

## **SPRINT DELIVERY – 3**

**Team ID**

**PNT2022TMID03620**

**Project Name**

Smart farming -IoT Enabled Smart  
Farming Application

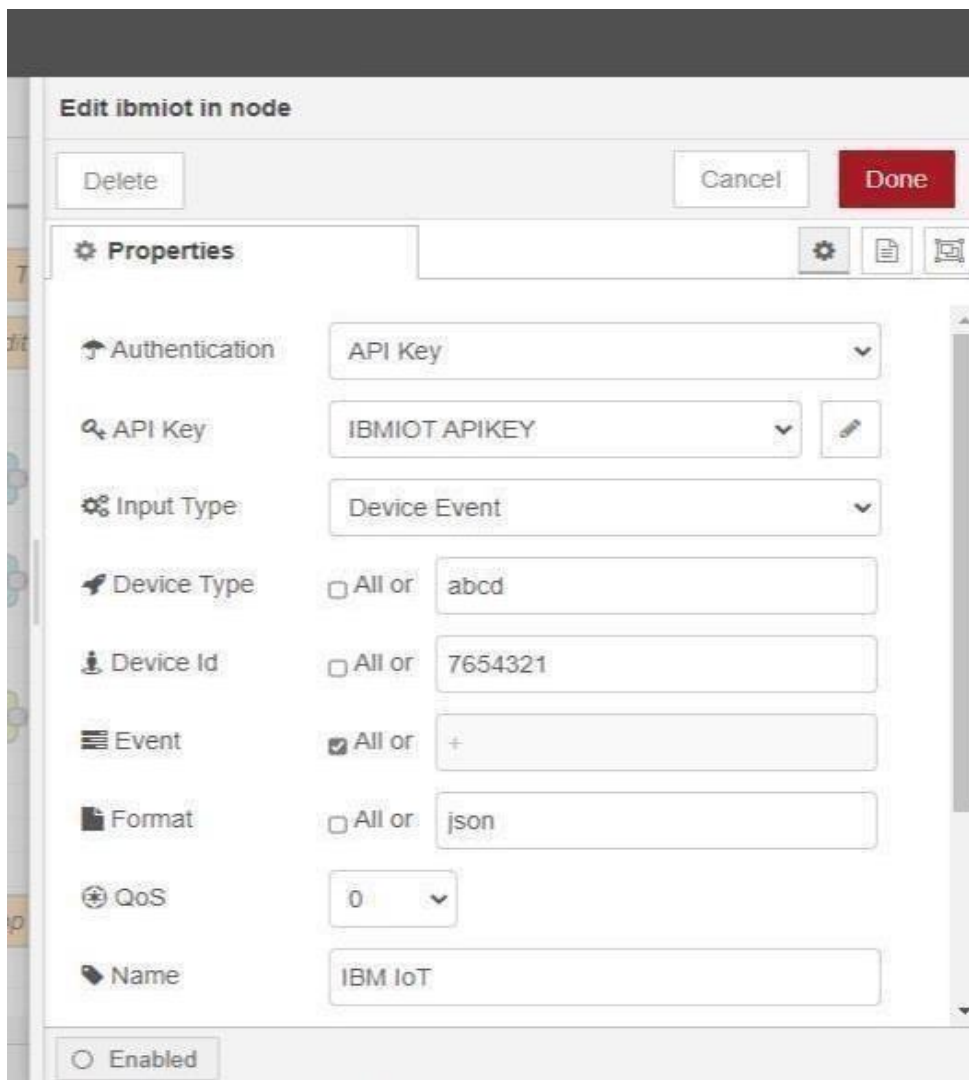
**Date**

12 November 2022

## Configuration of Node-Red to send commands to IBM cloud

ibmiot out node I used to send data from Node-Red to IBM Watson device. So, after adding it to the flow we need to configure it with credentials of our Watson device.

Here we add two buttons in UI



The screenshot shows the 'Edit ibmiot in node' configuration window. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below these is a 'Properties' tab with a settings icon, a document icon, and a preview icon. The configuration fields are as follows:

- Authentication:** A dropdown menu set to 'API Key'.
- API Key:** A text input field containing 'IBMIOT APIKEY' with a search icon on the left and an edit icon on the right.
- Input Type:** A dropdown menu set to 'Device Event'.
- Device Type:** A checkbox labeled 'All or' followed by a text input field containing 'abcd'.
- Device Id:** A checkbox labeled 'All or' followed by a text input field containing '7654321'.
- Event:** A checkbox labeled 'All or' followed by a text input field containing '+'.
- Format:** A checkbox labeled 'All or' followed by a text input field containing 'json'.
- QoS:** A dropdown menu set to '0'.
- Name:** A text input field containing 'IBM IoT'.

