Allsky Camera Network for Detecting Bolides

Semester 2

Tyler Turner
Vincent Quintero
Jean-Pierre Derbes
Charles Derbes

Client
Dr. Csaba Palotai

What We are Observing

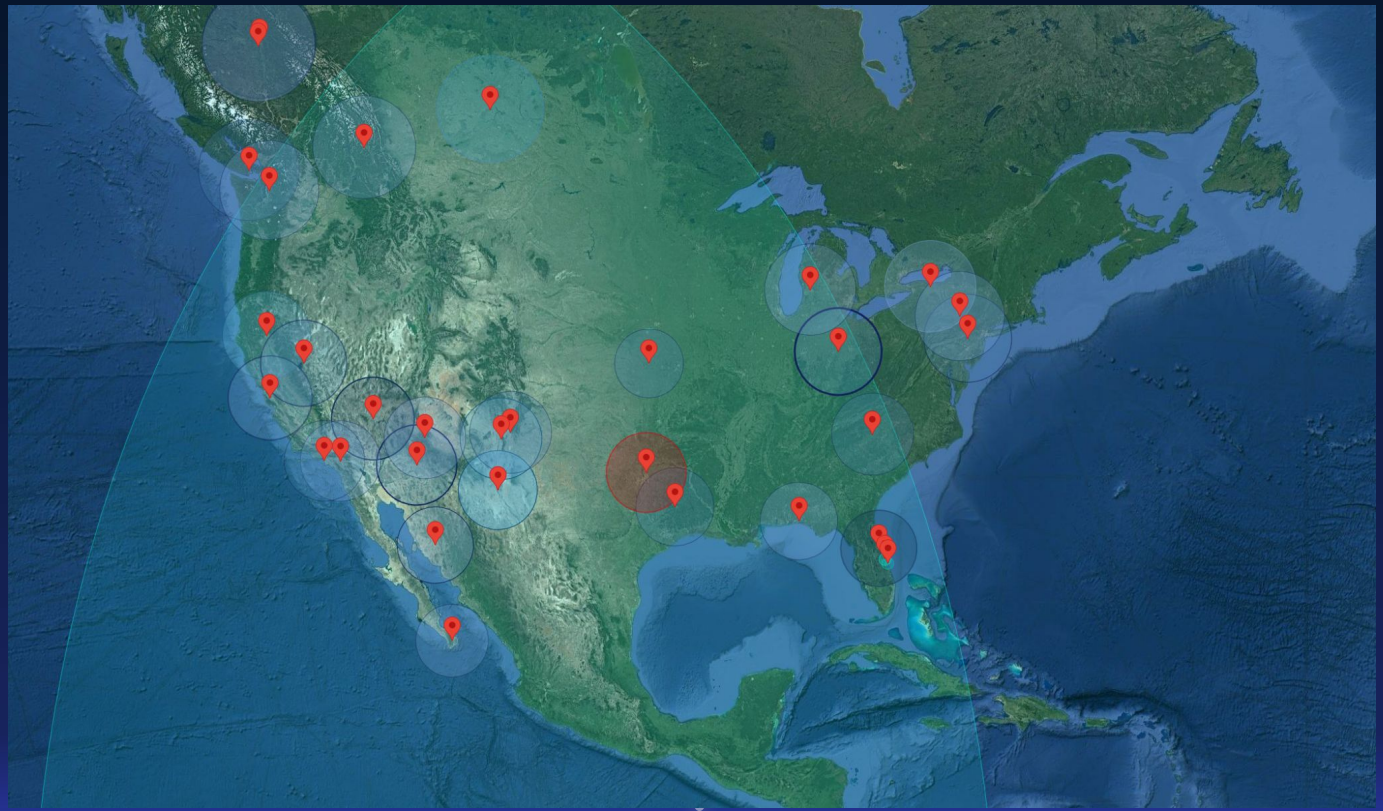
Fireball

A brilliant meteor that may trail bright sparks

Bolide

A large meteor (Fireball) especially one that explodes

Location of Nodes



Goals

- Build software with practical long term use
- Rework and upgrade the software so future devs can get moving faster
- Add documentation
- Improve UX for researchers trying to do the hard stuff

Motivation

- Current system is unstable, crashing sporadically
- UI lacks polish and is cumbersome to use
- Code is large, undocumented, and difficult to learn
- Researchers struggle to get new helpful features

Approach

Onboarding

- Improve onboarding process to minimize the risk of critical errors
- Previously a single mistake could lead to the box being sent back to the research team

Monitoring

- Centralized monitoring system allowing researchers to easily track down and configure boxes remotely.

