CHAPTER - 1

1.1 LITERATURE REVIEW

Muhammad (2010)[1], Proposed a simple approach to "Automatic Irrigation control problem using Artificial Neural Network Controller". The proposed system is compared with the ON/OFF controller, and it is shown that the ON/OFF Controller based System fails miserably because of its limitations. On the other hand, ANN based approach has resulted in possible implementation of better and more efficient control. These controllers do not require a prior knowledge of systems and have inherent ability to ANN based systems that can save a lot of resources (energy and water) and can provide optimized results to all types of agricultural areas.

Sanjukumar (2013)[2], Proposed "Advance Technique for Soil Moisture Content Based Automatic Motor Pumping for Agriculture Land Purpose" was developed and successfully implemented along with flow sensors. Salient features of the system are: Closed loop automatic irrigation system, temperature, and water usage monitoring. Users can easily preset the levels of the Moisture and are regularly updated about the current value of all Parameters on LCD display. In future, other important soil parameters namely soil pH, soil electrical conductivity will also be incorporated in the system.

S Nalini Durga (2018)[3], proposed "Smart Irrigation System Based on Soil Moisture Using Iot" Agriculture remains the sector which contributes the highest to India's GDP. But, when considering technology that is deployed in this field, we find that the development is not tremendous. Nowadays there is huge enhancement in technologies which have a significant impact on various fields like agriculture. Healthcare etc.

Anat Goldstein(2018)[4] proposed in"Applying machine learning on sensor data for irrigation recommendations", wireless sensor network is integrated with ZigBee to transmit soil moisture level and temperature values. The data is transmitted to a web server using GPRS through cellular network. The data monitoring can be achieved via internet using graphical application.

1.2 MOTIVATION OF THE PROJECT

India's population rate is increasing day by day and there will be serious problems in food, so the development of agriculture is necessary. In today's world, the farmers are facing many problems such as they do not know well about their lands, how the soil content is there and, in some season, there will be heavy rainfall which affects their already grown crops.
The main objective of this project is to provide information about their soils moisture index, humidity and temperature of the field. This may be useful for the farmers to save their crops before damage and then it is easy for every farmer to get information through their mobile phone as a message to alert them.

1.3 OBJECTIVE

In the field of agriculture, use of proper methods of irrigation is very important from a yield point of view. Man effort, Labour requirements, good yield and scarcity of water are the key issues in the agriculture field. The objective of the project is to monitor the field along with plants (crops) by sitting at one place and Receiving field information from sensors. Here the Provided sensors gives information about moisture level (Soil Moisture sensor) and other field parameters i.e. temperature and humidity which is transmitted to the mobile unit through the Bluetooth module. The Parameter Status is displayed in the android mobile. For wireless communication Bluetooth connectivity is used between famer handset and the system which transmits status of sensors as well as receives commands from handset. A camera is used to transmit live feed of the field to the farmer’s handset.

CHAPTER - 2

2.1 PROPOSED SYSTEM

In this project, two Arduino Nano as data points, an Arduino Mega as the Receiver and a Mobile Device has been used.

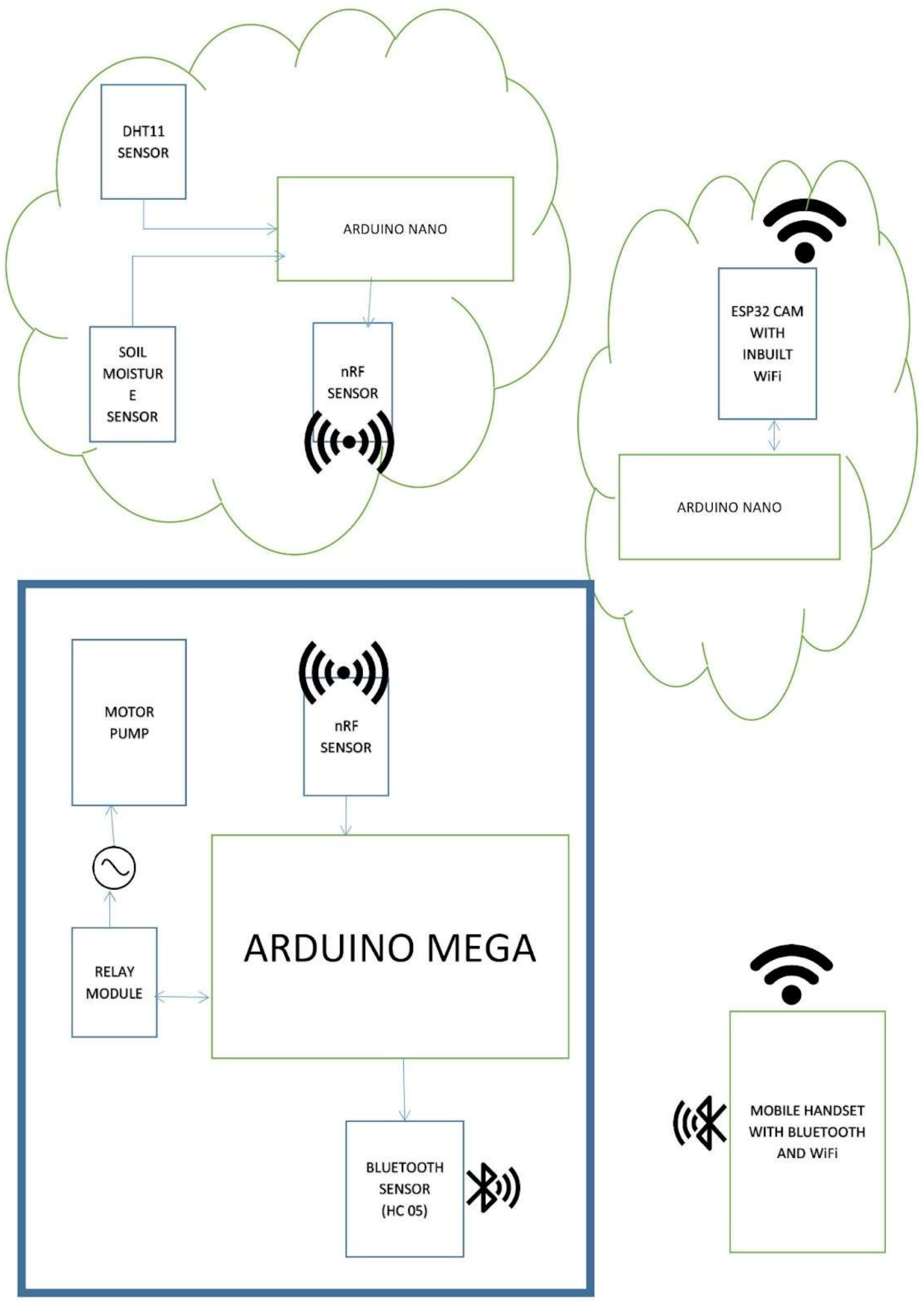
In an Arduino Nano, A soil moisture sensor, a DHT11 sensor and a NRF sensor are interfaced. These two sensors collect the values of Soil Moisture index, Temperature and Humidity respectively and transmit the value using the NRF sensor to the Arduino Mega Microcontroller.

An ESP32 Cam is interfaced with another Arduino Nano and the camera captures live footage of the field and transmits the video using the inbuilt WIFI module of the ESP32 to the mobile handset. The user can monitor the field using the camera during the day and night.

The Arduino Mega receives the sensor values from Arduino Nano using the NRF sensor, compares the values to the given threshold value and transmits the values to the handset using the Bluetooth sensor interfaced with the Arduino Mega. If the value of the soil moisture sensor is less than the threshold value, the relay module detects that soil is wet and turns off the pump. In the case of dry soil i.e., soil moisture sensor value is greater than the threshold value, the relay module will turn on thus the motor pump will turn on. If the temperature and humidity values of the sensor DHT11 is greater than the threshold value, another pump will turn on using the relay and the water will sprinkle over the leaves of the plants.

The user will receive the data of the all-sensor values and the status of the motor from Arduino Mega in this handset using the Bluetooth serial monitor and the live feed of the field from the Arduino Nano connected with ESP32 from via the static IP address given the ESP32 module.

2.2 BLOCK DIAGRAM OF THE SYSTEM



2.3 COMPONENTS

- Arduino
  - Arduino Mega
  - Arduino Nano
- Soil Moisture Sensor
- Temperature & Humidity Sensor (DHT11)
- Bluetooth sensor (HC-05)
- Transceiver nRF24L01
- Relays
- ESP 32 CAM
- Motor Pump
- Sprinkle Motor Pump

Arduino:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming.

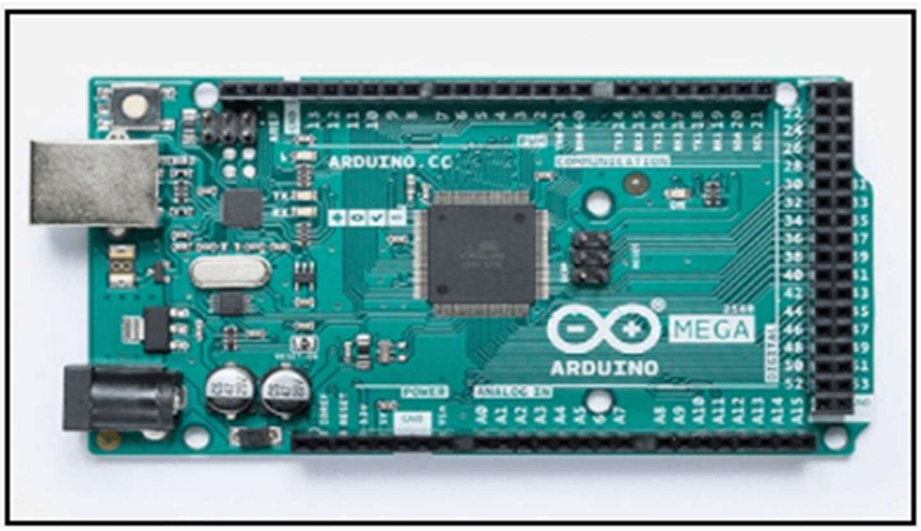
The devices required to start our projects using the Arduino board are Arduino IDE and mini-USB. The Arduino IDE software must be installed on our respected laptop or desktop. The mini-USB transfers the code from the computer to the Arduino board.

