



PRESIDENCY UNIVERSITY

(Established under the Presidency University Act, 2013 of the Karnataka Act 41 of 2013)

School of Engineering
Department of Computer Science

Internet of Things

CSE - 220

A report on the project

Panic Alarm for Elders

Batch of 2018 CSE

Under the guidance of-
Prof. Manasa C.M

Contents

Sl.No	Title	Page
1.	Acknowledgement	3
2.	Abstract	4
3.	Introduction	5
4.	Statement & Literature Survey	6
5.	Architecture	7
6.	Circuit design	8
7.	Design of Web-App	10
8.	Arduino Code of the watch	14
9.	Inference & Conclusion	21
10.	Bibliography & Contributions	22

Acknowledgement

We would like to extend our warm and heartfelt appreciation and thanks to our course instructor Prof. Manasa C.M who has encouraged us constantly to work on new aspects of IOT and explore different domains. The insights that were received from our instructor were indeed very helpful and it reminded us how we can approach a real life situation and provide a solution to it.

Secondly, we would like to thank all our classmates who suggested many improvements onto our project and who were very supportive in terms of constructively giving suggestions. During the pandemic it was indeed tough for all of us to execute this by meeting at a place and we would like to thank the almighty for keeping us safe at these tough times.

Lastly, we would like to thank all the members who were directly or indirectly involved in the completion of our project and for meeting the deadlines of the reviews in time.

Thank you.

Abstract

- The core functionality of this device is to alert the care-takers when a person who is differently abled, elderly or are in need of help.
- The device is wearable and has a simple interface and is capable of sending a pre saved message template in case of distress.
- The use of this device is aimed for the elderly being the top priority followed by any individual needing assistance for safety purposes.

Members of the project

Sl. No.	Member Name	ID number
1	Sai Ram. K	20181CSE0621
2	Satyam Mourya	20181CSE0643
3	Shaiq Iqbal	20181CSE0660
4	Shivam Singh	20181CSE0669
5	Shlaghana J.S	20181CSE0670

Introduction

- There can be any sudden situation of panic. It could be because of an intruder entering our house or bad health status. Situations can be many for panicking and may vary from person to person.
- During these situations we employ the use of a panic alarm that alerts the provided care-taker to rush for help.
- Our alarm comes in the form of a wrist watch. When the watch is in the autonomous mode then it continuously monitors the user's pulse and the moment at which the pulse rate exceeds the pulse rate's threshold, panic message is sent to multiple people at the same time to inform the person who is wearing the watch had a panic attack.
- Since, the watch works in real time it informs the well-wishers of the user that the user recently had panic attack and provokes them to take necessary action.
- We all are in the race of being the best. We live in a society where people keep working but at the same time, they have to be effective, this creates a lot of pressure on people both physically and mentally. In this race we keep on ignoring our health which leads to numerous health problems, as a result people become victims of "panic attack".
- That's why we bring "The panic alarm" which brings out a creative solution to this problem. Our alarm works in two distinct ways: One is the manual mode and the other one is fully autonomous mode. The alarm sends panic message to the people via mail whose email ids are already feed into the system.

Statement

- The purpose of a panic alarm is to allow a person under distress to quickly and silently call for help in the event of an emergency. Panic alarms are also called "distress alarms", "hold-up alarms", or "panic buttons".
- Panic alarms are used when it may be unsafe or uncomfortable to call for help in other ways.
- For example, if a belligerent person is standing in your lobby, it may be unwise to further escalate the situation by picking up a phone to call for assistance. A panic alarm can provide a quick and convenient way to summon help without drawing attention

Literature Survey

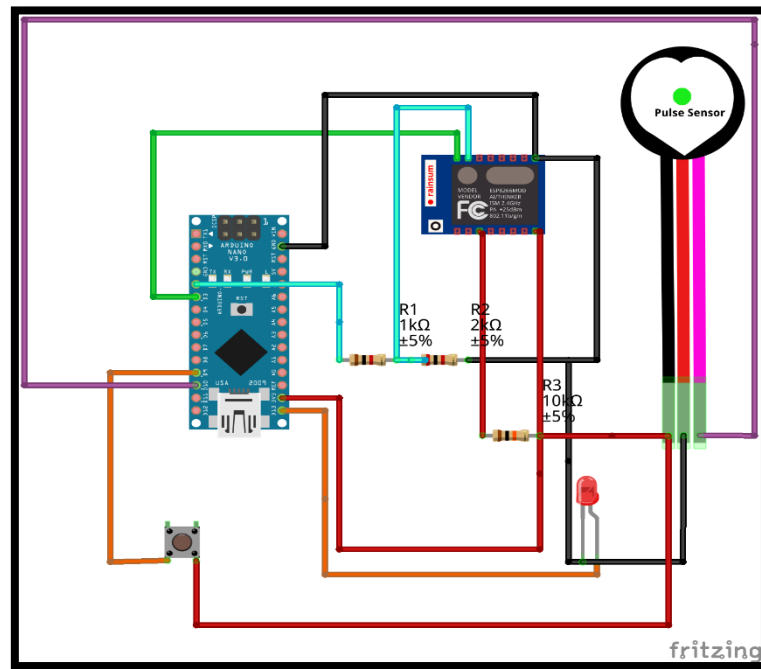
- Citation: -

1. B. Choudhury, T. S. Choudhury, A. Pramanik, W. Arif and J. Mehedi, "Design and implementation of an SMS based home security system," 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 2015, pp. 1-7, doi: 10.1109/ICECCT.2015.7226115.
2. S. D. Damayanti, M. Suryanegara, I. K. A. Enriko and M. I. Nashiruddin, "Designing A LoRa-Based Panic Button for Bali Smart Island Project," 2019 7th International Conference on Smart Computing & Communications (ICSCC), Sarawak, Malaysia, 2019, pp. 1-5, doi: 10.1109/ICSCC.2019.8843614.

