PROJECT KICKOFF

Project Manager: Avinash Subramaniam M

Project Name: Smart Agriculture system based on IoT - SB43859
Project ID: SPS_PRO_101

Project Summary:

Our project "Smart Agriculture system based on IoT" aims to improve the efficiency and productivity of crop harvesting. The essential factors for crop production like soil moisture, temperature, and weather conditions are provided to the user using a GUI. The user can control the motors to irrigate the land as per the weather condition and other requirements.

Project Requirements:

1. IoT Simulator Sensor
2. IBM Cloud Platform
3. IBM IoT Platform
4. Node Red
5. OpenWeather API
6. Python IDE

Functional Requirements:

1. Checking of soil moisture level, temperature and humidity.
2. Weather Forecast feasibility.
3. Control Motors for irrigation of the crop.
4. Monitoring of the crops through smartphone application.

Project Deliverables:

Cloud Establishment

1. Creating a IoT device using IBM IoT platform
2. Connection Establishment between IoT Sensor Simulator and IoT device
3. Graphical Visualisation of the data recieved in IBM Cloud

Node Red Configuration

4. Connection establishment between NodeRed and IBM IoT platform
5. Data representation in UI (Temperature, Humidity, Object Temperature)

OpenWeather Configuration

6. OpenWeather API Configuration
7. OpenWeather API Integration with NodeRed

GUI Integration

8. Motor Control Development
9. Building a Web App
10. GUI Development
11. Connection Establishment Between GUI and IBM IoT Platform

Project Schedule:

14/05/2020 - 22/05/2020 : Cloud Establishment , Node Red Configuration

23/05/2020 - 30/05/2020 : OpenWeather Configuration , GUI Integration

PROJECT REPORT

INTRODUCTION

1.1 Overview

Our project, Smart Agriculture system based on IoT - SB43859, based on IoT equips the farmer with a mobile app to monitor soil moisture, humidity and temperature to water his crop accordingly. The Weather Forecasting API informs the user of the surrounding Weather Conditions. The Motors used to irrigate the land is controlled by the user based on the moisture requirement for the crop.

1.2 Purpose

The day-to-day routine of a farmer is hectic and overloaded. In order to yield a profit out of his harvest he needs to keep a check on details like soil moisture, soil fertility, weather, and temperature etc. Single handedly, all these tasks take a lot of time. Our project “Smart Agriculture System Based on IoT” aims to make the routine tasks of a farmer simple and effective.

LITERATURE SURVEY

2.1 Existing Problem

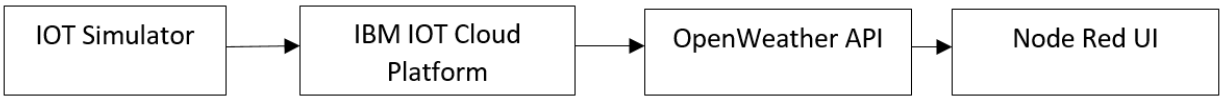
Farming is labour intensive work. Labour intensive work includes labor cost and susceptibility for error. Thus there a lot of problems that farmers face like Pest Infestation, Changing Weather Conditions and scarcity of ground water etc.

2.2 Proposed Solution

The solution developed incorporates a web application that can ease the farmers work load and also boost the chances of proper harvest. The Web Application shows the user necessary data like Weather Conditions, Temperature and Humidity, motor on/off buttons. The user can irrigate the land by using the motor on/off buttons.

THEORITICAL ANALYSIS

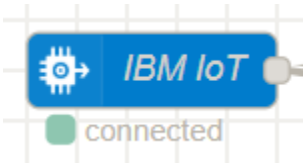
3.1 Block Diagram



3.2 Software Designing

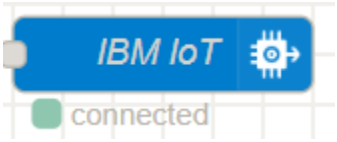
The Node RED platform offers a variety of functionalities for designing Web UI Application.
The Node Red employs the mqtt protocol.
Some of the node red nodes used in this project are listed are follows:

- IBM Input Node:



This Node offers functionality to connect the node red ui to the IBM IOT Device created in the IBM IOT Platform.

- IBM Output Node:

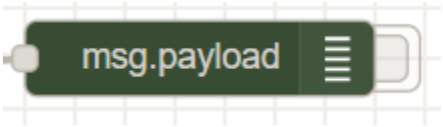


This node offers the functionality to send the command to the device in the IBM IOT Platform.

- Function Node:

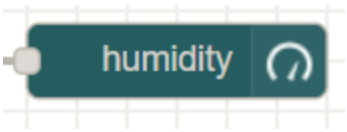
This node performs a function specified in the function block of the node.

- Debug Node:



This node displays the debug messages of the node connected to.

- Gauge Dashboard Node:



This node is used to display the data in the UI using a Gauge.

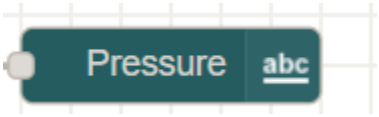
- Inject Node:

This node is used to kick start a flow. The flow execution starts as the node is clicked.

- Http Request Node:

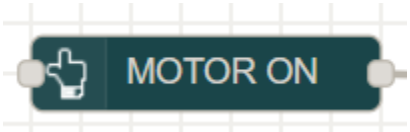
This node is used for http request from the node red to the OpenWeather API server.

- Label Dashboard Node:



This node is used to display the data as an plain text in node red ui.

- Button Dashboard Node:



This node behaves as button when clicked upon performs a certain task. This node is used for turning the motor on and off.

