# Project Report

## Project Title: Smart Agriculture System based on IOT

Presented by: Bhavesh Panjwani

Guided by: Durgaprasad Sir

## Table of Contents

# 1. Introduction:

## 1.1. Overview:

- Smart Agriculture System based on IoT can monitor soil moisture and climatic conditions to grow and yield a good crop.
- The farmer can also get the real time weather forecasting data by using external platforms like Open Weather API.
- Farmer is provided a mobile app using which he can monitor the temperature, humidity and soil moisture parameters along with weather forecasting details.
- Based on all the parameters he can water his crop by controlling the motors using the mobile application.
- Even if the farmer is not present near his crop he can water his crop by controlling the motors using the mobile application from anywhere.
- Here we are using the Online IoT simulator for getting the Temperature, Humidity and Soil Moisture values.

## 1.2. Purpose:

- To control motors from a remote location using mobile application.
- Automatic mode for watering the plants.

# 2. Literature Survey:

## 2.1. Existing Problem:

- Farmer cannot monitor the conditions of the farm from any remote location. Also to water the plants he/she needs to go to farm.

## 2.2. Purposed Solution:

- Using IOT we can develop an application which using sensors monitor the soil condition can automatically control the motors.

- Also the application will have a manual mode in which farmer can control the motors manually from a remote location.

