Sprint 03

Signs with Smart Connectivity for Better Road Safety

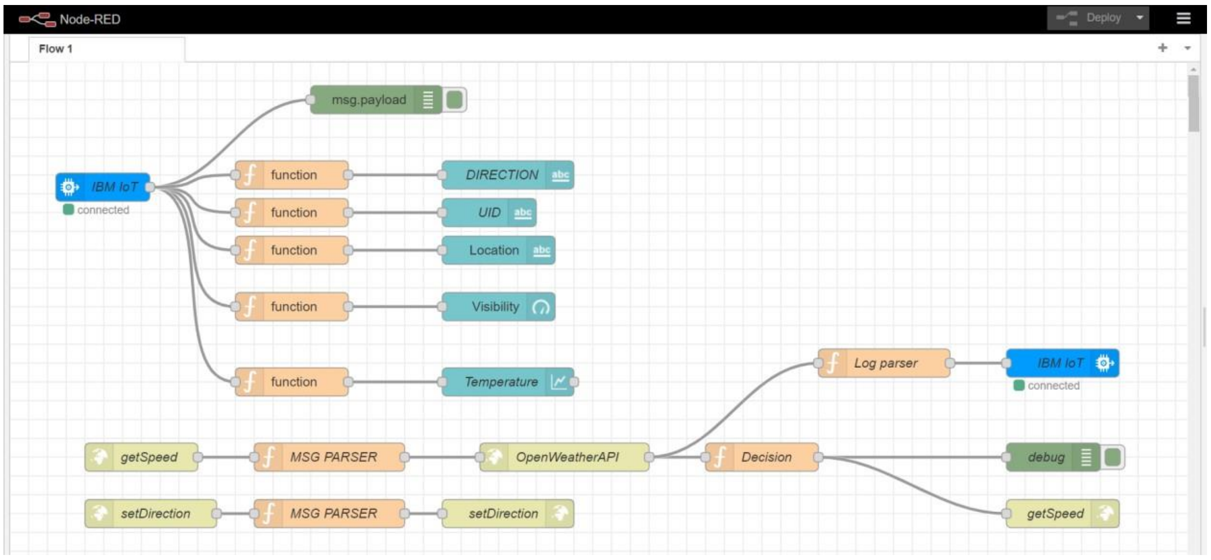
Team ID - PNT2022TMID50268

Sprint Goals :

Hardware & Cloud integration

Node RED :

Node RED flow :



