SMARTINTERNZ

Project Name: Smart Agriculture system based on IoT

Project Handled by : Ambati Venkata Sai Nithin

Project Duration : 1 month

Introduction:

Despite the perception people may have regarding the agricultural process, the reality is that today's agriculture industry is data-centered, precise, and smarter than ever. The rapid emergence of the Internet-of-Things (IoT) based technologies redesigned almost every industry including "smart agriculture" which moved the industry from statistical to quantitative approaches. Such revolutionary changes are shaking the existing agriculture methods and creating new opportunities along a range of challenges. IoT devices and communication techniques associated with wireless sensors encountered in agriculture applications are analyzed in detail. What sensors are available for specific agriculture application, like soil preparation, crop status, irrigation, insect and pest detection.Furthermore, the use of unmanned aerial vehicles for crop surveillance and other favorable applications such as optimizing crop yield is considered in this article. State-of-the-art IoT-based architectures and platforms used in agriculture are also highlighted wherever suitable. Finally, based on this thorough review, we identify current and future trends of IoT in agriculture and highlight potential research challenges.

Agriculture is considered as the basis of life for the human species as it is the main source of food grains and other raw materials. It plays vital role in the growth of country's economy. It also provides large ample employment opportunities to the people. Growth in agricultural sector is necessary for the development of economic condition of the country. Unfortunately, many farmers still use the traditional methods of farming which results in low yielding of crops and fruits. But wherever automation had been implemented and human beings had been replaced by automatic machineries, the yield has been improved. Hence there is need to implement modern science and technology in the agriculture sector for increasing the yield. Most of the papers signifies the use of wireless sensor network which collects the data from different types of sensors and then send it to main server using wireless protocol. The collected data provides the information about different environmental factors which in turns helps to monitor the system. Monitoring environmental factors is not enough and complete solution to improve the yield of the crops. There are number of other factors that affect the productivity to great extent. These factors include attack of insects and pests which can be controlled by spraying the crop with proper insecticide and pesticides. Secondly, attack of wild animals and birds when the crop grows up. There is also possibility of thefts when crop is at the stage of harvesting. Even after harvesting, farmers also face problems in storage of harvested crop. So, in order to provide solutions to all such problems, it is necessary to develop integrated system which will take care of all factors affecting the productivity in every stages like; cultivation, harvesting and post harvesting storage. This paper therefore proposes a system which is useful in monitoring the field data as well as controlling the field operations which provides the flexibility.

Project Description:

1. Smart Agriculture System based on IoT can monitor soil moisture and climatic conditions to grow and yield a good crop.
2. The farmer can also get the realtime weather forecasting data by using external platforms like Open Weather API.
3. Farmer is provided a mobile app using which he can monitor the temperature, humidity and soil moisture parameters along with weather forecasting details.
4. Based on all the parameters he can water his crop by controlling the motors using the mobile application.
5. 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.
6. Here we are using the Online IoT simulator for getting the Temperature, Humidity and Soil Moisture values.

Tools Required:

Software:

- IBM CLOUD
- IBM WATSON PLATFORM
- WEATHER API
- NODERED
- PYTHON IDE
- IBM WATSON SENSOR SIMULATOR

IBM CLOUD:

IBM cloud computing is a set of cloud computing services for business offered by the information technology company IBM. IBM Cloud includes infrastructure as a service (IaaS), software as a service (SaaS) and platform as a service (PaaS) offered through public, private and hybrid cloud delivery models, in addition to the components that make up those clouds.

NODERED:

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. Elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON. Since version 0.14, MQTT nodes can make properly configured TLS connections.

IBM Watson IoT platform.:

Get a quick start on your next IoT project. Watson IoT Platform is a managed, cloud-hosted service designed to make it simple to derive value from your IoT devices.

IBM Watson Sensor Simulator:

Its an IBM provided simulator for virtually connecting a sensor with device id from the service

Weather API:

Weather API is web based weather forecaster and describes the weather condition of any location. Its speciality is it can generate app key which can be used in various platforms for generate ontime weather prediction.In this project we used in nodered

PYTHON IDE:

Python is developed under an OSI-approved open source license, making it freely usable and distributable, even for commercial use. Python's license is administered by the Python Software Foundation. The Python Package Index (PyPI) hosts thousands of third-party modules for Python. Both Python's standard library and the community-contributed modules allow for endless possibilities.

