INTERNSHIP REPORT

Data Communication & Control (Pvt.) Ltd.

Ahsan Ashraf

Muhammad Haziq Saleem

Muhammad Saad Wasif

From NED University of Engineering and Technology.

Duration: 10st September – 13th October.

Company Overview

Data Communication & Control (Pvt.) Ltd. was incorporated in May of 1994 and has since been a trend setter in the application of information and computer technology, and design engineering in Pakistan. The Company's vision is to translate innovative ideas into products and services for the benefits of the common man and provide optimized solutions with indigenous resources.

With clients ranging from the armed forces to steel mills and electricity supply companies, the firm specializes in multiple engineering expertise. Moreover, the company culture is a seemless blend of fast paced technology development incorporating extensive research where the highest standard of end product is the only aim.

Acknowledgement

Working at DCC has been an absolute perk and has broadened our overall expertise and vision to new standards. Therefore, we want to use this platform and acknowledge Sir Samir's intellectual guidance and motivational support throughout our time at his firm, he is a wonderful and compassionate human being and an icon for Pakistani Engineers. I would also like to commend my supervisors; Engineers Abdul Rehman and Taha for their collaborative efforts and always making us feel part of the team and trusting in our skills and expertise. Moreover, the HR staff; Ms. Yasmin was tremendously cooperative and friendly too. DCC has provided us an unparalleled experience during the 5 weeks and has laid a solid foundation for us to commence our career, for which we are greatly appreciative. Nonetheless, all glory is to God.

Table of Contents

1. Objective

This report will provide a detailed insight into the tasks successfully carried out during our internship tenure at Data Communication & Control (Pvt.) Ltd. It will also cover the contributions made to various projects and their overview.

2. Assigned Projects

- A Modular Precision Irrigation System (An 'Internet of Things' implementation in the Agriculture Sector).

3. Executive Project Summary

Precision Agriculture (PA) is the use of Information Technology (IT) in agriculture. The basic purpose to use Precision Agriculture is to have more yield, sustainability and productivity from the same field using IT. Different sensors are used to monitor the crop in every aspect like nutrients in soil, moisture, humidity, temperature of the crops and surrounding. Our project deals with 'Precision Irrigation' which is a sub-domain of 'Precision Agriculture'. In our project temperature, humidity and moisture of soil as well as sun exposure is being measured. But any type of sensors can be easily integrated into the project. The data gained is sent to the communication hub. Communication hub consists of two parts i.e. gateway and uplink. Gateway collects the data from all the sensors in the field at one point and uplink/upload to the database. Communication hub also dictates a water valve based on the real time soil readings which fulfil water needs of the plant whenever necessary. The Data gained in the database is then analyzed by Decision Support System (DSS) and steps are taken accordingly. Future predictions can be made based upon the historic data collected. Also If any area in the field is known to be affected based on the readings of sensors, remedial actions are taken in specified area which reduces time as well as resources.

4. Introduction/Background

The use of precision agriculture is to increase the yield and efficiency of the field and save resources which are being wasted. Consequently more productivity on small area of field in order to meet the increasing demand of food.

Water is essential for the growth of field. Large acres of agricultural field requires lot of water which means wastage is also greater in this sector. PA ensures to save water by supplying water to only those areas where needed. This limits the wastage of water.

About one third of food is wasted during the production of agricultural food each year globally. It also affects the economic sector. PA can be helpful to reduce the food wasted during cultivation and harvest process.

Healthy crops give more yield. To monitor the health of crops Drones and GPS are used which provides real time images and data of the field. This lead to identify the healthy and unhealthy crops and irrigation and drainage problems.

Additionally, nutrients sensing technology can be used to identify the particular fertilizer needs of certain crops for healthy growth and thus saving the extra amount of fertilizer which is usually wasted.

5. Project Objective

In our project we aim at achieving precision irrigation and simultaneously the remote monitoring of the field. Crops are treated with water when needed based on the readings from sensors installed in the field. The field's monitoring is carried out by continuously uploading the field data to the internet. In this way water wastage is avoided and the famer is more aware about his field's conditions.

6. Client/Stakeholder and Their Requirements

The main stakeholders in this project are the farmers, their requirements include a low cost and efficient automation system which will reduce their extra expenditures on water resources, energy resources and labor while increasing the productivity of crops. They will be able to monitor the conditions of their crops remotely and also control the watering of the plants automatically.

The state is also a major stakeholder as the energy saved by this project will benefit the energy sector of the country. More productivity will result in more exports benefitting the economy of the country.

The civil society/consumers also reap benefits of the project as the water and energy saved can be helpful in fulfiling the needs of the people.

7. Engineering Design and Specifications

In our project the system would be able to collect data of surrounding environment through different sensors. The data about humidity, surrounding temperature, soil temperature, light intensity and soil moisture is sent to a webiste which is hosted locally (can be hosted publically). The website shows the real time data graphically. Water valve situated in the field is electronically triggered with these sensors'

readin. If the moisture of soil falls below the set threshold value then water valve is opened automatically which basically funtions to provide water only when soil is dry and needs water.

Humidity readings are taken with the accuracy of 5% within the range of 20-80% humidity. Environmental temperature readings are accurate with the error of ±2°C. Soil temperature readings have error of ±0.5°C between the range from -10°C to +85°C.

8. Benchmarking

Various projects relating to our solution are already up and working in developed countries. One example is **"Drip - Low-Cost Precision Irrigation for Developing Nations"**. It almost uses the same format for communication. The difference is the use of communication modules. Our radio module can be effective upto longer distances than the one used in the Drip project. Also we have created a custom database and designed a personal website which is used as the Graphical User Interface.

Another examplary project that is being implemented in Europe is called "FIGARO". It is a smarter, more precise irrigation system to help promote sustainable farming. EU-funded researchers working within the FIGARO project have developed an innovative, high-tech irrigation platform capable of accurately managing the quantity of water used.

9. Impact of the Solution

This project directly impacts the farmers in terms of more productivity in less resources as well as the remote monitoring of his field. It is also helpful in saving energy for the country because it precisely needs energy when required. The energy saved can be utilized to help fulfilling the energy deficit in our country.

By reading a 2011 report by the United Nations Food and Agriculture Organisation (FAO) we found that Pakistan's crops productivity per acre is behind regional as well as global yields per acre for almost all the crops mainly including wheat, rice, sugarcane and pulses. It is surprising to observe that even though Pakistan stands in the top 10 major wheat producers in the world its per hectare produce is only 2.6 tonnes compared to India's 2.8 tonnes and China's 4.7 tonnes.

