FINAL PROJECT REPORT

HAZARDOUS AREA MONITORING FOR INDUSTRIAL PLANT POWERED BY IOT

Team ID : PNT2022TMID15971

College Name : RMK ENGINEERING COLLEGE

Team Leader : DEVADHARSHINI S

Team Members : ARUNA VR

D SUJITHA

A VISHNUPRIYA

Industry Mentor(s) Name : Bharadwaj

Faculty Mentor(s) Name : Daniel Nareshkumar M

Project Report Format

1. **INTRODUCTION**

1.1 Project Overview

1.2 Purpose

2. **LITERATURE SURVEY**

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. **IDEATION & PROPOSED SOLUTION**

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

4. **REQUIREMENT ANALYSIS**

4.1 Functional requirement

4.2 Non-Functional requirements

5. PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1.INTRODUCTION:

1.1 PROJECT OVERVIEW:

- Through this, we can monitor the temperature parameters of the hazardous areas in industrial plants.
- The area is integrated with smart beacon devices which will be broadcasting the temperature of that particular area.
- Every person working in those areas will be given smart wearable devices which will be acting as beacon scanners.
- Whenever the person goes near the beacon scanners he can view the temperature on his wearable device and if the temperature is high, he will receive the alerts to the mobile through SMS using API.
- Through this wearable device, the data is sent to the cloud and through the dashboard, the admins of that particular plant can view the data and take necessary precautions if required.

1.2 PROJECT FLOW:

- Sending random Humidity and Temperature values will be sent to the IBM IoT platform.

• Sensors values can be viewed in the Web Application.

• Notifies the admin the random values cross the threshold value. To accomplish this, we have to complete all the activities and tasks listed below:

- Create and configure IBM Cloud Services
- Create IBM Watson IoT Platform.
- Create a device & configure the IBM IoT Platform.
- Create Node-RED service.
- Create a database in Cloudant DB to store location data.

• Develop a web Application using Node-RED Service.
o Develop the web application using Node-RED.

• Develop a python script to publish the sensor data to the IBM IoT platform.

2.LITERATURE SURVEY

S NO	TITLE	AUTHORS	ABSTRACT	DRAWBACKS
1	IoTBased Data Logger for Weather Monitorin g Using ArduinoBased Wireless Sensor Networks with Remote Graphical Applicati on and Alerts	Jamal Mabrouki , Mourade Azrour, Driss Dhiba, Yousef Farhaoui, and Souad El Hajjaji	In recent years, monitoring systems play significant roles in our life. So, in this paper, we propose an automatic weather monitoring system that allows having dynamic and real-time climate data of a given area. The proposed system is based on the internet of things technology and embedded system. The system also includes electronic devices, sensors, and wireless technology. The main objective of this system is sensing the climate parameters, such as temperature, humidity, and existence of some gases, based on the sensors. The captured values can then be sent to remote applications or databases. Afterwards, the stored data can be visualized in graphics and tables form.	No information about where we can implement this, just the monitoring thing is explained and done.

2	Design and Validation of a Multifunctional Android-Based Smart Home Control and Monitoring System	LUN-DE LIAO (Member, IEEE), YUHLING WANG YUNGCHUNG TSAO, IJAN WANG, DE-FU JHANG, TSUNGSHENG CHU, CHIA-HUI TSAO, CHIHNING TSAI, SHENG-FU CHEN, CHIUNGCHENG CHUANG, AND	Users often need to control and monitor the environmental variables of their homes, even when they are not at home. In this paper, we present a multifunctional, low-cost, and flexible system for smart home control and environmental monitoring. This system employs an embedded micro web server based on an Arduino Yún microcontroller with Internet connectivity that allows remote device control. The proposed system can be controlled via the Internet through an Android-based mobile app. To guarantee access regardless of Internet availability, the proposed system can also be controlled via standalone manual operation using a touch display. The proposed system transmits sensor data to a cloud platform and can receive commands from the server, allowing many devices to be automatically controlled. To demonstrate the feasibility and effectiveness of this system, devices such as light switches, power plugs, and various sensors, including temperature, gas, 2.5-μm particulate matter (PM2.5) and motion sensors, were integrated into a prototype of the proposed home control system. Finally, we implemented the prototype in a model home to validate the flexibility, scalability, usability, and reliability of the system.	Bounded only to mobile application and there is no web application or SMS for fast notification as we may not have our Internet connections on always.

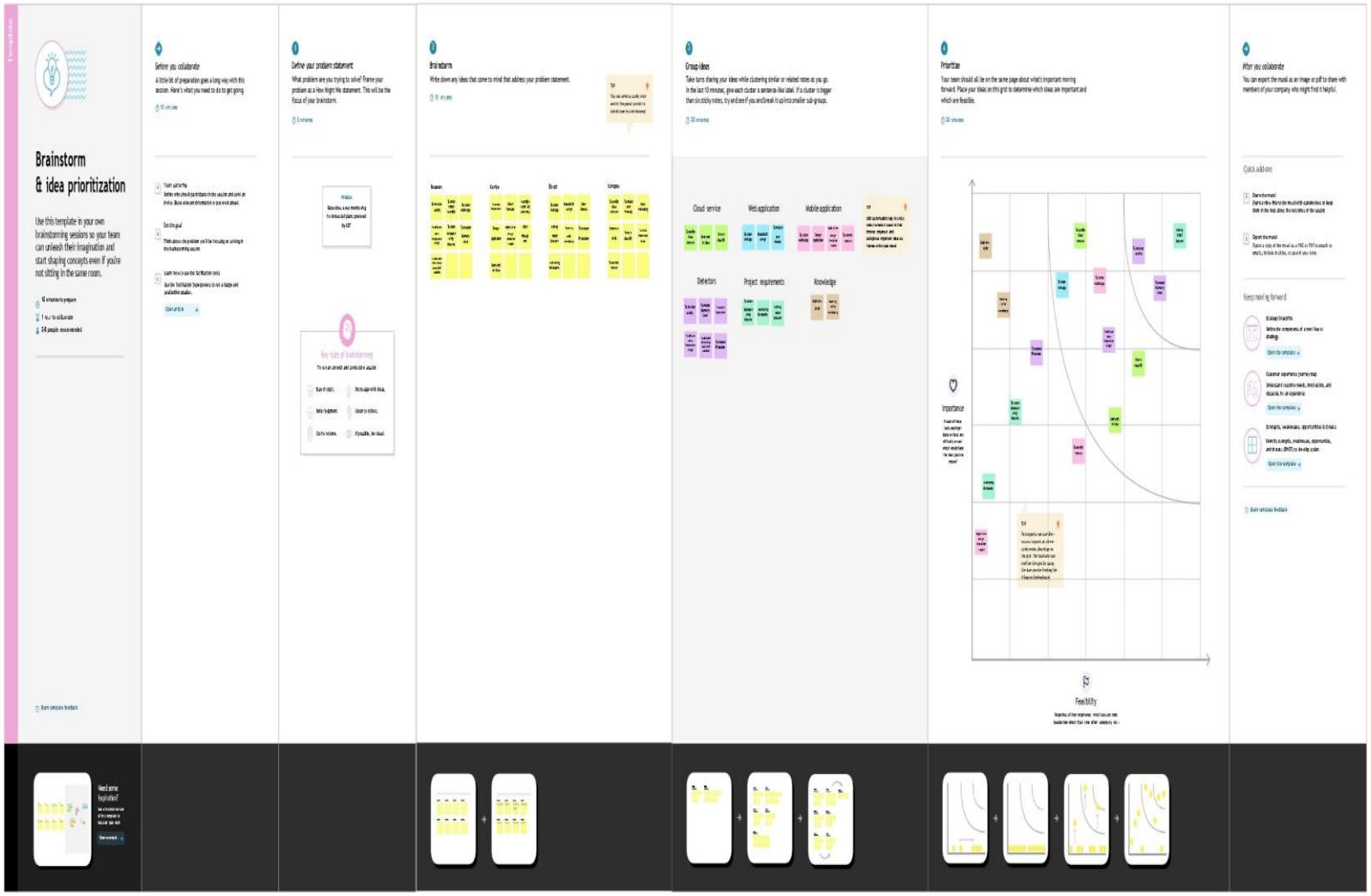
3	Micraspis: A Computer-Aided Proposal Toward Programming and Architecting Smart IoT Wearables	LONG-PHUOC TÔN, LAM-SON LÊ, (Member, IEEE), AND MINH-SON NGUYEN	A wearable is a lightweight body-worn device that relies on data-driven communications to keep people connected purposefully, for instance, for fire-fighting, prompting fast-food clients, and medical treatment. With the rise of wearable computing in the era of IoT-driven smart applications, programmers now expect the time to market for these devices to be shortened. While support for IoT programming in general has gathered traction, tool proposals that automate the development of smart solutions based on the Internet of Wearable Things, though of paramount importance, still stay on the sidelines. We propose a code generation tool called Micraspis that allows a wearable to be described both functionally and architecturally – as if they are two sides of the same coin. The tool has an underlying model-to-code transformation mechanism to generate source code that is executable on a specific IoT programming platform such as Arduino. Our experiments demonstrate that programming code generated by Micraspis amounts to at least 60% of the source code needed to fulfill the business logic of ordinary wearable devices. We conducted an interview to meticulously collect programmers' assessment on how Micraspis assists them in programming and architecting smart IoT wearables. A total of 161 programmers responded to a Likert scale questionnaire, with which at least 65% of them either agree or strongly agree. Overall, the results show that Micraspis has promising applicability in supporting IoWT-enabled smart solutions.	Sole usage of Wearable device only. This can cause limitations as we may not be able to monitor through other means.

