IOT based Pet tracker

Max Kotas, Timothy Thompson, Darren Lefever

FINAL REPORT

2 December 2024

FINAL REPORT
FOR
IOT based Pet Tracker

TEAM <34>

APPROVED BY:

_____
Project Leader Date

_____
Prof. Kalafatis Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
A	02/23/2024	Darren Lefever		Draft Release/ Basic Con Ops, FSR, ICD
B	9/25/2024	Darren Lefever		Revision
C	12/2/2024	Max Kotas		Final

IOT based Pet tracker

Max Kotas, Timothy Thompson, Darren Lefever

CONCEPT OF OPERATIONS

2 December 2024

Concept of Operations
for
IOT based Pet Tracker

Team <34>

Approved by:

_____
Project Leader Date

_____
Prof. Kalafatis Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
A	02/23/2024	Darren Lefever		Draft Release
B	9/25/2024	Darren Lefever		Revision
C	12/2/2024	Max Kotas		Final

Table of Contents

Table of Contents III
List of Tables IV
No table of figures entries found. IV
List of Figures V
No table of figures entries found. V
1. Executive Summary 1
2. Introduction 2
2.1. Background 2
2.2. Overview 2
2.3. Referenced Documents and Standards 2
3. Operating Concept 3
3.1. Scope 3
3.2. Operational Description and Constraints 3
3.3. System Description 3
3.4. Modes of Operations 3
3.5. Users 3
3.6. Support 3
4. Scenario(s) 4
4.1. Pet Safety Monitoring 4
4.2. Intentional Exiting of Safe Zone 4
5. Analysis 4
5.1. Summary of Proposed Improvements 4
5.2. Disadvantages and Limitations 4
5.3. Alternatives 4
5.4. Impact 4

List of Tables

List of Figures

1. Executive Summary

The Pet Collar Watchdog revolutionizes pet safety with its cutting-edge hardware and intelligent software features. Integrated into a sleek collar design, this innovative solution has a GPS module, live camera feed, buzzer, and communication modules, creating a comprehensive pet tracking system. By continuously monitoring pets' coordinates, the collar ensures they remain within the owner's predefined safe area, promptly alerting owners via email or phone alerts if any breach occurs. Notably, the system goes beyond mere tracking by leveraging sophisticated machine learning algorithms. These algorithms analyze pet movement data, dynamically suggesting optimized safe areas through a user-friendly interface. This proactive approach allows owners to adapt to their pet's behavior and environment, enhancing overall safety and peace of mind. In essence, the Pet Collar Watchdog is not just a safety tool—it's a companion for pet owners, offering unparalleled convenience, security, and connectivity. With its advanced features and intuitive design, it sets a new standard for pet care in the modern age.

2. Introduction

2.1. Background

Currently, pet owners face concerns for their pets safety at their own leisure. Traditional pet containment methods have limitations, prompting the development of the IOT-based pet tracking system. This project aims to enhance pet owner's options when ensuring their pet's safety by employing current generation hardware and intelligent software features. This collar addresses the shortcomings of conventional pet containment and surveillance systems.

2.2. Overview

The IOT-based Pet Tracker project aims to provide a sophisticated solution for pet owners looking to enhance their pet's safety, implemented into a sleek collar design. This collar will include features such as GPS, live camera feed, buzzer, and communication modules. These combined with a receiver system and phone app will work together to ensure pets stay within predefined safe areas as well as employing machine learning techniques to learn pet behavior.

2.3. Referenced Documents and Standards

The development and implementation of this project will adhere to relevant institutional and industrial guidelines. We will need to consider established protocols for radio and electronic communications, and GPS technology. Referenced documents and standards will be continuously reviewed and updated in future revisions to this document.

3. Operating Concept

3.1. Scope

The scope of the Pet Collar Watchdog project encompasses the development, deployment, and support of an advanced pet safety collar/system. This includes designing and manufacturing the receiver as well as the feature modules (GPS, video feed, buzzer) which are integrated into the pet collar. An intuitive software application for tracking, notifications, safe area setup and user interface interaction will be created. Additionally, the scope extends to providing ongoing support and maintenance to ensure the system's optimal functionality over time.

3.2. Operational Description and Constraints

The Pet Collar Watchdog operates as a comprehensive pet safety solution, employing a combination of hardware and software components to monitor and safeguard pets. The system is designed to continuously track the location and activities of pets using GPS technology integrated into the pet collar. Additionally, a camera captures live video footage of the pet's surroundings, providing visual confirmation and insight into the pet's environment.

When a pet strays from predefined safe areas, the system triggers alerts to notify the pet owner via email. Simultaneously, the collar activates an audible buzzer to attract attention and facilitate locating the pet. Furthermore, the collar streams live video footage when within range of an external home-mounted receiver, allowing pet owners to remotely monitor their pet's activities in real-time.

Constraints:

- Power Consumption: The system relies on battery power, necessitating careful consideration of power consumption to ensure continuous operation and prolonged battery life.
- Signal Strength: The effectiveness of GPS and communication modules may be influenced by factors such as signal strength and environmental conditions. Signal reception may vary depending on the location, terrain, and surrounding obstacles, potentially impacting the accuracy and reliability of the system.
- Data Privacy: The collection and transmission of pet location and video data raise privacy concerns. It is essential to implement robust security measures to safeguard pet owner privacy and comply with data protection regulations.
- Range Limitations: The communication range between the pet collar and the external receiver is limited by technical constraints. Users should be aware of the maximum range and potential limitations when monitoring their pets' activities to ensure effective tracking and notification capabilities.

3.3. System Description

User interface and database subsystem

- A graphical user interface (GUI) that allows the user to define a safe area for their pet and then saves that information on a database
- Using a gps module on the dog collar, coordinates are received and compared to the safe area
- If the collar exits the predefined safe zone, three features are activated:
    - A notification is sent to the user, saying the collar has left the safe zone
    - An option to view a live video feed is provided
    - A audible buzzer is automatically activated and will not shut off unless the collar returns to the safe zone or it is manually deactivated by the user in the GUI
- After collecting data on the different locations the collar exits the safe zone, machine learning is used to determine the likely exit points and train itself to predict potentially new and refined safe zones that can be setup in the GUI

Dog collar subsystem

- A camera and light module affixed to the front of the dog collar. This will be automatically activated when the pet leaves the safe area.
- A buzzer will be used to alert the owner to the pet's location. This buzzer will also activate automatically in case of an escape.
- Power module will include a small 12V battery and voltage step down converter. Voltage meter used to activate low battery functionality.
- Cellular antenna used for database and receiver connectivity. Antenna will be positioned directly upward on the dog collar for maximum signal strength

Receiver subsystem

- A wifi module will be connected to a single board computer, allowing:
    - constant downloading of gps coordinate data from the collar and uploading of this data to an offsite server for processing
    - Video streaming throughput from collar to server
    - Direct user interaction with the collar, such as activation of the locating buzzer, relayed through the receiver and sent directly to the collar subsystem.
- Power to the receiver module will be provided through a wired connection from power outlet to usb-c 5V
- Software libraries for data processing will include OpenCV and I/O handling, which will all be used for two way communication with the collar

3.4. Modes of Operations

The system operates in two main modes: Alert mode and off/disarmed 'mode'. In alert mode, the system continuously monitors pet coordinates and updates the user on their whereabouts and notifies the user if the pet leaves the safe area while activating an audible buzzer and providing option for live video. The off/disarmed 'mode', although not technically a mode, this feature just allows the user to turn off/on each feature individually to what state they desire.

3.5. Users

The primary users of the IoT-based Pet Tracker are pet owners seeking to ensure the safety and security of their pets. Users may vary in their level of technical expertise, but the system is designed to be intuitive and easy to use for all users. Additionally, pet care professionals and animal shelters may also benefit from the system's capabilities.

3.6. Support

Support for the IoT-based Pet Tracker includes user manuals and technical support to assist users in setting up and using the system. Additionally, regular updates and maintenance ensure the system remains up-to-date and functional.

4. Scenario(s)

4.1. Pet Safety Monitoring

A pet owner sets up safe areas for their pet using the IoT-based Pet Tracker website. They attach the GPS-enabled collar to their pet and set it to default (alert mode). Throughout the day, the system continuously monitors the pet's coordinates and updates the user on their whereabouts. If the pet strays from a predefined safe area, the system triggers the collar's features and sends notifications to alert the user, allowing for immediate intervention.

4.2. Intentional Exiting of Safe zone

A pet owner has a predefined safe area already set up, and they have to exit the safe area with their pet. The user disables the alert features (live video, buzzer, tracking, and notifications) to allow for a smooth intentional exit of the safe zone. The features can be turned off/on whenever the user pleases. (range according to receiver)

5. Analysis

5.1. Summary of Proposed Improvements

- Enhanced Pet Safety: The IoT-based Pet Tracker will offer an advanced level of pet safety by continuously tracking pet coordinates and automatically activating features such as audible buzzers and live video streaming when the pet exits predefined safe areas. This approach will ensure timely intervention and increase the likelihood of quickly locating lost pets.
- User-Friendly Interface: The system will feature a user-friendly interface that allows pet owners to define safe areas, receive notifications, and manually control collar features through a simple and intuitive graphical interface.
- Machine Learning Optimization: By leveraging machine learning algorithms, the system continuously analyzes pet movement data to identify patterns and predict potential exit points from safe areas. This adaptive approach allows the system to dynamically adjust safe area definitions, improving accuracy and effectiveness over time.

5.2. Disadvantages and Limitations

- Power Consumption: Continuous operation of GPS modules, communication modules, and video streaming may result in high power consumption, potentially reducing the collar's battery life. Pet owners may need to recharge or replace collar batteries frequently, especially during extended outdoor activities.
- Privacy Concerns: The use of live video streaming raises privacy concerns, as it involves capturing and transmitting visual data of the pet's surroundings. Pet owners must consider the implications of recording video footage in public or private spaces and ensure compliance with applicable privacy regulations.
- Dependency on Technology: The effectiveness of the IoT-based Pet Tracker relies heavily on technology, including GPS modules, communication protocols, and

software algorithms. Any technical issues or malfunctions in these components could compromise the system's functionality and reliability.

5.3. Alternatives

- Traditional Pet Containment Methods: Alternative solutions to the IoT-based Pet Tracker include traditional pet containment methods such as fences, leashes, and physical barriers. While these methods are straightforward and familiar to pet owners, they lack the advanced features and real-time monitoring capabilities offered by the IoT-based system.
- GPS-Only Tracking Devices: Some existing pet tracking devices rely solely on GPS technology to monitor pet locations. While these devices offer basic tracking functionality, they may lack features such as live video streaming and machine learning-based safe area optimization, limiting effectiveness in ensuring pet safety.
- Radio Frequency Identification (RFID) Systems: RFID-based pet tracking systems use radio frequency signals to identify and track pets within a defined area. While RFID systems can be effective for indoor tracking or short-range monitoring, they may not provide the same level of accuracy or real-time tracking capabilities as GPS-based systems.

5.4. Impact

- Societal Impact: The widespread adoption of pet tracking technologies has the potential to positively impact society by reducing the number of missing pets and promoting responsible pet ownership. By providing pet owners with tools to monitor their pets, the IoT-based Pet Tracker helps create stronger bonds between pets and their owners and contributes to safer communities.
- Ethical Concerns: Ethical considerations surrounding pet tracking technologies include issues related to pet privacy, data security, and consent. Pet owners must prioritize the welfare and rights of their pets while using tracking devices and ensure that their use complies with ethical principles and legal regulations. Additionally, pet tracking systems should incorporate important security measures to protect sensitive pet data from unauthorized access/misuse.

IOT based Pet tracker

Max Kotas, Timothy Thompson, Darren Lefever

FUNCTIONAL SYSTEM REQUIREMENTS

2 December 2024

Functional System Requirements
for
IOT based Pet Tracker

Prepared by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
A	2/23/2024	Darren Lefever		Draft Release
B	9/25/2024	Darren Lefever		Revision
C	12/2/2024	Max Kotas		Final

Table of Contents

1. Introduction **5**
1.1. Purpose and Scope 5
1.2. Responsibility and Change Authority 6
2. Applicable and Reference Documents **7**
2.1. Applicable Documents 7
2.2. Reference Documents 7
2.3. Order of Precedence 8
3. Requirements **9**
3.1. System Definition 9
3.2. Characteristics 10
3.2.1. Functional / Performance Requirements 10
3.2.2. Physical Characteristics 10
3.2.3. Electrical Characteristics 11
3.2.4. Environmental Requirements 12
3.2.5. Failure Detection 12
4. Support Requirements **12**
Appendix A: Acronyms and Abbreviations **13**
Appendix B: Definition of Terms **14**

List of Tables

Table 1: Subsystem Leads 2

List of Figures

Figure 1. Project Conceptual Image **1**
Figure 2. Block Diagram of System **4**

1. Introduction

1.1. Purpose and Scope

The scope of this project encompasses the development, deployment, and support of the IOT based Pet Tracker to increase pet safety and user ease of mind. This includes designing and manufacturing both the hardware components, such as the receiver and feature modules (GPS, video feed, buzzer), integrated into the pet collar, as well as the software application for tracking, notifications, safe area setup, and user interface interaction. Additionally, ongoing support and maintenance will be provided to ensure the system's optimal functionality over time.

Figure 1. Project Conceptual Image

1.2. Responsibility and Change Authority

We are each responsible for verifying all requirements within our assigned subsystems are met. These requirements can only be changed with the approval from our sponsor and Professor Stavros Kalafatis.

Subsystem	Responsibility
Dog Collar Hardware	Timothy Thompson
Receiver Communications	Max Kotas
Software (UI/Machine Learning)	Darren Lefever

Table 1: Subsystem Leads

2. Applicable and Reference Documents

2.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	Revision/Release Date	Document Title
DOD-HDBK-791	3/17/1998	Maintainability Design Techniques Metric
MIL-HDBK-217	Revision F – 2/28/1995	Reliability Prediction of Electronic Equipment
MIL-HDBK-263	Revision B – 7/31/1994	Electrostatic Discharge Control Handbook for Protection of Electrical and Electronic Parts, Assemblies and Equipment
MIL-HDBK-338	Revision B – 10/1/1998	Electronic Reliability Design
ANSI C119.6-2011	5/5/2011	American National Standard for Electrical Connectors

2.2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
N/A	2022-09-19 v0.5	ESP32S2SOLO2 Datasheet
N/A	2022 v2.0	Quectel EC25 Series Datasheet

2.3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

3. Requirements

3.1. System Definition

The IOT based Pet Tracker is designed to enhance pet safety and monitoring. By integrating hardware components like GPS, camera, and buzzer into a pet collar, and utilizing a receiver subsystem with a WiFi-connected single-board computer, the system enables continuous tracking and real-time communication. Accompanied with a user-friendly interface for defining safe areas, triggering alerts if the pet strays, and offering live video feeds. Additionally, machine learning algorithms analyze pet behavior to predict and refine safe zones over time. This comprehensive solution ensures pet owners can track, monitor, and safeguard their pets effectively.

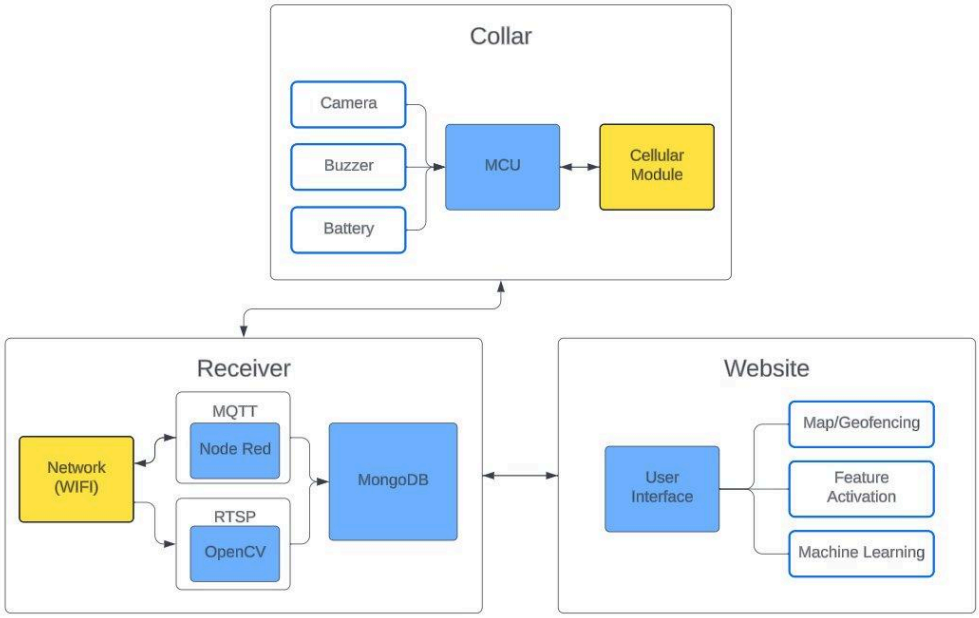


Figure 2: Block Diagram of System

Figure 2. Block Diagram of System

The Pet Collar Subsystem is at the heart of the project, containing a collar equipped with essential hardware components such as a GPS module, camera, and buzzer modules. These components work together to continuously track the pet's location and provide real-time alerts if the pet strays from predefined safe areas. The GPS module ensures accurate positioning, while the camera captures live video footage to provide visual confirmation of the pet's surroundings. Additionally, the buzzer serves as audible alerts to notify both the pet and the owner of any potential safety concerns.

The Receiver Subsystem acts as the intermediary between the dog collar and the user interface, facilitating communication and data exchange. It consists of a WiFi module connected to a single-board computer, which downloads GPS coordinate data from the collar and uploads it to an offsite server for processing. This subsystem enables two-way

communication, allowing users to remotely activate features such as the buzzer or access live video feeds from the collar.

The User Interface and ML Subsystem provide users with a user-friendly platform for managing and monitoring their pets' safety. The graphical user interface (GUI) allows users to define safe areas for their pets and stores these definitions in a database for future reference. When the pet's GPS coordinates are received, the system compares them to the predefined safe areas. If the pet exits a safe zone, the system triggers notifications to the user and activates features such as live video streaming and audible alerts. Lastly, using stored GPS coordinate data, a machine learning model will predict a more refined safe area to better fit the movement of the pet.

3.2. Characteristics

3.2.1. Functional / Performance Requirements

3.2.1.1. GPS Accuracy

The Pet Tracker system shall achieve a GPS location accuracy of at least 5 meters under normal operating conditions, ensuring precise tracking of the pet's whereabouts..

Rationale: This is the main system requirement. This is the main function of the system

3.2.1.2. Real-time Tracking

The system shall provide real-time tracking updates with a maximum latency of 10 seconds, allowing pet owners to monitor their pet's location promptly.

3.2.1.3. Time to Alert

With an uninterrupted Wi-Fi and power source, the entire system shall take no more than a 1 min to provide the user an alert ready for review

3.2.2. Physical Characteristics

3.2.2.1. Weight

The weight of the pet tracker device shall be less than 0.75lbs to avoid causing discomfort or hindering the pet's movements while wearing the collar.

Rationale: Allow for the pet to be comfortable while in use

3.2.2.2. Pet Collar Size

The pet tracker system shall have dimensions not exceeding 2 inches in width, 1.5 inch in height, and 0.5 inches in thickness, ensuring it is compact and lightweight for comfortable attachment to the pet's collar.

Rationale: Allow for the pet to be comfortable while in use

3.2.3. Electrical Characteristics

3.2.3.1. Battery Life

The device shall have a minimum battery life of 3 hrs under normal usage conditions, ensuring reliable tracking without frequent recharging.

Rationale: User shall be able to know the operation time of one charge

3.2.3.1.1 Power Consumption

The pet tracker device shall have a maximum current pull of 500 mA at 3.3V. This limit shall not be exceeded for power conservation purposes.

3.2.3.2. Outputs

3.2.3.2.1 Alert Notifications

The system shall provide audible and visual alerts to the pet owner via a smartphone application in case the pet leaves the predefined safe zone, ensuring timely notification and action.

Rationale: This allows user to be quickly alerted when the pet leaves the area

3.2.3.2.2 Data Logging

The system shall log location data periodically, storing it in a database enabling that data to be used later for machine learning.

Rationale: shall be used for geofencing detection and machine learning

3.2.3.2.3 Live Video Output

The pet collar shall include a camera module to support live video feed when alerted about a safe area breach

3.2.3.2.4 Buzzer Output

The pet collar shall include a buzzer module to output an audible sound

Rationale: User shall hear if safe zone is breached if in close proximity

3.2.3.3. Connectors

The Pet collar System shall follow the American National Standard for Electrical Connectors ANSI C119.6-2011.

3.2.3.4. Wiring

The Pet collar shall follow the guidelines set forth in the National Electric Code regarding electrical wiring

3.2.4. Environmental Requirements

The Pet Tracker shall be designed to withstand and operate in the environments and laboratory tests specified in the following section.

3.2.4.1. Durable

The Pet Tracker shall be designed to withstand everyday pet movement/activities

Rationale: The collar shall last while the pets goes about its daily activities

3.2.5. Failure Detection

To detect failures in the IoT pet Collar project effectively, implement a comprehensive monitoring system across all components, This includes using the Raspberry Pi's System monitoring tools for resource and temperature checks, regularly verifying the camera feed from the Arducam 5 MP modules, ensuring the reliable communication through the UART serial port integrity checks, testing WiFi connectivity, and monitoring battery levels for power supply consistency. Employing diagnostic tests on startup, maintaining detailed operation logs, and setting up alert mechanisms for identified issues will help catch and address failures promptly, ensuring the project's reliability and the safety of pets monitored by the system.

4. Support Requirements

Customers utilizing the pet tracker system will require a laptop or computer and a modern web browser, also the user must be able to have a WIFI connection and well power the receiver subsystem and be able to charge the pet collar system. The system package includes a comprehensive user manual, setup guide, and any necessary accessories for seamless installation. Technical support services are available, offering assistance with troubleshooting, operation guidance, and software updates. We highly encourage you to only use this product on only pets.

Appendix A: Acronyms and Abbreviations

BIT	Built-In Test
FOV	Field of View
GPS	Global Positioning System
GUI	Graphical User Interface
Hz	Hertz
ICD	Interface Control Document
kHz	Kilohertz (1,000 Hz)
LCD	Liquid Crystal Display
LED	Light-emitting Diode
mA	Milliamp
MHz	Megahertz (1,000,000 Hz)
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair
MQTT	Message Queuing Telemetry Transport
mW	Milliwatt
PCB	Printed Circuit Board
RMS	Root Mean Square
RTSP	Real Time Streaming Protocol
TBD	To Be Determined
TTL	Transistor-Transistor Logic
USB	Universal Serial Bus

Appendix B: Definition of Terms

IOT based Pet tracker

Max Kotas, Timothy Thompson, Darren Lefever

Interface Control Document
User interface, Receiver, Collar hardware

2 December 2024

Interface Control Document
FOR
IoT Pet Collar

Prepared by:

Max, Timothy, Darren
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
A	2/23/2024	Timothy Thompson		Draft Release
B	9/25/2024	Timothy Thompson		Revision
C	12/2/2024	Max Kotas		Final

Table of Contents

Table of Contents **IV**
List of Tables **V**
List of Figures **VI**
1. Overview **6**
2. References and Definitions **6**
2.1. References 6
2.2. Definitions 6
3. Physical Interface 6
3.1. Dimensions 6
3.1.1. Dimension of Collar 7
3.2. Mounting Locations 7
4. Thermal Interface **7**
5. Electrical Interface **7**
5.1. Primary Input Power 7
5.2. Signal Interfaces 7
5.3. Video Interface 7
5.4. User Control Interface 7
6. Communications / Device Interface Protocols **7**
6.1. Wireless Communications (WiFi) 8
6.2. Host Device 8
6.3. Video Interface 8
6.4. Device Peripheral Interface 8

List of Tables

Raspberry Pi

Component	Specification	Details
Raspberry Pi Zero 2 W	**Dimensions**	**65mm × 30mm**
	Weight	**16 g**
	CPU	**Broadcom BCM2710A1, quad-core 64-bit SoC (Arm Cortex-A53 @ 1GHz)**
	RAM	**512 MB LPDDR2**
	Connectivity	**2.4GHz IEEE 802.11b/g/n wireless LAN, Bluetooth 4.2, BLE, onboard antenna**
	Ports	**mini HDMI port, a micro USB port, and a micro USB power port**

Component	Specification	Details
Arducam 5 MP Camera Module	**Resolution**	**5 Megapixels**
	Interface	**Compatible with CSI port of Raspberry Pi**
	Usage	**Enables live video streaming for**

		pet monitoring

Component	Specification	Details
UART Serial Port	**Interface**	**GPIO pins of Raspberry Pi**
	Usage	**For diagnostics and command inputs between the collar and base station**

List of Figures

1. Overview

This ICD covers the design and functionality of an IoT-based Pet Collar developed to enhance pet safety through features like GPS tracking, live video streaming, and a buzzer system. It defines the interfaces between the collar hardware, the user interface on the web, and the external receiver/base-station.

2. References and Definitions

2.1. References

MIL-STD-810F
Environmental Engineering Considerations and Laboratories Tests
1 Jan 2000
Change Notice 2
30 Aug 2002

American National Standard for VME64 (ANSI/VITA 1-1994 (R2002))
4 Apr 1995

American National Standard for VME64 Extensions (ANSI/VITA 1.1-1997)
7 Oct 1998

2.2. Definitions

CCA	Circuit Card Assembly
mA	Milliamp
mW	Milliwatt
MHz	Megahertz (1,000,000 Hz)
TBD	To Be Determined
TTL	Transistor-Transistor Logic
VME	VERSA-Module Europe
RTSP	Real Time Streaming Protocol
MQTT	Message Queuing Telemetry Transport

3. Physical Interface

3.1. Dimensions

3.1.1. Dimension of Collar

The pet collar system shall have dimensions not exceeding 2 inches in width, 1.5 inch in height, and 0.5 inches in thickness, ensuring it is compact and lightweight for comfortable attachment to the pet's collar.

3.2. Mounting Locations

For weight distribution and ergonomics considerations, the battery will be mounted behind the camera on the front portion of the collar. The MCU and cellular communication modules will be mounted on the back side of the collar for mass distribution and effective connectivity.

4. Thermal Interface

None of the modules included in the collar will require a heatsink. Due to the small form factor of the device, the natural air circulation will account for any high computation tasks for this application. The receiver subsystem will employ small heatsinks for the larger ICs.

5. Electrical Interface

5.1. Primary Input Power

The pet collar will include a 3.7 V 8000 mAh LiPo battery combined with a power regulation circuit to step down the input voltage to 3.3 V at a current between 0.5 mA and 800 mA.

5.2. Signal Interfaces

The communication between the collar and the receiver subsystems will be facilitated by the MQTT messaging protocol over an internet connection. The collar has two modes for communication: WIFI and cellular. The ESP32 MCU module will connect to the user’s home network through the on-chip WIFI antenna when available. An LTE Cat 4 cellular module will be used when the collar is outside of the home WIFI range.

5.3. Video Interface

The camera module will communicate with the MCU through a DVP 24-pin goldfinger interface, enabling an 8-bit JPEG compressed data stream. From the MCU, raw video data is transmitted to the base station for processing and relay to the user.

5.4. User Control Interface

All user input will be assessed through the phone application and relayed through the receiver to the collar. This is done to ensure the most efficient use of battery capacity.

6. Communications / Device Interface Protocols

6.1. Wireless Communications (WiFi)

The IoT Pet Collar employs WiFi technology, specifically the IEEE 802.11ac standard, for robust and efficient data transmission capabilities. This connectivity ensures seamless communication with both the base station and user mobile devices, employing the MQTT protocol for streamlined payload exchanges and the RTSP for real-time video streaming. This approach allows users to access live streams and data through a user-friendly interface, ensuring a high level of engagement and control over pet monitoring activities.

6.2. Host Device

At the core of our system is the Raspberry Pi Zero 2 W, chosen for its reliability and powerful computing capabilities. This unit facilitates not only the charging of the device via its Micro USB port but also enables direct data transfer when necessary. This choice underscores our commitment to leveraging cutting-edge technology to deliver a high-performance, versatile pet tracking solution.

6.3. Video Interface

The IoT Pet Collar integrates a compact, high-quality Arducam 5 MP camera module, transmitting data through the wireless receiver system using the RTSP protocol. This setup guarantees that live video streaming is delivered with minimal latency, offering pet owners real-time visibility and peace of mind. The inclusion of such advanced imaging technology highlights our dedication to providing superior monitoring capabilities.

6.4. Device Peripheral Interface

For diagnostics and direct command inputs, the collar incorporates a UART serial port, facilitating efficient communication between the collar and the base station via a custom protocol. This interface is meticulously designed to ensure reliability and ease of use, enabling immediate responsiveness and advanced control over the device's functionalities.

