International Journal of Interdisciplinary Telecommunications and Networking
Volume 12 • Issue 2 • April-June 2020

Real-Time Indoor Geolocation Tracking for Assisted Healthcare Facilities

Kinjal Gala, San Diego State University, San Diego, USA
Paul David Bryden, University of Strathclyde, Glasgow, Scotland
Christopher Paolini, San Diego State University, San Diego, USA
https://orcid.org/0000-0001-6563-917X
Matthew Wang, University of California, Los Angeles, USA
Albena Dimitrova Mihovska, Aarhus University, Aarhus, Denmark
Mahasweta Sarkar, San Diego State University, San Diego, USA

ABSTRACT

A leading cause of physical injury sustained by elderly persons is the event of unintentionally falling. A delay between the time of fall and the time of medical attention can exacerbate injury if the fall resulted in a concussion, traumatic brain injury, or bone fracture. The authors present a solution capable of finding and tracking, in real-time, the location of an elderly person within an indoor facility, using only existing Wi-Fi infrastructure. This paper discusses the development of an open source software framework capable of finding the location of an individual within 3m accuracy using 802.11 Wi-Fi in good coverage areas. This framework is comprised of an embedded software layer, a Web Services layer, and a mobile application for monitoring the location of individuals, calculated using trilateration, with Kalman filtering employed to reduce the effect of multipath interference. The solution provides a real-time, low cost, extendible solution to the problem of indoor geolocation to mitigate potential harm to elderly persons who have fallen and require immediate medical help.

KEYWORDS

Assisted Healthcare Technology, Indoor Geolocation, Indoor Location System, Indoor Positioning System, Kalman Filtering, Location Tracking, Trilateration, Wireless Sensor Networks

INTRODUCTION

As reported by the World Health Organization's Global Report on Falls Prevention in Older Age, based on research conducted by Blake et al. (Blake et al., 1988), (*WHO Global Report on Falls Prevention in Older Age*, 2007), 35% of population 65 years and over experience at least one unintentional fall per year, due to tripping, dizziness, and blackouts. Elderly people are more likely to fall due to loss of handgrip strength used for control and stabilization when using walking aids such as walkers and canes. Arthritis, dizziness, neuromuscular, cognitive, and foot impairments also contribute to an increased prevalence of falling by older people when using stairs and steps, or while turning around or reaching for objects. Research by Tinetti found that 61% of elderly nursing home residents fell at some point during their first year of residence (Tinetti, 1987), a greater proportion than elderly people who live in residential communities. Those assigned to live in assisted living facilities may fall more frequently due to greater weakness in their lower extremities, a potential contributing factor for their

DOI: 10.4018/IJITN.2020040101

decision to live in such a facility since facility staff can assist with mobility. Many algorithms and embedded technologies have been in research and development for the past several years, which aim to work towards the basic goal of detecting a fall and reporting the location of the fallen individual to a respective authority.

Indoor geolocation tracking is an evolving technology which aims to calculate the location of an individual within an indoor environment. One of the promising classes of this emerging technology is to take advantage of existing infrastructure to determine the location of a user (Pahlavan, Xinrong, & Makela, 2002). Much work has been carried out to use GPS for indoor location tracking (Álvarez, de Cos, Lorenzo, & Las-Heras, 2010; Kjærgaard et al., 2010); however, GPS only has an accuracy of 5-50m in an indoor environment (Liu, Darabi, Banerjee, & Liu, 2007).

This paper presents the development of an open source software platform capable of tracking the movements of individuals using existing 802.11 Wi-Fi infrastructure. The modularity of the software framework developed and its strong use of C++11 design principles enables the framework to be easily adapted to a variety of hardware platforms and radio technologies, such as Bluetooth or LoraWAN. In addition, this solution provides flexibility of monitoring multiple people using a mobile app.

BACKGROUND

The ability to identify the geographic location of an individual residing outside of a building by latitude, longitude, and altitude is easily accomplished using a relatively inexpensive GPS receiver. Typical commercially available receivers for under $100 can provide coordinates within a sampling time of 30 seconds to an accuracy of 3m. For example, the Copernicus II 12-channel GPS module from Trimble is under $70 and can provide updated coordinates with a period of 3s. GPS receivers typically use a carrier wave in the L1 band at 1575.42 MHz on which navigation messages are modulated. Unfortunately, such microwave signals are significantly attenuated by building roofs and walls, rendering GPS unusable in indoor setups.

Indoor position measurements can be accomplished using different mechanisms such as radio signals, magnetic fields, and sound waves. Newer, emerging technologies employ computer vision to identify objects in a camera field of a view. A vision system can measure distances in between recognized objects, and between a user and recognized objects. These measurements provide a system with depth perception and can identify how far a user is away from a surface or other rigid body in a field of view.

The well-established Cricket indoor location system developed at MIT uses a combination of RF and ultrasound to provide location information via wall- and ceiling-mounted beacons placed throughout a building (Priyantha, Chakraborty, & Balakrishnan, 2000), (Priyantha, Miu, Balakrishnan, & Teller, 2001), (Teller, Chen, & Balakrishnan, 2003). Cricket uses *time difference of arrival* between RF and ultrasonic signals, which can accurately identify the indoor location of a static object but was shown to present difficulties tracking the location of an object in motion.

The indoor location of an object or person using received signal strength was investigated by (Álvarez et al., 2010), with the goal of improving the characterization of EM to provide a precise indoor location. Álvarez proposed a full wave-based method measurement setup and tested this idea in different scenarios. A practical implementation was carried out using a ZigBee-based sensor network and was able to achieve a desired accuracy requirement of less than 5% for location error.

Issues faced by estimating indoor location from received signal strength, time of arrival (TOA), or time difference of arrival was investigated by (Chitte & Dasgupta, 2008) who found the mean square error (MSE) of location measurement increases exponentially with noise power.

Topological discovery with boundary recognition and hole discovery in a wireless sensor network can be accomplished using methods based on the Poincare-Perelman Theorem (Wei, Yang, Shen, & Zhou, 2012).

A system named Simultaneous Localization and Configuration (SLAC) to locate devices and motion data from users was proposed by (Bulten, Rossum, & Haselager, 2016). The authors showed that room level accuracy can be established while simultaneously protecting the privacy of people being monitored.

Difficulties faced in indoor geolocation science, and requirements for design and performance evaluation of emerging geolocation systems was studied by (Pahlavan et al., 2002), where two emerging classes of indoor geolocation systems were identified. The first class uses reliable TOA measurements based on wideband, super resolution, or UWB location sensing approaches and employs triangulation techniques for positioning. The second class uses existing Wi-Fi infrastructure, more unreliable metrics, premeasurement data, and pattern recognition algorithms. The major drawback of using pattern recognition is that it requires generation and maintenance of a signature database, which is difficult to maintain for environments that continuously change. Both classes showed promising performance for emerging indoor geolocation applications.

This project utilizes strictly 802.11 radio signals from Wi-Fi access points to identify a person's position. After calculating the known location of these access points and the signal strength at a receiver, the distance can be derived using a modification of the free space equation. In this software framework, four access points are selected to derive the location of a user by calculating the distance between the user and each access point using trilateration.

