

Signs with Smart Connectivity for Better Road Safety

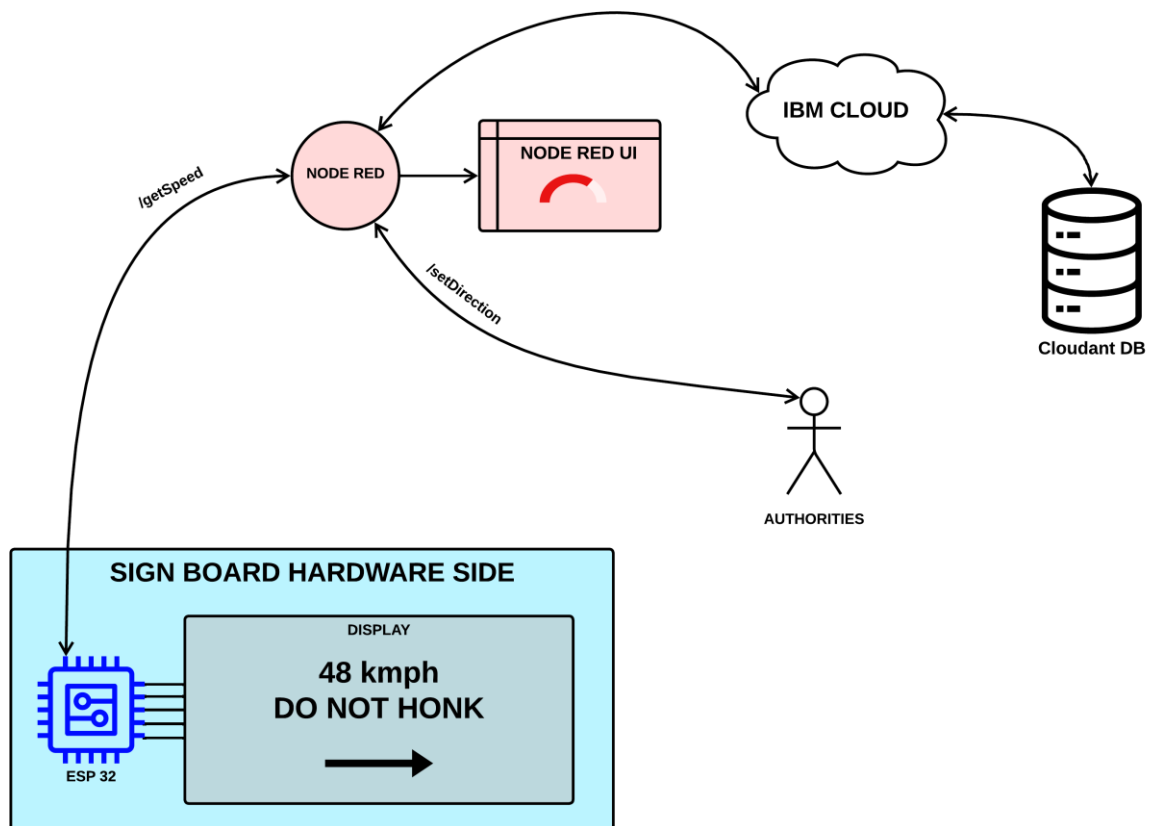
Sprint 03