Architecture

- Modification of the architecture to meet our requirements
- Complete rewrite of the codebase with an emphasis on simplicity, expansion, and modularity.

Algorithms and Tools

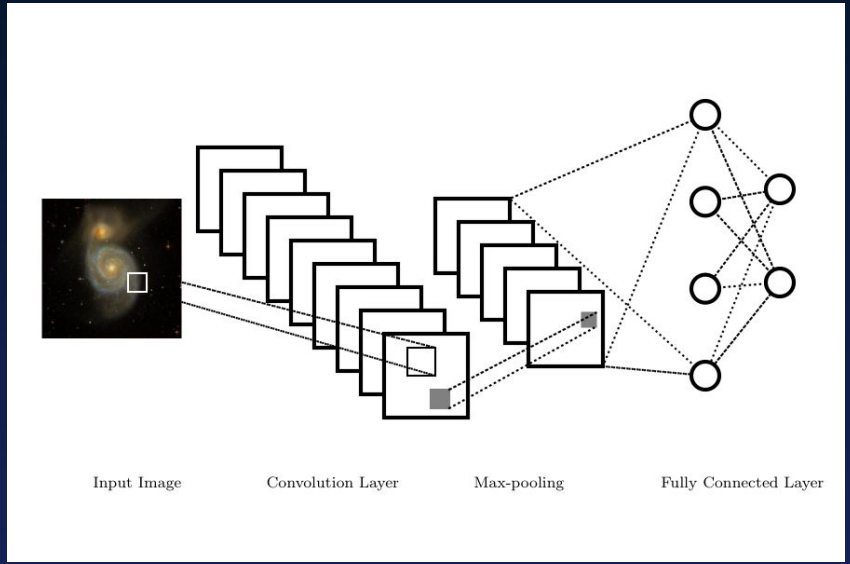
Python (Hardware IO)
OpenCV (Capture/Process video on Pi)
Ffmpeg (Capture/Process video on Pi)
Golang (Backend)
FastAPI (Create API for uploading files to server)
PyTorch (Classification Model for detecting bolides)
Ansible (Declarative configuration)
SQLite (Light database)
Tailscale (Networking)
Netdata (Monitoring)
Tailwind (GUI Styling)
HTMX (GUI Rendering)
GitHub Actions (CI/CD)
SwaggerDocs (Documentation)
Pytest (Unit tests)
Image Processing Algorithms

Novel Features & Functionalities

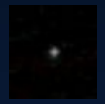
1. Classification
2. Centralized UI
3. IoT Setup

Classification

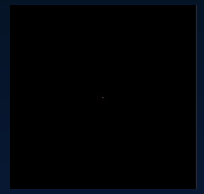
- Currently users must manually sort through events to determine which ones are interesting
- Classification will be used to do this process automatically for the user
- CNN trained on composites of videos captured by different nodes

