HEALTH MONITORING SYSTEM FOR ELDERLY BED-RIDDEN PEOPLE

19Z604 - Embedded Systems Lab Report

Mukesh Balaji K (21Z323)
Nandikaa G (21Z325)
Samendhra G (21Z342)
Shreya Ramesh (21Z350)
Shreya Thiagarajan (21Z351)

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

Of Anna University

April 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

Table of contents

Problem Statement …… ……3

Schematic Diagram …….. ……3

Annotated C code …… ……4

Output Screenshots …….. ……6

PROBLEM STATEMENT:

In light of the demographic shift towards an aging population and the increasing need for continuous health monitoring among elderly individuals confined to bed due to critical conditions, a pressing challenge emerges in the healthcare sector. Current medical systems often lack the capacity to provide real-time, remote monitoring of vital signs, leading to delays in necessary interventions and heightening the vulnerability of bedridden patients to adverse outcomes. Additionally, the complexity and high costs associated with existing monitoring solutions hinder their widespread adoption and scalability, exacerbating the issue.

Therefore, there is an urgent call for the development of an accessible, user-friendly, and comprehensive health monitoring framework tailored specifically for elderly bedridden individuals. This framework must be capable of seamlessly tracking a range of vital parameters, including temperature, humidity, motion, and heartbeat, while enabling remote access for caregivers and medical professionals to intervene promptly when necessary. By incorporating alert mechanisms, this envisioned system aims to usher in a new era of proactive healthcare management, centered on early anomaly detection and swift intervention, thereby improving patient outcomes and easing the burden on healthcare resources.

SCHEMATIC DIAGRAM:

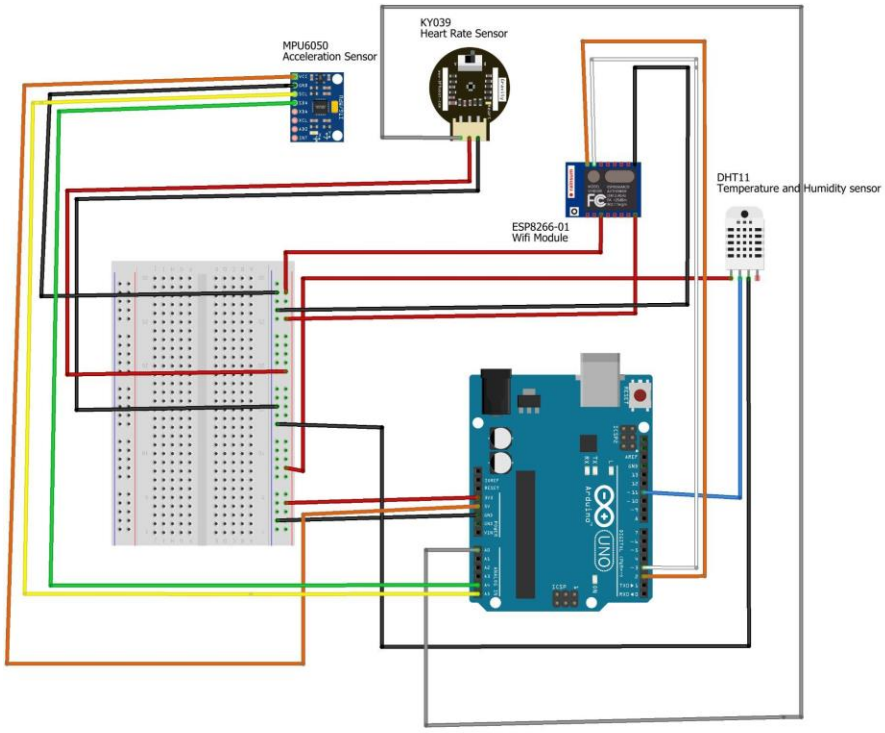

Fig 1: Schematic diagram of Elderly people monitoring system project

Components used in this project:

- Communication
    - ESP8266-01 Wi-Fi module
- Sensors
    - DHT11 Temperature and Humidity sensor – for monitoring temperature
    - MPU6050 Acceleration and Gyro sensor – for motion detection; sudden movements or fall
    - KY039 Heartbeat sensor – for monitoring heartbeat of the patient in care

These components are connected to the Arduino Uno board as specified in *Fig 1*. The sensors take the values from the environment and display them in the serial monitor for debug purposes. The data sensed by the sensor is sent to the ThingSpeak server (using the API Write key) which acts as the cloud from where using the API Read key the data is read and displayed in the web interface.