Highlights:

- Explore IBM Cloud Platform
- Install the Nodered Locally.
- Connect the IOT Simulator To Watson IOT Platform
- Building A Web App
- Configure Your Device to Receive The Data From The Web Application And Control Your Motors

Procedure:

- Create an account in IBM Cloud. Explore all services and apps in it
- Create a Nodered App and install all require nodes in it
- Create an IOT Platform service in IBM Cloud and link it with the Watson sensor simulator
- Add the device credentials in iot sensor to connect the sensor to IBM IOT Platform
- Note the API KEY and Authentication Token when you create the app in IOT PLATFORM
- IN Nodered, install IBMIOT , Dashboard nodes in Nodered platform
- Drag and drop ibmiot in and out nodes, function nodes, debug nodes in the nodered
- Connect them and deploy , you will find the output in Debug messages

- Configure the Nodered and get the data from IBM IOT platform and open weather api

- Get the data from IBM IOT Device to get the simulator data

- Create an account in Open weather API platform

- Configure your nodered to get the weather forecasting data using Http requests

- Configure the Nodes to display the Weather Parameters which we get from IOT simulator and open weather API in UI

- Configure the nodes for creating buttons and sending commands to IOT platform

- Write a python code to subscribe to IBM IOT platform and get the commands

- In python subsciber.py code, add your device id, credentials, api key and token in the python.

Theory:

In real World,Temperature sensor and Humidity sensor senses the temperature and humidity respectively and if the value crosses the threshold then room heater or cooling fan will be switched ON/OFF automatically providing temperature and humidity maintenance.Node2 will also controls water pump depending upon the soil moisture data sent by sensors.

Humidity sensor: The DHT11 is a basic, low-cost digital temperature and humidity sensor. It gives out digital value and hence there is no need to use conversion algorithm at ADC of the microcontroller and hence we can give its output directly to data pin instead of ADC. It has a capacitive sensor for measuring humidity. The only real shortcoming of this sensor is that one can only get new data from it only after every 2 seconds

Temperature Sensor LM35: The LM35 is precision IC temperature sensor. Output voltage of LM35 is directly proportional to the Centigrade/Celsius of temperature. The LM35 does not need external calibration or trimming to provide accurate temperature range. It is very low cost sensor. It has low output impedance and linear output. The operating temperature range for LM35 is -55° to $+150^{\circ}$C. With rise in temperature, the output voltage of the sensor increases linearly and the value of voltage is given to the microcontroller which is multiplied by the conversion factor in order to give the value of actual temperature.

Moisture sensor: Soil moisture sensor measures the water content in soil. It uses the property of the electrical resistance of the soil. The relationship among the measured property and soil moisture is calibrated and it may vary depending on environmental factors such as temperature, soil type, or electric conductivity. Here, It is used to sense the moisture in field and transfer it to microcontroller in order to take controlling action of switching water pump ON/OFF.

The water pump also gets turned ON if moisture level goes below fixed threshold value.

The main aim is to succesfully build an IOT model in agriculture which helps farmers to easily montor their crops and produce a better yield

Should successfully retrieve the temperature and humidity values from the sensor and show them to farmer using an app , So that based on all the parameters , he can water the crops by controlling the motors which were built to control by him from anywhere with a proper internet connection

We are using Online IOT simulator to bring down the sensor values and show them on the app.To show the various parameters of crop on the mobile application to farmer which can control from anywhere

Python Code:

1. Subscriber.py

```
import time

import sys

import ibmiotf.application # to install pip install ibmiotf

import ibmiotf.device


#Provide your IBM Watson Device Credentials

organization = " 2r9q54 " #replace the ORG ID

deviceType = " nodered "#replace the Device type wi

deviceId = " nodered123 "#replace Device ID

authMethod = "token"

authToken = "12345678" #Replace the authtoken


def myCommandCallback(cmd): # function for Callback

print("Command received: %s" % cmd.data)

if cmd.data['command']=='lighton':

print("MOTOR ON IS RECEIVED")


elif cmd.data['command']=='lightoff':

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()
```

