Smartphone Tracking w/ Hololens

Sd_may19-27
Cory Johannes, Jose Lopez, Ryan Quigley
Travis Harbaugh, Ben Holmes, Anthony House

Executive Summary

Our project is intended for localizing workers in large scale projects. These types of projects are generally too large for any one person to monitor all worker activity throughout the day. Our solution was to utilize a smartphone application which takes advantage of smartphone sensors like the accelerometer, gyroscope, bluetooth sensor, and wifi sensor in order to localize individuals. Location data would then be transferred to a server via a websocket, saved in a database, and sent out to a website and Hololens application for visual display.

Requirements Specification

Functional Requirements

Requirement	**Title**
Indoor/Outdoor User Tracking	The software shall track 1 individual walking an unguided trajectory both indoor and outdoors (proposed test site is in and around Durham).
Movement Sensitivity	The software shall detect users moving at basic walking and running speeds.
Store Trajectory Information in a Rethink Database	The software shall send tracking information from the Android device, to the Rethink database through a WiFi connection.
Distance Accuracy	The software shall track the locations of its users to an accuracy of ±1 meter.
Delay Accuracy	A delay time of no more than 5 seconds will exist between the time when data is collected, and the time when data is sent to the database.
HoloLens	Our solution will use the location data stored in our Rethink database for monitoring movement through the HoloLens.

Table 1. Functional Requirements

Use Cases

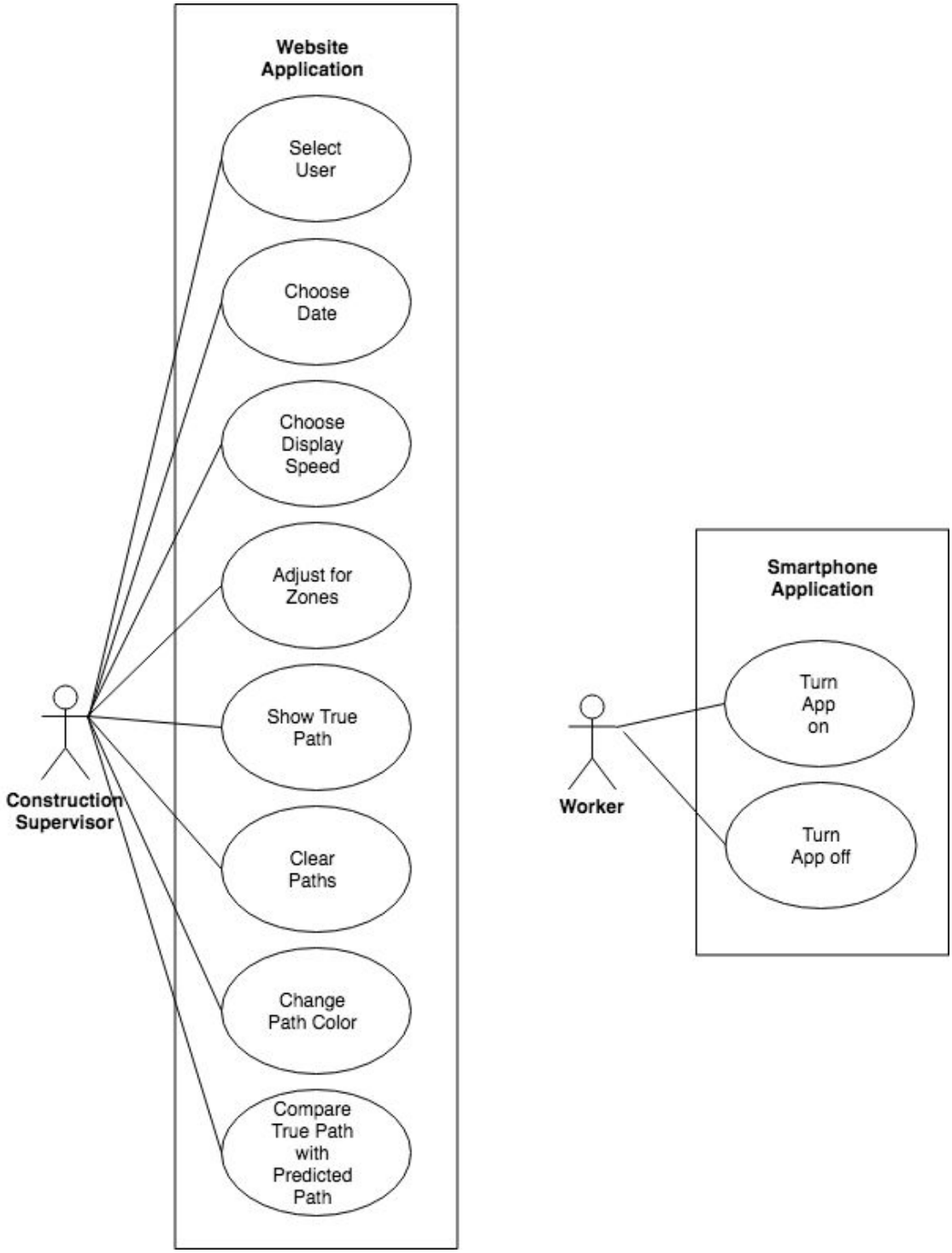


Fig 1. Use Case Diagram

Non-Functional Requirements

Battery Life Cycle	The Android application shall run in a background service, and last for an entire work day.
GPS Sensor	The tracking algorithm shall work without the use of GPS.
Look and Feel	The tracking device shall be comfortable enough to be worn for the entire work day.
Environmental	HoloLens simulation shall be used in indoor conference rooms

Table 2. Non-Functional Requirements

System Design and Development

Conceptual Diagram

Fig 2. Conceptual Diagram

Office Monitoring

From the client's office a supervisor will be sitting in the conference room with the Microsoft HoloLens monitoring the jobsite. The HoloLens renders the construction site, buildings, vehicles, and employee avatars in augmented reality though a connection with the Rethink database. Once the Rethink database receives a location update, Rethink will automatically notify all changes listeners attached to given data, and transmit the information through the corresponding sockets (to the HoloLens). The HoloLens will use an access point from the office to receive this transmission.

Construction Site

Before the construction workers enter the work site, the Android application will calibrate to the user's specific orientation and step size. The phone will be attached to the user's right arm using an armband and must be worn at all times during construction hours. As the construction workers walk about the site, the mobile application will record data from the phone's sensors. The data collected from these sensors will then be used to estimate the user's new position in real-time. Finally, these positions estimations will be sent to the server in order to be saved in the database.

Architectural Diagram

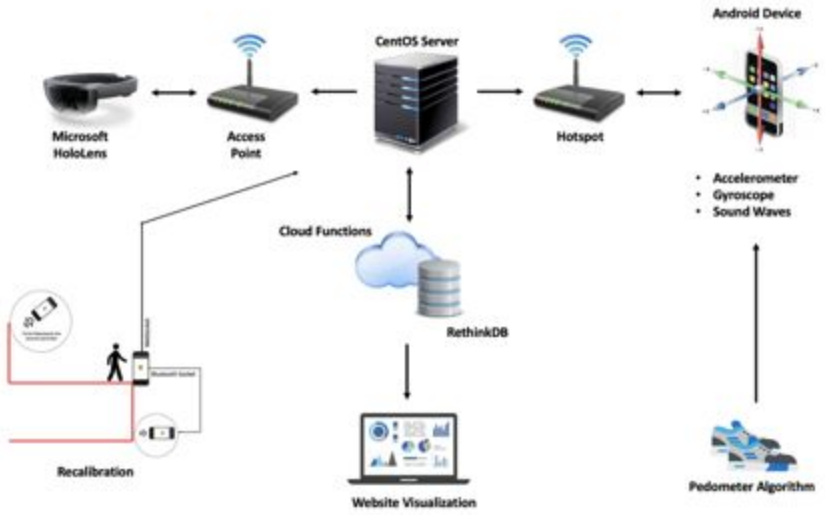


Fig 3. Architectural Diagram

Website

The website receives latitude/longitude coordinate data from the Android phone, saves the data in the database, and updates corresponding user markers representing the devices being tracked. The website will maintain a display which moves these markers through a 2D representation of the worksite (Durham) and trace the paths as they develop.

