

# CS244P- INTERNET OF THINGS

## Project Final Report: Smart Gardening System using IOT

Name	Student ID
Sameer Ganesh Shinde	19644693
Kevin Shailesh Jain	73082873

### DOCUMENT VERSION INFORMATION

Version	Date	Topic	Description	Page No.
1.1	9 <sup>th</sup> Nov 2018	Project proposal	Project proposal and initial scope	1
1.2	30 <sup>th</sup> Nov 2018	Progress Report	Progress and challenges of project	3
1.3	14 <sup>th</sup> Dec 2018	Project Final Report	Final Report and result	4

### OVERVIEW

#### 1. Abstract

**i** In this project we aim to build a smart gardening system that is capable of fine tuning the various parameters necessary for the healthy growth of the plant and monitor it continuously so as to provide the perfect environment that it required for the best results.

#### 2. Motivation

**i** Many of the plants that grow in their natural environment are subject to unfavourable conditions that hinder their growth to full potential and hence the yield is of low quality. So the system that we are planning to develop will take special care and cater the needs of the individual species of plants and provide them with the ideal environment that they require. Different species require different conditions, nutrients, sunlight, moisture and various other things which are quite difficult to manage manually. So a system that can take care of this will ease the burden on the person as well as make the process much more efficient.

#### 3. Proposed Solution

**i** The system that we plan to develop will make use of sensors and send the data to the cloud in real time at regular intervals. The sensors will keep track of the various parameters involved in maintaining the ideal environment required by the specific species of the plant. There are ideal conditions of the parameters that are specified for the plant and the sensors maintain those conditions continuously with the changing environment conditions.

Must have features that we are considering to build are:

- We aim to keep a track of the soil moisture that is required by the plant.
- We aim to adjust the sunlight that is required by the plant.
- We aim to monitor the temperature of the environment and change it accordingly.

Good to have features that we are considering to build are:

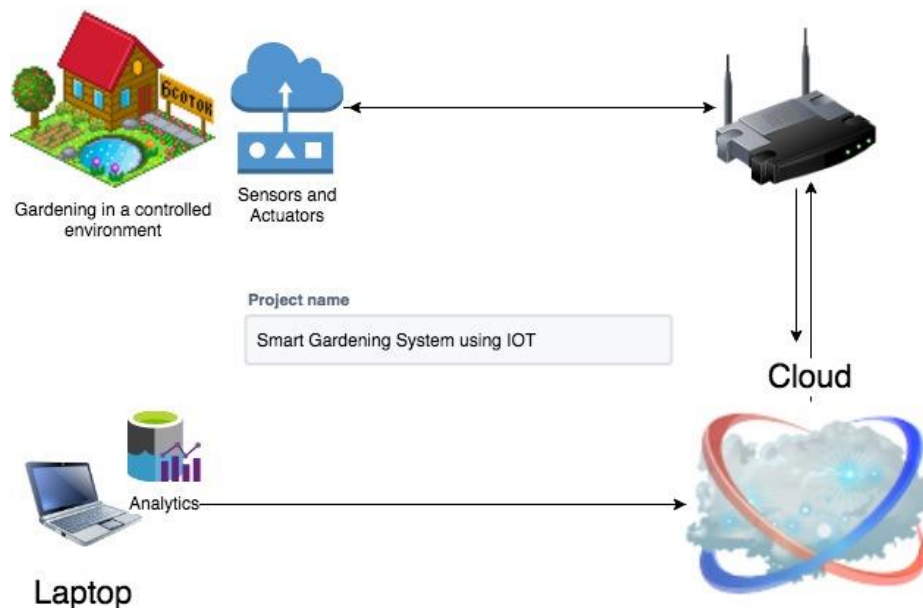
- We aim to automatically start sprinkler when the water level is low
- We aim to apply machine learning to the data sent by sensors to identify complex patterns like water level, soil moisture etc

## FUNCTIONAL ARCHITECTURE AND SENSORS

### 1. Wireless Protocol

**i** In this project, we plan to use Wi-Fi to be our wireless protocol. The detailed architecture is shown in the figure.

