Pegasus Protection Services

Christian Silva - CpE
Isaiah Williams - CpE
Dylan Sauerbrun - CpE
Aundre’ Fredericks - CpE

Project Goals

- Create a security system that could be used to monitor personnel secure locations.
- Be able to keep track of all individuals entering and leaving an area (known or unknown)
- Wanted to implement a project/system that is based around the IoT architecture

System Objectives

Real-Time Location Tracking

- Implement Trilateration
- Constant location updates

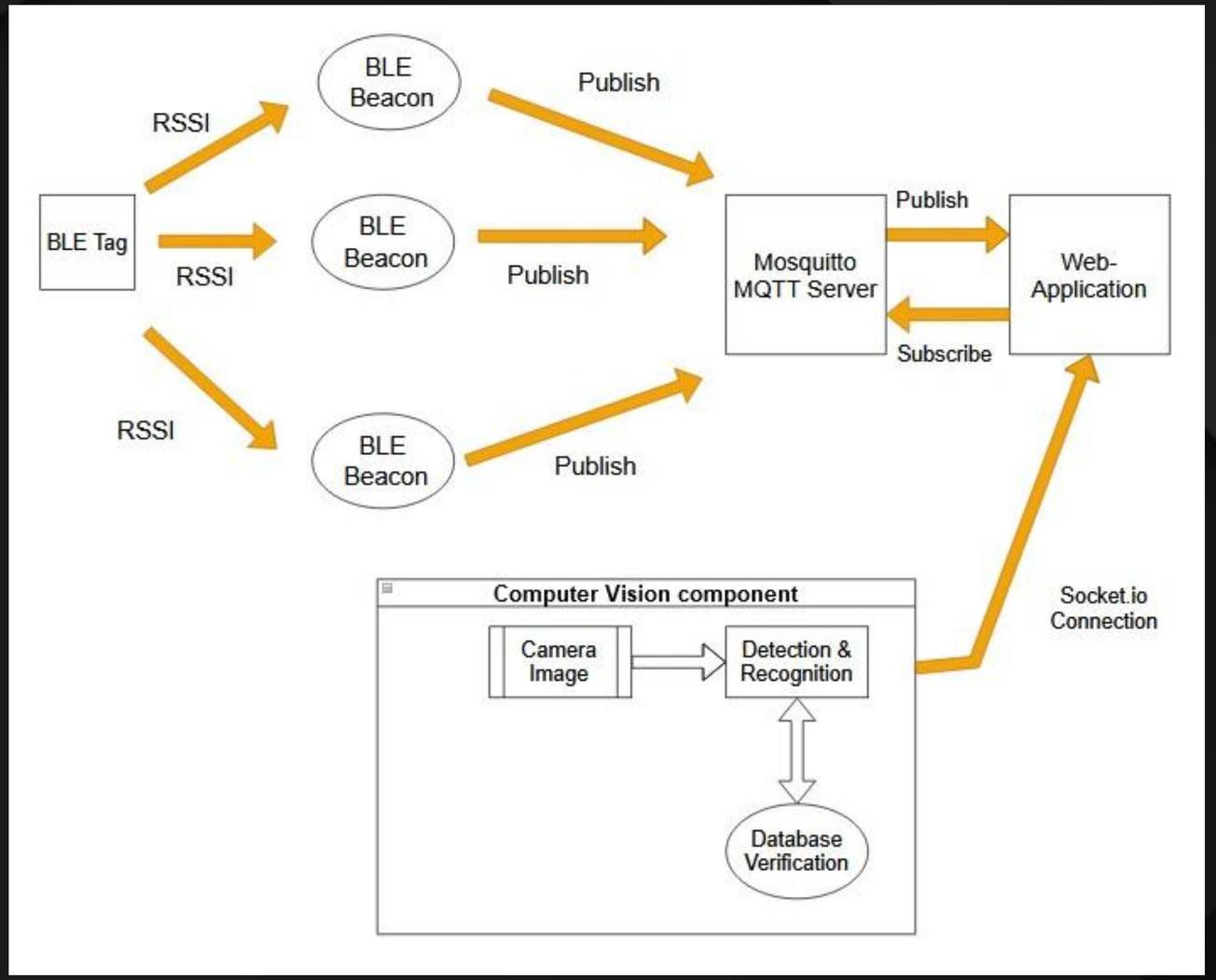
Computer Vision

- Face recognition
- Backend connection

Software App

- Security Infraction Alerts
- Tags visible on Map

System Overview



System Behavior Examples

Security Infraction Alerts:

- Non-employee face detected (not in database) or fails to recognize face
- Face detected, but no corresponding beacon detected in the area

Employee(in dataset):

- insufficient access level
- Employee travels too far from secure area
- Employee travels to close to beacon

Requirements and Specifications

- The system should be able to detect BLE tags within 1 meter of accuracy
- Facial Recognition system should have a detection range up to 3m
- Facial Recognition system should take less than 5s to identify a person
- Size of beacons should be no larger than 120 x 70mm x 40mm (should be handheld)
- Components should communicate wirelessly
- Live positioning should update within 1s
- Beacons should be able to detect tag within 10m

Existing Systems

indoors

inpixon Indoor Intelligence™

Camera Selection

Camera	Raspberry Pi Camera V2	Logitech C920s	Lenovo 500
Resolution	8 Megapixels	15 megapixel	1080p
Frames per second	30 fps @ 1080p	30 fps @ 1080p	30 fps @ 1080p
Field of View	62.2°	78°	75°

