



**ISLAMIC UNIVERSITY OF TECHNOLOGY  
(IUT)  
ORGANIZATION OF ISLAMIC COOPERATION (OIC)  
DEPARTMENT OF ELECTRICAL AND ELECTRONIC  
ENGINEERING**

**PROJECT REPORT**

**Project Title: Smart Agriculture and Gardening System**

Submitted To	Submitted By
<p><b>Mahbubur Rahman</b> <b>Lecturer, Department of</b> <b>EEE, Islamic University of</b> <b>Technology</b></p>	<p><b>Group Members</b></p> <p><b>1. Raiyan Ibne Hossain</b> <b>ID: 200021144 Section: A</b></p> <p><b>2. Jarin Tasnim Rahman</b> <b>ID: 200021316 Section: C</b></p> <p><b>3. Humayra Tasnim Farah</b> <b>ID: 200021318 Section: C</b></p> <p><b>4. M. Sadman Aster</b> <b>ID:200021344 Section: C</b></p> <p><b>5. Maimuna Biswas Noshin</b> <b>ID:200021347 Section:C</b></p>

<b>Introduction</b>	<b>4</b>
<b>1. Irrigation System with Soil Moisture Control &amp; Humidity, Temperature Sensor</b>	<b>4</b>
1.1 Module Summary	4
1.2 System Functionality	6
1.3 Flow Chart	6
1.4 Calibration Process	8
1.5 Code	8
1.6 Result	12
<b>2. Introduction to Food Spoilage System</b>	<b>12</b>
2.1 Preamble	12
2.2 Problem Statement	13
2.3 Feature Objective	13
2.4 Methodology	13
2.5 How to connect Buzzer, LEDs, MQ4 Sensor and NodeMCU to Arduino UNO R3	14
2.6 Program for NodeMCU	16
2.7 Program for Arduino	17
2.8 Hardware Components	19
2.9 Implementation	21
2.10 Process	23
<b>3. Automatic Rain Shield System</b>	<b>26</b>
3.1 Problem Statement	26
3.2 Feature Objective	26
3.3 Module Summary	27
3.4 Circuit Diagram	28
3.5 Steps & Process	28

3.6 Step By Step	30
3.7 Arduino Code	31
3.8 Serial Monitor Reading	32
3.9 Implemented Circuit Diagram	33
4. Renewable Energy Generation	34
4.1 Specifications	34
4.2 Working Procedure	34
4.3 Hardware Implementation	34
Cost Report	35
Conclusion	36

# Introduction

Our team has identified various issues with the prevalent farming and gardening scenarios of Bangladesh. The issues are as follows:

1. There is no provision for detecting produce spoilage.
2. In case of excessive rain (especially during winter), there is no system which can protect the crop from getting damaged.
3. The farming and gardening processes are totally manual; no part of it is automated.
4. The processes rely totally on conventional fossil fuel-based energy, which is bad for the environment.

This is where our project “Smart Agriculture and Gardening System” comes into play. It is capable of mitigating all these problems. Our project consists of four modules, namely:

1. Automated Smart Irrigation
2. Food Spoilage Detection System
3. Automated Rain Shield
4. Renewable Energy Generation

## 1. Irrigation System with Soil Moisture Control & Humidity, Temperature Sensor

In contemporary agricultural practices, inefficient water usage and lack of precise irrigation methods contribute to water wastage and suboptimal plant growth. Traditional irrigation systems often rely on fixed schedules rather than responsive, real-time data, leading to over-watering or under-watering of crops. Additionally, environmental factors such as temperature and humidity are crucial in determining the water requirements of plants.

To address these issues, there is a pressing need for an intelligent irrigation system that integrates soil moisture data, environmental monitoring, and responsive control mechanisms.

### 1.1 Module Summary

The irrigation system aims to provide an efficient and automated solution for irrigating garden soil based on soil moisture levels. The system utilizes a NodeMCU

ESP8266 microcontroller, soil moisture sensor, relay module, solenoid water valve, and additional sensors for temperature, humidity, and water level monitoring. NodeMCU ESP8266, Soil Moisture Sensor, Relay Module, Solenoid Water Valve, Breadboard, Jumpers, 12V Battery have been used here.

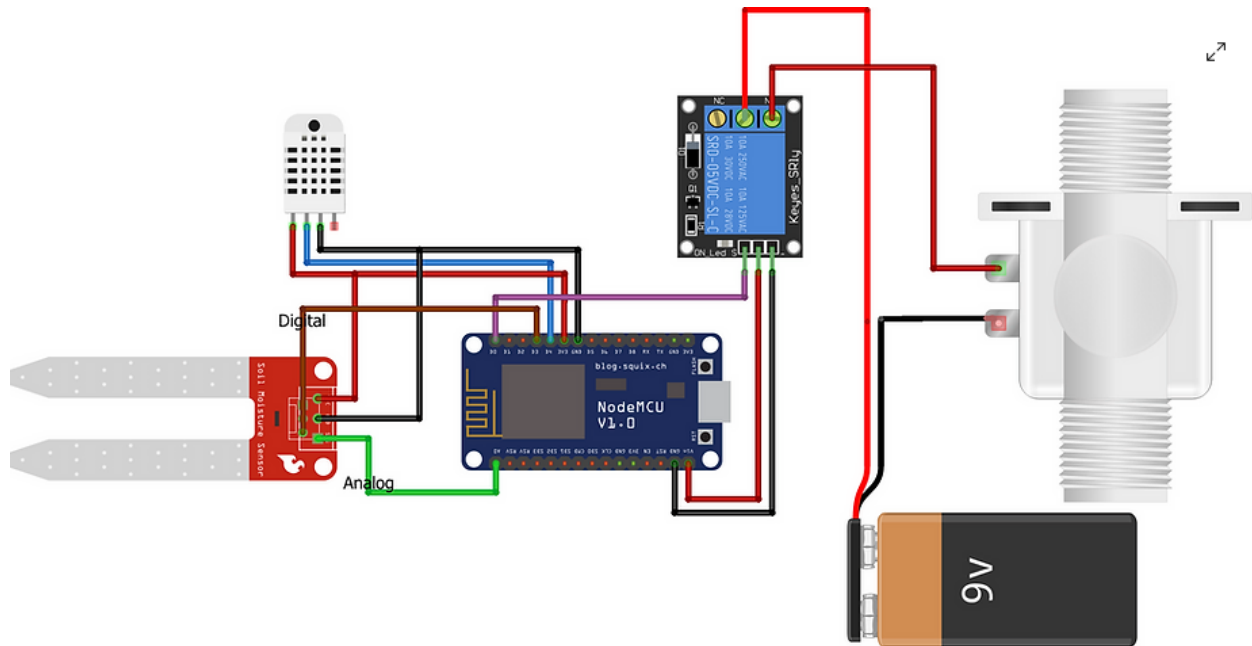


Fig: Circuit Schematic



Fig: Hardware Implementation

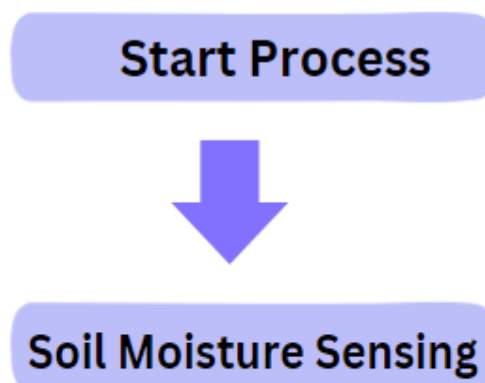
## 1.2 System Functionality

The irrigation system integrates multiple components to provide a comprehensive and automated solution for efficient soil management. Beginning with the soil moisture sensing, the calibrated Soil Moisture Sensor delivers readings in both dry and wet soil conditions. These readings are then processed by the NodeMCU, which determines the optimal moisture threshold for triggering irrigation events.

