1. Executive summary

The Robinson Observatory Telescope Refresh is an interdisciplinary project between Mechanical Engineers, Electrical Engineers, and Computer Scientists. What once started as a project to repair the observatory telescope to its prior functioning state, has now shifted to the creation of a model. The complete project as a group is focused on the creation of sub-sub-scale model. The teams will construct a moving telescope model that will take in coordinates from a program and point to that location. The main goal of the model is to create a well documented project that works similarly to the observatory telescope to help in repair of the telescope. The telescope is very complex machine with the proper documentation to help in any repair. The model can provide test data for the observatory without the scale of the telescope or damaging the telescope any further. A second stretch goal is to create a telescope that others, willing to, can build. Our instructions in creating the telescope movement pieces and the software that accompanies would be available for anyone to view and use. As computer scientists, we are tasked with the telescope control software. Furthermore, our team will create a space photo web application based on the organizing of photos along with several observatory upgrades.

The telescope control program will be based on either an open-source program we find or one we create. The program will allow a user to select a point in the sky and then communicate with the Arduino board that will connect and control the telescope's motors and gears. During communication between the two objects, the telescope will send its current location while receiving motor movement commands. The coordinates will need to be converted to motor instructions specific to the motors we use and gears we create to allow for proper function.

The next objective of this project will be creating an application for the storing and viewing of photos taken by the observatory. Our database will organize the photos of the observatory based upon the metadata of each photo into a clean and easy to use UI. A stretch goal for this search functionality will be machine learning. Machine learning will allow the sorting of photos that do not include any location metadata. The website will also include a community outreach program.

Finally, we will perform multiple activities on the enhancements of the observatory. First off, the control computer needs to be updated to Windows 10 as its current version will soon be phased out by Microsoft. Next, the observatory wants their server reconnected to the network. Their current server has not been used in years, which may cause the need to find a new server solution. Lastly, we will establish remote access of the control computer from the iMac downstairs. We will use the remote access ability provided in the Windows OS but will thus need to install Windows 10 onto the iMac.

2. Personal Motivation

I want to embark on this project primarily for my interest in space. Growing up near the Kennedy Space Center, I was viewing the shuttle launches from a young age and dreamed upon what is out there. A sight I will always want to see firsthand has been strengthened by the pictures provided by telescopes. Telescopes provide a great photos of our universe that can't be currently reached by humans like the Hubble Telescope showcasing the Pillars of Creation or a simple home telescope to view Jupiter. This passion that has grown stronger with the advancements being made for commercial trips to space, especially the advancements by SpaceX. The technological advancements SpaceX has made with getting their boosters to land safely back at a landing pad simultaneously is one of the coolest things I have seen in the last decade. SpaceX along with Virgin Galactic has brought excitement back to space exploration. Some great memories I have include watch parties that had telescope viewings of either a launch, or eclipse, or viewing of another planet.

I had two events that greatly enhanced my passion for space. The first one happened when I was in 7th grade and our class trip was to the Kennedy Space Center. We were able to tour the facilities and sleep under one of the rockets that had on display. The trip helped gain new knowledge and greater appreciation of space. The second event happened freshman summer of high school on a trip to North Carolina. I was family on the side of a mountain in some cabins. One night the mountain staff held a viewing party. My family along with other guests were able to view the stars and use a telescope to see planets only viewable at that time of the year. The event showed me how the topic of space can bring people together.

These two events are a cornerstone to why I wanted to be apart of this project. I know firsthand how viewing parties can bring people together and the knowledge can be beneficial. Not many people own their own personal telescope so this observatory can give those interested a better way to view the sky. To know, I will have a helping hand in allowing activities to once again be performed at the observatory is exciting. The observatory can bring back “Knights Under the Stars” allowing many the current unavailable opportunity of getting to see the stars up close. The observatory could even provide new research study opportunities for those at UCF.

The next reason I chose this project was to build experiences on ideas I have yet to work with. My initial interpretation for this project was a new software needs to be created to help the telescope return to working order. I have learned how to code basic programs and taught programming stepping stones over the past few years without getting any real world experience. I have been eager to apply my knowledge to develop something that will be used past my educational career here at UCF. Furthermore, it will be applying this knowledge to help build something in an industry I am passionate about.

Another motivation for this project is the ability to leave behind a legacy. Of course, the legacy could become poor if this project isn't completed properly but with proper completion, our solution will outlast my time here at UCF. This refresh could impact this university and space exploration for the next few years and that provides a strong legacy.

As this project has morphed into the creation of a sub scale mode instead of fixing the observatory telescope, I believe its community outreach will be broader than before. The current idea is to create a sub scale model that others can build from our instructions. This could provide a new group of astronomer enthusiasts with their own personal moving telescope. The idea of this telescope's design being available to all provides new motivation for this project. I am not only helping the university and those connected to it, I am now reaching past those borders.

Next, I am excited I can use knowledge from my previous major choices here at UCF. Before declaring as a computer scientist, I had first declared for a computer engineering degree than a mechanical engineering degree than a mathematic degree. With my role in the development of telescope control software, I will be able to use some knowledge I gained from the classes I took specific to those majors like some high-tier math classes, statics, and the freshman engineering class. These classes will help in the computation of telescope movements and allow for a better understanding of what the software needs to provide for the other teams involved.

3. Telescope Control Software Options

The interdisciplinary part of this project deals focuses on the creation of a sub-scale model similar to the Robinson observatory telescope. In order for this telescope to move, we need to either find a suitable open-source software to use, create our own telescope control software, or combine a mix of both. Currently, the observatory uses SkyX Professional as it is the software provided with the Paramount telescope they purchased. The software provides control outputs to a black box on the telescope, which then sends movement operations to the gears. When the telescope was functioning properly, the software worked well for those controlling the telescope. The problem now is the telescope is not in a properly functioning state and the exact reason can't be determined. The software gives multiple error messages when in use but it can't be determined in the software is no longer properly parsing the data because of a modifications to the settings or mechanical issues. With the connection to the black box, which is provided by Bisque, the cycle of information is currently unaccessible due to lack of information. There is no documentation on the black box so it is unknown how the black box parses input given by the software and creates gear movements. In order to properly dissect this box, the time needed would take too long for the scope of this project. This is only on reason we must find a new software.

The other reason we are looking for a new software is SkyX Professional used for observatory control is placed behind a paywall. SkyX does have a free version but this versions purpose is to help teach the user on space not provide telescope control. The professional version can be bought for $329.00 on www.bisque.com. While this is a steep price to pay, we could find a way to help pay for this cost as a Senior Design project. The issue the paywall brings up is limiting the community outreach our project could have with an open source program. SkyX and its price tag could cause many people to not use our project for themselves. So with the combination of the needed black box and the price tag, we want to pursue other options. Options that are easily accessible for everyone to provide a functioning telescope control program.