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

Node RED UI flow

/getSpeed API flow

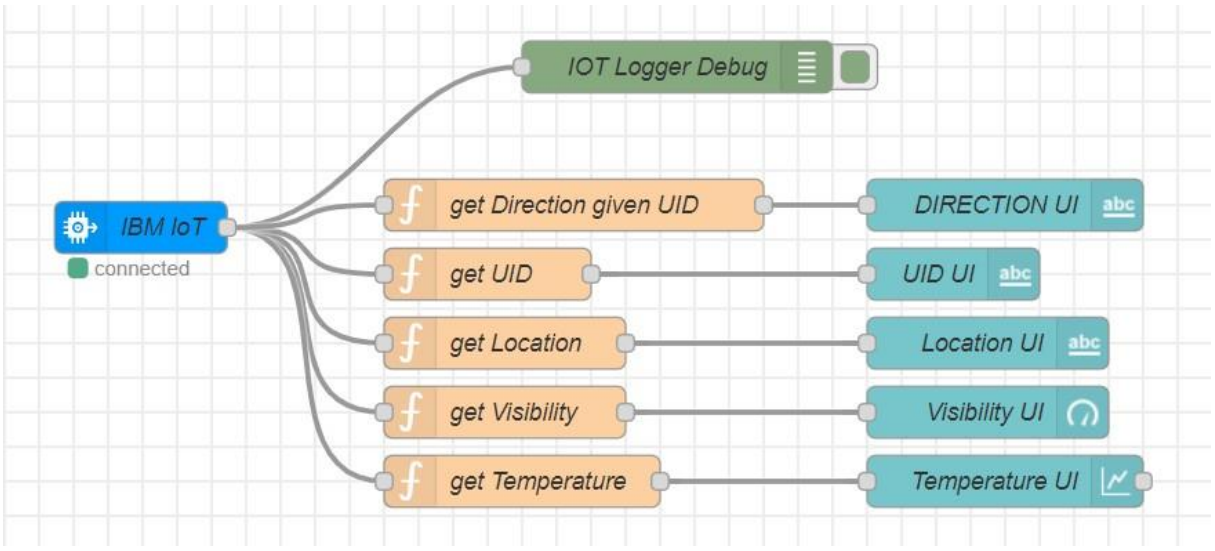
/setDirection API flow

1. Node RED UI flow :

UI FLOW

"IBM IOT" node connects the backend to Node RED UI

The function nodes such as "get Direction given UID", "get UID", "get Location", "get Visibility" & "get Temperature" extract the respective data out and provides them to the UI nodes "Direction UI", "UID UI", "Location UI", "Visibility UI" & "Temperature UI".



// 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;

"IOT Logger Debug" node logs the data at debugger.

2. /getSpeed API flow :

getSpeedAPI

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

"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;
```

"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) metioned 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

```
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;
```

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

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

3. /setDirection API flow :

UI FLOW

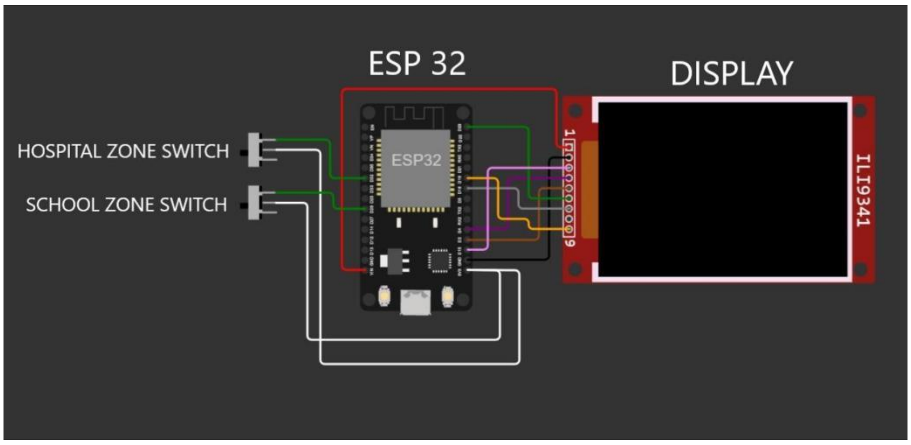
"setDirection In" node is an http end point. It accepts parameters like microcontroller UID & direction.

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

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

return msg;
```

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

Circuit Diagram:



ESP 32 CODE:

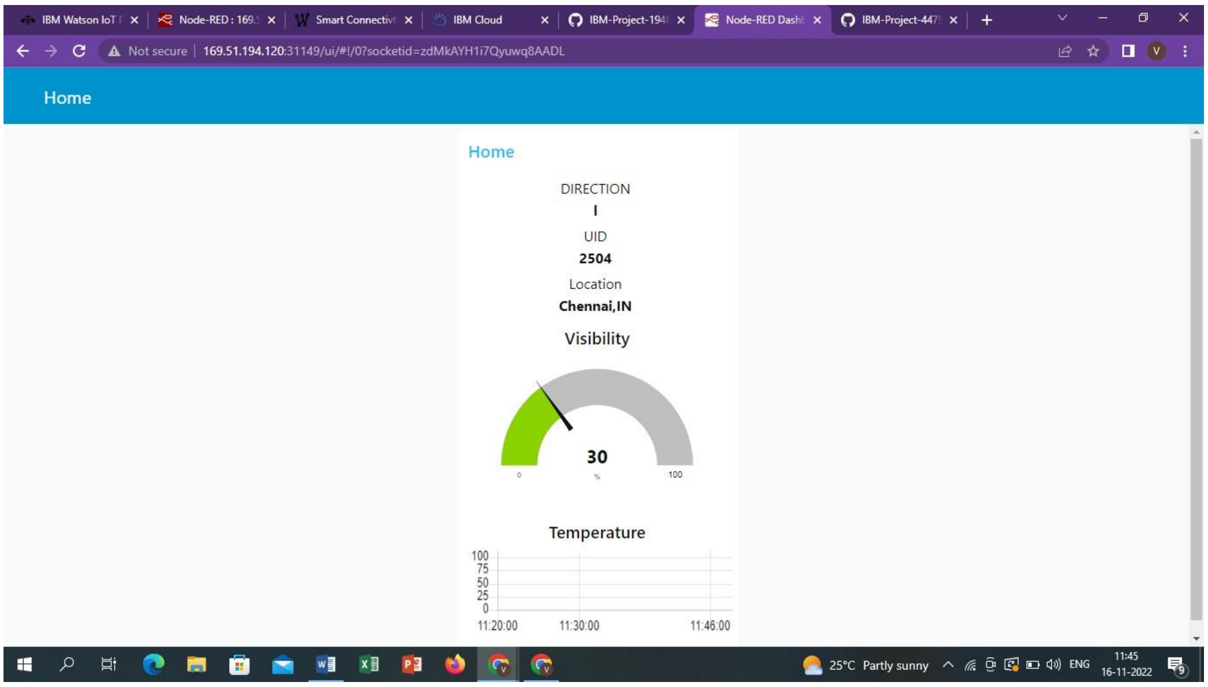
```
#include  <WiFi.h>

#include <HTTPClient.h>

#include <Adafruit_GFX.h>

#include <Adafruit_ILI9341.h>

#include <string.h>


const char* ssid = "Wokwi-GUEST";

const char* password = "";


#define TFT_DC 2

#define TFT_CS 15

Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);


String myLocation = "Chennai,IN";

String usualSpeedLimit = "70"; // kmph


int schoolZone = 32;

int hospitalZone = 26;


int uid = 2504; // ID Unique to this Micro Contoller


String getString(char x)

{

   String s(1, x);

   return s;

}


String stringSplitter1(String fullString,char delimiter='$')

{
   String returnString = "";
   for(int i = 0; i<fullString.length();i++) {
     char c = fullString[i];
     if(delimiter==c)
       break;
     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);
}

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:// 169.51.194.120:31149/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) {
  tft.fillScreen(ILI9341_BLACK);
  tft.setCursor(0, 20);
  tft.setTextSize(4);
  tft.setTextColor(ILI9341_RED);
  //tft.println(contents);

  tft.println(stringSplitter1(contents));
  String c2 = stringSplitter2(contents);
  if(c2=="s") // represents Straight
  {
    upArrow();
  }
  if(c2=="l") // represents left
  {
    leftArrow();
  }
  if(c2=="r") // represents right
  {
    rightArrow();
  }
}

void setup() {
  WiFi.begin(ssid, password, 6);

  tft.begin();
  tft.setRotation(1);

  tft.setTextColor(ILI9341_WHITE);
  tft.setTextSize(2);
  tft.print("Connecting to WiFi");

  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
    tft.print(".");
  }

  tft.print("\nOK! IP=");
  tft.println(WiFi.localIP());
}

void loop() {
  myPrint(APICall());
  delay(100);
}
```

Output :

Node RED Dashboard :

