Silesian
University
of Technology

FINAL PROJECT

StarTrckr - night sky tracking device

Nikodem BARTNIK
Student identification number: 293666

Programme: Control, Electronic, and Information Engineering
Specialisation: Informatics

SUPERVISOR
dr inż. Krzysztof Tokarz
DEPARTMENT Graphics, Computer Vision and Digital Systems
Faculty of Automatic Control, Electronics and Computer Science

Gliwice 2023

Thesis title

StarTrckr - night sky tracking device

Abstract

With advancements in digital photography and high-end equipment becoming less expensive, more and more amateurs are getting into astrophotography. Usually, the goal is to capture objects of the night sky, Moon, constellations, or deep sky objects. Some of the work can be done with just a camera and a tripod but taking more advanced pictures of objects that are light years away require special equipment and knowledge. Usually, the equipment is expensive; thus, most amateurs can not afford it. This thesis aims to design and build an inexpensive, easy-to-build, and open-source star tracker. Relying on popular components that are easy to buy worldwide and 3D printed parts. The thesis focuses on the requirements, approach to design, and different fields of study connected to achieve the end goal and to test the final result. The main problems of astrophotography are introduced together with existing solutions on the market. Requirements, together with the justification for design decisions, are described. In the end, field tests are performed, tracked and untracked images are presented as well as potential future upgrades. The work is shared on GitHub for free, so anyone can start their astrophotography journey.

Keywords

astrophotography, star tracker, 3-axis gimbal, electronics, python

Tytuł pracy

StarTrckr - system śledzenia nocnego nieba

Streszczenie

Wraz z postępem w fotografii cyfrowej oraz coraz niższymi cenami sprzętu wysokiej klasy, coraz więcej amatorów zaczyna zajmować się astrofotografią. Zwykle celem jest uchwycenie Księżyca, gwiazdozbiorów lub obiektów głębokiego nieba. Do wykonania niektórych zdjęć wystarczy aparat i statyw, ale aby wykonać bardziej zaawansowane zdjęcia obiektów oddalonych o lata świetlne, wymagany jest specjalny sprzęt i wiedza. Zwykle sprzęt ten jest drogi i przez to większość amatorów nie może sobie na niego pozwolić. Celem tej pracy jest zaprojektowanie i zbudowanie niedrogiego, łatwego do zbudowania urządzenia do śledzenia gwiazd zgodnie z koncepcją open-source. Praca koncentruje się na wymaganiach, podejściu do projektowania, różnych dziedzinach połączonych ze sobą, aby osiągnąć cel końcowy. Na początku przedstawiono główne problemy astrofotografii wraz

z istniejącymi na rynku rozwiązaniami. Przedstawiono wymagania projektu oraz uzasadniono decyzje projektowe. Na koniec przeprowadzono testy w terenie, zaprezentowane są zdjęcia ze śledzeniem oraz bez a także potencjalne przyszłe ulepszenia. Efekty pracy są udostępnione bezpłatnie w serwisie GitHub, aby każdy mógł rozpocząć swoją przygodę z astrofotografią.

Słowa kluczowe

astrofotografia, star tracker, gimbal 3 osiowy, elektronika, python

Contents

Chapter 1

Introduction

Since the beginning of human existence, people have been "staring at the sky wondering why we are here and what happens when we die" (quote from a song by Will Varley [19]). As the years passed, humans still do not have answers to those two questions, but certainly, we got better at staring at the sky. While some of the boldest, innovative telescopes are built as ground observatories, some are launched not only to LEO (Low Earth Orbit) or MEO (Middle Earth Orbit) but even Lagrange orbits, far away from Earth. A small branch of astronomy has been developing over the years, where even amateurs with significant results can observe deep-sky objects. It is called astrophotography. Astrophotography is a relatively new field of study made possible thanks to advancements in digital cameras and systems taken from traditional telescope-based observations. It is a mix of technology, science, and art that lets us observe our universe from the backyard of the house.

A star tracker is one of the most crucial devices for astrophotography. Basic models allow to counter-rotate the camera around the Earth's axis of rotation to keep looking at the same stars while the Earth rotates. This technology is becoming more and more accessible but is still based on the same concepts and has not changed over the years. Why? The main reason is that those are simple, tested, and just work. While mechanical simplicity is a huge advantage of the most popular GEM (German Equatorial Mounts), those devices are not easy to set up, require special knowledge, and the process gets even harder in harsh weather conditions in winter. Usually, such systems also lack the go-to functionality - only available in expensive models. Considering the reasons mentioned, this project aims to solve the problem differently. With StarTrckr, the electronics, software, and mechanics result in a more complicated system that simplifies the user experience and offers additional features at an affordable price.

A solar day on Earth is 24 hours long (in fact a stellar day is only 23h 56min 4s long) that is how much time it takes for our planet to perform one complete rotation and that is causing the apparent movement of the stars also referred to as Diurnal motion. Pointing a camera at a deep sky object i.e. galaxy or a nebula on a tripod would result in the object

slowly (or quickly, depending on the focal length of the lens) “escaping” from the frame forcing us to reposition the camera to keep it in the frame. Light emitted by deep sky objects that are dozens of light years away is so faded that the camera sensor can barely pick it up. That is why in astrophotography long exposure pictures are always taken, and the light is detected by the camera sensor for a longer period of time (a few seconds up to a few minutes per frame). The two problems mentioned above make it impossible to perform any astrophotography operations on a tripod and special mounts often called star trackers are used. The job of a simple star tracker is to counter-rotate the movement of the Earth and keep looking at the same point in the night sky, some more advanced models also offer go-to functionality, that is automatic pointing to requested objects. Different types of star trackers are described in detail in the problem analysis.

This thesis aims to design a new and innovative star tracker that is mechanically similar to a 3-axis gimbal, frequently used nowadays for camera stabilization. Such construction, while being mechanically complex, allows for a more straightforward setup and calibration and makes it easy to implement a go-to functionality. Unlike popular alt-az mounts (altitude-azimuth mounts), it also solves the field rotation problem by implementing an additional axis of rotation. Software and hardware solve a lot of problems present in currently available star trackers, and as for the author’s knowledge, so far, such gimbal like approach has never been implemented in the astrophotography star tracker.

The scope of the thesis is to design a prototype of a complete solution that can be used to take images of deep sky objects with a popular DSLR or mirror-less type camera. That includes the mechanical design in CAD software as well as producing the parts for a prototype with an FDM 3D printer. Developing mathematical formulas used to control the position of the tracker and keep tracking the objects as precisely as possible. Software developed in Python programming language for a microcontroller as well as simple desktop-based software with a user interface. Custom PCB with RP2040 microcontroller and stepper motor drivers communicating with a computer through USB, executing the commands and controlling the position of stepper motors. Testing the complete system and taking tracked and untracked images for comparison.

Chapter 2

Problem analysis

Astronomy has always been very accessible as some basic night sky observations can be performed with the naked eye such observations are referred to as unaided eye observations. With some simple instruments like a binocular or a small and inexpensive telescope, we can take a closer look at the surface of the moon, some star clusters, bigger moons of the planets in our solar system, and our largest artificial satellite so far - the International Space Station. While with some more powerful binoculars, it is possible to observe the planets or even see the rings of Saturn in general a more powerful and expensive telescope is needed for such observations. These kinds of observations can be categorized as visual astronomy. Observing deep sky objects is out of reach of our eyes even with telescopes, the problem is not the size of the objects in the sky but the huge distance that the light emitted by them has to travel in order to reach our instruments. The Andromeda galaxy is the closest galaxy to Earth at a distance of about 2.5 million light years. It would look 20 times larger than the moon [6] in the sky if it could be seen with the unaided eye.

Another limitation of the human eye is that it can only observe light in a visible spectrum. Because our universe is expanding all the time the light emitted by the deep sky objects is shifted (Doppler effect) [7]. It might be red or blue shifted depending on if it is moving towards or away from the observer.

Modern developed societies introduced new problems to astronomy and the biggest of them all is light pollution. Huge agglomerations and metropolis with artificial lighting make it hard or even impossible to observe the night sky. Unfortunately, light pollution is not a local problem, city lights can be observed from several kilometers and in highly developed countries finding a place nearby with a dark sky may be hard or even impossible [11, 16]

That is why astrophotography and the combination of modern digital camera sensors with astronomical instruments are so powerful. By capturing the light for a long period of time with long exposure photography and combining multiple frames in a process called stacking we can extract the vital information and visualize the deep sky objects. Digital sensors are not limited like the eye is, therefore it is possible to detect any kind

of wavelength that is desired. Light pollution can be greatly reduced with special filters.

Currently available star trackers are based on the solutions that were first used as telescope mounts. Star trackers can be divided into 3 different types barn door tracker, alt-az mount, and equatorial mount.

2.1 Barn door tracker

Sometimes referred to as a "book" type start tracker. The most primitive home build single-axis tracker. Its axis is aligned with the Earth's rotational axis manually. Usually, there is no polar scope, only a simple hinge, but more complicated solutions are possible. This type of tracker can only track the sky and has to be manually pointed to a target. The manual alignment process might be overwhelming for beginners and is not easy to perform appropriately on the first try. While this solution is far from perfect, the simplicity and easiness of building make it a popular option for just starting amateurs.

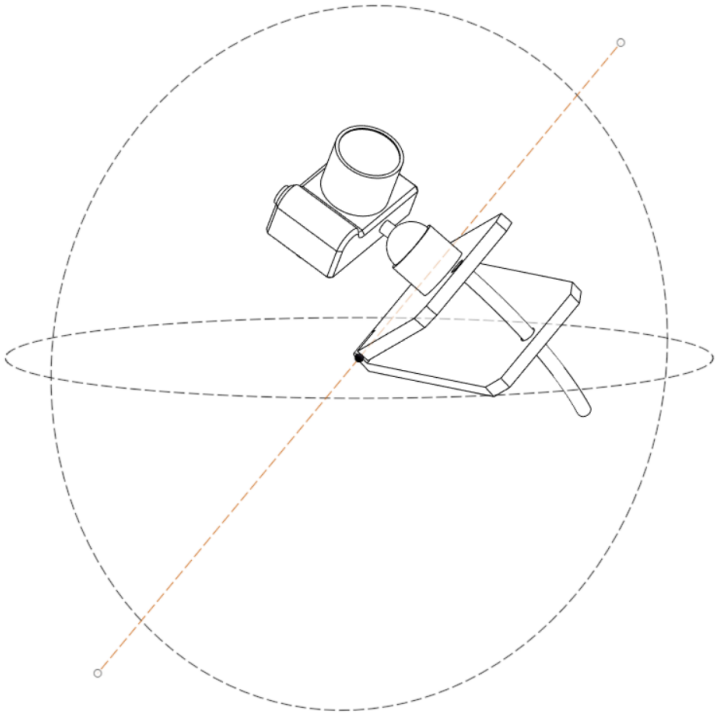

Figure 2.1: Simplified model of barn door star tracker

2.2 Alt-az mounts

Two-axis mount usually with a go-to system that can automatically point to any object in the sky. It does not require the user to polar align the mount, it just needs to stand on a leveled surface, and usually, the user has to point the tracker to a few objects in

the night sky so that the tracker can calculate its position. These systems are easy to use and often implemented in telescopes with Dobson mounts. One major disadvantage is also present because they are not aligned with Earth's rotation axis and have only two axes of rotation; they are exposed to the problem of field rotation. While those are capable of keeping the object in the frame during observation, the object slowly rotates in the frame. For that reason, exposure times are greatly limited.
To address this issue, special rotators [5] have been developed that are positioned between the telescope and the camera.

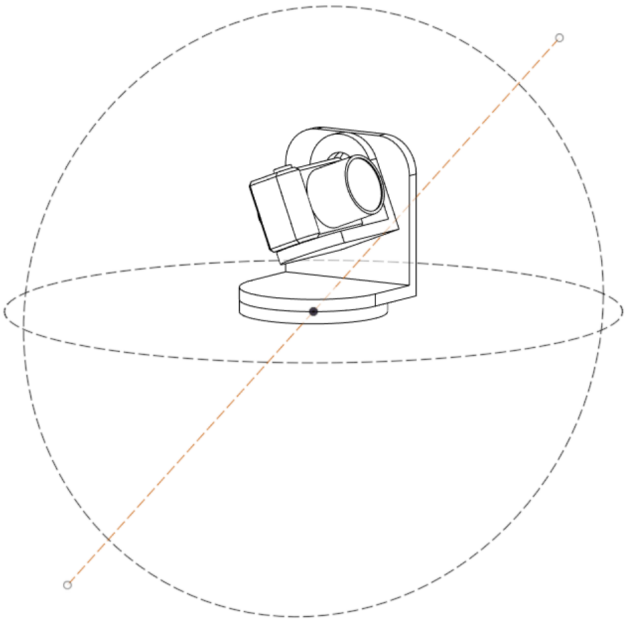

Figure 2.2: Simplified model of alt-az mount

2.3 Equatorial mounts

The most popular mount for telescopes. It requires alignment with a celestial pole. To realize that an additional optical element often referred to as a polar scope is needed. Equatorial mounts rotate only in two axes. There are fully manual mounts where the operator may control the position of the telescope with special knobs. Semi-automated trackers that feature motorized axis that compensate for the rotation of the Earth and allow for long exposure photographs. And lastly, full go-to systems with both axis motorized and usually an additional pendant controller or an interface that communicates with a computer. Equatorial mounts are very robust, often high quality, but also high price. The alignment process might be complex and overwhelming for beginners.

