1.INTRODUCTION

1.1 Overview

Creating Smart Health Monitoring System For Elderly People Using IOT. The Internet of Things (IoT) is how we describe the digitally connected universe of everyday physical devices. These devices are embedded with internet connectivity, sensors and other hardware that allow communication and control via the web.

1.2 Purpose

Taking care of our loved ones, this device helps to identify their health status in real time.The Heartbeat rate information knowing is very useful while doing exercise, studying, etc.

2. LITERATURE SURVEY

2.1 Existing problem

Many elderly people faced heart problems.some times they need proper surveillance under the doctor or by family members. Even now many youngsters face heart related problems. They want to track health status for proper treatment and for healthy life.

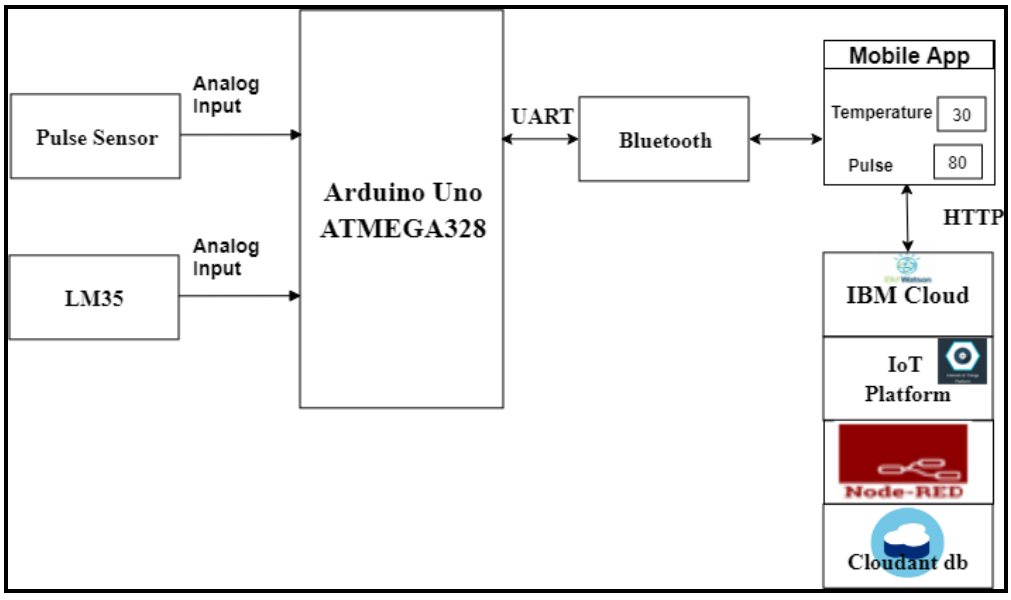
2.2 Proposed Solution

Nowadays with the help of IBM Watson internet of things (IOT) platform, we can build Smart Health Monitoring System which can be used In Hospitals for the the Patient with Heart Disease and Diabetes as well as Lower Blood Pressure also for elderly people.

