Final documents submission

# Title of project:

Department of Computer Systems Engineering

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

TSHWANE UNIVERSITY OF TECHNOLOGY

## COURESE DETAILS：

COURSE: COMPUTER SYSTEMS ENGINEERING

SUBJECT CODE: PJD301B

SUBMISSION DATE: 01 September 2020

LECTURER: Dr Chunling Du and MR. M.N.Muwanguzi

## STUDENT DETAILS:

SURNAME & INITIALS: MAHLUNGULU A J

STUDENT NUMBER: 216580095

EMAIL ADDRESS: mahlungulu7anelisiwe@gmail.com

CONTACT NUMBERS: 078 058 0913/ 063 298 0441

## TABLE OF CONTENTS

# PROPOSAL

## INTRODUCTION

With increasing fatal crashes, robbery, and many other emergency set-ups and conditions being reported at late hour and that being caused by sending or providing wrong location of the emergency incident. Emergency preparedness can be defined as pre-impact activities that establish a state of readiness to response to extreme events that could affect the community's good health and well-being. The practice of emergency response planning varies considerably among communities. Citizen-Watch ensures adequate emergency preparedness, and it has a capability of performing its basic emergency response functions. Emergency response program should look at both the big picture and the small details to determine the risks and hazards potential to your operations. A plan should be developed using available resources, trained personnel, and industry best practices keeping in mind the safety of people, the environment, property, and business continuity.

## PROJECT OBJECTIVES

Since emergencies will occur, like car accidents. Preplanning is essential. At the onset of an emergency, a lot of decisions need to be made in a short period of time. Time and circumstance can mean the normal chain of command is not accessible. Added to that, stress and being scared and terrified by the incident can result in poor judgment, reporting wrong emergency location and extensive losses.

## PROJECT DESCRIPTION

Citizen Watch is an embedded application that consist of both software and hardware. It is an application designed from the initiative of safety and an experience I once encountered. As many says and know that most of us females, we very bad when it comes to giving directions, it becomes much worse when we are under pressure or frightened. Am originally from the Eastern Cape which is -- kilometers from Pretoria. Two years back on my way home{Eastern Cape} I came across a terrifying car accident on the R57 road most people lost their lives because ambulances took so much time to arrive and that being caused by being given wrong directions of the incident. If they came in time, they could have saved most lives, that’s how the idea of Citizen Watch came about.

It is every Citizen right to be in good health and well-being, so if any citizen comes across any emergency set-up or condition, they should be able to report that incident using their correct location co-ordinates, simply by using Citizen Watch application. For testing purposes, I will be using a Bluetooth module for communication between the mobile application and the Arduino hardware application, which is a Hospital Department represented by an LCD for displaying location co-ordinates of the incident, a 4 digit 7- segment that will display the word “HELP” for the Hospital Department Administrator to know there’s a citizen seeking help at the location co-ordinates displayed on the LCD, and 2 LED’s: Red & blue. The red LED will blink to catch the Administrators attention when an emergency is being reported. The blue LED is going to switch on, and the red LED switched off once the hospital admin presses the push button to send feedback to the citizen which was reporting that help is coming right up in a form of a message.

## Android application: Hardware and Software

**Bluetooth Module:**

- Used for connection between the mobile application and the hardware circuit.

**Vibration Sensor:**

- If the vibration sensor senses vibration it sends the value “1” as reference that there’s an emergency incident that has occurred.

**Red LED’s:**

- They will blink when an emergency is being reported by the user.

**Blue LED’s:**

- Once the button is pressed by the department admin, they will be on, while sending emergency confirmation message to the user that help it’s on its way.

**7-segments:**

- Display “help” when an emergency is being reported, and “- - - - “when sending an emergency confirmation message.

**LCD:**

- It will display the current location co-ordinates {latitude and longitude} of the emergency incident since the integrated circuit will be executing in real time.

## PROJECT TECHNOLOGY

Citizen Watch system is going to be embedded software, which means it will compose of both hardware and software. The components and platforms that will be used when implementing emergency response system will be as follows:

### Hardware

- Arduino Mega
- Vibration Sensor
- Red, Orange and Blue LED's LCD (Liquid Crystal Display) Jumper wires
- 7-segments 4 digits
- Buttons
- Vero Board
- Resistors
- Soldering Ion
- Soldering Wire
- Bluetooth Module

### Platforms

- Arduino
- MIT App Inventor

### Programming languages

- C++
- Java

### Software

- Android application

### TIMETABLE

Provide detailed information on the expected timetable for the project. Break the project into phases, and provide a schedule for each phase.

| Activity | Description of work | Start and end dates |
|---|---|---|
| Enclosure | Build and drafting the structure of my system | 01-03 August 2020 |

| | | |
|---|---|---|
| Wiring | Do connections of the components into the Arduino board and the structure | 04-08 August 2020 |
| Programming | Building the user interface of the mobile app | 09-16 August 2020 |
| Integration | Interfacing the hardware with the software | 17-21 August 2020 |
| Integration | Connection to the database | 22-25 August 2020 |
| Deployment | Deploying and testing | 26-31 August 2020 |

**BUDGET**

State the proposed costs and budget of the project. Also include information on how you intend to manage the budget.

| Items | Supplier | Quantity | Unit price | Total |
|---|---|---|---|---|
| Arduino mega | Communica | 1 | R330.79 | R430.79 |
| LCD (Liquid Crystal Display) | Communica | 1 | R173.51 | R173.51 |
| Jumper wires | Communica | 3 | R33.13 | R99.83 |
| 7-segments 4-digits | Communica | 1 | R47.80 | R47.80 |
| Buttons | Communica | 1 | R15.00 | R15.00 |
| Vero board | Communica | 1 | R60.63 | R60.63 |
| Resistors | Communica | 15 | R146.00 | R146.00 |
| LED | Communica | 2 | R2.00 | R4.00 |
| Bluetooth Module | Communica | 1 | R110.00 | R110.00 |
| Vibration Sensor | Communica | 1 | R70.00 | R70.00 |
|  |  |  | Total budget | R1 157.56 |

## CONCLUSION

Citizen Watch Mobile app is designed for every citizen, it doesn't have age restriction, and if you have a smart phone you can use it to report. Once it becomes successful in the Hospital Department, it's going to be implemented to all 911 departments & will be trained on how to interact with incoming emergency reports for assistance in emergency response.

## REFERENCE

http://docwiki.embarcadero.com/RADStudio/Berlin/en/Running_Your_Android_Application_on_an_Android_Device#Manually_Uninstall_an_Android_Application

https://en.wikipedia.org/wiki/South_African_Police_Service

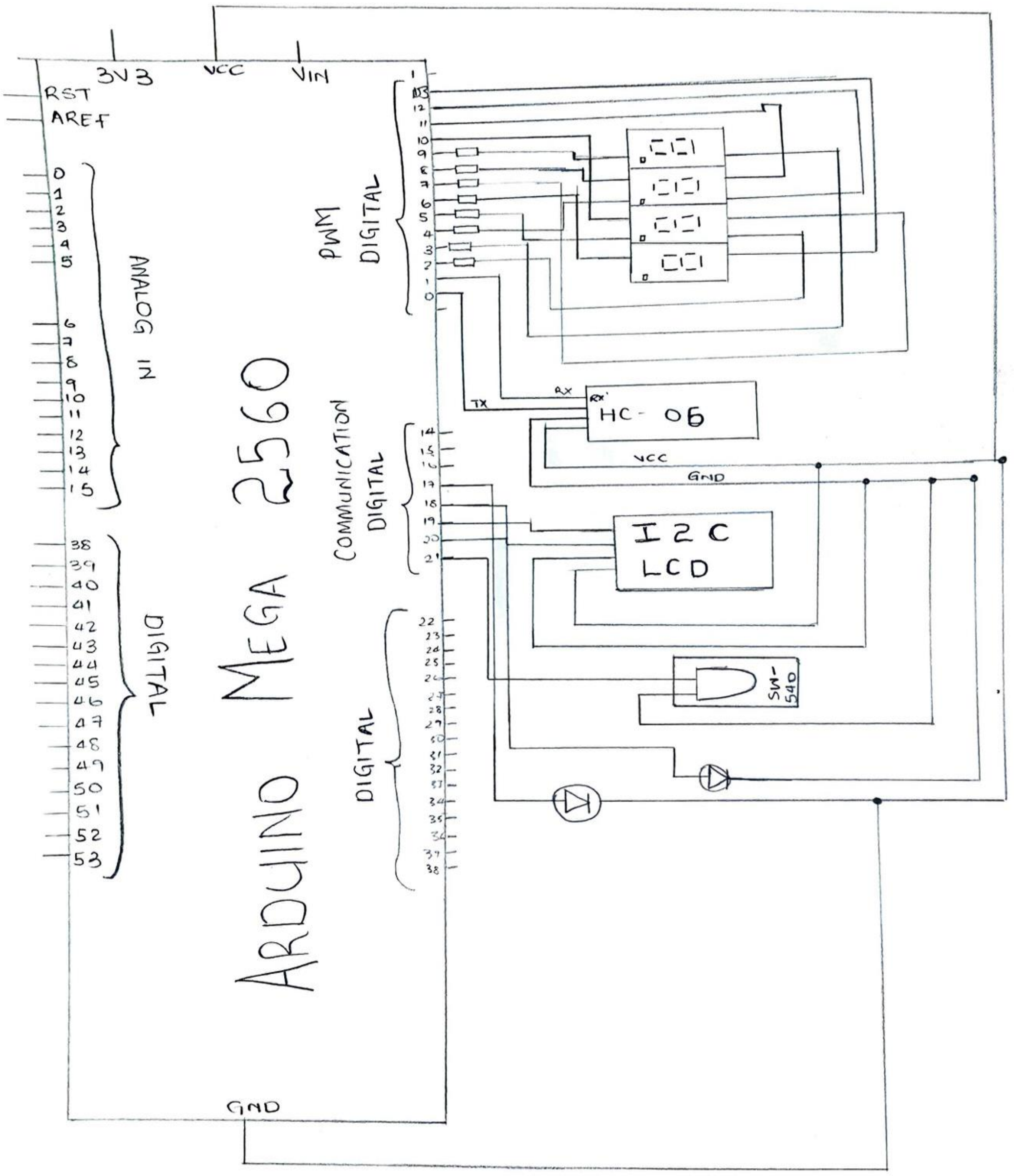
http://docwiki.embarcadero.com/RADStudio/Berlin/en/SDK_Manager

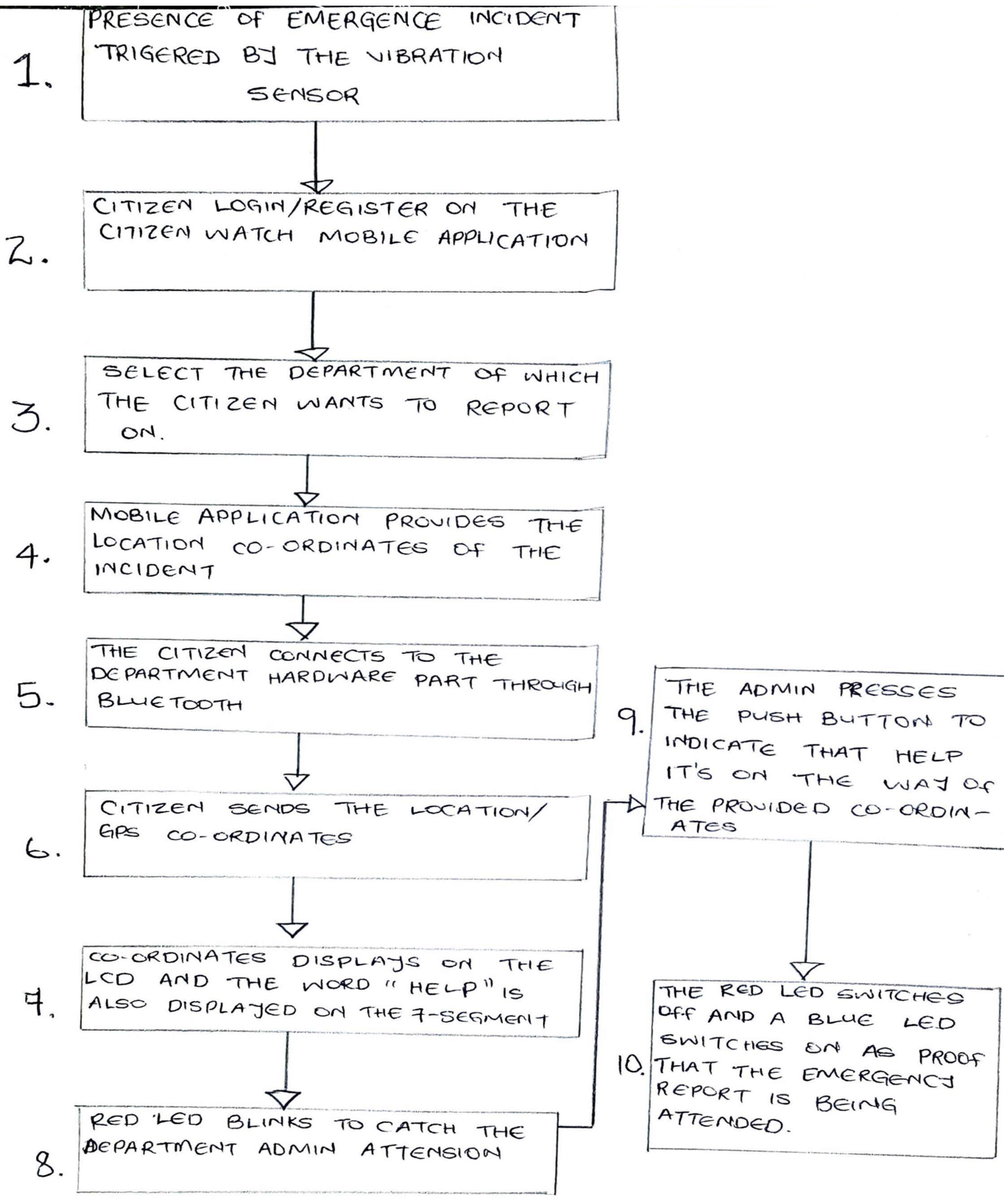
https://en.wikipedia.org/wiki/South_African_Health_Department

