ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

INDOOR LOCATION SYSTEM FOR MOBILE PHONES BASED ON BLUETOOTH LOW ENERGY ARRAYS

Autor: José Luis Martín Casarrubios

Director: Carol Davids

Chicago

June 2018

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Indoor Location System for Mobile Phones based on Bluetooth Low Energy Arrays

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2017/18 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

[illegible]

Fdo.: José Luis Martín Casarrubios Fecha: 19 / 08 / 2018

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Carol Davids

Fdo.: Carol Davids Fecha: 19 / 08 / 2018

de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- ➢ La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- ➢ La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- ➢ La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- ➢ La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 23 de agosto de 2018

ACEPTA [illegible]

Fdo. José Luis Martín Gascuñbias

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

INDOOR LOCATION SYSTEM FOR MOBILE PHONES BASED ON BLUETOOTH LOW ENERGY ARRAYS

Autor: José Luis Martín Casarrubios
Director: Carol Davids

Chicago

June 2018

Acknowledgment

I would like to thank my project director at Illinois Institute of Technology, Carol Davids, - “Director, Real-Time Communications Lab of Illinois Institute of Technology ”for his support and guidance in this work. She gave me the opportunity to become a part of her group as a volunteer to make this project.

Thank you to my friend and roommate Alvaro Sanchez- “Master of Science in Computer Engineering”, for all the support and the ideas he gave me when I was stuck. I will never forget those long nights working together to overcome some difficult situations. Thank you for making my stay more enjoyable in Chicago.

Lastly, I would like to acknowledge with gratitude the constant encouragement of my family, who have made my studies in Chicago possible. Thank you for your effort.

INDOOR LOCATION SYSTEM FOR MOBILE PHONES BASED ON BLUETOOTH LOW ENERGY ARRAYS.

Autor: Martín Casarrubios, José Luis.
Director: Davids, Carol.
Entidad Colaboradora: IIT Real-Time Communications Lab

RESUMEN DEL PROYECTO

El proyecto en cuestión ofrece una variedad de soluciones para el problema de localización de interiores en llamadas de emergencia y para mejorar la seguridad del campus en el *Illinois Institute of Technology* mediante el desarrollo de cuatro aplicaciones móviles. Con las soluciones propuestas, los servicios de emergencia pueden conocer la ubicación tan pronto como reciban la alerta y pueden acudir más rápido a la emergencia.

Palabras clave: Bluetooth, Móvil, Localización de Interiores, Seguridad, Servicios de Emergencia, Cloud, SIP

1. Introducción

La ubicación en interiores se ha convertido en un requisito en las llamadas de emergencia. Hoy en día, solo el 40% de ellos incluye la ubicación y en muchas ocasiones los servicios de emergencia no saben a dónde ir. Es crucial que los servicios de emergencia lleguen lo antes posible, pero esto no es factible cuando no tienen la ubicación. Se estima que se podrían salvar diez mil vidas si la ubicación estuviera incorporada dentro de la llamada. La FCC y las principales operadoras han acordado abordar esta situación mejorando constantemente el porcentaje de llamadas que llegan con la ubicación. Existen muchas soluciones basadas en la localización de interiores, pero ninguna de ellas funciona directamente con el Módulo de emergencias NG 9-1-1.

Por otro lado, el IIT y sus aledaños sufren aproximadamente veinte mil incidentes al año. Aunque hay una policía privada a cargo del campus, y hay muchas maneras de contactarlos, todos incluyen reportar el incidente mediante un método de llamada que no incluye la ubicación.

2. Definición del proyecto

El proyecto tiene como objetivo abordar estos dos problemas. Para las llamadas de emergencia, se han diseñado tres aplicaciones. El primero, habilita la posibilidad de realizar una llamada que primero recoge información sobre los beacons encontrados en su ubicación y las envía a la *BOSSA Platform*. La *BOSSA Platform* es una plataforma que controla tanto los dispositivos bluetooth como los algoritmos de localización de interiores. Cuando la *BOSSA Platform* recibe la información de los beacons, calcula la localización. Esta ubicación se incorpora dentro de la llamada SIP que irá al Módulo NG-9-1-1. La segunda es una aplicación de test que sirve para alimentar la *BOSSA Platform* con datos. Esta información es valiosa para mejorar la precisión de los algoritmos. La tercera aplicación se desarrolló utilizando un entorno multiplataforma. El funcionamiento de esta aplicación es igual que la primera, pero la llamada se realiza directamente a través de la PSTN a los servicios de emergencia. La llamada es un mensaje automático con la localización del usuario.

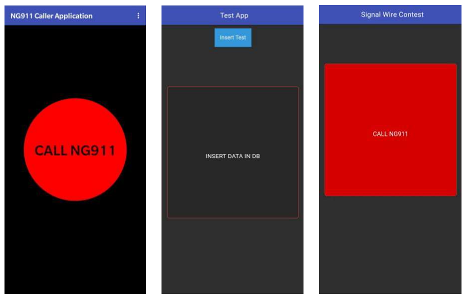
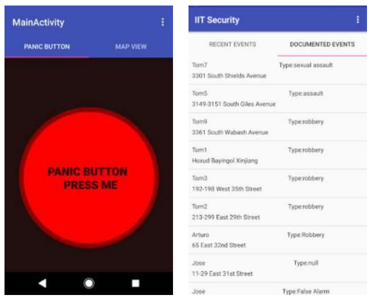


Ilustración 1: Pantallas principales de las tres aplicaciones

Para la seguridad del campus, hemos diseñado un sistema cerrado para estudiantes y autoridades. Es una aplicación de Android que interactúa con los estudiantes y las autoridades. Los estudiantes pueden presionar el botón de pánico para asistencia inmediata y pueden ver en un mapa los incidentes reportados. Las autoridades recibirán la alerta y podrán llegar al punto de emergencia más pronto. Además, deben reportar el incidente y establecer el tipo de emergencia.



Tom7 3301 South Shields Avenue	Type:sexual assault
Tom5 3149-3151 South Giles Avenue	Type:assault
Tom9 3361 South Wabash Avenue	Type:robbery
Tom1 Hoxud Bayingol Xinjiang	Type:robbery
Tom3 192-198 West 35th Street	Type:robbery
Tom2 213-299 East 29th Street	Type:robbery
Arturo 65 East 32nd Street	Type:Robbery
Jose 11-29 East 31st Street	Type:null
Jose	Type:False Alarm

Ilustración 2: Pantallas principales de la interfaz de usuario y autoridades

3. Descripción del modelo/sistema/herramienta

Dos de las aplicaciones se han realizado para móviles Android y otras dos en Apache Cordova. Las aplicaciones de Apache Cordova son multiplataforma y se pueden instalar en todos los móviles. La arquitectura propuesta es la que aparece en la siguiente ilustración. El móvil recoge la información de los dispositivos bluetooth. La aplicación de test envía esta información directamente a la base de datos. Las otras dos restantes envían dicha información a la *BOSSA Platform* para saber la localización. Una vez llegados a este punto, las aplicaciones realizan distintas funciones. La primera descrita hace la llamada al módulo NG 9-1-1. La tercera descrita envía la información en una llamada automática a la policía haciendo uso de texto a voz.

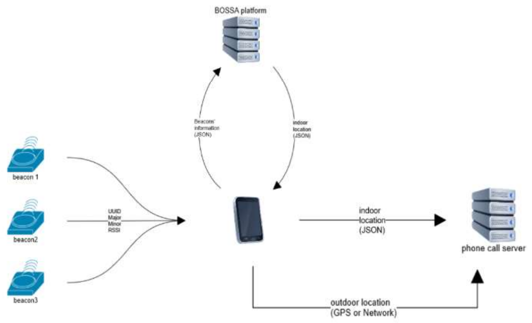


Ilustración 3: Arquitectura de las aplicaciones de emergencia

La aplicación del campus utiliza la plataforma Bluemix como apoyo para dar servicio a todas las necesidades.

4. Resultados

Como se puede apreciar, todas las aplicaciones son similares, todas se basan en apretar un botón que después realizará las funciones correspondientes. En el ámbito de los servicios de emergencia se ha conseguido mejorar el tiempo de latencia de 30-40 segundos que había con anterioridad a 15 segundos. Con esta solución, la integración con el módulo real de emergencias es inmediato ya que, se ha usado una réplica de la infraestructura de los servicios de emergencia. La solución es independiente de los dispositivos de bluetooth siempre que se registren en la plataforma. También se ha conseguido, gracias a las aplicaciones multiplataforma proporcionar localización de interiores a todos los usuarios móviles. En el ámbito de la seguridad del campus, esta plataforma habilita la posibilidad de asistencia en un tiempo menor al actual ya que, los servicios de emergencia permiten saber el lugar de la incidencia sin la necesidad de tener que hablar con el estudiante a priori. Es una aplicación rápida para que el estudiante pueda pedir ayuda en cuestión de 5 segundos.

Por otro lado, cabe destacar la aplicación de test. Proporcionar datos al sistema es crítico para que se puedan realizar test sobre los algoritmos y así poder mejorar la precisión. De nada sirve tener un sistema montado con aplicación y la infraestructura necesaria si los algoritmos no saben calcular con precisión la localización, pues estos, podrían dar localizaciones erróneas que incluso podrían ser más perjudiciales que no tener localización.

5. Conclusiones

El proyecto cubre con todos los objetivos propuestos. Parte de él consta como trabajo a futuro de un proyecto realizado por la universidad en donde se centraban en buscar la localización por los dispositivos Wi-Fi instalados en un edificio. No obstante, esta solución no era viable para apartamentos privados limitando su uso a centros comerciales o edificios que tuviesen la misma infraestructura Wi-Fi. La aplicación multiplataforma desarrollada servirá como primer paso para lanzar una aplicación que realice las mismas funciones que la realizada en Android (Llamada al módulo NG 9-1-1). La aplicación de test permitirá realizar los análisis correspondientes para mejorar la precisión de los algoritmos implementados en la plataforma. En el ámbito del campus de la universidad, el proyecto consigue una mejora respecto a los sistemas actuales.

Por último, el proyecto realizado completa el último modulo que faltaba de la arquitectura propuesta y los encargados pueden empezar a realizar pruebas para empezar a implementarlo en mas edificios de Chicago para así poder llegar a más gente y acabar siendo la solución para la localización de interiores en Estados Unidos.

INDOOR LOCATION SYSTEM FOR MOBILE PHONES BASED ON BLUETOOTH LOW ENERGY ARRAYS.

Author: Martín Casarrubios, José Luis.
Supervisor: Davids, Carol.
Collaborating Entity: IIT Real-Time Communications Lab.

ABSTRACT

The project at hand offers a variety of solutions to the indoor location problem in emergency calls and to improve campus security at the Illinois Institute of Technology through the development of four mobile applications. Using the proposed solutions, the emergency services can know a user's location as soon as they receive an emergency alert from their device and can therefore assist them quicker.

Keywords: Bluetooth, Mobile, Indoor Location, Security, Emergency services, Cloud, SIP

1. Introduction

Indoor location should be a key piece of information in emergency calls. Nowadays, only 40% of them include this location, so the emergency services frequently don't know where to go, thus preventing them from arriving at the scene as soon as possible. It is estimated that ten thousand lives could be saved if indoor location was embodied inside the call. The FCC and main cell phone carriers have agreed to tackle this issue by steadily improving the percentage of calls that do contain indoor location. There are many solutions to include indoor location in these calls, but none of them work directly with the NG 9-1-1 Module.

On the other hand, IIT and its surroundings suffer from approximately 20,000 reported incidents per year. Although there is a private police force in charge of campus security and there are many ways to contact them, they all involve reporting the incident by a calling method that doesn't include the victim's location.

2. Project Definition

The project aims to tackle the aforementioned problems. For emergency calls, three applications were designed. The first one allows a call to be made that will firstly collect information from beacons found near the caller's location and send it to the BOSSA Platform, a platform that controls both the beacons and the algorithms to determine the caller's indoor location. When the BOSSA Platform receives the beacons, it calculates the indoor location. This location will then be embodied inside the SIP call to the NG-9-1-1 Module. The second is a test application to feed the BOSSA Platform with raw data. This data will then be analyzed to improve the algorithms' accuracy. The third application was developed using a cross-platform environment. It is similar to the first one except the call is made directly through the PSTN to the emergency services. The call is an automated message containing the information pertaining to the caller's indoor location.

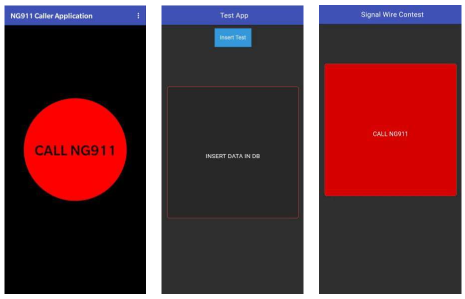
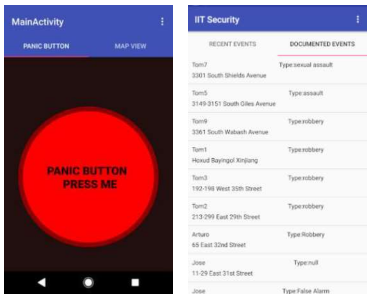


Illustration 1: Main screens of the emergency applications

With the purpose of campus security in mind, we have designed a closed system for both students and the authorities. It is an Android application which both can use. Students can press the panic button for immediate assistance and can view in a map of past reported incidents in their area. Authorities will receive the alert and will be able to arrive at the emergency point sooner. Also, they must report the incident by inputting the type of emergency assisted.



Tom7 3301 South Shields Avenue	Type:sexual assault
Tom5 3149-3151 South Giles Avenue	Type:assault
Tom9 3361 South Wabash Avenue	Type:robbery
Tom1 Hoxud Bayingol Xinjiang	Type:robbery
Tom3 192-198 West 35th Street	Type:robbery
Tom2 213-299 East 29th Street	Type:robbery
Arturo 65 East 32nd Street	Type:Robbery
Jose 11-29 East 31st Street	Type:null
Jose	Type:False Alarm

Illustration 2: Main screens for students and authorities

3. Model System Description

Two of the applications have been developed for Android phones and two others in Apache Cordova. Apache Cordova applications are cross-platform and can be installed on all mobile phones. The proposed architecture is the one that appears in the following illustration. The phone collects information from Bluetooth devices. The test application sends this information directly to the database. The two remaining applications send this information to the BOSSA Platform to find out the location. The applications perform different functions: the first one described makes the call to the NG 9-1-1 module, whereas the third one described sends the information in an automatic call to the police using text to speech.

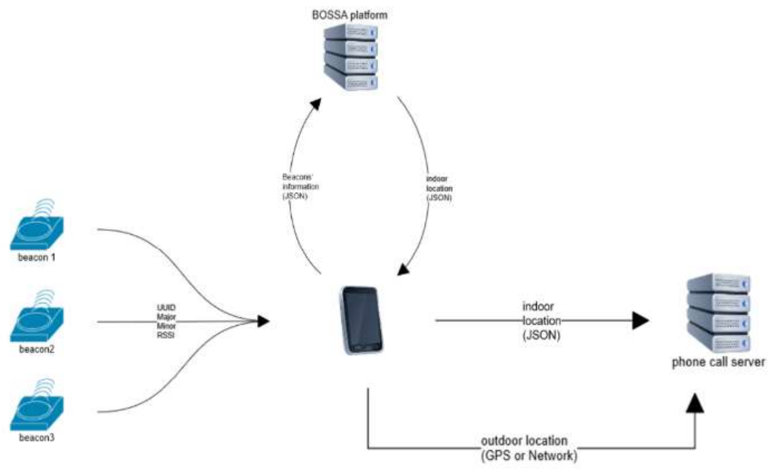


Illustration 3: Architecture of emergency applications

The campus security application uses the Bluemix Platform as a support to all the needed services.

4. Results

As seen in the illustrations, all four applications are similar. They are all based on the notion of pressing a button that will then perform the corresponding functions. In the field of emergency services, it has been possible to improve a latency time of 30-40 seconds to 15 seconds. Moreover, this solution makes the integration with the real emergency module immediate, since we have used a replica of the emergency services' infrastructure. The solution is independent of any Bluetooth devices as long as they are registered in the platform. It has also achieved, thanks to cross-platform applications, to provide the indoor location of all mobile users in case of emergency. It is also worth mentioning our test application, which provides critical data to the system in order to perform tests on the algorithms and improve their accuracy. It is useless to have a system mounted with the application and the necessary infrastructure if the algorithms do not know how to accurately calculate the user's location, as erroneous locations could even be more harmful than no location at all.

5. Conclusion

The project covers all the proposed objectives. Part of the project developed consists of another project carried in the future by the university, focused on finding the user's location using a building's Wi-Fi devices. However, this solution was not viable for private apartments, thus limiting its use to malls or buildings that had the same Wi-Fi infrastructure. The multiplatform application developed will serve as the first step to launch an application that performs the same functions as the one on Android (calling to the NG 9-1-1 module). The test application will allow us to analyze the algorithms to improve location accuracy. In the campus security field, the project achieves an improvement over the current systems in place.

Finally, the project completes the last module that was missing from the proposed architecture. Managers can begin to perform tests and start deploying the system in more buildings in Chicago in order to reach more people and this could end up being the solution for finding accurate indoor location in the United States.

Index

Figure Index

Table Index

Chapter 1. INTRODUCTION

The project being presented consists of two different parts with the same purpose: To improve the emergency system in the United States.

- The first part is based on the study of the current security situation of the campus of the Illinois Institute of Technology (IIT) campus, located in the South side of Chicago.

- The second is based on improving emergency services in the United States as a whole, including the location of people in emergency calls. This is done by scanning iBeacons to triangulate the location and embedding the indoor location inside the call.

Chicago has one of the highest crime rates in the United States. Figure 1, shows a map of Chicago where the different neighborhoods are colored in and ranked in terms of their safety. The university is located in the South side of Chicago, surrounded by marginal neighborhoods where crime and assaults are frequent.

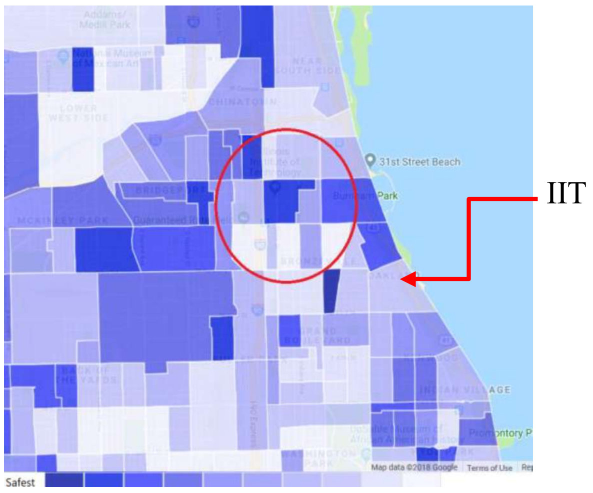


Figure 1: Chicago's security Map [1]

Although the university appears to be in a safe neighborhood and provides its own campus security, assaults are frequent, leaving students no choice but to report them by making calls from their private cell phones or nearest telephone post (see figure 2).

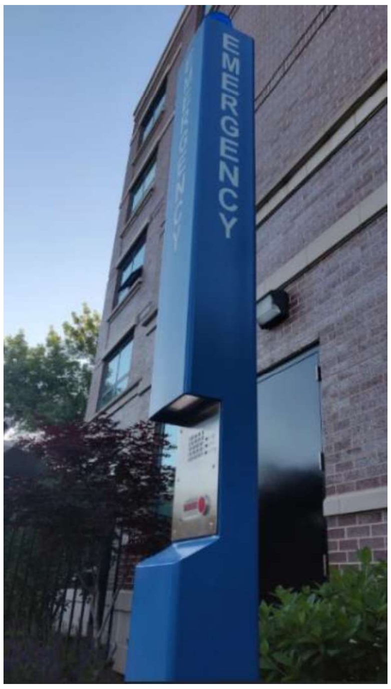


Figure 2: Emergency Post

These posts are found throughout the university campus at a distance of 500 meters from each other. Despite their homogeneous distribution, assailants are able to choose the opportune moment to carry out the assault in such a way that the student does not have time to notify security. In emergency situations at the university, the latter must either call the police or go to the nearest post to detail the exact situation is and where on campus it is taking place. This has proved to be inefficient because calls usually take place after the attack and from nearby telephone post, since in most cases attackers steal mobile phones.

Figure 2 and 3 provide a closer look at the pattern of reported crimes throughout the city, extracted from the Chicago Police Department's system, over the past year, the last seven days' worth of data notwithstanding. [2]

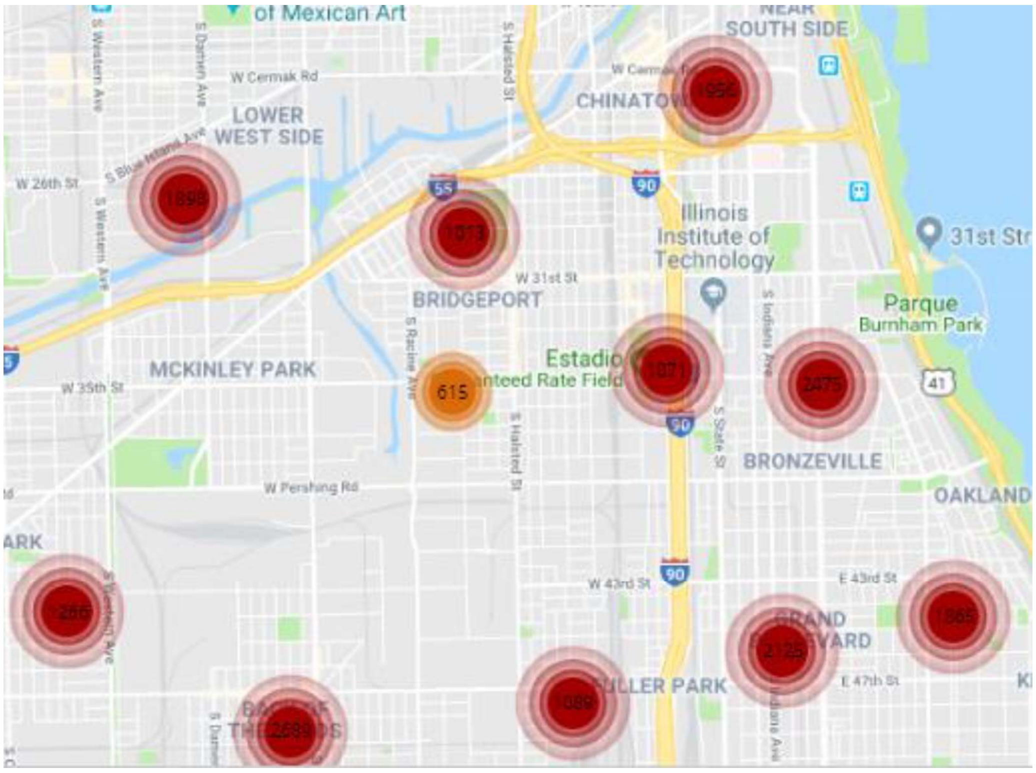


Figure 3: Crimes Reported over the last year [2]

As the figure above shows, there have been 18,062 reported incidents over the past year near IIT. When zooming in to streets encompassed by the IIT, we can see the 989 incidents were reported on the main campus. This amount to almost three attacks per day. Students also often leave the campus, so many of these incidents also occur outside of it. This emphasizes the need to improve security at IIT.

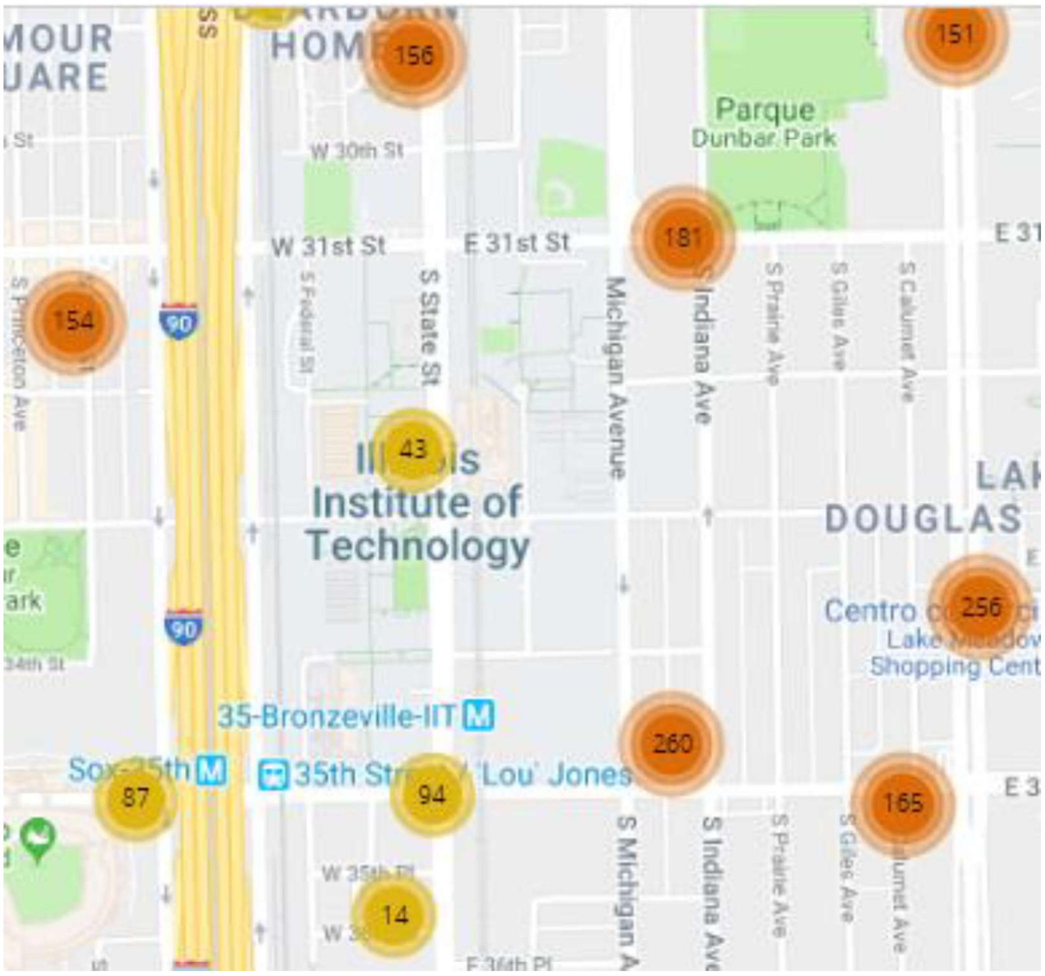


Figure 4: Crimes Reported over the last year (zoomed) [2]

On the other hand, the second part of the project focuses on the United States' emergency system as a whole. Most emergency calls are currently sent without the location of the call; it was reported that only 40% of the calls include an exact location. It is estimated that ten thousand lives a year could be saved; if this information were provided to the emergency services, as it would speed up their arrival.

Therefore, the project will tackle both of these issues in order to improve emergency services both in and outdoors.

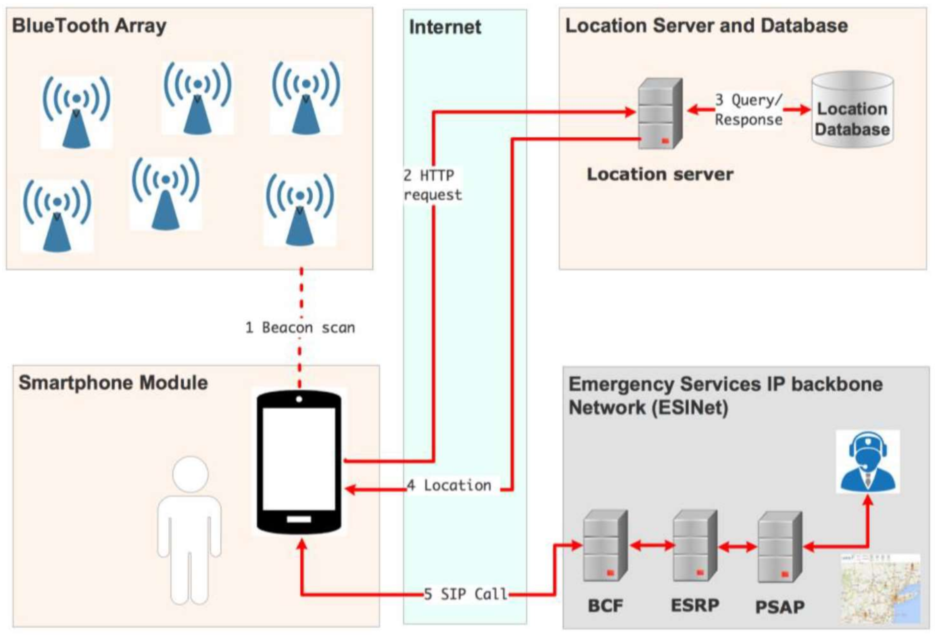


Figure 5: Actual Architecture of the Emergency System

Currently, there is a testing environment at the university where the emergency system is being developed. In the laboratory, they have the same servers (see figure 3) that are used for emergency calls nation-wide. Previously, there was an Android application that scanned the room's sensors, before making the SIP call to the ESINet (Emergency Service IP Backbone Network). This Android-based application is currently in disuse because of changes made to the ways in which infrastructure was managed, and the version is now obsolete, enabling only calls where the indoor location of the caller could be pinpointed to be made. The solution proposed in this project is developing an application that will send the indoor location and will make the call whether it finds a beacon or not, as the utmost priority is placing this call in the first place. There would be a second approach through the use a cross-platform application that delivers the indoor location through a cloud environment. Lastly, an application for inserting raw data into the system will be developed so that the engineers in charge of the algorithms can perform analyses to improve the location accuracy.