At the bottom left, there is a checkbox labeled 'Enabled' which is currently unchecked.

1 -> for motor on

2 -> for motor off

We used a function node to analyse the data received and assign command to each number.

The Java script code for the analyses is:

```
if(msg.payload===1)
```

```
msg.payload={"command": "ON"};
```

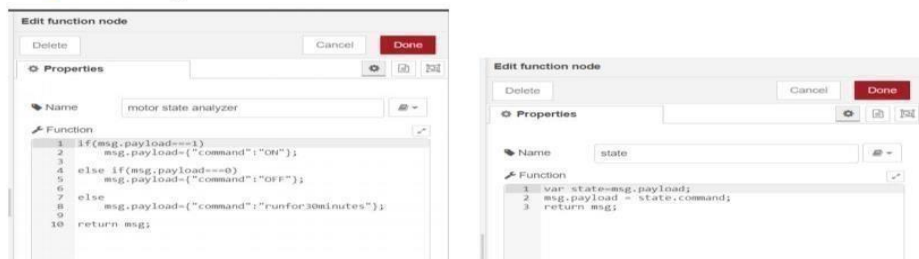
```
else if(msg.payload===0)
```

```
msg.payload={"command": "OFF"};
```

Then we use another function node to parse the data and get the command and represent it visually with text node.

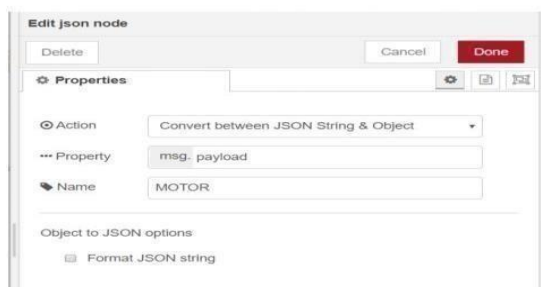
The Java script code for that function node is:

```
var state=msg.payload;  
msg.payload = state.command;  
return msg;
```

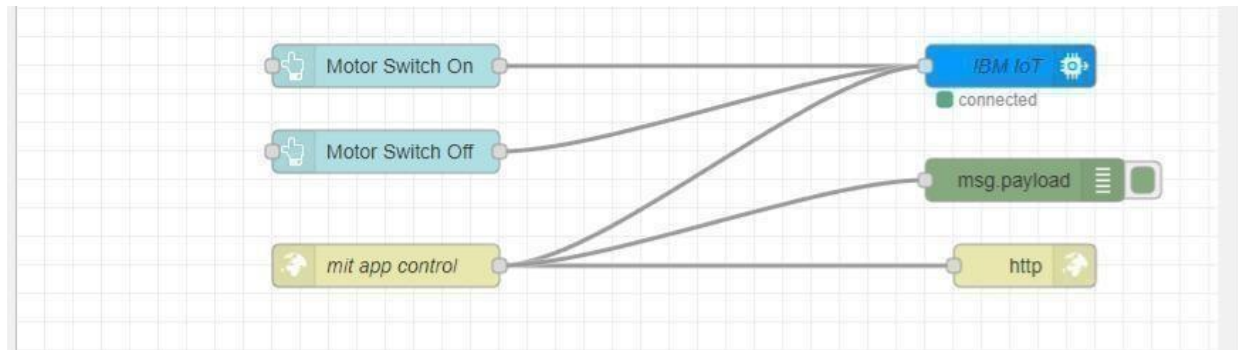


The above images show the java script codes of analyser and state function nodes.

Then we add edit json node to the conversion between JSON string & object and finally connect it to IBM IoT Out.



