SMART AGRICULTURE BASED ON INTERNET OF THINGS

INTRODUCTION:

1.1 Overview:

In modern world, every appliances humans use are connected to internet. Controlling machines and other useful devices via internet facilitates less time consumption and ease in control over the appliances. This project focuses on automation of the agricultural works carried out in the field by live monitoring and access to control by various IOT devices (like raspberry pie) by internet.

1.2 Purpose:

The purpose of this project is to develop a user friendly Mobile application interface that enables the farmer to monitor the temperature, climatic conditions, water level etc of the cultivation field even from long distances. The application must also enable the farmer to control the irrigation facilities to the land. The mobile handset must have access to internet services wirelessly for the full pledged working of the model.

2. LITERATURE SURVEY:

2.1 Existing problem:

Human beings witness extreme climatic changes, deteriorating soil fertility, dry lands and collapsed ecosystem these days. The increasing population and decreasing cultivable lands and farmers make will surely make people's future worse in case of healthy and sufficient food.
To encourage agriculture even when engaged in a full profession other than field work, and to facilitate the farmers an easy manipulation of the manual works that needs to be done crucially.

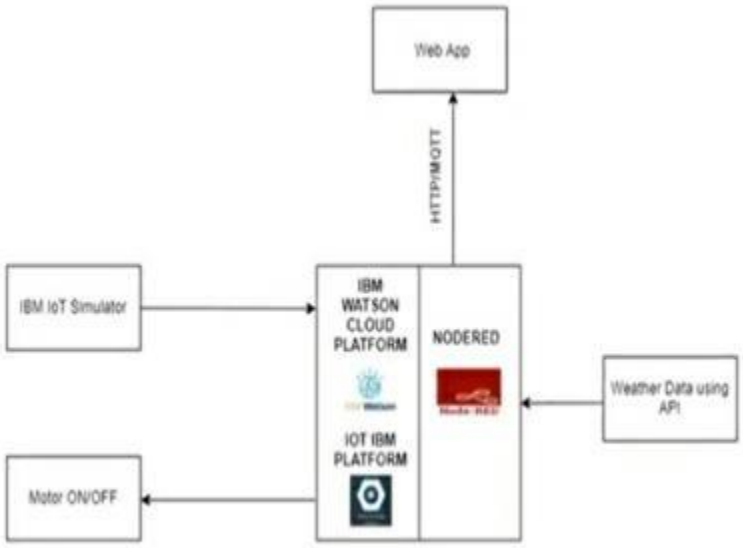
2.2 Proposed solution:

Internet of Things concept of automation and remote access needs to be applied in the scene. Providing the farmer a user friendly easy to use mobile application is the goal.

The android based Farming system is an automatic irrigation system which performs multiple operations in the field of agriculture; this project uses a centralized controller which is programmed to receive the input signal of multiple sensors of the field. Once the controller receives this signal, it generates an output that drives a relay for operating the water pump and

other circuitry which provides automatic control action on field. If the user sees the moisture level of ever cannel has sufficient amount then user can switch off the motor easily using GUI.

Block Diagram:



HARDWARE/SOFTWARE DESIGN:

Install the required tools and create the required accoounts.

1. SETTING UP PYTHON IDE

2.CREATE A DEVICE IN THE IBM WATSON IOT PLATFORM

3.INSTALL NODE-RED LOCALLY

4.CONNECT THE IOT DEVICE TO A SIMULATOR

5.VISUALISING THE DATA

6.CONNECTING TO OPENWEATHER API

7.CREATING THE UI USING NODE-RED

RESULT:

The response from the dashboard to the simulator matched and changed the measuring gauges perfectly. Using the http request node and open weather node, weather details of the location

from open weather api website were derived and displayed successfully in the dashboard.

Thus the UI perfectly shows variation from the simulator and also commands the control devices like motors and lights which facilitate the ease of access to irrigation and lighting to the field

APPLICATIONS:

1. Automation can be used in the fields of medical, home care, pet care etc.

2. Turning lights on and off by mobile is handy for old people.

3. In agriculture, building canals which directly lets the water to plant's surface can reduce water wastage. When irrigation controls is given via mobile, farmer's workload is reduced.

4. The same way as turning lights on and off, food containers and water containers placed above bowls of pets, can be controlled. This becomes helpful to the owner to feed the pet when he/she is away from home..

FUTURE SCOPE:

The project can be further extended to enabling the usage of Al in the agriculture ecosystem. It helps a lot in implementation of Precision farming. We can suggest crops based on the climatic conditions of the data. Based on the water level, we can alert the farmer or automatically turn the motor off.

APPENDIX:

. Source code:

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

#Provide your IBM Watson Device Credentials
organization = "glgcpo" #replace the ORG ID
deviceType = "Sim"#replace the Device type wi
deviceId = "Device2"#replace Device ID
authMethod = "token"
```

```
authToken = "Test@123" #Replace the authtoken

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

    elif cmd.data['command']=='lightoff':
        print("LIGHT OFF IS RECEIVED")
    if cmd.data['command']=='motoron':
        print("MOTOR ON IS RECEIVED")

    elif cmd.data['command']=='motoroff':
        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()
```