Moving to relay control, the NodeMCU takes charge of the Relay Module, which, in turn, manages the opening and closing of the Solenoid Water Valve based on real-time soil moisture levels. The irrigation system comes into play when the soil moisture falls below the predefined threshold, ensuring water is dispensed only when necessary.

The system extends its capabilities with additional sensors for humidity and moisture sensing, offering a holistic approach to environmental monitoring. To provide users with real-time control and monitoring capabilities, the NodeMCU seamlessly communicates with a dedicated mobile app. This integration allows users to receive notifications, monitor current soil conditions, and remotely control the irrigation system. The mobile app becomes a central hub for users to stay informed and make timely decisions regarding their garden's water management.

## 1.3 Flow Chart



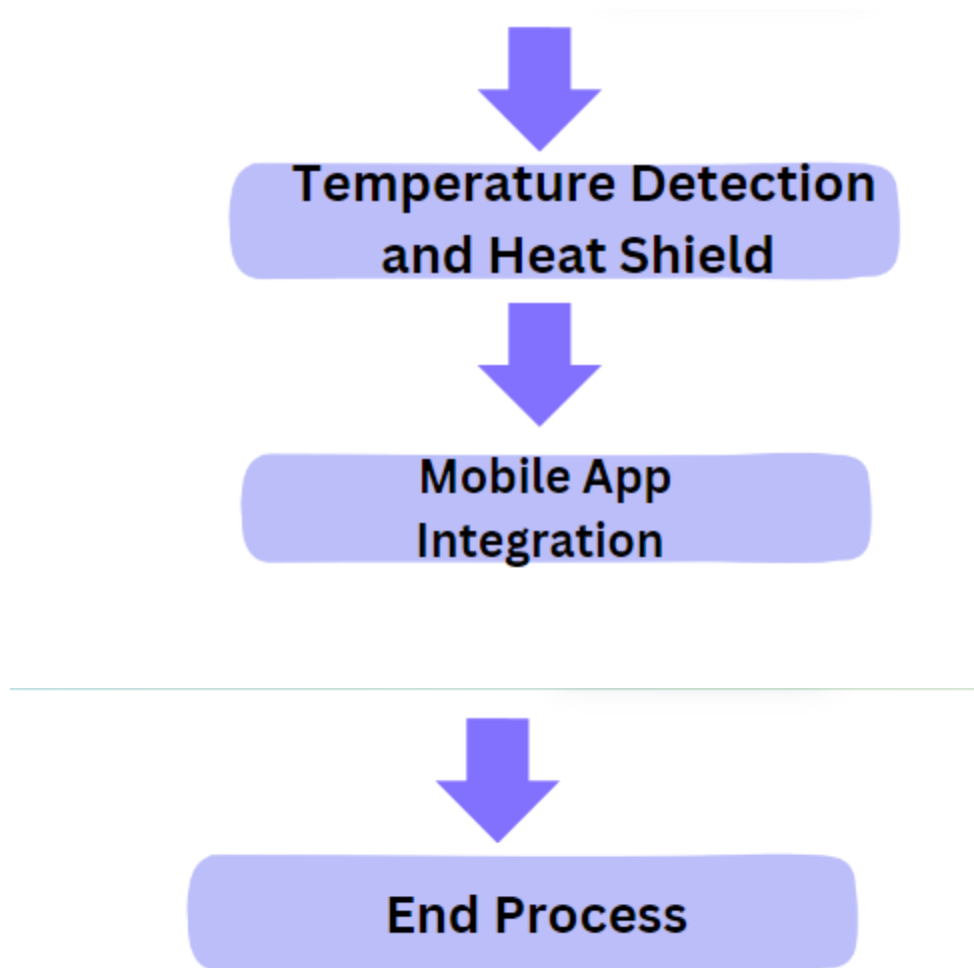
```
graph TD; A[ ] --> B[Determine Optimal Moisture Threshold]; B --> C[Relay Control Based on Moisture]; C --> D[Irrigation System]; D --> E[Humidity and Moisture Sensing];
```

**Determine Optimal  
Moisture Threshold**

**Relay Control Based  
on Moisture**

**Irrigation System**

**Humidity and Moisture  
Sensing**



#### 1.4 Calibration Process

The calibration process of the Soil Moisture Sensor involves obtaining readings in both dry and wet soil conditions. By exposing the sensor to these extremes, a baseline is established, allowing for the determination of the optimal moisture threshold. This threshold serves as a reference point for the irrigation system's configuration. The system is then set to trigger irrigation events only when the soil moisture falls below this predefined threshold. This calibration ensures that watering is carried out judiciously, optimizing water usage and promoting efficient resource management within the irrigation system.

#### 1.5 Code

```
#define BLYNK_TEMPLATE_ID "TMPL6wF5Z18cs"
```



```

#define BLYNK_TEMPLATE_NAME "smart plant2"

#define BLYNK_PRINT Serial
#include <OneWire.h>
#include <SPI.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS D2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
char auth[] = "P5oLRn9esTp86eA3x64EUbXdGCkfhO-t";
char ssid[] = "sadman";
char pass[] = "eee4518project";

#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
SimpleTimer timer;
void sendSensor()
{
float h = dht.readHumidity();
float t = dht.readTemperature();

if (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT sensor!");
return;
}

Blynk.virtualWrite(V5, h); //V5 is for Humidity
Blynk.virtualWrite(V6, t); //V6 is for Temperature
}

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

timer.setInterval(1000L, sendSensor);
Blynk.begin(auth, ssid, pass);

```

```
sensors.begin();
}
int sensor=0;
int output=0;
void sendTemps()
{
  sensor=analogRead(A0);
  output=(145-map(sensor,0,1023,0,100)); //in place 145 there is 100(it change with
  the change in sensor)
  delay(1000);
  sensors.requestTemperatures();
  float temp = sensors.getTempCByIndex(0);
  Serial.println(temp);
  Serial.print("moisture = ");
  Serial.print(output);
  Serial.println("%");
  Blynk.virtualWrite(V1, temp);
  Blynk.virtualWrite(V2,output);
  delay(1000);
}
void loop()
{
  Blynk.run();
  timer.run();
  sendTemps();
}
```

```
#define BLYNK_TEMPLATE_ID "TMPL6C_ULPFBO"
#define BLYNK_TEMPLATE_NAME "water"
#define BLYNK_AUTH_TOKEN "9XsuExoUkPpikioqa6UDz18az5lIKw9O"
#define BLYNK_PRINT Serial
#include <BlynkSimpleEsp8266.h>

char auth[] = "9XsuExoUkPpikioqa6UDz18az5lIKw9O";
char ssid[] = "CEEEEE";
char pass[] = "zerospark0";

int relayPin = D0; // Define the pin to which the relay is connected

BLYNK_WRITE(V0) // Virtual pin to control the relay
{
    int relayState = param.asInt();
    digitalWrite(relayPin, relayState);
}

void setup()
{
    Serial.begin(9600);
    pinMode(relayPin, OUTPUT);
    Blynk.begin(auth, ssid, pass);
}

void loop()
```

```
{  
  Blynk.run();  
}
```

## 1.6 Result

The smart irrigation system has yielded significant results and achievements in various aspects. Firstly, it has demonstrated efficient water management by leveraging real-time soil moisture data to irrigate the soil precisely when needed, thus optimizing water resources.

In terms of environmental monitoring, the integration of humidity and temperature sensors has provided a comprehensive view of the garden's conditions. This functionality ensures optimal plant health by considering not only moisture levels but also the broader environmental context.