4	A Privacy-Preservin g IotBased Fire Detector	ABDULLAH H. ALTOWAIJRI, MOHAMMED S. ALFAIFI, TARIQ A. ALSHAWI, (Member, IEEE), AHMED B. IBRAHIM, AND SALEH A. ALSHEBEILI	Fire detection has been an issue of interest to researchers due to its significant damage to lives and property within a very short time. One of the recent solutions developed to detect fire is to use Internet of Things (IoT) devices equipped with cameras for surveillance. The captured videos of surroundings may be processed by the IoT devices themselves or at the cloud. The latter case is required if the detection algorithm is computationally demanding. However, the use of clouds has a flaw. In fact, using the cloud could pose the threat of having the privacy of a place violated, either through hacking or unauthorized access to the footage of the place where the cloud is installed. In this paper, a fire detection system that preserves the privacy of surroundings, while maintaining a high level of accuracy for fire detection is proposed. The proposed system makes use of the cloud for fire detection; and that is achieved by sending to the cloud features extracted from the video captured by the IoT device, instead of sending the actual footage. Binary video descriptors and Convolutional Neural Network (CNN) have been used to develop the fire detection algorithm. The video descriptors are used to extract features, while CNN is used for classification. Videos with real fire and non-fire scenes have been used in this development. Results show that the performance of proposed fire detection algorithm can achieve 97.5% classification accuracy, that outperforms the state-ofthe art algorithms which make direct use of raw videos. Therefore, the proposed fire detector is as reliable as other available systems, with the advantage of having a privacy-preserving capability. It is also demonstrated that the proposed video descriptors can be implemented for real-time processing using an IoT device, Raspberry Pi 4 platform, with an average processing speed of 100ms per frame, which satisfies practical needs.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP

3.2 IDEATION & BRAINSTORMING



3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	➢ Develop an efficient system & an application that can monitor and alert the users(workers)
2.	Idea / Solution description	➢ This work describes a smart monitoring system for the detection of flammable gas residues, toxic gases and reduced oxygen concentration. ➢ The proposed system aims at reducing the risk of fires and explosions, thus increasing the safety of workers engaged in maintenance or inspection of gas storages.
3.	Novelty / Uniqueness	➢ Fastest alerts to the workers ➢ User friendly ➢ Monitored from time to time
4.	Social Impact / Customer Satisfaction	➢ Cost efficient ➢ Better workability ➢ Gives effective results
5.	Business Model (Revenue Model)	➢ The product is advertised all over the platforms. Since it is economical, even helps small scale industries from disasters. ➢ As the product usage can be understood by everyone, it is easy for them to use it properly for their safest organization

6.	Scalability of the Solution	➢ Since the product is cost efficient, it can be placed in many places in the industries. ➢ Even when the problem occurs, the product senses and alerts the workers effectively

3.4 PROBLEM SOLUTION FIT

Define CS, fit into CL

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
eg. working parents of 0-5 y.o. kids

6. CUSTOMER LIMITATIONS EG. BUDGET, DEVICES CL

What limits your customers to act when problem occurs?
Spending power, budget, no cash in the pocket? Network connection?
Available devices?

5. AVAILABLE SOLUTIONS PLUSES & MINUSES AS

Which solutions are available to the customer when he/she is facing the problem? What had he/she tried in the past? Pluses & minuses?

Explore AS, differentiate

Focus on PR, tap into BE, understand RC

2. PROBLEMS / PAINS + ITS FREQUENCY PR

Which problem do you solve for your customer?
There could be more than one, explore different sides.
eg. existing solar solutions for private houses are not considered a good investment (1).

How often does this problem occur?

9. PROBLEM ROOT / CAUSE RC

What is the root of every problem from the list?
eg. People think that solar panels are bad investment right now, because they are too expensive (1.1), and possible changes to the law might influence the return of investment significantly and diminish the benefits (1.2).

7. BEHAVIOR + ITS INTENSITY BE

What does your customer do about / around / directly or indirectly related to the problem?
eg. directly related: tries different "green energy" calculators in search for the best deal (1.1), usually chooses for 100% green provider (1.2).
indirectly related: volunteering work (Greenpeace etc)

How often does this related behavior happen?

Focus on PR, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TO ACT TR

What triggers customer to act?
eg. seeing their neighbor installing solar panels (1.1), reading about innovative, more beautiful and efficient solution (1.2)

4. EMOTIONS BEFORE / AFTER EM

Which emotions do people feel before/after this problem is solved?
Use it in your communication strategy.
eg. frustration, blocking (can't afford it) > boost, feeling smart, be an example for others (made a smart purchase)

10. YOUR SOLUTION SL

If you are working on existing business - write down existing solution first, fill in the canvas and check how much does it fit reality.

If you are working on a new business proposition then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour .

8. CHANNELS of BEHAVIOR CH

ONLINE

Extract channels from Behavior block

OFFLINE

Extract channels from Behavior block and use for customer development

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements:

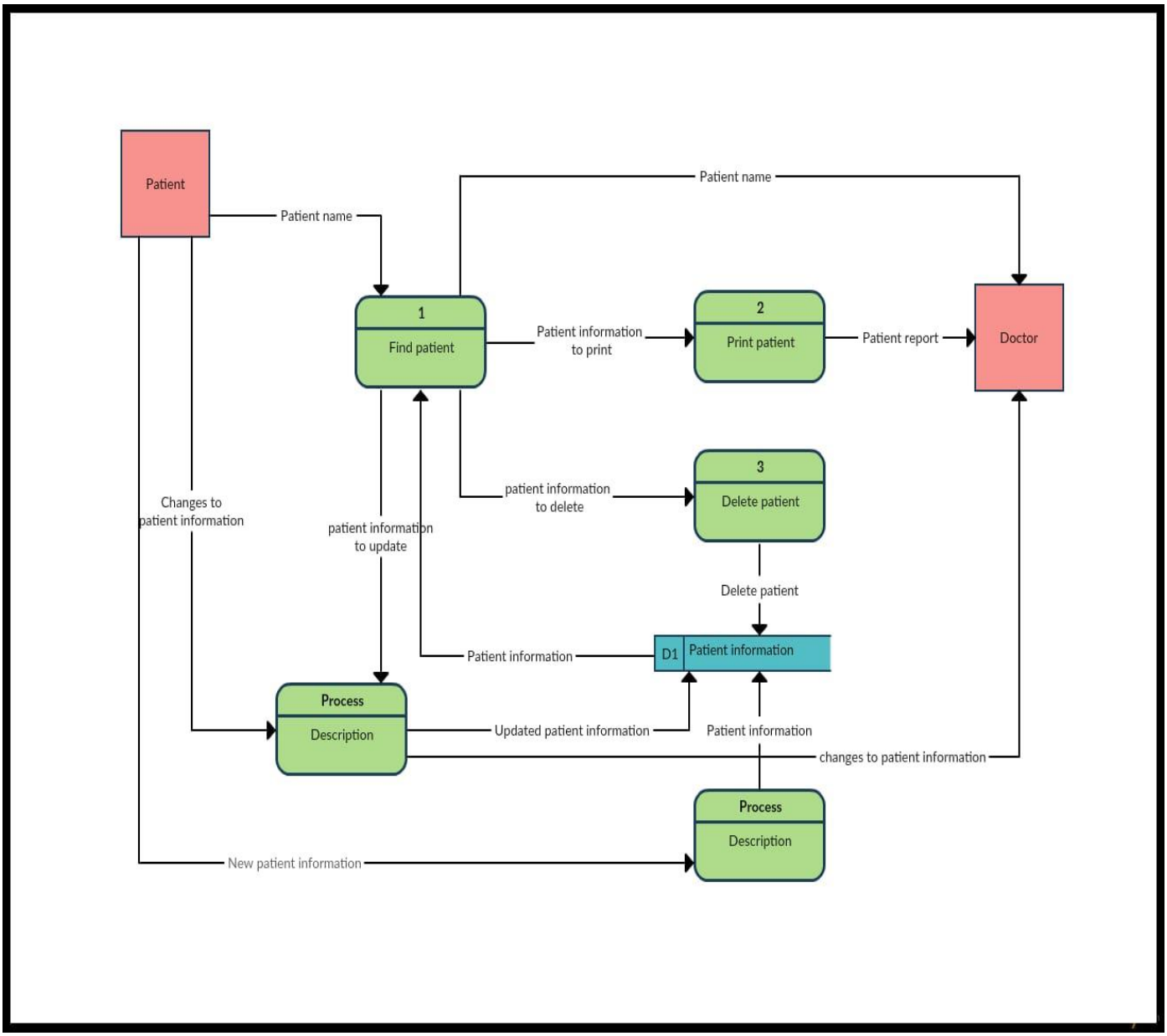
FR No.	Functional Requirement(Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Data Gathering	The smart beacon must be able to detect and the temperature of a particular area in real.
FR-2	Location Detection	The smart beacon must be able to detectwhen a wearable device has entered an area near it.
FR-3	Beacon Data Syncing	The smart beacon must be able to share its stored data with both the wearable device and admin dashboard through the cloud.
FR-4	Wearable Device Display	The wearable device must be able to display the temperature of the area where theworker is currently present.
FR-5	SMS Notification	If the temperature of the area is found to reach dangerous levels, the worker should be informed via SMS to their phone instructing them to leave the area.
FR-6	Admin Dashboard	If the temperature of the area is found to reach dangerous levels the admin is informed via the dashboard and must take the necessary precautions.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The wearable device should be slim and not annoy or disturb the workers who arewearing them. They should also reliably display the temperature without large delays and notifications should be clear in cases of detected danger.
NFR-2	**Security**	The connection of the beacons to the cloudand wearable devices should be secure. The security of the database housing allthe temperature data should also be bolstered.
NFR-3	**Reliability**	The wearable device should be able tofunction without any faults even at dangerous temperatures. If a fault is detected it should notify theuser and the admin to be immediately repaired and replaced. The beacons should also be regularly maintained to ensure reliability.
NFR-4	**Performance**	The device should update temperature readings in real time and requires high endsensors and processors to do so. The time to send data to the cloud and other devices should also be made as smallas possible.
NFR-5	**Availability**	The user should be able to check the temperature of the area no matter whereorat what time they are in the plant. The dashboard should be constantly

		activesoas to ensure safety precautions can be executed whenever danger is detected.
NFR-6	**Scalability**	If the area that needs to be monitored needs to be increased all one has to do isinstall new smart beacon devices and connect them to the same system as the previous beacons. It can also be replicated in different plantswith different factors to be monitored giving it highly scalability.

5.PROJECT DESIGN:

5.1 Data Flow Diagrams:

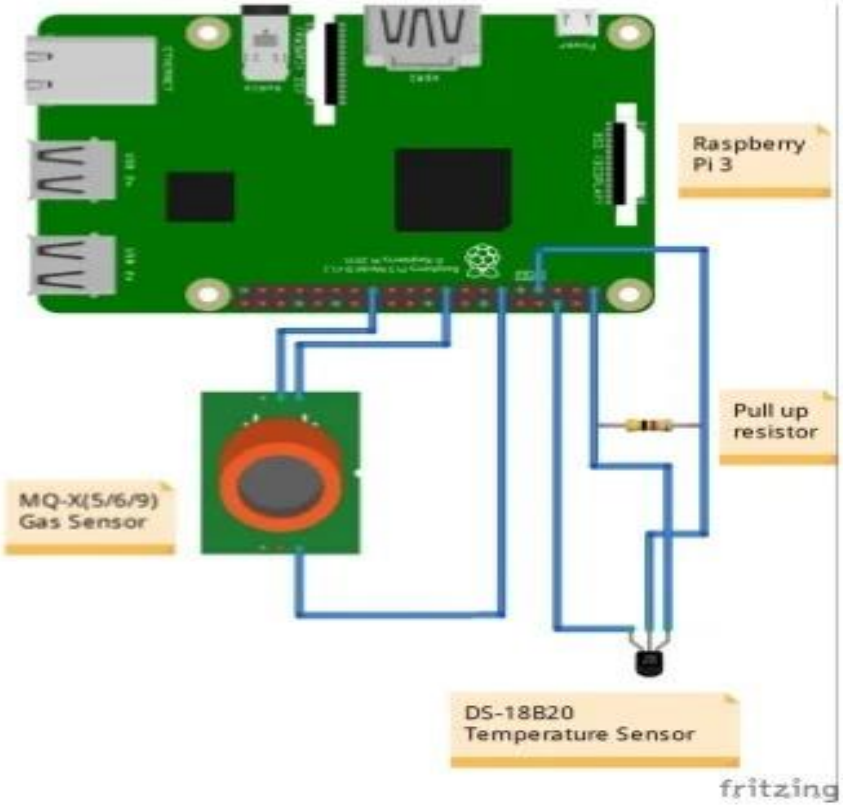


USER STORIES

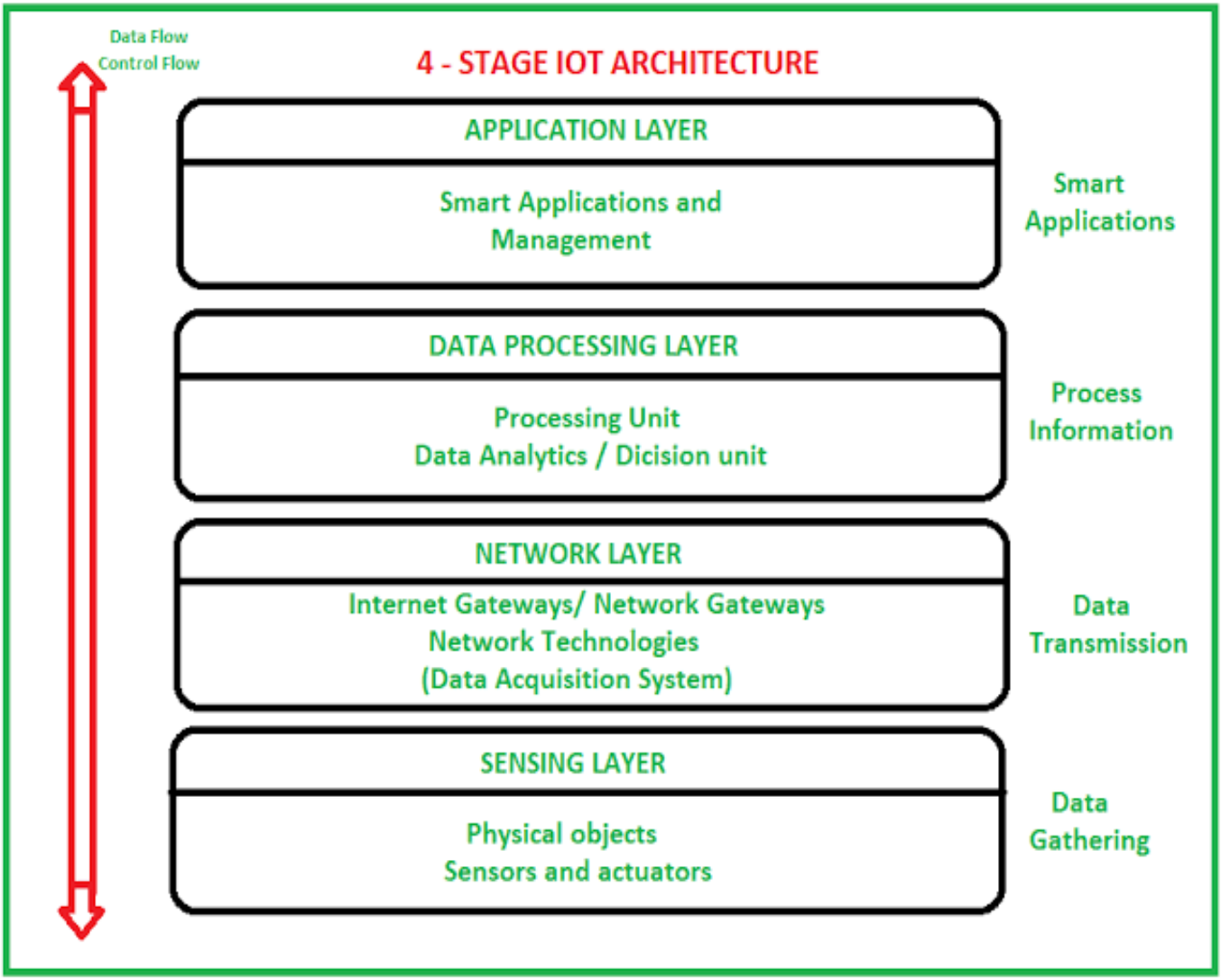
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Industrial Owner)	Registration	USN-1	As an Industrial Owner, I can register into the application by entering email & password	I can access my account / dashboard	High	Sprint-1
	Data Modules	USN-2	As an Industrial Owner, I can get message about the temperature and humidity	I can receive confirmation email & click confirm	High	Sprint-1

	Login	USN-3	As an industrial Owner, I can login into my account through email and Password	I can access my account	Medium	Sprint-2
	Dashboard	USN-4	As an Industrial Owner, I can monitor of temperature	I can access the dashboard with individual Login id/password	High	Sprint-1
Customer	Registration	USN-1	As an Industrial	I can access my	High	Sprint-1
(Industrial Worker)			Worker, I can register into the application by entering email & password	account / dashboard		
	Data Modules	USN-2	As an Industrial Worker, I can get message about the temperature and humidity	I can receive confirmation email & click confirm	High	Sprint-1
	Login	USN-3	As an industrial Owner, I can login into my account through email and Password	I can access my account	Medium	Sprint-2

6 Circuit Diagram and Layout



5.2 Technical Architecture:



REFERENCES

https://www.google.com/url?sa=t&source=web&rct=j&url=https://partheniu
mp
rojects.com/hazardous-area-monitoring-for-
industrialplants/&ved=2ahUKEwikhTo3f76AhVgyHMBHYa2At4QFnoECBI
QAQ&usg=AOvVaw1ram1VVkt1RmZ
https://www.google.com/url?sa=t&source=web&rct=j&url=https://partheniu
mprojects
.com/hazardous-area-monitoring-for-
industrialplants/%23:~:text%3DEvery%2520device%2520will%2520be%252
0acting,acting%25
20as%2520a%2520beacon%2520scanner.&ved=2ahUK
EwiTpaPt3f76AhWwHbcAHUsXDoYQFnoECEkQBQ&usg=AOvVaw1ram1
VVkt1RmZ7
DfaIiiPK
https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.ijser.
org/resea
rchpaper/Internet-of-Things-for-Industrial-Monitoring- and-
ControlApplications.pdf&ved=2ahUKEwiJ0Zi43v76AhUJGrcAHfM1BXsQF
noECDwQAQ&
usg=AOvVaw38Hy6dTeMJVE5yX5S-VzYW
https://www.rejigdigital.com/blog/iiot-changing-condition-monitoring-
forindustries/#:~:text=IIoT%20can%20intelligently%20monitor%20various,
offshore%20
drilling%20rigs%20or%20pipelines

6.PROJECT PLANNING & SCHEDULING

6.1 PREPARE MILESTONE AND ACTIVITY LIST

Proposed Solution	Proposed solution shows the current solution and it helps is going towards the desired result until it is achieved.	18 September 2022
Solution Architecture	Solution Architecture is a very complex process I.e. it has a lot of sub- processes and branches. It helps in understanding the components and features to complete our project.	29 September 2022
Customer Journey	It helps us to analyse from the perspective of a customer, who uses our project.	9 October 2022
Functional Requirement	Here functional and nonfunctional requirements are briefed. It has specific features like usability, security, reliability, performance, availability, and scalability.	16 October 2022
Data Flow Diagrams	Data Flow Diagram is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement.	14 October 2022

Prepare Milestone & Activity List	It helps us to understand and evaluate our own progress and accuracy so far.	29 October 2022
Spring Delivery Plan	Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved.	14 ovember 2022
Technology Architecture	Technology Architecture is a more well defined version of solution architecture. It helps us analyze and understand various technologies that needs to be implemented in the project.	15 October 2022

6.2 Sprint Delivery schedule

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Installation of Beacons	USN-1	First the Admin will be installing smart beacons at necessary places.	15	High	A.Vishnupriya Devadarshini.S D.Sujitha Aruna . VR
Sprint-1	Providing Wearables	USN-1	The Admin will be providing everyone at the Industry a wearable device.	5	Medium	A.Vishnupriya Devadarshini.S D.Sujitha Aruna . VR
Sprint-2	Cloud Setup	USN-2	The smart Beacons will connect with the cloud services. Where we can get the realtime data from the wearable	20	High	A.Vishnupriya Devadarshini.S D.Sujitha Aruna . VR \
Sprint-3	Online Monitoring via Web	USN-3	Websites will be created and connected with the cloud services.	20	High	A.Vishnupriya Devadarshini.S D.Sujitha Aruna . VR