SOFTWARE FRAMEWORK ARCHITECTURE

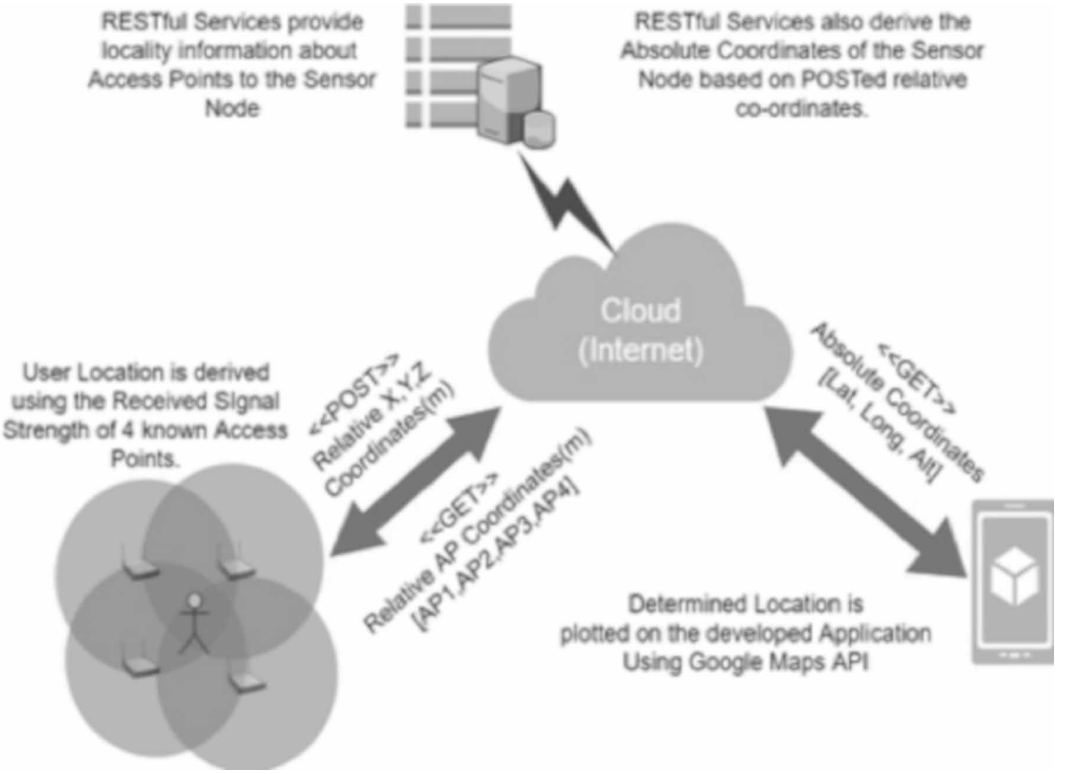
The software framework, presented in this paper, is com- posed of three components: (1) *Embedded Layer*, (2) *Middleware Layer*, and (3) *Mobile Application Layer*. The embedded layer is comprised of software which runs on a sensor node held by a user. The middleware layer is comprised of RESTful Java Web Services and a PostgreSQL database, which connects the Embedded and Mobile Application layers. The mobile application layer consists of an Android application, which is used to render a user's location. A diagram of the complete software framework is shown in Figure 1. The RESTful API is the fundamental interface between all components of the software framework. In terms of data flow, Web Services provide the relative coordinates of known access points to the sensor node, and the node responds with the relative Cartesian (x, y, z) coordinates (in meters) of its location. The Web Services then translate these coordinates into absolute latitude, longitude, and altitude units which are consumed by a mobile application. The three respective layers and their operation will now be discussed in detail.

A. The Embedded Layer

The embedded layer is the component of the software framework that runs on a sensory device operated or worn by a user. The embedded layer is written in C++ using object-oriented (OO) design principles and is portable across Linux, Windows, iOS, Android, and other platforms. This portability is possible due to the use of modern OO design principles and the employment of cross-platform libraries, such as BOOST and *CPPRestSDK*, which removes platform awareness for most of the system. As shown in Figure 2, the only part of the system that requires awareness of the underlying OS/Platform is the Node Scanner Module. A Node in the framework is defined as an 802.11 access point. Direct OS interaction is required at this layer due to the low level of access required to the wireless hardware. BOOST is used mainly to enable straightforward, cross-platform, thread safety and thread handling. The C++ REST SDK Microsoft project (CPPRestSDK) is used to handle JSON construction and parsing along with HTTP Requests.

The Node Reader Module is responsible for creating a known list of nodes and their locations. This has been implemented in two different flavors: a file reader and an HTTP GET reader. The recommended module to utilize is the HTTP GET reader. This module queries a known list of nodes

Figure 1. Software framework architecture overview



in JSON format from the middleware Java Web Services and populates a list of nodes known to the Location and REST Modules.

The Node Scanner Module is responsible for interfacing with the OS or Platform API, which provides data from a Wi-Fi scan or other radio platform. The module provides a list of Nodes populated with their respective RSS (Received Signal Strength).

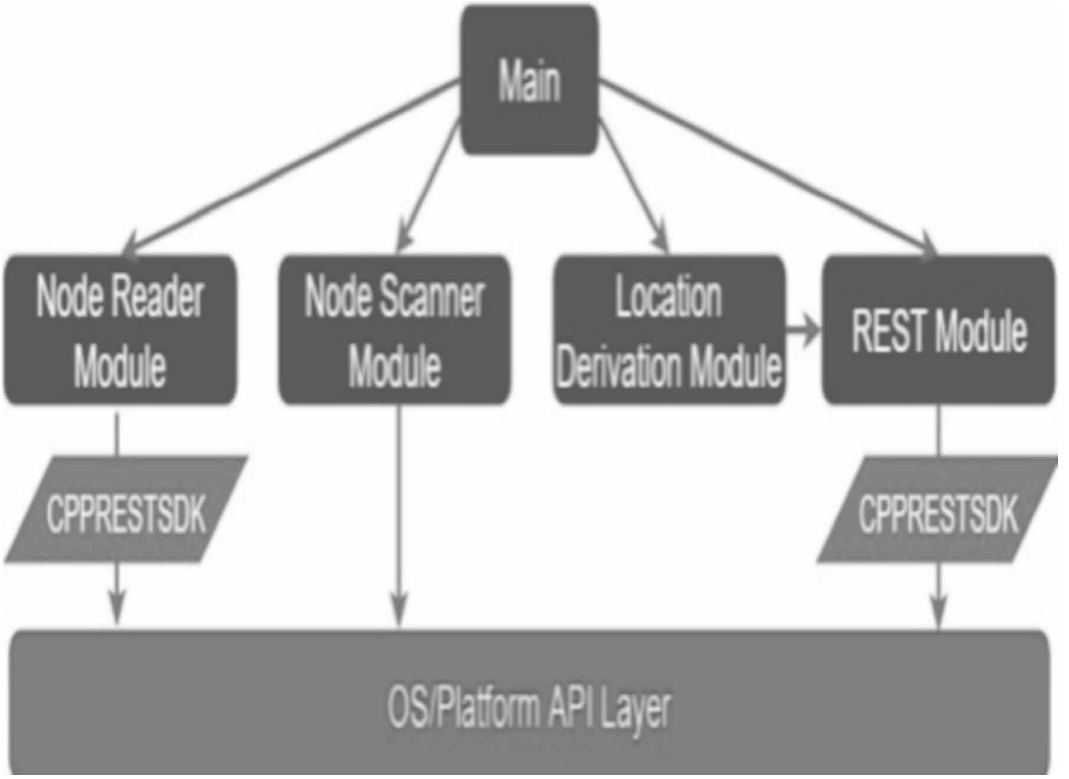
The Location Derivation Module is responsible for deriving the Cartesian (x, y, z) coordinates of the user node, based on data from the Node Reader and Node Scanner Modules. The details of the implementation will be discussed in the theory of trilateration and implementation sections.

The REST Module is responsible for posting data from the Location Derivation Module to the Web Services. This module is also responsible for providing the data to any interested party through an HTTP GET route. Trilateration is used to derive the relative location of a particular sensor node and depends on the measured Received Signal Strength (RSS) of four different Wi-Fi access points (APs) within 32m of the user. Typically, the received signal power within one meter of an AP will be approximately -32 dBm, and close to -90 dBm at a distance of 32m. RSS can be modeled using equation (1)

$$RSS = -\left(10N \log_{10} D - A\right)$$

where N represents a signal propagation constant, A the received signal power at 1m, and D a distance in meters (Aamodt, July 10, 2006; Chitte & Dasgupta, 2008). With four APs, one can use trilateration to locate a user using four distances computed from four RSS values. Let (x, y, z) represent

Figure 2. Embedded layer architecture



the unknown location of a user and (d_1, d_2, d_3, d_4) represent four distance values obtained from the measured RSS between a user's sensor and four APs. The system of Euclidian distances between the user and four APs is given by (2)

$$\begin{aligned}
\left(x - x_1\right)^2 + \left(y - y_1\right)^2 + \left(z - z_1\right)^2 &= d_1^2 \\
\left(x - x_2\right)^2 + \left(y - y_2\right)^2 + \left(z - z_2\right)^2 &= d_2^2 \\
\left(x - x_3\right)^2 + \left(y - y_3\right)^2 + \left(z - z_3\right)^2 &= d_3^2 \\
\left(x - x_4\right)^2 + \left(y - y_4\right)^2 + \left(z - z_4\right)^2 &= d_4^2
\end{aligned}$$

