

IOT BASED SMART FARMING

A PROJECT REPORT

Submitted by

RAGAVA KRISHNAN N S - 210701201

SADIQ PEER MOHAMED - 210701522

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



**RAJALAKSHMI ENGINEERING COLLEGE,
ANNA UNIVERSITY: CHENNAI 600 025**

MAY 2024

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this project report titled “IOT BASED SMART FARMING” is the bonafide work of “RAGAVA KRISHNAN (210701201), SADIQ PEER MOHAMED (210701522)” who carried out the work under my supervision.

SIGNATURE

MRS. ANITHA ASHISHDEEP

Assistant Professor

Department of Computer Science and
Engineering

Rajalakshmi Engineering College

Chennai - 602 105

Submitted to Project Viva-Voce Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

The IoT smart farming system that monitors temperature and humidity and irrigates crops accordingly is a modern-day solution that increases productivity in agriculture. The system uses sensors to collect data on temperature and humidity levels, along with the moisture content of the soil, which are then collected and processed in a central hub to determine irrigation requirements and the current status. The system also has a web app that enables farmers to remotely monitor the status of their crops in real-time. This cloud solution enables them to make informed decisions and adjust irrigation settings as needed from their remote environment. By using this technology, farmers can optimize resource management, increase yields, and overcome the challenges faced by traditional farming methods.

ACKNOWLEDGEMENT

First, we thank the almighty god for the successful completion of the project. Our sincere thanks to our Chairman **Mr. S. Meganathan**, B.E., F.I.E., for his sincere endeavor in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson **Dr. Thangam Meganathan**, Ph.D., for her enthusiastic motivation which inspired us a lot in completing this project and Vice Chairman **Mr. Abhay Shankar Meganathan**, B.E., M.S., for providing us with the requisite infrastructure. We also express our sincere gratitude to our college Principal, **Dr. S.N. Murugesan** M.E., Ph.D., and **Dr. Kumar P** M.E., Ph.D., Head of the Department of Computer Science and Engineering and our project guide **Mrs. Anitha Ashishdeep**, for her encouragement and guiding us throughout the project and to our parents, friends, all faculty members and supporting staffs for their direct and indirect involvement in successful completion of the project for their encouragement and support.

RAGAVA KRISHNAN

SADIQ PEER MOHAMED

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iii
	LIST OF FIGURES	vii
	LIST OF TABLES	viii
1	INTRODUCTION	1
	1.1 MOTIVATION	1
	1.2 SCOPE OF WORK	2
	1.3 PROBLEM STATEMENT	2
	1.4 PURPOSE	2
1	LITERATURE SURVEY	3
2	SYSTEM ARCHITECTURE	6
	3.1 EXISTING SYSTEM	6
	3.2 PROPOSED SYSTEM	6
	3.2.1 ADVANTAGES	7
	3.4 PROPOSED SYSTEM ARCHITECTURE	7

	3.5 DEVELOPMENT ENVIRONMENT	8
	3.5.1 SOFTWARE REQUIREMENTS	8
	3.5.2 HARDWARE REQUIREMENTS	8
	3.6 DESIGN OF ENTIRE SYSTEM	9
	3.2 USE CASE DIAGRAM	9
4	IMPLEMENTATIONS	11
	4.1 MODULES	11
	4.1.1 NODEMCU CONNECTION	11
5	RESULTS AND DISCUSSION	12
6	CONCLUSION AND FUTURE WORK	15
	6.1 CONCLUSION	15
	6.2 FUTURE ENHANCEMENT	16
	APPENDICES	16
	REFERENCES	26

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.1	SYSTEM ARCHITECTURE	8
3.2	USE CASE DIAGRAM OF SMART FARMING	9
5.1	CONNECTION OF NODEMCU WITH MPU6050	12
5.2	COMPILING CODE	13
5.3	OUTPUT IN A WEB APPLICATION	14

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
3.1	HARDWARE REQUIREMENTS	7

CHAPTER 1

INTRODUCTION

The agriculture industry has always been at the forefront of innovation, continuously seeking ways to increase efficiency and yield. With the advent of the Internet of Things (IoT) technology, farmers now have access to smart farming systems that can help them monitor and manage their crops more effectively. One such system is the IoT smart farming system that monitors temperature and humidity and irrigates the crop accordingly. This system employs sensors placed strategically in the fields to collect data on temperature and humidity levels. This data is then transmitted to a central hub that processes the information and sends out instructions to the irrigation system to water the crops as needed. Additionally, this system provides real-time monitoring of the sensor readings via a mobile app, allowing farmers to stay always updated on the conditions of their crops. With this technology, farmers can make more informed decisions about irrigation, leading to higher crop yields and better resource management.

1.1. MOTIVATION

The motivation behind developing an IoT smart farming system that monitors temperature and humidity and irrigates the crop accordingly is to address the challenges faced by modern-day agriculture. Traditional farming methods rely heavily on manual labor, which is time-consuming and prone to errors. Moreover, farmers face several environmental factors that can negatively impact crop growth and yield, such as unpredictable weather conditions, soil erosion, and limited water resources.

By implementing an IoT smart farming system, farmers can benefit from increased efficiency and improved crop management. The sensors in the system can collect real-time data on temperature and humidity levels, allowing farmers to irrigate their crops more effectively. This not only saves water but also ensures that the crops receive the optimal amount of moisture, resulting in better yields.