IOT based Pet tracker

Max Kotas, Timothy Thompson, Darren Lefever

SCHEDULE

2 December 2024

Schedule/Execution Plan for IOT based Pet tracker

	Feb. 19	Feb. 26	Mar. 4	Mar. 11	Mar. 18	Mar. 25	Apr. 1	Apr. 8	Apr. 15	Apr. 22	Apr. 29	**Date**
Order all collar hardware modules											Not Started	
Draft UI Design											Complete	
Design Receiver with modules											In Progress	
Get RTSP protocol working											Behind	
Get MQTT protocol working												
Design Power Delivery System												
Implement State Model for MCU												
Order Receiver Parts												
Map API integration/Geofencing set up												
Verify power delivery for normal and LPM (collar)												
Collar PCB design submission												
Coordinate and implement Transmission Data Structure												
First Site launch and Testing												

	Feb. 19	Feb. 26	Mar. 4	Mar. 11	Mar. 18	Mar. 25	Apr. 1	Apr. 8	Apr. 15	Apr. 22	Apr. 29	**Date**
First Site launch and Testing											Not Started	
GPS upload and geofence detection											Complete	
Collar Breadboard testing with all components											In Progress	
Video/Buzzer remote activation (website)											Behind	
Learn and implement NODE RED												
Machine Learning Dataset collection and model training												
Design collar equipment housing												
Finalize and Refine site and features												
ML prediction and accuracy testing												
Organize MongoDB Parsing and standard format												
Implement MQTT message Input/Output												
Fine tune extra Subsystem Communication												
Video transmission at full quality (collar)												
Solder and Complete PCB												

IOT based Pet tracker

Max Kotas, Timothy Thompson, Darren Lefever

VALIDATION

2 December 2024

Validation Plan for IOT based Pet tracker

Test Name	Success Criteria	Methodology	Status	Responsible Engineer
UI Draft design	Visually appealing design, navigational elements are intuitive and easy to understand	Conduct usability testing with representative users to gather feedback on the UI design	Tested	Darren Lefever
Map API	The map API accurately displays geographical information and coordinates.	Verify the accuracy of location displayed on the map by comparing it with known geographic coordinates	Tested	Darren Lefever
Geofencing	Geofencing accurately detects when a pet exits and re-enters a predefined safe area	Conduct field testing to validate the accuracy and reliability of geofencing functionality in real-world environments	Tested	Darren Lefever
Receiver Hardware Setup	Reliable Wi-Fi and 4G physical signal reception and detection	verify reliable connection and communication times for inter-subsystem interactions	Tested	Max Kotas
Power System	Delivers ~1.3A at 3.3V for at least 1 hour, with consistent performance.	Monitor the battery and step-down circuit run across a non-digital load with multimeter over time.	Tested	Timothy Thompson

Test Name	Success Criteria	Methodology	Status	Responsible Engineer
Collar State Machine	Coller system seamlessly switches from state to state given the correct inputs and operates within design parameters for each state.	Simulate receiver to collar communication and monitor the state in which the collar operates, ensuring proper response.	Tested	Timothy Thompson
Reliable RTSP Communication	Reliable video stream through RTSP communication	RTSP stream with average of >5 FPS over time with low latency	Tested	Max Kotas
MQTT Routing and Brokerage	Reliable, low-latency MQTT communication with the receiver acting as the broker	<100 ms latency for IoT on local network with working I/O manipulation on simulated collar hardware using MQTT	Tested	Max Kotas
MongoDB Storage and Database Querying	Ability to store and retrieve data as fast as possible with ability to query specific data entries	High Success rate of data storage and ability to retrieve specific data entries with reliable data querying	Tested	Max Kotas

Performance on Execution Plan: The execution plan was successfully carried out, despite some tasks being performed slightly out of sequence to accommodate the many challenges faced throughout the semester. Nevertheless, every task outlined in the plan, which did not require a PCB, was completed within the specified timeframe. The project was otherwise successfully concluded, meeting all the objectives outlined in the initial proposed solution.

Performance on Validation Plan: The validation plan was completed, with careful checks on each part of the individual subsystems. Tests were performed on all the available subsystems and their performance to ensure they will work correctly when integrated into the full system next semester. Any complications were communicated with the team, sponsor and professors and resolved. Specifically, delays posed by the arduous PCB design process were addressed with the professor, and any incomplete items will be completed before the start of next semester.

IOT based Pet tracker

Max Kotas, Timothy Thompson, Darren Lefever

SUBSYSTEM REPORT

25 April 2024

SUBSYSTEM REPORT
FOR
IOT based Pet Tracker

TEAM <34>

APPROVED BY:

_____
Project Leader Date

_____
Prof. Kalafatis Date

_____
T/A Date

Table of Contents

Table of Contents **2**
List of Tables **4**
List of Figures **5**
1. Introduction **7**
2. Receiver Subsystem Report **8**
2.1 Subsystem Introduction 8
2.2 Validation 10
2.2.1 Wifi 10
2.2.2 MQTT 11
2.2.3 RTSP 12
2.2.4 MongoDB 13
2.2.5 Node Red 16
2.3 Subsystem Conclusion 16
3. User Interface Subsystem Report **17**
3.1 Subsystem Introduction 18
3.2 Subsystem Details and Features 18
3.2.1 Website Homepage 18
3.2.2 User Defined Safe Area 20
3.2.3 Live Location/Notification 23
3.2.4 Toggle Features 26
3.2.5 Refined Safe Area/Tracking History 27
3.2.6 Machine learning 35
3.2.7 About/Contacts and Settings page 39
3.2.8 User Authentication and login 43
3.3 Subsystem Validation 45
3.4 Subsystem Conclusion 45
4. Pet Collar Subsystem Report **29**
4.1 Subsystem Introduction 29
4.2 Subsystem Details and Features 29
4.2.1 Microcontroller: Espressif ESP32-S2 Platform 29
4.2.2 Quectel EC25 Mini PCIe 30
4.2.3 Camera 30
4.2.4 Buzzer 31
4.2.5 Custom PCB 31
4.2.6 Battery Power Supply 32
4.3 Subsystem Validation 32
4.3.1 Collar State Machine Software 32
4.3.2 Inter-Module Communication 33
4.3.3 Wi-Fi and Cellular Data rate Capacity 34

4.3.4 MQTT Data rate and Latency 34
4.3.5 Power System Validation 34
4.3.6 GPS Accuracy and Reliability 35
4.3.7 Video Streaming Performance 35
4.4 Subsystem Conclusion 36

List of Tables

Table 2.2.4.1: Types of data stored in the MongoDB

List of Figures

Figure 2.1.1: *The receiver hardware in 3D printed designed case*
Figure 2.1.2: Block diagram of the Receiver communication with other parts of the subsystem
Figure 2.2.1.1: Wireless connection speed test from receiver graph using home network
Figure 2.2.1.2: Wireless strength test from receiver graph using home network
Figure 2.2.2.1: MQTT latency test over local network from receiver to external reading device
Figure 2.2.2.2: MQTT Test on the "mock" collar, showing that all MQTT communication was received from the receiver
Figure 2.2.3.1: FPS over time, measured from RTSP stream cast to receiver via local network over interval of 100 seconds
Figure 2.2.3.2: RTSP Stream Capture from "mock" collar
Figure 2.2.4.1: Latency for multiple trials of database querying for each database instance
Figure 2.2.4.2: Visual database display of image captures retrieved from MongoDB Database, collection "images"
Figure 2.2.4.3: Visual display of simulated GPS coordinates MongoDB instance
Figure 2.2.5.1: Node Red workflow for implementation phase of project
Figure 2.2.5.2: example RTSP stream dashboard for IoT pet collar project camera.
Figure 3.2.1.1: Homepage
Figure 3.2.1.2: Homepage pt. 2
Figure 3.2.2.1: Tracking Page
Figure 3.2.2.1: Create Geofence Name
Figure 3.2.2.3: View/Apply/Delete Saved Geofences
Figure 3.2.3.1: Live Tracking
Figure 3.2.3.2: Entering Geofence
Figure 3.2.3.3: Exiting Geofence
Figure 3.2.3.4: EmailJS template
Figure 3.2.3.5: Email sent to the User
Figure 3.2.4.1: Feature Page
Figure 3.2.5.1: Refine Safe Area Page
Figure 3.2.5.2: Visual Aid for Steps of the Refined Safe Area Program
Figure 3.2.5.3: 7 days of Geofence Data
Figure 3.2.5.4: Refined Safe Area using different algorithms
Figure 3.2.6.1: All points displayed for DBSCAN
Figure 3.2.6.2: Clustered points displayed after DBSCAN
Figure 3.2.6.3: Points inputted into table after DBSCAN
Figure 3.2.6.4: DBSCAN data and parameters in console output

Figure 3.2.7.1: About Page
Figure 3.2.7.2: Contacts/Settings Page
Figure 3.2.7.2: Edit Email Modal
Figure 3.2.7.3: Theme Toggle
Figure 3.2.7.4: Dark Mode Homepage
Figure 3.2.7.5: Contact Support for user -> team communication
Figure 3.2.7.6: Email from User using contact support
Figure 3.2.8.1: Register for website
Figure 3.2.8.2: Register for website
Figure 4.2.2.1: An Espressif ESP32-S2 Developer Board
Figure 4.2.2.1: A Queltec EC25 Mini PCIe module
Figure 4.2.4.1: Buzzer Module
Figure 4.2.3.1: ESP32-CAM
Figure 4.2.4.1: Buzzer Module
Figure 4.2.5.1: Custom PCB 2D
Figure 4.2.6.1: Battery
Figure 4.3.1.1: Collar State Machine Testing Output
Figure 4.3.2.1: Custom PCB 3D
Figure 4.3.3.1: ESP32-S2 Devkit In Use
Figure 4.3.6.1: Power System Schematic

1. Introduction

The smart pet collar tracking system is designed to serve as an autonomous guardian for pets, ensuring their safety and well-being around the clock. This innovative system is structured into three critical subsystems: the collar hardware, the receiver, and the user interface. Each of these subsystems has been rigorously validated and confirmed to function effectively, laying a solid foundation for seamless integration.

As part of the ECEN capstone project, we are poised to advance to the next phase where these subsystems will be harmoniously integrated to create a cohesive and efficient unit. This integration is anticipated to enhance the system's functionality and user experience, making it not only reliable but also intuitive for pet owners to use. With this strategic amalgamation, the smart pet collar tracking system is set to redefine pet safety and owner peace of mind.

2. Receiver Subsystem Report

2.1 Subsystem Introduction

The receiver subsystem is designed to facilitate communication and database storage in the workflow of the IoT-based smart pet collar subsystem workflow. This subsystem uses a wifi connected Raspberry Pi 5 8GB model as the hardware with a 128 GB sd card (as seen in *Figure 2.1.*) to store all data in a MongoDB database.

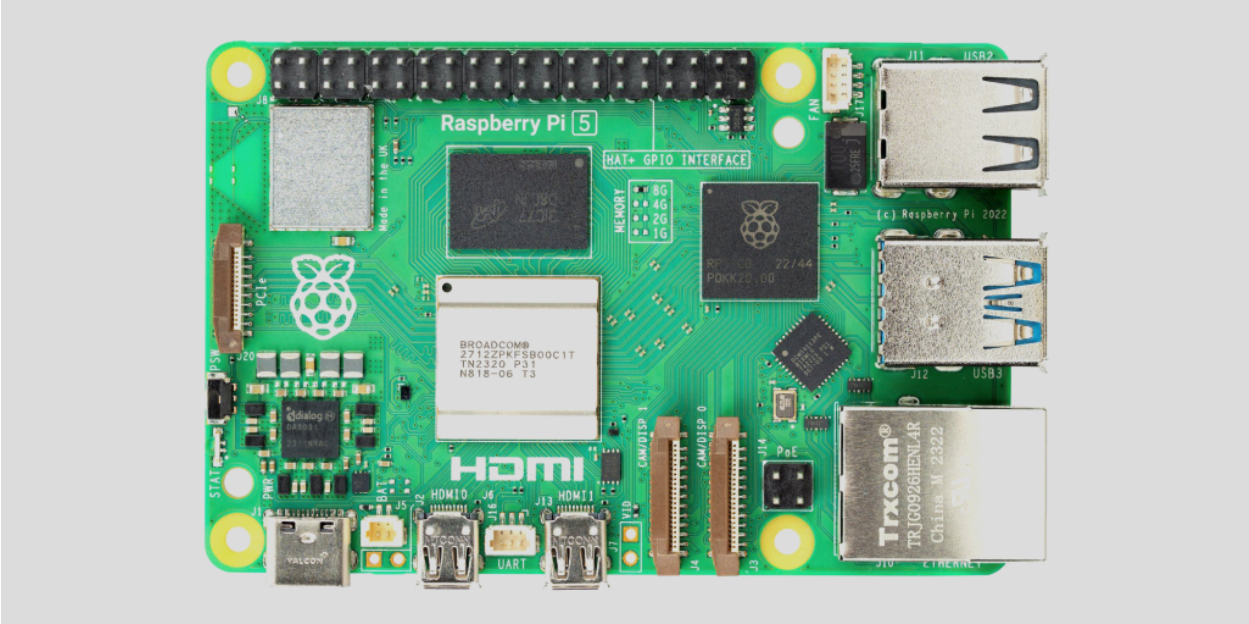


Figure 2.1.1: The *receiver hardware*

The receiver has the responsibility of facilitating communication between the collar hardware and the user interface. the code used for this project can be found at https://github.com/MKotTamu/ReceiverCode

Components

- Receiver/Broker: This receiver serves as the primary node for managing and directing data flows. It acts as both the receiving point for our data and a broker, utilizing MQTT to assign tasks and distribute payloads to their intended destinations. RTSP streams and All MQTT messages, both incoming and outgoing, are managed by this receiver, which also controls the collar hardware and collects data from sensors.
- Advanced Collar Equipment: Comes integrated with a camera, a buzzer, and GPS tracking capabilities. This equipment manages MQTT commands sent from the receiver, streams video content through RTSP, and sends GPS positions in real time.

- Interactive User Interface: Provides a visual dashboard for displaying data and alerts from the central hub. It's crucial for user interaction and continuous monitoring of the system.
- MongoDB Data Repository: Stored on the receiver, it holds various datasets including logs, GPS locations, and captured images. This database is designed for quick data retrieval and efficient management of storage space.

Communication Processes

- MQTT Messaging: The receiver, acting as the MQTT broker, facilitates all MQTT communications. It receives commands from the user interface, which are then sent to the collar to control the camera, buzzer, and GPS. It also receives data from the collar, such as GPS coordinates.
- RTSP Stream Handling: The camera integrated into the collar streams real-time video via RTSP directly to the receiver. The receiver processes these streams to capture images periodically, encoding them in binary before storage.
- Data Handling and Storage:
    - Logging: All activities, including data transmission times and RTSP image captures, are logged in JSON format.
    - GPS Logging: Captures and stores GPS data every 10 seconds to track the movement and location of the collar.
    - Image Capture and Storage: Depending on the safety mode, images are captured at different frequencies (every 10 minutes in safe mode and every 10 seconds in non-safe mode) and stored in the MongoDB.

2.2 Validation

2.2.1 Wifi

Starting off with Wifi connectivity from the receiver, there seems to be a constant wireless speed of 30-70 Mbps with an average of around 45.31 Mbps over the home wifi network as shown in *Figure 2.2.1.1* This is a drop strength of this connection is only 5% compared to the router wifi speed, which is within nearly negligible range for the scale of this application. Wifi connectivity speed will determine how fast the user can get an alert on their phone when their pet has left the safe zone, and determine how fast commands can be sent from the user to the collar and vice versa.

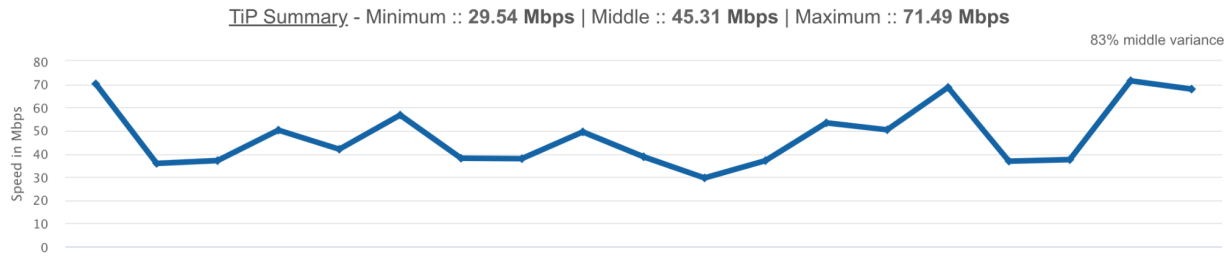


Figure 2.2.1.1: Wireless connection speed test from receiver graph using home network

The signal strength over a specific time period was also tested with a clear path to the router and monitored over time. The signal strength falls around -47 to -45 dBm as seen in *Figure 2.2.1.2*, which falls into the “excellent signal strength” category of -30 to -50 dBm. This is ideal for quick transmission of MQTT messages and large data streams such as RTSP

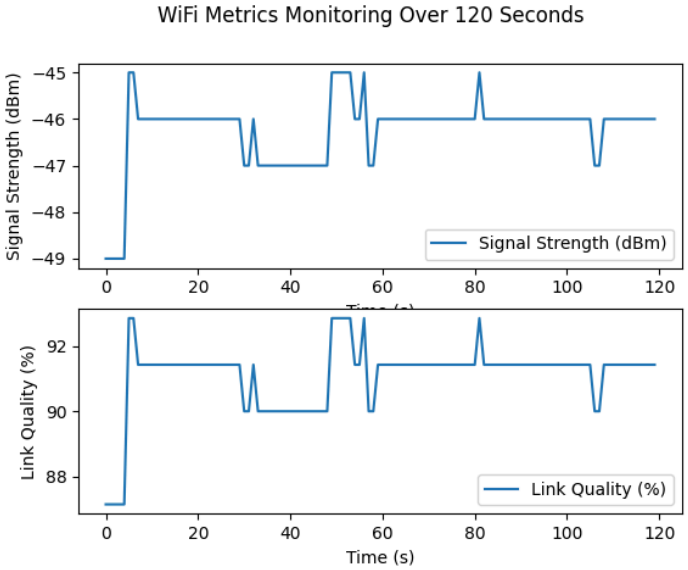


Figure 2.2.1.2: Wireless strength test from receiver graph using home network

2.2.2 MQTT

With MQTT, latency is the most important factor as it determines how quickly a message is received from the time that it is sent out. The latency is tested by logging the time that the receiver sends out a message and logging the time that the external device receives the message on the same network. The difference of these 2 times are logged, then the difference is found as its latency. The payload of the message sent in this validation test is an 8 byte string. A low latency via the local network is 2-12 ms, which the MQTT latency of the receiver achieves when read by another device on the network, in this case being another computer using these messages. The MQTT message latency seems to fall between 4-8 ms as seen in *Figure 2.2.2.1*, not counting outliers at startup.

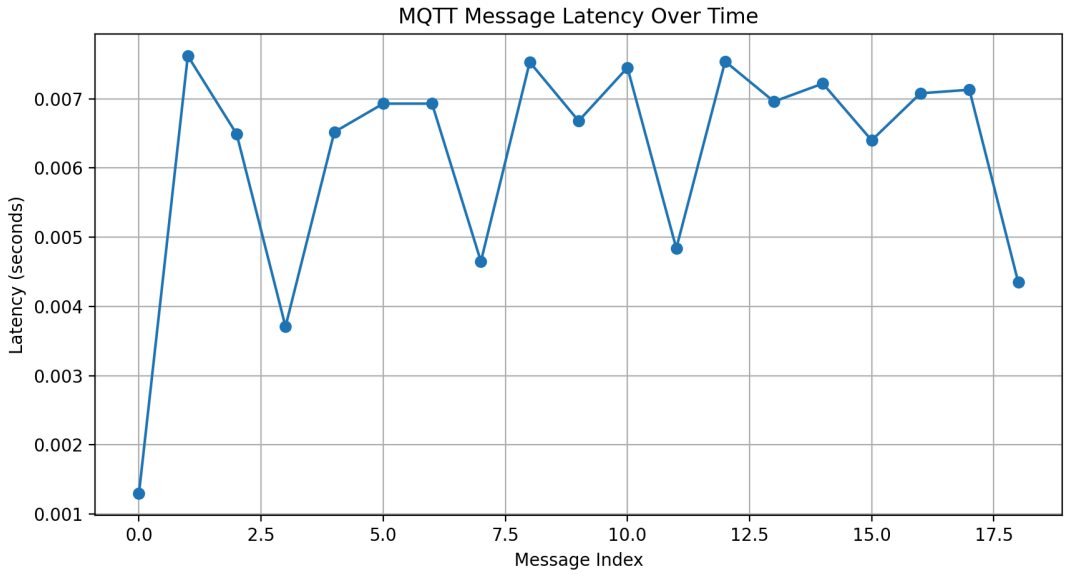


Figure 2.2.2.1: MQTT latency test over local network from receiver to external reading device

Another operating characteristic of the MQTT transmissions was ability to manipulate outputs from a given payload on a specific topic. This was tested on a “mock” version of the collar, emulated through a Raspberry Pi Zero 2 W, with a light MQTT message listener script, executing in a loop. This payload script then activates GPIO pins in order to simulate receiving messages from a user interface to turn off and on a buzzer. This worked perfectly as demonstrated in *Figure 2.2.2.2*

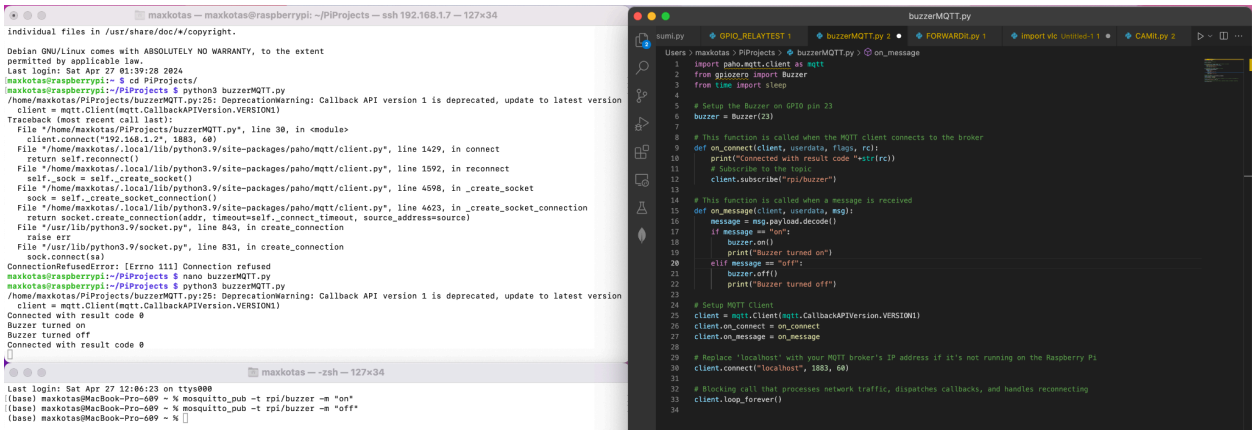


Figure 2.2.2.2: MQTT Test on the "mock" collar, showing that all MQTT communication was received from the receiver

2.2.3 RTSP

With the RTSP stream, we want to make sure that there is a reliable connection over time. This reliability was validated in *Figure 2.2.3.1* which shows that there is a reliable stream of 6-14 frames per second, more than enough to get a reliable sense of the pet's locations and activities at any given time. This RTSP stream will be take from the collar hardware and broadcast to the user interface during the integration phase, and the RTSP stream has enough leeway for drop in quality of 10-30% with extra broadcast connections while still being functional.

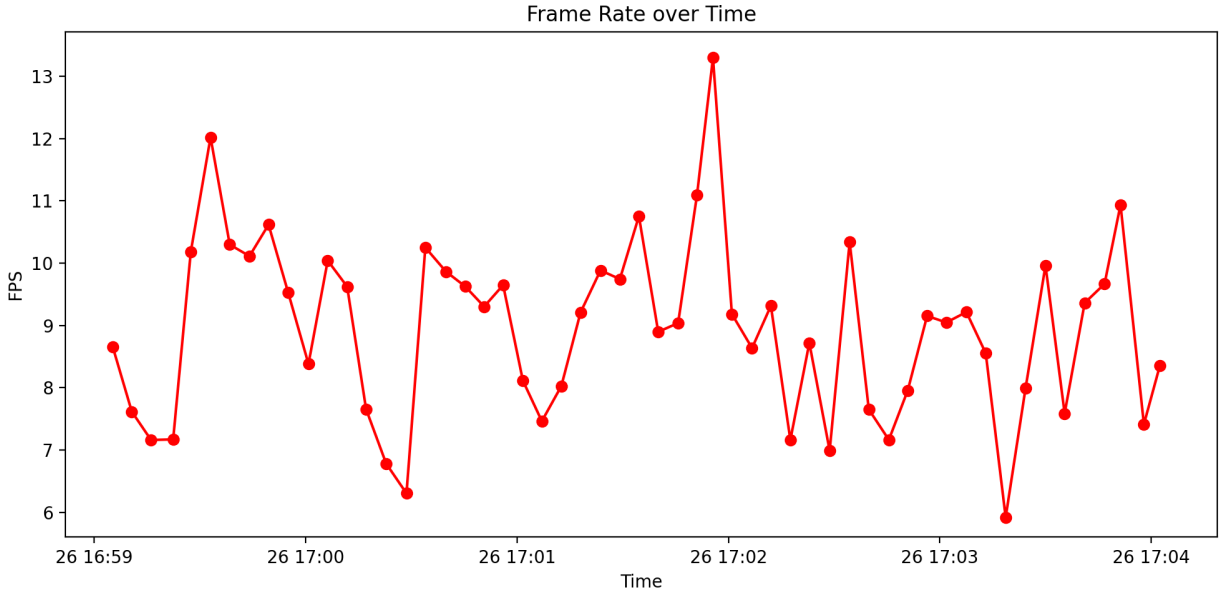


Figure 2.2.3.1: FPS over time, measured from RTSP stream cast to receiver via local network over interval of 100 seconds

The clarity of the stream is also vital to the success of the validation for the RTSP stream. The camera on the “mock” collar has shown success in generating 480p frames on a consistent basis. A capture of an apartment setting from the RTSP stream is shown in *Figure 2.2.3.2*.

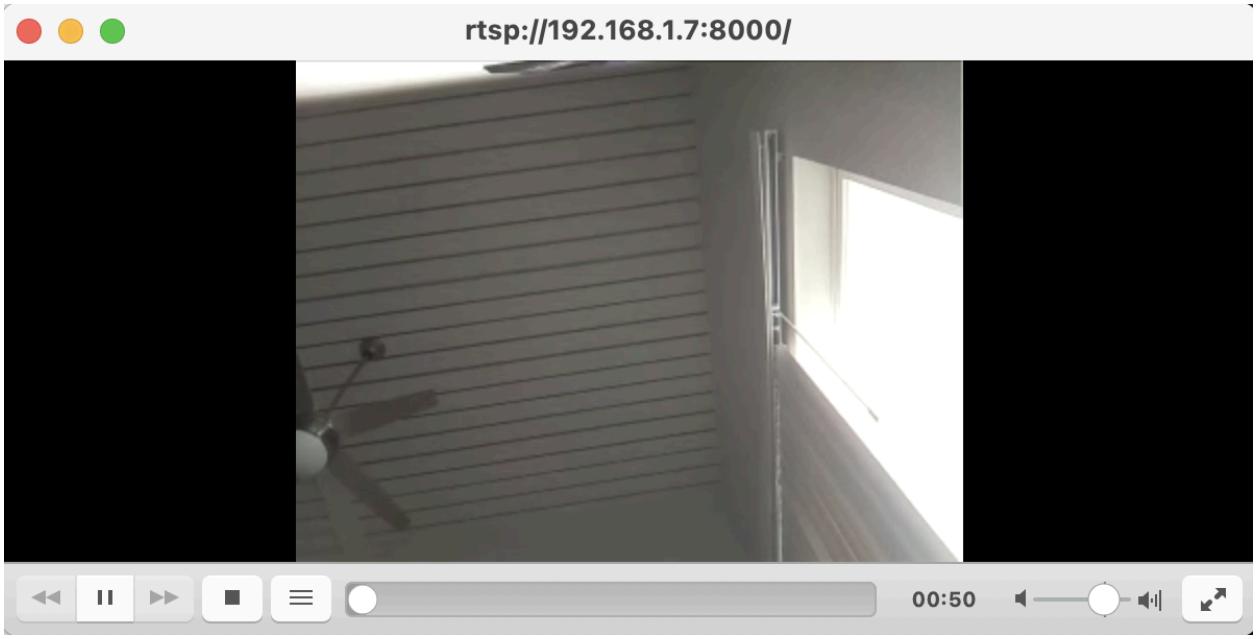


Figure 2.2.3.2: RTSP Stream Capture from “mock” collar

2.2.4 MongoDB

For the different data type storage, MongoDB will be utilized as the main database. The types of data stored from each component of the collar hardware is detailed in *Table 2.2.4.1*.