An estimate of wastage of water can be found out by reading a research carried out in Pakistan that an average of 4000 gallons of water over 6 months is used to produce 1000 kg cotton per acre while sophisticated irrigation techniques in America provide an approximate total of 2500 gallons of water over 6 months to produce 1000 kg per acre.

According to the Pacific Institute's "Sustaining California Agriculture in an Uncertain Future" report, a Kansas study found that precision irrigation reduced water use by 20%. The same report references a consulting firm in Washington, using AgriMet to provide irrigation scheduling and soil moisture monitoring services to farmers, had found that some reduced their water and energy use by as much as 50%. An

October, 2008 Maariv Business report on Israeli agricultural irrigation determined that use of field sensors reduces water use by 20% or more.

Therefore Our IoT project along with Precision Irrigation techniques like Drip irrigation, Micro-spray irrigation, Low-energy sprinklers irrigation and recycled waste water irrigation etc. should be implemented in Pakistan in order to avoid this wastage of water.

10. Design Concepts

During the process of ideation of the project, different ideas about the sensors and modules to be used arose in our minds. For example, the purpose of radio communication could be fulfiled by using a stand-alone microcontroller like Arduino Uno with a separate radio transceiver module like the NRF24l01+. But the utilization of the Loraduino board results in a more compact design. The Wi-Fi module used here is called NodeMCU while others like esp8266 could be used. However, the esp8266 has stability problems that's why the choice of a more stable developed board of esp8266 (NodeMCU) was made. Various kinds of temperature sensors like lm35,tmp35 could also be used but the use of DS18B20 was more suitable for our design of the probe. Different kinds of light sensors and moisture sensors that are more precise can also be bought but we decided to keep the project cost effective.

11. Design Details

Our project design is based on two main parts:

1. Field
2. Server

FIELD

The probe, communication hub and water valve are the main components situated in the field.

Probe:

Purpose:

To collect sensor data (temperatures, humidity, soil moisture and light intensity) at a point in the field and send the data wirelessly via Radio communication.

Components:

- LORADUINO:

  It is an arduino pro mini microcontroller with built in LoRa Radio transceiver (model SX1278). Arduino sketch can be uploaded via FTDI USB to Serial interface. It has low power consumption, long transmission distance and low error, board runs on 433MHZ frequency.

- DHT11:

  DHT11 is a digital output based sensor which measures humidity and temperature of surrounding. It is Low cost sensor and can be operated with 3 to 5V power. It is good for 20-80% humidity readings with 5% accuracy and for 0-50°C temperature readings with ±2°C accuracy.

- DS18B20:

  It is a digital temperature sensor which is used for the purpose of measuring soil temperature. It has range between -55°C to +125°C. It has ±0.5°C Accuracy from -10°C to +85°C. It requires 5v input to work.

- Photoresistor:

  Photoresistor is an analog sensor. It is used to measure the light intensity. It works on 5v power.

- Soil Moisture Sensor:

  Soil moisture sensor is also an analog sensor which works on 5v power. It measures the moisture of in the soil by measuring the conductivity between the plates.

- Buck Converter:

  A buck converter is a DC-DC Adjustable Step Down converter which steps down input voltage to its output load. The output is adjustable from 0-12V with maximum output load current 3A.

- 12V Battery:

  Battery is used to operate the system.

Working:

These sensors are attached to the LoRaduino which continuously transmit real time data. DS18B20 is fitted in the soil to measure temperature of the soil. The basic purpose of DHT11 is to measure humidity and temperature of the surrounding. Photo resistor is used to sense the sunlight falling on it. Soil Moisture sensor tells if crops need water by knowing about dryness of the soil. All this data is processed in

LoRaarduino and sent to the communication hub via LoRaduino transceiver. All these sensors and the arduino are housed inside a single packaging known as the probes.

Communication Hub:

Purpose:

To receive the radio signals sent by the probes (one probe in our case) in the field, process the received data, take necessary decisions for watering the plants and also upload the data to the internet via Wi-Fi.

Components:

- LORADUINO:

  It is an arduino pro mini microcontroller with built in LoRa Radio transceiver (model SX1278). Arduino sketch can be uploaded via FTDI USB to Serial interface. It has low power consumption, long transmission distance and low error, board runs on 433MHZ frequency.

- NODE MCU ESP8266 V1.0:

  It is a development board for the esp8266 Wi-Fi Module. It is a low powered microcontroller and has an onboard esp8266 Wi-Fi chip which gives our project the Wi-Fi capability needed to upload data to the internet. It can be programmed through the Arduino IDE by downloading and installing the NodeMCU board in Arduino IDE.

- RELAY MODULE:

  A 5V D.C. Relay module is used here which can also work with the 3.3V provided by the NodeMCU. It's switching is controlled by connecting its Input to an Output Pin of NodeMCU. A 1N4007 diode (used as a snubber diode) is connected across the relay's output because at the output a solenoid valve will be connected. The solenoid valve is an inductive load. Therefore, with the rapid change in current (when the source is disconnected), a voltage spike occurs across it which is dealt by the diode by letting the discharging current pass through it until it becomes zero thus resulting in safe closing of valve.

- Power Supply:

  A 12V battery is used to power the Loraduino and NodeMCU while a 12V D.C. Adapter is connected to the relay switch and consequently powers the Water Valve.

Working:

The probes attached at various points in the field are all configured to send the sensors' data to a central communication hub which is also situated in the field. The short distance and absence of obstacles

between the probes and the hub is the reason to use radio frequency for communication as it is a faster way to communicate where distance and obstacles are not a problem. An additional Power Amplifier/Low Noise Amplifier antenna can be used with the LoRa radio transceiver module if the distance between the probe and hub is greater than 80 meters. It receives radio signals of the probes and gives it as input to the LoRaduino. This received data is then sent to NodeMCU board with wired serial communication between the LoRaduino and the NodeMCU board. It processes the received message and uploads it to a MySQL Database (on a locally hosted server in our case. However, the server can be made live by purchasing a domain) and consequently each probes' sensor data reaches the database of our server. NodeMCU also checks the moisture value and compares it with a set threshold(a value at which soil is dry and needs water) and then decides wheter to turn the relay "ON" or "OFF".

Water Valve:

Function:

To water the field according to the decisions taken by the communication hub.

Components:

- SOLENOID WATER VALVE:

  A ½ inch electrically controlled solenoid water valve is used to control the water flow to the field. It takes 12V 1A D.C. Input.

Working:

The input supply to the solenoid valve is provided by the 12V D.C. Adapter (present in the hub) but the adapter's wires pass through a relay inside the hub before reaching the water valve. Therefore making it controllable by the relay and consequently by the NodeMCU microcontroller present in the Hub.