The incorporation of a mobile app has enhanced user experience and convenience. Users can remotely monitor and control the irrigation system, receiving real-time updates on soil conditions and irrigation activities. This feature provides flexibility and ease of management for users, allowing them to make informed decisions about their garden's water needs.

Moreover, the system's energy independence, facilitated by a 12V battery, is a notable achievement. This design choice enhances the system's autonomy, making it suitable for off-grid applications where a constant power source may not be readily available. Overall, these achievements underscore the effectiveness and versatility of the smart irrigation system in promoting efficient water usage and plant care.

## 2. Introduction to Food Spoilage System

### 2.1 Preamble

Consuming nutritious and fresh food is paramount for maintaining optimal health and energy levels, enabling us to tackle daily activities. Unfortunately, the prevalence of bacterial growth and chemical changes in food stored at room temperature poses a significant risk to our well-being. Hostel messes and school kitchens, in particular, often struggle to provide high-quality meals, contributing to widespread health issues. The importance of monitoring food safety throughout the

supply chain cannot be overstated, as decay produces harmful gasses like ethanol and methane. Addressing this, a proposed system, driven by Arduino-based sensors, aims to detect early signs of food spoilage, preventing the consumption of deteriorated food that can lead to various foodborne diseases.

The global impact of food poisoning, claiming around 351,000 lives annually, underscores the urgency of implementing effective food safety measures. In some regions, challenges such as food scarcity exacerbate the problem, as preservation methods and chemical additives contribute to illnesses. What makes food contamination particularly insidious is its often invisible nature; even seemingly normal-looking food can harbor dangerous toxins and bacteria. The proposed system seeks to empower consumers by providing a tangible means of assessing the freshness and quality of their food, promoting awareness about nutritional values and mitigating the risks associated with food poisoning. This innovative approach not only safeguards health but also addresses the broader issues of food wastage and the urgent need for quality monitoring in our food supply chains.

## **2.2 Problem Statement**

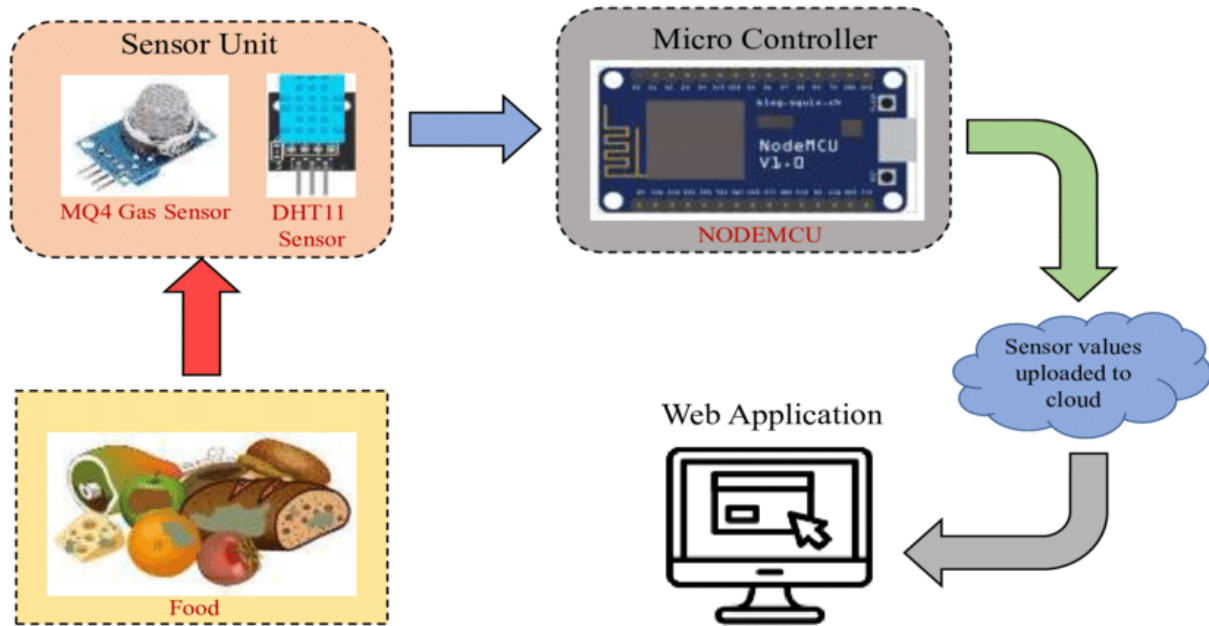
Now-a-days everyone is getting affected after consuming spoiled food, sometimes it leads to food poisoning. The spoiled food not only affects the health of people but also increases in the food waste. So, there is need to check the quality of the stored food time to time which will also help in reduction of food waste.

## **2.3 Feature Objective**

To develop a Hardware product which will sense the food specifically fruits and can identify their quality. Here we will use the methane sensor which will measures the alcohol content, CH<sub>4</sub> gas, LPG, Air as well as CO.

## **2.4 Methodology**

**BLOCK DIAGRAM OF FOOD SPOILAGE DETECTION SYSTEM:**



To power the Arduino, various sources such as a barrel adapter, USB connector linked to a PC/laptop, batteries exceeding 5V, or a dedicated battery shield can be utilized. In our setup, we opted for the USB connection to a PC. The essential components include an MQ4 Methane sensor, ESP8266 wifi wireless module, green LED, red LED, along with miscellaneous items like a breadboard, connectors, resistors, batteries for alternative power sources, and a buzzer. These components are seamlessly integrated into the Arduino using the breadboard. Establishing a network is facilitated through a mobile hotspot, detected by the ESP8266 wifi module. The Arduino then transmits sensor data to an IoT portal or cloud platform, allowing users to monitor the methane levels (in ppm) produced by spoiled food.

When the MQ4 Methane sensor detects food spoilage, an alert is triggered - the buzzer activates, and a red LED illuminates, providing a visual and auditory indication of the issue. Simultaneously, an IoT application, such as the Blynk app in our case, generates an email alert, promptly notifying the user of the Food Spoilage Detection (FSD) system. This comprehensive system ensures that users are promptly informed about potential food spoilage, enabling timely action to preserve food safety.

## 2.5 How to connect Buzzer, LEDs, MQ4 Sensor and NodeMCU to Arduino UNO R3

1) Connect Buzzer to Arduino UNO:

Positive terminal (long end) should be connected to any digital pin as digital pin 010 on Arduino

Negative terminal (short end) should be connected to common ground on breadboard

## 2) Connect LEDs to Arduino UNO:

Positive terminal to digital pin D13 or any other digital pin on Arduino

Negative terminal to common ground on breadboard

## 3) Connect NodeMCU to Arduino UNO:

i) Here we are using NodeMCU with ESP8266 12E WiFi Module. Serial communication is required to transfer data between Arduino and NodeMCU.

ii) Rx of NodeMCU to any digital pin say D9

iii) Tx of NodeMCU to any digital pin say D8

iv) GND to GND of Arduino for common GND

v) Vin pin of NodeMCU to 3.3V supply of Arduino

## 4) How to connect MQ4 Sensor to Arduino UNO R3:

i) Analog pin of sensor i.e. A0 to A5 or any analog pin of Arduino

ii) D0 digital pin is kept as it is

iii) GND pin of sensor is connected to GND pin of Arduino (or common GND)

iv) Vcc pin of sensor to 5V power supply pin of Arduino

For all these data captured by sensor to be transferred from Arduino to NodeMCU and also to the Blynk app cloud through ESP8266 12E WiFi module, few libraries are to be installed in Arduino IDE application.

It includes:

i) ESP8266 by community version

SoftwareSerial library

Blynk library

Also Blynk app should be installed on your Android phone.

