



**Cairo University**  
**Faculty of Engineering**  
**Electrical & Electronic Communications Department**

**Graduation project**

**Smart IOT monitoring system for agriculture with machine  
learning for early infection detection**

**Submitted by:**

Alaa Adel Araby

Loaa Ahmed Said

Mai Mohamed Abd Elhameed

Nada Abdelaal Mohamed

Nada Mohamed Magdy

Yomna Tarek Abdullah Mohamed

**Under supervision of**

Dr. Hassan Mostafa

Graduation project for Degree of

Bachelor submitted to Faculty of Engineering, Cairo University in

Electronics and Communications Engineering

Giza, Egypt

July 2018

# Thesis Outline

## Table of Contents

List of Figures.....	4
<b>1 Chapter 1: Introduction .....</b>	<b>7</b>
<b>2 Chapter 2: Architecture.....</b>	<b>12</b>
2.1 A brief explanation of the 3 layers.....	12
2.2 Block diagram of the system .....	13
<b>3 Chapter 3: Sensors .....</b>	<b>14</b>
3.1 Temperature and Humidity sensor (DHT11).....	14
3.1.1 Description.....	14
3.1.2 Specification .....	15
3.1.3 Communication Process: Serial Interface (Single-Wire Two-Way).....	15
3.1.4 Connection with Node MCU .....	17
3.2 Soil Moisture sensor (YL-69).....	19
3.2.1 Description.....	19
3.2.2 Features.....	20
3.2.3 Specifications .....	20
3.2.4 How does it work? .....	20
3.2.5 Connection it to Node MCU .....	21
<b>4 Chapter 4: Machine Learning.....</b>	<b>23</b>
4.1 Introduction.....	23
4.2 Composition.....	24
4.2.1 Datasets .....	24
4.2.2 Algorithm.....	26
4.3 How does Machine Learning work?.....	26
4.3.1 Gradient Descent vs Gradient Ascent .....	27
4.3.2 Cost Function .....	27
4.4 List of common machine learning algorithms and models.....	29
Here is the list of commonly used machine learning algorithms. These algorithms can be applied to almost any data problem:.....	29
4.4.1 Linear Regression .....	29
4.4.2 Logistic Regression.....	31
4.4.3 SVM (Support Vector Machine).....	33
4.4.4 KNN (k- Nearest Neighbors) .....	35
4.4.5 K-Means.....	36
4.5 Machine learning application .....	37
4.5.1 Difference between Classification and Regression in Machine Learning: .....	38
4.5.2 Classification Predictive Modeling .....	38
4.5.3 Regression Predictive Modeling .....	39
4.5.4 In conclusion: Classification vs Regression.....	40
4.6 The models chosen to be used in the scope of the project.....	41
4.6.1 General Procedures: .....	41
4.6.2 Dataset: .....	41
4.7 Machine learning in practice .....	45

<b>4.8</b>	<b>Results .....</b>	<b>49</b>
<b>5</b>	<b>Chapter 5: Gateway .....</b>	<b>49</b>
<b>5.1</b>	<b>IOT concept .....</b>	<b>49</b>
<b>5.2</b>	<b>MQTT protocol .....</b>	<b>51</b>
5.2.1	Overview .....	51
5.2.2	MQTT architecture .....	52
5.2.3	Topics and wild cards: .....	54
5.2.4	MQTT connection establishment: .....	56
5.2.5	Features of MQTT protocol: .....	57
5.2.6	MQTT implementation: .....	60
5.2.7	Python script on raspberry pi: .....	62
<b>5.3</b>	<b>GSM/GPRS Module.....</b>	<b>64</b>
5.3.1	Introduction.....	64
5.3.2	Why using GSM/GPRS?.....	65
5.3.3	Main Features of GSM/GPRS: .....	65
5.3.4	GSM Modem .....	65
5.3.5	SIM900 GPRS Arduino Shield .....	66
<b>6</b>	<b>Chapter 6: Node MCU .....</b>	<b>74</b>
<b>6.1</b>	<b>Node MCU developed on ESP8266: .....</b>	<b>74</b>
<b>6.2</b>	<b>ESP8266 Specifications: .....</b>	<b>74</b>
<b>6.3</b>	<b>Node MCU for IOT applications:.....</b>	<b>75</b>
<b>6.4</b>	<b>Node MCU versions:.....</b>	<b>75</b>
6.4.1	1 <sup>st</sup> generation (0.9/v1):.....	75
6.4.2	2 <sup>nd</sup> generation (1.0/v2):.....	76
6.4.3	2 <sup>nd</sup> generation (1.0/v3):.....	77
<b>6.5</b>	<b>x.5 selection of Node MCU version 3: .....</b>	<b>78</b>
<b>6.6</b>	<b>Programming Node MCU:.....</b>	<b>78</b>
6.6.1	MQTT and Wi-Fi related functions: .....	79
<b>6.7</b>	<b>Sensors related functions: .....</b>	<b>79</b>
<b>6.8</b>	<b>flow chart illustration: .....</b>	<b>81</b>
<b>7</b>	<b>Chapter 7: Application Layer .....</b>	<b>82</b>
<b>7.1</b>	<b>Logical Gateway:.....</b>	<b>82</b>
<b>7.2</b>	<b>Physical Gateway:.....</b>	<b>82</b>
<b>7.3</b>	<b>Cloud gateway .....</b>	<b>83</b>
<b>7.4</b>	<b>IoT Modular Gateway:.....</b>	<b>84</b>
<b>7.5</b>	<b>Architectural Overview .....</b>	<b>86</b>
<b>7.6</b>	<b>The Gateway Software.....</b>	<b>86</b>
<b>7.7</b>	<b>Sensor Consumers: .....</b>	<b>87</b>
<b>7.8</b>	<b>Gateway Data Transfer:.....</b>	<b>87</b>
<b>7.9</b>	<b>The IoT Expanding Connectivity:.....</b>	<b>88</b>
6.1	The new Edge Layer: .....	88
6.2	Conclusion .....	89
<b>8</b>	<b>Chapter 8: Web application .....</b>	<b>90</b>
<b>8.1</b>	<b>Introduction to Our web Application: .....</b>	<b>90</b>
<b>8.2</b>	<b>Prerequisites for web Development: .....</b>	<b>90</b>
<b>8.3</b>	<b>Implementation: .....</b>	<b>90</b>
8.3.1	Getting website alive on web .....	90
8.3.2	Connection Webpage .....	92
8.3.3	Login Webpage .....	93

8.3.4	Data and Advice Webpage.....	94
<b>8.4</b>	<b>Graphical user interface:.....</b>	<b>94</b>
<b>9</b>	<b>Chapter 9: Conclusion and future work .....</b>	<b>98</b>
<b>10</b>	<b>Chapter 10: References.....</b>	<b>100</b>

## List of Figures

Figure 3-1	Temperature and Humidity sensor. ....	14
Figure 3-2	MCU pulling down the data pin down then up(start signal) to initiate communication .....	16
Figure 3-3	DHT pulling down the data pin down then up as Response on receiving start signal .....	16
Figure 3-4	Data bit “0” .....	17
Figure 3-5	Data bit “1” .....	17
Figure 3-6	Connection of Temperature and Humidity sensor with Node MCU.....	18
Figure 3-7	Soil Sensor .....	19
Figure 3-8	YL-96 sensor parts .....	20
Figure 3-9	Soil Moisture sensor working Analogy.....	21
Figure 3-10	connection of Moisture sensor with Node MCU.....	22
Figure 4-1	Board Game dataset example .....	25
Figure 4-2	Gradient Descent Algorithm .....	27
4-5	SVM .....	33
Figure 4-6	Daily Blight Observation Training Dataset.....	42
Figure 4-7	Real Life Data measured of the field.....	43
Figure 4-8	Data modification to fit the ML Algorithm.....	43
Figure 4-9	Disease Severity Training Data set.....	44
Figure 5-1:	Forecast for the number of connected devices till 2050.....	51
Figure 5-2:	MQTT architecture .....	53
Figure 5-3:	Clients (B) and (C) subscribe .....	53
Figure 5-4:	Client (A) publishes a message .....	54
<b>Figure 5-5:</b>	<b>Topic format in MQTT.....</b>	<b>54</b>

<b>Figure 5-6: The use of (+) wild card in topic format.....</b>	<b>55</b>
Figure 5-7: Illustration of valid wildcards in single level .....	55
Figure 5-8: The use of (#) wild card in topic format.....	55
Figure 5-9: Illustration of valid wildcards in multi- level.....	56
Figure 5-10:Message transfer between client and broker .....	57
Figure 5-11: Control information for QoS 0.....	58
Figure 5-12: Control information for QoS 1 .....	58
Figure 5-13:Control information for QoS 2.....	59
Figure 6-7 Pin layout of v3 Node MCU .....	78
Figure 8-1:FileZilla Client Communicating with server.....	91
Figure 8-2: phpMyAdmin to handle tables in database on website .....	92
Figure 3:Login page example .....	93
Figure 8-4:home page .....	94
Figure 8-5:home page services .....	95
Figure 8-6: Login Form .....	95
Figure 8-7: Error Message .....	96
Figure 8-8:hyperlinks redirects to data and advice .....	96
Figure 8-9: sensors data table.....	97
Figure 8-10: Advice table .....	97

## **Acknowledgment**

This project would not have been possible without the support and guidance of our advisor, Dr.Hassan Mostafa. So, many thanks for him and every one who has helped us with his effort or time from our last year colleagues or other external engineers.

# 1 Chapter 1: Introduction

Agriculture is an important sector of the Egyptian economy. It contributes nearly one-seventh of the GDP, employs roughly one-fourth of the labor force, and provides the country, through agricultural exports, with an important part of its foreign exchange. The rapid increase in Egypt's population forms a burden on the agricultural sector, as the demand is increasing while the amount of production is not keeping in pace with it. This shortage in production is due to some problems that face the agricultural sector due to adopting the old-fashioned methods in the farming, irrigation and monitoring of the crop's status. These methods mostly depend on the human labor which is of high cost, inaccurate and not time efficient. However, using the new technological methods, as Internet of things (IoT) and machine learning, these processes can be automated which reduces labor requirement as well as cost and gives better performance. In this thesis, a proposed system, combining these two technologies, is presented to automatically measure the status of potato crop, performs predictive analysis on the collected data to detect the probability of infection with the late blight disease specifically or not and finally it uploads the measurements and the result of analysis to the website to be remotely monitored by the farmer. This approach is known as the precision agriculture.

Precision agriculture is a farming management concept based on observing, measuring and responding to inter and intra-field variability in crops. By collecting real-time data on weather, soil and air quality, crop maturity and other factors, predictive analytics can be used to make better decisions for the future.

Potato production in Egypt ranks the first in export and the second in acreage among the vegetable crops, occupying about **15%** of the total vegetable cultivation area. The cultivation of the potato was expanded to newly reclaimed areas. Total Potato production area in Egypt was **283,000 feddans** yielded about **2.5 million tons** annually. So, a disease like blight late (Phytophthora), that has been a great threat in the past years on our production, is a very critical issue that damages a huge percentage of the crop and in many cases, it can result in famine like what happened in Ireland. Crop losses were most

severe in Ireland, since the notorious 'Potato Famine' caused poverty, death of one million people and emigration of another **1.5 million**.

In agriculture field, there is a disease called late blight which can affect potato. Late blight of potato is identified by black/brown lesions (Figures 1-1, 1-2) on leaves and stems that may be small at first and appear water-soaked or have chlorotic borders, but soon expand rapidly and become necrotic. In humid conditions, *Phytophthora infestans* (pathogen of disease) produces sporangia and sporangiophores on the surface of infected tissue. This sporulation results in a visible white growth at the leading edge of lesions on abaxial (lower) surfaces of leaves.

As many lesions accumulate, the entire plant can be destroyed in only a few days after the first lesions are observed (Figure 1-3).



**Figure 1-1,2: Late blight lesion on a potato leaf.**



**Figure 1-3: late blight destroyed the entire plant.**



The infection usually starts with the deposition of a sporangium on a leaf or stem. The sporangium germinates either directly (18-24°C) or indirectly via zoospores (8-18°C).

Historically, late blight is best known as the cause of the Irish potato famine of the 1840s, which resulted in the death or emigration of over 2 million people from Ireland.

Although late blight disease has been present in Egypt at least since 1941 and it has been a continuous threat to potato cultivations.

Potato production in Egypt ranks the first in export and the second in acreage among the vegetable crops, occupying about 15% of the total vegetable cultivation area. The cultivation of the potato was expanded to newly reclaimed areas. Total Potato production area in Egypt was 283,000 feddans yielded about 2.5 million tons annually.

On the other hand, Internet of Things (IoT) and Machine Learning are new technologies which can help our country to reduce the potato late blight effects in agriculture sector.

The Internet of Things (IoT) is the concept of connecting any device - so long as it has an on/off switch - to the Internet and to other connected devices. The IoT is a giant network of connected things and people, all of which collect and share data about the way they are used and about the environment around them, that includes an extraordinary number of objects of all shapes and sizes.

Devices and objects with built in sensors are connected to an Internet of Things platform, which integrates data from the different devices and applies analytics to share the most valuable information with applications built to address specific needs.

These powerful IoT platforms can pinpoint exactly what information is useful and what can safely be ignored. This information can be used to detect patterns, make recommendations, and detect possible problems before they occur.

The benefits that agriculture can get from applying IoT is two pronged; firstly, farmers can optimize the use of inputs and also farmers can decrease production costs. Furthermore, the other benefits were, saves cost by effectively using inputs, Better monitoring of crops,

avoiding crop losses through disease or adverse weather, help in optimizing water use, better planning of farm activities.

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Machine learning algorithms are often categorized as supervised or unsupervised, Supervised machine learning algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

In contrast, unsupervised machine learning algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

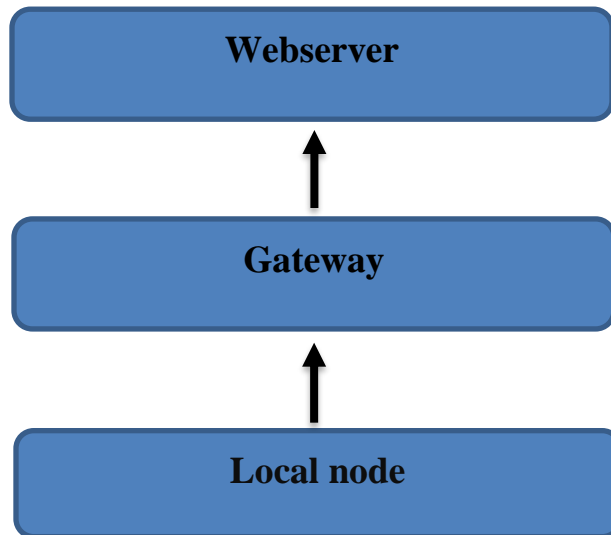
In this project, Internet of Things (IoT) and Machine Learning are integrated to solve the late blight problem, sensors are connected in the field to get the data from the soil and the environment, sending these data to the gateway, store it to a data base and uploading these data to the platform to be easily monitored by the farmer.

A machine learning algorithm which is called regression model is used to use data collected by sensors and predict if the late blight disease will infect the potato yield within the coming days, depending on this prediction, the algorithm will send an advice to the farmer on the platform and tell him what the best action to take regarding irrigation or even using pesticides in order to prevent the infection as much as possible.

The organization of the thesis is as follows: Chapter 2 describes the agriculture problem. Chapter 3 presents the architecture of the system. Chapter 4 describes the sensors that are used in the system. Chapter 5 presents the machine learning concept and the implemented algorithm. Chapter 6 presents the gateway of the system. Chapter 7 explains the cloud concept. Chapter 8 presents the database and the website. An analysis of future work and conclusion are presented in section 9.

## 2 Chapter 2: Architecture

The architecture of our proposed system consists of three layers. These layers are the local node, the gateway and the application layer.



This system is an IOT monitoring system for agriculture with machine learning for early infection detection. It monitors the climatological conditions as well as the soil conditions in the field and then perform analysis on these measurements to predict if an infection will occur in the crop due to these field conditions or not. The field's measurements as well as an advice to the farmer based on the machine learning analysis are stored in a database and uploaded to a website to be remotely monitored by the farmer.

### 2.1 A brief explanation of the 3 layers

- **Local node:**

It consists of the Node MCU - an IOT platform with esp8266 WIFI module embedded in it - and attached to it the sensors that measures the conditions of the field. These readings are sent to the gateway through WIFI using esp8266 module.

The sensors in the system measures three main conditions of the field, the environment temperature, humidity and the soil moisture.

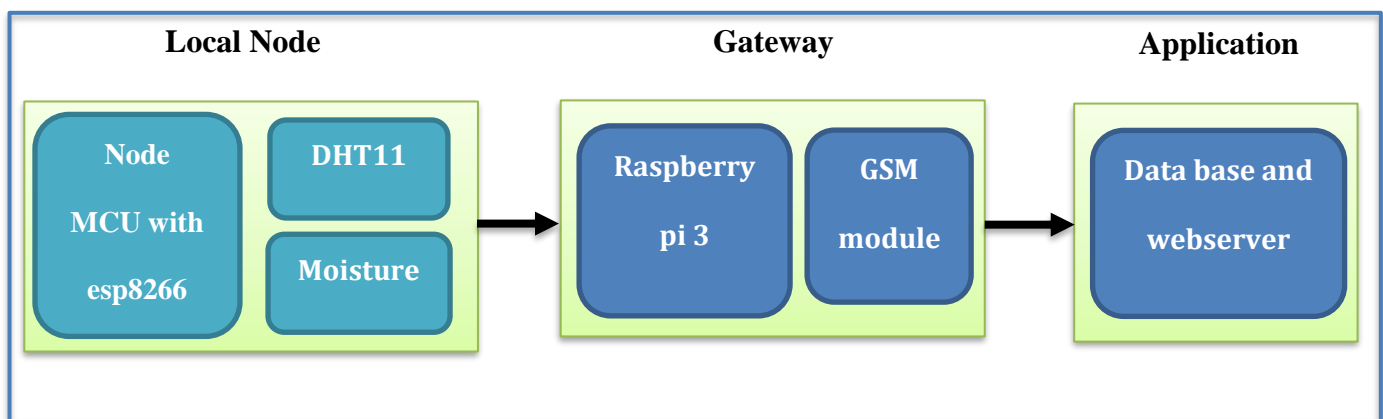
- **Gateway:**

It consists of the raspberry pi and the GSM modem. Raspberry pi 3 is used to receive the data sent from the Node MCU, then periodically update the database with the new readings and then upload it to the webserver using the GSM modem. Also, the raspberry pi hosts the machine learning Algorithm which is processing on the collected data and then the result is uploaded to the webserver in form of an advice message or sent as a SMS to the user number.

- **Application:**

It consists of the database and the webserver. The database is used to store every reading sent by the Node MCU and these readings are uploaded to the webserver to be accessed any time by the user.

## 2.2 Block diagram of the system



### 3 Chapter 3: Sensors

The sensors are used to collect data from the soil and the environment to be able to predict how they can affect the yield, there are two sensors that are used, first one is temperature and humidity sensor “DHT11” and the other sensor is soil moisture sensor “YL-69”.

#### 3.1 Temperature and Humidity sensor (DHT11)

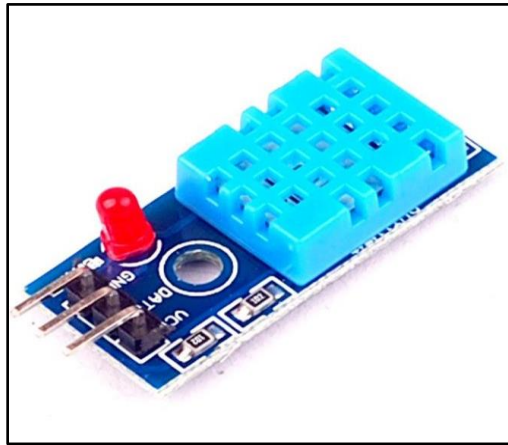


Figure 3-1 Temperature and Humidity sensor.

##### 3.1.1 Description

This DHT11 Temperature and Humidity Sensor features a calibrated digital signal output with the temperature and humidity sensor capability. It is integrated with a high-performance 8-bit micro-controller. Its technology ensures the high reliability and excellent long-term stability. This sensor includes a resistive element and a sensor for wet NTC temperature measuring devices. It has excellent quality, fast response, anti-interference ability and high performance.

Each DHT11 sensors features extremely accurate calibration of humidity calibration chamber. The calibration coefficients stored in the OTP program memory, internal sensors detect signals in the process, we should call these calibration coefficients. The single-wire serial interface system is integrated to become quick and easy. Small size, low power, signal transmission distance up to 20 meters, enabling a variety of applications

and even the most demanding ones. The product is 4-pin single row pin package. Convenient connection, special packages can be provided according to users need.

### **3.1.2 Specification**

- Supply Voltage: +5 V
- Temperature range :0-50 °C error of  $\pm 2$  °C
- Humidity :20-90% RH  $\pm 5\%$  RH error
- Interface: Digital

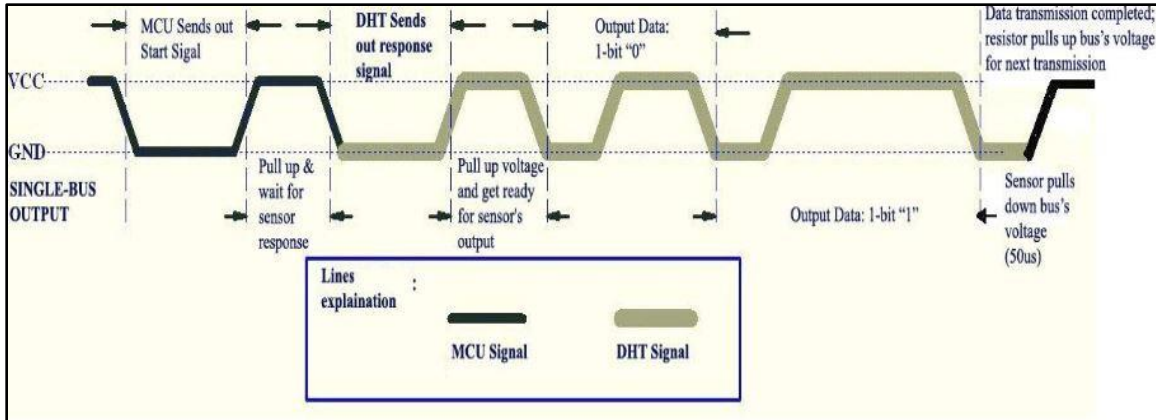
### **3.1.3 Communication Process: Serial Interface (Single-Wire Two-Way)**

Single bus data format is used for the communication and synchronization between micro controller unit “MCU” and the DHT11 sensor. Each communication process will last about 4ms.

The data is transmitted in this format:

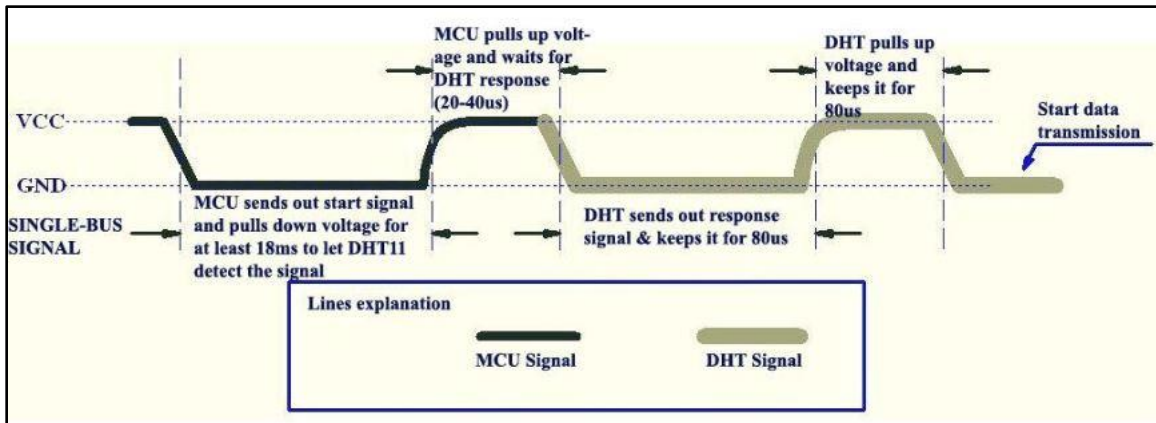
- 8bit integral RH data +
- 8bit decimal RH data +
- 8bit integral T data +
- 8bit decimal T data +
- 8bit check sum.

If the data transmission is correct, the check sum should equals to the lower 8bit of the result of “8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data”.



**Figure 3-2 MCU pulling down the data pin down then up(start signal) to initiate communication**

The default status of the DATA pin is high. When the communication between MCU and DHT11 starts, MCU will pull down the DATA pin for least 18ms. This is called “Start Signal” and it is to ensure DHT11 has detected the signal from MCU. Then MCU will pull up DATA pin for 20-40us to wait for DHT11’s response like in (Figure 3-2).



**Figure 3-3 DHT pulling down the data pin down then up as Response on receiving start signal**

Once DHT11 detects the start signal, it will pull down the DATA pin as “Response Signal”, which will last 80us. Then DHT11 will pull up the DATA pin for 80us and prepare for data sending.