Server

Our server is hosted 'locally' on the same internet connection as the Hub's Wi-Fi. However, it can be made public by purchasing a domain. It mainly consists of two parts:

1. Database
2. Graphical User Interface (Website)

Database :

A database is a structured collection of data in the form of tables that is used by the application systems of some given enterprise, and that is managed by a database management system. For the purpose of this project we are using MySQL database management system based on SQL database language to store our data transmitted from communication hub. MySQL is used by most modern websites and web-based services as a convenient and fast-access storage and data fetching solution for large volumes of data. MySQL can be accessed through many tools. It can be easily communicated via PHP (PHP Hypertext preprocessor) whose primary focus is to manipulate HTML. A user can submit queries to a database via PHP, allowing insertion, retrieval and manipulation of information into/from the database.

Node MCU in communication hub is transmitting readings of different probes to MySQL Database through Wi-Fi Communication. The Arduino code for writing probe data to Database is given in Appendix B.

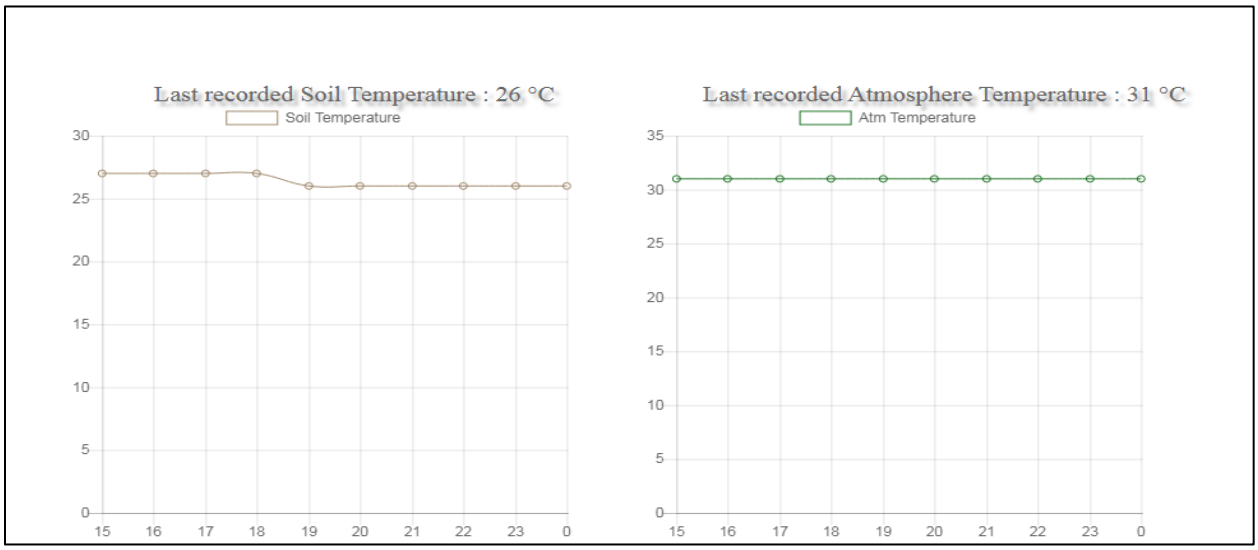
This data is then used by our website for analysis in graphical form. There are six columns in our database which are Time, Soil Temperature, Atmosphere Temperature, Humidity, Moisture and Light Intensity. . Time is in Hours which starts from zero after 23 entries.We can use these columns to see the trend over any period of time in order to observe the impact of temperature,humidity etc. on the production of crop.The code for fetching data from database to Website Backend is given at Appendix B.

User Interface:

User Interface displays graphs of Soil Temperature, Atmosphere Temperature, Humidity, moisture with respect to Time on x-axis. Our User Interface for this project is Website which is based on Python Django Framework. A Python Script(fetching.py) is fetching data from MySQL database and returning a separate array for each column.

{https://github.com/msw1998/Precise-agriculturewebsite/blob/master/mysite/fetching.py}

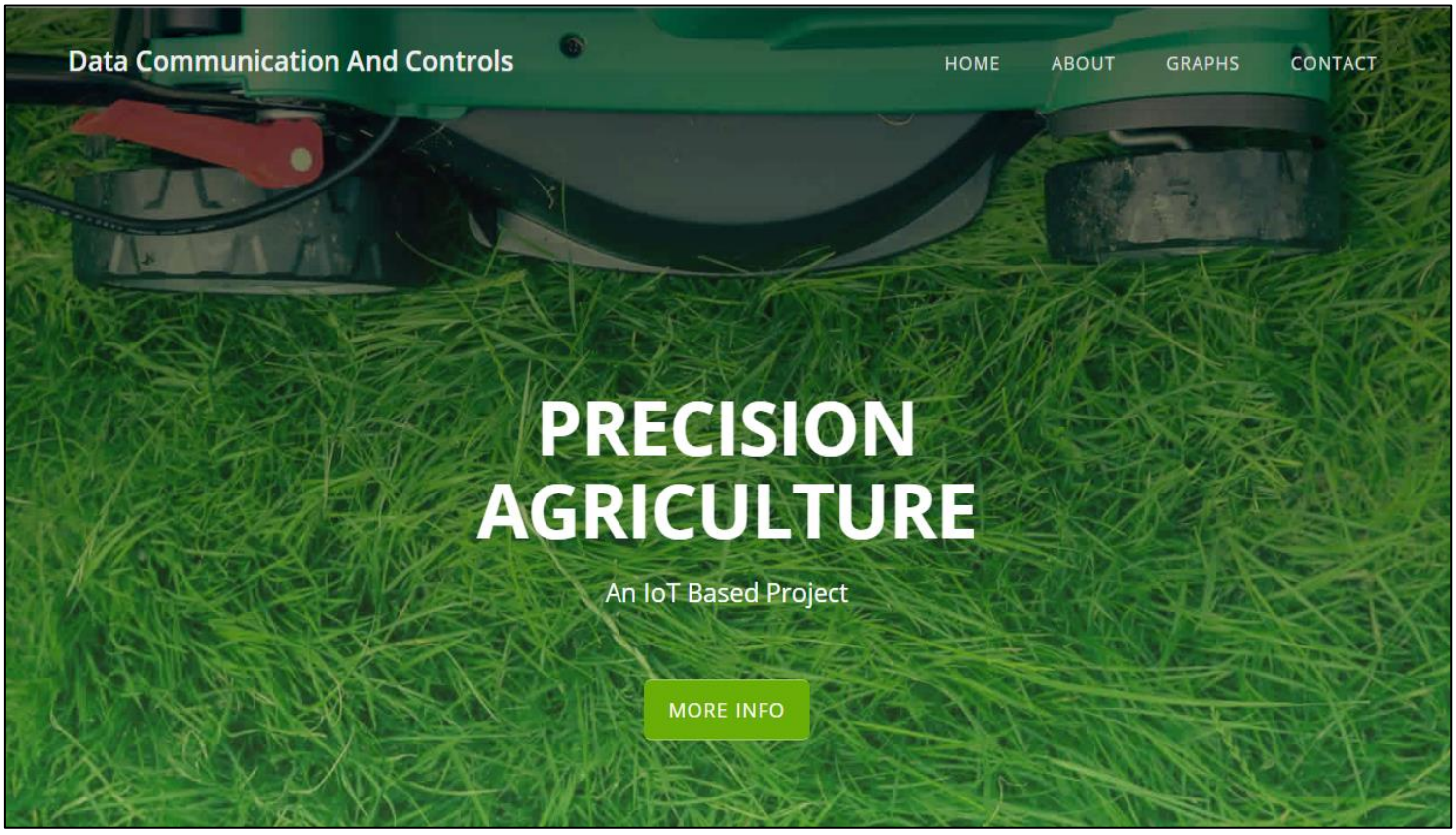
These arrays are imported in views.py file in which a function is passing these arrays to charts.html file where Charts are displayed using JavaScript library Chart.js