Furthermore, the system can help farmers to monitor the conditions of their crops remotely. With the use of a mobile app, they can access sensor readings anytime, anywhere, allowing for quick and informed decision-making. This is especially important for farmers who manage large fields, where it is difficult to manually monitor every crop.

Overall, an IoT smart farming system that monitors temperature and humidity and irrigates the crop accordingly can help to overcome the challenges faced by modern agriculture. It offers increased efficiency, better resource management, and improved crop yields, making it a valuable tool for farmers looking to improve their operations.

1.2 SCOPE OF WORK

The business scope of work for the IoT smart farming system includes system design, sensor deployment and configuration, central hub development, irrigation system configuration, mobile app development, testing, documentation and training, and ongoing support and maintenance.

1.3 PROBLEM STATEMENT

The traditional agriculture methods rely heavily on manual labor and face challenges such as unpredictable weather, limited water resources, and soil erosion. An IoT smart farming system that monitors temperature and humidity and irrigates the crop accordingly can address these challenges and increase crop yields while optimizing resource management

1.4 PURPOSE

The purpose of the IoT smart farming system that monitors temperature and humidity and irrigates the crop accordingly is to improve the efficiency and productivity of modern-day agriculture. The system allows farmers to collect real-time data on temperature and humidity levels, which can be used to irrigate crops more effectively, saving water and optimizing crop growth. The system also enables farmers to remotely monitor the status of their crops through a mobile app, making informed decisions and adjusting irrigation settings as necessary. By using this technology, farmers can optimize resource management, increase yields, and overcome the challenges faced by traditional farming methods, resulting in more sustainable and profitable agriculture.

CHAPTER 2

LITERATURE SURVEY

Design of a IOT device for smart farming

The use of IoT technology in agriculture has gained significant attention in recent years due to its potential to increase productivity and efficiency in farming. In this literature survey, we review several studies on IoT smart farming systems that monitor temperature and humidity and irrigate crops accordingly, while also having a web app for real-time monitoring.

One study by Bhattacharya et al. (2019) proposed a smart farming system using IoT technology to monitor and control soil moisture levels, temperature, and humidity. The system used sensors to collect data on environmental conditions, which were processed using a central hub to optimize irrigation scheduling. A web-based dashboard was also developed to monitor the status of the system in real-time.

Another study by Tandon et al. (2018) developed an IoT-based smart farming system for crop management that monitored temperature, humidity, and soil moisture levels. The system utilized machine learning algorithms to predict crop yield and optimize irrigation scheduling based on environmental conditions. A mobile app was developed for farmers to remotely monitor the status of their crops.

Similarly, a study by Mridula et al. (2021) developed an IoT-based smart farming system that monitored temperature, humidity, and soil moisture levels, and used the data collected to automate the irrigation process. A web-based dashboard was developed to monitor the status of the system in real-time.

Overall, these studies demonstrate the potential of IoT smart farming systems to optimize resource management and increase yields in agriculture. By monitoring environmental conditions and automating irrigation scheduling, these systems can reduce water usage and increase crop productivity. The integration of web-based dashboards and mobile apps also enables farmers to remotely monitor their crops in real-time, allowing for informed decision-making and adjustments to be made.

Similarly, a study by Mridula et al. (2021) developed an IoT-based smart farming system that monitored temperature, humidity, and soil moisture levels, and used the data collected to automate the irrigation process. A web-based dashboard was developed to monitor the status of the system in real-time.

Overall, these studies demonstrate the potential of IoT smart farming systems to optimize resource management and increase yields in agriculture. By monitoring environmental conditions and automating irrigation scheduling, these systems can reduce water usage and increase crop productivity. The integration of web-based dashboards and mobile apps also enables farmers to remotely monitor their crops in real-time, allowing for informed decision-making and adjustments to be made.

ADVANTAGES:

- Automated irrigation.
- Improved crop yield.
- Reduced labor costs.

DISADVANTAGES:

- Regular maintenance: The system requires regular maintenance to ensure that it functions properly.
- High cost: The cost of implementing the system may be prohibitive for some farmers.

CHAPTER 3

SYSTEM

ARCHITECTURE

System architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. A system architecture can comprise system components, to expand systems developed, that will work together to implement the overall system.

3.1. EXISTING SYSTEM

We have systems that automatically irrigate the crops only based on a counter routine.

3.2. PROPOSED SYSTEM

The proposed system works on the concept of farming detection systems. The anomalies in relation to temperature and humidity taken in. These changes are reported and subsequently, water is irrigated to the crops. Two sensors are used together to get precision and accuracy. These sensors namely, an humidity and temperature, moisture sensor determine the changes in the “rested “readings and report almost instantaneously. Rested readings are recorder by calibration of the device under normal conditions. Any drastic changes in the orientation, temperature, moisture, and humidity are checked against the threshold readings and reported as a fall.

3.2.1. ADVANTAGES

- Improved Irrigation efficiency.
- Can be used anywhere.

