PERIÓDICO TCHÊ QUÍMICA
ARTIGO ORIGINAL

SISTEMA DE MONITORAMENTO AMBIENTAL E DE SEGURANÇA AGRÍCOLA BASEADO EM IoT

IoT-BASED AGRICULTURE ENVIRONMENT AND SECURITY MONITORING SYSTEM

Priyanka Sarma
Assistant Professor, Department of Computer Science & Electronics, University of Science & Technology, Meghalaya, India.

Dr. Atowar ul Islam*
Associate Professor, Department of Computer Science & Electronics, University of Science & Technology, Meghalaya, India.

Tony Bayan
Assistant Professor, Department of Computer Science, K.C Das Commerce College, Gauhati University, India.

**Corresponding author*
e-mail: atowar91626@gmail.com

Received 06 March 2023; received in revised form 20 June 2023; accepted 10 July 2023

RESUMO

Introdução: A agricultura é uma parte vital da economia que alimenta a população mundial em expansão. No entanto, o setor enfrenta várias dificuldades, incluindo restrições de água, degradação do solo e mudanças climáticas. A demanda por maior segurança nas atividades agrícolas agrava esses problemas. Este artigo sugere um sistema de monitoramento ambiental e de segurança agrícola baseado em IoT para abordar essas questões. **Objetivos**: O principal objetivo deste trabalho é desenvolver um sistema inteligente de monitoramento e irrigação automatizada que minimize o uso de água com base em condições individuais e permita métodos ambientais em tempo real. **Métodos**: O microcontrolador Arduino Uno R3 baseado no Microchip ATmega328P é usado para coletar dados ambientais, como umidade, temperatura e níveis de umidade do solo. A placa Arduino é integrada a um relé e um módulo RTC para regar as plantas em momentos precisos. Também possui um sensor infravermelho passivo para detectar intrusos. A câmera ESP32 também é usada para registrar automaticamente a condição atual do solo agrícola. Uma amostra de leitura em intervalo de 1 hora é registrada. Esse método envolve a transmissão sem fio dos dados para plataformas baseadas em nuvem. **Resultados**: O valor do sensor no início de uma condição úmida é 0. Quando isso ocorre, a bomba do motor é desligada e o valor medido é comunicado ao microcontrolador por meio do NodeMCU. A bomba do motor liga e desliga automaticamente quando as plantas recebem quantidade suficiente de água. **Discussão**: Observou-se que a bomba de água começa assim que o limite de temperatura de 19,5 °C é atingido. Foi observado que, à medida que a umidade aumenta, não é necessário regar as plantas, pois elas absorvem o vapor de água da superfície das folhas em condições úmidas. **Conclusões**: Esta pesquisa oferece um modelo de protótipo conveniente que pode medir simultaneamente temperatura, umidade e umidade do solo, identificar intrusões e monitorá-las remotamente pela internet.

Palavras-Chave: *Internet das Coisas, sensores de temperatura, sensores de umidade, sensores de movimento*

ABSTRACT

Background: Agriculture is a vital part of the economy that feeds a part of the world expanding population. However, the sector faces several difficulties, including water constraints, soil degradation, and climate change. The demand for increased security and safety in agricultural activities worsens these problems. This paper suggests an IoT-based agriculture environment and security monitoring system address these issues. **Aims**: The main objective of this work is to develop an intelligent monitoring and automated irrigation system that minimizes water use based on individual conditions and enables real-time environmental **Methods**: The Arduino Uno R3 microcontroller based on Microchip ATmega328P is used to gather environmental data such as humidity, temperature, soil moisture levels. The Arduino board is integrated with a relay and RTC module to water plants at precise times. It also has a passive infrared sensor to detect intruders. ESP32 camera is also used to record the current condition of the agricultural ground automatically. A sample of reading in an interval of 1 hour is recorded. This method involves wirelessly transmitting the data to cloud-based platforms. **Results**: The value of the sensor at the start of a wet condition is 0. When this occurs, the motor pump turns off, and the measured value is communicated to the microcontroller through NodeMCU. The motor pump automatically turns ON and OFF when plants receive enough amount of water. **Discussion**: It has been observed the water pump starts as soon as the temperature threshold of 19.5 °C is achieved. It has been observed that as humidity increases, it is unnecessary to water the plants as the plants take up water vapor from the surface of the leaves under humid conditions. **Conclusions**: This research offers a convenient prototype model that can simultaneously measure temperature, moisture, and humidity, identify intrusion, and remotely monitor it over the internet. It was concluded that with the decrease in moisture of the soil, the plants need to be watered.

Keywords: *Internet of Things, temperature sensors, humidity sensors, motion sensors.*

1. INTRODUCTION

Agriculture plays a vital role in feeding the world's expanding population. Climate change, soil erosion, and a lack of water are just a few of the issues the sector must deal with. Productivity, crop yields, and food security are all significantly impacted by these issues. In addition, agricultural businesses are susceptible to theft, invasions, and other security risks that can result in large losses. A security and environment monitoring system for agriculture based on the Internet of Things can be utilized to address these issues. However, due to lack of time, manual monitoring of the agricultural ground by the farmers is very difficult. Although most farmers spend much time in crop fields, manual monitoring also takes time. In addition, farmers using the manual monitoring method will not be aware of any intruders—people or animals—in the crop who might harm or steal the plants. As a result, in this project, an automated agriculture monitoring system that can help farmers automatically and effectively water their plants when necessary as well as automatically check the quality of the agriculture soil nutrition and also automatically detect an intrusion is developed, taking into consideration the crucial desire to improve crop productivity as well as ensure water consumption efficiency and ensuring the quality of the crops.

The suggested system uses sensors to keep track of several environmental factors, including temperature, humidity, soil moisture, and water quality. These sensors can be positioned in the ground, the air, or the sea, continuously gathering data. The sensors are connected to a microcontroller, such as the Arduino Uno, which processes the information and wirelessly transfers it to a platform in the cloud. The data can be evaluated to make irrigation, fertilizing, and other agricultural methods more effective. The technology, for instance, can be used to choose the ideal irrigation period depending on soil moisture and environmental factors. The suggested system also has security components, including cameras and motion detectors, to deter theft and invasions. The cameras may be installed in key locations to monitor agricultural operations, and the cloud platform allows remote access. Motion sensors detect any unwanted movement in the agricultural fields, which can then raise the alarm or send a message to the farmers or security staff. The technology can also be used to follow vehicles, monitor livestock, and spot fires before they start. The main objective of this work is to develop an intelligent monitoring and automated irrigation system that minimizes water use based on individual conditions and enables real-time environmental

1.2 Literature Review

Some works of various authors are discussed based on IoT-based agriculture soil monitoring systems:

The authors R. Singh *et al*. (2020), first discuss the challenges faced by the agricultural sector in terms of crop production, such as climate change, unpredictable weather patterns, and increasing population. They then propose a solution in the form of a monitoring system that can help farmers make data-driven decisions and optimize their crop yields. The monitoring system comprises IoT sensors deployed in the agricultural field to collect data such as temperature, humidity, and soil moisture. This data is then transmitted to a cloud-based platform and processed using AI algorithms to generate insights and recommendations for farmers.

The paper discusses the technical details of the system, including the hardware and software components, the communication protocols used, and the data processing pipeline. The authors also describe the results of a field trial conducted to evaluate the effectiveness of the system in improving crop yields.

Overall, the paper presents an innovative solution that leverages the power of AI and IoT to address the challenges the agricultural sector faces. The proposed monitoring system can potentially improve crop yields and enhance the sustainability of agriculture by enabling farmers to make informed decisions based on real-time data.

