



LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT

Monitoring and Controlling the Crops in Warehouse using IOT

UE17CS490B – Capstone Project Phase – 2

Submitted by:

Arumilli Meghana	PES1201701279
Chaitra Vishnu	PES1201802403
Devanga	PES1201802436
Anusha T N	

Under the guidance of

Prof. Rachana B S

Designation
PES University

January - May 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100ft Ring Road, Bengaluru – 560 085, Karnataka, India

TABLE OF CONTENTS

1. Introduction	4
1.1 Overview	4
1.2 Purpose	4
1.3 Scope	5
2. Design Constraints, Assumptions	5
3. Design Description	5
3.1 Master Class Diagram	6
3.2 Module Name	6
3.3 Module 1	6
3.3.1 Description	6
3.4 Module 2	6
3.4.1 Description	6
3.5 Module 3	6
3.5.1 Description	6
3.6 Use Case Diagram	7
3.7 Sequence Diagram	8
3.8 Packaging and Deployment Diagram	9
4. Proposed Methodology / Approach	10
4.1 Algorithm and Pseudocode	15
4.2 Implementation and Results	21
Appendix A: Definitions, Acronyms and Abbreviations	41
Appendix B: References	42

1. Introduction

1.1. Overview

India is the country where the agricultural sectors play a major role in the economy. Every year farmers face numerous problems due to the storage requirements, lack of proper monitoring of the food stored. Warehouse in the agriculture sector is considered as the more crucial sector generally for ensuring food security. In earlier days, there were outmoded methods for storing the foods and grains which required a lot of the manual approach occasionally which is time-consuming and inefficient. Food and grains start to spoil once they are harvested. And only a small part of the food and grains are stored in warehouses. A large part of the crops is left without proper storage facilities. due to the fluctuations in the market supply both from one season to next and from one year to next, the losses that the country faces every year due to improper storage is about Rs.50,000 cores in monetary terms.

A warehouse provides protection of foods from loss and damage due to excessive heat, moisture, dust, and wind. The main aim must be to maintain the crop in good condition for as long as possible. Storage of crops is one of the functions of warehousing where protection of crops and risk bearing is an essential factor. In addition, it prevents any mishaps like theft or loss. As observed by the study of the Food and Agriculture Organization that the higher the temperature, the lower should be the moisture of the grain in order to make sure good conservation of the crops. Due to high temperature, food loses its weight slowly and in the end is rotten.

The main objective of this work is to develop a monitoring and controlling of yields in warehouses using IOT which will give live data of temperature, moisture and other parameters and also to control these parameters remotely over the internet. Means if an unexpected situation is created then admin can control the situation via the internet using a web page. Another purpose is to automate this process where everything like storage temperature, humidity is automatically maintained without human interaction.

1.2. Purpose

Our project aims to provide warehouse automation and security using sensors. These activities might include controlling warehouse temperature, humidity, and other systems, in order to provide the convenience, comfort and efficient usage of warehouse.

1.3. Scope

Aim:

Our aim is to provide farmers crops with good storage rooms to reduce food misfortunes and increase food safety. Reducing food misfortune which will consequently build high amounts of food availability.

Benefits:

The vital advantage of this cycle is that it permits the stakeholders to comprehend the present status of the undertaking, the means taken, and increase in food availability.

Usage:

Producers, Dealers, Traders, Distributors for good storage facility.

2. Design Constraints, Assumptions

The constraints of the system are:

- Revenue and Affordability
- Consumption of Power-Needs 24/7 Power Supply
- Communication Range
- Data Availability and Quality
- Quality of Wireless link

The general assumptions of our product are:

- Regular supply of goods into the market place by being able to store goods when supply exceeds demand and then releasing them when demand exceeds.
- Maintaining consistent stock levels helps prices to stay stable, making it easier for businesses to forecast production, profit and loss.

3. Design Description

Physical Implementation: Execution of Sensors, Actuators and Micro Controllers
Implementation of Network Equipment

System Framework:

Subsystems

Sensing: Collection of Data (environment) through Sensors.

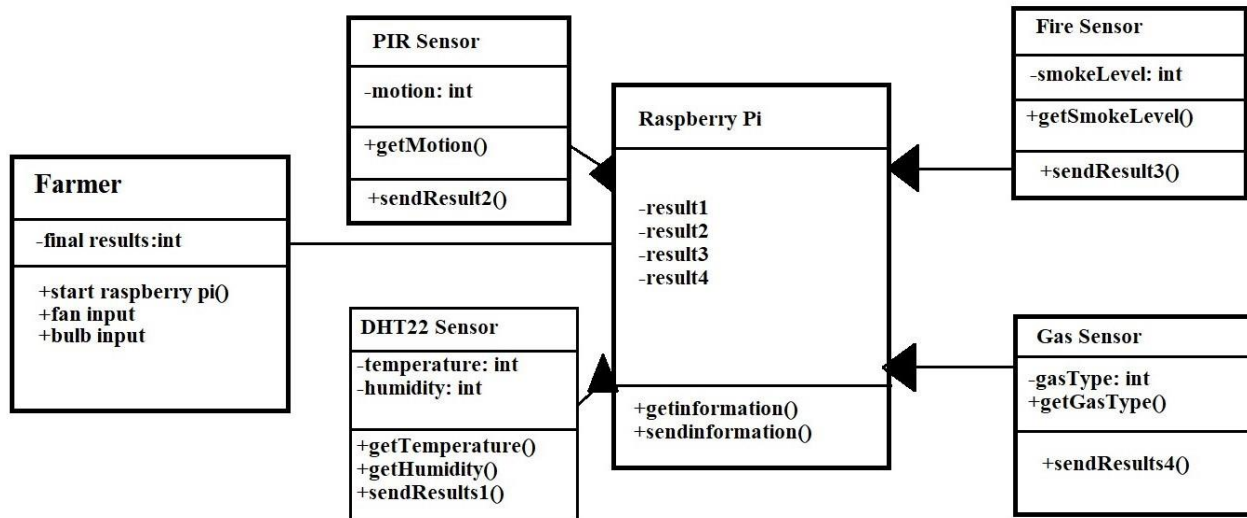
Data Communication: Communication between the focal and remote sensor hubs. Sensor Data is gathered into Cloud Storage.

LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT

Visualization: Visualization, processing and manipulation of data.

- **Data Analysis:** Data is analyzed using different algorithms. Through this the system is capable of decision making and execution based on manipulating sensor data.
- **Circuit Design:**
 - **Wireless Sensor Nodes:** Observing and Recording the states of being of the climate and putting together the gathered information at a central location.
 - **Central Node:** central and wireless nodes communication.
 - Sheets, Various Databases, Cloud Storage collect Data.
 - Decision making and execution is done by manipulated sensor data.
 - Information is gathered through different sensors and sent to servers by wireless sensor networks.
 - Data-information about ecological conditions so this is done to design the framework appropriately.
 - There are many factors other than just observing ecological conditions, yield quality, factors like monitoring temperature, humidity and movements like outsider invasion into the warehouse or motion of undesired objects are some which might also have an effect on food availability and quality and also food spoilage prediction.
 - Using IOT based Smart Warehouse techniques we can limit assets and ensure best utilization to improve availability of food.

3.1. Master Class Diagram



3.2. Module Name

- Device Module -
 - Raspberry Pi
 - DHT22
 - PIR
 - MQ135
 - Fire
- Cloud-IBM
- Web Application-Node Js

3.3. Module 1

Device Module

3.3.1. Description

All sensors are connected to Raspberry Pi which is the device module.

3.4. Module 2

IBM Cloud

3.4.1. Description

All the data which is collected by the sensors and sent to raspberry pi will be displayed and stored in IBM Cloud

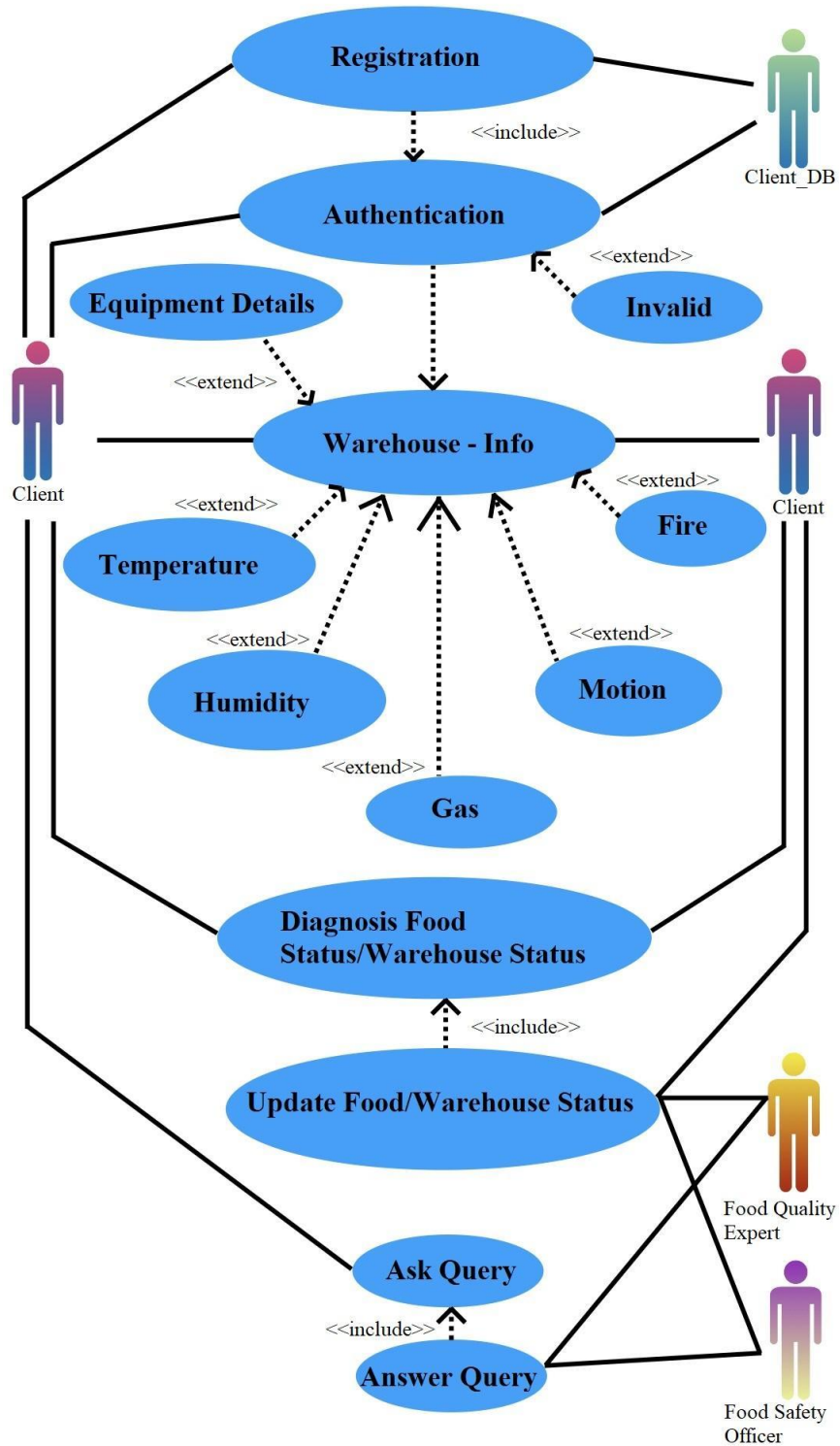
3.5. Module 3

Web Application

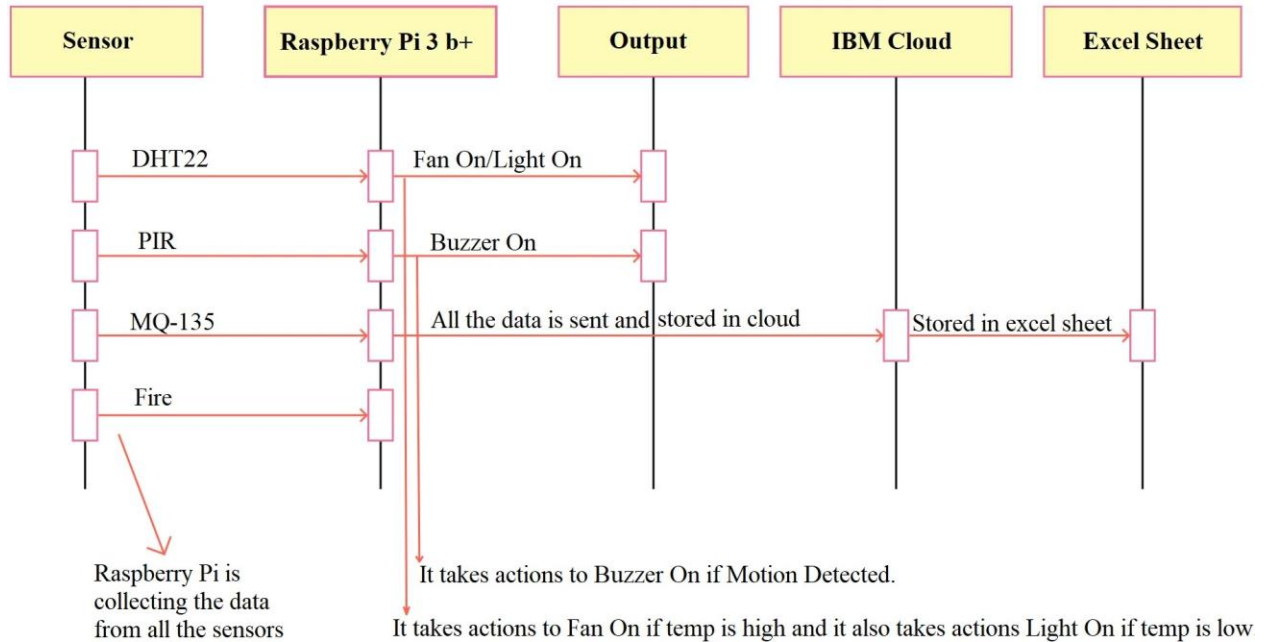
3.5.1. Description

All the data displayed in IBM cloud will also be displayed in the website in charts, gauge etc.

