NORTHERN ARIZONA UNIVERSITY

ASTRAEA

NAVY PRECISION OPTICAL INTERFEROMETER

Requirements Document

Team Members	*Clients*	*Mentor*
Adam Schilperoort	Jim Clark (NRL)	Isaac Shaffer
Brandon Horner	Teznie Pugh (Lowell)	
Michael Partridge		
Peter Kurtz		
Trey Tangeman		

December 2, 2019

Contents

1 Introduction

Improving astrometry, or the measurement of position, motion, and magnitude of stars, is a never-ending goal of astronomers and astrophysicists around the world. By building devices which can achieve higher precision astrometrics, scientists can improve upon previous models of stars and develop a more accurate representation of the cosmos. Breakthroughs in this area could lead to a better understanding of the universe, or potentially redefine the laws of physics altogether.

In the twentieth century, improvements in astrometry enabled physicists such as Albert Einstein and Stephen Hawking to make revolutionary contributions to the field of astrophysics. Following in these early innovator's footsteps, modern day astrophysicists are researching the properties of dark matter, modelling black holes, and searching for exoplanets. Improvements in observation technology could drastically accelerate research in these areas. Not only could this technology benefit scientists, but also the general public — for example, accurate star charts provide precise positioning for GPS satellites, which helps improve navigation around the planet.

From exploring foreign worlds to answering humanity's biggest questions, the impact of astrometry cannot be understated. Unfortunately, in the past few decades modern astrometry reached a point of diminishing returns, making it exponentially more expensive to improve on the existing technology. As a result, an alternative to the conventional single-aperture telescope design needed to be developed. Navy Precision Optical Interferometer (NPOI) tackled this design problem by developing an optical telescope system which utilizes multiple telescopes receiving images simultaneously, combining their images to achieve a much larger telescope. This design is capable of providing the highest angular resolution on Earth of any optical design at a significantly lower cost than equivalent conventional designs.

NPOI, an astronomical long-baseline optical interferometer has been in operation on Anderson Mesa, outside of Flagstaff, Arizona, since 1994. The 437 meter baseline array has a unique capacity for detecting and determining motions and orbits of binary systems, which is its current research focus. The site's unique precision is achieved through a high-tech mirror control system which simultaneously gathers light from three to six telescopes separated by some distance, and combines their respective images for analysis. NPOI's partners are Lowell Observatory, United States Naval Observatory (USNO), and Navy Research Laboratory (NRL). NRL's goal was to develop technology able to measure the positioning of stars better than any other instrument on Earth to allow precise navigation from stars in the event GPS systems fail. USNO became involved to generate star charts for astronomy and astrophysic analysis. Lowell joined due to the unique ability to study binary systems.

2 Problem Statement

Currently at NPOI, there are several spots on the array where a Siderostat station can be placed. A Siderostat station consists of a Siderostat mirror and a smaller Narrow Angle Tracker (NAT) mirror. During observations, star light is reflected off the Siderostat to the NAT then into the pipes on the array to be combined with starlight from other Siderostat stations. The Siderostat mirrors are moved by stepper motors, motors that move in discrete steps, and the NAT mirrors are pushed with piezoelectric actuators. The mirrors at each of these stations is controlled by a rack of computers. These computer racks act as the intermediary between an observer and a Siderostat station. When the observer wants to move the mirrors to look at a different object in the night sky, the observer has to send a command to a specific Siderostat station where a computer known as SidCon will pick up the signal over an optical fiber network. The command is then passed to another computer, PowerCon, where computations are performed to convert the command sent by the observer to a small voltage. The small voltage is then passed to either a Parker microstepper or an Ultravolt piezoelectric actuator. The Parker microstepper amplifies the signal which is then sent to a stepper motor on a Siderostat. The Ultravolt piezoelectric actuator amplifies the signal from 0 to 5 volts to 1000 to 0 volts in a negative correspondence. The signal is then sent to a piezoelectric actuator attached to the NAT.

There are several major issues plaguing NPOI's Siderostat stations currently. These problems fall into two different categories: hardware and software.

The current hardware at each Siderostat station is between 20 to 30 years old. This antiquated hardware is also custom-made, and the individuals who made the hardware are no longer available at NPOI. One of the side effects of the custom hardware is that there is poor insulation, and some of the hardware components need to be isolated away from other hardware to avoid cross communication between electronic signals, which can cause loss of precision of the system as the mirrors move more or less and are not actually at the location the observer specified. Over the past few years there has been an increase in hardware failure rates. Whenever a piece of the custom hardware failed, it would be replaced with a prefabricated backup. As of the end of 2019, the last prefabricated backup has been used and the next hardware failure at a Siderostat station cannot be fixed and observations will cease.

On the software front, each operation Siderostat station has a computer rack. Each rack has it's own operating system, which is some Linux distribution, some racks have one type while others have their own, there is no standardized version or distribution of Linux on every rack. Although there are different versions of Linux on the racks, each rack contains its own copy of the software that handles receiving the signal from the observer and the operations to convert the signal from the observer to the required output for the Parker microstepper or the Ultravolt piezoelectric actuator. This software is bloated with unnecessary lines of code, causing the software to be heavy and redundant. This becomes cumbersome on the system when any updates are made to the software, as any updates have to be done physically at each station which costs time and money.