The authors Baranwal and Pateriya (2016) discuss the motivation for the project, which is to address the challenges faced by farmers in protecting their crops from theft and damage. They then describe the technical details of the system, including the hardware and software components, communication protocols, and user interface. The system comprises various IoT devices, including sensors and actuators, that are connected to a central hub. The hub is a gateway to a cloud-based platform, where the system is managed and controlled. The authors also describe the communication protocols, including Wi-Fi and GSM, to enable seamless connectivity between the devices.

The paper goes on to discuss the features and capabilities of the system, which include real-time monitoring of environmental factors such as temperature and humidity, as well as remote control of security devices such as alarms and cameras.

The authors also evaluate the system's performance and reliability through various tests and simulations, demonstrating the system's ability to provide an efficient and user-friendly way to monitor and secure agricultural assets.

The authors R. Kumar & Rajasekaran (2016), provide an overview of the current state of patient monitoring systems and discuss the limitations of traditional approaches. They then describe the architecture of the proposed system, which includes various sensors, a central hub, and a cloud-based platform.

The system uses Bluetooth Low Energy (BLE) communication protocol to enable seamless connectivity between the devices, and the authors describe the different sensors used for monitoring various health parameters such as heart rate, blood pressure, and temperature. The authors also discuss the user interface of the system, which allows healthcare professionals to monitor the health status of patients in real-time remotely. They also describe the cloud-based platform used for storing and processing the data generated by the system, which can be used for further analysis and decision-making.

The paper goes on to discuss the performance and reliability of the system through various tests and simulations, demonstrating the ability of the system to provide an efficient and accurate way to monitor the health status of patients.

Overall, the paper presents a well-designed and useful IoT-based patient monitoring system, which can potentially improve healthcare quality by enabling healthcare professionals to remotely monitor patients in real time and make informed decisions based on the data generated by the system.

Ferrag *et al*. (2020) presented research challenges on security and privacy issues in green IoT-based agriculture and classified threat models against green IoT-based agriculture into five categories, including attacks against privacy, authentication, confidentiality, availability, and integrity properties. The authors also analyzed the privacy-oriented blockchain-based solutions and consensus algorithms for IoT applications and how they will be adapted for green IoT-based agriculture. The privacy-oriented blockchain-based solutions for IoT applications and how they will be adapted for green IoT-based agriculture are also discussed.

The authors T. A. Singh and Chandra (2018) discuss the motivation for the project, which is to address the challenges greenhouse farmers face in monitoring and controlling the environmental factors that affect crop growth and yield. They then present the architecture of the IoT-based monitoring system, which includes a set of sensors and actuators deployed throughout the greenhouse to measure and control environmental factors such as temperature, humidity, light, soil moisture, and CO_2 levels.

The paper also describes the software components of the system, which include a cloud-based IoT platform for data storage and management, a web-based dashboard for remote monitoring and control, and an analytics engine for data analysis and decision-making. The authors then present the results of experiments conducted to evaluate the performance of the system, which demonstrates its effectiveness in improving crop yield and reducing energy consumption.

The paper of Collado *et al*. (2021) suggests an agroclimatic monitoring system for greenhouses using Internet of Things (IoT) technology to solve the issues with agricultural productivity and food security brought on by climate change. The system comprises monitoring stations, a wireless communication network, and a data processing and visualization platform. The main goal of the paper is to offer a low-cost, effective, and simple-to-use instrument for tracking agroclimatic factors and forewarning against unfavorable climatic changes.

Muangprathub *et al*. (2019) proposed a wireless sensor network-based optimum crop irrigation system. The system consists of three components: hardware (control box and soil moisture sensors), a web application, and a mobile application. The web-based application allows the manipulation of crop data and field information. The system was implemented and tested in Makhamtia District, Suratthani Province, Thailand. The results showed that the system was useful in agriculture, maintaining appropriate soil moisture levels for vegetable growth. This led to reduced costs and increased agricultural productivity.

Kamienski *et al.* (2019), proposes an IoT-based smart water management platform known as a surface water ambient monitoring platform (SWAMP) for agricultural precision irrigation. The platform uses technology to provide plants with the precise quantity of water they require, boosting agricultural productivity, lowering expenses, and fostering environmental sustainability. The paper presents the deployment scenarios and highlights the practical implementation of the platform in real-world agricultural settings. Results analysis of the system indicate that the platform can meet the performance requirements of the pilots but may require specific configurations and re-engineering of components to achieve higher scalability using fewer computational resources.

Suma (2021), provide an overview of various IoT-based smart agriculture systems and technologies. The paper discusses the benefits of using IoT in agriculture, such as improved crop yield and reduced water usage, and the various components of an IoT-based smart agriculture system, such as sensors, actuators, and communication protocols. The authors also discuss the challenges and limitations of IoT-based smart agriculture systems, such as the high cost of implementing such systems and the need for reliable connectivity in remote agricultural areas. The paper concludes with discussing future research directions in IoT-based smart agriculture, including machine learning and big data analytics to further improve agricultural productivity.

Chen *et al*. (2022) present a design and implementation of an IoT-based water quality monitoring system for fish farming management. The system uses various sensors to measure the water quality parameters such as temperature, dissolved oxygen, pH, and ammonia concentration. The data is transmitted to a cloud server through a wireless network for storage and analysis. The system also provides real-time alerts and notifications to farmers in case of any critical deviation in water quality parameters. The authors claim that the proposed system can help farmers to improve the yield and quality of fish by providing an efficient monitoring and management system.

Kim *et al*. (2020) provided an overview of the various applications of IoT in agriculture. The authors discuss how IoT can be used for precision farming, crop management, irrigation management, and livestock monitoring. They also provide a detailed analysis of the challenges associated with implementing IoT in agriculture, including data privacy and security issues and the need for adequate infrastructure and technical expertise. The paper concludes with a discussion of the potential benefits of IoT in agriculture, including increased efficiency and productivity, reduced waste, and improved sustainability.

S. Kumar *et al*. (2019) described developing and implementing an IoT-based smart farming system that uses intelligent sensors to monitor temperature and moisture levels in soil and transmit data wirelessly to a cloud server for storage and analysis. A solar panel powers the system and incorporates an Arduino microcontroller for data processing and transmission. The paper also includes a detailed explanation of the system architecture and the results of the experimental system evaluation. The main contribution of the paper is the design and implementation of a low-cost, energy-efficient, and reliable IoT-based smart farming system that can help farmers optimize crop production by providing real-time data on soil moisture and temperature levels.

Kaur *et al*. (2023) presented a comparative analysis of an IoT-based plant growth monitoring system for hydroponic indoor vertical farms in controlled and uncontrolled environments. The study used two monitoring systems to compare the growth rate, nutrient uptake, and water consumption of lettuce plants grown under different conditions. The study results showed that the IoT-based controlled environment system provided better growth conditions for the lettuce plants with higher growth rates, better nutrient uptake, and lower water consumption. The paper concludes that IoT-based monitoring systems can significantly improve the efficiency and productivity of indoor vertical farms by providing real-time monitoring and control of environmental conditions.

In order to meet the problems posed by climate change and the rising need for food production, the author of the paper *Hernández-Morales et al.*, (2022) provides a scalable IoT-based monitoring system for agricultural applications. It provides a practical solution with cheap implementation and management costs.

Sensing, networking, processing, and applications are the four layers that make up the system. The system was successfully tested and certified by tracking the temperature and humidity in a greenhouse for six months. Accurate temperature forecasts 24 hours in advance were made possible by including a data-driven predictive algorithm. Overall, the results point to the potential of the proposed IoT framework to monitor crops, improve growth conditions, and boost agricultural production.

