Pegasus Protection Services' Indoor Positioning Security System

Christian Silva, Isaiah Williams, Dylan Sauerbrun, Aundre' Fredericks

Dept. of Electrical and Computer Engineering
University of Central Florida, Orlando, Florida
32816 – 2450

***Abstract* — Wireless Indoor Positioning Systems (IPS) are commonly used in commercial and industrial applications to monitor the movement of entities (objects and people) throughout secure locations where protocols such as GPS lack the required precision. While these systems excel at providing information about entity location, they fail at providing accurate entity identification as well without significant human involvement (having security personnel on-site to confirm identity). To remedy this issue of identification, a more holistic approach to security is required. This paper outlines the design and implementation of an indoor security system that utilizes Bluetooth Low Energy (BLE) beacons and tags in tandem with Computer Vision to provide both personnel positioning and personnel identification, respectively. The combination of these technologies results in an automated real-time security system that will provide the information about where personnel are located as well as their identities without the need of significant human involvement.**

***Index Terms* -- Security., Bluetooth, Indoor Positioning System, Computer Vision, Personnel tracking**

I. Introduction

In this day and age organizations across many industries make use of asset tracking systems to increase their operation efficiency as well as operation security. To maximize the amount of information available, many of these organizations make use of real-time tracking systems to gather information about the location of equipment, merchandise, and even personnel. One such use-case is the implementation of passive Radio-Frequency identification (RFID) cards and readers to determine and restrict location access of certain personnel. Many indoor tracking systems make use of a form of passive tracking, such as RFID or NFC as it reduces the technological overhead, while still providing asset location.

While these passive systems are acceptable for ascertaining the location history of assets, they fail at providing the real-time location accuracy required for more high-security use-cases. In such cases active (real-time) systems are required in order to provide an organization with accurate up-to-date information about an assets position. These active systems through the use of wireless technologies such as Bluetooth Low Energy (BLE) or Ultra-Wide Band (UWB). When concerning security, these systems solve the problem of obtaining position data of assets, however they do not completely solve the issue of asset identification.

When considering the tracking of people throughout secure locations one must consider a situation in which intruders can obtain or fabricate the credentials of validated personnel. In this situation having just the tracking system alone will not ensure site security, so a system of identity verification must be used in addition to these systems. To solve this aspect of security it is common to rely on biometric identification, as generally those cannot be easily replicated by bad actors. When concerning a real time system one valid solution is the introduction of a facial recognition component in order to provide identification and with the combination of these solutions one can achieve a real-time security system that provides position information as well as identification of the personnel that are being tracked.

The Security System designed in this project will solve the issues of obtaining real-time personnel location along with personnel identification. The issue of obtaining positioning is resolved through the use of Bluetooth Low Energy (BLE) tags and beacons along with the use of a trilateration algorithm to accurately calculate indoor position data of tagged personnel. The matter of identifying personnel is resolved with the implementation of a computer vision component of the system. This component utilizes facial detection and facial recognition to identify registered and non-registered personnel entering a secure location.

The BLE beacons and tags in this project make use of the Bluetooth component from the Espressif ESP32 microcontroller. Proper implementation of this system requires at minimum three stationary beacons, along with one tag that will be on the target that is being tracked. The three Beacons will calculate their relative Received Signal Strength Indicator (RSSI) values and will send this data to a local application that will process these values through a trilateration algorithm and display the estimated locations onto the application.

The computer vision system that will work in tandem with this BLE system will determine the identity of the target that enters a secure location, assuming that they are located within the system's database. The facial detection and recognition software were developed in Python using the open source OpenCV software library. The camera system will continuously monitor the entrance to a secure room and will detect and identify personnel entering the room. The information concerning the identity of these individuals are sent to the software application and this data is processed to provide security personnel with alerts depending on who is identified.

II. Project Objectives

A. Group Goals

For our group the main objective for this project was to use the knowledge gained through our education and prior

experiences and apply that to solving a real-world problem. Our group is composed of all CpEs so we wanted a project that included a significant software portion, but also had hardware and embedded components to it. With the inclusion of Computer vision, embedded programming, and BLE, beacons and tags we feel as if our security system idea accomplished our personal requirements for this project. We felt that this security system was sufficiently challenging while also containing elements that each member was comfortable with working on and could also be accomplished within the scope of this senior design course.

