*IOSR Journal of Computer Engineering (IOSR-JCE)*
*e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 22, Issue 1, Ser. I (Jan - Feb 2020), PP 08-30*
*www.iosrjournals.org*

# Newly Proposed Prototype Model for Riot Control Monitoring and Support System

SherifKamel Hussein Hassan Ratib
*Associate Professor- Department of Communications and Computer Engineering,*
*October University for Modern Sciences and Arts, Giza, Egypt*
*Head of Computer Science Department Arab East Colleges – Riyadh- KSA*

**Abstract:** *Riot control sector play a difficult role in every nation's internal affairs, however, in the past decade many challenges has been facing this sector internationally and nationally as well, resulting into some problems that have been identified. The Egyptian Central Security Forces are the main department that runs the riot control sector, it is noted that they do not utilize any form of modernized framework to monitor, track and manage their operations which causes many of the problems identified for example the inaccuracy and inefficacy of commands due to lack of data. A Riot Control Monitoring and Support System is proposed as a multi-functional management solution that aims to provide seamless operation command and increases the operational capabilities of the maneuver units from battalion to platform/single conscripts level. The design of this system is divided into three layers, Embedded systems layer which is represented by the in-field wearable and stationary devices that provide geospatial and physiological data to the system, Server-Side layer which communicates with the embedded systems layer, also is responsible for storing, structuring and feeding the entire system with the required data and Client-Side layer that enables the user to display visualized data, control and manage the entire system. The main and most important achieved objective of this system is to reduce the casualties in operations on both parties.*

**Keywords:** *Riot,Egyptian Central Security Forces (CSF), Baton, Battalion,Company, Platoon, Conscripts: ,Wireless Sensor Network (WSN),System on a Chip (SoC). Riot Control Monitoring and support systems (RCMSS),Java Script Object Notion (JSON),Application Programming Interface (API),Hyper Text Transfer Protocol (HTTP)*

---

Date of Submission: 26-12-2019 Date of Acceptance: 11-01-2020

---

## I. Introduction

Riots have become a very common thing to learn about from the news or by reading the morning newspaper in the 21$^{st}$century, especially in the last decade. United Kingdom, United States of America, Ukraine, Turkey, Greece and last but not least Egypt, all are countries that recently had riot occurrences.

In the past couple of years the Egyptian Central Security Forces (CSF) have been facing major challenges in the riot control department due to the country's political state, therefore the proposed system aims to provide effective solutions to some of the encountered problems which will improve the overall riot control process and most importantly is to reduce the causalities on both parties.

CSF deployment is divided into three parts - Baton, Gas, and Armed. The primary unit formation of the CSF is a Company which is deployed using various defense/offence formations, a company is commanded by a Police Captain and a senior Lieutenant as a secondary command, each company contains an average of 150 Conscripts which are structured into 3 platoons, Baton platoon, Gas platoon and Armed platoon. The Baton platoons are equipped with batons and shields, the gas platoon is responsible for the deployment of Teargas, and the armed platoon usually carries assault rifles. Approximately 2 to 4 companies form a CSF Battalion.

The Battalion is commanded by a ranking officer of either Lieutenant Colonel or Colonel rank, they monitor the operation on the clock through a central intelligence location. This location is the Operation Room, it provides tactical and strategical decisions or support to the companies' commanders through radio/wireless communication [1-3].

## II. Problem facing Central Security Forces

The technologies used by the CFS are outdated and lack a lot of key features that can effectively enhance the operations. The problems are categorized as the following:

### 2.1 No real-time health monitoring

Platoon conscripts are in constant contact with teargas, exposed to physical injuries and exhaustion, all these threats lead to sudden collapse or fainting. Above that, the relationship between conscripts and higher ranks does not easily permit them to report their physical condition, allof that eventually lead the conscripts to collapse in the operation field which in worst cases may lead to their death. Lack of vitals monitoring has a negative effect to the strength of the platoon.

### 2.2 No real-time position and movement tracking

The CFS Operations Room has no real-time position tracking of the CSF deployed formation in the field, the CFS operations report the updates and reports by relying solely on verbal communication being passed through the hierarchy of commanding ranks, this affects the accuracy of the operation decisions.

### 2.3 Insufficient operation data logging and management

In order to improve the future operations' effectiveness, studies and analysis are executed by CFS against previous operations data, without a means of logging and recording of the operations such practices are not as effective as they could be.

### 2.4 Lack of Operation Management System

The unavailability of a full operation management system, that encapsulates monitoring, tracking and data logging solutions, a system which provides tools to control deployments for the different scopes starting from Battalion view down to single conscript view, tools to manage all kinds of operation data such as personnel information or assigned wearable nodes and also provide visual feedback on different indicators and notifications to the users.

## III. Literature Review

After intensive research and review of a lot of conferences and papers, there has not been any system found that is identical to the proposed system or in other terms serves the riot control sector, however some reviewed systems have been found to have similar approaches in implementation and design. The selected systems are classified into 2 fields, Military Applications and Commercial Applications.

### 3.1 Real Time Soldier Tracking System[4]

The reviewed system aims to develop a system that will automatically trace the location of soldiers, also continuously track health parameters such as heart rate and body temperature of the soldier.

This system proposes a gadget joined in the soldier's pocket. This gadget contains a pulse sensor which constantly gives the perusing of pulse of warrior to the base station. The temperature sensor will give the information of body temperature. The framework additionally incorporates GPS and GSM modules. Through the GPS module, area of the warrior is followed and through GSM the officer can discuss specifically with the base unit. Another piece of this framework is the base unit which go about as collector where the total information of every last warrior is recorded. The framework provides two-way correspondence. The framework can execute questions from the collector side to know the status of the soldier and get programmed reaction from the framework side to know when the pulse of the soldier stops if there should arise an occurrence of death event.

The system is composed of two parts, soldier unit fixed on the soldier's jacket and a base unit used to monitor the data received from the soldier unit.

### 3.2 End-To-End Geospatial Solutions for Military Land Operations[5]

The document is published by LUCIAD, a large corporation specialized in providing military sized modern solutions.Many defense organizations still manage and distribute their geospatial data, intelligence and real-time information in isolated stovepipe applications. Command and Control systems at headquarter staff level operate separately from the units' Battle Management systems, which in turn are uncoupled from the deployed commander and dismounted warfighter. Sharing mission plans, mission reports and real-time information are often a difficult processes requiring a lot of human intervention, LUCIAD provides a high end and modern solution to these issues through its battlefield management system.

### 3.3 Vehicle Monitoring and Tracking System Using Android App [ 6]

This system introduces facts about the increasing number of vehicles that are stolen in the streets and unsecured parking lots. The paper also reviews many plans and methods that have been composed and actualized in the vehicles. However, planning a vehicle security framework and interfacing the observing by the proprietor's cell phone will be the outright answer for the present circumstance and need.

### 3.4 Pilgrim Tracking and Health Monitoring System[7]

This system discusses the diversity of India's religions, every year about thousands of mystic individuals devote their time for procession and cover miles of distance in worship of God. Tracking of such large amount of procession is really difficult and also inconvenient for managing authorities. The system also proposes a solution model that is divided into two parts, a Mobile Unit and a Server Unit.

Each pilgrim is accommodated with mobile unit, which is identified by its unique ID. Mobile unit consists of GPS chip, LPC2138 micro-controller, GSM module, LCD, sensors, keypad. With reference to received query, Mobile unit transmits its user identification (UID) and time stamp information to server.Central Server consists of GSM module, PC and LPC 2138. The central server keeps track of every individual pilgrim location and information.

Server will transmit the pilgrim's current location to their respective relative via GSM on request. The design provides optional keys on mobile unit for pilgrims use to request for help in case of medical and food emergency.

### 3.5 Previous Systems Analysis

The reviewed systems were selected from various fields and based on their similarity to the proposed system, however they were appropriately studied. And extensively analyzed, the reviewed systems were compared under several parameters explained in Table 1 and the results for the analysis in Table 2.

**Table 1.** Evaluation Parameters

| Parameters | Definitions | Possible Values |
|---|---|---|
| Robustness | The ability of a computer system to cope with errors and failures. | Yes/No |
| Extendibility | Ability of a system to adapt new features. | Yes/No |
| Ease of Use | User-friendly | Yes/No |
| Cost Effectiveness | Ability to produce good results without costing a lot of money | Yes/No |
| Reliability | How well the system performs in its claim time and in a particular environment | Yes/No |
| Effectiveness | Success rate of the system | Yes/No |

**Table 2.** Literature Review Comparisons

| System Name | Robustness | Extendibility | Ease of Use | Cost Effectiveness | Reliability | Effectiveness |
|---|---|---|---|---|---|---|
| **Real-Time Soldier Tracking** | No | No | No | Yes | No | No |
| **LUCIAD Battlefield Management** | No | Yes | Yes | No | Yes | Yes |
| **Vehicle Monitoring And Tracking** | Yes | No | Yes | Yes | No | No |
| **Pilgrim Tracking And Health Monitoring** | No | No | No | Yes | No | Yes |

## IV. The Newly Proposed System

### 4.1 Riot Control Monitoring and Support System (RCMSS)

The proposed system key feature is its multi-functionality, the system provides seamless integration between hardware, server and client layers providing multiple operations needed functions such as:

- Multiple Users Authentication
- View various indicators and statistics from custom designed Dashboard
- Health monitoring through indicators generated from each wearable node
- View, Control and Monitor Operations geospatially in real-time from single unit to entire battalion scope
- Receive and manage real-time warnings and notifications
- Manage Wearable Nodes
- Manage Base Stations
- Manage Battalions, Companies and Platoons
- Manage and Export data logs and records for further analysis

### 4.2 System Description

Figure 1 shows the main block diagram for RCMSS .The proposed system is divided into three main parts, Embedded System Layer, Server-Side Layer and Client-Side Layer. All layers are integrated together and will be explained in the next sections.

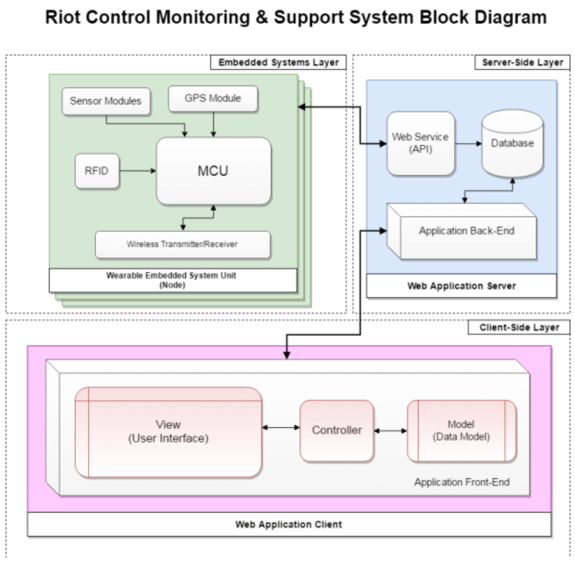


**Figure 1.** RCMSS Block Diagram

### 4.3 Embedded Systems Layer

This layer is specified as various standalone devices, aliased Wearable Node. Each of these devices has its own functionality that plays an important role in the system as a whole.
The wearable node device consists of a microcontroller open-source development board, wireless communication module, GPS module, Sensors, RFIDs and a power supply.