# 3. Theoretical Analysis:

## 3.1. Block Diagram:

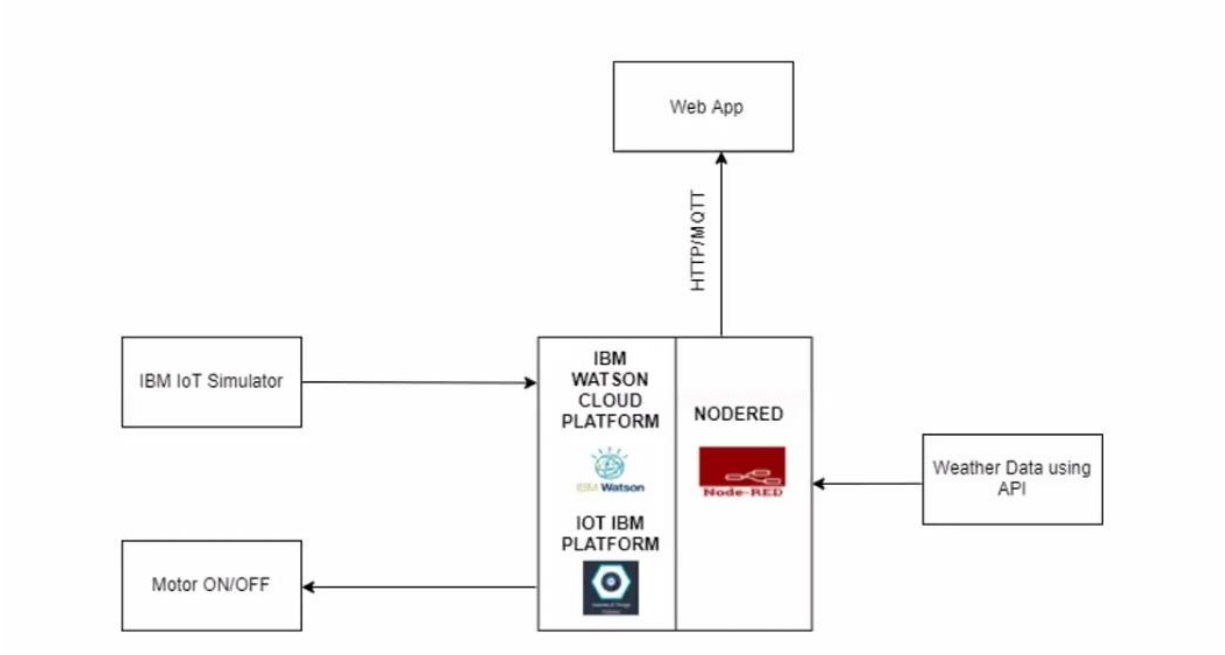



Figure 1: Block Diagram of smart agriculture system using IOT

## 3.2. Software Design:

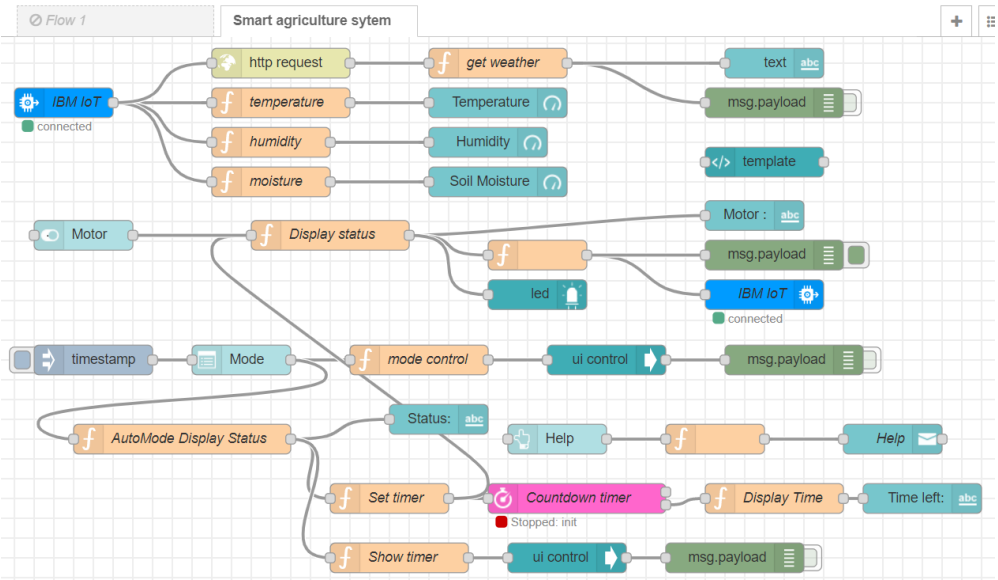
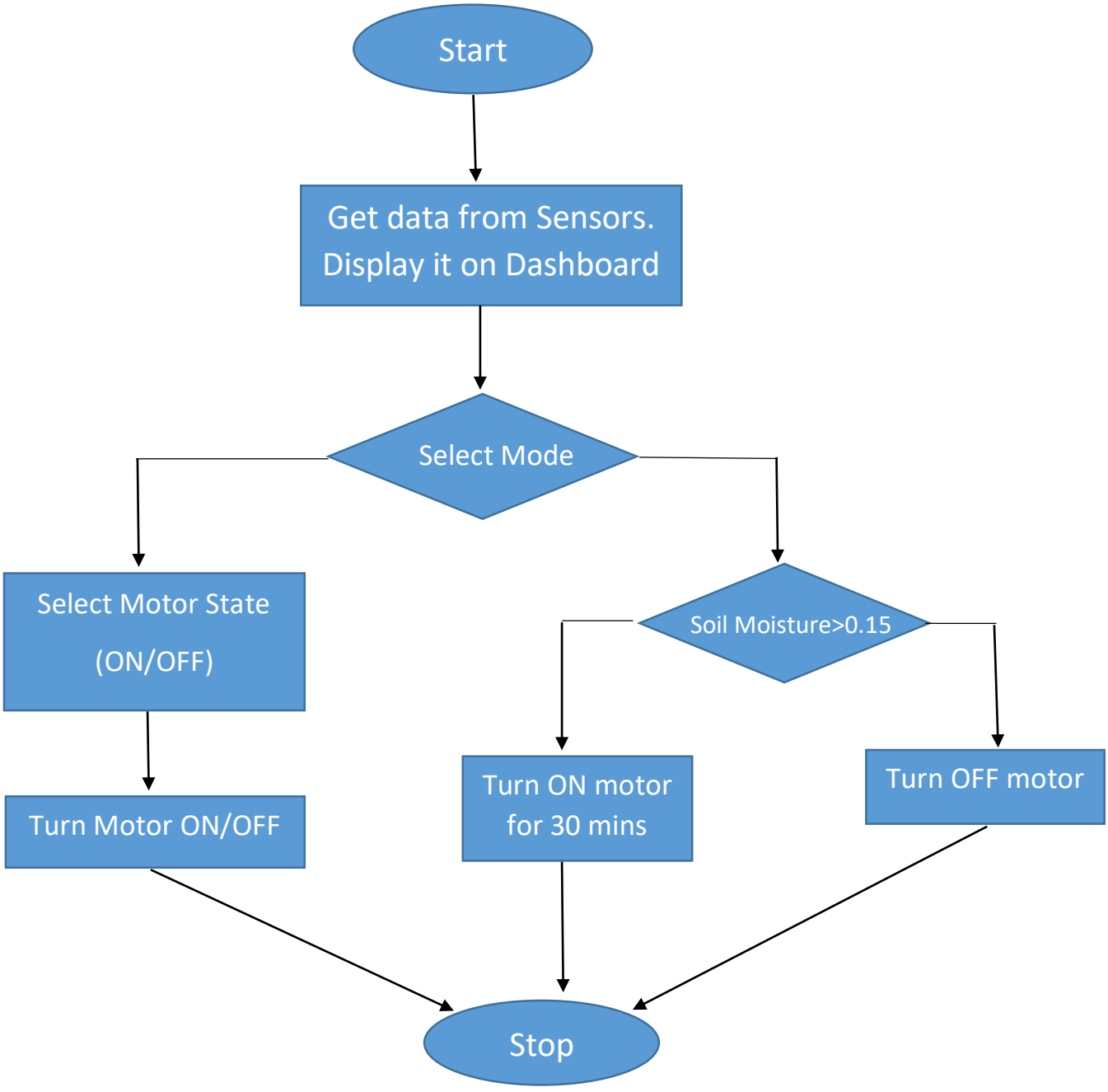