Since we will not being using the SkyX software, we need to find an option that mimics many of SkyX's core features to make a transition easier if the observatory finds they would prefer our program more. At the beginning of this project when we were tasked with repairing the observatory telescope, the faculty working at the observatory were atimitment about keeping the SkyX software. We would like to show another option for telescope control while providing the observatory with the functionality they like. With SkyX being a paid software, it is more polished and refined compared to open source programs. This does not translate into a head and shoulders better program though.

An important aspect our software will not replicate is the control of the dome at the observatory. With SkyX, the observatory is able to directly connect the control of the dome with SkyX. The dome protects the telescope and only has a small portion the telescope can see out of. When tracking an object, the dome moves along with the telescope to ensure the telescope still has a clear view of the sky. Our sub-scale model will not feature a dome and furthermore, the open source programs available do not allow for the direct connection like SkyX does.

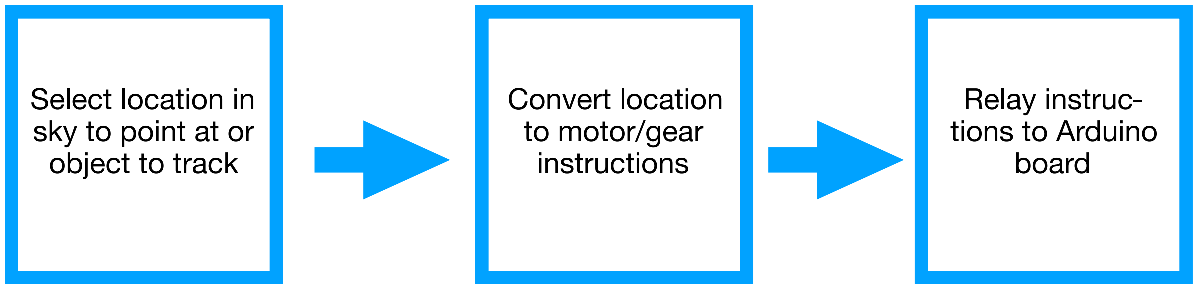
The software chosen will send information between an Arduino board. This board will connect all the functioning pieces of the telescope together. The board is programmed in C with a plethora of libraries available to use and will be developed through both us, the Computer Science team, and the Electrical engineering team. Our job will be providing the EE team with motor movement commands, either through inputs to the board or through functions within the Arduino software. These options depend on how the telescope software sends information.

If we chose open-source software, the only modifications that would need to be made are for the formulation of gear/motor movements for the telescope. So the selection of open-source software should be based on the interface the software provides. Is it easy to select where the telescope needs to point? If it is a better idea to calculate these movements in the Arduino software, the telescope software will not require modifications. There are multiple options for the telescope software that will be reviewed in the following paragraphs.

3.1 Technical Overview

The objective of our software development for telescope control is to provide a functioning program that allows a user to pick a point in the sky they wish the telescope to point at on the Windows platform. From there, the program will relay this position in either location strings or motor movements to our telescope for proper repositioning. Our program should also house the ability to track an object like the Sun or Moon. This requires the constant relay of instructions to the telescope. These base functions are the requirements for the program and can be tested through the images captured by the camera attached to the telescope. A stretch goal on top of these requirements is provide a way of tracking a faster moving object in the sky. This goal is more of a shared goal between the teams of this project as the telescope needs the proper movements associated to the output given by the control program.

The goal for the software development is provide a simple and easy to use interface for telescope control for beginners, experts, and everyone in between. The program should provide telescope control with simple setup for the whole range of users. The only modifications needed should be adjusting motor commands to the specific motors/gears used. Furthermore, our program should be similar in function to SkyX to help provide a well-documented project showing the general flow the observatory telescope would take. We want to show the control process can be completed without SkyX while still performing to the same standard the observatory had come to expect. The documentation created from this project could also lead to insight in where the current observatory telescope process is failing.


Select location in sky to point at or object to track
Convert location to motor/gear instructions
Relay instructions to Arduino board

3.2 Stellarium

Stellarium is the first open source program we came across. Stellarium was a project started back in 2001 by Fabien Chéreau (“Stellarium Astronomy Software.”). The software was first released in July of 2014 and had its most recent update in December of 2018. Each of its releases can be found on their GitHub repository with a change-log. The software can run on Linux, Windows, or Mac so no compatibility issues will arise.

Stellarium at its core is used to show a 3D model of a realistic sky. You tell the software what you want to view and it creates a visual 3d output of the sky at that location. It is not a current view of that location. For this project though, we would need to push beyond this core use and use their telescope control feature that was implemented in version 0.10.5. The software plugin allows you to control a GOTO telescope from the Stellarium UI through some basic instructions. This feature sends movement commands to the telescope and receives the current position of the telescope. By no means does the plugin allow for broad control of a telescope like SkyX does, the software does provide the functionality we are looking for. Supervision will need to happen during telescope control though as warnings for slew commands are not currently implemented by Stellarium. The issue that arises is when a slew command has the inverse effect of causing the telescope to hit itself. The software does not know the limitation of the telescope so it always assumes the telescope can be slewed to any position. It is also important to note the telescope can end up being pointed at the Sun, which is dangerous to view at without the proper filters on a camera that is mounted on our model.

Figure 1 - Stellarium main page (“Stellarium Astronomy Software.”)

With Stellarium, the telescope control feature works in two ways: direct or indirect connection. Direct connection will send slew commands and positional coordinates of the model through LX200 protocol. This protocol sends two strings for slew commands: one relates to right ascension and the other relates to declination. Depending on the type of mount used for the model, it may cause the need for conversions in coordinate systems. Stellarium only sends equatorial coordinates. These coordinates are a standard used by astronomers as it doesn't require constant recalculation. The current observatory uses an equatorial mount so it also deals with equatorial coordinates. If the sub-scale model follows this mount, no conversions need to be made. However, if an altazimuth mount is used, conversions will need to be made in the Arduino software from equatorial to the altazimuth coordinate plane. Thus, it would be in the best interest of this computer science team to follow the current observatory mount into the sub-scale model.

