

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

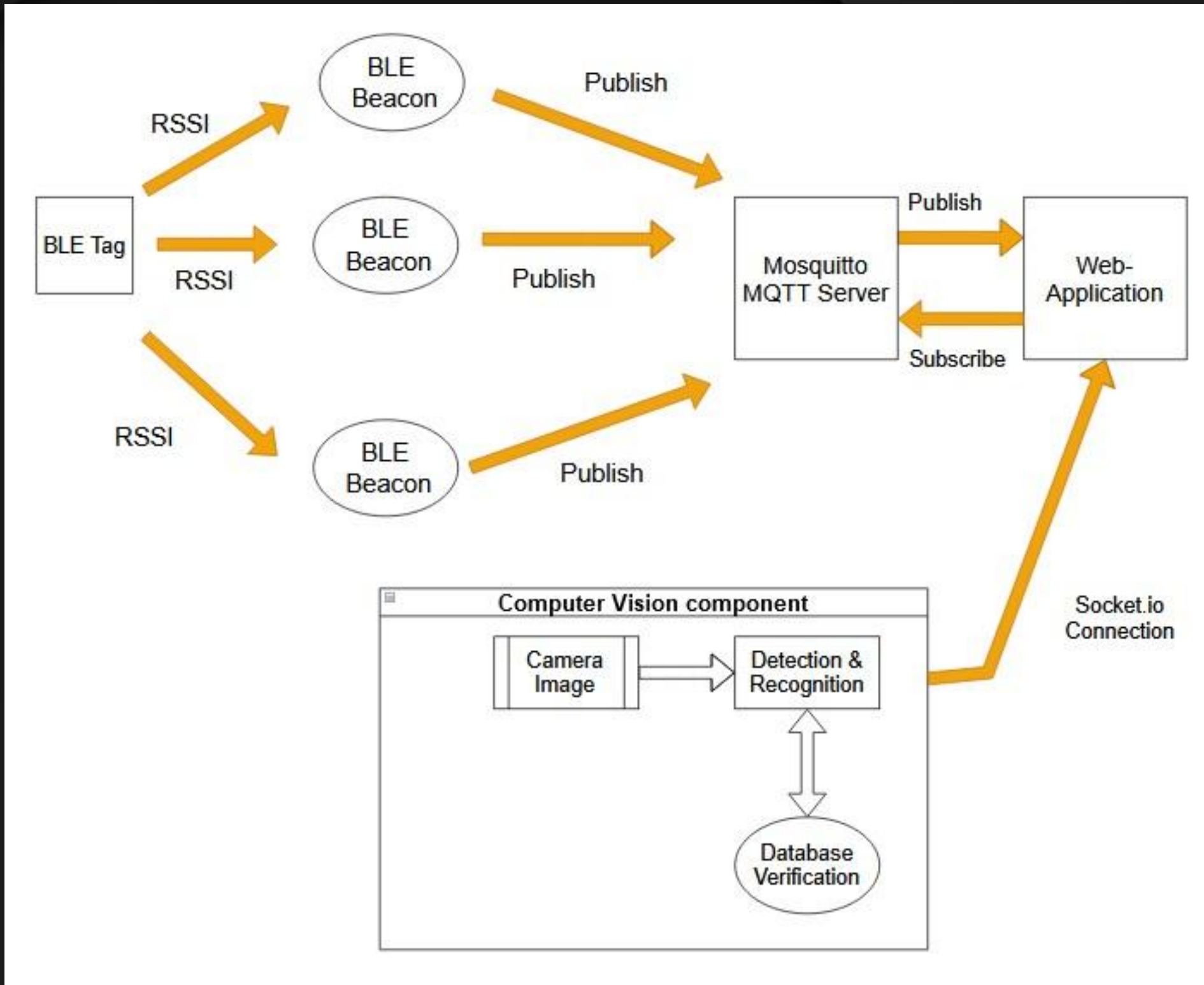


Security Infraction Alerts



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 too 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



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



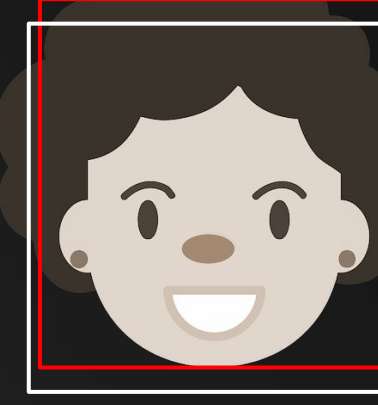
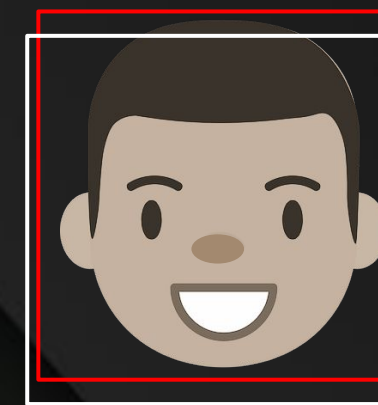
MicroController	Nvidia Jetson Nano
CPU	Quad-core ARM® A57 @ 1.43 GHz
GPU	128-core NVIDIA Maxwell
Memory	2 GB 64-bit LPDDR4 25.6 GB/s
I/O ports	1x USB 3.0 Type A, 2x USB 2.0 Type A, USB 2.0 Micro-B, 1x MIPI CSI-2 connector 1x HDMI