3. THEORITITICAL ANALYSIS

3.1 Block diagram

3.2



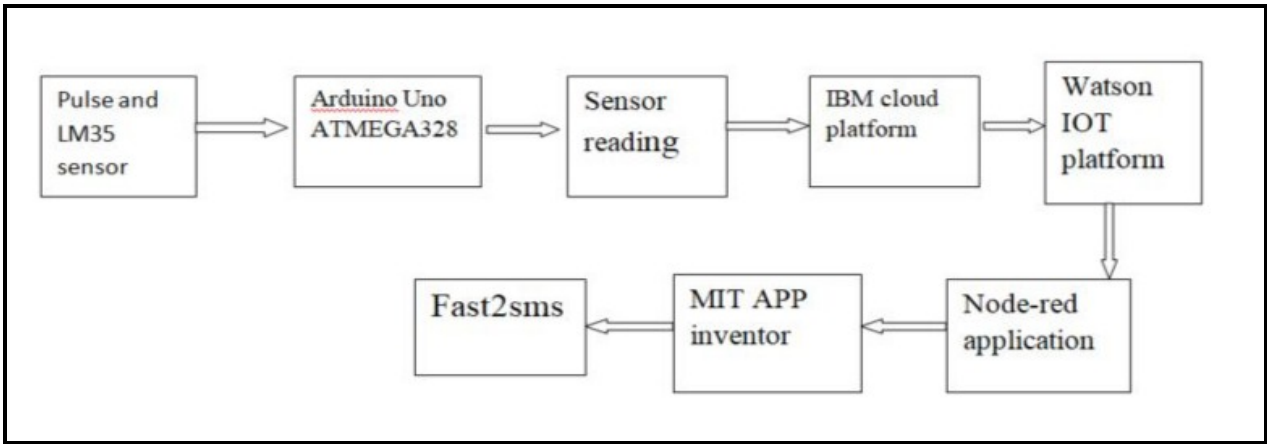
Hardware / Software designing

- We will take two sensors here i.e. Pulse and LM35 for pulse rate and
- temperature. Connect these two sensors with Arduino Uno ATMEGA328 Board.
- Theses sensor reading will be send to the cloud with the help of IBM cloud platform by creating IBM account.
- In IBM cloud, create watson IOT platform and get IBM Watson Device Credentials.
- These credentials are used in code for sending the data in IBM cloud.
- Create Node-red application service from IBM software and make connection through nodes to fetch the sensor data from the code also use Dashboard node for creating UI (Web App).
- Now with help of HTTP node data are to store in URL
- Through Node-red and this data will useful for fetching data for Mobile App.
- Create account in MIT APP inventor for the mobile app.
- In MIT APP inventor create front end and backend for an app.
- Create account in Fast2sms service for sending the message to the user or care person.

4. EXPERIMENTAL INVESTIGATIONS

a.Python concept
b.ANPR (openCV)
c.Communication protocol
d.Zigbee, RFID
e.the quality of service (QOS)
f.cloudant DB for Database
g.Flask
h.Json

