SMART AGRICULTURE SYSTEM

Computer Science and Engineering

Project Report
of
CSE2006 – MICROPROCESSOR & INTERFACING

Fall Semester 2021-22

Submitted to
Faculty: Prof Manish Kumar

List of Contributors

S.NO.	Name	Registration Number
1.	Ayush Sharma	19BCE0408
2.	Anant Tiwari	19BCE2178
3.	Harshit Vijay	19BCE2321
4.	Tuhin Chakrabarty	19BCE2691

Table of Contents

Topic **Page No**

Abstract

The growing population fields an added responsibility of feeding 8 billion people. The yield of agriculture has to increase and technology has to be included. The dominant solution provided earlier includes similar techniques but a complex user interface which makes it difficult for the not so tech savvy farmers to use the system. The data from the farm - soil moisture level, temperature and humidity levels are used to provide suggestions so that the major stakeholders of the system - the farmers can get better yields and get better profits. A full stack system consisting of frontend as an Android application, with the backend as Firebase and the hardware implementation in an IoT system forms the Smart Agriculture system. The system gives precise text suggestions upon reading the values and the irrigation on the field can be managed from the app itself providing a new level of remote agriculture management to farmers or even home farm enthusiasts.

Index Terms—Agriculture, Android, Firebase, Data Analysis, Remote irrigation

Going into the technical aspects

- Smart Agriculture developing model is a real time monitoring system .It monitors the soil properties like temperature, humidity and soil moisture.

- Sensors read data and using wifi module NodeMCU ESP8266 the data from the sensors are stored on the database and by that data using an android app it suggests to the farmer about the irrigation requirements in the field.

- NodeMCU ESP8266 will give the motor command to release water according to the data read from the sensors

LITERATURE SURVEY

S.No	Paper Title	Name of the Conference/Journal, Year	Technology Used
1	Internet-of-Things (IoT)-Based Smart Agriculture: Toward Making the Fields Talk	IEEE Access, 1st August 2019	Wireless Sensors, IoT based sensors, Harvesting Robots, Communication in Agriculture, Smartphones, Cloud Computing.
2	A Review: Smart Farming Using IOT in the Area of Crop Monitoring	Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 5, 2021, Pages. 3887 - 3896 Received 15 April 2021; Accepted 05 May 2021.	Node MCU, Arduino IDE Banana Pi, Beaglebone, Raspberry Pi,Arduino Yun
3	Smart Farming System using IoT for efficient crop growth	IEEE Xplore, 7th May 2020	NodeMCU and other sensors, Smartphones.

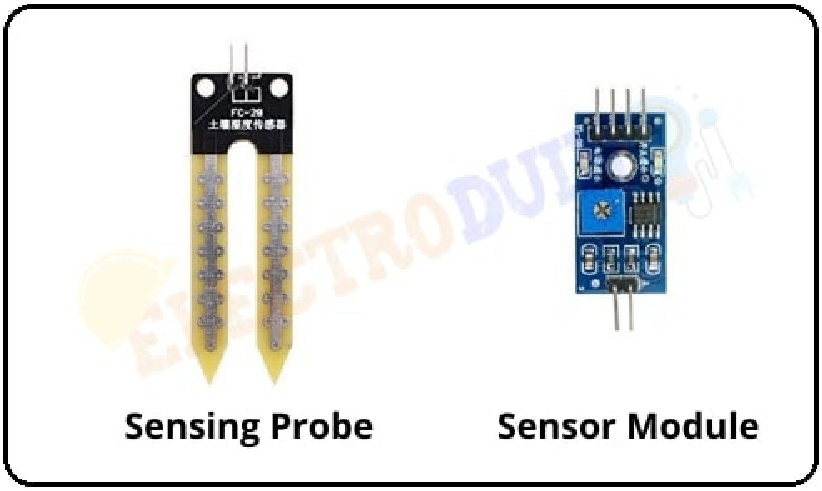
S.No	Paper Title	Name of the Conference/Journal, Year	Technology Used
4	Advances in IoT and Smart Sensors for Remote Sensing and Agriculture Applications	Citation: Ullo, S.L.; Sinha, G.R. Advances in IoT and Smart Sensors for Remote Sensing and Agriculture Applications. Remote Sens. 2021, 13, 2585.	IoT for remote sensing applications Smart sensors for remote sensing applications, Smart sensors and IoT for agriculture applications, Smart remote sensing systems

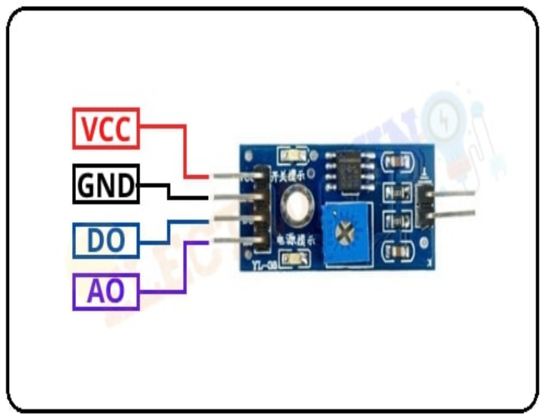
5	IoT Based Smart Agriculture Management System G. S. Nagaraja; Avinash B Soppimath; T. Soumya; A Abhinith	Published in: 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS) Date of Conference: 20-21 Dec. 2019 Publisher: IEEE	Iot, Big Data computing, Cloud and Mobile Computing.
6	Internet of Things for the Future of Smart Agriculture: A Comprehensive Survey of Emerging Technologies	IEEE/CAA JOURNAL OF AUTOMATICA SINICA, VOL. 8, NO. 4, APRIL 2021	Wireless technologies, open-source IoT platforms, software defined networking (SDN), network function virtualization (NFV) technologies, cloud/fog computing, and middleware platforms.