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

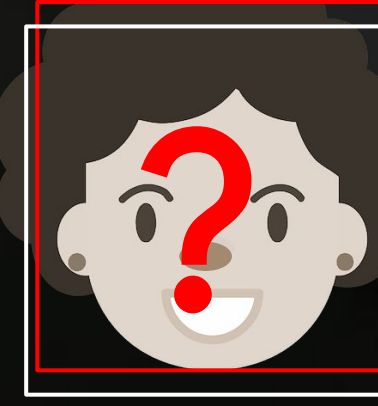
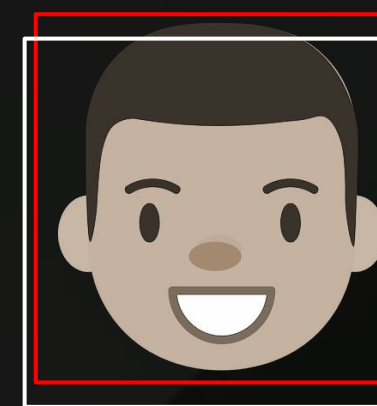
Computer Vision Overview



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
 - **O**pen-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

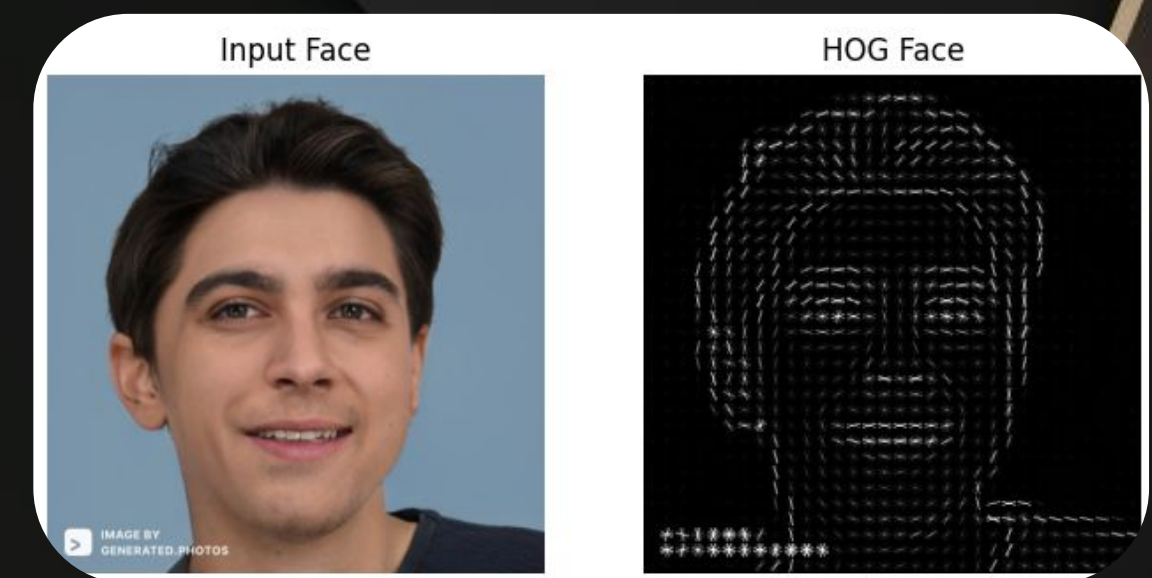
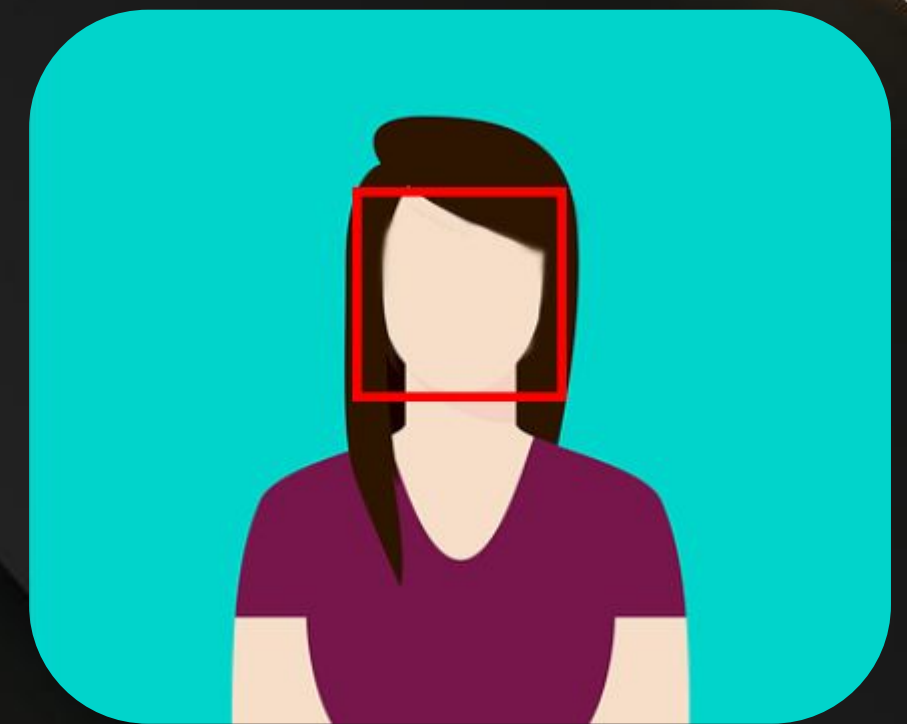


OpenCV



Classifying the images: a face or not?