5.FLOWCHART



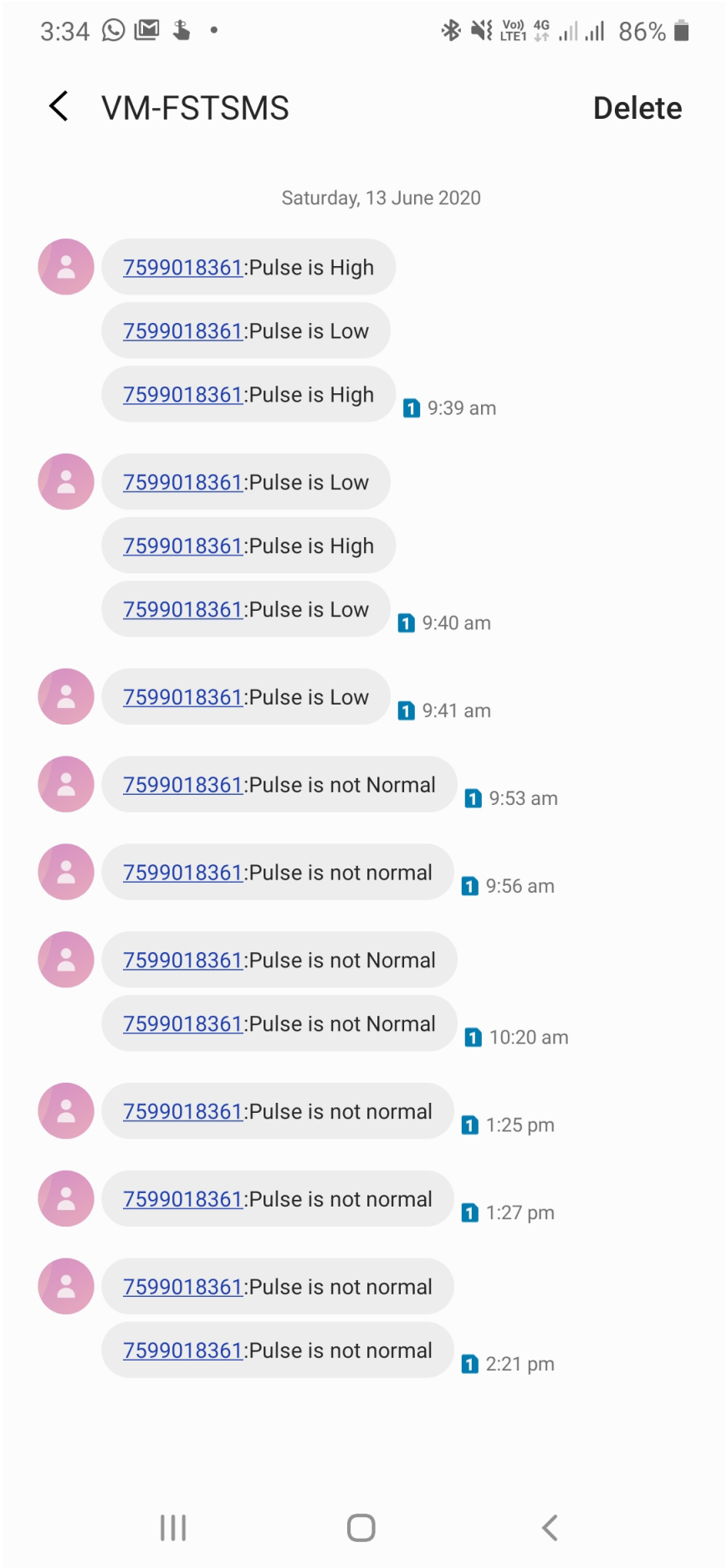
6.RESULT

Smart Health Monitoring System For Elderly People Using IBM Cloud is successfully completed.

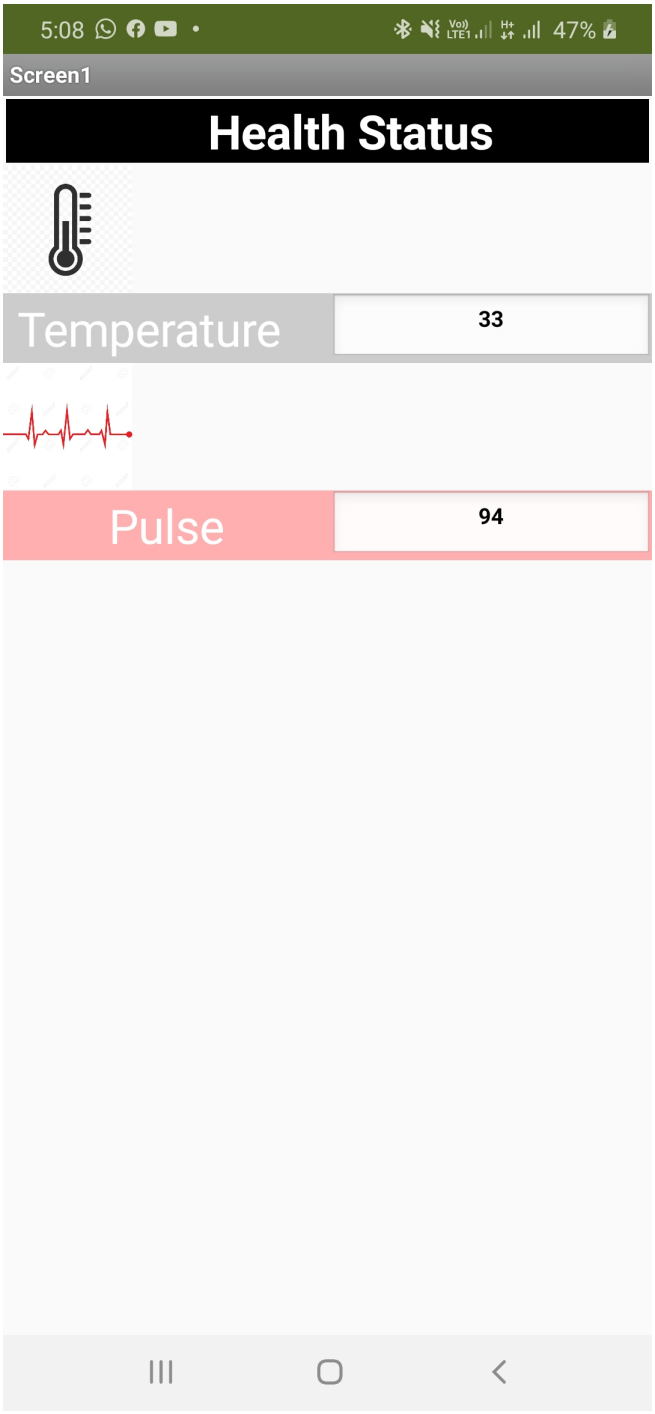
a. **Final Code Output:-**

```
Published Temperature = 36 C Pulse = 82 % to IBM Watson
{'Temperature': 32, 'Pulse': 78}
Published Temperature = 32 C Pulse = 78 % to IBM Watson
{'Temperature': 32, 'Pulse': 59}
Document successfully created.Published Temperature = 32 C
 Pulse = 59 % to IBM Watson
200
{'Temperature': 31, 'Pulse': 101}
Document successfully created.Published Temperature = 31 C
 Pulse = 101 % to IBM Watson
200
{'Temperature': 39, 'Pulse': 92}
Published Temperature = 39 C Pulse = 92 % to IBM Watson
{'Temperature': 33, 'Pulse': 83}
Published Temperature = 33 C Pulse = 83 % to IBM Watson
{'Temperature': 36, 'Pulse': 85}
Published Temperature = 36 C Pulse = 85 % to IBM Watson
{'Temperature': 37, 'Pulse': 102}
Document successfully created.Published Temperature = 37 C
 Pulse = 102 % to IBM Watson
200
```

