Connected Panic Button with audio-based trigger

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT
FOR THE DEGREE OF

Master of Technology

IN THE
FACULTY OF ENGINEERING

BY

JAYRAJ DESAI
&
ROBIN SINGH NANDA

GUIDED BY

PROF. BHARADWAJ AMRUTUR

DEPARTMENT OF ELECTRONIC SYSTEMS ENGINEERING
INDIAN INSTITUTE OF SCIENCE, BANGALORE

JUNE 2016

Synopsis

In this report, we propose a safety-service that helps to reduce criminal offense count in public transport vehicles. A device called *PanicButton device** mounted in these vehicles is at the core of this service. The *PanicButton device* is equipped with wireless modules like Global System for Mobile communications (GSM) for ubiquitous connectivity to the back-end servers and bluetooth low energy (BLE) for communicating with the users. To make the *PanicButton device* user friendly, we restrict physical user interface to only a panic button. The *PanicButton device* runs algorithm to detect any tampering and also provide smart-phone based validation. Audio-based trigger of this *PanicButton device* provides the safety-service to disabled and immobilized victims. If scream is detected in audio received by the *PanicButton device*, emergency is asserted. Support Vector Machine (SVM), a supervised machine learning method is used to detect screams. F1 Score is used for benchmarking these SVMs, we achieved detection rate of 95% for screams heard in laboratory environment. On receiving distress signal, police is notified and regularly updated with victim's location acquired using Global Positioning System (GPS), facilitating them to reach the victim.

*We refer to our product as '*PanicButton device*', this notation is followed through out the report.

Acknowledgements

We would like to express our deepest appreciation to all those who provided us the possibility to complete this project. We give a special gratitude to our guide Prof. Bharadwaj Amrutur (ECE, IISc), whose contribution in stimulating suggestions and encouragement, helped us to complete our project.

Furthermore, we would like to acknowledge with much appreciation the crucial role of the project reviewers Prof. T.V.Prabhakar (DESE, IISc) and Prof. Joy Kuri (DESE, IISc) for their valuable comments that helped us in fine tuning our project. A special thanks to speech application group at Electrical and Communication Department, Indian Institute of Science for their valuable guidance.

Abbreviation

- PCR- Police control room
- SOC- System on chip
- GPU- Graphics processing unit
- LPDDR2- Low power double data rate 2
- RAM- Random access memory
- Wifi- Wireless fidelity
- MicroSD- Micro secure digital
- HDMI- High definition multimedia interface
- USB- Universal serial bus
- CSI- Camera serial interface
- DSI- Display derial interface
- SPI- Serial peripheral interface
- I2C- Inter-integrated circuits
- UART- Universal asynchronous receiver and transmitter
- GPIO- General purpose input and output
- A-GPS- Assisted global positioning system
- GPS- Global positioning system
- EEPROM- Electrically erasable programmable read only memory
- BLE- Bluetooth low energy
- IC- Integrated circuit
- IPS- In-plane switching
- JSON- JavaScript object notation

- NMEA- National marine electronics association
- MFCC- Mel-frequency cepstrum coefficient
- DFT- Discrete Fourier transform
- DCT- Discrete cosine transform
- SVM- Support vector machine
- ANN- Artificial neural network
- IUEC- IIITD urban environment context database
- LPC- Linear predictive coefficients
- HMM- Hidden Markov model
- TFM- Time-frequency matrix
- MP-TFD- Matching-pursuit time-frequency distribution
- NCRBI- National crime record bureau of India
- TEA- Tiny encryption algorithm
- M-TEA- Modified-tiny encryption algorithm
- HTTP- Hypertext transfer protocol
- MQTT- Message queue telemetry transport
- CoAP- Constrained application protocol
- WWW- World wide web
- IoT- Internet of things
- TCP- Transmission control protocol
- UDP- User datagram protocol
- REST- Representation state transfer
- AES- Advanced encryption standard
- FOM- Figure of merit
- FFT- Discrete Fourier transform
- PCB- Printed circuit board
- SWD- Serial wire debug
- IDE- Integrated development environment

- TI-CCS- Texas Instruments-Code composer studio
- JAR- Java application archive
- WAR- Web application archive
- APK- Android application package

Contents

List of Figures

Chapter 1

Introduction

With increasing cases of atrocities on women and children during their commute, there is a need for an advanced system to quickly notify police in case of emergency. Most of the cases remain a mystery due to lack of evidence. We propose a system that help the victims to send a distress signal to police by pressing an emergency button or by activating it with their voice.

1.1 Background

The crime against women as reported by National Crime Record Bureau of India (NCRBI), has increased by 9.9% in 2014. Around 3,25,329 crimes against women were reported in India in 2014, which includes rape, kidnapping and abduction of women, dowry deaths, assault on women with intent to outrage her modesty, insult to the modesty of women, cruelty by husband or his relatives etc.

A total of 57,311 cases were reported under kidnapping and abduction of women during 2014. These cases have shown an increase of 10.5% during 2014 over the year 2013. Figure 1.1 summarizes the NCRBI report[1] on crimes against women.

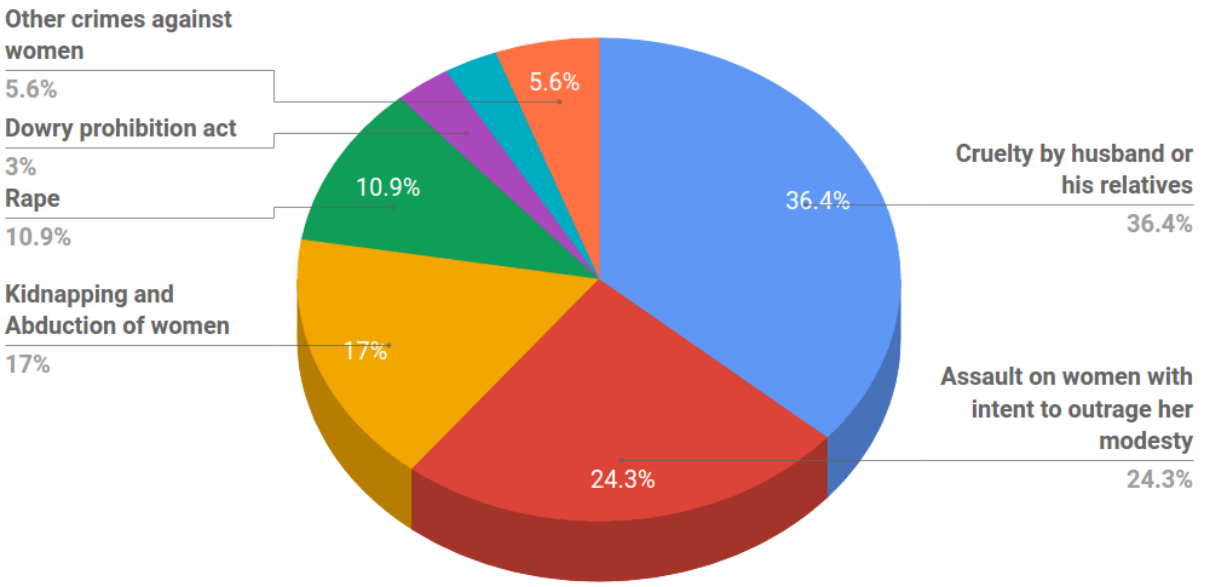


Fig. 1.1: Chart for recorded number of crime against women

Many of these happen in public transport vehicles like buses and cabs. Hence the government (Ministry of Highways and Road Transport) has recently issued a circular mandating a connected panic button which can be pressed by the victim in case of a distress situation. The button will send an SOS to the nearest Police Control Room (PCR) for prompt action. Next section describes the functional requirements of such device in detail.

1.2 Functional aspects

Our idea is to install *PanicButton device* in all public transport vehicles. It triggers an emergency, if either panic button is pressed or scream is detected. Audio-based trigger monitors the surrounding in the real time and asserts an emergency in case of scream detection. The *PanicButton device* conveys the message of emergency to the back-end server over a wireless link. Figure 1.2 shows the proceedings of a safe commute with our device installed in it.

The following secondary functions are also performed by the *PanicButton device*-

- self-check,
- indicate the status using back-light installed in emergency button,
- providing realtime vehicle location updates,

- providing panic button validation service and,
- providing driver and vehicle details to the user.

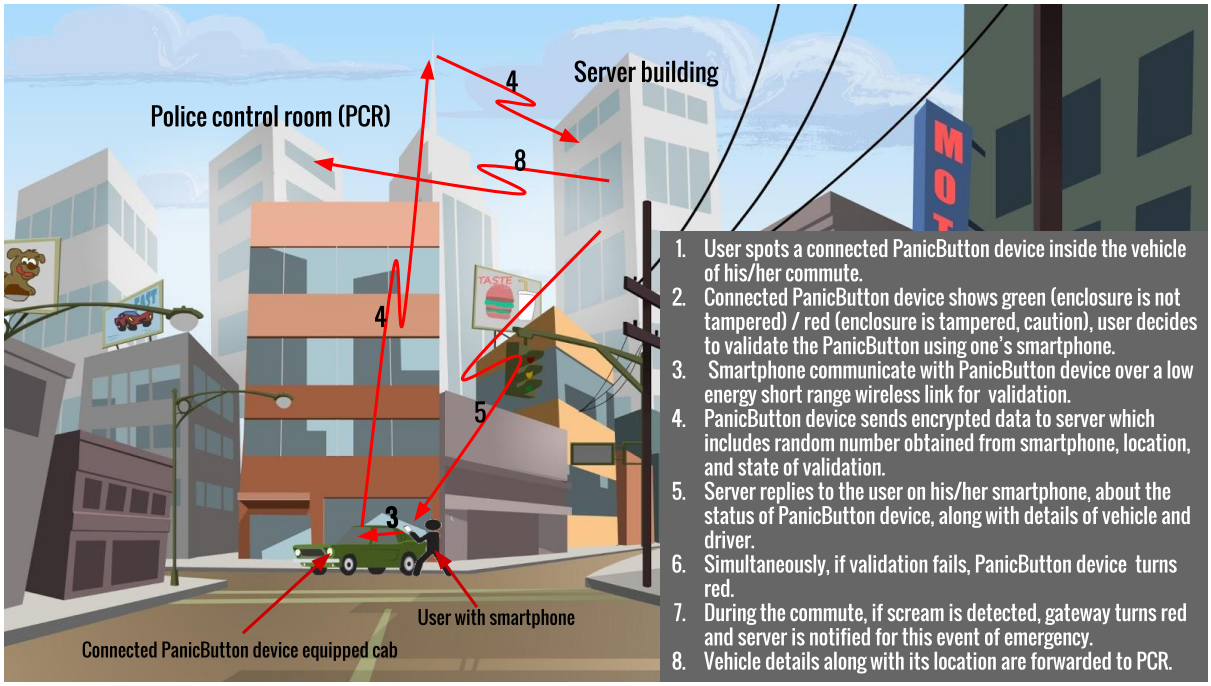


Fig. 1.2: A safe ride with our device in vehicles

1.3 User interfaces and communication channels

User can interact with the *PanicButton device* in following ways.

- using the panic button,
- using microphone available on the device and,
- using the dedicated android application.

Panic button is designed to be easily accessible; this was ensured by its large size and always-on back-light. Physically disabled and immobilized victims can use the internal microphone to interact with the *PanicButton device*. Audio-based trigger takes input from this microphone and scan the audio stream for screams. Emergency is asserted in case a scream is detected. User can interact with the *PanicButton device* for its validation using the dedicated android application.

The *PanicButton device* requires four communication channels in order to work at it's full potential.

- Wireless communication channel between the *PanicButton device* and the back-end server,
- Wireless communication channel between the *PanicButton device* and the user's smartphone,
- Internet link between the user's smartphone and the back-end server.
- Internet link between the back-end server and PCRs.

1.4 Power supply

Vehicle's battery is the primary power supply for the *PanicButton device*. The primary power supply also charges a secondary battery, which resides inside the *PanicButton device*. In case of breakdown in primary power supply, secondary battery can power the *PanicButton device* for couple of hours. In case of breakdown of both primary and secondary sources of power, an ultra low power circuitry powered by an independent source, takes the note of event and informs the back-end server on power restoration. Over all we have two power sources for complete system and an independent power source for an ultra low power system monitoring circuit.

1.5 Product/market Survey

Most of the safety device market is clustered towards the private safety services. Our *PanicButton device* provides public safety service for people, who uses public transport vehicle. Devices serving this purpose are yet to make impact in the safety-service market.

Wearable safety devices occupies major chunk of safety-service market. These devices uses smartphone as relay between the user and their guardians. These devices come in the form of a wrist band, a pendant, a necklace and an earrings. They all have small form factor and a constrained power supply. Generally these devices rely on paired smartphone in its vicinity for long-range communication. Wearables device communicates with the smartphone over low energy short range links mainly Bluetooth low enregy (BLE). Some wearable safety devices comes with a microphone to record audio as evidences. Some of the popular brands in this

category are Artemis, Amulyte, Stiletto, Cuff, Safer, Revolar and First sign. Table I summarizes their characteristics.

Brand	Trigger	Audio support	Standalone
Artemis(pendent,clips)[2]	3 Button presses	Microphone	No
Amulyte(pendent)[3]	Button press	Microphone & speaker	No
Stiletto(pendent)[4]	Button press	Microphone & speaker	No
CUFF (pendent)[5]	Button press	No	No
SAFER (pendent)[6]	2 Button presses	No	No
Revolar (clip)[7]	Button press	No	No
Fisrt sign (watch)[8]	2 Button presses	No	No

Table I: Characteristics of a few well known safety wearables in the safety-service market

The *PanicButton device* is a complete standalone package with all the necessary hardware and software packed in a 6”x6”x2” box.

1.6 Wish scope

Wish scope is to capture each and every distress moments in the vehicle and trigger an emergency alarm. Make the *PanicButton device* tamper-proof and have ubiquitous connectivity with all required devices. In the next chapter we take a look at available technology/algorithms and give a realistic scope.

Chapter 2

Study

This chapter gives an overview of the literature survey and investigation done for this project. We also discuss the reasoning behind some of the important design decisions.

2.1 Functional concept

The *PanicButton device* has to perform tasks like take user's input from the panic button, acquire audio data needed for audio-based trigger, communicate with the back-end server and user's smartphone, detect any tampering and run audio processing algorithms. Figure 2.1 and figure 2.2 shows the top level functional diagram of the *PanicButton device* when installed in cabs/autos and bus respectively.

PanicButton device is equipped with a panic button to take user inputs, bluetooth low energy (BLE) for communicating with the user's smartphone, global system for mobile communication (GSM) to communicate with back-end server, global positioning system (GPS) to locate the vehicle in case of emergency, microphone to record audio and speaker to play audio.

Users can validate *PanicButton device* using their smartphone through a dedicated android application. Communication between the user's smartphone and the *PanicButton device* is handled by the respective BLE module. After performing validation, user is notified with the status of microphone, emergency button, firmware and enclosure of the system.

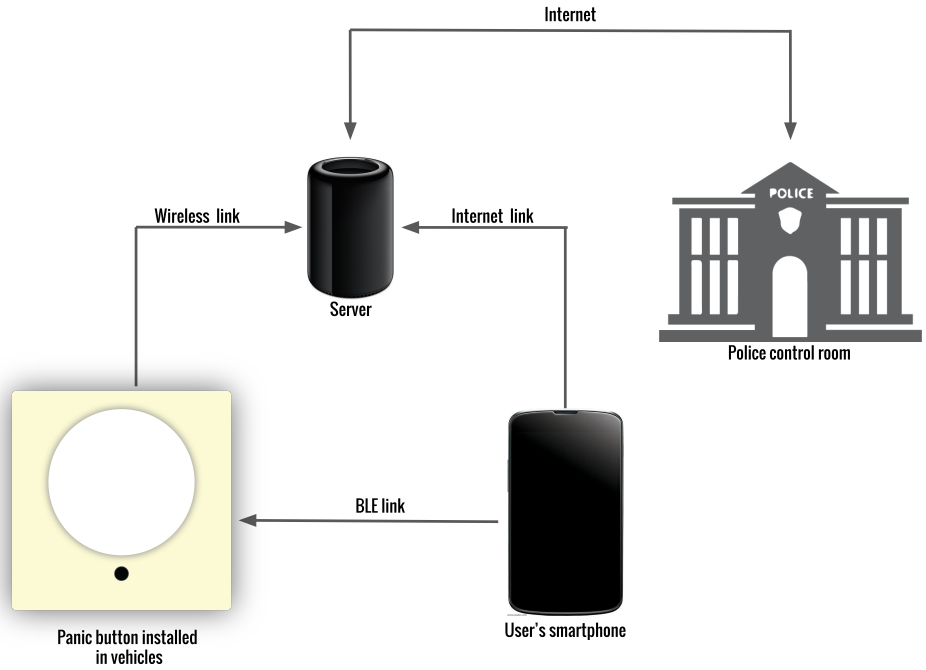


Fig. 2.1: Functional description of the *PanicButton device* with audio-based trigger in auto/cab

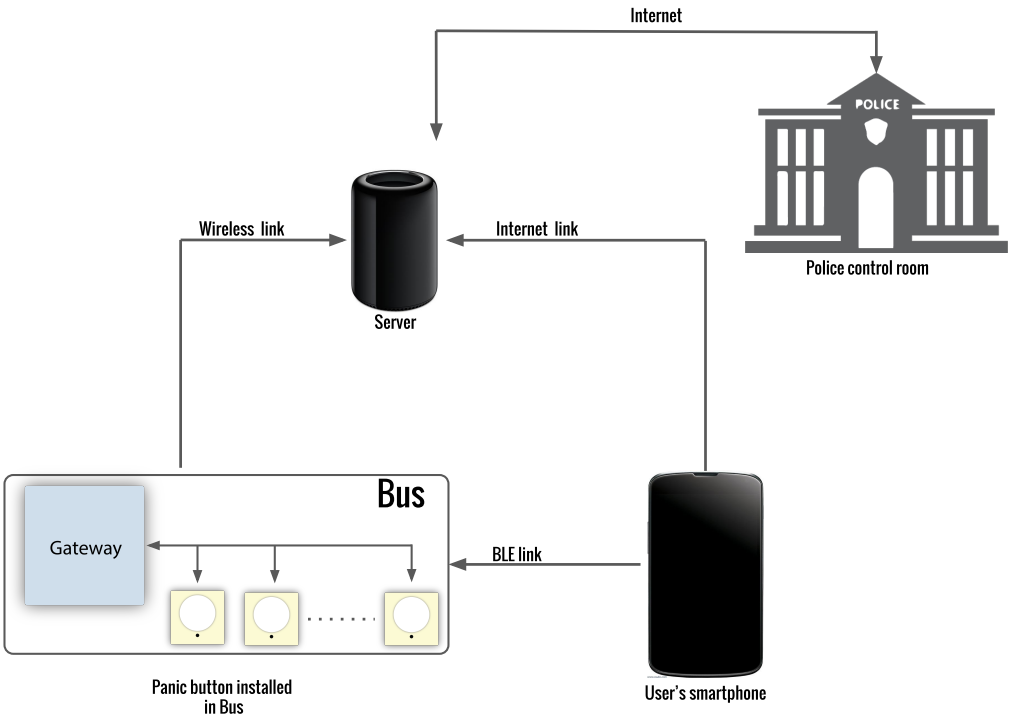


Fig. 2.2: Functional description of the *PanicButton device* with audio-based trigger in bus

Emergency is asserted, if either the panic button is pressed or scream is detected by audio-based trigger. In case of emergency, compressed stream of audio along with vehicle's location is sent to the server, for later examination.