#### 1.1.1. Microcontroller Unit

The microcontroller unit acts as the heart of the system,a microcontroller is a self-contained system with peripherals, memory and a processor that can be used as an embedded system. Most programmable microcontrollers that are used today are embedded in other consumer products or machinery including phones, peripherals, automobiles and household appliances for computer systems.

Based on system requirements, market availability and lower cost the Arduino Mega 2560 was selected to be the open-source microcontroller development board used for the Wearable Node and the Base Station as well. Arduino Mega 2560 development board picture and specifications are detailed in Figure 2 and Table 3 [8].

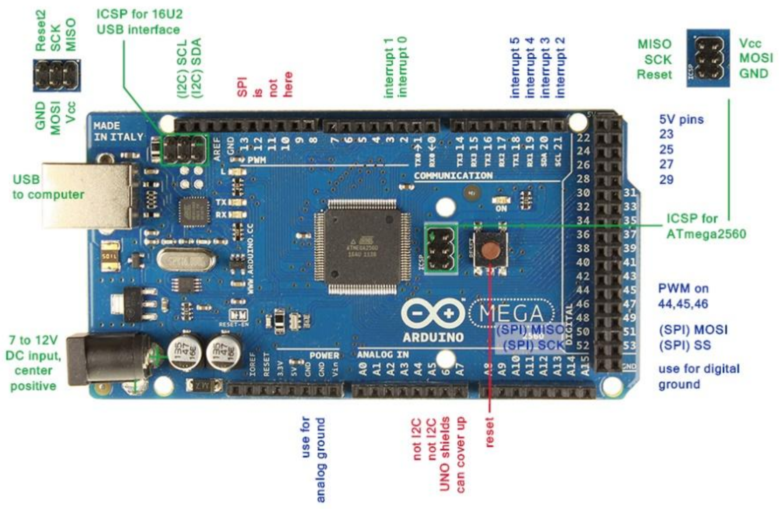


**Figure 2.** Arduino Mega 2560 Labeled Picture

**Table 3.** Arduino Mega 2560 Specifications

| | |
|---|---|
| Microcontroller | ATmega2560 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 54 (of which 15 provide PWM output) |

| | |
|---|---|
| Analog Input Pins | 16 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 256 KB of which 8 KB used by bootloader |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Clock Speed | 16 MHz |
| LED_BUILTIN | 13 |
| Length | 101.52 mm |
| Width | 53.3 mm |
| Weight | 37 g |

### 1.1.2. Global Positioning System (GPS) Module

The GPS is a global navigation satellite system that provides geolocation and time information to a GPS receiver in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The GPS system operates independently of any telephonic or internet reception, though these technologies can enhance the usefulness of the GPS positioning information.

The proposed system uses a GPS module to develop the geospatial tracking functionality, to increase the accuracy a different approach is implemented using a technology called Real-Time Kinematic (RTK).Two receivers are used, one of them is stationary and is called base station, the other one is wearable node. The base station measures errors, and knowing that it is stationary transmits corrections to the node. Figure 3 illustrates the differences between normal and RTK GPS after testing the positions on a path [9].

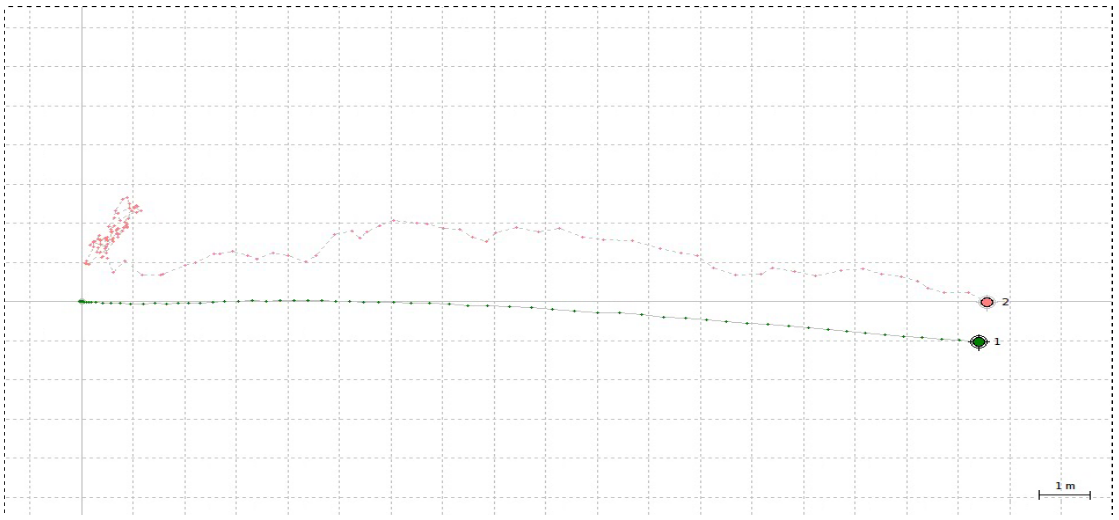


**Figure 3.** RTK vs Normal GPS

After surveying the market multiple GPS modules where found, they can be listed as the following:
- NS-HP
- Venus GPS
- GP-20U7

Each module has its own specifications thus, a comparison was conducted and the results are show in Table 4 [10-12]

**Table 4.** GPS Modules Comparison

| Specs | NS-HP | Venus | GP-20U7 |
|---|---|---|---|
| Update Rate | 1 Hz | 20 Hz | 1 Hz |
| Sensitivity | -160dBm | -165dBm | -163dBm |
| Accuracy CEP | 2.5m | 2.5m | 2.5m |
| Power | 50mA @ 3.3V | 30mA @ 3.3V | 50mA @ 3.3V |
| RTK Support | Yes | No | No |
| Antenna | External | External | Internal |
| Price | 50$ | 50$ | 78$ |

Based on modules comparisons NS-HP is selected .The NS-HP pin diagram and board description is shown in Figure 4, as well as a diagram explaining how to implement cm-level accuracy using the RTK features in the board is shown in Figure 5 [12].

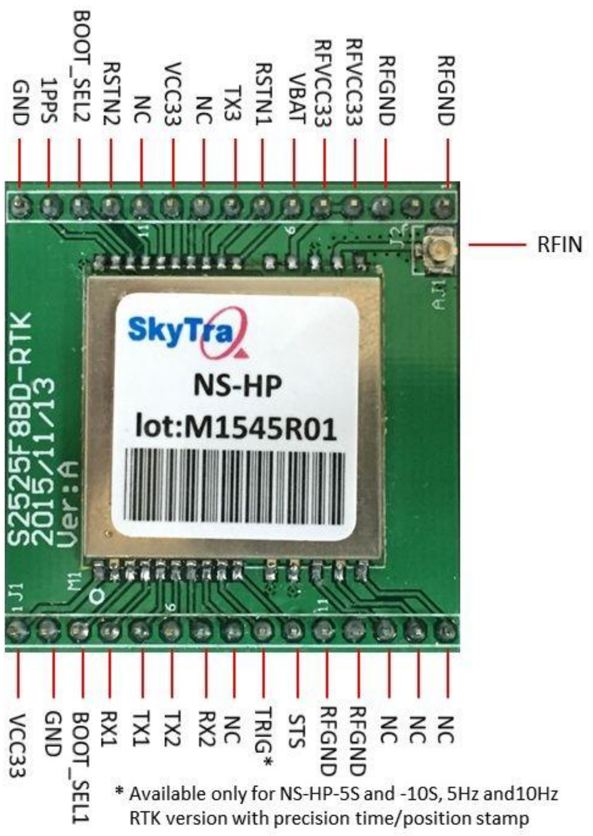


**Figure 4.** NS-HP Module

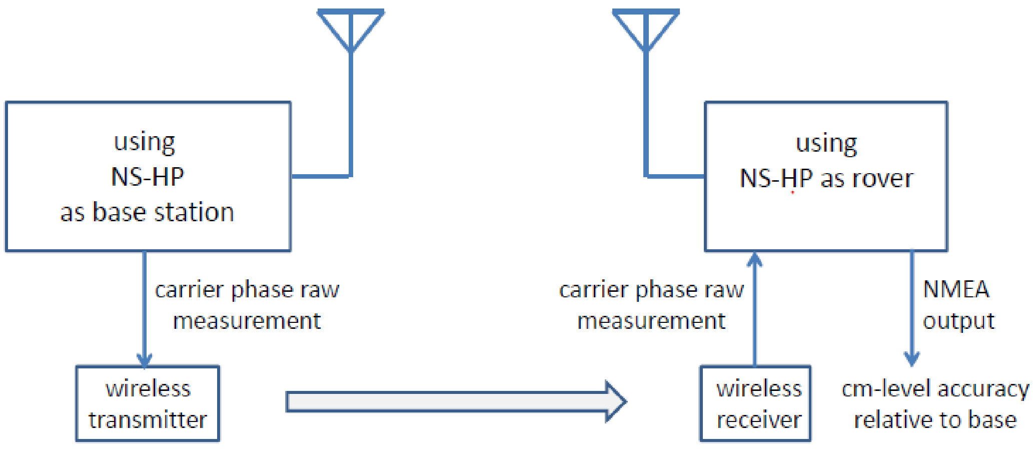

using NS-HP as base station
using NS-HP as rover
carrier phase raw measurement
carrier phase raw measurement
NMEA output
wireless transmitter
wireless receiver
cm-level accuracy relative to base

**Figure 5**. RTK Implementation Diagram using NS-HP

### 1.1.2.1. Global System for Mobile Communication Module

GSM/GPRS module is used to establish communication between a computer system and a GSM-GPRS system. Global System for Mobile communication (GSM) is an architecture used for mobile communication in most of the countries. Global Packet Radio Service (GPRS) is an extension of GSM that enables higher data transmission rate. The proposed system design uses this module as a means of wireless communication from the embedded to the server-side layer.GSM modules are also available in markets in various types, some of these types where surveyed and they are listed as the following in table 5: [13,14].

