscribe : “`Mosquitto_sub -d -t testTopic -m`”.

It is to be noted that we used our wifi hotspot to host the server for our project.
Our application could potentially use all digital platforms that have access to web services but we as a team decided to go with a phone application to use it as a publisher/subscriber. We are using test.mosquitto.org Which hosts a publicly available Eclipse Mosquitto MQTT server/broker. With this, we configured our application such that it could receive the data from the publisher nodes and send the data to the subscriber nodes. The implementation of this as well as some pictures of our hardware simulation can be found in the appendix.

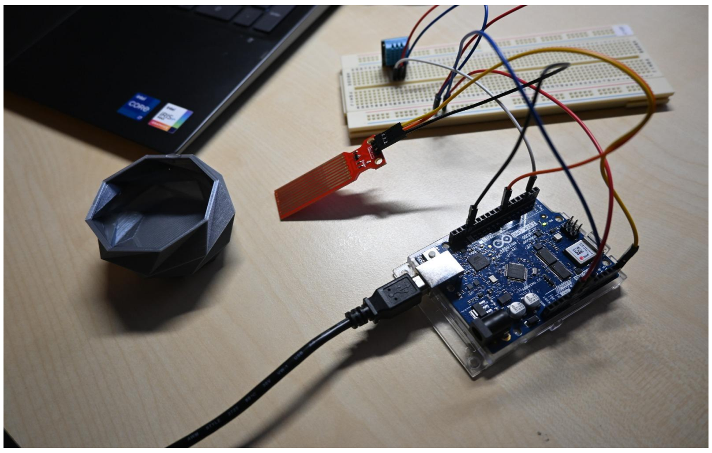
We then proceeded to set up the subscriber node. We Initially decide to start with setting up the Motor sensor node to show the implementation but due to a lack of compatible hardware and time, we decided to use a LED that could be turned ON/OFF representing the lighting system of our greenhouse prototype. With this, we completed our project implementation. The following section describes the different use cases of our project.

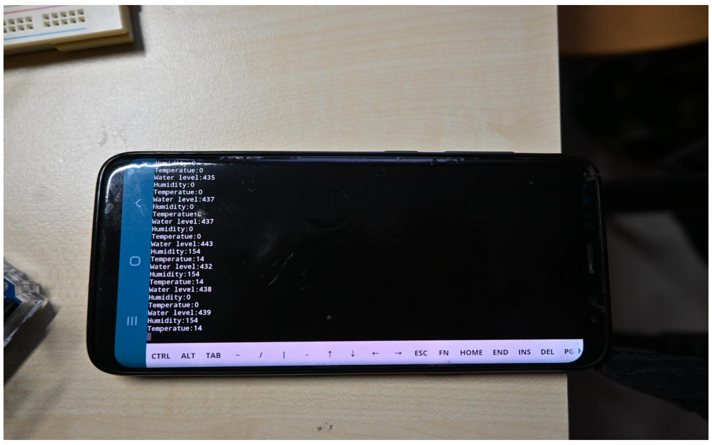
Use Case

Scenario 1

It is very vital to check the system daily. Maintaining a large greenhouse can be challenging. We use some of the sensors for better visibility as an owner. Here the user can be seen as a subscriber and the sensor a publisher for the MQTT broker. We used a water level sensor, and temperature and humidity sensor which will be placed inside the greenhouse. The water level sensor gives data about the water level of the container. The user can check the amount of water and refill the container for watering the plants. Temperature and humidity will send updated data to the users.

The wiring and actual output can be seen in the Screenshots below.