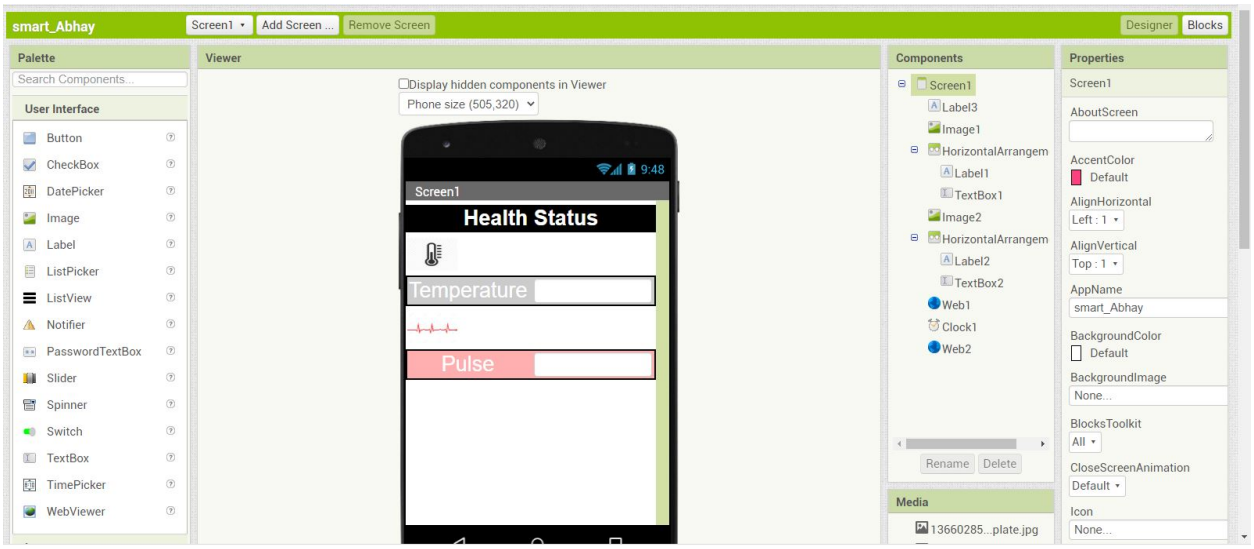
b. **SMS Notification on Mobile:-**


3:34
86%
VM-FSTSMS
Delete
Saturday, 13 June 2020
7599018361:Pulse is High
7599018361:Pulse is Low
7599018361:Pulse is High
9:39 am
7599018361:Pulse is Low
7599018361:Pulse is High
7599018361:Pulse is Low
9:40 am
7599018361:Pulse is Low
9:41 am
7599018361:Pulse is not Normal
9:53 am
7599018361:Pulse is not normal
9:56 am
7599018361:Pulse is not Normal
7599018361:Pulse is not Normal
10:20 am
7599018361:Pulse is not normal
1:25 pm
7599018361:Pulse is not normal
1:27 pm
7599018361:Pulse is not normal
7599018361:Pulse is not normal
2:21 pm