- EFCom Pro
- SM5100B

**Table 5.** GSM Modules Comparison

| Specifications | EFCom Pro | SM5100B |
|---|---|---|
| Quad-band | Yes | Yes |
| UART | Yes | Yes |
| AT Commands | Yes | Yes |
| Power Consumption | 1.5mA | 2mA |
| SIM Card Slot | Yes | No |
| Power Supply | 3.1 – 4.8V | 3.3 - 4.6V |
| Price | 31$ | 34$ |

EFCom Pro Module was selected based on market availability, cost effectiveness and Sim card support.A picture of the module is shown in Figure 6[14].

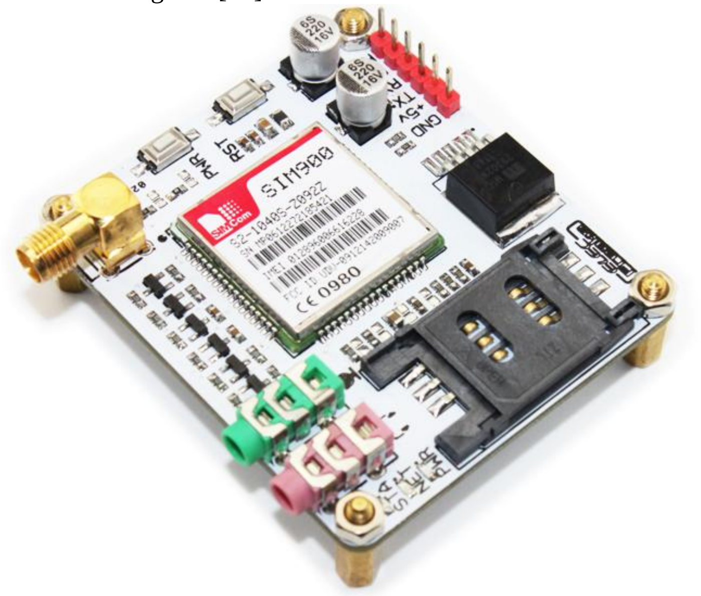


**Figure 6.** EFCom Pro GPS/GPRS Module

GPRS/GSM Module-EFCom Pro is an ultra-compact and reliable wireless module. It is a breakout board and minimum system of SIM900 Quad-band GSM/GPRS module. It can communicate with controllers via AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands). This module support software power on and reset.
EFCom Pro is based on SIM900 SoC, The main means of communication with this module is through AT Commands transferred through UART connection with the main microcontroller board carrying data to the server-side layer.

#### 1.1.2.2. Pulse Sensor

The heart rate acts as a very important factor in the system, it feeds the system with a major physiological indication for the unit wearer, the sensors reviewed uses SPO2 detection, SPO2 stands for peripheral capillary oxygen saturation, an estimate of the amount of oxygen in the blood, the reviewed sensors are : [15-17]

- Pulse Sensor SEN-11574
- Grove - Ear-clip Heart Rate Sensor

**Table 6**. Heart Sensors Comparison

| Specs | Grove | SEN-11574 |
|---|---|---|
| Input Voltage | 3 – 5.25V | 3 – 5V |
| Input Current | 6.5mA | 4mA |
| Price | 12$ | 10$ |

Based on the Coparison shown in Table 6 , the selected module was the Pulse Sensor; the module was selected based on its key features which are:

- Highly-integrated, small-size sensor
- Non-chest based heart-rate/$SpO_2$ detection
- Ultra-low power consumption

### 1.1.2.3. Radio Frequency Identification System

This part of the systems provides an extra layer of security to the embedded system layer, the RFID modules usually provide the user a means of identification using a magnetic card.The only market available model is the common RFID-RC522 Module, the modules come with magnetic cards as well. Figure 7shows a snapshot of how the module looks.

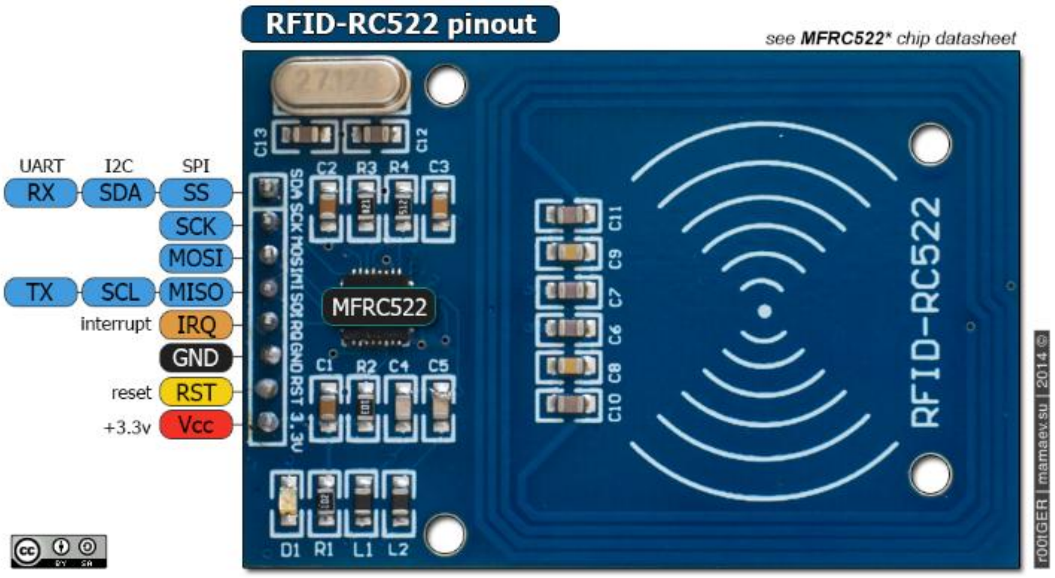


**Figure 7.** RFID-RC522 Module look

The specifications of this module are computable with the embedded systems approach:

- Module Name: MF522-ED
- Working current：13—26mA/ DC 3.3V
- Standby current：10-13mA/DC 3.3V
- sleeping current：<80uA
- peak current：<30mA
- Working frequency：13.56MHz
- Card reading distance ：0～60mm（mifare1 card）
- Protocol：SPI
- data communication speed：Maximum 10Mbit/s
- Dimension：40mm×60mm
- Environment
- Working temperature：-20—80 degree
- Max SPI speed: 10Mbit/s

### 1.1.2.4. Power Supply

To ensure the mobility of the embedded layer devices, a portable power source must be attached to each device. The Arduino board was first reviewed to decide the suitable voltages that are going to be supplied by the batteries and the method of connection. The illustration below in Figure 8 displays the possibilities of Arduino board power connection.

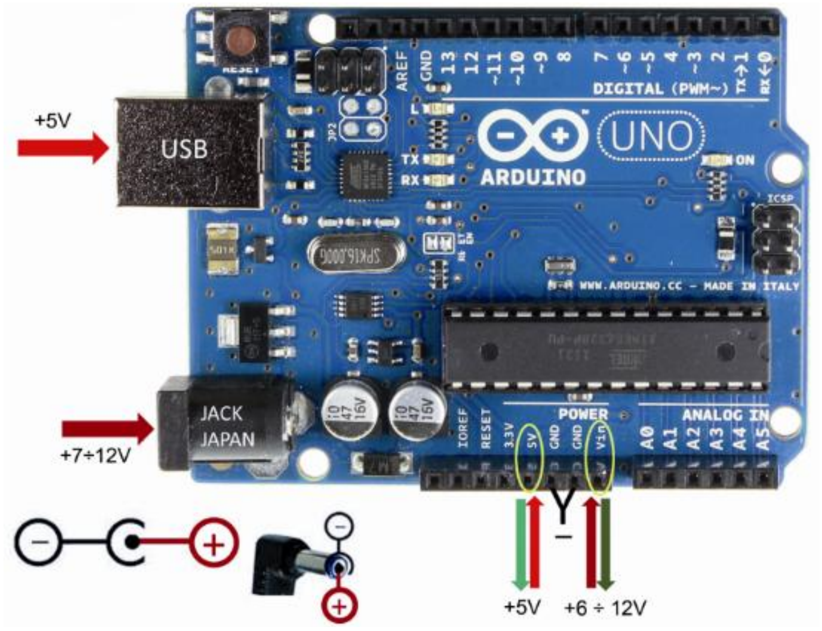


**Figure 8.** Arduino Board Power Connections

After market survey two batteries where found suitable for providing steady 5V power input to the Arduino board, they are listed as the following:[25-26]

- NiMH Rechargeable Battery (5V-1500mAh)
- Lithium polymer Battery (3.7V, 1000mAh)

Table 7 shows the batteries comparison.

**Table 7.** NiMH & LIPO Battery Specifications comparison

| Specs | NiMH | LIPO |
|---|---|---|
| Power Output | 5V @ 1500mAh | 3.7V @ 1000mAh |
| Rechargeable | Yes | Yes |
| Weight | 190g | 25g |
| Price | 4$ | 2.5$ |

The NiMH Rechargeable battery is selected as it has longer battery life and suitable size, in addition to the charger outlet.This is rechargeable NIMH (Nickel Metal Hydride) rated at 5:6 V (6 Volt when fully charged) and 1500mAh. It is a new generation of NIMH which does not suffer from memory effect [18].

### 1.1.3. Server-Side and Client-Side Layer

The server-side and client side layers form the rest of the entire system, they are responsible for handling, storing and displaying the incoming data from the embedded layer. These two layers combined together toserve the web application from which the users will gain access to the system functionality.

The embedded systems layer in the system is integrated with the server-side layer through wireless communication which is also integrated with the client-side to provide access through a dynamic web application for the end-user. Open-Source technologies are also used in the development of these layers, a web stack is utilized for the specifications of the proposed system.

Server-side refers to operations that are performed by the server in a client–server relationship in a computer network. Typically, a server is a computer program, such as a web server, that runs on remote server, reachable from a user's local computer or workstation.

The server-side technology is a part of the selected web stack, it provides a means of communication and control between the embedded system layer and the database and client side through its API.

Client-side refers to operations that are performed by the client in a client–server relationship in a computer network. A client is also a computer application, such as a web browser, that runs on a user's local computer or workstation and connects to a server as necessary.

The client-side will display the interface of the system through web browsers, its technology is also included in the web stack, mainly the user interface code will be written in HTML5, CSS3 and JavaScript languages.

A diagram explaining the operations of web software stacks is illustrated in Figure 9 below.

