Signs with Smart Connectivity for Better Road Safety

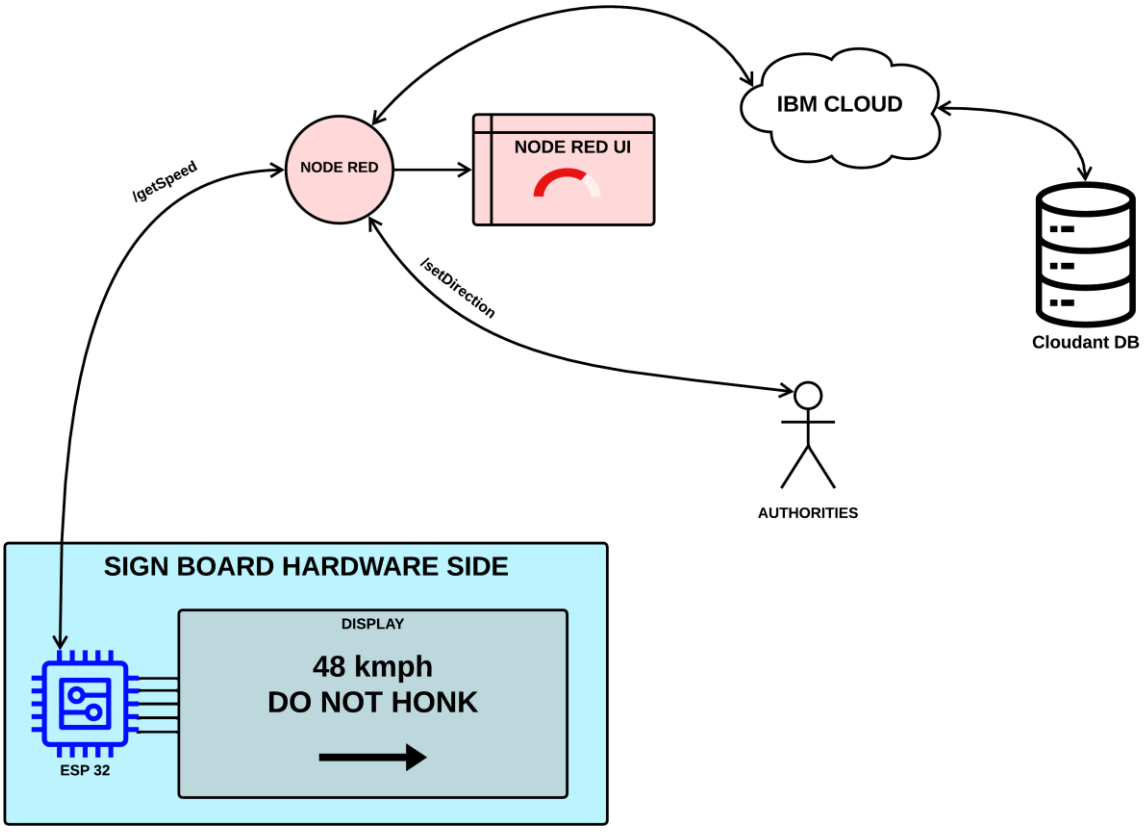
Sprint 04

Team ID - PNT2022TMID31848

Sprint Goals :

Hardware & Cloud integration

Process Flow :

