R.O.A.M.

Robotic Observer of Abandoned Materials
Group 14

Daniel Claassen
CpE
Computer Vision Hardware & Software

Claire Persinger
EE
PCB & Hardware

Nathan Little
CpE
Robot Guidance & Web Development

Ethan Robotham
CpE
Systems Integration & Communication

Overview and motivation

R.O.A.M. - Robotic Observer of Abandoned Material

- Autonomous robot to detect and report abandoned objects
- Aim to keep useful items out of landfills and clean up neighborhoods
- Created an interactive website for people to participate in the collection of it items and trash

Goals

Basic / Advanced

- Identify treasure and trash items.
- Autonomous sidewalk following.
- Upload item location and classification to the website.
- Request assistance through human intervention.

Stretch

- Further classification to identify treasure type.
- Integrate search feature via location or treasure type.
- Upload photos of identified artifacts.
- Solar battery charging system.
- Battery life indicator.

Engineering Specifications

Parameters	Specifications
Speed (legal requirement)	< 10 mph
Width (wheel to wheel)	< 46 cm
Traversable Obstacle Height	< 35 mm
Website Data Upload Time	< 10 seconds
Object Detection Response Time	< 2 seconds
Treasure Identification Accuracy	> 70%
Battery Life	> 15 minutes
GPS accuracy	Within 20 m
Distance traveled without going out of bounds	> 100 meters
Object Detection Range (R)	0.5m < R < 3m
Power consumption	< 55 W
Cost	< $1500

Core Attributes to demo

- Treasure Identification Accuracy
- Autonomous driving without going out of bounds
- Object, trash or treasure, detection range

Hardware Part Selection

Motor Technology and Part Selection

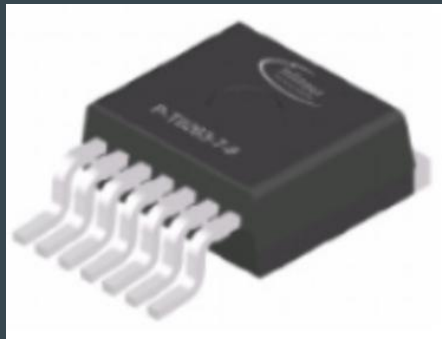
Motor	Pros	Cons
25D 12V 117 RPM Spur Gearmotor	• Compact and lightweight • Inexpensive • Easy to mount in small frames	• No encoder for feedback • Spur gears wear faster under load • Lower long-term durability • Less torque than planetary types
36mm 12V 170 RPM Planetary Gearmotor w/ Encoder	• Good balance of torque and speed • Planetary gearbox • Built-in encoder	• Generic • Encoder resolution is low • Shaft may require adapter for some hubs • Limited official documentation
goBILDA 5202 Yellow Jacket 19.2:1	• High-quality, branded motor • Built-in encoder • Strong planetary gearbox • Excellent documentation and support	• More expensive • 8mm REX shaft may not be compatible with non-goBILDA • Slightly larger and heavier • Possibly overbuilt for early prototypes

- Brushed DC 12V Motor
  - PWM control, high torque options
  - Acceptable stall current
  - Ease of integration
  - Excellent documentation

Motor Driver Part Selection

Technology	Supply (V)	Logic In (V)	Continuous Current (A)	Voltage Drop per Leg	Integration Effort
L298N Breakout	5–35	5	2.0	~2.0 V	Plug-and-play, not 3.3 V compatible
DRV8833 Breakout	2.7–10.8	1.8–6	1.5	~0.3 V	Low effort, but supply is marginal
TB6612FNG Breakout	2.5–13.5	1.8–5	1.2	~0.2 V	Ideal for 3.3 V GPIO, thermal limits apply
IRF3205 Dual-MOSFET Board	3–36	Requires ≥10 V	10	~0.05 V	No PCB needed, but not logic-compatible
IR2104 MOSFETs (Custom PCB)	5–20	3.3–20	≥1.5	~0.025 V	Medium effort, full layout control
Bts7960	6-27	3.3-5	2	~.2	Easy layout to incorporate into this design

- BTS7960
  - Can handle up to to 43A
  - Overcurrent protection
  - Undervoltage lockdown
  - Bidirectional use
  - PWM speed control
  - Smooth logic integration
  - Information widely accessible



Noise Emission Part Selection

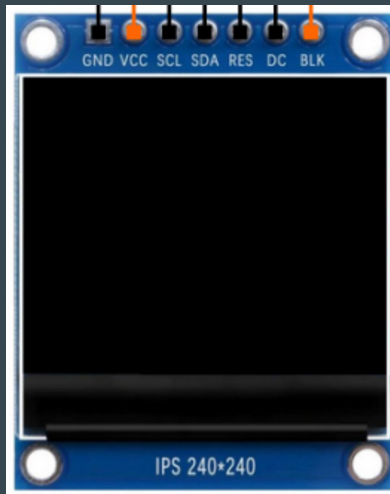
	Pro	Con
Audio Player PCM5102	• Versatile options • Explicit spoken instructions	• Higher energy demand • More system integration • Additional hardware needed
Magnetic Buzzer WST-1206UX	• Easy to incorporate • Cost efficient • Simple design	• Limited tone • Limited volume • Not suitable for outdoors
Piezo Buzzer PS1240	• Sufficient sound • Adjustable volume • Cost efficient • Easily incorporated • Suitable for outdoors	• No versatile noise

- Piezo Buzzer PS1240
  - Adjustable Volume
  - Cost Efficient
  - Low complexity

LCD Part Selection

	Pros	Cons
ILI9341 (2.8"–3.2")	• Touchscreen support (in some versions) • Higher resolution (320×240) • Compatible with ESP32 and TFT_eSPI • Larger display area for data and control	• Larger physical footprint • Higher power usage • Touch requires calibration and extra wiring • Needs protection from outdoor conditions
ST7735 (1.8")	• Very compact and lightweight • Low power consumption • Simple wiring with SPI • Supported by common graphics libraries	• No touchscreen support • Low resolution (160×128) • Small display limits layout complexity • Poor readability from distance
ST7789 (2.0")	• Higher resolution • Square layout offers flexible design • Good clarity for its size • SPI library support • Efficient design	• No touch functionality • Some modules lack reset or level shifting • Limited space for elements

- ST7789
- Cost efficient
- Small footprint
- Good clarity



Battery Part Selection

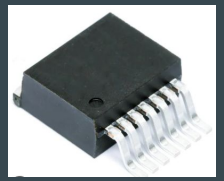
Battery	Pros	Cons
Bioenno 12V 12Ah $LiFePO_4$	• Steady voltage under load • Built-in BMS for protection • Long cycle life • Lightweight and compact • Reliable for mobile use	• More expensive • Needs $LiFePO_4$-specific charger • Heavier than smallest $LiFePO_4$ packs
TalentCell 12V 12Ah $LiFePO_4$	• Regulated 12V output • Lightweight and compact • Built-in protection • Plug-and-play wiring • Good cycle life	• Limited current output • Not ideal for high draw setups • Requires correct $LiFePO_4$ charger
Mighty Max 12V 12Ah SLA	• High capacity • Inexpensive • Easy to find • Simple to charge with basic charger	• Heavy and bulky • Voltage drops under load • Shorter lifespan • Poor energy efficiency for mobile use

- TalentCell 12V
  - Compact
  - 2000+ Cycles
  - 153.6Wh
  - Moderate Price

Regulator Part Selection

Type	Pro	Con
MP1584	• Fast switching freq • 4V-28V	• 2A • Thermal efficiency
TPS5430	• 3A • Moderate Efficiency	• Higher voltage range
LM2576	• 3A • 1.23V-37V • Variability	• Slower switching freq
LM2679	• 5A • 1.2V-37V • Variability	• Sensitive to noise

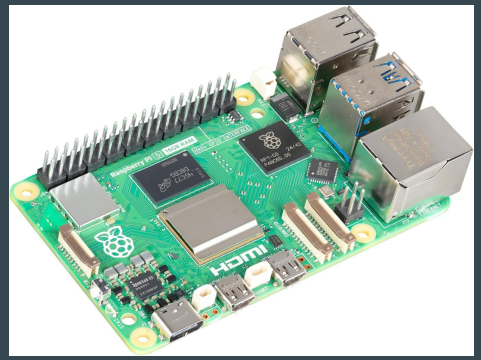
- LM2576SX-ADJ/NOPB
  - 3.4V 2A
  - Fixed 52kHz
  - Built in current limit
  - Availability
- LM2679SX-ADJ/NOPB
  - 5.2V 3A
  - 260kHz
  - Current limiting, thermal shutdown
  - Availability