Team ID - PNT2022TMID31848

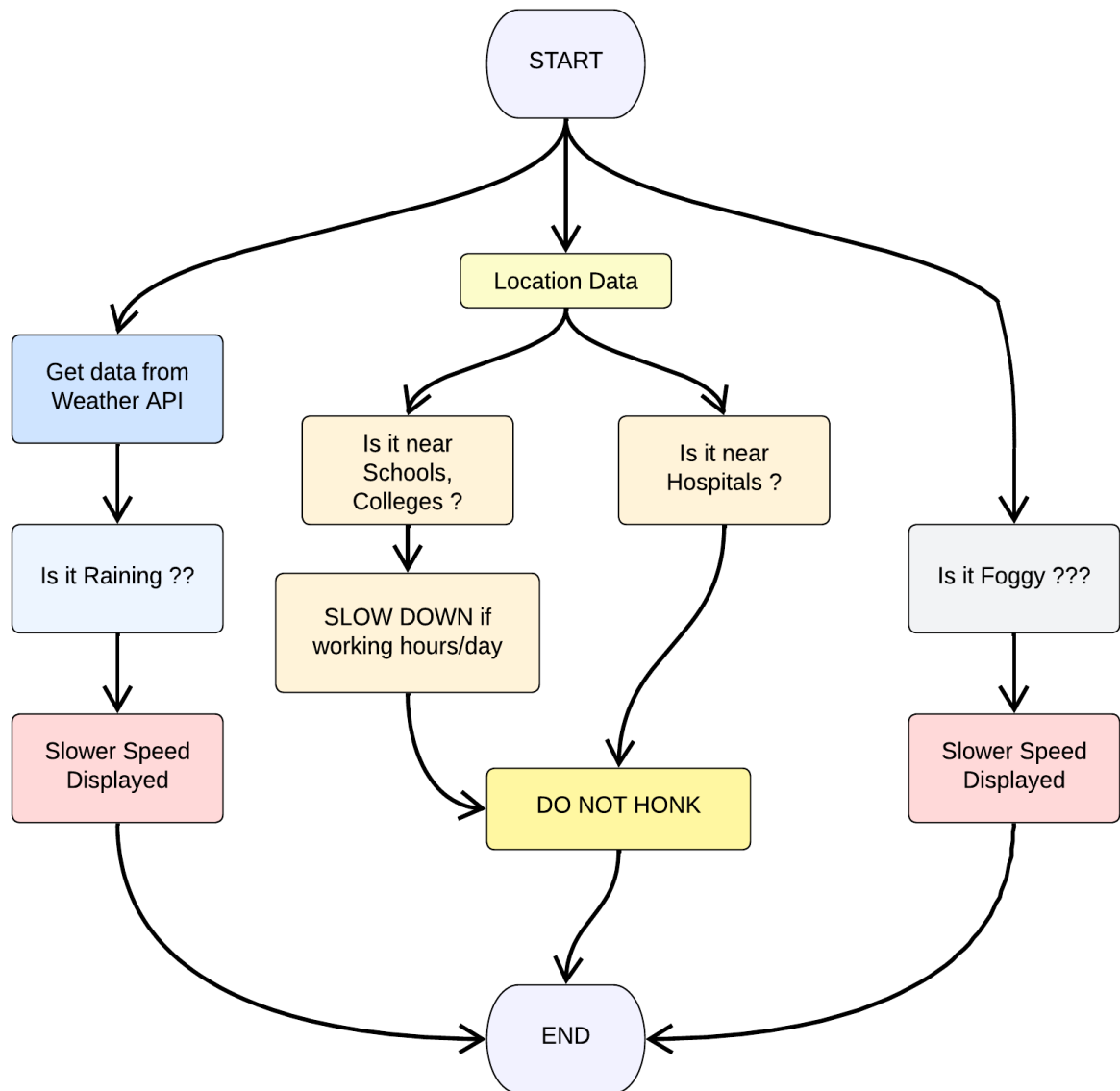
Sprint Goals :

Hardware & Cloud integration

Process Flow :