### 2. Architecture



### 3. Required Components

**i** List of Sensors required:

Soil Moisture Sensor

Temperature Sensor

Servo Motor

WiFi Module

## PROGRESS REPORT – 30<sup>th</sup> November 2018

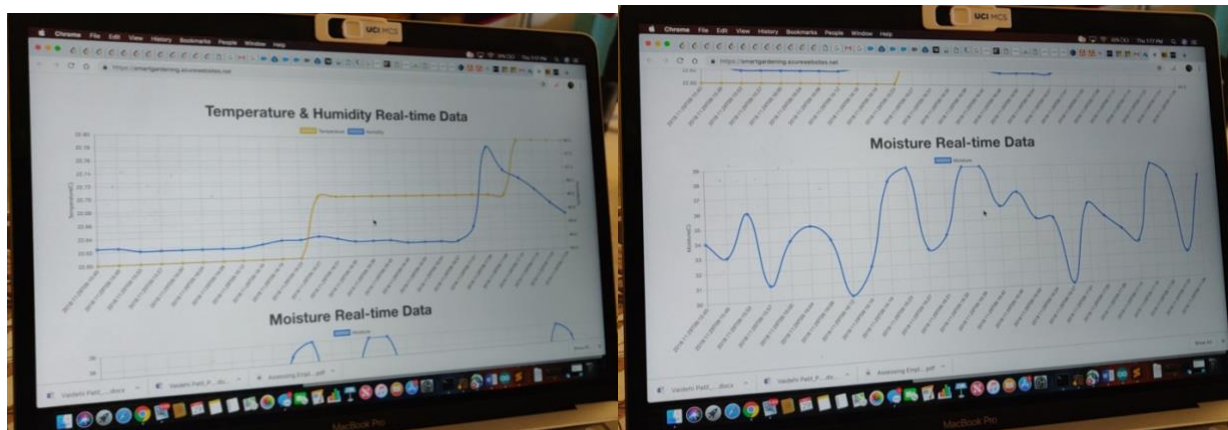
### 1. Initial Scope

**i** Develop an IoT based smart gardening system in which we monitor factors like temperature, humidity, soil moisture and the sunlight using the various sensors provided and then compare it to the ideal values for a given plant and adjust them accordingly or issue an alarm/indication to the owner to adjust the conditions manually such that they are favorable for the plant and it can grow to its full potential. We plan to upload the data on azure and monitor it consistently

### 2. Current Progress

**i** After the sensors were provided, we tested the sensors and tried to read the values. We successfully read the values using the Sparkfun ESP8266 board and then uploaded them on the azure cloud using that module on Wi-Fi. We made graphs on azure for all the three sensors to monitor the changes and read the values in real-time. Any changes in values can be visualized with accuracy on the graph

### Work Samples:



### 3. Future Work

**i** There are a few more tasks that need to be implemented in the system. All the individual sensors were tried and tested while reading and uploading the data, but all these sensors need to be assembled on the board together and they should be combined and send the data to the cloud in real-time. If the values being monitored on the graph fall below a certain threshold then the alarm/indication has to be set-off so that the ideal conditions can be restored.

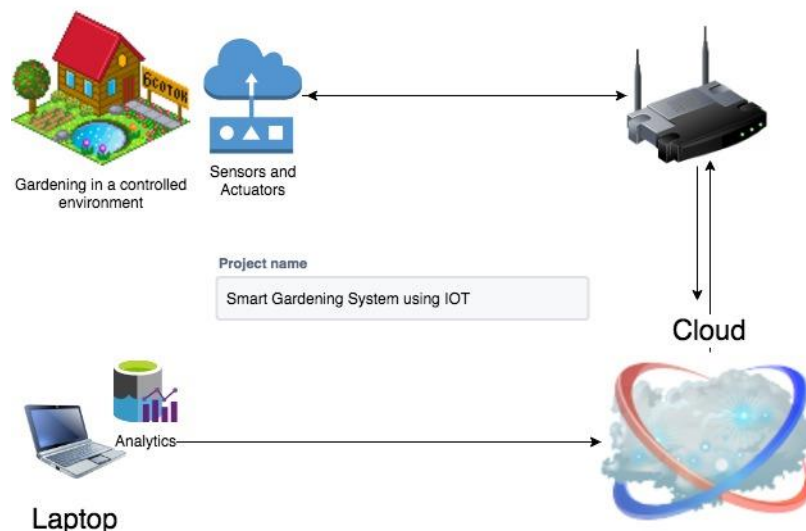
### 4. Challenges Faced

- i**
- We have to read data from 3 analog sensors but Sparkfun ESP8266 has only one analog input pin, which is forcing us to use Multiplexer for reading values from multiple sensors. If we get this Multiplexer, we will be able to use all sensors, otherwise we will have to use one analog sensor only and if possible, any other digital sensor(s).
  - Generating multiple graphs from the same message packets containing all the sensor's value. We have accomplished this part.

