



# Allsky Camera Network for Detecting Bolides

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

# Task Matrix (Milestone 2)

Task	Completion	Tyler	Vincent	Jean-Pierre	Charles
Continuously fix endless stream of issues	90%	20%	20%	40%	20%
Add logs for easier diagnosis of issues	100%	0%	90%	0%	10%
Replace current C++ camera code	70%	30%	50%	20%	0%
Implement Server API	90%	70%	5%	25%	0%
Implement Client API	80%	10%	0%	50%	40%
Begin writing CLI	0%	0%	0%	0%	0%
IoT Style Setup	90%	20%	0%	30%	50%

# Task Discussion

Continuously fix endless stream of issues -> Notification system bug and time zone bug fixed

Add logs for easier diagnosis of issues - > Logs are chunked and queryable

Replace current C++ camera code -> Switched to augmentation and background subtraction

Implement server API -> Wrote and tested routes for server API

Implement client API -> Wrote and tested routes for client API

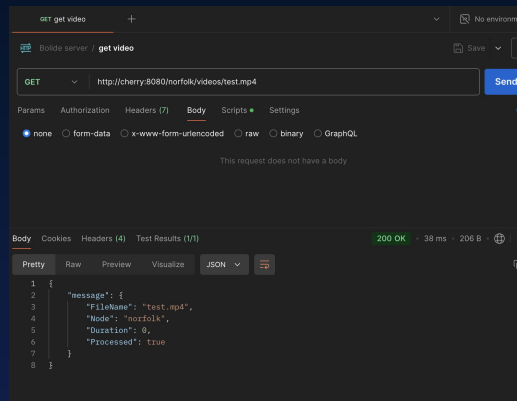
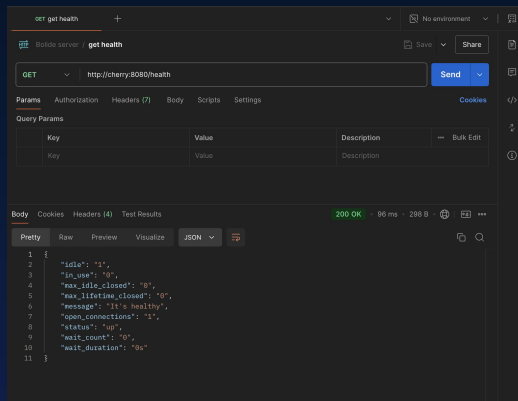
Begin writing CLI -> On hold until client is implemented and we have more events

IoT style setup -> Components are built but they still need to be glued together

# Client + Server Demo



# Server API



A screenshot of a web browser showing a database interface. The browser has three tabs: 'routes.go', 'database.go', and 'dev.db'. The 'dev.db' tab is active, showing a table named 'recordinas'. The table has five columns: 'file\_n...', 'node', 'duration', and 'processed'. The table contains one row of data: 'test.mp4', 'norfolk', 0, and 1. The interface includes a search bar, a filter button, and a table view toggle.

file_n...	node	duration	processed
test.mp4	norfolk	0	1

# Client API

GET get status + No environment

Bolide node (client) / get status Save Share

GET http://127.0.0.1:8000/status Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (4) Test Results 200 OK · 178 ms · 144 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": "online"
3 }
```

Postbot Runner Start Proxy Cookies Vault Trash

GET get config + No environment

Bolide node (client) / get config Save Share

GET http://127.0.0.1:8000/config Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (4) Test Results 200 OK · 147 ms · 426 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "archivePath": "dev/arc",
3   "devName": null,
4   "eventsPerHour": null,
5   "frameRate": null,
6   "gpsLatitude": null,
7   "gpsLongitude": null,
8   "gpsTimeOffset": null,
9   "noiseThreshold": null,
10  "numNew": null,
11  "numSaved": null,
12  "numTrashed": null,
13  "running": null,
14  "startTime": null,
15  "stopTime": null,
16  "sunThreshold": null,
17  "zenithAmplitude": null
18 }
```

Postbot Runner Start Proxy Cookies Vault Trash

main.py 2. M config.json M X

client > app > config.json > ...

