Allsky Camera Network for Detecting Bolides

Tyler Turner
Vincent Quintero
Jean-Pierre Derbes
Charles Derbes
Dr. Csaba Palotai

What We are Observing

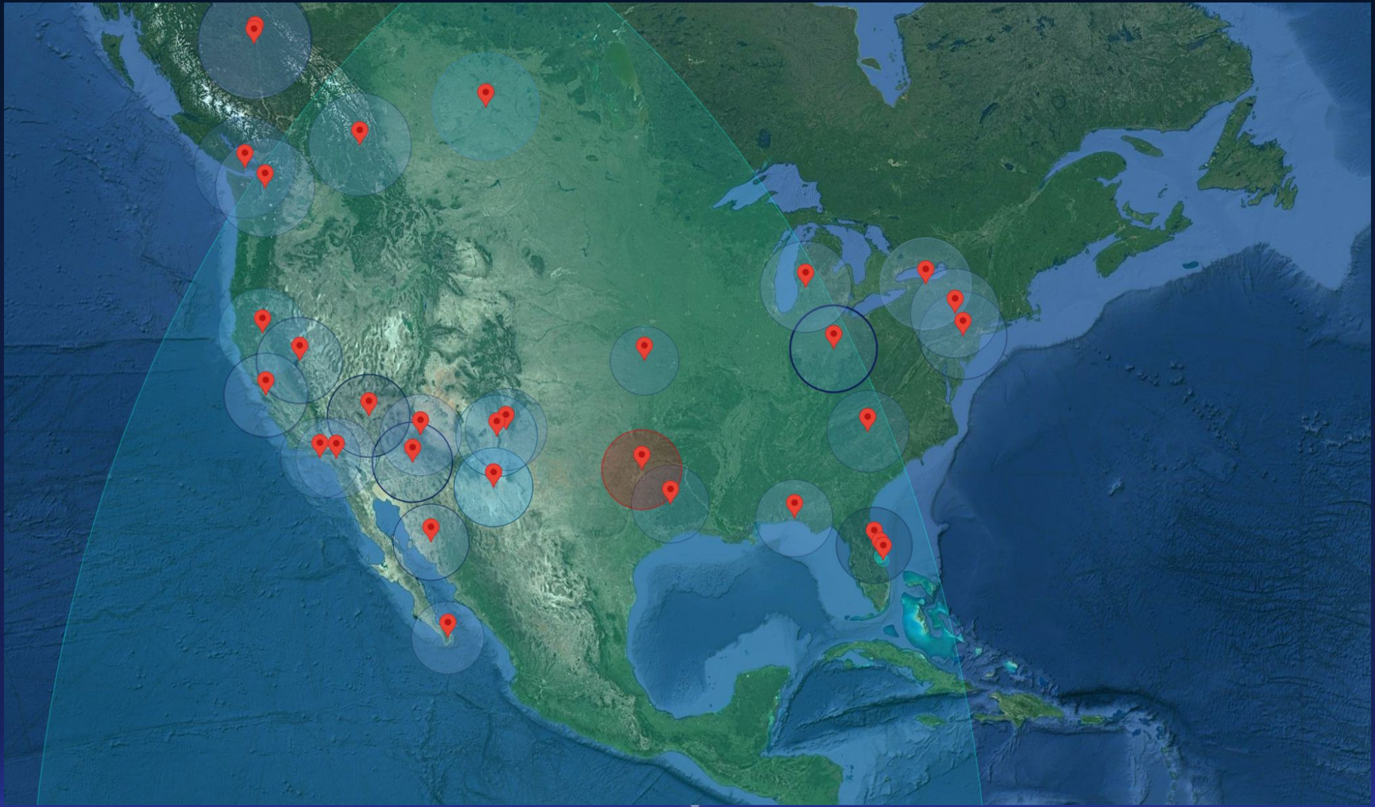
Fireball

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

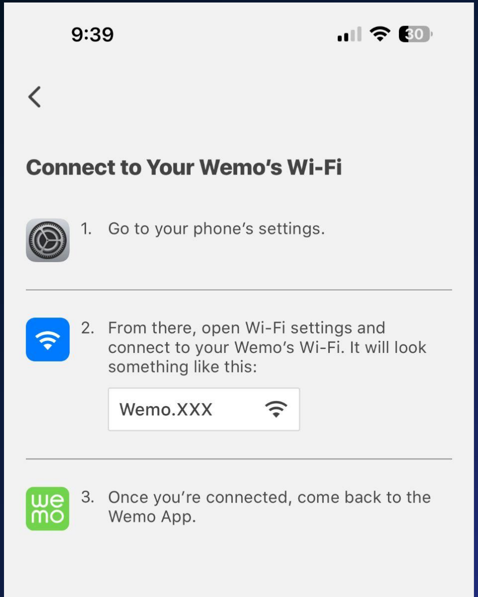
- Modification of the architecture to meet our requirements
- Complete rewrite of the codebase with an emphasis on simplicity, expansion, and modularity.

Novel features/functionalities

1. IoT style setup
2. Improved Box Operation
3. Centralized UI
4. Classification

IoT Style Connectivity

