Space Challenges

Summer 2025

EnduroSat

A Design Rewiev on CASSI

Celestial Alignment System for Satellite Inertial-control

Group members:
Michael Yan, Alexandra Nedela, Ivaylo Tsekov, Tsvetomir Staykov, Dilyana Vasileva

Mentor:

Simeon Baltadzhiev

Submitted: 3. August 2025

Contents

List of Tables

List of Figures

1 Abstract

This document presents the design and validation of CASSI (Celestial Alignment System for Satellite Inertial-control), a modular star tracker prototype optimized for CubeSat-class platforms. The system fuses optical and inertial data to enable real-time, drift-corrected attitude determination. Core components include a Sony IMX500 global-shutter CMOS sensor for celestial imaging, a BNO055 9-DOF IMU for inertial measurements, and a Raspberry Pi 4 Model B serving as the onboard processing unit. Orientation is output in quaternion form and transmitted via UART to an Arduino-based subsystem emulator. The star identification program implements image preprocessing, star centroiding, and geometric feature extraction. Star-pattern search is performed through angular pattern matching using a reduced Hipparcos star catalogue, stored in an optimized SQLite database with precomputed inter-star angular distances. Attitude estimation solves Wahba's problem using both the QUEST algorithm and Davenport's Q method, producing rotation quaternions with degree-level precision. Sensor fusion compensates for the long-term drift of the IMU by periodic realignment with optical data. Mechanical constraints are addressed via a PLA-printed prototype structure with detailed integration tolerances. Materials for a flight-grade implementation, such as CFRP for the housing and anodized aluminum for the optical baffle, were selected on the basis of thermal, structural, and optical performance metrics. Experimental validation was conducted using both synthetic (Stellarium-based) and real star field images. Results confirm the system's ability to autonomously determine attitude under simulated space conditions. CASSI demonstrates the feasibility of achieving arcminute-level orientation accuracy using low-cost, commercial-grade components, offering a scalable path toward space-qualified ADCS modules, accessable to small and medium space companies.

2 Introduction

The exploration and utilization of space have become increasingly sophisticated and require precise control and understanding of satellite behavior. At the heart of a satellite's operational capability lies its ability to accurately determine and maintain its orientation in the vacuum of space. This "attitude determination" is not merely a technical detail; it is a fundamental requirement for everything from maintaining communication links to precisely pointing scientific instruments, enabling Earth observation, or performing orbital maneuvers. Without accurate knowledge of its orientation, a satellite is essentially blind and incapacitated, unable to perform its designated mission. This document outlines the development of an innovative system that uses the stars to determine its orientation. Such systems are called 'star trackers'. They are designed to address the critical challenge of determining the attitude of the satellite.

2.1 Project Overview and Mission Statement

The mission entails developing an algorithm that identifies star patterns from images captured by a camera, as well as positional star data from sky catalogues to determine the satellite's orientation in space. Integration of the camera data with the IMU sensor data is used to track real-time movements, and both data sources are to be combined using sensor fusion. This will allow continuous and accurate orientation updates, even when star visibility is limited or temporarily lost. This project is conceived as a proof-of-concept and a foundational step towards developing a highly reliable and adaptable satellite attitude determination system. The proposed system will leverage consumer-grade hardware such as RaspberryPi 4 [16] and RaspberryPi AI camera [15], making it both accessible for educational purposes and a viable prototype for future flight-ready systems. At its core, the project focuses on two primary sensing modalities: an optical star tracker (camera) and an Inertial Measurement Unit (IMU). The optical component utilizes a high-resolution camera, specifically the RaspberryPi AI camera, to capture images of the celestial sphere. These images will then be processed by a pattern recognition algorithm designed to detect and identify known star constellations or individual bright stars. By accurately pinpointing the positions of these celestial landmarks within the camera's field of view, the system can precisely infer the camera's, and thus the satellite's, orientation relative to the inertial frame of reference (the stars) when compared to the star catalogue. Complementing the optical system is an Inertial Measurement Unit. For this project, the Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout, BNO055 [6], will be employed. This IMU provides real-time data on angular velocity, linear acceleration, and magnetic field strength. While IMUs offer high update rates and are excellent for tracking

short-term movements, they suffer from inherent drift over time, meaning their accuracy degrades progressively without external corrections. This project uses the sensor fusion mechanism. This technique will dynamically combine the high-precision, drift-free but intermittently available data from the star tracker with the high-frequency, continuous but drift-prone data from the IMU. This approach aims to overcome the individual limitations of each sensor. For instance, during periods when star visibility might be poor (e.g., due to atmospheric conditions in lower orbits, sensor occultation, or high angular velocities causing motion blur), the IMU can provide continuous orientation updates. Conversely, when star data is available, it will be used to correct the accumulated drift of the IMU, thereby providing a consistently accurate estimate of the satellite's attitude. The entire processing and control logic will be orchestrated by a Raspberry Pi 4, serving as the central computing unit, providing ample processing power for image analysis and sensor data integration. An Arduino [3] board will be further attached to simulate real satellite dynamic thruster control.

2.2 Motivation for Accurate Satellite Orientation

The demand for accurate satellite orientation, also known as attitude determination and control (ADCS), is paramount across nearly all space missions. The precise knowledge and control of a satellite's orientation in three-dimensional space directly impacts its ability to fulfill its mission objectives, ensure operational longevity, and guarantee safety. There are many motivations underlining the critical importance of accurate satellite orientation. Star trackers face significant challenges in space due to solar flares, moonlight, and stray light from the Sun and Earth. Solar flares cause false star detections, increased noise, and permanent sensor damage from high-energy particles. Moonlight and Sun/Earth stray light lead to sensor saturation, glare, and scattered light, making it difficult to detect stars and degrading attitude determination accuracy. Combined with the critical nature of satellite missions, it necessitates highly robust and reliable attitude determination systems. The integration of multiple sensing modalities, as proposed in this project, offers a path toward achieving the required levels of accuracy and resilience.

