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 a mobile Android service are used to demonstrate the use of ERS framework.

Nataraj Mocherla & Sandeep Manchem

12/2/2013

Contents

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.

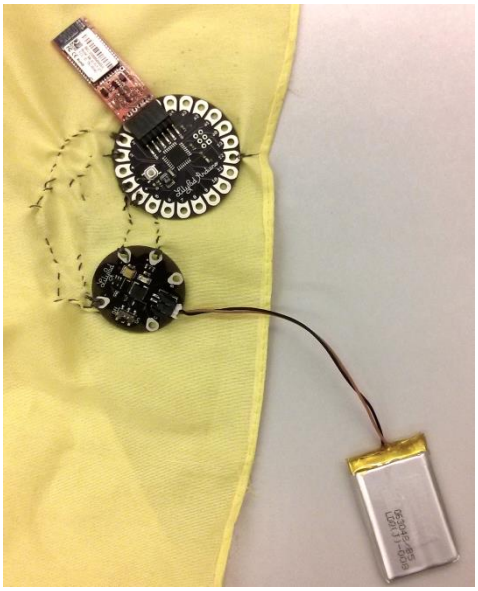


Figure 2 : Wearable Personal Security Bracelet

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].

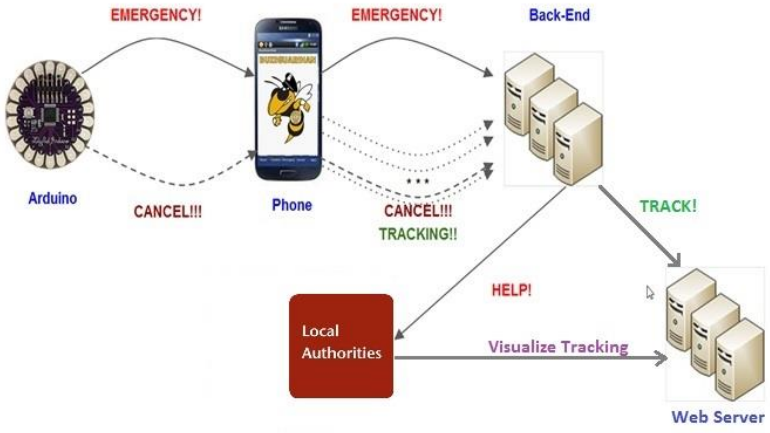


Figure 3 : Basic System Architecture

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]