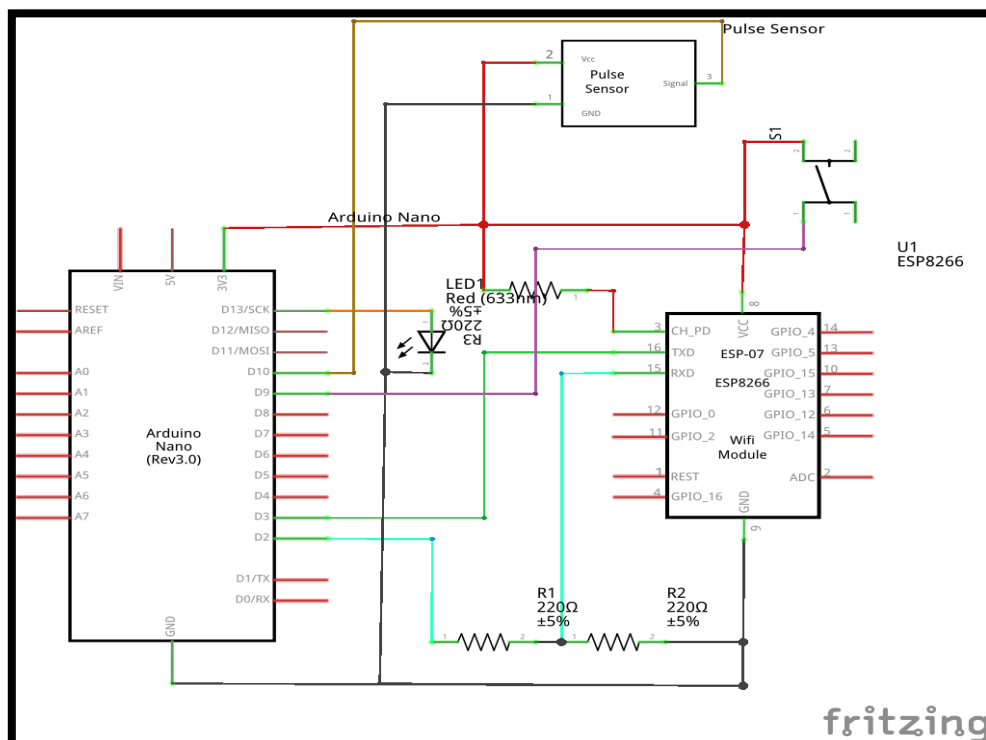
Architecture

The following are architecture diagrams depict the connections that can be embedded inside the watch prototype.