2.2 Module study

This section dwells in literature survey and design decision done for the individual modules.

2.2.1 Protocols for communication

We studied the protocols that can be used between the links present in the system. There are four different links in the system.

- link between the *PanicButton device* and the back-end server,
- link between the smartphone and the back-end server,
- link between the smartphone and the *PanicButton device*,
- link between the back-end server and the PCRs.

The following subsections will go into the details of each of them.

2.2.1.1 Protocol for messaging between the *PanicButton device*, the back-end server and PCRs

The message link between panic button and back-end server must be a low latency link. In order to achieve that we looked at different protocols with their merits and demerits. The following protocols can be used for this link.

- Hypertext Transfer Protocol 1.1 (HTTP)
- Message Queue Telemetry Transport (MQTT)
- Constrained Application Protocol (CoAP)

HTTP is one of most widely used protocol for data transfer. It is the foundation of data communication for the World Wide Web (WWW). HTTP is a text based application layer protocol, which uses Representation State Transfer (REST) with four methods *GET*, *PUT*, *POST* and *DELETE*.
MQTT is one of the emerging protocol in the Internet of Things(IoT) domain. MQTT uses Transmission Control Protocol(TCP) as transport layer. It uses publish subscribe pattern with only two methods *SUBSCRIBE* and *PUBLISH*. MQTT is a protocol with very small header overhead.
CoAP is another IoT protocol designed specifically to be used in the lightweight environments. CoAP is designed to easily translate to HTTP for simplified intergration with the web. CoAP runs over User Datagram Protocol (UDP) transport layer.
Table I compares different protocols.

Protocol	**REST/PubSub**	**Transport layer protocol**	**header**	**Min-header (bytes)**
HTTP	REST	TCP	Text	18
MQTT	PubSub	TCP	Binary	2
CoAP	REST	UDP	Text	4

Table I: Camparision of HTTP, MQTT and CoAP

The architecture of the our system can benefit from publish subscribe model for transparent transfer of messages between the *PanicButton device* and PCR, where back-end server will act as a message broker. The back-end server can relay messages transparently to different PCRs at different locations. MQTT uses binary header with smallest packet size of just 2 bytes, which helps in reducing the bandwidth requirement for the communication.

2.2.1.2 Protocol for audio streaming

The audio stream link between the *PanicButton device* and back-end server requires high throughput, low latency, guaranteed delivery link. The protocols under consideration are

- HTTP1.1 over TCP
- Raw over TCP
- Raw over UDP

We decided to use raw over TCP because it fulfills the requirements for this link. HTTP 1.1 also provides the same benefits as raw over TCP, but adds large header overhead which is not desired. The extra information regarding the stream can be embedded in the compressed audio stream explained in the next section.

2.2.2 Compression algorithm

The use of wireless link from the *PanicButton device* to the back-end server restricts the bandwidth which is available for audio transmission. The compression of audio stream is essential for reliable transfer of audio from panic button to back-end server under congestion.

Format	Lossless	Compression factor	Open-source/royalty-free
Raw	Yes	1:1	Yes
Wav	Yes	1:1	Yes
Mp3	No	5:1	No
Vorbis Ogg	No	10:1	Yes

Table II: Expermental comparision of different audio formats at 44100 sample rate and 16-bit per sample

Different audio formats are experimentally compared in table II. Vorbis is open-source, royalty free audio compression and streaming format. Vorbis ogg's performance is better compared to mp3 in terms of compression ratio. We decided to use Vorbis Ogg audio format for streaming audio from the *PanicButton device* to the back-end server.

2.2.3 Encryption algorithm

Encryption is used to avoid man-in-the-middle attacks between the *PanicButton device* and the back-end server. Two of the most popular encryption algorithms are Advanced Encryption Standard (AES) and RSA algorithm. RSA algorithm uses asymmetric key configuration, while AES algorithm uses symmetric key configuration. The advantage of using RSA algorithm over AES is that, even if public key(key stored in panic button) is compromised, the communication is not compromised. We decided to use RSA over AES due to this advantage.

2.2.4 Audio-based trigger module

This audio-based trigger comes under *sound event detection* domain of relatively broad research area of audio/sound analysis. Sound event detection is relatively new area compared to speech analysis, so most of the literature in the field of acoustic analysis is highly clustered around speech analysis. In section 2.2.4.1, we take a look at literature survey and experiments done to come up with required feature for scream detection; in section 2.2.4.2, we take a look at literature survey done for algorithm to be used for scream detection.

2.2.4.1 Feature for scream detection

Features extracted from acoustic signals are vectors that can faithfully represent them. In order to develop the best possible classification algorithm for reliable detection and classification of sound events, it is very important to select the feature set carefully. Although a large feature set has its benefits of generating accurate results, using a small feature set can reduce latency and make the design simpler.

Over the last few years, several audio feature extraction techniques have been introduced. They make use of one of the following two signal representation domains: temporal and spectral. Temporal domain features such as signal energy, pitch, zero-crossing [9] rate and entropy modulation [10] have been used for speech classification but are not enough to represent the non-stationary characteristics of sound events. Spectral features such as percentage of low energy frames, 4-Hz modulation energy, spectral roll-off point, mean frequency, spectral centroid, mel-frequency cepstral coefficients and frequency slopes are useful in audio classification, but they do not provide any information about the temporal evolution of the extracted features over the frame.

Initially researchers used popular mel-frequency cepstral coefficients (MFCC) as features for sound event detection. MFCC are short-term spectral based features. These features are extracted through a series of digital signal processing steps like windowing, Discrete Fourier Transform(DFT), logarithm and Discrete Cosine Transform (DCT). MFCC are still used for non-verbal sound recognition [11]. A paper[12] proposed a sound event classifier which use MFCC feature set; they achieved 74% accuracy. Another paper [13] successfully classified acoustic environmental sounds like office, soccer match, beach, laundrette, street noise, rail station, car, bar and bus using a Hidden Markov Model(HMM) based classifier which used MFCC as feature vectors.

Linear predictive coefficients (LPC) is another well known feature set used for sound classification of car noise, factory noise, street noise, babble and bus noise. A Quadratic Gaussian classifier using LPC feature set gives accuracy of 90% for car/factory noise and 60-80% for the rest.

Time-Frequency Matrix(TFM) feature set tends to capture non-stationary and discontinuous properties of sound events. This matrix is obtained using matching-pursuit time-frequency distribution (MP-TFD) technique, followed by non-negative matrix decomposition to decompose the TFM into its significant components [14].

After looking at individual feature sets, researchers started fusion of various feature sets. [11] used MFCC along with Pitch range based features, [15] used MFCC along with spectral features like centroid, flux, flatness, roll-off, harmonic-to-noise ratio and pitch. TFM when combined with a few spectral and MFCC features, gives 10% accuracy-rate improvement compared to only MFCC features based classifiers [14].

One paper [16] went in completely different direction to tackle the issue of sound event classification, they argued that sound events produced a unique texture, which can be visualized using a spectrogram image and could be analyzed for automatic sound event detection. Another paper [17] used pseudo-coloration to enhance the perception. Spectrogram is first normalized into grey-scale with a fixed range, then dynamic range is quantized in to regions, each of which is then mapped to form a monochrome image. Finally monochrome images are partitioned in to blocks and distribution statistics in each block are extracted to form the feature set. Robustness of spectrogram based methods comes from the fact that noise is normally more diffused than the signal and therefore the effect of noise is limited to a particular quantization region, leaving other regions less effected.

Feature used in detection/classification systems impacts their response time and power consumption. A non-realtime application can make use of large and time consuming feature sets to get highly accurate results, while a realtime application can only use feature set that fit into its time budget. Power consumption is important aspect in battery powered devices; Feature set that can give acceptable accuracy at lesser computation results in system that lasts longer on a single charge.

We decided to use MFCC along with spectral features like centroid, flux, flatness, roll-off, harmonics-to-noise ratio and pitch due to its promising performance [15]. We conducted an experiment to find out the execution time of this feature set. The task of detecting scream is to be performed in real-time. In our project we decided to analyze live audio stream every 1 second, which gives us 2 second to analyze and give results for audio received in previous

second, this decision was based on the analysis done by [15], which showed average scream duration to be around 2 second. We tried MFCC plus few spectral features as feature vector and computed feature extraction time for an audio clip of 2 second, Table I summarizes scream detection time on two hardwares.

Platform	Feature extraction time (in s)
Raspberry pi 3	13
Intel quad core i5-4440 @ 3.10GHz (16 GB)	1.5 to 2

Table III: Feature extraction time using MFCC and spectral feature as feature vector

With MFCC and spectral features as feature set, we had latency of 13 sec which is beyond the 2 sec limit. We had to shed computationally expensive part of this composite dataset to squeeze into the budget of 2 sec, we decided to use only MFCC as feature vector and Table IV summarizes the execution time of this dataset.

Platform	Feature extraction time (in s)
Raspberry pi 3	1.480
Intel quad core i5-4440 @ 3.10GHz (16 GB)	1.020 to 1.050

Table IV: Feature extraction time using MFCC as feature vector

With MFCC as feature vector we have latency of 1.480 sec, which is less than our time limit of 2 sec. We decided to go with MFCC as the feature vector for scream detection.

2.2.4.2 Classification algorithm for scream detection

These algorithms take features as input and output probabilities of all classification class for the given input. The class with highest probability for a given input is said to be the class of the input.

Initially researchers used classifiers like a quadratic Gaussian classifier, a least-square linear classifier, a neighbor classifier and a decision tree classifier [11]. These classifiers could preform well for particular sound events but couldn't generalize well to non-verbal sound events.

Lately researches have started using machine learning for the purpose of sound event classification. Algorithms like Artificial Neural Network (ANN) [11] and Support Vector Ma-

chine(SVM) [15] are being used widely. These algorithms fall under the category of supervised machine learning. These algorithms outperform nearly all other classifiers used in past due to their capability to model complex systems.

ANNs are mathematical models that emulates biology of a human brain. ANNs are parallel computing mechanisms that contains neurons laid out as a layers, interconnects and learning rules. SVMs are another set of mathematical models used for classification. SVMs constructs a hyperplane or set of hyperplanes in a high dimensional space, which is used for classification. ANN algorithms require more data to train the classifier as compared to SVMs. In case of SVMs, we need to set a limited number of parameters and choose among possible kernel. We decided to use SVM algorithm due to available limited dataset and easy convergence in training.

2.3 Industrial design

Being a product for public safety service, it should blend with the vehicular environment yet remain accessible. The panic button mounted on the *PanicButton device* has to be easily accessible 24x7. A button with area equal to the average area of adult hand and back-light will fulfill these requirements. Front panel of the *PanicButton device* mounts three essential part namely panic button with status back-light, speaker and microphone. Front panel also acts as lid for the *PanicButton device*. Secondary battery and electronics used in this project resides at the base of enclosure. Figure 2.3 shows a pictorial representation of the assembly.

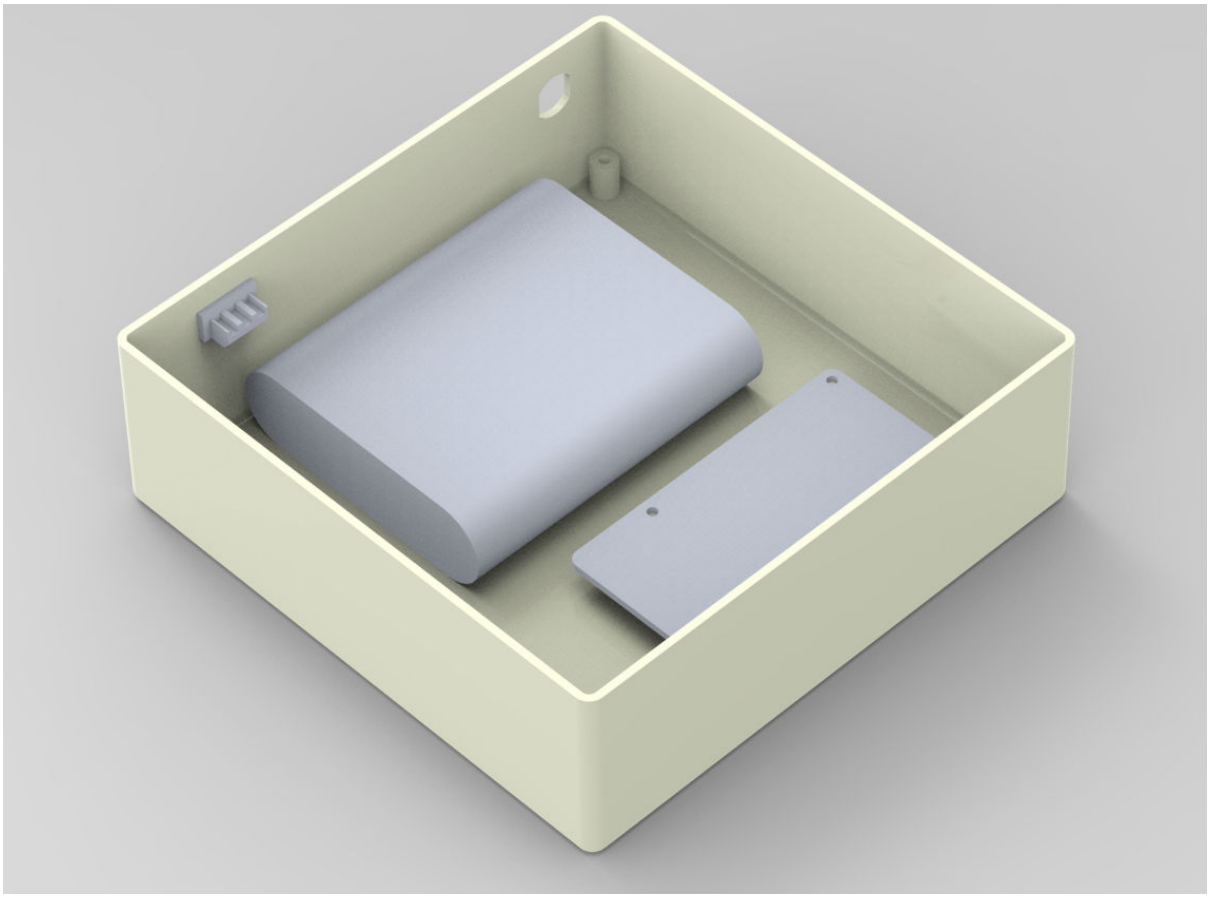


Fig. 2.3: Layout of battery and electronics at the base of the *PanicButton device*

Details of this enclosure will be covered in Industrial design section of chapter 3.

Chapter 3

Design

In this chapter, we dwell into the detailed design of the project. Section 3.1 lists the individual modules designed in this project. Section 3.2 gets into details of all the hardware design; section 3.3 does the same for software design. Finally section 3.4.2 dwells into details of industrial design of the project.

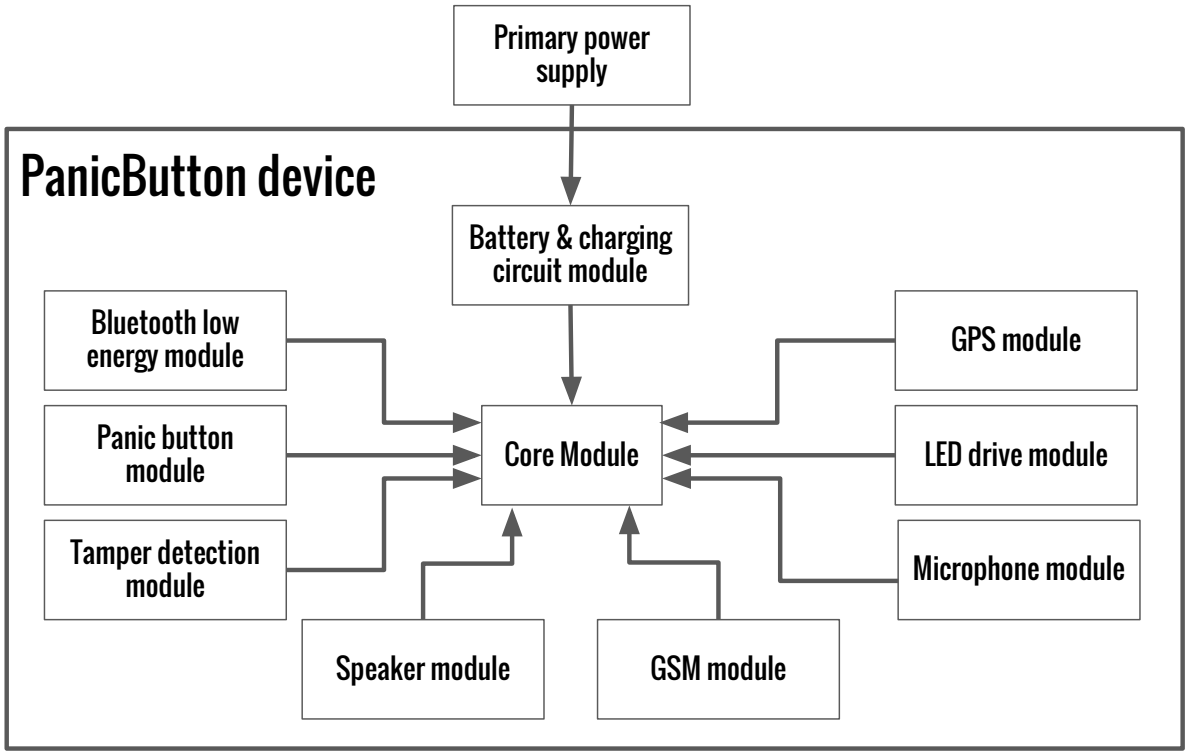


Fig. 3.1: Block diagram of the *PanicButton device*

3.1 Module partitioning

This section gives an overview for the content in next two sections. Section 3.2 describes the design of hardware used in the project. It is divided in three major sections-

- *PanicButton device*- describes the design of core module, tamper detection module, etc.
- Server- describes the hardware involved in server and,
- Smartphone- describes the hardware involved in smartphone.

Section 3.3, describes the design of the firmware used in the project. This section dwell in to the software design aspects of-

- System validation module,
- Audio streaming module,
- Global Positioning System module,
- Panic button module and,
- Audio-based trigger module.