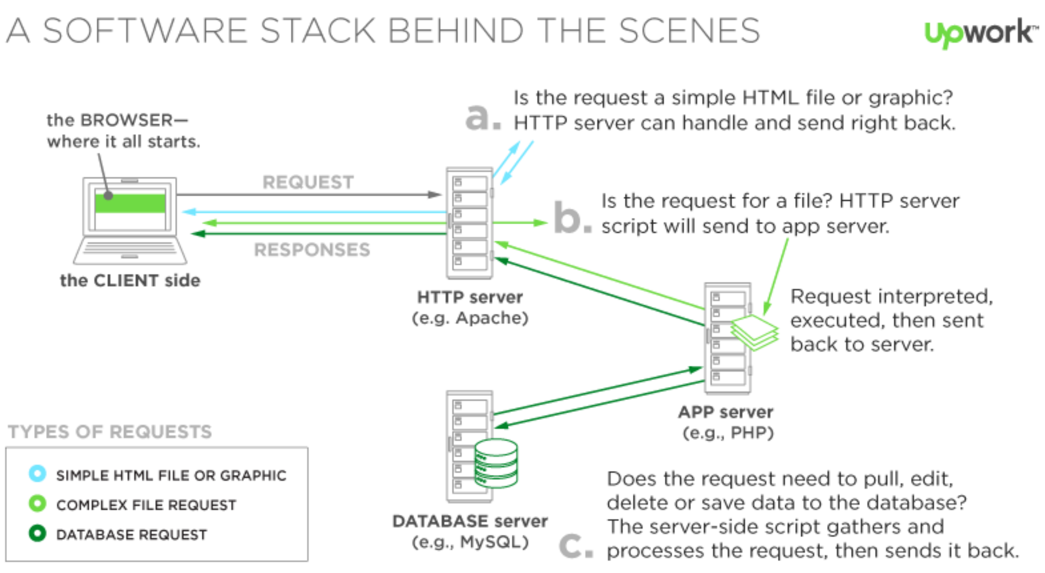

A SOFTWARE STACK BEHIND THE SCENES
upwork
the BROWSER— where it all starts.
REQUEST
RESPONSES
the CLIENT side
a. Is the request a simple HTML file or graphic? HTTP server can handle and send right back.
b. Is the request for a file? HTTP server script will send to app server.
HTTP server (e.g. Apache)
Request interpreted, executed, then sent back to server.
APP server (e.g., PHP)
TYPES OF REQUESTS
SIMPLE HTML FILE OR GRAPHIC
COMPLEX FILE REQUEST
DATABASE REQUEST
DATABASE server (e.g., MySQL)
c. Does the request need to pull, edit, delete or save data to the database? The server-side script gathers and processes the request, then sends it back.

**Figure 9.** Software Stack Operation Model

### 1.1.3.1. JavaScript
JavaScript is an object-oriented computer programming language commonly used to create interactive effects within web browsers, JavaScript will be the main language in developing the web application in addition to other styling and template languages such as XAML, HTML5 and CSS3.

### 1.1.3.2. Web Application Stack [19]
A Web stack is the collection of software required for Web development. At a minimum, a Web stack contains an operating system (OS), a programming language, database software and a Web server.Different stacks technologies are available, here are some examples:

- QuorraJS
- Meteor
- Derby
- Adonis’s
- Sails
- SANE
- MEAN.io

### 1.1.3.3. Meteor Platform Specifications[20]
The Meteor platform or framework is selected mainly because of its large community and developer base. Thus, this will ensure that any problems faced during development can be easily solved due to the vast amount of knowledge out there about the platform.

Command-line tool: The command line tool is the kernel that administrates the stack .It also has a package system built into it called Isobuild, an isomorphic package system that allows you to easily install packages via Atmosphere, NPM, and Cordova plugins.

Server: The server is a Node.js app built with certain libraries in place to make the communication happen over DDP and EJSON to the front-end:

- **Node.js** - A JavaScript server.
- **Connect** - A library to output http responses from an app
- **Database Driver (Mongo)** - A simple drive to interface with MongoDB data.
- **Livequery**- A library built to query and stream out Mongo data in a reactive way.
- **Fibers/Futures** - A wrapper library for Node.js, making it synchronous in an effort to reduce 'callback spaghetti'.

Communication Layer: The communication layer is the real magic that binds the client and server together. EJSON is used to serialize and de-serialize data moving across the wire via DDP.

- **DDP (Distributed Data Protocol)** - A protocol for sending data over web sockets.
- **EJSON** - An extension of JSON to support serializing more data types like Dates and Binary.

Browser: The browser part of the platform is sent over with minimal html and some JavaScript that loads up the environment.

A lot of codesare based onjQuery and underscore.js as the foundation. While the server is synchronous, browsers and JavaScript are asynchronous by nature. Let's look at the libraries that help make up the client and its reactive nature:

- **Tracker** - The backbone of the reactive front-end. It is the reactive 'glue' for any tracker aware libraries you build.
- **Spacebars** - A derivation of Handlebars, built to be reactive.
- **Blaze** - A reactive library built to marry Tracker & Spacebars up to create live updating user interfaces. Similar to Angular, Backbone, Ember, React, Polymer, or Knockout - just easier.
- **Minimongo** - A client side mongo library that synchronizes data over DDP and allows the client to reactively consume mongo data.
- **Session** - A library to handle reactive UI state variables, nothing like a session in Rails, PHP or Node.js.

### 1.1.3.4. Meteor Data on the Wire

The concept of Data on the Wire is very simple and in tune with the nested Model View Controller ( MVC) pattern; instead of having a server process everything, render content, and then send HTML across the wire, it sends the data across the wire and let the client decide what to do with it.

This concept is implemented in Meteor using the Distributed Data Protocol, or DDP. DDP has a JSON-based syntax and sends messages similar to the REST protocol. Additions, deletions, and changes are all sent across the wire and handled by the receiving service/client/device. Since DDP uses WebSockets rather than HTTP, the data can be pushed whenever changes occur.

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and the WebSocket API in Web IDL is being standardized by the W3C.

DDP provides a generic nature of communication. It doesn't matter what kind of system sends or receives data over DDP—it can be a server, a web service, or a client app—they all use the same protocol to communicate. This means that none of the systems know whether the other systems are clients or servers. With the exception of the browser, any system can be a server, and without exception, any server can act as a client. All the traffic looks the same and can be treated in a similar manner.Figure 10 illustrates how the protocol operates.

In other words, the traditional concept of having a single server for a single client goes away. You can hook multiple servers together, each serving a discreet purpose, or you can have a client connect to multiple servers, interacting with each one differently[20].

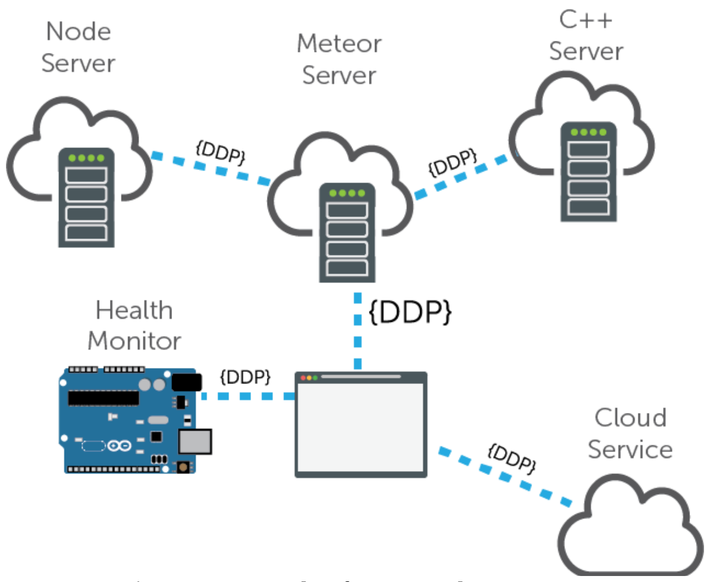


**Figure 10.** Sample of DDP implementation

**1.1.3.5. Meteor latency compensation**

Meteor employs a very clever technique called Mini Databases. A mini database is a "lite" version of a normal database that lives in the memory on the client side. Instead of the client sending requests to a server, it can make changes directly to the mini database on the client. Figure 11and 12 show how the design works.

This mini database then automatically syncs with the server using DDP, which has the actual database, for this Meteor uses MongoDB and Minimongo.

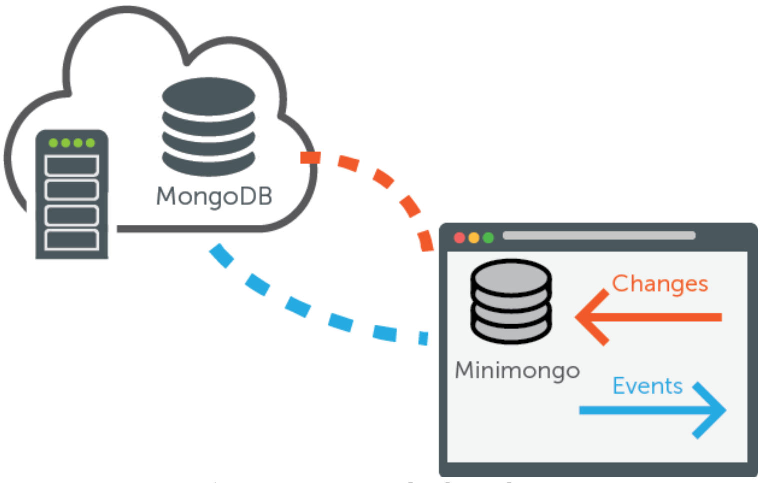


**Figure 11**. Meteor database driver

When the client notices a change, it first executes that change against the client-side Minimongo instance. The client then goes on its merry way and lets the Minimongo handlers communicate with the server over DDP. If the server accepts the change, it then sends out a "changed" message to all connected clients, including the one that made the change. If the server rejects the change, or if a newer change has come in from a different client, the Minimongo instance on the client is corrected, and any affected UI elements are updated as a result.

The entire process is asynchronous, and it's done using DDP. This means that the client doesn't have to wait until it gets a response back from the server. It can immediately update the UI based on what is in the Minimongo instance. What if the change was illegal or other changes have come in from the server? This is not a problem as the client is updated as soon as it gets word from the server.

In cases of a slow internet connection or connection going down temporarily, a normal client/server environment, doesn't allow changes to be done, or the screen would take a while to refresh while the client waits for permission from the server. However, Meteor compensates for this. Since the changes are immediately sent to Minimongo, the UI gets updated immediately.