Edit JSON node needs to be configured like this



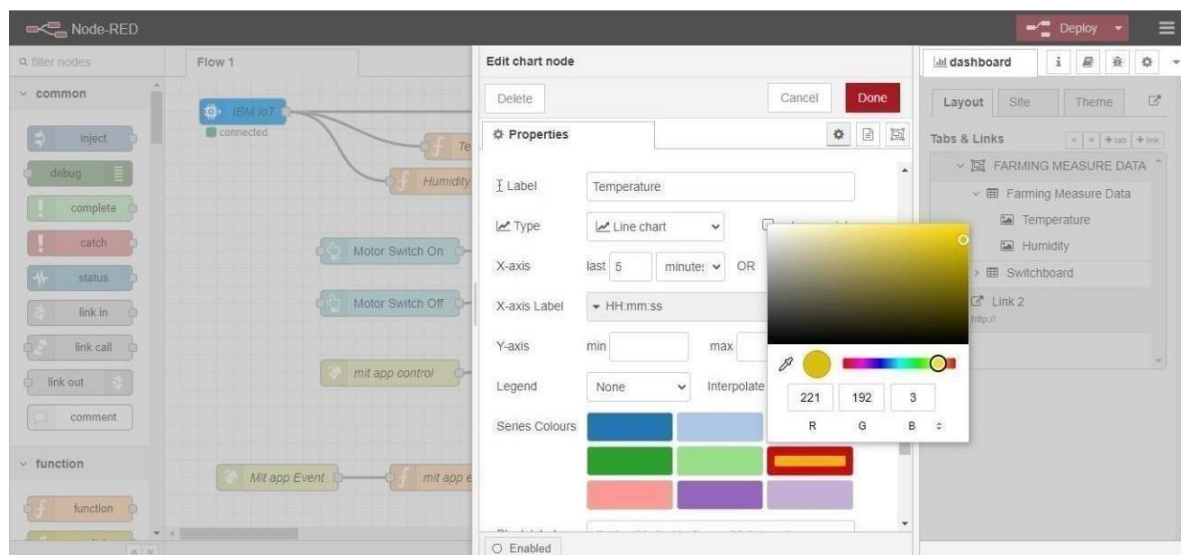
This is the program flow for sending commands to IBM cloud.

## Adjusting User Interface

In order to display the parsed JSON data a Node-Red dashboard is created

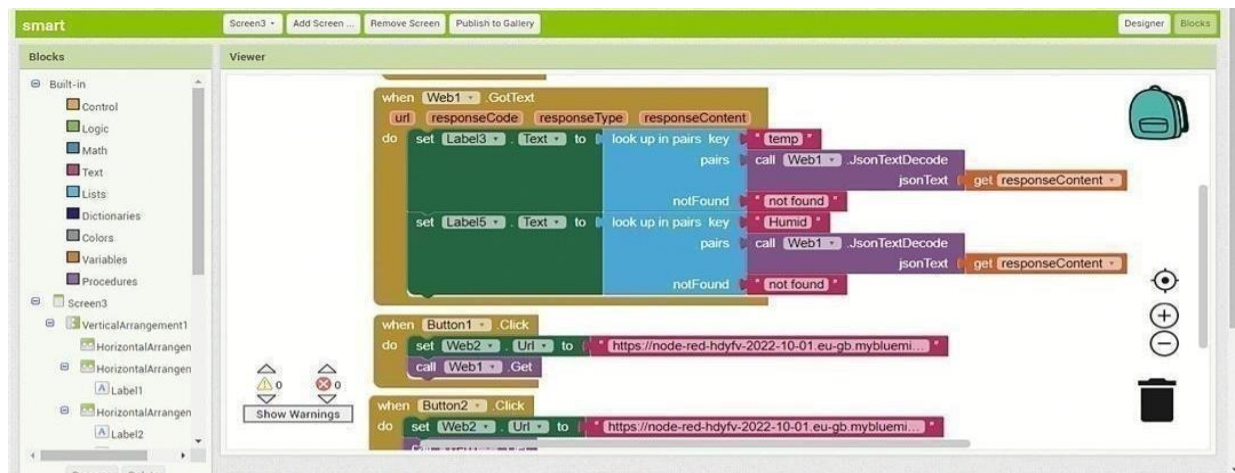
Here we are using Gauges, text and button nodes to display in the UI and helps to monitor the parameters and control the farm equipment.

Below images are the Gauge, text and button node configurations.



