Spectral Telescope for Star Classification

Addison Long, Aaron Moreno, Alejandro Olivo, and Sebastian Rowe

College of Engineering and Computer Science,
College of Optics and Photonics,
University of Central Florida, Orlando, Florida, 32826, U.S.A.

***Abstract* — Stellar spectroscopy has remained an important area of study since Newton first observed that the sun's spectrum had missing colors. Today, the James Webb Space Telescope uses spectroscopy to analyze the atmosphere of extrasolar planets in the search for life. Spectroscopy enables us to extract information by splitting white light into its constituent wavelengths and mapping them in position space. In our project, we aim to create our stellar spectrometer and extract as much information as possible to classify stars. Our primary goal is to determine star temperature by analyzing the blackbody curve emitted from stars.**

***Index Terms* — Astrophysics, automation, imaging, spectroscopy, stellar classification**

I. Introduction

The creation of this spectral telescope used for star classification has allowed us to capture the spectrum of stars and classify them by the Morgan Keenan system. Capturing star spectra and classifying them will allow us to discover where on the main sequence the star being observed sits. From this information, we will be able to discover relative mass temperature and elemental makeup. Along with this information on the star, we can also determine whether or not the star is a binary star system. To complete these feats, our group designed a spectrometer, tracking system, guiding system, and imaging system.

The spectrometer uses a modified Czerny turner design optimized for scotopic conditions. The guiding system is designed to resolve stars with short exposure times so that we can ensure tracking accuracy. To ensure perfect focus, both the guide camera and telescope are equipped with an autofocus system. Lastly, our tracking system, after calibration, creates a map of the night sky and gives us go-to functionality so that we can easily find whatever object we wish and take its spectrum. We designed our own tracking system because it is important to track accurately. The software for this system is written and developed by our group and accessible through an on-device display. To ensure our system can be used in all environments, we used an advanced image processing technique known as image stacking. This approach increases our dynamic range and signal-to-noise ratio at the cost of time. With this system, we have an all-in-one system capable of providing amateur astronomers with research tools that were previously unaffordable.

II. Spectrometer Design

The spectrometer in our system is of the Czerny Turner design, which uses catoptric optics to decrease signal loss. The simulated system in Zemax has a 1 nm spectral resolution following the Morgan Keenan stellar spectral classification standard.

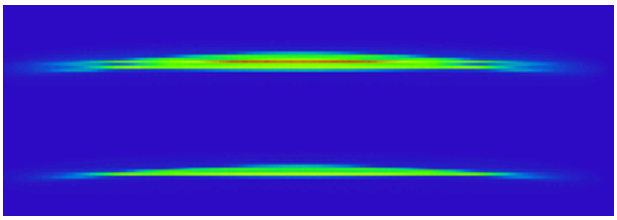

Fig. 1. Detector view in Zemax Non sequential mode showing the simulated wavelengths 550 nm, 551 nm, and 560 nm. False color intensity image showing separation between 1 nm and 10 nm.

The spectrometer in our system is of the Czerny Turner design, which uses catoptric optics to decrease signal loss. The simulated system in Zemax has a 1 nm spectral resolution following the Morgan Keenan stellar spectral classification standard. Since we designed a ground-based system, we aimed to cover the entire optical spectrum not absorbed by the atmosphere from 400-700 nm. We chose to use 250 mm silver-coated mirrors with high reflectance from 400 nm - 2 um, covering the entire optical range and eliminating any chromatic aberration. Lastly, we used a ruled diffraction grating with 600 lines/mm, giving adequate separation of spectral lines and preserving as much signal as possible since ruled grafting put most of the signal into a single mode. To ensure that the camera's sensitivity has minimal effect on the black body curve we retrieve, we decided on a cooled deep sky camera with a relatively flat spectral response across the optical range. This camera also has the benefit of being a mirrorless camera; in a DSLR, the mirror flipping out of the way to expose the sensor causes a vibration that can smear an image, reducing spectral resolution. Furthermore, the camera has a full well 63.7Ke sensor, providing a high dynamic range for any imaging task and more contrast levels.

The last component of the spectrometer is the telescope choice. Since we must prevent chromatic aberration from

affecting our spectrometer, we had to choose a reflecting telescope. The best choice was a 6-inch SCT with a 1500mm focal length. This telescope gives us a focal ratio of f/10, which, although slow, prevents star crowding and makes it easier to pick out individual star's spectrums.