A. Circuit Diagram



B. Wireframe Diagram

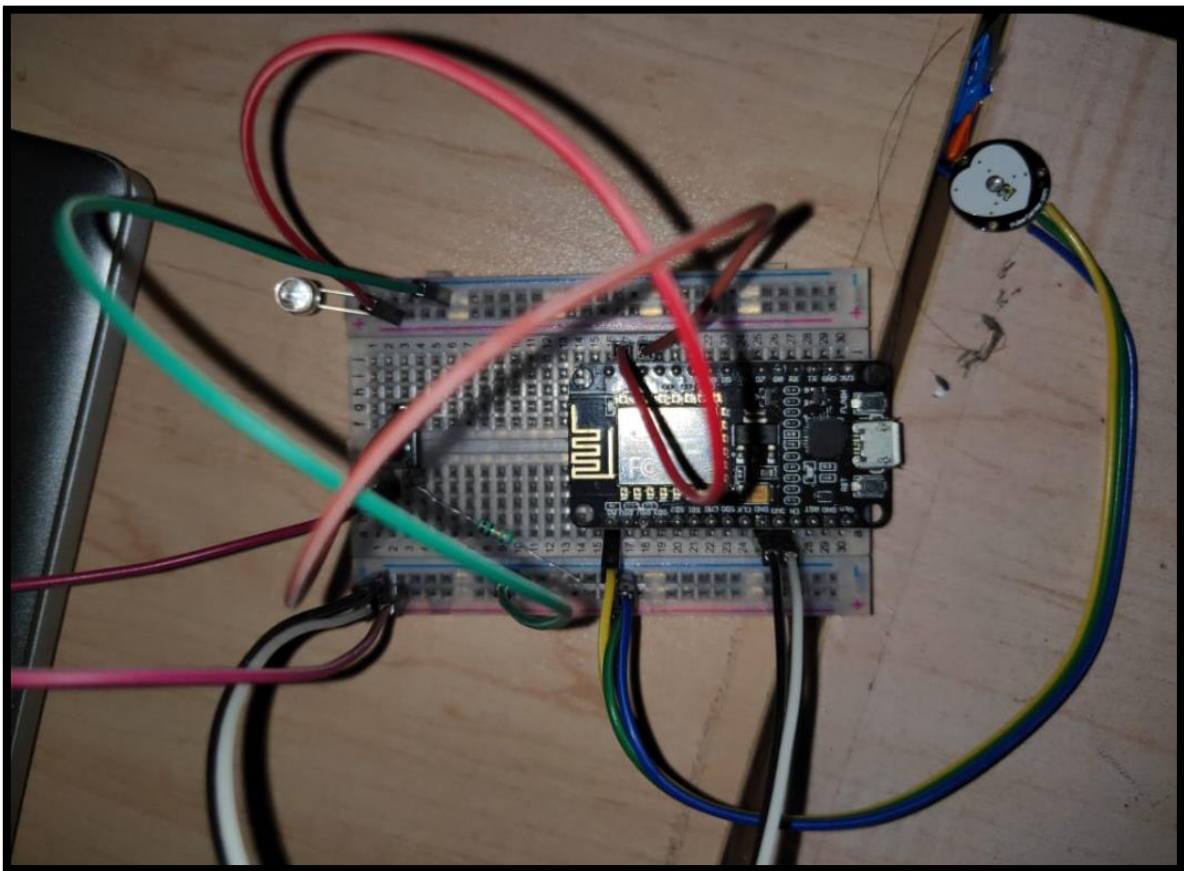


Circuit design

The initial circuit design that has the integrated collection of all the components used.

Components Used: -

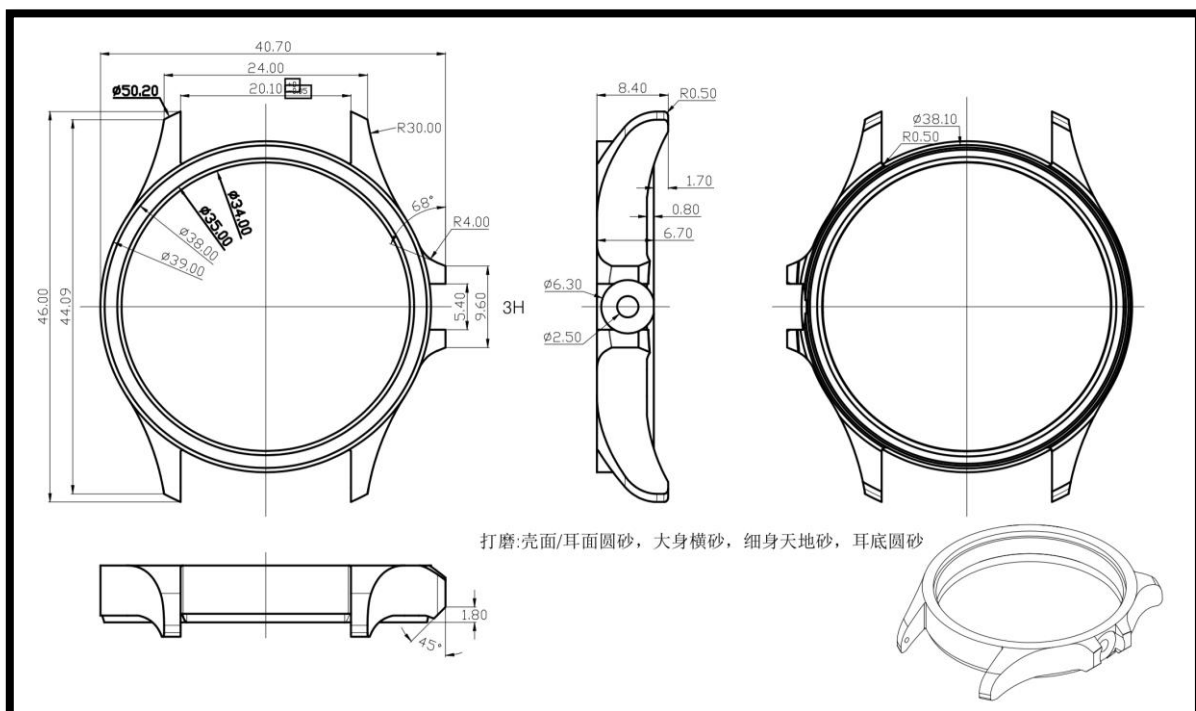
- LED
- Resistor 10 k Ω
- Pulse Sensor
- Jumper Wires
- ESP8266 Node MCU
- Breadboard Small



A] Connections of the components



B] Initial working model that would be developed further to meet the watch prototype.



C] Prototype of the final watch design.

