Title: Building a Smart Garden with Azure IoT

In this project, we'll build a smart garden system using Azure IoT that allows us to monitor and control the irrigation of our plants remotely. We'll use a Raspberry Pi as the device to collect data and control the irrigation system, and we'll use Azure IoT Hub and Azure Functions to process and visualize the data in real time.

Prerequisites

Before getting started, you'll need the following:

An Azure account

A Raspberry Pi

Sensors to collect data (we'll use a soil moisture sensor and a temperature and humidity sensor in this example)

An irrigation system (we'll use a solenoid valve and a water pump in this example)

The Azure IoT Device SDK for Python

Step 1: Set up an Azure IoT Hub and Register a Device

The first step is to set up an Azure IoT Hub and register a device in the hub. This will allow the Raspberry Pi to send and receive data from the cloud. To set up an IoT Hub and register a device, follow the steps in the "Getting Started with Azure IoT".Make sure to copy the connection string for your device, as you'll need it later.

Step 2: Collect Data from the Sensors

Next, you'll need to set up your sensors and collect data from them. This will typically involve connecting the sensors to the Raspberry Pi and writing code to read the sensor values.

Here's an example of how to collect soil moisture and temperature and humidity data from a DHT11 sensor using Python:

```
import Adafruit_DHT

# Set up the DHT11 sensor
sensor = Adafruit_DHT.DHT11

# Read the soil moisture
moisture = read_soil_moisture()

# Read the temperature and humidity
humidity, temperature = Adafruit_DHT.read_retry(sensor, 4)

# Print the values
print("Soil Moisture: {0:0.1f}%".format(moisture))
print("Temperature: {0:0.1f}C".format(temperature))
print("Humidity: {0:0.1f}%".format(humidity))
```

Step 3: Control the Irrigation System

Now that you're collecting data from the sensors, you can use it to control the irrigation system. To do this, you'll need to connect the solenoid valve and water pump to the Raspberry Pi and write code to control them based on the sensor values.

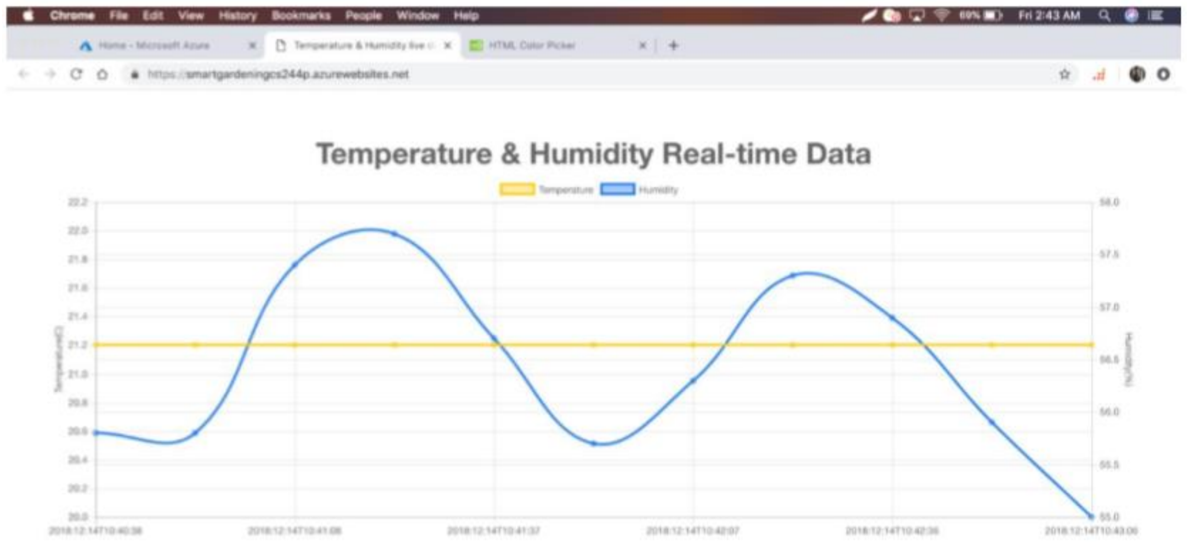
Here's an example of how to turn on the irrigation system if the soil moisture is below a certain threshold:

```
import RPi.GPIO as GPIO

# Set up the GPIO pins for the solenoid valve and water pump
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)

# Turn on the water pump
GPIO.output(22, GPIO.HIGH)

# Turn on the solenoid valve if the soil moisture is below a certain threshold
if moisture < 50:
    GPIO.output(17, GPIO.HIGH)
```