B. Requirement Specifications

The engineering specifications we decided on were aimed at the accuracy and speed of our positioning component and recognition software. We felt that these attributes were important as we wanted to create a real-time system where it is necessary to set specific time constraints. The facial recognition component is the backbone of the security system as it provides identification of individuals, so it was important to ensure that it was able to detect and recognize an individual the instant they came into view of the camera. We felt as if 5 seconds was the absolute maximum time the system should take to identify before it became a security issue. In the implementation of our system, it takes approximately 0.5 seconds to recognize a face and provide that data to our security application.

As for the positioning component of our system, the accuracy and maximum detection range is important to ensure that the security system is aware of what tag is being detected and where they are as early as possible. We decided on the maximum range of 10m as we felt that this distance is suitable for accurate positioning in most medium sized rooms, we tested the system in. This range can be increased by improving the actual broadcasting power of the BLE antennas. We felt that the positioning being within 1 meter of accuracy was sufficient enough to represent the location of an individual for security purposes. Finally, since this is meant to be an automated system with relatively low involvement from individuals we wish to detect we wanted the beacons to be as small as possible so that they could be placed in a user's pocket or perhaps attached to an ID card of some kind so we decided to restrict the size of the beacon to at least 120mm x 70mm x 40mm. Figure 1 shows the demonstrable engineering specifications from those discussed above.

Some standards we adhered to include IEC 62676-5:2018,ISO/IEC 19794-5:2005 and ISO/IEC 30137-1:2019 for quality requirements for security camera images, as well as proper specifications for implementing a biometric recognition system. For the Bluetooth component we also followed standards such as ISO/IEC 18305:2016 and IEEE 802.15.1-2002 which regard the testing and evaluation of tracking systems, as well as specifications for wireless personal area networks.

Engineering Specifications	Values
Beacon Size	< 120mm x 70mm x40mm
Successful Recognition	< 5s
BLE tag detection range	Up to 10m
Detection Range	< 3 Meters
BLE position accuracy	1 meter accuracy

Fig. 1. Demonstrable engineering Specifications

IV. *System Design*

Our system is made up of three major components, the indoor positioning component, computer vision component, and the software application. The indoor position component involves the use of 3 BLE stationary beacons and 1 BLE tag. This tag will be in possession of the person we are tracking and when activated will be constantly broadcasting a Bluetooth signal to the 3 beacons. These beacons will store a tag's MAC address and Received Signal Strength indicator (RSSI) value into a data payload that will be sent to a Mosquitto MQTT server. This MQTT will then send (publish) the payload to our software application where the distance is calculated using RSSI, and with that distance we will determine the position of the tag through trilateration.

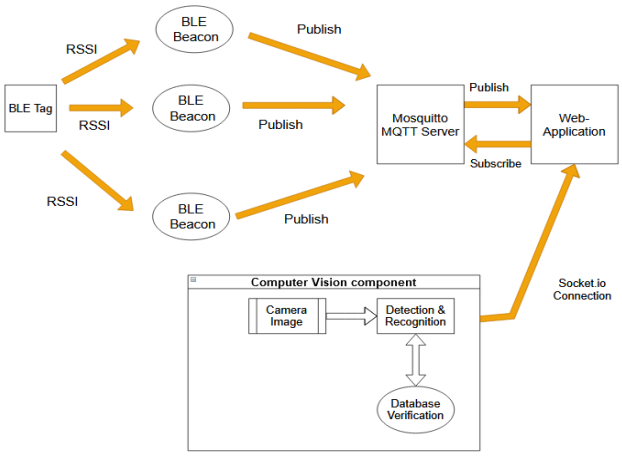


Fig. 2. System Block Diagram

The purpose of the computer vision component of this system is to identify personnel entering a room for our security system. All verified personnel will have their pictures placed in a dataset and we will train the recognition software when a new user is registered within our system. When a user enters the frame of a video the recognition software will determine if they are in our dataset and if they are not then they are given an 'unknown' tag. This information is sent to the software application where it will be processed accordingly.

Finally, the software application is where the data from our edge devices is collected and manipulated to give us the automated alerts. This software Application is where we manipulate the data from the beacons and create alerts based on security infractions that we determine. The software application will also display the real-time position of the BLE tags on a map of the location we are monitoring. Any BLE devices or faces that are detected by the network will populate tables within our main dashboard page.

A. PCB Design

For our project we have two generations of design. Our printed circuit boards (PCBs) were designed in Eagle and EasyEDA due to the extensive library available. The microcontroller we selected based on popularity, price, and extensive documentation was the ESP32 WROOM 32D module. The ESP32 module satisfied our requirement for Bluetooth and Wi-Fi components since Bluetooth will be used for the implementation of our indoor tracking and Wi-Fi will be used to send over data from our printed circuit board to our web application.