Rabka *et al*. (2022) discussed developing an IoT-based horticulture monitoring system, which monitors and controls environmental parameters such as temperature, humidity, and light to optimize crop growth and yield. The system consists of various sensors that measure environmental parameters, a microcontroller that collects and sends data to a cloud server, and a web application that allows users to monitor and control the system remotely. The system is designed to be flexible and adaptable to different types of crops and growing environments. The paper also discusses the potential benefits of the system, such as increased yield, reduced resource consumption, and improved crop quality.

Ramaprasad *et al.* (2019) proposed a scientific approach to irrigation using a smart system based on soil moisture content. The system utilizes an Arduino microcontroller and sensors to monitor the moisture level in the soil. Based on this information, the system controls the water pump, ensuring optimal water usage scientifically. The system also measures the temperature and humidity of the agricultural field using dedicated sensors. The implementation of a scientific irrigation system using these technologies offers several advantages. It reduces the need for manpower and physical monitoring of crop fields, leading to increased efficiency.

Ul Islam *et al*. (2022) proposed an IoT-based web interface to monitor LPG gas linkage systems. It also provides more safety and security and updates the owner regularly.

It has been noted that these research publications show the rising interest in IoT-based soil monitoring systems for agriculture and emphasize the potential advantages of adopting such systems to enhance agricultural yields. Despite the great potential these background studies have revealed, certain research gaps still need to be filled. Here are several examples:

- **Interoperability and data standardization**: One of the difficulties with IoT-based monitoring systems is that data formats and communication protocols are frequently not standardized. This may make it challenging to combine various tools and systems and may reduce the value of the gathered data.
- **Security and privacy:** IoT-based monitoring systems are susceptible to hacks and data breaches, which may expose the security and privacy of the information gathered. Research is required to find efficient security measures and privacy-preserving methods that can safeguard sensitive data.
- **Usability and user acceptance:** For farmers to use IoT-based monitoring systems effectively, they must be simple and intuitive. Research is required to determine the best way to construct these systems to accommodate the demands and preferences of farmers and other stakeholders.
- **Cost-effectiveness:** Implementing and maintaining IoT-based monitoring systems can be expensive. Research is required to determine how to make these systems more affordable, especially for small-scale farmers who do not have access to the same resources as larger farms. The full potential of IoT-based agriculture environment and security monitoring systems will need addressing these research gaps.

Therefore, this research aims to develop an intelligent agricultural system that fills in the previously identified research gaps and provides farmers with a remote, safe, and user-friendly solution that they can use anytime and anywhere.

2. MATERIALS AND METHODS:

2.1. Materials

Temperature and humidity sensor:
Temperature and humidity sensors are electronic devices designed to measure and monitor the surrounding temperature and humidity levels. The manufacturing process of temperature and humidity sensors can vary depending on the specific type and technology used. It involves techniques such as thin-film deposition, etching, and encapsulation, to create the sensor elements. The sensors are then integrated with suitable signal conditioning circuitry and packaged for

protection and ease of use. The final product undergoes rigorous quality testing to ensure accurate and reliable measurements.

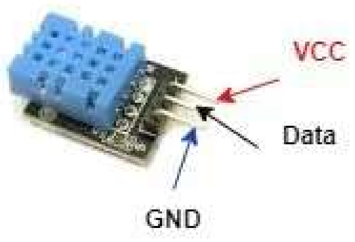

Figure 3. Temperature, humidity Sensor

This research employed a temperature humidity sensor in our suggested system, which is connected to the microcontroller board and measures both humidity and temperature simultaneously from the testing location as soon as the board is turned on. Figure 3 below shows the image of the temperature and humidity sensor.

Soil Moisture Sensor: Soil moisture sensors are devices designed to measure the water content in the soil. They are crucial in agriculture, horticulture, and environmental monitoring applications. They come in different models, including capacitive sensors, tensiometers, and time domain reflectometry (TDR) sensors, each employing different technologies to measure soil moisture. The manufacturing process of soil moisture sensors involves several steps. It begins with designing the sensor element based on the chosen technology and fabricating sensor elements using specific techniques. Signal conditioning circuitry is incorporated to process the sensor output, and the sensors are packaged and enclosed for protection. Calibration is performed to ensure accurate measurements, and extensive quality testing is conducted before the sensors are released for commercial use.

Figure 4 depicts a soil moisture sensor, a device that measures the amount of moisture in the sand. We used a soil moisture sensor in our proposed system, which is connected to a microcontroller board and detects a scarcity of water in agricultural areas when modules produce high output.

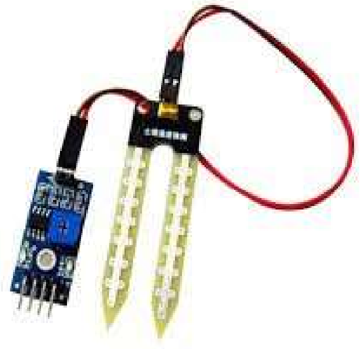

Figure 4. Soil Moisture Sensor

IR sensor: An infrared (IR) sensor is a device designed to detect and measure infrared radiation. It consists of various components that form its architecture. These include an IR source (such as an IR LED), optics for gathering and focusing the IR radiation, a sensor element (such as a photodiode or pyroelectric sensor) responsible for converting the IR radiation into an electrical signal, signal processing circuitry for amplifying and conditioning the signal, and an output interface for delivering the processed signal. Additionally, some IR sensors also combine a distance sensor and processing circuit, providing outputs such as distance measurement, an analog representation of the detected object, and a binary signal indicating the presence of any object alongside the analog representation...

It proposed a system that used an IR sensor to detect when something entered the field. An IR sensor is shown in Figure 5. When this sensor is turned on, data is sent to the user and the cloud.

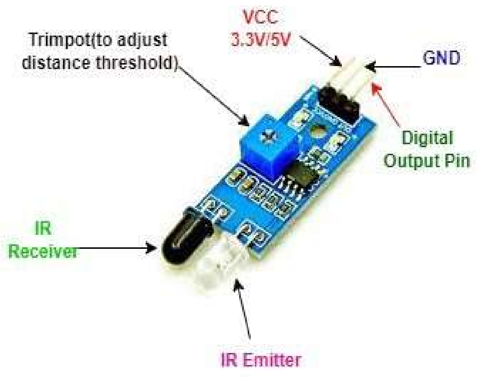

Figure 5. IR Sensor

Relay: Relays are electromechanical devices used to control and switch electrical circuits. Relays are widely used in various industries, including automotive, industrial automation, telecommunications, and power systems. The specific model and manufacturing process can vary between manufacturers, but the overall principles consistently produce reliable and efficient relays for different applications. When selecting a relay for agricultural monitoring, it is important to consider the specific requirements of the application, such as the voltage and current ratings of the devices being controlled, the expected operating conditions, and the reliability and longevity required. In the proposed system it was used a relay to maintain level of the moisture of the crop, and a relay is used to activate the water pump. Figure 6 depicts an electrically operated switch that uses a relay.

Figure 6. Relay

Water Pump: A cheap, compact submersible pump motor is shown in 7 below. It is a DC 3-6V tiny Micro Submersible water pump. It requires a power supply ranging from 2.5 to 6V to function. It has a very low current consumption of 220mA and can pump up to 120 liters per hour.

Figure 7. Water Pump

2.2. Methods

2.2.1. Experimental conditions.

The experiment was performed in outdoor conditions in the fields. State that the environmental conditions were almost the same when collecting data. To be precise, the data was collected in bright, normal sunlight during the day.

