



Faculty of Engineering
Cairo University

Cairo University
Faculty of Engineering
Electronics and Electrical Communications Engineering Department

Precision Agriculture

Fourth Year
Graduation Project

Submitted by:

Omar Osama Shalaby
Rania Fahmy Abbas
Ahmed Samir Mohamed
Mohamed Ashraf Abd El-Hameed
Nada Osama Hosny
Mariam Saad Hassan

Submitted to:

Dr. Tamer El-Batt
Dr. Hassan Mostafa

Acknowledgment

This project would not have been possible without the support and guidance of our advisor, Dr. Hassan Mostafa, Dr. Tamer El-Batt and specially Eng. Mahmoud Mahgoub whom we owe a lot of thanks to him because of his efforts and time throughout the whole year.

Thanks for our last year colleagues or other external engineers. It was a great honor to work under their supervision and with their guidance along the way.

Thank you to Dr. Amr Talaat and Eng. Ahmed Tag El-Din for helping us with their ideas and efforts.

And of course we are very grateful for our families and friends for supporting us all over this tough year.

Table of Contents

Acknowledgment	2
Table of Figures	6
Chapter 1: Introduction	8
1.1 Overview	8
1.2 Goals of the research	8
1.3 Previous Work.....	9
1.4 Structure of the research	10
Chapter 2: Architecture	11
2.1 Layers Explanation	12
2.2 Block Diagram of the System	13
Chapter 3: Main Components	14
Sensors.....	14
3.1 Temperature and Humidity Sensor.....	15
3.1.1 Features	16
3.1.2 Specifications	16
3.1.3 Communication Process.....	17
3.1.4 Connection with NodeMCU.....	19
3.2 Soil Moisture Sensor	20
3.2.1 Specifications	20
3.2.2 Sensing Technique.....	20
3.2.3 Connection with NodeMCU.....	21
3.3 Code Snippets.....	22
Chapter 4: Network	23
4.1 Wireless Sensor Network (WSN)	23
4.1.1 Advantages	23
4.1.2 Design Requirements	24
4.1.3 Deployment Topologies.....	25
4.2 Communication Techniques	26
4.2.1 First Topology	26

4.2.2 Second Topology	28
4.3 Nodes distribution	29
4.4 Network Components	30
4.5 Architecture.....	30
4.6 Timeline	30
4.7 Gateway	31
4.7.1 GPRS	31
4.7.2 Mi-Fi	32
4.8 Code Snippets.....	34
Chapter 5: Power Analysis	36
5.1 Operating Currents	36
5.2 Theoretical Analysis	38
5.3 Practical Analysis	40
5.3.1 Instrumentation Amplifier in Power measurements	41
Chapter 6: Packaging.....	43
6.1 Packaging Phases	43
6.2 Why Packaging?.....	43
6.3 Choosing the material.....	43
Chapter 7: Webserver and Application	45
7.1 Definition	45
7.2 Code Snippets.....	47
Chapter 8: Machine Learning	48
8.1 General Approach.....	48
8.2 Machine Learning.....	48
8.2.1 Methods of Machine Learning	48
8.3 Analysis	55
8.3.1 Dataset 1: Potato Blight	56
8.3.2 Dataset 2: Cotton Leaf Worm	61
8.4 Machine Learning Analysis	63
8.4.1 Analysis on Potato Blight Dataset	64
8.4.2 Analysis on Cotton Leaf Worm Dataset	65

8.5 Results.....	66
8.6 ML Algorithms Simulator	70
8.6.1 GUI Simulator	70
8.6.2 GUI Programming in Python	70
8.6.3 Application.....	71
8.7 Code Snippets.....	72
Chapter 9: Future Work.....	73
9.1 On-Site Deployment	73
9.2 Line to Line Irrigation	73
9.3 Water Pump Actuation	74
9.4 Disease Detection with ML	74
9.5 Developing Algorithm Application.....	75
9.6 Access point	75
9.6.1 Mi-Fi	75
9.6.2 GPRS Module.....	75
Chapter 10: References.....	76

Table of Figures

Figure 1: Network Layers.....	11
Figure 2: Layers Explanation.....	12
Figure 3: Block Diagram of the System.....	13
Figure 4: Comparison of Temp Sensors.....	15
Figure 5: DHT22.....	16
Figure 6: Overall Communication Process of DHT22 Sensor	17
Figure 7: DHT22 Pulling down the data	18
Figure 8: Data Pin '0'	18
Figure 9: Data Pin '1'	19
Figure 10: Connection with NodeMCU	19
Figure 11: Soil Moisture Sensor (YL-69)	20
Figure 12: YL-69 Working Analogy	21
Figure 13: Connection of YL-69 with NodeMCU.....	21
Figure 14: Code Snippets for sensors interface with NodeMCU	22
Figure 15: Randomly Deployed WSN	25
Figure 16: Regular deployed WSN.....	25
Figure 17: First Topology Structure.....	26
Figure 18: Second Topology Structure	28
Figure 19: Time Division	31
Figure 20: GPRS Module	31
Figure 21: Mi-Fi.....	32
Figure 22: Client Node Snippets.....	34
Figure 23: Server Node Snippets	35
Figure 24: NodeMCU Sleep Modes.....	37
Figure 25: Instrumentation Amplifier using Op-Amps.....	41
Figure 26: Simulation using Instrumentation Amplifier	42
Figure 27: Package Design.....	44
Figure 28: Webserver Code.....	47
Figure 29: Machine Learning Algorithms	50

Figure 30: Logistic Regression example results	51
Figure 31: SVM example results	52
Figure 32: Original State of Potato Light Dataset	57
Figure 33: Disease Severity over Time	58
Figure 34: Accumulated Rain over Time	58
Figure 35: Tmax & Tmin over Time	58
Figure 36: Additional calculated Features for Potato Blight Dataset	59
Figure 37: Infection Period over Time in days	60
Figure 38: Additional Calculated Features for Cotton Leaf Worm Dataset.....	61
Figure 39: Disease Severity over Time	62
Figure 40: Relative Humidity over Time	62
Figure 41: Tmax and Tmin over Time	62
Figure 42: Analysis on Potato Blight without Previous Disease Severity	64
Figure 43: Analysis on Potato Blight with Previous Disease Severity	65
Figure 44: Analysis on Cotton Leaf Worn Dataset with Random Sampling	65
Figure 45: Results of applying Algorithms on Dummy Data.....	66
Figure 46: Results of applying Algorithms on Cotton Leaf Worm Dataset.....	67
Figure 47: Predicted vs Actual for Cotton Leaf Worm Dataset	68
Figure 48: Results of applying Algorithms on Potato Blight Dataset.....	68
Figure 49: Predicted vs Actual for Potato Blight Dataset	69
Figure 50: GUI Front-End	71
Figure 51: GUI Code Snippets	72
Figure 52: Half to Half Irrigation System	73
Figure 53: Line to Line Irrigation	74
Figure 54: Water Pump	74

Chapter 1: Introduction

1.1 Overview

Precision agriculture is a rising management method for modern farms, aiming to gather and analyze data about different crops, whether it is temporal data, spatial data, or data concerning the well being of crops. This data is then processed and analyzed to help produce management decisions concerning the farm, such as the amount of water needed for each crop, or the right percentage of humidity needed for the crop to grow in a healthy environment.

It is not surprising to see how technology is increasingly adopted in farms, as farming is a labor-intensive activity and is essential for the food industry. Thus, farmers have been seeking modern ways to efficiently develop more crops with manageable costs, in shorter time, and with more accurate and controlled outputs than these produced without use of technology.

Using wireless sensor network technology with means of machine learning and app development, it is very achievable to produce a system that can remotely monitor crops, control the amount of water being pumped to crops, predict the right temperature to grow a certain crop, and last but not least, predict the occurrence of common crop diseases and decrease the probability of this occurrence.

1.2 Goals of the research

The basic goal of the conducted research is to find the most efficient, power saving, cheapest ways to create a monitoring system for a farm. In addition to using means of machine learning techniques and models to get better results and better fits for the data.

This piece of work is mainly concerned with and focused on developing a monitoring system for a certain crop, and that is grapes. Means of measuring temperature, soil moisture, and humidity are discussed, as well as means to build a network of sensors, means to remotely control the collected data, and the theoretical results of certain applied machine learning models. The collected data from sensors is then uploaded to a custom web server through the ESP 8266 Wi-Fi module.

It is also important to note that grapes industry in Egypt has been growing rapidly since the year 2001. It is statistically known that grapes are one of the most widely grown fruits, second only to citrus. Another interesting fact is that Egypt ranks fourth worldwide in the global production of table grapes. In 2016 only, the production of grapes in Egypt reached 1,691,194 tons. Moreover, Egypt's 2015 exports of grapes have increased 18 times more than these in 2001, starting from 46,000 tons in 2001 to 167,000 tons in 2015. Based on the market statistics, grape is a crucial crop in the industry, thus, it is a clever move to enhance its production even more with technology. A step that will bring more income, and ensure greater quality.

1.3 Previous Work

A 2019 research was conducted on the same topic. The results of this research have been a great help, and an inspiration to build on.

The 2019 project research paper presented a similar system to the one discussed in this research. Consisting of 3 main components: a local node, gateway, and webserver. The local node consists of a Node MCU, with environment-measuring sensors connected to it. These sensors measure each of temperature, humidity, and soil moisture. The readings of the sensors are then sent to the gateway through the wifi using the esp8266 module. As for the gateway, it consists of a raspberry pi and a GSM modem. The raspberry pi receives the new readings coming from the sensors, stores them in a database, then sends them to a webserver using the GSM module.

Forming an integration between IoT and machine learning, the 2019 project aimed to prevent the happening of late blight in potatoes. Certain machine learning models, like the regression model, were used to predict the probability of late blight infection in the potato yield. Based on this, the system will send an advice to the user(s) in order to take action; such as using pesticides or alike agricultural actions.

This project builds on the work done in 2019 in many ways. The data collected was unfortunately not accurate enough to build a disease prediction system, thus it was decided to make a monitoring system to harvest more accurate data, and so use the data later in future work to deploy machine learning models, and make predictions based on them. It is to be noted that last year's project was applied on potatoes planted in greenhouses, while this project, while

addressed for grapes, is generic for a wide range of crops. It is also accustomed to work with wide areas of land in open air, and not only greenhouses.

1.4 Structure of the research

This piece of work is divided into several sections, such that it covers every aspect in the research. In chapter 2, the architecture of the system will be discussed, as in the main blocks that define the system, and how they are connected together. In chapter 3, the sensors used in the system will be discussed thoroughly, as in each sensor's function, how much power they consume, how they are integrated in the system forming nodes, and other specifications. In chapter 4, the network architecture is presented, as in the way nodes are distributed on the land to achieve the best communication between them, in addition to the process in which data is transferred between different communication nodes. As for chapter 5, it discusses how much each component in the system consumes power, and how much power the system as a whole consumes. It also discusses the best ways to save power, such as putting certain components to sleep mode when they are not required to function. In chapter 6, means of packaging for nodes are discussed. Electronic components need to be protected from water and erosion factors, that is why packaging was necessary to the system nodes, in order to be sustainable in open air areas. In chapter 7, machine learning theoretical results are discussed. As for chapter 8, web server and application are illustrated, how data is transferred through the WIFI module into the database established on the custom web server. Moreover, the means of creating this server and linking it between the ESP 8266 and the database are also discussed.

Chapter 2: Architecture

The addressed irrigation system is an IOT monitoring system, it is developed to be a monitoring, and a water supply controlling system to the field. It aims to maintain water supply and avoid water shortage problem.

The system monitors the needed conditions that will affect the agriculture during different seasons of the year, such as, surrounding air temperature, air humidity and soil moisture. Then the average of these field measurements is calculated, stored in a database and uploaded to the webserver to be accessed, and observed by the farmer remotely.

The architecture of this irrigation system is divided into three main layers which are: Local node, Gateway and Webserver.

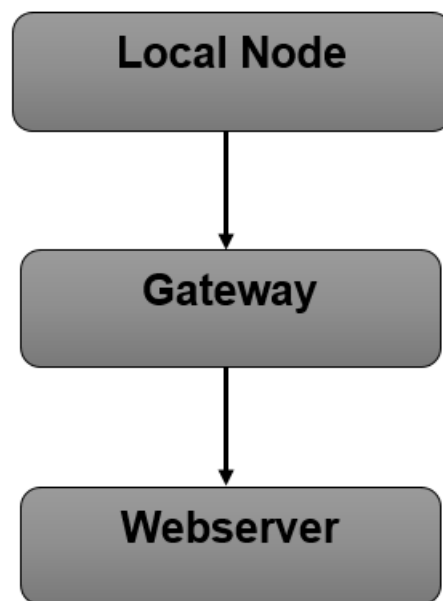


Figure 1: Network Layers

2.1 Layers Explanation

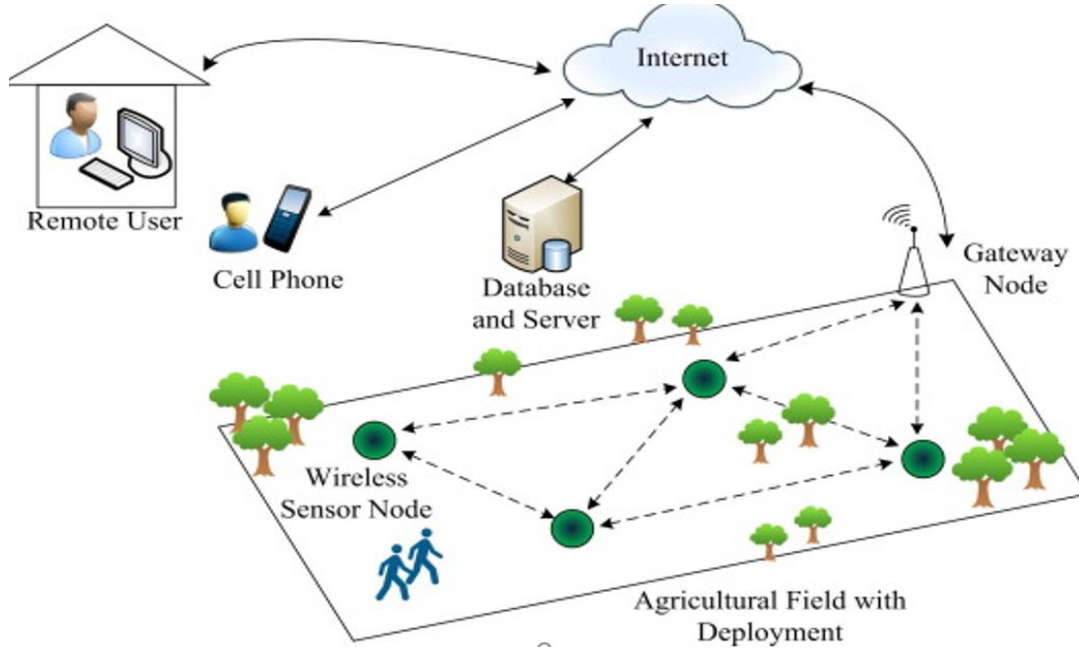


Figure 2: Layers Explanation

● Local Node

It consists of:

- **Node MCU:** is an open source IoT platform with the ESP8266 wi-fi module embedded on it. It includes firmware that runs on the ESP8266.
- **Sensors:** There are two sensors connected to Node MCU responsible for measuring the environmental conditions, which are environment temperature, humidity and soil moisture.
- **Battery:** It supplies the Node MCU with the power needed to work efficiently.

● Gateway

It consists of:

- Arduino UNO, SIM card, GPRS module and battery.
- GPRS module is used as a router to establish communication between mobile device or computer and the GPRS system.

- Arduino's output voltage is 5v which is the voltage needed to power the GPRS module.

● Webservice

It is used to display the readings of the sensors, which are uploaded by the Node MCU to be stored in a database server, and then to be accessed by the user anytime.