Originally our first-generation design was made with the intention that it would be a standalone beacon. Our first-generation design was mainly a voltage regulator designed to power on a 30 pin ESP32 development kit module that we had purchased prior for testing our initial code. Due to its relatively large size of (115mm x 67mm) we decided that it would be best for this design to act as just a stationary beacon that will emit Bluetooth signal using the ESP32 module from the development kit, as it would be too big for an individual person to carry around as a tag. The main voltage regulator being used in this design is an LM7805 voltage regulator. The regulator is an old component that turned out to be inefficient. To power on our first generation design we used 12 volts to 9 volts male power jack, which the regulator outputted to 5 volts giving us an efficiency rating around 41% - 55%.

For our second generation design we decided we wanted to essentially create our own development kit and get rid of other features that we do not need. The main component of this design was again the ESP32 chip module. Unlike our other design, which consisted of through-hole components, our second-generation design used surface-mount devices which are smaller, cheaper, and more efficient. Our second-generation design consists of three main subsystems, the ESP32 module, the power circuitry for the module and the USB-C circuitry which is used for uploading and pushing code onto our ESP32 module. Our new printed circuit board design can be powered on by either a USB-C input or by connecting a LiPo battery pack, giving it the option to be either stationary as a beacon or mobile as a tag. The LiPo battery is also able to be charged via USB-C connection by the MCP73831T component to increase the lifespan of our beacon/tag printed circuit board design. The surface mounted regulator we used is the XC6220B331MR-G. This regulator is more efficient than our first-generation design, taking in input voltages of 3.7 volts from LiPo and 5 volts from the USB-C connection we see an efficiency rating of 66% to 89% which is a huge increase compared to our first-generation design.

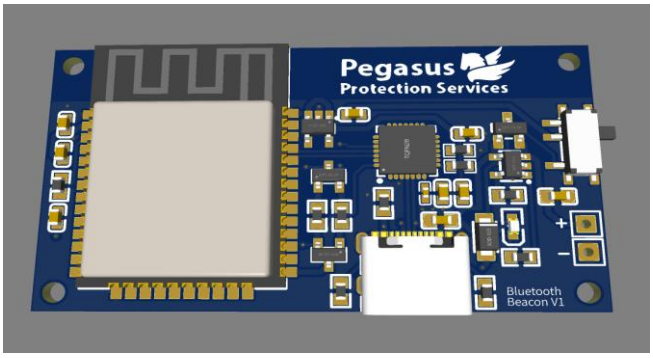


Fig. 3. 2nd. generation PCB design for use as BLE Tag/Beacon

The main improvement our second generation had was size. Due to the use of surface mount devices, we were able to reduce the size to 50mm x 28mm x 10mm.As a result, our new design is now able to act as either a beacon or a tag depending on what code we plan to upload on them and whether you want it to be powered on via a regular USB-C connection or LiPo battery. This reduction in size makes it easier for consumers to place around their specialized location(beacon) and easy enough for them to carry around in their pockets(tags). In figure 3 you can see the finalized design of our second-generation beacon.

B. Communication Protocols

In order to send data from our beacons to our software application, a communication network must be established. For our indoor tracking system, we decided to choose the popular open-source message broker known as Mosquitto. Mosquitto implements the Message Queuing Telemetry Transport (MQTT) protocol to establish a bi-directional communication server to allow messaging. This connection acts as a bridge between the beacons/tags and the web application. Because the protocol is lightweight in nature [1], sending RSSI data at a high rate is no issue. With the ability to publish and subscribe to MQTT, tracking data visually is made easier and more efficient.

For our face recognition system, we decided to use Socket.io to pull incoming data. Unlike MQTT, Socket.io is an event-based communication protocol. Socket.io implements the WebSocket communication protocol which provides a full-duplex and low latency channel between the server and the

browser [2].We use Socket.io to implement a connection between the facial recognition software (client) and the Web Application utilizing the Socket.io WebSocket protocol (server).

C. Trilateration

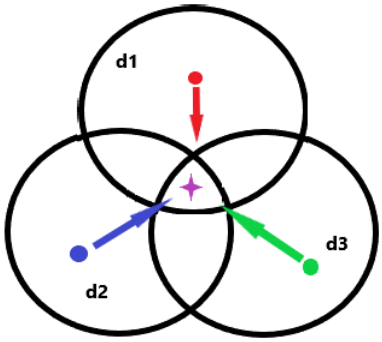


Fig. 4. Trilateration Implementation

Trilateration uses distance estimation against RSSI data to indicate where our tag is in reference to the three stationary beacons. The three stationary beacons all hold an (x, y) coordinate and based on signal strength from the tag in comparison to the coordinates, estimated coordinates for the tag are calculated. Trilateration requires multiple steps before the algorithm reaches a result that accurately represents indoor position within 1 meter. Firstly, the RSSI value from the incoming tag is received from beacon. The RSSI is calculated using ESP32's built in getRSSI() function. Once the RSSI value from the tag to its respective beacon is transmitted, the data is transferred over to the software application using MQTT where the data will be manipulated.