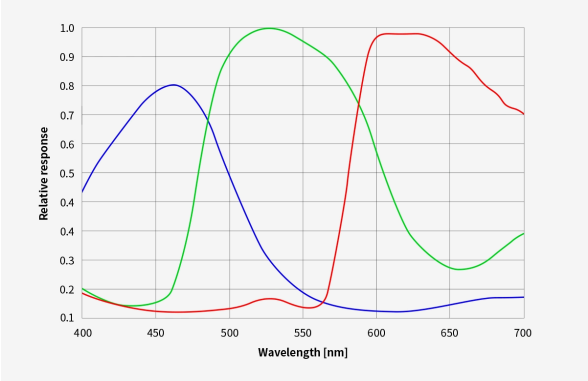

Fig. 2. Spectral response of the ZWO ASI294MC Pro from 400 nm - 700 nm from the manufacturer's website.

III. Stellar Classification

The Morgan Keenan classification system systematically labels any star given a spectrum with a 1 nm spectral resolution. Our initial plan was to implement this system of classification so that we could look at any star and place it in the main sequence; however, the following constraints and limitations prevented us from reaching the required resolution. First, the telescope and mount we are using require lightweight materials. At most, we can add an extra 3 pounds, and we need to fit the spectrometer and guide scope within that weight. This constraint limits us to using privately owned 3D printers. Unfortunately, the printing accuracy we experienced was less than desired, and our method for mounting mirrors leaves no room for adjustments. The slight angles introduced due to printing imperfections add aberration, significantly limiting resolution. Since we must work with low-signal sources such as stars, a catoptric system is vitally important; however, mirrors are significantly more sensitive to misalignments than refractive optics, worsening the aberration encountered from the print errors.

Lastly, the ocean temperatures and overall atmospheric moisture this summer have prevented us from having more than two clear nights of actual testing time, with one of those nights happening before the spectrometer was assembled. Given our inability to test in ideal environments, we must pivot and classify stars by temperature, which can be performed without high-resolution spectroscopy. To determine star temperature, we will use the fact that stars are blackbody irradiators. By analyzing the blackbody radiation, we can estimate star temperature by simply knowing the peak wavelength of the blackbody curve. To reduce computation, we will use Wein's law to determine star temperature. The significant advantage of Wein's law is that it relies on a single constant of proportionality, while Wein's law falls apart for sources that peak in the ultraviolet regime. The stars we will be analyzing peak in the optical regime where Wein law holds.

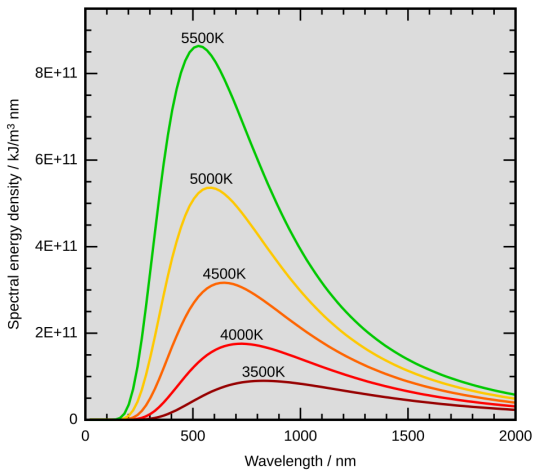

Fig. 3. Black body radiation curve from Wein's law for various temperatures.

IV. Results

We had to choose a light source to approximate a black body irradiator to test our system in the absence of clear nights. First, we considered using a candle, but unfortunately, any room we could test in had airflow, which caused the candle flame to move and flicker, which would muddy the spectrum. Filament lights are also an option, but they are too big. Our system is designed to capture the spectrum from stars with an angular size of only a few arc seconds. Given the focal length of our telescope, we would be unable to place a light bulb sufficiently far away for testing. Finally, we determined that a White LED would sufficiently approximate a black body source. LEDs are fundamentally not blackbody sources; however, because the LED, even placed far away, is much bigger than a star, the spectrum captured is low enough resolution that the spectrum retrieved is similar to what we would expect from a blackbody source.