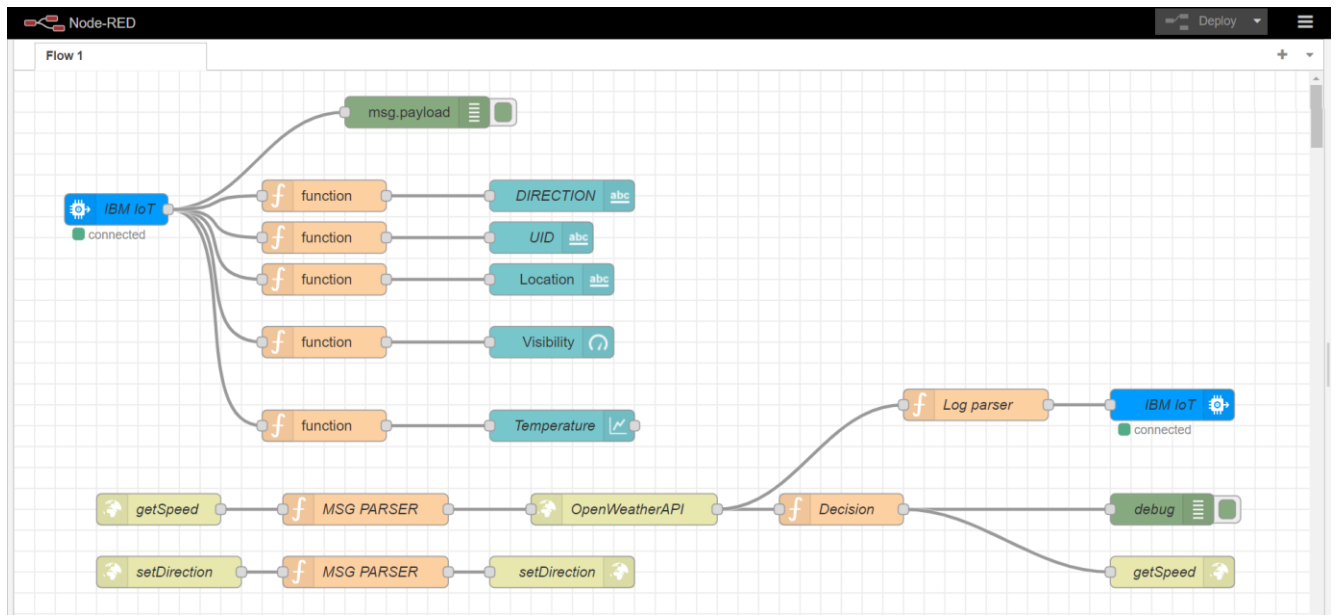


Code Flow :



Node RED :

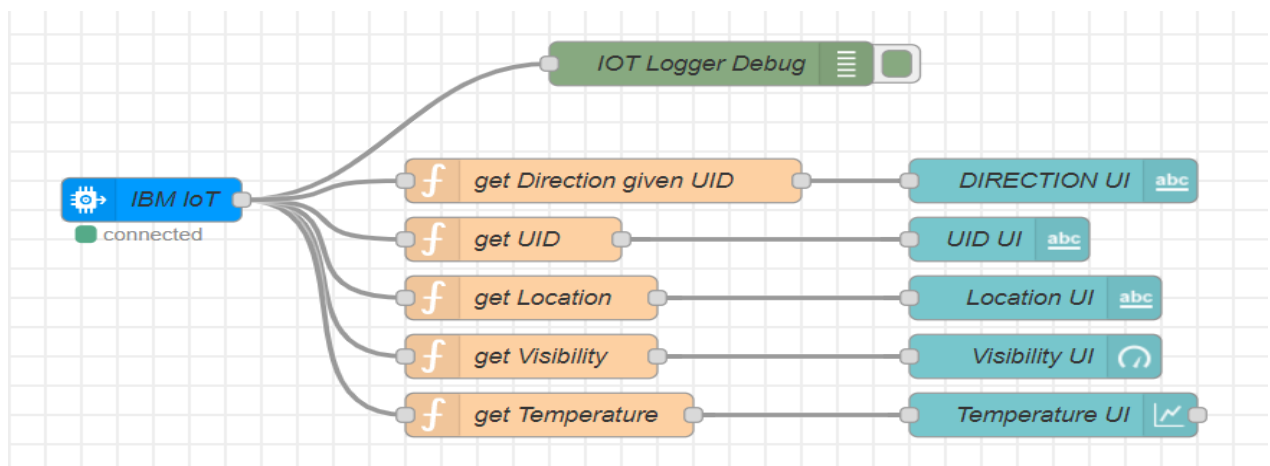
Node RED flow :