Hardware Components

Soil Moisture Sensor

- A Soil Moisture Sensor is one kind of low-cost electronic sensor that is used to detect the moisture of the soil.
- This sensor can measure the volumetric content of water inside the soil. This sensor consists of mainly two parts, one is **Sensing Probe** and another one is the **Sensor Module**.
- The probes allow the current to pass through the soil and then it gets the resistance value according to moisture value in soil.
- The Sensor Module reads data from the sensor probes and processes the data and converts it into a digital/analog output.
- So, the Soil Moisture Sensor can provide both types of output **Digital output (DO)** and **Analog output(AO)**.





VCC - +5 v power supply

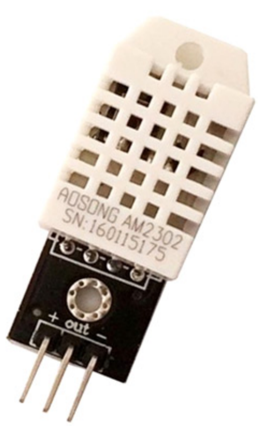
GND - Ground (-) power supply

DO - Digital Output (0 or 1)

AO - Analog Output (range 0 to 1023)

DHT22 Sensor

- DHT22 is a highly accurate humidity and temperature sensor.
- This sensor measures relative humidity values.
- It uses the capacitive sensor element to measure Humidity.
- For measuring temperature it uses an NTC thermistor.
- This sensor can be used in harsh conditions also.



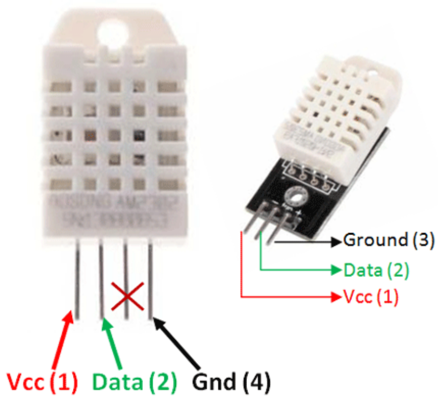
<u>DHT Sensor</u>

Vcc - Power supply 3.5V to 5.5V

Data - Outputs both Temperature and Humidity through serial Data

NC - No Connection and hence not used

Ground - Connected to the ground of the circuit



DHT Module

Vcc - Power supply 3.5V to 5.5V

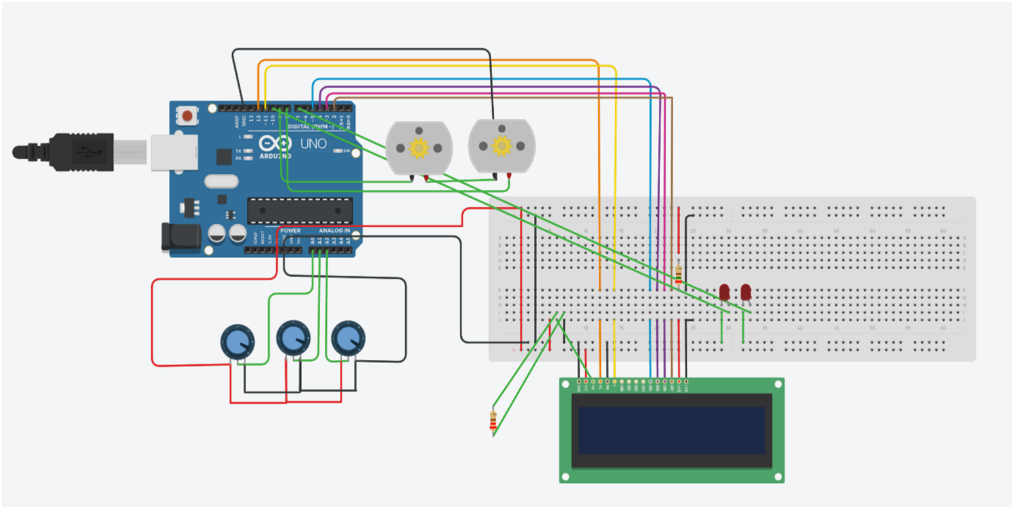
Data - Outputs both Temperature and Humidity through serial Data

Ground - Connected to the ground of the circuit

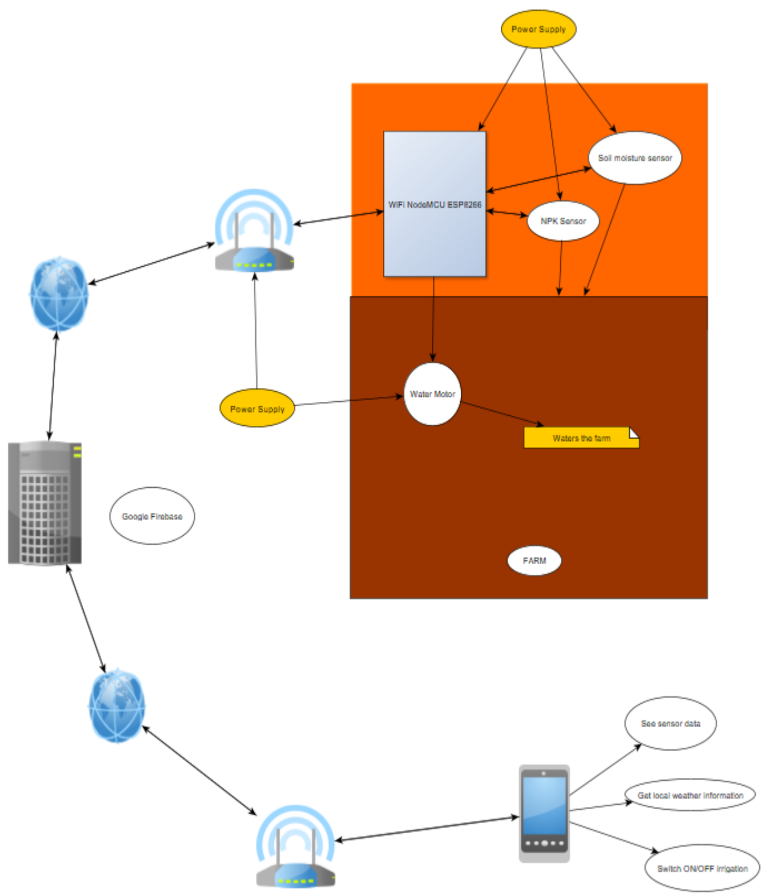
Drawback of the existing work and the Proposed work

