|  |
| --- |
| **Emergency Response System** |
| CS 6235 Project Report |
|  |
| Emergency Response System (ERS) is a framework that provides fast and efficient real-time processing and dispatch of distress signals to relevant authorities via SMS. An Arduino-based wearable panic button and an Android service running in the background on a phone are used to demonstrate the use of ERS framework. |
|  |
| **Nataraj Mocherla & Sandeep Manchem** |
| **12/2/2013** |
|  |

Contents

[*Abstract* 3](#_Toc374110810)

[Motivation 4](#_Toc374110811)

[Objective 5](#_Toc374110812)

[Features of ERS 5](#_Toc374110813)

[Architecture 7](#_Toc374110814)

[Implementation 9](#_Toc374110815)

[Wearable Personal Security Bracelet 9](#_Toc374110816)

[*Arduino Lilypad* 9](#_Toc374110817)

[*Bluetooth module* 9](#_Toc374110818)

[Android Mobile Service 9](#_Toc374110819)

[Backend Server 10](#_Toc374110820)

[*Node.JS* 10](#_Toc374110821)

[*MongoDB* 10](#_Toc374110822)

[Web-based Visualization 10](#_Toc374110823)

[User Interface 10](#_Toc374110824)

[SMS Format 10](#_Toc374110825)

[Android Service 10](#_Toc374110826)

[Location & History Map 10](#_Toc374110827)

[Results 10](#_Toc374110828)

[Related Work 10](#_Toc374110829)

[Lessons Learned 11](#_Toc374110830)

[Future Work 11](#_Toc374110831)

[Conclusion 11](#_Toc374110832)

[References 11](#_Toc374110833)

# *Abstract*

Emergency Response System (ERS) is a framework that provides quick processing and dispatch of distress signals via SMS. We use Node.js, an event-driven, non-blocking I/O model perfectly suited for large-scale data intensive real-time applications that run across distributed devices, to implement the backend server that receives an emergency message via SMS, processes it and forwards the augmented message to relevant authorities who can then help the person who sent the distress signal. We have implemented one use-case of this framework wherein an emergency signal is sent from a wearable Arduino-based panic button which is connected to the user’s phone via Bluetooth, the lightweight Android service running on the phone in the background then quickly adds location information to the signal and sends an SMS to the backend server. The server processes the SMS and augments it with user information along with human readable location of the user and sends it the relevant Police Authorities. Subsequently, the Android service sends tracking information to the server which plots the location of the phone in real-time over a map which the Police can access and act accordingly.

# Motivation

Safety is everybody’s concern. It is often the case that when you need help, you may not have the time or it may not be feasible to pick up the phone and dial for help. In case of an emergency such as an impending robbery, a heart attack or a tumble down the stairs it is crucial to contact authorities as soon as possible. This line of thought led us to the notion of a panic button that when pressed, quickly transmits all the relevant information including the user’s location to appropriate authorities and/or family and friends, so that help can be dispatched immediately.



Figure 1 : Georgia Tech Emergency Posts

On Georgia Tech campus we find Emergency Alarms installed on specific locations throughout the campus as seen in the image above [Fig.1] that allow people to send an emergency message to Georgia Tech/Atlanta Police, with just a push of a button. However, these posts are not really useful when the student is not in the vicinity of one of them. To solve this problem we propose the idea of a **wearable panic button** to be worn all the time that can be pressed in the blink of an eye and send a distress signal. The idea of a wearable panic button is particularly appealing to those who want to conceal the panic button and send a distress signal without anybody realizing it, for example during an ongoing robbery. The panic button could be part of a bracelet, belt buckle, necktie, or simply kept in a pocket.

# Objective

We intend to create a fast and efficient Emergency Response System (ERS) to send distress signals to appropriate authorities with as much ease as possible. We develop a **wearable panic button** that when pressed will send an emergency signal to the user’s phone via Bluetooth, which in turn forwards it to a backend server along with the phone’s location information in the form of an SMS. The backend server will process the SMS and augment it with user information and send the distress signal with all relevant information to appropriate authorities (Police, Friends/Family etc).

We must also ensure that accidental triggering of the panic button can be corrected i.e., we should be able to cancel an Emergency Signal that was sent unintentionally. To achieve this we need to not only have a mechanism to send a Cancel Signal, but also inform the user that an Emergency Signal was sent so as to initiate a cancel procedure.

The distress signal (SMS) sent to the authorities should be easy to read and immediately actionable i.e., the SMS must contain all the required information to reach the location of the incident and also enable easy identification of the user (victim).

It is desirable to have constant contact between the user and the helper (Police/Friends/Family) once a distress signal has been sent so as to ensure that any updates in the situation can be transmitted quickly and course correction could take place if required.