HoloLens

The Hololens connects to the office access point in order to receive updates from the server. The server ends a notification when new location data is received from the mobile application. The HoloLens then takes this location data from the database and render 3D avatars representing the current positions of all devices being tracked.

Android Application

The Android application detects user movement via the phone's sensors. The application determines if the user has moved from his/her current position, and if so, makes a new position estimation based on both the previous known location, and the collected data. The application then connects to an access point within the construction site (Durham) and sends this new position estimate to the server in order to be saved in the database.

Database

The database we are using is RethinkDB. Our schema will store the user position coordinate data. When the database receives a request from the server to store a new user position or piece of data, the database updates its storage and broadcast the new data to the Microsoft HoloLens and Website.

Previous Work And Literature

The purpose of this section is to present a few of the basic concepts we have considered for this project.

Our project aims to track individuals more accurately than current tracking systems utilizing GPS alone. Our client's last team implemented a Raspberry Pie which used RSSI triangulation (WiFi), and turned that information into latitude and longitude coordinates. Their solution suffered from floating point inaccuracy when converting RSSI based distance estimates to latitude/longitude coordinates. We aim for a solution implementing a Dead Reckoning system with Bluetooth/LE calibration for more accurate results.

Truong Et Al.

The first piece of literature that was read refers to Truong et al, an indoor tracking solution that utilizes insole sensors. In order to track a location this literature proposes using insole sensors which can estimate walking distance by summing up the total number of steps detected, and multiplying by a designated stride length. They used accelerometer and pressure sensors to record each movement. They then transmitted the data to a cellphone which filters out the error and records the distance that the user has walked. The advantage of this approach is more accurate step detection because of the placement of the sensors (in the shoe). The cons of this approach are large error accumulations when distances exceed ~80 meters due to the inconsistencies in stride lengths.

Gallagher Et Al.

The second piece of literature refers to Gallagher et al, a human posture tracking solution utilizing the accelerometer, gyroscope and magnetometer sensors of 3 different phones oriented in a fixed position around a user's waist, equidistant from the bodies center. The basic premise is that since all devices experience acceleration relative to a center of motion, the readings of the 3 different phones can be combined in order to obtain more accurate results. They found the change in posture by integrating the angular velocities read from the gyroscopes, and adding that integration result to a previous posture estimate. This technique has a big advantage over magnetometer based orientation tracking solutions because the gyroscope is not as affected by stagnant electromagnetic radiation in the environment, whereas the presence of a strong magnetic field will greatly disrupt magnetometer readings.

Park, Kwanghyo Et Al.

The third piece of literature refers to Park, Kwanghyo et al, a pedestrian tracking solution that uses a combination of the accelerometer, magnetometer and step detection in order to determine if a user has made a 90 degree turn. The literature focuses on having a map and using the dimensions of a building to determine where a user is based on where they have been. Their solution uses machine learning to determine if a user turns down some specific hallway, or corridor. The disadvantage of this approach is that implementing a machine learning algorithm would be time consuming, and CPU intensive on an Android phone.

A. R. Jiménez Et Al.

The fourth piece of literature I would like to discuss is A. R. Jiménez et al, and their implementation of LE/UWB radio waves for distance estimation and triangulation in a narrow hallway. The literature claims similarly high levels of accuracy in distance estimation for both LE and UWB through a simple Path Loss formula.

RSS is the received power in decibels, RSS0 is a mean RSS value obtained at the reference distance d0=1m, d is the distance between emitter and receiver, p is the path loss exponent, and v is a Gaussian random variable with zero mean and standard deviation σRSS that accounts for the random effect of shadowing (A.R. Jiménez et al).

Our Plan

We used a slightly different approach from anything discussed above. We implemented a Pedestrian Dead Reckoning system combined with Bluetooth/Sound recalibration. We used eight Nexus 7 tablets as our Sound emitters, and connected to the tablets via Bluetooth Sockets. Once our client phone recognized a sound emitted from one of the Nexus 7 tablets, it immediately begins connecting to the next two nearest tablets and starts listening for their sound emissions. By doing this, we can track the user along a path.

Objectives

The objective is to create an Android application capable of monitoring user movement on a construction site. This app will use the various sensors available through the phone in order to predict the user's trajectory. It will then relay this information to a server, where it can be translated and used for a 3D visual representation through the HoloLens. The application should be able to store the tracking data locally on the phone's storage in the event that the device is disconnected from the network, or is unable to upload for a period of time.

Constraints

Our goal is to provide an indoor/outdoor tracking solution for construction workers to wear during a work day. The constraints to our solution must not impede the workers ability to perform on the job site. Please reference table 5 for more details.

Functional Decomposition

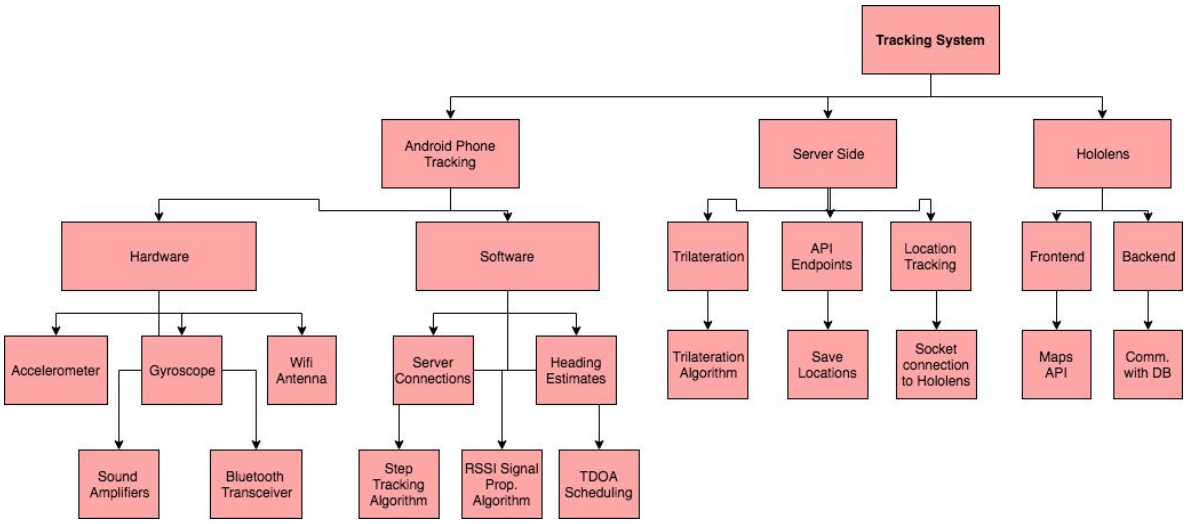
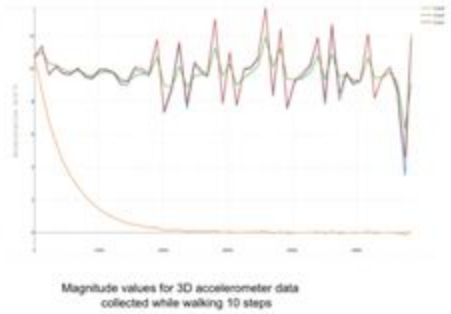


Fig 4. Functional Decomposition

Step Tracking Algorithm



Magnitude values for 3D accelerometer data collected while walking 10 steps

```
Basic Algorithm
max = 11
min = 9
peakDetected = false
currentPeak = -1
stepCount = 0;

for item in data:
    if (item >= max)
        peakDetected = true
        currentPeak = item
    if(peakDetected)
        if(item > currentPeak)
            currentPeak = item
        else if (item <= min)
            stepCount++
            peakDetected = false
            currentPeak = -1
```

Fig 5. Step Algorithm

To determine a user's location, we have to create an algorithm that can determine the distance a user travels based on their stride. We have two thresholds that will determine if the (x, y, z) accelerometer data is a peak or valley. The peak is the maximum value that passed the max threshold. After the peak is

detected and the accelerometer data is decreasing, and it goes below the minimum threshold, then we know that the user has walked a step. We have three vectors. We have a start point, peak, and end point for our algorithm. The distance (foot stride) can be calculated by computing the difference between the start and end point.

