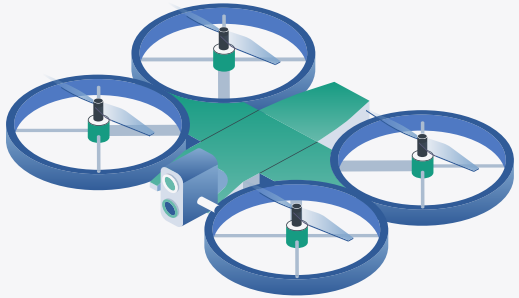


DRONE UAV
Autonomous System for Tracking, Recon, and Assistance

Benjamin Kim (bkim222@ucr.edu), Curtis Ly (cly046@ucr.edu), Pryce Matsudaira (pmats004@ucr.edu)
2/26/2025

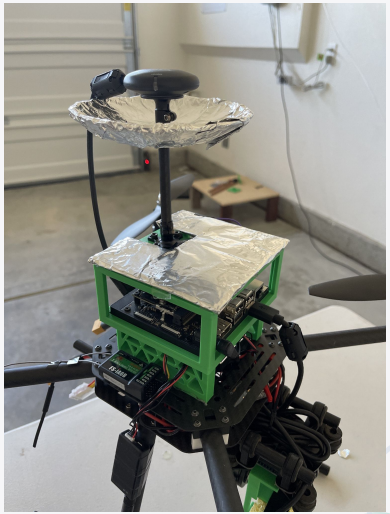
ASTRA (Autonomous System for Tracking, Recon, and Assistance)

Mission Statement:

To enhance SAR efficiency through operation in hazardous areas, providing aid to human rescuers. This method utilizes a higher degree of freedom providing scalability and adaptability to allow for a versatile use in different scenarios.

INTRODUCTION

- **Concept and Design Overview**
  - Core Components: computer vision (YOLOv8), GPS tracking, and emergency data communication systems for real-time situational awareness
- **Technical Principles:**
  - Autonomous Navigation: PX4 FC with GPS, Waypoints, and MAVLink telemetry
  - Computer Vision: Pre-trained YOLOv8 model for real-time human detection
  - Communication System: GPS coordinates, video feed, & alert to ground station in real time via simple WiFi protocols
  - Power System: 14.8V 4S LiPo Battery, ensuring at least 15 minutes of flight time
- **Specific Scenarios and Applications**
  - Lost at Sea
  - Forest fires

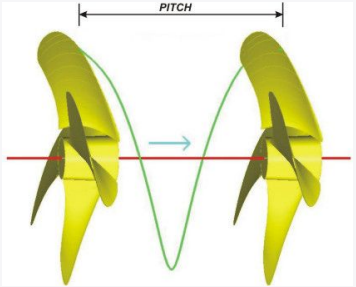


RELATION TO ELECTRICAL ENGINEERING

- **Embedded Systems & Microcontrollers**
  - Pixhawk 4 flight controller handles sensor data processing, flight control, and motor actuation
  - Jetson Orin Nano acts as an embedded AI computing unit, running machine learning models for object detection
  - Organize systems to ensure efficient communication and data transmission between components
- **Power System & Electronics**
  - Required to ensure proper current draws and voltage specifications
  - Meticulous organization and arrangement of components and wiring
- **Signal Processing & Communication**
  - Filtering external EMI signals and ensuring reliable data transmission
  - Setup wireless communication protocols, telemetry systems, and real-time data processing