2. publisher.py

```
import time

import sys

import ibmiotf.application

import ibmiotf.device


#Provide your IBM Watson Device Credentials

organization = " 2r9q54 " #replace the ORG ID

deviceType = " nodered "#replace the Device type wi

deviceId = " nodered123 "#replace Device ID

authMethod = "token"

authToken = "12345678" #Replace the authtoken


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;
        ot=45
#Send Temperature & Humidity to IBM Watson
data = {'d':{ 'Temperature' : T, 'Humidity': H,'objTemp':ot }}
#print data
def myOnPublishCallback():
print (data, "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)


# Disconnect the device and application from the cloud
```

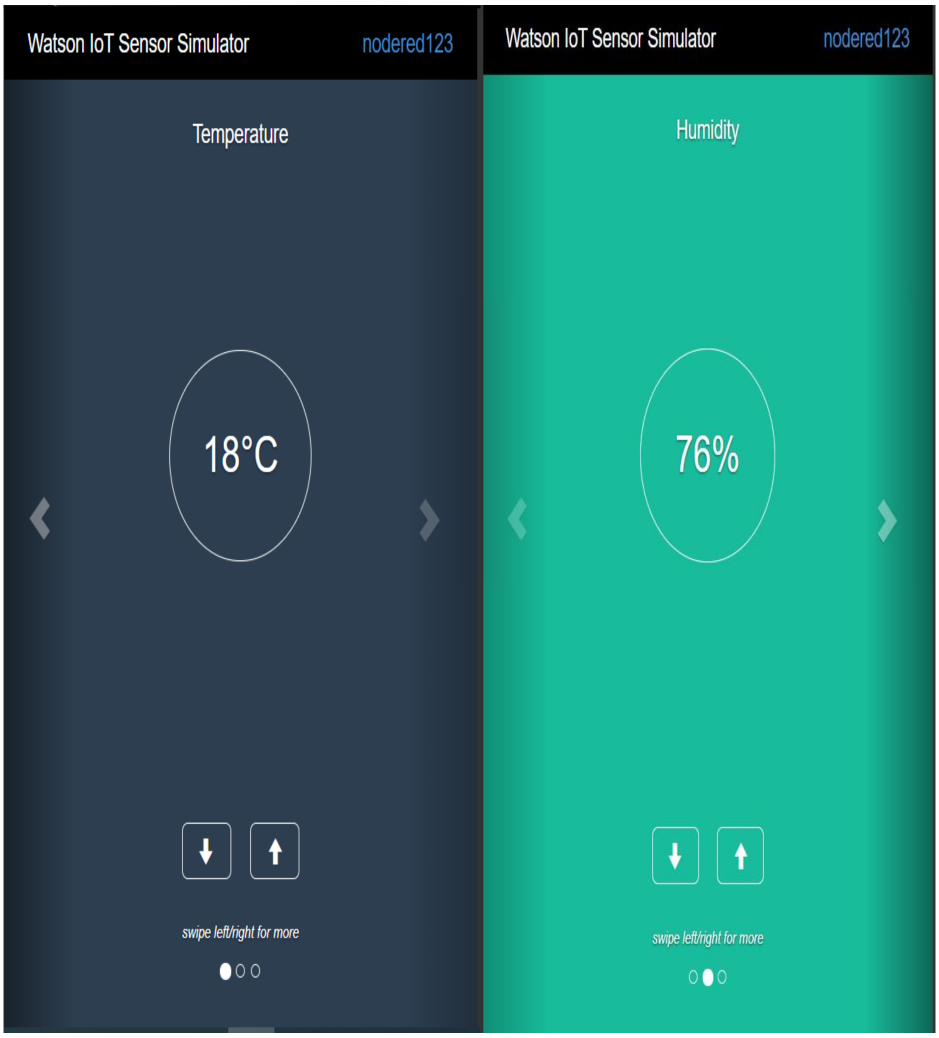
```
deviceCli.disconnect()
```

3.pubsubibm.py

```
import time

import sys

import ibmiotf.application

import ibmiotf.device



#Provide your IBM Watson Device Credentials

organization = " 2r9q54 " #replace the ORG ID

deviceType = " nodered "#replace the Device type wi

deviceId = " nodered123 "#replace Device ID

authMethod = "token"

authToken = "12345678" #Replace the authtoken

myCommandCallback(cmd):

print("Command received: %s" % cmd.data)

if cmd.data['command']=='lighton':

print("LIGHT ON")

elif cmd.data['command'] == 'lightoff':

print("LIGHT 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)
```