2.3 Project Goals and Objectives

Building upon the mission statement, this project is structured around a set of clear goals and specific objectives designed to guide the development process and measure its success.

1. To develop a prototype system capable of continuous and accurate satellite orientation determination in a simulated space environment.

2. To demonstrate the efficacy of sensor fusion in combining optical star tracker data and IMU data to overcome individual sensor limitations.

3. To establish a foundation for further development of flight-ready attitude determination systems integrable as CubeSat Modules.

3 Research

In developing our star tracker system, we began with a focused analysis of the most widely used technologies in the space industry. We examined both hardware, such as leading commercial star trackers, and the software algorithms they rely on for accurate attitude estimation. Analysis was made of their key challenges, such as sensor noise, motion blur, and stray light. These findings directly shaped our design decisions, from sensor selection to algorithm integration.

3.1 Study of Star Trackers

Sodern HYDRA Star Tracker [18]

The Sodern HYDRA Star Tracker is a high-performance, modular star tracking system designed for use in demanding space environments. It features up to three optical heads linked to a centralized processing unit, allowing for redundancy and enhanced robustness. Each optical head is equipped with a radiation-hardened CMOS sensor, a precision lens assembly, and a baffle that suppresses stray light, enabling sub-arcsecond attitude determination accuracy even under harsh conditions like radiation and mechanical vibrations. The hardware is constructed using low-expansion composite materials to ensure thermal stability, while space-qualified optical glass is used in the lenses to maintain optical precision. The housing and baffle are typically made from lightweight, high-strength carbon or aluminum alloys, and internal surfaces are treated with specialized coatings to reduce reflections and maximize image clarity. The onboard processing software carries out real-time image acquisition, star pattern recognition, and attitude computation using the Quaternion Estimator (QUEST) algorithm, with update rates of up to 30 Hz. Additional features include gyro-aided tracking, redundant star matching, and error correction algorithms for improved reliability. The HYDRA offers a pointing accuracy better than 1 arcsecond (typically around 0.5 arcsec), with each optical head weighing approximately 1.5 kg and the entire system (including processing unit) around 4.5–5 kg. The typical dimensions per optical head are roughly $10 \times 10 \times 15$ cm. Power consumption is about 10–15 W depending on configuration

Figure 3.1: Sodern HYDRA Star Tracker

Blue Canyon Technologies Nano Star Tracker (NST) [11]

The Blue Canyon Technologies Nano Star Tracker (NST) is a compact, lightweight star tracker specifically designed for CubeSats and small satellite platforms, offering a balance between performance, size, and power efficiency. It incorporates a miniaturized CMOS sensor, a compact optical lens assembly, and an onboard processor capable of executing key star-tracking functions. With a wide field of view of approximately 20–25 degrees, the NST is optimized for broad sky coverage, making it well-suited for missions where moderate pointing accuracy is acceptable in exchange for wider visibility. The housing is made of lightweight aluminum alloy, while all internal components are constructed from low-outgassing materials to comply with small satellite cleanliness standards. Its lenses are crafted from optical glass coated with anti-reflective materials to enhance light transmission and reduce internal scattering. The NST's onboard software performs star centroiding, pattern recognition, and attitude estimation using modified versions of lightweight algorithms such as QUEST, tailored for low processing overhead. It outputs attitude quaternions along with health and status indicators, and integrates easily with miniature Attitude Determination and Control Systems (ADCS). In terms of performance, the NST provides pointing accuracy typically in the range of 20–50 arcseconds depending on configuration and mission conditions. Its mass ranges from approximately 150 to 350 grams, with dimensions around $5 \times 5 \times 10$ cm, allowing seamless integration into tight payload spaces. Power consumption is very low, typically under 2 W, making it ideal for energy-constrained platforms.

Figure 3.2: Blue Canyon Technologies Nano Star Tracker

Leonardo's A-STR and AA-STR Star Trackers [12]

Leonardo's A-STR and AA-STR star trackers are high-reliability, compact attitude determination systems designed for a wide range of satellite missions, from small satellites to agile, high-dynamic spacecraft. Both models incorporate a high-resolution CMOS sensor and radiation-hardened electronics capable of withstanding harsh space environments. The A-STR is tailored for standard LEO and GEO missions with compact form factors, while the AA-STR is engineered to support rapid slew rates, making it suitable for highly agile platforms that require precise tracking during dynamic maneuvers. Housed in robust, space-qualified enclosures made from aerospace-grade aluminum, these trackers offer excellent mechanical integrity and thermal stability. The optical assemblies use radiation-tolerant glass with anti-reflective coatings to ensure high transmission efficiency and minimize stray light. Internally, space-grade printed circuit boards and components are shielded with materials such as tantalum or tungsten to enhance fault resilience in high-radiation environments. Onboard processing includes real-time star identification, centroiding, and attitude estimation algorithms, with the AA-STR featuring advanced motion compensation to maintain accuracy during slews. Fault-tolerant software further ensures continuous operation by mitigating radiation-induced upsets and errors. Both trackers deliver high pointing accuracy, typically better than 1 arcsecond (AA-STR achieving even higher precision under motion). The A-STR weighs approximately 1.5 to 2.0 kg, while the more advanced AA-STR ranges from 2.0 to 2.5 kg. Their dimensions are compact, usually within $10 \times 10 \times 15$ cm envelopes. Power consumption is around 5–8 W depending on model and operational mode.