Component	Collection Name	Data Stored	Interval of Capture	Datatype
Camera	realtime_collar[images]	Time, Captured Video Frame	Safe Mode: 10 min Unsafe Mode: 10 sec	Binary Encoded .jpg
Buzzer	realtime_collar[buzzer]	Time, Command	User Initiated	String
GPS	realtime_collar[gps]	Time, Latitude, Longitude	Safe Mode: 1 min Unsafe Mode: 10 sec	String

Table 2.2.4.1: Types of data stored in the MongoDB

This data will be stored by a python script running on the receiver on loop, to accurately retrieve the data from the MongoDB database instance. Since this project is not yet in the integration phase, three methods were used to simulate data collection. For this aspect of the subsystem to be validated, low latency retrieval methods need to be employed. Each of the database collections were queried multiple times to ensure reliability. These tests are graphed below in *Figure 2.2.4.1*. The binary encoded images ended up having the highest latency times, while the MQTT messages for the buzzer had the lowest. The highest latency was ~35 ms, which is well within the acceptable threshold for a functional receiver.

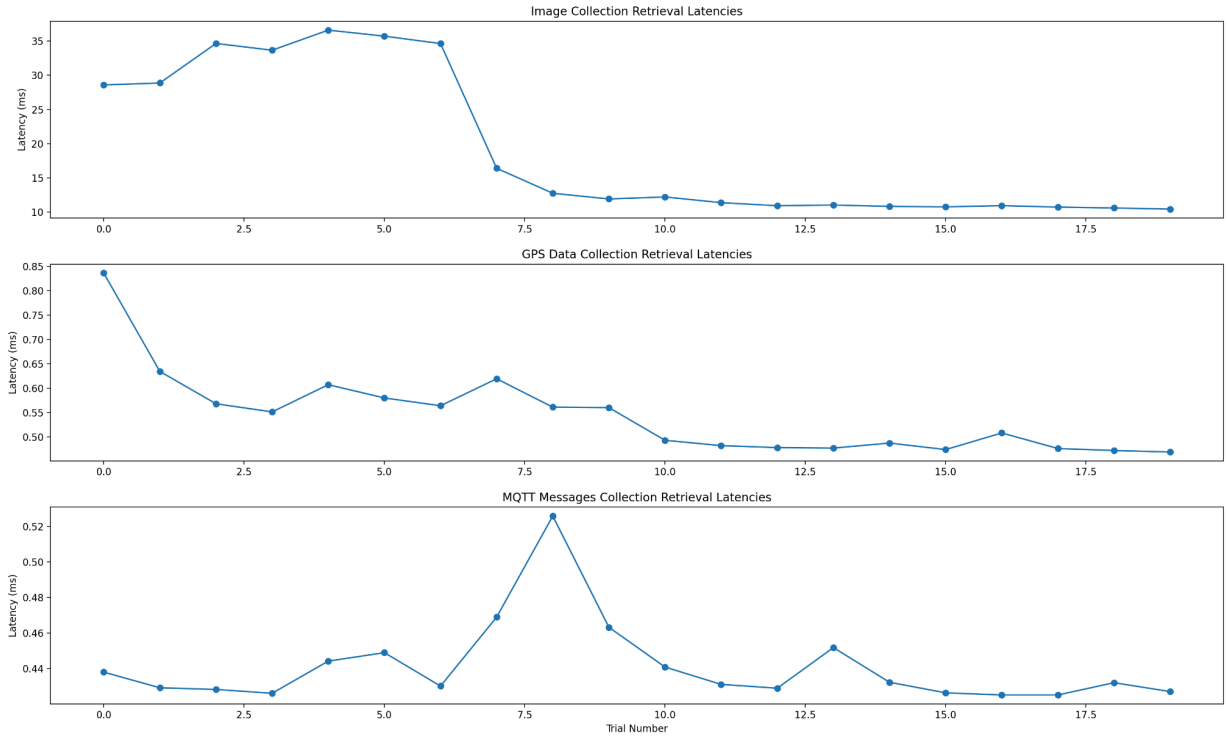


Figure 2.2.4.1: Latency for multiple trials of database querying for each database instance

For the camera, images from an RTSP stream running on a simulated “mock” collar were taken from the RTSP stream which were then encoded with a binary encoding function, and stored in a MongoDB database instance. These images were accessed by timestamp, and were monitored for latency. The images then needed to be displayed in a visual fashion, so matplotlib was used to create a visual image database display. A portion of this database is shown in *Figure 2.2.4.2*.

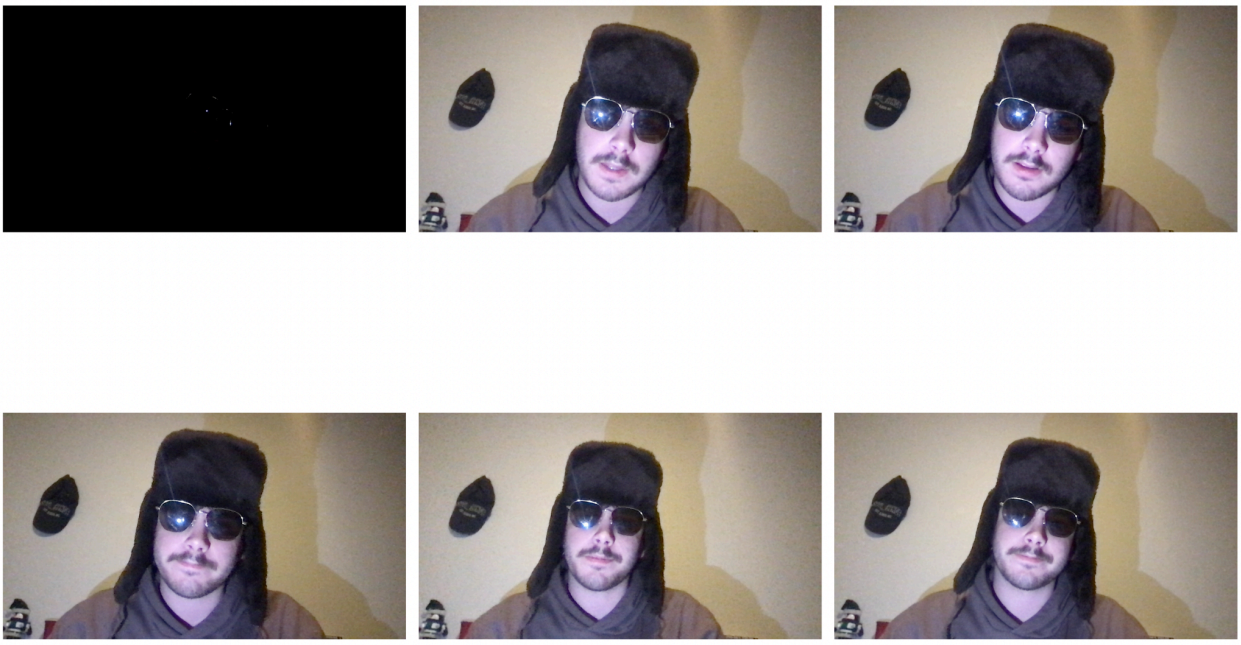


Figure 2.2.4.2: Visual database display of image captures retrieved from MongoDB Database, collection “images”

For the GPS data, since this project is not into the integration phase, a function was implemented that generated random coordinates. These coordinates were stored as strings, just like the MQTT buzzer messages. The string data display method was completed using the pandastable framework, and the visualization of this data is shown in *Figure 2.2.4.3*. This same type of data visualization is used for the buzzer MQTT messages as well.

GPS Data

	Timestamp	Latitude	Longitude
1	2024-04-02 23:39:58.588000	-61	90.86
2	2024-04-02 23:39:59.701000	28.11	154.15
3	2024-04-02 23:40:00.707000	76.33	-18
4	2024-04-02 23:40:01.715000	-28	-56
5	2024-04-02 23:40:02.721000	72.87	-52
6	2024-04-02 23:40:03.728000	-2.6	-11
7	2024-04-02 23:40:04.734000	-28	-87
8	2024-04-02 23:40:05.739000	2.14	-1.8e+02
9	2024-04-02 23:40:06.745000	-34	-1.7e+02
10	2024-04-02 23:40:07.751000	78.34	0.96
11	2024-04-02 23:40:08.756000	-13	-1.5e+02
12	2024-04-02 23:40:09.762000	27.44	-1.4e+02
13	2024-04-02 23:40:10.768000	59.72	20.41
14	2024-04-02 23:40:11.774000	-22	-1.2e+02
15	2024-04-02 23:40:12.778000	79.88	-1.2e+02
16	2024-04-02 23:40:13.785000	54.47	30.62
17	2024-04-02 23:40:14.790000	44.08	-28

28 rows x 3 columns

Figure 2.2.4.3: Visual display of simulated GPS coordinate MongoDB instance

2.2.5: Control Flow Diagrams

The control flow of all processes being received and manipulated by the receiver are shown in figures below:

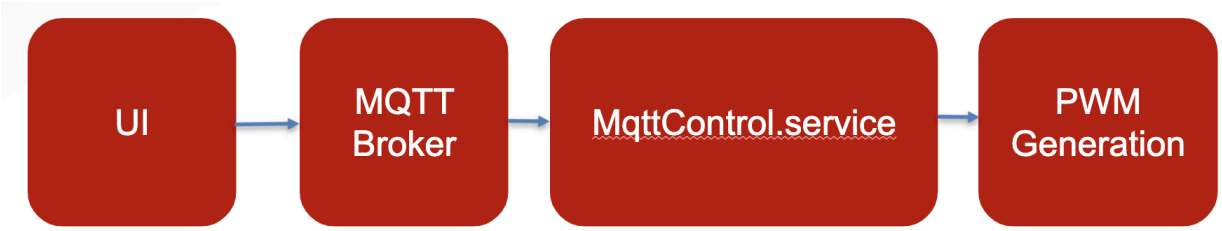


Figure 2.2.5.1: Buzzer Control Flow Diagram

Figure 2.2.5.2: Video Control Flow Diagram

Figure 2.2.5.3: GPS Control Flow Diagram

All of the methodologies and communication protocols were implemented together with the design presented in *Figure 2.2.5.1, Figure 2.2.5.2,* and *Figure 2.2.5.3*. These were integrated successfully in a testbench that was designed on a Raspberry Pi Zero 2 W.

2.3 Subsystem Conclusion

All of the communication and database protocols function well and according to plan. Despite some irregularities in the consistency of the wifi signal from the receiver at times, the subsystem functions with low latency, exceeding expectations and validating this subsystem to be prepared for the implementation phase of this project. The communication that this receiver facilitates is

crucial to getting the other subsystems to work together in a thorough and fast-paced manner, executing data transfer protocols at specified times autonomously.

3. User Interface Subsystem Report

3.1 Subsystem Introduction

The user interface serves as the central hub for managing all functionalities of the pet collar. It enables users to seamlessly create and personalize a safe area for their pet using geofencing APIs, complete with live GPS tracking, an adjustable audio buzzer, and a video camera toggle. Additionally, a rapid notification system is integrated to promptly alert users of any breaches in the designated safe zone. Leveraging machine learning, the geofencing feature continuously refines its accuracy by analyzing the pet's travel patterns on a weekly basis, while daily GPS data is securely stored for future reference.

GITHUB Link: https://github.com/DarrenElijah/ECEN_404

3.2 Subsystem Details and Features

3.2.1 Website/Homepage

The website's user interface is designed and implemented using fundamental client-side technologies to ensure a seamless and responsive user experience. Hypertext Markup Language (HTML) is utilized to establish the structural foundation of the site, while Cascading Style Sheets (CSS) are employed to define its visual styling, ensuring a polished and aesthetically pleasing appearance. JavaScript (JS) is integrated to enable dynamic functionality across the platform. The website comprises six distinct pages, each serving a unique purpose. The Home Page, serving as the entry point, features an intuitive and visually appealing design showcasing the product name prominently. This page provides an overview of the website's features, offering users a high-level introduction to its capabilities. A strategically placed "Get Started" button invites users to explore the product further and begin engaging with its functionalities.

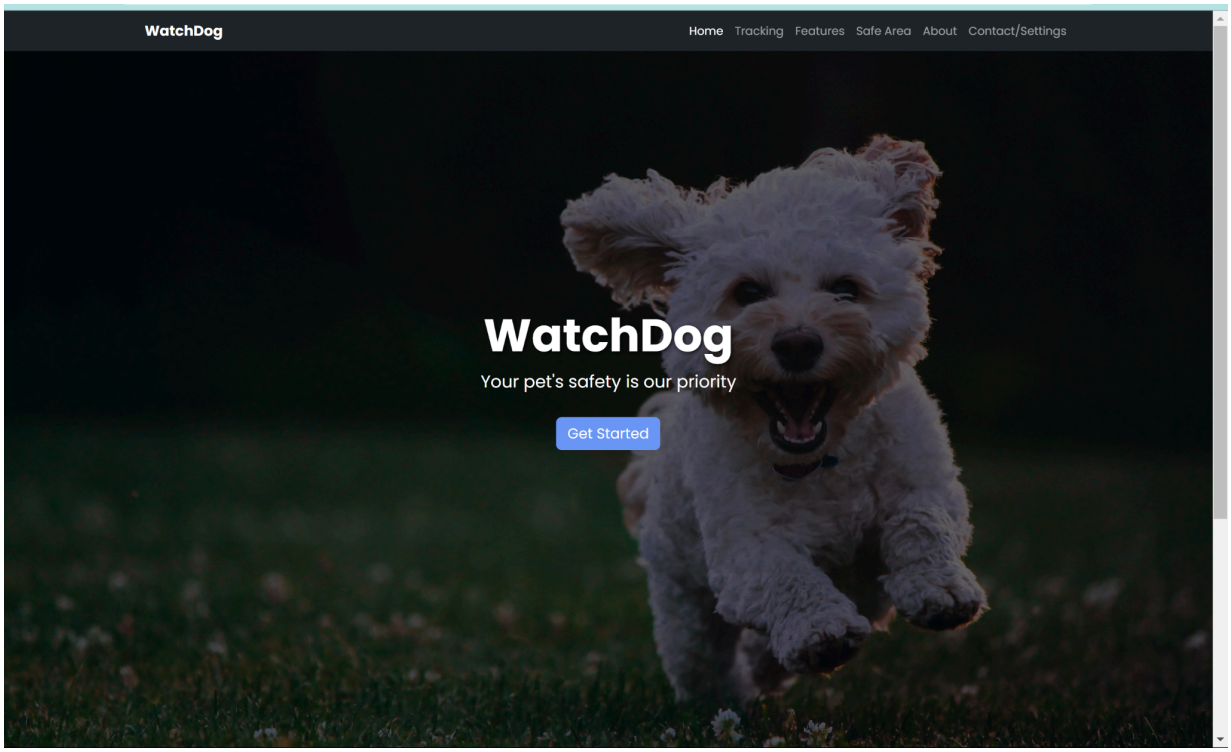


Figure 3.2.1.1: Homepage

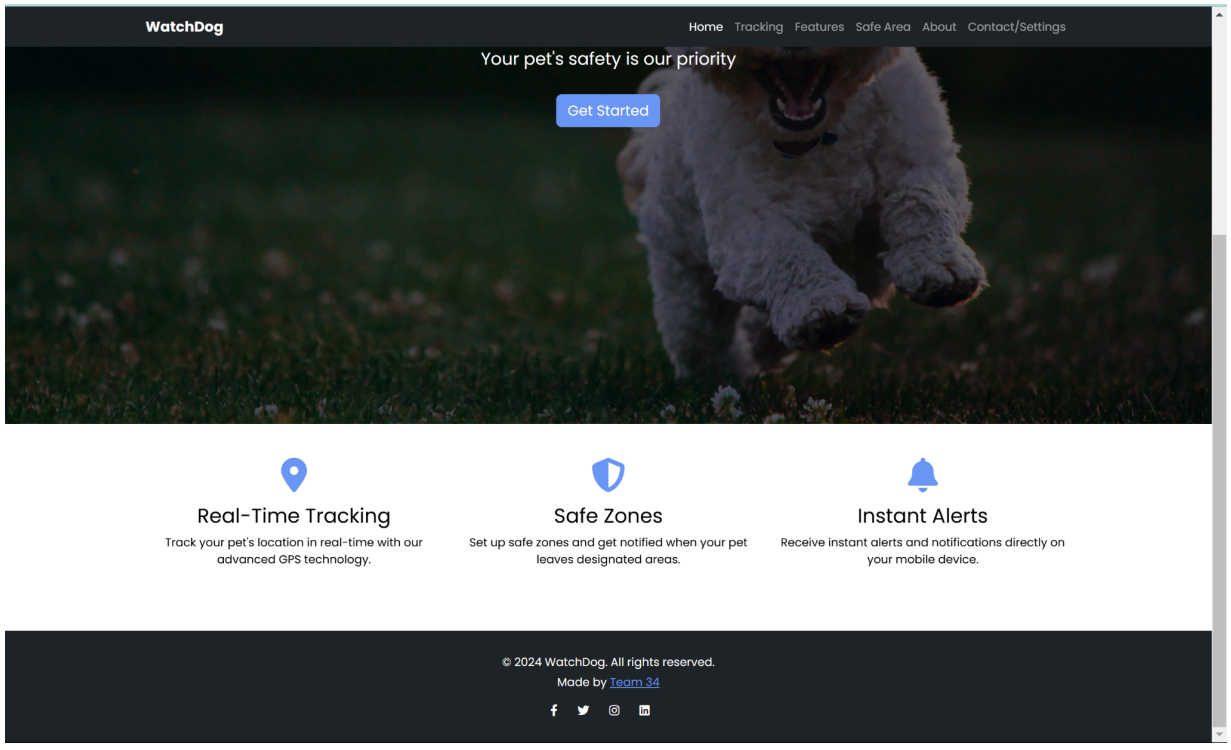


Figure 3.2.1.2: Homepage pt. 2

3.2.2 User Defined Safe Area

The Tracking Page is a core component of the application, designed to enable users to monitor their pets' locations and manage geofences with ease. Leveraging the Google Maps API, the page offers a dynamic and intuitive interface for geofencing and real-time GPS tracking. At the heart of the page is an address input field labeled "Enter Address", where users can specify a location. Using the Google Maps Geocoding API, the application processes the input and centers the map on the desired address, allowing users to begin creating geofences.

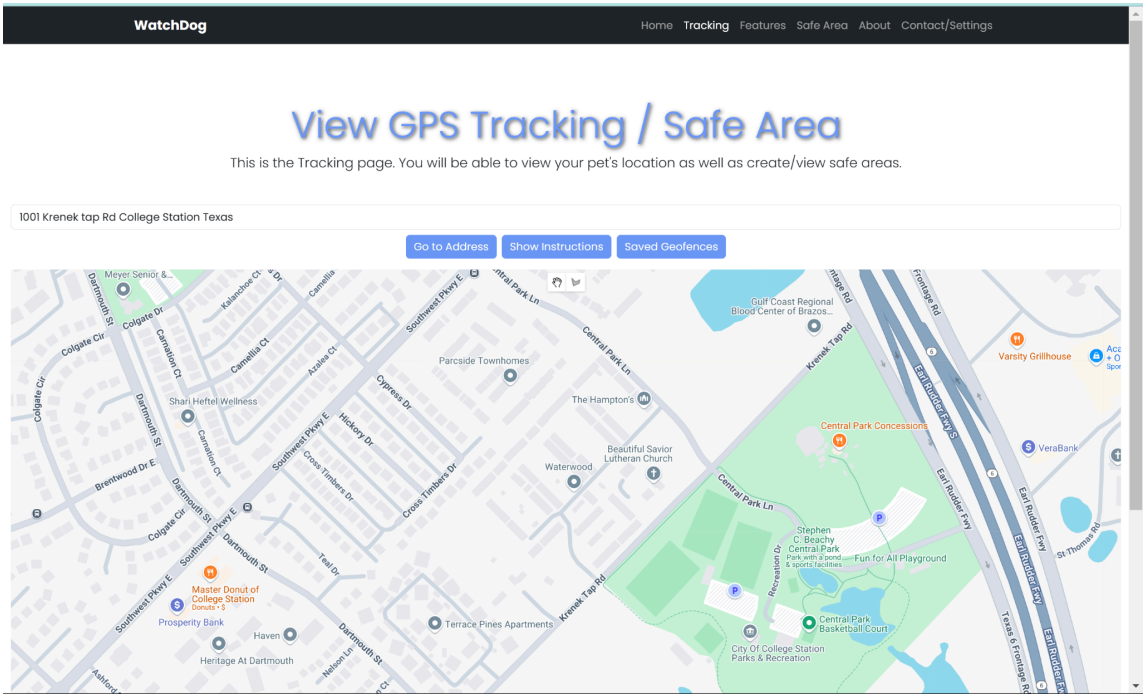


Figure 3.2.2.1: Tracking Page

Users can draw custom polygon geofences directly on the map using the Google Maps Drawing Manager, with the ability to adjust boundaries by dragging the polygon's vertices. Once a geofence is drawn, a modal prompts the user to name and save it. Saved geofences are stored with location context, such as the city name, determined via reverse geocoding. These geofences can be accessed in the "Saved Geofences" tab, where users can apply a selected geofence to monitor their pet's movement or delete unwanted ones through a confirmation process. When applied, the geofence is rendered on the map, and the pet's real-time GPS location is updated continuously using data received via Socket.IO. The application uses the Google Maps Geometry Library to determine if the pet is inside or outside the geofence, triggering notifications

for significant events like geofence entry or exit. These notifications are displayed on-screen using dynamic toast messages, ensuring users are promptly alerted to any changes.

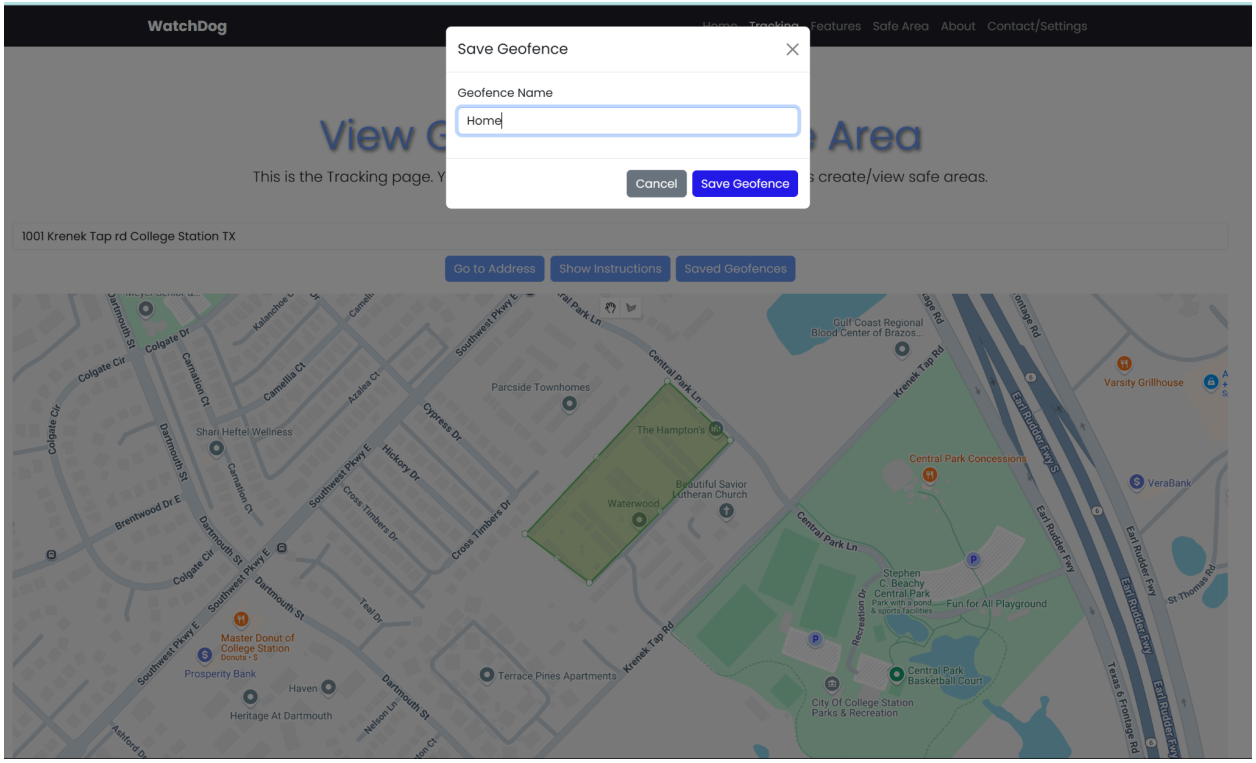


Figure 3.2.2.1: Create Geofence Name

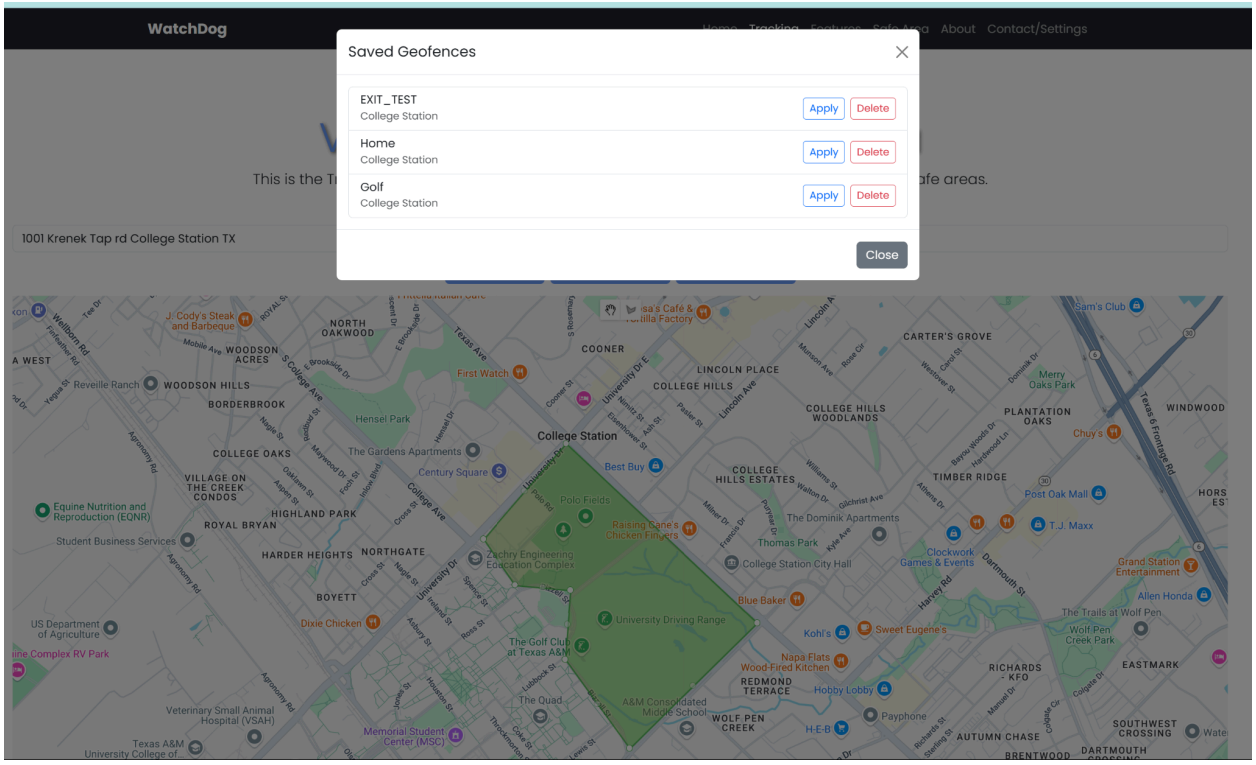


Figure 3.2.2.3: View/Apply/Delete Saved Geofences

To enhance usability, a collapsible "Show Instructions" section provides step-by-step guidance on geofencing and tracking, making the interface accessible even for first-time users. Additionally, email notifications are sent for critical events, such as geofence exits, using EmailJS, ensuring users are informed even when away from the application. The page also supports a dark mode theme, which adapts the interface for low-light environments based on the user's preferences stored in localStorage. These preferences, along with the currently applied geofence, are persistent across sessions and synchronized between browser tabs for a seamless user experience.