3.2 Hardware design

3.2.1 *PanicButton device*

3.2.1.1 Core-module

The core-module for the system is based on Raspberry pi 3. Raspberry pi uses BCM2837 SOC from Broadcom, with four ARM Cortex-A53 64-bit cores running at 1.2GHz, a Broadcom VideoCore IV Graphics Processing Unit (GPU), 1 GB LPDDR2 Random access Memory (RAM) which runs at 900MHz, 10/100 Ethernet, 2.4GHz 802.11n Wireless Fidelity (WiFi) and micro secure digital (microSD) card with maximum size of 128 GB. Peripherals connects to a High-Definition Multimedia Interface (HDMI) port, four Universal Serial Bus (USB) 2.0 ports, one audio-video port, Camera Serial Interface (CSI) and Display Serial Interface (DSI) available on board. Raspberry pi 3 has a 40-pin Header, which contains one Serial peripheral

interface (SPI) with two chip select pins, one Inter-integrated Circuits (I2C), one Universal Asynchronous Receiver Transmitter (UART) and 18 General Purpose Input Output (GPIO) pins. The schematic of the core-module is shown in figure 3.2.

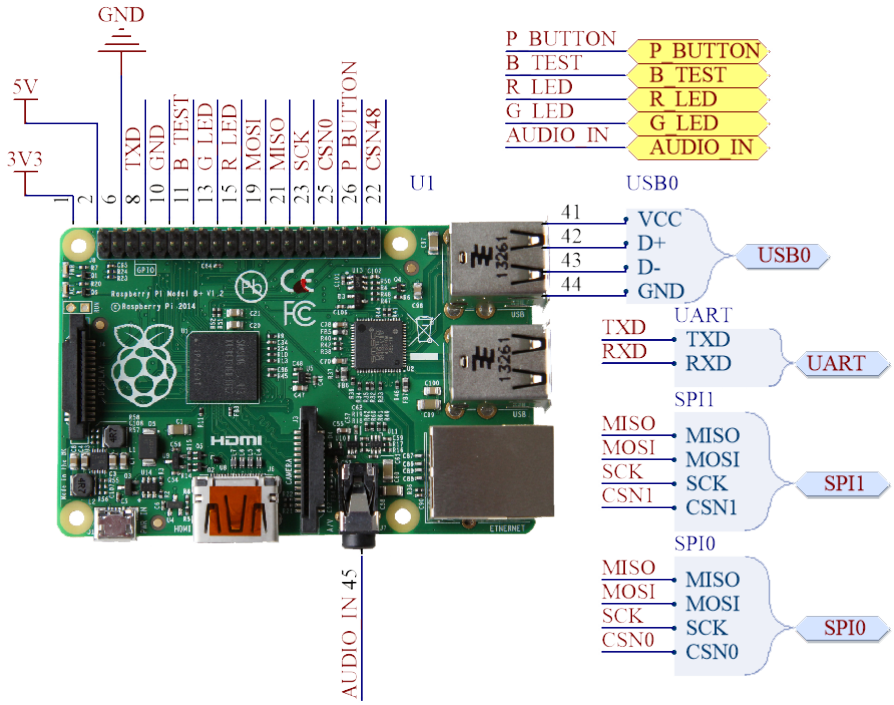


Fig. 3.2: Schematic of the core-module of the *PanicButton device*

3.2.1.2 Tamper detection module

The *PanicButton device* is installed in public transport, which makes it very vulnerable to tampering. In order to tamper with the device, they must first open the enclosure. The tamper detection module detects opening of the lit/top panel by using a switch and stores that information in non-volatile memory. The normally closed type switch is connected between the base of enclosure and the top lid; It changes its state when lid is opened. The tamper detection module uses an ultra lower power MSP430G2553 microcontroller running on CR2032 Lithium ion coin cell auxiliary battery. SPI interface is used to communicate between the microcontroller and the core-module.

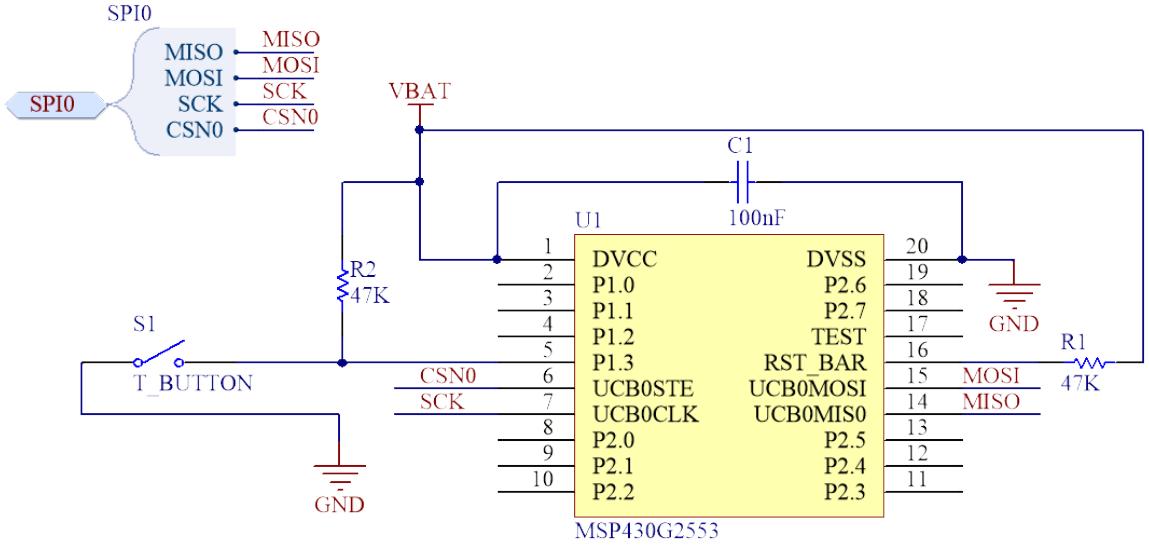
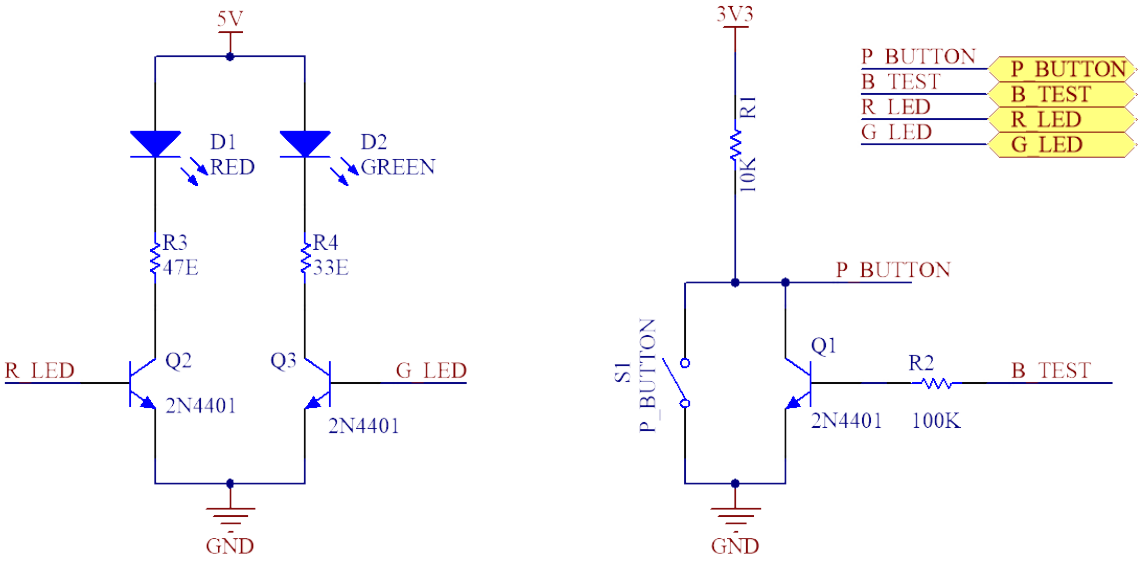


Fig. 3.3: Schematic of Tamper detection module

3.2.1.3 Panic button & LED drive module

Panic button is a normally open non-latching type switch with 10 kOhm pull resistor which pulls it up to 3.3 V. A 2N4401 transistor is used in parallel with the switch to emulate the switch press to test the functioning of the switch. 100 kOhm resistor connected to the base terminal of the transistor, limits the current flowing through the base and emitter terminals. *B_TEST* is output from core-module and *P_BUTTON* is input to the core-module. Two 2N4401 are used to drive four red and four green LEDs for status indication. *R_LED* and *G_LED* are output from the core-module to drive these LEDs. The schematic of module is shown in figure 3.4.



(a) (b)

Fig. 3.4: Schematic of (a) LED driver (b) Panic button

3.2.1.4 GPS module

The Global Positioning System (GPS) module is NEO-6M module from Ublox. NEO-6M uses AssistNow Autonomous Technology, which provides functionality similar to Assisted GPS (A-GPS) without the need of external network connection. Based on previously broadcast satellite ephemeris data downloaded to and stored by the GPS receiver. AssistNow Autonomous Technology generates accurate satellite orbital data(“AssistNow Autonomous data”) that is used for the future GPS position fixes. Electrically Erasable Programmable Read-Only Memory (EEPROM) is used to store parametric data of GPS receiver (NEO-6M). UART protocol is used to connect GPS module to the core-module as shown in figure 3.5.

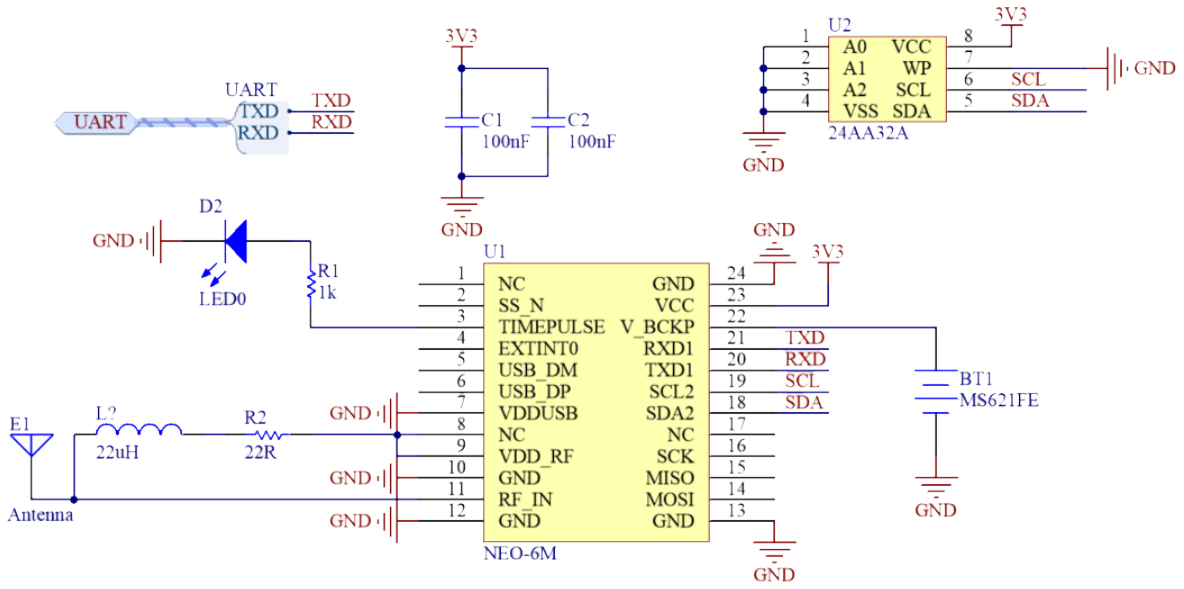


Fig. 3.5: Schematic of GPS module

3.2.1.5 Bluetooth low energy module

The Bluetooth Low Energy (BLE) module uses nRF51822QFAC SOC from Nordic Semiconductors. nRF51822 SOC has ARM® Cortex™-M0 32-bit processor running at 16 MHz with 256 kB of embedded flash program memory and 32 kB of RAM. It requires one crystal for system clock and one 32.768 kHz crystal for real time clock functionality. It uses 50 Ω impedance matching circuit for maximum radiation as shown in right hand side of figure 3.6.
4-wire SPI interface is used to connect the SOC to the core-module.

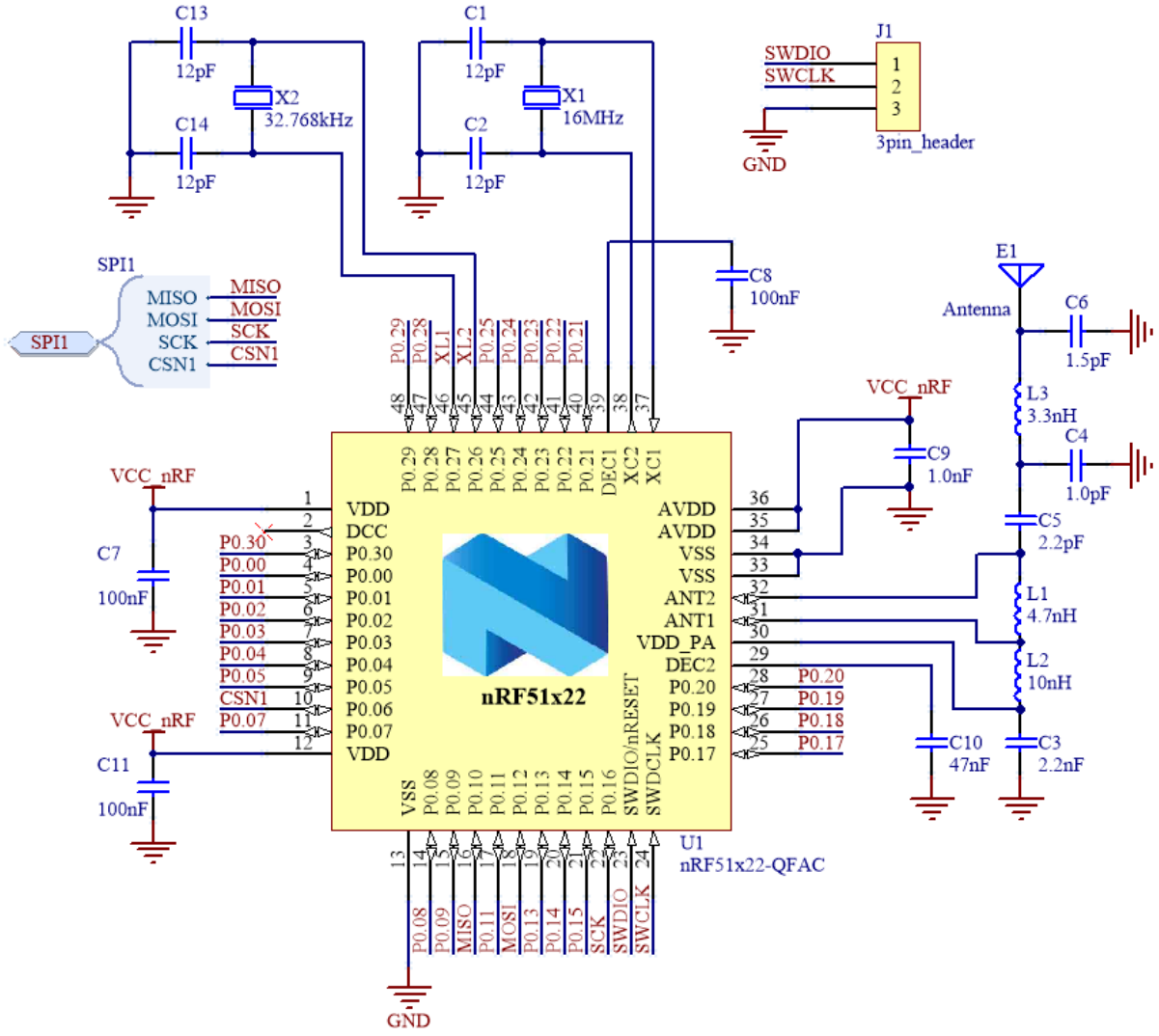


Fig. 3.6: Schematic of Bluetooth low energy module

3.2.1.6 Speaker module

A 8 Ω 0.5 watt speaker and microphone pair is used to validate the microphone in a closed loop. As core-module cannot drive the 8 Ω speaker directly, we designed LM386 Integrated Circuit (IC) based amplifier to drive the speaker. A 10 kΩ potentiometer is used in series with 10 uF bypass capacitor to adjust the gain manually as shown in figure 3.7.

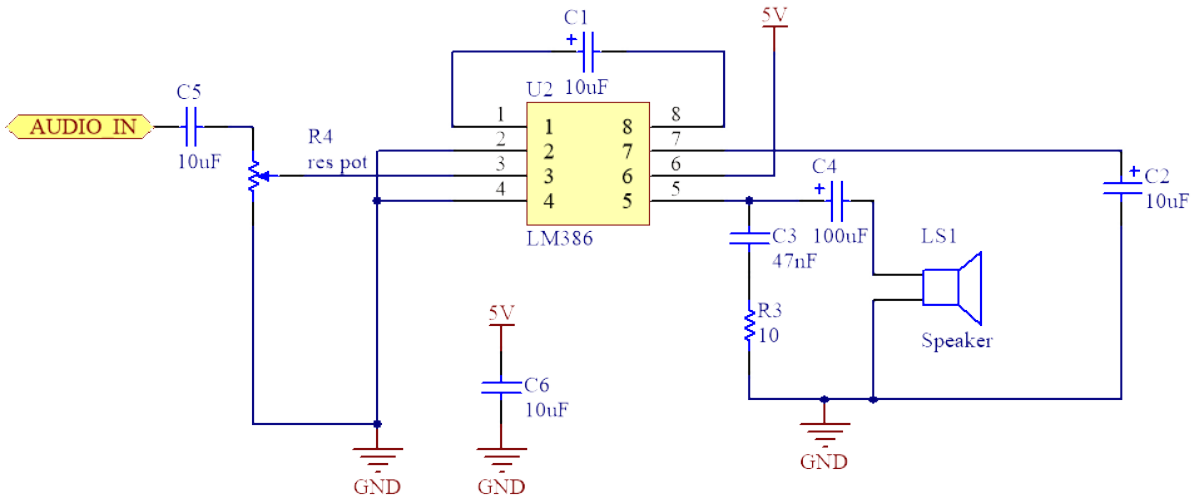


Fig. 3.7: Schematic of speaker amplifier module schematic

3.2.1.7 Microphone module

Core-module does not have an on-board audio input support. We used external USB sound-card to add audio input(microphone) to the core-module as shown in the figure 3.8.

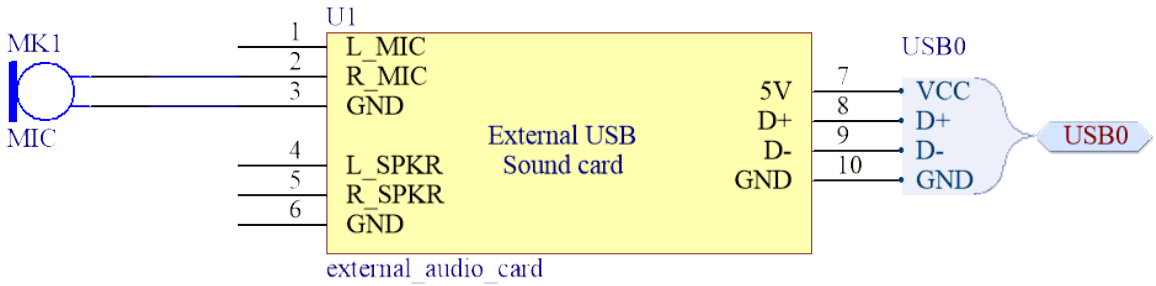


Fig. 3.8: Schematic for microphone module

3.2.2 Back-end server

We used a desktop computer as the back-end sever with specification given below.