Figure 3.3: Leonardo's A-STR Star Tracker

3.2 Star Tracking Algorithms

The most fundamental part of completing a star tracking algorithms comes from determining the rotational matrix when trying to match vectors of image stars to catalogue stars. This is known as Wahba's problem. First posed by Grace Wahba in 1965 []. This section will explore some of the most optimised and sophisticated algoriths used in the inudstry.

The QUEST [2](QUaternion ESTimator) algorithm is a widely used method for determining a spacecraft's attitude from a set of known reference vectors $\{\mathbf{r}_i\}$ in the inertial frame and their measured counterparts $\{\mathbf{b}_i\}$ in the body frame. It solves Wahba's problem by minimizing the cost function:

$$J(\mathbf{q}) = \frac{1}{2}\sum_{i=1}^{n} a_i \left\|\mathbf{b}_i - \mathbf{R}(\mathbf{q})\mathbf{r}_i\right\|^2, \tag{3.1}$$

where a_i are positive weights, $\mathbf{R}(\mathbf{q})$ is the rotation matrix parameterized by the unit quaternion $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$.

QUEST formulates this into an eigenvalue problem by constructing the Davenport K matrix:

$$K = \begin{bmatrix} S - \sigma I & \mathbf{z} \\ \mathbf{z}^T & \sigma \end{bmatrix}, \tag{3.2}$$

where

$$B = \sum_{i=1}^{n} a_i \mathbf{b}_i \mathbf{r}_i^T, \quad S = B + B^T, \quad \sigma = \text{trace}(B), \quad \mathbf{z} = \begin{bmatrix} B_{23} - B_{32} \\ B_{31} - B_{13} \\ B_{12} - B_{21} \end{bmatrix}. \tag{3.3}$$

The optimal quaternion $\mathbf{q}^*$ corresponds to the eigenvector associated with the maximum eigenvalue $\lambda_{\max}$ of K. QUEST is computationally efficient and numerically stable, making it suitable for real-time onboard attitude determination in small satellites.

Davenport's Q-Method[1]

Davenport's Q-Method is a classical approach to solving Wahba's problem by reformulating it as a quadratic optimization over quaternions. Given the weighted attitude profile matrix

$$B = \sum_{i=1}^{n} a_i \mathbf{b}_i \mathbf{r}_i^T, \tag{3.4}$$

the method constructs the symmetric 4×4 Davenport K matrix:

$$K = \begin{bmatrix} S - \sigma I & \mathbf{z} \\ \mathbf{z}^T & \sigma \end{bmatrix}, \tag{3.5}$$

with the same definitions for S, σ, and $\mathbf{z}$ as above.

The optimal quaternion $\mathbf{q}^*$ maximizes the quadratic form

$$\mathbf{q}^T K \mathbf{q}$$

subject to the unit norm constraint $\|\mathbf{q}\| = 1$. This reduces to solving the eigenvalue problem:

$$K\mathbf{q} = \lambda \mathbf{q}. \tag{3.6}$$

The quaternion $\mathbf{q}^*$ corresponding to the largest eigenvalue $\lambda_{\max}$ provides the best-fit attitude. Although computationally more intensive than QUEST, Q-Method serves as the fundamental mathematical basis and is preferred for batch or ground-based high-precision computations.

OpenCV in Star Tracking[4]

OpenCV (Open Source Computer Vision Library) provides robust tools for image processing in star trackers. The raw star field image $I(x, y)$ undergoes pre-processing steps such as thresholding:

$$I_T(x, y) = \begin{cases} 1, & \text{if } I(x, y) > T, \\ 0, & \text{otherwise} \end{cases}, \tag{3.7}$$

where T is a chosen intensity threshold.

Blob detection algorithms then identify connected regions of pixels representing star images. The centroids (x_c, y_c) of these blobs are computed as:

$$x_c = \frac{\sum_{x,y} x \cdot I(x, y)}{\sum_{x,y} I(x, y)}, \tag{3.8}$$

$$y_c = \frac{\sum_{x,y} y \cdot I(x, y)}{\sum_{x,y} I(x, y)}. \tag{3.9}$$

OpenCV's modularity and support for Python and C++ make it ideal for rapid prototyping of star detection and centroiding pipelines, crucial in CubeSat projects where flexibility and cost-efficiency are priorities.

Centroiding and Pattern Recognition[19]

Centroiding refines star positions by calculating the intensity-weighted centers of star blobs to improve measurement precision:

$$\mathbf{c}_i = (x_{c,i}, y_{c,i}) = \left(\frac{\sum_{x,y} x I_i(x,y)}{\sum_{x,y} I_i(x,y)}, \frac{\sum_{x,y} y I_i(x,y)}{\sum_{x,y} I_i(x,y)} \right), \tag{3.10}$$

where I_i is the intensity within the i-th star's blob.

Pattern recognition algorithms then attempt to match these centroids $\{\mathbf{c}_i\}$ to catalogued star patterns by comparing geometric invariants such as inter-star angular distances:

$$d_{ij} = \cos^{-1}\left(\hat{\mathbf{s}}_i \cdot \hat{\mathbf{s}}_j\right), \tag{3.11}$$

where $\hat{\mathbf{s}}_i, \hat{\mathbf{s}}_j$ are unit vectors corresponding to star directions from the catalog.

Fast, robust pattern matching enables the star tracker to quickly identify its orientation even under noisy or partially obscured conditions. Together, centroiding and pattern recognition form the core of the star identification step before attitude estimation.