## 2.6 Program for NodeMCU

methane\_nodemcu | Arduino 1.8.19 (Windows Store 1.8.57.0)

File Edit Sketch Tools Help

```
methane_nodemcu

#define BLYNK_TEMPLATE_ID "TMPL6kIeLJvPW"
#define BLYNK_TEMPLATE_NAME "IOT plant monitoring2"
#define BLYNK_AUTH_TOKEN "zs8OGIUL6HLhBCLyMMrKGADhEiUqdOeY"

#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <SoftwareSerial.h>

char auth[] = "zs8OGIUL6HLhBCLyMMrKGADhEiUqdOeY"; // Replace with your Blynk auth token

// Your WIFI credentials. Set password to "" for open networks.
char ssid[] = "Maidenless56";
char pass[] = "sadEEE34";

String myString = ""; // Complete message from Arduino, which consists of sensor data

BlynkTimer timer;
```

methane\_nodemcu | Arduino 1.8.19 (Windows Store 1.8.57.0)

File Edit Sketch Tools Help

```
methane_nodemcu

void setup() {
  // Debug console
  Serial.begin(9600);

  Blynk.begin(auth, ssid, pass);

  timer.setInterval(1000L, sensorValuel); // Define your timer interval for sending data

  myString.reserve(64); // Reserve memory for the String to avoid fragmentation
}

void loop() {
  if (Serial.available() > 0) {
    char rdata = Serial.read();
    myString += rdata;

    if (rdata == '\n') {
      // If a newline character is received, process the data and reset myString
      sensorValuel();
    }
  }

  Blynk.run();
  timer.run();
}
```



```

void sensorValue1() {
    String sdata = myString;


    // Send the sensor data to Blynk (assuming you have Virtual Pin V12 in your Blynk project)
    Blynk.virtualWrite(V12, sdata);

    myString = ""; // Clear the string to prepare for the next set of data
}

```

---

## 2.7 Program for Arduino

 methane\_arduino | Arduino 1.8.19 (Windows Store 1.8.57.0)

File Edit Sketch Tools Help



methane\_arduino

```

#include <SoftwareSerial.h>

SoftwareSerial nodemcu(8, 9);

int pinRedLed = 12;
int pinGreenLed = 11;
int pinSensor = A5; // Use A0 as the analog pin for the MQ4 sensor
int THRESHOLD = 250;
int buzzer = 10;

int rdata = 0;
String mystring;

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

    pinMode(buzzer, OUTPUT);
    pinMode(pinRedLed, OUTPUT);
    pinMode(pinGreenLed, OUTPUT);
    pinMode(pinSensor, INPUT);
}

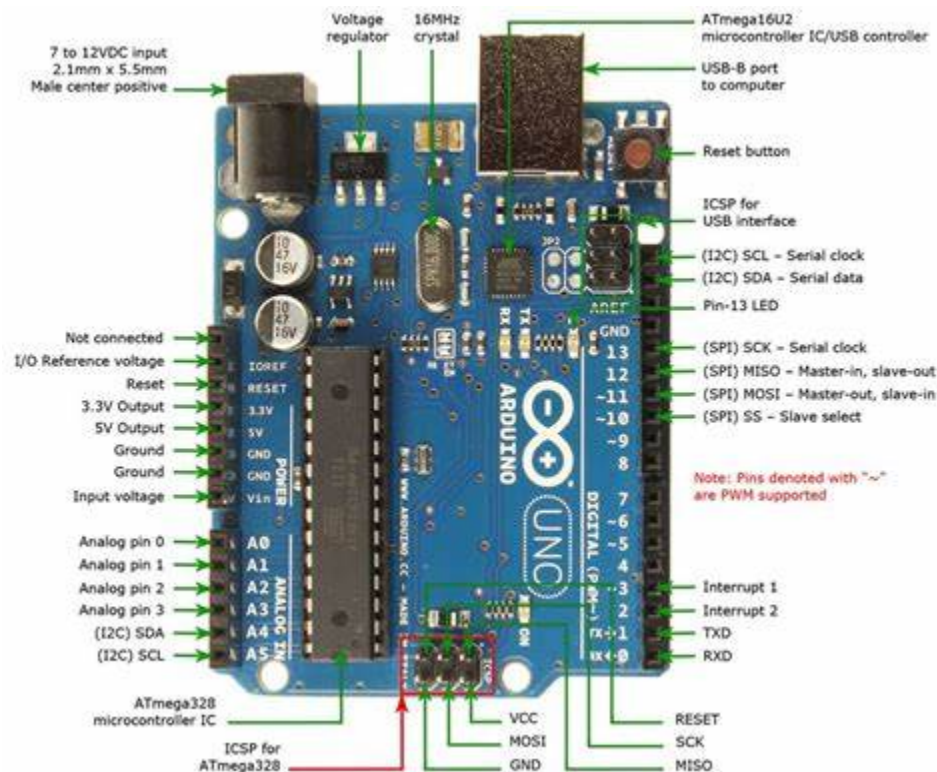
```

```
void loop() {  
  int rdata = analogRead(pinSensor);  
  
  Serial.print("Methane Range: ");  
  Serial.println(rdata);  
  
  if (rdata >= THRESHOLD) {  
    digitalWrite(pinRedLed, HIGH);  
    digitalWrite(pinGreenLed, LOW);  
    digitalWrite(buzzer, HIGH);  
    delay(50);  
  } else {  
    digitalWrite(pinRedLed, LOW);  
    digitalWrite(pinGreenLed, HIGH);  
    digitalWrite(buzzer, LOW);  
  }  
  
  if (nodemcu.available() > 0) {  
    char data;  
    data = nodemcu.read();  
    Serial.println(data);  
  }  
  
  if (rdata < 250) {  
    mystring = "Methane Range: " + String(rdata);  
    nodemcu.println(mystring);  
    Serial.println(mystring);  
  } else {  
    mystring = "Food Spoiled";  
    nodemcu.println(mystring);  
    Serial.println(mystring);  
  }  
  
  mystring = "";  
  delay(1000);  
}
```

---

## 2.8 Hardware Components

### 1) Arduino UNO R3:



The Arduino Uno R3 is a microcontroller board featuring a removable dual-inline-package (DIP) ATmega328 AVR microcontroller.

- Equipped with 20 digital input/output pins, of which 6 are suitable for PWM outputs and 6 can function as analog inputs.
- Loading programs onto the board is a seamless process using the user-friendly Arduino computer program.

### 2) MQ4 Sensor

- Specializing in detecting methane gas concentration in the air, this sensor provides readings in the form of analog voltage output.

- Tailored for leak detection, it covers a concentration sensing range from 300 ppm to 10,000 ppm.
- Operating efficiently in temperatures ranging from -10 to 50°C, the sensor consumes less than 150 mA at 5 V.



### 3) WIFI ESP8266 MODULE

- The ESP8266 is a low-cost Wi-Fi microchip, with a full TCP/IP stack and microcontroller capability with additional facilities..
- This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections



### 4) BREADBOARD

- A Breadboard is a commonly used tool for designing and testing circuits.
- Soldering is unnecessary when using a breadboard to create circuits.
- Mounting and reusing components is simplified.
- The absence of soldering allows for easy modification of circuit designs at any stage.



## 5)USB:

The USB port in the Arduino board is used to connect the board to the computer using the USB cable. The cable acts as a serial port and as the power supply to interface the board.



## 2.9 Implementation

Upon installation in a food store, this Arduino-based FSD system seamlessly connects to the internet through the Wi-Fi module upon powering up. It begins reading data from the interfaced MQ4 Sensor, which is adept at detecting methane-type gases emitted when food or fruits undergo spoilage. As the MQ4 sensor identifies the concentration of these gases, it produces an analog voltage proportionate to the gas concentration. The Arduino, equipped with an inbuilt ADC, then converts this analog output to a digital value.