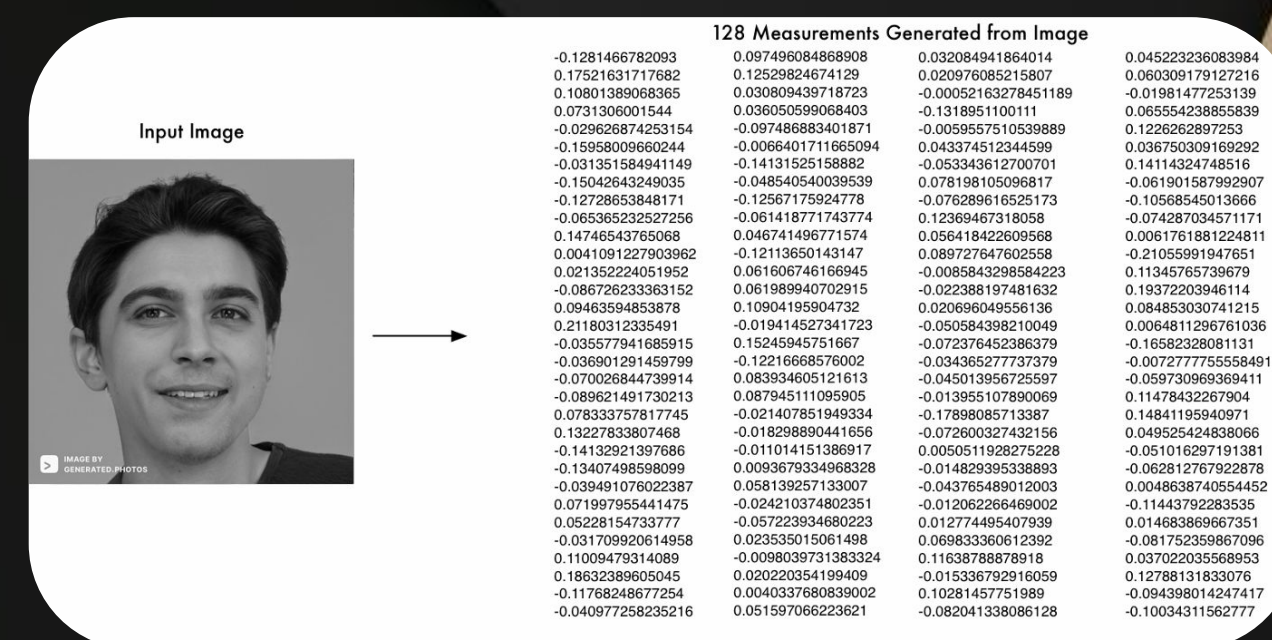
- Face detection
 - Used for locating faces in images
 - Classifies an image as containing a face or not
 - If a face is found, proceed to identifying
 - Saves us computational resources (don't have to always try to recognize a face in continuous streaming images)
- On the face detection front, we utilized 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



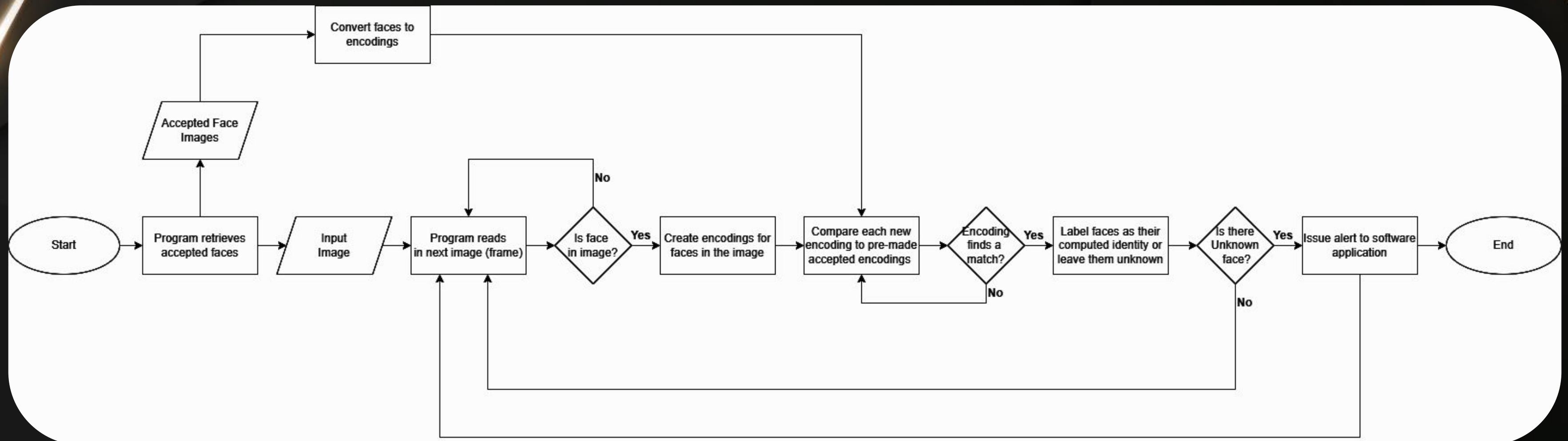
Face represented in encoding

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.