Edge Device

Raspberry pi 5 was selected for its ease of use, affordability, and adequate performance

Device	Cost	Familiarity	Power Consumption	Performance
Raspberry PI 5	~ $80	Very	< 10W	Medium
NVIDIA Jetson Orin Nano	~ $250	None	< 15W	High
Arduino Portenta H7	~ $100	Little	< 2W	Low



Camera Part Selection

Ideal Camera

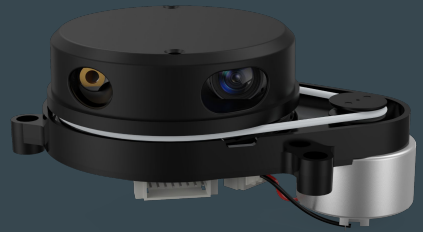
> 7 MP
~ 100° HFOV
Global Shutter
HDR Support

Device	MP	HFOV	Shutter Type	HDR Support	Cost
Raspberry PI Camera Module 3	12 MP	66° - 102°	Rolling	Yes	~ $25
Arducam OV9782	1 MP	70°	Global	No	~ $60
Intel RealSense D435	2 MP	85°	Global	No	~ $300

LIDAR

LiDAR Model	Pros	Cons	Cost
RPLIDAR A1	• Extremely affordable • Straightforward integration	• Coarse scan • Limited refresh rate	$100
YDLIDAR X4	• Higher scan frequency • Low integration effort	• Shorter range • Similar angular fidelity	$100
HLS-LFCD2	• Lowest price point • Ideal for testing hardware stacks	• Unstable data • Lacks official support	$80
Benewake TF02	• Best balance of range and detail for field trials	• Requires parser • UART wiring considerations	$220
Yahboom LD06	• High-density output • Stable DTOF performance	• Heavier • Specialized control circuitry needed	$250

- No significant scan rate difference in the 2D LIDAR market.
- ROS1 and Docker support.
- Same price as the A1 but with 2Hz higher scan rate.
- More expensive models add range, not data.



SIM Card Module

Component	Pros	Cons
SIM7600G-H	• Fast upload speeds (150 Mbps) • GNSS support • Includes antennas and cables • Official support	• Larger footprint than other boards • Slightly higher power consumption
SIM7000G	• Energy efficient • Built in GNSS	• Slow upload speeds (375 Kbps) • Limited network availability
A7670E	• Cheaper than CAT4 modules • Multiple interfaces (UART,USB)	• Slower upload rates (10 Mbps) • Can be unreliable for multiple simultaneous protocols

- Best data speeds in price range.
- Accurate and Built-in GNSS for gps position data
- Comes as a kit complete with a breakout board and antennas.
- Documentation on AT commands and detailed datasheet.

Microcontroller

Microcontroller	Pros	Cons
ESP32-S3-WROOM-1U-N16R8	• Three UART pins • 16 MB flash + 8 MB PSRAM = huge head-room • Massive Arduino/ESP-IDF ecosystem	• About 50 mA active, higher than ultra-low-power counter parts
MSP-EXP432P401R	• Three UART pins • Ultra-low-power modes	• Lower clock rate (48 MHz) • Low SRAM (64kB)
Nordic nRF52840	• Tiny current draw when in deep sleep (Micro amps) • BLE radio could be enabled later for field debug	• Only 2 UART pins

- Well supported
- Lots of space
- Cost efficient

Software & Technology Selection

Web Stack

Requirement	UI Choice	Backend Choice	Ancillary Service	Fit Rating
Fast Load Over 4G	Tailwind + Alpine.js	Flask WSGI + Gunicorn	OpenTelemetry + Prom	High
Low Memory Footprint	Tailwind (10 KB CSS)	SQLite Embedded	Redis for burst cache	Very High
Ease of Development	Utility-first CSS, minimal JS	Flask microframework	Docker CI/CD pipeline	High
Offline Resilience	Static files, ServiceWorker	SQLite local buffering	Celery offline queue	Moderate
Scalability	Alpine.js for controls	Flask async + WSGI	Kubernetes orchestration	High
Observability	Grafana dashboards	Prometheus monitoring	OT Collector pipeline	Very High

- Flask + Gunicorn — lightweight, Python-friendly, easy to deploy on a small VM.
- SQLite — embedded, low-memory database, perfect for simple data storage from the ESP32.
- Tailwind + Alpine.js — minimal CSS/JS for fast loads over 4G with reactive UI updates.
- Nginx reverse proxy — secures and optimizes traffic between API and static assets.

Map Service

Service	Pros	Cons
OpenStreetMap +Leaflet	• Free and open-source • Lightweight • Community support • Self-hosted offline caching	• Public servers can throttle • Lack of built-in styling • DIY cache and proxy setup
Google Maps API	• Best-in-class images and points of interest • Built-in traffic, geocoding and search • Mature documentation	• Expensive $7/1000 loads • No built-in offline caching • Costs may vary
MapLibre GL	• GPU-accelerated rendering • Mapbox-style compatible • No licence fees	• Must host vector tiles • Higher overhead, limited map-scaling
Mapbox	• Polished ecosystem • Global tiles with low-latency	• Usage fees after free tier • Vendor lock-in • Offline caching is paywalled

- Lightweight and optimized for quick rendering over 4G for ideal mobile and desktop performance.
- Supports caching so maps remain usable even with intermittent connectivity.
- Easy integration with the Flask backend and front-end stack, requiring minimal extra dependencies.
- Free and no rate-limiting concerns.

Communication Protocol

UART

- Simple to implement
- Low data rate but low usage
- Low CPU overhead

Technology	Pros	Cons
UART	• Simple to implement • Low CPU overhead • Two wires • Best for sending small bits of data • Built in support	• Slow for large data • More prone to • timing errors since clock is asynchronous • Limited error detection
I2C	• Supports multiple devices on same bus • Two wires • Best for low-speed sensors • Built in support	• Slow for large data • Not reliable for large data • Sensitive to noise • High chance for bus collisions
SPI	• Full-duplex and synchronous • Supports DMA • Very high data rates • Good for sending large data chunks • Reliable • Low latency	• Requires 4 or more wires • Short range (<1 m) • More complex software

Object Classification Model

YOLOv8n

- Lightweight
- Easy to implement
- Effective in Real-Time
- Highly accurate

Model	Accuracy	Speed	Model Size	Resource Demand	Ease of Use
YOLOv8	High	Real-Time	Light	Moderate	Plug- and Play
EfficientDet + EfficientNetB1	Very High	Slow	Heavy	High	Complex
SSD	Moderate	Near Real-Time	Moderate	Low	Simple

Inference Backends for YOLOv8n

ONNX Runtime

- Fast
- Simple to implement
- Effective Optimization
- Less Prone to Error than NCNN