2.2.2. Experimental Work Flow

2.2.2.1 Read Data from Sensors

Input: The values from the three sensors
S1← Soil Moisture, S2←Temperature,
S3 ←water level sensor, S4 ← Humidity
Output: Text message notification to the farmer about the condition of the crop.

(Arduino + nrf24L01 module + Soil Moisture Sensor)

- Start
- Receive the Soil Moisture Sensor's value and
- Then data is forwarded as input to an Arduino
- Then data is forwarded as input to an Arduino
- Next, wirelessly send the value using nrf24L01.

2.2.2.2 Send Data to the Server

**(ESP8266 Wi Module, Arduino, nrf24L01 Module, and
DHT11 Sensor)**

- Start
- Get the value from the soil moisture sensor node using the nrf24L01 module.
- Receive the temperature and humidity values from the DHT11 sensor, then process them on the Arduino.
- After that, transmit all values via the ESP8266 Wi Module to the ThingSpeakServer.

(Mobile App + Relay Module)
- Start
- NodeMCU processes the values after reading them from the thing speak server.
- Transmit the values via a mobile app (Blynk).
- Choose whether to switch the motor on or off manually or automatically.
- The pump automatically turns on if the soil moisture measurement exceeds its threshold value.

- The pump automatically shuts off if the soil moisture exceeds its threshold value.
- Via the Blynk mobile app, the pump may be turned on or off between 61% and 79%.
- Stop.

2.2.2.3 Working Principle

Step 1: First, the code is written in the Arduino IDE, and then it is uploaded to the Arduino board. Arduino sends data to a cloud server named ThingSpeak based on sensor behavior.

Step 2: Then, the Arduino board is connected to all the sensors, the Wi-Fi module, the relay switch, the led, and the buzzer.

Step 3: When using a data cable to link an Arduino board and an IDE (Integrated Development Environment), the cable supplies the electricity needed to run the Arduino Board's hardware and view the serial output.

Step 4: The project begins to function after the data has been uploaded to the Arduino hardware and connected to an Arduino IDE.

Step 5: The Arduino board then begins to function based on the behavior of the sensor.

2.2.3 Source codes

The codes are included below with the proper comments, describing the inputs, output, data transfer, and data acquisition rate.

2.2.3.1 Arduino code

```
//Include the needed library, it was used softer serial
communication with the ESP8266
#include <SoftwareSerial.h>
#include <avr/power.h>
//LCD config
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,20,4);  //sometimes the
LCD address is not 0x3f. Change to 0x27 if it does not
work.
//Define the used
#define ESP8266_RX 10  //Connect the TX pin from
the ESP to this RX pin of the Arduino
#define ESP8266_TX 11  //Connect the TX pin from
the Arduino to the RX pin of the ESP
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
int Potentiometer_1 = A0;
int Potentiometer_2 = A1;
int Potentiometer_3 = A2;
int Potentiometer_4 = A3;
int switch1 = 7;
int switch2 = 8;
int switch3 = 9;

const char SSID_ESP[] = "MiFibra-4132";        //Give
EXACT name of your WIFI
const char SSID_KEY[] = "vzP5anY5";            //Add
the password of that WIFI connection
const char* host =
"noobix.000webhostapp.com";  //Add the host without
"www" Example: electronoobs.com
String NOOBIX_id = "99999";                   //This is the
ID you have on your database, and I have used 99999
because there is a maximum of 5 characters
String NOOBIX_password = "12345";             //Add
the password from the database, also maximum 5
characters and only numerical values
String location_url = "/TX.php?id=";          //location of
your PHP file on the server. In this case, the TX.php is
directly in the first folder of the server
                                              //If you have the files in
a different folder, add that as well. Example:
"/ESP/TX.php?id="     Where the folder is ESP
//Used variables in the code
String url = "";
String URL_withPacket = "";
unsigned long multiplier[] =
{1,10,100,1000,10000,100000,1000000,10000000,10
0000000,1000000000};
//MODES for the ESP
const char CWMODE = '1';//CWMODE 1=STATION,
2=APMODE, 3=BOTH
const char CIPMUX = '1';//CWMODE 0=Single
Connection, 1=Multiple Connections

//Define the used functions later in the code, thanks to
Kevin Darrah, YT
channel:  https://www.youtube.com/user/kdarrah1234
boolean setup_ESP();
boolean read_until_ESP(const char keyword1[], int
key_size, int timeout_val, byte mode);
void timeout_start();
boolean timeout_check(int timeout_ms);
void serial_dump_ESP();
boolean connect_ESP();
void connect_webhost();
unsigned long timeout_start_val;
char scratch_data_from_ESP[20];//first byte is the
length of bytes
char payload[200];
byte payload_size=0, counter=0;
char ip_address[16];
//Variable to SEND to the DATABASE
bool sent_bool_1 = 0;
bool sent_bool_2 = 0;
bool sent_bool_3 = 0;
int  sent_nr_1 = 0;
int  sent_nr_2 = 0;
int  sent_nr_3 = 0;
int  sent_nr_4 = 0;
int  sent_nr_5 = 0;

//Variable RECEIVED from the DATABASE
bool received_bool_1 = 0;
bool received_bool_2 = 0;
bool received_bool_3 = 0;
bool received_bool_4 = 0;
bool received_bool_5 = 0;
int  received_nr_1 = 0;
int  received_nr_2 = 0;
int  received_nr_3 = 0;
int  received_nr_4 = 0;
int  received_nr_5 = 0;
String received_text = "";
//Store received chars in these variables
char t1_from_ESP[5];  //For time from web
char d1_from_ESP[2];  //For received_bool_2
char d2_from_ESP[2];  //For received_bool_2
char d3_from_ESP[2];  //For received_bool_3
char d4_from_ESP[2];  //For received_bool_4
char d5_from_ESP[2];  //For received_bool_5
char d9_from_ESP[6];  //For received_nr_1
char d10_from_ESP[6]; //For received_nr_2
char d11_from_ESP[6]; //For received_nr_3
char d12_from_ESP[6]; //For received_nr_4
char d13_from_ESP[6]; //For received_nr_5
char d14_from_ESP[300]; //For received_text


//DEFINE KEYWORDS HERE
const char keyword_OK[] = "OK";
const char keyword_Ready[] = "Ready";
const char keyword_no_change[] = "no change";
const char keyword_blank[] = "#&";
const char keyword_ip[] = "192.";
const char keyword_rn[] = "\r\n";
const char keyword_quote[] = "\"";
const char keyword_carrot[] = ">";
const char keyword_sendok[] = "SEND OK";
const char keyword_linkdisc[] = "Unlink";

const char keyword_t1[] = "t1";
const char keyword_b1[] = "b1";
const char keyword_b2[] = "b2";
const char keyword_b3[] = "b3";
const char keyword_b4[] = "b4";
const char keyword_b5[] = "b5";
const char keyword_n1[] = "n1";
const char keyword_n2[] = "n2";
const char keyword_n3[] = "n3";
const char keyword_n4[] = "n4";
const char keyword_n5[] = "n5";
const char keyword_n6[] = "n6";
const char keyword_doublehash[] = "##";
SoftwareSerial ESP8266(ESP8266_RX,
ESP8266_TX);// rx tx
void setup(){//       SETUP     START
 lcd.init();              //Init the LCD
 lcd.backlight();            //Activate backlight

 //Pin Modes for ESP TX/RX
 pinMode(ESP8266_RX, INPUT);
 pinMode(ESP8266_TX, OUTPUT);
 pinMode(LED1, OUTPUT);
 pinMode(LED2, OUTPUT);
 pinMode(LED3, OUTPUT);
 pinMode(LED4, OUTPUT);
 pinMode(LED5, OUTPUT);
 pinMode(Potentiometer_1, INPUT);
 pinMode(Potentiometer_2, INPUT);
 pinMode(Potentiometer_3, INPUT);
 pinMode(Potentiometer_4, INPUT);
 pinMode(switch1, INPUT);
 pinMode(switch2, INPUT);
 pinMode(switch3, INPUT);
 digitalWrite(LED1,LOW);
 digitalWrite(LED2,LOW);
 digitalWrite(LED3,LOW);
 digitalWrite(LED4,LOW);
 digitalWrite(LED5,LOW);

 ESP8266.begin(9600);//default baud rate for ESP
 ESP8266.listen();//not needed unless using other
software serial instances
 Serial.begin(9600); //for status and debug
 delay(2000);//delay before kicking things off
 setup_ESP();//go set up the ESP
 }
void loop(){
 sent_nr_1 = analogRead(Potentiometer_1);
 sent_nr_1 = analogRead(Potentiometer_2);
 sent_nr_1 = analogRead(Potentiometer_3);
 sent_nr_1 = analogRead(Potentiometer_4);
 sent_bool_1 = digitalRead(switch1);
 sent_bool_2 = digitalRead(switch2);
 sent_bool_3 = digitalRead(switch3);
 send_to_server_1();
 digitalWrite(LED1,received_bool_1);
 digitalWrite(LED2,received_bool_2);
 digitalWrite(LED3,received_bool_3);
 digitalWrite(LED4,received_bool_4);
 digitalWrite(LED5,received_bool_5);

 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("N1: "); lcd.print(received_nr_1);
lcd.print("  N2: "); lcd.print(received_nr_2);
 lcd.setCursor(0,1);
 lcd.print("N3: "); lcd.print(received_nr_3);
lcd.print("  N4: "); lcd.print(received_nr_4);
 delay(1000);//5 seconds between tries
 send_to_server_5();
 digitalWrite(LED1,received_bool_1);
 digitalWrite(LED2,received_bool_2);
 digitalWrite(LED3,received_bool_3);
 digitalWrite(LED4,received_bool_4);
 digitalWrite(LED5,received_bool_5);
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print(received_text);

 delay(1000);

}//End of the main loop
```