The Arduino, acting as a central hub, gathers and transforms the sensor data into strings. These data-laden strings are transmitted via the Wi-Fi module to a server for further processing. The server utilizes a predefined algorithm, comparing the received values to threshold levels to determine the freshness status of the food. The processed results are then relayed back to the Arduino, which in turn displays the output on the Blynk app dashboard. Depending on the methane content and freshness level, the Blynk app signals whether the food is spoiled, additionally providing information on the range of methane content in parts per million (ppm).

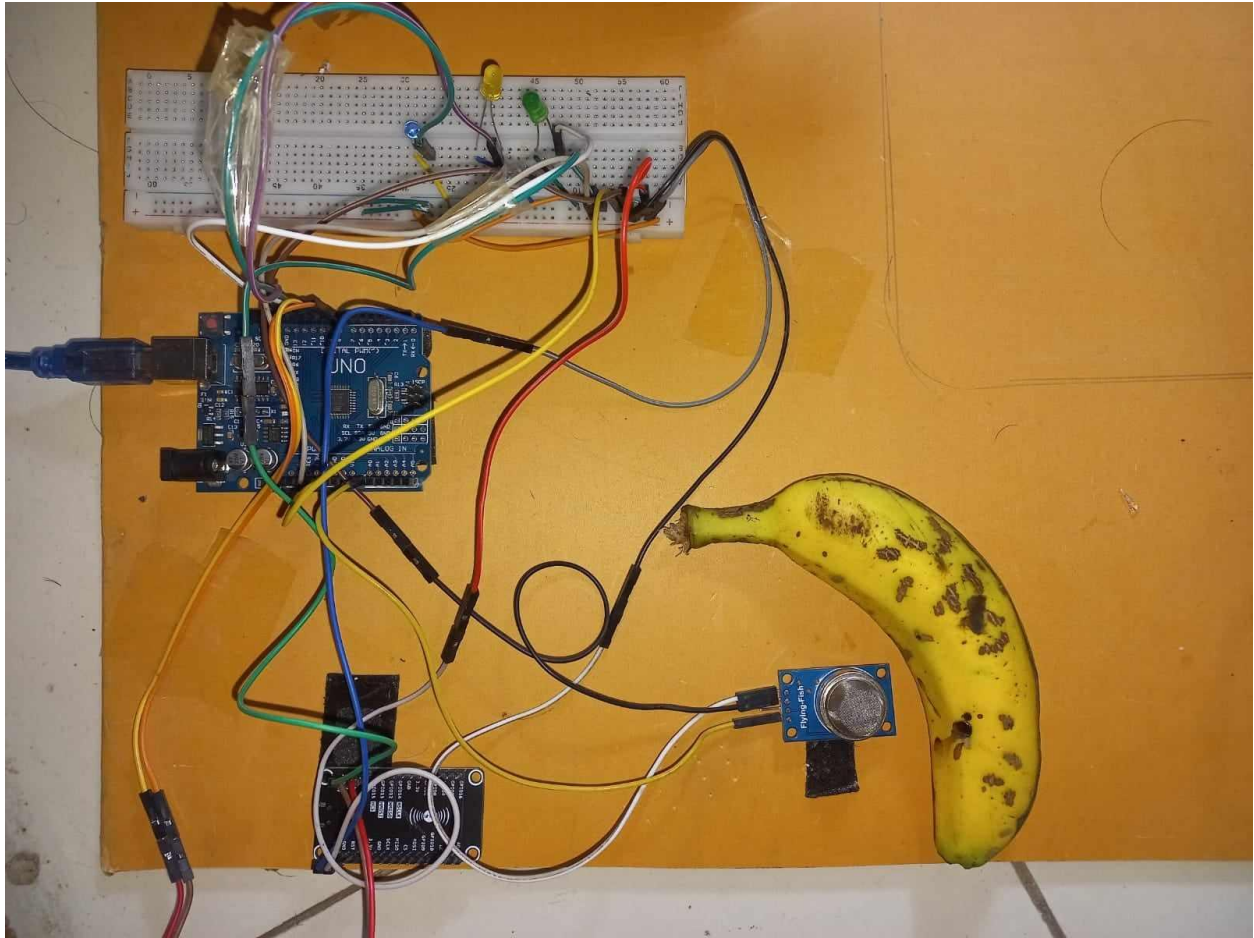


Fig: Hardware Implementation

## 2.10 Process

At first the agent will login to Blynk IOT platform via email credentials . At Blynk there are some template ID, template name, and authentication token. This token is instrumental in programming the NodeMCU, facilitating its connection to the Blynk server. Agent needs to include these information to the nodemcu code, for connecting the wifi module to the Blynk IOT platform.

In the subsequent steps, the customer configures the NodeMCU by specifying the Wi-Fi SSID and password in the requisite program, establishing connectivity with the FSD System and the internet. The agent, equipped with the Blynk app, gains access to



comprehensive reports, including the methane range sensed by the sensor in fruits. Notably, the agent is promptly alerted with a "Food Spoiled" message when the detected range exceeds 250 ppm.

Upon receiving notifications on the Blynk app dashboard, the manager is expected to take immediate actions concerning spoiled food, ensuring swift response to maintain food safety. Additionally, fruits at the early stages of spoilage warrant attention. The system requirements for monitoring and managing the FSD System include a hardware setup with either a PC/laptop boasting a 2 GB hard disk and 256 MB RAM or an Android phone, which can serve as the power source for the Arduino. This ensures a robust and efficient operation of the food spoilage detection system.



The FSD System is initially indicated by a glowing green LED, signifying normal operation until it encounters spoiled food emitting methane, particularly fruits. Upon sensing spoiled fruit, the system activates a buzzer and switches on a red LED for immediate notification. The methane concentration, measured in parts per million (ppm) by the sensor, is prominently displayed on the Blynk app dashboard. Additionally, leveraging the ESP8266 Wi-Fi module allows users to access this specific methane range through various IoT project-based platforms such as



Thingspeak.com or applications like Blynk and other clouds serving as IoT portals.

These platforms provide a visual representation of the methane levels associated with spoiled fruits. Typically, fresh fruits exhibit methane content below 250 ppm. Any reading surpassing this threshold serves as a clear indication of fruit spoilage. Utilizing IoT project-based applications, a distinct "Food Spoiled" message is promptly displayed on the Blynk app screen, facilitating real-time monitoring and response to potential food spoilage events.

### 3. Automatic Rain Shield System

In modern agriculture, technology plays a pivotal role in enhancing efficiency and overcoming environmental challenges. The proposed Rain Sensor and Shield System respond to the dual nature of rainfall, both a vital resource and a potential threat. This innovative system integrates advanced sensors to detect rain in real-time and autonomously deploys a protective shield over plants. Beyond optimizing plant health, this technology reflects a dedication to sustainable farming practices.

The Rain Sensor and Shield System signify a paradigm shift from traditional weather-dependent crop protection methods. By harnessing sensor technology and automation, the system offers an intelligent solution to shield plants from excessive rain. The outcomes include improved crop yields, minimized agricultural losses, and a proactive step toward eco-friendly farming practices. This preamble introduces the transformative essence of the system, paving the way for a deeper exploration of its problem statement and development objectives.

#### 3.1 Problem Statement

The vulnerability of crops to unpredictable weather, particularly excessive rainfall, is a pressing concern. Issues such as soil erosion, waterlogging, and heightened susceptibility to diseases arise from traditional, manually dependent methods of protecting crops. These approaches are often inefficient and lack timeliness.

