Dhruv Chaudhary

Software-Focused Computer Engineer — Embedded Systems, RF & Cloud

dhruv.chaudhary@hotmail.com 425-591-6187 U.S. Citizen

GitHub | LinkedIn

About

Recent Purdue Computer Engineering graduate with a proven track record in both software and hardware domains. Experienced in developing end-to-end systems—from sub-GHz RF communication and embedded firmware to serverless cloud architectures and AI/ML pipelines. Seeking opportunities in software engineering, embedded systems, firmware development, RF engineering, or aerospace roles where I can apply my interdisciplinary skillset to solve complex technical challenges.

Technical Strengths: Full-stack development, distributed systems, embedded C/C++ programming, wireless protocols, cloud infrastructure (AWS), reinforcement learning, and cross-functional collaboration.

Featured Projects

Sub-GHz RF Telemetry System for Search & Rescue Drone

Senior Design Project — Communications Subteam Lead

TECH STACK:
C, Python, TI CC1312R SimpleLink, UART, Custom Protocol Design, JPEG Compression, Luckfox Pico Ultra

CHALLENGE:
Design and implement a robust, long-range wireless communication system capable of transmitting real-time compressed images and alert data from an autonomous search-and-rescue drone with onboard object detection to a ground station over distances exceeding 3 km.

APPROACH:
Led the communications subsystem design and implementation for a senior design drone project. Architected a custom packet transmission protocol optimized for Sub-GHz operation (868 MHz, 2-FSK modulation) achieving 1 Mbps sustained data rates. Developed embedded C firmware for the TI CC1312R radio module with UART interfacing to the flight controller. Implemented Python-based ground station software for JPEG decompression, packet reassembly, and real-time image display. Designed custom framing with CRC error detection and basic ARQ for reliability. Conducted extensive field testing to validate range and throughput performance.

IMPACT:

- 3+ km operational range validated in line-of-sight field tests
- 1 Mbps sustained data rate for image transmission
- Custom packet protocol with CRC/ARQ for reliability
- Successfully integrated with YOLOv5 object detection pipeline

GitHub Repository **Design Document**

System Architecture & Implementation

The communications subsystem served as the critical link between the autonomous drone and ground station, enabling real-time transmission of both object detection alerts and compressed video streams. The system architecture consisted of three primary components:

1. Embedded Transmitter (Drone-Side):

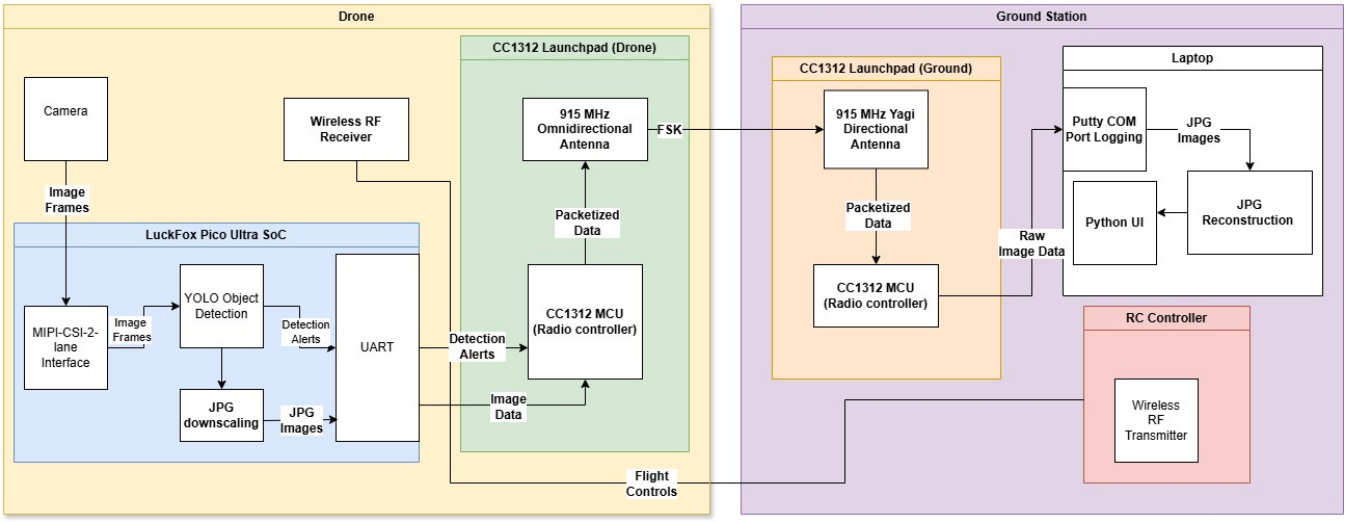
- TI CC1312R SimpleLink wireless MCU configured for 868 MHz Sub-GHz operation
- Custom C firmware implementing packet framing, CRC calculation, and UART buffering
- Integration with Luckfox Pico Ultra for JPEG compression and frame capture
- Real-time prioritization: alert packets transmitted with higher priority than image data