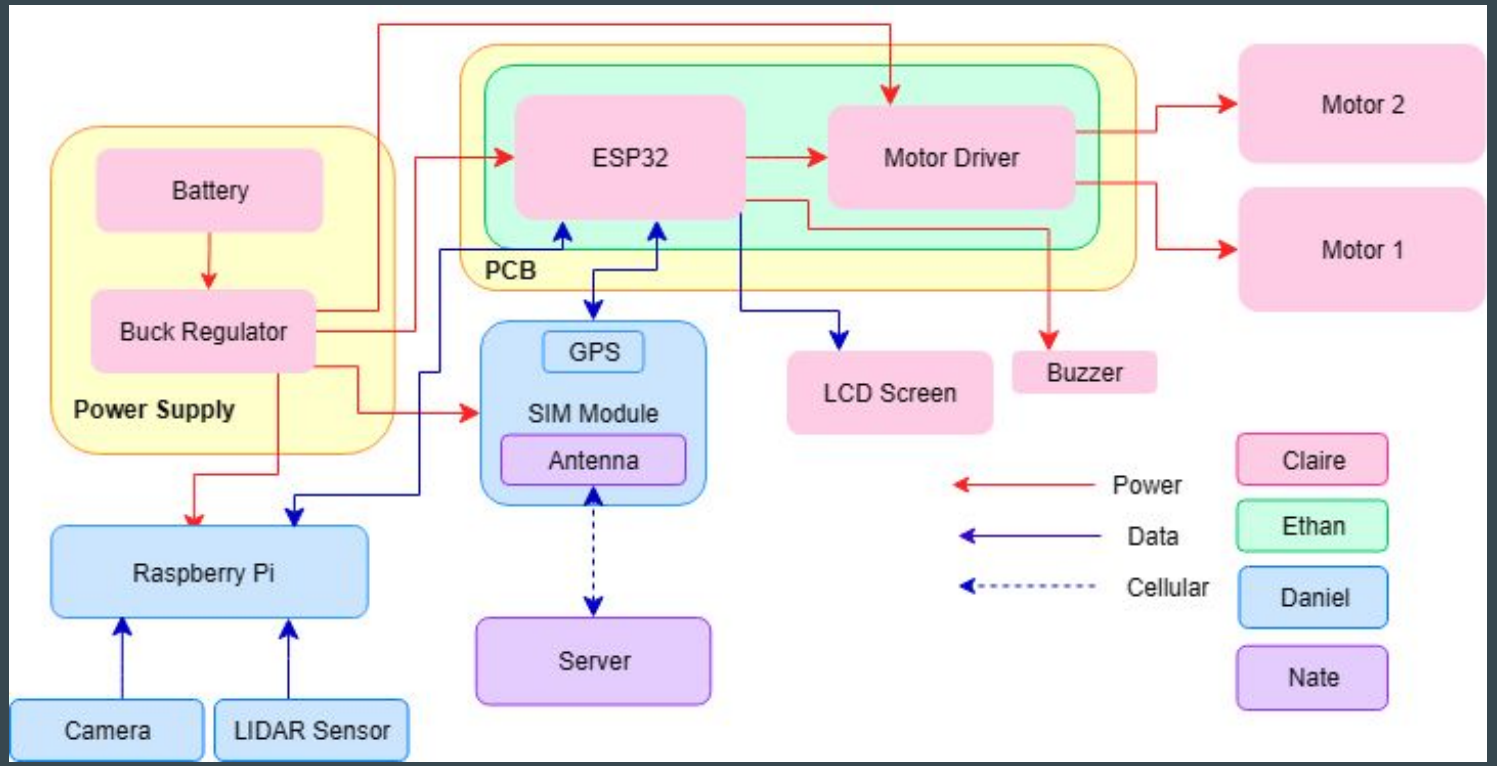
Backend	Speed	Deployment Complexity	Optimization	Resource Demand
ONNX	Fast	Moderate	Graph Optimizations, Quantization	Light
PyTorch	Slow	Minimal (Default)	Dynamic graphs	Highest
NCNN	Fastest	Highest	Lightweight Kernels	Very Light

Programming Language

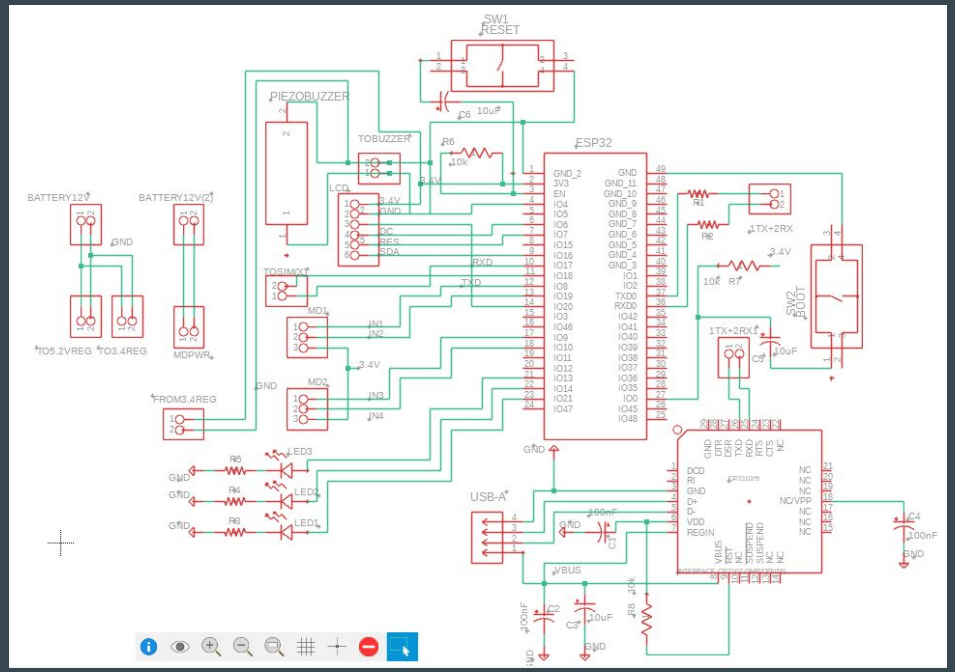
Language	Pros	Cons
C++	• Fast and efficient at runtime • Uses less power and memory • Good for real-time tasks and multithreading • Gives full control over hardware and memory • Works well with libraries like OpenCV	• Harder to learn and write • Debugging is more difficult • Memory errors can crash the system
Python	• Easy to learn and use • Great for fast development and testing • Large library support • Strong community help	• Slower performance • Uses more power and memory • Can't run true multithreading • Errors can be harder to catch during runtime

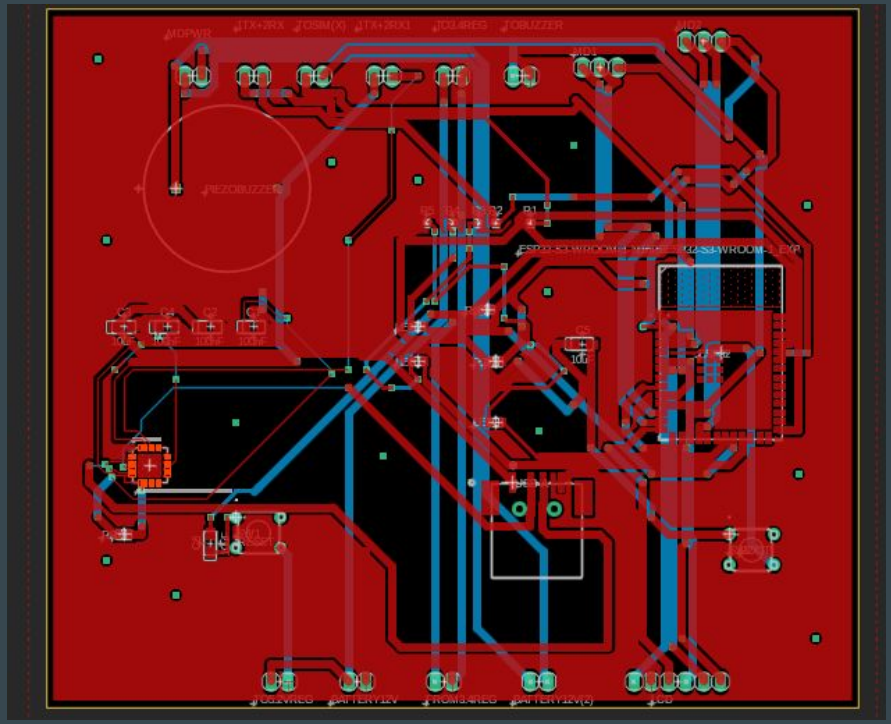
Hardware Design

Hardware Block Diagram



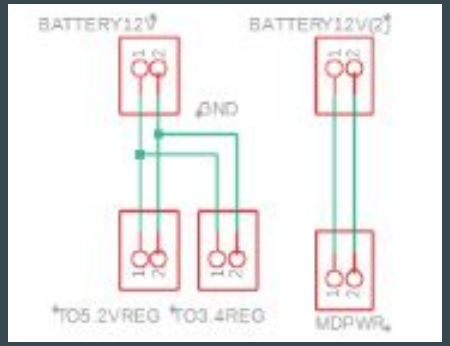
PCB Design


SW1
RESET
PIEZOBUZZER
TOBUZZER
ESP32
LCD1
TOSIMVX
MD1
MD2
BATTERY12V
BATTERY12V(2)
FROM3.4REG
TO5.2VREG
TO3.4REG
MDPWR
USB-A
LED1
LED2
LED3
SW2
BOOT
1TX+2RX
RXD
TXD
3.4V
GND
VBUS