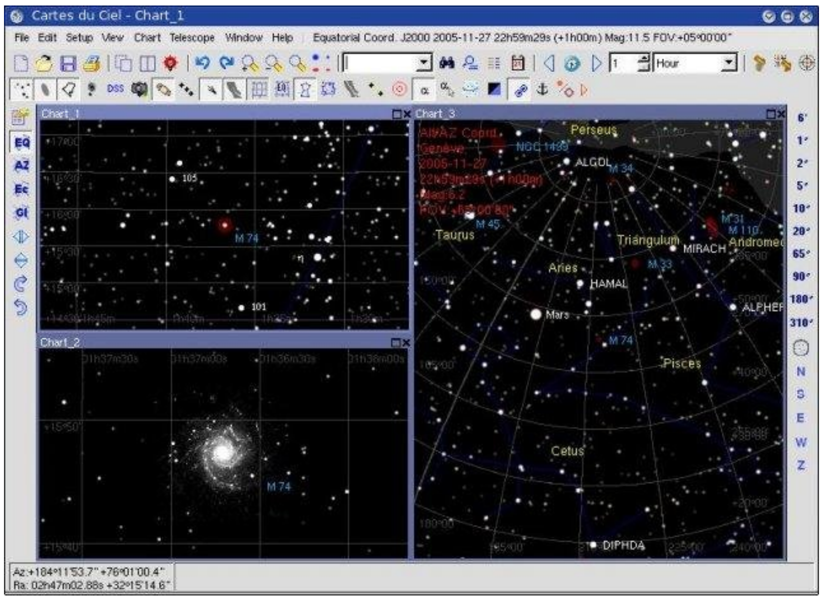

Cartes du Ciel - Chart_1
File Edit Setup View Chart Telescope Window Help
Equatorial Coord. J2000 2005-11-27 22h59m29s (+1h00m) Mag:11.5 FOV:+05°00'00"
Chart_1
Chart_2
Chart_3
Perseus
ALGOL
Taurus
Triangulum
MIRACH
Andromeda
Aries
HAMAL
Mars
Pisces
Cetus
DIPHDA
ALPHERATZ
M 74
Az:+184°11'53.7" +76°01'00.4"
Ra: 02h47m02.88s +32°15'14.6"

Figure 2 - Carte du Ciel main page (“Chevalley”)

3.3 Cartes du Ciel / SkyChart

The next option available is Cartes du Ciel written by Patrick Chevalley (Chevalley). SkyChart's base use, like the name implies, is draw sky charts. A sky chart is similar to a map as it shows the location of stars in the current sky. This software is open-source and available on Windows, Mac, or Linux. Since this software is based in the drawing of SkyCharts, its open library has a plethora of information. People can upload their charts to the library for others to use. This mass database is useful for stargazing on specific stars and space areas, something our model will not need to feature. Our model will not have the camera range to prove if we are looking in the correct area for many of the charts available. This would be a great feature for the observatory though, as it has a camera strong enough for detailed viewings of space.

Through the use of these charts, the software provides a vast library of objects to choose to slew a telescope to. To do so, the software can talk through the LX200 protocol like Stellarium. Furthermore, it supports INDI protocol for Mac and Linux while supporting ASCOM protocol for Windows. The ASCOM and INDI protocols are used to bridge the connection between control software and telescope. They are both open source frameworks designed specifically for the astronomy field. These protocols work through middleware causing the need for more software to be involved. The simplest option would to use the LX200 protocol to keep all operations within the software.

SkyChart adds more options for how to send data between the software and our telescope. These protocol options would be more helpful though with the observatory telescope. The sub-scale model will be very simplistic in features. We do not need to worry about camera focus or dome control like the observatory does. These features would find use with the INDI and ASCOM protocols as you can send more information between telescope and software. As we only need slew commands and position strings, the LX200 protocol is still most useful for the model.

3.4 Modifications needed

Stellarium and Cartes du Ciel are both viable solutions providing us with telescope control. Yet, the output they produce would not be readily available for the Electrical engineering team since both softwares output coordinates for each motor as a string. In both instances, we need coordinate conversions to motor instructions and the calculation of the meridian flip. This flip is needed because of the use of the german equatorial mount. When the telescope is tracking an object trough the sky that passes the meridian, the telescope would run into its own mount. To stop this from happening, at the meridian, we need a recalculation for the location that includes reversing the orientation of the telescope. Once flipped, the telescope can proceed tracking to the East.

These modifications can be attacked two ways. The first option is programming the needed calculations in the Arduino software after being provided the command from the software. The other option is developing a new software to manipulate the strings outputted by the open source program. If we make all changes on the Arduino board, it allows us to segregate all specific motor instructions to the board instead of trying to modify the source code of a program. Even though documentation is provided with the open source programs, we could end up altering some state within the program that should not be changed. An example would be deleting some function you think is no longer needed yet when you run the new version, the program no longer runs.

3.5 Personally Developed Software

The creation of a new software would provide us the opportunity to develop controls specifically for our telescope. Furthermore, it would take some processing power needed for motor movement calculations off the Arduino board and onto a computer. Finally, this new software could be written in the programming language of our choice and the ability to develop around Gazebo allowing for easy testing.

We can still develop software without creating a new planetarium program. The great thing about open-source software, besides it being free to use, is the ability to modify the software so long as we provide our developed software as open-source too. This software can use some of the features from the open-source programs like the vast knowledge of locations of stars in the sky held in Stellarium and SkyCharts.

Personally developing software a new program or even just modifying existing source code for our telescope may hurt the community outreach available. A hope of the model is that people can follow our instructions and software to create their own telescope. Some users may not have the same motors as us though. With our own software, the point of development would be to customize a program to our specific telescope. This may create areas where we are hard coding specifications of our telescope into the source code of the program. This could make the software unusable for others if they don’t build the telescope to our specifications. This may be avoidable by ensuring all telescope specific operations are handled by the Arduino but may also be unavoidable. Another possibility to maintain our outreach is providing detailed instructions on how to alter the code we create to suit the motor capabilities of the user at hand. Some people would not want to alter source code though for reasons of inexperience or not feeling comfortable. Another possibility to offset this possible negative consequence is providing the user with a telescope motor configuration option when connecting their telescope. The user would just input the specifications of the motor and gear setup they have. The software would then use this information to calculate the proper movements of the their specific telescope. This solution provides the broadest positive impact to our outreach but may produce issues we are unaware of. Our main goal is to provide our sub-scale model with a properly functioning control software. If we become stuck on an issue of compatibility with other telescope configurations, we could hurt the functionality of our telescope.

3.6 Comparing Options

Our options come down to two prior developed programs or developing our own. As we are more focused on the controlling the telescope than just a planetarium program, we should stick to prior developed programs. These programs already provide the mapping of stars needed to provide the correct slew strings to our telescope so there is no need to build a program from scratch.

Next our concern shifts to user experience. To help choose the best option, we can look at the process of telescope control in both programs and decide what provides the best way. Stellarium provides a minimalist viewing screen where most menus are hidden until you drag your mouse over the menu. These two menus that help provide most of the functionality are on the left side toward the bottom and at the bottom toward the left side. Then when a valid object is chosen in the sky, details that cover most of the left side of the screen pop up. SkyCharts also provides information on a selected object in the same fashion but with much less information compared to Stellarium. Stellarium provides a long list of the object in different coordinates like equatorial, cartesian, and galactic. Most of the information many people would not care about but nonetheless its provided.