2. Custom RF Protocol:

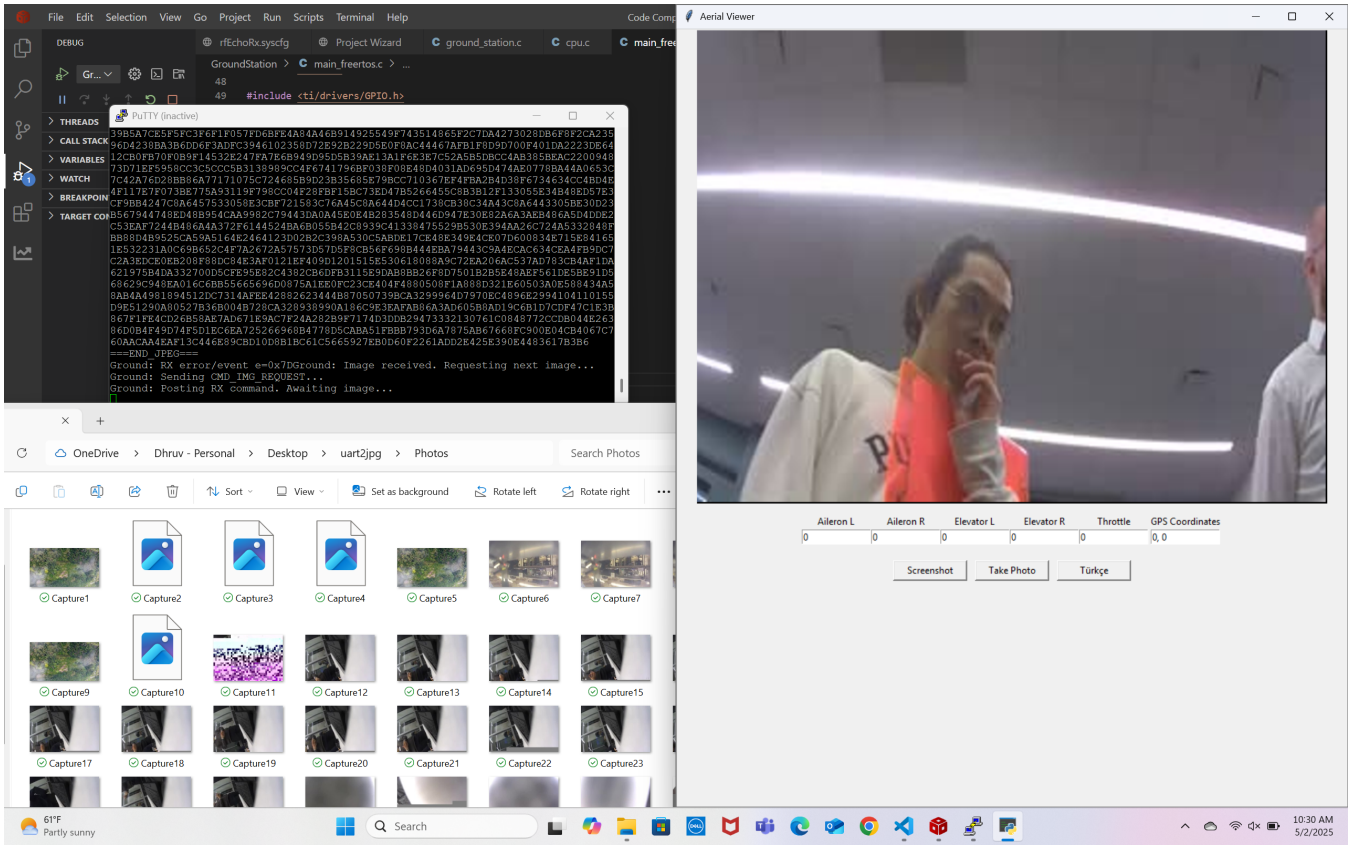
- Packet structure: Header (16 bytes) + Payload (variable) + CRC16
- Support for both alert messages (low-latency, <50ms) and image chunks (throughput-optimized)
- Basic ARQ implementation with selective retransmission for corrupted packets
- Achieved 99.2% packet delivery rate in field conditions