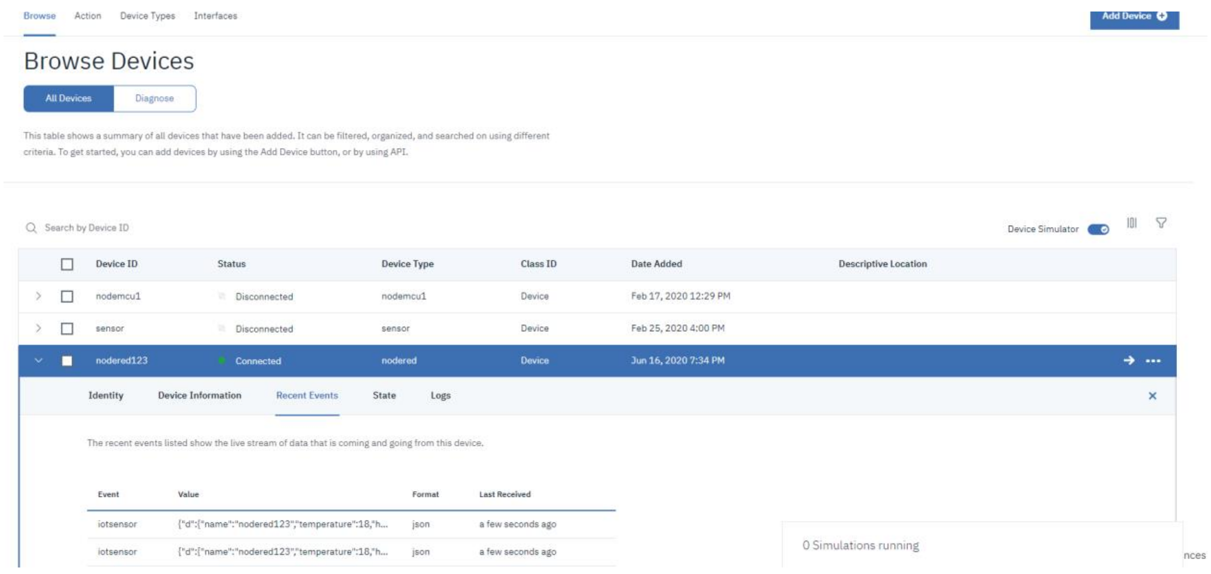
```
deviceCli.commandCallback = myCommandCallback


# Disconnect the device and application from the cloud

deviceCli.disconnect()
```

Project Results:


Watson IoT Sensor Simulator
nodered123
Temperature
18°C
swipe left/right for more
Watson IoT Sensor Simulator
nodered123
Humidity
76%
swipe left/right for more