In figure 3, you see the spectrum and a 1D profile, which has been expanded and cast to grayscale. From there, we can plot the entire spectrum from 400-700 nm, knowing from the spectral responsivity of the camera that the edges of our spectrum will be at 400 nm and 700 nm, respectively. With a peak wavelength of around 560 nm,

we get a temperature of 5175k. Our system is working as intended because daylight represents near-perfect white at 5700k, and the LED appears to be a near-perfect white to the eye. Given that our testing environment was flooded with excess room lighting of a cooler yellow-orange, the system should read a little lower than the true value color temperature of the LED.

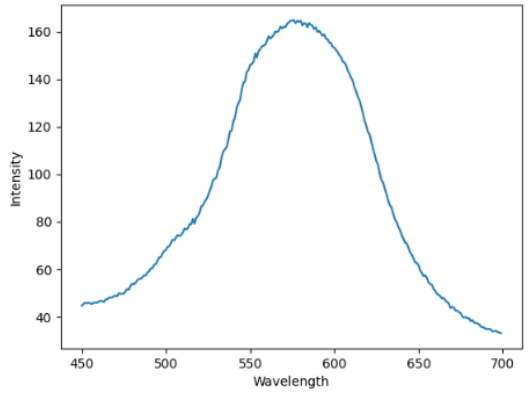

Fig. 4. Test spectrum of the LED placed at a distance from our system. Peak is at around 560 nm.

V. User Interface

The user interface being used for this project is a task that was developed completely from scratch. While pulling references and inspiration from different astrophotography software, we decided it best to create our very own user interface. In this way, we will have a program that is completely within our control and understanding while also being able to challenge ourselves. Developing our own user interface will create more independence for us because it will solidify the individuality and uniqueness of this project.

The best approach to creating our user interface was to implement the design and functions using the programming language known as Python. Python is a great, easy-to-use, modern programming language with limitless functions and applications. Many programmers find Python to be one of the easiest languages to learn and work with because of its higher-level script language. More specifically, we are using the Python libraries known as Tkinter and CustomTkinter to form our user interface.

Python was selected as our language of choice because of its previously mentioned ease of use as well as its limitless applications. Python is a high-level language best used for data analysis, data visualization, task automation, as well as web development. Each of these features deem Python more than suitable for the tasks of our project.

A. CustomTkinter

The purpose of implementing CustomTkinter is because we want our user interface to appear as modern and pleasing to the eye as possible. CustomTkinter is a user interface based extension of Python that allows not only for the creation of elements in a window, but the ability to customize said elements. Using CustomTkinter will allow this project to be presented in a way that is more appealing to us and others.

B. Properties of the User Interface

The development of the user interface began with the display. The parameters for the window consisted of a 840x720 display to support a smaller design on any monitor. We then wanted to create multiple pages so we can implement inputs as well as a live view of the telescope. To do this, a button was created to separate the main window and the live view window.

The main window contains very simple features that allow the user to input parameters for the telescope. Features such as temperature control, exposure time, and more will let the user set up the system to his or her liking. Once all of the necessary information is set, the user simply needs to press the “Submit” button at the bottom of the page so that the software can recognize the information provided. Using code from an outside source, we are able to not only implement our own inputs, but also analyze and capture stellar images through PHD2.

In the second window is where all of the live viewing will be done. Our user interface is set up in a way that we can view everything the telescope sees. Furthermore, the second window will allow users to control the position of the telescope and even capture an image of the telescope’s view.

The features implemented in the user interface are designed to appear very straightforward and easy to use. We also made sure to provide features absolutely necessary for the success of our spectrometer. With the help of Python and its extension CustomTkinter, we were able to create a well functioning user interface suitable for all.

VI. Image Processing

Image processing is used in this project to capture and understand the images we are receiving from the telescope. This can be done using certain astrophotography software. These programs use image stacking software to pre-process and calibrate data from images. A few known image processing software include Adobe Photoshop, PixInsight, Siril, DeepSkyStacker, and Registax. Although each software uses similar pre-processing and post-processing steps as well as meets

the same end goal, they each have their own qualities that help astrophotographers decide on the one that suits them.

The most important factor considered when deciding which software we would like our software to resemble is the type of astrophotography being done as well as its ease of use. The softwares we analyzed included Night Sky Photography, Piggyback Astrophotography, Prime Focus Astrophotography, Spectroscopy, and Deep Sky Astrophotography. When comparing the previously listed astrophotography software, the central focuses were on tools, complexity, and appearance.