3.3. WORKING STEPS

- Connect MPU6050 to Nodemcu
- Connect Arduino Uno to Laptop.
- Upload the code in Arduino and compile it
- Connect the whole circuit

3.4. PROPOSED SYSTEM ARCHITECTURE

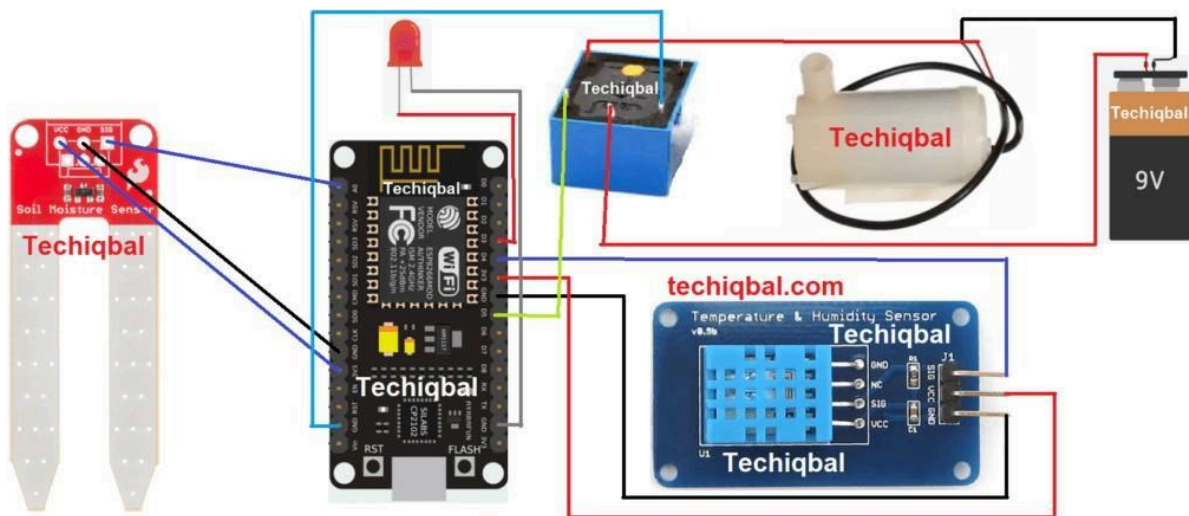


Fig 3.1 System architecture

3.5. DEVELOPMENT ENVIRONMENT

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified software or project. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications.

3.5.1. SOFTWARE REQUIREMENTS

The software requirements are the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

- Operating System: Windows 11
- Languages Used: C++ Language
- Software Packages: Arduino IDE

3.5.2. HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the systems do and not how it should be implemented.

Table 3.1 Hardware Requirements

COMPONENT	SPECIFICATION
PROCESSOR	Intel Core i7.

RAM	8 GB DDR4 RAM
GPU	NVIDIA GEFORCE 960
MONITOR	15" COLOR
HARD DISK	1TB
PROCESSOR SPEED	MINIMUM 500MHZ

3.6. DESIGN OF ENTIRE SYSTEM

3.6.1 USE CASE DIAGRAM

Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. Here user and Admin act as actors and Data, Compute Result and Performance, View Trade Exchange, Companies Stock and View predicted outcome act as a use case.