Subtracting rows 2, 3, and 4 each from row 1, we have

$$\begin{aligned}
2\left(x_2 - x_1\right)x + 2\left(y_2 - y_1\right)y + 2\left(z_2 - z_1\right)z &= x_2^2 - x_1^2 + y_2^2 - y_1^2 + z_2^2 - z_1^2 - d_2^2 + d_1^2 \\
2\left(x_3 - x_1\right)x + 2\left(y_3 - y_1\right)y + 2\left(z_3 - z_1\right)z &= x_3^2 - x_1^2 + y_3^2 - y_1^2 + z_3^2 - z_1^2 - d_3^2 + d_1^2 \\
2\left(x_4 - x_1\right)x + 2\left(y_4 - y_1\right)y + 2\left(z_4 - z_1\right)z &= x_4^2 - x_1^2 + y_4^2 - y_1^2 + z_4^2 - z_1^2 - d_4^2 + d_1^2
\end{aligned}$$

System (3) can be expressed in $A\bar{x} = \bar{b}$ matrix form as

$$\begin{bmatrix} (x_2 - x_1) & (y_2 - y_1) & (z_2 - z_1) \\ (x_3 - x_1) & (y_3 - y_1) & (z_3 - z_1) \\ (x_4 - x_1) & (y_4 - y_1) & (z_4 - z_1) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_2^2 - x_1^2 + y_2^2 - y_1^2 + z_2^2 - z_1^2 - d_2^2 + d_1^2 \\ x_3^2 - x_1^2 + y_3^2 - y_1^2 + z_3^2 - z_1^2 - d_3^2 + d_1^2 \\ x_4^2 - x_1^2 + y_4^2 - y_1^2 + z_4^2 - z_1^2 - d_4^2 + d_1^2 \end{bmatrix}$$

and solved using a least squares approximation (Xu, Ge, Chen, Chen, & Ling, 2012). We then solve the general over-determined system using a Gauss–Newton Least Squares Estimate:

$$\left(A^T A\right) X = A^T b$$

While the same four access points are being used, the coefficient matrix *A* does not vary, and the Gramian A^TA can be factored using LU decomposition and can be solved repeatedly using back substitution to obtain a least squares estimate.

Multipath effects (He, Geng, & Pahlavan, 2014), co-channel interference ("CCI"), and adjacent-channel interference ("ACI") can reduce the SNR, resulting in location measurement error. Additional APs can be incorporated to reduce trilateration error. More than three APs form an overdetermined system of equations. We can form three equations in three unknowns by doing additional row operations, as we have shown in (4). The optimal signal propagation constant for a noisy indoor environment was found to be approximately 2.5. To mediate the error introduced by such environmental effects, Kalman filtering is employed. In an ideal system, an RSS value depends directly, and only, on the distance between the 802.11 Wi-Fi transmitter and the receiver. Multipath interference, electromagnetic interference, and signal fading all play a significant role in causing fluctuations in the received signal strength. Hence, it is required that the RSS values are filtered in the time domain so to mitigate the effect of such noise. One such method is to employ an extended Kalman filter. The equations representing the implemented Kalman filter and given in and are executed on each "tick" of the system (Lauszus, 2018, September 10, 2012),

$$\begin{aligned} p &= p + q \\ k &= p / (p + r) \\ x &= x + k\left(i - x\right) \\ p &= \left(1 - k\right) p \end{aligned}$$

where *q* is the process noise covariance, *r* is the measurement noise covariance, *x* is the Kalman filtered value, *p* is the Estimation Error Covariance, *k* is the Kalman Gain, and *i* is the most recently measured RSS value. Measurement Covariance is the standard deviation of a set of measured results, determined to be 2.36 dBm. This value was derived by taking several measurements of the RSSI (dBm) at 1m distance from an 802.11 wireless access point in the test environment. The process covariance is the susceptibility of the system to change, determined in this study to be 0.001. This value was determined through trials to best match the typical speed at which an individual may move around a facility.

The Embedded Layer was implemented in a very modular fashion using OO and C++11 design principles. These include but are not limited to polymorphism, inheritance, decorator, and singleton patterns. Such modern C++ features such as shared pointers, scoped locks, tasks and initializer lists were also heavily utilized. The employment of these principles and features has enabled a very flexible, thread-safe, memory safe system to be constructed and be portable across multiple platforms. The UML for the completed system can be seen in Figure 3. The Module classes are the core drivers

of the system. Each module provides the system with discrete functionality. Some modules, such as the Rest Module, rely on other modules to enable operation, however, they do not depend on the implementation of those modules. There are four key modules which make up this system: *Location Module*, *Rest Module*, *Scan Module*, and the *Node Reader Module*.

The Location Module Class is responsible for deriving the exact location of the user based on the four nodes with the best signal strength in the target node container. In this open source framework, the LAPACK linear algebra library is used to solve through calls to DGESV and DGELS for least squares approximation and DGETRS for back substitution. The Rest Module Class is responsible for providing the location, scan, and target node list data to any interested parties by hosting the JSON on a GET Route. The module is also responsible for POSTing derived location data to the remote Web Services server. The Scan Module Class is responsible for populating a list of nodes with the RSS evident at a Beaglebone (BeagleBoard.org Foundation, 2019) single board computer (SBC) for each access point. This list is then used to update the target nodes. The Node Reader Module class is responsible for populating a list of Target Nodes which are of known location. This list can be obtained from a file in JSON format using the file reader variant or from a web address using the GETreader variant.

Additionally, there are data classes used by all modules, such as the *Node*, *Node Container*, and *Location* classes that store information relating to access point information or user location information. Node class objects are used to identify any node identified in a scan. Target Nodes correspond to nodes in which the location is known and can be used for trilateration. Each target node is responsible for updating its stored Kalman filtered location when a new RSS value is provided. The location data class simply contains the (x, y, z) coordinates for a location and helper methods for accessing and modifying this data.

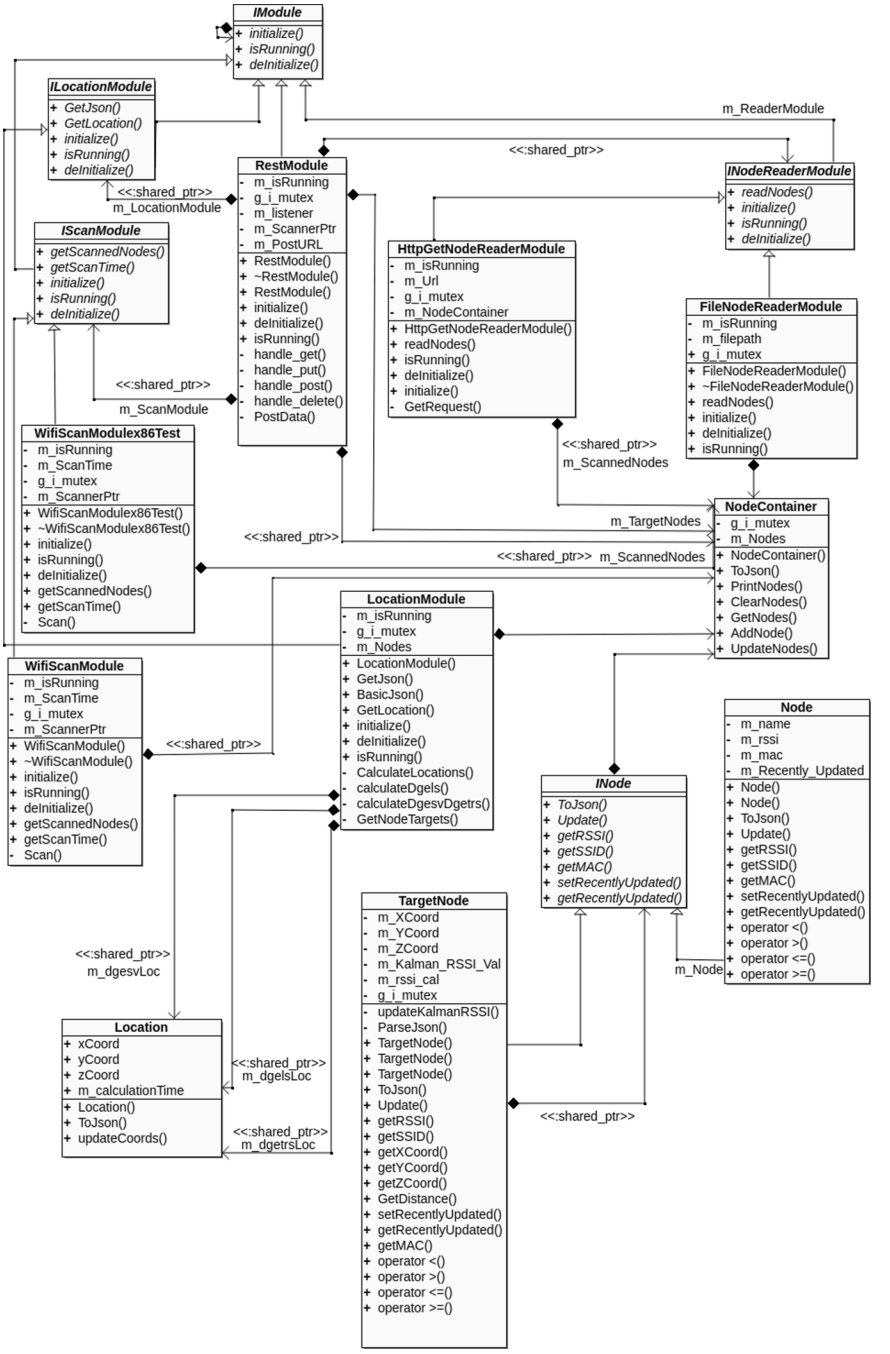
The system is driven by the main class, where all the modules are created and initialized. The main loop ensures the system is continuously scanning and updating a user's location. A target node container is created and shared between the location module, node reader module, and scan module. When the modules are initialized, the node reader module updates that container with a list of target nodes, of known location, which is updated with RSSI measurements as is necessary through the scan module. The location module then employs this information to derive relevant, up to date location information. This information is then POSTed by the REST Module to a web server and provided to a mobile app listener through an HTTP GET route.