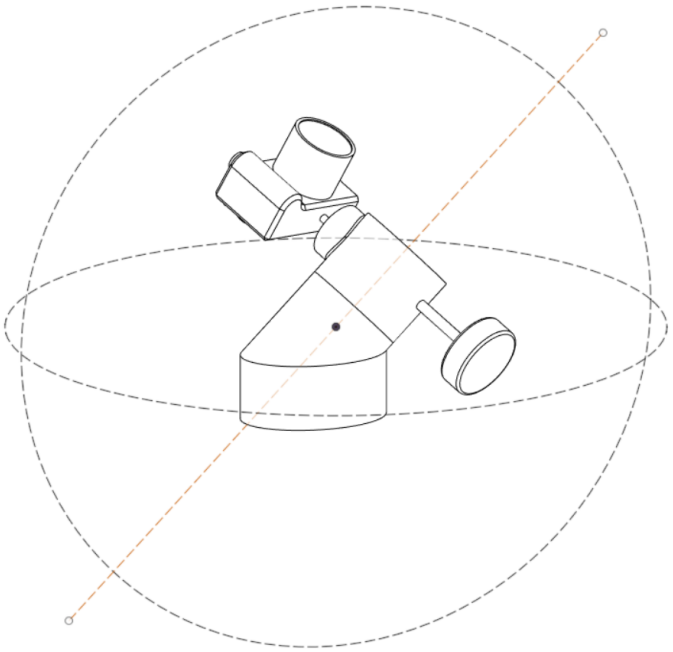

Figure 2.3: Simplified model of equatorial mount

While the solutions presented above are appealing due to their mechanical simplicity and reliability, a different approach can be used. Current astrophotography star trackers possess some limitations and are usually very similar in construction. As far as the author of the work is aware no 3-axis solution similar to the objective of this thesis has been developed. The most similar solution is the Alt-az mounts with an additional camera rotator attachment. Similar systems are often used in telescopes with Dobsonian mount, there is an additional rotator element that is rotating just the camera attached to the telescope to cancel the field rotation.

2.4 Literature research

Most literature that touches on star trackers refers to devices used on CubeSats (nanosatellites) used to determine the attitude of a satellite in orbit based on stars position [8]. While the topic is widely interesting it is only similar in name to the objective of this thesis. There are very few articles touching on the subject of astrophotography star tackers for that reason author also analyzed some articles touching on astrophotography in general.

2.4.1 In general about astrophotography

Astrophotography, for a newcomer, might seem overwhelming but also incredibly interesting. Getting accustomed to plenty of new terminologies as well as special equipment and all the physics that describe space events takes time and research. Also, special software-based techniques are required for post-processing and filtering the captured data. All of the problems mentioned above are briefly described in an article by Neil Adake [1]. Unfortunately, the author does not mention any hardware required to capture the images on either a professional or amateur level.

2.4.2 Simple tracking solutions

Some works focus on barn door trackers. The simplicity and limited number of pieces needed are the primary causes. This kind of tracker can be taken into consideration if one does not require further futures and is searching for a straightforward and affordable solution. Most of them consist of geared brushed DC motors or stepper motors for position control [12] but also manually controlled constructions have been developed [4, 2]. Only among armatures are manual constructions utilized due to the unreliability of human precision; nevertheless, since modern electric motor technology has advanced and become more affordable, these solutions are no longer practical.

2.4.3 More advanced tracking solutions

The majority of research on sophisticated star trackers that have been discovered focuses on software problems rather than the mechanical aspect of the issue. Software solutions include CV (computer Vision) systems for detecting the stars' movement and compensating for it [21]. The authors described in detail the software and electronics but have not realized the mechanical part properly. Unreliable stepper motors 28BYJ48 were used. Although motors are notorious for being extremely inexpensive, the internal gears generate a backlash that is so great that it prevents them from being used in any application where precision is essential. Interestingly, an additional laser-based solution was in development for testing purposes but was not used because of time constraints.

Other researchers focused on developing a simple solution using COTS (Commercial off-the-shelf) components to build a very affordable project both in terms of price and easiness of build [13]. In the article, we can find a very well-conducted market analysis and a brief description of all the available solutions for astrophotography star trackers. It is stressed that alt-az trackers have the simplicity advantage over equatorial mounts, but also the field rotation problem is mentioned as well as appropriate methods to solve

it. Mechanical design is rather primitive, small, and unable to handle a proper DSLR or mirrorless camera, that is caused by the design being adjusted for a mini astrophotography camera combined with a standard Raspberry Pi camera. Instead of relying on celestial pole alignment, authors used autoguiding camera that, combined with special software, can constantly calculate stars' movement and compensate for it. Electronics for the project was built on a breadboard - a solution acceptable for early prototypes but unsuitable for field testing. The article also lacks proper testing and verification of the final solution.

2.4.4 Existing commercial solutions

Since astrophotography becomes increasingly popular each year and more aspiring space explorers want to take their own pictures of deep space objects, there is a big variety of commercial star trackers available. Usually, bigger telescopes are sold together with an equatorial mount as a kit. Simple equatorial mounts like that are used just for positioning with special knobs. As the author of this thesis had an opportunity to use such a telescope, it can be confirmed that solutions work well, but polar alignment takes time and practice and additional equipment. Special motorized single-axis attachments are available in replacement for the knobs that enable tracking functionality. This kind of mounts are designed for rather big and heavy telescopes (for example, Sky-Watcher EQ3-2 that can handle a 5kg telescope, still small compared to professional solutions but very capable for an amateur [14]) and feature a heavy counterweight. The total weight of said mount is 25kg without the telescope, but it can be reduced by replacing a steel tripod with one made of aluminum by 10kg. This mount is made for telescopes but can successfully be used with standard cameras and lenses, often the cameras are attached to bigger telescopes anyway.

Sky-Watcher a popular manufacturer of space observation equipment, also designed a mount with astrophotographers in mind [15]. It is smaller and lighter than the bigger solutions used for telescopes but priced rather high at over 400 USD. It also lacks go-to features and has to be positioned at an object manually. Motorized trackers with go-to suitable for taking pictures of the night sky are also the same systems used in telescopes and for that reason are expensive. Prices start at high 1000 USD and can go as high as over 10 000 USD just for a tracker with a tripod not even mentioning cameras, lenses, and filters. Having a big and heavy tripod makes it stable, so it is definitely an advantage when taking pictures but not when carrying it through the forest or in the mountains to find a spot free of light pollution.

So far the only popular open-source star tracker project is the Open Astro Tracker [18]. It is an almost fully 3D-printed tracker meant for astrophotography with digital cameras. The project features a unique mechanical approach compared to traditional solutions. Unfortunately, the complexity and size make it far from an ideal solution. The

first iterations of this design used 28BYJ48 stepper motors which are unsuitable for any high-precision problem. Later OpenAstroTracker switched to Nema17 motors. Considering how it is built, carrying it around is definitely not easy. Since astrophotography requires a trip to a dark light polluted-free location for most people it is a big downside.

Another downside of the project is the latitude limitation. When building or buying this kind of tracker one has to decide within 10-degree precision at which latitude the device will be used as it only has 2 axes and still is inspired by traditional equatorial mounts. The project's open-source philosophy and the ease with which anyone may construct it by obtaining some basic electronic components from any supplier and 3D printing the necessary components are also major pluses. The project appears to be well-documented and being worked on continuously.

2.5 Mathematical problem formulation

Before the project began, it was not anticipated that tracking the objects with a non-polar aligned tracker and field derotation would be a challenging task to tackle. There were other approaches taken in an effort to find anything that worked, including trying to get an analytical answer using a simplified model in a CAD environment and Excel, as shown in figures 2.4 and 2.5, but no satisfactory outcomes were reached with this approach.

	At 45 deg angle			At 30 deg angle			At 10 deg angle			At 1 deg angle			At 60 deg angle		
Angle rela	A [deg]	B [deg]	C [deg]	A [deg]	B [deg]	C [deg]	A [deg]	B [deg]	C [deg]	A [deg]	B [deg]	C [deg]	A [deg]	B [deg]	C [deg]
-90	90	45	45	90	60	60	90	80	80	90	89	89	90	30	30
-80	76	44,1	35,1	70,6	58,5	43,3	44,6	75,9	35,4	5,7	80	4,7	78,5	29,5	24,3
-70	62,8	41,6	26,1	53,9	54,5	29,4	25,5	67,7	17,3	2,7	70	1,9	67,2	28	18,8
-60	50,8	37,8	18,4	40,9	48,6	19,1	16,7	58,5	9,4	1,7	60	1	56,3	25,7	13,9
-50	40,1	32,8	12,3	30,8	41,6	11,9	11,7	49	5,3	1,2	50	0,6	45,9	22,5	9,6
-40	30,7	27	7,5	22,8	33,8	7	8,3	39,3	3	0,8	40	0,3	36	18,7	6,1
-30	22,2	20,7	4,1	16,1	25,7	3,7	5,7	29,5	1,5	0,6	30	0,2	26,6	14,5	3,4
-20	14,4	14	1,8	10,3	17,2	1,6	3,6	19,7	0,6	0,4	20	0,1	17,5	9,8	1,5
-10	7,1	7,1	0,4	5	8,6	0,4	1,8	9,8	0,2	0,2	10	0	8,7	5	0,4
-1	0,7	0,7	0	0,5	0,9	0	0,2	1	0	0	1	0	0,9	0,5	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	-0,7	-0,7	0	-0,5	-0,9	0	-0,2	-1	0	0	-1	0	-0,9	-0,5	0
10	-7,1	-7,1	0,4	-5	-8,6	0,4	-1,8	-9,8	0,2	-0,2	-10	0	-8,7	-5	0,4
20	-14,4	-14	1,8	-10,3	-17,2	1,6	-3,6	-19,7	0,6	-0,4	-20	0,1	-17,5	-9,8	1,5
30	-22,2	-20,7	4,1	-16,1	-25,7	3,7	-5,7	-29,5	1,5	-0,6	-30	0,2	-26,6	-14,5	3,4
40	-30,7	-27	7,5	-22,8	-33,8	7	-8,3	-39,3	3	-0,8	-40	0,3	-36	-18,7	6,1
50	-40,1	-32,8	12,3	-30,8	-41,6	11,9	-11,7	-49	5,3	-1,2	-50	0,6	-45,9	-22,5	9,6
60	-50,8	-37,8	18,4	-40,9	-48,6	19,1	-16,7	-58,5	9,4	-1,7	-60	1	-56,3	-25,7	13,9
70	-62,8	-41,6	26,1	-53,9	-54,5	29,4	-25,5	-67,7	17,3	-2,7	-70	1,9	-67,2	-28	18,8
80	-76	-44,1	35,1	-70,6	-58,5	43,3	-44,6	-75,9	35,4	-5,7	-80	4,7	-78,5	-29,5	24,3
90	-90	-45	45	-90	-60	60	-90	-80	80	-90	-89	89	-90	-30	30

Figure 2.4: Angle of each joint at different inclinations generated with the simplified CAD model

In the end, the algorithm is created based on vectors that are rotated with a rotation matrix. The process could be improved and optimized but finding the most efficient algorithm was out of the scope of this thesis. Several attempts were made with different methods before deciding on the final solution. Finding bugs in the code and the algorithm took a significant amount of time and required developing simple visualization scripts in Python using MatPlotLib.

$$\vec{A} = [1, 0, 0] \vec{B} = [0, 0, 1] \tag{2.1}$$

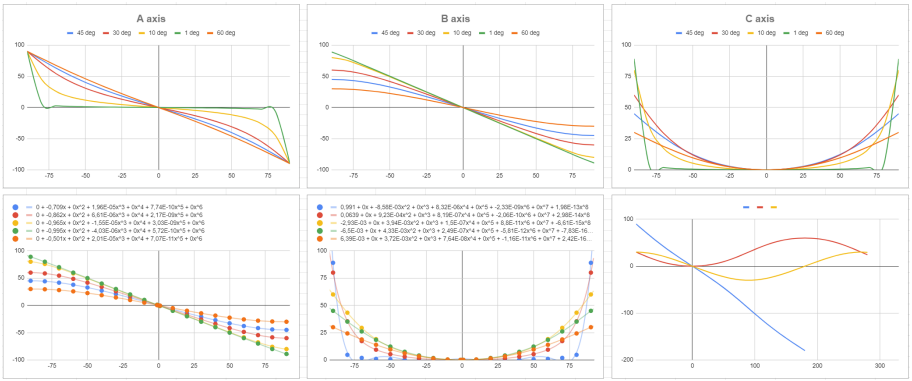

Figure 2.5: Plots created based on the values

Those two vectors are used as the origin reference frame for the manual rotation of the tracker. Once we align the tracker with the celestial pole, the vectors mentioned above are copied and rotated with a rotation matrix by the angles at which the tracker was rotated in order to align it.

$$\begin{bmatrix} \cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos\alpha & \cos\gamma\sin\beta\cos\alpha + \sin\gamma\sin\alpha \\ \sin\gamma\cos\beta & \sin\gamma\sin\beta\sin\alpha + \cos\gamma\cos\alpha & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha \end{bmatrix} \tag{2.2}$$

The vectors mentioned above are used as a new reference frame and as the axes of rotation to perform the tracking. A rotation matrix around a vector is used to track an object and rotate in all 3 axes simultaneously while looking at the same part of the sky.

$$\begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix} \tag{2.3}$$

And then using the following formulas the angles for each joint are computed based on the rotated copies of vectors A and B from 2.1. The angles are sent to the tracker as angles to be used for joints A, B and C

$$\alpha = \text{sign}(\mathrm{A_y})\arccos\left(\frac{\mathrm{A_x}}{\sqrt{\mathrm{A_x^2} + \mathrm{A_y^2}}}\right)\frac{180}{\pi} \tag{2.4}$$

$$\beta = \arccos\left(\mathrm{A_z}\right)\frac{180}{\pi} - 90 \tag{2.5}$$

$$\gamma = \text{sign}(\text{B}_\text{y}) \arccos{(B_z)} \frac{180}{\pi} \tag{2.6}$$

The plot depicted in figure 2.6 was made to aid in the visualization of the vectors and the resolution of issues that arose during the algorithm's development. Some issues were brought on by incorrect assumptions and notions underlying the algorithm, while others, like a missing element in the rotation matrix 2.3, were simple to solve but difficult to locate.

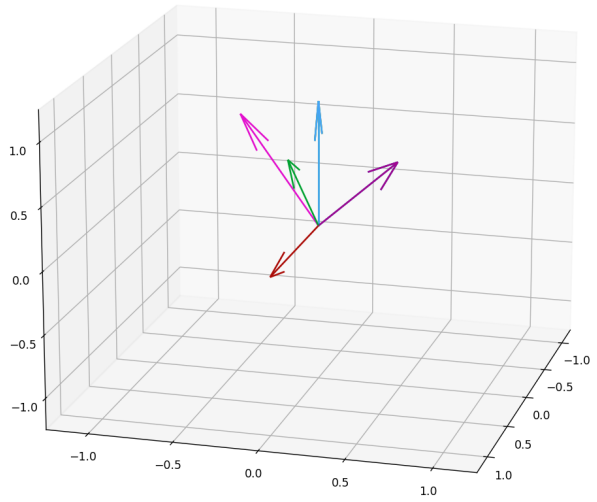

Figure 2.6: Plot showing vectors used to rotate the tracker. Red and blue vectors represent the X and Z axis of the reference frame, the green vector is the axis parallel to the celestial pole and purple and pink vectors represent the position of the tracker

Chapter 3

Requirements, tools and design process

Before the design process started, the requirements given below were established, and they served as the main parameters for all design choices made as the project was being developed. Tools used to develop it were selected based on the author's experience, knowledge, and their availability.