- **Existing work**
  - It is not user-friendly provided that the majority of the audience are not technologically educated.
  - Expensive method
  - Unconventional
- **Proposed Work**
  - Non-Portability
  - Not waterproof
  - Lack of extra external factors

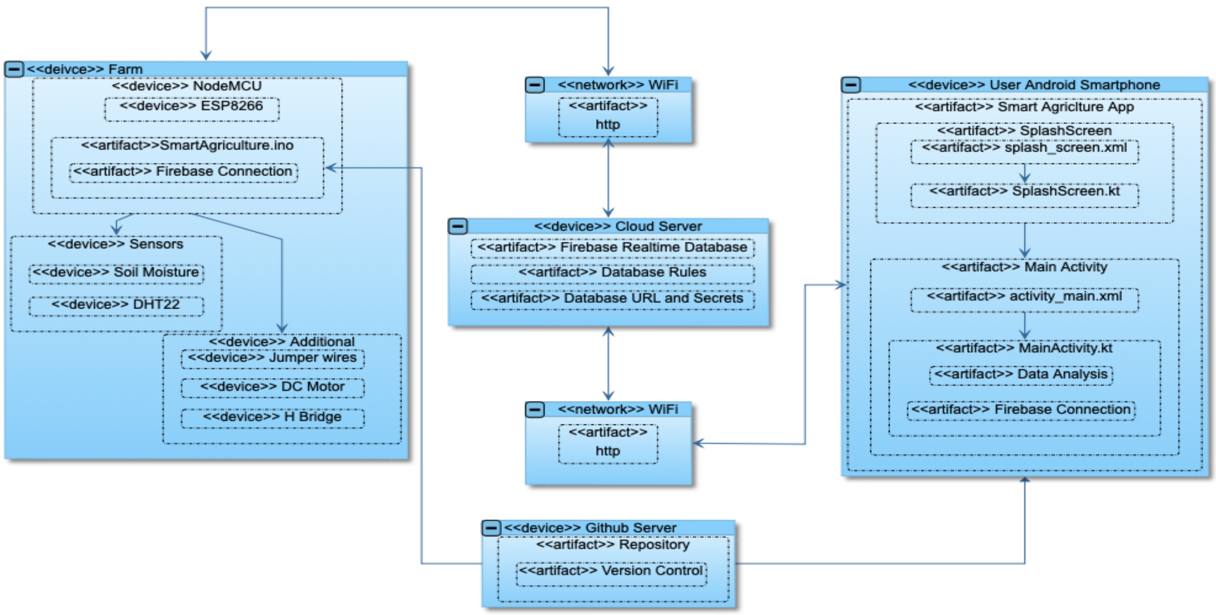
Circuit Diagram



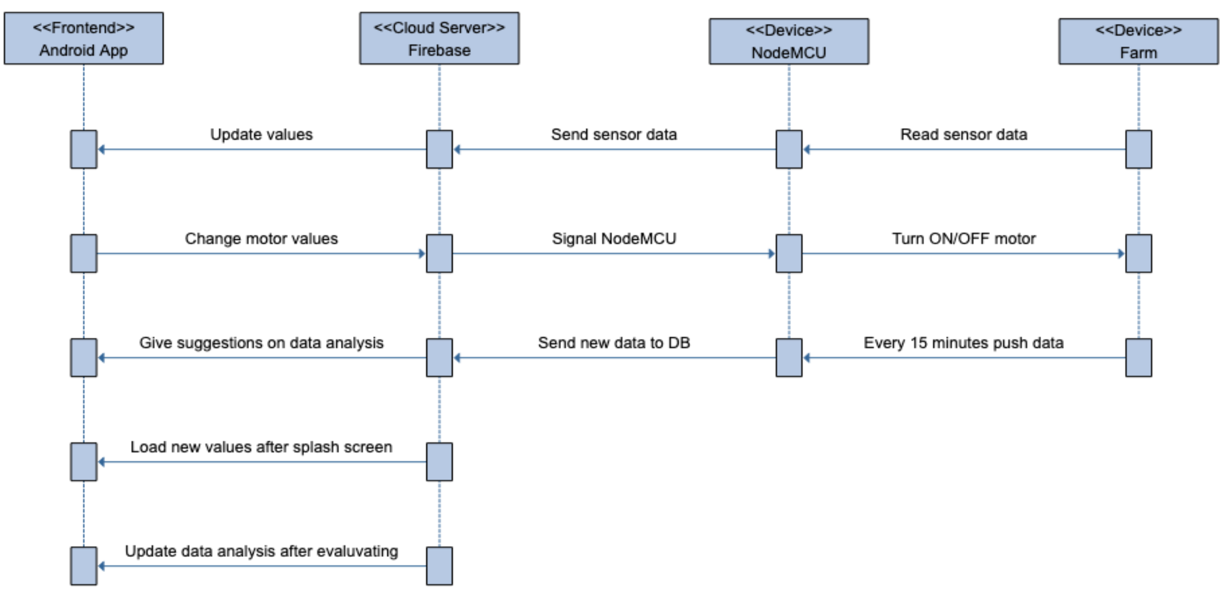
Workflow Diagram



Deployment Diagram



System Visualised



Implementation

Architecture Layers

The system is implemented as a Three Layer(Tier) or in a Layered Architecture Style. An Agile approach is followed for smooth development. The Smart Agriculture System is an end to end IoT system. The layered architecture simplifies the development into three building blocks :

- Perception Layer : Sensors, actuators and devices that interact directly with the environment.
- Network Layer : Connects devices and layers and performs coordination and communication.
- Application Layer : Data storage and processing layer and also consists of UI/UX for end users

Working of components

Sensors: - We have used two sensors i.e. soil moisture sensor which will detect moisture in soil and DHT22 which will detect temperature and humidity. These two sensors will read values from soil after every 15 minutes and update the same in the firebase.