Here we have used Arduino Mega and Arduino Nano microcontrollers.

Arduino Mega:

The Arduino Mega is based on the ATmega2560 Microcontroller. The ATmega2560 is an 8-bit microcontroller. We need a simple USB cable to connect to the computer and the AC to DC adapter or battery to get started with it.

The Arduino Mega board is shown below:



Technical Specifications of Arduino Mega

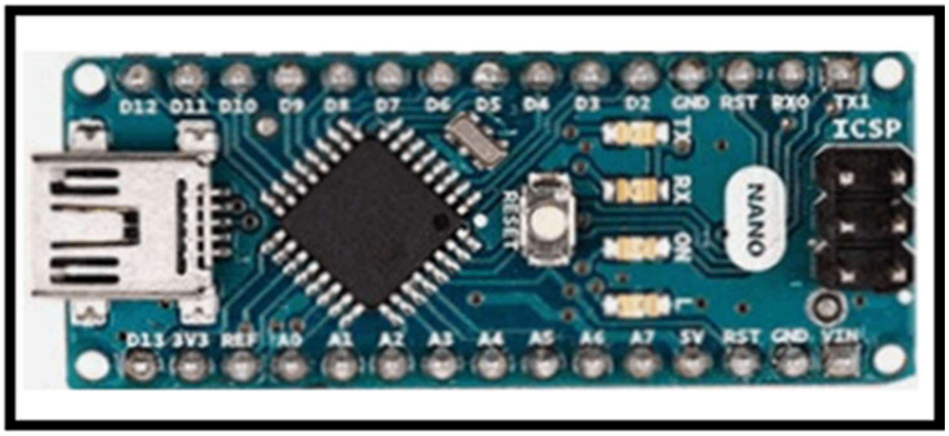
The technical specifications of Arduino Mega are listed below:

- There are 54 Input/Output digital pins and 16 Analog Input/Output (I/O) present on the Mega board.
- The 15 pins from the 54 digital I/O pins are PWM output pins. The PWM pins are Pulse Width Modulation capable pins.
- The input voltage of the Mega board varies from 7V to 12V.
- The crystal oscillator present in Arduino Mega comes with a frequency of 16MHz.
- The operating voltage of the Arduino Mega is 5 V.
- It is generally used to create complex projects due to its structure.
- The projects that use Arduino Mega board are IOT Applications, 3D Printers, temperature sensing, monitoring of real-time applications, etc.

Arduino Nano:

The Arduino Nano is a small Arduino board based on the ATmega328P or ATmega628 Microcontroller. The connectivity is the same as the Arduino UNO board.

The Nano board is defined as a sustainable, small, consistent, and flexible microcontroller board. It is small in size compared to the UNO board. The


D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
ICSP
TX
RX
RESET
NANO
ON
L
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN

Technical Specifications of Arduino Nano

The technical specifications of Arduino nano are listed below:

- Microcontroller: Microchip ATmega328P
- Operating voltage: 5 volts
- Input voltage: 6 to 20 volts
- Digital I/O pins: 14 (6 optional PWM outputs)
- Analog input pins: 8
- DC per I/O pin: 40 mA
- DC for 3.3 V pin: 50 mA
- Flash memory: 32 KB, of which 0.5 KB is used by bootloader
- Clock speed: 16 MHz

Soil Moisture Sensor:

The moisture of the soil plays an essential role in the irrigation field as well as in gardens for plants. As nutrients in the soil provide the food to the plants for their growth. Extreme soil moisture levels can guide to anaerobic situations that can encourage the plant's growth as well as soil pathogens. This article discusses an overview of the soil moisture sensor, working and its applications.

The soil moisture sensor is one kind of sensor used to gauge the volumetric content of water within the soil. As the straight gravimetric dimension of soil moisture needs eliminating, drying, as well as sample weighting.

The relation among the calculated property as well as moisture of soil should be adjusted & may change based on ecological factors like temperature, type of soil, otherwise electric conductivity.

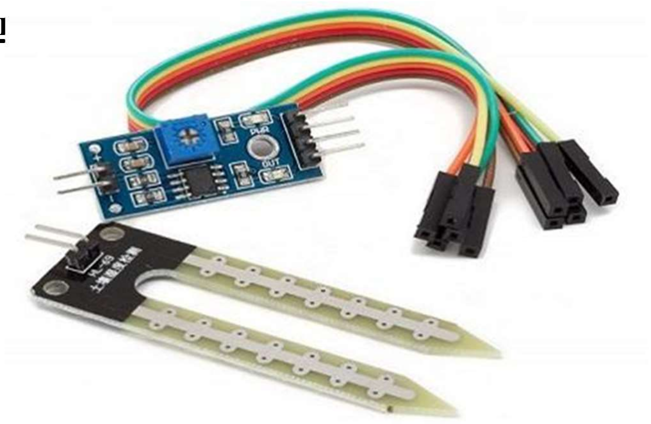
These sensors are normally used to check volumetric water content, and another group of sensors calculates a new property of moisture within soils named water potential. Generally, these sensors are named as soil water potential sensors which include gypsum blocks and tensiometers.

This module also includes a potentiometer that will fix the threshold value, & the value can be evaluated by the comparator-LM393. The LED will turn on/off based on the threshold value

Soil Moisture Sensor Pin Configuratio

The FC-28 soil moisture sensor includes 4pins

- VCC pin is used for power
- A0 pin is an analog output
- D0 pin is a digital output
- GND pin is a Ground



Specifications:

- The required voltage for working is 5V
- The required current for working is <20mA
- Type of interface is analog.
- The required working temperature of this sensor is 10°C~30°C

Temperature & Humidity Sensor (DHT11):

DHT11 is a low-cost digital sensor for sensing temperature and humidity. This sensor can be easily interfaced with any micro-controller such as Arduino, Raspberry Pi etc… to measure humidity and temperature instantaneously.

DHT11 humidity and temperature sensor is available as a sensor and as a module. The difference between this sensor and module is the pull-up resistor and a power-on LED. DHT11 is a relative humidity sensor. To measure the surrounding air this sensor uses a thermistor and a capacitive humidity sensor.

Working Principle of DHT11 Sensor:

DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measures, processes these changed resistance values and changes them into digital form.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature. To get a larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.

The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. Humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz.i.e., it gives one reading for every second. DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.

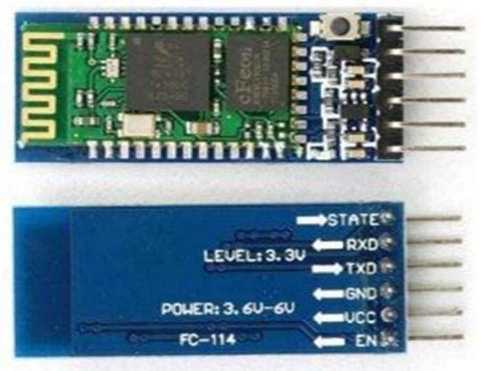
The DHT11 sensor has four pins- VCC, GND, Data Pin and a not connected pin. A pull-up resistor of 5k to 10k ohms is provided for communication between sensor and microcontroller.

Bluetooth Sensor:

Bluetooth is a technology for wireless communication. It is designed to replace cable connections. It uses serial communication to communicate with devices. It communicates with a microcontroller using a serial port (USART). Usually, it connects small devices like mobile phones, PDAs and TVs using a short-range wireless connection to exchange documents. The maximum range is 10 meters. The transfer rate of the data is 1Mbps.

HC-05 Specifications:

- Bluetooth protocol: Bluetooth Specification v2.0+EDR
- Frequency: 2.4GHz ISM band
- Modulation: GFSK(Gaussian Frequency Shift Keying)
- Emission power: ≤4dBm, Class 2
- Sensitivity: ≤-84dBm at 0.1% BER
- Speed: Asynchronous: 2.1Mbps(Max) / 160 kbps, Synchronous: 1Mbps/1Mbps
- Security: Authentication and encryption
- Profiles: Bluetooth serial port
- Power supply: +3.3VDC 50mA
- Working temperature: -20 ~ +75 Centigrade
- Dimension: 26.9mm x 13mm x 2.2 mm



Pin Description:

1. Key/EN: It is used to bring Bluetooth modules in AT commands mode. By default, this pin operates in data mode. Key/EN pin should be high to operate Bluetooth in command mode. The default baud rate of HC-05 in command mode is 38400 bps and 9600 in data mode.

2. VCC: Connect 5 V or 3.3 V to this Pin.

3. GND: Ground Pin of module.

4. TXD: Connect with Microcontroller RXD pin of Microcontroller. Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)

5. RXD: Connect with Microcontroller TXD pin of Microcontroller. Received data will be transmitted wirelessly by Bluetooth module.

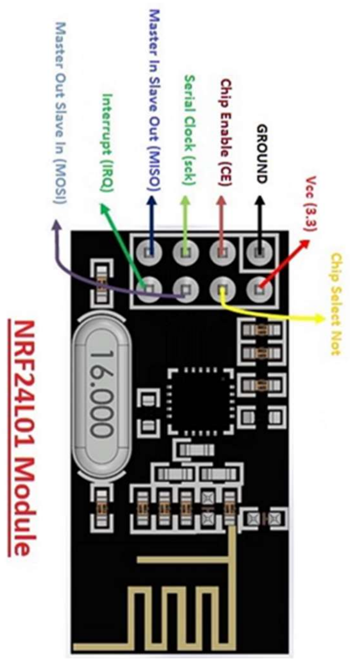
NRF24L01:

The nRF24L01 module is a very popular choice for wireless communication when using Arduino.

The power consumption of this module is just around 12mA during transmission, which is even lower than a single LED. The operating voltage of the module is from 1.9 to 3.6V, but the good thing is that the other pins tolerate 5V logic, so we can easily connect it to an Arduino without using any logic level converters.