Next, is to calculate the distance from the respective tag to beacon using that incoming RSSI value (1).

$$d = 10^{((TxPower - RSSI) / (10 * n))} \quad (1)$$

The distance formula consists of three parameters that must be accounted for. The first parameter TxPower, is known as Measured Powered, in other words the 1-meter RSSI [6] which for the ESP32 module is -60 at 1 meter. Next is the value of the incoming RSSI from the detected device. In this case, the tag is being detected by beacons and emitting respective RSSIs readings to its appropriate beacon. Lastly, is the 'N' parameter. 'N' is a constant that we choose to set based on the environmental factors. This variable usually ranges from 2 to 4, for our project we ran multiple tests to determine the optimal value of 'N'. Once all the parameters are filled in by the incoming data, we can now calculate the distance 'R_i' from the respective beacons to tag. Each beacon should theoretically receive a different RSSI value unless the tag is in the center of the coordinate grid. Our stationary beacons will each have their own (x, y) coordinates accompanied with a relative distance, where distance will be calculated by the incoming RSSI, and this will form the matrix in equation (2):

$$\text{Tag Position} = \begin{bmatrix} x1, y1, R1 \\ x2, y2, R2 \\ x3, y3, R3 \end{bmatrix} \quad (2)$$

From here, the trilateration implementation uses a point estimation equation based on Pythagoras Theorem [7], to obtain a resulting coordinate pair (3) to approximate the location of the tag that the system is currently detecting.

$$Tag\ X = \frac{\begin{vmatrix} R1^2\text{-}R2^2\text{-}(X1^2\text{-}X2^2)\text{-}(Y1^2\text{-}Y2^2) & 2(Y2\text{-}Y1) \\ R1^2\text{-}R3^2\text{-}(X1^2\text{-}X3^2)\text{-}(Y1^2\text{-}Y3^2) & 2(Y3\text{-}Y1) \end{vmatrix}}{\begin{vmatrix} 2(X2\text{-}X1) & 2(Y2\text{-}Y1) \\ 2(X3\text{-}X1) & 2(Y3\text{-}Y1) \end{vmatrix}} \quad (3)$$

$$Tag\ Y = \frac{\begin{vmatrix} 2(Y2-Y1) & R1^2-R2^2-(X1^2-X2^2)-(Y1^2-Y2^2) \\ 2(Y3-Y1) & R1^2-R3^2-(X1^2-X3^2)-(Y1^2-Y3^2) \end{vmatrix}}{\begin{vmatrix} 2(X2\text{-}X1) & 2(Y2\text{-}Y1) \\ 2(X3\text{-}X1) & 2(Y3\text{-}Y1) \end{vmatrix}}$$

Theoretically, the trilateration algorithm gives us an approximation within 1 meter of the actual position. In practice, each beacon reaches within .14 meters of the actual position at our tested distance. Achieving accuracy while remaining computationally efficient (low cost) was important and trilateration made this possible. On average, the response time between movement of the tag to a new location and publishing the change on the software application stays around 1 - 2 seconds due to latency.

D. Computer Vision

Our group designed the computer vision component so that it would accomplish the following:

1. Locate faces on images coming from the cameras
2. For each face, determine if that face is 'accepted' or not (according to database)
3. Update the web app with data pertaining to what the camera is currently seeing.
4. Issue alerts to the web app if a face could not be identified.

With a list of tasks to be done, we started off by preparing a machine with the tools necessary to carry out computer vision. From our research, this included installing the popular code editor VSCode, the Python programming language, and several helpful python libraries (pandas, pickle, opencv, dlib, face_recogniton, etc.). These tools gave us the critical components for programming computer vision into our system. We now share a brief description on what some of these tools provided us in the code design:

- OpenCV - All-in-one open-source computer vision library, enabling us to establish a connection with the camera, read individual frames at a time, and define shapes/text on images to show computer vision results.

- Dlib - open-source collection of machine learning algorithms, which for our case, provided us means for implementing face detection/face recognition algorithms. Originally a C++ library, but thanks to the provided Python API, enables software coded in Python to use it.
- Face_recognition - One of the most helpful libraries throughout this endeavor, providing excellent documentation [3] for building face recognition/detection, as well as including several examples for guiding users through the different features provided.
- Pickle - Useful library making the training stage as least repetitive as possible. For each dataset we worked with, we could train the set once, without having to waste time during every performance test training on the set again.