Firebase: - Once all the sensors read values it will update in this firebase. So after every 15 minutes the values will be updated in firebase.

App: - When these sensors will read values from soil and once it gets updated on firebase, these values will also get updated on app once you refresh the app. You will also receive notification in the app if soil moisture goes below a particular value and temperature value increases above a particular value.

Firebase URL and Key

Firebase URL: - Firebase Cloud Storage helps users to share user-generated content, such as images and video, and allows them to build rich media content into their applications. Users use their Firebase Data URL to access this stored information.

Firebase key: - API keys are used to identify your Firebase project when interacting with Firebase/Google services. Specifically, they're used to associate API requests with your project for quota and billing.

Realtime Database Rules

Rule Types

.read- Describes if and when data is allowed to be read by users.

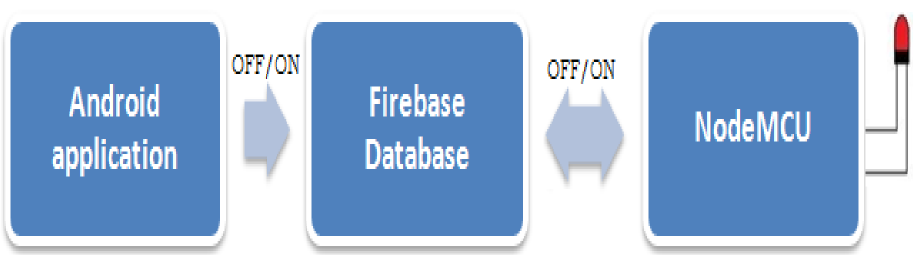
.write- Describes if and when data is allowed to be written.

.validate- Defines what a correctly formatted value will look like, whether it has child attributes, and the data type.

.indexOn- Specifies a child to index to support ordering and querying.

Firebase connection with Nodemcu and Android

The Android app sends the serial data 1 or 0 to the Firebase database. The Firebase database interacts with Wi-Fi NodeMCU and this NodeMCU acts on the basis of data received from Firebase Database. If NodeMCU receives serial data 1, it turns ON the LED, and if NodeMCU receives serial input 0 then it turns OFF the LED.



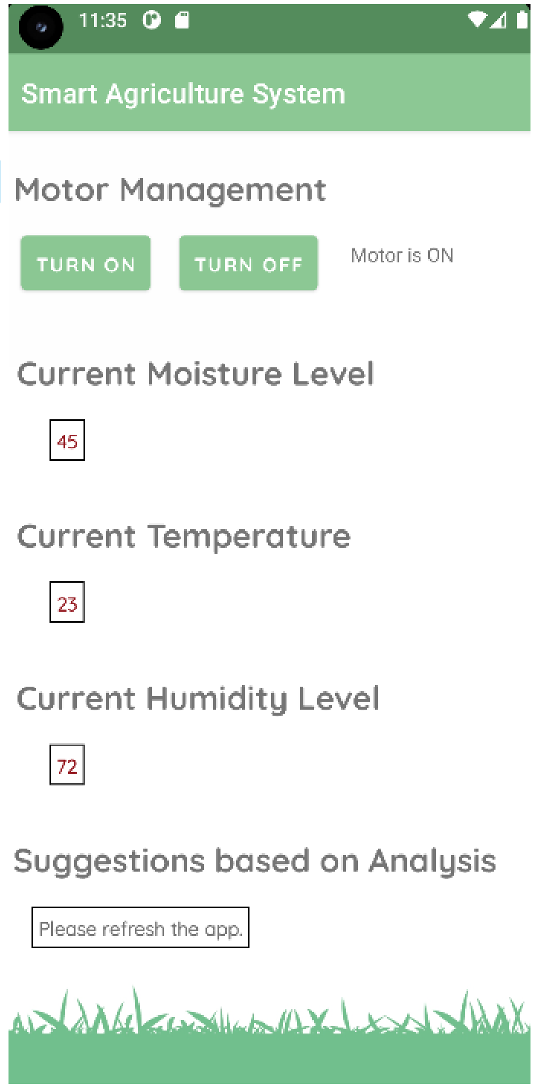
HTTP protocol

Hypertext Transfer Protocol (**HTTP**) is a type of stateless protocol that transfers information between the clients and the web server. For this purpose it uses a set of rules and standards. Under normal circumstances it uses TCP/IP or UDP protocol.

Why we used: -

- **Addressing-** It assigns IP addresses with recognizable names so that it can be identified easily in the World Wide Web.
- **Flexibility**
- **Security**
- **Latency**
- **Accessibility**