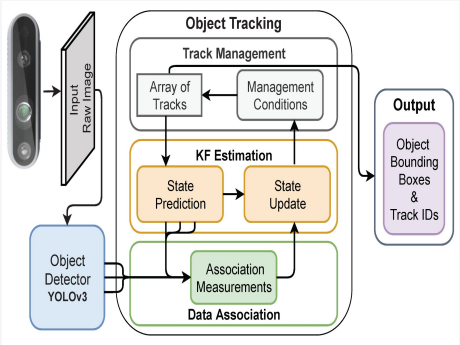
TECHNICAL DESIGN OBJECTIVES

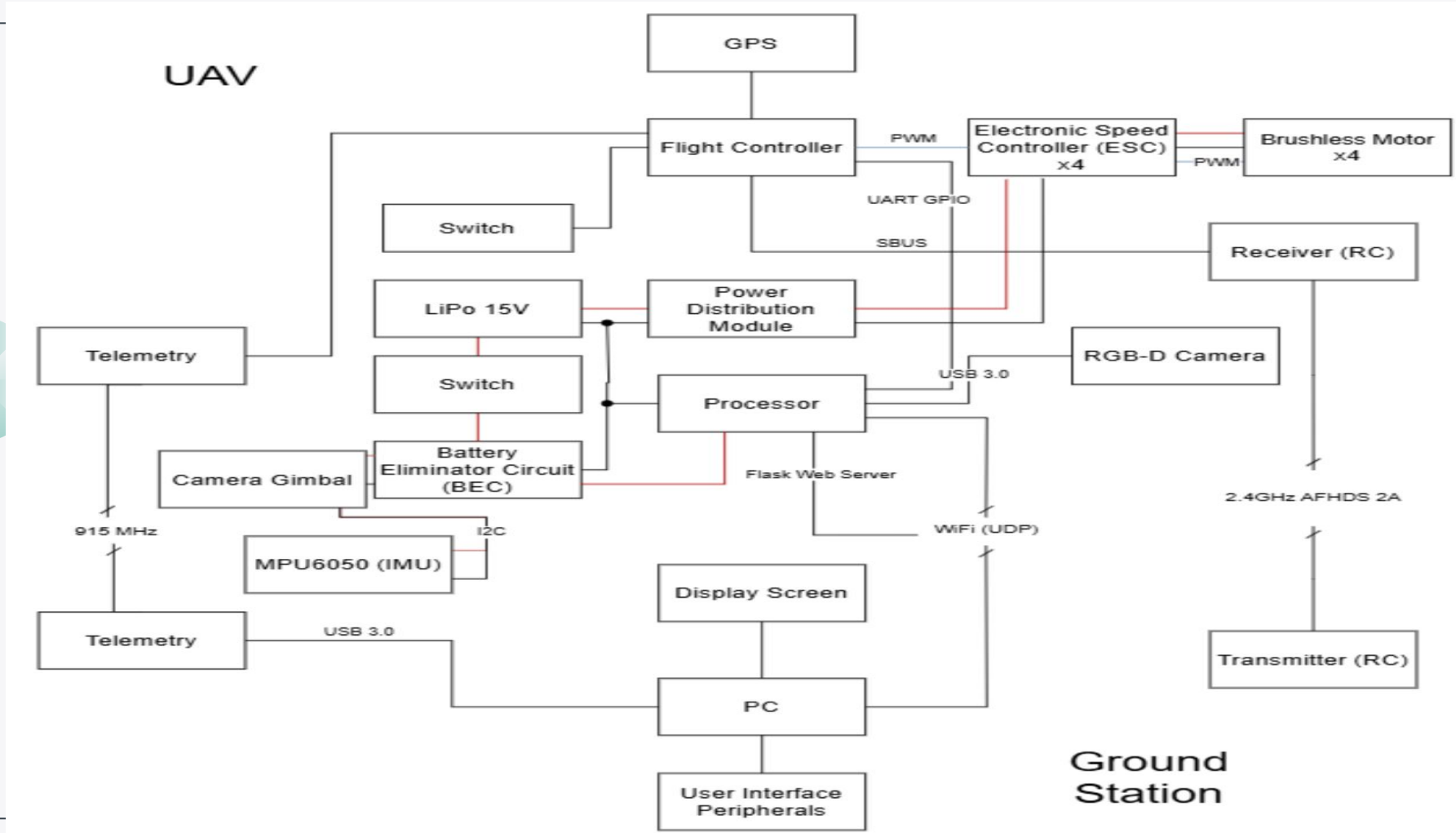
- Waypoint Navigation/Simulated Autonomy: Achieve a stable simulated autonomy covering a specified region for search
  - Altitude parameter 9-10 ft
  - Speed: 5 mph
  - Maximize stability (CoM, MoI)
    - 1147 propellers
- Power and Weight Optimization (IDLE/STANDING):
  - Pixhawk FC (P=4.5W=15V*0.3A)
  - Pixhawk FC + Jetson (P=15W=15V*1A)
  - Pixhawk FC + Jetson + Active AI Processing (P=19.5W=15V*1.3A)
  - Battery Specifications: 4S 3300mAh 50C
    - 3.7 nominal voltage ⇒ 3.7*4 = 14.8V
    - 4.2 fully charged ⇒ 4.2*4 = 16.8 V
    - milliamp-hour represents capacity ⇒ 3300mAh = 3.3 A for 1 hour or 6.6A for 30 min ...
  - Total takeoff weight ~2.5kg or 5.5lbs



TECHNICAL DESIGN OBJECTIVES

- Video Transmission & Real Time Performance: Ensure low-latency to provide real-time monitoring
  - Protocol: WiFi (UDP and Flask Web Server)
  - Latency Target <= 800ms (Real-time transmission)
  - Camera Performance: >= 20fps (balance between quality and hardware requirements)
- Human Detection Algorithm
  - Max Detection Range: 50m
  - Detection accuracy: >=90%




UAV
GPS
Flight Controller
PWM
Electronic Speed Controller (ESC) x4
PWM
Brushless Motor x4
UART GPIO
Switch
SBUS
Receiver (RC)
LiPo 15V
Power Distribution Module
Telemetry
Switch
RGB-D Camera
USB 3.0
Processor
Camera Gimbal
Battery Eliminator Circuit (BEC)
Flask Web Server
2.4GHz AFHDS 2A
915 MHz
I2C
WiFi (UDP)
MPU6050 (IMU)
Display Screen
Telemetry
USB 3.0
Transmitter (RC)
PC
Ground Station
User Interface Peripherals