- **Processor:** Intel® Core™ i5-4440 quadcore processor running at 3.1 GHz.
- **Graphics processor:** Integrated Intel® HD Graphics 4600 running at 350 MHz.

- **RAM:** Kingston value RAM 16 GB DDR3 running at 1600 MHz.
- **Hard disk:** Seagate SATA SSHD 1 TB desktop internal hard drive 7200RPM.

3.2.3 Smartphone

We used a Google Nexus 4 smartphone running Android 5.1.1 to test designed application. The specification of the smartphone are given below.

- **SOC:** Qualcomm APQ8064 Snapdragon S4 Pro with four Krait running at 1.5 GHz and Adreno 320 GPU running at 400 MHz.
- **Radio:** Wi-Fi 802.11 a/b/g/n and Bluetooth v4.0.
- **Battery:** 2100 mAh.
- **Screen:** 4.7 inches 768 x 1280 pixels In-Plane Switching (IPS) display.

Note: Any smartphone with Bluetooth v4.0 or above and running Android 4.3 or above can be used.

3.3 Software design

3.3.1 System validation module

The system validation module is responsible for self checking and notifying the user about the status of the system. The system validation module tests the microphone, panic button, firmware and enclosure tamper state.
The microphone and panic button validation runs every time a new user boards the vehicle or a user wants to re-validate the system. The microphone and the panic button has a five second timeout period after which the validation state is invalidated. The firmware validation is done after every reset and it doesn't have a timeout period. The enclosure tamper detection is an always-on validation sequence, which syncs its state every minute with tamper detection hardware.
The flow chart of system validation procedure is shown in figure 3.9.
The components involved in the validation of the system are as follows.

- User's smartphone
- nRF51822 BLE SOC
- Core-module
- Server

User's smartphone connects to the nRF51822 and sends a random *connectionID* to it and closes the connection. User's smartphone waits for the response from the server. The nRF51822 relays the *connectionID* to the core-module. Core-module checks whether the previous validation results are still valid. If it is not valid, then core-module will start the validation sequence to validate the state of the system. If the previous validation results are valid or the current validation sequence completes, then core-module sends the validation results, *connectionID*, *date-time* and encrypted *panicButtonID* to the server. If the server finds the *panicButtonID* in the database, then it sends the driver details, vehicle details and validation results to the user's smartphone. The user's smartphone will show a notification to interrupt the user, which contains the information received from the server.
The flowchart of the system validation module is shown in figure 3.9. Next we will look into the working of individual components.

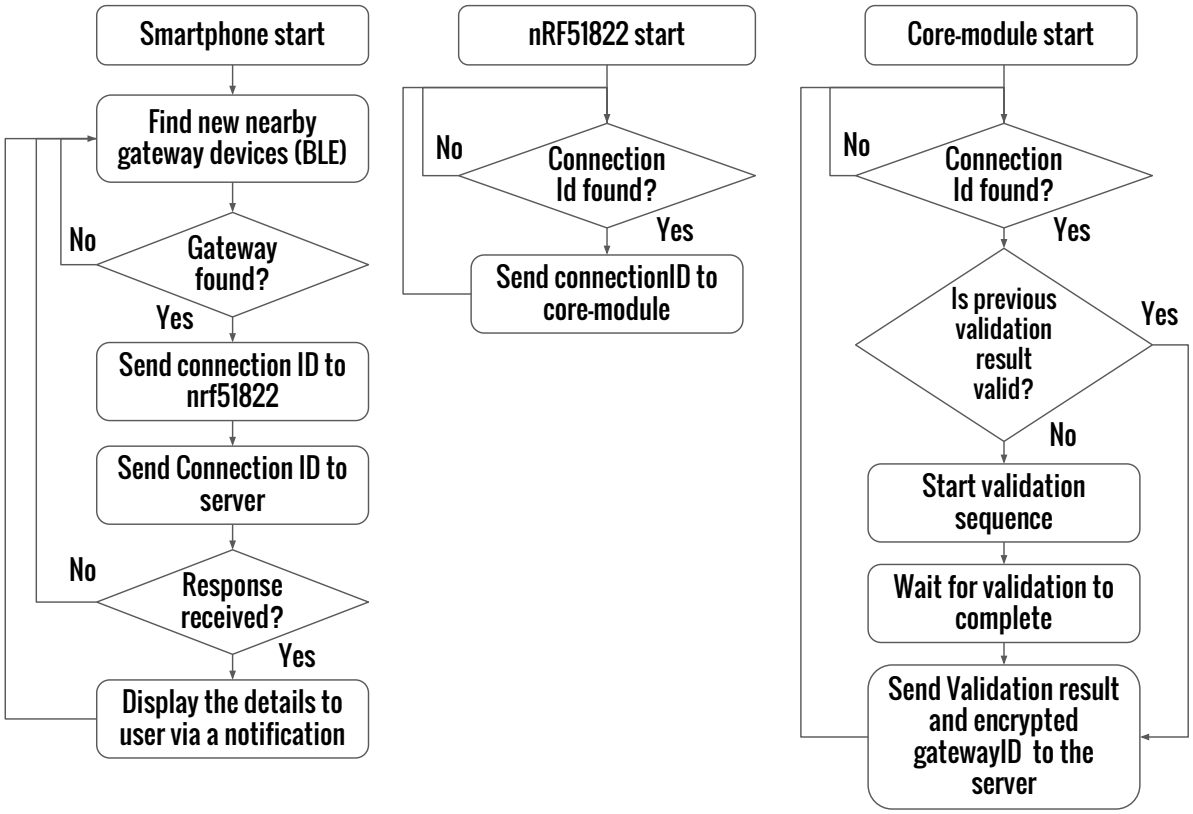


Fig. 3.9: Flow chart for BLE system validation module

nRF51822 acts as a relay between core-module and user's smartphone. nRF58122 uses Generic Access Profile (GAP) to control the connections and advertising. BLE's GAP layer is responsible to broadcast connectible beacons to show *PanicButton device*'s presence to the nearby devices for them to connect.

nRF51822 is running a Generic Attribute Profile (GATT) server to serve the validation service with *connID* characteristic. The UUID used are shown in table I.

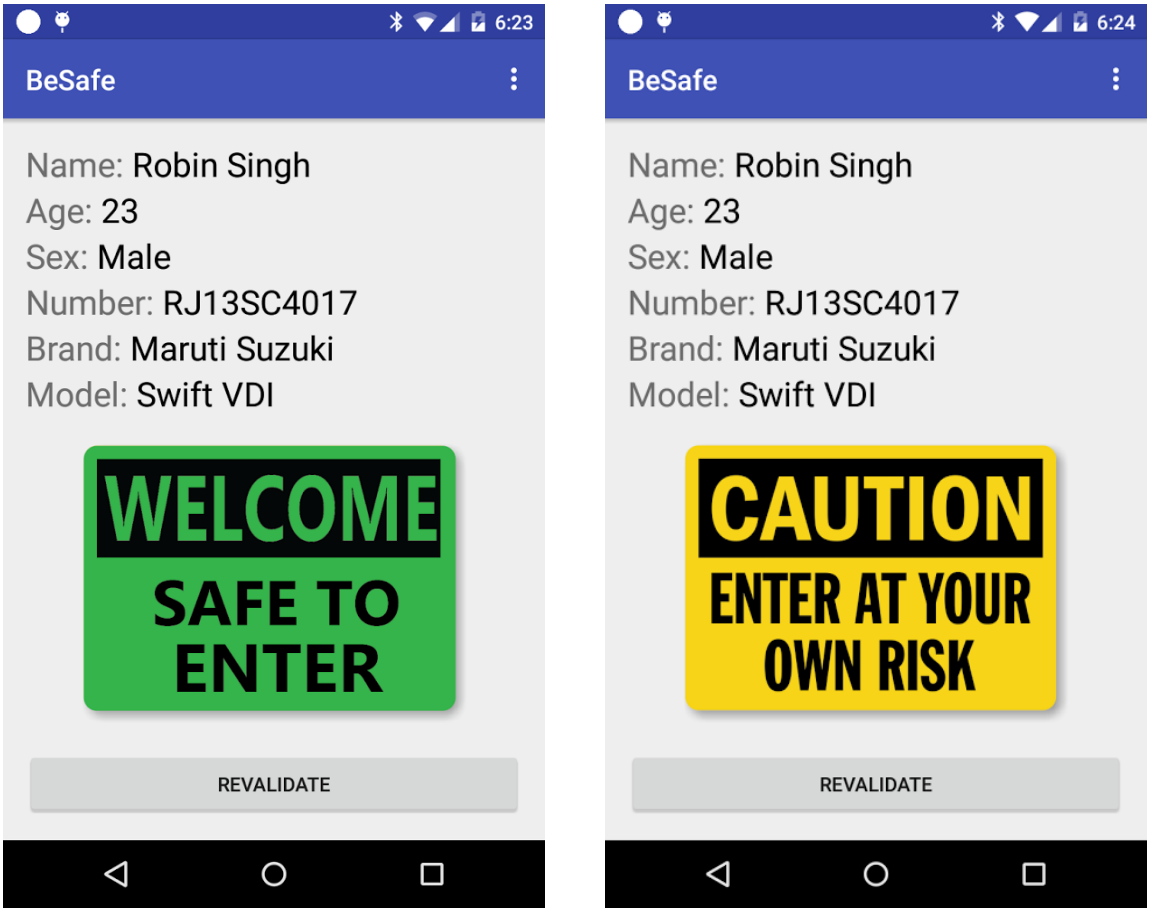
	UUID
Validation service	6E400001-B5A3-F393-E0A9-E50E24DCCA9E
ConnID characteristic	6E400002-B5A3-F393-E0A9-E50E2DCCA9E

Table I: UUIDs of the BLE services

nRF51822 forwards the *connectionID* received from user's smartphone to the core-module using SPI protocol.

Android app runs on user's smartphone as a background service to scan for nearby *PanicButton devices*. App differentiates the *PanicButton device* from other devices using services they offer i.e., if some device has the service and characteristics IDs as mentioned in table I, it considers the device as a valid *PanicButton device*.

As soon as a valid *PanicButton device* is found, the app connects to the *PanicButton device* and sends a pseudo random 32-bit number to it and disconnects. After disconnection from the *PanicButton device*, app sends the same number to the server and waits for server's response. Once the server's response is received, app builds a notification with information received from server and display it to the user. In case the validation of the *PanicButton device* is successful, it will display "Welcome, safe to enter", otherwise it will display "Caution, enter at your own risk". The screeshots of the android app is shown in figure 3.10.



(a) Screenshot of "Welcome, safe to enter" (b) Screenshot of "Caution, enter at your own risk"

Fig. 3.10: Screenshots of Android app

Core-module receives *connectionID*, when a user's smartphone connects to nRF51822. The core-module will add the *connectionID* to the queue of *connectionIDs*. Each *PanicButton de-*

vice has it own unique and secret 128-bit *panicButtonID* stored in the core-module associated with it, embedded in the non-volatile memory. The *panicButtonID* is used to validate the identity of the *PanicButton device* to the server. After receving the *connectionID* for the user's smartphone, the core-module will start a new validation sequence if the previous validation result is not valid anymore due to timeout event. Once a valid validation result is ready, the core-module will merge *connectionID* and *panicButtonID* into *mergedID*. The data flow for merging the IDs is shown in figure 3.11.

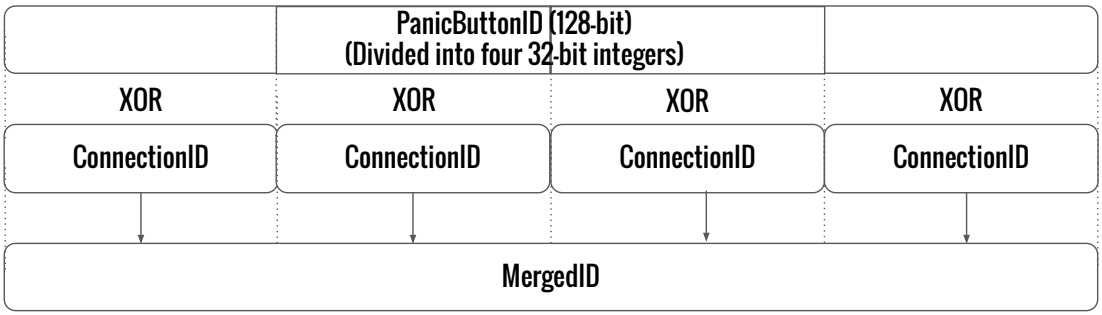


Fig. 3.11: Data flow for merging the connectionId and panicButtonID

MergedID is encrypted to get *cipherCode* by RSA algorithm using 256-bit public encryption key, which is stored in non-volatile memory of core-module.
The core-module will send Message Queuing Telemetry Transport (MQTT) message to *"/gateway/validation"* topic in JavaScript Object Notation (JSON) format containing *connectionID*, *cipherCode*, *date-time* and *validationResult* fields.

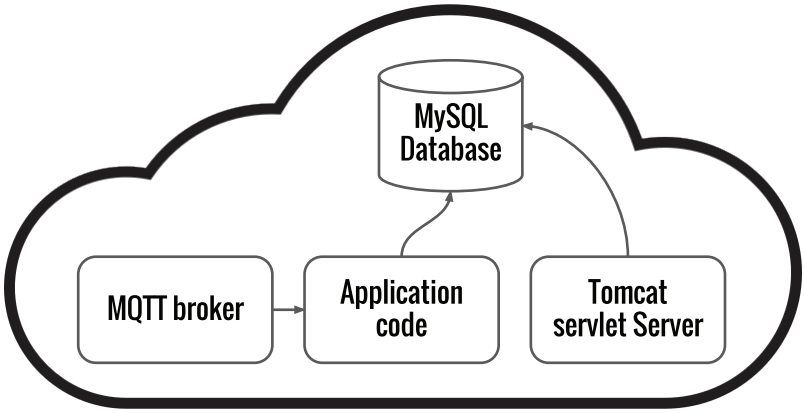


Fig. 3.12: Overview of server running in the cloud

Server is running a MQTT broker, a Tomcat servlet server, a MySQL database and application

code to handle incoming MQTT messages as shown in figure 3.12.
Server is connected to the MySQL database, which has information of all the gateways *PanicButton devices* and the vehicles. When a user's smartphone connects to server, it sends a unique random 32-bit *connectionID*. When the *PanicButton device* sends the validation results, it also sends the *connectionID* to the server. The *connectionID* is used to match the two connections (MQTT from *PanicButton device* side and HTTP from smartphone side). Server waits for both connection to be established to synchronize the processing of the validation request.

The server holds the connection from user's smartphone until, it receives the same *connectionId* in MQTT message in topic *"/gateway/validation"* from *PanicButton device*. Once the synchronization completes, the server decrypts the *panicButtonID* using the secure private key stored in it. If the *panicButtonID* exists in the database, the server sends the details to the smartphone or else it sends a NULL object to the smartphone. Server also sends validation results and date-time to the user's smartphone in JSON format. After sending the data, server closes the connection.

Next we will look into individual modules that gets validated.

3.3.1.1 Microphone validation

Microphone is one the most important module in the system. To Validate the microphone functionality, we are using a closed loop validation procedure as shown in figure 3.13.

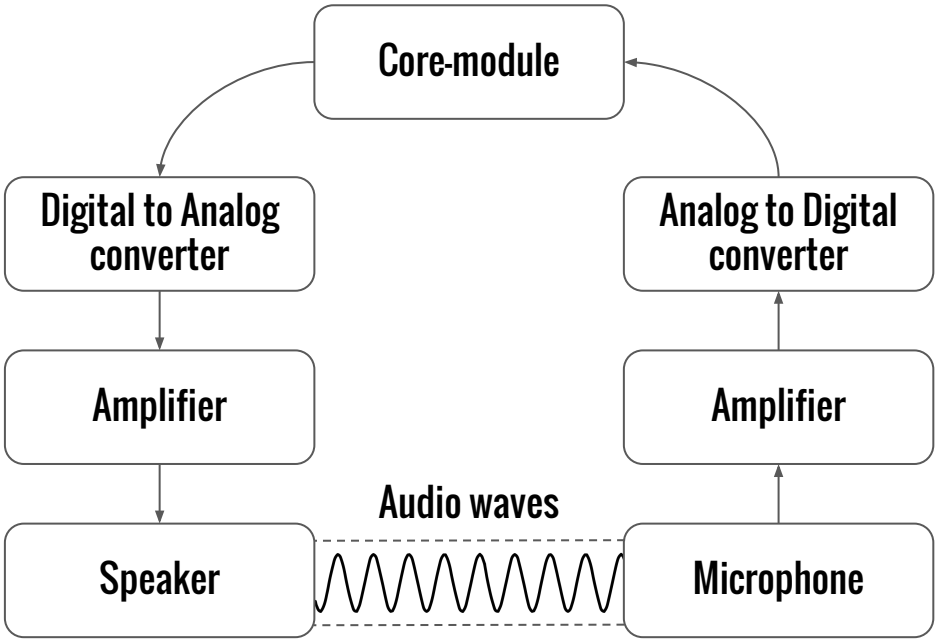


Fig. 3.13: Microphone validation procedure

Core-module generates a stream of pulsed single tone audio which corresponds to binary code of 10100101 and single tone frequency of 19 kHz as shown in figure 3.14.

