A PROJECT REPORT

on

“Smart Irrigation System”

Submitted by

Syeda Nowshin Ibnat(ID: 17183103020)

Mahmuda Begum(ID: 17183103030)

Nusrat Jahan Anka(ID: 17183103008)

Nawrin Zaman Prova(ID: 17183103044)

Supervised by

Md. Hasibur Rahman
Lecturer
Department of CSE, BUBT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY (BUBT)
DHAKA-BANGLADESH

SUMMER 2021

08 December, 2021

DECLARATION

We hereby declare that the project entitled “Smart Irrigation System” submitted for the IoT Lab project (CSE 426) work in the semester, Summer 2021 in the faculty of Computer Science and Engineering of Bangladesh University of Business and Technology (BUBT) is our original work and that contains no material which has been accepted for the award to the candidates of any other degree or diploma, except where due reference is made in the next of the project to the best of our knowledge, it contains no materials previously published or written by any other person except where due reference is made in this project.

Syeda Nowshin Ibnat
Id: 17183103020
Intake: 39th

Mahmuda Begum
Id: 17183103030
Intake: 39th
Section-01

Nusrat Jahan Anka
Id: 17183103008
Intake: 39th

Nawrin Zaman Prova
Id: 17183103044
Intake: 39th

APPROVAL

This project titled “Smart Irrigation System” report was submitted by, Syeda Nowshin Ibnat, Mahmuda Begum, Nusrat Jahan Anka, and Nawrin Zaman Prova students of the Department of Computer Science and Engineering, Bangladesh University of Business and Technology (BUBT), under the supervision of Md. Hasibur Rahman, Lecturer, Department of Computer Science and Engineering has been accepted as satisfactory for the partial requirements for the degree of Bachelor of Science Engineering in Computer Science and Engineering.

Md. Hasibur Rahman
Lecturer
Department of CSE, BUBT

ACKNOWLEDGEMENTS

We would like to thank the following people for their help in the production of this project Md. Hasibur Rahman, project supervisor for all of his ongoing assistance with the project, without whose help and support throughout, this project would not have been possible. And we would like to thank the CSE department faculty members for their support and guidance which helped us to nurture our knowledge and skill.

ABSTRACT

Nowadays, technology is playing a tremendous role in serving humanity, and food is, without a doubt, a person's most fundamental and primary requirement. More over 85 percent of Bangladesh's population is dependent on agriculture, either directly or indirectly. Due to frequent power outages, the lack of grid lines in rural places, and the scarcity/cost of fuel to drive pumps, proper irrigation by water pump is impossible to maintain. This IOT-based smart irrigation system is designed to provide a sustainable irrigation system and a intrusion detection for better crop development and productivity. The irrigation system is controlled and monitored using IOT and Wi-Fi in this system. IoT is utilized to collect both stored data and real-time monitoring of diverse soil contents. It is a totally wireless system that is user-friendly and properly irrigating water to the field. Sensors of several types are utilized. The "Thinkspeak Cloud Server" is used to control and monitor. The soil's moisture level is constantly monitored. There is a app for this for monitoring moisture value, weather precipitation, and IR value. The system gives the user freedom to take decision to turn the pump On/Off.

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 Introduction

Agriculture is a major source of income for Bangladesh's largest population and a major contribution to the nation's economy. However, technology engagement and usability in the agro-industry in Bangladesh must continue to be grown and developed. With the inclusion of smart irrigation systems, agricultural practices can be made more efficient. IoT is transforming agriculture and allowing farmers to overcome the enormous challenges they face. Agriculture must address growing water shortages and limited land availability while fulfilling the growing consumption demands of a global population. These concerns are being addressed by new innovative IoT applications that improve the quality, quantity, sustainability, and cost-effectiveness of agricultural produce. In our system, the moisture level in the soil is monitored and the IR sensor is used for intrusion detection. The user can turn on/off the motor for irrigation. Smart irrigation systems reduce the chances of over-watering the soil and waterlogging in the fields. Thus water conservation is possible due to smart irrigation systems. In this project, we used ESP32 Node MCU, ThingSpeak IoT cloud and we made an app using MIT APP Inventor.

1.2 Problem Background

Since the dawn of civilization, humans have used numerous techniques to produce food and large-scale crops were required. As a result, irrigation was a crucial task for success. However, the main issue occurred when insufficient human labor slowed agricultural development. As a result, when automation and IoT became a reality, this problem had only one solution. To develop irrigation that would reduce human labor and increase output speed and efficiency. In our system even if could not do automation but this system can be helpful to reduce human labor and can be useful to save water.

1.3 Project Objectives

- To develop an IoT-based Smart irrigation system having low-cost equipment.
- To monitor moisture contents at different conditions.
- To get accurate IR sensor value.
- To improve the system by storing data on the cloud and using Mobile Phone App.
- To do proper circuit design for the project.
- To do proper testing and debugging of the device configuration.

1.4 Motivations of the Project

Bangladesh's economy is based on agriculture. Agriculture is becoming more crucial to support the demands of the human race in today's globe, as we see tremendous growth in the worldwide population. Agriculture, on the other hand, necessitates irrigation, and as we consume more water each year than we receive in rainfall, growers must find ways to conserve water while still achieving the best yield. Farmers, on the other hand, have started employing irrigation techniques in recent years. We are motivated to develop a system by which farmers can get soil moisture value, can detect intrusion. By using the app farmers can also turn the pump on or off.

1.5 Project Contributions

The Internet of Things (IoT) is a technology where in a mobile device can be used to monitor the function of a device and now IoT using in agro-tech wildly for development. We aimed to develop a system to get accuiented with IoT and agro-tech .Our system has been designed to overcome the unnecessary water flow into the agricultural lands. Moisture and IR readings are continuously monitored and send these values to Cloud and android application continuously shows data.

CHAPTER 2

BACKGROUND

2.1 Existing Systems

There are some existing system which use arduino based remote irrigation system developed for the agricultural plantation, which is placed at the remote location and required water provides for plantation when the humidity of the soil goes below the set-point value. But in this system we used ESP 32 Node MCU to develop our system. Use of ESP 32 makes our system economically feasible than the existing system.

2.2 Problem Analysis

IoT has been a burgeoning concept for decades. Wireless sensor network (WSN) system has been proven to be highly useful for irrigation control in terms of water conservation. WSN is a system that consists of a mesh network of sensor nodes that are connected to one another, and the nodes immediately gather data from the environment and deliver real-time data to farmers, which is extremely beneficial to them. This system can be utilized as a data collection device as well as a decision-making tool for real-time monitoring. Farmers are aware that there is a water constraint, and that overwatering can harm the output. They must be aware of when and how much water is required for certain crops. The majority of farmers have minimal knowledge of their farm and are uninformed of how to increase their farm's output through agricultural practices. All of these conflicts necessitate the development of agricultural support systems.

CHAPTER 3

DEVELOPMENT

3.1 Feasibility Analysis

A feasibility analysis evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. Our project includes technical feasibility, economic feasibility, Organizational feasibility.

3.1.1 Technical Feasibility

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. In our project, the user needs android devices. Besides, the system needs an internet connection. It is also not an issue in this era where almost every home or office has Wi-Fi.