The Tracking Page is a robust tool that combines modern technologies like the Google Maps API, EmailJS, and Socket.IO with user-friendly design principles. It offers a comprehensive solution for real-time GPS tracking and geofence management, empowering users to ensure their pet's safety efficiently and effectively.

3.2.3 Live Location/Notification

The Tracking Page provides users with the ability to monitor the real-time location of their pet through a red marker displayed on the map. This functionality is powered by the Google Geolocation API, which retrieves the current location of the designated tracker device, enabling live and accurate tracking. When a geofence, or "safe area," is established, the system evaluates the tracker’s position relative to this boundary upon fetching the live location. Two scenarios may occur: if the tracker is within the defined geofence, the system remains passive, ensuring uninterrupted monitoring. Conversely, if the tracker is detected outside the geofence, an automated email notification is immediately dispatched to the user, alerting them to the breach. This proactive approach ensures users are promptly informed, allowing for timely intervention to ensure their pet's safety.

To facilitate email notifications, the system utilizes EmailJS, a JavaScript library that enables direct email transmission from client-side code. Upon detecting a geofence breach, the system triggers an email notification via EmailJS, delivering critical information about the incident. This feature provides users with real-time updates on their pet’s status, even when they are not actively monitoring the application. The integration of EmailJS ensures seamless communication and enhances the system’s reliability, empowering users to take swift and appropriate action to address potential safety concerns.

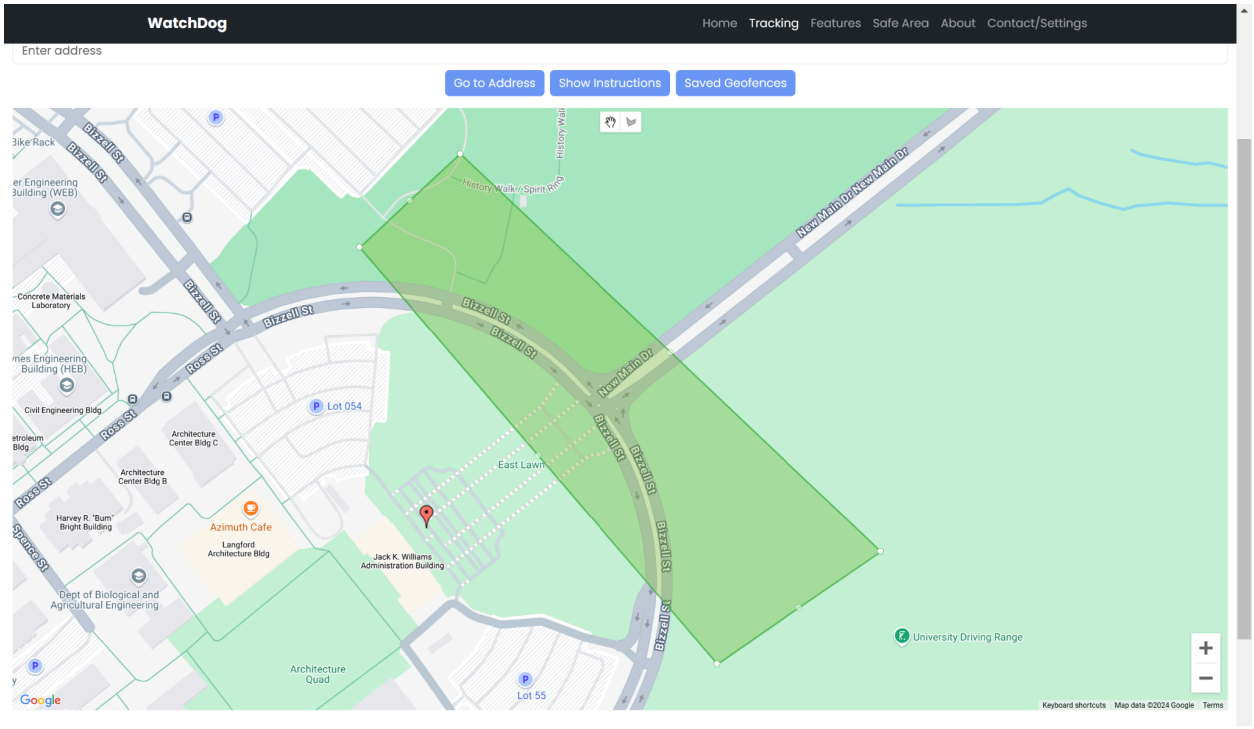


Figure 3.2.3.1: Live Tracking

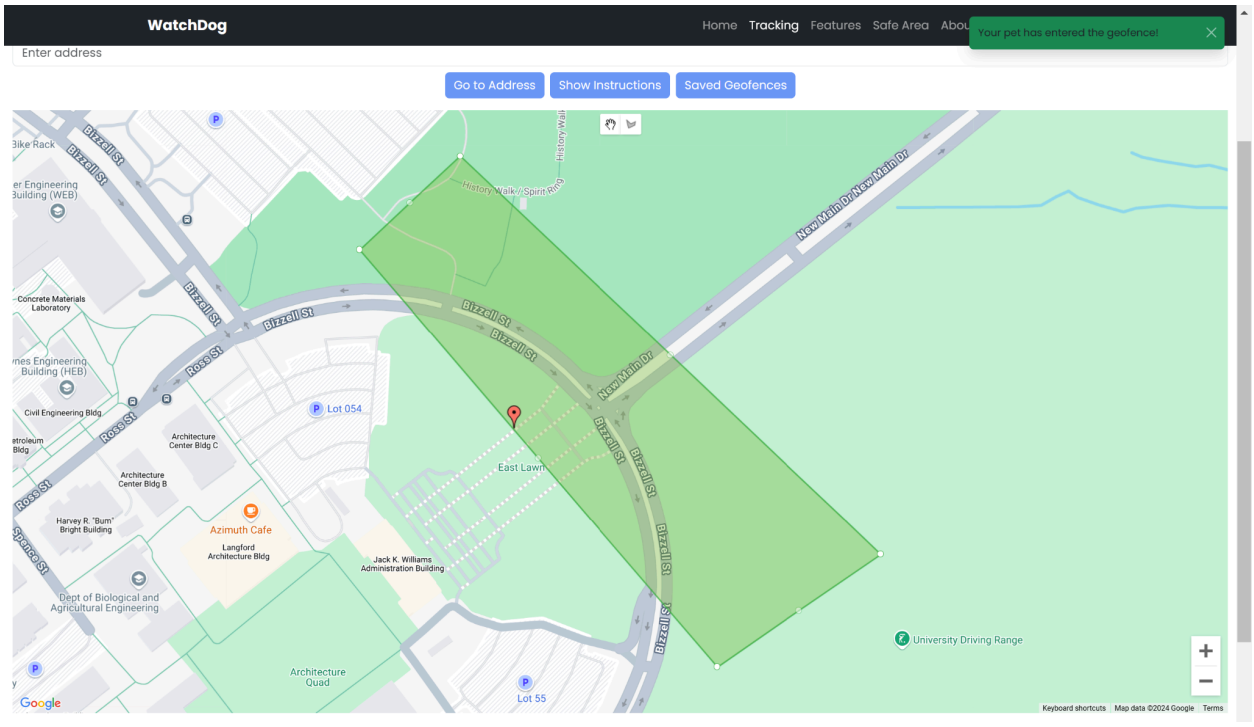


Figure 3.2.3.2: Entering Geofence

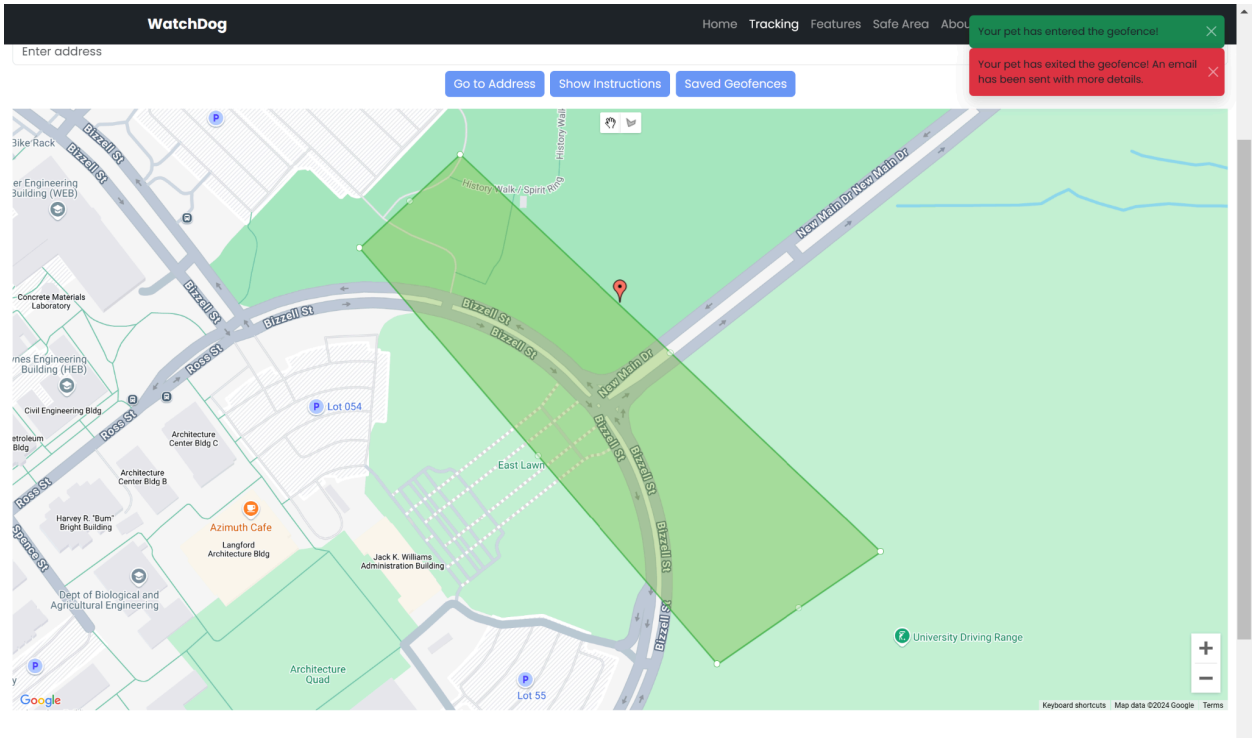


Figure 3.2.3.3: Exiting Geofence

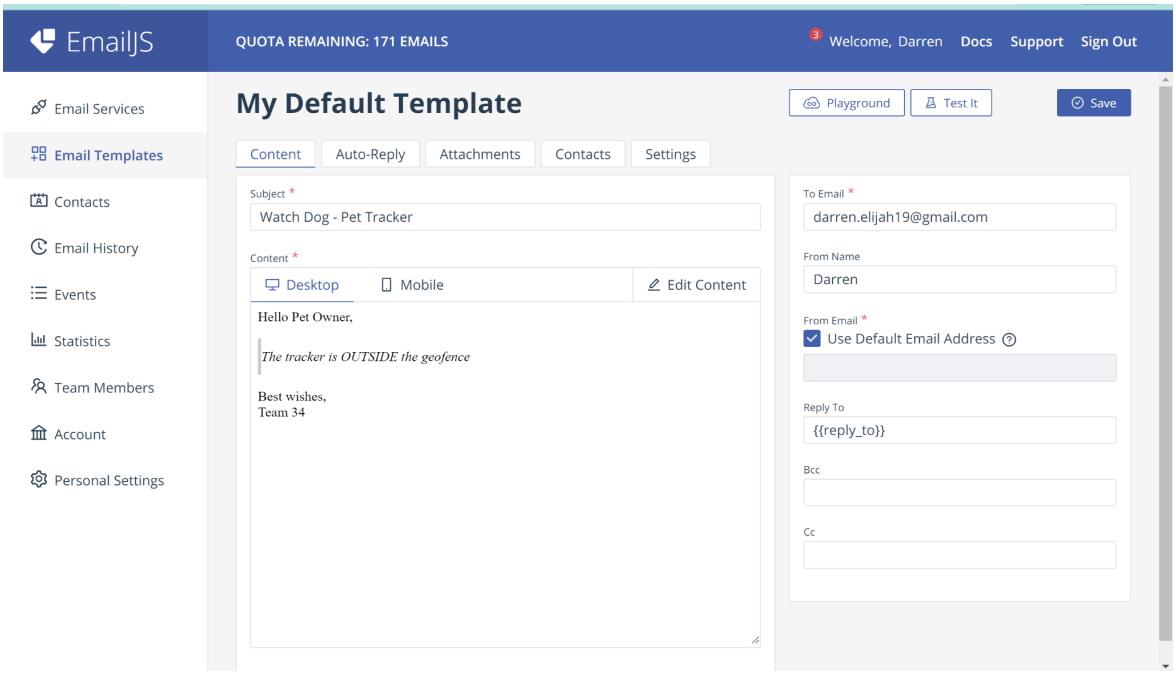


Figure 3.2.3.4: EmailJS template

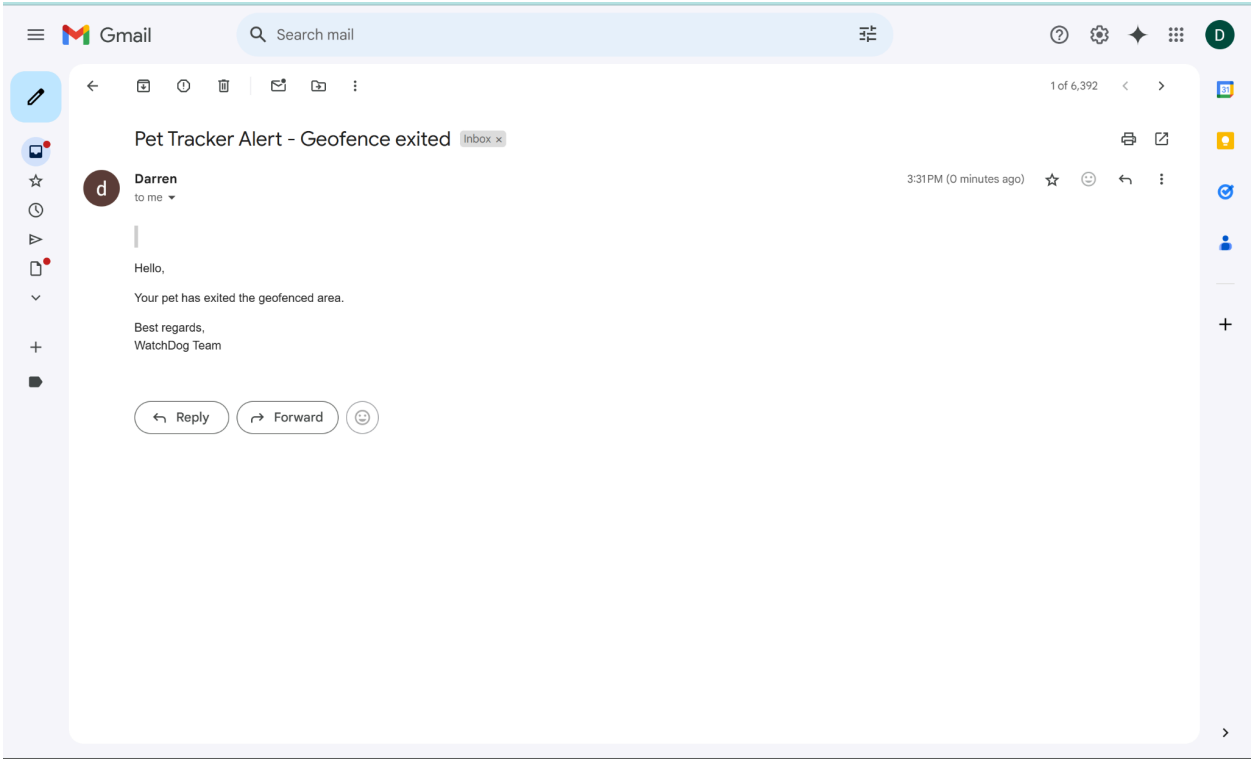


Figure 3.2.3.5: Email sent to the User

3.2.4 Toggle Features

The Feature Activation Page provides users with the capability to control the live camera feed and audio buzzer functionalities through an intuitive interface. Users can toggle these features on or off seamlessly, enhancing the overall user experience and ensuring convenience. The activation and deactivation of the live camera feed are managed by the toggleCamera() function, implemented within the <script> section of the HTML file. When the "Video Feed" button is clicked, this function is triggered, enabling the user to effortlessly toggle the camera feed on or off with a single button press. This streamlined functionality ensures efficient control over the live video feed, contributing to an enhanced and user-friendly experience.

Similarly, the activation of the audio buzzer is facilitated through event listeners and JavaScript functions embedded within the same <script> section. When a user interacts with the "Buzzer" button, an event listener detects the click event and executes the corresponding JavaScript function. This interaction allows users to easily enable or disable the buzzer sound with a single action. Like the camera control, this mechanism enhances the page's usability, providing users with a straightforward and efficient way to manage these critical features.

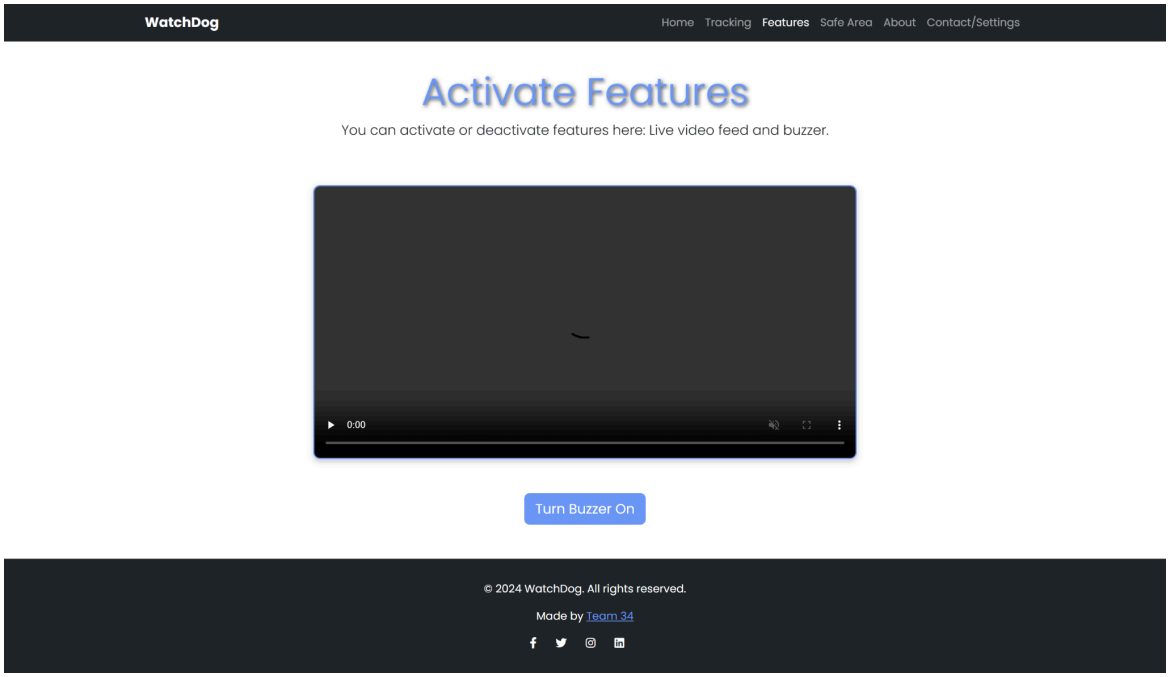


Figure 3.2.4.1: Feature Page

3.2.5 Refined Safe Area

The Refined Safe Area page is a sophisticated feature of the pet tracking application that combines GPS data processing, machine learning, and geospatial analysis to create a highly accurate and reliable geofence. Over seven days, the system collects tracker data and generates geofences for each day using advanced algorithms. These daily geofences are displayed in individual tabs labeled by the days of the week, allowing users to inspect their pet's movement patterns for each specific day. Once the weekly data is fully collected, users can navigate to the Refined Safe Area tab, where the system synthesizes the daily geofences into a single polygonal safe zone. This refined geofence provides a comprehensive representation of the pet's activity zone and can be applied in the Tracking Page for real-time monitoring.

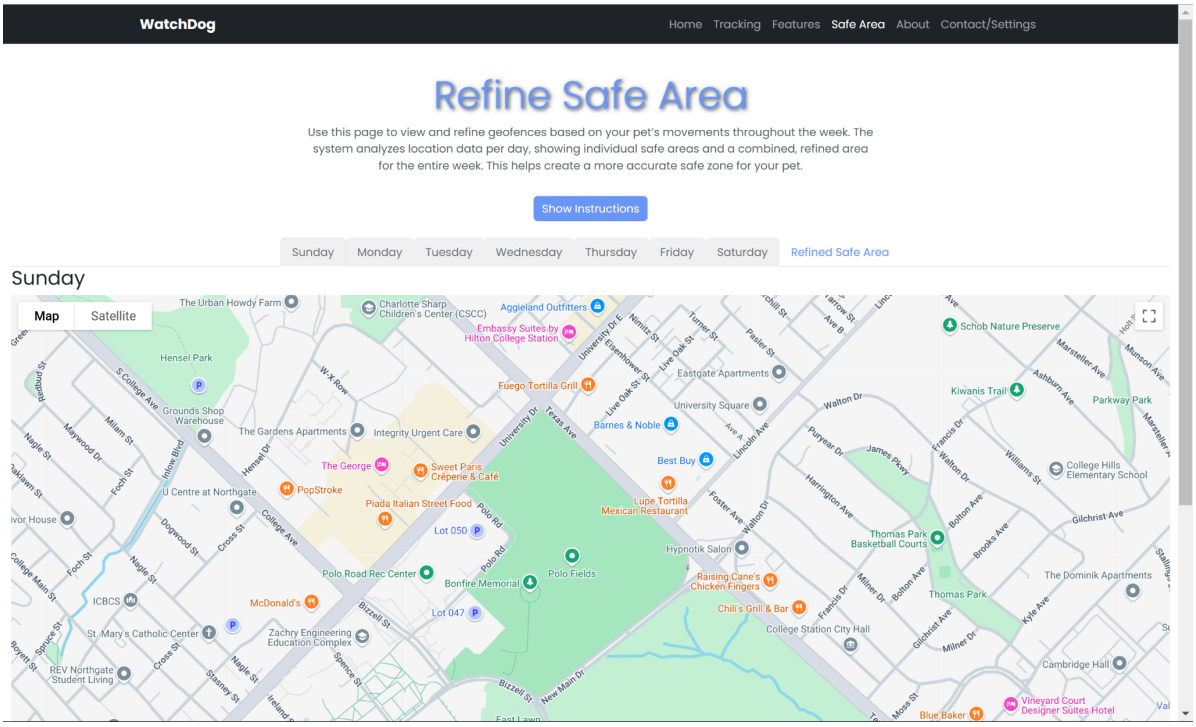


Figure 3.2.5.1: Refine Safe Area Page

A key component of this process is the integration of machine learning to improve data accuracy by eliminating outliers. The application uses the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm to filter out erroneous or anomalous GPS data points before generating geofences. DBSCAN identifies clusters of high-density points while classifying points that fall outside these clusters as noise or outliers. By removing these outliers, the system ensures that the geofences are based solely on meaningful movement data, enhancing the reliability of both daily and refined safe areas.

The workflow for creating the refined safe area begins with daily GPS data processing. Each day's data is cleaned using DBSCAN to exclude outliers, leaving only valid points for geofence generation. The remaining points are then used to construct a convex hull polygon, which defines the boundary encompassing the pet's movements for that day. These polygons are visualized on individual tabs for user review. Once the daily geofences are created, they are aggregated into a combined polygon representing the union of all weekly geofences. This combined polygon undergoes rasterization, where it is converted into a grid. The grid cells are analyzed to determine the probability of each cell being within the geofence, and a Gaussian blur is applied to smooth the probability distribution. The smoothed grid is then thresholded to produce a binary grid, highlighting

the most probable safe areas. Finally, this binary grid is vectorized to yield the refined safe area, which is displayed as a new polygon.

The script to find the average polygon utilizes rasterio, numpy, and scipy libraries to process GPS data collected over seven days, represented as polygonal geofences for each day. Initially, the script creates individual polygons for each day's GPS data. These polygons are then combined into a single polygon representing the union of all seven days. Subsequently, the script rasterizes the combined polygon into a grid, calculating the probability of each grid cell being covered by the polygon. Gaussian blur is applied to smooth the edges of the probability array. The resulting smoothed probability array is thresholded to obtain a binary grid representing areas within the refined safe zone. Finally, this binary grid is converted back into vector format, yielding the simplified average polygon representing the refined safe area. The script thus facilitates the creation of a precise geofence based on the aggregated GPS data, enhancing the accuracy of pet tracking systems.

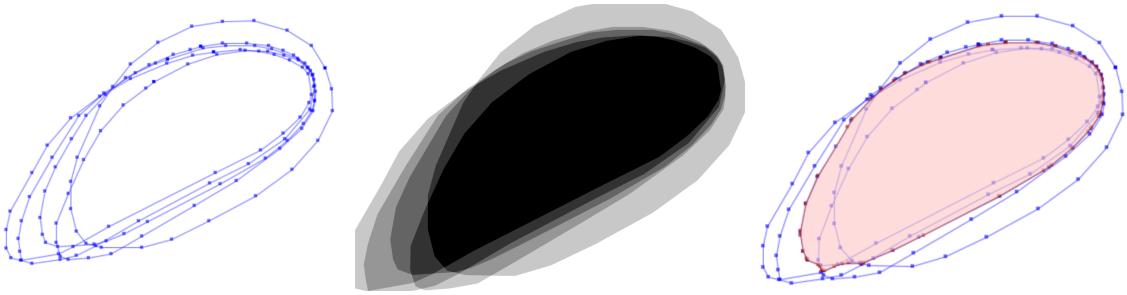


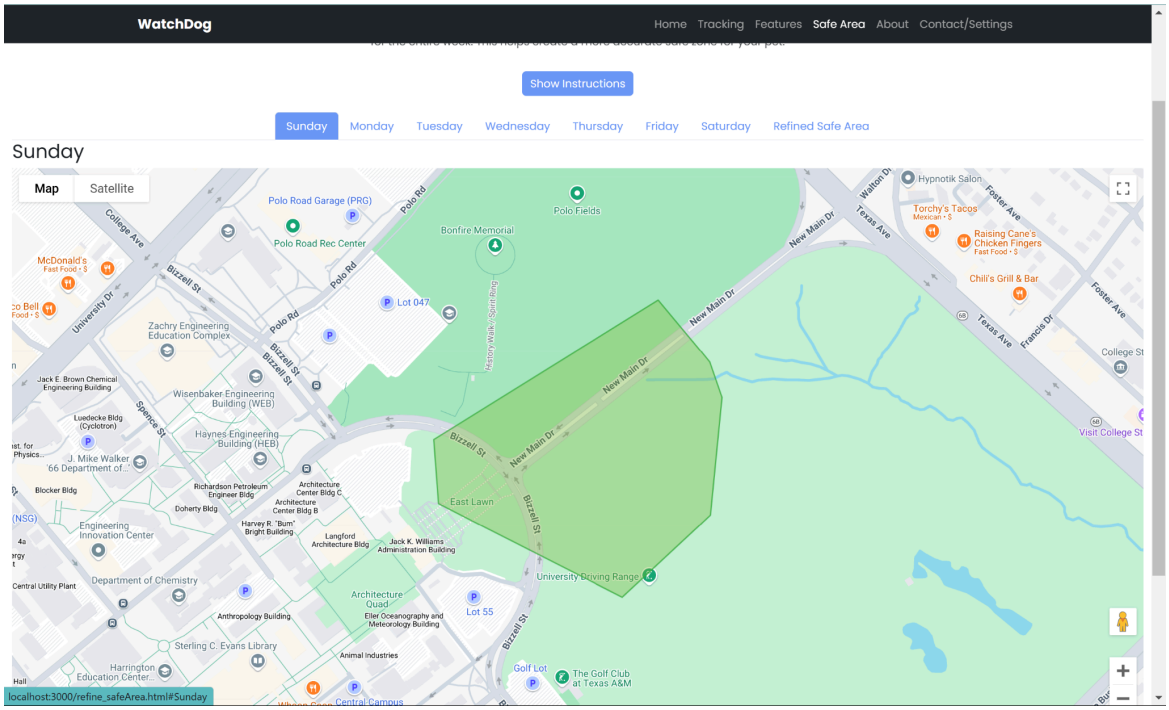
Figure 3.2.5.2: Visual Aid for Steps of the Refined Safe Area Program

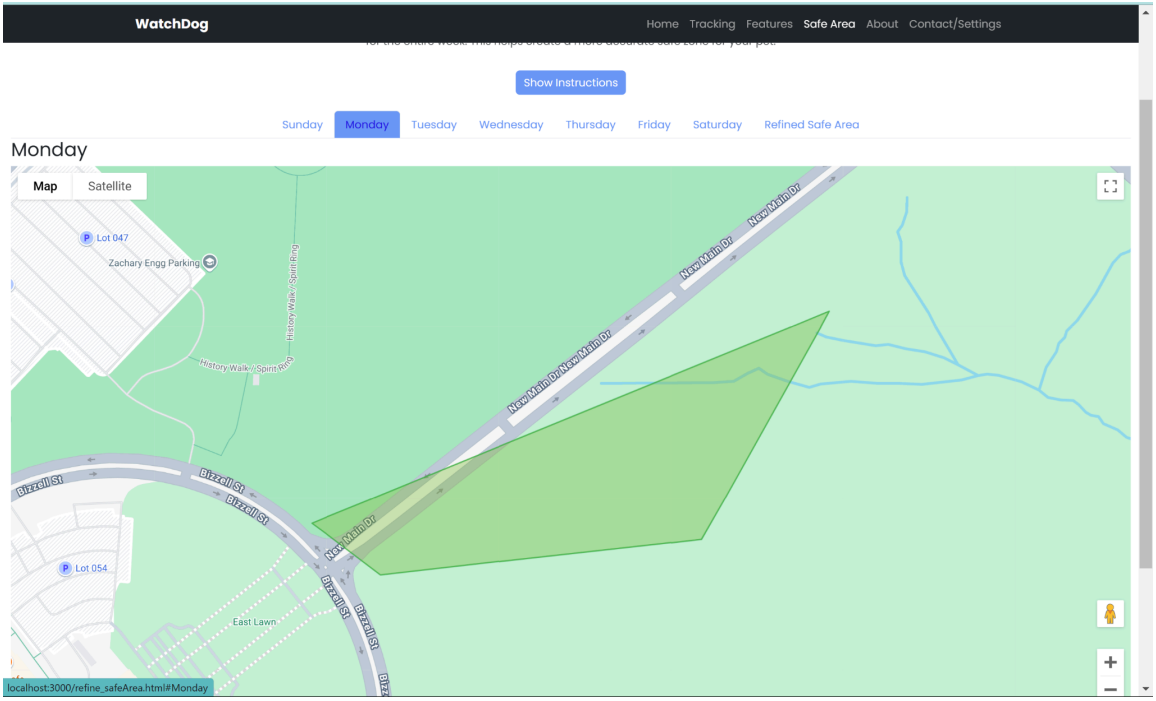
The Refined Safe Area tab displays this consolidated geofence with distinct styling, and users can zoom and pan the map for closer inspection. The refined polygon can be applied in the Tracking Page using the "Use Refined Safe Area" button, seamlessly integrating it with the tracking system's features, such as real-time location monitoring and geofence breach notifications. Users can also switch back to previously created geofences at any time, offering flexibility to adapt the tracking setup to their needs.