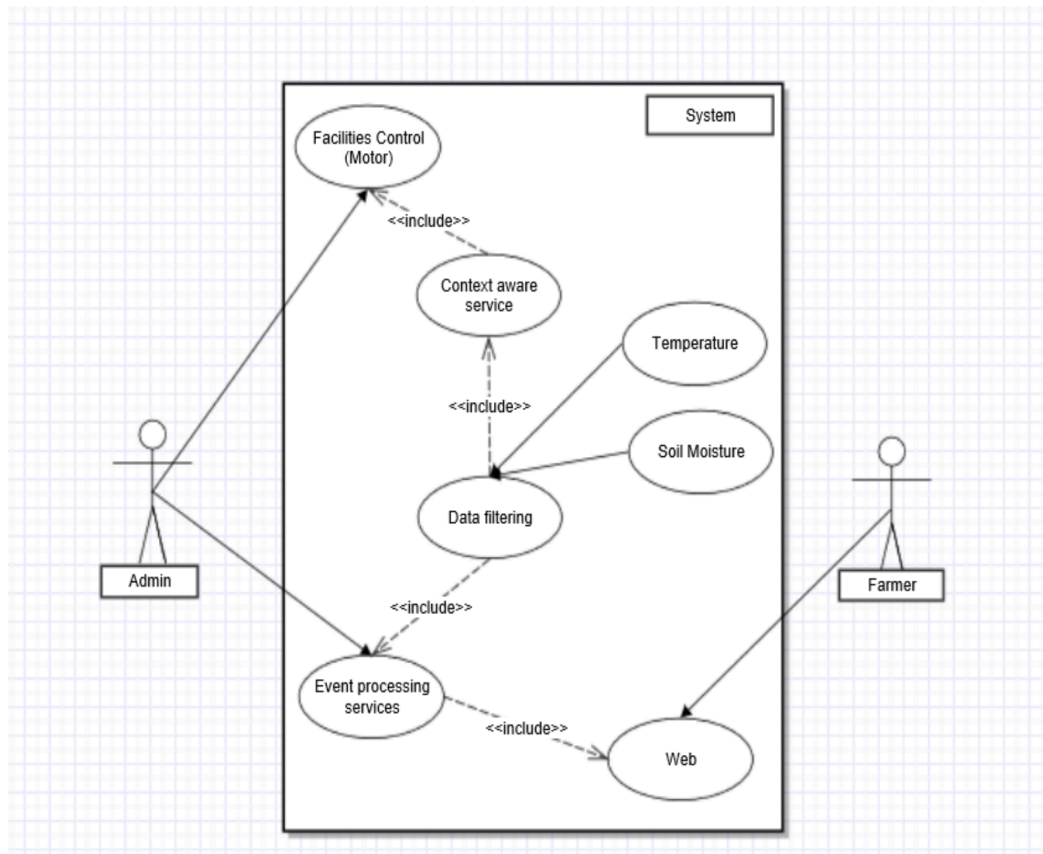


Fig 3.2 Use case diagram for fall detection

Implementation is the phase where vision and plans become reality. This is the logical conclusion after evaluating, deciding, planning and finding the financial resources of a project. Technical implementation is one part of executing a project.

CHAPTER 4

IMPLEMENTATION

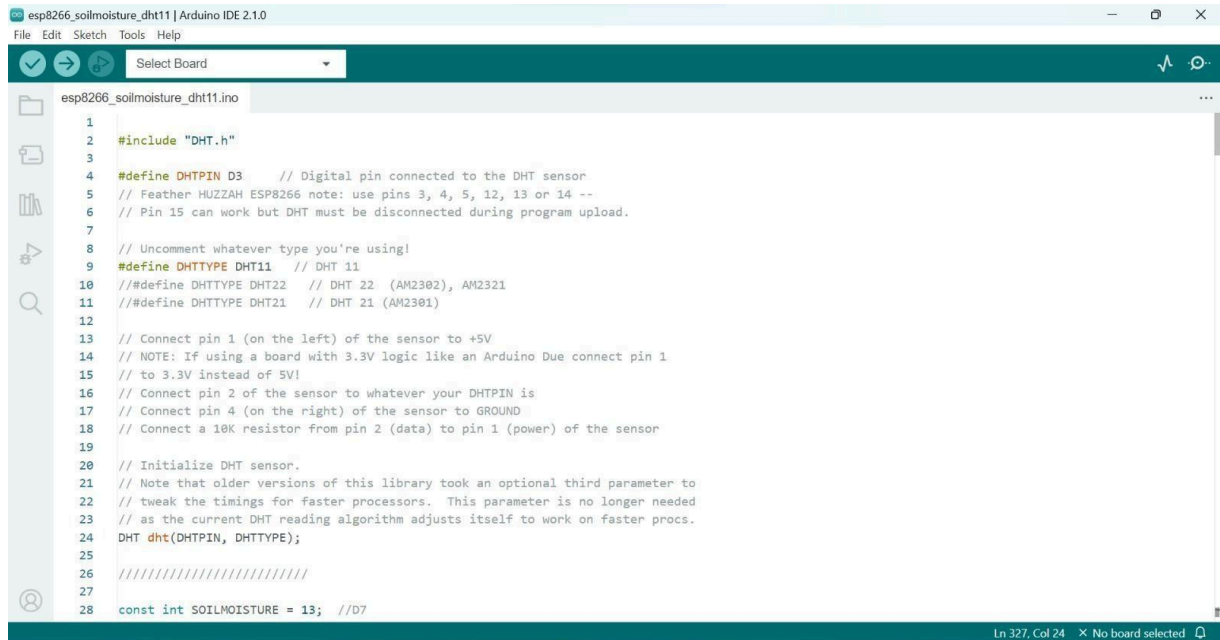
Implementation is the phase where vision and plans become reality. This is the logical conclusion after evaluating, deciding, planning and finding the financial resources of a project. Technical implementation is one part of executing a project.

4.1. MODULES

4.1.2. NodeMCU

Technically speaking, NodeMCU is a firmware for ESP8266 developed using C Programming Language, Espressif NON-OS SDK and Lua scripting language.

Traditionally, we write code for our Microcontrollers like Arduino, STM32, 8051 etc., either in C or C++ and compile it with a set of tools and generate a binary file. This binary file is then uploaded into the flash memory of the microcontroller, and it gets execute. Things are quite different with NodeMCU. You can consider the NodeMCU firmware as an interpreter for Lua Scripts. So, if your ESP8266 is loaded with NodeMCU Firmware, you can simply write your application in Lua and send it to the ESP8266.



```
1
2 #include "DHT.h"
3
4 #define DHTPIN D3 // Digital pin connected to the DHT sensor
5 // Feather HUZZAH ESP8266 note: use pins 3, 4, 5, 12, 13 or 14 --
6 // Pin 15 can work but DHT must be disconnected during program upload.
7
8 // Uncomment whatever type you're using!
9 #define DHTTYPE DHT11 // DHT 11
10 // #define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
11 // #define DHTTYPE DHT21 // DHT 21 (AM2301)
12
13 // Connect pin 1 (on the left) of the sensor to +5V
14 // NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
15 // to 3.3V instead of 5V!
16 // Connect pin 2 of the sensor to whatever your DHTPIN is
17 // Connect pin 4 (on the right) of the sensor to GROUND
18 // Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor
19
20 // Initialize DHT sensor.
21 // Note that older versions of this library took an optional third parameter to
22 // tweak the timings for faster processors. This parameter is no longer needed
23 // as the current DHT reading algorithm adjusts itself to work on faster procs.
24 DHT dht(DHTPIN, DHTTYPE);
25
26 ///////////////////////////////////////////////////
27
28 const int SOILMOISTURE = 13; //D7
```

Ln 327, Col 24 × No board selected

Fig 4.1 Nodemcu

CHAPTER 5

RESULTS AND DISCUSSION

The feature introduced and applied are efficient irrigation based on the readings from the sensors. Changes in the readings were chosen through the simple threshold and then applied to the MPU to solve the problems such as deviation of interpersonal irrigation patterns and similar irrigating actions.

