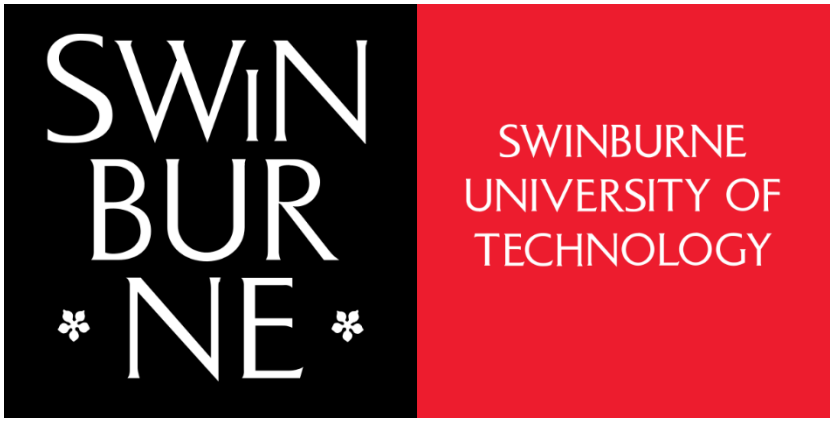


SWE30011- IoT Programming

SMART FARMING SYSTEM

Pham Le Yen Chi - 103430040

Tran Duc Anh - 103565751

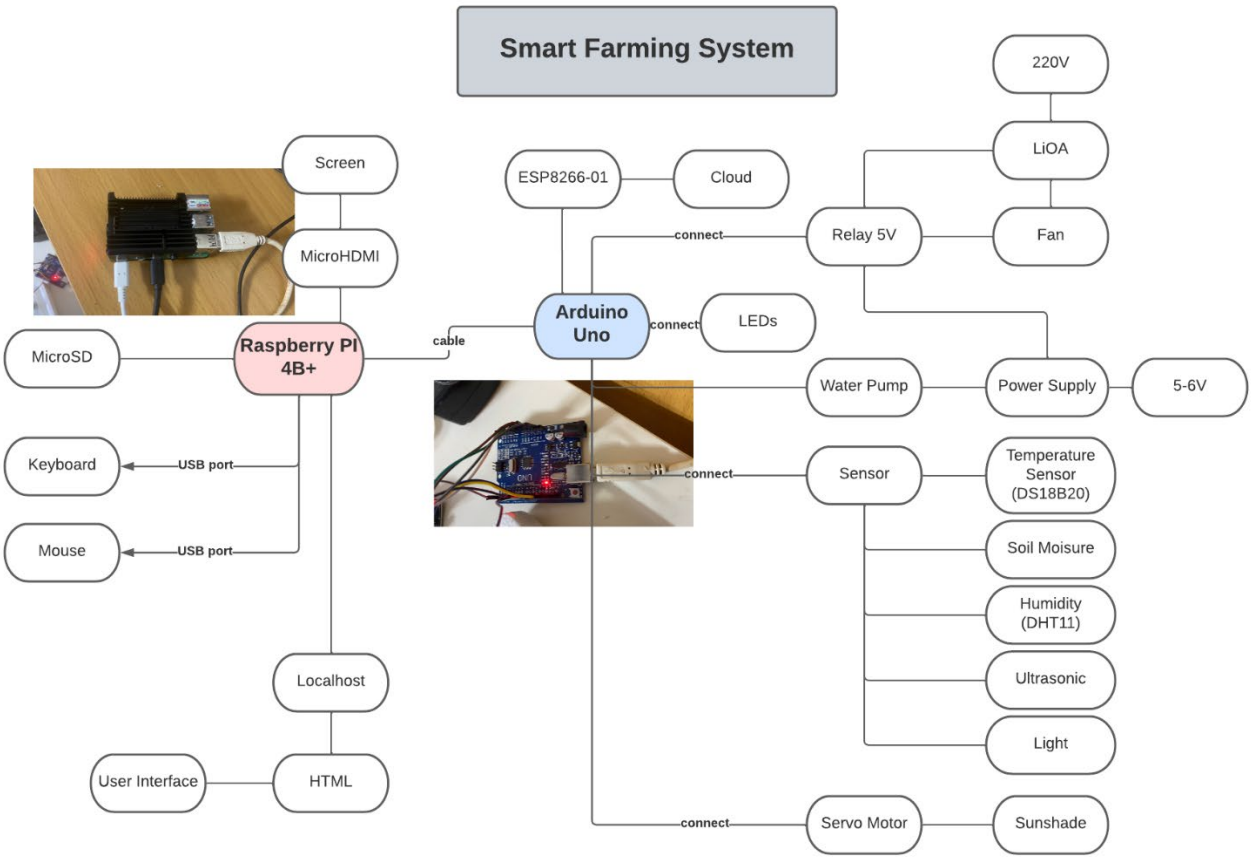
Nguyen Huynh Thao An - 103430367

TABLE OF CONTENT

INTRODUCTION

In this modern age, argiculture still plays an indispensable role in the economy. Nevertheless, with the advancement of technology, people, particularly farmers, have had an easier time growing crops comparing to the past. They can adjust the environment around the fields to keep the crops healthy and growing. They have a wide range of improved varieties for better productivity. They have the technology for more efficient farming. This paper will report on our proposed system for this specific topic- a smart farming system. In general, the system helps the farmers keep track of the environmental conditions of the farmland, such as temperature, humidity or light and from those gathered information, control the different appliances to help the crops grow better, without the need to manually do the tasks. It is a way to do farming more efficiently, increase the productivity while reducing the cost and minimizing the effort needed.

CONCEPTUAL DESIGN



TASK BREAKDOWN

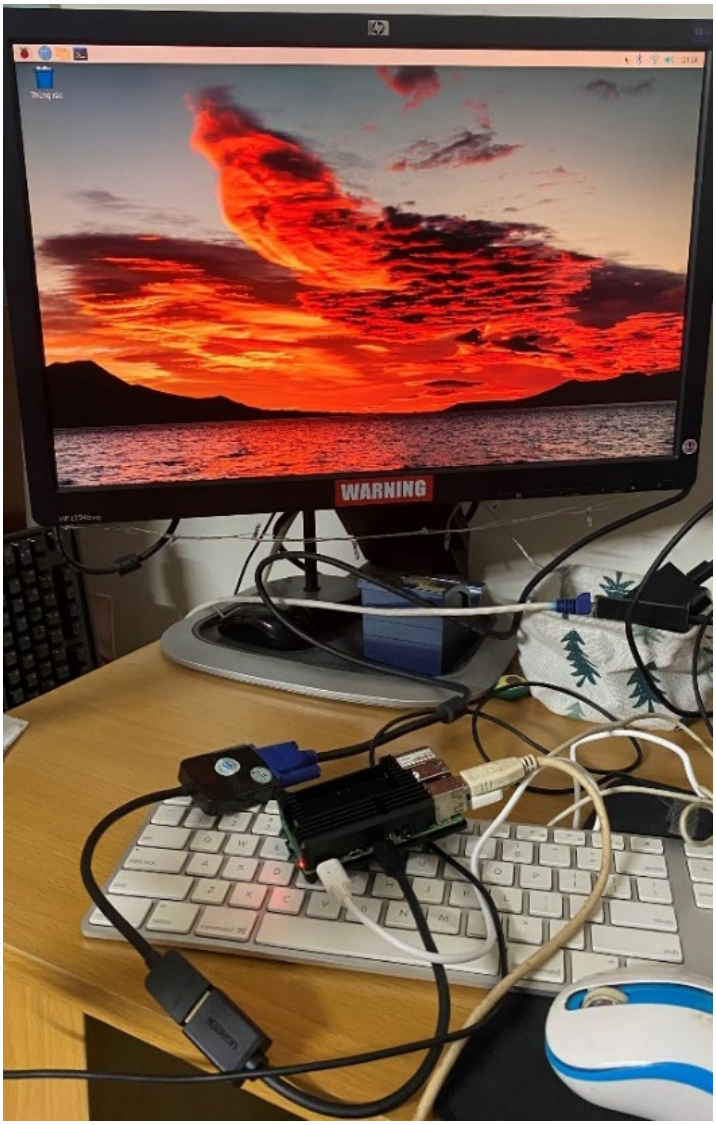
Tasks	Pham Le Yen Chi	Tran Duc Anh	Nguyen Huynh Thao An
Conceptualize the project and identify the components	Providing ideas on the topic and components used for the project.	Providing ideas on the topic and components used for the project.	Providing ideas on the topic and components used for the project.
Develop an IoT system	Coding for ultrasonic. Air humidity and fan. Ensemble the hardware and the finish code. Program the connection	Coding for temperature sensor, light sensor and sunshade. Help in building the hardware.	Coding for soil moisture sensor and water pump. Help in building the hardware.

	between the Arduino board and the WiFi module (ESP8266-01)		
IoT communication protocol (MQTT)	Program for the connection between the Arduino board and Cloud platform.	Researching the method and provide ideas.	Researching the method and provide ideas.
Define and develop rules analysing input data	Program for the connection between the Arduino board and Cloud platform to collect the data and control the actuators.	Program the sunshades to open when certain conditions (light or temperature) hit.	Program the water pump to turn on when certain condition (soil moisture) hits.
Website development	Research on possible approach. Upload the website on localhost in Raspberry Pi	Program the HTML code for the website.	Research on possible approach.
Cloud computing platform	Prepare the Cloud platform and connection between the Arduino board and Cloud to gather data, provide visualizations.	Research on possible approach.	Research on possible approach.
Documentation	Conceptual design report. Finalizing the report. Video demonstration.	Limitation report.	Writing the report.