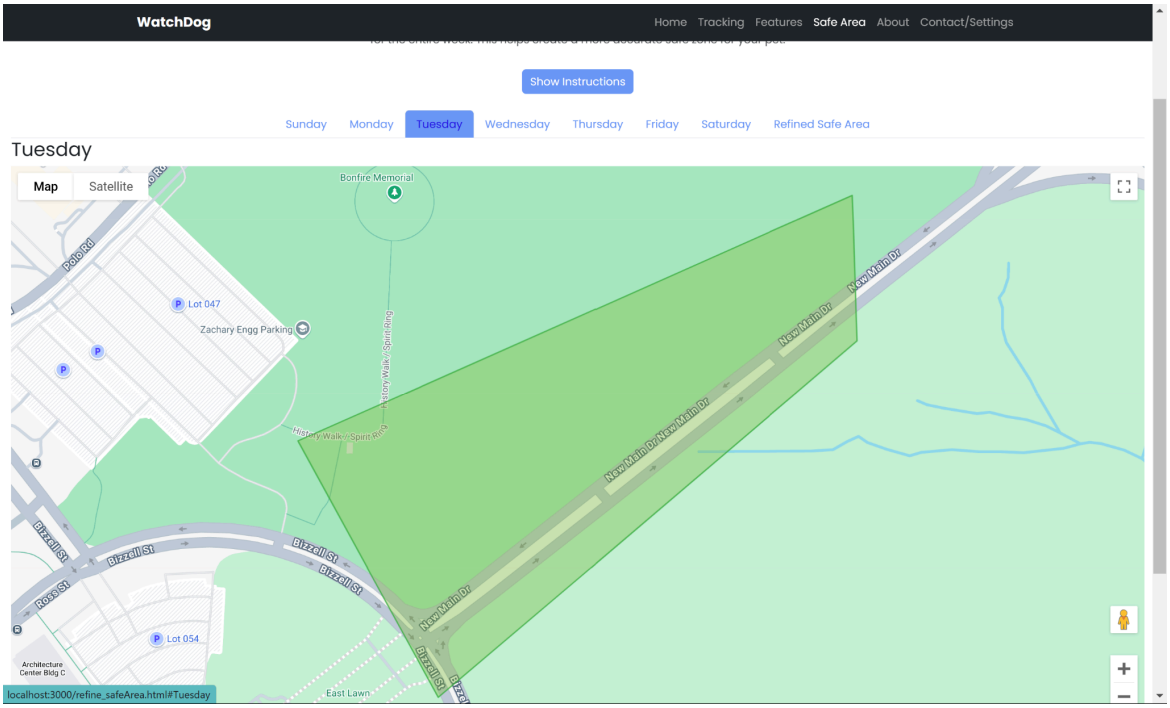
In addition to its geofence generation capabilities, the system includes features to ensure smooth operation and usability. An automated weekly reset occurs at midnight every Saturday, clearing previous data and geofences to prepare for the next week's tracking. Real-time updates fetch new GPS data at regular intervals, dynamically updating the geofences as new information becomes available. The interface also

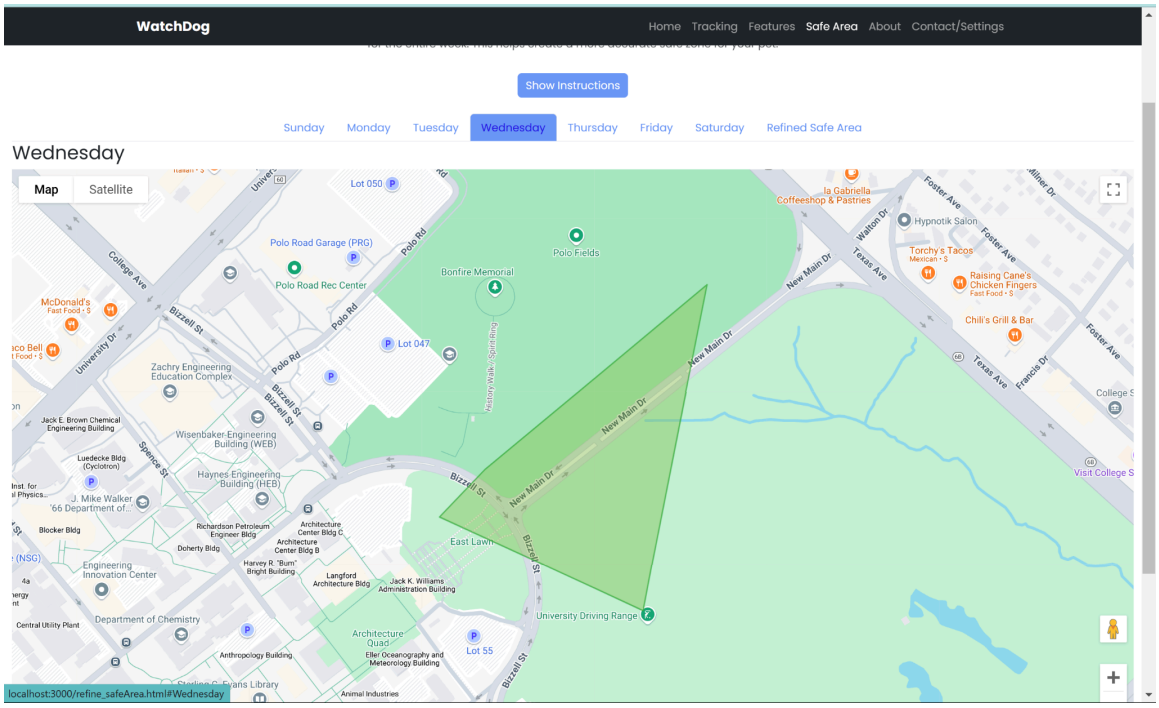
supports dark mode, adapting its appearance to user preferences for optimal usability in different lighting conditions.

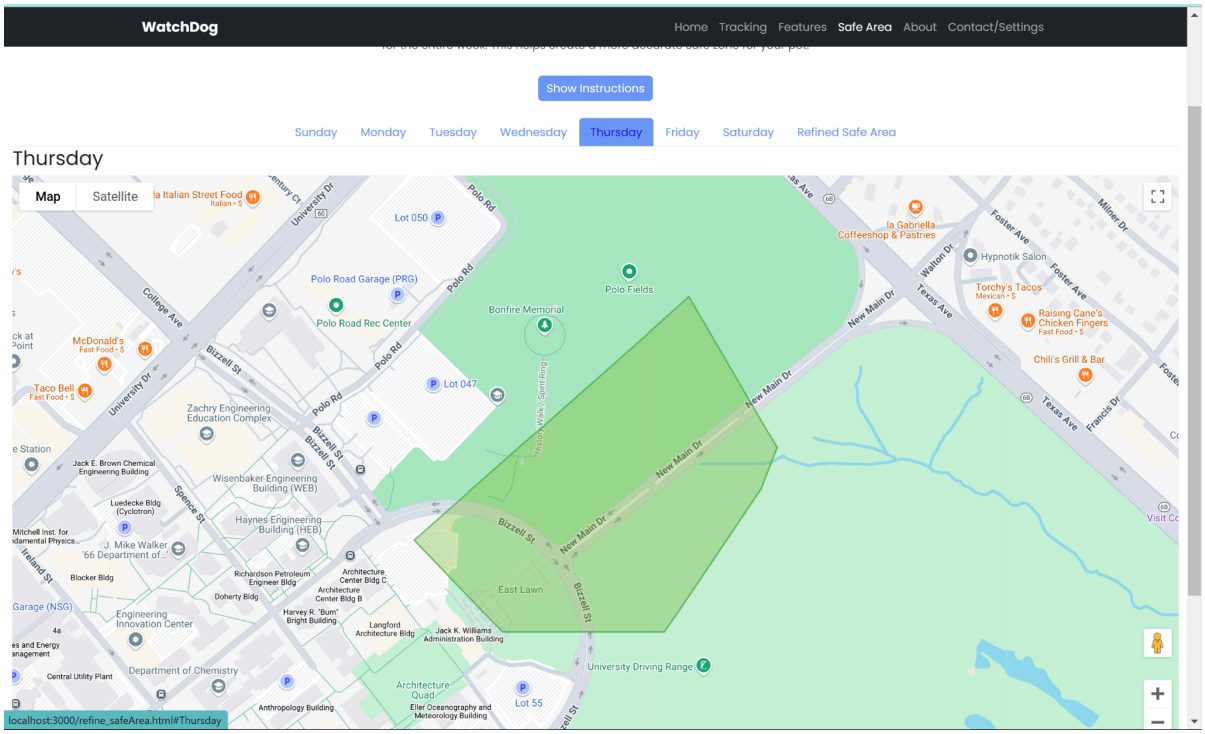
By combining machine learning, geospatial processing, and intuitive visualizations, the Refined Safe Area page provides a cutting-edge solution for pet tracking. The integration of DBSCAN ensures that outliers are effectively removed, resulting in more accurate and reliable geofences. This feature empowers users to monitor their pets with confidence, knowing that the safe zones are precise and tailored to their pet's actual movement patterns.


WatchDog
Home Tracking Features Safe Area About Contact/Settings
Show Instructions
Sunday Monday Tuesday Wednesday Thursday Friday Saturday Refined Safe Area
Sunday
Map Satellite
Polo Road Garage (PRG)
Polo Road Rec Center
Polo Fields
Bonfire Memorial
Lot 047
Zachry Engineering Education Complex
Wisenbaker Engineering Building (WEB)
Haynes Engineering Building (HEB)
Engineering Innovation Center
Department of Chemistry
Architecture Quad
Lot 55
University Driving Range
East Lawn
Sterling C. Evans Library
Golf Lot
The Golf Club at Texas A&M
Central Campus
New Main Dr
Bizzell St
Polo Rd
Texas Ave
College Ave
University Dr
Spence St
Foster Ave
College St
Visit College St
Hypnotik Salon
Torchy's Tacos
Raising Cane's Chicken Fingers
Chili's Grill & Bar
McDonald's
localhost:3000/refine_safeArea.html#Sunday


WatchDog
Home Tracking Features Safe Area About Contact/Settings
Show Instructions
Sunday Monday Tuesday Wednesday Thursday Friday Saturday Refined Safe Area
Monday
Map Satellite
Lot 047
Zachary Engg Parking
Lot 054
East Lawn
History Walk - Spirit Ring
New Main Dr
Bizzell St
localhost:3000/refine_safeArea.html#Monday


WatchDog
Home Tracking Features Safe Area About Contact/Settings
Show Instructions
Sunday Monday Tuesday Wednesday Thursday Friday Saturday Refined Safe Area
Tuesday
Map Satellite
Bonfire Memorial
Polo Rd
Lot 047
Zachary Engg Parking
History Walk - Spirit Ring
New Main Dr
Bizzell St
Ross St
Lot 054
East Lawn
localhost:3000/refine_safeArea.html#Tuesday


WatchDog
Home Tracking Features Safe Area About Contact/Settings
Show Instructions
Sunday Monday Tuesday Wednesday Thursday Friday Saturday Refined Safe Area
Wednesday
Map Satellite
Lot 050
Polo Road Garage (PRG)
Polo Road Rec Center
Polo Fields
Bonfire Memorial
Lot 047
Zachry Engineering Education Complex
Wisenbaker Engineering Building (WEB)
Haynes Engineering Building (HEB)
Engineering Innovation Center
Department of Chemistry
Architecture Quad
Lot 55
University Driving Range
East Lawn
New Main Dr
Bizzell St
Polo Rd
College Ave
University Dr
Texas Ave
Foster Ave
Hypnotik Salon
Chili's Grill & Bar
Raising Cane's Chicken Fingers
Torchy's Tacos
McDonald's
localhost:3000/refine_safeArea.html#Wednesday


WatchDog
Home
Tracking
Features
Safe Area
About
Contact/Settings
Show Instructions
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Refined Safe Area
Thursday
Map
Satellite
Polo Road Garage (PRG)
Polo Road Rec Center
Polo Fields
Bonfire Memorial
Lot 050
Lot 047
Lot 55
McDonald's
Taco Bell
Zachry Engineering Education Complex
Wisenbaker Engineering Building (WEB)
Haynes Engineering Building (HEB)
J. Mike Walker '66 Department of...
Engineering Innovation Center
Department of Chemistry
Architecture Quad
University Driving Range
East Lawn
Ia Gabriella Coffeeshop & Pastries
Hypnotik Salon
Torchy's Tacos
Raising Cane's Chicken Fingers
Chili's Grill & Bar
College Ave
University Dr
Bizzell St
Polo Rd
New Main Dr
Texas Ave
Foster Ave
Walton Dr
Milner Dr
Francis Dr
Spence St
Ireland St
localhost:3000/refine_safeArea.html#Thursday

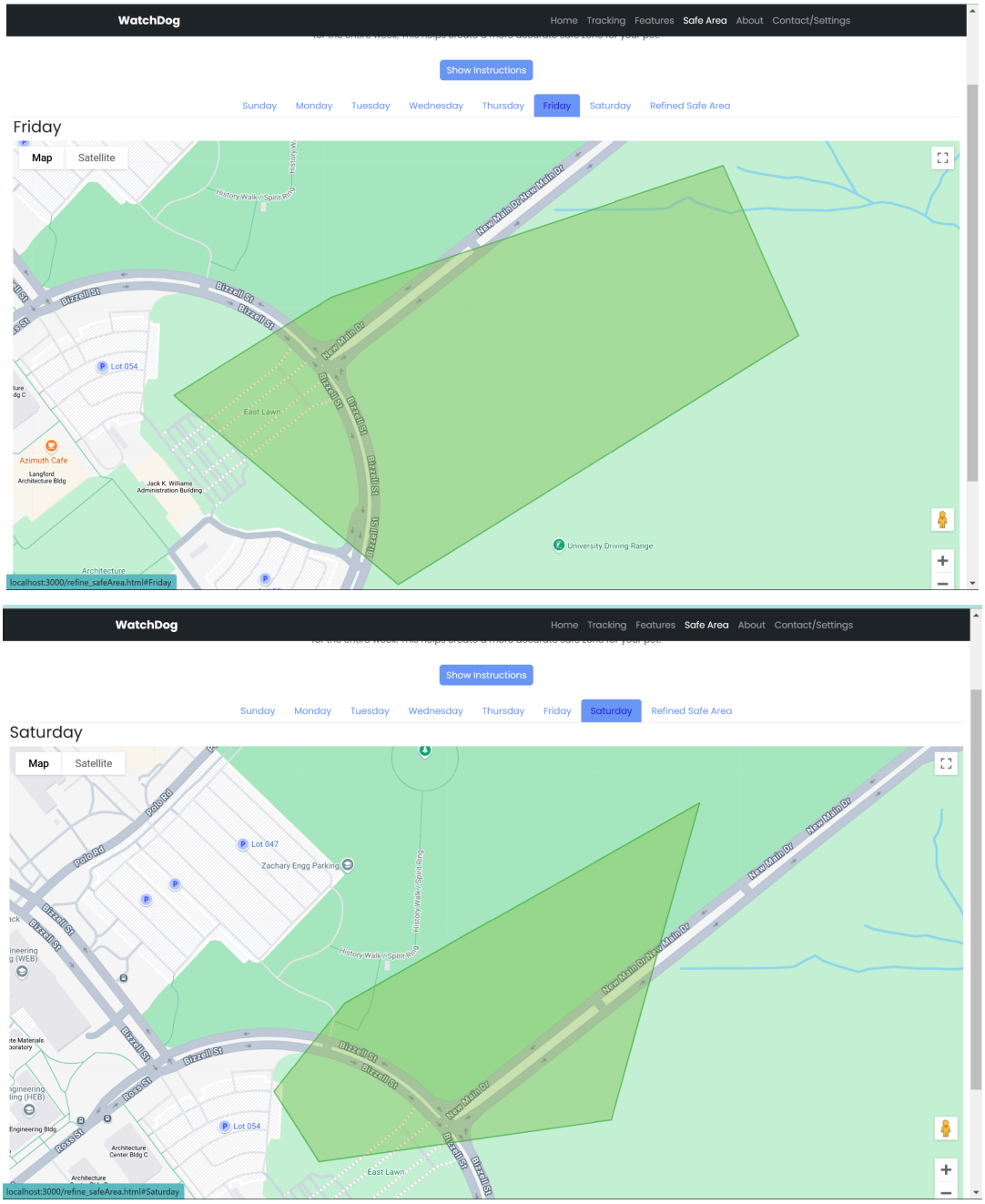


Figure 3.2.5.3: 7 days of Geofence Data

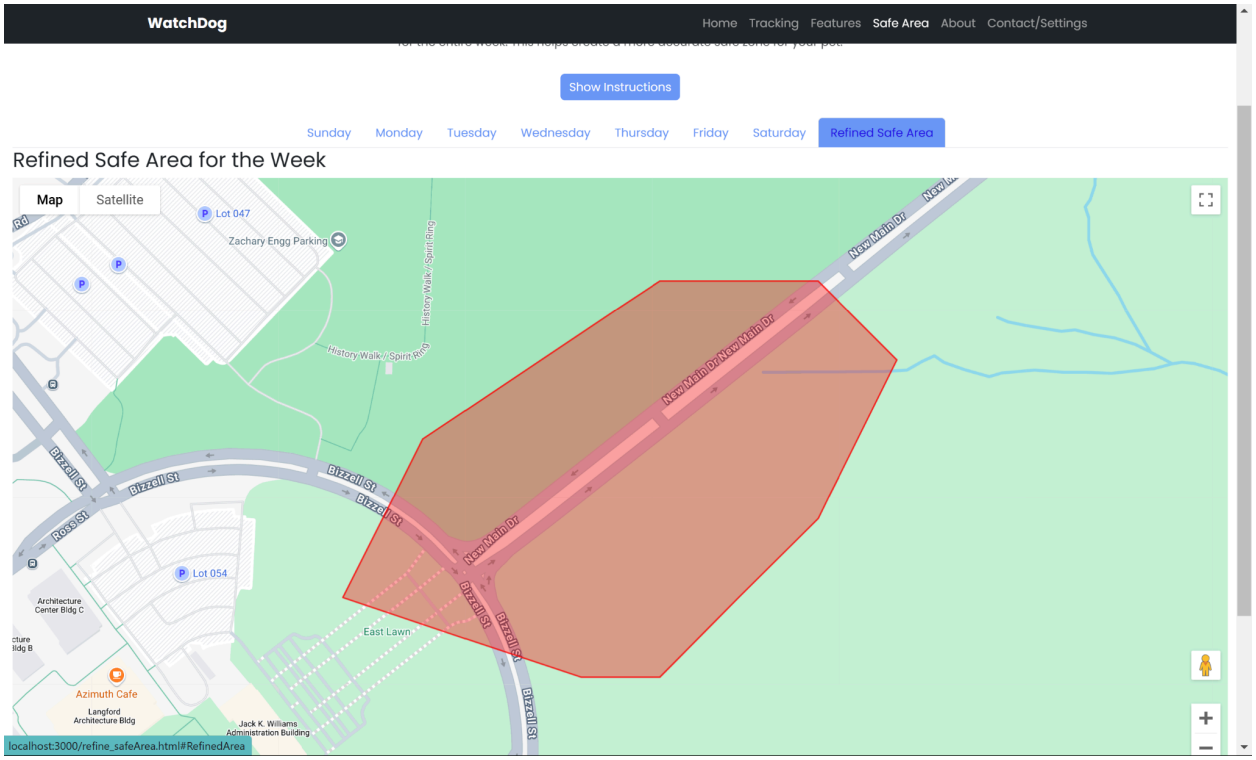


Figure 3.2.5.4: Refined Safe Area using different algorithms

3.2.6 Machine Learning

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a powerful clustering algorithm that identifies clusters in spatial data based on density while effectively isolating outliers. Unlike traditional clustering methods, DBSCAN does not require the number of clusters to be defined beforehand. Instead, it groups points into clusters based on two parameters: epsilon (eps), which specifies the radius of a point's neighborhood, and minPts, the minimum number of points required to form a dense cluster. During the clustering process, points are classified as core points, border points, or noise points (outliers). Core points have at least minPts neighbors within the radius of eps and serve as the foundation for clusters. Border points lie within the neighborhood of a core point but lack enough neighbors to form a cluster on their own. Points that fail to meet either criterion are considered noise and are excluded from clusters.

DBSCAN Clustering Demo - Texas A&M University

Clustering Visualization

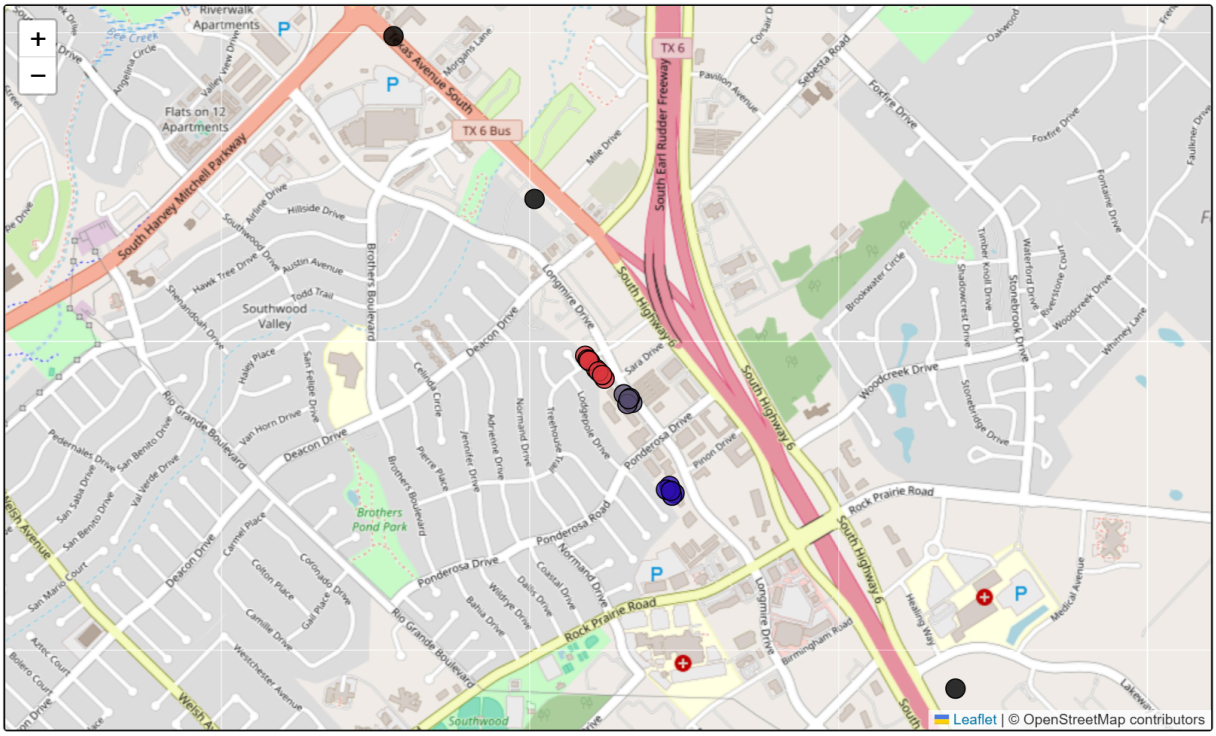


Figure 3.2.6.1: All points displayed for DBSCAN

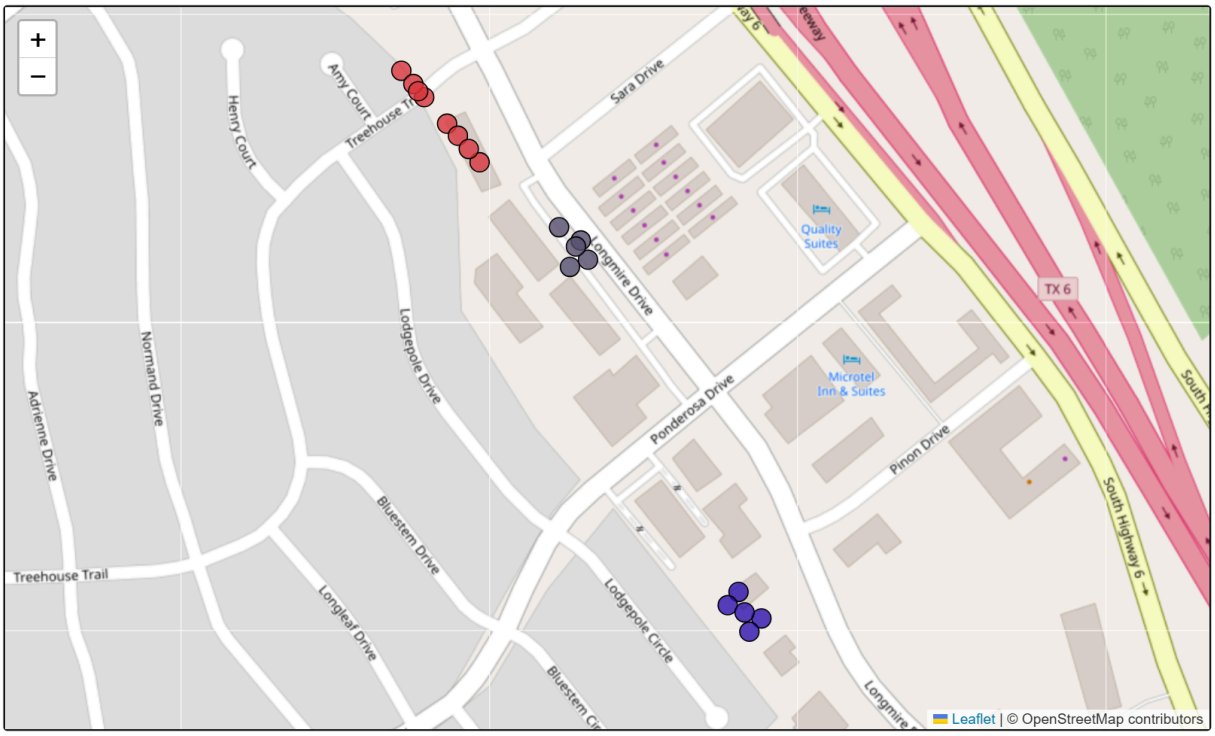


Figure 3.2.6.2: Clustered points displayed after DBSCAN

In the implementation, points are assigned numerical labels to indicate their status. Unprocessed points are initially labeled as 0. When a core point is identified, it is assigned a positive cluster ID, and the cluster is expanded to include all density-reachable points. If a point does not meet the density requirements, it is labeled as -1, designating it as an outlier or noise point. This approach ensures a clear distinction between clustered points and outliers. Each cluster is assigned a unique positive integer (e.g., 1, 2, 3), while noise points retain the -1 label. This labeling scheme allows for efficient filtering of data and the isolation of meaningful clusters from irregular or erroneous points.