2.2 Block Diagram of the System

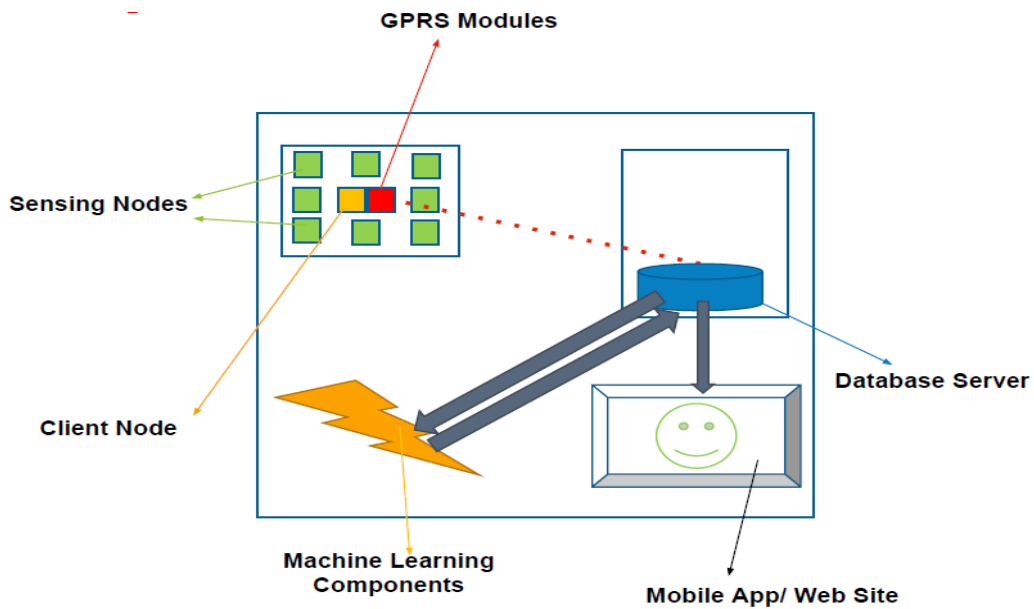
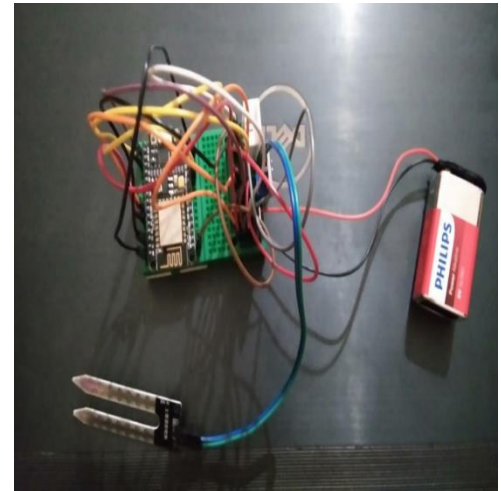


Figure 3: Block Diagram of the System

Chapter 3: Main Components

The system's most important components are the sensors and the Node MCU, as sensors are responsible for measuring the environmental conditions, and Node MCU responsible for receiving and processing these measurements to be sent after that to the database server.



Sensors

A sensor is a device, module, machine, or subsystem whose purpose is to detect events or changes in its environment, and send the information to other electronics, frequently a microprocessor. It converts signals from one energy domain to electrical domain.

There are several classifications of sensors made by different authors and experts:

In the first classification of the sensors, they are divided into Active and Passive. Active Sensors are those which require an external excitation signal or a power signal, while passive sensors do not require any external power signal and directly generate output response.

The second classification is based on the means of detection used in the sensor. Some of the means of detection are Electric, Biological, Chemical, Radioactive...etc.

The third classification is based on conversion phenomenon, which are the input and the output. Some of the common conversion phenomena are Photoelectric, Thermoelectric, Electrochemical, Electromagnetic, Thermo-optic...etc.

The final classification of the sensors are Analog and Digital sensors. Analog sensors produce an analog output which is a continuous output signal with respect to the quantity being measured, while digital sensors, in contrast to analog sensors, work with discrete or digital data. The data in digital sensors, which is used for conversion and transmission, is digital in nature.

There are two sensors that have been used in this system to collect data from the surrounding environment and soil, which are temperature and humidity sensor “DHT22” and soil moisture sensor “YL-69”.

3.1 Temperature and Humidity Sensor



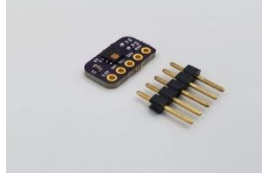
Point of comparison	DHT11	DHT22	HDC2080
Temperature Range	0°C to 50°C	-40°C to 80°C	-40°C to 125°C
Humidity Range	20% to 90%RH	0% to 100%RH	0% to 100%RH
Accuracy	±2°C and ±5%RH	+/- 0.5°C and ±2%RH	±0.2°C and ±2%RH
sampling rate	1 Hz (once every second)	0.5 Hz (once every 2 seconds)	1 Hz (once every second)
Operating Voltage	3 V to 5V	3 V to 5V	1.62 V to 3.6 V
Operating current	2.5 mA	2.5mA	3 mA
Price	80-85	120	200
Shape			

Figure 4: Comparison of Temp Sensors

In our wireless network, we decided to use DHT22 sensor mainly because in our agricultural field may the temperature in some cold days becomes less than 0°C.

DHT22 sensor is a basic, low-cost digital temperature and humidity sensor. It's fairly simple to use, but requires careful timing to grab data. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air.

This sensor features an output calibrated digital signal. It applies exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability. Its sensing elements is connected with 8-bit single-chip computer.

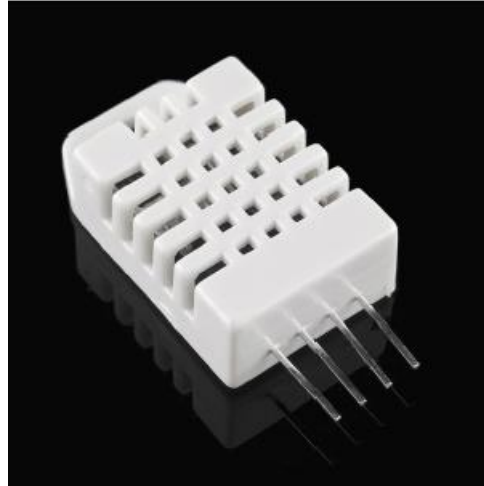


Figure 5: DHT22

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of program in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

The only real downside of this sensor is you can only get new data from it once every 2 seconds, so its readings can be up to 2 seconds old.

3.1.1 Features

- Low power consumption
- Long transmission distance, up to 100 meters
- 4 pins packaged and fully interchangeable
- Relative humidity and temperature measurement
- Extra components not needed
- Outstanding long-term stability

3.1.2 Specifications

- Operating voltage: 3v to 5v
- Operating current: 2.5 mA
- Sampling rate: 0.5 Hz (once every 2 seconds)

- Temperature Range: -40°C to 80°C
- Humidity Range: 0% to 100%RH
- Accuracy: +/- 0.5°C and ±2%RH

3.1.3 Communication Process

Single-bus data is used for communication between Node MCU and DHT22, it takes 5mS for single time communication. Data is comprised of integral part and decimal part.

The following is the formula for data, DHT22 send out higher data bit first.

DATA = 8 bits integral RH data + 8 bits decimal RH data + 8 bits integral Temperature data + 8 bits decimal Temperature data + 8 bits check-sum.

If the data transmission is right, check-sum should be the last 8 bits of "8 bits integral RH data + 8 bits decimal RH data + 8 bits integral Temperature data + 8 bits decimal Temperature data".

When Node MCU sends start signal, DHT22 changes from low-power-consumption-mode to running-mode. When Node MCU finishes sending the start signal, DHT22 will send response signal of 40-bit data that reflects the relative humidity and temperature information to Node MCU. Without starting signal from Node MCU, DHT22 will not give response signal to Node MCU. One start signal for one time's response data that reflect the relative humidity and temperature information from DHT22. DHT22 will change to low-power-consumption-mode when data collecting finished if it doesn't receive start signal from the Node MCU again.

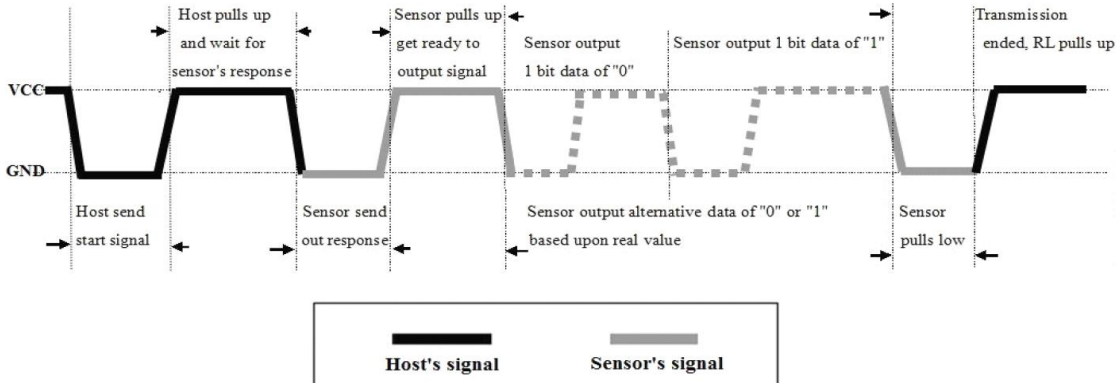


Figure 6: Overall Communication Process of DHT22 Sensor

Steps:

Step 1: Node MCU sends out start signal to DHT22 and DHT22 sends response signal to Node MCU.

Data-bus's free status is high voltage level. When communication between Node MCU and DHT22 begins, Node MCU will pull low data-bus and this process must beyond at least 1~10ms to ensure DHT22 could detect the Node MCU's signal, then

Node MCU will pulls up and wait 20-40us for DHT22's response.

When DHT22 detects the start signal, DHT22 will pull low the bus 80us as response signal, then DHT22 pulls up 80us for preparation to send data.

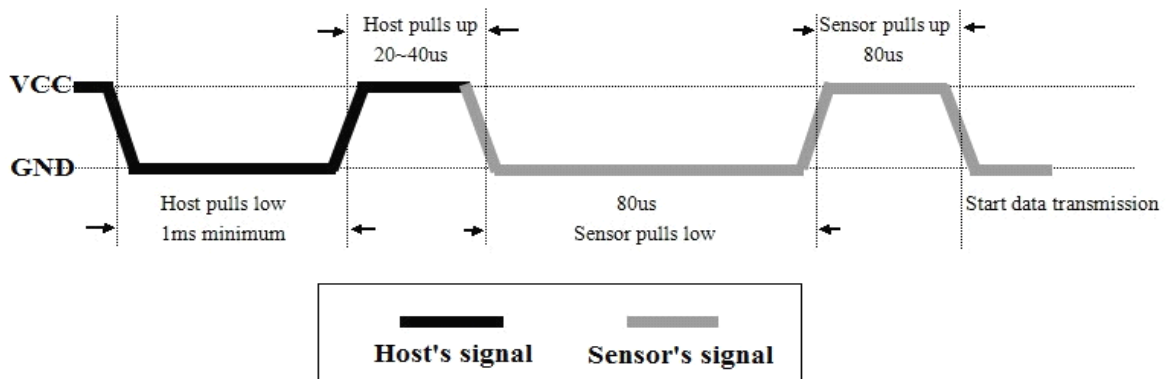


Figure 7: DHT22 Pulling down the data

Step 2: When DHT22 is sending data to MCU, every bit's transmission begins with low-voltage-level that last 50us, the following high-voltage-level signal's length decide the bit is "1" or "0".

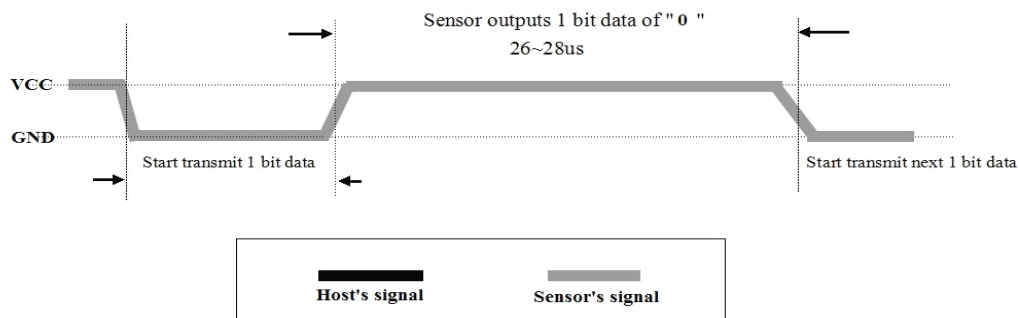


Figure 8: Data Pin '0'

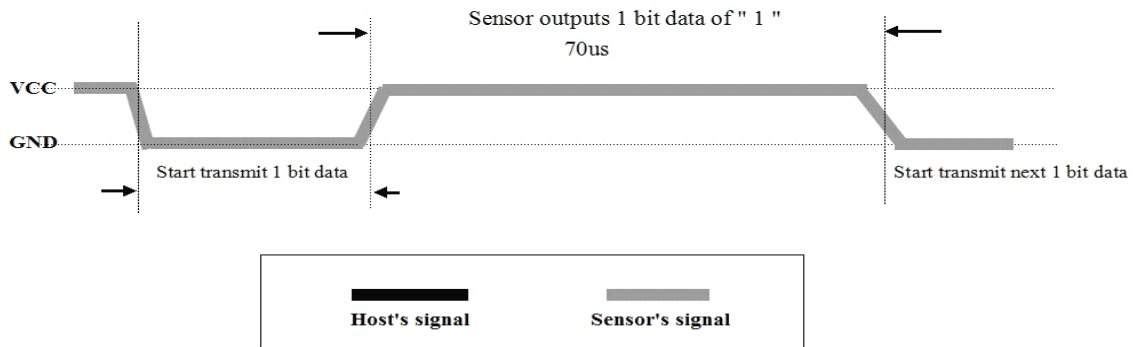


Figure 9: Data Pin '1'

3.1.4 Connection with NodeMCU

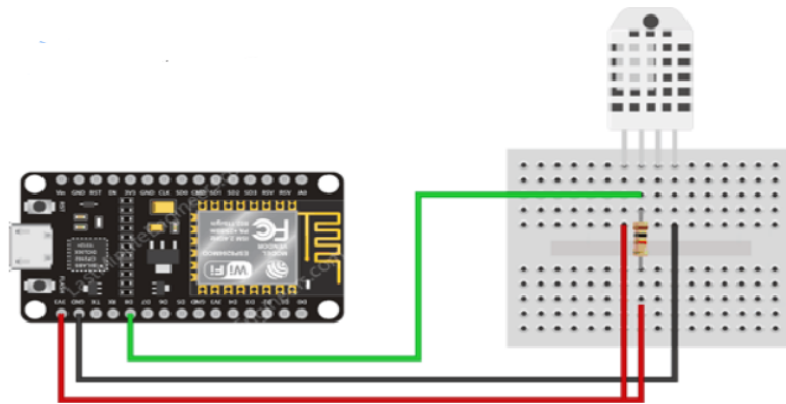


Figure 10: Connection with NodeMCU

Connecting DHT22 sensor to ESP8266 Node MCU is very simple as the DHT22 sensor has 4 pins (from left to right):

Pin 1: is power supply pin, connected to Node MCU 3.3v pin.

Pin 2: data output pin, connected to Node MCU digital pin 5 (D8).

Pin 3: not connected pin.

Pin 4: GND (ground), connected to Node MCU GND pin.

3.2 Soil Moisture Sensor

The soil moisture sensor is usually used to detect the humidity of the soil, it can read the amount of moisture present in the soil surrounding it, so it allows monitoring the water content in the soil.

The sensor has a built-in potentiometer for sensitivity adjustment of the digital output (D0), a power LED and a digital output LED.

It uses Immersion Gold which protects the nickel from oxidation. Electroless nickel immersion gold (ENIG) has several advantages over more conventional (and cheaper) surface plating such as HASL (solder), including excellent surface planarity (particularly helpful for PCB's with large BGA packages), good oxidation resistance, and usability for untreated contact surfaces such as membrane switches and contact points.