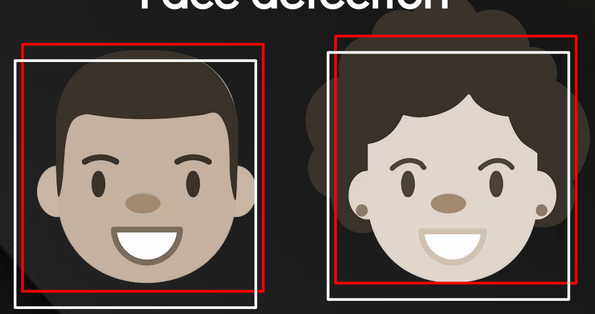
Microcomputer Selection

MicroController	Nvidia Jetson Nano
CPU	Quad-core ARM® A57 @ 1.43 GHz
GPU	128-core NVIDIA Maxwell
Memory	2 GB 64-bit LPDDR4 25.6 GB/s
I/O ports	1x USB 3.0 Type A,2x USB 2.0 Type A, USB 2.0 Micro-B, 1x MIPI CSI-2 connector 1x HDMI

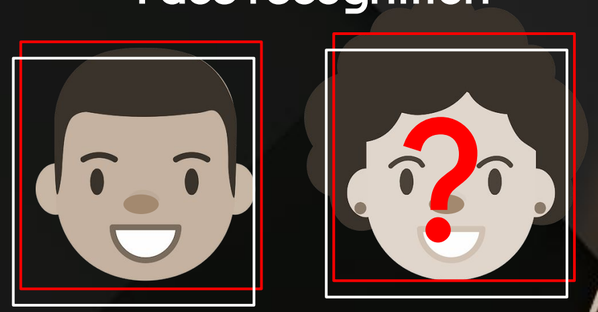
- Popular for computer vision projects
- Supports up to 4K @ 60 fps video
- Will be used support our projects Facial Recognition system

Computer Vision Overview

Face detection



Face recognition



Leo

Unknown

Possible Intruder...

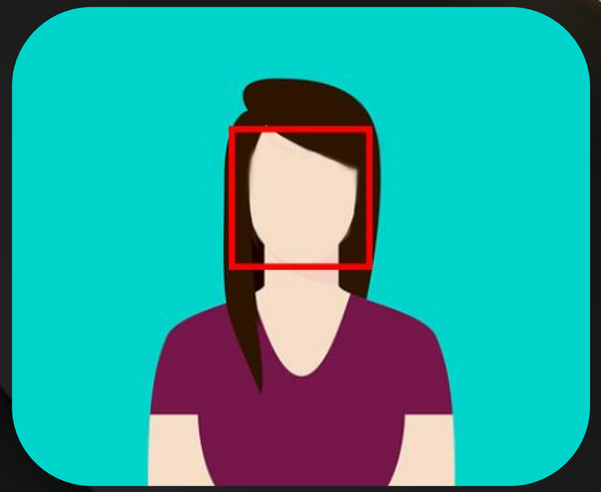
ALERT!

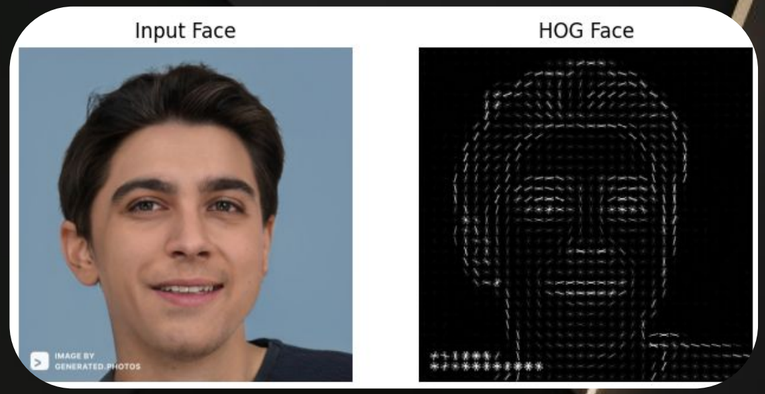
Python, OpenCV, and Dlib: The three resources

- Using Python language
  - Comfortable language for our group
  - OpenCV is supported by Python
  - Useful for future career plans
- What is OpenCV?
  - **C**omputer **V**ision library
  - Provides several modules for different aspects in robot vision, machine learning, image augmentation
  - **Open**-source (free to use!)
- What is Dlib?
  - Like OpenCV, is a library providing algorithms and methods to perform computer vision concepts.
  - Written in C++, but provides Python API

Classifying the images: a face or not?

- Face detection
  - Used for locating faces in images
  - Classifies an image as containing a face or not
  - If a face is found, proceed to identifying
  - Saves us computational resources (don't have to always try to recognize a face in continuous streaming images)
- On the face detection front, we utilized the Histogram of Gradients method.
  - For each pixel in the image, determine a direction in which it becomes darker (gradient)
  - HOG images compared to provide a fair comparison
- Why this method specifically?
  - When compared to the other options, it is generally faster and less computationally expensive.
  - Intrigued by how the computer sees faces from the camera



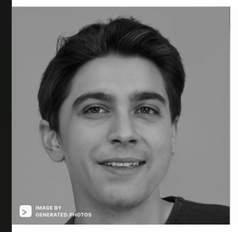


HOG representation of face

Putting a name to the face: resolve!

- Facial recognition.
  - Compare observed face to the faces in the ‘accepted faces’ set.
  - If a face cannot be ruled as ‘accepted’, the program will then issue an alert on the software application.
- Encoding the images with measurements:
  - Deep learning model identifies measurements to be taken
  - Allows us to extract ‘face encodings’ from images
  - Encodings are sent to be classified as a face in the set
- Things we have noticed on this matter:
  - Angle of face matters
  - Facial accessories (masks, hats) make it difficult
  - Light has to be decent enough (so face can show up better in image)
  - Distance can vary depending on the brightness in room