Three of these pins are for the SPI communication and they need to be connected to the SPI pins of the Arduino, but note that each Arduino board has different SPI pins. The pins CSN and CE can be connected to any digital pin of the Arduino board and they are used for setting the module in standby or active mode, as well as for switching between transmit or command mode. The last pin is an interrupt pin which doesn't have to be used.

Features of NRF24L01:



- It is a single chip GFSK transceiver.
- It has a complete OSI hardware layer.
- It has an on-air data rate of 1 to 2 Mbps.
- Its operation is 124 RF channels.
- It is fully compatible with nRF24XX.
- It has a 20-pin package (QFN 20 4×4 mm).
- Its power supply range is 1.9 to 3.6 V.
- Its nominal current is 50 mA. Its operating current is 250 mA.
- It uses SPI protocol for communication.
- Its baud rate is 250 kbps to 2 Mbps.
- Its channel range is 125.
- Its Maximum Pipeline or node is six.
- It is a low-cost wireless solution.
- Its antenna can send and receive data up to 250 kb and it can cover a distance of 100 meters.
- Its operating temperature is -40°C to 85°C and storage is 40°C to 125°C.
- It has a PA gain of 20 dB and LNA gain of 10 dB.
- Its maximum output power is +20 DBm.

THE RELAY MODULE:

The module with the SRD-05VDC-SL-C relay has three high voltage terminals (NC, C, and NO) which connect to the device you want to control. The other side has three low voltage pins (Ground, VCC, and Signal) which connect to the Arduino.

- NC: Normally closed 120-240V terminal
- NO: Normally open 120-240V terminal
- C: Common terminal
- Ground: Connects to the ground pin on the Arduino
- 5V VCC: Connects the Arduino's 5V pin
- Signal: Carries the trigger signal from the Arduino that activates the relay

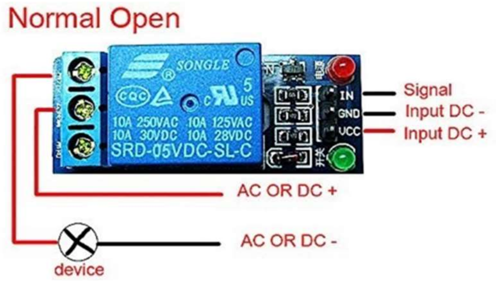
Inside the relay is a 120-240V switch that's connected to an electromagnet. When the relay receives a HIGH signal at the signal pin, the electromagnet becomes charged and moves the contacts of the switch open or closed.
It works in two modes Normally Open or Normally Closed:

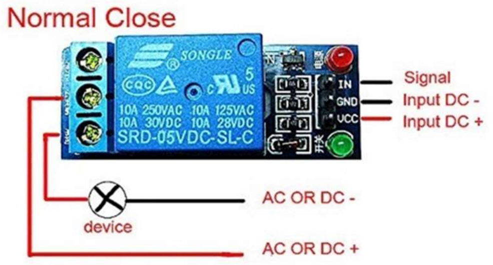
Normally Open

In the normally open configuration, when the relay receives a HIGH signal the 120-240V switch closes and allows current to flow from the C terminal to the NO terminal. A LOW signal deactivates the relay and stops the current. So, if you want the HIGH signal to turn ON the relay, use the normally open terminal.

Normally Closed

In the normally closed configuration, a HIGH signal opens the switch and interrupts the 120-240V current. A LOW signal closes the switch and allows current to flow from the C terminal to the NC terminal. Therefore, if you want the HIGH signal to turn OFF the 120-240V current, use the normally closed terminal.





ESP32CAM:

ESP32 is the name of the chip that was developed by Espressif Systems. This provides Wi-Fi (and in some models) dual-mode Bluetooth connectivity to embedded devices. While ESP32 is technically just the chip, modules and development boards that contain this chip are often also referred to as "ESP32" by the manufacturer. The differences between these are explained further on in the article.
The ESP32 chip has a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, with a clock rate of over 240MHz.
Models are available with combined Wi-Fi and Bluetooth connectivity, or just Wi-Fi connectivity.
The ESP32 is most commonly engineered for mobile devices, wearable tech, and IoT applications – such as Nabto. Moreover, with Mongoose OS introducing an ESP32 IoT Starter Kit, the ESP32 has gained a reputation as

ESP32 Functions:

- **Data Processing:** Includes processing basic inputs from analog and digital sensors to far more complex calculations with an RTOS or Non-OS SDK.
- **P2P Connectivity:** Creates direct communication between different ESPs and other devices using IoT P2P connectivity.
- **Web Server:** Access pages written in HTML or development languages.

ESP32 Applications:

- The ESP32 modules are commonly found in the following IoT devices:
- Smart industrial devices, including Programmable Logic Controllers (PLCs)
- Smart medical devices, including wearable health monitors
- Smart energy devices, including HVAC and thermostats Smart security devices, including surveillance cameras and smart locks
- Many international markets require shielded Wi-Fi devices, as Wi-Fi produces a lot of Radio Frequency Interference (RFI), and shielding minimizes this interference. This should, therefore, be a key consideration for all developers and embedded device manufacturers.

2.4 WORKING PRINCIPLE

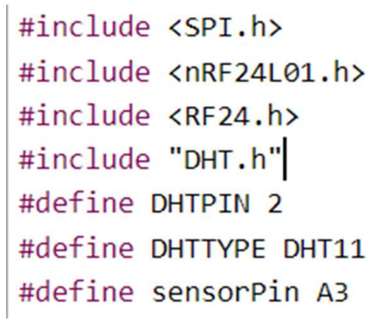
This project is designed to develop an automatic irrigation system which switches the motor pump ON/OFF on sensing the moisture content of the soil and monitors the real time temperature and humidity of the farm and starts another motor if the temperature or humidity is too high. In the field of agriculture, the use of proper methods of irrigation is important. The advantage of using this method is to reduce human intervention and still ensure proper irrigation. In this project the Moisture sensor, Temperature and humidity sensor, Motor pump are interfaced with Arduino. Moisture sensor and temperature sensor provides analog signal to microcontroller. Microcontroller needs to convert this analog signal into digital value.

The status of the water pump, soil moisture and temperature are displayed on the phone, which is interfaced to the microcontroller. Thus, this automatic irrigation system depends on the output of the moisture sensor and DHT11 (Temperature and humidity sensor). When the moisture sensor senses the soil is dry then the microcontroller turns ON the motor pump and if the moisture sensor senses the soil is moist enough then the microcontroller turns OFF the motor pump. Transmitter nrf24 is connected in the Arduino Nano. The soil moisture index, humidity and temperature are transmitted by the nrf24 sensor which is connected in Arduino Nano. In the receiver side another nRF24 and a Bluetooth sensor (HC-05) is connected with Arduino Mega. And based on the mechanism microcontroller controls the motor pump based on the weather. Bluetooth module sends the data (soil moisture index, humidity, temperature) to the mobile phone.

For the surveillance of the field ESP32 WIFI camera module is used in the system. This WIFI module is interfaced with another Arduino nano, a camera is used with this module. The WIFI module used the mobile hotspot or LAN WIFI and it became activated and creates a network. The camera part sends the image and video in the surveillance device. The WIFI module creates an IP address and it can be accessed by any device.

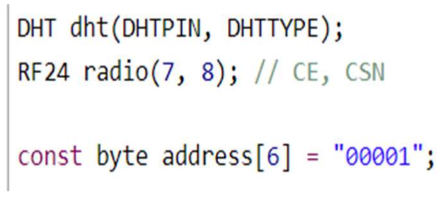
The Transmitter Side:

For Sensor Values: Arduino Nano is being used as data points in the fields. From where data is collected and sent to the receiver side Arduino Mega. First SPI library is defined, which is Serial Peripheral Interface (SPI), synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances.
For communication between the Nano and Mega Radio Frequency Transceiver is used, i.e., nRF24L01 sensor which can Transmit and receive data.
Temperature and humidity of the air is being recorded from a DHT11 sensor. For that purpose, first of all DHT library and which type of DHT is defined. A sensor pin#2 is defined for taking input of the DHT data as "DHTPIN 2".
For recording the value of Soil Moisture sensor an analog pin#3 is used as a "sensorPin".



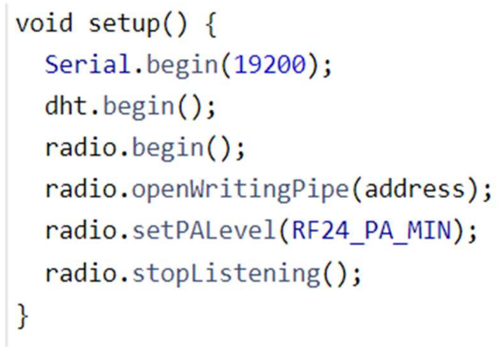
```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
#define sensorPin A3
```

Then the dht parameter is defined to initialize the DHT sensor.
The radio of the RF module is initialized by the CSN and CE pins connected to digital pin#7 and #8 respectively. The address is defined to communicate with another RF module using a unique string as the address array.



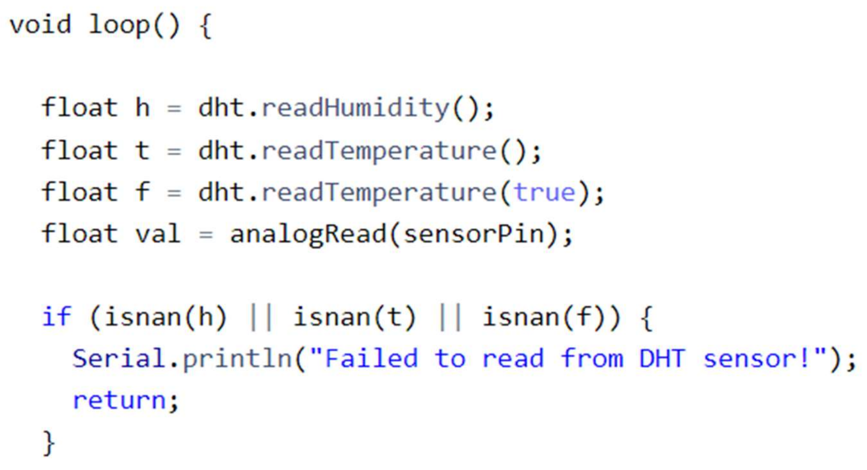
```
DHT dht(DHTPIN, DHTTYPE);
RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001";
```

Inside the void setup function, initialize a Serial Communication at the baud rate of 19200. Then the DHT sensor, radio of RF sensor got initialized. Using the radio.openWritingPipe() function, address of the transmitter is set.
Then the power amplification level is set to MIN for closer range work. When the RF modules are set in a long-distance range PA Level should be set to MAX.
Finally, using the radio.stopListening() function, the module sets as a transmitter.