3.1.2 Economic Feasibility

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. Here, we find the total cost and benefit of the proposed system over the current system. For this project, the main cost is that user have to by the required hardware components.

3.1.3 Organizational feasibility:

This shows the management and organizational structure of the project. This project is built by a team or group. The management tasks are all to be carried out by the team and it won't create any management issues and increase the feasibility of the project.

3.2 Requirement Analysis

This system is being built keeping in mind the generally available hardware and software compatibility.

3.2.1 Hardware Requirements

No	Hardware	Unit(s)	Specification
1.	Breadboard	1	To set up the components.
2.	Jumper Wire	1	To make connection.
3.	Resistor	1	To control current flow.
4.	USB Cable	2	Power Supply.
5.	ESP 32 - 3.3V	1	System on Chip (Bluetooth and WiFi module integrated).
6.	Soil Moisture Sensor- 5V	1	Measure soil moisture.
7.	IR Sensor- 5V	2	Intrusion detection.
8.	Relay - 5V	1	Used as a switch for pump.
9.	Pump - 5V	1	Used for irrigation.

Table 3.1

ESP 32:

Espressif Systems, the makers of the ESP8266 SoC, have introduced the ESP32, a low-cost System on Chip (SoC) Microcontroller. Tensilica's 32-bit Xtensa LX6 Microprocessor with integrated Wi-Fi and Bluetooth is the successor to the ESP8266 SoC and is available in single-core and dual-core varieties. The ESP32, like the ESP8266, has built-in RF components such as a Power Amplifier, a Low-Noise Receive Amplifier, an Antenna Switch, Filters, and an RF Balun. This makes designing hardware for the ESP32 platform relatively simple because we only need a few external components.

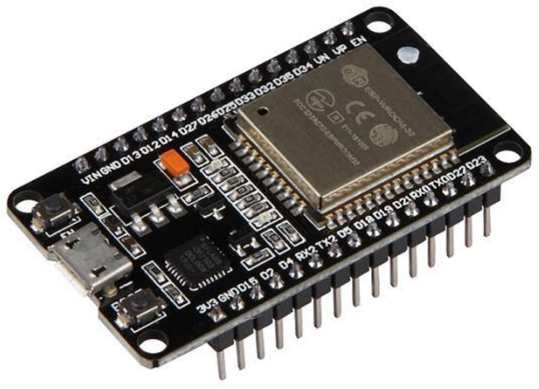


Figure1: ESP 32 Chip

There are many ESP32 Boards based on ESP-WROOM-32 Module available in the market. The layout, pinout and features vary from board to board.The board which we have has 30 Pins (15 pins on each side). There are some board with 36 Pins and some with slightly less Pins. So, double check the pins before making connections or even powering up the board.

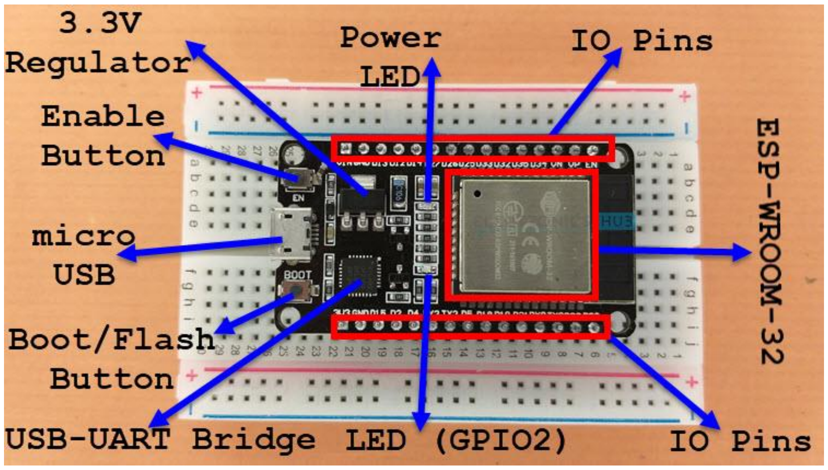


Figure2: Pin Layout of ESP 32
Curtesy: electronicshub.org

The ESP32 Board consists of the following:

- **ESP-WROOM-32 Module**
- **Two rows of IO Pins (with 15 pins on each side)**
- **CP2012 USB – UART Bridge IC**
- **micro–USB Connector (for power and programming)**
- **AMS1117 3.3V Regulator IC**
- **Enable Button (for Reset)**
- **Boot Button (for flashing)**
- **Power LED (Red)**
- **User LED (Blue – connected to GPIO2)**
- **Some passive components**

Soil Moisture Sensor:

The soil moisture sensor module is used to detect the moisture of the soil. It measures the volumetric content of water inside the soil and gives us the moisture level as output. The module has both digital and analog outputs and a potentiometer to adjust the threshold level.

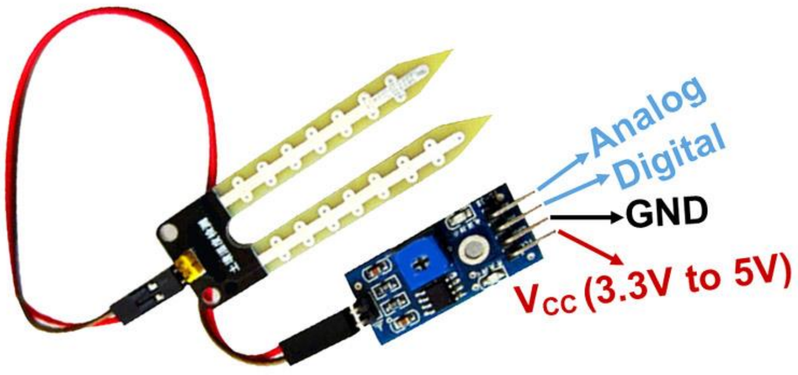


Figure3: Soil Moisture Sensor Module
Curtesy: components101.com

Soil Moisture Sensor Module Pinout Configuration

Pin Name	Description
VCC	The Vcc pin powers the module, typically with +5V.
GND	Power Supply Ground.
DO	Digital Out Pin for Digital Output.
AO	Analog Out Pin for Analog Output.

Table 3.2

Soil Moisture Sensor Module Features & Specifications

- Operating Voltage: 3.3V to 5V DC
- Operating Current: 15mA
- Output Digital - 0V to 5V, Adjustable trigger level from preset
- Output Analog - 0V to 5V based on infrared radiation from fire flame falling on the sensor
- LEDs indicating output and power
- PCB Size: 3.2cm x 1.4cm
- LM393 based design
- Easy to use with Microcontrollers or even with normal Digital/Analog IC
- Small, cheap and easily available

IR Sensor:

An infrared sensor is an electronic device, that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. These types of sensors measure only infrared radiation, rather than emitting it that is called a passive IR sensor. Usually, in the infrared spectrum, all the objects radiate some form of thermal radiation.