IMPLEMENTATION

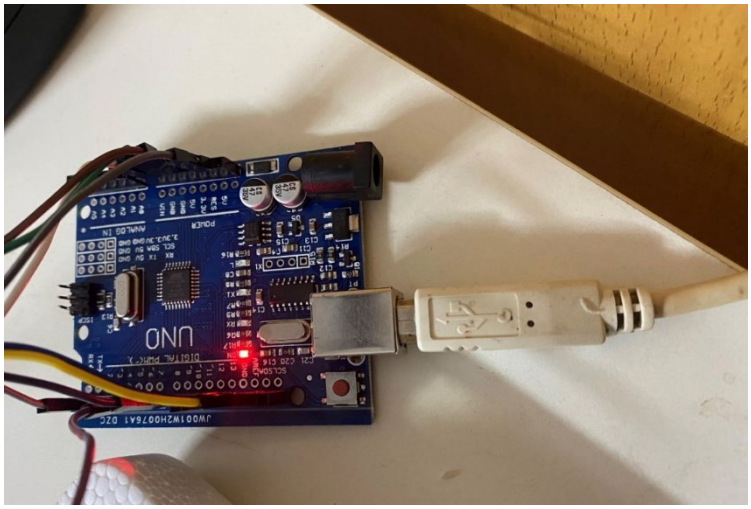
1. Raspberry and Arduino Setup

a. Raspberry Pi 4B+



- Raspberry Pi is connected to the screen by the micro-HDMI.
- The microSD in Raspberry Pi was written by the Raspberry Imager with the Linux environment.
- Keyboard and mouse are also plugged in via the USB port.

b. Arduino Uno

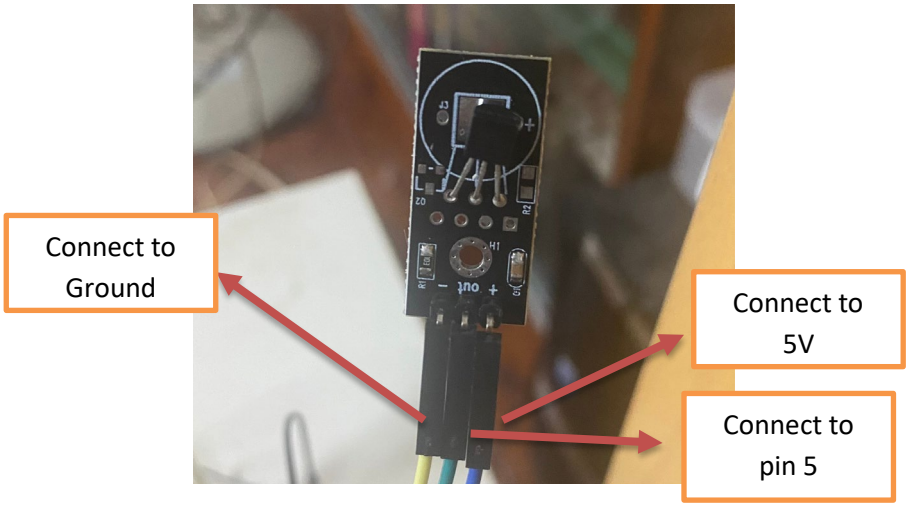


2. Sensors and Actuators implementation

Implement the needed sensors- temperature, humidity, soil moisture and light. Connect the sensors to the Arduino board and program for them to collect the information.

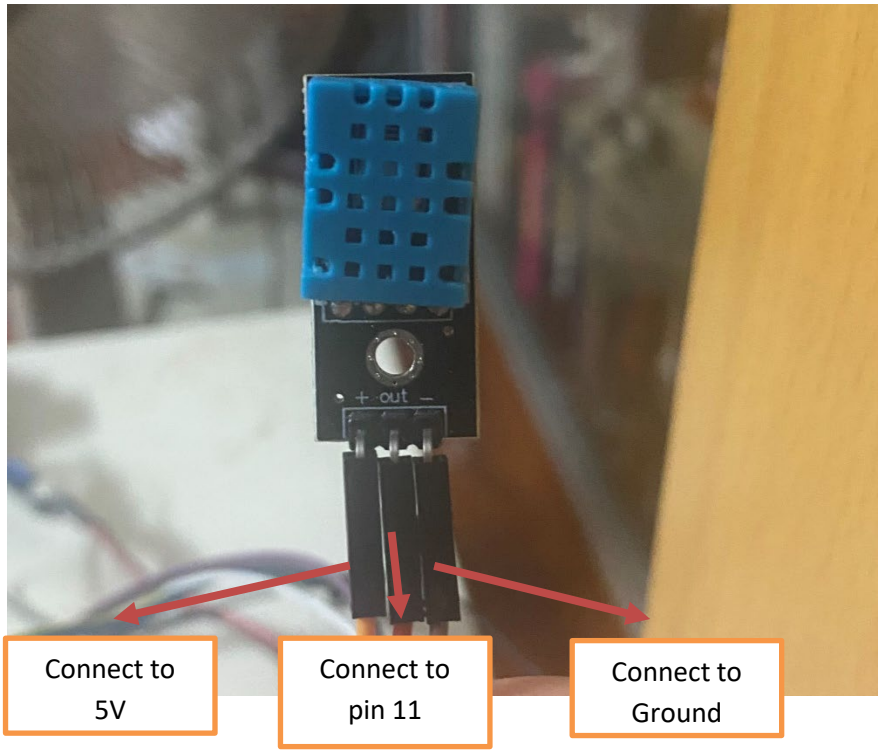
a. Temperature Sensor DS18B20

- The DS18B20 sensor has 3 pins connected to the Arduino pins as the note above

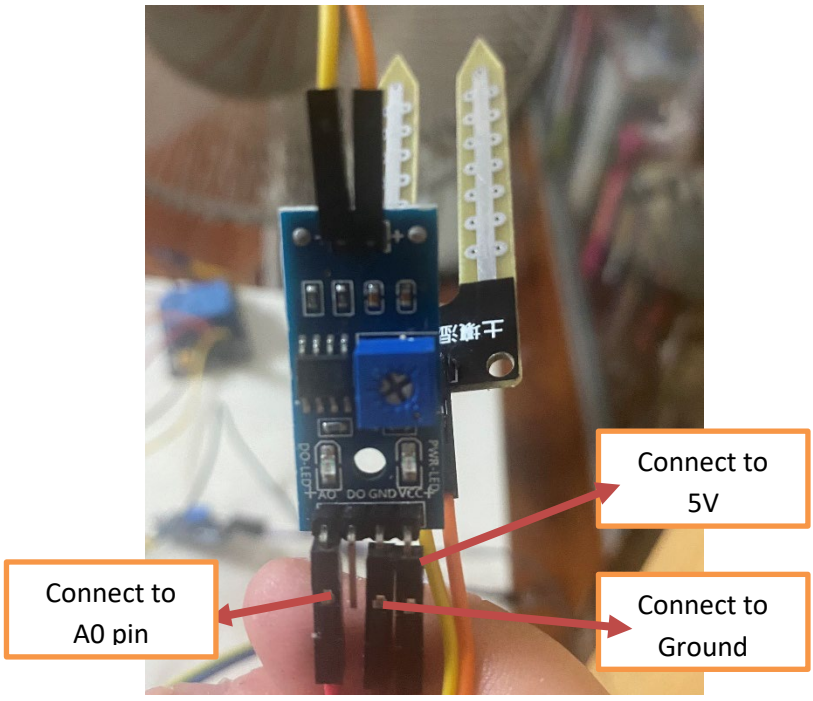


b. Humidity Sensor DHT11

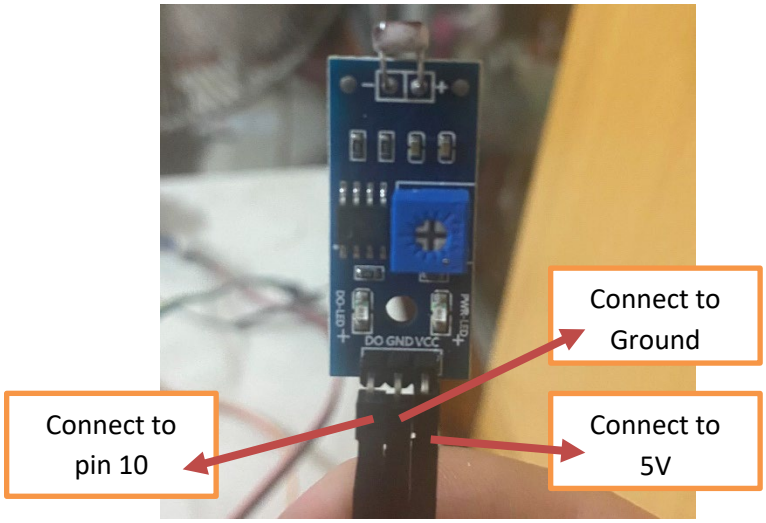
To implement this sensor, the code have to include the DHT library.