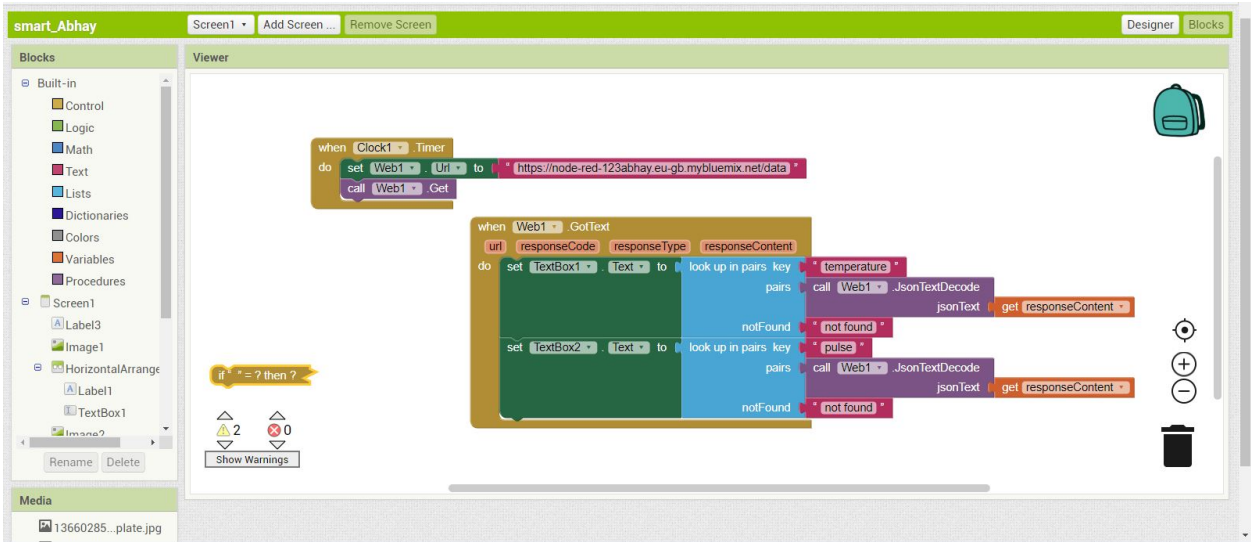
C.MIT APP Inventor Output:-



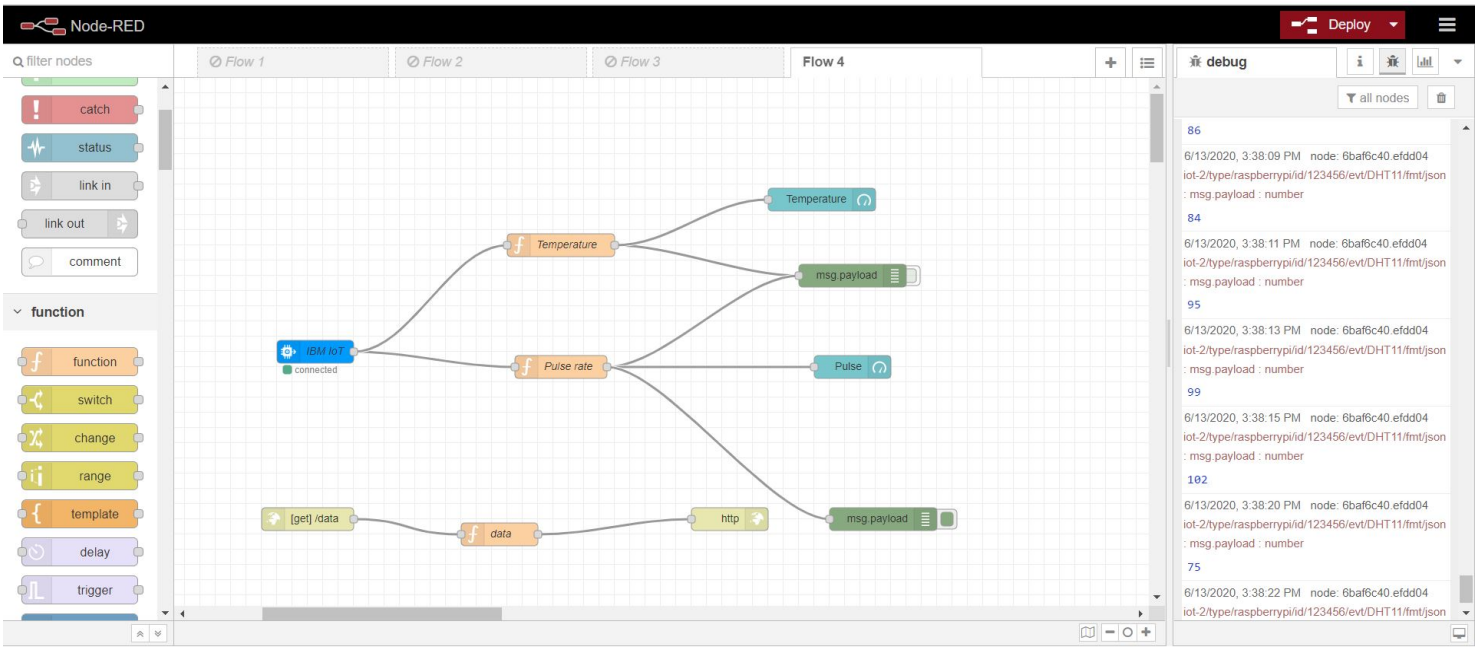
d. MIT APP Inventor Frontend :-



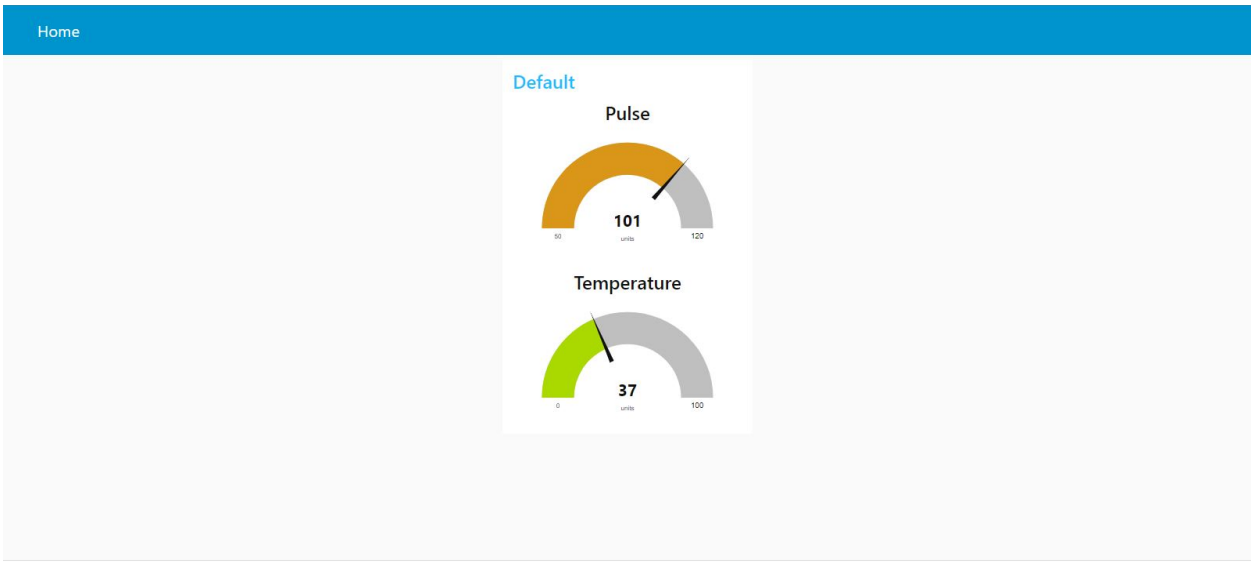
e. MIT APP Inventor Backend :-



f. Node-RED Application:-



i. Node-RED UI (Web App):-



7. ADVANTAGES

- **Reduced Errors** – IoT allows for the accurate collection of data, automated workflows and minimized waste, but most importantly it reduces the risk of error.

- **Decreased costs** – With IoT, patient monitoring can be done in real-time, drastically cutting down the need for doctors going out and making visits. Connected home care facilities will also help reduce hospital stays and re-admissions.

- **Better patient experience** – A connected healthcare system creates an environment that meets each patient's needs. Dedicated procedures, enhanced treatment options and improved diagnosis accuracy make for a better patient experience.

- **Improved disease management** – With real-time data healthcare providers can continuously monitor patients. This means that they can spot any disease before it spreads and becomes serious.