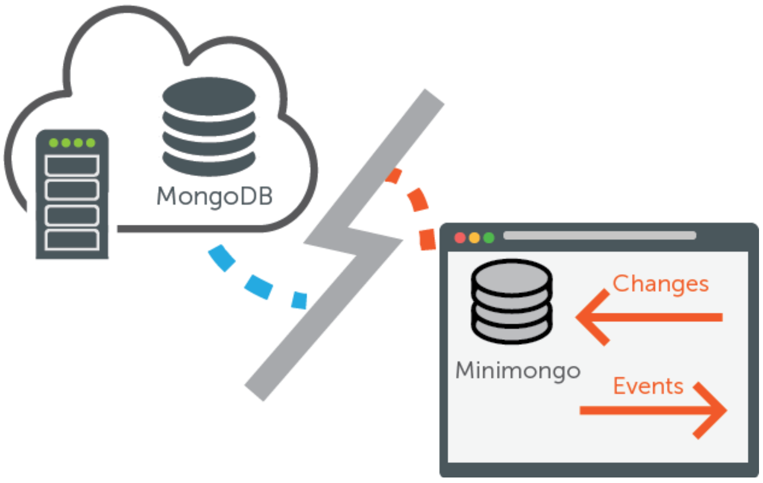


**Figure 12.** Meteor Data driver at bad connections

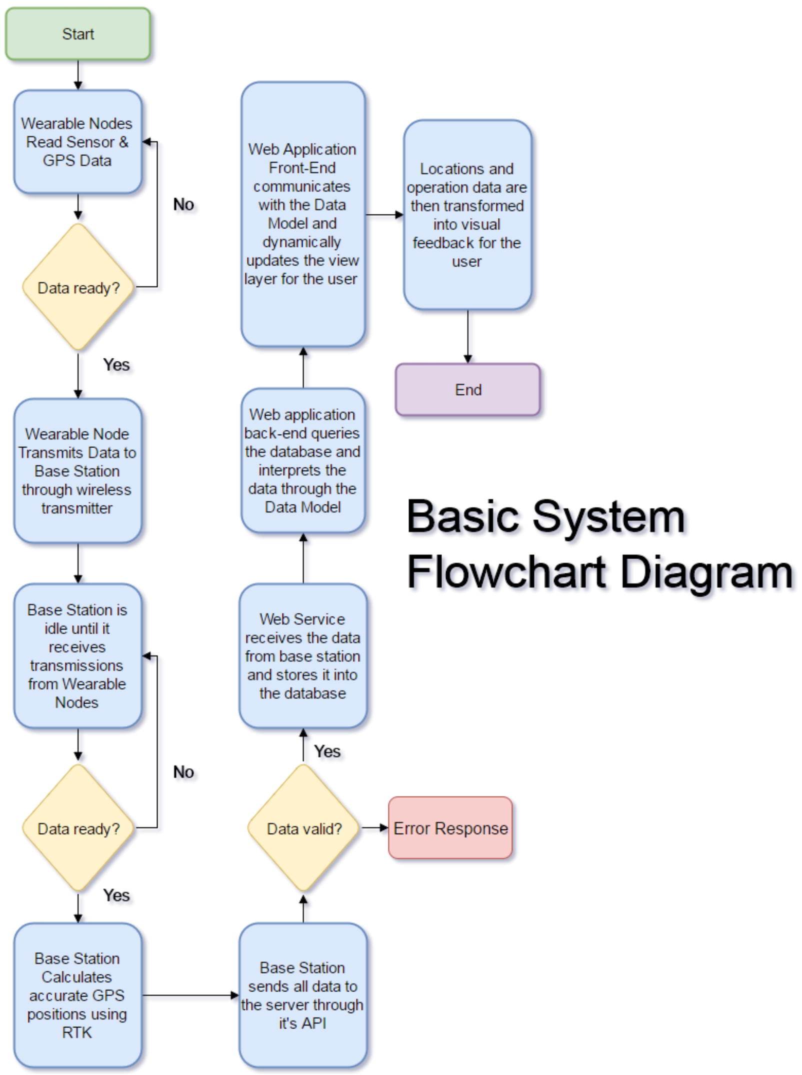
All the changes madegets reflected in the UI, based on the data in Minimongo. When the connection comes back, all the queued changes are sent to the server, and the server will send authorized changes to the client[20].

### 1.1.3.6. Meteor Full Reactivity

Reactivity is integral to every part of Meteor. On the client side, Meteor has the Blaze library, which uses HTML templates and JavaScript helpers to detect changes and render the data in the UI. Whenever there is a change, the helpers re-run themselves and add, delete, and change UI elements, as appropriate, based on the structure found in the templates. These functions that re-run themselves are called reactive computations[20].

### 1.1.4. Main Flow chart diagram for RCMSS

Figure 13shows the core system flow Chartfor the Proposed System operation.



Start
Wearable Nodes Read Sensor & GPS Data
Data ready?
No
Yes
Wearable Node Transmits Data to Base Station through wireless transmitter
Base Station is idle until it receives transmissions from Wearable Nodes
Data ready?
No
Yes
Base Station Calculates accurate GPS positions using RTK
Base Station sends all data to the server through it's API
Data valid?
Error Response
Yes
Web Service receives the data from base station and stores it into the database
Web application back-end queries the database and interprets the data through the Data Model
Web Application Front-End communicates with the Data Model and dynamically updates the view layer for the user
Locations and operation data are then transformed into visual feedback for the user
End
Basic System Flowchart Diagram

**Figure 13.** Flow Chart Diagram

## 4.4 System Design

The embedded systems layer was implemented using a set of components that were integrated together using the Universal Asynchronous Receiver/Transmitter (UART) serial communication method. Figure14 shows simple circuit diagram illustrates the connection across the different components of the proposedsystem.

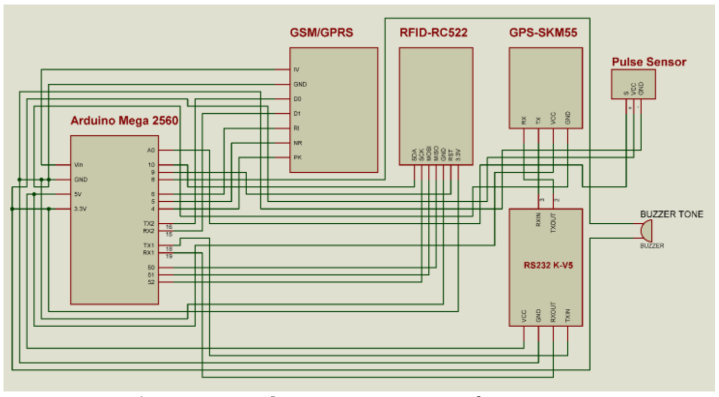


**Figure 14.** Schematics Diagram for RCMSS

The goal is to implement the discussed design producing two prototypes, a breadboard porotype purposed to illustrate the connections between the different system modules and a final prototype enclosed in a plastic case that can be attached to the unit trousers belt or body armor.

The hosting architecture intended to be used as the cloud ecosystem for the web application was chosen to be the Virtual Private Server infrastructure as it provides on demand scaling capabilities and of course it is cheaper than renting a dedicated server as well as providing much powerful recourses than using a shared hosting plan.

The proposed platform is intended to be deployed on specific type of Virtual Private Server (VPS ) which is cloud-based VPS, it shares the same operations as the regular VPS by sharing the resources and utilizing a virtual container on top of the original machine however the difference is that it draws its resources from a pool of server clusters not a single physical machine. After studying the available commercial options, the Ubuntu Linux operating system was selected to be the operating system running behind the cloud-based VPS.

## 4.5 System Implementation

The embedded systems implementation can be sectioned into two parts, the first part is implementing both hardware circuitry and physical connections, the second part is developing the software functionality that operates the different hardware parts which heavily relies on C and C++ programming languages.

To be able to connect all the functionalities in the system, two routines are designed .A setup routine that runs once and a main routine that keeps the system in a finite loop of operation. The setup routine simply calls the initialization functions of all the different modules in the system.

Meanwhile, the Main routine is designed to first authenticate the user then continuously communicate the current device data to the API web service. Figure 15 illustrates how the main routine operates in a simple flowchart.

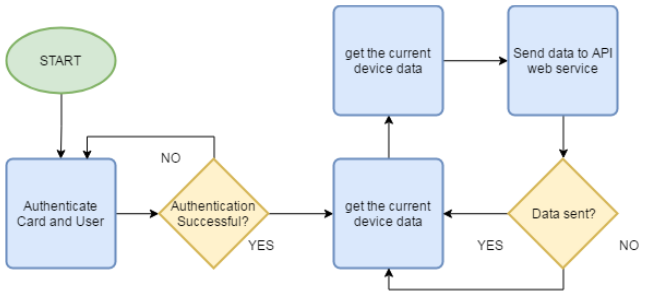

START
Authenticate Card and User
NO
Authentication Successful?
YES
get the current device data
get the current device data
Send data to API web service
Data sent?
YES
NO

**Figure 15.** Embedded System Layer Main routine flowchart

The server side and client side implementation can be subdivided into three sections, front-end which resides in the client side, back-end and API web service which both reside in the server side.

The API acts as a key element that is responsible for the integration between embedded systems layer and the server and client side layer. The API responds to HTTP requests coming from the embedded systems layer by executing CRUD operations on the MongoDB. Create, Read, Update and Delete operations(CRUD) each responds to a specific API call.

The first attempt to develop the API was to use the existing Distributed Data Protocol (DDP) by establishing a connection from the Arduino Mega serving as the client, however some limitations appeared as the GPRS network doesn't allow for bidirectional communication due to some restrictions enforced by

Vodafone network. As a means to overcome this limitation another approach for developing a solid API is to rely on RESTful communication, which is an advanced implementation over the regular HTTP protocol.

Building a RESTful API as part of the Meteor stack is not a very common task as the framework heavily relies on DDP communication, however after some research open source alternatives where found, a community backed package called Restivus provides a perfect integration with the meteor stack. The API runs only on the server side, the structure of the API is designed with two main endpoints, GET and POST.

When dealing with embedded systems, it's always a necessity to validate whether the device is still online and connected or not, for this purpose a special algorithm was designed to simulate the heartbeat of the human heart.

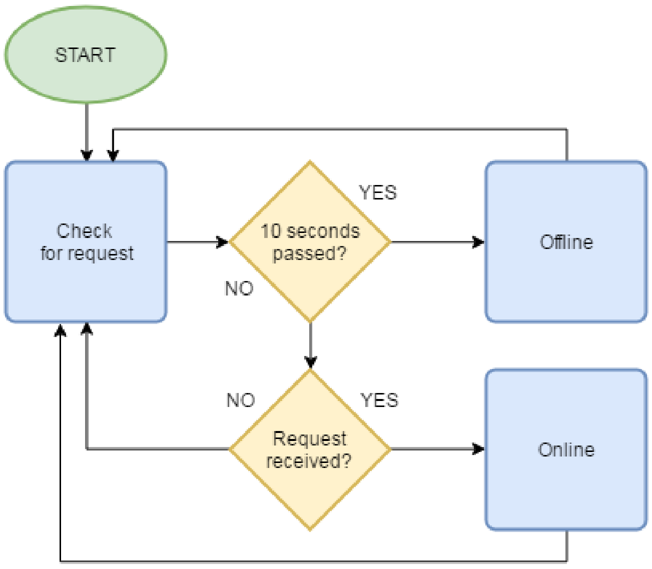


**Figure 16.** heartbeat algorithm flowchart

The algorithm simply requires the API to expect a request every 10 seconds from a device to maintain its online presence, unless a request is received within this interval the algorithm will update the users record to reflect a false online presence. Any request to any endpoint will be sufficient enough for the algorithm to maintain an online presence of the designated device. The algorithm flowchart is illustrated in Figure 16 This approach is implemented using the Meteor framework and browser interval and timeout features.