Pin Configuration:

- DHT11

Sensor pins	Arduino pins
S/D	Pin 11
Vin	3.3V
GND	GND

- KY039

Sensor pins	Arduino pins
IN	Pin A0
V*in*	3.3V in
GND	GND

- ESP8266-01

Sensor pins	Arduino pins
GND	GND
TX	Pin 2
CH_PD	3.3V
RX	Pin 3
ACC	3.3V

- MPU6050

Sensor pins	Arduino pins
Vcc	5V
GND	GND
SCL	A5
SDA	A4

Note:

- Due to the unavailability of the exact components in the Fritzing software, the RHT sensor has been used instead of the DHT, ESP8266 (16 pin model) instead of the 8 pin model, and another Heartbeat sensor has been used in the schematic diagram.
- Any update in realtime data takes a minimum of 16k msec to be shown in the ThingSpeak server.

C CODE:

```
#include <SoftwareSerial.h>
#include <dht11.h>
#include <Wire.h>
#include "MPU6050.h"

#define RX 2
#define TX 3
#define dht_apin 11

dht11 dhtObject;          // Create an object of the DHT11 temperature and humidity sensor
MPU6050 mpu;              // Create an object of the MPU6050 accelerometer and gyroscope

int16_t ax, ay, az;       // Variables to store accelerometer values
int16_t gx, gy, gz;       // Variables to store gyroscope values

String AP = "ReverGenie";         // WiFi network name (SSID)
String PASS = "Passpass";         // WiFi network password
String API = "GC9XYHZF9A7PM1FD";   // ThingSpeak API Key
String HOST = "api.thingspeak.com";
String PORT = "80";

int countTrueCommand;
int countTimeCommand;
boolean found = false;
int valSensor = 1;

SoftwareSerial esp8266(RX,TX);   // Create a software serial port named esp8266, connected to RX and TX pins

struct MyData {
  double X;
  double Y;
  double Z;
};
MyData data;

void setup() {
  Serial.begin(9600);             // Start serial communication with the computer
  mpu.initialize();               // Initialize MPU6050 sensor
  pinMode(A0, INPUT);             // Set pin A0 as input for heart rate sensor

  esp8266.begin(115200);          // Start communication with ESP8266 WiFi module
  sendCommand("AT",5,"OK");        // Send AT command to check ESP8266 connection
  sendCommand("AT+CWMODE=1",5,"OK");// Set ESP8266 mode to station mode
  sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",20,"OK"); // Connect to the specified WiFi network
}

void loop() {
 String getData = "GET /update?api_key="+ API +"&field1="+getTemperatureValue()+"&field2="+getHumidityValue()+"&field3="+getHeartRateValue()+"&field4="+
 sendCommand("AT+CIPMUX=1",5,"OK");   // Enable multiple connections
 sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+ PORT,15,"OK");  // Start a TCP connection to the ThingSpeak server
 sendCommand("AT+CIPSEND=0," +String(getData.length()+4),4,">");       // Send data length to ESP8266 for sending
 esp8266.println(getData);        // Send the data to ThingSpeak
 delay(16000);                     // Wait for data to be sent and received
 countTrueCommand++;
 sendCommand("AT+CIPCLOSE=0",5,"OK");  // Close the TCP connection
}

String getTemperatureValue(){
   dhtObject.read(dht_apin);      // Read temperature and humidity values from DHT11 sensor
   int temp = dhtObject.temperature; // Get temperature value from DHT11 sensor
   return String(temp);            // Return temperature value as a string
```

```
String getHumidityValue(){
   dhtObject.read(dht_apin);      // Read temperature and humidity values from DHT11 sensor
   int humidity = dhtObject.humidity; // Get humidity value from DHT11 sensor
   return String(humidity);        // Return humidity value as a string
}

String getHeartRateValue(){
  int sum = 0;
  for (int i = 0; i < 20; i++) {
    sum += analogRead(A0);        // Read analog value from heart rate sensor
  }
  float pulse = sum / 200.00;      // Calculate average pulse rate
  return String(pulse);            // Return pulse rate as a string
}

String getAccelerationMag(){
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);  // Read acceleration data from MPU6050 sensor
  data.X = map(ax, -17000, 17000, 0, 255 );      // Map X-axis accelerometer value to a range of 0-255
  data.Y = map(ay, -17000, 17000, 0, 255);       // Map Y-axis accelerometer value to a range of 0-255
  data.Z = map(az, -17000, 17000, 0, 255);       // Map Z-axis accelerometer value to a range of 0-255
  float mag = sqrt(data.X*data.X + data.Y*data.Y + data.Z*data.Z); // Calculate magnitude of acceleration
  return String(mag);             // Return magnitude as a string
}

void sendCommand(String command, int maxTime, char readReplay[]) {
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  while(countTimeCommand < (maxTime*1)) {
    esp8266.println(command);      // Send command to ESP8266
    if(esp8266.find(readReplay)) { // Check for expected response
      found = true;
      break;
    }
    countTimeCommand++;
  }

  if(found == true) {
    Serial.println("OK");
    countTrueCommand++;
    countTimeCommand = 0;
  }

  if(found == false) {
    Serial.println("Fail");
    countTrueCommand = 0;
    countTimeCommand = 0;
  }

  found = false;
}
```