PROGRAM FLOW CHART

01

Define Region

Set pre-defined waypoints and flight parameters to simulate autonomous flight

02

Begin Search

The drone navigates autonomously along the defined waypoints, running CV process to survey an area

03

Transmit Video Feed

Live drone camera feed is streamed via a Flask Web Server for real-time monitoring

04

Person Detected

AI based object detection processes frames to detect humans

05

Alert System

If a person is detected, GPS coordinates and detection logs are transmitted to the gs

06

Data and Mapping

The collected data is then visualized on a UI displaying a map for SAR

ASTRA LAUNCH

Overview of Software

01

Human Detect

YOLOv8-based human detection

02

GUI

Tkinter-based live stream of camera feed, map view, & GPS data

03

GPS Sender

Reads GPS data via MAVLink, & transmits latitude, longitude, & altitude data via UDP to ground station

04

GPS Receiver

Acts as a UDP server to receive GPS data, & outputs location

05

SOS Transmitter

Monitors & reads from Human Detect log file, & transmits output via UDP to ground station

06

SOS Receiver

Receives & displays UDP human detection results, & plays an alert sound upon detection

TECHNICAL CHALLENGES

Challenge	Solution
GPS Denied Regions and Inaccuracy	There were many challenges with getting the GPS to lock. Mostly because of electromagnetic interference caused by other communication systems such as our camera. The fix was to add absorbers and shielding using copper tape to reflect and absorb interfering signals.
Camera Feed Latency Issue	Originally we had about 2.7s of lag on the camera feed. This was fixed setting a frame queue size to 2 to prevent excessive buffering.
Camera Stability Issue	During flight, the camera would be unreliable for human detection because there is external noise caused by vibration. To fix this, added sponges to dampen the effects of vibration.
Accessible IP Address	Since the project relies heavily on wireless communication, the IP address of each system must be known and accessible at all times. Therefore we manually set a static IP address for each system
Offline Mapping	Since real-time mapping of the GPS coordinates requires Internet, a work around was to simulate by manually importing a static map and have two conditions (online vs offline mode) so that the UI still proceeds in the case if internet is not accessible

TECHNICAL CHALLENGES

Challenge	Solution
Processor Selection	For the scope of our project we chose the Jetson Orin Nano to handle autonomous and Computer Vision functions. Purely out of wanting to use more operations per second
PWM Control	Originally believed PWMs 500 Hz needed to be utilized for the ESCs, this would involve editing the parameters in QGroundControl. Turns, out our motors can be controlled using DShot which utilizes digital signal.
Serial Communication	Had to modify parameters and set Baud Rate (57600) to establish UART communication between Jetson and Flight Controller
Electrical Shorts	Added insulators to prevent connectors from shorting.Fixed issue where ESC was shorting because of contact with the carbon fiber frame.
Weight Balancing	Needed to account for battery weight with a good amount of flight time. 9000mah was too heavy causing propellers to rip off during flight test. Needed to account for moment of inertia i.e. where to place Jetson and GPS module on the frame.

COMPUTER VISION: YOLOV8

Algorithm giving accuracy, speed, & flexibility for object detection, segmentation, & classification tasks

Edge Deployment

- **DepthAI Pipeline**

OpenVINO IR → blob file → DepthAI

- **Low-Power Efficiency**

Supports faster inference, optimal for low-power hardware & efficient drone operation

- **Ideal Balance of Speed & Efficiency Hardware Deployment**

Backbone Architecture

- **CSPDarknet53**

Extracts features from image with an optimized process of reduced redundant calculations & improved speed

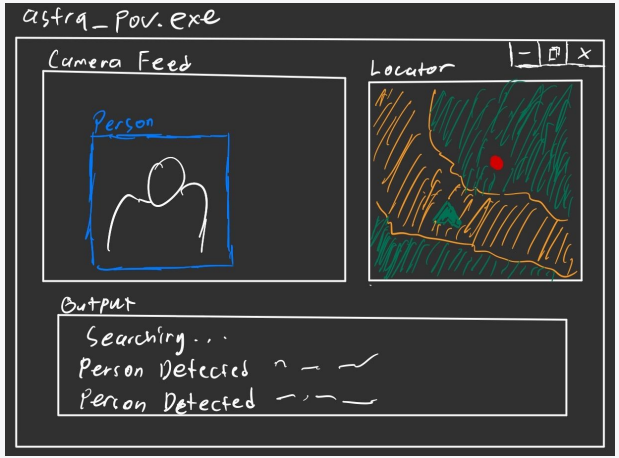
- **Additional Features**

C2f (Cross-Stage Partial Fusion), Revised Path Aggregation Network (PAN), CIoU (Complete Intersection over Union), Anchor-free

ASTRA UI

The goal of this GUI is to provide a single click application to monitor drone operations. AstraUI supplies users with a friendly interface that visualizes the drone's camera feed, displays GPS telemetry data, and maps your precise location. ASTRA UI Utilizes Tkinter which is a standard GUI (Graphical User Interface) library for Python.