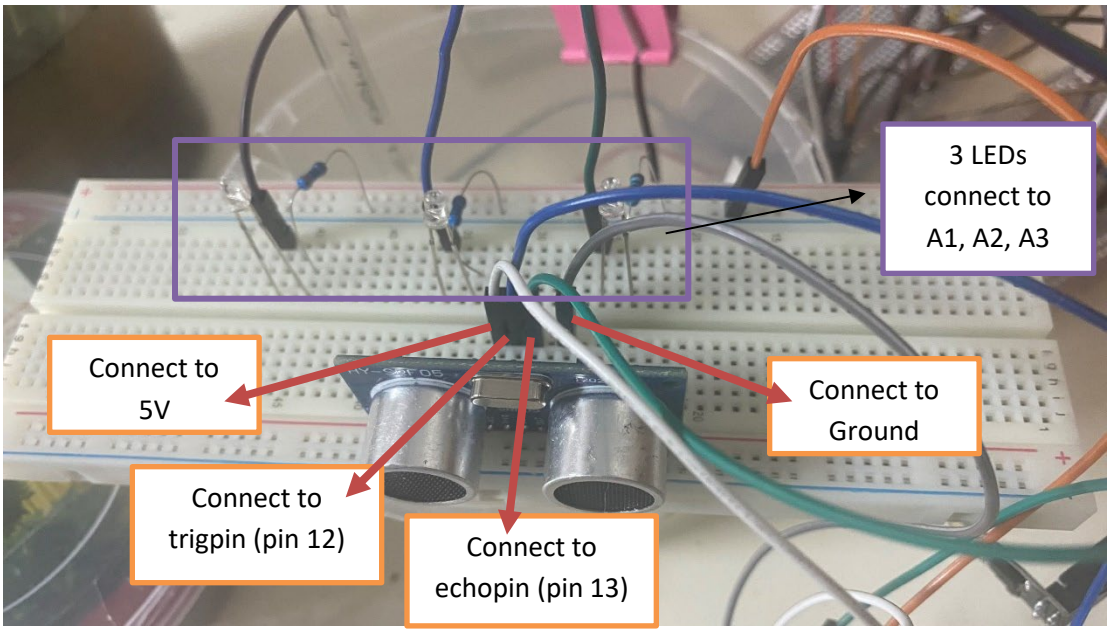
c. Soil Moisture Sensor



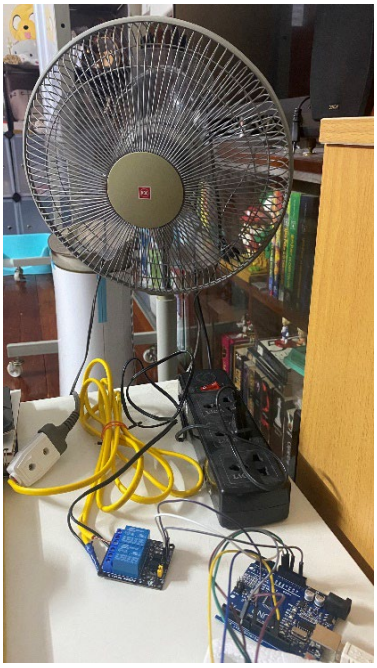
d. Light Sensor

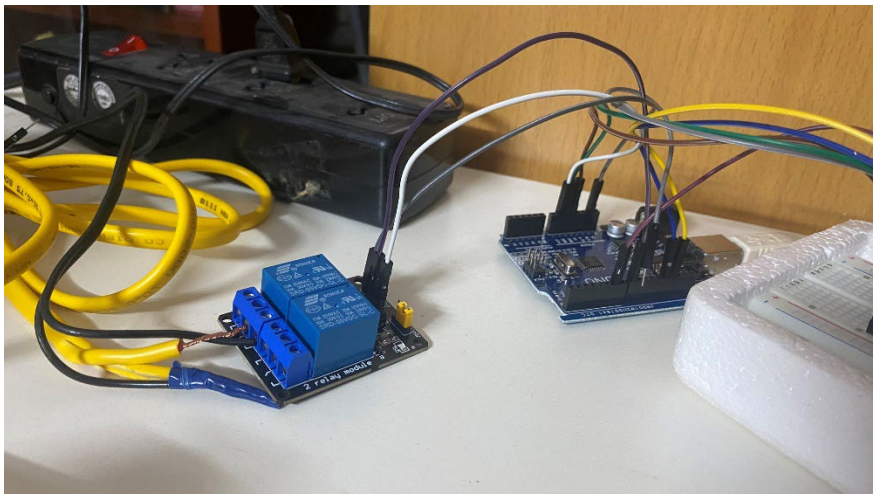


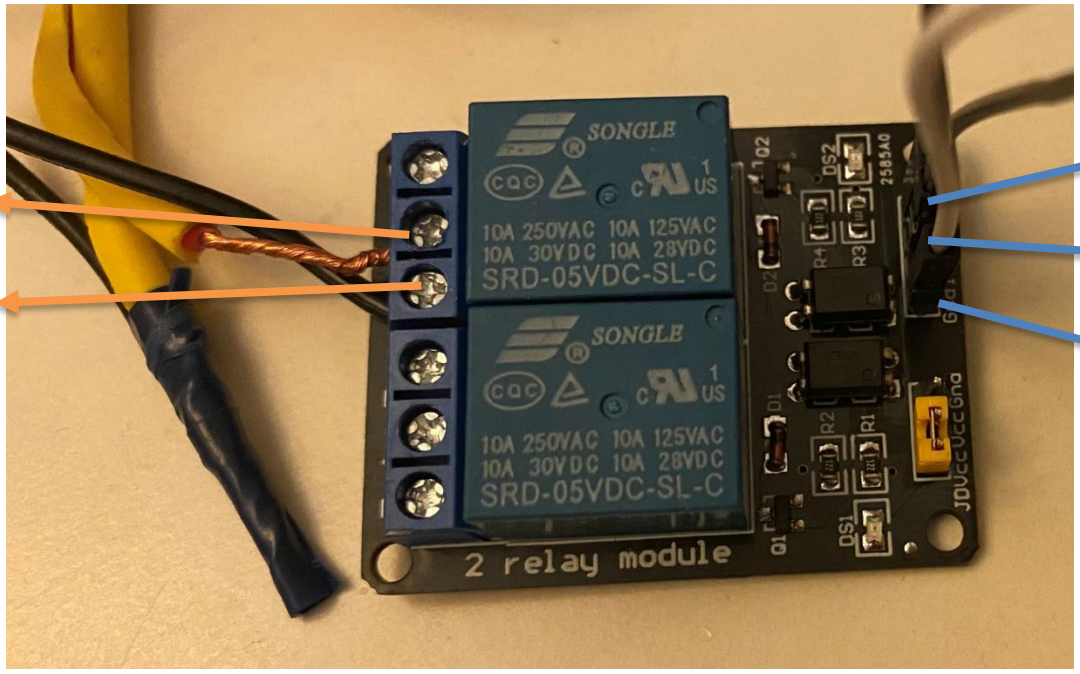
e. Ultrasonic Sensor



f. Fan & Relay 5V module

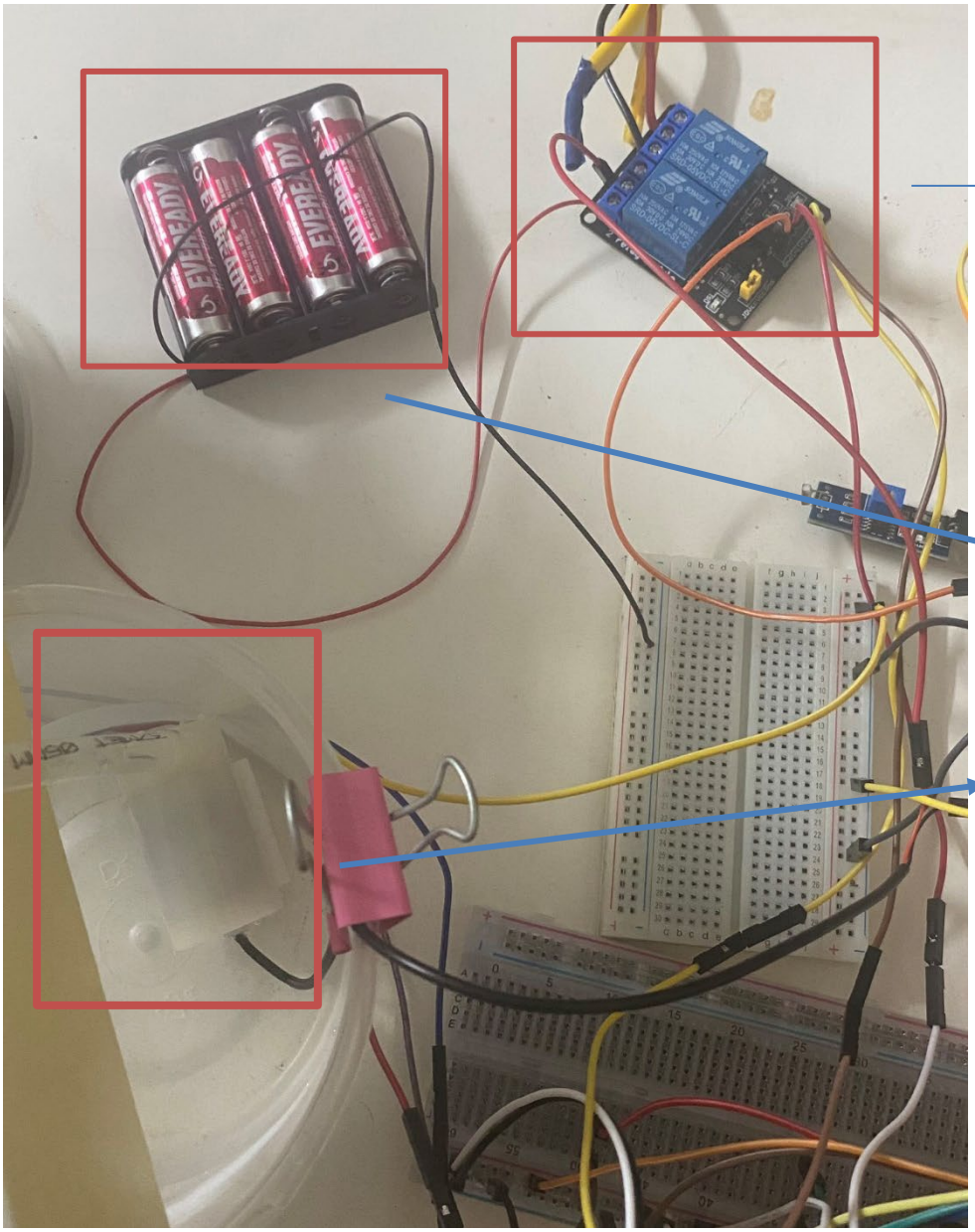






- Relay 5V is an automatic switch that help to turn on and turn off the fan. It connects to the LiOA by the 'normally closed' pin and then go to the fan to enhance the safety of the project because the system is connected to 220V.

g. Water Pump and Relay 5V module



- Power Supply has 2 wires, one goes to the ground and one goes to Relay 5V module (Normally Open).
- Water Pump has 2 wires, one goes to ground and one goes to Relay 5V module (Common Contact).

h. Servo Motor and Sunshade

- The Servo has 3 wires which are 5V, ground and pin 9 on Arduino.
- The Sunshade is connected to the Servo, so the servo will spin the sunshade with the condition of Light Sensor.
- To implement the servo motor, the code has to include the Servo library.

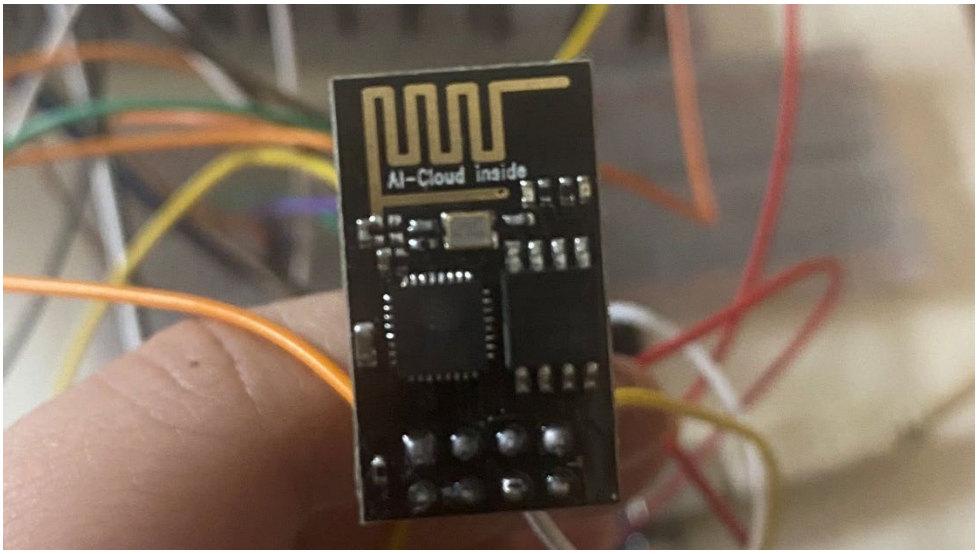
⇨ **Test the sensors with Serial print**: (the value is changed with the manual intervention.

```
Temperature: 24.00
Humidity: 63.00
Light in digital: 1