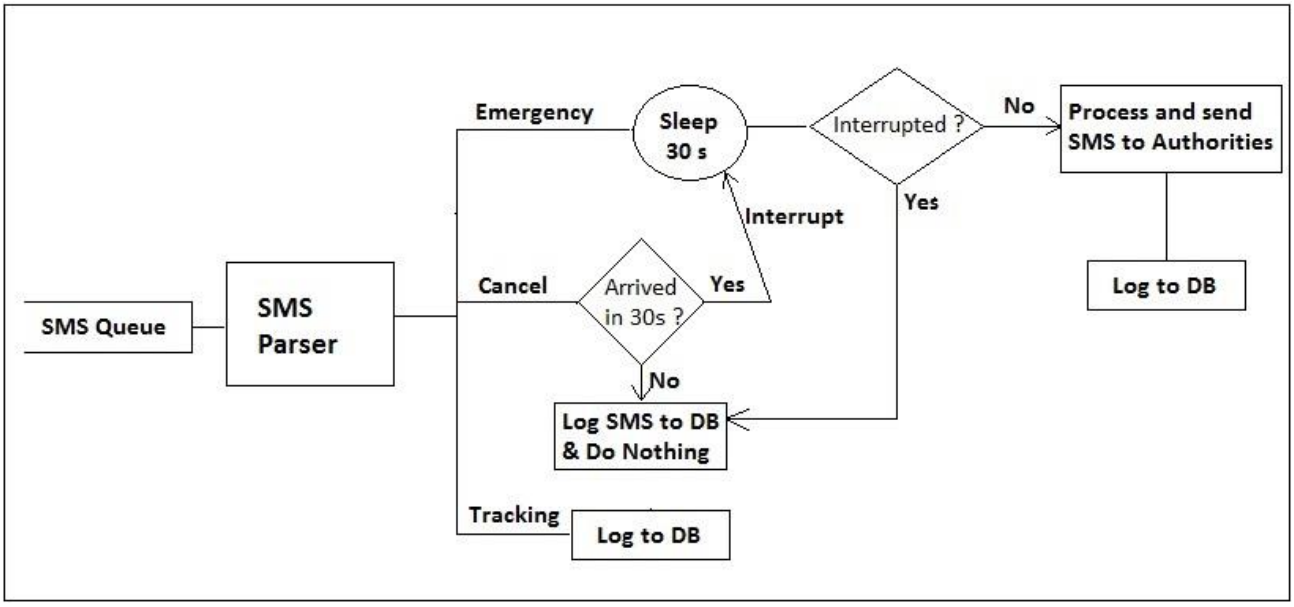


Figure 4: Architecture of Backend Server

Implementation

Wearable Personal Security Bracelet

The wearable personal security bracelet embraces the latest developments in the field of ***e-textiles*** by leveraging Lilypad Arduino microcontroller and Bluetooth LiPo Add-On with a rechargeable power supply. It is sewn onto a cloth using conductive threads, so that it can be easily tied around one's wrist. The circuit information is depicted in Fig.5

Lilypad Arduino

The LilyPad Arduino [7] is a microcontroller board designed for ***e-textiles*** and wearables. It can be sewn to fabric and similarly mounted power supplies, sensors and

actuators with conductive thread. The board is based on theATmega168V which is the low-power version of the ATmega168. The Lilypad is powered by a 3.7 volt Lithium Polymer battery that is plugged directly into the on-board *JST connector*.

Bluetooth module

We use Bluetooth LiPo Add-On in conjunction with Lilypad Arduino to pass data wirelessly to a phone. The Bluetooth HID connection is used to transmit data to the Bluetooth enabled phone, for processing emergency and cancel signals. It powered through a 2-pin JST connector.

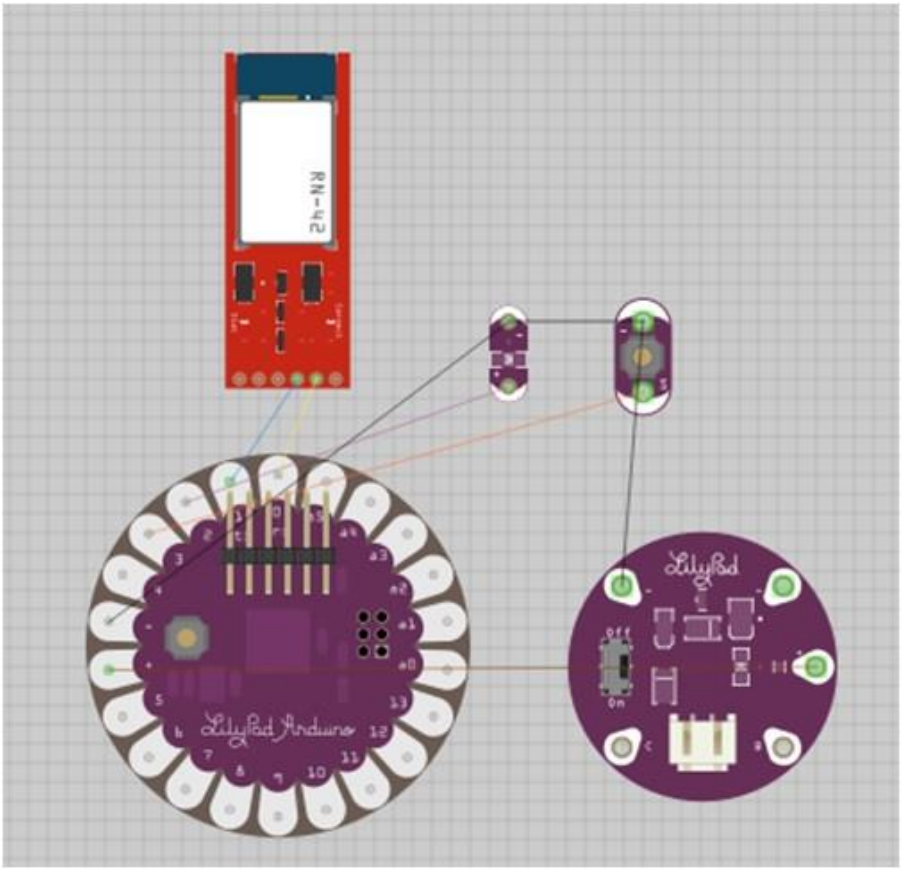


Figure 5: Lilypad Arduino and Bluetooth Module

Android Mobile Service

The mobile device that is paired with Lilypad Arduino's Bluetooth Add-On runs a service that constantly listens to Bluetooth messages. We have chosen Bluetooth as the means of communication primarily because of the simplicity of the protocol and the relatively short range distances we work with. The Android service has a user interface

through which we can start or stop the service. At this point, we have implemented this service on the Android platform and plan on an iPhone implementation in the future.