ARCHITECTURE



ASTRA UI Concept

- **Main Frame:**
  - Grid Layout
    - Left Grid → Camera Feed
    - Right Grid → Map View
    - Bottom Grid → Terminal
- **Camera Feed Processing:**
  - GUI reads transmission coming from the video camera (CV detection algorithm, video frames) and displays this camera feed to the application.
- **GPS Data Handling:**
  - GUI reads telemetry data (coordinates) from the GPS and displays it to the terminal.
  - GUI pulls telemetry data from script “test_receiver.py” and displays coordinate location and Google Maps View into the feed
- **Internet connectivity:**
  - GUI examines internet connectivity and initializes GUI in offline map view mode or online map view mode

SOFTWARE

```
# Start output monitoring (for GPS telemetry)
self.start_output_monitoring()
```

```
if "Coordinate:" in line:
    coord_str = line.split("Coordinate:")[1].strip()
    lat, lon = map(float, coord_str.split(','))
    self.coordinates_received = True
    self.root.after(0, self.update_coordinate_display, lat, lon)

    # Start terminal output monitoring
    self.output_thread = threading.Thread(target=read_output, daemon=True)
    self.output_thread.start()
```

Starts output monitoring by running receiver and extracting GPS coordinates stored in (lat,lon)

```
def stream_reader(self):
    #Reads video stream from the camera.
```

Separates Thread for fetching frames so it does not interfere with main processes in main GUI loop

```
# Setup
self.frame_queue = queue.Queue(maxsize=2)
```

To mitigate camera feed latency this creates a queue with a max size of 2 frames which prevents excessive buffering

```
def update_frame(self):
    #Updates video frame in the Astra UI.
```

update_frame function handles updating the video frame and resizing the frames UI display

```
if self.running:
    self.root.after(10, self.update_frame)
```

Frames update every 10ms and grabs from the queue ensuring it only displays the most recent frame.

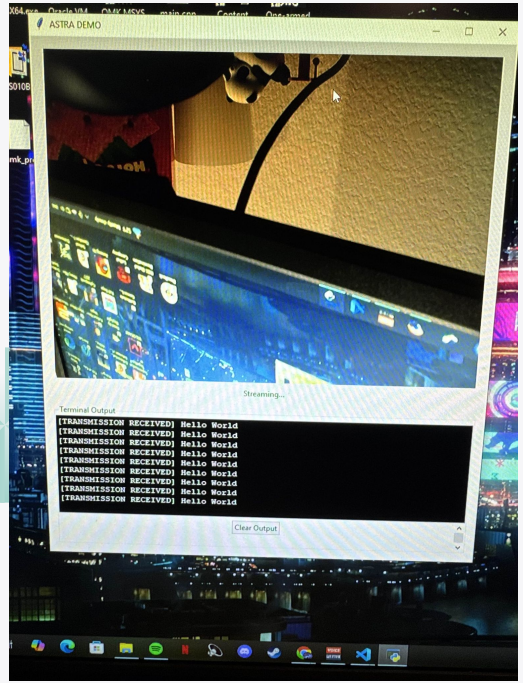
```
# UDP Setup
UDP_IP = "0.0.0.0"  # Listen on all network interfaces
UDP_PORT = 5005     # Same port as sender

# Sound settings
BEEP_FREQ = 750    # Frequency in Hz
BEEP_DURATION = 100 # Duration in milliseconds

# Create and bind socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #AF_INET specifies IPv4
sock.bind((UDP_IP, UDP_PORT)) #SOCK_DGRAM specifies UDP which uses datagrams.

print(f"Listening for detections on port {UDP_PORT}...")

last_play_time = 0  # To avoid rapid sound playing

while True:
    try:
        data, addr = sock.recvfrom(1024)  # Receive UDP packet, maximum packet size 1024
        message = data.decode().strip()
```

Initializes the UDP IP which listens for UDP packets on any connection. Initializes port 5005 for the UDP packets

