IOT-BASED CHILD MONITORING SYSTEM SURVEY USING THE RASPBERRY Pi

NALAIYA THIRAN - PROJECT REPORT

PROJECT ID: PNT2022TMID00858

Submitted by

M.BALA SUBRAMANIAN [211419104037]

M.GOKUL NITHISH [211419104085]

B.ABISHEK [211419104002]

D.HARISH [211419104092]

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO ANNA UNIVERSITY)

NOVEMBER 2022

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO ANNA UNIVERSITY)

BONAFIDE CERTIFICATE

Certified that this project report

IOT-BASED CHILD MONITORING SYSTEM SURVEY USING THE RASPBERRY Pi

is the bonafide work of

M.BALA SUBRAMANIAN (211419104037)

M.GOKUL NITHISH (211419104085)

B.ABISHEK (211419104002)

D.HARISH (211419104092)

who carried out the NALAIYA THIRAN project work under the supervision.

MR.BARADWAJ
INDUSTRY MENTOR
IBM

Dr. N. PUGHAZENDHI
FACULTY MENTOR
Department of CSE
Panimalar Engineering College

Table Of Contents

1. INTRODUCTION

1.1. PROJECT OVERVIEW

The Internet of Things (IoT) is a significant part of daily living. The main distinction between embedded systems and IoT is that embedded systems have a specific protocol or software installed in the chip, but IoT devices are intelligent devices that can make decisions based on their surroundings. The advancement of sensor technology, the accessibility of devices with internet connections, and data analysis algorithms enable IoT devices to respond intelligently and autonomously to emergencies. IoT devices are so used in a variety of industries, including agriculture, medicine, industry, security, and communications. IoT solutions are helpful for performing deeper automation, analysis, and integration inside a system. IoT advancements in software, hardware, and current tools contribute to technology

1.2. PURPOSE

Basically, children cannot complain about abusements which they face in their daily life to their parents. They can’t even realize what actually happens to them at their age. It is also difficult for parents to identify their children are being abused. Since to prevent children before being attacked, an autonomous real-time monitoring system is necessary for every child out there. In this system, the collected values from every sensor like temperature sensor, pulse rate detection sensor, metal detection sensor, and the location value from GPS are used to detect the status of the child .

2. LITERATURE SURVEY

2.1. EXISTING PROBLEM

Real-Time Child Abuse and Reporting System In the existing system, we use a voice recognition module in which the alert commands from the child are stored and kept for further reference. If the same child delivers the same command, it will compare with the alert command which was previously stored and sets an emergency level according to the alert command. The GSM has a SIM which is used to send an alert message or an alert call to the trusted peoples. GPS is used to track the live location and it is used when needed. The server will search the respective device ID from the database and search for respective contacts according to that device ID and helps in alerting the registered guardians. The disadvantage of this project are, i. The child could not produce the exact alert command during a panic condition. ii. The command produced may not match with the previously stored command. iii. This project requires manual intervention.

2.2. REFERENCE

[1] A. Industries "Raspberry Pi Model B+ 512MB RAM ID: 1914 - \$29.95: Adafruit Industries Unique & funDIY electronics and kits" Adafruit.com2016[online] Available: https://www.adafruit.com/product/1914

[2] A. Industries "Raspberry Pi Camera Board ID: 1367 - \$19.95: Adafruit Industries Unique & fun DIY electronics and kits" Adafruit.com 2016 [online] Available: https://www.adafruit.com/product/1367.

[3] "IoT Based Smart Cradle System with an Android App for Baby Monitoring", 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA).

[4] "Video-based IoT baby monitor for SIDS prevention" in 2017 IEEE Global Humanitarian Technology Conference (GHTC)

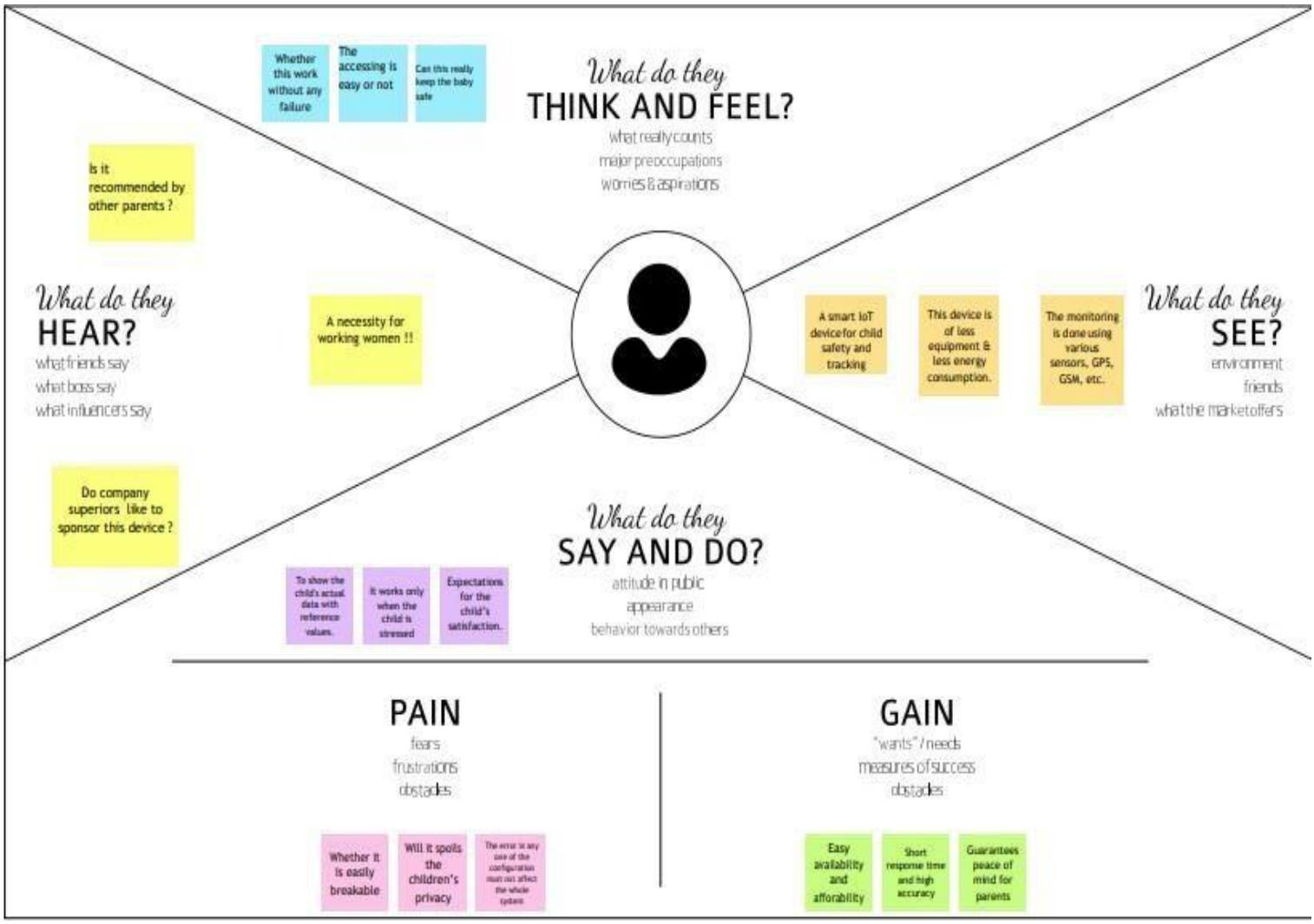
[5] Rameesa. O, C Periasamy and Priyanka. M. T. "Real Time Child Monitoring System based on Raspberry Pi and Beacon Technology using Android App". Int. J. Adv. Res.6(3), pp. 1410-1416, ISSN:2320-5407.

2.3. PROBLEM STATEMENT DEFINITION

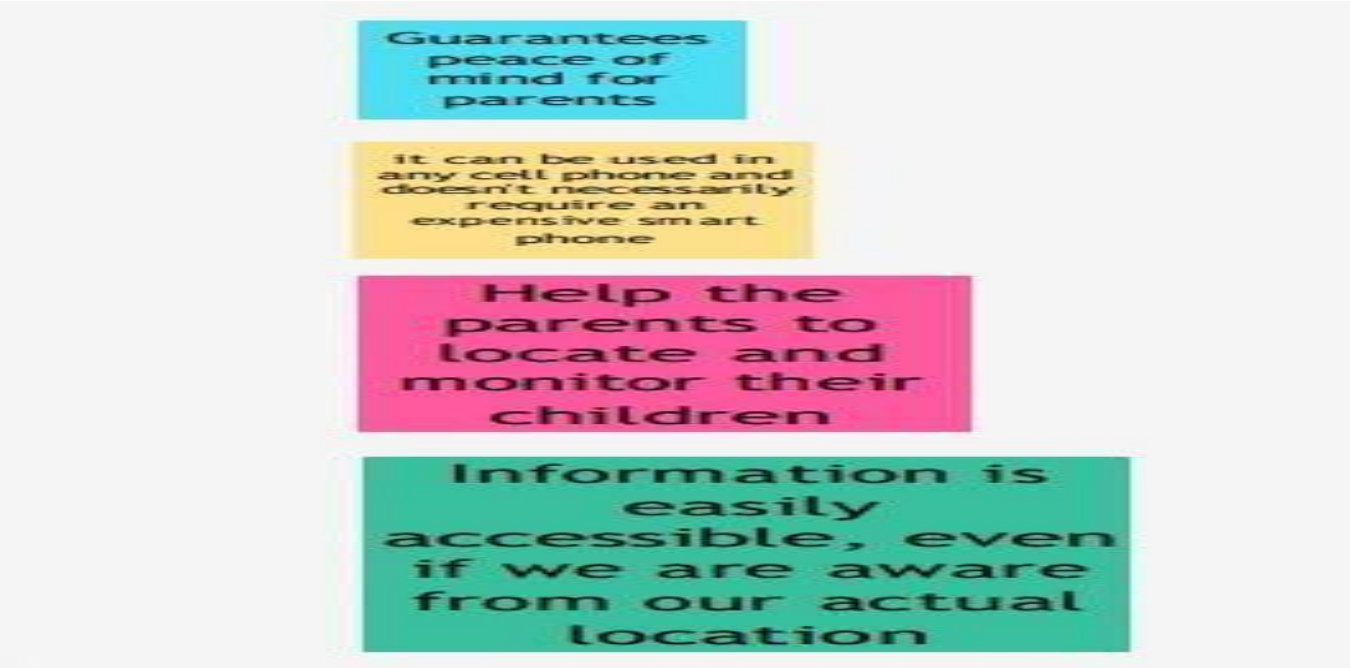
Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	Customer	Monitor the Child safety and tracking	The lack of necessary technology	Because of technical difficul es, poor data quality	Insecure
PS-2	Parent	Provide security to the children	The ba ery should be recharged regularly	Some mes we may forget to do	Concerns regarding the child's safety