3.1 Requirements

Requirements were divided into functional and nonfunctional requirements and additionally divided into mechanical, electrical, and software.

3.1.1 Functional requirements

1. General
    (a) Project shall track the deep sky objects with enough precision to take a 1-minute exposure with a 100mm lens without noticeable star trailing
2. Mechanical
    (a) Shall be possible to point the camera to any object in the night sky
    (b) Shall allow for easy alignment with the celestial pole using the north star or compass application on the smartphone
    (c) Shall be possible to use the StarTrckr with any popular DSLR or mirrorless camera
    (d) 3D printed parts shall fit on a build plate that is 20x20x20cm
    (e) All screws shall be metric

3. Electronic
   (a) Shall be powered with 12V power adapter or an external 3S LiPo battery
   (b) All electronic components shall fit on a single side of a two-layer PCB that fits inside the project's printed case
   (c) Pancake stepper motors shall be used with 200 steps/rev
   (d) Main controller shall communicate with the computer via USB B cable
4. Software
   (a) Shall be easy to operate for the end user with a computer and Windows/macOS/Linux
   (b) Control software shall communicate with the project through a serial connection

3.1.2 Nonfunctional requirements

1. Shall be easy to build for everyone using a 3D printer and some easy-to-buy components
2. Complete easy-to-transport solution
3. Setup shall take no more than 5 minutes
4. Everything that is not required for the user to see shall be hidden
5. Nice to have feature: control with Stellarium - open-source planetarium software
6. Nice to have feature: tracking satellites

3.2 Tools and software used

Considering different areas that were connected in the project in order for it to be developed multiple tools were used. That includes software and hardware tools all of which are described below in detail.

3.2.1 Software

Fusion360

As a CAD software Fusion360 was used. The software was chosen based on the author's experience, the author of the software - Autodesk provides a free license for students.

FreeCAD was considered as an open-source alternative but it lacks some important features and is not as ergonomic as Fusion360. Additionally, special plugins make it easy to design various gears and mechanisms. Nice renders, animations, and simulation can be generated.

KiCad

KiCad was chosen as a schematic and PCB design software. It is a popular, open-source, multi-platform software that the Author of the work is familiar with. Eagle was also considered as a potential PCB design software but as making the project as open source as possible it was decided that KiCad is the most suitable choice. At the same time, KiCad enables easy preview of the PCB in 3D and export for production. It is also possible to export a 3D model of a PCB and import it into the CAD software and that functionality was used for designing the PCB case module. After developing the PCB the production files were sent to JLCPCB and an order was placed. Considering the price and quality at which the PCBs are produced as well as the very thin traces that were required it would be a great challenge to develop such PCB at home with simple DIY methods.

Thonny

Code was developed with Thonny IDE, a very simple environment recommended by Raspberry Pi to be used when working with RP2040. While easy to use in the beginning it lacks some features and for bigger projects, it is recommended to replace it with a more powerful one like Visual Studio Code. Since the firmware that was created is rather simple and small it was done in Thonny but further development and adding more functionalities might require switching to a more powerful IDE.

Visual Studio Code

Visual Studio Code was used to develop the desktop application for controlling the tracker. It is a lightweight all-platform IDE perfect for developing any kind of software.

Python

The author of this thesis has the most experience of programming in Python, it is also one of the fastest languages to develop software these days. An additional factor was the possibility to use the same programming languages for both firmware and desktop application. Python is also easy to read, understand and modify for beginners.

GitHub

GitHub was used to store files for the project and share them with the world. It also was a tool used to keep different versions of the project and keep track of the changes. The GitHub Desktop application was used to push and pull new files from the server. The repository with the project is publicly available, people can download the files but also contribute to the project and add potential issues or requests.

3.2.2 Hardware

3D printer - Ender3

Ender3 is a great example of one of the most popular 3D printers. Its build plate size is similar to another popular printer - Prusa i3 so it was verified that parts can be printed with the most popular printers. Parts were printed with PLA material.

Soldering station

A soldering station was used to solder the PCB. Both standard and hot air modules were used to solder all the components and fix potentially bad solder joints or shorts. Soldering such small components, especially a very small microcontroller required great precision and some fixes in order to get it right.

Soldering microscope

As mentioned very tiny SMD components are not that easy to solder and often a visual inspection is required to confirm proper alignment and solder joints. For that reason, a soldering microscope was of great use in the project. It was used during soldering, but also as an additional verification tool before powering the PCB.

Oscilloscope

Oscilloscope helped to check signals going from the PCB to the stepper motor drivers and the noise in the power lines to confirm that there are no unexpected behaviors or problems.

3.3 Methodology of design

There was no one clear obvious methodology of design for the whole project. It started similar to the waterfall method with the most time-consuming problems like PCB design (because its manufacturing had to be ordered and shipped from China) and CAD design (printing that many parts take a significant amount of time). Later as the project

was progressing and especially when the software development was started the methodology changed into Agile. Changes were implemented very quickly and prototypes were generated and tested either with the graphic simulation prepared with MatPlotLib or a physical prototype. Sometimes parts of the program were modified during field testing and committed on GitHub to keep track of the changes. Later in a similar manner, new versions of mechanical construction were developed. Iterations were very fast and allow for quick improvement and testing of the new designs. 3D printing was of great use in the project and proved to be an outstanding rapid prototyping technology. Different versions of the parts could mostly be tested on the same day they were designed and by the end of it modified to be printed at night and tested the next day.

3.4 Design process

Since the project was quite complicated and required connecting different fields of study (electronics, mechanical design, programming, mathematics) in order to achieve the final goal it was divided into stages. Some were overlapping and developed in parallel and the hardest part was integrating everything at the end and testing.

3.4.1 Mechanical design

The project started with paper sketches of different mechanical approaches. Some of those are presented in figure 3.1. It speed up the design process and allowed for dimension-free design and quick iteration through ideas without worrying about proper dimensions and maneuverability.

Once the ideas were developed a design in the CAD software was started. Everything was created with 3D printing in mind from the outset. The possibility of replacing some components with 3-4 mm laser-cut and bent sheet metal was also considered, and this could be one of the most significant future improvements. The majority of mechanical prototype and idea validation was done in CAD, although certain problems were not discovered until after the parts had been printed and the assembly or programming had begun. Many parts had to be slightly modified and printed again as a result. Three comprehensive solutions were created in the end.

Different designs are labeled v2, v3, and v4 as v1 was started but never finished as the wrong approach was used, parts were bigger than necessary but there was not enough space for the third axis and it would not be possible to print them with 20x20x20cm build plate 3D printers which apparently are the most popular. For that reason as well as new and better concepts for new versions the design of the first one was never completed.

As seen in figure 3.2 v2 and v3 are very similar, motors are flipped outside of the tracker to enable a bigger range of motion and crucial parts are significantly reinforced to

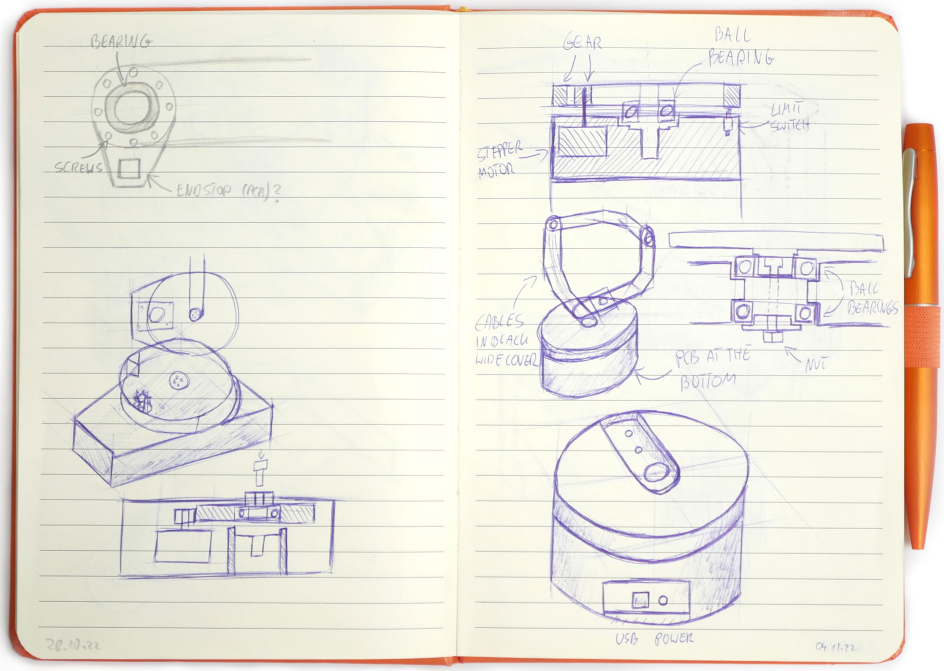

Figure 3.1: First paper sketches of the mechanical design

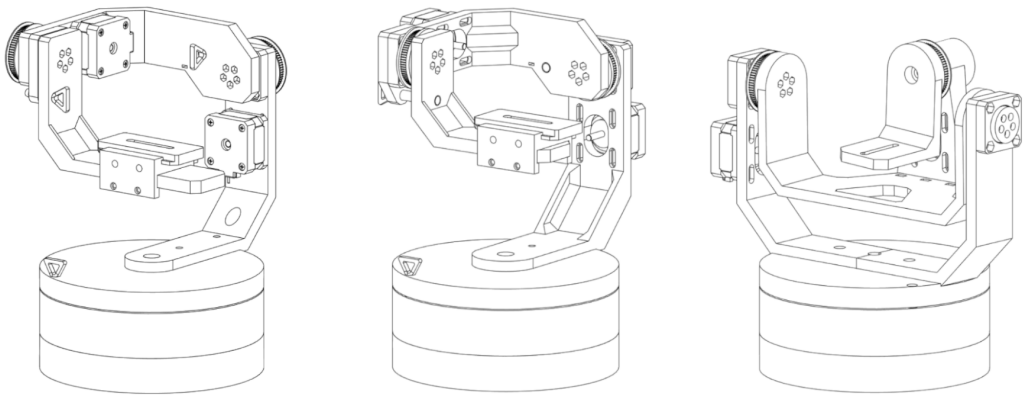

Figure 3.2: Wireframe view of three different designs of the mechanical construction (from left: v2, v3 and v4)

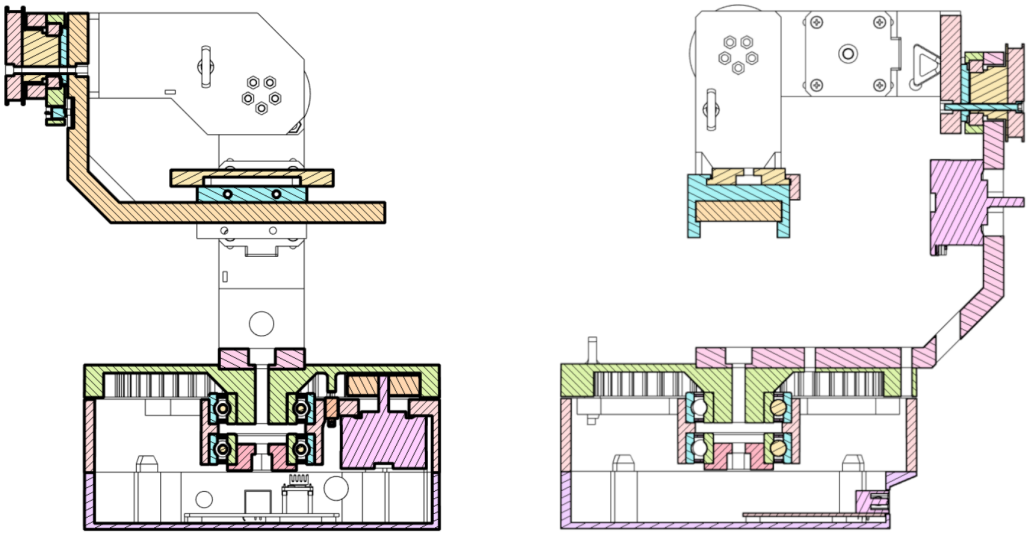

Figure 3.3: Section views of v2 StarTrckr

limit flexing of the parts when the camera is mounted. Parts were printed with PLA so by adjusting the design like this we can only limit the flexing, using a different material like PETG could potentially help but not much. In order to eliminate flexing completely arms would have to be made out of metal. First, two completed designs are heavily inspired by gimbals used for filmmaking. Version 4 is the biggest of all and features an additional arm that supports one of the joints on the other side to reduce the bending.

3D printing was a big part of the project and replacing as many parts with 3D printed alternatives was one of the goals from the beginning. For that reason, gears and custom pulleys for the GT2 belt were printed as well as bearing housings and shafts. GT2 pulleys were generated with an online tool as DXF files and converted to a 3D object in a CAD environment. Special notches and holes were designed to accommodate M3 and M5 nuts and simplify the assembly as well as make it more appealing to the end user by making it cleaner and minimalist. Still, pulleys with belts and motors are exposed for easier maintenance, problem-solving, and reduction in printing required for the project.

Standard metric bolts and popular bearings were used that are easy to buy despite the location. In case of parts unavailability, the design can be adjusted easily for the new parts.

For position control, stepper motors were used. Because of the lack of space and weight reduction, special pancake-type motors were chosen. This type of motors have reduced height, a feature convenient in compact designs like this that do not require significant torque from the motor. The motor shaft is 5mm in diameter and that allows to use popular small 16 teeth pulleys that would be hard to print reliably and would also wear out very quickly. Pancake-type motors are available from different manufacturers, so it should not

be hard to find alternatives.