As a whole, the project aims to solve the problems relating to indoor location and helps improve its accuracy by feeding data into the system. It will furthermore, help students get assistance sooner, ensuring a safer environment for all. They will also know which places to avoid and which ones are safer.

Chapter 2. DESCRIPTION OF THE TECHNOLOGIES

2.1 PROTOCOLS AND STANDARDS: APPLICATION LAYER

2.1.1 SIP

Session Initiation Protocol (SIP) is a protocol developed by the MMUSIC (Multiparty Multimedia Session Control) working group of the IETF with the intention of being the standard for the initiation, modification and termination of interactive user sessions where multimedia elements such as video, voice, instant messaging, online games and virtual reality intervene.

The syntax of its operations resembles those of HTTP and SMTP, the protocols used in the services of Web pages and distribution of e-mails respectively. This similarity is natural since SIP was designed so that telephony would become one more service on the Internet. [3]

In the RFC 3261 it is defined as: “an agile, general-purpose tool for creating, modifying, and terminating sessions that works independently of underlying transport protocols and without dependency of the type of session that is being established”. [4]

SIP is classified as a Client- Server Protocol in which SIP Agents can act as both clients and servers. The client sends requests and the servers sends responses. This protocol can be used to create peer-to-peer applications on the internet and in our case is a telephone application.

The elements contained in SIP are the following

- **User Agent (UA):** Is an internet endpoint that generates SIP requests and responses. We will create a User Agent for our Android Application to send an invite to the ESINet (Emergency Services IP backbone Network). The user wants to create a stream of media between him and a distant party. The user is in charge of setting up, modifying and disconnecting said media streams. [5]

- **Proxy:** The proxy helps route request to the user's current location. It authenticates and authorizes users for service and allows user agent to establish a call.

- **Registrar:** It is a server that accepts registration requests and places the information contained in those requests into the location service for the domain it handles. This function can run on the same platform. Before making the call, we will have to register so that we will be able to make a call to the ESINet.

- **Redirect Agent:** It is a server that generates responses to clients request and redirects the client.

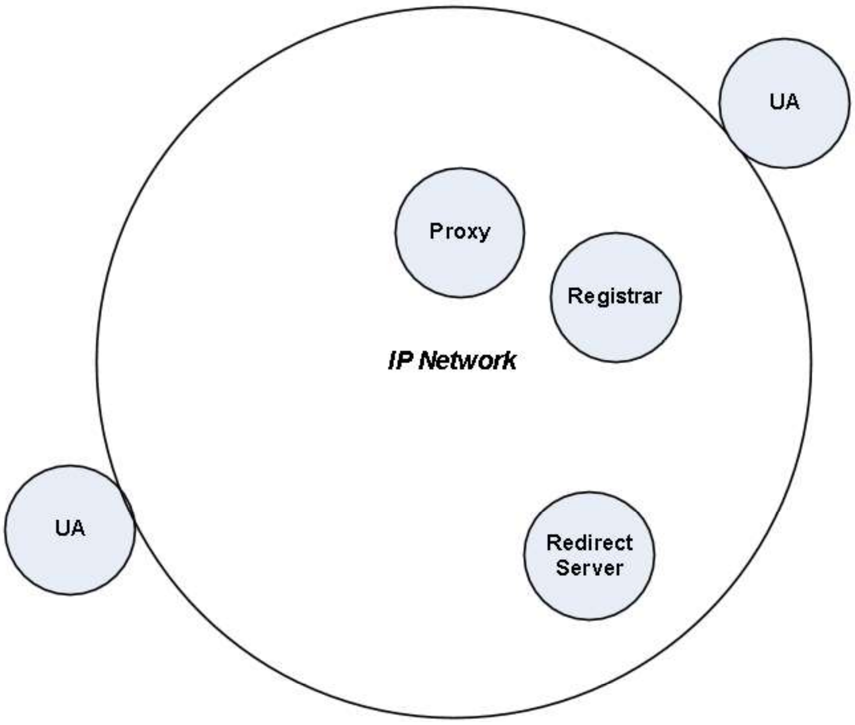


Figure 6: SIP Elements [5]

2.1.1.1 SIP Methods

There are 6 SIP methods that all SIP agents must understand. These methods are the following [6]:

- **REGISTER:** Communicates user location (host name, IP),
- **INVITE:** Establishes a session.
- **ACK:** Confirms an INVITE request,
- **BYE:** Ends a session.
- **CANCEL:** Cancels establishing of a session.
- **OPTIONS:** Communicates information about the capabilities of the calling and receiving SIP phones.

Also, there are some optional methods defined in the SIP extension that are:

- **PRACK** = Provisional Acknowledgement.
- **SUBSCRIBE** = Subscribes for Notification from the notifier.
- **NOTIFY** = Notifies the subscriber of a new event.
- **PUBLISH** = Publishes an event to the Server.
- **INFO** = Sends mid-session information.
- **REFER** = Asks the recipient to issue call transfer.
- **MESSAGE** = Transports Instant Messages.
- **UPDATE** = Modifies the state of a session.

2.1.1.2 Response Codes

The response codes for a SIP request can be the following:

- **1xx – Provisional:** Request received, continuing to process the request.
- **2xx – Success:** The action was successfully received, understood, and accepted.
- **3xx – Redirection:** Further action needs to be taken in order to complete the request.

- **4xx – Client Error:** The request contains bad syntax or cannot be fulfilled at this server.
- **5xx – Server Error:** The server failed to fulfill an apparently valid request.
- **6xx – Global Failure:** The request cannot be fulfilled at any server.

2.1.1.3 Normal SIP INVITE

It is important to mention how a SIP INVITE works because in the project at hand, when packaging an invite, we insert the indoor location in the body of the invite. In order to study how it's done, we must present the normal SIP INVITE. A normal SIP INVITE has the following headers:

- **Request-Line-URI:** The Request-Line-URI includes the destination of the call. It contains the same information as the To field, omitting the display name.

- **Via:** Every proxy in the request path adds to top of the "Via" the address and port on which it received the message, then forwards it onwards.

- **From:** The "From" header field indicates the identity of the initiator of the request similar in construction to email addresses (user@domain – where "user" is, for example, the extension number, and "domain" is the server domain or IP address). In our case **caller1@64.131.109.30.**

- **To:** Specifies the desired "logical" recipient of the request, or the address-of-record of the user or resource that is the target of this request. This may or may not be the ultimate recipient of the request. Typically, the "To" field contains a SIP URI that points to the first (next) hop (proxy) that will process the request, but not necessarily the SIP URI of the eventual recipient.

- **Contact:** The "Contact" header field provides a single SIP URI that can be used to contact the sender of the INVITE for subsequent requests.

- **Content-Type:** indicates the media type of the message-body sent to the recipient. This one will differ in our SIP INVITE as we are going to change the body to a MIME Body.

```
INVITE sip:767@ng911main.iit.edu SIP/2.0
Via: SIP/2.0/UDP 192.168.2.2:44125;rport;branch=z9hG4bK07692
Max-Forwards: 70
To: <sip:767@ng911main.iit.edu>
From: <sip:caller1@ng911main.iit.edu>;tag=z9hG4bK66690496
Call-ID: 282795648461@192.168.2.2
CSeq: 1 INVITE
Contact: <sip:caller1@192.168.2.2:44125;transport=udp>
Expires: 3600
User-Agent: Sipdroid/3.7 beta/ONEPLUS A6003
Content-Length: 391
Content-Type: application/sdp

   v=0
   o=caller1@ng911main.iit.edu 0 0 IN IP4 192.168.2.2
   s=Session SIP/SDP
   c=IN IP4 192.168.2.2
   t=0 0
   m=audio 21000 RTP/AVP 9 8 0 97 3 106 101
   a=rtpmap:9 G722/8000
   a=rtpmap:8 PCMA/8000
   a=rtpmap:0 PCMU/8000
   a=rtpmap:97 speex/8000
   a=rtpmap:3 GSM/8000
   a=rtpmap:106 BV16/8000
   a=rtpmap:101 telephone-event/8000
   a=fmtp:101 0-15
   m=video 21070 RTP/AVP 103
   a=rtpmap:103 h263-1998/90000
```

Figure 7: SIP INVITE FORMAT

2.1.1.4 MIME SIP INVITE

The MIME Body is the one that we use in this project. The content-type now is multipart/mixed and the messages are separate between boundaries that we define. In our case, the boundary is defined by NG911 followed by a random string of 50 characters generated for each invite. The first body message is the same as the one found in the normal SIP INVITE. The second body message found defined by the boundaries is the location in which the caller is. The next figure shows a whole INVITE modified to use the multipart/mixed and send the indoor location.

```
INVITE urn:service:sos SIP/2.0
Via: SIP/2.0/UDP 192.168.2.2:40132;rport;branch=z9hG4bK28899
   Max-Forwards: 70
   To: urn:service:sos
   From: <sip:caller1@192.168.2.11>;tag=z9hG4bK27347248
   Call-ID: 230908050442@192.168.2.2
   CSeq: 1 INVITE
   Contact: <sip:caller1@192.168.2.2:40132;transport=udp>
   Expires: 3600
   User-Agent: Sipdroid/3.7 beta/ONEPLUS A6003
   Accept-Language: en
   Priority: emergency
   Geolocation: <cid:caller1@192.168.2.11>; inserted-by="192.168.2.11"; used-for-
routing
   Date: Wed, 18 Jul 2018 13:02:19 GMT-05:00
   Content-Length: 1775
   Content-Type: multipart/mixed;

boundary=NG911HrzpRtMWWYggCkRmaFVMitl4O1YFqoXxhamWeLSCdO7x7
VSTou
   --NG911HrzpRtMWWYggCkRmaFVMitl4O1YFqoXxhamWeLSCdO7x7VSTou
   MIME-Version: 1.0
   Content-ID: <230908050442@192.168.2.2>
   Content-Type: application/sdp
   Content-Transfer-Encoding: 8bit
   v=0
   o=caller1@192.168.2.11 0 0 IN IP4 192.168.2.2
   s=Session SIP/SDP
   c=IN IP4 192.168.2.2
   t=0 0
   m=audio 21000 RTP/AVP 9 8 0 97 3 106 101
   a=rtpmap:9 G722/8000
   a=rtpmap:8 PCMA/8000
   a=rtpmap:0 PCMU/8000
   a=rtpmap:97 speex/8000
   a=rtpmap:3 GSM/8000
   a=rtpmap:106 BV16/8000
   a=rtpmap:101 telephone-event/8000
   a=fmtp:101 0-15
   m=video 21070 RTP/AVP 103
   a=rtpmap:103 h263-1998/90000
   --NG911HrzpRtMWWYggCkRmaFVMitl4O1YFqoXxhamWeLSCdO7x7VSTou
   MIME-Version: 1.0
   Content-ID: <caller1@192.168.2.11>
   Content-Type: application/pidf+xml
   Content-Transfer-Encoding: 8bit

   <?xml version="1.0" encoding="ISO-8859-1"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
      xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
      xmlns:ca="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
     xmlns:gml="http://www.opengis.net/gml"
      entity="sip:100@102.168.2.16">
    <tuple id="id82848">
     <status>
      <gp:geopriv>
       <gp:location-info>
         <ca:civicAddress>
          <ca:country>us</ca:country>
          <ca:A1>IL</ca:A1>
          <ca:A2>chicagosouth</ca:A2>
          <ca:LOC>Used iBeacon Location system solution</ca:LOC>
          <ca:FLR> 2  </ca:FLR>
          <ca:X> 11.514269969684792  </ca:X>
          <ca:Y> 60.32377653343406  </ca:Y>
                    <ca:PLC> 4</ca:PLC>
         </ca:civicAddress>
       </gp:location-info>
       <gp:usage-rules/>
       <gp:method>Manual</gp:method>
      </gp:geopriv>
     </status>
    <contact priority="0.8">sip:100@192.168.2.16</contact>
   <timestamp>2018-07-18 13:02:19.308</timestamp>
    </tuple>
   </presence>
   --NG911HrzpRtMWWYggCkRmaFVMitl4O1YFqoXxhamWeLSCdO7x7VSTou
--
```

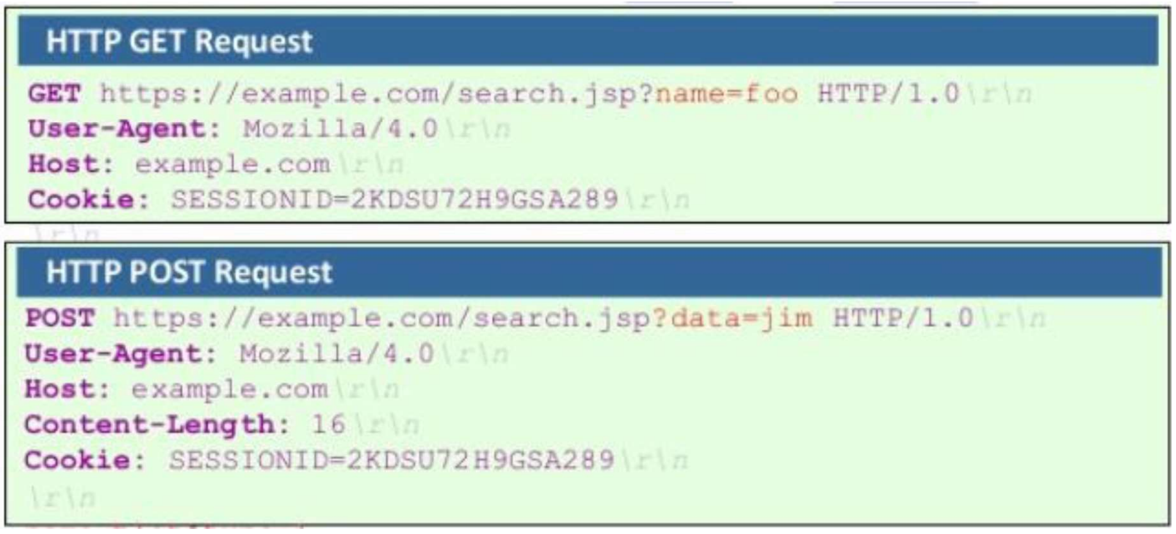
Figure 8: INVITE SIP with PIDF-LO XML

2.1.2 HTTP

Hypertext Transfer Protocol is the communication protocol that allows the transfers of information in the World Wide Web. HTTP defines the syntax and semantics that the software elements of the web architecture (clients, servers, proxies) use to communicate. HTTP is a stateless protocol, that is, it does not store any information about previous connections. The development of web applications frequently needs to maintain state. For this, cookies are used, which is information that a server can store in the client's system. This allows web applications to institute the notion of a session, and also allows users to be tracked since cookies can be stored on the client for an indefinite period of time [7]. In HTTP we can find several methods which are explained below:

- **GET**: Is used to request data from a specified resource and is one of the most common HTTP methods.
- **POST**: Is used to send data to a server to create/update a resource. The data sent to the server with POST is stored in the request body of the HTTP request.
- **PUT**: Is used to send data to a server to create/update a resource. The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.
- **HEAD**: Is almost identical to GET, but without the response body. In other words, if GET /users returns a list of users, then HEAD /users will make the same request but will not return the list of users.
- **DELETE**: Deletes the specified resource.
- **PATCH**: Applies partial modifications to a resource.
- **OPTIONS**: Describes the communication options for the target resource.

[8]


HTTP GET Request

```
GET https://example.com/search.jsp?name=foo HTTP/1.0\r\n
User-Agent: Mozilla/4.0\r\n
Host: example.com\r\n
Cookie: SESSIONID=2KDSU72H9GSA289\r\n
\r\n
```

HTTP POST Request

```
POST https://example.com/search.jsp?data=jim HTTP/1.0\r\n
User-Agent: Mozilla/4.0\r\n
Host: example.com\r\n
Content-Length: 16\r\n
Cookie: SESSIONID=2KDSU72H9GSA289\r\n
\r\n
```

Figure 9: HTTP Request (Get vs Post) [9]

In this project we will use HTTP using the GET method to request for data. These requests are sent to our Bluemix cloud Platform, where we have developed an API that retrieves the data.

2.2 *PROTOCOLS AND STANDARDS: PHYSICAL LAYER*

2.2.1 WI-FI

Wi-Fi is a standard through which devices connect wirelessly to networks. It is a very extended technology of high speed. The 802.11n was ratified by the organization IEEE with a speed of 600 Mbps in physical layer. Being a half-duplex technology, it can reach up to 300 Mbps. It operates on free frequencies that do not require a license for their use: 2.4 and 5 GHz. Each of them presents its advantages and disadvantages. First, the 2.4 GHz frequency has greater penetration and works better indoors. However, it has more interference with home appliances, since they operate at the same frequency. The frequency of 5 GHz does not interfere with that of household appliances, but, having a longer wavelength (γ), the obstacle attenuation is also greater, decreasing its range. [10]

Figure 10: Wi-Fi Connection [11]

Among the advantages of using this technology are:

- Low cost
- High Speed
- Connection universality
- Good level of security

The disadvantages of using this technology are:

- Signal interference.
- Limited distance for reception.
- Relatively high consumption

2.2.2 BLUETOOTH LOW ENERGY

Also known as Bluetooth Smart or Version 4.0+, this new technology is the Bluetooth version created specifically for the IoT. It is mainly based on reducing consumption and minimizing the transmission power of the radio signal used and the coverage radius, making the device in question optimal for operation for long period of time using small batteries. It can maintain confidentiality through 128-bit AES encrypted connections, and redundancy codes that minimize erroneous transmissions. It obtains improvements in the speed of up to 24 Mbps and saves energy, which is the reason why it is considered a key standard to give support to "wearables" devices. It is a cheap and easy to use technology when developing applications. In addition, it has a friendly architecture, as it is assembled by the developer, and thus, is not subject to the restrictions of a fixed structure. [12]

The key features of this technology are its:

- **Interoperability:** It is a standard wireless protocol that allows its use by all types of vendors.
- **Low consumption:** Allows you to operate for months with a small battery.
- **Standardized applications:** Low costs during development and operation.
- **Security:** Allows you to obtain encryption security with 128-bit AES.

2.2.3 IBEACONS

iBeacons are based on Bluetooth 4.0 or Bluetooth Low Energy technology. They can be found at fifty meters away, having to create an application for the final devices that understands the signals that these beacons emit. The principle is similar to that of NFC technology, with the difference that it has a higher reception range. The beacons transmit their state and identification information.

Figure 11: Axa Beacon

In the project, the iBeacons have a unique identification composed of:

- **Minor:** This is the identification of the beacon. It is a unique identifier in every building.
- **Major:** This is the identification for the building. It must be unique throughout the buildings involved in the project.

Both together form the unique identifier for each beacon. Each beacon is located in a certain place and recorded in the database, this way we know the exact positions of our iBeacons and based on the RSSI received we can triangulate our x and, y coordinates to determine the indoor location. The iBeacons are located inside three buildings: the Stuart Building, the Alumni Hall and the Carman Hall. These buildings can be found at the Illinois Institute of Technology Main Campus.

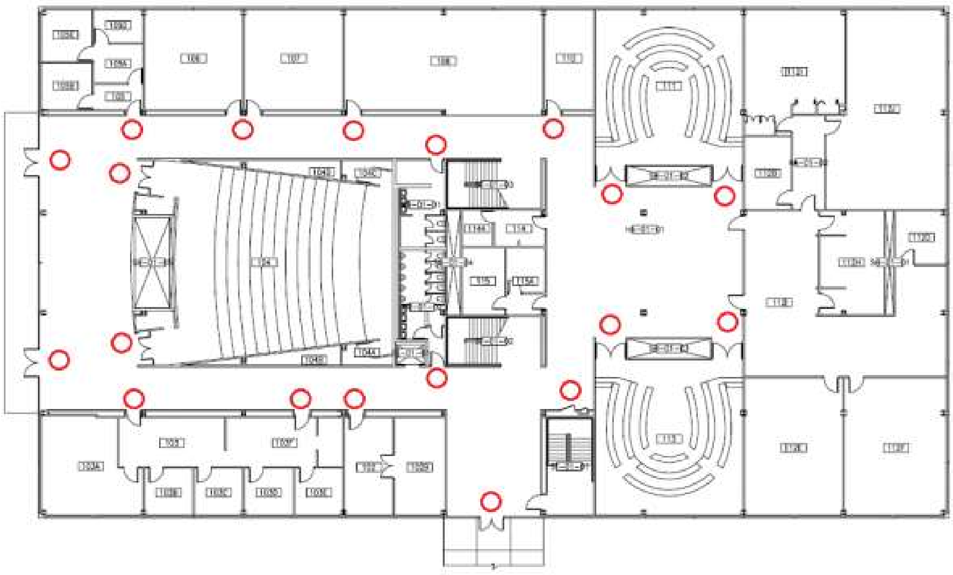


Figure 12: Stuart Building 1st Floor with the iBeacons distribution

2.2.4 GPS

The GPS (Global Position System) is a U.S.-owned utility that provides users with positioning, navigation, and timing (PNT) services. It provides service to civilian and military users. The civilian service is freely available to all users on a continuous, worldwide basis. The GPS is used on the campus security application but not on the emergency service application. This is because when using the emergency service, we want to know the floor and building of the caller and the GPS does not provide this information. Moreover, when an attack takes place on the IIT campus, it always takes place outdoors and GPS is sufficient to locate the person in danger. [13]

2.3 HARDWARE USED

2.3.1 RASPBERRY PI

The Raspberry Pi 3 is a minicomputer with the size of a credit card that has a four-core ARM Cortex-A53 1.2 GHz processor (it is the first Raspberry Pi processor with 64-bit support). It is particularly useful in the development of projects far more complex than those developed with Arduino, being able to use high-level programming languages such as Python, C ++ and Java. Raspberry Pi also allows multiprogramming (having several programs enabled simultaneously). It is one of the most widespread plates on the market, it can also be used as a Linux Low Cost server and is oriented towards "non-dedicated" projects, that is, projects that do not need to be exclusively dedicated to a single task, since, by having an operating system, they do not have direct access to the hardware and the response time is greater than that of a dedicated device. Depending on the Raspberry Pi in question, it will have more or less USB ports and different connections. The latest model has Bluetooth 4.0, Wi-Fi and Ethernet port. There are also external modules such as the Xbee module that can be added to enable different protocols in the physical layer that allow you to establish other types of connections (in the case of the Xbee module, it enables the Zigbee connection). [14]

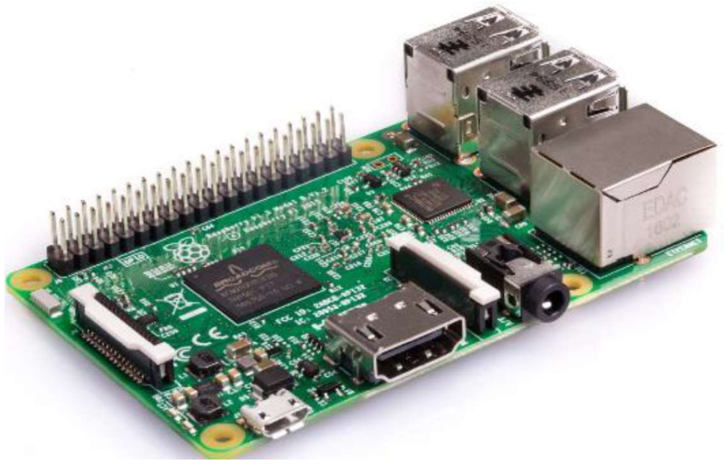


Figure 13: Raspberry Pi 3 [15]

Its cost is around $39, with dimensions of 3.34 x 2.12 inches. The biggest disadvantage of the Raspberry Pi 3 is its lack of inputs for analog sensors, making necessary an analog-digital converter for analogous sensors. Nevertheless, for the use of this project we will use the Raspberry Pi as a SIP server to test the emergency mobile application.

2.3.2 ESINET

The ESINet (Emergency Services IP backbone Network) is a set of hardware modules (SBC, ESRP) that form a network module call to reach the correct PSAP. The IIT RTC Lab has a replica that will be used to test the applications developed. This module was created to simulate the behavior of a real ESINet infrastructure and it is compliant with the NG911 NENA i3 standards. [16] By this, we know that our solution will be fully compliant when the solution is ready to be deployed to the users.

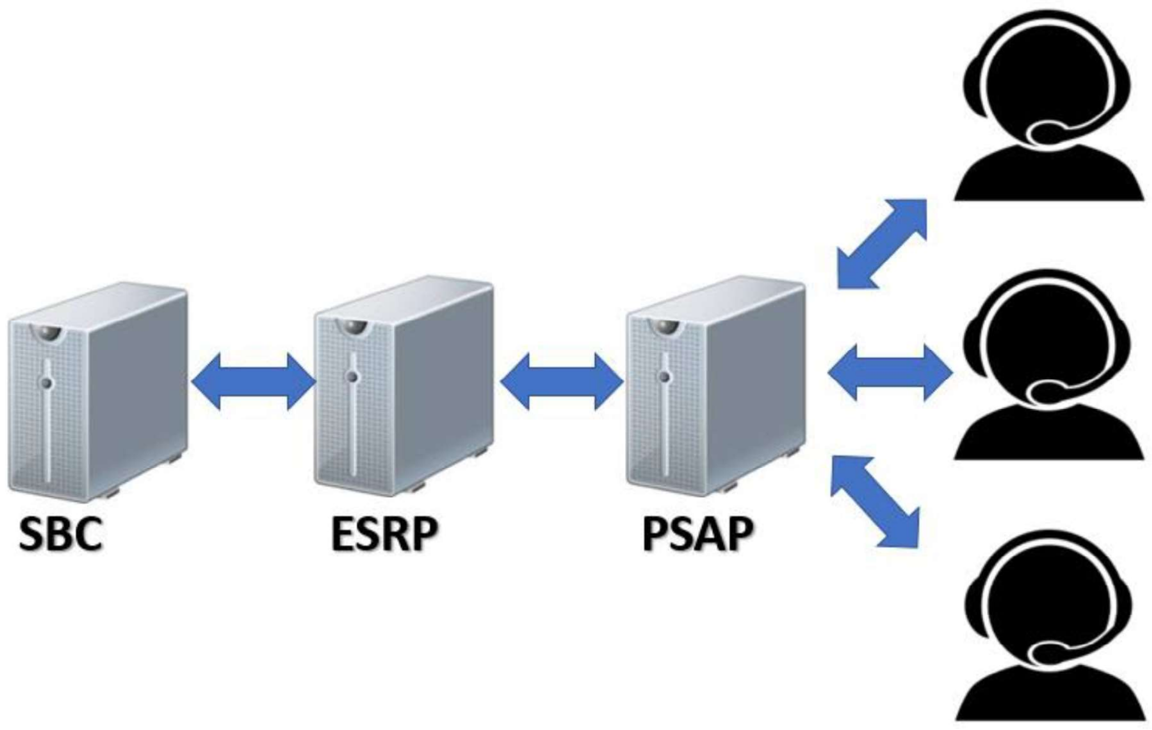


Figure 14: NG 9-1-1 Module

The ESINet is fundamentally three hardware modules that will be explained below:

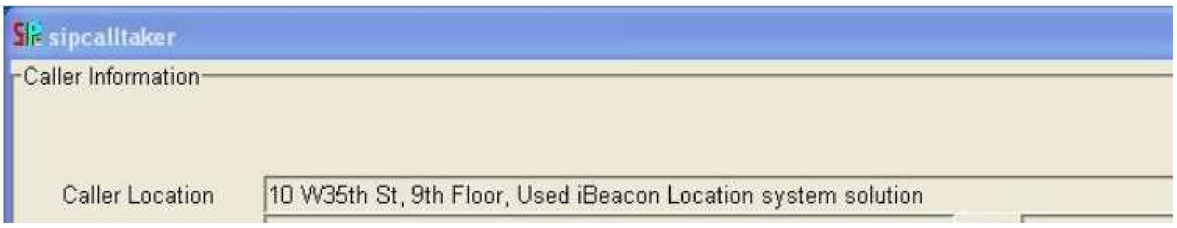
- **SBC:** Known as the Session Border Controller is in charge of controlling the signaling. It is involved in setting up, conducting and tearing down telephone calls.

It is the entry to the infrastructure and it has many security features to protect the network from malicious attacks such as denial-of-service. It also provides the ability to hide the topology of the infrastructure. When a call comes in it verifies its identity and if valid, it will redirect the call to the ESRP.

- **ESRP:** The ESRP will route the call to the specified PSAP. It will look at a table the location and will match the location with the PSAP.

- **PSAP:** Is the Public-Safety Answering Point, it is a call center responsible for answering the emergency telephone number of any emergency call. In the lab, there are three PSAPs activated and the call will go through any of these three depending on the call origination.

When the PSAP answers the call, the operator will see indoor location where the call is being made by getting the PID-LO xml imbedded in the invite.