3 Solution Vision

Given the multitude of problems with the hardware systems at NPOI, a solution has been envisioned to solve them. Since a lot of the problems stem from the fact that the hardware is custom made, outdated, and is redundant across all stations, the many computers at each station that handle observer commands are going to be replaced with a single, centralized computer built with modern consumer hardware, or what is going by the name BrainCon.

The job of BrainCon is to host a server over the network at NPOI from an environmentally secure location, and then handle observer commands to operate the Siderostat and NAT at each station – all from a single location. BrainCon will achieve the operation of the Siderostat and NAT by communicating with a Raspberry Pi, or a microcontroller, at the respective station, which will have the software necessary to control the micro steppers on the Siderostat and the piezoelectric actuators on the NAT.

BrainCon aims to solve the problems at NPOI by replacing the many-to-many software system with a one-to-many system and replacing the custom motherboards and proprietary PCI cards that effectively control the Siderostat stepper motors and piezoelectric actuators on the NAT. The new software system will alleviate software redundancy and allow for easier troubleshooting and maintenance of the system. On the hardware end, BrainCon and the Raspberry Pi will remove the failing custom hardware and reduce the points of failure, which allows for NPOI to continue operation and become a maintainable site in the long-term.

In terms of specifics, BrainCon will aim to have the following functionality:

- Communicate back and forth with the observer over a server, providing feedback and receiving commands.
- Communicate back and forth with the Raspberry Pi at each respective station over a server, forwarding commands and receiving feedback.
- Provide a graphical user interface, which will have the system status and interface for providing commands.

However, BrainCon is only the central point of command in the system, so, in order to get a full purview of the system, the functionality of the Raspberry Pi needs to be described as well:

- Listen for commands over the network.

- Drive the stepper motors on the Siderostat by sending signals to the Parker micro stepping driver.
- Drive the piezoelectric actuators on the NAT by outputting appropriate voltages, which then get stepped up. 0-5 volt output corresponds to 1000-0 volt through the piezoelectric actuator control.
- Create a feedback loop between the Raspberry Pi and the limit switches, sending the feedback to BrainCon.

4 Project Requirements

In this section, the functional, performance, and environmental constraints will be discussed. These requirements will outline the details of how the envisioned solution will look and the kind of functionality it will provide.

Since the envisioned solution is effectively split into two discrete pieces of hardware and software, the requirements for BrainCon and the Raspberry Pi will be discussed separately. BrainCon handles the communication and direction of commands, so the software implications will be discussed there. On the other hand, the technology on the Raspberry Pi is very low-level, so the hardware specifics for driving the hardware at each station will be discussed.

4.1 BrainCon

Given that BrainCon is an off-the-shelf computer that handles observer commands and communicates with the Raspberry Pi at each observing station, these are the functional requirements that were determined to be necessary and how they expand out into other requirements:

1. Network with observer
    a. TCP server hosted on BrainCon
        i. Network wrapper API
            1. Persistent connections
            2. Modular design
        ii. Multi-threaded network
            1. Job system
            2. Thread pool
            3. Thread safety
    b. Graphical user interface

        i. Graphical user interface API
            1. Dear ImGui
        ii. User interface design
            1. Configuration
            2. Individual station control

2. Send commands to Raspberry Pi
    a. BrainCon job creation
    b. Packet handling
        i. Parse destination and command
        ii. Data Processing
            1. Handle conversion to stepper motor counts
        iii. Handle connection to Raspberry Pi
            1. Convert destination to IP address

3. Handle feedback loops
    a. Designated thread listening for feedback
    b. Handle the feedback
        i. Error detection
            1. Issue commands to stop failing system
        ii. Keep track of metrics reported by feedback loops
        iii. Produce system status reports on user interface

Now that the baseline functional requirements for BrainCon have been established, more detail can be shown on how these functions might be implemented and how they will be interfaced with by the user.

4.1.1 Network with observer and AlignCon

This is one of the major functions that BrainCon provides. Networking with the observer from BrainCon will define how a user actually interacts with the system. As a result, this has caused this requirement to expand out into a few other requirements:

1. TCP server hosted on BrainCon

The type of server that will be used to network not just with the observer, but with all the stations as well, is the TCP protocol. This entails that the BrainCon software will be fairly low-level networking code, as no framework is being used.

A TCP server will be created by using the built-in GNU/Linux network API in C/C++ . In particular, a network wrapper API will be programmed, which allows for ease of use and modularity in the code. The modularity means it will not matter whether the software is networking with a Raspberry Pi at a station or the observer, the implementation stays the same.

The network wrapper API will simply take an IP address and a port and create and bind a socket to listen and accept requests on. This is where the importance of TCP presents itself. Having the ability to keep a connection between, for example, a station and BrainCon means packets can freely be sent back and forth – effectively creating a feedback loop.

