Team Number: **sdmay18-34**
Client: **Optical Operations**
Adviser: **Dr. Daji Qiao**
Project Manager: **Chandler Chockalingam**
Report Manager: **Chris Stapler**
Software Architect: **Jason Ramirez**
Co-Chief Engineer: **Josua Gonzalez-Neal**
Co-Chief Engineer: **Victor Da Silva**
QA Lead: **Logan Highland**
sdmay18-34@iastate.edu
http://sdmay18-34.sd.ece.iastate.edu/

Eagleye: Integration of Personnel Tracking in an Augmented Reality Environment

Final Report

Table of Contents

Introduction

Problem Statement

There are inherently a lot of problems with not knowing where people are located at. Whether they are in the location they say they are, or if they are missing and others are trying to find them. When workers cannot be located, there are safety concerns, for example if you are working with something dangerous that could harm somebody such as demolition on a building. It can also mean people can get hurt doing something they should not be doing, and then lie because you have no way of seeing where they were and what they were doing. Specifically for this project, Optical Operations LLC wants to solve the problem of general contractors not having access to their personnels' locations for construction megaprojects (projects costing upwards of $1 billion). Not having accountability for workers leads to loss of money and time on these projects and can make a huge difference in budget and schedule. This problem boils down to both a safety issue and a productivity issue.

Project Statement

This project is meant to ultimately create an MVP or minimum viable product that our client can show to investors to gain capital for his ventures. This project involves three distinct parts: tracking, the HoloLens mixed reality application, and the microservices that serve as the "middleman" between the tracking token and the HoloLens application. We divided our team into three separate subteams accordingly, having two members work on each team. This divide-and-conquer method allowed us to maximize the time we had this semester to accomplish the goals stated below.

Project Goals

The goal of the project was to create a Microsoft HoloLens application that shows a live map of where different people are at any given time. The personnel are tracked through a small token, which is attached to the workers Personal Protective Equipment (PPE). This small token communicates with a distributed tracking system that relays data to the HoloLens, and a supervisor will utilize this HoloLens application for monitoring purposes.

Intended Uses and Users

The primary user for this system is a supervisor on a construction site. They will be using the HoloLens application to view the workers on their site. The secondary users will be the construction workers who are wearing the token on their belt clips. The comfort and satisfaction of both types of users are important to us because we want to make sure the token does not

interfere with the equipment they are required to wear or with their daily tasks. Since the companies will be investing a significant amount of money in the HoloLens to use the product, we want to make sure they are investing in a product that is useful for them and provides a pleasant experience for the user.

The primary use of our system is to allow a construction supervisor to visualize where their workers are at a given time and to be able to play back a visual to see data within the day. This use represents the core functionality of the product we will be building.

The aforementioned use is primarily for productivity purposes. Having real time locations and saved daily locations of all personnel on a megaproject can hugely help increase productivity. It is estimated that every day that a mega project goes over the allotted time scheduled costs an average of 3.3 million dollars. Therefore, increasing productivity can save construction companies massive amounts of money.

Additionally, the HoloLens application will be used for safety purposes. One in five worker deaths in 2015 occurred in a construction work environment [1]. These numbers could be decreased if supervisors were able to know where their employees were to avoid dangerous situations.

Expected End Product and Deliverables (Established During First Semester of Work)

- The end product shall have a tracking service.
- The end product shall be a scalable outdoor personnel device.
- The end product shall have documented source code.
- The end product shall be a Proof of Concept for the client to showcase to investors.
- The end product shall have an UI for tracking for Microsoft HoloLens.
- The end product shall have some possible casing for the product.
- The end product shall have reliable communication between devices.
- The end product shall be demonstrable to future customers and investors.

Design

Specifications

The main focus of the project is to get a system in place to track multiple personnel and display their locations on a virtual map. The initial product should be able track up 6 people on the worksite in real time with a same delay. Another design specification is the accuracy of our tracking. We would like to make our solution as accurate as possible, with a minimum accuracy of less than 5m in error.

In addition, the tracking token that will be placed on each personnel must last at least an average work day (10 hours). It also has to be able to relay readable data to the HoloLens to be displayed.This token has to be able to withstand rough conditions, must be water resistant, and reasonably sized so it will not hinder the worker wearing it. We would like it to attach to PPE so it must always be worn. The relay will also be secure so unauthorized users cannot access the information or create false data. The final requirement is that the system must be scalable, so it will be able to handle more people at larger construction sites, as well as maintainable to last the length of the construction project.

Functional Requirements:

- 3-part system: token, services, and HoloLens
- System will utilize construction sites' wifi
- Worker will keep token on belt and it will use wireless signal strength data for localization
- Admin website
- Must store data for future playback

Non-Functional Requirements

- Must track at least 6 people in a playground-sized environment (20 x 20 m): the end goal is to be able to track our group successfully in a small, coned-off simulated outside work environment.
- Must be accurate within 5 meters: the tracking must show the avatar as accurately as possible to track people in real time.
- Token relays information in a readable format to HoloLens.
- If active sensor, battery life = 1 work day (10 hours): needs to be able to last an entire workday without being charged.
- Sensor communication range: 10m: must have a minimum range of 10 m to limit the number of sensors needed.
- Real time tracking: acceptable delay of 1 second: if there is too long of a delay, the positioning will not be accurate on a moving target.
- Track people moving at maximum of 5 mph: this will allow the positioning to still be accurate on someone moving at a reasonable speed.