During the data transition, every bit of data begins with the 50us low-voltage-level and ends with a high-voltage-level signal. The length of the high-voltage-level signal decides whether the bit is “0” or “1”, **Data bit “0”** has **26-28us** high-voltage length, While **Data bit “1”** has **70us** high-voltage length, illustrated in Figures(3-4), ()



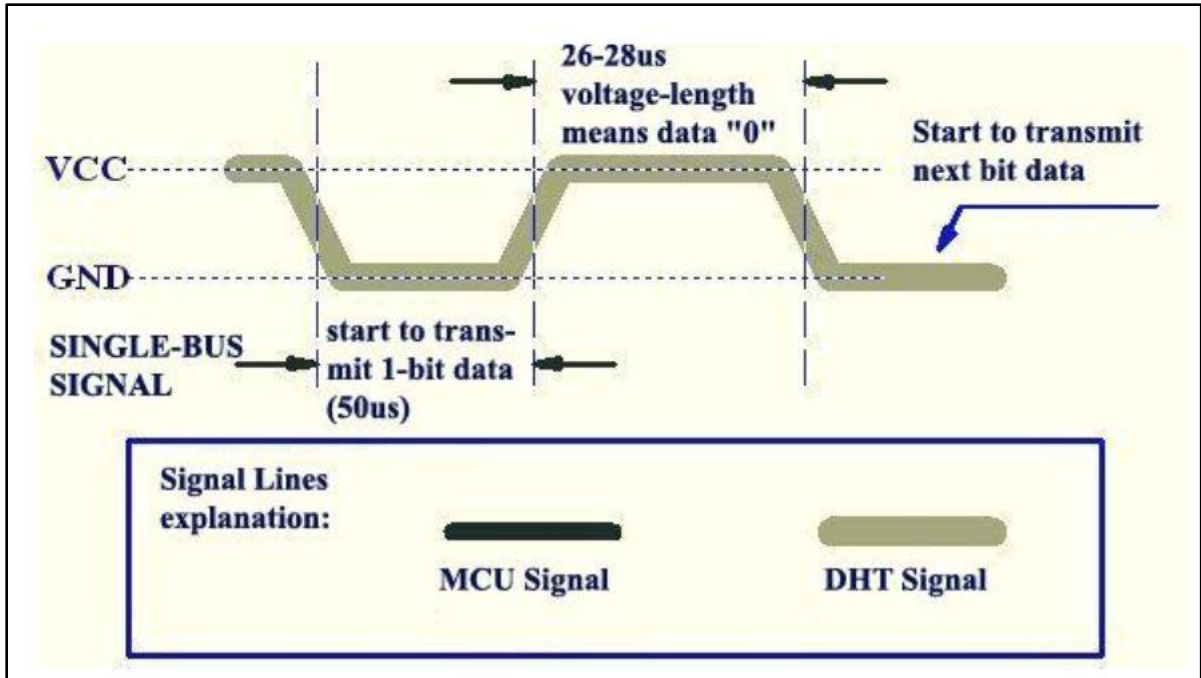


Figure 3-4 Data bit "0"

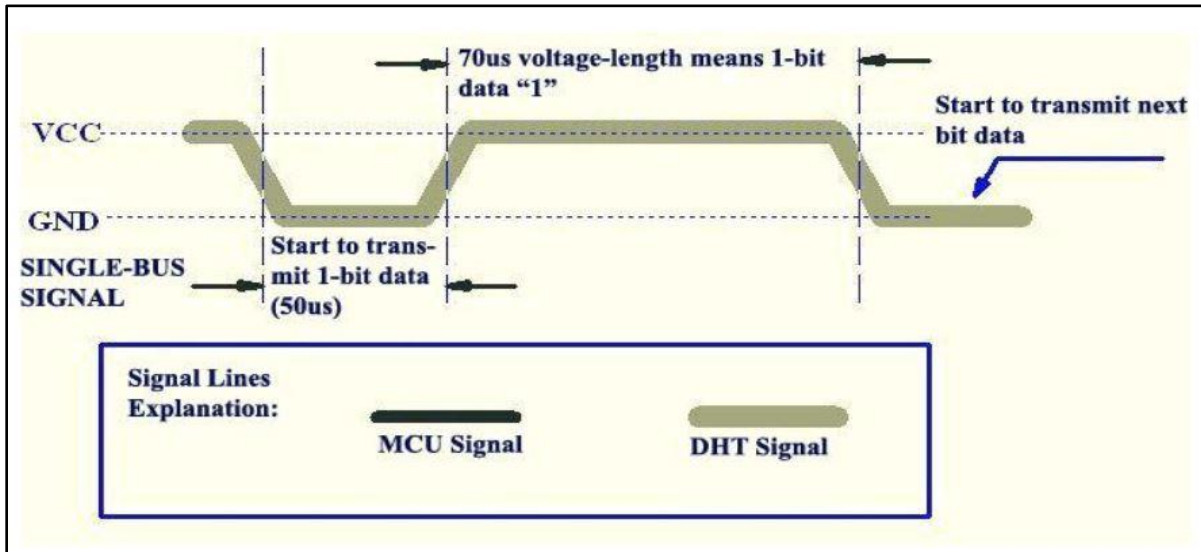


Figure 3-5 Data bit "1"

### 3.1.4 Connection with Node MCU

Besides the VCC and GND, we just need to use pin on the Node MCU to make use of the DHT11 module.

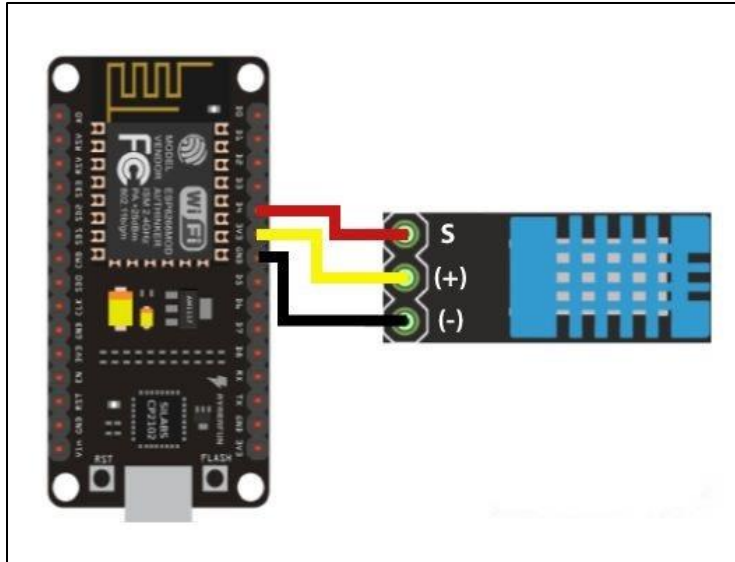


Figure 3-6 Connection of Temperature and Humidity sensor with Node MCU.

Node MCU	DHT11 Module
3.3v	VCC(V)
GND	GND(G)
D1	DATA(S)

## 3.2 Soil Moisture sensor (YL-69)

### 3.2.1 Description

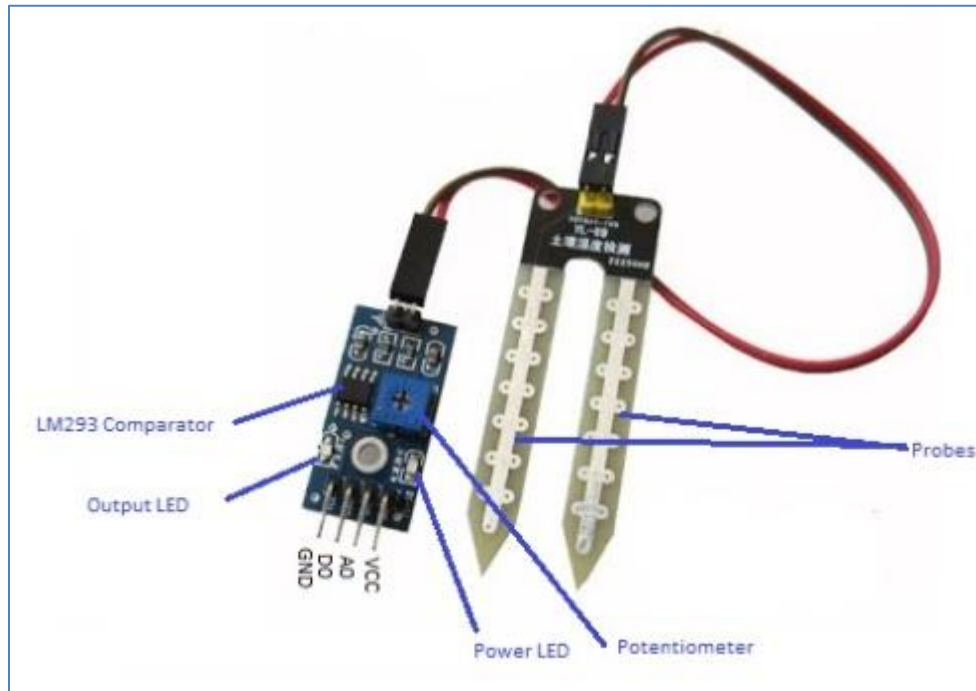


Figure 3-7 Soil Sensor

The soil moisture sensor or the hygrometer is usually used to detect the humidity of the soil. So, it is perfect to build an automatic watering system or to monitor the soil moisture of the plants, the **sensor** is set up by **two** pieces: the **electronic board** (at the Left), and the probe with two pads, that detects the water content (at the Right), the sensor has a **built-in potentiometer** for sensitivity adjustment of the digital output (DO), a power LED and a digital output LED, as seen in the Figure (3-8).

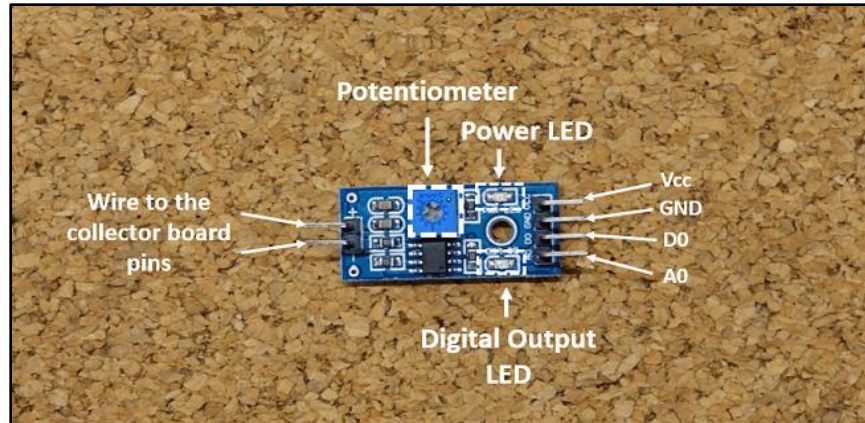


Figure 3-8 YL-96 sensor parts

### 3.2.2 Features

- Soil moisture sensor based on soil resistivity measurement.
- Easy to use.
- 2.0cmX6.0cm grove module.

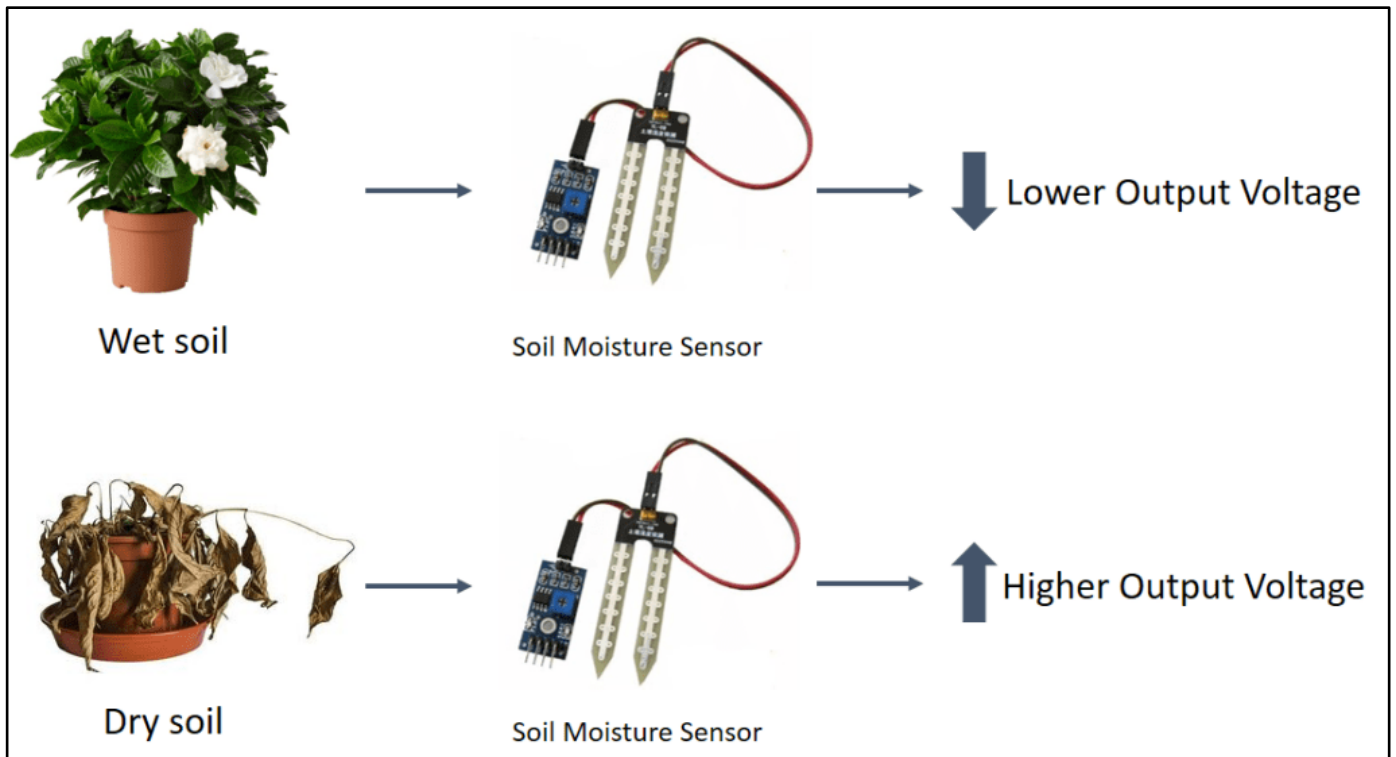
### 3.2.3 Specifications

- Operating voltage: DC 3.3V - 5V
- Output voltage signal: 0 ~ 4.2V
- Current: 35mA
- LED: Power indicator (Red) and Digital switching output indicator (Green)

### 3.2.4 How does it work?

The voltage that the sensor outputs changes accordingly to the water content in the soil.  
When the soil is:

- **Wet:** the output voltage decreases.
- **Dry:** the output voltage increases.



**Figure 3-9 Soil Moisture sensor working Analogy**

The Module also contains a potentiometer which will set the threshold value and then this threshold value will be compared by LM393 comparator. The output LED will light up and down according to this threshold value.

Analog mode is used in the connections, to connect the sensor in the analog mode, we will need to use the analog output of the sensor. When taking the analog output from the soil moisture sensor, the sensor gives us the value from 0-1023. The moisture is measured in percentage, so we will map these values from 0 -100 and then we will show these values on the serial monitor.

It can be further set different ranges of the moisture values and turn on or off the water pump according to it.

### **3.2.5 Connection it to Node MCU**

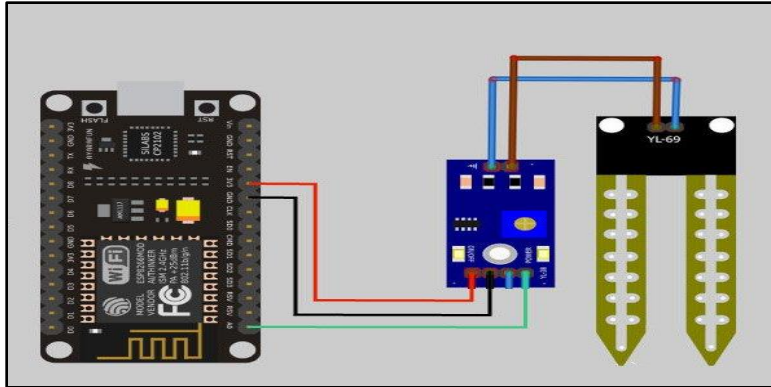


Figure 3-10 connection of Moisture sensor with Node MCU.

Node MCU	YL-96 sensor
3.3 V	VCC
GND	GND
A0	DATA

## 4 Chapter 4: Machine Learning

### 4.1 Introduction

**Arthur Samuel**, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1952 when he wrote the first computer learning program. The program was a game of checkers, and the IBM computer improved at the game the more it played, studying which moves made up winning strategies and incorporating those moves into its program. **Frank Rosenblatt**, designed the first neural network for computers (the perceptron), which simulate the thought processes of the human brain at 1957.

So what is Machine learning? **Machine learning** is a subfield of **artificial intelligence** (AI) which is a simulation of human intelligence processes by machines, especially computer systems. These processes include learning, reasoning and self-correction, and it is defined to be the science of getting computers to act without being explicitly programmed. In the past decade, machine learning has introduced self-driving cars, practical speech recognition, effective web search, and improved understanding of the human genome.

The difference between **Machine learning** and **data mining** is that: Machine learning focuses on prediction, based on previously known properties learned from the training data, but data mining focuses on the discovery of unknown properties of the data in other words the goal is the extraction of patterns, features and knowledge from large amounts of data. Data mining uses many machine learning methods, and machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy.

Machine learning has strong correlation with the meaning of optimization: many learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre-assigned labels of a set of examples). The difference between the two fields arises from the goal of

generalization: while optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples.

How does **Machine learning** differ from **Traditional computing**? In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or solve a problem. Instead Machine learning algorithms allow computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range, which allows computers to automate decision-making processes based on only the data inputs.

To summarize the introduction; Machine Learning can best be understood through some progressive lenses:

1. Machine Learning is predicting things, usually based on what they've done in the past, like the checkers program.
2. Machine Learning tries to find relationships in the data that can help to predict what will happen next, which is more like simple regression in math.
3. Machine Learning uses statistical methods to predict the value of a target variable using a set of input data.

## 4.2 Composition

Any type of Machine Learning problem can be broken down into 2 major parts:

1. The datasets
2. The algorithms

### 4.2.1 Datasets

Datasets may be images or csv files which is typically organized as rows, with columns representing features of the data. What Machine Learning methods will do is try to find **patterns** in this data. Any number of real patterns can actually exist within the data, for higher accuracy, larger samples of data is needed to be used, There are many available datasets online: for example, **Mnist** which contains handwritten numbers in many formats for train and test also there are built in datasets in used tools as **iris** datasets in Python and the dataset from **BoardGameGeek**, which contains data on different board games; these



data were used and collected by gaming companies from surveys for example; one of its purposes is to help gaming companies to define what are the games trending the market, their type whether adventurous, war, etc.. , and the age target of their consumers, then this information was scraped into csv format and made available to be downloaded. The dataset contains several data points about each board game. Here's a list of the interesting ones:

Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
owned	mechanic	age	image_url	num_votes	geek_rating	avg_rating	year	max_time	min_time	avg_time	max_players	min_players	names	game_id	bgg_url	rank
25928	Action / Mc	12	<a href="https://cf.g">https://cf.g</a>	15376	8.61858	8.98893	2017	120	60	120	4	1	Gloomhaven	174430	<a href="https://boa">https://boa</a>	1
41605	Action Poir	13	<a href="https://cf.g">https://cf.g</a>	26063	8.50163	8.6614	2015	60	60	60	4	2	Pandemic Leg	161936	<a href="https://boa">https://boa</a>	2
15848	Action Poir	14	<a href="https://cf.g">https://cf.g</a>	12352	8.30183	8.60673	2015	240	180	240	4	2	Through the Ag	182028	<a href="https://boa">https://boa</a>	3
33340	Card Draft	12	<a href="https://cf.g">https://cf.g</a>	26004	8.19914	8.38461	2016	120	120	120	5	1	Terraforming M	167791	<a href="https://boa">https://boa</a>	4
42952	Area Contr	13	<a href="https://cf.g">https://cf.g</a>	31301	8.19787	8.33954	2005	180	120	180	2	2	Twilight Struggl	12333	<a href="https://boa">https://boa</a>	5
20682	Area Contr	14	<a href="https://cf.g">https://cf.g</a>	13336	8.16545	8.47439	2016	240	180	240	4	2	Star Wars: Rek	187645	<a href="https://boa">https://boa</a>	6
38279	Area Contr	14	<a href="https://cf.g">https://cf.g</a>	29219	8.108	8.29158	2016	115	90	115	5	1	Scythe	169786	<a href="https://boa">https://boa</a>	7
32637	Area Contr	12	<a href="https://cf.g">https://cf.g</a>	29309	8.09296	8.23703	2012	150	60	150	5	2	Terra Mystica	120677	<a href="https://boa">https://boa</a>	8
16869	Deck / Poc	12	<a href="https://cf.g">https://cf.g</a>	13521	8.04047	8.28574	2016	150	75	150	4	2	Great Western	193738	<a href="https://boa">https://boa</a>	9
12543	Co-operati	14	<a href="https://cf.g">https://cf.g</a>	6057	8.02458	8.67228	2017	1000	5	1000	4	1	The 7th Contine	180263	<a href="https://boa">https://boa</a>	10
6243	Modular Bc	12	<a href="https://cf.g">https://cf.g</a>	4860	8.00732	8.68217	2017	150	60	150	4	1	Gaia Project	220308	<a href="https://boa">https://boa</a>	11
43992	Dice Rollin	12	<a href="https://cf.g">https://cf.g</a>	31613	8.00679	8.12383	2011	90	30	90	4	2	The Castles of	84876	<a href="https://boa">https://boa</a>	12
49800	Card Draft	10	<a href="https://cf.g">https://cf.g</a>	32165	8.00627	8.14559	2015	30	30	30	2	2	7 Wonders Duk	173346	<a href="https://boa">https://boa</a>	13
13216	Area Contr	13	<a href="https://cf.g">https://cf.g</a>	8217	7.98727	8.37781	2012	180	150	180	4	2	War of the Ring	115746	<a href="https://boa">https://boa</a>	14

Figure 4-1 Board Game dataset example

- **names** -- name of the board game.

The following labels can be chosen to be the inputs or the features of the dataset:

- **owned** – the number of users downloaded the game or registered on the game official website.
- **minplaytime** -- the minimum playing time (given by the manufacturer).
- **maxplaytime** -- the maximum playing time (given by the manufacturer).
- **users\_rated** -- the number of users who rated the game.
- **average\_rating** -- the average rating given to the game by users. (0-10)
- **Weight** is a subjective measure that is made up by BoardGameGeek. It's how "deep" or involved a game is.
- **Category** -- Adventurous, Fighting, card games, etc...

The following label can be chosen to be the output of the dataset which is needed to be predicted for future games release.

- **Rank** – the most popular game.

## 4.2.2 Algorithm

In Machine Learning, the algorithm is just the method that will be used to find the relationships within the data. Algorithms can be complex or simple, big or small, or any permutation of things: but at the core, they're just ways of figuring out what drives the changes you're trying to predict. So there are different types of algorithms which can help to achieve **different goals**, for example If the **goal** is to explain the relationships within data like vocals found in human speech or **predicating the forecast**, a simple algorithm like **Linear Regression** is probably a good bet, but if the **goal** is the **accuracy**, **neural networks** can achieve higher accuracy rates but in exchange of the training time, this is called the **accuracy- tradeoff**, So, in order to choose which algorithm is to be used that depends on some factors like specifying the problem clearly, studying the tradeoffs and studying the required results.

For all algorithms the dataset is divided into two majors, **train data** and **test data** the train data is used at first to calculate the Algorithm or the model parameters which then will define the relation between the input and output, then the test data is used to measure how good the algorithm parameters in predicting correct values for outputs.

## 4.3 How does Machine Learning work?

The concepts behind how ML (Machine Learning) works are actually very simple, even if the underlying algorithms can get complex. The majority of ML algorithms use one method to find the relationships required for solving any problem, it is called **Gradient Descent**; the method is very simple, as it starts at some random point and tries to improve the predictions by resetting the position of this random point with every iteration.

The objective of a ML model is to find parameters of model or in relations, weights or a structure that minimizes the **cost function**. Approach that best illustrates statistical learning; minimizing a cost function. **So models learn by minimizing a cost function**, but how then the cost function is minimized, by using **gradient descent** method.

### 4.3.1 Gradient Descent vs Gradient Ascent

Gradient is another word for "slope". The higher the gradient of a graph at a point, the steeper the line is at that point. A negative gradient means that the line slopes downwards. Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. If instead one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as gradient ascent, **Gradient descent** is also **known** as **steepest descent**. However, gradient descent should not be confused with the method of steepest descent for approximating integrals.

Gradient decent function is implemented in python.

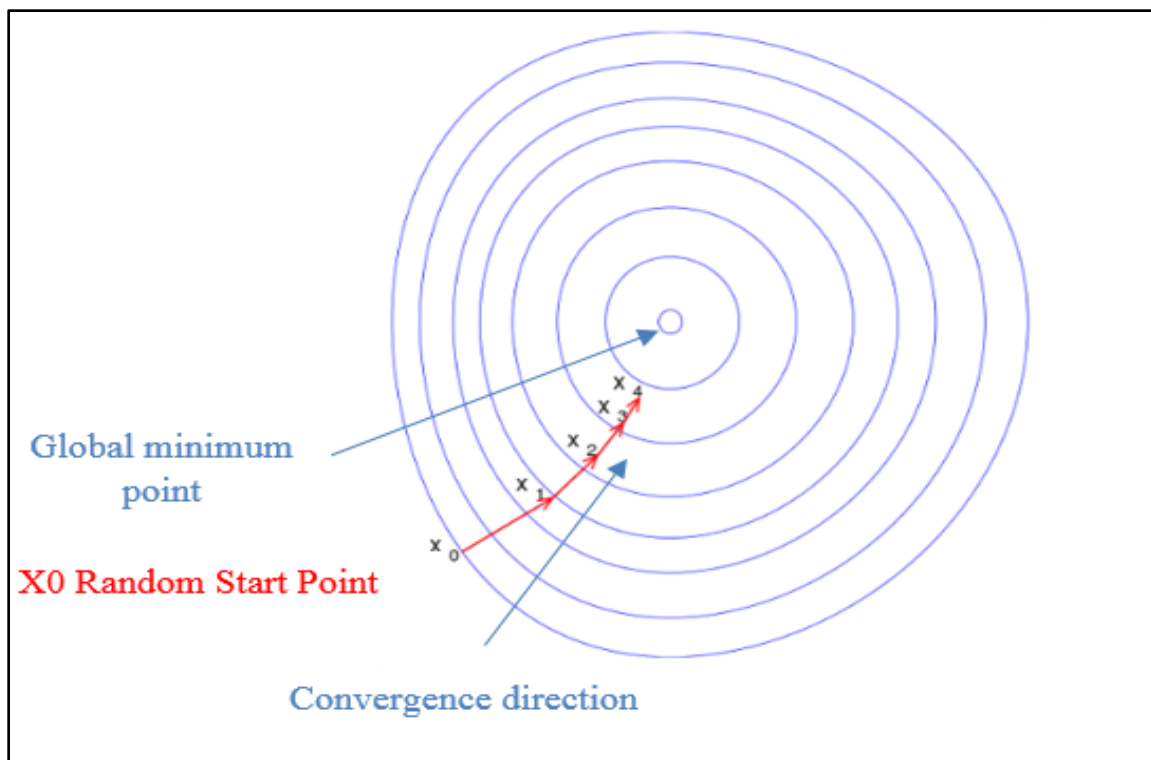
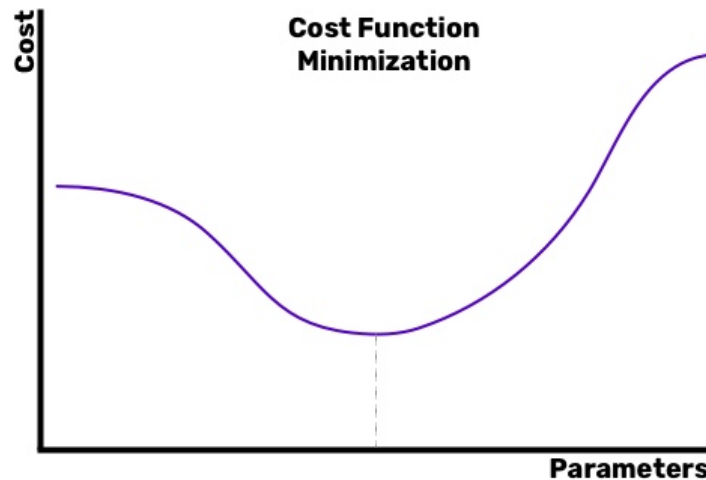


Figure 4-2 Gradient Descent Algorithm

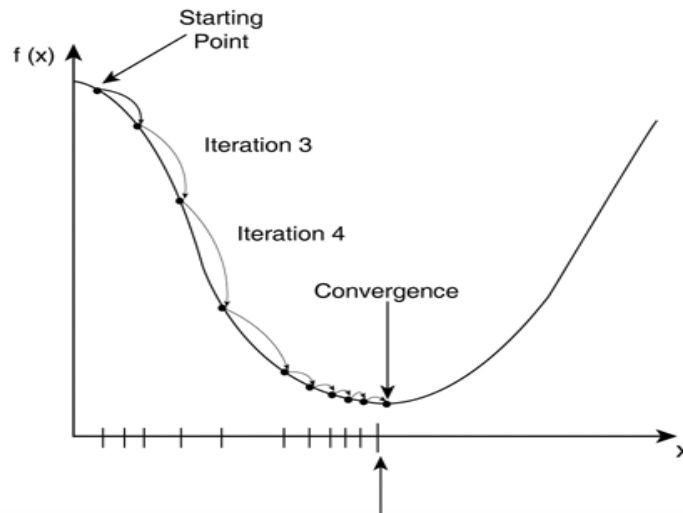
### 4.3.2 Cost Function

In ML, cost functions are used to estimate how badly models are performing. Simply, a cost function is a measure of how wrong the model is in terms of its ability to estimate the accurate relationship between X (features or inputs) and Y (Outputs). This cost function (it may be considered to be the error) can be estimated by iteratively running the model to compare estimated predictions from test input samples against the known values of the test outputs(y).