Soil Moisture: 1018
Water Level: 5
Temperature: 24.00
Humidity: 63.00
Light in digital: 1
Soil Moisture: 1017
Water Level: 5
```

```
Temperature: 26.30
Humidity: 67.00
Light in digital: 1
Soil Moisture: 0
Water Level: 5
Temperature: 26.60
Humidity: 66.00
Light in digital: 1
Soil Moisture: 0
Water Level: 5
```

3. ESP8266-01 configuration

Implement and test the performance of the WiFi module then connect it to the Arduino board with sensors to collect the data and prepare to upload those data to Cloud.



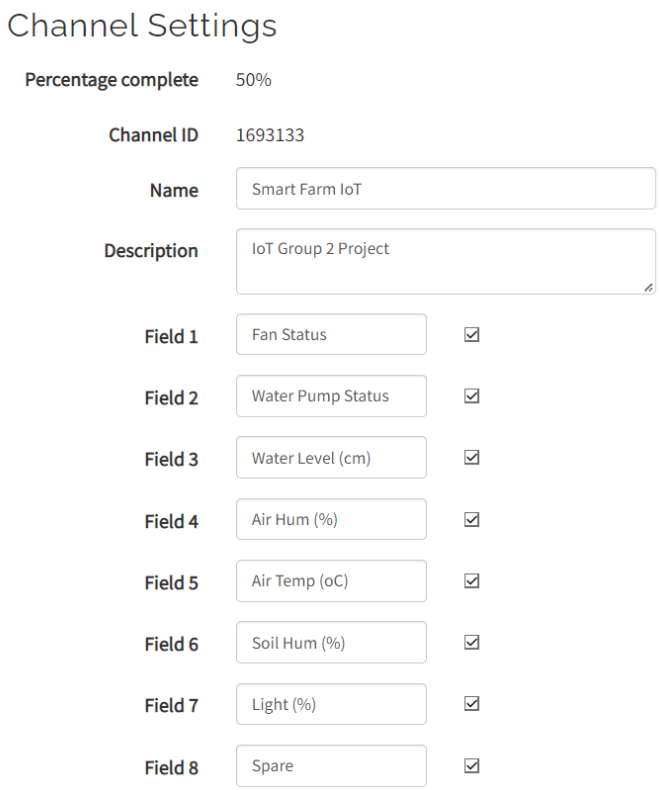
- The required library is SoftwareSerial
- Use AT command to set up and connect to WiFi:

```
AT+GMR
AT version:0.40.0.0(Aug  8 2015 14:45:58)
SDK version:1.3.0
Ai-Thinker Technology Co.,Ltd.
Build:1.3.0.2 Sep 11 2015 11:48:04
OK
```

```
AT+CWMODE=1

OK
AT+CWJAP="HomeNet_2.4G_plus","24101966"
WIFI CONNECTED
WIFI GOT IP

OK
AT+CIFSR
+CIFSR:STAIP,"192.168.1.14"
+CIFSR:STAMAC,"5c:cf:7f:1b:4d:27"

OK
```

4. ThingSpeak setup

Sign up and create an account on ThingSpeak - a Cloud computing platform. Create a channel for the sensors and actuators information. Also, ThingSpeak just allow users to create 8 fields. This cloud also provides users the write and read key to easily connect with Arduino.