System Design

The design that our team has chosen is using RSSI (Received Signal Strength Indicator) data and hardware tokens to track the personnel, and then send that information to our server and display it for the user. The team also chose to have a base station that will increase the accuracy of the system. The job site will be completely covered in WiFi with an appropriate amount of access points to guarantee that anywhere the personnel go, they will be in the range of at least three access points.

The token will be a Raspberry Pi Zero, which will be attached to the toolbelt of the worker. This token will gather the RSSI data, and send it through the network to a hub that uses a path loss function to convert the RSSI data into an accurate location for the multiple tokens. Once it generates the location of the tokens, it will send this data to an offsite server. From this server, we will get the information and generate avatars at the locations on a 3D mapping of the site on the HoloLens.

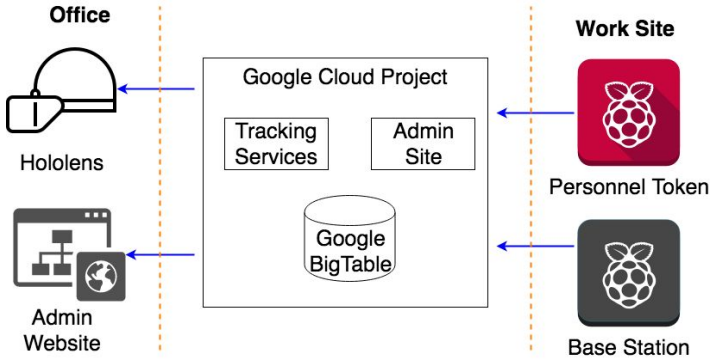
The base station is also a Raspberry Pi Zero that gathers RSSI data from multiple fixed locations and will find the optimal TxPower and Gamma Value for the Path Loss Function. It will then send this data over to our server to use those values when calculating distance.

$$L = 10\,n\,\log_{10}(d) + C$$

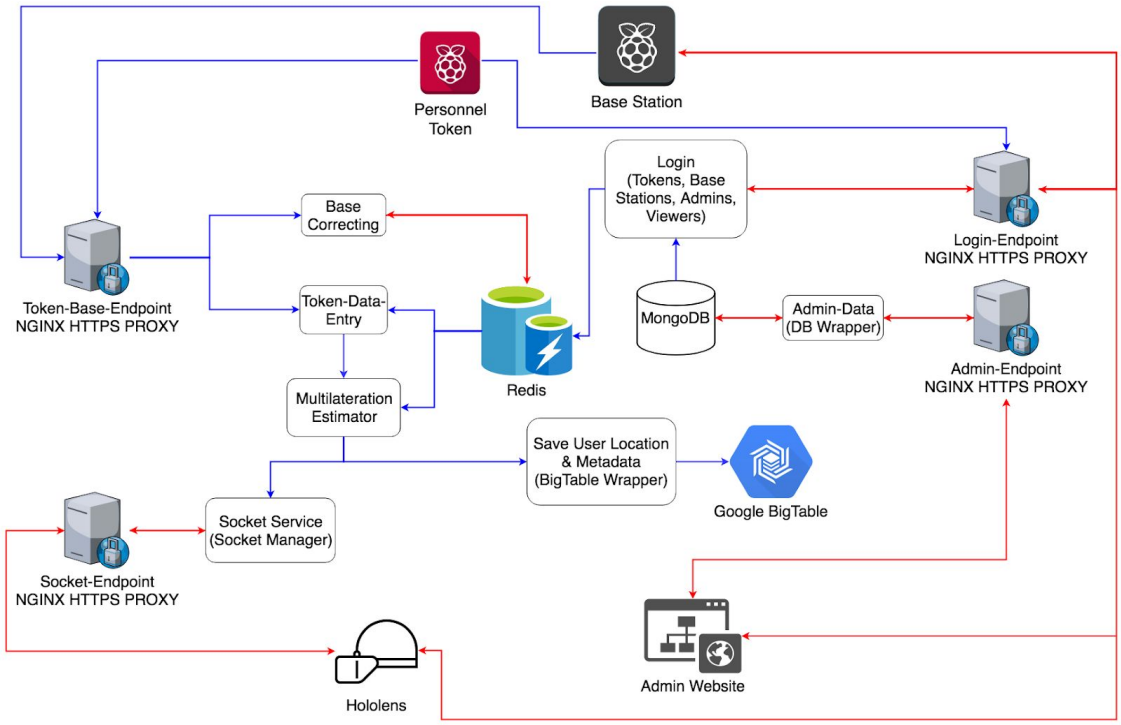
L is the path loss in decibels, n is the path loss exponent, d is the distance between the transmitter and the receiver, usually measured in meters, and C is a constant which accounts for system losses

In order for a customer to set up a worksite, they will be able to use the Admin Website that we have created to place access points based on latitude, longitude and altitude. It also allows admins to create users for work sites.