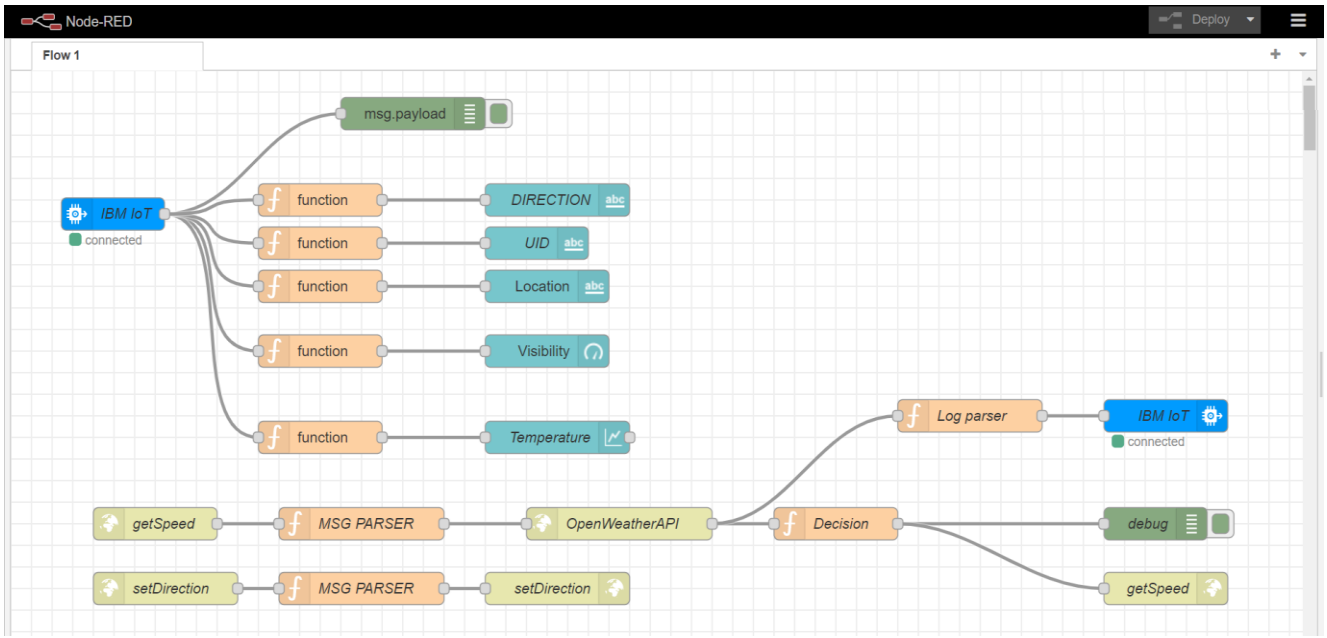


Code Flow :


START
Location Data
Get data from Weather API
Is it near Schools, Colleges ?
Is it near Hospitals ?
Is it Raining ??
SLOW DOWN if working hours/day
Is it Foggy ???
Slower Speed Displayed
DO NOT HONK
Slower Speed Displayed
END

Node RED :

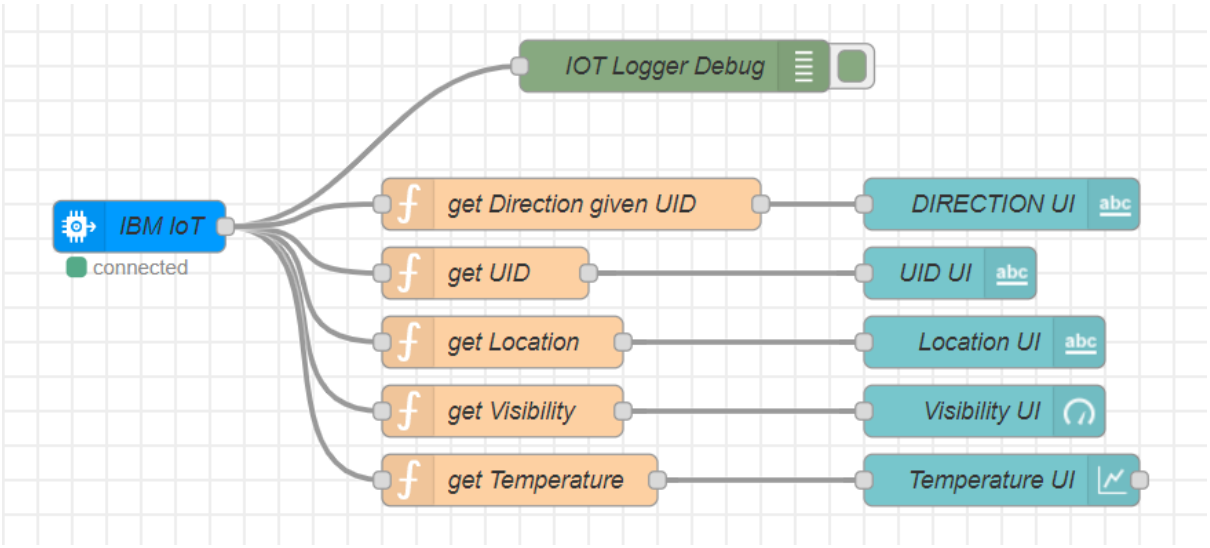
Node RED flow :