The objective of a ML model, therefore, is to find parameters, weights or a structure that minimizes the cost function. Approach that best illustrates statistical learning; minimizing a cost function. **So models learn by minimizing a cost function**, but how then the cost function is minimized, by using gradient descent method described in previous section.



As function is defined gradient descent is performed to locate global minimum iteratively.



#### 4.4 List of common machine learning algorithms and models

Here is the list of commonly used machine learning algorithms.

These algorithms can be applied to almost any data problem:

1. Linear Regression
2. Logistic Regression
3. SVM
4. KNN
5. K-Means

##### 4.4.1 Linear Regression

It is used to estimate real values (cost of houses, number of calls, total sales etc.) based on continuous variable(s). Here, we establish 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  $Y = a * X + b$ .

The best way to understand linear regression is to relive this experience of childhood. Let us say, you ask a child in fifth grade to arrange people in his class by increasing order of weight, without asking them their weights! What do you think the child will do? He / she would likely look (visually analyze) at the height and build of people and arrange them using a combination of these visible parameters. This is linear regression in real life! The

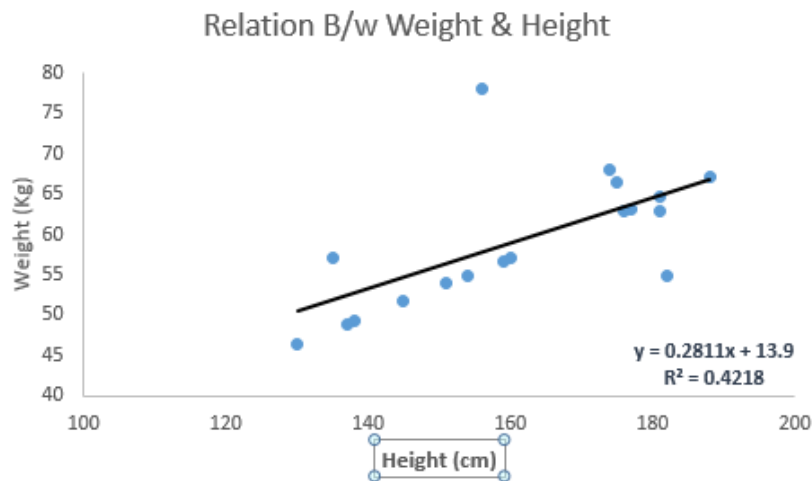
child has actually figured out that height and build would be correlated to the weight by a relationship, which looks like the equation above.

In this equation:

- Y – Dependent Variable
- a – Slope
- X – Independent variable
- b – Intercept

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

Look at the below example. Here we have identified the best fit line having linear equation  $y=0.2811x+13.9$ . Now using this equation, we can find the weight, knowing the height of a person.



### linear regression example

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. And these are known as polynomial or curvilinear regression.

### Python Code

```
#Import Library  
  
#Import other necessary libraries like pandas, numpy...
```

```

from sklearn import linear_model

#Load Train and Test datasets

#Identify feature and response variable(s) and values must be numeric and numpy arrays

x_train=input_variables_values_training_datasets

y_train=target_variables_values_training_datasets

x_test=input_variables_values_test_datasets

# Create linear regression object

linear = linear_model.LinearRegression()

# Train the model using the training sets and check score

linear.fit(x_train, y_train)

linear.score(x_train, y_train)

#Equation coefficient and Intercept

print('Coefficient: \n', linear.coef_)

print('Intercept: \n', linear.intercept_)

#Predict Output

predicted= linear.predict(x_test)

```

#### 4.4.2 Logistic Regression

It is a classification not a regression algorithm. 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. Hence, it is also known as **logit regression**. Since, it predicts the probability, its output values lies between 0 and 1 (as expected).let us try and understand this through a simple example. Let's say your friend gives you a puzzle to solve. There are only 2 outcome scenarios – either you solve it or you don't. Now imagine, that you are

being given wide range of puzzles / quizzes in an attempt to understand which subjects you are good at. The outcome to this study would be something like this – if you are given a trigonometry based tenth grade problem, you are 70% likely to solve it. On the other hand, if it is grade fifth history question, the probability of getting an answer is only 30%. This is what Logistic Regression provides you. Coming to the math, the log odds of the outcome is modeled as a linear combination of the predictor variables.

odds=  $p / (1-p)$  = probability of event occurrence / probability of not event occurrence  
 $\ln(\text{odds}) = \ln(p/(1-p))$   
 $\text{logit}(p) = \ln(p/(1-p)) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 \dots + b_kX_k$

Above,  $p$  is the probability of presence of the characteristic of interest. It chooses parameters that maximize the likelihood of observing the sample values rather than that minimize the sum of squared errors (like in ordinary regression).

Now, you may ask, why take a log? For the sake of simplicity, let's just say that this is one of the best mathematical way to replicate a step function. I can go in more details, but that will beat the purpose of this article.

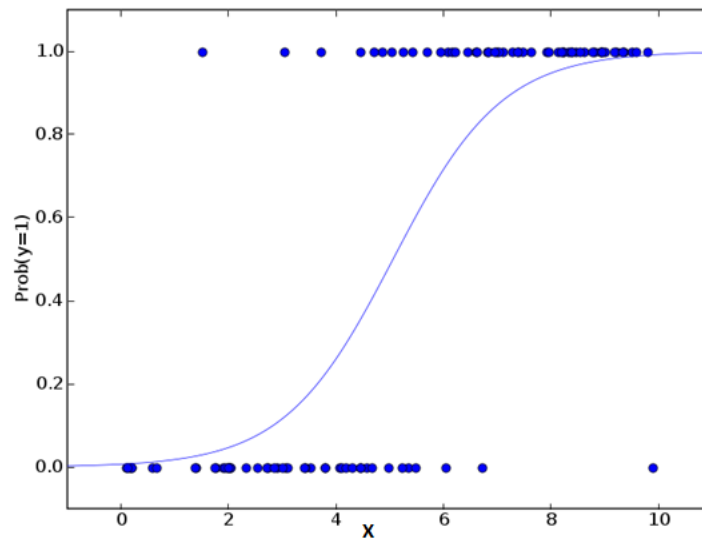


Figure 4-4 Logistic regression algorithm

### Python Code

```
#Import Library
from sklearn.linear_model import LogisticRegression
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predicto
r) of test_dataset
```



```
# Create logistic regression object
model = LogisticRegression()
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Equation coefficient and Intercept
print('Coefficient: \n', model.coef_)
print('Intercept: \n', model.intercept_)
#Predict Output
predicted= model.predict(x_test)
```

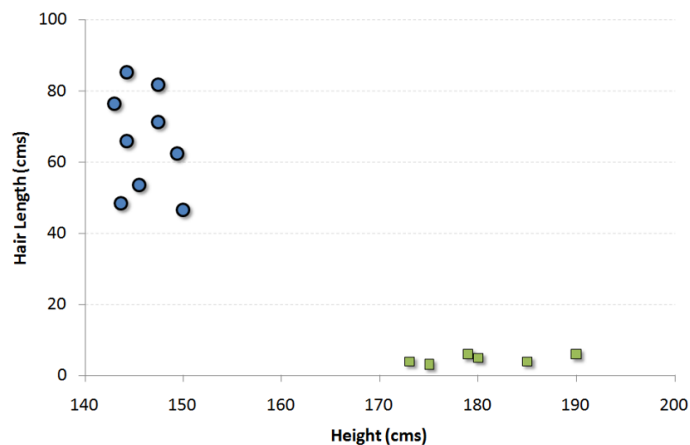
There are many different steps that could be tried in order to improve the model:

- including interaction terms
- removing features
- regularization techniques
- using a non-linear model

### 4.4.3 SVM (Support Vector Machine)

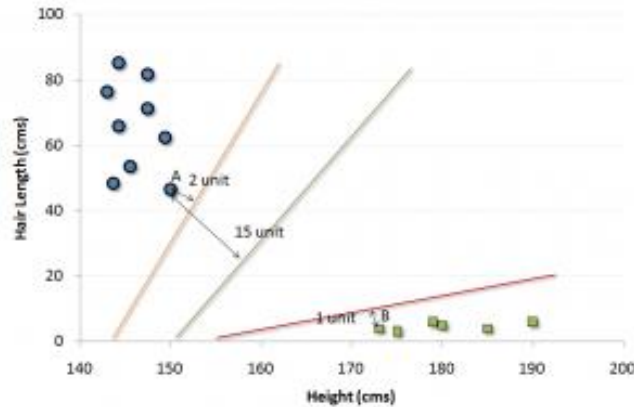
It is a classification method. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

For example, if we only had two features like Height and Hair length of an individual, we'd first plot these two variables in two dimensional space where each point has two co-ordinates (these co-ordinates are known as **Support Vectors**).



4-3SVM

Now, we will find some *line* that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away.



In the example shown above, the line which splits the data into two differently classified groups is the *black* line, since the two closest points are the farthest apart from the line. This line is our classifier. Then, depending on where the testing data lands on either side of the line, that's what class we can classify the new data as.

**Think of this algorithm as playing JezzBall in n-dimensional space. The tweaks in the game are:**

- You can draw lines / planes at any angles (rather than just horizontal or vertical as in classic game)
- The objective of the game is to segregate balls of different colors in different rooms.
- And the balls are not moving.

Python Code

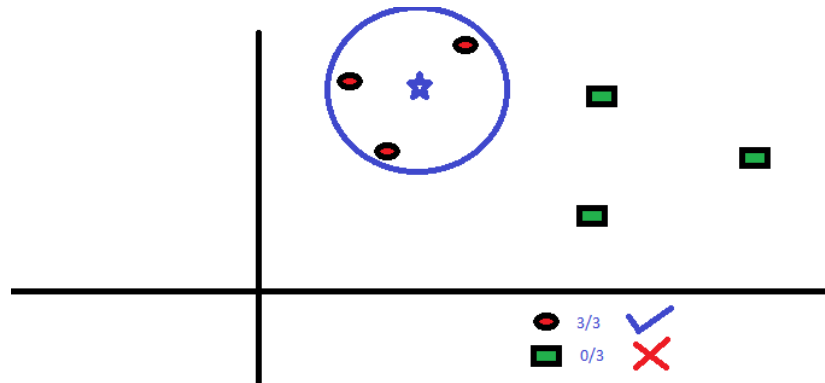
```
#Import Library

from sklearn import svm
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset
# Create SVM classification object
model = svm.svc() # there is various option associated with it, this is simple for classification. You can refer link, for more detail.
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)
```

#### 4.4.4 KNN (k- Nearest Neighbors)

It can be used for both classification and regression problems. However, it is more widely used in classification problems in the industry. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function.

These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance. First three functions are used for continuous function and fourth one (Hamming) for categorical variables. If  $K = 1$ , then the case is simply assigned to the class of its nearest neighbor. At times, choosing K turns out to be a challenge while performing kNN modeling.



KNN can easily be mapped to our real lives. If you want to learn about a person, of whom you have no information, you might like to find out about his close friends and the circles he moves in and gain access to his/her information!

#### Things to consider before selecting kNN:

- KNN is computationally expensive
- Variables should be normalized else higher range variables can bias it
- Works on pre-processing stage more before going for kNN like outlier, noise removal

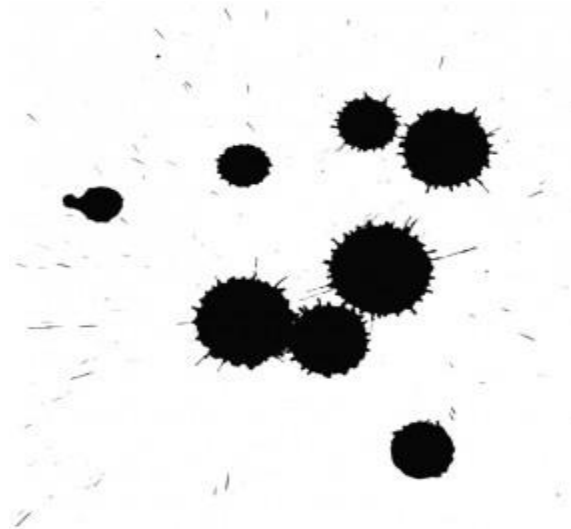
#### Python Code

```
#Import Library
from sklearn.neighbors import KNeighborsClassifier
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predicto
r) of test_dataset
# Create KNeighbors classifier object model
```

```
KNeighborsClassifier(n_neighbors=6) # default value for n_neighbors is 5
# Train the model using the training sets and check score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)
```

#### 4.4.5 K-Means

It is a type of unsupervised algorithm which solves the clustering problem. Its procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume  $k$  clusters). Data points inside a cluster are homogeneous and heterogeneous to peer groups. Remember figuring out shapes from ink blots?  $k$  means is somewhat similar this activity. You look at the shape and spread to decipher how many different clusters / population are present.



##### How K-means forms cluster:

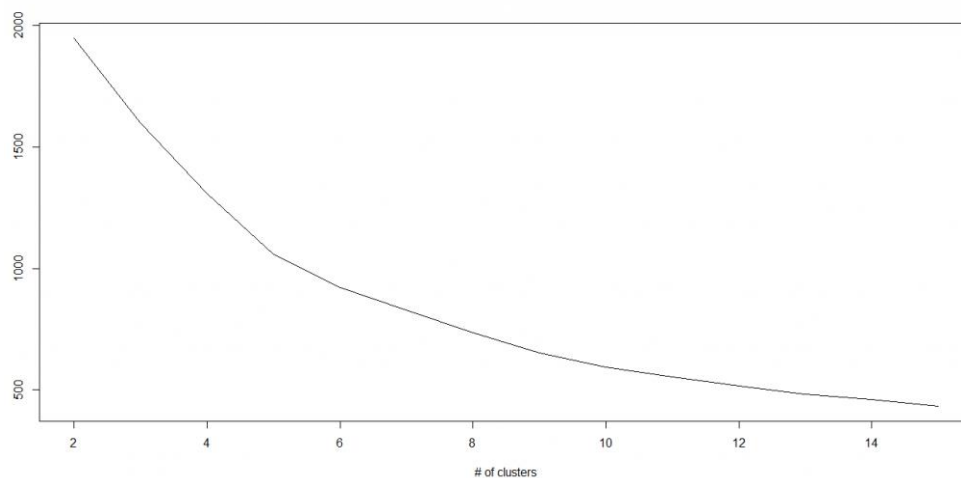
1. K-means picks  $k$  number of points for each cluster known as centroids.
2. Each data point forms a cluster with the closest centroids i.e.  $k$  clusters.
3. Finds the centroid of each cluster based on existing cluster members. Here we have new centroids.

4. As we have new centroids, repeat step 2 and 3. Find the closest distance for each data point from new centroids and get associated with new k-clusters. Repeat this process until convergence occurs i.e. centroids does not change.

### How to determine value of K:

In K-means, we have clusters and each cluster has its own centroid. Sum of square of difference between centroid and the data points within a cluster constitutes within sum of square value for that cluster. Also, when the sum of square values for all the clusters are added, it becomes total within sum of square value for the cluster solution.

We know that as the number of cluster increases, this value keeps on decreasing but if you plot the result you may see that the sum of squared distance decreases sharply up to some value of k, and then much more slowly after that. Here, we can find the optimum number of cluster.



### Python Code

```
#Import Library
from sklearn.cluster import KMeans
#Assumed you have, X (attributes) for training data set and x_test(attributes) of test_data
set
# Create KNeighbors classifier object model
k_means = KMeans(n_clusters=3, random_state=0)
# Train the model using the training sets and check score
model.fit(X)
#Predict Output
predicted= model.predict(x_test)
```

## 4.5 Machine learning application

Another categorization of machine learning tasks arises when one considers the desired *output* of a machine-learned system:

- In classification, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised manner. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".
- In regression, also a supervised problem, the outputs are continuous rather than discrete.
- In clustering, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
- Dimensionality reduction simplifies inputs by mapping them into a lower-dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked with finding out which documents cover similar topics.

#### **4.5.1 Difference between Classification and Regression in Machine Learning:**

Fundamentally, classification is about predicting a label and regression is about predicting a quantity. Predictive modeling is the problem of developing a model using historical data to make a prediction on new data where we do not have the answer. Predictive modeling can be described as the mathematical problem of approximating a mapping function ( $f$ ) from input variables ( $X$ ) to output variables ( $y$ ). This is called the problem of function approximation. The job of the modeling algorithm is to find the best mapping function we can give the time and resources available. Generally, we can divide all function approximation tasks into classification tasks and regression tasks.

#### **4.5.2 Classification Predictive Modeling**

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.

**For example**, an email of text can be classified as belonging to one of two classes: "spam" and "*not spam*".

- A classification problem requires that examples be classified into one of two or more classes.
- A classification can have real-valued or discrete input variables.
- A problem with two classes is often called a two-class or binary classification problem.
- A problem with more than two classes is often called a multi-class classification problem.
- A problem where an example is assigned multiple classes is called a multi-label classification problem.

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.

For **example**, a specific email of text may be assigned the probabilities of 0.1 as being “spam” and 0.9 as being “not spam”. We can convert these probabilities to a class label by selecting the “not spam” label as it has the highest predicted likelihood.

There are many ways to estimate the skill of a classification predictive model, but perhaps the most common is to calculate the classification accuracy. The classification accuracy is the percentage of correctly classified examples out of all predictions made. For **example**, if a classification predictive model made 5 predictions and 3 of them were correct and 2 of them were incorrect, then the classification accuracy of the model based

$$1 \text{ accuracy} = \text{correct predictions} / \text{total predictions} * 100$$

$$2 \text{ accuracy} = 3 / 5 * 100$$

$$3 \text{ accuracy} = 60\%$$

on just these predictions would be:

An algorithm that is capable of learning a classification predictive model is called a classification algorithm.

### 4.5.3 Regression Predictive Modeling

In applied machine learning we will borrow, reuse and steal algorithms from many different fields, including statistics and use them towards these ends.

As such, linear 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.

For example, a house may be predicted to sell for a specific dollar value, perhaps in the range of \$100,000 to \$200,000.

- A regression problem requires the prediction of a quantity.
- A regression can have real valued or discrete input variables.
- A problem with multiple input variables is often called a multivariate regression problem.
- A regression problem where input variables are ordered by time is called a time series forecasting problem.

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.

For example, if a regression predictive model made 2 predictions, one of 1.5 where the expected value is 1.0 and another of 3.3 and the expected value is 3.0, then the RMSE would be:

```
1 RMSE = sqrt(average(error^2))
2 RMSE = sqrt(((1.0 - 1.5)^2 + (3.0 - 3.3)^2) / 2)
3 RMSE = sqrt((0.25 + 0.09) / 2)
4 RMSE = sqrt(0.17)
5 RMSE = 0.412
```

A benefit of RMSE is that the units of the error score are in the same units as the predicted value.

An algorithm that is capable of learning a regression predictive model is called a regression algorithm. Some algorithms have the word “regression” in their name, such as linear regression and logistic regression, which can make things confusing because linear regression is a regression algorithm whereas logistic regression is a classification algorithm.

#### 4.5.4 In conclusion: Classification vs Regression

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.

There is some overlap between the algorithms for classification and regression.

For **example**: A classification algorithm may predict a continuous value, but the continuous value is in the form of a probability for a class label. A regression algorithm may predict a discrete value, but the discrete value in the form of an integer quantity.

Some algorithms can be used for both classification and regression with small modifications, such as decision trees and artificial neural networks. Some algorithms cannot, or cannot easily be used for both problem types, such as linear regression for

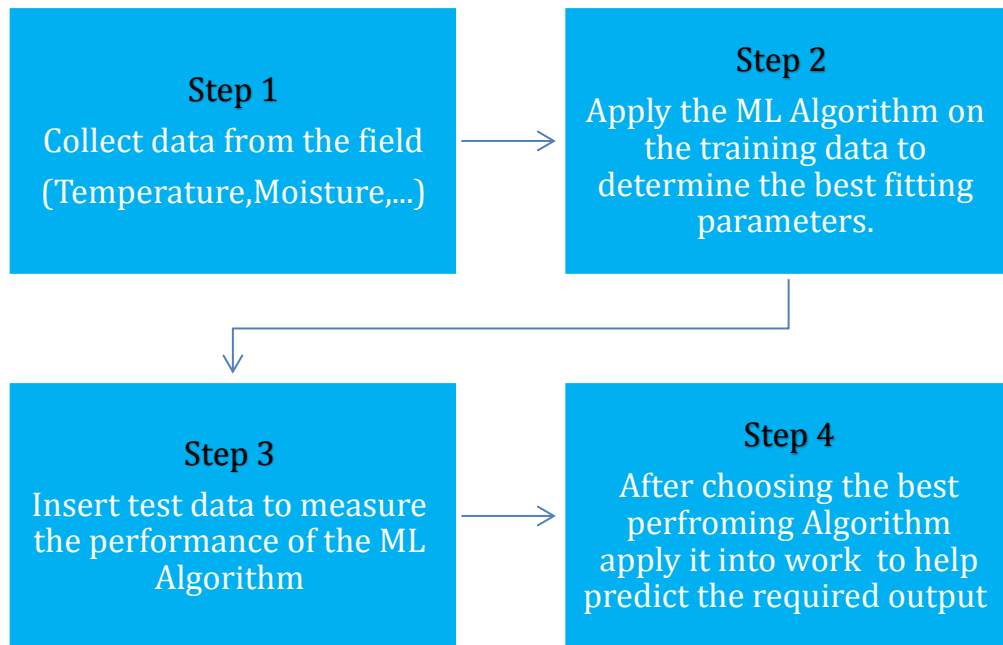


regression predictive modeling and logistic regression for classification predictive modeling.

Importantly, the way that we evaluate classification and regression predictions varies and does not overlap. For example: Classification predictions can be evaluated using accuracy, whereas regression predictions cannot. Regression predictions can be evaluated using root mean squared error, whereas classification predictions cannot.

## 4.6 The models chosen to be used in the scope of the project

### 4.6.1 General Procedures:



### 4.6.2 Dataset:

The dataset used, consists of two parts, the first part is gathered from **Dr. Fahem** and could be used to build few different models, the limitations on it was the lack of the training samples number and the unavailability of some required sensors.

**The first model** is based on the data samples shown in Figure(4-6) and these samples are used to **detect the rate of disease spreading after the first observance of the diseases** so as to predict the overall losses in the annual yield, or to predict whether this would be an epidemic year or the crops could be saved, Linear Regression algorithms were tested

with different orders but again due to the lack of the training samples; even though the accuracy reached satisfying results that was not satisfying to predict any further results.

Also Dr. Fahem 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, a sample of the data is shown in Figure (4-6), as before by observing the dataset samples the feature which can be used to predict somehow useful info is:

Rain days > 0.1mm: but this feature is not useful as along all the data set it remained zeros.

Accu. GDD: Accumulative GDD

The output (the results needed to be predicted)

Daily blight observation: how much the disease spread each day after first occurrence of infection.

DAP	TMIN	TMAX	Rain days>0.1mm	GDD	Accu. GDD	Daily blight obsrv.	DS_Incrm.	IP					IP Result
69	11.6	25	0	8	349	0.00	0.00	1	0	0	1	0	0
70	10.8	23.1	0	7	356	0.00	0.00	1	1	0	2	0	0
71	10.8	22.8	0	7	362	0.00	0.00	1	1	0	2	0	0
72	8.2	24.3	0	6	369	0.00	0.00	1	1	0	2	0	0
73	4.8	23.3	0	4	373	0.00	0.00	0	1	0	1	0	0
74	5.1	23.8	0	4	377	0.10	0.00	0	1	0	1	0	0
75	5.8	23.2	0	5	382	0.23	0.13	0	1	0	1	0	0
76	8	24.1	0	6	388	0.36	0.13	0	1	0	1	0	0
77	7.8	28.2	0	8	396	0.49	0.14	0	0	0	0	0	0
78	8.5	22.2	0	5	401	0.62	0.14	1	1	0	2	0	0
79	6.7	20.9	0	4	405	0.75	0.14	0	1	0	1	0	0
80	7.1	19.3	0	3	408	0.88	0.15	0	1	0	1	0	0
81	6.4	19.7	0	3	411	1.01	0.16	0	1	0	1	0	0
82	4.9	19.9	0	2	414	1.14	0.16	0	1	0	1	0	0
83	5.6	20.5	0	3	417	1.27	0.16	0	1	0	1	0	0
84	4.5	21.5	0	3	420	1.40	0.17	0	1	0	1	0	0
85	11.5	20.5	0	6	426	1.57	0.17	1	1	0	2	0	0

Figure 4-4 Daily Blight Observation Training Dataset

After feeding the model with Accumulative GDD and The Daily blight observation and by applying the mean square error algorithm between the predicted outputs of the test data and the real outputs; the 5th order Linear Regression gives an accuracy equal to 98.8, so now the model is built and is ready to be used to predict the daily blight, Figure (4-7) shows

a sample of data gathered from real sensors applied on the field or gathered from the weather station inside Egypt in the areas of interest, The data gathered must be processed to match the training data that was handled by few code lines to turn the temperature measures into GDD according to equation (1) then sum all the reading to get Accu. GDD.

$$\text{GDD} = \frac{T_{max} + T_{min}}{2} - 10^{\circ}\text{C} \text{ equ(1)}$$

DAP	Temp Max	Temp Min
40	22	7
41	22	11
42	23	10
43	26	8
44	20	10
45	21	5
46	21	7
47	25	12
48	20	10
49	25	12
50	23	6
51	21	9
52	26	12
53	20	8
54	25	6
55	25	6
56	24	12

Figure 4-5 Real Life Data measured of the field

GDD	Accu. GDD
4.5	4.5
6.5	9
6.5	15.5
7	22
5	29
3	34
4	37
8.5	41
5	49.5
8.5	54.5
4.5	63
5	67.5
9	72.5
4	81.5
5.5	85.5
5.5	91
8	96.5
4.5	104.5

Figure 4-6 Data modification to fit the ML Algorithm

	A	B	C
1	Accu. GDD	IPS	DS
12	91.85	0	2
13	100.75	0	3.6
14	109.7	0	5.2
15	118.65	0	6.8
16	124.75	0	8.4
17	130.2	0	10
18	136.7	0	12.4
19	145.2	0	14.8
20	152.65	0	17.2
21	162.1	0	19.6
22	169.5	0	22
23	174.35	1	24.4
24	181.85	2	26.8
25	188.7	3	29.2
26	196.9	3	31.6

Figure 4-7 Disease Severity Training Data set

Variable: Temperature translated into GDD ( )

1	Temp_Max	Temp_Min
2		14
3		29
4		34
5		20
6		29
7		31
8		28
9		20
10		26
11		27
12		29
13		28
14		15



1	Accu. GDD
2	7.25
3	15.35
4	24.6
5	32.8
6	41.4
7	49.4
8	57.4
9	64.3
10	73.4
11	82.5
12	91.85
13	100.75
14	109.7

As they are related by relation of

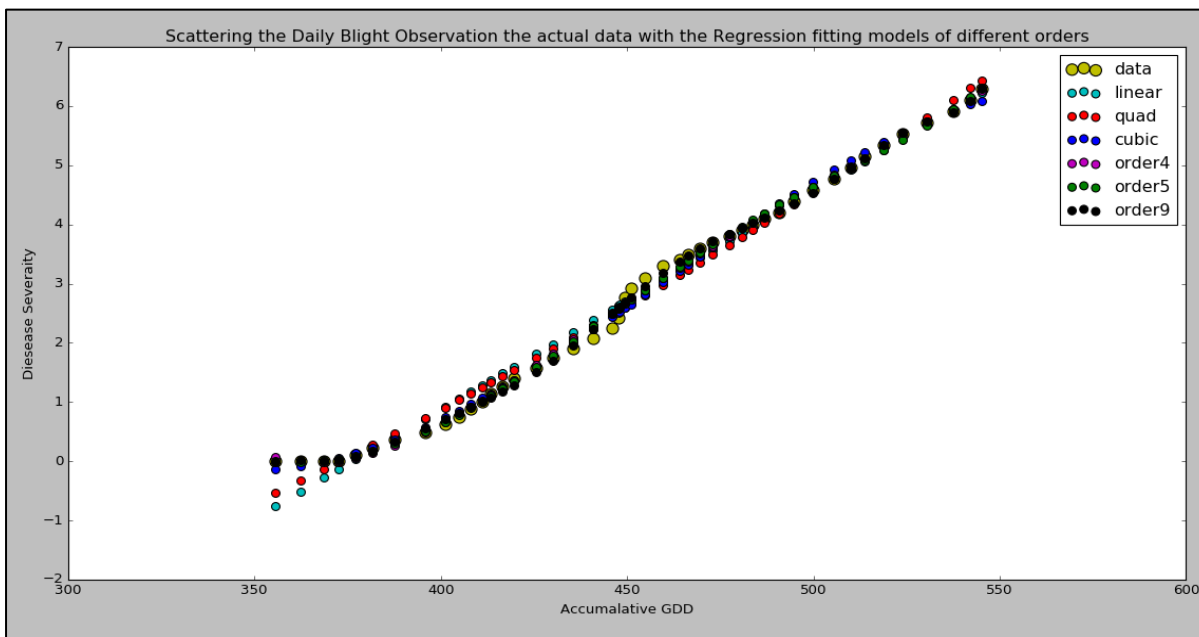
Output: DS (disease severity) which shows the severity of infection

Where detection of disease is likely to happen between 18-20 °C and accumulative GDD is calculated after 40 days from cultivation.