On top of having this network wrapper API, the TCP server on BrainCon will be multi-threaded. BrainCon is going to use a thread pool system for multi-threading. Essentially, there will be a job queue that the main thread – the thread which will receive packets on the TCP socket – will enqueue a job on based off the type of packet received. Then, on the job queue, there will be a number of threads, or a thread pool, waiting to dequeue a job and handle it. This system will allow for standardization of handling and processing data, while doing so on multiple threads and in a safe way, making networking between many stations and the observer trivial.

2. Graphical user interface

Given that BrainCon will have this networking API, which is multi-threaded and allows for networking between all the stations and the observer, the missing link is the method of interaction between BrainCon and the observer. In order to provide a user interface that is easy-to-use and does not require programming experience, a graphical user interface will be created.

The first requirement for creating a graphical user interface is deciding on a graphical user interface API. The BrainCon graphical user interface will use a lightweight, immediate-mode API called Dear ImGui. Dear ImGui provides the necessary ability to draw graphical user interface, but does so without needing a massive framework or IDE to compile.

Using Dear ImGui, BrainCon will need to be able to provide the interface necessary for the observer to submit commands and see system status. The interface will have a fairly simple design, where there will be a configuration button that allows the user to add, delete, and edit stations in a configuration file hosted by BrainCon. There will be a new window created for each station.

For each window, which corresponds to a station, the user can then select a type of command to submit to it, which will be listed as a grid of buttons with the name of each command. The BrainCon software will then receive a command, and process it and redirect it to the appropriate station. Some examples of command are: stow Siderostat, move EL+/- (elevation positive or negative), move AZ+/- (move azimuth positive/negative).

In terms of status reporting, the graphical user interface will report a few metrics, which it calculated based on feedback from the Raspberry Pi. To begin with, each station window will have a light, which toggles between red and green, that will indicate whether a connection can be established to the desired stations selected. Next, the user interface will have few text boxes that list the following metrics for each station:

- Stepper motor counts
- Siderostat position
- Station status

The stepper motor count will be a simple integer which will be received from the Raspberry Pi. The Siderostat position is a more complex metric, which will require BrainCon to compute the position of the Siderostat based on feedback from the Raspberry Pi. The status of the station will be a simple message, such as “OK,” or an error message reporting that there is an issue.

4.1.2 Send commands to Raspberry Pi

The next piece to the BrainCon software is the communication interface between it and the Raspberry Pi. Since the network interface was outlined above, this section will only discuss the additional functionality required to send commands to Raspberry Pi. Here is how that expands:

1. BrainCon job creation

The main hurdle for achieving communication between BrainCon and the Raspberry Pi depends on the processing of packets received on BrainCon. Everytime BrainCon obtains a packet, it will need to parse the packet to determine the type of command and determine if any data processing needs to occur before sending out the command. The workflow of packet handling will be discussed in this section.

After a packet is received by BrainCon over the network, it will first push the packet as a job onto the job queue. The job will contain the packet data and the destination, or which station the commands want to be redirected to. A single worker thread will then pull the job off the job queue, where the worker thread will then proceed to handle the packet.

2. Packet handling

The first step in packet handling is parsing the packet. Parsing the packet involves separating the packet header from the packet data. For the most part, the packet header will be ignored, as BrainCon is not using a predefined protocol, such as HTTP, but instead using its own packet structure. Inside the packet data, the type of command will be parsed out, along with its parameters. For example, a command that states to move the Siderostat five counts in elevation would translate to a packet type of elevation move and a packet parameter of five counts.

Once the packet is parsed, the worker thread will determine if any computations are needed based on the command type. For example, the worker thread would need to translate an elevation move of five counts to a number of counts for the stepper motor, as well as pack which motor the counts needs to be done on in the output packet. With this determined, the worker thread will take the destination passed to it in the job queue, and create an outgoing packet. The outgoing packet will contain at least two fields: the number of motors to step the stepper motor and which axis the counts correspond to.

Finally, the worker thread will send the outgoing packet to the respective Raspberry Pi by establishing a connection to the IP address associated with the destination, which will be stored inside the BrainCon configuration file. The Raspberry Pi will receive the packet and continue from there. See the Raspberry Pi section to see how it will handle commands from BrainCon.

4.1.3 Handle feedback loops

The final high-level functional requirement for BrainCon is receiving feedback from the Raspberry Pi and other systems such as AlignCon, which provides the feedback for the angular position of the NAT. This will involve listening for feedback from the devices and handling it. This functional requirement expands straightforwardly:

1. Designated thread listening for feedback

Rather than having each thread that is handling a command listen for feedback after sending a packet, a designated thread will listen on a socket on BrainCon. Having a single thread assigned to this task allows for the Raspberry Pis, for example, to send feedback as desired, since there is not a one-to-one correlation between sending a command and waiting for feedback.