Screenshots

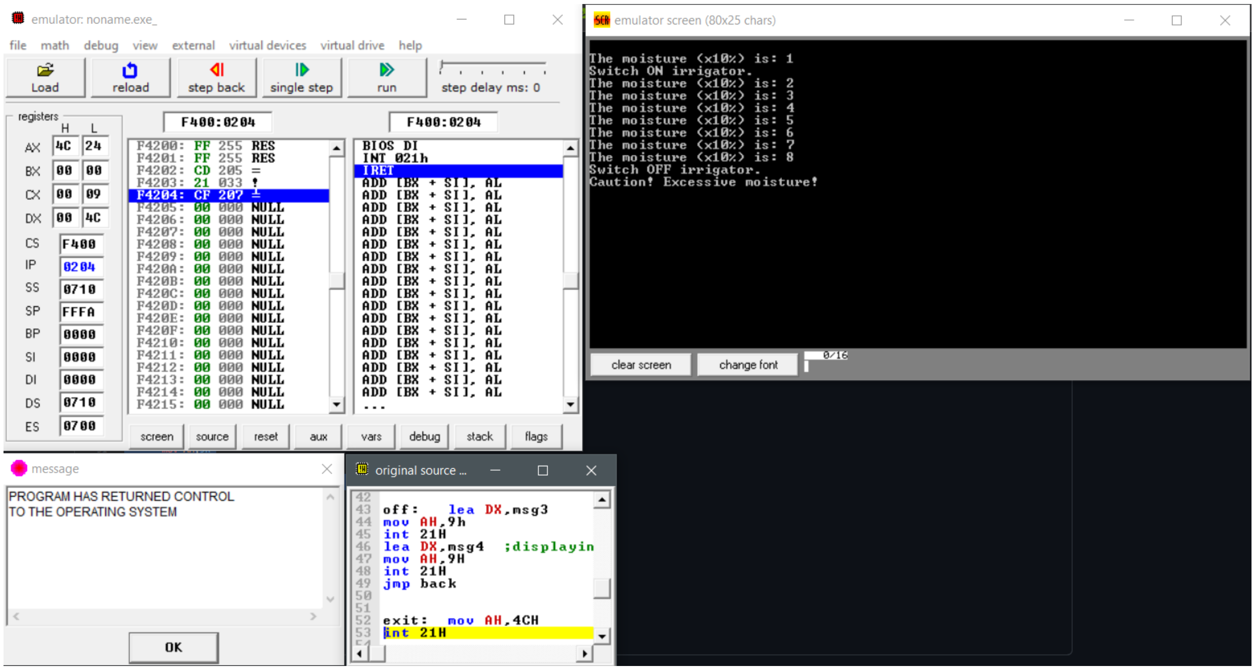


Splash Screen

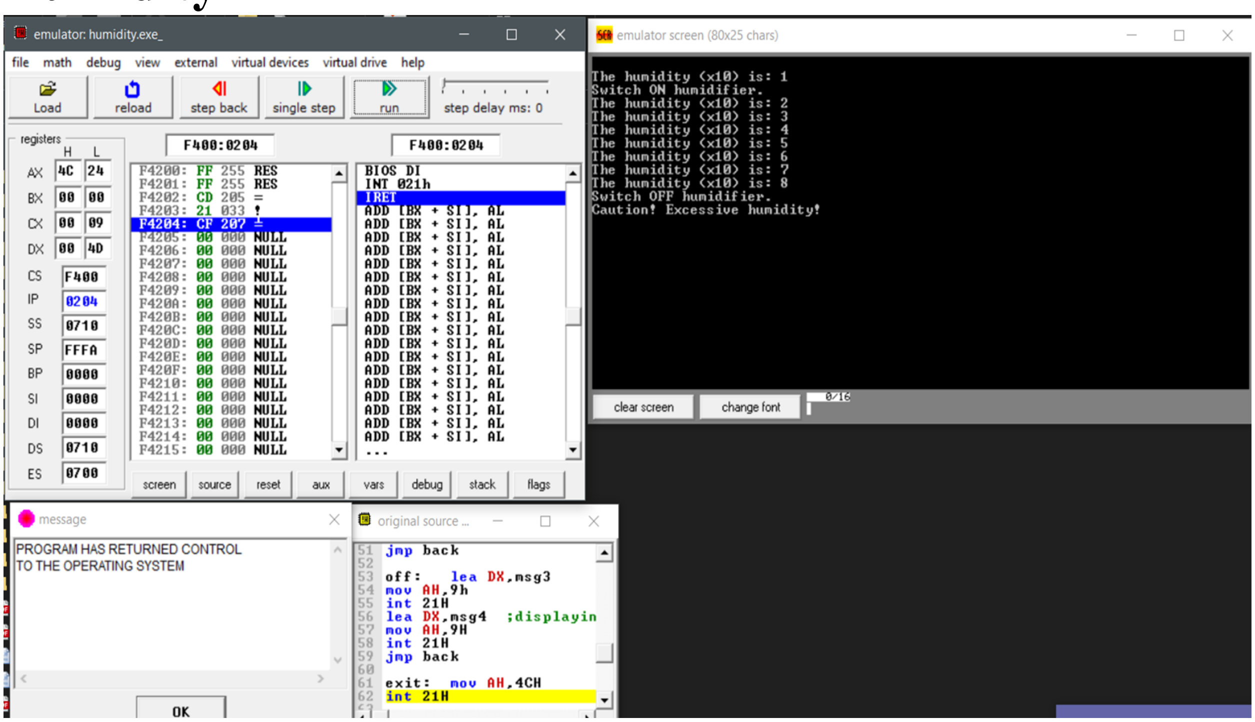
Landing Screen

EMU8086

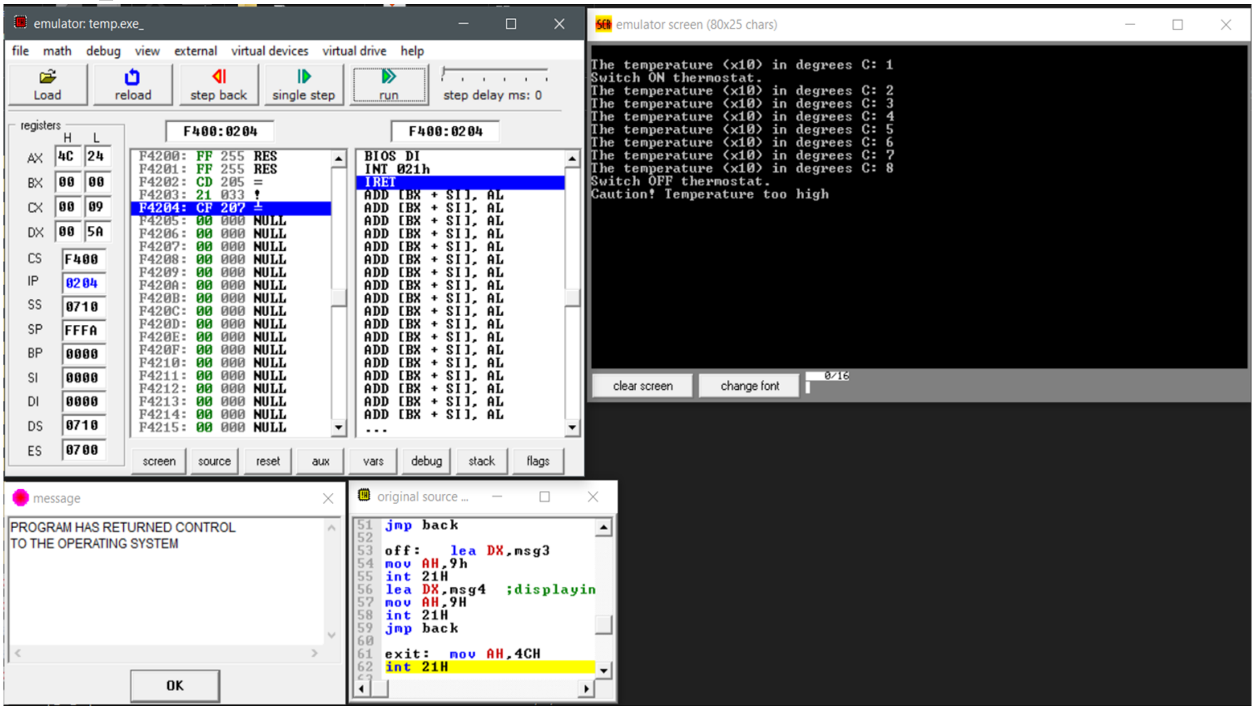
Moisture



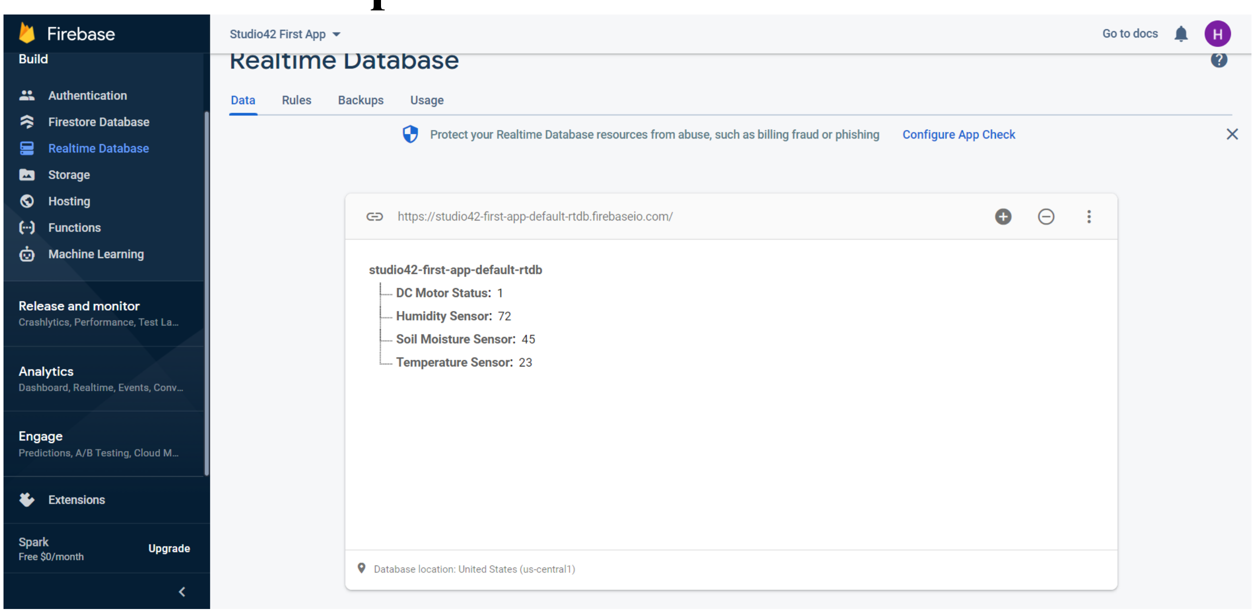
Humidity



Temperature



Results and Graphs



Conclusion

All observations and experimental tests prove that the project is a complete solution to field activities, irrigation problems using smart irrigation systems . Implementation of such a system in the field can definitely help to improve the yield of the crops and overall production.

For future developments it can be enhanced by developing this system for large acres of land. Also the system can be integrated to check the quality of the soil and the growth of crops in each soil. The sensors and microcontroller are successfully interfaced and wireless communication is achieved between various nodes. All observations and experimental tests prove that this project is a complete solution to field activities and irrigation problems. Implementation of such a system in the field can definitely help to improve the yield of the crops and overall production.

Also it would be focused more on increasing sensors on this stick to fetch more data especially with regard to Pest Control and by also integrating GPS modules in this IoT Stick to enhance this Agriculture IoT Technology to full-fledged Agriculture Precision ready product.