Of the software options mentioned, Siril is free and easy to learn. Siril has many beneficial features such as noise reduction, image alignment, stacking, photometric color calibration, and can run the third-party tool StarNet. StarNet is a program that allows users to remove stars from astrophotography images. The only issue with Siril is the amount of storage a file may swallow because of its stacking capabilities.

Recently, artificial intelligence has also been used for more complex image processing. Because of the increase in volume of astronomical data, tools are being used and developed to withhold and create better images from the data. Artificial intelligence is primarily being used to combine images from different sources and detecting and classifying astral objects.

Analyzing and identifying astral objects that may not be easily viewable is a very costly task. Being able to identify these objects requires higher complexity and is very time consuming. With AI technology, the cost of identifying these astral objects decreases significantly because astronomers can utilize computer vision techniques to classify the objects.

To achieve image processing, we focused on astrophotography software with functions that resembled that of Siril because Siril's user interface aligns with how we designed our user interface to work. Pre-processing is done in a much easier way because of the scripts written within the software. The tools Siril provides will also be extremely useful to us and what we would like to accomplish with our project. By primarily using the C and C++ programming languages, Siril captures and processes data from a telescope, and stacks the data to create images of the celestial bodies seen in outer space. Siril is "specially tailored for noise reduction and improving the signal noise ratio of an image from multiple captures." Using Siril, we will be able to capture celestial bodies and enhance them to obtain high definition images.

The way Siril works is it takes in astrophotography data for pre-processing. During the pre-processing phase, the data is going under flat field correction as well as the correction of other basic imaging problems. The next step is stacking. Stacking is a technique used to increase the quality of an image by compiling multiple, roughly 10,000, layers of an image into a singular image. The process takes these images, aligns them over one another to reduce noise and increase the signal-to-noise ratio. Once stacking is complete, then begins the processing. During processing, the astrophotographer does whatever is needed to further enhance the image to their liking. Actions include but are not limited to color correction, noise reduction, and image sharpening.

From this information, we hope to gather significant data from the spectra and celestial objects we encounter using the telescope. This data includes star classification, temperature of the object, as well as the object's mass.

Siril is being used as the blueprint for our group's user interface because of its many advantageous and effective features. For example, Siril's calibration allows for quality image enhancement. Siril can utilize files for BIASES, DARKS, and FLATS. Siril is also great with analysis in that it can recognize and process the scientific values found within astronomical images. Siril is equipped to provide the necessary information we require from objects in space such as star brightness. It can also provide astrometry interactions between various astral objects within the image.

The program contains several pre-processing and processing algorithms designed specifically for astronomical imaging. Siril is great at processing deep-sky lucky imaging data because it supports a video file format known as "Sequence Extracted from a RAW" (SER) videos. All these features allow for multiple image processing from astral objects seen through the telescope. This is all done with a good storage capacity as well as an admirable processing speed.

The user interface designed for this project will contain inspiration from software programs such as Stellarium and Siril. These two programs are at the forefront of the design of the user interface because they are both great for locating stellar objects while collecting images and data from said objects.

The user interface that will be used in this project will contain few but necessary features. These features will include, but are not limited to, the ability to have input settings, mount control, live telescope view, image capture, and more. The study of multiple astrophotography software programs and the necessities of our project has allowed us to determine the tools we find most useful.

VII. GUIDING

Astronomy is highly noise sensitive and there are multiple techniques used to limit noise and increase maximum exposure time. Tracking is a primarily hardware based approach that focuses on moving the mount in a smooth and linear fashion. Guiding however is

a computer vision problem that is focused on interpreting image data to detect stars, correct tracking error, and translate data to the mounts controller via the INDI communication interface.

Guiding is used as the means of correcting mechanical error in the tracking processes. Tracking serves as a way of locking onto stars and enables us to have longer exposures. Tracking of our system works by manipulating the right ascension of the mount at 15.04 arcseconds per second. Tracking however will always be prone to errors caused by the mechanical components of a mount. Tracking error can be a result of things such as cable snagging, gear wear and tear and alignment errors can all lead to tracking errors. Guiding works by leveraging a guide scope which is a camera capable of live processing image data to correct tracking error. Within our system guiding will leverage the Raspberry Pi Compute 4, Raspberry Pi High Quality Camera, and PHD2.