Python:

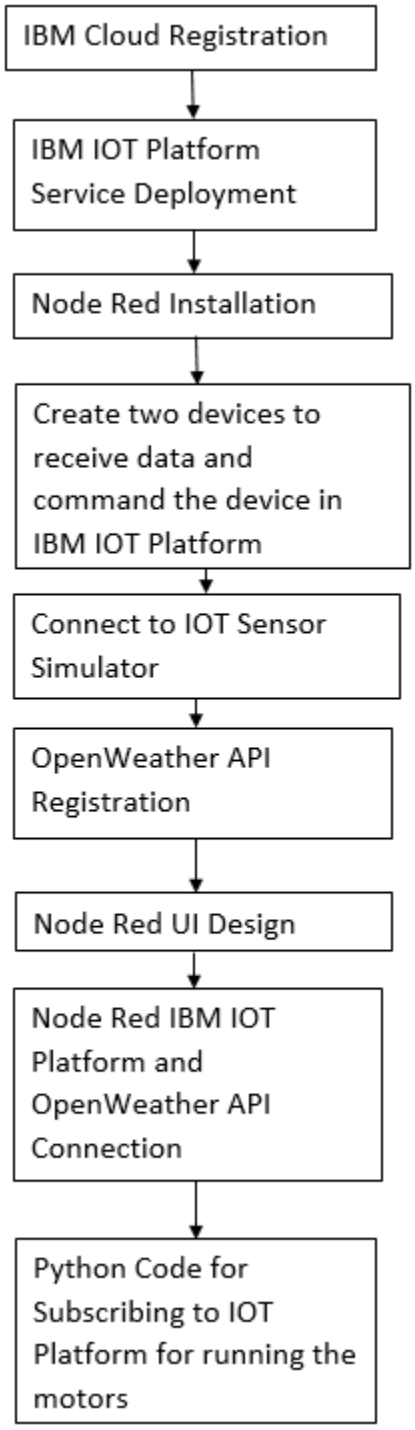
Python is used to subscribe to the IBM IOT Platform and issue the motor commands to the deivce.

EXPERIMENTAL INVESTIGATION

The project is built on the Node Red Platform which runs on mqtt protocol. The Node Red Platform has various packages for designing a Web UI. The following are the packages used in the Node Red for this project:

- node-red-contrib-scx-ibmiotapp
- node-red-dashboard

FLOWCHART


IBM Cloud Registration
IBM IOT Platform Service Deployment
Node Red Installation
Create two devices to receive data and command the device in IBM IOT Platform
Connect to IOT Sensor Simulator
OpenWeather API Registration
Node Red UI Design
Node Red IBM IOT Platform and OpenWeather API Connection
Python Code for Subscribing to IOT Platform for running the motors

RESULT

The developed web application performs all the functionalities listed. The web application shows the user the weather data, soil moisture, temperature and access buttons for motors.

ADVANTAGES & DISADVANTAGES

Advantages

1. The work load for the farmer is reduced.
2. Cost effective Solution
3. Labour Cost Reduced

Disadvantages

1. Internet Network requirement
2. Incorporation of enhancements needs time
3. Scalability for example: when employed for a large farm

APPLICATIONS

The Web UI implementation covers:

1. Temperature and Humidity Reading
2. Weather Condition Monitoring
3. Irrigation Control

CONCLUSIONS

The Web UI designed can be deployed and enhanced to include more functionalities to ease the farmer's labour. Reliability and Cost effectiveness are the key factors provided. The harvest of the farmer can be boosted using this application.

FUTURE SCOPE

The future enhancements that can be added to this project are:

- Pest Control - Pest Control can be incorporated by using EMR
- Ploughing - Mechanism to plough seeds in the desired location in the field
- Automatic Irrigation Capsule - Water Capsule to irrigate the entire field based on the weather conditions automatically

All these enhancements can increase the application's efficiency.

BIBILOGRAPHY

- https://www.researchgate.net/publication/271530086_A_Review_on_the_Development_of_Integrated_Pest_Management_and_Its_Integration_in_Modern_Agriculture
- https://www.researchgate.net/publication/260303884_Automated_Irrigation_System_Using_a_Wireless_Sensor_Network_and_GPRS_Module
- https://nodered.org/
- https://cloud.ibm.com/docs/IoT/index.html

APPENDIX

Source Code

```
import time
import sys
import ibmiotf.application
import ibmiotf.device

#Provide your IBM Watson Device Credentials
organization = "24uoem" # repalce it with organization ID
deviceType = "Motors" #replace it with device type
deviceId = "12345" #repalce with device id
authMethod = "token"
authToken = "rH-JL-@iElw52Kp_0s"#repalce with token

def myCommandCallback(cmd):
        print("Command received: %s" % cmd.data)
        if cmd.data['command']=='motoron':
                print("MOTOR ON")
        elif cmd.data['command'] == 'motoroff':
            print("MOTOR OFF")

try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method":
    authMethod, "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
    #.....................................
```

```
except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()

deviceCli.connect()

while True:
      T=50;
      H=32;
      #Send Temperature & Humidity to IBM Watson
      data = { 'Temperature' : T, 'Humidity': H }
      #print data
      #def myOnPublishCallback():
         #print ("Published Temperature = %s C" % T, "Humidity = %s %%" % H, "to IBM
    Watson")

      #success = deviceCli.publishEvent("event", "json", data, qos=0,
    on_publish=myOnPublishCallback)
      #if not success:
         #print("Not connected to IoTF")
      time.sleep(1)
      deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```