It is important to discuss the algorithms we wanted our face detection and face recognition to use for our system. The face recognition library supports two models to use for face detection, the two being the Histogram of Gradients (HOG) and using a Convolutional Neural Network (CNN). It should be noted as per the documentation [4], the HOG method works generally faster and easier for a CPU, whereas the CNN model uses a GPU in combination with CUDA libraries to achieve a higher accuracy. We decided to go with the HOG method as we wanted to identify faces as fast as possible so that it could then proceed to recognize faces on the camera as soon as possible. To summarize how the HOG method works, you look at each pixel in an image, and for each pixel, compare it to the surrounding pixels, and determine a direction in which it becomes darker. Here is an example of what the computer generates for each face it would see on the camera:

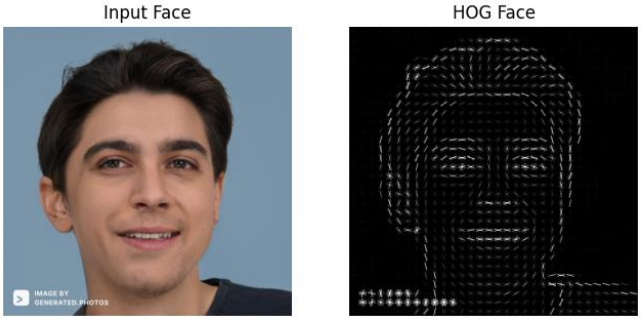


Fig. 5. Conversion to HOG Face

Notice that from the HOG version of the face, there are some notable features that you can extract fairly easily (the eyes, noise, and mouth). The face detection works by comparing the HOG version of the image it sees to a pre-trained HOG image of a face (imagine the left image was derived from a collection of HOG versions of faces), such as this:

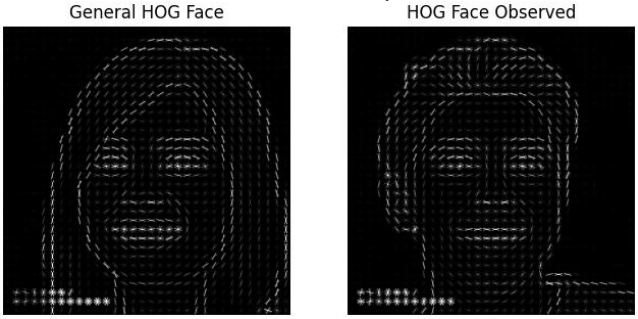


Fig. 6. Comparing General HOG to Observed

From here you can see the similar features one would expect from a human face, and thus the computer would deem the image observed as having a face. For facial recognition, the computer takes multiple measurements from a face on the perceived image to create an encoding of that image. Next, the encoding is compared to a pre-established dataset of encodings for people that are to be recognized. The computer uses a deep learning model to determine the measurements that closely relate 2 images of the same face, and separate images of different people. With this approach, we can take a raw image and convert it to a set of numbers the computer can work with [5].

Computer Vision requires a computer to perform the algorithms developed for machine learning. Furthermore, the system was intended to be portable, and so a microcomputer was needed. We opted to experiment with the Jetson Nano as the computer that would be responsible for performing the computer vision for our security system. After preparing the Nano with the tools necessary to perform some computer vision, we began testing. It is worth mentioning that while face detection and recognition of a single image each time functioned without issue, introducing the complexity of multiple images at a time through video presented problems. Particularly speaking, the Nano having 2 GB memory capacity meant there was little room to compute all the incoming frames from the camera. With different attempts to free up the memory (running Nano in headless mode to avoid any applications on screen from taking up memory, uninstalling any unnecessary software from the device), nothing seemed to allow the computer vision to run in real time. In addition, extra processing power and space would be needed to stream the output computations to the software application. From there we then tried setting up the Nano to stream the raw footage (using RTSP protocol) to an external PC (one of our computers at home) and have the computer do the computation, but there ended up being difficulty in getting a consistent smooth stream between the Nano and the computer. This was not ideal, as we want to make sure there were no extended periods of dropped frames, or else the security of the system would be in jeopardy. It was then that we decided to just run a direct connection from the camera to the computer, which seemed to allow the program to function at the very least.

E. *Software Application*

The software application is the central component of our system, it brings together our indoor tracking and our facial recognition system into one. Data coming in will be constantly updated in real time since packets will be continuously coming in from our beacons and our facial recognition system. Our software application consists of two main components, a dashboard to display alerts and a live indoor tracking feed.