3.3 Data Collection

Star trackers use star libraries, also called star catalogs, which are databases containing precise information about stars, including their celestial coordinates, brightness (magnitude), and sometimes color. These catalogs provide the reference data needed for the tracker to identify stars captured in its images and determine the spacecraft's orientation. Popular star catalogs include Hipparcos[9], Tycho-2[14], and Gaia[8], which offer millions to billions of stars with very accurate positions. The onboard software uses these catalogs to match observed star patterns with known star arrangements, enabling fast and reliable attitude determination. For CubeSat[7] or educational star trackers, smaller, optimized subsets of these catalogs are often used to reduce memory and processing requirements while maintaining accuracy.

The Hipparcos catalog is one of the most important and widely used star libraries in astronomy and star tracking. Created from data collected by the European Space Agency's Hipparcos satellite[10] launched in 1989, it contains high-precision measurements of about 118,000 stars, including their positions, parallaxes, and proper motions. This catalog provides extremely accurate celestial coordinates, which are essential for reliable star identification and attitude determination in star trackers. Because of its relatively manageable size and precision, Hipparcos is often used as a primary reference catalog in spacecraft navigation systems and CubeSat star trackers where memory and processing power are limited, making it a practical choice for onboard star pattern matching.

Collecting accurate data for star trackers poses several challenges. One major issue is light pollution and stray light, reflections from the Earth, Moon, or Sun can flood the image sensor and obscure star patterns, especially in low Earth orbit. Radiation in space can degrade sensor performance or generate false signals ("hot pixels"), which can confuse pattern recognition algorithms. Additionally, motion blur during rapid satellite maneuvers can lead to inaccurate centroiding of stars. Temperature variations and lens contamination from outgassing materials can also affect calibration and sensor alignment. All these factors require advanced filtering, robust algorithms, and rigorous pre-launch calibration to ensure reliable star tracking under all orbital conditions.

4 Design

4.1 System Design

The system takes star images from a camera and IMU data (gyroscope, accelerometer, and magnetometer) as input, and outputs the determined attitude in the form of quaternions on a user interface (UI). Simultaneously, it communicates via I^2C with an Arduino, which manages the remaining satellite subsystems.

4.1.1 System Requirements

S.R.1: Determine the orientation of the system based on captured star images.

S.R.2: Track orientation using IMU without continuous visual input once calibrated.

S.R.3: Automatically recalibrate when drifting occurs due to environmental variables, power loss, or other factors.(this can be done by periodic calibration)

S.R.4: Display orientation and camera image on UI.

S.R.5: Communicate with rest of system using Arduino (tested using LED input).

4.1.2 Working Principle of CASSI

1. Image Capture: A camera, in our case the Raspberry Pi AI camera, captures a wide-field live image of the sky.

2. Database Filtering: The star catalog is filtered to include only stars with an absolute magnitude brighter than 4. Each star's right ascension (RA) and declination (DEC) are converted into unit vectors $\hat{\mathbf{s}}_i$ in the inertial frame:

$$\hat{\mathbf{s}}_i = \begin{bmatrix} \cos(\mathrm{DEC}_i)\cos(\mathrm{RA}_i) \\ \cos(\mathrm{DEC}_i)\sin(\mathrm{RA}_i) \\ \sin(\mathrm{DEC}_i) \end{bmatrix}. \tag{4.1}$$

A table of possible angles θ_{ij} between every pair of stars is computed using:

$$\theta_{ij} = \cos^{-1}(\hat{\mathbf{s}}_i \cdot \hat{\mathbf{s}}_j). \tag{4.2}$$

Pairs with horizontal angular separations greater than 66° are discarded since they cannot appear together in the camera's field of view.

3. Star Detection: Image processing algorithms detect bright points of light (candidate stars) within the captured frame, using thresholding and filtering to exclude cosmic rays, hot pixels, and noise.

4. Image Data Centroiding and Angle Calculation: For each detected star, a centroiding algorithm calculates precise pixel coordinates (x, y). Using the camera's focal length f, pixel coordinates are converted into unit vectors in the camera frame $\hat{\mathbf{b}}_i$:

$$\hat{\mathbf{b}}_i = \frac{1}{\sqrt{x_i^2 + y_i^2 + f^2}} \begin{bmatrix} x_i \\ y_i \\ f \end{bmatrix}. \tag{4.3}$$

Angles between pairs of these unit vectors are then computed:

$$\phi_{ij} = \cos^{-1}(\hat{\mathbf{b}}_i \cdot \hat{\mathbf{b}}_j). \tag{4.4}$$

This forms an observed angle table for pattern matching.

5. Pattern Matching: The observed angle table $\{\phi_{ij}\}$ is searched against the database angle table $\{\theta_{ij}\}$ to find corresponding star pairs. This matching determines the orientation of the camera in terms of pitch and yaw.

6. Attitude Calculation: Once stars are identified, the known inertial frame vectors $\hat{\mathbf{s}}_i$ and measured body frame vectors $\hat{\mathbf{b}}_i$ form the input to an attitude determination algorithm (e.g., QUEST or Davenport Q-Method). This yields the spacecraft's rotation matrix $\mathbf{R}$ or quaternion $\mathbf{q}$. The roll angle can be computed by solving for the rotation matrix that aligns the sets of vectors, completing the full attitude estimation.