Input Image



128 Measurements Generated from Image

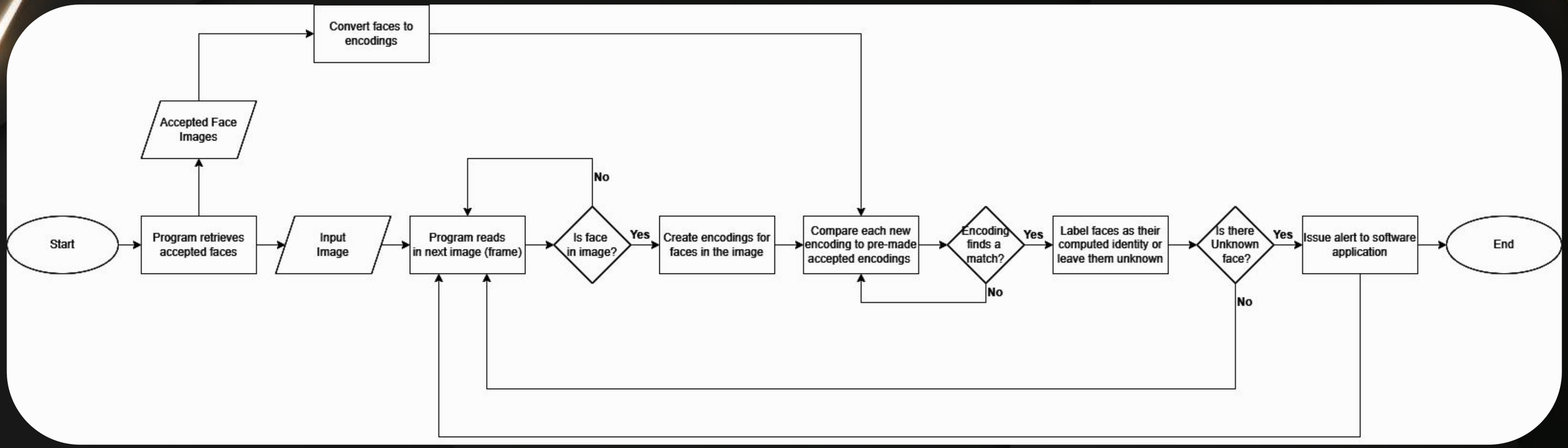
-0.1281466782093	0.097496084868908	0.032084941864014	0.045223236083984
0.17521631717682	0.12529824674129	0.020976085215807	0.060309179127216
0.10801389068365	0.030809439718723	-0.00052163278451189	-0.01981477253139
0.0731306001544	0.036050599068403	-0.1318951100111	0.065554238855839
-0.029626874253154	-0.097486883401871	-0.0059557510539889	0.1226262897253
-0.15958009660244	-0.0066401711665094	0.043374512344599	0.036750309169292
-0.031351584941149	-0.141315225158882	-0.053343612700701	0.14114324748516
-0.15042643249035	-0.048540540039539	0.078198105096817	-0.061901587992907
-0.12728653848171	-0.12567175924778	-0.076289616525173	-0.10568545013666
-0.065365232527256	-0.061418771743774	0.12369467318058	-0.074287034571171
0.14746543765068	0.046741496771574	0.056418422609568	0.006176188122481
0.0041091227903962	-0.121136501431 47	0.089727647602558	-0.21055991947651
0.021352224051952	0.061606746166945	-0.0085843298584223	0.11345765739679
-0.086726233363152	0.061989940702915	-0.022388197481632	0.19372203946114
0.094635948853878	0.10904195904732	0.020686049556136	0.084853030741215
0.21180312335491	-0.019414527341723	-0.050584398210049	0.006481129676 1036
-0.035577941685915	0.152459457516 67	-0.072376452386379	-0.16582328081131
-0.036901291459799	-0.122166685760 02	-0.034365277737379	-0.007277775558491
-0.070026844739914	0.083934605121613	-0.045013956725597	-0.059730969369411
-0.089621491730213	0.087945111095905	-0.013955107890069	0.114784322677904
0.078333757817745	-0.021407851949334	-0.17898085713387	0.148411959407971
0.13227833807468	-0.018298890441656	-0.072600327432156	0.049525424838066
-0.14132921397686	-0.011014151386917	0.005051192827522 8	-0.051016297191381
-0.13407498598099	0.009367933496832 8	-0.014829395338893	-0.062812767922878
-0.039491076022387	0.058139257133007	-0.043765489012003	0.004863874055445 2
0.071997955441475	-0.024210374802351	-0.012062266469002	-0.11443792283535
0.05228154733777	-0.057223934680223	0.012774495407939	0.014683869667351
-0.031709920614958	0.023535015061498	0.069833360612392	-0.081752359867096
0.11094793114089	-0.009803973138332 4	0.163878878818	0.037020035568953
0.18632389605045	0.020220354199409	-0.015336792916059	0.12788131833076
-0.11768248677254	0.004033768083900 2	0.10281457751989	-0.094398014247417
-0.040977258235216	0.051597066223621	-0.082041338086128	-0.10034311562777

Face represented in encoding

What's needed for the computations

- Face detection
  - Lots of existing online resources that provide datasets with faces (Labeled Faces in the Wild, CelebFaces, etc.), which we used for testing
  - Want to prioritize frontal face detection
  - As live camera feed gets sent to computer, face detection will be performed on the incoming frames.
- Facial recognition
  - Need images for each person that wants to be recognized by the computer
  - If pictures are too small, may not be able to resolve
  - Accepted faces stored in dataset used by program

Program Flowchart for Computer Vision System



Computer Vision: The ups and downs

- The helpful stuff:
  - Thanks to OpenCV, we are able to get suitable face detection running on the Nano
  - Lots of resources online that show various strategies for implementing computer vision
  - Code for throwing alert can be easily triggered by using the results of the facial recognition program.
- Problems that have arised:
  - When using Neural Networks, limited by memory of Nano
  - How delayed is the video computation result to when it is fed into the Nano?
  - What if people want to wear masks?
  - If we opt to use multiple models, how can the Nano's memory handle it?
  - Because of computation, Nano begins to run hot.