PHD2 is an open-source project which can be leveraged via a python based client to allow for integration via a python GUI. PHD2's primary guiding feature set allows us to leverage star detection algorithms along with machine learning based predictive guiding algorithms. This allows for proactive error correction rather than reactive by leveraging known error values. The main parameters the user must set are the minimum movement parameters which are used as a means of avoiding small movement corrections which would not have effect on star shape. The next parameter is the hysteresis parameter which defines how long into the past values should be considered. The last user set parameter is the aggression which is a percent based value and determines how much of the actual expected correction the mount should actually do.

PHD2 guiding parameters are user set and can be corrected and modified via their live view as well as a correction graph which would show two different states. Overcorrection is the first state which is where jagged lines crossing over the Y-Axis occur. This is an indicator that the min move parameter is too great. Undercorrection on the other hand is described as a near vertical line displaying that mount commands issues are not being taken seriously enough this is counteracted by increasing the min movement command. These parameters are how we can decrease our guiding error and increase our signal to noise ratio.

VII. Astronomy Peripheral Control

Peripheral control plays a major role in automation of the spectrum acquisition. Astronomy equipment however such as the Celestron Nexstar 6SE, the ZWO camera and even the Raspberry Pi High Quality camera can only be interfaced with programs and controlled via code through use of INDI drivers. INDI is an open source driver level software used to automate and control astronomical equipment.

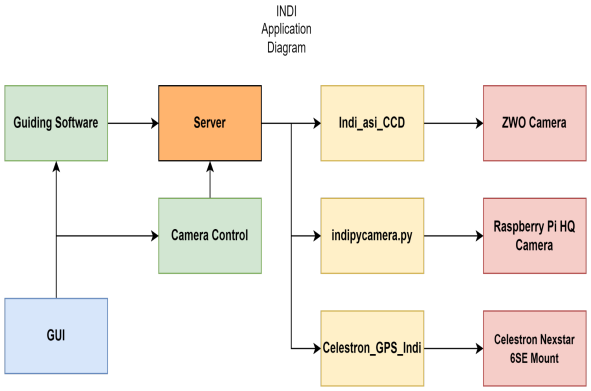

Fig. 5. The different layers of communication between the User Level Elements down to the Hardware.

Integrating the INDI library on the ARM architecture requires downloading the source library onto the Raspberry Pi Compute Module 4 and building it from source. It then requires virtualizing an environment in order to allow for the server to launch as Raspberry Pi OS disables allocation of environment variables outside of a virtualized environment. Finding the corresponding INDI drivers involves downloading them from their open source third party library and then building the required library files in order to activate them inside of the server. Next, comes the software level which handles calling the respective driver functions in order to control the astronomy peripherals.

PHD2 handles delivering instructions through the INDI architecture to the Celestron mount and to the Raspberry Pi HQ Camera. Python is needed however to properly interface with the ZWO camera. This software leverages the existing INDI drivers to enable us to create an intuitive user interface for controlling the camera. The primary functionality from the drivers that we will call via python is enabling the live view functionality, setting exposures, and controlling color balances and gain. The nature of the application also requires automated control of the camera which requires automated imaging.

IX. Spectrum Analysis Software

Spectrum analysis is performed on the Raspberry Pi Compute Module 4 and is a python based program. The GUI elements are established using tkinter which is a python based GUI maker. First, a canvas is established to hold the spectrum image which can be selected via the file option. Next, by pressing "p" placement mode is entered

which allows to determine the center of the spectrum. The center of the spectrum must be accurately determined in order to help with the bounding processes. Pressing “r” enables entering rotation mode which allows us to line up the reticle with the spectrum; this is necessary as the spectrum's orientation relative to the X-axis will not be the same. This is how the GUI elements work and define key parameters needed for analyzing the spectrum. The image will generally require scaling to fit displays as the native resolution of the image is 4144 x 2822. The scaling factor plays a major role in calculating the center point, rotation, and the overall mapping of the image. The scaling is done by taking the (size of the screen) / (original image size) this gives a scalar value which we can scale the image by.