B. The Middleware Layer

The middleware layer is comprised of two main components: RESTful Web Services and a PostgreSQL back-end database.

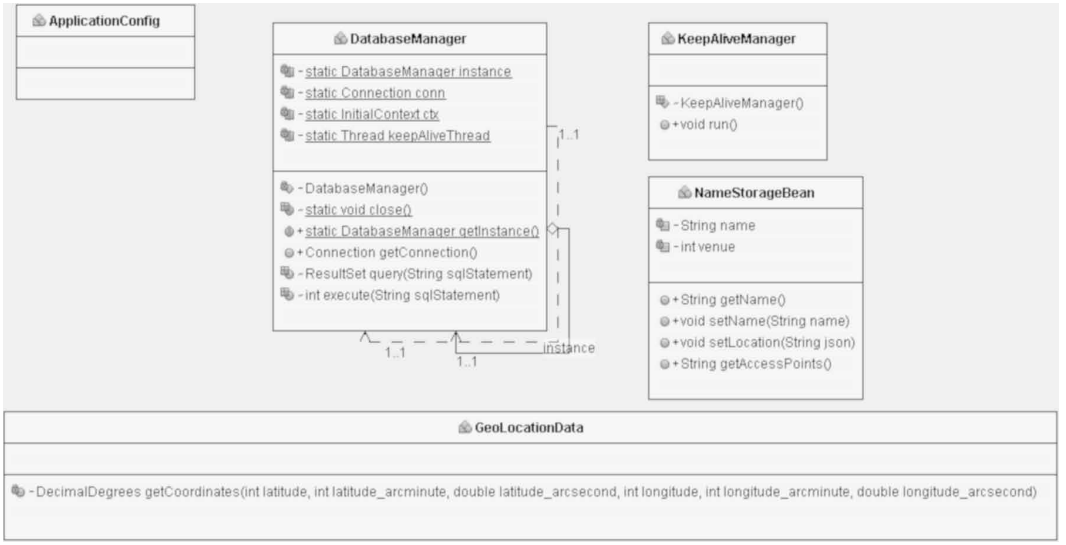
RESTful Web Services provide a API that both the Mobile Application and the Embedded Layer interact with. The back-end database is accessed through RESTful Web Service invocations and is where location data related to known access points and user locations are stored. RESTful (Representational State Transfer) architectural style Web Services were built using the Java Glassfish library and framework. These RESTful services provide an interface to query and retrieve data contained within the database, and a mechanism to write calculated location data to the database. Two HTTP GET routes were provided: (1) a route which lists the known access points for a location in JSON format, and (2) a route which lists a user's absolute location as (longitude, latitude, and altitude) coordinates. There is also an HTTP POST route which accepts Cartesian (x, y, z) coordinates in meters and a physical site number (in JSON format) derived by the embedded layer, and writes the corresponding absolute latitude, longitude, and altitude into a PostgreSQL database table. PostgreSQL was chosen as a platform to implement a backend database where all data retrieved through Web Service invocations is stored, such as the absolute, real-time location of persons. The database consists of several tables, including one containing information of known access point locations, and a moving window table containing the last 30 timestamped known locations of sensor nodes. To obtain the

Figure 3. UML diagram of the embedded layer



relative coordinates of each access point in a physical venue, such as a building, a common origin was placed in the southwest corner of each venue, then the absolute latitude and longitude of each access point were obtained by offsetting the latitude and longitude from the origin. Cartesian coordinates (m) of each AP relative to the corner of the building were stored in the database.

Figure 4. UML class diagram of the middleware layer



A UML class diagram for the RESTful Web Services that implement the middleware layer is shown in Figure 4. The database manager module maintains a persistent connection with a PostgreSQL database and handles errors that occur during a database connection. The *KeepAliveManager* class continuously sends queries to the database to maintain a persistent connection. The *NameStorageBean* class contains a method that updates the location of each sensor in the database using an SQL INSERT statement to insert new values into the database and DELETE to delete entries to be aged out. The *NameStorageBean* class also executes an SQL SELECT query operation to retrieve all Wi-Fi access point relative Cartesian coordinates from the database to provide a sensor for calculating distances. The *Geolocation* module consumes relative Cartesian coordinates from a sensor and converts the relative coordinates into absolute latitude, longitude, and altitude coordinates that are stored in the database and used to render a user's position in a Google map.

C. The Mobile Application Layer

A mobile Android application was developed to render the real-time location of people rendered on a Google Map. The mobile application was developed using Android Studio which displays a visual representation of an individual's real-world location. The app employs the Google Maps API with an overlay of a building's architectural floorplan. The application HTTP GETs longitude, latitude, and altitude coordinates through Java Web Services deployed in a Glassfish container on host marconi.sdsu.edu and renders each person's absolute location as an inverted-drop-shaped icon using the Google Maps API. With the architectural floor plan overlay, one can determine which room in a building a person is located. In this work, the fourth-floor floorplan of the San Diego State University College Engineering building was used as an overlay. Each inverted-drop-shaped icon that marks locations in Google Maps represents a different person. Markers also provide information about a particular person in terms of location, most recent measured timestamp of location, and the status of the person. The status determines if the person it is inside or outside a building. This application is published and available in the Google Play Store under the name *Fall Prevention Tracker* (Proximity Networking, 2018), and it is possible for multiple individuals on different Android devices to utilize the application simultaneously. A screen-shot of the application is shown in Figure 5.

Google Maps API authentication and authorization information is specified in the AndroidManifest.xml file in the Android build. The app is configured with a Google Maps API key (Google, 2019)

which is defined as a string resource inside the app code. Each API key is related to an encryption key used to sign the resulting build APK file. A different Google Maps API key is required for each encryption key. The following functions were implemented within the app's MapsActivity file to carry out different features:

- consumeWebService()requests a string response from the URL http://marconi.sdsu.edu:8080/GeoLocation/resources/app, an endpoint of a Java Web Service to query the fixed coordinates of Wi-Fi access points in buildings used in equation 2.
- processJSON() unmarshalls JSON-encoded data returned by Web Services and stores the result in variables such as timestamp, alt, latitude and longitude.
- modifyLocation(string timestamp, string data, double latitude and double longitude) repositions an inverted-drop-shaped icon in the app display to indicate a person's updated location.
- updateMarkers() modifies the location of a person by implementing an HTTP GET request to a Web Service, and then converting the returned JSON data into HTML text for formatting and displaying in an app popup window.
- checkIfPeopleOutside() generates an app notification by changing the color of the marker from green to red if a tracked person is found to reside outside of a building. This function is used to provide immediate assistance if a person leaves a designated area.
- moveCamera(), createOverlay(), addPeopletoApp() are used to assist the above methods in displaying images within the app by enabling focus and by rendering new inverted-drop-shape icons to indicate the presence of new people being tracked.

Class Person stores location and identity data of each person being tracked, and a CustomInfoWindow Adapter is used to customize the content and layout of a window that displays the latest information about each person.

Figure 6 shows a UML class diagram of the mobile Android Application. The *CustomInfoWindowAdapter* class is used to customize the window that renders inverted-drop-shaped icons. The *Person* class gets information about a single person using methods *getName()*, *getTimestamp()*, *getLatLng()* and *getStatus())*. This information is then shown on a Google Map.