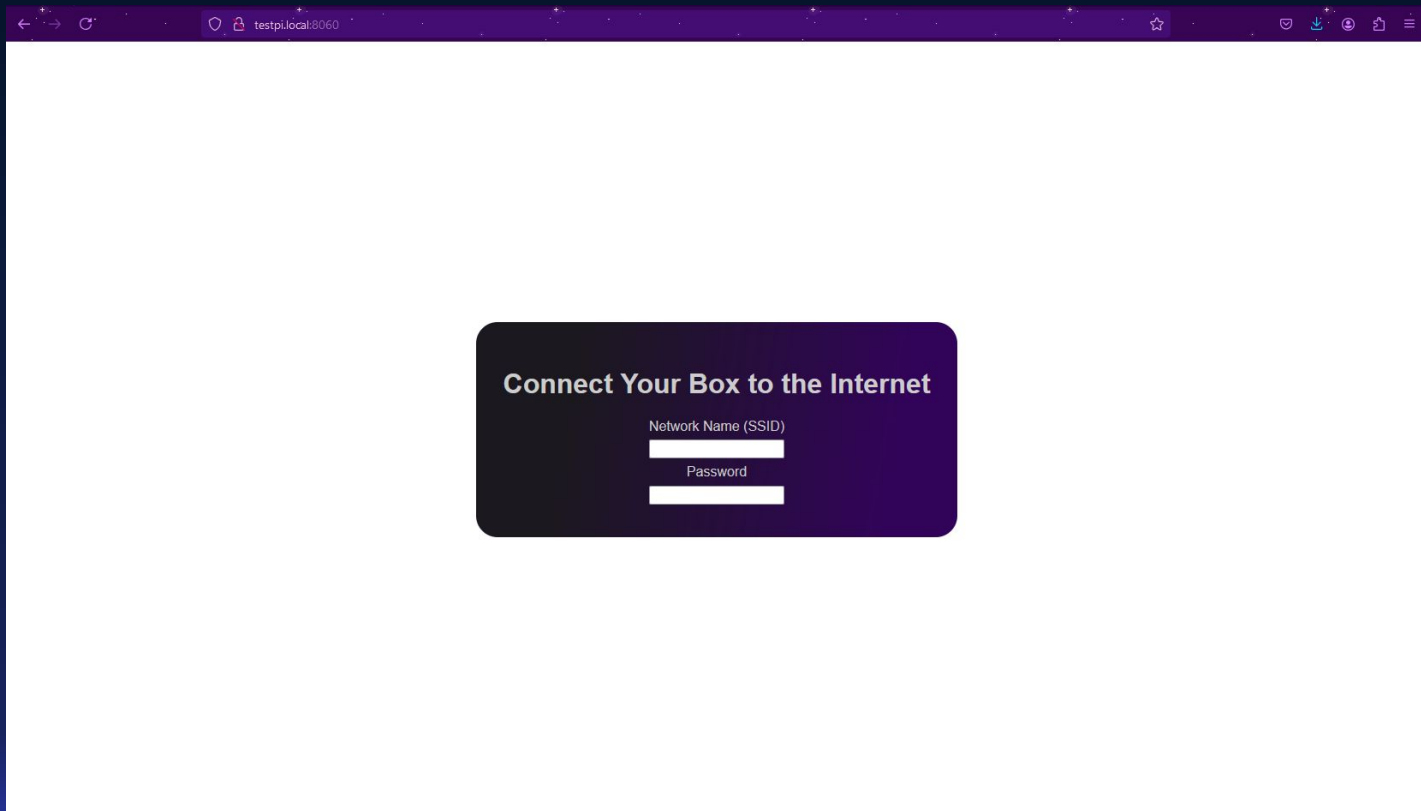
```
1 {
2   "archivePath": "dev/arc",
3   "devName": null,
4   "eventsPerHour": null,
5   "frameRate": null,
6   "gpsLatitude": null,
7   "gpsLongitude": null,
8   "gpsTimeOffset": null,
9   "noiseThreshold": null,
10  "numNew": null,
11  "numSaved": null,
12  "numTrashed": null,
13  "running": null,
14  "startTime": null,
15  "stopTime": null,
16  "sunThreshold": null,
17  "zenithAmplitude": null
18 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
python3 - client
bolides@cherry:~/code/fl-tech-bolides/client$ python3 main.py
bolides@cherry:~/code/fl-tech-bolides/client$ cd ../
bolides@cherry:~/code/fl-tech-bolides/client$ uvicorn app.main:app --reload
Usage: uvicorn [OPTIONS] APP
Try 'uvicorn -h' for help.

Error: No such option: -r
bolides@cherry:~/code/fl-tech-bolides/client$ uvicorn app.main:app --reload
INFO: Will watch for changes in these directories: [/home/bolides/code/fl-tech-bolides/client]
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [945998] using WatchFiles
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:8076 - "GET /status HTTP/1.1" 200 OK
INFO: 127.0.0.1:154174 - "GET /config HTTP/1.1" 200 OK
INFO: 127.0.0.1:152824 - "GET /config HTTP/1.1" 200 OK
```