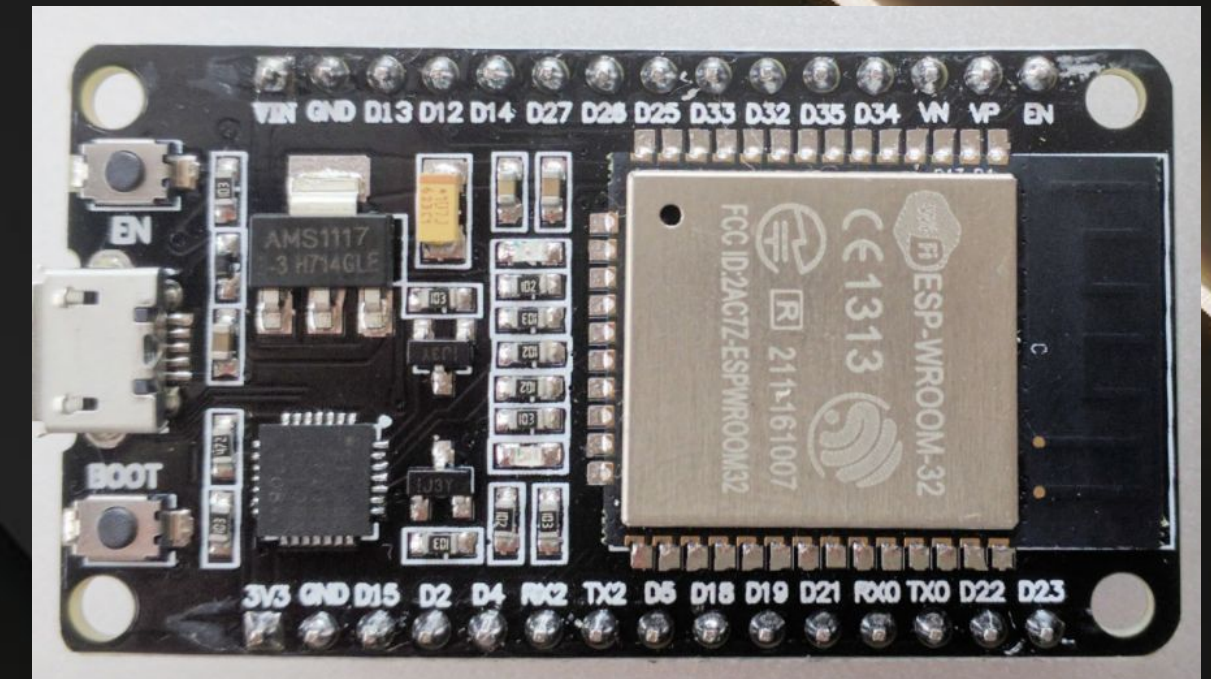
Communication Method Selection

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	Espressif	Espressif	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)