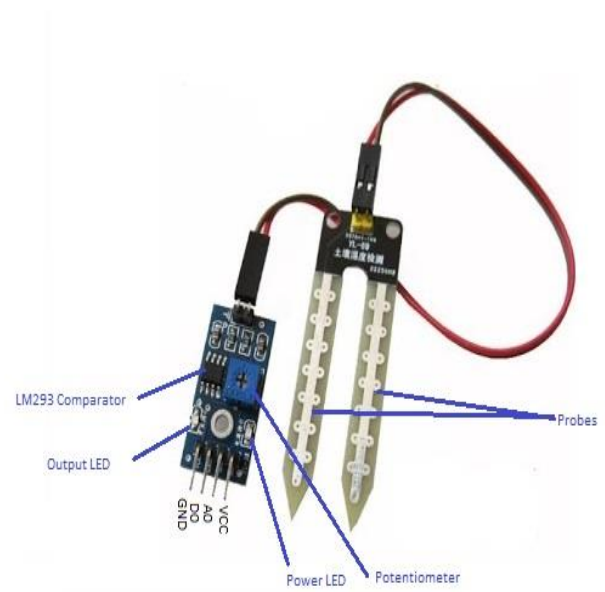


Figure 11: Soil Moisture Sensor (YL-69)

3.2.1 Specifications

- Operating voltage: DC 3.3 v ~ 5 v
- Output voltage: 0 ~ 4.2 v
- Current: 35 mA
- LED: Power indicator (Red) and Digital switching output indicator (Green)
- Size: 60 x 20 x 5mm

3.2.2 Sensing Technique

Soil moisture sensor uses its two probes to pass current through the soil, and then it reads that resistance to get the moisture level. More water makes the soil conduct electricity more easily (less resistance), while dry soil conducts electricity poorly (more resistance). The output voltage of the sensor changes accordingly to the water content in the soil.

When the soil is wet, the output voltage of the sensor decreases. While dry soil leads to increasing of the output voltage.

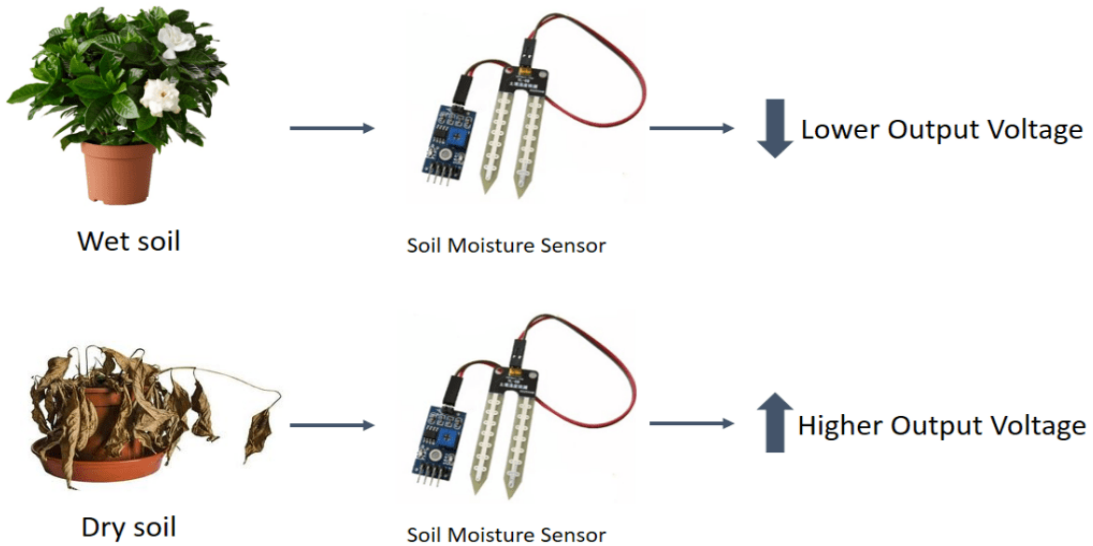


Figure 12: YL-69 Working Analogy

3.2.3 Connection with NodeMCU

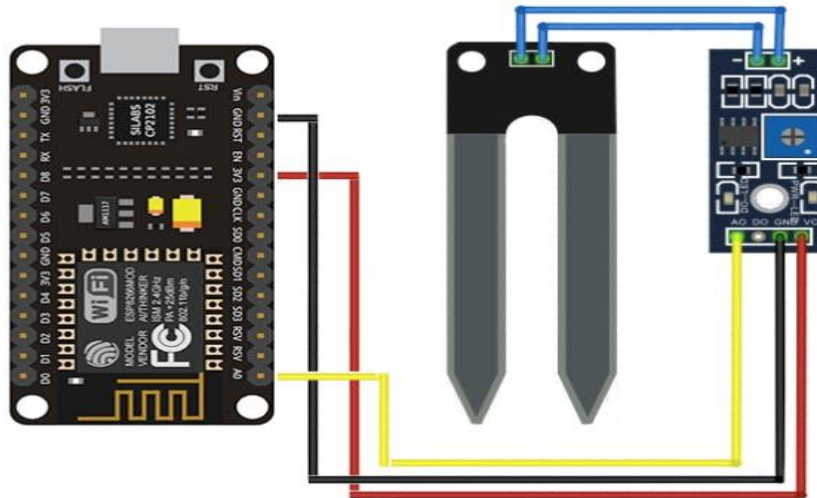


Figure 13: Connection of YL-69 with NodeMCU

Connect the VCC pin (first pin from right) of the sensor to the 3.3V pin of the Node MCU.

Connect the GND pin of the sensor (second pin from right) to the GND pin of the Node MCU.

Connect the Analog pin (fourth pin from right) to the A0 pin of the Node MCU.

3.3 Code Snippets

```
void setup()
{
  dht.begin();

  pinMode(rainPin, INPUT);

  Serial.begin(baud);
}

void loop()
{
  // Read the input on analog pin 0:

  S32 sensorValue = analogRead(rainPin);    // Read soil moisture sensor value
  Serial.print(sensorValue);                // Print soil moisture sensor value
  if(sensorValue < thresholdValue)         // Comparing reading of sensor with threshold value
  {
    Serial.println(" - Doesn't need watering");
  }
  else
  {
    Serial.println(" - Time to water your plant");
  }

  hum = dht.readHumidity();                 // Read humidity value from sensor
  temp= dht.readTemperature();             // Read temperature value from sensor

  //Print temp and humidity values to serial monitor

  Serial.print("Humidity: ");
  Serial.print(hum);                       // Print humidity percentage
  Serial.print(" %, Temp: ");
  Serial.print(temp);                     // Print temperature value in celsiu
  Serial.println(" Celsius");

  delay(del);                               // Adjustabled delay between measurements
}
```

Figure 14: Code Snippets for sensors interface with NodeMCU

Chapter 4: Network

4.1 Wireless Sensor Network (WSN)

Wireless sensor network is a network consists mainly of spatially distributed sensor nodes over a specific area.

A sensor node is a device that converts a physical attribute or environmental condition into readings that could be understood by humans.

Node's components depend mainly on the application and network needs but any sensing nodes' main components are usually: sensing module, communication module source of power and memory.

4.1.1 Advantages

- **Ease of deployment:**

The wireless sensors can be deployed easily in the desired field or area without the need of professional persons or any preorganization, which results in increasing flexibility of components arrangement and reducing the installation cost.

- **Mobility:**

It is the most important advantage of wireless sensor networks as nodes' places aren't fixed anymore and there is no constraints on them as these wireless sensors are equipped with a battery. Thus, if a region becomes unmonitored or any problem happens in certain node, we can easily rearrange the nodes to be distributed efficiently achieving their target.

- **Extended range:**

Before wireless sensor networks appear, a macro wired sensor was used. Such wired sensor's size is huge and can only sense limited region.

After wireless sensor networks appear, this huge wired sensor is replaced with many smaller wireless sensors forming network with the same cost of a single macro sensor.

These wireless sensors are distributed over a wider area allowing us to cover larger area with same cost and nearly same power consumption.

- **Fault tolerance:**

In macro sensors, the failure of one node makes that area completely unmonitored until this node is replaced.

While in wireless sensor networks, the failure of one node may not affect the network operation because of the large number of deployed sensors and data redundancy as all sensors collect same data.

In fact, non-functionality or failure of one node or more will affect the accuracy of collected data so it depends on the importance of data accuracy in the application, but overall the entire area will still be covered and the whole network will still be monitored and functioning

4.1.2 Design Requirements

1- Nodes in wireless sensor network should be self aware, self adapting and self configurable such that when any failure occurs in any node, it should be able to fix it without any need to external intervention.

2- Exploit spatial diversity and density of sensor nodes to build an adaptive node sleep schedule such that if a certain area have high sensors density we can make use of that by making one sensing node only take readings while other surrounding nodes are asleep and so on..

This could be achieved by time division multiplexing technique as in our network.

3- Characterize the relationship between deployment density and network size by taking into consideration the trade off between network size (density) and protocol complexity.

4- Consider the tradeoff between data redundancy and bandwidth consumption, so if our system is band-limited we won't be able to take so much readings so data redundancy will be reduced which may leads to reducing the accuracy.

4.1.3 Deployment Topologies

1) Randomly deployed sensor network:

In this deployment topology, nodes (sensors) are distributed randomly in empty regions by ejecting them from low-flying plane or drone or even by human.

The information sensed by these sensors must be collected at a central location known as a base station (BS) or sink node and it can be located either far away from the sensors or centered in the network.

This topology is often used in regions exposed regularly to storms or volcanos or generally in regions that have mysterious nature.

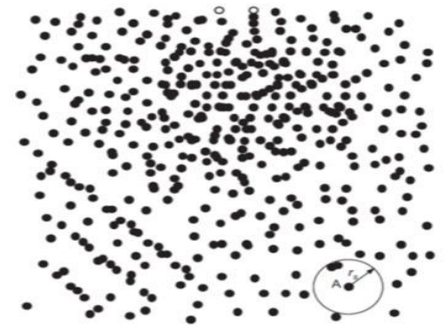


Figure 15: Randomly Deployed WSN

2) Regularly deployed sensor network:

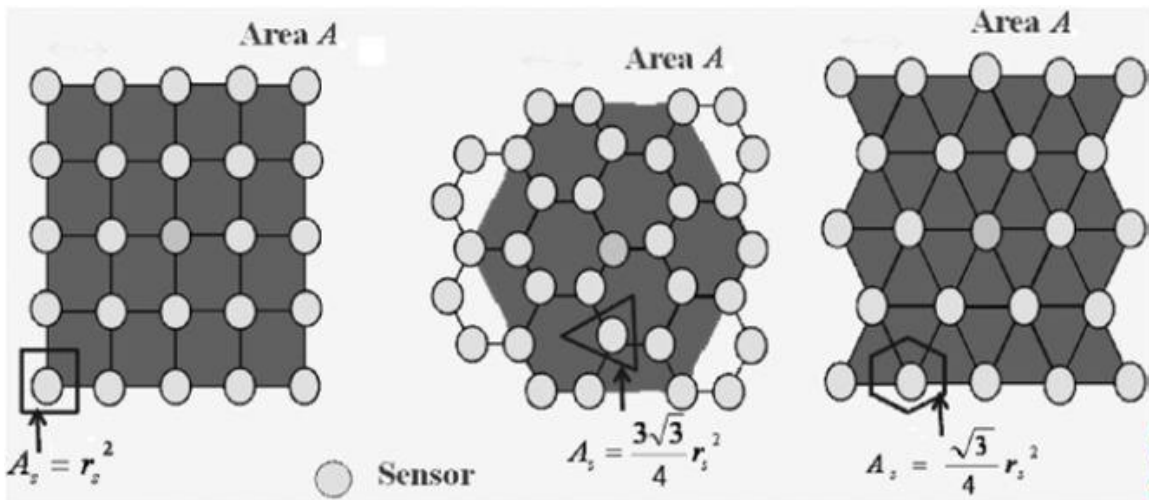


Figure 16: Regular deployed WSN

In such topology, sensors are distributed following a regular pattern, and many distribution topologies are possible. But square, hexagonal, and triangular patterns allow the same pattern to be repeated to cover a larger area.

So, here we consider only three modes, even though these three themselves could be combined to form a composite cell.

In square shaped pattern, sensors are put in the square sensors and the coverage area of each sensor is nearly square shaped.

In hexagonal shaped pattern, sensors are put in the hexagonal corners while the coverage area of each sensor is nearly triangular shaped.

In triangular shaped pattern, sensors are put in the triangle corners while the coverage area of each sensor is nearly hexagonal shaped.

After some calculations, it is found that the triangular shaped pattern has the highest coverage area so it has minimum cost.

This topology is often used in civilian applications such as buildings or in agricultural fields.

4.2 Communication Techniques

Communication nodes are the nodes responsible of transferring data from sensors to the web server.

4.2.1 First Topology

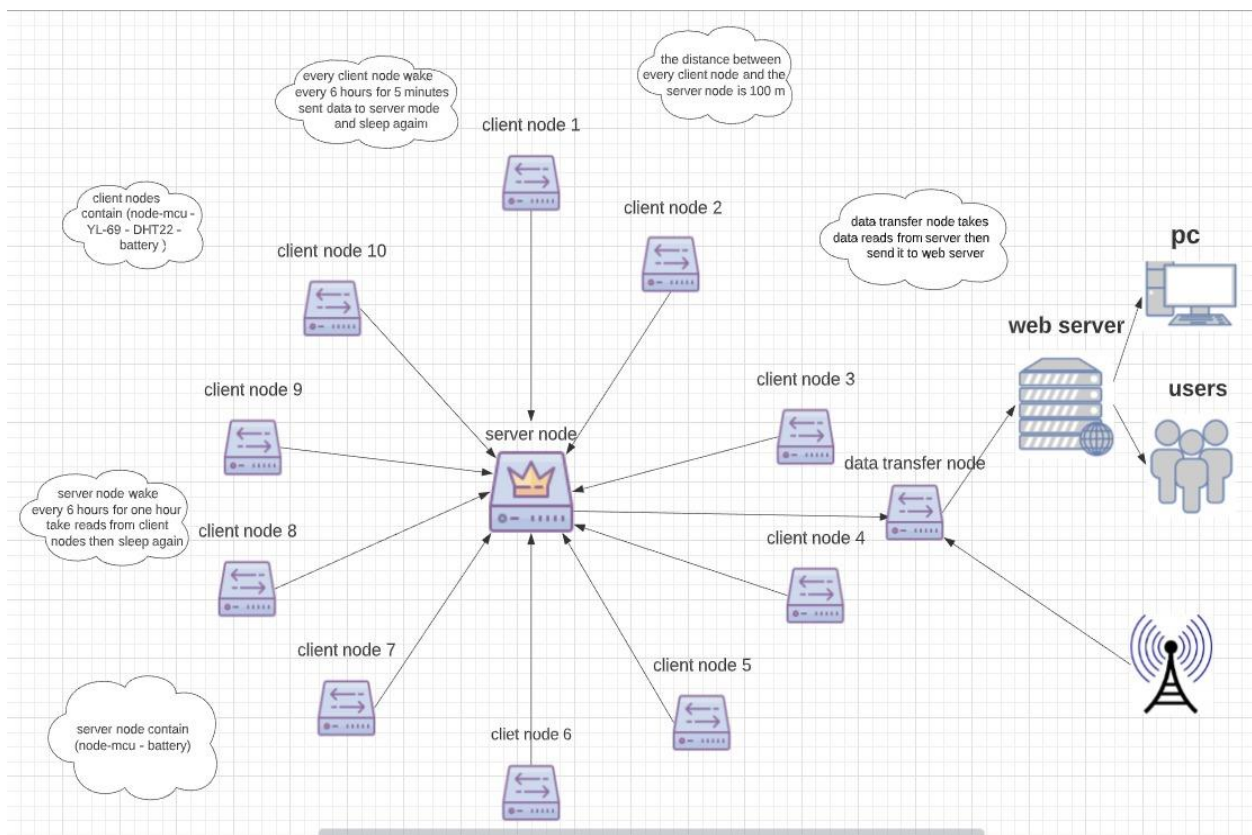


Figure 17: First Topology Structure

How it works

This topology consists of master node which is the server node and 10 surrounding client nodes.

Its idea depends on time division multiplexing (TDM) technique; as each client node wakes up for five minutes every 6 hours sensing the environmental changes either air temperature and humidity or soil moisture and access the server node to send its readings to it and then it goes in sleep mode.

Each client node takes more than one reading in its wakeup period to increase the accuracy of reading and for data redundancy.

