Allsky Camera Network for Detecting Bolides

Members
Tyler Turner, tturner2021@my.fit.edu
Vincent Quintero, vquintero2021@my.fit.edu
Jean-Pierre Derbes, jderbes2021@my.fit.edu
Charles Derbes, cderbes2021@my.fit.edu
Faculty Advisor & Client
Csaba Palotai, APSS, cpalotai@fit.edu

Client Meetings:
Meetings will be in person on an as-needed basis. This could mean twice a day or once a month.

Goal and Motivation:

Motivation:

- Current system is unreliable; certain controls are conducive to crashes and or unexpected behavior.
- We believe the software side of the project lacks polish and UX.
- We know the software side of the project is of poor quality and is unmaintainable
- We know the project lacks documentation

Goals:

- To build a product for a customer that has practical long term use.
- To rebuild the software part of the project, making it more maintainable and modular for future engineers.
- To significantly enhance UX.

Approach:

- Improved the onboarding process for new boxes to minimize the risk of critical errors during installation. New users can set up boxes more confidently, reducing the likelihood of breaking mistakes. Previously, any setup error required sending the box back to the research team, wasting valuable time and money. Users will be able to more smoothly and easily set up boxes on their own with this update, removing the worry of a simple mistake leading to another post office trip.
- A more streamlined box monitoring process. A dedicated domain enables users to easily track and configure all deployed boxes whenever needed. Detailed event and system logs track the status of each Pi (box) deployed so issues can be solved remotely.
- A project completely rebuilt with a strong emphasis on simplicity, expansion, and modularity. This redesign ensures that future developers can easily onboard and contribute towards new project needs. Users will also benefit from a more simplified and

expandable code base as updates and changes can be shipped out at a faster rate.

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 Algos

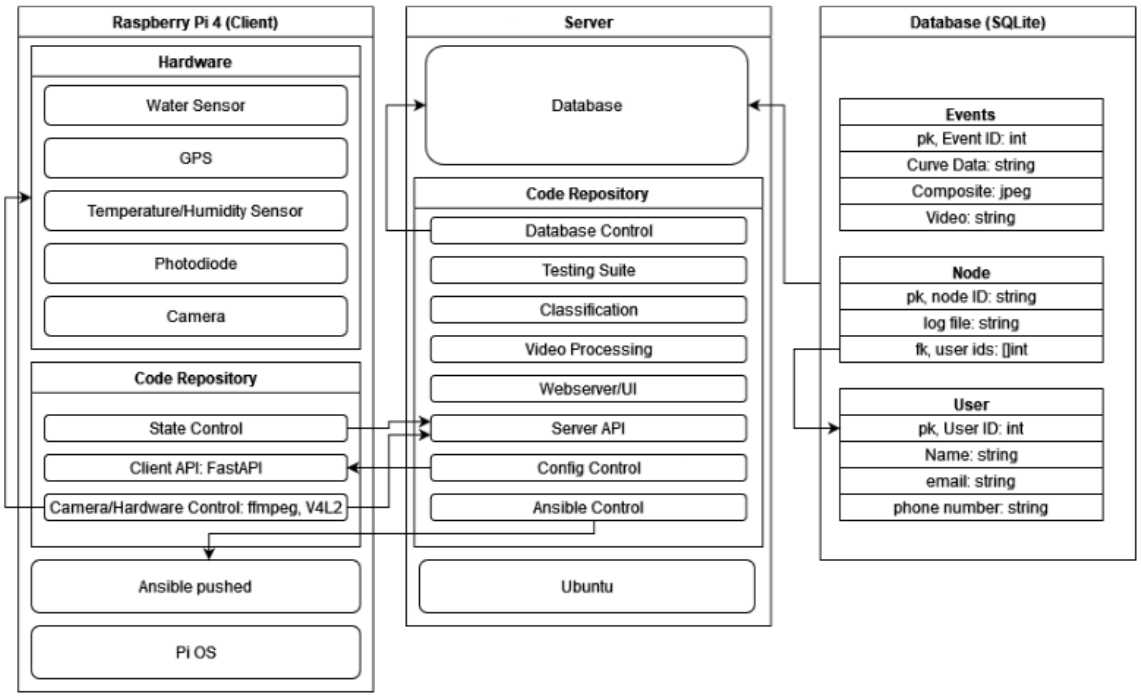
Novel features/functionalities:

- Classification - Right now a person has to do video triage to decide what is or is not an interesting event. A classification model would classify videos as either interesting (e.g. a meteor or comet) or uninteresting (e.g. a fly flew over the camera). This would reduce the amount of manual checks the user does.
- Centralized GUI - Currently each box has its own frontend and processes its own data. We would like to centralize this to a server to reduce workload on the Pi. Users will be able to access all events and all boxes in a single location rather than logging on to each Pi separately.
- IoT - Currently the user has to connect a keyboard, mouse, and monitor to the box to connect to the internet through PiOS. An IoT style setup would solve this cumbersome problem by eliminating the need for external hardware to connect the box to the internet. Users will be able to access a web page that allows them to connect their box to the internet and manage the status of their box.

Technical Challenges:

- The system is unnecessarily complex - The previous system lacks abstraction, documentation, and modularity.
- Package management for nodes - Many of the past bugs and issues in the software have arisen due to unsupported updates to certain dependencies.
- Frontend user experience - Backend services must be able to handle queries on the frontend that may occur from different sessions.

Design:



Evaluation:

- Time in seconds to sort all incoming events
- Accuracy of classification model
- Web page load time, average button response time
- Classification pipeline runtime, particularly the proposer
- How easy (out of 10) is it for the researchers to access all of the necessary information for a given event and perform their necessary tasks.
- Percent of false alarms on humidity sensors
- How long does it take a researcher who has never seen the UI to access a specific piece of information (light curve of a given event at a given date, for example)

Progress Summary

Module/feature	Completion %	To do
UI	25	Tests, sessions, backend connection, a couple more menus to flesh out
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 transformer, tests and bugfixes
General performance and stability tests	10	Stress tests (reliability), 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 video, documentation, user/developer manual

Task matrix for 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 1: Goal is to have a working frontend that we can present to the researchers for testing. We will use their feedback to enhance UX. The centralized UI should make accessing information much easier for them and significantly speed up their workflow.

Task 2: The server will most likely have bugs and/or performance issues and we will most likely want to change a few things around. We will need to stress test the server with all of the nodes sending videos at once to the server. The server also needs a set of pipelines that are triggered after receiving the videos.

Task 3: Non camera related C++ code needs some looking at. Project structure might need to be revised and more testing around state needs to be done. Video sending will need some more testing as it is the core purpose of the node.

Task 4: Communication between the Pi and the hardware needs more testing and needs to be more organized. This code also needs to be written exclusively in python in order to make it easier for researchers to maintain in the future. Ensuring that the software works properly will allow researchers to immediately diagnose issues as hardware related.

Task 5: Continuation of last milestone, node setup just needs a hardware testing suite. This would test all of the hardware making sure nothing is broken. The tests would be checking if the Pi is properly connected to all hardware components, and then testing the transmission and receiving of data between the Pi and the hardware component. For example it would test to make sure that the camera is connected and is capturing video correctly.

Approval from Faculty Advisor:

"I have discussed with the team and approved this project plan. I will evaluate the progress and assign a grade for each of the three milestones."
Signature: _____ Date: _____