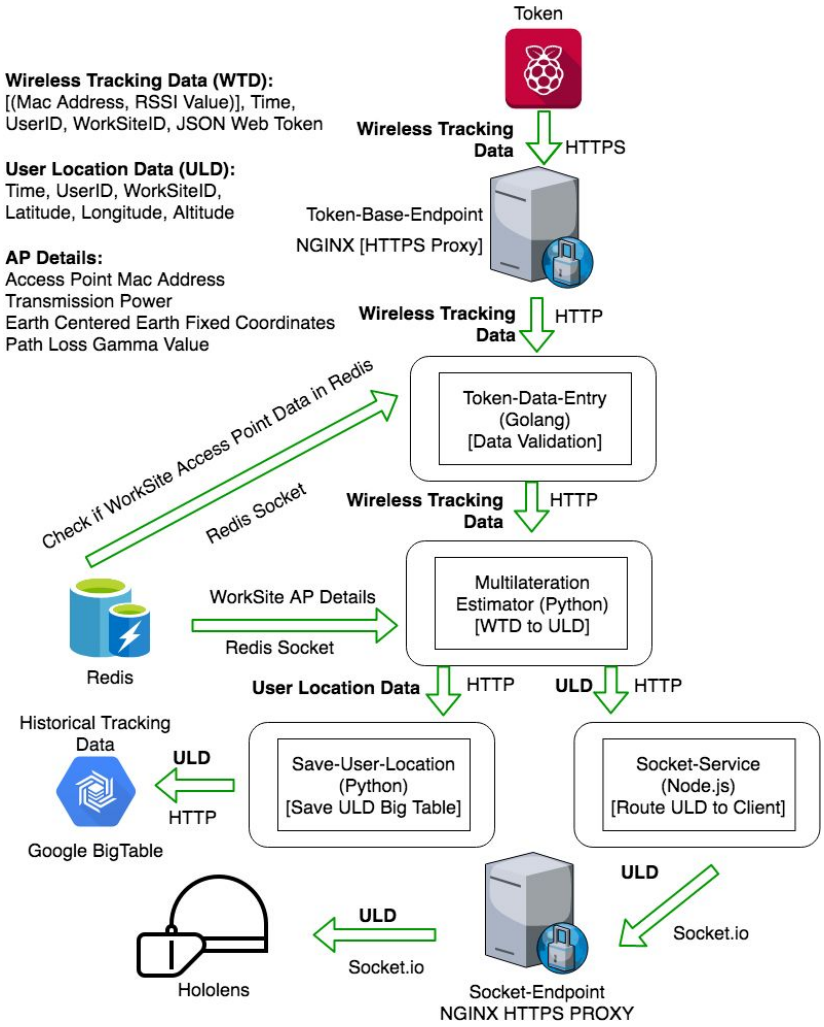
High Level Overview



Overall System



Personnel Tracking Dataflow



Design Analysis

We have compiled the areas of weakness and strength of this design:

- Areas of Weakness
    - Accuracy: The accuracy of our tracking with RSSI will not be as strong as with another technology such as CSI (Channel State Information). It will likely be within 5 m, whereas CSI could have provided us with less than 1 m accuracy.
    - IP Addresses: Due to working in a Cloud Environment that is not ours, the client does not have a Static IP Address if they choose to go from the existing Cloud Infrastructure (GCP) to another (AWS, Azure)
    - Issue with scaling the Socket-Service as it always has to hold the websocket it is currently connected to. No way of serializing a web-socket.
- Areas of Strength
    - We are able to use any programming language for the backend services of the system.
    - With RSSI, every WiFi-connected device can be tracked and see the data easier.
    - It can be used with a closed wifi network
    - The solution with RSSI that we have designed is scalable for several worksites.
    - We are secure with HTTPS routing, JSON Web Tokens for authentication and limited access to our database
    - With Kubernetes for our container management system, we are easily able to switch from any Cloud Environment (AWS, Azure, GCP, IBM) and allows for auto scaling so our client does not have to worry about how to deal with an increase in load.
    - Our design is a platform that can be added on to in the future. We have created an API to easily get data from Google BigTable.
    - Since we are using the Microsoft HoloLens over HTC Vive or Oculus Rift, the user will be using a mixed reality device, which is less draining and uncomfortable than a virtual reality device. The Microsoft HoloLens allows users to still interact with their environment as well.

Implementation Details

Hardware and Software

- Hardware
    - Raspberry Pi Zero W (Token/Base Station)
        - Attached to users PPE
    - Cisco Meraki MR74 Access Points
        - Act as reference points for multilateration
    - Microsoft HoloLens
        - Worn by construction supervisor in indoor setting
- Software
    - Tracking
        - Connect to Services via WiFi
        - Scan for surrounding wireless access points
        - Report findings
    - Services
        - Triangulation
            - RAM redis server
            - Algorithms
            - Scheduling Algorithm
        - Save User Location
            - Database Wrapper
        - Location Tracking Data (LTD)
            - Socket Connection
            - RAM redis server
        - Setup
            - Backend
            - Frontend
    - Microsoft HoloLens
        - Frontend functionality
            - Menu display
            - Display Map
        - Backend functionality
            - Communicate with LTD
            - Push information to Frontend

Interface Specifications

Hardware

We used Raspberry Pi Zeros as a main hardware tool for tracking due to their flexibility and low cost. In order to keep the form factor of the tracking device small, we use the microcontroller's onboard Wifi chipset. We will be gathering RSSI on the Raspberry Pis using the linux built-in tool for scanning wireless networks, iwlist. Our client has bought 6 Raspberry Pi Zeros for us to use for our demo. For displaying the 3D map of the the worksite the HoloLens, the only additional configuration required for the HoloLens is network connectivity to interface with the HoloLens service, so it can get and display personnel location on its map. The Access Points that our client plans on using for the purpose of known locations in our tracking system is Cisco's Meraki APs. Currently we have 5 APs setup in the parking lot of the ISU Startup Factory.