PROJECT OUTPUT:

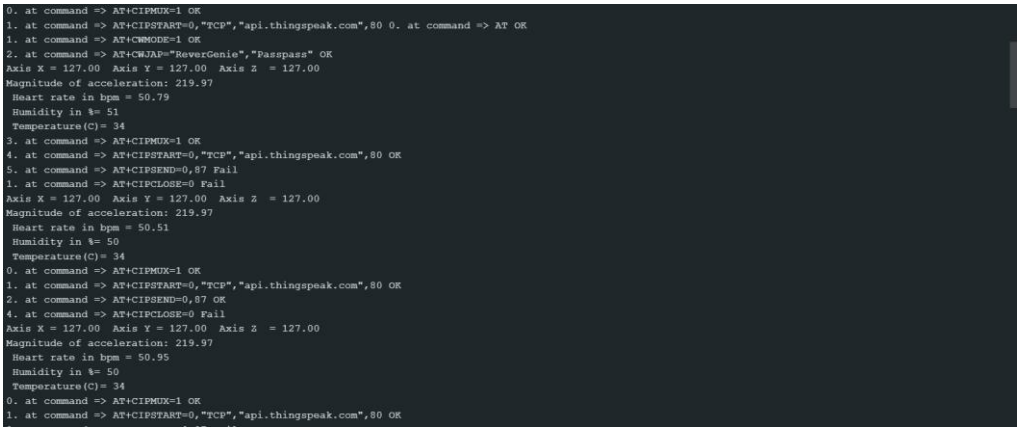

Fig 2. Serial monitor output for debug purpose

```
Magnitude of acceleration: 219.97
 Heart rate in bpm = 50.75
 Humidity in %= 50
 Temperature(C)= 34
0. at command => AT+CIPMUX=1 OK
1. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 OK
2. at command => AT+CIPSEND=0,87 OK
4. at command => AT+CIPCLOSE=0 OK
Axis X = 127.00  Axis Y = 127.00  Axis Z  = 127.00
Magnitude of acceleration: 219.97
 Heart rate in bpm = 51.10
 Humidity in %= 50
 Temperature(C)= 34
5. at command => AT+CIPMUX=1 OK
6. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 OK
7. at command => AT+CIPSEND=0,87 OK
9. at command => AT+CIPCLOSE=0 OK
Axis X = 127.00  Axis Y = 127.00  Axis Z  = 127.00
Magnitude of acceleration: 219.97
 Heart rate in bpm = 51.42
 Humidity in %= 50
 Temperature(C)= 34
10. at command => AT+CIPMUX=1 OK
11. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 OK
12. at command => AT+CIPSEND=0,87 Fail
1. at command => AT+CIPCLOSE=0 Fail
Axis X = 127.00  Axis Y = 127.00  Axis Z  = 127.00
Magnitude of acceleration: 219.97
 Heart rate in bpm = 43.51
 Humidity in %= 50
 Temperature(C)= 34
0. at command => AT+CIPMUX=1 OK
```

