SPARSE APERTURE SPECKLE INTERFEROMETRY TELESCOPE ACTIVE OPTICS CONTROL SYSTEM

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Mechanical Engineering

by

Matthew Clause

December 2015

COMMITTEE MEMBERSHIP

TITLE:	Sparse Aperture Speckle Interferometry Telescope Active Optics Control System
AUTHOR:	Matthew Clause
DATE SUBMITTED:	December 2015
COMMITTEE CHAIR:	John Ridgely, Ph.D.
	Professor of Mechanical Engineering
COMMITTEE MEMBER:	William Murray, Ph.D.
	Professor of Mechanical Engineering
COMMITTEE MEMBER:	James Widmann, Ph.D.
	Professor of Mechanical Engineering
COMMITTEE MEMBER:	Russell Genet, Ph.D.
	Research Scholar in Residence

ABSTRACT

Sparse Aperture Speckle Interferometry Telescope Active Optics Control System

Matthew Clause

A conventional large aperture telescope required for binary star research is typically cost prohibitive. A prototype active optics system was created and fitted to a telescope frame using relatively low cost components. The active optics system was capable of tipping, tilting, and elevating the mirrors to align reflected star light. The low cost mirror position actuators have a resolution of 31 nm, repeatable to within 16 nm. This is accurate enough to perform speckle analysis for the visible light spectrum. The mirrors used in testing were not supported with a whiffletree and produced trefoil-like aberrations which made phasing two mirrors difficult.

The active optics system was able to successfully focus and align the mirrors through manual adjustment. Interference patterns could not be found due to having no method of measuring the mirror surfaces, preventing proper mirror alignment and phasing. Interference from air turbulence and trefoil-like aberrations further complicated this task. With some future project additions, this system has the potential to be completely automated. The success of the active optics actuators makes for a significant step towards a fully automated sparse aperture telescope.

Keywords: Telescope, Sparse aperture, Active optics, Actuator, Nanopositioning, Phasing

ACKNOWLEDGMENTS

The author would like to thank everyone who helped during the course of this project. John Ridgely, Russell Genet, and Dave Rowe were instrumental advisors who helped to guide this project. Michael Nidetz, Reed Estrada, and Chris Estrada were invaluable in designing and welding together the test frame used for this project. Chris Noone was an incredible help during welding, his skills helped make the project come together right the first time. Additional thanks goes to Hubble Optics who made the mirrors used in this project and provided valuable information and feedback.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

Telescopes used for astronomical research are typically very expensive and require costly equipment. A telescope design was drafted using separated mirrors to save on costs, creating what is known as a sparse aperture. Sparse apertures designs have been tried before, but previous designs have been quite costly. The purpose of this project was to design a low cost method for aligning telescope mirrors.

Statement of Problem

Various telescopes have been used to measure the orbits of binary stars over past decades. However, observing time on suitable telescopes has been becoming increasingly difficult to obtain. Records of known binary stars may no longer be kept up to date and new binary stars may receive no observation at all. A dedicated telescope is needed for the observation of binary stars to continue.

General Approach

The sparse aperture telescope was designed to be a low cost telescope for observing binary stars. For this prototype, three mirrors were spaced apart to maximize angular resolution, but close enough for speckle interferometry to be used. Each mirror was controlled using three actuators supports. An actuator to support telescope mirrors was designed capable of nanopositioning, using fine screws for supporting the mirror. The high thread count of the adjustor screws and a stepper motor achieve the desired accuracy at a fraction of the cost of similar off the shelf products. A test frame was created to test the actuators using a light source simulating star light. A camera was placed next to the light source to and the mirrors were controlled to focus and align the three mirrors.

Background

Active Optics

Active optics is the technology used to correct a telescope mirror from deformations due to thermal and gravity effects. This is accomplished by supporting the back of the telescope mirror using position actuators capable of nanopositioning. Furthermore, the largest telescopes today use segmented mirrors, where a multiple mirrors are aligned to act as one. Active optics is essential for the alignment of segmented mirror telescopes because the mirror surfaces need to be aligned and phased with one another, requiring nanometer precision. Mirrors without active control need to be made much thicker and require a rigid support system, increasing cost exponentially in relation the mirror diameter. By using a dynamic support method, larger mirrors can be used without an astronomical price increase [1].

Active optics systems typically operate at 0.1 Hz or less and can be open or closed loop. This can correct for gravity and thermal factors, but active optics does not correct errors from atmospheric turbulence, earthquakes, or other high frequency vibrations. These sources of error demand a much faster system response speed and are classified as adaptive optics. Adaptive optics must be closed loop systems and able to operate at 50 Hz or faster [2].

Speckle Interferometry

Speckle Interferometry is a technique used to overcome the limits of turbulent atmosphere and improve angular resolution of telescopes. A high-speed camera is used to take images with exposure times of 100 ms or less to minimize the atmospheric effects on the image. The light of captured images needs to be within phase to one quarter of the wavelength of light being measured. Fourier analysis is then used to reconstruct the ideal image through the atmospheric aberrations. This process is used to improve the capabilities of Earth telescopes past their normal seeing limits [1].

Binary Stars

Binary stars, often simply referred to as binaries, are a set of two stars which orbit each other around a common center of gravity [3]. By recording the orbits of binary stars of a known distance from earth, astronomers can calculate the stars' mass. Binary stars of a low light magnitude are often difficult to record in this way due to atmospheric aberrations. When the light of the brighter star passes through our atmosphere, the light blurs outward creating what is known as a seeing disc. This seeing disc can completely hide the presence of the dimmer star, making observations of orbits impractical with traditional telescopes [4].

Explanation of Limitations

The largest limiting factor for this project was finding parts within a small budget. Modern telescopes use hardware that can cost between ten to a hundred times more than what this project allowed. A drawback of the sparse mirror array is the inability to use edge sensors between mirrors, which are used on other telescopes to provide position feedback. The largest challenge of this approach was the difficulty to achieve tolerances using parts not originally intended for astronomical use.

This project only addressed the active optics system of the sparse aperture design. The telescope frame, automation software, and image processing techniques will be covered by other projects another time. The scale of work involved is beyond the scope one project. This project did not develop an adaptive optics system due to the high cost of necessary equipment and extensive controls design required.

II. LITERATURE REVIEW

Related Work

Many different active optics designs exist today. Some previously high budget designs have been made relatively inexpensive due to the advancement of electronics and manufacturing processes. Some design ideas from the CELT and KECK telescopes were reworked to fit the low cost approach of this project. The design ideas which contributed to the final design of this project are described below.

WIYN Telescope

The WIYN telescope is a 3.5 m telescope originally completed in 1994. The design uses stepper motors with toothed belts to actuate the mirror surface to correct mirror distortion. This design is significant in that it “floats” the mirror on a secondary pneumatic support system. This floating support greatly reduces the amount of force the position actuator needs to exert on the mirror.

A pneumatic support would be costly to reproduce for the sparse aperture telescope. A stepper motor with rotary reduction is something which can be done cheaply however, and was used for this project. The tooth belt used for the reduction was eliminated however; otherwise an encoder would be needed to deal with the effects of backlash.

KECK Telescope

The KECK telescope is a 10 meter segmented mirror telescope first built in 1996. The individual mirrors were designed to be controlled with a simple, compact position actuator. Each actuator has a mounted servo motor which drives a lead screw. This actuates a nut mounted on a linear ball slide. The vertical actuation is then reduced using a pneumatic diaphragm [5].

The KECK motor driven lead screw concept was implemented in this project. The linear ball slide was implemented as well, but was used in a different way. The pneumatic diaphragm is not a common part, relatively expensive compared to the other components, and is prone to failure problems. The pneumatic diaphragm was not used in this project because of these reasons.

CELT Telescope

The CELT (California Extremely Large Telescope) is a 30 meter segmented mirror telescope planned to be operational in 2022. The CELT position actuator uses a voice-coil and trim motor assembly to control mirror position. This assembly is built into a combination of spiral and lever flexures. Flexures have a distinct advantage in that they do not generate backlash. Furthermore, a position sensor is used to provide feedback for controlling position and stiffness [6].

A voice-coil and trim motor assembly provides high accuracy actuation, but has a limited range of travel. This limited range of travel made voice-coils unsuitable for this project. The flexure assembly of the CELT is quite complex and would be problematic to recreate in this project. A simplified flexure design was designed for this project which was inspired by the CELT design.

Previous Work

Conventional large aperture telescopes required for binary star research are typically cost prohibitive. A number of workshops and meetings have already taken place to address this issue. These workshops have devised a solution; an automated telescope design with a sparsely populated aperture could potentially be built for 1/100th of the price of the recent Discovery Channel Telescope at Lowell Observatory. By using spherical mirrors sparsely arranged to form a larger virtual mirror, mirror costs can be drastically reduced while achieving the angular resolution required for binary star research [4].

Segmented mirror telescopes require either active optics or adaptive optics systems to ensure the mirror array is aligned and in phase. To keep project costs low, an inexpensive active optics design is needed. Additionally, active optics is a key step towards automating the sparse aperture telescope. With the active optics design completed, teams under Russell Genet can move forward onto other parts of the telescope design.

III. SYSTEM DESIGN

The sparse aperture active optics system was designed to satisfy the optical requirements needed for speckle analysis. A support frame was designed out of steel to attach the necessary hardware. The actuators needed to be designed to be both precise and low cost. A design goal was to use as many off the shelf components as possible to make future development easier. This ended up working well for most of the parts chosen, but certain parts needed to be custom made to house the components. Once the physical assembly was finished, a suitable electronic controller was chosen and mirror control program created.

Optical Requirements

Speckle analysis requires a surface accuracy roughly one quarter the wavelength measured. This system was designed for observation of visible light. Therefore, the active optics system needed to be accurate to at least 100 nm. Each mirror needs to be able to travel 1 cm to focus the image. Tip, tilt, and elevation control are required to align segmented mirrors, requiring at least 3 position control points.

The three mirrors were placed to create a 686 mm aperture. This diameter was chosen to give the best aperture resolution while still allowing for speckle analysis. If the mirrors are spaced too far apart, shorter wavelengths of light can no longer be analyzed using speckle analysis. This mirror spacing was then simulated using speckle simulator software and confirmed to be valid by Dave Rowe, an optics and astronomy expert from PlaneWave Instruments.

Mirror Properties

The three spherical mirrors used for this project were created as a matched set by Hubble Optics. Each mirror has a radius of curvature of 4972 mm, giving a focal ratio of 7.25 with the 686 mm aperture. Each Pyrex glass mirror is 267 mm in diameter and 25 mm thick, theoretically allowing for simple 3-point support. Each mirror was supported at three evenly spaced points on two-thirds the mirror's diameter. No whiffletree was used to support the mirror, giving a stiffer design but subject to higher surface error. Finite element analysis (FEA) was used to estimate deflection due to gravity to be 173 nm, with the maximum allowable being 200 nm. As seen in Figure 1, the surface deflection is greatest at the edge furthest from the support points.

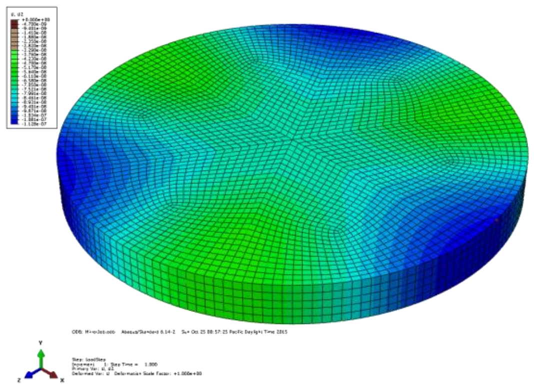


Figure 1: FEA analysis of mirror surface deflection

Mirror Geometry

Each mirror has 6 DOF and needs to have tip, tilt, and elevation control. The mirror support fixture prevents lateral motion and axial rotation of the mirror, accounting for 3 DOF. The tip, tilt, and elevation are controlled by three actuators against the bottom surface of the mirror. Each actuator is equally spaced on 2/3rds of the mirror's diameter.

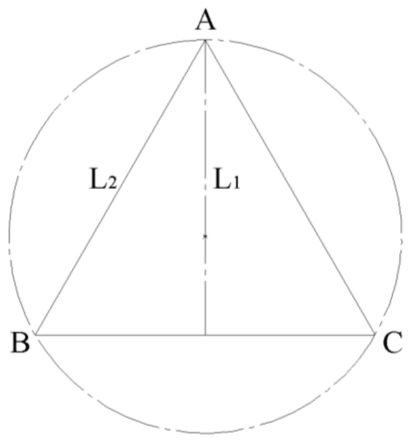


Figure 2: Geometry of mirror support points

As seen in Figure 2, an equilateral triangle can be drawn between the actuator contact points. Actuator A is the inner actuator, being closest to the center of the mirror array. Using simple geometry, the dimensions of the triangle are determined in relation to the radius of the support circle.

$$L_1 = \sqrt{3}\,R \tag{3.1}$$

$$L_2 = \frac{3}{2}\,R \tag{3.2}$$

The simplest mirror variable to control is elevation, designated h. Mirror elevation is calculated by taking the average height of the actuator, seen in 3.3.

$$h = \frac{A + B + C}{3} \tag{3.3}$$

Calculating the mirror tip required finding the angle of the mirror towards A, the center of the mirror array. In this assembly, there is no way to directly measure or control the center of the mirror, so the midpoint of B and C was be used instead. The tip angle can be calculated by calculating the inverse tangent of the difference in height L$_1$. The difference of actuator height between A and the midpoint of B and C are used to calculate the tip α in 3.8.

$$\alpha = \tan^{-1}\left(\frac{A - \frac{1}{2}(B + C)}{L_1}\right) \qquad (3.8)$$

The mirror tilt β is controlled by B and C. A is defined on the directly on the tilt axis. Because of this, A is not used in the tilt calculation seen in 3.9. Similar to 3.8, inverse tangent is used to calculate the angle.

$$\beta = \tan^{-1}\left(\frac{B - C}{L_2}\right) \qquad (3.9)$$

Inverse tangent calculations are resource heavy for simple microcontrollers to calculate. In this test setup the mirrors are never tipped further than 3°, which makes this problem a good candidate for small angle approximation. The tip and tilt equations used for this project's microcontroller are shown in 3.10 and 3.11.

$$\alpha = \frac{A - \frac{1}{2}(B+C)}{L_1} \qquad (3.10)$$

$$\beta = \frac{B-C}{L_2} \qquad (3.11)$$

Using the tip, tilt, and elevation equations 3.3, 3.10, and 3.11, the actuator height equations can be determined. For given a mirror position, 3.12, 3.13, and 3.14 are used to calculate the required actuator heights. These equations rely on small angle approximations, but are a fast and efficient way for low power microcontrollers to determine actuator positions.

$$A = \frac{2\,L_1}{3}\,\alpha + h \qquad (3.12)$$

$$B = -\frac{L_1}{3}\,\alpha + \frac{L_2}{2}\,\beta + h \qquad (3.13)$$

$$C = -\frac{L_1}{3}\alpha - \frac{L_2}{2}\beta + h \qquad (3.14)$$

Mechanical System Design

Mirror Support Frame

The frame was constructed from steel square tubing because of its high stiffness and low cost. Additionally, this A500 alloy steel is simple to machine and weld. The frame has three mirror mount locations with two holes for each mirror. The mirror cell mounting plates are secured to the support frame using ½" bolts. The mirror support frame uses a bolt to fix it to the test frame, locking it into place. FEA analysis indicates this support frame should have a first harmonic above 20 Hz, which should overcome low frequency vibrations due to the outdoor environment.

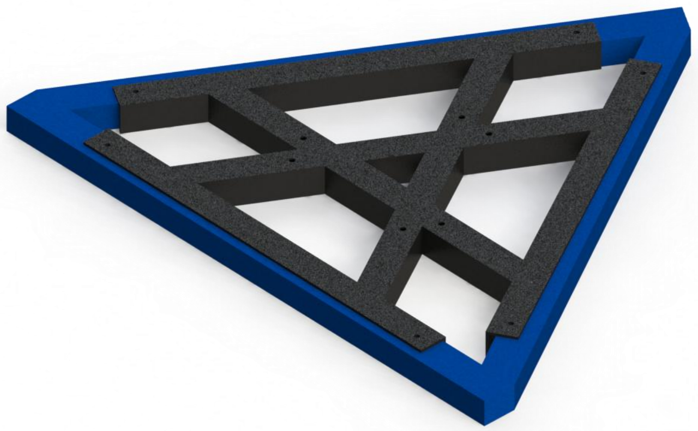


Figure 3: SolidWorks render of mirror support frame

Active Optics Actuator

The actuators used in for the active optics system attach to the mirror cell mounting plate using two screws. The actuator consists of a steel housing, stepper motor, linear bearing, solid shaft coupling, and a fine adjustment screw. These components are protected from dust and other contaminants with a side cover cut from ABS plastic. Each actuator is also equipped with a limit switch to ensure the mirror cannot collide with the actuator's steel housing. The actuator's component design is seen in Figure 4.

This actuator design is open loop, utilizing no feedback other than a limit switch. This removes the cost of high resolution encoders, but introduces the possibility of error. The sparse aperture telescope will eventually have position feedback from measurement of star images, but could not be included in this project. Because of this, the actuators need to be controlled manually for this project.