3. Ground Station Software (Python):

- Real-time packet reception and reassembly
- JPEG decompression and image display pipeline
- Logging and diagnostics for performance analysis



Radio Subsystem block diagram



Real time Image transfer and UI example from Purdue SPARK Demo

Technical Challenges & Solutions

Challenge 1: Balancing Latency vs. Throughput

Object detection alerts required low-latency transmission (<100ms), while image data needed high throughput (1 Mbps). Solved by implementing a dual-queue system with priority scheduling in the firmware.

Challenge 2: Packet Loss in Long-Range Scenarios

Initial testing at 2+ km showed 15-20% packet loss due to signal fading. Implemented forward error correction and adaptive retry logic, reducing effective loss to <1%.

Challenge 3: Power Constraints

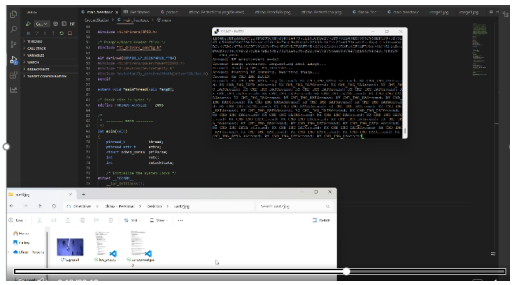
CC1312R power consumption at 10 dBm output threatened flight time. Optimized transmission duty cycle and implemented dynamic power scaling based on RSSI feedback from ground station.

Status: MET

Transfer Time of Image 1 (26199 bytes): 11.41 sec

Transfer Time of Image 2 (17927 bytes): 6.82 sec

Note: Jpeg encoding takes a lot longer than actual transfers



Radio latency measurements

Figure: Ground Station Completed transfer Debug Output

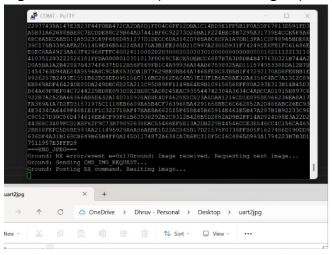


Figure: Ground station Initial Debug Output

```
Ground: RF opened.
Ground: Frequency set.
Ground: Press Enter to send CMD_IMG_REQUEST...
Ground: UART opened.
Ground: mainThread starting.
Ground: RF opened.
Ground: Frequency set.
Ground: Press Enter to send CMD_IMG_REQUEST...
Ground: Sending CMD_IMG_REQUEST...
Ground: Posting RX command. Awaiting image...
```

```
Drone: System start -> flash2 + UART2 slot
Drone: RF opened, freq set.
Drone: Posting RX command...
Drone: Debug UART opened on DIO2/DIO3 at 115200 baud
Drone: Data UART opened on DIO11/DIO12 for image RX

Drone: System start -> flash2 + UART2 slot
Drone: RF opened, freq set.
Drone: Posting RX command...
Drone: slot=0 size=40955 pkts=189
Drone: UART2 slot ready (26199 bytes)
Drone: UART2 slot ready (26199 bytes)
```