Program the board to get data to ThingSpeak channel through ESP8266-01.

```
void writeThingSpeak(void)
{

  startThingSpeakCmd();

  // preparacao da string GET
  String getStr = "GET /update?api_key=";
  getStr += statusChWriteKey;
  getStr +="&field1=";
  getStr += String(fan);
  getStr +="&field2=";
  getStr += String(waterp);
  getStr +="&field3=";
  getStr += String(distance);
  getStr +="&field4=";
  getStr += String(airHum);
  getStr +="&field5=";
  getStr += String(airTemp);
  getStr +="&field6=";
  getStr += String(soilHum);
  getStr +="&field7=";
  getStr += String(light);
  getStr +="&field8=";
  getStr += String(spare);
  getStr += "\r\n\r\n";

  sendThingSpeakGetCmd(getStr);
}
```

Testing with Serial print:

```
enviado ==> Start cmd: AT+CIPSTART="TCP","184.106.153.149",80
enviado ==> lenght cmd: AT+CIPSEND=114
enviado ==> getStr: GET /update?api_key=LN2ADS20QCM8P8PE&field1=0&field2=0&field3=4&field4=68&field5=24&field6=2&field7=1&field8=0
```

5. Actuators and ThingSpeak connection

Coding in the IDE for actuators control - read from the data in ThingSpeak and make sure the actuators reponse accordingly to what have been programmed.

```
int readThingSpeak(String channelID)
{
  startThingSpeakCmd();
  int command;
  // preparacao da string GET
  String getStr = "GET /channels/";
  getStr += channelID;
  getStr +="/fields/1/last";
  getStr += "\r\n";

  String messageDown = sendThingSpeakGetCmd(getStr);
  if (messageDown[5] == 49)
  {
    command = messageDown[7]-48;
    Serial.print("Command received: ");
    Serial.println(command);
  }
  else command = 9;
  return command;
}
```

Testing with Serial print:

```
enviado ==> Start cmd: AT+CIPSTART="TCP","184.106.153.149",80
enviado ==> lenght cmd: AT+CIPSEND=37
enviado ==> getStr: GET /channels/1694448/fields/1/last

MessageBody received: +IPD,1:0CLOSED
Command received: 0
Fan: 0
```

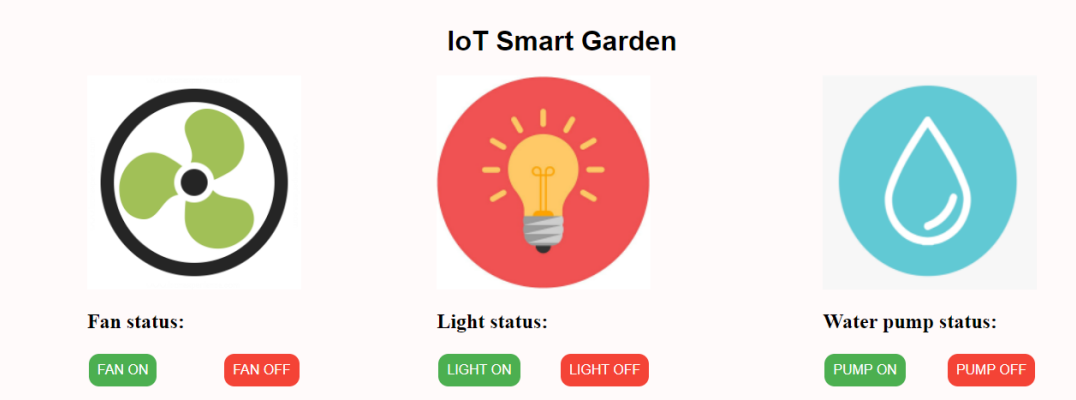
Turn on fan and water pump using buttons on the website

```
enviado ==> Start cmd: AT+CIPSTART="TCP","184.106.153.149",80
enviado ==> lenght cmd: AT+CIPSEND=37
enviado ==> getStr: GET /channels/1694447/fields/1/last

MessageBody received: +IPD,1:1CLOSED
Command received: 1
enviado ==> Start cmd: AT+CIPSTART="TCP","184.106.153.149",80
enviado ==> lenght cmd: AT+CIPSEND=37
enviado ==> getStr: GET /channels/1694448/fields/1/last

MessageBody received: +IPD,1:1CLOSED
Command received: 1
Fan: 1
Water Pump: 1
```

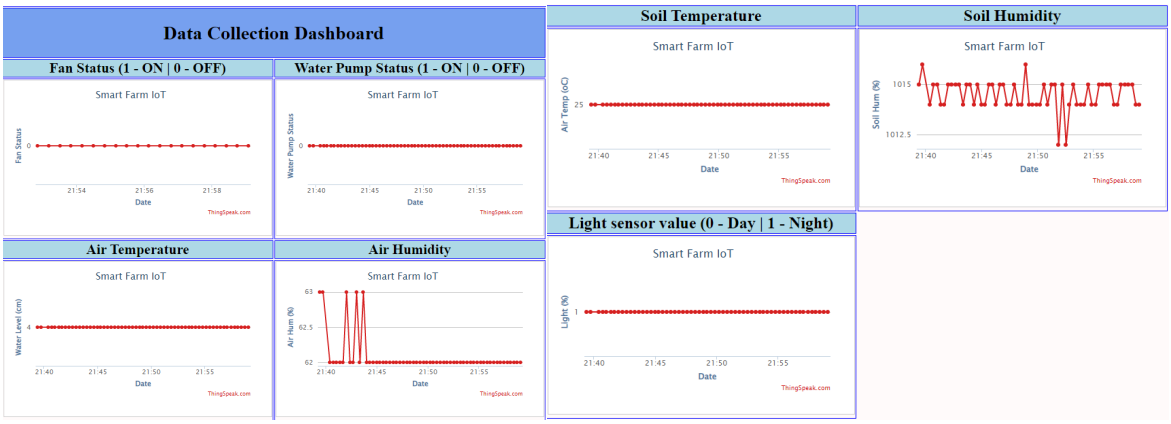
6. Website

The website has two main parts: Control Actuators and Monitor Sensors

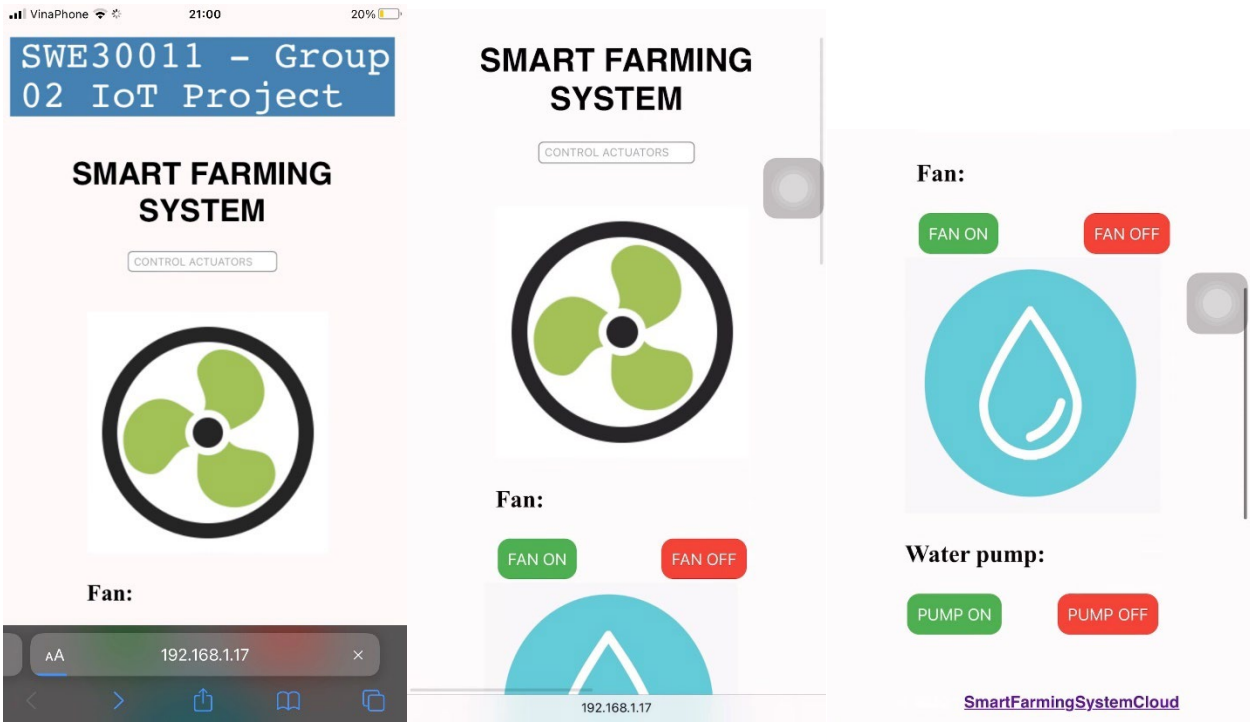
- Program a HTML website and connect it to ThingSpeak for the ability to control the appliances manually, instead of waiting for when conditions are met.