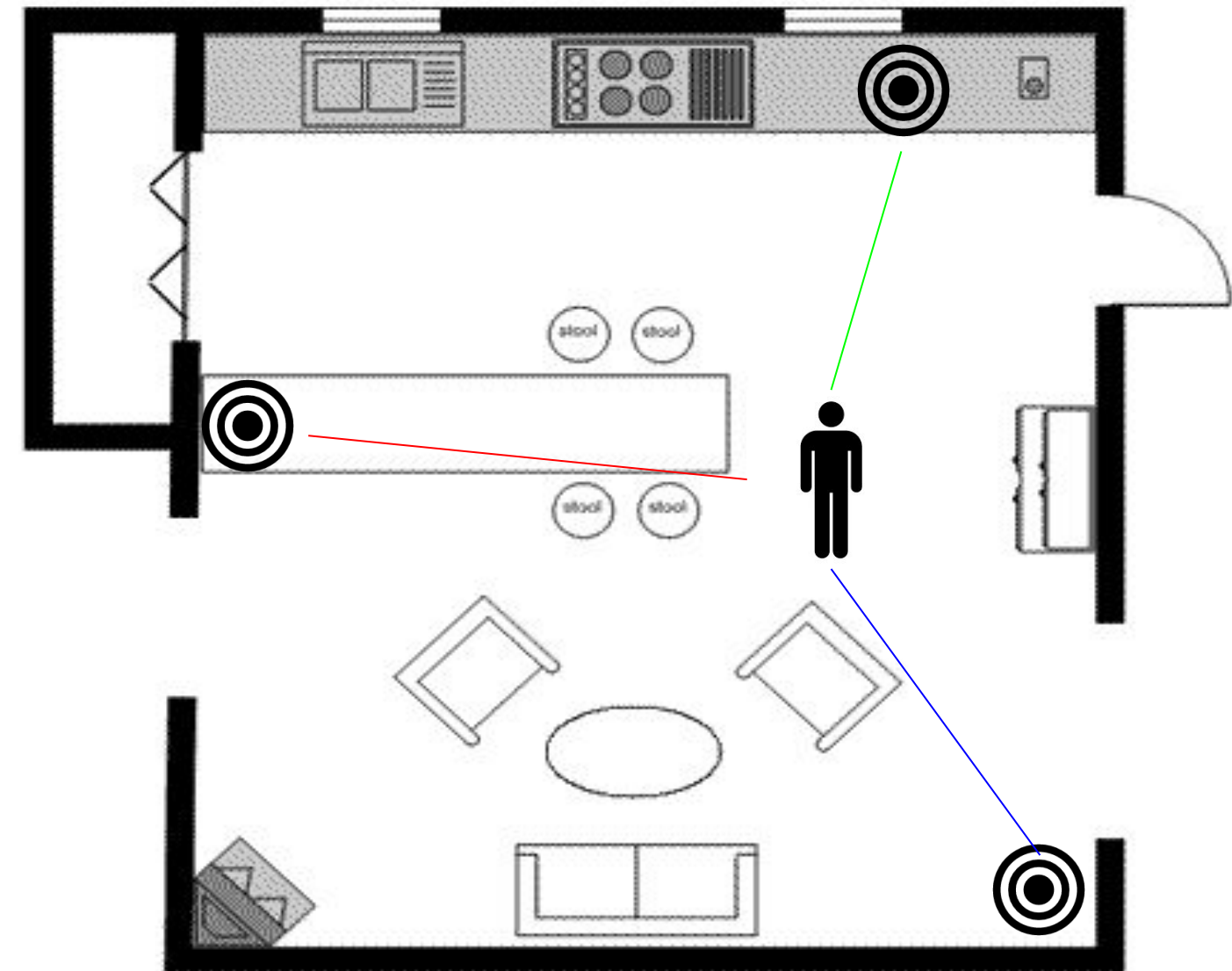
Web-App

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

$RSSI(d) = -10n \log(d) - C$

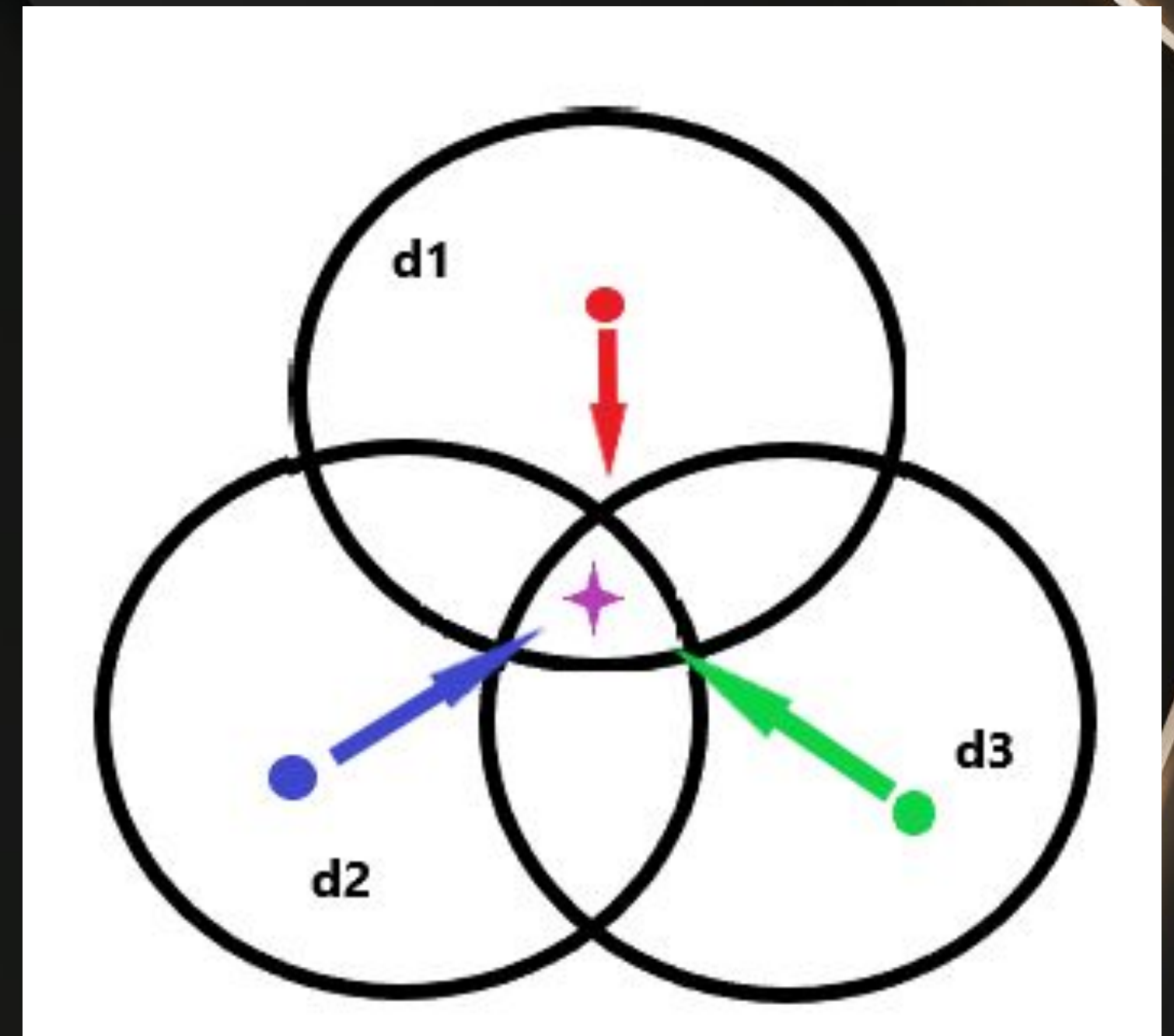
Point Estimation:

$d1 = (x_1, y_1)$

$d2 = (x_2, y_2)$

$d3 = (x_3, y_3)$

$d0 = (x_0, y_0) = \min \|AX^* - B\|^2$ (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)