Sprint-4	Monitoring via Mobile	USN-4	Mobile Application will be created and fast sms will be used to alert abnormality to the user.	20	High	A.Vishnupriya Devadarshini.S D.Sujitha Aruna . VR

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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		6 Days	24 Oct 2022	28 Oct 2022		29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	04 Nov 2022		05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	11 Nov 2022		12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	18 Nov 2022		19 Nov 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

CODING AND SOLUTIONS

SPRINT-1

CODE:

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

WiFiClient wifiClient;

#define ORG "mxyrim"
#define DEVICE_TYPE "NodeMCU"
#define DEVICE_ID "12345"
#define TOKEN "12345678"
#define speed 0.034

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char topic[] = "iot-2/cmd/home/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
PubSubClient client(server, 1883, wifiClient);
void publishData();

const int trigpin=5;
const int echopin=18;
String command;
String data="";
String lat="13.356563";
String lon="80.141428";
```

```
String name="point1";
String icon="fa-fire";

long duration;
int dist;

void setup()
{
  Serial.begin(115200);
  pinMode(trigpin, OUTPUT);
  pinMode(echopin, INPUT);
  wifiConnect();
  mqttConnect();
}

void loop() {

  publishData();
  delay(500);

  if (!client.loop()) {
    mqttConnect();
  }
}

void wifiConnect() {
  Serial.print("Connecting to "); Serial.print("Wifi");
  WiFi.begin("Wokwi-GUEST", "", 6);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.print("WiFi connected, IP address: ");
Serial.println(WiFi.localIP());
}

void mqttConnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting MQTT client to "); Serial.println(server);
    while (!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(1000);
    }
    initManagedDevice();
    Serial.println();
  }
}

void initManagedDevice() {
  if (client.subscribe(topic)) {
```

```
     Serial.println(client.subscribe(topic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}
void publishData()
{
  digitalWrite(trigpin,LOW);
  digitalWrite(trigpin,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin,LOW);
  duration=pulseIn(echopin,HIGH);
  dist=duration*speed/2;
  dist=dist/4;
  dist=100-dist;
  if(dist>80){
    lat="13.356563";
    lon="80.141428";
  }else{
    lat="0.000000";
    lon="0.000000";
  }
  DynamicJsonDocument doc(1024);
  String payload;
  doc["Name"]=name;
  doc["Latitude"]=lat;
  doc["Longitude"]=lon;
  doc["Icon"]=icon;
  doc["GasPercent"]=dist;
  serializeJson(doc, payload);
  delay(3000);
  Serial.print("\n");
  Serial.print("Sending payload: ");
  Serial.println(payload);
  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish OK");
  } else {
    Serial.println("Publish FAILED");
  }
}
```

SPRINT-2

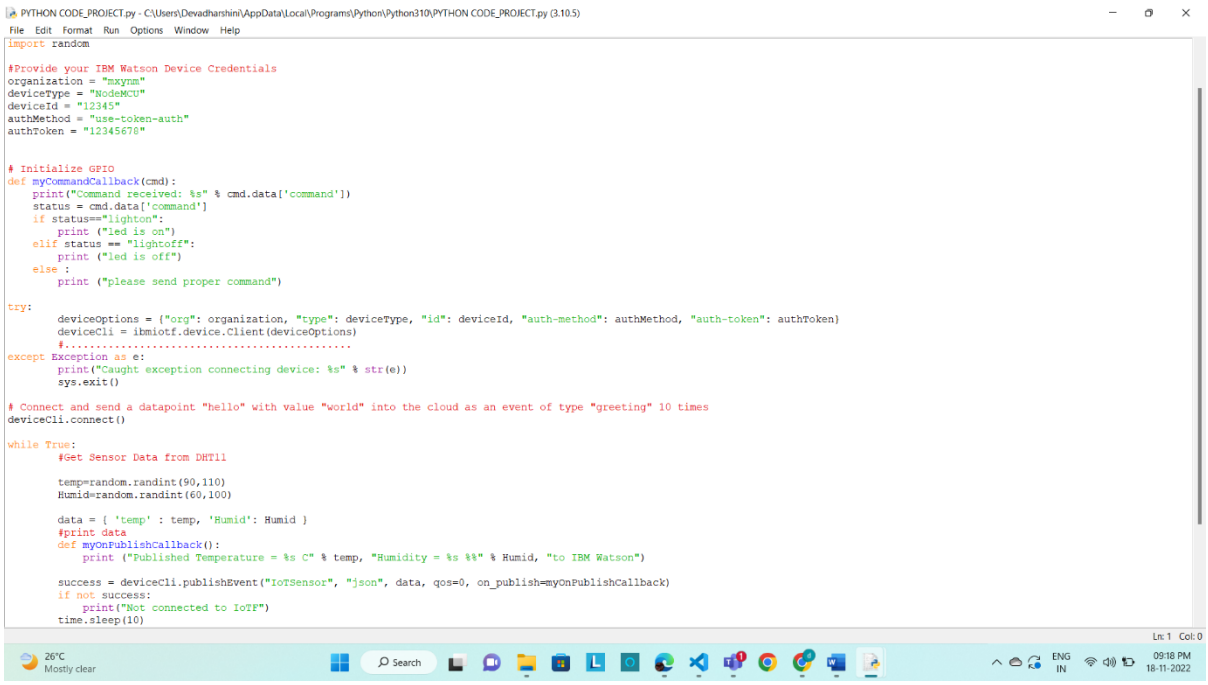
```
import time

import sys

import ibmiotf.application

import ibmiotf.device

import random



#Provide your IBM Watson Device Credentials

organization = "mxynm"

deviceType = "NodeMCU"

deviceId = "12345"

authMethod = "use-token-auth"

authToken = "12345678"



# Initialize GPIO

def myCommandCallback(cmd):

    print("Command received: %s" % cmd.data['command'])

    status = cmd.data['command']

