

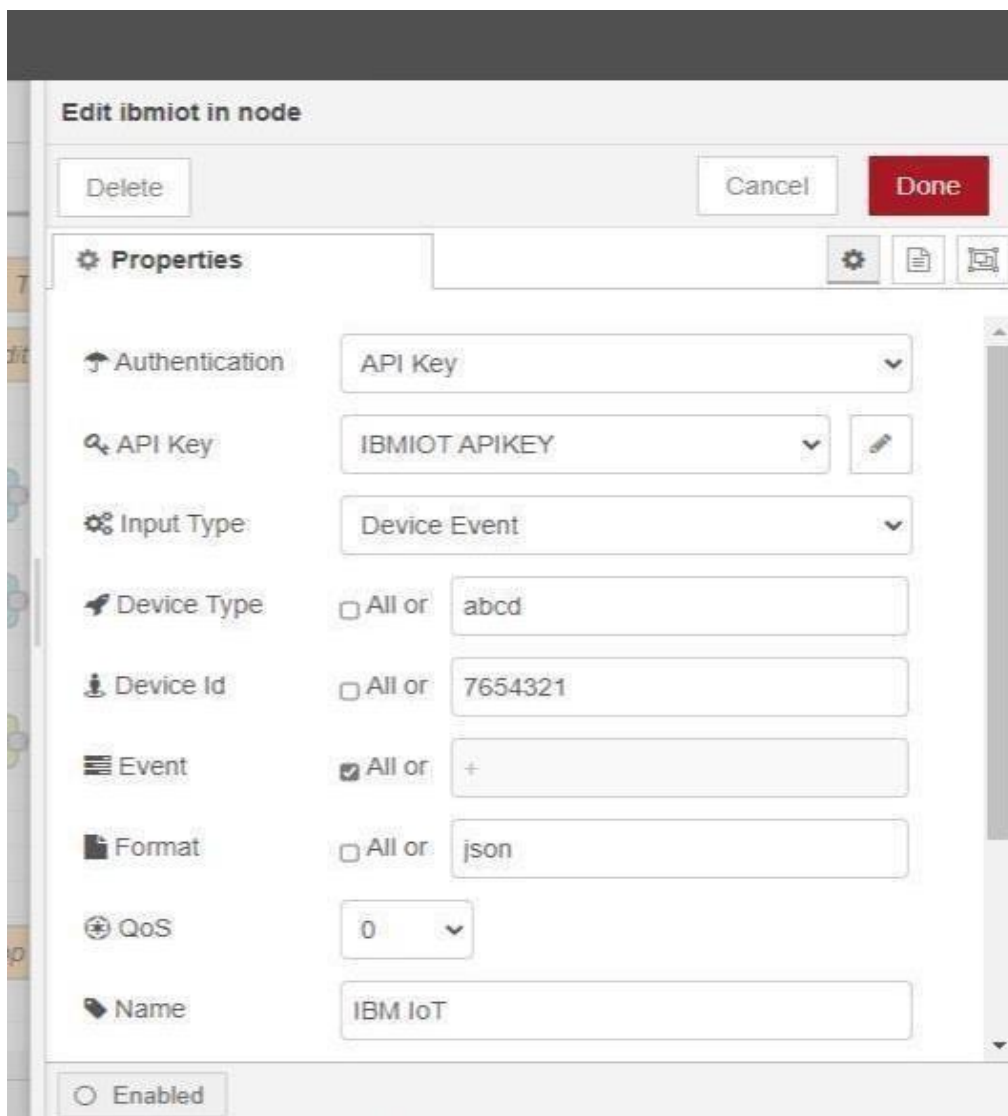
## **SPRINT DELIVERY – 3**

|                     |  |
|---------------------|--|
| <b>Team ID</b>      | PNT2022TMID34341                         |
| <b>Project Name</b> | IoT Enabled Smart<br>Farming Application |
| <b>Date</b>         | 15 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' section with a gear icon and a search 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 and a pencil icon.
- 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 radio button labeled 'Enabled'.

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"};
```

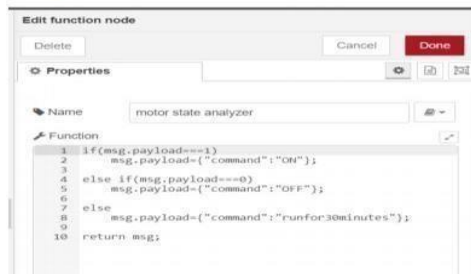
```
else if(msg.payload===0)
```

```
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.