Software

In terms of software, our team uses a multitude of different frameworks, languages, and platforms, including: Unity, Raspbian, C#, Python, Git, Nodejs, Kubernetes, Google Big Table, and NGINX. Unity and Visual Studio are needed to run, build, and test the Microsoft HoloLens application. The code for the HoloLens is first built and tested in Unity and then using the Unity built-in tools for building an application for various platforms, it then must be built for UWP. Once built for the HoloLens, it can be sent to the HoloLens or HoloLens Emulator via Microsoft Visual Studio. Raspbian lite is the base OS for Raspberry Pis. We use Raspbian lite because it does not have a desktop environment, for the purpose of saving resources so that the device can last longer on battery power. The program that runs on the Pis relies on python and packages for sending http request to our web services. For the various backend services we run to support the the tracking system the technologies we use varies greatly. Each publicly facing endpoint uses an NGINX server to process requests and implement a secure HTTPS connection to the connected party. Taking the path of a tracker/base station request, the first services they will hit after the NGINX server will be Base Correction and Data Entry Services. These services then talk to Redis and the multilateration estimator. Other software routes on the services side are detailed above in the overall system diagram.

Testing Process and Results

Testing Process

- Token relays information in a readable format to HoloLens.
    - We tested this by continually changing data format to ensure that the HoloLens was retrieving the data correctly.

- Sensor communication range: 10 meters: must have a minimum range of 10 meters to limit the number of sensors needed.
    - Walked around demo area and confirmed that anywhere inside the area could access information from at least three access points.
- Real time tracking: acceptable delay of one second: if there is too long of a delay, the positioning will not be accurate on a moving target.
    - Tracked people as they walk around the environment, confirmed the location they were in was being represented within one second, given that we already knew their location

- Realistically-sized token: wearable device attached to belt that is no bigger than an iPhone 8
    - Our test strategy was to test multiple devices to find one that felt the most natural

- Create a system that is scalable to megaprojects (projects greater than one billion dollars): must be able to scale for projects that have a lot more than six workers.
    - We tested this by attempting to connect to services with as many people as possible to get an idea of how many people can initially connect without it breaking
    - Also we attempted to design code for design clarity and modifiability to make sure when the number of necessary users change, our code can be changed to fit needs
- Maintainable for length of project: for construction site, must be able to be easily maintained for length of construction project for at least five months at a time.
    - Tested by continuously testing the project as time goes on to ensure that it is maintained
- If active sensor, battery life = one work day (10 hours): needs to be able to last an entire workday without being charged.
    - We tested this by leaving tester on for length of a work day with program running
- Must be accurate within five meters: the tracking must show the avatar as accurately as possible to track people in real time.
    - We tested this by having people carry the tokens in known locations, allowing us to compare the location we detect from the token and the location we know.
- Must track at least six people in a playground-sized environment (20 x 20 m): the end goal is to be able to track our group successfully in a small, coned-off simulated outside work environment.
    - We tested this by having six or more users wear the hardware token (Raspberry Pi Zero devices) in an area that was 40 x 40 m in size, and monitor the tracking data to ensure it fits within our accuracy.

Testing Results

- Must track at least six people in a playground-sized environment (20 x 20 m): the end goal is to be able to track our group successfully in a small, coned-off simulated outside work environment.
    - We ended up changing our testing strategy to enhance our accuracy, so we now have less people with multiple tokens receiving data.
- Must be accurate within five meters: the tracking must show the avatar as accurately as possible to track people in real time.
    - From the testing, we ended with an average accuracy just outside five meters accuracy, hovering around six meters.
- Token relays information in a readable format to HoloLens.
    - The test results for this were very positive, as by the end of the project we were able to successfully relay all important information from the token, through the services, and to the HoloLens.
- If active sensor, battery life = one work day (10 hours): needs to be able to last an entire workday without being charged.
    - Using portable battery packs, we were able to get well over the 10 hour goal we set for ourselves.

- Sensor communication range: 10 meters: must have a minimum range of 10 meters to limit the number of sensors needed.
    - Walked around demo area and confirmed that anywhere inside the area could access information from at least three access points.
- Real time tracking: acceptable delay of one second: if there is too long of a delay, the positioning will not be accurate on a moving target.
    - Our results for this goal were harder to determine, as we ended up having to change our connection from sockets to http request, slowing down the data transfer. This led to a longer delay in the tracking.
- Maintainable for length of project: for construction site, must be able to be easily maintained for length of construction project for at least five months at a time.
    - Have been successfully running the tests throughout the whole semester, which has been approximately the amount of time we wanted to prove.