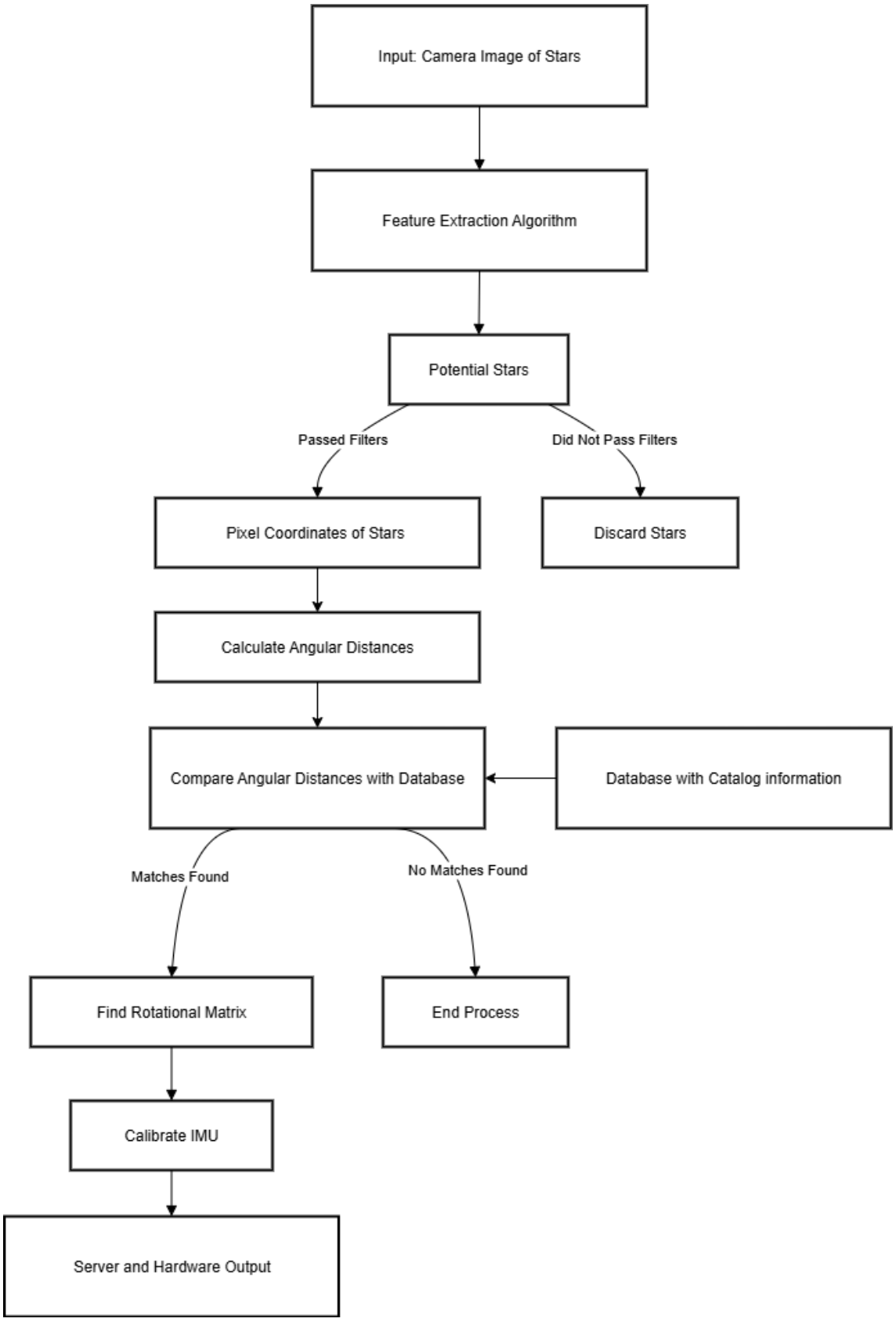

Figure 4.1: System Design

4.2 Hardware

Raspberry Pi 4 is used as the central processing unit, BNO055 IMU for orientation sensing, and the Raspberry Pi AI Camera (based on the Sony IMX500 sensor) for star imaging. While this setup is more modest than industry-grade systems like the Sodern HYDRA[18], it mirrors their key architectural concepts.

The Raspberry Pi 4 serves as an accessible and flexible onboard computer, similar to the centralized processing units in professional systems. The camera, though not radiation-hardened, uses a CMOS sensor like those found in the HYDRA[18] and Leonardo star trackers[12], enabling decent low-light performance needed for capturing stars with the addition of a buffer.

The BNO055 IMU adds real-time attitude data, akin to how advanced star trackers combine optical data with inertial measurements for greater accuracy. Although not space-qualified, this setup allows for prototyping a functional and educational star tracker that follows principles used in aerospace applications. See Figure 7.2 in Appendix for complete schematic.

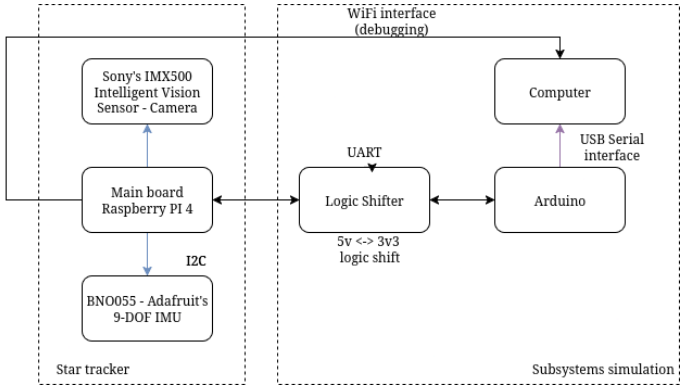

Figure 4.2: Hardware Interfacing

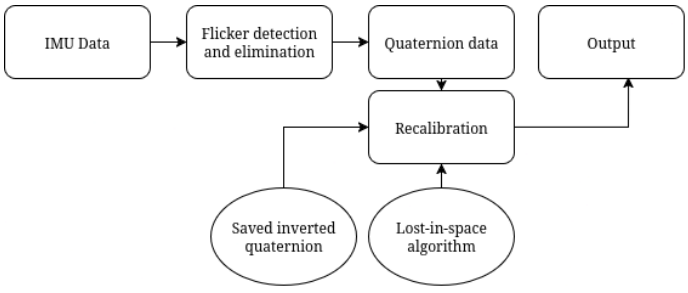

Figure 4.3: System Logic Overview

4.2.1 Hardware Requirements

H.R.1: At least 1GB of RAM and sufficient flash storage for star catalogue.