### USECASE

**i** We plan to design a smart gardening system to record the temperature, humidity and soil moisture in real time and monitor it so that we do not have to take care of it in person. This is quite a tedious task and requires a lot of manpower. So, we plan to implement this system to ease the life of people who are quite busy and cannot afford to devote time to this task. It works quite hassle free and the user can be updated as and when necessary to bring their attention to the task whenever required rather than having a constant check on the it.

The data is captured using different sensors and Wi-Fi modules board ESP 8266 Sparkfun Thing Dev is used to send that data to a cloud server on Azure and the data is displayed in real time if the user wants to check it anytime from anywhere to see what the temperature, humidity and soil moisture values are. Now the user can set the threshold values so that if any of the attribute falls beyond this value then the user can be updated about it or the necessary action as given by the user can be prompted. In our case if the temperature falls below a certain value, we use a motor to turn on the shades so that the plant is now in shade.


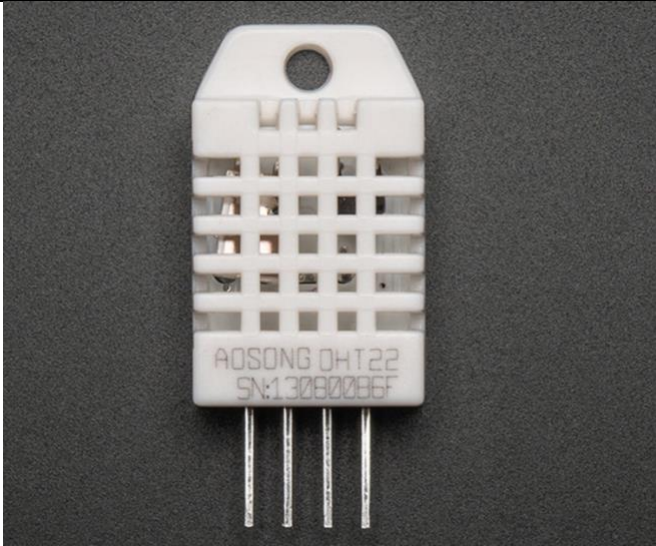
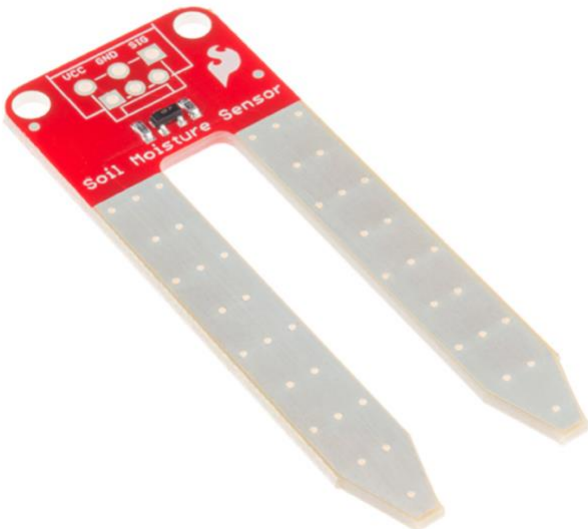


### SMART APPLICATION

**i** -The data that is sent from the sensors is captured on the cloud and that is used for making predictions on the probability of rain. Here we can set the conditions for the upcoming days also and hence it can be safeguarded against unforeseen conditions also.

-We use servo motor to turn the shades as and when the temperature increases or decreases to maintain the suitable temperature without any help from the user. .

## COMPONENTS USED

Component	Cost
	Sparkfun ESP8266 Thing Dev Board: \$16.95
	DHT22 temperature humidity sensor - \$9.95
	Soil Moisture Sensor - \$5.95

 <p>Analog Servo Motor with a white plastic horn, a red, yellow, and blue braided cable, a white servo housing, a white servo horn, a white servo gear, and a white servo horn.</p>	<p>Grove Servo - \$5.90</p>
 <p>Arduino Uno R3 board, Arduino Uno R3 box, Arduino Uno R3 USB cable, Arduino Uno R3 power cable, and Arduino Uno R3 documentation.</p>	<p>Arduino Uno - \$19.50</p>