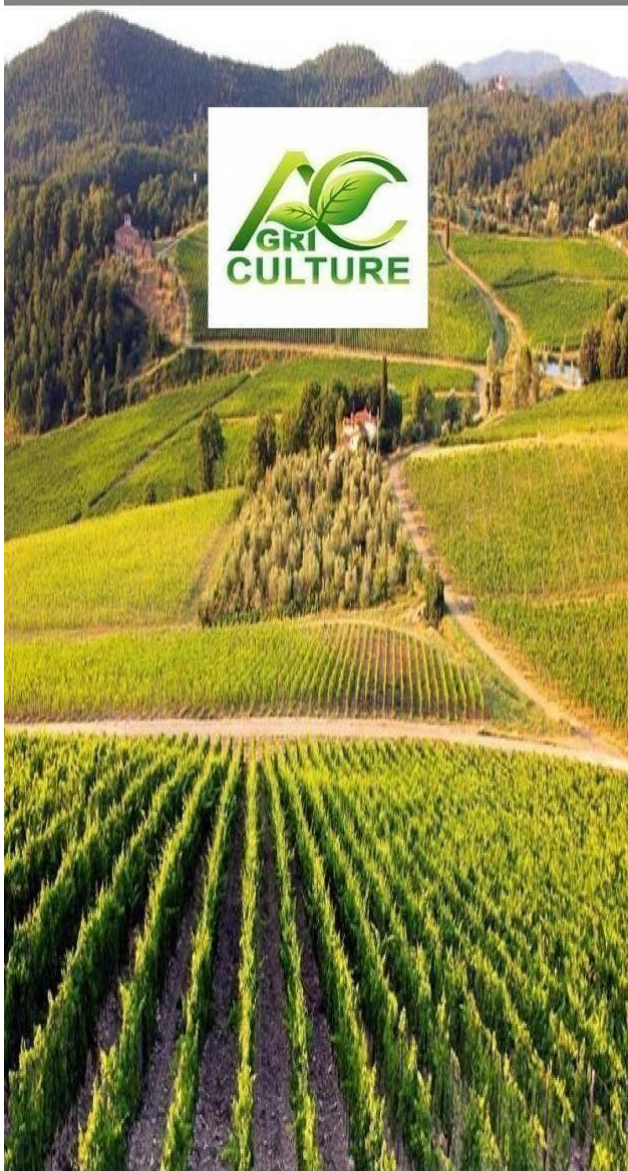
The screenshot displays the MIT App Inventor IDE interface. On the left, a sidebar lists various widgets: slider, numeric, text input, date picker, colour picker, form, text, gauge, chart, audio out, notification, ui control, and template. The central workspace shows a flowchart titled 'Flow 1'. The flowchart begins with an 'IBM IoT' sensor block (connected) that branches into three function blocks: 'Temperature', 'Humidity', and 'Moisture'. Each function block connects to a corresponding 'msg payload' block. These three 'msg payload' blocks then connect to three 'IBM IoT' actuator blocks: 'Motor Switch On', 'Motor Switch Off', and 'MIT App Control'. The 'MIT App Control' block connects to an 'http' block. At the bottom, a 'MIT App Event' block connects to a 'mit app event' function block, which then connects to an 'http' block. On the right, a 'debug' console shows the output of the app, including JSON data for sensor readings and motor control commands.