{https://github.com/msw1998/Precise-agriculture-website/blob/master/mysite/personal/templates/personal/charts.html}.

On x-axis is Time and on y-axis are Sensor Readings.

The styling of website is done using CSS/Html. {https://github.com/msw1998/Precise-agriculture-website/blob/master/mysite/personal/templates/personal/header.html}



12. Prototype

Our prototype consists mainly of two separately boxed circuits which can be powered by a 12V D.C. supply or by a 5V USB cable.

One of the two boxes is named “Probe”. The probe’s box is mounted on top of a PVC pipe which will be inserted in the field/plant’s soil.

The second box is named “Communication Hub”. It can also be mounted on a PVC pipe if intended to insert in the soil or can be kept un-mounted if intended to place it on a horizontal solid surface. The codes uploaded in all (3) the microcontrollers are provided in Appendix-B. The codes are explained step by step in the comments.

The second main part is the web server which is hosted locally on a computer having an internet connection. The Hub’s Wi-Fi is then connected to the same internet connection.

I. Construction of the Probe:

The construction of the probe’s circuit has been carried out on a vero-board. The circuit’s schematic has been provided in Appendix-A. The code to be uploaded in the Probe’s Loraduino has been provided in Appendix-B.

➢ Power Supply:

There are two ports for external power connection.

i. A U.S.B. Female Port (5V D.C.)
ii. A 2 terminal block connector (12V D.C.)

Both ports are followed by a protection diode. The 12V supply is stepped down to 5V by means of a buck converter. After conversion both the supplies are now capable of supplying 5V. There is an SPDT switch to choose the supply path that leads to the final 5V on-board output terminal.

➢ Sensor Interfacing with Loraduino:

Two female headers are soldered for mounting the Loraduino board on it. There are total four sensors as described in the ‘Design Details’ section. All the sensors are provided VCC and GND directly by the 5V output terminal. The LDR is connected in series with a 10k Ohms resistor. LDR’s free terminal is connected to VCC and resistor’s free terminal is connected to GND. The common terminal between resistor and LDR is connected to analog data pin A0 of Loraduino. DS18B20’s digital data pin is connected to

D9 of Loraduino. Moisture sensor's analog data pin is connected to A3 of Loraduino. Finally the DHT-11's digital data pin is connected to D7 of Loraduino. The LDR and DHT-11 sensors are exposed to the atmosphere through holes in the box while the moisture and DS18B20 sensors go through the pipe into the soil.

II. Construction of the Communication Hub:

The construction of the probe's circuit has been carried out on a vero-board. The circuit's schematic has been provided in Appendix-A. The code to be uploaded in the Hub's Loraduino and Hub's NodeMCU has been provided in Appendix-B.

➢ Power Supply:

There are two ports for external power connection.

i. A U.S.B. Female Port (5V D.C.)
ii. A 2 terminal block connector (12V D.C.)
iii. A 2 terminal block converter for solenoid valve's power (12V 1A D.C.)

Both ports are followed by a protection diode. The 12V supply is stepped down to 5V by means of a buck converter. After conversion both the supplies are now capable of supplying 5V. There is an SPDT switch to choose the supply path that leads to the final 5V on-board output terminal.

➢ Serial Communication between Loraduino and NodeMCU:

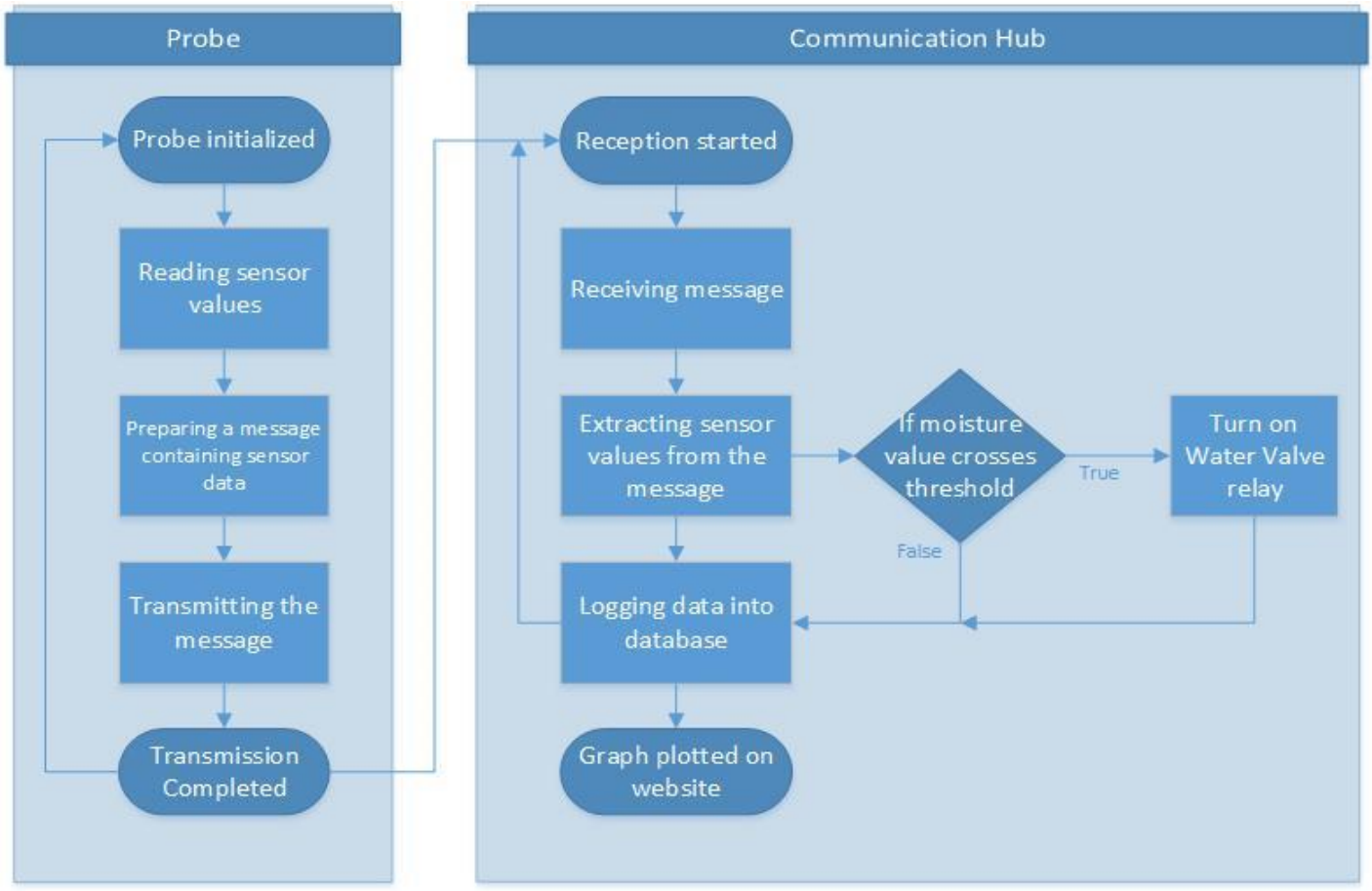
We have to transfer the received data from Loraduino to the NodeMCU so the NodeMCU can upload it to the database. For this purpose Serial Communication is established between the two microcontrollers by connecting the RX pin of NodeMCU with the TX (D1) pin of Loraduino and connecting the TX pin of NodeMCU with RX(D0) pin of Loraduino.