https://en.wikipedia.org/wiki/South_African_Fire_Fighting_Department

https://www.youtube.com/watch?v=2pc2ehkSEAI

# PROCESS FLOW AND DESIGN DOCUMENT


3V3
VCC
VIN
RST
AREF
ANALOG IN
DIGITAL
ARDUINO MEGA 2560
PWM
DIGITAL
COMMUNICATION
DIGITAL
DIGITAL
HC-06
RX
TX
VCC
GND
I2C
LCD
SW-540
GND



1. PRESENCE OF EMERGENCE INCIDENT TRIGERED BY THE VIBRATION SENSOR

2. CITIZEN LOGIN/REGISTER ON THE CITIZEN WATCH MOBILE APPLICATION

3. SELECT THE DEPARTMENT OF WHICH THE CITIZEN WANTS TO REPORT ON.

4. MOBILE APPLICATION PROVIDES THE LOCATION CO-ORDINATES OF THE INCIDENT

5. THE CITIZEN CONNECTS TO THE DEPARTMENT HARDWARE PART THROUGH BLUETOOTH

6. CITIZEN SENDS THE LOCATION/ GPS CO-ORDINATES

7. CO-ORDINATES DISPLAYS ON THE LCD AND THE WORD "HELP" IS ALSO DISPLAYED ON THE 7-SEGMENT

8. RED LED BLINKS TO CATCH THE DEPARTMENT ADMIN ATTENSION

9. THE ADMIN PRESSES THE PUSH BUTTON TO INDICATE THAT HELP IT'S ON THE WAY OF THE PROVIDED CO-ORDIN- ATES

10. THE RED LED SWITCHES OFF AND A BLUE LED SWITCHES ON AS PROOF THAT THE EMERGENCY REPORT IS BEING ATTENDED.

# DESIGN DOCUMENT

**Introduction (1/2 page)**

Provide an overview of the entire document:

- Describe the purpose of this document
- Describe the scope of this document
- Describe this document's intended audience

With increasing fatal crashes, robbery, and many other emergency set-ups and conditions being reported at late hour and that being caused by sending or providing wrong location of the emergency incident. Emergency preparedness can be defined as pre-impact activities that establish a state of readiness to response to extreme events that could affect the community's good health and well-being. The practice of emergency response planning varies considerably among communities. Citizen-Watch ensures adequate emergency preparedness, and it has a capability of performing its basic emergency response functions. Emergency response program should look at both the big picture and the small details to determine the risks and hazards potential to your operations. A plan should be developed using available resources, trained personnel, and industry best practices keeping in mind the safety of people, the environment, property, and business continuity.

Since emergencies will occur, like car accidents. Preplanning is essential. At the onset of an emergency, a lot of decisions need to be made in a short period of time. Time and circumstance can mean the normal chain of command is not accessible. Added to that, stress and being scared and terrified by the incident can result in poor judgment, reporting wrong emergency location and extensive losses.

**Systems overview (1/2 page)**

Provide a general description of the software system and hardware system, including its functionality and matters related to the overall system and its design

Citizen Watch is an embedded application that consist of both software and hardware. It is an application designed from the initiative of safety and an experience I once encountered. As many says and know that most of us females, we very bad when it comes to giving directions, it becomes much worse when we are under pressure or frightened. Am originally from the Eastern Cape which is -- kilometers from Pretoria. Two years back on my way home{Eastern Cape} I came across a terrifying car accident on the R57 road most people lost their lives because ambulances took so much time to arrive and that being caused by being given wrong directions of the incident. If they came in time, they could have saved most lives, that’s how the idea of Citizen Watch came about.

It is every Citizen right to be in good health and well-being, so if any citizen comes across any emergency set-up or condition, they should be able to report that incident using their correct location co-ordinates, simply by using Citizen Watch application. For testing purposes, I will be using a Bluetooth module for communication between the mobile application and the Arduino hardware application, which is a Hospital Department represented by an LCD for displaying location co-ordinates of the incident, a 4 digit 7- segment that will display the word “HELP” for the Hospital Department Administrator to know there’s a citizen seeking help at the location co-ordinates displayed on the LCD, and 2 LED’s: Red & blue. The red LED will blink to catch the Administrators attention when an emergency is being reported. The blue LED is going to switch on, and the red LED switched off once the hospital admin presses the push button to send feedback to the citizen which was reporting that help is coming right up in a form of a message.

**Design considerations (1/2 page)**

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

Power Consumption

Embedded systems software developers must often create effective solutions to processing challenges while dealing with system size, weight and power constraints. It can be a challenge to reduce power consumption when system resources are limited, and processing demands are high. Oftentimes, the use of multicore architectures can help meet power consumption challenges, due to their various energy saving mechanisms. Additionally, the ARM architecture deserves some definite consideration, as low power consumption is their focus.

Memory Allocation

Memory allocation is an important factor in providing strong performance and reliability for embedded systems. Memory storage, organization, and data sharing options must be reviewed and even redesigned to achieve new solutions for types of embedded system software. Software designers may employ measures like two-phase memory, which allocates memory to large and long-term objects first, so they are guaranteed the required memory, then allocates memory to smaller objects in the second phase to reduce system memory failures.

Semaphores

As the name implies, semaphores are hardware/software flags. They indicate the status of common resources inside a multitasking system. Semaphores
can be very helpful in ensuring effective and efficient data processing in industrial embedded systems. These design tools can synchronize the execution of multiple tasks or provide direct access to shared resources within a system. Semaphores can even lock some resources for further analysis by other built-in processes.

Connectivity and Security

Security is always a key consideration. Doubly so with many embedded systems developers now integrating TCP/IP into their designs to achieve connectivity. System designers must always incorporate stronger threat detection and response mechanisms to ensure the integrity of systems data. Designing such software is a continuous process as threats change and increase all the time.

Compiler

A developer’s choice of compiler will affect what families of microprocessors and microcontrollers they can use, as well as which real-time operating system. Due to the challenges and restrictions of developing software for embedded systems, developers must use cross-compilers. They need to consider standard libraries and the compiler’s startup code. While Java is ever popular, its memory management performance requires a larger system to operate effectively and so many developers choose C++ to compile their designs.

As embedded systems become an ever-more crucial and complex tool in industrial automation and processing, developers never stop creating new software options to help meet evolving performance targets. There are many permutations and the whole consideration can easily become confusing, even overwhelming, for management and IT personnel. This is when it is best to call upon the skilled assistance of consultants specializing in embedded systems.

**Assumptions and dependencies (1/2 page)**

Describe any assumptions or dependencies regarding the software and hardware and their use. These may concern such issues as related software and hardware, end user characteristics etc.

- **Related software and hardware**

Citizen Watch system is going to be embedded software, which means it will compose of both hardware and software. The components and platforms that will be used when implementing emergency response system will be as follows:

Hardware

- ϖ Arduino Uno
- ϖ Red and Blue LED's
- ϖ LCD (Liquid Crystal Display)
- ϖ Jumper wires
- ϖ 7-segments 4 digits
- ϖ Button
- ϖVibration Sensor
- ϖ Vero board
- ϖ Resistors

Platforms

- Arduino

Programming languages

- o C++/C
- o Java

Software

- Android application

- **End-user characteristics**

Citizen Watch Mobile app is designed for every citizen, it doesn't have age restriction, as long as you have a smart phone you can use it to report. All 911 departments will be trained on how to interact with incoming emergency reports.

- **General Constraints**

Getting false reports alert, can cause conflict and waste of 911 resources. To eliminate that if a user sends false emergency alert, that user will be blocked and deleted from the database of the app.

On end-user environment, the Arduino operating temperature will be safe in weather that can reach -26 up to 40 degrees Celsius. Which means the 911 department admins should make sure that where the system microcontroller should be installed there’s a proportional temperature control to keep the temperature in the required condition.

### Goals and guidelines (1/2 page)

Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system's software. Such goals might be:

- The KISS principle ("Keep it simple stupid!")
- Emphasis on speed versus memory use
- Working, looking, or "feeling" like an existing product

For each such goal or guideline, unless it is implicitly obvious, describe the reason for its desirability. Feel free to state and describe each goal in its own sub-subsection if you wish.

o Do not rely on usability guidelines-am always testing with users.
o I base UI design on user’s tasks as expressed in use case.
o I always ensure the sequence of actions to complete a task correctly are simple as possible.
o I make sure the user always knows what to do, and what should be done next, and what will happen when he or she does it.
o The system provide good feedback, including effective error messages and warnings
o The system have back and exit buttons for assuring that the user can always get out, go back or undo an action.
o The system response time is adequate
o The labels and other encoding techniques used are understandable.
o The UI’s appearance is neat and uncluttered.
o The system is designed and suitable to be used by different groups of users. o I have ensured that users can easily and quickly access relevant and easy-to-understand help about anything they are trying to do.
o Once the users learn how to use one application, it is a big advantage to them if other applications and dialogs work the same way.

**Development Methods (1/2 page)**

Briefly describe the method or approach used for this software design. If one or more formal/published methods were adopted or adapted, then include a reference to a more detailed description of these methods. If several methods were seriously considered, then each such method should be mentioned, along with a brief explanation of why all or part of it was used or not used.

Agile development methodology
I use the agile development methodology to minimize risk (such as bugs, cost overruns, and changing requirements) when adding new functionality. In all agile methods, I develop the software in iterations that contain mini increments of the new functionality. There are many different forms of the agile development method, including scrum, crystal, extreme programming (XP), and feature-driven development (FDD).
Agile development methodology

Pros: The primary benefit of agile software development is that it allows software to be released in iterations. Iterative releases improve efficiency by allowing me to find and fix defects and align expectation early on. They also allow users to realize software benefits earlier, with frequent incremental improvements.

Cons: Agile development methods rely on real-time communication, so new users often lack the documentation they need to get up to speed. They require a huge time commitment from users and are labor intensive because developers must fully complete each feature within each iteration for user approval.

Agile development methods are like rapid application development (see below) and can be inefficient in large organizations. Programmers, managers, and organizations accustomed to the waterfall method (see below) may have difficulty adjusting to an agile SDLC. So, a hybrid approach often works well for them.

Get the Agile Security Manifesto

DevOps deployment methodology DevOps is not just a development methodology but also a set of practices that supports an organizational culture. DevOps deployment centers on organizational change that enhances collaboration between the departments responsible for different segments of the development life cycle, such as development, quality assurance, and operations. DevOps deployment methodology

Pros: DevOps is focused on improving time to market, lowering the failure rate of new releases, shortening the lead time between fixes, and minimizing disruption while maximizing reliability. To achieve this, DevOps organizations aim to automate continuous deployment to ensure everything happens smoothly and reliably. Companies that use DevOps methods benefit by significantly reducing time to market and improving customer satisfaction, product quality, and employee productivity and efficiency.

Cons: Even considering its benefits, there are a few drawbacks to DevOps: Some customers don't want continuous updates to their systems. Some industries have regulations that require extensive testing before a project can move to the operations phase. If different departments use different environments, undetected issues can slip into production. Some quality attributes require human interaction, which slows down the delivery pipeline.