The server node in this topology acts as access point that collects all data sent by the surrounding client nodes and store them.

The server node after storing data sent it to another node working as client node which sent the data to the web server after connecting to the access point using GPRS module.

Disadvantages

This topology's main disadvantage is the bad maintenance; because if there is a problem in certain client node or it doesn't work, it will be hard and complicated to know the spoiled node and its actual problem.

Also, client node's code is more complicated than server node's code so if there is certain problem it will be harder to be traced and repaired. And it is harder to duplicate this code to the 10 nodes.

The server node cannot sent the data to the server node directly but it sent it to another node to do this which leads to use more hardware.

Nodes Components

Server Node: NodeMCU and battery.

Client Node: NodeMCU, DHT22, YL-69 and battery.

4.2.2 Second Topology

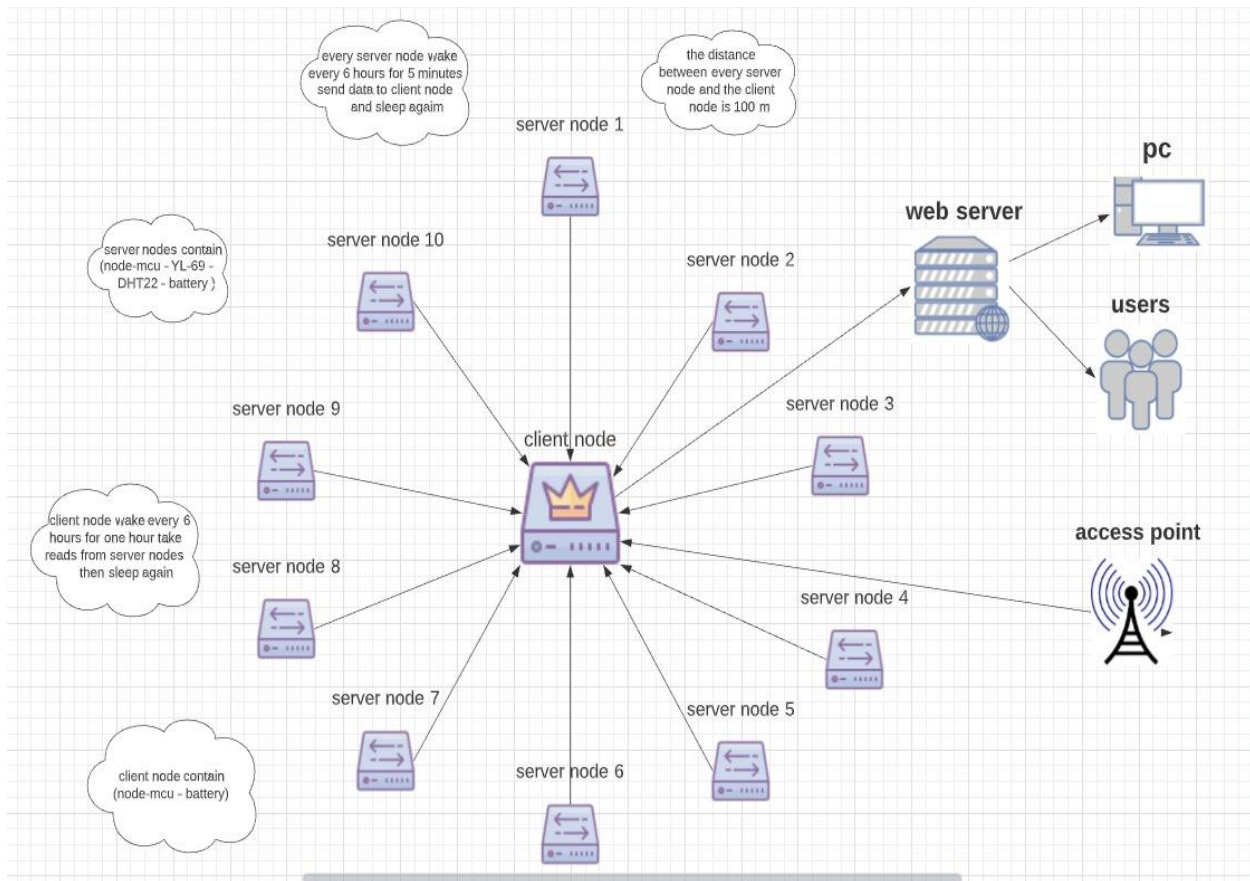


Figure 18: Second Topology Structure

How it works

This topology consists of master node which is the client node and 10 surrounding client nodes.

Its idea depends on time division multiplexing (TDM) technique; as each server node wakes up for five minutes every 6 hours sensing the environmental changes either air temperature and humidity or soil moisture.

Each server node takes more than one reading in its wakeup period to increase the accuracy of reading and for data redundancy.

The client node in this topology acts as access point that collects all data sent by the surrounding client nodes and store them.

The client node wakes up every 6 hours for one hour, within this one hour it accesses one node every 5 minutes collect the readings from it the sleep for one minute.

The client node after collecting data from the node it sent them to the web server and in this time it is connecting to access point using GPRS module.

Advantage of the topology

This topology has solved the main disadvantage of the first topology which was bad maintenance , as it makes it easier to detect the node which have problem either in sensing the environment or in the battery .

It makes this by accessing the web pages of all nodes then detect the node which have problems in sensing , but if it doesn't find the web page of the node it means that this node have problems in the power and need to change the battery.

The complex code become in one node which is the master node (client node) and this make it easier to update it , as updating one node is easier from updating 10 nodes.

The client sent the data to the web server directly after connecting to the access point without the need to another node which less hardware component.

Nodes component

Client Node: NodeMCU and battery.

Server Node: NodeMCU, DHT22, YL-69 and battery.

That's why we choose the second topology to use it.

4.3 Nodes distribution

Every node will be put in the agricultural field with 100 meters distance separation between any node and another, this distance have been chosen depending on two main factors which are communication range of the wifi module, and sensing range of both sensors.

The agricultural field's watering system is divided into two different parts, each of them will be treated as separated field.

4.4 Network Components

The details of each sensor are discussed thoroughly in previous chapters. As for the ESP 8266, it is a 3 V Wi-Fi module, and it is very popular in IoT applications. It is known for capability of functioning in industrial environments, due to its wide operating temperature range.

4.5 Architecture

Nodes: Each node consists of an ESP 8266, DHT 22 soil moisture sensor, YL69 temperature and humidity sensor, and a battery to power up the node. Each node establishes its own local server to upload data from sensors to it. C code with the required functionalities is burnt on the ESP 8266, which is connected to the sensors.

Client node: It established in the center. Client node treats the other surrounding nodes as access points, it accesses each node respectively, collects the data from each node's local server, then stores this data as parameters. Later on, this data is uploaded to the main live web server, and gets updated approximately every 5 minutes.

The client node will be in the center of 9 nodes, it will be composed of an ESP8266 module, and the total of 10 nodes will be spread over 5 acres of land.

4.6 Timeline

The network is of nodes works according to a specific timeline, where each component does its job, then goes to sleep until it is needed again. This is obviously a great way to save power consumption and overall costs.

The timeline goes as follows; a node is active for 5 minutes, then all nodes go to sleep for 1 minute, after that the next node is active for 5 minutes, then all nodes go to sleep for 1 minute, and so and so forth. This goes on for an hour, then the whole network goes to sleep for 5 hours, as the data will be redundant before this period passes. There is no benefit of collecting environmental data from the same place continuously, it is not efficient, thus the sleep mode was crucial for power and cost efficiency.

The network timeline is illustrated in the next figure for more clarity.

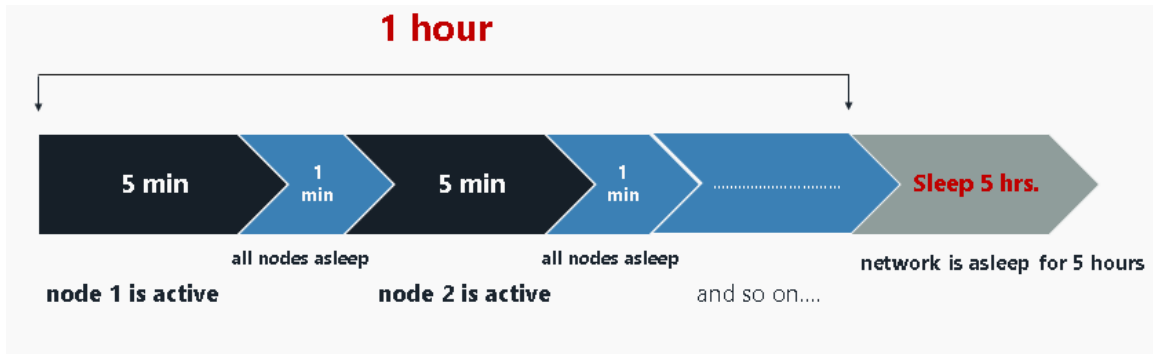


Figure 19: Time Division

4.7 Gateway

4.7.1 GPRS

General Packet Radio Service (GPRS) is a packet oriented mobile data service on the 2G and 3G cellular communication system's global system for mobile communications (GSM). GPRS was originally established by European Telecommunications Standards Institute (ETSI) in response to the earlier CDPD and i-mode packet-switched cellular technologies. It is now maintained by the 3rd Generation Partnership Project (3GPP).

GPRS wireless data module is the ready solution for remote wireless applications, machine to machine or user to machine and remote data communications.



Figure 20: GPRS Module

Features:

- Small, lightweight and easy to integrate.
- Low power consumption.
- Internal SIM card reader and option on external SIM card reader.
- Always connected with higher data transmission speed.

- High performance on low price.
- Single supply voltage: 3.4V – 4.5V.
- Typical power consumption in SLEEP mode is 1.5mA.
- Communication by using AT commands.

4.7.2 Mi-Fi

MiFi is a brand name used to describe a wireless router that acts as mobile Wi-Fi hotspot.

It allows multiple end users and mobile devices to share a 3G or 4G mobile broadband Internet connection and create an ad-hoc network.

An ad-hoc network is a network that builds as devices connect to it. For MiFi, your mobile devices and possibly your laptop connect to the MiFi device, forming a small LAN (Local Area Network).

As MiFi evolves, it's quite likely that the number of users and devices that can connect to the system will increase and the devices will have a longer range.



Figure 21: Mi-Fi

How it works?

A MiFi will pick 3G or 4G LTE internet from the cell tower of the mobile operator. That's why most MiFis come with a sim card which accurately identifies the device on the network. Some MiFi might include a Micro SD card for extra functionality, but we have really found no use of an SD card slot.

To share internet among multiple users, MiFis come with WiFi chips. Most MiFi support WiFi 802.11 b/g/n which operates in the 2.4 GHz frequency band. Very few support the latest faster WiFi standard 802.11 ac which works on 5GHz band.

Features:

- It is small in size as its dimensions in tens of mm
- It weights nearly 80-130 g
- It needs Nano SIM card
- High Speed Data up to 150 Mbps
- Innovative and compact MIFI which allows sharing High Speed Internet with up to 15 users through standard WiFi operation.

4.8 Code Snippets

Client Node

```
void loop() {

    unsigned long currentMillis = millis();

    if(currentMillis - previousMillis >= interval) {
        // Check WiFi connection status
        if ((WiFiMulti.run() == WL_CONNECTED)) {
            temperature = httpGETRequest(serverNameTemp);
            humidity = httpGETRequest(serverNameHumi);
            soilmoisture = httpGETRequest(serverNameSmois);
            Serial.println("Temperature: " + temperature + " *C - Humidity: "
                + humidity + " % - Soilmoisture: " + soilmoisture + " %");
            delay (60000);

            // save the last HTTP GET Request
            previousMillis = currentMillis;
        }
        /*
        else {
            Serial.println("WiFi Disconnected");
        }*/
    }
}

String httpGETRequest(const char* serverName) {
```

Figure 22: Client Node Snippets

```

WiFiClient client;
HTTPClient http;

// Your IP address with path or Domain name with URL path
http.begin(client, serverName);

// Send HTTP POST request
int httpResponseCode = http.GET();

String payload = "--";

if (httpResponseCode>0) {
  Serial.print("HTTP Response code: ");
  Serial.println(httpResponseCode);
  payload = http.getString();
}
else {
  Serial.print("Error code: ");
  Serial.println(httpResponseCode);
}
// Free resources
http.end();

return payload;

```

Server Node

```

void setup() {
  // Serial port for debugging purposes
  Serial.begin(115200);
  Serial.println();

  // Setting the ESP as an access point
  Serial.print("Setting AP (Access Point)...");
  // Remove the password parameter, if you want the AP (Access Point) to be open
  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid, password);
  WiFi.softAPConfig(ip, gateway, subnet);
  Serial.print("AP IP address: ");
  Serial.println(ip);

  server.on("/temperature", HTTP_GET, [] (AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readTemp().c_str());
  });
  server.on("/humidity", HTTP_GET, [] (AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readHumi().c_str());
  });
  server.on("/soilmoisture", HTTP_GET, [] (AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readSmois().c_str());
  });
}

```

Figure 23: Server Node Snippets

Chapter 5: Power Analysis

Mainly this chapter will talk about the amount of power dissipated by this Wireless Sensor Network. But it will be divided into 2 sections which are:

- 1) The operating current of each component.
- 2) Ways of calculating the power dissipated by this network.

And the goal of this analysis is only to know our battery lifetime, how long it will survive feeding our circuit with full efficiency.

5.1 Operating Currents

This section will show the operating current of each circuit component we use under different circumstances.

Let's start with the core of our system:

1) **ESP8266 NodeMCU**

NodeMCU is the most power consuming part of our Network due to its functionality as it is the core of our system that collects the data and send them to the gateway node which also have a NodeMCU. This sending and receiving processes are done through the ESP-12E chip.

ESP8266 mainly have varying power consumption between **15uA** and **400mA** depending on its usage. In its idle state with the WiFi connection up its power consumption is around **70mA** with operating voltage **3.3V** (to have full operation) as **2.5V** are sufficient to operate it.

In order to reduce power consumption as it is used in multiple IoT applications this module has many sleep modes.

NodeMCU Sleep modes

	Modem-Sleep	Light-Sleep	Deep-Sleep
Wi-Fi	OFF	OFF	OFF
System Clock	ON	OFF	OFF
RTC	ON	ON	ON
CPU	ON	Pending	OFF
Substrate Current	15 mA	0.4~0.9 mA	10~20 uA

Figure 24: NodeMCU Sleep Modes

2) Temperature and Humidity Sensor (DHT22)

As mentioned in the sensors chapter it is a low power consumption sensor with a very high accuracy as its **operating current** is **2.5mA** during measuring which is a very low value and also its **operating voltage** is between **2V** and **5V** which can be easily found.

3) Soil Moisture Sensor (YL-69)

Not like the temperature and humidity sensor our soil moisture sensor's operating current is slightly higher which is around **35mA** but still not a big deal. And it can be operated with **3.3V** or **5V** voltage sources.

- In our project there are two types of nodes:
 - A) Client node:** The node that has all the sensors and collect data then send it to the server node.
 - B) Server node:** The node that collects the data from all the client nodes, process it and send it to our main webserver. This node don't interface with any sensors but only the NodeMCU.

5.2 Theoretical Analysis

In the calculations we will deal with all the circuit components as they are connected in parallel because they all have the same power supply and connected to a common Ground.

First: Client node

A) Idle mode

It is the mode when Wi-Fi is on but not sending any data so the NodeMCU current consumption ($I_{NodeMCU}$) is around 70 mA. While the other sensors are collecting data normally so there operating currents are 2.5 mA (I_{Temp}) for DHT22 and 35 mA (I_{SM}) for YL-69.