H.R.2: Field of View (FOV) of at least 50 degrees.

H.R.3: Image sensor capable of detecting magnitude 4 to 5 stars.

H.R.4: Interface with the rest of the satellite.

H.R.5: Keep track of orientation without image sensor.

4.2.2 Hardware Choices

The Raspberry Pi 4 Model B was chosen as the central computing unit. It has 4GB of VI RAM with connective possible over WiFi. Quad-core ARM Cortex-A72 CPU provides significant processing power for image processing (star detection, centroiding), complex sensor fusion algorithms, and overall system management. It has a 40-pin GPIO header and MIPI CSI camera interface, enabling direct connection to the Sony IMX500, crucial for this project.The Linux OS provides a robust, multi-tasking environment with extensive libraries and development tools (e.g., Python, C++). Its low cost in conjunction with these technical abilities enable it to be perfect for facilitating rapid prototyping.

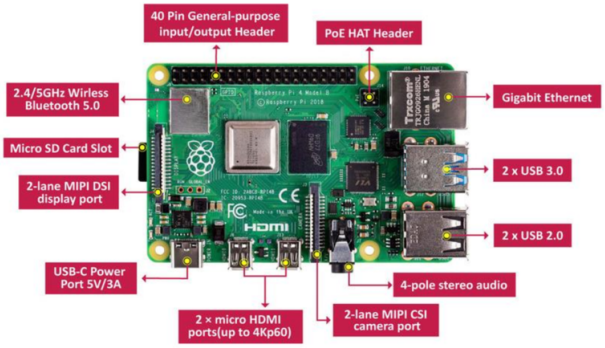

Figure 4.4: Raspberry Pi 4 Model B Board

The 12MP Sony IMX500 Intelligent Vision Sensor is chosen as the core of the optical star tracker. (4056 x 3040 pixels) Field of View (FoV): Approximately 76 degrees (diagonal). Frame Rate: Capable of high frame rates at lower resolutions, which can be leveraged for faster image processing. Selection Rationale: High Resolution (12MP): Provides a large number of pixels for precise star centroiding and allows for a wide Field of View (FOV) while maintaining angular resolution. Global Shutter: Crucial for star tracking in dynamic environments. A global shutter captures the entire frame simultaneously, preventing distortion (e.g., "smear") of star images during satellite rotation, which would occur with a rolling shutter. Raspberry Pi Compatibility: Its MIPI CSI interface is directly compatible with the Raspberry Pi camera port, simplifying integration.

Figure 4.5: 12MP Sony IMX500 Intelligent Vision Sensor i

The Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055 is selected for inertial measurements. Containing a 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer, it provides industry standard attitude determination at a minimal cost. At the same time, the BNO055 provides fused absolute orientation data directly, significantly simplifying the data processing load on the main Raspberry Pi 4. While we will perform our own fusion with the star tracker, the BNO055's internal fusion provides a very clean and stable base for angular rates and acceleration. Its small size and breakout board format also make it easy to integrate into a prototype system. With an I2C Interface, it allows easy interfacing with the Raspberry Pi.

Figure 4.6: Adafruit 9-DOF Absolute Orientation IMU F

4.2.3 Software Interfaces

The firmware serves as the core logic that operates directly on the hardware, enabling seamless integration between the software stack and the physical system. Our setup runs on a 32-bit Debian Linux distribution in headless mode, optimized for embedded deployment. Upon boot, the main application initializes by loading all necessary libraries and subsystems, including modules for camera and IMU configuration, sensor fusion algorithms, data processing, subsystem communication, lost-in-space detection, and internal database management.

To optimize performance and minimize resource usage, the operating system has been stripped of non-essential services and drivers such as Bluetooth, Ethernet, and unused kernel modules. Only critical components remain. Currently, Wi-Fi is the sole method of accessing the user interface and retrieving real-time system information.

The primary communication and control protocols in use are:

1. SSH; for secure remote access and debugging
2. WebSockets (bidirectional);for real-time data exchange
3. Flask Web Server; to transfer the camera live feed into the web-based user interface

Interface between the RaspberryPi and the Arduino nano will be a one-way data pipeline for transmitting real-time orientation data in quaternions. The Raspberry Pi acts as the data source and master controller, while the Arduino serves as the data sink, processing the orientation data. Communication is established over a UART serial link, enhanced by a custom binary protocol to ensure reliability and data integrity.

On the Raspberry Pi, the primary UART (/dev/serial0, mapped to GPIO 14 (TX) and GPIO 15 (RX)) is utilized. To achieve maximum performance and stability, the PL011 UART is assigned to the GPIO pins by adding the dtoverlay=miniuart-bt directive to the /boot/config.txt file, which relegates the lower-performance mini UART to the onboard Bluetooth module. A critical consideration in the hardware interface is the logic level mismatch between the 3.3V GPIO of the Raspberry Pi and the 5V I/O of the Arduino Nano. To ensure system safety and prevent damage to the Raspberry Pi, a bidirectional logic level shifter is implemented on the communication line. The connection is wired in a crossover configuration: the Raspberry Pi's TX pin is connected directly to the Arduino's RX pin, and the Arduino's TX pin is connected through the level shifter to the Raspberry Pi's RX pin, effectively dropping the voltage to 3.3V . A common ground (GND) connection between all devices is essential and has been established to provide a stable voltage reference.