Design of Web-App

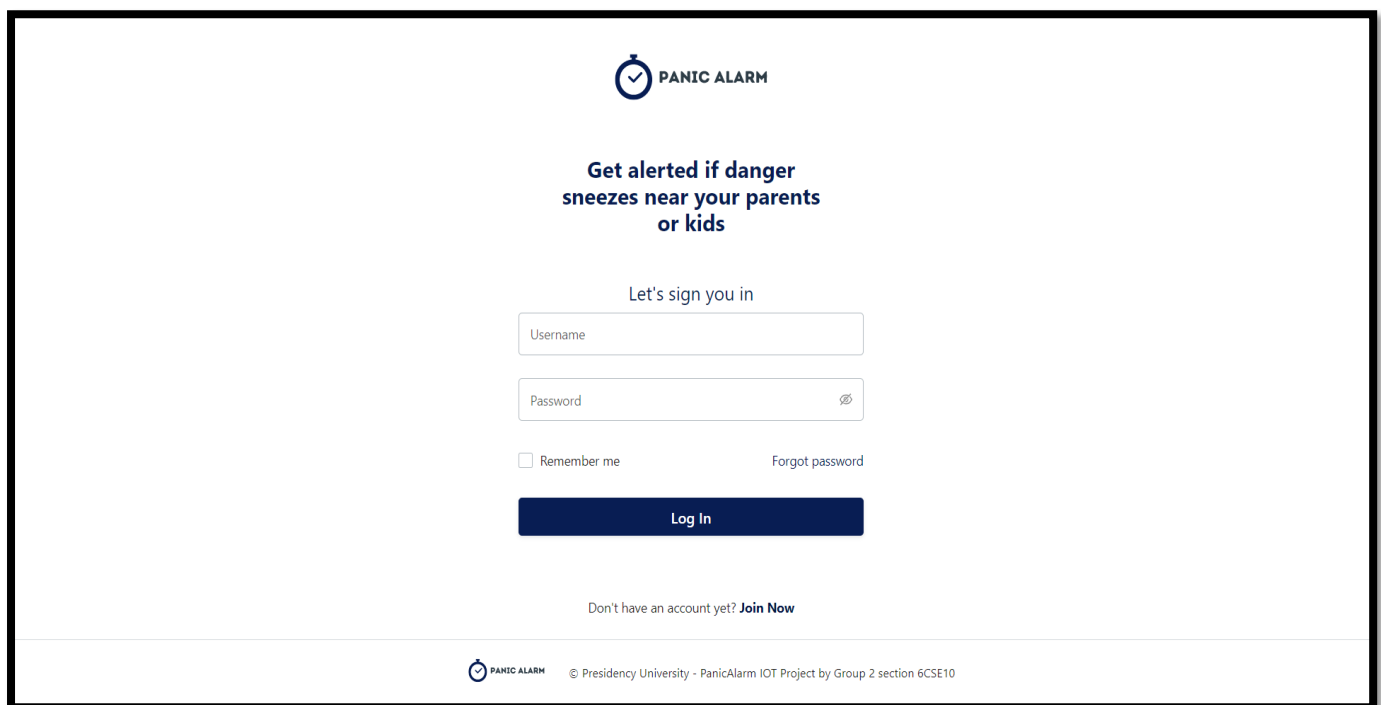
The watch monitors the pulse constantly and in the event of the button of the alarm being triggered the data monitored is sent to the cloud server which gives an interface to know the alert ID, time, location, directions along with an e-mail sent to the dependents who have registered in the website.

The design of the web-app was done using Next.JS and MongoDB.

The link to the entire source code of the app can be found below in the GitHub repository - [Panic Alarm Source Code](#) , [Panic Alarm Web-App](#)

The screenshots of the web-app can be found below.