This algorithm was written in Tensor flow based on python.

## 4.7 Machine learning in practice

For the First part of samples which is about the Potato Daily Blight Observation, after applying the Regression curves fitting tools the curves are shown in Figure (1), after observing the Figure, the curves are pretty similar to each other or not clearly exposing the best order so instead by applying the mean square error Algorithm and observing the accuracy results in Figure (2), the best cost function with respect to the Linear regression is chosen to be of ninth order with accuracy percentage= 99.84% ,in Figure (3) the last two rows are in order, predicted output due to Regression cost function of ninth Dimensions, the real outputs of the tested samples.

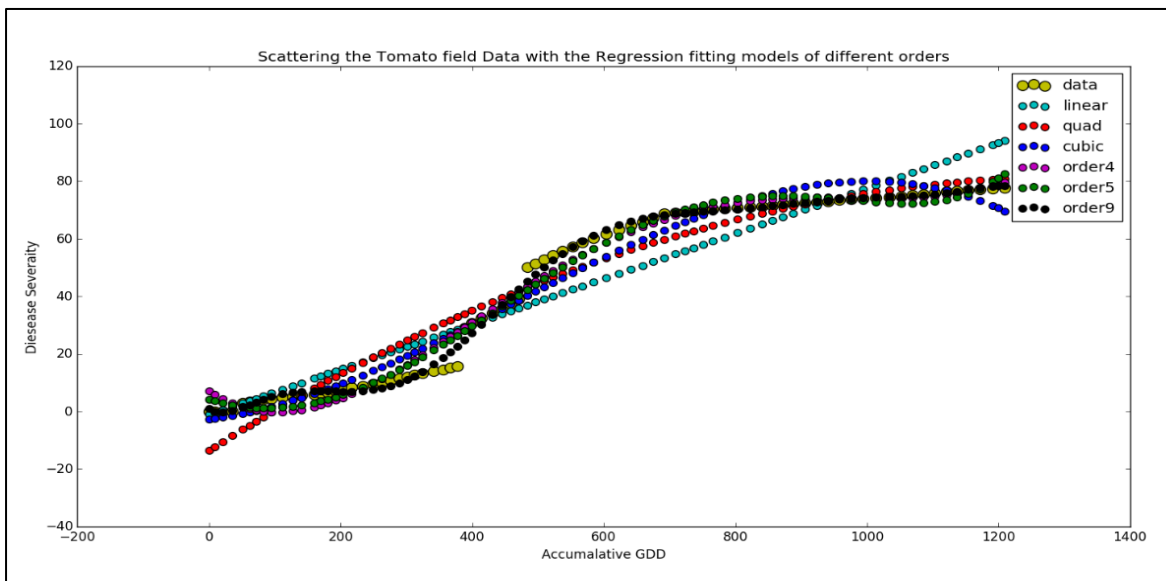


```
In [47]: runfile('C:/Users/loloa/Desktop/finalnada/potatodailyblight.py', wdir='C:/Users/loloa/Desktop/finalnada')
46
('Accuracy of Regression of order 1=', 98.702925703925956)
('Accuracy of Regression of order 2=', 98.961032123377578)
('Accuracy of Regression of order 3=', 99.595935731039489)
('Accuracy of Regression of order 4=', 99.743206237063148)
('Accuracy of Regression of order 5=', 99.75825383827393)
('Accuracy of Regression of order 9=', 99.8485430662531)
```

Predict_x	float64	(6L)	array([ 518.75, 524. , 530.5 , 537.5 , 542.1 , 545.1 ])
Predict_y_1_order	float64	(6L)	array([ 5.24894803, 5.44247403, 5.68207766, 5.94011234, 6.10967798, ...
Predict_y_2_order	float64	(6L)	array([ 5.31525143, 5.53485845, 5.80946467, 6.10854982, 6.30698611, ...
Predict_y_3_order	float64	(6L)	array([ 5.38826175, 5.5535715 , 5.74202512, 5.92290021, 6.02819996, ...
Predict_y_4_order	float64	(6L)	array([ 5.28765271, 5.46925609, 5.6988157 , 5.95596702, 6.13304794, ...
Predict_y_5_order	float64	(6L)	array([ 5.25334024, 5.4309412 , 5.66531877, 5.94620732, 6.15355196, ...
Predict_y_9_order	float64	(6L)	array([ 5.34653607, 5.54814596, 5.73628958, 5.90014119, 6.0838942 , ...
Predict_y_test	float64	(6L)	array([ 5.34, 5.53, 5.72, 5.91, 6.1 , 6.3 ])

%% not the same samples those for disease severity

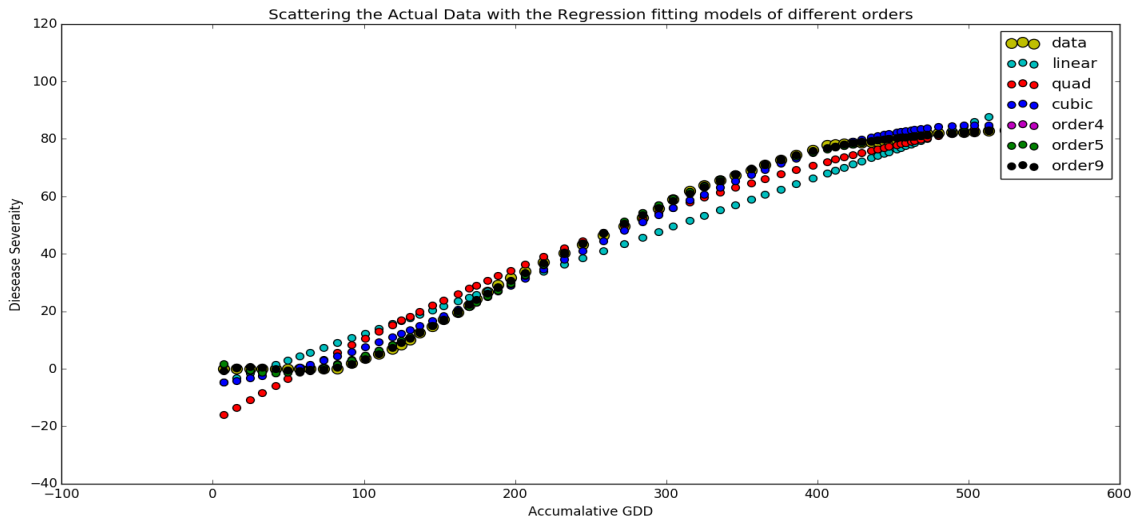
Also, the same samples are gathered for tomatoes, but for the tomatoes the accuracy is quite worse due to the discontinuities in the curve



Predict_y_1_order	float64	(5L.)	array([ 95.57532466, 96.25015281, 96.469016 , 96.83378797, 96.97057746])
Predict_y_2_order	float64	(5L.)	array([ 87.93834689, 88.20675641, 88.292571 , 88.43424877, 88.48694399])
Predict_y_3_order	float64	(5L.)	array([ 82.1783256 , 81.76865505, 81.62908552, 81.3891213 , 81.29675699])
Predict_y_4_order	float64	(5L.)	array([ 84.17714513, 84.33953972, 84.39454197, 84.48890751, 84.52519202])
Predict_y_5_order	float64	(5L.)	array([ 84.10336282, 84.21193748, 84.24813081, 84.30962084, 84.33307473])
Predict_y_9_order	float64	(5L.)	array([ 84.19742179, 84.28417389, 84.30932381, 84.34770394, 84.36090408])
Predict_y_test	float64	(5L.)	array([ 84.2, 84.4, 84.6, 84.8, 84. ])

```
In [40]: runfile('C:/Users/lolea/Desktop/finalnada/codetomatods.py', wdir='C:/Users/lolea/Desktop/finalnada')
('Accuracy of Regression of order 1=', 89.540740448874516)
('Accuracy of Regression of order 2=', 93.816087102211014)
('Accuracy of Regression of order 3=', 96.460880245289843)
('Accuracy of Regression of order 4=', 98.398372095668108)
('Accuracy of Regression of order 5=', 98.571221401335947)
('Accuracy of Regression of order 9=', 99.644221273546492)
```

Potato disease severity



Predict_x	float64	(5L)	array([ 1034.0575, 1051.6575, 1068.7575, 1085.8075, 1103.0575])
Predict_y_1_order	float64	(5L)	array([ 80.26794172, 81.6530239, 82.99875715, 84.34055551, 85.69809344])
Predict_y_2_order	float64	(5L)	array([ 76.98011015, 77.52344196, 78.01942201, 78.48263337, 78.91945327])
Predict_y_3_order	float64	(5L)	array([ 79.83414249, 79.51907068, 79.07209383, 78.48413313, 77.74057259])
Predict_y_4_order	float64	(5L)	array([ 74.12842168, 74.16014039, 74.25089353, 74.41719617, 74.68045009])
Predict_y_5_order	float64	(5L)	array([ 72.46814381, 72.33771042, 72.3452356, 72.52013897, 72.90579848])
Predict_y_9_order	float64	(5L)	array([ 74.57816188, 74.74599668, 74.94028805, 75.19433181, 75.53986547])
Predict_y_test	float64	(5L)	array([ 74.51, 74.8, 75.09, 75.38, 76. ])

```
In [24]: runfile('C:/Users/loloa/Desktop/finalnada/ml1.py',
wdir='C:/Users/loloa/Desktop/finalnada')
('Accuracy of Regression of order 1=', 95.21616681084241)
('Accuracy of Regression of order 2=', 97.635842621167114)
('Accuracy of Regression of order 3=', 99.469916822607985)
('Accuracy of Regression of order 4=', 99.932565796407943)
('Accuracy of Regression of order 5=', 99.93415900286206)
('Accuracy of Regression of order 9=', 99.975168452128457)
```

- Using Python's popular **scikit-learn** framework, we can train a **Support Vector Machine** (type of algorithm) on a set of data pretty quickly.

```
#Import the support vector machine module from the sklearn framework
```

```
from sklearn import svm
```

```
#Label x and y variables from our dataset
```

```
x = ourData.features
```

```
y = ourData.labels
```

```
#Initialize our algorithm
```

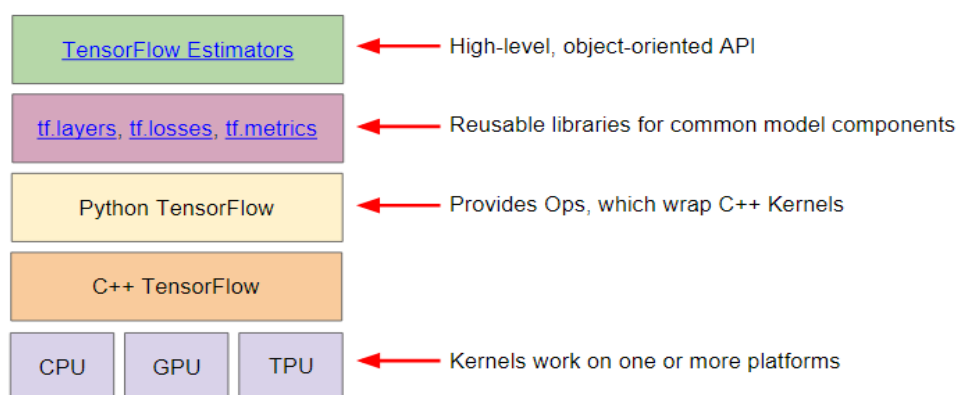
```
classifier = svm.SVC()
```

```
#Fit model to our data
```

```
classifier.fit(x,y)
```

Scikit-learn is one of many packages that developers use to make Machine Learning easier and more powerful. [It](#) is an extremely popular open-source ML library in Python, with over 100k users, including many at Google. Some other popular ones which is used in this project include:

- **Tensorflow** is a computational framework for building machine learning models. TensorFlow provides a variety of different toolkits that allow you to construct models at your preferred level of abstraction. You can use lower-level APIs to build models by defining a series of mathematical operations. Alternatively, you can use higher-level APIs (like `tf.estimator`) to specify predefined architectures, such as linear regression or neural networks. The following figure shows the current hierarchy of TensorFlow toolkits:



**Figure 4-10 TensorFlow toolkit hierarchy.**

<https://developers.google.com/machine-learning/crash-course/first-steps-with-tensorflow/toolkit>

TensorFlow consists of the following two components:

- a [graph protocol buffer](#)
- a runtime that executes the (distributed) graph

These two components are analogous to Python code and the Python interpreter. Just as the Python interpreter is implemented on multiple hardware platforms to run Python code, TensorFlow can run the graph on multiple hardware platforms, including CPU, GPU, and [TPU](#).

Using the highest level of abstraction that solves the problem. `tf.estimator` is compatible with the scikit-learn API.

Example:

```
import tensorflow as tf
```

```
# Set up a linear classifier.
```

```
classifier = tf.estimator.LinearClassifier(feature_columns)
```

```
# Train the model on some example data.
```

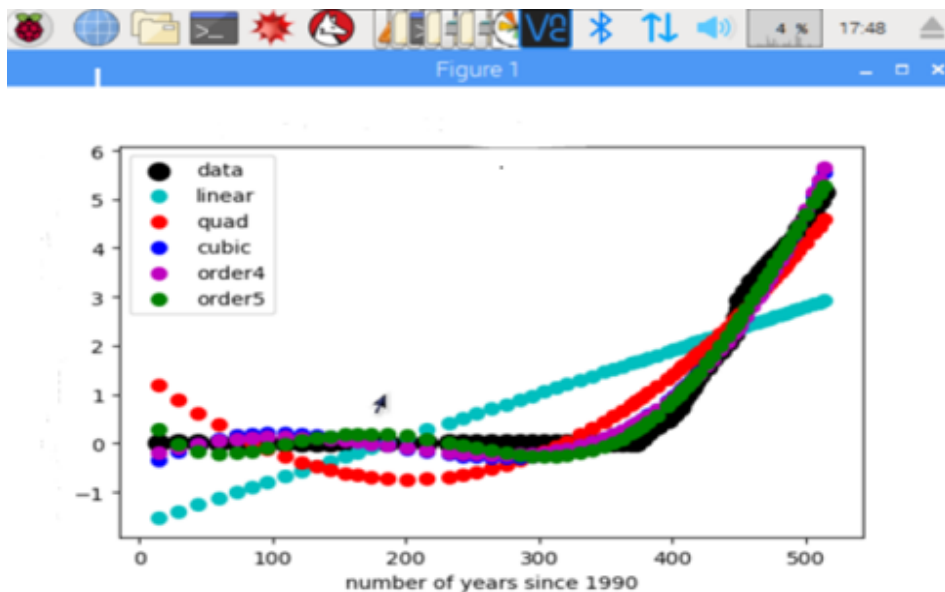


```
classifier.train(input_fn=train_input_fn, steps=2000)
```

```
# Use it to predict.
```

```
predictions = classifier.predict(input_fn=predict_input_fn)
```

## 4.8 Results



- **Fitting curves of different models to the real data sets**

Accuracy of 99 % was obtained at higher orders as shown in the following figure

```
pi@raspberrypi:~$ sudo python ML.py
524. 531. 538. 542. 545.]
'R_linear', 56.852659393347253)
'R_quad ', 89.809281718403568)
'R_cubic ', 97.868115774920057)
'R_order4', 97.942928438187565)
'R_order5', 98.82660292152066)
'Predict_y_linear=', array([ 4.92500603, 5.17335818, 5.42704613, 5.57440635,
5.6860699 ]))
```

Figure 4-11 accuracy of machine learning code using different orders models

## 5 Chapter 5: Gateway

### 5.1 IOT concept

The internet of things, or IOT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (IDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. Thus, IOT is used to describe the practice of connecting devices through the use of the Internet.

A thing in the internet of things refers to a physical device connected to the internet. It can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low or any other natural or man-made object that can be assigned an IP address and is able to transfer data over a network.

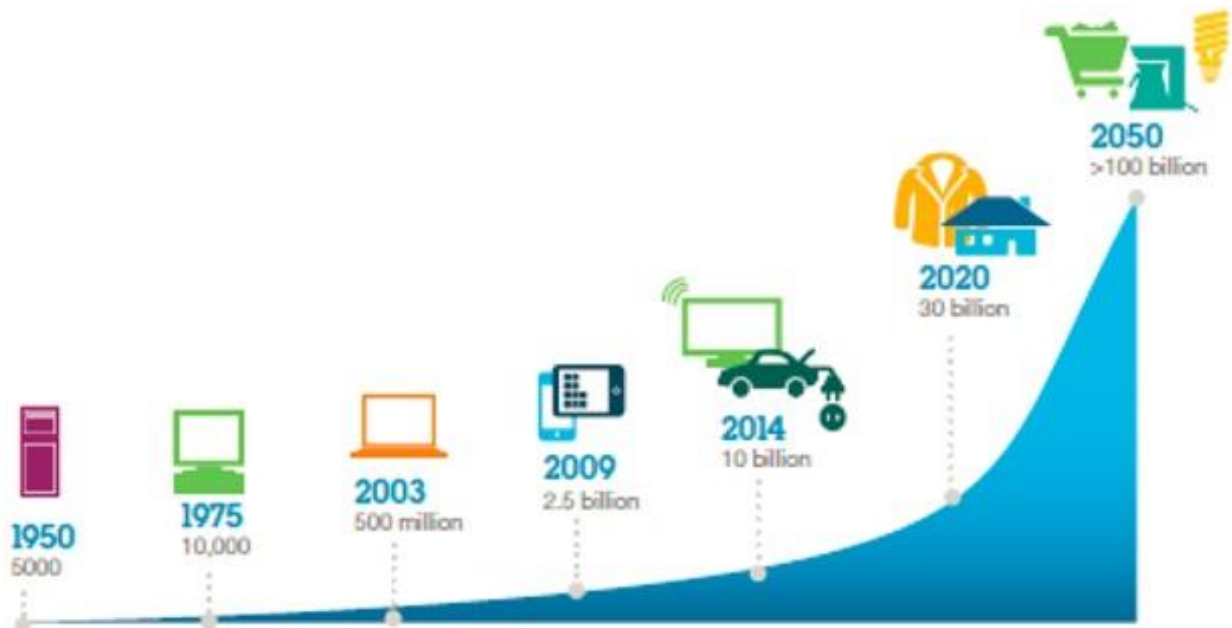
Increasingly, organizations in a variety of industries are using IOT to operate more efficiently, better understand customers to deliver enhanced customer service, improve decision-making and increase the value of the business.

An IOT ecosystem consists of web-enabled smart devices that use embedded processors, sensors and communication hardware to collect, send and act on data they acquire from their environments. IOT devices share the sensor data they collect by connecting to an IOT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices - for instance, to set them up, give them instructions or access the data.

The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IOT applications deployed.

Historically, the term Internet of Things was first mentioned by Kevin Ashton, co-founder of the Auto-ID Center at MIT, in a presentation he made to Procter & Gamble (P&G) in 1999. The idea of IOT came to Ashton when he was searching for ways that Procter & Gamble could improve its business by linking RFID information to the Internet. The concept was simple but powerful. If all objects in daily life were equipped with identifiers and wireless connectivity, these objects could be communicating with each other and be managed by computers.

In a report published by IBM on the future of Internet of Things, the number of connected devices is forecasted to surpass **30 billion** in **2020**, up from **2.5 billion** in **2009** and **10 billion** today. The connected devices need a protocol using which they could communicate only when it is required. Devices with constrained resources should be able to communicate with various other heterogeneous devices.



**Figure 5-8: Forecast for the number of connected devices till 2050**

In order to fulfill these aspirations of the increasing number of connected devices that can communicate with each other as well as sending their data to the internet and receiving back a response or recommended actions for the devices or the people, IOT has faced challenges regarding the connectivity, data transmission and resource constrains of the connected devices. To conquer these challenges and accelerate the process of innovation in IOT, the birth of new technologies was a must. These technologies were designed especially for features and applications that IOT would provide. Now we shall introduce the MQTT (Message Queuing Telemetry Transport) protocol that is considered as one of the most promising protocols when talking about the transfer of messages or data that belong to IOT.

MQTT is designed to overcome the challenges of connecting the rapidly expanding physical world of sensors, actuators, phones, and tablets with established software processing technologies. These principles also turn out to make this protocol ideal for the emerging IOT or machine to machine (M2M) world of connected devices where bandwidth and battery power are at a premium quality.

## 5.2 MQTT protocol

### 5.2.1 Overview

MQTT (Message Queuing Telemetry Transport) protocol is a lightweight messaging protocol that provides resource-constrained network clients with a simple way to distribute telemetry information. The protocol, which uses a publish/subscribe communication pattern, is used in M2M communication and plays an important role in the IOT. It requires very low bandwidth and has less battery consumption. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small device scenarios.

It is also ideal for mobile applications because of its small size, low power usage, minimized data packets, and efficient distribution of information to one or many receivers.

In light of that, it is important to mention a little bit about the history of the protocol. In 1999, Dr. Stanford-Clark from IBM collaborated with Arlen Nipper from Arcom to author the first version of MQTT protocol. The "MQ" in "MQTT" came from IBM's MQ Series message queuing product line. However, queuing itself is not required to be supported as a standard feature in all situations.

### **5.2.2 MQTT architecture**

MQTT is a publish/subscribe messaging protocol (often called pub-sub), which is based on client/server model, where every sensor is a client and connects to a server, known as a broker, over TCP. MQTT is a message oriented where every message is published to specific address, called topics. Clients may subscribe to multiple topics and once a client subscribes to a certain topic, it can receive every message that is being published to that topic. The main distributor of the messages is a node that called a MQTT broker which is responsible for forwarding the messages between the sender and multiple receivers. If there are more than one node, then these nodes (clients) are arranged around the broker in a star topology, where every client publishes a message including a topic. The topic is the routing information for the broker. In the other hand, if any other client wants to receive messages, it should subscribe to this topic and the broker delivers all messages with the matching topic to the client.

It is clear now that the different clients related to same network don't have to know each other, as they communicate over the topic. Topics in MQTT have a particular syntax, where they are arranged in hierarchy using forward slash character (/) as a separator much like the path of the URL. So, humidity temperature sensor in your home might publish to a topic like "sensor/home/temp\_hum".

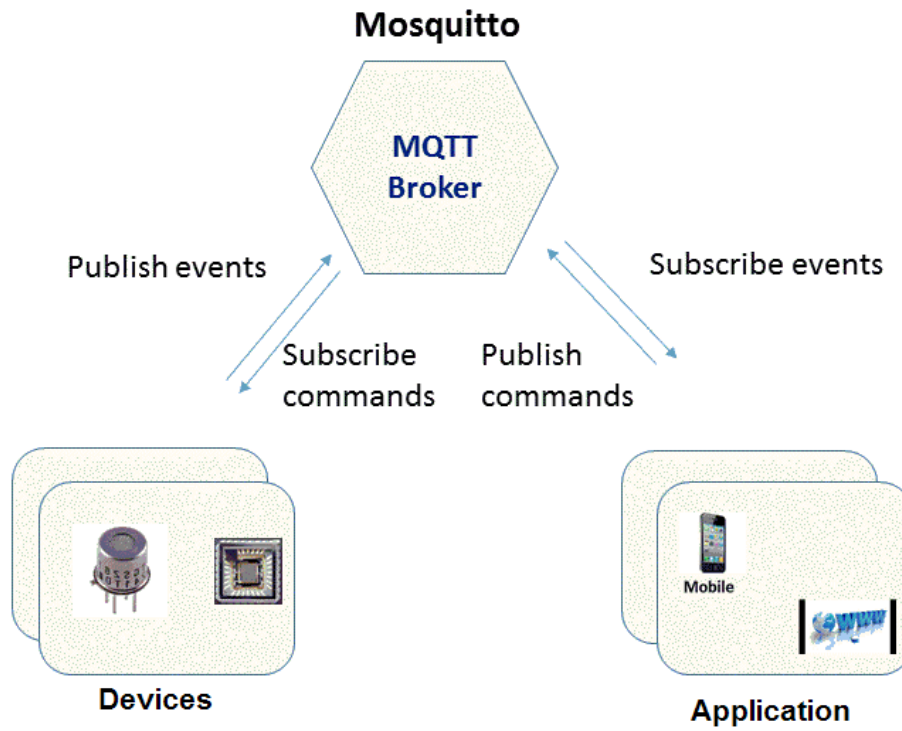


Figure 5-9: MQTT architecture

As an illustrating **example**, we can imagine a simple network with three clients and a central broker. All three clients open TCP connections with the broker. Clients B and C subscribe to the topic `temp`.