Our live indoor tracking feed is implemented by the use of trilateration. For the live tracking, our web application uses a system of coordinates, in this case pixels, to display the real time location. Icons will appear on the site once three beacons

are detected. The Bluetooth icon represents the beacons which can be moved around and placed relative to the location of your floorplan. The user icons represent the tag which cannot be manually placed and are calculated relative to the 3 PCB beacons that are setup around the room. The beacons hold positions X and Y relative to their position on the map in pixels. This X and Y position for each beacon icon is found using JavaScript's Mouse event functions. Using the distance equation (1), the software application calculates the distance from each beacon to that current tag it is detecting and prepares a matrix (2) to pass to the trilateration implementation discussed above. JavaScript has its own built-in Trilateration function, based on the point estimate equations (3). The built- in Trilateration function will then return an X and Y position for the detected tags. On the software application the tag icon will update to display location relative to the floorplan that has been uploaded to the site.

Our dashboard, which is the main page of our software application, is used to display alerts. The dashboard has four tables, a table to display latest indoor tracking, latest tracking alerts, latest face tracking and latest face tracking alerts. The latest indoor tracking table publishes the payload data coming in from the Mosquitto server. This data includes the detected tag MAC address, the name associated with the tag, which beacon is emitting data, the associated RSSI reading along with a measured distance in meters and a timestamp. The purpose of this table is to display the activity going on in the secure location the beacons are placed around. Next is the Latest Tracking Alerts table. This table is responsible for populating alerts that our system detects, some of the alerts that will appear are shown below:

- User tags shows that they do not have access to this area
- User is too close to secured beacon
- User is too far from designated area

The 'Latest Face Tracking' table displayed the data coming in from the socket connection, in this case the faces that the camera system is currently detecting. This data populates who it is currently identifying, its respective location and the timestamp of the detection. The 'Latest Face Tracking' Alerts displays alerts that are prompted by the face recognition system, some of the alerts that will appear are displayed below:

- Unknown user in the area has been detected
- User has no access to this area
- A user with no tag has been detected

Using the incoming data from the Mosquitto server and the Socket we are able to manipulate the data to produce alerts and display live location.

IV. Design Implementation & Testing

A. *BLE Signal Interference*

In the Distance equation (1) mentioned in the section above, our team decided to run some tests to determine the optimal value for 'N' for our indoor tracking. As stated before, 'N' is a constant that depends on Environmental Factor, so the more obstacles that are in the way between two emitted signals, the more it will negatively affect the Received Signal Strength. In the test we conducted we decided to place a beacon and tag 3 meters apart and tested three different scenarios, Nothing in between the beacon and tag, a Human body in between the beacon and tag, and a wall between the beacon and tag. In Figure 7 you can see the results of our testing:

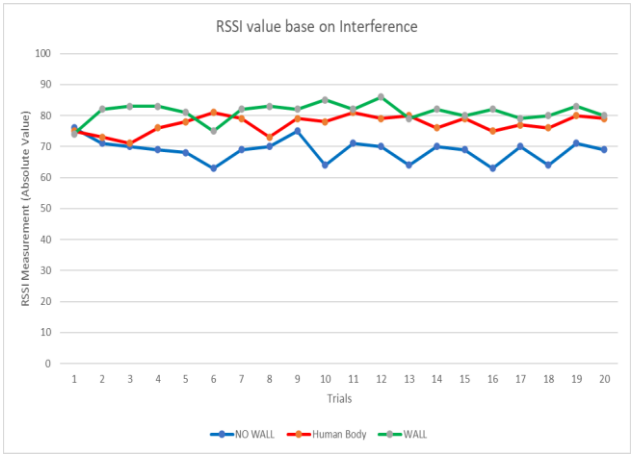


Fig. 7. RSSI value base on Interference

In figure 7, the blue line represents the signal with no physical interference between the beacon and the tag, the Orange represents a human body interference, and the green represents a wall interference. The blue line represents the strongest signal strength at an average of -68.8 for the RSSI. With Bluetooth signals the value of RSSI can fluctuate when there is a physical object between a transmitter and a receiver. When testing the effects of a human body or wall between the tags and beacons we observed an average decrease of 8.45 and 12.35 respectively. In situations where a wall or human body will be between the tags and beacons, we can adjust for the environmental factors in our distance equation. For this implementation we decided to select a value of 3 for our environmental factor variable 'N'. These tags will ideally be carried in the pockets or close to the body of the monitored personnel, so to calculate accurate distances we must consider that the incoming received signal strength is inflated due the environment around the beacon and the tag.