This functionality will be implemented by using the network API that was outlined in the first functional requirement for BrainCon. As it is designed modularly, BrainCon simply needs to configure the IP addresses that correspond to each feedback loop in the system. BrainCon can then simply listen for packets, push the packet data as a job to be handled by the other worker threads, and immediately go back to listening for feedback. Using this workflow, BrainCon can maintain a state of the system and make more accurate assumptions.

2. Handle the feedback

After the feedback is pushed onto the job queue, a worker thread will pull it off and handle the packet. Inside a feedback packet, it will simply have the data for the feedback it is providing and a type field indicating the kind of feedback. For example, the Raspberry Pis will have a feedback loop that reports the step count.

Using the feedback data, BrainCon can maintain metrics about the system. BrainCon will use these metrics to aid in issuing commands to the stations, as well as providing messages or errors to the observer through the user interface. Each station will have a series of toggling lights or text fields indicating types of feedback in the user interface, as outlined in the first functional requirement on the graphical user interface.

Based on the feedback metrics, BrainCon can determine if the system is failing or if a soft limit has been reached. A soft limit is simply a software limit that BrainCon maintains, which helps keep the observer aware that the Siderostat or NAT are reaching their hard limits. If a failure is reported, which might result from a hard limit being reached, BrainCon will need to issue the commands necessary to stop the failing system. Error reporting will be handled in the graphical user interface as stated above.

4.2 Raspberry Pi

Given that the Raspberry Pi is a microcontroller that handles input commands from BrainCon over the network, communicates with piezo controllers for the NAT and stepper controllers for the Sid, while responding to feedback from limit switches, there are a few functional requirements that were determined to be necessary:

1. Network with BrainCon
    a. Send feedback to BrainCon
        i. Send packets containing relevant data
    b. Receive inputs from BrainCon
        i. BrainCon command processing
            1. Packet parsing
            2. Data processing
2. Control mirror positioning
    a. Drive stepper motor
        i. Send pulses to Parker microstepper
        ii. Receive feedback from limit switches
    b. Drive piezoelectric actuator
        i. Send command to Ultravolt Piezoelectric Actuator Control
        ii. Receive feedback from limit switches

3. Track position of NAT and Siderostat
    a. Motor step counts
    b. Piezoelectric actuator voltage
    c. Log limit switches
4. Be electronically insulated from EM interference

Given this outline of the requirements for the Raspberry Pi, each subsection may be elaborated on to fully detail the necessary functionality of each part of the system.

4.2.1 Network with BrainCon

Because BrainCon is handling all major calculations of the system, distributing observer commands to individual stations, the Raspberry Pi will need to interface with BrainCon through the network to operate the mirrors and provide feedback information to the system. Networking with BrainCon includes the following:

1. Send feedback to BrainCon

Because the Raspberry Pi's will be distributed individually to each station, feedback for mirror positioning necessary for BrainCon software will have to be handled at each station and reported back to BrainCon over the network. When BrainCon issues a command to the Pi to move a stepper motor or a piezoelectric actuator, the Pi will need to perform the action and update local variables corresponding to mirror positioning, continually reporting the updated variables to BrainCon. Limit switches on the Sid and Nat, acting as absolute positioning feedback, will need to trigger an event on the Pi which submits a packet to BrainCon informing its software a limit switch has been reached. The software on BrainCon will then be able to calibrate positioning more precisely.

2. Receive inputs from BrainCon

Individual Pi's act as liaisons for BrainCon, allowing for individualized control of stations. To allow for remote control of each station's piezoelectric actuators and stepper motors, the Pi will need to receive packets from BrainCon, parse the data provided, and respond to commands given. Inputs received from BrainCon may tell the Pi to push a stepper motor a certain number of counts, apply a certain voltage to the piezoelectric actuators, or halt.

4.2.2 Control mirror positioning

Previously it has been mentioned that the Raspberry Pi is responsible for controlling mirror positioning for the Sid and NAT. Because the Raspberry Pi doesn't directly interface with the stepper controls on the Sid or the Piezoelectric actuators on the Nat, to change the Siderostat positioning, the Pi must interface with a 'middle man' which receives signals from the Pi. These are outlined with the following requirements:

1. Drive stepper motor

To move the Siderostat, the Pi must move one or both of the Siderostat stepper motors. To accomplish this, the Raspberry Pi must interface with a Parker Microstepping Drive for each axis of control. Each motor step corresponds to a 5V, 200ns pulse applied to the Parker device, so for the Pi to control one Siderostat mirror, it must be able to apply these pulses to two different Parker Microstepping Drives.

2. Drive piezoelectric actuator

To move the NAT, the Pi must interface with two Ultravolt A-series Piezoelectric Actuator Control devices. Each piezoelectric actuator (one for each axis) accepts a voltage from 0-1000, so upon initialization of the NAT, each piezo should be initialized to 500 Volts for maximum range. Because the Ultravolt device is responsible for stepping up the voltage to the acceptable range, the Pi only needs to apply a voltage in the 0-5V range, 2.5V corresponding to 500 Volts on each piezo. Thus, to change the position of the NAT according to BrainCon commands, the Raspberry Pi simply needs to increase or decrease the applied DC voltage in a 0-5V range.