3. IDEATION & PROPOSED SOLUTION

3.1. Empathy Map Canvas



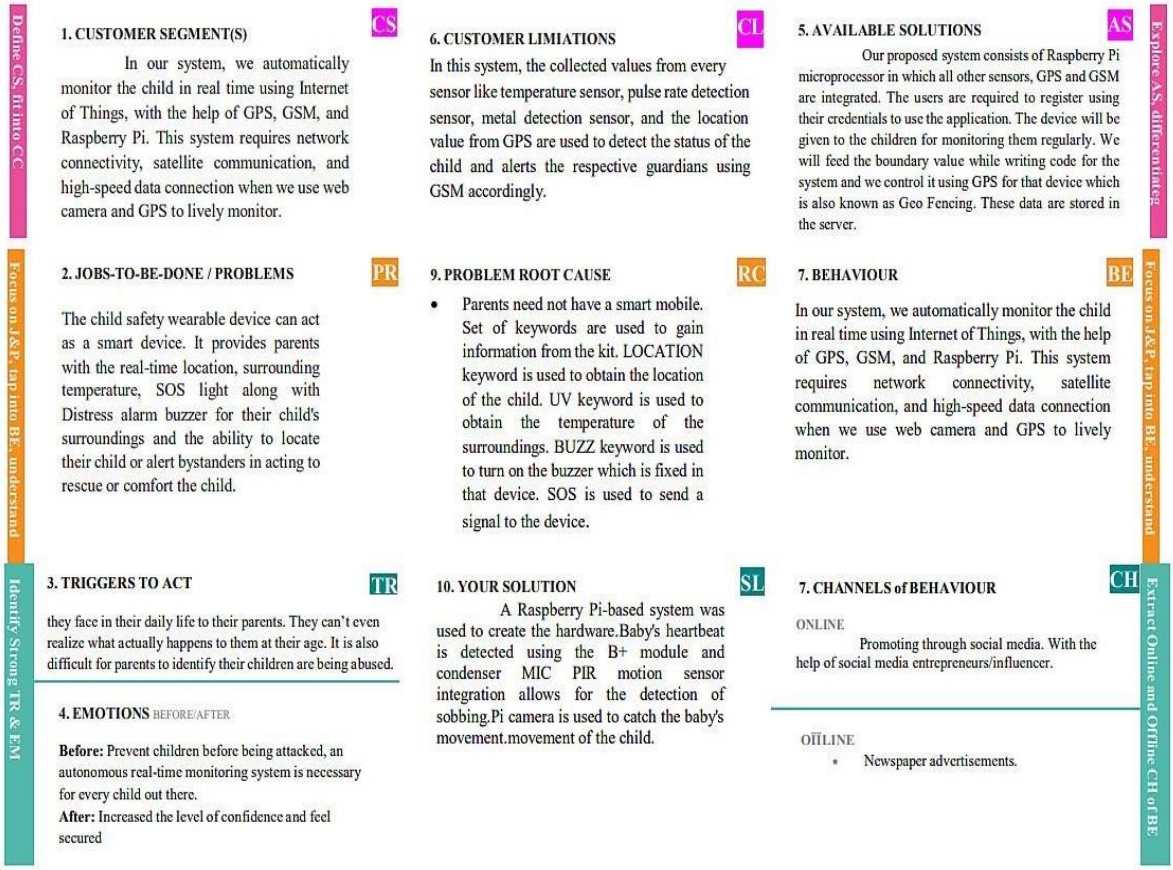
3.2. Ideation & Brainstorming



3.3. Proposed Solution

Working parents face a difficult task when taking care of a baby. In this article, we show a smart child monitoring system that enables parents to keep an eye on their child in real time and from a distance. The "Raspberry Pi 3 B +" card, a Pi camera, a sound sensor, and a temperature sensor are the foundation of the suggested system. This system uses a convolutional neural network to identify and analyse the baby status in his cradle, making it more effective. The proposed system's installation and trial findings show how effective and precise it is and how much it may aid parents in caring for their infant.

3.4. Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1. Functional requirement

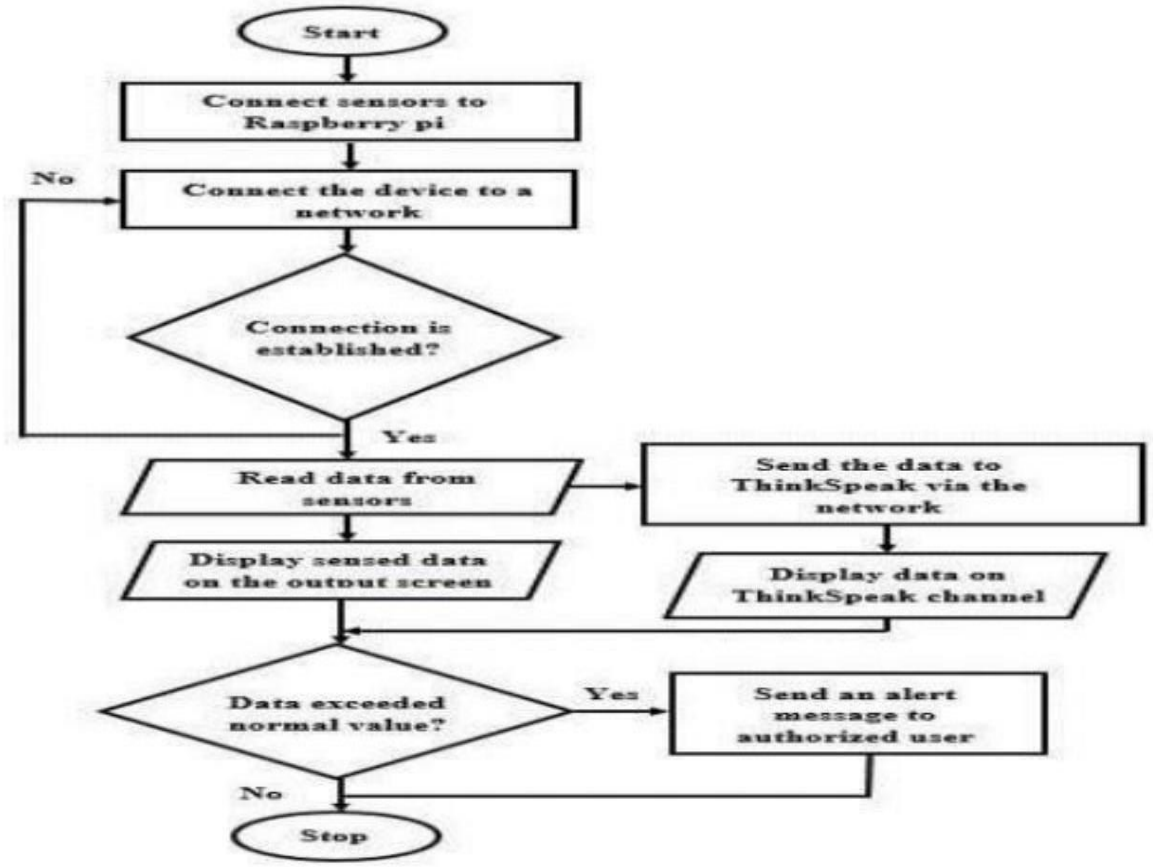
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through website Registration on through app
FR-2	User Confirmation	Confirmation on via Email Confirmation on via OTP
FR-3	User login	Configuring the user ID and password
FR-4	App permission	Give the app access to your location, contacts, and other information.
FR-5	Interface with the Device	Use the device ID to connect the device with the registered app.
FR-6	Seeing Geo-location	Establishing the Geo-location on region on the map.
FR-7	Database	The cloud is used to store location on history. accessible via the dashboard.
FR-8	Tracking location	Using an application to locate someone. using a website to locate someone.