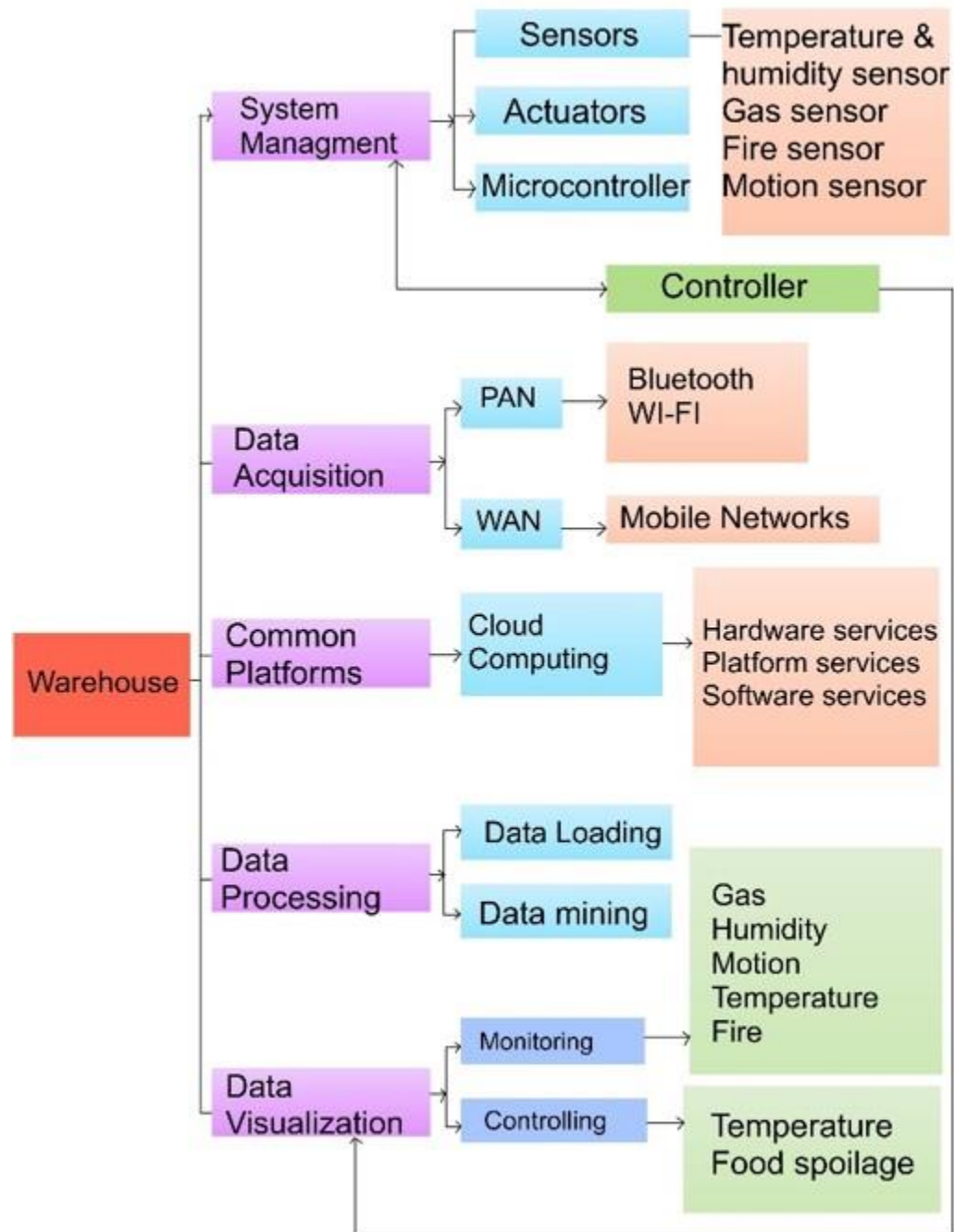
3.6. Use Case Diagram



3.7. Sequence Diagram



3.8. Packaging and Deployment Diagrams

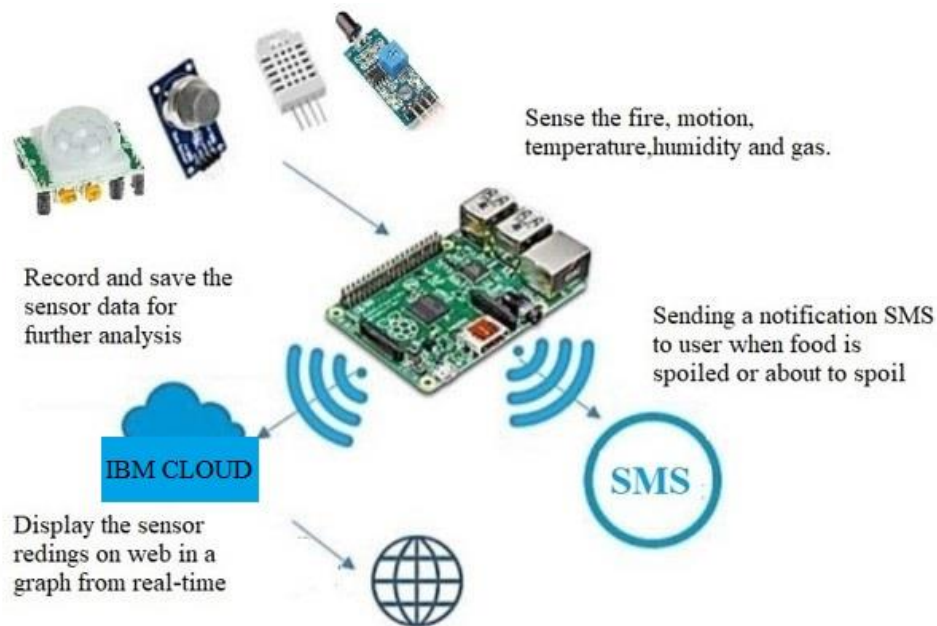


4. Proposed Methodology / Approach

Monitoring and Controlling crops in the warehouse have been proposed with six major steps involved. Implementation these steps in precise order will produce classification and/or recognition as a final outcome.

Steps in the proposed methodology:

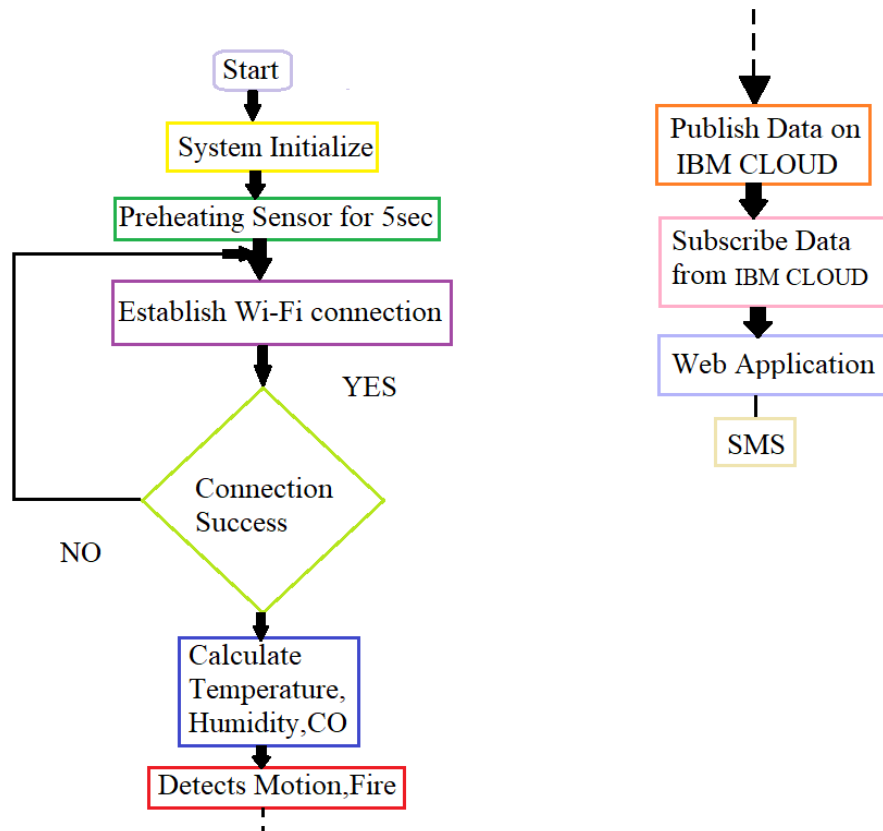
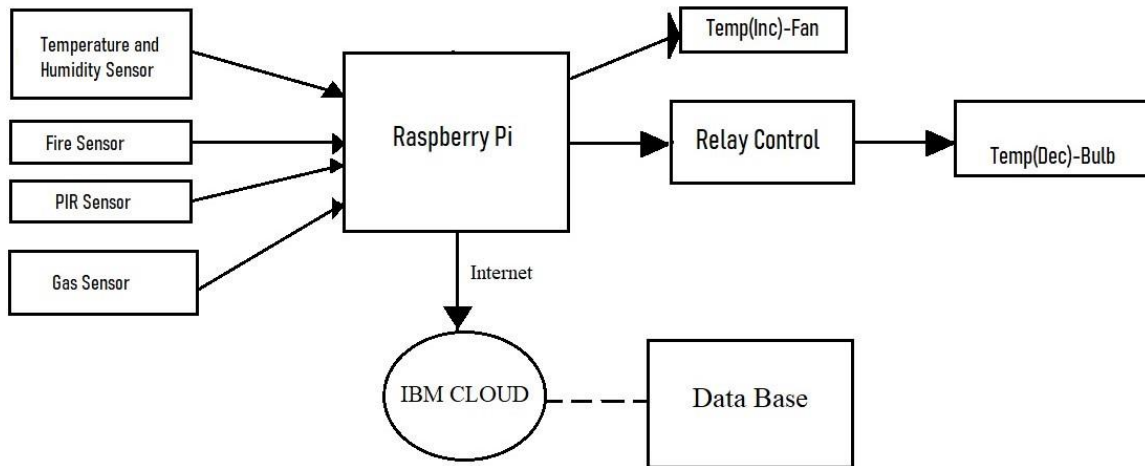
- Check temperature, humidity, CO₂ values in the warehouse and record it.
- Input Temperature and humidity range for specific crops stored in the warehouse.
- If fire is detected then SMS will be sent to the farmer or the manager.
- If motion is detected then the buzzer will switch on.
- If the temperature decreases than the required amount then the light will switch on.
- If the temperature increases than the required amount then the fan will switch on.



In the above proposed model, raspberry pi 3b+ model controller adopts IoT to convey the messages. The effective open source language python is used for programming. This is a very lucrative product for future use. The daring thing is to capture the ambient changes inside the warehouse. The DHT22 checks for the moisture and temperature contents. The PIR sensor checks

for the motion in the warehouse. The MQ-135 checks for carbon dioxide contents. The Fire sensor checks for the fire in the warehouse. The Raspberry pi has an internal Wi-Fi module through which the IoT is connected. It also has a SD slot which stores a limited range of each sensor. The controller checks at regular intervals when the range exceeds it gives an alert. We monitor this by connecting with HDMI. Also, we use the HDMI port for scrutinizing the whole process. The Raspberry pi has an internal Wi-Fi module through which the IoT is connected. It also has a SD slot which stores a limited range of each sensor. Python language is a high level and highly interpreted programming language. It was founded and created by Guido van Rossum and was first released in 1991, It has a philosophy of design that has readability of the code, with significant whitespace. It enables and provides clear coding knowledge in both large and small scales. The main features of Python are automatic memory management and dynamic type system. It supports object-oriented, including imperative, multiple programming paradigms, procedural, and functional and also has a very large library with comprehensive functions. Many operating systems interpret the python programming language. It is an open-source Code Python and has a model with community-based development. as do nearly variant executions. It is a nonprofit foundation of the software industry. In this system basically we are checking how many days can a crop sustain in specific conditions which is compared with the crops left outside the box. With the help of python, we are executing and running our system. When running the system first we will login to IBM cloud and the run the code. After running the code all values collected by the sensors are sent to raspberry pi and then raspberry pi will send this data to IBM cloud to display. Again, in IBM cloud simultaneously this data will be stored in cloudant which a cloud database. And we are also storing the data in sheets.

LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT

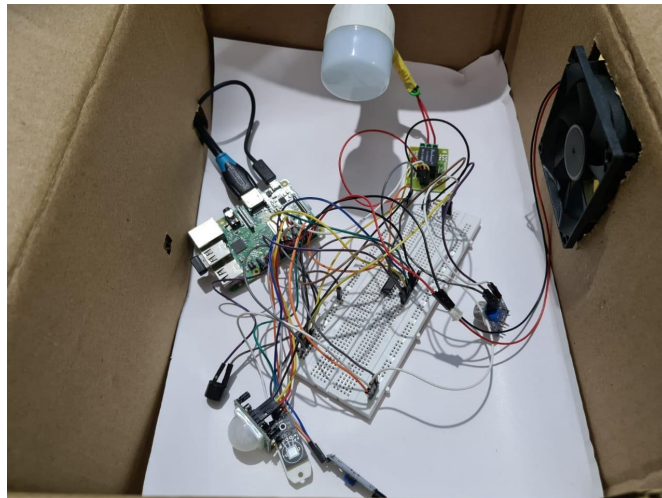


Advantages

- Reduced investment risk.
- Flexibility to design to specifications.
- Greater direct control on warehousing activities.
- If the volume is sufficient, this may workout cheaper.

Disadvantages

- Cost of manpower.
- Warehouses cause high prices of goods due to rent charges by owners of the warehouse.



Risks

1. THE SECURITY FACTOR
2. TECHNICAL FAILURE AND FALSE ALARMS
3. CHANCES OF RASPBERRY PI GETTING DAMAGED
4. DATA SECURITY AND PRIVACY
5. NOT IN TIME ALERTS
6. CONTEXT-AWARENESS (META DATA)

Functional Requirements

The progressive necessities indicate the capacities and units of the proposed framework. They characterize the conduct of the framework identifying with need:

1. Measure Temperature
2. Gauge Humidity
3. Sense Carbon Dioxide
4. Sense Fire
5. Sense Motion
6. Estimate the Light intensity
7. Connecting Devices
8. Permit clients to alter the ideal qualities for the sensor
9. React to sensor readings and send alarms to the client.