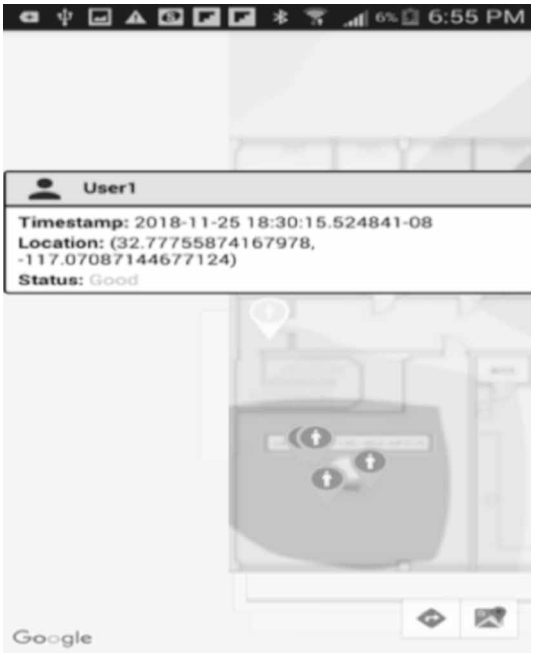
TESTING AND RESULTS

A modular, flexible, portable, end to end, open source framework has been developed, capable of deriving the location of people using existing Wi-Fi infrastructure. To evaluate location accuracy, five Beaglebone embedded ARM Linux single board computers, named (*User0*, *User1*, *User2*, *User3*, and *User4*), were employed, each with a TI WiLink 8 radio. Two 6dB, vertically polarized, high gain antennas were employed on each device to boost the RSSI (Received Signal Strength Indicator) present at the WiLink chip. The solution was powered by a 2000mAh battery and was contained within a 3D printed housing. The devices are shown in Figure 7. Multiple tests were completed to evaluate the Kalman Filter, the execution time of the LAPACK functions, and the effectiveness of the location derivation.

The performance of the Kalman Filter was evaluated by stationing the test platform in a single location and comparing the raw RSSI data with the Kalman filtered RSSI data. The location was the San Diego State University IoT lab with high incidences of multipath interference and obstacle induced path loss. The results of the sampling can be seen in Figure 8.

Three LAPACK functions were utilized for solving the system of equations shown in . DGELS solves an over or underdetermined system using the LU factorization of matrix *A*. DGESV solves an over or underdetermined system through partial pivoting and row interchanges to factorize *A*. The factored form of *A* is then used to solve the system of equations. DGETRS solves a system of equations

Figure 5. Mobile android application using the google maps API



given the LU factorization of *A*, which will be a constant matrix while a person's associated access points do not change. The LU factorization of *A* can be computed a single time, thus enabling DGETRS to be employed to repeatedly solve a system of equations, which is computationally less expensive. The time taken for each of these methods to solve the system was measured: 18000 ns for DGELS, 10000 ns for DGESV, and 6000 ns for DGETRS. As one can see, DGETRS is three times faster than DGELS and, if this system were to be synthesized for hardware or low-level system software, it would be beneficial to utilize DGETRS instead of DGELS or DGESV wherever possible. However, it is worth noting that at least one call to DGESV will be required to derive the factorized *A* matrix.

The system was tested in a relatively noisy university engineering environment with electromagnetic interference and large multipath losses. The fourth floor of the San Diego State University (SDSU) Engineering Building was the chosen environment. The test platform was found to have a scan refresh rate of around 2 Hz. There was also a settle time of 10 seconds for the Kalman filter.

The efficiency of the system was tested based on three parameters: precision, accuracy, and delay. Numerous exercises were conducted with the help of five Beaglebone boards each representing a person. Precision determines the efficiency of the system to compute the distance of the user and showcase it on the map. An experimental setup with various locations and different Wi-Fi coverage areas was created. Table 1 showcases locations of all the access points used for precision testing which are represented as green dots in Figure 9. Tables 2 and 3 represent various test cases where the boards were moved around different locations in the same room. The actual distance and calculated distances were recorded. It was seen that in a good coverage area zone (RSSI -38 dBm) the 3D error was found out to be in the range of 1 to 2.25 meters whereas in a weak Wi-Fi zone (RSSI -63dBm) a 3D error of 3 to 5.5 meters was experienced. Figure 9 shows test cases of weak Wi-Fi coverage as black dots.

Accuracy measures if a person is correctly identified to reside in the room shown on the app. Boards were placed at various locations on the entire floor each having different Wi-Fi strength. Figure 10 is a screenshot of a scenario where all the BeagleBoards were placed at different locations in a single room.

Figure 6. UML class diagram of the mobile application layer

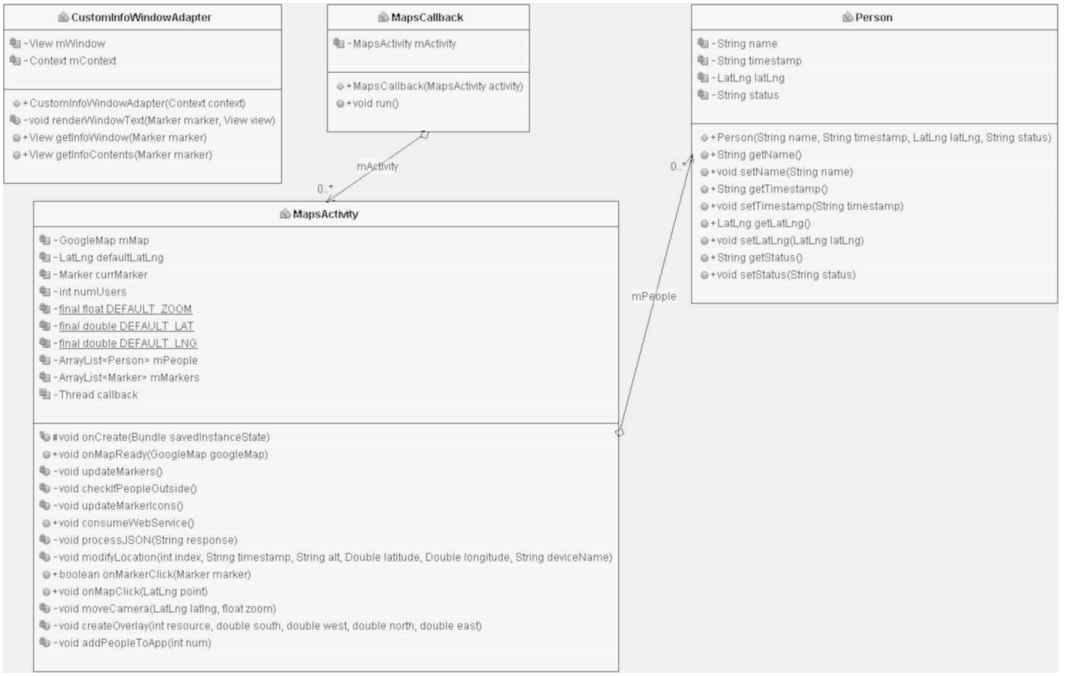


Figure 7. Developed Beaglebone platform (user0, user1, user2, user3, and user4)

Figure 8. Kalman filtered result

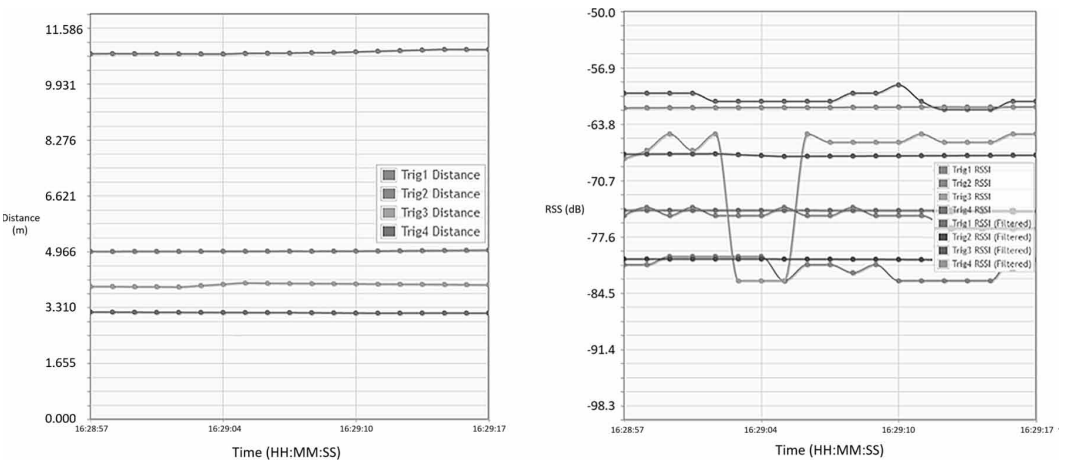


Table 1. Location of access points used for precision test cases

Access Points	Location Coordinates (x, y, z) (meters)
Trig 1	(0.61, 0.56, 1.52)
Trig 2	(6.68, 0.26, 2.49)
Trig 3	(1.9, 4.75, 1.84)
Trig 4	(9.3, 7.35, 2.41)

Another scenario was examined where three boards, *User2*, *User1*, and *User0*, were placed at different locations in the hallway on the fourth floor and two boards, *User3* and *User4*, in a single room. The boards could successfully detect their location, as shown in Figure 11.