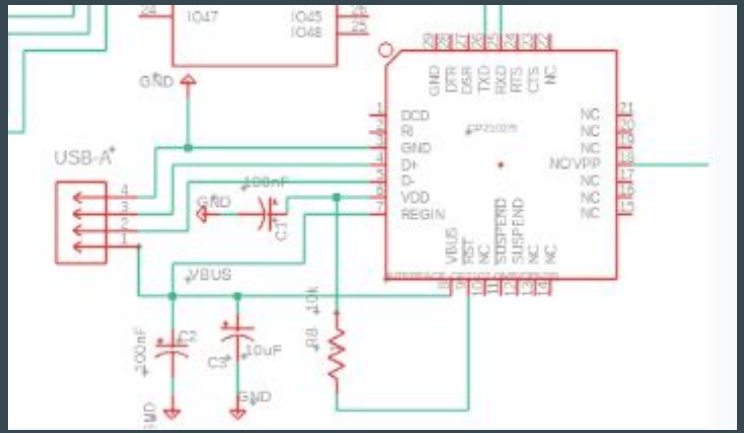

MDPWR
1TX+2RX
TOSIM(X)
COLORES
TOBUZZER
PIEZOBUZZER

PCB Design

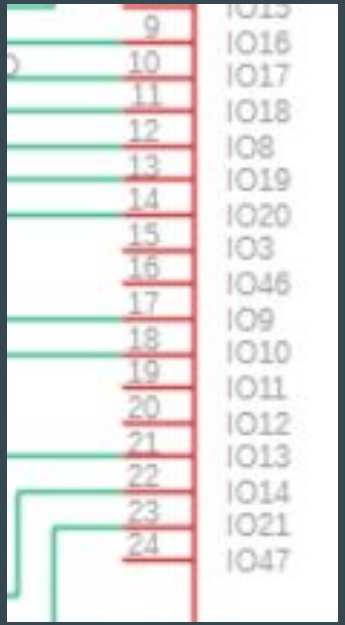
Battery to Reg



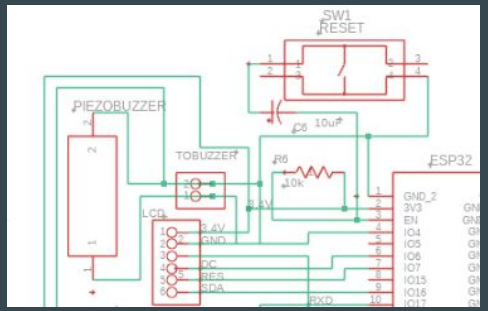
UART to USB



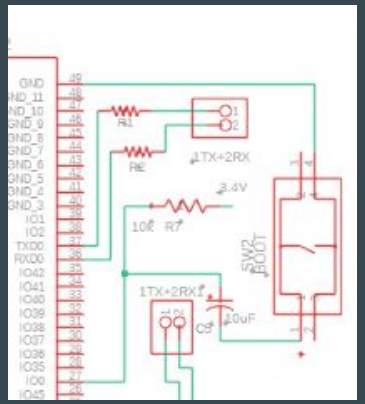
MD + LEDs IO



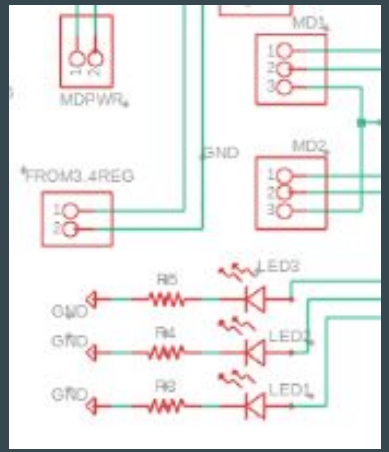
Buzzer & LCD



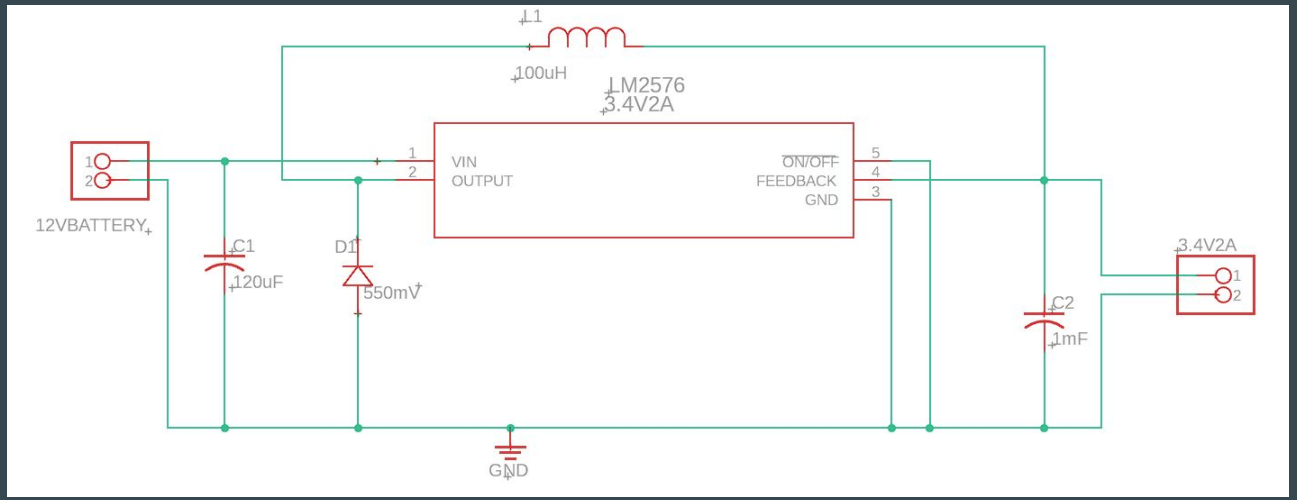
BOOT SW

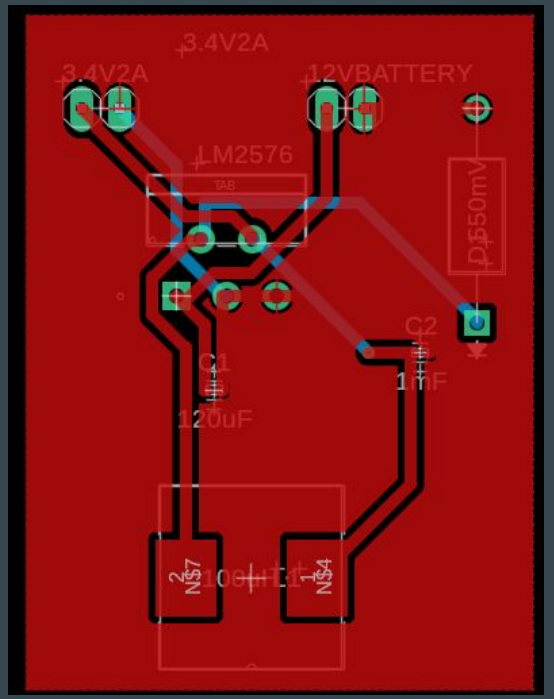


MD + LEDs

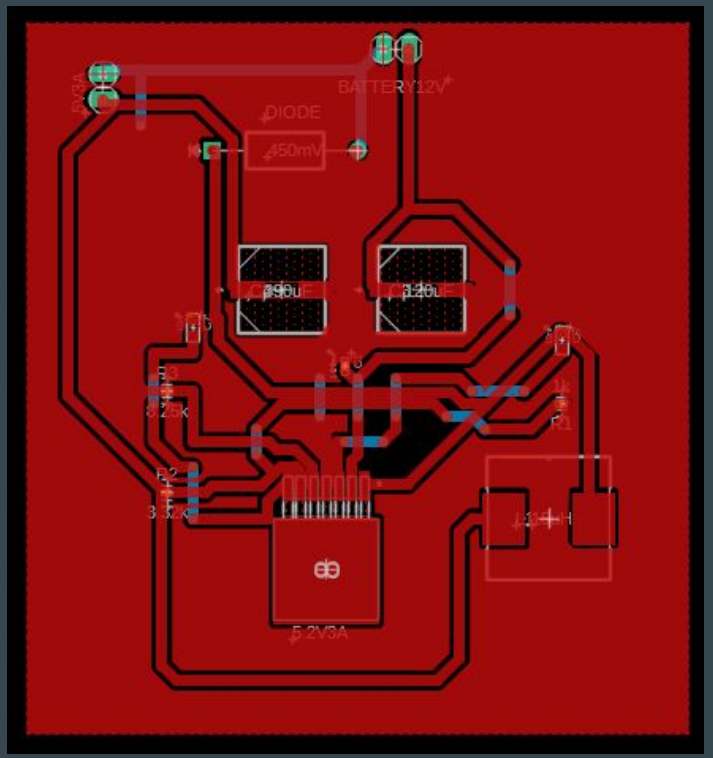


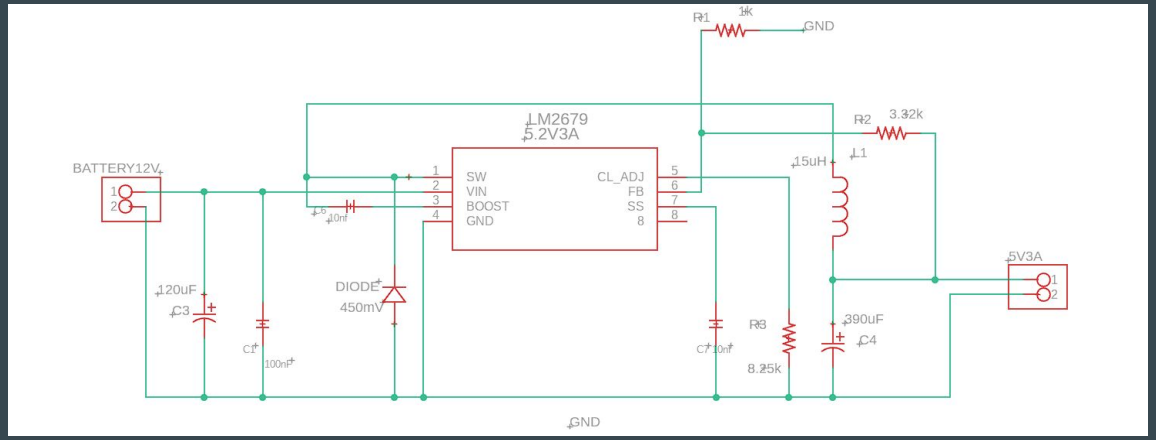
PCB Regulator Design 3.4V 2A



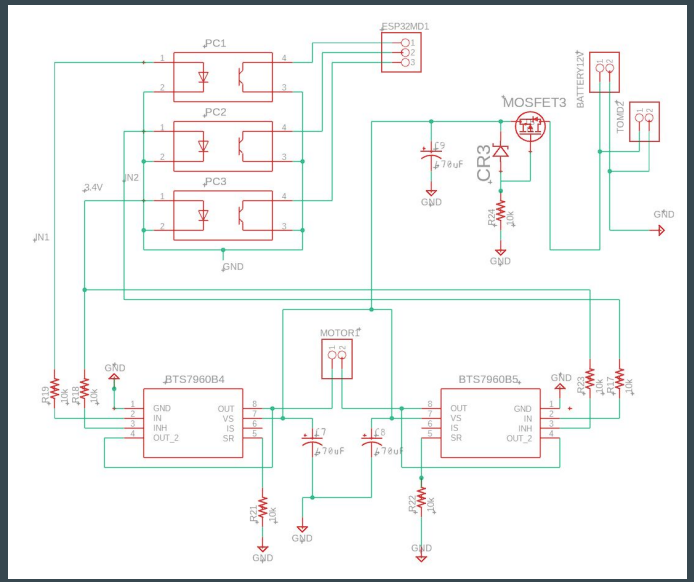


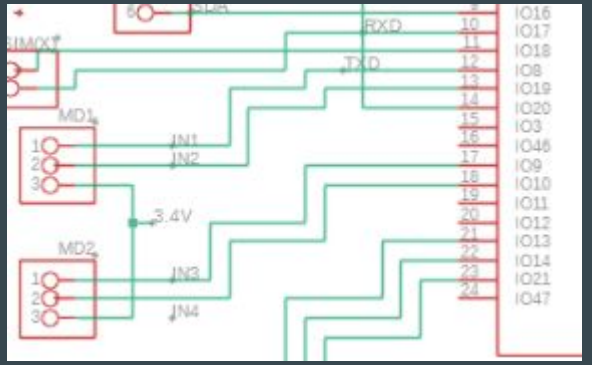