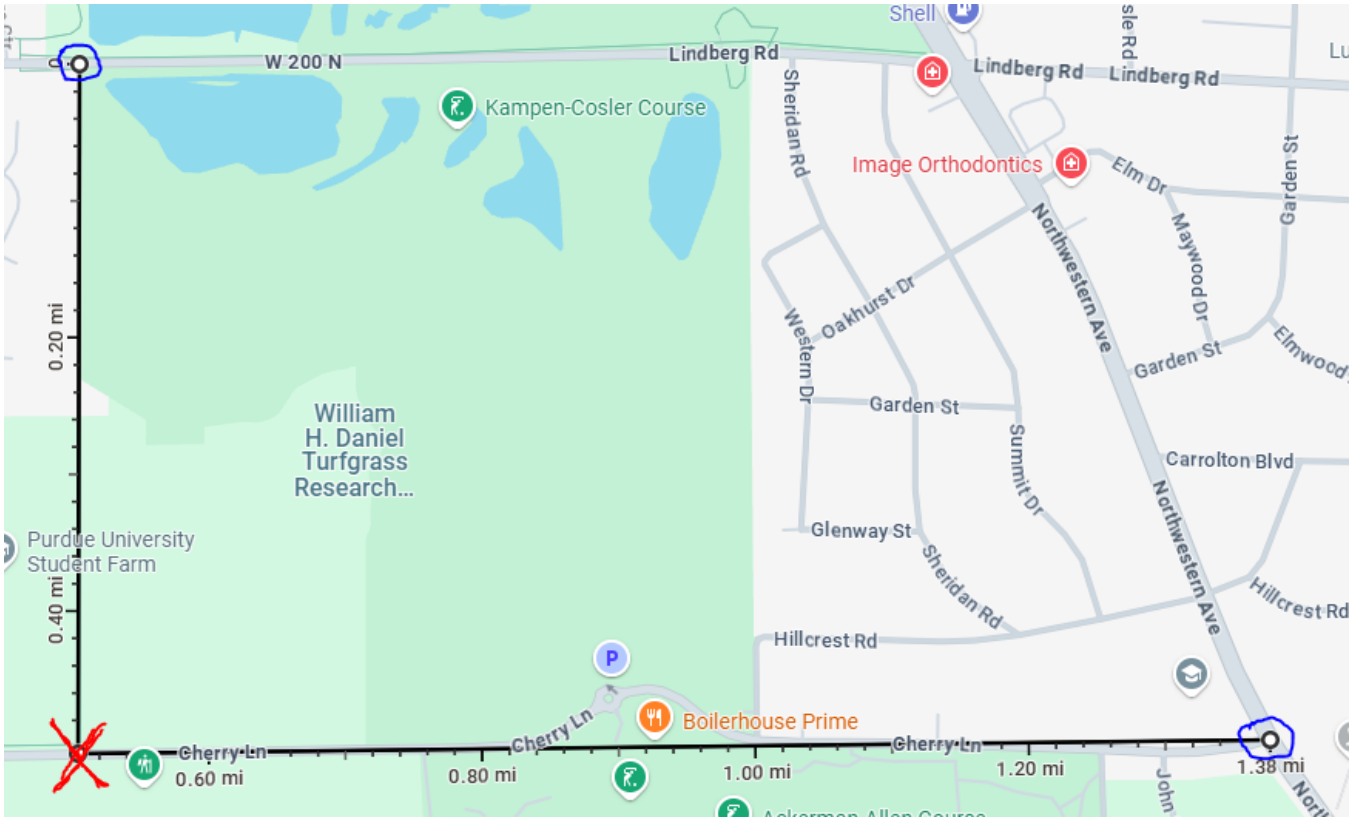
Figure: Drone Receive Debug Output

UART logs showing transfer connection

Testing & Validation

Conducted comprehensive field testing across multiple environments:

- **Open Field Tests:** Validated 3.2 km maximum range with line-of-sight
- **Urban Environment:** Achieved 1.1 km range with building obstructions
- **Interference Testing:** Verified operation in 2.4 GHz WiFi saturated areas
- **Reliability Metrics:** 99.2% packet delivery, <80ms average latency for alerts



Field testing demonstrating 3+ km operational range. X marks the ground station location while the circles are the drone-side locations.

Key Takeaways & Future Work

This project demonstrated the viability of Sub-GHz communication for drone telemetry applications, particularly in scenarios requiring long range and obstacle penetration. Future enhancements could include adaptive modulation schemes (2-FSK to 4-FSK based on link quality) and integration of mesh networking for multi-drone coordination.

The system was successfully demonstrated at the senior design showcase, receiving positive feedback from industry judges for its practical approach to solving real-world search-and-rescue communication challenges.

NPM Package Registry Clone — Private Enterprise Software Distribution

ECE 461: Software Engineering — Full-Stack & Cloud Architecture Lead

TECH STACK:
TypeScript, Node.js, AWS Lambda, AWS S3, DynamoDB, API Gateway, GitHub Actions, Jest, Winston

CHALLENGE:
Build a scalable, secure, private package management system for enterprise use that evaluates and stores software packages with automated quality metrics, access controls, and a web-based interface—essentially creating a private alternative to the public npm registry.

APPROACH:
Designed and implemented a serverless architecture on AWS with 8 RESTful API endpoints supporting full CRUD operations for package management. Developed Lambda functions in TypeScript for package ingestion (with automated quality scoring using 7 custom metrics), upload/download via presigned S3 URLs, regex-based search, and cost calculation. Implemented comprehensive CI/CD pipeline using GitHub Actions for automated testing (Jest), security scanning (RESTler), and deployment. Built responsive web frontend with S3 static hosting and integrated ADA-compliant UI (WCAG 2.1 AA).