➢ NodeMCU-Relay connections:

In order to control the relay module the NodeMCU's D0 pin is connected to the Input pin of relay. The relay's VCC is connected to 3V3 pin of NodeMCU and the GND of relay is connected to GND of NodeMCU.

The 12V positive wire coming from the terminal block for solenoid valve's power connects to the relay's NO (Normally Open) terminal and then the COM (common) terminal of the relay is connected to the diode's negative terminal. The negative wire coming from the 12V terminal block is connected to the positive terminal of the diode. The valve is then connected parallel (polarity doesn't matter here) to the diode.

III. Flowchart:



12. Society, Environment, and Sustainability Considerations:

Approximately 61% of Pakistan's population is directly/indirectly related to the profession of agriculture. Their economic conditions and well-being is connected to the benefits they gain from this profession. By implementing precision irrigation techniques water wastage is reduced hence the cost the people pay for irrigation water also gets reduced. It also results in increase of productivity of crops which benefits them economically. Pakistan is a water-deficit country. At present, the annual per capita water- availability in Pakistan is about 1,100 cubic meter (m^3); below 1,000 m^3 , countries begin experiencing chronic water stress (Population Action International, 1993). Hence saving water has become a necessity for ensuring the comfort of our future generations. Developed countries in Europe and America have already started sophisticated irrigation techniques for saving water which include Drip irrigation, Micro-spray irrigation, Low-energy sprinklers irrigation and recycled wastewater

irrigation etc. Pakistan is yet to utilize these developed irrigation techniques as it mostly uses flood irrigation which results in the wastage of water. Therefore, implementing this project and using drip irrigation techniques the water wastage can be reduced upto an average of 20% as indicated in the California Agriculture studies mentioned in the “Impact of the Solution” section.

13. Conclusion, Future Works

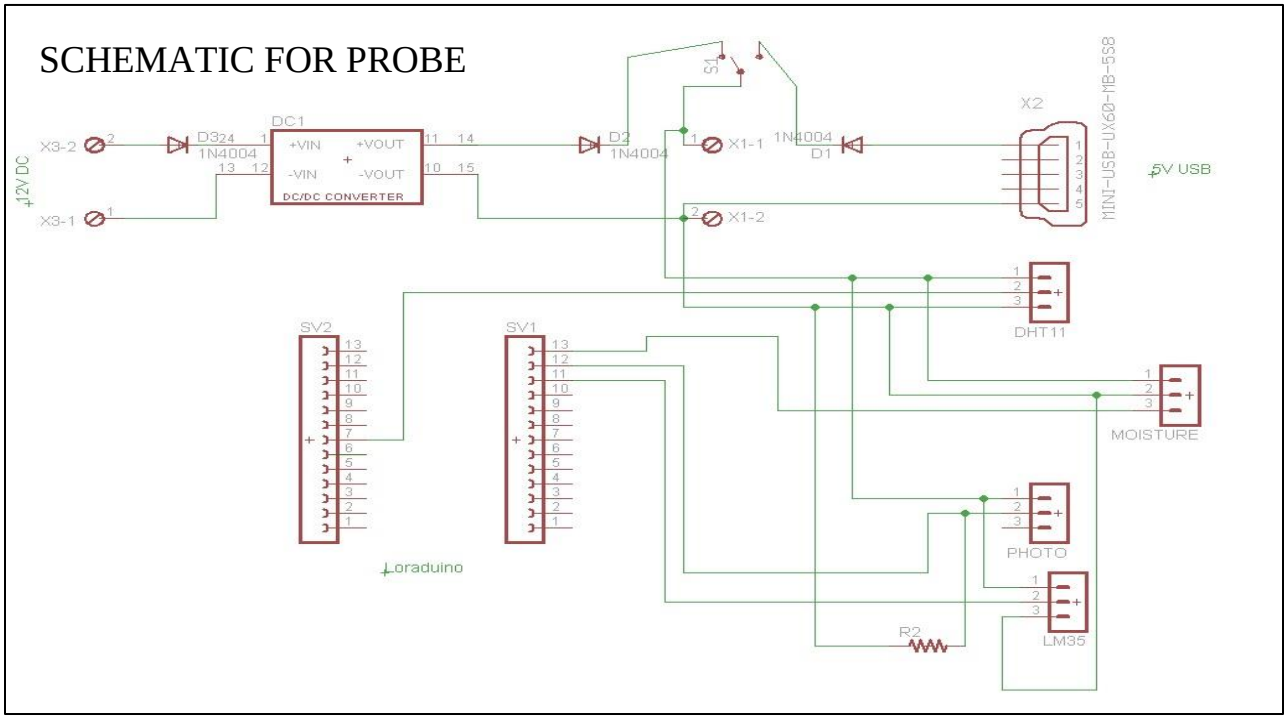
This project provides a base which can be expanded to industrial levels for real world application in the greens of our country. It will bring efficiency and productivity in the farming techniques used in Pakistan.