The necessity for an automated and responsive solution is clear. The Rain Sensor and Shield System aims to bridge this gap by detecting rainfall in real-time and autonomously deploying a protective shield over plants. This not only shields crops from the detrimental effects of excess rain but also eliminates the reliance on manual intervention, ensuring a proactive and efficient response to weather-related challenges in agriculture.

#### 3.2 Feature Objective

Our core objective revolves around providing comprehensive protection for plants, particularly during unforeseen rain and storm events. Simultaneously, we aim to minimize the manual intervention needed when abrupt weather changes occur. To fulfill this goal, we have developed an innovative solution – an automatic rain shield. This cutting-edge system is designed to detect rainfall promptly through its built-in sensors. Upon sensing rain, the system initiates an immediate response by automatically opening a protective cover over the plants. This swift and automated

deployment ensures that the plants are shielded from the unexpected precipitation and adverse weather conditions.

Furthermore, the intelligence embedded in the system doesn't just stop at opening the shield. Once the rain subsides, and the environmental conditions revert to a favorable state, the system seamlessly engages in an automatic closure of the shield. This intelligent and responsive mechanism not only safeguards the plants during adverse weather but also significantly reduces the need for manual intervention, allowing for a more efficient and hassle-free approach to managing sudden weather shifts.

### 3.3 Module Summary

For implementing the rain shield module we have used the following components:

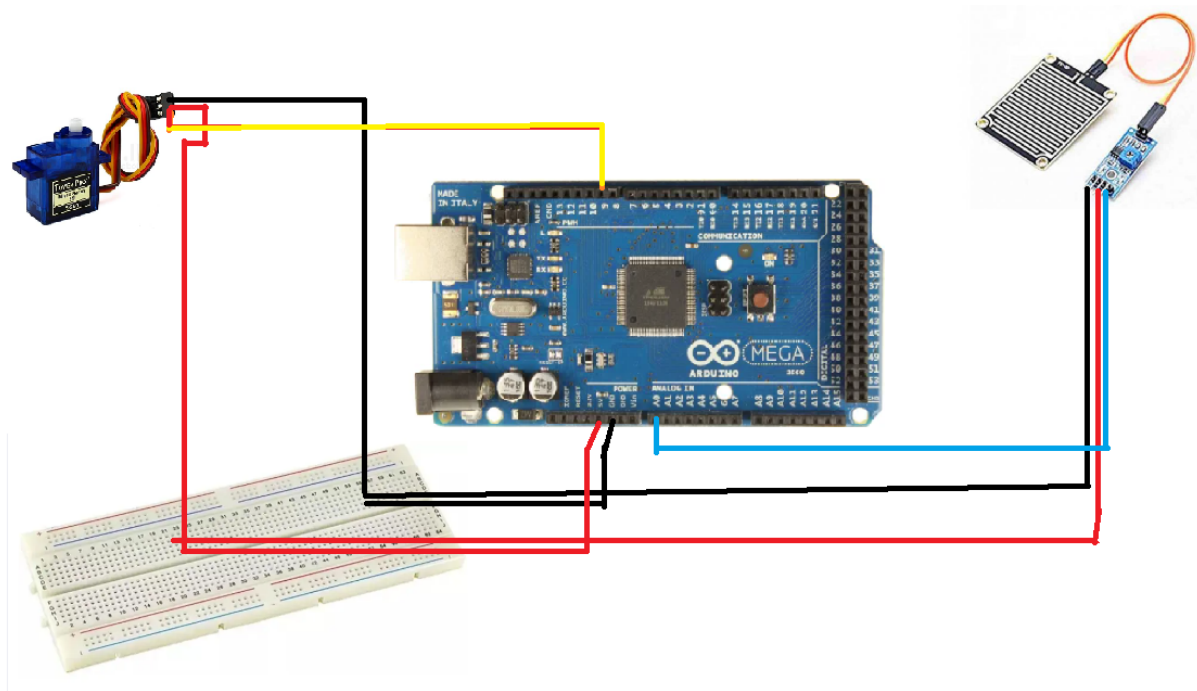
#### Key components

- **Arduino Mega** : It has been used to compile and run the code as well as we have used it as the power source of our module
- **Rain sensor** : A rain sensor is a device designed to detect and measure the presence of rain or precipitation, typically used to trigger automated systems or responses in various applications such as irrigation control, vehicle wipers, or weather monitoring
- **Servo motor** : In order to build the automatic opening and closing of the shield we have used the servo motor, which rotates at different angles according to the reading from the rain sensor.

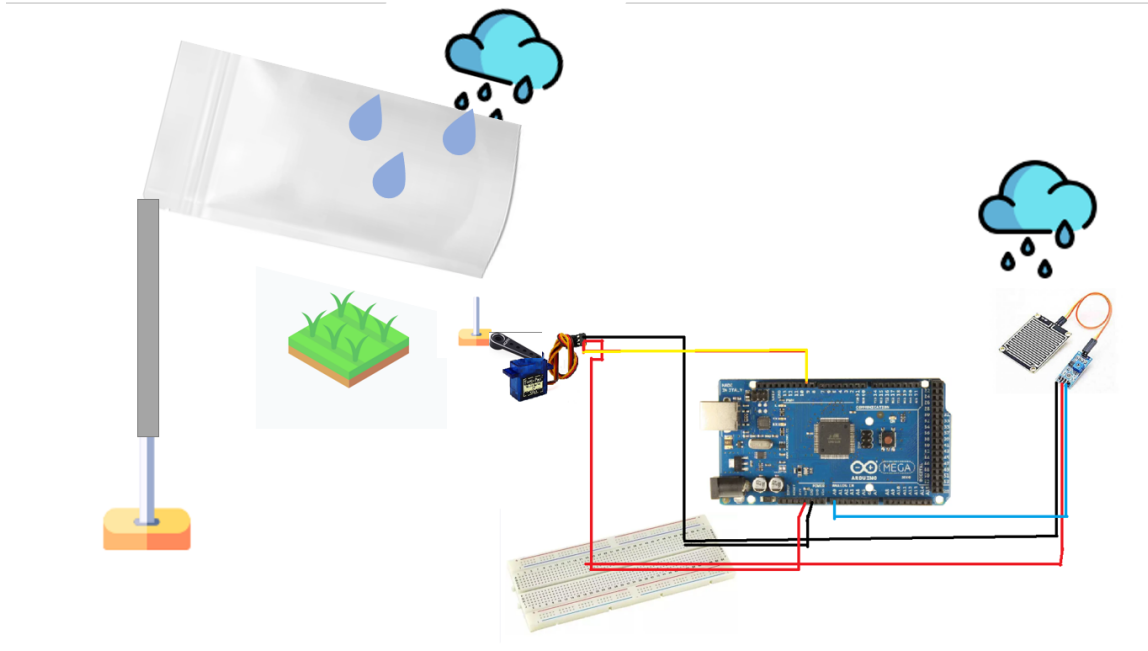
#### Other components :

- Bread Board
- Jumpers
- Transparent Shield paper
- Wooden base
- Straws to build the poles