Caller Location: 10 W35th St, 9th Floor, Used iBeacon Location system solution

Figure 15: Indoor Location in the PSAP

In this example the call is being made in the RTC Lab located in the IIT Tower (W35th St), floor 9.

2.3.3 BOSSA PLATFORM

The Bossa Platform (**B**lueT**o**oth and **S**en**s**ors **A**rray) is the central piece for the indoor location service. It has the Bluetooth arrays registered and the algorithms to find the location of the user. All is managed through an API that the Bossa Platform provides. It infrastructure is the location server, the Bluetooth arrays and a database.

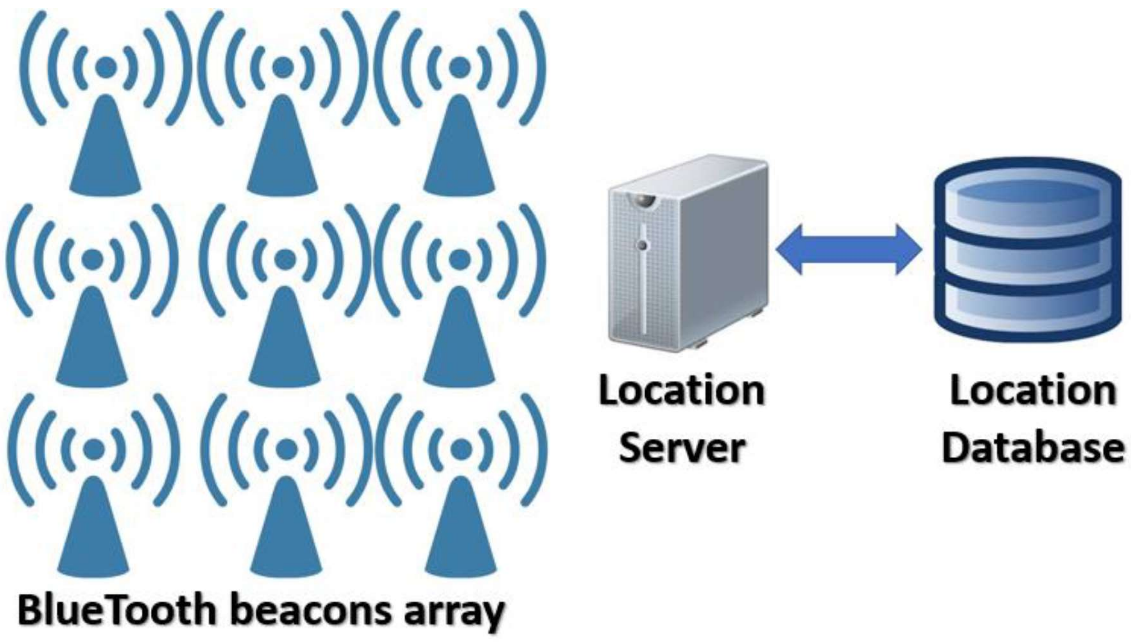


Figure 16: BOSSA Platform Module

- **BlueTooth beacons array**: Active devices that are placed in the buildings in a particular position and they are recorded in the location database. They are the ones that will help to determine the indoor location based on the power that the user receives from them. The beacons are programmed to emit once every half second their information.

- **Location server:** will apply the algorithm which will return the estimated indoor location to the cellphone.

- **Location database:** Resides in the location server and contains all the indoor location data.

For estimating the current location, the location server performs the location algorithm. This algorithm uses a process called trilateration together with non-linear least squares fitting to calculate the location of the caller. It uses the length of the sides rather than the size of the angle (triangulation) to calculate the distance.

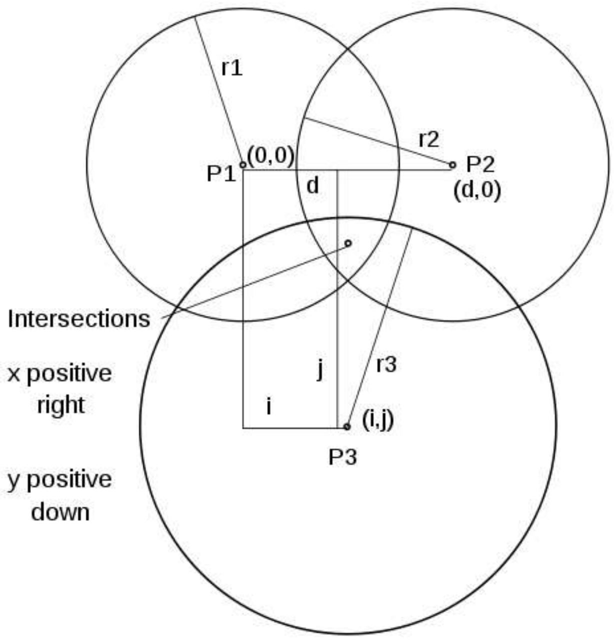


Figure 17: Location Algorithm

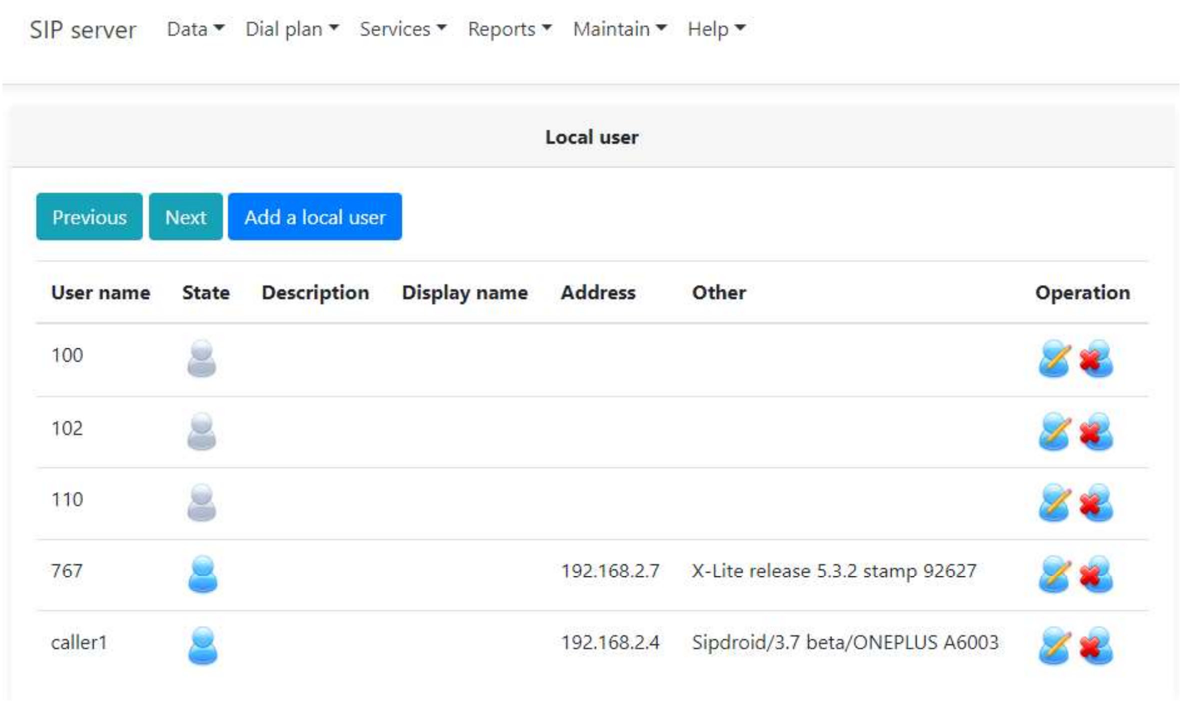
2.4 SOFTWARE USED

2.4.1 MINI-SIP-SERVER

MiniSIPServer is a server used in the Raspberry PI to play the role of the registrar and the SIP Proxy. It has a web interface where you can access all of the configurations settings and monitor what is happening in real time.

Figure 18: Mini-SIP-Server Login Page

To set up the SIP server we install a OS into the raspberry PI and we download the server from the myvoipapp web page. [17] When downloading we have two modes: SIP Server for 20 and 50 clients. It was decided to pick the 20 clients servers as we only need two user agents registered to the SIP server for testing purposes.



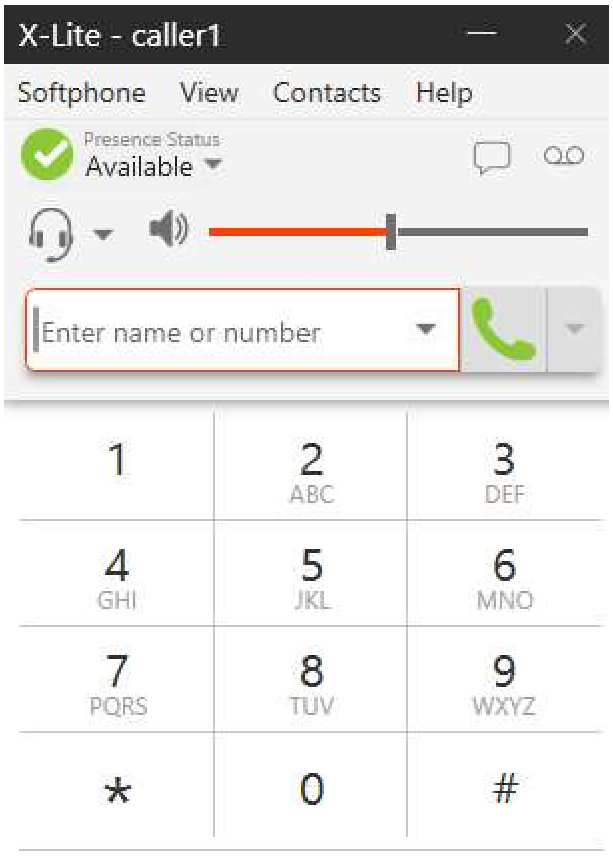
User name	State	Description	Display name	Address	Other	Operation
100						
102						
110						
767				192.168.2.7	X-Lite release 5.3.2 stamp 92627	
caller1				192.168.2.4	Sipdroid/3.7 beta/ONEPLUS A6003	

Figure 19: Mini-SIP-Server Interface

As we can see the SIP server is used in the local network where we connect both and X-lite and the application developed to test the SIP makes calls between them.

2.4.2 X-LITE

X-Lite is a proprietary freeware VoIP soft phone that uses the Session Initiation Protocol. We used it to test the emergency call application from local environments to remote environments. We first registered to our local SIP proxy and after making the test we registered to an IIT Kamailio SIP Proxy to establish the SIP Call between the X-Lite and the application.



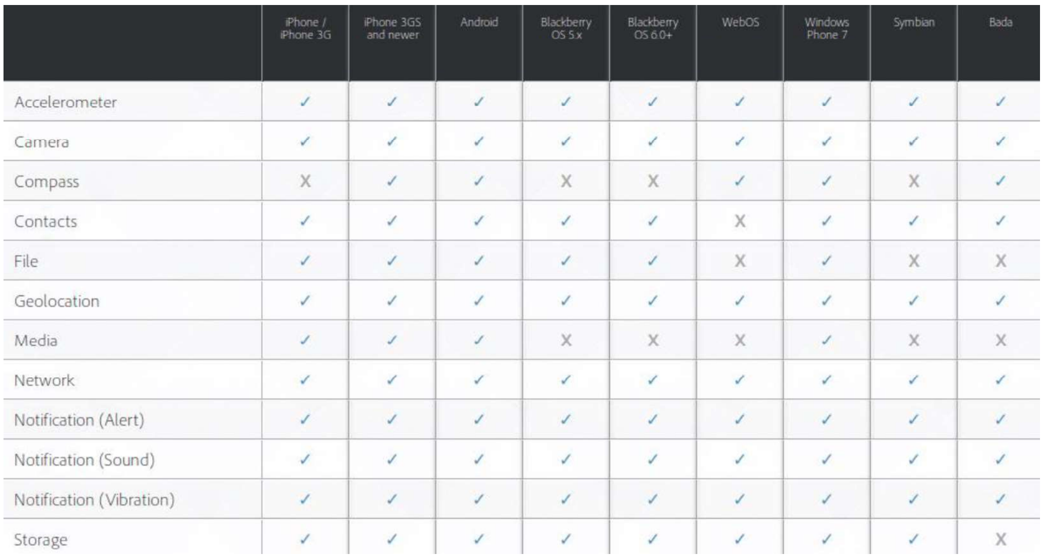
2.4.3 APACHE CORDOVA

Our first goal was to create a multiplatform application, so the emergency service could reach the entire population of the United States. This is why Apache Cordova was the first attempt in developing the emergency service application. The main reason for this is that Apache Cordova is a multiplatform framework on which you can develop applications for almost all devices, regardless of the OS they are running.

Figure 20: Apache Cordova Logo

Apache Cordova allows you to build applications for mobile devices using CSS3, HTML5, and JavaScript instead of using platform-specific APIs such as Android, iOS, or even Windows Phone. It allows to generate the application with CSS, HTML, and JavaScript code independently on the device's platform. In addition, it extends the HTML and JavaScript features so as to work with the device. To access the mobile resources there are libraries that manage the complexity behind the usage of this for the different platforms. Having basic knowledge of JavaScript and HTML enables a developer to produce an application for several platforms the same way they would build a web-based application.

Below is a table that clearly shows which services are accessible through Apache Cordova depending on each mobile device's OS, from which the application will be executed:



	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 5.x	Blackberry OS 6.0+	WebOS	Windows Phone 7	Symbian	Bada
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓	✓
Compass	X	✓	✓	X	X	✓	✓	X	✓
Contacts	✓	✓	✓	✓	✓	X	✓	✓	✓
File	✓	✓	✓	✓	✓	X	✓	X	X
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	X	X	X	✓	X	X
Network	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✓	X

Table 1: Apache Cordorva Versatility: Services vs OS

Although this was the initial idea, the framework used Evothings and; didn't support SIP calls, and we therefore decided to build the app in a native framework. Despite this, Apache Cordova has been used in the project for a test application to upload raw data into the database that, through the use of Big Data technologies, will help the algorithms' performance. The application uses the Bluetooth Service to scan for iBeacons, store the information and the power in which the smartphone receives the signal and, after 15 seconds, upload all the data into the database.

2.4.3.1 Evothings Studio

Evothings Studio is a platform that allows for the development of cross-platform mobile applications, that is, based on HTML5, JavaScript and CSS3, in a much faster way than by using other environments. The environment has several examples on how to integrate the smartphone's Bluetooth resource with the application. [18]

Figure 21: Evothings Studio Logo [18]

The development using this environment works in the following way. first, we have to download the desktop application, and the mobile application called the viewer. We create a new project, whose local folder will be, located in our computer locally. Once the project is created, we can begin to work and develop the application in our favorite editor. So far, there is no difference between it and any other developmental environment.

The main difference becomes apparent when we execute and test the code. Evothings Studio allows for the connection with a mobile device (the use of a device for the application instead of using a simulator in the same computer) by means of a token and its own application, Evothings Viewer, Evothings' own server. This way, every time we save the progress made on the project, we can see it live on the mobile device.

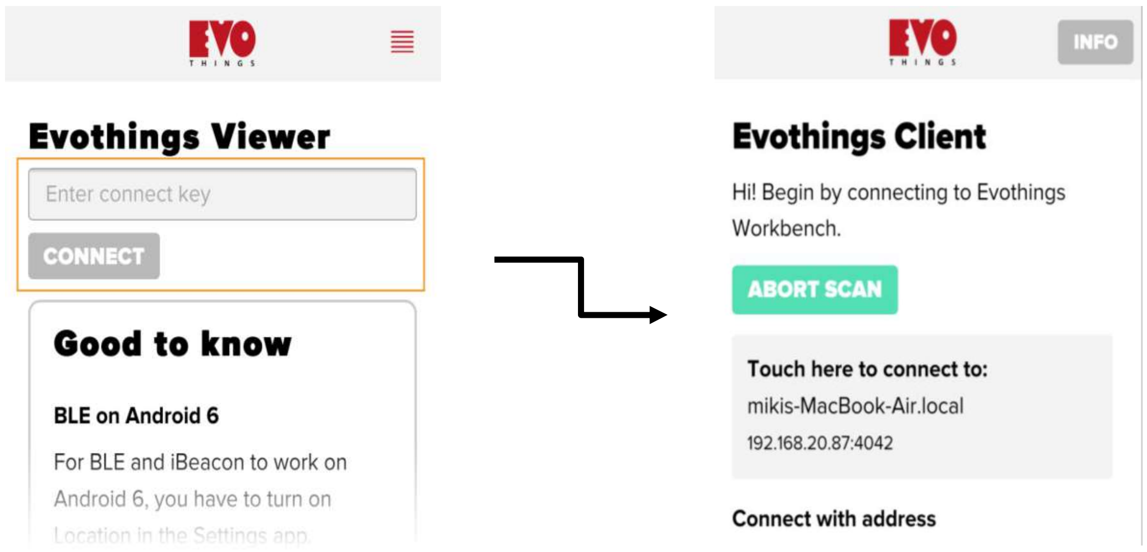


Figure 22: Conection method done by token

By using this tool, you save a lot of time when it comes to the application, as it allows you to quickly test the code that is being written. There is no need to compile and build the application, since it automatically loads when you save any file involved in the project.

2.4.4 ANDROID STUDIO

Both applications are developed using the Android platform. Before choosing the native platform in which we should develop the application, we performed an analysis of the mobile market. The main challenge when developing mobile applications is the lack of portability among the different platforms. Whenever someone is developing a mobile application they have to select which type of market to reach. This decision has to be made because sometimes there is not enough time and money to reach all platforms and it is critical to launch the application as soon as possible. Often, the approach; is to focus on Android or on iOS depending on the type of customer they want to focus on, and depending on the application that the business is launching. A study of this field has revealed that if businesses want to reach as many people as possible, they ought to target the Android market, as it is the most widespread, worldwide.

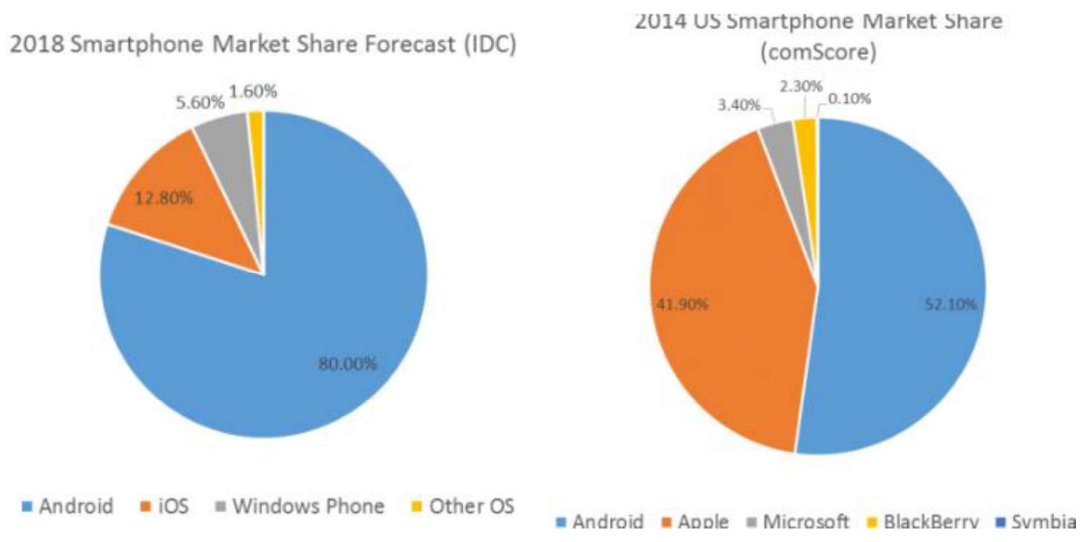


Figure 23: 2018 Smartphone Market Share Forecast [19]

Furthermore, the business should study the country where the application is going to ultimately work, as user percentages are variable. For example, although Android occupies 80% of the mobile market worldwide and iOS only 12.8%, we can see that in the United States shares is nearly equal between them. This reinforces the need to study the mobile market in the specific country the application is targeting.

After this analysis and considering that Cordova was not suitable for the timely development of an application with these characteristics, we decided to choose the Android Platform using Android Studio to develop the applications. For the emergency services we used an open source project named Sipdroid.

2.4.4.1 *Sipdroid*

The Sipdroid project is open source and is a softphone for SIP calls. [20] It was imported into Android Studio and was merged with the emergency service application to establish the call between the user and the emergency system. Sipdroid was chosen because it is open source code, and it is possible to modify the code to insert the MIME Body inside the SIP INVITE.

2.5 CLOUD COMPUTING

Cloud computing is a complementary service that provides all types of computational resources – applications, data centers, hardware, software – through the internet [21]. There are three types of cloud computing alternatives compared to the traditional model which are: PaaS, IaaS and SaaS.

2.5.1 ON-PREMISE; TRADITIONAL MODEL

Traditionally companies used to have their resources and services locally in their buildings. To make this resources and services available, the company needs to hire experts to manage and interconnect the different resources, and to design the correct redundancies so that the services are the available in case there is some failure in some resources. Also, it is important to provide the necessary resources and be able to scale the system as much as demands grows.

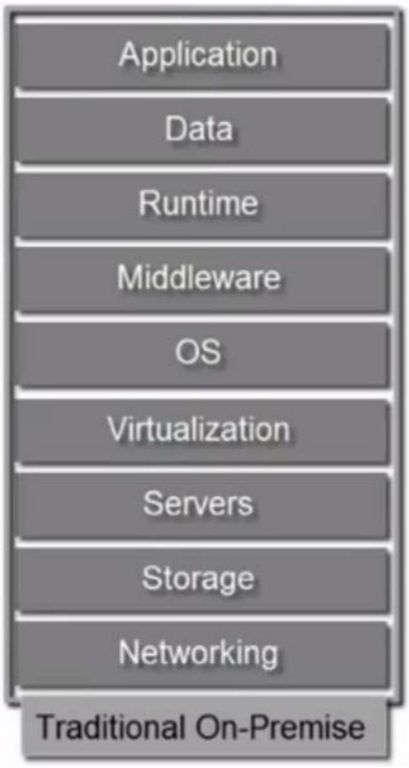


Figure 24: Cloud traditional model: On-Premise [22]

2.5.2 PAAS: PLATFORM AS A SERVICE

Platform as a Service is a cloud platform that enables developers to develop applications and services that work over the internet and they can be accessed by the web browser [28]. Using such platforms simplifies the management as no one has to deal with load balancing between the servers, also, the cloud provides a way to scale as the demands grow automatically. The developers are only in charge of the development, the deployment of the apps and the management of the data received.

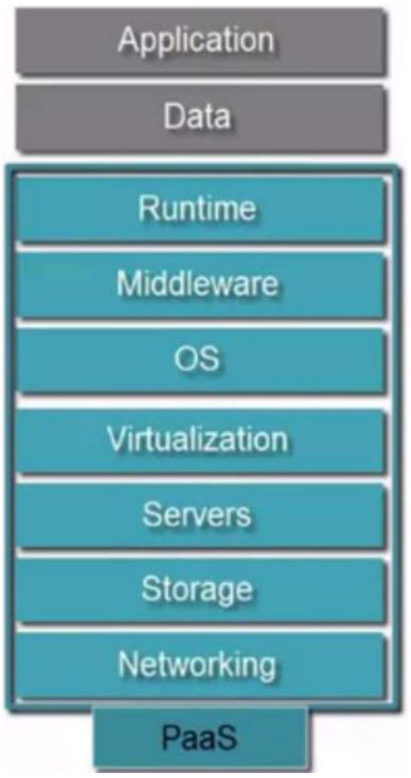


Figure 25: Platform as a Service [22]

2.5.3 IAAS: INFRASTRUCTURE AS A SERVICE

The Infrastructure as a Service is based on the utilization of infrastructure over the cloud. This means that you can rent the hardware through the cloud and use them as you want without the need to maintain the hardware. The client only manages the virtual components and is able to select what type of hardware he will be using such as: operating system, licenses, and capacity.

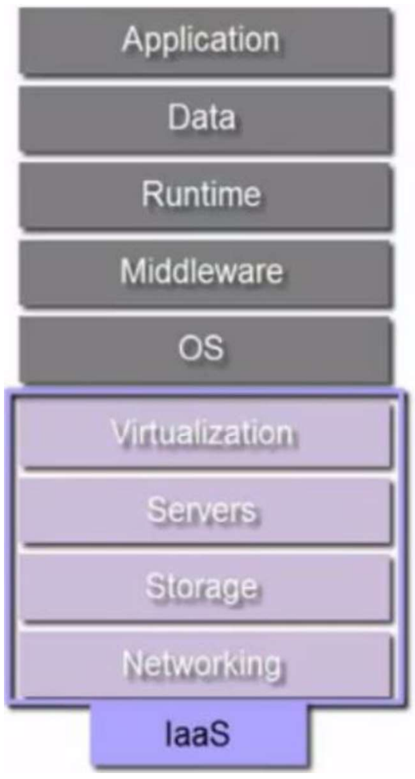


Figure 26: Infrastructure as a Service [22]

2.5.4 SAAS: SOFTWARE AS A SERVICE

Software as a Service (SaaS or Software as a Service) is a software distribution model in which a cloud service provider makes its applications available to the customer through the Internet. In this way, it prevents companies from installing such applications in their own computers and data centers, which leads to savings in the acquisition of the corresponding hardware and maintenance.

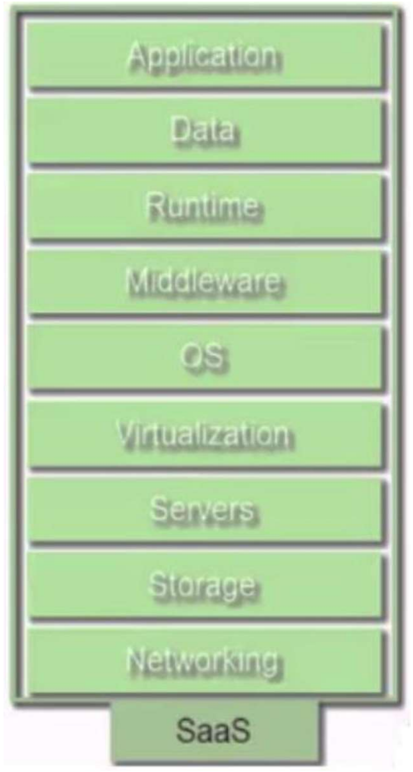


Figure 27: Software as a Service [22]

2.5.5 DIFFERENCES BETWEEN ON-PREMISE, IAAS, PAAS AND SAAS

As shown in Figure 28, the traditional model offers more personalized services, which leads to greater investment due to the demanding management and maintenance of the infrastructure. Making resources available and operational is a huge investment of time. On the other hand, cloud services allow greater optimization of time and resources, since they have ready-to-serve services that are immediate (unlike the traditional model, which inevitably involves the corresponding delay of the set-up of the acquired resources). For this reason, companies are migrating their infrastructures to the cloud as it tends to be more effective. This change of paradigm involves the developers to change the way they code to adjust to the cloud, often applications that run on-premise need to be changed before moving to cloud.

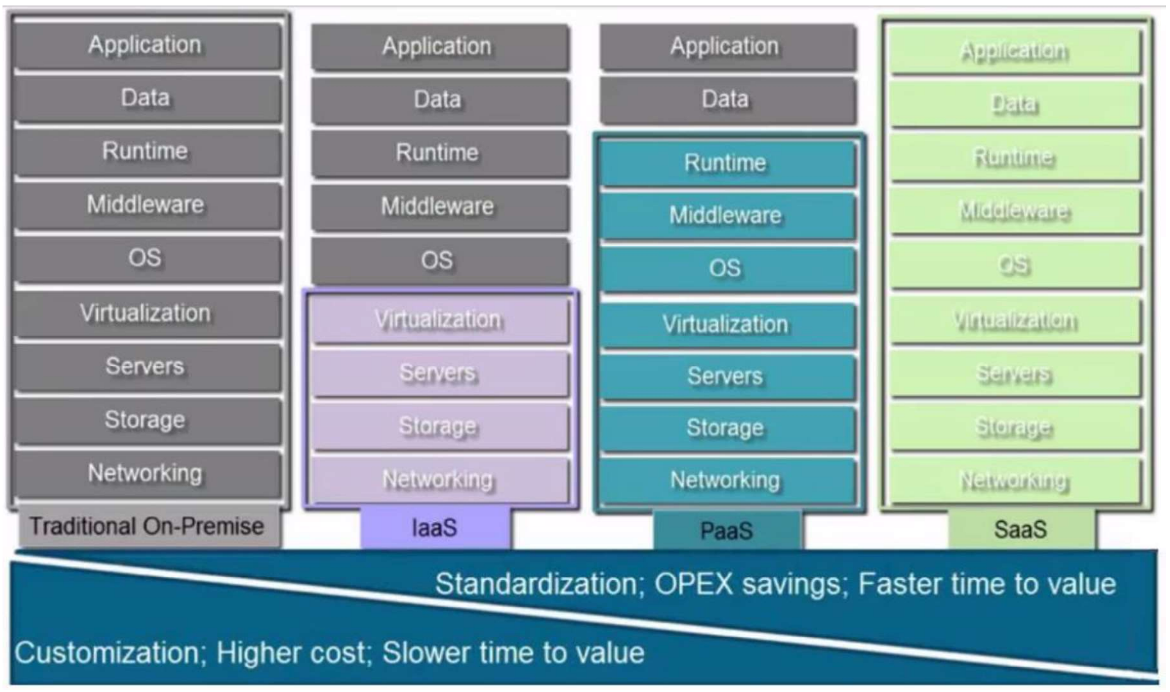


Figure 28: Comparison between On-Premise, PaaS, IaaS and SaaS [22]

2.6 IBM BLUEMIX

For the project at hand, we used IBM Bluemix. A platform as a service in which we could select the environment that we wanted to work in, and were provided with several tools to make it easy to deploy. [23]

Inside IBM Bluemix we found applications and services. The application often uses services that are already created and ready to use. Applications are unique, whereas services can be used across different application, simplifying the use of shared resources. This makes the development of applications fast, but it requires the developer to learn how to use the tools provided by the cloud.