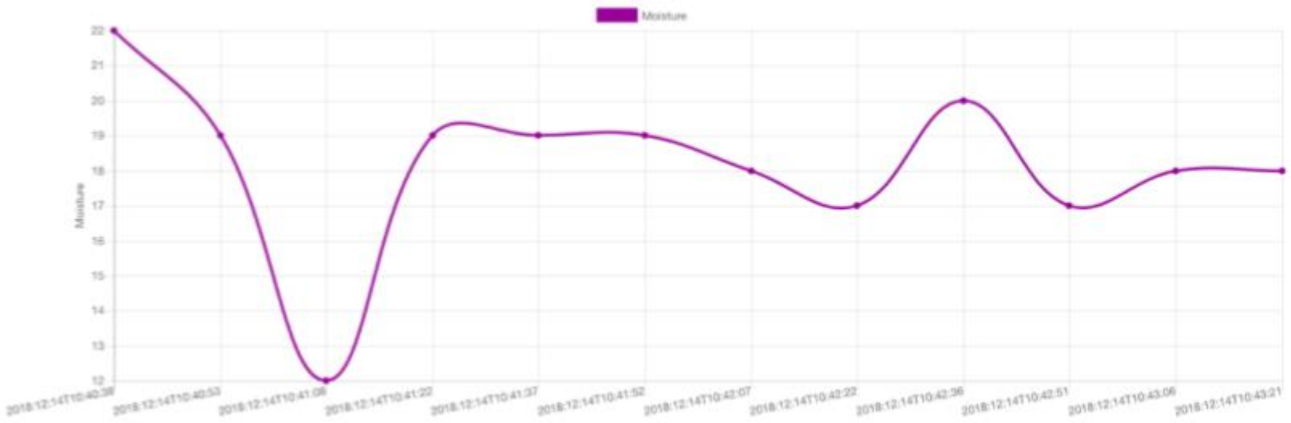
Step 4: Send Data to the IoT Hub

Now that you're collecting and processing data from the sensors,

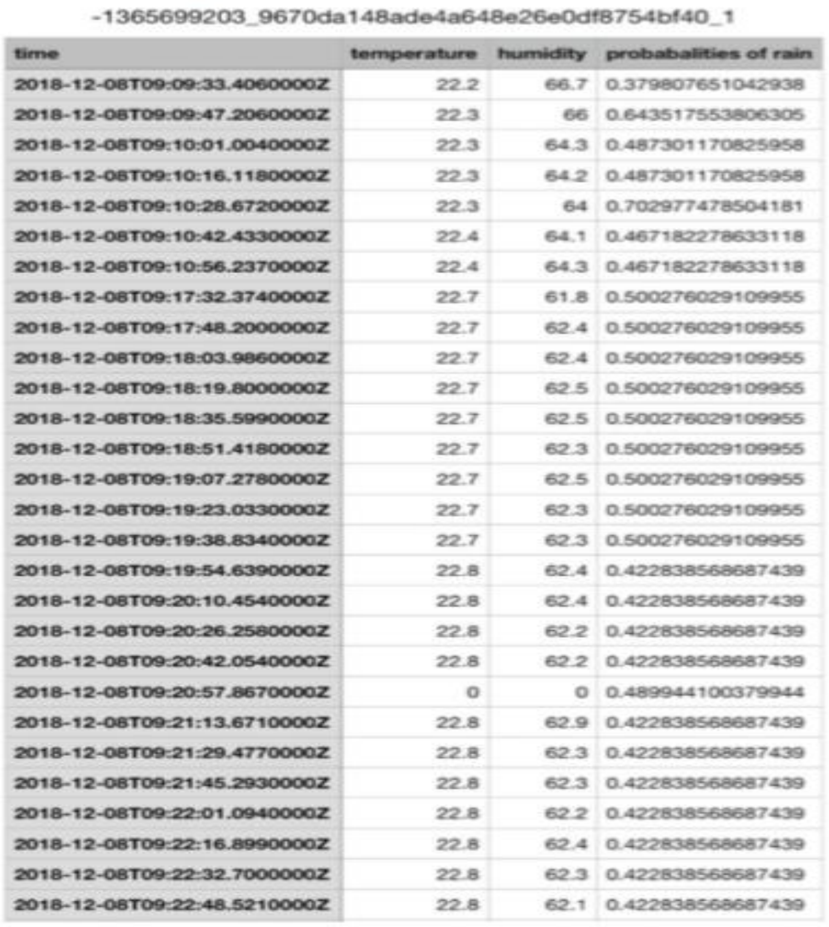
1.Dashboard for statistics


Temperature & Humidity Real-time Data
Temperature
Humidity
Temperature(C)
Humidity(%)
https://smartgardeningcs244p.azurewebsites.net