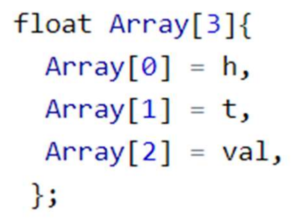
```
void setup() {
  Serial.begin(19200);
  dht.begin();
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();
}
```

In the void loop codes will run over and over.
Float variables are initialized and stored with the values of humidity, temperature in Celsius and Fahrenheit scale. The “val” variable is storing the soil moisture pin data.
In the if loop, if any of the sensor data isn’t available, it will print.
“Failed to read from DHT sensor!” and the loop will not execute.



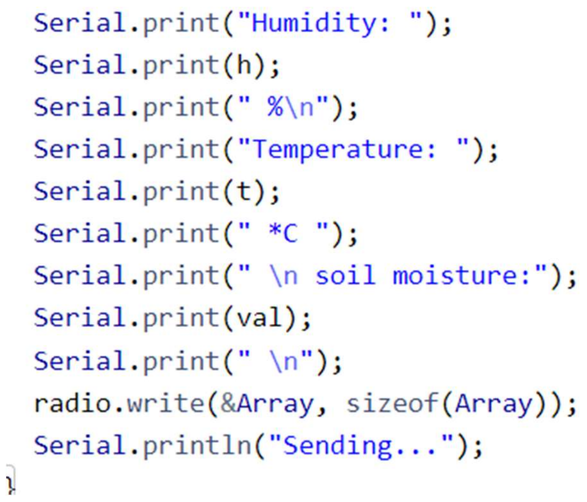
```
void loop() {

  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  float val = analogRead(sensorPin);

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

An Array of size 3 is declared for storing the humidity, temperature and soil moisture sensor value.



```
float Array[3]{
  Array[0] = h,
  Array[1] = t,
  Array[2] = val,
 };
```

Now each sensor value is printed in the serial monitor via the command Serial.print and send the array to the Arduino Mega using the nRF24 sensor via the command radio.write.

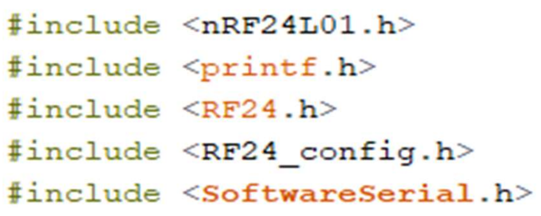


```
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\n");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(" \n soil moisture:");
  Serial.print(val);
  Serial.print(" \n");
  radio.write(&Array, sizeof(Array));
  Serial.println("Sending...");
}
```

The Transmitter Side (For ESP32 Cam):
ESP32 cam module is enabled in the download mode by shorting the GND and GPIO0 pin. Then the inbuilt Webcam Server is uploaded to the ESP32. After releasing the pins, the live feed is streaming successfully.

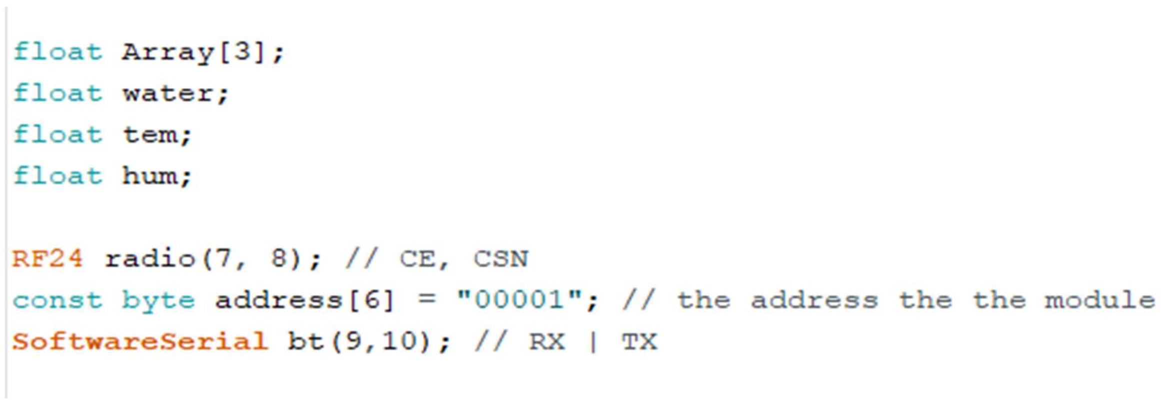
The Receiver Side:

Arduino Mega receives the signal from the transmitter side and analyses the data and turns the pump on or off and finally sends the data to the mobile handset.
nRF24 is being used for the communication between the Arduino Mega and the Arduino Nano. So the nRF libraries were included at first. For the Bluetooth module the “SoftwareSerial.h” library is used.



```
#include <nRF24L01.h>
#include <printf.h>
#include <RF24.h>
#include <RF24_config.h>
#include <SoftwareSerial.h>
```

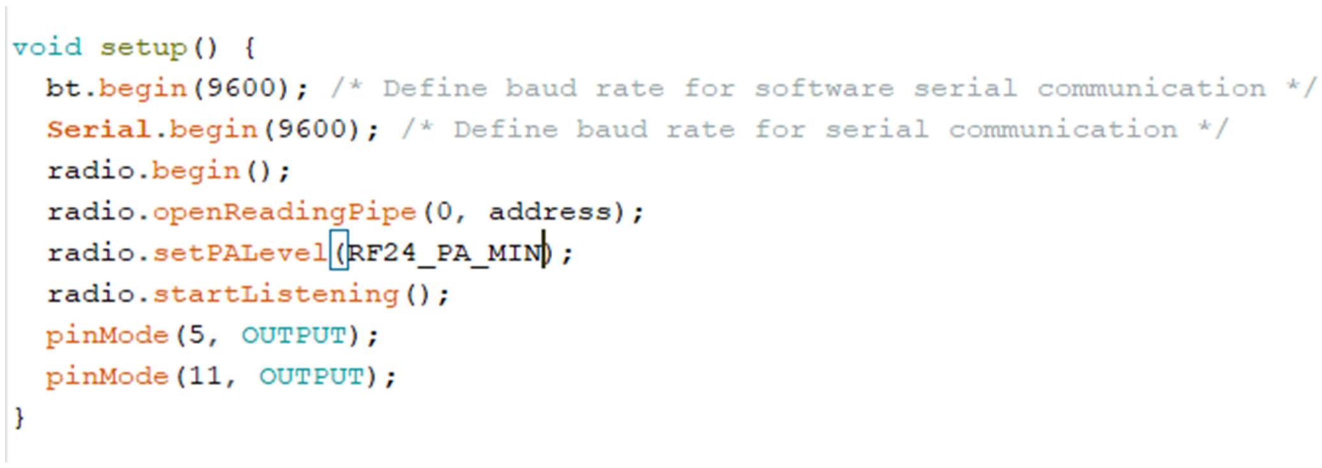
Then an array of float values with size 3 and 3 float variable named “water”,”tem”, and “hum” is declared.
The radio of the RF module is initialized by the CSN and CE pins connected to digital pin#7 and #8 respectively. The address is defined to communicate with another RF module using a unique string as the address array.
The “SoftwareSerial” library allows serial communication on other digital pins of an Arduino board, using software to replicate the functionality. The parameters in the bracket defining the RX and TX pins.



```
float Array[3];
float water;
float tem;
float hum;

RF24 radio(7, 8); // CE, CSN
const byte address[6] = "00001"; // the address the the module
SoftwareSerial bt(9,10); // RX | TX
```

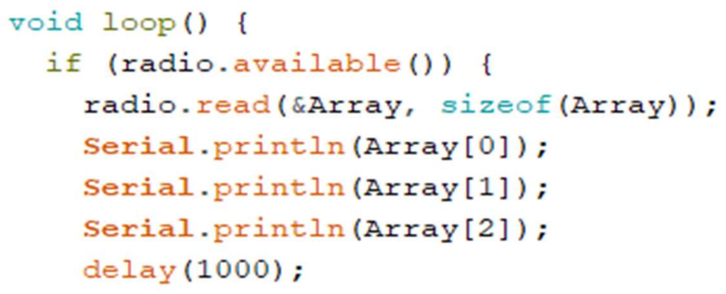
Inside the void setup function, a Bluetooth serial communication at baud rate 9600 and a Serial Communication at the baud rate of 9600 is initialized. Then

the RF sensor got initialized. The radio.openWritingPipe() function is used to set the address of the transmitter.
Then the power amplification level is set to MIN for closer range work. When the RF modules are set in a long-distance range PA Level should be set to MAX. Then the radio.startListening() function which sets the module as a receiver is used.
Finally, the 5 and 11 pin is used as output to operate the Relay module.



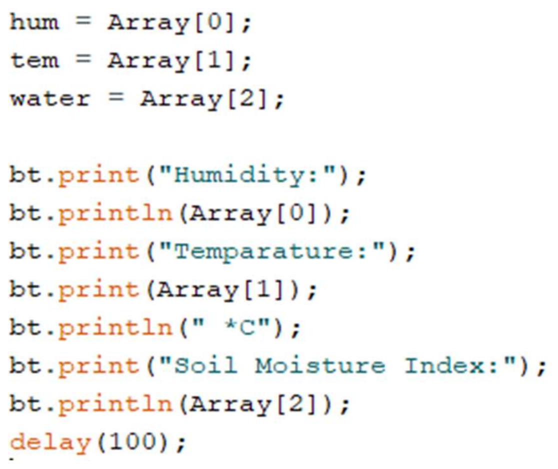
```
void setup() {
  bt.begin(9600); /* Define baud rate for software serial communication */
  Serial.begin(9600); /* Define baud rate for serial communication */
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();
  pinMode(5, OUTPUT);
  pinMode(11, OUTPUT);
}
```

In the void loop the codes will run over and over.
If there is radio available in the NRF sensor, the NRF sensor reads the sent Array and its size from the Arduino Nano using the radio.read command.
Then it will print the values of the received Array in the Serial Monitor using the Serial.println command, followed by a delay of 1 second.