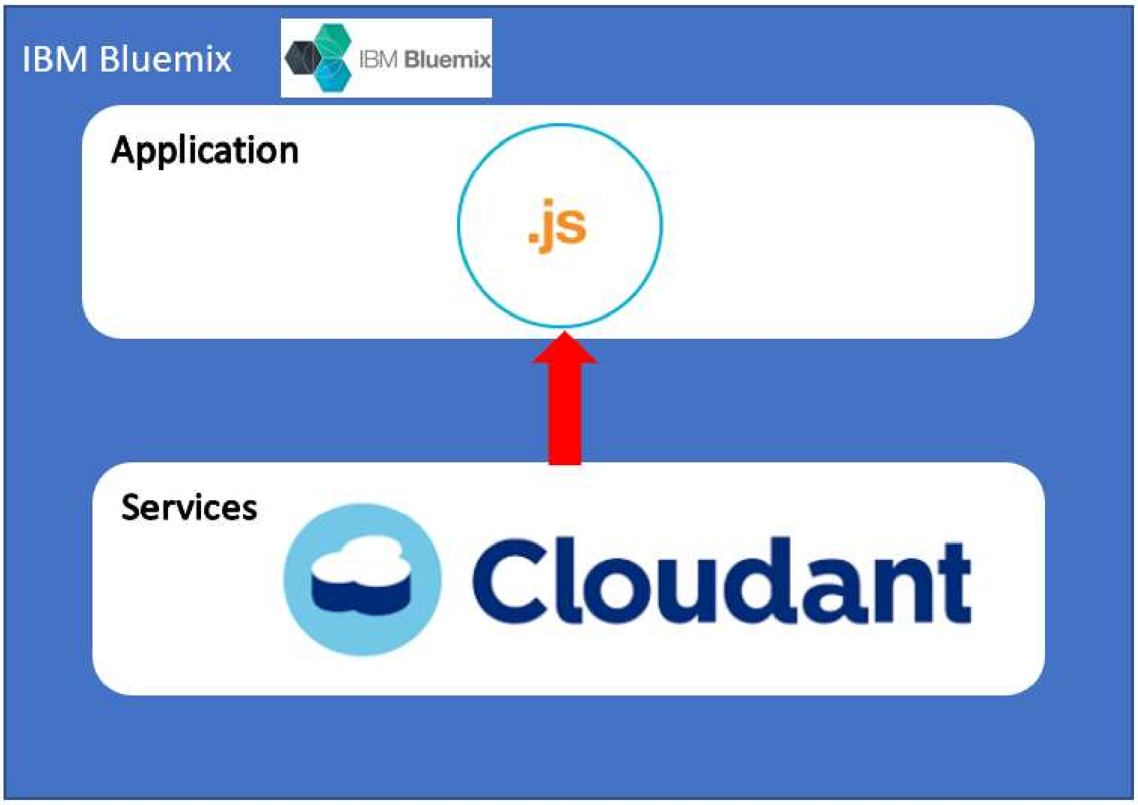


Figure 29: IBM Bluemix Architecture

In the figure above, we can see the scheme we followed for every application. Our application uses node.js for the backend and the database Cloudant as a service to provide the client with all the data they need.

2.7 NODE.JS

Node.js is an open source multiplatform environment. Its interpreter is JavaScript which changes the notion of how a server should work. The main goal for node.js is to allow the programmer to build highly scalable applications that can manage thousands of simultaneous connections in the same physical machine. For this purpose, we have created our backend in Node-red which will be describe in the next section.

2.8 NODE-RED

Node-RED is an open-source visualization tool developed by IBM that enables to connect all the elements created for the Internet of Things. These elements can vary from hardware devices, online services or APIs. In our case, we will create APIs which can be accessed by an HTTP request.

Node-RED is provided by a web browser interface that enables to create flow of events and interconnect them through a light work and development environment. It abstracts all the coding behind nodes and it is very simple to visualize what the application does. Many times, the code for a whole application can be reduced from a thousand lines of code to just a hundred. In the next sections we are going to explore the nodes used in this project.

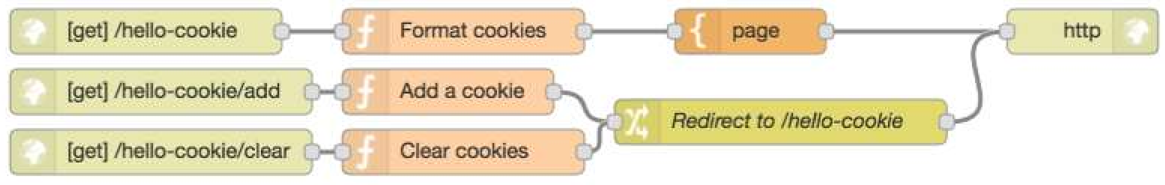


Figure 30: Example of a node-red flow [24]

2.8.1 NODE-RED INPUTS

Figure 31: Inject Node [25]

Injects a message into a flow either manually or at regular intervals. The message payload can be a variety of types, including strings, JavaScript objects or the current time. [25]

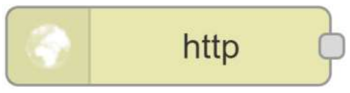


Figure 32: HTTP Node [25]

Creates an HTTP end-point for creating web. [25]

2.8.2 NODE-RED OUTPUTS

Figure 33: Debug Node [25]

Displays selected message properties in the debug sidebar tab and optionally the runtime log. [25]

Figure 34: HTTP Node [25]

Sends responses back to requests received from an HTTP Input node. [25]

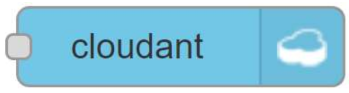


Figure 35: Cloudant Node [25]

A simple Cloudant output node. Stores the message in a chosen database. The application has to use a cloudant service for this purpose. [25]

2.8.3 NODE-RED INTERMEDIATE NODES

Figure 36: Function Node [25]

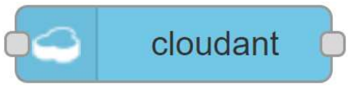
A node used to write code to personalize the Flow and the behavior. [25]

Figure 37: CSV Node [25]

Converts between a CSV formatted string and its JavaScript object representation, in either direction. [25]

Figure 38: JSON Node [25]

Converts between a JSON string and its JavaScript object representation, in either direction. [25]

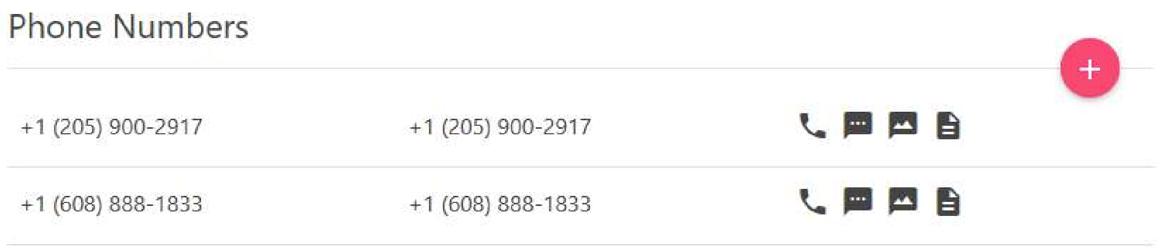


A node for searching documents in a Cloudant database.

Figure 39: Cloudant Node [25]

2.9 *SIGNALWIRE CLOUD*

SignalWire is a new cloud platform for building advanced communications products, applications and, enterprise infrastructure. [26]



Phone Numbers

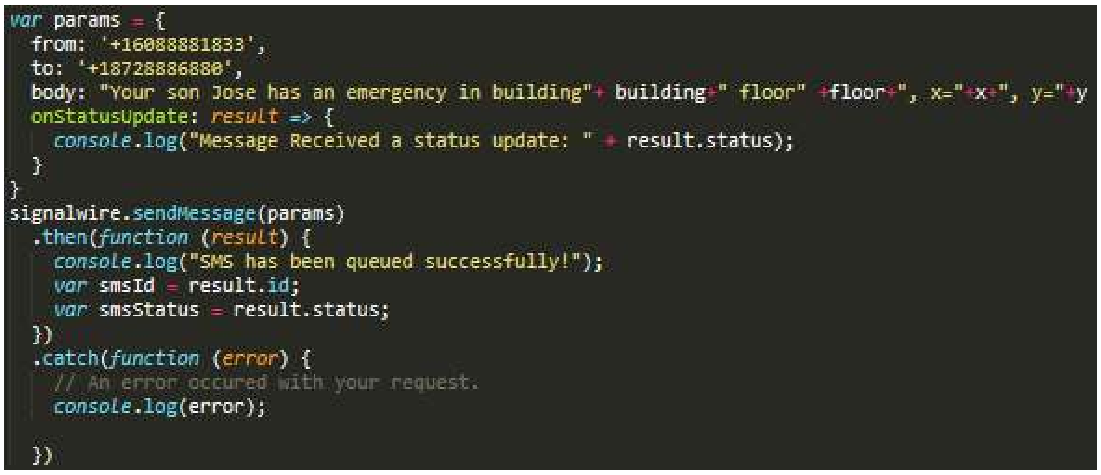
+1 (205) 900-2917	+1 (205) 900-2917
+1 (608) 888-1833	+1 (608) 888-1833

Figure 40: Phone numbers purchased in SignalWire

The SignalWire cloud enables you to purchase mobile numbers and us them for different purposes. In this section, we are going to talk about the services that have been implemented in this project.

SignalWire Services:

- **SMS Message:** The user can send a message by just inserting the destination phone and the message in the body.

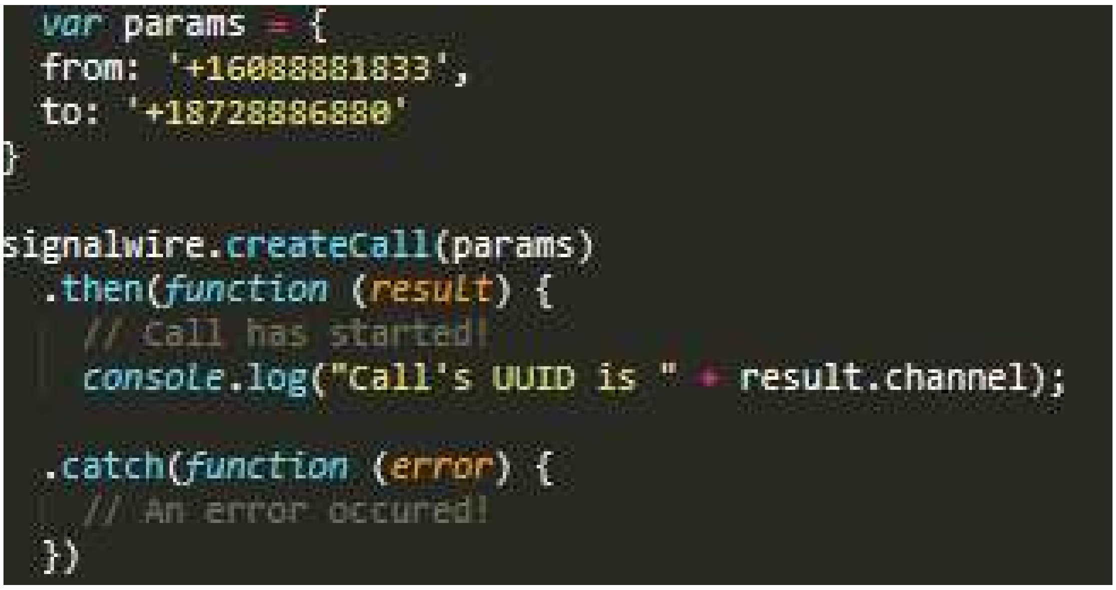


```
var params = {
  from: '+16088881833',
  to: '+18728886880',
  body: "Your son Jose has an emergency in building"+ building+" floor" +floor+", x="+x+", y="+y
  onStatusUpdate: result => {
    console.log("Message Received a status update: " + result.status);
  }
}
signalwire.sendMessage(params)
  .then(function (result) {
    console.log("SMS has been queued successfully!");
    var smsId = result.id;
    var smsStatus = result.status;
  })
  .catch(function (error) {
    // An error occured with your request.
    console.log(error);

  })
```

Figure 41: JavaScript code for SMS service

- **Call Service:** A call is made by inserting the destination number. The UUID is important if someone wants to add service to the call such as, text-to-speech.



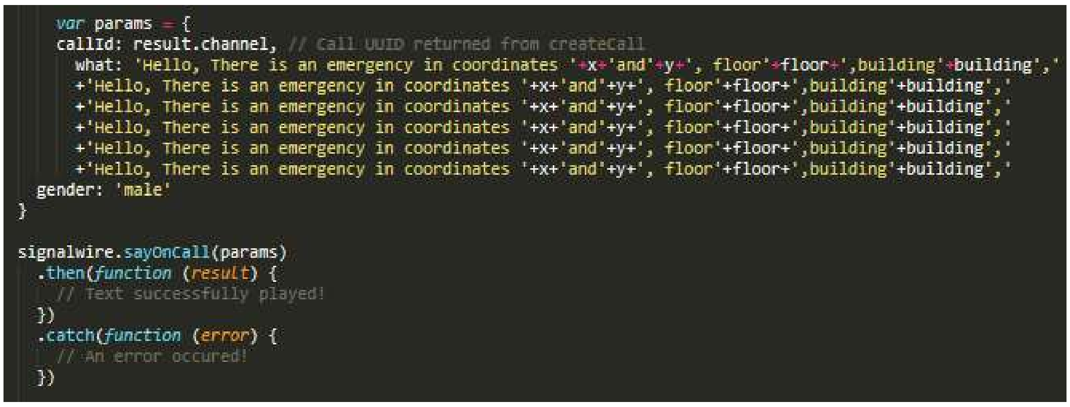
```
  var params = {
  from: '+16088881833',
  to: '+18728886880'
}

signalwire.createCall(params)
  .then(function (result) {
    // Call has started!
    console.log("Call's UUID is " + result.channel);

  .catch(function (error) {
    // An error occured!
  })
```

Figure 42: JavaScript code for call service

- **Text-To-Speech Service:** The call can be made by sending automated voice messages to the destination. In our application the call will send the indoor location information to the 911 services.

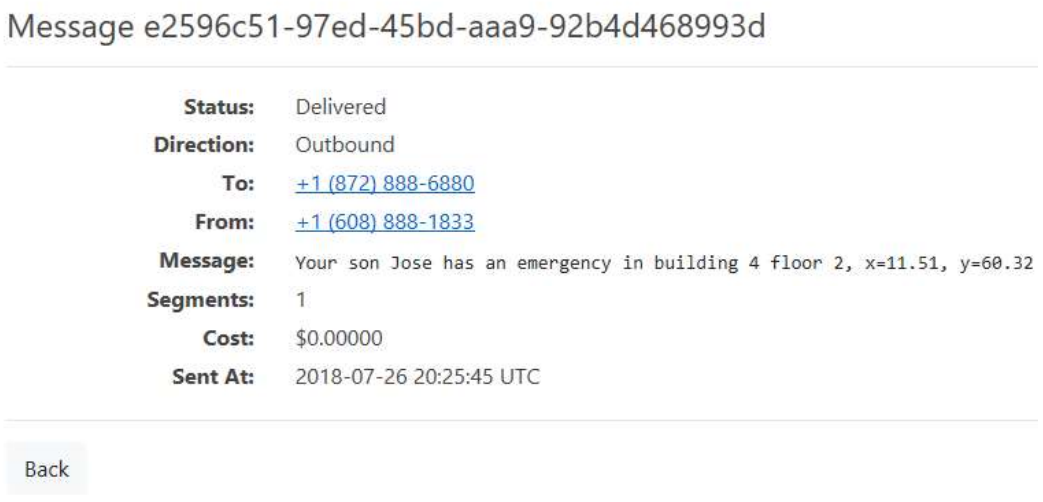


```
    var params = {
    callId: result.channel, // Call UUID returned from createCall
      what: 'Hello, There is an emergency in coordinates '+x+'and'+y+', floor'+floor+',building'+building','
      +'Hello, There is an emergency in coordinates '+x+'and'+y+', floor'+floor+',building'+building','
      +'Hello, There is an emergency in coordinates '+x+'and'+y+', floor'+floor+',building'+building','
      +'Hello, There is an emergency in coordinates '+x+'and'+y+', floor'+floor+',building'+building','
      +'Hello, There is an emergency in coordinates '+x+'and'+y+', floor'+floor+',building'+building','
      +'Hello, There is an emergency in coordinates '+x+'and'+y+', floor'+floor+',building'+building','
  gender: 'male'
}

signalwire.sayOnCall(params)
  .then(function (result) {
    // Text successfully played!
  })
  .catch(function (error) {
    // An error occured!
  })
```

Figure 43: JavaScript code for text-to-speech service

In the SignalWire cloud, the owner can see the history records of the numbers' activity.



Message e2596c51-97ed-45bd-aaa9-92b4d468993d

Status:	Delivered
Direction:	Outbound
To:	+1 (872) 888-6880
From:	+1 (608) 888-1833
Message:	Your son Jose has an emergency in building 4 floor 2, x=11.51, y=60.32
Segments:	1
Cost:	$0.00000
Sent At:	2018-07-26 20:25:45 UTC

Figure 44: Record of messages in SignalWire

Chapter 3. STATUS OF THE WORK

3.1 COMMERCIAL APPLICATIONS

There are many solutions for indoor and outdoor location. Most of the commercial applications have proprietary solutions to attack this issue and their use case is different for those specified in this project. There is one application set to work with the emergency services, but it does not provide the tools to be integrated with the NG911 module. We will go through the commercial apps in this section.

3.1.1 GINA GO

Gina GO is a location tracking app very similar to our campus security application that includes a panic button to increase safety. The Gina GO application is available for the big OS such as Android and iOS. It uses the phone Internet connection to share the location to send a panic signal to the security operators or chosen recipients. The application offers real-time tracking, private mode (so that the app does not reveal the location through the entire time), SOS button and two-way messaging. It is also made for other services that are out of the scope for this project such as vehicle tracking. [27]

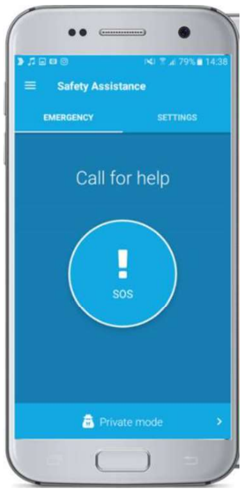


Figure 45: Gina GO Application [27]

3.1.2 BUTTON MESSENGER APPS

Asio is a software company that has announced new messaging concept called Button Messaging. The company has launched the Button Messenger apps. These apps are designed to send SMS messages which contain location information. The content of these messages could be anything, but the application allows us to locate the user with a map using the same application. [28]

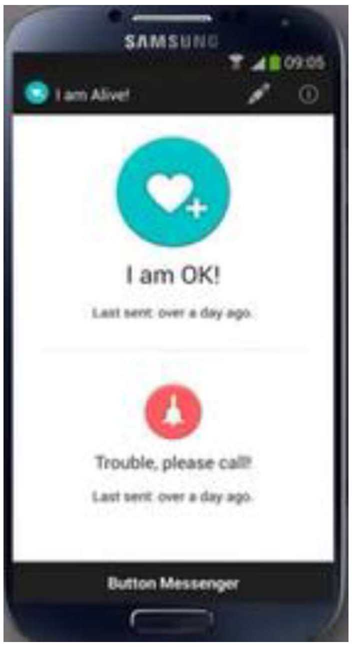


Figure 46: Button Messenger Application [28]

3.1.3 NEXTNAV

NextNav is a company involved in the indoor location for emergency service, it provides a solution that works indoors, including altitude location with floor-level precision. NextNav's Context and Visualization service, though a unique set of apps and APIs, allows 3D tracking of first responders in difficult environments and dangerous situations for improved safety and situational awareness. [29]

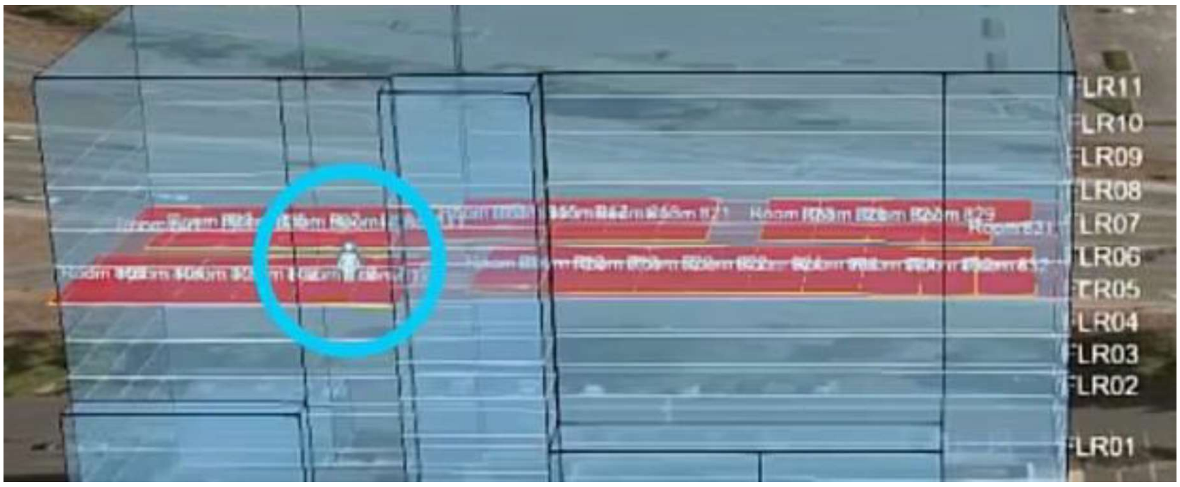


Figure 47: Netx Nav's 3D map view solution: Indoor Location [29]

NextNav offers a next-generation solution for E911, enabling the location of wireless devices where they are used most frequently: indoors. A wireless E911 call placed from an indoor location, though, has no location accuracy standards and often defaults to extremely coarse location methods. NextNav's solution integrates with existing systems at a minimal cost to carriers and leverages existing call flows to public safety answer points (PSAPs) [29]. The main drawback of using Next Nav's solution is that the device has to have NextNav's technology, which makes it difficult for it to reach to the majority of users.

3.1.4 POLESTAR

The Polestar application helps to determine the indoor location of a user inside a building. More than an emergency service, it is a "where I am" service. It provides an interactive map to guide the user through the building or show points of interest. This solution works with proprietary beacons and their own cloud platform. [30]

Figure 48: Polestar solution to indoor location with proprietary beacons [30]

3.1.5 CONCLUSION

Although there are many solutions, none of them offers an open environment. Even the closest one, NextNav, requires special devices to work with this environment. The solution this project proposes is open source where we only need the beacons to be registered in the BOSSA Platform. Moreover, looking at the campus security application, there are many solutions for outdoor location, but in the project, we suggest a closed dedicated environment between students and public safety.

3.2 RESEARCH PAPER

The research project worth mentioning is the *"Dispatchable Indoor Location for Mobile Phones Calling for Emergency Services"* [31]. The paper describes a proof of concept that provide indoor location who calls a NG911 public safety answering point using a smartphone. It was written at the IIT's Real-Time Communications Lab, the same one where this project was carried out. The paper describes how to get indoor location based on Wi-Fi access points that have configured Wi-Fi to be public. Therefore, the proposed solution is not applicable to apartment buildings, for instance, as each unit will provide its own solution to the Internet access. Furthermore, the latency derived by adding the indoor location takes between 30 to 40 seconds.

The paper states that future work needs to be carried out in order to integrate their solution with Bluetooth beacons, so that the building management can install these beacons throughout the edifice. This showcases why the project at hand has aimed and managed to reduce the latency by 50%, down to 15 seconds.

Chapter 4. WORK DEFINITION

4.1 JUSTIFICATION

This is a project of great interest to the emergency services, providers of cellular services, the FCC and the IIT. The solutions presented and developed in this project will help both save lives and improve campus safety. It is intended to become a global solution within the United States and for all companies to incorporate location in their emergency calls.

The justification of this project becomes apparent when reviewing the history emergencies and location in the United States. Here is relevant news:

"In California, more than half of cellphone calls didn't transmit location to 911 from 2011 to 2013, and it's getting worse. Last year, about 12.4 million, or 63%, of California's cellphone calls to 911 didn't share location." [32]

"In Texas, two-thirds of cellphone calls in a sample of calls from major cities – including Austin and Houston – reached 911 without an instant fix on location from 2010 through 2013." [32]

Due to the lack of location information, emergency services often reach the location when it is already too late. Many lives could have been saved if they had been in possession of the information earlier.

"*More than* ***10,000 people****, who would otherwise be saved,* ***die every year*** *when calling 911 from a cellphone because emergency dispatchers can't* ***get a quick and accurate location on them***" [33]

The location emergency block will be beneficial to the FCC and the four largest cell phone carriers, who are already doing their best to address this problem. They have been working together on a new federal court ruling that requires carriers to increase the percentage of cellphone calls to 911 that transmit location data. This rule states that the calls' delivery of location data must be 40% by 2017 and 80% by 2021. If the suggested solutions were to be implemented inside of buildings too, they would contribute towards this goal, as the figures for 2015 show that location data was available in under 40% of emergency calls in many communities. [32]

Student location in the campus block, will be very beneficial to the IIT community because students will feel safer knowing that there is a tool that acts rapidly against assaults.

4.2 OBJECTIVES

The project can be seen as two large blocks:

Block 1: Emergency Services at the Illinois Institute of Technology.

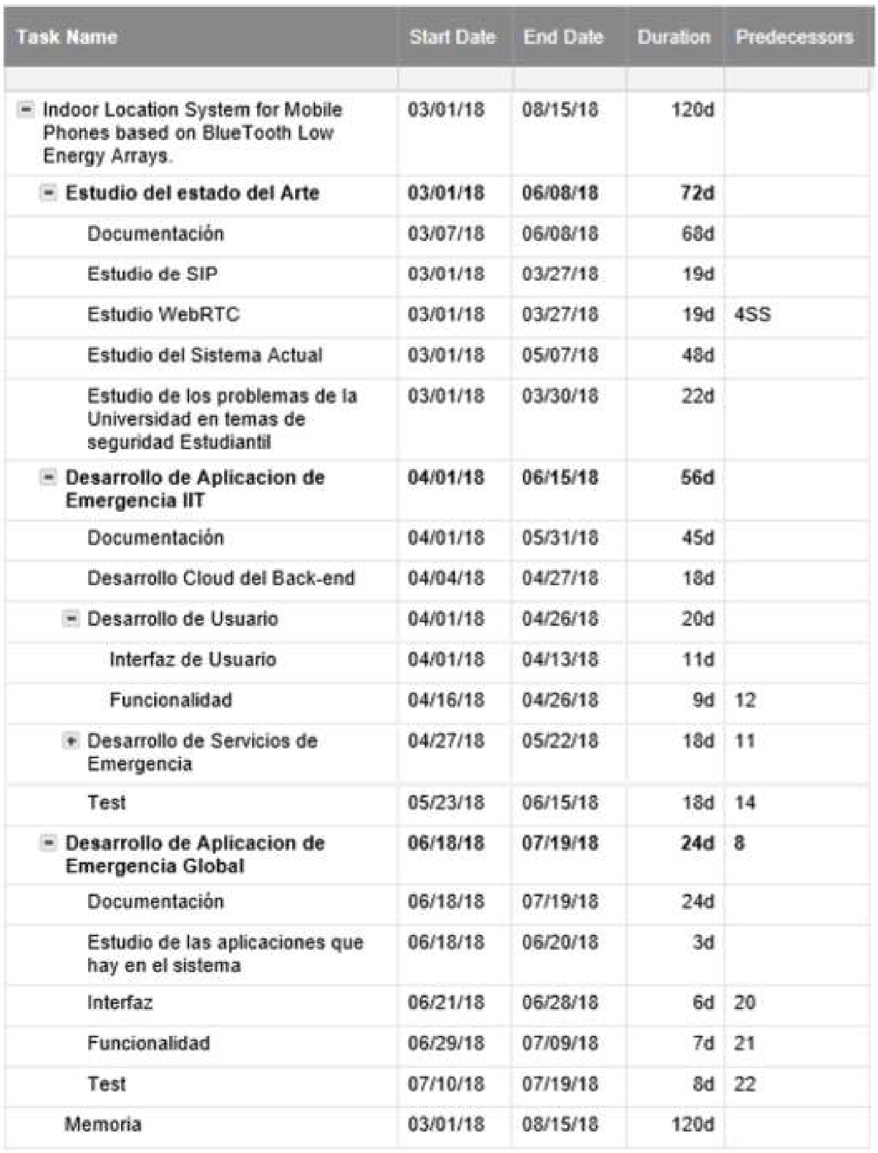
- Propose an alternative system to the current one that is more efficient and helps improve the security on campus.
- Develop a simple and user-friendly application so that everyone knows how to use it and make it functional so that it reaction time is quick.