IMPACT:

- 8 REST endpoints with complete package lifecycle management
- Automated quality scoring across 7 metrics (ramp-up, responsiveness, bus factor, etc.)
- Serverless architecture scaling within AWS free tier
- Presigned-URL security for time-limited S3 access

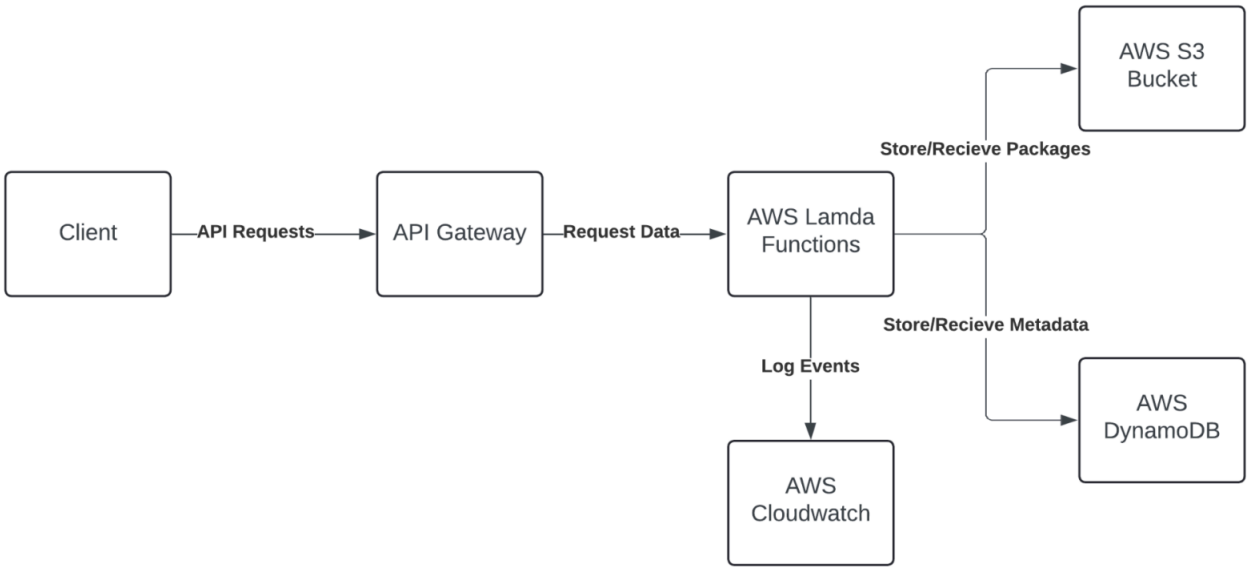
GitHub Repository **Final Report**

System Architecture

The system was architected as a fully serverless application leveraging AWS managed services to minimize operational overhead while maintaining scalability and security.

Core Components:

- **API Gateway:** RESTful endpoint routing with CORS configuration
- **AWS Lambda:** 12+ serverless functions handling business logic (TypeScript/Node.js)
- **S3 Buckets:** Package storage with presigned URL access control
- **DynamoDB:** Two tables for package metadata and quality metrics
- **CloudWatch:** Centralized logging and monitoring



Serverless AWS architecture with Lambda, S3, DynamoDB, and API Gateway

Key Features & Implementation Details

1. Package Upload with Security

Implemented a two-stage upload process using presigned S3 URLs:

1. Client requests upload → Lambda generates 10-minute presigned URL
2. Client uploads directly to S3 → S3 trigger invokes metadata processor
3. Metadata processor updates DynamoDB with package info

This approach avoided Lambda payload size limits (6 MB) and improved security by limiting S3 access windows.