There are 3 flows in the above Node RED flow. They are

1. Node RED UI flow
2. /getSpeed API flow
3. /setDirection API flow

1. Node RED UI flow :



```
// get Direction given UID
msg.payload = global.get(String(msg.payload.uid));
return msg;

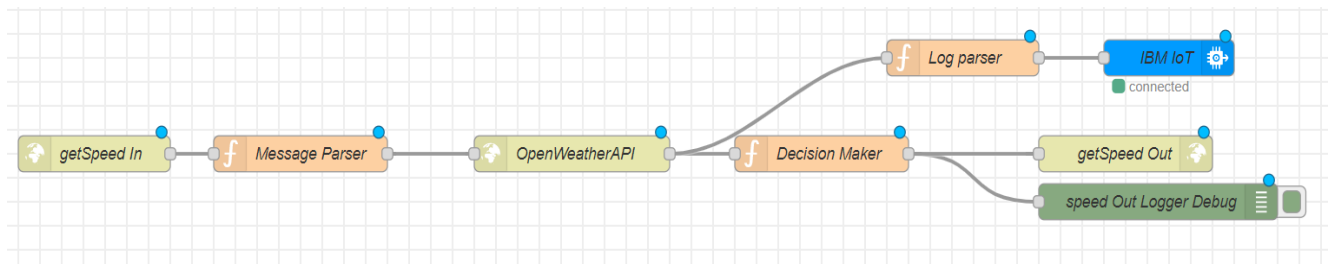
// get UID
msg.payload = msg.payload.uid;
return msg;

// get Location
msg.payload = msg.payload.location;
return msg;

// get Visibility
msg.payload = msg.payload.visibility;
return msg;

// get Temperature
msg.payload = msg.payload.temperature;
return msg;
```

2. /getSpeed API flow :



```
global.set("data",msg.payload);
```

```
msg.payload.q = msg.payload.location;
```

```
msg.payload.appid = "bf4a8d480ee05c00952bf65b78ae826b";
```

```
return msg;
```

"OpenWeatherAPI" node is a http request node which calls the OpenWeather API and send the data to the next node.

"Log Parser" node extracts specific parameters from the weather data and and sends it to the next node.

```
weatherObj = JSON.parse(JSON.stringify(msg.payload));
```

```
localityObj = global.get("data");
```

```
var suggestedSpeedPercentage = 100;
```

```
var preciseObject = {
```

```
    temperature : weatherObj.main.temp - 273.15,
```

```
    location : localityObj.location,
```

```
    visibility : weatherObj.visibility/100,
```

```
    uid : localityObj.uid,
```

```
    direction : global.get("direction")
```

```
};
```

```
msg.payload = preciseObject;
```

```
return msg;
```

"IBM IoT" node here (IBM IoT OUT) connects the "IBM IoT" node (IBM IoT IN) mentioned in the Node RED UI flow which enables UI updation and logging.

"Decision Maker" node processes the weather data and other information from the micro controller to form the string that is to be displayed at the Sign Board

Code:

```
weatherObj = JSON.parse(JSON.stringify(msg.payload));
localityObj = global.get("data");
var suggestedSpeedPercentage = 100;
var preciseObject = {
  temperature : weatherObj.main.temp - 273.15,
  weather : weatherObj.weather.map(x=>x.id).filter(code =>
code<700),
  visibility : weatherObj.visibility/100
};
if(preciseObject.visibility<=40)
  suggestedSpeedPercentage -=30
  switch(String(preciseObject.weather)[-1]) //
https://openweathermap.org/weather-conditions refer weather codes
meaning here
{
  case "0" : suggestedSpeedPercentage -=10;break;
  case "1" : suggestedSpeedPercentage -=20;break;
```

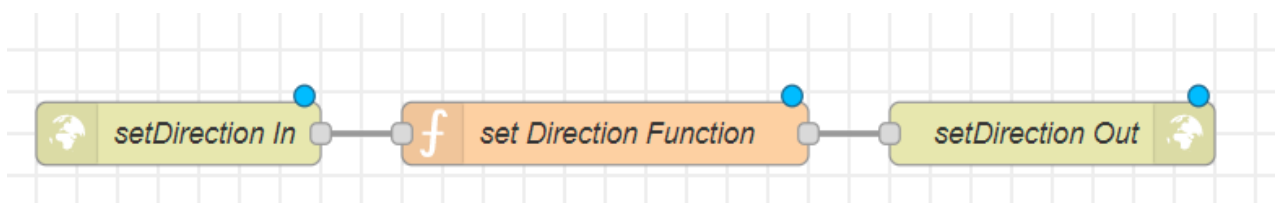
```

    case "2" : suggestedSpeedPercentage -=30;break;
}
msg.payload = preciseObject;
var doNotHonk = 0;
if(localityObj.hospitalZone=="1"||localityObj.schoolZone=="1")
    doNotHonk = 1;
var returnObject = {
    suggestedSpeed :
    localityObj.usualSpeedLimit*(suggestedSpeedPercentage/100),
    doNotHonk : doNotHonk
}
msg.payload = String(returnObject.suggestedSpeed) + " kmph \n\n" +
(returnObject.doNotHonk==1?"Do Not Honk:") + "$" +
global.get(String(localityObj.uid));

return msg;