- The sensor monitor can be created by using the code provided in ThingSpeak.

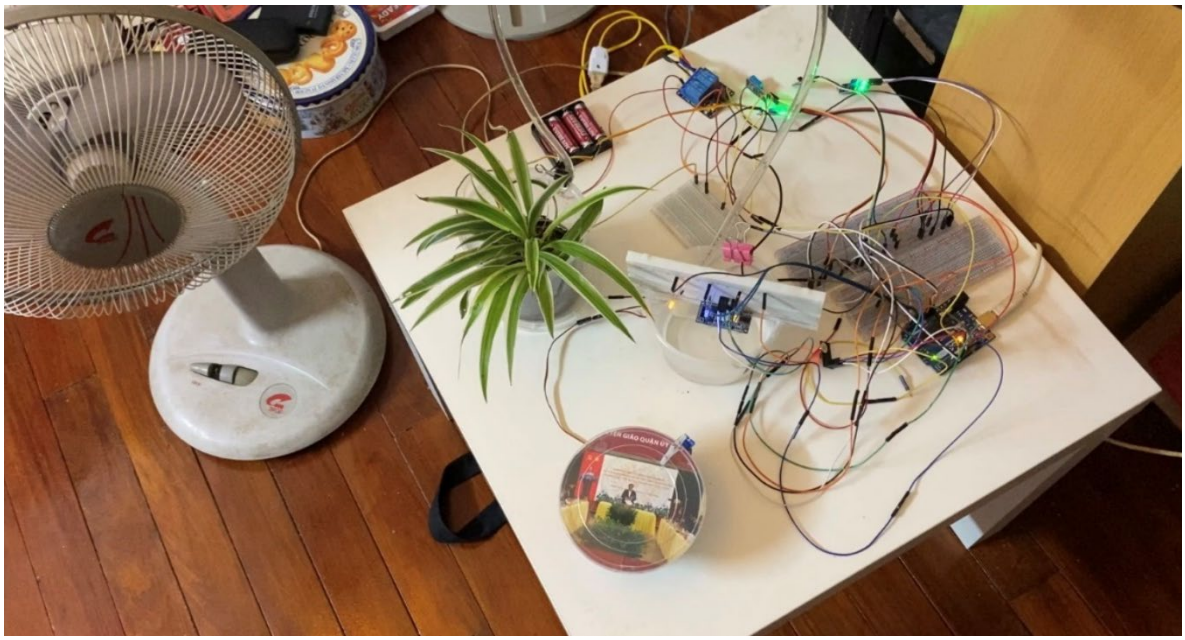


- The website is uploaded to localhost on Raspberry Pi, so users can access in many devices. The example below will demonstrate the website on smart phone.



7. Prototype ensemble

Ensemble the full code and the hardware into a workable prototype. Ensure the system is running smoothly and according to plan.



USER MANUAL

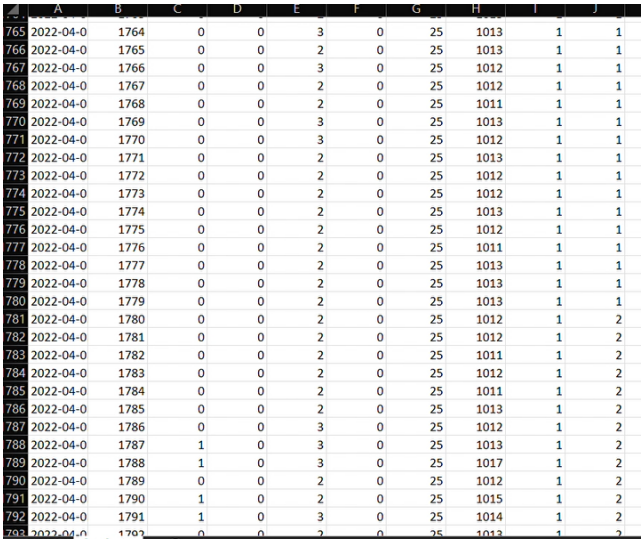
After running the system, it will automatically collect the environment data through the implemented sensors. When certain conditions are met, the corresponding appliances will do the pre-defined tasks:

- When the light reach a specific condition, the sunshade will be used to reduce sunlight exposure and cool the fields. This can help retain the water in the soil and help shade tolerant plants thrive with minimum effort. In this system, the sensor is used to read in digital value, which are 0 and 1 with 0 is for daytime and 1 is for nighttime.
- When the humidity is too high, a fan will be turned on to cool the atmosphere as high humidity levels will make it hard for the plant to make water evaporate or to draw nutrition from the soil. The plant may even rot if this - high humidity levels - continue for a prolonged length of time. In this system, the condition of humidity

value is set to 64.0. Therefore, when the sensor measures the air humidity that is higher than 64.0, the fan will be turned on.

- When the soil is too dry, an automatic water system will be turned on to water the plants as the soil moisture need to be at a certain level for the plant to continue growing. If soil moisture is too low, the plant's growth will be disrupted, and productivity may lower.
- The watering system consists of a water tank and a water pump inside. An ultrasonic is used to measure the water levels inside the tank, and there are 3 led lights to alert the user about the status of the water tank: the lower the water levels are, the more LEDs are turned off.

There is a downloadable report on Cloud platform about the collected data if users want to read and keep track of the environment around the fields. The administrator can also export the database (csv. file) in ThingSpeak.



	A	B	C	D	E	F	G	H	I	J
765	2022-04-0	1764	0	0	3	0	25	1013	1	1
766	2022-04-0	1765	0	0	2	0	25	1013	1	1
767	2022-04-0	1766	0	0	3	0	25	1012	1	1
768	2022-04-0	1767	0	0	2	0	25	1012	1	1
769	2022-04-0	1768	0	0	2	0	25	1011	1	1
770	2022-04-0	1769	0	0	3	0	25	1013	1	1
771	2022-04-0	1770	0	0	3	0	25	1012	1	1
772	2022-04-0	1771	0	0	2	0	25	1013	1	1
773	2022-04-0	1772	0	0	2	0	25	1012	1	1
774	2022-04-0	1773	0	0	2	0	25	1012	1	1
775	2022-04-0	1774	0	0	2	0	25	1013	1	1
776	2022-04-0	1775	0	0	2	0	25	1012	1	1
777	2022-04-0	1776	0	0	2	0	25	1011	1	1
778	2022-04-0	1777	0	0	2	0	25	1013	1	1
779	2022-04-0	1778	0	0	2	0	25	1013	1	1
780	2022-04-0	1779	0	0	2	0	25	1013	1	1
781	2022-04-0	1780	0	0	2	0	25	1012	1	2
782	2022-04-0	1781	0	0	2	0	25	1012	1	2
783	2022-04-0	1782	0	0	2	0	25	1011	1	2
784	2022-04-0	1783	0	0	2	0	25	1012	1	2
785	2022-04-0	1784	0	0	2	0	25	1011	1	2
786	2022-04-0	1785	0	0	2	0	25	1013	1	2
787	2022-04-0	1786	0	0	3	0	25	1012	1	2
788	2022-04-0	1787	1	0	3	0	25	1013	1	2
789	2022-04-0	1788	1	0	3	0	25	1017	1	2
790	2022-04-0	1789	0	0	2	0	25	1012	1	2
791	2022-04-0	1790	1	0	2	0	25	1015	1	2
792	2022-04-0	1791	1	0	3	0	25	1014	1	2
793	2022-04-0	1792	0	0	2	0	25	1013	1	2

Through the website, users can also manually control the appliances if they so desired, instead of waiting for the pre-set conditions, like turning on/off the fans or the water pump (by pressing the buttons). The report on ThingSpeak will also be updated to the website for monitoring purpose. Because the website is uploaded to localhost of Raspberry Pi, users can access via many devices by using the IP of the Pi.

LIMITATIONS AND IMPROVEMENT

1. Limitations

- The system is currently just a small and simple prototype, and it is hard programmed to react to a fixed range of sensor values. As such, it is only suitable for one plant, and one species at a time. The code will have to be changed to suit the needs of different plant species.
- There hardware is exposed and vulnerable to the outside environment as there are no covers.
- While ThingSpeak Cloud is free, it just allows 8 fields for all sensors and actuators. Therefore, the system can help users monitor all sensors but just control two main actuators that are fan and water pump.