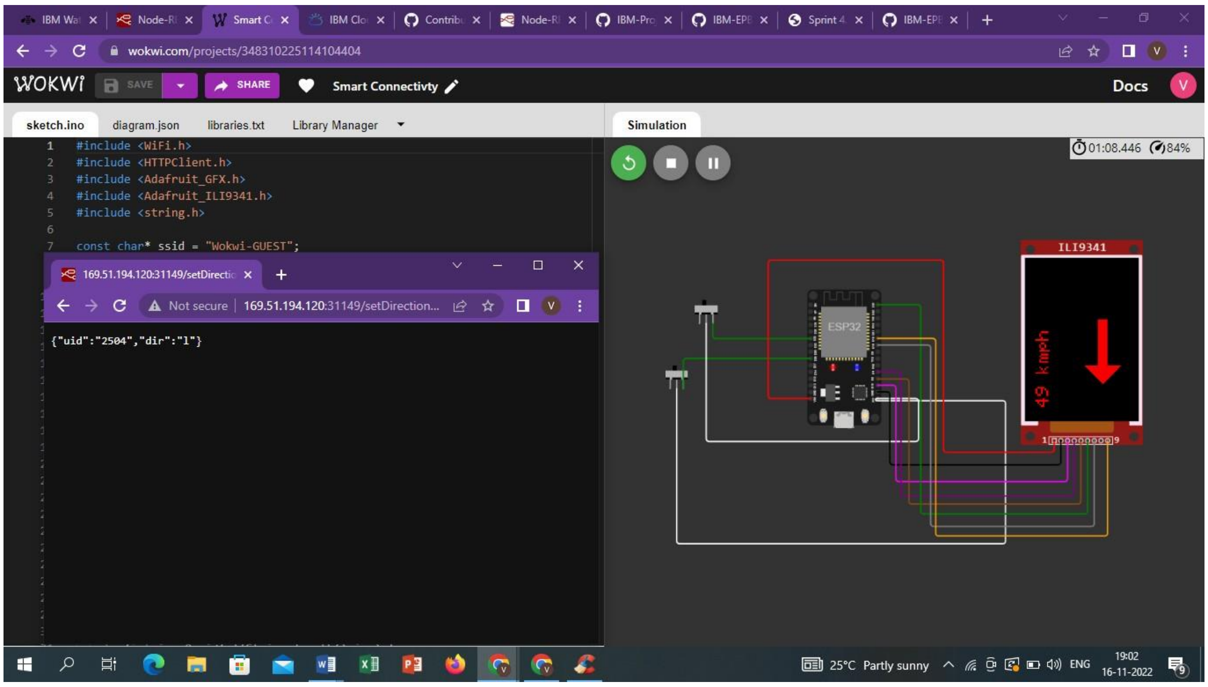
LINK TO NODE RED DASHBOARD



Home

DIRECTION
l
UID
2504
Location
Chennai,IN

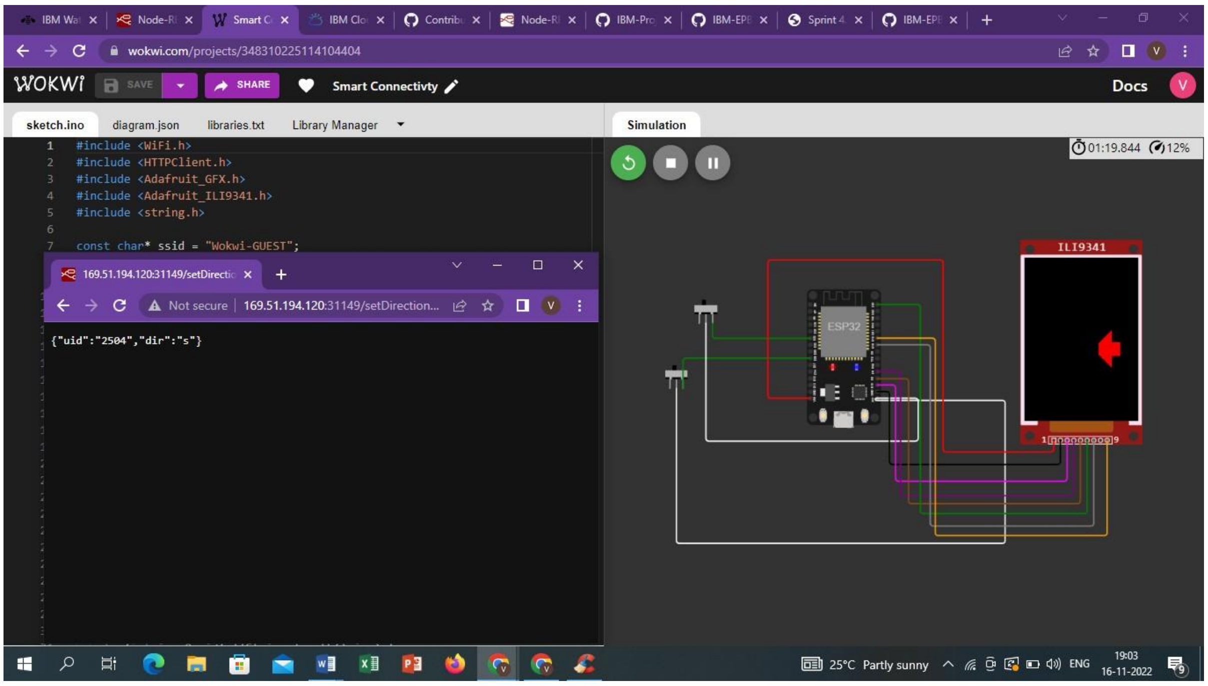
OUTPUT WOKWI



```
#include <WiFi.h>
#include <HTTPClient.h>
#include <Adafruit_GFX.h>
#include <Adafruit_ILI9341.h>
#include <string.h>

const char* ssid = "Wokwi-GUEST";
```