So total current consumption (I_{Total}) will be:

$$I_{Total} = I_{NodeMCU} + I_{Temp} + I_{SM}$$

$$I_{Total} = 70mA + 2.5mA + 35mA$$

$$I_{Total} = 107.5mA$$

So the total power consumption (P_{Total}) when connected to a 5V power supply will be:

$$P_{Total} = V * I_{Total}$$

$$P_{Total} = 3.3V * 107.5mA$$

$$\boxed{P_{Total} = 0.354 \text{ Watts}}$$

B) Tx mode

So the NodeMCU is working as a Tx with current $I_{NodeMCU} = 170mA$

$$\therefore I_{Total} = I_{NodeMCU} + I_{Temp} + I_{SM}$$

$$I_{Total} = 170mA + 2.5mA + 35mA$$

$$I_{Total} = 207.5mA$$

So the total power consumption will be:

$$\therefore P_{Total} = V * I_{Total}$$

$$P_{Total} = 3.3V * 207.5mA$$

$$\boxed{P_{Total} = 0.684 \text{ Watts}}$$

C) Deep-Sleep mode

When in deep-sleep mode all the modules are off but only the RTC is on so $I_{NodeMCU} = 10 \mu A$

and all the sensors stop taking readings so $I_{Temp} = I_{SM} \approx 0 A$

$$\therefore I_{Total} \approx 10 \mu A$$

$$\therefore P_{Total} \approx 3.3V * 10 \mu A$$

$$\boxed{P_{Total} \approx 33 \mu \text{Watts}}$$

Second: Server node

Since the server node has only NodeMCU $\therefore I_{Total} = I_{NodeMCU}$

A) Idle mode

As the Client node $I_{NodeMCU}$ is around 70 mA

$$\therefore P_{Total} = V * I_{NodeMCU}$$

$$P_{Total} = 3.3V * 70mA$$

$$\boxed{P_{Total} = 0.231 \text{ Watts}}$$

B) Rx mode

When NodeMCU works as receiver its $I_{NodeMCU} \approx 85 mA$

$$\therefore P_{Total} = V * I_{NodeMCU}$$

$$P_{Total} = 3.3V * 85mA$$

$$\boxed{P_{Total} = 0.2805 \text{ Watts}}$$

C) Deep-Sleep mode

The same as the client node $\therefore P_{Total} \approx 33 \mu\text{Watts}$

5.3 Practical Analysis

In our circuit the current consumption is very low (all in the range of mille amperes). This is made a huge problem in using many current sensors as they can't sense this small amount of current. From the current sensors used but couldn't sense the flowing current in the circuit is **ACS712** and **MAX471**.

In order to solve this problem we used **Instrumentation Amplifier (AD620AN)** with a certain wirings in order to sense the small currents.

This type of amplifiers was mainly used because it is used to amplify very low-level signals, rejecting noise and interference signals.

So Instrumentation amplifier has many characteristics:

- Very low input signal.
- Easy adjustable gain using single control (resistance).
- High input impedance and low output impedance to avoid loading.

5.3.1 Instrumentation Amplifier in Power measurements

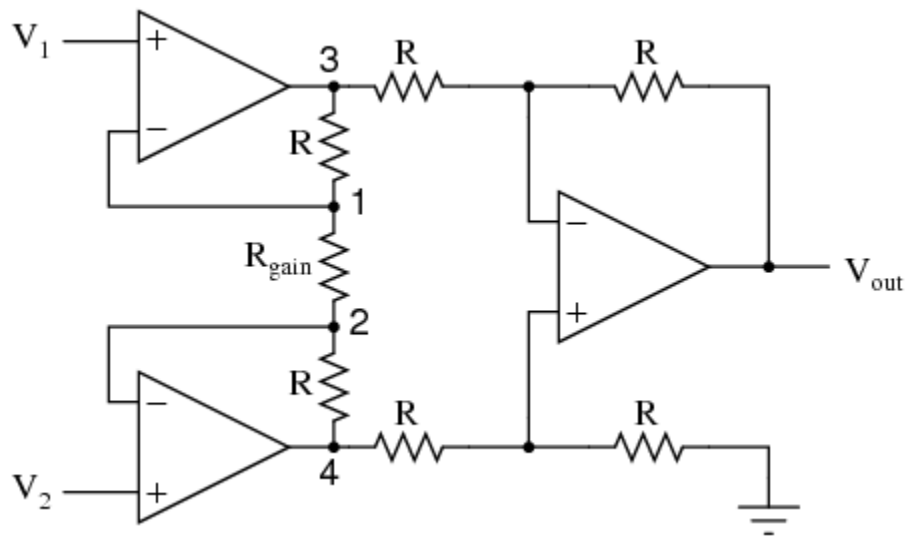


Figure 25: Instrumentation Amplifier using Op-Amps

The gain of the instrumentation amplifier is controlled by only a resistance called (R_{Gain}) so the gain formula is as follows:

$$Gain = \frac{49.4 \text{ k}\Omega}{R_{Gain}} + 1$$

$$\therefore R_{Gain} = \frac{49.4 \text{ k}\Omega}{Gain - 1}$$

How do we use the instrumentation amplifier to serve our needs?

We connect our circuit as follows:

In this connection we put a resistance R with very small value approximately equal 1 so when the voltage divided between load resistance and R, the voltage at node Vx will be equal to the current flowing in the loop according to ohms law $I = \frac{V}{R}$, $\therefore I = V$

Then this voltage (and current) is amplified by a controllable gain factor at the output Vout. This reading is inputted to an Arduino in order to be processed.

For more accuracy, we take 50 samples/second of the current then calculate its average and multiply it by the operating voltage and consider this as our first power reading. Repeat this process 1000 times so we will be having 1000 power readings which will form a curve integrate the area under the curve to get the energy of 1000 times then divide by 1000 to get our accurate energy reading.

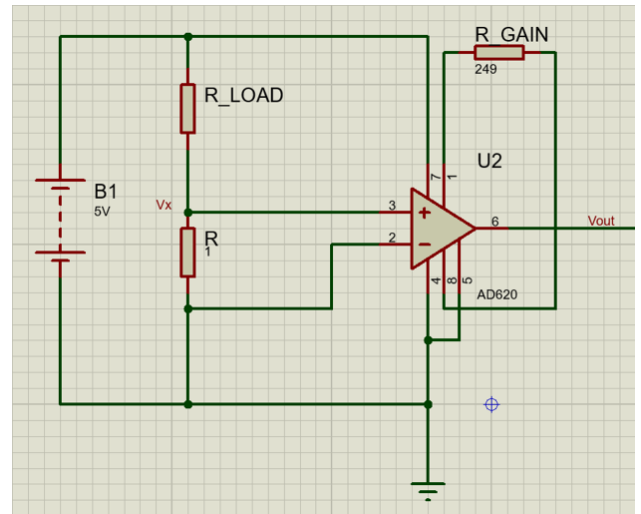


Figure 26: Simulation using Instrumentation Amplifier

Chapter 6: Packaging

Packaging is the science, art, and technology of enclosing or protecting products. Packaging also refers to the process of design, evaluation, and production of packages.

6.1 Packaging Phases

- Shape designing which depend mainly on the dimensions and shape of the product that you want to cover and protect.
- Choosing Suitable material which mainly depend on the surrounding environment, material of the product and the packaging reasons.
- Manufacturing it.
- Testing it to ensure that it fits the requirements.

6.2 Why Packaging?

Generally, packaging is very important as the objects enclosed in the package may require protection from many damaging factors such as: mechanical shock, vibration, electrostatic discharge, compression, temperature, dust, pollution, etc...

In our irrigation system, nodes are placed in the field in the open air so each node requires to be packaged to protect the electronic components from irrigation water, rain water, climate changes, human stepping on land, bird's messing up, etc...

6.3 Choosing the material

Firstly, we exclude metals because they are good conductors of heat so, in high temperature they may affect the sensors readings and cause battery damage.

Then, we tend to choose wood as our packaging material because it is bad conductor of heat, strong material it can easily be shaped, but our only constrain was that wood material may be affected by exposing to water, so we find out that wood can be painted by polyurethane to insulate it from water and prevent it from damage.

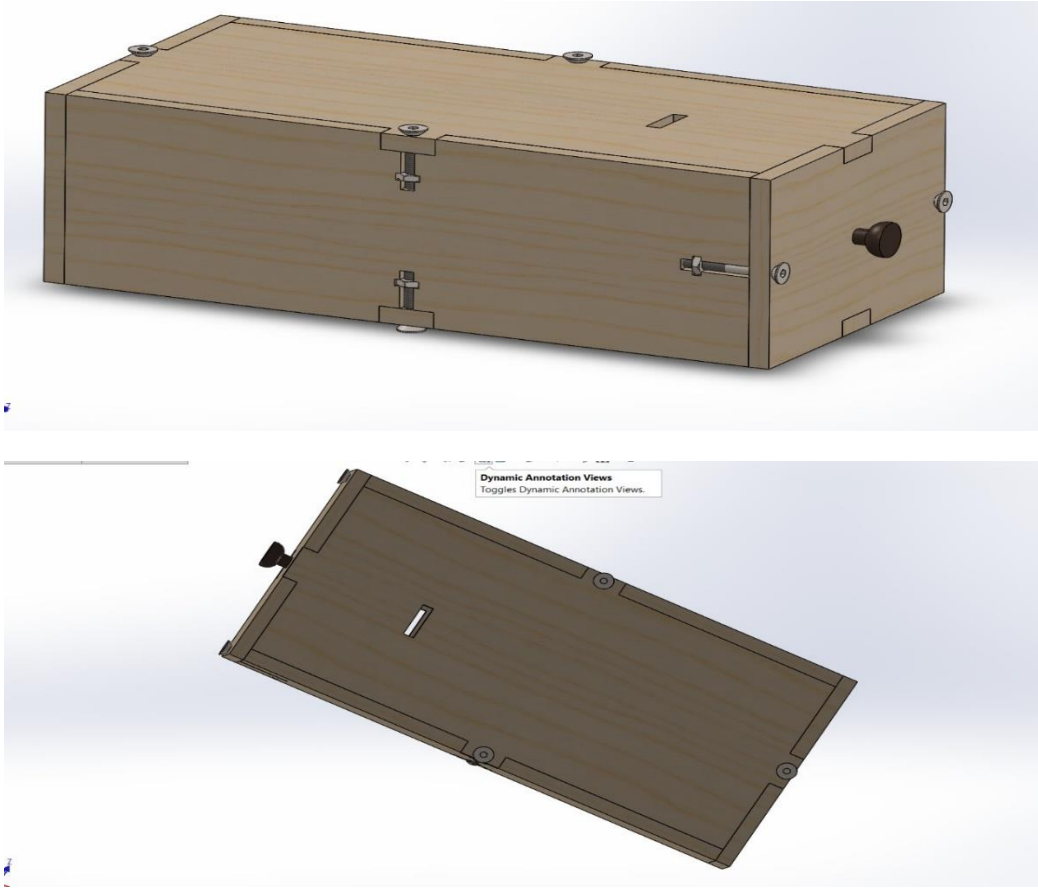


Figure 27: Package Design

After that we have to exclude wood material also because we find out that polyurethane may cause cancer.

Finally, we decided to use acrylic material as acrylic is a transparent plastic material with outstanding strength, stiffness, and optical clarity. Acrylic sheet is easy to fabricate, bonds well with adhesives and solvents, and is easy to thermoform. It has superior weathering properties compared to many other transparent plastics, so it fit our needs of protection.

Chapter 7: Webserver and Application

7.1 Definition

Web servers can be defined on two related levels, the hardware level, and the software level.

The hardware level, it is a computer connected to the internet, it stores the software parts of the server; such as the HTML and CSS, JavaScript codes.

As for the software level, it is responsible for the user's experience, how they access files and data. An HTTP server is a software that understands URLs and HTTP. The connection between client and server is done through HTTP requests. When a client needs a file/piece of data, it requests it using HTTP requests (e.g.: get request), then the server sends the requested file to the client also using HTTP requests.

A good way to classify a server is to define whether it is static or dynamic. Static servers contain static content, no need for updating or generating data on the fly. As for dynamic servers, it can update the content on the fly from the back-end database connected to it. From which it can be concluded that dynamic web servers are more complex and harder to build than static ones.

The web server discussed here is a dynamic server. Data on the server gets updated on the fly as new data is read from sensors. The main objective of a server in this project is to pass the data from sensors to an end-user, to be able to control and access this data remotely.

The process of creating the web server underwent several steps to reach the current results. These steps will be discussed respectively in the following:

- 1) Uploading sensors data to Thingspeak. Thingspeak is an open-source IoT application to upload, store, and access data using HTTP protocol.

Thingspeak was used in early stages in the project to access the data from sensors. In the following stages, it was necessary to make a custom server, for more security and confidentiality for the collected data.

2) Creating a server

A test local server written in JavaScript in the form of a jar file was made, to test basic HTTP requests like get and post on dummy data.

A jar is a file format based on the ZIP file format. It stands for Java ARchive and it is a collection of Java code. It is used to combine many Java class files together. Jar files can be unzipped using terminal commands, such as, “java -jar .\jar-file-name.jar”.

The server was tested using Postman API, to test Get and Post requests from and to the database. The tests were carried out successfully.

The coding process was carried out upon the successful testing. An HTTP client instance is created, and the server is established. Post requests are then tested using json format, containing dummy sensor data. The server responds to this post request with a certain ID. The first post request is then terminated, and a second post request is established containing the ID response in the json format. In this way, the readings are uploaded to the URL that is in the form <the_ip>:<the_port_number>/<api>. For example it could be 192.168.1.7:7000: average-moisture. The server response should be 200 in order to confirm the post request is successful.

7.2 Code Snippets

```
void setup() {
  Serial.begin(115200); //Begin Serial at 115200 Baud
  WiFi.begin(ssid, password); //Connect to the WiFi network

  while (WiFi.status() != WL_CONNECTED) { //Wait for connection
    delay(500);
    Serial.println("Waiting to connect...");
  }

  Serial.print("IP address: ");
  Serial.println(WiFi.localIP()); //Print the local IP

  server.on("/", handle_index); //Handle Index page

  server.begin(); //Start the server
  Serial.println("Server listening");
}

void loop() {
  server.handleClient(); //Handling of incoming client requests
}

void handle_index() {
  //Print Hello at opening homepage
  server.send(200, "text/plain", "Hello! This is an index page.");
}
```

Figure 28: Webservice Code

Chapter 8: Machine Learning

8.1 General Approach

The agriculture sector faces multiple challenges linked to diseases, pests and water shortage as well as improper soil treatment. so we resort to Artificial intelligence with its vast learning capabilities to solve these agriculture problems.

The work will focus on the data gathering, analysis and decision-making process to ensure the application of the correct amount of water at the correct moment.

Machine learning in this part is needed to support in the decision-making process of the farmer to follow the correct treatment plans and to document all necessary data.

8.2 Machine Learning

Machine learning (ML) offers an alternative to the conventional engineering flow when problems are too complex to develop a solution with guarantees.

It is the application of artificial intelligence (AI) that provides the ability to automatically learn and improve from training sets without explicitly programmed instructions. The job of the modeling algorithm is to find the most applicable mapping function from input variables to output variables and aid in the discovery of rules and patterns in the data sets. This paper reviews what ML can do in the agricultural sector, specifically in Egypt with the objective of developing a disease detection system that is robust and easy to adapt to different applications and crops, while following the criteria above.

8.2.1 Methods of Machine Learning

The dataset is divided into a train and a test set, both consisting of the features of analysis and the KPI. The train set is used to fit the model and define the relation between the input features and the KPI, whereas the test set is used to measure how accurate the model is predicting the output given the test features

Different types of algorithms and models can help achieve different goals. We have two types of problem classification and regression.

Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y). The output variables are often called labels

or categories. The mapping function predicts the class or category for a given observation. It is common for classification models to predict a continuous value as the probability of a given example belonging to each output class. The probabilities can be interpreted as the likelihood or confidence of a given example belonging to each class. A predicted probability can be converted into a class value by selecting the class label that has the highest probability. A classification problem requires that examples be classified into one of two or more classes, and can have real valued or discrete input variables

Regression was developed in the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables, but has been borrowed by machine learning. It is both a statistical algorithm and a machine learning algorithm.

Regression predictive modeling is the task of approximating a mapping function (f) from input variables (X) to a continuous output variable (y). A continuous output variable is a real-value, such as an integer or floating-point value. These are often quantities, such as amounts and sizes.

Because a regression predictive model predicts a quantity, the skill of the model must be reported as an error in those predictions.

There are many ways to estimate the skill of a regression predictive model, but perhaps the most common is to calculate the root mean squared error, abbreviated by the acronym RMSE

- Regression problem requires the prediction of a quantity
- Regression can have real valued or discrete input variables.