The authentication endpoint is very simple and vital to the system, it acquires the card identification string that is requested from the device through the request parameters, searches the database user collection for a record match and finally responds back with a Java Script Object Notion (JSON) that consists of a status of the authentication that indicates whether the card identification string was found in the database records or not. The Application Programming Interface (API) endpoint can be accessed through the URL /api/auth.

These endpoints are responsible for updating user records by which the entire system reacts and adapts. These endpoints require the Hyper Text Transfer Protocol (HTTP) request to be overloaded with an authenticated user identification string. The endpoints designed with this algorithm are:

- /api/location: sets the location latitude and longitude values
- /api/bpm: sets Beats Per Minute(BPM) values for the user
- /api/hbw: sets the Boolean value for the Heartbeat failure warning
- /api/sos: sets the Boolean value for the Sensor Observation Service (SOS) warning.

These endpoints then use the request user identification string and searchthe database user collection for the matching user record, updates the designated endpoint operation value and finally responds with a Java Script Object Notation(JSON) that contains the results of the API call.

## V. System Commissioning and Testing

The first prototype is based on breadboard connection; Figure 16 shows the final integration of the prototype.

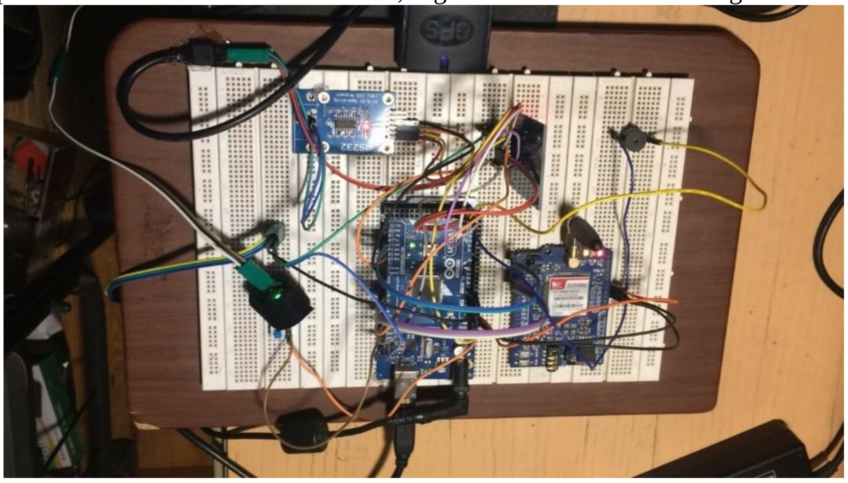


**Figure 177.** Breadboard prototype

### 5.1 Observable Class

The observables are the most important feature in the implementation, it enables the web application to become aware of the data changes that are happening to its database. Observables are native MeteorJS classes, they drive the main reactivity behind the map and location tracking.
Figure 18 shows a flowchart that explains how the algorithm works.

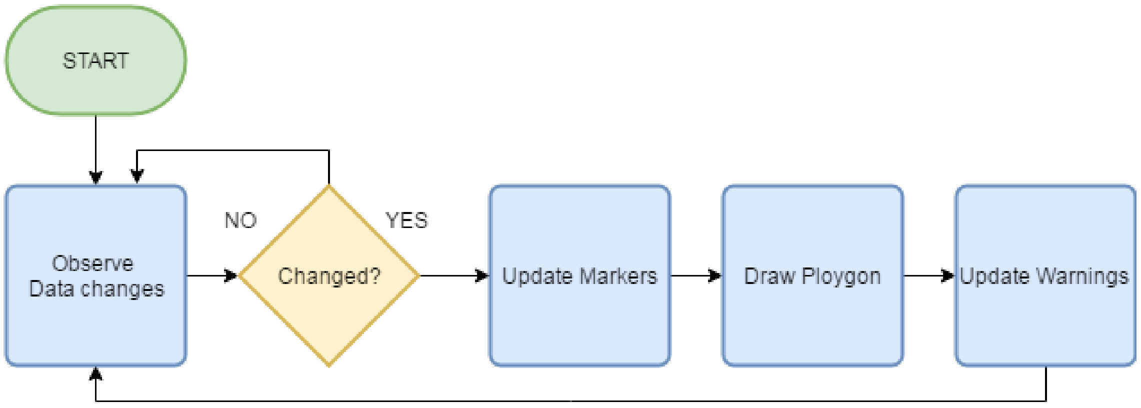


**Figure 18.** Observables flowchart

### 5.2 Generating Markers

Markers are the icons that show on the map to indicate that a deployed unit is at a certain location, the marker icon represents the rank of the unit. Figure 19 shows a marker example for an officer rank unit.

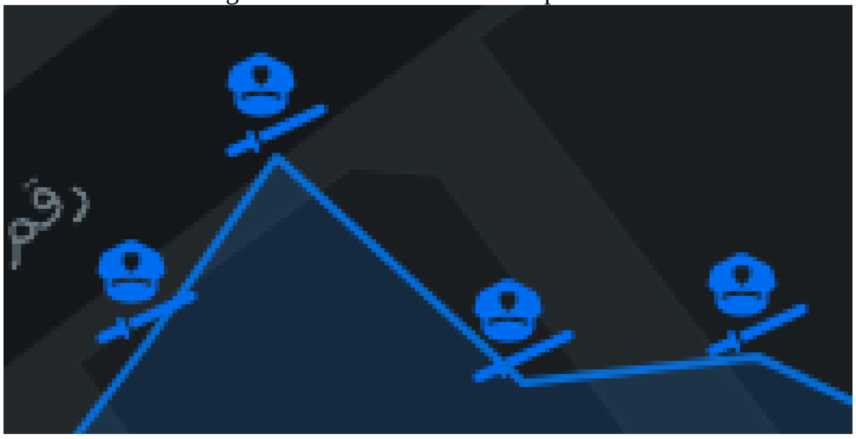


**Figure19.** Markers sample

The markers are generated using the Google Maps API, a marker creation function is overloaded with some settings but most importantly the location longitude and latitude values.

When the marker values are set, its graphical representation changes as well on the rendered map, thus yielding the movement of the trackers.

### 5.3 Polygon graphs

The polygon graphs main objective is to illustrate the deviation of the units from their initial formation, polygons are also generated using Google Maps API.The creation function is overloaded with an array of positions that is generated from the positions of the moving unit data model. Figure 20 shows a generated polygon from a deployed formation.

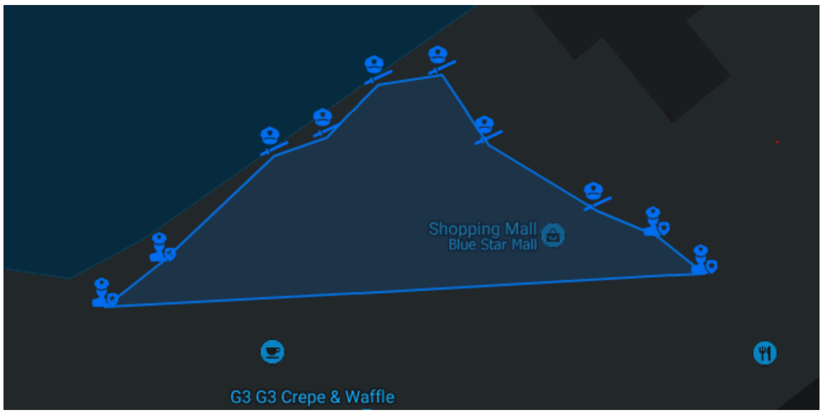


**Figure 20.** Polygon graph of a formation

### 5.4 User authentication and account management

The user login module provides the system a layer of security, each user is registered by administration generating a login email and password that are used to authenticate the user to the web application. The accounts management can be executed through manipulating the user collection from the database. Figure 21 shows the login page which is the entry point to the web application.

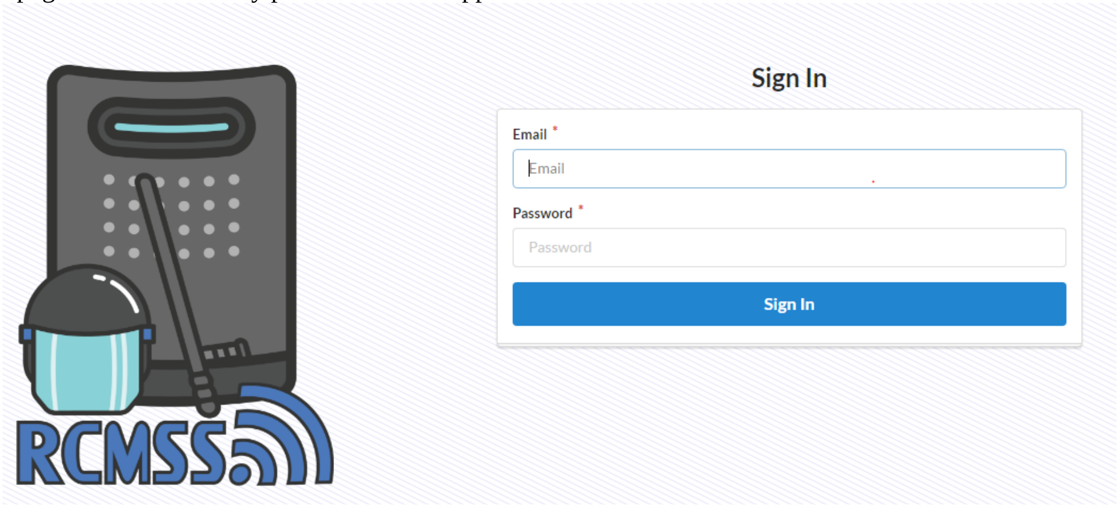


**Figure 21.** Sign in template rendered

### 5.5 Rating

A rating module was developed to provide visual feedback and sorting functionality for the system units. The rating control populates using the data extracted from the data model of the database. Figure 22 shows a snap shot of a rating control rendered in the deployed unit template.

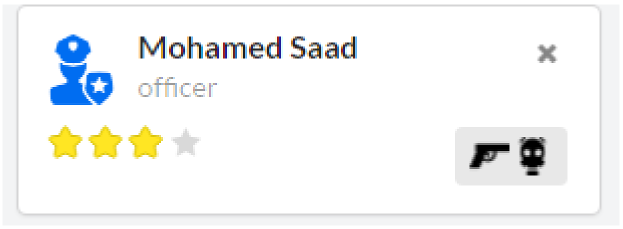


**Figure 182.** Rating control

### 5.6 Simulation

To demonstrate the system's full capability, a simulation module was designed and developed using server side timers, this simulation handles the entire deployment by changing the positions and states of the units using a random pattern. Multiple functions where developed to enable a smooth simulation, these functions are called through server side methods.