Functional Design Diagram

Bluetooth Communication Diagram

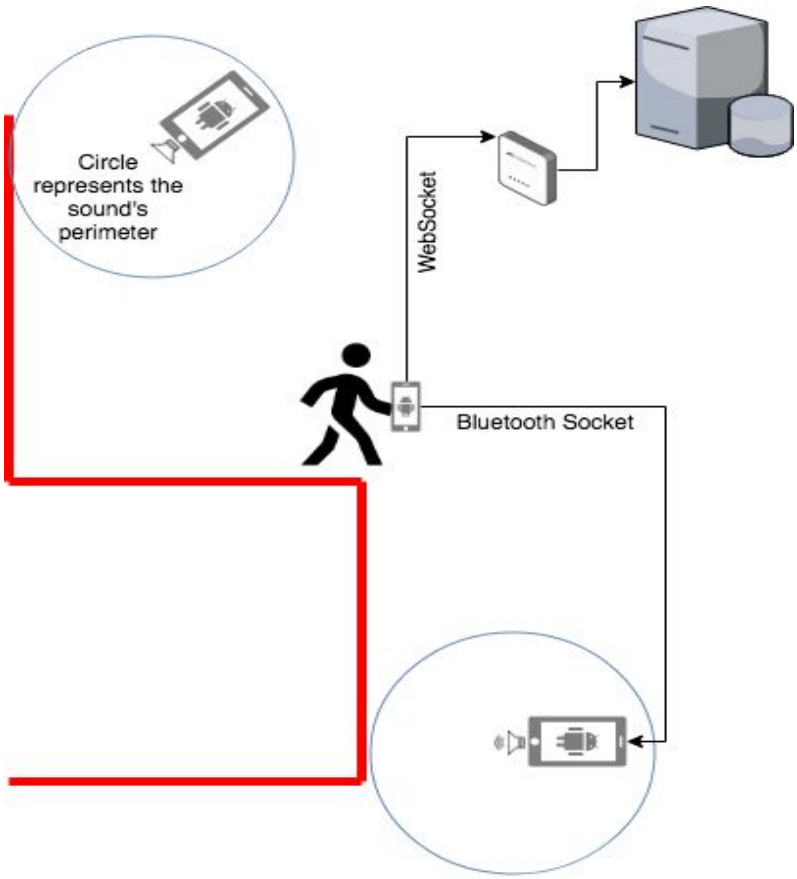


Fig 6. Bluetooth Communication Diagram

Modules

Module	**Description**
Pedometer Module	This module is stored inside the Android app, and contains the step tracking algorithm for predicting steps, as well as a socket connection to send coordinate predictions to the server.
Bluetooth Service Module	Module operates as a service inside the Android app. Contains all code for communicating with Nexus 7 tablets, as well as the code to send and receive sound. This module does not however send any data to the server, all data is sent to the server via the Pedometer Module.
Analytics Module	This module is stored in the cloud, and uses Firebase Analytics to record information about the Android application. This includes information like: ANR's, Crashes, number of downloads, current users, most viewed screens… etc.
Server API Modules	These are the API's our Pedometer module uses to communicate with the server. They include an API for sending coordinate data, as well as an API for trilateration between Nexus 7 tablets. (The trilateration module is currently not being used because the Nexus 7 tablets are not capable of emitting a sound loud enough to be heard over large distances, so instead we use the Tablets as short range beacons)
Hololens Module	This module connects to the RethinkDB with a websocket, and retrieves coordinate information to display on the Hololens.
Website Module	This module pulls data from the RethinkDB, and displays paths on an HTML canvas.
Test Accuracy Module	This module runs on the Website, and includes functions for calculating error between predicted paths, and actual paths.

Table 3. Modules

Interfaces

Interface	**Description**
Website Interface	A visual interface for displaying step data being collected in real time from our Android Application. This interface includes options for retrieving data from previous days, filtering by users, comparing predicted paths to actual paths, path correction, scaling, delay speed, and color adjustments.
Hololens Interface	This interface uses Unity to display a more realistic visual of data being collected in real time.
Firebase Analytics Interface	When our Android App crashes, or receives an ANR, gets a new download, is opened, or updated, all this information is stored in the Firebase Analytics Dashboard.

Table 4. Interfaces

Constraints

The phone shall be helded flat in front of the user, parallel to the ground with the front of the phone always facing in the direction of motion.
The application must be turned on before going inside a building.
There shall be no movement until the application has been initialized and calibrated to the user's current position.
The cellphone shall be an Android device with Accelerometer, Magnetometer, Gyroscope, and Bluetooth sensors.
The Android phone shall use API 21 or above.
A WiFi connection will not always be available.

Table 5. Constraints

Implementation

Implementation Diagrams

Front End

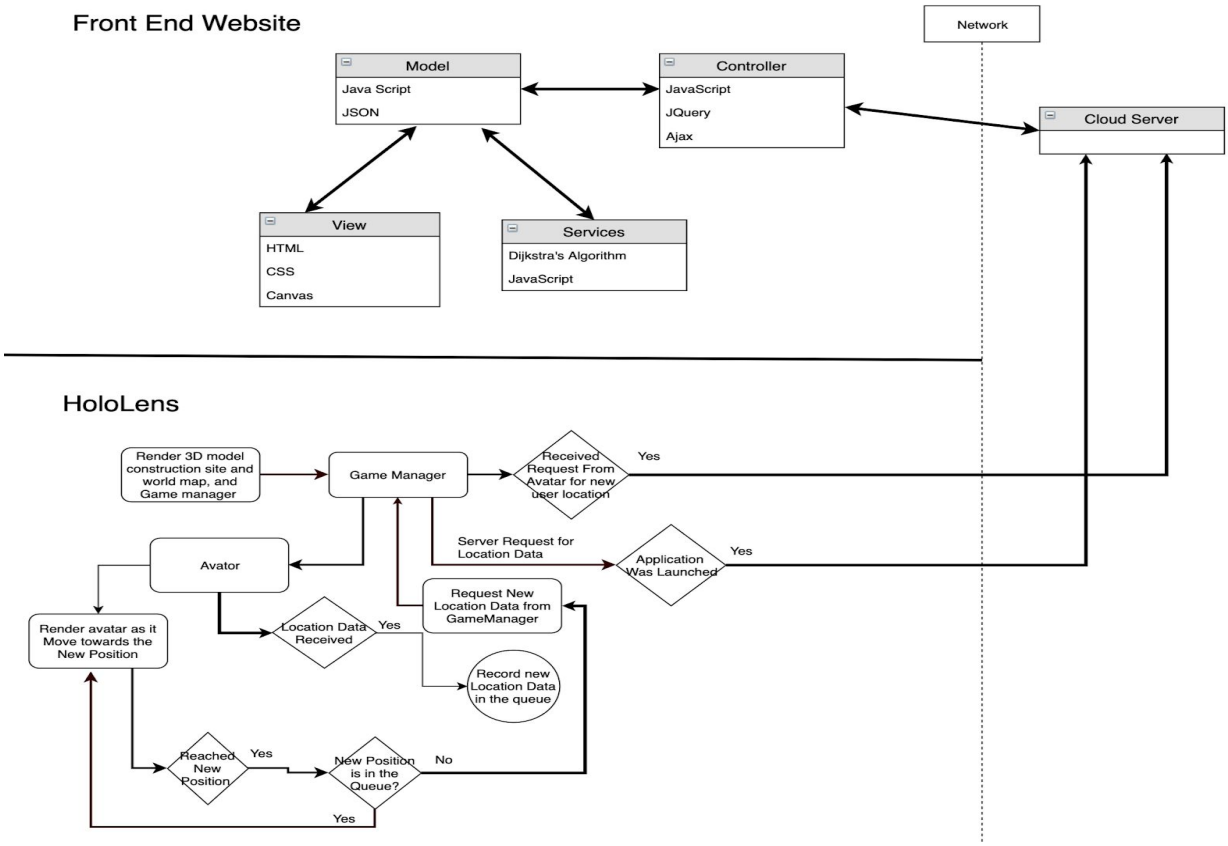


Fig 7. Front End Implementation Diagram

Camera Scripts

The building zone camera script will allow office personal to monitor different zones of the jobsite by using an xbox controller, or hand gestures. The script moves the camera's angle of view based on the input received from the office personal.