2.2.3.2 Server-side code

```
void send_to_server_1(){
url = location_url;
url += NOOBIX_id;
url += "&pw=";
url +=  NOOBIX_password;//sensor value
url += "&un=1";
url += "&n1=";
url +=  String(sent_nr_1);//sensor value

URL_withPacket = "";
URL_withPacket = (String("GET ") + url +
" HTTP/1.1\r\n" +
                "Host: " + host + "\r\n" +
                "Connection:
close\r\n\r\n");

 counter=0;//keeps track of the payload
size
 payload_size=0;
 for(int i=0;
i<(URL_withPacket.length()); i++){//using
a string this time, so use .length()
   payload[payload_size+i] =
URL_withPacket[i];//build up the payload
   counter++;//increment the counter
 }//for int
 payload_size =
counter+payload_size;//payload size is
just the counter value - more on this
when we need to build out the payload
with more data

 if(connect_ESP()){//this calls
'connect_ESP()' and expects a '1' back if
successful

 //Serial.println("connected ESP");
 if(read_until_ESP(keyword_t1,sizeof(key
word_t1),5000,0)){//go find t1 then stop
 if(read_until_ESP(keyword_doublehash,si
zeof(keyword_doublehash),5000,1)){//our
data is next, so change mode to '1' and
stop at ##
 //got our data, so we quickly stored it
away in d1
 for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)//go
see 'setup' and how this was done with
the ip address for more info
 t1_from_ESP[i] =
scratch_data_from_ESP[i];
 t1_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);

 //we still have more data to get out of
this stream, and now we want d1
  if(read_until_ESP(keyword_b1,sizeof(ke
yword_b1),5000,0)){//same as before -
first d1
```

```
  if(read_until_ESP(keyword_doublehash,s
izeof(keyword_doublehash),5000,1)){//now
## and mode=1
  for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)
  d1_from_ESP[i] =
scratch_data_from_ESP[i];
  d1_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);

  if(read_until_ESP(keyword_b2,sizeof(ke
yword_b2),5000,0)){//same as before -
first d1
  if(read_until_ESP(keyword_doublehash,s
izeof(keyword_doublehash),5000,1)){//now
## and mode=1
  for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)
  d2_from_ESP[i] =
scratch_data_from_ESP[i];
  d2_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);

  if(read_until_ESP(keyword_b3,sizeof(ke
yword_b3),5000,0)){//same as before -
first d1
  if(read_until_ESP(keyword_doublehash,s
izeof(keyword_doublehash),5000,1)){//now
## and mode=1
  for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)
  d3_from_ESP[i] =
scratch_data_from_ESP[i];
  d3_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);

  if(read_until_ESP(keyword_b4,sizeof(ke
yword_b4),5000,0)){//same as before -
first d1
  if(read_until_ESP(keyword_doublehash,s
izeof(keyword_doublehash),5000,1)){//now
## and mode=1
  for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)
  d4_from_ESP[i] =
scratch_data_from_ESP[i];
  d4_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);
```

```
   if(read_until_ESP(keyword_b5,sizeof(ke
yword_b5),5000,0)){//same as before -
first d1
   if(read_until_ESP(keyword_doublehash,s
izeof(keyword_doublehash),5000,1)){//now
## and mode=1
   for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)
   d5_from_ESP[i] =
scratch_data_from_ESP[i];
   d5_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);

   if(read_until_ESP(keyword_n1,sizeof(ke
yword_n1),5000,0)){//same as before -
first d1
   if(read_until_ESP(keyword_doublehash,s
izeof(keyword_doublehash),5000,1)){//now
## and mode=1
   for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)
   d9_from_ESP[i] =
scratch_data_from_ESP[i];
   d9_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);

   if(read_until_ESP(keyword_n2,sizeof(ke
yword_n2),5000,0)){//same as before -
first d1
   if(read_until_ESP(keyword_doublehash,s
izeof(keyword_doublehash),5000,1)){//now
## and mode=1
   for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)
   d10_from_ESP[i] =
scratch_data_from_ESP[i];
   d10_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);

   if(read_until_ESP(keyword_n3,sizeof(ke
yword_n3),5000,0)){//same as before -
first d1
   if(read_until_ESP(keyword_doublehash,s
izeof(keyword_doublehash),5000,1)){//now
## and mode=1
   for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)
   d11_from_ESP[i] =
scratch_data_from_ESP[i];
   d11_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);
```

```

   if(read_until_ESP(keyword_n4,sizeof(ke
yword_n4),5000,0)){//same as before -
first d1
   if(read_until_ESP(keyword_doublehash,s
izeof(keyword_doublehash),5000,1)){//now
## and mode=1
   for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)
   d12_from_ESP[i] =
scratch_data_from_ESP[i];
   d12_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);

   if(read_until_ESP(keyword_n5,sizeof(ke
yword_n5),5000,0)){//same as before -
first d1
   if(read_until_ESP(keyword_doublehash,s
izeof(keyword_doublehash),5000,1)){//now
## and mode=1
   for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)
   d13_from_ESP[i] =
scratch_data_from_ESP[i];
   d13_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);

   if(read_until_ESP(keyword_n6,sizeof(ke
yword_n6),5000,0)){//same as before -
first d1
   if(read_until_ESP(keyword_doublehash,s
izeof(keyword_doublehash),5000,1)){//now
## and mode=1
   for(int i=1;
i<=(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1); i++)
   d14_from_ESP[i] =
scratch_data_from_ESP[i];
   d14_from_ESP[0] =
(scratch_data_from_ESP[0]-
sizeof(keyword_doublehash)+1);
   Serial.println("FOUND DATA &
DISCONNECTED");
   serial_dump_ESP();//now we can clear
out the buffer and read whatever is still
there
   Serial.println("");
   Serial.print("Time ");
   if(t1_from_ESP[0] > 3){
   Serial.print(t1_from_ESP[1]);
   Serial.print(t1_from_ESP[2]);
   Serial.print(":");
   Serial.print(t1_from_ESP[3]);
   Serial.println(t1_from_ESP[4]);
   }
   else{
   Serial.print(t1_from_ESP[1]);
   Serial.print(":");
   Serial.print(t1_from_ESP[2]);
   Serial.println(t1_from_ESP[3]);
   }

   Serial.print("RECEIVED_BOOL_1 =
");//print out LED data and convert to
integer
   received_bool_1 = 0;
   for(int i=1; i<=d1_from_ESP[0]; i++){
   //Serial.print(d12_from_ESP[i]);
   received_bool_1 = received_bool_1 +
((d1_from_ESP[i] - 48) *
multiplier[d1_from_ESP[0] - i]);
   }
   Serial.println(received_bool_1);

   Serial.print("RECEIVED_BOOL_2 =
");//print out LED data and convert to
integer
   received_bool_2 = 0;
   for(int i=1; i<=d2_from_ESP[0]; i++){
   //Serial.print(d12_from_ESP[i]);
   received_bool_2 = received_bool_2 +
((d2_from_ESP[i] - 48) *
multiplier[d2_from_ESP[0] - i]);
   }
   Serial.println(received_bool_2);

   Serial.print("RECEIVED_BOOL_3 =
");//print out LED data and convert to
integer
   received_bool_3 = 0;
   for(int i=1; i<=d3_from_ESP[0]; i++){
   //Serial.print(d12_from_ESP[i]);
   received_bool_3 = received_bool_3 +
((d3_from_ESP[i] - 48) *
multiplier[d3_from_ESP[0] - i]);
   }
   Serial.println(received_bool_3);

   Serial.print("RECEIVED_BOOL_4 =
");//print out LED data and convert to
integer
   received_bool_4 = 0;
   for(int i=1; i<=d4_from_ESP[0]; i++){
   //Serial.print(d12_from_ESP[i]);
   received_bool_4 = received_bool_4 +
((d4_from_ESP[i] - 48) *
multiplier[d4_from_ESP[0] - i]);
   }
   Serial.println(received_bool_4);

   Serial.print("RECEIVED_BOOL_5 =
");//print out LED data and convert to
integer
   received_bool_5 = 0;
```

```
   for(int i=1; i<=d5_from_ESP[0]; i++){
   //Serial.print(d12_from_ESP[i]);
   received_bool_5 = received_bool_5 +
((d5_from_ESP[i] - 48) *
multiplier[d5_from_ESP[0] - i]);
   }
   Serial.println(received_bool_5);
  Serial.print("RECEIVED_NUMBER_1 =
");//print out LED data and convert to
integer
   received_nr_1 = 0;
   for(int i=1; i<=d9_from_ESP[0]; i++){
   //Serial.print(d12_from_ESP[i]);
   received_nr_1 = received_nr_1 +
((d9_from_ESP[i] - 48) *
multiplier[d9_from_ESP[0] - i]);
   }
   Serial.println(received_nr_1);

   Serial.print("RECEIVED_NUMBER_2 =
");//print out LED data and convert to
integer
   received_nr_2 = 0;
   for(int i=1; i<=d10_from_ESP[0]; i++){
   //Serial.print(d12_from_ESP[i]);
   received_nr_2 = received_nr_2 +
((d10_from_ESP[i] - 48) *
multiplier[d10_from_ESP[0] - i]);
   }
   Serial.println(received_nr_2);

   Serial.print("RECEIVED_NUMBER_3 =
");//print out LED data and convert to
integer
   received_nr_3 = 0;
   for(int i=1; i<=d11_from_ESP[0]; i++){
   //Serial.print(d12_from_ESP[i]);
   received_nr_3 = received_nr_3 +
((d11_from_ESP[i] - 48) *
multiplier[d11_from_ESP[0] - i]);
   }
   Serial.println(received_nr_3);

   Serial.print("RECEIVED_NUMBER_4 =
");//print out LED data and convert to
integer
   received_nr_4 = 0;
   for(int i=1; i<=d12_from_ESP[0]; i++){
   //Serial.print(d12_from_ESP[i]);
   received_nr_4 = received_nr_4 +
((d12_from_ESP[i] - 48) *
multiplier[d12_from_ESP[0] - i]);
   }
   Serial.println(received_nr_4);

   Serial.print("RECEIVED_NUMBER_5 =
");//print out LED data and convert to
integer
   received_nr_5 = 0;
   for(int i=1; i<=d13_from_ESP[0]; i++){
   //Serial.print(d12_from_ESP[i]);
   received_nr_5 = received_nr_5 +
((d13_from_ESP[i] - 48) *
multiplier[d13_from_ESP[0] - i]);
   }
   Serial.println(received_nr_5);

   Serial.print("RECEIVED_TEXT_1 =
");//print out LED data and convert to
integer
   received_text = "";
   for(int i=1; i<=d14_from_ESP[0]; i++){
   //Serial.print(d12_from_ESP[i]);
   received_text = received_text +
d14_from_ESP[i];
   }
   Serial.println(received_text);

   Serial.println("");
   //that's it!!
   }//##
   }//n6
   }//##
   }//n5
   }//##
   }//n4
   }//##
   }//n3
   }//##
   }//n2
   }//##
   }//n1
   }//##
   }//b5
   }//##
   }//b4
   }//##
   }//b3
   }//##
   }//b2
   }//##
   }//b1
   }//##
   }//t1


  }//connect ESP


}//connect web host
```