"startSim": starts the simulation
"stopSim": stops the simulation
"allOffline": sets all users to offline state
"allOnline": sets all users to online
"rate":sets the rating for the users.

#### 1.1.5. Navigation Template

The navigation template is designed to serve as a navigation menu fixed on the top of the view to help the user access different parts of the web application at ease.It is composed of navigation buttons, system logo and a sign out button, Figure 23shows the rendered navigation menu.

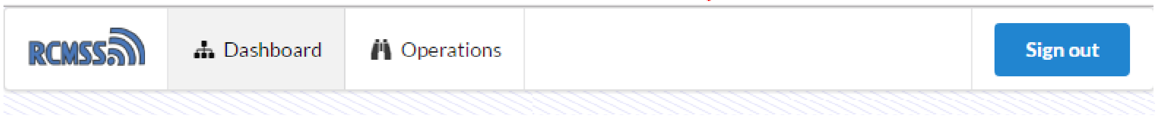


**Figure 23.** Navigation template

#### 1.1.6. Dashboard Template

The dashboard template acts as a key view in the web application as it allows the user to manage the available and deployed units, Figure 24 shows dashboard view.

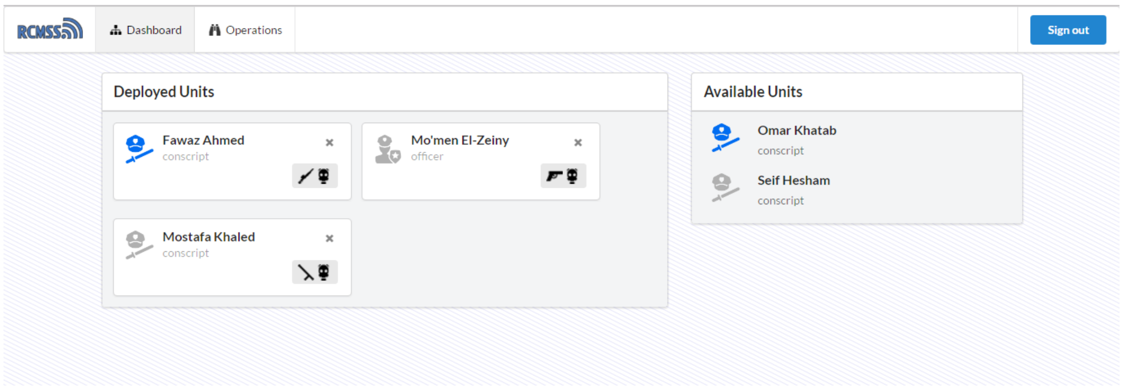


**Figure 194.** Full dashboard view

The dashboard is constructed using nested templates, Deployed Units and Available Units. Each template has its features and roles in the system.

The Available Units template displays the current units connected to the system and their status, it automatically reacts based on the user collection in the database, aided by a blaze template it renders each user document as a single item in the template. Figure 25 shows the rendered template.

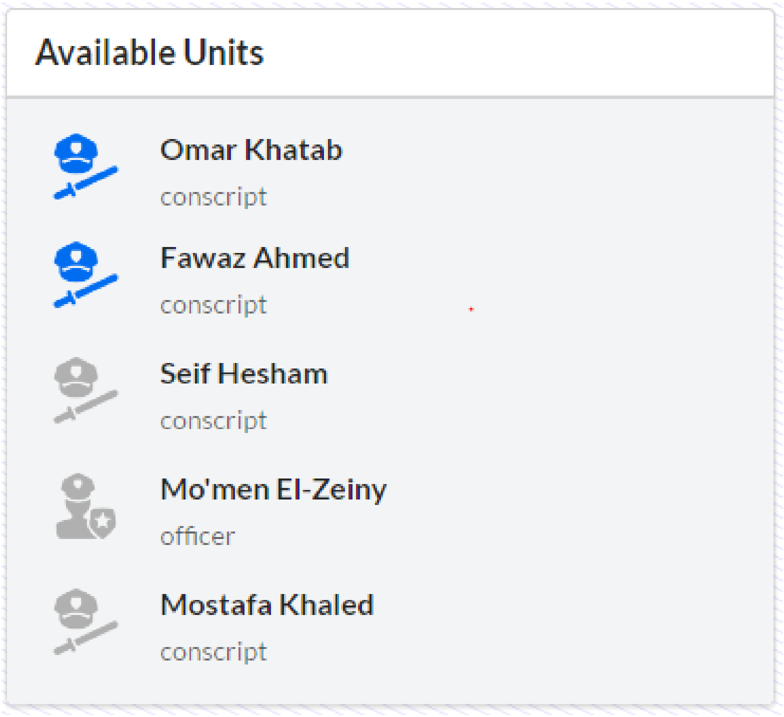


**Figure 205.** Available Units Template

The other template that forms the main Dashboard is the Deployed Units Template,it is responsible for managing the deployment of the units. The control was designed to be simple and efficient, a simple drag and drop from the Available Units segment to the Deployed Units segment activates the deployment and adds this unit to the monitor. Figure 26 shows the drag and drop control.

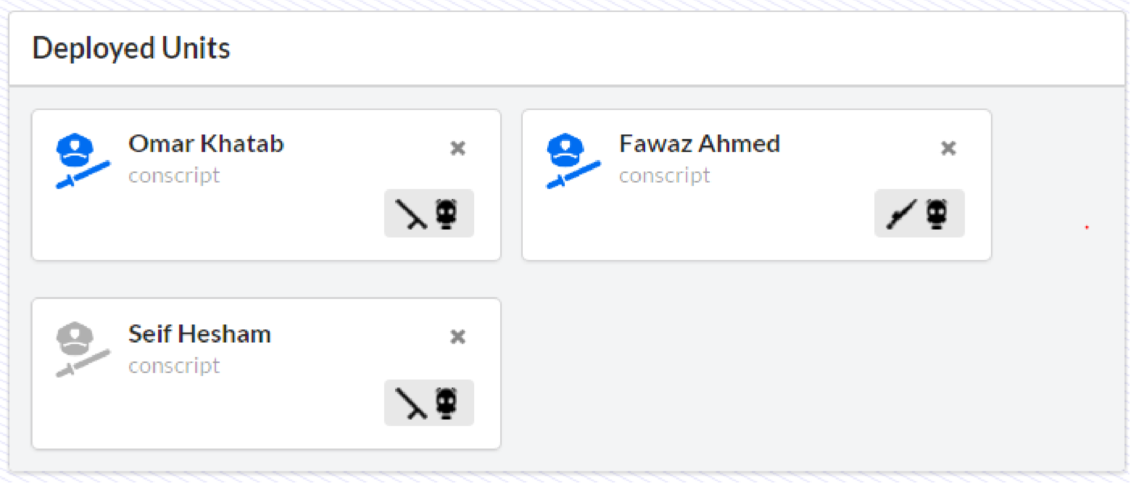
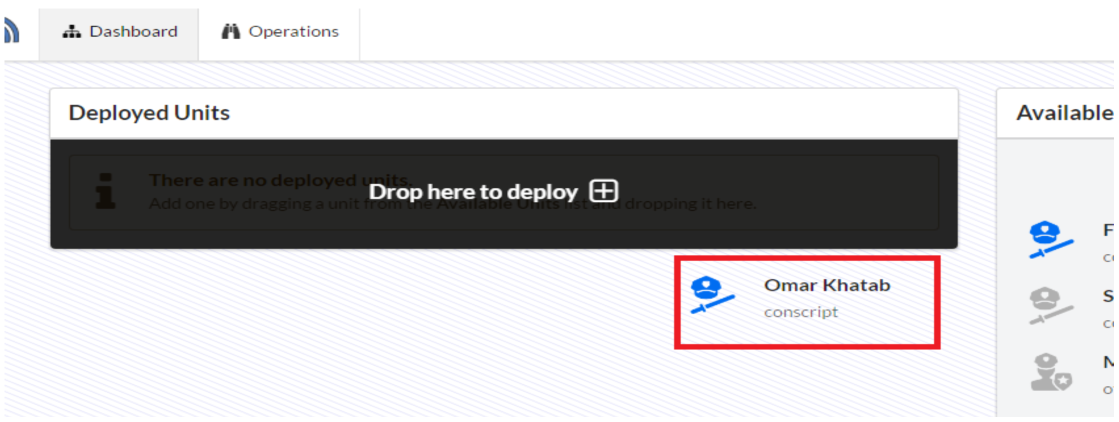


**Figure 216.** Deployed Units Template

The segment also adapts to the dropped unit and renders more information such as arming types and can also render other information as well that might help the users. Figure 27 shows a full snapshot of the deployed unit segment with the rendered deployed users.



There are no deployed units.
Add one by dragging a unit from the Available Units list and dropping it here.

**Figure 27.** Drag and Drop control

**Operations Template**

After the user selects the deployed units from the available units, the next template takes control which is the operations template. The operation template is responsible for rendering the most critical part of the system which is the map controls and warnings system. Figure 28 shows a snapshot of the complete operations view.

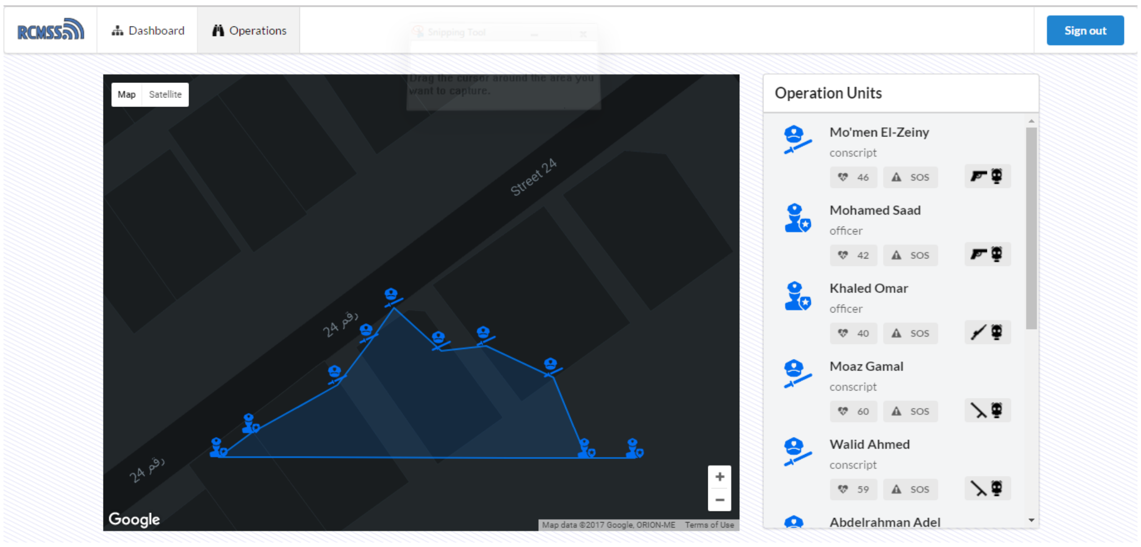


**Figure 28.** Operations View

The operations view has multiple functionalities, it contains two nested templates, the map template and the operation units.