Worker Movement Scripts

The player control script will allow the avatar to move from the current position to its new position received from the mobile application and then translate their movement in the HoloLens augmented relativity environment where it can be monitor by the client. When the database script receives a response from rethink database it will send new location information to the Software controller interface Script and determine what avatar needs to be retranslated and

send the movement script the new coordinates. The movement script will translate the current GameObject to be moved from its current position to its new position.

Absolute Path

The absolute path script will render Durham floor path by taking the creating the dimensions from the floor path and translating into 3D world space by producing 3D objects to represent the walls, doors and pillars of the Durham building. It will also render the absolute true test paths with polylines that will allow us to visualize the ground truth of the current user's location.

Animation Script

The animation script will animate (idle state, walking state, running state, etc.) objects in the HoloLens depending on their perceived state. We will create animation clips for the avatar objects. Each state will have a transition to the next animation state. Each avatar instance will have an animation script that is an instance to the animator controller component that controls which state the avatar is currently in. The script will pass in the animation instance variables that controls which state is active. This script will be attached to the avatar game object and the state will be controlled by the data received from the mobile application.

Game Manager Script

The game manager script responsibility to send and receive user Location data from the Rethink Database script and process the user location information and if the user location that was received is currently working at the jobsite the game manger will send that data to avatar game object and the avatar movement script will translate the game object to its new location.

UI Menu Input Scripts

UI menu scripts will receive finger gestures and voice commands. These commands will be used for switching between zones and view perspectives. The UI Menu script will receive input from user wearing the HoloLens. The Software controller interface Script will receive the input and send it to the UI interface controller. The UI interface controller will have an array of UI components that implements the Menu UI interface and will call the interface method. Each component of the UI menu will perform the operation that is requested from the user.

Rethink Database Script

The Database script will be used to send and receive data from the database. If the database sends data to the HoloLens, this script will parse the data received, and send it to the player or vehicle script in order to render the appropriate translation. The Rethink script will send a https request to the database for new user location giving the last time stamp the HoloLens have received and it will send back all the users location data in sorted order.

Back End

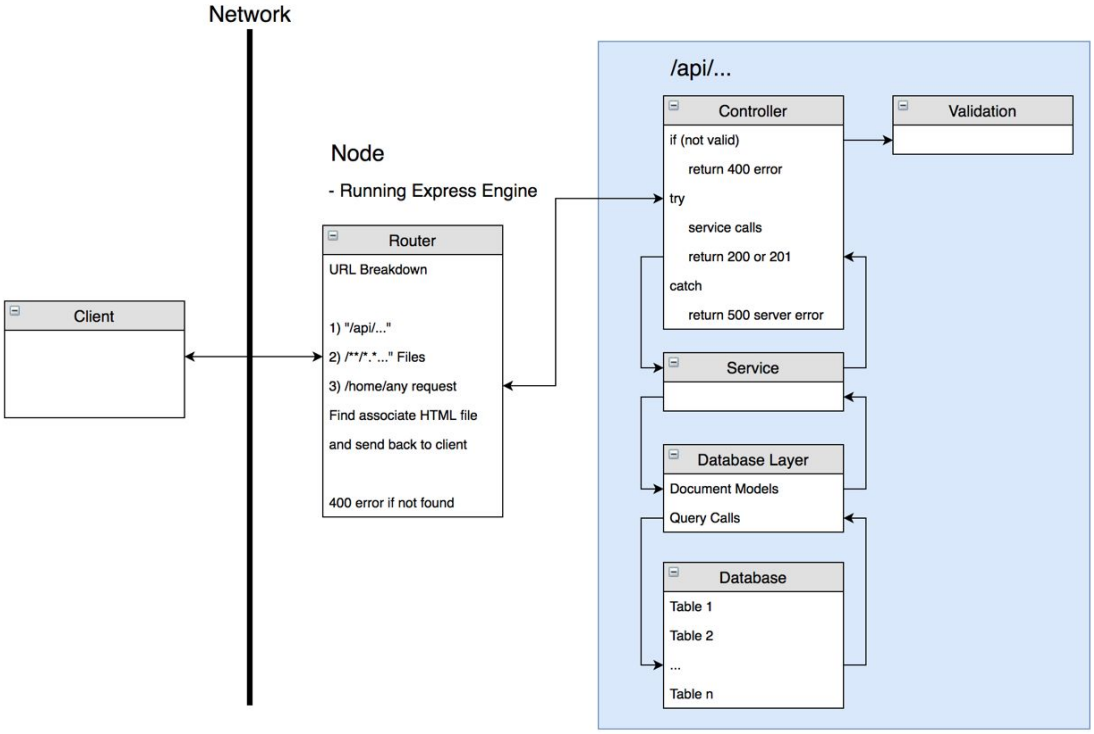


Fig 7. Backend Implementation Diagram

Technologies

- Android/Android Studio 2/Java/Realm DB/Bluetooth/Firebase/Socket.io for Java
    - Used for developing the Android application
- Materialize/HTML/Canvas/Javascript
    - Used for developing the front end website
- Unity3D/C#
    - Used for developing the Hololens Application
- RethinkDB/Socket.io
    - Database used for storing information

Software Used

- Unity3D
    - Used to create the visual Hololens Display
- Android Studio 2
    - This is the defacto IDE for all Android Development, it is supported by Google.
- Firebase Analytics
    - This is another Google product, it allows a quick and easy way for gathering application analytics such as ANR's and crashes.

Standards/Best Practices

Our team reviewed several IEEE standards that would apply to our project:

1.) IEEE NO2571964 - Burst Measurements

The IEEE Technical Committee Report in recommended practices for Burst Measurements in the Time domain is about looking at peak ranges of noise caused by bursts. It shows a study how the results that are recording from reading sensor data can be analyzed. This can be done by recording the magnitude and duration of each burst. This standard is intended to classify the noise of the transmitted data from signals that are sent from hardware. Our team can use this standard because our solution will have to deal with noise surrounding beacons, and mobile sensors. The accuracy of the beacon distance signals from the device can have a lot of error due to RSSI values that are transmitted from the beacon to the mobile device. Based on this article, magnitude can be measured by the under bust and lower bust intervals and a bust duration. You can also view different burst magnitudes of sensors such as instantaneous average and peak which is the highest output value. We are using a similar method for determine our step count and length by using bursts to find the peaks and valleys. We then compare the magnitude of the burst to determine a user stride length. We can apply this IEEE article for the beacon noise as well to determine the peaks and valleys of noise in order to find the best time accuracy to achieve a functional requirement. The standard also talks about the awareness of energy busts that can be applied to our accelerometer data which interferes with signals and create noise.

2.) IEEE NO29148:2011 - Lifecycle Processes

The IEEE International Standard- Systems and software engineering Life Cycle Processes use fir defining requirements that the process of the software undergoes throughout the stages of development. The process starts from a project blastoff meeting that will discover the components, scope and identifying your stakeholders. There are two types of requirements: Functional and nonfunctional that our team will need to consider. Non-Functional are quality requirements that specify the characteristic of your system. Functional are the implementation requirements that serve as a contract to both the developers and stakeholders. Requirements are the language that is used to clarify what the system must meet after the software life cycle is completed. The requirements should have attributes that contain identification information to allow traceability. Requirements are intended to solve the client's business problem. It defines the requirements that your software must meet and any restrictions that your software should satisfy. The System and software life cycle process can apply to our project because we have to meet with our client which is like the project blast off and define the functional and non-functional requirements along with the constraints of our project. We spend several meetings to refine the requirements and make sure that they are accurate and can be traced and tested.