Additionally, the Pi will need to interface with and receive signals from optical limit switches. These limit switches are placed on the Sid and NAT and will trigger when either mirror reaches a limit. While the stepper motor control will be highly precise, the mirror has mechanical issues which prevent the stepper control from being able to reliably predict the positioning of the mirror, making sensing limits incredibly important. Detecting absolute positioning of either mirror and reporting this information back to BrainCon is essential for knowing the different mechanical requirements of each station. Reaching a limit will also inform BrainCon to decelerate the mirror to prevent from damaging hardware, so it's essential that the Raspberry Pi can detect these switches and report the information to BrainCon.

4.2.3 Track positioning of NAT and Siderostat

Utilizing the limit switches, motor counts for Siderostat stepper motors, and voltages for the NAT's piezoelectric actuators, it's possible to approximate and track the positioning of the NAT and Sid. While the positioning data may be somewhat unreliable, it's important to report back to the observer GUI what the approximate position is. To accomplish this, each stepper motor pulse, applied voltage to a piezoelectric actuator, and activated limit switch must be logged and reported over the network to BrainCon, which then reports this information back to the observer GUI. Through successful tracking of Sid and NAT behavior, 'expected' vs 'observed' behavior of the mirrors can be a valuable asset for detecting mechanical problems in the system, allowing the site maintenance staff to monitor and fix issues.

4.2.4 Electronic insulation

One of the major issues of the previous system is that it behaved strangely due to electronic crosstalk. Thus, the client has expressed interest in insulating the Pi from the rest of the devices at each station so that crosstalk is virtually impossible. This requirement means not only placing the device in an individualized case, but physically separating it from other devices.

In summary, the Raspberry Pi will need to network with BrainCon, control mirror Positioning, track positioning of NAT and Sid, and be electronically insulated. Interfacing with BrainCon requires receiving and interpreting packets sent from BrainCon and reporting data back. Pushing mirrors requires applying a signal to a Parker Microstepping Drive and an Ultravolt Piezoelectric Actuator Control, while tracking positioning requires monitoring limit switches and logging any mirror commands. Electronic insulation can be accomplished through an insulated, physically separated box.

5 Performance

This section will explain the quantitative components associated with some functional requirements listed above. These requirements were determined by the client and the project team in order to maintain the current precision of the interferometer.

5.1 BrainCon

5.1.1 Network with observer and AlignCon

1. Quick processing of incoming packets

    a. BrainCon will be receiving observer commands, AlignCon input (60Hz), and will do data processing (about 30Hz) is desired for each of these subjects for a total of 90Hz.

5.1.2 Send commands to Raspberry Pi

1. Quick processing of outgoing packets

    a. BrainCon must be able to process packets at about 900Hz to not bottleneck the Raspberry Pis at the six currently available station which can process outputs at around 150Hz (see Control Mirror Positioning 3.a. below).

5.1.3 Handle feedback loops

1. Quick feed back loops

    a. In order to effectively use the feedback loop currently in the system, BrainCon must be able to process at an additional 30Hz in order to keep the expediency of the feedback loop.

To sum up the above performance requirements, BrainCon will be required to process instructions at a rate of at least 1.02KHz. However, most modern processors exceed this value by two to four billion times so this is very reasonable.

5.2 Raspberry Pi

5.2.1 Network with BrainCon

1. Quickly send feedback to BrainCon
   a. About 30Hz is desired for processing feedback from the limit switches and sending it to BrainCon.
2. Quickly receive inputs from BrainCon
   a. About 30Hz is desired for processing inputs from BrainCon.

5.2.2 Control mirror positioning

1. Accurate electric pulses
   a. The Raspberry Pi must be able to send 5V, 200ns pulses to the Parker Microstepping Drive which will in turn drive the stepper motors.
   b. The Pi must be able to send 0-5V pulses to the Ultravolt Piezoelectric Actuator Control which will in turn drive the piezoelectric actuators on the NAT
2. Quick mirror operation
   a. Each Pi must be able to process outputs at about 120Hz. At least 60Hz is desired to send two commands to the Parker Microstepping Drive and another 60Hz for sending the two commands to the Ultravolt Piezoelectric Actuator Control associated with the Pi for a total of 120Hz.
3. Precise mirror control
   a. The Raspberry Pi solution must maintain a precision of 59 arcseconds when driving the Sid.
   b. A precision of 10 arcseconds must be maintained when controlling the NAT.
4. Quick feedback from limit switches
   a. In order to prevent the Raspberry Pi from bottlenecking the system, at least 30Hz of processing must be dedicated to receiving the feedback from the limit switches.

To sum up the performance requirements of the Raspberry Pi, around 210Hz is required. Modern Raspberry Pis can handle around 500 million times this speed. The Pi must also be able to output two separate 5V signals to the Parker Microstepping Drive and Ultravolt Piezoelectric Actuator Control.

6 Environmental Requirements

BrainCon must interface with existing hardware. This includes the local network and the observer computer. BrainCon must receive commands from the observer computer, perform the necessary calculations, and pass them on to appropriate Raspberry Pi at each station. It receives and sends these commands over the local network, via Ethernet.