A custom packet-based binary protocol was designed. Each data packet is a fixed size of 18 bytes, structured to encapsulate a single quaternion. The packet frame consists of a Start Byte, a 16-byte data payload, and a single-byte checksum. The Start Byte is a constant value (0xAA) that serves as a unique marker, allowing the Arduino to synchronize with the data stream at any point, even if it starts up mid-transmission. The 16-byte payload contains the four 32-bit floating-point numbers (w, x, y, z) of the quaternion. To validate

the integrity of the payload, a simple 8-bit checksum is calculated by summing the values of all 16 payload bytes. This checksum is appended to the packet, enabling the Arduino to verify that the received data has not been corrupted during transit. Any packet that fails the checksum validation is discarded, preventing the system from acting on erroneous data.

The system's logic is implemented through two distinct software components. On the Raspberry Pi, a Python 3 script orchestrates the data transmission. This script utilizes the pyserial library to manage the serial port and the struct library to efficiently pack the floating-point quaternion data into its binary byte representation. For development, the script is executed within a Python virtual environment to ensure dependency isolation and system stability.

On the receiving end, the Arduino Nano is programmed with a C++ sketch that implements the packet-parsing logic. The sketch continuously monitors the hardware serial buffer for incoming bytes. It employs a state machine to first detect the 0xAA Start Byte, then read the 16-byte payload and the final checksum byte. A key feature of the Arduino implementation is the use of a union data structure. This allows the 16-byte payload to be read into a byte array and then be instantly reinterpreted as a structure of four floats without any processing overhead or memory copy operations. Upon successful checksum validation, the quaternion data is made available for use within the Arduino's main application loop.

4.2.4 User interface

To simplify the configuration and interaction with the module. It is designed for system control and data analysis. This UI makes it easier to visualize information, monitor the camera feed, and test specific features such as force calibration and mode switching.

The current version includes six panels:

1. 3D Visualization of quaternion data (powered by Three.js)
2. Graph Display of quaternion data (using Chart.js)
3. Terminal Output(showing received data)
4. Live Camera View
5. Quaternion and System Status (overview)
6. Control Panel (with interactive buttons)

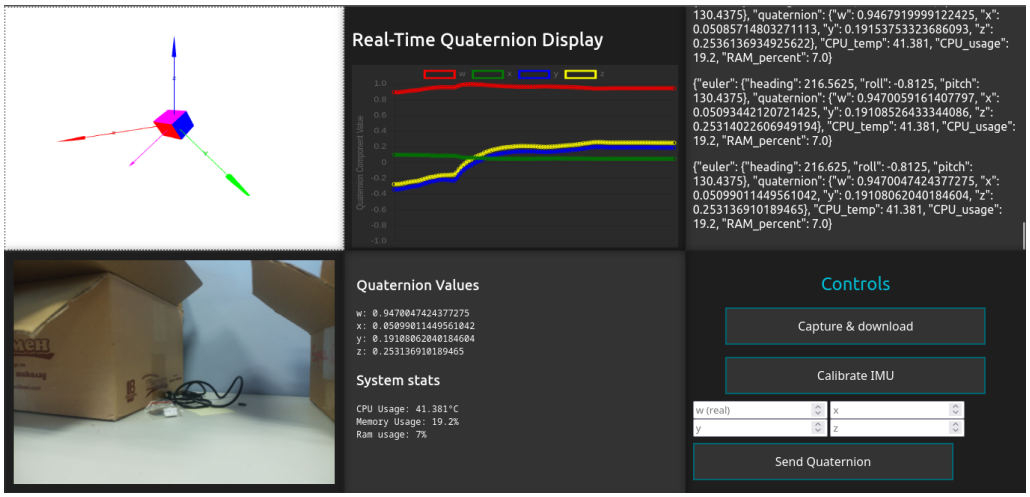

Figure 4.7: User Interface

4.3 Software

The system's software is based on a deterministic Lost-in-Space algorithm for star pattern recognition, which works in conjunction with IMU data processing and a sensor fusion approach for continuous attitude determination and recalibration. The core components include a structured database, image processing and feature extraction, angular distance calculations, star ID search in the catalogue, and finally attitude determination.

4.3.1 Software Requirements

S.R.1: Catalogue should contain data of celestial coordinates of each star.

S.R.2: Entire attitude determination should be completed within one second.

S.R.3: Program should have an attitude determination error of less than 1 degree.

S.R.4: Software should be able to be stored and run on the Raspberry Pi 4.

4.3.2 Database and Star Catalog

A specialized script processes the Hipparcos star catalog to generate a streamlined database optimized for star tracker systems. The core function is to filter the extensive Hipparcos data for the brightest stars and then pre-calculate the angular distances between every possible pair within this filtered set. This pre-computation is the key to offloading intensive calculations from the real-time tracking system, which enables swift star pattern identification. The result is a single SQLite database file containing two main tables: one for star data and another for the pre-computed angular separations. To achieve this, the script relies on Python 3 and several key libraries: Pandas for data manipulation, NumPy for numerical calculations, tqdm for monitoring progress, and the standard sqlite3 library for database interaction.The process of generating this database involves two primary stages. First, the script loads the Hipparcos catalog from a CSV file and filters out stars

Column	Data Type	Description
hip_id	INTEGER	Primary key. The unique Hipparcos catalog identifier for the star
vmag	REAL	The star's apparent visual magnitude. A lower value indicates a brighter star.
x	REAL	The X-component of the star's unit vector in the ICRS/equatorial system
y	REAL	The Y-component of the star's unit vector in the ICRS/equatorial system
z	REAL	The Z-component of the star's unit vector in the ICRS/equatorial system

Table 4.1: Star Data Table

based on a configurable brightness threshold (apparent magnitude). It then converts the celestial coordinates (Right Ascension and Declination) into a 3D Cartesian unit vector for each star. This refined star data includes the Hipparcos ID, magnitude, and unit vector components. It is then inserted into a newly created "Stars" table within the SQLite database.