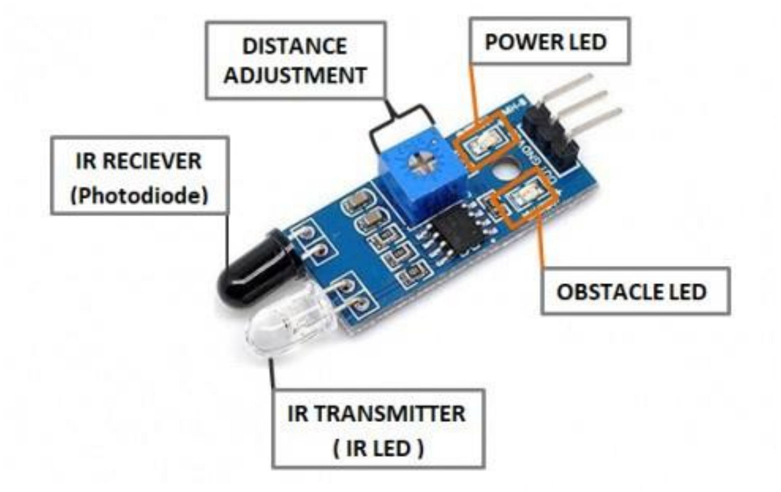


Figure4: IR Sensor Module

IR Moisture Sensor Module Pinout Configuration

Pin Name	**Description**
VCC	Power Supply Input
GND	Power Supply Ground
OUT	Active High Output

Table 3.3

IR Sensor Module Features

- 5VDC Operating voltage
- I/O pins are 5V and 3.3V compliant
- Range: Up to 20cm
- Adjustable Sensing range
- Built-in Ambient Light Sensor
- 20mA supply current
- Mounting hole

Relay:

Relay is an electromechanical device that uses an electric current to open or close the contacts of a switch. The single-channel relay module is much more than just a plain relay, it comprises of components that make switching and connection easier and act as indicators to show if the module is powered and if the relay is active or not.

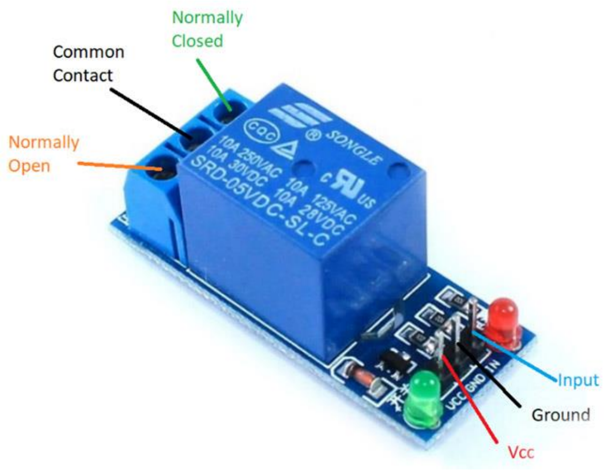


Figure5: Single-Channel Relay Module Pinout
Curtesy: components101.com

Single-Channel Relay Module Pin Description

Pin Number	Pin Name	Description
1	Relay Trigger	Input to activate the relay.
2	Ground	0V reference.
3	VCC	Supply input for powering the relay coil.
4	Normally Open	Normally open terminal of the relay.
5	Common	Common terminal of the relay.
6	Normally Closed	Normally closed contact of the relay.

Table 3.4

Single-Channel Relay Module Specifications

- Supply voltage – 3.75V to 6V
- Quiescent current: 2mA
- Current when the relay is active: ~70mA
- Relay maximum contact voltage – 250VAC or 30VDC
- Relay maximum current – 10A

Pump:

It’s a Low voltage water pump Using 5V voltage. There are many 5V water pumps available on the market, from the cheaper to more expensive, but there are basically two options of 5V water pumps. The pump that we used are cheap and useful for prototyping but they wont last very long under intensive work load, so they are good for prototyping and projects that require watering from time to time and not a continuous water flow.

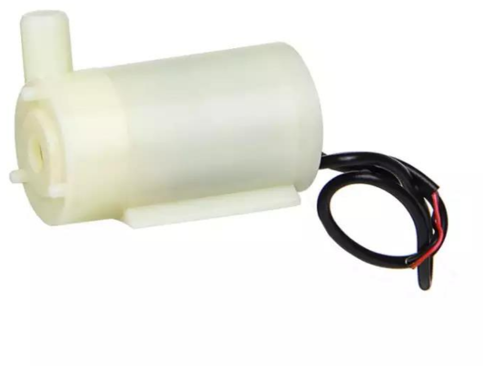


Figure6: Micro Water Pump

- Voltage: CC 3-5 V.
- Operating current: 100-200mA.
- Load power: 0,4-1,5 W.
- Max height: 40-110 cm/15,75 “-43,4”.
- Water flow: 80-120L/H.
- Continuous working life: 500 hours.

3.2.2 Software Requirements

No	Code Development	Language used	Cloud Server	Mobile App
1.	Arduino IDE	Arduino C	ThingSpeak	MIT App Inventor

Table 3.5

3.2.2 Network Connection

ESP32 - Wifi Communication: The ESP32 board can act as Wi-Fi Station, Access Point, or both. To set the Wi-Fi mode, we need to use WiFi.mode() and set the desired mode as an argument. When the ESP32 is set as a Wi-Fi station, it can connect to other networks (like routers). In this scenario, the router assigns a unique IP address to our ESP board. We can communicate with the ESP using other devices (stations) that are also connected to the same network by referring to the ESP unique IP address.

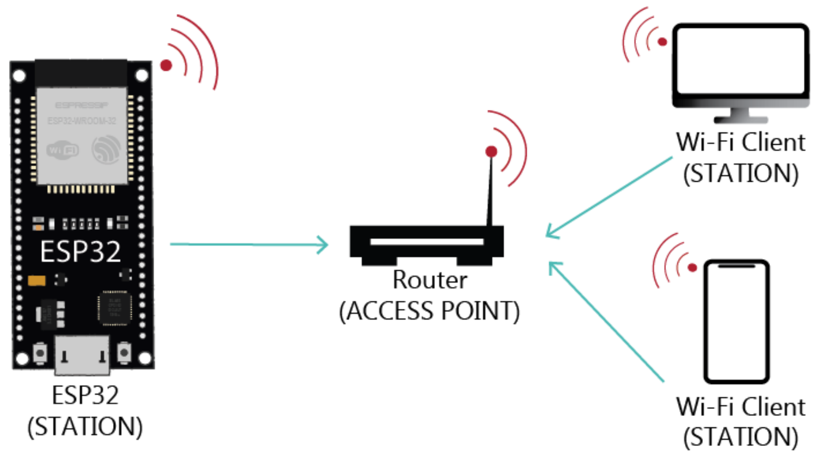


Figure7: ESP 32 WiFi Station

In some cases, this might not be the best configuration – when we don’t have a network nearby and we still want to connect to the ESP to control it. In this scenario, we must set the ESP board as an access point.

