Allsky Camera Network for Detecting Bolides
Milestone 6

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

Progress Matrix for Milestone 6:

Task	Completion	To Do	Tyler	Vincent	Jean-Pierre	Charles
Client Hardware Interaction	100%	N/A	50%	0%	25%	25%
Orbit, trajectory, velocity, mass (of bolide)	Dropped	Not a requirement of a project (sort of a “nice to have”), dropped because it was too complicated to get accurate results	0%	0%	20%	80%
Client connectivity logic	100%	N/A	10%	0%	20%	70%
Poster, User/Developer Manual, Demo Video, Documentation	100%	N/A	25%	25%	25%	25%
Finish UI	100%	N/A	20%	80%	0%	0%

Full system server and client tests	80%	Integration tests (leaving it outside for a long period of time and just making sure the system works)	40%	30%	20%	10%
Evaluation	100%	N/A	25%	25%	25%	25%

Discussion of each accomplished task for the current Milestone:

- Task 1: Water sensor code added. The need for a radiometer was eliminated as the light curve analysis is done using the event. Camera calibration and starmap code are not needed as we are not triangulating the position of the bolide. GPS code is functional and used to display node location in the UI.
- Task 2: This task seemed simple on the surface however it turned out to be challenging for a variety of reasons. Some boxes may have different cameras (or all of the boxes will get upgraded cameras) which makes camera undistortion challenging. Even if the undistortion were to be accurate the program would then have to figure out an error "cone" where the bolide could be, projected from the camera (this requires the starmap and the gps in order to orient the camera, it's a pyramid/cone because the actual position of the bolide is somewhere inside the "ball" of light visible in the event).Within the cone there is an error pyramid due to the limited resolution of the image (the cone is essentially a collection of pyramids, one per pixel). The intersection of the two pyramid sets from separate events that occurred at the same time is the area where the bolide could be. If the error is high enough two separate events that occur at the same time might be considered the same on one frame, then disassociate on another frame. Thus the only way to know if two events contain the same bolide is through a calculation that might give the wrong impression. It becomes a complete mess if there are multiple bolides at the same time, as then they have to be separated from each other and somehow the program has to know there are multiple bolides in the first place.
- Task 3: Hostname issues solved by implementing a middleware, events that fail to send to the server are put in a retry queue. This makes sure that events will always be sent to the server in case of connectivity issues. State variable is used to prevent the auto connector from running whilst the user is on the internet login page. Logic changed slightly to give the node a chance to auto connect and to avoid locking the node into a hotspot state because of simple wifi restarts or small disconnections.
- Task 4: Poster created with a minimalist style. Demo video created showcasing typical use of the app. Complete documentation of the project linked as user/developer manual. Documentation written and developed in a clean UI with clear separation between technical and user documentation.
- Task 5: The centralized UI is completely done and polished. Elements were changed around for better consistency, visibility, accessibility, and user experience. Pages were optimized to have faster load speeds. Implemented the connection between frontend and HCS. Bug fixes regarding displaying bolide confidence and displaying a page instead of a json error when there are no events in the analyze queue.

- Task 6: Integration tests done to make sure UI works as intended, nodes could do with a little more full night tests. Postman used to test all endpoints. Stress tested HCS by putting it through 400 events at once. Continuously tested nodes' cameras to ensure that video quality and exposure were correct.
- Task 7: Several researchers were shown the system (they had never seen it before), and given several tasks to perform. We recorded how long it took for them to do these tasks. They are happy with the simplicity and ease of use of the system compared to the previous system.

Discussion of contribution of each team member to the current Milestone:

- Tyler Turner:
  - Fixed docker bugs
  - Conceived HCS idea
  - Icarus and Heimdall deployment
  - Fixed numerous Heimdall backend bugs
  - Wrote documentation for Heimdall
  - Faster SQL queries
  - Event trash can and delete feature
  - Node deletion on Heimdall backend
  - Created node setup ansible playbook
  - Added Heimdall status check
- Vincent Quintero:
  - Finalize UI integration
  - Project documentation
  - Implement researcher requested modifications
  - Node Retry/Failure architecture
  - Testing
  - Designed and created showcase poster
- Jean-Pierre Derbes:
  - Retry system if node events fail to send to server
  - More reliable node camera snapshots
  - Fixed laggy nighttime recordings due to auto exposure taking too long to capture light
  - User and technical documentation for Icarus (code that goes on nodes)
  - Created some scripts for easier debugging of Icarus system
  - Designed and implemented the Heimdall Classification Service (bolide classification)
  - Helped with poster
- Charles Derbes:
  - Fix HCS bugs
  - Designed and implemented light curve algorithm
  - Fix event going to trash instead of event tab
  - Fix errors when no events are left in analyze queue
  - Improved proposer
  - Documentation
  - HTTP middleware for Icarus

- Integration testing of HCS and UI
- Helped with poster

Lessons Learned

- Spend more time thinking about design: Certain components such as IoT and frontend/backend interaction had to be remade due to a realization later in development that their design was hard to work around.
- Better time estimation: We found that we underestimated the time and effort it would take us to complete many of our project related tasks.
- Limit use of libraries: Libraries can make life easier however they reduce visibility into how your code works at a low level and they should be avoided for simple tasks.
- Document everything continuously: Documenting and commenting code and systems as you are writing them is much easier and better than going back after a few weeks or months and trying to remember all the nitty gritty details.

Dates of meetings with Client during the current milestone:

see Faculty Advisor Meeting Dates below

Client feedback on the current milestone:

see Faculty Advisor Feedback below

Dates of meetings with Faculty Advisor during the current milestone:

- March 26
- April 9

Faculty Advisor feedback on each task for the current Milestone:

- Task 1:

- Task 2:

- Task 3:

- Task 4:

- Task 5:

- Task 6:

- Task 7:

Faculty Advisor Signature: _____ Date: _____