Upon receiving a communication from the personal security bracelet, the Android Services determines the current location of the device using Location Detection techniques that involve caching and best estimates. The location is recorded from various sources: GPS, Wi-Fi and Cellular Towers (Network Provider). Fig.6 shows how the location is detected on the Android Device. An SMS is constructed with the location information (Latitude and Longitude) of the device and the user's phone number, appending the nature of the signal (Emergency or Cancel) and sent to a phone number associated with the Backend Server.

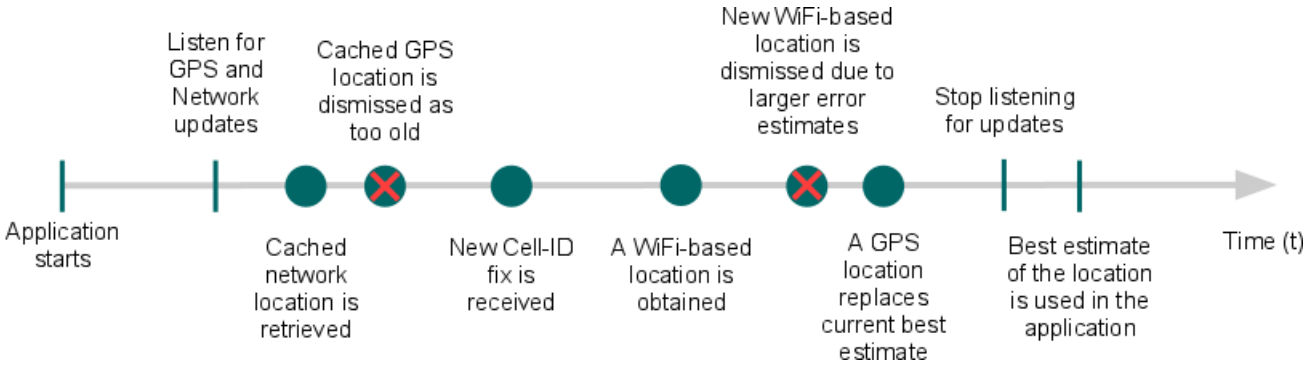


Figure 6: Location Detection on Android Device [12]

Upon sending an Emergency SMS to the Backend server, the Android Service sends Tracking SMS to the server with location information every minute, so that the current location of the user is known to the authorities.

Backend Server

The Backend Server is built using Node.js, an event driven non-blocking I/O platform that processes every SMS received and logs them to a database, MongoDB which is a highly available lightweight document database.

Node.JS

Node.js is a server side JavaScript execution engine written on top of Google Chrome's V8 JavaScript interpreter. Node.js utilizes JavaScript as its scripting language, and achieves high throughput via non-blocking I/O and a single-threaded event loop. We used Node.js to build the SMS processing engine and to host the web interface which renders out user interface. Node.js also provides easy integration with Google Voice API and MongoDB which were used in our backend.

The server determines the area from where the Emergency SMS originated by doing earth distance calculation and based on that determines the appropriate authorities to contact. For example, if an Emergency SMS is sent from Georgia Tech Campus, it will send the HELP! SMS to Georgia Tech Police Department (GT PD), otherwise it will contact Atlanta Police Department.

MongoDB

MongoDB is a cross-platform documented-oriented database that uses JSON-like documents with dynamic schemas as opposed to traditional relational database tables– different tables and ease and speed of MongoDB, which makes it ideal for our application. It is not only easy to use but also really fast in terms of performance. We use the database for storing user information, SMS Logs and usage statistics. The Data stored can be used to evaluate the performance of the system and identify patterns in occurrence of incidents and user behavior in terms of false alarms. The Tracking SMS logs are queried by Web-Server to map out the movement of Android device during an emergency.

Web-based Visualization

Node.js is used to host the web interface which renders out user interface to visualize location of the Android device on a map to track its movements during an emergency. We use Google Maps API to render the maps and MongoDB is queried to get the tracking information.

The web-server also hosts a registration and login pages for users and authorities. The user information along with a recent photograph is stored in the MongoDB Database, which is pulled up and displayed right next to the map during an emergency. This enables the authorities to obtain all the relevant information (current location and user information) on a single with just one click.