This would simplify the setup process and give users more room to make mistakes



Connect to Your Wemo's Wi-Fi

1. Go to your phone's settings.
2. From there, open Wi-Fi settings and connect to your Wemo's Wi-Fi. It will look something like this:

Wemo.XXX

3. Once you're connected, come back to the Wemo App.

Improved Node Operation

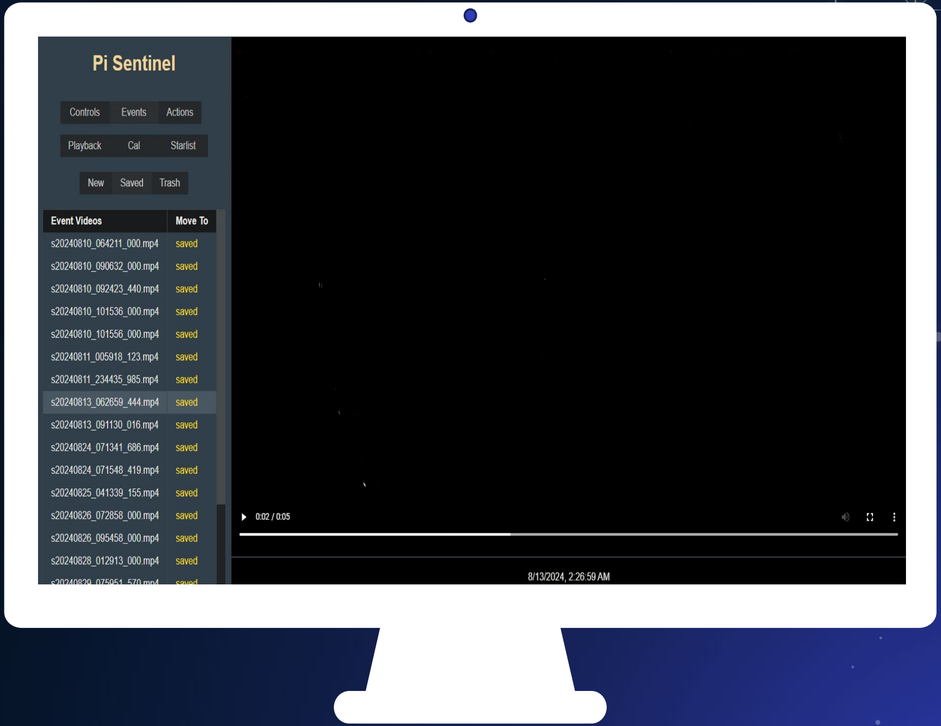
- Nodes will collect data and operate offline until connection is restored
- Nodes will be able to quickly notify owners of critical issues as they occur
- Nodes will be able to download updates remotely