**Architectural Strategies (1/2 page)**

The programming language that I've used when building the system is C++. I have used C++ because of its portability, object-oriented, multi-paradigm, low-level-manipulation, Error detection and recovery, Memory management policies, Communication mechanisms, Concurrency and synchronization, Distributed data or control over a network, generalized approaches to control, Concurrency and synchronization.

1. Portability C++ offers the feature of portability or platform independence which allows the user to run the same program on different operating systems or interfaces at ease.

2. Object-oriented One of the biggest advantages of C++ is the feature of object-oriented programming which includes concepts like classes, inheritance, polymorphism, data abstraction, and encapsulation that allow code reusability and makes a program even more reliable.

Not only this, it helps us deal with real-world problems by treating data as an object. C lacked this feature and hence it was created, proving to be of great significance.

This feature gave birth to numerous job prospects and technologies. It is fascinating to note that C++ was created by combining features not only from C but Simulate 67, the first object-oriented programming language.

3. Multi-paradigm C++ is a multi-paradigm programming language. The term "Paradigm" refers to the style of programming. It includes

logic, structure, and procedure of the program. Generic, imperative, and object-oriented are three paradigms of C++.

Let us now try to understand what generic programming means. Generic programming refers to the use of a single idea to serve several purposes. Imperative programming, on the other hand, refers to the use of statements that change a program's state.

4. Low-level Manipulation Since C++ is closely associated with C, which is a procedural language closely related to the machine language, C++ allows low-level manipulation of data at a certain level. Embedded systems and compiler are created with the help of C++.

5. Memory Management C++ gives the programmer the provision of total control over memory management. This can be considered both as an asset and a liability as this increases the responsibility of the user to manage memory rather than it being managed by the Garbage collector. This concept is implemented with the help of DMA (Dynamic memory allocation) using pointers.

6. Large Community Support C++ has a large community that supports it by providing online courses and lectures, both paid and unpaid. Statistically speaking, C++ is the 6th most used and followed tag on Stack Overflow and GitHub.

7. Compatibility with C C++ is pretty much compatible with C. Virtually, every error-free C program is a valid C++ program. Depending on the compiler used, every program of C++ can run on a file with .cpp extension. Page 10 of 15

8. Scalability Scalability refers to the ability of a program to scale. It means that the C++ program can run on a small scale as well as a large scale of data. We can also build applications that are resource intensive.

9. Use of Pointers Pointers in C/C++ are a relatively difficult concept to grasp and it consumes a lot of memory. Misuse of pointers like wild pointers may cause the system to crash or behave anomalously.

10. Security Issue Although object-oriented programming offers a lot of security to the data being handled as compared to other programming languages that are not object-oriented, like C, certain security issues still exist due to the availability of friend functions, global variables and, pointers.

11. Absence of Garbage Collector As discussed earlier, C++ gives the user complete control of managing the computer memory using DMA. C++ lacks the feature of a garbage collector to automatically filter out unnecessary data.

12. Absence of Built-in Thread C++ does not support any built-in threads. Threads is a relatively new concept in C++ which wasn't initially there. Now, C++ is capable of supporting lambda functions.

## System Architecture(1page)

This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves (there is a subsequent section for detailed component descriptions). The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.

At the top-most level, describe the major responsibilities that the software must undertake and the various roles that the system (or portions of the system) must play. Describe how the system was broken down into its components/subsystems (identifying each top-level component/subsystem and the roles/responsibilities assigned to it). Describe how the higher-level components collaborate with each other in order to achieve the required results. Don't forget to provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). Feel free to make use of design patterns, either in describing parts of the architecture (in pattern format), or for referring to elements of the architecture that employ them.

If there are **any diagrams**, **models**, **flowcharts**, **documented scenarios** or use-cases of the system behavior and/or structure, they may be included here (unless you feel they are complex enough to merit being placed in the Detailed System Design section). Diagrams that describe a particular component or subsystem should be included within the particular subsection that describes that component or subsystem.

*Note:*

This section (and its subsections) really applies only to newly developed (or yet-to-be developed) portions of the system. If there are parts of the system that already existed before this development effort began, then you only need to describe the pre-existing parts that the new parts of the system depend upon, and only in enough detail sufficient to describe the relationships and interactions between the old parts and the new parts. Pre-existing parts that are modified or enhanced need to be described only to the extent that is necessary for the reader to gain a sufficient understanding of the nature of the changes that were made.

## Subsystem Architecture

If a particular component is one which merits a more detailed discussion than what was presented in the System Architecture section, provide that more detailed discussion in a subsection of the System Architecture section (or it may even be more appropriate to describe the component in its own design document). If necessary, describe how the component was further divided into subcomponents, and the relationships and interactions between the subcomponents (similar to what was done for top-level components in the System Architecture section).

If any subcomponents are also deemed to merit further discussion, then describe them in a separate subsection of this section (and in a similar fashion). Proceed to go into as many levels/subsections of discussion as needed in order for the reader to gain a high-level understanding of the entire system or subsystem (but remember to leave the gory details for the Detailed System Design section).

If this component is very large and/or complex, you may want to consider documenting its design in a separate document and simply including a reference to it in this section. If this is the option you choose, the design document for this component should have an organizational format that is very similar (if not identical to) this document.

**Detailed System Design (1 page)**

Most components described in the System Architecture section will require a more detailed discussion. Other lower-level components and subcomponents may need to be described as well. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

***Classification***

The kind of component, such as a subsystem, module, class, package, function, file etc.

***Definition***

The specific purpose and semantic meaning of the component. This may need to refer back to the requirements specification.

***Responsibilities***

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

***Constraints***

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, post-conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

***Composition***

A description of the use and meaning of the subcomponents that are a part of this component

***Resources***

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

*Processing*

A description of precisely how this components goes about performing the duties necessary to fulfill its responsibilities. This should encompass a description of any algorithms used; changes of state; relevant time or space complexity; concurrency; methods of creation, initialization, and cleanup; and handling of exceptional conditions.

*Interface/Exports*

The set of services (resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. .... For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code. Any referenced diagrams or source code excerpts should be provided at any design reviews.

**Functional Flow**

- **Arduino Mega Board**: serves as the backbone and brain of the system. It is the one responsible for effective communication between the hardware and the software. The mobile app and the database they're a sub-system for login and registration functional requirement.
- **Location sensor** and the web browser serves as the heart of the system as they will pumping the essential need and serving the purpose of Citizen Watch app, providing the GPS co-ordinates and the map of the accident, for emergency response. Arduino
- **Bluetooth Module**: Used for connection between the mobile application and the hardware circuit.
- **Vibration Sensor**: If the vibration sensor senses vibration it sends the value "1" as reference that there's an emergency incident that has occurred.
- **Red LED's**: • They will blink when an emergency is being reported by the user.
- **Blue LED's**: • Once the button is pressed by the department admin, they will be on, while sending emergency confirmation message to the user that help it's on its way.
- **7-segments**: • Display "help" when an emergency is being reported, and "- - - - "when sending an emergency confirmation message.
- **LCD**: • It will display the current location co-ordinates {latitude and longitude} of the emergency incident since the integrated circuit will be executing in real time.

Process Flow Table

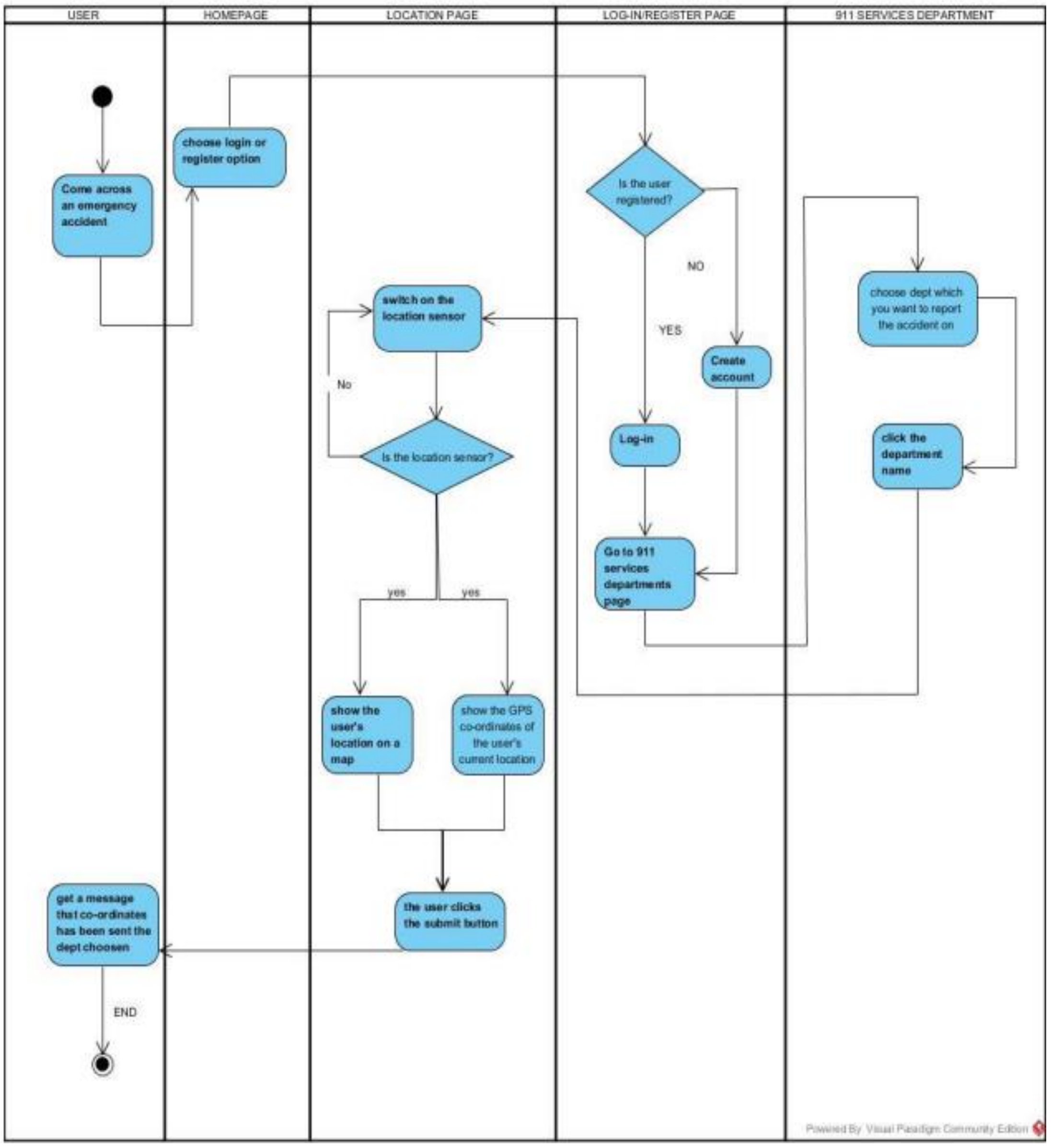
| No. | Who (to whom) | Action |
|---|---|---|
| 1. | User/Citizen | Come across an emergency accident |
| 2. | User goes to the home page of the system | Report the emergency accident |
| 3. | User to the database | Login or register |
| 4. | User to software system | Choose the department of which you want to report the accident on by clicking its name |
| 5. | User to the app location page | Switch on the location sensor |
| 6. | System to the Use | Provide location co-ordinates and map |

|  |  |  |
|---|---|---|
| 7. | User to the Hardware system | Submit the GPS co-ordinates |
| 8. | Software system to hardware system | Display the coordinates on the LCD and stores them on the database |

**Status Flow Table:**

| No. | Action | Status |
|---|---|---|
| 1. | Come across an emergency accident | implemented |
| 2. | Report the emergency accident | Implemented |
| 3. | Login or register | Implemented |
| 4. | Choose the department of which you want to report the accident on by clicking its name | Implemented |
| 5. | Switch on the location sensor | Implemented |
| 6. | Provide location co-ordinates and map | Implemented |
| 7. | Submit the GPS co-ordinates | Implemented |
| 8. | Display the coordinates on the LCD and stores them on the database | implemented |

**Activity Diagram:**



USER
HOMEPAGE
LOCATION PAGE
LOG-IN/REGISTER PAGE
911 SERVICES DEPARTMENT
Come across an emergency accident
choose login or register option
Is the user registered?
NO
YES
Create account
Log-in
Go to 911 services departments page
choose dept which you want to report the accident on
click the department name
switch on the location sensor
No
Is the location sensor?
yes
yes
show the user's location on a map
show the GPS co-ordinates of the user's current location
the user clicks the submit button
get a message that co-ordinates has been sent the dept choosen
END
Powered By Visual Paradigm Community Edition

## Conclusion

Citizen Watch Mobile app is designed for every citizen, it doesn't have age restriction, as long as you have a smart phone you can use it to report. All 911 departments will be trained on how to interact with incoming emergency reports for assistance in emergency response.