Block 2: Indoor location in emergency calls

- Provide the necessary tools so that the emergency services receive the indoor location of the person and are thus better equipped to save lives.
- Study the current infrastructure and create an application that interacts with it.
- Delivery of location data in 80% of cellphone calls.
- Reach as many users as possible.
- Provide a test environment to improve algorithms accuracy.

4.3 METHODOLOGY

The methodology to be used is a scrum methodology. Each week the objectives are defined, and an incremental development is made. The chronogram presents 3 differentiated phases: State of the Art, Development of Emergency Application IIT and the Development of Global Emergency Application. In each of these phases, scrum will be applied independently. The project will last four months.



Task Name	Start Date	End Date	Duration	Predecessors
Indoor Location System for Mobile Phones based on BlueTooth Low Energy Arrays.	03/01/18	08/15/18	120d	
Estudio del estado del Arte	**03/01/18**	**06/08/18**	**72d**	
Documentación	03/07/18	06/08/18	68d	
Estudio de SIP	03/01/18	03/27/18	19d	
Estudio WebRTC	03/01/18	03/27/18	19d	4SS
Estudio del Sistema Actual	03/01/18	05/07/18	48d	
Estudio de los problemas de la Universidad en temas de seguridad Estudiantil	03/01/18	03/30/18	22d	
Desarrollo de Aplicacion de Emergencia IIT	**04/01/18**	**06/15/18**	**56d**	
Documentación	04/01/18	05/31/18	45d	
Desarrollo Cloud del Back-end	04/04/18	04/27/18	18d	
Desarrollo de Usuario	04/01/18	04/26/18	20d	
Interfaz de Usuario	04/01/18	04/13/18	11d	
Funcionalidad	04/16/18	04/26/18	9d	12
Desarrollo de Servicios de Emergencia	04/27/18	05/22/18	18d	11
Test	05/23/18	06/15/18	18d	14
Desarrollo de Aplicacion de Emergencia Global	**06/18/18**	**07/19/18**	**24d**	8
Documentación	06/18/18	07/19/18	24d	
Estudio de las aplicaciones que hay en el sistema	06/18/18	06/20/18	3d	
Interfaz	06/21/18	06/28/18	6d	20
Funcionalidad	06/29/18	07/09/18	7d	21
Test	07/10/18	07/19/18	8d	22
Memoria	03/01/18	08/15/18	120d	

Figure 49: Gantt diagram overview

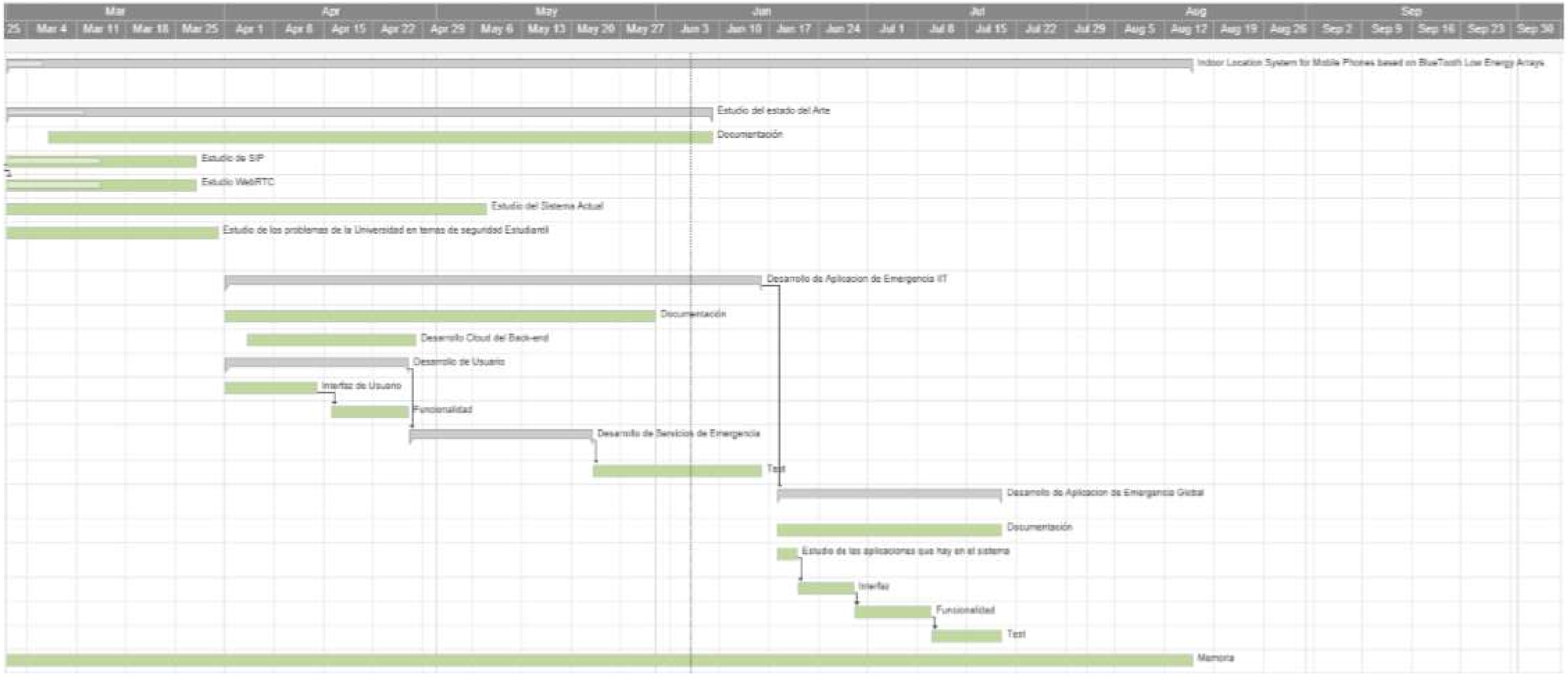


Figure 50: Gantt diagram

4.4 ECONOMIC ESTIMATE

AS per above Gantt diagram, the project covers a times span from March to mid-August. Each day counts marks as eight workings hour.

4.4.1 COST: HUMAN RESOURCES

To calculate the cost per day we have selected the roles of Senior Software Engineers and Mobile Developers in the United States. In Indeed, the average salary for a mobile developer is $103,304 annually and for the senior software engineer $117,458. [34]

$$\text{Eq 1: Salary}_{hour} = \frac{Salary_{annual}}{40h/w*52w/year}$$

The following table determines the salary per hour:

	Annual Salary $	Hour Salary $	Daily Salary $
Mobile Developer	103,304	50	400
Senior Software Engineer	117,458	56.4	451.76

Table 2: Engineers Salary description

The budget of the human resources dedicated to the project is detailed in following table:

Task	Total Days	Senior Engineer	Mobile Developer
Project Plan	15	10	90
Analysis	20	5	95
Design	30	5	95
Implementation	60	0	100
Documentation	28	0	100
Total	153	4	149

Table 3: Total days worked for each engineer

The total cost of the project in human resources is the following:

	Days	Salary $
Mobile Developer	149	59,600
Senior Software Engineer	4	2,167
Total		61,767

Table 4: Budget dedicated to human resources

4.4.2 COST: HARDWARE

We will only consider the cost of the computer and smartphone used. We had free access to infrastructure like ESINet and the BOSSA Platform, which were not strictly part of the project, although we did use them. The computer is a Toshiba Satellite, Intel Core i5 with Windows 7. The smartphone is a OnePlus 6 with 128GB of storage. We have to calculate the depreciation of the computer to calculate the real cost. For this, we will calculate the amortization based on Eq 2.

$$\text{Eq 2: } f(x) = \frac{A}{B} * C * D$$

Where f(x) is the amortization cost, A is the duration in months, B the depreciation (in computers is 60 months), C the real cost and D the use in percentage.

	Duration	Depreciation	Cost	%Use	Amortization $	Final Cost f(x) $
Toshiba Satellite	4 months	60 months	$1000	95%	63.33	936,67
One Plus 6	4 months	60 months	$550	70%	25.66	524,33
Total						1,461

Table 5: Budget dedicated to Hardware

4.5 COST: SOFTWARE

All of the software used was free. IBM Bluemix provides a free trial for six months and SignalWire is in trial. Therefore, there are no cost associated with the software.

4.5.1 INDIRECT COST

Indirect costs are estimated and calculated ahead of direct costs, representing 15% of the latter, and amounting to $9,484.24.

4.5.2 ECONOMIC SUMMARY

From the economic cost explained above we can now calculate the total cost of the project.

Description	**Cost $**
Human Resources	*61,767*
Hardware	*1,461*
Software	*0*
SubTotal	***63,228***
Indirect	*9,484.24*
Total	***72,712.45***

Table 6: Cost of the project detailed

The total budget for this project amounts to **seventy-two thousand seven hundred and twelve dollars and forty-five cents**.

Chapter 5. PROJECT IMPLEMENTATION AND ANALYSIS OF RESULTS

The project consists of four applications developed to attack the problem of locating emergency calls:

The first one to be developed is an Android application to locate students at the Illinois Institute of Technology. Although it is intended specifically for the IIT's campus security, the use of said application can be extended to all kinds of users as long as the emergency services also have it installed on their equipment and the owner of the system gives bot civilians and the emergency services access to it.

The second is also an Android application that uses SIP to create a call. The application scans for beacons in the building and then sends them to the BOSSA Platform. The BOSSA Platform will respond with the location in a JSON format and embed the location in the SIP INVITE body. The call goes through the ESINet and will redirect the call to the correct PSAP. After that, the call is established between the user and the emergency services.

The third application is developed in Apache Cordova using Evothings. The application sends raw data to the database so that a study of the algorithms may be conducted in order to improve its accuracy.

The fourth application is also developed in Apache Cordova using Evothings. It is similar to the second application mentioned above, except that it does not use SIP to create the call, but creates an automated call through the PSTN containing information pertaining to indoor location with the help of SignalWire cloud API, which in turn provides tools for text-to-speech, call service, and SMS service.

In this chapter we are going to describe each of the applications in detail.

5.1 MOBILE APPLICATION FOR CAMPUS SECURITY

The mobile application intended for campus security is one that can be divided into two separate applications. With the credentials provided in the login page, the application will know if the user is a student or an emergency assistant and will display correspondent layout. The application uses IBM Bluemix for the backend and it has a database with all of its users' credentials stored.

Figure 51: IIT Security Logo

5.1.1 DESIGN OF CAMPUS APPLICATION

The following are the requirements for the entire application's development:

- The application must identify if the user is a student or the authority.
- Back-end for collecting data.
- Back-end for retrieving data.
- Stores user credentials locally so they do not have to log in every time
- Panic button must be an activity with just that purpose.
- Local Map view displaying panic pins.
- Brief pin description on click (assault, robbery, sexual assault, false alarm)
- Color coded pins: to identify the type of assault.
- Application service that periodically checks for updates.

5.1.1.1 Design of Student application

The application is intended to be play and go. In order to achieve this goal, the user need only login once for their credentials to be stored. Once stored, if the application were closed, it would remember the user's information when reopened. However, you may always exit the application through the settings menu to erase your credentials and start fresh. This is highly important when it comes to an emergency situation because reaction time is vital.

5.1.1.1.1 Student Use Case and Activity Diagram

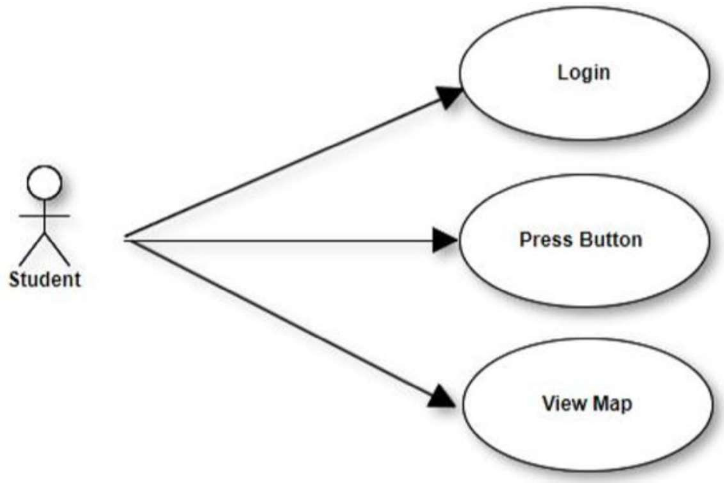


Figure 52: Student Use Case

- **Login:** Login will be mandatory the first time the application is opened. Once completed, the application will remember the user's credentials so that, if student were to find themselves in an emergency situation, they could press the panic button immediately upon opening the application.

- **Press Button:** When pressed, the panic button notifies the authorities straight away. This is the main purpose of the application, the aim being for the appropriate measures to be taken on time.

- **View Map:** There is a record of every location where the panic button has been pressed so that students can view those locations on the map.

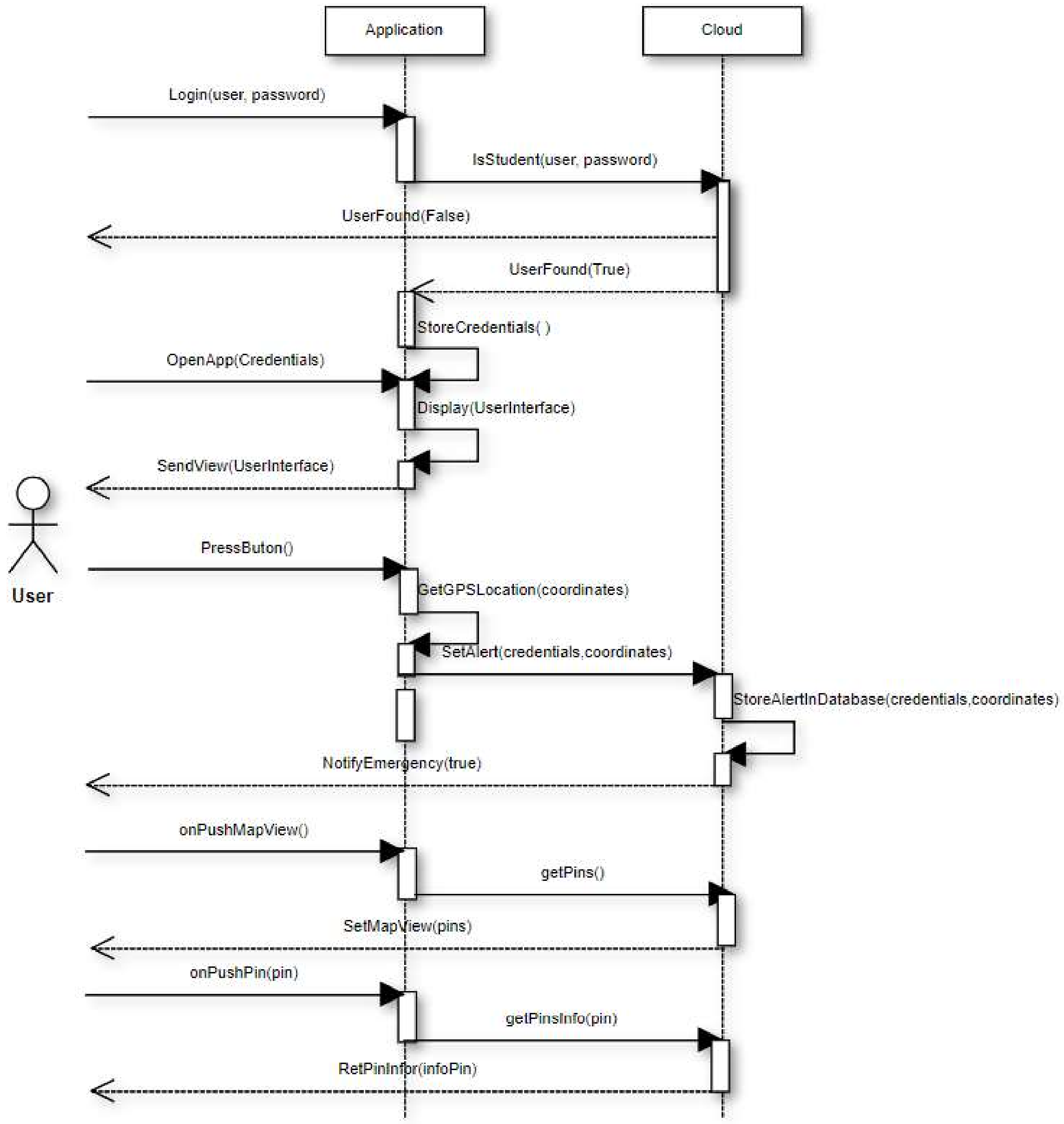


Figure 53: Student Activity diagram

When the application starts up, it does a prior credential check before loading the view. If no credentials have been input to the application, the user must log in to the system. Once logged in, the application will send this information to the cloud in order to authenticate the user and to determine if it is a student or an emergency assistant. If the user is not in the database, an error will be sent back and they he will have to undergo the authentication process once again with the right credentials. Once the later are verified they are stored in the application using SharedPreferences, and sending the user the appropriate view (in this case, the student interface).

Once the application has stored its credentials, the user can do one of two things:

- **Press button**: In this case the application will use the GPS service to determine the user's location and will send it along with the credentials. The cloud will store this information in a database so that the emergency services can get the alert.

- **View Maps**: When this option is selected, the application will send a notification to the cloud asking for feedback on all of the incidents recorded in the database. It, in turn will return all the information to the application, where it will be displayed for the student to see.

5.1.1.2 Design of Emergency Assistant Application

The application is intended to receive immediate notifications once a student has pressed the panic button. As was mentioned before, to achieve this, the login need only be done once for the credentials to be stored. Once completed, if the application closes, the application will remember the user's identity when the application reopens. The emergency assistant can have this app running in a smartphone when he/she is on duty and the notifications will automatically appear. If the notification is pressed, Google Maps will display the fastest route to the emergency point. After providing this assistance, the emergency services must specify the type of alert received in order to erase the notification and store it as part of the history records.

5.1.1.2.1 Student Use Case and Activity Diagram

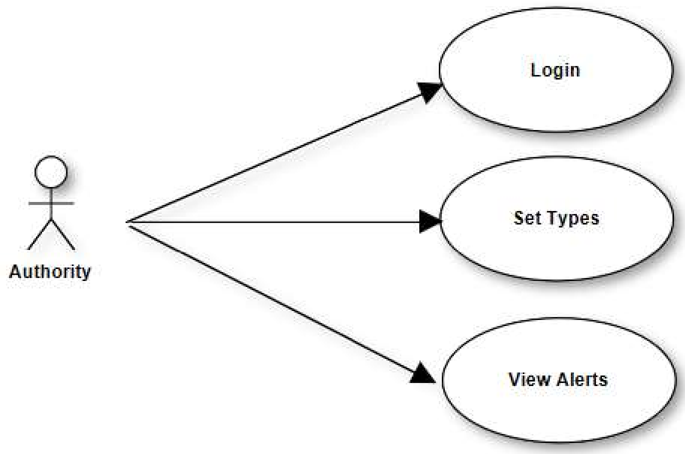


Figure 54: Authority Use Case

- **Login:** Login will be mandatory for the first time the Application is opened. Once completed, the application will remember the user's credentials so that it can immediate retrieve the authority layout.

- **View Alerts:** Authorities have access to new alerts and can view all the events that have taken place. There are two types of events: Recent Events and Documented Events. If the panic button is clicked by a student, a service running in the application's background will detect the change and update the list, as well as send a notification to the user.

- **Set Types:** Once they have attended to the emergency, they have to set the emergency type.

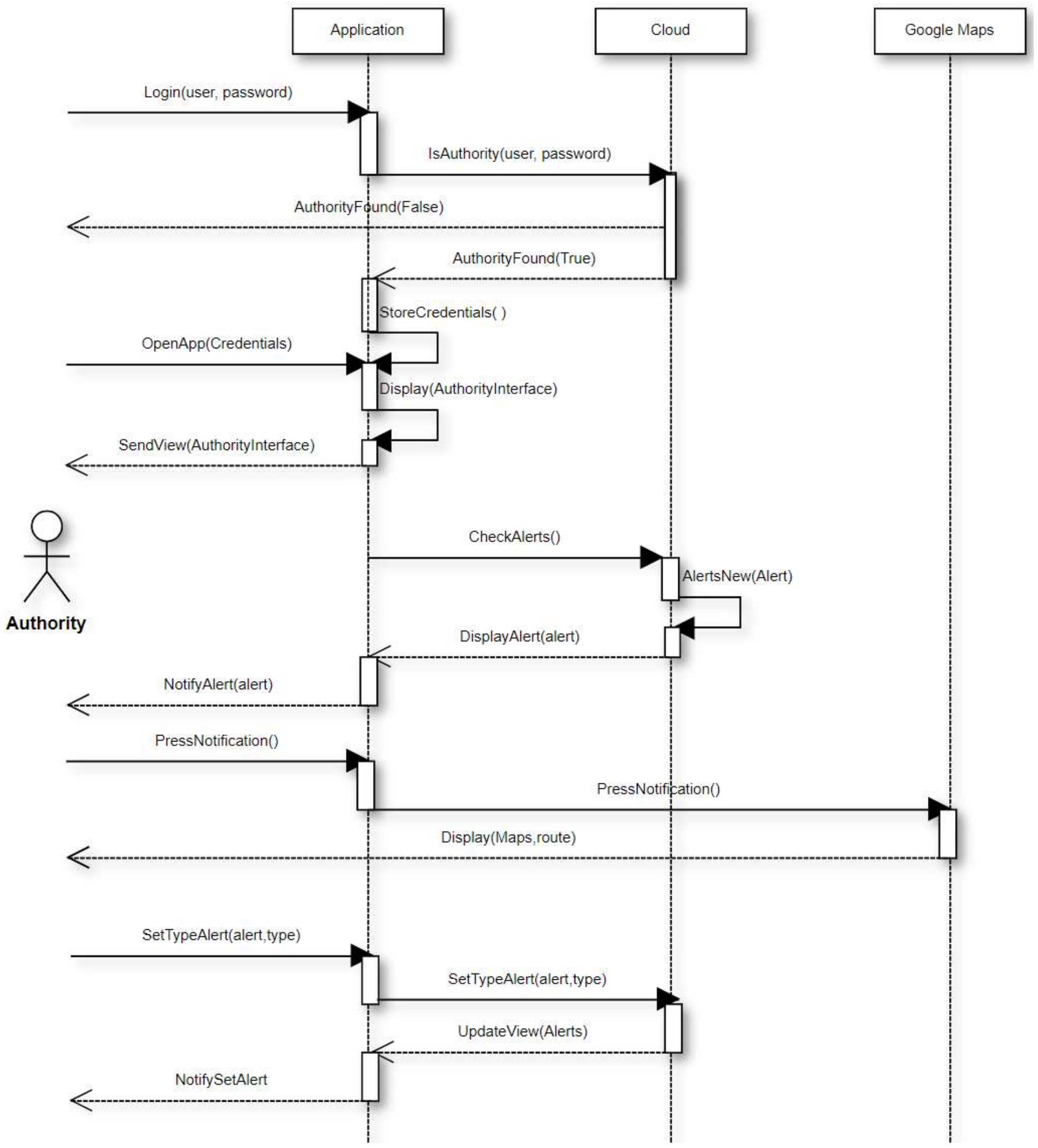


Figure 55: Authority diagram

When the application starts up, it does a prior credential check before loading the view. If no credentials have been input to the application, the user must log in to the system. Once logged in, the application will send this information to the cloud in order to authenticate the user and to determine if it is a student or an emergency assistant. If the user is not in the database, an error will be sent back and they he will have to undergo the authentication process once again with the right credentials. Once the later are verified they are stored in the application using SharedPreferences, and sending the user the appropriate view (in this case, the authority interface).

Once the application has its credentials the user can do two things:

- **Press Notification**: In this case the application will collect the student location and ask google maps to display the fastest route.

- **Set Types:** When this option is pressed the application will send a notification to the cloud to insert the type of the specific emergency. The cloud will set the type and modify the view for the authority.

5.1.1.3 Backend Design

The backend has been designed in IBM Bluemix using NodeJs and Node-red. The service used is CloudantDB, which is attached to the backend via nodered outputs and intermediate queries. As we can see in the figure below, the backend its an API to access different functions and is accessed through the Internet via an HTTP GET request.

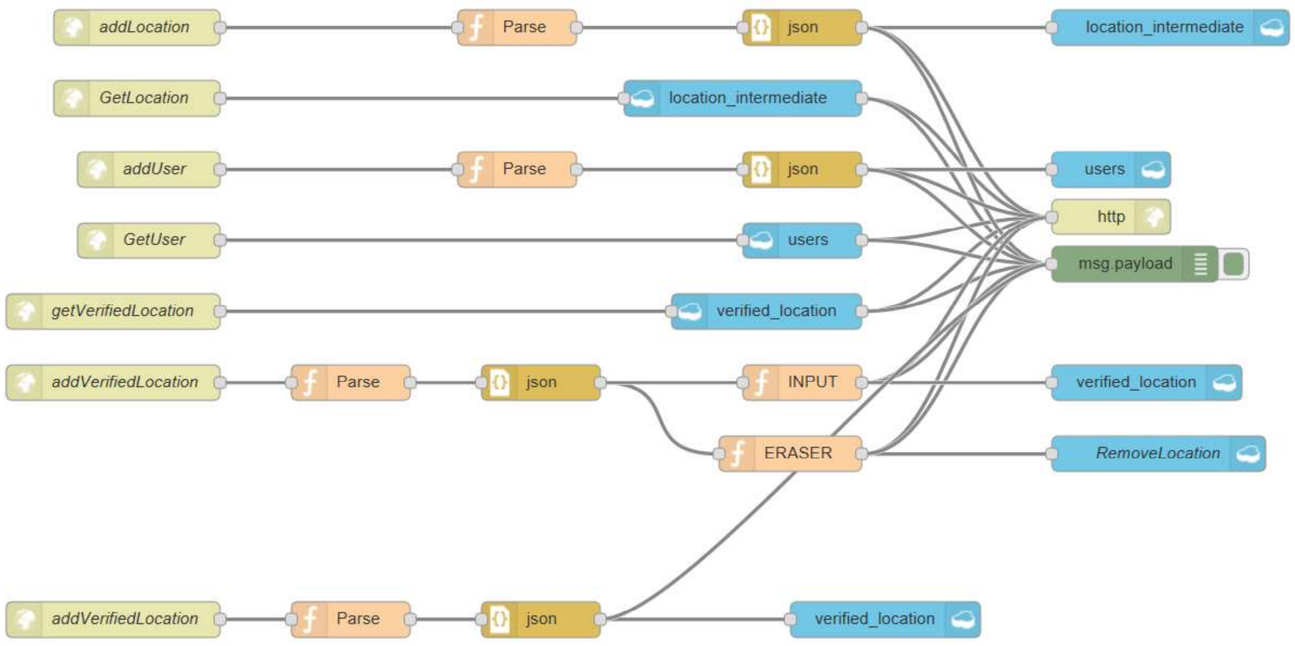


Figure 56: Backend Design in IBM Bluemix

We have three tables in the database:

- **Users:** Here we find the credentials of all of the application's users in IIT and public safety. The information is retrieved from the backend and compared to the users' provided login information. If they are a match, their credentials are then stored in shared preferences. Passwords are stored using hashing so that they are not compromised if DB is.

- **Location_intermediate:** This table stores the data produced when the panic button is pressed. This information is then processed by public safety officials who will input the type of event occurred at the location where the button was pressed.

- **Verified_Location:** Once the event type is added, the backend automatically erases the event from location_intermediate and stores the information on verified_location. This table is used to show the pins on the map that document past events.

5.1.2 IMPLEMENTATION OF CAMPUS APPLICATION

The figure below displays, the login page of the application. The information required is the students/authorities university email and a password. As mentioned before, this login page will appear only if you haven't logged in before and, it's the same for both students and authorities.

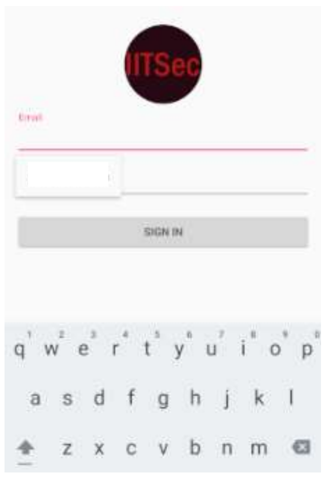


Figure 57: Campus Application Login

5.1.2.1 Implementation of Student Interface

Once the student has logged in, the figure below shows the screen they will encounter, as we can see there is panel with two tabs: panic button and map view.