### 3.4 Circuit Diagram



### 3.5 Steps & Process

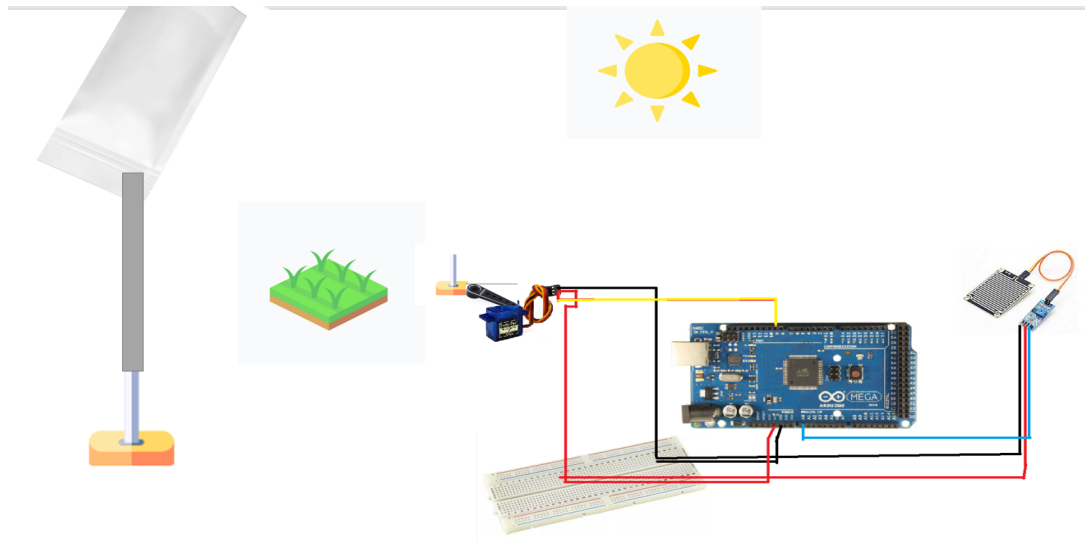


The rain detector functions through a carefully orchestrated process, ensuring reliable protection against precipitation. When the rain sensor identifies the presence of rain, it records a decrease in its analog reading, setting off a sequence of actions to safeguard the designated area.

Upon rain detection, the rain sensor initiates the activation of a servo mechanism. This process involves the analog pin receiving a signal, which serves as a trigger for the motor driver. The motor driver, in turn, interprets and responds to this triggering output received from Arduino pin 9. Once the signal reaches the motor driver, it promptly engages in rotation, spanning an angle between 0 and 180 degrees.

This rotational movement of the motor is pivotal, as it directly influences the positioning of the rain shield. The rain shield, intricately connected to the servo mechanism, responds to the rotational input by smoothly opening to cover and shield the designated area from the falling precipitation.

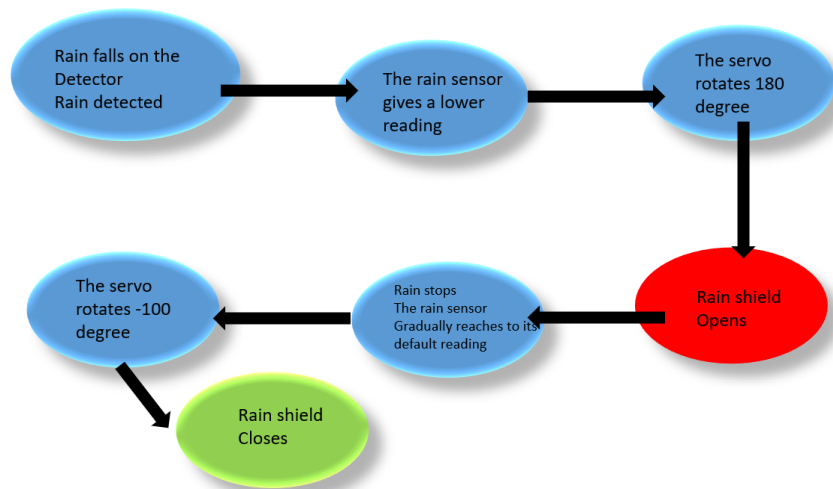
In essence, the rain detector's systematic operation involves a cascade of events triggered by the rain sensor, signaling the servo mechanism and motor driver to collaboratively orchestrate the controlled movement of the rain shield. This coordinated process ensures that the protective shield efficiently opens in response to rain, providing effective coverage and safeguarding the intended area from the adverse effects of precipitation.



As precipitation persists, the rain shield is intentionally maintained in the open position to effectively safeguard against the rain. Throughout this period, the rain sensor constantly monitors the weather conditions, and as the rainfall diminishes, the sensor gradually reverts to its default state. This change in the sensor reading triggers a systematic response whereby the servo motor smoothly rotates back to its initial position, ranging from 0 to -100 degrees. Consequently, this prompts the rain shield to undergo a controlled and gradual closure.

This operational sequence is designed with precision to ensure that the rain shield is deployed judiciously based on real-time weather conditions. It seamlessly opens during rain events, providing protection, and gracefully closes as the weather clears. The gradual adjustments made by the rain sensor and servo, coupled with the automated opening and closing mechanism of the rain shield, collectively exemplify an efficient and responsive system. This system effectively mitigates the need for manual intervention in managing sudden weather changes, streamlining the overall process and enhancing the system's adaptability to varying environmental conditions.

### 3.6 Step By Step



## 3.7 Arduino Code

```
sketch_bothsideside | Arduino IDE 2.2.1
File Edit Sketch Tools Help

sketch_bothsideside.ino
1  #include<Servo.h>
2  int rain_sensor =A0,servo =9;
3  Servo myServo;
4  int rainDetected = 0;
5  void setup ()
6  {
7  Serial.begin (9600);
8  myServo.attach(servo);
9  myServo.write(0);
10 }
11
12 void loop() {
13   int sensorValue = analogRead(rain_sensor); // Read the analog value from the rain sensor
14   int motor;
15
16   // Check rain status
17   if (sensorValue > 800) {
18     if (rainDetected == 0) {
19       // Rain is detected, rotate in one direction
20       motor = map(sensorValue, 1019, 800, 100, 180);
21       rainDetected = 1;
22     }
23   }
24 }
```

```
File Edit Sketch Tools Help
sketch_bothside.ino
17 { sensorValue > 800 } {
18   if (rainDetected == 0) {
19     // Rain is detected, rotate in one direction
20     motor = map(sensorValue, 1019, 800, 100, 180);
21     rainDetected = 1;
22   }
23 } else {
24   if (rainDetected == 1) {
25     // Rain has dried out, rotate in the opposite direction
26     motor = map(sensorValue, 800, 0, 100, 0);
27     rainDetected = 0;
28   }
29 }
30
31 myServo.write(motor); // Set the servo motor angle
32 Serial.println("Sensor value is ");
33 Serial.println(sensorValue);
34 Serial.println("Servo motor rotates by angle ");
35 Serial.println(motor);
36
37 delay(1000); // Delay for 1 second before reading the sensor again
38
39 }
```