## Glossary

An ordered list of defined terms and concepts used throughout the document.

1. Emergency – a situation requiring urgent assistance.
2. Response – a reply to an objective in formal disputation
3. Preparedness – the state of being prepared.
4. Location – a particular point on place in physical space.
5. GPS Co-ordinates – are unique identifier of a precise geographic location on the earth, usually expressed in alphanumeric.
6. Good-health and Well-Being – Ensuring healthy lives and promoting well-being for all ages is important to building prosperous societies.
7. Incident – a relatively minor event.
8. Accident – an unexpected event with negative consequences occurring without the intention of the one suffering the consequences
9. Scene – the location of an event that attracts attention.

## Bibliography

A list of referenced and/or related publications.

Object-Oriented Software Engineering second edition by Timothy C.Laganiere Centers for Medicare & Medicaid Services CMS eXpedited /life Cycle(XLC) https://www.codementor.io/learn-development/what-makes-good-software-architecture-101
http://docwiki.embarcadero.com/RADStudio/Berlin/en/Running_Your_Android_Application_on_an_Android_Device#Manually_Uninstall_an_Android_Application
https://en.wikipedia.org/wiki/South_African_Police_Service

http://docwiki.embarcadero.com/RADStudio/Berlin/en/SDK_Manager
https://en.wikipedia.org/wiki/South_African_Health_Department
https://en.wikipedia.org/wiki/South_African_Fire_Fighting_Department

# USER DOCUMENT

## 1. Scope

any emergency accident and incidents by use of location.
The two major users of the Emergency Response system are the emergency reporters and 911 departments Administrators. The .apk file of the system is available on playstore

The system uses UDPclient for communication between the hardware and software

1.1 **System overview**. This paragraph shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and support agencies; identify current and planned operating sites; and list other relevant documents.

This system is designed to be used at The main and the only aspect that triggered my mind to do project of this kind is safety. Emergencies can happen anywhere and at any time. The very nature of an emergency is unpredictable and can change in scope and impact. Being prepared and planning ahead is critical to protecting lives, the environment, and property. An emergency response plan specifies procedures for handling sudden or unexpected situations. The objectives of Citizen Watch App:

- Prevent fatalities and injuries.
- Reduce damage to buildings, stock, and equipment.
- Protect the environment and the community.
Accelerate the resumption of normaloperations

1.2 **Document overview**. This paragraph shall summarize the purpose and contents of this manual and shall describe any security or privacy considerations associated with its use.

This document tracks the necessary information required to effectively define architecture and system design in order to give the users guidance on the architecture of the system that has been developed. User manuals documents are incrementally and iteratively produced during the system development life cycle, based on the particular circumstances of the information technology (IT) project and the system architecture used for developing the system. Its intended audience is the project manager, project team, and development team. Some portions of this document, such as the user interface (UI), may be shared with the client/user, and other stakeholders whose input/approval into the UI is needed.

## **1.** REFERENCED DOCUMENTS

This section shall list the number, title, version/revision, and date of all documents referenced in this manual. This section shall also identify the source for all documents not available through normal Government stocking activities.

- XLC Process Change Request (CR) (https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/XLCProcessChangeRequestCR.docx).
- ArboWebFoorest: arbonaut User's Manual

# 2. SOFTWARE SUMMARY

3.1 **Software application**. This paragraph shall provide a brief description of the intended uses of the software or hardware. Capabilities, operating improvements, and benefits expected from its use shall be described.

With increasing fatal crashes, robbery, and many other emergency set-ups and conditions being reported at late hour and that being caused by sending or providing wrong location of the emergency incident. Emergency preparedness can be defined as pre-impact activities that establish a state of readiness to response to extreme events that could affect the community's good health and well-being. The practice of emergency response planning varies considerably among communities. Citizen-Watch ensures adequate emergency preparedness, and it has a capability of performing its basic emergency response functions. Emergency response program should look at both the big picture and the small details to determine the risks and hazards potential to your operations. A plan should be developed using available resources, trained personnel, and industry best practices keeping in mind the safety of people, the environment, property, and business continuity. Experts in writing Emergency Response Plans, such as Black Gold Emergency Planners, can be consulted to help develop a program for any industry

3.2 **Software inventory**. This paragraph shall identify all software files, including databases and data files that must be installed for the software to operate. Identify software necessary to continue/resume operation in case of emergency. The identification shall include security and privacy considerations for each file and identification of the software necessary to continue or resume operation in case of an emergency.

The application will consist of mySql Database for storing the user's name, app encrypted password, date of birth, gender, and age. The database will also keep record of the GPS Location co-ordinates of where the emergency occurred.

It is also important to keep record of the accident time, age group that uses the application and see which departments has the highest number of people that report emergency accident, because that will also help in the transport and health department when drawing statics of 911 services, and also get to gather enough equipment to for emergency responses.

3.3 **Software environment**. This paragraph shall identify the hardware, software, equipment, manual operations, and other resources needed for a user to install and run the software. Included, as applicable, shall be identification of:

a. Computer equipment that must be present, including amount of memory needed, amount of auxiliary storage needed, and peripheral equipment such as terminals, printers, and other input/output devices
b. Communications equipment that must be present
c. Other software that must be present, such as operating systems, databases, data files, utilities, and other supporting systems
d. Forms, procedures, or other manual operations that must be present
e. Other facilities, equipment, or resources that must be present

3.4 **Software organization and overview of operation**. This paragraph shall provide a brief description of the organization and operation of the software from the user's point of view. The description shall include, as applicable:

a. Logical components of the software, from the user's point of view, including databases and data files the user can access, Database Management Systems (DBMSs), and communications paths, and an overview of the purpose/operation of each component
b. Performance characteristics that can be expected by the user, such as:
   1) Types, volumes, rate of inputs accepted
   2) Types, volume, accuracy, rate of outputs that the software can produce
   3) Typical response time and factors that affect it
   4) Typical processing time and factors that affect it
   5) Limitations, such as number of events that can be tracked
   6) Error rate that can be expected
   7) Reliability that can be expected
c. Relationship of the functions performed by the software with interfacing systems, organizations, or positions
d. Supervisory or security controls that can be implemented (such as passwords) to manage the software

The app will consist of three departments which the user will be reporting the emergency incident on: SAPS department, hospital department and fire-fighting department. Department will be represented by 2-LED's, button, 4-digits 7-segments displays and an LCD (Liquid Crystal Display). I will be using Embarcadero C++ builder because it has a build-in location sensor for tracking the emergency incident location co-ordinates and also Arduino Uno board. Android application Red LED's: • They will blink when an emergency is being reported by the user. Blue LED's: • Once the button is pressed by the department admin, they will be on, while sending emergency confirmation message to the user that help it's on its way. 7-segments: • Display "help" when an emergency is being reported, and " ------ "when sending an emergency confirmation message. LCD: • It will display the current location co-ordinates {latitude and longitude} of the emergency incident since the integrated circuit will be executing in real time.

Citizen Watch system is going to be embedded software, which means it will compose of both hardware and software. The components and platforms that will be used when implementing emergency response system will be as follows: Hardware

- Arduino Uno-microcontroller of the system
- Red and Blue LED's – red LED's will
- LCD (Liquid Crystal Display)
- Jumper wires
- 7-segments 4 digits
- Buttons
- Breadboard
- Resistors

3.5 **Contingencies and alternate states and modes of operation**. This paragraph shall explain differences in what the user will be able to do with the software at times of emergency and in various states and modes of operation, if applicable.

With increasing fatal crashes, robbery, and many other emergency set-ups and conditions being reported at late hour and that being caused by sending or providing wrong location of the emergency incident. Emergency preparedness can be defined as pre-impact activities that establish a state of readiness to response to extreme events that could affect the community's good health and well-being. The practice of emergency response planning varies considerably among communities. Citizen-Watch ensures adequate emergency preparedness, and it has a capability of performing its basic emergency response functions.

**Security and privacy**. This paragraph shall contain an overview of the security and privacy considerations associated with the software. A warning shall be included regarding making unauthorized copies of software or documents, if applicable.

3.6 **Assistance and problem reporting**. This paragraph shall identify points of contact and procedures to be followed to obtain assistance and report problems encountered in using the software.

The software application have contacts us pages which have the contacts of the software developer, administrator and software maintainer. In case of errors encountered by the application user they can send emails to the application maintainer and the email address is provided on the contacts page, and those emails will be stored in the database. There will be also a comment session for user's to send feedback about the application.

# 3. ACCESS TO THE SOFTWARE

This section shall contain step-by-step procedures oriented to the first time/occasional user. Enough detail shall be presented so that the user can reliably access the software before learning the details of its functional capabilities. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable.

4.1

4.1.1 **Access control**. This paragraph shall present an overview of the access and security features of the software that are visible to the user. The following items shall be included, as applicable:

a. How and from whom to obtain a password
b. How to add, delete, or change passwords under user control
c. Security and privacy considerations pertaining to the storage and marking of output reports and other media that the user will generate

4.1.2 **Installation and setup**. This paragraph shall describe any procedures that the user must perform to be identified or authorized to access or install software on the equipment, to perform the installation, to configure the software, to delete or overwrite former files or data, and to enter parameters for software operation.

4.2 **Initiating procedure**. This paragraph shall provide step-by-step procedures for beginning work, including any options available. A checklist for problem determination shall be included in case difficulties are encountered.

4.3 **Description of inputs**.

4.3.1 **Input conditions**. This paragraph shall describe the conditions to be observed in preparing each type or class of input to the software. The conditions shall include the following, as applicable:

a. Reason for input, such as normal status report, need to update data
b. Frequency of input, such as monthly, on demand
c. Origin of input, such as the organization or station authorized to generate the input
d. Medium of input, such as magnetic tape
e. Related inputs that are required to be entered at the same time as this input
f. Other applicable information, such as priority; security and privacy considerations

4.3.2 **Input formats**. This paragraph shall illustrate the layout formats to be used in the preparation of inputs to the software and shall explain the information that may be entered in the various sections and lines of each format.

4.3.3 **Composition rules**. This paragraph shall describe any rules and conventions that must be observed to prepare inputs. The rules of syntax, usage of punctuation, etc., shall be explained. The rules shall include the following, as applicable:

a. Input transaction length, such as 100 characters maximum
b. Format conventions, such as all input items must be left-justified
c. Labeling, such as usage of identifiers to denote major data sets to the software
d. Sequencing, such as order and placement of items in the input
e. Punctuation, such as spacing and use of symbols (virgule, asterisk, character combinations, etc.) to denote start and end of input, of data groups, and of fields
f. Restrictions, such as rules forbidding use of particular characters or parameter sets

4.3.4 **Input vocabulary**. This paragraph shall explain the legal character combinations or codes that must be used to prepare inputs. An appendix may be provided containing an ordered listing of these codes.