4.2. Non-Functional requirements

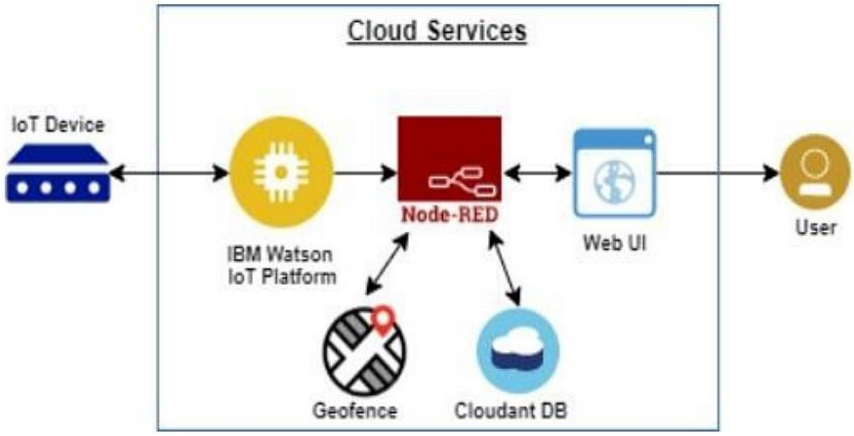
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The gadget and its apps are simple to use. The gadget is lightweight and simple to operate.
NFR-2	**Security**	Only the user has the authority to provide authorization for certain types of information. Only the user is able to access location data.
NFR-3	**Reliability**	If any issues are discovered with the gadget, an update will be sent.
NFR-4	**Performance**	When there is no network, the device's performance suffers. No cross-user interference .The monitoring of locations will be precise.
NFR-5	**Availability**	The device won't be able to function for a while if there is an upgrade.
NFR-6	**Scalability**	One device may be watched over by two individuals.

5. PROJECT DESIGN

5.1. Data Flow Diagrams



5.2. Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register my account by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered myself	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through apple account	I can register & access the dashboard with apple account Login	High	Sprint-2
	Login	USN-4	As a user, I can log into the application by entering user id & password		High	Sprint-1
Customer Care Executive	Login		As I enter I can view the working of the application and scan for any glitches and monitor the operation and check if all the users are authorized	I can login only with my provided credentials	Medium	Sprint - 3
Administrator	Login		Maintaining and making sure the database containing the locations are secure and accurate and updated constantly.	I can login only with my provided credentials	High	Sprint - 3

6. PROJECT PLANNING & SCHEDULING

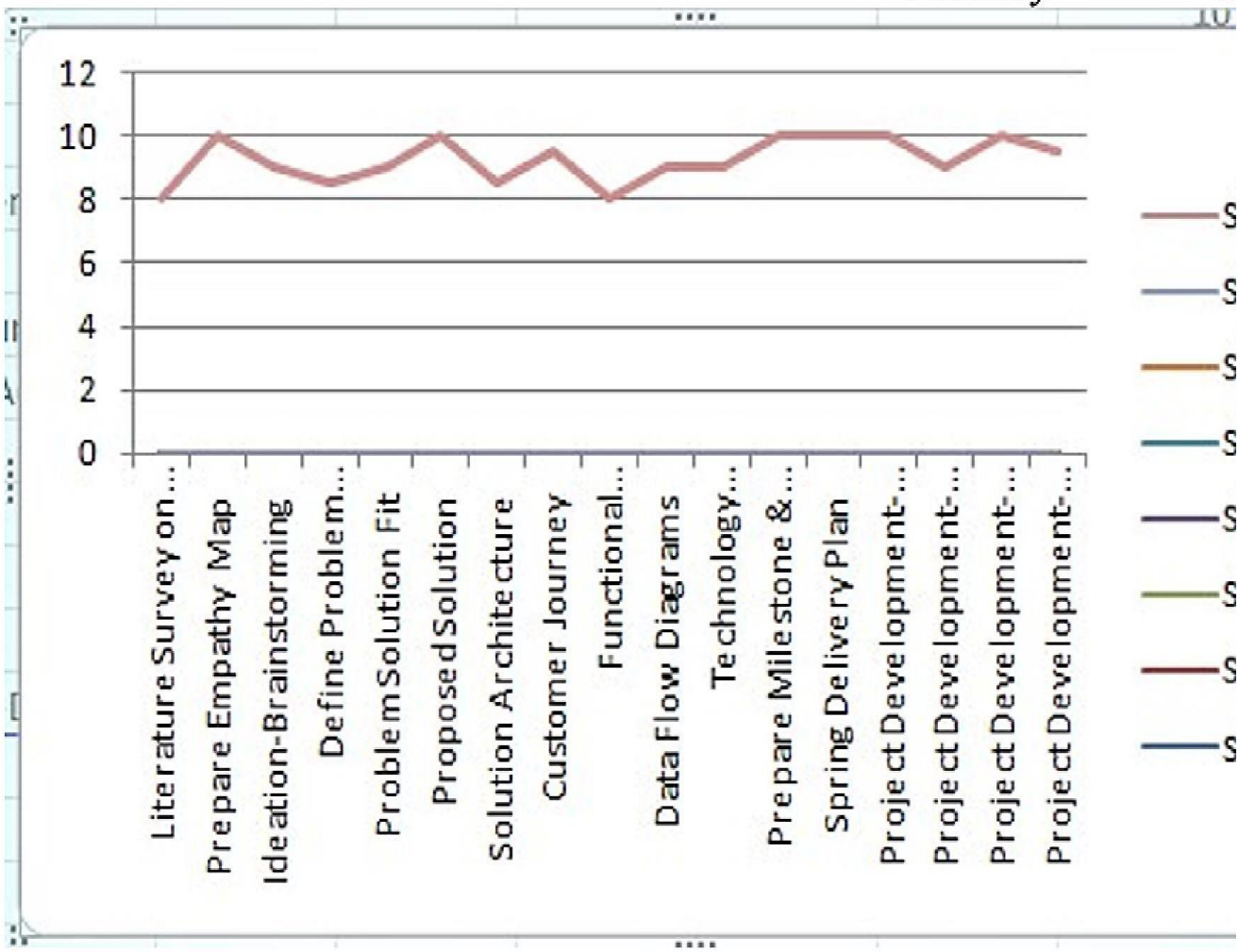
6.1 Sprint Planning & Estimation

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Releas (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	31 Oct 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	07 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	14 Nov 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the tea (points per sprint). Let's calculate the team's average velocity (A iteration unit).

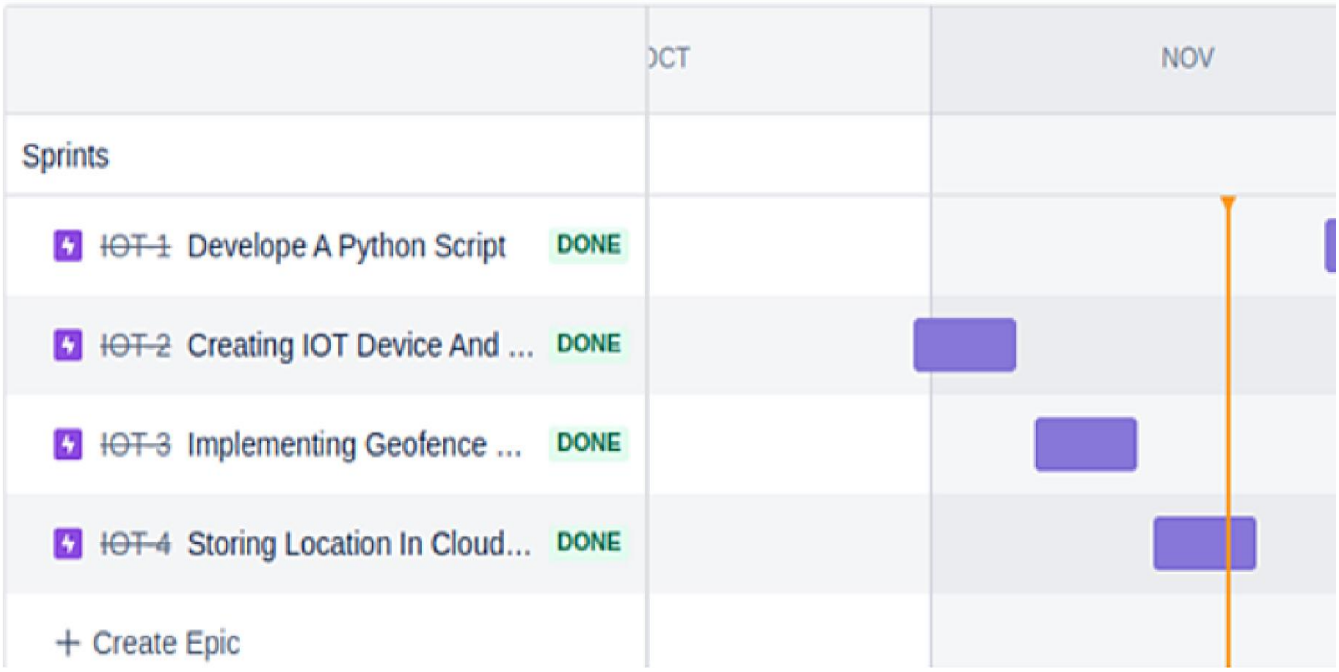
$$AV = \frac{sprint\ duration}{velocity} = \frac{2}{1}$$