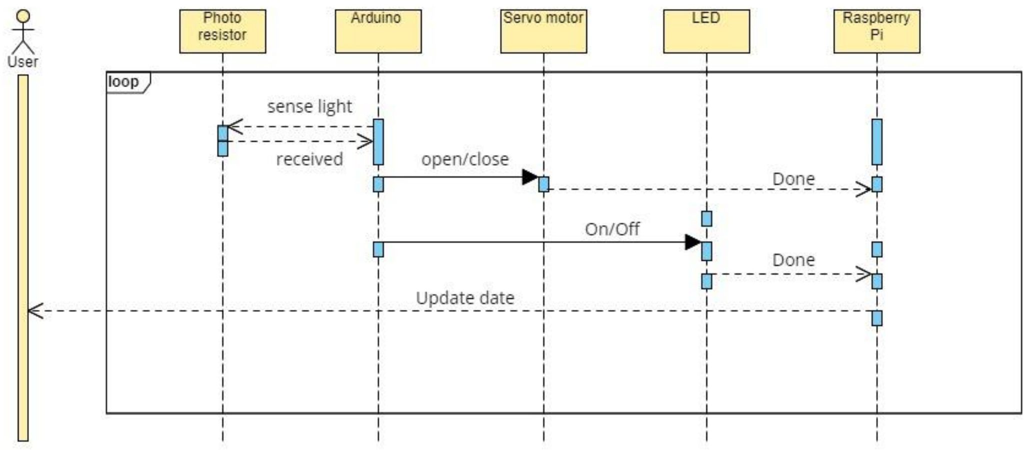




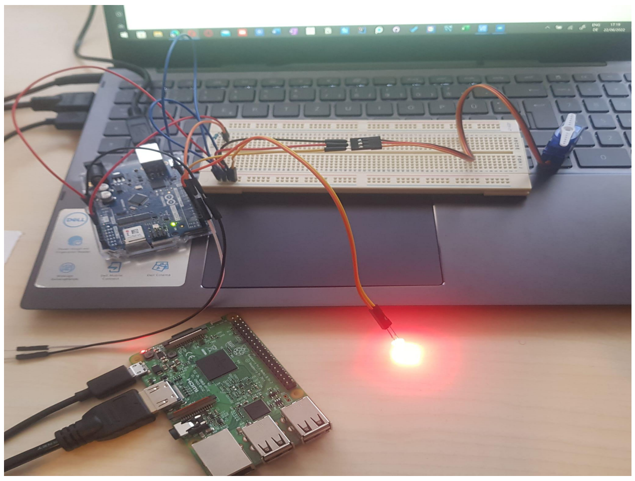
```
Humidity:0
Temperatue:0
Water level:435
Humidity:0
Temperatue:0
Water level:437
Humidity:0
Temperatue:0
Water level:437
Humidity:0
Temperatue:0
Water level:443
Humidity:154
Temperatue:14
Water level:432
Humidity:154
Temperatue:14
Water level:438
Humidity:0
Temperatue:0
Water level:439
Humidity:154
Temperatue:14
```

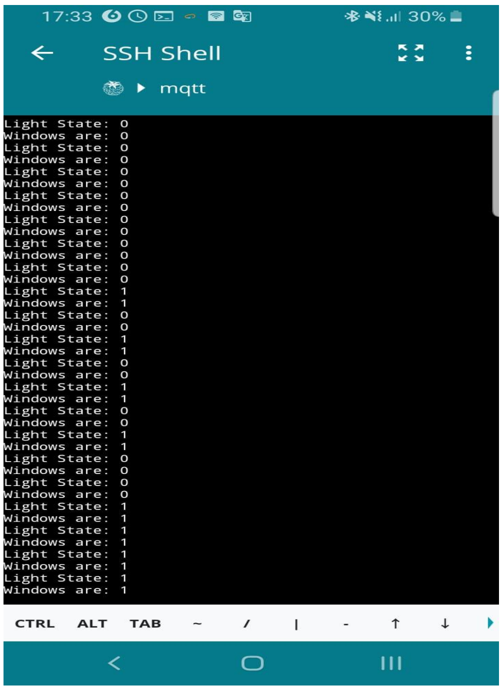
Scenario 2

In summer it is better to keep our windows open because trees do not need a greenhouse summer and alternatively in winter depending on the amount of sunlight the windows will remain open. In this scenario, the photoresistors sense the sunlight in the morning which will open the window using a servo motor and also turn the LED On. The user can see the LED and know if the windows are open or closed. So the window will be closed when the sun sets. As we know, the temperature falls at night compared to daytime.



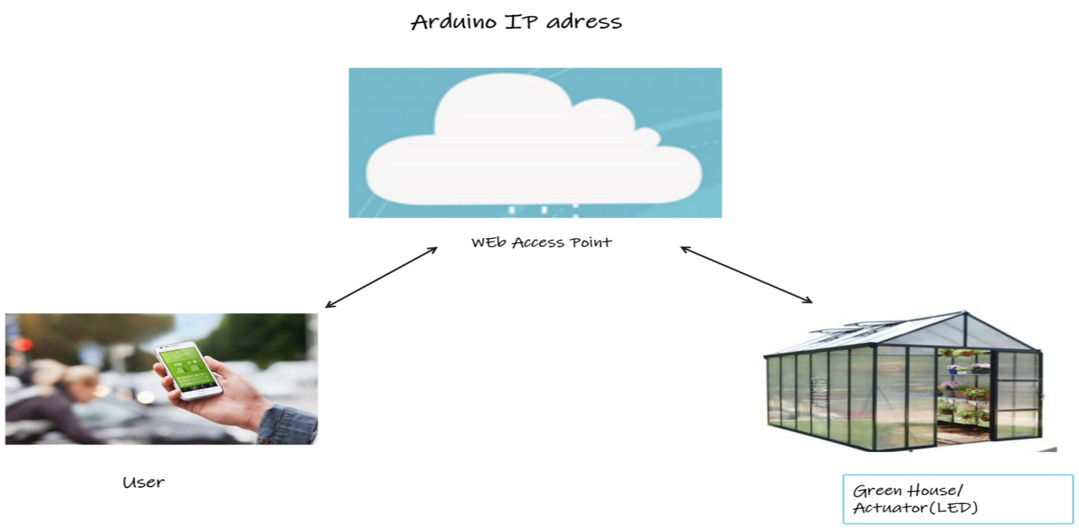
The wiring and actual output can be seen in the Screenshots below.