4.3.5 **Sample inputs**. This paragraph shall provide examples that illustrate and explain each type or class of input acceptable by the software. Included shall be information on the following types of inputs, as applicable:

a. Headers denoting the start of input
b. Text or body of the input
c. Trailers denoting the end of input
d. Portions of the input that may be omitted
e. Portions of the input that may be repeated

## 4.4 **Description of outputs**.

4.4.1 **General description**. This paragraph shall provide the following information, as applicable, for each type or class of output:

a. Reasons why the output is generated
b. Frequency of the output, such as monthly, on demand
c. Any modifications or variations of the basic output that is available
d. Media, such as printout, display screen, tape

e. Location where the output will appear, such as in the computer area or remotely

f. Any additional characteristics, such as priority, security and privacy considerations, or associated outputs that complement the information in this output

4.4.2 **Output formats**. This paragraph shall illustrate and explain the layout of each type or class of output from the software. The following aspects shall be explained, as applicable:

a. Security and privacy markings
b. Data that may appear in headers
c. Information that may appear in the body or text of the output, including column headings and subsets or sections in the output format
d. Data that may appear in trailers
e. Additional characteristics, such as the meaning of special symbols

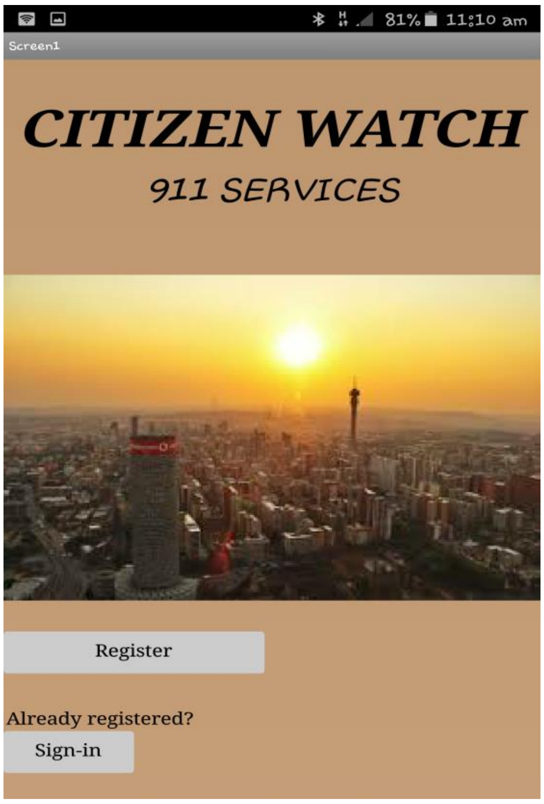
4.4.3 **Sample outputs**. This paragraph shall provide illustrations of each type or class of output from the software. A description of each sample shall be provided, including, as applicable:

a. Meaning and use of each column, entry, etc.
b. Source, such as extracted from database, calculated
c. Characteristics, such as when omitted, range of values, unit of measure

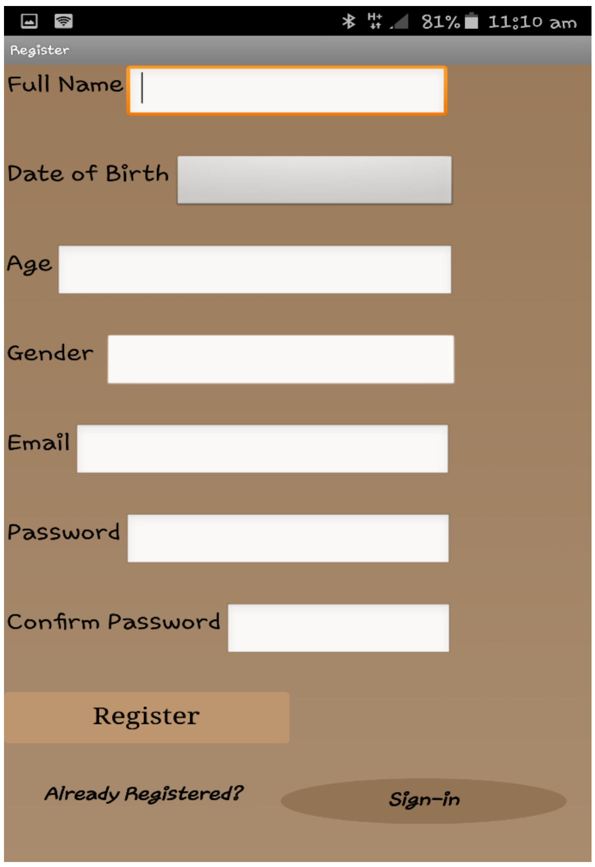
4.5 **Recovery and error correction procedures**. This paragraph shall list the error codes generated by the software, give their meanings, and describe the corrective actions to be taken by the user. Also included shall be the procedures to be followed by the user with respect to restart, recovery, and continuity of operations in the event of emergencies.

4.6 **Stopping and suspending work**. This paragraph shall describe how the user can cease or interrupt use of the software and how to determine whether normal termination or cessation has occurred.

- **User will get the application on application store, like PlayStore, they just have to press “Citizen Watch” on the search space, then enter**
- **The Application will pop-up and there’s an install button for the downloading of the application using smart phone. With screenshots and a short description about the application**
- **Press install then the application will be downloaded and installed to the user’s smart phone, and a shortcut will be created on the home screen**

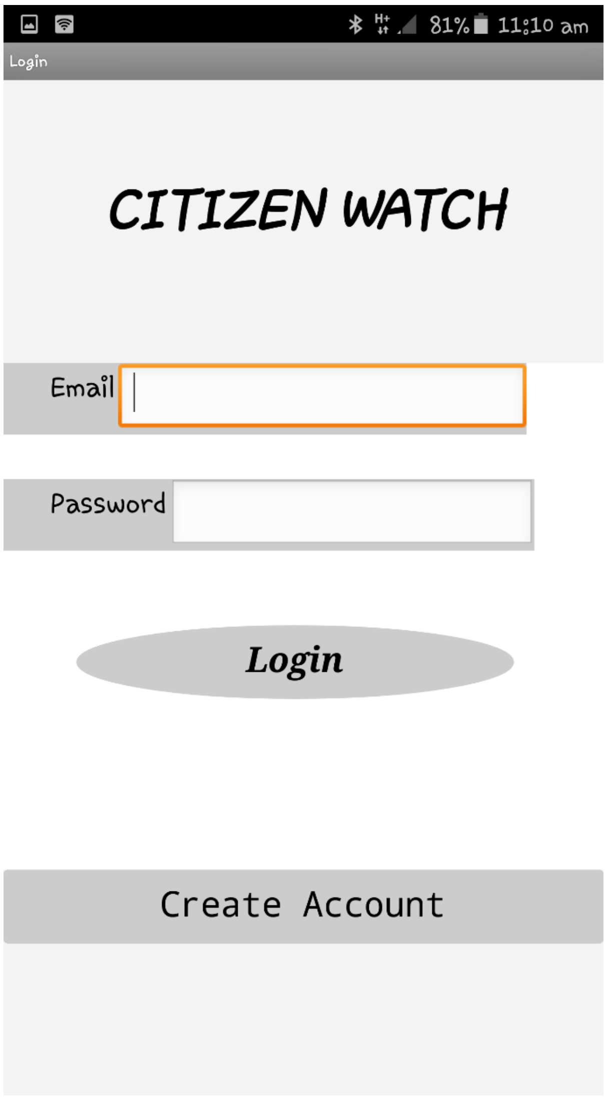


- 
- **This is how the home page of the application look like. From here the user Sign-in if the user is already registered or Register if a first time user**

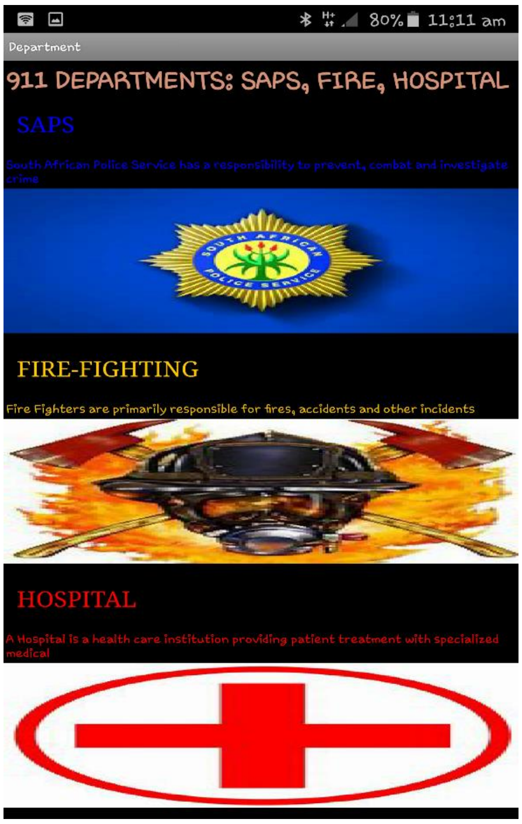


- 
- And registration page look like this. The user will then provide his or her credential's and data will be stored on the database.

- 
    - **After filling the registration form, Then press “Register” button and data will be stored on the database.**



For registered users, they can just sign-in in this login form by providing their email and password which they registered with and they going to be directed to the department page.


80% 11:11 am

Department

911 DEPARTMENTS: SAPS, FIRE, HOSPITAL

SAPS

South African Police Service has a responsibility to prevent, combat and investigate crime

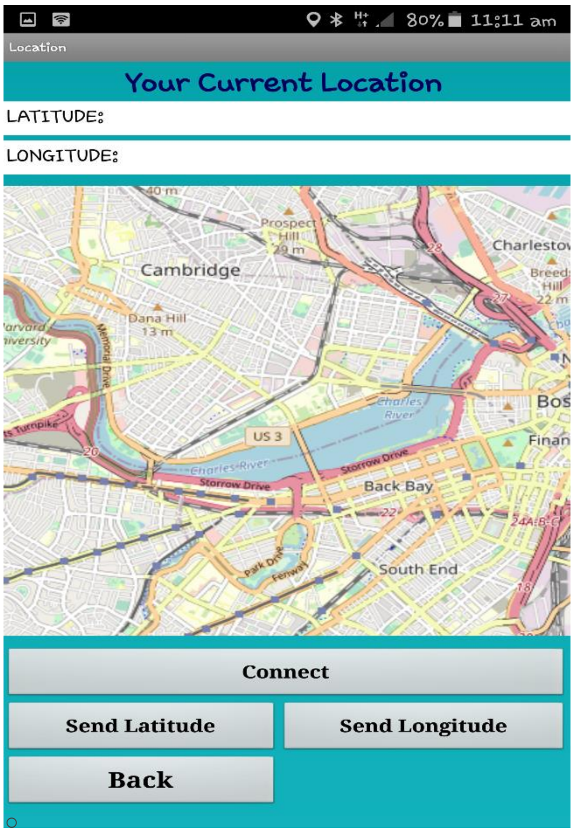
FIRE-FIGHTING

Fire Fighters are primarily responsible for fires, accidents and other incidents

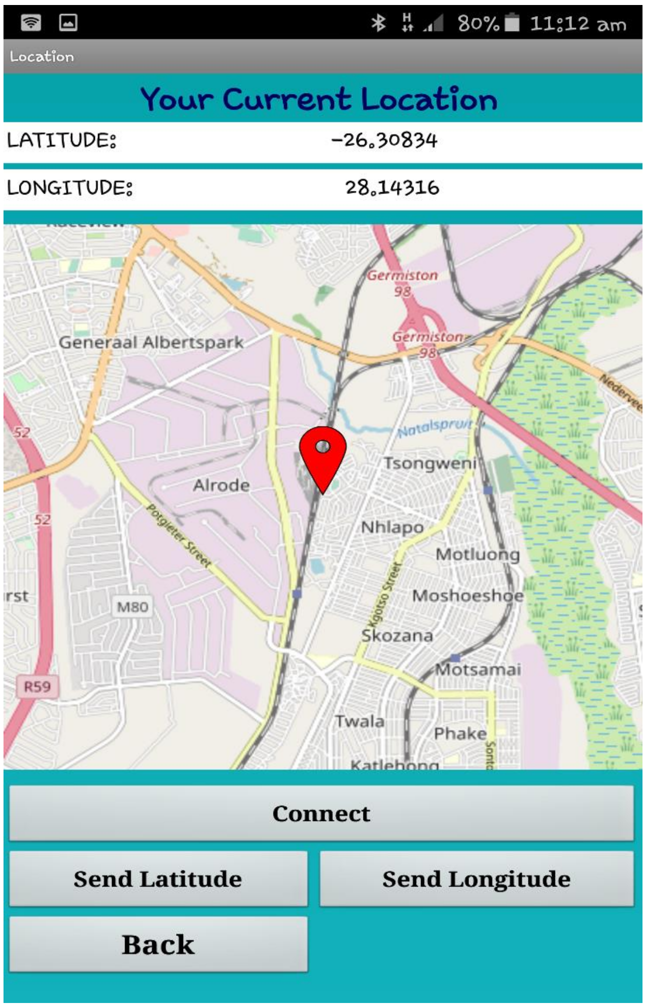
HOSPITAL

A Hospital is a health care institution providing patient treatment with specialized medical

- 
- **After Signing-in or Registering the application is going to take the user to the Department page. On the department page the user has an option to choose the department of which the user wants to report on by clicking the department name.**



- After clicking the department name the application take the user to the location page or the user can go back to the department page if the user has mistakenly chosen the wrong department, or switch on location sensor.