3.3 System Architecture

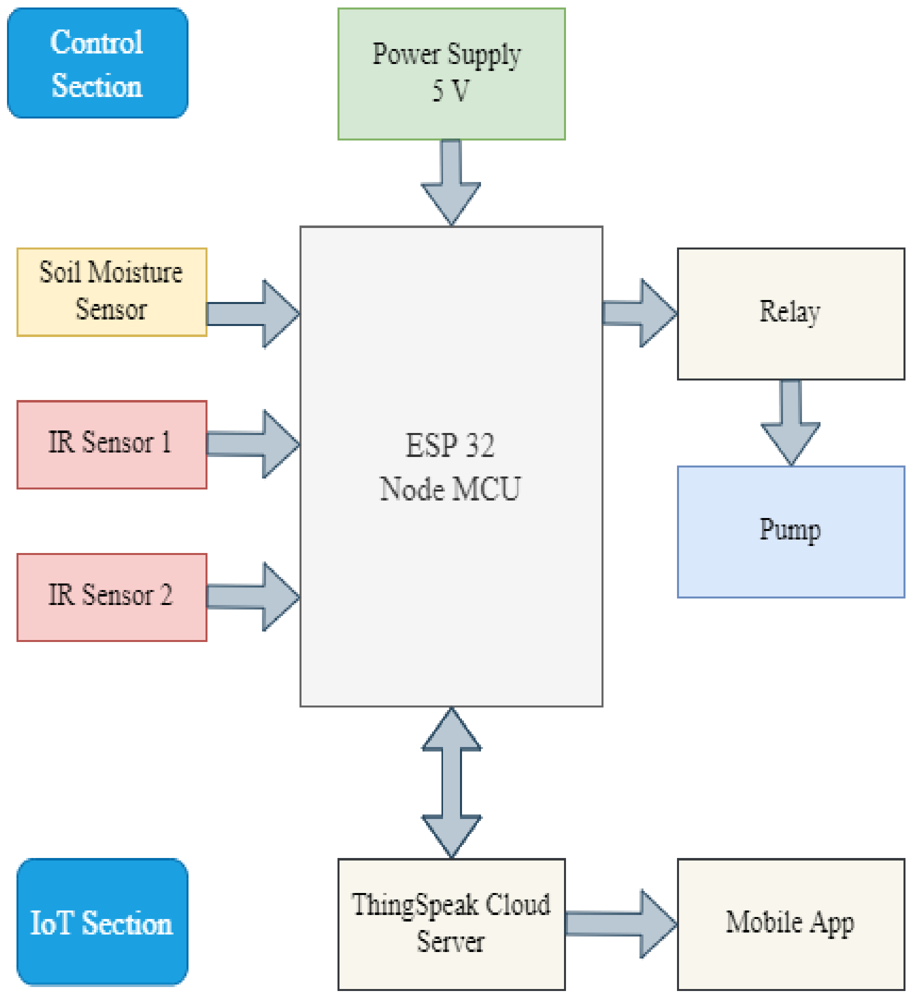


Figure8: Block Diagram of Smart Irrigation System

3.4 System Analysis

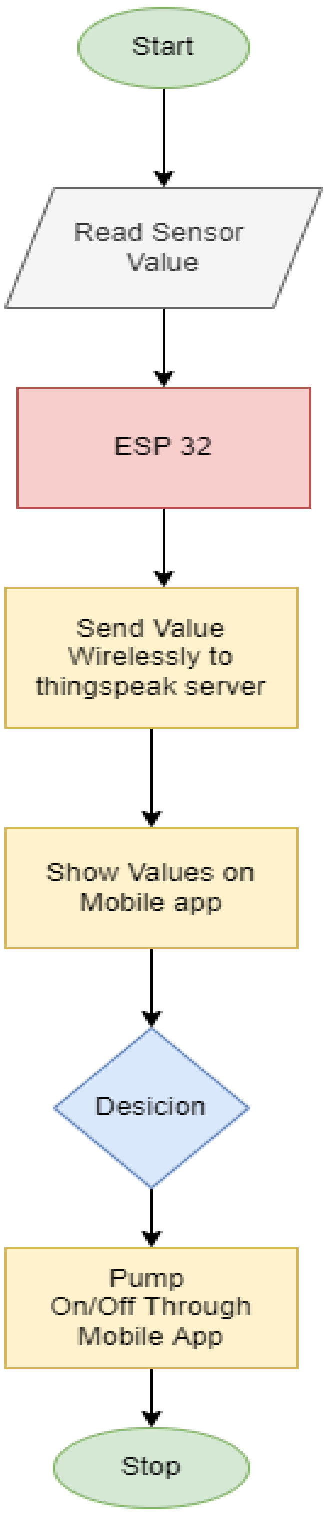

Start
Read Sensor Value
ESP 32
Send Value Wirelessly to thingspeak server
Show Values on Mobile app
Desicion
Pump On/Off Through Mobile App
Stop