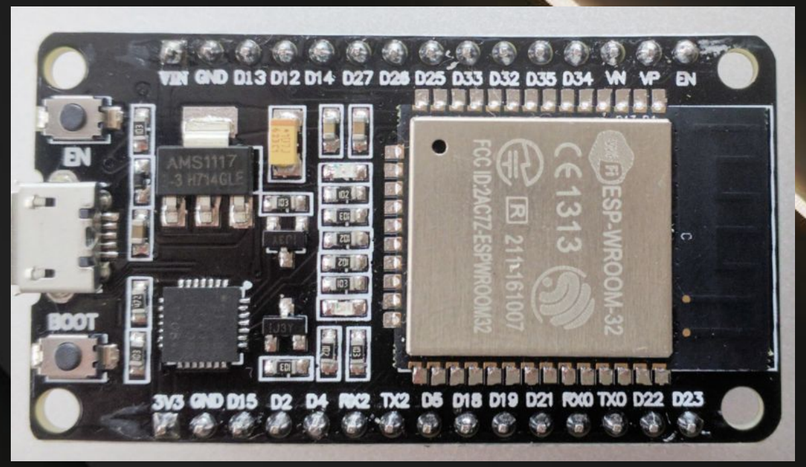
Communication Method Selection

Type	BLE	Wi-Fi
Location Accuracy	< 5 m	< 10 m
Range	Up to 100 m	Up to 500m
Latency	3-5 seconds	3-5 seconds
Power Consumption	Low	Moderate
Frequencies	2.4 GHz	5 GHz
Data Rate	2 Mbps	1 GBps

- Bluetooth Low Energy
  - Primarily being used for indoor positioning system
- Wi-Fi
  - Sending data from components to our cloud platform
  - MQTT Server

Microcontroller Selection

Board	ESP32	ESP8266	CC3200
Manufacturer	Expressif	Expressif	Texas Instruments
Operating Voltage	3.3v - 5v	2.7v - 4.0v	2.3v - 3.6v
Power Consumption	Low	Low	Low
Price	$10.00	$7.39	$55.00



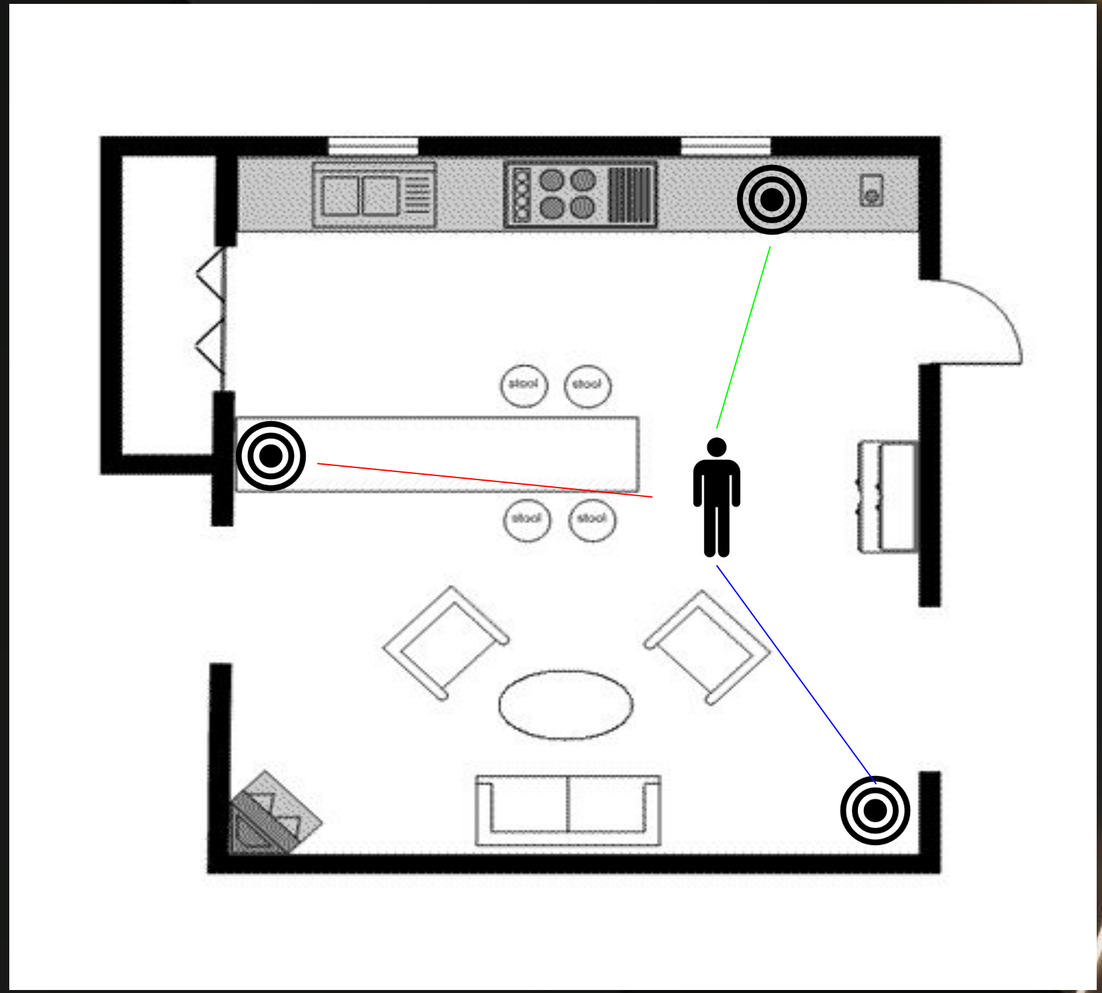
- Supports Classic Bluetooth and BLE
- Supports 802.11 b/g/n Wi-Fi Connectivity
- UART IC
- I2S
- Popular component for IOT projects

Trilateration Implementation

Hardware Requirements:
3 Stationary Beacons
1 mobile Tag

Software Requirements:
Arduino (ESP32)
Web-App
Database

How it works:
3 stationary beacons work together to create an interwoven field using RSSI. Using both distance and point estimations, transform signal data into visual output.