## BASIC PROJECT WORKFLOW

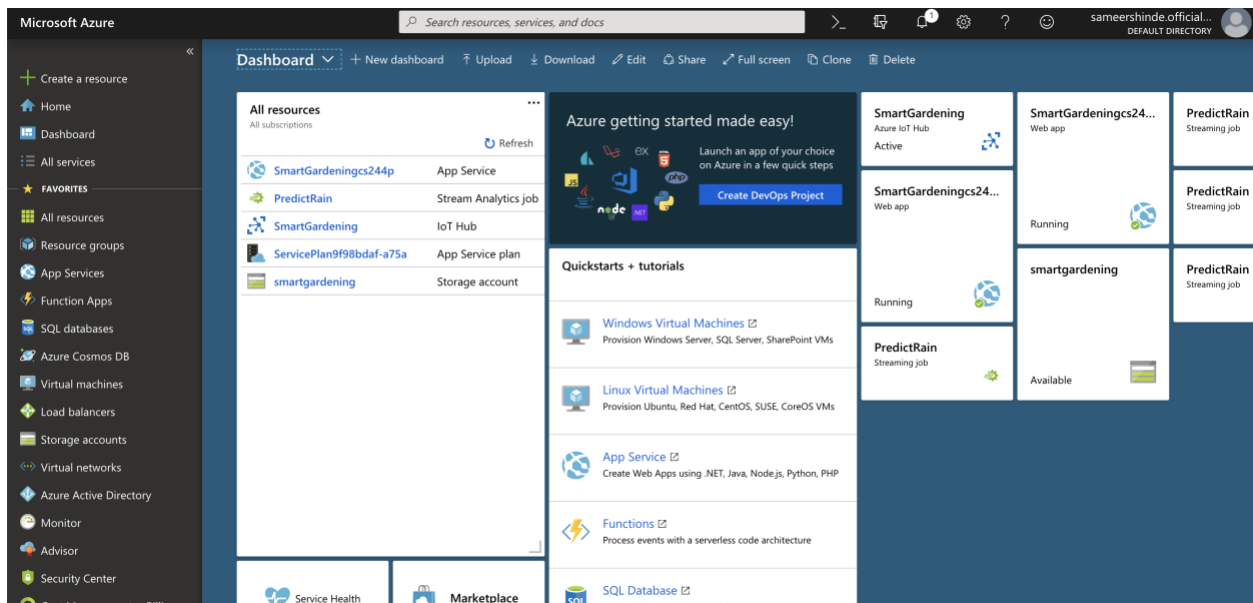
### 1. User Side

- i** The setup at the user side includes all the sensors: temperature and humidity sensor and soil moisture sensor. These sensors continuously sense the environment for temperature and humidity fluctuations and keep track of the soil moisture content as well. This data is sent to the cloud at small periodic intervals so that it can be monitored by the user at any time and from anywhere at their convenience and the plant/garden can be maintained at ideal conditions at all times without having to be physically present there.