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 :

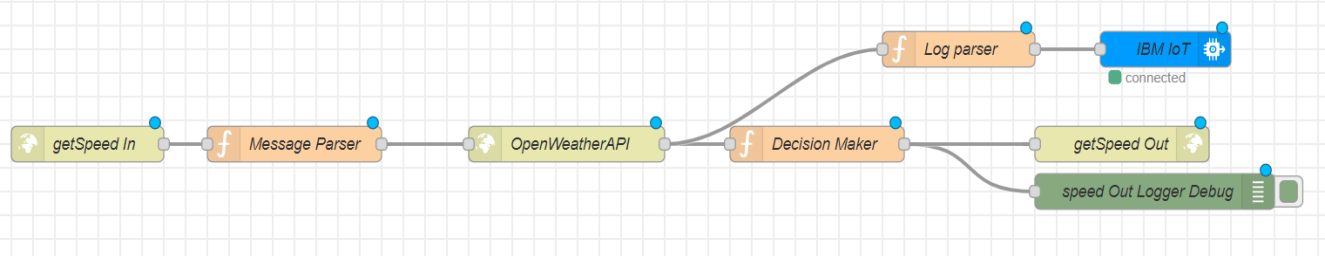


Code:

```
// 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 :



1. "getSpeed In" node is an http end point. It accepts parameters like microcontroller UID, location, school & hospital zones info.

2. "Message Parser" node parses the data and passes on only required information to the next node

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

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

msg.payload.appid = "bf4a8d480ee05c00952bf65b78ae826b";

return msg;
```

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

4. "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;
```

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

6. "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

```
weatherObj = JSON.parse(JSON.stringify(msg.payload));
localityObj = global.get("data");
var suggestedSpeedPercentage = 100;
var preciseObject = {
```

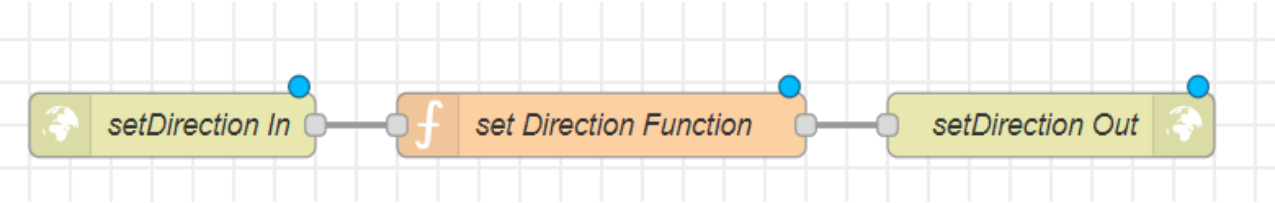
```
   temperature : weatherObj.main.temp - 273.15,
   weather : weatherObj.weather.map(x=>x.id).filter(code =>
code<700),
   visibility : weatherObj.visibility/100
};