With SkyCharts, you are presented with a screen much like an old version of Windows Office. The top portion of the screen is dedicated to a ribbon bar. On this bar, you are provided with all the functionality SkyCharts provide. Some options are buttons that turn on or off some feature while other options open up a pull down menu. SkyCharts does allow for multiple charts to be open in the same screen. This is a nice feature when looking for a star or planning out a future telescope control but would not be used too often for our purpose. For the user, Stellarium provides a cleaner look. You are not overwhelmed by the plethora of options that SkyCharts shows you while still providing most of those options. This can make it more difficult for someone to find an option in Stellarium over SkyCharts though.

Finally, some small features for each that are useful for development. Both programs are still being updated constantly to fix bugs and help keep the programs up to date with the current OS versions of the time. Stellarium uses GitHub to house their source code, which is convenient and provides and plethora of information including change logs and documentation. SkyCharts does not have a GitHub repository link and rather links to www.sourceforge.net to download the source files. SkyCharts still provides a documentation on their official site for all details relating to SkyCharts where items like change logs and a roadmap for future versions can be found.

3.7 Challenges

The first challenge, and most likely the biggest, will be relating software commands to telescope movements. As a computer scientist, we mostly live our lives in a virtual realm. However, this project is breaking that barrier. Our software development will have a direct impact on the movement of the model telescope. We need to find a way to turn location strings into proper movements based on our specific functionality of the telescope. With our telescope being designed from the ground up, the gear ratios and motor torque will be unique. So, the Arduino board will not be able to directly execute the strings given by the software we choose. We will need calculate conversions for the coordinates given by Stellarium as well as formulating a meridian flip. Luckily, I have taken some mechanical engineering courses that have dealt with these topics giving me confidence when tackling these issues. I also have a minor in mathematics to help us through the conversion processes. The problem not only lies within conversions but also coordinating with the electrical team of how they want the strings to be presented. We need to ensure we are in constant communication with them throughout this process so neither team ends up with unusable code.

The next challenge is reviewing the current code of Stellarium to find the areas where we will place in our modifications. Currently, most of the telescope control program does not feature documentation informing you of what the current function is accomplishing. Thus, we will need to take the extra time of reading through the plethora of code so we can pin point the areas we want to modify. When trying to decipher code, the easiest path is talking to the writers of this code. As this is not always possible no matter where your project is occurring, personal or work life, this will be good practice for the team.

Furthermore, a challenge we face is testing. How will we know whether our software is correctly controlling a telescope while our sub-scale model is being built? This is where we will use Gazebo. Ideally, our control plugin we create will be able to work in Gazebo where it will control a robot until the model is built. Once the model is built, testing will be handled through tracking objects with a GoPro and/or laser pointer. The GoPro can show success or failure in tracking objects through video. A laser pointer can show proper movement to a set coordinate and even tracking but someone would need to constantly be checking the accuracy of the laser to the object. Can we control the telescope to track the moon or sun? This will be our big test on the system as a whole. Testing is also not just important to ensure we can control a telescope but also are our coordinate conversions correct. Yes, math conversions can be checked on paper but what if we are missing an important step between motor instructions and celestial coordinates. This issue arises from the time dependency of celestial coordinates and the inactivity of the telescope between uses. When dealing with this kind of issue, it can usually be troubleshooted through trial and error. The problem is the design of the telescope has not been finalized as of yet so we cannot put our conversions through a process of trial and error since we don't know gear ratios and stepper counts for the motors.

4. Observatory Refreshes

A portion of our project is some technology refreshes for some items in the observatory. The current technology is old and mostly outdated. Furthermore, some of the components at the observatory have not been used or even touched in a few years. These tasks are more IT related though than computer science and do not involve any program development or analysis.

First off, the control computer needs to be upgraded to Windows 10. It is currently running on Windows 7, which is being phased out by Microsoft. This will be achieved by first making a complete copy of the current hard drive. We run a disk imaging process through Linux on a usb stick. This will be done to ensure the current state of the control computer can be reset at any time in case of some type of failure once upgraded to Windows 10. Once updated, it will be important to ensure all drivers are fully functioning still. The control computer contains multiple drivers to control all operations of the observatory from the telescope to the dome. We will be able to run an update check on the drivers to ensure they are updated to the proper version for Windows 10 functionality. A current check has shown all drivers can function with Windows 10.

Next, the observatory faculty would like the ability to control the telescope control computer from their downstairs iMac. This feature would not only allow them to control the telescope from downstairs but also view the current progress of the telescope when it is running. To do this, the iMac hard drive will first be backed up just like when upgrading the control computer for the Windows 10 upgrade. We want the same ability to roll back the computer in case anything goes awry with dual operating systems on one machine. Then, we can install either Parallel or Bootcamp to run Windows on the iMac. Once the Windows operating system is installed, remote access to another Windows computer is feature of the operating system. Bootcamp is a feature already built into the Mac operating system but requires a reboot to switch between operating systems and the system storage options pertaining to the Windows operating side are not customizable like Parallels. When a storage size is selected at installation, it cannot be adjusted in the future like Parallels allows. This means if the Mac or Windows operating side runs out of memory, the user would not be able to adjust the memory to get more from the other operating side. Parallels runs like a virtual machine on the computer allowing the system to run Mac and Windows at the same time without the need for rebooting the computer. This can save a hassle of constant rebooting so one operating system can be used but the processing of the computer will take a toll. For the purpose of the observatory, we will use Bootcamp so Windows can be used. This decision is based on the computing power of the iMac as its an older generation and will have a hard time if running Windows and Mac concurrently. In extension, the memory customization Parallels provides would most likely go unused. The iMac is not the main operations computer of the observatory so the main memory has a large vacancy for space. Finally, Bootcamp also does not need to be purchased like Parallels does. Parallels cost $79.99 for a one time license of the program.

Another feature the observatory team wants is the creation of live, or similar to live, stream of the telescope photos being taken. This is a feature that will be within our web application but its true purpose will not be able to be tested. We can send test data to their server to be pulled by our website but the full function from telescope to website will not be able to be tested.

Finally, the observatory has a server they want reinstated. The server was used to house all the pictures taken by the telescope and allowed for remote viewing. This update will be hard to completely test as the telescope will not be fixed during our project so we will need to send test data instead. The current server has no been used in some time and has possible water damage according to the observatory personnel. The server does boot upon testing but has yet to be hooked up to retrieve file transfers. We were able to backup their current server data to portable disk. The data we retrieved is in tact, allowing us to use their previous telescope pictures as test data within our website.