3.) IEEE 1008-1987 - IEEE Standard for Software Unit Testing

The IEEE Standard for Software Unit Testing is to specify a standard approach for unit testing that is used for a based line for evaluating critical software components. The IEEE standard unit testing process has three phases:

- Perform the test planning
- Acquire the test set
- Measure the test unit

The Standard gives you guidance for implementing test procedures against your requirements to ensure that your software meets your client expectation.

The document discusses a general approach on how you can come up with a schedule and what resources your testing team will require to meet the requirements specifications. Once you have developed a schedule and an idea of how to apply unit tests such as what technology is required and how long will it take to write tests, you can define a test set. This is important because you can test for every case. For example, if you have to test all possible combinations of 10 items, you will have 10,1000 combinations of tests that you must deploy. Instead you should define a set that covers the whole domain. With a domain agreed upon, the next step is to execute and evaluate the results to determine how to fix them. The Standard for Software Unit Testing will help our team define coverage for meeting our 1-meter accuracy requirement. This will involve input from many sensors including:

- Magnetometer
- Gyroscope
- Accelerometer

4.) Bluetooth IEEE 802.15.1

Defines physical layer and MAC specification for wireless connectivity.

5.) Wifi IEEE 802.11

Defines protocols for Wifi connections.

Testing/Evaluation

Test Plan

Functional Requirements Test Plan

Functional Requirement:	**Test Plan:**	**Verification:**	**Validation:**
Indoor/outdoor user tracking	We will setup 3 android devices at certain starting locations for tracking. We will provide the user with a specific path to walk which includes a start point and an end point. We will record their path and display it on Mapbox and compare their path with the test path.	Integration Testing: The AR team will design unit tests: An avatar game object instance gets created for each individual set up to be monitored via our application. Compare the avatar path with the test path and report the results The Web team will design Unit tests: Will store the user's path from the start point to their end point and compare their path with the test path and report their results. The backend team will create unit tests: Verifies that the schema tables get the correct user path from the mobile application. The test from all 3 teams will have code walkthroughs and	AR validation: The tester shall verify that there are 3 avatars that are rendered in the 3D worksite. They will verify that their path matches the test path specified by the test plan and it is rendered in the correct location on the map. Web Validation: The tester will verify visually that the users path taken has the same longitude and latitude coordinates as the test plan. Database Validation: It will verify that the table schemas recorded all the location data from the 3 users' path from the mobile device. After the tests are completed, the tester will do a report analysis of the errors that they found and

		reviews to verify that the unit tests are tracking 3 individuals according to their path trajectories.	will be reviewed by the development team.
Movement Sensitivity	We will turn on the mobile application and then have the user perform normal walking and running speeds.	Usability Mobile Test: The user will launch the application and perform a walking movement and running movement and report the results. The mobile team will have a walkthrough to verify how this test is going to meet moving sensitivity requirement.	Tester: The user will turn on the mobile application and will perform running and walking movement and look to see if the mobile device detects their walking and running speeds. After the tests are completed, the tester will do a report analysis of the errors that they found and will be reviewed by the development team.

Store tracking to Rethink DB	We will connect the mobile device to a Wi-Fi and will then perform a step and make a http post request to the database.	Mobile Unit Test: Verifies that the mobile application does an HTTP request when the device is connected to a WI-FI access point. Backend unit Test: Verifies that the data received from mobile application matches with the recorded data from the mobile application. The backend and mobile team will have code walkthroughs and reviews to verify that they the unit tests are fulfilling sending data through an access point according to the specifications of the storing tracking information requirement.	Mobile Tester: Verifies that the application does an HTTP request to the server and the connection was successful. Backend Tester: The tester will verify that the data recorded on the phone matches with the data stored on the database table scheme. After the tests are completed the tester will do a report analysis of the errors that they found and will be reviewed by the development team.
Distance Accuracy	We will give a user a new starting location to start the application at. We will instruct him/her to move 1 meter and see if the new current location is within meter.	Mobile Unit Test: Create a unit test that starts at Location A and moves to location B and then verifies the user new location is within our 1-meter accuracy requirement. The mobile test team will perform a code walkthrough that explains how their test meets the specification of the distance accuracy requirement. The review team will check documents and files to	Mobile Tester: Verifies that the movement from Location A to Location B is within 1-meter accuracy. After the tests are completed the tester will do a report analysis of the errors that they found and will be reviewed by the development team

		ensure that the code meets our coding standards.	
Delay Accuracy	We will use a timestamp when the mobile application has gathered all the sensor data and starts to perform the smoothing/prediction algorithm while the software creates a 5 second delay. After the delay is completed the software will create another time stamp and compute the time difference.	Mobile Unit Test: Create a unit test that takes the time stamp after the data has been collected and after the x second delay has been completed. Then compute the time difference and compare it with delay requirement. The mobile test team will perform a code walkthrough that explains how their test meets the specification of the delay accuracy requirement. The review team will look over the documents and files and confirm it meets our coding standards.	Mobile Tester: Verifies that the delay time of the application is within the 5 second threshold by running the test script and report the results. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents.
Drift Accuracy	We will start a user at a starting point by Durham. We will then have the user move 2 meters in any direction and record the position and then do a computation of the possible positions and see if the user is within the 1 meter of the computed location.	Mobile Unit Test: Create a unit test that takes the starting position of the test plan and then computes all the possible new position paths the user could had taken and then compare those position with the user's current position and report the results. The mobile test team will perform a code walkthrough that explains how their new	Mobile Testers: Verifies the users new position is within 1-meter of the possible computed positions that was computed. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents.

		positions will meets the specification for the drift accuracy requirement. The reviewers will look at the code to see if it meets our coding standards.	
HoloLens Monitoring	The HoloLens will receive the new recorded location data from the Rethink database. It will then render the users new position in the augmented reality map. The user that is wearing the device should see the avatar move from the user's current position to the new position that was received.	AR Unit Test: Create a unit test that takes the newly recorded data from RethinkDB and calculate the new position that the avatar transform should read after unity renders the new position. Compare the avatar's new position with the computed position and report the results. The AR team will perform a code walkthrough that explains how their new computed transform position meets the specification of the HoloLens monitoring requirement. The reviewers will look at the code and see if it meets the unity coding standards.	AR Testers: Verifies the new transform position the avatar moves to in the AR map is the same as the newly computed position. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents and address the software errors that were reported.

Monitor Accuracy	We will create a timestamp and send a location packet by using a http request to the database. When the database has received the packet, it will create another timestamp. The database will then send the location packet to the website and HoloLens software. The HoloLens and website will then create a new timestamp and calculate the difference and see if it is within 10 second threshold.	AR Unit Test: Create a unit test that take the recorded timestamp from the mobile application and create a new timestamp. Then compute the difference and compare it with the 10 second requirement and report the results. Web Unit Test: Create a unit test that takes the recoded timestamp from the mobile application and creates a new timestamp. Then compute the difference and compare it with the 10 second requirement and reports the results. Backend Unit Test: Create a unit test that takes the recoded timestamp from the mobile application and create a new timestamp. Then compute the difference and compare it with the 10 second requirement and reports the results. The AR, Web, and Backend will perform a code walkthrough that explains how their time calculation meets the monitor accuracy requirement. The reviews will look at the code and see if it meets the coding standards.	AR Testers: Verifies the that recorded location is being displayed on the map in AR within the 10 second threshold from the time it was recorded on the mobile device. Web Testers: Verifies the that recorded location is being displayed on the website map within the 10 second threshold from the time it was recorded on the mobile device. Backend Testers: Verifies the that recorded location is being displayed on the website map within the 10 second threshold from the time it was recorded on the mobile device. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents and address the software errors that were reported.