DISADVANTAGES

- **Security and privacy**- Security and privacy remain a major concern deterring users from using IoT technology for medical purposes, as healthcare monitoring solutions have the potential to be breached or hacked. The leak of sensitive information about the patient's health and location and meddling with sensor data can have grave consequences, which would counter the benefits of IoT.

- **Risk of failure**- Failure or bugs in the hardware or even power failure can impact the performance of sensors and connected equipment placing healthcare operations at risk. In addition, skipping a scheduled software update may be even more hazardous than skipping a doctor checkup.

8.APPLICATIONS

- Smart Health Monitoring System can be used for Sleep Tracking.
- This can be used for Anxiety monitoring.
- This can be used in remote patient monitoring or alarm system.
- This can be used in Health bands.
- This can be used in complex gaming consoles.

9.CONCLUSION

Today, many services can be reached with internet technology and the number of applications that use this technology is constantly increasing. IoT technology is expanding day by day to include different sectors and applications. One of them is the smart health sector and this sector offers incredible opportunities for us with new applications. The monitoring of the patients, who should be kept under constant surveillance, in the hospital environment is very difficult with the existing infrastructure and methods. Patients under surveillance in hospitals are dependent on bedding and this makes the patients uncomfortable. Many health problems that require early diagnosis may cause vital problems for the patient because they cannot be monitored on time.

10.FUTURE SCOPE

According to a Business Insider report, the market for IoT healthcare technology will rise to $400 billion by 2022. Such growth will be due to the increasing demand, the improvement of 5G connectivity and IoT technology and the growing acceptance of healthcare IT software. The plans of tech giants like Apple, Google and Samsung to invest in bridging the gap between fitness tracking apps and actual medical care are sure to contribute to the process too.

Despite the downsides, further digital transformation in healthcare is inevitable and the concept of IoT will continue to capture and change the landscape of healthcare services. Thus, it seems to be high time to look beyond the challenges and embark on the journey to connected healthcare devices.

11.BIBILOGRAPHY

-IBM Cloud
-Node-Red Application
-IBM Watson IOT Platform

-FAST2SMS(Bulk SMS Service Provider)
- Mobile App With (HTTP Requests To Communicate)
-UI(Web App)

APPENDIX

A. Source code

1. https://cloud.ibm.com/registration
2. https://thesmartbridge.com/documents/pdf/Node-red-Creation.pdf
3. https://thesmartbridge.com/documents/pdf/IoT-Device-Creation.pdf
4. https://www.fast2sms.com/
5. https://docs.fast2sms.com/
6. https://docs.fast2sms.com/#bulk-sms-api
7. https://flows.nodered.org/node/node-red-dashboard
8. https://developer.ibm.com/recipes/tutorials/ui-dashboard-for-iot-device-data-using-node-red/

Final Project Code:-

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random #for sensor values
import requests # for sending sms
#Provide your IBM Watson Device Credentials
organization = "blavqr"
deviceType = "raspberrypi"
deviceId = "123456"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO

def myCommandCallback(cmd):
        print("Command received: %s" % cmd.data)
```

```
        print(type(cmd.data))
        i=cmd.data['command']
        if i=='lighton':
                print("light is on")
        elif i=='lightoff':
                print("light is off")

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:

        pulse=random.randint (58, 102)#normal pulse rate of
human body is btw (60- 100)
        #print(pulse)
        temp = random.randint (30, 40)
        #Send Temperature & Humidity to IBM Watson
        data = { 'Temperature' : temp, 'Pulse': pulse }
        print (data)
```

```
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temp,
"Pulse = %s %%" % pulse, "to IBM Watson")

        success = deviceCli.publishEvent("DHT11", "json",
data, qos=0, on_publish=myOnPublishCallback)
        if (pulse>100 or pulse<60): #setting the if condition
for sending sms through fast2sms
            print(f"Document successfully created.")
            r =
requests.get('https://www.fast2sms.com/dev/bulk?authorization
=CT8hIwZG0p7ROvsZ1xg9KH9SwOfAVjswzoEx0vugk6axVZbI9Tyk7bVao0Gv
&sender_id=FSTSMS&message=Pulse%20is%20not%20normal&language=
english&route=p&numbers=9810768394')
            print(r.status_code)
        if not success:
            print("Not connected txo IoTF")
        time.sleep(2)

        deviceCli.commandCallback = myCommandCallback

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

...**THE END**...