The Raspberry Pi must interface with existing hardware. This includes the local network, limit switch wiring, and sending appropriate output to the stepper and piezo motors. The Raspberry Pi must communicate with BrainCon over the local network, via Ethernet. It must receive feedback from the limit switches; this input is determined by the limit switches already in place, thus the Raspberry Pi must have the limit switches directly wired to its pins. The stepper and piezo motors accept their own uniquely formatted input. The format of this input is determined by the stepper and piezo motors currently in use. The Raspberry Pi must be able to send movement command input to these motors in the exact format they expect. The Raspberry Pi must be able to withstand temperatures of -20 to 115 °F, as determined by its location at the site.

7 Potential Risks

In this section, potential risks to the project's success are considered based on how likely the risk is to occur and how much of an impact that risk would have on the overall success of the development of the project. Risk likelihood ranges from low to high. Low meaning the risk is unlikely to ever occur, medium means that it is uncommon to occur and high meaning the risk is very likely to occur. Medium likelihood means that it is A high level of impact would entail destruction of hardware which extends further than the hardware the team is introducing to the system. A medium impact would cause a one to two week delay in the project completion. These are risks that are deemed most relevant to the project and the client by the project team, however these do not cover every possible risk, as it is unreasonable to know them all this early in the project.

7.1 Risk Overview

Risk	**Likelihood**	**Severity**
New or overlooked requirements	Medium	Medium
Raspberry Pi communicating through existing network	Low	Medium
Overlooked or incorrect interface hardware	Low	Medium
Weather or fire hazards	Low	High
Humidity	Low	High

7.2 Risk Mitigation

1. New or overlooked requirements

It should be addressed that the system at NPOI is very complex. There is a decent chance that a requirement was overlooked in elicitations with the clients. Given that new hardware must be ordered through a chain of command, the turnaround time to getting parts could slow the project down by upwards of 2 weeks. To mitigate this risk, these additional requirements will be addressed to the clients as the team tests. The client will give feedback for each prototype. A revision to this document can be made if seen necessary by the project team or the client.

2. Raspberry Pi communicating through existing network

The next risk is that the Raspberry Pi solution may not be able to communicate to BrainCon over the existing network at NPOI. There is a very small chance of this being a problem, as the network has Ethernet ports available for plug and play. However, until there are on-site tests, this remains a risk. If an alternative method is required to communicate over the network it would set us back a couple weeks to order new parts. This risk can be mitigated by communicating concerns to the clients and coming up with a solution for early testing.

3. Overlooked or incorrect interface hardware

A handful of cords in the current system are custom made. While the team has looked into the correct interface hardware, there is a small chance that further interface hardware is needed. This risk again, would just require ordering new hardware. This risk can be mitigated by further detailing the connections required for each device to assure everything is in place. Technological demos will also aid in the understanding of what needs to be bought for the project or manufactured by the team or an electrical engineer.

4. Weather or fire hazards

Weather related hazards relevant to the project include lightning, wildfires, floods and tornadoes. Each of these hazards could damage the Raspberry Pi or BrainCon and the existing hardware that each interacts with. Given that Flagstaff is already above 7000 feet in altitude, there is a non-zero chance for all of these to occur. The Raspberry Pi solutions are to be placed in Siderostat stations around the array which are small buildings, compact with electronics.

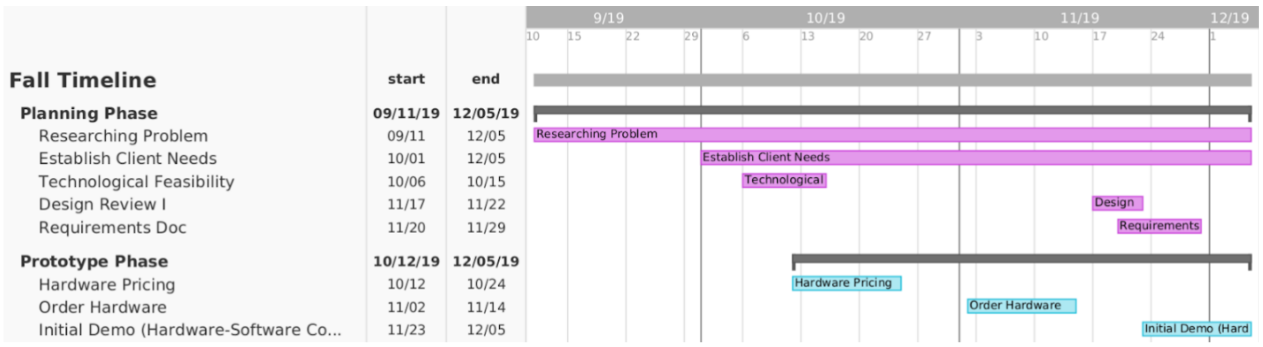
First, lightning strikes have killed people in Flagstaff in recent years and could hit one of the Siderostat stations, possibly destroying a lot of the equipment. Lightning can also cause wildfires, so fire hazards are included. Fire, for example, could also be caused by nearby campers or individuals throwing cigarettes out of cars on the nearby public road. There is a small lake north-east of the NPOI site, if heavy rain were to occur for a long period of time, flooding could become a problem. Last but not least, tornadoes have been sighted in Flagstaff before and would cause detrimental damage to the site. While there is little defense for an above ground system against tornadoes, nearby damage from nearby tornadoes could be mitigated by installing a protective barrier between the roof and the Raspberry Pi solution.