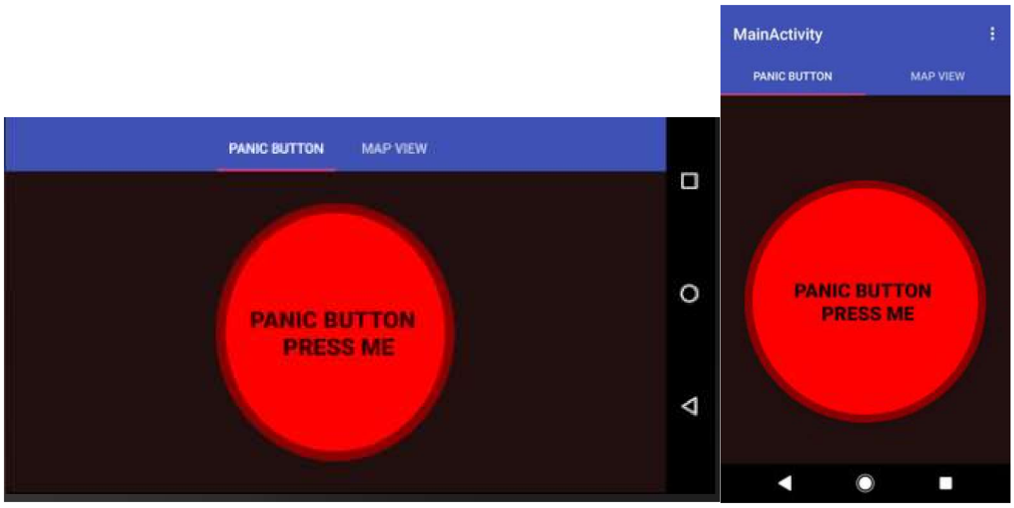


Figure 58: Student main interface

When the panic button is pressed, a pop-up window with a countdown appears. This allows the student a short time frame to cancel the emergency call if the button was pressed by mistake. While this pop-up is active, the GPS service is obtaining the student's location.

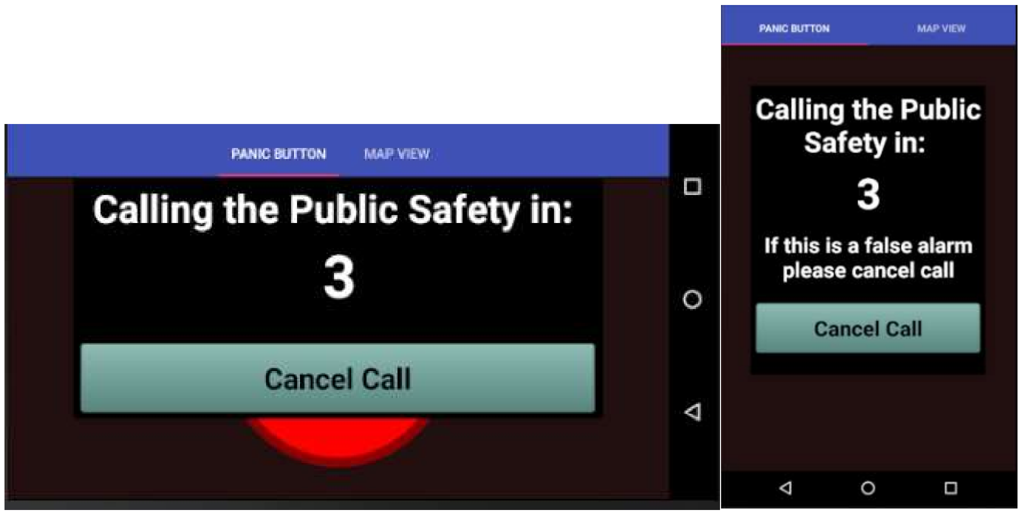

PANIC BUTTON
MAP VIEW
Calling the Public Safety in:
3
If this is a false alarm please cancel call
Cancel Call

Figure 59: Pop-up when student has pressed the button

Once the timer has expired, the location is sent to the IBM Bluemix Platform so that the authorities are notified, and the application reverts to showing the main screen again.

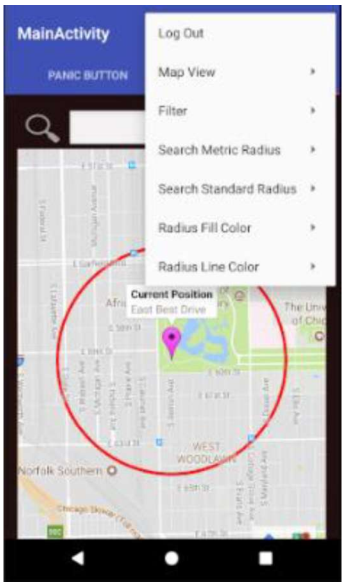


Figure 60: Student Settings

If, on the other hand, the student wants to view de map, there are some options they can select:

- **Filter:** The student can filter certain types of emergencies.
- **Search Metric radius:** Miles or kilometers
- **Search Standard Radius:** The student can select the length of the radius in which the selected event types is displayed.
- **Radius Fill Color:** The student can select the color of the circle.
- **Radius Line Color:** The student can select the color of the radius.

The map shows all of the past events contained within the selected radius. Another interesting feature is that the student can plan a trip from point A to point B, and the map them any emergencies that happened in route.

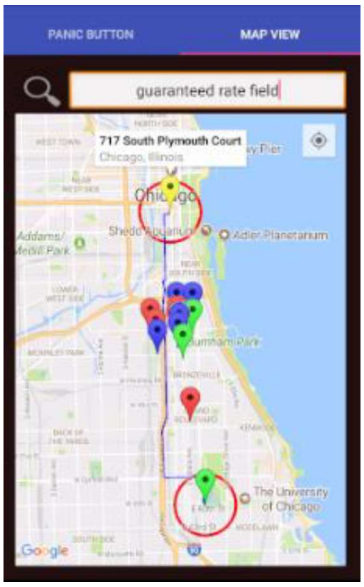


Figure 61: Student route

Moreover, if one of these pins is selected, the information related to that particular event is displayed. In the image below we can see an example of the events that had happened inside the radius where the student is located.

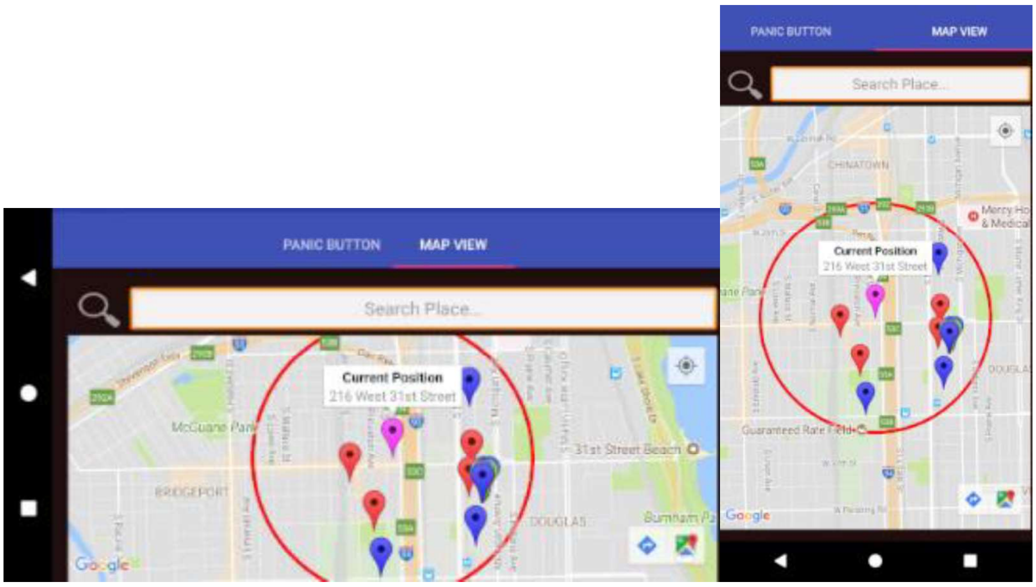
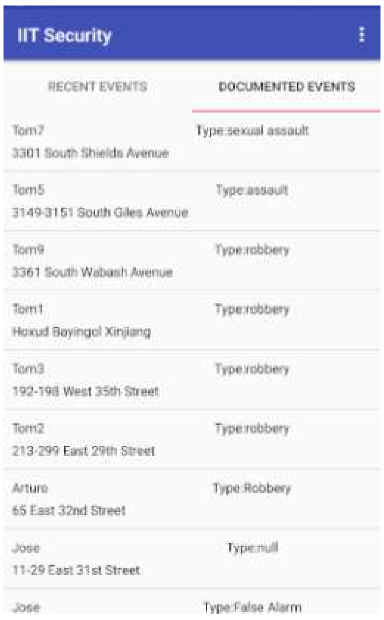


Figure 62: Student radius to see events

5.1.2.2 Implementation of Authority Interface

If the ones logged in are the authorities, the main screen will show any recent events, signaling that someone is in need of immediate assistance. Normally this list will be empty but will be updated with any new notifications as soon as someone needs assistance. Furthermore, the authorities can see past documented events. Each event is linked to a particular set of credentials belonging to the person who set off the alarm, making them easily identifiable and traceable, as well as enabling the identification of people who may be misusing the app by creating fake alerts.



Tom7 3301 South Shields Avenue	Type:sexual assault
Tom5 3149-3151 South Giles Avenue	Type:assault
Tom9 3361 South Wabash Avenue	Type:robbery
Tom1 Hoxud Bayingol Xinjiang	Type:robbery
Tom3 192-198 West 35th Street	Type:robbery
Tom2 213-299 East 29th Street	Type:robbery
Arturo 65 East 32nd Street	Type:Robbery
Jose 11-29 East 31st Street	Type:null
Jose	Type:False Alarm

Figure 63: Authority main screen

If the authorities receive a notification and press the emergency event on the list, Google Maps will calculate the fastest route to the location. The figure below is an example of what the authorities will see when attending an emergency.

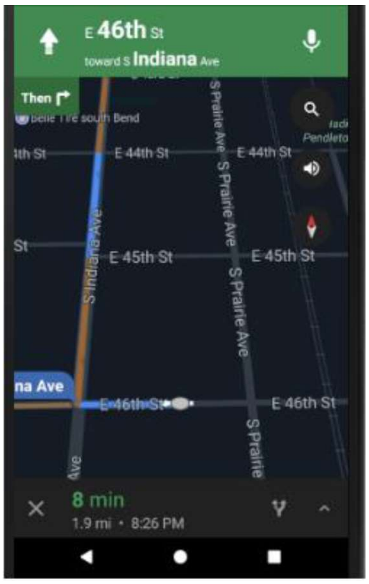


Figure 64: Google map route to the emergency

Once the event has been resolved, the authorities have to clarify the type of event taken place so that it disappears from the list of recent, unattended events and moves under documented events. This enables them to keep track of those events that have been taken care of.

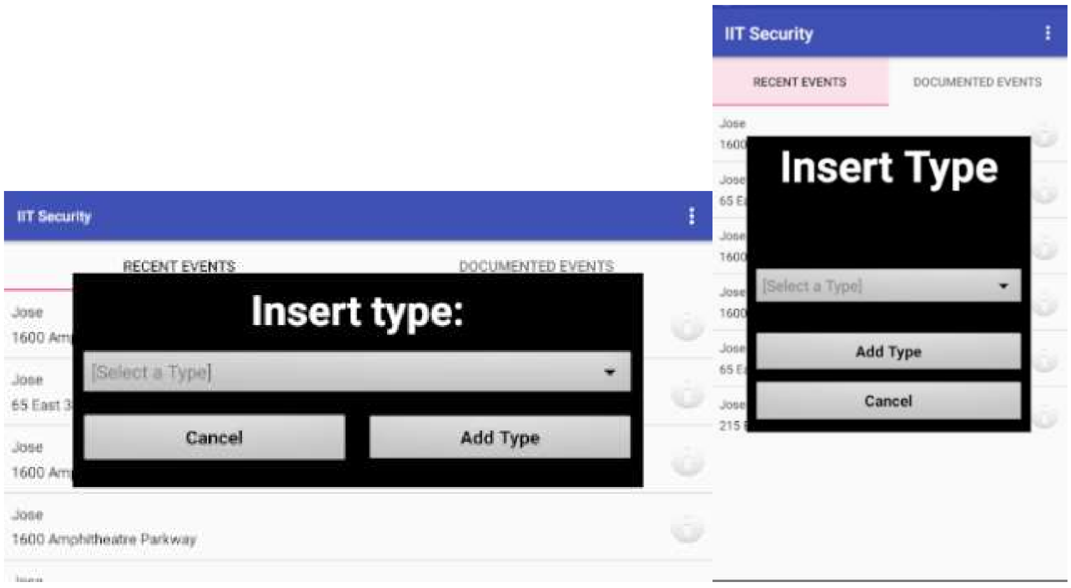


Figure 65: Authority: Insert Type

There are four types of emergency set, including false alarms. If the authorities report the event as a false alarm, it will be registered but it will not appear on the map.

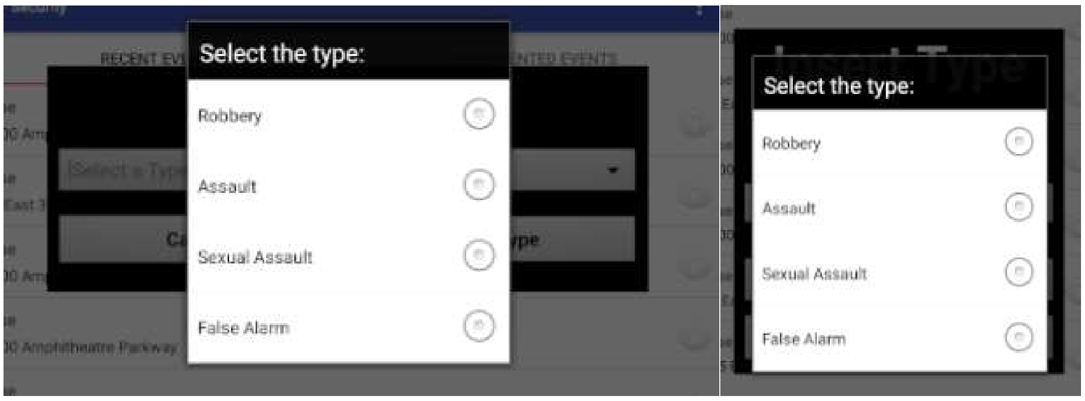


Figure 66: Type of assaults

The application has been developed to be responsive in all Android devices, both in landscape and portrait views.

5.1.3 ANALYSIS OF RESULTS

The application has a purposefully calculated latency of five seconds so that the user can cancel the alert if they have pressed the button by error. The authorities list of events refreshes itself once every five seconds, so the latency on the authorities' side is also, at most, five seconds.

5.2 MOBILE APPLICATION FOR EMERGENCY SERVICES

The mobile application for the emergency services was developed to handle SIP calls with the emergency infrastructure that proves the user's the indoor location. Its requirements include:

- The application must be finished by the end of July.
- There must be a countdown so that the calls can be cancelled before actually calling.
- The countdown must be sufficient for the smartphone to collect information from the beacons.
- The priority is the call. If no information has been received from the BOSSA Platform, the call must be made.
- Settings must be configured once.

Figure 67: NG 9-1-1 Logo

5.2.1 DESIGN OF INDOOR LOCATION APPLICATION

The architecture proposed for the application is the following.

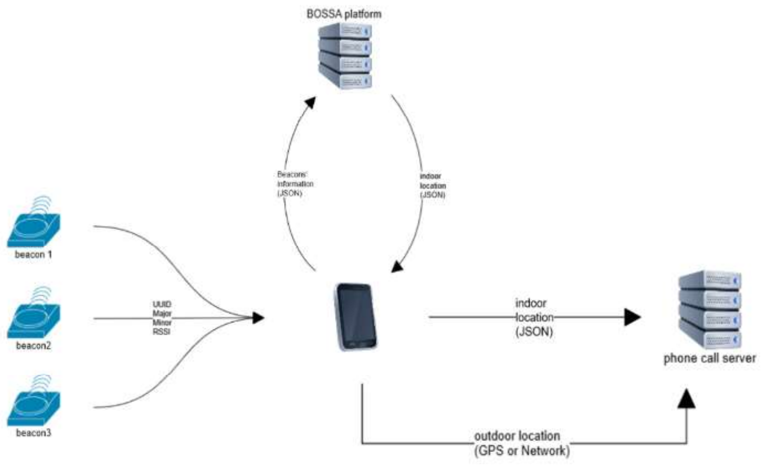


Figure 68: Architecture of the Indoor Location Application

5.2.1.1.1 Use Case and Activity Diagram

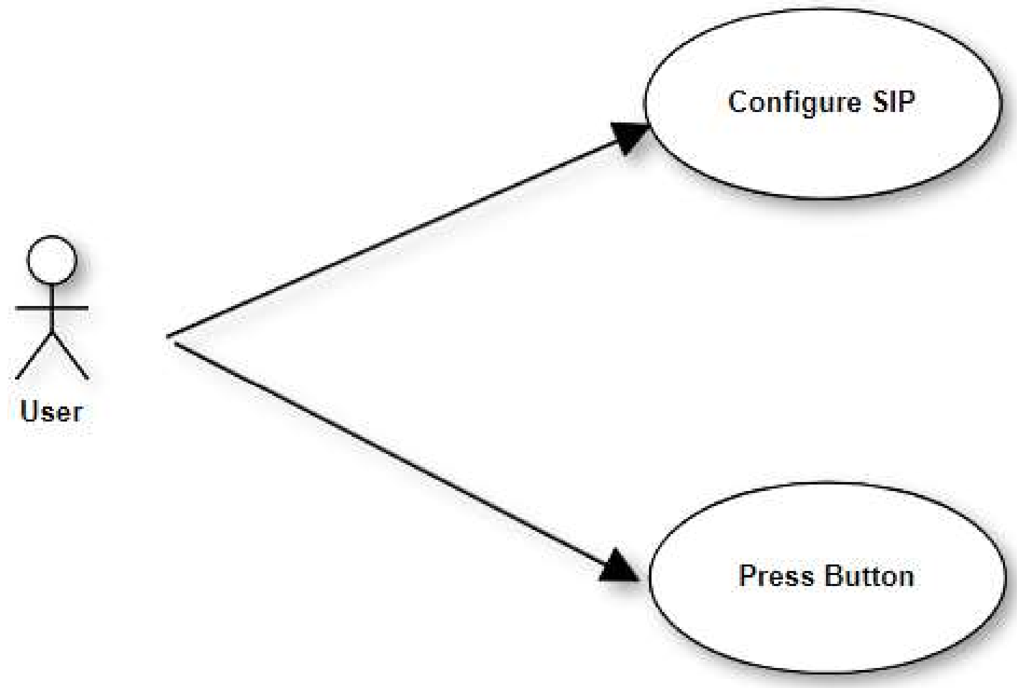


Figure 69: Indoor location Use Case

- **Configure SIP:** The user must configure SIP to register to the ESINet. The registration involves username, password, the SBC IP, domain and the type of communication accepted, such as 3G, Wi-Fi, 4G.

- **Press Button:** Once registered, the user can press the button to start scanning for beacons and request the location so that the SIP call embeds the location inside the SIP INVITE.

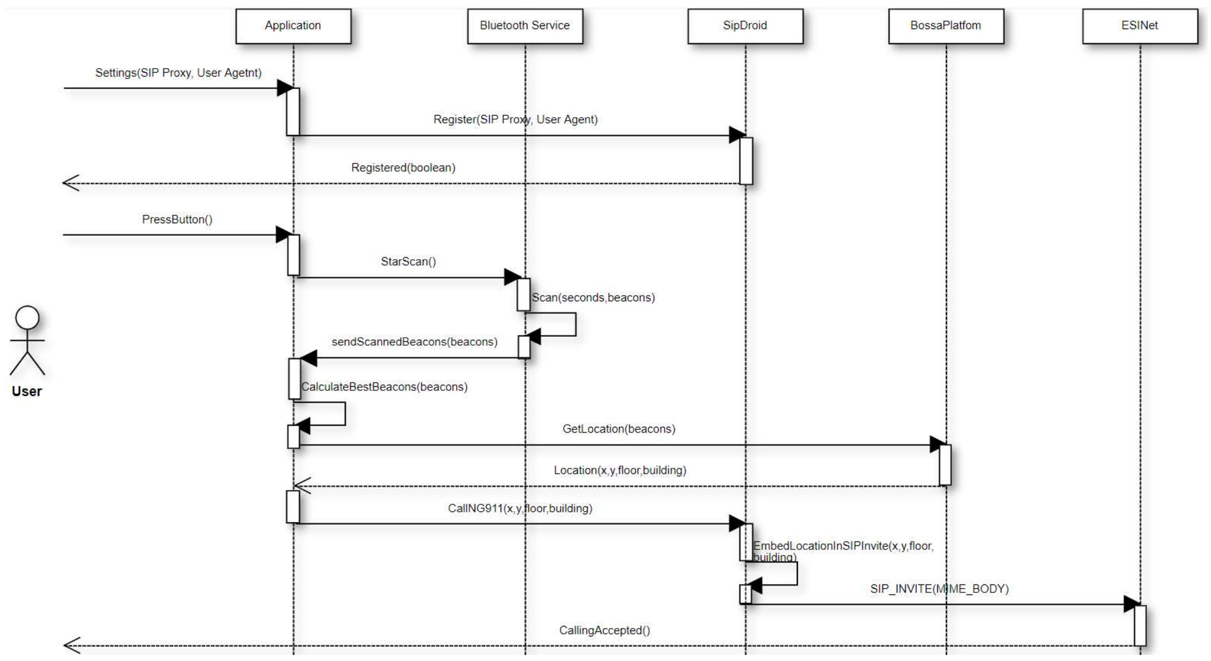

Application
Bluetooth Service
SipDroid
BossaPlatfom
ESINet
Settings(SIP Proxy, User Agetnt)
Register(SIP Proxy, User Agent)
Registered(boolean)
PressButton()
StarScan()
Scan(seconds,beacons)
sendScannedBeacons(beacons)
CalculateBestBeacons(beacons)
GetLocation(beacons)
Location(x,y,floor,building)
CallNG911(x,y,floor,building)
EmbedLocationInSIPInvite(x,y,floor, building)
SIP_INVITE(MIME_BODY)
CallingAccepted()
User

Figure 70: Indoor Location Activity Diagram

The activity diagram shows the sequence of events followed by the emergency application. Upon pressing the panic button, the application requests the Bluetooth service to start scanning for a period of fifteen seconds. After that, the application will select the beacons to send the information collected to the BOSSA Platform. Many of the results involve the same beacons, si an average is calculated to stop the application from sending redundant information. The BOSSA Platform returns the user's location in a JSON Format.

https://example.com ?json[]={“major”:1000,“minor”:540,“rssi”:-91.0}&json[]={“major”:1000,“minor”:518,“rssi”:-92.0}&algorithim=1

Figure 71: example of HTPP call requesting location

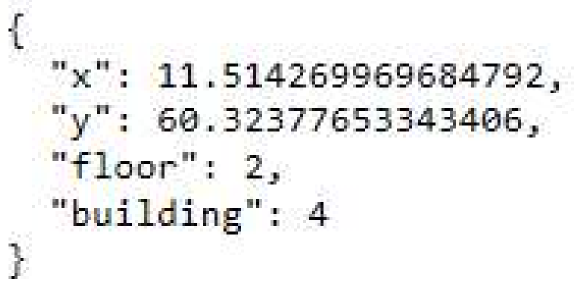

```
{
  "x": 11.514269969684792,
  "y": 60.32377653343406,
  "floor": 2,
  "building": 4
}
```

Figure 72: JSON Response from BOSSA Platform

The application prepares the xml body to insert it into the SIP INVITE and makes the call through SipDroid. When the call is accepted, communication is established between the user and the agent.

SipDroid has been modified so that the invite carries the indoor location of the user, and so that the call is made automatically to emergency services. The modifications made are the following:

- **org.sipdroid.sipua.ui.Sipdroid.java:** This class generates the main activity of the Sipdroid SIP client. It has been modified to be the countdown screen. It was edited for executing a call to 767 automatically after fifteen seconds. It was also modified to do the beacons scanning.

- **org.zoolu.sip.call.ExtendedCall.java:** The method callNG911 was created to handle NG9-1-1 calls. The method is the same as the call method inside this call but it redirects the flow to a new method in the InviteDialog called inviteNG911

- **org.zoolu.sip.dialog.InviteDialog.java:** The method inviteNG911 was created in this class to handle NG9-1-1 calls. It collects the location from the global environment as an xml and it embeds this message in the INVITE.

5.2.2 IMPLEMENTATION OF THE INDOOR LOCATION APPLICATION

When the user opens the application, the red button to call NG911 is displayed. However, it won't call NG911 unless the user provides the correct information for the SIP User Agent.

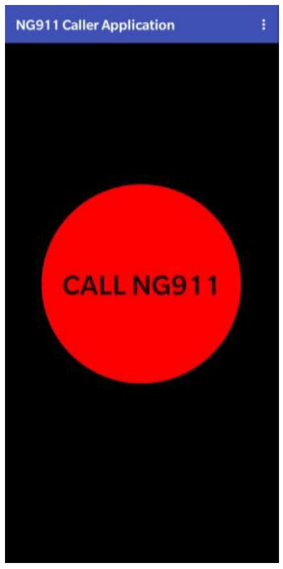


Figure 73: Indoor Location Application: Main Screen

The options tab containes the settings for SIP and the exit button. If the user presses the exit button, the user agent will unregister from the ESINet. If the user presses settings, a form which will be explained in detail in the following paragraph will be displayed.

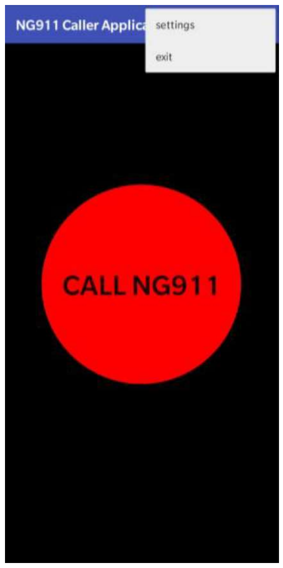
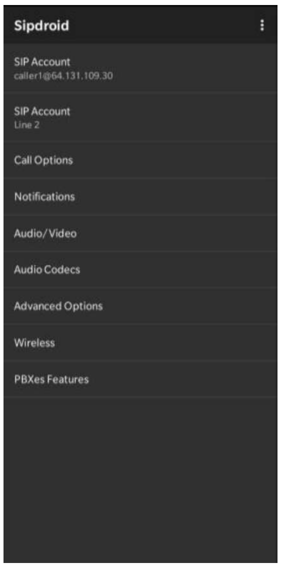


Figure 74: Indoor Location Application: Settings

The Sipdroid project provides a form to create two user agents and has several options such as Call Options, Notifications, Audio/Video, AudioCodecs, Advanced Options, Wireless and PBx features. For this project, we only need a single user agent.



Sipdroid

SIP Account
caller1@64.131.109.30

SIP Account
Line 2

Call Options

Notifications

Audio/Video

Audio Codecs

Advanced Options

Wireless

PBXes Features

Figure 75: SIP Configuration

Inside the user agent settings, you can find:

- **Authorization Username:** This is the username that we will register with to the ESINet. The username is place before the @ such as: **username**@proxyIP.

- **Password:** This is the password for the username.

- **Server or Proxy:** This is the server that we are going to register and send the invites through. The server or proxy usually comes as an IP address and it is placed after the @, such as: username@**192.169.2.1.**

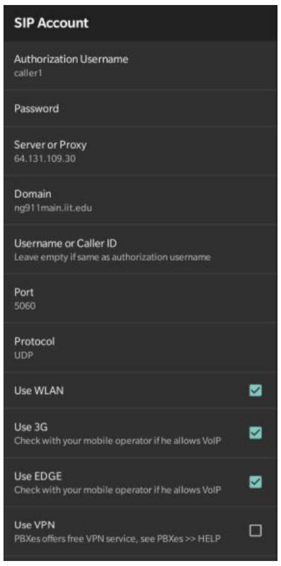
- **Domain:** This can be left blank if the domain is the same of the server or proxy.

- **Username or Caller ID:** This can be left blank if the username is the same as the authorization username.

- **Port:** For SIP call we will use 5060.

- **Protocol:** The protocol used is UDP.

- **Communication Services:** It is a section where we can define how the call will be established, though WLAN, 3G, EDGE and/or VPN. If the user is in a place where no WLAN is detected and they have only enabled the Wi-Fi option, then the call cannot go through. By default we will have all of this enabled except for the VPN.



SIP Account

Authorization Username
caller1

Password

Server or Proxy
64.131.109.30

Domain
ng911main.iit.edu

Username or Caller ID
Leave empty if same as authorization username

Port
5060

Protocol
UDP

Use WLAN ✓

Use 3G ✓
Check with your mobile operator if he allows VoIP

Use EDGE ✓
Check with your mobile operator if he allows VoIP

Use VPN ☐
PBXes offers free VPN service, see PBXes >> HELP

Figure 76: SIP Account Settings, User Agent

Once the SIP settings have been configured, a notification will appear on the upper part of the smartphone screen. If this notification is green, it means the user agent has been created succesfully and the user will able to succesfully make the call if the button is pressed.