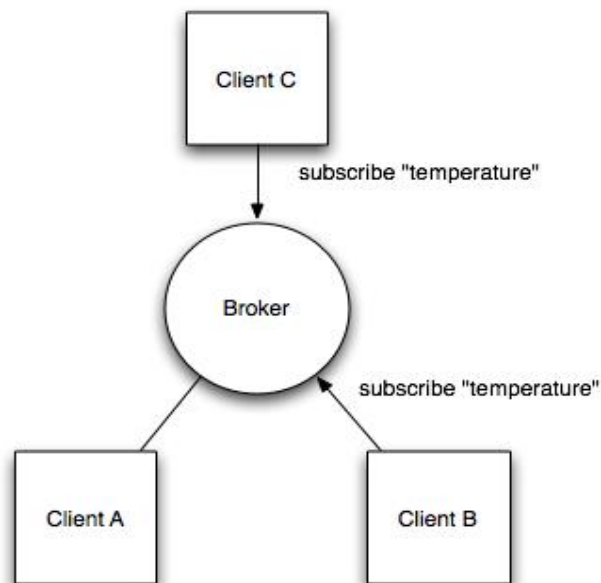
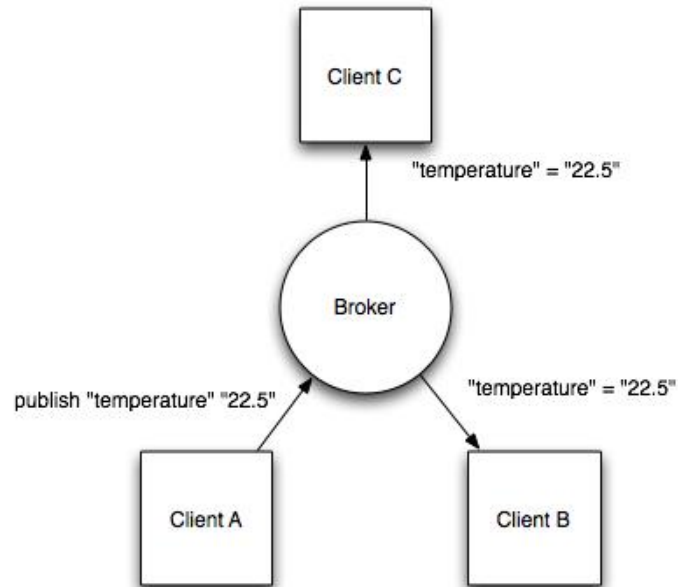


Figure 5-10: Clients (B) and (C) subscribe

At a later time, Client A publishes a value of 22.5 for topic `temp`. The broker forwards the message to all subscribed clients.



**Figure 5-11: Client (A) publishes a message**

The publisher subscriber model allows MQTT clients to communicate one-to-one, one-to-many and many-to-one.

## 5.2.3 Topics and wild cards:

### 5.2.3.1 Topic format:

A topic is a UTF-8 string, which is used by the broker to filter messages for each connected client. A topic consists of one or more topic levels. Each topic level is separated by a forward slash (/), a topic level separator.



**Figure 12-5: Topic format in MQTT**

In comparison to a message queue, the topic is very lightweight. There is no need for a client to create the desired topic before publishing or subscribing to it, because a broker accepts each valid topic without any prior initialization. Each topic must have at least 1 character to be valid and it can also contain spaces. Also, a topic is case-sensitive, which makes *myhome/temperature* and *MyHome/Temperature* two individual topics.

### 5.2.3.2 Wild cards:

When a client subscribes to a topic it can use the exact topic the message was published to or it can subscribe to more topics at once by using wildcards. A wildcard can only be used when subscribing to topics and is not permitted when publishing a message. In the following we will look at the two different kinds one by one: *single level* and *multi-level* wildcards.

Single Level (+): as the name already suggests, a single level wildcard is a substitute for one topic level. The plus symbol (+) represents a single level wildcard in the topic.



Figure 5-13: The use of (+) wild card in topic format

Any topic matches to a topic including the single level wildcard if it contains an arbitrary string instead of the wildcard.

For example, a subscription to *myhome/groundfloor+/temperature* would match or not match the following topics:

- ✓ myhome / groundfloor / livingroom / temperature
- ✓ myhome / groundfloor / kitchen / temperature
- ✗ myhome / groundfloor / kitchen / brightness
- ✗ myhome / firstfloor / kitchen / temperature
- ✗ myhome / groundfloor / kitchen / fridge / temperature

Figure 5-14: Illustration of valid wildcards in single level

**Multi-level (#):** while the single level wildcard only covers one topic level, the multi-level wildcard covers an arbitrary number of topic levels. In order to determine the matching topics, it is required that the multi-level wildcard is always the last character in the topic and it is preceded by a forward slash.



Figure 5-15: The use of (#) wild card in topic format

- ✓ myhome / groundfloor / livingroom / temperature
- ✓ myhome / groundfloor / kitchen / temperature
- ✓ myhome / groundfloor / kitchen / brightness
- ✗ myhome / **firstfloor** / kitchen / temperature

Figure 5-16: Illustration of valid wildcards in multi-level

A client subscribing to a topic with a multi-level wildcard is receiving all messages, which start with the pattern before the wildcard character, no matter how long or deep the topics will get. If you only specify the multilevel wildcard as a topic (#), it means that you will get every message sent over the MQTT broker.

In general, you are totally free in naming your topics, but there is one exception. Each topic, which starts with a \$-symbol will be treated specially and is for example not part of the subscription when subscribing to #. These topics are reserved for internal statistics of the MQTT broker. Therefore, it is not possible for clients to publish messages to these topics. At the moment there is no clear official standardization of topics that must be published by the broker. It is common practice to use \$SYS/ for all this information and a lot of brokers implement these, but in different formats. For example: \$SYS/broker/clients/connected.

## 5.2.4 MQTT connection establishment:

### 5.2.4.1 Client:

A MQTT client can be a publisher or a subscriber or both at the same time. It is any device from a micro controller up to a full-fledged server that has a MQTT library running and is connecting to an MQTT broker over any kind of network. This could be a really small and resource constrained device that is connected over a wireless network and has a library strapped to the minimum or a typical computer running a graphical MQTT client for testing purposes, basically any device that has a TCP/IP stack and speaks MQTT over it. The client implementation of the MQTT protocol is very straight-forward and really reduced to the essence. That's one aspect, why MQTT is ideally suitable for small devices

### 5.2.4.2 Broker:

The counterpart to a MQTT client is the MQTT broker, which is the heart of any publish/subscribe protocol. A broker can handle up to thousands of concurrently connected MQTT clients. The broker is responsible for receiving all messages, filtering them, deciding who is interested in it and then sending the message to all subscribed clients. Another responsibility of the broker is the authentication and authorization of clients.



### 5.2.4.3 MQTT connection:

The MQTT protocol is based on top of TCP/IP and both client and broker need to have a TCP/IP stack. The MQTT connection itself is always between one client and the broker, no client is connected to another client directly. The connection is initiated through a client sending a CONNECT message to the broker. The broker response with a CONNACK and a status code. Once the connection is established, the broker will keep it open as long as the client doesn't send a disconnect command or it loses the connection.

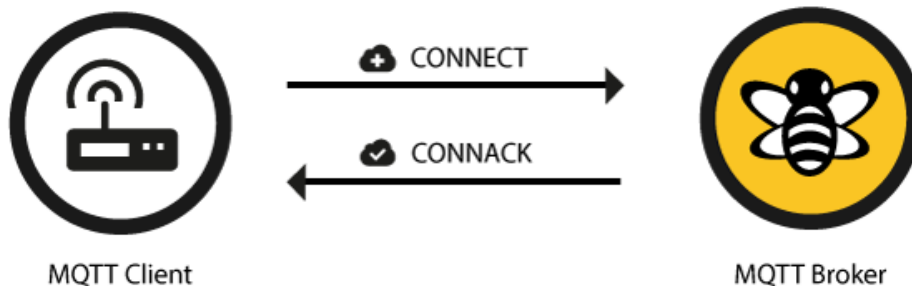


Figure 5-17: Message transfer between client and broker

#### CONNECT message:

It is sent from the client to the broker to initiate a connection. If the CONNECT message is malformed (according to the MQTT spec) or it takes too long from opening a network socket to sending it, the broker will close the connection. This is a reasonable behavior to avoid that malicious clients can slow down the broker.

#### CONNACK message:

When a broker obtains a CONNECT message, it is obligated to respond with a CONNACK message.

### 5.2.5 Features of MQTT protocol:

#### 5.2.5.1 Quality of service (QoS):

There are three different quality of service levels that determine how the content is managed by the MQTT protocol and act as an agreement between the sender and receiver of a message concerning the guarantees of receiving a message. Although higher levels of QoS are more reliable, they have more latency and bandwidth requirements, so subscribing clients can specify the highest QoS level they would like to receive.

**QoS 0 - at most once:** Client configured with QoS level 0 will publish messages only once. This is just a confirmation message which will not be stored and redelivered by the sender or responded by the receiver client. It guarantees a best effort delivery and is often called “fire and forget” and provides the same guarantee as the underlying TCP protocol. QoS 0 doesn't negatively affect device performance but has the possibility of message loss.



Figure 5-18: Control information for QoS 0

**QoS 1 - at least once:** Client configured with QoS level 1 will publish messages more than once. It is guaranteed that a message will be delivered at least once to the receiver. This message will be stored by the sender until a confirmation message is received from the other Client.

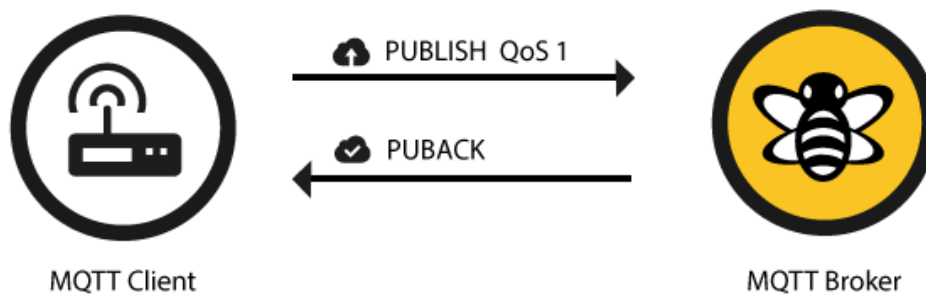


Figure 5-19: Control information for QoS 1

The sender will store the message until it gets an acknowledgement in form of a PUBACK command message from the receiver. The association of PUBLISH and PUBACK is done by comparing the packet identifier in each packet. If the PUBACK isn't received in a reasonable amount of time the sender will resend the PUBLISH message. If a receiver gets a message with QoS 1, it can process it immediately, for example sending it to all subscribing clients in case of a broker and then replying with the PUBACK.

**QoS 2 - exactly once:** Client configured with QoS level 2 ensures publishing the message exactly once and also that it is received by the counterpart. This is the most secure level of publishing messages but causes low performance as it is slow.

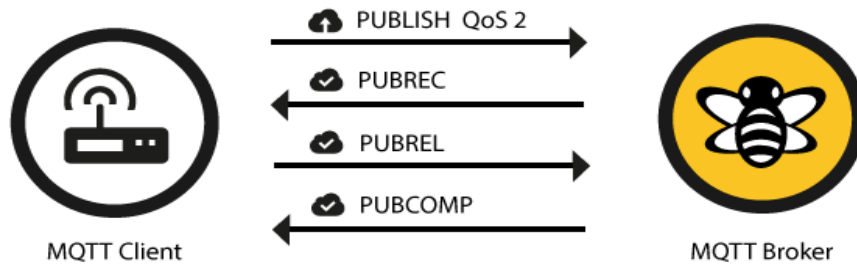


Figure 5-20:Control information for QoS 2

The receiver will store a reference to the packet identifier until it has send the **PUBCOMP**. This is important to avoid processing the message a second time. When the sender receives the **PUBREC** it can safely discard the initial publish, because it knows that the counterpart has successfully received the message. It will store the **PUBREC** and respond with a **PUBREL**.

After the receiver gets the **PUBREL** it can discard every stored state and answer with a **PUBCOMP**. The same is true when the sender receives the **PUBCOMP**.

When the flow is completed both parties can be sure that the message has been delivered and the sender also knows about it. Whenever a packet gets lost on the way, the sender is responsible for resending the last message after a reasonable amount of time. This is true when the sender is a MQTT client and also when a MQTT broker sends a message. The receiver has the responsibility to respond to each command message accordingly.

There is a simple rule when considering performance impact of QoS. It is *“The higher the QoS, the lower the performance”*. MQTT provides flexibility to the IOT devices, to choose appropriate QoS they would need for their functional and environment requirements.

### 5.2.5.2 Last Will and Testament:

MQTT is often used in scenarios were unreliable networks are very common. Therefore, it is assumed that some clients will disconnect ungracefully from time to time because they lost the connection, the battery is empty or any other imaginable case. Thus, using this feature if a connected client has disconnected gracefully, it can notify other clients about its disconnection. This lessens the probability of missing transmissions due to communicating with a device that’s no longer available for communication.

Each client can specify its last will message (a normal MQTT message with topic, retained flag, QoS and payload) when connecting to a broker. The broker will store the message until it detects that the client has disconnected ungracefully. If the client disconnects abruptly, the broker sends the message to all subscribed

clients on the topic, which was specified in the last will message. The stored LWT (Last Will and Testament) message will be discarded if a client disconnects gracefully by sending a **DISCONNECT** message.

## 5.2.6 MQTT implementation:

### 5.2.6.1 Flow of implementation:

- The NodeMCU (Arduino) client using the esp8266 WIFI module embedded in it, which will be discussed in details in the next chapter, tries to **CONNECT** to the WIFI network.
- Then, the NodeMCU tries to **CONNECT** to the raspberry pi broker.
- The raspberry pi, that acts as the broker and also the other client here, responds to the **CONNECT** message with a **CONNACK** message.
- The NodeMCU client creates three different topics to publish the data through them. These topics are the: the temperature, humidity and moisture topic. Then, it sends a **PUBLISH** command to the raspberry pi broker.
- On the other hand, the raspberry pi executes a **SUBSCRIBE** command to these topics to receive the data.
- Both the NodeMCU client and the raspberry pi client periodically **reconnect** to keep the connection established.

### Client's libraries:

Client's libraries are available in almost all popular languages now. The paho MQTT python client library is one of the best resources. It provides a client class which enable applications to connect to an MQTT broker to publish messages, and to subscribe to topics and receive published messages. It also provides some helper functions to make publishing messages to an MQTT server very straightforward.

### Mosquitto Broker:

Mosquitto is an open-source message broker service that implements the MQTT protocol. The Mosquitto project also provides a C library for implementing MQTT clients, and the very popular `mosquitto_pub` and `mosquitto_sub` command line MQTT clients.

### 5.2.6.2 MQTT related functions:

#### **on\_connect(client, userdata, flags, rc):**

Creates a client instance using the default parameters, assigns a callback to be called once a successful connection has occurred, and starts the connection. The callback for this function is when the client receives a **CONNACK** response from the server.

An example for using this function:

```
def on_connect(client, userdata, flags, rc):
    print("CONNACK received with code %d." % (rc))
    client.subscribe(temperature_topic)
    client = paho.Client()
    client.on_connect = on_connect
```

### **subscribe(topic\_name):**

Subscribing in any of the topics, `on_connect()` means that if we lose the connection and reconnect again then subscriptions will be renewed.

### **on\_message(client, userdata, msg):**

This function is called back for each message received from the server and the `msg` variable is an MQTT message class, which has members' topic, payload, QOS and other optional fields.

An example for using this function:

```
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.qos)+" "+str(msg.payload))
```

### **mqtt.client():**

It creates a MQTT client, it's default structure is as follows: `mqtt.client(clientid, keep alive, username, password, clean session)`. It takes the client ID, keep alive timer value and some other optional parameters.

**Keep alive:** MQTT defines an alive monitoring process with the keep alive timer to verify the connection is active. The connection will be disconnected if the client does not send any message for the duration of  $1.5 * \text{the value of the keep alive timer set on the CONNECT command}$ . If the client is not going to send any message within this duration, it needs to send the PINGREQ command. More information can be found at the documentation of the MQTT library that is used.

### **connect(host="local host", port\_number, keep alive value):**

It sends an MQTT CONNECT packet. The function takes as arguments, the client ID, and the port number to connect through it, which is 1883, a reserved port number for TCP/IP connections and it is used for MQTT connections and the keep alive timer value.

### **client.loop\_forever():**

It Will block processing the network traffic and reconnects automatically as necessary. This function is most useful if you are only subscribing to the broker and acting on the messages you receive.

### 5.2.7 Python script on raspberry pi:

For the main python script on the raspberry pi to work with different functions, some libraries must be imported. Here is a definition for these imported libraries:

```
import RPi.GPIO as GPIO
```

The RPi.GPIO module is used as the driving force behind our Python examples. This set of Python files and sources is included with Raspbian. By including it you can refer to it as just GPIO through the rest of your script.

```
import paho.mqtt.client
```

The core of the client library is the client class which provides all of the functions to publish messages and subscribe to topics.

```
import time
```

Time intervals are floating-point numbers in units of seconds. Particular instants in time are expressed in seconds since 12:00am, January 1, 1970(epoch). There is a popular time module available in Python which provides functions for working with times, and for converting between representations.

```
import sys
```

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter

```
import MySQLdb
```

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard. MySQLdb is a thread-compatible interface to the popular MySQL database server that provides the Python database API.

```
import csv
```

The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases. There is no “CSV standard”, so the format is operationally defined by the many applications which read and write it. The lack of a standard means that subtle differences often exist in the data produced and consumed by different applications. These differences can make it annoying to process CSV files from multiple sources. Still, while the delimiters and quoting characters vary, the overall format is similar enough that it is possible to write a single module which can efficiently manipulate such data, hiding the details of reading and writing the data from the programmer.

The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats.

The csv module’s reader and writer objects read and write sequences. Programmers can also read and write data in dictionary form using the DictReader and DictWriter classes.

### **import urllib2**

This module provides a high-level interface for fetching data across the World Wide Web. In particular, the urlopen() function is similar to the built-in function open(), but accepts Universal Resource Locators (URLs) instead of filenames. Some restrictions apply — it can only open URLs for reading, and no seek operations are available.

### **import os**

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

### **import threading**

This module constructs higher-level threading interfaces on top of the lower level \_thread module.

### **import requests**

Requests allows you to send organic, grass-fed HTTP/1.1 requests, without the need for manual labor. There's no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic, thanks to urllib3.

After importing the different libraries, we connect to the database using “MySQLdb.connect” and it takes the database name and the username and password and the database which we want to store the data in. Then compile the connection function with “conn.cursor”.

Finally, the python script inserts the received readings from the local node in a CSV file to be the input for the machine learning algorithm as the test set. CSV file is opened by the **open ()** that takes the type of the opening file and its name, then the desired columns to be filled is specified through the **csv.writer ()** function. Then filling the CSV file row by row with the data every reading time.

## 5.3 GSM/GPRS Module

### 5.3.1 Introduction

**GSM** (Global System for Mobile communications) is a standard developed by the **ETSI** (European Telecommunications Standards Institute) to describe the protocols for **2G** (second generation) digital cellular networks, it was first deployed in Finland in December 1991, the standard originally described as a digital, **CS** (circuit switched) network optimized for full duplex voice telephony, but later was evolved to be a **PS** (packet switched) network after the integration with the **GPRS** (General Packet Radio Service) technology, the technologies used in 2G are either **TDMA/FDMA** (Time Division Multiple Access over Frequency Division Multiple Access) or **CDMA** (Code Division Multiple Access) which allocates a special code to each user so as to communicate over a multiplex physical channel, the 2G system operates in the 900 MHz and 1.8 GHz bands throughout the world with the exception of the Americas where they operate in the 850 MHz and 1.9 GHz band.

**TDMA/FDMA** explanation: **FDMA** is a standard that allows multiple users to access a specific radio frequency band and eliminates the interference of message traffic; by splitting the available amount of bandwidth which is 25MHz in 2G case into 124 carrier frequencies of 200 kHz each. Each frequency is then divided using the **TDMA** scheme into eight time slots and allows eight simultaneous calls on the same frequency, **TDMA** breaks down data transmission, such as a phone conversation, into fragments and transmits each fragment in a short burst, assigning each fragment a time slot. With a cell phone, the caller does not detect this fragmentation; so 2G wireless technologies can handle some data capabilities such as fax and short message service, but it is not suitable for web browsing and multimedia applications due to the limited speed. An evolution which is the so-called 2.5G (**GPRS**) systems enhanced the data capacity of **GSM** and mitigate some of its limitations.

**GPRS** is referred to as the in-between both 2G and 3G, it provides both a means to aggregate radio channels for higher data bandwidth and the additional servers required to off-load packet traffic from existing **GSM** circuits, theoretical maximum speeds of **GPRS** are about ten times as fast as current Circuit Switched Data services on **GSM** networks in condition that all 8 time slots are allocated to one user. However, it should be noted that it is unlikely that a network operator will allow all time slots to be used by a single **GPRS** user. Additionally, the initial **GPRS** terminals (phones or modems) are only supporting only one to four time slots; thus the bandwidth available to a **GPRS** user is limited, it is a best-effort service, implying variable throughput and latency that depend on the number of other users sharing the service concurrently, as opposed to circuit switching, where a certain quality of service (**QOS**) is guaranteed during the connection.



	2G	2.5G (GPRS)	3G
Data Rate	14.4 Kbps	114 Kbps	The max speed is around 2 Mbps for non-moving devices
Billing	Charges based on long distance calls and real time billing.	Charges for the amount of data sent rather than the connection time or distance.	Increased data transmission at a lower cost.
Challenges	Cannot Handle Data and Video communication.(Internet)	Data cannot be sent while a voice call is in progress.	It required a new infrastructure which does not cover all areas of interest.

Table 3-1: Comparison between 2G, 2.5G and 3G

### 5.3.2 Why using GSM/GPRS?

From (Table 3-1) 3G offers high transmission speeds but its drawback is in the installation costs of its infrastructure so it isn't quiet spread all over Egypt; especially at the Agricultural areas which are commonly rural; a rural area or countryside is a geographic area that is located outside overcrowded towns and typical rural areas have a low population density and small settlements in other words not the best investment for any network operators; so NTRA made an agreement with all network operators in Egypt to provide coverage to those areas to make sure that all citizens have coverage in case of any circumstances as calls is an essential service like water and transportation in case of emergencies and must be provided to everyone, so NTRA helped the network operators by lowering the license cost for 2G, 2.5G network in those areas or by partially funding the infrastructures to be installed; and for the rural area high speed internet coverage is not essential to be provided that is why mainly the agriculture areas are covered by 2G, 2.5G networks.

### 5.3.3 Main Features of GSM/GPRS:

- Improved spectrum efficiency compared to 1G.
- Allowed International roaming.
- Compatibility with integrated services digital network (ISDN).
- Real time clock with alarm management.
- High-quality speech.
- Short message service (SMS).
- Uses encryption to make phone calls more secure.

### 5.3.4 GSM Modem

A GSM modem is a standard cellular radio that can be used to allow a computer or any other processor to transmit IP data and SMS over the GSM cellular network without additional peripherals, a GSM modem requires a SIM card to be operated under network range subscribed by the sim card network operator. It can be connected to a computer through serial, USB or Bluetooth connection, the bare GSM modem does not, in general, have any antenna, power supply, microphone, speaker, and these must be externally connected. So the module is the basic core innards of a GSM handset.

UART is an asynchronous serial communication protocol, meaning that it takes bytes of data and transmits the individual bits in a sequential fashion, Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver agree on timing parameters in advance and special bits called 'start bits' are added to each word and used to synchronize the sending and receiving units. UART is commonly used on the Pi as a convenient way to control it over the GPIO, or access the kernel boot messages from the serial console (enabled by default).

The callings or the transmissions of (SMS, IP data) are managed by a program on an external processor or an application program on a computer that controls the GSM module which defines what data or SMS is to be sent, not on anything inherent in a GSM; that is why for all different modules the same AT commands are used, these commands are sent by the controller or the processor. The modem sends back a replay after it receiving and executing an AT command.

### 5.3.5 SIM900 GPRS Arduino Shield

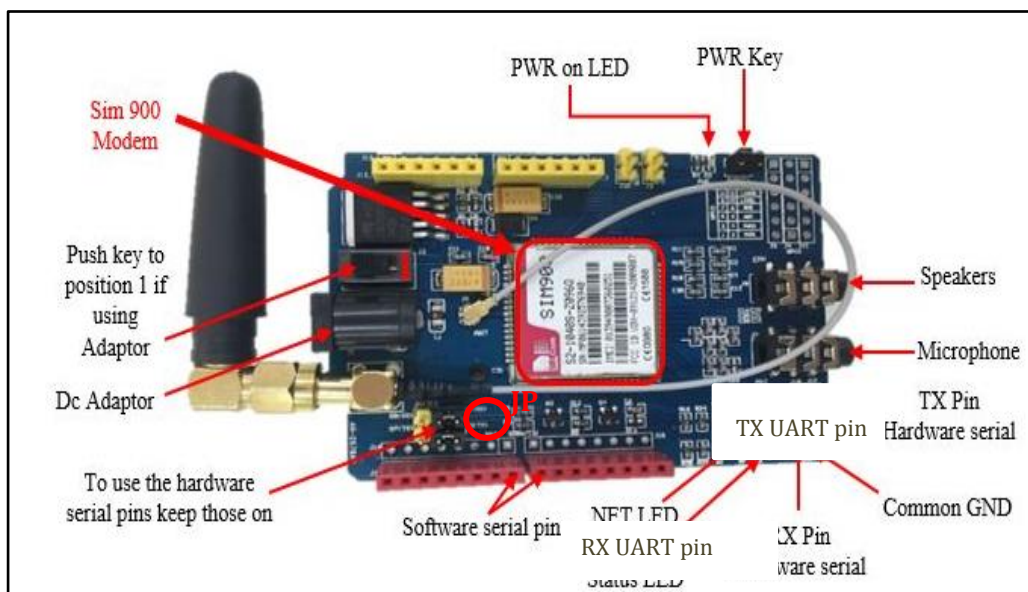


Figure 3-1: Sim900 GPRS arduino shield Front view

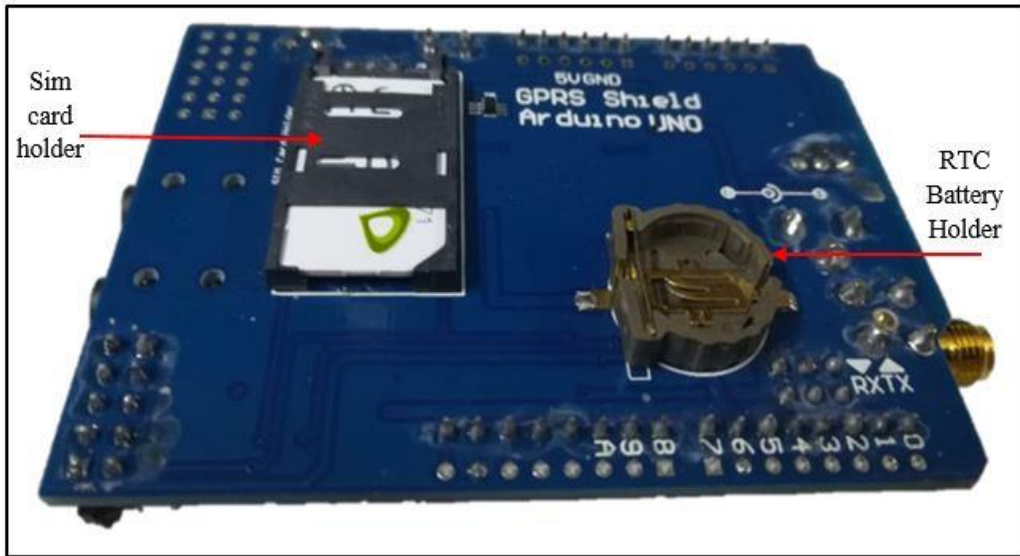


Figure 5-13 Sim900 GPRS Arduino shield back view

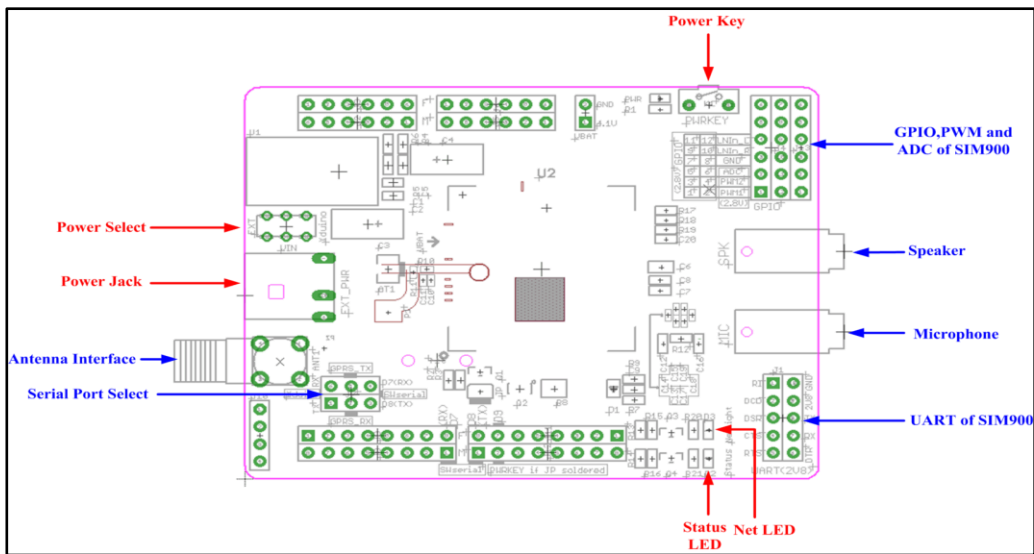


Figure 5-14 A Schematic of Sim 900 GPRS Arduino shield

### 5.3.5.1 Features

Figures 3-1, 3-2 show a front and a back view of the module used.