Bluetooth Sensor	The android device will be paired with the beacon so that they our connected to each other. We will set the beacon at a location x inside the jobsite. When the android device receives a signal from the beacon, it will send a location update to the android device.	Mobile Unit Test: Create a unit test that takes the updated the location from the beacon and then compares it with the current location that is stored on the device. It should compare these two and report the result. The Mobile team will perform a code walkthrough that explains how their comparisons meet the Bluetooth sensor requirement. The reviews will look at the code to make sure that the developers are meeting the coding standards.	Mobile Tests: The user will stand within 1 meter of the beacon to trigger the location update. They will then verify that the mobile devices current location is getting updated. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents and address the software errors that were reported.
Android Low Battery Notification	The android device battery must be drained to 10% of its total capacity. Then we will run the application on the android device.	Usability Mobile Test: Let the device get below 10 percent of battery and display a low battery notification. The Mobile team will perform a code walkthrough that explains how their notification meets the Android low battery notification requirement. The reviews will look at the code to make sure that the developers are meeting the coding standards.	Mobile Tester: Tester should see if the low battery notification displays when the devices battery is below 10 percent. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents and address the software errors that were reported

HoloLens Low Battery Notification	The HoloLens battery must be drained to 10% of its total capacity and then run the application on the device.	Usability AR Test: Let the device get below 10 percent of battery and display a low battery notification. The AR team will perform a code walkthrough that explains how their notification meets the HoloLens low battery notification requirement. The reviews will look at the code to make sure that the developers are meeting the coding standards.	AR Tester: Tester should see if the low battery notification displays when the devices battery is below 10 percent. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents and address the software errors that were reported

Table 6. Function Requirements Test Plan

2.13.1 Non-Functional Test Plan

Non-Functional Requirements	**Test Plan**	**Verification**	**Validation**
Battery Life Cycle	A user will turn on the mobile device and launch the application as a background process to monitor the battery and CPU usage throughout an 8-hour work day.	Usability Mobile test: The user will launch the app and monitor the mobile application every hour to report the results. The Mobile team will explain why the monitoring test meets the Battery Life Cycle requirement.	Mobile tester: The user will launch the mobile application and monitor the Android profiler every hour for 8 hours. After the test is completed, the tester will complete a report analysis of the CPU usage. The development team will review these documents and address the software battery issues, if any.

GPS Sensor	We will turn off the GPS location service on the mobile device and create a pre-planned walking path around Durham. We will record the path walked, compare it with the preplanned walking path and show the differences.	Mobile Unit Test Create a unit test which takes the pre-planned route, compares it with the user's route and reports the results. The mobile team will perform a code walkthrough that explains how their unit test meets the GPS sensor requirement. The reviewers will look at the code to ensure the developers are meeting the coding standards.	Unit test where you create a walking path, perform the walk, compare the expected path with the actual path and return the result
Look and Feel	We will place the mobile device inside of the armband and wear the device for 8 hours.	Usability Test: We will create a test that requires the tester to wear the mobile device for a normal work day and report the results.	Mobile Tester: The tester will wear the device around his/her arm for 8 hours and then report the comfortability to the development team.
Environment	We will launch the HoloLens application at the clients facility inside of their conference room.	Usability Test: We will create a test that requires the tester to wear the HoloLens inside of the clients facility.	AR Tester The tester will go to the client's facility and see if their facility has the proper environment for running the HoloLens application. The tester will complete a report analysis of their findings to the HoloLens development team.

Table 7. Non-Functional Requirements Test Plan

Interface Testing

- Website Interface
    - We tested the website with test coordinates following specific paths such as:
        - Straight line
        - Sharp Turns
        - T-Shaped
        - U-Shaped
    - We then compared the display on the website with the display the coordinates should have actually made. If they were the same, it was displaying correctly.
    - We also used test coordinates that included out of bound areas to verify that the Canvas would not display points in these areas.
    - We tested collecting data for different users to verify the websites display correctly filtered by selected user.
    - We tested over multiple days to verify that previous days data was also being stored for selected users.
    - We practiced walking paths in real time, and verified the website updated correctly.
- Firebase Analytics Interface
    - No testing is needed for this, this is a Google software, we did not create this.
- Hololens Interface
    - Testing was done in a similar manner to the Website Interface, only instead of testing for multiple users and filtering by user and day, we only tested on real time data.

System Integration Testing

Overview

Our system started with the Android application. We knew it was important to get this at least started and ready before integrating any other systems because all other systems rely on the data from the Android application. Once the application implemented the Pedometer algorithm, we began incorporating the server and frontend simultaneously. First we created a server api for sending step data from the Android app to the server to be saved in the RethinkDB. Once this was completed, we set up an Ajax call in the frontend website for retrieving data from the server to be displayed on Canvas.

Once we had all three parts essentially communicating, we were able to really begin developing our individual components. The Bluetooth service was added to the Android application for recalibration, A canvas visual display was added to the front end which continued to develop overtime, and websockets were implemented on the Server for more reliable communication.

Testing

To test our integrated system, we outlined a series of paths in Durham. We then walked these paths, and compared the data we received from the Android application with what the data should have looked like. At one point steps were getting dropped from the data due to wifi connections. This was fixed by implementing RealmDB on the Android application for saving all data before sending over the websocket. At another point we realized the heading predictions would freeze and stop updating. This was because electromagnetic fields in Durham were affecting our heading readings. We fixed this by switching from using the Android compass, to using the Gyroscope.

We essentially continued like this, walking paths, comparing our predicted paths with the actual paths, and updating different components of our system in order to get the best results.

User-Level Testing

All of our user-level testing was attached to the frontend website. We would test in a number of ways:

1. Walking paths, and verifying data was displayed in real time
2. Selecting a time delay from settings, and again walking paths, and verifying data was displayed with selected delay
3. Walking paths as different users, and verifying the website correctly filtered step data by user. (I.e. if you have personA selected in real time, you should only see real time step data from personA, and similarly, if you select a date, you should only see data on that date related to the selected person)
4. Walking paths on multiple days as multiple users, and verifying the website was able to filter out the collected data when specific users and dates were selected
5. Walking predetermined paths, and overlaying the correct path on top of the predicted path in order to determine accuracy
6. Modifying colors of different paths
7. Clearing the screen, and restarting
8. Adjusting the X-scale and Y-scale
9. Testing using a zone-adjust algorithm to keep points from being displayed in restricted areas
10. Testing without using a zone-adjust algorithm
11. Verifying that with zone-adjust on, data is still displayed in either real time, or with the appropriately selected delay
12. Walking paths which cause a user to loose their network connection, then walking back into a connected area, and verifying steps were not lost, and are also sent in correct order
13. Walking past re-calibration points, and verifying that they are recognized and pulling the users into the correct areas without disrupting the step order

Validation/Verification

Our validation technique was to walk predetermined paths, and compare the predicted paths of our application, with the actual predetermined paths. Generally there would be some error here, so we would sum up the distances between the predicted points and predetermined points, then average them in order to get our error. If our error was within ~1m, then our system was working correctly. Otherwise, we needed to change parts of our system in order to make it more accurate.

Evaluation

We determined that using a zone-adjust algorithm when collecting data, our predicted path accuracy was generally within ~1m, which was our functional requirement. We tested four different predetermined

paths, and compared the results. The figure below shows the four predetermined paths, from left to right in Fig 8 (below) are: straight path, turns path, and loop path. Fig 9 (below) shows the final path, the long path.