Location

Your Current Location

LATITUDE: -26.30834

LONGITUDE: 28.14316

Connect

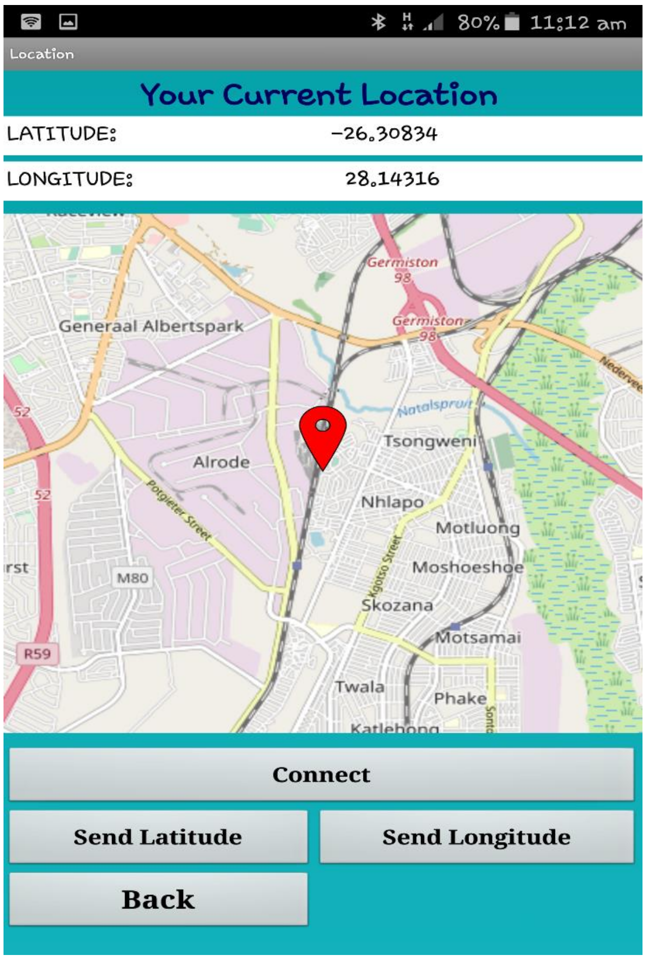
Send Latitude

Send Longitude

Back

○

- After switching on the location sensor, the application will show the user his or her GPS location co-ordinates which is the one that will be sent to the department the user wants to report the emergency accident or incident on
-



Your Current Location

| | |
|---|---|
| LATITUDE: | -26.30834 |
| LONGITUDE: | 28.14316 |

- It will display the map showing the user his/her own location, the user must then press the submit button then wait for report stating that the help is on its way.

# **4.** PROCESSING REFERENCE GUIDE

This section shall describe the functionality provided by and specify procedures for using the software. The organization of the document will depend on the characteristics of the software being documented. If procedures are complicated or extensive, additional Sections 6, 7, ... may be added in the same paragraph structure as this section and with titles meaningful to the sections selected. For example, one approach is to base the sections on the organizations in which users work, their assigned positions, their work sites, or the tasks they must perform. For other software, it may be more appropriate to have Section 5 be a guide to menus, Section 6 be a guide to the command language used, and Section 7 be a guide to functions. Detailed procedures are intended to be presented in subparagraphs of paragraph 5.3. Depending on the design of the software, the subparagraphs might be organized on a function-by-function, menu-by-menu, transaction-by-transaction, or other basis. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable. Whatever the method of organization, the format for presenting information must have a consistent style.

5.1 Capabilities. This paragraph shall briefly describe the interrelationships of the transactions, menus, functions, or other processes in order to provide an overview of the use of the software.

5.2 Conventions. This paragraph shall describe any conventions used by the software, such as the use of colors in displays, the use of audible alarms, the use of abbreviated vocabulary, and the use of rules for assigning names or codes.

5.3 Processing procedures. This paragraph shall explain the organization of subsequent paragraphs, e.g., by function, by menu, by screen. Any necessary order in which procedures must be accomplished shall be described.

5.4 Aspect of software use. The title of this paragraph shall identify the function, menu, transaction, or other process being described. This paragraph shall describe and give options and examples, as applicable, of menus, graphical icons, data entry forms, user inputs, inputs from other software or hardware that may affect the software's interface with the user, outputs, diagnostic or error messages or alarms, and help facilities that can provide on-line descriptive or tutorial information. The format for presenting this information can be adapted to the particular characteristics of the software, but a consistent style of presentation shall be used, i.e., the descriptions of menus shall be consistent; the descriptions of transactions shall be consistent among themselves.

5.5 Related processing. This paragraph shall identify and describe any related batch, off-line, or background processing performed by the software that is not invoked directly by the user and is not described in paragraph 5.3. Any user responsibilities to support this processing shall be specified.

5.6 Data backup. This paragraph shall describe procedures for creating and retaining backup data that can be used to replace primary copies of data in event of errors, defects, malfunctions, or accidents.

5.7 Recovery from errors, malfunctions, and emergencies. This paragraph shall present detailed procedures for restart or recovery from errors or malfunctions occurring during processing and for ensuring continuity of operations in the event of emergencies.

5.8 Messages. This paragraph shall list, or refer to an appendix that lists, all error messages, diagnostic messages, and information messages that can occur while accomplishing any of the user's functions. The meaning of each message and the action that should be taken after each such message shall be identified and described.

5.9 Quick-reference guide. If appropriate to the software, this paragraph shall provide or reference a quick-reference card or page for using the software. This quick-reference guide shall summarize, as applicable, frequently-used function keys, control sequences, formats, commands, or other aspects of software use.

## 5. QUERY PROCEDURES

6.1 Database/data file format. This paragraph shall provide a user's view of the format and content of each database and data file that can be queried. Figure 1 provides an example. Information such as the following shall be provided for each data element, as applicable:

a. Data element name
   NAME as FULL NAME
   USERNAME as USERNAME
   DOB as DATE OF BIRTH
b. Synonymous names
   Used SIGN-IN instead of LOG-IN
c. Definition
d. Format
   I have used font-family: arial and size of 12
e. Range and enumeration of values
f. Unit of measurement
g. Data item names, abbreviations, and codes

6.2 Query capabilities. This paragraph shall identify and describe the preprogrammed and ad hoc query capabilities provided by the software.

6.3 Query preparation. This paragraph shall provide instructions for preparing queries. Figure 3 shows an example of the format for a preprogrammed query.

6.4 Control instructions. This paragraph shall provide instructions for the sequencing of runs and other actions necessary to extract responses to query requests. These instructions shall include control statements that may be required by the computer system or software.

- **User will get the application on application store, like PlayStore, they just have to press “Citizen Watch” on the search space, then enter**
- **The Application will pop-up and there’s an install button for the downloading of the application using smart phone. With screenshots and a short description about the application**
- **Press install then the application will be downloaded and installed to the user’s smart phone, and a shortcut will be created on the home screen**
- **This is how the home page of the application look like. From here the user Sign-in if the user is already registered or Register if a first time user**
- **And registration page look like this. The user will then provide his or her credential’s and data will be stored on the database.**

**Then press “Register” button and data will be stored on the database**

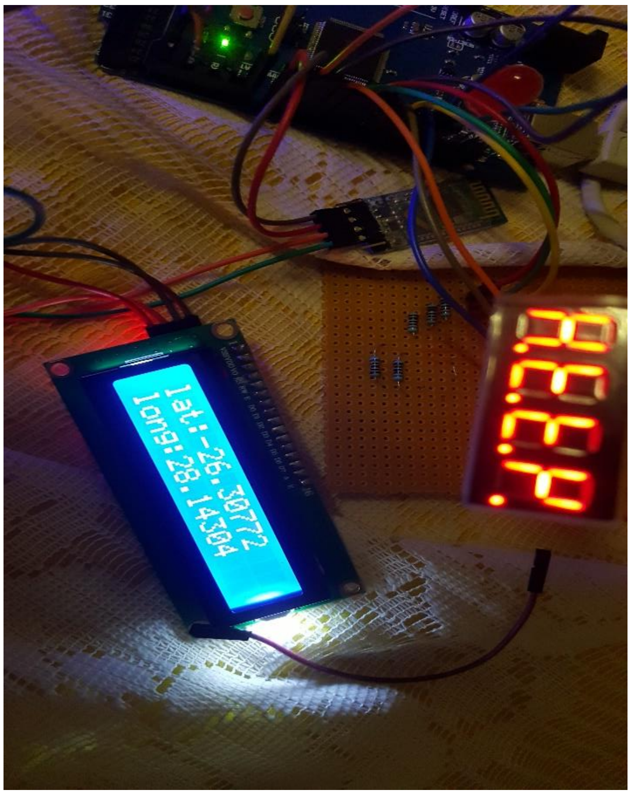
- **After Signing-in or Registering the application is going to take the user to the Department page. On the department page the user has an option to choose**

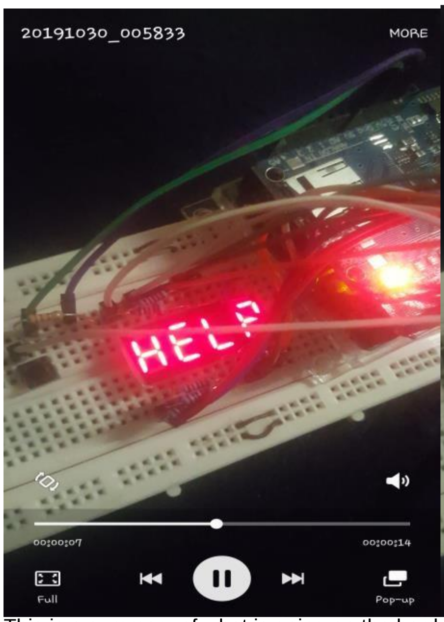
**the department of which the user wants to report on by clicking the department name.**

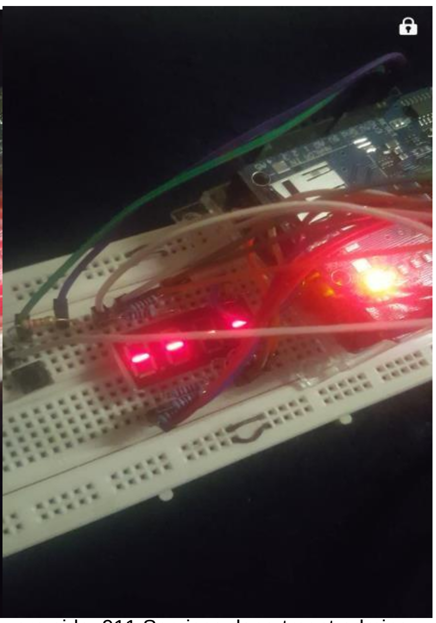
- **After clicking the department name the application take the user to the location page or the user can go back to the department page if the user has mistakenly chosen the wrong department, or switch on location sensor.**
- **After switching on the location sensor, the application will show the user his or her GPS location co-ordinates which is the one that will be sent to the department the user wants to report the emergency accident or incident on**
- **It will display the map showing the user his/her own location, the user must then press the submit button then wait for report stating that the help is on its way.**
-

## **6.** NOTES

This section shall contain any general information that aids in understanding this document (e.g., background information, glossary, rationale). This section shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of terms and definitions needed to understand this document.






This is a summary of what is going on the hardware side: 911 Services department admins

## **7.** REFERENCE

http://docwiki.embarcadero.com/RADStudio/Berlin/en/Running_Your_Android_Application_
on_an_Android_De vice#Manually_Uninstall_an_Android_Application
https://en.wikipedia.org/wiki/South_African_Police_Service
http://docwiki.embarcadero.com/RADStudio/Berlin/en/SDK_Manager
https://en.wikipedia.org/wiki/South_African_Health_Department
https://en.wikipedia.org/wiki/South_African_Fire_Fighting_Department

## A. APPENDIXES

Appendixes may be used to provide information published separately for convenience in document maintenance (e.g., charts, classified data). As applicable, each appendix shall be referenced in the main body of the document where the data would normally have been provided. Appendixes may be bound as separate documents for ease in handling.  Appendixes shall be lettered alphabetically (A, B, etc.).