Clustering Results

Latitude	Longitude	Cluster ID
30.588756	-96.291539	0
30.589000	-96.291800	0
30.588900	-96.291600	0
30.588700	-96.291700	0
30.588850	-96.291650	0
30.586000	-96.290000	1
30.586200	-96.290200	1
30.585900	-96.290100	1
30.586100	-96.290300	1
30.586050	-96.290150	1
30.590000	-96.293000	2
30.590200	-96.293200	2
30.589800	-96.292800	2
30.590100	-96.293100	2
30.590050	-96.293050	2
30.589500	-96.292500	2
30.589700	-96.292700	2
30.589600	-96.292600	2
30.600000	-96.300000	-1
30.580000	-96.280000	-1
30.595000	-96.295000	-1

Figure 3.2.6.3: Points inputted into table after DBSCAN

In the Refined Safe Area tab, DBSCAN is a critical preprocessing step for creating refined geofences. The algorithm clusters GPS data points and filters outliers by removing those labeled as -1. These outliers often result from anomalous or sparse data and are excluded from the geofence generation process. Once the valid clusters are identified, their points are used to construct daily geofences through convex hull algorithms, which define the boundaries around each cluster. These daily geofences are then aggregated to form a refined safe area for the week. By isolating noise and processing only meaningful clusters, DBSCAN ensures that the generated geofences are accurate and reliable. This refined geofence provides a precise representation of the pet's movement patterns, enhancing the system's ability to monitor and protect the pet effectively.

Console Output

```
Begin DBSCAN clustering using JavaScript

Data:
      30.588756       -96.291539
      30.589000       -96.291800
      30.588900       -96.291600
      30.588700       -96.291700
      30.588850       -96.291650
      30.586000       -96.290000
      30.586200       -96.290200
      30.585900       -96.290100
      30.586100       -96.290300
      30.586050       -96.290150
      30.590000       -96.293000
      30.590200       -96.293200
      30.589800       -96.292800
      30.590100       -96.293100
      30.590050       -96.293050
      30.589500       -96.292500
      30.589700       -96.292700
      30.589600       -96.292600
      30.600000       -96.300000
      30.580000       -96.280000
      30.595000       -96.295000


Clustering with epsilon = 50 meters and min points = 3
Clustering done.

Clustering results:
  0   0   0   0   0   1   1   1   1   1   2   2   2   2   2   2   2  -1  -1  -1
```

Figure 3.2.6.4: DBSCAN data and parameters in console output

3.2.7 About/Contacts and Settings Page

The About page further explains the product in more detail. The Contacts/Settings web page is a component of the WatchDog pet tracking application, designed to allow users to manage their personal settings and access contact information. This page enables users to view and edit their registered email address, which is essential for receiving notifications and updates. The email can be updated through a modal form that validates the input and stores it using localStorage, ensuring consistency across the application. Additionally, the page offers a theme toggle feature, allowing users to switch between light and dark modes, with their preference saved in localStorage and applied throughout the site. The page also includes contact information for team members, providing direct links to their email addresses for user support. A "Contact Support" modal is available for users to send messages directly to the support team via EmailJS, facilitating prompt assistance with any issues or inquiries. For account management, a logout button is provided, which communicates with the server to terminate the user's session and redirects them to the login page upon successful logout. The interface is enhanced with responsive design using Bootstrap, animations through Animate.css and AOS (Animate On Scroll), and interactive elements like toast notifications to inform users about the status of their actions, such as successful email updates or theme

changes. Overall, the page serves as a centralized hub for users to personalize their experience and access support within the WatchDog application.

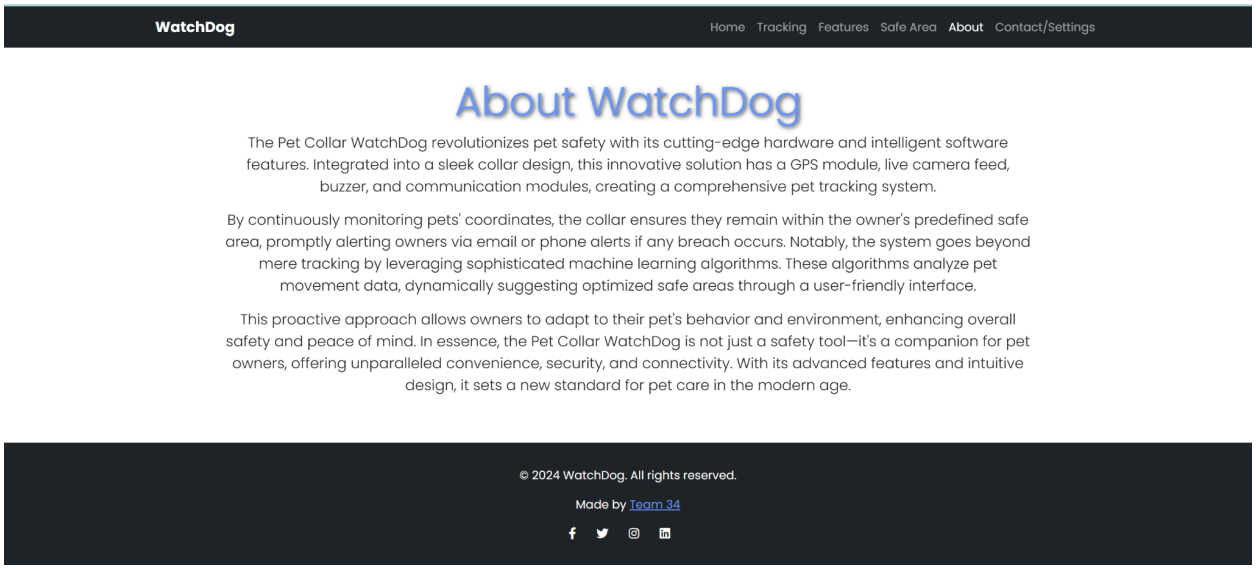


Figure 3.2.7.1: About Page

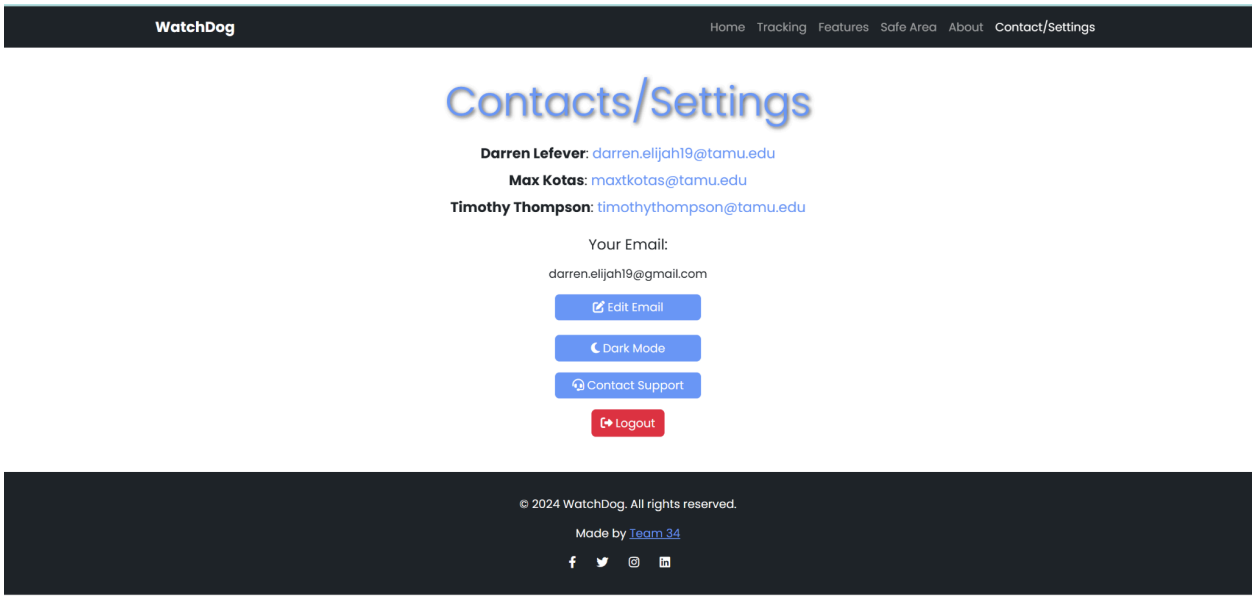


Figure 3.2.7.2: Contacts/Settings Page

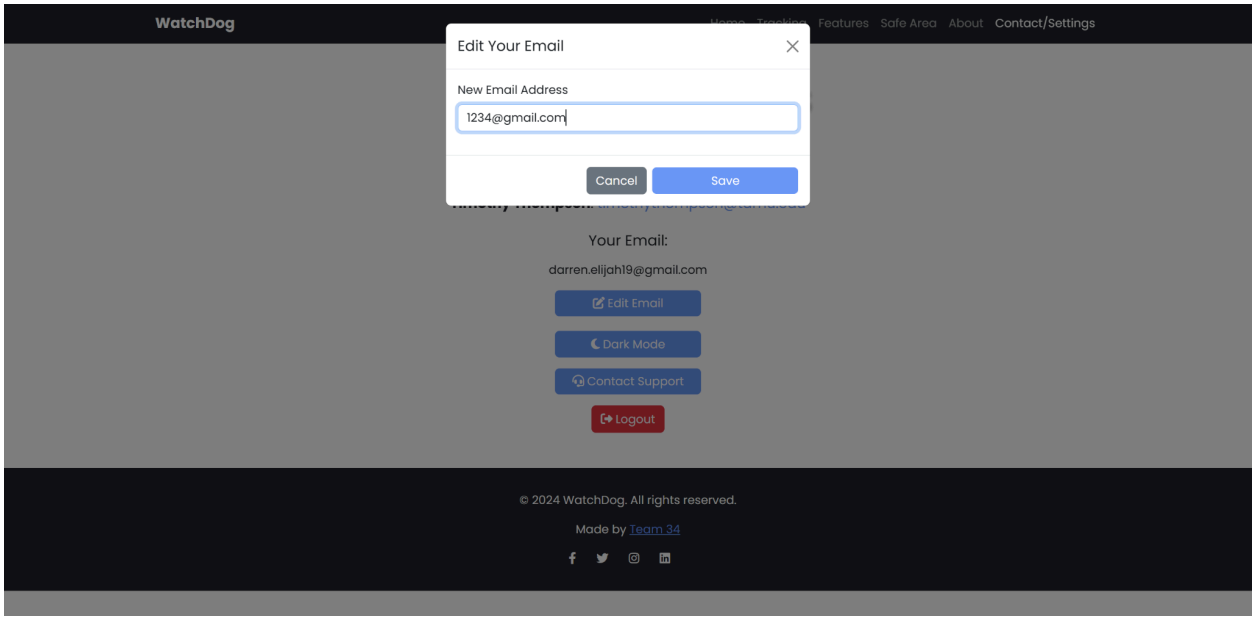


Figure 3.2.7.2: Edit Email Modal

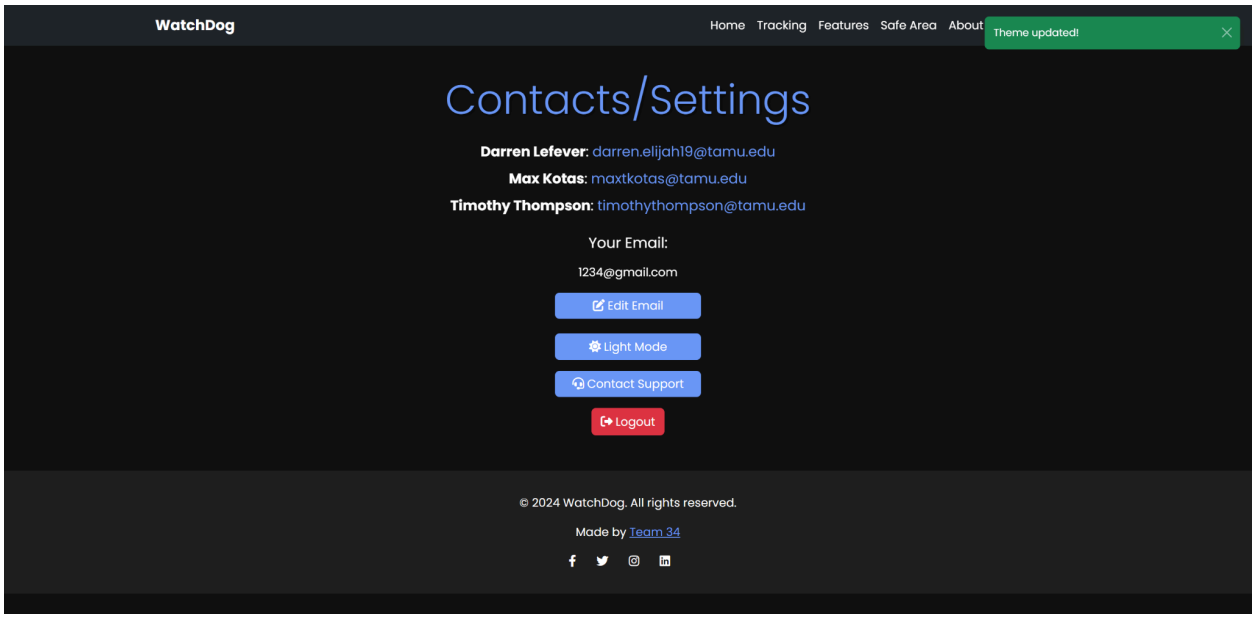


Figure 3.2.7.3: Theme Toggle

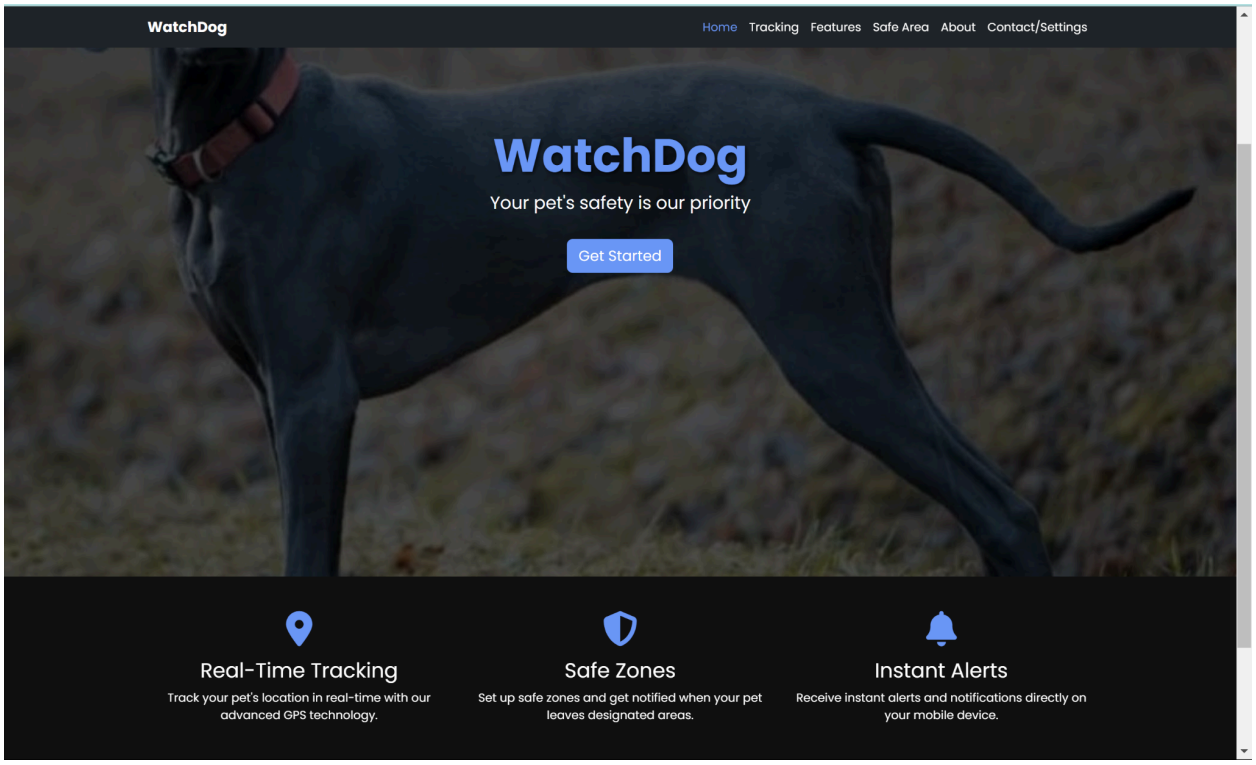


Figure 3.2.7.4: Dark Mode Homepage

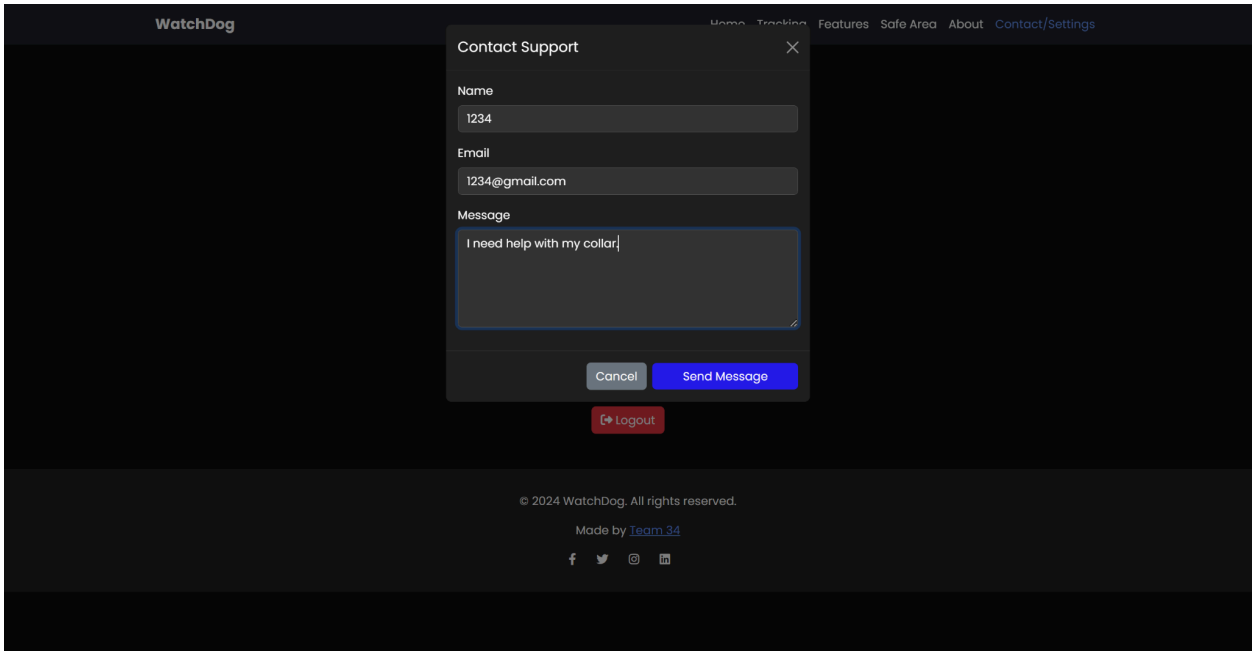


Figure 3.2.7.5: Contact Support for user -> team communication

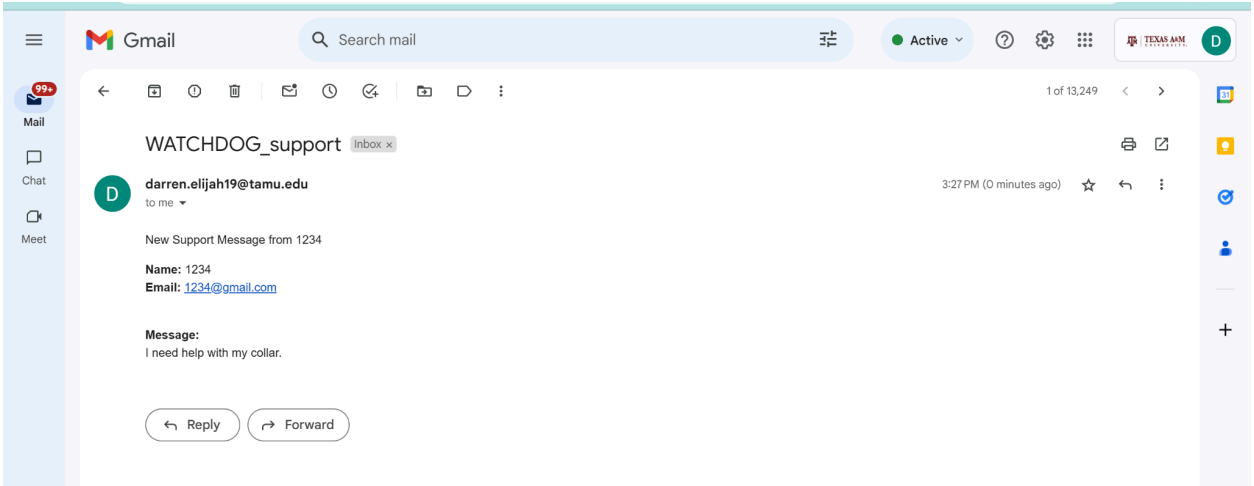


Figure 3.2.7.6: Email from User using contact support

3.2.8 User Authentication and Login

The WatchDog pet tracking application enables users to create and log into personal accounts, providing a secure and personalized experience for managing geofences. The website features two main pages for authentication: register.html for new user registration and login.html for existing users. On the registration page, users can sign up by entering a unique username, a valid email address, and a password. Client-side validation ensures that all inputs meet the required criteria before submission. Upon successful registration, the server securely stores the user credentials, initiates a user session, and redirects the user to the homepage.
The login page allows existing users to access their accounts by entering their username and password. The server validates the credentials and establishes a user session upon successful authentication. If authentication fails, the user receives immediate feedback to correct any issues.Once logged in, users can create, manage, and save geofences that are specifically associated with their accounts. The application securely stores geofence data in a user-specific context, ensuring that each user's geofences are only accessible to them. This personalization enhances security and allows for tailored tracking setups based on individual preferences.

Additionally, users can edit their email addresses and adjust settings such as theme preferences through the Contacts/Settings page. Changes are saved and applied consistently throughout the application. A logout function is provided to securely terminate user sessions and protect personal data when the application is not in use.By integrating user authentication with geofence management, the WatchDog application ensures that users can securely manage their pet tracking settings in a personalized environment, with all data protected and associated with their individual accounts.

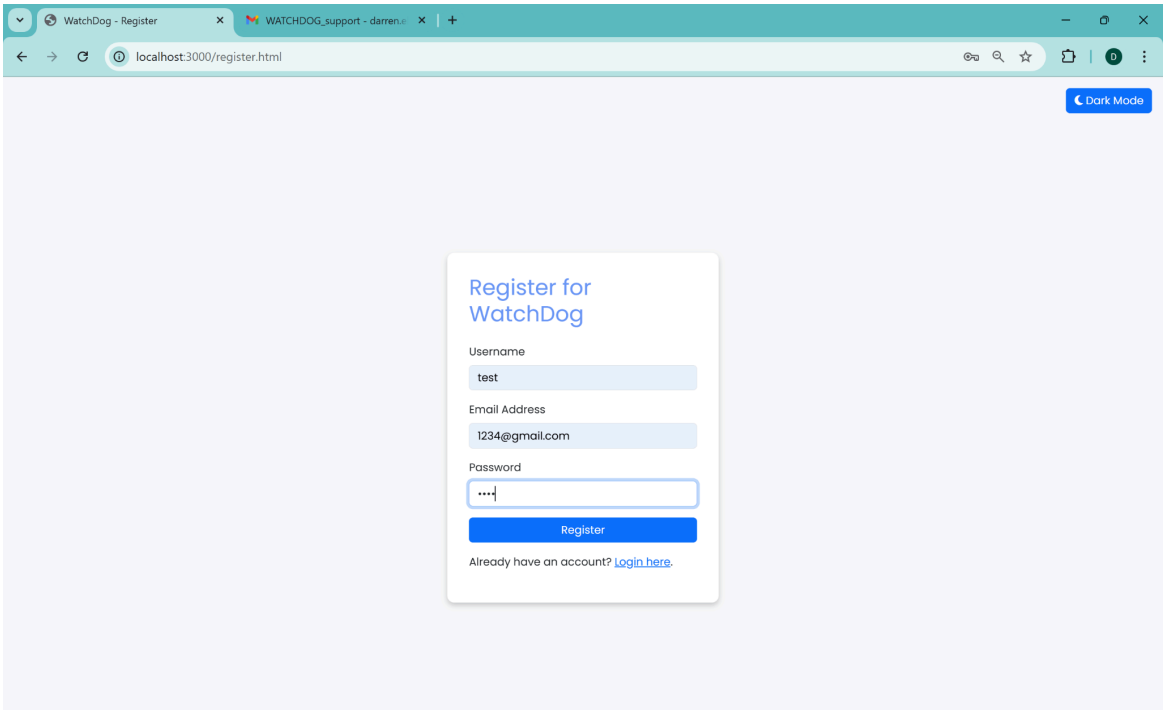


Figure 3.2.8.1: Register for website

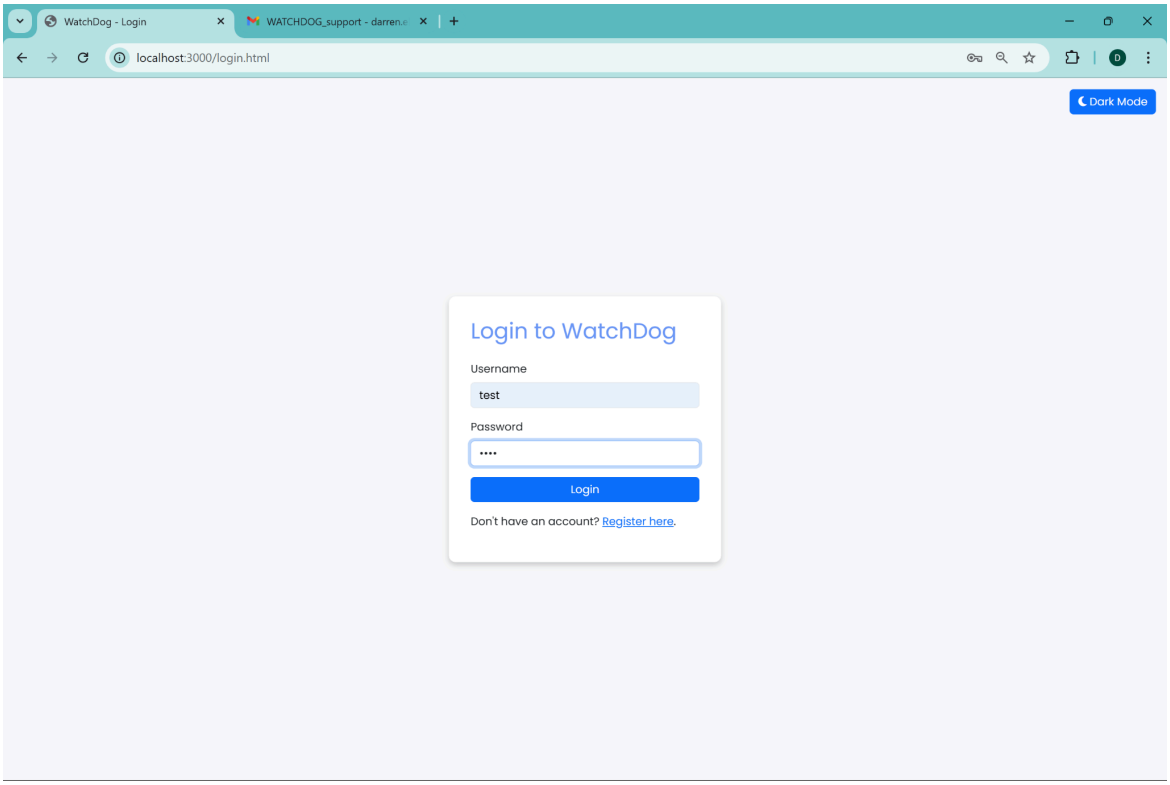


Figure 3.2.8.2: Register for website

3.3 Subsystem Validation

To validate the website's functionality, various testing approaches were employed. First, unit testing was conducted for individual components, such as the geofencing feature, live tracking, email notification system, and toggle features. This involved testing each function and method to ensure they produce the expected outputs under different conditions, such as inputting different addresses and radiuses, and viewing the video feed and hearing the buzzer. Integration testing was then performed to evaluate the interactions between different components and ensure seamless operation across the entire system. For example, the integration of Google Maps API with geocoding functionality was thoroughly tested to ensure accurate conversion of user-defined addresses into geographic coordinates. Additionally, end-to-end testing was carried out to simulate real-world scenarios and validate the complete user experience, from creating a safe area to receiving email notifications for breaches. Continuous monitoring and user feedback were also utilized to identify any issues or areas for improvement, ensuring that the user interface subsystem meets the users' needs effectively and efficiently. Overall, subsystem validation served as a comprehensive approach to ensure the reliability, functionality, and usability of the pet collar's user interface.