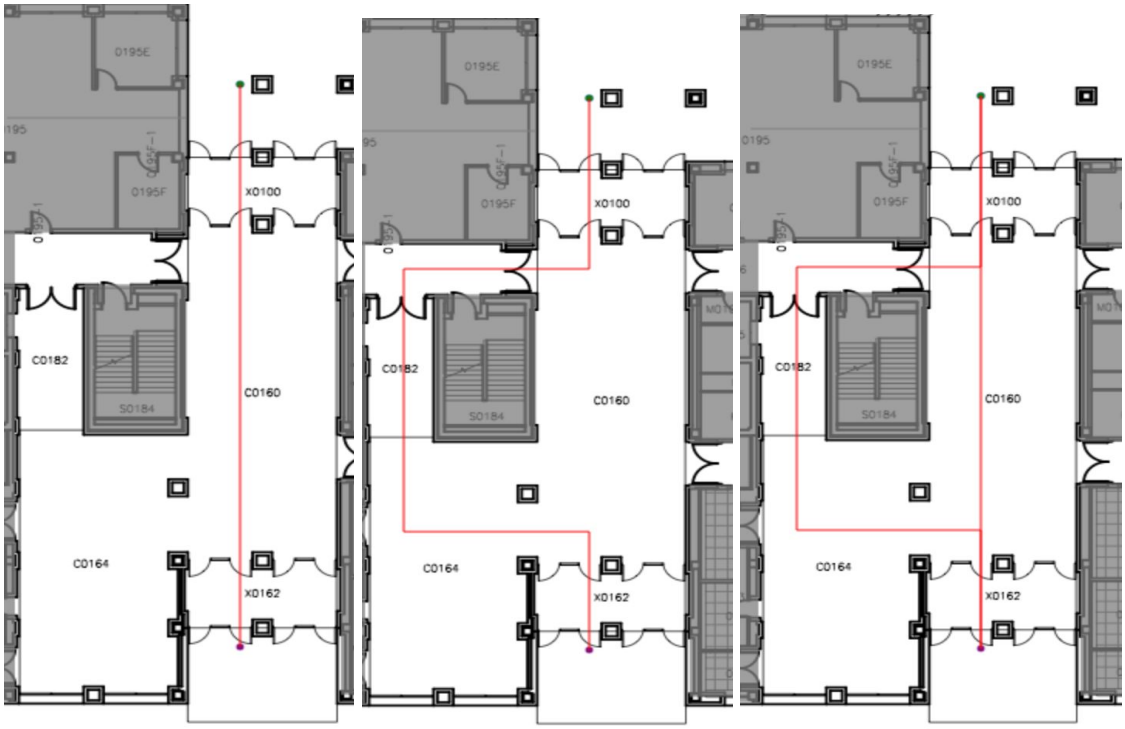


Fig 8. Test Path Figure

Fig 9. Long Path

Test Path Accuracies:

1. Straight Path:
    a. 0.58 meters
2. Turn Path:
    a. 0.462 meters
3. Loop Path:
    a. 1.108 meters
4. Long Path
    a. 1.21 meters

Project and Risk Management

Roles & Responsibilities

Cory Johannes - Frontend/Research
Jose Lopez - Frontend
Ryan Quigley - Android/Step Tracking
Travis Harbaugh - Hololens/Android
Ben Holmes - Android/Bluetooth/Sound
Anthony House - Frontend/Server/Backend

Project Schedule

Proposed Spring Schedule

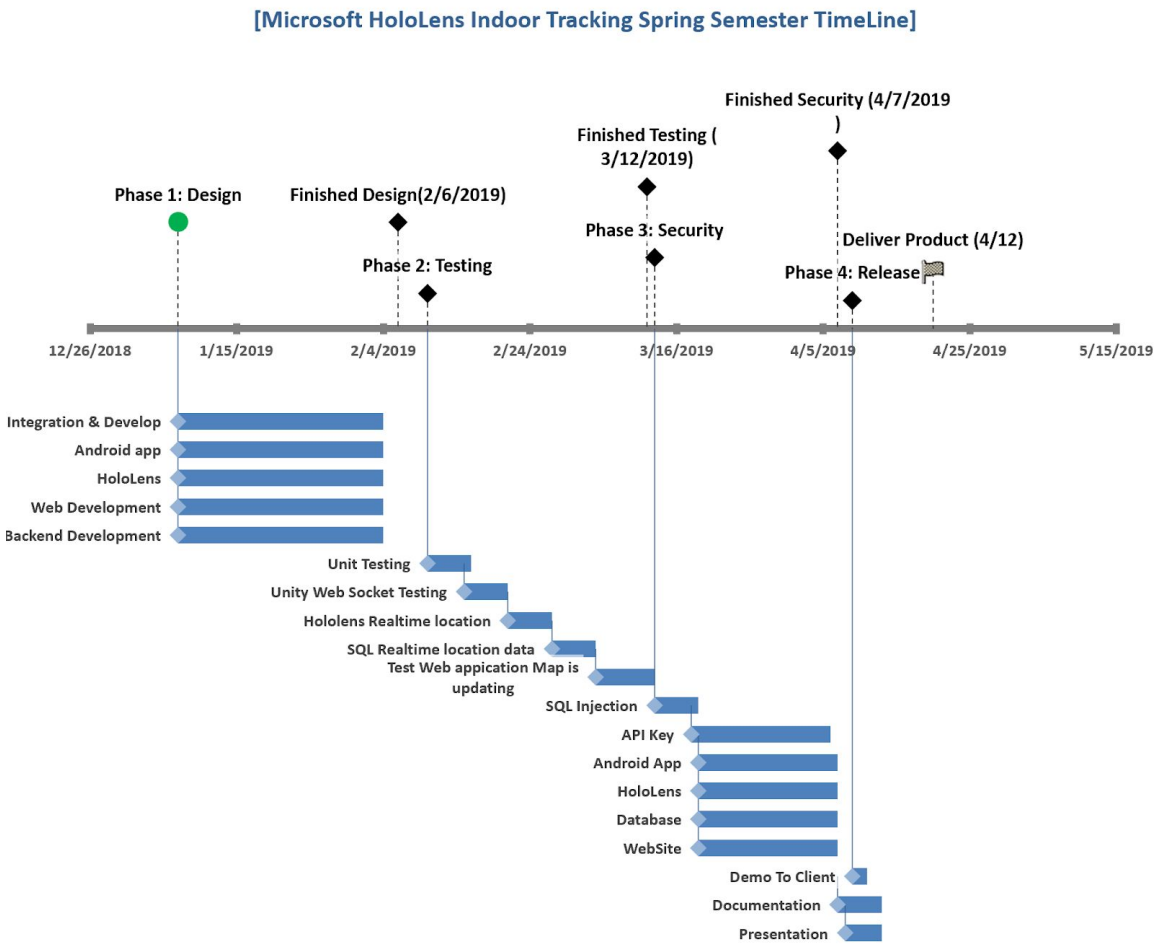
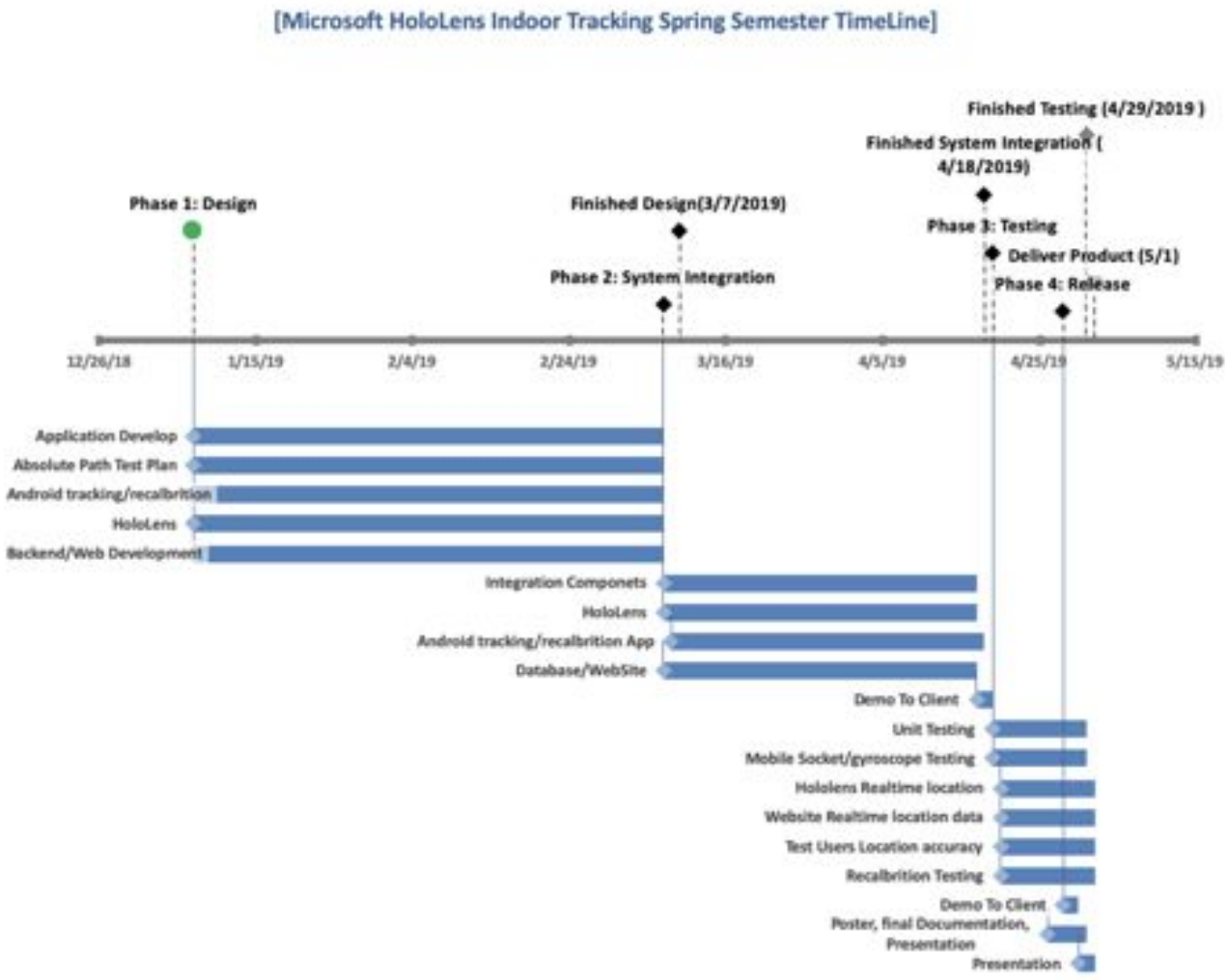