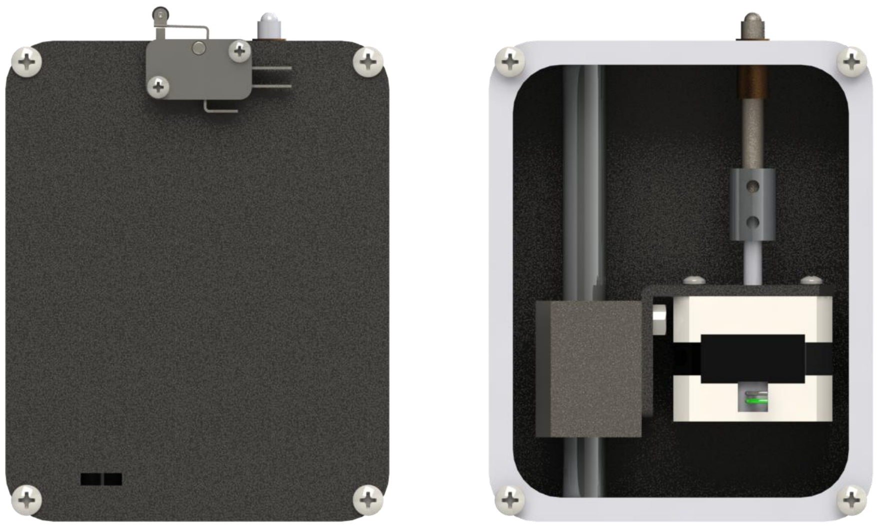


Figure 4: SolidWorks render of actuator

With 254 threads per inch, the fine adjustor screw has a pitch of 0.1 mm. The stepper motor chosen has 200 steps per revolution and the motor driver supports 1/16 microstepping. This provides 3200 steps per revolution. Dividing the pitch by the steps per revolution, screw height can be controlled to 31.25 nm, which is less than 1/10th of the shortest wavelength of visible light.

Actuator Housing

The actuator housing was a custom steel enclosure made from 4” x 5” rectangular tubing. Each housing has a set of holes to press fit a linear bearing shaft into place. The fine adjustment screw bushing is slip fit into another hole on the top of the housing, with 2 mounting holes located directly below it on the bottom of the housing. Each enclosure has tapped holes on its side edge to secure the side covers. The ABS plastic side covers have slots for the motor wires and a set of mounting holes for a limit switch.

Fine Adjustment Screw and Bushing

The actuators used Kozak fine adjustment screws with matched threaded bushings. The adjustor screws are ¼” in diameter with 254 threads per inch. Each adjustor screw is 2” long, but has approximately 2 cm of travel once installed in an actuator. Each screw supports up to 40 lb axially, being limited by fine threads. The adjustor screws are made with 303 stainless steel which helps to minimize the risk of part failure due to corrosion. Each fine adjustor screw has a steel ball bearing tip with which contacts a sapphire pad adhered to the back of the mirror. The sapphire pad is detailed in Table 1.

Table 1: Sapphire pad general specifications

Diameter	0.160" ± 0.001"
Thickness	0.020" ± 0.001"
Parallelism	3.5 arcmin
Coefficient of Friction	0.15 Against Steel

Stainless steel, like any other material, is vulnerable to thermal expansion. 304 stainless steel has a coefficient of expansion of $17.3 \cdot 10^{-6}$ mm/mm K. When the screw is actuated 23mm (max), then the actuator tip will experience deflection of approximately 398 nm per degree Celsius due to thermal expansion. This change of temperature is enough to disrupt mirror phasing. To combat this, the stepper motors power off when not in motion to avoid unnecessary heat generation.

Although it is stainless steel, the adjustor screw operates best in non-condensing conditions. The fine threads are susceptible to dust and dirt and should be covered when not in use. Furthermore, the bronze bushing is self lubricating, meaning this part should not be lubricated. Failure to follow these guidelines could cause the screw to experience heavy friction or seize [7].

Stepper Motor

NEMA 17 bipolar stepper motors were used to drive the adjustor screw. The stepper motor chosen for this project is specified in Table 2. Paired with a microstepping motor driver, stepper motors can rival the precision of a servo motor at a fraction of the price. Stepper motors were chosen for their achievable accuracy with minimal cost. The permanent magnet stepper motors used in this project also retain a light holding torque when unpowered, which helped to prevent loss of position when the motor was powered off.

Table 2: Stepper motor general specifications

Manufacturer	Changzhou Songyang Machinery & Electronics Co
Part Number	SY42STH38-1684A
NEMA size:	17
Weight:	285 g
Shaft type:	5 mm "D"
Steps per revolution:	200
Current rating:	1680 mA
Voltage rating:	2.8 V
Holding torque:	51 oz•in
Coil resistance:	1.65 Ohm
Inductance per phase:	3.2 mH
Number of leads:	4

A linear bearing was used to secure the stepper motor under the fine adjustor screw. Once properly adjusted, the stepper motor is free to travel vertically with approximately 14 arcmin of rotary backlash. Due to the placement of the linear bearing in the assembly, vertical backlash is not a concern. It is because of this reason that an inexpensive linear bearing can be used.

A solid shaft coupler connects the stepper motor shaft to the fine adjustor screw, ensuring minimal torsional deflection. Set screws secure the coupler onto the stepper motor and fine adjuster screw.

Mirror Support Fixture

The support fixture provides lateral mirror support while allowing vertical actuation. This fixture design, using 0.06” thick ABS, is capable of vertical deflection exceeding 2 cm. Furthermore, it limits horizontal deflection to less than 2 mm. A ½” nut is fixed onto the fixture using Cyanoacrylate (super glue) for attachment to the mirror support plate. Access to a laser cutter makes this part fast and cheap to make. FEA simulation was used to estimate 0.06” ABS can deflect 2 cm without fracturing. Figure 5 shows where the greatest stress occurs under deflection.

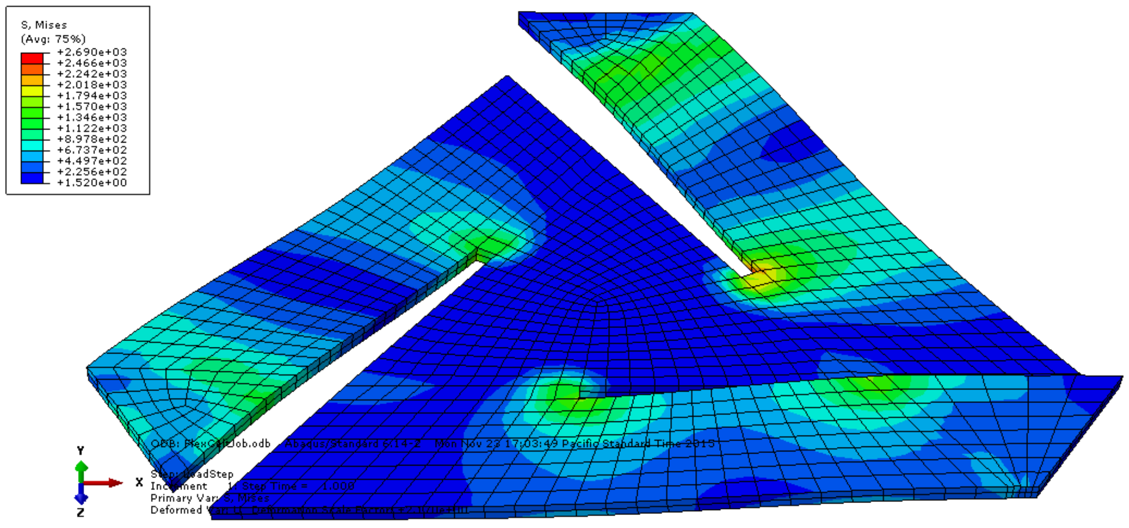


Figure 5: Abaqus simulation of flexure stress concentrations after 2 cm of actuation

Mirror Cell Mounting Plate

The mounting plate was made from ¼" plate steel. Three actuators are connected to each mounting plate using two ¼" screws. A render of a fully assembled mirror cell can be seen in Figure 6. A ½-6" bolt is threaded through the center of the mounting plate to connect the mirror support fixture. This bolt can be adjusted to set the mirror's lowest height. Each mounting plate has ½" holes through which the plate can be bolted to the mirror support frame. The frame with mirror cells attached is seen in Figure 7.

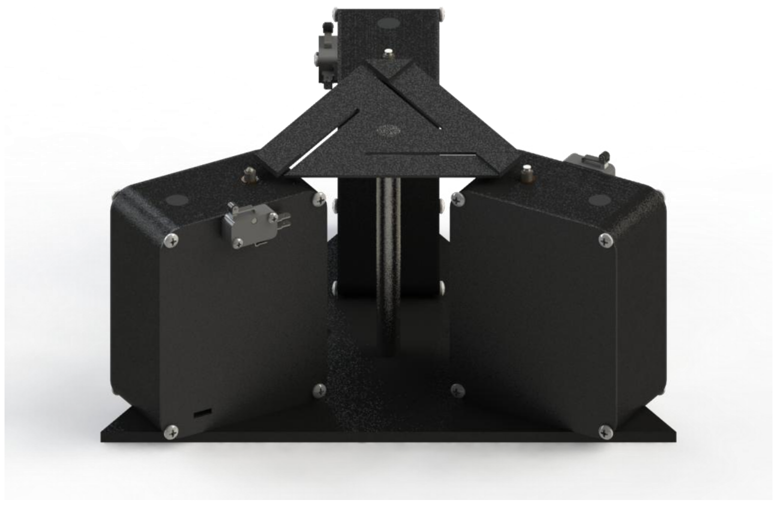


Figure 6: SolidWorks render of open mirror cell

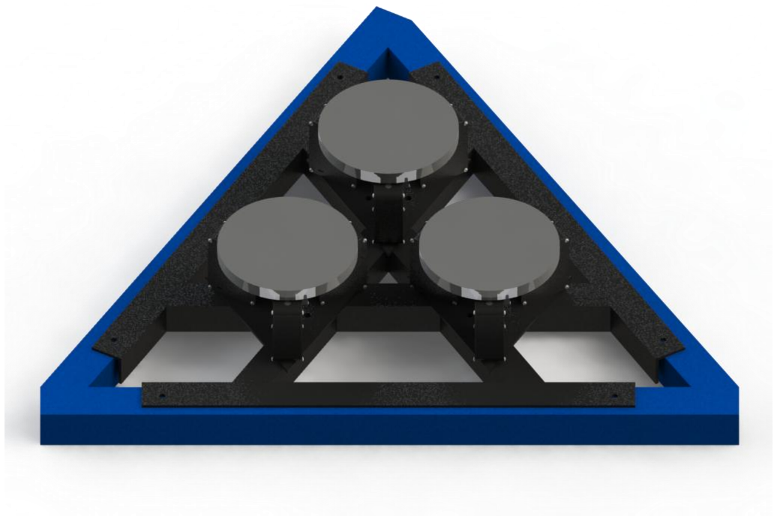


Figure 7: SolidWorks render of finished assembly

Electrical System Design

Microcontroller

An Arduino Mega 2560 was used to control each mirror cell. This board uses an Atmega 2560 microcontroller and can be programmed using C++. Arduino has an extensive set of preexisting libraries which makes creating new programs easy. The board operates at 16 MHz, allowing plenty of time for motor control and communication protocols. Furthermore, a programming header is available for installing a custom configuration, such as the real time operating system FreeRTOS.

Motor Shield

An inexpensive motor shield developed by the 3D printing community was found to be a perfect fit for this project. The relevant component capabilities of the RAMPS shield are shown in Table 3. The RAMPS and attached Arduino are powered by a 12V power supply. Higher voltages may damage the connected Arduino, but the RAMPS board can be modified to prevent this [8]. Each motor controller used 3 stepper drivers, one stepper driver per actuator. Basic limit switches require no special hardware because the Arduino Mega utilizes internal pull-up resistors. The I2C header allows for future projects to simultaneously control a large number of Arduino control boards. Due to its popularity in the 3D printing community, the electronics needed for this project can be purchased together as a set at a reduced cost. This set includes an Arduino Mega, RAMPS motor shield, and 5 stepper motor drivers.

Table 3: RAMPS 1.4 hardware support

Stepper Drivers	5
Servo Motors	4
End Stops	6
I2C Connector	1

Motor Driver

The A4988 motor driver is available on a breakout board made to interface with the RAMPS 1.4 shield. This motor driver is capable of 1/16 microstepping, providing the stepper motors the required angular resolution at a low price. Motor torque can be manually adjusted by increasing the motor current via a potentiometer on each motor driver. Each stepper motor was given 1 amp of current, the recommended upper limit of the A4988 motor driver. Motors can be turned off when not in use, which conserves power and prevents unwanted heat. Powered off motors will not lose their position under normal circumstances, but may be vulnerable to heavy vibration or user handling.

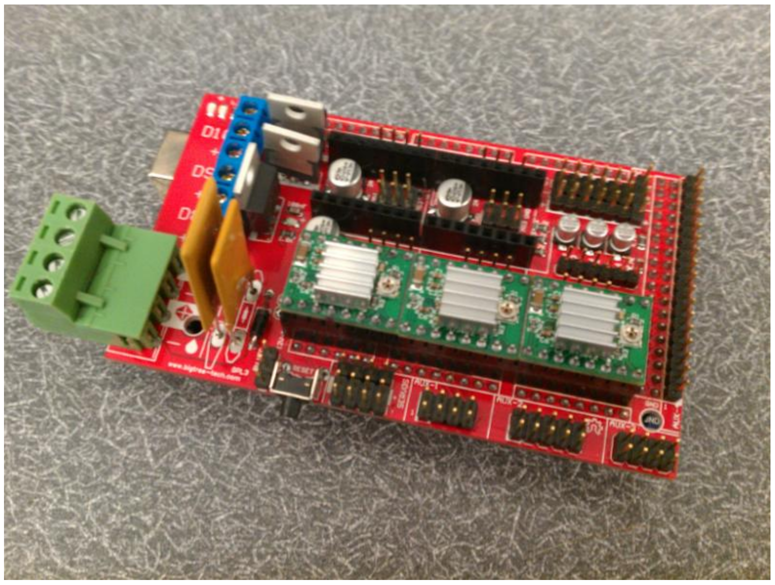
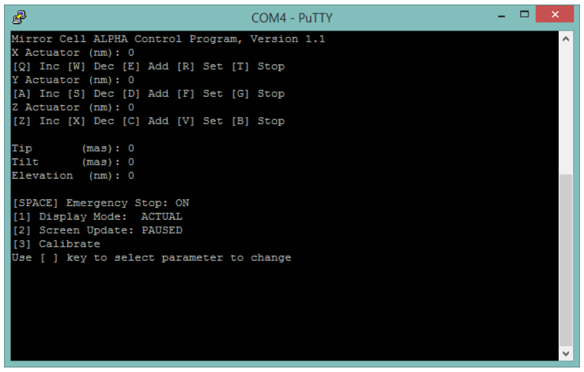


Figure 8: Arduino MEGA with RAMPS shield and A4988 motor controllers

When RAMPS board loses motor power but the Arduino remains powered through USB, the Arduino will not detect the loss of motor power. If the Arduino main program tries to send a step command to an unpowered motor driver, the main program will no longer reflect accurate position due to lack of feedback. Because of this, it is important to double check all power connections prior to operation. Alternately, an opto-isolator could be connected to the RAMPS 12V line to detect power loss, but this was not done for this project.

Software

A custom Arduino control program was written to control the actuators during these tests. The program is capable of controlling each actuator individually, displaying mirror tip, tilt, and elevation. Communicating with the Arduino control program requires a serial USB connection. The Arduino control program requires ANSI escape commands to refresh the display values. The free software PUTTY was chosen to interface with the control program. PUTTY is available on both Windows and UNIX systems.



```
Mirror Cell ALPHA Control Program, Version 1.1
X Actuator (nm): 0
[Q] Inc [W] Dec [E] Add [R] Set [T] Stop
Y Actuator (nm): 0
[A] Inc [S] Dec [D] Add [F] Set [G] Stop
Z Actuator (nm): 0
[Z] Inc [X] Dec [C] Add [V] Set [B] Stop

Tip       (mas): 0
Tilt      (mas): 0
Elevation  (nm): 0

[SPACE] Emergency Stop: ON
[1] Display Mode:  ACTUAL
[2] Screen Update: PAUSED
[3] Calibrate
Use [ ] key to select parameter to change
```

Figure 9: Arduino command program interface through PUTTY

Speckle Interferometry

While not a part of this project, speckle interferometry will be incorporated in future projects. This provides position feedback needed for a closed loop system. Besides encoders, other types of sensor do not have the range of travel necessary for this project application [9]. Additionally, an interferometer can be used to determine the mirrors' surface position. Position feedback from the interferometer could then be paired with an image processor, creating the feedback loop in Figure 10.