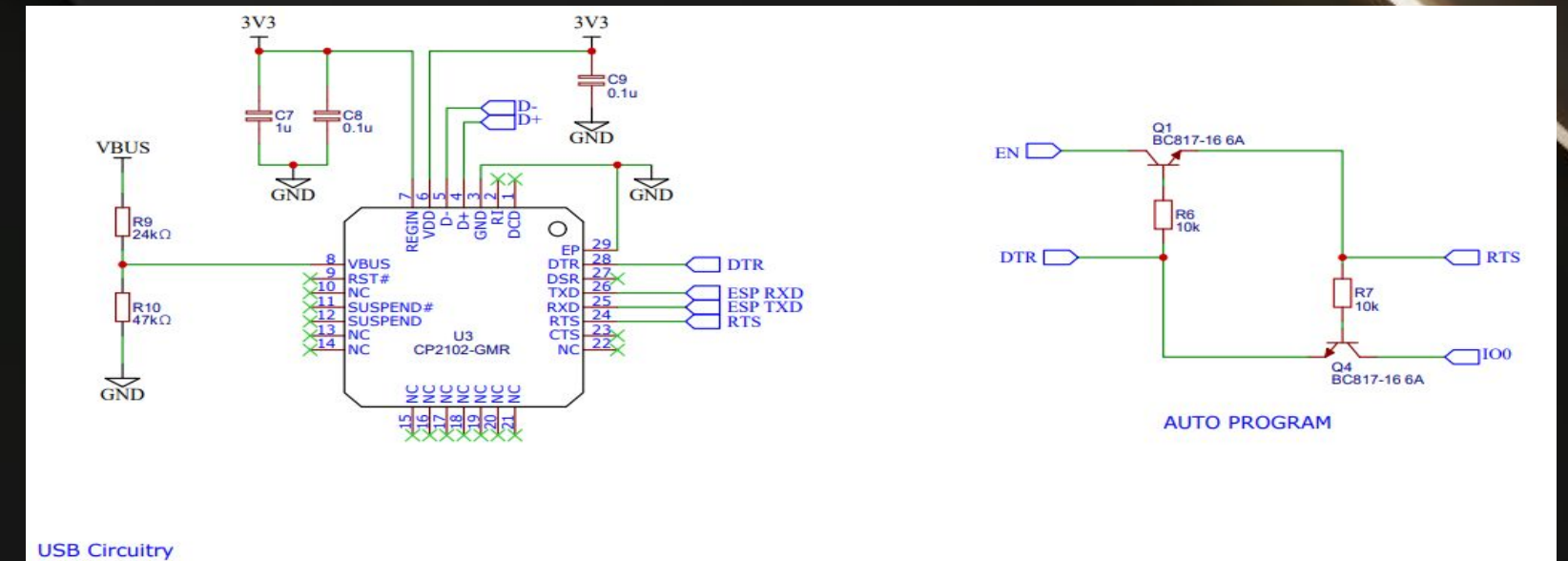
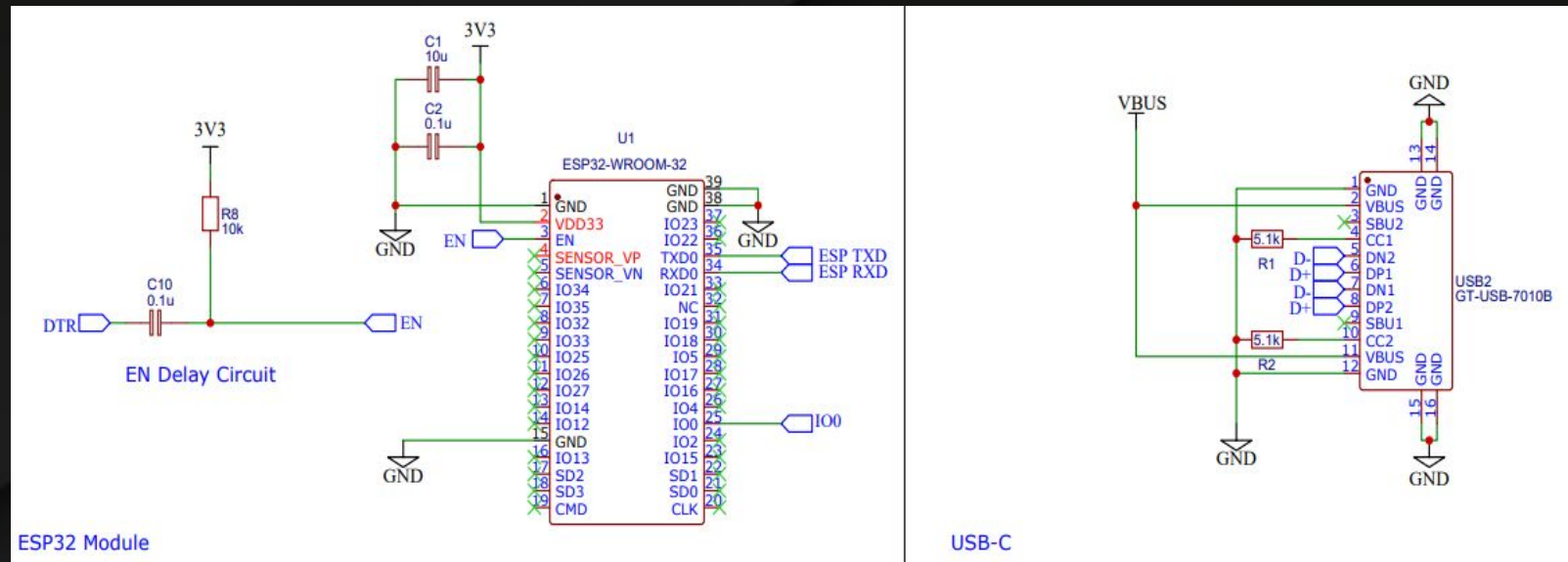
Previous Considerations(Wireless)

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.