3.4 Subsystem Conclusion

In conclusion, the user interface subsystem of the pet collar system has been designed and validated to provide users with a seamless and intuitive experience. Through rigorous testing and validation processes, each functionality, including geofencing, live tracking, notification systems, and toggle features, has been thoroughly evaluated to ensure reliability and accuracy. The integration of various technologies, such as Google Maps API, geocoding, and EmailJS, enhances the system's capabilities and usability. By prioritizing user feedback and continuous improvement, the user interface subsystem delivers on its promise of facilitating effective pet monitoring and ensuring pet safety. During this semester's validation phase, I encountered challenges due to the lack of communication with other subsystems. To overcome this limitation, I had to generate my own data, including coordinate data, and utilize resources available to me, such as the live location of my laptop and its camera. Additionally, I incorporated an audio file for testing purposes. These adaptations allowed me to validate the functionalities of the user interface subsystem effectively, despite the constraints imposed by the absence of access to other subsystems.

4. Pet Collar Subsystem Report

4.1 Subsystem Introduction

The pet collar subsystem serves as the central hardware component of the IoT-based Pet Tracker project. This subsystem integrates various hardware components to ensure seamless and low-power real-time tracking of pets. At its core, the Collar Subsystem comprises several essential hardware modules, including an ESP32-S2 microcontroller, Quectel EC25 cellular and GPS module, camera, and a buzzer. These components collaborate to offer efficient tracking of the pet's location, instant alerts in case of straying, and video streaming.

GITHUB Link: https://github.com/TimothyThompson1/capstone-iot-pet-tracker

4.2 Subsystem Details and Features

4.2.1 Microcontroller: Espressif ESP32-S2 Platform

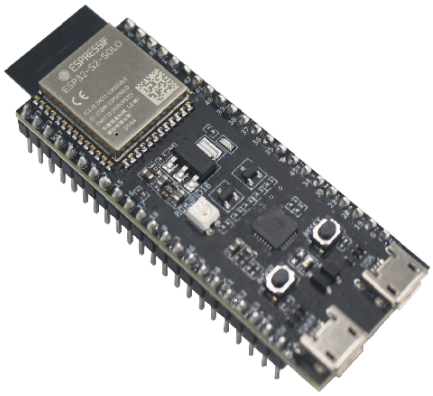


Figure 4.2.1: An Espressif ESP32-S2 Developer Board

The ESP32 microcontroller, known for its low power consumption and robust processing capabilities, forms the backbone of the collar subsystem. The MCU handles communication protocols (MQTT, RTSP), and manages other peripherals, including the GPS, camera, and buzzer. Its low power consumption and high processing power make it ideal for continuous real-time monitoring without draining the battery quickly. The integrated Wi-Fi antenna facilitates relatively low-power communication with the receiver subsystem, ensuring timely data transmission and control functionalities.

4.2.2 Quectel EC25 Mini PCIe

Figure 4.2.2.1: A Queltec EC25 Mini PCIe module

The collar subsystem integrates the Quectel EC25 module, a versatile component that seamlessly manages cellular connectivity and location tracking. Chosen for its exceptional features including a low-power sleep mode, robust GPS connectivity, and high data rate capacity, the EC25 module ensures reliable and efficient real-time data transmission for pet tracking and monitoring. Its cellular capabilities provide uninterrupted connectivity, ensuring system functionality regardless of the pet's location. With a Cat 4 LTE designation and a 50 Mbps upload speed, this module excels in wireless communication, empowering the collar system with full-featured functionality and responsive data handling.

Moreover, the integration of GPS and cellular modules into one unit reduces unnecessary bulk and weight on the collar, enhancing the overall comfort and convenience for pets while maintaining the system's advanced capabilities. This streamlined design not only improves the aesthetics of the collar but also optimizes its performance, making it an ideal solution for pet owners seeking efficient and reliable pet tracking and safety features.

4.2.3 Camera

Figure 4.2.3.1: ESP32-CAM

The collar subsystem now features the ESP32-CAM module with OV2640 sensor, significantly improving video streaming functionality. This module replaces the OV5642 ArduCAM with onboard image processor, which introduced issues during integration stemming from lack of proper documentation. The new module includes a dedicated microcontroller to handle image processing and RTSP, while improving overall stream stability and latency. The camera maintains a small form factor, ensuring it seamlessly integrates into the collar without adding unnecessary bulk.

4.2.4 Buzzer

Figure 4.2.4.1: Buzzer Module

An audible buzzer serves multiple functions, including aiding in pet retrieval by emitting sound signals for location assistance. In case of a breach or unauthorized exit from safe zones, the buzzer automatically activates to alert nearby individuals and facilitate quick response. The module is activated through MQTT commands from the receiver subsystem. The user interface allows manual control of the buzzer, providing pet owners with additional features for ensuring the safety of their pets.

4.2.5 Custom PCB

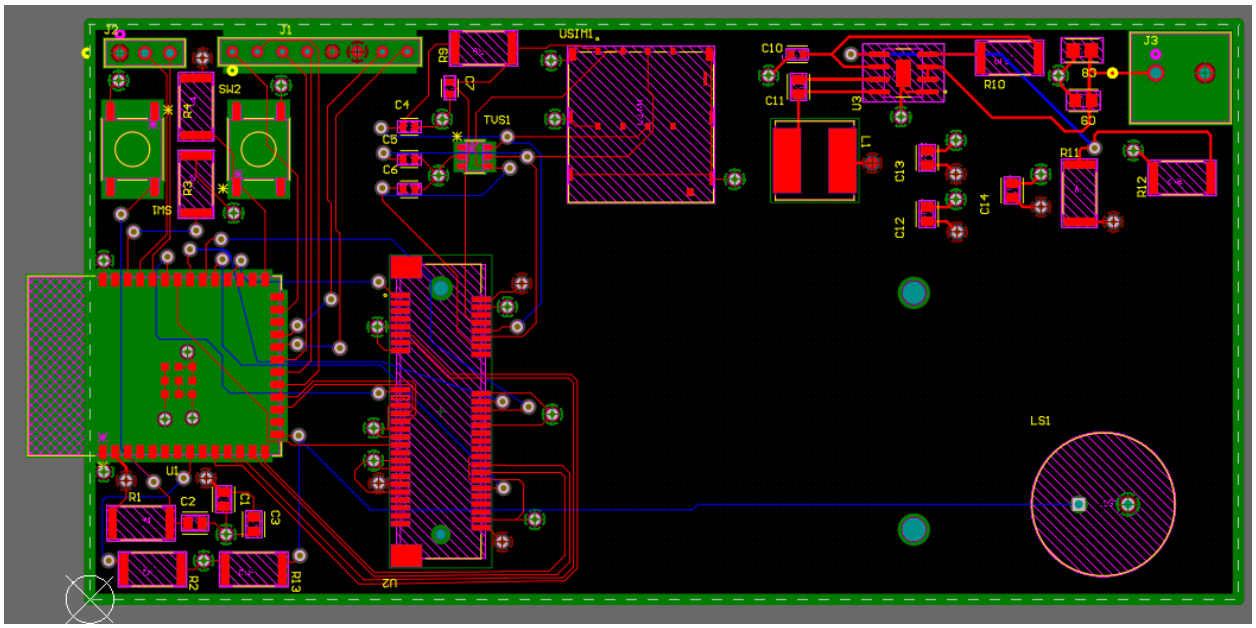


Figure 4.2.5.1: Custom PCB 2D

The custom PCB is pivotal within the collar subsystem, acting as a central hub that efficiently integrates and organizes essential hardware components like the ESP32 microcontroller, Quectel EC25 cellular module, ESP32-CAM, and other critical elements. Its tailored design ensures optimized space utilization, minimal signal interference, and streamlined connections, all of which collectively enhance the reliability and performance of the pet tracking system.

However, it's worth noting that this phase of development posed significant challenges and consumed considerable time. With no prior experience in PCB design or the Altium software, I encountered numerous obstacles along the way, necessitating external assistance to overcome them. Consequently, many of the subsequent validation steps experienced notable delays due to these challenges. Despite these hurdles, the custom PCB successfully integrates all of the peripheral modules indirectly with the user interface and remains a fundamental component that plays a vital role in the seamless operation of the IOT Pet Tracker.

4.2.6 Battery Power Supply

Figure 4.2.6.1: Battery

The collar subsystem features a battery power system using an 11.1V 2500mAh lithium polymer (LiPo) battery. To regulate voltage, it integrates the TI LMR33640ADDAR step-down converter, stepping the battery voltage down to 3.3V. Power efficiency is key for this system, the power supply handles both full load conditions (1.8A) and sleep mode conditions (0.2A) during periods of inactivity. This low-power mode ensures the collar can operate for extended periods before charging is necessary. The compact power supply optimizes power delivery, extends battery life, and maintains stable voltage levels for consistent performance across subsystem components.

4.3 Subsystem Validation

4.3.1 Collar State Machine Software

Verified the functionality of the collar state machine software, ensuring it transitions seamlessly between the different states (e.g., normal operation (safe), temporary disconnection (trouble), emergency alert (danger), and low-power mode). Testing scenarios included manually triggering MQTT state transitions, and simulating safe zone entrance and exit, and validating software's responsiveness and feature activation.

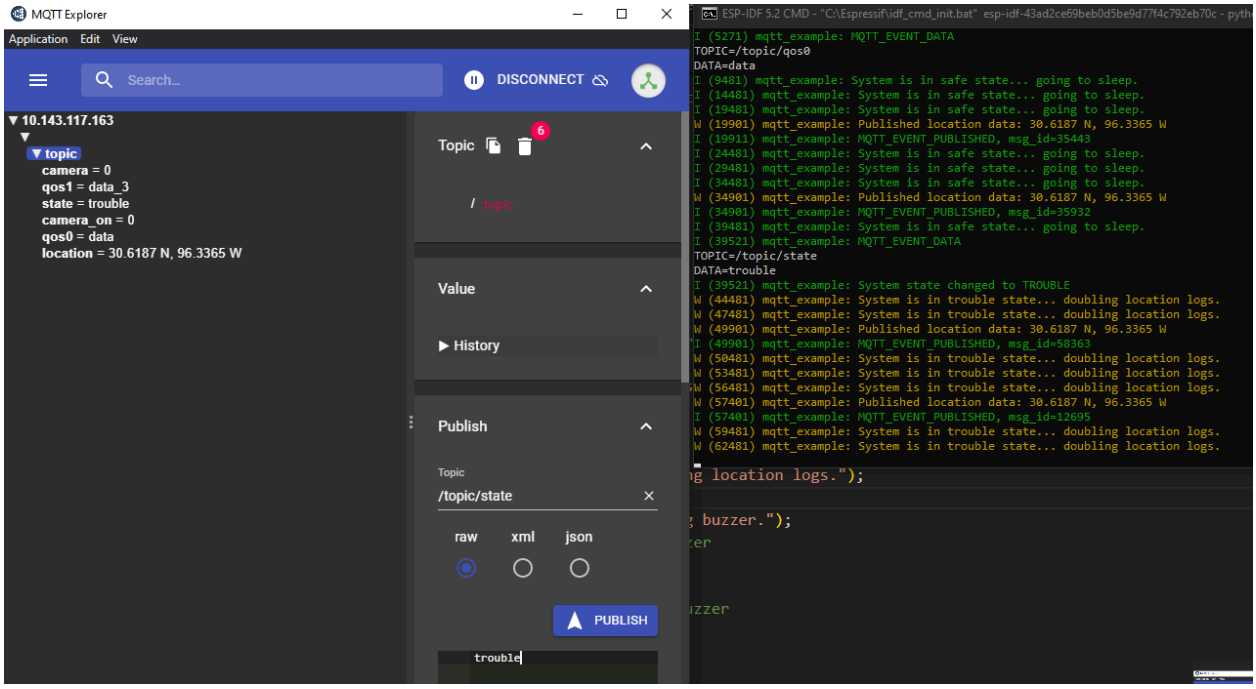


Figure 4.3.1.1: Collar State Machine Testing Output

4.3.2 Inter-Module Communication

Verified seamless communication and data exchange between hardware modules within the collar subsystem, including microcontroller (MCU), Camera, and Cellular/GPS module. Testing scenarios included continuity checks, signal tracing, monitoring live program logs for handshake and error handling messages, data integrity verification, and voltage measurements at critical points. This method ensures that the PCB design meets expected performance standards and aligns with the system's power management requirements.

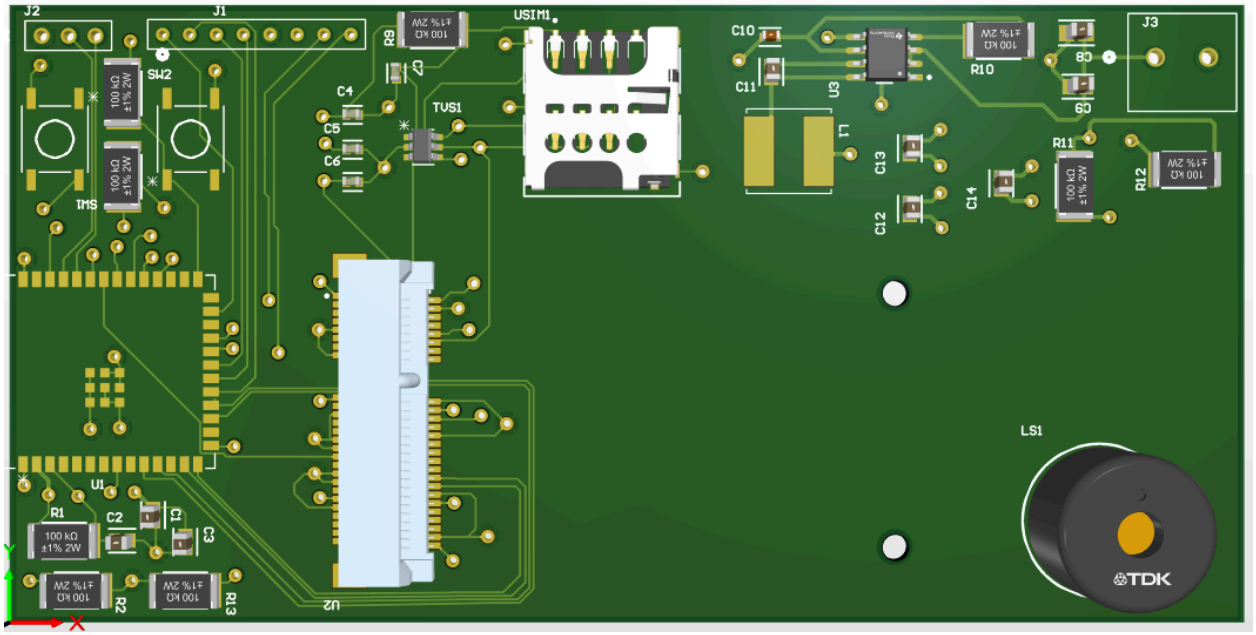


Figure 4.3.2.1: Custom PCB 3D

4.3.3 Wi-Fi and Cellular Data rate Capacity

Separately tested the data rate and reliability of Wi-Fi and cellular communication by measuring the upload and download speeds, signal strength, and packet loss to ensure consistent and high-speed data transmission. The testing process encompassed various scenarios, including the creation of detailed test cases, their implementation on the microcontroller unit (MCU), and continuous monitoring of the system's output to validate performance metrics. The collar demonstrated reliable LTE connectivity with speeds measured up to 50 Mbps upload, ensuring fast data transmission and low video latency.

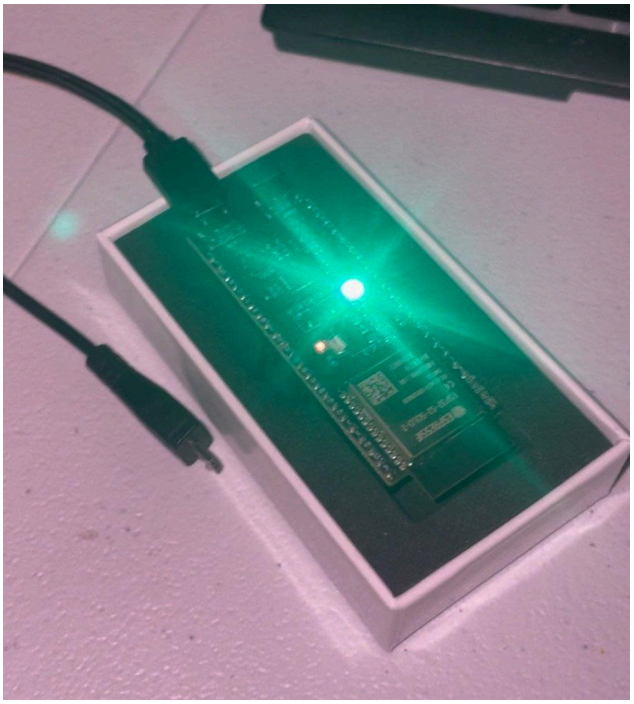


Figure 4.3.3.1: ESP32-S2 Devkit In Use

4.3.4 MQTT Data rate and Latency

Evaluated the MQTT protocol data rate and latency to assess the communication efficiency between the collar subsystem and the receiver. The testing process measured data transmission speed, latency in message delivery, and overall MQTT protocol performance while varying the wireless signal strength. These results show an example latency of up to 3ms for a large transfer.

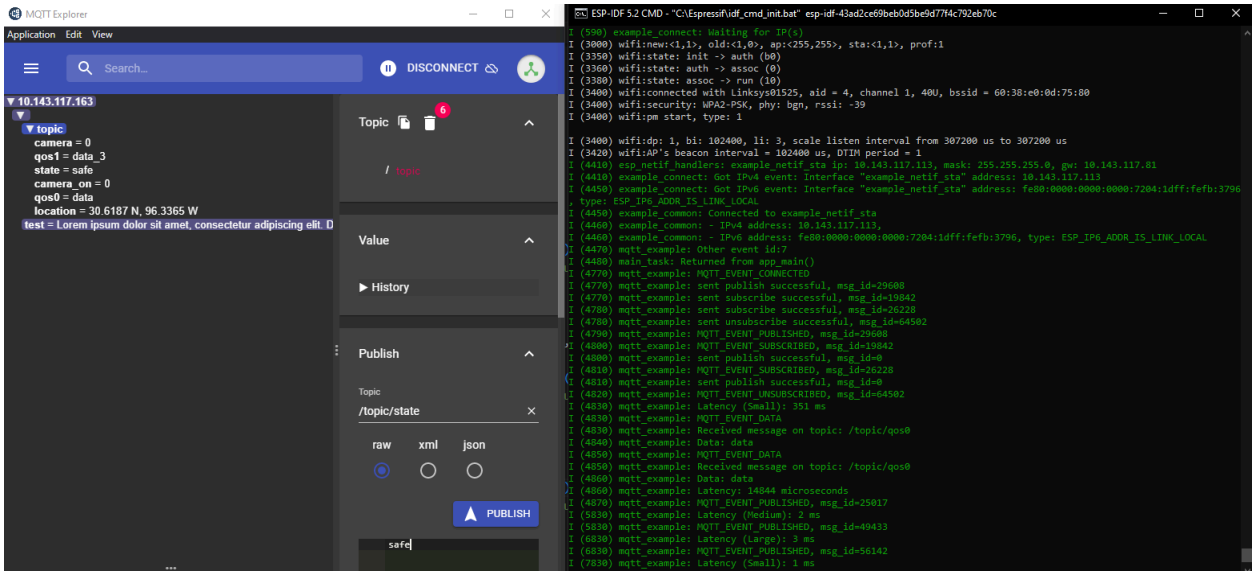


Figure 4.3.6.1: Power System Schematic

4.3.5 Power System Validation

Conducted thorough testing on the power system to ensure it meets expected performance standards, especially focusing on voltage requirements for the ESP32-S2 and Quectel EC25 modules. This includes comprehensive validation of battery life across various usage scenarios to ensure continuous operation without frequent recharging. Additionally, power management features such as low-power sleep modes and voltage monitoring were manually tested for efficiency and reliability. The testing process involved taking multimeter readings at different points on the PCB, allowing for precise measurement and assessment of power consumption and management mechanisms to ensure that voltage levels remain within the specified ranges (~3.3V) for optimal performance of the subsystem components.

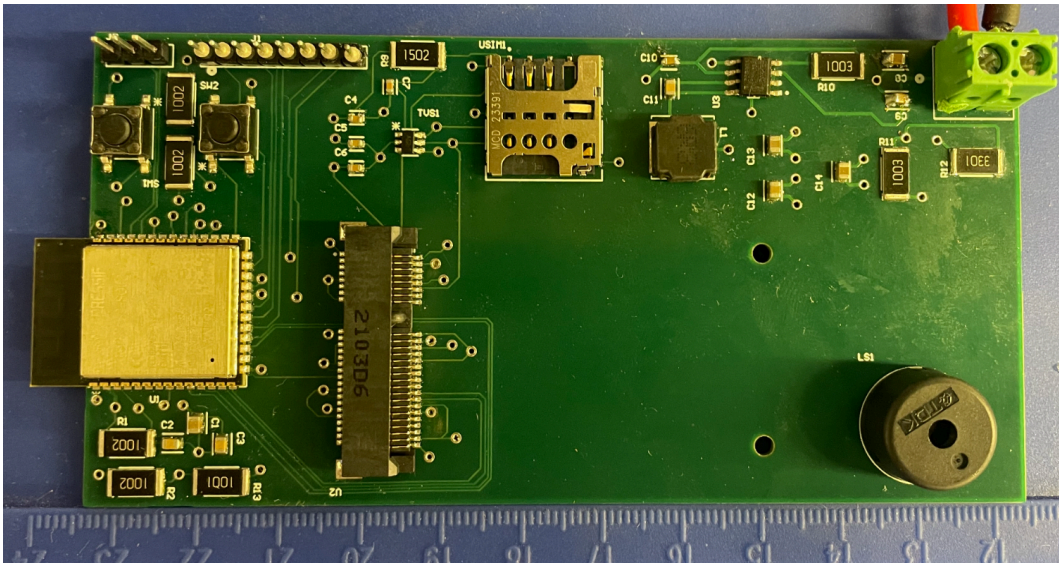


Figure 4.3.6.1: PCB Testing

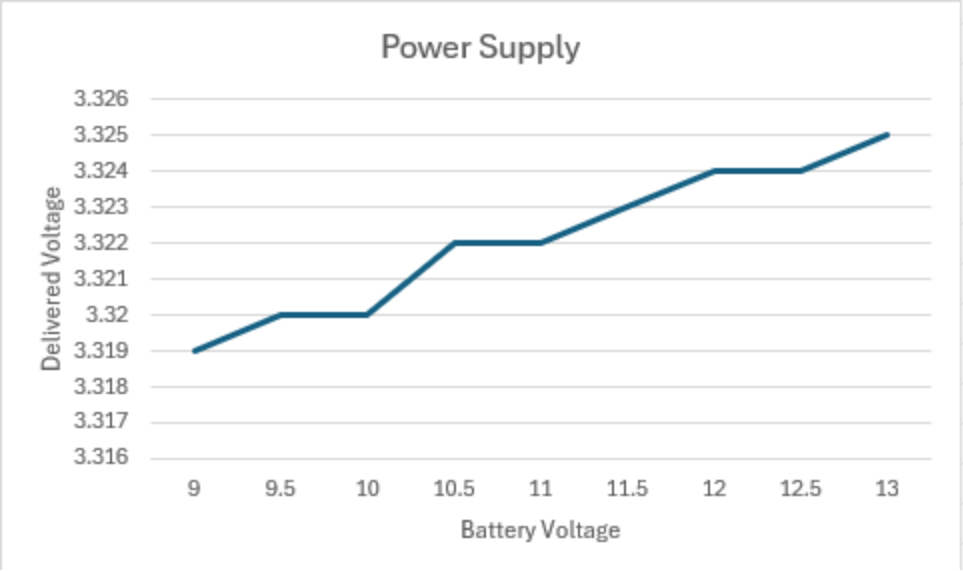


Figure 4.3.6.2: Voltage Validation at MCU

4.3.6 GPS Accuracy and Reliability

Validated the GPS module's accuracy and reliability in determining the pet's location within predefined safe zones. This validation process involved creating a simple-shaped geo-fence and physically moving the module in and out of the designated area to assess its accuracy within 3 meters. The testing was conducted across multiple environments, including indoor settings and areas with low GPS signal strength. By simulating real-world scenarios and varying environmental conditions, we were be able to ensure that the GPS module consistently delivers accurate location data, even in challenging situations.

4.3.7 Video Streaming Performance

The video streaming performance has been extensively evaluated. The ESP32-CAM provides 720p resolution video at a stable framerate, and the RTSP feed is processed by the receiver subsystem, displayed on the user interface with minimal packet loss. The latency averages at 4.5 seconds from collar to UI. This measured video quality, stability, and bandwidth consumption guarantee smooth video streaming for remote monitoring purposes without hindering other functionality.

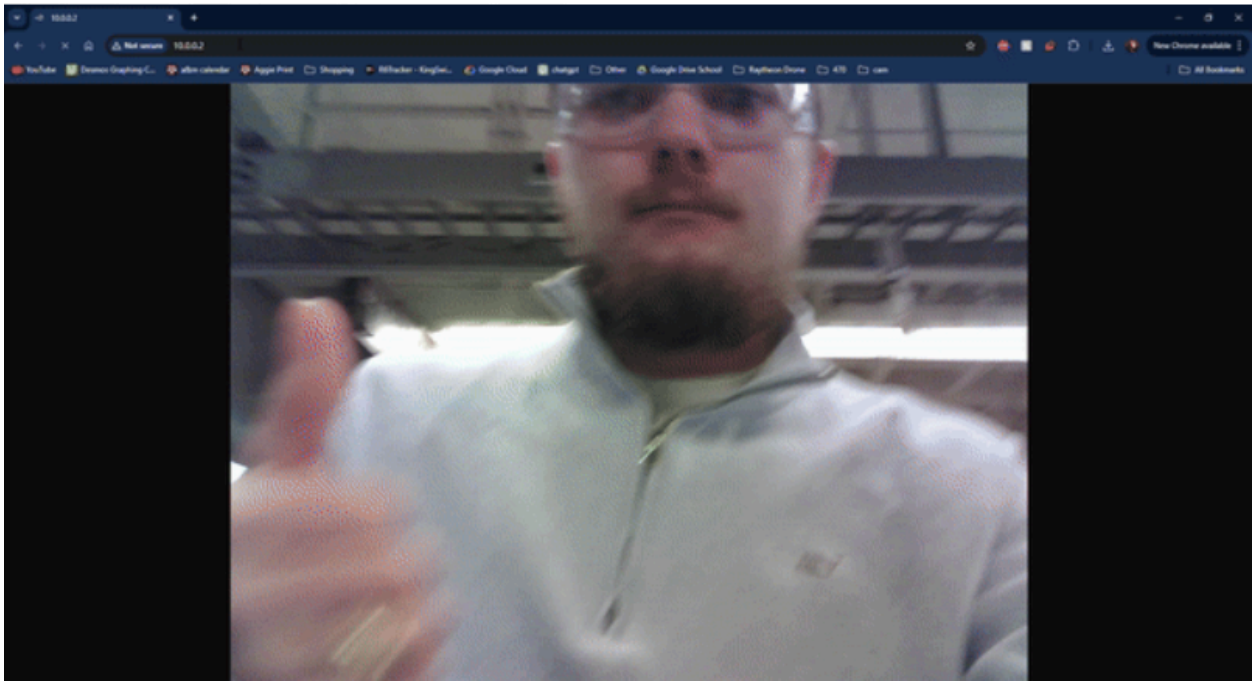


Figure 4.3.7.1: Video Stream Demo

4.4 Subsystem Conclusion

The collar subsystem represents a meticulously crafted hardware core for real-time pet monitoring within the IoT-based Pet Tracker project. This subsystem, integrated with the receiver and UI, represents a fully functional and reliable solution for pet tracking. It integrates essential hardware such as the ESP32 microcontroller, Quectel EC25 cellular module, ESP32-CAM, and buzzer, enabling seamless tracking, alerts, and security. Its power system, featuring an 11.1V 2500mAh lithium polymer battery, coupled with the TI LMR33640ADDAR step-down converter, optimizes power delivery, extends battery life, and maintains stable voltage levels. Furthermore, software functionality ensures smooth transitions between operational states, while robust inter-module communication guarantees efficient operation and data integrity. Rigorous validation processes, covering communication capacities, MQTT performance, real-time GPS accuracy, video streaming, and power system efficiency, underscore the subsystem's reliability and effectiveness, culminating in a robust and efficient pet tracking solution.