Figure9: Flow Chart of Smart Irrigation System

3.5 Circuit Design

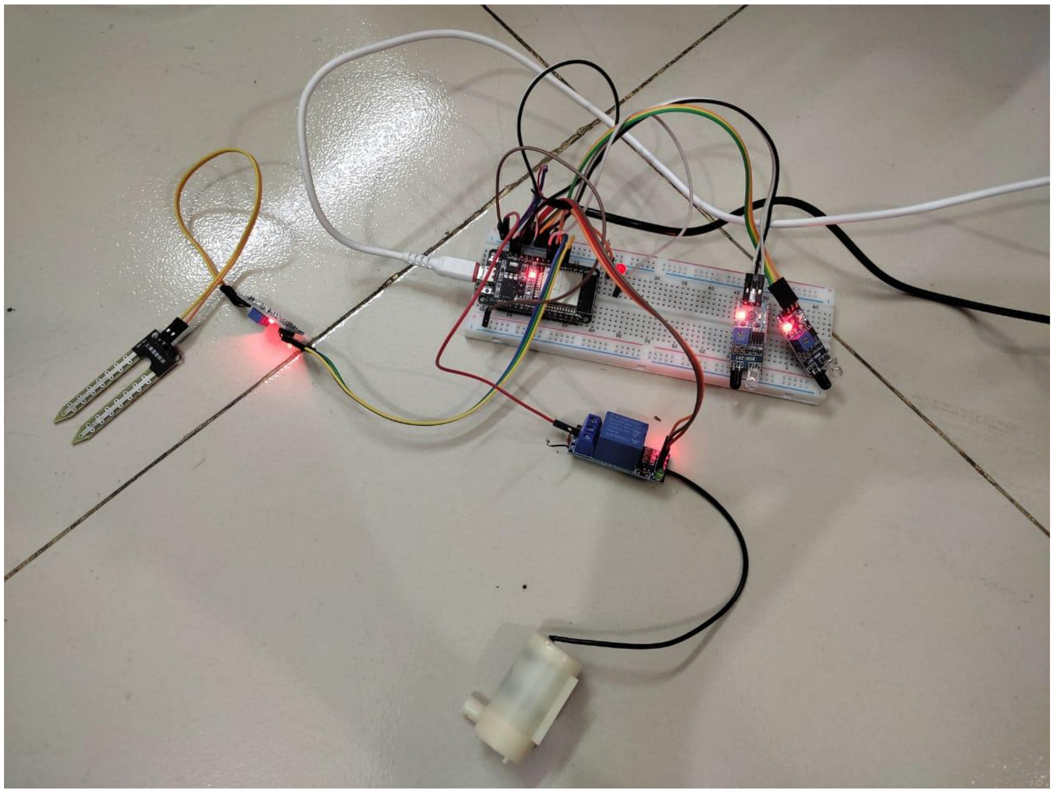


Figure10: Circuit Design of Smart Irrigation System

3.6 Code

```
#include <WiFi.h>
#include "ThingSpeak.h"

#define IRS1        32
#define IRS2        33
#define MOISTURE1   35
#define PUMP        25
#define LED         2
#define WAIT_TIME   1500
#define UPLOAD_TIME 45000
#define READ_TIME   10000
```

```
const char* ssid = "SRSP";   // your network SSID (name)
const char* password = "10-16562-2";   // your network password

WiFiClient  client;

//unsigned long myChannelNumber = 808584;
//const char * myWriteAPIKey = "0IE4X07ZCDKYH4GS";

unsigned long myChannelNumber = 1572174;
const char * myWriteAPIKey = "IE05PHEEC79SPRUM";



long time_ms;
long time_s;
long last_time_ms;
long last_time_ms2;

long time_wait;

bool pumpON;
bool ir1;
bool ir2;

long adc_mositure;
long adc_mositure_average;
long adc_mositure_count = 0;
int moist_percent;
int rainData;

bool upload_lock = false;
bool read_lock = false;

//=========================================================================

void setup() {
  // put your setup code here, to run once:
  Serial.begin (115200);
  pinMode (PUMP, OUTPUT);
  pinMode (LED, OUTPUT);
  pinMode (IRS1, INPUT);
  pinMode (IRS2, INPUT);

  digitalWrite (PUMP, HIGH);
  digitalWrite (LED, LOW);

  Serial.println ("Smart Irrigation System");
  Serial.println ("Group Name: Spartans");
  Serial.println ("Course: IoT Lab");

  delay (2000);

  ThingSpeak.begin(client);  // Initialize ThingSpeak
  connectWifi (ssid,password);

  time_wait = millis() + WAIT_TIME;
  last_time_ms = millis ();
  last_time_ms2 = last_time_ms;
}
//===================================================================

void loop() {
  // put your main code here, to run repeatedly:
  time_ms = millis();
  time_s = time_ms/1000;

  serilaCheck ();

  moist_percent = getMoisture ();
  readIR ();
  pumpControl ();
  debugPrint ();

  thingsSpeakUp ();
  thingaSpeakRead();
}
//=================================================================
void pumpControl () {
  if (pumpON)
  digitalWrite (PUMP, LOW); // PUMP ON
  else
  digitalWrite (PUMP,HIGH); // PUMP OFF - RELAY OFF
}
//=====================================================================
void serilaCheck () {
  if (Serial.peek () == '1')
  pumpON = true;
  if (Serial.peek () == '2')
  pumpON = false;
  Serial.read ();
}

//=====================================================================

void debugPrint () {
    if (millis() > time_wait) {
    Serial.print ("RUNTIME    : "); Serial.println (time_ms);
    Serial.print ("IR1        : "); Serial.println (ir1);
    Serial.print ("IR2        : "); Serial.println (ir2);
    Serial.print ("MOIST      : "); Serial.println (moist_percent);
    Serial.print ("RAIN PRECIP: "); Serial.println (rainData);
    Serial.print ("PUMP       : "); Serial.println (pumpON);
    Serial.print ("LAST UPLOAD: "); Serial.println (last_time_ms);
    Serial.print ("LAST READ  : "); Serial.println (last_time_ms2);
    Serial.print ("UPLOAD IN  : "); Serial.println ((UPLOAD_TIME - (time_ms-last_time_ms)));
    Serial.print ("READ IN    : "); Serial.println ((READ_TIME - (time_ms-last_time_ms2)));
    Serial.println (" ");

    time_wait = millis() + WAIT_TIME;
  }
}
//=====================================================================
int getMoisture () {

  //Serial.println ("get moisture");

   adc_mositure = analogRead (MOISTURE1);
   //adc_mositure = 1000;
   adc_mositure_average = adc_mositure_average + adc_mositure;
   adc_mositure_count ++;

   if (adc_mositure_count > 100){
    adc_mositure_average = adc_mositure_average / adc_mositure_count;
    adc_mositure_count = 0; }

  return map(adc_mositure, 0, 4095, 100, 0); }
//=================================================================
void readIR () {

    if (!digitalRead (IRS1))
    ir1 = true;
    if (!digitalRead (IRS2))
    ir2 = true; }
//==================================================================
void uploadChannel (int channle, int dataVal) {
  upload_lock = true;
    int x = ThingSpeak.writeField(myChannelNumber, channle, dataVal, myWriteAPIKey);
    if(x == 200){
    Serial.print("Channel update successful: "); Serial.println (channle);
    if (ir1)
    ir1 = false;
    if (ir2)
    ir2 = false; }
    else{
      Serial.println("HTTP error code " + String(x)); }
    upload_lock = false; }
//=================================================================
void thingsSpeakUp ( ) {

    if (time_s < 3)
    return;

    if (read_lock)
    return;

    if ((time_ms - last_time_ms > UPLOAD_TIME)  && !upload_lock) {
      upload_lock = true;
      Serial.println("Moisture Regular Upload");
//    uploadChannel (1, moist_percent);
//    delay (500);
    rainData =   getWeatherinfo ();
    delay (500);
    ThingSpeak.setField(1, moist_percent);
    ThingSpeak.setField(4, rainData);
    int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
    if (x=200)
    Serial.println ("Upload Successful");
    else
    Serial.print ("Error: "); Serial.println (x);
//    delay (10000);
//    uploadChannel (4, rainData);
    last_time_ms = time_ms;
    upload_lock = false; }

    if ((moist_percent < 0 || moist_percent > 100) && !upload_lock) {
       Serial.println("Moisture Out of Range");
      uploadChannel (1, moist_percent); }
    if (ir1 && !upload_lock) {
      Serial.println("IR 1 Detected");
      uploadChannel (2, ir1); }

    if (ir2 && !upload_lock) {
      Serial.println("IR 2 Detected");
      uploadChannel (3, ir2); } }

//==============================================================
void thingaSpeakRead() {
  if (upload_lock)
  return;
  if (time_ms - last_time_ms2 > READ_TIME){
     read_lock = true;
    Serial.println("Reading Channel");
  int reply = ThingSpeak.readFloatField(myChannelNumber,5);    // rad the last data of the field 1
 Serial.println (reply);
 if(reply == 1) {
   Serial.print("Pump ON Command Received - "); Serial.println (reply);
   pumpON = true; }
 else if(reply == 2) {
   Serial.print("Pump OFF Command Received - ");Serial.println (reply);
   pumpON = false; }
 last_time_ms2 = time_ms;
 read_lock = false;}}

//===========================================================================

void connectWifi (const char* id, const char* pass){
  int retryCount = 60;
  WiFi.mode(WIFI_STA);
  delay (100);
  Serial.printf("Connecting to %s ", id);
  WiFi.begin(id, pass);
  while (retryCount>0) {
  retryCount --;

  delay(500);
  Serial.print(".");

  if ( WiFi.status() == WL_CONNECTED){
  Serial.println(" CONNECTED");
  retryCount = 0;  }  } }
//=======================================================================
void disconnectWifi () {
  WiFi.disconnect(true);
  WiFi.mode(WIFI_OFF); }
//==========================================================================
int getWeatherinfo () {

//  const char* host = "api.openweathermap.org";
//  const int httpPort = 80;
//
//  String url = "/data/2.5/onecall?";
//
//  url += "lat=29.798";
//  url += "&lon=90.365";
//  url += "&exclude=alerts,current,daily,hourly";
//  url += "&appid=13e06b2a0606c1e072372f5fac43cb97";
```