PCB Regulator Design 5.2V 3A

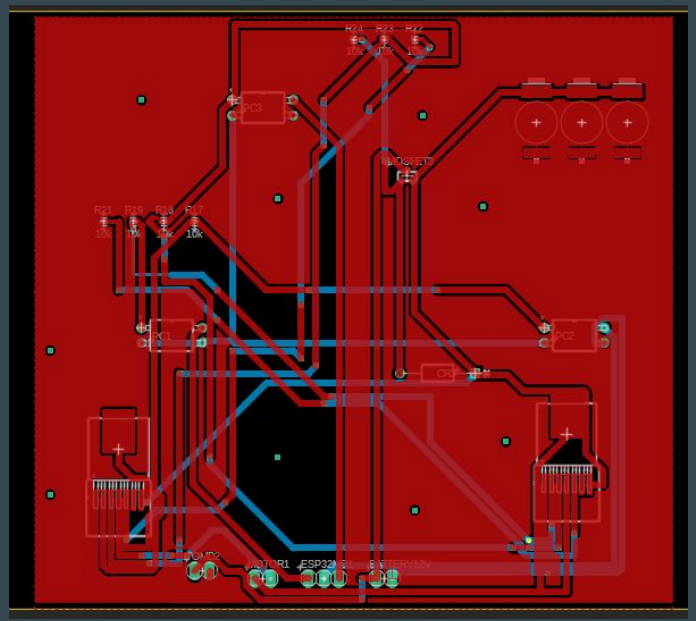




PCB Regulator Design Motor Driver



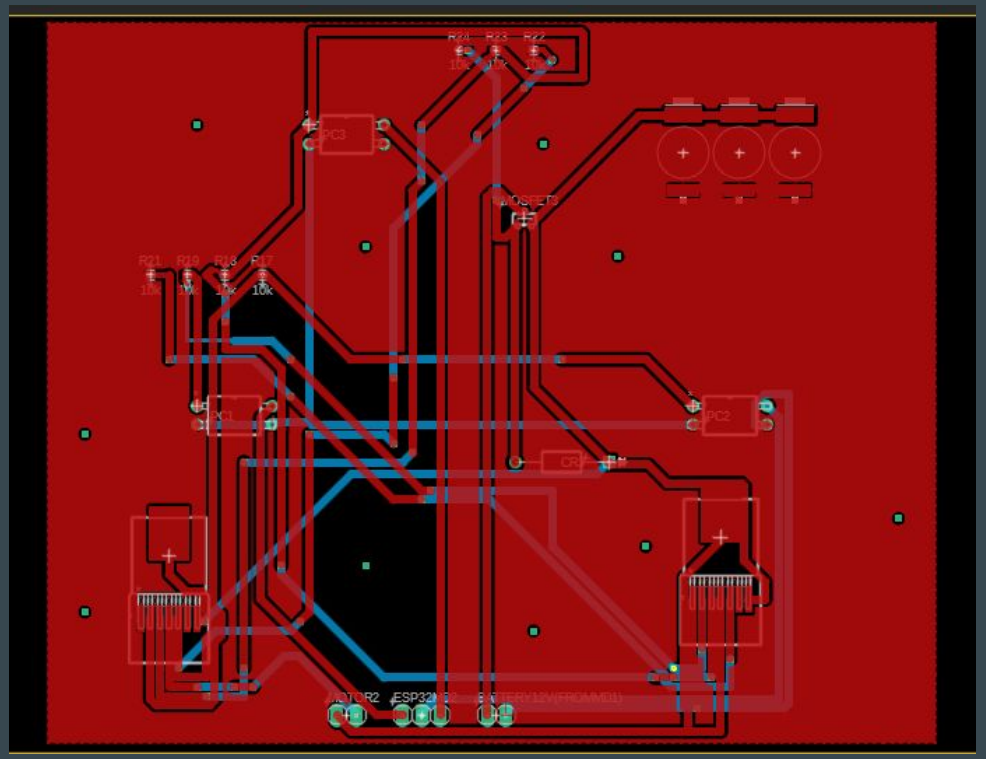


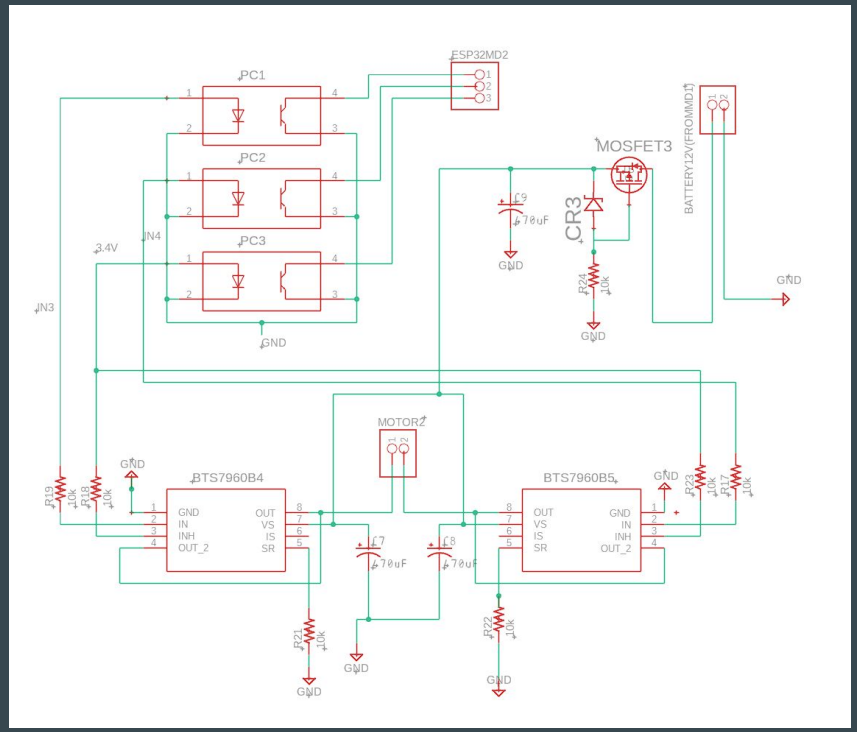


Logic
- IO8/IO19
- IO9/IO10

3.4V Logic Pwr

PCB Regulator Design Motor Driver





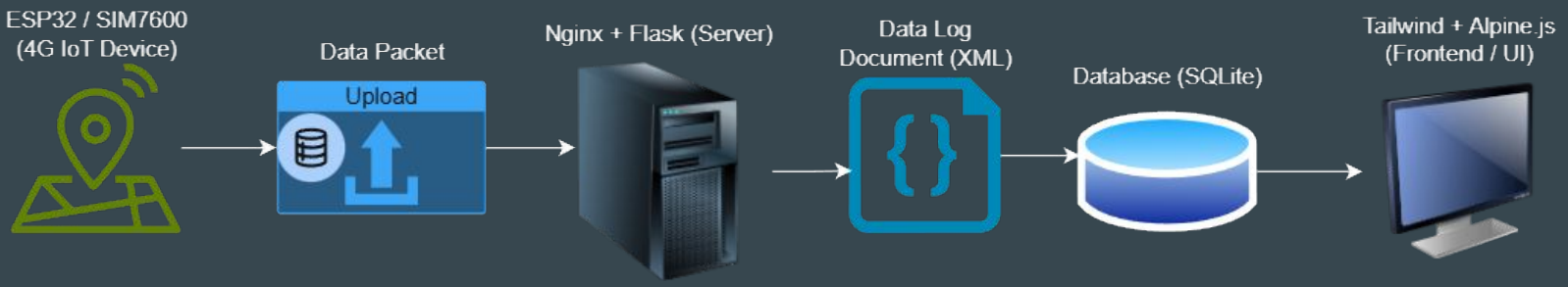
Software Design

Object Classification Pipeline

Responsible for Capturing, Packaging, and sending the image of an identified furniture and amount of garbage

Detection Report
image: JPEG
Object_type: str
conf: float

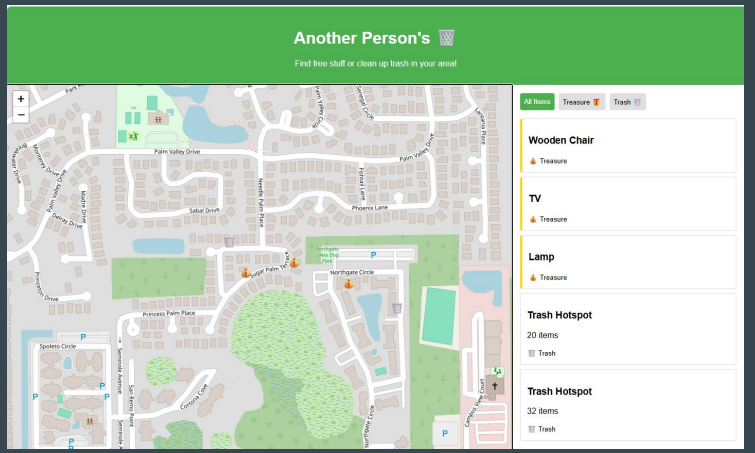
Web Service Pipeline

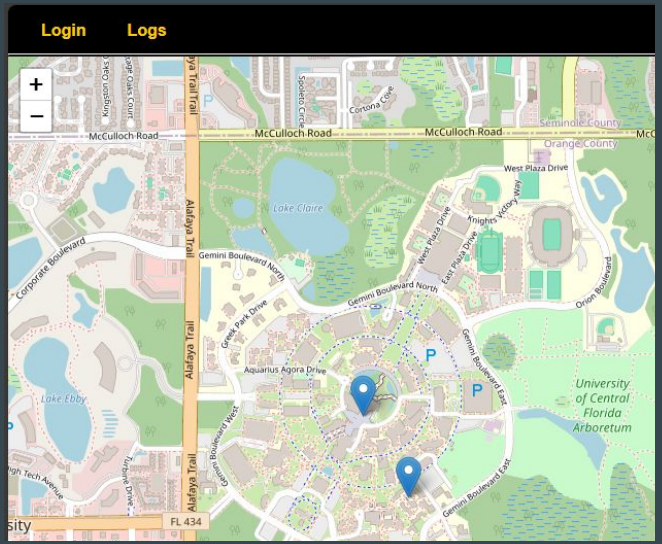