Related Products and Literature

Related Products

Construction

- Buildertrend - The #1 Construction Management Software for home builders & remodelers
- B2W Track - Field Tracking & Analysis
- Accuware - Indoor and GPS location tracking components(https://www.accuware.com)
- QTRX systems - Locate, Map, Track. Indoors (https://www.trxsystems.com).

Augmented Reality Construction

DAQRI Worksense - Professional Augmented Reality. DAQRI designed a suite of AR features for professionals. The Worksense platform allows clients to use Augmented Reality in the workspace by providing communication tools, 3D scanning, tracking tags, real-world models, and AR guiding. (https://daqri.com/worksense/)

Overview

Many related products are associated to individual aspects of our product, but our project does not have any competitor that provides our exact solution. As a result, many of the related products are not directly competing with our client. The products discussed earlier are products that relate to software products that have intentions to sell to construction companies. DAQR's product is closer to our solution, but does not seem to have the intention to sell a product that tracks construction workers or vehicles.

Related Literature

Research Topics

- Tracking-Learning-Detection (https://ieeexplore.ieee.org/document/6104061/)
- Enhanced RSSI-Based Real-Time User Location Tracking System for Indoor and Outdoor Environments (https://ieeexplore.ieee.org/abstract/document/4420422/)
- RSSI-based localization for wireless sensor networks with a mobile beacon (https://ieeexplore.ieee.org/document/6411544/)
- GPS-less low-cost outdoor localization for very small devices (https://ieeexplore.ieee.org/abstract/document/878533/)

- Application of WiFi-based indoor positioning system for labor tracking at construction sites (https://www.sciencedirect.com/science/article/pii/S092658051000107X)
- A Self-Adaptive Model-Based Wi-Fi Indoor Localization Method (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5191055/)
- Unsupervised Learning for Solving RSS Hardware Variance Problem in WiFi Localization (https://link.springer.com/content/pdf/10.1007%2Fs11036-008-0139-0.pdf)
- Using Pattern Matching Algorithm and Signal Propagation Model to Localise Mobile Devices in a WLAN (https://ieeexplore.ieee.org/document/4078850/)
- A study on outdoor positioning technology using GPS and WiFi networks (https://ieeexplore.ieee.org/document/4919345/)
- A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.78.1407&rep=rep1&type=pdf)
- Accurate Indoor Localization With Zero Start-up Cost (https://groups.csail.mit.edu/drl/wiki/images/6/6a/sgil_mobicom2014.pdf)
- Decimeter Level Localization Using WiFi (https://web.stanford.edu/~skatti/pubs/sigcomm15-spotfi.pdf)
- A Learning-Based Approach for Indoor Localization (http://conferences.sigcomm.org/co-next/2011/papers/1569469851.pdf)
- Open Wireless Positioning System: A Wi-Fi Based Indoor Position System (https://ieeexplore.ieee.org/abstract/document/5378966/)
- Survey of Wireless Indoor Positioning Techniques and Systems (https://ieeexplore.ieee.org/document/4343996/#full-text-section)
- A Real Time Location Search Primer (https://www.promotionworld.com/se/articles/article/100531-Real-Time-Location-Search-Primer)
- An Algorithm for Fast, Model-Free Tracking Indoors (https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/32882.pdf)
- Robust 3D people tracking and positioning system in a semi-overlapped multi-camera environment (https://ieeexplore.ieee.org/abstract/document/4712340/)
- Hierarchical Geographical Modeling of User Locations from Social Media Posts (https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/40840.pdf)
- Extracting Patterns from Location History (https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/37522.pdf)
- UPTIME: Ubiquitous pedestrian tracking using mobile phones (https://ieeexplore.ieee.org/abstract/document/6214359/?reload=true)
- Self-corrective knowledge-based hybrid tracking system using BIM and multimodal sensors (https://www.sciencedirect.com/science/article/pii/S147403461630252X)
- Personnel tracking on construction sites using video cameras (https://www.sciencedirect.com/science/article/pii/S147403460900038X)
- Bayesian Device-Free Localization and Tracking in a Binary RF Sensor Network (http://www.mdpi.com/1424-8220/17/5/969/htm)

Patents

- Estimation of position using WLAN access point radio propagation characteristics in a WLAN positioning system (https://patents.google.com/patent/US7515578)
- System of three dimensional positioning and tracking (https://patents.google.com/patent/US6316934)
- Wireless mobile indoor/outdoor tracking system (https://patents.google.com/patent/US7852262)
- A multi-range location tracking system for tracking a location objects has one or more GPS-enabled accessory devices (https://patents.google.com/patent/US20080062120)

Conclusion

Overall, the final product that we created was a complete system that takes in personnel locations on a construction site and outputs an augmented reality map displaying the location of the personnel. The separate sections that the project was divided into was the Location Tracking Division, the Services Division, and the HoloLens Division. The Location Division used the Raspberry Pi Zeros in order to get the RSSI information and sent this information to the services division. Once received by the Location Division, this RSSI information would be taken and pushed through triangulation algorithms in order to get locations in relation to our access points. This information would be translated into latitude and longitude coordinates and passed to the HoloLens Division. The HoloLens Division takes the latitude and longitude information and displays a 3D model of the map in the augmented reality environment.