However, there is a key limitation of the system and that is an inability to differentiate between signal attenuation due to distance and signal attenuation due to obstacle-induced path loss. This means the system may derive the location of an individual to reside outside the boundaries of a building instead of the actual location, as shown in Figure 12.

Also, another example of this limitation is when the board enters a low Wi-Fi coverage zone. Figure 13 is a map displaying the WLAN coverage area on the fourth floor of the SDSU Engineering building where the test was conducted. As shown, the red color represents the areas where the Wi-Fi coverage is the strongest and the green colored areas are the low Wi-Fi coverage zones. During the test, it was experienced that when a user enters these green zones (RSSI -75dBm) the calculated distances have errors of 2 meters. This is due to not having enough APs in range of the board.

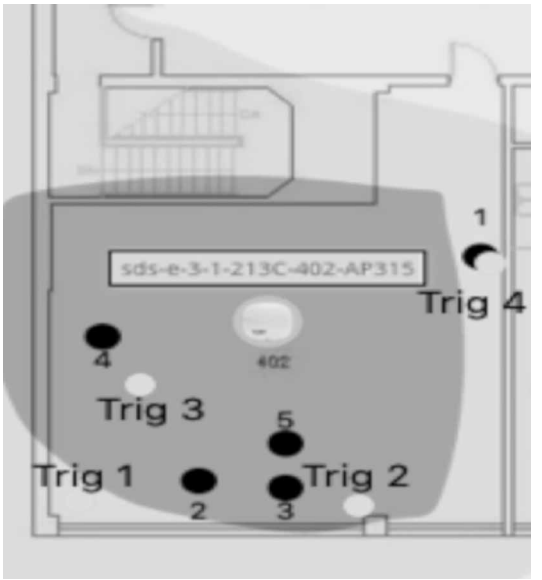
Figure 14 shows an event where *User2* enters a weak Wi-Fi zone of the building (RSSI -75dBm). Hence the system derives the location of the individual to be a few meters away from their actual location as seen on the app.

Delay was calculated as the difference between the time a person walks into a location and the time when the movement is updated on the app. A detailed analysis of situations where people carrying different boards were made to enter low to high Wi-Fi coverage areas was carried out. Results for these experiments can be seen in Tables 4 and 5. The difference in time when a person actually enters a location, and timestamp shown on the app, was computed. It was observed that in a good coverage area, delay calculated was about 3 to 5 seconds. Whereas in a low Wi-Fi coverage area this delay increases to about 12 seconds.

Table 2. Precision values for weak Wi-Fi coverage areas (RSSI: -63 dBm) and less

Actual Coordinates (x_1, y_1, z_1), meters	Calculated Coordinates (x_2,y_2, z_2), meters	3D Euclidian Distance Error, meters	Number
(9.26, 7.35, 0.93)	(7.67, 4.80, 5.40)	5.38	1
(4.22, 1.89, 0.73)	(4.01, 2.75, 2.35)	5.12	2
(5.59, 1.74, 0.75)	(5.94, 2.45, 3.56)	2.91	3
(1.32, 5.74, 1.12)	(3.81, 9.11, 4.71)	5.51	4
(5.59, 2.49, 0.75)	(7.89, 0.76, 3.15)	3.74	5

Figure 9. Weak Wi-Fi test cases



FUTURE WORK

Thus far, the developed system is fully capable of deriving the location of a user, if four or more known access points are within range. However, several avenues exist for future work. The employment of machine learning technologies could help improve the accuracy of the derived location. For example, one use of this technology could include measuring the wireless signal strengths at several known locations within a facility. This information could then be used to generate a set of decision trees which could be employed to more accurately derive a location. As was previously mentioned, the advantage of this software framework is that it can be easily compiled and run on several platforms by simply developing platform specific scan module implementations. Future work will include the implementation of several platform-specific modules for iOS, Android, and Windows.

CONCLUSION

This paper presents the development of a multiplatform, indoor, geolocation framework that uses the standard 802.11 protocol to determine a person's physical location within an indoor facility within a typical accuracy of 3m. The complete solution is entirely new, has not been applied in any form

Table 3. Precision values for good Wi-Fi coverage area (RSSI: -38 dBm)

Actual Co-ordinates (x_1, y_1, z_1), meters	Calculated Coordinates (x_2, y_2, z_2), meters	3D Euclidian Distance Error, meters
(8.57, 3.05, 0.74)	(7.68, 2.45, 0.97)	1.09
(4.22, 3.54, 0.73)	(4.79, 3.06, 2.85)	2.24
(6.69, 2.24, 0.75)	(7.13, 2.53, 2.17)	1.51
(8.01, 6.44, 0.90)	(7.43, 4.82, 1.74)	1.91
(7.41, 2.49, 0.75)	(6.56, 3.98, 0.30)	1.77

elsewhere, and provides a framework for monitoring the real-time location of individuals through a mobile device. The presented software framework provides opportunities for further expansion. The system employs modern C++11 and object-oriented design principles and is portable, expandable, thread safe and memory safe. The performance of the system has been profiled and many areas for further development have been outlined. This project is an open source indoor geolocation framework for tracking the location and movements of individuals. It is hoped that this technology will be employed and used by others, especially in assisted living facilities to help reduce the time it takes for elderly persons in need of care to be quickly located, potentially saving lives.

All source code for the *Embedded Layer* that executes on the BeagleBone Black single board computer, the *Middleware Layer* Netbeans (The Apache Software Foundation, 2019) project that implements RESTful Web Services in a Glassfish Web Application container, and the *Mobile Application Layer* Android Studio project, is publicly available in GitHub (Gala & Paolini, 2019).

ACKNOWLEDGMENT

We thank Professor Harsimran Baweja, Director of the Neuromechanics and Neuroplasticity Laboratory in the School of Exercise and Nutritional Science (ENS) at San Diego State University, for allowing us to use his laboratory facilities and for providing comments and ideas that initiated this research. This work was supported by the University Grants Program (UGP) of San Diego State University.

Figure 10. Example of all BeagleBoards placed in a single room. (The figure was obtained from a screen capture on a Samsung Galaxy S3 mobile phone with a 4.8in frame at a pixel density of 306ppi (pixels per inch). While not clear in this printed image, architectural details and room numbers can be clarified using the zoom (pinch/spread) finger gesture in the app.)