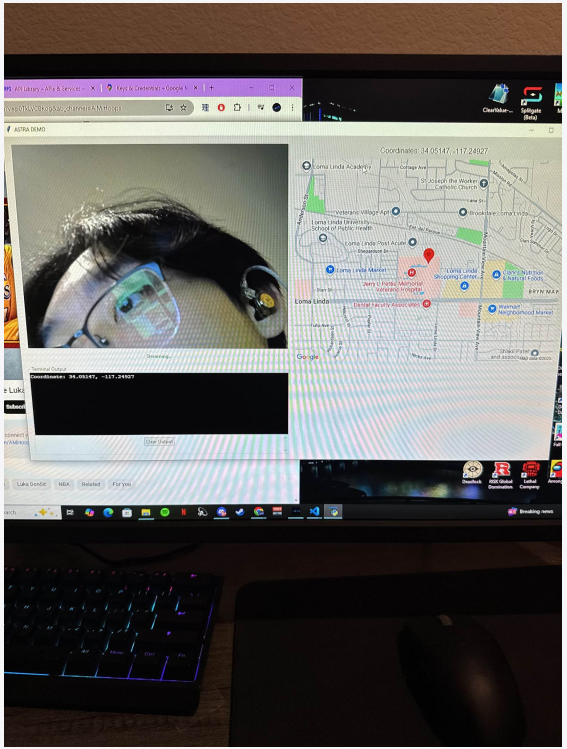
```
zoom = 17  # zoom level
map_size = "400x300"

# Maps static URL API
map_url = (
    f"https://maps.googleapis.com/maps/api/staticmap?"
    f"center={lat},{lon}&zoom={zoom}&size={map_size}" # handles map sizing and zoom
    f"&markers=color:red%7Clabel:A%7C{lat},{lon}" # drops a pin at the coordinates (lat,lon)
    f"&key={self.maps_api_key}" #our API key
)
```

If online and coordinates are received, API pull request is created and image is resized and displayed to map feed.

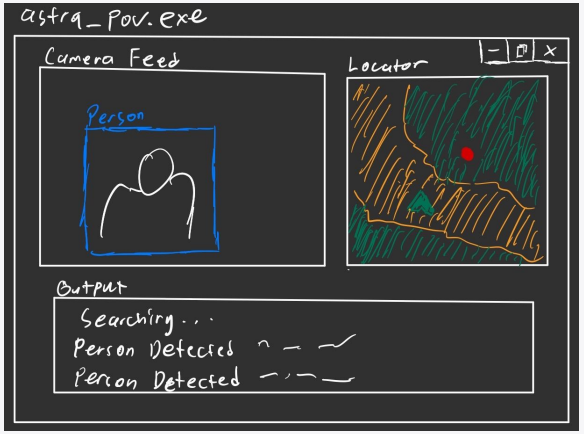
```
def load_default_map(self):
    maps_folder = os.path.join(self.script_dir, "maps") #finds maps directory inside of ASTRA
    default_map = "map_default.png"
    map_path = os.path.join(maps_folder, default_map)
```



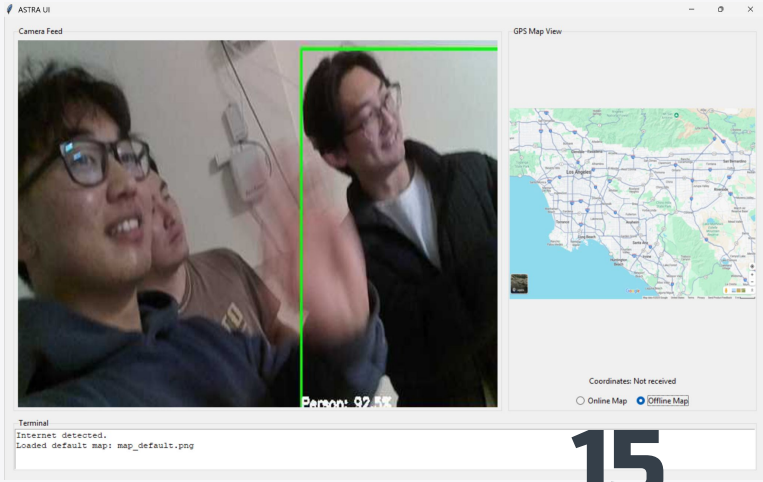
Transmission + camera test



Beta (no offline)

