|  |  |  |  |
| --- | --- | --- | --- |
| Artifact ID:  DT-001 | Artifact Title:  Decision Tracking | |  |
| Revision:  1.1 | Revision Date:  23 OCT 2019 | |
| Prepared by:  Nick Merriman | | Checked by:  Jesse Krage |
| Purpose:  The purpose of this artifact is to provide an in-depth and extensive record of our decision-making process throughout the concept development stage. | | |

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision: | Revised by: | Checked by: | Date: |
| 1.0 | Joe Hansen | Jesse Krage | 18 OCT 2019 |
| 1.1 | Nick Merriman | Jesse Krage | 23 OCT 2019 |

# References

|  |  |  |
| --- | --- | --- |
| Artifact ID: | Revision: | Title: |
| CD-001 | 1.0 | Brainstorm Notes |
| CD-004 | 1.0 | Reactive Tracking Concept Definition |
| CODE-001 | 1.0 | Code for Tracking Simulation |
| NOTES-005 | 1.0 |  |

# Requirement Breakdown

**Feasibility Analysis I: 10/9/2019**

**Activity Description:** After brainstorming initial concept ideas as a team, concepts were evaluated to determine which ones were feasible and were worth further consideration. The full list of brainstormed concepts can be seen in artifact CD-001.

Processor Ideas Eliminated:

* Onboard Chromebook – takes up too much space and would be difficult to weatherproof. Costs outweigh any potential benefits.

Positioner Ideas Eliminated

* Octopus Powered – removed for obvious reasons.
* Human Powered – removed for obvious reasons.
* Ropes and pulleys – would be overly complicated and less precise than off the shelf positioner.
* Self-leveling tripod – overly complicated, outside scope of project.
* Put positioner on track – overly complicated.
* Dish big enough to cover entire area of operation – outside scope of project
* Electronically controlled beam direction – overly expensive and complex, outside of project scope.

GUI Ideas Eliminated

* Alexa controlled – impractical, likely security risks, suggested somewhat jokingly in the first place.
* Controls the drone – outside scope of project, likely many security/safety risks.

After eliminating many of the ideas we were left with a much smaller and more manageable list of concepts to consider. Several concepts include ideas that can be joined together and will potentially be pursued in future prototyping efforts.

For the processor itself we have decided to use a Raspberry Pi for the onboard computer, see DJ-001 for the decision justification.

For the positioner we will use a pan and tilt positioner selected by IMSAR. See DJ-001 for the decision justification. However, some of the concepts generated may still be implemented later. For prototyping purposes an inexpensive pan-tilt-zoom camera that communicates with Pelco D protocol will be purchased. This will allow us to test our control software without the entire system which won’t be available until around January 2020.

**Initial Prototyping Concepts: 10/22/2019**

**Activity Description:** As a team we realized that we had somewhat missed the mark in some of our initial concept development activities. We cut back the scope of our concept generation and focused more on methods used in the tracking method, system architecture, and GUI interface. The tracking method is going to be the main portion that determines the success of our project, so this is where we wanted to focus our attention. Some new ideas were brainstormed and a plan for prototyping was developed.

GUI Prototyping Plan

We have decided to pursue two primary paths for the GUI. We will develop concepts for both a desktop version and a mobile version. We will first create a graphical layout (e.g. in PowerPoint, no actual coding done) to help improve the idea. After the basic design is determined, we will then begin coding and making a functional GUI.