Future prospects include integrating a GPS transmitter inside the probe which will be helpful in creating graphical maps of the field on which the field parameters will be continuously plotted. Using GSM communication to upload the field data because a stable Wi-Fi connectivity cannot be achieved in most of the rural field areas. Making well thought PCB layouts for transferring the circuits to a PCB and making the packaging more reliable, durable and tolerant to different weather conditions. Using industrial grade sensors like SM150T for moisture and temperature sensing in order to make it more reliable against rust and corrosion etc. Making the server public in order to make the user interface accessible from anywhere in the world.

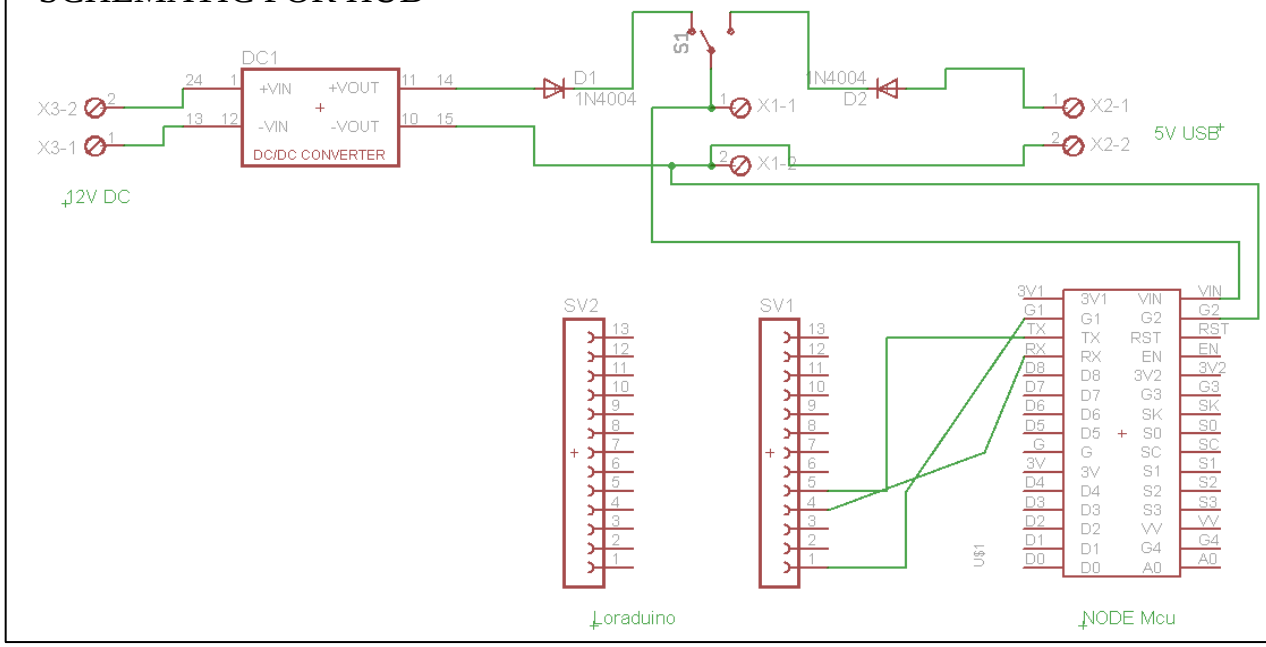
APPENDIX-A

(Schematics)

SCHEMATIC FOR PROBE



SCHEMATIC FOR HUB



APPENDIX-B

(Libraries & Codes)

- External libraries used in the codes:

1. DHT Sensor Library:
   https://github.com/RobTillaart/Arduino/tree/master/libraries/DHTlib
2. One Wire:
   https://github.com/PaulStoffregen/OneWire
3. Dallas Temperature:
   https://github.com/milesburton/Arduino-Temperature-Control-Library
4. Radio Head (RH_RF95):
   http://www.airspayce.com/mikem/arduino/RadioHead/
5. ESP8266 Wi-Fi:
   https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi
6. MySQL Connector Arduino:
   https://github.com/ChuckBell/MySQL_Connector_Arduino
7. Arduino JSON:
   https://github.com/bblanchon/ArduinoJson