Printed parts do not possess significant stiffness, so while being suitable for a prototype and proof of concept validation, those are not ideal for a final solution. Some parts, especially the tracker's arms that hold the camera, can be redesigned and adapted for steel laser cutting and later bent with a metal bending machine. That would improve the project and reduce the price of components at scale since 3D printing is very time-consuming, unlike laser cutting.

The project's final mechanical components were released as 3MF files, a new file format that is gaining popularity and enables simple 3D printing and editing with any CAD program.

3.4.2 Electronics

As of writing this thesis, there are big problems with electronics shortages. Plenty of parts are unavailable, and the prices are going up every month. That made the author rethink the electronics part of the project and look for an alternative solution to the problem. Various microcontrollers have been considered for the project starting with popular ATmega chips, ESP32, or STMs. In the end, a very new yet promising solution was selected - RP2040 from Raspberry Pi. It is the first microcontroller developed by Raspberry Pi, the popular producer of SBCs (Single Board Computers). It had three key advantages it was available at a low price and can be programmed in Python. Moreover, the author, with great success, used it in other projects and proved its reliability.

Choosing a proper stepper driver was another important aspect of electronics. Because of the emphasis on low end price of the project, it was assumed that popular in 3D printers A4988 drivers, could be used. These enable easy driving with STEP and DIR pins as well as microstepping up to 1/16 of the full step. As an alternative backup solution, more expansive TMC2209 drivers with a handful of additional features and microstepping up to 1/256 of the full step have been considered. PCB design was prepared for an easy swap of those drivers. It was planned to start testing with the cheapest possible solution (A4988 driver) and later replace it if needed or just for comparison.

Additional electronic components have been chosen according to the needs of the project, the parts list is available in the internal specification. A slot for the SD card was added just in case but in the end, it was never used. At the PCB design stage, the mechanical construction was not yet secured so a screw terminal header was added as power input for flexibility. A proper DC connector mounted on the PCB shall be used and it was temporarily implemented outside of the 3D printed case.

3.4.3 Programming

Programming is divided into the embedded and desktop parts. In both cases, Python was used as a programming language which simplified the development.

Embedded programming

Python is not a popular choice for embedded programming, but with new microcontrollers and development boards, its popularity is growing. This language was chosen despite its slow speed since it is interpreted in favor of speed of development, and thanks to using the same language for the desktop app it does not require context switching between different languages. Code was divided into different modules and classes and will be described in detail in internal specification 5.

Desktop application

The initial concept was to use Stellarium [17] - open-source planetarium software for controlling the tracker, problems faced during development of the project delayed it enough that there was no time to find a way to integrate the StarTrckr with this software. Stellarium can be used with equatorial and alt-az mounts, but after brief research, no information has been found that it can output the field rotation value that is required for a three-axis system like this. If this is the case, the issue can most likely be overcome by calculating it based on the tracker's current position and writing a special plugin, but time constraints did not allow this development to happen. For those reasons, custom simplified Python software was created to control the project.

Packages NumPy, PyGame, time and Serial were used to develop the application. Additionally, custom classes and modules were written to simplify programming, reuse the components, and create easier-to-maintain code. The program's primary function is to create an interface between the human and the tracker. It enables changing the tracker's position, pointing to a desired location, reading the current position, setting the axis according to the celestial pole, and enabling and disabling the tracking.

An additional script and a class for visualization were written that allowed to create an interactive plot showing all the vectors required for the algorithm, and that made debugging and verification easier.

Chapter 4

External specification

4.1 Hardware and software requirements

The desktop controller should run on any modern computer with Windows, Linux, or macOS. It is important to check if Python 3.8 or newer is installed. It does not require any special hardware other than a standard USB port so that the StarTrckr can be connected to the computer. For the tracker itself, an additional external 12V power supply or a battery is required as the system does not have a built-in power system. 3S LiPo or a 12V gel battery can be used. While using the tracker, it must be placed on a leveled surface, that can be just a flat piece of ground or a table. Using a foldable table is recommended as it simplifies using the tracker as well as the computer outside and asses comfort for long sessions.

4.2 Build instruction

The following instruction will describe the build process and parts required to build prototype v3 of the StarTrckr. All versions are very similar, and there are some minor differences also names of the printed elements might differ slightly.

4.2.1 Parts

Mechanical

Table 4.2.1 presents all parts that can not be printed and have to be bought separately

Electronics

Additionally, a 12V power supply that can output at least 2A is needed to power the tracker. To use the tracker outside where there is no access to mains electricity an external

Name	Amount	Description
Stepper motor 17HS08-1004S	3	Pancake type stepper motor, height: 20mm
6205 bearing	2	ball bearing
6804 bearing	2	ball bearing
GT2 belt 200mm	2	6mm wide
GT2 pulley	2	16 teeth
M6x60 screw	1	
M6 nut	1	
M5 screws	a lot	lengths and amounts depend a lot on the version you are building
M5 nuts	a lot	
M3 screws	a lot	lengths and amounts depend a lot on the version you are building
M3 nuts	a lot	

Table 4.1: List of mechanical parts for the StarTrckr project

Name	Amount	Description
RP2040	1	Raspberry Pi microcontroller
A4988	3	stepper driver, TMC2209 can be used as an upgrade to increase the accuracy
W25Q32JVSS	1	Flash memory
USB socket	1	type B
LM1117	1	3.3V linear voltage regulator
LED	3	size: 0805 preferably different colors for each
Buzzer	1	THT, 12mm
12MHz crystal	1	size: 5032
27 Ω resistor	2	size: 0603
100 Ω resistor	3	size: 0603
1k Ω resistor	2	size: 0603
10k Ω resistor	1	size: 0603
27pF capacitor	2	size: 0603
1uF capacitor	2	size: 0603
10uF capacitor	3	size: 0603
100nF capacitor	10	size: 0603
2 pin screw terminal	1	
Some pin headers		female and male, raster 2.54mm

Table 4.2: List of electronic components required to assemble the PCB

Name	Amount
arm1.3mf	1
arm2.3mf	1
arm3.3mf	1
base_pcb_box.3mf	1
base_motor_holder.3mf	1
base_gear.3mf	1
base_bearing_cap.3mf	1
base_gear_small.3mf	1
gt2_pulley.3mf	2
shaft.3mf	2
shaft_cap.3mf	2
bearing_flap.3mf	2
quick_release_plate.3mf	1
quick_release1.3mf	1
quick_release2.3mf	1
pin.3mf	3

Table 4.3: List of files that must be 3D printed

battery must be used that can provide 12V.

4.2.2 3D printing

Files available in the GitHub repository are in 3MF format and that makes it easy to prepare a gcode with any slicing software as well as import to any CAD software and edit. It offers more and is more flexible than popular STL files. Table 4.2.2 presents all the files and the amounts that have to be printed. Parts were printed with PLA, PETG also can be used. It is recommended to use at least 40% cubic infill with 3 perimeters to ensure that the parts are strong. Another recommendation is to use a high-contrast color to print the pins so that they are visible and hard to forget when using the device.

4.2.3 Assembly

With all the parts prepared for assembly, we can start assembling the project according to the following pictures. Those are simplified drawings of the subassemblies and do not include screws and nuts for clarity.

Assembly shall be started with the base as presented in figure 4.1. At the very bottom element of the base PCB shall be attached, as mentioned the prototype does not have a DC power jack on the board so an external one should be attached to the screw terminal and attached to the case on the outside. Later motors have to be connected to the PCB so we will need access to the bottom part. Two 6205 bearings shall be placed in the base_motor_holder as well as the motor with small printed gear attached. M6 nut

pressed into base_bearing_cup shall screw together arm1 with the bearings mounted in the motors holder part with M6x60 screw. While it is not presented on the drawing arm1 shall also be fixed with the M6 screw.

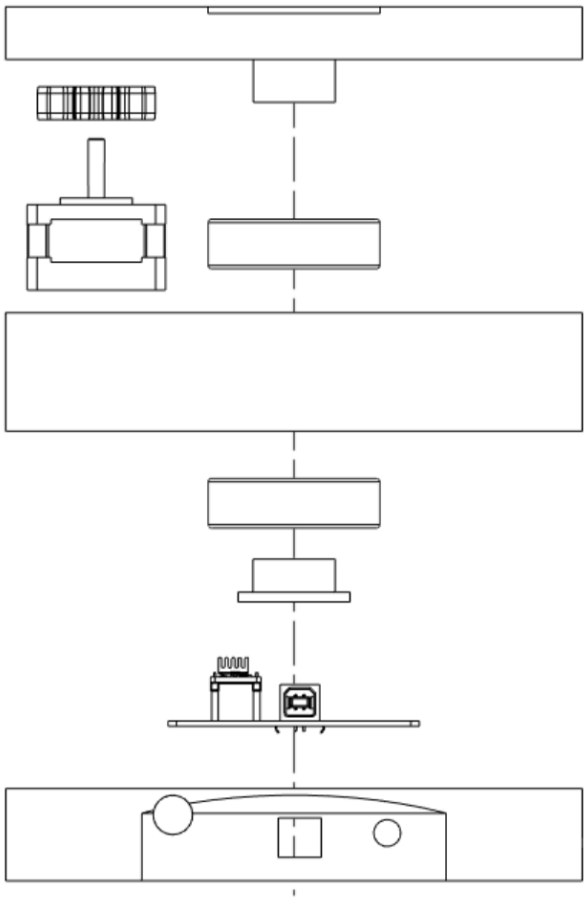

Figure 4.1: Assembly of the base of the StarTrckr

Printed GT2 pulleys together with shafts must be assembled with 5 M3 screws each 4.2. After placing 6804 bearings in the bearings flaps those can be attached with M5 screws and nuts.

Locking pins are designed to simplify transporting the tracker and make it safer 4.4. At the same time, those are used as a very simple system for homing the tracker. Pins must be removed after powering the tracker on. That way the zero position will be set.