6.2 Sprint Delivery Schedule

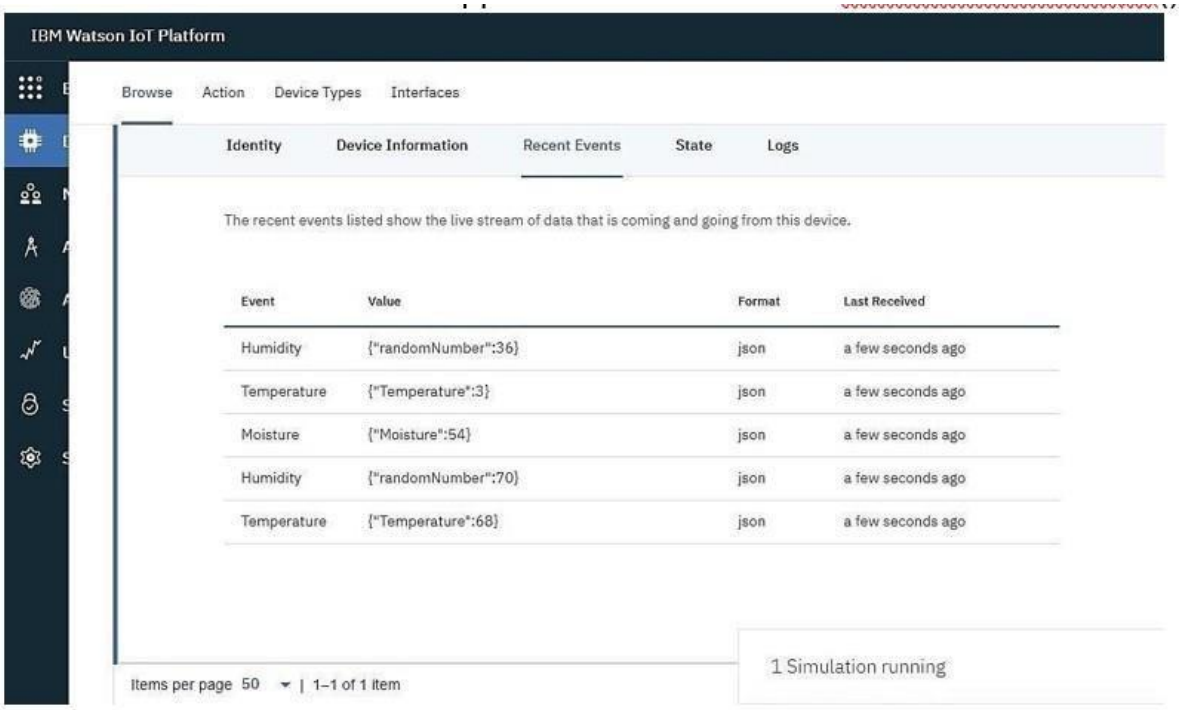
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	31 Oct 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov2022	20	09 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov2022	20	14 Nov 2022

6.3 Reports from JIRA



7. CODING & SOLUTIONING

7.1 Feature 1



7.2 Feature 2

import random

import ibmiotf.applic ation import ibmiotf.devi ce from time import sleep import sys

#IBM Watson Device Credentials.

organization = "op701j" deviceType = "Lokesh" deviceId = "Lokesh89" authMethod = "token" authToken = "122333444 4" def myCommand

Callback(cmd):

print("Comma nd received: %s" % cmd.data['co mmand'])

status=cmd.d ata['command '] if status=="spri nkler_on": print ("sprinkler is ON") else :

print ("sprinkler is OFF") #print(cmd) t r y : deviceOptions = {"org": organization, "type": deviceType, "id":

deviceId, "auth-method": authMethod, "authtoken": authToken} deviceCli = ibmiotf.device.Client(deviceOptions) except Exception as e:

print("Caught exception connecting device: %s" % str(e)) sys.exit()

#Connecting to IBM watson. deviceCli.connect() while True: #Getting values from sensors.

temp_sensor = round(random.uniform(0 ,80),2) PH_sensor = round(random.uni form(1,14),3)

camera = ["Detected","Not Detected","Not Detected","Not Detected","Not Detected","Not Detected",]

camera_reading = random.choice(camera) flame = ["Detected","Not Detected","Not Detected","Not Detected","Not Detected",] flame_reading = random.choice(flame) moist_level = round(random.uniform(0,100),2)

water_level = round(random.uniform(0,30),2) #storing the sensor data to send in json format to cloud. temp_data = { 'Temperature' : temp_sensor }

PH_data = { 'PH Level' : PH_sensor } camera_data = { 'Animal attack' : camera_reading} flame_data = { 'Flame' : flame_reading }

moist_data = { 'Moisture Level' : moist_level} water_data = { 'Water Level' : water_level}

publishing Sensor data to IBM Watson for every 5-10 seconds. success = deviceCli.

publishEvent("Tempera ture sensor", "json", temp_data, qos=0) sleep(1) if success: print (" .....publish ok ") print ("Published Temperature =

%s C" % temp_sensor, "to IBM Watson") success = deviceCli.publishEvent("PH sensor", "json", PH_data, qos=0) sleep(1)

if success: print ("Published PH Level = %s" % PH_sensor, "to IBM Watson") success = deviceCli.publishEvent("camera ", "json", camera_data, qos=0) sleep(1)

if success: print ("Published Animal attack %s " % camera_reading, "to IBM Watson") success = deviceCli.publishEvent("Flame sensor", "json", flame_data, qos=0) sleep(1)

if success: print ("Published Flame %s " % flame_reading, "to IBM Watson") success = deviceCli.publishEvent("Moisture sensor", "json", moist_data, qos=0) sleep(1)

if success: print ("Published Moisture Level = %s " % moist_level, "to IBM Watson") success = deviceCli.publishEvent("Water

sensor", "json", water_data, qos=0) sleep(1)

if success: print ("Published Water Level = %s cm" % water_level, "to IBM Watson") print ("") #Automation to control sprinklers by present temperature an to send alert message to IBM Watson.

if (temp_sensor > 35): print("sprinkler-1 is ON") success = deviceCli.publishEvent("Alert1", "json",{ 'alert1' : "Temperature(%s) is high, sprinkerlers are turned ON" %temp_sensor } ,

qos=0) sleep(1) if success: print('Published alert1 : ', "Temperature(%s) is high, sprinkerlers are turned ON" %temp_sensor,"to IBM Watson")

print("") else: print("sprinkler-1 is OFF") print("") #To send alert message if farmer uses the unsafe fertilizer to crops.

if (PH_sensor > 7.5 or PH_sensor < 5.5): success = deviceCli.publishEvent("Alert2", "json",{ 'alert2' : "Fertilizer PH level(%s) is not safe,use other fertilizer" %PH_sensor } ,

qos=0) sleep(1) if success: print('Published alert2 : ' , "Fertilizer PH level(%s) is not safe,use other fertilizer" %PH_sensor,"to IBM Watson") print("")

#To send alert message to farmer that animal attack on crops. if (camera_reading == "Detected"): success = deviceCli.publishEvent("Alert3", "json", { 'alert3' : "Animal attack on crops detected" }, qos=0) sleep(1)

if success: print('Published alert3 : ' , "Animal attack on crops detected","to IBM Watson","to IBM Watson") print("")

#To send alert message if flame detected on crop land and turn ON the splinkers to take immediate action.

if (flame _readi ng == "Detec ted"): print(" sprinkl er-2 is ON") success = deviceCli.publishEvent("Alert4", "json", { 'alert4' : "Flame is detected crops are in danger,sprinklers turned ON" }, qos=0) sleep(1)

if success: print('Published alert4 : ' , "Flame is detected crops are in danger,sprinklers turned ON","to IBM Watson")

#To send alert message if Moisture level is LOW and to Turn ON Motor-1 for irrigation.

if (moist_level < 20): print("Motor-1 is ON") success = deviceCli.publishEvent("Alert5", "json", { 'alert5' : "Moisture level(%s) is low, Irrigation started" %moist_level }, qos=0) sleep(1)

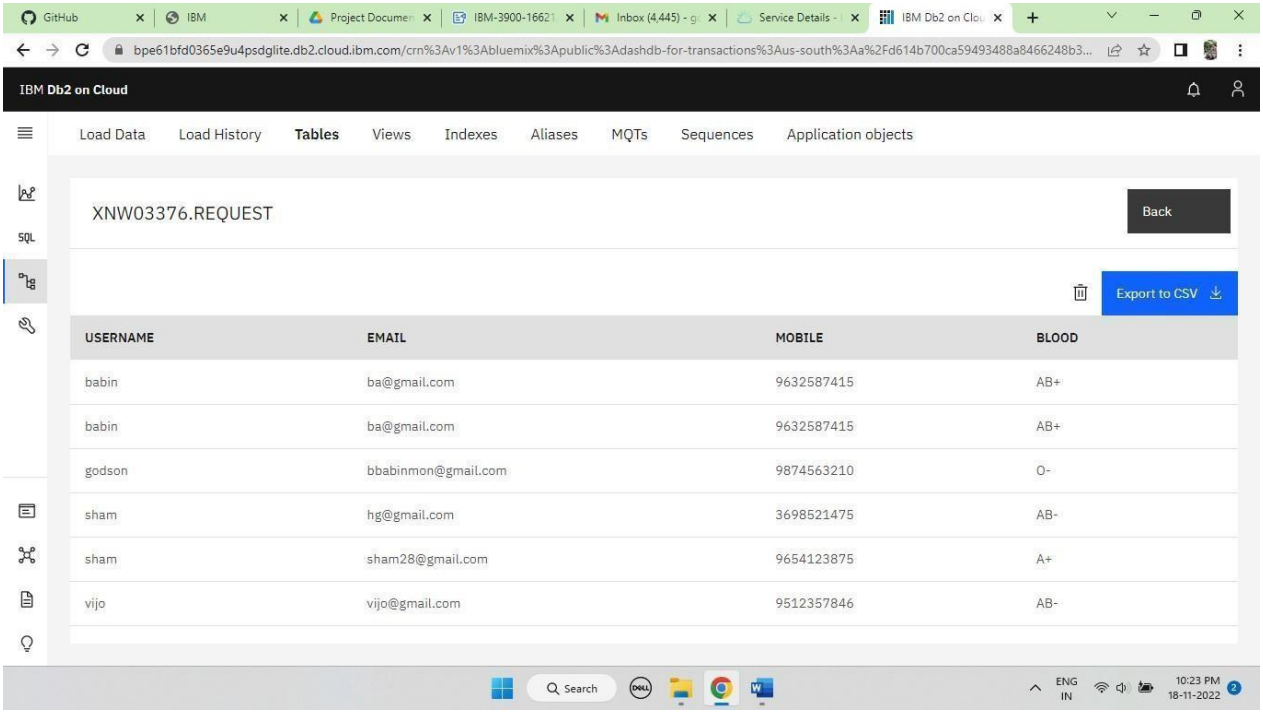
if success: print('Published alert5 : ' , "Moisture level(%s) is low, Irrigation started" %moist_level,"to IBM Watson") print("")