- Code Uploaded in Probe's LoRaduino Board:

```
//***Probe***
//Following is the code for Data sensing and transmitting it to the Hub via built-in
radio transceiver SX-1278 on the Loraduino board


/*************************************************************************************
**********************/

#include <dht.h> //For DHT-11 Sensor.
#include <OneWire.h> //For 18B20 sensor.
#include <DallasTemperature.h> //For 18B20 sensor.
#include <SPI.h> //For radio communication.
#include <RH_RF95.h> //For radio communication.

/*********************************************************************/

#define ONE_WIRE_BUS 9 //18B20 data wire is plugged into digital pin 9 on the
Loraduino.
#define DHT11Pin 7 //DHT-11 data wire is plugged into digital pin 7 on the Loraduino.
int LdrPin = 0; //Photo resistor on analog 0 of Loraduino.
int MoisturePin=3; //Moisture sensor on analog 3 of Loraduino.

/*********************************************************************/

dht DHT; //dht object for dht-11 sensor.
OneWire oneWire(ONE_WIRE_BUS); //Setup a oneWire instance to communicate with any
OneWire devices (not just Maxim/Dallas temperature ICs).
DallasTemperature sensors(&oneWire); //Passing our oneWire reference to Dallas
Temperature.
RH_RF95 SX1278; //radio object (SX1278 is the model number of our built-in radio
transceiver).

/*********************************************************************/

char payload[150]; //a char array in which the message to be sent (via radio) will be
stored.
float TempVal; //Soil temp var.
float DHTTemp; //Surrounding temp var.
float DHTHumidity; //Surrounding humidity var.
float LdrVal; //Light intensity var.
float MoistureVal; //Moisture value var.

/*********************************************************************/

void setup(){

  DHT.read11(DHT11Pin); //function to start listening to dht11 input.
  Serial.begin(9600);
  sensors.begin(); //function to start listening to 18B20 sensor.

  while (!Serial) ; // Wait for serial port to be available.

    if (!SX1278.init()) //Initializing the radio and checking if it is initialized or
not.
        Serial.println("Notice:init failed");
    //Default initialization parameters are 434.0MHz, 13dBm, Bw=125 kHz, Cr=4/5,
Sf=128chips/symbol, CRC on.
}
```

```
void loop()
{

  /*****************************************************************/

  //Reading Soil Temp by 18B20 sensor
  sensors.requestTemperatures(); //to read the temperature provided as input.
  Serial.print("Soil Temperature= ");
  TempVal= sensors.getTempCByIndex(0); //there can be multiple sensors with multiple
indices, in our case there is only one at default index 0.
  Serial.println(TempVal);

  /*****************************************************************/

  //Again initializing the radio because after calling "sensors.requestTemperatures()"
in the above code, radio stops working and doesn't transmit final message.
  if (!SX1278.init())
        Serial.println("Notice:init failed");

  /*****************************************************************/

  //Reading Surrounding Temp and Humidity by DHT-11 sensor.
  Serial.print("Surrounding Temperature = ");
  DHTTemp = DHT.temperature; //Temp in celsius stored in our variable.
  Serial.println(DHTTemp);
  Serial.print("Surrounding Humidity = ");
  DHTHumidity = DHT.humidity; //Humidity in percentage stored in our variable.
  Serial.println(DHTHumidity);

  /*****************************************************************/

  //Reading Light Intensity by Photo resistor
  Serial.print("Light Intensity: ");
  LdrVal = safeAnalogRead(LdrPin); //Analog reading stored in LdrVal. The function
"safeAnalogRead" is defined at the end of the code.
  LdrVal = (LdrVal/1023.0)*100.0; //Converting analog reading (0-1023) to percentage.
  Serial.print(LdrVal);
  Serial.println("%");

  /*****************************************************************/

  //Reading Soil Moisture by analog soil moisture sensor
  MoistureVal = safeAnalogRead(MoisturePin); //analog reading is stored in our
variable.
  MoistureVal = 100-((MoistureVal/1023.0)*100.0); //converting the moisture to
percentage. (also subtracting it from 100 so that "0" means "dry" and "100" means
"wet".
  Serial.print("Soil Moisture: ");
  Serial.println(MoistureVal);
  Serial.println("\n\n\n");

  /*****************************************************************/

  //creating a "String" which contains all the measured values.
  //the String has angle brackets at start and end which act as start and end markers
for the message.
  //The markers are required for synchronized wired serial communication between the
receiver Loraduino and the NodeMCU.
  //If the brackets are removed the string looks exactly like JSON format, this helps
in parsing the string and extracting the integer values from it after reception.
  String s1 = "<{\"tempsoil\" : " + (String)((int)TempVal) + ", \"tempair\" : " +
(String)((int)DHTTemp) + ", \"humidity\" : " + (String)((int)DHTHumidity) + ",
```

```
\"moisture\" : " + (String)((int)MoistureVal) + ", \"ldrval\" : " +
(String)((int)LdrVal)+"}>";
  s1.toCharArray(payload,150); //Converting the string to a char array (because the
radio can't send "String" but can send a char array) and saving it in payload.

  /**********************************************************************/

  //Sending the message to the receiver Loraduino.
  Serial.println(payload);
  uint8_t lennn = sizeof(payload); //saving the size of the message in lennn (uint8_t
(i.e. char) data type is used because it is required as parameter in the .send()
function).
  SX1278.send(payload, lennn); //transmitting the message.
  Serial.println("mark0"); //The message is loaded and ready to be transmitted.
  SX1278.waitPacketSent(); //Wait till complete transmission of the message.
  Serial.println("mark1"); //The message is transmitted and is on air.
  // Now wait for a reply (we are not receiving an acknowledgement here so the
following four lines can be ignored).
  uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
  uint8_t len = sizeof(buf);
  delay(1000);
  Serial.println("mark2"); //After receiving the empty acknowledgement.

  /**********************************************************************/

  delay(2000); //2 seconds delay before again starting the sensing process.

}



/**********************************************************************/

//A function is created to avoid reading false analog values which are generated
because we are using multiple analog pins here on the Loraduino.
//When we are reading any analog input pin on the Loraduino and we switch to another
pin to read the other sensor, the first reading that we get from the second pin is
false.
//In order to avoid this unstability we take 2 readings from the sensor, the first is
false so we overwrite it with the second reading.

int safeAnalogRead(int pin)
{
  int x = analogRead(pin);  //Make an initial reading to set up the ADC
  delay(10);                //Let the ADC stabilize
  x = analogRead(pin);      //Toss the first reading and take one we will keep
  delay(10);                //Delay again to be friendly to future readings
  return x;
}

/**********************************************************************************
*********************/
```

- Code Uploaded to Hub's LoRaduino Board:

```
//***Receiver***
//Following is the code for receiving the transmitted message (from the probe) via
built-in radio transceiver SX-1278 on the Loraduino board.


/*************************************************************************************
/

#include <SPI.h> //For radio communication.
#include <RH_RF95.h> //For radio communication.

/*********************************************************************/

RH_RF95 SX1278; //radio object (SX1278 is the model number of our built-in radio
transceiver).

/*********************************************************************/


void setup()
{
    Serial.begin(115200);
    while (!Serial) ; // Wait for serial port to be available
    if (!SX1278.init()) //Initializing the radio and checking if it is initialized or
not.
        Serial.println("init failed");
    //Default initialization parameters are 434.0MHz, 13dBm, Bw=125 kHz, Cr=4/5,
Sf=128chips/symbol, CRC on.
}

/*********************************************************************/

void loop()
{

/*********************************************************************/

    if (SX1278.available()) //Checking if there is a knock knock on the receiver's
door.
    {
        // Should be a message for us now

        uint8_t buf[RH_RF95_MAX_MESSAGE_LEN]; //initializing char array buf with the
size of the available-to-receive message.
        uint8_t len = sizeof(buf); //checking and saving the length of the avaialable
message in len.

        /*********************************************************************/

        if (SX1278.recv(buf, &len)) //if reception is successful
        {
            Serial.println("");
            Serial.print("Receive Message: ");
            Serial.println((char*)buf);
            Serial.println("\n\n\n");
            //Serial.print("RSSI: "); //uncomment this to also print the received
signal strength.
            //Serial.println(SX1278.lastRssi(), DEC); //uncomment this to also print
the received signal strength.
```

```
            Serial.write((char*)buf); //sending the received message to NodeMCU via RX
TX serial lines.
        }

        else //if reception is failed
        {
            Serial.println("recv failed");
        }

        /**********************************************************************/
    }

/**************************************************************************/

}
```