The observatory team also asked for a few more upgrades that wouldn't be feasible for our project as they require the telescope to be in operation. They wanted scripting capabilities for their SkyX software used in controlling the telescope. This is a feature we can work into our scale model system though. Scripting is available in Stellarium as a plugin option and we can test scripting through tracking objects and viewing image data afterwards. Image data from the GoPro provides accurate timestamps allowing for precise review of the scripting success of our program(s). Our scale model will not run with

SkyX but we can provide a possible overview into how it could work. They also wanted an update in the time given to the telescope picture's metadata. It currently is given the wrong time but we have no way of producing a similar experience without the telescope working.

5. Software design overview

After research into open source programs, our group has decided with using Stellarium as the frontend software control program. We will then add onto to Stellarium's code to create the needed strings the EE team needs in their Arduino board. The EE team is looking to receive information containing which motors need to move, the direction needed to move and how far the motor needs to move. These calculations will be made before we send any information to the Arduino board through USB. After the movement is complete, our software will receive a string informing us if the movement was successful or not.

The choice for Stellarium was a combination of popularity and a simpler user interface. An interface that is not as jumbled as SkyCharts, which could lead to confusion with inexperienced users. Since both programs offer an extensive library for object coordinates and provide telescope control through the LX200 protocol, the decision came down to simplicity. Stellarium offers an easier to use interface in our opinion and through research seems to be the choice made by many for an open source control program. Finally we decided to use an open source program instead of creating our own so we can focus strictly on the instructions we will be giving to the Arduino board. Stellarium provides everything and is in a stable working condition. Thus, there is no need to develop our own planetarium program when we are only interested in telescope control. The modifications will be made in the plugins section of the Stellarium code. All telescope control features are housed within the plugins section for telescope control. These files are written in C++ and include header files as well.

We want to still modify the telescope control plugin provided because we believe there is a better way of handling control we need than Stellarium does. The control plugin lacks instructions and can be hard to find unless. In the source code, you can find simple instructions for use commented within the code. We also want to add more features to the telescope control plugin to be discussed in further detail in the accompanying sections.

5.1 Modifying Stellarium

To convert the output strings sent by Stellarium for the telescope control, modifications will be made to the telescope control plugin. This plugin can be easily found within the directory for plugins in the Stellarium source code. All code pertaining to telescope control is written in C++, which can be edited with nearly any text editor. Our edits will be done within an integrated development environment (IDE) though. An IDE like Visual Studio Code allows us to view errors within our modifications before compiling the software as a whole. We plan to only modify code pertaining to the LX200 coordinate output along with the windows created for telescope control. We will only modify LX200 outputs as this is the protocol we will use. There is no need to alter other protocols we are not familiar with and possibly break those protocols within Stellarium. Next, we will modify the control windows to add features like saving coordinates and viewing the live stream of the GoPro on the telescope. We also want to add more text within the windows to help provide a greater walkthrough of the telescope control process. This would allow a user to simply follow the onscreen instructions when setting up and using the telescope control instead of reading through documentation as many people don’t read the documentation anyway. A user friendly approach instead of simply creating the telescope modifications needed ensures for a pleasant using experience. An experience that could make the difference between wanting to use our software or looking for a different option.

5.2 Compiling Stellarium

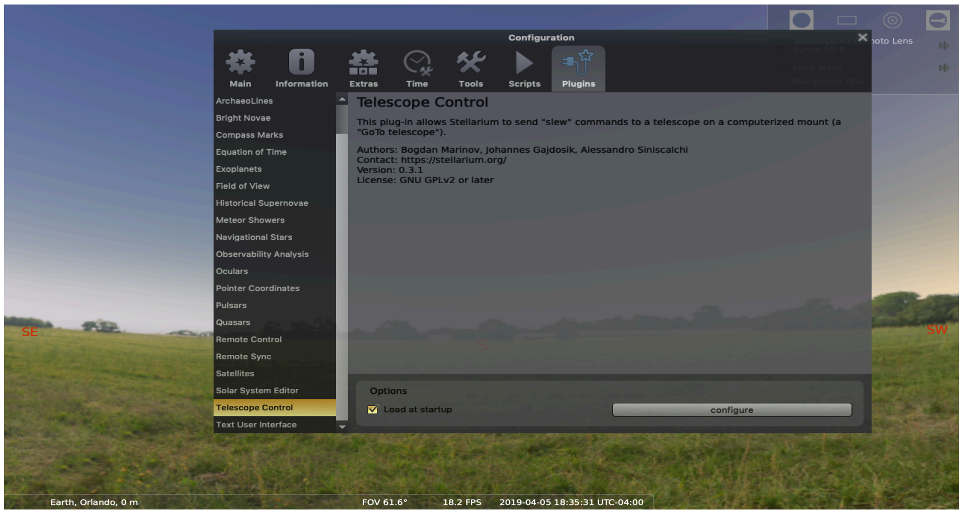
Since Stellarium is open source, the source code is obtainable online but the program can be installed straight from www.stellarium.org. In our case, we want the source code version so we can manipulate some of the outputs of Stellarium. In order to compile the code after modifications, QT5, CMake, and Visual Studios 2017 are all need to be downloaded. As a side note, Visual Studios 2017 is a minimum requirement. You can use a more recent version but this process is guaranteed to work with that version. All these downloads are available online as open source options. The community edition of Visual Studios works for this process. The source code can be obtained through git requests in a terminal, downloaded online, or through GitHub Desktop. If using Github Desktop, after cloning the repository, you should not be sending your modification back to the Stellarium repository.

To compile the code, Qt5 is first used to configure debug and release builds. When using Qt5, it is important to match the version of Visual Studios to the appropriate Qt5 version as the IDE within Qt5 depends on Visual Studios. Next, CMake is used to configure some user options onto the build. It is recommended to use version 3.4.1 or better. Finally, Visual Studios is used to change the build mode to release and build the program. This process can be started in one of two ways: by executing this file, C:\Devel\stellarium\builds\msvc2017\Stellarium.sln, or by clicking the open project button from the CMake GUI. After this process is completed, the program has been complied and is thus ready to be used.

5.3 Using Stellarium

First Time Use Setup - The telescope needs to be configured within Stellarium.

1. The telescope will be connected through USB. Once plugged in, ensure Stellarium recognizes the connection.



Telescope Control

This plug-in allows Stellarium to send "slew" commands to a telescope on a computerized mount (a "GoTo telescope").