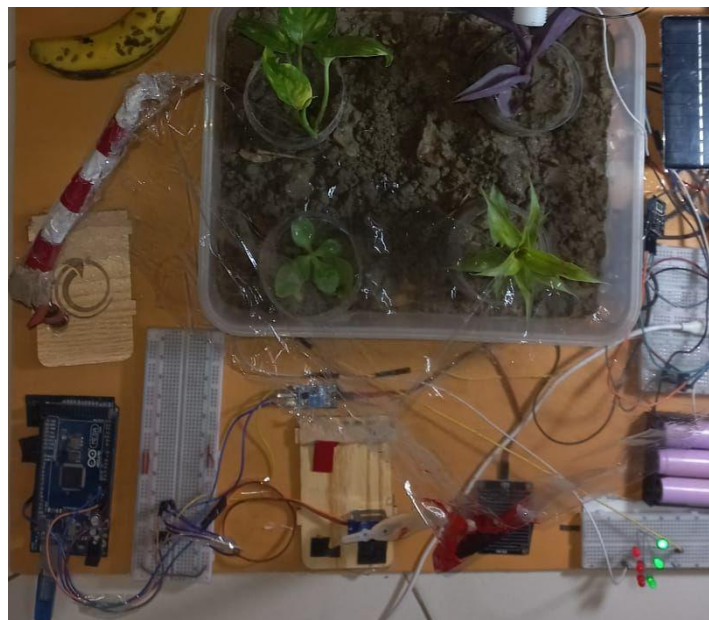
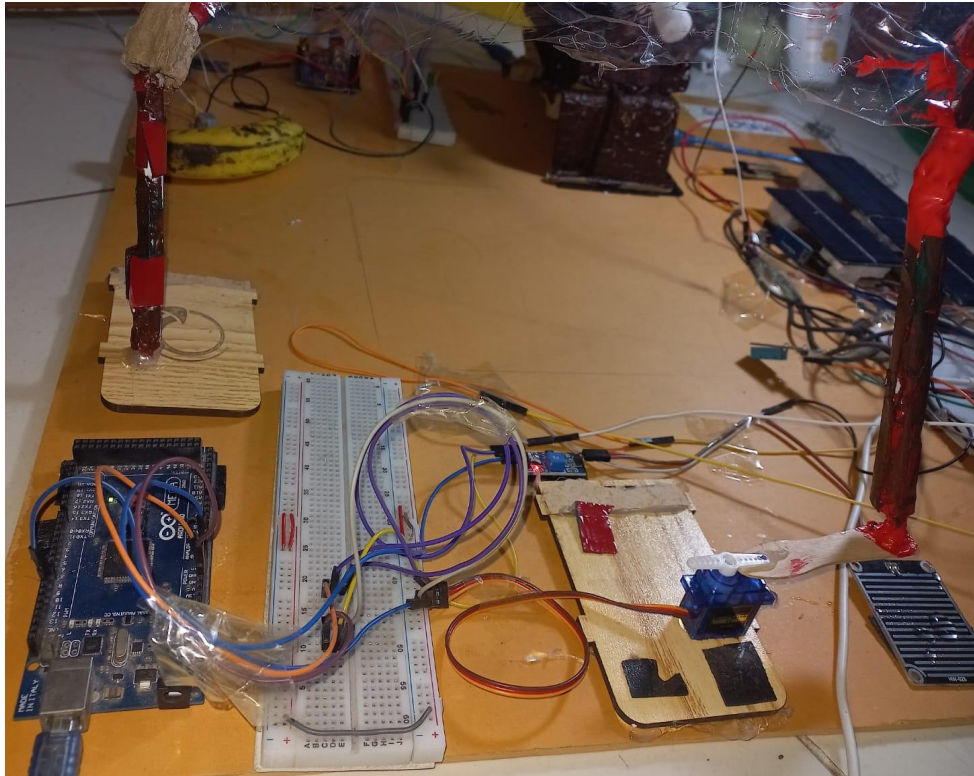
## 3.8 Serial Monitor Reading

```
Output Serial Monitor x
[Message (Enter to send message to 'Arduino Mega or Mega 2560' on 'COM9')]
Servo motor rotates by angle 37
Sensor value is 654
Servo motor rotates by angle 37
Sensor value is 676
Servo motor rotates by angle 37
Sensor value is 681
Servo motor rotates by angle 37
Sensor value is 682
Servo motor rotates by angle 37
Sensor value is 682
Servo motor rotates b

Output Serial Monitor x
[Message (Enter to send message to 'Arduino Mega or Mega 2560' on 'COM9')]
Servo motor rotates by angle 100
Sensor value is 1014
Servo motor rotates by angle 172
Sensor value is 1013
Servo motor rotates by angle 172
Sensor value is 1012
Servo motor rotates by angle 172
Sensor value is 1013
Servo motor rotates by angle 172
Sensor value is 1013
Servo motor rotates by angle 172
```



### 3.9 Implemented Circuit Diagram



## 4. Renewable Energy Generation

The objective of the renewable energy generation module is to offer continuous power supply to all other modules of the system, hence ensuring the use of clean zero-carbon energy.

### 4.1 Specifications

The properties of this system are as follows:

- The module consists of 3 solar panels.
- They are connected serially.
- They provide a collective output of 13-14 V.

### 4.2 Working Procedure

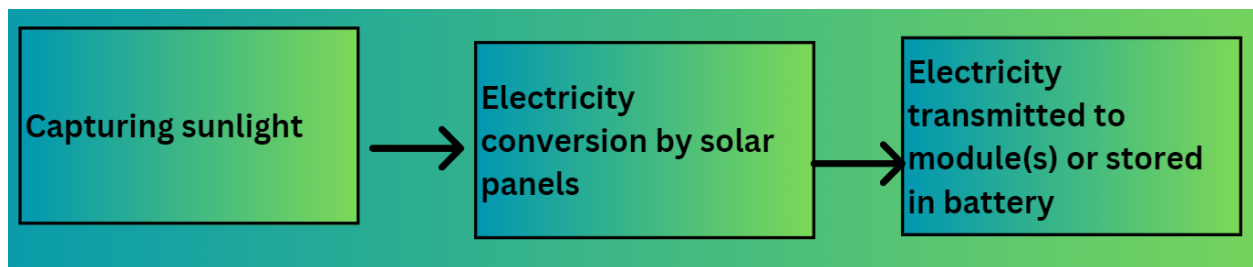


Fig: Flowchart depicting the working procedure

### 4.3 Hardware Implementation



## Cost Report

Sl No.	Name of Component	Price (in BDT)
1	Soil Sensor	80
2	Solenoid Valve	420
3	2x ESP 8266	560
4	2x DHT11 Sensors	220
5	2x 5V Relay Modules	110
6	2x 12 V Relay Modules	130
7	5x 14500 Batteries	200
8	3x 5V Mini Solar Panels	450
9	2x XL6009	150
10	Rain Sensor	70
11	9205	350
12	Jumper Cables	120
13	SG-90 Servo Motor	110
14	LED	10
15	MQ4 Sensor	100
16	9 V Charger	270
17	PVC Board	350
18	Miscellaneous	1000
		Total= 4700

## Conclusion



Fig: Whole Project's Hardware Implementation

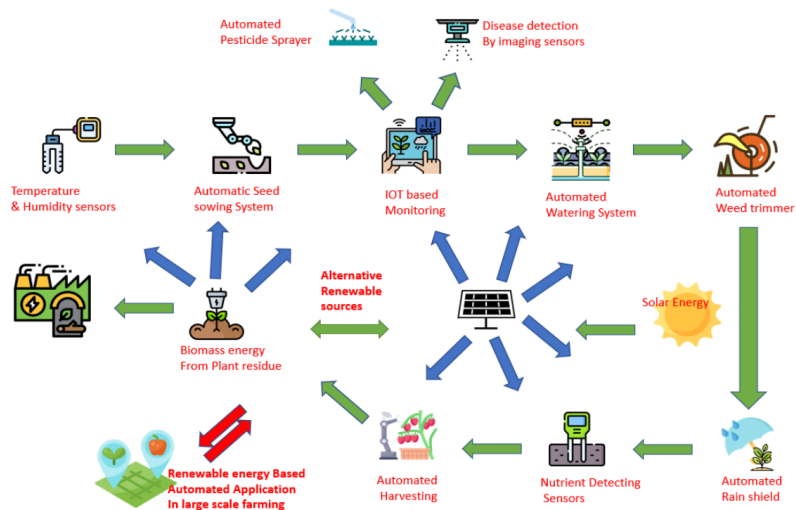


Fig: Visionary Blueprint for Scaling Up Our Project: Elevating Content Rephrasing to Unprecedented Levels

So, from small attempts of rearranging an automated gardening system energized by renewable sources, much larger attempts can proceed in techno-based farming. In fact, smart and fully automated farming with high efficiency can be ensured in the near future. And further development on such automated systems will ensure not only minimum labor and time consumption but also can play a great initiative role to minimize energy crisis.