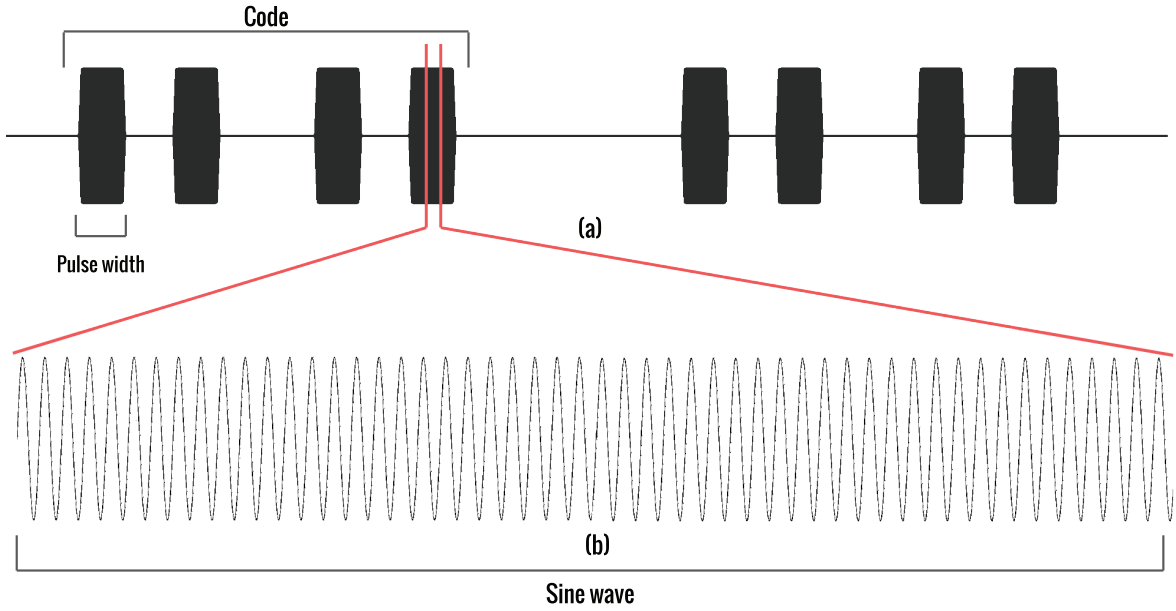


Fig. 3.14: Waveform of pulse single tone audio data (a) code sent via speaker (10100101) (b) sine wave of frequency 19000kHz

Validation sequence

For the validation of the microphone, we need two threads. First thread generates the audio stream and play it on the speaker and second thread analyses the audio stream received from the microphone to validate the microphone.

Speaker thread initializes the speaker to the default state and drains the existing audio date from the stream. Rise time of the waveform is not infinite, instead it rises slowly and linearly from zero to full amplitude in 3 ms. This helps in reducing high frequency harmonics. The audio data shown in figure 3.14 is played through the speaker. After this operation, the speaker thread is suspended.

Microphone thread initializes the microphone to the default state and flushes the existing audio data from the stream. Microphone waits for 441 samples to perform 441 point FFT. After calculating FFT, power around 19 kHz is calculated. The power calculated is then added to the time series queue of 50 samples, where the oldest power points are discarded. The *correlation* is calculated for received time-series with the original data sent from the speaker. If the *correlation* value exceeds the defined threshold then validation passes or else the process repeats until the timeout occurs. After the timeout it will retry microphone validation for two more times. If even after third try, the timeout occurs then validation is considered to be failed. The algorithm is explained in the flowchart shown in the figure 3.15.

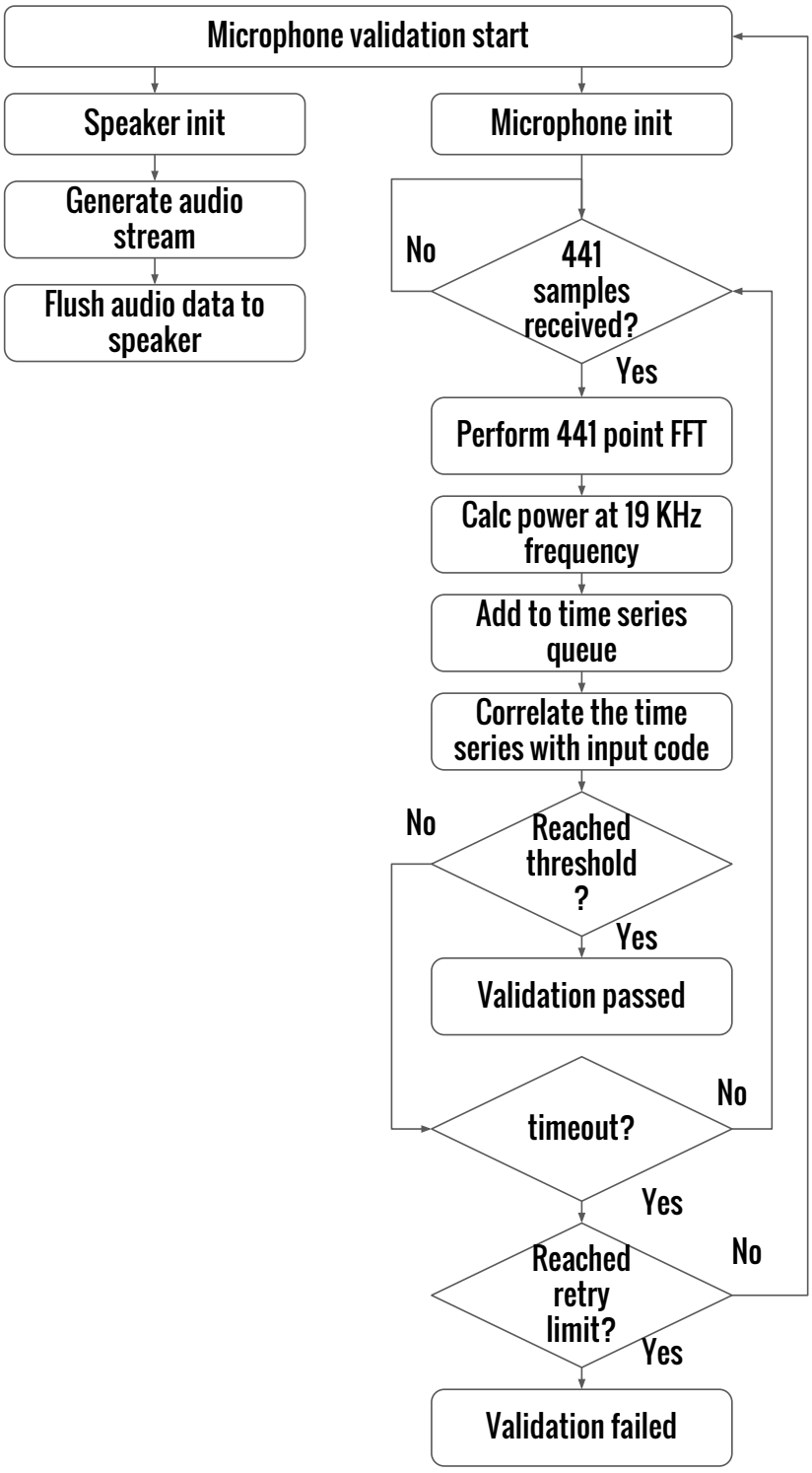


Fig. 3.15: Flow chart of microphone validation procedure

3.3.1.2 Panic Button validation

To validate the panic button, we emulate button press by shorting the pins of panic button by using the transistor as a switch, shown in figure 3.4a. First the initial condition is tested for open. If the test passes then transistor pin is set to emulate button press. Second condition is tested for short. If the test passes then the validation succeeds. The transistor pin is reset which emulates button release. If any test fails then it will retry the procedure two more times. If even after third try, the test still fails then validation is considered to be failed.

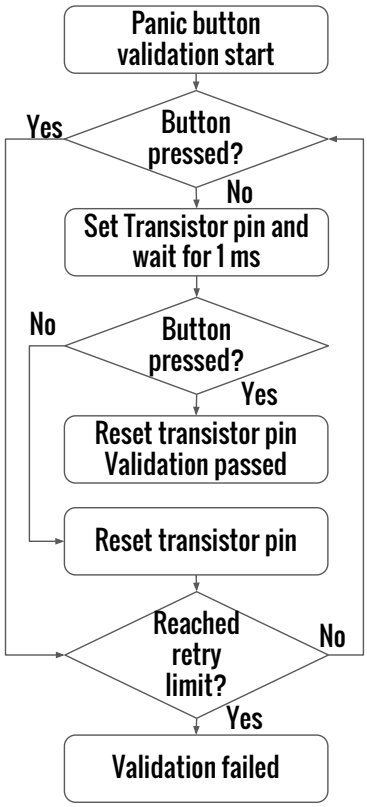


Fig. 3.16: Flow chart of Panic Button Validation

3.3.1.3 Firmware validation

Firmware validation is done after every soft/hard reset. To validate the firmware MD5 checksum is calculated for entire firmware and it is compared with the MD5 checksum stored in a non-volatile memory of core-module. If both match then firmware validation passes.

3.3.1.4 Tamper detection

We use an ultra low power MSP430 microcontroller for always-on tamper detection with its independent battery. Tamper is detected using a switch mounted inside the enclosure. MSP430 detects the tamper by detecting the change in logic level (from high level to low level) of this switch's pin. Once tamper is detected by MSP430 microcontroller, it permanently deletes the private key stored in the flash of the microcontroller. While servicing the device, lid opening will trigger the tamper detection event, but it can be suppressed by the server by the request of an authorized personal. The private key can be retrieved from the server and re-flashed into the flash of the microcontroller by an authorized personal.
The core-module regularly communicates with MSP430 and sends it a unique random number. MSP430 encrypts the number using TEA encryption algorithm (explained below) and sends it back. The core-module decrypts the number back from received data, on receiving mismatched numbers, tampering is notified to the server. The flow chart is shown in the figure 3.17.

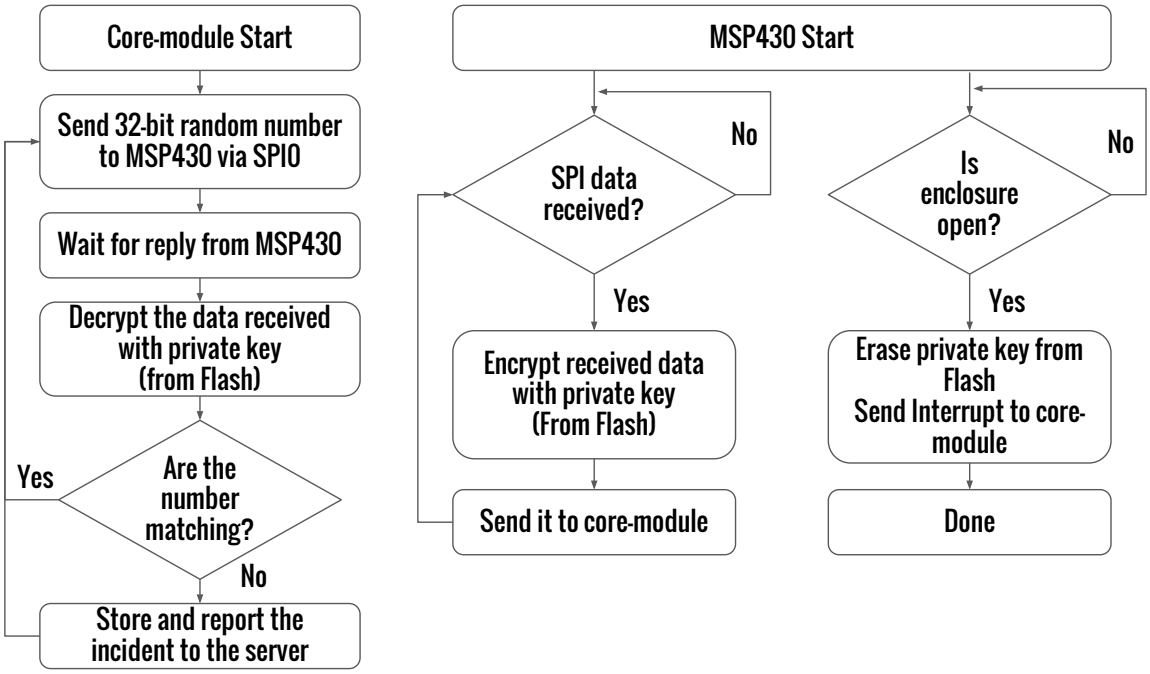


Fig. 3.17: Flow chart of tamper detection module

Encryption Algorithm

The algorithm used is a modified version of Tiny Encryption Algorithm[19]. Pseudo code of encrypt and decrypt subroutines are described in the figure 3.18, where data variable is a 32-bit data to be encrypted and key is the 64-bit private key used to encrypt it.

```
//encryption subroutine
encrypt(data,key)
   {v0,v1} = data //break 32-bit data into 2*16-bit
   {k0,k1,k2,k3} = key //break 64-bit key into 4*16-bit
   sum = 0
   delta = 0x9e37 //(√5−1)×2^15
   for i:(0:15)
       sum = sum+delta
       v0 = v0 + (((v1<<4) + k0) XOR (v1 + sum) XOR ((v1>>5) + k1))
       v1 = v1 + (((v0<<4) + k2) XOR (v0 + sum) XOR ((v0>>5) + k3))
   data = {v0,v1} //combine 2*16-bit into 32-bit

//decryption subroutine
decrypt(data,key)
   {v0,v1} = data //break 32-bit data into 2*16-bit
   {k0,k1,k2,k3} = key //break 64-bit key into 4*16-bit
   sum = 0xE370
   delta = 0x9e37 //(√5−1)×2^15
   for i:(0:15)
       v1 = v1 - (((v0<<4) + k2) XOR (v0 + sum) XOR ((v0>>5) + k3))
       v0 = v0 - (((v1<<4) + k0) XOR (v1 + sum) XOR ((v1>>5) + k1))
       sum = sum - delta;
    data = {v0,v1} //combine 2*16-bit into 32-bit
```

Fig. 3.18: The pseudo code for the TEA encryption algorithm

The constants used are delta 0x9e37 ($(\sqrt{5}-1)*2^{15}$) and sum 0xE370 ((delta*16)*0xFFFF).

3.3.2 Audio streaming module

Streaming of realtime audio data with minimum latency is essential to the system for further analysis and storage as evidence in the hard-drives of the server. Raw data is not suitable for streaming because of the sheer size of it. To compress the data, we have used Vorbis ogg[20] open-source patent free audio compression format, which is suitable the streaming and storage of audio data.

we used raw TCP socket to stream the audio data from the *PanicButton device* to the server. We need a single stream of audio from the device, so the overhead of HTTP header is not required. The extra information required by the server is embedded inside the Vorbis ogg comments

page. The data flow is shown in figure 3.19.

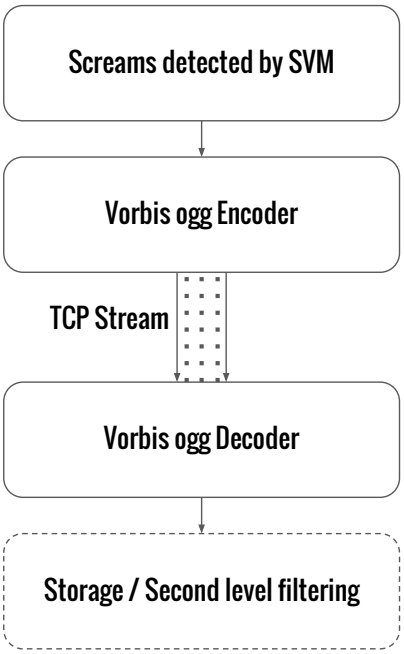


Fig. 3.19: Data flow in streaming of audio data

We used libvorbis, libogg and libvorbisenc open source libraries to encode and decode from the vorbis ogg format.

3.3.3 Global positioning system module

The NEO-6M module from ublox uses UART protocol for sending the GPS data to the host controller. It sends data in National Marine Electronics Association(NMEA) format. NMEA format consists of sentences, whose first word is called the data type, which defines the interpretation of the rest of the sentence. The NEO-6M module sends data with GPGGA and GPRMC data types. Core-module decodes the location and date-time information from the NMEA format.

3.3.4 Panic button module

The system needs realtime button response. To minimize the response latency, we used pin interrupt and timer combination debouncing technique instead of using only timer based debouncing as shown in figure 3.20

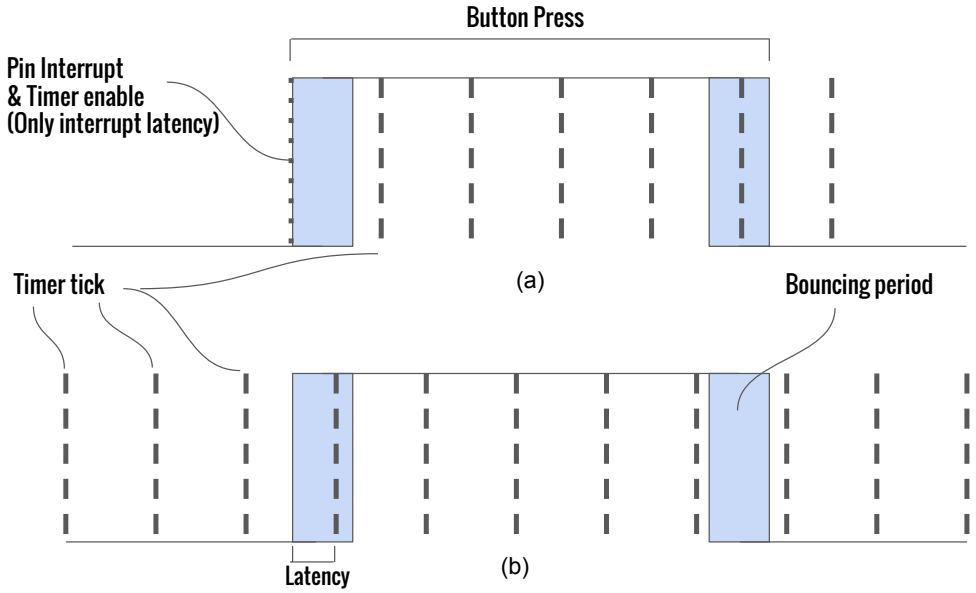


Fig. 3.20: Deboncing technique(a) Combination of pin interrupt and timer to reduce latency. (b) Only using timer interrupt.

The maximum period of debouncing is expected to be around 10ms[21]. We used timer tick period of 15 ms, which is greater than maximum debouncing period. In case (b) maximum latency is 15 ms and in case (a) maximum latency is of only six cycles (0.375 us) for pin interrupt. Hence we opted for method (a).

3.3.5 Audio-based trigger module

As described in study section, we decided to use Support Vector Machine (SVM) as underlying machine learning algorithm and Mel-Frequency Cepstral Coefficients (MFCC) as feature set for this algorithm to detect scream. Let's have a look at feature extraction procedure used in this project.

3.3.5.1 Feature Extraction

Mel-Frequency Cepstral Coefficients (MFCC), which acts as feature set, are extracted from live stream of input audio. Figure 3.21 depicts the Feature extraction procedure.