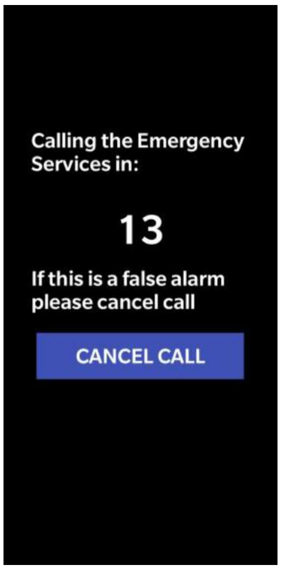

Calling the Emergency Services in:
13
If this is a false alarm please cancel call
CANCEL CALL

Figure 77: Indoor Location Application: collecting data

If the button is pressed, the smartphone will start a countdown and will collect the beacons' information. If there are no beacons around, the smartphone will insert the default location (everything to zero) once the countdown has expired.

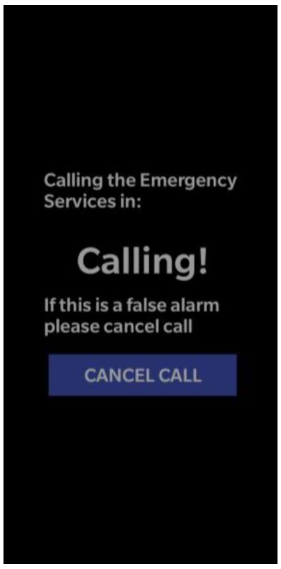

Calling the Emergency Services in:
Calling!
If this is a false alarm please cancel call
CANCEL CALL

Figure 78: Indoor Location Application: Ending scanning

While the countdown is happening the user can cancel the process by pressing "cancel call". Otherwisse, the call will go through to the SBC, displaying the call screen shown below.

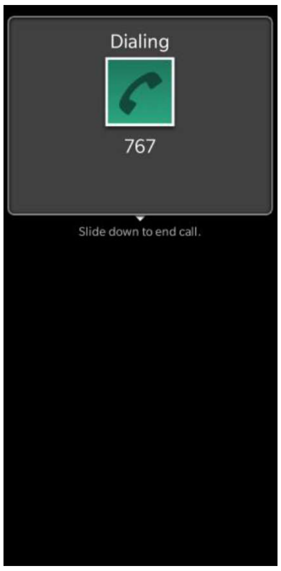


Figure 79: Indoor Location Application: Calling Emergency services

Once the ESINet has received the call and redirected it to the correct PSAP, the call is transferred to an operator who will be able to locate the user's the indoor location.

5.2.3 ANALYSIS OF RESULTS

The application has managed to reduce the latency of the adding indoor location by 50%, as compared to the previous application developed in this laboratory. We have reduced it to just a button, whereas before it was a dial-pad and the user had to call the emergency services by inserting the appropriate number. A user is now able to put a call through that is compatible with a real world Next Generation (NG) 911 Network module.

5.3 MOBILE APPLICATION FOR INSERTING RAW DATA

The mobile application to insert raw data in the database has been developed using the cross-platform approach. It either runs on iOS or on an Android Phone. It is important to feed the database with raw data. This is crucial in the testing and improving of algorithms. When t comes to indoor location, it is necessary for the algorithms to be accurate, so that the emergency services can reach the scene on time. This application uses IBM Bluemix to store the data and, using an URL, a csv can be downloaded.

5.3.1 DESIGN OF TEST APPLICATION

5.3.1.1.1 Student Use Case and Activity Diagram

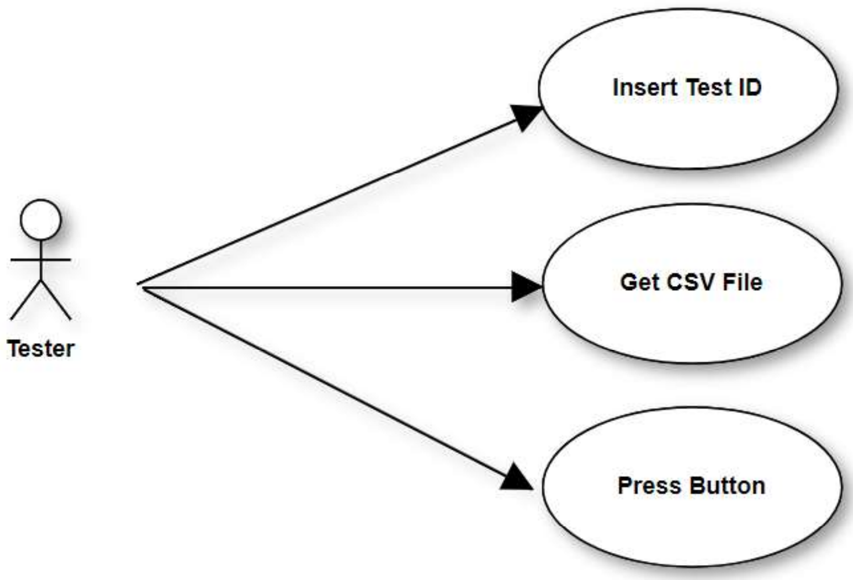


Figure 80: Tester Use Case

- **Insert Test ID:** For data collection, the tester must insert the test ID they are going to collect the data from.

- **Get CSV File:** The user must be able to download a csv with all the data tested.

- **Press Button:** The user must press the button to collect data and send it to the platform.

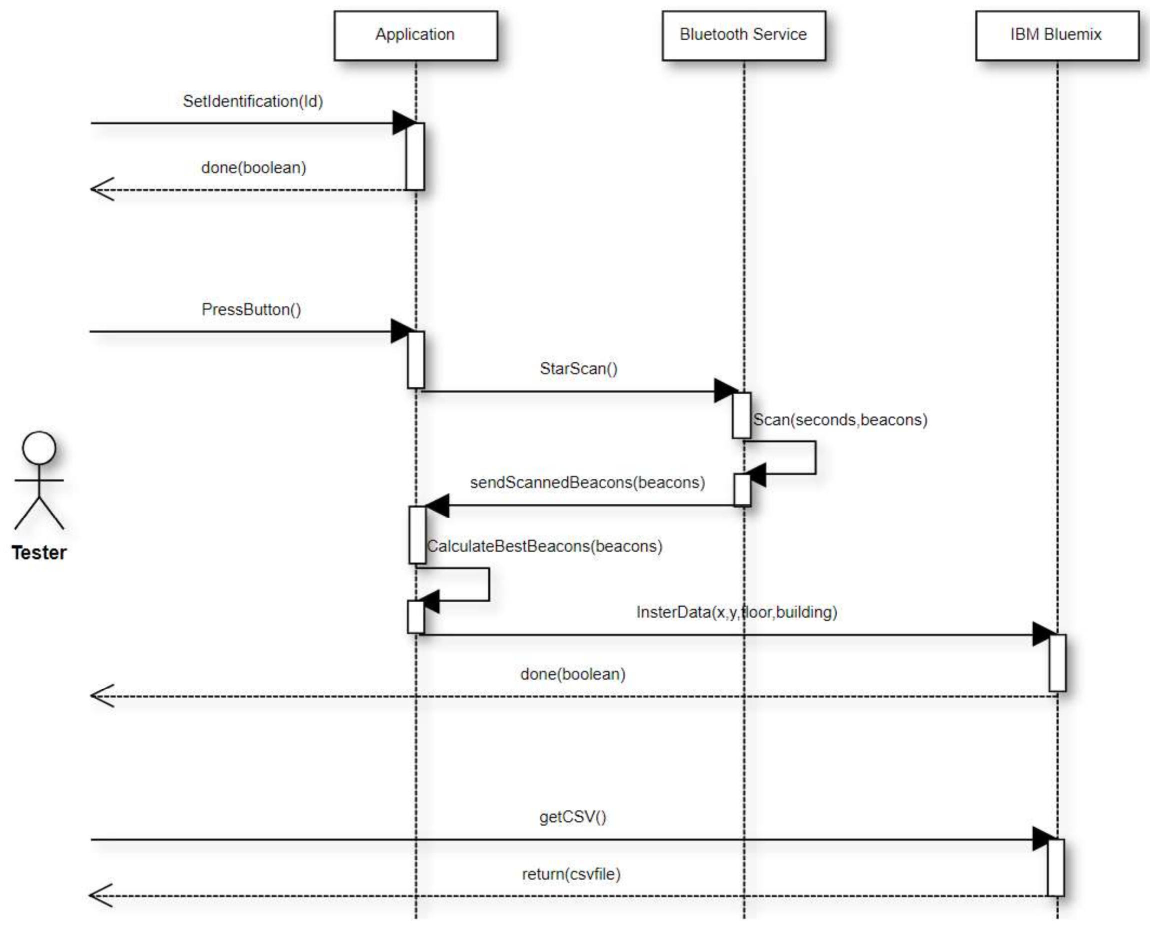
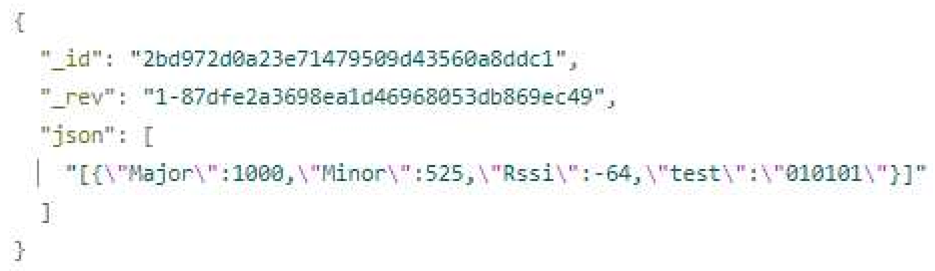


Figure 81: Tester Activity Diagram

The tester needs to select the identification number of the test they are going to collect the data from. The identification consists of six digits. The first two determine the building, the next two determine the floor, and the final two determine the place of the test. If the tester does not set the identification, the button is deactivated.

Once the test ID is selected, the tester can press the button and the application the application will begin scanning for beacons and finding the minor, major, rssi. It will then perform the calculation to choose the best beacons and will through an HTTP request to the API for each beacon found. The cloud will insert the data into the database in JSON Format.



```
{
  "_id": "2bd972d0a23e71479509d43560a8ddc1",
  "_rev": "1-87dfe2a3698ea1d46968053db869ec49",
  "json": [
    "[{\"Major\":1000,\"Minor\":525,\"Rssi\":-64,\"test\":\"010101\"}]"
  ]
}
```

Figure 82: Example of data in the Cloudant database

The tester can ask for the csv and the cloud platform will download one containing all the data automatically. At this point, the people in charge of the algorithms can start doing their work on improving accuracy.

5.3.1.2 *Backend Design*

The backend has been designed in IBM Bluemix using NodeJs and Node-red. The service used is CloudantDB, which is attached to the backend via node-red outputs and intermediate queries. As we can see in the figure below, the backend its an API to access different functions and is accessed through the Internet via an HTTP GET request.

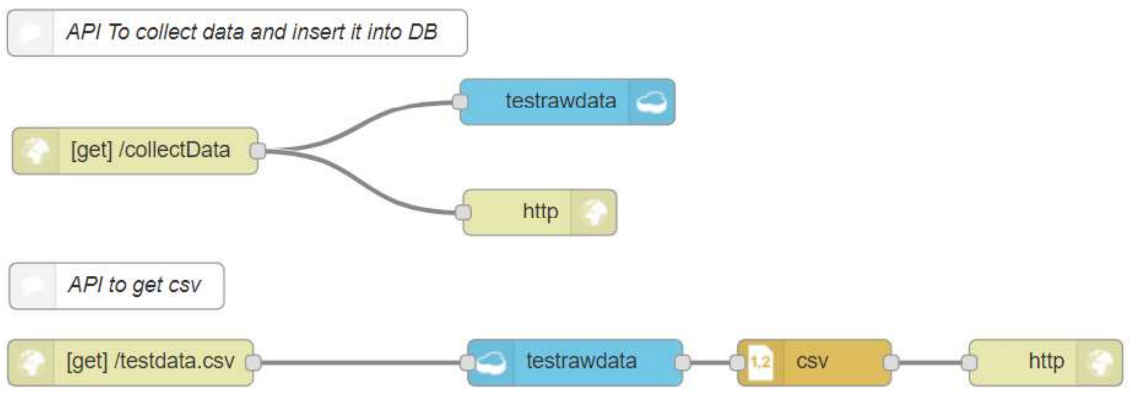


Figure 83: test Application: Backend design

The API contains the following:

- **collectData:** It is used to insert data into the database called *"testrawdata"*.

- **testdata.csv:** It is used to download a csv file.

5.3.2 IMPLEMENTATION OF THE TEST APPLICATION

The application has been developed using CSS, HTML and JavaScript. The Evothings Studio platform supports the scan for beacons. The latter send their major and minor inside a hexadecimal number inside the scanRecord. The scanRecord iss then parsed so that the minor and major are displayed.

To start off, when the application is opened we can see that there is no test ID inserted and therefore the button is disabled.

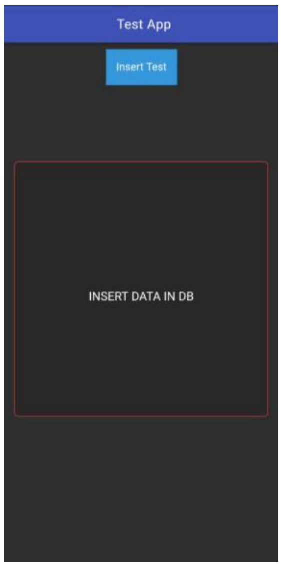


Figure 84: Test Application: Main Screen

When pressing insert test, a dropdown menu is displayed where we can insert the test type. Right now, there are three test positions:

- Test 010101: Building 01, floor 01, test 01.
- Test 030201: Building 03, floor 02, test 01.
- Test 030202: Building 03, floor 02, test 02.

The test ID identifies a unique and singular test place. This is because it is a six digit number, which has been considered sufficient for the system to scale all test types. It covers over a hundred buildings, a hundred floors on each building, and a hundred tests conducted per floor.

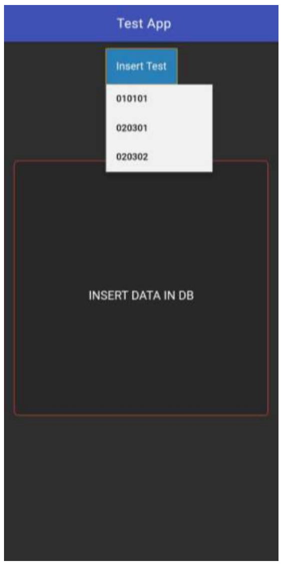


Figure 85: Test Application: Dropdown Menu

Once the test ID has been established, the button is activated and the tester can start testing.

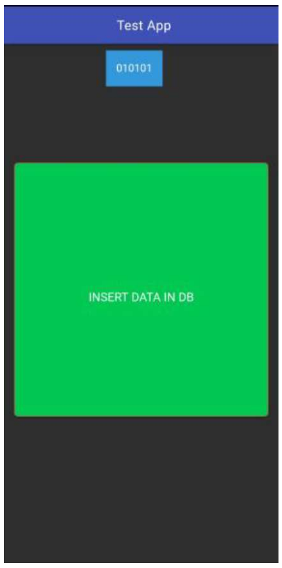
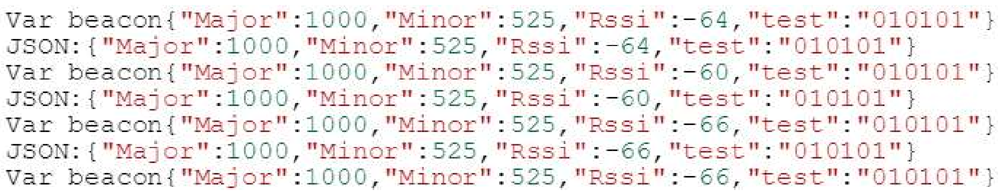


Figure 86: Test Application, prepared to scan

If the button is pressed, the application will start scanning for fifteen seconds and storing the data from the beacons. The figure below shows the application's log when it is collecting the data.



```
Var beacon{"Major":1000,"Minor":525,"Rssi":-64,"test":"010101"}
JSON:{"Major":1000,"Minor":525,"Rssi":-64,"test":"010101"}
Var beacon{"Major":1000,"Minor":525,"Rssi":-60,"test":"010101"}
JSON:{"Major":1000,"Minor":525,"Rssi":-60,"test":"010101"}
Var beacon{"Major":1000,"Minor":525,"Rssi":-66,"test":"010101"}
JSON:{"Major":1000,"Minor":525,"Rssi":-66,"test":"010101"}
Var beacon{"Major":1000,"Minor":525,"Rssi":-66,"test":"010101"}
```

Figure 87: Test Application, console log when Scanning for beacons

While collecting the data, the button is deactivated and the countdown is shown.

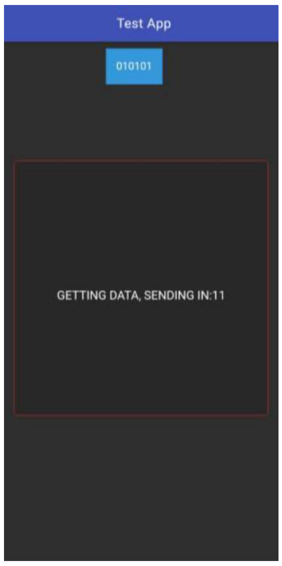
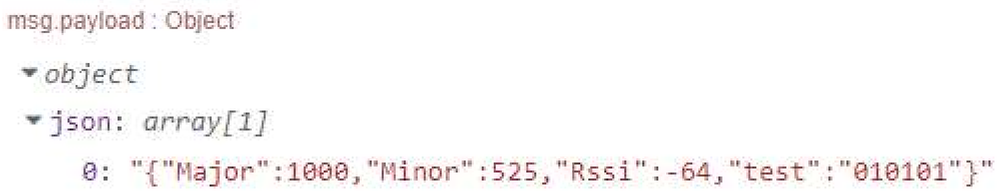


Figure 88: Test Application, screen while collecting data

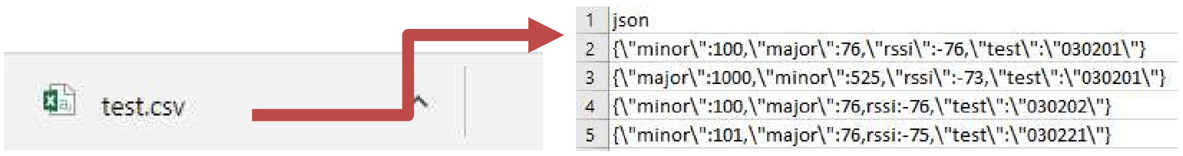
Once the application has finished scanning, it sends the information to IBM Bluemix as shown in the figure below:



```
msg.payload : Object
▾object
▾json: array[1]
    0: "{"Major":1000,"Minor":525,"Rssi":-64,"test":"010101"}"
```

Figure 89: IBM Bluemix data received

The csv can be downloaded onto any platform, so that the tester can download it directly to the computer where they store the algorithms and start working with it right away. For security purposes, the URL cannot be public but, as an example, the URL could be something like https://www.myapplication.net/testdata.csv



```
1 json
2 {\"minor\":100,\"major\":76,\"rssi\":-76,\"test\":\"030201\"}
3 {\"major\":1000,\"minor\":525,\"rssi\":-73,\"test\":\"030201\"}
4 {\"minor\":100,\"major\":76,rssi:-76,\"test\":\"030202\"}
5 {\"minor\":101,\"major\":76,rssi:-75,\"test\":\"030221\"}
```

Figure 90: Data from test downloaded and view

5.3.3 ANALYSIS OF RESULTS

The application takes 15 seconds to collect data. The beacons transmit twice per second, so the application will get 30 measures per test, which reduces anomalous readings, especially when only the average of the signal received is taken into account. The application also allows for the improvement the algorithms' accuracy by using the BOSSA Platform.

5.4 CROSS-PLATFORM MOBILE APPLICATION WITH SIGNALWIRE CLOUD API

We incorporated the cross-platform approach, with the Signal Wire Cloud API. This approach serves the same purpose as the emergency call application, except the call to the emergency services is placed automatically and an SMS message is sent to the user's family. With SignalWire we can buy a telephone number and recreate a call through their platform. This approach does not use SIP Call, but the normal PSTN call.

5.4.1 DESIGN OF CROSS PLATFORM APPLICATION

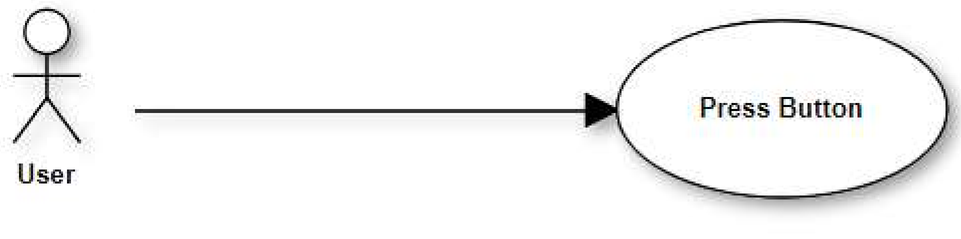


Figure 91: Cross-Platform Indoor Location Use Case

- **Press Button:** The user can press the button for assistance.

In this case the user does not need prior configuration so reaching the emergency services is quicker.

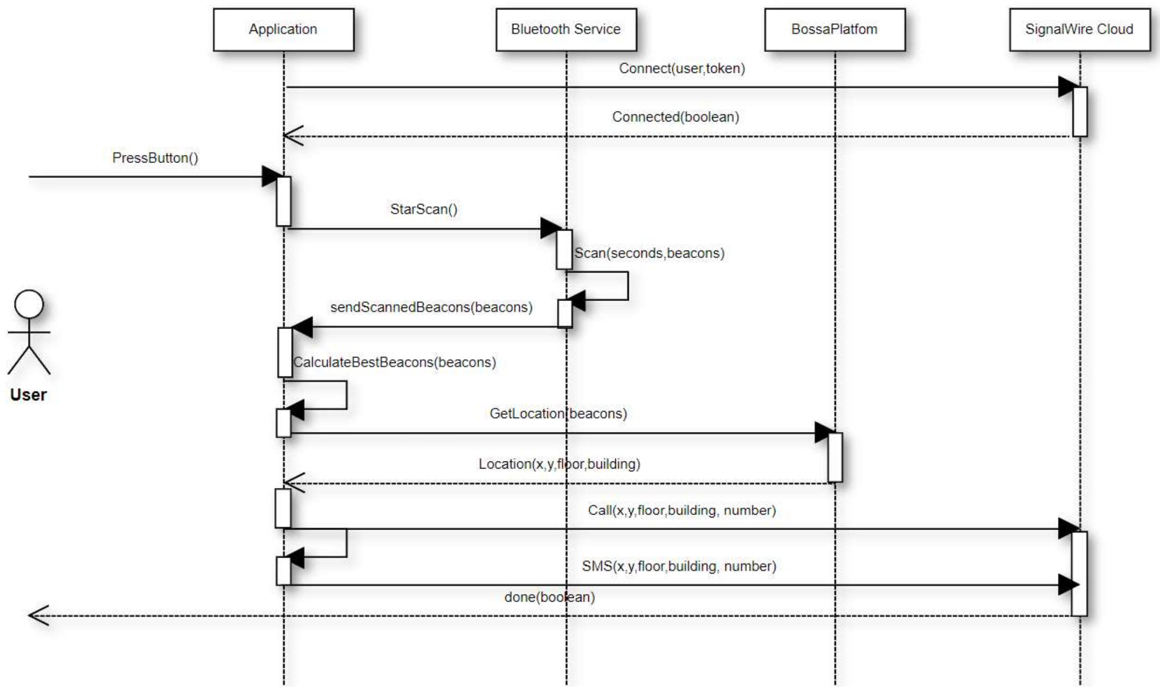


Figure 92: Cross-Platform Indoor Location, Activity Diagram

When the application is launched it will automatically connect to the SignalWire cloud platform. If the user pressed the button, the Bluetooth service starts scanning for fifteen seconds, after which the application will select which beacons' information to send to the BOSSA Platform, and will once again calculate an everage for the information received, thus avoiding any redundancies. The BOSSA Platform returns the location of the user in a JSON Format to the application. The application will then use the SignalWire API to create the automatic call and the SMS message.

5.4.2 IMPLEMENTATION OF THE CROSS-PLATFORM APPLICATION

The cross-platorm application has the same interface as the test application, except for the dropdown menu. Its functionality is different as well, as it places a call and sends an SMS.

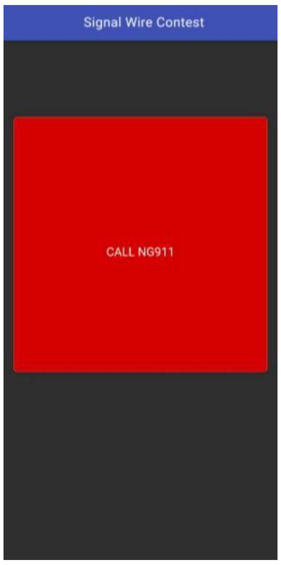
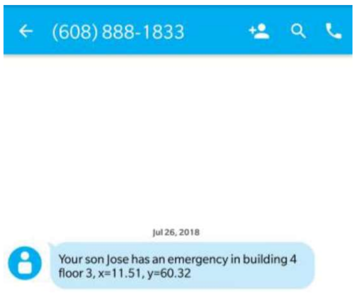


Figure 93: Cross-Platform Indoor Location, main screen

Once the button is pressed and the pertinent data collected, an SMS message is sent to a family member (see figure below).



Your son Jose has an emergency in building 4 floor 3, x=11.51, y=60.32

Figure 94: Cross-Platform Indoor Location SMS

5.4.3 ANALYSIS OF RESULTS

The application gives the opportunity to everyone who owns a smartphone (since it has been done as a cross-platform application) to make an automated call to public safety officials and provide them with their exact indoor location. The call can simply be routed through the usual 911 number so that no additional infrastructure is required. Furthermore, the SMS messaging service will let the user's family members know about their emergency.

Chapter 6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

All of the objectives were covered after the development and implementation the four applications.

The application designed for campus security offers a closed environment for all Android users to report an emergency, regardless of where they are without having to manually specify their location or detail the problem they are faced with. Event description can be introduced later, but the priority is getting public safety officials on the scene as soon as possible. The application's response time is 5 seconds, giving the student enough time to cancel the emergency call if the button is pressed by mistake. This solution is an efficient way to tackle this problem of campus security (at IIT?) and overcome the main setbacks encountered by solutions previously implemented at the university.

The application for the SIP call to the NG911 infrastructure has covered the future work introduced by the research paper explained in the status of the work. It also takes half as much time as the previous application was developed for this purpose and uses the iBeacon technology solution. Nevertheless, the project wants to reach all of the smartphone users and, due to the lack of portability among the different mobile platforms this application was supposed to be working in a cross-platform environment. Granted, this new application is left for future work, but this project can be used as a reference, especially as it uses the cross-platform environment for the other two applications left to explain. The main reasons for not achieving the cross-platform approach was that Evothings Studio is prepared for IoT solutions and SIP goes out of the scope.

Inserting raw data into the system is a critical application for the system because of its importance when improving algorithm accuracy. Both the infrastructure and the application would be worthless if the algorithms didn't provide an accurate location, including the correct floor as well as the exact coordinates. The application therefore allows us to fine tune the algorithms' parameters to improve their accuracy, making the system more reliable.

6.2 FUTURE WORK

There are five main things to be done after the conclusion of this project.

The campus security application can improve its features by using the timestamp of the alerts to showcase the latest mishap on the map. Things might change over the time but, for now, we are getting all the history records.

The second is to feed the database with raw data and work on the algorithms to improve their accuracy.

To develop the NG911 application in either iOS or follow the native approach. The main problem with the native approach is that every change has to be made twice, whereas, developing the application as a cross-platform application, the developers will only have to change one application. For this purpose, the application is very advanced, and it would only require adding the SIP library inside the Apache Cordova project and embed the indoor location inside the INVITE SIP message.

To migrate the raw database into the BOSSA Platform. The current database resides in IBM Bluemix and needs to be moved to be part of the system. It's a matter of changing the URL to point to the BOSSA API when inserting raw data.

Lastly, future work must be done in maintaining the application to ensure that all the beacons work perfectly. Nowadays, the beacons installed inside buildings aren't monitored. An application ought to be developed to monitor and handle said beacons in real-time, for instance, erasing those without battery and replacing them with new beacons with the same location information. This could be done by just replacing the minor of the beacon inside the BOSSA Platform. This might not sound like a priority right now because there are just three buildings involved but, if the system starts to upscale, it will be of the utmost importance to keep all of its devices under strict control.