Conclusion:

- .Classification problems are different from Regression problems
- .Classification is the task of predicting a discrete class label
- .Regression is the task of predicting a continuous quantity

Six different algorithms were implemented which are: Random Forest, Extra Tree Regression, Linear Regression, Xtreme Gradient Boosting, Support vector Machines and Logistic Regression. The following table compares between these algorithms in terms of their types, descriptions and applications.

Method	Type	Description	Applications
Logistic Regression	Classification	Assumption of existing logistic relationship between KPI and features	Trauma and Injury Severity Score (TRISS)
Support Vector Machine (SVM)	Classification	Segregation of data points using non linear hyperplanes	Identifying the classification of genes, patients on the basis of genes and other biological problems.
Linear Regression	Regression	Finding linear relation between input and output while minimizing mean squared error	financial portfolio prediction, salary forecasting, real estate predictions and in traffic in arriving at ETAs.
Random Forest	Regression	Decision tree analysis	In clinical decision support system (CDSS).
XGBoost	Regression & Classification	Different models are combined reducing the sum of errors of all models (Decision tree approach)	In view of prediction techniques of hourly PM2.5 concentration in China.
Extra Trees	Classification	Combines the predictions from many decision trees.	For time-series classification and for the automatic inference of near-optimal sequential decision.

Figure 29: Machine Learning Algorithms

This table shows an overview of some commonly used ML algorithms and their applications. The following subsections will describe these algorithms and their mathematical and statistical background further.

Logistic Regression

Logistic Regression is a classification method used when the dependent variable is categorical, It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variable(s). In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function like to predict whether an email is spam, or whether a tumor is malignant or not.

The output of the model is the estimated probability. This is used to determine how confident the model is regarding its prediction given any input.

Logistic regression is based on the basic assumption of an existing logistic relationship between the dependent variable (KPI) and the independent features or observations, by fitting data to a logit function, it predicts the probability of occurrence of the event.

To explain this through an example, having a group of patients and by collecting data regarding the physical conditions of the patients like blood pressure, glucose, skin thickness, insulin, pregnancy and age and based to this information it's determined if they are diabetic or not.

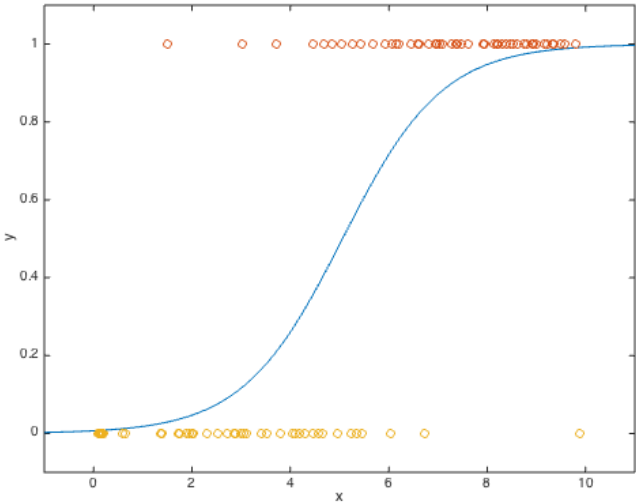


Figure 30: Logistic Regression example results

This form of regression chooses parameters that maximize the likelihood of observing the sample values rather than that minimizing the sum of squared errors (like in ordinary regression).

Support Vector Machine (SVM)

SVM is a supervised ML algorithm, mostly used in classification problems. Each data point is plotted in an n-dimensional space (n-features) with the value of each feature being the value of the particular coordinate. The classification is performed by finding the hyperplane/line that differentiates the categories and segregates the data points.

The hyperplane is selected, that segregates the two classes better, while maximizing the distances between the nearest data point (either class). This distance is called the Margin. The hyperplane with higher margin is selected due to its robustness. If the selected hyperplane has a low margin, there will be a high chance of miss-classification.

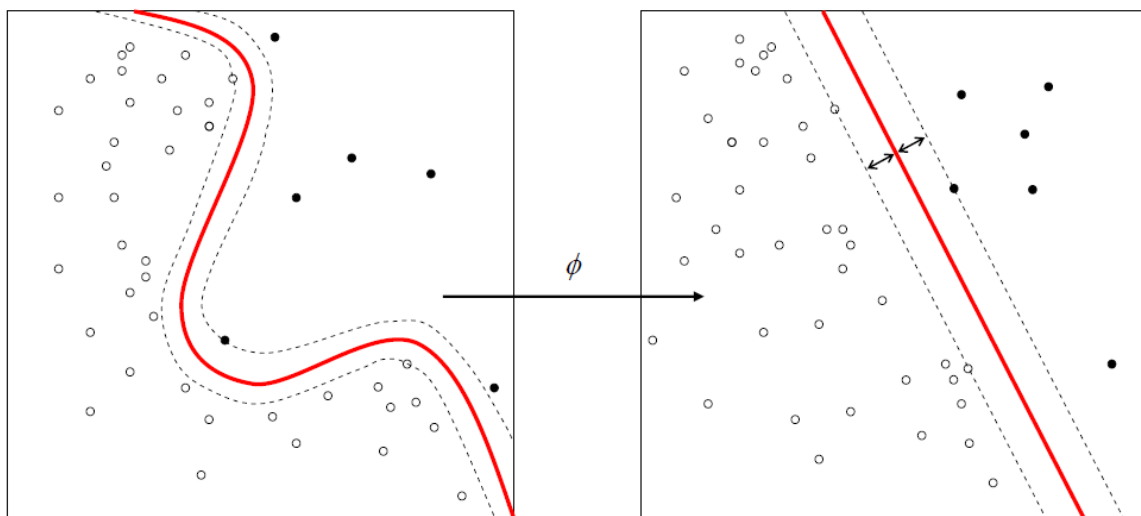


Figure 31: SVM example results

The figure shows a distribution of data points in a 2-dimensional space with a non-linear hyperplane segregating the values. It solves the problems by introducing additional features to convert non-separable problems to separable problems using so called kernels.

SVM also has the ability to ignore outliers and find the hyper-plane that has maximum margin. It is mostly useful in non-linear separation problem using complex data transformations, then finding the function to separate the data based on the labels or outputs defined.

Linear Regression

Linear Regression predicts values of a KPI as a linear combination of the independent observations/features. The linear coefficients are determined so as to optimize the error function (mean squared error) of the predictions. In summary, a set of independent features is used to predict one dependent KPI.

Establishing relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation $Y = a * X + b$.

In this equation:

- **Y** is the Dependent Variable.
- **a** is the Slope.
- **X** is the Independent variable.
- **b** is the Intercept.

These coefficients **a** and **b** are derived based on minimizing the sum of squared difference of distance between data points and regression line.

Linear Regression is of mainly two types: Simple Linear Regression and Multiple Linear Regression. Simple Linear Regression is characterized by one independent variable. And, Multiple Linear Regression (as the name suggests) is characterized by multiple (more than 1) independent variables. While finding best fit line, you can fit a polynomial or curvilinear regression.

For this and other ML-algorithms, the dataset is split into a training and a test set. During the training step, the linear coefficients of the model are calculated and used to predict the values of any unknown pattern, which is not provided in the training set. The patterns of features in the test set are then used to predict the corresponding KPIs, with the error for the evaluation of the model being calculated as the mean squared error between the predicted values of the test set and the actual values.

The linear regression works with the basic assumption of the existence of a linear relationship between the KPI and the independent features measured. and as mentioned previously When applied with multiple independent variables, it's referred to as multiple linear regression

Random Forest

Random Forest (RF) is a ML algorithm making predictions based on multiple decision trees. To classify a new object based on attributes, each tree gives a classification and a rank for that class. The classification with the highest rank (over all the trees in the forest) is chosen and in case of regression, the average of outputs by different trees is used for predictions.

In Random Forest, each tree is set up as follows: N cases in the training set are defined. Samples of these cases are taken at random but with replacement They will be the training set for growing the decision tree.

eXtreme Gradient Boost (XGBoost)

XGBoost is an implementation of gradient boosted decision trees designed for more speed and performance.

:The algorithm provides a system for use in different computing environments such as

- Parallelization of the tree constructions using the entire CPU cores during the training phase
- .Distributed Computing to train large models using a cluster of machines
- .Out-of-Core Computing for large datasets that would not fit into the memory
- .Cache Optimization of data structures and algorithms making the most of the hardware

XGBoost is mostly used for execution speed and model performance. It is an approach where new and different models are created that can predict the errors of the other models to make the final prediction. It uses a gradient descent algorithm to reduce the loss while .constantly adding new models and it can be used for both regression and classification problems

Extra Trees

Extra-Tree method (standing for **extremely randomized trees**) has the main objective of further randomizing tree building in the context of numerical input features, where the choice of the optimal cut-point is responsible for a large proportion of the variance of the induced tree.

With respect to random forests, the method drops the idea of using bootstrap copies of the learning sample, and instead of trying to find an optimal cut-point for each one of the K randomly chosen features at each node, it selects a cut-point at random.

This idea is rather productive in the context of many problems characterized by a large number of numerical features varying more or less continuously as it leads often to increased accuracy thanks to its smoothing and at the same time significantly reduces computational burdens linked to the determination of optimal cut-points in standard trees and in random forests.

From a statistical point of view, dropping the bootstrapping idea leads to an advantage in terms of bias, whereas the cut-point randomization has often an excellent variance reduction effect.

This method has yielded state-of-the-art results in several high-dimensional complex problems.

From a functional point of view, the Extra-Tree method produces piece-wise multilinear approximations, rather than the piece-wise constant ones of random forests.

8.3 Analysis

Now we will describe in detail the ML algorithms used and their outcomes and uncertainties over two obtained data sets. The insights from these tests can then be used to implement the backed layer of the IoT in the field.

Datasets

A lot of research has previously been done on protected crops in greenhouses to control pests and diseases by biological means instead of the use of pesticides.

This analysis will cover two separate data sets, one obtained within such a greenhouse and the other obtained in a plain field. As these analyses and models need a lot of data points which can

take up to years to gather, the insights gained from the two available test sets will then be applied on the running system with a window for optimization

One important problem domain here is the quality of the available data. Real data can be imperfect in the sense that it can be:

- **Incomplete:** missing values for some attributes and objects.
- **Irrelevant:** some attributes do not relate to the problem at hand but are mistakenly recorded
- **Redundant:** involving unknown and unexpressed relations between attributes
- **Noisy:** attributes can have measurement errors.
- **Erroneous:** transcribed incorrectly.

ML algorithms need to be stable enough to deal with imperfect data and to discover correlations that are useful for the problem at hand

8.3.1 Dataset 1: Potato Blight

The first dataset used for this analysis was collected by Dr. Mohamed Fahim from the department of Plant Pathology at Cairo University. The set contains data of weather conditions within potato areas that were collected during four consecutive seasons, i.e. 2002/2003, 2003/2004, 2004/2005 and 2005/2006. The weather data were recorded manually in Badrashin region.

Also Dr. Mohamed Fahim in gathering this data generalized some features which made it impossible to try some of multi feature ML algorithms such as the Relative Humidity to be above 90% during the whole observation time and the start of observing to be after 40 Days of planting that leads the data to be a one feature dataset.

Original State

The dataset concludes 303 records in four seasons from 2002 until 2006. Each season lasted around four months between October and February. The measurements were not started until mid-November with around 40 day after planting when the crop is already on the surface

and lasted until 115 days after planting, when the crop was harvested. Each record represents one day of measurement.

The next table shows the columns in the dataset provided by Dr. Fahim., as well as information on how the data was recorded. These features are then visualized in Figures after the table. They are plotted over time which is represented by the index of each record. As stated above, the records are all measured between the 40th day and the 115th day after planting.

Feature	Type of Feature	Description
Date	Observed	Date of observation
DAP	Observed	Days after planting
Tmin	Observed	Lowest Temperature recorded this day
Tmax	Observed	Highest Temperature recorded this day
Rain days>0.1mm	Observed	The accumulated number of rain days with rain more than 0.1 mm
Season	Observed	The current season of the measurements
Daily blight obsrv	Observed	Observation of disease severity on the current day

Figure 32: Original State of Potato Light Dataset

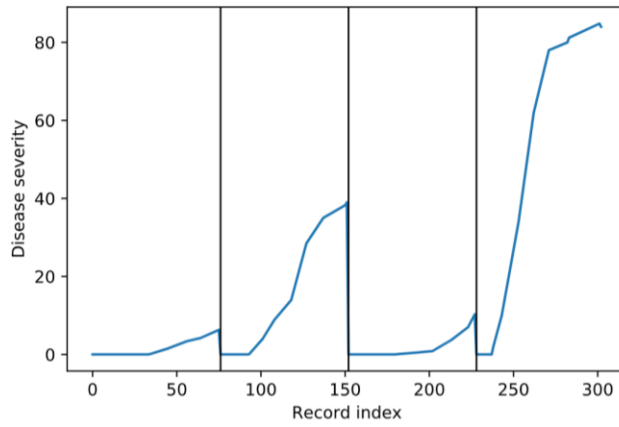


Figure 33: Disease Severity over Time

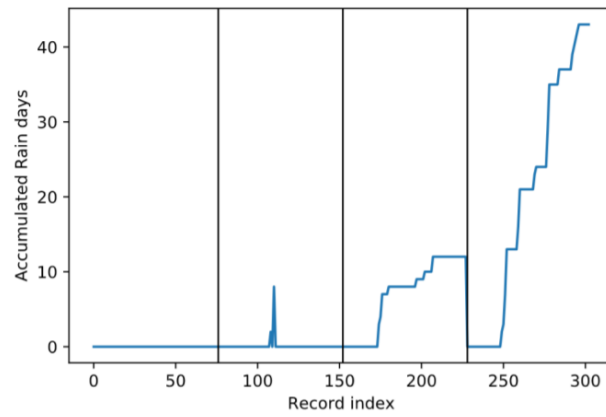


Figure 34: Accumulated Rain over Time

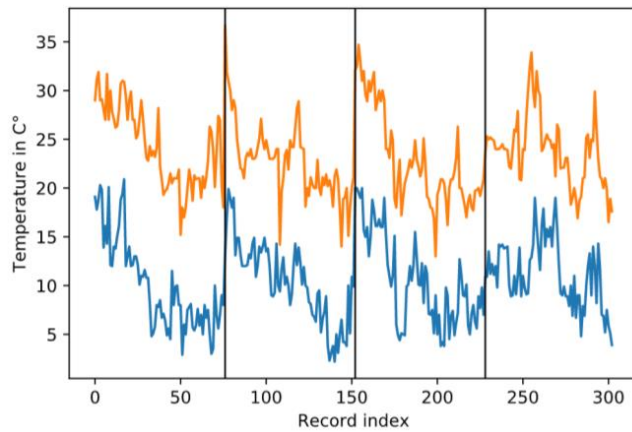


Figure 35: Tmax & Tmin over Time

The first subplot show the disease severity (DS) over time in each season. It is clearly visible that the DS in season two (peak at 39%) was higher than in season one (peak at 6.3%) and three (peak at 10.3%). The DS peaked it season four at 84%. Pesticides were not used during these experiments which lead to a steady increase in the DS over time due to the lack of treatment.

It is important to note that the infections always started between DAP 50-70 of each season.

In the first season it started on Day 73, in the second season on day 56, in the third season on day 68 and in the fourth season on day 49, which explains the differences in the peaks of the DS.

The second and third plot show the amount of rain during the season as well as the highest and lowest temperature during the four season. It is notable that the rain in season four was much higher compared to the other seasons which could indicate a connection between the DS and the rain in future analysis.

The highest and lowest temperature of the day is fluctuating throughout the season but a negative trend is visible indicating that it is getting colder towards January which matches reality.

Additional features for Analysis

The original features are used to calculate additional weather related features to help boost the ML models. The additional features are described in next table.

Record	Type of record	Description
Tmean	Calculated	Daily average temperature
GDD	Calculated	Growing day degree
Accu. GDD	Calculated	Accumulated growing day degree
IP result	Calculated	Length of infection period
lastDS	Calculated	Disease severity on previous day
meanlast3DS	Calculated	Average of disease severity of last three days

Figure 36: Additional calculated Features for Potato Blight Dataset

GDD calculation: The GDD is calculated as $GDD = T_{\text{mean}} - 10$

Infection period calculation: Infection periods (IPs) were calculated from daily temperature and rainfall obtained from the weather stations. The Hannukkala method requires five consecutive days with minimum temperature 8 °C or above, maximum temperature below than 25 °C and ten days with rain > 0.1mm.

