

PROJECT CODE

| | |
|--------------|--|
| Team ID | PNT2022TMID15198 |
| Project Name | IoT Based Safety Gadget for Child Safety Monitoring & Notification |

Python script

```
pythoncode.py - G:\IBM\Develop the python script\pythoncode.py (3.7.0)
File Edit Format Run Options Window Help

import json
import wiotp.sdk.device
import time
import ibmiotf.application
import ibmiotf.device

myConfig = {
    "identity": {
        "orgId": "3m6fjh",
        "typeId": "ESP",
        "deviceId": "esp-vlsm"
    },
    "auth": {
        "token": "1234567890"
    }
}

client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
    name= "Vishal"

    #INSIDE GEOFENCE
    #Latitude = 10.986764
    #Longitude = 76.935569

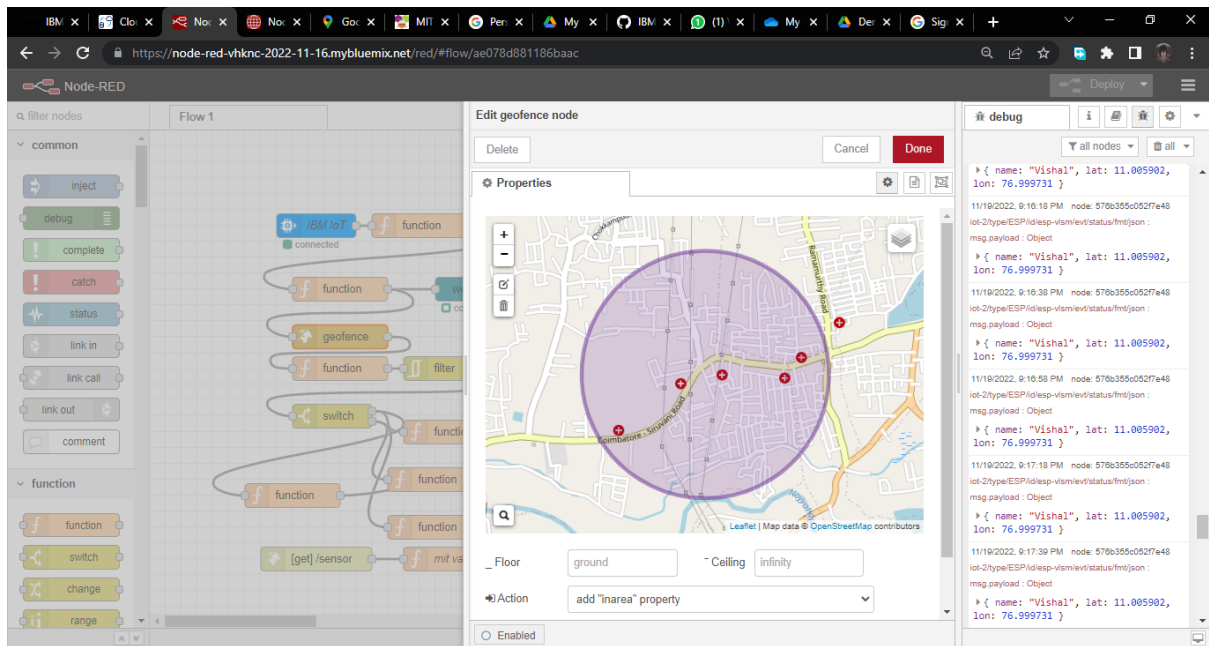
    #OUTSIDE GEOFENCE
    #Latitude = 11.005902
    #Longitude = 76.999731

    latitude= 11.005902
    longitude= 76.999731
    mydata={'name': name, 'lat':latitude, 'lon':longitude}
    client.publishEvent(eventId="status", msgFormat="json", data=mydata, qos=0, onPublish=None)
    print("Data published to IBM IoT platform: ",mydata)
    time.sleep(20)

client.disconnect()
```