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;
```

7. "getSpeed Out" node returns a http response for the request at node "getSpeed In".

8. "speed Out Logger Debug" logs the data for debugging.

3./setDirection API flow :



1. "setDirection In" node is an http end point. It accepts parameters like microcontroller UID & direction.

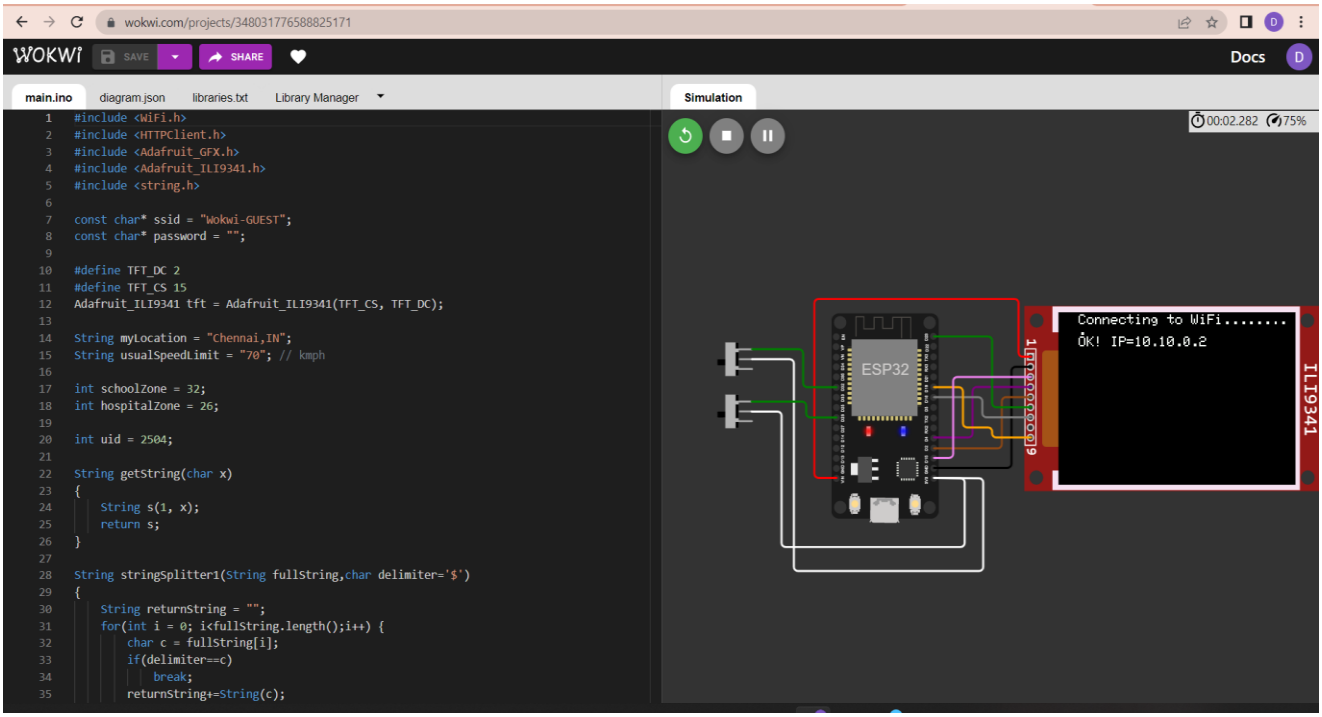
2. "set Direction Function" node sets the direction for the given UID.

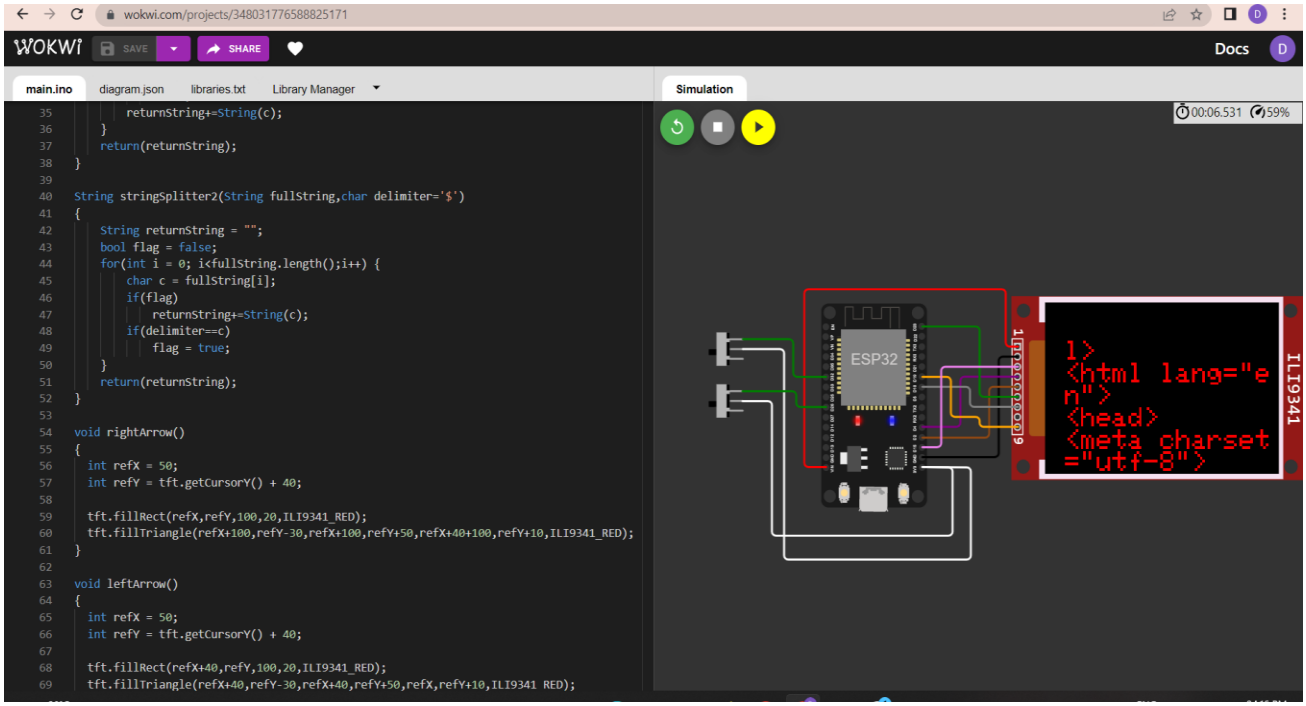
global.set(String(msg.payload.uid),msg.payload.dir);

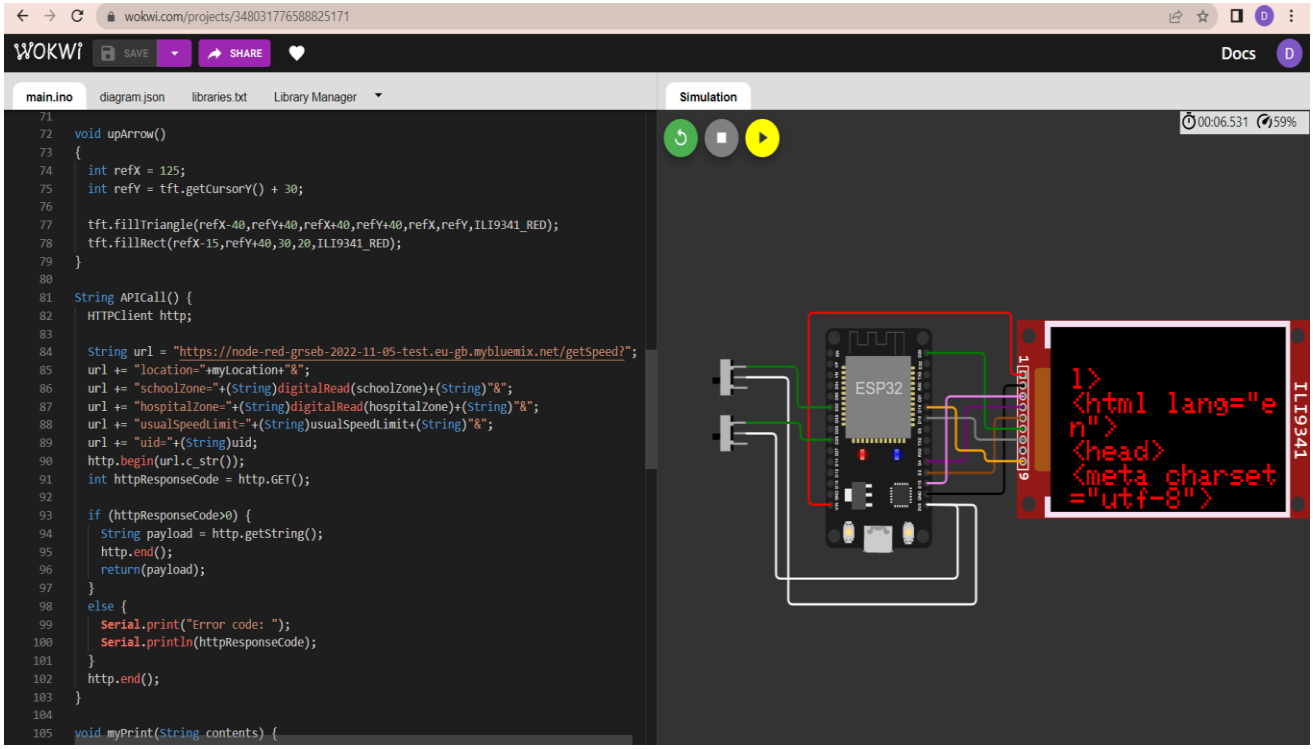
return msg;

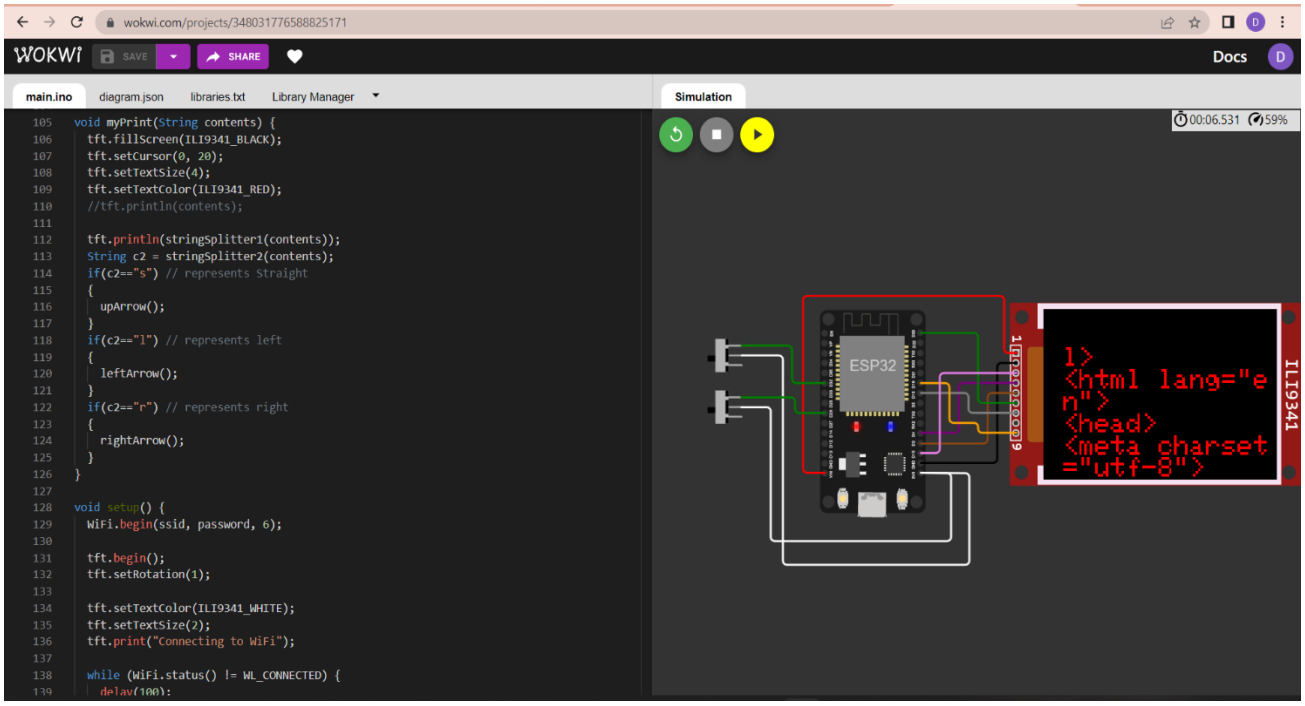
3."setDirection Out" node returns a http response for the request at node "setDirection In".