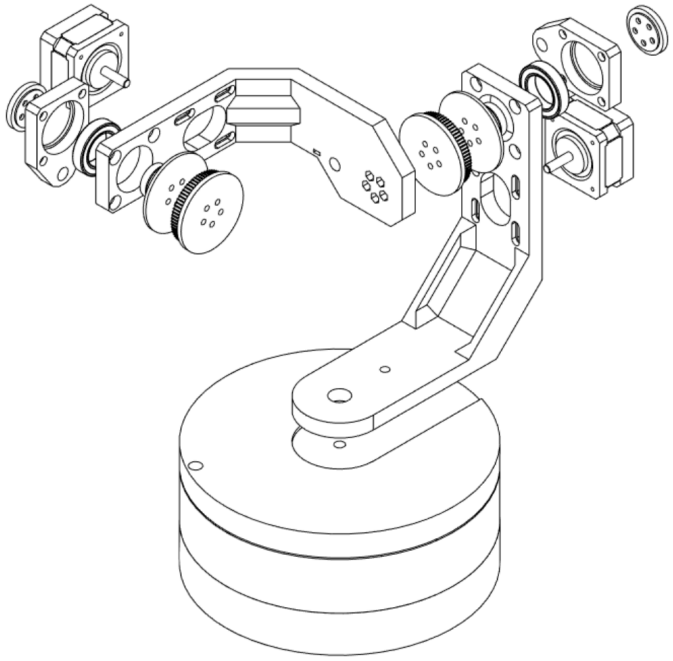

Figure 4.2: Arm 1 and arm 2 assemblies

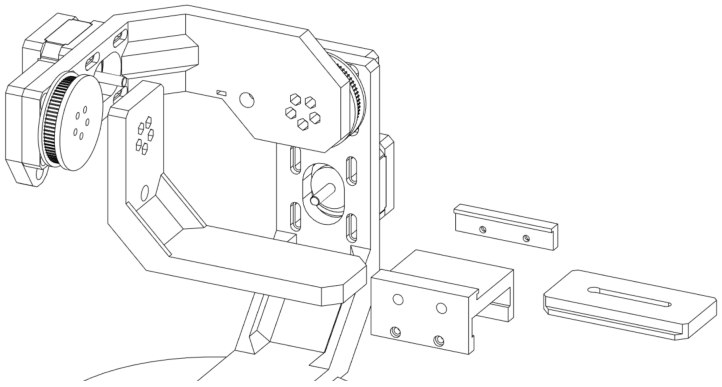

Figure 4.3: Arm3 and quick release plate assembly

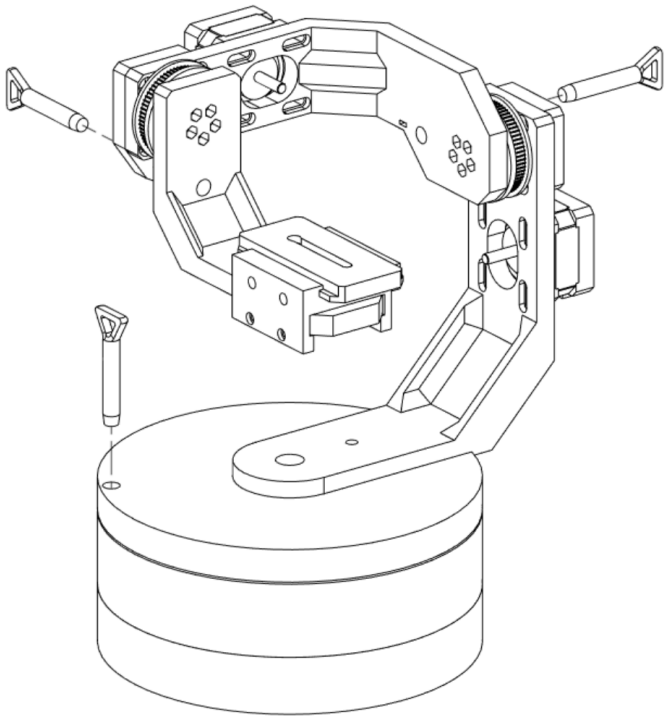

Figure 4.4: Axis locking pins assembly

Figure 4.5: Subassemblies before integration

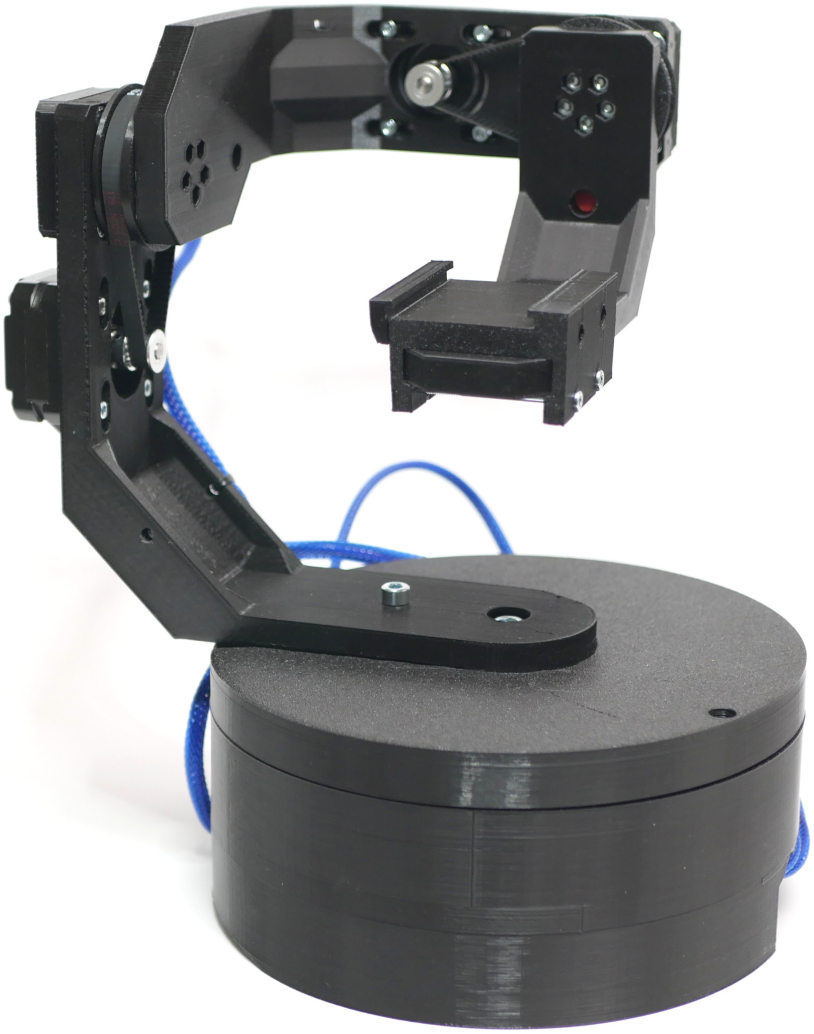

Figure 4.6: Completed assembly

4.3 PCB assembly

PCB manufacturing was ordered at JLCPCB.com but it had to be hand soldered 4.7. To reduce size, cost and simplify soldering SMD components were used in the majority. Soldering was started with the smallest components like resistors and capacitors and was performed with soldering paste applied with an SMT stencil, components were placed by hand and then soldered with hot air soldering iron. Soldering the RP2040 microcontroller required some rework and the microscope 4.8 was a very useful tool to check the soldering joints and confirm that it is properly soldered before powering up the circuit. Through-hole components were soldered with a normal soldering iron. To assure easy interchangeability of stepper drivers, female pin headers were used 4.9.

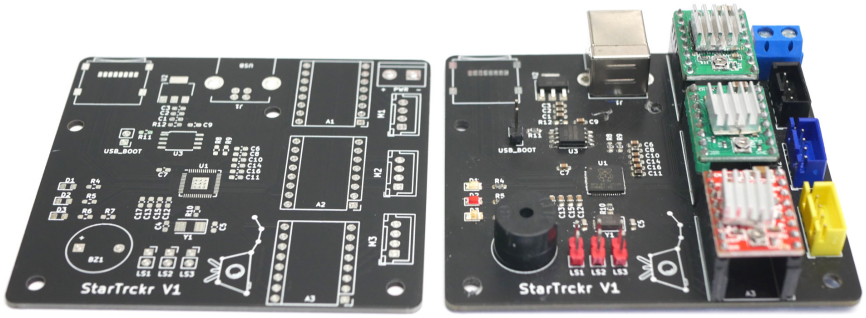

Figure 4.7: PCB before and after assembly

The unavailability of some components required changing the plans and for that reason, the SD card slot was not soldered on the PCB. In fact, it was added as an additional feature that could be implemented in the future and there was no use for it at this point of the project. After testing it was found that additional storage on the PCB is not necessary as the PCB is at all times connected to the computer for that reason its internal storage can be used if necessary. Shortage of components on the market was the reason why W25Q32JVSS flash memory was not available and for that reason, a more expensive module that utilizes this memory had to be purchased and the chip was unsoldered from the PCB and soldered to StarTrckr PCB. First tests of the completed PCB were performed with a laboratory power supply with a current limit setting set to 50mA to ensure that in case there is a bad soldering joint the components will not be burned. Later python interpreter was installed according to the tutorial provided by Raspberry Pi [10] and a simple Python script was uploaded to test the LEDs and buzzer.

Figure 4.8: Soldering the PCB with a microscope

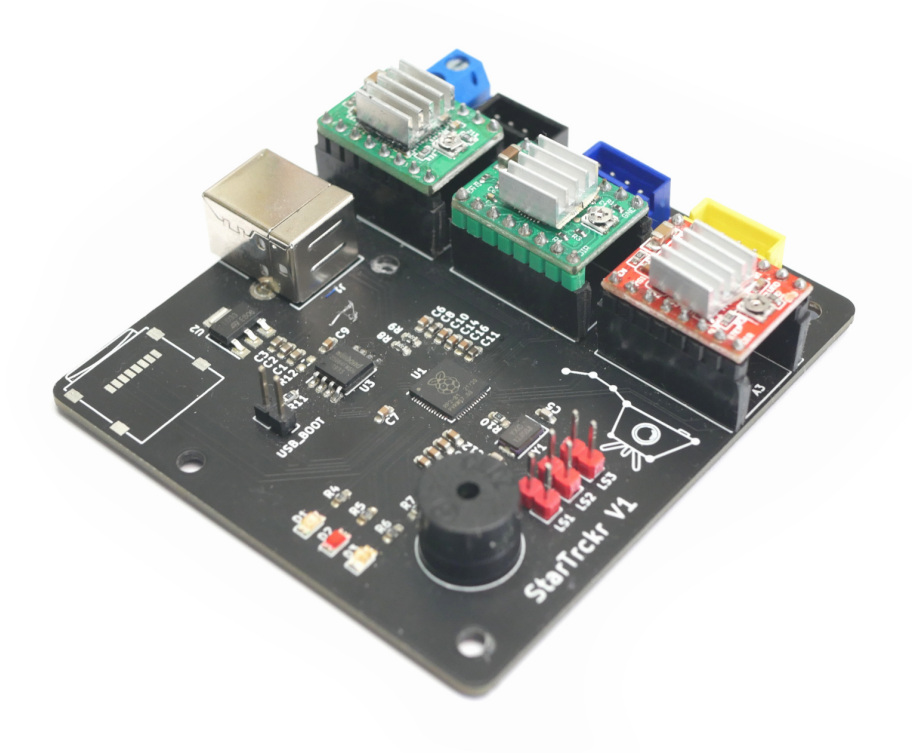

Figure 4.9: Assembled PCB with A4988 stepper drivers installed

4.4 Installation procedure

To use the software it is necessary to clone the following GitHub repository:

`https://github.com/NikodemBartnik/StarTrckr`

After navigating to the controller folder in the repository's main directory following command can be used to run the program:

```
python controller.py
```

If there are errors displayed when running the application it might happen that required libraries are not installed or require newer versions. All required libraries can be installed with the command

```
pip install pygame numpy pyserial
```

If the visualizer part of the software is needed additionally MatPlotLib has to be installed with the command

```
pip install matplotlib
```

4.5 User manual

The instruction manual focuses on how to use the StarTrckr and does not touch on how to properly perform astrophotography and all the steps necessary to capture great images or stack them together. Links to the articles explaining astrophotography were mentioned in the 2.

4.5.1 Planning the observation

The first step is to find not only a proper place where we will not disturb anybody, a place that is as free of light pollution as possible but also to arrange the observations at a proper time and weather. Clouds appearing in the frame while capturing the images will drastically impact the images in a negative way. Capturing objects that appear close to the full moon in the sky will be rather difficult. Temperature is of great importance too, if too low it might not only be hard to withstand for the operator but also for the equipment especially StarTracker's, laptop's and camera's batteries. Street lights as well as car headlights flashing directly into the camera's lens also distort the images and influence the final result. It is also worth mentioning that in some places the atmosphere

is polluted not only with light but also with small particles and smoke, especially during winter. Effects of light pollution can be reduced with special light pollution filters mounted between the sensor and the lens or like a standard photography filter screwed to the lens. Ventusky is a very useful tool to check current and future weather conditions. It is a web-based map of current and future weather conditions. It allows seeing temperature, clouds, winds, rain data, and so on put on a world map.

4.5.2 Preparation

Preparing for the observation includes packing all the necessary equipment: laptop, StarTrckr, batteries, and cables. Additionally, a folding table and a chair might be required to ensure comfort during observation. Once arrived at the observation destination the first step is to level the table. Usually, a compass application on a smartphone has a level built in, compass app will also be useful in the next steps. With the astronomy app installed on a smartphone, one can check the position of the object of interest before starting the observation. Before mounting the camera flip out the screen if the camera has one, once the camera is installed the screen will collide with the tracker 4.10. It is also possible to use an external monitor for convenience and easier focusing with a bigger screen 4.11. The very last step of the preparation is installing the camera and securing it with screws that are installed in the quick-release mount 4.12.

Figure 4.10: Flip out screen of the DSLR after installing the camera

Figure 4.11: External monitor connected to the camera

Figure 4.12: Installing the camera in the quick-release mount

4.5.3 Connecting with desktop application

Before connecting the StarTrckr to the computer it shall be checked that the locking pins are in position because that is the way of setting a zero point on the tracker. Once the battery or a power adapter is plugged to the tracker as well as a USB cable between the tracker and a computer we can check the serial port name in the code. To do that we have to open the controller.py file and modify line number 9:

```
SERIALPORT ='/dev/cu.usbmodem14101'
```

With a serial port name that the StarTrckr was detected as. In the case of macOS computers, it will look similarly as it does in the example above in the case of windows computers it will be COMX (where X is a number). After saving the modified file we can start the application with the following command and remove the locking pins.

```
python controller.py
```

After running the program user will see the screen presented in figure 4.13. The big circular element is the joystick used to change the altitude and azimuth of the tracker. With "L" and "R" buttons third axis of the tracker can be rotated. The slider below the joystick adjusts the step and the buttons "set axis" and "track" set the celestial pole axis and start tracking the sky accordingly.

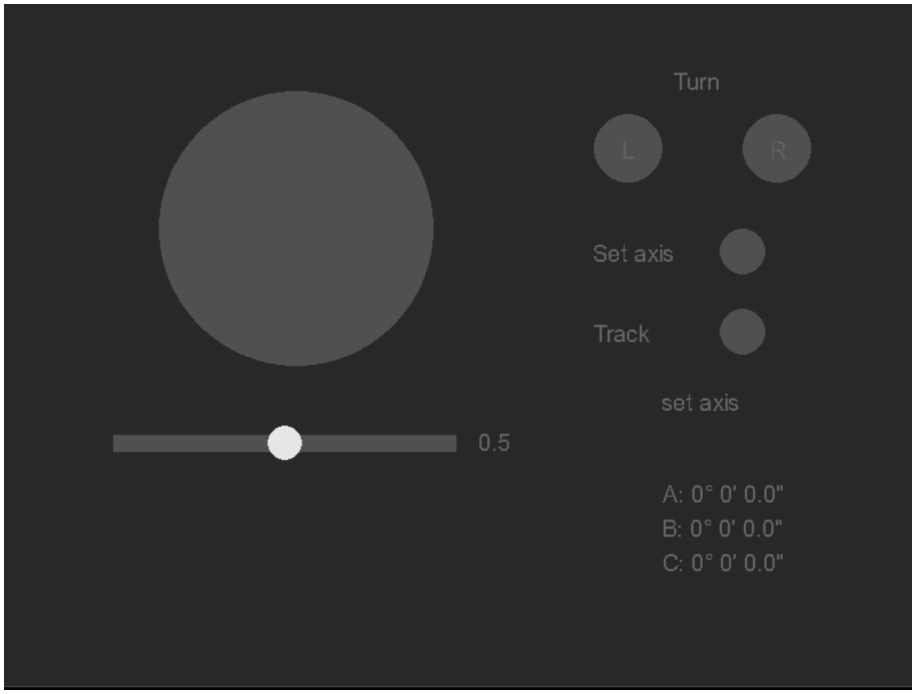

Figure 4.13: User interface of the controller desktop application

4.5.4 Calibration

After the setup calibration has to be performed, there are two ways to do this. The first one requires pointing the StarTrckr on the North Star. Unfortunately, it is not always visible and might be hard to do. An easier method has been found during testing. With a level application on a smartphone tracker can be set perfectly to the north as shown in figure 4.14

Figure 4.14: Calibration procedure with a smartphone

Then it is possible to once again use the application and set the altitude of the tracker to the geographical latitude of the place of observation. In the author's case, that was 50.2 deg (place: Bytom, Poland). Latitude can be obtained online, from a map, or once again with a compass app. This was found to be the easiest, most reliable, and precise method of calibration. The altitude of the tracker can be observed in the desktop controller.

Once the tracker is aligned with the Earth's axis of rotation "set axis" button in the app can be clicked. The text box below it shall change to "Axis set".

4.5.5 Observation

Before starting, the observation tracker has to be moved to a desired location. That can be done with a joystick and buttons available in the UI of the controller application or with keyboard arrow keys. With the use of astronomy software like Stellarium apparent position of the object can be obtained, and camera pointed to this position manually with the mentioned interface. After clicking the track button tracker will start tracking the object. It is advised to use a remote shutter release or an intervalometer to avoid touching the camera during observation.

4.6 Outcomes

Specific results that can be obtained will vary based on small details, camera, lens, and filters used but also polar alignment and the environment. Taking excellent astrophotography images requires practice, time, willingness to learn, and, most importantly, patience. Example images were captured in the south of Poland at the end of December 2022 with Canon 600D and 18-55mm, f3.5-5.6 kit lens with no additional filters.

In figure 4.15 we can see a perfect example with no artifacts regarding to tracking. The red dots visible in the image are actually dead pixels on the camera's sensor. This image is one of many frames captured for the stacking test. The results of stacking are presented in figure 4.17. This was the author's first experience with stacking images for astrophotography and it is worth mentioning that the results are very poor and could be drastically improved by conducting more experiments.