```

3. /setDirection API flow :



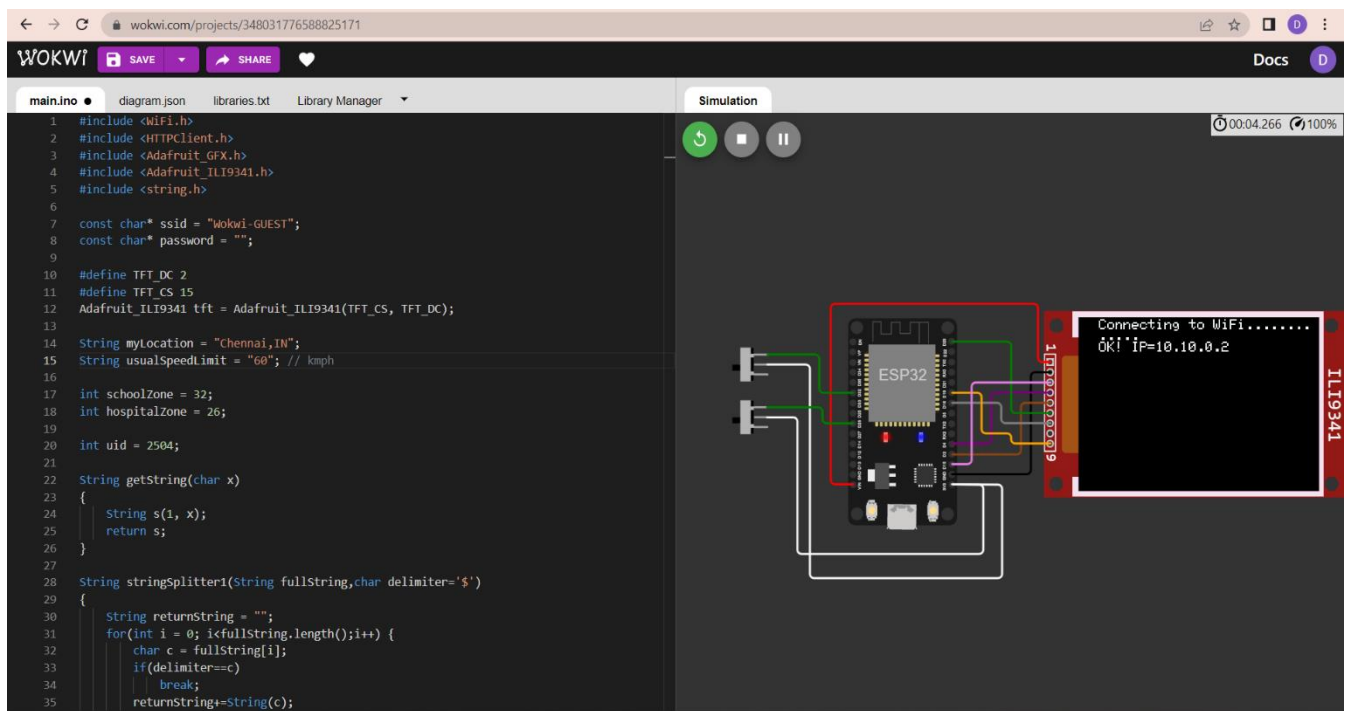
- 1."setDirection In" node is an http end point. It accepts parameters like microcontroller UID & direction.
- 2."set Direction Function" node sets the direction for the given UID.