- It is based on SIMCom's SIM900 Modem.
- It is a Quad-Band 850/900/1800/1900 MHz.
- The module supports software power on and reset besides the power key.
- Its GPRS is **configured** and controlled via its **UART** breakout pins or its **serial software pins**, Controlled by AT commands - Standard Commands: GSM 07.07 & 07.05 or the Enhanced Commands (SIMCOM AT Commands).
- Supports Short Message Service.
- Embedded TCP/UDP stack - allows to upload data to a web server.
- Speaker and Headphone jacks.

- SIM Card holder and GSM Antenna are present onboard.
- 12 GPIOs, 2 PWMs and an ADC (all 2.8 volt logic) to augment Arduino.
- Low power consumption - 1.5mA(sleep mode)
- Industrial Temperature Range -40°C to +85 °C
- It needs a 5V-2A, 9V-1A or 12V-1A DC power adaptor to work properly.

Even though Sim900 is an EOL product (End of Life for a product) but it was one of the few options offered to be bought locally; for long term industrial usages it is advised to use 800x series as it is still technically supported by the manufactory and offers other features like Bluetooth, FM radio.

This shield is made for Arduino so there're two choices for GPRS shield to communicate with the Arduino board, while plugging the two jumpers on to the Software serial or Hardware serial positions. If using Software serial, D7 and D8 of the Arduino will be used by SIM900 GPRS Shield combined with a serial library to define the basic setup for the connection like the baud rate; if using Hardware serial, D0 (RX) and D1 (TX) of the Arduino will be used, but it is advised to use the software serial pins and leave the hardware serial pins of the Arduino so in case of any wrong configuration or input commands, the AVR of the Arduino stay well configured, but cautions needed as not all Arduino boards support D7, D8 as a software serial pins.

For a RPI3 usage, just use the UART breakout pins.

### ***5.3.5.2 The Prototype Applications***

It can be used to perform a call, send a text, upload or download data from a web server. this prototype offers two options, the first is to send a warning SMS when temperature range during a consecutive period of time exceeds a certain threshold and offers different suggestions depending on the current case and its similarities with a previous recorded cases, the second option is to give the user an access on the climatically and soil condition changes of his agriculture field to keep a close eye without being near the field.

### ***5.3.5.3 Cautions***

- Make sure the SIM card is unlocked.
- If the module goes out of power after couple of seconds after power up then check the power source to be at least 5V-2A.
- Make sure to connect the UART GND pin while using an external power adaptor otherwise, the internet session will not be initiated correctly.
- For uploading or downloading data from the internet make sure, that the sim card has internet data or a 603 error (cannot resolve DNS) or 601 error (Network error) will pop up while operating.
- For sending SMSs or making calls make sure the sim card is charged with enough credit.

- The product is provided without an insulating enclosure. Please observe ESD precautions especially in dry (low humidity) weather.
- The factory default setting of the GPRS Shield UART is 115200 bps (It can be changed by AT commands).

#### 5.3.5.4 Getting Started

##### Connection Steps:

- Power jack is connected to 5V-2A Dc power adaptor.
- Power select is moved to position 1 showed in Figure 3-3 to configure the module to withdraw the power from the power jack.
- UART breakout pins are connected to TTL to USB module as raspberry pi3 board contain only 1 pair of UART pins but 4 USB input, the baud rate of the connection need to be defined then programed by using serial library at RPI3, the connection is showed in Figure 3-5.
- The module can be powered manually by pressing the power key for 1 second, or it can be powered on or off through D9 pin by adding software triggering in the firmware, note that the JP for pin 9 on the shield must be soldered in order to use software power up/down.
- The UART GND pin need to be connected to the RPI3 or the common GND node of the circuit if an external adaptor is used or the serial commands will not be sent to the modem, the sim card is installed at the sim card holder at the back side of the module.
- Note if the other GND of the GPRS shield is used instead of UART GND the internet connection whether download or upload will not be available.

##### Led status:

LED	State	Function
Status	Off	Power Off
	On	Power On
Netlight	Off	SIM900 is not working
	64ms On/800ms Off	SIM900 does not find the network
	64ms On/3000ms Off	SIM900 finds the network
	64ms On/300ms Off	GPRS communication

Figure 5-15 The Led status meaning

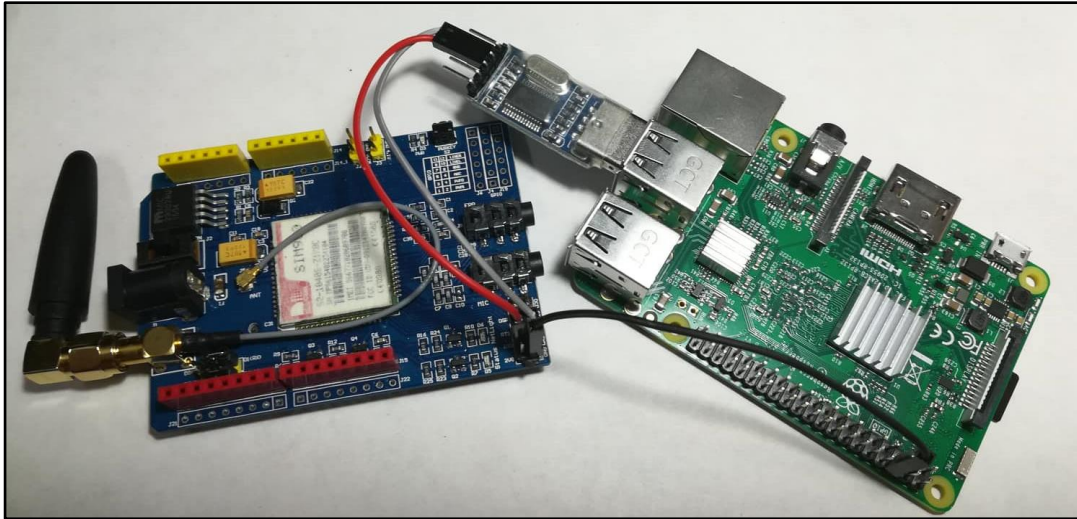
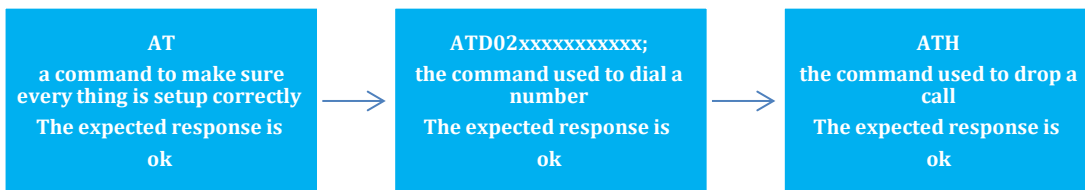


Figure 5-16 Connecting the GPRS modem to the RPI3 through TTL to USB modem

Using AT Commands with the GPRS Module:

Prepare the Sim900 AT-Commands PDF, it can be downloaded from SIMCOM website.

1. To perform a call, the following chart is based on SIMCOM commands to initiate and drop a call and is needed to be executed at the same sequence.



```
import serial
```

At first the serial library is imported so as to set the baud rate to be 115200 of the connection.

```
def calling(number,path='/dev/ttyUSB0'):
```

The number to be called is passed as a string variable and the serial path is defined so as the RPI3 can direct the AT commands to the GPRS modem through TTL to USB connection.

```
ser = serial.Serial(path, baudrate = 115200, timeout=1)
```

Opens a serial connection and setting its baud rate to be 115200.

```
ser.write ('AT\r')

ser.write("\x0D\x0A")

print ser.read(10)
```

Expected response is: OK

(AT) checks on the connection of the module if the GPRS modem does not response back with ok, the connections must to be checked.

(\r) expresses pressing the enter key and that is needed to be performed after each command so as the module receives and performs the commands correctly.

time.sleep(1) is to give the module enough time to perform the command and replay back, without overlapping between consequent commands at the serial monitor or the terminal.

ser.write("\x0D\x0A"): gives the module the permission to replay back through the connection.

print ser.read(10): prints message received from the module.

```
ser.write (' ATE0\r')
```

This command is used to disable the echo of the module without it, the response will not be just an OK, but it will type back the command and it' response.

```
ser.write('ATD%s;\r' % number)
```

```
ser.write("\x0D\x0A")
```

```
print ser.read(10)
```

Expected response is: OK which means the number is currently being dialed.

('ATD%s;\r' % number): ATD command is used to dial a given number based by the variable number in the previous code line.

Note ";" is a must after the number otherwise no call will be initiated.

The previous line of code shows that; the variable is passed from the definition of the function only the type of the variable must be defined in the definition of the calling function and it is defined to be string by using (%s).

time.sleep(5) in this command has 2 functionalities; one of them is the same as previous command and the other one is to separate between the start of ringing and hanging up the call.

```
ser.write('ATH')
```

Expected response is: OK which means the call is dropped.

This is the command used to hang up the call.

```
ser.close()
```

Terminates the serial connection.

Gathering the code lines all together:

```
import serial

def calling(number,path='/dev/ttyUSB0'):

    ser = serial.Serial(path, baudrate = 115200, timeout=1)

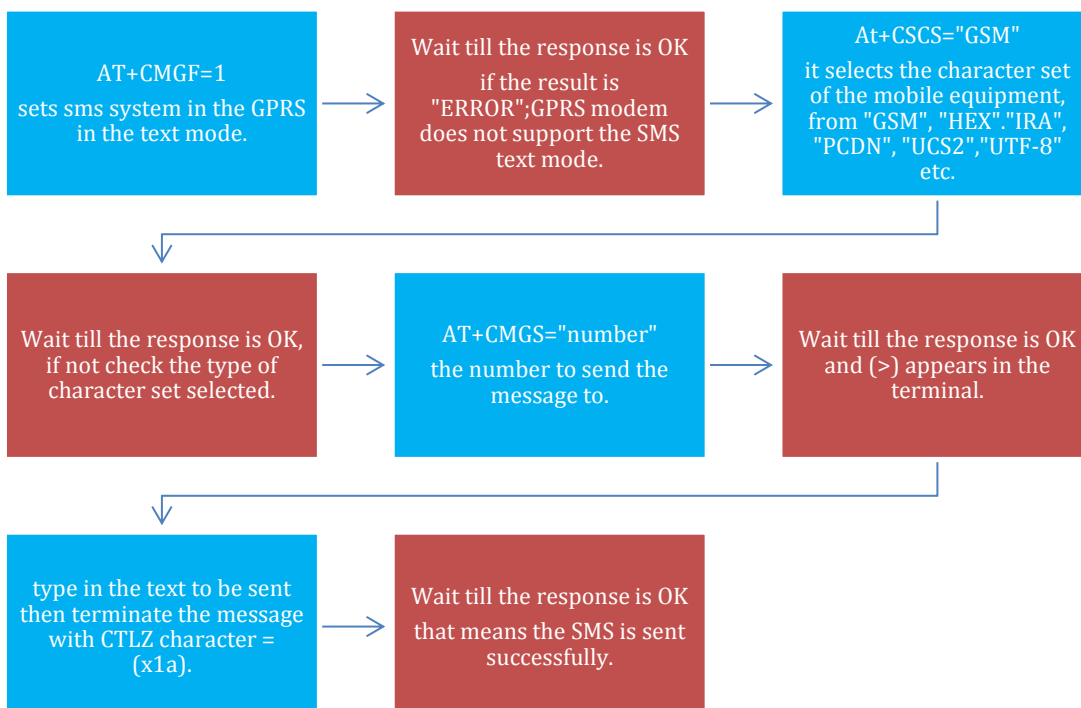
    ser.write ('AT\r')

    time.sleep(1)

    ser.write (' ATE0\r')

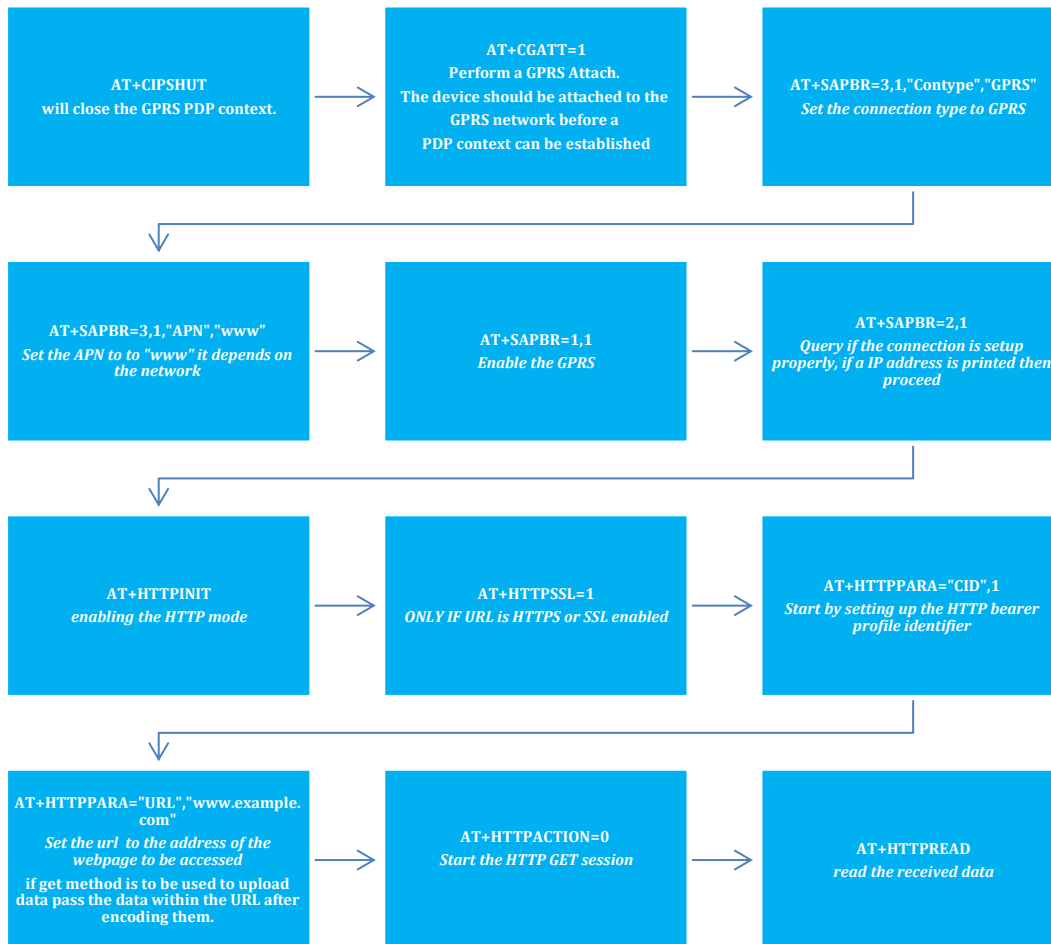
    ser.write('ATD%s;\r' % number)
```

2. A similar sequence can be used to send an SMS to a certain number, by following the flowchart below, the commands are in the blue squares and the response in case of disabling the ECHO is in the orange squares.





3. The following chart is to use the internet capabilities of the GPRS module in terms of HTTPGET commands, which allows both downloading and uploading data from/ to an internet server. Although SIM900 PDF mentioned that HTTPPOST can also be used to upload data and its commands and steps are specified on the PDF, it is more complex to be executed with certain limitations and additional precautions, so if the data is not important for example passwords, just use the HTTPGET method.



Note: if the choice was to use HTTPGET method to upload the data, additional library and function are needed so as to encode the data into a way suitable to be send with in the URL.

```
import urllib # that is for python 2.7 but for python 3 please refer to their official
```

Import the library which contains urlencoding functions.

If a reading from a moisture detector sensor is a string with space

For example: m1="Water Detected", no spaces are allowed in the URL; that is the reason to use URL encoding to translate the spaces into special characters; could be passed through URL.

```
moist=urllib.quote_plus(m1)
```

(moist) is the encoded form of m1 that is suitable to be sent with in a URL.

The URL quoting functions focus on taking data and making it safe for use as URL components by quoting special characters and appropriately encoding non-ASCII text. They also support reversing these operations to recreate the original data from the contents of a URL component if that task isn't already covered by the URL parsing functions above, Replace special characters in string using the %xx escape. Letters, digits, and the characters '\_.-~' are never quoted. By default, this function is intended for quoting the path section of URL.

## 6 Chapter 6: Node MCU

### 6.1 Node MCU developed on ESP8266:

The Node MCU is an open source software and hardware development environment that is built around a SoC called the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications. The ESP8266, designed and manufactured by Espressif Systems, contains all crucial elements of the modern computer: CPU, RAM, networking (Wi-Fi), and even a modern operating system and SDK. ESP8266 advertises itself as a self-contained Wi-Fi networking solution offering itself as a bridge from existing micro controller to Wi-Fi and is also capable of running self-contained applications that makes it an excellent choice for IoT projects of all kinds. Espressif Systems produced different development boards having different features. The Node MCU is one of them.

As Arduino.cc began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the Arduino IDE so that it would be relatively easy to change the IDE to support alternate tool chains to allow Arduino C/C++ to be compiled down to these new processors. Some creative ESP8266 enthusiasts have developed an Arduino core for the ESP8266 Wi-Fi SoC This is what is popularly called the “ESP8266 Core for the Arduino IDE” and it has become one of the leading software development platforms for the various ESP8266 based modules and development boards, including Node MCUs.

### 6.2 ESP8266 Specifications:

Voltage and current consumption	3.3V and 10uA – 170mA
Flash memory attachable	16MB max (512K normal)
Processor type and speed	Tensilica L106 32 bit with 80-160MHz speed
RAM	32K instruction + 80K user data
GPIOs	17 (multiplexed with other functions)
Max concurrent TCP connection	5

### 6.3 Node MCU for IOT applications:

Lua is a lightweight, embeddable scripting language designed primarily for embedded use in applications. It's cross-platform and has a relatively simple C API. Node MCU is an eLua based firmware for the ESP8266 Wi-Fi SOC from Espressif. The firmware is based on the Espressif NON-OS SDK and uses a file system based on spiffs. The Node MCU project uses the SPIFFS filesystem to store files in the flash chip. using the Node MCU board with the Arduino IDE, the Lua firmware will be deleted and replaced by the sketch.

Node MCU development kit can use its GPIO pins to connect to other peripherals without help of any other micro-controller while simple ESP8266 can only be used to connect other micro-controllers like Arduino to internet with its inbuilt Wi-Fi chip. That means you can program Node MCU to work with other devices using GPIO along with connected Wi-Fi.

This makes Node MCU ideal for use in IoT projects as it can be programmed and then connected with other hardware like relays, buttons or others via GPIO and also to Wi-Fi. Over Wi-Fi they can contact with some more powerful machines like raspberry pi using protocols like MQTT. Node MCU supports the MQTT IoT protocol, using Lua to access the MQTT broker.

### 6.4 Node MCU versions:

There are various versions of Node MCU.

Generation	Version	"Common" Name
1st	0.9	v1
2nd	1.0	v2
2nd	1.0	v3

Figure 6-1 Node MCU versions

#### 6.4.1 1<sup>st</sup> generation (0.9/v1):

The original board but now outdated. Dev kit is usually sold with an outstanding yellow board and is very wide. Its 47mm x 31mm mean that it covers all 10 pins of a regular bread board which makes it very inconvenient to use. It comes with an ESP-12 module and 4MB flash memory.



Figure 6-2 1st generation Node MCU

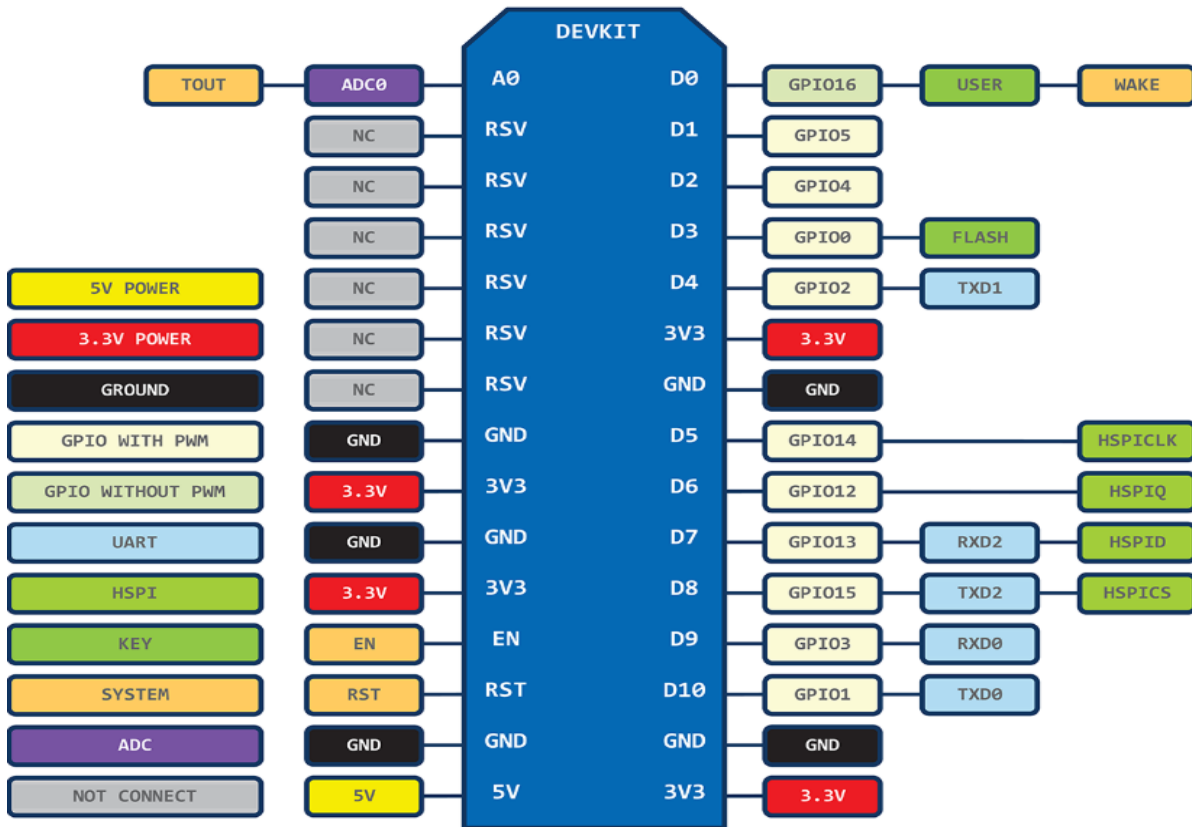


Figure 6-3 Pin layout of 1st generation Node MCU

### 6.4.2 2<sup>nd</sup> generation (1.0/v2):

V2 fixes the short comings of the initial board, it's narrower. The chip was upgraded from an ESP-12 to an ESP-12E. ESP-12E and ESP-12 are basically the same, the only difference is that ESP-12E is modified to have some extra pins.