Figure 4.15: 120s exposure captured at ISO200 after editing

Figure 4.16 is a great example of bad polar alignment, for that reason stars appear as weird shapes in the image. It was captured at a higher ISO and in an area that was less light-polluted. Unprocessed, raw images from the camera are presented in chapter 6.

Figure 4.16: 240s exposure captured at ISO800 after editing

Figure 4.17: First results of stacking 22 images together

Chapter 5

Internal specification

5.1 Desktop software

Software specification is divided into a desktop application which is the controller used on a computer to position the tracker and perform all the necessary calculations, and the firmware used on the RP2040 to communicate with the tracker and control the motors.

5.1.1 Architecture

The class diagram of desktop software is presented in figure 5.1. Custom classes were developed to create the user interface as well as control the tracker, perform mathematical operations and keep the code clean. PyGame, a library used to create games in Python was used to create the user interface. While it may not seem like a good choice, it was easy to work with and allowed the author to experiment with writing custom classes and more advanced code. Using PyQt could be a more straightforward approach. PySerial was used for communication with the StarTrckr through a serial port. And lastly, NumPy was used as a library to perform all the operations on matrices and vectors. Controller.py is the main program that uses other classes to execute the commands. It is responsible for displaying the user interface, connecting and sending the data to the serial port, and reading the input from the user. gui.py holds class definitions of user interface elements and based on these classes, objects are created in the controller.py. TrackerController.py stores the information on the tracker's position and all the vectors necessary for rotations. There are also various functions to rotate the tracker around different axes and read the position of the tracker as well as separate vectors for displaying with the PlotVisualizer.py. TrackerMath.py stores just two functions with rotation matrices that allow to rotate the object around the origin axes as well as around an arbitrary axis defined by a unit vector. An additional class was developed for debugging purposes - PlotVisualizer it was of great use while working on the algorithm to visualize all the vectors and how those behave with different rotations. It is not used in the final program but might be useful to modify or

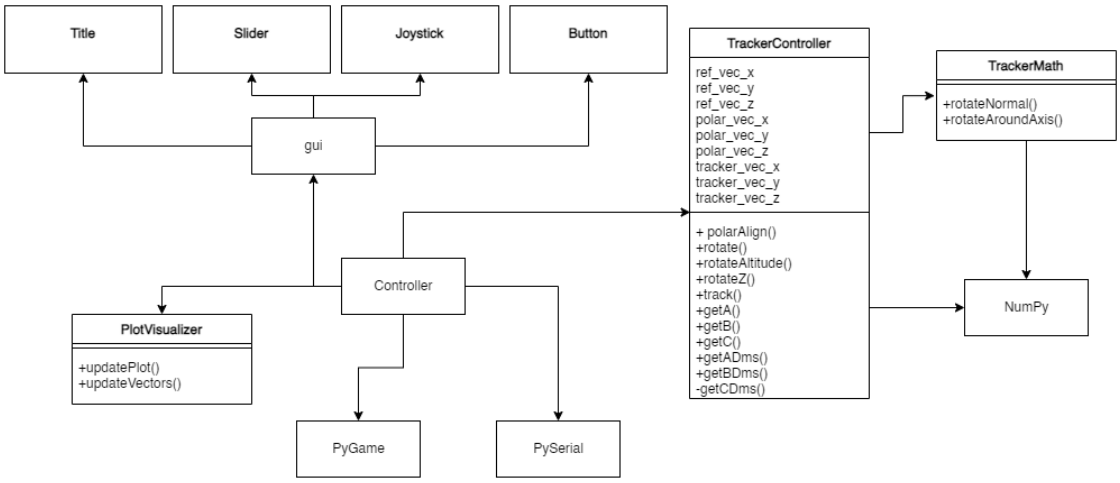

Figure 5.1: UML diagram of desktop software

extend in the future.

5.1.2 Communication

Data from the controller software is sent to the tracker through a serial port as a string similar to gcode used in CNC machines. So far, only basic movement commands are used in the following form:

```
A10.2 B53.1 C30.2
```

The command is interpreted by the firmware explained in section 5.2. More advanced commands can be developed in the future and implemented both in the firmware and the desktop application. This simple communication system proved to work well during tests and was enough to perform observations.

5.1.3 Matrix rotations

The most interesting part of the desktop application are the functions used for rotating vectors with rotation matrices [20]. The function presented in figure 5.2 in the end was not used as every rotation is performed around an axis with rotateAroundAxis function presented in 5.3. The first step in each function is to convert degrees to radians and perform the rotation by calculating the rotation matrix and multiplying it with the vector. Both functions return a new rotated vector which can be used in the TrackerController to calculate position as well as for future rotations. The tracker controller stores multiple vectors representing different reference frames and is required to perform rotations prop-

```
def rotateNormal(vector, angle_x, angle_y, angle_z):
    a = np.pi/180 * angle_x
    b = np.pi/180 * angle_y
    c = np.pi/180 * angle_z
    rm = np.array([[np.cos(b)*np.cos(c), np.sin(a)*np.sin(b)*np.cos(c)-np.cos(a)*np.sin(c),
                    np.cos(a)*np.sin(b)*np.cos(c)+np.sin(a)*np.sin(c)],
                   [np.cos(b)*np.sin(c), np.sin(a)*np.sin(b)*np.sin(c)+np.cos(a)*np.cos(c),
                    np.cos(a)*np.sin(b)*np.sin(c)-np.sin(a)*np.cos(c)],
                   [-np.sin(b), np.sin(a)*np.cos(b), np.cos(a)*np.cos(b)]])
    return np.dot(rm, vector)
```

Figure 5.2: rotateNormal method from TrackerMath

```
def rotateAroundAxis(vector, axis, angle):
    #vector: [ux, uy, uz]
    t = np.pi/180 * angle
    rm = np.array([[np.cos(t) +pow(axis[2], 2)*(1-np.cos(t)), axis[0]*axis[1]*(1-np.cos(t)) -
                   axis[2]*np.sin(t), axis[0]*axis[2]*(1-np.cos(t)) + axis[1]*np.sin(t)],
                   [axis[1]*axis[0]*(1-np.cos(t)) + axis[2]*np.sin(t), np.cos(t)+pow(axis[1], 2)
                   *(1-np.cos(t)), axis[1]*axis[2]*(1-np.cos(t))-axis[0]*np.sin(t)],
                   [axis[2]*axis[0]*(1-np.cos(t))-axis[1]*np.sin(t), axis[2]*axis[1]
                   *(1-np.cos(t))+axis[0]*np.sin(t), np.cos(t)+pow(axis[2], 2)*(1-np.cos(t))]])
    return np.matmul(rm, vector)
```

Figure 5.3: rotateAroundAxis method from TrackerMath

erly around all axes. In total, there are four different reference frames defined - origin, reference, polar, and tracker. Each frame is defined with three vectors. The tracker's position is defined by the last one, polar is used for tracking to rotate around the celestial pole, and reference is primarily used for alt-az rotations.

5.2 Firmware

RP2040 firmware was developed in MicroPython, the class diagram is presented in figure 5.4. The main program is responsible for Reading the data from the serial port and extracting the information required to move the tacker. Once the position for each axis is obtained it is sent to the Tracker object with moveTo method. Movement is performed using Bresenham's algorithm [3] originally developed for two-dimensional line drawing with a plotter here was adopted to move in 3 dimensions.

5.2.1 Tracker class

The tracker object also stores the stepper motor, LEDs, and buzzer objects. This class is responsible for operating the tracker. Class StepperMotor stores pin definitions that are connected to the stepper motor driver and functions to enable, and disable the driver

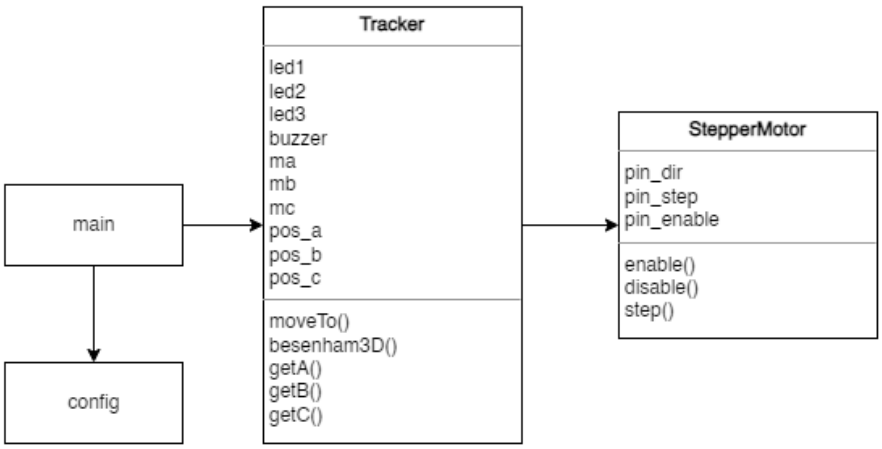

Figure 5.4: UML diagram of firmware for RP2040

and perform step in a specified direction. Additionally, there is a config file where we can configure all the settings like pin numbers for each element, reverse step direction or modify steps per mm setting.

5.2.2 Bresenham's algorithm

Bresenham's algorithm, first introduced in 1965, is still greatly used today in computer graphics, CNC machines, plotters, or 3D printers. Implementation of the algorithm for the StarTrckr project can be found in figure 7.2.8. When the function is called, the axis with the greatest movement is detected, and based on that, we move by one step on each iteration while keeping track of the error on the other two axes. When the error is bigger than one step, we perform a step on the axis and subtract one from the error. With such a simple algorithm, it is possible to complete the movement and calculations very efficiently.

5.2.3 Command interpreter

Serial commands are interpreted with readCommand function 5.5. Firstly one line from the serial port is read. There is an if instruction checking if the line is not empty, and if so, we can start extracting the data. With `split(' ')` commands A, B and C are separated, and now the program has to deal with three separate commands instead of one command from the serial port. The command is transformed into a list from which the program can easily read all the values. At the end when all the values are ready, those are sent to the tracker object with .moveTo() function. The command has to be organised exactly as specified, and all three subcommands (A, B, C) must be included.

```
def readCommand():
  global pos_a, pos_b, pos_c
  cm_list = sys.stdin.readline().strip().split(' ')
  print(cm_list)
  if cm_list[0] != '':
      targets = [tracker.getA(), tracker.getB(), tracker.getC()]
      if len(cm_list) >= 1:
          if cm_list[0][0] == 'A':
              print('Moving A axis by: ', float(cm_list[0][1:]), ' deg\n')
              targets[0] = float(cm_list[0][1:]) * A_STEPS_PER_DEGREE

      if len(cm_list) >= 2:
          if cm_list[1][0] == 'B':
              print('Moving B axis by: ', float(cm_list[1][1:]), ' deg\n')
              targets[1] = float(cm_list[1][1:]) * B_STEPS_PER_DEGREE
      if len(cm_list) >= 3:
          if cm_list[2][0] == 'C':
              print('Moving C axis by: ', float(cm_list[2][1:]), ' deg\n')
              targets[2] = float(cm_list[2][1:]) * C_STEPS_PER_DEGREE

      tracker.moveTo(targets[0], targets[1], targets[2])
```

Figure 5.5: Simple serial command interpreter

5.3 Electronics

Electronics was developed without a preceding prototype. For that reason PCB used in the project was the first prototype and while no errors have been detected in the design some improvements for the design are included in the conclusions. Schematic and PCB were designed in KiCad and the source files can be accessed in the GitHub repository of the project together with zipped Gerber files required for production.

5.3.1 Schematic diagram

The schematic diagram presented in figure 5.6 was divided into subsystems, and those were connected with each other with labels to keep the schematic easy to read. Microcontroller, memory, and filtration were connected according to the hardware design recommendations prepared by Raspberry Pi [9]. The logic part is powered via USB through a 3.3V linear voltage regulator, while the stepper motors are powered externally through a screw terminal connector. Additional LEDs were used in the schematic. Those were used during the testing phase of the PCB but have no actual use at the final stage of the project. Similarly, a buzzer is installed but is of no special use in the project. Those are potential devices to be used in future versions. SD card slot was never soldered and tested, and in future versions, shall be removed from the PCB. The microcontroller uses an external 12MHz crystal, and an external quad SPI flash memory that is required as RP2040 has no built-in memory and otherwise would not be able to store the program.

The memory chosen for the project is W25Q128JVS, as recommended in the hardware design manual [9]. Most likely, a smaller memory size can be utilized in the project at this stage but considering future development and expandability, a bigger one was used.