B. *Indoor Positioning Accuracy*

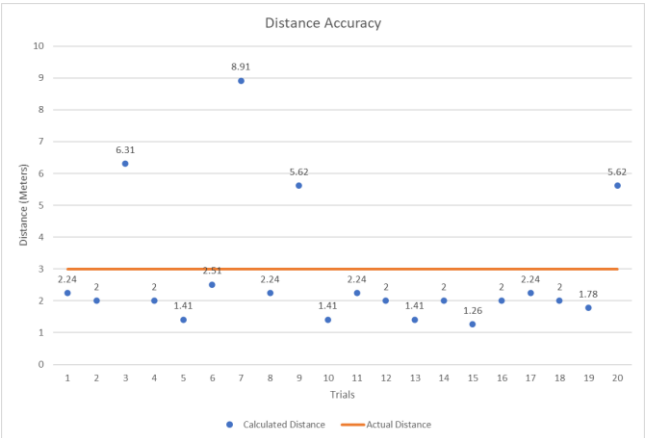


Fig. 8. Distance accuracy test for Beacon

Figure 8 represents the results of the testing to check that the variance for the calculated distance was acceptable. In this test we set a tag 3 meters away from a beacon and recorded 20 calculated distance values. When averaging out the calculated distance over this span we found that on average the system calculated a distance of 2.86 meters (14 cm error) which is within our engineering specifications for the positioning system.

C. *Computer Vision Metrics*

When implementing the computer vision component, there were two metrics that we investigated to understand the proper conditions for the camera to operate in.

1. Effective Distance Range (at 800 lumens)
2. Effective Light Range (at 7 ft)

Several trials for both metrics were conducted, with the camera in a stationary position.

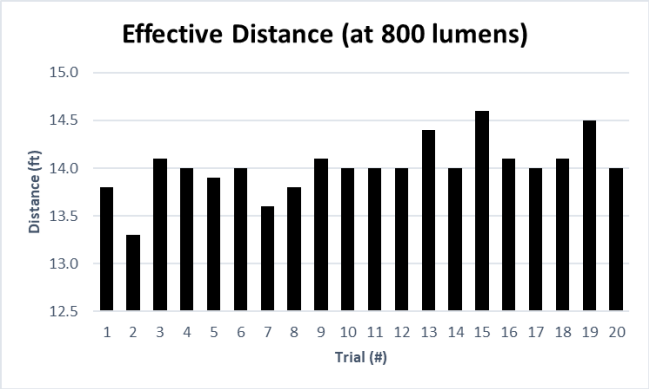


Fig. 9. Effective Distance Chart

Figure 9. depicts several trials of the effective distance, where the lights used are kept at 800 lumens (100% brightness). Note that on average, we see the effective distance at this brightness is 14 ft. Trials that performed slightly better/worse than average can be attributed to factors such as blurriness due to auto-focus, sunlight making the room brighter, and different locations for testing.

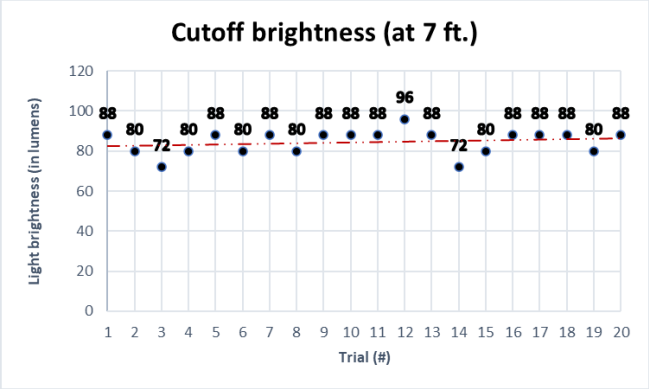


Fig. 10. Cutoff brightness at 7 ft.

The light range trials were conducted using Philips Hue A21 light bulbs, and by varying the brightness, this has shown that around 80-90 lumens is when the computer vision will fail on a face. Some outliers can be attributed to the blurriness of the camera at that point, or sunlight increasing the overall brightness in the room.

V. Conclusion

The purpose of this project was to automate the monitoring of secured locations using an Indoor Positioning System (IPS) in conjunction with facial recognition software. This provides a method of tracking an individual's path throughout these locations, while also obtaining the individual's identity. The IPS component was implemented using BLE 5.0 tags and beacons. These tags/beacons make use of the ESP32 Bluetooth antenna module. With three stationary beacons and at least one tag, trilateration can be used to approximate the position of the tag using the beacon's RSSI values. The computer vision component uses Python, OpenCV and the face recognition library to detect and recognize faces of people that enter a location. This identification info is then sent to our software application. In this software application alerts are created using the data obtained from our edge devices (camera, beacons, etc.), and all of this is done in real-time to provide the most accurate information about people's whereabouts.