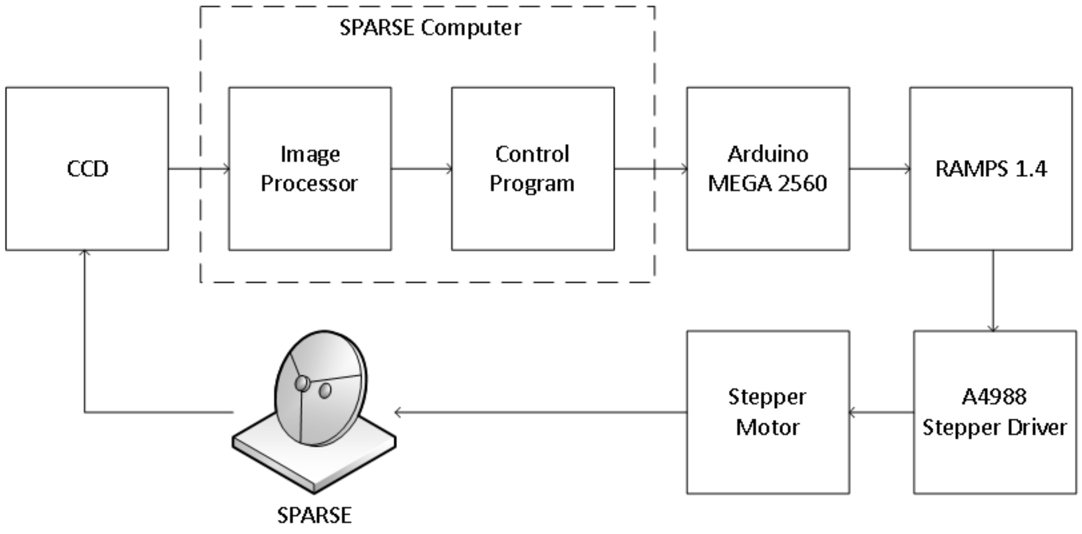


Figure 10: Proposed Sparse Aperture Interferometry Telescope closed loop model

System Scalability

Future sparse aperture designs may use larger, heavier mirrors, requiring the actuators to be upgraded. Larger diameter precision lead screws can be used to support heavier mirrors, but have fewer threads per inch. This upgrade requires a toothed belt and pulley be added to achieve the accuracy required. The backlash introduced by the saw toothed belt and larger thread would require an encoder be attached to the lead screw for position feedback. The stepper motors can be upgraded to NEMA 23 size, providing more torque for heavier loads. The A4988 motor drivers can be upgraded to DRV8825 motor drivers. These motor drivers have a higher power capacity adding extra power for use with bigger stepper motors.

Alternate Designs

Differential threads were initially considered for their high positional resolution. It is possible to design a custom differential thread mechanism, but this would add unnecessary complexity to the final assembly. This was rejected due to short range travel and lack of market availability.

A planetary gearbox was considered for use with a stepper motor as it provides a significant step up in accuracy. Low cost gear boxes have several degrees of backlash however, which would make mirror calibration difficult. Higher precision and low backlash gearboxes are available, but were rejected due to their high cost.

Voice coils were considered due to their simple construction, accuracy, and travel. They have no backlash and some models are accurate better than 1 µm. The downside is that voice coils require continuous power and can be large and heavy depending on system requirements. Additionally, voice coils are substantially higher in cost because cheaper models would not withstand the mirror weight, which is why they were rejected.

Piezo actuators are well suited for astronomy applications. They have no backlash, high accuracy, and can support mirror weight. Unfortunately, they are very expensive and well above the budget for this project. Additionally, piezo actuators often require higher voltages, which would drive up project cost and could pose a potential safety hazard.

IV. SYSTEM IMPLEMENTATION

All custom parts were machined and welded with some minimal tolerance issues. The actuators were finished and electronics systems checked for proper operation before assembly. The finished parts were then assembled in the Cal Poly HVAC lab room. The final design was completed on budget, although there were project delays due to machining time.

Bottom Assembly

Support Frame

During manufacturing, the mounting holes for the mirror cells shifted out of tolerance. This was due to thermal expansion caused by welding the frame. This thermal expansion offset the mounting positions by up to 1/8”. As a result, each mounting plate had to be fitted to an individual set of mounting holes on the frame. Although this was a problem, these effects were minimized by a careful distribution of heat during welding. The heat effects would have been much worse without the help of the welder Chris Noone. Figure 11 shows the final test fitting of the support frame to the telescope base, a critical step before making the final welds.

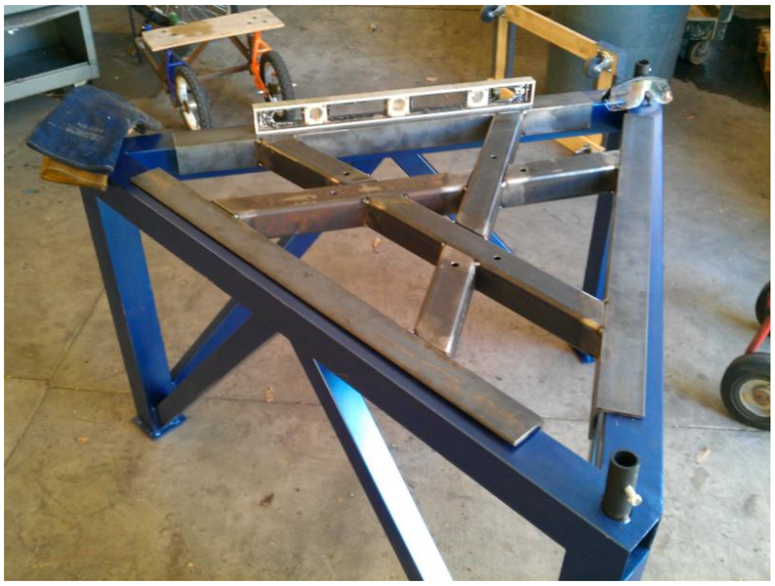


Figure 11: Mirror support frame being test fit onto telescope base

Actuators

The actuator housings were cut from 4” x 5” hollow rectangular tubing. The steel tubing was stiff, inexpensive, and available from a local steel supply store. Drilling the holes for the actuator components proceeded without issue. There was issue cutting the tubing to length however; the steel tubing did not fit under a chop saw, which caused the side edge to not be flat. Because this only affected the side covers, it was deemed not important enough to devote more machining to.

The first step to assembling the actuators was to press fit the linear bearing and shaft into the housing. The stepper motors were attached to the linear bearing with a NEMA 17 L-bracket. The holes for mounting to the linear bearing were manufactured to fit M4 screws, but the linear bearing used M5 bolts. To correct this, the L-brackets holes had to be expanded with a 5 mm drill bit. This was a much faster solution rather than fabricating brackets from scratch, and proceeded without issue.

Each threaded bushing is slip fit into its housing and secured with Loctite 312 according to best practice. Due to the tight thread tolerances, press fitting the bushings could result in seizing of the screw [7]. Once glued into place, the shaft alignment was adjusted using the slots on the L-brackets holding the stepper motor. This process turned out to be more challenging than expected, and ended up being a minor issue later in the project.

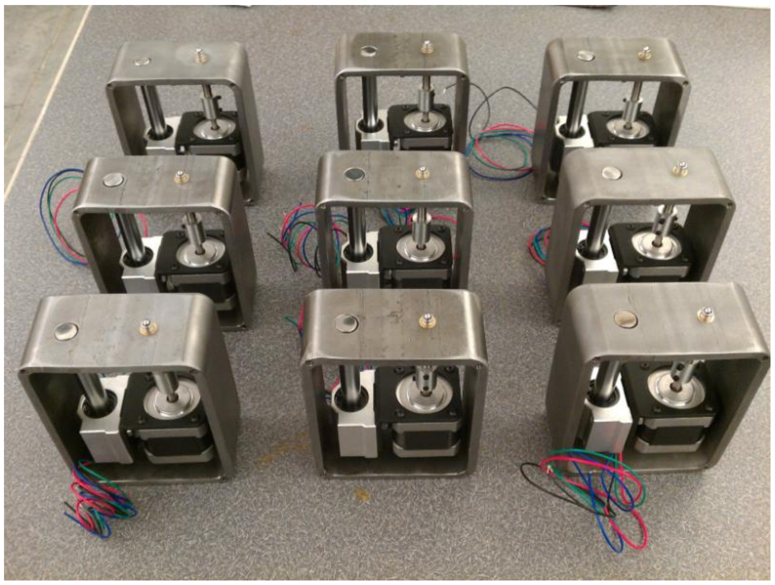


Figure 12: Actuators before side covers and paint

ABS side covers were laser cut to fit the housing. Additionally, holes were laser cut into the ABS side covers to mount the limit switch and to route the stepper motor wires. The holes for mounting the limit switches were designed undersized, allowing the metal screws to tap the soft plastic. Unfortunately, the laser cutter tolerance was not sufficient for this task and the holes were too large. The limit switches were mounted to the side covers by filling the oversized holes with Cyanoacrylate.

Before assembly, all the steel components were primed using acid etch and then painted with Krylon Ultra Flat Black spray paint. This paint was chosen because of its low reflectivity [10] and emissivity [11]. Additionally, this off-the-shelf paint is relatively inexpensive compared to many telescope paints. This black paint prevents unwanted light from bouncing back into the camera during testing. This was also important for preventing rust, ensuring a longer part life.

Mirror Cell

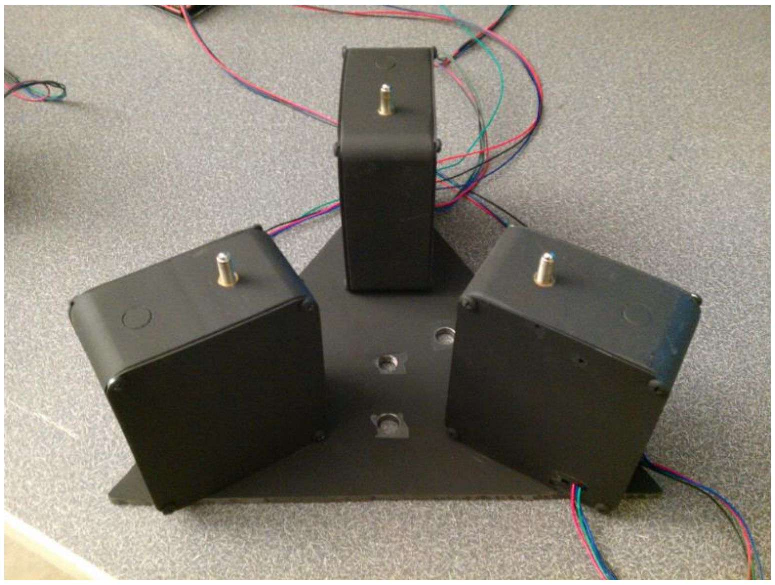


Figure 13: Mirror cell with actuators before installation of mirror

As seen in Figure 13, the mirror cell was bolted together before being attached to the support frame. A sapphire pad was adhered to the back of the mirror surface using Cyanoacrylate matched to the placement of each actuator, as seen in Figure 14. This sapphire pad prevents wear on the back of the mirror and ensures a low coefficient of friction.

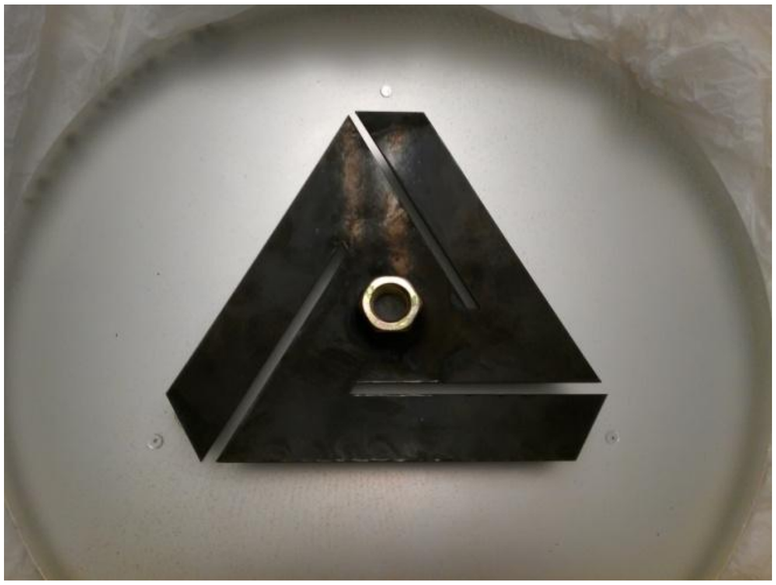


Figure 14: Attached mirror support fixture and sapphire pads

Electronics

Each mirror cell was controlled by an Arduino connected to a laptop. The control software written for the Arduino could control the tip, tilt, and pitch of each mirror. The Arduino program was controlled using PUTTY. This system was open loop and required manual positioning of the actuators.

The electronics used in this project were never mounted to the assembly. This did not impact the project itself in anyway, but did make routing wires inconvenient. A laser cut housing was planned for mounting purposes, but was not completed due to time constraints. Future builds should include this as a safety precaution. Figure 15 shows how the electronics were placed on a thick plastic sheet to protect against short circuits.

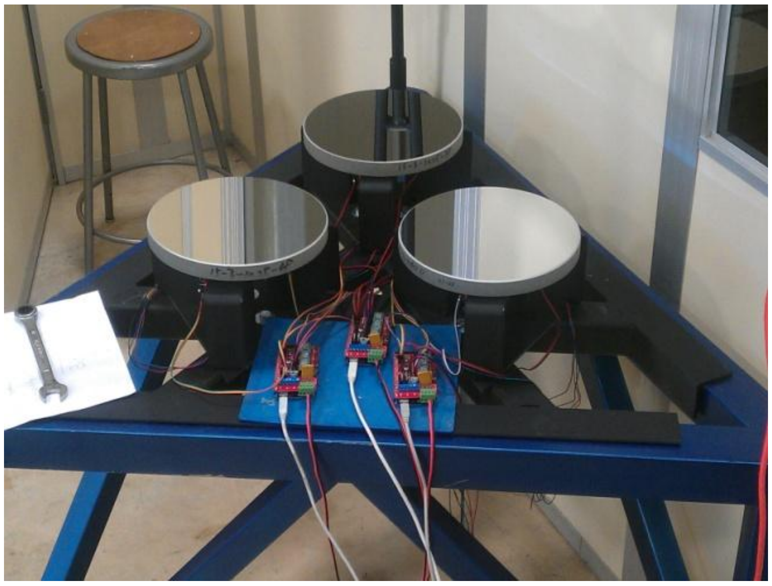


Figure 15: Sparse active optics test setup

Top Assembly

Camera

The QHY5L-II M astronomy camera was used for the fine mirror alignment tests. This camera was attached to a Barlow of approximately 2.5 times magnification. The camera was mounted to the top assembly and could be adjusted up and down approximately 15 cm. This rough adjustment was used to get the mirrors close to focused before using the actuators for fine adjustment.

The QHY camera was controlled with the free Windows software FireCapture. This program controlled exposure time and gain control for the QHY camera. It also provided a fast capture option for rapid image acquisition, useful for speckle interferometry. Light images were automatically tracked and cropped, drastically reducing file size when taking large batches of pictures. If the QHY camera is connected to a Linux system, the free software oaCapture can be used instead.

Light

A green LED with an approximate bandwidth of 10% was used for the fine mirror alignment tests. A 10 µm pinhole was placed over this LED to simulate starlight. This light simulates the light power of a binary star, but can only generate one point of light. The light and camera were installed with a 36 mm separation. The camera and LED, seen in Figure 16, needed to be placed in close proximity to one another to protect against optical aberrations from distorting images.

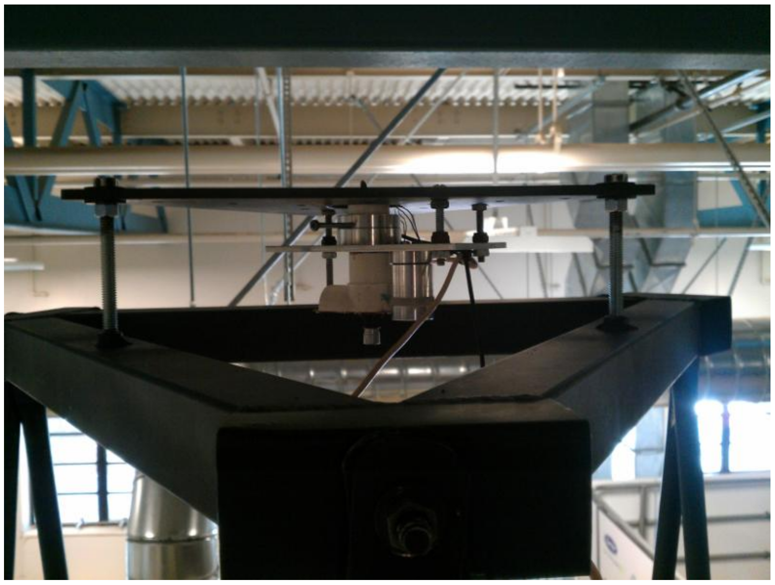


Figure 16: White camera housing (left) and silver LED housing (right) on top assembly

Final Assembly

The test frame was designed by architecture student Michael Nidetz for his senior project. The frame was welded by Reed Estrada and Chris Estrada. Figure 17 shows the finished assembly used to test the active optics system. The test frame was quite tall, measuring over 16 ft tall. The final assembly was set up in the HVAC lab room to make use of its high ceiling and upper balcony. The mirror support frame and mirror cells were bolted together on the frame and set up for testing.

The mirrors were controlled by the actuators to reflect the light onto the camera. The light from the LED was aligned and focused onto the camera, but was initially too far out of focus for the active optics system. The height of the top assembly needed to be adjusted several times until the focal length was close enough for the system to work. When not in use, the mirrors were covered with black dust blocking buckets. These buckets were also used to isolate mirrors during testing. Once this last adjustment was finished, the active optics system was ready for testing.