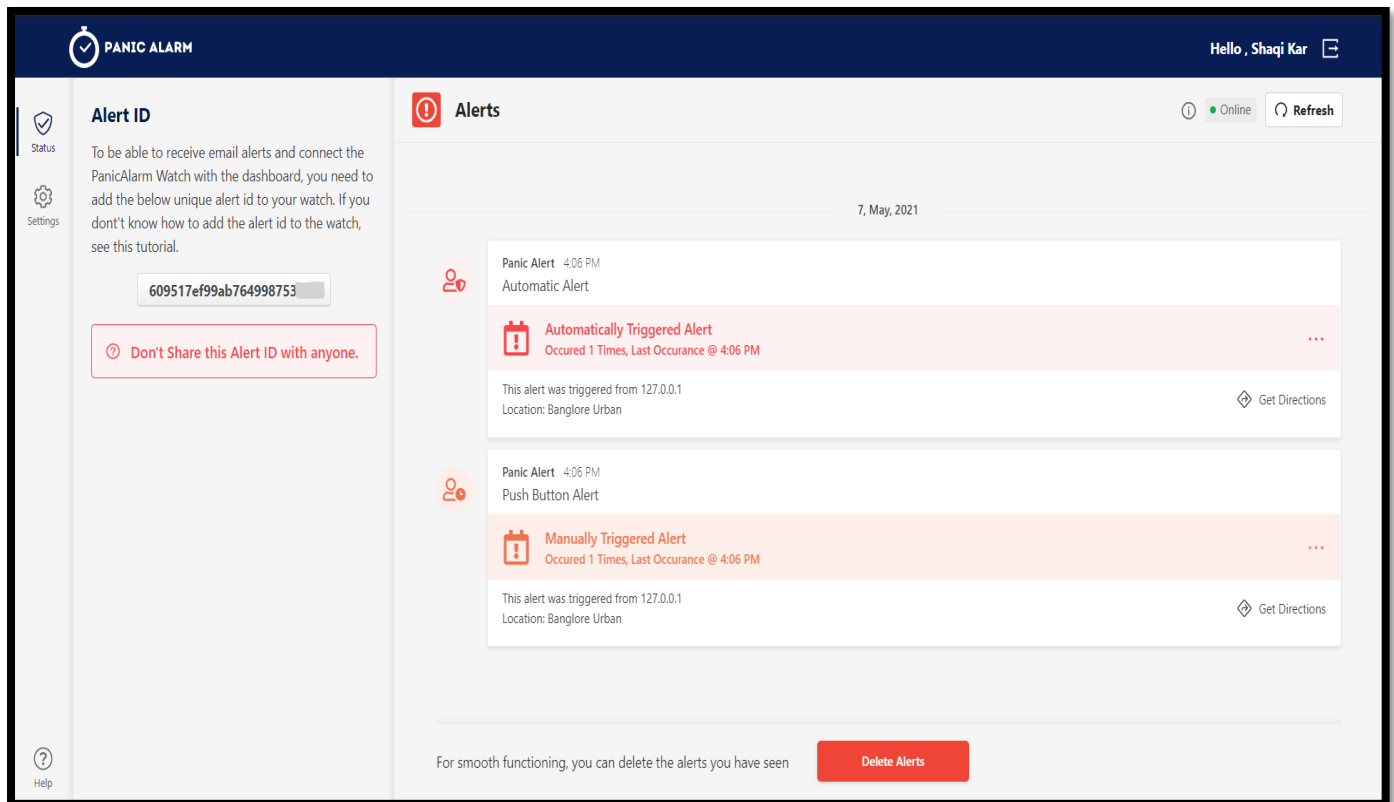
Log – In Page



The screenshot displays the login interface for the 'PANIC ALARM' application. At the top, there is a logo consisting of a clock icon with a checkmark inside, followed by the text 'PANIC ALARM'. Below the logo, a message reads: 'Get alerted if danger sneezes near your parents or kids'. Underneath this, it says 'Let's sign you in'. The login form includes a 'Username' input field, a 'Password' input field with a toggle icon for visibility, a 'Remember me' checkbox, and a 'Forgot password' link. A dark blue 'Log In' button is positioned below the password field. At the bottom of the form, there is a link that says 'Don't have an account yet? Join Now'. The footer of the page contains the 'PANIC ALARM' logo and the copyright notice: '© Presidency University - PanicAlarm IOT Project by Group 2 section 6CSE10'.

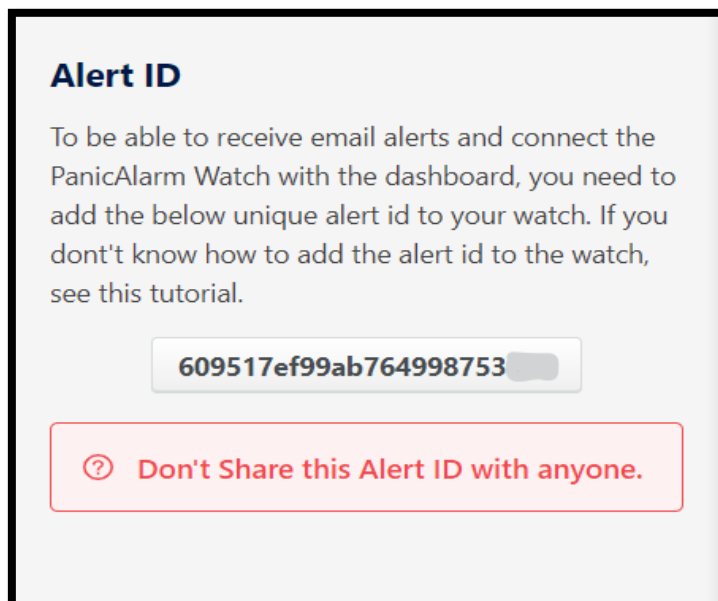
A. The login page of the WebApp that allows users to login or create a new account.