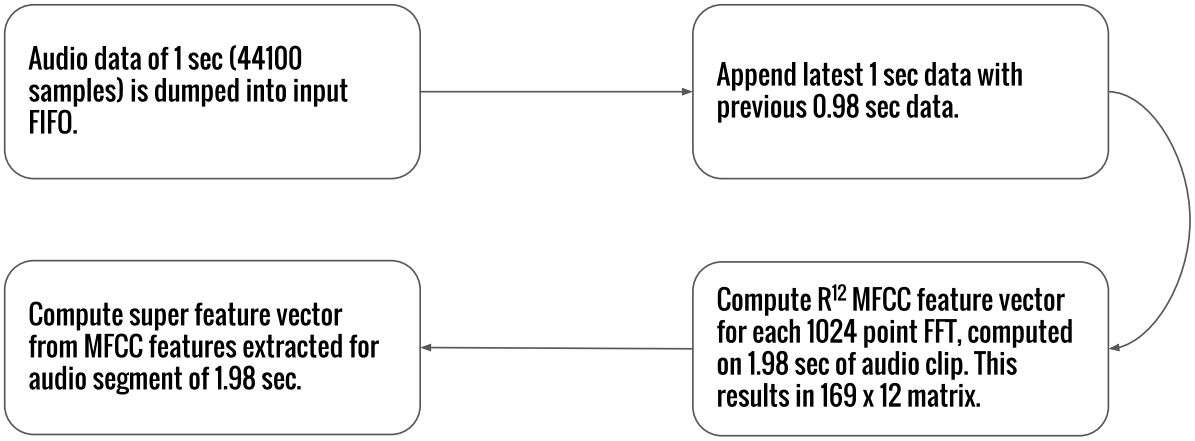


Fig. 3.21: Feature extraction process

Every second audio data is fetched from sound card and dumped into an input fifo, this 1 second of audio data is then appended with previous 0.98 second (0.98 second is multiple of 30 ms, which is the window size used in the FFT operation) of audio data. MFCC features are extracted for this composite audio clip.

MFCC are computed by first chopping the audio signal into tiny overlapping segments, these segments are multiplied with a window function like Hamming window; Fourier transform is computed over this windowed audio segment. The power spectrum is wrapped according to Mel-scale to adapt the frequency resolution of human ear. This spectrum is then segmented into number of critical bands using a triangular filter bank; Inverse Discrete Cosine Transform (DCT) applied to the logarithm of filter bank's outputs result in MFCC vector.

We obtain super feature vector of R^{24} for each 1.98 sec of audio data. This super feature vector is obtained by taking column mean and standard deviation of 169 x 12 matrix, which was obtained by extracting MFCC(ignoring 0^{th} MFC coefficient) for audio data of 1.98 sec. Once features are extracted, they are used for training the SVM.

3.3.5.2 Training of SVM

Training-

Like all machine learning algorithms, we need to train our SVM using a labeled data set. Labeled data set contains data that is painstakingly labeled by a human, classifying whether a particular 1.98 second clip is a scream. Clip is labeled 1 for a scream and -1 for not a scream. Figure 3.22 depicts the training process.

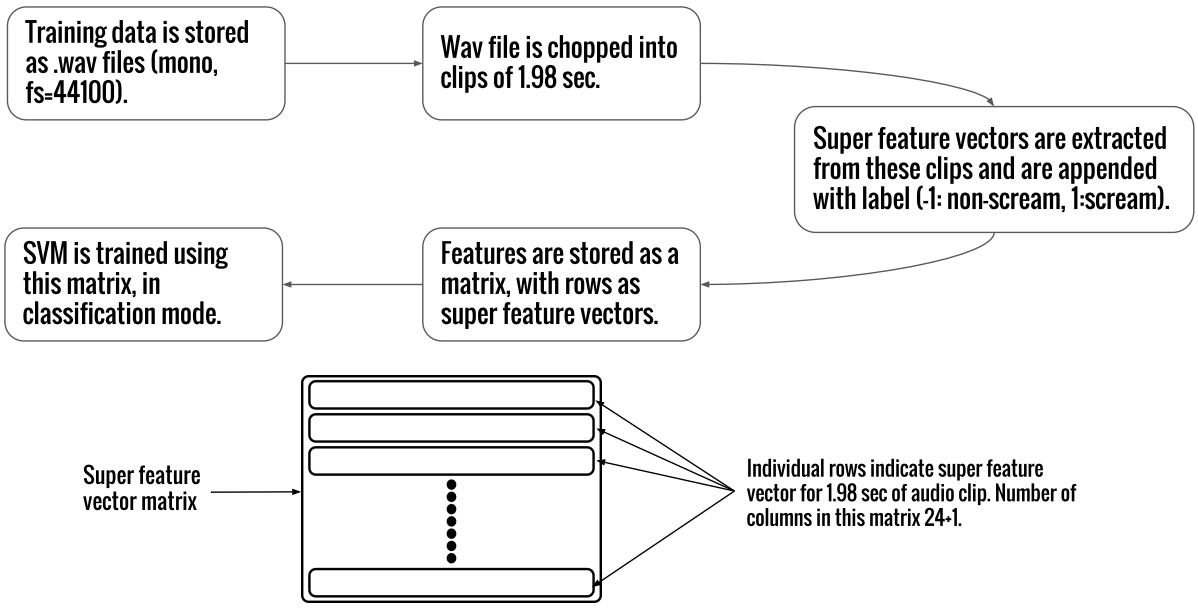


Fig. 3.22: Training process of the SVM

Support Vector Machine (SVM) uses a kernel to transform data into higher dimension space and finds a linear separating hyper plane with maximal margin in this higher dimensional space. We are using radial basis function (RBF) as kernel, due to its ability to model complex data patterns. RBF kernel $K(x_i, x_j)$ is defined as-

$$K(x_i, x_j) = exp(-\gamma \parallel (x_i - x_j) \parallel^2), \gamma > 0.$$

For training SVM that uses RBF kernel, we need to specify values of two parameters -

1. C (penalty factor used in cost function) and
2. γ (variance of the RBF kernel).

We used following steps to get optimum value of C and γ and use them for the training-

1. Arrange data in proper format i.e., [data,label],
2. Randomize the data in cross-validation set,
3. Consider the RBF kernel $K(x, y) = e^{(-\gamma \| (x_i - x_j) \|^2)}$,
4. Use cross-validation set to find the best values for parameters C and γ,
5. Use these values of C and γ, to train the SVM from training data set and

6. test.

We will dwell into details of these steps in section 3.3.5.3, first lets have a look at data set used in this project.

Data set-

Data set used in this project is called IIITD Urban Environment Context database (IUEC)[15]. It contains audio clips from following environmental contexts-

1. Conversation (narrations)
2. Human gathering (clubs and meetings)
3. Indoors (indoor sounds)
4. machinery (fan, shaver, laptop-fan, exhaust, oven, mixer, washing machine and vacuum cleaner)
5. Multimedia (movie audio and TV audio)
6. Outdoors (bus stand, road traffic, market, temple, rickshaw, rain, metro and playground)
7. distress (scream and crying sounds)

Table II quantifies the content of this dataset.

Context	**Samples of 2 sec**
Conversation	623
Human gathering	1621
Indoors	4169
Machinery	3548
Multimedia	3698
Outdoors	5142
Scream	464
Total	**19265= 10 Hrs 42 Mins**

Table II: IUEC Data set details

Apart from this data set, we collected data using the *PanicButton device* prototype. Table III quantifies its content.

Context	**Samples of 2 sec**
Conversation	1144
Multimedia	1828
Outdoors	2623
Scream	464
Total	**6059= 3 Hrs 22 Mins**

Table III: Details of data collected from panic button prototype

Scream samples were obtained by re-recording scream data available in IUEC data set. We combined this two data sets to get a composite data set, whose content is quantified in Table IV

Context	**Samples of 2 sec**
Conversation	1767
Human gathering	1621
Indoors	4169
Machinery	3548
Multimedia	5526
Outdoors	7765
Scream	928 (3.6% of complete data set)
Total	**25324= 14 Hrs 4 Mins**

Table IV: Composite Data set details

This complete data set gets divided into two equal parts called training data set and cross-validation data set. Training data set is used for training the SVM and cross-validation dataset is used for tuning the SVM. Our data set is highly skewed i.e., scream samples constitutes only 3.6% of the complete data set. This makes benchmarking of trained SVMs non-conventional. Section 3.3.5.3 addresses this problem in detail.

Finally, a test clip of 22 minutes was created to quantify performance of the trained SVM. This 22 minute clip was carefully designed to cover audio samples from every context. Obtained

results are mentioned in Section 3.3.5.4.

3.3.5.3 Finding C and gamma

To find the optimum value of C and γ, we performed grid search for these parameters on cross-validation data set. Five fold cross-validation data set (4 part for training and 1 part for testing in circular fashion for a particular combination of C and γ) is used. Figure 3.23 shows the process of finding optimum C and γ.

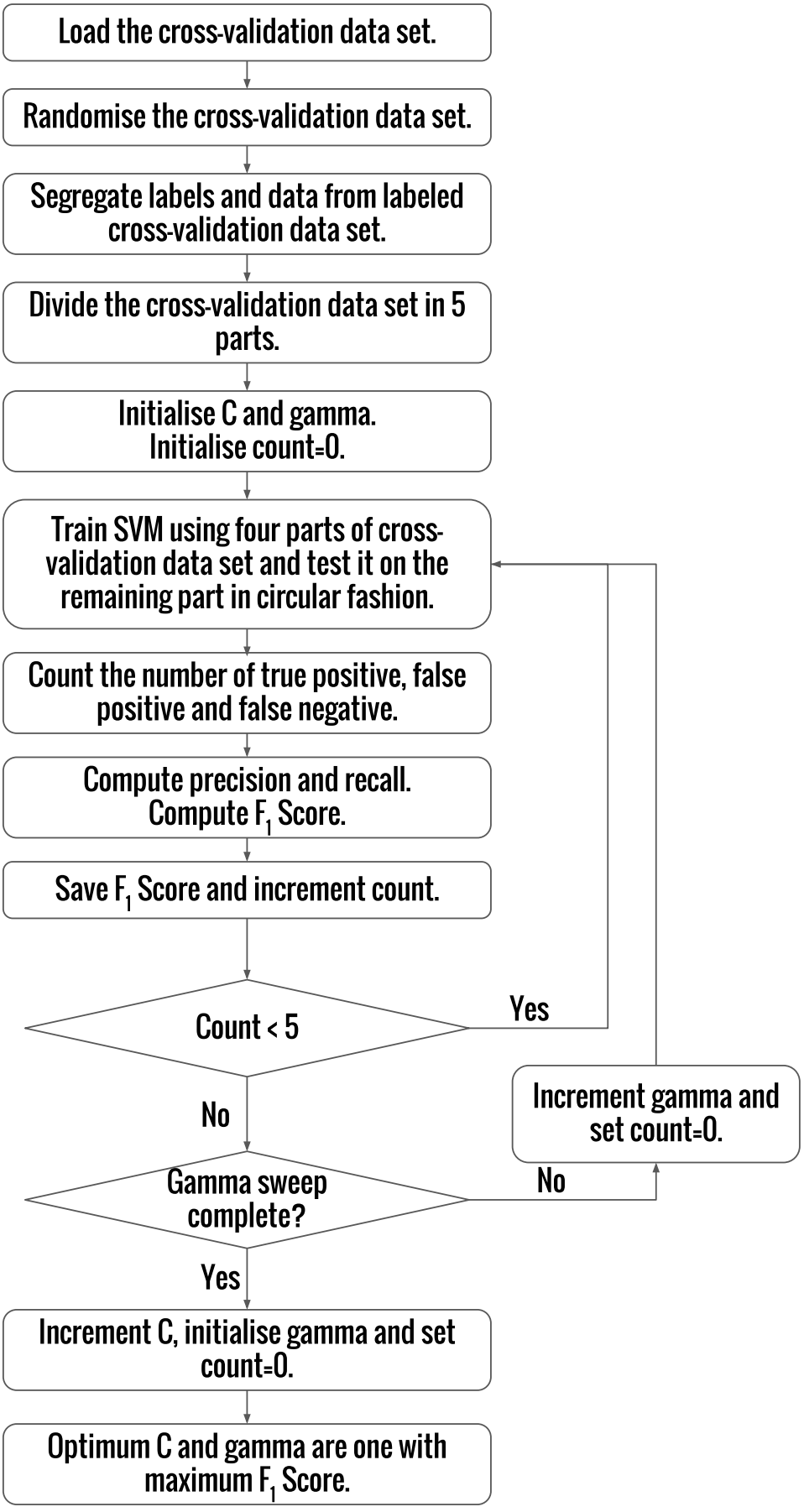


Fig. 3.23: Cross-validation process for obtaining optimum C and gamma

Method shown in figure 3.23 is a brute force method to find optimum value of C and γ. We use F_1 score as figure of merit (FOM); Accuracy can't be used as FOM due to skewed nature of data set.

F_1 score is computed using precision (prec) and recall (rec):

$$F_1 = \frac{2.prec.rec}{prec + rec},$$

We can compute precision and recall by:

$$prec = \frac{t_p}{t_p + f_p}$$

$$rec = \frac{t_p}{t_p + f_n}$$

where

- t_p is the number of true positives: the ground truth label says it's a scream and our algorithm correctly classified it as a scream.
- f_p is the number of false positives: the ground truth label says it's **not** a scream, but our algorithm incorrectly classified it as a scream.
- f_n is the number of false negatives: the ground truth label says it's a scream, but our algorithm incorrectly classified it as **not** a scream.

In any automated alarm system, ideally we should have zero false negative, practically it should be as small as possible. Following figure show us the F_1 score obtained by performing grid search for the parameters C and γ on the cross-validation set.

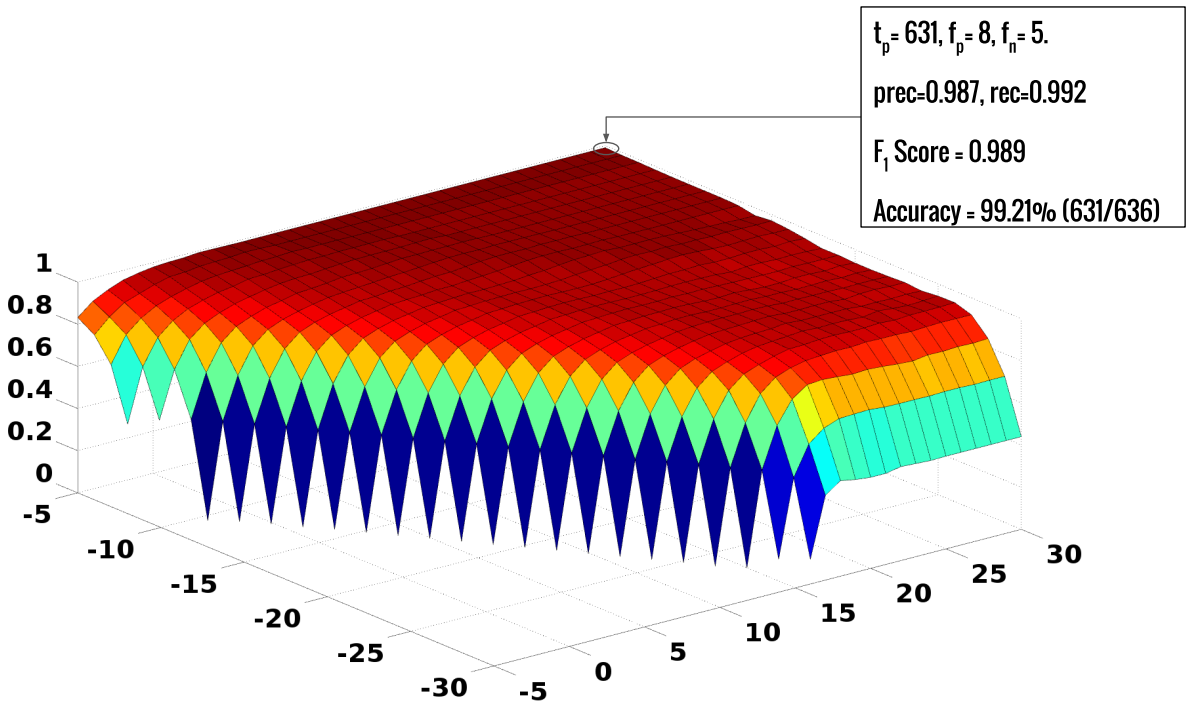


Fig. 3.24: F1 score obtained from grid search for C and gamma on CV set

Once we have the value of C and γ, we can use that value to train our SVM. This process of first obtaining appropriate value of C and γ and then training the final SVM, helps us in avoiding machine learning traps like over-fitting and under-fitting. Performance of the trained SVM is quantified in section 3.3.5.4, besides figure 3.24 also quantifies the performance of best SVM, over the five fold cross-validation set.

3.3.5.4 Performance of the SVM

We performed benchmarking using a 22 minute test set, this test set contains 24 screams of different lengths and strengths. We were able to detect 23 screams with 1 false negative and 3 false positive, giving us accuracy of **95.8%** and F_1 score of **0.92** with latency of 1.480 ms.

3.4 Industrial design

3.4.1 Module design

Connected panic button's industrial design is divided as follows-

- Design of top panel,
- Design of panic button,
- Design for the tamper detection circuit and,
- Design of the base.

3.4.1.1 Design of top panel

Top panel of the *PanicButton device* acts as host to emergency button, speaker and microphone. The panic button's stem passes through the center of the front panel. Just beside the grove for passing the stem of emergency button, is a whole of radius 4 cm, which acts as speaker housing. One more hole of radius 0.5 cm is kept besides the house of speaker for mounting microphone.

Speaker and microphone are assembled in a manner that the sound produced from speaker gets directly coupled to the microphone from outside the enclosure. A cap was designed around the speaker, to minimize the coupling of sound from inside the enclosure. Bottom part of the panic button plays an important role in this coupling of audio and is discussed in section 3.4.1.2. Figure 3.25 shows the top view of the top panel and figure 3.26 shows the bottom view of the top panel.