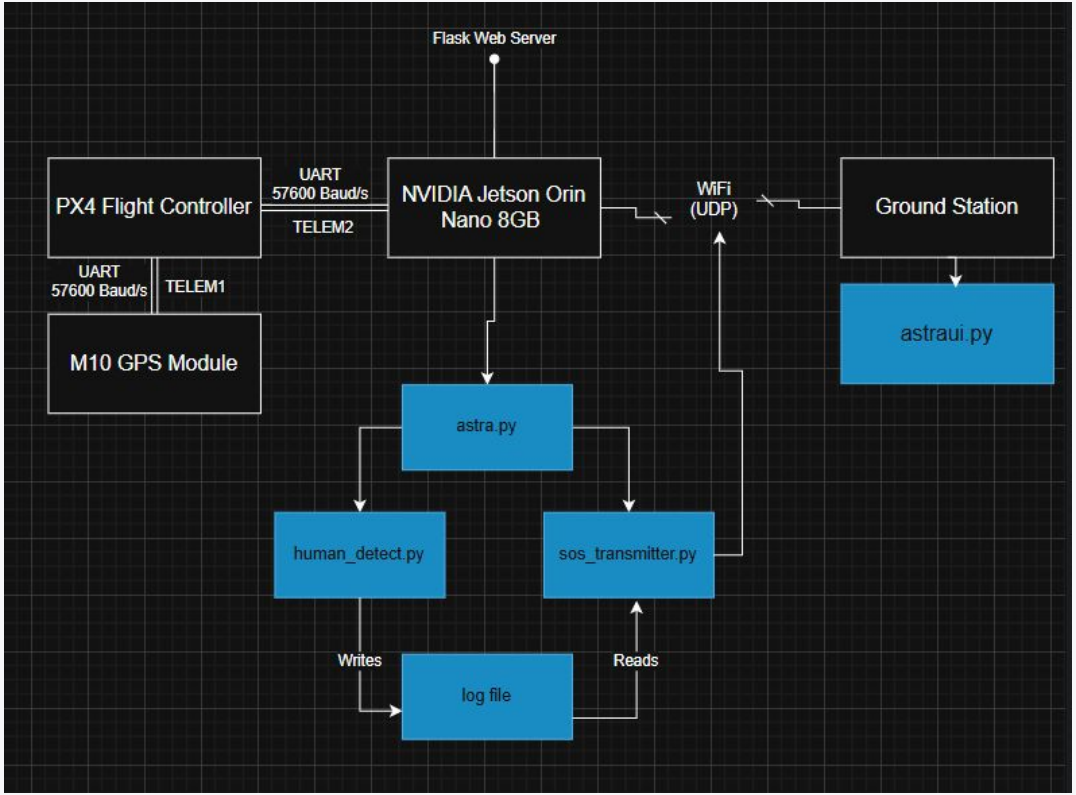


Concept

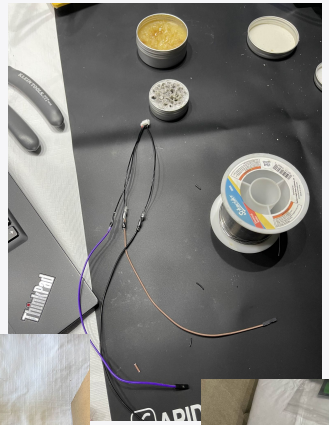


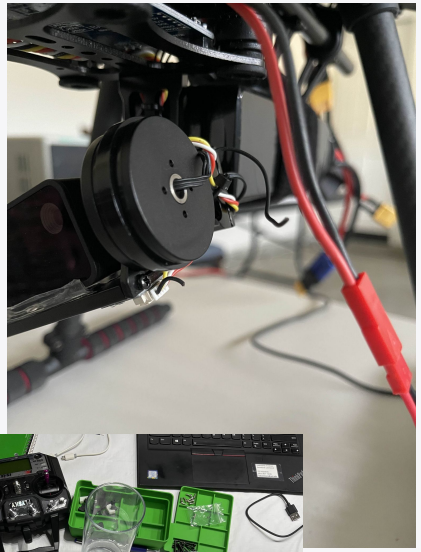
UI

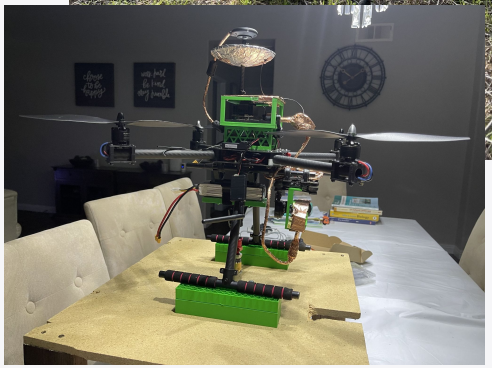
DATA COMMUNICATIONS

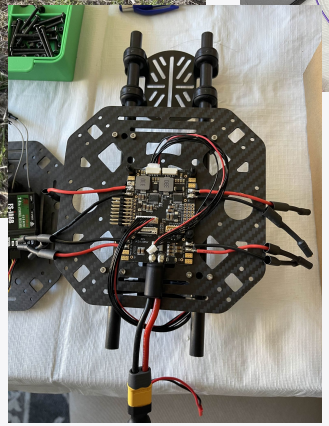


HARDWARE - UAV

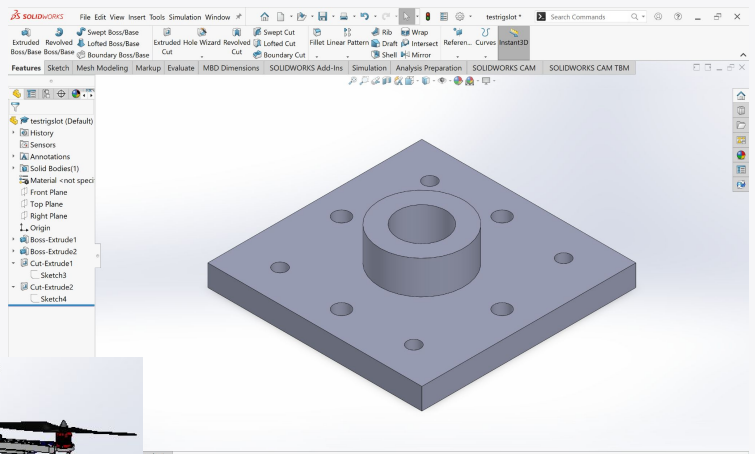


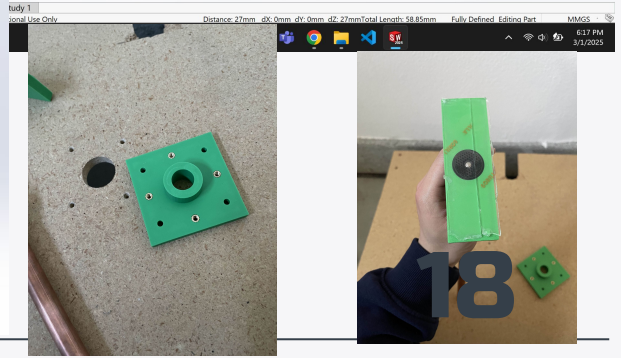






HARDWARE - DRONE TEST RIG





SIMULATED AUTONOMY

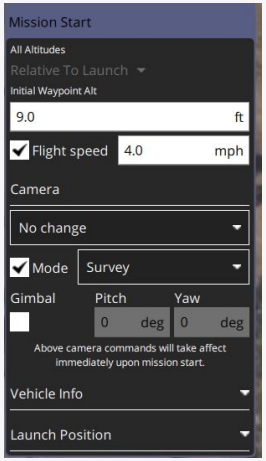
Simulated Autonomy through the use of QGroundControl

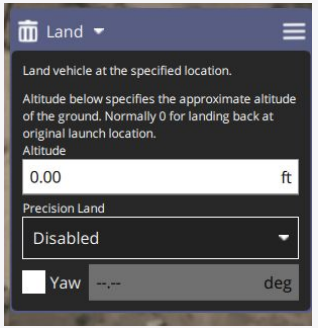
- Method:
  - Waypoint Tracking
- Additional Waypoint Features:
  - Adjusting Flight Speed
  - Setting Altitude
  - Configuring Hold Times
- Objective:
  - Simulate drone scanning a survey area detecting individuals in need of rescue
- Execution:
  - Drone starts mission → initiates takeoff
  - Drone maneuvers waypoints through pre-planned mission path
  - At designated waypoints (5,8,9), drone initiates a hold which simulates the drone autonomously finding a person and running our detection and alert software

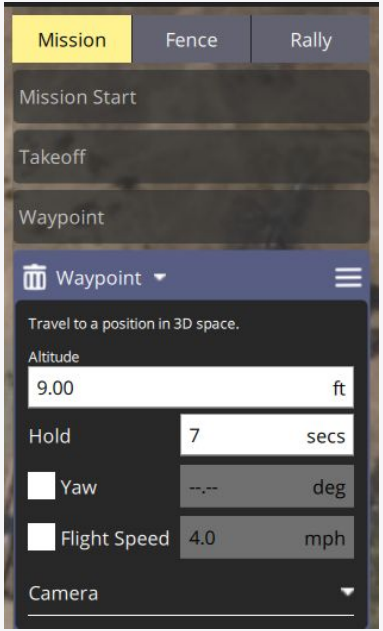
SIMULATED AUTONOMY

Future Implementation:

- ➔ Optimized Path Planning
  - A-star algorithm (finds the optimal pathing)
  - Integration of A-star with camera (real-time image processing) and LiDAR







DESIGN CONSIDERATIONS

Realistic Constraints:

- Time
  - Two quarters (Maximum of 20 weeks)
- Skill
  - Undergraduate-level expertise
- Weight/Size