This project had a few different specification changes throughout the two semesters that the team was working on it for. Many of these specification changes occurred with the way that location would be determined. These specification changes made it possible for the team to learn about real-world project requirements. They were setbacks, but the broad set of specifications gave the team valuable experience. Overall, the project covered a full-stack development process and gave the team experience with software and hardware development, communication with clients, and adaptation, which we can all use in the future.

References

[1] "Commonly Used Statistics" Occupational Safety and Health Administration. [Online]. Available: www.osha.gov/oshstats/commonstats.html

Appendix I: Operation Manual

How to Set Up Tracking (Tokens and Access Points)

1. **Access Point Setup:**
    1. Arrange access points in a hexagonal fashion (See above picture for Access points setup in the parking lot of the Vermeer Research Hub).
2. **Token Setup:**
    1. Install Raspbian lite on Raspberry Pis designated for use as tokens:
        a. Download Raspbian lite Stretch from main site https://www.raspberrypi.org/downloads/raspbian/
        b. Flash Raspbian lite on the microSD card via a flashing program (we used Etcher)
        c. Add empty file titled ssh to boot partition of microSD drive.
        d. Change /etc/wpa_supplicant file in rootfs partition to include network credentials of hotspot or WPA2 WiFi (Syntax can be found here)
        e. Add microSD card back to Raspberry Pi
    2. For each Raspberry Pi:
        a. Add a user cstapler with no password sudo permission (Help found here).
        b. Download tracking code on Raspberry Pi (currently hosted at https://git.ece.iastate.edu/sd/sdmay18-34)
        c. Make sure proper dependencies are installed with install_dependencies script under application parent folder Tracking folder.
        d. Configure mDns on Raspberry:
            i. Edit /etc/hostname to the desired mDns name (e.g. opticalrp1)
            ii. Add same name to new line on /etc/hosts 127.0.1.1 *desired mDs name*
    3. For Setup\Controller computer:
        a. Install Ansible from System's package manager
        b. Download tracking code on Raspberry Pi (currently hosted at https://git.ece.iastate.edu/sd/sdmay18-34)
        c. Ensure that from this device each pi is accessible over ssh with cstapler user. (To access the Raspberry Pis via their mDns name, mDns daemon must be installed on the computer AND the pis and Setup computer must be on the same network)
        d. Add mDNS hostnames to sdmay18-34/Tracking/hosts file after [all].
        e. To Start, View Status, And Stop Tracking code on all pis run ansible-playbook -i hosts start_tracker.yaml, ansible-playbook -i hosts status_tracker.yaml, ansible-playbook -i hosts stop_tracker.yaml respectively

How to Set Up Services

1. Create a Google Cloud Account
    a. https://cloud.google.com
2. Enable Google Cloud Billing
    a. https://cloud.google.com/billing/docs/how-to/modify-project
    b. Google will give you $300 free dollars to use
3. Create a Project
    a. https://cloud.google.com/resource-manager/docs/creating-managing-projects
4. Enable Container Build API
    a. https://console.cloud.google.com/flows/enableapi?apiid=cloudbuild.googleapis.com&redirect=https://cloud.google.com/container-builder/docs/quickstart-gcloud
5. In your computer terminal, setup Google Cloud SDK
    a. https://cloud.google.com/sdk/docs/
6. Login to Google Cloud SDK from Terminal
    a. gcloud auth login
7. Set the project ID
    a. gcloud config set project [PROJECT_ID]
8. From inside the git project folder, go to Services/Tracking-Cluster/app/
9. Run in your terminal: "gcloud container builds submit --tag gcr.io/[PROJECT_ID]/[SERVICE-NAME] [SERVICE-NAME]/." For all the services to create the docker containers
    a. admin-data
    b. base-correcting
    c. http-routing
    d. multilateration-estimator
    e. other-login
    f. save-user-location
    g. token-data-entry
    h. token-login
10. In your browser, use the Google Cloud Shell
    a. https://console.cloud.google.com/cloudshell
11. Create SSL Certificates in the Google Shell for the frontend nginx servers
    a. https://cloud.google.com/compute/docs/load-balancing/http/ssl-certificates
    b. admin-viewer-frontend
    c. login-frontend
    d. token-base-frontend
12. Pull our GitLab repository into Google Cloud Shell
    a. git clone https://git.ece.iastate.edu/sd/sdmay18-34.git
13. Create Static IP Addresses for endpoints
    a. https://cloud.google.com/compute/docs/ip-addresses/reserve-static-external-ip-address
    b. tracking-cluster-admin-viewer