4.1 Algorithm and Pseudocode

PSEUDO CODE :

- **PIR Motion Detection Sensor**

```
GPIO.setup(23, GPIO.IN)
```

```
sleep(0.5) #Wait 5 seconds and read again
```

```
if GPIO.input(23):#pir
```

```
    M='detected'
```

```
    GPIO.output(24, True)#buzzer
```

```
    print("Motion Detected...")
```

```
    time.sleep(1)
```

```
else:
```

```
    M='Not Detected'
```

```
    GPIO.output(24, False)
```

```
q=requests.get('https://www.fast2sms.com/dev/bulkV2?authorization=CZajTYBFxqW
```

```
9thEluMd3QKlceHgikzSyRo0JL7b6UPVn4fpvsDJd6pvkVMt71eGDcnKIRBqQHgzN
```

```
o5L9&message=Food Spoil
```

```
Alert&language=english&route=q&numbers=7993397777')
```

```
print(q)
```

- **DHT22 Sensor**

```
import Adafruit_DHT
DHT_SENSOR = Adafruit_DHT.DHT22
DHT_PIN = 4
GPIO.setup(16,GPIO.OUT)#fan
GPIO.setup(27,GPIO.OUT)#Light
T=0
H=0
while True:
    H, T = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
    sensorValue = _range(channel_0.value, 0, 60000, 0, 1023)#gas sensor
    if H is not None and T is not None:
        print("Temp={0:0.1f}*C Humidity={1:0.1f}%".format(T, H))
    else:
        print("Failed to retrieve data from humidity sensor")
    if T>30:
        GPIO.output(16,True)
        GPIO.output(27,False)#light
        print("Fan ON")
    else:
        GPIO.output(27,True)
        GPIO.output(16,False)#fan
        print('Light ON')
```


- **Fire Sensor**

```
channel=25
```

```
GPIO.setup(channel,GPIO.IN)#flame
```

```
flame=0
```

```
def callback(channel):
```

```
    print("channel",channel)
```

```
    global flame
```

```
    flame=1
```

```
    print("flame detected")
```

```
    r=
```

```
requests.get('https://www.fast2sms.com/dev/bulkV2?authorization=CZajTYBFxqW9thEI  
uMd3QKlceHgikzSyRo0JL7b6UPVn4fpvsDJd6pvkVMt71eGDcnKIRBqQHgzNo5L9&  
message=Alert!!Fire Detected in
```

```
Warehouse&language=english&route=q&numbers=7993397777')
```

```
    print(r)
```

```
GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300) # let us know  
when the pin goes HIGH or LOW
```

```
GPIO.add_event_callback(channel, callback) # assign function to GPIO PIN,  
Run function on change
```

- **MQ-135**

```
import adafruit_mcp3xxx.mcp3008 as MCP

from adafruit_mcp3xxx.analog_in import AnalogIn

# Create the SPI bus

spi = busio.SPI(clock=board.SCK, MISO=board.MISO, MOSI=board.MOSI)

# Create the cs (chip select)

cs = digitalio.DigitalInOut(board.D5)

# Create the mcp object

mcp = MCP.MCP3008(spi, cs)

while True:

    H, T = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)

    sensorValue = _range(channel_0.value, 0, 60000, 0, 1023)#gas sensor

    if (T>40 and sensorValue>150):

        print("Spoil Alert!!!")

q=requests.get('https://www.fast2sms.com/dev/bulkV2?authorization=CZajTYBFxqW9t
hEIuMd3QKlceHgikzSyRo0JL7b6UPVn4fpvsDJd6pvkVMt71eGDcnKIRBqQHzgNo5L
9&message=Food Spoil Alert&language=english&route=q&numbers=7993397777')

print(q)
```

- **MCP-3008**

```
from spidev import SpiDev

class MCP3008:

    def __init__(self, bus = 0, device = 0):

        self.bus, self.device = bus, device

        self.spi = SpiDev()

        self.open()

        self.spi.max_speed_hz = 1000000 # 1MHz

    def open(self):

        self.spi.open(self.bus, self.device)

        self.spi.max_speed_hz = 1000000 # 1MHz

    def read(self, channel = 0):

        cmd1 = 4 | 2 | (( channel & 4) >> 2)

        cmd2 = (channel & 3) << 6

        adc = self.spi.xfer2([cmd1, cmd2, 0])

        data = ((adc[1] & 15) << 8) + adc[2]

        return data

    def close(self):

        self.spi.close()
```

- **IBM CLOUD**

```
deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-  
method": authMethod, "auth-token": authToken}
```

```
deviceCli = ibmiotf.device.Client(deviceOptions)
```

- **Call Back**

```
def myOnPublishCallback():
```

```
print ("Published Temperature = %s C" % T, "Humidity = %s %% " % H, "to IBM Watson")
```

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

```
if not success:
```

```
    print("Not connected to IoTF")
```

```
time.sleep(1)
```

```
flame=0
```

```
deviceCli.commandCallback = myCommandCallback
```

```
data = { 'Temperature' : T, 'Humidity': H , 'Motion':M, 'Flame':flame , 'Gas_Sensor':sensorValue}
```

```
print (data)
```

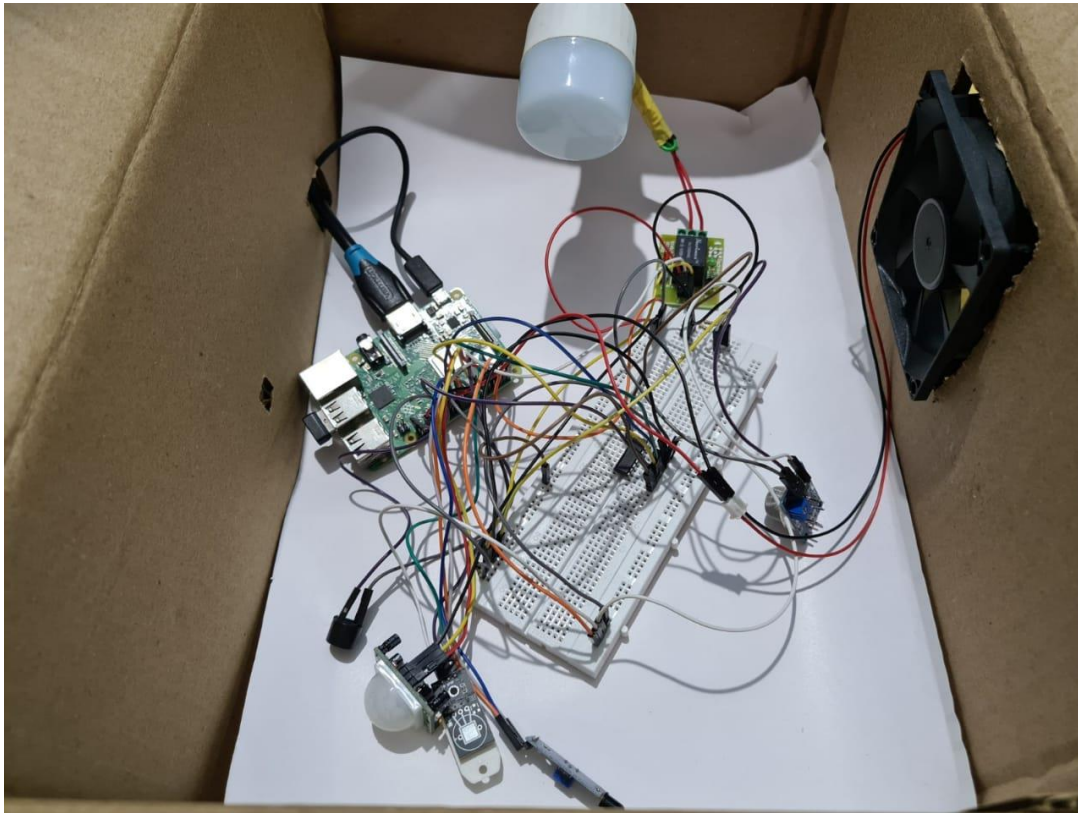
4.1 Implementation and Results

In this project the analysis of the results are Crop monitoring and Visualizing the data in Cloudant and storing the data in the sheets and alerts regarding the warehouse is sent to the user.

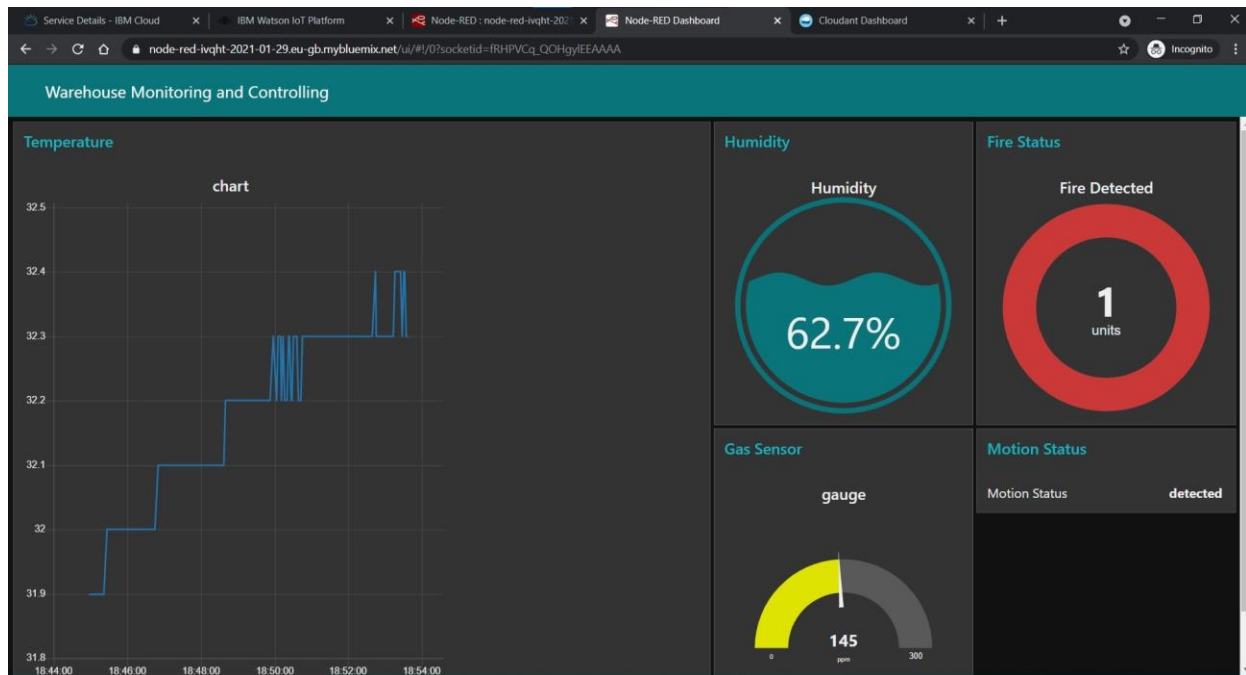
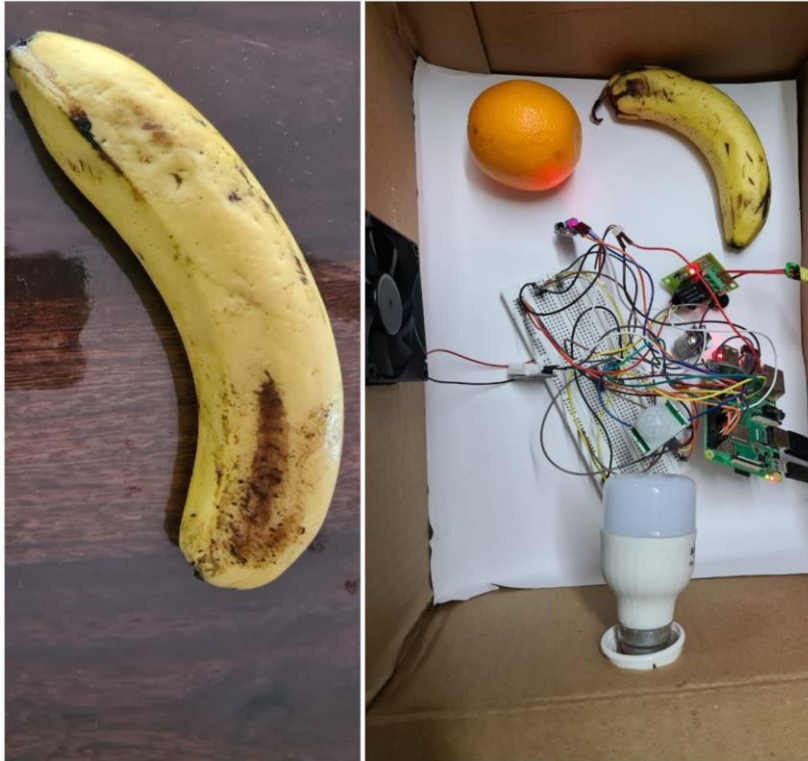
Crop Monitoring:

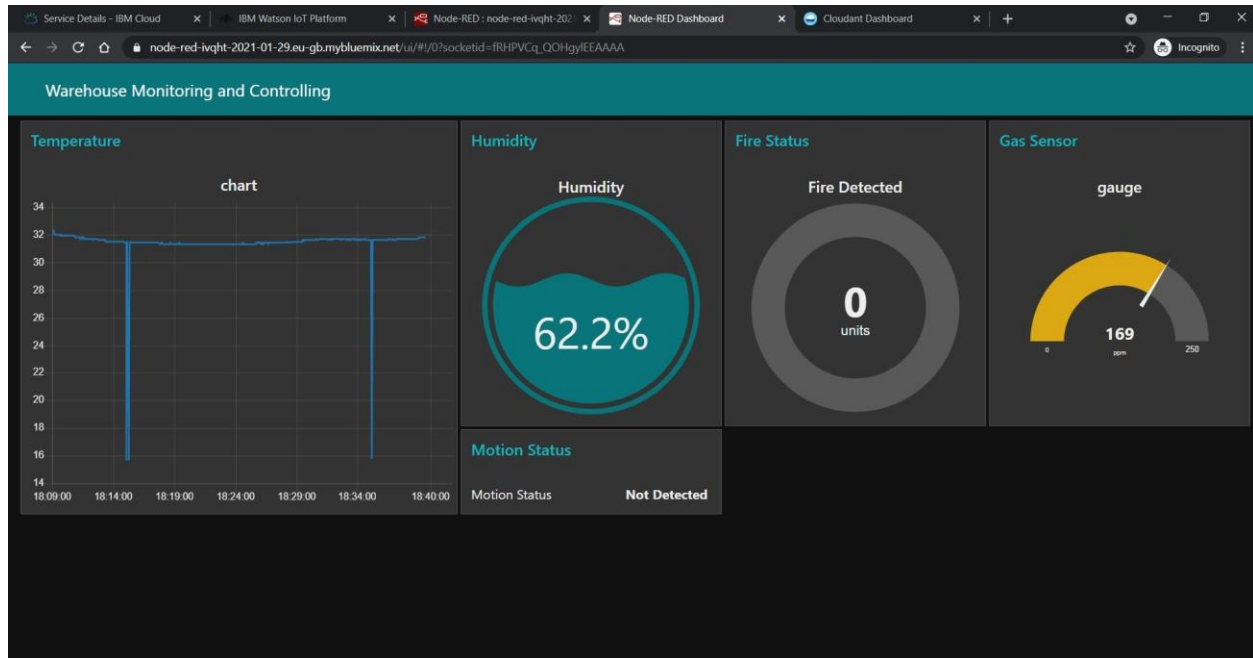
In this module it monitors the complete crop with the help of sensors and the data is visualized in cloudant and stored in sheets.