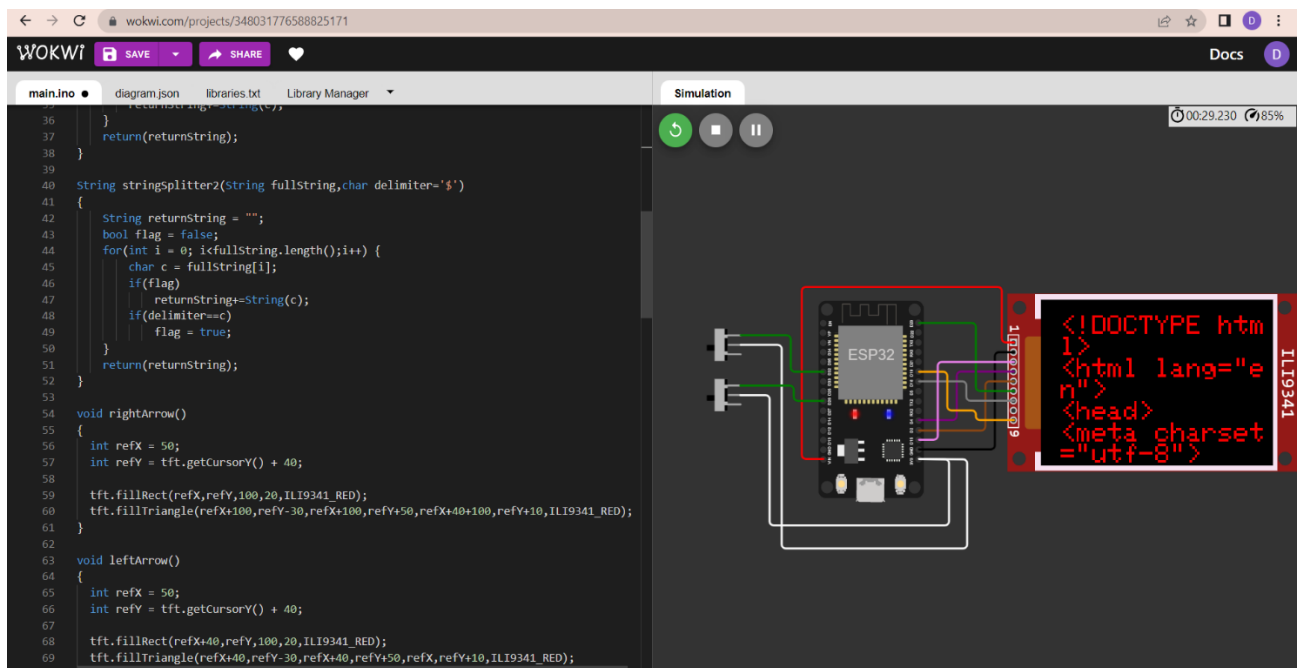
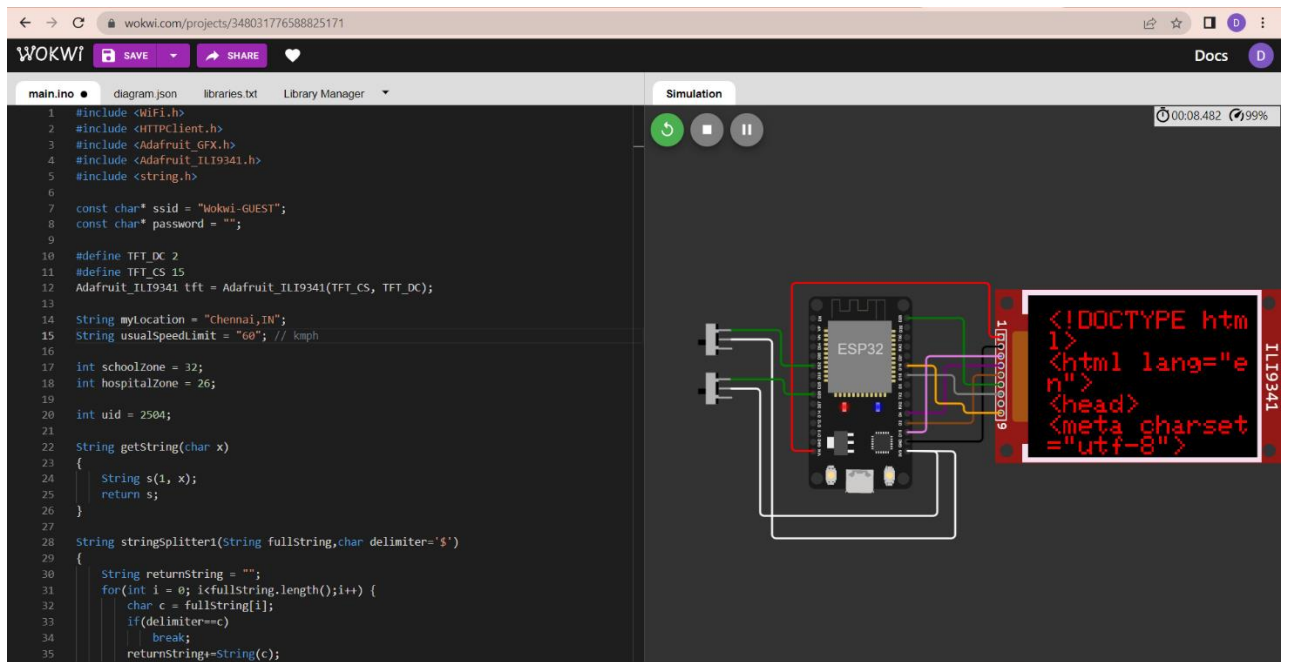
```
global.set(String(msg.payload.uid),msg.payload.dir);
```

```
return msg;
```