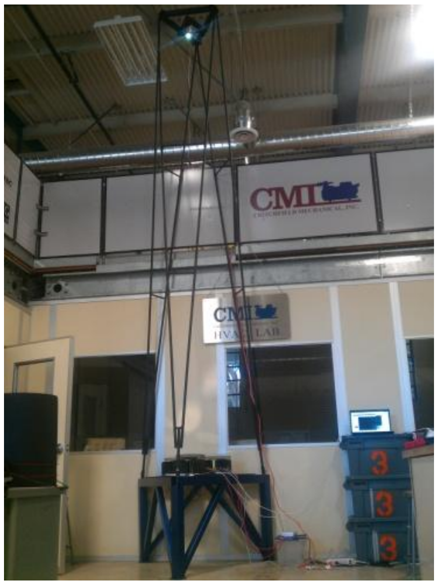


Figure 17: Completed sparse aperture active optics test assembly

V. TESTING AND EVALUATION

Several tests needed to be conducted to determine the ability of the active optics actuators. The fine adjustor screw test was performed to measure backlash of thread. Each subsequent test then built off of the previous results. The rough alignment test was done to determine the basic performance of the actuators and to find issues in the assembly. The fine alignment test was done to determine if the actuators were able to focus and align the mirrors. Lastly, the mirror phasing was adjusted in search of an interference pattern.

Adjustor Screw Microstepping

This test was done to measure the backlash and repeatability of the fine adjustor screw. Figure 18 shows the equipment used to measure angle changes projected across an 8.3 meter room. A green laser was glued to the top of the adjustor screw and then the adjustor screw was put under a 5 kg load to simulate mirror weight. Using an Arduino with a microstepping motor driver, the adjustor screw was actuated to test for position error.

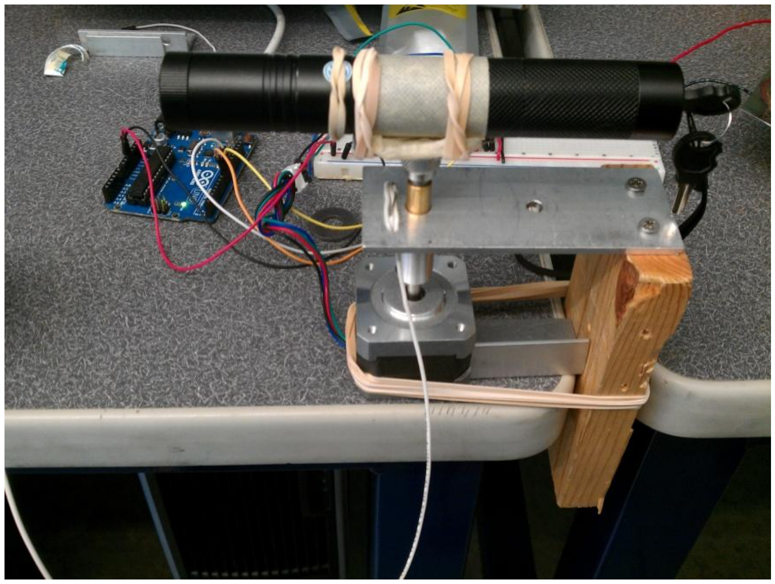


Figure 18: Fine adjustor screw test setup

As seen in Figure 19, positional error was found to be within 20 arcmin of the target. This is due to the backlash of the thread between the adjustor screw and bushing, approximately 40 arcmin. When the effects of backlash are accommodated for, positional accuracy is within 4 arcmin. For the screw height, this converts to an expected backlash of 182 nm. When backlash is accommodated for, screw height is expected to be accurate to within 17 nm.

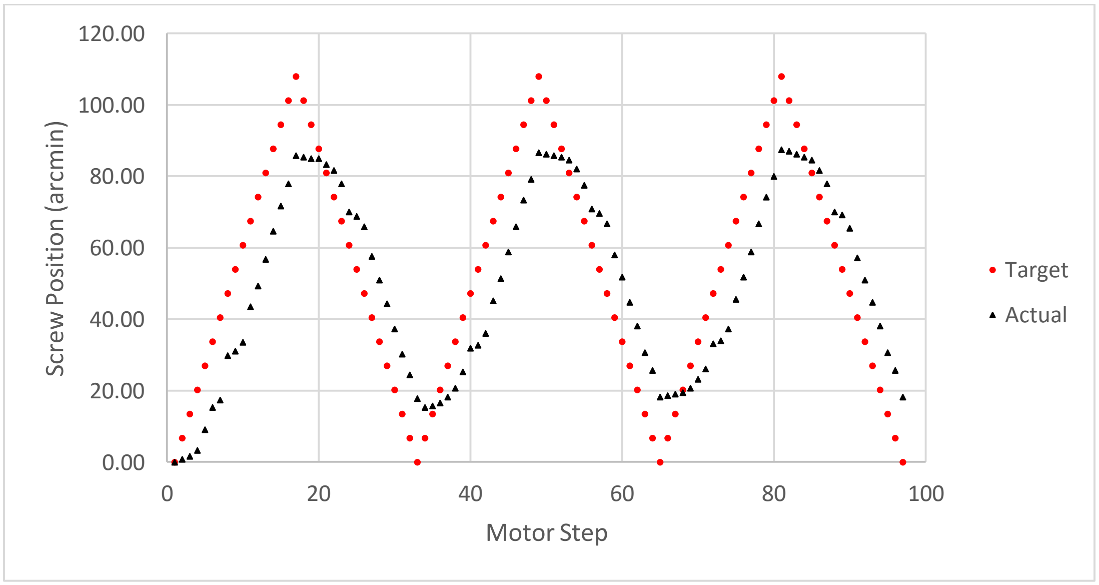


Figure 19: A4988 microstepping test results

This test was an important step in determining the backlash of the fine adjustor screw thread under the mirror's weight. Not only was the measured backlash reasonably small, but it was also consistent. The laser displacement test worked quite well, but results would have been easier to record using a longer room.

Mirror Phasing: Coarse Positioning

The finished sparse mirror array, seen in Figure 15, was tested using a webcam and a bright white LED. The test was performed to determine the capability of the active optics actuators. Any issues in the assembly needed to be found and corrected during this step before precision tests were to take place. To isolate a mirror, black dust blocking buckets were placed on top of the other mirrors. Two mirrors were individually positioned such that the LED circuit was in focus on the webcam, seen in Figure 20. Then, a roughly 100 µm pinhole was put onto the LED to simulate a bright star image. Both mirrors were then refocused and aligned. The webcam was limited in that it could not capture an airy disk from the LED. Chromatic aberration and air turbulence were also key factors to the image quality as seen in Figure 21.

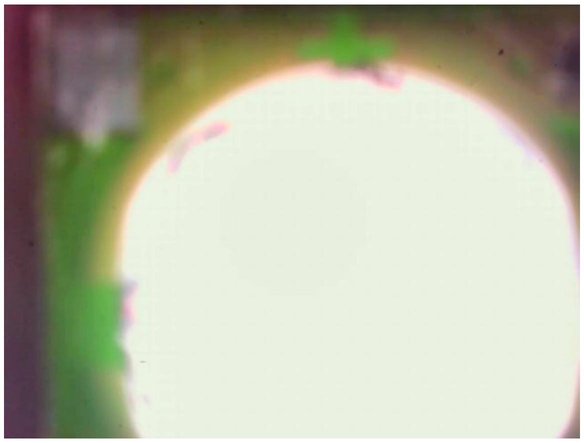


Figure 20: Focused LED image with pinhole removed

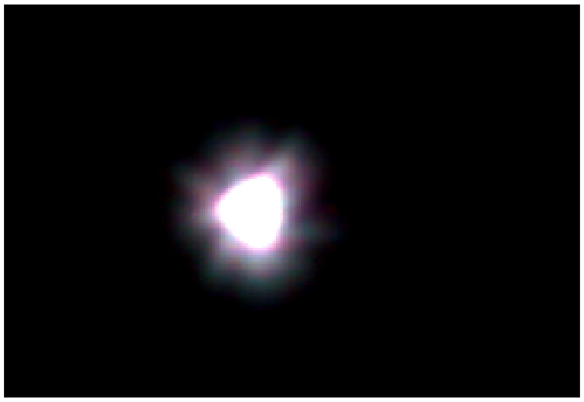


Figure 21: Webcam capture for alignment mirrors using white LED

During the initial test, some issues were discovered with the assembly. The sapphire pads needed to be more precisely placed to match the actuators. The sapphire pads were only 0.16” diameter, leaving little room for error. The Cyanoacrylate holding the sapphire pads was first dissolved using acetone, and then removed with a razor blade. The tips of the actuators were coated with a black marker and the mirror was placed on top to transfer the black ink. The pads were then adhered to the transferred marks to correct the issue.

It was initially assumed the friction between the mirror support fixture and the mirror cell would prevent mirror motion. However, the mirrors vibrated off the sapphire pads due to stepper motor vibration. A light application of Cyanoacrylate was used to hold the mirror support fixture to the mirror cell. The glue was strong enough to protect the mirror from vibration and weak enough to break away easily during disassembly.

A minor amount of shaft misalignment was detected from the actuators. During focusing, the image on camera would make circles as the shafts rotated. The effect was only noticeable for large rotations of the adjustor screw and had no noticeable impact on small image adjustments. The misalignment was not corrected during testing because it did not pose a significant issue for manual adjustment.

Mirror Phasing: Fine Adjustment

The active optics system needed to be tested to determine if the mirrors could be properly focused and aligned to one another. To accomplish this, the optical equipment and camera needed to be upgraded. The webcam was replaced with a QHY astronomy camera and the large pinhole was replaced with a 10 µm ± 1µm precision pinhole. The white LED was replaced with a green LED with an approximate bandwidth of 10%. As before, each mirror was set up, focused, and aligned using this new configuration.

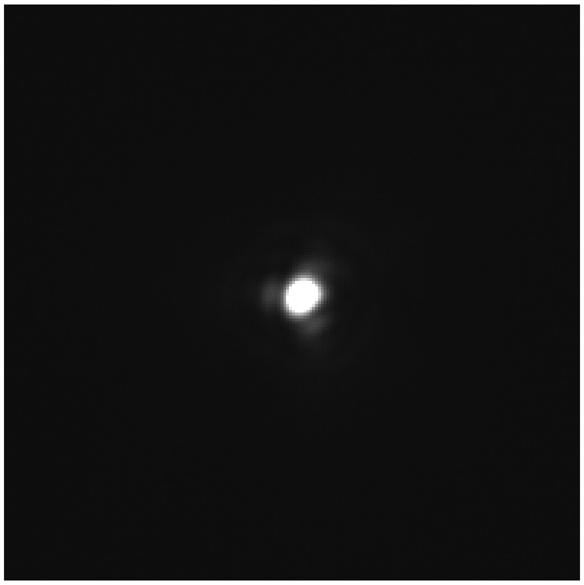


Figure 22: QHY capture for aligned mirrors using green LED

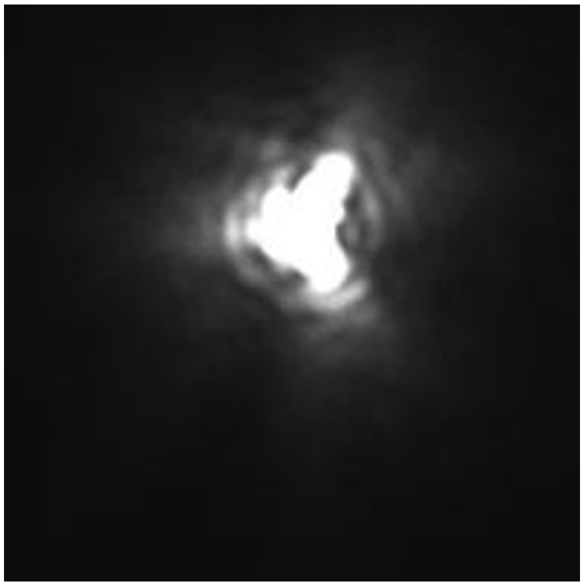


Figure 23: QHY overexposed capture

The focusing and aligning of the mirrors were successful with an airy disk present for each mirror image. However, trefoil-like aberrations were an issue as seen in Figure 22. FEA simulation estimated a whiffletree to not be necessary, but these tests determined otherwise. Additionally, air turbulence in the room created a constantly shifting image, obscuring the quality of images. This was likely due to the HVAC equipment in the lab. These issues prevented confirmation of an interference pattern necessary for determining mirror phasing. The overexposed image seen in Figure 23 highlights the detrimental effects from air turbulence.

Mirror Phasing: Searching for Interference Patterns

Once the images were aligned and focused, the main goal of the project was complete. However, a clear interference pattern would indicate the mirrors to be in phase and ready for speckle interferometry. The goal was to find an interference pattern strong enough to show up through the air turbulence and trefoil-like aberration. But with no way to measure the position of the mirror surface, there was no suitable method to find an interference pattern.

By the recommendation of Dave Rowe, one mirror was elevated in increments of 2 microns to attempt to phase the mirrors. After each 2 micron increment, 100 high speed images were captured with a low gain and then another 100 were taken with a high gain. This was repeated until one mirror had been elevated by 12 microns. After 12 microns of actuation, the captured images showed no discernible change from Figure 22 and Figure 23. These images were then sent to Dave Rowe to analyze using speckle analysis to determine if an interference pattern was present. Unfortunately, an interference pattern could not be found due the effects of the trefoil-like aberration. This could also be a result of using an LED due to its 10% bandwidth. Using a laser as a light source would likely improve results on subsequent tests.

VI. SUMMARY AND CONCLUSIONS

The components required to build this system are relatively cheap, effective, and will hopefully lead to a fully automated sparse aperture telescope in the future. The fine adjustor screw made for a relatively lost cost actuator design with nanopositioning capability. During testing, the project had some setbacks due to deformation of the mirror and air turbulence. The mirrors were able to be properly focused and aligned using the active optics actuators, but with no whiffletree the mirrors could not be brought into phase. The actuators were able to control each mirror's tip, tilt, and elevation.

Contributions to Astronomy

The success of the active optics actuators is a significant step for the sparse aperture speckle interferometry telescope. With the completion of this prototype, it is planned to eventually build a 4 meter sparse aperture telescope. Additionally, with the addition of speckle interferometry the active optics system is planned to eventually be completely automated. This will allow the observation of binary stars to continue on a dedicated telescope. The cost of the active optics components significantly lowers the barrier of entry for large aperture segmented mirror telescopes. This will hopefully contribute to not just binary star research, but low-budget telescope projects everywhere.

Project Strengths

The active optics actuators are made using low cost parts, offering an economic solution to expensive off the shelf alternatives. With the exception of the actuator housing, each actuator is made from readily available parts. The actuator design is relatively simple, making future maintenance and repair straight forward. If a component fails, it can easily be replaced with simple tools. Assembly requires only several wrenches, screw drivers, a hex key, and Cyanoacrylate.

Project Weaknesses

The actuator housing requires a fair amount of custom machining to make. Without access to tools capable of machining steel, the housing may be costly to have produced. Some changes to the design need to be made to simplify the machining process. Additionally, it took much longer to finish machining the parts that initially anticipated, approximately 20 hours for the mirror support frame and approximately 15 hours each mirror cell’s components. However, a large portion of this time was spent developing the machining plan.

Each actuator has two primary sources of position error. The adjustor thread has mechanical backlash as discussed in Chapter V. Additionally, each adjustor screw has a varying degree of shaft misalignment in the final actuator assembly. This was not originally tested for and the effects were only observed during system testing.

Future Work

A simple, low cost whiffletree is needed before future testing can take place. Ideally, the whiffletree should have a high stiffness with low to no backlash. Alternately, an air filled bag support could be used to “float” the mirror’s weight similar to the WIYN telescope [5].

The actuator housing would benefit from some minor design changes. The side covers should be attached using another method. Eight tapped holes per housing added a lot of machining time.

During testing, vibration from the stepper motors was noticeable on the camera image. This vibration was low in amplitude and decayed quickly, posing no serious consequence other than annoyance. This is simple to resolve however, vibration reducing cork NEMA 17 gaskets are cheap and readily available. The low stiffness of the gaskets may introduce some backlash. If future tests determine this to be negligible, then these gaskets should be incorporated for future projects.

The laser cut ABS mirror support fixture fully satisfied the design requirements for this test setup. If the aperture is pivoted on an extreme angle however, the fixture may not tolerate the lateral weight of larger mirror. A thinner fixture made of steel may be needed to support the lateral weight. Additionally, the fixture may need to be redesigned to allow room for a whiffletree.

One actuator experienced 3 times its normal backlash during testing which initially made aligning the images difficult. It was determined this extra backlash was due to a loose bolt holding the stepper motor inside the actuator. Future versions should include a mechanism to prevent this. This could be a combination of initial calibration and a preventative mechanism, such as the cork dampers. Additionally, some actuators had small alignment issues with the adjustor screw. The correct alignment of the screws will be crucial to future projects, so a calibration tool is needed to correct this issue.

Conclusion

The active optics system was able to focus and align the mirrors through manual adjustment. The mirrors surfaces could not be brought close enough together to find an interference pattern, but this was due to the inability to measure the surface position of the mirror. The position resolution of the active optics system should be able to generate interference patterns for future projects. With some future project additions, this system has the potential to be completely automated. The success of the active optics actuators makes for a significant step towards a fully automated sparse aperture telescope.

BIBLIOGRAPHY

[1] J. Cheng, The Principles of Astronomical Telescope Design, New York, NY: Springer, 2009, pp. 223-286.

[2] G. R. Lemaitre, Astronomical Optics and Elasticity Theory: Active Optics Methods, Berlin: Springer, 2009, p. 90.

[3] R. Genet, "Kitt Peak Speckle Interferometry of Close Visual Binary Stars," *American Association of Variable Star Observers*, p. 2, 2014.