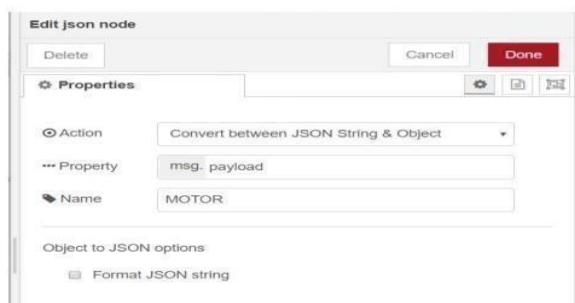
The Java script code for that function node is:

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

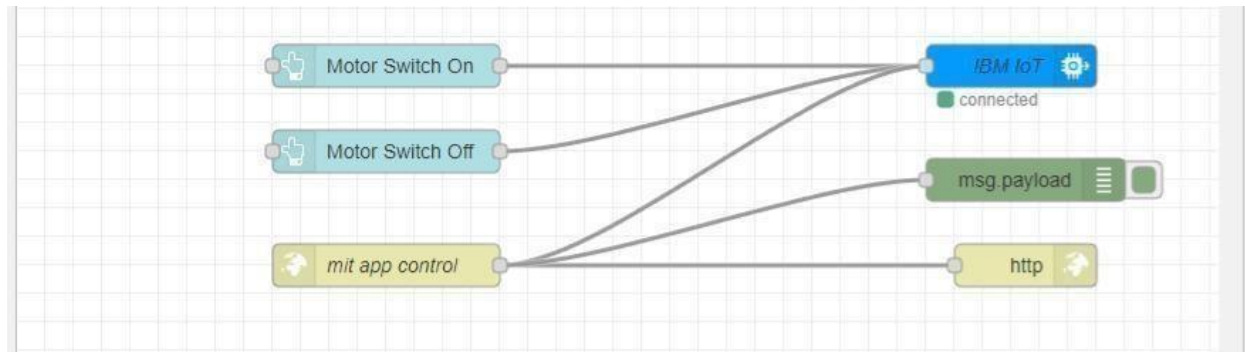


The above images show the java script codes of analyser and state function nodes.

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



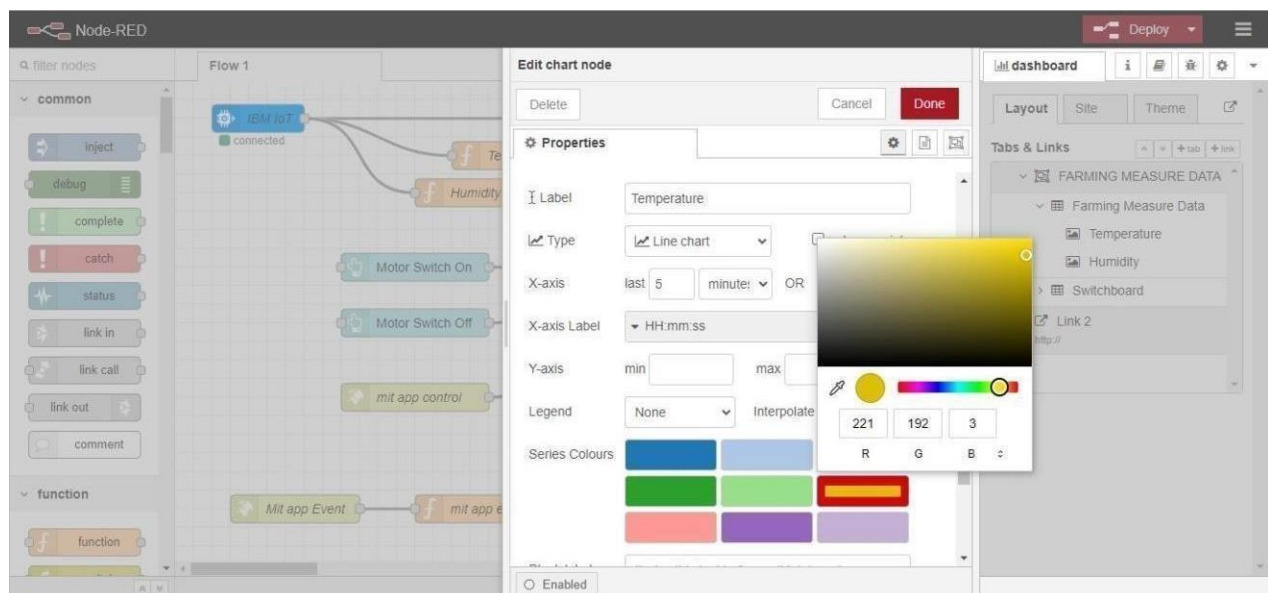
This is the program flow for sending commands to IBM cloud.