  - Given the time constraints, LiDAR and GPS on the drone together was not feasible
- Power Consumption
  - Changed to lighter battery to conserve drone's moment of inertia

TOOLS & FRAMEWORKS

Robotics and Flight Control:

- ➔ Pixhawk 4 - open source FC
- ➔ MAVLink - lightweight messaging protocol
- ➔ QGroundControl (QGC) - software to control and monitor drone system and hardware

AI & Computer Vision:

- ➔ YOLOv8 - lightweight model for object detection
- ➔ OpenCV - open source library for image processing
- ➔ PyTorch - Learning framework for training and running neural networks

Embedded Systems & Computing:

- ➔ NVIDIA Jetson Orin Nano - AI edge computing platform
- ➔ Ubuntu 22.04 (Linux OS) - used for development
- ➔ Python - primary language used for software development

Network & Communication:

- ➔ Telemetry
- ➔ UART (TX/RX GPIO)
- ➔ Ground Control Station (GCS) - custom made to monitor system

INDUSTRY STANDARD

- ➔ UART (Universal Asynchronous Receiver-Transmitter) - Digital Equipment Corporation, 1960
  - FC Communication with Jetson, GPS Module, & Camera
- ➔ I2C (Inter-Integrated Circuit) - Philips Semiconductor, 1982
  - Integrates IMU (Inertial Measurement Unit) on FC for motion data
- ➔ USB 3.0 - USB Implementers Forum, 2008
  - Jetson to Camera for high-speed data transfer
- ➔ Wi-Fi (802.11ac) - IEEE, 2013
  - User Datagram Protocol - David P Reed, 1980
    - Low latency Communication Between Hardware
    - Remote Access
    - Communication with MAVLink for Telemetry & Drone Control