Reference

Prof K.A. Patil and Prof N.R. Kale “ A Model of Smart Agriculture Using IoT ” in Institute of Electrical and Electronic Engineers, 2016.

Nageswara Rao and B. Sridhar “ IoT based smart crop-field monitoring and automation irrigation system ” in Institute of Electrical and Electronic Engineers, 2018.

G. Sushanth and S.Sujatha” IoT Based Smart Agriculture System ” in Institute of Electrical and Electronic Engineers, 2018.

Prachi Patil, Akshay Narkhede, Ajita Chalke, Harshali Kalaskar and Manita Rajput “ Real Time Automation of Agricultural Environment” in Institute of Electrical and Electronic Engineers, 2014.

R. Madhumathi, T. Arumuganathan and R. Shruthi, ”Soil NPK and Moisture analysis using Wireless Sensor Networks,” 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2020, pp. 1-6, doi: 10.1109/ICCCNT49239.2020.9225547.

Dr. Narayan G. Hedge, “Water Scarcity and Security in India”, BAIF Development Research Foundation, Pune.

Marvin T. Bate, “Changing computer use in agriculture: evidence from Ohio”, Computers and Electronics in Agriculture, Elsevier science publishers, vol. 47, 1–13, 2005. Nguyen, T.; Roehrig, F.; Grosjean, G.; Tran, D.; Vu, T. Climate Smart Agriculture in Vietnam; International Jayaraman, P.; Yavari, A.; Georgakopoulos, M.; Arkady, Z. Internet of Things Platform for Smart Farming: Experiences and Lessons Learnt. Sensors 2016, 16, 1804–1282.