## 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.

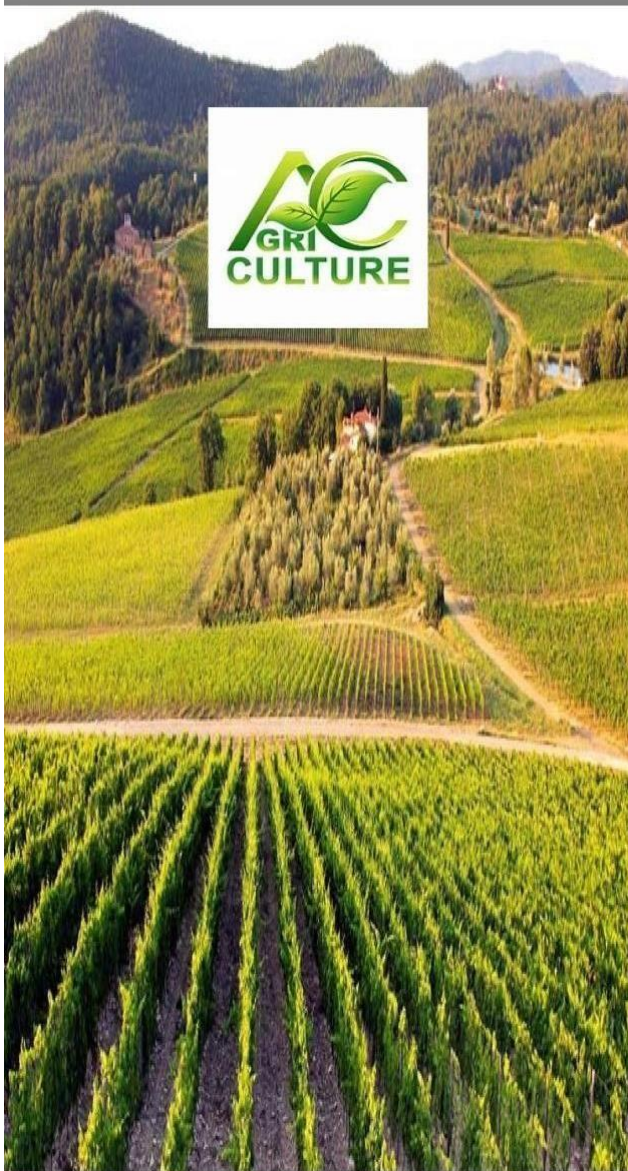
Below images are the Gauge, text and button node configurations.



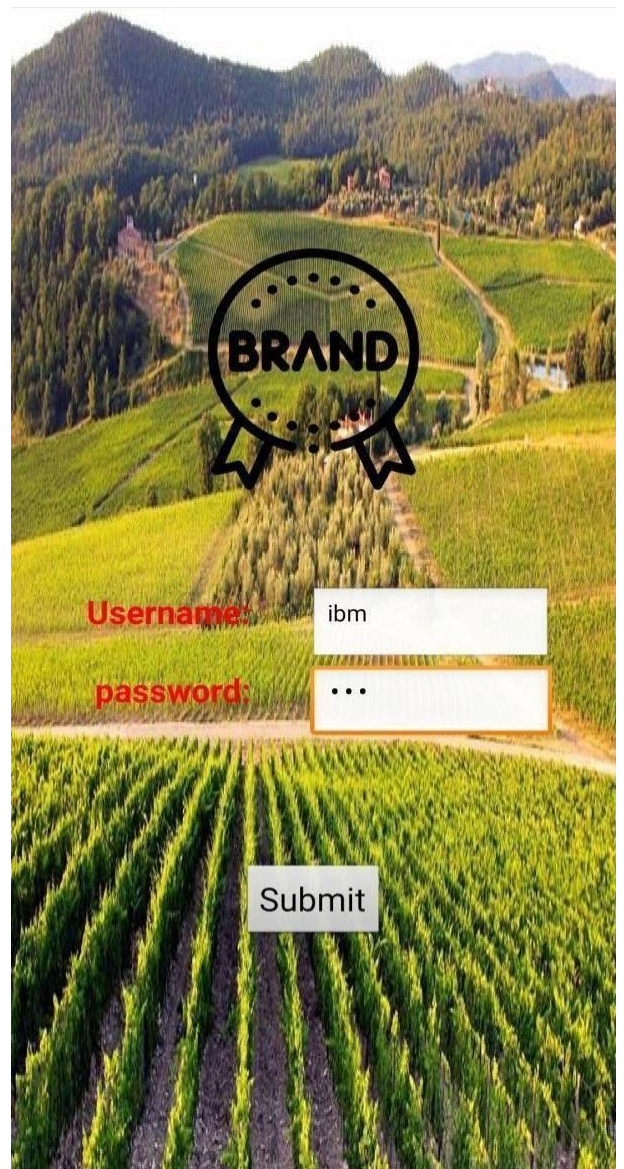
The screenshot shows the Node-RED web interface. On the left, the 'filter nodes' sidebar lists various nodes like slider, numeric, text input, date picker, colour picker, form, text, gauge, chart, audio out, notification, ui control, and template. The main workspace displays a flow diagram titled 'Flow 1'. The flow starts with an 'IBM IoT' sensor node (green) connected to three function nodes: 'Temperature', 'Humidity', and 'Moisture' (orange). These function nodes are connected to a 'msg payload' node (green). The 'msg payload' node is connected to a 'Motor Switch On' node (blue), a 'Motor Switch Off' node (blue), and another 'msg payload' node (green). The 'Motor Switch On' and 'Motor Switch Off' nodes are connected to a 'mit app control' node (yellow). The 'mit app control' node is connected to a 'http' node (yellow). The 'http' node is connected to a 'Mit app Event' node (yellow), which is connected to a 'mit app event' node (orange), which is connected to another 'http' node (yellow). The right sidebar shows the 'debug' console with logs for the 'msg payload' and 'http' nodes. The logs show the 'msg payload' node outputting an object with 'temp', 'Humid', and 'Mois' properties, and the 'http' node outputting a 'command' property with values 'motoron' and 'motoroff'.