3. RESULTS AND DISCUSSION:

3.1. Results

Several scenarios are used to test the smart farm monitoring system. The soil is tested for moisture under different climatic situations using a soil moisture sensor, and the results are properly interpreted. Specific Arduino code is written and imported into the microcontroller for sensing soil moisture, pH, and temperature. The wireless transmission is accomplished using Wi-Fi. The output values of the soil moisture sensor only depend on the soil's resistivity. The sensor's value at the start of a wet condition is 0. When this occurs, the motor pump turns off, and the measured value is communicated to the microcontroller through NodeMCU. The microprocessor activates the relay, and the motor turns on when the sensed value from the sensor surpasses the threshold value. The motor pump automatically turns ON and OFF when plants receive enough water. Below the given tables, Table 1, Table 2, Table 3, and Table 4, show the test result we obtained after a successful system test. Here the temperature, humidity, and soil moisture are measured in terms of degrees Celsius, and if the object is detected or not by the PIR sensor, the on/off status of the water motor and the buzzer will be displayed on the mobile screen of the user. To display content on the mobile application, the device first goes online after being turned on. After starting up quickly (within 10 to 15 seconds), it immediately begins sending data continually to mobile devices. After a system test was completed successfully, the test results are shown in the tables below. Here, the user's mobile device will display the temperature in degrees Celsius, humidity, and soil moisture in percentages, the object's PIR sensor's ability to detect it or not, the water motor's on/off status, and the buzzer's state.