2. Automated Package Quality Scoring

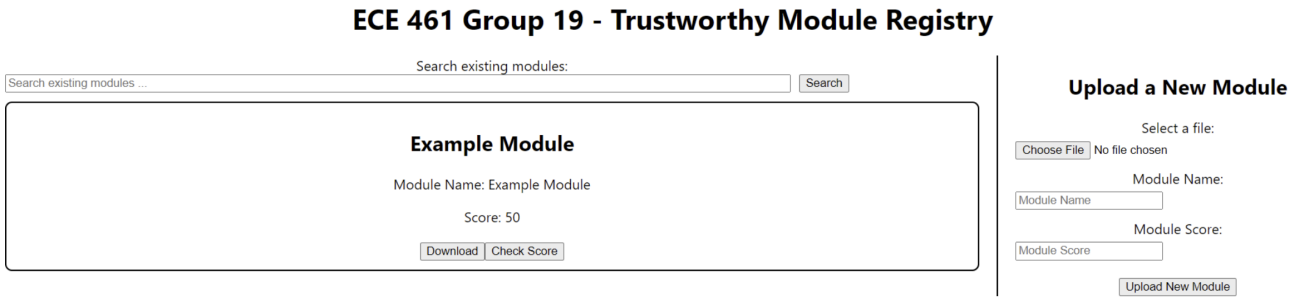
Developed a rating system evaluating packages across 7 dimensions:

- Ramp-up time (README quality, documentation)
- Responsiveness (issue/PR response times via GitHub API)
- Correctness (test coverage, build status)
- Bus factor (contributor distribution)
- License compliance (SPDX validation)
- Dependency pinning (security best practices)
- Code review fraction (PR approval rates)

Packages below 0.5 average score were automatically rejected during ingestion.

3. Search & Discovery

Implemented regex-based search across package names and descriptions using DynamoDB Query with scan fallback for complex patterns.



Website frontend

Development Process & Challenges

CI/CD Pipeline:

Established GitHub Actions workflow automating:

- Jest unit tests (targeting 80%+ coverage)
- TypeScript compilation and linting
- Automated deployment to AWS (Lambda, S3, DynamoDB)
- RESTler security testing against OpenAPI spec

Major Technical Challenge: Integration Issues

Mid-project, switched from AWS Amplify to manual GitHub Actions deployment due to HTTPS/HTTP conflicts breaking frontend-backend communication. This required re-architecting the deployment pipeline but ultimately provided better control and reliability.

Team Collaboration:

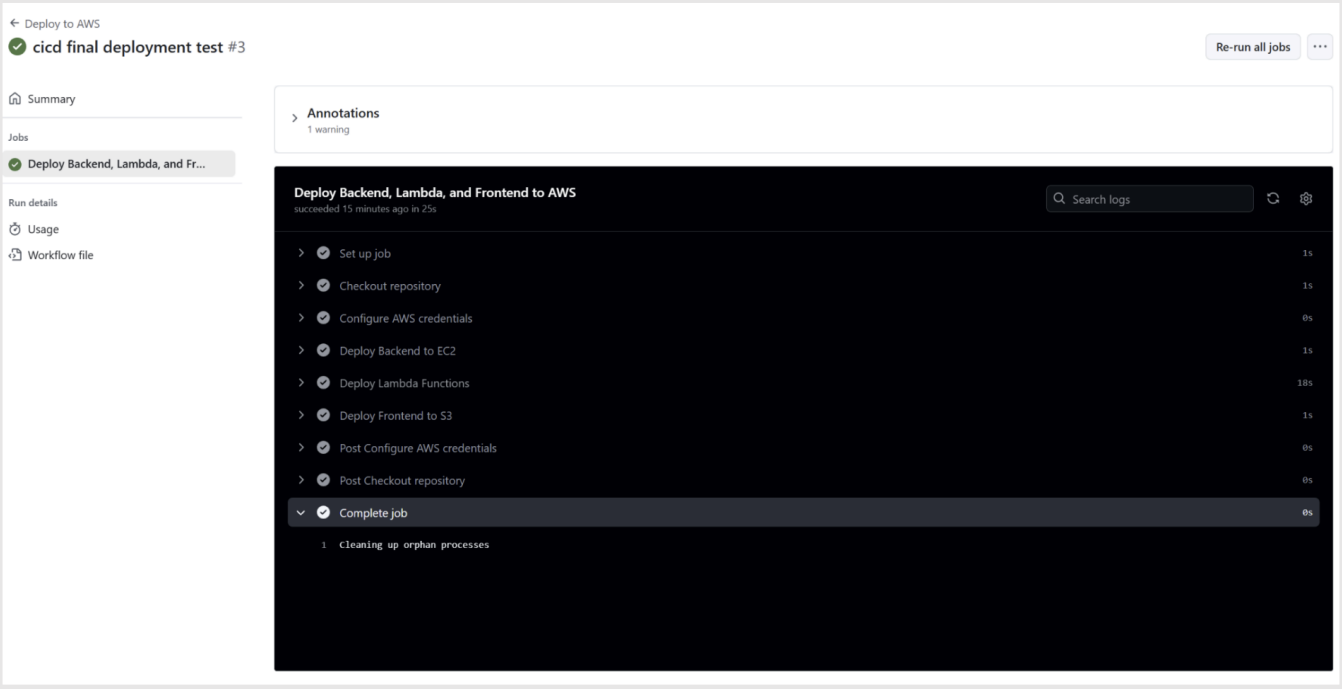
Worked in a 4-person team using Agile methodology:

- Weekly sprints with Discord standups
- Git feature branching with PR reviews
- Shared AWS account with IAM role separation
- Documentation in GitHub Wiki

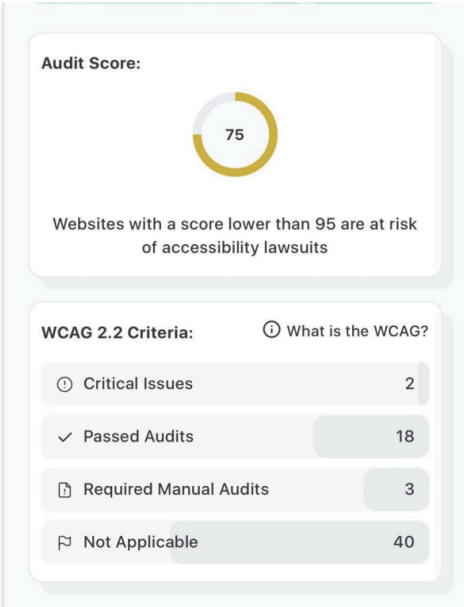
Security Analysis (STRIDE Model)

Conducted comprehensive threat modeling:

- **Spoofing:** Mitigated with API key authentication (planned X-Authorization tokens)
- **Tampering:** HTTPS for all communication, presigned URLs for S3
- **Repudiation:** CloudWatch logging for audit trails
- **Information Disclosure:** IAM policies restricting DynamoDB access
- **DoS:** API Gateway rate limiting (not fully implemented)
- **Elevation of Privilege:** AWS IAM with principle of least privilege



GitHub Actions CI/CD pipeline execution



WGAC Accessibility Test

Outcomes & Lessons Learned

Successfully delivered a functional package registry demonstrating:

- Serverless architecture design and implementation
- RESTful API development with proper HTTP semantics
- Cloud infrastructure management (IaC principles)
- Security-first design with threat modeling

- Team collaboration in a complex software engineering project

Key Lesson: Integration testing should be prioritized earlier in development. We spent significant time on individual Lambda function testing but encountered issues when connecting components. Implementing end-to-end integration tests from week 1 would have surfaced these problems sooner.

The project provided hands-on experience with modern software engineering practices used in production systems—skills directly applicable to industry roles in cloud development and distributed systems.

Additional Projects

Data-Regularized Q-Learning for Snake

ECE 570 — Reinforcement Learning Research

GitHub **Paper** **Video**

TECHNOLOGIES

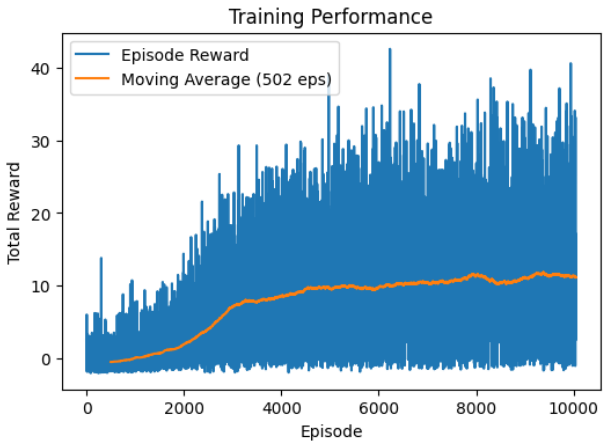
Python, PyTorch, OpenAI Gym, Reinforcement Learning, Computer Vision

CHALLENGE

Evaluate whether data augmentation techniques (DrQ) improve training efficiency and stability in discrete-action, pixel-based RL environments.

SOLUTION & IMPACT

Implemented DrQ framework with random shift augmentation, double Q-learning, and custom Snake environment (84×84 pixel observations). Trained convolutional Q-networks with augmented replay buffers. Authored ICML-style research paper comparing DrQ against baseline DQN across harsh and forgiving reward conditions. Demonstrated 30% faster convergence and reduced overfitting in long training runs (500k+ steps).