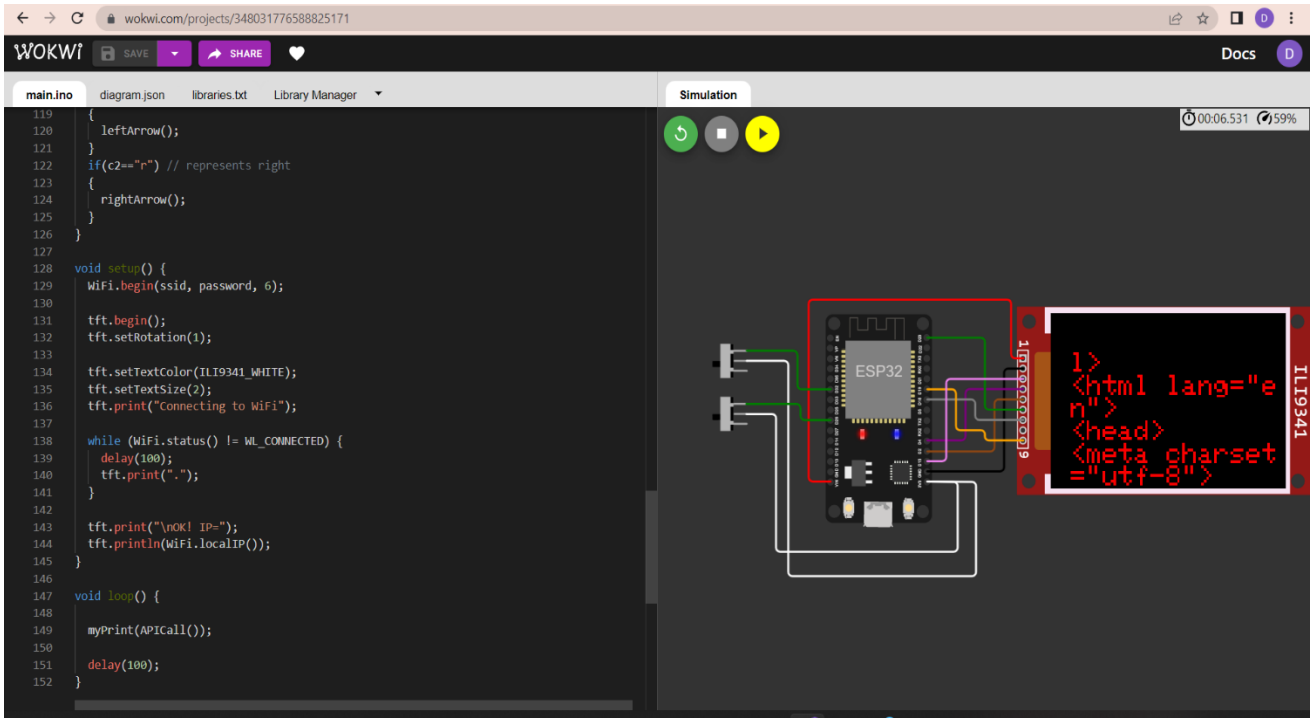
Wokwi Circuit :




wokwi.com/projects/348031776588825171
WOKWI
SAVE
SHARE
Docs
main.ino
diagram.json
libraries.txt
Library Manager
Simulation
00:06.531
59%
returnString+=String(c);
}
return(returnString);
}
String stringSplitter2(String fullString,char delimiter='$')
{
String returnString = "";
bool flag = false;
for(int i = 0; i<fullString.length();i++) {
char c = fullString[i];
if(flag)
returnString+=String(c);
if(delimiter==c)
flag = true;
}
return(returnString);
}
void rightArrow()
{
int refX = 50;
int refY = tft.getCursorY() + 40;
tft.fillRect(refX,refY,100,20,ILI9341_RED);
tft.fillTriangle(refX+100,refY-30,refX+100,refY+50,refX+40+100,refY+10,ILI9341_RED);
}
void leftArrow()
{
int refX = 50;