The map template renders a Google Map API, which is modified specifically to serve the objectives of the system, it materializes the GEO location data received from the Wearable Nodes into tracker markers that update automatically based on the data model. Figure 29 shows a rendered map template tracking deployment of three operation units one of them has a heartbeat warning.

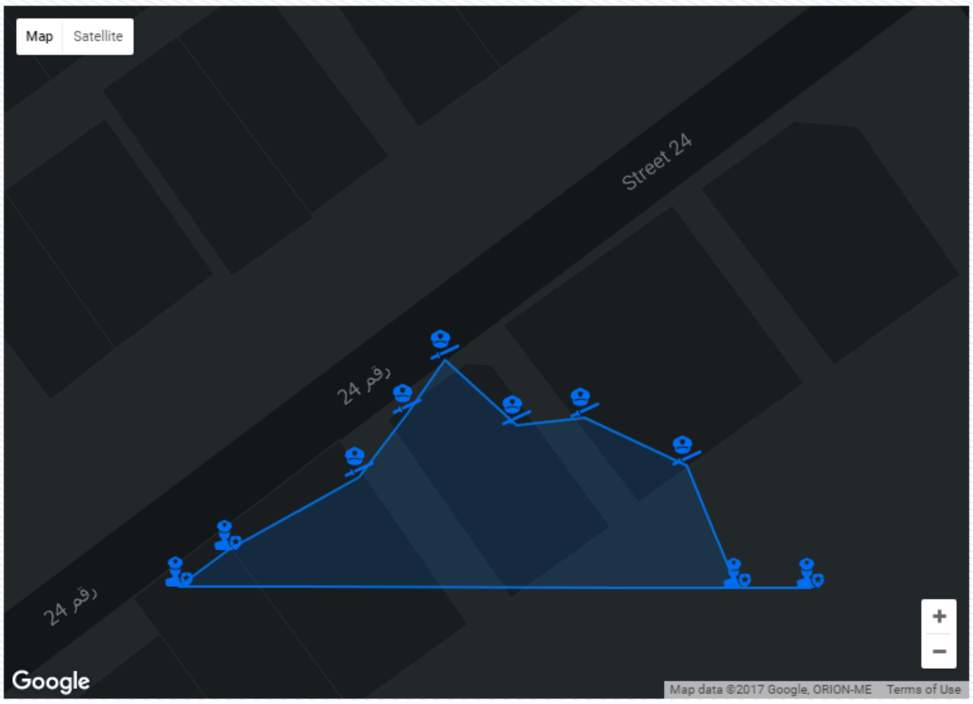


**Figure 29**. Rendered map template

The Operation Units template renders the operation units, providing visual feedback that enables the user to monitor the beat per minute. Figures 30 and 31 show a rendered operation unit and how the warnings and heart beat per minute is displayed.

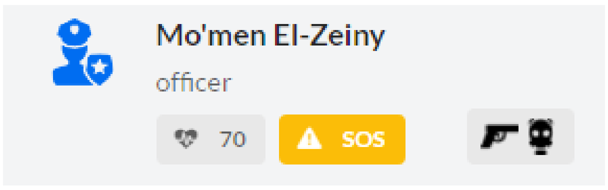


**Figure 30.** SOS warning

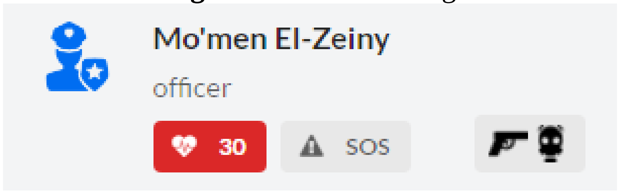


**Figure 31.** heart beat warning

## VI. Cost Analysis

This section previews the calculations that were made to review the cost effectiveness of the proposed system and set an expected budget as well. The analysis was held on each part independently across the different layers of the system as shown in table 8 and table 9.

### 6.1 Wearable Node

**Table 8.** Wearable Node Budget

| Component | Cost/Unit | Economical Cost/100 Units |
|---|---|---|
| Arduino Mega 2560 Board | 13$ | 11$ |
| EFCom Pro GSM/GPS | 30$ | 30$ |
| SKM55RM | 12$ | 10$ |
| PulseSensor | 10$ | 8$ |
| NIMH Battery | 4$ | 3.8$ |
| Total | 69$ | 62$ |

### 6.2 Web Application

**Table 9.** Web Application Budget

| Service | Subscription Cost |
|---|---|
| Meteor Platform | Open-Source |
| Cloud Hosting | Free Plan |
| Domain | Free Plan |
| Total | 0$ |

## VII. Conclusion and Future work

### 7.1 Conclusion

The riot control sector faces major challenges and several problems have been identified, these problems have direct effects towards causing potential loss of lives, slow response, management difficulties, inaccurate decisions and inability of future improvements. The challenges are as follow:

- No real-time health monitoring
- No real-time position and movement tracking
- Insufficient operation data logging and management
- Lack of Operation Management System

The main objective of the system is design and implement a system that ensures seamless operation command and increases the operational capabilities of the maneuver units from battalion to platform/single conscript level, such a system has not been developed before and after review of literature it became clear how the system was architected, designed and implemented to impact the riot control sector nationally and internationally.

### 7.2 Future Work

The Riot Control Management and Support System provides a basic solution that utilizes state of art technologies, however, more research has to be conducted in order to provide a more complete organization ready solution, the proposed future work could possibly address the following parts of the system:

1. Reduce the device size
2. Fabricate the components instead of relying on market available components
3. Optimize the device power consumption
4. Increase the accuracy of the GPS by either investing in a better module or RTK technology
5. Provide a more accurate heart biometrics rather than market available options
6. Provide a paid cloud infrastructure rather than free one used
7. Develop an administration module responsible for data entry of relational data
8. Develop analysis module to work with the stored location data

## References:

[1]. Library of Congres, 5th ed., H. C. Metz, Ed., Library of Congress Cataloging-in-Publication Data, 1990, pp. 140-400.

[2]. S. Bibi, U. Nazir and M. Sishar, "Analytical Survey for Assuring Quality Standards in GPS based Tracking System," *International Journal of Computer Applications (,* vol. 111, no. 8, pp. 30 - 37, 2015.

[3]. J. Dineshkumar and R. S. Sanja, "Real Time Tracking and Health Monitoring of Soldiers using ZigBee Technology: a Survey," *International Journal of Innovative Research in Science,Engineering and Technology,* vol. 4, no. 7, pp. 5560 - 5575, 2015.

[4]. S. M. Pangavhane, S. Choudharyand . B. Pathak, "Real Time Soldier Tracking System," *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE),* pp. 21-24, 2015.

[5]. Luciad, "END-TO-END GEOSPATIAL SOLUTIONS," Luciad, 2016. [Online]. Available: www.luciad.com.

[6]. R. Arunthavanathan and K. P. Navod, "VEHICLE MONITORING CONTROLLING AND TRACKING SYSTEM BY USING ANDROID APPLICATION," *International Journal of Technical Research and Applications,* vol. 4, no. 1, pp. 114-119, 2016.

[7]. S. Supriya and P. Shivani, "Pilgrim Tracking and Health Monitoring System," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 4, no. 12, pp. 172-175, 2015.

[8]. Arduino, "Arduino Mega," Arduino, 2016. [Online]. Available: https://www.arduino.cc/en/Main/arduinoBoardMega.

[9]. Emild, "RTK GPS DEMONSTRATION," Emild, 2016. [Online]. Available: https://emlid.com/rtk-gps-demonstration/.

[10]. SparkFun, "GPS Receiver - GP-20U7 (56 Channel)," SparkFun, 2016. [Online]. Available: https://www.sparkfun.com/products/13740.

[11]. NavSpark, "NS-HP : RTK CAPABLE GPS/GNSS RECEIVER," NavSpark, 2016. [Online]. Available: http://navspark.mybigcommerce.com/ns-hp-rtk-capable-gps-gnss-receiver/.

[12]. SparkFun, "GSM/GPRS Module - SM5100B," SparkFun, 2016. [Online]. Available: https://www.sparkfun.com/products/9533.

[13]. ElecFreaks, "GPRS/GSM Module-EFCom Pro EFComPro," ElecFreaks, 2016. [Online]. Available: http://www.elecfreaks.com/store/gprsgsm-moduleefcom-pro-efcompro-p-450.html.

[14]. Seed, "Grove - Ear-clip Heart Rate Sensor," Seed, 2016. [Online]. Available: https://www.seeedstudio.com/Grove-Ear-clip-Heart-Rate-Sensor-p-1116.html.

[15]. SparkFun, "Pulse Sensor," SparkFun, 2016. [Online]. Available: https://www.sparkfun.com/products/11574.
[16]. Maxim Integrated, "MAXREFDES117#: HEART-RATE AND PULSE-OXIMETRY MONITOR," Maxim Integrated, 2016. [Online]. Available: https://www.maximintegrated.com/en/design/reference-design-center/system-board/6300.html.
[17]. Future Electronics, "Lithium Polymer Battery (7.4v - 1000 mAH)," Future Electronics, 2016. [Online]. Available: http://store.fut-electronics.com/products/lithium-polymer-battery-7-4v-1000-mah.
[18]. Future Electronics, "NiMH Rechargeable Battery (5V-1500mAh)," Future Electronics, 2016. [Online]. Available: http://store.fut-electronics.com/products/nimh-rechargeable-battery-5v-1500mah.
[19]. C. Wodehouse, "Choosing the Right Software Stack," UpWork, 2016. [Online]. Available: https://www.upwork.com/hiring/development/choosing-the-right-software-stack-for-your-website/.
[20]. Meteor, "Application Structure," Meteor, 2016. [Online]. Available: https://guide.meteor.com/structure.html.

SherifKamel Hussein Hassan Ratib. "Newly Proposed Prototype Model for Riot Control Monitoring and Support System." *IOSR Journal of Computer Engineering (IOSR-JCE)*, 22.1 (2020), pp. 08-30.