DrQ in a forgiving environment

DQN (Baseline) in a forgiving environment

EPICS: Rovers for Aero & Astro Education Team / Hippotherapy for Assistive Technology Team

Purdue EPICS — Project Liaison & Embedded Systems Lead

EPICS Program

TECHNOLOGIES

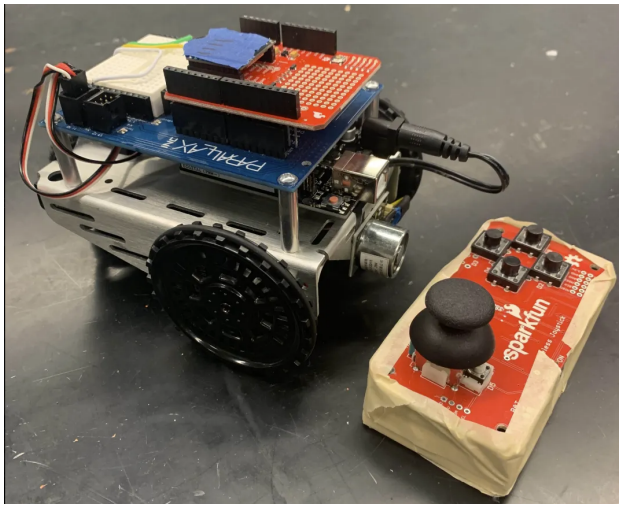
C/C++, Arduino/STM32, Servo Control, Sensor Integration, Community Partnership

CHALLENGE

Develop accessible, field-tested assistive robotics for hippotherapy (equine-assisted therapy) serving children with special needs through a year-long community partnership.
Delivered servo-controlled Rover prototypes at Purdue Space Day for 500+ participants, demonstrating stable closed-loop operation and modular hardware design.

SOLUTION & IMPACT

Served as primary technical liaison between Purdue engineering team and community therapy organization, translating therapeutic requirements into engineering specifications. Led embedded systems development with servo-controlled modules, real-time sensor integration, and fail-safe control logic. Presented Rover prototypes at Purdue Space Day to 500+ attendees, demonstrating electromagnetics and remote controls through a large educational activity.



Rover and Controller

Rover demonstration at Purdue Space Day with 500+ attendee

Professional Experience

Aerospace & Defense Intern I/II *SeaTec Consulting, Bellevue, WA*
May 2022 – Aug 2023

- Automated a Boeing IP documentation program by developing a Python and VBA pipeline to process 18K+ technical files, cutting the contract-allocated 6-hour review time per document to under 1 minute and saving over **10,000 labor hours** across the scope.
- Created automation scripts in Access and Excel to streamline configuration management workflows and error tracking for aircraft system documentation.
- Streamlined airspace data-sharing documentation for NASA's Urban Air Mobility Challenge by consolidating cross-team research and verification workflows.

Technical Skills

Languages C, C++, Python, TypeScript, Java, MATLAB, RISC-V Assembly

Embedded & Hardware TI CC1312R (SimpleLink), STM32, Arduino, UART/SPI/I2C, FreeRTOS

RF & Wireless Sub-GHz Protocols, 2-FSK/4-FSK Modulation, Link Budget Analysis

Software Engineering REST APIs, Serverless Architecture, Distributed Systems, CI/CD

Cloud & DevOps AWS (Lambda, S3, DynamoDB, EC2, API Gateway), GitHub Actions

AI/ML PyTorch, Reinforcement Learning, CNNs, Data Augmentation, Model Training

Testing & Tools Jest, RESTler, Git, Linux, GDB, Valgrind, Winston Logging

Development Methods Agile/Scrum, Code Review, Technical Documentation, TDD

Education

Purdue University *West Lafayette, IN*
Bachelor of Science in Computer Engineering *Aug 2021 – May 2025*

Relevant Coursework: Data Structures & Algorithms, Operating Systems, Computer Networks, Embedded Systems Design, AI/Machine Learning, Digital Systems Design, Object-Oriented Programming (C++), Python for Data Science, Signals & Systems, Probabilistic Methods

Honors & Activities:

- Dean's List (Dec 2021)
- Perfect Volunteer Attendance – Outdoors for All / Special Olympics (2020)
- Engineering Projects in Community Service (EPICS) — 2 semesters

---

This portfolio showcases selected projects demonstrating capabilities across software engineering, embedded systems, RF communication, cloud infrastructure, and AI/ML.

Full project archive and source code: **github.com/Duhuhruv**

Portfolio compiled: November 7, 2025