[4] R. Genet, "Speckle Interferometry of Close Visual Binaries," *Journal of Double Star Observations*, pp. 183-197, 2015.

[5] N. Roddier, D. Blanco, L. Globe and C. Roddier, "The WIYN telescope active optics system," *Telescope Control Systems*, pp. 4-6, 1995.

[6] L. Noethe, "Active Optics in Modern, Large Optical Telescopes," *Progress in Optics*, pp. 3-4, 7 Nov 2001.

[7] Thorlabs, "http://www.thorlabs.us/newgrouppage9.cfm?objectgroup_ID=4999," 2015. [Online]. [Accessed 10 January 2015].

[8] RepRap, "http://reprap.org/wiki/RAMPS_1.4," 23 November 2015. [Online]. [Accessed 28 November 2015].

[9] A. J. Fleming and K. K. Leang, Design, Modeling and Control of Nanopositioning Systems,

London: Springer, 2014, p. 147.

[10] C. Kerns, D. Owen, A. Raymond and S. Pordes, "Relative reflectivity of various black paints," Fermi National Accelerator Lab, Batavia, IL, 1983.

[11] NASA MASTER, "http://masterweb.jpl.nasa.gov/reference/paints.htm," 2 June 2012. [Online]. [Accessed 10 November 2015].

APPENDICES

A. Bill of Materials

Table 4: Test frame bill of materials

Vendor	Part Number	Description	PPU	QTY	Total
B&B Steel and Supply	N/A	2-1/2" x 2-1/2" x 0.12 HST 36"	$36.00	3	$108.00
B&B Steel and Supply	N/A	2-1/2" x 2-1/2" x 0.25 HR Angle 36"	$36.00	3	$108.00
B&B Steel and Supply	N/A	1/2" HR Round 24"	$11.00	1	$11.00
B&B Steel and Supply	N/A	1/4" HR Plate 12" x 24"	$30.00	1	$30.00
Ace Hardware	1602	Ultra-Flat Black Krylon Spray Paint	$3.99	1	$3.99
			Grand Total		$260.99

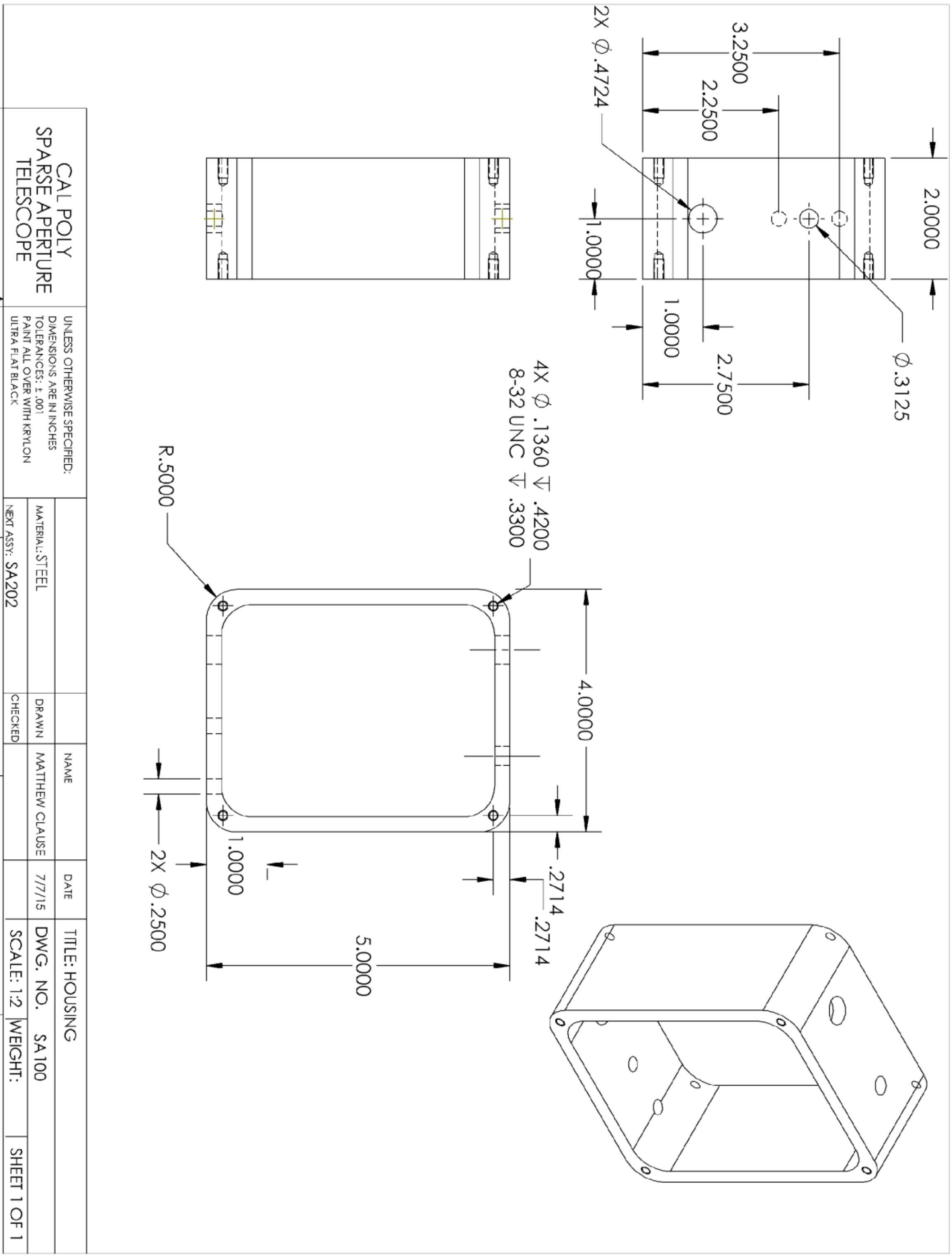
Table 5: Actuator bill of materials for three actuators

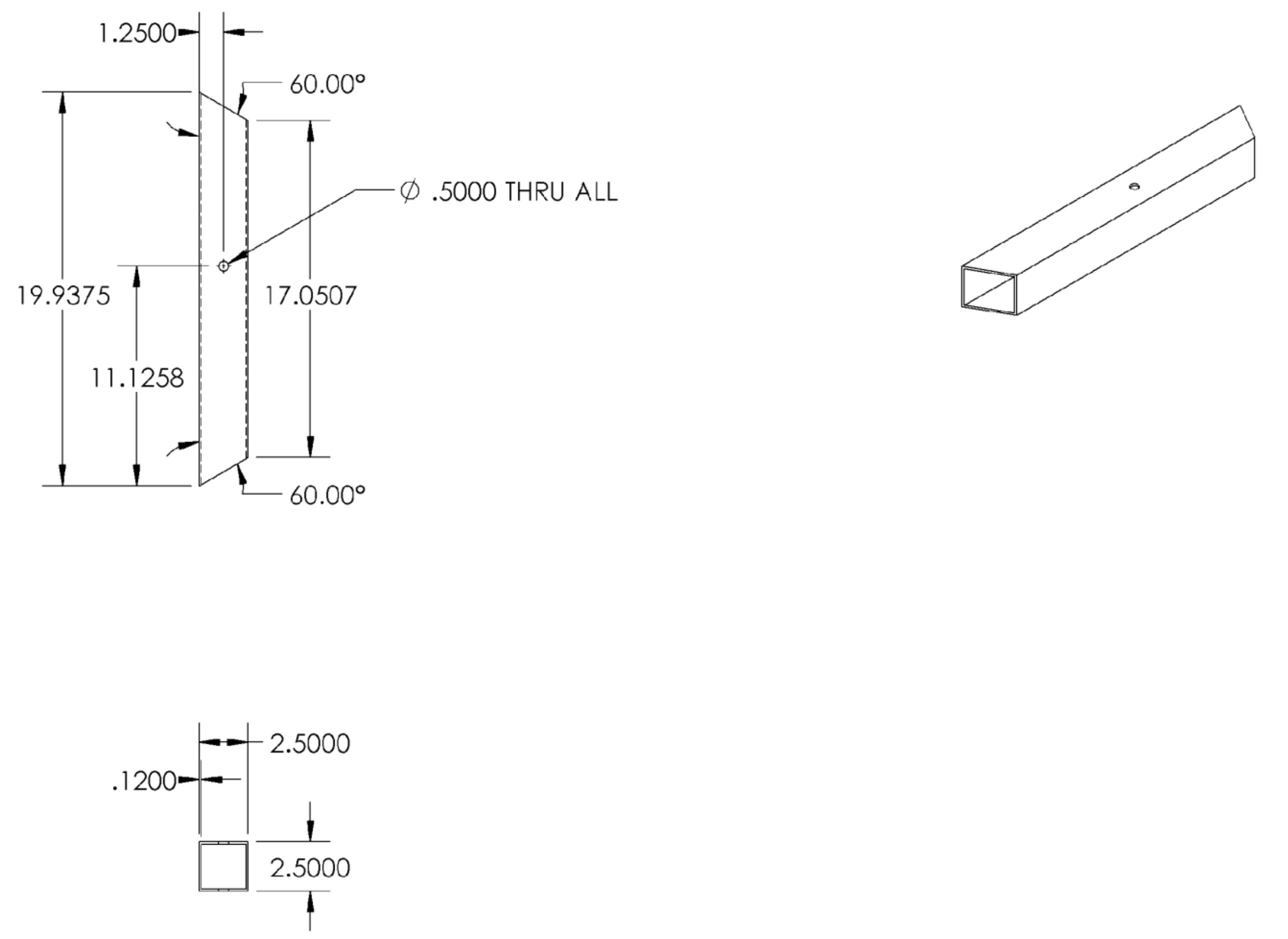
Vendor	Part Number	Description	PPU	QTY	Total
Amazon	B00F35HBEA	#8-32 Thread, 1/4" screws (100)	$4.25	1	$4.25
Amazon	B005DZZ2KK	HEX, M3 Size, 6mm Length, .5mm Pitch (100)	$7.78	1	$7.78
Amazon	a12112300ux0262	SCS12UU 12mm Metal Linear Ball Bearing	$7.51	9	$67.59
Amazon	kit1002	Linear Motion 12 mm Shaft, 13"	$8.82	5	$44.10
Amazon	308175	ABS Sheet - .060" Thick, Black, 12" x 12"	$17.70	1	$17.70
Amazon	3325	Loctite 03325 Adhesive Kit - 24 ml	$31.69	1	$31.69
B&B Steel & Supply	N/A	5" x 4" x 0.25" HST 24"	$35.00	1	$35.00
ESG	251392777142	Shaft Coupler - 1/4" to 5mm (x3)	$10.25	3	$30.75
Kozak	TSB250-254-2000/625-AA	1/4-254 TPI screw and bushing matched set	$25.32	9	$227.88
Pololu	2267	NEMA-17 Bipolar 42mm Stepper	$16.95	9	$152.55
Pololu	2266	Pololu Stamped Aluminum L-Bracket for NEMA 17	$3.95	9	$35.55
			Grand Total		$654.84

Table 6: Electronics bill of materials for three actuators

Vendor	Part Number	Description	PPU	QTY	Total
Amazon	B00D7CWSCG	12v 30a Dc Universal Regulated Switching Power Supply	$23.97	1	$23.97
Amazon	P007-006	Power Cord 14AWG 15A	$7.29	1	$7.29
Amazon	a14061000ux0612	2.54mm 3 Pin F/F Jumper Wire Connector (5)	$5.97	2	$11.94
Ebay	271864081924	MEGA2560,RAMPS1.4,3D PRINTER BOARD & 5PCS A4988	$34.50	3	$103.50
Ebay	V-155-1C25 SPDT	Momentary Limit Micro Switch Snap Action Switch Roller	$1.59	9	$14.31
Pololu	2006	Pre-crimped 50-Piece Rainbow Wires, 24"	$24.95	1	$24.95
Pololu	1903	0.1" (2.54mm) Crimp Connector Housing: 1x4-Pin (10)	$0.59	1	$0.59
			Grand Total		$186.55

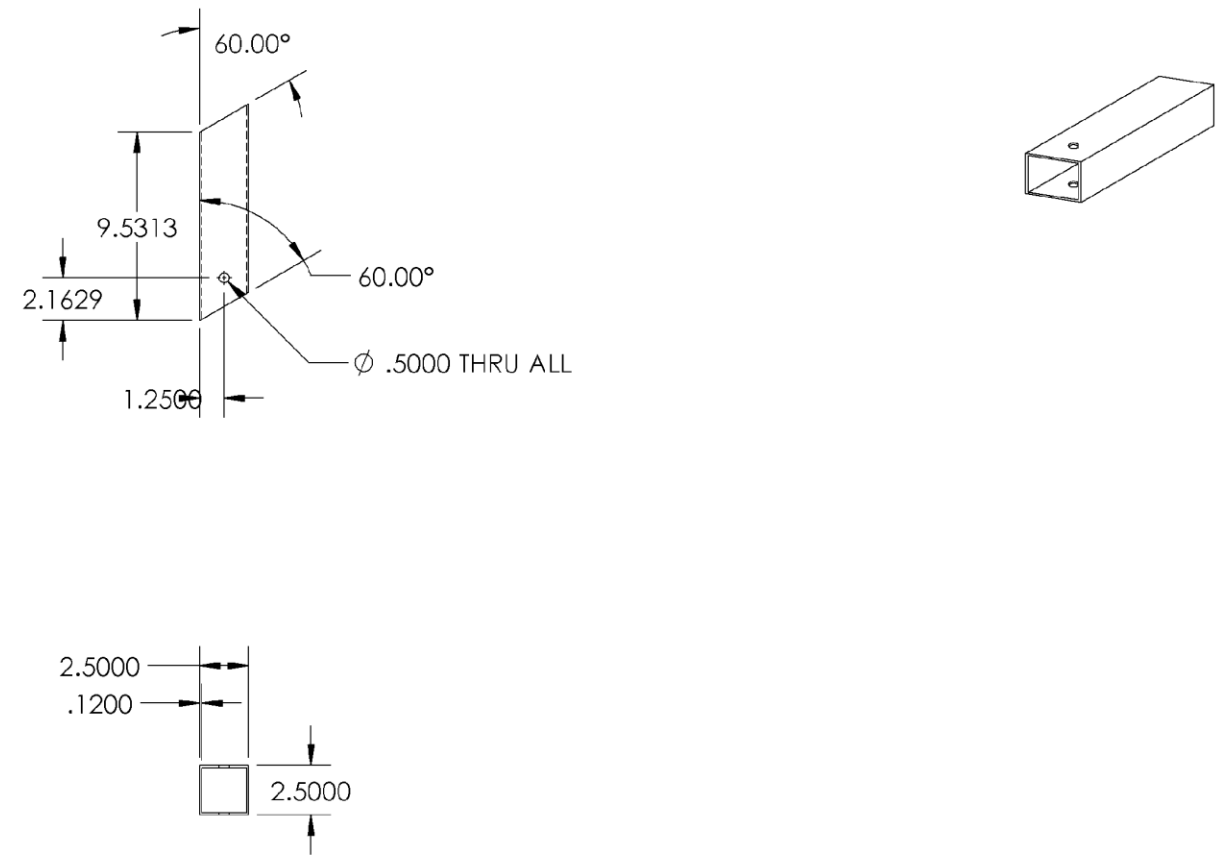
B. Part Drawings





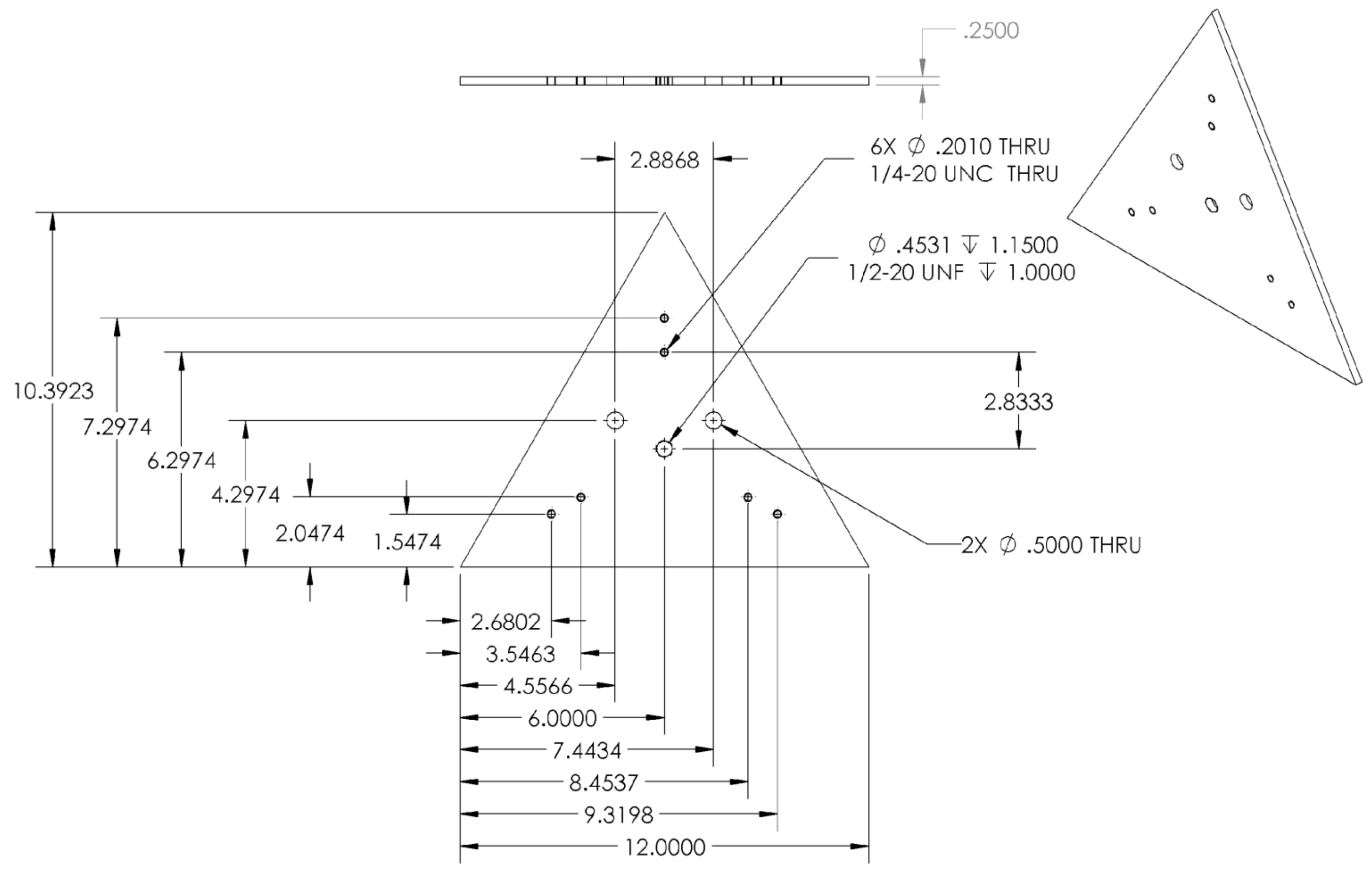
CAL POLY SPARSE APERTURE TELESCOPE	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: ± .031		NAME	DATE	TITLE: MIRROR FRAME LONG SUPPORT
		MATERIAL: STEEL	DRAWN MATTHEW CLAUSE	7/7/15	DWG. NO. SA101
		NEXT ASSY: SA201	CHECKED		SCALE: 1:8 WEIGHT: SHEET 1 OF 1