Appendix

EMU8086

Moisture

```
DATA SEGMENT

    msg1 db 10,13,"The moisture (x10%) is: $"

    msg2 db 10,13,"Switch ON irrigator. $"

    msg3 db 10,13,"Switch OFF irrigator. $"

    msg4 db 10,13,"Caution! Excessive moisture! $"

DATA ENDS
```

```
CODE SEGMENT
   ASSUME DS:DATA,CS:CODE
START: mov AX,@data ;intialize data segment
       mov DS,AX
       mov CL,1H
L1:    lea DX,msg1  ;displaying moisture level message
       mov AH,9H
       int 21H
       add CL,30H   ;ASCII adjust before displaying
       mov DL,CL
       mov AH,2H    ;display
       int 21H
       sub CL,30H   ;ASCII adjust after displaying
       cmp CL,8H    ;switch off irrigator
       je off       ;jump to off if = 8
       cmp CL,1H    ;switch on irrigator
       je on        ;jump to on if = 1
back:  inc CL       ;increase moisture level by 1
       cmp CL,8H    ;check if moisture level is excessive
       jle l1
```

```
        jmp exit
on:     lea DX,msg2
        mov AH,9h
        int 21H
        jmp back
off:    lea DX,msg3
        mov AH,9h
        int 21H
        lea DX,msg4  ;displaying caution
        mov AH,9H
        int 21H
        jmp back
exit:   mov AH,4CH
        int 21H
CODE ENDS
END START
```

Humidity

```
DATA SEGMENT
   msg1 db 10,13,"The humidity (x10) is: $"
   msg2 db 10,13,"Switch ON humidifier. $"
   msg3 db 10,13,"Switch OFF humidifier. $"
   msg4 db 10,13,"Caution! Excessive humidity! $"
DATA ENDS
CODE SEGMENT
   ASSUME DS:DATA,CS:CODE
START: mov AX,@data ;intialize data segment
      mov DS,AX
      mov CL,1H
L1:   lea DX,msg1  ;displaying moisture level message
      mov AH,9H
      int 21H
      add CL,30H   ;ASCII adjust before displaying
      mov DL,CL
      mov AH,2H    ;display
      int 21H
      sub CL,30H   ;ASCII adjust after displaying
```

```
        cmp CL,8H    ;switch off irrigator
        je off       ;jump to off if = 8
        cmp CL,1H    ;switch on irrigator
        je on        ;jump to on if = 1
back:  inc CL       ;increase moisture level by 1
        cmp CL,8H    ;check if moisture level is excessive
        jle l1
        jmp exit
on:    lea DX,msg2
        mov AH,9h
        int 21H
        jmp back
off:   lea DX,msg3
        mov AH,9h
        int 21H
        lea DX,msg4  ;displaying caution
        mov AH,9H
        int 21H
        jmp back
exit:  mov AH,4CH
        int 21H
```

```
CODE ENDS
END START
```

Temperature

```
DATA SEGMENT
   msg1 db 10,13,"The temperature (x10) in degrees C: $"
   msg2 db 10,13,"Switch ON thermostat. $"
   msg3 db 10,13,"Switch OFF thermostat. $"
   msg4 db 10,13,"Caution! Temperature too high $"
DATA ENDS
CODE SEGMENT
   ASSUME DS:DATA,CS:CODE
START: mov AX,@data ;intialize data segment
       mov DS,AX
       mov CL,1H
L1:    lea DX,msg1   ;displaying moisture level message
       mov AH,9H
       int 21H
       add CL,30H   ;ASCII adjust before displaying
       mov DL,CL
```

```
        mov AH,2H    ;display
        int 21H
        sub CL,30H   ;ASCII adjust after displaying
        cmp CL,8H    ;switch off irrigator
        je off       ;jump to off if = 8
        cmp CL,1H    ;switch on irrigator
        je on        ;jump to on if = 1
back:   inc CL       ;increase moisture level by 1
        cmp CL,8H    ;check if moisture level is excessive
        jle l1
        jmp exit
on:     lea DX,msg2
        mov AH,9h
        int 21H
        jmp back
off:    lea DX,msg3
        mov AH,9h
        int 21H
        lea DX,msg4  ;displaying caution
        mov AH,9H
        int 21H
```

```
      jmp back
exit:  mov AH,4CH
      int 21H
CODE ENDS
END START
```