BLE Signal Discussion

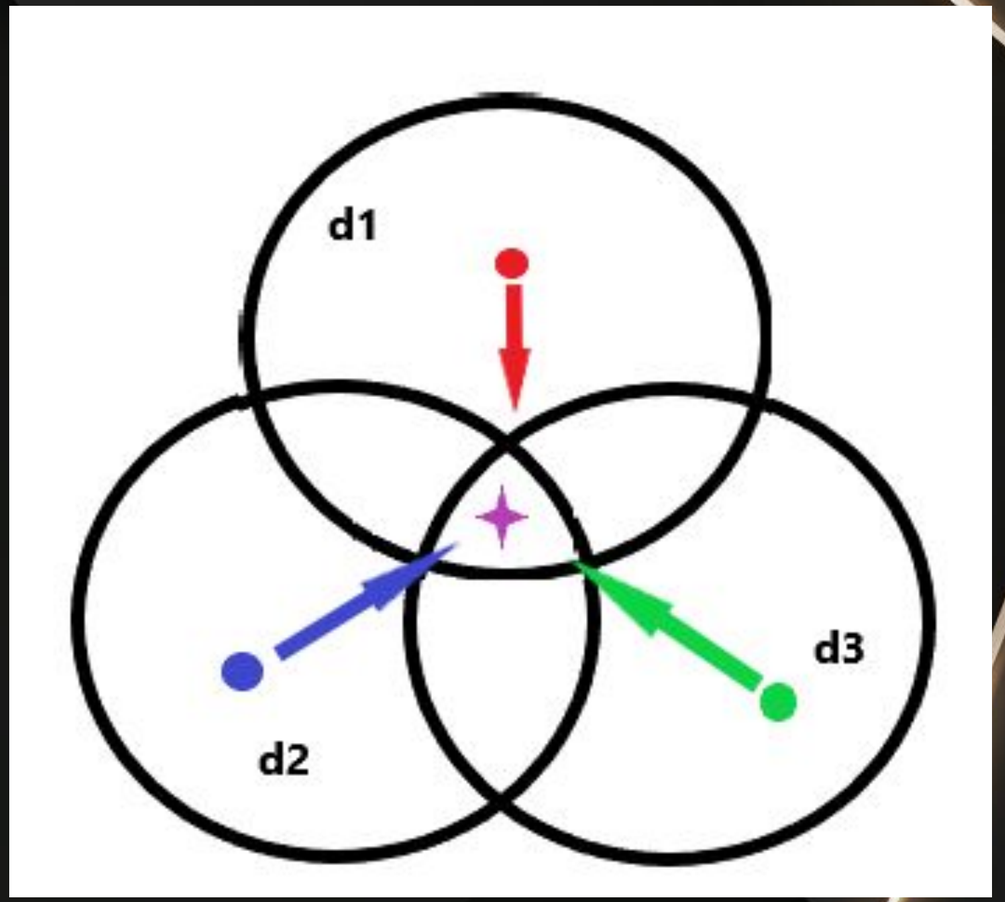
Signal Specification:

UART Signals will be used at a baud rate between 9600 and 115200

Distance Estimation:

d represents distance from tag
C represents environment constant
n represents path loss exponent
RSSI (d) = -10nlog (d) - C

Point Estimation:
d1 = (x_1, y_1)
d2 = (x_2, y_2)
d3 = (x_3, y_3)
d0 = (x_0, y_0) = min $||AX^* - B||^2$ (Purple star)



Data Transmission

Microcontroller -> BLE Tag

- Sends Serial Data via UART with UUID attached

BLE Tag -> Microcontroller

- Sends Serial Data via UART with UUID attached

Microcontroller -> Jetson Nano

- Sends packaged data to backend hosted by Microcomputer
  - UUID from Tag
  - UUID from Microcontroller
  - Timestamp

Jetson Nano -> Software

- Stores received data to database
- Compares UUID with those available in database
- Output Tag data on Dashboard (Tag w/ UUID should already exist)

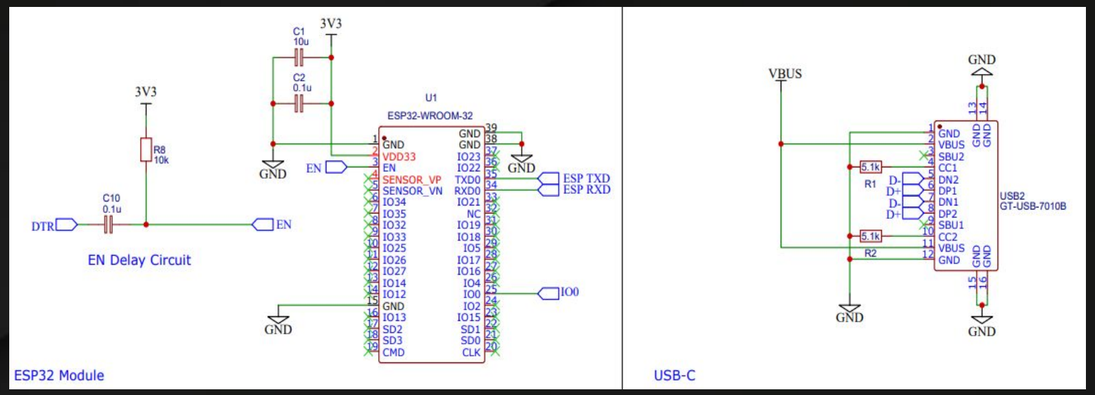
Previous Considerations(Wireless)

RFID - Radio Frequency Identification considered for implementation, but deemed unfit for the goals the group set.