Chapter 7. BIBLIOGRAPHY

[1] "Chicago, IL Crime Rates", Neighborhood Scout, Copyright © 2000-2018 Location Inc®, https://www.neighborhoodscout.com/il/chicago/crime

[2] "Chicago Data Portal", © 2018 City of Chicago, https://data.cityofchicago.org/Public-Safety/Crimes-Map/dfnk-7re6/data

[3] "SIP Protocol", July 14 2018 https://es.wikipedia.org/wiki/Protocolo_de_iniciaci%C3%B3n_de_sesi%C3%B3n

[4] SIP: Session Initiation Protocol, RFC 3261 Section 1, Copyright (C) The Internet Society (2002). https://tools.ietf.org/html/rfc3261

[5] ITM 446/546 Voice Over IP, Session Initiation Protocol Slides, Carol Davids © 2017

[6] "What are the SIP Methods/ Requests and Responses", 3CX, https://www.3cx.com/pbx/sip-methods/

[7] "HTTP Protocol", July 18 2018, https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto

[8] "HTTP request Methods", Copyright 1999-2018 by Refsnes Data, https://www.w3schools.com/tags/ref_httpmethods.asp

[9] "HTTP Basics", Slides, LinkedIn, https://www.slideshare.net/eoinkeary/01-http-basics-v27-66301583

[10] "Las diferencias entre la Frecuencia Inalámbrica de 2.4GHz y 5GHz". Copyright © TP-LINK Technologies Co., Ltd. 2016. http://www.tp-link.ec/FAQ-499.html

[11] https://armandof.cubava.cu/2015/06/25/compartir-conexion-a-internet-via-wifi/

[12] "Low energy". Bluetooth, © 2016 Bluetooth SIG, Inc. https://www.bluetooth.com/what-is-bluetooth-technology/bluetoothtechnology-basics/low-energy

[13] "GPS: The Global Positioning System", Space-Based Positioning Navigation & Timing, August 3 2018, https://www.gps.gov

[14] "Raspberry Pi 3 Model B". Raspberrypi.org. Raspberry Pi Foundation, UK. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[15] "Raspberry Pi 3 - Model B". Modmypi.com. http://www.modmypi.com/raspberrypi/rpi3-model-b/raspberry-pi-3-model-b

[16] "NENA Detailed Functional and Interface Standards for the NENA i3 Solution" Copyright 2016 National Emergency Number Association, Inc., https://cdn.ymaws.com/www.nena.org/resource/resmgr/standards/NENA-STA-010.2_i3_Architectu.pdf

[17] "SIP server for you", Myvoipapp, ©2007-2018 All Rights Reserved. https://www.myvoipapp.com

[18] "Evothings" https://evothings.com

[19] "iOS Versus Android? Does it Mater", Intertech, Jim White, https://www.intertech.com/Blog/ios-versus-android-matter/

[20] "Sipdroid", i-p-tel, Jason Long. https://www.sipdroid.org

[21] "What is cloud computing?". IBM Cloud, © Copyright IBM Corporation 1994, 2016. https://www.ibm.com/cloud-computing/what-is-cloud-computing

[22] Innes, Brian. "A developer's guide to the Internet of things (IoT)". IBM, USA. Coursera Inc., 2016. https://www.coursera.org/learn/developer-iot

[23] IBM Cloud (formerly IBM Bluemix and IBM SoftLayer, May 2017, https://searchcloudcomputing.techtarget.com/definition/IBM-Bluemix

[24] "Work with cookies", Example, JS Foundation, https://cookbook.nodered.org/http/work-with-cookies

[25] "The Core Nodes", JS Foundation, https://nodered.org/docs/user-guide/nodes

[26] "An Elastic Cloud Communications Platform", © 2018 SignalWire | Palo Alto, CA, https://signalwire.com/

[27] Gina GO products, Copyright © 2018 GINA Software s.r.o., http://ginasoftware.com/gina-go.php

[28] "Crean una app para enviar mensajes de alerta en un solo clic", December 12 2013, © La Vanguardia, https://www.lavanguardia.com/tecnologia/moviles-dispositivos/aplicaciones/20131210/54395353567/button-messanging-app-alertas-un-solo-clic.html

[29] "Public Safety and E911 Location Services", NextNav, © 2016 NextNav LLC." http://www.nextnav.com/services

[30] "Indoor Mapping", PoleStar Tracing The Way, https://www.polestar.eu/products/nao-maps/?gclid=Cj0KCQjw45_bBRD_ARIsAJ6wUXR-lGRbQpDtV1AA9ASwaiM4euBfiOVh1kDeKu1CvE5PszfChWkhi7kaAuXwEALw_wcB

[31] Carol Davids, Javier Moreno, Bartlomiej Dworak, Curz Tovar, Bharat Ramaswamy, Mahak Patil. Dispatchable Indoor Location for Mobile Phones Calling for Emergency Services.

[32] "911's deadly flaw: Lack of Location data", John Kelly and Brendan Keefe, USA today, February 22, 2016, https://www.usatoday.com/story/news/2015/02/22/cellphone-911-lack-location-data/23570499/

[33] "Why a Cellphone Lobby Win on 911 Calls Means 'More People Will Die' , Allan Holmes, The Daily Beast, https://www.thedailybeast.com/why-a-cellphone-lobby-win-on-911-calls-means-more-people-will-die

[34] "Salarios en Mobile Application Software Engineer en the United States", Indeed, https://www.indeed.com/salaries/Mobile-Application-Software-Engineer-Salaries

ANNEX A

Test Application: JavaScript and HTML

JavaScript

```
/ JavaScript code for the BLE Scan example app.
// The code is inside a closure to avoid polluting the global scope.

;(function()
{
// Dictionary of found devices.
var devices = []
var testid= 1;
// Timer that updates the displayed list of devices.
var updateTimer = null
var timerCountdown=15
var signalwire
var paramsMessage
function main()
{
  $(function()
  {
    // When document has loaded we attach FastClick to
    // eliminate the 300 ms delay on click events.
    FastClick.attach(document.body)
    // Event listener for Back button.
    $('.app-back').on('click', function() { history.back() })
  })
  // Event handler called when Cordova plugins have loaded.
  document.addEventListener(
    'deviceready',
    onDeviceReady,
    false)
}

function onDeviceReady()
{

  // Un-gray buttons.
  $('button.app-start-scan')
    .addClass('mdl-button--disabled')
    .removeClass('mdl-color--green-A700')
  $('button.app-stop-scan')
    .removeClass('mdl-button--disabled')
    .addClass('mdl-color--deep-orange-900')
document.getElementById("test").onclick = myFunction;
document.getElementById("010101").onclick = test010101;
document.getElementById("020301").onclick = test020301;
document.getElementById("020302").onclick = test020302;


  // Attach event listeners.
  $('.app-start-scan').on('click', startScan)
  $('.app-stop-scan').on('click', stopScan)
}
function startScan()
{


   $('button.app-start-scan')
    .removeClass('mdl-color--green-A700')
    .addClass('mdl-button--disabled')
  stopScan()
  timeScan = Date.now()
  count=0;

  var devices=[]
  evothings.ble.startScan(

    function(device)
    {
      if (device.rssi <= 0  && device.name =="IIT-IL")
      {
          device.timeStamp = Date.now()
          var byteArray = base64DecToArr(device.scanRecord);
                    var pos = 0;
            while (pos < byteArray.length)
            {
                var length = byteArray[pos++];
                if (length == 0) {
                  break;
                }
                length -= 1;
                var type = byteArray[pos++];
                switch(type){
                  case 0x08:
                  case 0x09: // Local Name..
                    break;
                  case 0xFF:
                    pos+=4;
                    pos+=16;
                    device.major = bigEndianToUint16(byteArray,pos)
                    pos+=2;
                    device.minor = bigEndianToUint16(byteArray,pos);
                    pos+=2;
                    break;
                  default:pos+=length;
                    break;
              }
            }
            var beacon = 
'{"Major":'+device.major+',"Minor":'+device.minor+',"Rssi":'+device.rssi+',"test"
:"'+testid+'"}';
            console.log("Var beacon"+ beacon)
            var beaconj=JSON.parse(beacon)
            console.log("JSON:"+ JSON.stringify(beaconj))
         // Store device in table of found devices.
            devices[count]=JSON.parse(beacon)
            count=count+1;
            var millis = Date.now() - timeScan;
            document.getElementById("Call").innerHTML = "GETTING DATA, SENDING 
IN:"+ (timerCountdown-Math.floor(millis/1000));
             if(Math.floor(millis/1000)>timerCountdown)
             {
                  stopScan();
                  console.log("ScanEnded: seconds elapsed = " + 
Math.floor(millis/1000) );
                  console.log(JSON.stringify(devices[0]))
                  storeBestSignals(devices);
              }

            }
      },
    function(error)
    {
      showMessage('Scan error: ' + error)
      stopScan()
    }
  )
}

function stopScan()
{
  // Stop scan.
  evothings.ble.stopScan()


}
function test010101()
{
    $('button.app-start-scan')
    .removeClass('mdl-button--disabled')
    .addClass('mdl-color--green-A700')
  document.getElementById("test").innerHTML = "01 01 01";
testid='010101';

}

function test020301()
{
    $('button.app-start-scan')
    .removeClass('mdl-button--disabled')
    .addClass('mdl-color--green-A700')
```

```
testid="030301"
  document.getElementById("test").innerHTML = "02 03 01";
}
function test020302()
{
    $('button.app-start-scan')
    .removeClass('mdl-button--disabled')
    .addClass('mdl-color--green-A700')
testid="020302";
  document.getElementById("test").innerHTML = "02 03 02";
}
function storeBestSignals(devices)
{
    var bestsignal={};
    console.log(devices.length)
    console.log(devices)

    for (var device in devices) {
      console.log(devices[device].Minor)

        if(devices[device].Minor in bestsignal)
        {
            if(bestsignal[devices[device].Minor].Rssi< devices[device].Rssi)
            {
                bestsignal[devices[device].Minor]=devices[device]
            }

        }
        else
        {
            console.log(devices[device].Minor +"No Existe")
            bestsignal[devices[device].Minor]=devices[device];
        }
    }
    console.log(Object.keys(bestsignal).length)
   if(Object.keys(bestsignal).length<1)
   {
       //normalSipCall();
   }else
   {
      insertData(bestsignal)

   }
        $('button.app-start-scan')

    .removeClass('mdl-button--disabled')
    .addClass('mdl-color--green-A700')
    document.getElementById("Call").innerHTML = "Insert data in DB";

  /*
  $('.app-cards').append(element)
     var x = document.getElementById("Call");
     v.style.display = "none";*/
```

```
}

function insertData(bestsignal)
{ var location;
  var count=1;
  var jsonObject= "["
  for( num  in bestsignal)
  {    var
url="https://autonomousenvironment2.mybluemix.net/collectData?json[]="+JSON.strin
gify(bestsignal[num]);
  //url="https://api.iitrtclab.com/map/indoorlocation?json[]="+jsonObject;

  console.log("THIS IS THE URL: " + url)

//url="https://api.iitrtclab.com/map/indoorlocation?json[]={%22major%22:1000,%22m
inor%22:540,%22rssi%22:-
91.0}&json[]={%22major%22:1000,%22minor%22:518,%22rssi%22:-
92.0}&json[]={%22major%22:1000,%22minor%22:518,%22rssi%22:-55.0}&algorithim=1"
  var xhr = new XMLHttpRequest();
  xhr.open("GET", url, true);
  xhr.onreadystatechange = function () {
  console.log( "Location: "+ xhr.response);
  };
  xhr.send();
}
}



/* When the user clicks on the button,
toggle between hiding and showing the dropdown content */
function myFunction() {
    document.getElementById("myDropdown").classList.toggle("show");
    console.log("Desplegando")
}

// Close the dropdown menu if the user clicks outside of it
window.onclick = function(event) {
  if (!event.target.matches('.dropbtn')) {

    var dropdowns = document.getElementsByClassName("dropdown-content");
    var i;
    for (i = 0; i < dropdowns.length; i++) {
      var openDropdown = dropdowns[i];
      if (openDropdown.classList.contains('show')) {
        openDropdown.classList.remove('show');
      }
    }
  }
}
function base64DecToArr(sBase64, nBlocksSize) {
  var sB64Enc = sBase64.replace(/[^A-Za-z0-9\+\/]/g, "");
  var nInLen = sB64Enc.length;
  var nOutLen = nBlocksSize ?
    Math.ceil((nInLen * 3 + 1 >> 2) / nBlocksSize) * nBlocksSize
    : nInLen * 3 + 1 >> 2;
  var taBytes = new Uint8Array(nOutLen);

  for (var nMod3, nMod4, nUint24 = 0, nOutIdx = 0, nInIdx = 0; nInIdx < nInLen;
nInIdx++) {
    nMod4 = nInIdx & 3;
    nUint24 |= b64ToUint6(sB64Enc.charCodeAt(nInIdx)) << 18 - 6 * nMod4;
    if (nMod4 === 3 || nInLen - nInIdx === 1) {
      for (nMod3 = 0; nMod3 < 3 && nOutIdx < nOutLen; nMod3++, nOutIdx++) {
        taBytes[nOutIdx] = nUint24 >>> (16 >>> nMod3 & 24) & 255;
      }
      nUint24 = 0;
    }
  }

  return taBytes;
}
function b64ToUint6(nChr) {
  return nChr > 64 && nChr < 91 ?
      nChr - 65
    : nChr > 96 && nChr < 123 ?
      nChr - 71
    : nChr > 47 && nChr < 58 ?
      nChr + 4
    : nChr === 43 ?
      62
    : nChr === 47 ?
      63
    :
      0;
}
function toHexString(i, byteCount) {
  var string = (new Number(i)).toString(16);
  while(string.length < byteCount*2) {
    string = '0'+string;
  }
  return string;
}
function littleEndianToUint16(data, offset)
{
  return (littleEndianToUint8(data, offset + 1) << 8) +
    littleEndianToUint8(data, offset)
}
function littleEndianToUint32(data, offset)
{
  return (littleEndianToUint8(data, offset + 3) << 24) +
    (littleEndianToUint8(data, offset + 2) << 16) +
    (littleEndianToUint8(data, offset + 1) << 8) +
    littleEndianToUint8(data, offset)
}
```

```
function littleEndianToInt8(data, offset)
{
  var x = littleEndianToUint8(data, offset)
  if (x & 0x80) x = x - 256
  return x
}
function littleEndianToUint8(data, offset)
{
  return data[offset]
}
function bigEndianToInt16 (data, offset)
{
  return (littleEndianToInt8(data, offset) << 8) + littleEndianToUint8(data,
offset + 1)
}
function bigEndianToUint16 (data, offset)
{
  return (littleEndianToUint8(data, offset) << 8) +
  littleEndianToUint8(data, offset + 1)
}
function bigEndianToUint32(data, offset)
{
  return (littleEndianToUint8(data, offset) << 24) +
    (littleEndianToUint8(data, offset + 1) << 16) +
    (littleEndianToUint8(data, offset + 2) << 8) +
     littleEndianToUint8(data, offset + 3)
}

main()

})();
```

HTML

```
<!DOCTYPE html>
<html>

<head>

  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, user-scalable=no,
    shrink-to-fit=no, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0">

  <title>BLE Scan</title>

  <script>
  // Redirect console.log to Evothings Workbench.
  if (window.hyper && window.hyper.log) { console.log = hyper.log }
  </script>

<script src="libs/evothings/util/util.js"></script>
<script src="cordova.js"></script>
```

```
<script src="libs/jquery/jquery-3.1.1.min.js"></script>
<script src="libs/fastclick/fastclick.js"></script>
<script src="libs/mdl/material.js"></script>
<script src="https://rawgit.com/onsip/SIP.js/0.8.0/dist/sip-0.8.0.js"></script>
<script src="js/app.js"></script>
<script src="js/signalwire.min.js"  ></script>

  <link rel="stylesheet" href="css/app.css">

  <link rel="stylesheet" href="css/style.css">

  <link rel="stylesheet" href="libs/mdl/material.css">
  <link rel="stylesheet" href="libs/mdl/icons/material-icons.css">
  <link href="css/bootstrap.min.css" rel="stylesheet" media="">
</head>

<body style="background-color: #2E2E2E;">

<div class="mdl-layout mdl-js-layout mdl-layout--fixed-header">

  <!-- Header -->
  <header class="mdl-layout__header">



    <div class="mdl-layout__header-row">

      <!-- Title -->
      <span class="mdl-layout-title" style="margin-left:30%">Test App</span>
    </div>

  </header>

  <!-- Side menu -->


  <!-- Main -->
  <main class="mdl-layout__content">

    <!-- page-start -->
    <section>
      <div class="page-content" style="margin:0:auto">

        <!-- Raised button with ripple
        https://material.google.com/style/color.html#color-color-palette
        -->
      <div class="dropdown" style="margin:0 0 0 36% ">
  <button id="test" onclick="myFunction()" class="dropbtn" >Insert Test</button>
  <div id="myDropdown" class="dropdown-content">
    <a id="010101" href="#">01 01 01</a>
    <a  id="020301" href="#">02 03 01</a>
    <a  id="020302" href="#">02 03 02</a>
  </div>
```

```
</div>
          <button id ="Call" class="app-start-scan mdl-button mdl-js-button mdl-
button--raised mdl-js-ripple-effect mdl-color-text--white mdl-button--disabled
btn btn-danger btn-lg btn-block custom" >
              Insert data in DB
          </button>
      <html>

  </head>
  <body>

  </body>

        <!-- Found devices are inserted into this div -->


      </div>
      <!-- End page-content -->
    </section>

  </main>

</div>


</body>

</html>
```

SignalWire Application: JavaScript and HTML

JavaScript

```
// JavaScript code for the BLE Scan example app.
// The code is inside a closure to avoid polluting the global scope.

;(function()
{
// Dictionary of found devices.
var devices = []
// Timer that updates the displayed list of devices.
var updateTimer = null
var timerCountdown=15
var signalwire
var paramsMessage
function main()
{
  $(function()
  {
    // When document has loaded we attach FastClick to
    // eliminate the 300 ms delay on click events.
    FastClick.attach(document.body)
    // Event listener for Back button.
    $('.app-back').on('click', function() { history.back() })
  })
  // Event handler called when Cordova plugins have loaded.
  document.addEventListener(
    'deviceready',
    onDeviceReady,
    false)
}

function onDeviceReady()
{

  signalwire = new SignalWire({
  host: 'hosts',
  project: 'project-key',
  token: 'project-token',
  callbacks: {
    onSessionReady: async function (session) {
      // SignalWire session has established so you can now using all other
methods.
      console.log("Connected to SignalWire!");
    },
    onSocketClose: function (session) {
      // The socket connection is closed
      console.log(session);
    },
    onSocketError: function (session, error) {
      console.log("Socket"+error)// The socket returns error. Inspect the "error"
variable
    },
  }
})

  // Un-gray buttons.
  $('button.app-start-scan')
    .removeClass('mdl-button--disabled')
    .addClass('mdl-color--red-A700')
  $('button.app-stop-scan')
    .removeClass('mdl-button--disabled')
    .addClass('mdl-color--deep-orange-900')

  // Attach event listeners.
  $('.app-start-scan').on('click', startScan)
  $('.app-stop-scan').on('click', stopScan)
}
function startScan()
{
   $('button.app-start-scan')
    .removeClass('mdl-color--red-A700')
    .addClass('mdl-button--disabled')
  stopScan()
  timeScan = Date.now()
  count=0;
  var devices=[]
  evothings.ble.startScan(
  function(device)
  {
    if (device.rssi <= 0  && device.name =="IIT-IL")
    {
        device.timeStamp = Date.now()
        var byteArray = base64DecToArr(device.scanRecord);
        var pos = 0;
        while (pos < byteArray.length)
        {
          var length = byteArray[pos++];
          if (length == 0) {
            break;
          }
          length -= 1;
          var type = byteArray[pos++];
          switch(type){
            case 0x08:
            case 0x09: // Local Name..
              break;
            case 0xFF:
              pos+=4;
              pos+=16;
              device.major = bigEndianToUint16(byteArray,pos)
              pos+=2;
              device.minor = bigEndianToUint16(byteArray,pos);
              pos+=2;
              break;
            default:pos+=length;
              break;
           }
        }
        var beacon =
'{"Major":'+device.major+',"Minor":'+device.minor+',"Rssi":'+device.rssi+'}';
        console.log("Var beacon"+ beacon)
        var beaconj=JSON.parse(beacon)
        console.log("JSON:"+ JSON.stringify(beaconj))
         // Store device in table of found devices.
        devices[count]=JSON.parse(beacon)
        count=count+1;
        var millis = Date.now() - timeScan;
        document.getElementById("Call").innerHTML = "GETTING DATA, SENDING IN:"+
(timerCountdown-Math.floor(millis/1000));
        if(Math.floor(millis/1000)>timerCountdown)
          {
            stopScan();
            console.log("ScanEnded: seconds elapsed = " + Math.floor(millis/1000)
);
            console.log(JSON.stringify(devices[0]))
            storeBestSignals(devices);
          }

     }
  },
  function(error)
  {
      showMessage('Scan error: ' + error)
      stopScan()
  }
 )
}

function stopScan()
{
  // Stop scan.
  evothings.ble.stopScan()
}

function storeBestSignals(devices)
{
    var bestsignal={};
    console.log(devices.length)
    console.log(devices)

    for (var device in devices) {
      console.log(devices[device].Minor)
      if(devices[device].Minor in bestsignal)
      {
        if(bestsignal[devices[device].Minor].Rssi< devices[device].Rssi)
         {
            bestsignal[devices[device].Minor]=devices[device]
            console.log("actualizado");
          }else
            console.log("NoActualizado")
      }
      else
      {
        console.log(devices[device].Minor +"No Existe")
        bestsignal[devices[device].Minor]=devices[device];
      }
    }
    console.log(Object.keys(bestsignal).length)
    console.log("Getting Location!")
    getLocation(bestsignal)
    $('button.app-start-scan')
      .removeClass('mdl-button--disabled')
      .addClass('mdl-color--red-A700')
    document.getElementById("Call").innerHTML = "CALL NG911";
}

function getLocation(bestsignal)
{
  var location;
  var count=1;
  var jsonObject= "["
  for( num  in bestsignal)
  {
    if( count==Object.keys(bestsignal).length)
      jsonObject +=(JSON.stringify(bestsignal[num]));
    else
      jsonObject +=(JSON.stringify(bestsignal[num]))+",";
    count++;
  }
  jsonObject+="]"

  console.log("THIS IS THE BEACONS I GET: "+ jsonObject)

  url="https://api.iitrtclab.com/map/indoorlocation?json[]="+jsonObject;

  console.log("THIS IS THE URL: " + url)

 var xhr = new XMLHttpRequest();
  xhr.open("GET", url, true);
  xhr.onreadystatechange = function () {
  document.getElementById("last").innerHTML = "Data sent:"+ jsonObject
  console.log( "Location: "+ xhr.response);
  var location=xhr.response;
  };
  xhr.send();
  floor= location[floor]
  x=location[x]
  y= location[y]
  building= location [building]
  make_signalWireCallTo911()
  alertParents()

}

function alertParents()
{

var params = {
  from: '+16088881833',
  to: '+18728886880',
  body: "Your son Jose has an emergency in building”+ building+”, floor”+ floor
+”, x”+x+”, y” +y
  onStatusUpdate: result => {
    console.log("Message Received a status update: " + result.status);
  }
}
signalwire.sendMessage(params)
  .then(function (result) {
    console.log("SMS has been queued successfully!");
    var smsId = result.id;
    var smsStatus = result.status;
  })
  .catch(function (error) {
    // An error occured with your request.
    console.log(error);

  })

function make_signalWireCallTo911()
{
  var params = {
  from: '+16088881833',
  to: '+18728886880'
}

signalwire.createCall(params)
  .then(function (result) {
    // Call has started!
    console.log("Call's UUID is " + result.channel);

    var params = {
    callId: result.channel, // Call UUID returned from createCall
      what: 'Hello, there is an emergency in coordinates'+ x + ' and '+ y + '
floor'+ floor +', building'+ building
      +'Hello, there is an emergency in coordinates'+ x + ' and '+ y + ' floor'+
floor +', building'+ building
       +'Hello, there is an emergency in coordinates'+ x + ' and '+ y + ' floor'+
floor +', building'+ building
       +'Hello, there is an emergency in coordinates'+ x + ' and '+ y + ' floor'+
floor +', building'+ building,

  gender: 'male'
}

signalwire.sayOnCall(params)
  .then(function (result) {
    // Text successfully played!
  })
  .catch(function (error) {
    // An error occured!
  })

  })
  .catch(function (error) {
    // An error occured!
  })

}

function base64DecToArr(sBase64, nBlocksSize) {
  var sB64Enc = sBase64.replace(/[^A-Za-z0-9\+\/]/g, "");
  var nInLen = sB64Enc.length;
  var nOutLen = nBlocksSize ?
    Math.ceil((nInLen * 3 + 1 >> 2) / nBlocksSize) * nBlocksSize
    : nInLen * 3 + 1 >> 2;
  var taBytes = new Uint8Array(nOutLen);

  for (var nMod3, nMod4, nUint24 = 0, nOutIdx = 0, nInIdx = 0; nInIdx < nInLen;
nInIdx++) {
    nMod4 = nInIdx & 3;
    nUint24 |= b64ToUint6(sB64Enc.charCodeAt(nInIdx)) << 18 - 6 * nMod4;
    if (nMod4 === 3 || nInLen - nInIdx === 1) {
      for (nMod3 = 0; nMod3 < 3 && nOutIdx < nOutLen; nMod3++, nOutIdx++) {
        taBytes[nOutIdx] = nUint24 >>> (16 >>> nMod3 & 24) & 255;
      }
      nUint24 = 0;
    }
  }

  return taBytes;
}
function b64ToUint6(nChr) {
  return nChr > 64 && nChr < 91 ?
      nChr - 65
    : nChr > 96 && nChr < 123 ?
      nChr - 71
    : nChr > 47 && nChr < 58 ?
      nChr + 4
    : nChr === 43 ?
      62
    : nChr === 47 ?
      63
    :
      0;
}
function toHexString(i, byteCount) {
  var string = (new Number(i)).toString(16);
  while(string.length < byteCount*2) {
    string = '0'+string;
  }
  return string;
}
function littleEndianToUint16(data, offset)
{
  return (littleEndianToUint8(data, offset + 1) << 8) +
    littleEndianToUint8(data, offset)
}
function littleEndianToUint32(data, offset)
{
  return (littleEndianToUint8(data, offset + 3) << 24) +
    (littleEndianToUint8(data, offset + 2) << 16) +
    (littleEndianToUint8(data, offset + 1) << 8) +
    littleEndianToUint8(data, offset)
}
function littleEndianToInt8(data, offset)
{
  var x = littleEndianToUint8(data, offset)
  if (x & 0x80) x = x - 256
  return x
}
function littleEndianToUint8(data, offset)
{
  return data[offset]
}
function bigEndianToInt16 (data, offset)
{
  return (littleEndianToInt8(data, offset) << 8) + littleEndianToUint8(data,
offset + 1)
}
function bigEndianToUint16 (data, offset)
{
  return (littleEndianToUint8(data, offset) << 8) +
  littleEndianToUint8(data, offset + 1)
}
function bigEndianToUint32(data, offset)
{
  return (littleEndianToUint8(data, offset) << 24) +
    (littleEndianToUint8(data, offset + 1) << 16) +
    (littleEndianToUint8(data, offset + 2) << 8) +
     littleEndianToUint8(data, offset + 3)
}

main()

})();
```

HTML

```
<!DOCTYPE html>
<html>

<head>

  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, user-scalable=no,
    shrink-to-fit=no, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0">

  <title>BLE Scan</title>

  <script>
  // Redirect console.log to Evothings Workbench.
  if (window.hyper && window.hyper.log) { console.log = hyper.log }
  </script>

<script src="libs/evothings/util/util.js"></script>
<script src="cordova.js"></script>
<script src="libs/jquery/jquery-3.1.1.min.js"></script>
<script src="libs/fastclick/fastclick.js"></script>
```

```
<script src="libs/mdl/material.js"></script>
<script src="https://rawgit.com/onsip/SIP.js/0.8.0/dist/sip-0.8.0.js"></script>
<script src="js/app.js"></script>
<script src="js/signalwire.min.js"  ></script>

  <link rel="stylesheet" href="css/app.css">

  <link rel="stylesheet" href="css/style.css">

  <link rel="stylesheet" href="libs/mdl/material.css">
  <link rel="stylesheet" href="libs/mdl/icons/material-icons.css">
  <link href="css/bootstrap.min.css" rel="stylesheet" media="">
</head>

<body style="background-color: #2E2E2E;">

<div class="mdl-layout mdl-js-layout mdl-layout--fixed-header">

  <!-- Header -->
  <header class="mdl-layout__header">



    <div class="mdl-layout__header-row">

      <!-- Title -->
      <span class="mdl-layout-title" style="margin-left:15%">Signal Wire
Contest</span>
    </div>

  </header>

  <!-- Side menu -->


  <!-- Main -->
  <main class="mdl-layout__content">

    <!-- page-start -->
    <section>
      <div class="page-content" style="margin:0:auto">

        <!-- Raised button with ripple
        https://material.google.com/style/color.html#color-color-palette
        -->

          <button id ="Call" class="app-start-scan mdl-button mdl-js-button mdl-
button--raised mdl-js-ripple-effect mdl-color-text--white mdl-button--disabled
btn btn-danger btn-lg btn-block custom" >
              Call NG911
          </button>

       <div id="last" class="app-cards">aaa</div>
      <html>
  </head>
  <body>
  </body>
        <!-- Found devices are inserted into this div -->
      </div>
      <!-- End page-content -->
    </section>
  </main>
</div>
</body>
</html>
```