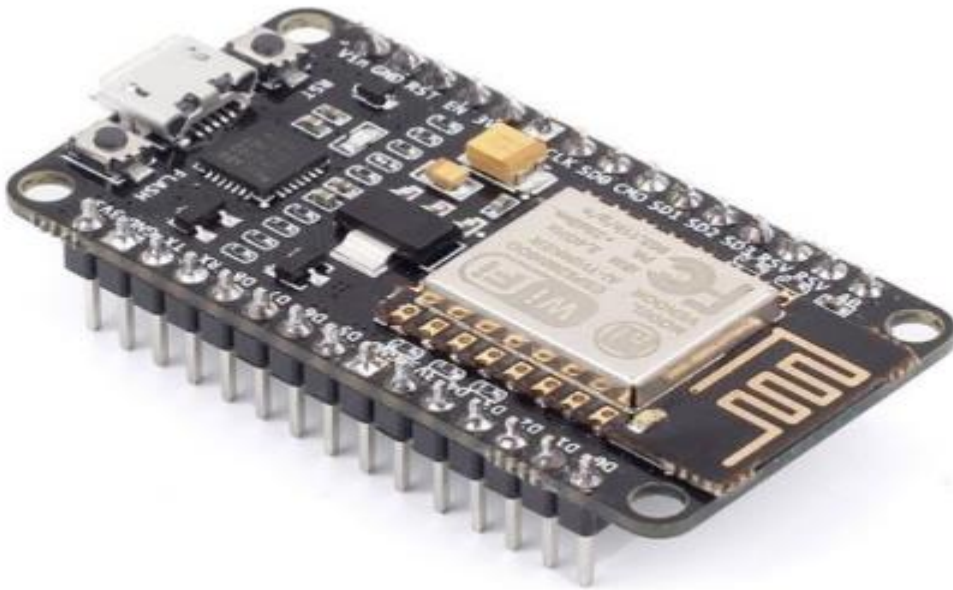


Figure 6-4 2nd generation Node MCU

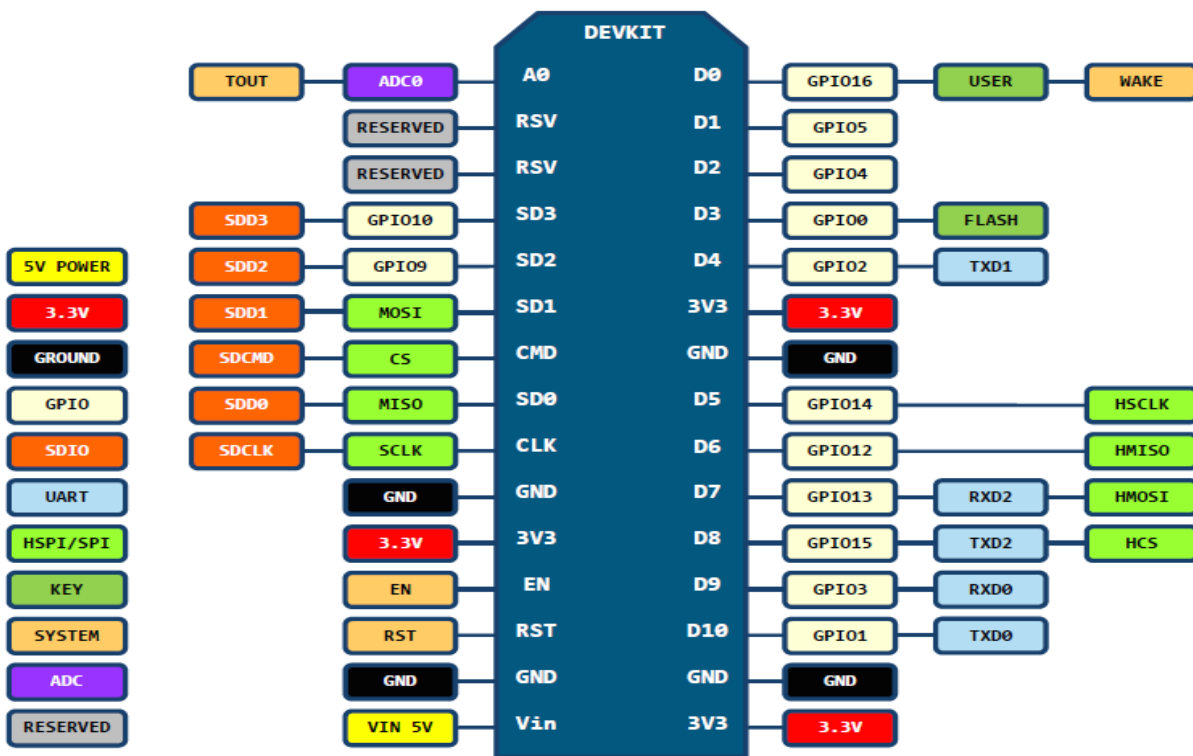


Figure 6-5 Pin layout of 2nd generation Node MCU

### 6.4.3 2<sup>nd</sup> generation (1.0/v3):

Node MCU haven't released a new specification so far. Hence, officially there's no 3rd generation board. Turns out that V3 is a "version" invented by producer LoLin to signify minor improvements to the V2 boards. Among others they claim their USB port to be more robust.



Figure 6-6 2nd generation V3 Node MCU

Comparing the pin layout of v2 and v3, there's only a tiny difference to the V2 layout. LoLin decided to use one of the two reserve pins for USB power out and the other for an additional GND.

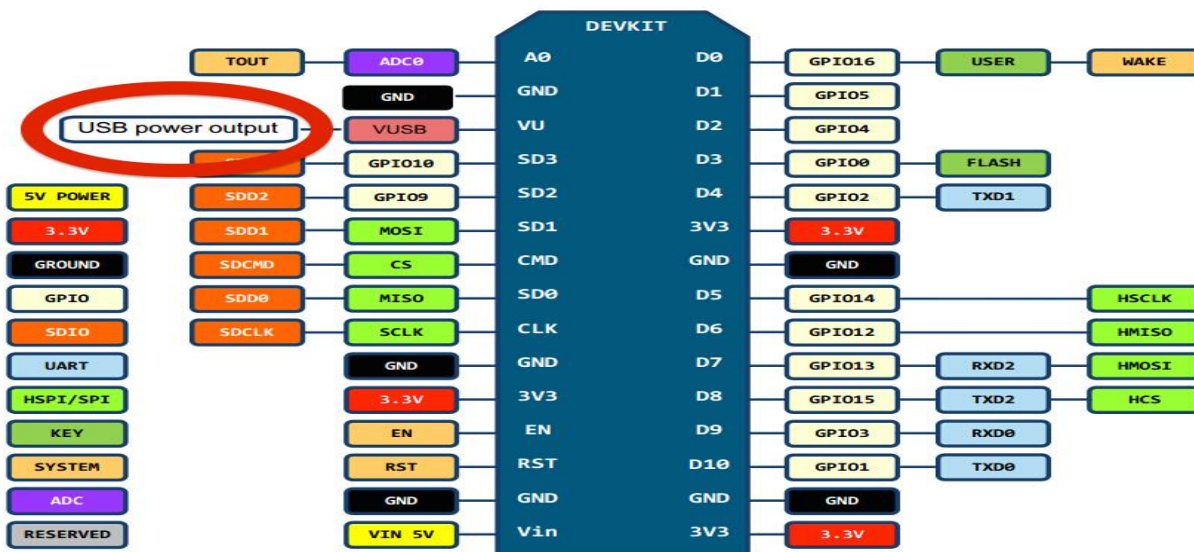


Figure 6-21 Pin layout of v3 Node MCU

### 6.5 x.5 selection of Node MCU version 3:

In this project, Node MCU V3 is used. For programming and powers; Node MCU is connected to computer using USB. The Status LED in Node MCU is basically used to show current status like flashing and booting. Reset/Flash buttons perform actions on board. GPIO pins connect board with other peripherals.

### 6.6 Programming Node MCU:

The Node MCU is ready to perform its two vital processes which are maintaining Wi-Fi connectivity, collecting sensor readings and sending it to the Raspberry pi.

In this part, Functions of Node MCU code used in Arduino IDE will be demonstrated; their inputs, their output and, each function's job. These functions are divided into two categories which are: MQTT and Wi-Fi related functions and, Sensors related functions.

### 6.6.1 MQTT and Wi-Fi related functions:

```
WiFi.begin(ssid, password);
```

This function is defined in "WiFiClient.h" in ESP8266WiFi library. It initializes the Wi-Fi library's network settings and provides the current status.

ssid: the SSID (Service Set Identifier) is the name of the Wi-Fi network you want to connect to.

password: WPA encrypted networks use a password in the form of a string for security. Returns:

WL\_CONNECTED when connected to a network

WL\_IDLE\_STATUS when not connected to a network, but powered on

```
Reconnect();
```

This is a user-defined function. It is used to reconnect to the Wi-Fi and, to the MQTT server. Using client.connected() function to check if the client still alive. It requires no input parameters as the Wi-Fi client is defined as a global variable.

Returns: None

```
client.setServer(mqtt_server, 1883);
```

This function is defined in "PubSubClient.h". It's used to connect Node MCU to server

Mqtt\_server: is the name of the mqtt server or the Ip address which is the raspberry pi in this project

1883: is the port number for mqtt using TCP/IP protocol

```
client.publish(topic, payload, retained);
```

This publish function is defined in "PubSubClient.h". It publishes the payload message from client to MQTT topic for server or broker to be able to subscribe this message.

Topic: message topic.

Payload: message content

Retained: it returns true if publish succeeded and false if publish failed.

### 6.7 Sensors related functions:

```
dht(DHTPIN, DHTTYPE);
```

This function is defined in "DHT.h". It is a constructor for a class called DHT. The DHT class is related to all the functionality of the DHTxx sensor.

DHTPIN: the pin connected to the output of the DHT sensor.

DHTTYPE: The type of the sensor.

Returns: None

```
dht.begin();
```

This function is defined in "DHT.h". It initializes the pin number and resets the variable which stores the last time read to adjust timings.

It has no input parameters.

Return: None.

```
Serial.begin(baud rate);
```

This is an Arduino defined function. It starts the serial connection using the USB comm port assigned from the menu bar -> tools -> port.

baud rate: is the rate at which information is transferred in a communication channel.

Returns: None

```
dht.readHumidity();
```

This function is defined in "DHT.h". It is used to extract the humidity value from the received data from the sensor.

It has no input parameters.

Returns: humidity as a percentage.

```
dht.readTemperature();
```

This function is defined in "DHT.h". It is used to extract the temperature value from the received data from the sensor.

Returns: temperature in Celsius.

```
analogRead(sense_Pin);
```

This function senses the analog pin (Ao) to measure the moisture level by converting the output voltage to value from 0 to 1023 then mapping this value to percentage from 0 to 100

Sense pin: the input parameter measure output voltage of moisture value

Returns: moisture level

```
Serial.print(val);
```

This is an Arduino defined function. It prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as it is.

val: the value to print -any data type

```
Serial.println(val)
```

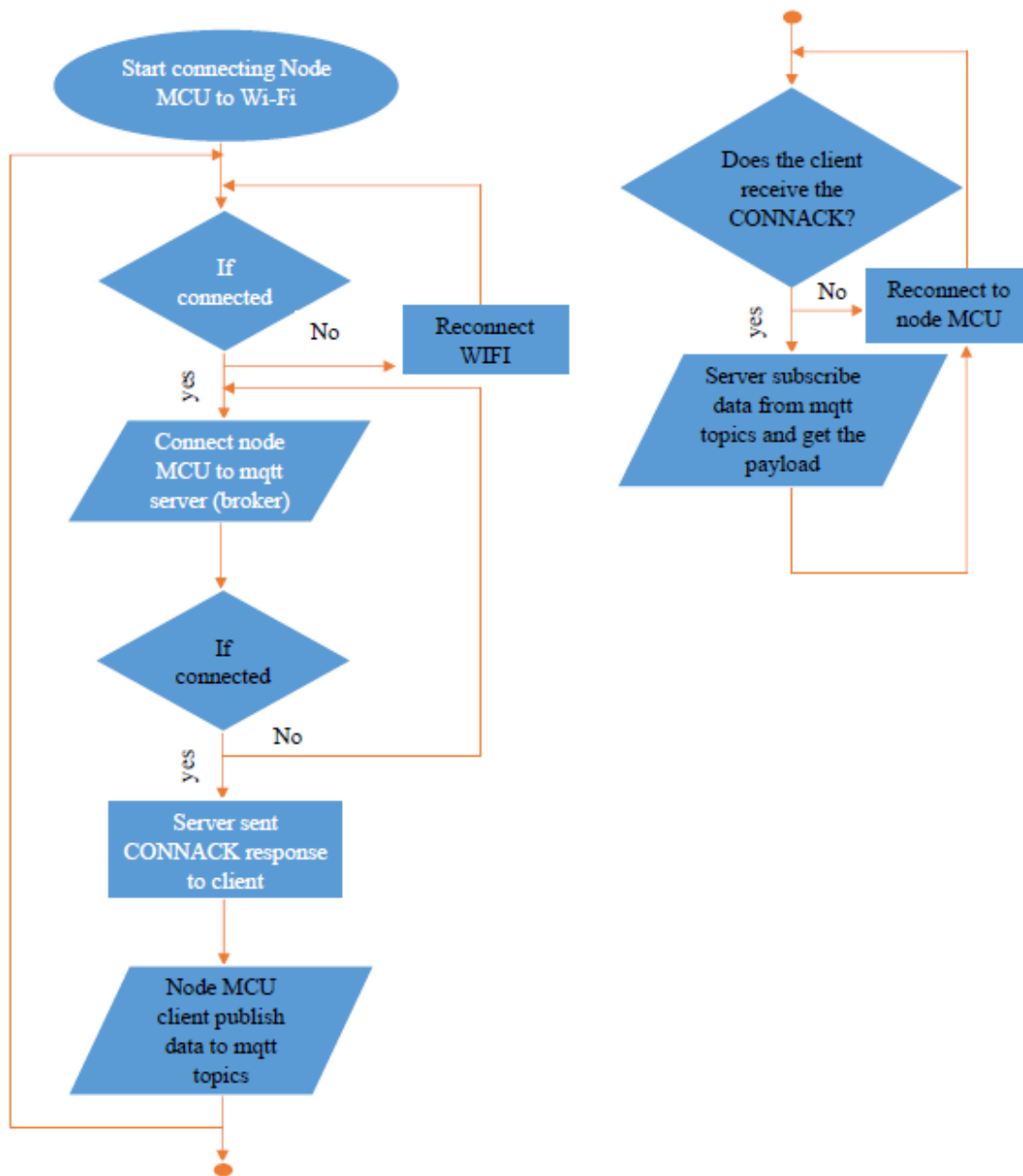
This is an Arduino defined function. It prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n').

This command takes the same forms as Serial.print().

val: the value to print -any data type.



## 6.8 flow chart illustration:



## 7 Chapter 7: Application Layer

### 7.1 Logical Gateway:

The gateway operates at the network layer (Layer 3) of the OSI Model. The gateway is used when transmitting packets. When packets are sent over a network, the destination IP address is examined. If the destination IP is outside of the network, then the packet goes to the gateway for transmission outside of the network. The gateway is on the same network as end devices. The gateway address must have the same subnet mask as host devices. Each host on the network uses the same gateway.

The gateway should have a static address, as changing the address would cause packets not to be delivered. The gateway is typically assigned either the highest or lowest network address. This is not a requirement, but many organizations use a consistent addressing scheme to facilitate network planning.

### 7.2 Physical Gateway:

The gateway also operates at the data link layer (Layer 2) of the OSI network model. The physical gateway address is called the media access control (MAC) address. The physical address is assigned when the device is manufactured and cannot be changed. When a frame is sent to a device not on the local network, the gateway's MAC address is used in the frame header.

#### **Server Types:**

##### Proxy Server:

A proxy server sits between a client program and an external server to filter requests, improve performance, and share connections.

##### Mail Server:

Almost as ubiquitous and crucial as Web servers, mail server's move and store mail over corporate networks through LANs, WANs and across the Internet.

### Web Server:

A Web server serves static content to a Web browser by loading a file from a disk and serving it across the network to a user's Web browser. This entire exchange is mediated by the browser and server talking to each other using HTTP.

### Application Server:

Application servers occupy a large chunk of computing territory between database servers and the end user, and they often connect the two.

### Real-Time Communication Server:

Real-time communication servers, formerly known as chat servers or IRC Servers, and still sometimes referred to as instant messaging (IM) servers, enable large numbers users to exchange information near instantaneously.

### FTP Server:

File Transfer Protocol makes it possible to move one or more files securely between computers while providing file security and organization as well as transfer control.

### Telnet Server:

A Telnet server enables users to log on to a host computer and perform tasks as if they're working on the remote computer itself.

## **7.3 Cloud gateway**

A cloud storage gateway is a network appliance or server which resides at the customer premises and translates cloud storage APIs to block-based storage protocols. Cloud storage gateways enable companies to integrate private cloud storage into applications without moving the applications into a public cloud thereby simplifying data protection.

A cloud server is a server that is built, hosted and delivered through a cloud computing platform over the Internet. Cloud servers possess and exhibit similar capabilities and functionality to a typical server but are accessed remotely from a cloud service provider.

A cloud server is primarily an Infrastructure as a Service (IaaS) based cloud

service model. There are two types of cloud server:

1. Logical
2. Physical

A cloud server is considered to be logical when it is delivered through server virtualization. In this delivery model, the physical server is logically distributed into two or more logical servers, each of which has a separate OS, user interface and apps, although they share physical components from the underlying physical server. Whereas the physical cloud server is also accessed through the Internet remotely, it isn't shared or distributed. This is commonly known as a dedicated cloud server.

**Key benefits of cloud servers:**

- Flexibility and scalability; extra resource can be accessed as and when required
- Cost-effectiveness; whilst being available when needed, clients only pay for what they are using at a particular time
- Ease of set up; Cloud servers do not require much initial setup
- Reliability; due to the number of available servers, if there are problems with some, the resource will be shifted so that clients are unaffected.

**Cloud server hosting**

Cloud server hosting is when hosting services are made available to customers on demand via the Internet. Rather than being provided by a single server or virtual server, cloud server hosting services are provided by multiple connected servers that comprise a cloud. Cloud server hosting is also sometimes referred to as cluster server hosting or server on- demand hosting.

Cloud server hosting offers the advantages of increased accessibility and reliability, seamless scalability and potential cost savings, as customers are freed from having to invest in on-premises servers and hardware, and they pay only for the resources they consume. On the other hand, security and lack of access and full control are potential concerns with cloud server hosting.

## **7.4 IoT Modular Gateway:**

An Internet of Things (IoT) Gateway provides the means to bridge the gap between devices in the field (factory floor, home, etc.), the Cloud, where data is

collected, stored and manipulated by enterprise applications, and the user equipment (smart phones, tablets etc.). The IoT Gateway, provides a communication link between the field and the Cloud and can also offer local processing and storage capabilities to provide offline services and if required real time control over the devices in the field. To use the full potential of IoT the interconnected devices communicate using lightweight protocols that don't require extensive CPU resources. C, Java, Python and some scripting languages are the preferable choices used by IoT applications. To handle any needed protocol conversion, database storage or decision making (e.g. collision handling), IoT nodes use separate IoT gateways in order to supplement the low-intelligence within the IoT node. A large number of manufacturers are involved in the IoT Gateways design and production.

## **The Architecture of IoT Gateways**

The typical architecture of IoT solutions is usually far more complex than the architecture of most enterprise systems. One of the main factors that increases the complexity of IoT systems is that backend services residing in the data center, which is the heart of most enterprise systems, are actually just a piece of the bigger IoT picture. With IoT solutions, we have to deal with a myriad of devices working in the field. Because the nature of these devices is very different from web, desktop, or even mobile clients, we need an intermediate architectural element that will act as a proxy between the world of field devices and the enterprise data center. What we need is an IoT gateway.

## **Why the Need of an IoT Gateway?**

First, sensors usually have very limited capabilities in terms of networking connectivity. Your sensors can likely utilize Bluetooth Low Energy (BLE), just as the majority of beacons available on the market can; there is also the possibility that some of your sensors offer connectivity using the ZigBee protocol. There are also a bunch of other protocols that can be found in the Local Area Network (LAN), Home Area Network (HAN), or Personal Area Network (PAN). All of these protocols have one thing in common; they can't directly connect to larger networks like Wide Area Network (WAN) or the Internet. You need a gateway that can provide your sensors with a single point of contact with external networks.

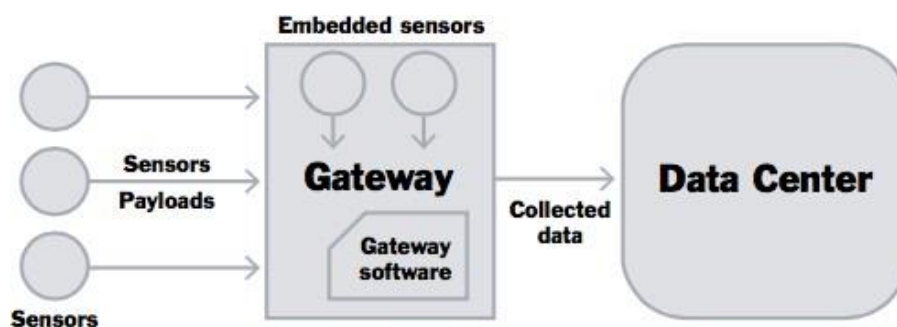
Keep in mind that a gateway is not just a dump proxy that forwards data from sensors to backend services. Sending all the information collected by sensors to a

data center would be highly ineffective in terms of performance and network utilization. An IoT gateway is needed to perform the pre-processing of information in the field, before they're sent to the data center. Such pre-processing includes message filtering and aggregation.

The gateway should also act as a single point of access for monitoring the selected area of the operational field. You don't want to connect to every sensor with your monitoring software; it is easier to monitor only the gateway, which in turn is responsible for gathering all the necessary metrics from the sensors.

## 7.5 Architectural Overview

The following gateway architecture diagram is the most common architectural design where the gateway itself is not equipped with sensors. The gateway software installed on the device is responsible for collecting data from the sensor, pre-processing that data, and sending the results to the data center.



## 7.6 The Gateway Software

The software application is the heart of the gateway. The gateway software is responsible for collecting messages from the sensors and storing them appropriately until they can be pre-processed and sent to the data center. The gateway software decides if the data at a given stage of processing should be temporary, persistent, or kept in-memory.

The gateway software should be designed with failure and disaster recovery in mind. Since gateway devices are often operated in the field, you should prepare for working conditions that are far from ideal. For example, the gateway software should be prepared for a power outage or other actions that may result in an interruption of gateway processing. The gateway software should be bootstrapped and started

automatically as soon as power returns to the device, and it should continue to work from the point where it was interrupted.

Gateway software should also be smart enough to properly handle system logging. It has to find the right balance between the number of log entries stored on the device and those sent to the data center.

## **7.7 Sensor Consumers:**

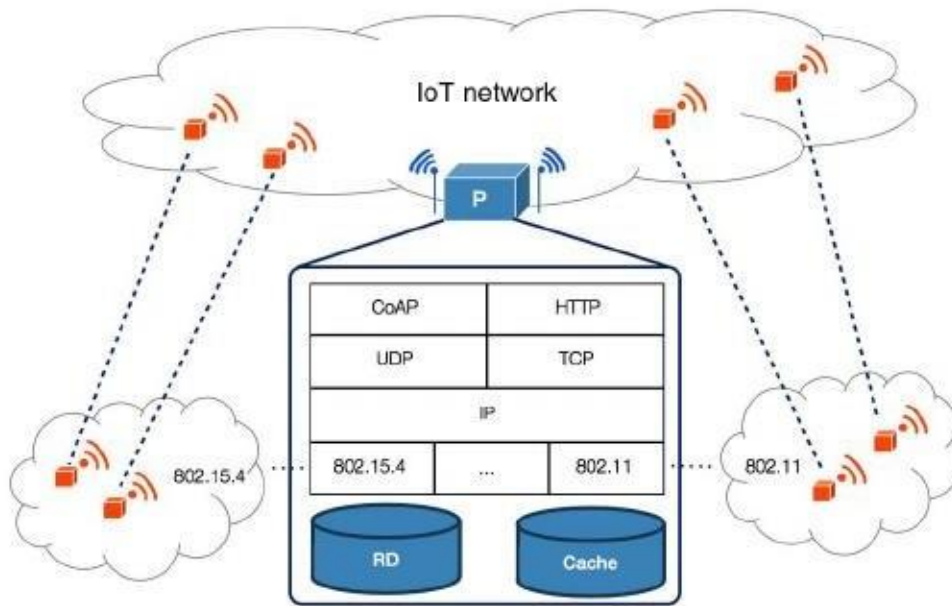
If the software application is the heart of the gateway, then the sensors are the eyes and ears of the gateway.

The messages collected by the gateway from the sensors are usually small in size. For example, the value of the temperature measured by the sensor is just a decimal number. GPS coordinates are two decimal numbers, which represent longitude and latitude. This is an important thing to remember: the gateway operates on a large number of small messages.

While the sensors themselves can generate messages frequently, it is important to anticipate how many messages we really need to gather from the sensors. For example, we can read the temperature from a sensor every millisecond, but do we really need this kind of precision when measuring temperature changes? In the majority of cases, reading the sensor value a few times per second is more than enough. Gateway software usually polls the sensor data periodically. Good gateway software allows you to easily configure the polling interval for every sensor. You definitely don't want to put unnecessary sensor data into the gateway, as obsolete messages consume the precious processing power of your constrained gateway device.

## **7.8 Gateway Data Transfer:**

Usually gateways are connected to the Internet using GPS, WiFi, or Ethernet. Some gateways can also work in both GPS and WiFi modes. Some gateways will be constantly connected to inexpensive local networks, but those using GPS connectivity should be very conservative in terms of what data they send to the data center. The gateway should apply business logic against the data it collects to understand which messages should be sent over expensive GPS networks, and which data can be cached on the device for deferred offline processing.



## 7.9 The IoT Expanding Connectivity:

MuleSoft's mission has always been to connect the world's applications to their devices. We firmly believe that the way people compete today depends upon how efficiently they can do this. But the notion of devices is going through another sea change; a new aspect of the concept of "devices" has become smart devices like watches, connected light bulbs and smart appliances.

### 6.1 The new Edge Layer:

The Edge Layer is responsible for connecting devices locally and manages the data collection and connection to the server. The benefits of this approach are:

- First pass data filtering reduces the amount of data transmitted but retains the meaning of the data
- Device connectivity doesn't fail if the network fails, or there is an intermittent connection. The Edge Layer is responsible for handling outages and store and forward of data.
- Enables site level orchestration across devices from different vendors using different protocols

The Edge layer has three main components in a typical IoT deployment.



The device or sensor itself. In IoT this is the client that generates data and/or receives commands to execute.

Most devices will connect to a gateway that enables access to the internet or private network. Typically, these gateways speak a proprietary protocol between the connected devices and then allow connectivity through the gateway using a standard protocol such as HTTP.

The Edge Controller is responsible for connecting to all the gateways and independent devices in a physical location. The Edge control collected and collates data from all the devices, transmits data and accepts commands from the server to execute across some or all the devices.

## **6.2 Conclusion**

The gateway is a key component of every IoT solution. Before you decide what kind of hardware you would like to purchase as your gateway platform, spend some time analyzing your message flow and the data formats of the payloads, and try to filter out or aggregate as much data as you can before sending it from the gateway to the data center. Also, while the choice of proper hardware for your IoT solution is very important, you have to keep in mind that picking up the right gateway software and management infrastructure is a factor that will highly impact the total maintenance cost of your system

## **8 Chapter 8: Web application**

### **8.1 Introduction to Our web Application:**

In this chapter, we present that web application that we developed to help the farmer access the field's data and check the predicted advice according to field conditions. The application is for the remote access to what's going on in the field periodically and alert if there is any change in climate or soil that need to take action. The reading of the sensors is stored in an online database and connected to GUI that allow users interact with it. The web site has a username and password to provide security to the data and customize the different privilege to different users and monitors. So that each field has a different data; climate data, soil data and field status if the plants need water, fertilizers or pesticides and all this data can't be accessed by all workers but the responsible for it only.

### **8.2 Prerequisites for web Development:**

To develop web site, we used PHP, HTML, CSS, MySQL and python.

Python used to convert from an environment condition to understandable readings and open http session with the website, MySQL used to store the readings in database, PHP programming language is used to connect between gateway and web server to get the sensors data and communicating with MYSQL database on website to insert the sensor readings, HTML and CSS used to build the style of the web pages and display the readings from the database in tables to the monitors.

### **8.3 Implementation:**

#### **8.3.1 Getting website alive on web**

The first step to implement our website is to register a domain name to have our own website URL so we registered for a free domain name for months to test our website. To get our website alive on web, Web hosting is needed. Web hosting means providing spaces on server to serve website. Some companies such as [www.000webhost.com](http://www.000webhost.com) provide free

webhosting and also free sub-domain. A sub-domain is just a domain that is part of another domain. So, the URL of website will be <https://ourwebsite.000webhosting.com>.

The next step is moving website files onto server, FTP is used to transfer files from computer to host's server. There are several FTP clients that is installed on computer such as FileZilla server which is used in building this website. FileZilla is a fast and reliable FTP client which supports resume on both downloads and uploads. Using FileZilla enable merging any updated website files on PC automatically on host's server.

As FileZilla uses FTP to transfer files and folders. FTP Client open the internet connections, one for command and one for data. For command port, FTP has a standard port number on which the FTP server "listens" for connections. A port is a logical connection point for communicating using the Internet IP. The standard port number used by FTP servers is 21 and is used for sending commands. For data port, the port number that is used for transferring data will vary depending on the mode of the connection whether it's active or passive. In this website, passive FTP connection is used; where the FTP server opens a port and listens passively and the client connects to it. For Passive FTP connection to succeed, the FTP server administrator must set its firewall to accept all connections to any ports that the FTP server may open.