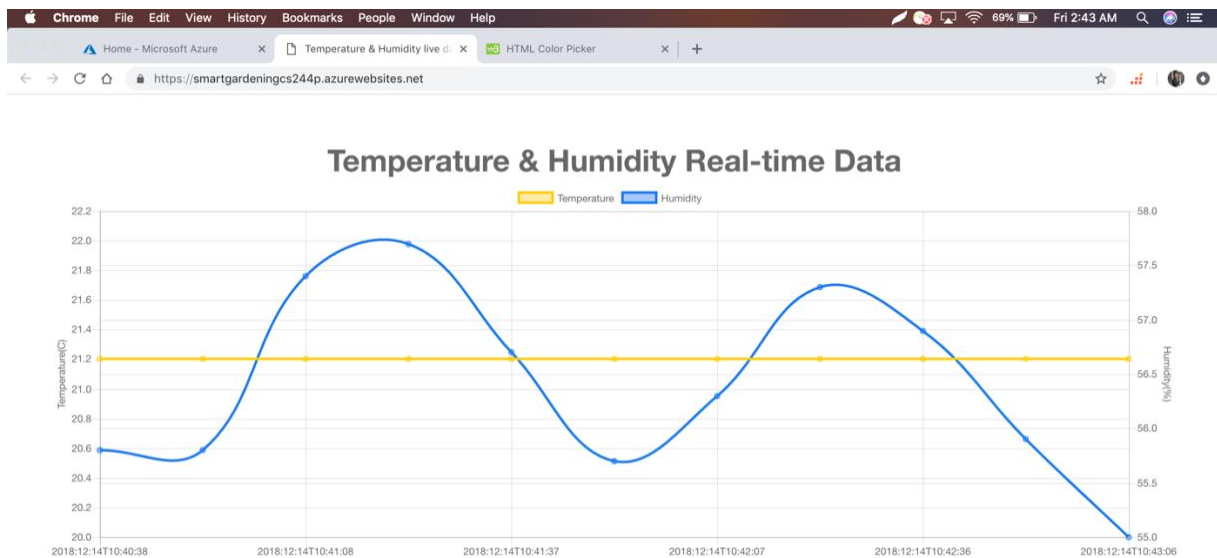
### 2. Cloud Server

- i** We are sending data from the user's garden to the cloud server where we have hosted our web application. Our device is registered at the server and using our IOT Hub's connection key and device registration connection key, we are communicating between the server and our system. On the cloud we collect the data and display it in the form of a graph which changes continuously as the data is sent by the sensors to the cloud at periodic intervals. Also, we store that data in our azure storage in BLOB containers and then use that data to predict the chances of rain. For this purpose, we use the machine learning weather forecast model already embedded as a service in Azure.



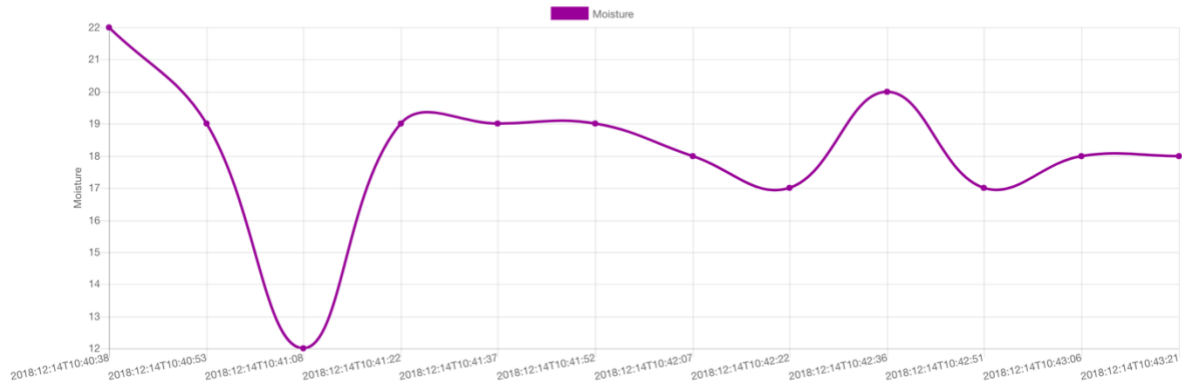
## SCREENSHOTS:

### 1. Dashboard for statistics





## Moisture Real-time Data



## 2. Rainfall prediction:

-1365699203\_9670da148ade4a648e26e0df8754bf40\_1

time	temperature	humidity	probabilities of rain
2018-12-08T09:09:33.4060000Z	22.2	66.7	0.379807651042938
2018-12-08T09:09:47.2060000Z	22.3	66	0.643517553806305
2018-12-08T09:10:01.0040000Z	22.3	64.3	0.487301170825958
2018-12-08T09:10:16.1180000Z	22.3	64.2	0.487301170825958
2018-12-08T09:10:28.6720000Z	22.3	64	0.702977478504181
2018-12-08T09:10:42.4330000Z	22.4	64.1	0.467182278633118
2018-12-08T09:10:56.2370000Z	22.4	64.3	0.467182278633118
2018-12-08T09:17:32.3740000Z	22.7	61.8	0.500276029109955
2018-12-08T09:17:48.2000000Z	22.7	62.4	0.500276029109955
2018-12-08T09:18:03.9860000Z	22.7	62.4	0.500276029109955
2018-12-08T09:18:19.8000000Z	22.7	62.5	0.500276029109955
2018-12-08T09:18:35.5990000Z	22.7	62.5	0.500276029109955
2018-12-08T09:18:51.4180000Z	22.7	62.3	0.500276029109955
2018-12-08T09:19:07.2780000Z	22.7	62.5	0.500276029109955
2018-12-08T09:19:23.0330000Z	22.7	62.3	0.500276029109955
2018-12-08T09:19:38.8340000Z	22.7	62.3	0.500276029109955
2018-12-08T09:19:54.6390000Z	22.8	62.4	0.422838568687439
2018-12-08T09:20:10.4540000Z	22.8	62.4	0.422838568687439
2018-12-08T09:20:26.2580000Z	22.8	62.2	0.422838568687439
2018-12-08T09:20:42.0540000Z	22.8	62.2	0.422838568687439
2018-12-08T09:20:57.8670000Z	0	0	0.489944100379944
2018-12-08T09:21:13.6710000Z	22.8	62.9	0.422838568687439
2018-12-08T09:21:29.4770000Z	22.8	62.3	0.422838568687439
2018-12-08T09:21:45.2930000Z	22.8	62.3	0.422838568687439
2018-12-08T09:22:01.0940000Z	22.8	62.2	0.422838568687439
2018-12-08T09:22:16.8990000Z	22.8	62.4	0.422838568687439
2018-12-08T09:22:32.7000000Z	22.8	62.3	0.422838568687439
2018-12-08T09:22:48.5210000Z	22.8	62.1	0.422838568687439