- Device → Server: ESP32/SIM7600 sends GPS + classification data over HTTPS
- Server → Database: Nginx + Flask handle requests, log activity, and store results in SQLite
- Database → Frontend: Tailwind/Alpine.js UI with Leaflet maps displays real-time data

UI/UX Design

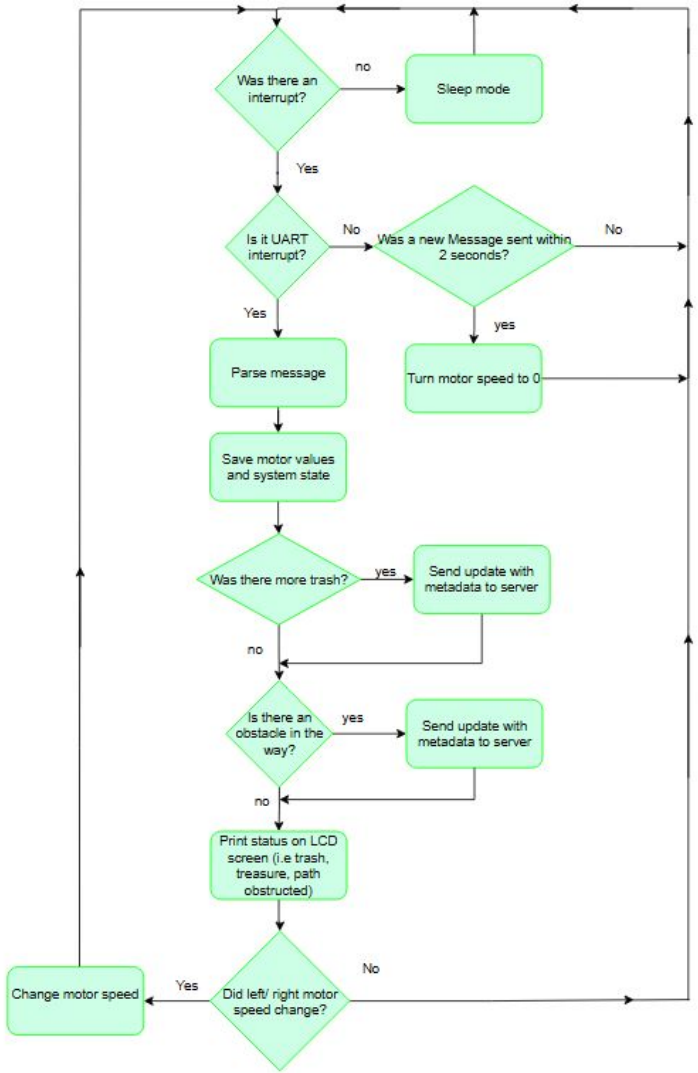
- Current map of items is on the bottom and goal design is shown on top.
- Tailwind + Alpine.js used for fast, reactive UI with minimal load over 4G.
- Map built with OpenStreetMap tiles for free, cacheable GPS plotting.
- Flask backend structured to support real-time updates and mobile-first layout.