    if status=="lighton":
```

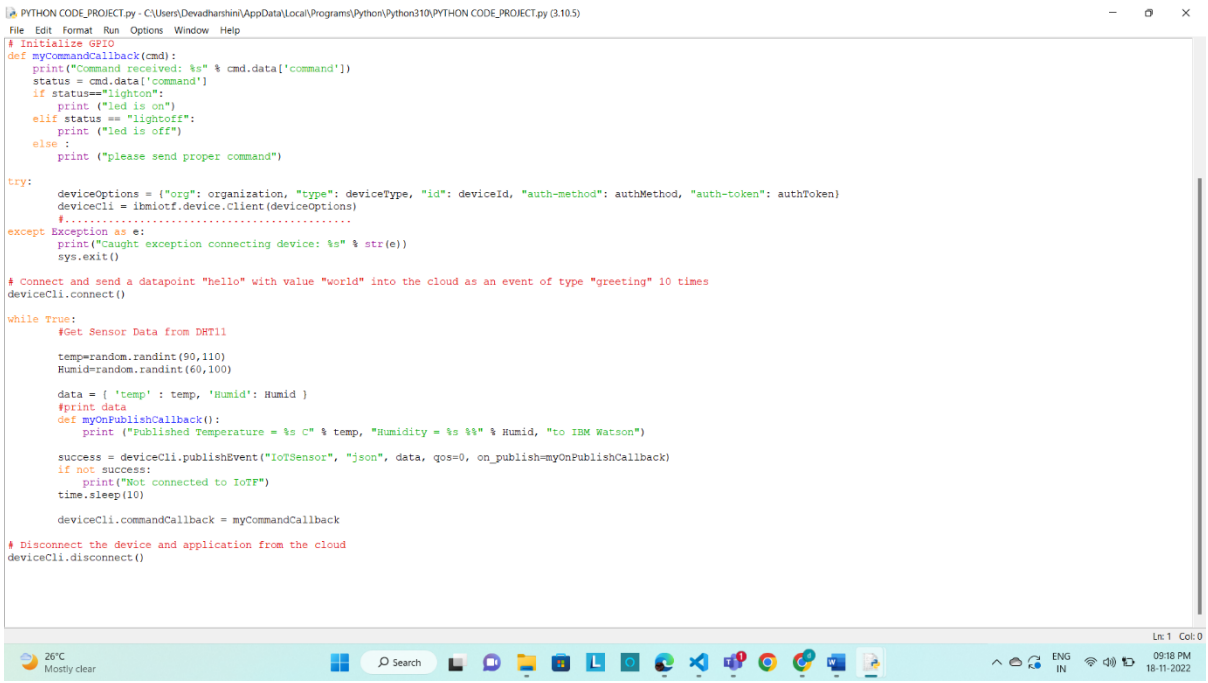
```
            print ("led is on")
        elif status == "lightoff":
            print ("led is off")
        else :
            print ("please send proper command")

    try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
        deviceCli = ibmiotf.device.Client(deviceOptions)
        #.....
    except Exception 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 of type "greeting" 10 times
    deviceCli.connect()

    while True:
```

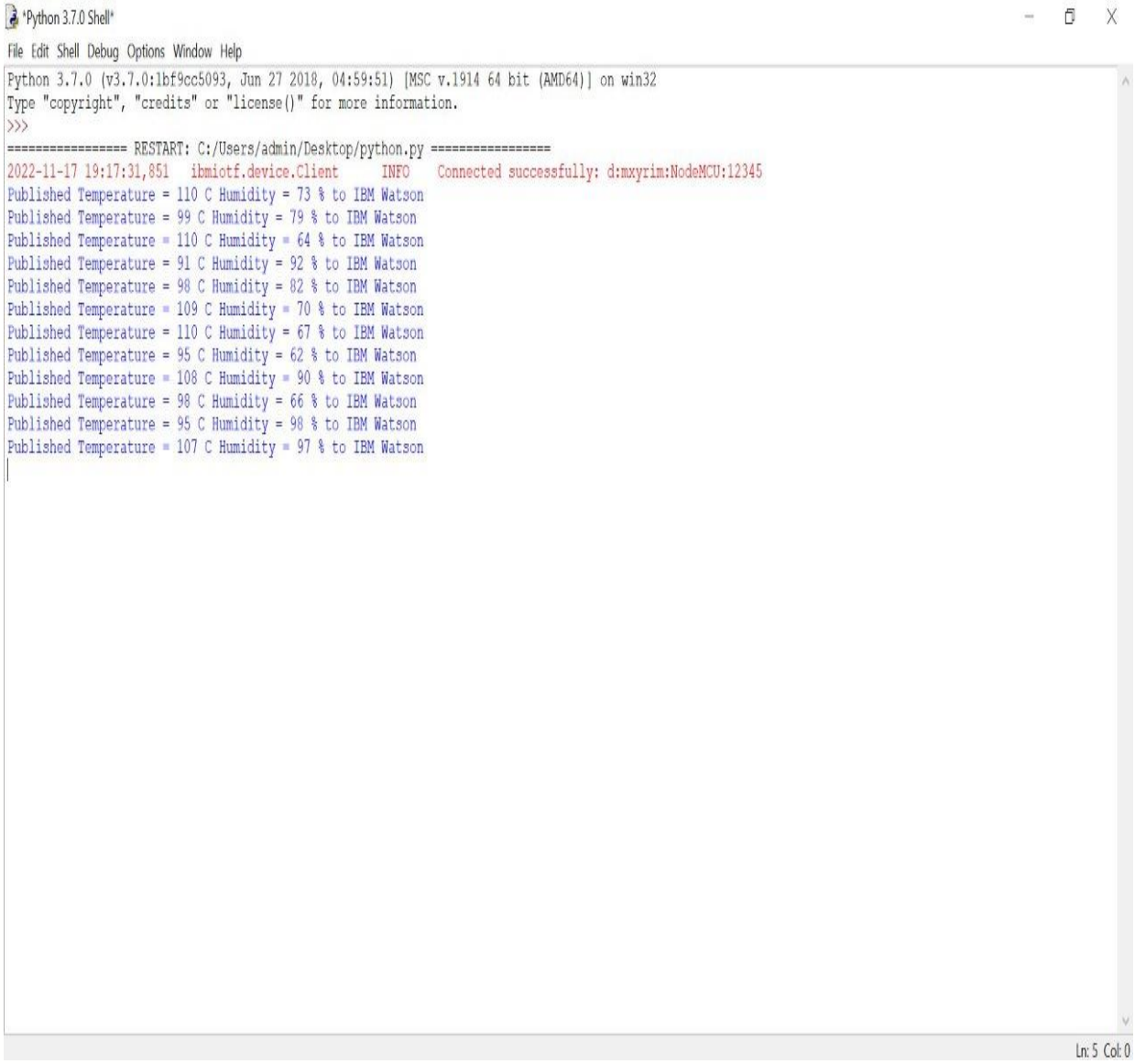
```
        #Get Sensor Data from DHT11


        temp=random.randint(90,110)

        Humid=random.randint(60,100)


        data = { 'temp' : temp, 'Humid': Humid }

        #print data

        def myOnPublishCallback():

            print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % Humid, "to IBM
Watson")


        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)

        if not success:

            print("Not connected to IoTF")

        time.sleep(10)


        deviceCli.commandCallback = myCommandCallback


    # Disconnect the device and application from the cloud
```

deviceCli.disconnect()

SPRINT-3






Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/admin/Desktop/python.py ==========
2022-11-17 19:17:31,851 ibmiotf.device.Client INFO Connected successfully: d:mxyrim:NodeMCU:12345
Published Temperature = 110 C Humidity = 73 % to IBM Watson
Published Temperature = 99 C Humidity = 79 % to IBM Watson
Published Temperature = 110 C Humidity = 64 % to IBM Watson
Published Temperature = 91 C Humidity = 92 % to IBM Watson
Published Temperature = 98 C Humidity = 82 % to IBM Watson
Published Temperature = 109 C Humidity = 70 % to IBM Watson
Published Temperature = 110 C Humidity = 67 % to IBM Watson
Published Temperature = 95 C Humidity = 62 % to IBM Watson
Published Temperature = 108 C Humidity = 90 % to IBM Watson
Published Temperature = 98 C Humidity = 66 % to IBM Watson
Published Temperature = 95 C Humidity = 98 % to IBM Watson
Published Temperature = 107 C Humidity = 97 % to IBM Watson
Ln: 5 Col: 0

SPRINT-4

```
import time

import sys

import ibmiotf.application

import ibmiotf.device

import random



#Provide your IBM Watson Device Credentials

organization = "mxynm"

deviceType = "NodeMCU"

deviceId = "12345"

authMethod = "use-token-auth"

authToken = "12345678"