Our system was able to determine the position of an individual with approximately 1 meter of accuracy. And when interference is expected, we can account for this by altering our RSSI calculations accordingly to calculate the values more accurately. Although we can account for this, the security system is best used if there are no solid walls between the Beacons and Tags to minimize signal loss. In regard to the computer vision component, the biggest environmental factor was the brightness of the room, and testing found that ~80 Lumens was the minimum light level at a distance of about 7ft, and at 800 lumens the maximum detectable range was about 14ft. The alert functionality was important for this system as it lets the person monitoring the software know when there are security infractions being committed. The system is able to detect if users are cleared for access based on their face and/or tag. The system is also able to detect if users are getting too far from or too close to the secure location as well. And the facial recognition system acts as a last point of failure, so if a person is caught by the camera trying to enter the area without a BLE tag being detected, an alert will be sent to the application.

The system was successful at providing a variety of security alerts in real time, but there is still room for improvement in the development of a system like this. Getting more accurate RSSI is crucial to obtaining more precise position data, and one way of getting that RSSI is to use stronger Bluetooth antennas. Another alternative communication protocol is Ultra-Wide Band, which provides low energy usage while also detecting small changes in distance and direction. Another improvement to consider is the use of a camera system with pan-tilt-zoom functionality as it provides much more coverage and recognition range than the current system's implementation.

VI. References

[1] S. Cope, "Beginners guide to the MQTT protocol", 2018. [Online]. Available: http://www.steves-internet-guide.com/mqtt/

[2] M.Othman, "An Introduction to Socket.IO", May 16th, 2020. [Online].Available: https://dev.to/mohamedashrafothman/an-introduction-to-socket-io-3ncb

[3] A. Geitgrey, D. King , "Face recognition", version '87a8449a', 2017. [Online]. Available: https://face-recognition.readthedocs.io

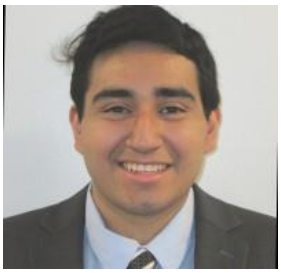
[4] A. Geitgrey, D. King , "Face_recognition package", version '87a8449a', 2017. [Online]. Available: https://face-recognition.readthedocs.io

[5] A. Geitgey, "Machine learning is fun! part 4: Modern face recognition with deep learning," Medium, 24-Sep-2020. [Online]. Available: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78. [Accessed: 16-Apr-2022].

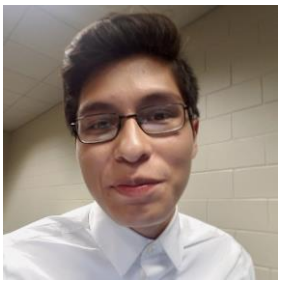
[6] A. Ghosh, "Calculating distance from RSSI value of BLE Devices", March 18, 2020. [Online]. Available: https://thecustomizewindows.com/2020/03/note-on-calculating-distance-from-rssi-value-of-ble-devices/

[7] B. Yang, L. Guo, R. Guo, M. Zhao, T. Zhao, " A novel trilateration algorithm for RSSI-based indoor localization" , July 15th, 2020, pp. 8164-8172. [Online]. Available: https://ieeexplore.ieee.org/document/9036937.

VII. Authors



Christian M. Silva will graduate with his bachelor's degree in Computer Engineering from the University of Central Florida. He will start work after graduating in the field of IT and Software engineering with Leidos. After working for some time, he plans on returning to the University of Central Florida to pursue his master's degree.



Dylan Sauerbrun is going to be graduating with a Bachelor's in Computer Engineering granted by the University of Central Florida. Intrigued by the capabilities cloud computing has to offer, Dylan will pursue companies that allow him to work more closely with cloud services.

Isaiah Williams will be graduating with a bachelor's degree in Computer Engineering from the University of Central Florida. After graduation he plans to pursue a career in software engineering.

Aundre' Fredericks will be graduating with a bachelor's degree in Computer Engineering from the University of Central. After graduation, he will start working for Texas Instruments as a Reliability Engineer as part of his first Rotation in their 2-year rotational program. After completing the rotation, he plans to leverage the skills he has learned to advance his career and eventually begin contract work or start a business of his own.