Alerts Dashboard



B. Alerts Dashboard that appears when a user logs into their account and displays the triggers.

Alert ID

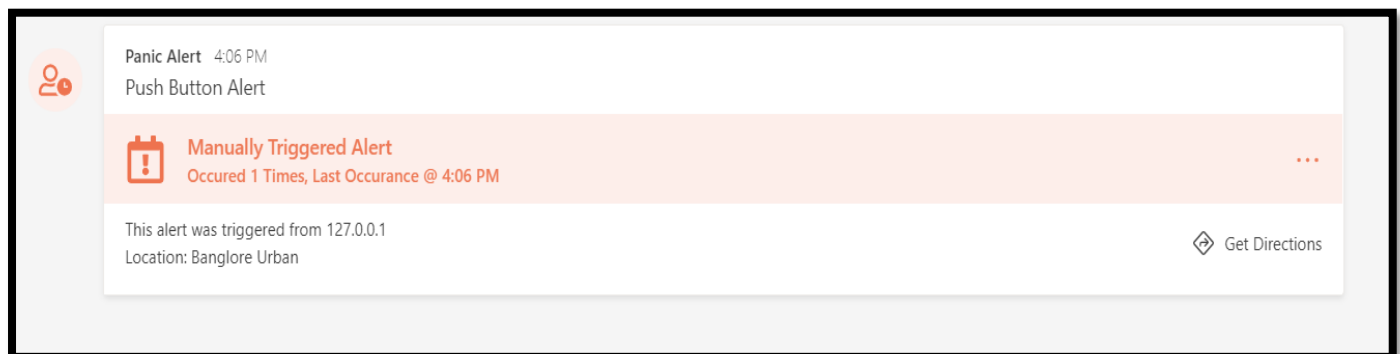


C. An alert ID pane that has a unique ID for each user and is expected to be confidential.

Alerts Display Pane

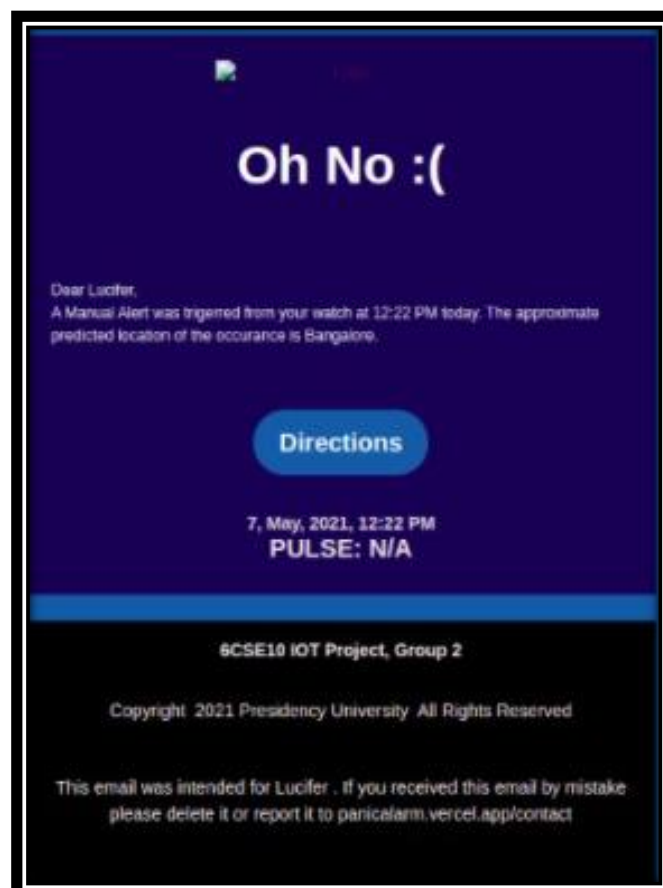
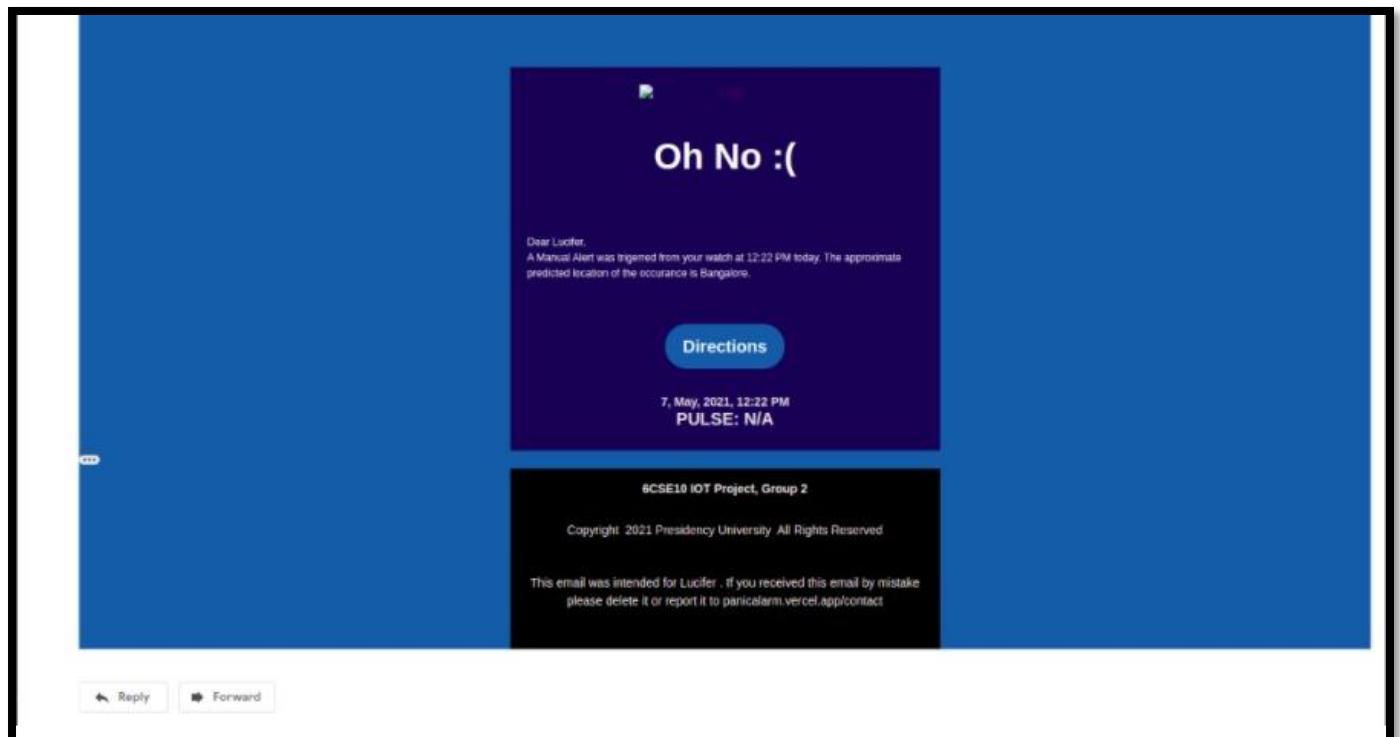