Fig 3. Serial monitor output for debug purpose

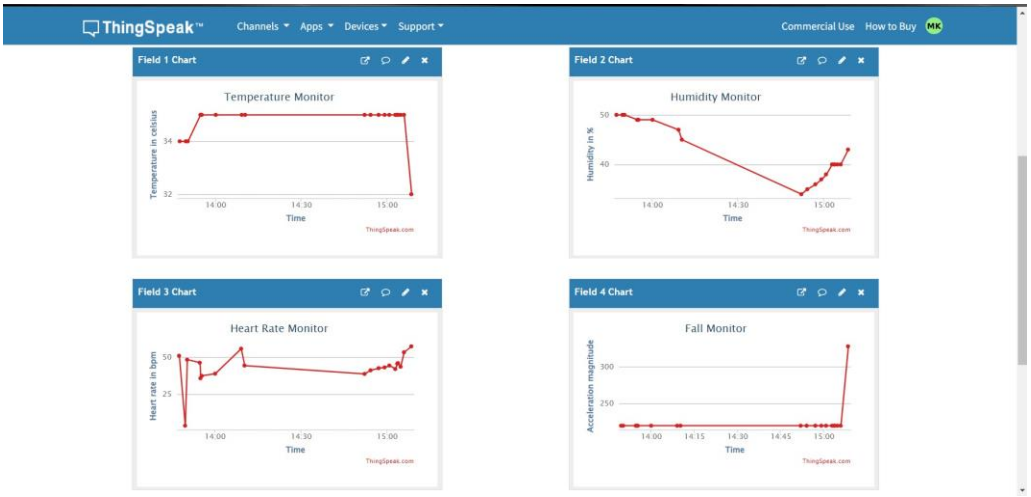

Fig 4. ThingSpeak cloud

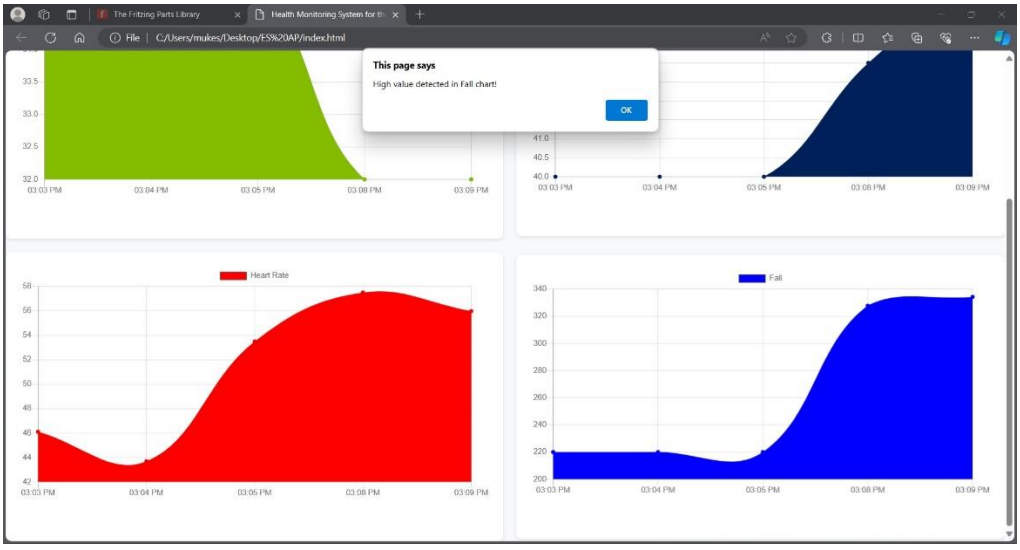

Fig 5. High acceleration alert on web interface

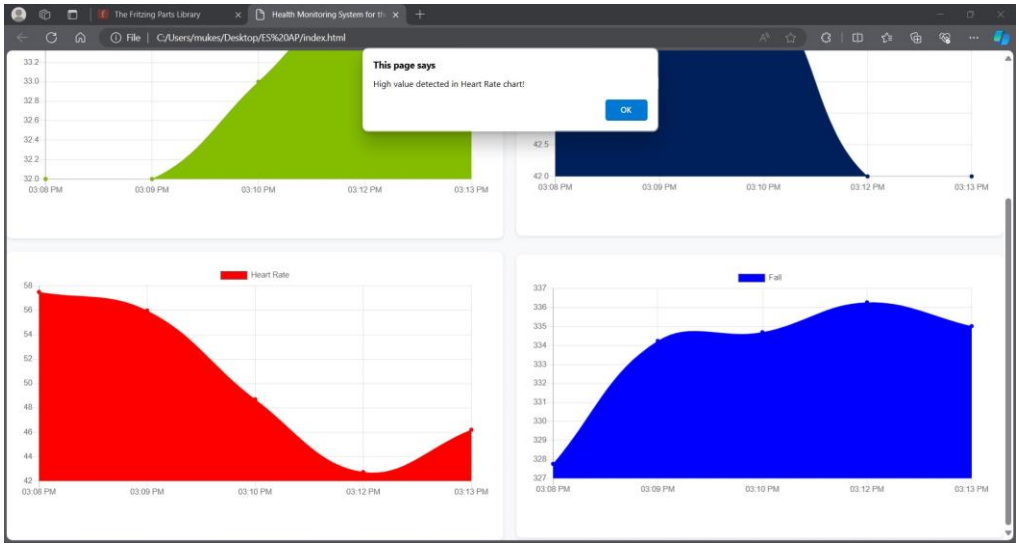

Fig 6. High heart rate alert on web interface