***Table 1**. Recorded Data from temperature Sensor*

Time of Reading (Approx.)	Temperature (Average)	Water Pump
8:00 AM	16.7 C	NO
9:00 AM	17.8 C	NO
10:00 AM	18.4 C	NO
11:00 AM	19 C	NO
12:00 PM	20.5 C	YES
1:00 PM	20.8 C	YES
2:00 PM	20.6 C	YES
3:00 PM	20 C	YES
4:00 PM	18.2 C	NO
5:00 PM	16.5 C	NO

***Table 2**. Recorded Data from Humidity Sensor*

Time ofof Reading (Approx.)	Temperature (Average)	Water Pump
8:00 AM	92 C	NO
9:00 AM	90 c	NO
10:00 AM	84 C	NO
11:00 AM	60 C	NO
12:00 PM	55 C	YES
1:00 PM	54 C	YES
2:00 PM	50 C	YES
3:00 PM	72 C	NO
4:00 PM	68 C	NO
5:00 PM	80 C	NO

***Table 3**. Recorded Data from Moisture Sensor*

Time of Reading (Approx.)	Temperature (Average)	Water Pump
8:00 AM	80 C	NO
9:00 AM	75 C	NO
10:00 AM	68 C	NO
11:00 AM	56 C	NO
12:00 PM	50 C	YES
1:00 PM	46 C	YES
2:00 PM	48 C	YES
3:00 PM	44 C	YES
4:00 PM	60 C	NO
5:00 PM	78 C	NO

3.2. Discussion

Figure 8 shows that as soon as the temperature threshold of 19.5’C is achieved, the water pump starts automatically to water the plants. Normally during the day time when the temperature is very high.

According to Figure 9, it has been observed that as humidity increases, it is not necessary to water the plants as the plants take up water vapor from the surface of the leaves under humid conditions.

From the results of Table 3 depicted in Figure 10, we conclude that with the decrease in moisture of the soil, the plants need to be watered.

After carefully examining the data from the experiment, it was concluded that the plants require watering when the temperature is high, the humidity is low, and the moisture content is low. The overall outcome of the data collected in January 2023 is displayed in Figure 11.

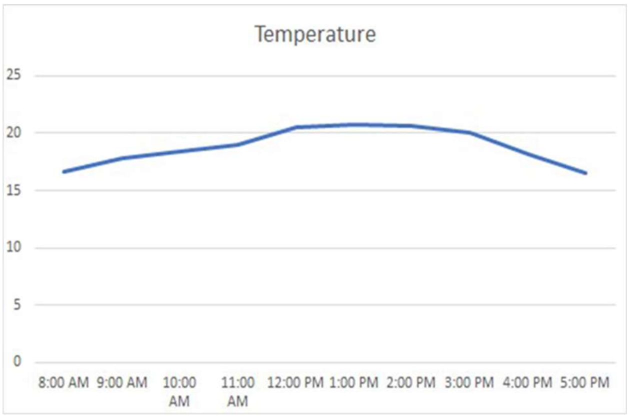

***Figure 8**. Graphical representation of Table 1*

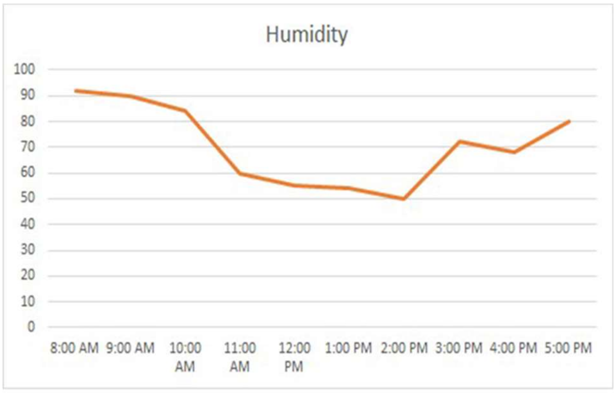

***Figure 9.** Graphical representation of Table 2*

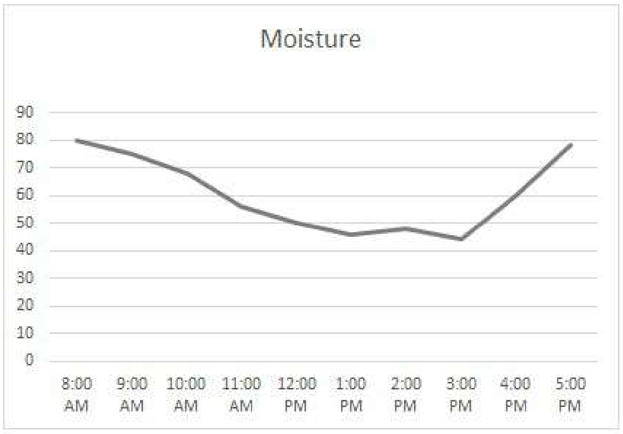

***Figure 10**. Graphical representation of Table 3*

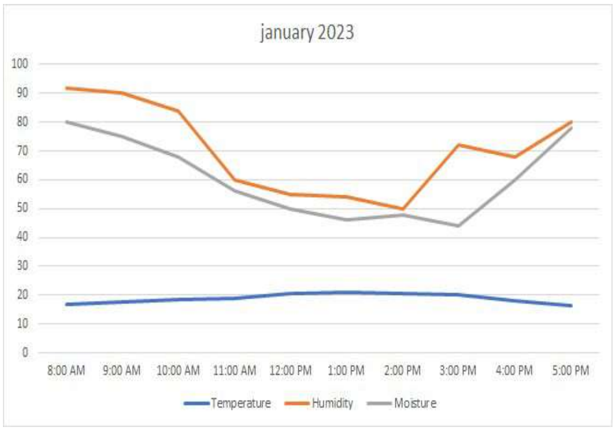

Figure 11. *Temperature vs. Humidity vs. Moisture*

Intrusion in the fields was seen mostly during the morning hours and mid-day time. It was noticed from Table 4 that the intrusion was from birds and small animals that came near the plants to eat insects and water.

Table 4*. Recorded Data from PIR Sensor*

Time of Reading (Approx.)	Temperature (Average)	PIR	Buzzer
08:00:00	16.7 C	NOTHING DETECTED	OFF
09:00:00	17.8 C	NOTHING DETECTED	OFF
10:00:00	18.4 C	INTRUSION DETECTED	ON
11:00:00	19 C	NOTHING DETECTED	OF
12:00:00	20.5 C	INTRUSION DETECTED	ON
13:00:00	20.8 C	INTRUSION DETECTED	ON
14:00:00	20.6 C	INTRUSION DETECTED	ON
15:00:00	20 C	INTRUSION DETECTED	ON
16:00:00c	18.2 C	NOTHING DETECTED	OF
17:00:00	16.5 C	NOTHING DETECTED	OF

The final prototype of the system is shown in Figure 12 below.