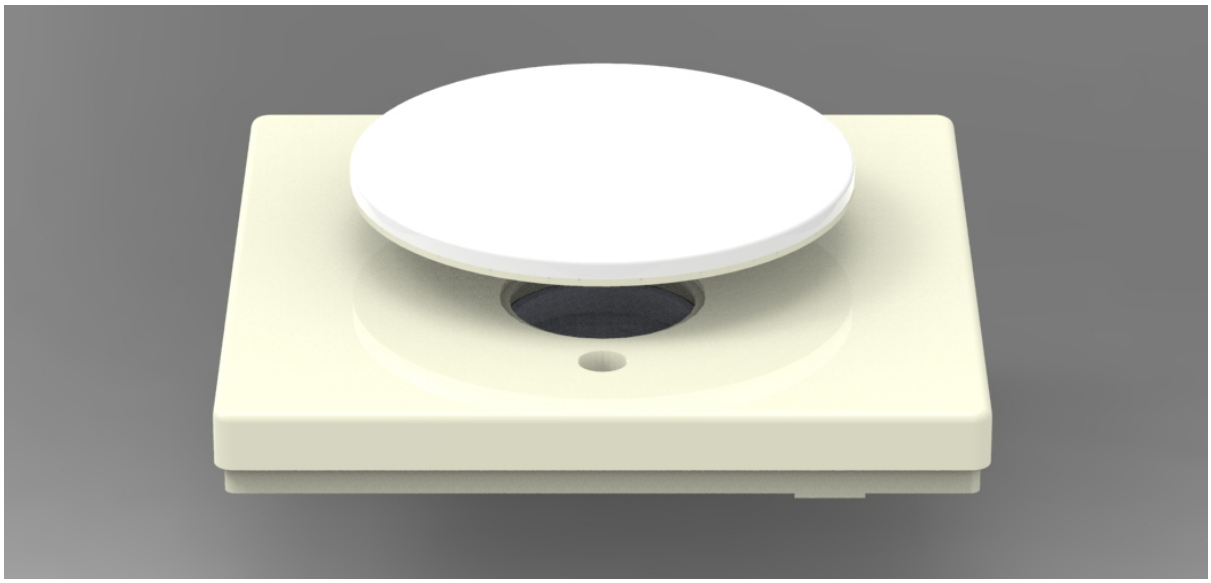


Fig. 3.25: Top view of the top panel of the *PanicButton device*

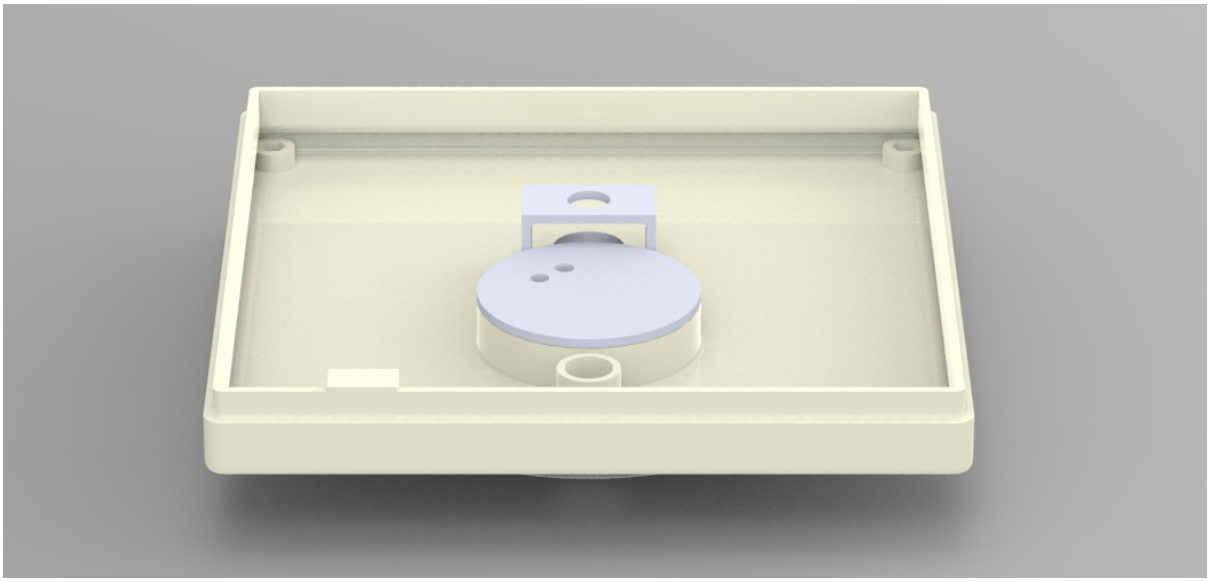


Fig. 3.26: Bottom view of the top panel of the *PanicButton device*

3.4.1.2 Design of panic button

To make panic button easily accessible at all times, we took care of following two things.

- Area of the panic button equals the average area of a adult human's palm and
- Back-light is added to the panic button to make sure that it is easily visible in the night.

An important part of the emergency button design is to make sure that the button does not cover the microphone, while completely covering the speaker; Proper placement of these three component and size of button can take care of this.

Bottom part of the panic button acts as a reflector for sound produced by the speaker. For efficient performance of microphone validation procedure, it is important that the waves produced by the speaker are channeled to the microphone. To accomplish this task, bottom part of panic button was given a concave shape such that microphone lies at the focal point of the concave surface.

Stem of the emergency button was made hollow to pass wires for the back-light mounted inside the emergency button.

3.4.1.3 Design for the tamper detection circuit

Tamper detection circuit takes the input from a push button, that gets pushed once the top panel is mounted on the base of the enclosure. We designed a projection coming out of the wall to support the mounting of the switch. Figure 3.27 shows the placement and shape of the projection.

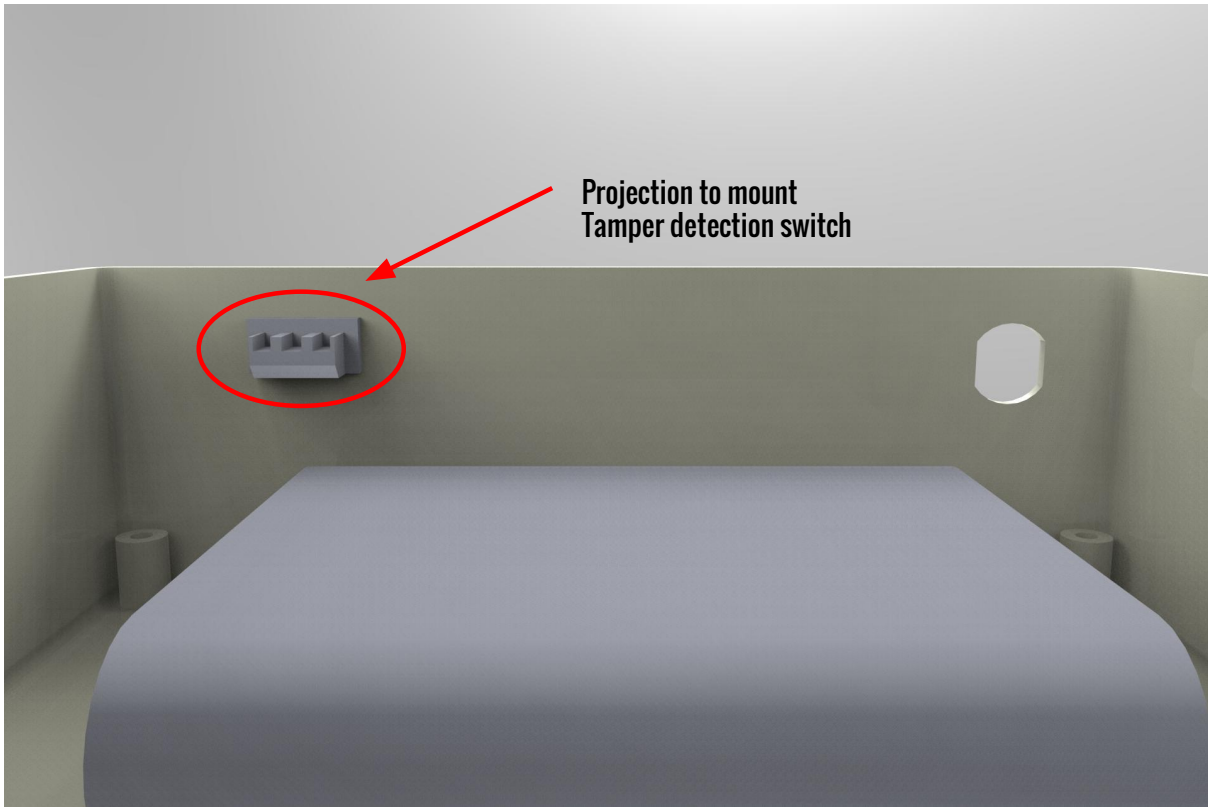


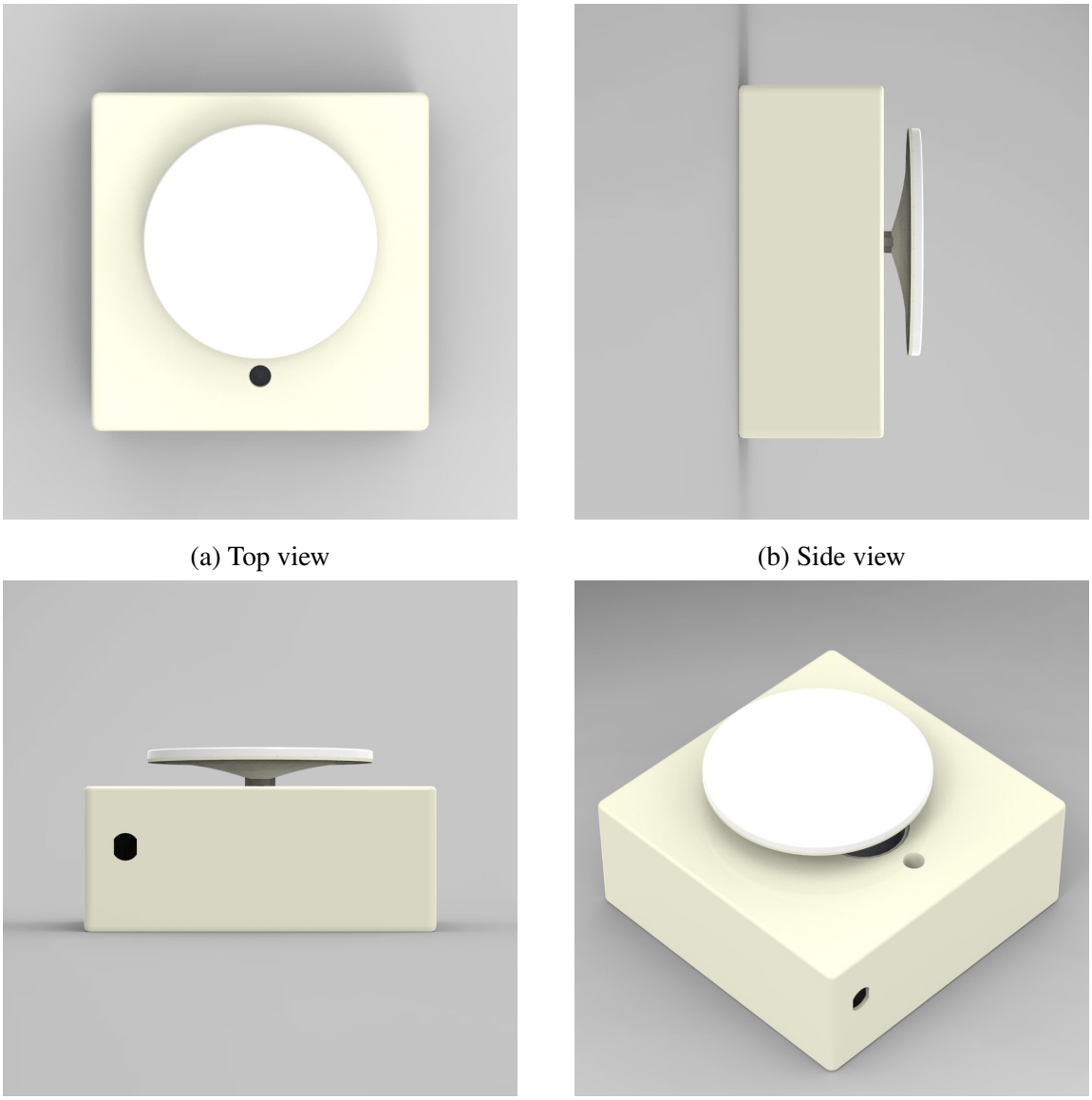
Fig. 3.27: The projection to mount the tamper detection switch

3.4.1.4 Design of the base

The base of the enclosure act as the host for all components residing inside the *PanicButton device*. We placed 4 mounting holes for the raspberry pi and 4 m3 screw holes for bringing top and base together.

3.4.2 Putting modules together

Figure 3.28 shows the *PanicButton device*, once all modules are brought together and figure 3.29 shows the exploded view of the *PanicButton device*.



(a) Top view

(b) Side view

(c) Front view

(d) Isometric view

Fig. 3.28: Top, side, front and isometric views of the enclosure

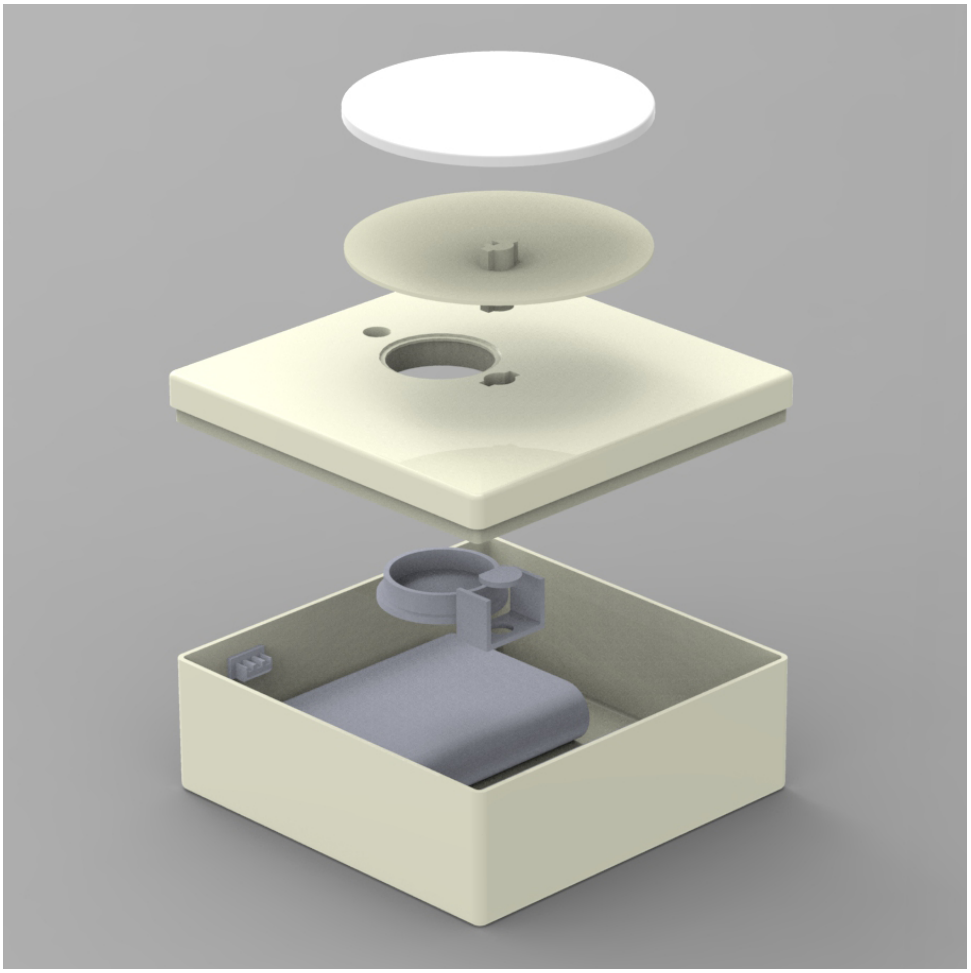


Fig. 3.29: Exploded views of the enclosure

Chapter 4

Engineering and fabrication

4.1 Product structure

The product looks like a simple box from outside but it's more than that. Beneath the button is a speaker and beside it lies a microphone. Top panel is a 3D printed panel, housing emergency button, speaker and microphone. Bottom side of top panel takes care of connection between the custom build button assembly and a push button switch, provides support for speaker housing and a projection to press push button of tamper detection module. The base hosts most of the electronics. Half of the base part is occupied by raspberry pi and its add-on board, other half is occupied by the battery and its charging circuitry. Side wall of base part exposes power connector to charge the secondary battery and power the entire electronics. Figure 4.1 shows the block diagram of the complete design.
In this chapter, we look at hardware tools, software tools and assembly/fabrication procedures required to make *PanicButton device*.

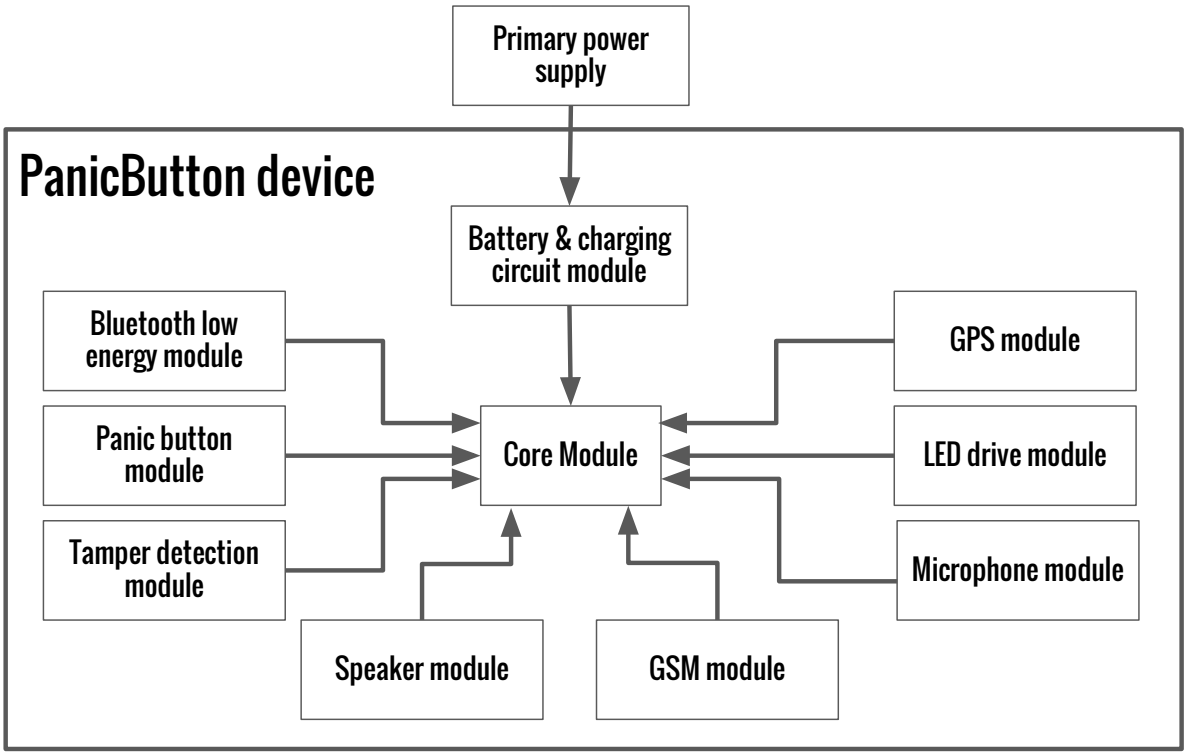


Fig. 4.1: Block diagram of the *PanicButton device*

4.2 Hardware modules

The *PanicButton device* can be divided into three parts namely raspberry pi, add-on board for raspberry pi and the enclosure. Base part of the enclosure acts the hosing for these modules.

4.2.1 Raspberry pi

Raspberry pi's sd card with modified Raspbian image, must be installed into the raspberry. Raspberry pi should be mounted on the base using four M2 screws of 0.5 cm length.

4.2.2 Add-on board for raspberry pi

Add-on board contains the BLE module, programming header for BLE module, GPS module, speaker module, emergency button headers and tamper detection module. Figure 4.2 shows the circuit schematic for the raspberry pi's add-on board.