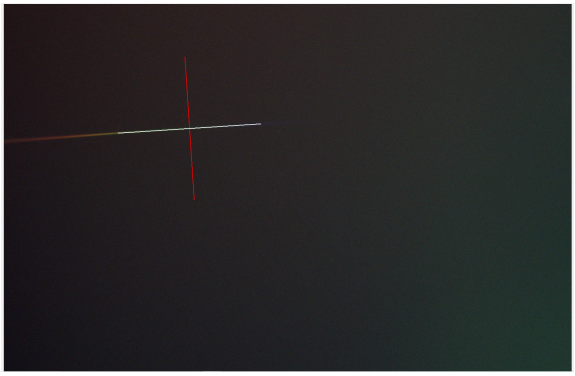

Fig. 6. The GUI for determining key parameters needed for spectrum bounding and processing.

Once the key features such as the center point and rotation have been defined the spectrum can be bounded. Bounding the spectrum is the most complicated of the processes as due to the smooth gradient of the spectrum there are very few methods that could work for defining consistent bounds. Assumptions must be made in order to accomplish spectrum analysis on stars. 1.) stars all provide the same size spectrum due to being the same size relative to their distance from the telescope. 2.)The distance on either side of the spectrum is the same from the center. 3.) The middle point of the spectrum will always be visible regardless of the star type. With these assumptions made, bounding becomes a more simple problem.

With the assumptions made spectrum bounding can happen first we take a 1-D profile simplifying image down to a single array with the size being the amount of pixels in the 1-D line. In the case that the line covers an area smaller than the full width of the image the rest of the pixels are filled with black pixels. This one 1-D array stores the intensity values and we will be using this array to determine the amount of indexes left and right of the center which we must look at to determine 400 and 750 nm.

Fig. 7. This image displays what the 1D profile looks like for the star rasalhague.

The 1-D profile’s values from the start and end index get extracted which allows us to graph wavelength vs intensity. This graph will allow us to display our intensity at each given pixel; these values don’t mean anything intrinsically but in a single star system their values follow a gaussian-like distribution. The simplified array that holds only the values between the start and end index is then divided into 250 equal sections signifying each different wavelength in their respective nm. Once the values have been grouped and averaged the peak location can be found by finding the first derivative of the best fit line and finding when the line's slope crosses 0. This wavelength value is then passed into Wein's law to determine the light source's temperature.

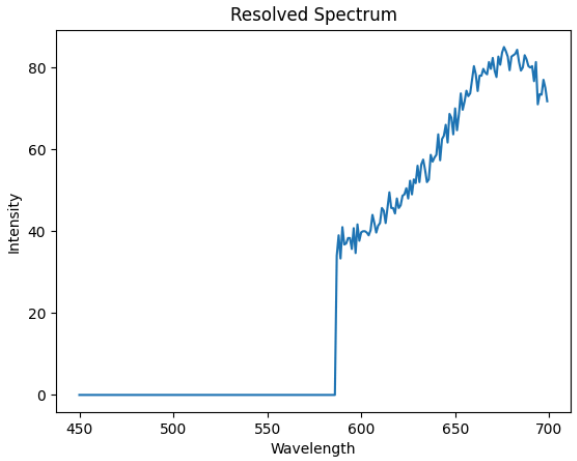

Fig. 8. The following is the results of the graphed spectrum once the pixels have been grouped and the plotted vs. their intensity values.

X. HARDWARE

In order to exercise control of our mount and in order to take in the camera data as input and analyze it, we will need a hardware platform upon which to execute these tasks. We have built a printed circuit board that has been designed around having the Raspberry Pi Compute Module 4 as its main processing unit. We simply needed to provide the necessary power and Input/Output to this unit in order to access all of its on-board utility needed to follow through on our projects engineering requirements.

The Raspberry Pi Compute Module 4 is powered by a quad-core ARM Cortex-A72 processor running at 1.5 GHz, It supports dual HDMI output with 4K resolution,

USB 3.0, and Gigabit Ethernet, ensuring high-speed connectivity. Additionally, the Raspberry Pi Compute Module 4 includes PCIe support, allowing for further expansion with peripherals. Its compact form factor and extensive GPIO make it ideal for our application. It provides an all-in-one package from which we can run Linux OS and develop software at a high level, which will be required in order to leverage existing spectrometry libraries and programs.