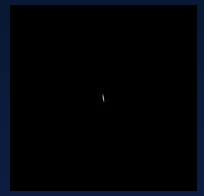


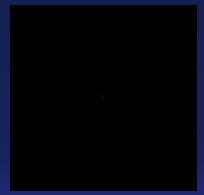
Composite



Object proposer







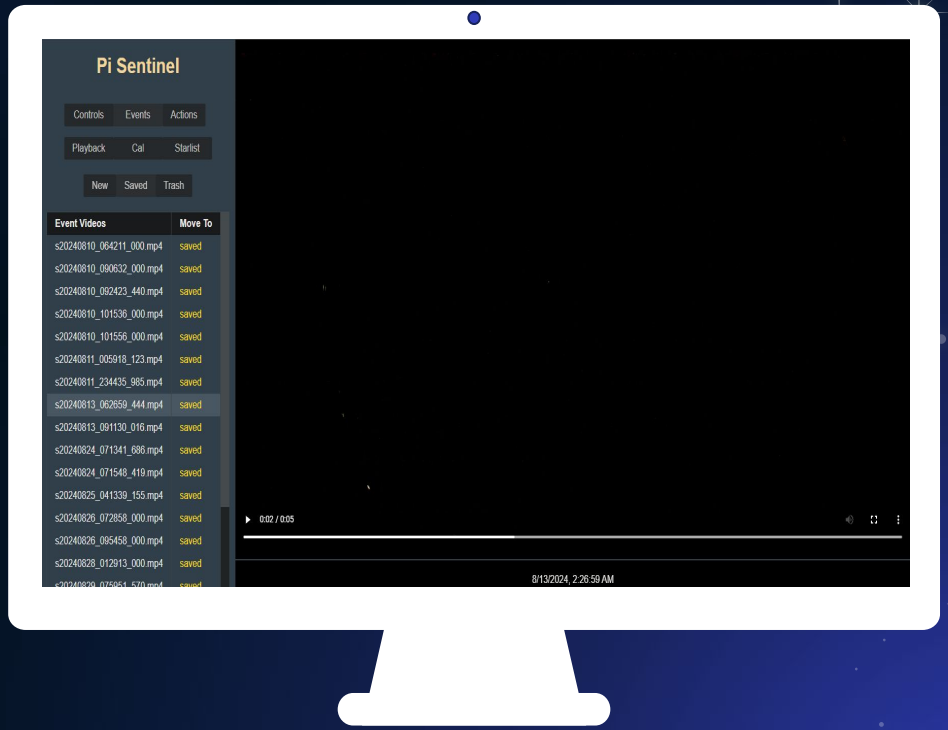
Molding to 512x512

Centralized UI

- Goal is to have a central server that would:
  - Hosts a frontend that controls and views nodes' data
  - Process video data from nodes (reduce workload of Raspberry Pi)