Authors: Bogdan Marinov, Johannes Gajdosik, Alessandro Siniscalchi
Contact: https://stellarium.org/
Version: 0.3.1
License: GNU GPLv2 or later

Options
☑ Load at startup

2. Press F2 to open the configuration settings window.

3. Select “Plugins” from the tabs available.

4. Select “Telescope Control” from the options on left (image above shows telescope control plugin page).

5. Ensure “Load at startup” is selected at the bottom on the window and Select “Configure” next to it.

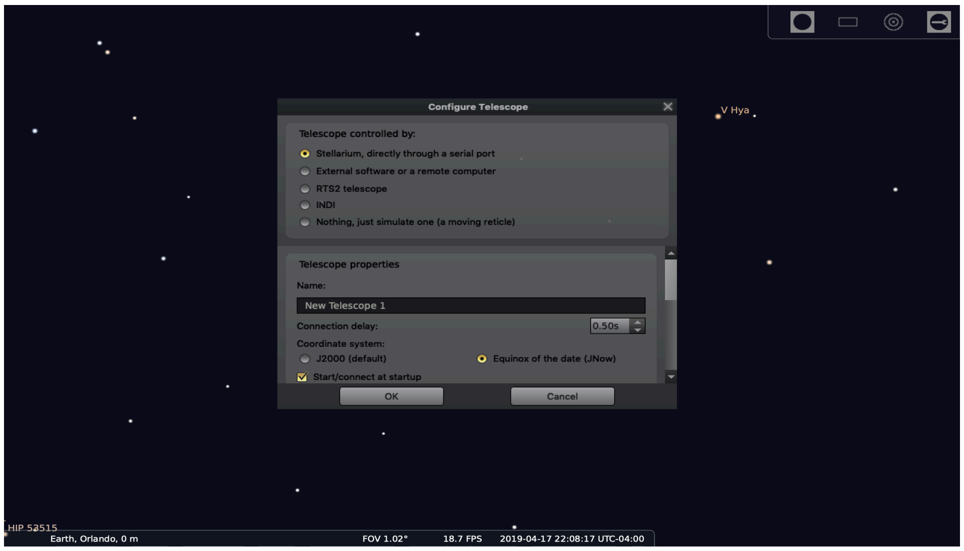
6. Select Add to create a new telescope.

7. In the new window (displayed in the image below), select the first option “Stellarium, directly though a serial port”.

8. Next ensure the coordinate system is set to “Equinox of the date (Jnow)” and the checkbox for “Start/connect at startup” is checked

-You can also rename your telescope but this is unnecessary

9. Finally, select Ok at the bottom of the menu. You will be placed back to the telescopes overview page where you can see whether the telescope is connected and the number associated with the telescope.



Configure Telescope

Telescope controlled by:
[✓ Stellarium, directly through a serial port]
External software or a remote computer
RTS2 telescope
INDI
Nothing, just simulate one (a moving reticle)

Telescope properties
Name: New Telescope 1
Connection delay: 0.50s
Coordinate system: J2000 (default) [✓ Equinox of the date (JNow)]
[✓ Start/connect at startup]

Each Subsequent Use Setup: Ensure Stellarium recognizes the connection to your telescope by visiting the configurations menu again.

Telescope Control

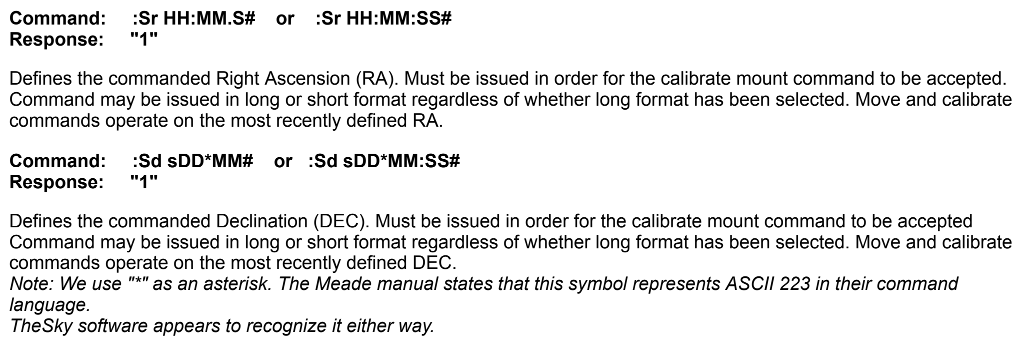
1. Select and click the object you wish the telescope to point at and
2. Press CRTL + (the number associated with the telescope)

Finally, it is important to always ensure Stellarium is in equatorial mode and not azimuth mode. While this toggle does not change the output strings to the telescope, it does help the user view the correct orientation of the sky related to the equatorial mount being used. Objects in the sky are displayed in different locations when switching between modes, which can lead to confusion. This confusion can lead to the thought the telescope may not be pointed in the correct location. This option can be toggled between using CRTL + M or by selecting the option from the menu on the bottom. The toggle can be seen on the bottom menu in the fifth box from the left in the first slot. The button is a telescope with a dotted ‘x’ in the back. It is in equatorial mount mode when the toggle is lit up. You can also have the program display a message indicating the mode everytime you switch. Under default settings this option is not chosen, so the user needs to open the configuration settings window and select tools from the available tabs. In the tools window about halfway down the options, there is a checkbox named “Indication for mount mode” located to the left of “Dithering” option.

6.0 Celestial Coordinates

In Stellarium, the current coordinates provided for right ascension(RA) and declination(DEC) are sent as celestial coordinates, also known as equatorial coordinates. Cartesian coordinates are the standard for most locations like determining the location of the Robinson observatory or Paris, France. These locations are constructed through latitude and longitude and do not rely on time. These coordinates can be easily manipulated with sine, cosine, and tangent to find other pieces of the location. The same is not present with celestial coordinates.

Since the sky is constantly moving, astronomers formulated coordinates to use for this constant change instead of using typical cartesian coordinates. Somewhat relating to



Command: :Sr HH:MM.S# or :Sr HH:MM:SS#
Response: "1"

Defines the commanded Right Ascension (RA). Must be issued in order for the calibrate mount command to be accepted. Command may be issued in long or short format regardless of whether long format has been selected. Move and calibrate commands operate on the most recently defined RA.

Command: :Sd sDD*MM# or :Sd sDD*MM:SS#
Response: "1"

Defines the commanded Declination (DEC). Must be issued in order for the calibrate mount command to be accepted Command may be issued in long or short format regardless of whether long format has been selected. Move and calibrate commands operate on the most recently defined DEC.
Note: We use "" as an asterisk. The Meade manual states that this symbol represents ASCII 223 in their command language.*
TheSky software appears to recognize it either way.

normal coordinates, right ascension is similar to longitude while declination is similar to latitude. Declination starts at the celestial equator. The celestial equator extends from the Earth's equator onto the celestial sphere. This sphere is fictional but helps you understand what parts of the sky you can see. This sphere is not relative to the horizon but rather where you are and informs you anything above the horizon you can see but anything below the horizon can't be seen. Right ascension relates to the vernal equinox. This is an arbitrary point chosen each year where the Sun appears to cross, since the celestial sphere is fictional, the celestial equator when moving from south to north **(reference)**. Starting at this point, right ascension is 0 degrees and increases eastward while declination is 0 degrees and increases upward. The coordinates are sent by Stellarium in hours, minutes, and seconds for right ascension and degrees, arcminutes, and arcseconds for declination. These two strings are currently sent in the template provided below and are distinguished by the start with either :Sr for right ascension and :Sd for declination.