Figure 2: Flow of Node-red.

4. Flowchart:




5. Result:

- Smart Agriculture system based on IOT successfully implemented.

6. Advantages and Disadvantages:
    6.1. Advantages:
        - Motors can be controlled from any remote location.
        - Weather conditions and soil parameters can be monitored and viewed from anywhere.
        - Automatic mode if farmer is not available.
    6.2. Disadvantages:
        - Internet is needed to send and receive data from the sensors and the application.
        - Cost of some sensors is high.
7. Applications:
    - Can be used at any farm.
8. Conclusion:

    Smart Agriculture system based on IOT implemented successfully using IBM cloud Watson IOT platform, Sensor simulator, Node red and python.
9. Future Scope:
    - All the motors are controlled simultaneously in future we can give separate controls for every motor.
    - According to the crops planted the automatic mode needs to be adapted.
    - Alert the user with the upcoming weather conditions and actions that needs to be taken.
10. Bibliography:
    - Node-red : https://nodered.org/
    - IBM cloud: https://cloud.ibm.com/login
    - Http request
    - Openweather API
    - Create UI using Node red
    - Python code for Smart System based on IOT

# Appendix

## A. Source Code:

```
import time
import sys
import ibmiotf.application # to install pip install ibmiotf
import ibmiotf.device

#Provide your IBM Watson Device Credentials
organization = "okpvr0" #replace the ORG ID
deviceType = "iotdevice"#replace the Device type wi
deviceId = "12345678"#replace Device ID
authMethod = "token"
authToken = "0T(cyX+0lQC7O@4mx0" #Replace the authtoken

def myCommandCallback(cmd): # function for Callback
        print("Command received: %s" % cmd.data)
        if cmd.data['command']=='motor_on':
                print("MOTOR ON IS RECEIVED")

        elif cmd.data['command']=='motor_off':
                print("MOTOR OFF IS RECEIVED")

        if cmd.command == "setInterval":

                if 'interval' not in cmd.data:
                        print("Error - command is missing required information:
'interval'")
                else:
                        interval = cmd.data['interval']
        elif cmd.command == "print":
                if 'message' not in cmd.data:
                        print("Error - command is missing required information:
'message'")
```

```
            else:
                output=cmd.data['message']
                print(output)

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()

# Connect and send a datapoint "hello" with value "world" into the cloud as
an event of type "greeting" 10 times
deviceCli.connect()

while True:

      deviceCli.commandCallback = myCommandCallback

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