For our project we needed one HDMI output for our monitor so that we can show our GUI, four USB ports for keyboard, mouse, and other peripherals, two FPC connectors, one for the connection to our tracking camera and another for use as a data bus to a potential, but yet unrealized motor driver board for the building of a custom mount, and finally the custom I/O mount for the Raspberry Pi Compute Module 4.

For power, we elected to use three entirely separate barrel jacks for our input. One for the USB and other peripherals, one for the potential, but yet unrealized motor control board for a custom mount, and one for the Raspberry Pi Compute Module 4 itself. This was done for the sake of simplicity as the Raspberry Pi is very sensitive when it comes to power, so the more isolated we can make its power, the better, as well as for the reason that the unrealized motor control board would be drawing a significant amount of power compared to the rest of the board in order to drive the motors to move the mount. Future iterations of this board could potentially reduce this down to one, higher powered, power input with some more robust power control, but as of right now we have gotten the same functionality with these three separate inputs.

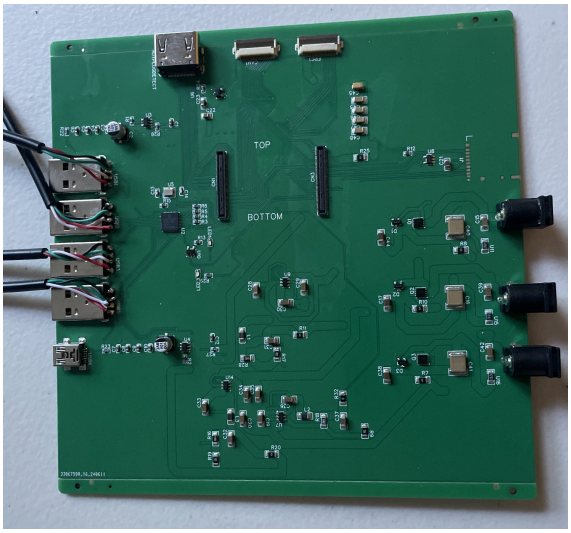

Fig. 9. Version 1 of our printed circuit board without the Raspberry Pi Compute Module 4 attached and peripheral female USB cords soldered into our erroneously placed male USB ports.

The second iteration of our printed circuit board made three main improvements. First, we corrected the placement of male USB ports with the placement of female USB ports. No wiring changes were needed as the USB port wiring was verified to be correct with the use of soldered on temporary female USB cords onto version 1 of our PCB. The second "quality-of-life" improvement was the addition of two power switches, one for the Raspberry Pi Compute Module 4 and one for the USB ports and the rest of the board. As well as some power indication LEDs for these switches. The third and most critical change for version 2 of the PCB was the correcting of polarity on three data lines on the FPC connector for the tracking camera. This enables us to actually utilize this second FPC connector port.

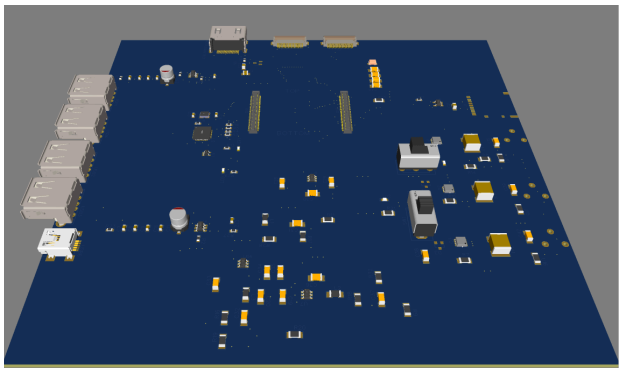

Fig. 10. 3D model of the second iteration of the printed circuit board.

XI. Test Star

To test the full-stack functionality of our product on actual stars, the weather must be optimal so that our guiding, tracking, and capture is unobstructed. The night sky must be "astro dark". The darkness of the night sky is measured in Bortle units, where Bortle 1 is perfect and optimal darkness and Bortle 9 is city lights at night. With some travel, we have deemed anything under Bortle 4 as an acceptable degree of "astro dark". The night sky must also be clear of all types of precipitation as there is a significant amount of risk of damage as our equipment (especially the printed circuit board) does not have strong water resistance. The night sky must also be very clear of clouds as whenever a cloud crosses between our capture equipment and the star we are attempting to capture, we must cease our exposure because the imaging is greatly weakened and the tracking becomes impossible.