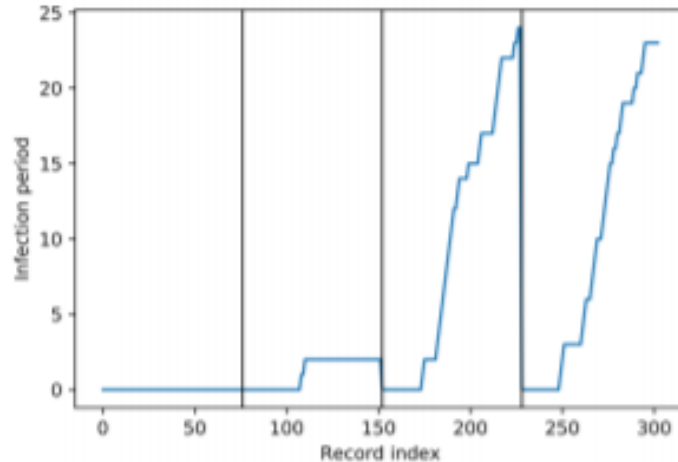


Figure 37: Infection Period over Time in days

Previous disease severities: In Dr. Fahim's experiments, no pesticides were used, leading to a steady growth of the blight in the field depending on the environmental factors. To capture this increase, two features were introduced giving an information about the situation of the potato blight over the past three days and specifically on the previous day.

Additional measurements of wind speed and soil moisture can be valuable for the prediction model and should be added in future data collections but will not be taken into consideration in the current example.

8.3.2 Dataset 2: Cotton Leaf Worm

The second dataset used for this analysis was collected by Dr. Haitham from the department of electrical engineering at Cairo University. The set contains data of weather conditions inside a controlled greenhouse system where Egyptian Cotton is being planted. The weather data has been collected manually for the past two years and is planned to be recorded for another year.

Original State

The dataset concludes 108 records between 09/2017 and 09/2019 with each record representing one day of measurement. The Cotton plantation works as a continuous process with new plantations and harvests everyday. It takes 35 to 40 days for each individual plant to from seedling to harvest. This leads to a constant state during the entire time of measurement where there are always young and old plants living simultaneously in the greenhouse.

Next table shows the columns in the dataset provided by Dr. Haitham, as well as information on how the data was recorded. These features are then visualized in Figures after the table. They are plotted over time which is represented by the index of each record. As stated above, each record represents a whole week of measurement.

Record	Type of record	Description
No. S. littoralis	Observed	Disease severity
H Temp	Observed	Highest Temperature recorded this week
L Temp.	Observed	Lowest Temperature recorded this week
RH	Observed	Relative humidity
Bt	Observed	Biological control protocol

Figure 38: Additional Calculated Features for Cotton Leaf Worm Dataset

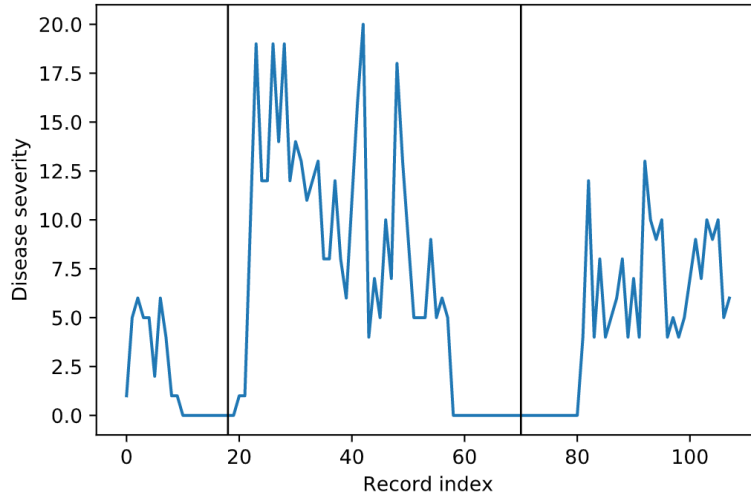


Figure 39: Disease Severity over Time

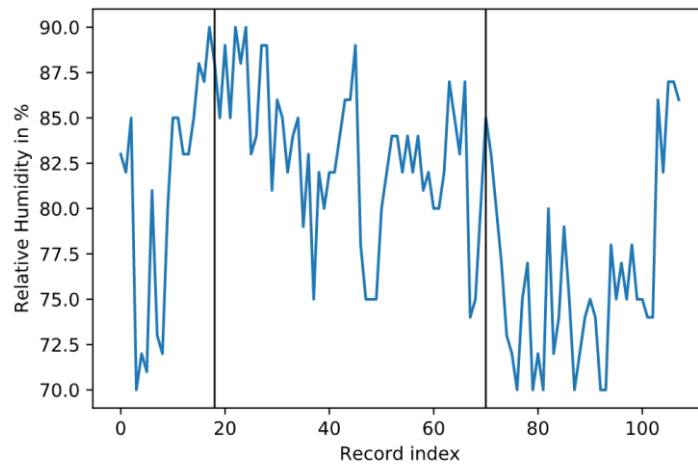


Figure 40: Relative Humidity over Time

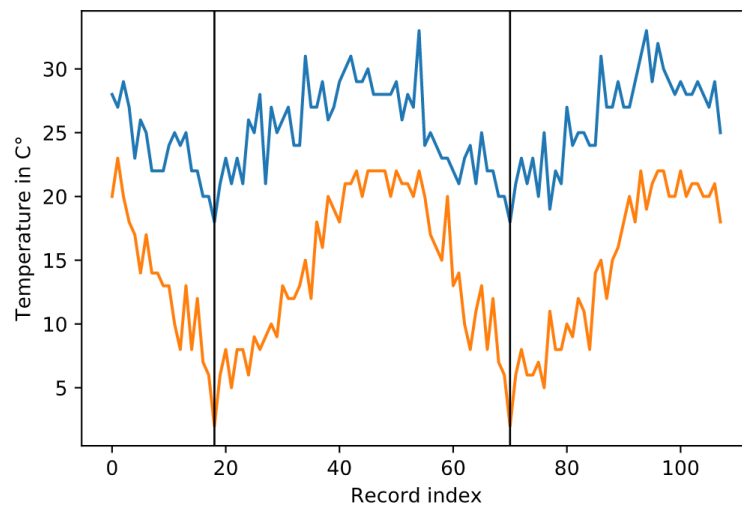


Figure 41: Tmax and Tmin over Time

8.4 Machine Learning Analysis

The datasets described in section 8.3 will be analyzed using the ML models described in section 8.2 to determine the accuracy of each model and rank their performance. The best performing ML models will then be used for hyper-parameter tuning to achieve better results.

To rank the performance of the ML models an error function needs to be introduced. This error function compares the predicted values with actual data points provided from the test set. The most popular error functions are the **RRMSE** and the **MAPE**.

RRMSE: The Relative Root Mean Square Error is the relative standard deviation of the residuals. These prediction errors measure how distant from the actual values the predicted values are. It is also a measure for how spread out they are. It simply explains how concentrated the data points are around the line of best fit and is also commonly used in climatology, forecasting, and regression analysis to verify experimental results. With d_i being the actual value and f_i being the forecast value, the formula is given by

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (d_i - f_i)^2}$$

The RMSE is then divided by the average of the actual values to get the RRMSE for analysis

MAPE: The mean absolute percentage error is a measure for how accurate a forecast system is. With d_i being the actual value and f_i being the forecast value, the formula is given by

$$MAPE = \frac{1}{n} \sum_{i=1}^n \frac{d_i - f_i}{d_i}$$

As described in subsection 8.2.1, the dataset is split into a training and a test set. In the following analysis, the split is 80:20 as is typical in most ML analysis. The sampling will differ in both analysis.

Six different algorithms were tested, Random Forest, Extra Tree Regression, Linear Regression, Xtreme Gradient Boosting, State vector Machines and Logistic Regression and the results are collected in the next tables.

After fitting and training the model with the data points and the DS for each record, the model is applied on the features of the test set. The predicted values for these combinations of environmental data are then compared with the actual DS of these records using the error functions. The Table shows the RRMSE and the MAPE achieved for the predictions using each .of the above algorithms

In the first part of this analysis all the observed features were used and the model was enhanced by the additional environmental features like the GDD and the IP.Information on previous DS .was not introduced at this point

The Five different sampling methods were used for analysis, splitting the dataset in five different ways. For the first four predictions, the set was split for each of the four seasons, meaning that for the first prediction, the training set consisted of all the records from season two-four while the data from season one was predicted using that fitted model. The DS of season one was then used for evaluation.

A fifth and more common method was then used, the random sampling. Here the dataset is split randomly using the above mentioned 80:20 rule using records from all seasons in the training as well as in the test set. This method ensures that other differences between the seasons are captured as well.

8.4.1 Analysis on Potato Blight Dataset

This subsection describes the ML analysis performed on the dataset provided by Dr. Fahim on Potato Blight from subsection 8.3.1.

Sampling	RF		ExtraTrees		Linear		XGB		SVM		Logistic	
	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE
S1	1392%	1044%	1231%	964%	1461%	1049%	541%	314%	1790%	1252%	1640%	1214%
S2	125%	101%	120%	95%	129%	98%	124%	95.15%	122%	87.27%	121%	90.5%
S3	2138%	1669%	2973%	2404%	1749%	1448%	2336%	169 8%	2506%	2053%	4007%	3142%
S4	111%	94.36%	111%	94.74%	172%	143%	114%	97.25%	118%	99.84%	118%	100%
Random	18.18%	9.42%	14.59%	7.89%	75.96%	36.40%	89.09%	44.36%	156.04%	76.78%	64.18%	29.41%

Figure 42: Analysis on Potato Blight without Previous Disease Severity

The same predictions were then performed an additional time adding the information on the previous DS to each record. Again the predictions were made using all six ML models and the five different sampling techniques for comparison of the two error metrics. The results are listed in the next figure

Sampling	RF		ExtraTrees		Linear		XGB		SVM		Logistic	
	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE
S1	25.88%	14.81%	25.58%	14.11%	28.89%	20.81%	268%	47.22%	3453%	3014%	44.94%	23.15%
S2	40.45%	23.71%	24.74%	18.41%	2.63%	2.02%	98.82%	60.89%	173%	133%	108%	74.87%
S3	24.11%	13.46%	303%	234%	20.53%	16.57%	420%	98.13%	3656%	2333%	60.42%	32.71%
S4	64.35%	51.31%	63.08%	50.01%	1.95%	1.66%	62.22%	49.90%	63.12%	53.37%	61.56%	49.95%
Random	6.99%	3.54%	4.18%	1.84%	0.96%	0.40 %	55.84%	14.69%	241%	196%	11.16%	6.37%

Figure 43: Analysis on Potato Blight with Previous Disease Severity

The prediction errors can be seen to have improved after introducing the previous disease information while the classification algorithms still underperform with regards to the other algorithms.

8.4.2 Analysis on Cotton Leaf Worm Dataset

This subsection describes the ML analysis performed on the dataset provided by Dr. Haitham on Cotton Leaf Worm from subsection 8.3.2.

Sampling	RF		ExtraTrees		Linear		XGB		SVM		Logistic	
	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE	RRMSE	MAPE
W Temp features	50.97%	34.67%	69.20%	42.20%	63.78%	51.27%	73.35%	40.22%	262.55%	233.91%	104.98%	65.22%
W Prev DS	52.25%	34.97%	47.25%	33.30%	41.24%	33.07%	73.00%	42.39%	168.17%	117.39%	71.01%	36.96%
W/o Temp & prev DS	57.81%	34.60%	66.19%	39.89%	64.46%	52.25%	90.63%	54.35%	180.97%	152.17%	103.48%	63.04%
W Temp & prev DS	46.27%	32.51%	37.20%	26.85%	44.14%	33.53%	73.00%	42.39%	99.39%	63.04%	77.32%	47.83%

Figure 44: Analysis on Cotton Leaf Worm Dataset with Random Sampling

8.5 Results

After we managed to make an implementation for the six algorithms, we applied them on dummy data downloaded from the internet, there were 2 files the first one contains data regarding years of experience vs expected salary. We managed to get very low error percentage and very high accuracy (up to 97%) and we were able to predict the salary given the years of experience. And listed below is the code snippets of the data and results.

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

In [2]: data1=pd.read_csv('Salary_Data.csv')

In [3]: data1.head()
Out[3]:
  YearsExperience  Salary
0                1.1  30343.0
1                1.3  46205.0
2                1.5  37731.0
3                2.0  43525.0
4                2.2  39891.0

In [4]: x=data1.iloc[:, :-1].values

In [5]: y=data1.iloc[:, 1].values

In [8]: from sklearn.model_selection import train_test_split

In [9]: xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=0)

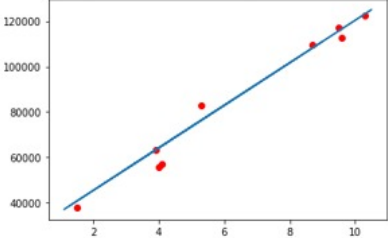
In [10]: from sklearn.linear_model import LinearRegression

In [11]: regressor=LinearRegression()

In [12]: regressor.fit(xtrain,ytrain)
Out[12]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [13]: y_pred=regressor.predict(xtest)

In [14]: plt.scatter(xtest,ytest,color='red')
plt.plot(xtrain,regressor.predict(xtrain))
Out[14]: [<matplotlib.lines.Line2D at 0x9889430>]
```



```
In [15]: from sklearn.metrics import r2_score
z=r2_score(ytest,y_pred)

In [16]: z
Out[16]: 0.9740993407213511
```

Figure 45: Results of applying Algorithms on Dummy Data

The second file contained data about patients such as glucose, blood pressure, age, pregnancies, skin thickness, insulin and by applying classification algorithms on this data the implemented model was able to predict if the patient is diabetic or not. And listed below is code snippets of the second dummy data file.

```
In [17]: data2=pd.read_csv('diabetes.csv')
In [18]: data2.head()
Out[18]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [19]: x2=data2.iloc[:,0:8].values
In [20]: x2
Out[20]: array([[ 6. , 148. , 72. , ..., 33.6 , 0.627, 50. ],
 [ 1. , 85. , 66. , ..., 26.6 , 0.351, 31. ],
 [ 8. , 183. , 64. , ..., 23.3 , 0.672, 32. ],
 ...,
 [ 5. , 121. , 72. , ..., 26.2 , 0.245, 30. ],
 [ 1. , 126. , 60. , ..., 30.1 , 0.349, 47. ],
 [ 1. , 93. , 70. , ..., 30.4 , 0.315, 23. ]])
In [21]: y2=data2.iloc[:,8].values
```

Then the algorithms were then applied on the original data which are 2 files the first one is the cotton leaf worm data gathered by Dr. Haitham in 2 years and listed below is code snippets for the relative mean absolute error which is quite acceptable since the data was imperfect.