Detailed System Integration Report for the WatchDog Project

Max Kotas, Darren Lefever, Timothy Thompson

December 3, 2024

Contents

Executive Summary **3**

1 Introduction **3**
1.1 Background . . . 3
1.2 Purpose of the Report . . . 3
1.3 Scope . . . 3

2 System Overview **4**
2.1 System Architecture . . . 4
2.2 Hardware Components . . . 4
2.2.1 Testbench (Simulated Smart Collar) . . . 4
2.2.2 Receiver (Base Station) . . . 5
2.3 Software Components . . . 5
2.3.1 Backend Server . . . 5
2.3.2 Frontend Web Interface . . . 5
2.3.3 Testbench Software . . . 5

3 System Integration **6**
3.1 Communication Framework . . . 6
3.1.1 Protocols Used . . . 6
3.1.2 MQTT Communication . . . 6
3.1.3 WebSocket Communication . . . 6
3.2 Data Flow and Subsystem Communication . . . 6
3.2.1 Frontend and Receiver Integration . . . 6
3.2.2 Receiver and Testbench Integration . . . 7
3.3 Data Processing and Storage . . . 7
3.3.1 MongoDB Database . . . 7

4 Frontend Integration **10**
4.1 Features Page (features.html) . . . 10
4.1.1 Live Video Streaming . . . 10
4.1.2 Buzzer Control via MQTT . . . 10
4.2 Tracking Page (tracking.html) . . . 11
4.2.1 Real-Time GPS Tracking . . . 11

4.2.2 Geofence Creation and Management 11
4.2.3 Email Notifications 11

5 Receiver and Testbench Integration 12
5.1 Testbench Setup 12
5.2 Testbench Software Implementation 12
5.2.1 GPS Data Reading and Publishing 12
5.2.2 Buzzer Control 12
5.2.3 Video Streaming 13
5.3 Receiver's Stream Conversion and Serving 13
5.4 Integration with the Frontend 14
5.5 Overall Benefits and Impact 14
5.6 Receiver's MQTT Broker and Backend Integration 14

6 Challenges and Solutions 15
6.1 Integration Challenges 15
6.1.1 Hardware Communication Issues 15
6.1.2 Protocol Compatibility 15
6.2 Software Challenges 15
6.2.1 Real-Time Data Handling 15
6.2.2 Security Concerns 16

7 Validation and Testing 16
7.1 System Testing 16
7.1.1 Testbench Validation 16
7.1.2 Receiver and Frontend Integration 16
7.2 User Interface Testing 16
7.3 Performance Testing 16
7.4 Validation Results 17

8 Conclusion and Future Work 17
8.1 Conclusion 17
8.2 Future Enhancements 17

Acknowledgments 17

References 17

Executive Summary

The **WatchDog** project is an innovative Internet of Things (IoT) solution designed to enhance pet safety through real-time monitoring and control. The system integrates advanced hardware and software components, including a smart collar (simulated by a testbench) and a base station (receiver), to provide pet owners with features such as GPS tracking, live video streaming, geofence management, and remote control of auditory alerts via a user-friendly web interface. This report provides a comprehensive overview of the project's objectives, system architecture, integration processes, communication protocols, challenges encountered, and validation methods.

1 Introduction

1.1 Background

Pet safety is a significant concern for many pet owners, particularly when pets are left unsupervised or allowed to roam outdoors. Existing pet tracking solutions often lack comprehensive features that combine real-time location tracking, live video streaming, geofencing capabilities, and interactive user interfaces. The WatchDog project aims to address these gaps by integrating multiple subsystems into a cohesive system.

1.2 Purpose of the Report

This report details the system integration of the WatchDog project, focusing on how the various subsystems communicate and work together. It includes in-depth explanations of hardware and software components, communication protocols used, and the integration of the frontend with the receiver and the receiver with the testbench. Code snippets are provided to illustrate key integration points.

1.3 Scope

The report covers:

- Detailed system architecture and component descriptions.
- Integration of hardware and software components.
- Communication frameworks and protocols used.
- Frontend user interface design and features.
- Backend server functionalities and API endpoints.
- Testbench development and integration.
- Data flow and subsystem communication.
- Challenges faced and solutions implemented.
- Validation and testing methodologies.
- Conclusions and recommendations for future work.

2 System Overview

2.1 System Architecture

The WatchDog system comprises three main components:

1. **Smart Collar (Testbench)**: Simulated using a Raspberry Pi Zero 2 W, GTU7 GPS module, passive piezo buzzer, and an ArduCam.
2. **Receiver (Base Station)**: A Raspberry Pi acting as a server, hosting the backend application, MQTT broker, and database.
3. **Frontend Web Interface /Machine Learning**: A user-friendly web application for real-time monitoring and control.

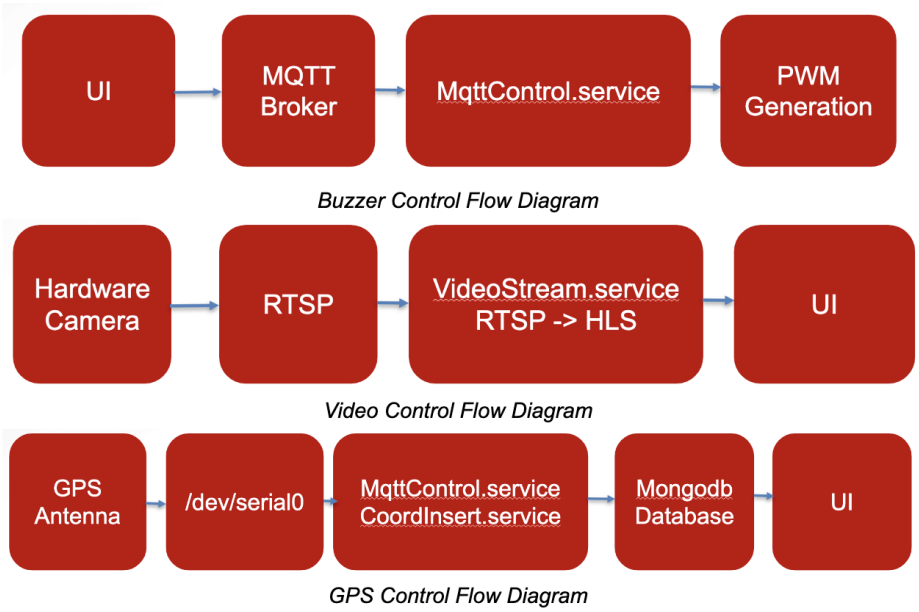


Figure 1: WatchDog System Integration And Communication Architecture

2.2 Hardware Components

2.2.1 Testbench (Simulated Smart Collar)

Due to challenges with PCB communication, a testbench was created to simulate the smart collar's functionalities. The testbench includes:

- **Raspberry Pi Zero 2 W**: Serves as the main processing unit.
- **GTU7 GPS Module**: Provides real-time GPS data.
- **Passive Piezo Buzzer**: Used for auditory alerts.
- **ArduCam**: Captures live video for streaming.

2.2.2 Receiver (Base Station)

The receiver is responsible for data processing, communication management, and hosting the user interface:

- **Raspberry Pi 5 8 GB**: Hosts the backend server, MQTT broker, and MongoDB database.
- **MongoDB Database**: Stores user data, geofences, GPS coordinates, and session information.

2.3 Software Components

2.3.1 Backend Server

Implemented using Node.js and Express.js, the backend server handles:

- **API Endpoints**: For user authentication, geofence management, and data retrieval.
- **WebSocket Communication**: Uses Socket.IO for real-time GPS data updates.
- **Session Management**: Manages user sessions with express-session and connect-mongo.
- **Database Operations**: Interacts with MongoDB for data persistence.

2.3.2 Frontend Web Interface

Developed using HTML, CSS, and JavaScript, the frontend provides:

- **Real-Time GPS Tracking**: Displays pet location on Google Maps.
- **Geofence Management**: Allows users to create, edit, and delete geofences.
- **Feature Control**: Enables activation of live video streaming and buzzer control.
- **User Authentication**: Supports user registration and login.

2.3.3 Testbench Software

The testbench runs a Python script that:

- Reads GPS data from the GTU7 module.
- Publishes GPS data to the MQTT broker.
- Listens for buzzer control commands via MQTT.
- Streams video using the ArduCam.

3 System Integration

3.1 Communication Framework

3.1.1 Protocols Used

- **MQTT**: For communication between the testbench (simulated collar) and the receiver.
- **WebSockets (Socket.IO)**: For real-time communication between the frontend and the backend server.
- **HTTP/HTTPS**: For API calls between the frontend and backend.

3.1.2 MQTT Communication

MQTT topics used:

- `collar/gps`: Publishes GPS data from the testbench to the receiver.
- `collar/buzzer`: Receives buzzer control commands from the receiver to the testbench.

3.1.3 WebSocket Communication

Socket.IO is used to:

- Emit real-time GPS updates from the backend server to the frontend clients.
- Receive acknowledgments and handle client connections/disconnections.

3.2 Data Flow and Subsystem Communication

3.2.1 Frontend and Receiver Integration

The frontend communicates with the receiver (backend server) through:

- **HTTP API Calls**: For user authentication, geofence management, and data retrieval.
- **Socket.IO**: For receiving real-time GPS updates and notifications.
- **MQTT.js (in features.html)**: For controlling the buzzer by sending messages to the MQTT broker.

User Authentication Users register and log in through API endpoints:

```
app.post('/api/register', async (req, res) => {
    // Registration logic
});
```

Listing 1: User Registration Endpoint in app.js

Geofence Management Users create and manage geofences via the frontend, which communicates with the backend:

```
app.post('/api/geofences', isAuthenticated, async (req, res) => {
    // Geofence creation logic
});
```

Listing 2: Create Geofence API Endpoint in app.js

Real-Time GPS Updates The frontend listens for real-time GPS updates using Socket.IO:

```
const socket = io();
socket.on('new-coordinate', (data) => {
    updateLocation(data);
});
```

Listing 3: Socket.IO Client in tracking.html

3.2.2 Receiver and Testbench Integration

The testbench simulates the smart collar and communicates with the receiver:

- Publishes GPS data to the MQTT broker hosted on the receiver.
- Listens for buzzer control commands via MQTT.
- Streams video to the receiver using RTSP (Real Time Streaming Protocol).

Publishing GPS Data The testbench reads GPS data and publishes it:

```
client.publish('collar/gps', json.dumps(gps_data))
```

Listing 4: Publishing GPS Data from Testbench

Receiving Buzzer Commands The testbench subscribes to buzzer commands:

```
client.subscribe('collar/buzzer')
def on_message(client, userdata, msg):
    if msg.payload.decode() == 'on':
        # Activate buzzer
```

Listing 5: Subscribing to Buzzer Commands

MQTT Broker Configuration The receiver runs an MQTT broker using Mosquitto, configured to accept connections from the testbench and frontend clients.

3.3 Data Processing and Storage

3.3.1 MongoDB Database

The MongoDB database is a cornerstone of the WatchDog system, providing a robust and scalable solution for data storage and management. It seamlessly integrates with the backend server hosted on the receiver (Raspberry Pi), enabling efficient handling of user accounts, session information, geofence data, and GPS coordinates received from the testbench. This integration is pivotal in facilitating real-time data processing and ensuring a responsive user experience.

Database Structure and Collections The database is organized into several collections, each serving a specific purpose:

- **Users Collection**: Stores user account information, including usernames, email addresses, hashed passwords, and timestamps for account creation. This collection is essential for managing authentication and authorization within the system.
- **Sessions Collection**: Maintains session data for authenticated users, enabling persistent login sessions and secure access to protected resources. Session management is handled using `express-session` and `connect-mongo`, which store session information directly in MongoDB.
- **Geofences Collection**: Contains geofence definitions created by users. Each geofence document includes the geofence name, coordinates defining the boundaries, associated user ID, city information, and timestamps. This collection allows users to manage multiple geofences and customize safe areas for their pets.
- **GPS Coordinates Collection** (`CoordsGPS`): Stores real-time GPS data received from the testbench. Each entry includes latitude, longitude, and a timestamp. This collection is crucial for tracking the pet's location and providing historical data for analysis.

Integration with the Backend Server The backend server, implemented using Node.js and Express.js, connects to the MongoDB database using the official MongoDB Node.js driver. Upon server initialization, a connection is established to the database, and the relevant collections are initialized for use throughout the application.

The integration allows the backend server to perform CRUD (Create, Read, Update, Delete) operations on the data, facilitating functionalities such as user registration, login, geofence management, and GPS data processing.

User Accounts and Authentication When a user registers, their information is stored in the Users collection. Passwords are securely hashed using `bcrypt` before storage to ensure data security. The backend server provides API endpoints for user registration and login, which interact directly with the database.

Geofence Management Users can create, edit, and delete geofences through the frontend interface. When a geofence is created, its data is sent to the receiver server, which validates and stores it in the Geofences collection. Each geofence is associated with the user who created it, ensuring that users can only access and modify their own geofences.

The geofence data includes detailed coordinate information defining the polygonal boundaries of the safe area. This data is essential for geofence breach detection and is efficiently stored in MongoDB due to its flexible document-based structure.

Real-Time GPS Data Handling The testbench publishes GPS data to the MQTT broker, which the backend server subscribes to. Upon receiving new GPS data, the server inserts it into the GPS Coordinates collection. MongoDB's efficient write performance allows for high-frequency data insertion without significant overhead.

Utilizing MongoDB Change Streams A key feature leveraged in the WatchDog system is MongoDB's Change Streams. Change Streams enable applications to subscribe to real-time updates on database operations. The backend server uses this feature to monitor the GPS Coordinates collection for new entries.

When a new GPS coordinate is inserted, the Change Stream triggers an event that the backend server listens for. The server then emits the new GPS data to all connected frontend clients via Socket.IO. This mechanism ensures that users receive real-time updates on their pet's location without the need for continuous polling.

Geofence Breach Detection The backend server processes incoming GPS data to determine if the pet has entered or exited a defined geofence. By comparing the latest GPS coordinates against the geofence boundaries stored in the database, the system can detect breaches promptly.

This processing is performed on the backend to leverage server resources and ensure consistent results. When a breach is detected, the server can trigger notifications to the user, such as sending an email or displaying an alert on the frontend interface.

Data Integrity and Security Throughout the system, data integrity is maintained through validation and error handling mechanisms. The backend server validates all incoming data before inserting it into the database. For example, when a new user registers, the server checks for existing usernames or emails to prevent duplicates.

Security is further enhanced by:

- **Input Validation**: Ensuring that all data conforms to expected formats and constraints before processing.
- **Authentication Middleware**: Protecting API endpoints by requiring authentication for sensitive operations.
- **Secure Session Handling**: Using secure cookies and appropriate flags (e.g., `httpOnly`, `secure`, `sameSite`) to protect session data.
- **Password Hashing**: Storing passwords in a hashed form using `bcrypt` to prevent unauthorized access even if the database is compromised.

Benefits of MongoDB Integration The choice of MongoDB offers several advantages for the WatchDog system:

- **Scalability**: MongoDB's horizontal scaling capabilities allow the system to handle increasing amounts of data and user load.
- **Flexibility**: The schemaless nature of MongoDB collections accommodates evolving data models, which is beneficial for handling complex geofence data and dynamic GPS entries.
- **Performance**: MongoDB provides high-performance data operations, which is crucial for real-time GPS data insertion and retrieval.
- **Real-Time Capabilities**: The Change Streams feature enables the system to react to data changes instantaneously, enhancing the responsiveness of the application.

Database Connection Initialization To illustrate the simplicity of integrating MongoDB with the backend server, the following code snippet demonstrates the database connection setup:

```
const { MongoClient } = require('mongodb');
const uri = 'mongodb://localhost:27017';
const client = new MongoClient(uri);

async function connectToMongoDB() {
  try {
    await client.connect();
    console.log('Connected successfully to MongoDB');
  } catch (error) {
    console.error('Error connecting to MongoDB:', error);
    process.exit(1);
  }
}
```

Listing 6: MongoDB Connection Initialization

Database Integration Conclusion The integration of MongoDB into the WatchDog system is integral to its functionality, enabling efficient data management and real-time processing capabilities. By leveraging MongoDB's features, the system provides a reliable and scalable platform for pet tracking, ensuring that users have timely and accurate information about their pets' whereabouts and safety.

4 Frontend Integration

4.1 Features Page (features.html)

4.1.1 Live Video Streaming

The `features.html` page allows users to view live video streams:

- Uses **HLS.js** to play HLS streams converted from RTSP feeds.
- The backend uses `ffmpeg` to convert RTSP streams to HLS segments.

```
if (Hls.isSupported()) {
    const hls = new Hls();
    hls.loadSource(streamURL);
    hls.attachMedia(videoElement);
}
```

Listing 7: Initializing HLS.js in features.html

4.1.2 Buzzer Control via MQTT

Users can control the buzzer using the `mqtt` library:

```
// MQTT Configuration
const client = mqtt.connect(MQTT_BROKER);

function toggleBuzzer(state) {
```

```
    client.publish(MQTT_TOPIC_BUZZER, state);
}
```

Listing 8: MQTT Buzzer Control in features.html

4.2 Tracking Page (tracking.html)

4.2.1 Real-Time GPS Tracking

The `tracking.html` page displays the pet's location on a Google Map:

- Uses the Google Maps JavaScript API for map rendering.
- Listens for GPS updates via Socket.IO.
- Animates the pet's marker smoothly to new positions.

```
const socket = io();
socket.on('new-coordinate', (data) => {
    updateLocation(data);
});
```

Listing 9: Socket.IO Client in tracking.html

4.2.2 Geofence Creation and Management

Users can create, edit, and delete geofences:

- Uses the Google Maps Drawing Library for polygon drawing.
- Saves geofences to the backend via API calls.

```
// Event listener for polygon completion
google.maps.event.addListener(drawingManager, 'polygoncomplete',
    function(polygon) {
    // Open modal to save geofence
});
```

Listing 10: Creating a Geofence in tracking.html

4.2.3 Email Notifications

When the pet enters or exits a geofence, an email notification is sent:

- Uses **EmailJS** for sending emails from the frontend.
- Triggered when geofence status changes are detected.

```
function sendMail(status) {
    emailjs.send(serviceID, templateID, templateParams);
}
```

Listing 11: Sending Email Notifications in tracking.html

5 Receiver and Testbench Integration

5.1 Testbench Setup

The testbench simulates the smart collar's functionalities:

- **GPS Module (GTU7)**: Provides GPS data via UART.
- **Buzzer**: Controlled via GPIO pins.
- **Camera (ArduCam)**: Streams video via RTSP.

5.2 Testbench Software Implementation

5.2.1 GPS Data Reading and Publishing

The testbench reads GPS data and publishes it to the MQTT broker:

```
import serial
import pynmea2
import paho.mqtt.client as mqtt

# MQTT Setup
client = mqtt.Client()
client.connect(MQTT_BROKER, MQTT_PORT, 60)

# Read GPS Data
def read_gps():
    with serial.Serial('/dev/serial0', 9600) as ser:
        line = ser.readline().decode('ascii', errors='replace')
        if line.startswith('$GPRMC'):
            msg = pynmea2.parse(line)
            gps_data = {
                "latitude": msg.latitude,
                "longitude": msg.longitude,
                "timestamp": msg.timestamp.isoformat()
            }
            client.publish('collar/gps', json.dumps(gps_data))
```

Listing 12: Testbench GPS Data Publishing

5.2.2 Buzzer Control

The testbench listens for buzzer commands via MQTT:

```
def on_message(client, userdata, msg):
    if msg.topic == 'collar/buzzer':
        if msg.payload.decode() == 'on':
            # Activate buzzer
        else:
            # Deactivate buzzer

client.subscribe('collar/buzzer')
client.on_message = on_message
```

Listing 13: Testbench Buzzer Control

5.2.3 Video Streaming

The video streaming component is a critical feature that allows users to receive live video feeds from the pet's perspective. This functionality is achieved through the integration of the Real-Time Streaming Protocol (RTSP), FFmpeg for stream conversion, and HLS.js for frontend playback.

RTSP Streaming from the Testbench The testbench utilizes an ArduCam connected to the Raspberry Pi Zero 2 W to capture live video footage. An RTSP server is set up on the Raspberry Pi using `v4l2rtspserver`, which streams the video over the network. RTSP is chosen for its efficiency in real-time streaming and its support for low-latency video transmission.

Implementing the RTSP Server The RTSP server captures video input from the ArduCam and broadcasts it using the RTSP protocol. Configuration settings optimize the balance between video quality and network bandwidth, ensuring smooth streaming even over limited network resources.

Challenges in RTSP Streaming While RTSP is effective for streaming, most web browsers do not support RTSP natively. This limitation necessitates converting the RTSP stream into a format compatible with web browsers for frontend playback.

5.3 Receiver's Stream Conversion and Serving

Converting RTSP to HLS on the Receiver The receiver (Raspberry Pi acting as the base station) receives the RTSP stream from the testbench and uses FFmpeg to convert it into HLS format. FFmpeg reads the incoming RTSP stream, segments it into small chunks, and generates an `.m3u8` playlist file for HLS streaming.

Using FFmpeg for Stream Conversion The FFmpeg command used for conversion:

```
ffmpeg -i rtsp://<testbench_ip>:8554/ -c:v copy -f hls \
       -hls_time 2 -hls_list_size 5 -hls_flags delete_segments \
       /path/to/hls/stream.m3u8
```

Listing 14: FFmpeg Command for RTSP to HLS Conversion

This command:

- Specifies the RTSP input stream from the testbench.
- Copies the video codec directly to reduce processing overhead.
- Outputs the stream in HLS format suitable for HTTP delivery.
- Segments the stream into 2-second intervals, facilitating adaptive streaming.

Serving the HLS Stream The HLS output files (playlist and segments) are served over HTTP by the backend server. The server ensures that only authenticated users can access the stream, enhancing security.

Handling Multiple Clients HLS is inherently scalable, allowing multiple clients to access the stream simultaneously. The server can handle multiple requests without significant additional overhead, making it suitable for systems where multiple users may need to view the stream concurrently.

5.4 Integration with the Frontend

Frontend Playback with HLS.js As detailed earlier, HLS.js is used on the frontend to play the HLS stream. The integration ensures cross-browser compatibility and provides a consistent user experience.

Optimizing User Experience To enhance the streaming experience:

- **Adaptive Bitrate Streaming**: Although not implemented in the initial version, HLS supports adaptive bitrate, which can be leveraged to adjust video quality based on network conditions.
- **Buffer Management**: HLS.js settings are configured to optimize buffer size, reducing latency and minimizing buffering interruptions.
- **Error Handling**: The frontend includes error handling to manage stream interruptions gracefully, providing feedback to the user.

Security and Access Control Access to the video stream is controlled through session management and authentication. Users must be logged in to access the `features.html` page and, by extension, the video stream. This measure prevents unauthorized access and protects user privacy.

5.5 Overall Benefits and Impact

Integrating RTSP streaming and converting it to HLS enhances the WatchDog system by providing real-time visual monitoring capabilities. This feature:

- Increases user engagement by allowing them to see their pet's environment.
- Adds value to the system, differentiating it from basic GPS trackers.
- Enables future enhancements, such as integrating machine learning for object detection or behavior analysis using the video feed.

5.6 Receiver's MQTT Broker and Backend Integration

The receiver:

- Hosts the MQTT broker to facilitate communication.
- Subscribes to MQTT topics to receive GPS data.
- Inserts received GPS data into the MongoDB database.
- Watches the database for changes and emits updates via Socket.IO.

```
async function watchCollection() {
    const changeStream = coordsGPSCollection.watch();
    changeStream.on('change', (change) => {
        if (change.operationType === 'insert') {
            const newCoordinate = change.fullDocument;
            io.emit('new-coordinate', {
                latitude: newCoordinate.latitude,
                longitude: newCoordinate.longitude,
                timestamp: newCoordinate.timestamp,
            });
        }
    });
}
```

Listing 15: Watching MongoDB Collection in app.js

6 Challenges and Solutions

6.1 Integration Challenges

6.1.1 Hardware Communication Issues

Challenge: Initial attempts to use a PCB-based smart collar faced communication inaccuracies between the receiver and the collar.

Solution: Developed a testbench using readily available hardware components to simulate the collar's functionalities. This allowed for effective testing and debugging of the system without hardware constraints.

6.1.2 Protocol Compatibility

Challenge: Ensuring seamless communication between different protocols (MQTT, Socket.IO, HTTP).

Solution: Clearly defined the responsibilities of each protocol and established a communication framework where:

- MQTT handles device-to-server communication (testbench to receiver).
- Socket.IO handles server-to-client real-time updates (receiver to frontend).
- HTTP handles client-to-server requests (frontend to receiver).

6.2 Software Challenges

6.2.1 Real-Time Data Handling

Challenge: Achieving low-latency updates for GPS data and video streams.

Solution: Utilized efficient libraries and optimized code:

- Used Socket.IO for real-time communication with minimal overhead.
- Employed HLS.js for video playback with adjusted buffer settings.
- Optimized database queries and change streams.

6.2.2 Security Concerns

Challenge: Protecting user data and securing communication channels.
Solution:

- Implemented user authentication with password hashing (bcrypt).
- Used express-session with secure cookies and session storage in MongoDB.
- Employed middleware to protect API endpoints.

7 Validation and Testing

7.1 System Testing

7.1.1 Testbench Validation

The testbench was used to validate:

- Accurate GPS data transmission via MQTT.
- Correct buzzer activation in response to MQTT commands.
- Reliable video streaming from the ArduCam.

7.1.2 Receiver and Frontend Integration

Tests were conducted to ensure:

- Real-time GPS data updates are correctly displayed on the frontend map.
- Geofence breaches trigger appropriate notifications and actions.
- User interface elements function as intended (e.g., geofence creation, feature activation).

7.2 User Interface Testing

- Verified cross-browser compatibility and responsiveness.
- Ensured smooth animations and transitions.
- Tested usability and accessibility features.

7.3 Performance Testing

- Monitored system performance under various load conditions.
- Evaluated latency in data updates and video streaming.

7.4 Validation Results

The system demonstrated reliable performance with successful integration of all components. Key functionalities were validated through tests, and the system is ready for further enhancements and deployment.

8 Conclusion and Future Work

8.1 Conclusion

The WatchDog project successfully integrates hardware and software components to provide a comprehensive pet safety solution. By focusing on the integration of subsystems and utilizing efficient communication protocols, the project addresses the challenges of real-time monitoring and control. The testbench proved instrumental in validating the system's functionalities in the absence of the final hardware.

8.2 Future Enhancements

- **Hardware Development**: Finalize the PCB-based smart collar for deployment.
- **Scalability**: Extend support for multiple devices and users.
- **Security Enhancements**: Implement encryption for MQTT and WebSocket communications.
- **Mobile Application**: Develop native mobile apps for improved user experience.
- **Machine Learning Integration**: Use predictive analytics for pet behavior analysis.

Acknowledgments

We extend our gratitude to our advisors and peers for their support and guidance throughout this project.

References

[1] MQTT Protocol Documentation: `https://mqtt.org/documentation`

[2] Socket.IO Documentation: `https://socket.io/docs/v4`

[3] MongoDB Change Streams: `https://docs.mongodb.com/manual/changeStreams/`

[4] HLS.js Library: `https://github.com/video-dev/hls.js`

[5] Google Maps JavaScript API: `https://developers.google.com/maps/documentation/javascript`

[6] EmailJS Documentation: `https://www.emailjs.com/docs/`

[7] Express.js Documentation: `https://expressjs.com/`

[8] Node.js Documentation: https://nodejs.org/en/docs/

[9] Paho MQTT Python Client: https://www.eclipse.org/paho/index.php?page=clients/python/index.php