To mitigate lightning damage, a well insulated computer case would potentially help save the Raspberry Pis if a Siderostat station was struck. If the project receives more funding, fire damage can be avoided through fireproof computer cases. These cases are usually water resistant as well, so sprinkler systems would not be an issue. Flooding could be mitigated by raising the elevation of the Raspberry Pis and BrainCon within their respective rooms. There is little that can be done in the case that a tornado sweeps over the interferometer. However, if NPOI is narrowly missed by a tornado, the wind could tear the roofs down above the Raspberry Pis or BrainCon. To mitigate this risk, some internal structure could be created above the computer racks to support cave ins from the roofs. BrainCon and the Raspberry Pis should also be placed away from the windows and doors of each building.

5. Humidity

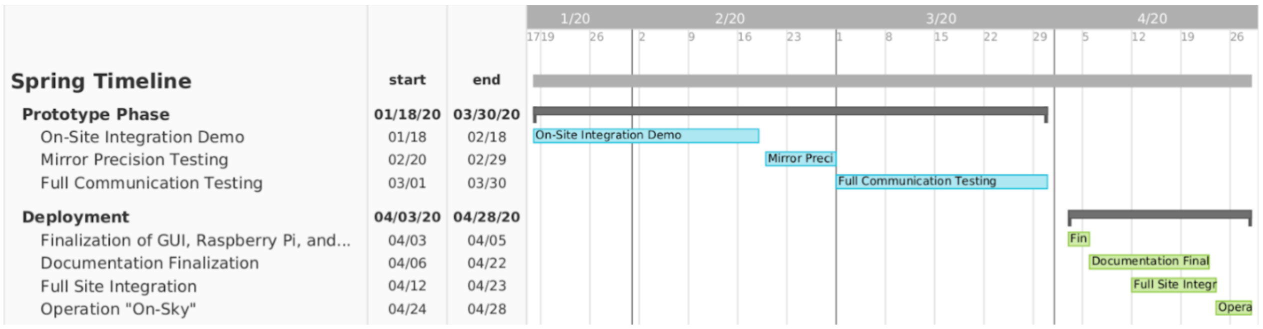
The buildings that house the hardware for the Siderostat and NAT are humidity controlled. If an observer was to accidentally leave the door open for too long, the humidity levels could raise substantially. Overexposure to high humidity could cause corrosion to circuit boards, requiring them to be repaired or replaced. The mitigation for this risk is also to supply a solution to insulate the Raspberry Pis.

8 Project Plan



Fall Timeline	start	end
Planning Phase	**09/11/19**	**12/05/19**
Researching Problem	09/11	12/05
Establish Client Needs	10/01	12/05
Technological Feasibility	10/06	10/15
Design Review I	11/17	11/22
Requirements Doc	11/20	11/29
Prototype Phase	**10/12/19**	**12/05/19**
Hardware Pricing	10/12	10/24
Order Hardware	11/02	11/14
Initial Demo (Hardware-Software Co...	11/23	12/05

Figure 1: Fall Timeline



Spring Timeline	start	end
Prototype Phase	**01/18/20**	**03/30/20**
On-Site Integration Demo	01/18	02/18
Mirror Precision Testing	02/20	02/29
Full Communication Testing	03/01	03/30
Deployment	**04/03/20**	**04/28/20**
Finalization of GUI, Raspberry Pi, and...	04/03	04/05
Documentation Finalization	04/06	04/22
Full Site Integration	04/12	04/23
Operation "On-Sky"	04/24	04/28

Figure 2: Spring Timeline

To measure the progress of this project and focus the team on common goals, clear milestones have been defined in the timeline of this project. The key milestones approaching as seen in Fig 1 and 2 include: a system demo, on-site integration demo, precision testing, communication testing, component finalization, documentation finalization, integration into the site, and "On-Sky" operation.

The system demo will show the fundamental components that make up the solution, performing the fundamental tasks required to execute the solution. This includes sending signals from a central computer to a Raspberry Pi controller over Ethernet, to ultimately drive a stepper motor. More importantly this demo will show the teams ability to use the chosen tools to execute a working demonstration of the components that will make up the final solution.

The on-site integration demo expands on the initial system demo by adding a layer of functionality and precision to the foundational capability established in the initial demo.

The testing phase includes precision testing and communication testing. The systems will be measured for precision to ensure the accuracy of the system has been maintained. Communication systems will be thoroughly investigated to ensure all components at each station can be reached through the network. Tests will prove communication can occur both ways across the network.