5 4 3 2 1



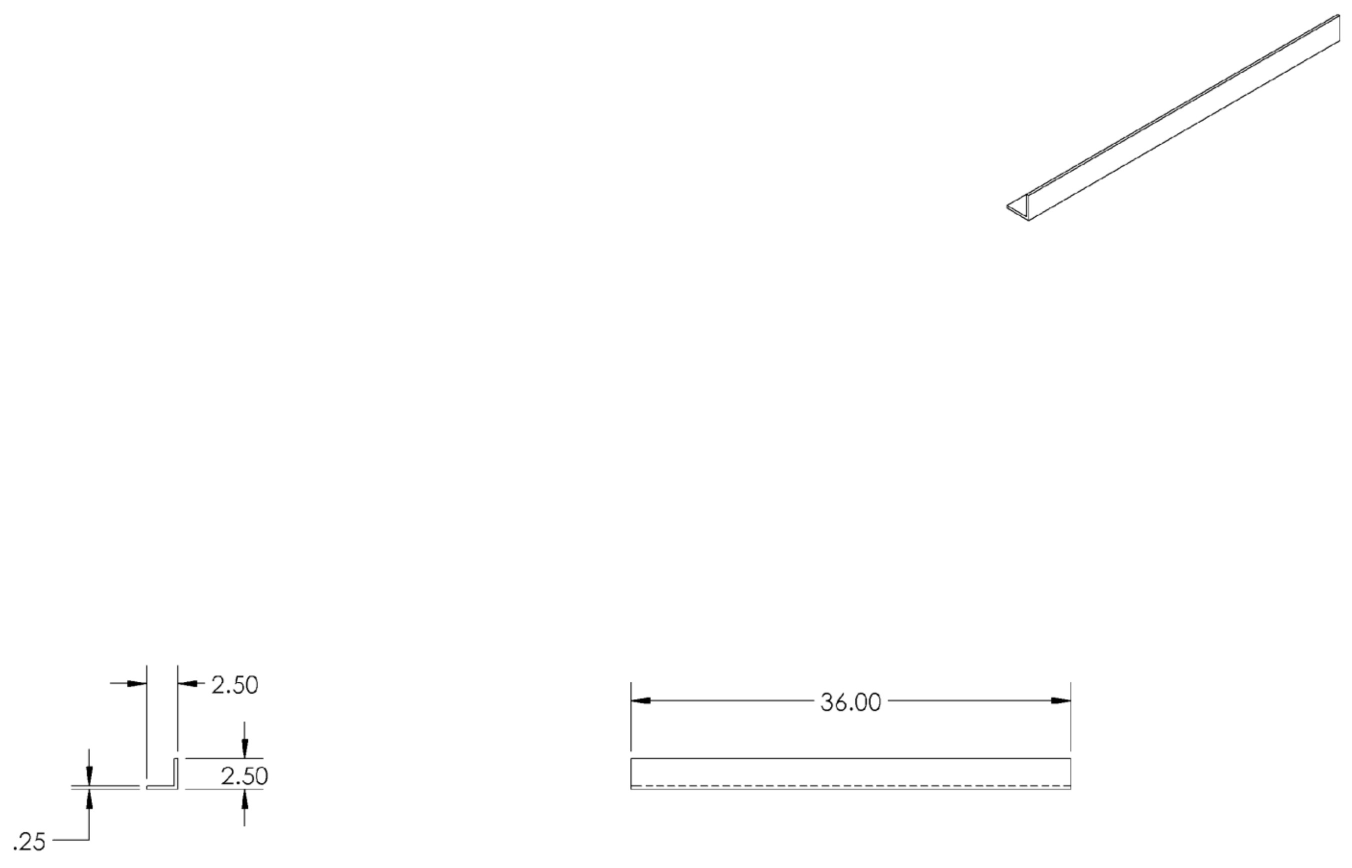
CAL POLY SPARSE APERTURE TELESCOPE	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: ± .016			NAME	DATE	TITLE: MIRROR FRAME SHORT SUPPORT
		MATERIAL: STEEL	DRAWN	MATTHEW CLAUSE	7/7/15	DWG. NO. SA102
		NEXT ASSY: SA201	CHECKED			SCALE: 1:8 WEIGHT: SHEET 1 OF 1

5 4 3 2 1



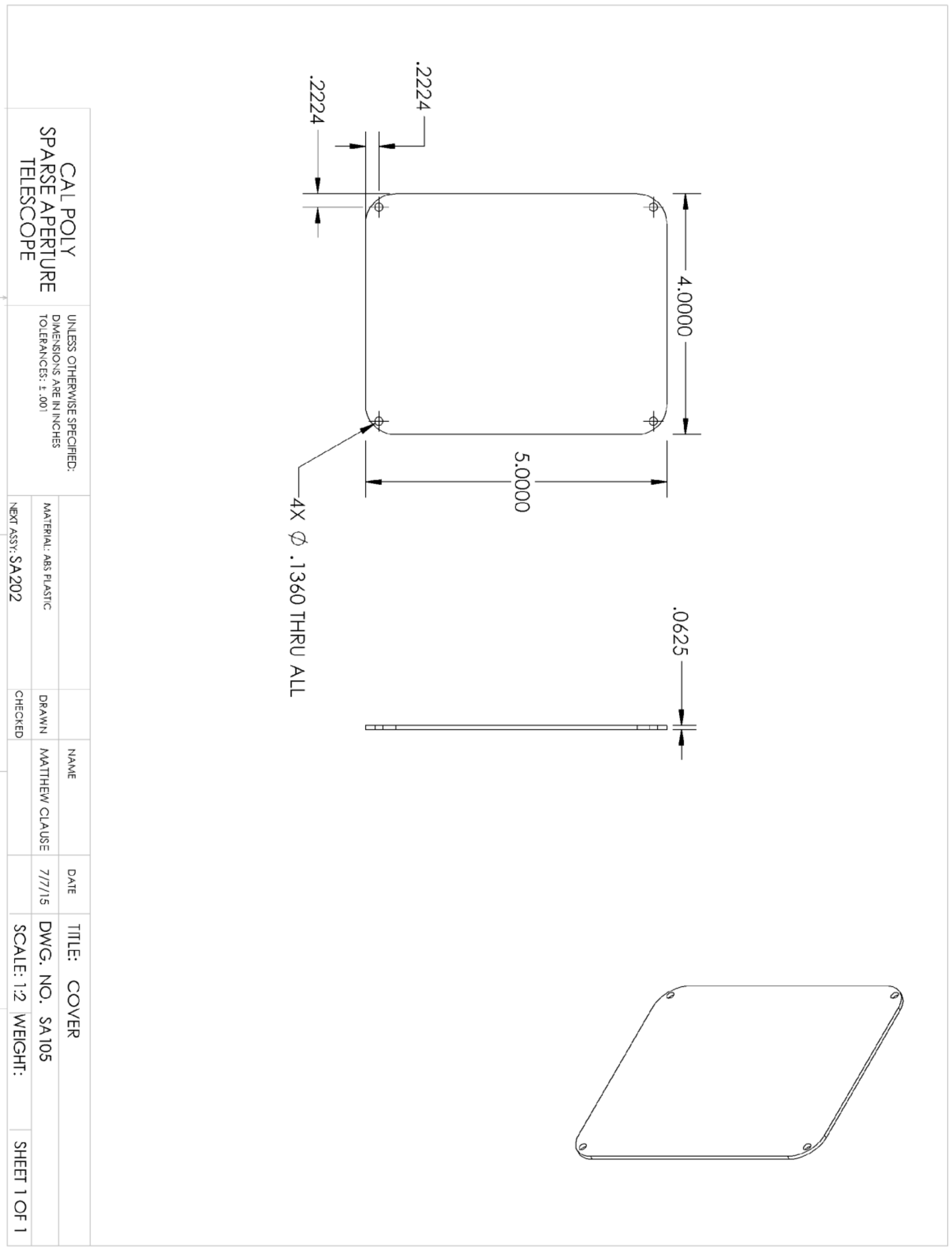
CAL POLY SPARSE APERTURE TELESCOPE	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: ± .001 PAINT ALL OVER WITH KRYLON ULTRA FLAT BLACK			NAME	DATE	TITLE: ACTUATOR PLATE
		MATERIAL: STEEL	DRAWN	MATTHEW CLAUSE	7/7/15	DWG. NO. SA103
		NEXT ASSY: SA301	CHECKED			SCALE: 1:4 WEIGHT: SHEET 1 OF 1

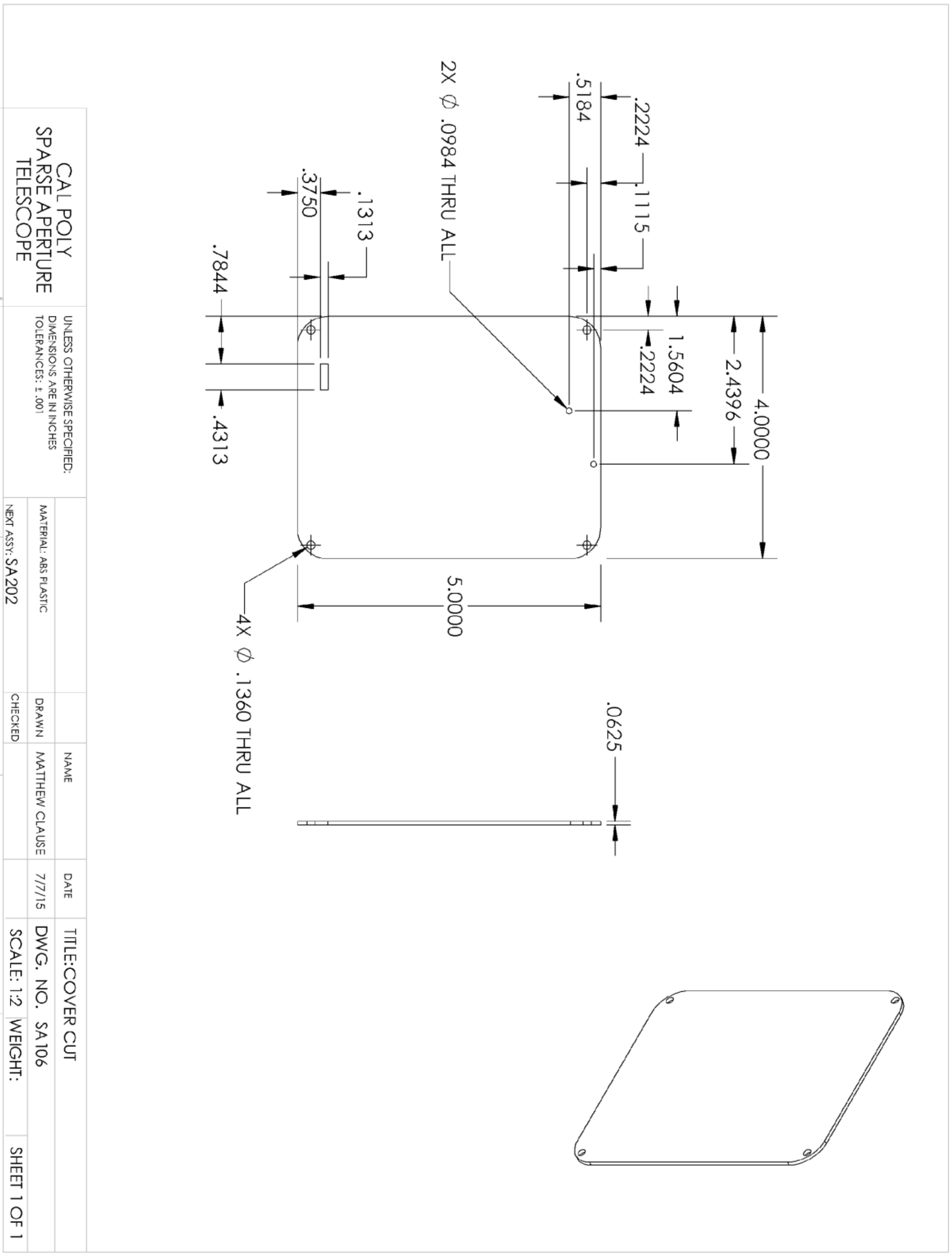
5 4 3 2 1



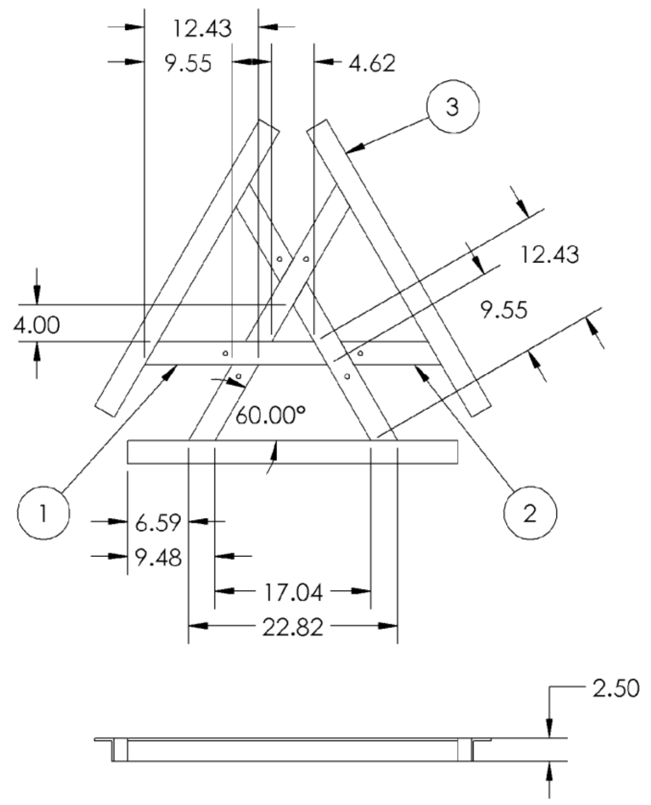
CAL POLY SPARSE APERTURE TELESCOPE	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: ± .001			NAME	DATE	TITLE: SOLID ANGLE BAR
		MATERIAL: STEEL	DRAWN	MATTHEW CLAUSE	7/7/15	DWG. NO. SA104
		NEXT ASSY:	CHECKED			SCALE: 1:12 WEIGHT: SHEET 1 OF 1

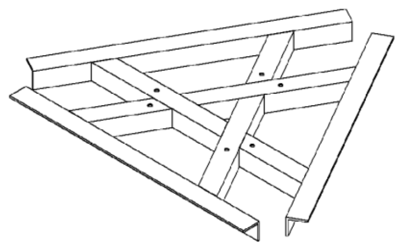
5 4 3 2 1


.2224
.2224
4.0000
5.0000
4X ∅ .1360 THRU ALL
.0625
CAL POLY SPARSE APERTURE TELESCOPE
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: ±.001
MATERIAL: ABS PLASTIC
NEXT ASSY: SA202
NAME
DATE
DRAWN
MATTHEW CLAUSE
7/7/15
CHECKED
TITLE: COVER
DWG. NO. SA105
SCALE: 1:2
WEIGHT:
SHEET 1 OF 1
5
4
3
2
1