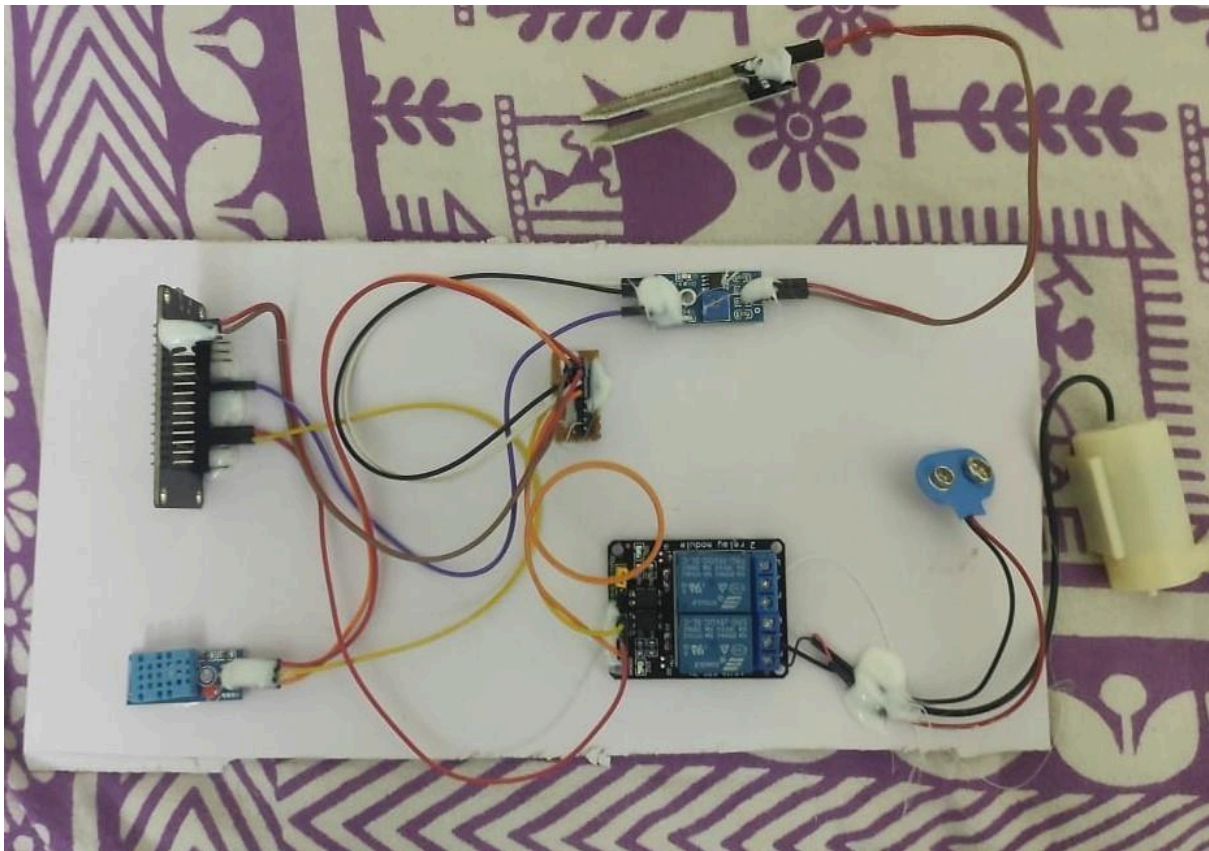
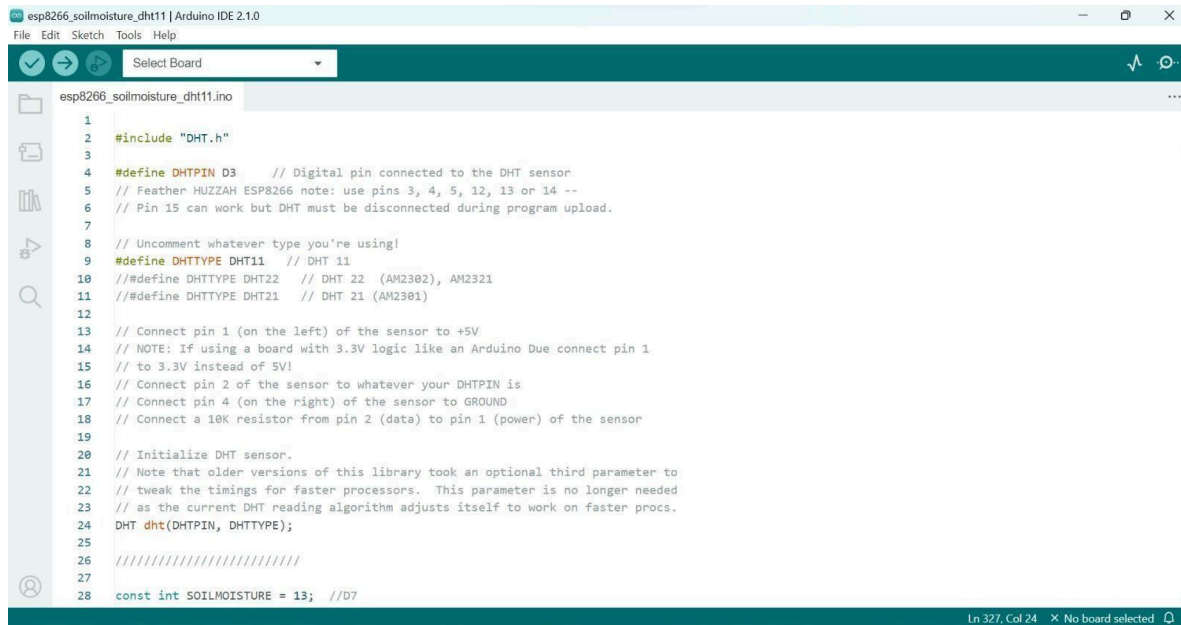