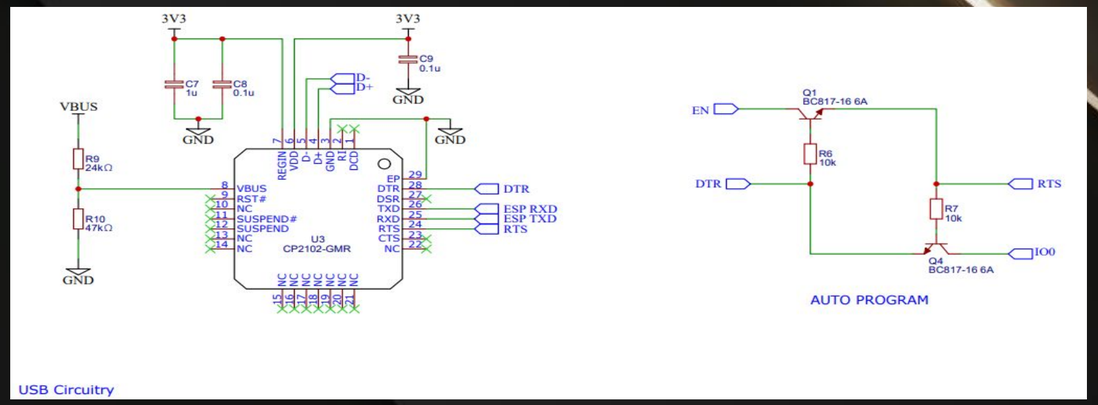
WiFi - Cell phones receiving 802.11a signal used as tag for indoor tracking. User would need to sign into our application and opt-in to allowing location services to be used.

UWB - Ultra Wide Band Technology considered for tracking indoors over a larger area, but deemed less beneficial than BLE