#To send alert message if Water level is HIGH and to Turn ON Motor-2 to take water out. if (water_level > 20): print("Motor-2 is ON")

success = deviceCli.publishEvent("Alert6", "json", { 'alert6' : "Water level(%s) is high, so motor is ON to take water out " %water_level }, qos=0) sleep(1)

if success: print('Published alert6 : ' , "water level(%s) is high, so motor is ON to take water out " %water_level,"to IBM Watson")

print("") #command recived by farmer deviceCli.commandCallback = myCommandCallback # Disconnect the device and application from the cloud deviceCli.disconnect()



BUZZER

Specifications:

1. RatedVoltage : 6V DC
2. Operating Voltage : 4 to 8V DC
3. Rated Current*: ≤30mA
4. SoundOutput at 10cm* : ≥85dB
5. Resonant Frequency : 2300 ±300Hz
6. Tone: Continuous A buzzer is a loud noise maker

Most modern ones are civil defense or air- raid sirens, tornado sirens, or the sirens on emergency service vehiclessuch as ambulances, police cars and fire trucks. There are two general types, pneumatic and electronic.

FEATURE-2

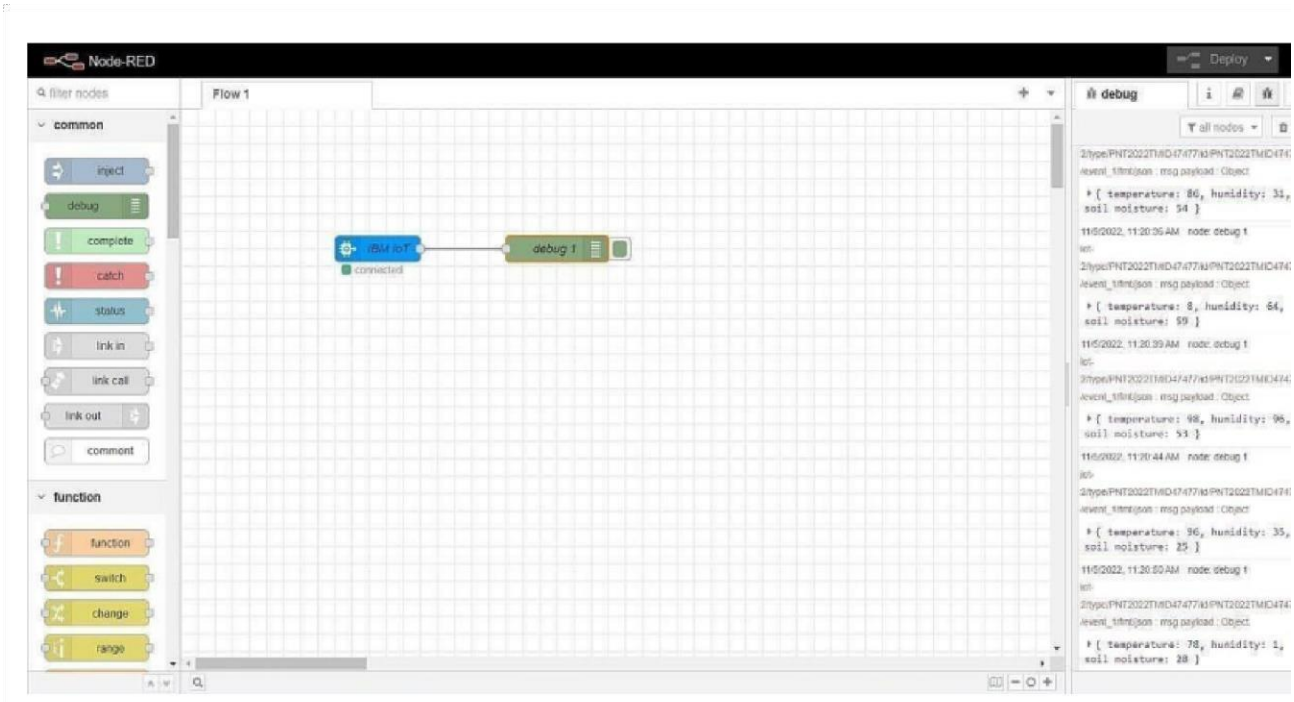
i. Goodsensitivity to Combustible gas in wide range
ii. Longlife and low cost.
iii. iii Simpledrive circuit