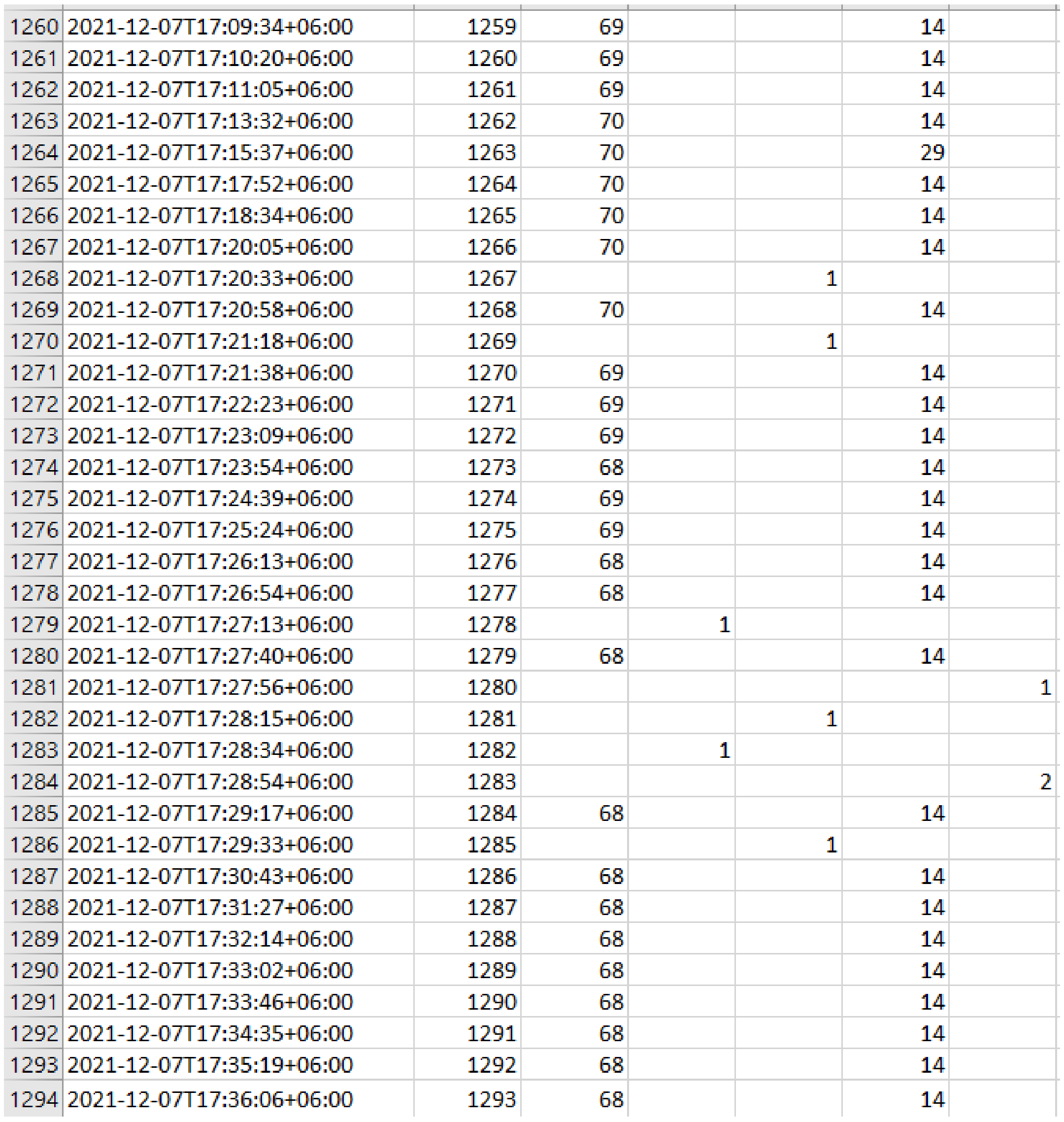
```
  const char* host = "api.weather.com";
  const int httpPort = 80;

  String url = "/v3/wx/forecast/daily/5day?";

  url += "geocode=23.7657016,90.4405956";
  url += "&format=json";
  url += "&units=m";
  url += "&language=en-US";
  url += "&apiKey=0e9a1a844a4c4d3c9a1a844a4c8d3c05";

  if (!client.connect(host, httpPort)) {
      Serial.println("Connection failed");
      return 0; }
    else{
      Serial.println("Connection succeeded");}

    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n\r\n");

    // Give Weather Underground five seconds to reply.
    unsigned long timeout = millis();
    while (!client.available()) {
      if (millis() - timeout > 5000) {
          Serial.println("Timeout");
          client.stop();
         return 0;}}

    String line;

    while(client.available())
    {
       line = client.readStringUntil('\r');
      Serial.print(line);
    }
    Serial.println(" ");

//    int index = line.indexOf("precipitation");
//    Serial.println(index);
//    String rainString = line.substring(index-1, index+17);
//    Serial.println(rainString);
//    int rain = rainString.toInt() ;
//    Serial.println(rain);

    int index = line.indexOf("precipChance");
    Serial.println(index);
    String rainString1 = line.substring(index-1, index+22);
    Serial.println(rainString1);

    index = rainString1.indexOf(",");
    Serial.println(index);
    String rainString2 = rainString1.substring(index+1, index+3);
    Serial.println(rainString2);

    int rain = rainString2.toInt() ;
    Serial.println(rain);

    return rain; }
//==========================================================
===
void testAll () {
    if (millis() > time_wait) {
    ir1 = digitalRead (IRS1);
    ir2 = digitalRead (IRS2);
    adc_mositure = analogRead (MOISTURE1);

    digitalWrite (LED, !digitalRead(LED));
    digitalWrite (PUMP, !digitalRead(PUMP));

    Serial.print ("IR1: "); Serial.println (ir1);
    Serial.print ("IR2: "); Serial.println (ir2);
    Serial.print ("ADC: "); Serial.println (adc_mositure);

    time_wait = millis() + WAIT_TIME;}}
```

CHAPTER 4

TESTING AND ANALYSIS

4.1 Experimental Data

We have collected over 1335 data stored in the Thingspeak. While doing the experiment we come to an point on some different issues like data transmision become slow sometimes . We have made some data tables which has been given bellow. Some specific values of Soil Moisture Percentage, IR, Pump Status from the 1335 data on December 07, 2021 have been given bellow:



1260	2021-12-07T17:09:34+06:00	1259	69			14	
1261	2021-12-07T17:10:20+06:00	1260	69			14	
1262	2021-12-07T17:11:05+06:00	1261	69			14	
1263	2021-12-07T17:13:32+06:00	1262	70			14	
1264	2021-12-07T17:15:37+06:00	1263	70			29	
1265	2021-12-07T17:17:52+06:00	1264	70			14	
1266	2021-12-07T17:18:34+06:00	1265	70			14	
1267	2021-12-07T17:20:05+06:00	1266	70			14	
1268	2021-12-07T17:20:33+06:00	1267			1		
1269	2021-12-07T17:20:58+06:00	1268	70			14	
1270	2021-12-07T17:21:18+06:00	1269			1		
1271	2021-12-07T17:21:38+06:00	1270	69			14	
1272	2021-12-07T17:22:23+06:00	1271	69			14	
1273	2021-12-07T17:23:09+06:00	1272	69			14	
1274	2021-12-07T17:23:54+06:00	1273	68			14	
1275	2021-12-07T17:24:39+06:00	1274	69			14	
1276	2021-12-07T17:25:24+06:00	1275	69			14	
1277	2021-12-07T17:26:13+06:00	1276	68			14	
1278	2021-12-07T17:26:54+06:00	1277	68			14	
1279	2021-12-07T17:27:13+06:00	1278		1			
1280	2021-12-07T17:27:40+06:00	1279	68			14	
1281	2021-12-07T17:27:56+06:00	1280					1
1282	2021-12-07T17:28:15+06:00	1281			1		
1283	2021-12-07T17:28:34+06:00	1282		1			
1284	2021-12-07T17:28:54+06:00	1283					2
1285	2021-12-07T17:29:17+06:00	1284	68			14	
1286	2021-12-07T17:29:33+06:00	1285			1		
1287	2021-12-07T17:30:43+06:00	1286	68			14	
1288	2021-12-07T17:31:27+06:00	1287	68			14	
1289	2021-12-07T17:32:14+06:00	1288	68			14	
1290	2021-12-07T17:33:02+06:00	1289	68			14	
1291	2021-12-07T17:33:46+06:00	1290	68			14	
1292	2021-12-07T17:34:35+06:00	1291	68			14	
1293	2021-12-07T17:35:19+06:00	1292	68			14	
1294	2021-12-07T17:36:06+06:00	1293	68			14	

Figure 12: Experimental Data Collected From Thingspeak

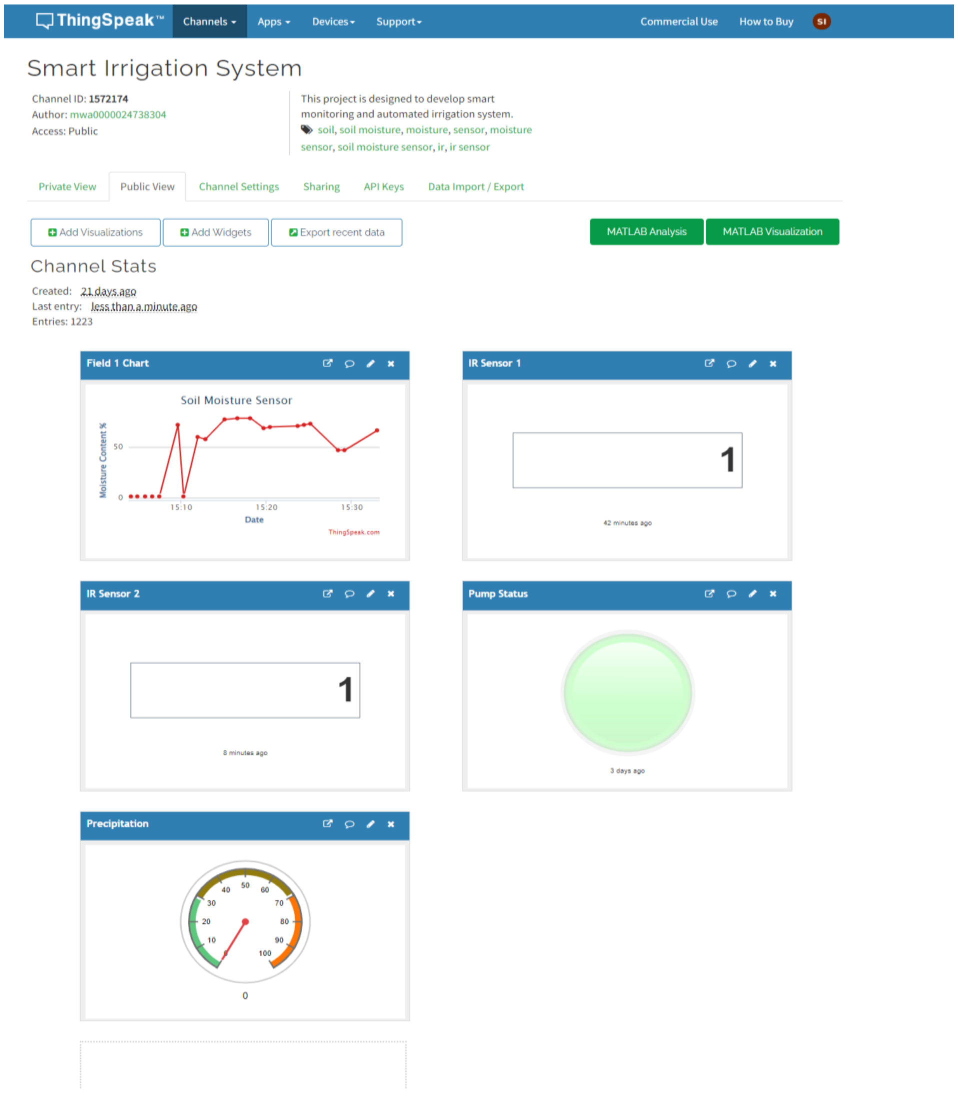
4.2 Result Analysis

During the time of our experiment we had faced some di¨culties collecting data. The moisture content was suddenly fluctuated with the drastic change of outdoor & indoor temperature. For this scenario, there some error happened at the time of taking our moisture content. But after doing some calibration we could successfully did our work. Now, we can get accurate data.

4.3 System Outcome

This project can be extended in future studies in order to improve the system in various aspects such as this project can be used vastly in the rural areas of Bangladesh.Then it would be made possible for the agricultural officers to monitor the farms without going to the lands. For this the farmers will be so much benefitted & at the same time production rate can be increased.In the near future we can also make this project an automatic irrigation system which can be useful without ant doubt. By doing this project we got to learn a lot of things of IoT and about different hardware components. Our system can get accurate sensor value and can also store data on cloud.