2. Improvement

- Upgrade and expand the code to accommodate different types of plant at a time and can sense, or can be changed accordingly to what species is being grown.
- Make a cover for the hardware.
- Expand the coverage of the system to a larger field, instead of just one plant for each model.
- Add more sensors- like wind, sound- and other devices like cameras to widen the scope of the project.

RESOURCES

1. The lessons from Week 1 to Week 12 of IoT Programming unit.
2. Information of devices
   - Relay 5V module: https://www.elprocus.com/5v-relay-module/#:~:text=A%205v%20relay%20is%20an,ranges%20from%200%20to%205V.
   - DS18B20 sensor: https://www.elprocus.com/ds18b20-temperature-sensor/
   - DHT11 sensor: https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/
   - Light sensor: https://arduinogetstarted.com/tutorials/arduino-light-sensor

- Soil Moisture sensor: https://www.instructables.com/Arduino-Soil-Moisture-Sensor/#:~:text=Connect%20the%20two%20pins%20from,m%20interested%20in%20Analog%20Data).
- Ultrasonic sensor: https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6

3. Set up ESP8266-01 module: https://www.instructables.com/Getting-Started-With-the-ESP8266-ESP-01/
4. Software:
   - Sublime Text: HTML code
   - Arduino IDE
   - Adobe Premiere: edit video

APPENDIX

1. Code Arduino

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos = 0;
#define servo_pin 9

// Thingspeak
String statusChWriteKey = "LN2ADS20QCM8P8PE";  // Status Channel id: 385184
String canalID1 = "1694447"; //Actuator1
String canalID2 = "1694448"; //Actuator2

#include <SoftwareSerial.h>
SoftwareSerial EspSerial(6, 7); // Rx,  Tx
#define HARDWARE_RESET 8

// DS18B20
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 5 // DS18B20 on pin D5
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);
int airTemp = 0;
```

```
#define VCC2 2 //5V
#define GND2 3 //ground

//DHT
#include "DHT.h"
#include <stdlib.h>
int pinoDHT = 11;
int tipoDHT =  DHT11;
DHT dht(pinoDHT, tipoDHT);
int airHum = 0;

// LDR (Light)
#define ldrPIN 10
int light = 0;

// Soil humidity
#define soilHumPIN A0
int soilHum = 0;

//Water level
#define trigpin 12
#define echopin 13

// Variables to be used with timers
long writeTimingSeconds = 17; // ==> Define Sample time in seconds to send data
long readTimingSeconds = 10; // ==> Define Sample time in seconds to receive data
long startWriteTiming = 0;
long elapsedWriteTime = 0;

long startReadTiming = 0;
long elapsedReadTime = 0;

// Variables to be used with Actuators
#define ACTUATOR1 A5 //fan
#define ACTUATOR2 A4 //waterpump

//LEDs
#define led1 A1
#define led2 A2
#define led3 A3
```

```
int duration, distance;

boolean fan = 0;
boolean waterp = 0;

int spare = 0;
boolean error;

void setup()
{
 Serial.begin(9600);

 pinMode(ACTUATOR1,OUTPUT);
 pinMode(ACTUATOR2,OUTPUT);
 pinMode(HARDWARE_RESET,OUTPUT);

 pinMode(trigpin, OUTPUT);
 pinMode(echopin, INPUT);

 digitalWrite(HARDWARE_RESET, HIGH);

 DS18B20.begin();
 dht.begin();

 myservo.attach(servo_pin); //servo

 pinMode(VCC2,OUTPUT);
 digitalWrite(VCC2, HIGH);
 pinMode(GND2,OUTPUT);
 digitalWrite(GND2, LOW);

 digitalWrite(ACTUATOR1, LOW); //o módulo relé é ativo em LOW
 digitalWrite(ACTUATOR2, LOW); //o módulo relé é ativo em LOW

 //LEDs
 pinMode(led1, OUTPUT);
 pinMode(led2, OUTPUT);
 pinMode(led3, OUTPUT);

 digitalWrite(led1, LOW);
```

```
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);

  EspSerial.begin(9600); // Comunicacao com Modulo WiFi
  EspHardwareReset(); //Reset do Modulo WiFi
  startWriteTiming = millis(); // starting the "program clock"
}

void loop()
{


  start: //label
  error=0;

  elapsedWriteTime = millis()-startWriteTiming;
  elapsedReadTime = millis()-startReadTiming;

  if (elapsedWriteTime > (writeTimingSeconds*1000))
  {
    readSensors();
    writeThingSpeak();
    startWriteTiming = millis();
  }

  if (elapsedReadTime > (readTimingSeconds*1000))
  {
    int command = readThingSpeak(canalID1);
    if (command != 9) fan = command;
    delay (5000);
    command = readThingSpeak(canalID2);
    if (command != 9) waterp = command;
    takeActions();
    startReadTiming = millis();
  }

  if (airHum >= 64.0){   //độ ẩm trên 64 thì bật quạt
    digitalWrite(ACTUATOR1, HIGH);
    fan=1;
  }
```

```
if (airHum < 64.0){
  digitalWrite(ACTUATOR1, LOW);
  fan=0;
}

if (light >= 0 && light <=0)
{
  for (pos = 0; pos <= 180; pos += 1)
  { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos); // tell servo to go to position in variable 'pos'
    delay(15);         // waits 15ms for the servo to reach the position
  }
}

if(light >=1 && light <= 1)
{
  for (pos = 180; pos >= 0; pos -= 1)
  { // goes from 180 degrees to 0 degrees
    myservo.write(pos);             // tell servo to go to position in variable 'pos'
    delay(15);                      // waits 15ms for the servo to reach the position
  }
}

if(soilHum < 800){
  digitalWrite(ACTUATOR2, LOW);
  waterp=0;
}

if(soilHum > 800){
  digitalWrite(ACTUATOR2, HIGH);
  waterp=1;
}

if(  (distance > 0) && (distance <= 1)   )
{
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
} else
```

```
 if(  (distance > 1) && (distance <= 3)  )
 {

   digitalWrite(led1, LOW);
   digitalWrite(led2, HIGH);
   digitalWrite(led3, HIGH);

 } else

 if(  (distance > 3) && (distance <= 6)  )
 {

   digitalWrite(led1, LOW);
   digitalWrite(led2, LOW);
   digitalWrite(led3, HIGH);
 }

 if (error==1) //Resend if transmission is not completed
 {
   Serial.println(" <<<< ERROR >>>>");
   delay (2000);
   goto start; //go to label "start"
 }
}

/********* Read Sensors value *************/
void readSensors(void)
{
 digitalWrite(trigpin, HIGH);

 delayMicroseconds(1000);
 digitalWrite(trigpin, LOW);

 duration = pulseIn(echopin,HIGH);
 distance = ( duration / 2) / 29.1;

 airHum = dht.readHumidity();

 DS18B20.requestTemperatures();
 airTemp = DS18B20.getTempCByIndex(0); // Sensor 0 will capture Soil Temp in Celcius
```

```
  light = digitalRead(ldrPIN);
  soilHum = analogRead(soilHumPIN);
}

/********* Take actions based on ThingSpeak Commands *************/
void takeActions(void)
{
  Serial.print("Fan: ");
  Serial.println(fan);
  Serial.print("Water Pump: ");
  Serial.println(waterp);
  if (fan == 1) digitalWrite(ACTUATOR1, HIGH);
  else digitalWrite(ACTUATOR1, LOW);
  if (waterp == 1) digitalWrite(ACTUATOR2, HIGH);
  else digitalWrite(ACTUATOR2, LOW);
}

/********* Read Actuators command from ThingSpeak *************/
int readThingSpeak(String channelID)
{
  startThingSpeakCmd();
  int command;
  // preparacao da string GET
  String getStr = "GET /channels/";
  getStr += channelID;
  getStr +="/fields/1/last";
  getStr += "\r\n";

  String messageDown = sendThingSpeakGetCmd(getStr);
  if (messageDown[5] == 49)
  {
    command = messageDown[7]-48;
    Serial.print("Command received: ");
    Serial.println(command);
  }
  else command = 9;
  return command;
}


/********* Conexao com TCP com Thingspeak *******/
```

```
void writeThingSpeak(void)
{

  startThingSpeakCmd();

  // preparacao da string GET
  String getStr = "GET /update?api_key=";
  getStr += statusChWriteKey;
  getStr +="&field1=";
  getStr += String(fan);
  getStr +="&field2=";
  getStr += String(waterp);
  getStr +="&field3=";
  getStr += String(distance);
  getStr +="&field4=";
  getStr += String(airHum);
  getStr +="&field5=";
  getStr += String(airTemp);
  getStr +="&field6=";
  getStr += String(soilHum);
  getStr +="&field7=";
  getStr += String(light);
  getStr +="&field8=";
  getStr += String(spare);
  getStr += "\r\n\r\n";

  sendThingSpeakGetCmd(getStr);
}

/********* Reset ESP *************/
void EspHardwareReset(void)
{
  Serial.println("Reseting.......");
  digitalWrite(HARDWARE_RESET, LOW);
  delay(500);
  digitalWrite(HARDWARE_RESET, HIGH);
  delay(8000);//Tempo necessário para começar a ler
  Serial.println("RESET");
}