2X ∅ .0984 THRU ALL
.5184
.2224
.1115
.3750
.1313
.7844
1.5604
.2224
2.4396
4.0000
.4313
5.0000
4X ∅ .1360 THRU ALL
.0625
CAL POLY SPARSE APERTURE TELESCOPE
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: ±.001
MATERIAL: ABS PLASTIC
NEXT ASSY: SA202
NAME
DATE
DRAWN
MATTHEW CLAUSE
7/7/15
CHECKED
TITLE:COVER CUT
DWG. NO. SA106
SCALE: 1:2
WEIGHT:
SHEET 1 OF 1
5
4
3
2
1

ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	SA101	SQUARE STEEL PIPE - LONG	3
2	SA102	SQUARE STEEL PIPE - SHORT	3
3	SA104	SOLID ANGLE BAR	3

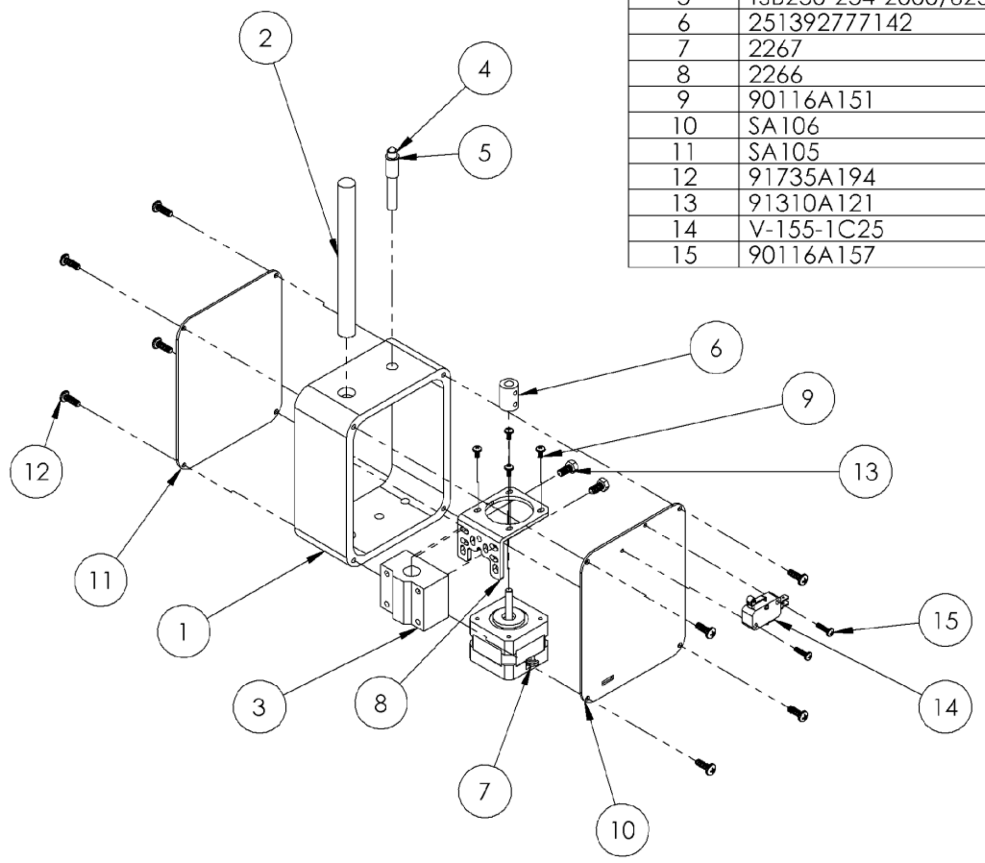


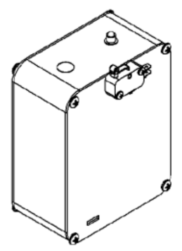


CAL POLY SPARSE APERTURE TELESCOPE	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: ± .031 PAINT ALL OVER WITH KRYLON ULTRA FLAT BLACK			NAME	DATE	TITLE: FRAME
		MATERIAL: STEEL	DRAWN	MATTHEW CLAUSE	7/7/15	DWG. NO. SA201
		NEXT ASSY: SA400	CHECKED			SCALE: 1:16 WEIGHT: SHEET 1 OF 1

5 4 3 2 1

ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	SA100	HOUSING	1
2	KIT1002	CHROME HARDENED SHAFT	1
3	A12112300UX0262	SCS12LUU LINEAR BEARING	1
4	TSB250-254-2000/625-AA	FINE ADJUSTOR SCREW	1
5	TSB250-254-2000/625-AA	BRONZE BUSHING	1
6	251392777142	SOLID COUPLER	1
7	2267	NEMA 17 STEPPER MOTOR	1
8	2266	NEMA 17 MOUNTING BRACKET	1
9	90116A151	M3 5MM HEX SCREW	4
10	SA106	COVER WITH WIRE HOLES	1
11	SA105	COVER	1
12	91735A194	#8-1/4 BUTTON HEAD SCREW	8
13	91310A121	M5 10MM BOLT	2
14	V-155-1C25	MircoSwitch	1
15	90116A157	M3 12MM BUTTON HEAD SCREW	2





CAL POLY SPARSE APERTURE TELESCOPE	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES		NAME	DATE	TITLE: ACTUATOR
		MATERIAL:	DRAWN MATTHEW CLAUSE	7/7/15	DWG. NO. SA202
		NEXT ASSY: SA301	CHECKED		SCALE: 1:4 WEIGHT: SHEET 1 OF 1

5 4 3 2 1

C. Test Data

Fine adjustor screw position test results, measured across 8.29 meters.

The A4988 stepper driver was used with 1/16th microstepping and supplied 12 V.

A 5 lb load was applied to simulate mirror weight.

Step	Target (Deg)	Target (mm)	Angle (Deg)	Actual (mm)	Tip Height (nm)
0	0.00	0.00	0.000	118	0.00
1	0.11	31.25	0.014	120	3.84
2	0.23	62.50	0.028	122	7.68
3	0.34	93.75	0.055	126	15.36
4	0.45	125.00	0.152	140	42.23
5	0.56	156.25	0.256	155	71.02
6	0.68	187.50	0.290	160	80.62
7	0.79	218.75	0.498	190	138.21
8	0.90	250.00	0.518	193	143.96
9	1.01	281.25	0.560	199	155.48
10	1.13	312.50	0.726	223	201.55
11	1.24	343.75	0.822	237	228.43
12	1.35	375.00	0.947	255	262.98
13	1.46	406.25	1.078	274	299.46
14	1.58	437.50	1.196	291	332.10
15	1.69	468.75	1.299	306	360.90
16	1.80	500.00	1.431	325	397.38
15	1.69	468.75	1.424	324	395.46
14	1.58	437.50	1.417	323	393.54
13	1.46	406.25	1.417	323	393.54
12	1.35	375.00	1.389	319	385.86
11	1.24	343.75	1.361	315	378.18
10	1.13	312.50	1.299	306	360.90
9	1.01	281.25	1.168	287	324.42
8	0.90	250.00	1.147	284	318.66
7	0.79	218.75	1.099	277	305.22
6	0.68	187.50	0.961	257	266.82
5	0.56	156.25	0.850	241	236.11
4	0.45	125.00	0.739	225	205.39
3	0.34	93.75	0.622	208	172.76
2	0.23	62.50	0.504	191	140.12
1	0.11	31.25	0.408	177	113.25
0	0.00	0.00	0.297	161	82.54
1	0.11	31.25	0.256	155	71.02
2	0.23	62.50	0.263	156	72.94
3	0.34	93.75	0.276	158	76.78
4	0.45	125.00	0.304	162	84.46

5	0.56	156.25	0.346	168	95.98
6	0.68	187.50	0.422	179	117.09
7	0.79	218.75	0.532	195	147.80
8	0.90	250.00	0.546	197	151.64
9	1.01	281.25	0.601	205	167.00
10	1.13	312.50	0.753	227	209.23
11	1.24	343.75	0.857	242	238.03
12	1.35	375.00	0.981	260	272.58
13	1.46	406.25	1.099	277	305.22
14	1.58	437.50	1.223	295	339.78
15	1.69	468.75	1.320	309	366.66
16	1.80	500.00	1.444	327	401.22
15	1.69	468.75	1.437	326	399.30
14	1.58	437.50	1.431	325	397.38
13	1.46	406.25	1.424	324	395.46
12	1.35	375.00	1.410	322	391.62
11	1.24	343.75	1.368	316	380.10
10	1.13	312.50	1.292	305	358.98
9	1.01	281.25	1.182	289	328.26
8	0.90	250.00	1.161	286	322.50
7	0.79	218.75	1.113	279	309.06
6	0.68	187.50	0.967	258	268.74
5	0.56	156.25	0.864	243	239.95
4	0.45	125.00	0.746	226	207.31
3	0.34	93.75	0.636	210	176.60
2	0.23	62.50	0.511	192	142.04
1	0.11	31.25	0.428	180	119.01
0	0.00	0.00	0.304	162	84.46
1	0.11	31.25	0.311	163	86.38
2	0.23	62.50	0.318	164	88.30
3	0.34	93.75	0.325	165	90.22
4	0.45	125.00	0.346	168	95.98
5	0.56	156.25	0.387	174	107.49
6	0.68	187.50	0.435	181	120.93
7	0.79	218.75	0.553	198	153.56
8	0.90	250.00	0.567	200	157.40
9	1.01	281.25	0.622	208	172.76
10	1.13	312.50	0.760	228	211.15
11	1.24	343.75	0.864	243	239.95
12	1.35	375.00	0.981	260	272.58
13	1.46	406.25	1.113	279	309.06
14	1.58	437.50	1.237	297	343.62
15	1.69	468.75	1.334	311	370.50
16	1.80	500.00	1.458	329	405.06
15	1.69	468.75	1.451	328	403.14
14	1.58	437.50	1.437	326	399.30
13	1.46	406.25	1.424	324	395.46
12	1.35	375.00	1.410	322	391.62

11	1.24	343.75	1.361	315	378.18
10	1.13	312.50	1.299	306	360.90
9	1.01	281.25	1.168	287	324.42
8	0.90	250.00	1.154	285	320.58
7	0.79	218.75	1.092	276	303.30
6	0.68	187.50	0.954	256	264.90
5	0.56	156.25	0.850	241	236.11
4	0.45	125.00	0.746	226	207.31
3	0.34	93.75	0.636	210	176.60
2	0.23	62.50	0.511	192	142.04
1	0.11	31.25	0.428	180	119.01
0	0.00	0.00	0.304	162	84.46