User Interface

SMS Format

SMS sent to the Backend Server:

=== EMERGENCY latitude= 33.77682 longitude= -84.39607 timestamp= 2013-04-25 18:48:28.05

=== CANCEL latitude= 33.77682 longitude= -84.39607 timestamp= 2013-04-25 18:48:28.05

=== TRACKING latitude= 33.77682 longitude= -84.39607 timestamp= 2013-04-25 18:48:28.05

SMS sent to the Authorities:

+14049695785 needs HELP at 704 Cherry Street Northwest, Georgia Institute of Technology, Atlanta, GA 30332, USA

Android Service

The Android Service needs Bluetooth to be ON. It doesn't require the GPS and Wi-Fi to be ON, but with either of them ON the accuracy of location will increase manifold. The Service needs to be simply *started* and it will continue to run in the background till it's *stopped*. Fig.7 shows a screenshot of the Android Service.

Figure 6: Android Service

Location Visualization

The homepage shows the most recent occurrence of an emergency and the corresponding user's information is displayed to the right of the map which shows the location of the incident and the current location. When multiple emergencies occur at the same time, Authorities can select each user from the dropdown menu which will display the user's information and update the map to show the user's location. Fig.7 shows a screenshot.

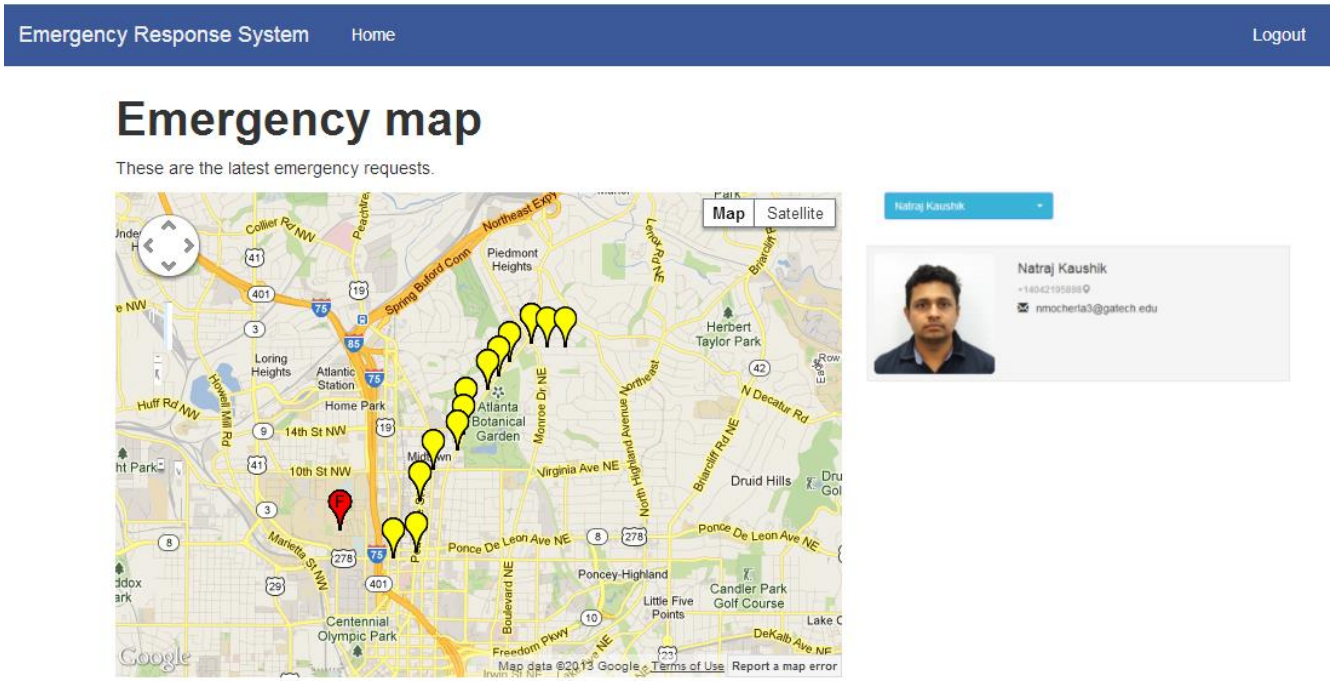
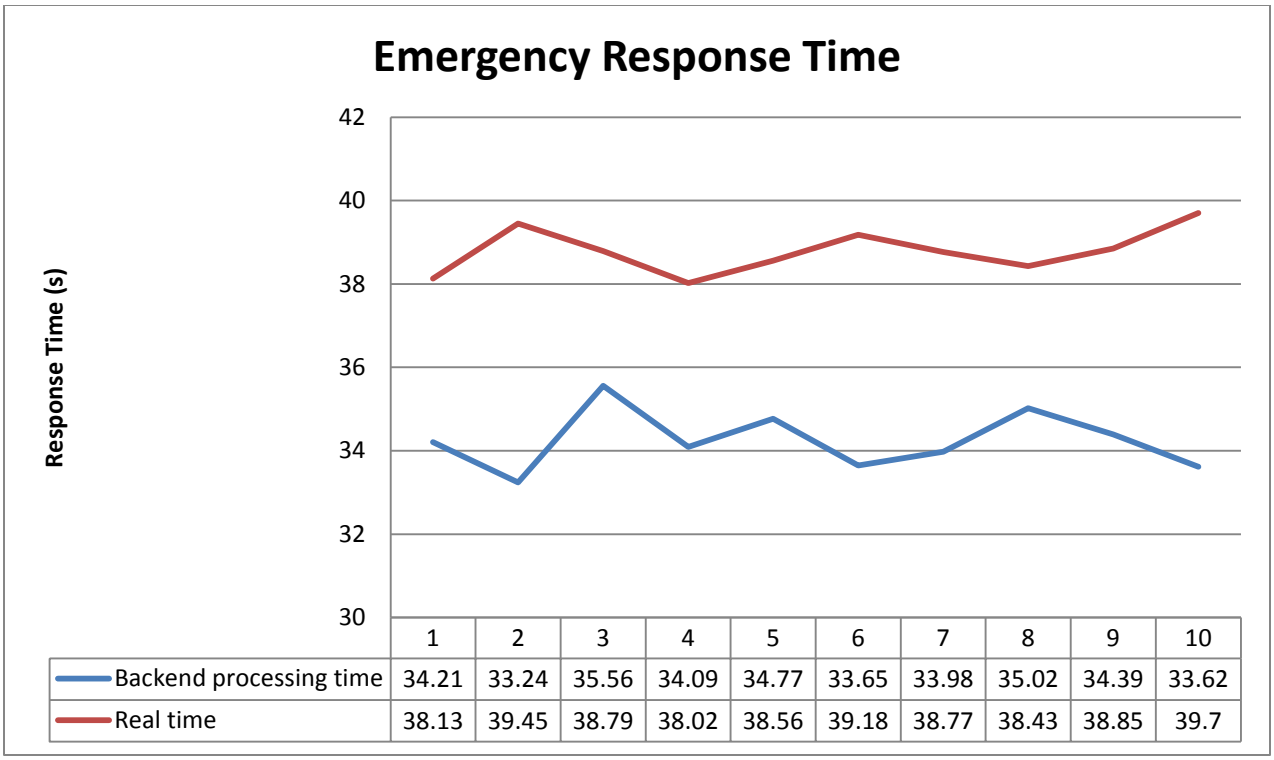