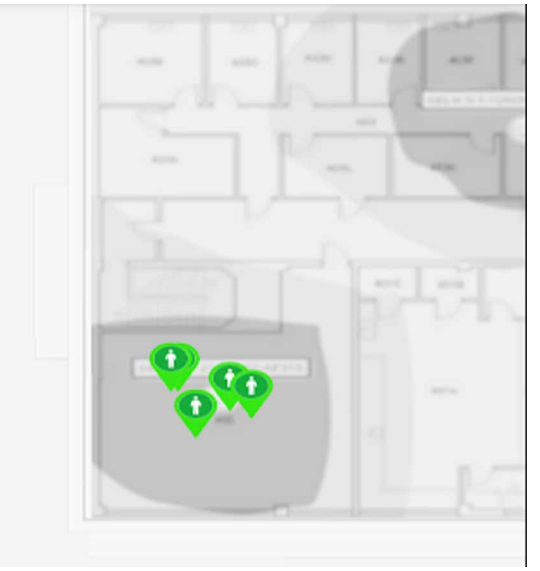


Figure 11. Example of good coverage

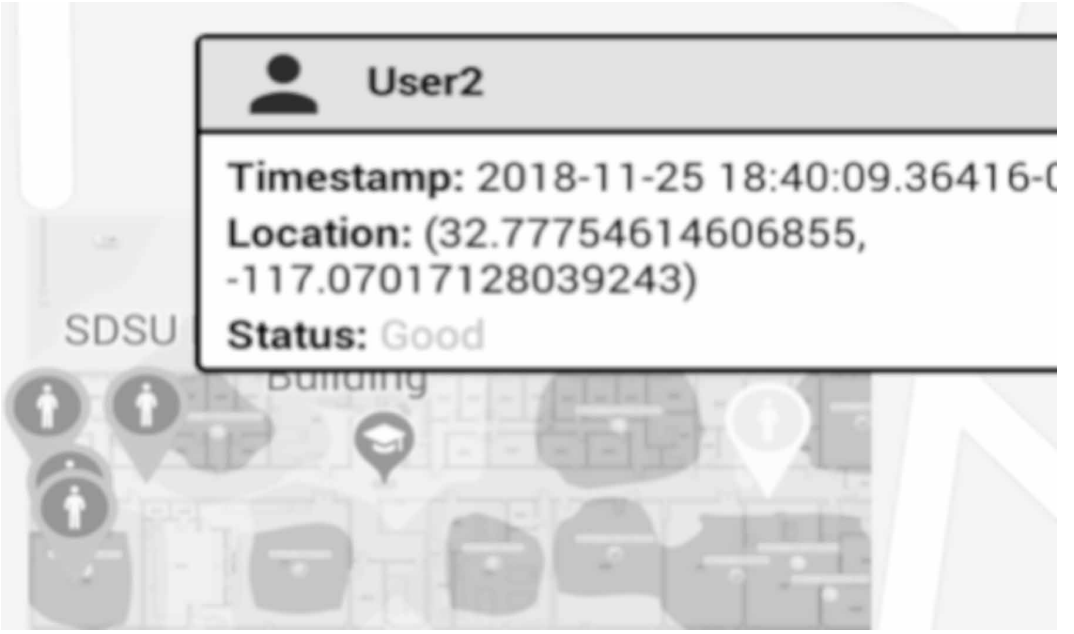


Figure 12. Example of location error due to signal attenuation. (The following figure was obtained from a screen capture on a Samsung Galaxy S3 mobile phone with a 4.8in frame at a pixel density of 306ppi (pixels per inch). While not clear in this printed image, architectural details and room numbers can be clarified using the zoom (pinch/spread) finger gesture in the app.)

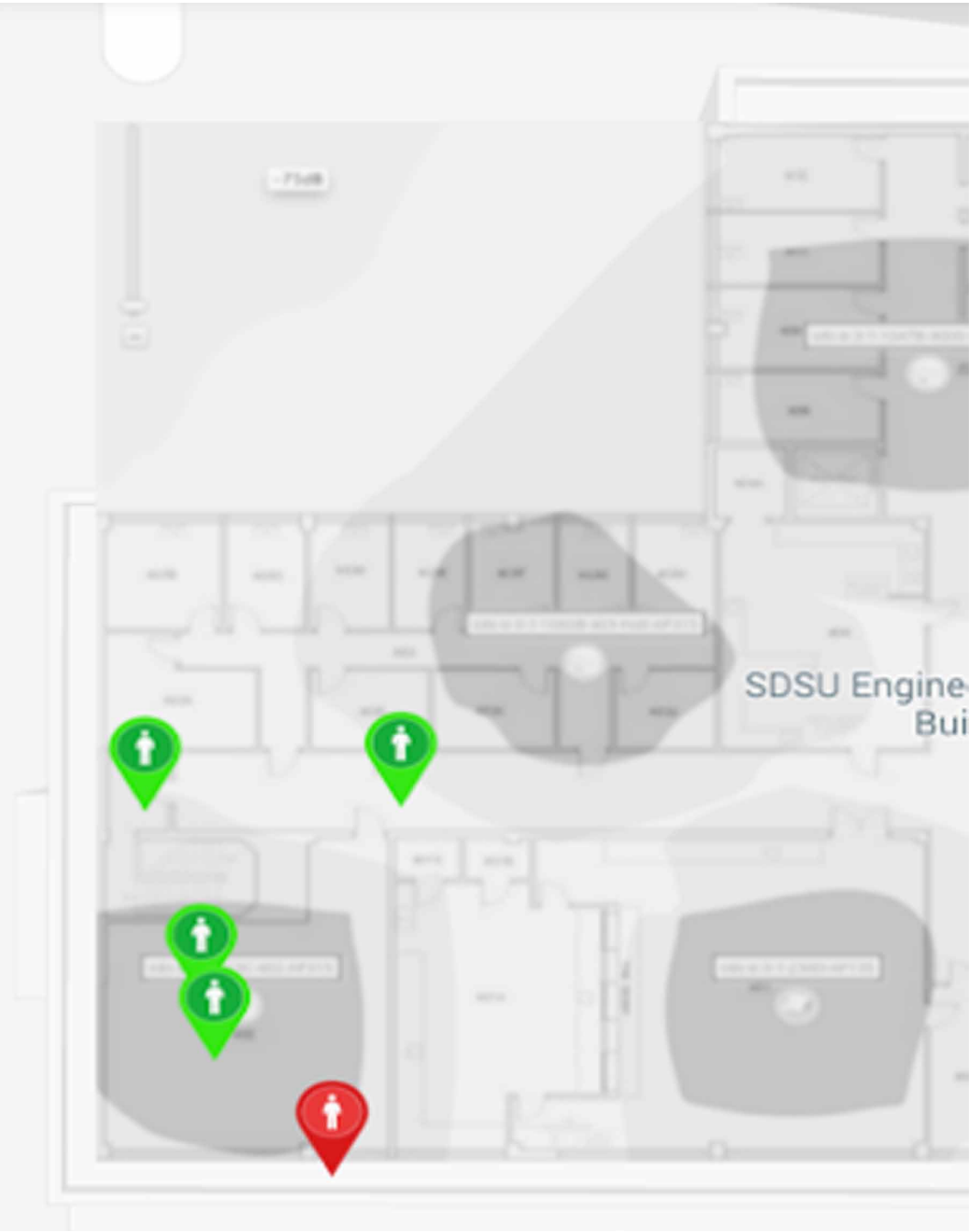


Figure 13. WLAN coverage map for the fourth floor of the SDSU engineering building

Figure 14. Example of a poor Wi-Fi coverage area

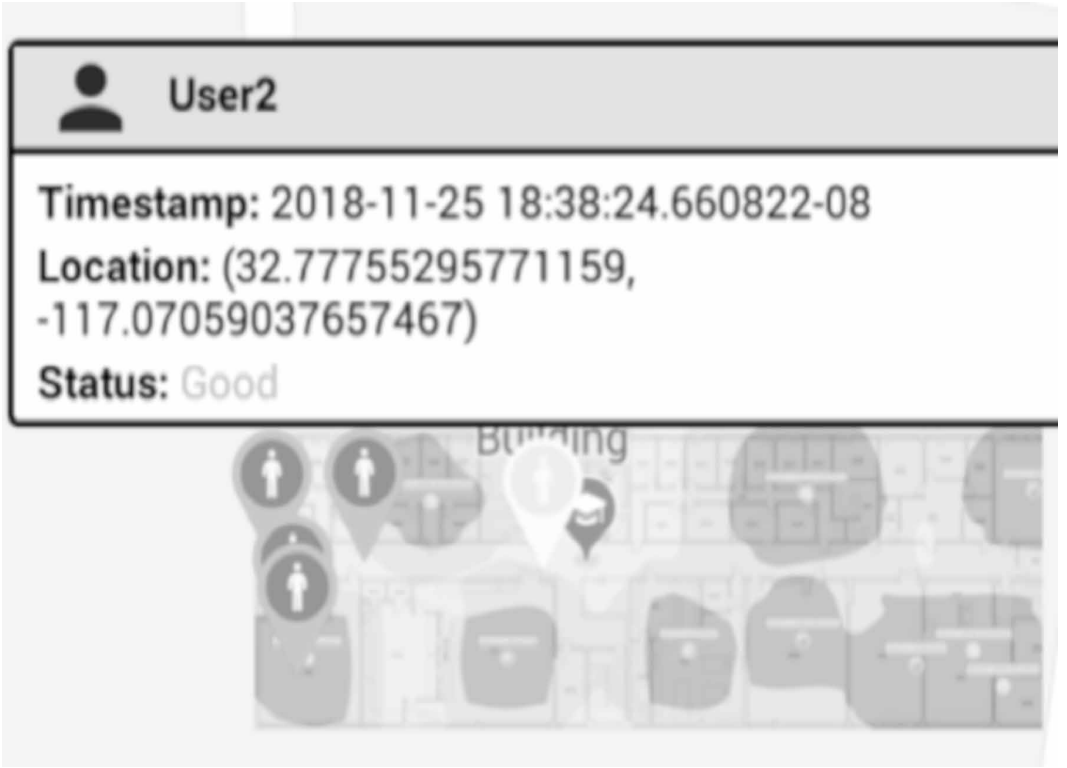


Table 4. Delay values for weak Wi-Fi coverage areas (RSSI: -63 dBm)