User of Cloud Server:



Smart Irrigation System

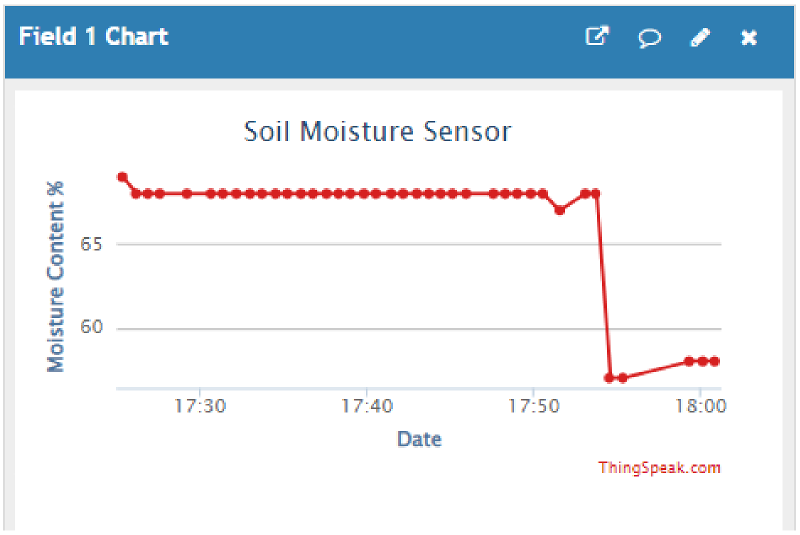
Channel ID: **1572174**
Author: mwa0000024738304
Access: Public

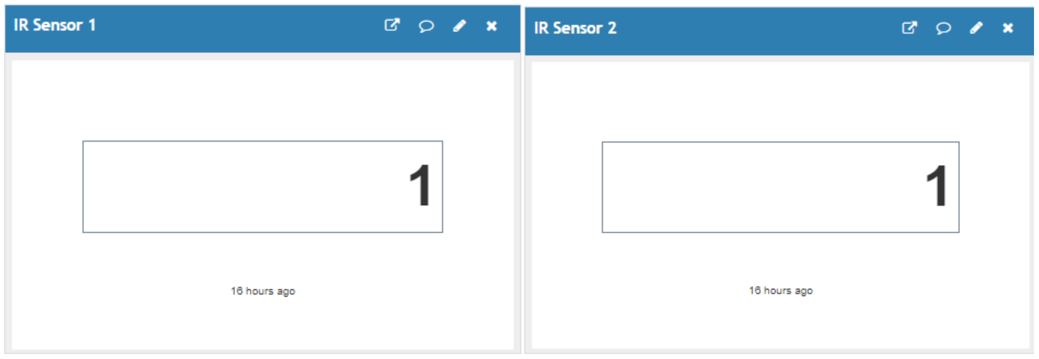
This project is designed to develop smart monitoring and automated irrigation system.
soil, soil moisture, moisture, sensor, moisture sensor, soil moisture sensor, ir, ir sensor

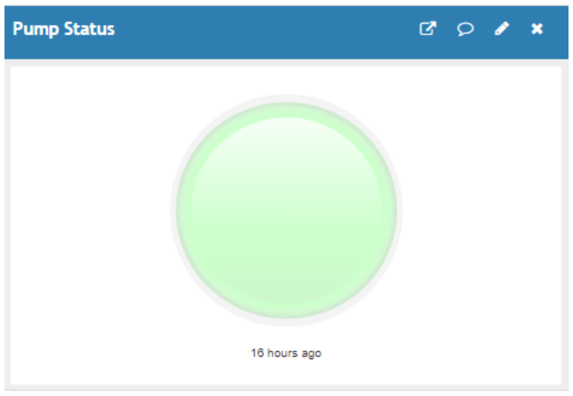
Channel Stats

Created: 21 days ago
Last entry: less than a minute ago
Entries: 1223

Figure13: ThingSpeak Cloud Server







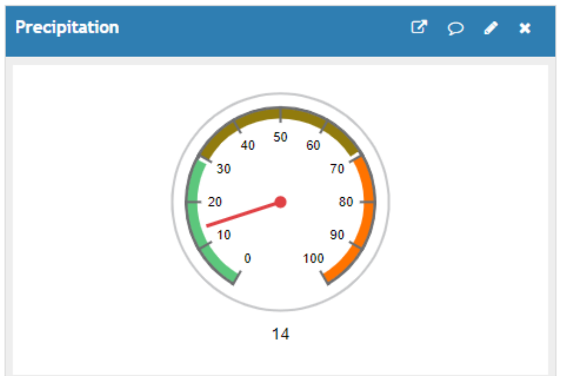


Figure14: Different Data Field for Thingspeak server

Mobile Application:

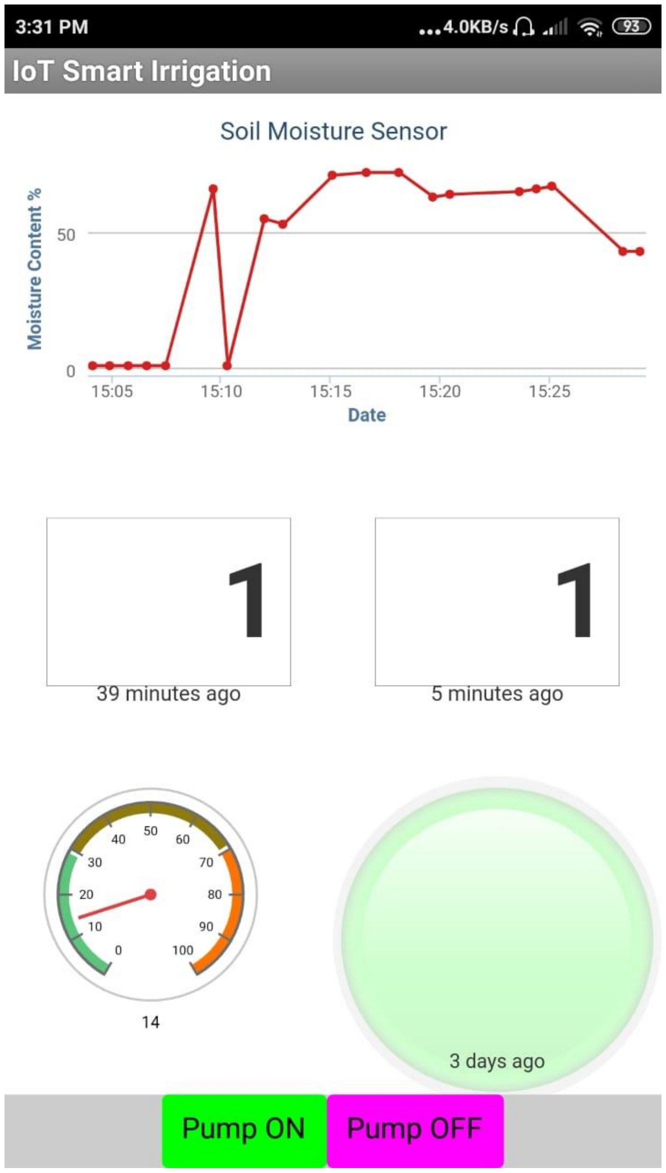


Figure15: Mobile Application of our system

CHAPTER 6

CONCLUSION

6.1 Future Works

- Notification system for intrusion detection.
- Monitoring of soil moisture content.
- Automatic Control system.

6.2 Conclusion

We developed a system by which users can get soil moisture value and can detect intrusion. Sensor values will store on Thingspeak. By using the app users can turn the pump on or off.

6.3 Reference

[1] Introduction to ESP32 | Specifications, ESP32 DevKit Board, Layout,

[2] ESP32 Useful Wi-Fi Library Functions (Arduino IDE)

[3] IR Sensor : Circuit Diagram, Types Working with Applications

[4] Best 5V arduino water pumps - ThePlantBot.com