int refY = tft.getCursorY() + 40;
tft.fillRect(refX+40,refY,100,20,ILI9341_RED);
tft.fillTriangle(refX+40,refY-30,refX+40,refY+50,refX,refY+10,ILI9341 RED);
ESP32
ILI9341
1>
<html lang="e
n">
<head>
<meta charset
="utf-8">
ILI9341


wokwi.com/projects/348031776588825171
WOKWI
SAVE
SHARE
Docs
main.ino
diagram.json
libraries.txt
Library Manager
Simulation
00:06.531
59%
void upArrow()
{
int refX = 125;
int refY = tft.getCursorY() + 30;
tft.fillTriangle(refX-40,refY+40,refX+40,refY+40,refX,refY,ILI9341_RED);
tft.fillRect(refX-15,refY+40,30,20,ILI9341_RED);
}
String APICall() {
HTTPClient http;
String url = "https://node-red-grseb-2022-11-05-test.eu-gb.mybluemix.net/getSpeed?";
url += "location="+myLocation+"&";
url += "schoolZone="+(String)digitalRead(schoolZone)+(String)"&";
url += "hospitalZone="+(String)digitalRead(hospitalZone)+(String)"&";
url += "usualSpeedLimit="+(String)usualSpeedLimit+(String)"&";
url += "uid="+(String)uid;
http.begin(url.c_str());
int httpResponseCode = http.GET();
if (httpResponseCode>0) {
String payload = http.getString();
http.end();
return(payload);
}
else {
Serial.print("Error code: ");
Serial.println(httpResponseCode);
}
http.end();
}
void myPrint(String contents) {
ESP32
ILI9341
1>
<html lang="e
n">
<head>
<meta charset
="utf-8">
ILI9341


wokwi.com/projects/348031776588825171
WOKWI
SAVE
SHARE
Docs
main.ino
diagram.json
libraries.txt
Library Manager
Simulation
00:06.531
59%
ESP32
ILI9341
1)
<html lang="e
n">
<head>
<meta charset
="utf-8">
105 void myPrint(String contents) {
106 tft.fillScreen(ILI9341_BLACK);
107 tft.setCursor(0, 20);
108 tft.setTextSize(4);
109 tft.setTextColor(ILI9341_RED);
110 //tft.println(contents);
111
112 tft.println(stringSplitter1(contents));
113 String c2 = stringSplitter2(contents);
114 if(c2=="s") // represents Straight
115 {
116 upArrow();
117 }
118 if(c2=="l") // represents left
119 {
120 leftArrow();
121 }
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) {


wokwi.com/projects/348031776588825171
WOKWI
SAVE
SHARE
Docs
main.ino
diagram.json
libraries.txt
Library Manager
Simulation
00:06.531
59%
ESP32
ILI9341
1)
<html lang="e
n">
<head>
<meta charset
="utf-8">
119 {
120 leftArrow();
121 }
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 }