Actual Time	Timestamp	Delay (seconds)
15:03:42	15:03:48	6
15:11:13	15:11:15	8
16:22:06	16:22:12	6
16:22:57	16:23:09	12
19:19:49	19:19:55	6

Table 5. Delay values for good Wi-Fi coverage areas (RSSI -38 dBm)

Actual Time	Timestamp	Delay (seconds)
13:04:01	13:04:05	4
13:04:50	13:04:55	5
14:30:41	14:30:45	4
15:19:23	15:19:26	3
15:22:28	15:22:32	4

REFERENCES

Aamodt, K. (2006, July 10). *CC2431 Location Engine*. Retrieved from Álvarez, Y., de Cos, M., Lorenzo, J., & Las-Heras, F. (2010). Novel Received Signal Strength-Based Indoor Location System: Development and Testing. *EURASIP Journal on Wireless Communications and Networking*, *2010*(1), 254345. doi:10.1155/2010/254345

BeagleBoard.org Foundation. (2019). BeagleBone Black. Retrieved from https://beagleboard.org/black

Blake, A. J., Morgan, K., Bendall, M. J., Dallosso, H., Ebrahim, S. B., Arie, T. H., & Bassey, E. J. et al. (1988). Falls by elderly people at home: Prevalence and associated factors. *Age and Ageing*, *17*(6), 365–372. doi:10.1093/ageing/17.6.365 PMID:3266440

Bulten, W., Rossum, A. C. V., & Haselager, W. F. G. (2016, 4-8 April 2016). *Human SLAM, Indoor Localisation of Devices and Users.* Paper presented at the 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI).

Chitte, S. D., & Dasgupta, S. (2008, 26-29 Oct. 2008). *Distance estimation from received signal strength under log-normal shadowing: Bias and variance.* Paper presented at the 2008 9th International Conference on Signal Processing.

Gala, K., & Paolini, C. (Producer). (2019, 4 May 2019). IJITN 2019 Real-Time Indoor Geolocation Tracking Project. Retrieved from https://github.com/kinjalgala/Geolocation

Google. (2019, January 24, 2019). Maps Embed API: Get API Key. Retrieved from https://developers.google.com/maps/documentation/embed/get-api-key

He, J., Geng, Y., & Pahlavan, K. (2014). Toward Accurate Human Tracking: Modeling Time-of-Arrival for Wireless Wearable Sensors in Multipath Environment. *IEEE Sensors Journal*, *14*(11), 3996–4006. doi:10.1109/JSEN.2014.2356857

Kjærgaard, M. B., Blunck, H., Godsk, T., Toftkjær, T., Christensen, D. L., & Grønbæk, K. (2010). *Indoor Positioning Using GPS Revisited*. Berlin, Heidelberg. doi:10.1007/978-3-642-12654-3_3

Lauszus, K. (September 10, 2012). A practical approach to Kalman filter and how to implement it. Retrieved from http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/

Lauszus, K. (2018). longcongduoi/KalmanFilter. Retrieved from https://github.com/longcongduoi/KalmanFilter

Liu, H., Darabi, H., Banerjee, P., & Liu, J. (2007). Survey of Wireless Indoor Positioning Techniques and Systems. *IEEE Transactions on Systems, Man and Cybernetics. Part C, Applications and Reviews*, *37*(6), 1067–1080. doi:10.1109/TSMCC.2007.905750

Pahlavan, K., Xinrong, L., & Makela, J. P. (2002). Indoor geolocation science and technology. *IEEE Communications Magazine*, *40*(2), 112–118. doi:10.1109/35.983917

Priyantha, N. B., Chakraborty, A., & Balakrishnan, H. (2000). *The Cricket location-support system*. Paper presented at the Proceedings of the 6th annual international conference on Mobile computing and networking, Boston, Massachusetts, USA.

Priyantha, N. B., Miu, A. K. L., Balakrishnan, H., & Teller, S. (2001). *The cricket compass for context-aware mobile applications*. Paper presented at the Proceedings of the 7th annual international conference on Mobile computing and networking, Rome, Italy. doi:10.1145/381677.381679

Proximity Networking. (2018). Fall Prevention Tracker Android App. Google Play Store, https://play.google.com/store/apps/details?id=edu.sdsu.geolocation.myapplication: San Diego State University Retrieved from https://play.google.com/store/apps/details?id=edu.sdsu.geolocation.myapplication

Teller, S., Chen, J., & Balakrishnan, H. (2003). Pervasive Pose-Aware Applications and Infrastructure. *IEEE Computer Graphics and Applications*, *23*(4), 14–18. doi:10.1109/MCG.2003.1210859

The Apache Software Foundation. (2019). Apache NetBeans. Retrieved from https://netbeans.org/

Tinetti, M. E. (1987). Factors associated with serious injury during falls by ambulatory nursing home residents. *Journal of the American Geriatrics Society*, *35*(7), 644–648. doi:10.1111/j.1532-5415.1987.tb04341.x PMID:3584769

Wei, W., Yang, X.-L., Shen, P.-Y., & Zhou, B. (2012). Holes Detection in Anisotropic Sensornets: Topological Methods. *International Journal of Distributed Sensor Networks*, *8*(10), 135054. doi:10.1155/2012/135054

WHO Global Report on Falls Prevention in Older Age. (2007). Retrieved from Xu, B., Ge, Y., Chen, J., Chen, Z., & Ling, Y. (2012). Elderly Personal Safety Monitoring in Smart Home Based on Host Space and Travelling Pattern Identification. *Information Technology Journal, 11*, 1063-1069. doi:10.3923/itj.2012.1063.1069

Kinjal Gala is a Master of Electrical Engineering student graduating in August 2019 from San Diego State University. She is a graduate research assistant at Wireless Communications and Networks Lab, San Diego State University with expertise in Computer Networks, Mobile Application, Wireless Communications and Embedded Systems.

Paul David Bryden is a final year Master of Computer & Electronic Systems student graduating in June 2019 from Strathclyde University. He is an internationally experienced researcher and developer with a proven skill set spanning Embedded Systems, Mobile Applications, FPGAs and Wireless Communication Systems.

Christopher Paolini is an Assistant Professor in the Department of Electrical and Computer Engineering at San Diego State University. Chris received a B.S. degree in Computer Science in 1991, M.S. degree in Computer Science in 1998, and his Ph.D. degree in Computational Science in 2007, all from San Diego State University.

Matthew Wang is currently an undergraduate at the University of California, Los Angeles and expects to graduate with a Computer Science B.S. degree in June 2021. He has a passion for developing frontend mobile applications and loves to discuss topics within the field of information technology.

Albena Mihovska, PhD (2008), is an Associate Professor at Aarhus University, Department of Business Development and Technologies, where she is with the CTIF Global Capsule and MBIT research group. Her main activities relate to research in the area of smart dense connectivity and related applications; 5G ultradense access networks, Internet of Things technologies for healthcare and smart grid, and most recently, holographic communications, digital technologies and their impact on business.

Mahasweta Sarkar is an Associate Professor in the Department of Electrical and Computer Engineering at San Diego State University (SDSU). She joined SDSU as a tenure-track faculty member in August 2006. Her research interest lies in the area of wireless data networks. Her research work addresses issues like scheduling, optimal resource allocation, power management, design, analysis and evaluation of network protocols in various wireless applications like embedded biosensors, smart health, brain computer interfaces, cloud computing, IoT, vehicular technologies and real-time social networks. She has published her research work in over 80 peer reviewed journals, technical conference proceedings and book chapters with two best paper awards. She currently serves as the Education Co-Director of the NSF funded Engineering Research Center (ERC) - Center for Neurotechnology (CNT). Dr. Sarkar received her B.S in Computer Science (Summa Cum Laude) from SDSU in 2000 and her Ph.D. from UCSD in Computer Engineering in 2005.