ESP32 software flow chart

- Responsible for:
  - Motor control
  - Sending AT commands/data to SIM module
  - Parsing data from Pi
  - Controlling user interface



Prototyping and Testing

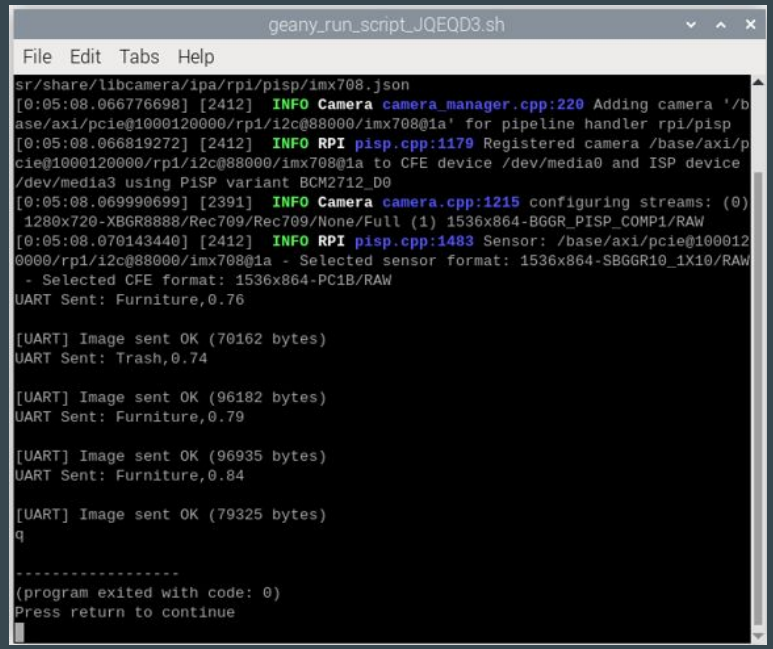
Object Classification Accuracy

Object classification accuracy is based on:

- Dataset size and quality
- Captured frames
- Environmental Variability

Pi to ESP32 UART Communication

- The PI successfully sent the images of observed objects and metadata to the ESP32.
- However, it was slow compared to PC transfer speeds.



ESP32 and SIM Upload Test

The ESP32 sends AT commands to the SIM7600 module to obtain GPS data. This data is then stored and transmitted to a server. The server parses the GPS data and uses the latitude and longitude values to plot the location on a map via the OpenStreetView API.

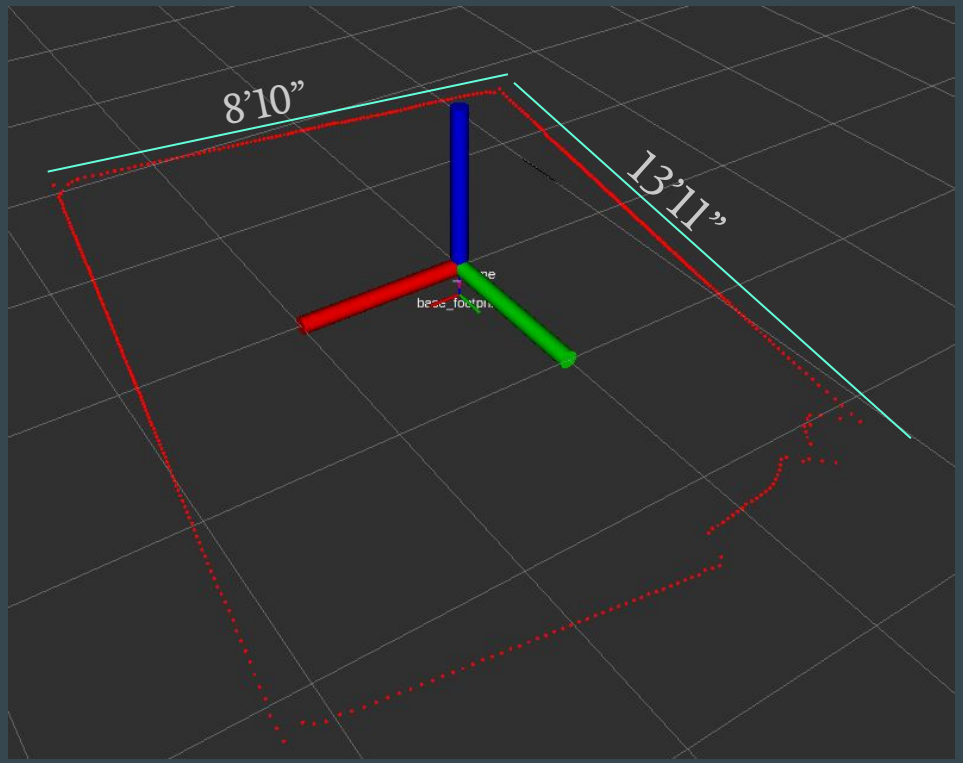
Server Logs (last 30 minutes)

- **2025-09-12 09:13:01**: POST /data: {'lat': 28.032, 'lon': -81.0023, 'timestamp': datetime.datetime(2025, 9, 12, 9, 13, 1, 31171)}
- **2025-09-12 09:13:02**: POST /data: {'lat': 28.1, 'lon': -81.05, 'timestamp': datetime.datetime(2025, 9, 12, 9, 13, 2, 340910)}
- **2025-09-12 09:13:03**: POST /data: {'lat': 28.15, 'lon': -81.08, 'timestamp': datetime.datetime(2025, 9, 12, 9, 13, 3, 670983)}

Front and Backend Interfaces

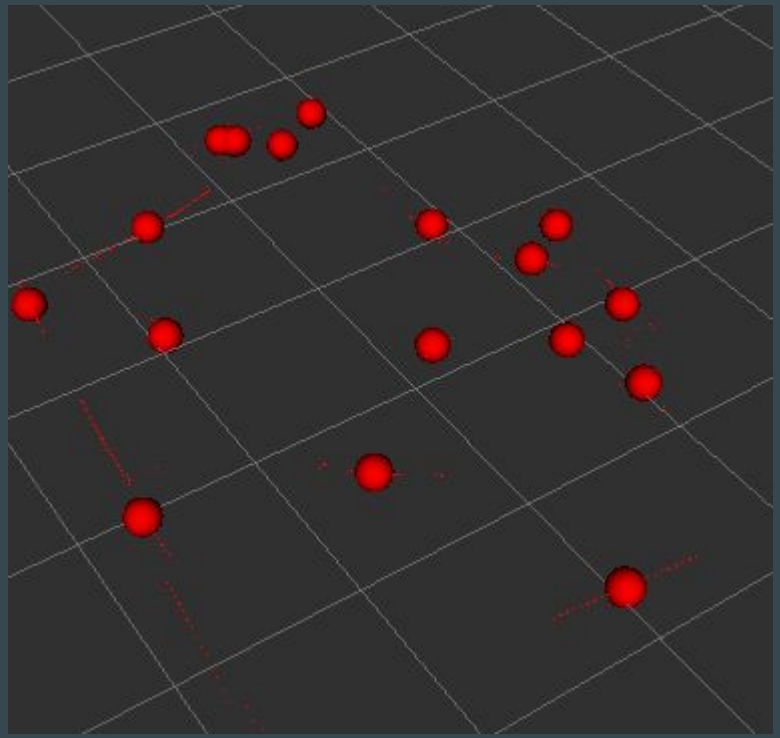
- Flask + Gunicorn backend came online quickly on a small(free) Oracle VM, with SQLite for lightweight storage of GPS and classification data.
- Nginx reverse proxy configured with relaxed TLS (TLSv1.1–1.3, !DH ciphers) to ensure SIM7600/ESP32 compatibility, while also handling HTTPS termination.
- Automated certificate management is a best-practice step for production which we don't have right now.
- Added logs page in Flask for quick debugging of ESP32/SIM7600 uploads
- Verified end-to-end data flow: device → Nginx → Flask → SQLite → map display
- Stress-tested API endpoints with malformed and valid requests to confirm resilience and reliable Flask responses.