/********* Start communication with ThingSpeak*************/
```

```
void startThingSpeakCmd(void)
{
  EspSerial.flush();//limpa o buffer antes de começar a gravar

  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // Endereco IP de api.thingspeak.com
  cmd += "\",80";
  EspSerial.println(cmd);
  Serial.print("enviado ==> Start cmd: ");
  Serial.println(cmd);

  if(EspSerial.find("Error"))
  {
    Serial.println("AT+CIPSTART error");
    return;
  }
}

/********* send a GET cmd to ThingSpeak **********/
String sendThingSpeakGetCmd(String getStr)
{
  String cmd = "AT+CIPSEND=";
  cmd += String(getStr.length());
  EspSerial.println(cmd);
  Serial.print("enviado ==> lenght cmd: ");
  Serial.println(cmd);

  if(EspSerial.find((char *)">"))
  {
    EspSerial.print(getStr);
    Serial.print("enviado ==> getStr: ");
    Serial.println(getStr);
    delay(500);//tempo para processar o GET, sem este delay apresenta busy no próximo comando

    String messageBody = "";
    while (EspSerial.available())
    {
      String line = EspSerial.readStringUntil('\n');
      if (line.length() == 1)
      { //actual content starts after empty line (that has length 1)
        messageBody = EspSerial.readStringUntil('\n');
```

```
    }
   }
   Serial.print("MessageBody received: ");
   Serial.println(messageBody);
   return messageBody;
  }
  else
  {
   EspSerial.println("AT+CIPCLOSE");     // alert user
   Serial.println("ESP8266 CIPSEND ERROR: RESENDING"); //Resend...
   spare = spare + 1;
   error=1;
   return "error";
  }
}
```

2. HTML Code (Website)

```
<!DOCTYPE html>
<html lang="en">
  <head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
          <meta http-equiv="Access-Control-Allow-Origin" content="*">

     <script src="http://ajax.googleapis.com/ajax/libs/jquery/2.0.0/jquery.min.js"></script>

    <title>Smart Farming System</title>
    <style>
                  th {
                          font-size: 1.5em;
                          background-color:lightblue;
                  }

                  .icon {
                          width: 250px;
                          height: 250px;
                  }

                  .button {
                          border: none;
                          color: white;
                          text-align: center;
                          text-decoration: none;
```

```
                display: inline-block;
                font-size: 16px;
                margin: 4px 2px;
                padding: 10px 10px;
                cursor: pointer;
                border-radius: 12px;
        }

        button:hover {
                opacity: 0.5
        }

        .buttonON {
                background-color: #4CAF50;
        }

        .buttonOFF {
                float:right;
                background-color: #f44336;
        }

    </style>
  </head>
<body style="background-color:snow;">
  <header style="background-color:steelblue;">
        <p style="font-family: Courier; color:white; font-size:40px; padding-
left:10px;">SWE30011 - Group 02 IoT Project</p>
    </header>

    <center>
        <h1 style="font-family: Helvetica;color: black;">SMART FARMING SYSTEM</h1>
    </center>

  <div class="center">
        <div class="form">
          <form action="" method="get">
                        <center><input type="text" id="ip" class="ip" placeholder="CONTROL
ACTUATORS"></input><br></center>

                        <br><br>
                        <div style="float:left; padding-left:20%">
                                <img src="fan.png" class="icon">
                                <p><h2>Fan: </h2></P>
        <button type="button" id="D1-on" class="button buttonON" >FAN ON</button>
                                <button type="button" id="D1-off" class="button buttonOFF"
>FAN OFF</button>
                                <br>
```

```
                                </div>

                                <div style="float:right; padding-right:20%">
                                        <img src="water.png" class="icon">
                                        <p><h2>Water pump: </h2></P>
                                        <button type="button" id="D2-on" class="button buttonON"
>PUMP ON</button>
                                        <button type="button" id="D2-off" class="button buttonOFF"
>PUMP OFF</button>
                                        <br>
                                </div>

        </form>
                        <br><br>
                </div>
        </div>

        <div style="clear:both; padding-top:30px">
                <table border=2 bordercolor="#0000FF" align="center" >
                <tr>
                        <td colspan="2" style="background-color: #76a0ef;">
                                <h1 align="center" >Data Collection Dashboard</h1>
                        </td>
                </tr>

                <tr>
                        <th>Fan Status (1 - ON | 0 - OFF)</th>
                        <th>Water Pump Status (1 - ON | 0 - OFF)</th>
                </tr>

                <tr>
                        <td>
                                <iframe width="450" height="260" style="border: 1px solid #cccccc;"
src="https://thingspeak.com/channels/1693133/charts/1?bgcolor=%23ffffff&color=%23d62020&dynam
ic=true&results=20&type=line"></iframe>

                        </td>
                        <td>
                                <iframe width="450" height="260" style="border: 1px solid #cccccc;"
src="https://thingspeak.com/channels/1693133/charts/2?bgcolor=%23ffffff&color=%23d62020&dynam
ic=true&results=60&type=line&update=15"></iframe>

                        </td>
                </tr>

                <tr>
                        <th>Air Temperature</th>
```

```
                <th>Air Humidity</th>
            </tr>

            <tr>
                <td>
                    <iframe width="450" height="260" style="border: 1px solid #cccccc;"

    src="https://thingspeak.com/channels/1693133/charts/3?bgcolor=%23ffffff&color=%23d62020
&dynamic=true&results=60&type=line"></iframe>

                </td>
                <td>
                    <iframe width="450" height="260" style="border: 1px solid #cccccc;"

    src="https://thingspeak.com/channels/1693133/charts/4?bgcolor=%23ffffff&color=%23d62020
&dynamic=true&results=60&type=line&update=15"></iframe>

                </td>
            </tr>

            <tr>
                <th>Soil Temperature</th>
                <th>Soil Humidity</th>
            </tr>

            <tr>
                <td>
                    <iframe width="450" height="260" style="border: 1px solid #cccccc;"

    src="https://thingspeak.com/channels/1693133/charts/5?bgcolor=%23ffffff&color=%23d62020
&dynamic=true&results=60&type=line&update=15"></iframe>
                </td>
                <td>
                    <iframe width="450" height="260" style="border: 1px solid #cccccc;"

    src="https://thingspeak.com/channels/1693133/charts/6?bgcolor=%23ffffff&color=%23d62020
&dynamic=true&results=60&type=line&update=15"></iframe>
                </td>
            </tr>

            <tr>
                <th>Light sensor value (0 - Day | 1 - Night)</th>
            </tr>

            <tr>
                <td>
                    <iframe width="450" height="260" style="border: 1px solid #cccccc;"
```

```
        src="https://thingspeak.com/channels/1693133/charts/7?bgcolor=%23ffffff&color=%23d62020
&dynamic=true&results=60&type=line&update=15"></iframe>
                        </td>
                </tr>

        </div>

        <footer class="footer">
                <center>
                        <h4 style="font-family: Helvetica;color: white;">&copy; 2022 | <a
href="https://thingspeak.com/channels/1693133/">SmartFarmingSystemCloud</a> </h4>
                </center>
        </footer>

  </body>
  <script>
                document.getElementById('D1-on').addEventListener('click', function() {
                        var ip = document.getElementById('ip').value;

                        var url =
"https://api.thingspeak.com/update?api_key=2V3J8Z6LQ7UQW14W&field1=1"
                        $.getJSON(url, function(data) {
                                console.log(data);
                        });
                });

                document.getElementById('D1-off').addEventListener('click', function() {
                        var ip = document.getElementById('ip').value;
                        var url =
"http://api.thingspeak.com/update?api_key=2V3J8Z6LQ7UQW14W&field1=0"
                        $.getJSON(url, function(data) {
                                console.log(data);
                        });
                });


                document.getElementById('D2-on').addEventListener('click', function() {
                        var ip = document.getElementById('ip').value;
                        var url =
"http://api.thingspeak.com/update?api_key=04TU8CRPYU6WX09I&field1=1"
                        $.getJSON(url, function(data) {
                                console.log(data);
                        });
                });

                document.getElementById('D2-off').addEventListener('click', function() {
```

```
                                        var ip = document.getElementById('ip').value;
                                        var url =
"http://api.thingspeak.com/update?api_key=04TU8CRPYU6WX09I&field1=0"
                                        $.getJSON(url, function(data) {
                                                console.log(data);
                                        });
                        });

        </script>
</html>
```