

# Delivery of Sprint –iv

## PROJECT TITLE :

IoT Based Safety Gadget for Child Safety Monitoring and Notification

TEAM ID : PNT2022TMID42176

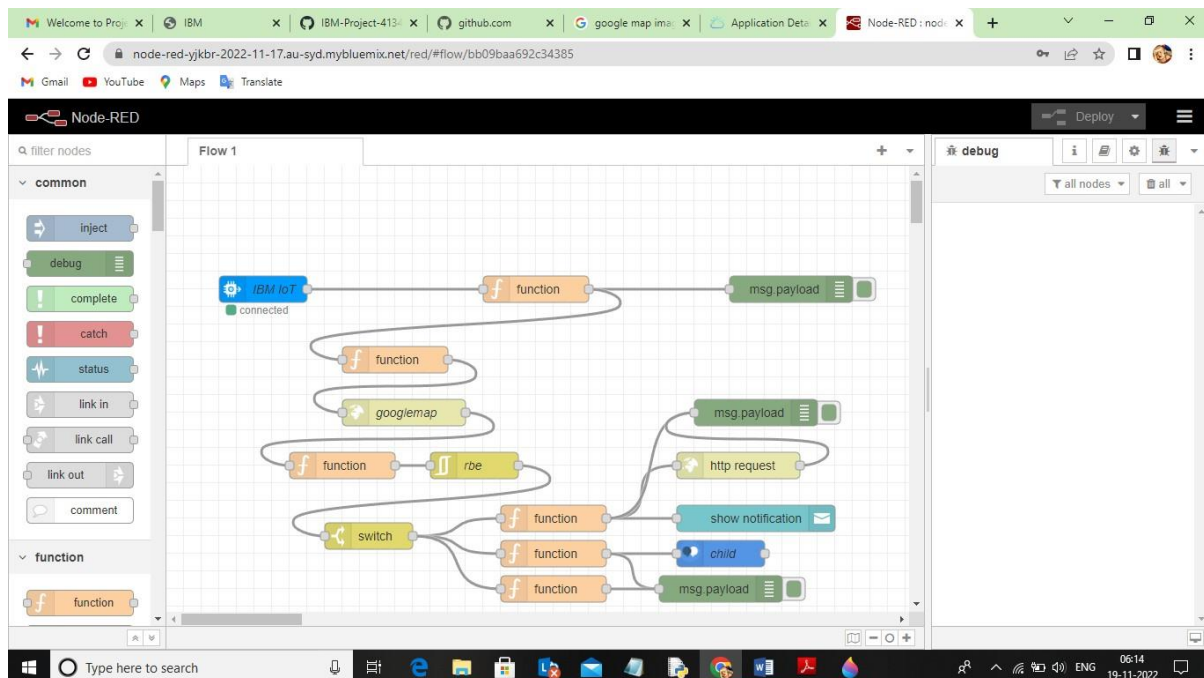
TEAM LEADER : SAMEERA N

TEAM MEMBER 1 : MYTHILI T

TEAM MEMBER 2 : YOGESHWARI V

TEAM MEMBER 3 : SOWMIYA S

The node red application is developed with the required function



## Python Code:

```
#include<wifi.h>
```

```
#include <WiFiClient.h>
```

```
#include <PubSubClient.h>
```

```
#include <ArduinoJson.h>
```

```
#include<TinyGPS++.h>
```

```
#define RXD2 16
```

```
#define TXD2 17
```

```
HardwareSerial neogps(1);
```

```
TinyGPSPlus gps; char
```

```
arr[100];
```

```
const char* ssid = "Redmi"; const
```

```
char* password = "krish@08";
```

```
#define ID "17cmwk"
```

```
#define DEVICE_TYPE "Tracker"
```

```
#define DEVICE_ID "gps1"
```

```
#define TOKEN "childtracker1"
```

```
char server[] = ID ".messaging.internetofthings.ibmcloud.com";
```

```
char publish_Topic1[] = "iot-2/evt/Data1/fmt/json"; char
```

```
publish_Topic2[] = "iot-2/evt/Data2/fmt/json"; char
```

```
authMethod[] = "use-token-auth"; char token[] = TOKEN; char
```

```
clientId[] = "d:" ID ":" DEVICE_TYPE ":" DEVICE_ID;
```

```
WiFiClient wifiClient;
```

```
PubSubClient client(server, 1883, NULL, wifiClient);
```

```
void setup() {  
    Serial.begin(115200);  
  
    Serial.println();    wifi_init();  
  
}
```

```
long previous_message = 0;
```

```
void loop() {    client.loop();
```

```
    String payload = getLocationPayload();  
    if(payload==""){  
  
        return;  
    }
```

```
    Serial.print("Sending payload: ");  
    Serial.println(payload);    if  
(client.publish(publish_Topic1, arr)) {  
  
        Serial.println("Published successfully");  
    } else {  
        Serial.println("Failed");  
    }  
    delay(2000);  
}
```

```
void wifi_init(){ WiFi.begin(ssid, password);  
neogps.begin(9600,SERIAL_8N1,RXD2,TXD2);  
while (WiFi.status() != WL_CONNECTED) {  
  
    delay(500);
```

```

        Serial.print(".");
    }

    Serial.println("");

    Serial.println(WiFi.localIP());

    if (!client.connected()) {

        Serial.print("Reconnecting client to ");

        Serial.println(server);    while (!client.connect(clientId,
authMethod, token)) {

            Serial.print(".");
            delay(500);

        }

        Serial.println("Connected TO IBM IoT cloud!");
    }
}

String getLocationPayload(){    boolean newData =
false;    for(unsigned long start = millis();millis()-
start<1000;){    while(neogps.available()){
if(gps.encode(neogps.read())){        newData = true;

        }

    }

}

    String payload;    if(newData == true){
newData = false;    payload =
locationPayloadGenerator();

    }

    else{