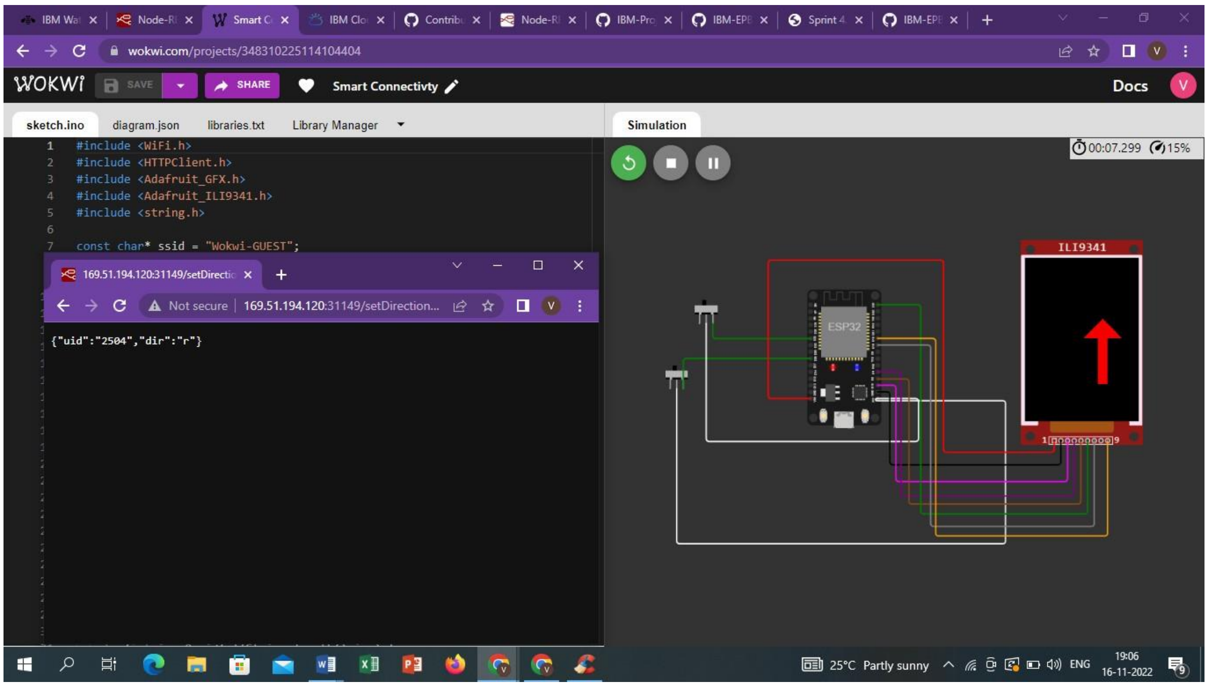
169.51.194.120:31149/setDirection...

```
{"uid":"2504","dir":"l"}
```



```
#include <WiFi.h>
#include <HTTPClient.h>
#include <Adafruit_GFX.h>
#include <Adafruit_ILI9341.h>
#include <string.h>

const char* ssid = "Wokwi-GUEST";
```

```
{"uid":"2504","dir":"s"}
```



```
#include <WiFi.h>
#include <HTTPClient.h>
#include <Adafruit_GFX.h>
#include <Adafruit_ILI9341.h>
#include <string.h>

const char* ssid = "Wokwi-GUEST";
```

```
{"uid":"2504","dir":"r"}
```