3."setDirection Out" node returns a http response for the request at node "setDirection In".

Wokwi Circuit :





WOKWI SAVE SHARE Docs

main.ino diagram.json libraries.txt Library Manager

```

68 tft.fillRect(refX+40,refY,100,20,ILI9341_RED);
69 tft.fillTriangle(refX+40,refY-30,refX+40,refY+50,refX+10,ILI9341_RED);
70 }
71
72 void upArrow()
73 {
74   int refX = 125;
75   int refY = tft.getCursorY() + 30;
76
77   tft.fillTriangle(refX-40,refY+40,refX+40,refY+40,refX,refY,ILI9341_RED);
78   tft.fillRect(refX-15,refY+40,30,20,ILI9341_RED);
79 }
80
81 String APICall() {
82   HTTPClient http;
83
84   String url = "https://node-red-grseb-2022-11-05-test.eu-gb.mybluemix.net/getSpeed?";
85   url += "location="+myLocation+"&";
86   url += "schoolZone="+String(digitalRead(schoolZone))+"&";
87   url += "hospitalZone="+String(digitalRead(hospitalZone))+"&";
88   url += "usualSpeedLimit="+String(usualSpeedLimit)+"&";
89   url += "uid="+String(uid);
90   http.begin(url.c_str());
91   int httpStatusCode = http.GET();
92
93   if (httpStatusCode>0) {
94     String payload = http.getString();
95     http.end();
96     return(payload);
97   }
98   else {
99     Serial.print("Error code: ");
100    Serial.println(httpStatusCode);
101  }
102  http.end();

```

Simulation 00:41.495 69%

WOKWI SAVE SHARE Docs

main.ino diagram.json libraries.txt Library Manager

```

122 if(c2=="r") // represents right
123 {
124   rightArrow();
125 }
126
127
128 void setup() {
129   WiFi.begin(ssid, password, 6);
130
131   tft.begin();
132   tft.setRotation(1);
133
134   tft.setTextColor(ILI9341_WHITE);
135   tft.setTextSize(2);
136   tft.print("Connecting to WiFi");
137
138   while (WiFi.status() != WL_CONNECTED) {
139     delay(100);
140     tft.print(".");
141   }
142
143   tft.print("\nOK! IP=");
144   tft.println(WiFi.localIP());
145 }
146
147 void loop() {
148
149   myPrint(APICall());
150
151   delay(100);
152 }

```

Simulation 01:34.642 98%

abort() was called at PC 0x4014d7da on core 1

Backtrace: 0x40083632:0x3ffb2550:0x4008c5f9:0x3ffb2570:0x400917b1:0x3ffb2590
0x4014d7da:0x3ffb2610:0x400d2bcd:0x3ffb26b0:0x400d2e45:0x3ffb27e0
0x400db359:0x3ffb2820