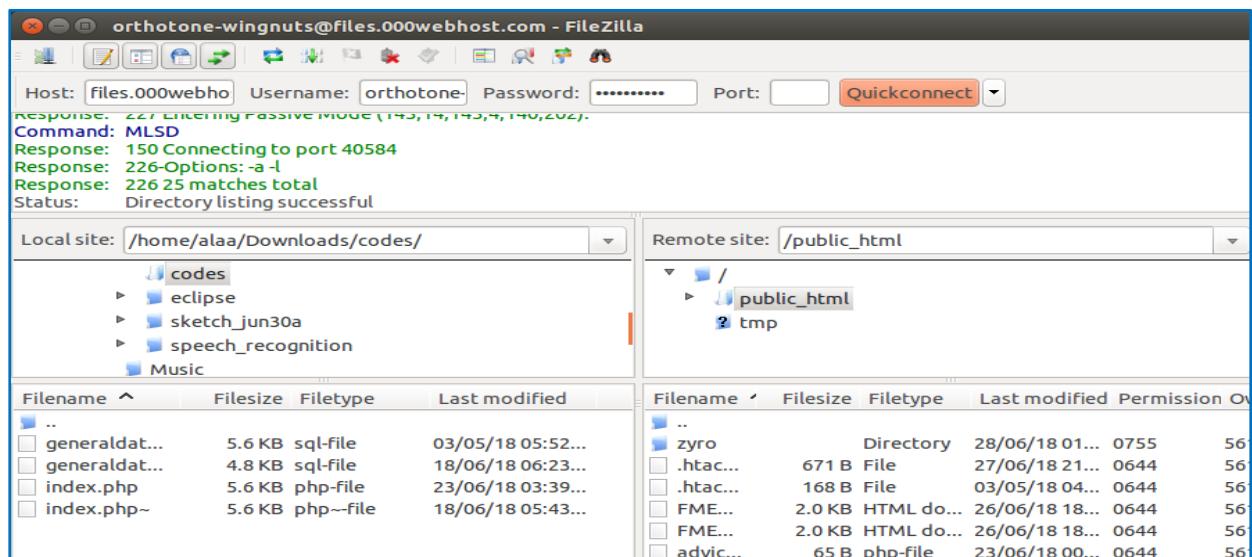


Figure 8-32:FileZilla Client Communicating with server

In our website the FTP server uses passive mode open port 21 for connection with FTP client, giving FTP client the hostname, the password and the port number; it starts communicating with the server.

### 8.3.2 Connection Webpage

The connection web page is coded by PHP and used to connect between the gateway or the local server which is the raspberry pi and the website server. The data is received from local server using POST method written in python code then stored in a database, in our application, the received data from the POST method is extracted and stored in a database which is provided from the webhost. This database can be managed using phpMyAdmin which is a free software tool written in PHP, intended to handle the administration of MySQL over the Web and supports a wide range of operations on MySQL.

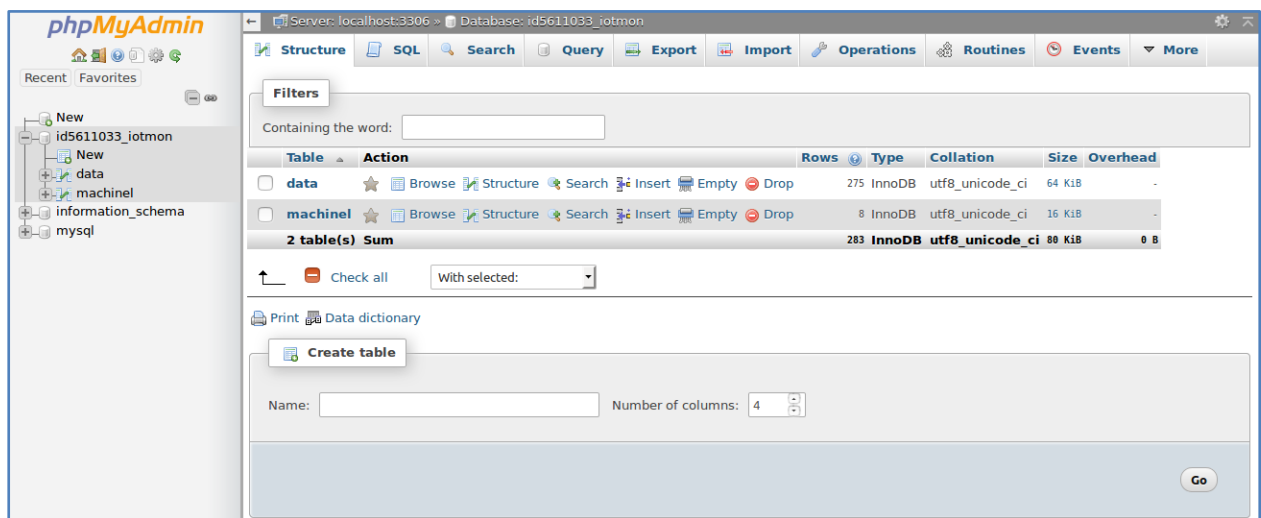


Figure 8-23: phpMyAdmin to handle tables in database on website

The `$_POST` variable is an array of variable names and values provided by PHP to access all sent information using the HTTP POST method. The `$_POST` variable is used to collect values sent using `method="post"`. Information sent using the POST method is invisible to others and has no limits on the amount of information to send; So that we used “`$_POST`” is to extract the data from POST request with a specific key for every sent data. And the key is agreed between the sender and the receiver to avoid missing the data.

Calling the MySQL function **mysqli\_connect()** opens a new connection to the MySQL server and connect to the database to do any action on the database such as insertion, delete or anything else. It takes the database name, username, password and the server name. To store the data; **INSERT INTO** command and **mysqli\_query ()** function is use.

**INSERT INTO** command to insert the extracted data in the right table.

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

The **mysqli\_query ()** function is to execute the query on a MySQL database and return true if the insertion is done or false if any failure happened. it takes the query from MySQL **INSERT** command the connection from **mysqli\_connect ()** function

### 8.3.3 Login Webpage

The login web page consists of 2 boxes of username and password

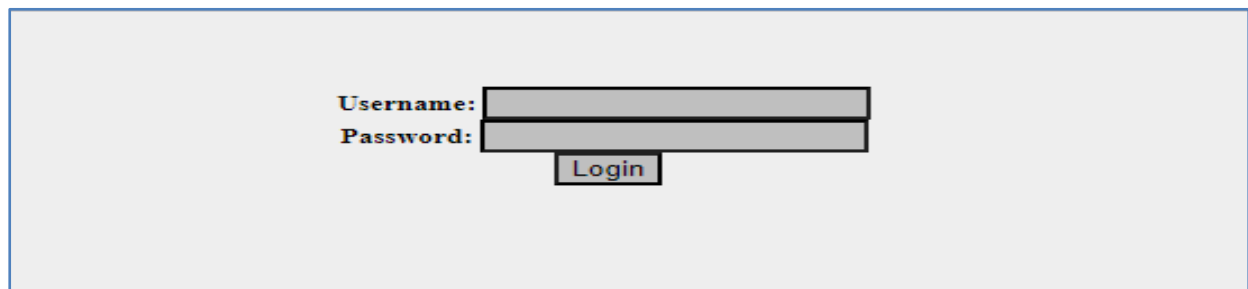
A screenshot of a login form. It features two input fields: the first is labeled 'Username:' and the second is labeled 'Password:'. Below these fields is a button labeled 'Login'. The entire form is enclosed in a light gray rectangular box with a thin blue border.

Figure 34:Login page example

This web page is coded in HTML that compare the entered username and the password with the stored username and password and if they are the same, login to the required web page. Every username and password used to login to specific web page that contain the user's data to make the data more secure.

If the username and the password is correct, the **href** attribute sets or returns the entire URL of the current page to redirect to the required webpage. If the username and password is incorrect it shows and error message to reenter the correct login information.

**alert()** function is used to show up the error message, the **alert()** method displays an alert box with a specified message and an OK button. An alert box is often used if you want to make

sure information comes through the right user, the alert box takes the focus away from the current window and forces the browser to read the message. This method prevents the user from accessing other parts of the page until the box is closed.

### 8.3.4 Data and Advice Webpage

Data and advice webpages are coded in PHP, HTML and CSS to insert data from website database to table format for simply user interfacing. PHP connects to table in side database provided by the webhost, passing variables row by row to the HTML table.

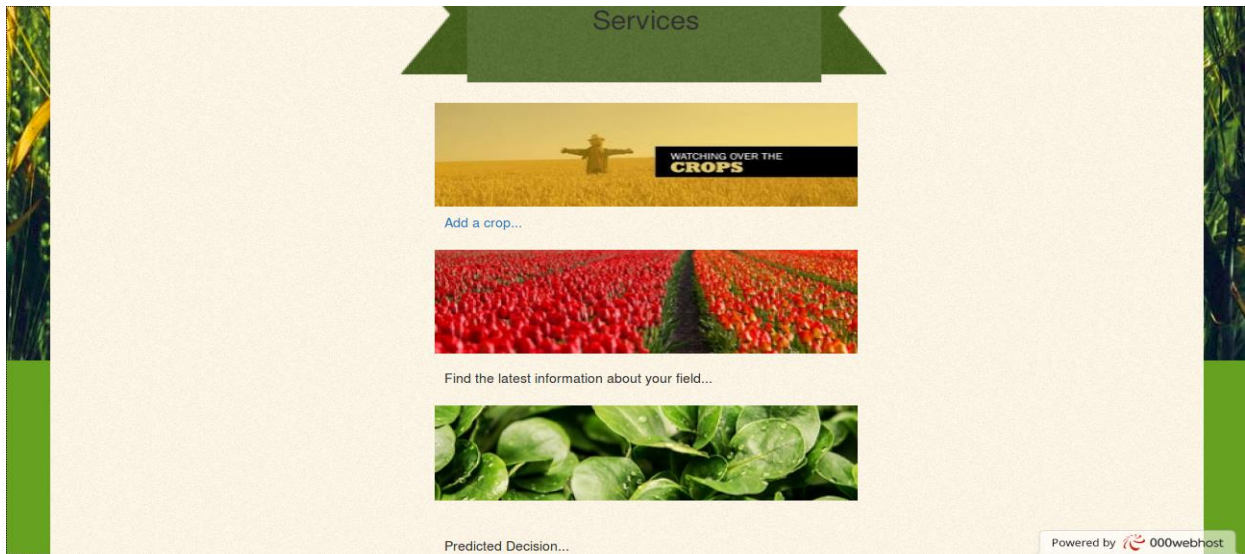
## 8.4 Graphical user interface:

This website is designed for greenhouse owners and on a large scale it can be applicable on field. Any user has installed the system and sensors has a permission to login using his account to access the information related to his field or greenhouse.

The first webpage that appears after opening website is the home page



Figure 8-25:home page

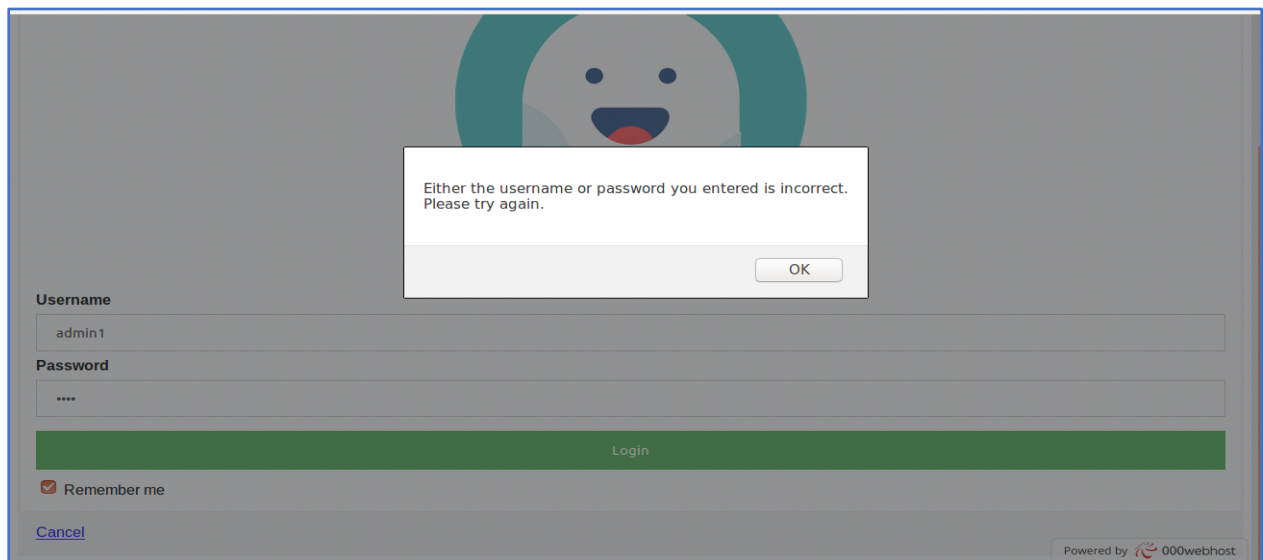


**Figure 8-26:home page services**

The home page contains the services provided by the website. The user has to login to find the latest data information about his field or show any predictive information due to conditions. The login button on the top of the home page redirect the user to the login webpage in the next picture.

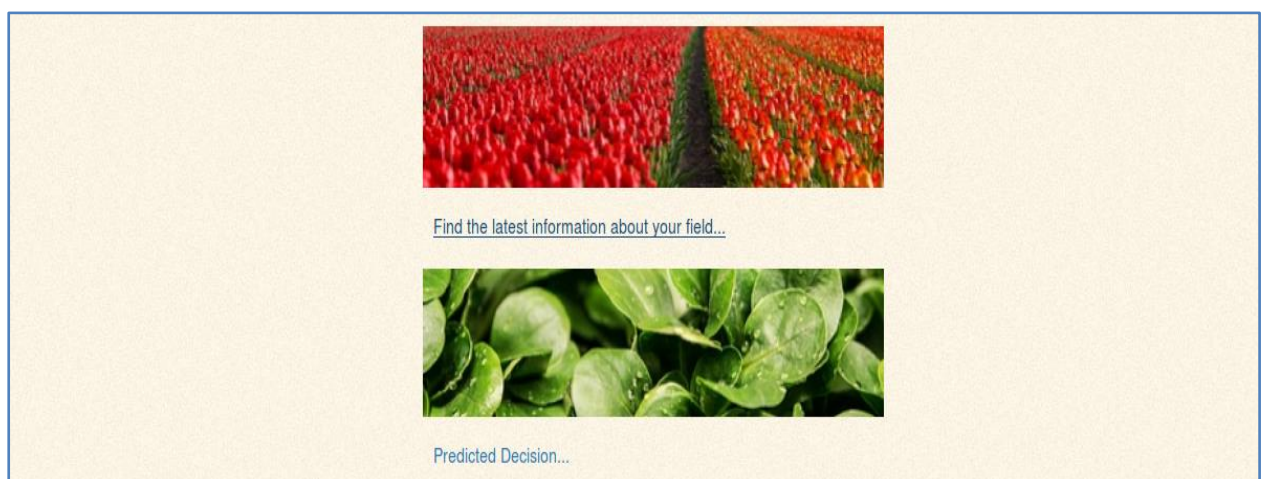
**Figure 8-27: Login Form**

The login webpage contains login form for user to access data or advice information using the user own username and password. The website will get the account information from the table in the database stored on the website server and compares it with the values entered by the user. If the values are the same, the website will redirect the user to home page and permit the user to access all sensors activity and know the field status. If not, the website will show the user an error message to tell the user that the username or password is incorrect.



**Figure 8-28: Error Message**

After login the user has the permission to access the data of his filed and show tables by clicking on the hyperlinks associated to data or advice as shown in the following picture



**Figure 8-29: hyperlinks redirects to data and advice**



Clicking on this hyperlink will redirect the user to the table of the sensors data or the advice if the user click on the predicted decision hyperlink. These tables are filled from the sensors data sent from the gateway to the website database. The tables contain the readings of all sensors installed in the system and the date and time of the sent readings which in turn allow the user to monitor his field or greenhouse. The system usually sent the field status to the website and If any irregular changes happens the gateway sends an advice message to the user contain the appropriate action according to the field status.

DATEANDTIME	TEMP	HUMIDITY	MOISTURE
2018-06-23 03:49:43pm	27.00	14.00	water undetected
2018-06-23 03:50:46pm	27.00	14.00	water undetected
2018-06-23 03:51:51pm	27.00	14.00	water undetected
2018-06-23 03:52:56pm	27.00	14.00	water undetected
2018-06-23 03:54:00pm	27.00	14.00	water undetected
2018-06-23 03:55:07pm	26.00	14.00	water undetected
2018-06-23 03:56:14pm	26.00	15.00	water undetected

**Figure 8-30: sensors data table**

DATEANDTIME	PLANT STATE
2018-07-10 08:19:43pm	This is an advisory message: The Current Accumulative GDD = 137 approached the thershold value for a regular unepidemic session therefor you have an approximatly from 10 to 14 days to apply the protective fungisides to prevent the spreading of the disease
2018-07-10 08:19:44pm	This is an advisory message: note the current Accumulative GDD = 254 so starting from approximately 5 days of today the disease spreading will become linear with the accumaltive GDD which will cause sevier losses in the yield if no action is taken
2018-07-10 08:19:45pm	Warning !! The Accumulative GDD is = 397 so if no action was taken during the previous days THERE WILL BE an EPIDEMIC SPREAD

**Figure 8-31: Advice table**

## **9 Chapter 9: Conclusion and future work**

Using the IOT in different application enriches the usage of the hardware that existing many years ago to improve the quality of life and save wasted efforts.

As known the agriculture field is decreasing day by day due to the exhausting work and monitoring and it leads to starvation.

So that we can replace large of workers with accurate sensors provide us with climate conditions and the soil conditions and predict the diseases and infection before happening.

It is decreasing the amount of the corrupted crops.

Our application makes predictions based on many observations over years to connect the result with the different factors of weather and soil and develop its self to get accurate result.

But it's limited to warn the user only without taking any actions and the future work of the applications is taking actions to avoid the human mistakes.

From the future work:

- Irrigation systems
- Pests detection and combating with microwave technology
- Adopting temperature in green houses

### **Irrigation systems**

One of the most dangerous factors in agriculture is the amount of the water in the soil cause the increasing rots the roots of the plants and the decreasing dries the plants.

So that after calculating the accurate amount of water and sensing the amount of water every reasonable time, it can irrigate the crops every time needed water only and if any increasing water accidently there another action to drainage quickly before ruining the crops almost without human interference.

### **Pests detection and combating with microwave technology**

One of the greatest results of the machine learning is the things detections and it appears clearly in agriculture in pest detection and detect the type of it from the huge number of types of pests with high percentage of success.

And by redirect the microwave after detecting the place of the pest, it can be got rid of it without using any chemicals or poisons material and without harming the crops and improve the human health by decreasing the chemicals.

### **Adopting temperature in green houses**

Many countries suffer from the lack of the good agricultural Land or the hard weather conditions to grow up different plants.

So that it's created the green houses to plant almost all crops by controlling the temperature and the other factors to grow up good crop to the destitute countries.

For example, tomatoes need 3 to 4 months of warm, clear, fairly dry weather to produce best. Tomatoes need consistent night temperatures between 55°F and 75°F to set fruit and by sensing the temperature every reasonable time, any change in the temperature can be controlled by controlling the thermostat of the green house.

## 10 Chapter 10: References

- 1) Geipel, Jakob. *Implementation and Improvement of an Unmanned Aircraft System for Precision Farming Purposes*. 2016.
- 2) Hernandez, André, et al. "Modular Sensor Architecture for Automated Agricultural Data Collection on the Field †." *Proceedings*, vol. 1, no. 2, 2016, p. 9., doi:10.3390/ecsa-3-e001.
- 3) "Raspberry Pi Based Global Industrial Process Monitoring Through Wireless Communication." *International Journal of Recent Trends in Engineering and Research*, vol. 3, no. 3, Aug. 2017, pp. 129–133., doi:10.23883/ijrter.conf.20170331.026.krjtz.
- 4) Bagal, Nalini, and Prof. Shivani Pandita. "Real-Time Transmission of Voice Over 802.11 Wireless Networks Using Raspberry Pi." *International Journal of Emerging Trends in Science and Technology*, 2015, doi:10.18535/ijetst/v2i8.01.
- 5) Ferdoush, Sheikh, and Xinrong Li. "Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications." *Procedia Computer Science*, vol. 34, 2014, pp. 103–110., doi:10.1016/j.procs.2014.07.059.
- 6) Anire, Roselle B., et al. "Environmental Wireless Sensor Network Using Raspberry Pi 3 for Greenhouse Monitoring System." *2017 IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, 2017, doi:10.1109/hnicem.2017.8269426.
- 7) Xin Dong, Mehmet C. Vuran, and Suat Irmak, "Autonomous precision agriculture through integration of wireless underground sensor networks with center pivot irrigation systems." *Ad Hoc Networks*, Volume 11, Issue 7, September 2013, Pages 1975–1987, DOI: 10.1016/j.adhoc.2012.06.012 Online. Available:
- 8) Flores, Kristoffer O., et al. "Precision Agriculture Monitoring System Using Wireless Sensor Network and Raspberry Pi Local Server." *2016 IEEE Region 10 Conference (TENCON)*, 2016, doi:10.1109/tencon.2016.7848600.
- 9) Grgic, Kresimir, et al. "A Web-Based IoT Solution for Monitoring Data Using MQTT Protocol." *2016 International Conference on Smart Systems and Technologies (SST)*, 2016, doi:10.1109/sst.2016.7765668.
- 10) Rau, Amogh Jayaraj, et al. "IoT Based Smart Irrigation System and Nutrient Detection with Disease Analysis." *2017 IEEE Region 10 Symposium (TENSYP)*, 2017, doi:10.1109/tenconspring.2017.8070100.
- 11) Grgic, Kresimir & Speh, Ivan & Hedi, Ivan. (2016). A web-based IoT solution for monitoring data using MQTT protocol. 249-253. 10.1109/SST.2016.7765668.
- 12) A Atmoko, R & Riantini, R & K Hasin, M. (2017). IoT real time data acquisition using MQTT protocol. *Journal of Physics: Conference Series*. 853. 012003.

10.1088/1742-6596/853/1/012003.\

- 13) Lampkin V et al 2012 *Building smarter planet solutions with MQTT and IBM WebSphere MQ telemetry IBM, ITSO*
- 14) Colitti W, Steenhaut K and De Caro N 2011 *Proc. IP+SN (Chicago, USA) Integrating Wireless Sensor Networks with the Web*
- 15) "Contiki: The Open Source Operating System for the Internet of Things." Online. . Available: <http://www.contiki-os.org/index.html>. [Accessed: 16-Sep-2013].
- 16) Abdul-Latef, Lamees Munef, and Iyad Abdul-Muhsan Ahmed. "Evaluation of Some Pesticide Residues in Fruits Import by High Performance Liquid Chromatography." *Al-Mustansiriyah Journal of Science*, vol. 27, no. 4, 2017, doi:10.23851/mjs.v27i4.34.
- 17) Abouziena, H.f., and W.m. Haggag. "Weed Control in Clean Agriculture: A Review1." *Planta Daninha*, vol. 34, no. 2, 2016, pp. 377–392., doi:10.1590/s0100-83582016340200019.
- 18) Eltholth, Mahmoud, et al. "Characterisation of Production, Marketing and Consumption Patterns of Farmed Tilapia in the Nile Delta of Egypt." *Food Policy*, vol. 51, 2015, pp. 131–143., doi:10.1016/j.foodpol.2015.01.002.
- 19) El-Kader, Sherine M. Abd, and Basma M. Mohammad El-Basioni. "Precision Farming Solution in Egypt Using the Wireless Sensor Network Technology." *Egyptian Informatics Journal*, vol. 14, no. 3, 2013, pp. 221–233., doi:10.1016/j.eij.2013.06.004.
- 20) "DHT11 Humidity & Temperature Sensor Module." *UUGear*, [Online], [www.uugear.com/portfolio/dht11-humidity-temperature-sensor-module/](http://www.uugear.com/portfolio/dht11-humidity-temperature-sensor-module/).
- 21) "DHT11–Temperature and Humidity Sensor." *Components101*, [Online], [components101.com/dht11-temperature-sensor](http://components101.com/dht11-temperature-sensor).
- 22) "Guide for Soil Moisture Sensor YL-69 or HL-69 with the Arduino." *Random Nerd Tutorials*, [Online], [randomnerdtutorials.com/guide-for-soil-moisture-sensor-yl-69-or-hl-69-with-the-arduino/](http://randomnerdtutorials.com/guide-for-soil-moisture-sensor-yl-69-or-hl-69-with-the-arduino/).
- 23) "Arduino and Soil Moisture Sensor -Interfacing Tutorial." *Electronic Circuits and Diagrams-Electronic Projects and Design*, 10 Mar. 2017, [www.circuitstoday.com/arduino-soil-moisture-sensor](http://www.circuitstoday.com/arduino-soil-moisture-sensor).
- 24) "Getting to Know NodeMCU and Its DEVKIT Board." *IBM Cognitive Advantage Reports*, IBM Corporation, 7 Aug. 2017, [www.ibm.com/developerworks/library/iot-nodemcu-open-why-use/index.html](http://www.ibm.com/developerworks/library/iot-nodemcu-open-why-use/index.html). [online]

- 25) "Robot-R-Us Singapore." *6 Degrees of Freedom Robotic Arm (Assembled) | Robot Kit | Robot R Us*, [Online], [www.robot-r-us.com/vmchk/wireless\\_wifi/wireless-module-ch340-nodemcu-v3-lua-wifi-esp8266-esp-12e.html](http://www.robot-r-us.com/vmchk/wireless_wifi/wireless-module-ch340-nodemcu-v3-lua-wifi-esp8266-esp-12e.html).
- 26) "ESP8266 And the DHT22 Sensor." *The Free Physicist*, [Online], 29 Aug. 2017, [thefreephysicist.com/esp8266-dht22-sensor/](http://thefreephysicist.com/esp8266-dht22-sensor/).
- 27) Sharma, Ruchir. "Raspberry Pi Talking to ESP8266 Using MQTT." *Hackster.io*, [Online], 25 Apr. 2018, [www.hackster.io/ruchir1674/raspberry-pi-talking-to-esp8266-using-mqtt-ed9037](http://www.hackster.io/ruchir1674/raspberry-pi-talking-to-esp8266-using-mqtt-ed9037).
- 28) Instructables. "Programming ESP8266 ESP-12E NodeMCU Using Arduino IDE - a Tutorial." *Instructables.com*, Instructables, 30 Sept. 2017, [www.instructables.com/id/Programming-ESP8266-ESP-12E-NodeMCU-Using-Arduino-/](http://www.instructables.com/id/Programming-ESP8266-ESP-12E-NodeMCU-Using-Arduino-/).
- 29) Pujar, Ravi. "SIM900 GPRS HTTP AT Commands." *Embedded World*, [Online], 28 Jan. 2018, [www.raviyp.com/embedded/194-sim900-gprs-http-at-commands?showall=&limitstart=](http://www.raviyp.com/embedded/194-sim900-gprs-http-at-commands?showall=&limitstart=)
- 30) <https://www.hivemq.com/blog/mqtt-essentials/>