# IoT



A screenshot of a web browser window with a purple header bar. The address bar shows "testpi.local:8060". The main content area is white and contains a dark purple rounded rectangle with the text "Connect Your Box to the Internet". Below this text are two input fields: "Network Name (SSID)" and "Password".

testpi.local:8060

**Connect Your Box to the Internet**

Network Name (SSID)

Password

# Contribution of Each Member

## Tyler Turner

- Server API endpoint implementation
- Video processing queue implementation
- Notification service implementation

## Vincent Quintero

- Rewrote event detection software
- Improved image processing
- Improved event detection algorithm
- Event logging for recordings and processing



# Contribution of Each Member

## Jean-Pierre Derbes

- Fixed None/null errors in node state fields
- Implementation of server and client API endpoint handlers
- Database design and implementation

## Charles Derbes

- IoT style setup
- Client API endpoint implementation
- Logging for node/client processes

# Task Matrix (Milestone 3)

Task	Tyler	Vincent	Jean-Pierre	Charles
Replace current C++ camera code	10%	35%	45%	10%
Implement Server API	50%	0%	50%	0%
Implement Client API	20%	0%	20%	60%
Begin writing CLI	30%	10%	50%	10%
IoT style setup	20%	0%	10%	70%
Classification	0%	33%	33%	34%
Start writing UI	20%	70%	0%	10%
Create setup process for node	75%	0%	25%	0%

# Task Discussion

## Task 1: Replace current C++ camera code

- Event triggering + recording: C++ -> Python
- Detection algorithm and image processing improved
- Client side recording chunking

## Task 2: Implement Server API

- Missing functionality on a few endpoints
- Need to stress test server with expected video sizes

## Task 3: Implement Client API

- One endpoint handler needs implementation
- Need to test client api integration with server

# Task Discussion

## Task 4: Begin writing CLI

- Enough data to start CLI
- CLI exposes internal system functionality
  - Generate composites
  - Classify composites + classification report

## Task 5: IoT Style Setup

- FastAPI + HTML + CSS + JS
- Edge cases need to be solved
- Communication with server API

## Task 6: Classification

- Binary Classifier
- Need to augment data through transforms and simulations
- Tune CNN

# Task Discussion

## Task 7: UI

- On hold until client and server infrastructure is implemented and well tested
- UI mockups will be modified based on feedback from past presentations

## Task 8: Node Setup Process

- Workflow to set up and test newly built node
  - Install OS + required packages
- Ansible to manage node updates
- Easier to diagnose node issues

**Thanks!**