D. An alert that was generated automatically based on the monitoring.



E. An alert that was generated when a person triggered the push button.

Alert E-mail



F. Screenshot of the e-mail as sent by the panic alarm when the alert was triggered.

Arduino Code of the watch

The source code can also be found in the GitHub repository - [Panic Alarm](#)

```
/*
 * PanicAlarm Watch
 * IOT Project
 * Group 2 - 6CSE10
 * Presidency University, Bangalore
 */

#define __HALT__ while(1)
#define __DEBUG__ 1
#define __READ_RESPONSE__ 1
#define __max(i, j, k) i > j? (i > k? i: k): (j > k? j: k)

#include <Arduino.h>
#include <EEPROM.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <WiFiManager.h>
#include <Scheduler.h>

void __log__(String msg);
void sendDataToHost(int pulse);
int serializeHeartBeat(int counts[]);

// Global Constants
const int MANUAL_TRIG_BUTTON = 4;
const int INDICATOR_LED = 5;
const int EEPROM_SIZE = 512;
const int ALERT_ID_LENGTH = 24;
const int HTTP_PORT = 443;
const char *WATCH_HOTSPOT_NAME = "PanicAlarm Watch - 6CSE10";
String API_HOST = "panicalarm.vercel.app";
const char *ALERT_ID = "";

// Network Configuration
WiFiManager wifiManager;
WiFiClientSecure client;

// Sensor Configurations And Functions

const int PULSE_SENSOR_PIN = A0;
const int PULSE_SENSOR_THRESHOLD = 550;
```

```
const int PULSE_SENSOR_INTERVAL_TIMEOUT = 10000; // 10 Seconds

const int BPM_LOW = 55;
const int BPM_HIGH = 110;

float PULSE__Value = 0;
int PULSE__HeartBeat = 0;
boolean PULSE__Counted = false;
int PULSE__IntervalCount = 9;
int PULSE__SerializeFrequency = 6;
unsigned long PULSE__MonitorStartTime = 0;

// Watch States
int STATE__ManualTrigButton = 0;

// Tasks
class TASK__AnalogTrigg : public Task {
public:
    void setup() {
        pinMode(MANUAL_TRIG_BUTTON, INPUT);
        pinMode(INDICATOR_LED, OUTPUT);
    }
    void loop() {

        digitalWrite(INDICATOR_LED, LOW);

        STATE__ManualTrigButton = digitalRead(MANUAL_TRIG_BUTTON);
        if (STATE__ManualTrigButton) {

            digitalWrite(INDICATOR_LED, HIGH);
            sendDataToHost(-1);
            // Halt for atleast 1 minute
            delay(1000 * 60);
        }
    }
} __AnalogTriggTask;

class TASK__PulseSensor : public Task {
public:
    void loop() {

        /*
        * Pulse Sensor doesn't provide accurate reading, so the watch
        * will monitor the pulse readings for a minute. In one minute
        * it will take 6 readings and the average of the 6 will be
        * considered. The sensor is not that accurate and there are some
        * unexpected behaviours with it. So it can be said that an alert will
        * always be a successfull outcome. But the chance of hapenning that is less.
        */
    }
}
```

```
*/

int PULSE__Counts[PULSE__SerializeFrequency];

for (int i=0; i<PULSE__SerializeFrequency; i++) {
    PULSE__MonitorStartTime = millis();
    while (millis() < PULSE__MonitorStartTime + PULSE_SENSOR_INTERVAL_TIMEOUT) {
        PULSE__Value = analogRead(PULSE_SENSOR_PIN);
        if (PULSE__Value > PULSE_SENSOR_THRESHOLD && !PULSE__Counted) {
            PULSE__IntervalCount++;
            PULSE__Counted = true;
            delay(50);
        }
        else if (PULSE__Value < PULSE_SENSOR_THRESHOLD) {
            PULSE__Counted = false;
        }
        yield();
    }

    PULSE__Counts[i] = PULSE__IntervalCount * 6;
    PULSE__IntervalCount = 0;

    __log__("Heart Beat = " + String(PULSE__Counts[i]));
}

PULSE__HeartBeat = serializeHeartBeat(PULSE__Counts);
yield();

if (PULSE__HeartBeat >= 0 && (PULSE__HeartBeat <= BPM_LOW || PULSE__HeartBeat >=
BPM_HIGH))
    sendDataToHost(PULSE__HeartBeat);

    delay(1000 * 60);
}
} __PulseSensorTask;

void setup() {

    if (__DEBUG__)
        Serial.begin(9600);

    pinMode(BUILTIN_LED, OUTPUT);
    digitalWrite(BUILTIN_LED, HIGH);

    EEPROM.begin(EEPROM_SIZE);

    // On startup check if ALERT ID exists in the memory
```



```
if (char(EEPROM.read(0)) == 'A') {
    for (int i=1; i<=ALERT_ID_LENGTH; i++)
        ALERT_ID += char(EEPROM.read(0x0F+i));
}

__log__("[ALERTID] " + String(ALERT_ID));

WiFiManagerParameter alertId("alertid", "Alert ID", ALERT_ID, 40);
wifiManager.addParameter(&alertId);
wifiManager.autoConnect(WATCH_HOTSPOT_NAME);

__log__("[WIFI] Connected");

digitalWrite(BUILTIN_LED, LOW);

const char *TEMP_ALERT_ID = alertId.getValue();
__log__(String(TEMP_ALERT_ID));

if (strlen(TEMP_ALERT_ID) == ALERT_ID_LENGTH && strcmp(TEMP_ALERT_ID, ALERT_ID) != 0) {
    ALERT_ID = TEMP_ALERT_ID;
    EEPROM.write(0, 'A');
    for(int i=1; i <= ALERT_ID_LENGTH; i++) {
        EEPROM.write(0x0F+i, ALERT_ID[i]);
    }
    EEPROM.commit();
}
else {
    // Indicate there is an error by glowing the LED
    // and bring the board to a halt state
    __HALT__{
        digitalWrite(INDICATOR_LED, HIGH);
    };
}

client.setInsecure();

Scheduler.start(&__AnalogTriggTask);
Scheduler.start(&__PulseSensorTask);
Scheduler.begin();
}

void loop() {
    /*
    * All the tasks [Manual Alert Triggering & Pulse Monitor] are running
    * simultaneously as two separate processes, so loop function is not
```

```
* required in this case.
*/
}

/*
 * Function : sendDataToHost
 * Sends a GET request to the panicalarm server to
 * generate alerts. @param pulse indicates weather it will
 * be an auto or a manual alert.
 *
 */
void sendDataToHost(int pulse) {

    if (!client.connect(API_HOST, HTTP_PORT)) {
        __log__(F("Connection failed"));
        return;
    }
    yield();

    // Send HTTP request
    client.print(F("GET "));

    // Pulse < 0 indicates a manual alert.
    // Pulse >= 0 indicates an automatic alert
    if (pulse < 0)
        client.print("/api/alerts/" + String(ALERT_ID));
    else
        client.print("/api/alerts/" + String(ALERT_ID) + "?pulse=" + pulse);
    client.println(F(" HTTP/1.1"));

    // Headers
    client.print(F("Host: "));
    client.println(API_HOST);
    client.println(F("Cache-Control: no-cache"));

    if (client.println() == 0) {
        __log__(F("Failed to send request"));
        return;
    }

    char status[32] = {0};
    client.readBytesUntil('\r', status, sizeof(status));

    if (strcmp(status, "HTTP/1.1 200 OK") != 0) {
        __log__(F("Unexpected response: "));
        __log__(String(status));
    }
}
```

```
        return;
    }

    char endOfHeaders[] = "\r\n\r\n";
    if (!client.find(endOfHeaders)) {
        __log__(F("Invalid response"));
        return;
    }

    /*
     *
     * This parts reads the response from the GET Request
     * It is not required in Production, as it will increase
     * the workload on the CPU.
     *
     */

    #ifdef __READ_RESPONSE__
        while (client.available() && client.peek() != '{') {
            char c = 0;
            client.readBytes(&c, 1);
            __log__(String(c));
        }

        while (client.available()) {
            char c = 0;
            client.readBytes(&c, 1);
            __log__(String(c));
        }
    #endif
}

/*
 * Function : serializeHeartBeat
 * It serializes the BPM counts of an interval to an average
 * value and maps it with the highest no of occurances of a
 * paticular kind [LOW, NORMAL, HIGH]. It can further improvised
 * using some advanced statistical techniques
 *
 */
int serializeHeartBeat(int counts[]) {
    int PulseSum = 0;
    int fLow = 0, fHigh = 0, fNorm = 0;
    for (int i=0; i<PULSE__SerializeFrequency; ++i) {
        int _p = counts[i];
        PulseSum += _p;
    }
}
```

```
    if (_p <= BPM_LOW ) fLow += 1;
    else if (_p >= BPM_HIGH) fHigh +=1;
    else fNorm += 1;
}
int avgHeartBeat = PulseSum / PULSE__SerializeFrequency;
if (
    (avgHeartBeat <= BPM_LOW && __max(fLow, fNorm, fHigh) == fLow)
    ||
    (avgHeartBeat >= BPM_HIGH && __max(fLow, fNorm, fHigh) == fHigh)
) {
    return avgHeartBeat;
}

return -1;
}

/*
 * Function : __log__
 * Logs data to the Serial Monitor only in
 * DEBUG or Dev Mode. Using Serial Monitor in
 * production can cause unexpected behaviour.
 *
 */
void __log__(String msg) {
    if(__DEBUG__){
        Serial.println(msg);
    }
}
```