D. Arduino Program Code

```
/*
SPARE APERTURE MIRROR CELL CONTROL PROGRAM
PROGRAMMER: MATTHEW CLAUSE
CONTACT:    MAKLAUSE@GMAIL.COM
UPDATED ON: 10/24/2015
VERSION 1.1

CHANGE LOG:
7/12/2015   MATTHEW CLAUSE    1.0
      -PROGRAM CREATED
10/23/2015  MATTHEW CLAUSE    1.1
      -ADD FUNCTION ADDED
      -STOP FUNCTION ADDED
      -ZERO FUNCTION REMOVED
      -CALIBRATE FUNCTION ADDED
      -CHANGED ARCS TO MAS
      -SCREEN UPDATES ON DISPLAY TOGGLE
      -[BUG] MOVED INVISIBLE CURSOR COMMAND FROM SYSTEM UPDATE TO STARTUP
      -CHANGED TO MULTI-STEPPER UPDATE METHOD

*/

#include <Arduino.h>

// Stepper Motor Class
class stepperMotor
{
      private:
      // Enable pin
      int enPin;
      // Step pin
      int stepPin;
      // Set direction pin
      int dirPin;
      // Limit switch pin
      int minPin;
      // Current position
      int32_t position;
      // Target position
      int32_t target;
      // Pointer to emergency stop flag
      bool* eStop;

      public:
      // Stepper motor constructor
      stepperMotor (int inEnPin, int inStepPin, int inDirPin, bool* inEStop = 
NULL, int inMinPin = NULL);
      // Step once in given direction
      void step(boolean dir);
      // Power motor on
      void on () {digitalWrite(enPin, LOW);};
      // Power motor off
      void off () {digitalWrite(enPin, HIGH);};
      // Set motor position to target position
      void zero () {position = 0; target=position;};
      // Return motor position
      int32_t getPosition () {return position;};
      // Return motor target position
      int32_t getTarget () {return target;};
      // Set target position
      void setTarget (int32_t x) {target = x;};
      // Tests limit switch if limit of travel has been reached
      bool limit () {if (minPin){return !digitalRead(minPin);}else{return
false;}};

      void operator=  (int32_t x) {target=x;};
      void operator++ () {target++;};
      void operator-- () {target--;};
      void operator++ (int) {target++;};
      void operator-- (int) {target--;};
      void operator+= (int32_t x) {target+=x;};
      void operator-= (int32_t x) {target-=x;};

      // Steps motor towards target position based on given variables
      void update (stepperMotor* stepperPtr1, stepperMotor* stepperPtr2);
      // Updates three stepper motors to minimze delay time
      friend void multiStepperUpdate (stepperMotor* stepperPtr1,
stepperMotor* stepperPtr2, stepperMotor* stepperPtr3);
};

void multiStepperUpdate (stepperMotor* stepperPtr1, stepperMotor* stepperPtr2,
stepperMotor* stepperPtr3);

void updateInput ();
void updateMainProgram ();
void actToMirror (int32_t x, int32_t y, int32_t z);
void mirrorToAct ();

void systemStatusMessage ();
void systemStatusUpdate ();
void systemInputMessage ();

// Variable Declaration
int xEn = 38;
int xStep = A0;
int xDir = A1;
int xMin = 3;

int yEn = A2;
int yStep = A6;
int yDir = A7;
int yMin = 14;

int zEn = A8;
int zStep = 46;
int zDir = 48;
int zMin = 18;

int32_t tip = 0;
int32_t tilt = 0;
int32_t elev = 0;
```

```
// Pauses screen update and convert update
bool pauseUpdate = false;
// Toggles whether to display target or actual motor position
bool displayTarget = false;
// Emergency stop which turns all motors off
bool eStop = true;

// System state for main program
uint8_t state = 0;
// Updates variables on the display
bool flagUpdate = false;
// Converts the tip, tilt, elev variables
bool flagConvert = false;
// Calibrates the mirror position against the three limit switches
bool flagCalibrate = false;

// Distance from Y to Z in nm (5in/25.4mm)
static const uint32_t L1 = 127000000;
// Distance from X to midpoint of Y and Z in nm (5.768in/25.4mm)
static const uint32_t L2 = 146507200;
// Conversion factor from radians to arc seconds
static const uint32_t K = 206265;
// This constant sets the thread pitch in nm
// static const uint32_t threadPitch = 100000;
// This constant sets the steps per rev
// static const uint8_t stepsPerRev = 200;
// This constant sets the microstepping reduction
// static const uint8_t stepReduction = 16;

// Input data passed to main program
uint8_t inByte = 0;

// Three stepper motors to be controlled
stepperMotor xStepper(xEn, xStep, xDir, &eStop, xMin);
stepperMotor yStepper(yEn, yStep, yDir, &eStop, yMin);
stepperMotor zStepper(zEn, zStep, zDir, &eStop, zMin);

//=============================================================================
============================

// Initialize Program
void setup() {
      // open the serial port at 9600 bps:
      Serial.begin(9600);
}

void loop() {
    // Check for serial input
    updateInput ();
    // Update program state
    updateMainProgram();

    // Update serial output
    if (flagConvert) {
        if (displayTarget) {
            actToMirror (xStepper.getTarget(), yStepper.getTarget(),
zStepper.getTarget());
        } else {
            actToMirror (xStepper.getPosition(), yStepper.getPosition(),
zStepper.getPosition());
        }
    }
    if (flagUpdate) {
        systemStatusUpdate();
    }

    // Update steppers
      //xStepper.update (&yStepper, &zStepper);
      //yStepper.update (&zStepper, &xStepper);
      //zStepper.update (&xStepper, &yStepper);
      multiStepperUpdate (&xStepper, &yStepper, &zStepper);
    // delay(20);
}

//============================================================================
============================

// Stepper motor constructor
stepperMotor::stepperMotor (int inEnPin, int inStepPin, int inDirPin, bool*
inEStop, int inMinPin) {
            enPin = inEnPin;
            stepPin = inStepPin;
            dirPin = inDirPin;
            eStop = inEStop;
            minPin = inMinPin;
            position = 0;
            target = 0;

            pinMode(enPin, OUTPUT);
            pinMode(stepPin, OUTPUT);
            pinMode(dirPin, OUTPUT);
            pinMode(minPin, INPUT);

            this->off();
}

// Steps the motor in the given direction and delays. This delay is
determined by the A4988 motor driver datasheet.
void stepperMotor::step (boolean dir) {
      // Set motor direction
      digitalWrite(dirPin,dir);
    // Raise step pin giving time for IC to update
      digitalWrite(stepPin, HIGH);
      delayMicroseconds(200);
      digitalWrite(stepPin, LOW);
      delayMicroseconds(200);

      if (dir) {
        this->position++;
      } else {
        this->position--;
      }
```

```
}

// Updates stepper motor position. If the limit switch is depressed, the
motor will instead shut off. eStop will cause the motor to power off. If
flagCalibrate is set and all three limit switches are depressed, the motor
position will zero.
void stepperMotor::update (stepperMotor* stepperPtr1, stepperMotor*
stepperPtr2) {
      if ( this->limit() && stepperPtr1->limit() && stepperPtr2->limit() &&
flagCalibrate) {
        this->zero();
        flagCalibrate = false;
    }

      if (!(*eStop) && (position < target) && (stepperPtr1->getPosition() <
stepperPtr1->getTarget()) && (stepperPtr2->getPosition() < stepperPtr2-
>getTarget()) ) {
          this->on();
            this->step(true);
    } else if (!(*eStop) && (position < target) && (!stepperPtr1->limit()) &&
(!stepperPtr2->limit())) {
            this->on();
            this->step(true);
      } else if (!(*eStop) && (position > target) && (!this->limit())) {
            this->on();
            this->step(false);
      } else {
            this->off();
      }
}

// Updates three stepper motors simultaneously to minimze delay time
void multiStepperUpdate (stepperMotor* stepperPtr1, stepperMotor* stepperPtr2,
stepperMotor* stepperPtr3) {
    static bool update = false;

    // Check for bottom limits
    if ( stepperPtr1->limit() && stepperPtr2->limit() && stepperPtr3->limit()
&& flagCalibrate) {
        stepperPtr1->zero();
        stepperPtr2->zero();
        stepperPtr3->zero();
        flagCalibrate = false;
    }

    // Start 1st Stepper Update
    if ( (!eStop) && (stepperPtr1->getPosition() < stepperPtr1->getTarget())
&& (stepperPtr2->getPosition() < stepperPtr2->getTarget()) && (stepperPtr3-
>getPosition() < stepperPtr3->getTarget()) ) {
          stepperPtr1->on();
            digitalWrite(stepperPtr1->dirPin,true);
        digitalWrite(stepperPtr1->stepPin, HIGH);
        stepperPtr1->position++;
        update = true;
    } else if ( (!eStop) && (stepperPtr1->getPosition() < stepperPtr1-
>getTarget()) && (!stepperPtr2->limit()) && (!stepperPtr3->limit())) {
            stepperPtr1->on();
            digitalWrite(stepperPtr1->dirPin,true);
        digitalWrite(stepperPtr1->stepPin, HIGH);
        stepperPtr1->position++;
        update = true;
      } else if ( (!eStop) && (stepperPtr1->getPosition() > stepperPtr1-
>getTarget()) && (!stepperPtr1->limit())) {
            stepperPtr1->on();
            digitalWrite(stepperPtr1->dirPin,false);
        digitalWrite(stepperPtr1->stepPin, HIGH);
        stepperPtr1->position--;
        update = true;
      } else {
            stepperPtr1->off();
      }

      // Start 2nd Stepper Update
     if ( (!eStop) && (stepperPtr2->getPosition() < stepperPtr2->getTarget())
&& (stepperPtr1->getPosition() < stepperPtr1->getTarget()) && (stepperPtr3-
>getPosition() < stepperPtr3->getTarget()) ) {
          stepperPtr2->on();
            digitalWrite(stepperPtr2->dirPin,true);
        digitalWrite(stepperPtr2->stepPin, HIGH);
        stepperPtr2->position++;
        update = true;
    } else if ( (!eStop) && (stepperPtr2->getPosition() < stepperPtr2-
>getTarget()) && (!stepperPtr1->limit()) && (!stepperPtr3->limit())) {
            stepperPtr2->on();
            digitalWrite(stepperPtr2->dirPin,true);
        digitalWrite(stepperPtr2->stepPin, HIGH);
        stepperPtr2->position++;
        update = true;
      } else if ( (!eStop) && (stepperPtr2->getPosition() > stepperPtr2-
>getTarget()) && (!stepperPtr2->limit())) {
            stepperPtr2->on();
            digitalWrite(stepperPtr2->dirPin,false);
        digitalWrite(stepperPtr2->stepPin, HIGH);
        stepperPtr2->position--;
        update = true;
      } else {
            stepperPtr2->off();
      }

      // Start 3rd Stepper Update
     if ( (!eStop) && (stepperPtr3->getPosition() < stepperPtr3->getTarget())
&& (stepperPtr2->getPosition() < stepperPtr2->getTarget()) && (stepperPtr1-
>getPosition() < stepperPtr1->getTarget()) ) {
          stepperPtr3->on();
            digitalWrite(stepperPtr3->dirPin,true);
        digitalWrite(stepperPtr3->stepPin, HIGH);
        stepperPtr3->position++;
        update = true;
    } else if ( (!eStop) && (stepperPtr3->getPosition() < stepperPtr3-
>getTarget()) && (!stepperPtr2->limit()) && (!stepperPtr1->limit())) {
            stepperPtr3->on();
            digitalWrite(stepperPtr3->dirPin,true);
        digitalWrite(stepperPtr3->stepPin, HIGH);
        stepperPtr3->position++;
        update = true;
      } else if ( (!eStop) && (stepperPtr3->getPosition() > stepperPtr3-
>getTarget()) && (!stepperPtr3->limit())) {
            stepperPtr3->on();
            digitalWrite(stepperPtr3->dirPin,false);
        digitalWrite(stepperPtr3->stepPin, HIGH);
        stepperPtr3->position--;
        update = true;
      } else {
            stepperPtr3->off();
      }

    // If updating, delay then lower step pins
    if (update) {
        delayMicroseconds(200);
        digitalWrite(stepperPtr1->stepPin, LOW);
        digitalWrite(stepperPtr2->stepPin, LOW);
        digitalWrite(stepperPtr3->stepPin, LOW);
        delayMicroseconds(200);
        update = false;
    }
}

//=============================================================================
============================

// Checks for serial input to pass to main control program
void updateInput () {
    if(Serial.available() > 0)
    {
        inByte = Serial.read();
    }
}

// Manages program updates, controls system state, and handles input
void updateMainProgram () {
    static char param = 0;
    static bool sign = true;
    static char charArray[10] = {0,0,0,0,0,0,0,0,0,0};
    static uint8_t charPtr = 0;
    static int64_t temp = 0;
    static uint8_t updateCounter = 0;

    // The following interface utilizes ANSI escape commands
    switch (state) {
            // Initial entry state
        case (0):
            // Set cursor to invisible using ANSI escape command
            Serial.write(27);
            Serial.print("[?25l");     // Invisible cursor
            state = 1;
            break;

            // Display screen main display and move to state 2
        case (1):
            // Update screen with main display message
            systemStatusMessage();
```

```
          state = 2;
          break;

          // Manages screen update and waits for input before moving to
state 3
        case (2):
          //
          if ((updateCounter >= 10) && !pauseUpdate) {
              flagUpdate = true;
              flagConvert = true;
              updateCounter = 0;
          } else {
              updateCounter++;
          }

          if(inByte){
              state = 3;
          }
          break;

          // Determines which method to perform for a  given input
        case(3):
          // X Stepper
          if (inByte=='Q' || inByte=='q') {
              if (xStepper.getPosition() < 300000000) {
              xStepper++;
              }
          } else if (inByte=='W' || inByte=='w') {
              if (xStepper.getPosition() > -300000000) {
              xStepper--;
              }
          } else if (inByte=='E' || inByte=='e') {
              param = 'E';
              state = 4;
              systemInputMessage();
              break;
          } else if (inByte=='R' || inByte=='r') {
              param = 'R';
              state = 4;
              systemInputMessage();
              break;
          } else if (inByte=='T' || inByte=='t') {
              xStepper.setTarget(xStepper.getPosition());
          }
          // Y Stepper
          else if (inByte=='A' || inByte=='a') {
              if (yStepper.getPosition() < 300000000) {
              yStepper++;
              }
          } else if (inByte=='S' || inByte=='s') {
              if (yStepper.getPosition() > -300000000) {
              yStepper--;
              }
          } else if (inByte=='D' || inByte=='d') {
              param = 'D';
              state = 4;
              systemInputMessage();
```

```
    break;
} else if (inByte=='F' || inByte=='f') {
    param = 'F';
    state = 4;
    systemInputMessage();
    break;
} else if (inByte=='G' || inByte=='g') {
    yStepper.setTarget(yStepper.getPosition());
}
// Z Stepper
else if (inByte=='Z' || inByte=='z') {
    if (zStepper.getPosition() < 300000000) {
    zStepper++;
    }
} else if (inByte=='X' || inByte=='x') {
    if (zStepper.getPosition() > -300000000) {
    zStepper--;
    }
}  else if (inByte=='C' || inByte=='c') {
    param = 'C';
    state = 4;
    systemInputMessage();
    break;
} else if (inByte=='V' || inByte=='v') {
    param = 'V';
    state = 4;
    systemInputMessage();
    break;
} else if (inByte=='B' || inByte=='b') {
    zStepper.setTarget(zStepper.getPosition());
}

// Emergency stop
else if (inByte==' ') {
    eStop ^= 1;
    Serial.write(27);
    Serial.write("[13;25H");
    if (eStop) {
        Serial.write("ON ");
    } else {
        Serial.write("OFF");
    }
}

// Toggle display mode
else if (inByte=='1') {
    displayTarget ^= 1;
    Serial.write(27);
    Serial.write("[14;20H");
    if (displayTarget) {
        Serial.write("TARGET");
    } else {
        Serial.write("ACTUAL");
    }
    pauseUpdate = false;
}
```

```
        // Toggle screen update
        else if (inByte=='2') {
            pauseUpdate ^= 1;
            Serial.write(27);
            Serial.write("[15;20H");
            if (pauseUpdate) {
                Serial.write("PAUSED");
            } else {
                Serial.write("LIVE  ");
            }
        }

        // Recalibrate
        else if (inByte=='3') {
            xStepper.setTarget(-300000000);
            yStepper.setTarget(-300000000);
            zStepper.setTarget(-300000000);

            flagCalibrate = true;
        }

        // By default, clear any input recieved then move to state 2
        inByte = 0;
        state = 2;
        break;

        // Converts the ASCII input characters to binary
    case (4):
        if (inByte>='0' && inByte<='9' && charPtr < 9)
        {
            charArray[charPtr] = inByte;
            charPtr++;
            Serial.write(inByte);
        } else if (inByte == '+') {                 // Potitive
            sign = true;
            Serial.write(27);
            Serial.print(7);
            Serial.write(27);
            Serial.print("[1;22H");
            Serial.write(' ');
            Serial.write(27);
            Serial.print(8);
        } else if (inByte == '-') {                 // Negative
            sign = false;
            Serial.write(27);
            Serial.print(7);
            Serial.write(27);
            Serial.print("[1;22H");
            Serial.write('-');
            Serial.write(27);
            Serial.print(8);
        } else if (inByte == 127 && charPtr > 0) {    // Backspace
            charPtr--;
            charArray[charPtr] = 0;
            Serial.write("\b \b");
        } else if (inByte == 13 && charPtr == 0) {
            // Clean up
```

```
                state = 1;
                param = 0;
                sign = true;
                memset(charArray, 0, 10);
                temp = 0;
            } else if (inByte == 13 && charPtr > 0) {                    //
Enter
                // Add array variabls and apply sign
                temp = 0;
                for (uint32_t i = 1; charPtr > 0; i*=10) {
                    temp += (charArray[--charPtr]-'0')*i;
                }

                // Check for overflow conditions
                if (temp < 0 || temp > 300000000) {
                    temp = 300000000;
                }

                if (!sign) {
                    temp *= -1;
                }
                // Convert from nm to step position
                temp *= 4;
                temp /= 125;
                // Set position of selected motor
                switch (param) {
                    case('E'):
                        if ( ((xStepper.getPosition() + temp) >= -300000000)
&& ((xStepper.getPosition() + temp) <= 300000000) ) {
                        xStepper+=temp;
                        }
                        break;
                    case('R'):
                        xStepper=temp;
                        break;

                    case('D'):
                        if ( ((yStepper.getPosition() + temp) >= -300000000)
&& ((yStepper.getPosition() + temp) <= 300000000) ) {
                        yStepper+=temp;
                        }
                        break;
                    case('F'):
                        yStepper=temp;
                        break;

                    case('C'):
                        if ( ((zStepper.getPosition() + temp) >= -300000000)
&& ((zStepper.getPosition() + temp) <= 300000000) ) {
                        zStepper+=temp;
                        }
                        break;
                    case('V'):
                        zStepper=temp;
                        break;
                }
                // Clean up
                state = 1;
                param = 0;
                sign = true;
                memset(charArray, 0, 10);
                charPtr = 0;
                temp;
            }

            inByte = 0;
            break;

            // Error handling state
        default:
            Serial.println("Error: Interface in wrong state!");
            state = 1;
            break;
        // Interface needs to be designed
        // Add Tip, Tilt, Elevation Control
    }
}

// Converts three actuator heights to tip, tilt, elevation
void actToMirror (int32_t x, int32_t y, int32_t z) {
    int64_t temp;
    /// Warning: Calculation relies on small angle approximation!
      // Calculation is split into steps to avoid math overflow
    temp  = x;
    temp *= 2;
    temp -= y;
    temp -= z;
    temp *= 15625*K;
    temp /= L1;
    tip = -temp;

    temp  = y;
    temp -= z;
    temp *= 31250*K;
    temp /= L2;
    tilt = temp;

    temp  = x;
    temp += y;
    temp += z;
    temp *= 125;
    temp /= 3*4;
    elev = temp;

    flagConvert = false;
}

// Converts tip, tilt, elevation to three actuator heights
void mirrorToAct () {
    int64_t temp1, temp2;
    /// Warning: Calculation relies on small angle approximation!
      // Calculation is split into steps to avoid math overflow
    temp1 = tip;
    temp1 *= 2*L1;
    temp1 /= 3*K;
    temp1 += elev;
    temp1 *= 4;
    temp1 /= 125;
    xStepper = temp1;

    temp1 = 3*tilt;
    temp1 *= L2;
    temp2 = -2*tip;
    temp2 *= L1;
    temp1 += temp2;
    temp1 /= K;
    temp1 += elev;
    temp1 *= 2;
    temp1 /= 375;
    yStepper = temp1;

      temp1 = -3*tilt;
    temp1 *= L2;
    temp2 = -2*tip;
    temp2 *= L1;
    temp1 += temp2;
    temp1 /= K;
    temp1 += elev;
    temp1 *= 2;
    temp1 /= 375;
    zStepper = temp1;
}

// Main display information for display with PUTTY
void systemStatusMessage () {
      Serial.write(27);
      Serial.write("[2J");    // clear screen
      Serial.write(27);
      Serial.write("[H");     // cursor to home command

    Serial.println("Mirror Cell GAMMA Control Program, Version 1.1");

    Serial.println("X Actuator (nm): ");
    Serial.println("[Q] Inc [W] Dec [E] Add [R] Set [T] Stop");

    Serial.println("Y Actuator (nm): ");
    Serial.println("[A] Inc [S] Dec [D] Add [F] Set [G] Stop");

    Serial.println("Z Actuator (nm): ");
    Serial.println("[Z] Inc [X] Dec [C] Add [V] Set [B] Stop\n");

    Serial.println("Tip       (mas): ");
    Serial.println("Tilt      (mas): ");
    Serial.println("Elevation  (nm): \n");

    Serial.write  ("[SPACE] Emergency Stop: ");
    if (eStop) {
        Serial.println("ON ");
    } else {
        Serial.println("OFF");
    }
    Serial.write  ("[1] Display Mode:  ");
    if (displayTarget) {
        Serial.println("TARGET");
    } else {
        Serial.println("ACTUAL");
    }
    Serial.write  ("[2] Screen Update: ");
    if (pauseUpdate) {
        Serial.println("PAUSED");
    } else {
        Serial.println("LIVE");
    }
    Serial.println("[3] Calibrate");

    Serial.println("Use [ ] key to select parameter to change");

    if (displayTarget) {
        actToMirror (xStepper.getTarget(), yStepper.getTarget(),
zStepper.getTarget());
    } else {
        actToMirror (xStepper.getPosition(), yStepper.getPosition(),
zStepper.getPosition());
    }
    systemStatusUpdate ();
}

// Updates system information for display with PUTTY
void systemStatusUpdate () {
    Serial.write(27);
      Serial.write("[2;18H");     // Move cursor to position
      if (displayTarget) {
        Serial.print(xStepper.getTarget()*125/4);
      } else {
        Serial.print(xStepper.getPosition()*125/4);
      }
      Serial.write(27);
      Serial.write("[K");         // Clear line right of cursor

      Serial.write(27);
      Serial.write("[4;18H");     // Move cursor to position
      if (displayTarget) {
        Serial.print(yStepper.getTarget()*125/4);
      } else {
        Serial.print(yStepper.getPosition()*125/4);
      }
      Serial.write(27);
      Serial.write("[K");         // Clear line right of cursor

      Serial.write(27);
      Serial.write("[6;18H");     // Move cursor to position
      if (displayTarget) {
        Serial.print(zStepper.getTarget()*125/4);
      } else {
        Serial.print(zStepper.getPosition()*125/4);
      }
      Serial.write(27);
      Serial.write("[K");         // Clear line right of cursor

      Serial.write(27);
      Serial.write("[9;18H");     // Move cursor to position
      Serial.print(tip);
      Serial.write(27);
      Serial.write("[K");         // Clear line right of cursor

      Serial.write(27);
      Serial.write("[10;18H");     // Move cursor to position
      Serial.print(tilt);
      Serial.write(27);
      Serial.write("[K");         // Clear line right of cursor

      Serial.write(27);
      Serial.write("[11;18H");     // Move cursor to position
      Serial.print(elev);
      Serial.write(27);
      Serial.write("[K");         // Clear line right of cursor

    flagUpdate = false;
}

// Clears screen and prompts for input information with PUTTY
void systemInputMessage () {
    Serial.write(27);       // ESC command
    Serial.write("[2J");    // clear screen command
    Serial.write(27);       // ESC command
    Serial.write("[H");     // Home cursor
    Serial.write("Input position (nm):  ");
}
```