```
void loop() {
  if (radio.available()) {
    radio.read(&Array, sizeof(Array));
    Serial.println(Array[0]);
    Serial.println(Array[1]);
    Serial.println(Array[2]);
    delay(1000);
```

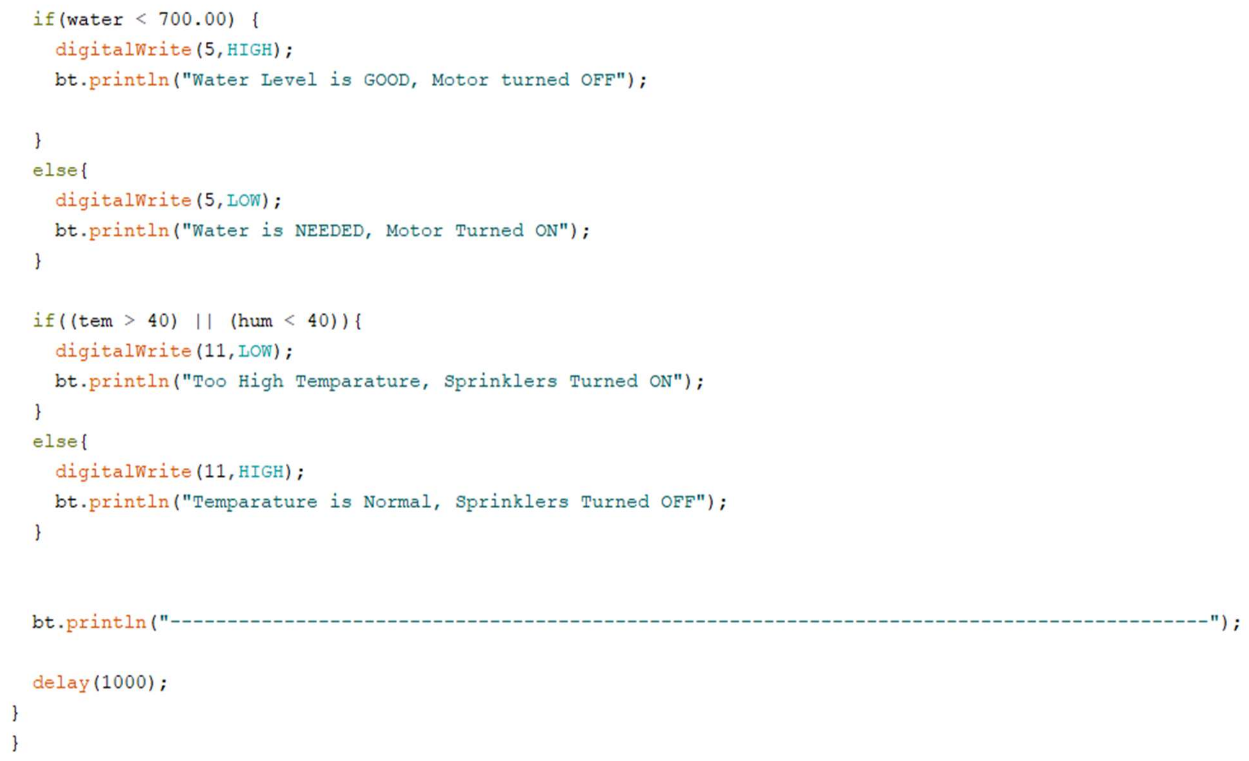
Now, the values of the Array stored in the variables declared above. Then to print the sensor value in the Bluetooth serial monitor, the bt.println command is used, followed by a delay of 0.1 second.



```
hum = Array[0];
tem = Array[1];
water = Array[2];

bt.print("Humidity:");
bt.println(Array[0]);
bt.print("Temparature:");
bt.print(Array[1]);
bt.println(" *C");
bt.print("Soil Moisture Index:");
bt.println(Array[2]);
delay(100);
```

In the 1st if loop, if the water level is below 700 then the pump will turn on by making the pin low i.e., providing low voltage to the pin. Else the pump will on.

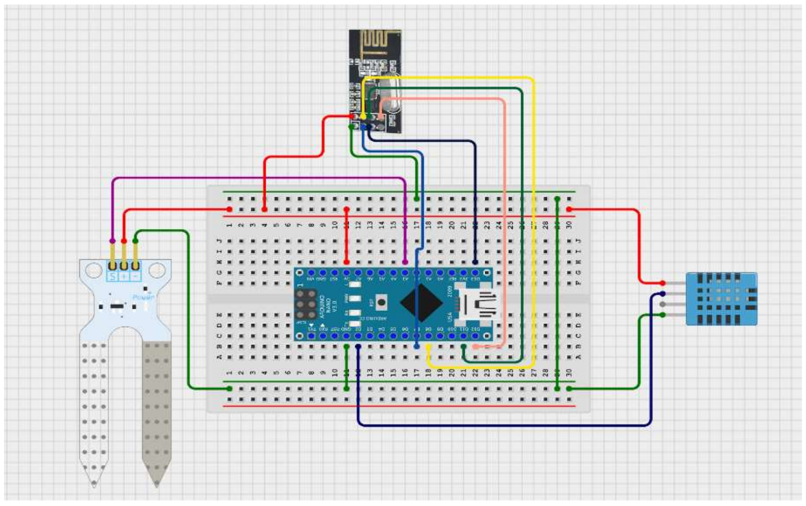
And if the temperature and humidity is over 40*c and less than 40% respectively, another pump will turn on that will sprinkle the water over the plant leaves.



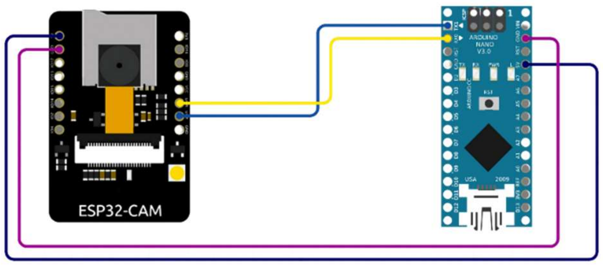
```
  if(water < 700.00) {
    digitalWrite(5,HIGH);
    bt.println("Water Level is GOOD, Motor turned OFF");

  }
  else{
    digitalWrite(5,LOW);
    bt.println("Water is NEEDED, Motor Turned ON");
  }

  if((tem > 40) || (hum < 40)){
    digitalWrite(11,LOW);
    bt.println("Too High Temparature, Sprinklers Turned ON");
  }
  else{
    digitalWrite(11,HIGH);
    bt.println("Temparature is Normal, Sprinklers Turned OFF");
  }


  bt.println("------------------------------------------------------------------------------------------");

  delay(1000);
}
}
```

2.5 CIRCUIT DIAGRAM

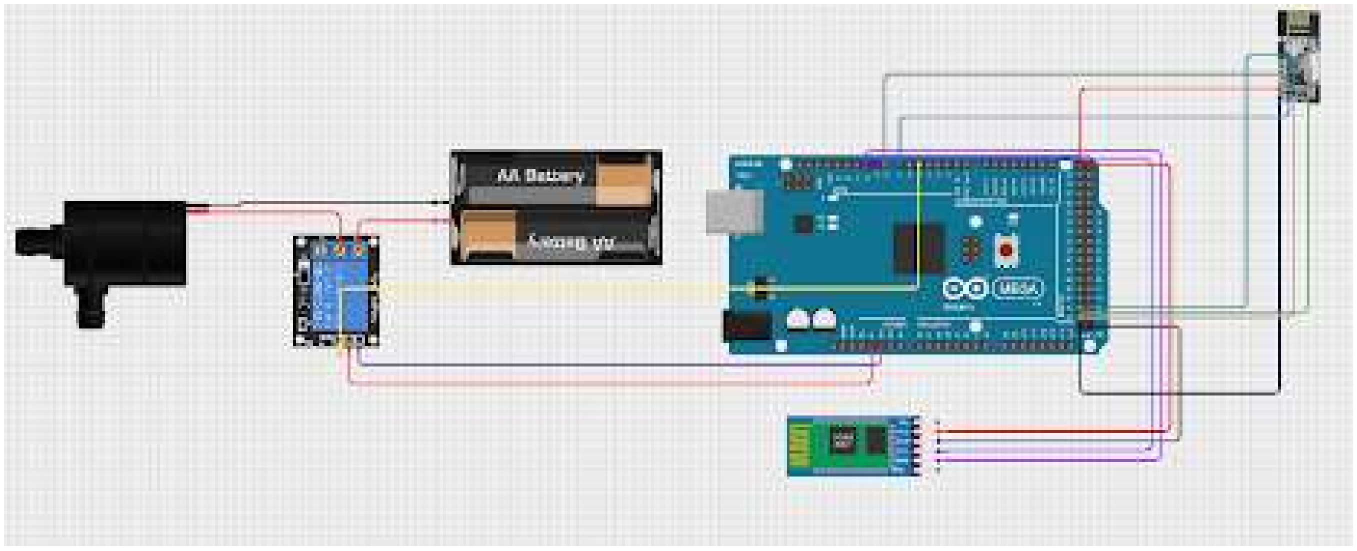
Connection with Arduino Nano at the Transmitter End:



Connection with Arduino Nano at the Transmitter End with ESP32:



Connection with Arduino Mega at the Receiver End:

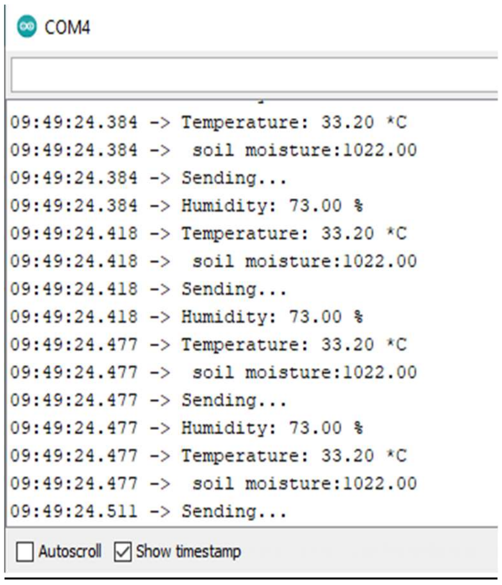


2.6 Results

Case	Status	Soil Moisture Index	Humidity	Temperature (°C)	Motor Pump Status	Sprinkler Pump Status
1	Normal Temperature & Dry Soil	1022	73%	33.20	ON	OFF
2	Normal Temperature & Semi-Wet Soil	617	73%	33.20	OFF	OFF
3	Normal Temperature & Wet Soil	280	72%	33.80	OFF	OFF
4	High Temperature & Dry Soil	1022	48%	43.30	ON	ON

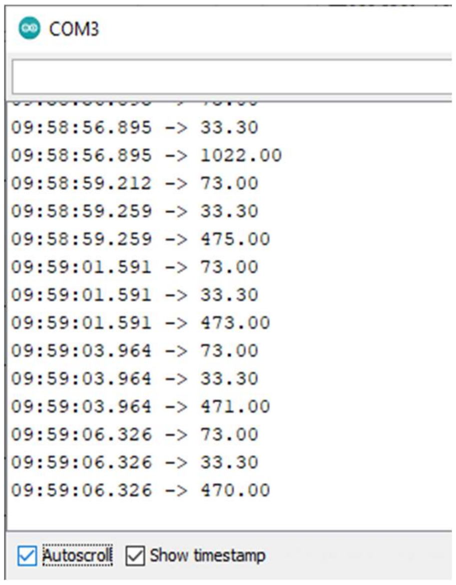
Case 1:

At Transmitter Side:



```
09:49:24.384 -> Temperature: 33.20 *C
09:49:24.384 ->  soil moisture:1022.00
09:49:24.384 -> Sending...
09:49:24.384 -> Humidity: 73.00 %
09:49:24.418 -> Temperature: 33.20 *C
09:49:24.418 ->  soil moisture:1022.00
09:49:24.418 -> Sending...
09:49:24.418 -> Humidity: 73.00 %
09:49:24.477 -> Temperature: 33.20 *C
09:49:24.477 ->  soil moisture:1022.00
09:49:24.477 -> Sending...
09:49:24.477 -> Humidity: 73.00 %
09:49:24.477 -> Temperature: 33.20 *C
09:49:24.477 ->  soil moisture:1022.00
09:49:24.511 -> Sending...
```

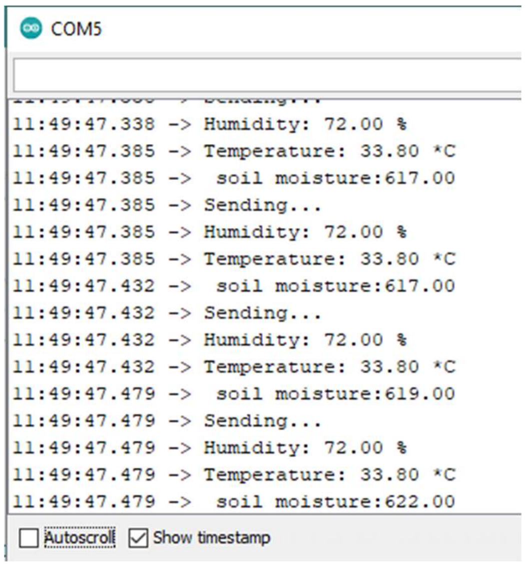
At Receiver Side:



```
09:58:56.895 -> 33.30
09:58:56.895 -> 1022.00
09:58:59.212 -> 73.00
09:58:59.259 -> 33.30
09:58:59.259 -> 475.00
09:59:01.591 -> 73.00
09:59:01.591 -> 33.30
09:59:01.591 -> 473.00
09:59:03.964 -> 73.00
09:59:03.964 -> 33.30
09:59:03.964 -> 471.00
09:59:06.326 -> 73.00
09:59:06.326 -> 33.30
09:59:06.326 -> 470.00
```

Case 2:

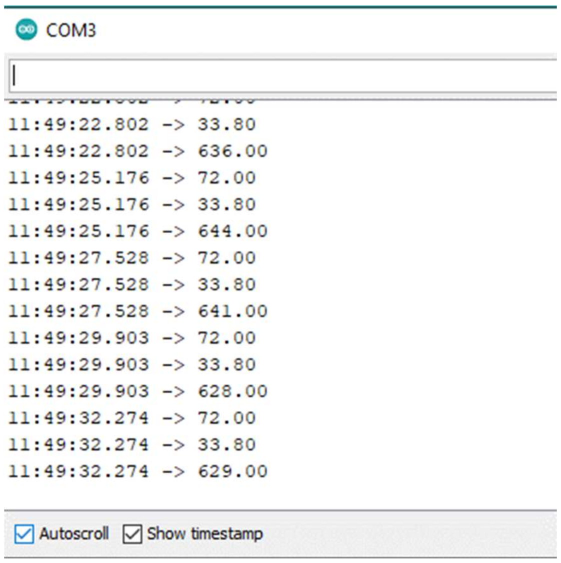
At Transmitter Side:


COM5

```
11:49:47.338 -> Humidity: 72.00 %
11:49:47.385 -> Temperature: 33.80 *C
11:49:47.385 ->  soil moisture:617.00
11:49:47.385 -> Sending...
11:49:47.385 -> Humidity: 72.00 %
11:49:47.385 -> Temperature: 33.80 *C
11:49:47.432 ->  soil moisture:617.00
11:49:47.432 -> Sending...
11:49:47.432 -> Humidity: 72.00 %
11:49:47.432 -> Temperature: 33.80 *C
11:49:47.479 ->  soil moisture:619.00
11:49:47.479 -> Sending...
11:49:47.479 -> Humidity: 72.00 %
11:49:47.479 -> Temperature: 33.80 *C
11:49:47.479 ->  soil moisture:622.00
```

☐ Autoscroll ✓ Show timestamp

At Receiver Side:

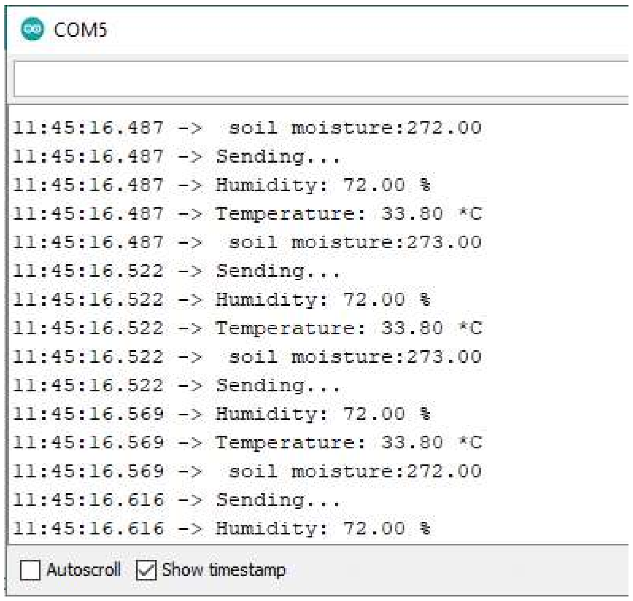

COM3

```
11:49:22.802 -> 33.80
11:49:22.802 -> 636.00
11:49:25.176 -> 72.00
11:49:25.176 -> 33.80
11:49:25.176 -> 644.00
11:49:27.528 -> 72.00
11:49:27.528 -> 33.80
11:49:27.528 -> 641.00
11:49:29.903 -> 72.00
11:49:29.903 -> 33.80
11:49:29.903 -> 628.00
11:49:32.274 -> 72.00
11:49:32.274 -> 33.80
11:49:32.274 -> 629.00
```

✓ Autoscroll ✓ Show timestamp

Case 3:

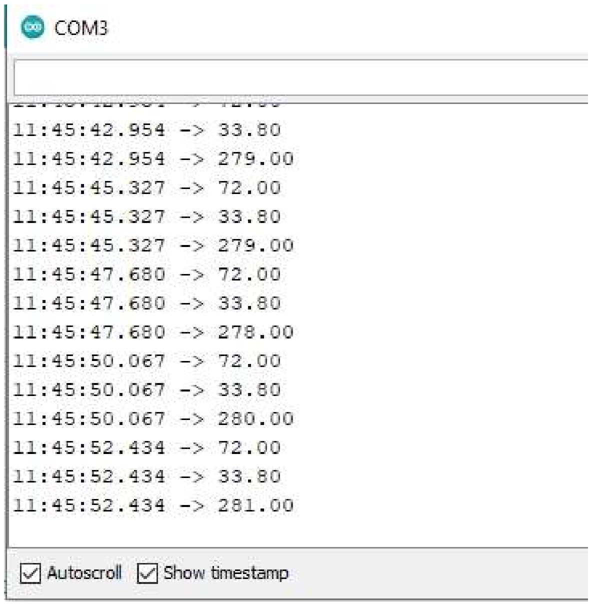
At Transmitter Side:


COM5