# Initialize GPIO

def myCommandCallback(cmd):

    print("Command received: %s" % cmd.data['command'])

    status = cmd.data['command']
```

```
        if status=="lighton":
            print ("led is on")
        elif status == "lightoff":
            print ("led is off")
        else :
            print ("please send proper command")


    try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
        deviceCli = ibmiotf.device.Client(deviceOptions)
        #.....
    except Exception 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 of type "greeting" 10 times
    deviceCli.connect()
```

```
while True:

        #Get Sensor Data from DHT11



        temp=random.randint(90,110)

        Humid=random.randint(60,100)



        data = { 'temp' : temp, 'Humid': Humid }

        #print data

        def myOnPublishCallback():

           print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % Humid, "to IBM
Watson")



        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)

        if not success:

           print("Not connected to IoTF")

        time.sleep(10)



        deviceCli.commandCallback = myCommandCallback
```

Disconnect the device and application from the cloud

deviceCli.disconnect()

7. CODING & SOLUTIONING

7.1 FEATURE 1

STEP1:

1. Find the temperature and humidity of the particular region where hazardous activities occur in industrial power plant.

2. Connect it to IBM waston IOT platform

3. Get the output in recent events.

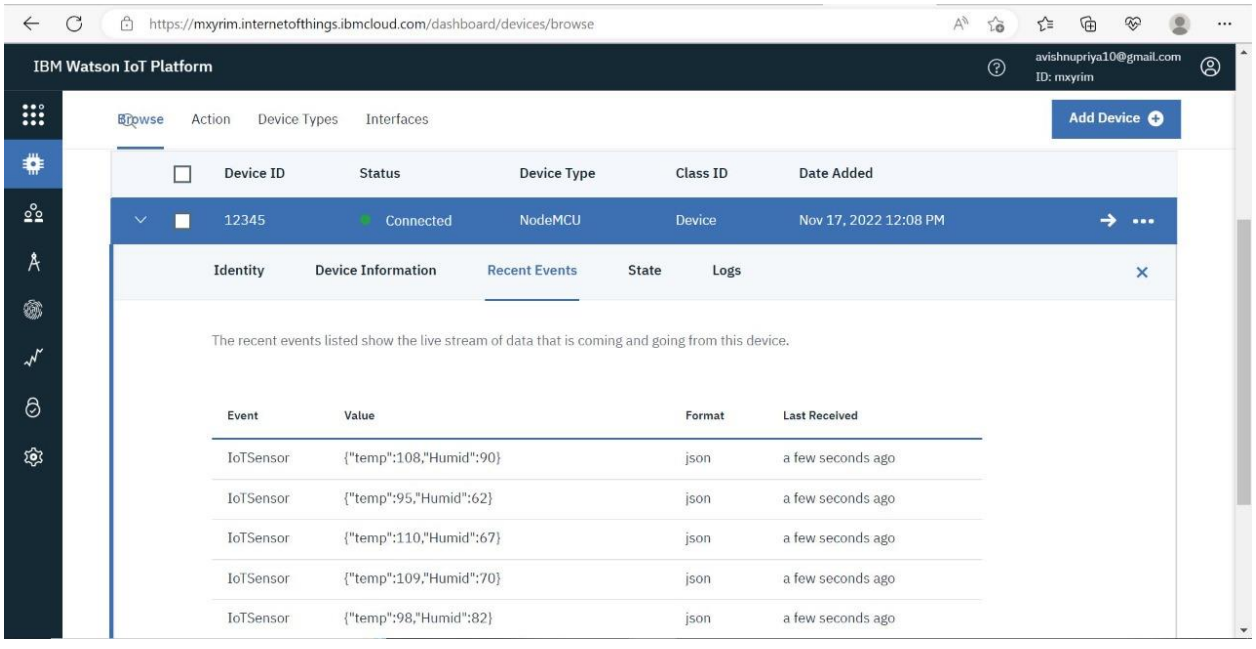
```
        data = { 'temp' : temp, 'Humid': Humid }
        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % Humid, "to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(10)

        deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

Ln: 8 Col: 22

Step 2:

1. Output displayed on IBM IOT platform

2. By changing the user details we publish our code to IBM IOT Watson platform and our results are displayed in recent events



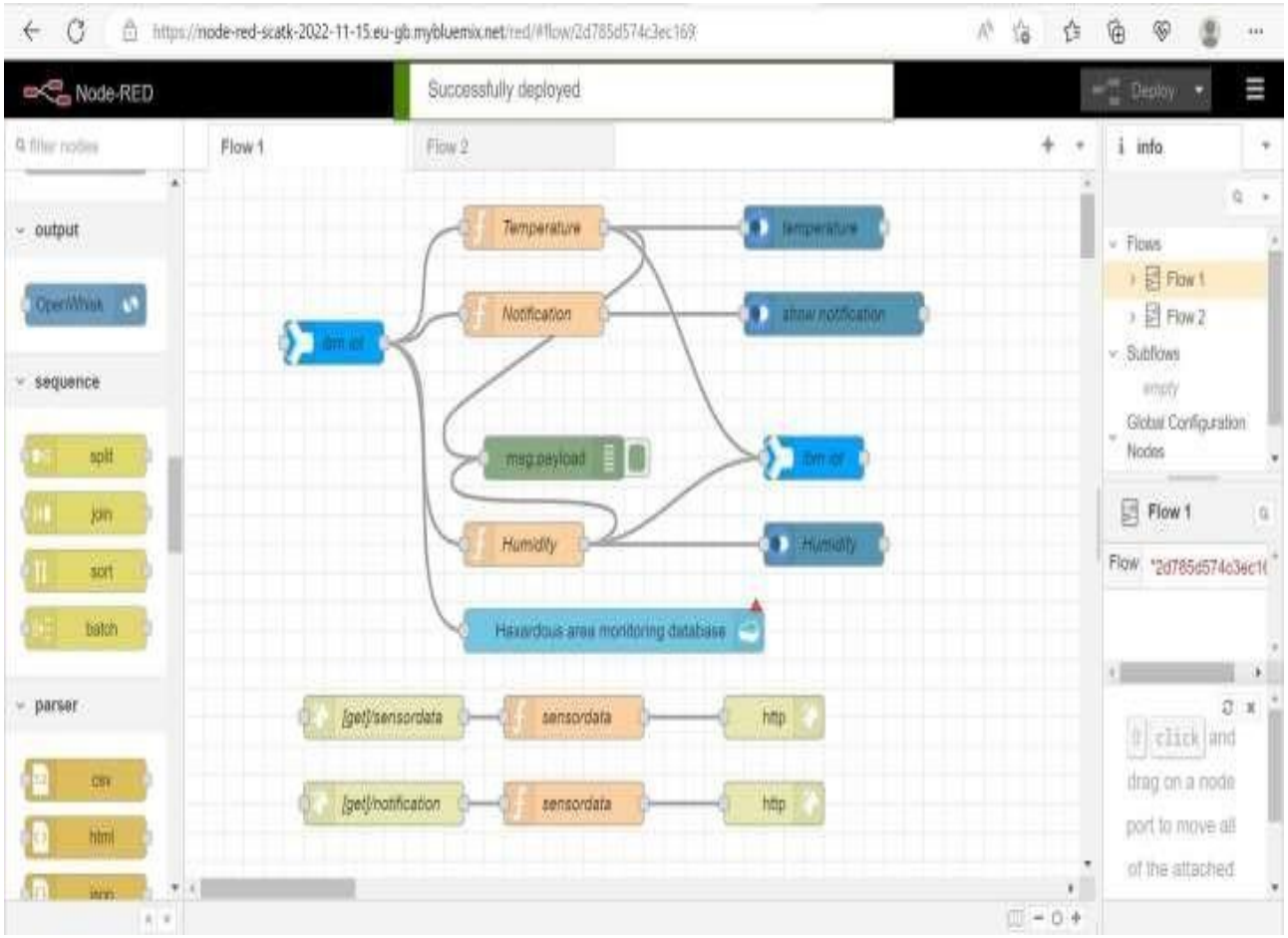
7.2 FEATURE 2

STEP 1:

1. Create node red service

2. Perform necessary connections to receive desired output.

3. Connect the node red to IBM waston IOT Platform. 4. Get the output in recent events



STEP 2:

1. Connect our device to mobile app

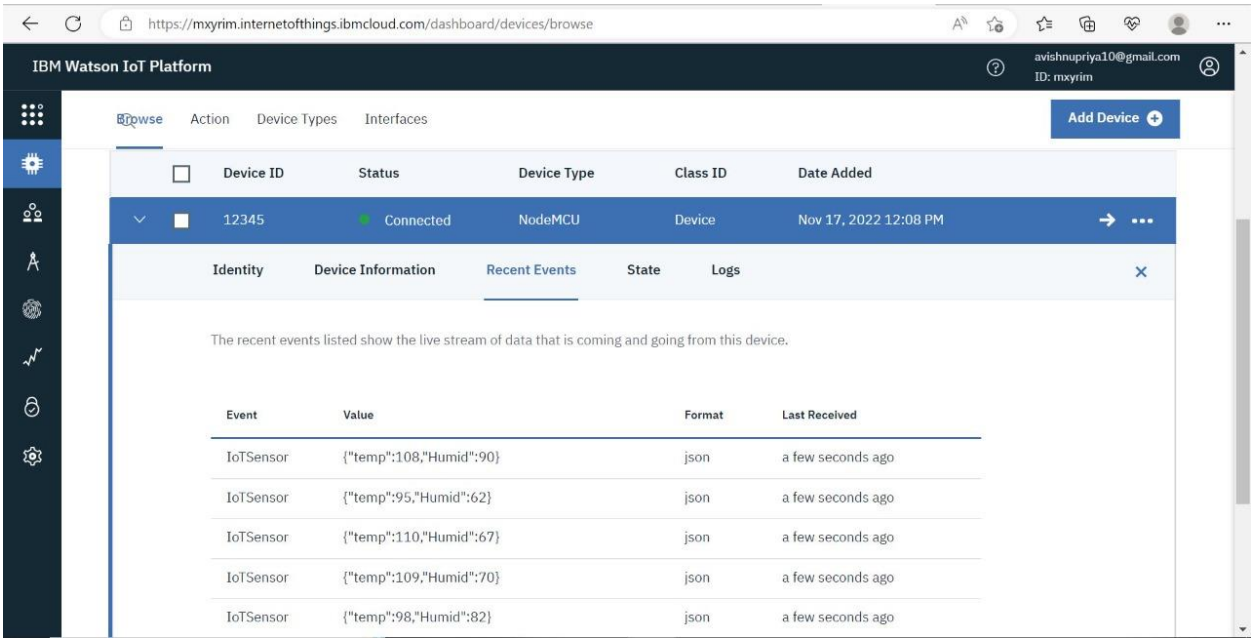
2. Get Output in the output screen

```
Command received: motoron
motor in on
Published Temperature = 100 C Humidity:68
Published Temperature = 63 C Humidity:7
Published Temperature = 32 C Humidity:67
Command received: motoroff
motor is off
```

8.TESTING:

8.1 TEST CASE

```
SD card is initialized. Ready to go
Time,Humidity,Temperature_C,Temperature_F,Heat_index
0:0:0,  40.00,    24.00,       75.20,      74.30
0:0:2,  40.00,    24.00,       75.20,      74.30
0:0:4,  40.00,    24.00,       75.20,      74.30
0:0:6,  40.00,    24.00,       75.20,      74.30
0:0:8,  40.00,    24.00,       75.20,      74.30
0:0:10,  40.00,    24.00,       75.20,      74.30
0:0:12,  40.00,    24.00,       75.20,      74.30
0:0:14,  40.00,    24.00,       75.20,      74.30
0:0:16,  40.00,    24.00,       75.20,      74.30
0:0:18,  40.00,    24.00,       75.20,      74.30
0:0:20,  40.00,    24.00,       75.20,      74.30
0:0:22,  40.00,    24.00,       75.20,      74.30
0:0:24,  40.00,    24.00,       75.20,      74.30
0:0:26,  40.00,    24.00,       75.20,      74.30
0:0:28,  40.00,    24.00,       75.20,      74.30
0:0:30,  40.00,    24.00,       75.20,      74.30
```



8.2 USER ACCEPTANCE TESTING PURPOSE OF DOCUMENT

The purpose of this document is to briefly explain the test coverage and open issues of the Hazardous area monitoring for industrial plant powered by IOT project at the time of the release to User Acceptance Testing (UAT).

ADVANTAGES AND DISADVANTAGES:

ADVANTAGES:

• With the help of watch tower wide range of area can be monitored easily.

• It provides security to the people working in the area with the help of cloud server and sensor is used to note the temperature change.

• Improved route performance. The wearables are more advanced and customizable to ones need

. • This system is flexible, it is user friendly and affordable.

• Provides safety to all the employees working and provides smart reliability overlay to overall system.

DISADVANTAGES:

• Maintenance cost of the system is high, replacing the defected components can crash the system.

• There can be a delay in sending alerts to the mobile through SMS using API.

• The carelessness of the workers in that particular plant, without viewing the data they cannot take necessary actions which leads to an accident.

CONCLUSION :

We hope to gain hands-on experience with the trending technologies of "Embedded System" and "Internet of Things" through this project. IoT-enabled industrial monitoring systems have become increasingly popular in a variety of industries because they improve safety standards by providing real-time monitoring of critical parameters such as temperature, humidity, and smoke, as well as alerting officials and workers regularly. The implementation is not only for safety reasons, but it also has the potential to increase industry yields. In our project, the Internet of Things (IoT) is used to collect data and communicate through the internet. We hope that our project will be beneficial enough to be implemented in industries across India, saving lives and property from accidents and risks that are often overlooked by industry personnel and users. Companies in the industrial and logistics sectors can better meet the new era of instant needs by utilizing the Industrial Internet of Things (IoT).

APPENDIX

SOURCE CODE:

```
import time

import sys

import ibmiotf.application

import ibmiotf.device

import random



#Provide your IBM Watson Device Credentials

organization = "mxynm"

deviceType = "NodeMCU"

deviceId = "12345"

authMethod = "use-token-auth"

authToken = "12345678"



# Initialize GPIO

def myCommandCallback(cmd):
```

```
        print("Command received: %s" % cmd.data['command'])
        status = cmd.data['command']
        if status=="lighton":
            print ("led is on")
        elif status == "lightoff":
            print ("led is off")
        else :
            print ("please send proper command")

    try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
        deviceCli = ibmiotf.device.Client(deviceOptions)
        #.....
    except Exception 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 of type "greeting" 10 times
```

```
deviceCli.connect()



while True:

        #Get Sensor Data from DHT11



        temp=random.randint(90,110)

        Humid=random.randint(60,100)



        data = { 'temp' : temp, 'Humid': Humid }

        #print data

        def myOnPublishCallback():

            print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % Humid, "to IBM
Watson")



        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)

        if not success:

            print("Not connected to IoTF")

        time.sleep(10)
```

```
        deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud

deviceCli.disconnect()
```

GITHUB AND DEMOLINK:

GITHUB:

https://github.com/IBM-EPBL/IBM-Project-30544-1660148313/upload/main

VIDEO LINK:

https://screenrec.com/share/iSVZdbgyos