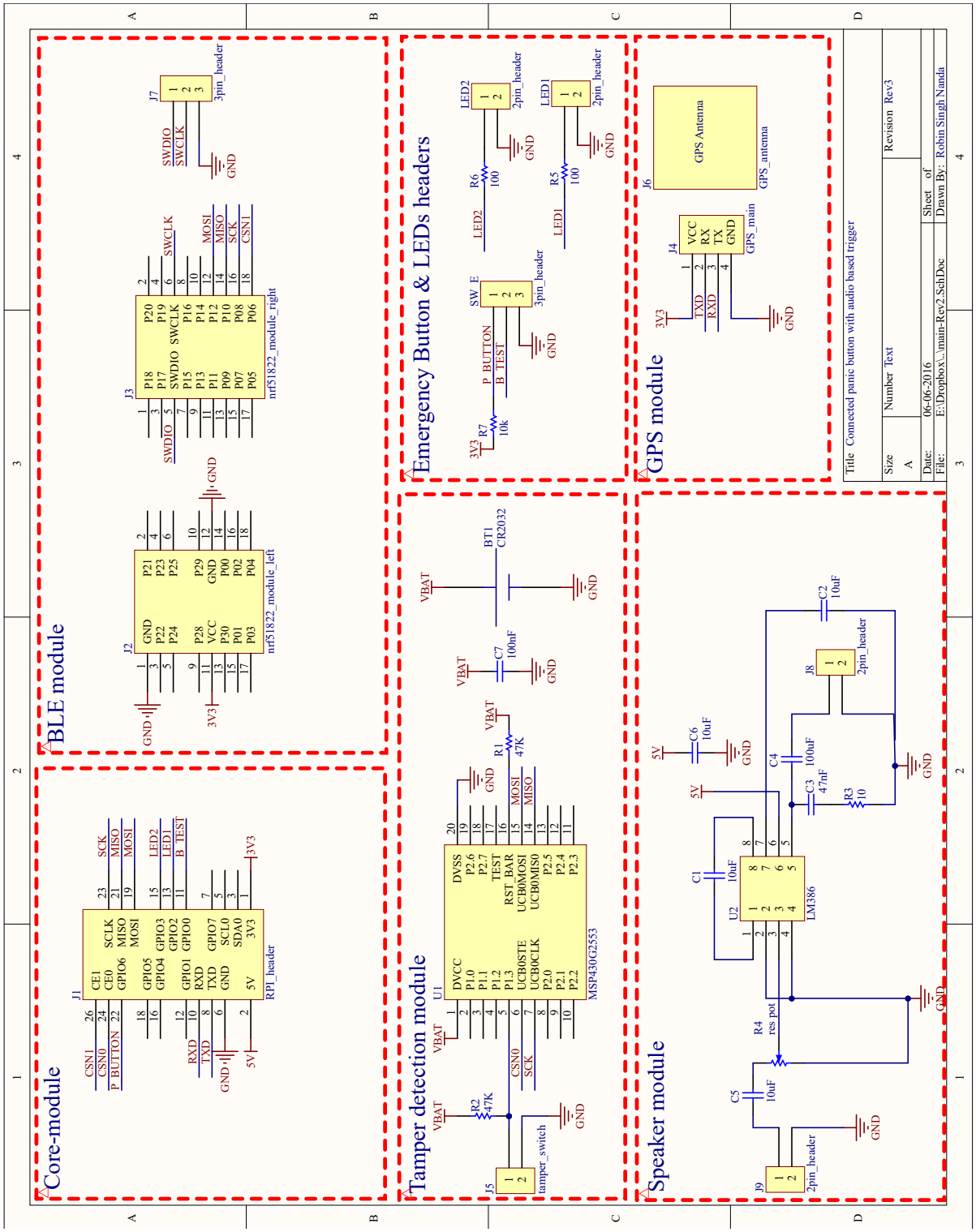
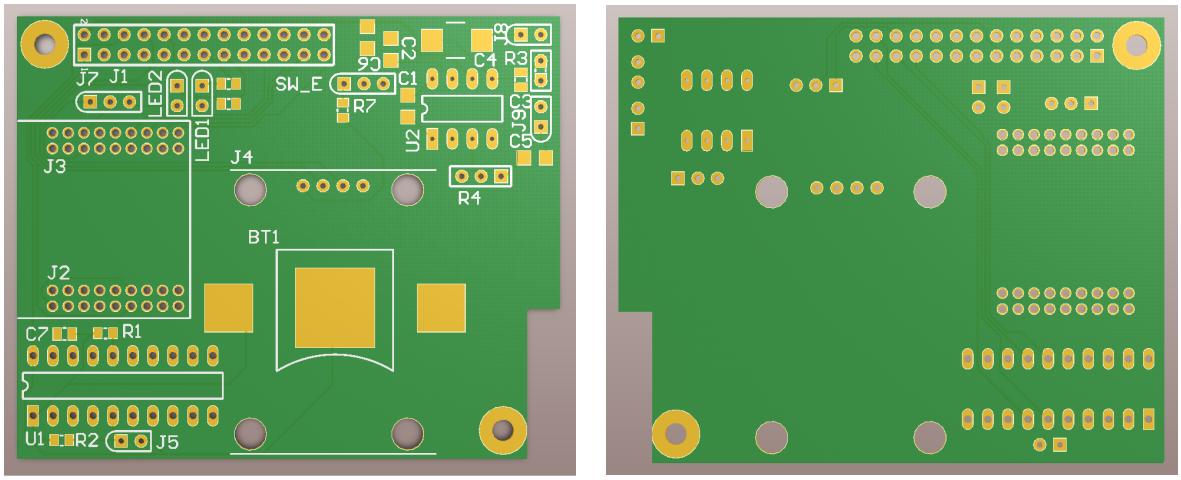


Fig. 4.2: Schematic of raspberry Pi add-on board

The add-on board has a non rectangular shape because of the ethernet and USB connectors of the raspberry pi. Mounting holes of the add-on board are aligned to the mounting holes of the raspberry pi board. Figure. 4.5a shows the top layer of add-on board's PCB and figure. 4.3b shows the bottom layer of the add-on board's PCB. Our application did not impose any constrain on the thickness of the substrate, hence common substrate of 1.6 mm thickness was used. All signal tracks on the PCB are of 10 mil width. The add-on board connects to raspberry pi via 26 pin female header and can be secured using M3 mounting screws.



(a) Top side

(b) Bottom side

Fig. 4.3: Top side and bottom side of PCB of Raspberry pi add-on board

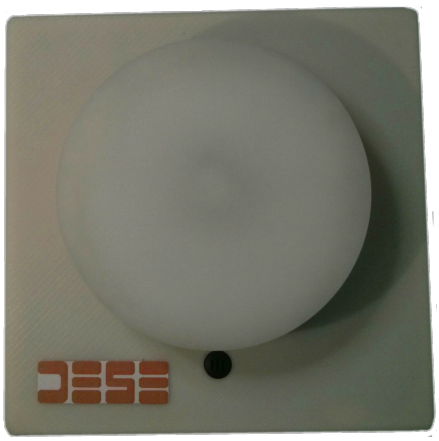
Figure 4.4 shows the bill of materials used in this project.

4.2.3 Enclosure of *PanicButton device*

Enclosure was fabricated using a 3D printer that uses fused deposition modeling technique. Many parts of top panel were fabricated individually. Figure 4.5 shows different views of the fabricated panic button enclosure.

Description	Unit price	Quantity	Total price	Supplier	Supplier part number	Manufacturer Part Number	Supplier Link
CR2032 battery holder	23.4255	1	23.4255	Digi-Key	BK-5067-ND	BK-5067	http://goo.gl/OPV7HR
10uF	25.4334	4	101.7336	Digi-Key	399-3684-1-ND	T491A106K010AT	http://goo.gl/d1GI0D
47nF	6.693	1	6.693	Digi-Key	399-1166-1-ND	C0805C473K5RACTU	http://goo.gl/eZvN97
100uF	66.93	1	66.93	Digi-Key	399-8364-1-ND	T491D107K006AT	http://goo.gl/3U4Z0H
100nF	6.693	1	6.693	Digi-Key	1276-1007-1-ND	CL21F104ZBCNNNC	http://goo.gl/aH2RzF
26-pin 2.54mm Female	211.5657	1	211.56573	Digi-Key	147046-1-ND	147046-1	http://goo.gl/n6rkQn
18-pin 2mm Female	124.4898	2	248.9796	Digi-Key	609-2663-ND	63453-118LF	http://goo.gl/lYxqtb
4-pin 2.54mm Female	40.158	1	40.158	Digi-Key	S7002-ND	PPTC041LFBN-RC	http://goo.gl/5NrLL2
3-pin 2.54mm male	17.4018	2	34.8036	Digi-Key	3M9448-ND	961103-6404-AR	http://goo.gl/3byLdM
2-pin 2.54mm Male	12.0474	4	48.1896	Digi-Key	609-4434-ND	77311-118-02LF	http://goo.gl/GsNBLI
47K	6.693	1	6.693	Digi-Key	RMCF0805JT47K0CT-ND	RMCF0805JT47K0	http://goo.gl/dyGuVS
10K	6.693	2	13.386	Digi-Key	RMCF0805JT10K0CT-ND	RMCF0805JT10K0	http://goo.gl/zY0Qnz
10E	6.693	1	6.693	Digi-Key	311-10.0CRCT-ND	RC0805FR-0710RL	http://goo.gl/rtFvEF
100E	6.693	2	13.386	Digi-Key	311-100CRCT-ND	RC0805FR-07100RL	http://goo.gl/4QYSJk
10K	68.2686	1	68.2686	Digi-Key	3362P-104LF-ND	3362P-1-104LF	http://goo.gl/xwMcrS
MSP430G2553	178.0338	1	178.0338	Digi-Key	296-28429-5-ND	MSP430G2553IN20	http://goo.gl/c16m5z
LM386	45.52511	1	45.525116	Digi-Key	296-43959-5-ND	LM386N-3/NOPB	http://goo.gl/QrmK5Q
Raspberry pi 3	3000	1	3000	Element14	2525226	RASPBERRYPI3-MODB-1G	http://goo.gl/StQ0rx
Ublox NEO-6M	1199	1	1199	Amazon	1465206367	GPS	http://goo.gl/HpX7aC
nRF51822 module	2162	1	2162	Amazon	1465206419	BLE	http://goo.gl/cLuHwO
Emergency switch	76.9695	1	76.9695	Digi-Key	EG2021-ND	PS1024ALBLK	http://goo.gl/fyWlWX
Red LED	10.0395	4	40.158	Digi-Key	C503B-RAS-CY0B0AA1-ND	C503B-RAS-CY0B0AA1	http://goo.gl/8GxoQ8
Green LED	16.0632	4	64.2528	Digi-Key	C503B-GAN-CB0F0791-ND	C503B-GAN-CB0F0791	http://goo.gl/asyfRn
8ohm speaker	145.9074	1	145.9074	Digi-Key	102-2502-ND	CDM-20008	http://goo.gl/QfAf6D
Limit switch	48.8589	1	48.8589	Digi-Key	CKN10157-ND	ZMA00A150L04PC	http://goo.gl/RM0Fhe
Enter USB sound card	93	1	93	Amazon	1465206950	Sound card	http://goo.gl/Y11f6
Collar microphone	345	1	345	Amazon	1465206968	Mic	http://goo.gl/2rUIFf
		43	8296.3037				

Fig. 4.4: Bill of materials



(a) Top view

(b) Side view

(c) Front view

(d) Isometric view

Fig. 4.5: Different views of the enclosure

4.3 Software modules

4.3.1 Raspberry pi core-firmware

All the firmware of raspberry pi core is implemented in java language, which requires Oracle JavaSE v8 to run, can be installed using the following command.
sudo apt-get install oracle-java8-jre
The firmware for raspberry pi is developed using Intellj Idea Ultimate 2016.1 Integrated Development Environment (IDE). The firmware is packaged in Java archive (JAR) file. The raspberry pi must run the jar file on boot up.

4.3.2 nRF51822 firmware

The firmware for nRF51822 based Bluetooth low energy module is developed using Keil μVision 5 IDE. The firmware is written in plain C and packaged into a hex file, which can be programmed to nRF51822 SOC using Serial Wire Debug (SWD) interface using Segger Jlink debugger.

4.3.3 MSP430 firmware

The firmware for MSP430 module is developed using Texas Instruments Code Composer Studio (TI-CCS). The firmware is written in plain C and packaged into a hex file, which can be programmed to MSP430 microcontroller using Texas Instruments MSP-FET430UIF debugger or Texas Instruments MSP430 Launchpad debugger.

4.3.4 Scream detection firmware

Scram detection algorithm was implemented in GNU Octave 3.8.2. Octave is a high-level interpreted language, primarily intended for numerical computations. It provides capabilities for the numerical solution of linear and nonlinear problems, and for performing other numerical experiments. GNU Octave can be installed in linux machine using following command-
sudo apt-get install octave
Signal and image package of octave are also required for machine learning firmware development. These packages can be install using following commands.

sudo apt-get install octave-signal
sudo apt-get install octave-image

4.3.5 Server code

The server code is tested for running on Mosquitto MQTT borker v1.4.8, Apache tomcat server v8 and MySQL database server v5.7.12, other versions of servers might not work as expected. The server code is developed in Intellj Idea Ultimate 2016.1. The server code can be divided into the following parts.

- Mosquitto application code
- Apache tomcat servlet code

4.3.5.1 Mosquitto application code

The mosquitto application code is packaged into a JAR file, which requires Oracle JavaSE v8 or higher. The mosquitto application requires Mosquitto MQTT broker and MySQL database to be running.

4.3.5.2 Apache tomcat servlet code

The apache tomcat servlet code in packed into a Web-application archive (WAR) file, which also requires Oracle JavaSE v8 or higher. The apache tomcat servlet code requires Apache tomcat server and MySQL database server to be running.

4.3.6 Android app

The Android application requires Android version 4.3 or higher and bluetooth v4.0 or higher to run reliably. The android application was developed using Android Studio v2.0. The app is packaged into a Android application package (APK) file, which can be side-loaded into the Android phone or tablet.

4.4 Key configuration parameters

The key configuration parameters a

Parameter name	size	module	value
panicButtonID	128-bit	core-module	From server
publicRSAKey	256-bit	core-module	From server
teaKey	64-bit	core-module	From server
teaKey	64-bit	tamper-detection module	From server

Table I: Feature extraction time using MFCC and spectral feature as feature vector

4.5 System integration

In the following sections, we look at internal and external connections going into *PanicButton device*.

4.5.1 Internal connections

- The raspberry pi add-on board connects to raspberry pi using 26-pin female header which is paired to 40-pin male header.
- The USB sound card should be connected to a USB port of the raspberry pi.
- The audio output from raspberry pi connects to the header J9 on the add-on board.
- The speaker connects to J8 header of add-on board.
- The microphone connects to the mic port on the USB sound card.
- The panic button connects to 3-pin SW_E header on add-on board.
- The red and green LEDs connects to LED1 and LED2 header on the add-on board respectively.
- The GPS module connects 4-pin header J4 on the add-on board.
- The nRF51822 BLE module connects to header J2 and J3 on the add-on board.

- The tamper detection button connects to J5 header on add-on board.
- The USB power from USB power bank connects to Raspberry pi power in port.
- The USB power from USB charger connects to Power bank's power in.

4.5.2 External connections

The *PanicButton device* requires a 12V DC voltage from vehicle battery or any other source, going into dc-power jack on the right side of the enclosure.

Chapter 5

Concluding remarks

5.1 User instructions

5.1.1 Auto validation

User should give the Android app all the necessary runtime permissions. The smartphone internet data and BLE should be turned on. The app will automatically detect nearby panic button system and notify user once, the validation procedure completes and the details of vehicle and driver are available. If the validation fails the phone will warn user about the situation.

5.1.2 Manual validation

In case of manual validation, user should go to app on his android smartphone and tap on Validate button. This will start the validation procedure with the nearest gateway device and once the validation completes, the user will be notified with the status of validation and the details of the vehicle and driver.

5.1.3 Emergency request

The user can either press the panic button on the *PanicButton device* or scream. In either case the emergency request will be sent to the nearby PCR.

5.2 Suggestions for next gen

In the next gen of this device, following things can be modified-

- Form factor of the device, it can be made smaller.
- Performance of the scream detection unit can be improved, through application specific training of the SVMs.
- Porting the audio detection algorithm to a low-power microcontroller.

5.3 Future scope

This product has a great potential at reducing the number of crime in public transport vehicle. That being said, the current status of product is incapable of fulfilling this vision due to its large form factor and few field trials.

Bibliography

[1] National Crime Records Bureau of India. http://ncrb.nic.in.

[2] Artemis safety wearable. http://www.artemisfashion.com/.

[3] Amulyte safety wearable. http://vandrico.com/wearables/device/amulyte.

[4] Stiletto safety wearable. http://stiletto.is/.

[5] CUFF safety wearable. https://www.cuff.io/.

[6] SAFER safety wearable. https://www.leafwearables.com/.

[7] Revolar safety wearable. https://revolar.com/.

[8] First sign safety wearable. http://www.firstsign.us/.

[9] M. Carey, E. Parris, and H. Lloyd-Thomas. "A comparison of features for speech,music discrimination,". In *Proc. Int. Conf. Acoust., Speech,Signal Process.,*, Phoenix, AZ, volume 1, pages 149–152, Mar. 1999.

[10] J. Pinquier, J.-L. Rouas, and R. A. Obrecht. "Robust speech/music classification in audio documents,". In *Proc. Int. Conf. Spoken Lang. Process.,*, Denver, CO, volume 3, pages 2005–2008, Sep. 2002.

[11] B. D. Barkana, B. Uzkent, and I. Saricicek. "Normal and Abnormal Non-Speech Ausio Event Detection Using MFCC and PR- Based Feature Sets". In *Adv.Mater. Res.,*, volume 601, pages 200–208, Dec. 2012.

[12] G. Tzanetakis and P. Cook. "Musical genre classification of audio signals". In *IEEE Transection on Speech and Audio Processing,*, volume 10, July. 2002.

[13] L. Ma, B. Milner, and D. Smith. "Acoustic Environment Classification". In *ACM Transactions on Language Processing (TSLP),*, volume 3, pages 1-22, July. 2006.

[14] B. Ghoraani and S. Krishnan. "Time-Frequency Matrix Feature Extraction and Classification of Environmental Audio Signals". In *IEEE Trans. Audio, Speech Lang.processing.,*, volume 19, pages 2197-2209, 2011.

[15] A. Sharma and S. Kaul, "Two-Stage Supervised Learning-Based Method to Detect screams and Cries in Urban Environments". volume 24, pages 290-299, 2016.

[16] R.V Sharan and T. J. Moir, "Robust audio survaillance using spectrogram image texture feature". pages 1956-1960, 2015.

[17] J. Dannis, H. D. Tran, and H. Li, "Spectrogram Image Feature for Sound Event Classification in Mismatched Conditions,". In *IEEE Signal Process. Lett.,*, volume 18, pages 130-133, 2011.

[18] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, "A Practical Guide to Support Vector Classification".

[19] David J. Wheeler, Roger M. Needham, "TEA, a Tiny Enryption Algorithm", Computer Laboratory, Cambridge University, England.

[20] Vorbis ogg audio format https://xiph.org/vorbis/.

[21] Jack G. Ganssle, "A guide to debouncing", The Ganssle Group.