### 3. Data Transmission

```
02:43:05.568 -> Sending message: {"deviceId":"Feather HUZZAH ESP8266 WiFi","messageId":29,"temperature":21.2,"humidity":21.2}
02:43:05.568 -> IoTHttpClient accepted the message for delivery.
Message sent to Azure IoT Hub
Sucessfully in sendMessage.
02:43:18.255 -> Sending message: {"deviceId":"Feather HUZZAH ESP8266 WiFi","messageId":30,"temperature":21.3,"humidity":21.3}
02:43:18.255 -> IoTHttpClient accepted the message for delivery.
Message sent to Azure IoT Hub
Sucessfully in sendMessage.
02:43:33.030 -> Sending message: {"deviceId":"Feather HUZZAH ESP8266 WiFi","messageId":31,"temperature":21.3,"humidity":21.3}
02:43:33.067 -> IoTHttpClient accepted the message for delivery.
Message sent to Azure IoT Hub
Sucessfully in sendMessage.
02:43:47.848 -> Sending message: {"deviceId":"Feather HUZZAH ESP8266 WiFi","messageId":32,"temperature":21.3,"humidity":21.3}
02:43:47.848 -> IoTHttpClient accepted the message for delivery.
Message sent to Azure IoT Hub
Sucessfully in sendMessage.
02:44:02.627 -> Sending message: {"deviceId":"Feather HUZZAH ESP8266 WiFi","messageId":33,"temperature":21.3,"humidity":21.3}
02:44:02.665 -> IoTHttpClient accepted the message for delivery.
Message sent to Azure IoT Hub
Sucessfully in sendMessage.
02:44:20.208 -> Sending message: {"deviceId":"Feather HUZZAH ESP8266 WiFi","messageId":34,"temperature":21.3,"humidity":21.3}
02:44:20.208 -> IoTHttpClient accepted the message for delivery.
Message sent to Azure IoT Hub
Sucessfully in sendMessage.
02:44:41.371 -> Sending message: {"deviceId":"Feather HUZZAH ESP8266 WiFi","messageId":35,"temperature":21.3,"humidity":21.3}
02:44:41.371 -> IoTHttpClient accepted the message for delivery.
Message sent to Azure IoT Hub
Sucessfully in sendMessage.
02:44:51.405 -> Sending message: {"deviceId":"Feather HUZZAH ESP8266 WiFi","messageId":36,"temperature":21.3,"humidity":21.3}
02:44:51.405 -> IoTHttpClient accepted the message for delivery.
```

☒ Autoscroll ☒ Show timestamp

### BENEFITS

- The user doesn't have to be present at all times to keep a check on the garden/plant.
- Some actions are taken care of by the system itself, so the task of the user becomes quite easy. They just have to go through the initial phase of installing the system and then not have to worry about it.

### CHALLENGES OVERCOME



- We overcame the problem of integrating the cloud data along with the weather forecasting model and both work together generating the required outputs.
- The data is now displayed on the graph in real time and getting stored in the Azure storage for rainfall prediction at the same time. It also monitors the temperature level and takes the necessary action

## CONCLUSION

**i** We were successfully able to demonstrate a smart gardening system in which the user just had to go through the initial setup phase in which they setup the sensors in their gardening environment and after that monitor it from anywhere and at any time using cloud. Some actions like turning the shades on when the temperature rises above a threshold were automatically performed while some other actions have to be done by the user when they are informed about it.

## FUTURE SCOPE

- We could send an email to the user to inform them about what actions need to be performed to maintain ideal conditions for the garden/plant.
- We could install a sprinkler that could water the soil automatically when the soil moisture falls down below a value.
- Using the machine learning model for rain prediction we could inform the user about what conditions to maintain for the day.