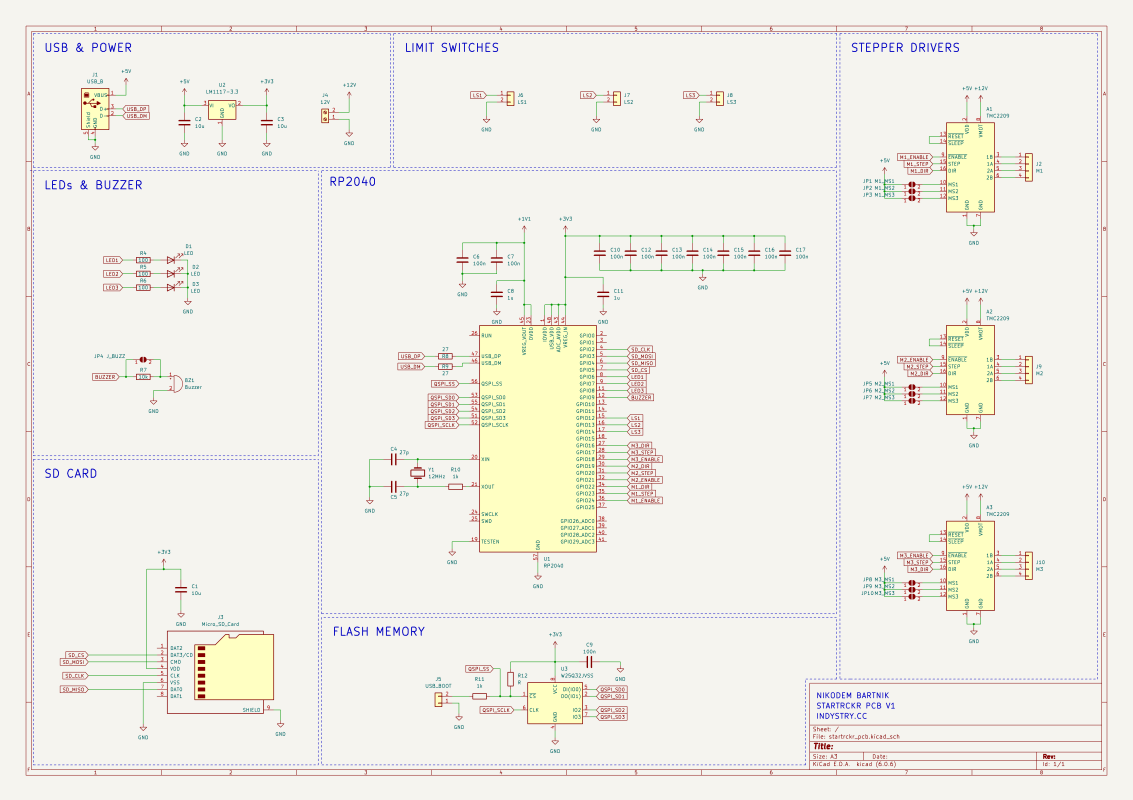

Figure 5.6: Schematic diagram

5.3.2 PCB layout

The layout of the PCB (top and bottom layer) is presented in figure 5.7. The width of the traces has been optimized where necessary, and a copper pad was used to distribute 12V to stepper motor drivers as high currents may flow through this trace. All capacitors and resistors are in a 0603 package that ensures a small footprint leading to a smaller PCB and easy hand soldering. PCB was designed according to the capabilities of the JLCPCB manufacturer. If a different manufacturing company were to be used, it must be verified if the used standards are alike.

5.3.3 Stepper motor drivers

At the moment of designing the PCB, it was not yet secured what drivers will be used in the final project for that reason, to ensure flexibility, it was decided that female header pins would be used and a stepper driver in the form of a separate module will be plugged into the headers and that way it will connect to the PCB. This is a similar approach used

in the early days of 3D printing, especially the RepRap project. Stepper drivers often referred to as step sticks in this form, are standardized and multiple different versions can be purchased at a low price. Additionally, there are special jumper pads on the bottom of each stepper driver to configure microstepping by soldering the pads together. In the end, A4988 drivers were used in the project and configured to 1/16 of the step so MS1, MS2, and MS3 jumper pads had to be soldered. If A4988 were to be replaced by TMC2209 microstepping can go as low as 1/256 of a full step with additional configuration or 1/64 of the step with MS1 and MS2 configuration pins. If those changes were to be made, also config.py and especially the steps/deg setting would have to be adjusted. While StarTrckr could benefit from the changes, it would increase the price of the tracker significantly, and one of the objectives was to keep the price low.

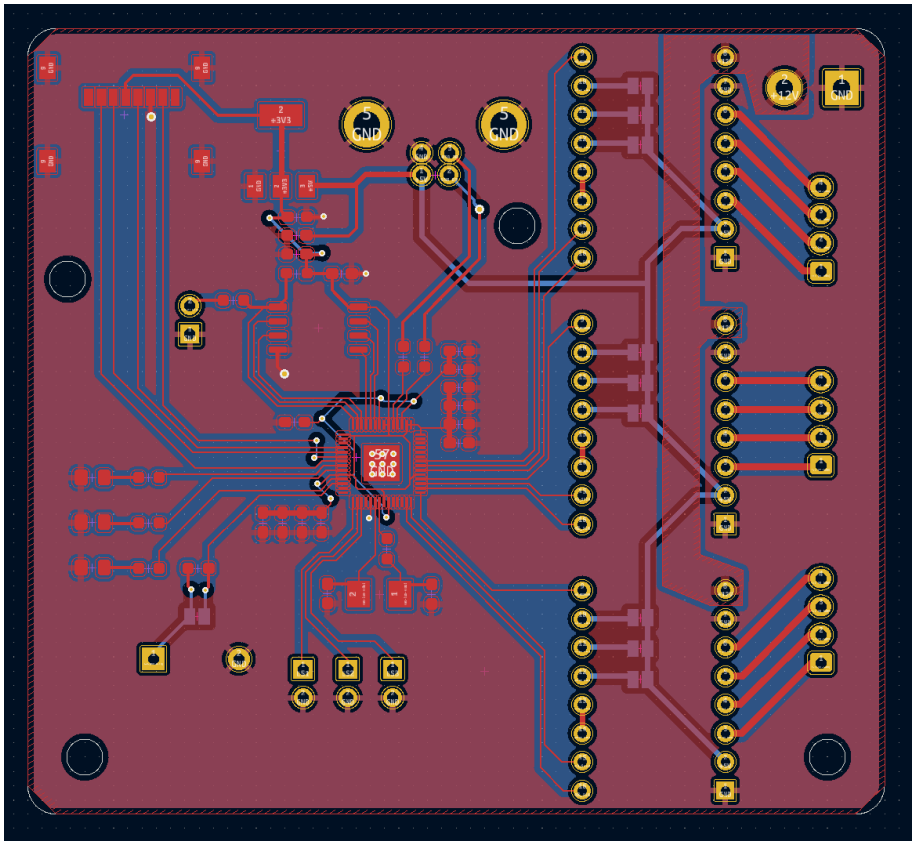

Figure 5.7: PCB layout

Chapter 6

Verification and validation

6.1 Software testing

No automated tests were performed in the project. All software was tested manually. An incremental testing approach was used, and each functionality was tested just after being added to the project. Many tests required printing the state of variables with the print command. As previously mentioned, an additional class for displaying interactive vectors was developed to quickly solve bugs and test the rotation and tracking algorithms. Firmware for the RP2040 microcontroller was developed in a similar manner but required less testing as its complexity is lower. Thanks to the fact that MicroPython was used to develop the firmware and is quite similar to standard Python, for example, Bresenham's algorithm was developed and tested on the computer, and once its correctness was proven, it was implemented on the microcontroller.

6.2 Electronics and hardware tests

Testing of the electronics was applied already during soldering. Once the most crucial components were in place, the circuit was tested with a laboratory power supply with the current limit to ensure that in case of a short, nothing will burn on the PCB. Even before powering anything, a visual inspection was performed with a microscope. When the PCB was completely soldered, and after installing the Python interpreter, all the signals were tested with a simple test script that turned on all LEDs, buzzer, and pins connected to the stepper drivers that were not installed. To check if those pins work, an oscilloscope was used.

During Silesian Science Festival, where the project was presented first more extended power-on test was performed. During the test it was found that stepper motors overheat after about 20-30 minutes. That was caused by the wrong current setting on the stepper motor drivers. After adjusting, the effect was greatly reduced, motors are still warm but

at an acceptable level.
Project's mechanical part, especially 3D printed parts, were tested and replaced frequently. The process was very iterative and dynamic, thanks to 3D printing technology. One of the first versions of the tracker, referred to in work as v2, was very flexible. When the camera was mounted the displacement of the camera holder was greater than 8mm, which was not acceptable. Improved v3 of the design reduced the displacement to about 2-4mm. There is still room for improvement, but the stiffness of the construction is limited by the technology used in the project.
One of the parts (arm2.3mf) was forcing the camera to be mounted in a way that made it impossible to use the camera's screen. It was a design mistake that originated from the first designs and different approaches used. Mirroring this part and printing again gave access to the camera screen and also an HDMI port to which an external monitor can be connected.
During an unintentional drop test of the construction, it was found that everything was intact except for pins that lock the axis so new pins were printed at higher infill.

Primary reason for the design v3, which is similar to v2, was the poor design of some parts that were not reinforced at the most crucial spots. Holes for the motors significantly reduced the stiffness of the construction as there was very little material connecting the part. Parts were extended to the sides to improve the stiffness; additional reinforcing features were added, and some parts were printed with more perimeters at higher infill. Small 3D printed gear responsible for rotating the A axis placed in the base and attached to the stepper motor with a press fit was getting lose quickly. Two locking screws hidden between the teeth of the gear were added, and the gear was printed with PETG instead of PLA. The problem was never faced again.

6.3 First tracking tests

The lack of appropriate weather forced the author to be very creative. Time constrained required to perform some tests as early as possible, and the weather was not going to help. That is how the idea for an indoor test was developed. Using Stellarium software image of the sky was displayed on the computer's monitor, the tracker was aligned with the North Star, and the tracking test was started 6.1. During the test, a common error in computer science was found a wrong sign error that made the tracker rotate in the opposite direction. After fixing the problem and performing more similar tests indoors, results were promising and allowed for the first field tests a few days later.

Figure 6.1: Indoor test of the StarTrckr with Stellarium software

6.4 Field tests

The most important part of the testing and the whole project was the field tests and trying to capture the first images of the night sky. Those tests did not focus on any specific part of the sky, constellations, or deep sky objects. It was just about checking whether the tracker can be used for astrophotography, how well the tracking works if it is easy to use in the field and how it performs during longer use. Unfortunately, the weather conditions were very unfortunate for a long period of time. In the end, only two observation sessions were performed with the v2 StarTrckr, and there was no opportunity to field test the upgraded v3 tracker. Additionally, due to the project's short timeline, there was no time to plan a trip to an area with minimal light pollution. This would have greatly improved the final images and highlighted the contrast between tracked and untracked shots.

First tests were performed on 27.12.2022 when the sky was clear. Some clouds interrupted the observation session, but it was possible to obtain the first pictures and analyze the results. Although the tracker was not precisely aligned with the celestial pole, there is a noticeable difference between untracked 6.2 and tracked 6.3 images. It is worth noting that the pictures were captured in a light-polluted area, and clouds started entering the frame during the untracked image capturing. A portion of a building can be seen in the bottom right corner of the image. While it is sharp on the untracked image since the camera was stationary, the tracked image presents a blurry building because the camera was moving while tracking the sky. Both images were captured using the same camera, lens and settings looking at the same part of the sky for the untracked test the camera was also mounted on the tracker, but tracking was disabled. The first test was performed in a small village at the south of Poland with bigger cities nearby.

Figure 6.2: Stable camera mounted on the StarTrckr with tracking disabled f/5.6, ISO800, 240s, 55mm, Canon 600D

Figure 6.3: First tracked picture captured by the StarTrckr f/5.6, ISO800, 240s, 55mm, Canon 600D

On 28.12.2022, another cloudless night occurred, so another observation and testing session was performed at a different location near the city of Bytom. Light pollution was a more significant problem this time, but a rather dark spot in the forest helped in reducing its influence. This time more care was put into calibrating the tracker and leveling the table. After capturing a, few rather promising images, a longer 600 second exposure was taken twice with tracking turned on and off. While the focus was not set perfectly and the stars are still blurred, and some artifacts can be observed, there is a significant improvement in the trailing of the stars.

Figure 6.4: untracked, f/5.6, ISO200, 600s, 55mm, Canon 600D

Figure 6.5: tracked, f/5.6, ISO200, 600s, 55mm, Canon 600D

600 second exposure made the final image too bright. The cause is light pollution. The objective of this test was to push exposure time as high as possible and compare the results. In this sense, it was successful, but with this configuration taking exposures that long is not precise enough and results in slightly blurred stars. The test contributed to improvements and changes in the next iterations of the design that, unfortunately, due to weather conditions, have not been tested at this point.

Figure 6.6: Test setup used during field testing

During the second testing session, additional 22 exposures each about 90 seconds long were captured of the same part of the sky with the intention of trying to stack the images together and form a single more detailed, and noise-free image. The result 6.7 is far from the author's expectations but considering the low number of frames captured and the early prototype of the project used it is quite promising and more experiments need to be conducted in order to improve feature stacking experiments.

Figure 6.7: tracked, f/5.6, ISO200, 22 frames each about 90s long, 55mm, Canon 600D stacked with Sequator software

6.5 Requirements verification

Verification of the requirements described in chapter 3.

6.5.1 Functional requirements

1. General

   (a) Project shall track the deep sky objects with enough precision to take a 1 minute exposure with a 100mm lens without noticeable star trailing - was not tested because of the weather. Considering it was possible to take a 180s and 240s exposure with no signs of trailing as well as 600s exposure with minimal trailing with a 55mm lens it shall be possible to capture 60s exposure at 100mm

2. Mechanical

   (a) Shall be possible to point the camera to any object in the night sky - **requirement met completely.**

   (b) Shall allow for easy alignment with the celestial pole using the north star or compass application on the smartphone - **requirement met completely.**

   (c) Shall be possible to use the StarTrckr with any popular DSLR or mirrorless camera **requirement met completely.** Tested with Canon 600D and Lumix G7, that way most of Canon's EOS DSLR and any mirrorless camera shall fit on the tracker. It was also tested with various lenses from small wide-angle

lenses to big and heavy telephoto lens like Tamron 70-300mm. Using lenses of even bigger size might be a problem but the project was not designed for this kind of lenses from the beginning.

(d) 3D printed parts shall fit on a build plate that is 20x20x20cm - **requirement met completely.** Parts for v2, v3 and v4 can be printed with Ender3, Prusa i3 or any similar printer.

(e) All screws shall be metric - **requirement met completely**

3. Electronic

(a) Shall be powered with 12V power adapter or an external 3S LiPo battery - **requirement met completely.** A 12V gel battery or car lighter socket can be used as well.

(b) All electronic components shall fit on a single side of a two-layer PCB that fits inside the project's printed case - **requirement met completely.**

(c) Pancake stepper motors shall be used with 200 steps/rev - **requirement met completely.**

(d) Main controller shall communicate with the computer via USB B cable - **requirement met completely.**

4. Software

(a) Shall be easy to operate for the end user with a computer and Windows/macOS/Linux - **requirement met completely.**

(b) Control software shall communicate with the project through serial connection - **requirement met completely.**

6.5.2 Nonfunctional requirements

1. Shall be easy to build for everyone using a 3D printer and some easy-to-buy components - **requirement met completely.** Requirement is hard to verify and some questions would have to be asked to the users to confirm it. But in fact, the project can be easily printed on a 3D printer with minimal support needed as is designed with 3D printing in mind. All parts are widely available and in some cases, replacements can be used without modifying the design.

2. Complete easy to transport solution - **requirement met completely.**

3. Setup shall take no more than 5 minutes - **requirement met completely.** Verified during field testing, which includes preparing the table, computer, and polar alignment process.