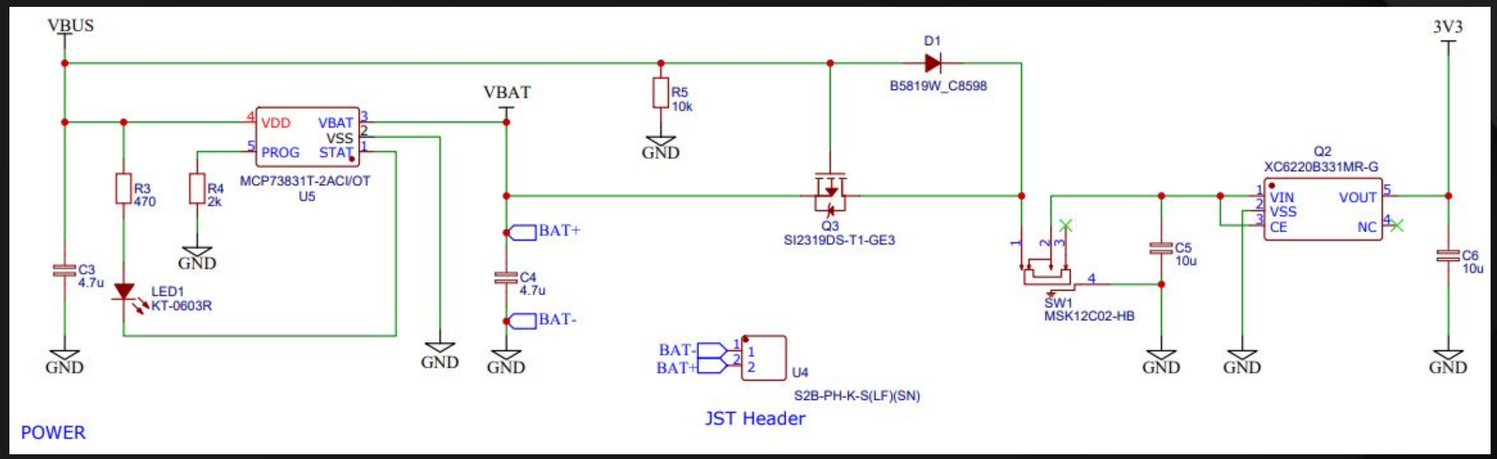
Final PCB design



ESP32 Module + USBC Component



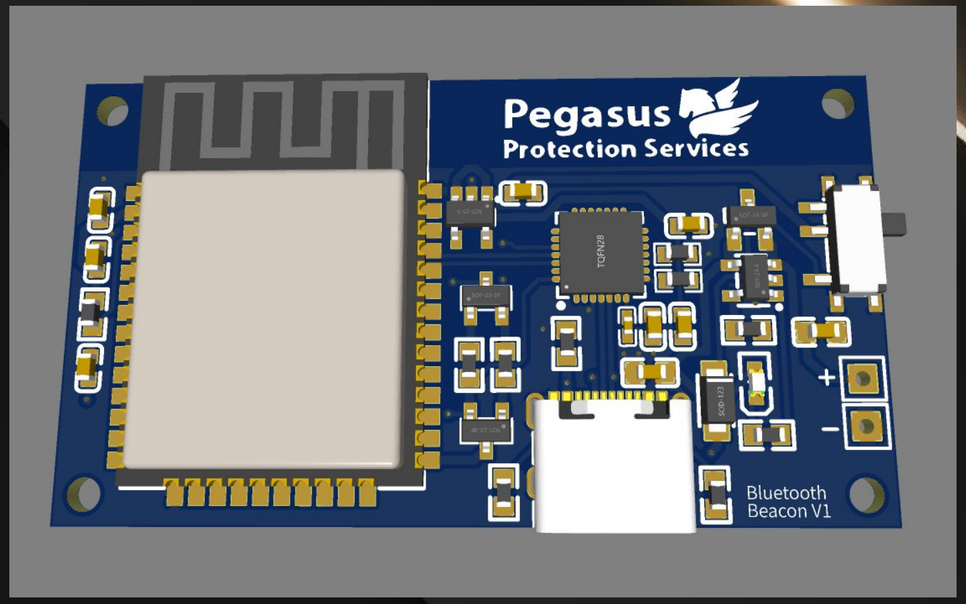
USB Circuitry + Auto Program



Power Component

Final PCB design

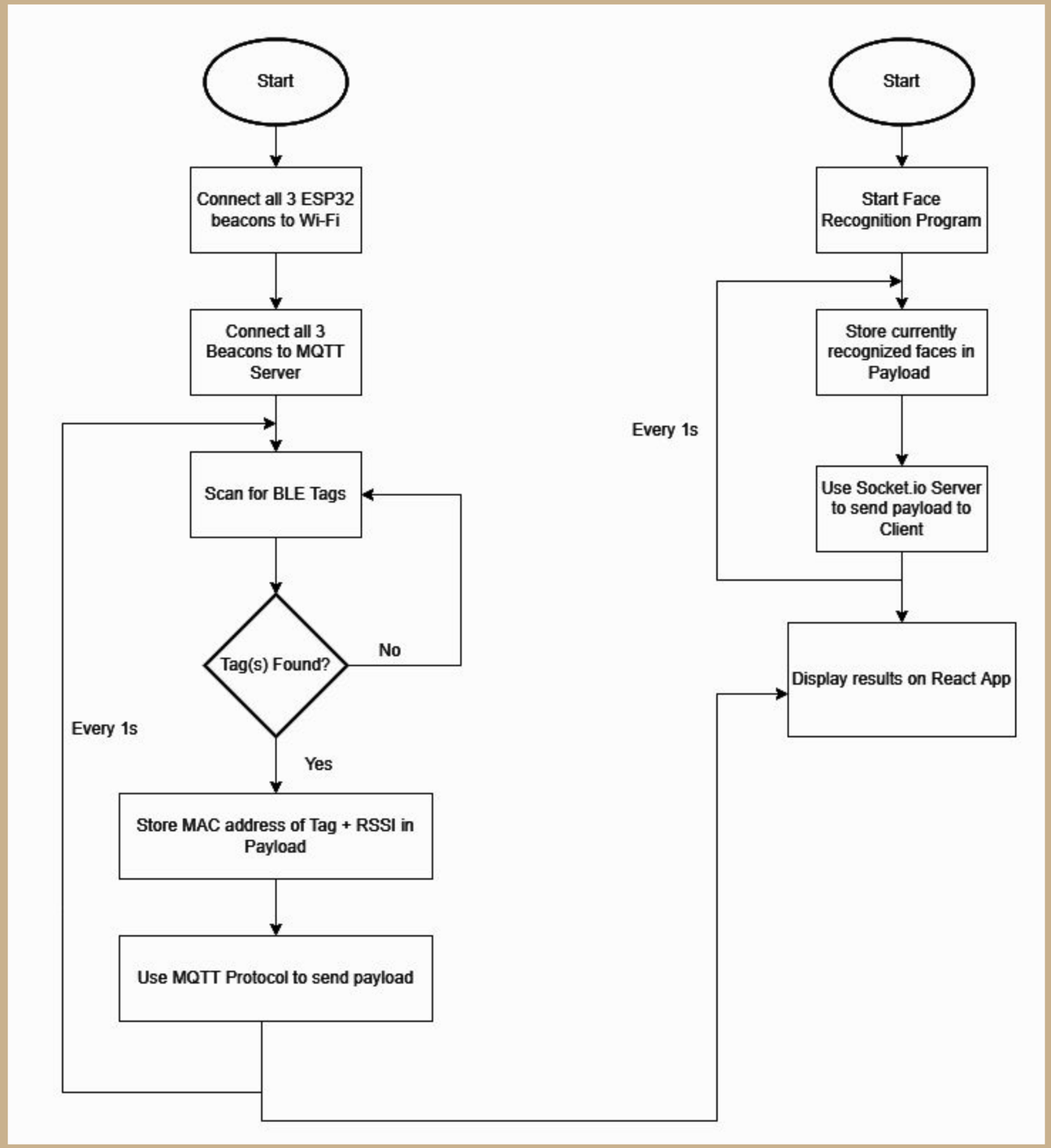
- **More efficient design and Reduced beacon size**
  - 50mm x 28mm (2 x 1 inches)
  - SMD components
- **Can be powered on by**
  - Single cell Lipo Battery input (3.7V)
  - USB-C (5V max)