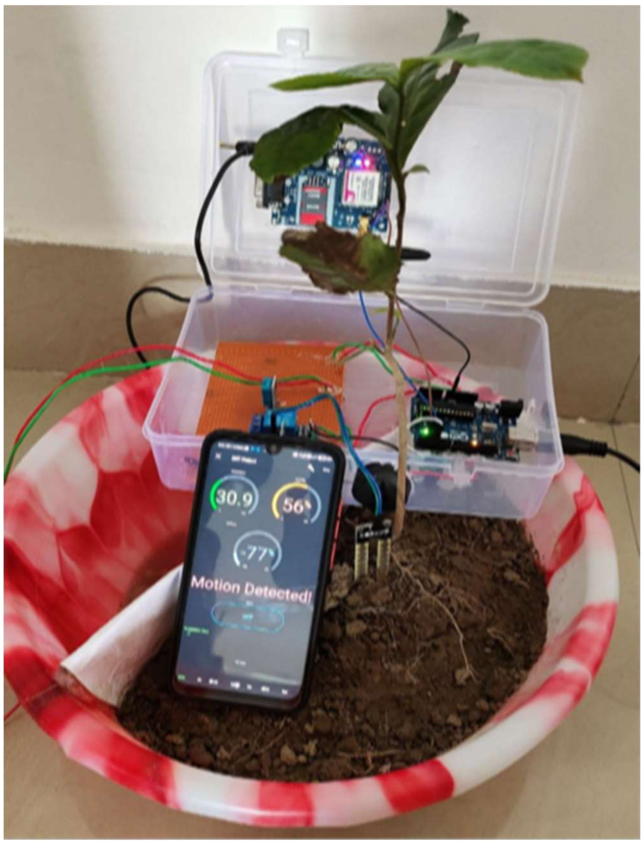

Figure 12. *Prototype of the system*

4. CONCLUSIONS:

Implementing the IoT-based agriculture environment and security monitoring system presented in this paper potentially fulfill the need for an intelligent agriculture monitoring system. The proposed work yielded significant advantages and positive outcomes and has demonstrated remarkable efficiency in optimizing water usage. On average, about 70% of water is saved when it comes to watering the plants. Also, the rate of human intervention is reduced to about 40%. By automatically monitoring and analyzing environmental data such as humidity, temperature, and soil moisture levels, the system enables farmers to implement precise and targeted irrigation practices, minimizing water wastage and reducing overall production costs, thus yielding production.

The ability to automatically record the current condition of the agricultural ground using the ESP32 camera provides additional visual evidence and monitoring capabilities, allowing farmers to respond promptly to any anomalies or threats. Furthermore, integrating a motion detector into the system has proven valuable in detecting vulnerability from harmful animals. This feature leaps security measures, mitigates the risk of crop damage, and helps safeguard agricultural assets effectively.

Overall, the proposed IoT-based system serves as a powerful tool to bridge existing gaps in agricultural practices. By enabling real-time environmental monitoring, precise irrigation management, and enhanced security measures, it significantly contributes to increasing crop yield while reducing production costs. The system empowers farmers to make data-driven decisions, optimize resource utilization, and achieve sustainable and efficient agricultural operations.

5. DECLARATIONS

5.1. Study Limitations

The study is limited to the experimental conditions.

5.2. Acknowledgements

I would like to extend my heartfelt appreciation to all the individuals who have provided support throughout the research process. Although no other individuals were directly involved in this project, their indirect contributions and invaluable support have played a significant role in its success. I would also like to extend my appreciation to the reviewers and editors who have provided constructive feedback and suggestions to improve the quality of this research paper.

5.3. Funding source

This research was funded by the authors.

5.4. Competing Interests

The authors of this research paper declared that there are no conflicts of interest that could influence the objectivity or integrity of this study.

5.5. Open Access

6. REFERENCES:

1. Chen, C.-H., Wu, Y.-C., Zhang, J.-X., & Chen, Y.-H. (2022). IoT-Based Fish Farm Water Quality Monitoring System. *Sensors*, *22*(17), 6700. Doi: https://doi.org/10.3390/s22176700

2. Ferrag, M. A., Shu, L., Yang, X., Derhab, A., & Maglaras, L. (2020). Security and privacy for green IoT-based agriculture: Review, blockchain solutions, and challenges. *IEEE Access,8*,32031–32053. DOI: 10.1109/ACCESS.2020.2973178

3. Kaur, G., Upadhyaya, P., & Chawla, P. (2023). Comparative analysis of IoT-based controlled environment and uncontrolled environment plant growth monitoring system for hydroponic indoor vertical farm. *Environmental Research*, *222*,115313.Doi:https://doi.org/10.1016/j.envres.2023.115313

4. Kim, W.-S., Lee, W.-S., & Kim, Y.-J. (2020). A review of the applications of the internet of things (IoT) for agricultural automation. *Journal of Biosystems Engineering*, *45*, 385–400. DOI: 10.1007/s42853-020-00078-3

5. Kumar, R., & Rajasekaran, M. P. (2016). An IoT based patient monitoring system using raspberry Pi. *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*,1–4. DOI: 10.1109/ICCTIDE.2016.7725378

6. Kumar, S., Tiwari, P., & Zymbler, M. (2019). Internet of Things is a revolutionary approach for future technology enhancement: A review. *Journal of Big Data*, *6*(1), 1–21. https://doi.org/10.1186/s40537-019-0268-2

7. Rabka, M., Mariyanayagam, D., & Shukla, P. (2022). IoT-Based Horticulture Monitoring System. *Intelligent Sustainable Systems: Selected Papers of WorldS4*

2021, Volume 2, 765–774. https://doi.org/10.1007/978-981-16-6369-7_68

8. Singh, R., Srivastava, S., & Mishra, R. (2020). Ai and iot based monitoring system for increasing the yield in crop production. *2020 International Conference on Electrical and Electronics Engineering (ICE3)*, 301–305. DOI: 10.1109/ICE348803.2020.9122894

9. Singh, T. A., & Chandra, J. (2018). IOT Based Green House Monitoring System. *J. Comput. Sci.*, *14*(5), 639–644. DOI: 10.3844/jcssp.2018.639.644

10. Suma, V. (2021). Internet-of-Things (IoT) based smart agriculture in India-an overview. *Journal of ISMAC*, *3*(01), 1–15. DOI: https://doi.org/10.36548/jismac.2021.1.001

11. ul Islam, A., Borkakoty, S., Kalita, D., Sarma, P., Sharmah, D., & Islam, M. (2022). DESIGNING WEB INTERFACE AND LPG SAFETY KIT AS AN ARDUINO BASED LPG MONITORING & ALERT SYSTEM. *Journal of East China University of Science and Technology*, *65*(3),20–25. DOI:10.5281/ZENODO.6805856.

12. Baranwal, T., & Pateriya, P. K. (2016). Development of IoT based smart security and monitoring devices for agriculture. *2016 6th International Conference-Cloud System and Big Data Engineering (Confluence)*, 597–602.

13. Collado, E., Valdés, E., García, A., & Sáez, Y. (2021). Design and implementation of a low-cost IoT-based agroclimatic monitoring system for greenhouses. *AIMS Electronics and Electrical Engineering*, *5*(4), 251–283. doi: 10.3934/electreng.2021014

14. Hernández-Morales, C. A., Luna-Rivera, J. M., & Perez-Jimenez, R. (2022). Design and deployment of a practical IoT-based monitoring system for protected cultivations. *Computer Communications*, *186*, 51–64. https://doi.org/10.1016/j.comcom.2022.01.009

15. Kamienski, C., Soininen, J.-P., Taumberger, M., Dantas, R., Toscano, A., Salmon Cinotti, T., Filev Maia, R., & Torre Neto, A. (2019). Smart Water Management Platform: IoT-Based Precision Irrigation for Agriculture. *Sensors*, *19*(2), Article 2. https://doi.org/10.3390/s19020276

16. Muangprathub, J., Boonnam, N., Kajornkasirat, S., Lekbangpong, N., Wanichsombat, A., & Nillaor, P. (2019). IoT and agriculture data analysis for smart farm. *Computers and Electronics in Agriculture*, *156*, 467–474. https://doi.org/10.1016/j.compag.2018.12.011

17. Ramaprasad, S. S., Kumar, B. S., Lebaka, S., Prasad, P. R., Kumar, K. S., & Manohar, G. N. (2019). Intelligent Crop Monitoring and Protection System in Agricultural fields Using IoT. *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, 1527–1531.