```
[29] data3_cmp=pd.DataFrame(list(zip(y_pred, ytest.values)))
data3_cmp['Difference']=abs(data3_cmp[0]-data3_cmp[1])
data3_cmp.rename(columns={0:"Predicted",1:"Actual"},inplace=True)
data3_cmp.set_index(np.array(xtest.index),inplace=True)
data3_cmp.sort_index(inplace=True)

[30] RMSE = np.sqrt((data3_cmp['Difference'] ** 2).mean())
print("RMSE: %.2f" % (RMSE))
RMSE: 1.50

[31] Relative_RMSE = RMSE /np.mean(data3_cmp["Actual"])
print("Relative squared Error: %.2f%%" % (Relative_RMSE * 100.0))
Relative squared Error: 30.06%

[32] ME = (data3_cmp['Difference']).mean()
print("Mean error: %.2f" % (ME))
Mean error: 1.18

[33] Relative_ME = ME /np.mean(data3_cmp["Actual"])
print("Relative Mean Error: %.2f%%" % (Relative_ME * 100.0))
Relative Mean Error: 23.54%
```

Figure 46: Results of applying Algorithms on Cotton Leaf Worm Dataset

```
[34] plt.plot(data3_cmp.Actual, color="r")
      plt.plot(data3_cmp.Predicted, color="b")
```

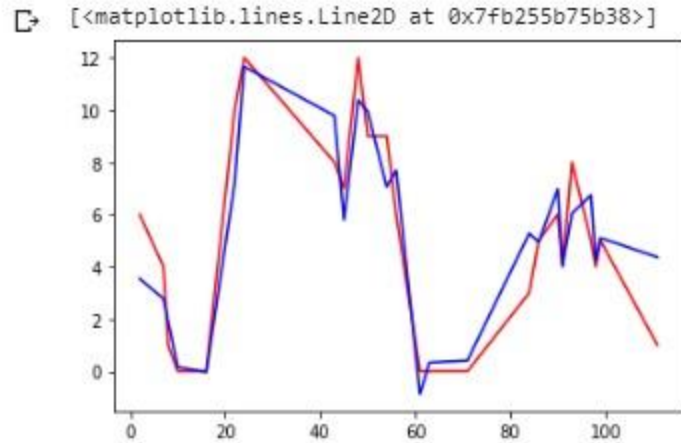


Figure 47: Predicted vs Actual for Cotton Leaf Worm Dataset

The second data file was the potato blight data gathered by Dr. Mohamed Fahim and it was more accurate data and gathered in 4 years instead of 2 which leads to the much better accuracy of the model and less relative mean absolute error (4.66%). And listed below is the code snippets of the results.

```
[17] data2_cmp.head()

   Predicted  Actual  Difference
5         [0.6]     [0]         [0.6]
7         [0.58]    [0]         [0.58]
8         [0.52]    [0]         [0.52]
12        [0.49]    [0]         [0.49]
15        [0.44]    [0]         [0.44]

[18] RMSE = np.sqrt((data2_cmp['Difference'] ** 2).mean())
      print("RMSE: %.2f" % (RMSE))

   RMSE: 0.78

[19] Relative_RMSE = RMSE / np.mean(data2_cmp["Actual"])
      print("Relative RMSE: %.2f%%" % (Relative_RMSE * 100.0))

   Relative RMSE: 5.80%

[20] MAE = (data2_cmp['Difference']).mean()
      print("Mean absolute error: %.2f" % (MAE))

   Mean absolute error: 0.62

[21] Relative_MAE = MAE / np.mean(data2_cmp["Actual"])
      print("Relative Mean absolute Error: %.2f%%" % (Relative_MAE * 100.0))

   Relative Mean absolute Error: 4.66%
```

Figure 48: Results of applying Algorithms on Potato Blight Dataset

```
[22] plt.plot(data2_cmp.Actual, color="r")
      plt.plot(data2_cmp.Predicted, color="b")
```

```
[<matplotlib.lines.Line2D at 0x7efe8533c9b0>]
```

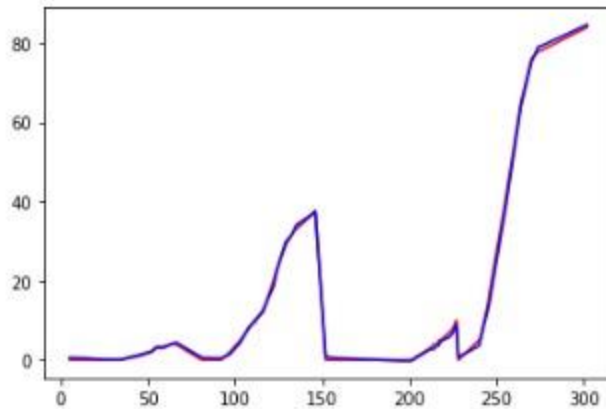


Figure 49: Predicted vs Actual for Potato Blight Dataset

After applying the algorithms on excel files http requests were used instead to fetch data (which in our case is the sensor readings) from and upload results (which is the data after applying ML Algorithms on it) to the data base through get and post requests and listed below is code snippets of the get and post request.

```
r = requests.get('http://localhost:7000/csv/cotton_leaf_1.csv')
data = r.content.decode('utf8')
data3 = pd.read_csv(io.StringIO(data))
```

```
requests.post('http://localhost:7000/accuracy', json.dumps({'rmse': RMSE, "r_rmse": Relative_RMSE, "me": ME, "r_me": Relative_ME, "class": "cotton_leaf"}))
```

8.6 ML Algorithms Simulator

8.6.1 GUI Simulator

GUI: Stands for "Graphical User Interface" and is pronounced "gooey." It is a user interface that includes graphical elements, such as windows, icons and buttons. The term was created in the 1970s to distinguish graphical interfaces from text-based ones, such as command line interfaces. However, today nearly all digital interfaces are GUIs.

8.6.2 GUI Programming in Python

Python provides various options for developing graphical user interfaces (GUIs). But Tkinter is the only framework that's built into the Python standard libraries

Tkinter : is the Python interface to the Tk GUI toolkit shipped with Python.

Tkinter is the standard GUI library for Python, Python when combined with Tkinter provides a fast and easy way to create GUI applications. Thus, Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter will done by the following steps:

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Widgets: are the bread and butter of the Python GUI framework Tkinter. They are the elements through which users interact with your program. Each widget in Tkinter is defined by a class.

Here are some of the widgets available:

- **Label:** A widget used to display text on the screen.
- **Button:** A button that can contain text and can perform an action when clicked.
- **Frame:** A rectangular region used to group related widgets or provide padding between widgets.
- **Canvas:** The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.

- **Check button:** The check button widget is used to display a number of options as check boxes. The user can select multiple options at a time.
- **List box:** The List box widget is used to provide a list of options to a user.
- **Menu button:** The menu button widget is used to display menus in your application.
- **Menu:** The Menu widget is used to provide various commands to a user. These commands are contained inside menu button.

8.6.3 Application

In this section we will introduce an application which will help the user to show the result without the need to run the code every time.

The application consists of 7 buttons 6 of them represent the six algorithms that will be used (Logistic regression - Support vector machine (SVM) - Linear regression - Random Forest - XGBoost - Extra trees), and the last button which is delete text and that's pressed to clear the results (Accuracy of model and error percentage) of any of the 6 algorithms selected previously to be able to select any other algorithm to be simulated and to show new results.

The user will press the button of the desired algorithm to be run on the data from the database and the result will be shown in the screen as shown in the bottom right corner following figure representing the front-end of the simulator designed.

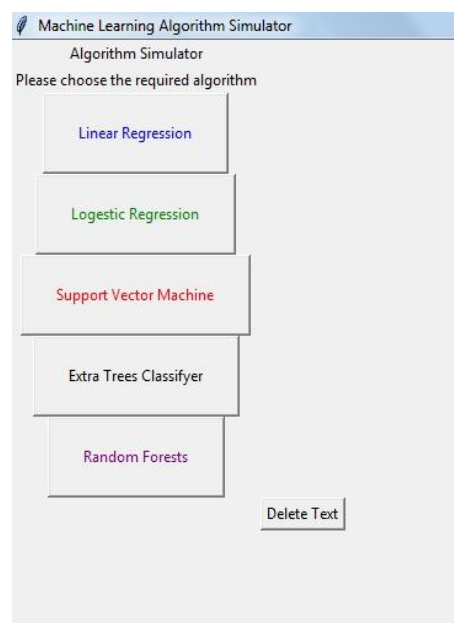


Figure 50: GUI Front-End

8.7 Code Snippets

```
1  from tkinter import *
2
3  root = Tk()
4
5
6
7  def LiRf(): ...
106 def LoRf(): ...
206 def SVMf(): ...
306 def ETCf(): ...
405
406 def RFf(): ...
505
506 #####
507 def myDelete():
508     Label11.grid_forget()
509     Label12.grid_forget()
510     Label21.grid_forget()
511     Label22.grid_forget()
512     Label31.grid_forget()
513     Label32.grid_forget()
514     LiR['state']=NORMAL
515     LoR['state']=NORMAL
516     SVM['state']=NORMAL
517     ETC['state']=NORMAL
518     RF['state']=NORMAL
519     #print(LiR.winfo_exists())
520 root.title("Machine Learning Algorithm Simulator")
521 #e = Entry(root)
522 #e.pack()
523 #e.get
524 #creating label widget
525 mylabel = Label(root, text="Algorithm Simulator")
526 Label2 = Label(root, text="Please choose the required algorithm")
527 #mylabel2 = Label(root, text="webpp")
```

Figure 51: GUI Code Snippets

Chapter 9: Future Work

9.1 On-Site Deployment

Our project is monitoring prototype, the system is just collecting the data and due to covid-19 period we was unable to deploy our network in the farm , so our first target is to deploy the network in the farm.

9.2 Line to Line Irrigation

The irrigation system now in the farm is half to half irrigation system , which means the water pump of the farm can't pull water less than the half of the farm , so we need communication module have high rang to cover this space and we will use few numbers of nodes to less the redundancy of the data.

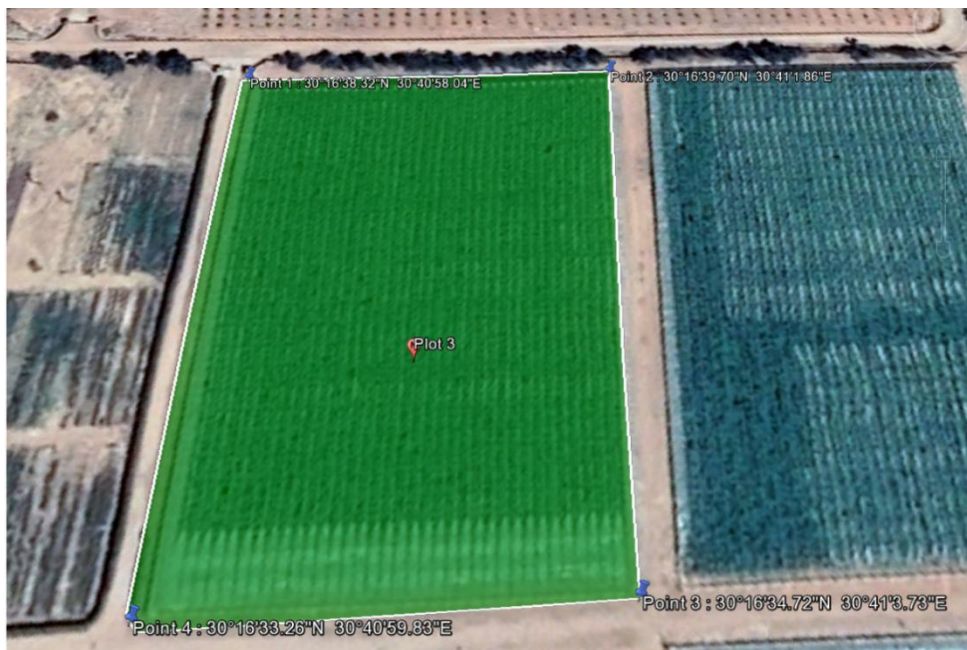


Figure 52: Half to Half Irrigation System

So our target is to make the irrigation system is line to line irrigation system , this will require to use more nodes to put one node at every line , so the distance between the nodes become 3 meters instead of 47 meters which was in the topology used with Wi-Fi module , in this case we can use different topology which use different communication module like Bluetooth module or ZigBee module which there features are they have less range and good power consumption and this will make the result is more accurate.

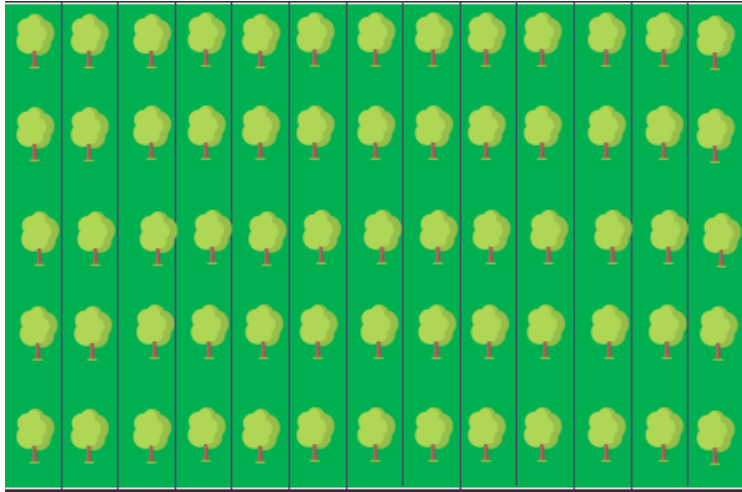


Figure 53: Line to Line Irrigation

9.3 Water Pump Actuation

We need to analyze the data and based on that we can take smart decision when to open the water pump , which part of the farm need to irrigate and the amount of water it need.

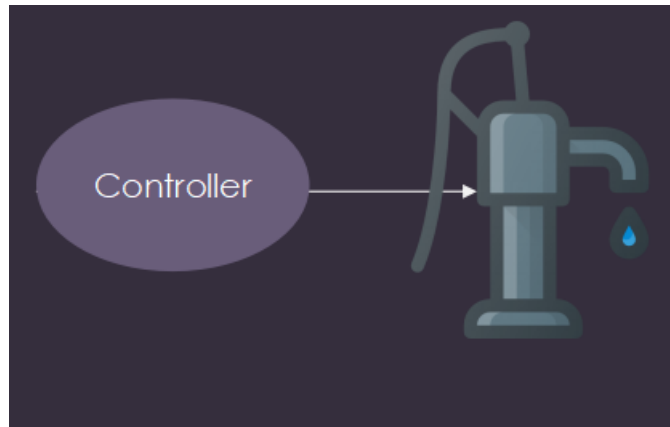


Figure 54: Water Pump

9.4 Disease Detection with ML

- Using collected data to predict possible diseases.
- Predict how crops react to different environmental conditions.

9.5 Developing Algorithm Application

- Enhancing the front end of the Machine learning simulator.
- Connecting app to server.

9.6 Access point

We have 2 options to make access point to establish wi-fi which we use to send the readings to the web server.

9.6.1 Mi-Fi

It is easy way to use and wireless module so we can put it away from the node like put it In the control room.



9.6.2 GPRS Module

Its features are:

- Require serial connection with client node.
- Need its power source to be in the central node.
- Use AT commands to send the data.

So our target is to use it as access point as its power consumption is good.



Chapter 10: References

- 1) INTERNET OF THINGS PROJECT – COMMUNICATION BETWEEN ESP8266 MODULES ([LINK](#))
- 2) A Beginner's Guide to the ESP8266 ([LINK](#))
- 3) Read Data from Thingspeak: Arduino-ESP8266-NodeMCU ([LINK](#))
- 4) ESP8266 Client-Server Wi-Fi Communication Between Two Boards (NodeMCU) ([LINK](#))
- 5) ESP32 HTTP GET and HTTP POST with Arduino IDE (JSON, URL Encoded, Text) ([LINK](#))
- 6) ESP8266 DHT11/DHT22 Temperature and Humidity Web Server with Arduino IDE ([LINK](#))
- 7) Reduce the ESP8266 power consumption? ([LINK](#))
- 8) Interface GSM SIM900A With Arduino ([LINK](#))
- 9) ESP32 Sleep Modes & Their Power Consumption ([LINK](#))
- 10) Gouravmoy Bannerjee et al. “Artificial Intelligence in Agriculture : A Literature Survey”. In: 7.3 (2018).
- 11) Robert J. McQueen et al. “Applying machine learning to agricultural data”. In: Computers and Electronics in Agriculture 12.4 (1995), pp. 275–293. issn: 01681699. doi: 10.1016/0168-1699(95)98601-9.
- 12) Osvaldo Simeone. “A brief introduction to machine learning for engineers”. In: Foundations and Trends in Signal Processing 12.3-4 (2018), pp. 200–431. issn: 19328354. doi: 10.1561/2000000102.
- 13) Lev V. Utkin. “An imprecise extension of SVM-based machine learning models”. In: Neurocomputing 331 (2019), pp. 18–32. issn: 18728286. doi: 10.1016/j.neucom.2018. 11.053.

- 14) “Measuring Bluetooth® Low Energy Power Consumption” By Sandeep Kamath & Joakim Lindh.
- 15) Instrumentation Amplifier ([LINK](#))
- 16) Precision farming solution in Egypt using the wireless sensor network technology