# APPENDICES

(Source code and picture of project)

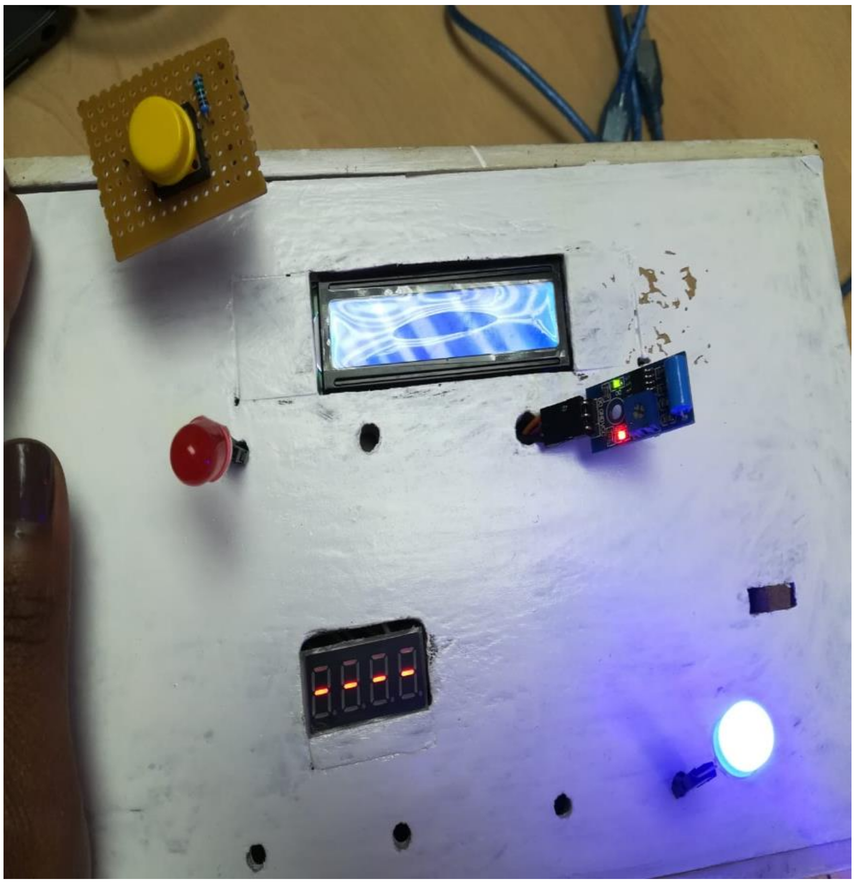
## SOURCE CODE

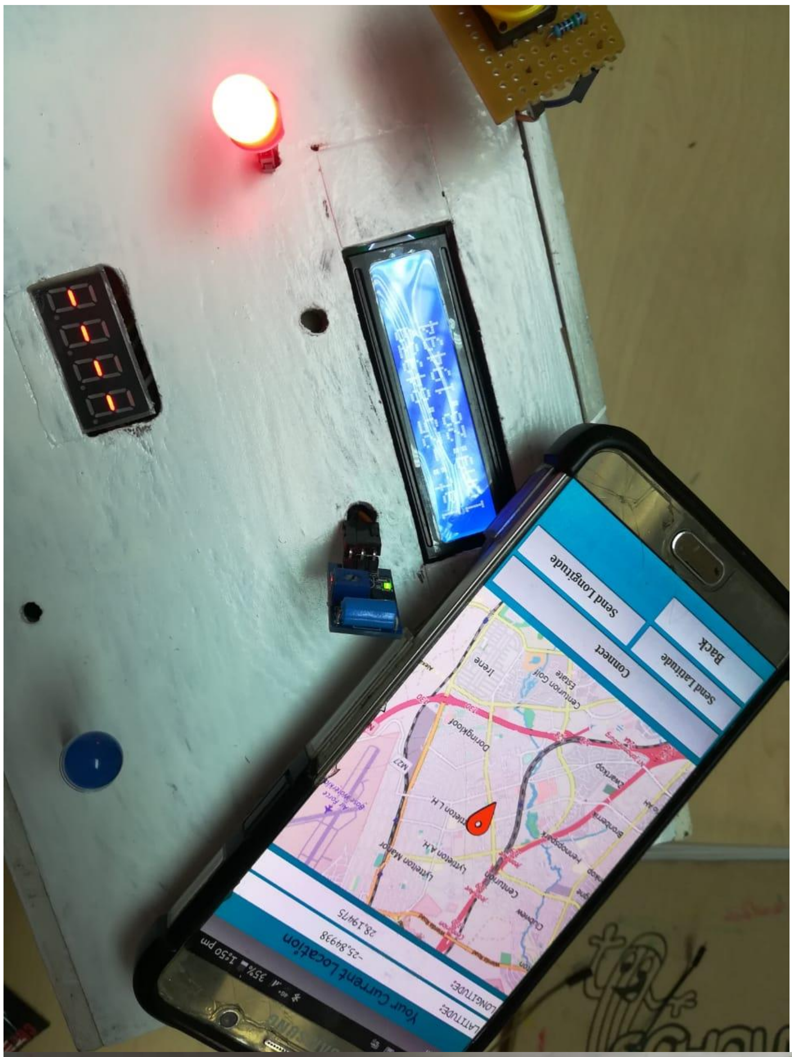
```
#include <TimerOne.h>
#include <Wire.h>
#include <FastIO.h>
#include <I2CIO.h>
#include <LCD.h>
#include <LiquidCrystal.h>
#include <LiquidCrystal_I2C.h>
#include <LiquidCrystal_SR.h>
#include <LiquidCrystal_SR2W.h>
#include <LiquidCrystal_SR3W.h>


//I2C pins declaration
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
 int var=0;
int val=0;
int vib=16;
int ledPower = 18;  //blue LED
int butt=19;
int ledEmergency = 13;  //RED LED for emergency reporting
String latitude = "";
String longitude = "";

//7-SEGMENT

int a = 7;
int b = 3;
int c = 4;
int d = 5;
int e = 6;
int f = 15;
int g = 8;
int p = 9;
int d4 = 10;
int d3 = 11;
int d2 = 12;
int d1 = 14;
long n = 0;// n represents the value displayed on the LED display. For example, when n=0, 0000 is displayed. The maximum
value is 9999.
int x = 100;
int del = 5;//Set del as 5; the value is the degree of fine tuning for the clock
int count = 0;//Set count=0. Here count is a count value that increases by 1 every 0.1 second, which means 1 second is
counted when the value is 10



void setup() {
  int var=0;
val=digitalRead(butt);

  pinMode(vib, INPUT);

lcd.begin(16,2);//Defining 16 columns and 2 rows of lcd display
lcd.backlight();//To Power ON the back light

  Serial.begin(9600);
   Wire.begin();
 // pinMode(led, OUTPUT);

   pinMode(d1, OUTPUT);
  pinMode(d2, OUTPUT);
  pinMode(d3, OUTPUT);
  pinMode(d4, OUTPUT);
  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);
  pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);
  pinMode(e, OUTPUT);
  pinMode(f, OUTPUT);
  pinMode(g, OUTPUT);
  pinMode(p, OUTPUT);

    Timer1.initialize(100000); // set a timer of length 100000 microseconds (or 0.1 sec - or 10Hz => the led will blink 5 times,
5 cycles of on-and-off, per second)
  Timer1.attachInterrupt( add ); // attach the service routine here

pinMode(ledPower,OUTPUT);//initialize the ledPin as an output

pinMode(butt,INPUT);//initialize the ledPin as an output

pinMode(ledEmergency,OUTPUT);//initialize the ledPin as an output
pinMode(2,INPUT);//set pin2 as INPUT
digitalWrite(2, HIGH);//set pin2 as HIGH

digitalWrite(ledPower,LOW);
}

void loop()
{
  int digitalVal = digitalRead(vib);//Read the value of pin2
if(HIGH == digitalVal)//if tilt switch is not breakover
{
  digitalWrite(led,LOW);



  if(Serial.available()>0)
```

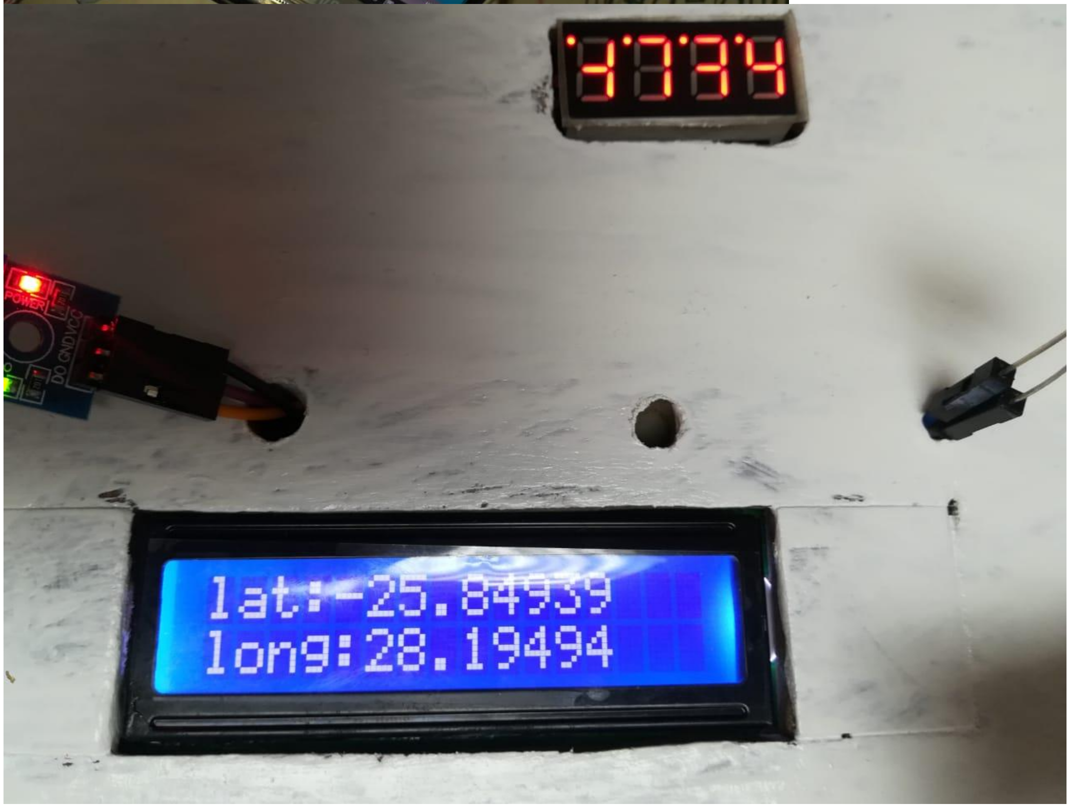
```
    {
       latitude= Serial.readString(); //
        //lcd.setCursor(0,0); //Defining positon to write from first row,first column .
        lcd.setCursor(0,0);
         lcd.print("lat:");
         lcd.setCursor(4,0);
        lcd.print(latitude);


          delay(2000);

           longitude= Serial.readString();
           lcd.setCursor(0,1);
            lcd.print("long:");
            lcd.setCursor(5,1);
            lcd.print(longitude);


if(val==LOW)
{
  digitalWrite(led,LOW);

  while(var<5)
{


digitalWrite(ledPin,HIGH);
delay(1000);
digitalWrite(ledPin,LOW);
delay(1000);
var++;
}

}

else
if(val==HIGH)
{
  digitalWrite(led,LOW);
}

}

  clearLEDs();//clear the 7-segment display screen
  pickDigit(0);//Light up 7-segment display d1
h();

  clearLEDs();//clear the 7-segment display screen
  pickDigit(1);//Light up 7-segment display d2
e();

  clearLEDs();//clear the 7-segment display screen
  pickDigit(2);//Light up 7-segment display d3
  l();

  clearLEDs();//clear the 7-segment display screen
  pickDigit(3);//Light up 7-segment display d4
  pp();

}

  else////if tilt switch breakover
if(LOW == digitalVal  )
{
digitalWrite(led,HIGH);//turn the led on

clearLEDs();//clear the 7-segment display screen
  pickDigit(0);//Light up 7-segment display d2
desh();

  clearLEDs();//clear the 7-segment display screen
  pickDigit(1);//Light up 7-segment display d2
desh();

  clearLEDs();//clear the 7-segment display screen
  pickDigit(2);//Light up 7-segment display d4
  desh();

  clearLEDs();//clear the 7-segment display screen
  pickDigit(3);//Light up 7-segment display d4
 desh();
}

}


void pickDigit(int x) //light up a 7-segment display
{
  //The 7-segment LED display is a common-cathode one. So also use digitalWrite to  set d1 as high and the LED will go out
  digitalWrite(d1, HIGH);
  digitalWrite(d2, HIGH);
  digitalWrite(d3, HIGH);
  digitalWrite(d4, HIGH);

  switch(x)
  {
    case 0:
    digitalWrite(d1, LOW);//Light d1 up
    break;
    case 1:
    digitalWrite(d2, LOW); //Light d2 up
    break;
    case 2:
    digitalWrite(d3, LOW); //Light d3 up
    break;
    default:
    digitalWrite(d4, LOW); //Light d4 up
    break;
  }
}

void clearLEDs() //clear the 7-segment display screen
{
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
  digitalWrite(p, LOW);
}
void h() //the 7-segment led display 0
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, HIGH);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
  digitalWrite(p, HIGH);
}

void e() //the 7-segment led display 0
{
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  digitalWrite(p, HIGH);
}

void l() //the 7-segment led display 0
{
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
  digitalWrite(p, HIGH);
}

void pp() //the 7-segment led display 0
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  digitalWrite(p, HIGH);
}
```