Inference & Conclusion

Inference -

This project is about a panic alarm that can be made in a wearable size like watch or band. This works in two modes, manual as well as automatic using pulse sensor. Here we have used a push button for manual trigger of the alarm and the message will be sent to the provided email. On the other hand, we have installed pulse sensor, which collect the heartbeat and calculate the bpm. If the pulse rate is unusual like too high or too low, it will send mail to the provided email. This model works on both mode parallelly. It also keeps a track of our heart beat. This also has a function where, along with sending the message to the email, it also sends the latitude and longitude of the location which can be helpful. It also has unique alert id so that the data cannot be manipulated. This record of the data can be saved and can be shown to the doctor when visited.

Conclusion –

This model was basically developed to help the elderly people who are unaware of what is happening to them. This can be used as an alert if the person is going through cardiac arrest. So, when the alert is sent to the concerned person, he/she can approach this person and save their life. The web app also will keep the record of the pulse behaviour and we can monitor our health.

On the other hand, it can be used by anyone and everyone in this world. For example, we can make a kid to wear it when it goes out. So, if the kid is lost or being kidnapped, he or she can press the push button which sends alert along with the location to its parents and can be found. This will be really helpful in times of danger

Bibliography & Contributions

The idea of 'Panic Alarm' was derived from the research paper that was developed by S. D. Damayanti, M. Suryanegara, I. K. A. Enriko and M. I. Nashiruddin, "Designing A LoRa-Based Panic Button for Bali Smart Island Project," which was proposed in the year of 2019. This research paper cited the use of a remote device that can be used in case of danger to alert the known dependents in case of an event of emergency. We made changes to this idea and implemented a prototype in the form of a watch that can give location as well as directions to the location.

The intuition behind the idea for the watch was to make the device to constantly stay in contact with wrist of the user so that it can monitor the pulse and send the details to the web-app for further insights.

The link to the web-app can be found here - [Panic Alarm](#)

Contributions of Group Members

The implementation of the watch and the design of both the front and back-end was individually divided amongst the group.

Individual contributions are as follows: -

Shaiq Iqbal – 20181CSE0660 :

Implemented the Watch code and designed the multiprocessing pipeline for it.

Backend engineering on the web app - designed pipeline for Next JS to work with GraphQL and Apollo.

Sai Ram. K – 20181CSE0621 :

Developed the API Endpoint for the watch and implemented the same on the watch.

Designed frontend UI interactions.

Satyam Mourya – 20181CSE0643 :

Designed the web app layout and developed the major app components in JS ES6 and Ant Design Library

Worked as an editor for the report.

Shlaghana J.S – 20181CSE0670:

Helped in developing database queries to reduce the runtime load
Wireframing the web app.

Shivam Singh – 20181CSE0669:

Arranged the hardware.

Worked closely with Satyam in designing the UI.