LIDAR Testing



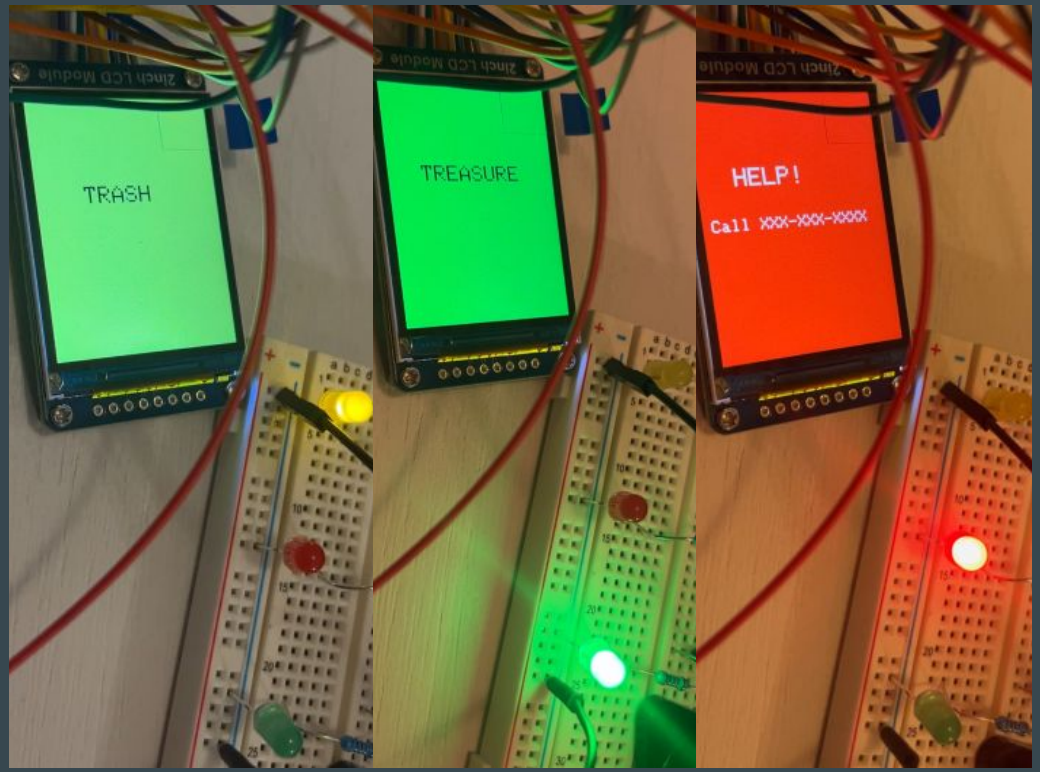
- Concerns of connectivity and usability have been put to rest.
- Map of my room with a 25° mounting angle.
- Distance 8’10”x 13’11” and points were unstable past 15 ft. at a 10Hz sample rate.
- Does not require PWM signal wire as the default kit comes with.

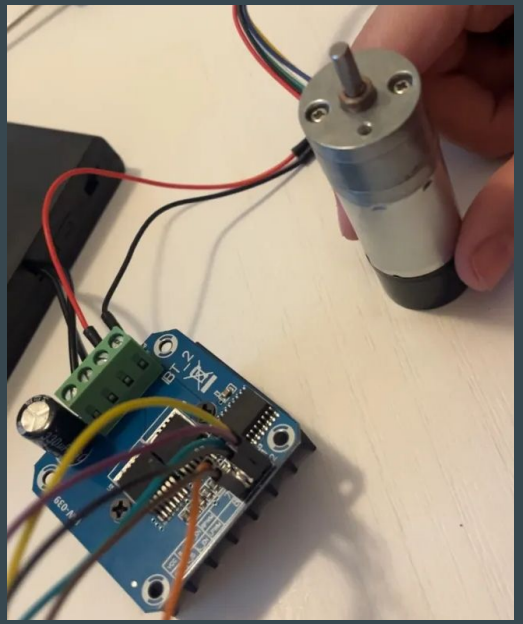
LIDAR Testing



- Using algorithms like DBSCAN (Density-Based Spatial Clustering of Applications with Noise) to group nearby points into clusters.
- Adapting from indoor tests to outdoor application.
- Calibrating distance with ESP32 for ground truth.

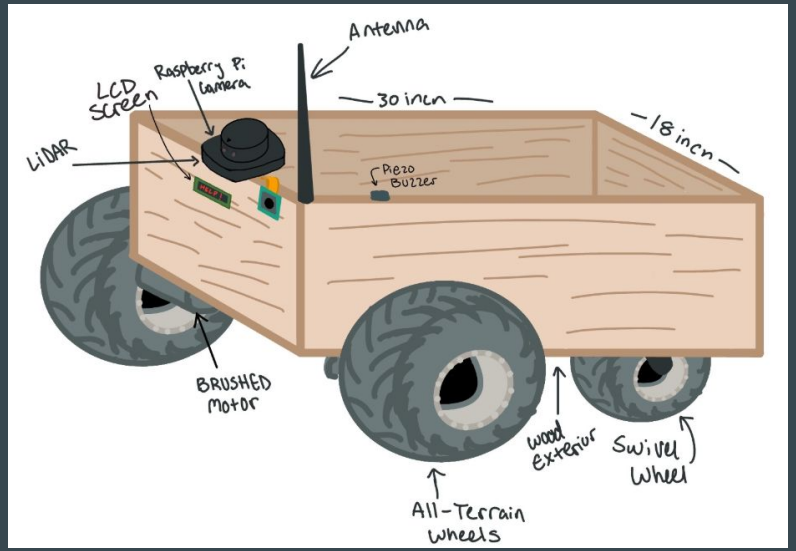
Peripheral testing





Construction of Chassis

- Constructing frame out of wood
- Securing motor
- Attaching wheels



Administrative Content

Bill Of Materials

Component	Sub-system	Quantity	Unit cost	Total
Raspberry Pi	Autonomous driving	1	$300	$300
LIDAR	Autonomous driving	1	$100	$100
Esp32 chip	Autonomous driving	1	$6	$6
PCB	All	5	$100	$250
PCB Components	All	*	$250	$300
Camera	Trash Detection	1	$25	$25
Chassis car kit	All	1	$120	$120
Antenna	Website	1	$50	$50
Sim Card	Website	1	$20	$20
Cell Plan	Website	1	$25/month	$25/month
Domain	Website	1	$5	$5
Miscellaneous	All		$64	$64
Ethernet Cable	Autonomous Driving + Trash Detection	1	$8	$8
Total				$1273

Bill Of Materials

- Under $1500
- Attempting to be cost efficient
- Software (~$200)
- Hardware (~$1000)
- Overhead (~$280)

Division of labor

Person	Role / Work Distribution
Ethan (main)	Embedded software and cellular communication.
Claire	
Claire (main)	Power supply, PCB design, and autonomous driving hardware.
Ethan	
Daniel (main)	Object classification CV software.
Nathan	
Nathan (main)	Autonomous driving software and website.
Daniel	

Plan for Completion

Construction of chassis.

Fully Assemble PCB and debug.

Improve object classification accuracy.

Calibrate sidewalk following from LIDAR to motor drivers.

Integrate object information on website from the Pi.