2.Moinster Real Time Data



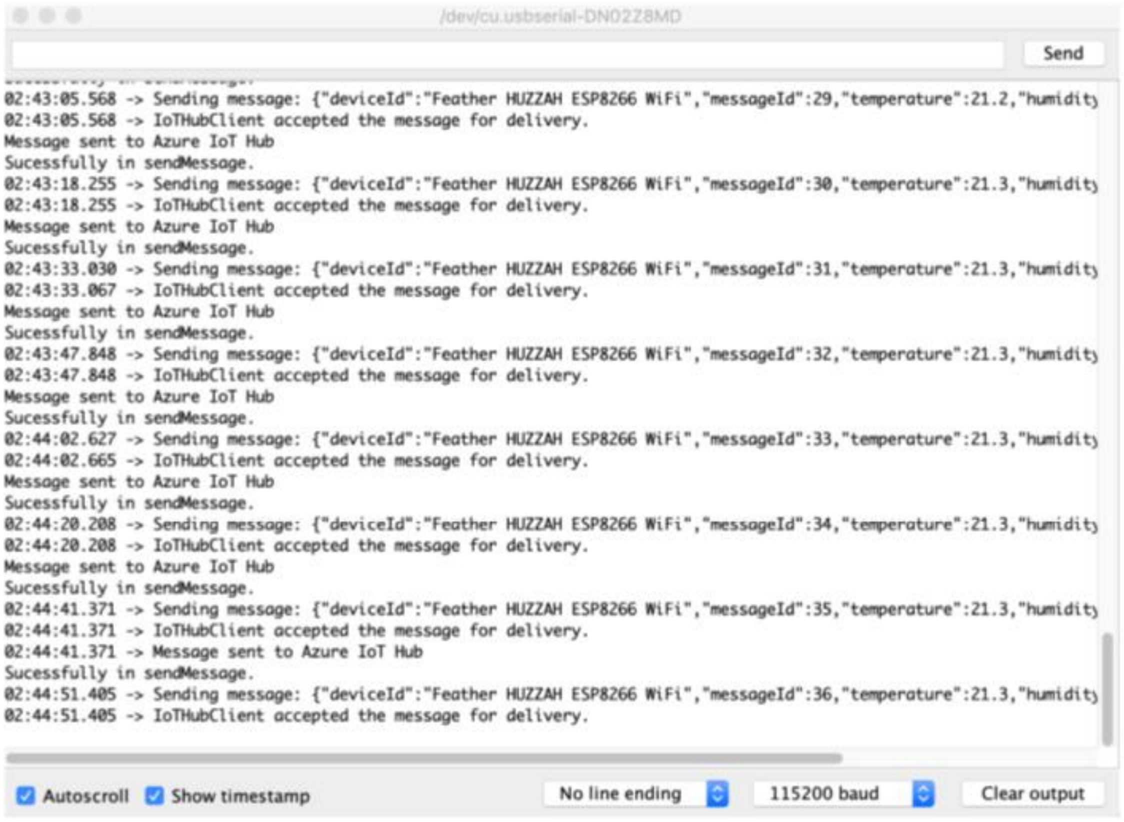
3.Rainfall prediction



-1365699203_9670da148ade4a648e26e0df8754bf40_1

time	temperature	humidity	probabalities 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
02:43:05.568 -> IoTHubClient 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
02:43:18.255 -> IoTHubClient 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
02:43:33.067 -> IoTHubClient 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
02:43:47.848 -> IoTHubClient 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
02:44:02.665 -> IoTHubClient 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
02:44:20.208 -> IoTHubClient 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
02:44:41.371 -> IoTHubClient accepted the message for delivery.
02:44:41.371 -> 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
02:44:51.405 -> IoTHubClient accepted the message for delivery.
```

High-level overview of the architecture for a smart garden system using Azure IoT:

```
+----------+          +------------+          +-----------+
| Smart    |          | Azure IoT  |          | Azure     |
| Phone or |  sends   | Hub        |  sends   | Functions |
| Computer |---------> data to     |---------> data to   |
+----------+          | cloud      |          | cloud     |
                      +------------+          +-----------+
                                                    |
                                                    | receives
                                                    |
                                               +-----------+
                                               | Azure     |
                                               | Stream    |
                                               | Analytics |
                                               +-----------+
                                                    |
                                                    | processes
                                                    |
                                               +-----------+
                                               | Azure     |
                                               | Power BI  |
                                               |           |
                                               +-----------+
```

In this architecture, the smart phone or computer acts as the user interface for the system, allowing the user to view and control the irrigation of the garden remotely. The Raspberry Pi is the device that is installed in the garden and is responsible for collecting data from the sensors and controlling the irrigation system.

The Raspberry Pi sends the sensor data to the Azure IoT Hub, which acts as a message hub for communication between the device and the cloud. The data is then sent to Azure Functions, which can process and analyze the data as needed.

Finally, the processed data is sent to Azure Stream Analytics, which can analyze the data in real time and send it to Azure Power BI for visualization and reporting. The user can access the

Power BI dashboard from their smart phone or computer to view the data and control the irrigation system.

Data flow

1. The smart phone or computer sends a request to the Raspberry Pi to collect data from the sensors or control the irrigation system.
2. The Raspberry Pi collects data from the sensors and sends it to the Azure IoT Hub using the Azure IoT Device SDK.
3. The Azure IoT Hub receives the sensor data and stores it in a message queue.