LX200 commands formulated by Stellarium

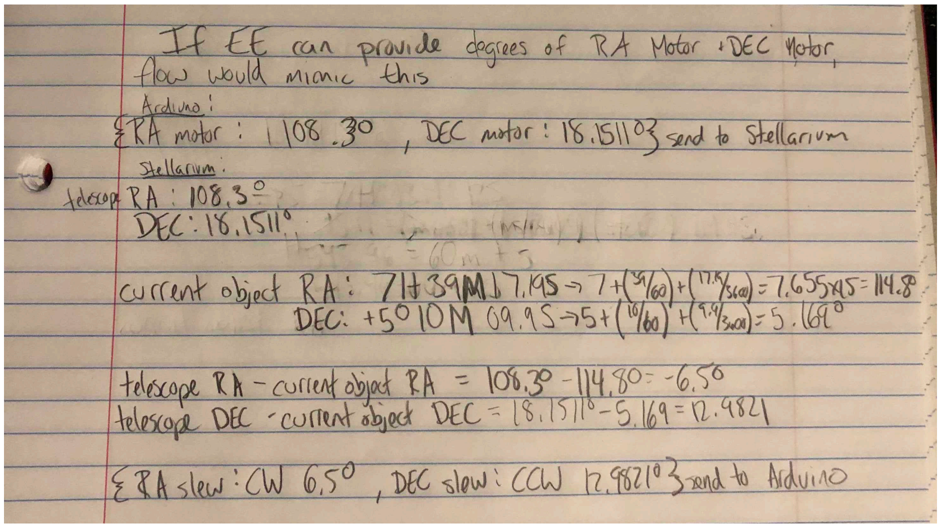
The tricky part of celestial coordinates are they are constantly changing because the celestial coordinates are directly impacted by the orientation of the Earth. An orientation that goes through 26,000 year cycles. With this constant change, coordinates are derived from the current epoch or through the equinox of the date. An epoch is reference point used for a moment in time **(reference)**. We are currently in J2000 with the next update coming in 2050. It is important to note Stellarium lets us decide what reference of time we will use for object locations. If the system is ever changed in the settings, the telescope should be immediately recalibrated or the telescope will not point to the correct location. With this dependency with time, a real time clock needs to be in use when coordinates are formulated along with slew commands. This requires all conversions to be handled within Stellarium and not the Arduino board. The board has no room to hold the micro controller needed for a real time clock.

Conversions can be made between celestial and cartesian coordinates. The distance of the object in either light years or parsecs becomes apart of the equation though. These conversions are usually only helpful when dealing with an altazimuth mount. This mount has two motors like the german equatorial mount but deal directly with vertical and horizontal movement relative to the ground below. German equatorial mounts take into consideration the poles of the Earth before the assembly of the right ascension and declination motors. With these differences, converting between celestial and cartesian would provide extra steps when calculating slew commands and thus are unnecessary. We will keep our coordinates in celestial to maintain a standard with the slew command process.

6.1 Mathematical process for telescope pointing

First off, it will be important to calibrate the telescope so that the home position relates exactly to the vernal equinox. The vernal equinox for this year, 2019, will happened on March 20 at 5:58 Eastern Standard time. To help calibrate the telescope to this position when the time comes, we will use the current location of the Sun at that time in altitude and azimuth. These coordinates work well here as they are absolute positions in the current sky and don't rely on time. We would use this position as our base then use the Sun's celestial coordinates to move to the vernal equinox. This is done by moving to 0 degrees RA and 0 degrees DEC. This step will only need to happen upon first use of the telescope. After that, the home position will be saved to memory.

To provide the slew instructions for an object selected, we need to receive the current degree placements of both motors from the Arduino board. These readings sent from the Arduino board will tell the change from the standard home position of the telescope instead of the telescope's actual location. So first, we will need to take the readings from home position and compare them to the standard home position. This will give us the



If EE can provide degrees of RA Motor + DEC Motor, flow would mimic this

Arduino:

{RA motor : 108.3° , DEC motor : 18.1511°} send to Stellarium

Stellarium:

telescope RA : 108.3°

DEC : 18.1511°

current object RA : 7H 39M 17.19S → $7 + (39/60) + (17.19/3600) = 7.655 \times 15 = 114.8$

DEC : +5° 10M 09.9S → $5 + (10/60) + (9.9/3600) = 5.169°$

telescope RA − current object RA = $108.3° - 114.8° = -6.5°$

telescope DEC − current object DEC = $18.1511° - 5.169 = 12.9821$

{RA slew : CW 6.5° , DEC slew : CCW 12.9821°} send to Arduino

current viewing location of the telescope. Then from that position, we can take the difference between the telescope position and the current object to be tracked position to provide the correct instructions. This requires us to convert the Stellarium strings for RA and DEC to degrees. The math is provided in the image below. Declination is pretty straight forward in conversion to degrees but right ascension requires the conversion to degrees than the multiplication by 15. This accounts for the fact the Earth moves about 360 degrees in 24 hours resulting in 15 degrees an hour. Furthermore for both conversions, minutes for RA and arcminutes for DEC are divided by 60 to account for 60 minutes in an hour. Next seconds for RA and arcseconds for DEC are divided by 3600 to account for 60 seconds of 60 minutes in an hour. One last step for declination is applying the sign of the original coordinates given by Stellarium to the newly converted degrees. This only happens for declination because right ascension is always positive. Right ascension only ever calculates eastward. Declination can calculate both upwards and downwards relative to the celestial sphere.