219

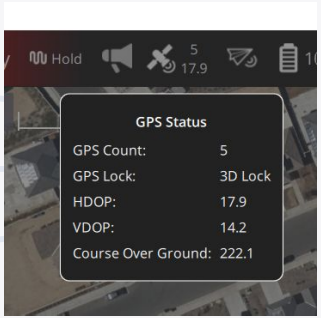
DEMO

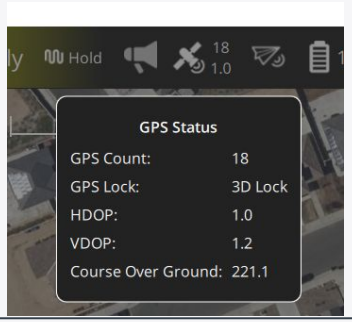
Demo4

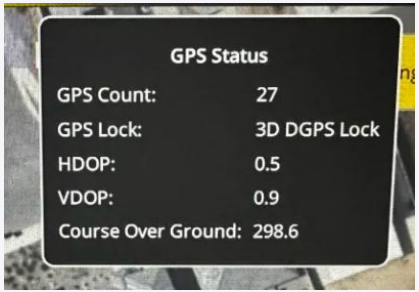
Demo2

TEST REPORT: GPS (EMI) ISSUE

- Many onboard components such as the OAK-D camera emits heavy amounts of EMI (Electromagnetic Interference) which causes the GPS to lose stability in it's signal.
  - Virtual Dilution of Precision
  - Horizontal Dilution of Precision
- Solution: To wrap foil around components to absorb and reflect unwanted electromagnetic signals
- Add a Ferrite core (LPF) to a USB 3.0 cable to act as a filter
- GPS loses connection immediately after camera is running in proximity within 2 ft.

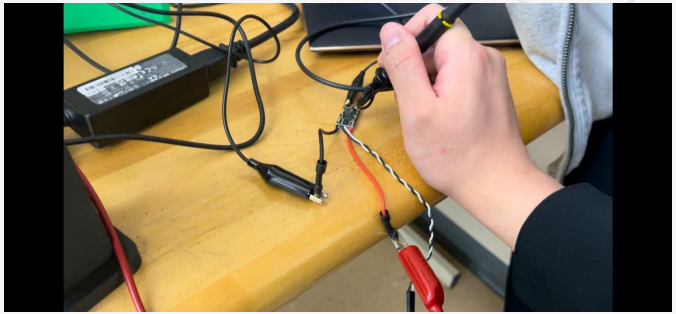




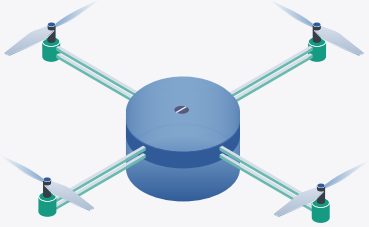


TEST REPORT: Hardware Damage

- Weight of the original battery caused propellers to detach from drone mid-flight as the motors were compensating for the extreme weight.
  - Caused drone to crash and damaged one of the motor ESCs
  - Debugging process to ensure it was only the ESC that was damaged included swapping to undamaged motors. Which narrowed it down to the ESC.
  - Desoldered the ESC and used a DC power supply to check if it was receiving the correct power specifications.
  - ESC received high current indicating a possible short further testing with multimeter found that the IC component controlling current regulation was damaged.



SUMMARY



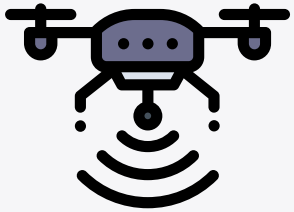
PROJECT OVERVIEW

- ASTRA is an autonomous UAV designed for SAR mission, specifically designed for hazardous areas
- Utilizes CV, GPS tracking, and autonomous navigation to assist humans

FUTURE IMPROVEMENTS

- Improve video feed with lower latency and higher resolution for better monitoring and detection performance
- System should accurately pinpoint human location and keep track of additional movements
- Improve on and or implement fully autonomous navigation and localization using SLAM

ACKNOWLEDGEMENTS

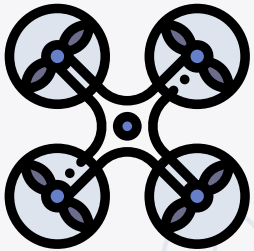


Dr. Chomko

- Recommendations on simplifying our project for realistic applications
- Wi-Fi for communication protocol instead of RF

TA Mehrnoosh

- Weekly Guidance/Suggestions on Technical Needs



THANKS!

DO YOU HAVE ANY QUESTIONS?

cly046@ucr.edu

pmats004@ucr.edu

bkim222@ucr.edu