From the point of view of the helper (Police), an appropriate interface must be provided to cross-check user information and also handle multiple emergencies occurring at the same time.

# Features of ERS

The Emergency Response System is primarily a framework for fast and efficient framework for receiving, processing and dispatching distress signals via SMS. We use the framework to implement an end-to-end solution using a Wearable Panic Button and an Android Service running on a phone. Our solution has the following components:

1. **Wearable Personal Security Bracelet** (PSB) [Fig.2] built using an Arduino Lilypad Microcontroller [7] with Bluetooth module [8].
2. **Android Mobile Service** which listens to signals from the PSB via Bluetooth.
3. **Backend servers** to process the incoming panic messages and routing them to the police (GATech or Atlanta police depending on location of crime scene).
4. **Web based visualization** of data for Police.



Figure 2 : Wearable Personal Security Bracelet

We have achieved each of our objectives by having the following features:

1. The wearable Personal Security Bracelet includes a **button** and an **LED light** that will begin blinking upon sending the emergency signal. The LED light acts as a feedback mechanism to the user to indicate that an emergency signal has been sent and can be cancelled if required.
2. A “**double press**” of the panic button will send an *Emergency Signal* and a “**prolonged press**” (press and hold) will send a *Cancel Signal*.
3. A window of 30 seconds is provided for the user to cancel an emergency signal, after which the backend servers will process the signal and forward the message to appropriate authorities.
4. Button press translates into a **Bluetooth Signal** that is transmitted to the user’s mobile which can be within **10 meters** from the panic button thereby removing the constraint of having the phone on person all the time.
5. The Android mobile service is responsible for **determining the location of the device.** This information is sent via SMS to the server in the form of latitude and longitude, which is then translated into an actual address that is easy to read and reach.
6. The Android Service continues to send **Tracking SMS** to the backend servers with location information to help the authorities track the mobile phone and approach it sooner.
7. The web based user interface for the Police enables them to easily visualize the location of incident and subsequent tracking of victim/assailant.

# Architecture

Our solution has the below Basic System Architecture [Fig.3].

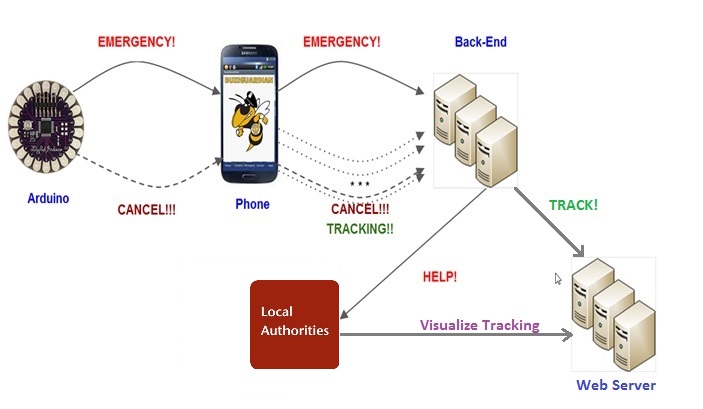


Figure 3 : Basic System Architecture

The Backend server’s primary purpose is to parse and process the SMS to extract the location information (latitude and longitude) embedded in it and construct a HELP! SMS with an address for the location and the user’s phone number, which is then sent to relevant authorities. It initiates a timeout of 30 seconds and listens for a cancel request, which if received will interrupt the timeout and cancel the emergency. After timeout, the HELP! SMS is constructed and sent to authorities and any cancel requests received at that time will be ignored. All SMSs are logged to DB to maintain a record of the user behavior and incidents. Architecture of the Backend Server is shown in [Fig.4]