Figure 7: Location Visualization

Results

We have successfully built an Emergency Response System that transmits distress signals to the appropriate authorities so that help can be dispatched at the earliest. The user needs to do just push the panic button and help will be dispatched within the minute. We have gathered statistics to determine time taken for the Emergency signal to travel from Lilypad Arduino to the Authorities. We have also analyzed the time taken for processing the SMS at the server after receiving from the Android Service. The mean processing time is 3.33 seconds and the mean end-to-end time taken is 38.78 seconds, which includes 30 seconds of wait time for the cancel signal.



	1	2	3	4	5	6	7	8	9	10
Backend processing time	34.21	33.24	35.56	34.09	34.77	33.65	33.98	35.02	34.39	33.62
Real time	38.13	39.45	38.79	38.02	38.56	39.18	38.77	38.43	38.85	39.7

Figure 8: Response Time of ERS

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. Its price is more reasonable than the previous technologies surveyed, $129.99, but the price is still not that affordable. This is still a luxury item. It's biggest downfall, however, is that it doesn't provide any PERS functionality at all.

Lessons Learned

Bluetooth modules in some mobile phone sets are **not robust**. This introduced a certain risk of failure due to non-pairing of Lilypad Arduino with Android Device. We rely on the cellular networks to deliver the SMS in real-time and which becomes a point of delay if not failure.

Future Work

We believe that our platform can be used for much more than as an Emergency Response System. We have with us, a working implementation of a microcontroller which communicates with a paired Smartphone and this can be used as a base to venture into areas like Heart Attack Detection and Personal Health Monitoring. We also plan to explore more communication technologies like Radio Frequency Identification to enable longer range communication between the microcontroller and the Smartphone.

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

[9] e-textiles http://www.ccm.ece.vt.edu/etextiles/

[10] Juels, A. (2006). RFID security and privacy: A research survey. Selected Areas in Communications, IEEE Journal on, 24(2), 381-394.

[11] Permanent RFID garment tracking system. Quartararo Jr, P. J. (1998). *U.S. Patent No. 5,785,181*. Washington, DC: U.S. Patent and Trademark Office.

[12] Android Location Strategies. http://developer.android.com/guide/topics/location/strategies.html