4. An Azure Function is triggered by the arrival of new data in the message queue. The function retrieves the data from the queue and processes it as needed. This might involve performing calculations, aggregating data, or saving the data to a database.
5. The processed data is sent to Azure Stream Analytics, which analyzes the data in real time and sends it to Azure Power BI for visualization and reporting.
6. The user can access the Power BI dashboard from their smart phone or computer to view the data and control the irrigation system.
7. If the user sends a request to control the irrigation system, the request is sent from the smart phone or computer to the Raspberry Pi.
8. The Raspberry Pi receives the request and controls the irrigation system accordingly. This might involve turning on or off the solenoid valve or water pump, or adjusting their settings.

In a smart garden system using Azure IoT, you can use the following APIs to build and manage the system:

- Azure IoT Hub REST API: This API allows you to send and receive messages from devices, manage device identities, and access other features of Azure IoT Hub. You can use this API to send sensor data from the Raspberry Pi to the IoT Hub, and to receive commands from the user to control the irrigation system.
- Azure Functions HTTP Trigger API: This API allows you to trigger an Azure Function via an HTTP request. You can use this API to trigger an Azure Function when new data is received in the IoT Hub, or to send commands to the Raspberry Pi to control the irrigation system.
- Azure Stream Analytics REST API: This API allows you to create and manage Stream Analytics jobs, which can process and analyze data in real time. You can use this API to

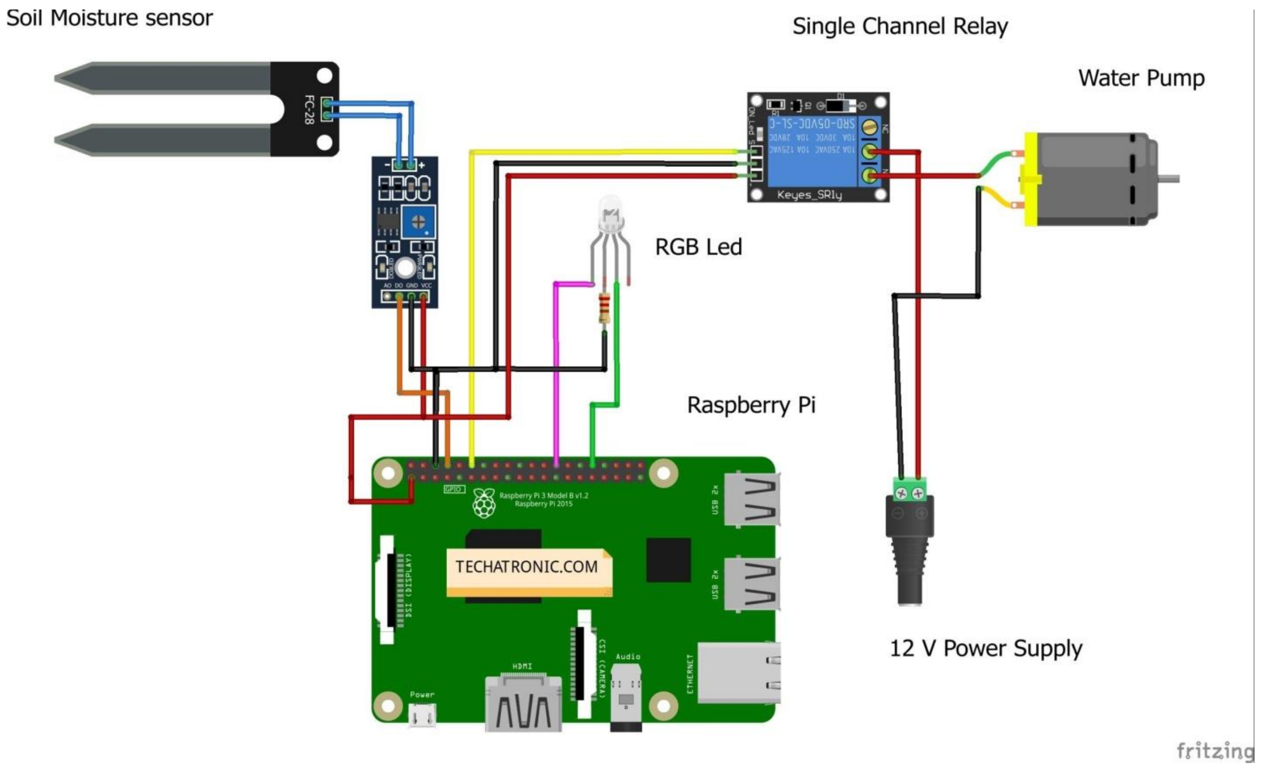
create a Stream Analytics job that processes the sensor data and sends it to Power BI for visualization and reporting.

- Azure Power BI REST API: This API allows you to create and manage Power BI dashboards and reports. You can use this API to create a dashboard that displays the sensor data and allows the user to control the irrigation system.

BASIC PROJECT WORKFLOW

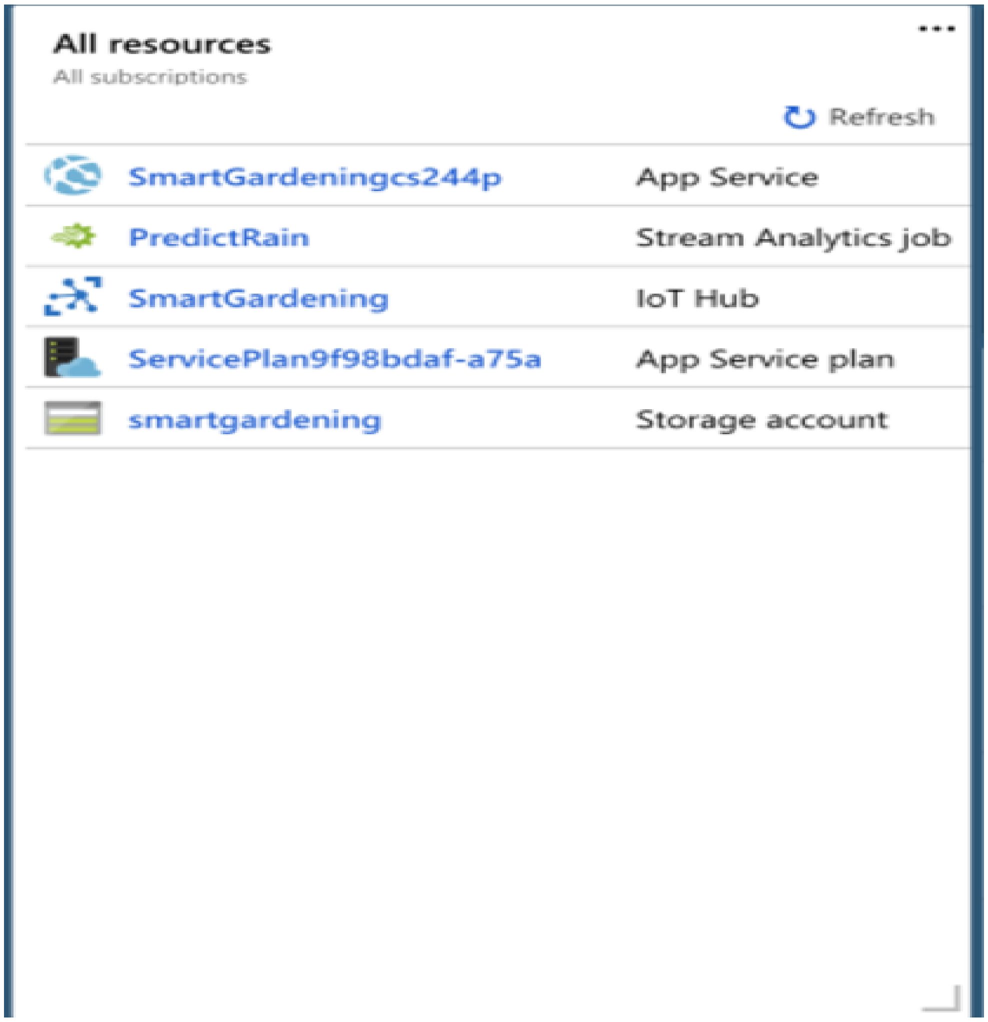
1.User Side:

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.

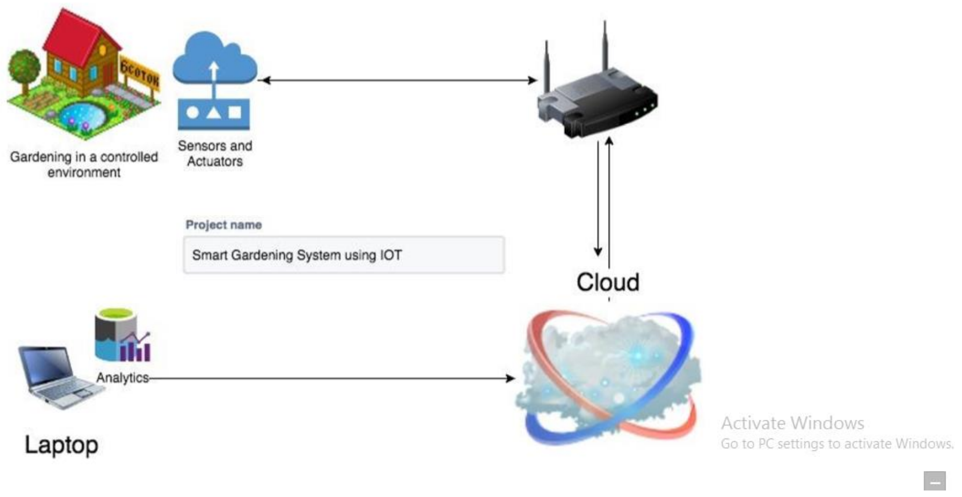


2. Cloud Server

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.



SmartGardeningcs244p	App Service
PredictRain	Stream Analytics job
SmartGardening	IoT Hub
ServicePlan9f98bdaf-a75a	App Service plan
smartgardening	Storage account


Gardening in a controlled environment
Sensors and Actuators
Project name
Smart Gardening System using IOT