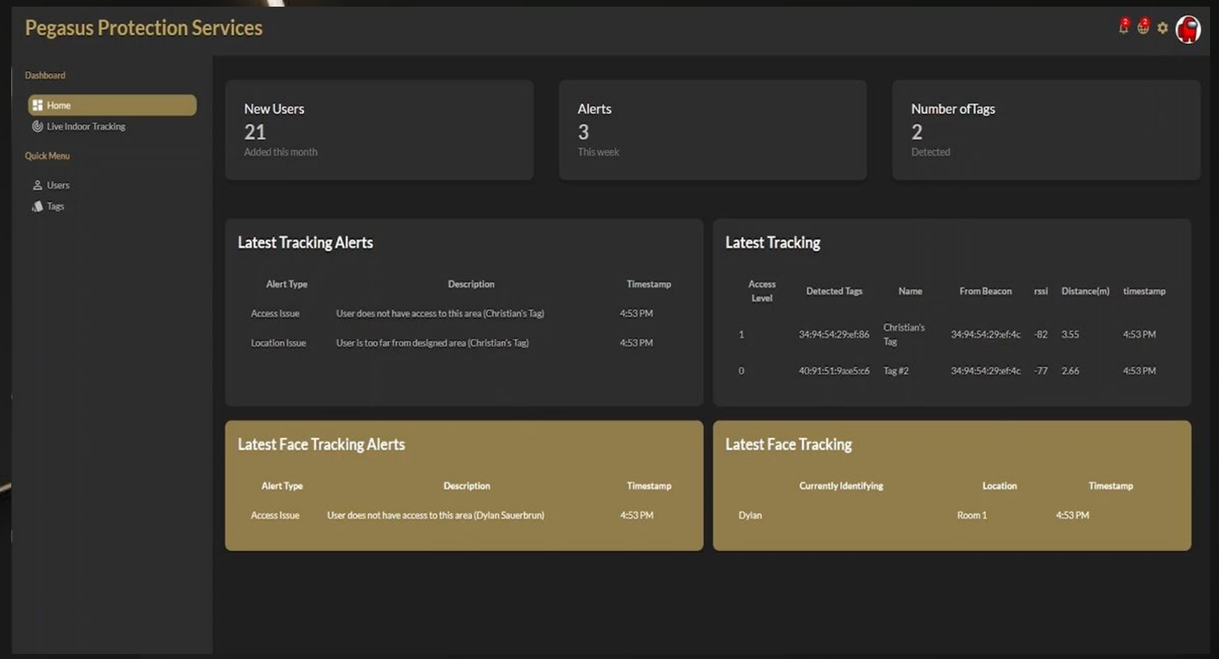
Web Application

- Allows user to interface with IoT so they can view and send data
- Display recently captured activity on admin dashboard
- Admin are able to assign tags to specific user
- Admin are able to upload images for our facial recognition system
- Using Mosquitto to acquire data sent via MQTT
  - MQTT protocol is used to carry messages from devices (ESP32)
- Using Socket.io to acquire real time incoming data from our Face Recognition program
- Frontend : React
- Backend : Node.js/Express

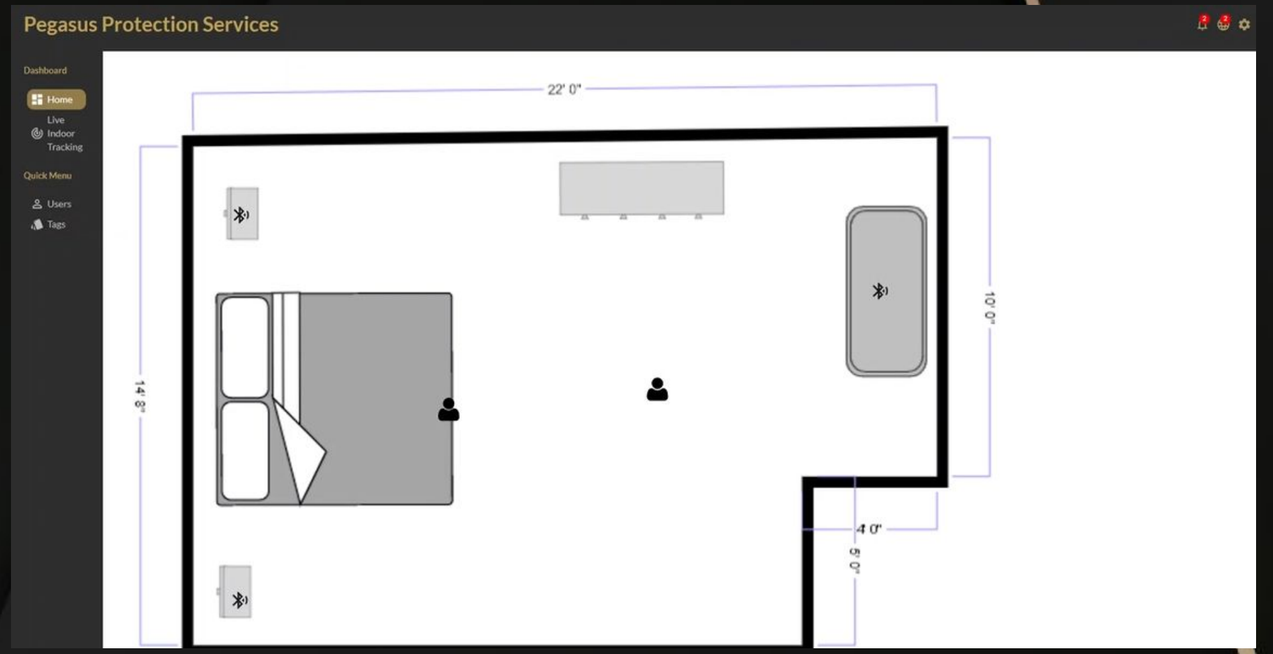
System Flowchart



Web Application



Admin Dashboard



Indoor Tracking

Workload Distribution

P- Primary Focus
S- Secondary Focus

	Members			
Project Work	Christian	Dylan	Aundre	Isaiah
Web-App	P	S		
PCB Design	P		P	
Beacon Firmware			S	P
IoT	S	S		P
Facial Recognition Training		P		
BLE indoor tracking software			P	S

Project Budget and Financing

Parts	Cost
3x 38 pin ESP32 Dev Board	$23.30
Arducam raspberry pi Camera module v2	$51.35
Nvidia Jetson Nano 2GB	$60
5x 32 pin ESP32 Dev board	$41.59
5x power supply PCBs	$11.28
Noctua NF-A4 Fan for jetson nano	$15
Pan-tilt servo platform	$29.01
Circuit Components	~$20
Total Estimated Cost	$251.53

Successes

- 2nd Gen PCB design size requirement met
- 2nd Gen PCB design works as either Beacon/Tag
- All components able to communicate wirelessly
- Achieved sufficient accuracy with Indoor tracking
- Camera can reliably detect faces in frames
- Recognition is accurate on faces that are provided in the training images
- Successfully able to display alerts based on indoor tracking system and computer vision system

What we could have done better

- Obtain more accurate location
  - Speeding up beacon transmission
  - Using stronger Antenna rather than ESP module
  - Ultra-Wideband
- Pick better MCU for Computer Vision Component
  - Nvidia Jetson Nano 4GB
- Pan-tilt-zoom implementation
  - Provide more field of view for face recognition
- Data Storage
  - Keeping tracking of previous alert datas
  - Video feed into cloud for later review

Questions?