Monitoring and Controlling System:



Day 1:

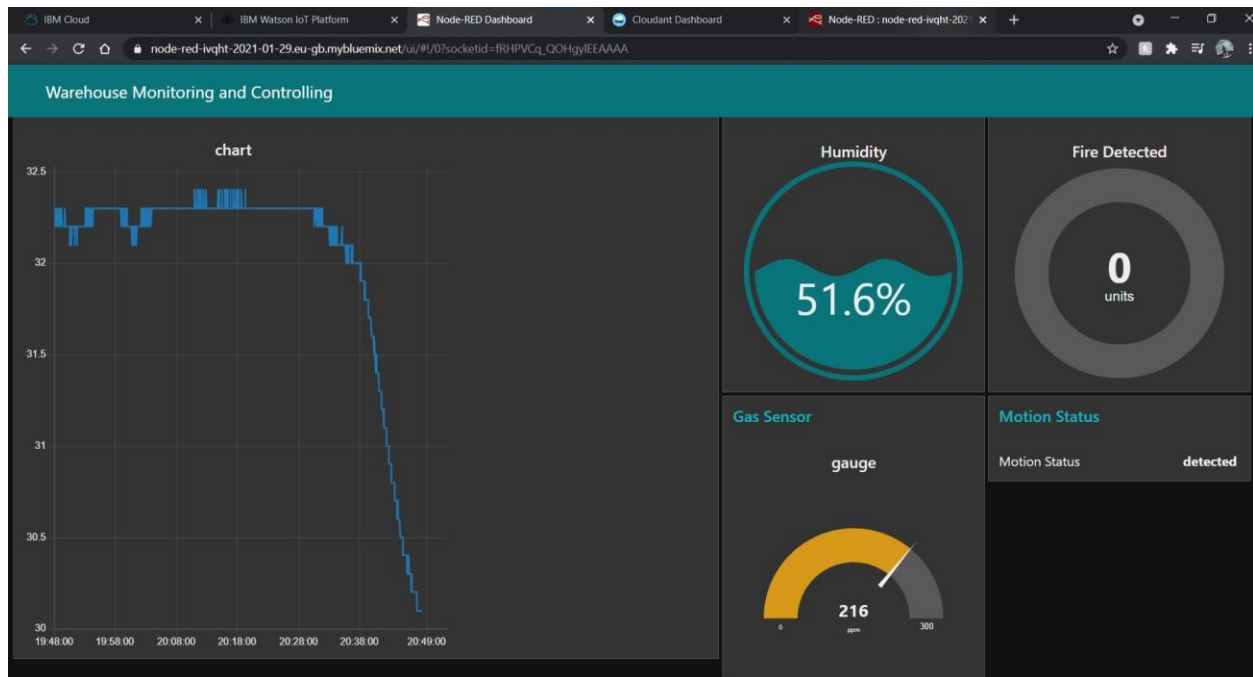
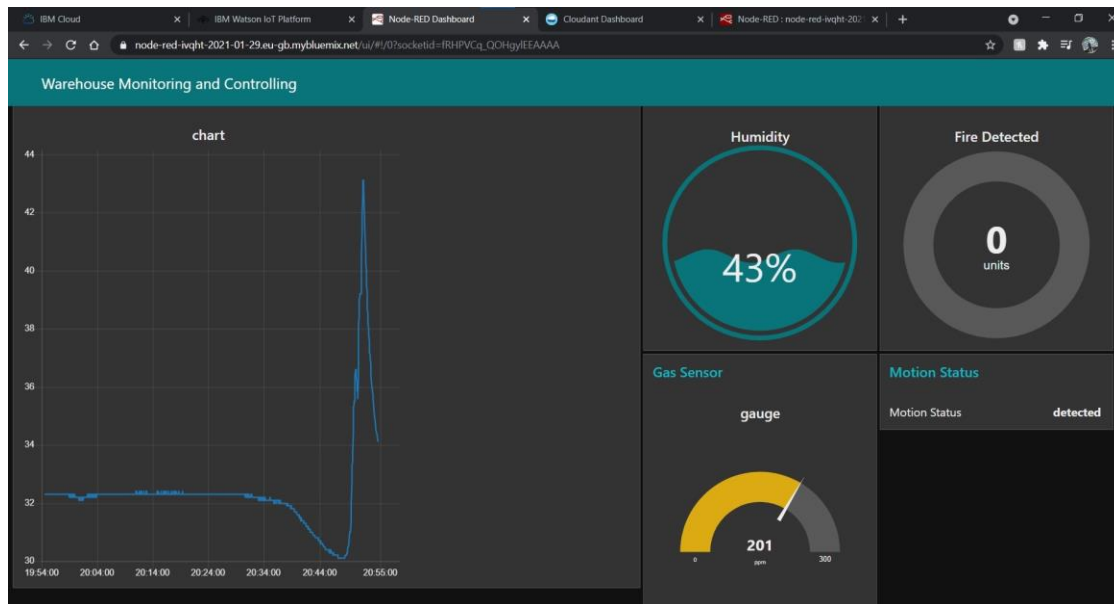




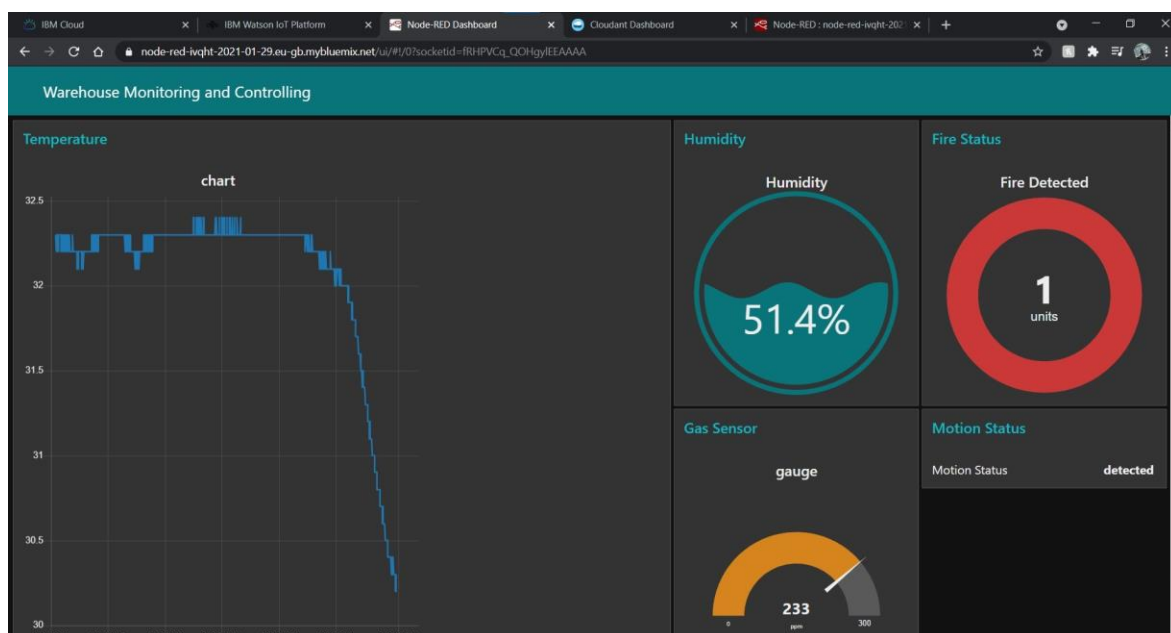
Day 2:

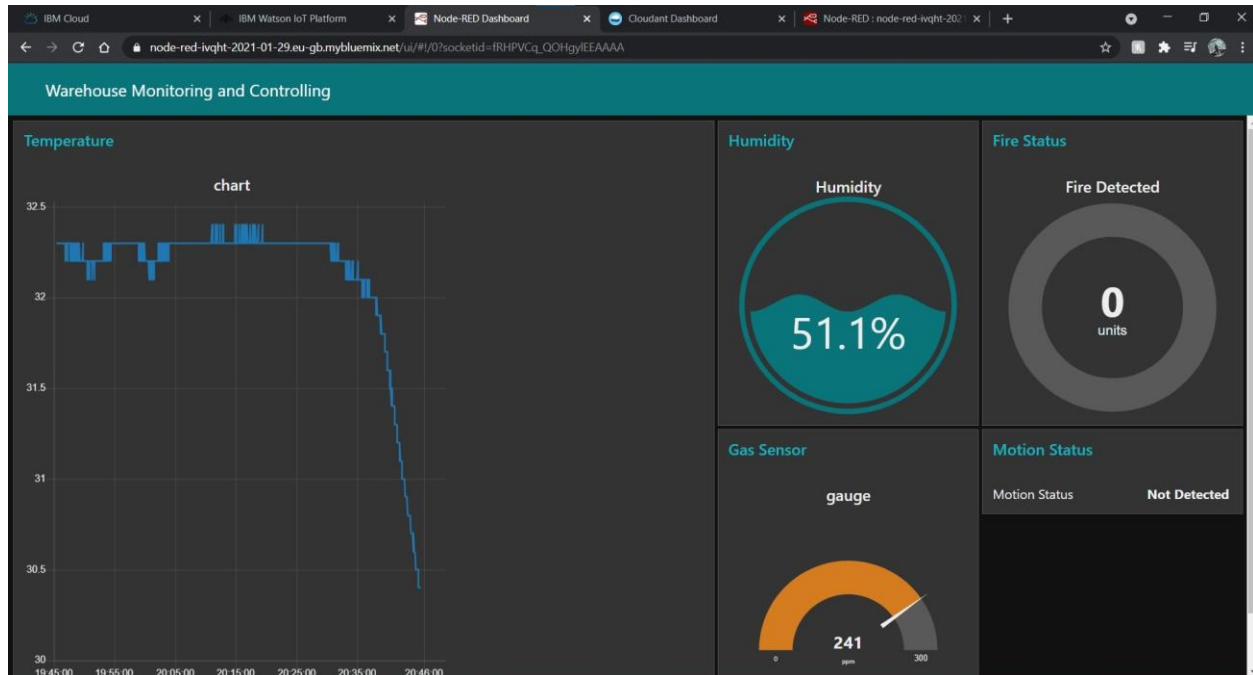


LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT



Day 3:

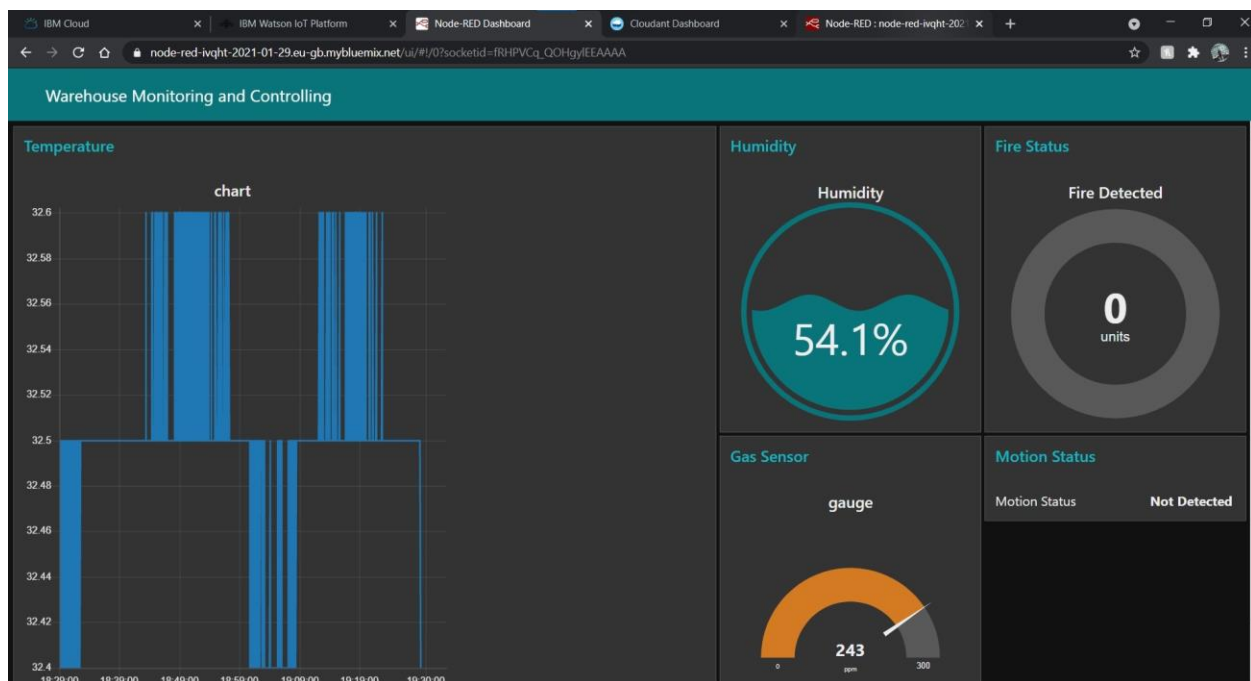
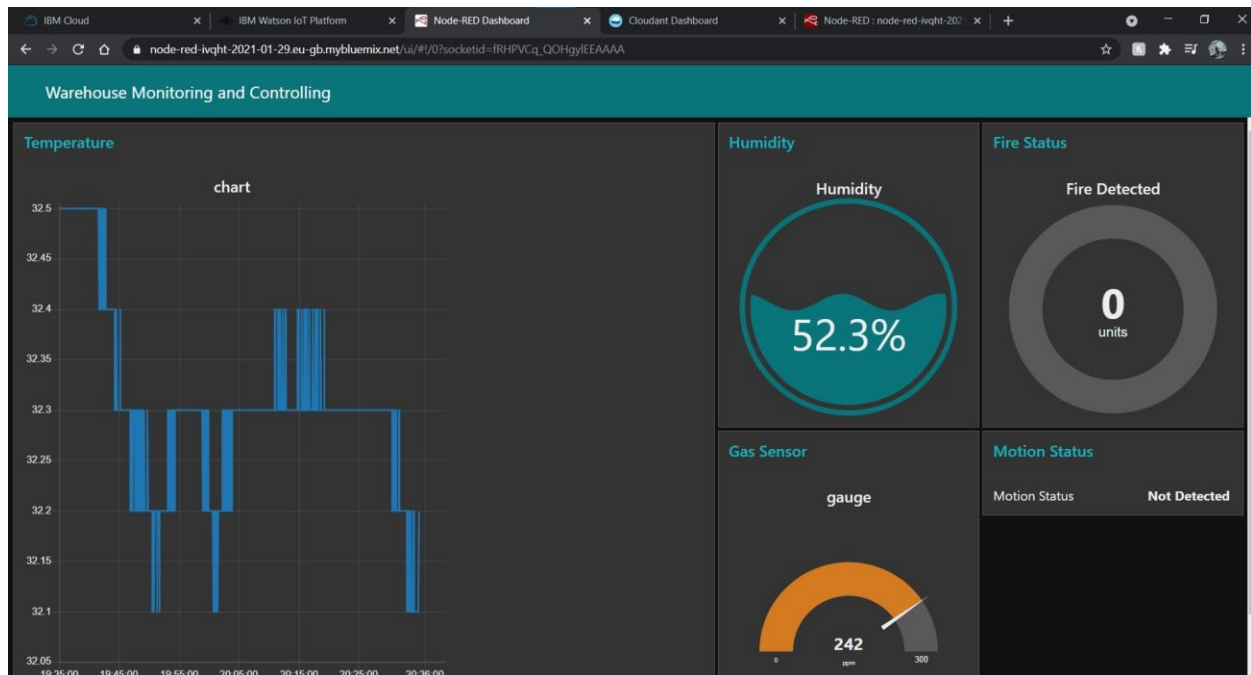