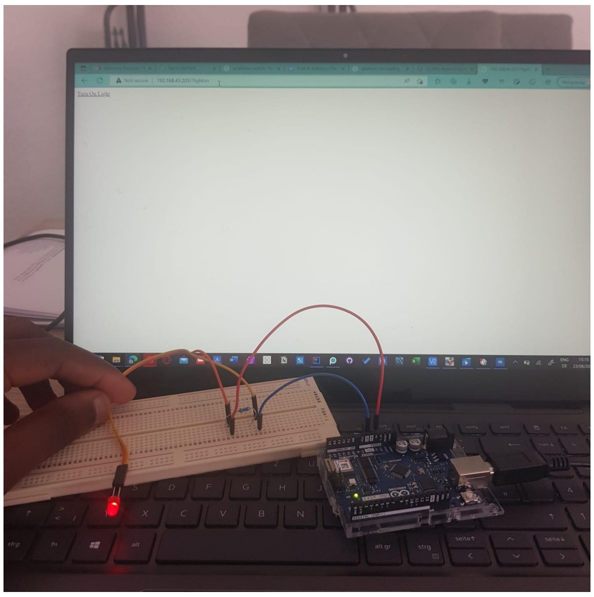


```
Light State: 0
Windows are: 0
Light State: 0
Windows are: 0
Light State: 0
Windows are: 0
Light State: 0
Windows are: 0
Light State: 0
Windows are: 0
Light State: 0
Windows are: 0
Light State: 0
Windows are: 0
Light State: 1
Windows are: 1
Light State: 0
Windows are: 0
Light State: 1
Windows are: 1
Light State: 0
Windows are: 0
Light State: 1
Windows are: 1
Light State: 0
Windows are: 0
Light State: 1
Windows are: 1
Light State: 0
Windows are: 0
Light State: 0
Windows are: 0
Light State: 1
Windows are: 1
Light State: 1
Windows are: 1
Light State: 1
Windows are: 1
Light State: 1
Windows are: 1
```

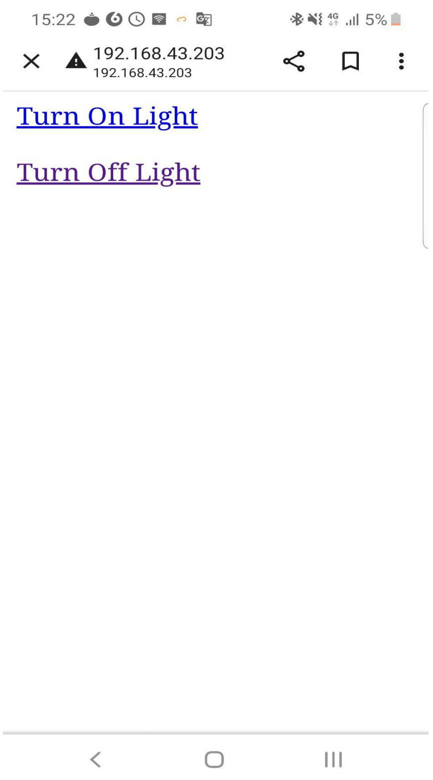
Scenario 3



This scenario describes the interaction between the user and an actuator(LED). It is not only useful to have access to the sensor's details from the phone but also, it is quite important to have a certain degree of control. The control of the LED by the user can be seen from the diagram below whereby the user can turn on and off the LED through an access point(Arduino IP Adress), in this case through a web app. Screenshots of the web app user interface were provided below as the user can turn ON and OFF the LED from his phone.



The wiring and actual output can be seen in the Screenshots below.


15:22
5%
192.168.43.203
192.168.43.203
Turn On Light
Turn Off Light

Sources/References

link to our GitHub:
https://github.com/aditya-kumar23/Advance_Embedded/tree/main/ES_Lab

[1] How to Use a Photoresistor - Arduino Project Hub

[2] In-Depth: Interface DHT11 Module With Arduino (lastminuteengineers.com)

[3] https://mqtt.org/

[4]In-Depth: How Water Level Sensor Works and Interface it with Arduino - Last Minute Engineers

[5] How to Control Servo Motors with Arduino - Complete Guide (howtomechatronics.com)

References Images:

user - Bing images

photo resistor sensor - Bing

servo motorsensor - Bing

water level sensor - Bing

ledtemperature sensor - Bing

led - Bing