## BLOCK DIAGRAM

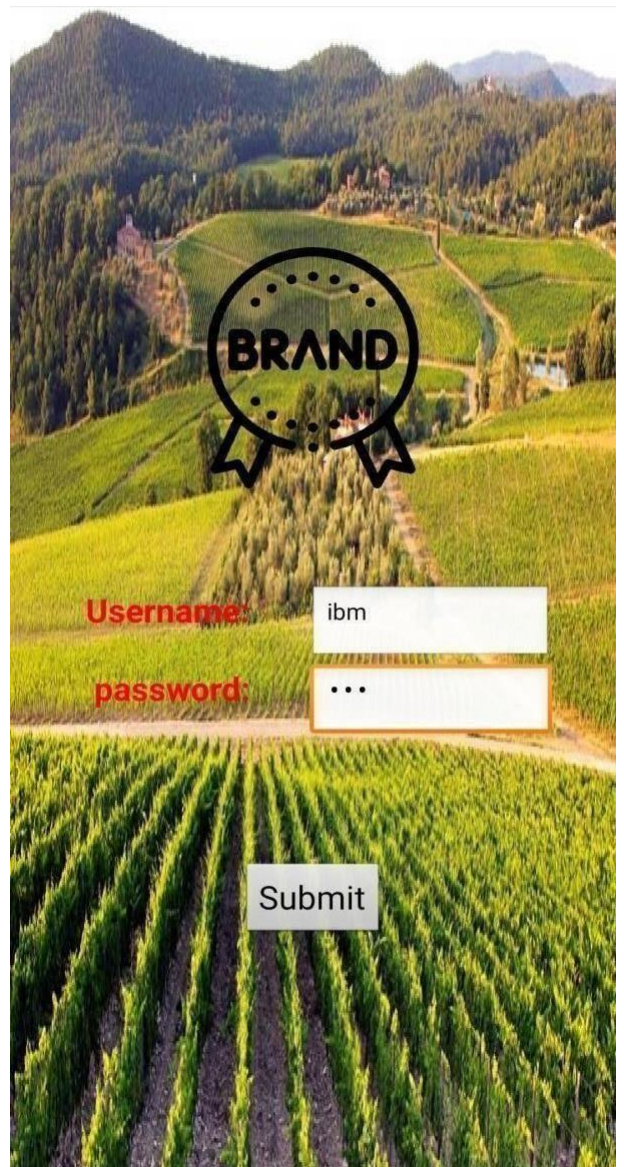




Screen1

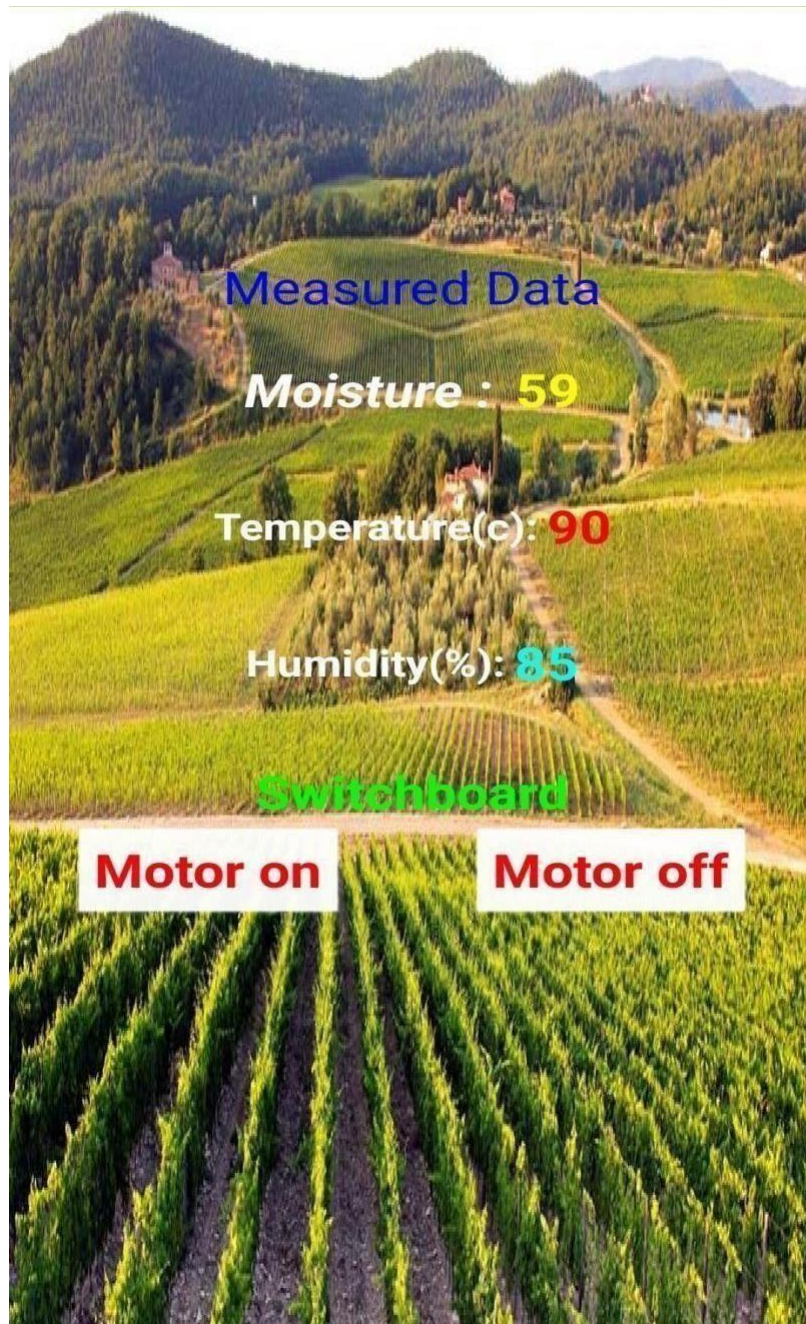


SCREEN – 1



SCREEN - 2





**SCREEN - 3**

## Web APP UI Home Tab