Tracking Method Prototyping Plan

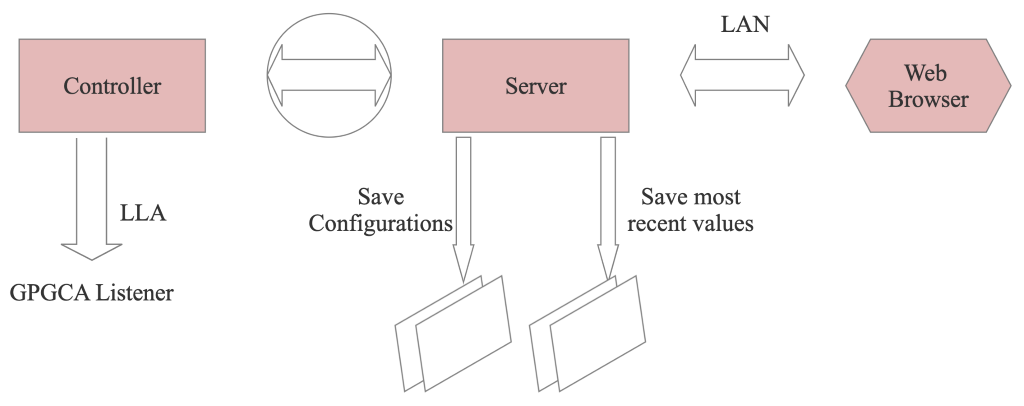
The team has decided to pursue both a reactive and a predictive tracking system. Both will be developed and tested with the prototyping camera that has been purchased. This will allow us to determine which method will provide better results and know how to move forward and spend most of our time.

Tracking Method ideas added:

* Point at current drone location (Reactive Tracking)
* Point at predicted location based on drone’s velocity (Predictive Tracking)

System Architecture Prototyping Plan

The team will seek to prove that all necessary components of the system will be able to communicate with each other with low enough latency to track the drone’s radar. This will include ensuring communication between the browser-based UI and the tracking control system. This will also include ensuring that the tracking control system will be able to communicate back to the browser-based UI. It will be noted that the purpose of the small test positioner is merely to confirm that we can send PELCO-D commands via a browser-based UI.



We divided ourselves into teams to approach the above prototyping plans.

**Prototyping Report: 11/4/2019**

**Activity Description:**

GUI

Feedback from meeting with the campus user experience club helped influence the design of the GUI. In addition, email correspondence with Mark helped remove unnecessary features and make the interface as simple as possible.

Tracking Simulation

The Python Tracking simulation showed that even with conservative estimates, the reactive tracking method was exceptional at maintaining the simulation aircraft in the field of view (FOV). See CODE-001. We decided to proceed with the reactive concept because of its effectiveness in the simulation. See CD-004 (or whatever I call it. It’ll be the Reactive Tracking Concept Definition)

System Architecture

NGINX is a well-documented server solution that is lightweight enough to run quickly on a Raspberry Pi. We found a library that allows for communication between our server and our controller called FastCGI. We learned that the control software will simply need to output an angle for the tilt and pan positioner system to move to. We will not need to develop equations of motion based on timing and the positioner’s angular velocity. We implemented a “single producer single consumer” queue from the boost library to ensure that the controller receives all information sent from the server without missing commands. This technique resolves timing issues and makes the communication thread safe. This method will be used to communicate from the controller to the server.

**Prototype Refinement: 11/8/2019**

**Activity Description**

We met with Mark Catanzaro at the IMSAR facilities to report on our progress and receive feedback. See NOTES-005

GUI

We clarified some minor points related to the inputs and functionality of the GUI. We also learned that a mobile version would not be necessary.

Architecture

Mark was thrilled with our server setup. His only question was related to the latency of communication. Our initial research showed that the latency for an NGINX server with 100 concurrent users was approximately 100ms. He was impressed with this. His largest concern was that the latency would be on the order of 1 second. We also discussed the idea of being able to host the browser-based GUI on a Local Area Network (LAN). Mark admitted that while this would be convenient in some instances, would not be necessary because of practical signal strength constraints. We determined to identify this as a stretch goal.

Requirements

Example page 25

Important to include

Subheadings: feasibility analysis rounds (discussion summary to narrow down to concepts we want to test), prototyping rounds (changes to design made up in prototypes what works and what doesn't), reference to final scoring matrix to justify the selection of a prototype.

Feasibility analysis

Note discarded ideas

Note ideas with promise

Post-prototype analysis

Note observations and conclusions

Note changes to prototype