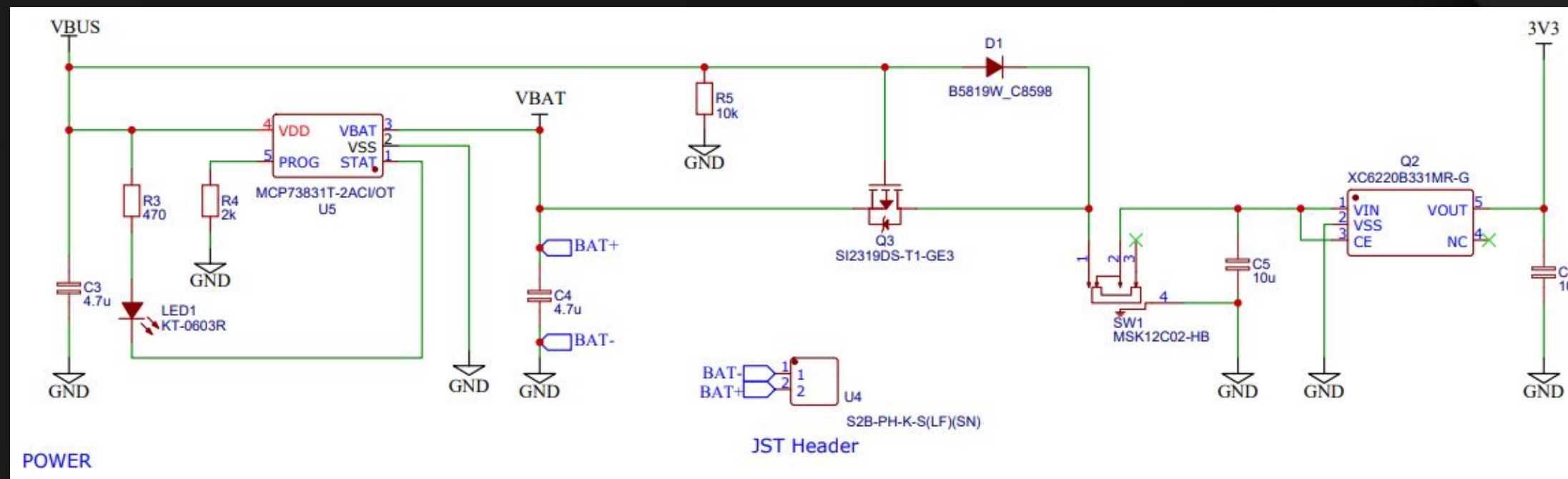
UWB - Ultra Wide Band Technology considered for tracking indoors over a larger area, but deemed less beneficial than BLE

Final PCB design



ESP32 Module + USBC Component

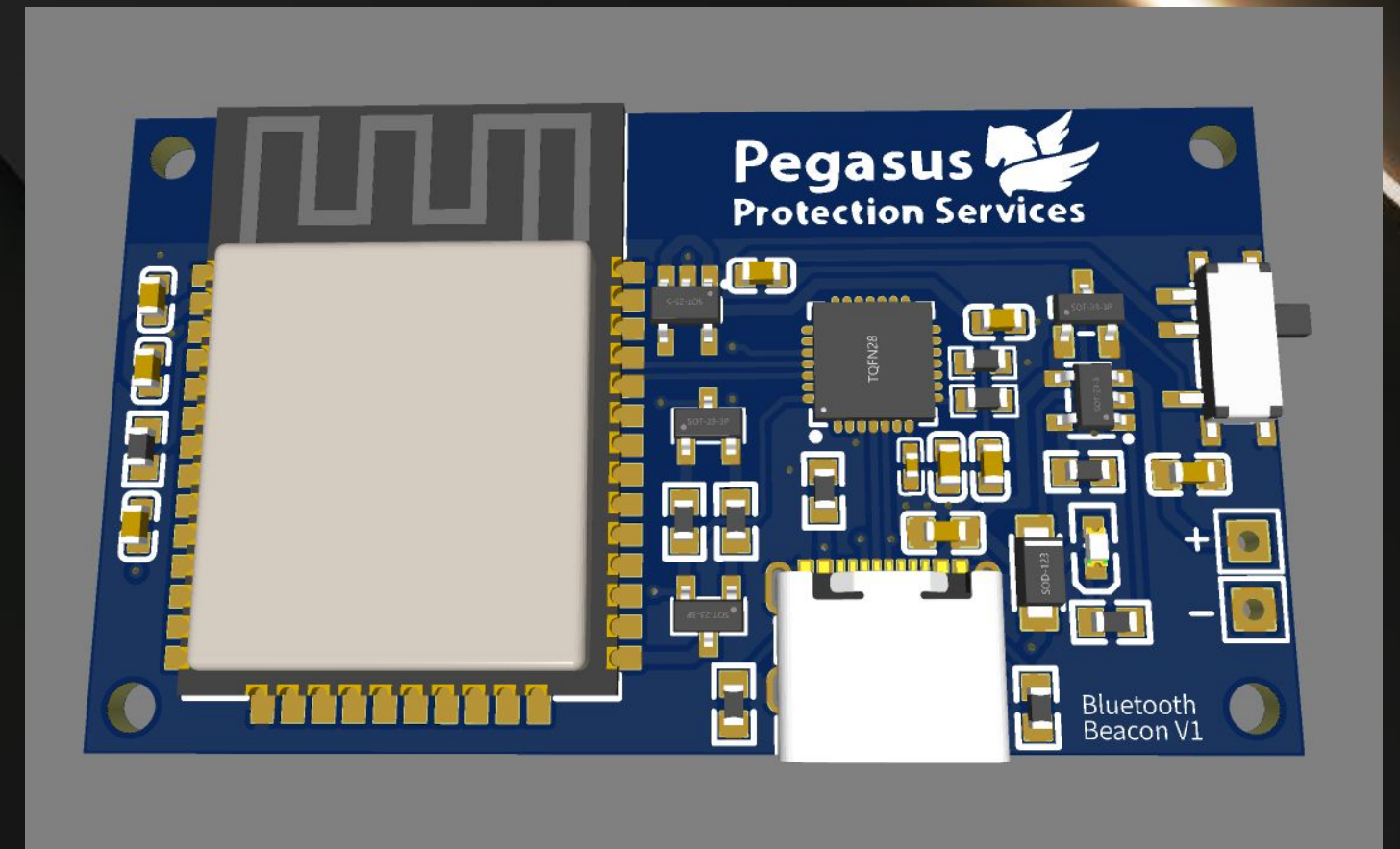
USB Circuitry + Auto Program



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)