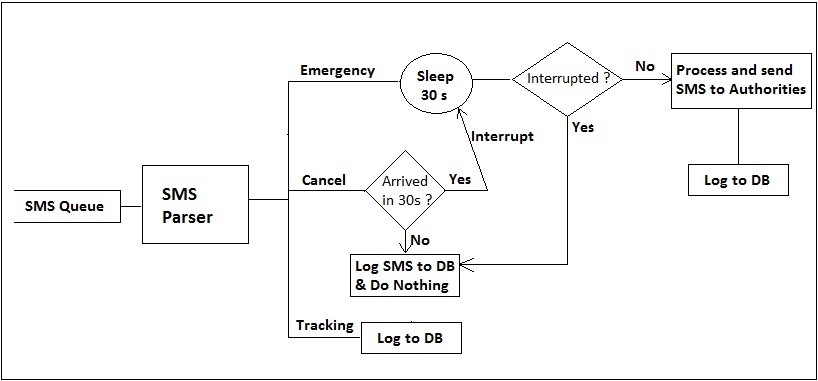


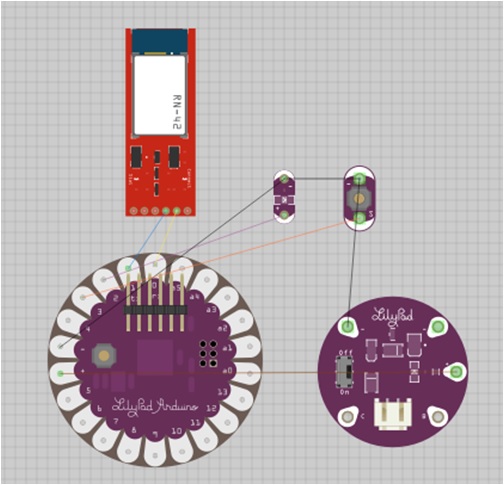
Figure 4 : Architecture of Backend Server

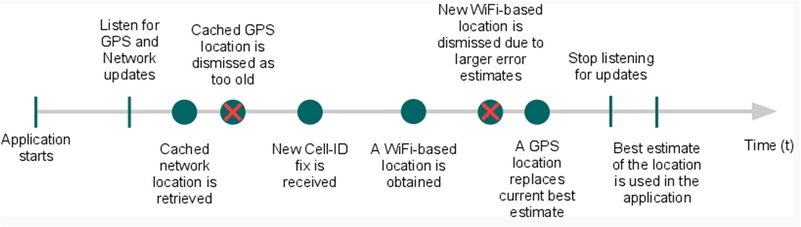
# Implementation

## Wearable Personal Security Bracelet

### *Arduino Lilypad*

### *Bluetooth module*



Android Mobile Service 

## Backend Server

### *Node.JS*

Node.js server – include “contacting correct authority”

### *MongoDB*

MongoDB Database. – different tables and ease and speed of MongoDB.

## Web-based Visualization

Node.js webserver.

# User Interface

## SMS Format

SMS format sent to server and sent to police.

## Android Service

Screen shots

## Location & History Map

Screen shots

# Results

End to end time

Server processing time

Server processing time when multiple signals sent simultaneously.

# Related Work

There are two main categories of solutions that are already available in the market in the area of emergency applications.

1. Emergency Mobile Applications
2. Panic Buttons

There are emergency mobile applications like the “In Case of Crisis” [1] and [2] which let the user set up specific lists of emergency contacts for specific scenarios and enables the user to send them emergency messages. The problem with this approach is the time it takes for someone to reach for the mobile phone which makes it useless in case of incidents like mugging.

There are systems like Alert 1 [3] which is a **PERS (Personal Emergency Response System)** designed for elderly people in case of medical emergencies. This system is quite expensive since it requires monthly plans. Also, the technology is designed for home use only since it requires to be connected to a base station. Other systems like the Geoskeeper Personal Cellular Security Communication Bracelet [4] solve some of the issues of Alert 1. The Geoskeeper tries to go beyond traditional PERS systems. Geoskeeper can be used anywhere and it doesn’t require a monthly service. It also has the advantage of being fully self-contained. But it is also quite expensive, priced at $299, and it is bulky and it doesn’t look fashionable either. Another interesting related technology is the Up [5] by Jawbone. Up is a lightweight bracelet for life tracking functionality that connects to iPhone smart phones. It tracks statistics such as number of steps taken during the day and which time the person fell asleep. This device looks very comfortable, fashionable and lightweight. It’s price is more reasonable than the previous technologies surveyed, $129.99, but the price is still not that affordable. This still a luxury item. It’s biggest downfall, however, is that it doesn’t provide any PERS functionality at all.

# Lessons Learned

# Future Work

# Conclusion

# References

[1] Georgia Tech – In Case of Crisis application. Retrieved 19 Sep 2013 from <https://play.google.com/store/apps/details?id=com.iba.incaseofaca>

[2] Emergency Panic Button for Android. Retrieved 19 Sep 2013 from <https://play.google.com/store/apps/details?id=com.incorporateapps.emergency&hl=en>

[3] Alert1 : Medical Alert Systems . Retrieved 18 Sep 2013 from <http://www.alert-1.com/>

[4] Geoskeeper Personal Cellular Security Communication Bracelet. Retrieved 18 Sep 2013 from <http://www.locationbasedgps.com/geoskeeper-personal-cellular-security-communication-bracelet/>

[5] Jawbone Wristband. Retrieved 19 Sep 2013 from <https://jawbone.com/up>

[6] Crime Alerts from Georgia Tech Police. Retrieved 20 Sep 2013 from <http://police.gatech.edu/crimeinfo/crimealerts/>

[7] Arduino. Retrieved 19 Sep 2013 from <http://www.arduino.cc/>

[8] Arduino Bluetooth Module. Retrieved 19 Sep 2013 from <http://arduino.cc/en/Guide/ArduinoBT>