    c. tracking-cluster-hololens
    d. tracking-cluster-login
    e. Tracking-cluster-token-base
14. Update Static IP Addresses in kubernetes config service files
    a. sdmay18-34/Services/Tracking-Cluster/kuberentes/services/admin-viewer-frontend.yaml
    b. sdmay18-34/Services/Tracking-Cluster/kuberentes/services/hololens-frontend.yaml
    c. sdmay18-34/Services/Tracking-Cluster/kuberentes/services/login-frontend.yaml
    d. sdmay18-34/Services/Tracking-Cluster/kuberentes/services/token-base-frontend.yaml
15. Add a persistent disk to save user information on MongoDB
    a. https://cloud.google.com/compute/docs/disks/add-persistent-disk#create_disk
    b. Google Cloud Engine Persistent Disk Name: mongo-tracking-cluster
16. Create a kubernetes cluster on Google Cloud Project
    a. https://cloud.google.com/kubernetes-engine/docs/quickstart
    b. Make sure to have cluster size be at least 4
17. Back in Google Cloud Shell, go into our project kubernetes configurations
    a. cd sdmay18-34/Services/Tracking-Cluster/kubernetes/
18. Start up the server on kubernetes in Google Cloud Shell
    a. Setup SSL Certificates Secrets in Kuberentes
        i. kubectl create secret generic admin-viewer-tls-certs --from-file=[ADMIN-VIEWER-CERTIFICATE-FOLDER-NAME]/
        ii. kubectl create secret generic login-tls-certs --from-file=[LOGIN-CERTIFICATE-FOLDER-NAME]/
        iii. kubectl create secret generic token-base-tls-certs --from-file=[TOKEN-BASE-CERTIFICATE-FOLDER-NAME]/
    b. Setup nginx Configurations in Kuberentes
        i. kubectl create configmap nginx-admin-viewer-frontend-conf --from-file=nginx-configs/admin-viewer-frontend-configs/
        ii. kubectl create configmap nginx-login-frontend-conf --from-file=nginx-configs/login-frontend-configs/
        iii. kubectl create configmap nginx-token-base-frontend-conf --from-file=nginx-configs/token-base-frontend-configs/
        iv. kubectl create configmap nginx-hololens-frontend-conf --from-file=nginx-configs/hololens-frontend-configs/
    c. Set up Kubernetes Deployments and services
        i. Wait until the pod is fully created and running
            1. To check, in Google Cloud Shell, run kubectl get pods
        ii. MongoDB Service
            1. kubectl create -f volumes/pv_mongo.yaml
            2. kubectl create -f volumes/pvc_mongo.yaml
            3. kubectl create -f services/mongo.yaml

4. kubectl create -f deployments/mongo.yaml

iii. Redis Service
   1. kubectl create -f services/redis.yaml
   2. kubectl create -f deployments/redis.yaml

iv. Save-User-Location Service
   1. kubectl create -f services/save-user-location.yaml
   2. kubectl create -f deployments/save-user-location.yaml

v. Admin-Data Service
   1. kubectl create -f services/admin-data.yaml
   2. kubectl create -f deployments/admin-data.yaml

vi. Other-Login Service
   1. kubectl create -f services/other-login.yaml
   2. kubectl create -f deployments/other-login.yaml

vii. Token-Login Service
   1. kubectl create -f services/token-login.yaml
   2. kubectl create -f deployments/token-login.yaml

viii. HTTP-Routing Service
   1. kubectl create -f services/http-routing.yaml
   2. kubectl create -f deployments/http-routing.yaml

ix. Multilateration-Estimator Service
   1. kubectl create -f services/multilateration-estimator.yaml
   2. kubectl create -f deployments/multilateration-estimator.yaml

x. Admin-Viewer-Frontend Service
   1. kubectl create -f services/admin-viewer-frontend.yaml
   2. kubectl create -f deployments/admin-viewer-frontend.yaml

xi. Login-Frontend Service
   1. kubectl create -f services/login-frontend.yaml
   2. kubectl create -f deployments/login-frontend.yaml

xii. Token-Data-Entry Service
   1. kubectl create -f services/token-data-entry.yaml
   2. kubectl create -f deployments/token-data-entry.yaml

xiii. Base-Correcting Service
   1. kubectl create -f services/base-correcting.yaml
   2. kubectl create -f deployments/base-correcting.yaml

xiv. Token-Base-Frontend Service
   1. kubectl create -f services/token-base-frontend.yaml
   2. kubectl create -f deployments/token-base-frontend.yaml

xv. HoloLens-Frontend Service
   1. kubectl create -f services/hololens-frontend.yaml
   2. kubectl create -f deployments/hololens-frontend.yaml