4. Everything that is not required for the user to see shall be hidden - **requirement met partially.** The user does not see the PCB, the majority of the cables, or the mathematical equations that turn the tracker and track the sky (at the using stage, he does during building the tracker on his own). Unfortunately, because it was not anticipated during the PCB design phase, the DC connector used to connect the power source to the motors is exposed outside of the casing.

5. Nice to have feature: control with Stellarium - feature not implemented because of the time constraint

6. Nice to have feature: tracking satellites - feature not implemented because of the time constraint

Chapter 7

Conclusions

7.1 Results

The requirements specified in chapter 3 mainly were fully or partially satisfied as concluded in 6. The development of this project allowed the author to dive deeply into the world of astrophotography and face all the problems he was unaware of at the beginning of this journey. It was proven that with an idea and determination, inexpensive solutions can be created to look deeply into the night sky. Solving the problems was not only a fun challenge but also a valuable lesson. Taking a peek at the night sky with tracked long-exposure images encouraged the author to continue working on the project. Hopefully, it will be a fully developed and mature star-tracking solution at some point in the future.

7.2 Future development and flaws of the design

Numerous alternate approaches and more effective answers to various issues have been discovered during the project's development. It was impossible to incorporate them into the project due to time constraints, but development will continue, and maybe the ideas listed below will be adopted in the future.

7.2.1 Metal parts

Printed parts are convenient for the project because they are easy to produce for anyone with access to a printer, and those are pretty popular nowadays. But materials used for FDM printing obviously have limitations that can not be overcome. Replacing some of the parts with laser-cut steel would remarkably influence the stiffness of the construction and, at a scale, even reduce costs and production time. A fully 3D printed version shall still be developed and available for the users as the most approachable and open-source, but an additional improved version can be created.

7.2.2 Integration with Stellarium

Stellarium is a very popular open-source astronomy software that many astronomy and astrophotography enthusiasts are familiar with. While being easy to use, it offers many essential tools and allows for controlling telescopes and trackers with standardized interfaces. Integrating the control of the tracker with Stellarium would simplify taking pictures of deep sky objects and the overall user experience.

7.2.3 Smaller PCB

PCB can be reduced in size also some components can be removed. If it were to be smaller and the shape was changed, one of the base modules designed to hold the PCB would be unnecessary. PCB could fit together with the motor holder in a single printed part, reducing the printing time and filament use. Also, smaller PCB reduces the price at a scale. As previously mentioned SD card slot can be removed entirely as it is not necessary for the project, but some additional, rather small elements could be added. Keeping all SMD components on one side is recommended to simplify the PCB assembly process.

7.2.4 Stepper drivers integrated on the PCB

More testing would have to be done to confirm if the tracker has enough accuracy with A4988 stepper drivers, together with additional tests using TMC2209 for comparison. Once an appropriate driver is chosen, it can be integrated into the PCB to reduce size and height of the PCB as well as heat dissipation from the drivers. Alternatively, two versions of the PCB could be designed with different drivers so that the end user can choose whether he prefers a cheaper or more expensive solution.

7.2.5 Accelerometer and compass on the PCB

Assuming the latitude of the observation specified by the user in the control software, StarTrckr might align itself with the celestial pole by integrating the accelerometer and compass. A GPS module might be used to automate the procedure and eliminate the user's involvement. To correct for any flaws in the mechanical design and leveling of the surface that the tracker is standing on, an accelerometer might be placed just below the camera. As a result, more sophisticated hardware and software would be needed to complete the task. Still, for the user, a fully automatic procedure would eliminate any errors and the amount of time required for the process.

7.2.6 DC jack on the PCB

DC jack shall be added next to the USB socket instead of the screw terminal that was used on the prototype. It was a temporary solution, and the DC jack would look more professional and reduce the need for additional external socket and cables.

7.2.7 Less 3D printed parts

Smaller PCB would allow to reduce the number of components required for 3D printing. Also, replacing some with steel parts could help reduce the printed parts used in the project. Considering the idea of scaling the project and making it available as a product to purchase, it is an important factor to reduce production costs.

7.2.8 More testing

Weather and time constraints made it impossible to perform more field tests. Validation and improvement in future versions of the tracker would require more testing, preferably with more users involved, to find all the flaws of the design. Taking more night sky pictures could help verify different celestial pole alignment methods. Using different cameras, lenses and taking pictures of the most popular deep sky objects would allow for comparing the images with commercial solutions.

Bibliography

[1] Neil Adake. 'Trust the Process: An Investigation into Astrophotography'. In: (2022). URL: https://commons.erau.edu/aiaar2sc/2021/freshman-sophomore-open-topic/4/.

[2] arpruss. *Simple Manual Star Tracker for Astrophotography*. 2020. URL: https://www.instructables.com/Simple-Manual-Star-Tracker-for-Astrophotography/ (visited on 25/12/2022).

[3] J. E. Bresenham. 'Algorithm for computer control of a digital plotter'. In: *IBM Systems Journal* 4.1 (1965), pp. 25–30. DOI: 10.1147/sj.41.0025.

[4] Andrew Davidhazy. 'How to make a "leaf" or "book" style star tracker mount'. In: (Sept. 2003). URL: https://scholarworks.rit.edu/article/238.

[5] Niels Haagh. *Telescopemount.org*. 2019. URL: https://telescopemount.org/alt-az-mounts-for-long-exposure-astrophotography-camera-rotators/ (visited on 23/12/2022).

[6] Robert Irion. 'A Sprawling Andromeda Galaxy Startles and Puzzles Observers'. In: *Science* 157.5729 (2005), pp. 1733–1733. DOI: 10.1126/science.308.5729.1733a. eprint: https://www.science.org/doi/pdf/10.1126/science.308.5729.1733a. URL: https://www.science.org/doi/abs/10.1126/science.308.5729.1733a.

[7] Robert Lambourne. 'The Doppler effect in astronomy'. In: *Physics Education* 32.1 (Jan. 1997), p. 34. DOI: 10.1088/0031-9120/32/1/017. URL: https://dx.doi.org/10.1088/0031-9120/32/1/017.

[8] Christopher Ryan McBryde and E. Glenn Lightsey. 'A star tracker design for CubeSats'. In: (2012), pp. 1–14. DOI: 10.1109/AERO.2012.6187242.

[9] Raspberry Pi. *Hardware design with RP2040*. URL: https://datasheets.raspberrypi.com/rp2040/hardware-design-with-rp2040.pdf (visited on 04/01/2023).

[10] Raspberry Pi. *Raspberry Pi Pico Python SDK*. URL: https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-python-sdk.pdf (visited on 03/01/2023).

[11] Kurt W. Riegel. 'Light Pollution'. In: *Science* 179.4080 (1973), pp. 1285–1291. DOI: 10.1126/science.179.4080.1285. eprint: https://www.science.org/doi/pdf/10.1126/science.179.4080.1285. URL: https://www.science.org/doi/abs/10.1126/science.179.4080.1285.

[12] Benjamin R. Seratt. *Integrated Design and the Backyard Astronomer*. 2018. URL: https://scholarworks.umt.edu/umcur/2018/amposters/17/.

[13] JAMES YUAN SING WONG. 'Star Tracker: Your Gateway to Astronomy'. In: (2022). URL: https://kastner.ucsd.edu/ryan/wp-content/uploads/sites/5/2022/06/admin/star-tracker.pdf (visited on 25/12/2022).

[14] Sky-Watcher. *Sky-Watcher EQ3 mount*. URL: https://www.skywatcher.com/product/eq3-mount-with-steel-tripod/ (visited on 27/12/2022).

[15] Sky-Watcher. *Sky-Watcher Star Adventurer*. URL: https://www.skywatcher.com/product/star-adventurer/ (visited on 27/12/2022).

[16] Jurij Stare. *lightpollutionmap.info*. URL: https://www.lightpollutionmap.info/ (visited on 23/12/2022).

[17] Stellarium. *Stellarium*. URL: https://stellarium.org/ (visited on 30/12/2022).

[18] Fabian Uehleke. *OpenAstroTech Github*. 2021. URL: https://github.com/OpenAstroTech/OpenAstroTracker (visited on 23/12/2022).

[19] Will Varley. *Weddings and Wars*. 2014.

[20] Wikipedia. *Rotation matrix*. URL: https://en.wikipedia.org/wiki/Rotation_matrix (visited on 04/01/2023).

[21] Joy Gu Yuyi Shen Kenny Ramos. 'Asterism: Motorized Astrophotography Mount'. In: (2020). URL: http://course.ece.cmu.edu/~ece500/projects/s20-teamb1/wp-content/uploads/sites/83/2020/05/Team_B1.pdf (visited on 25/12/2022).

List of Figures

List of Tables

Appendices

Index of abbreviations and symbols

LEO	Low Earth Orbit
MEO	Middle Earth Orbit
PCB	Printed Circuit Board
DIY	Do It Yourself
CAD	Computer Aided Design
IDE	Integrated Development Environment
SBC	Single Board Computer
CNC	Computer Numerical Control

Listings

```
    def rotateAltitude(self, a):
        self.tracker_vec_x = tm.rotateAroundAxis(self.tracker_vec_x,
        ↪  self.tracker_vec_y, a)

    def rotateAzimuth(self, a):
        self.tracker_vec_x = tm.rotateAroundAxis(self.tracker_vec_x,
        ↪  self.origin_vec_z, a)
        self.tracker_vec_y = tm.rotateAroundAxis(self.tracker_vec_y,
        ↪  self.origin_vec_z, a)
        self.tracker_vec_z = tm.rotateAroundAxis(self.tracker_vec_z,
        ↪  self.origin_vec_z, a)
        self.ref_vec_x = tm.rotateAroundAxis(self.ref_vec_x, self.origin_vec_z, a)
        self.ref_vec_y = tm.rotateAroundAxis(self.ref_vec_y, self.origin_vec_z, a)
        self.ref_vec_z = tm.rotateAroundAxis(self.ref_vec_z, self.origin_vec_z, a)

    def rotateField(self, x):
        self.tracker_vec_y = tm.rotateAroundAxis(self.tracker_vec_y,
        ↪  self.tracker_vec_x, x)
        self.tracker_vec_z = tm.rotateAroundAxis(self.tracker_vec_z,
        ↪  self.tracker_vec_x, x)
```

Figure 1: Functions used for manual rotations

```
    def track(self, angle):
        self.tracker_vec_x = tm.rotateAroundAxis(self.tracker_vec_x, self.polar_vec_x,
        ↪  angle)
        self.tracker_vec_y = tm.rotateAroundAxis(self.tracker_vec_y, self.polar_vec_x,
        ↪  angle)
        self.tracker_vec_z = tm.rotateAroundAxis(self.tracker_vec_z, self.polar_vec_x,
        ↪  angle)
```

Figure 2: Function used for tracking

```
    def getADms(self):
        d, m, s = self.__decimalToDms(self.getA())
        return f"{d}° {m}' {s}\""

    def getBDms(self):
        d, m, s = self.__decimalToDms(self.getB())
        return f"{d}° {m}' {s}\""

    def getCDms(self):
        d, m, s = self.__decimalToDms(self.getC())
        return f"{d}° {m}' {s}\""

    def __decimalToDms(self, decimal):
        degrees = int(decimal)
        minutes = int((decimal - degrees) * 60)
        seconds = round((((decimal - degrees) * 60) - minutes) * 60, 2)
        return (degrees, minutes, seconds)
```

Figure 3: Functions converting degrees to degrees minutes and seconds

```
def bresenham3D(self, dest_a, dest_b, dest_c):
        da, db, dc = abs(dest_a - self.pos_a), abs(dest_b - self.pos_b), abs(dest_c -
        ↪  self.pos_c)
        dir_a = 1 if dest_a > self.pos_a else -1
        dir_b = 1 if dest_b > self.pos_b else -1
        dir_c = 1 if dest_c > self.pos_c else -1
        temp_a, temp_b, temp_c = 0, 0, 0
        if not (da == 0 and db == 0 and dc == 0):
            if da >= db and da >= dc:
                step_b = db / da
                step_c = dc / da
                for x in range(int(da)):
                    self.pos_a = self.pos_a + dir_a
                    self.ma.step(dir_a)
                    temp_b = temp_b + step_b
                    temp_c = temp_c + step_c
                    if temp_b >= 1:
                        temp_b = temp_b - 1
                        self.pos_b = self.pos_b + dir_b
                        self.mb.step(dir_b)
                    if temp_c >= 1:
                        temp_c = temp_c - 1
                        self.pos_c = self.pos_c + dir_c
                        self.mc.step(dir_c)
                    time.sleep(DELAY_BETWEEN_STEPS)
            elif db >= da and db >= dc:
                step_a = da / db
                step_c = dc / db
                for x in range(int(db)):
                    self.pos_b = self.pos_b + dir_b
                    self.mb.step(dir_b)
                    temp_a = temp_a + step_a
                    temp_c = temp_c + step_c
                    if temp_a >= 1:
                        temp_a = temp_a - 1
                        self.pos_a = self.pos_a + dir_a
                        self.ma.step(dir_a)
                    if temp_c >= 1:
                        temp_c = temp_c - 1
                        self.pos_c = self.pos_c + dir_c
                        self.mc.step(dir_c)
                    time.sleep(DELAY_BETWEEN_STEPS)
            elif dc >= da and dc >= db:
                step_a = da / dc
                step_b = db / dc
                for x in range(int(dc)):
                    self.pos_c = self.pos_c + dir_c
                    self.mc.step(dir_c)
                    temp_a = temp_a + step_a
                    temp_b = temp_b + step_b
                    if temp_a >= 1:
                        temp_a = temp_a - 1
                        self.pos_a = self.pos_a + dir_a
                        self.ma.step(dir_a)
                    if temp_b >= 1:
                        temp_b = temp_b - 1
                        self.pos_b = self.pos_b + dir_b
                        self.mb.step(dir_b)
                    time.sleep(DELAY_BETWEEN_STEPS)
```


Figure 4: Three-dimensional Bresenham's algorithm

List of additional files in electronic submission

Additional files uploaded to the system include:

- source code of the desktop application,
- source code of the firmware,
- 3MF files of the V3 design,
- KiCad PCB design files
- test images,
- video showcasing the project, alternatively the video can be accessed here in higher quality: https://youtu.be/M618ToFtBnA