First image shows formulas, Second image provides example of process

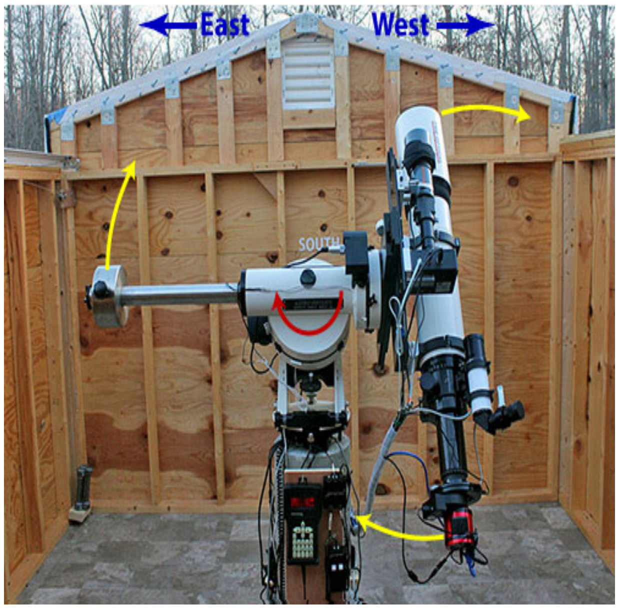
6.2 Meridian Flip

Coordinate conversions is not the only math we need to add to Stellarium. We also need to formulate a meridian flip. A meridian flip can be implemented in a few ways on a telescope like through a physical switch or through software implementation. We will focus on the software implementation where before we send a slew string to the Arduino we will have already accounted for the meridian flip if needed to complete the slew.

This flip is only needed on equatorial mounts because the telescope is aligned with the polar axis. The meridian is another circle in the celestial sphere that passes through the zenith, nadir and the two celestial poles. The zenith is directly above where a person is located. This point is marked on the celestial sphere. Conversely, the nadir is directly beneath where a person is located. This point in marked on the opposite end of the sphere. Once these points are found, the meridian can be determined.

When the telescope passes this circle in the celestial sphere, a reorientation needs to be performed. Without this reorientation of the telescope, it may lead to the damage of the mount or motors if the telescope is pinned against itself but keeps trying to move. So this flip needs to be implemented in two ways. The first is when tracking an object from the West side of the meridian to the East side. The second is when tracking an object from the East side of the meridian to the West side.

A meridian flip is unique to each telescope as the dimensions of telescopes are different so there is not a standard where your telescope will collide with itself at a certain position past the meridian. This limit will need to be researched once our telescope is built. If a slew will cause the telescope to travel past the limit we define, we will reorientate the telescope so further tracking of the object is possible.



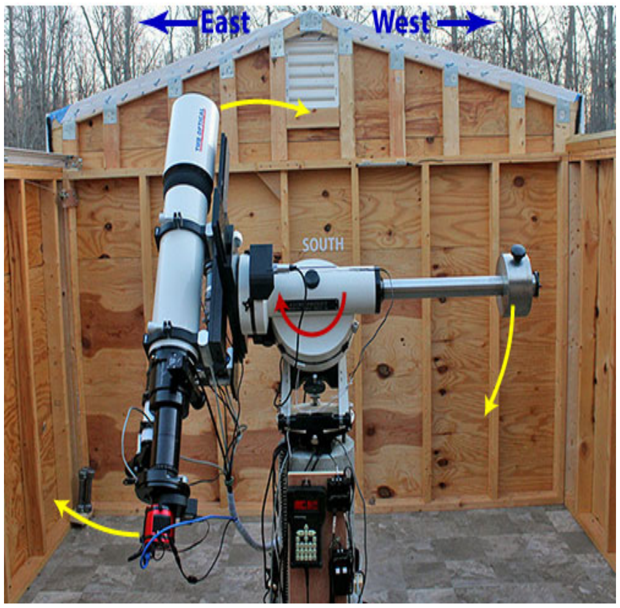
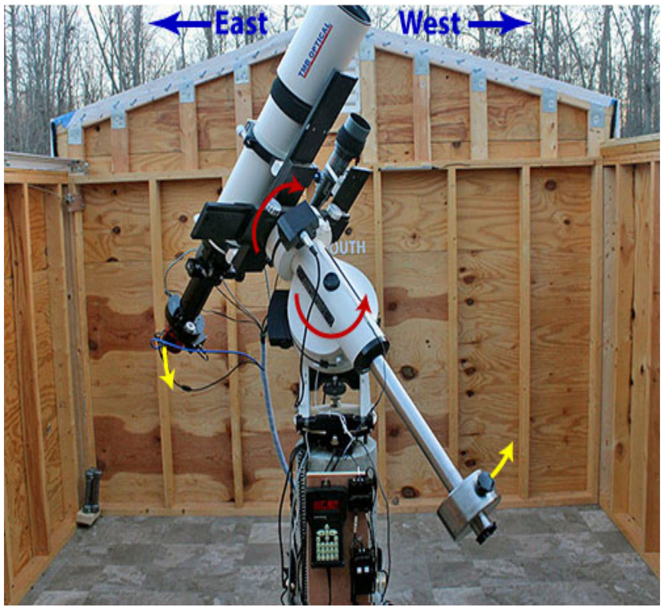


Image on left is start of tracking. Image on right shows tracking after flip

To help give a visual idea of the meridian flip, three images are supplied below. The object being tracked started to the east of the meridian and moves westward while remaining at a DEC of 0 degrees. As the telescope tracks westward, the first image shows the end of the telescope would hit the mount if the telescope stayed on its current course. To avoid this, we would send to the Arduino to start rotating the RA motor counterclockwise from its current location while the DEC motor rotates 180 degrees clockwise. This flip is shown in image two. The telescope will not end up pointing to a new location. Rather the mix of these movements will result in flip over the 'y' axis of the mount as shown in the third image. The DEC motor will once again be pointed to 0 degrees while the RA motor will be pointed to the same degree it had before the flip but can now keep



tracking westward.

Telescope while performing meridian flip.

What does this mean for our communications between Stellarium and the Arduino? If the flip happens during the tracking of an object, tracking must halt and the end coordinates need to be saved while the flip occurs. We will take control of the telescope while the flip happens thus the coordinates need to be saved so we can reinitialize the slew after the flip has been processed. This will be implemented so our flip commands and slew commands do not tamper with each other. The strings outputted by the meridian flip will not be telling the telescope about a new point in the sky and thus will be the same no matter where the flip occurs. The DEC motor will rotate 180 degrees and then the RA motor will rotate counterclockwise until the original DEC location is achieved. Next, Stellarium will need to take in a new command outside the LX200 protocol for a manual annual meridian flip initialized by a button on the telescope. This command will call the same meridian flip function as used during object tracking but our software would not have known the need to do so without the command from the Arduino.

7.0 Budget

Product	Price
Windows 10 License	$120 - Subsidized by UCF Senior Design
GoPro Camera	$350 - Provided by FSI
Backup HDD	$70 - Provided by FSI
Website Hosting	Free - Hosted by UCF or through student credits
Gazebo	Open Source
Stellarium	Open Source

http://spiff.rit.edu/classes/phys373/lectures/radec/radec.html - celestial coordinates

http://stars.astro.illinois.edu/celsph.html - celestial sphere