<u>IBM WATSON SENSOR SIMULATOR</u>

Sevices



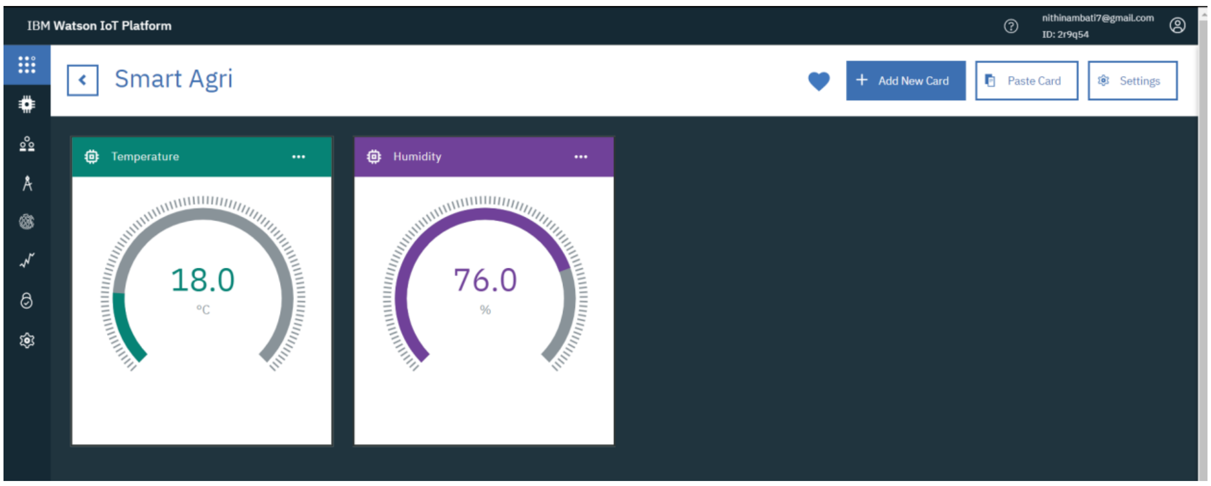
Browse Devices

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

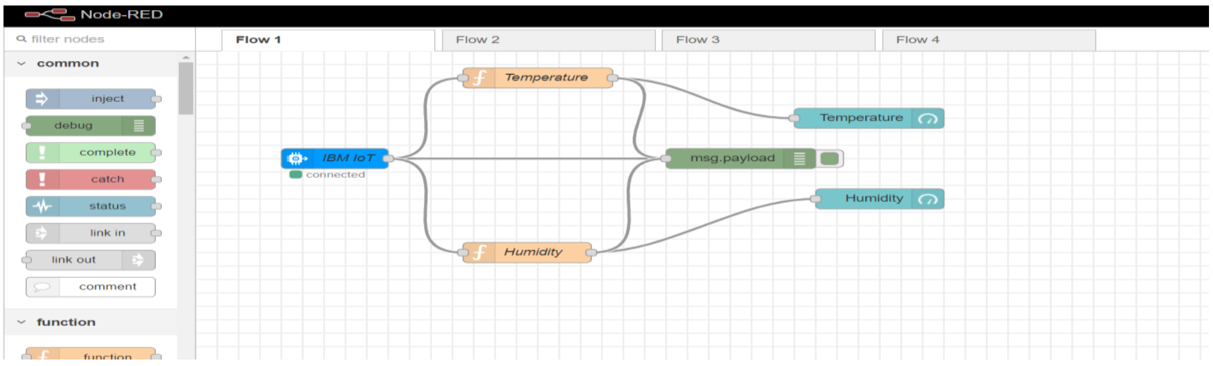
Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
nodemcu1	Disconnected	nodemcu1	Device	Feb 17, 2020 12:29 PM	
sensor	Disconnected	sensor	Device	Feb 25, 2020 4:00 PM	
nodered123	Connected	nodered	Device	Jun 16, 2020 7:34 PM	

The recent events listed show the live stream of data that is coming and going from this device.