```
11:45:16.487 ->  soil moisture:272.00
11:45:16.487 -> Sending...
11:45:16.487 -> Humidity: 72.00 %
11:45:16.487 -> Temperature: 33.80 *C
11:45:16.487 ->  soil moisture:273.00
11:45:16.522 -> Sending...
11:45:16.522 -> Humidity: 72.00 %
11:45:16.522 -> Temperature: 33.80 *C
11:45:16.522 ->  soil moisture:273.00
11:45:16.522 -> Sending...
11:45:16.569 -> Humidity: 72.00 %
11:45:16.569 -> Temperature: 33.80 *C
11:45:16.569 ->  soil moisture:272.00
11:45:16.616 -> Sending...
11:45:16.616 -> Humidity: 72.00 %
```

☐ Autoscroll ✓ Show timestamp

At Receiver Side:

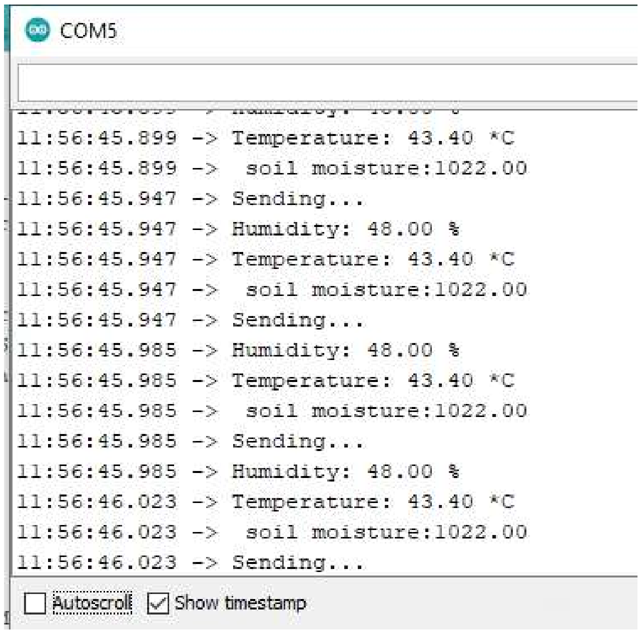

COM3

```
11:45:42.954 -> 33.80
11:45:42.954 -> 279.00
11:45:45.327 -> 72.00
11:45:45.327 -> 33.80
11:45:45.327 -> 279.00
11:45:47.680 -> 72.00
11:45:47.680 -> 33.80
11:45:47.680 -> 278.00
11:45:50.067 -> 72.00
11:45:50.067 -> 33.80
11:45:50.067 -> 280.00
11:45:52.434 -> 72.00
11:45:52.434 -> 33.80
11:45:52.434 -> 281.00
```

✓ Autoscroll ✓ Show timestamp

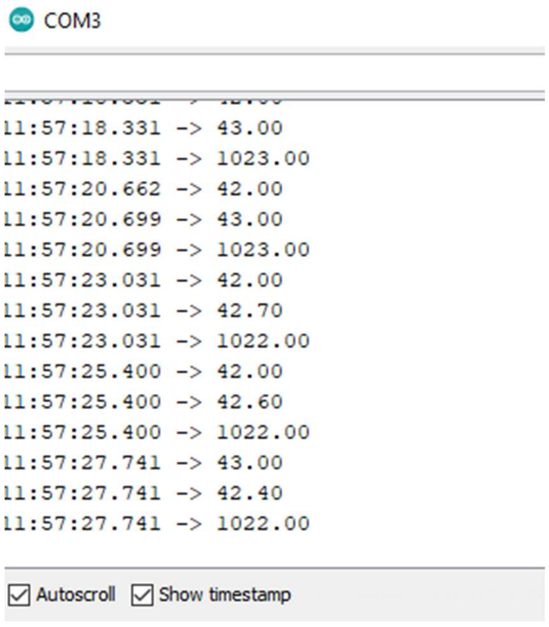
Case 4:

At Transmitter Side:



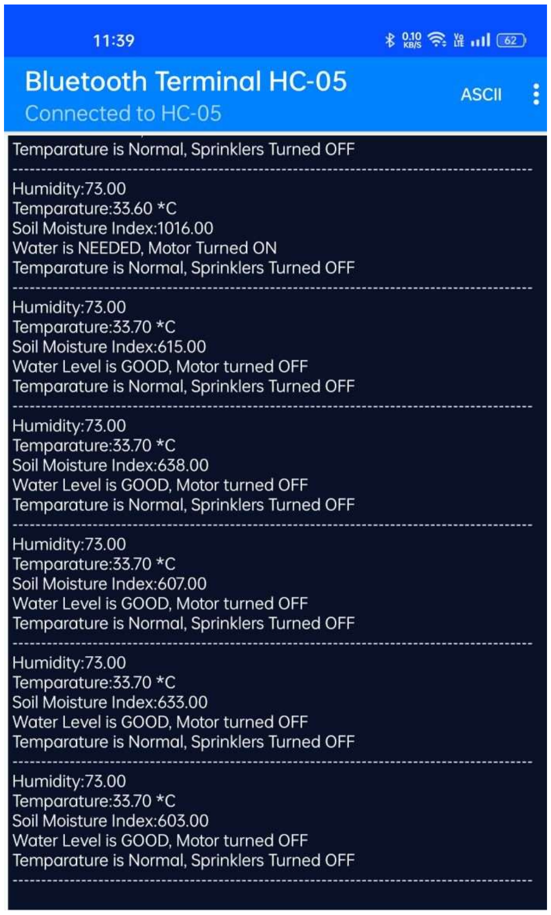
```
COM5

11:56:45.899 -> Temperature: 43.40 *C
11:56:45.899 ->  soil moisture:1022.00
11:56:45.947 -> Sending...
11:56:45.947 -> Humidity: 48.00 %
11:56:45.947 -> Temperature: 43.40 *C
11:56:45.947 ->  soil moisture:1022.00
11:56:45.947 -> Sending...
11:56:45.985 -> Humidity: 48.00 %
11:56:45.985 -> Temperature: 43.40 *C
11:56:45.985 ->  soil moisture:1022.00
11:56:45.985 -> Sending...
11:56:45.985 -> Humidity: 48.00 %
11:56:46.023 -> Temperature: 43.40 *C
11:56:46.023 ->  soil moisture:1022.00
11:56:46.023 -> Sending...
```

At Receiver Side:



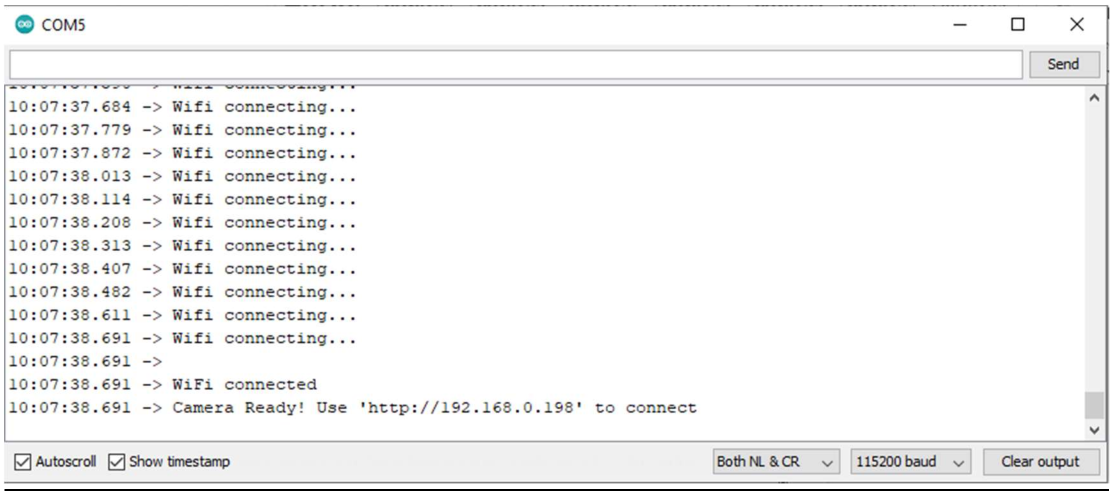
```
COM3