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

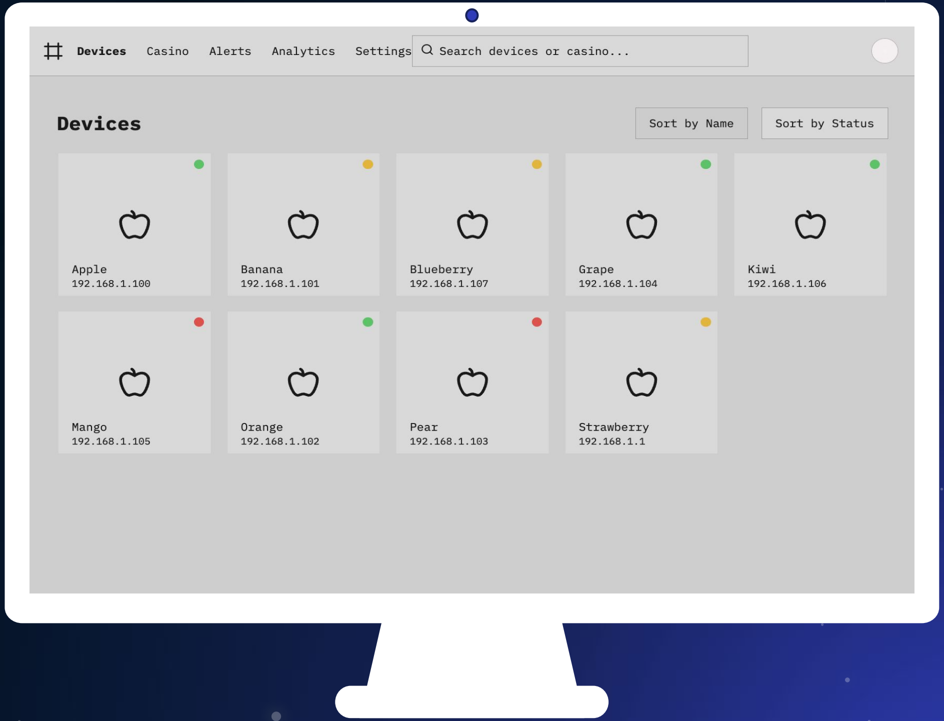


Event Videos	Move To
s20240810_064211_000.mp4	saved
s20240810_090632_000.mp4	saved
s20240810_092423_440.mp4	saved
s20240810_101536_000.mp4	saved
s20240810_101556_000.mp4	saved
s20240811_005918_123.mp4	saved
s20240811_234435_985.mp4	saved
s20240813_062659_444.mp4	saved
s20240813_091130_016.mp4	saved
s20240824_071341_686.mp4	saved
s20240824_071548_419.mp4	saved
s20240825_041339_155.mp4	saved
s20240826_072858_000.mp4	saved
s20240826_095458_000.mp4	saved
s20240828_012913_000.mp4	saved
s20240829_075951_570.mp4	saved