Cloud
Analytics
Laptop
Activate Windows
Go to PC settings to activate Windows.

There are several benefits to building a smart garden system using Azure IoT:

1. Remote monitoring and control: With Azure IoT, you can monitor and control the irrigation of your garden remotely using a smart phone or computer. This allows you to keep an eye on your plants and make sure they are getting the right amount of water, even when you're not physically present.
2. Real-time data visualization: Azure IoT allows you to visualize the data from your garden in real time using tools like Azure Power BI. This helps you understand the current conditions in your garden and make informed decisions about how to care for your plants.
3. Scalability: Azure IoT is a cloud-based platform, which means it can scale to meet the needs of your garden as it grows. You can easily add more sensors or devices to the system as needed, without having to worry about running out of capacity.
4. Integration with other Azure services: Azure IoT integrates seamlessly with other Azure services, such as Azure Functions and Azure Stream Analytics. This allows you to build a complete end-to-end solution for your smart garden, with all the data processing and visualization capabilities you need.
5. Cost-effective: Azure IoT offers a range of pricing options, including a free tier, which makes it an affordable option for building a smart garden system.

Future Scope

1. Integration with other systems: You could integrate the smart garden system with other systems, such as a home automation system or a weather forecasting service, to provide additional functionality and improve the user experience.
2. Advanced data analytics: You could use machine learning and other advanced data analytics techniques to improve the accuracy and reliability of the system. For example, you could use machine learning to predict when plants need watering, or to optimize the irrigation schedule for different types of plants.
3. Improved user interface: You could improve the user interface for the system by adding features like voice control or support for multiple languages.
4. Support for multiple gardens: You could expand the system to support multiple gardens, allowing users to monitor and control multiple gardens from a single dashboard.
5. Connected devices: You could add connected devices to the system, such as smart sprinklers or soil sensors, to further improve the accuracy and functionality of the system.

Project link :

https://github.com/DPR-4/Smart_garden_iot_azure

THANK YOU