```

```

        Serial.println("No data");

payload = "{}";

    }

    return payload;

}

String locationPayloadGenerator(){ String payload = "{}";

if(gps.location.isValid()){ float lat = gps.location.lat(); float lon =
gps.location.lng(); payload = "{\"latitude\" : "+String(lat)+", \"longitude\" :
"+String(lon)+"}"; create_json(lat,lon);

}

return payload;

}

void create_json(float lat,float lon){
StaticJsonDocument<100> doc;

JsonObject root = doc.to<JsonObject>();

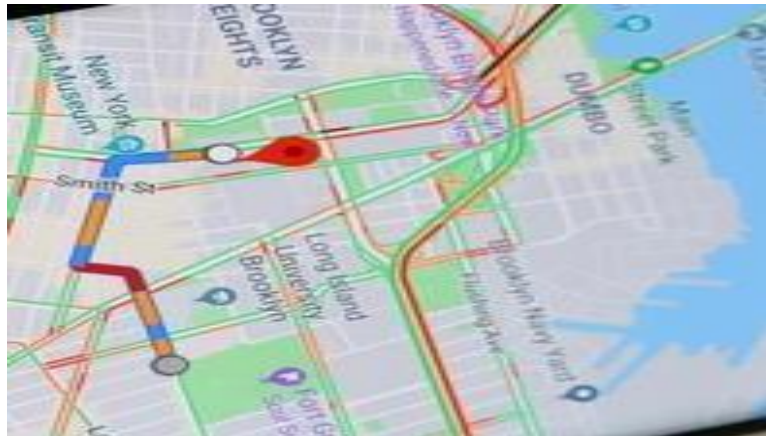
root["name"]="Child"; root["latitude"] =
lat; root["longitude"] = lon;

serializeJsonPretty(doc,arr);

}