Fig 5.1 Connection of with sensors



The image shows the Arduino IDE 2.1.0 interface. The title bar reads "esp8266_soilmoisture_dht11 | Arduino IDE 2.1.0". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checking, running, and uploading, along with a "Select Board" dropdown menu. The main editor area displays the code for "esp8266_soilmoisture_dht11.ino". The code includes comments for pin connections and sensor initialization. The status bar at the bottom indicates "Ln 327, Col 24" and "No board selected".

```
1
2 #include "DHT.h"
3
4 #define DHTPIN D3 // Digital pin connected to the DHT sensor
5 // Feather HUZZAH ESP8266 note: use pins 3, 4, 5, 12, 13 or 14 --
6 // Pin 15 can work but DHT must be disconnected during program upload.
7
8 // Uncomment whatever type you're using!
9 #define DHTTYPE DHT11 // DHT 11
10 // #define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
11 // #define DHTTYPE DHT21 // DHT 21 (AM2301)
12
13 // Connect pin 1 (on the left) of the sensor to +5V
14 // NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
15 // to 3.3V instead of 5V!
16 // Connect pin 2 of the sensor to whatever your DHTPIN is
17 // Connect pin 4 (on the right) of the sensor to GROUND
18 // Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor
19
20 // Initialize DHT sensor.
21 // Note that older versions of this library took an optional third parameter to
22 // tweak the timings for faster processors. This parameter is no longer needed
23 // as the current DHT reading algorithm adjusts itself to work on faster procs.
24 DHT dht(DHTPIN, DHTTYPE);
25
26 ///////////////////////////////////////////////////
27
28 const int SOILMOISTURE = 13; //D7
```

Fig 5.2 Code compilation in Arduino

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 CONCLUSIONS

In conclusion, the use of an IoT smart farming system that monitors temperature and humidity and irrigates crops accordingly, while also having a web app for real-time monitoring, can significantly improve the efficiency and productivity of modern-day agriculture. By collecting real-time data on environmental conditions, the system can automate irrigation scheduling, optimize resource management, and increase yields. The integration of web-based dashboards and mobile apps also enables farmers to remotely monitor the status of their crops in real-time, making informed decisions and adjusting irrigation settings as needed. While there are challenges associated with the implementation of these systems, such as cost and maintenance, the benefits of increased productivity and sustainability make it a worthwhile investment for the future of agriculture. Overall, a successful working IoT smart farming system has the potential to revolutionize the way we approach farming and address the challenges faced by traditional farming methods.

6.2 FUTURE ENHANCEMENT

In this IOT project, machine learning algorithm shall be implemented so that more irrigation and crop yield efficiency can be achieved. pH monitoring sensors and soil monitoring sensors can be embedded so that we can prevent soil erosion and supply the crops with optimal nutrients.