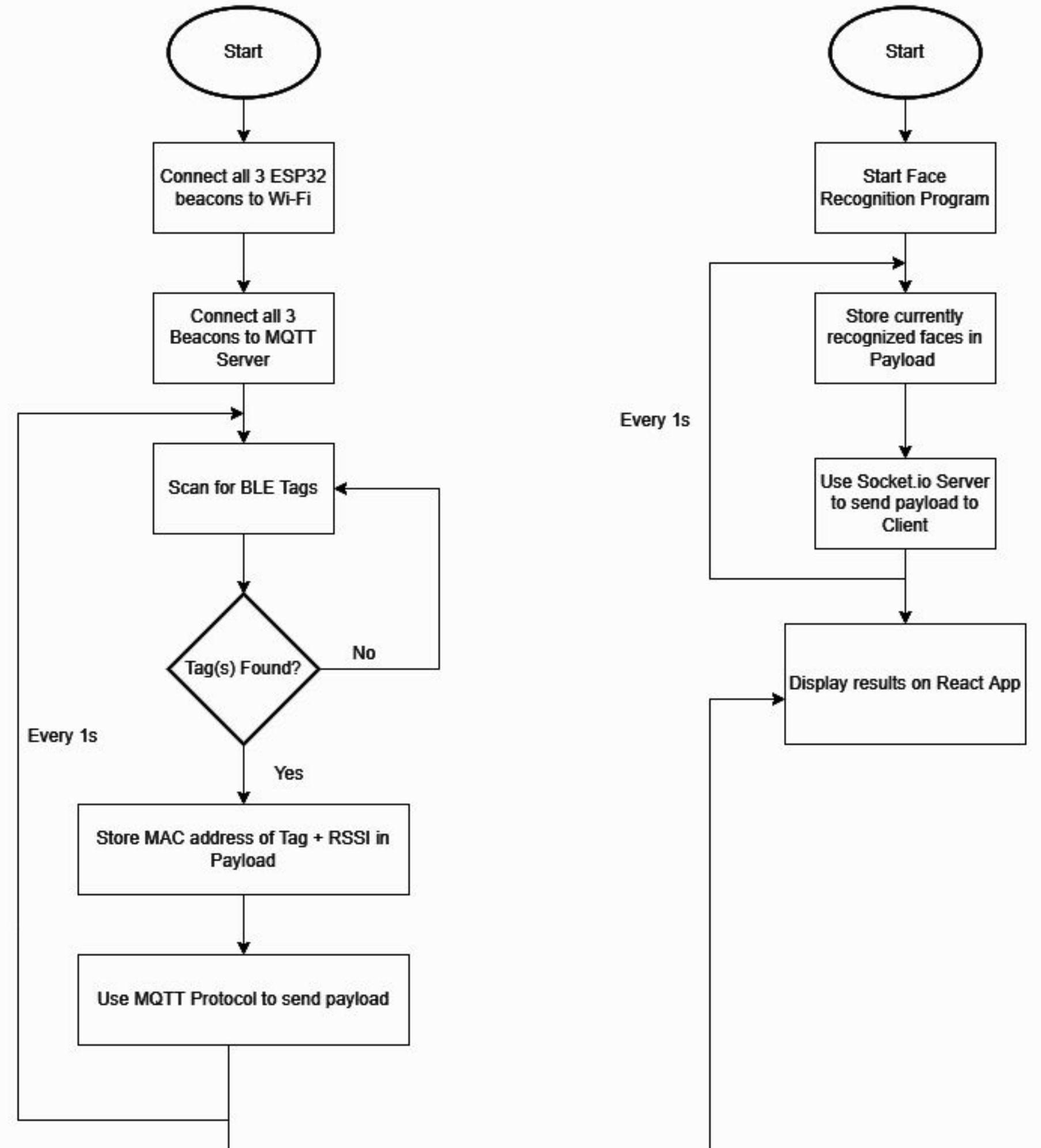


Web Application

- 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

Pegasus Protection Services

Dashboard

- Home
- Live Indoor Tracking

Quick Menu

- Users
- Tags

New Users: 21 (Added this month)

Alerts: 3 (This week)

Number of Tags: 2 (Detected)

Latest Tracking Alerts

Alert Type	Description	Timestamp
Access Issue	User does not have access to this area (Christian's Tag)	4:53 PM
Location Issue	User is too far from designed area (Christian's Tag)	4:53 PM

Latest Tracking

Access Level	Detected Tags	Name	From Beacon	rssi	Distance(m)	timestamp
1	34-94-54-29-ef-f86	Christian's Tag	34-94-54-29-ef-f4c	-82	3.55	4:53 PM
0	40-91-51-92-ee-5cc6	Tag #2	34-94-54-29-ef-f4c	-77	2.66	4:53 PM

Latest Face Tracking Alerts

Alert Type	Description	Timestamp
Access Issue	User does not have access to this area (Dylan Sauerbrun)	4:53 PM

Latest Face Tracking

Currently Identifying	Location	Timestamp
Dylan	Room 1	4:53 PM

Admin Dashboard

Pegasus Protection Services

Dashboard

- Home
- Live Indoor Tracking

Quick Menu

- Users
- Tags

22' 0"

14' 0"

10' 0"

4' 0"

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?