11:57:18.331 -> 43.00
11:57:18.331 -> 1023.00
11:57:20.662 -> 42.00
11:57:20.699 -> 43.00
11:57:20.699 -> 1023.00
11:57:23.031 -> 42.00
11:57:23.031 -> 42.70
11:57:23.031 -> 1022.00
11:57:25.400 -> 42.00
11:57:25.400 -> 42.60
11:57:25.400 -> 1022.00
11:57:27.741 -> 43.00
11:57:27.741 -> 42.40
11:57:27.741 -> 1022.00
```

At Mobile Handset:



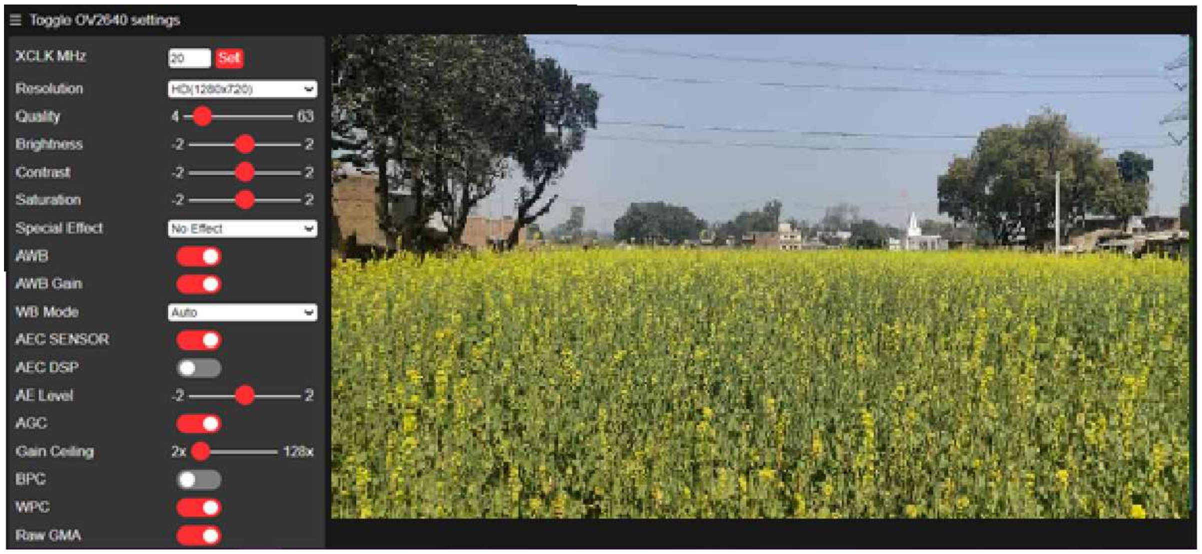
```
Temparature is Normal, Sprinklers Turned OFF
--------------------------------------------
Humidity:73.00
Temparature:33.60 *C
Soil Moisture Index:1016.00
Water is NEEDED, Motor Turned ON
Temparature is Normal, Sprinklers Turned OFF
--------------------------------------------
Humidity:73.00
Temparature:33.70 *C
Soil Moisture Index:615.00
Water Level is GOOD, Motor turned OFF
Temparature is Normal, Sprinklers Turned OFF
--------------------------------------------
Humidity:73.00
Temparature:33.70 *C
Soil Moisture Index:638.00
Water Level is GOOD, Motor turned OFF
Temparature is Normal, Sprinklers Turned OFF
--------------------------------------------
Humidity:73.00
Temparature:33.70 *C
Soil Moisture Index:607.00
Water Level is GOOD, Motor turned OFF
Temparature is Normal, Sprinklers Turned OFF
--------------------------------------------
Humidity:73.00
Temparature:33.70 *C
Soil Moisture Index:633.00
Water Level is GOOD, Motor turned OFF
Temparature is Normal, Sprinklers Turned OFF
--------------------------------------------
Humidity:73.00
Temparature:33.70 *C
Soil Moisture Index:603.00
Water Level is GOOD, Motor turned OFF
Temparature is Normal, Sprinklers Turned OFF
--------------------------------------------
```

ESP32 Cam:



```
10:07:37.684 -> Wifi connecting...
10:07:37.779 -> Wifi connecting...
10:07:37.872 -> Wifi connecting...
10:07:38.013 -> Wifi connecting...
10:07:38.114 -> Wifi connecting...
10:07:38.208 -> Wifi connecting...
10:07:38.313 -> Wifi connecting...
10:07:38.407 -> Wifi connecting...
10:07:38.482 -> Wifi connecting...
10:07:38.611 -> Wifi connecting...
10:07:38.691 -> Wifi connecting...
10:07:38.691 ->
10:07:38.691 -> WiFi connected
10:07:38.691 -> Camera Ready! Use 'http://192.168.0.198' to connect
```

Demo Image taken at outdoor:



```
XCLK MHz        20
Resolution      HD(1280x720)
Quality         4 — 63
Brightness      -2 — 2
Contrast        -2 — 2
Saturation      -2 — 2
Special Effect  No Effect
AWB
AWB Gain
WB Mode         Auto
AEC SENSOR
AEC DSP
AE Level        -2 — 2
AGC
Gain Ceiling    2x — 128x
BPC
WPC
Raw GMA
```

2.7 ADVANTAGES

- Reduced labor
- Timely irrigation — plants being watered when needed
- Management of higher flow rates
- Accurate cut-off of water compared to manual checking
- Reduced runoff of water.
- There is camera installed for surveillance of the field.

2.8 DISADVANTAGES

- Constant power is required for smooth operation of every component.
- Different programming is required for different types of soils and different corps.
- Extreme heat can damage some components.
- If any component get damaged, only an engineer can fix that.

2.9 CONCLUSION

The primary applications for this project are for farmers and gardeners who do not have enough time to water their crops/plants. It also covers those farmers who waste lot of water during irrigation. As water supplies become scarce and polluted, there is a need to irrigate more efficiently in order to minimize water use and chemical leaching. Recent advances in soil water sensing make the commercial use of this technology possible to automate irrigation management for vegetable production. However, research indicates that different sensors types perform under all conditions with no negative impact on crop yields, with reductions in water use ranges as high as 70% compared to traditional practices.

It can hereby be seen that the combination of hardware and software provides an economic irrigation controlling system which is extremely user friendly because it requires very less human interference for its operations once it is manufactured and implemented. It not only saves the most precious gift of nature, i.e., water, but also helps the farmers to grow their crops under controlled conditions and under continuous observation by temperature monitoring. Hence, this project “AUTOMATED IRRIGATION AND SURVILLENCE SYSTEM WITH ARDUINO” also helps in increasing quantity of productions and in doing quality productions of crops.

CHAPTER – 3

3.1 FUTURE SCOPE

- This project work can be modified in future in terms of storage, Speed, Accuracy, more data collection, and multiple operations.
- In this system an LCD display can be implemented.
- A fertilizer or pesticides tank can be installed in the system, and it can be used for multiple operations.
- Fire and smoke Detection sensors can also be installed for safety.
- Face detection and Buzzer system can also be installed on the system.
- A four-wheeler automatic vehicle can also be made which will collect data from various points of the field.
- We can use an GSM Module to upload the data in cloud and make a Database to use AI-ML to predict the future weather.
- Also, we can develop this system by using renewable energy which is solar power instead of batteries, using solar energy will help to reduce future cost.

3.2 REFERENCES

1. Muhammad U, “Automatic Irrigation control problem using Artificial Neural Network Controller” in International Journal of Electrical & Computer Sciences IJECS-IJENS Vol:10 No:02, Jan 2010.
2. Sanjukumar, R.V. Krishnaiah, "Advance Technique for Soil Moisture Content Based Automatic Motor Pumping for Agriculture Land Purpose".
3. S Nalini Durga, "Smart Irrigation System Based on Soil Moisture Using Iot" in International Research Journal of Engineering and Technology (IRJET), Jun 2018.
4. Goldstein Anat, “Applying machine learning on sensor data for irrigation recommendations” in Precision Agric, 2018.
5. HS Kalsi, "Electronic instrumentation", Tata McGraw Hill Ltd., New Delhi, 1999-96.
6. Introduction to LCD programming tutorial by Craig Steiner Copyright 1997 - 2005 by vault information services.
7. Abhinav Rajpal, Sumit Jain, Nistha Khare and Anil Kumar Shukla "Microcontroller-based Automatic Irrigation System with Moisture Sensor" Proceedings of the International Conference on Science and Engineering (ICSE 2011).
8. Karthikeshwar M, Mithraderi P Automated Irrigation. System in Agriculture using wireless Sensor Technology International journal of Advanced research in Electrical, Electronics and Instrumentation Engineering Vol.3, issue 12, December 2014 p. 13622 13627.
9. Samy Sadeky, Ayoub Al-Hamadiy, Bernd Michaelisy, Usama Sayedz," An Acoustic Method for Soil Moisture Measurement ", IEEE 2004.
10. Ms. Sweta S. Patil, Prof. Mrs. A.V. Malvijay, "Review for ARM based agriculture field monitoring system", International Journal of Scientific and Research Publications, Volume 4, Issue 2, February 2014.
11. Abhinav Rajpal, Sumit Jain, Nistha Khare and Anil Kumar Shukla, "Microcontroller based Automatic Irrigation”.
12. System with Moisture Sensors", Proceedings of the International Conference on Science and Engineering, 2011.

3.3 APPENDIX

Code for Transmitter Side:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
#define sensorPin A3

DHT dht(DHTPIN, DHTTYPE);
RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001";

void setup() {
Serial.begin(19200);

dht.begin();

radio.begin();
radio.openWritingPipe(address);
radio.setPALevel(RF24_PA_MIN);
radio.stopListening();
}



void loop() {

float h = dht.readHumidity();
float t = dht.readTemperature();
float f = dht.readTemperature(true);
float val = analogRead(sensorPin);

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

float Array[3]{
Array[0] = h,
Array[1] = t,
Array[2] = val,
};
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\n");
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" *C ");
Serial.print(" \n soil moisture:");
Serial.print(val);
Serial.print(" \n");
radio.write(&Array, sizeof(Array));
Serial.println("Sending...");
}
```

Code for Receiver Side:

```
#include <nRF24L01.h>
#include <printf.h>
#include <RF24.h>
#include <RF24_config.h>
#include <SoftwareSerial.h>

float Array[3];
float water;
float tem;
float hum;

RF24 radio(7, 8); // CE, CSN
const byte address[6] = "00001"; // the address the the module
SoftwareSerial bt(9,10); // RX | TX

void setup() {
bt.begin(9600); /* Define baud rate for software serial communication */
Serial.begin(9600); /* Define baud rate for serial communication */
radio.begin();
radio.openReadingPipe(0, address);
radio.setPALevel(RF24_PA_MIN);
radio.startListening();
pinMode(5, OUTPUT);
pinMode(11, OUTPUT);
}

void loop() {
//Serial.println("HI");
if (radio.available()) {
radio.read(&Array, sizeof(Array));
Serial.println(Array[0]);
Serial.println(Array[1]);
Serial.println(Array[2]);
delay(1000);




hum = Array[0];
tem = Array[1];
water = Array[2];

bt.print("Humidity:");
bt.println(Array[0]);
bt.print("Temperature:");
bt.print(Array[1]);
bt.println(" *C");
bt.print("Soil Moisture Index:");
bt.println(Array[2]);
delay(100);

if(water < 700.00) {
digitalWrite(5,HIGH);
bt.println("Water Level is GOOD, Motor turned OFF");

}
else{
digitalWrite(5,LOW);
bt.println("Water is NEEDED, Motor Turned ON");
}

if((tem > 40) || (hum < 40)){
digitalWrite(11,LOW);
bt.println("Too High Temperature, Sprinklers Turned ON");
}
else{
digitalWrite(11,HIGH);
bt.println("Temperature is Normal, Sprinklers Turned OFF");
}

bt.println("--------------------------------------------------------------------------------------------");

delay(1000);
}
}
```