APPENDIX I

```
#include "DHT.h"

#define DHTPIN D3 // Digital pin connected to the DHT sensor
// Feather HUZZAH ESP8266 note: use pins 3, 4, 5, 12, 13 or 14 --
// Pin 15 can work but DHT must be disconnected during program upload.

// Uncomment whatever type you're using!
#define DHTTYPE DHT11 // DHT 11
// #define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
// #define DHTTYPE DHT21 // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V
// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin
1
// to 3.3V instead of 5V!
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.
// Note that older versions of this library took an optional third parameter to
// tweak the timings for faster processors. This parameter is no longer
needed
// as the current DHT reading algorithm adjusts itself to work on faster
procs.
DHT dht(DHTPIN, DHTTYPE);

////////////////////////////////////

const int SOILMOISTURE = 13; //D7

////////////////////////////////////

#include "ESP8266WiFi.h"
#include "Adafruit_MQTT.h"
```



```
#include "Adafruit_MQTT_Client.h"

/***** WiFi Access Point *****/

#define WLAN_SSID    "@j@" // User ID
#define WLAN_PASS    "87654321" // Pass

/***** Adafruit.io Setup *****/

#define AIO_SERVER    "io.adafruit.com" // Web Side
#define AIO_SERVERPORT 1883 // use 8883 for SSL
#define AIO_USERNAME  "projecttopic56" // N@llapwd202
#define AIO_KEY       "aio_GuHM497sfloweJzOh0HCcbqJhZAU"
// User Name : sathak_adafruit
// Password : 136221

/**** Global State (you don't need to change this!) *****/

// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
// or... use WiFiClientSecure for SSL
//WiFiClientSecure client;

// Setup the MQTT client class by passing in the WiFi client and MQTT
server and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER,
AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

/***** Feeds *****/

// Setup a feed called 'photocell' for publishing.
// Notice MQTT paths for AIO follow the form:
<username>/feeds/<feedname>
Adafruit_MQTT_Publish photocell1 = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/temp");
```




```
Adafruit_MQTT_Publish photocell2 = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/humy");

Adafruit_MQTT_Publish photocell3 = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/msg2");

// Setup a feed called 'onoff' for subscribing to changes.
Adafruit_MQTT_Subscribe onoffbutton =
Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/relay1");

/***** Sketch Code *****/

// Bug workaround for Arduino 1.6.6, it seems to need a function declaration
// for some reason (only affects ESP8266, likely an arduino-builder bug).
void MQTT_connect();

////////////////////////////////////

char arr1[] = "000.00";

char arr2[] = "100";

char arr3[] = "YES";

long temperature2;

char humidity;

char soilmoiture_flag;

char cnt = 0;

void setup()
{
  Serial.begin(115200);
```



```
delay(10);

Serial.println(F("DHTxx test!"));

dht.begin();

pinMode(2, OUTPUT); digitalWrite(2, HIGH);

pinMode(SOILMOISTURE, INPUT);

////////////////////////////////////

Serial.println(F("Adafruit MQTT demo"));

// Connect to WiFi access point.
Serial.println(); Serial.println();
Serial.print("Connecting to ");
Serial.println(WLAN_SSID);

WiFi.begin(WLAN_SSID, WLAN_PASS);
while (WiFi.status() != WL_CONNECTED)
{
  delay(500);
  Serial.print(".");
}
Serial.println();

Serial.println("WiFi connected");
Serial.println("IP address: "); Serial.println(WiFi.localIP());

// Setup MQTT subscription for onoff feed.
mqtt.subscribe(&onoffbutton);
}

void loop()
{
```



```
// Reading temperature or humidity takes about 250 milliseconds!
// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
float h = dht.readHumidity();
// Read temperature as Celsius (the default)
float t = dht.readTemperature();
// Read temperature as Fahrenheit (isFahrenheit = true)
float f = dht.readTemperature(true);

// Check if any reads failed and exit early (to try again).
if (isnan(h) || isnan(t) || isnan(f)) {
  Serial.println(F("Failed to read from DHT sensor!"));
  return;
}

// Compute heat index in Fahrenheit (the
// default) float hif = dht.computeHeatIndex(f, h);
// Compute heat index in Celsius (isFahreheit =
// false) float hic = dht.computeHeatIndex(t, h, false);

Serial.print(F("Humidity:
")); Serial.print(h);
Serial.print(F("% temperature2: "));
Serial.print(t);
Serial.print(F("°C
")); Serial.print(f);
Serial.print(F("°F Heat index: "));
Serial.print(hic);
Serial.print(F("°C "));
Serial.print(hif);
Serial.println(F("°F"));

////////////////////

temperature2 = (long)(t *
100);
```

```
if(temperature2 < 10)
{
    arr1[0] = ' ';
    arr1[1] = ' ';
    arr1[2] = '0';
    arr1[3] = ' ';
    arr1[4] = '0';
    arr1[5] = temperature2 / 1 % 10 + 48;
}
else if(temperature2 < 100)
{
    arr1[0] = ' ';
    arr1[1] = ' ';
    arr1[2] = '0';
    arr1[3] = ' ';
    arr1[4] = temperature2 / 10 % 10 + 48;
    arr1[5] = temperature2 / 1 % 10 + 48;
}
else if(temperature2 < 1000)
{
    arr1[0] = ' ';
    arr1[1] = ' ';
    arr1[2] = temperature2 / 100 % 10 + 48;
    arr1[3] = ' ';
    arr1[4] = temperature2 / 10 % 10 + 48;
    arr1[5] = temperature2 / 1 % 10 + 48;
}
else if(temperature2 < 10000)
{
    arr1[0] = ' ';
    arr1[1] = temperature2 / 1000 % 10 + 48;
    arr1[2] = temperature2 / 100 % 10 + 48;
    arr1[3] = ' ';
    arr1[4] = temperature2 / 10 % 10 + 48;
    arr1[5] = temperature2 / 1 % 10 + 48;
}
```



```
}
else
{
  arr1[0] = temperature2 / 10000 % 10 + 48;
  arr1[1] = temperature2 / 1000 % 10 + 48;
  arr1[2] = temperature2 / 100 % 10 + 48;
  arr1[3] = '.';
  arr1[4] = temperature2 / 10 % 10 + 48;
  arr1[5] = temperature2 / 1 % 10 + 48;
}

humidity =
(char)h;
if(humidity < 10)
{
  arr2[0] = '.';
  arr2[1] = '.';
  arr2[2] = humidity / 1 % 10 + 48;
}
else if(humidity < 100)
{
  arr2[0] = '.';
  arr2[1] = humidity / 10 % 10 + 48;
  arr2[2] = humidity / 1 % 10 + 48;
}
else
{
  arr2[0] = humidity / 100 % 10 + 48;
  arr2[1] = humidity / 10 % 10 + 48;
  arr2[2] = humidity / 1 % 10 + 48;
}

//////////

if(digitalRead(SOILMOISTURE) == LOW)soilmoiture_flag = 0;
```



```
else soilmoiture_flag = 1;
```

```
if(soilmoiture_flag == 0)
{
  arr3[0] = 'Y';
  arr3[1] = 'E';
  arr3[2] = 'S';
}
else
{
  arr3[0] = 'N';
  arr3[1] = 'O';
  arr3[2] = ' ';
}
```

```
// Ensure the connection to the MQTT server is alive (this will make the
first
```

```
// connection and automatically reconnect when disconnected). See the
MQTT_connect
```

```
// function definition further below.
```

```
MQTT_connect();
```

```
// this is our 'wait for incoming subscription packets' busy subloop
```

```
// try to spend your time here
```

```
Adafruit_MQTT_Subscribe *subscription;
```

```
while ((subscription = mqtt.readSubscription(3000)))
```

```
{
  if (subscription == &onoffbutton)
```

```
  {
    Serial.print(F("Got: "));
    Serial.println((char *)onoffbutton.lastread);
```

```
    if(0 == strcmp((char *)onoffbutton.lastread, "OFF"))digitalWrite(2,
HIGH);
```



```
    if(0 == strcmp((char *)onoffbutton.lastread, "ON"))digitalWrite(2,
LOW);
    }
}
```

```
++cnt; if(cnt > 3)cnt = 1;
```

```
    if(cnt == 1)photocell1.publish(arr1);
else if(cnt == 2)photocell2.publish(arr2);
else if(cnt == 3)photocell3.publish(arr3);
```

```
// // Now we can publish stuff!
// Serial.print(F("\nSending photocell val "));
// Serial.print(x);
// Serial.print("...");
// if (! photocell.publish(x++)) {
//   Serial.println(F("Failed"));
// } else {
//   Serial.println(F("OK!"));
// }
```

```
// ping the server to keep the mqtt connection alive
// NOT required if you are publishing once every KEEPALIVE seconds
/*
    if(! mqtt.ping()) {
        mqtt.disconnect();
    }
*/
}
```

```
// Function to connect and reconnect as necessary to the MQTT server.
// Should be called in the loop function and it will take care if connecting.
void MQTT_connect() {
    int8_t ret;
```



```
// Stop if already connected.
if (mqtt.connected()) {
return;
}
Serial.print("Connecting to MQTT... ");
uint8_t retries = 3;
while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
Serial.println(mqtt.connectErrorString(ret));
Serial.println("Retrying MQTT connection in 5 seconds...");
mqtt.disconnect();
delay(5000); // wait 5 seconds
retries--;
if (retries == 0) {
// basically die and wait for WDT to reset me
while (1);
}
}
Serial.println("MQTT Connected!");
}
```