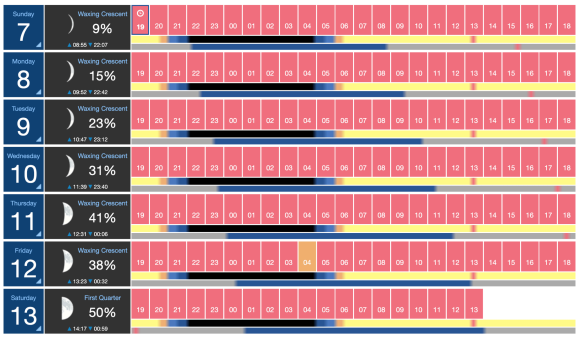

Fig. 11. Display of a weeklong weather forecast from clearoutside.com. Each row is one 24 hour day, with the red hours representing cloud coverage over 50% and green hours (if present) representing under 10% cloud coverage. The yellow-to-black gradient represents the brightness of the day, with the black being “astro dark”.

As weather has been exceedingly poor and difficult to manage, we needed a controlled test environment to ensure that our product is functional. For this we decided to build what we are calling a “Test Star”. Our test star essentially consists of two groups of LED’s one group is the control and is simply a singular white LED. From this, we were able to grab a clear rainbow spectrum of all visible wavelengths. The second group is a bundle of one red LED, one green LED, and one blue LED. When all of these are illuminated and viewed from a distance the Test Star appears white to the human eye, but to the spectrometer we can observe the three separate wavelengths coming through as well as the locational offset of each LED (which is similar to how Binary Stars behave).

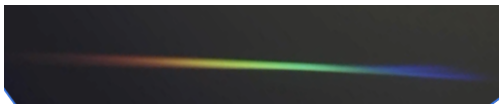

Fig. 12. The continuous rainbow spectrum of all visible wavelengths of light received from the one white LED.

Fig. 13. The discontinuous and offset spectrums of the red, green, and blue LEDs.

Our test star utilizes a 9V battery and a 9V to 5VDC converter before sending the 5V rail to four switches, these four switches then go to four 100K potentiometers which are then sent out to the red, green, blue, and white LEDs. As the red LED shows a visible lower brightness compared to the other three LEDs when attached to the same voltage, the potentiometers were added so that a degree of dimming could be achieved so that the clustered red, green, and blue LEDs could all be of similar brightnesses and thus appear to be a more accurate white.

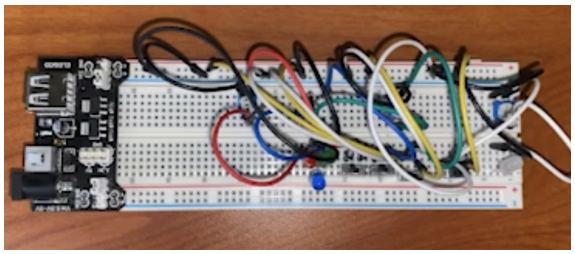

Fig. 14. The Test Star pictured here is wired into a solderless breadboard. Future Iterations will be on a soldered down prototype board.

It is important to note that the Test Star is a response to the difficulties that the weather posed when it came time for testing, for this reason the Test Star is not considered a part of the final product, but is a critical part of testing the final product. As it is not a part of the final product and is simply to test and reliably demonstrate, we are using prototype methods of actualizing circuitry rather than ordering and printing a circuit board.

XII. Conclusion

In conclusion, STSC-MS is an endeavor comprising four interconnected systems geared towards capturing and analyzing stellar spectra using the Morgan Keenan classification system. This approach will facilitate the determination of a star's position on the main sequence, offering insights into its relative mass, temperature, and elemental composition, while also enabling the identification of binary star systems. To achieve these objectives, while prioritizing simplicity and accessibility, we developed a suite of subsystems using specialized instruments such as: a spectrometer, tracking system, guiding system, and imaging system. Modular construction and 3D-printable parts engineered with broad tolerances ensure ease of assembly and customization, enabling amateur astronomers to tailor the system to our individual needs without necessitating high-quality 3D printing capabilities. Using these methods have allowed us to complete a functioning spectral telescope.

Acknowledgement

To the professors, mentors, friends, and family who have motivated and guided us to the end of our undergraduate journey. We thank you all for being a part of our experience at the University of Central Florida.