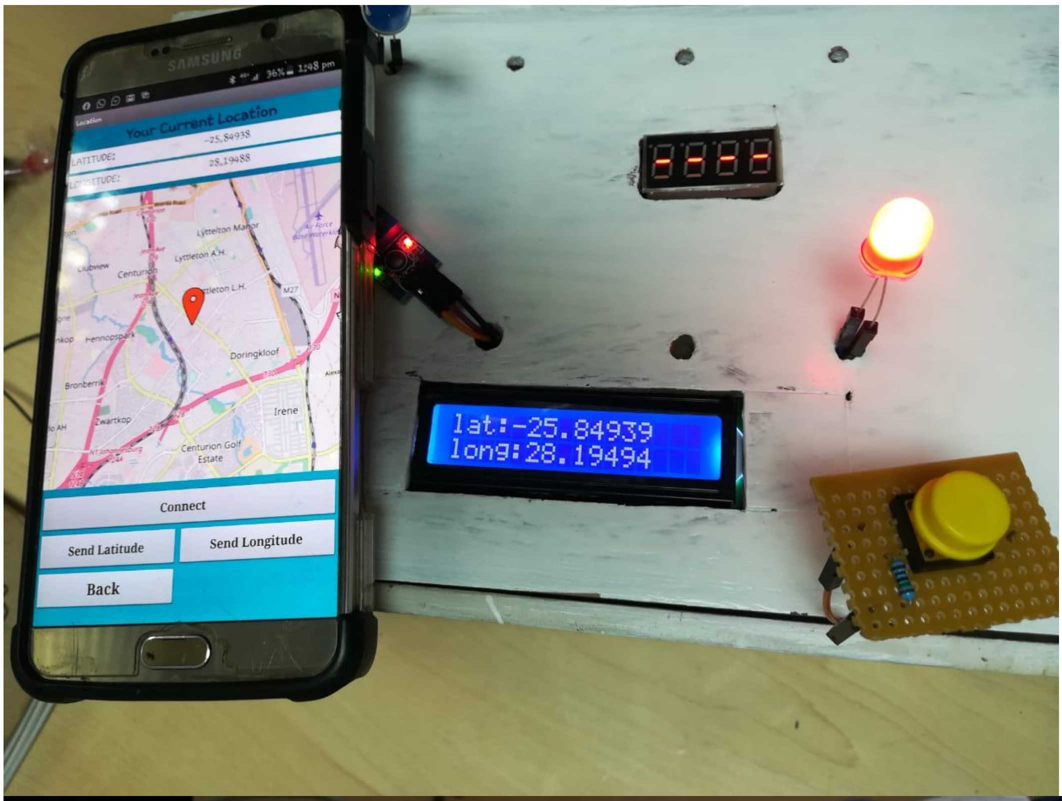
```
void desh() //the 7-segment led display 0
{
   digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
  digitalWrite(p, LOW);
}

void add()
{
  // Toggle LED
  count ++;
  if(count == 10)
  {
    count = 0;
    n++;
    if(n == 10000)
    {
      n = 0;
    }
  }
}
```

## PICTURES



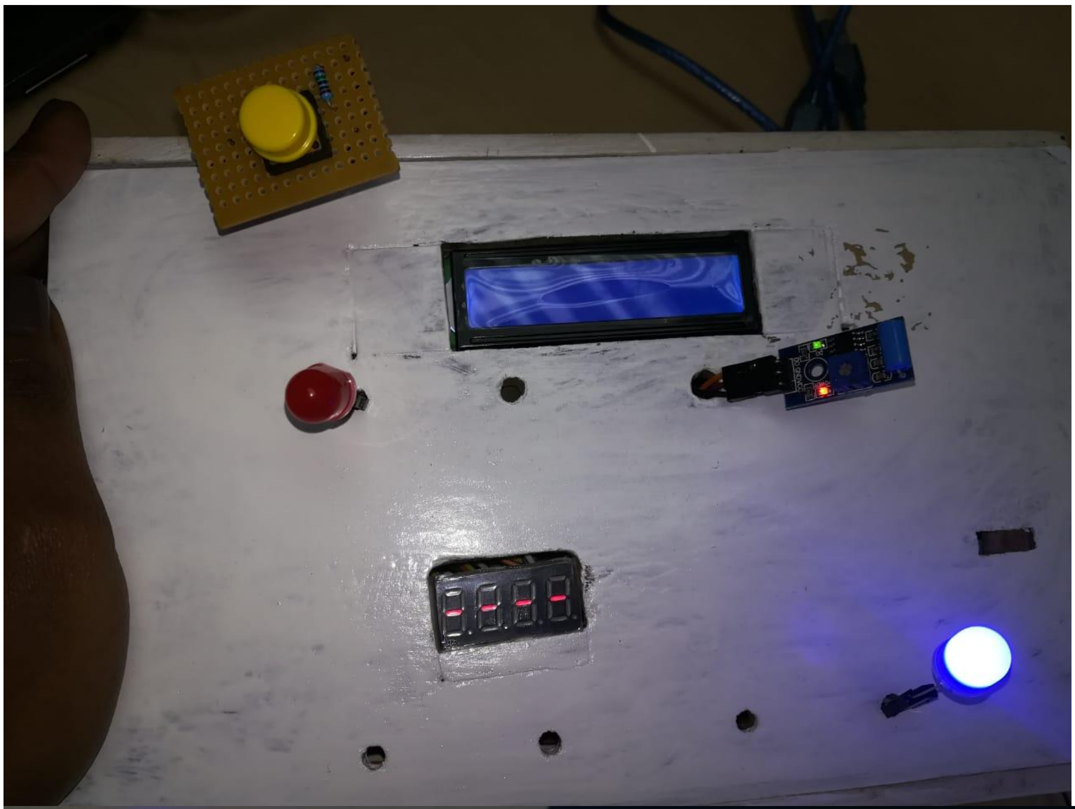
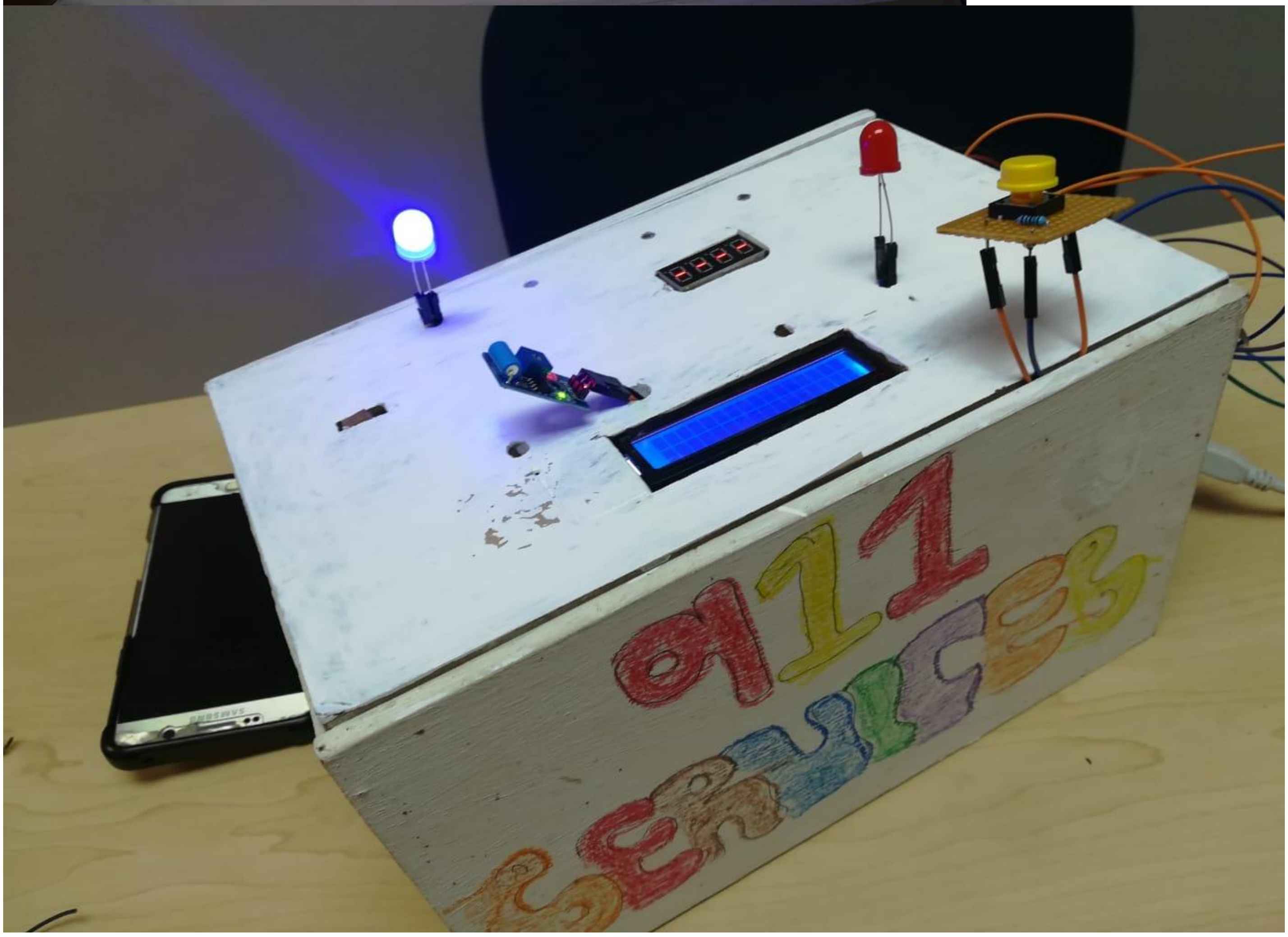

Your Current Location
LATITUDE: -25.84938
LONGITUDE: 28.19475
Send Longitude
Connect
Back
Send Latitude

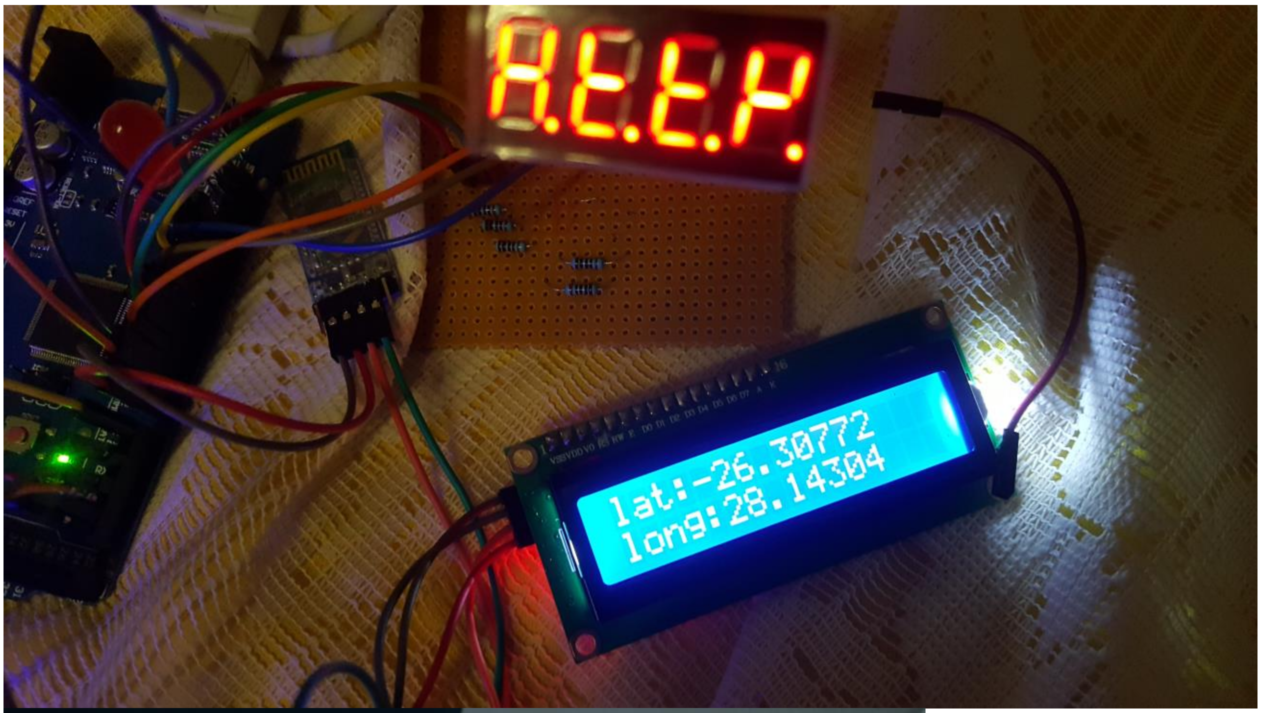

4734
lat:-25.84939
long:28.19494



SAMSUNG
Your Current Location
LATITUDE: -25,84938
LONGITUDE: 28,19488
Connect
Send Latitude
Send Longitude
Back
lat:-25.84939
long:28.19494

citizen



911
SERVICES


H.E.L.P.
lat:-26.30772
long:28.14304