```

## Create a geofence:

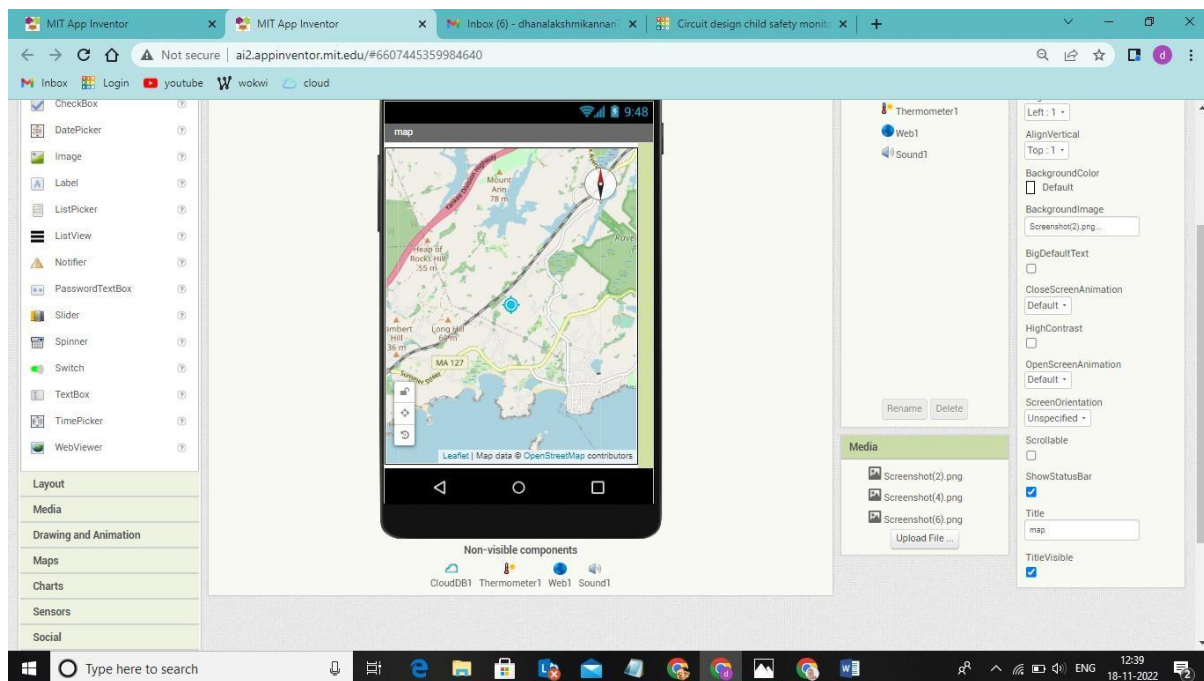


## Edit the http request URL

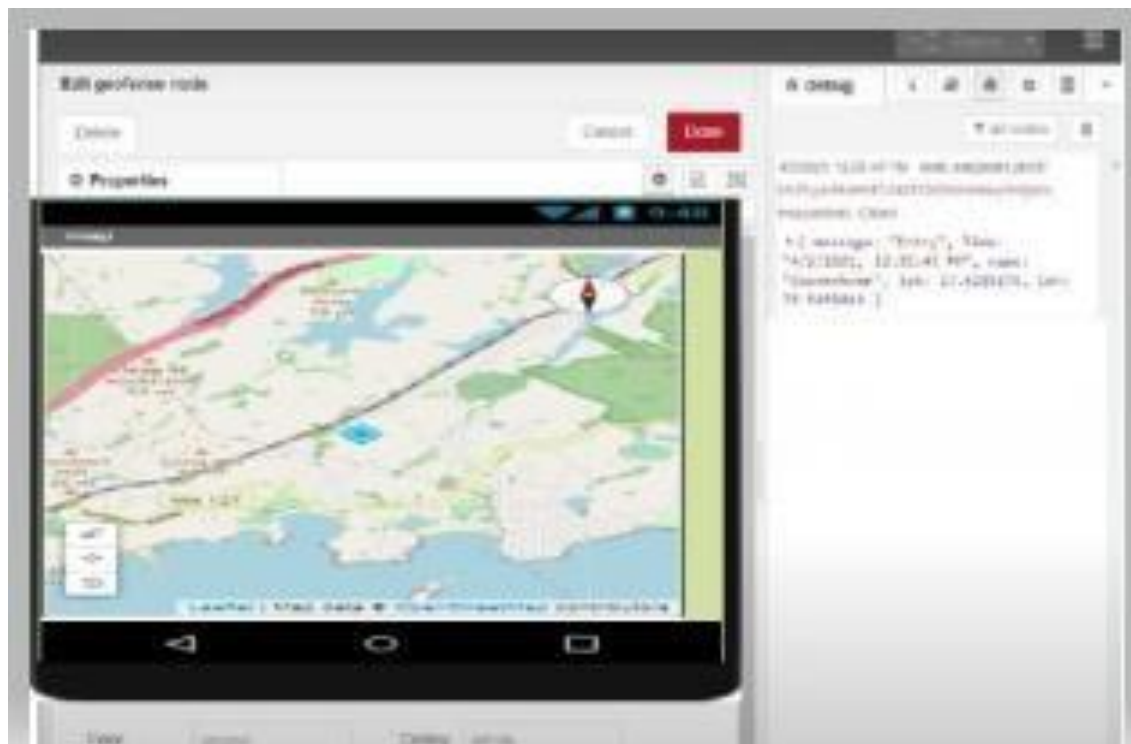
The screenshot displays the Node-RED web interface in a browser window. The address bar shows the URL: `node-red-vjkb-2022-11-17.au-syd.mybluemix.net/red/#flow/bb09baa692c34385`. The interface is divided into several sections:

- Left Sidebar:** Contains a search bar and two categories of nodes: 'common' (including inject, debug, complete, catch, status, link in, link call, link out, comment) and 'function' (including function, switch).
- Central Workspace:** Shows a flow diagram with nodes connected. A blue 'IBM IoT' node is connected to a 'connected' node, which then connects to a 'function' node, followed by a 'geofence' node, another 'function' node, and finally a 'switch' node.
- Right Panel:** The 'Edit http request node' configuration panel is open. It includes a 'Delete' button, 'Cancel' and 'Done' buttons, and a 'Properties' section. The 'Method' is set to 'GET'. The 'URL' field is highlighted with a blue selection box containing the text `payload)))&language=english&flash=0&numbers=`. The 'Payload' is set to 'Ignore'. There are several checkboxes for advanced options: 'Enable secure (SSL/TLS) connection', 'Use authentication', 'Enable connection keep-alive', 'Use proxy', and 'Only send non-2xx responses to Catch node'. The 'Return' type is set to 'a UTF-8 string'. The 'Name' field is set to 'geofence'. At the bottom of the panel, there is an 'Enabled' checkbox.
- Bottom Bar:** The Windows taskbar is visible at the bottom, showing the search bar and various application icons. The system clock indicates the time is 22:55 on 18-11-2022.

## Locate the child.



### Create the geofence code:



The web application is developed.