Fig 10. Spring 2019 Gantt Chart

Actual Spring Schedule



1/20/19-2/3/19: Test Plan, Hololens visual display, Backend development, Android App
2/4/19-2/10/19: Test Plan, Hololens visual display, Backend development, Android App
2/11 - 2/17: Hololens, Android application, Backend development, GPS initial position, Test plan
2/18 - 2/24: Hololens, Android application, Restricted zoning, Redeveloped Test Plan, backend, testing
2/25 - 3/3: Hololens, Android, backend development, Android application, testing
3/4 - 3/10: Configuring Bluetooth infrastructure, websockets for android app, database, front end update, updated restricted zone, testing, live demo
3/11 - 3/24: Web development, Android App, Website, testing
3/25 - 4/6: Hololens, Web development, Android development, testing
4/7 - x: Documentation, demo to client, presentation

Risks and Mitigation

Few risks were foreseen in the project, and none occurred. We considered that due to the need for this system to have many phones in a construction area for several hours each day, there is a risk to the devices being dropped or damaged. This threat can be minimized by using padded and secured phones. There is also the risk our client is taking when purchasing supplies for our project. This can be lessened by avoiding expensive purchases whenever possible, either finding cheaper alternatives or using equipment we already own.

Lessons Learned

We learned that while sound recalibration is possible with Android devices, it will be inaccurate in particularly loud areas like construction sites without the use of extremely powerful speakers and ultrasonic waves. We chose to use sound over RSSI trilateration because of the wild floating point inaccuracies associated with converting RSSI trilateration values to the lat and lon coordinates. Overall it seems that in order to make this a commercially applicable system, expensive phones with Radio/UWB antennas would be needed over Bluetooth/Wifi antennas. Despite the problems associated with the recalibration methods, our Pedestrian Dead Reckoning system worked surprisingly well, and was unhindered by external forces such as noise, or electromagnetic radiation.

Conclusions

Closing Remarks

One important problem with this type of project is the level of hardware being used. With typical Android phones, the hardware is not powerful enough to do things like time dilation of arrival calculations, or to send sound frequencies long distances. Because of this, we relied on close range soundexchange, and were not able to use trilateration. With the correct hardware, this type of project is perfectly manageable, but given our functional requirement of localizing without the use of a Wifi connection, or any GPS/Network provider, it resulted in us needing a relatively large amount of infrastructure for recalibration.

Future Work

Our system could be used on large scale sites in the future, assuming that phones with UWB/Radio antennas, or extremely powerful speakers capable of sending frequencies in the range of 20,000hz -

40,000hz are used. If UWB/Radio antennas are used, more accurate trilateration is also possible over long distances.

References

1. A. Jimenez and F. Seco, “Finding objects using UWB or BLE localization technology: A museum-like use case,” 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2017.
2. L.-H. Chen, E. H.-K. Wu, M.-H. Jin, and G.-H. Chen, “Intelligent Fusion of Wi-Fi and Inertial Sensor-Based Positioning Systems for Indoor Pedestrian Navigation,” IEEE Sensors Journal, vol. 14, no. 11, pp. 4034–4042, June 2014.
3. Z. Chen, H. Zou, H. Jiang, Q. Zhu, Y. Soh, and L. Xie, “Fusion of WiFi, Smartphone Sensors and Landmarks Using the Kalman Filter for Indoor Localization,” Sensors, vol. 15, no. 1, pp. 715–732, Jan. 2015.
4. A. Gallagher, Y. Matsuoka, and W.-T. Ang, “An efficient real-time human posture tracking algorithm using low-cost inertial and magnetic sensors,” 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Feb. 2005.
5. Y. Jin, H.-S. Toh, W.-S. Soh, and W.-C. Wong, “A robust dead-reckoning pedestrian tracking system with low cost sensors,” 2011 IEEE International Conference on Pervasive Computing and Communications (PerCom), May 2011.
6. M. Oner, J. A. Pulcifer-Stump, P. Seeling, and T. Kaya, “Towards the run and walk activity classification through step detection - An android application,” 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2012.
7. K. Park, H. Shin, and H. Cha, “Smartphone-based pedestrian tracking in indoor corridor environments,” Personal and Ubiquitous Computing, vol. 17, no. 2, pp. 359–370, Feb. 2013.
8. P. Truong, J. Lee, A.-R. Kwon, and G.-M. Jeong, “Stride Counting in Human Walking and Walking Distance Estimation Using Insole Sensors,” Sensors, vol. 16, no. 6, p. 823, June 2016.
9. IEEE Technical Committee Report on Recommended Practices for Burst Measurements in the Time Domain," in IEEE No 257-1964 , vol., no., pp.1-12, 15 May 1964
10. ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes --Requirements engineering," in ISO/IEC/IEEE 29148:2011(E) , vol., no., pp.1-94, 1 Dec. 2011
11. "IEEE Standard for Software Unit Testing," in ANSI/IEEE Std 1008-1987 , vol., no., pp.1-28, 17 June 1987

Team Information

Cory Johannes - johannes@iastate.edu - Frontend/Research
Jose Lopez - jdlopez@iastate.edu - Frontend
Ryan Quigley - rquigley@iastate.edu - Android/Step Tracking
Travis Harbaugh - travis@iastate.edu - Hololens/Android
Ben Holmes - btholmes@iastate.edu - Android/Bluetooth/Sound
Anthony House - houseant@iastate.edu - Frontend/Backend/Server Admin