Component finalization is where the solution is deemed working, tested, and ready for integration. This milestone marks completion of all major hardware and software requirements. Documentation finalization comes shortly after.

Integration into the site is where the team moves the solution from the test rig and plugs the components into NPOI's working array. On-Sky operation is the final milestone marking the teams success in fitting their solution into NPOI's existing infrastructure. On-Sky operation confirms the solution works in its intended application.

9 Conclusion

The goal of the project is to update several components at NPOI, some hardware related, some software related. The current system in place at NPOI is outdated, hardware is constantly breaking down, and the existing system is overly complex. As a result, team Astraea has developed a plan to solve their problem. This solution will reduce the complexity of the system, reduce the rate of failure, and stay robust so that NPOI can continue to operate as hardware and software change in the future.

The requirements detailed in this document cover the specifics of each piece of technology deemed critical to the development of a working system for NPOI. These requirements consider constraints, capabilities, and risks to each piece of hardware or software to be integrated into the mirror control network. These devices and their specifications include the following:

- A centralized computer, BrainCon, which networks with an observer computer, sends commands to 3-6 Raspberry Pis and handles feedback loops. Specifically BrainCon's software will host a TCP server, enabling persistent connections, modular design, and a multithreaded network. A GUI will allow for an easy user interface for configuration and individualized station control. BrainCon will relay mirror commands to a Raspberry Pi in the form of packets of a job queue, and will receive feedback from networked devices through a dedicated thread.
- Each mirror station will host a Raspberry Pi, which will receive inputs from BrainCon, outputs electrical pulses to Siderostat stepper motor micro stepping drives, outputs DC signals to Ultravolt Piezoelectric Actuator Controllers, and receives feedback from optical limit switches. The Raspberry Pi software will need to network with BrainCon, receiving packets and relaying relevant data back to the central brain. It will also control mirror positioning, accomplished through sending pulses to microstepping drives for the Sid, or by sending DC signals to piezoelectric actuator controls for the NAT. Position of the mirrors will be tracked by utilizing limit switches, counting applied micro steps, and monitoring voltages applied. Finally, each Pi will be electronically insulated, preventing cross-talk with other electronic systems.

Each requirement aforementioned has additional performance or environmental specifications in the document, which offer additional insight to the hardware and software added to the system.

Although this solution attempts to solve many of NPOI's hardware and software issues, a few risks threaten the success of Astraea's project. These include learning new requirements during development, having issues with communicating over the existing network, realizing overlooked or incorrect interfacing solutions for existing hardware, and exposing the devices to fire, humidity, lightning, or other weather hazards. The team will attempt to mitigate these risks to ensure the project succeeds.

To conclude, Astraea is excited to re-design the mirror control system for NPOI, and would like to emphasize a desire to succeed in creating a working solution. Not only will the team save NPOI millions by replacing custom hardware, but Astraea will provide a significantly better, simplified system which reduces points of failure. Updated software and hardware will be much easier to understand, cheaper to replace, and will maintain precision of the instrument. Moving forward, the team hopes the system will allow NPOI to continue contributing to the field of astrometry and assisting in the exploration of the cosmos.

10 Glossary

AlignCon	Internal NPOI software to help align the mirror system.
Arcsecond	$\frac{1}{3600}^{th}$ of a degree.
BrainCon	A Dell server rack running Linux which acts a central computational unit.
Dear ImGui	Immediate-mode API for creating lightweight graphical user interfaces.
GUI (Graphical User Interface)	A user interface that allows users to interact with a device through graphical icons.
Limit Switch	An optical feedback mechanism for detecting when an object crosses through the switch.
Lowell Observatory	An astronomical observatory in Flagstaff, Arizona, known as one of the oldest observatories in the United States, famous for having discovered the coordinates of Pluto.
Microstep	An angular change of a stepper motor smaller than a 1.8 degree step.
NAT (Narrow Angle Tracker)	High-precision mirror responsible for reflecting light into NPOI's vacuum tube system.
NPOI (Navy Precision Optical Interferometer)	A specialized astronomical telescope system co-owned by Lowell Observatory, Navy Research Laboratory, and United States Naval Observatory.
NRL (Navy Research Laboratory)	The corporate research laboratory for the United States Navy and United States Marine Corps. It conducts basic scientific research, applied research, technological development and prototyping.

On-Sky	An astronomer colloquialism for when an instrument is in operation and gathering data from the sky.
Piezoelectric actuator	A mechanical translator which converts electrical energy directly into linear motion.
Siderostat (Sid)	A flat mirror which is used at NPOI for reflecting light from a celestial object to the NAT.
Siderostat station	A physical building located on NPOI's array which houses a Siderostat mirror, NAT, and all other devices associated with those mirrors.
Stepper motor	A type of DC motor that works in discrete steps, or angular positions.
TCP (Transmission Control Protocol)	A networking protocol that allows applications to maintain communications to each other through which to send and receive data.
Ultravolt Piezoelectric Actuator Control	A device that steps up low voltages to higher voltages and applies them to a piezoelectric actuator.