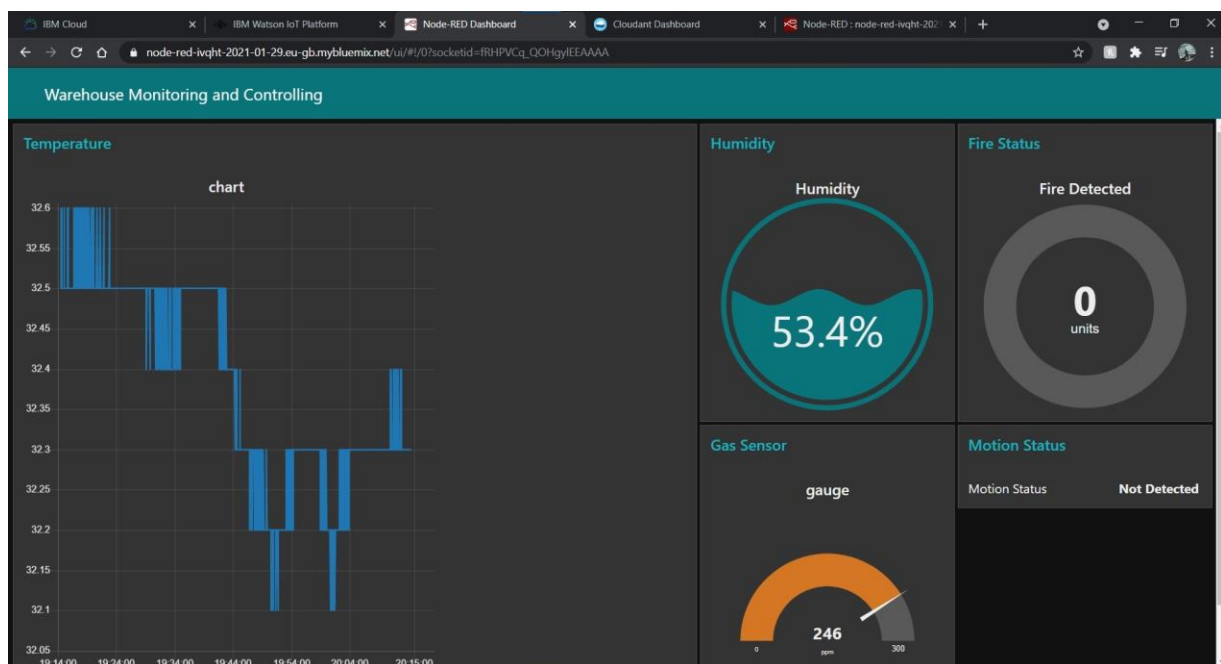
Day 4:

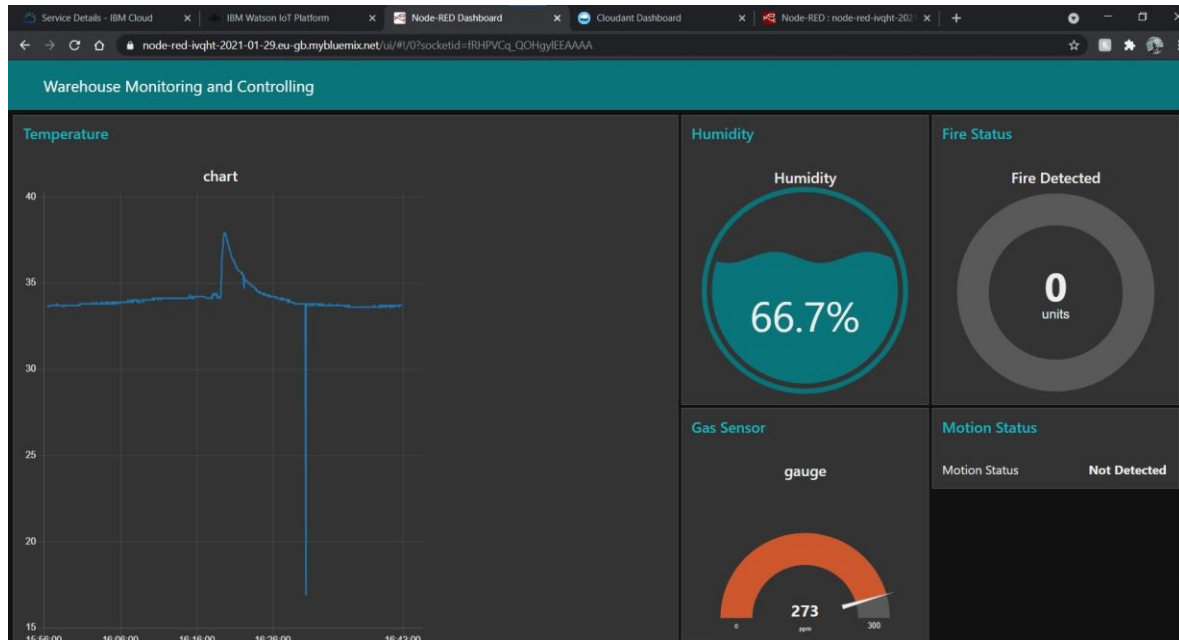


LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT

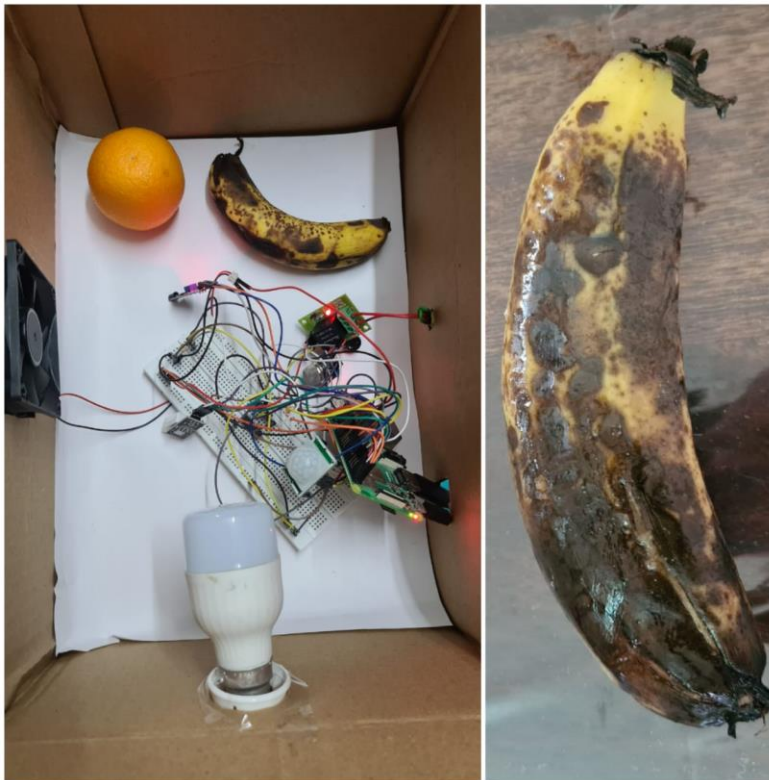


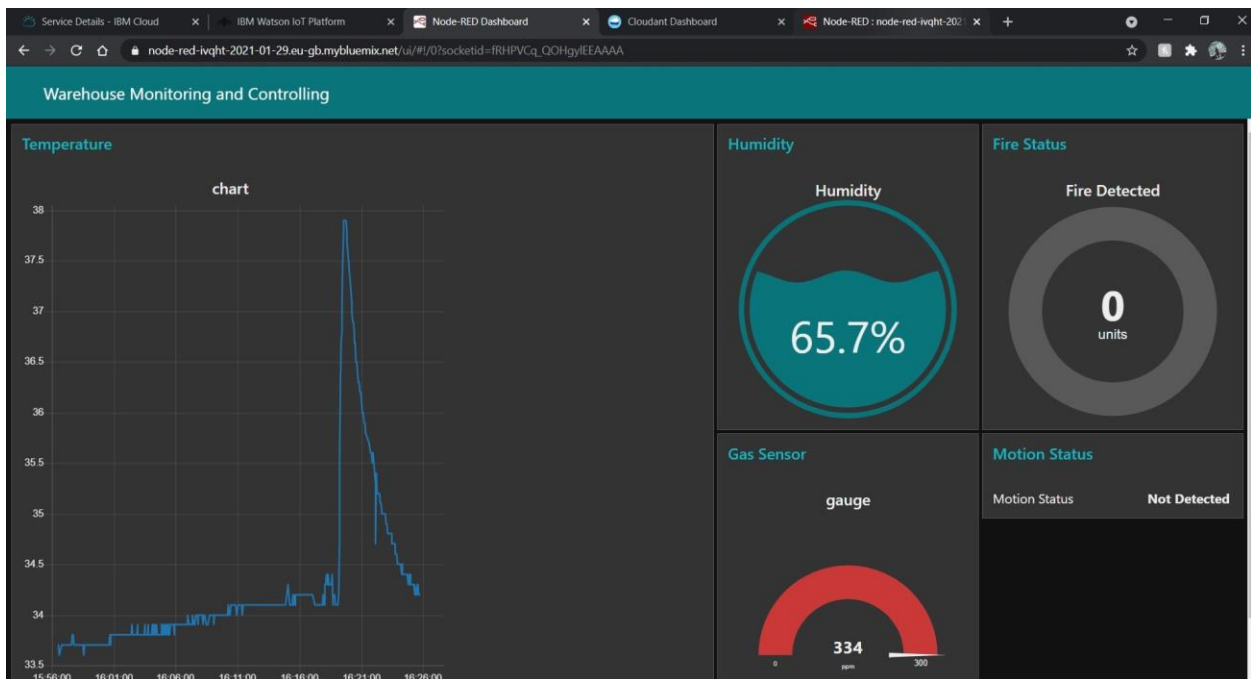
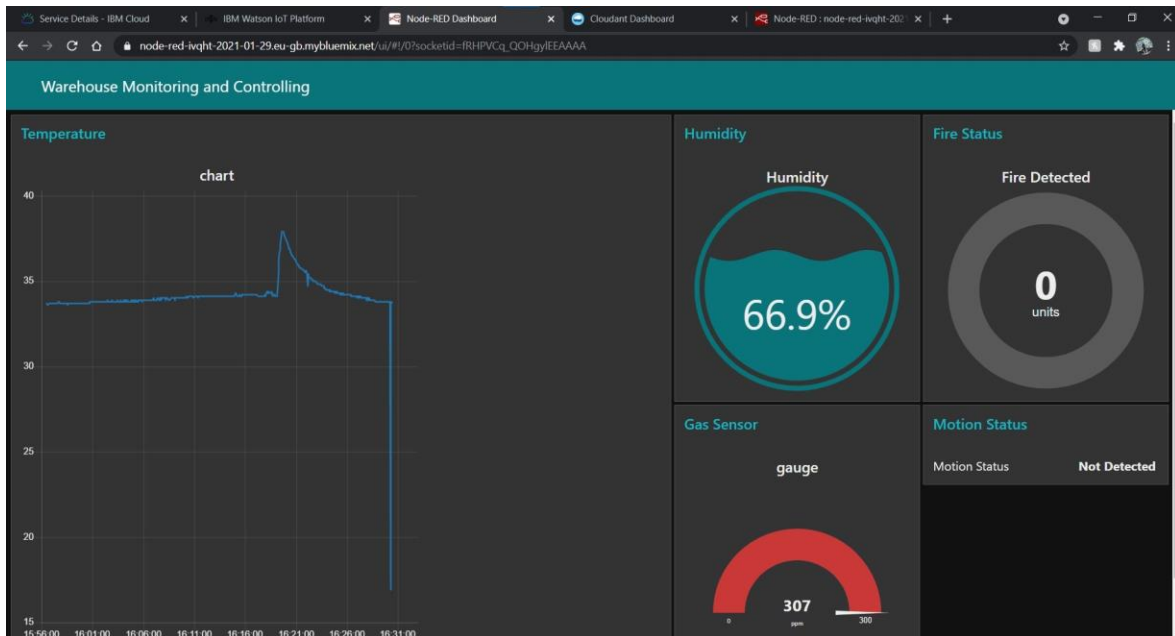
Day 5:





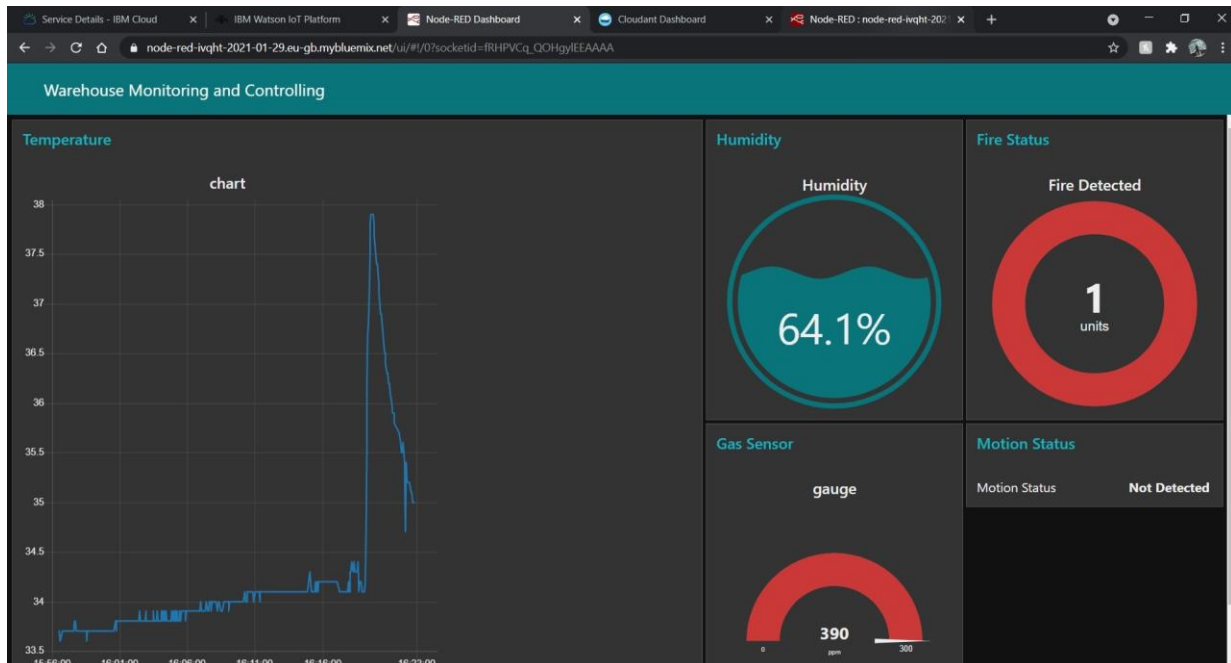
Day 6:

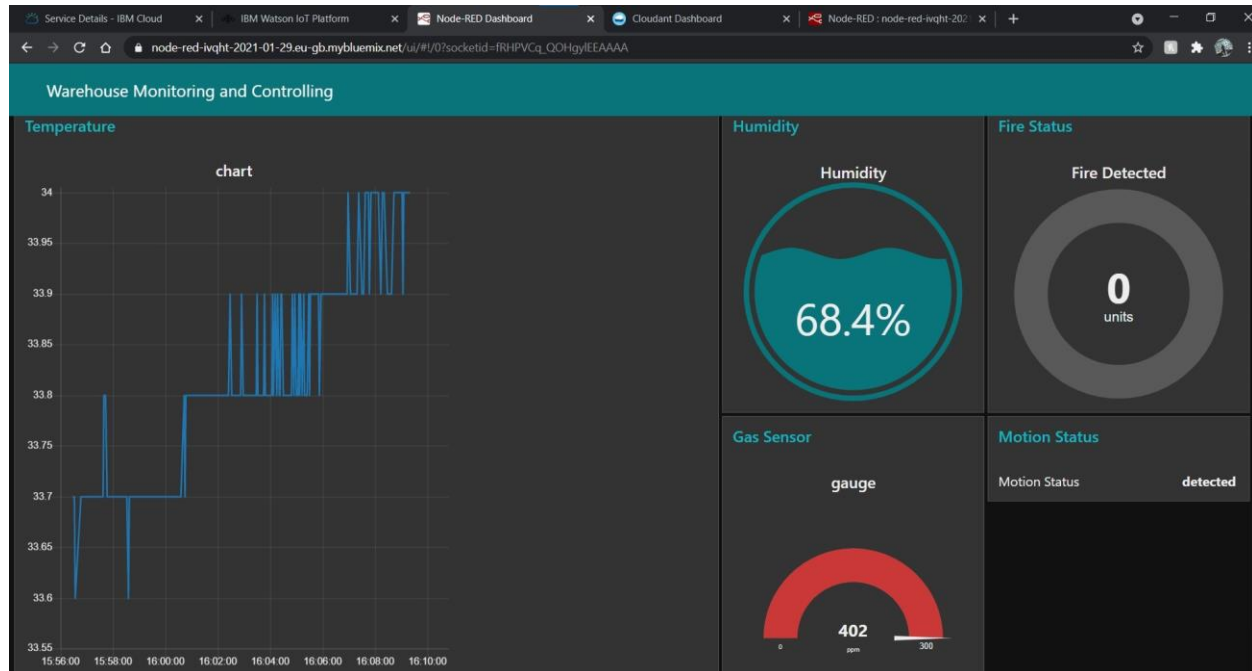




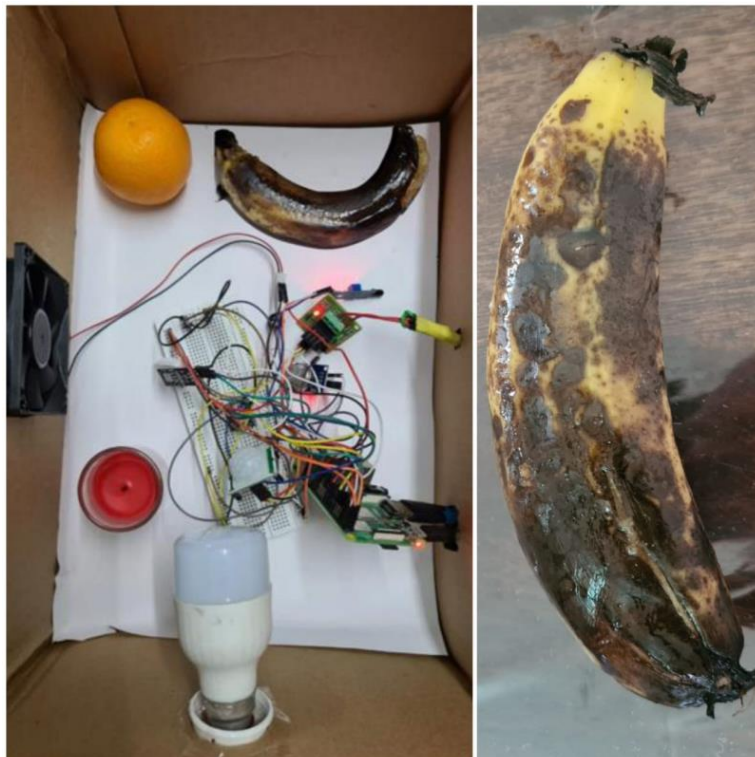
Day 7:

LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT

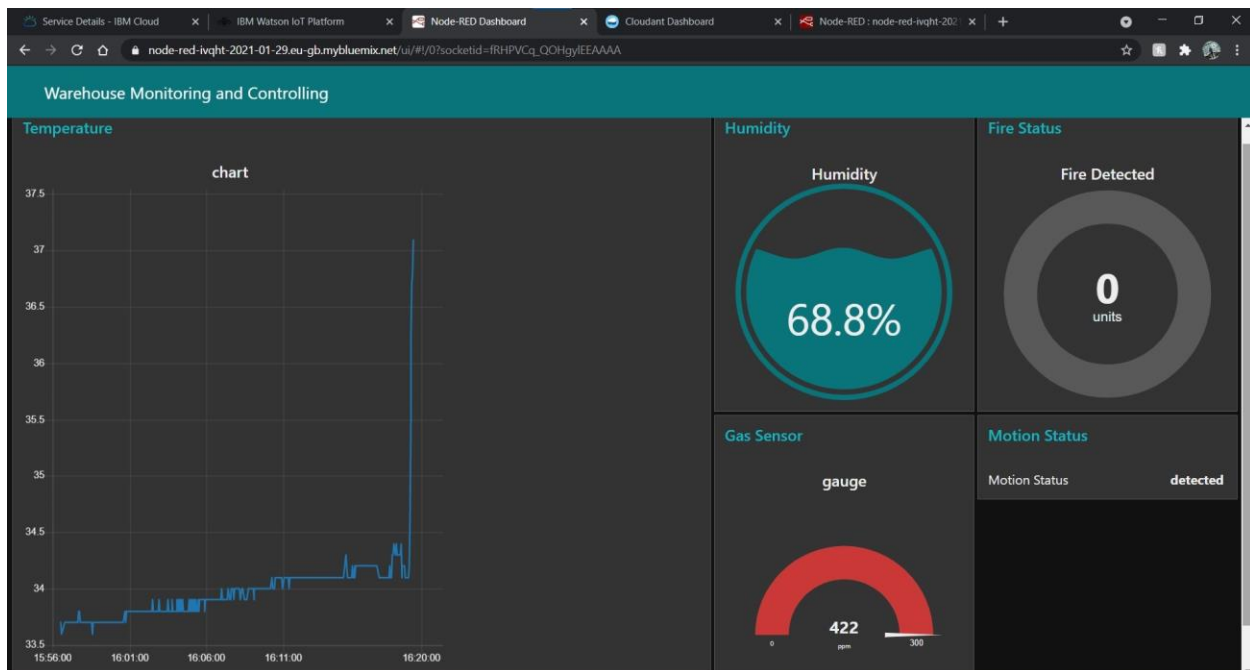
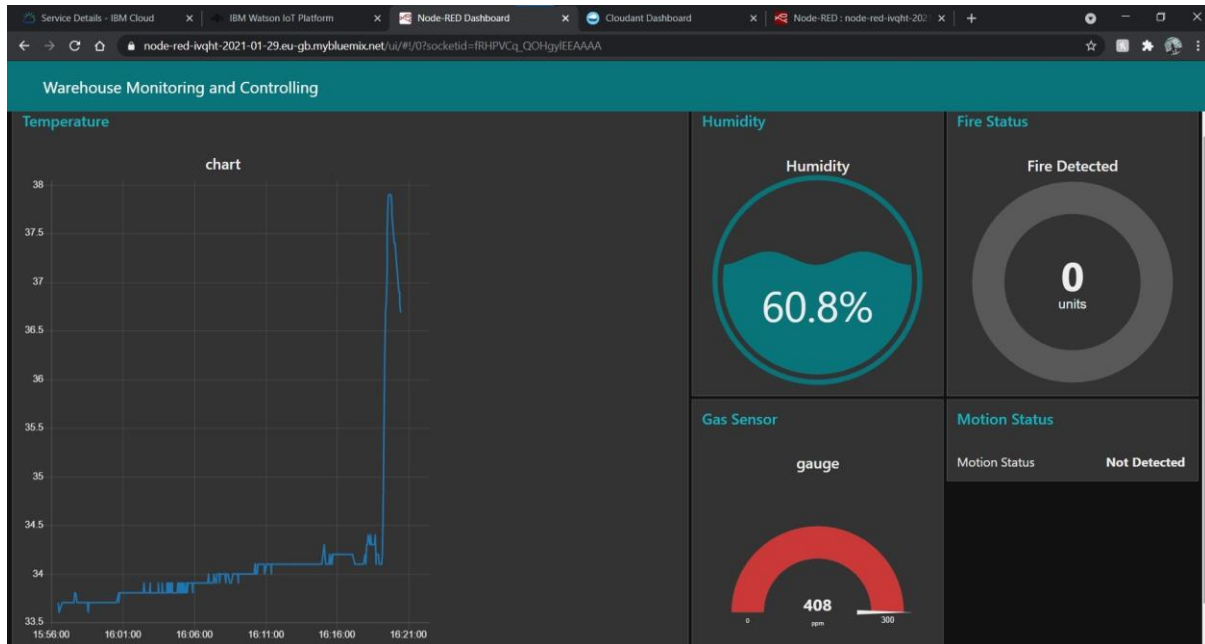




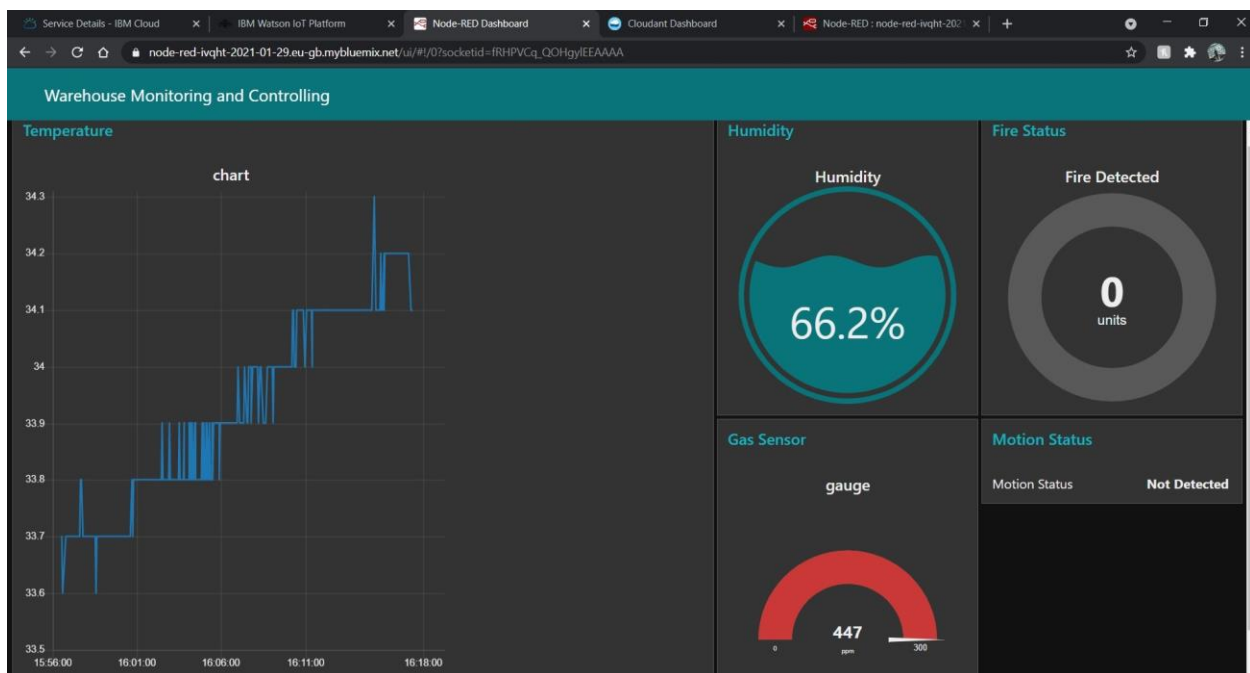
Day 8:



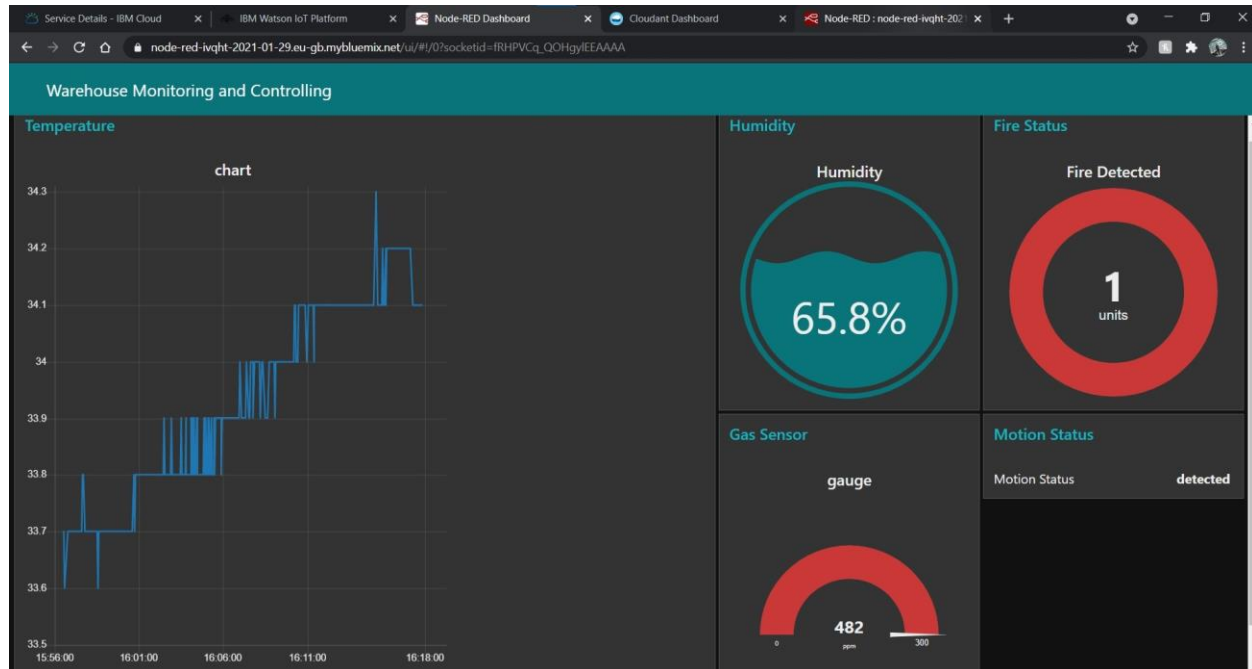
LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT



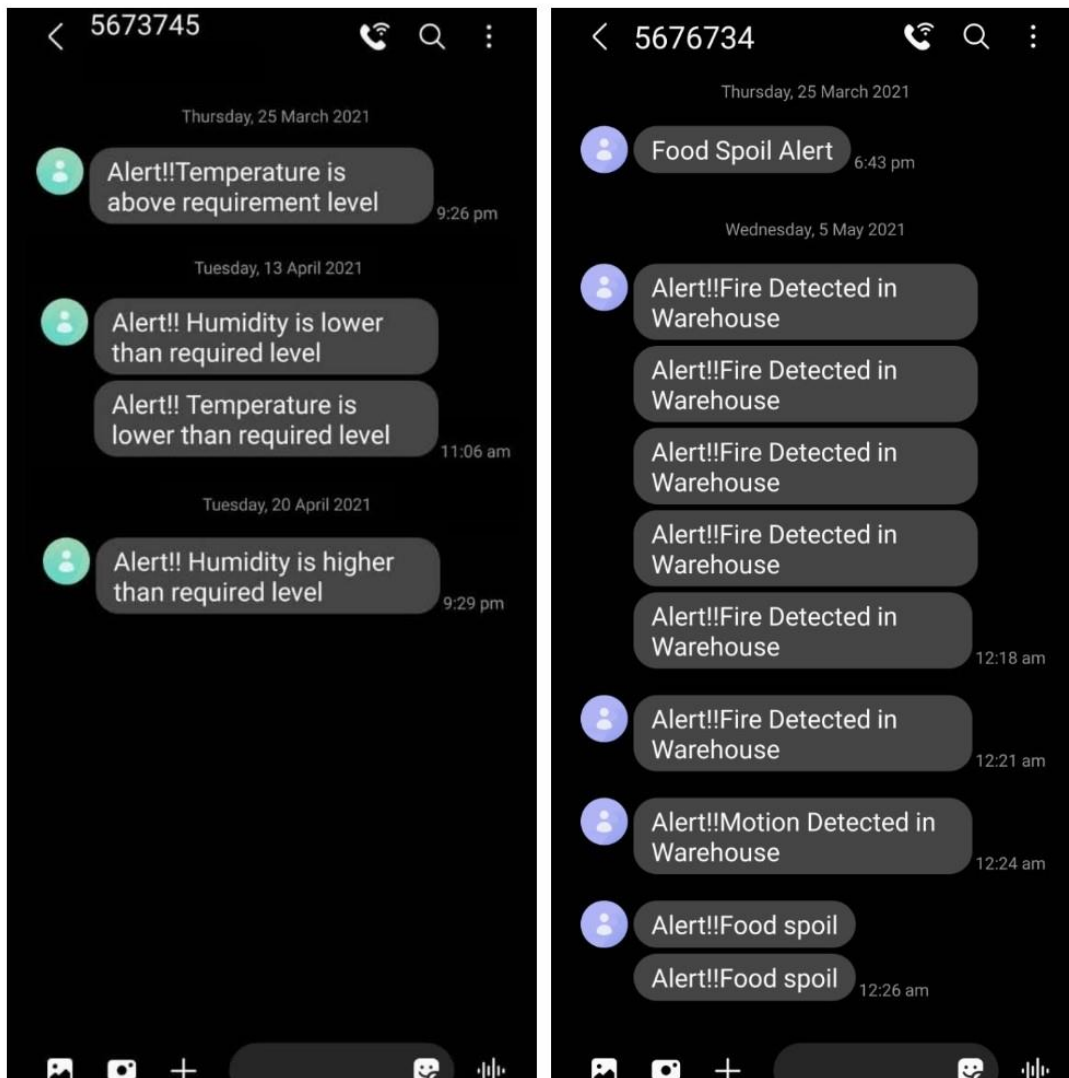
Day 9:

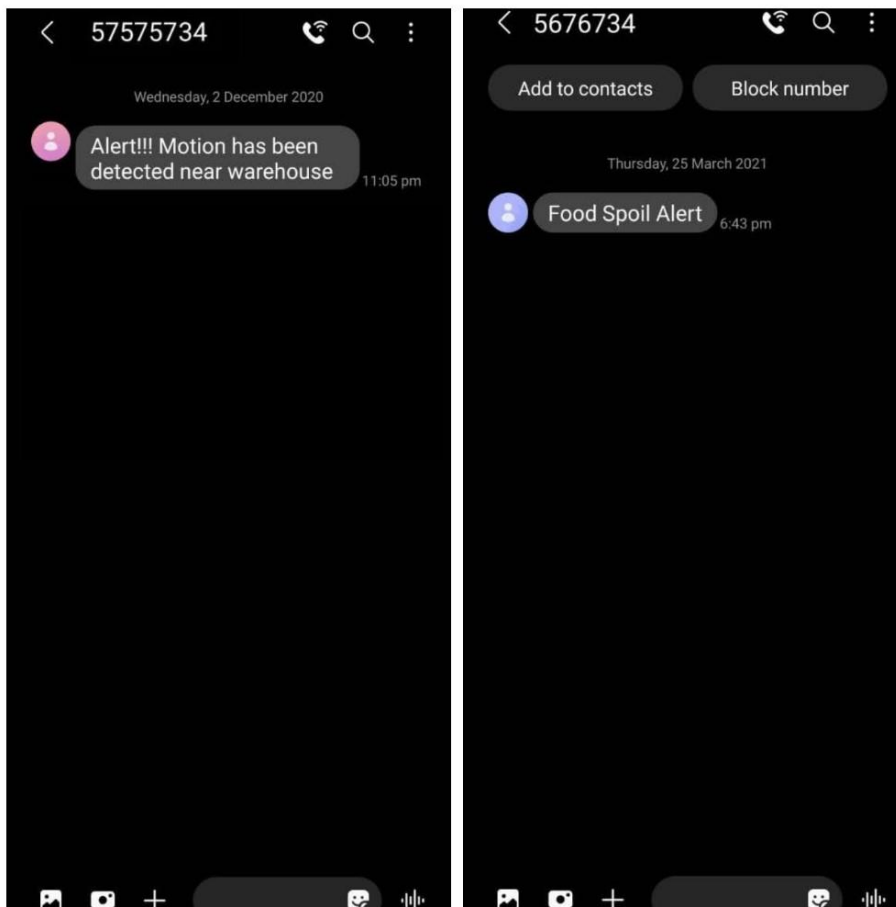


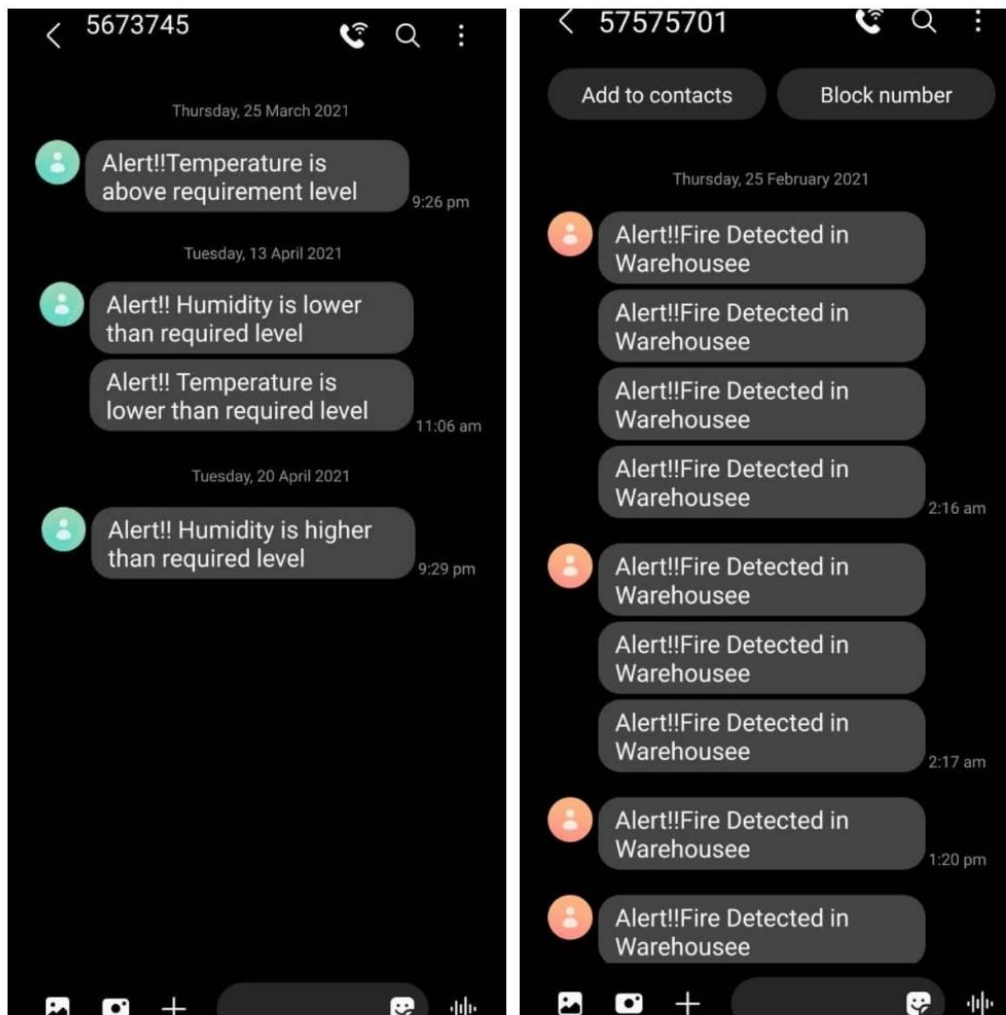
LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT



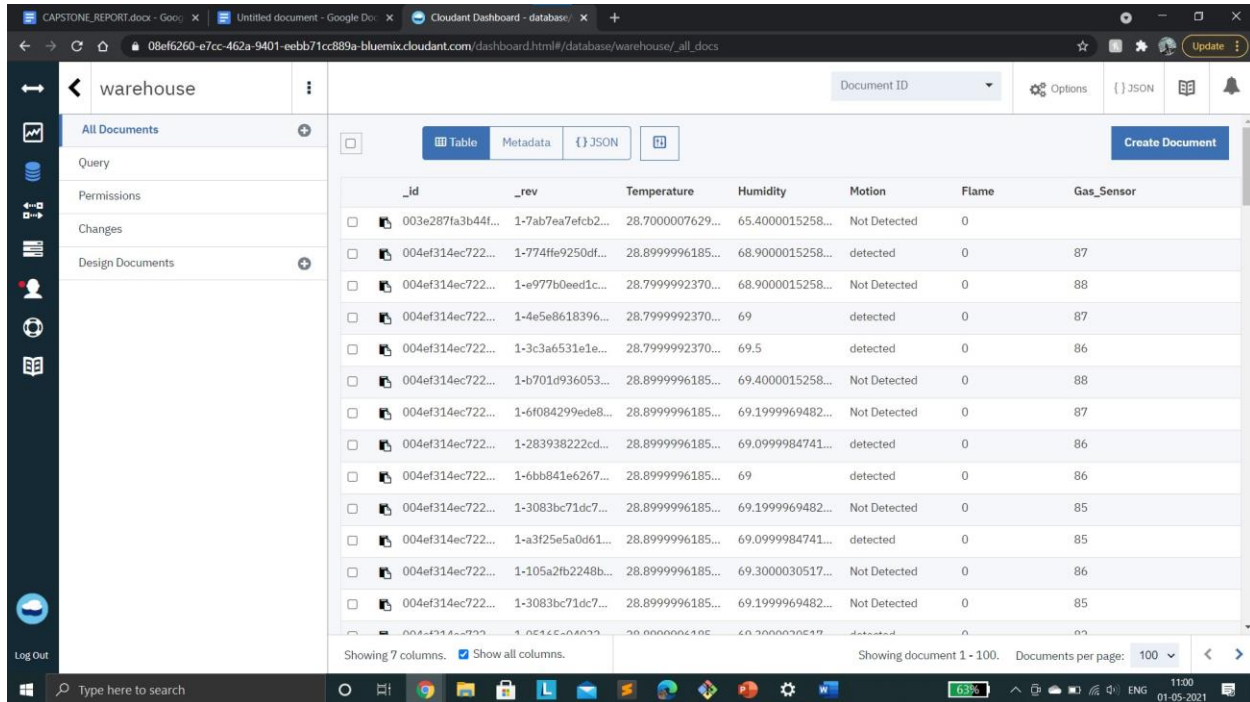
Messages:





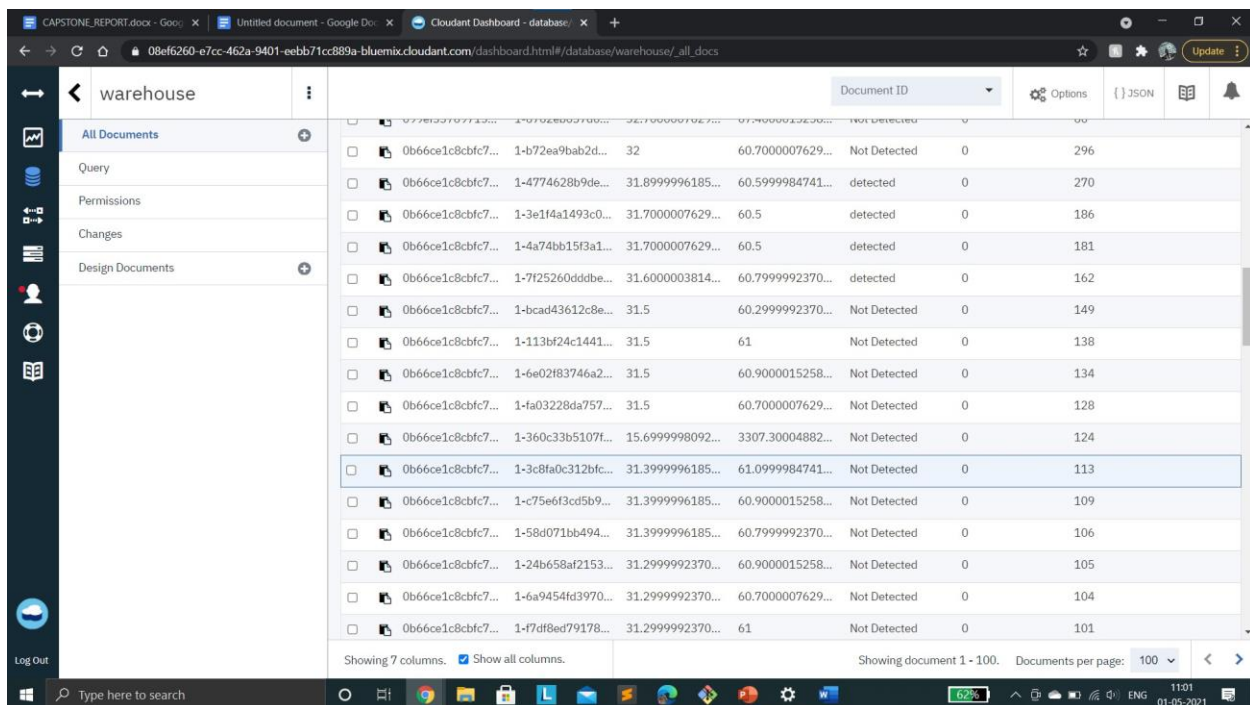


Database:



The screenshot shows the Cloudant Dashboard interface for a database named 'warehouse'. The table displays sensor data with the following columns: _id, _rev, Temperature, Humidity, Motion, Flame, and Gas_Sensor. The data is paginated, showing documents 1 through 100.

_id	_rev	Temperature	Humidity	Motion	Flame	Gas_Sensor
003e2871a3b44f...	1-7ab7ea7efcb2...	28.7000007629...	65.4000015258...	Not Detected	0	
004ef314ec722...	1-7747fe9250df...	28.8999996185...	68.9000015258...	detected	0	87
004ef314ec722...	1-e977b0eed1c...	28.7999992370...	68.9000015258...	Not Detected	0	88
004ef314ec722...	1-4e5e8618396...	28.7999992370...	69	detected	0	87
004ef314ec722...	1-3c3a6531e1e...	28.7999992370...	69.5	detected	0	86
004ef314ec722...	1-b701d936053...	28.8999996185...	69.4000015258...	Not Detected	0	88
004ef314ec722...	1-6f084299ede8...	28.8999996185...	69.1999969482...	Not Detected	0	87
004ef314ec722...	1-283938222cd...	28.8999996185...	69.0999984741...	detected	0	86
004ef314ec722...	1-6bb841e6267...	28.8999996185...	69	detected	0	86
004ef314ec722...	1-3083bc71dc7...	28.8999996185...	69.1999969482...	Not Detected	0	85
004ef314ec722...	1-a3f25e5a0d61...	28.8999996185...	69.0999984741...	detected	0	85
004ef314ec722...	1-105a2fb2248b...	28.8999996185...	69.3000030517...	Not Detected	0	86
004ef314ec722...	1-3083bc71dc7...	28.8999996185...	69.1999969482...	Not Detected	0	85



The screenshot shows the continuation of the sensor data table in the Cloudant Dashboard. The columns remain the same: _id, _rev, Temperature, Humidity, Motion, Flame, and Gas_Sensor. The data is paginated, showing documents 1 through 100.

_id	_rev	Temperature	Humidity	Motion	Flame	Gas_Sensor
0b66ce1c8cbfc7...	1-b72ea9bab2d...	32	60.7000007629...	Not Detected	0	296
0b66ce1c8cbfc7...	1-4774628b9de...	31.8999996185...	60.5999984741...	detected	0	270
0b66ce1c8cbfc7...	1-3e114a1493c0...	31.7000007629...	60.5	detected	0	186
0b66ce1c8cbfc7...	1-4a74bb15f3a1...	31.7000007629...	60.5	detected	0	181
0b66ce1c8cbfc7...	1-7f25260dddb...	31.6000003814...	60.7999992370...	detected	0	162
0b66ce1c8cbfc7...	1-bcad43612c8e...	31.5	60.2999992370...	Not Detected	0	149
0b66ce1c8cbfc7...	1-113bf24c1441...	31.5	61	Not Detected	0	138
0b66ce1c8cbfc7...	1-6e02183746a2...	31.5	60.9000015258...	Not Detected	0	134
0b66ce1c8cbfc7...	1-fa03228da757...	31.5	60.7000007629...	Not Detected	0	128
0b66ce1c8cbfc7...	1-360c33b5107f...	15.6999998092...	3307.30004882...	Not Detected	0	124
0b66ce1c8cbfc7...	1-3c8fa0c312bfc...	31.3999996185...	61.0999984741...	Not Detected	0	113
0b66ce1c8cbfc7...	1-c75e6f3cd5b9...	31.3999996185...	60.9000015258...	Not Detected	0	109
0b66ce1c8cbfc7...	1-58d071bb494...	31.3999996185...	60.7999992370...	Not Detected	0	106
0b66ce1c8cbfc7...	1-24b658af2153...	31.2999992370...	60.9000015258...	Not Detected	0	105
0b66ce1c8cbfc7...	1-6a9454fd3970...	31.2999992370...	60.7000007629...	Not Detected	0	104
0b66ce1c8cbfc7...	1-f7df8ed79178...	31.2999992370...	61	Not Detected	0	101

Sheets:

LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT

Project Data ☆ □ ☰
File Edit View Insert Format Data Tools Add-ons Help Last edit was 11 days ago

100% 123 Times New... 24 B I A

Temperature(C)	Humidity(%)	Gas	Motion	Fire	Fan	Light
31.5	61.09999847		127 Detected		0 ON	OFF
31.60000038	61.40000153		127 Detected		0 ON	OFF
31.5	61.29999924		127 Detected	Detected	ON	OFF
31.5	61.8		127 Not Detected	Detected	ON	OFF
31.5	61.8		127 Not Detected	Detected	ON	OFF
31.5	61.8		127 Not Detected	Detected	ON	OFF
31.5	61.8		125 Not Detected	Detected	ON	OFF
31.70000076	61.8		125 Not Detected	Detected	ON	OFF
31.70000076	61.8		125 Not Detected	Detected	ON	OFF
31.70000076	61.8		125 Not Detected		0 ON	OFF
31.70000076	61.09999847		125 Not Detected		0 ON	OFF
31.5	61.09999847		127 Not Detected		0 ON	OFF
31.5	61.09999847		127 Not Detected		0 ON	OFF
31.5	61.09999847		127 Detected		0 ON	OFF
31.5	61.09999847		127 Detected		0 ON	OFF
31.5	61.6		127 Detected		0 ON	OFF
31.5	61.6		127 Detected		0 ON	OFF
15.8	61.6		126 Detected	Detected	OFF	ON
15.8	61.6		126 Detected	Detected	OFF	ON
15.8	61.6		126 Detected	Detected	OFF	ON
15.8	61.6		126 Detected	Detected	OFF	ON
16.4	61.6		126 Detected	Detected	OFF	ON
16.4	61.6		126 Detected	Detected	OFF	ON
16.4	61.6		120 Detected	Detected	OFF	ON
16.4	45.7		120 Detected		0 OFF	ON
16.4	45.7		120 Not Detected		0 OFF	ON
16.4	45.7		120 Not Detected		0 OFF	ON
40.2	45.7		120 Not Detected		0 OFF	ON

Sheet1

Data ☆ □ ☰
File Edit View Insert Format Data Tools Add-ons Help Last edit was 11 days ago

100% 123 Times New... 24 B I A

Temperature(C)	Humidity(%)	Gas	Motion	Fire	Fan	Light
52.20000076	61.09999847		200 Detected		0 ON	OFF
52.20000076	61.40000153		230 Detected		0 ON	OFF
52.20000076	61.29999924		230 Detected	Detected	ON	OFF
52.20000076	61.8		230 Not Detected	Detected	ON	OFF
52.20000076	61.8		200 Not Detected	Detected	ON	OFF
51.29999924	61.8		250 Not Detected	Detected	ON	OFF
51.29999924	61.8		250 Not Detected	Detected	ON	OFF
51.29999924	61.8		250 Not Detected	Detected	ON	OFF
51.29999924	61.8		250 Not Detected	Detected	ON	OFF
51.29999924	61.8		250 Not Detected		0 ON	OFF
51.29999924	61.09999847		210 Not Detected		0 ON	OFF
48.20000076	61.09999847		210 Not Detected		0 ON	OFF
42.20000076	61.09999847		210 Not Detected		0 ON	OFF
42.20000076	61.09999847		210 Detected		0 ON	OFF
41.20000076	61.09999847		210 Detected		0 ON	OFF
40.20000076	61.6		210 Detected		0 ON	OFF
40.20000076	61.6		250 Detected		0 ON	OFF
39.79999924	61.6		250 Detected	Detected	OFF	ON
39.79999924	61.6		260 Detected	Detected	OFF	ON
39.79999924	61.6		260 Detected	Detected	OFF	ON
38.79999924	61.6		260 Detected	Detected	OFF	ON
38.79999924	61.6		250 Detected	Detected	OFF	ON
38.79999924	61.6		260 Detected	Detected	OFF	ON
37.70000076	45.7		260 Detected		0 OFF	ON
37.70000076	45.7		200 Not Detected		0 OFF	ON
37.70000076	45.7		200 Not Detected		0 OFF	ON
37.70000076	45.7		230 Not Detected		0 OFF	ON

Sheet1

Appendix A: Definitions, Acronyms and Abbreviations

- PIR : A passive infrared sensor
- IOT : Internet Of Things
- Food : Any nutritious substance that people or animals eat or drink or that plants absorb in order to maintain life and growth.

Appendix B: References

1. <https://ieeexplore.ieee.org/document/9073248>
2. <https://www.ijeat.org/wp-content/uploads/papers/v8i6/F9355088619.pdf>
3. <http://nebula.wsimg.com/4ce7d6b9a18e1399174c5fceab0326e4?AccessKeyId=DFB1BA3CED7E7997D5B1 disposition=0&alloworigin=1>
4. <https://www.irjet.net/archives/V7/i4/IRJET-V7I41124.pdf>
5. <https://www.ijsr.net/archive/v8i3/ART20196559.pdf>
6. <https://ijrti.org/papers/IJRTI1706080.pdf /2017>
7. http://www.ijresm.com/vol1%2Ciss4%2CApril18/IJRESM14_38.pdf /2018
8. <https://ieeexplore.ieee.org/document/8403537 /2018>
9. <https://medcraveonline.com/MOJFPT/monitoring-food-storage-humidity-and-temperature-data-using-iot.html /2018>
10. <http://www.ijstr.org/final-print/dec2019/Automated-Granary-Monitoring-And-Controlling-System-Suitable-For-The-Sub-saharan-Region.pdf /2019>
11. <https://ieeexplore.ieee.org/document/9231104 /2020>
12. <https://ieeexplore.ieee.org/abstract/document/9198549/authors#authors /2020>