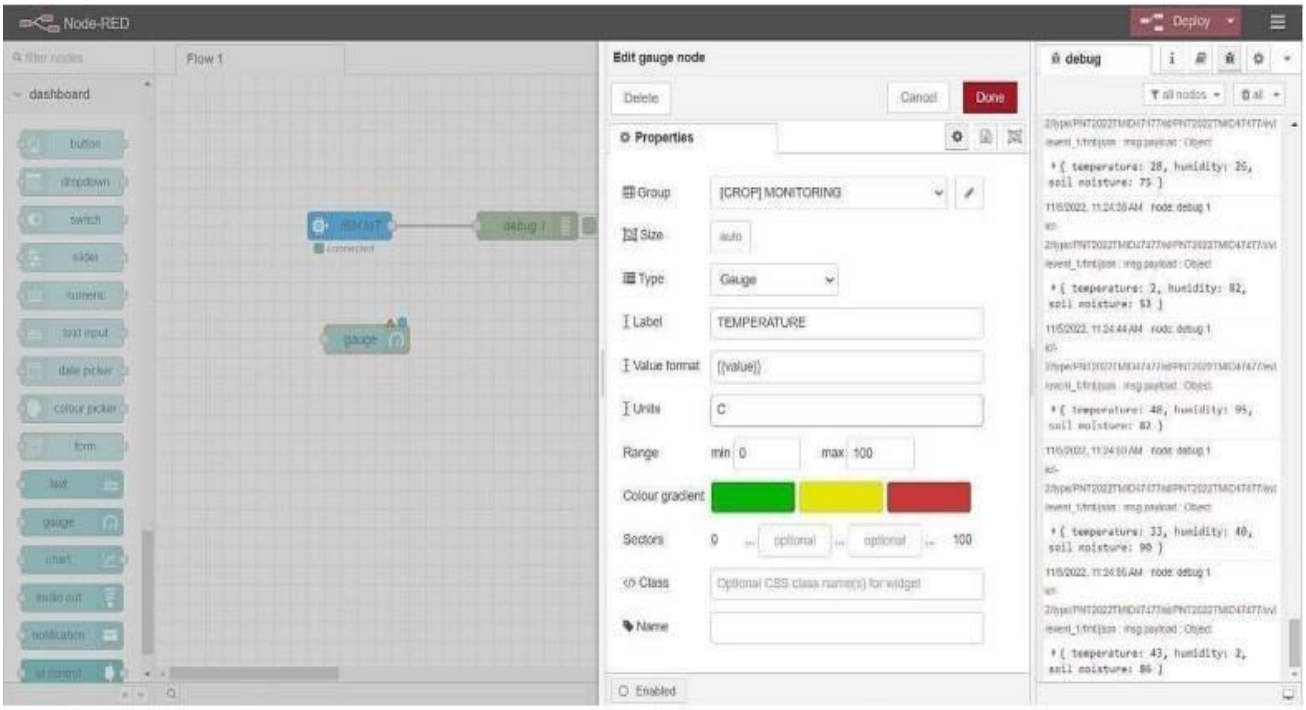
8. TESTING

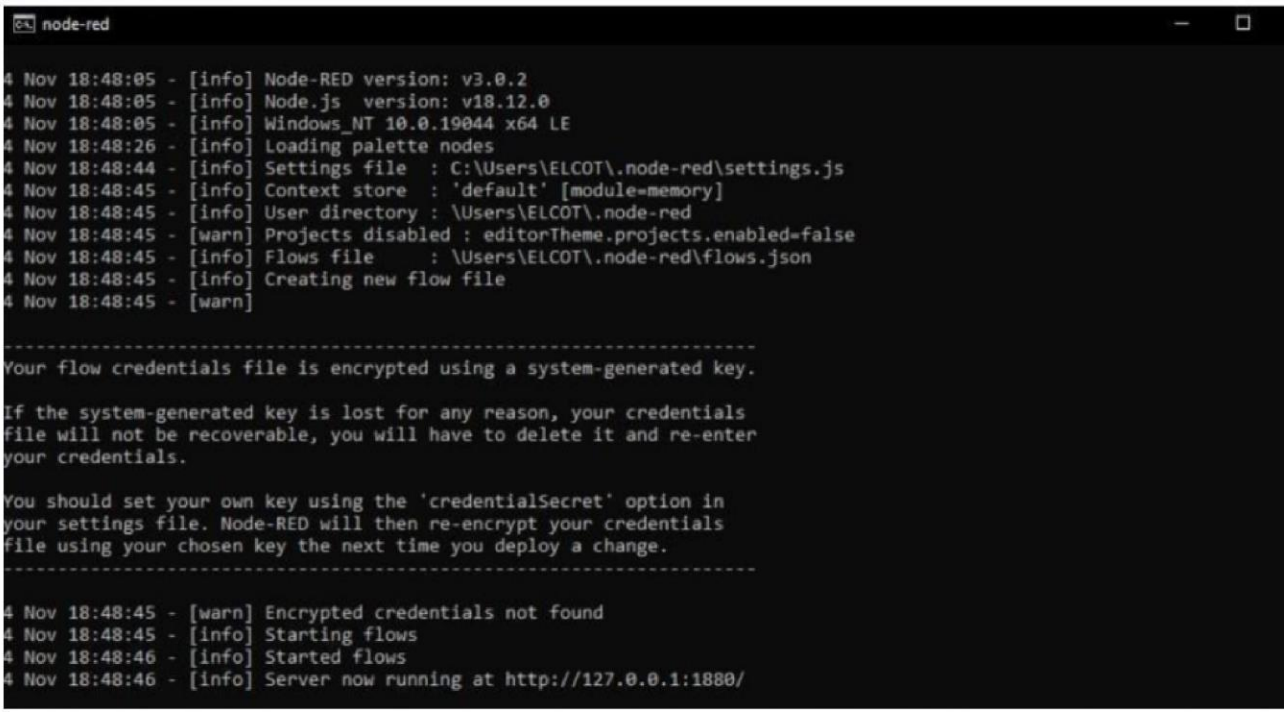
8.1 Test Cases

sno	parameter	Values
1	Model summary	-
2	accuracy	Training accuracy- 95% Validation accuracy- 72%
3	Confidence score	Class detected- 80% Confidence score-80%

8.2 USER ACCEPTANCE TESTING




Node-RED
Deploy
Flow 1
dashboard
button
dropdown
switch
slider
numeric
text input
date picker
colour picker
form
text
gauge
chart
audio out
notification
ui control
MQTT
connected
debug 1
gauge
Edit gauge node
Delete
Cancel
Done
Properties
Group
[CROP] MONITORING
Size
auto
Type
Gauge
Label
TEMPERATURE
Value format
{{value}}
Units
C
Range
min 0
max 100
Colour gradient
Sectors
0
optional
optional
100
Class
Optional CSS class name(s) for widget
Name
Enabled
debug
all nodes
all
{ temperature: 28, humidity: 20, soil moisture: 75 }
11/5/2022, 11:24:38 AM node: debug 1
{ temperature: 2, humidity: 82, soil moisture: 53 }
11/5/2022, 11:24:44 AM node: debug 1
{ temperature: 48, humidity: 95, soil moisture: 82 }
{ temperature: 33, humidity: 40, soil moisture: 90 }
11/5/2022, 11:24:56 AM node: debug 1
{ temperature: 43, humidity: 3, soil moisture: 86 }


node-red
4 Nov 18:48:05 - [info] Node-RED version: v3.0.2
4 Nov 18:48:05 - [info] Node.js version: v18.12.0
4 Nov 18:48:05 - [info] Windows_NT 10.0.19044 x64 LE
4 Nov 18:48:26 - [info] Loading palette nodes
4 Nov 18:48:44 - [info] Settings file : C:\Users\ELCOT\.node-red\settings.js
4 Nov 18:48:45 - [info] Context store : 'default' [module=memory]
4 Nov 18:48:45 - [info] User directory : \Users\ELCOT\.node-red
4 Nov 18:48:45 - [warn] Projects disabled : editorTheme.projects.enabled=false
4 Nov 18:48:45 - [info] Flows file : \Users\ELCOT\.node-red\flows.json
4 Nov 18:48:45 - [info] Creating new flow file
4 Nov 18:48:45 - [warn]
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
4 Nov 18:48:45 - [warn] Encrypted credentials not found
4 Nov 18:48:45 - [info] Starting flows
4 Nov 18:48:46 - [info] Started flows
4 Nov 18:48:46 - [info] Server now running at http://127.0.0.1:1880/

9. RESULTS

One of the modules in our project is a temperature sensor, which is used to gauge both the child's internal temperature and the ambient temperature. The device will alert the user according to the programmed time delay as shown in the figure if there is any unusual rise or dip in the child's body temperature or that of the immediate environment. The user will be alerted based on predefined value abnormal fall or rise scenarios as the temperature and humidity values are displayed

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

1. **Easy Availability& Affordability:**

Gone are the days when buying a GPS enabled We are able Device for kids was considered a luxury. Today, however, the scenario is different. There are plenty of options readily available. It is easy to buy a smart watch for kids of your choice online. What's more, these magnificent tech gadgets don't burn a big hole in your pockets and make up for an affordable buy. Now a smart watch is just a click away! Besides ,these smart-watches lend a style statement to your fashion conscious kids.

2. **Tracking Made Easy:**

Fueled by IOT, the GPS enabled Wearable Device act as a saviour for parents who are always clouded with worries about their kids. Tracking a child was never this easy. These Wearable Device allow parents to track their children in crowded/public places or when they are out of sight say at school, picnic or an outing. Parents can use these smart-watches to track the location of their lost kids

3. **Smart watch is Technology in Disguise:**

No matter how tech advanced the smart watches are, the hardly look like one. Most manufacturers have worked hard to mold their tech wonders in a time piece that looks everything but a tech piece! Their childish designs and bright colour combination is perfect to disguise them. This is precisely why most people can hardly spot the difference between a smart watch and an ordinary watch. Good for kids who use them, as their adorable designs keep these watches safe from the prying eyes

4. **Watches Over Your Kids:**

GPS tracker watches are a boon for parents as they help in watching over your kids when either they are away or you are away from them. These devices:

a. Tracks kids when they reach school or arrive home from school.

b. Track kids when they are untraceable in a crowded space.

c. Track kids when they are away from home and out of your sight

5. **Guarantees Peace of Mind to Parents:**

Parents, whether at home or office, are always worried about the safety of their kids. The fear of losing your child to avoidable circumstances is the concern area for all mommies and daddies. On the other hand, a smart watch equipped kid is always traceable and reachable in case of contingencies and emergencies. This in fact, offers great solace for parents, who are relieved at the thought of maintaining an uninterrupted connectivity with their children, anytime, anywhere. Enough to of course, guarantee the much-needed peace of mind.

DISADVANTAGES:

High cost, but once done, the cost can be decreased Battery.

11. CONCLUSION

The child safety wearable device is capable of acting as a smart IOT device. It provides parents with the real-time location, surrounding temperature, UV radiation index and SOS light along with Distress alarm buzzer for their child's surroundings and the ability to locate their child or alert bystanders in acting to rescue or comfort the child. The smart child safety wearable can be enhanced much more in the future by using highly compact Arduino modules such as the Lilypad Arduino which can be sewed into fabrics. Also a more power efficient model will have to be created which will be capable of holding the battery for a longer time .

12. FUTURE SCOPE

In our system, we automatically monitor the child in real time using Internet of Things, with the help of GPS, GSM, and Raspberry Pi. This system requires network connectivity, satellite communication, and high-speed data connection when we use web camera and GPS to lively monitor. It is difficult to monitor when there occurs any hindrance to satellite communication or any network issue. There also occurs time delay in video streaming through the server. Hence in the future, these issues can be overcome by using Zigbee concept or accessing the system without internet and using highspeed server transmission.