REFERENCES

- [1]. Dr. H. Shaheen, E. Himabindu, Dr. T. Sreenivasulu, Dr. Rajasekar Rangasamy; International Journal of Engineering and Technology; 2019
- [2]. J. Tomkun and B. Nguyen, —Design of a Fall Detection and Prevention System for the Elderly, In EE 4BI6 Electrical Engineering Biomedical Capstones, Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada, April 23, 2010.
- [3]. G. Sannino and G.D. Pietro, —An Advanced Mobile System for Indoor Patients Monitoring, In Proc. 2nd International Conference on Networking and Information Technology (ICNIT 2011), pp. 17, Singapore, IACSIT Press, 2011
- [4]. A.H. Nasution and S. Emmanuel, —Intelligent Video Surveillance for Monitoring Elderly in Home Environments, In Proc. IEEE 9th Workshop on Multimedia Signal Processing (MMSP 2007), pp. 203-206, Greece, 2007
- [5]. Berg RL et al.; National Academy of Sciences; Institute of Medicine (US): 1992
- [6]. FalinWu, Hengyang Zhao, Yan Zhao and Haibo Zhong, Fei Hu: International Journal of Telemedicine and Applications: 2015. Find the paper here.
- [7]. George F. Fuller, COL, MC, USA, White House Medical Clinic, Washington, D.C.; Am Fam Physician, 2000. Find the paper here.
- [8]. M.E. Tinetti and M. Speechley, —Prevention of Falls Among the Elderly, The New England Journal of Medicine, vol. 320, no. 16, 1989, pp. 1055-1059. [9]. A.K. Bourke, P.V.D. Ven, M. Gamble, R. OConnor, K. Murphy, E. Bogan, E. McQuade, P. Finucane, G. O'laighin, and J. Nelson, —Evaluation of Waist-mounted Tri-axial Accelerometer Based Fall-detection Algorithms During Scripted and Continuous Unscripted Activities, Journal of Biomechanics, vol. 43, no. 15, 2010, pp. 3051-3057