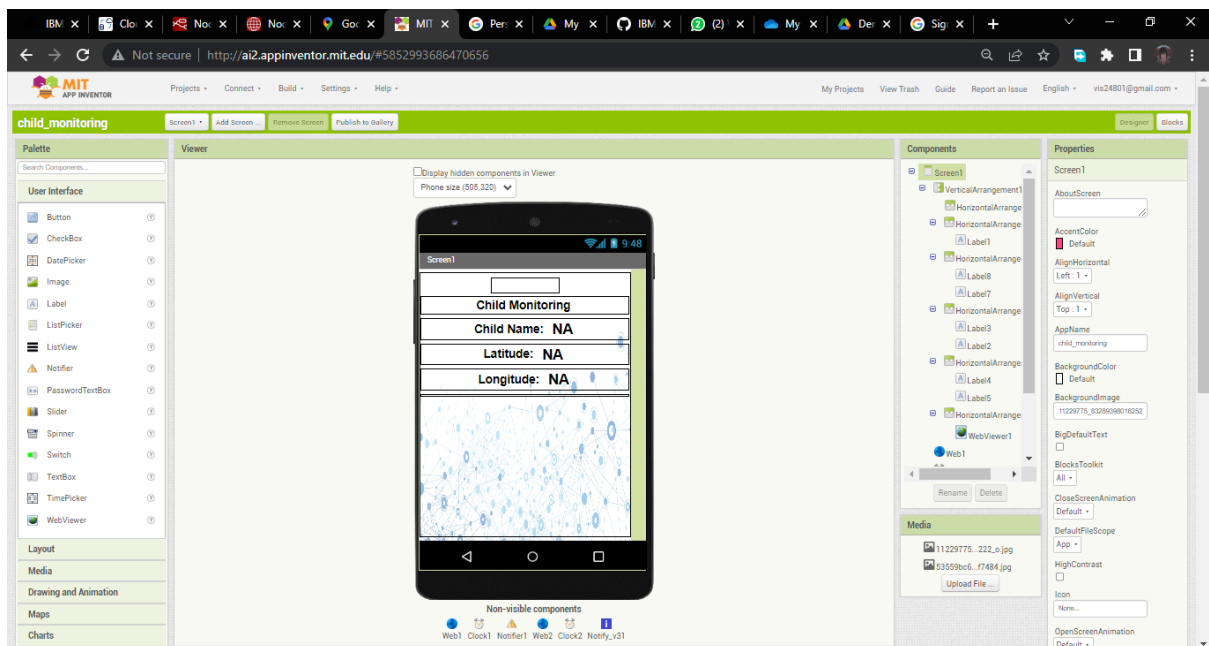
Node-Red Application

The screenshot shows the Node-RED web interface. The top bar includes navigation icons and a search bar. The left sidebar contains a 'common' section with nodes like 'inject', 'debug', 'complete', 'catch', 'status', 'link in', 'link call', 'link out', and 'comment'. Below it is a 'function' section with nodes like 'function', 'switch', 'change', and 'range'. The main workspace displays a flow diagram for 'Flow 1'. The flow starts with an 'inject' node, followed by a 'function' node, then a 'msg payload' node. It branches into two paths: one through a 'worldmap' node and another through a 'geofence' node. Both paths merge and pass through a 'filter' node. The flow then splits into two parallel paths, each containing a 'function' node, which merge and pass through a 'switch' node. The 'switch' node outputs to a 'function' node, which then connects to a 'msg payload' node. The flow also includes a '[get]/sensor' node connected to a 'mit values' node, which then connects to an 'http' node. The right sidebar shows the debug console with logs for the 'Vishal' location tracking system, displaying coordinates and timestamps.



Android Application – MIT APP Inventor

Front-End



Back-End

The screenshot shows the MIT App Inventor web interface for a project named "child_monitoring". The "Blocks" palette on the left includes categories like Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, and Procedures. The "Media" section shows two uploaded images. The main workspace displays the following logic blocks:

- Initialize global alarm1 to 0**
- when Clock1.Timer** (do block)
 - set Web1.Uri to https://node-red-vhkn-2022-11-16.mybluemix.net/**
 - call Web1.Get**
- when Web1.GetText** (do block)
 - set Label7.Text to look up in pairs key names** (pairs: call Web1.JsonTextDecode, jsonText: get responseContent)
 - set Label2.Text to look up in pairs key latitude** (pairs: call Web1.JsonTextDecode, jsonText: get responseContent)
 - set Label5.Text to look up in pairs key longitude** (pairs: call Web1.JsonTextDecode, jsonText: get responseContent)
 - set global alarm1 to look up in pairs key alarm** (pairs: call Web1.JsonTextDecode, jsonText: get responseContent)
- if get global alarm1 = 1** (then block)
 - call Notifier1.ShowAlert** (notice: "The child is outside the Geofence")

The screenshot shows the same MIT App Inventor web interface, but with additional logic blocks added to the "if" statement's then block:

- call Notifier1.ShowAlert** (notice: "The child is outside the Geofence")
- call Notify_v31.Build** (notification configuration)
 - icon**: android.R.drawable.ic_launcher
 - color**: [blue icon]
 - title**: Child Monitoring
 - text**: The child is outside the Geofence
 - numberID**: 1
 - showWhen**: true
 - autoCancel**: true
 - startValue**: screen1