13. APPENDIX

13.1 Source Code

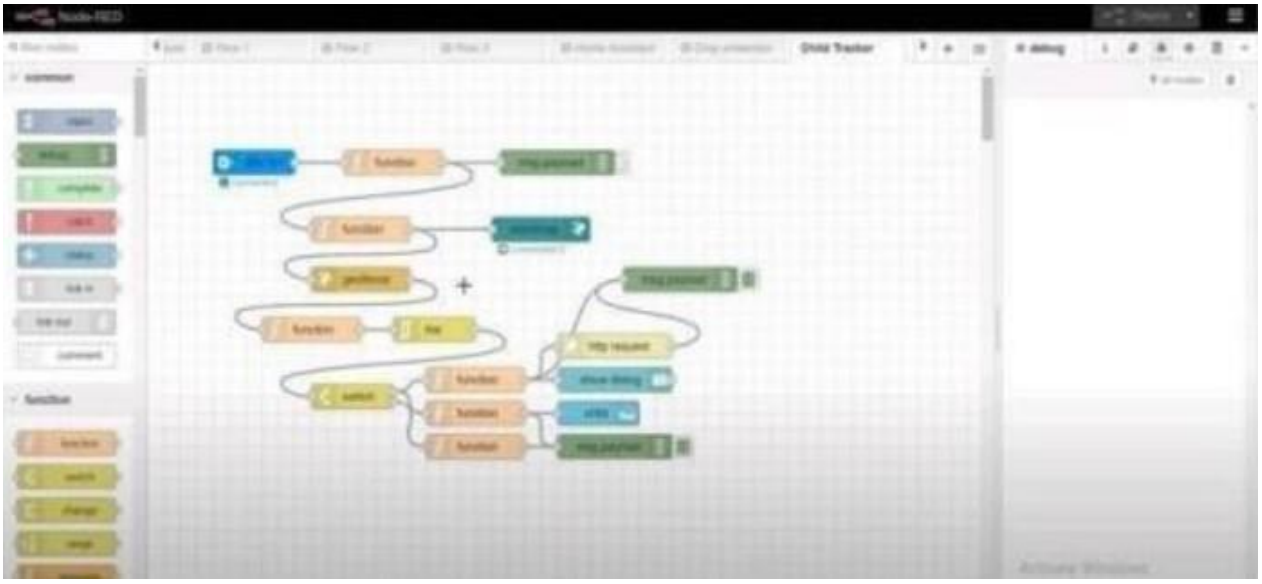
```
import time importsys
 import ibmiotf.application # toinstallpip install ibmiotf importibmiotf.device
# Provide your IBM Watson Device Credentials organization = "8gyz7t"
 # replace the ORG ID deviceType = "weather_monitor"
 #replace the Device type deviceId = "b827ebd607b5"
 # replace Device ID authMethod = "token" authToken = "LWVpQPaVQ166HWN48f"
 # Replace the authtoken def myCommandCallback(cmd):
# function for Callbackif
cm.data['command'] == 'motoron':
 print("MOTOR ON IS RECEIVED")
elif cmd.data['command'] == 'motoroff':print("MOTOR OFF IS RECEIVED")
 if cmd.command == "setInterval":
else: if 'interval' not in cmd.data:
 printf(“output”) try: deviceOptions = {"org": organization, "type": deviceType, "id":
deviceId,"authmethod": authMethod, "auth-token": authToken} deviceCli =
ibmiotf.device.Client(deviceOptions)
# exceptException as e:
 print("Caught exception connecting device: %s" % str(e))sys.exit()
# Connect and send a datapoint "hello" with value "world" into the cloud as an event
oftype "greeting" 10 times deviceCli.connect()
while True: deviceCli.commandCallback = myCommandCallback
 # Disconnect the device and application from the cloud deviceCli.disconnect()
```

SENSOR.PY

```
import time
 import sysimport ibmiotf.application
 importibmiotf.device import random
# Provide your IBM Watson Device Credentials organization = "8gyz7t" # replace the ORG ID
```

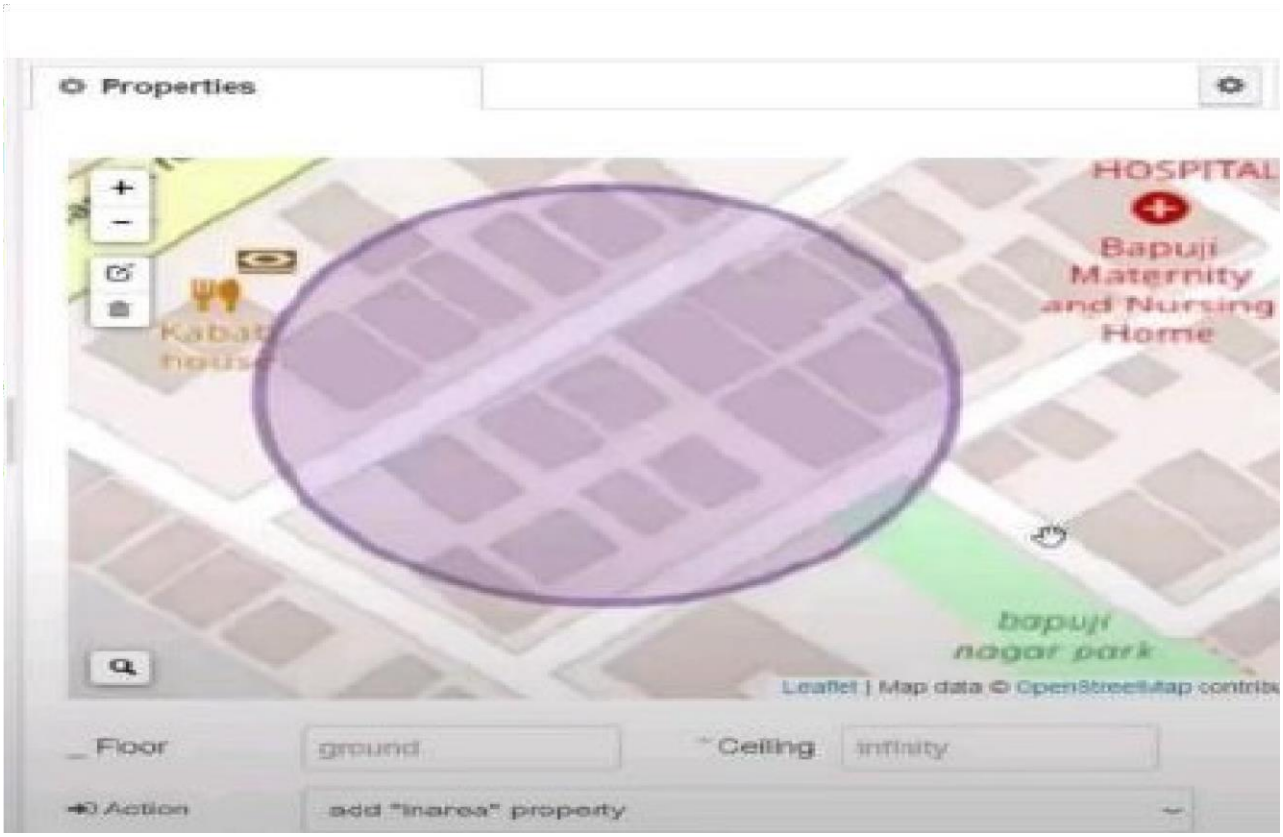
```
deviceType = "weather_monitor" #replace the Device type deviceId = "b827ebd607b5"
 # replace Device ID authMethod = "token" authToken = "LWVpQPaVQ166HWN48f"
 # Replace the authtoken def myCommandCallback(cmd):
 print("Command received: %s" % cmd.data['command'])
 print(cmd)
 try: deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-
method": authMethod, "auth-token": authToken} deviceCli =
ibmiotf.device.Client(deviceOptions)
exceptException as e:
print("Cau ght exception connecting device: %s" % str(e))sys.e xit()
# Connect and send a datapoint "hello" with value "world" into the cloud as an event of
type "greeting" 10 times deviceCli.connect() 00)
 pulse=random.randint(0,100) soil=random.randint(0,100)
data = { 'temp' : temp, 'pulse': pulse ,'soil':soil} #print data def myOnPublishCallback():
print ("Published Temperature = %s C" % temp, "Humidity = %s %%"
%pulse,"Soil Moisture = %s %%" % soil,"to IBM Watson")
deviceCli.commandCallback = myCommandCallback
# Disconnect the device and application from the cloud deviceCli.disconnect()
```