Android Studio

```
package com.sidchiku9.studio42firstapp


import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.database.*
import java.lang.NumberFormatException



class MainActivity : AppCompatActivity() {
```

```
    private var mDatabase: DatabaseReference? = null
    private var moistureReference: DatabaseReference? = null
    private var temperatureReference: DatabaseReference? = null
    private var humidityReference: DatabaseReference? = null
    private var dcStatus: DatabaseReference? = null
    private var dataAnalysisOne : DatabaseReference? = null
    private var dataAnalysisTwo : DatabaseReference? = null
    private var moistureLevel : String = ""
    private var temperature : String = ""
    private var moistureDA : Int = 0
    private var temperatureDA : Int = 0
    //this is a test comment


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)


        val turnOnbutton = findViewById<Button>(R.id.onButton)
        val turnOffbutton = findViewById<Button>(R.id.offButton)
        val moistureTextView = findViewById<TextView>(R.id.moistureUpdate)
        val temperatureTextView =
findViewById<TextView>(R.id.temperatureUpdate)
```

```
val humidityTextView = findViewById<TextView>(R.id.humidityUpdate)
val dcMotorStatus = findViewById<TextView>(R.id.dcMotorStatus)
val suggestionsUpdate = findViewById<TextView>(R.id.suggestionUpdate)

//DC MOTOR

mDatabase = FirebaseDatabase.getInstance().reference

turnOnbutton.setOnClickListener {
    mDatabase!!.child("DC Motor Status").setValue(1)
}
turnOffbutton.setOnClickListener {
    mDatabase!!.child("DC Motor Status").setValue(0)
}

//DC MOTOR READ VALUE

dcStatus = FirebaseDatabase.getInstance().getReference("DC Motor Status")

dcStatus?.addValueEventListener(object : ValueEventListener{
    override fun onDataChange(snapshot: DataSnapshot) {
```

```
            val dcStatus = snapshot.value.toString()


            if(dcStatus == "0"){
                dcMotorStatus.text = "Motor is OFF"
            }
            else{
                dcMotorStatus.text = "Motor is ON"
            }
        }

        override fun onCancelled(error: DatabaseError) {
            Toast.makeText(this@MainActivity, "Unable to fetch Firebase data",
Toast.LENGTH_SHORT).show()
        }
    })


    //SOIL MOISTURE SENSOR


    moistureReference = FirebaseDatabase.getInstance().getReference("Soil
Moisture Sensor")


    moistureReference?.addValueEventListener(object : ValueEventListener{
```

```
            override fun onDataChange(snapshot: DataSnapshot) {
                moistureLevel = snapshot.value.toString()
                moistureTextView.text = moistureLevel
            }

            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(this@MainActivity, "Unable to fetch Firebase data",
Toast.LENGTH_SHORT).show()
            }

        })

        //TEMPERATURE SENSOR

        temperatureReference =
FirebaseDatabase.getInstance().getReference("Temperature Sensor")

        temperatureReference?.addValueEventListener(object : ValueEventListener{
            override fun onDataChange(snapshot: DataSnapshot) {
                temperature = snapshot.value.toString()
                temperatureTextView.text = temperature
            }
```

```
            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(this@MainActivity, "Unable to fetch Firebase data",
Toast.LENGTH_SHORT).show()
            }

        })

        //HUMIDITY SENSOR

        humidityReference = FirebaseDatabase.getInstance().getReference("Humidity
Sensor")

        humidityReference?.addValueEventListener(object : ValueEventListener{
            override fun onDataChange(snapshot: DataSnapshot) {
                val text : String = snapshot.value.toString()
                humidityTextView.text = text
            }

            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(this@MainActivity, "Unable to fetch Firebase data",
Toast.LENGTH_SHORT).show()
```

```
    }

}) 

//SUGGESTIONS DATA ANALYSIS PART
dataAnalysisOne = FirebaseDatabase.getInstance().getReference("Soil
Moisture Sensor")
dataAnalysisTwo =
FirebaseDatabase.getInstance().getReference("Temperature Sensor")

dataAnalysisOne?.addValueEventListener(object : ValueEventListener{
    override fun onDataChange(snapshot: DataSnapshot) {
        moistureDA = (snapshot.value as Long).toInt()
    }

    override fun onCancelled(error: DatabaseError) {
        Toast.makeText(this@MainActivity, "Unable to fetch Firebase data",
Toast.LENGTH_SHORT).show()
    }
})

dataAnalysisTwo?.addValueEventListener(object : ValueEventListener{
```

```
            override fun onDataChange(snapshot: DataSnapshot) {
                temperatureDA = (snapshot.value as Long).toInt()


                if(temperatureDA <= 25 && moistureDA >= 80){
                    suggestionsUpdate.text = "Ideal Temp and Moisture. The field is in an
ideal condition. Maintain this to expect good yield."
                }
                else if(temperatureDA >= 30 && moistureDA <= 75){
                    suggestionsUpdate.text = "Low moisture. Please water the fields."
                }
                else{
                    suggestionsUpdate.text = "Please refresh the app."
                }
            }


            override fun onCancelled(error: DatabaseError) {
                Toast.makeText(this@MainActivity, "Unable to fetch Firebase data",
Toast.LENGTH_SHORT).show()
            }


        })
    }
```

}

Arduino

```
#include<ESP8266WiFi.h>

#include<DHT.h>

#include<FirebaseArduino.h>


#define ssid "hello"

#define password "chikusid9"

#define firebaseHost "studio42-first-app-default-rtdb.firebaseio.com"

#define firebaseAuth "tPPoGYE2nNRYyeNcOkLjAfmoQUKhhEP6WGV1NOuH"


#define DHTPIN 7

#define DHTTYPE DHT22


int dcMotorpositive;

int dcMotornegative;

int soilMoisturePin = A0;


int soilMoistureLevel;

int temperature;
```

```
int humidity;
int dcMotorstatus;

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  getConnection();
  Firebase.begin(firebaseHost, firebaseAuth);
  dht.begin();
}

void loop() {

  dcMotorstatus = readMotor();

  temperature = dht22Temperature();

  humidity = dht22Humidity();

  soilMoistureLevel = soilMoistureWriteFunction();
```

```
  //<!-- Send data to firebase code
  //send data to firebase
  Firebase.setFloat("Soil Moisture Sensor", soilMoistureLevel); // delay(3600000);
for actual real world testing
  if(Firebase.failed()){
    Serial.print("setting /number failed:");
    Serial.println(Firebase.error());
    return;
    }
  Firebase.setFloat("Temperature Sensor", temperature);
  if(Firebase.failed()){
    Serial.print("setting /number failed:");
    Serial.println(Firebase.error());
    return;
    }
  Firebase.setFloat("Humidity Sensor", humidity);
  if(Firebase.failed()){
    Serial.print("setting /number failed:");
    Serial.println(Firebase.error());
    return;
    }
```

```
    //send data to firebase code ends here --!>


  delay(1000); //for simulation purposes
}

//function to connect the board to WiFi
void getConnection(){
  WiFi.begin(ssid, password);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());
}

//functions for temperature and humidity (update)
int dht22Temperature(){
  int temp;
```

```
  temp = dht.readTemperature();
  return temp;
}

int dht22Humidity(){
  int hum;
  hum = dht.readHumidity();
  return hum;
}

//function for soil moisture sensor (update)
int soilMoistureWriteFunction(){
  int soil;
  soil = analogRead(soilMoisturePin);
  return soil;
}

//function to read DC Motor Status (read)
int readMotor(){
  int dcStatus;
  dcStatus = Firebase.getFloat("DC Motor Status");
```

```
 return dcStatus;
}
```

Plagiarism Report