19. Done!

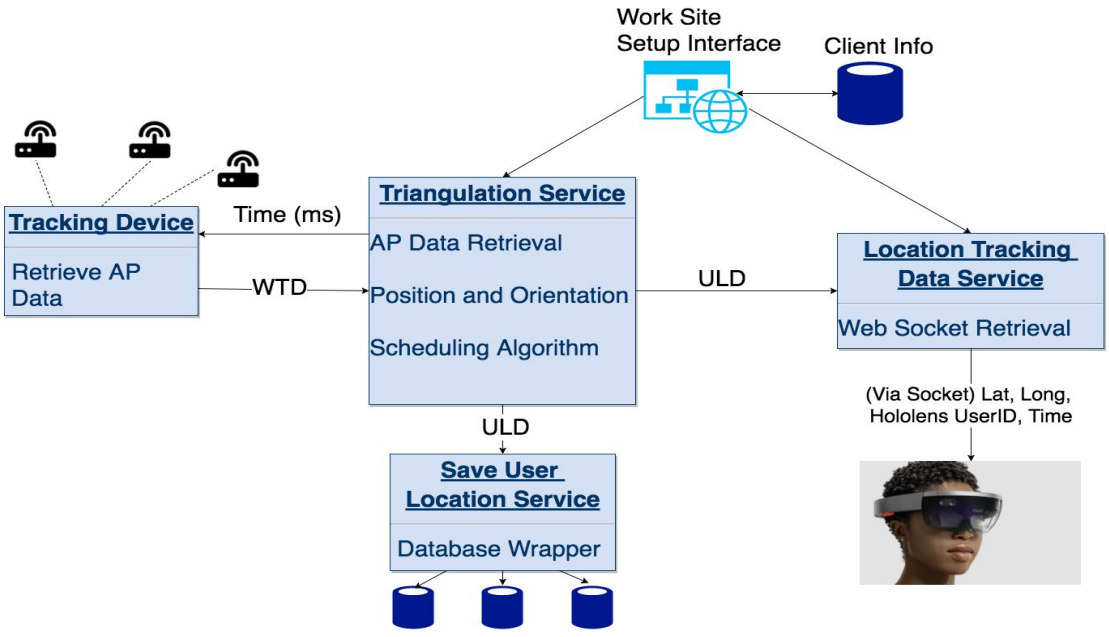
How to Set Up HoloLens

1. Download Unity
    a. https://unity3d.com
2. Create a free account
    a. https://id.unity.com/en/conversations/dc674c9c-5277-4c34-8c28-5f4cb0daaaf7004f
3. Pull HoloLens project from github
4. Install MixedRealtyToolkit for Unity
    a. Follow Instructions from the Toolkit's Getting Started (https://github.com/Microsoft/MixedRealityToolkit-Unity/blob/master/GettingStarted.md)
5. Open HoloLens project in Unity
6. Import Toolkit
    a. Asset -> Import Package -> Custom Package…
7. Setup Unity project settings
    a. Chapter 6 - Build and deploy to device from Visual Studio
8. Install JSONObject asset
    a. Go to the Asset Store
    b. Search for JSONObject
    c. Download JSONObject by Defective Studios
9. Build
    a. File -> Build Settings -> UWP Platform -> HoloLens Type -> Unity C# project -> Latest Install -> Build
    b. Open up the App in Visual Studio
        i. Name "app"
    c. Build X86
    d. Release with HoloLens plugged in
    e. Choose HoloLens device to build to

Appendix II: Initial Versions of the Design

Our initial version of the design of our system involved implementing tracking using CSI (or Channel State Information) data, rather than RSSI (Received Signal Strength Data), like the system we ultimately implemented uses. We planned to implement a system that used CSI data based off of an existing product called SpotFi, which is documented in a paper titled, "*SpotFI: Decimeter Level Localization Using WiFi* and *Decimeter-Level Localization with a Single WiFi Access Point*". We originally wanted to use CSI data because it is more accurate than RSSI (historically, systems that use CSI data have been able to obtain decimeter-level accuracy, whereas RSSI-based tracking systems typically get accuracy within 5-10 meters). This seemed like an ideal optional; however, we ran into substantial issues when we were using the Intel 5300 chip for gathering CSI data using a PC running Linux. We spent many weeks trying to get it to work. In the end, we decided it was not worth our time to keep trying, and instead decided to implement the tracking system using RSSI.

Our initial version of the backend also was much smaller, with fewer individual services.



Appendix III: Lessons Learned

During this project, we learned a variety of lessons about technical work, scheduling, time management, and working with a client on a project. One of the biggest lessons we learned was in being realistic about the scope of our project and working accordingly throughout the two semesters. Specifically, we all spent all of last semester researching tracking technologies in order to try and achieve a higher accuracy than we could with RSSI technology. We spent a lot of time trying to get a technology called CSI (Channel State Information) to work. We ignored two thirds of our project, the HoloLens application and the microservices, during the first semester of our project because we underestimated how long these parts would take. These parts ended up taking longer than we anticipated. If we had not used so much time focusing on tracking, we could have come up with more improved solutions for the other parts of our project.

For scheduling and time management, we had two meeting times during the week devoted to senior design, but did not have a lot of team availability outside of those times to work together due to other classes and commitments. In the end, we probably would have either selected a project with a lesser scope and time commitment, or made an effort to cancel commitments or skip classes to find times that worked for everyone.

Another lesson we learned was related to meeting our client's expectations throughout the project. We had established a set of requirements and project guidelines at the beginning of the project with our client. We updated this set of requirements at the end of the semester. However, throughout the spring semester, our client would change his mind, update the project requirements, or come up with a new idea, but would not communicate changes to us, or communicated them at the end of the semester. This could have been avoided by bringing up expectations every week with our client and staying in the loop, but instead we spent our weekly standup meetings discussing about what we had accomplished and issues we had, and doing testing that needed to happen at the Startup Factory.

Appendix IV: Code

Our source code is located in an Iowa State University GitLab repository at https://git.ece.iastate.edu/sd/sdmay18-34. We do not have our source code posted to a public repository, per our client's request.