Screen1

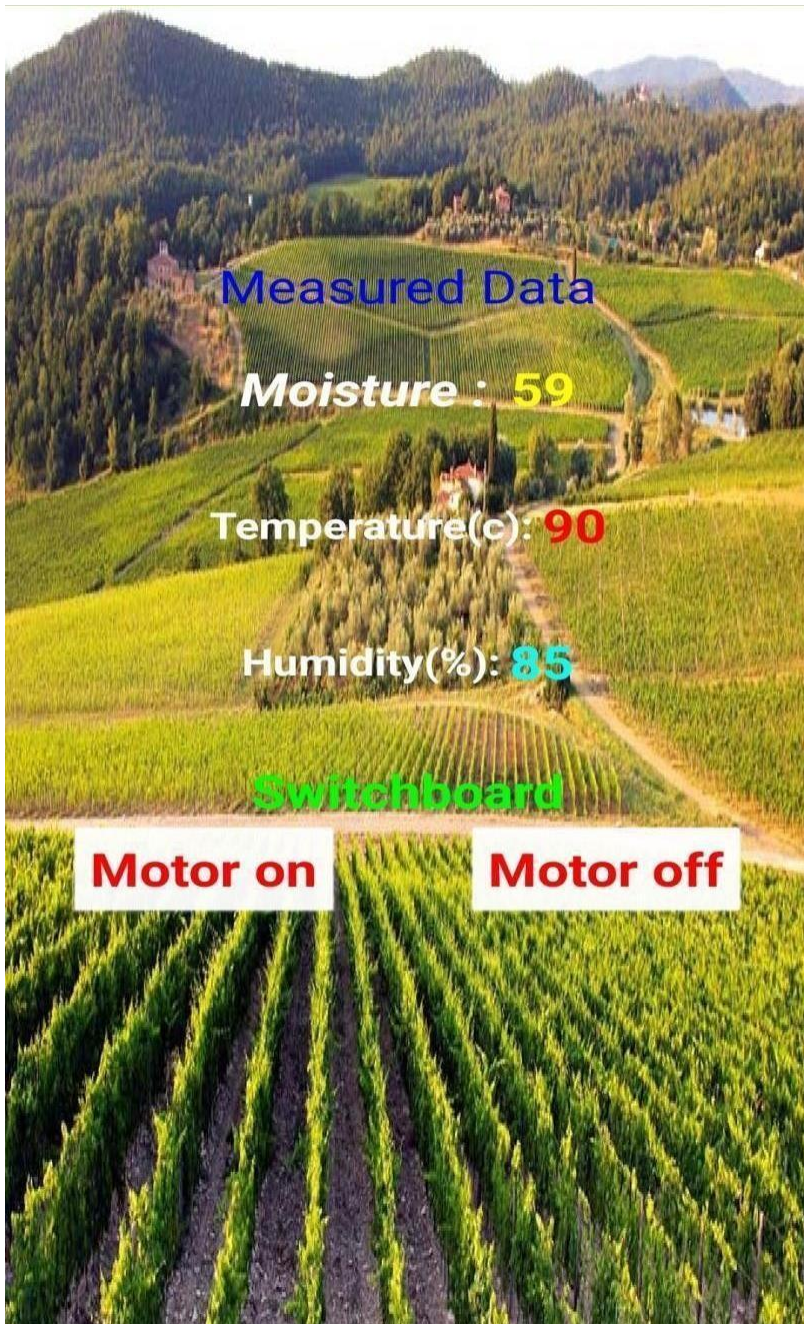


SCREEN – 1



SCREEN - 2





**SCREEN - 3**

## Web APP UI Home Tab