1.OPEN A NODE-RED PROJECT

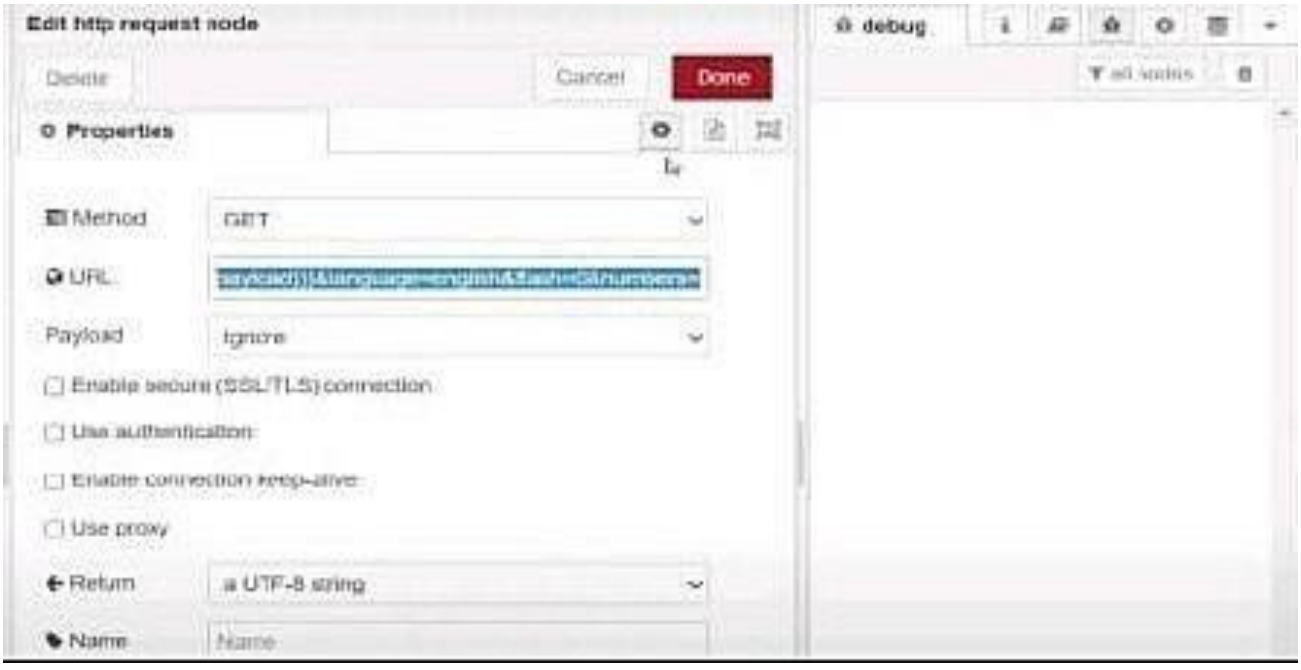


2.ADD CODE TO GET CHILD LOCATION

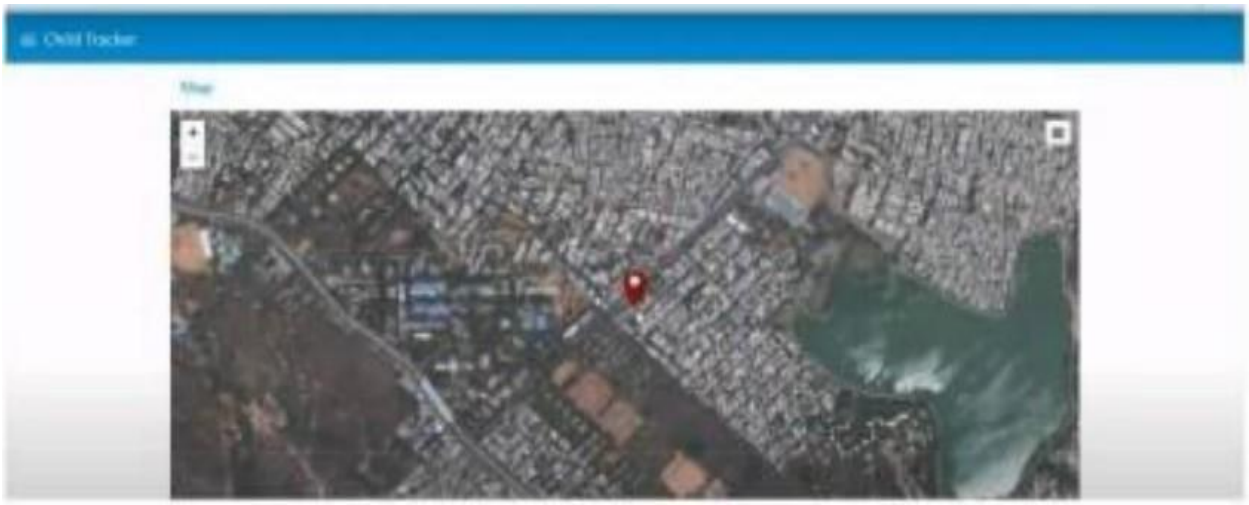
```
import json
import wiotp.sdk.device
import time

myConfig = {
    "identity": {
        "orgId": "hj5fmy",
        "typeId": "NodeMCU",
        "deviceId": "12345"
    },
    "auth": {
        "token": "12345678"
    }
}
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
        name= "Smartbridge"
        #in area location

        latitude= 17.4225176
        longitude= 78.5458042

        #out area location

        #latitude= 17.4219272
        #longitude= 78.5488783
        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(5)

client.disconnect()
```



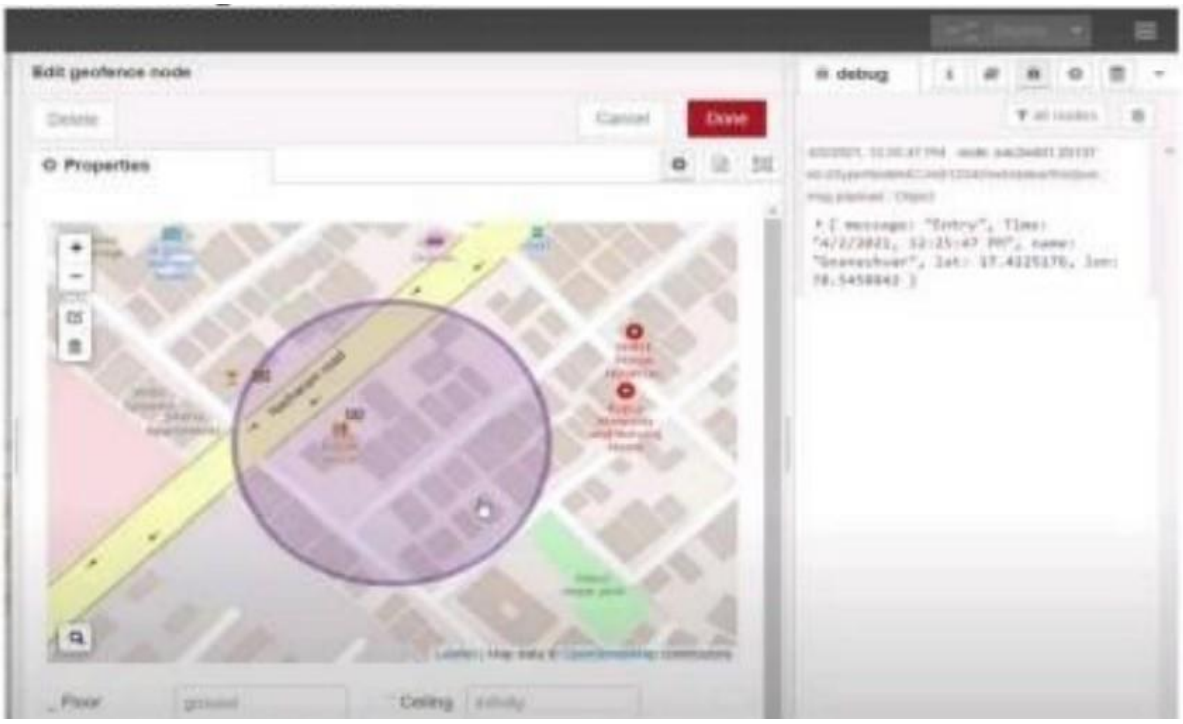
3.EDIT THE HTTP PROTOCOL



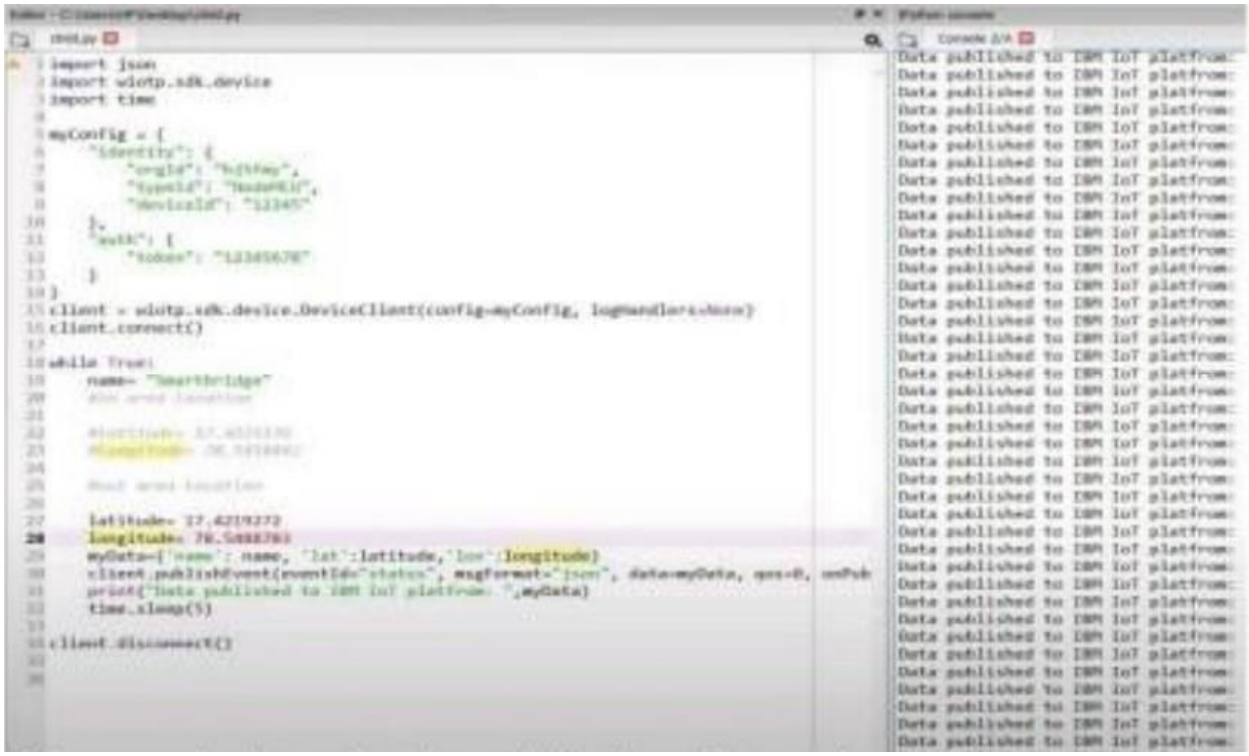
4.LOCATE THE CHILD



5.CREATE THE GEOFENCE NODE



6. PYTHON SENDS REQUEST TO IBM CLOUD



7. AFTER RUNNING THE SCRIPT ,THE WEB UI SHOWS "CHILD IS NOT IN THE LOCATION.

CONCLUSION:

DEVELOP THE WEB APPLICATION USING CODE-RED SUCCESSFULLY.

GITHUB LINK

https://github.com/balarock95

DEMO LINK

https://vimeo.com/776588723