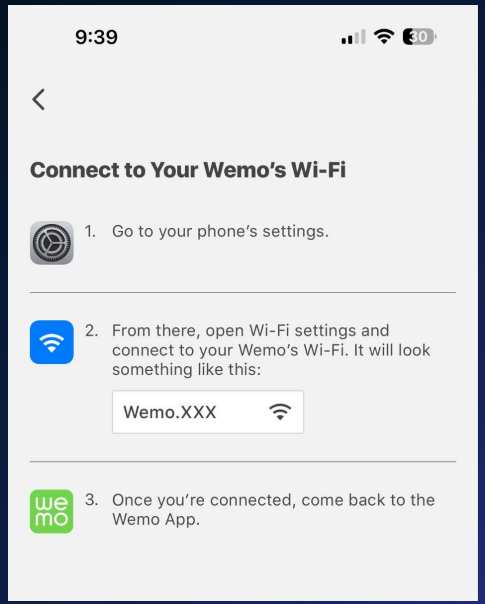
Current UI

UI is running on the local hardware of each node



IoT Style Connectivity

Simplify node setup process



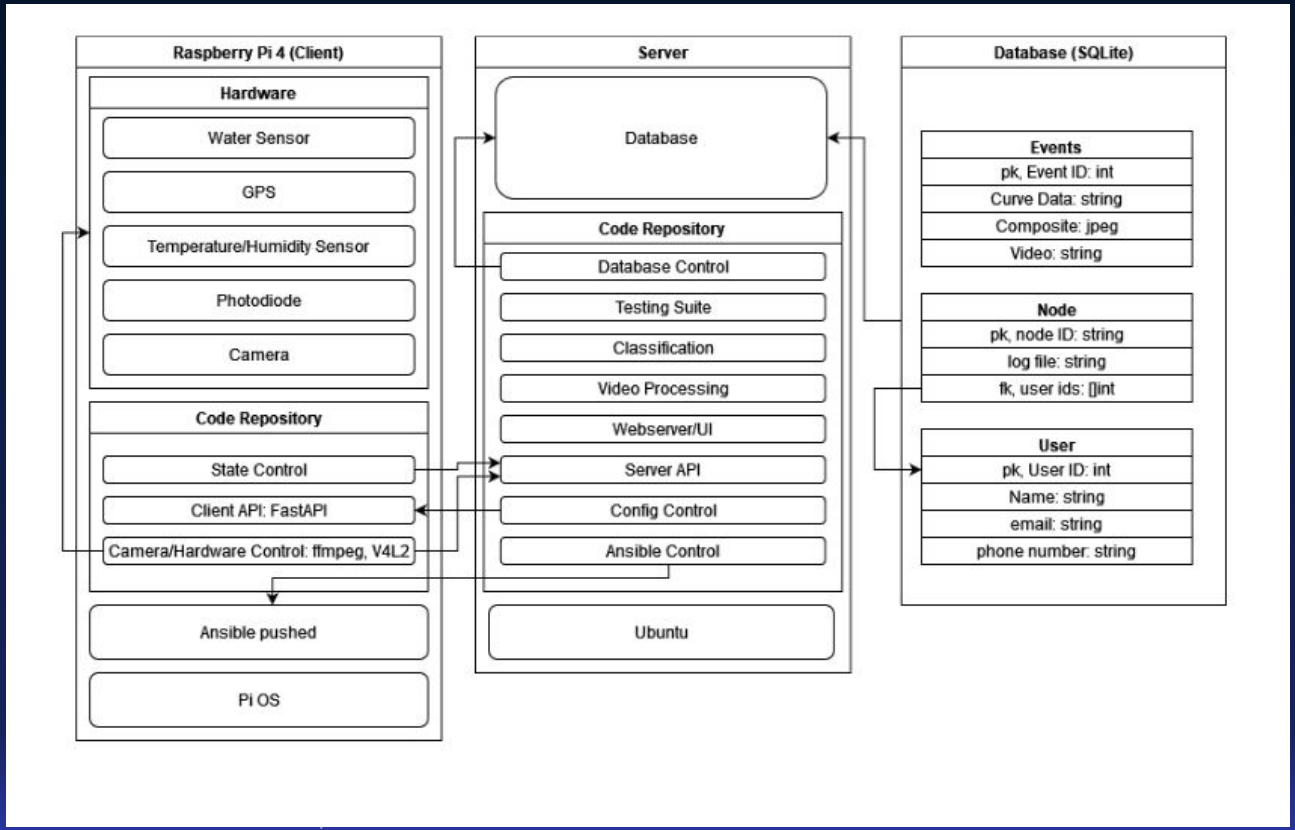
Technical Challenges

Unnecessary Complexity
- Previous system lacks abstraction, documentation, and code modularity

Node package management & versioning
- Bugs & issues arise from unsupported updates to dependencies

Frontend User Experience
- Backend must handle many users making requests

Design


Raspberry Pi 4 (Client)
Hardware
Water Sensor
GPS
Temperature/Humidity Sensor
Photodiode
Camera
Code Repository
State Control
Client API: FastAPI
Camera/Hardware Control: ffmpeg, V4L2
Ansible pushed
Pi OS
Server
Database
Code Repository
Database Control
Testing Suite
Classification
Video Processing
Webserver/UI
Server API
Config Control
Ansible Control
Ubuntu
Database (SQLite)
Events
pk, Event ID: int
Curve Data: string
Composite: jpeg
Video: string
Node
pk, node ID: string
log file: string
fk, user ids: []int
User
pk, User ID: int
Name: string
email: string
phone number: string

Project Evaluation Criteria

1. Time in seconds to sort all incoming events
2. Accuracy of classification model
3. Web page load time, average button response time
4. Classification pipeline runtime, emphasis on object proposer speed
5. How easy (out of 10) is it for the researchers to access all information for a given event and perform their tasks
6. Percent of false alarms on humidity sensors
7. Time taken for a researcher who has not interacted with the UI to access a specific piece of information (light curve of an event, for example)

Progress Summary

Module/Feature	Completion %	To Do
UI	25	Tests, session, backend connection, flesh out menus
Replace current camera code	99	Tests and bugfixes
Server API	99	Tests and bugfixes
Client API	99	Tests and bugfixes
IoT	99	Captive portal, tests and bugfixes
Node setup process	75	Hardware testing process
Classification	99	Explore using transformer architecture, tests and bugfixes
General performance and stability tests	10	Server stress tests, performance tests, UX evaluation

Milestone 4 (Feb 24)

- Initial frontend implementation
- Client hardware interaction
- Client package management and setup process
- Server and client polish

Milestone 5 (Mar 26)

- Frontend testing
- Server and client testing
- UX measurement and general evaluation
- Senior design poster

Milestone 6 (Apr 21)

- UI Polish
- Final UX modification and polish
- Final evaluation
- Final testing
- Demo videos, documentation, user/developer manual

Task Matrix (Milestone 4)

Task	Tyler	Vincent	Jean-Pierre	Charles
Implement UI	10	50	0	40
Polish Server	50	20	30	0
Polish Client	30	20	20	30
Client hardware interaction	50	0	0	50
Create setup process for node	75	0	25	0

Task Discussion

- Implement UI -> Implement a UI and use researcher feedback to enhance UX
- Polish Server -> Bugs and performance issues
- Polish Client -> Video sending needs more testing since it is a core functionality
- Client hardware interaction -> Hardware code needs to be written in python so that researchers find it easier to maintain the code in the future.
- Create Setup Process for Node -> Hardware testing suite

Thanks!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**