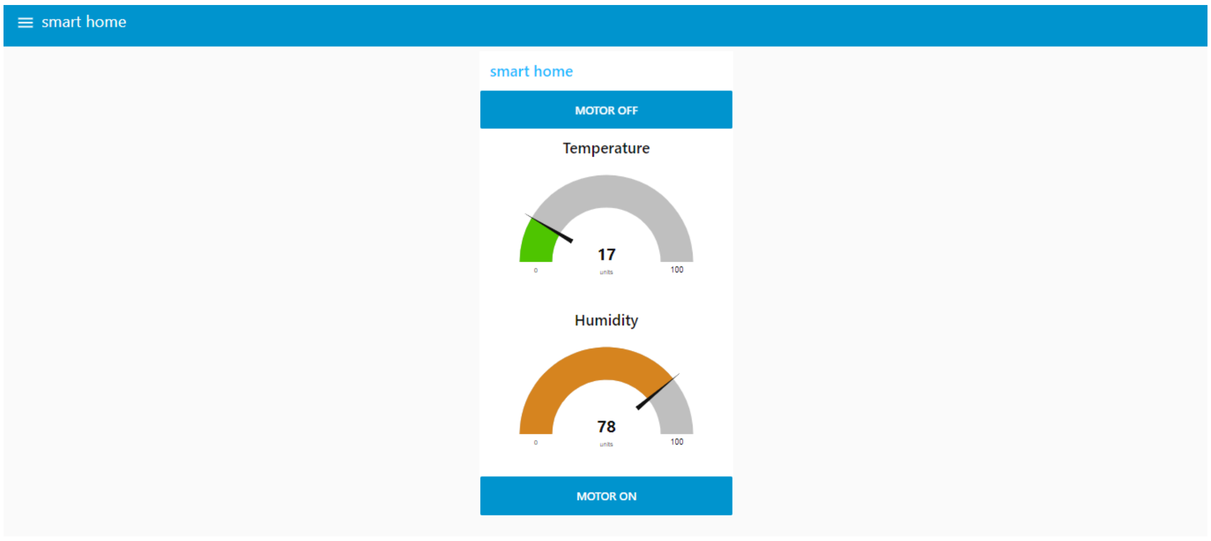
Event	Value	Format	Last Received
iotsensor	{"d":{"name":"nodered123","temperature":18,"h...	json	a few seconds ago
iotsensor	{"d":{"name":"nodered123","temperature":18,"h...	json	a few seconds ago

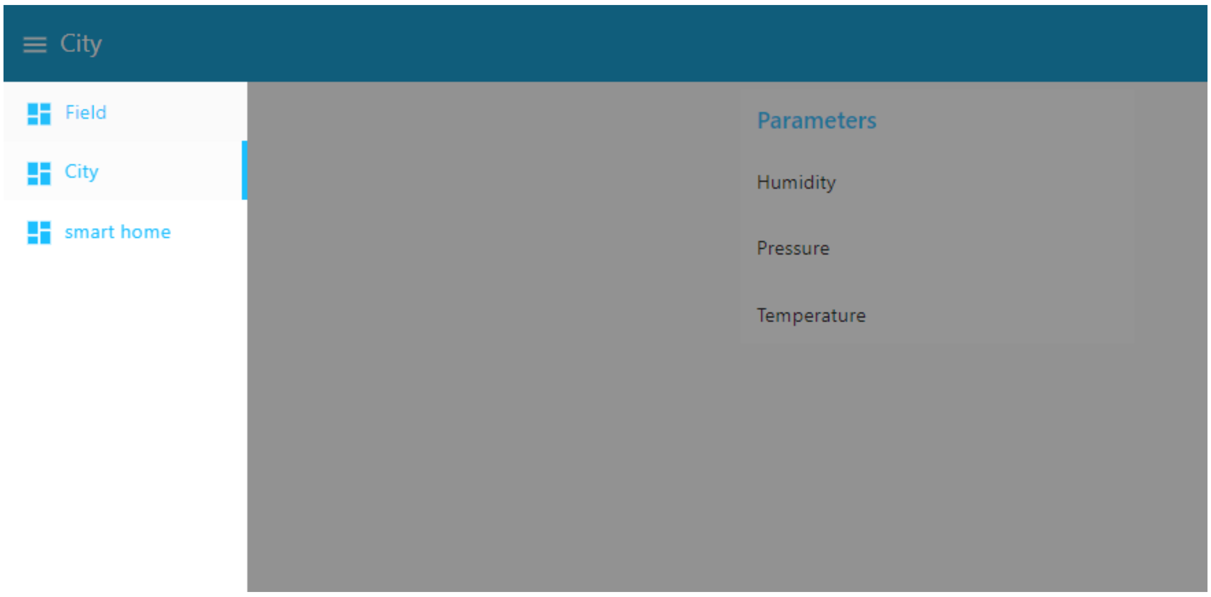


NODERED



NODERED-WEBAPP





Acknowledgement:

I am sincerely thankful to SMARTINTERN for their guidance for my Project . Without their help it was tough job for me to accomplish this task.