- Code Uploaded to Hub's NodeMCU Board:

```
//***Uplink***
//The following code is used to upload the field-data to the database via NodeMCU
esp8266 Wi-Fi module.

/************************************************************************/

#include <ESP8266WiFi.h> //NodeMCU has esp8266 wifi module built-in.
#include <MySQL_Connection.h> //to connect to the mysql database and perform
operations like insertion etc.
#include <MySQL_Cursor.h> //to connect to the mysql database and perform operations
like insertion etc.
#include <ArduinoJson.h> //in order to parse the received message and extract integer
values from it.

/************************************************************************/

int relaypin = 16; //relay is connected to D0 of the NodeMCU which is equivalent to
pin 16 with respect to Arduino IDE.

/************************************************************************/

char ssid[] = "TP-LINK_CE64";          // your Wi-Fi SSID
char pass[] = "dccinterns18";      // your Wi-Fi Password
IPAddress server_addr(192,168,0,35);  // IP of the MySQL server here
char user[] = "node";               // MySQL user login username
char password[] = "123";         // MySQL user login password

/************************************************************************/

int tt=0; //a variable which will increment by '1' and reset to '0' after '23' to
represent/simulate 'hours' i.e. 0-23.
const byte numChars = 150; //a safe maximum lenth of our received message via RX TX
Serial communication between gateway/receiver and uplink.
char receivedChars[numChars]; //array initialization for received message.
bool newData = false; //variable to keep check of new incoming data.

/************************************************************************/

//mysql query for insertion where %d will later be replaced by parsed integer values.
//here database name is "dcc", table name is "pa", column names are
"Id","Time","Temp_soil","Temp_atm","Humidity","Moisture","Light".
char INSERT_SQL[] = "INSERT INTO dcc.pa (Id,Time, Temp_soil, Temp_atm, Humidity,
Moisture, Light) VALUES ('',%d,%d,%d,%d,%d,%d)";

/************************************************************************/

WiFiClient client;              //creating an instance for Wi-Fi Connection.
MySQL_Connection conn(&client); //passing "client" as a reference for mysql
connection.
MySQL_Cursor* cursor;           //this pointer object is used for data insertion
operation into mysql.

/************************************************************************/


void setup()
{

  Serial.begin(115200);
```

```
  pinMode(relaypin,OUTPUT);
  digitalWrite(relaypin,HIGH); //turning the relay "off". the relay board turns "on"
when written "LOW".

  while (!Serial); // wait for serial port to connect. Needed for Leonardo only

  /*****************************************************************/

  //Wi-Fi connection attempt
  Serial.printf("\nConnecting to %s", ssid);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED){ //printing dots till the connection is
established.
    delay(500);
    Serial.print(".");}
  //Printing out info about the Wi-Fi connection
  Serial.println("\nConnected to network");
  Serial.print("My IP address is: ");
  Serial.println(WiFi.localIP()); //prints the IP Address of the NodeMCU.

  /*****************************************************************/

  //MySQL connection attempt
  Serial.print("Connecting to SQL...  ");
  if (conn.connect(server_addr, 3306, user, password)) //here 3306 is the port of
Mysql database on our server.
    Serial.println("OK."); //successfully connected to mysql database.
  else
    Serial.println("FAILED."); //connection to mysql database failed.
  cursor = new MySQL_Cursor(&conn); //assigning the cursor to the database for
insertion operation.

  /*****************************************************************/

}

void loop()
{

  /*****************************************************************/

   //receiving data serially via rx tx lines from the gateway/receiver Loraduino,
   recvWithStartEndMarkers(); //function defined at the end of the code
   showNewData(); //function defined at the end of the code

  /*****************************************************************/

  //Parsing the received JSON format char array/string message to extract the sensor
data from it in the integer form.
   StaticJsonBuffer<200> jsonBuffer; //200 is a safe size for our message extraction
variable (we can calculate it from online JSON assistant).
   JsonObject& root = jsonBuffer.parseObject(receivedChars); //received char contains
the serially received message.
   if(!root.success()){
  Serial.println("parseObject() failed"); //parsing fails when message is empty or
isn't in standard JSON format.
  return; //restarts loop()
  }
  //now extracting and saving the received values in newly created variables.
  int tempsoil=root["tempsoil"];
  int tempair=root["tempair"];
  int humidity=root["humidity"];
```

```
  int moisture=root["moisture"];
  int ldrval=root["ldrval"];

  /*****************************************************************/

  //Turning the relay "ON" if the soil moisture drops below a threshold.
  if (moisture<40) //here 40 is the threshold.
  { digitalWrite(relaypin,LOW);
    delay(2000); //relay will be "ON" for 2 seconds before turning "OFF" again.
    digitalWrite(relaypin,HIGH);
  }

  /*****************************************************************/

  //preparing the final mysql insertion command.
  char query[150]; //a new char array for the modified mysql query.
  //now replacing "%d"s in the INSERT_SQL[] array with variable values like
tt,tempsoil... etc. and storing it in "query".
  sprintf(query, INSERT_SQL, tt,tempsoil,tempair,humidity,moisture,ldrval);

  /*****************************************************************/

  //Running our insertion query which stores the sensor values in the database.
  if (conn.connected()) //if mysql is still connected.
  cursor->execute(query); //run the insertion command in "query".
  Serial.println("Data recorded."); //data recorded in the database.

  /*****************************************************************/

  //tt is incremented in each loop and reset after 23 thus acting like an hour counter
which is fed to the database.
  tt +=1;
  if(tt==24)tt=0;

  /*****************************************************************/

  delay(5000); //waiting 5 seconds before another insertion in the database.


}

/*****************************************************************/

//a function to receive data from RX TX serial communication between the receiver
Loraduino and NodeMCU.
//it ensures synchronization by detecting the start and end markers in the message and
only saving the message that is in between the markers.
void recvWithStartEndMarkers() {

    static bool recvInProgress = false;
    static byte ndx = 0;
    char startMarker = '<';
    char endMarker = '>';
    char rc;

    while (Serial.available() > 0 && newData == false) {
         //Serial.println("into the recv function...");
        rc = Serial.read();

        if (recvInProgress == true) {
            if (rc != endMarker) {
                receivedChars[ndx] = rc;
                ndx++;
```

```
                if (ndx >= numChars) {
                    ndx = numChars - 1;
                }
            }
            else {
                receivedChars[ndx] = '\0'; // terminate the string
                recvInProgress = false;
                ndx = 0;
                newData = true;
            }
        }

        else if (rc == startMarker) {
            recvInProgress = true;
        }
    }
}

/*****************************************************************/

//a function to show the newly received data
void showNewData() {
    if (newData == true) {
        Serial.println("Received Serially: ");
        Serial.println(receivedChars);
        newData = false;
    }
}
```

Code Files For User Interface:

Please follow the following Link:

https://github.com/msw1998/Precise-agriculture-website