8/13/2024, 2:26:59 AM

Mocked UI

Central UI where user can view all nodes from one place



Devices

Sort by Name

Sort by Status

🟢 Apple 192.168.1.100

🟡 Banana 192.168.1.101

🟡 Blueberry 192.168.1.107

🟢 Grape 192.168.1.104

🟢 Kiwi 192.168.1.106

🔴 Mango 192.168.1.105

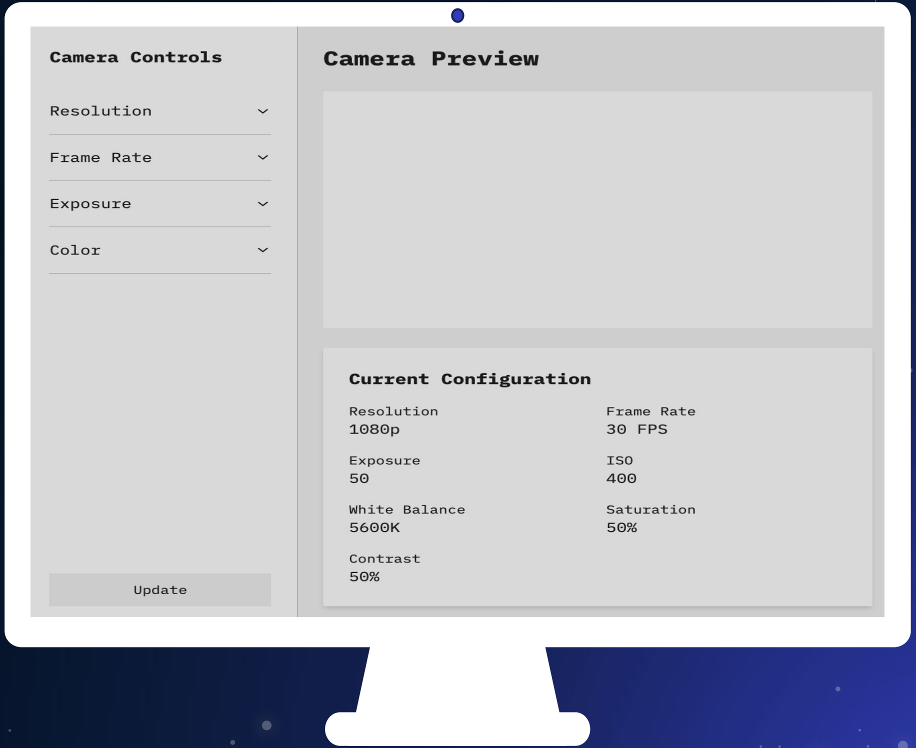
🟢 Orange 192.168.1.102

🔴 Pear 192.168.1.103

🟡 Strawberry 192.168.1.1

Mocked UI

Central UI where user can view all nodes from one place



Current Configuration

Resolution
1080p

Frame Rate
30 FPS

Exposure
50

ISO
400

White Balance
5600K

Saturation
50%

Contrast
50%

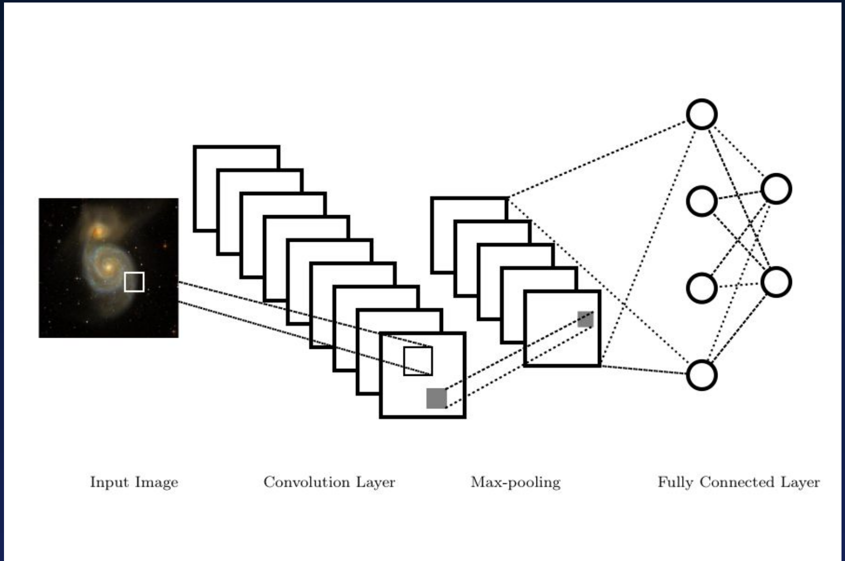
Classification

- 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

- This will require a much larger data set than what we currently have

- Once the data set is augmented we will be training a convolutional neural network to classify the events



Algorithms and Tools

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

Technical Challenges

Codebase has to be rewritten

- Code is currently conglomerated into huge files
- The system is unnecessarily complex and needs to be “untangled” before we proceed
- Program is littered with peculiar implementation choices

Classification

- Dataset needs to be augmented and labeled
- Must be careful not to filter out interesting events

Algorithms and Tools

- Lots of tools means gluing lots of different technologies together

IoT Style Setup

- Might be harder than it seems due to edge cases such as user losing connection to box during setup

Milestone One

- Stabilize system to facilitate testing, data collection, and collecting metrics on the performance of the current infrastructure, whilst the rewrite is occurring.

- Create system architecture diagram

- Storyboard the frontend

- API design

- Design CLI for image processing pipeline

Milestone Two

- Implement API
- Implement CLI
- Show that event can be detected and sent to server

Milestone Three

- Classification
- Frontend MVP

Task Matrix (Milestone 1)

Task	Tyler	Vincent	Jean-Pierre	Charles
Diagnose current issues	Fix stop camera error	Frontend	Fix state (consolidate into single source of truth)	Fix unknown camera control errors (particularly when changing exposure modes)
Create system architecture diagram	All contribute	All contribute	All contribute	All contribute
Storyboard the frontend	Brainstorm	Create	Brainstorm	Brainstorm
API design	All contribute	All contribute	All contribute	All contribute
Design CLI for image processing pipeline	Create	Brainstorm	Brainstorm	Brainstorm

Thanks!

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