Following the creation of the star table, the second stage focuses on calculating the angular distances. For every pair of stars, it computes the angular distance using the dot product of their unit vectors and the arcosine function. These calculated distances are then inserted into a second table named "Angular Distances", which links pairs of stars by their Hipparcos IDs to their calculated angular separation. It is then indexed on both star ID columns to ensure the rapid search speeds essential for real-time star pattern matching. Additionally, a supplementary script can be used to further reduce the database size by removing all pre-calculated angular distances that exceed the star tracker's FOV, thus eliminating data irrelevant for in-frame pattern matching.

Column	Data Type	Description
hip1	INTEGER	FOREIGN KEY. The hip_id of the first star in the pair.
hip2	INTEGER	FOREIGN KEY. The hip_id of the second star in the pair.
angular_distance	REAL	The calculated angular separation between hip1 and hip2, measured in decimal degrees.
y	REAL	The Y-component of the star's unit vector in the ICRS/equatorial system
z	REAL	The Z-component of the star's unit vector in the ICRS/equatorial system

Table 4.2: Angular Distances Table

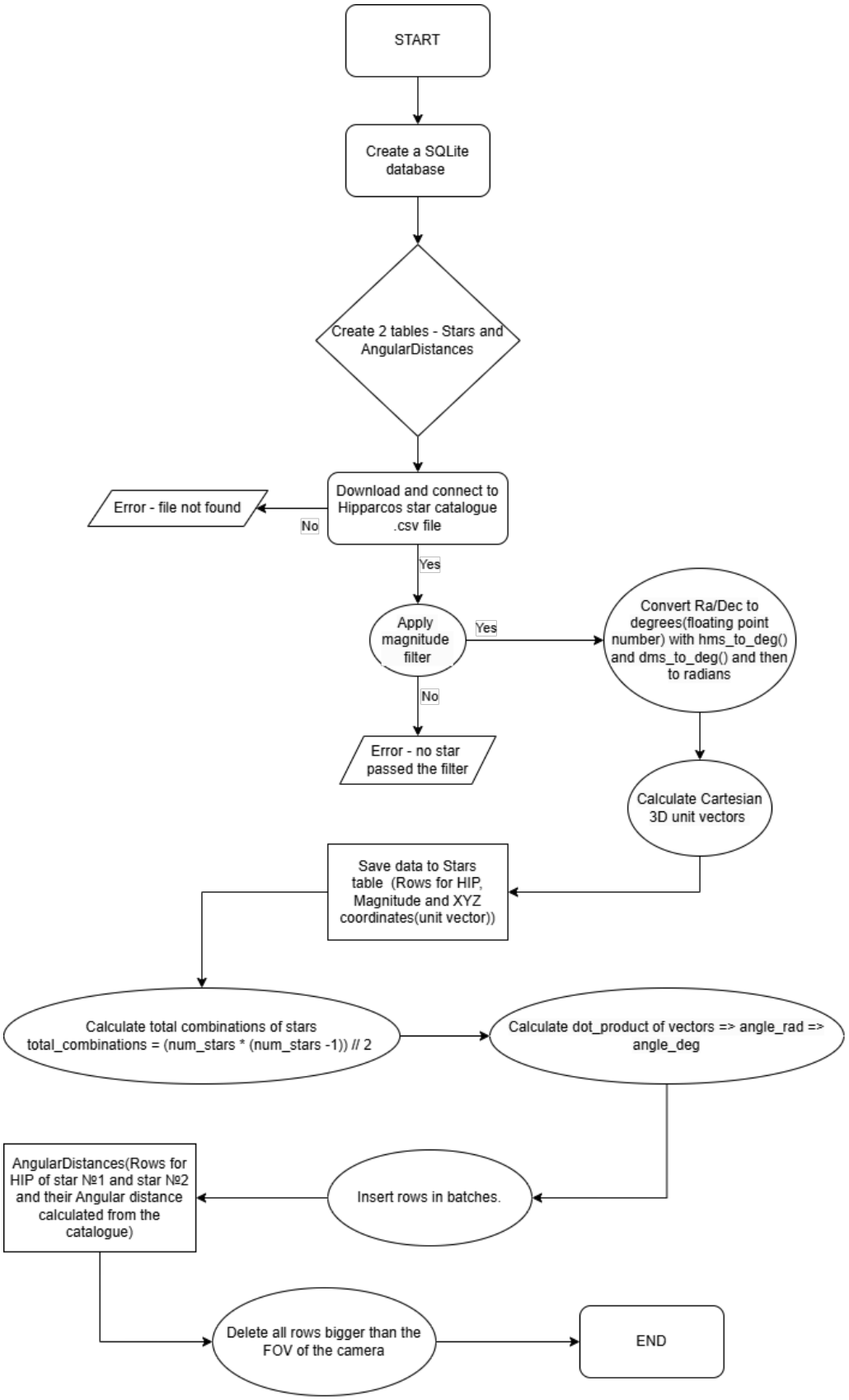

Figure 4.8: Data Base Diagram

4.3.3 Star Identification

Star identification involves two main steps: image processing and feature extraction. The initial phase is producing a high-fidelity list of star coordinates from an input image. This is accomplished through a multi-stage filtering pipeline designed to differentiate true stars from noise like hot pixels, cosmic rays, and other artifacts. This process, which uses Python libraries such as OpenCV for image processing, NumPy for numerical operations, and Matplotlib for visualization, results in a list of the most prominent star centroids, sorted by brightness.

The algorithm pipeline is a series of filters, and a candidate object must pass through every stage to be considered a valid star. It begins by ingesting a grayscale image, applying a Gaussian blur to reduce minor noise, and then using a binary threshold to isolate the brightest objects. The contours of these objects are then identified as potential star candidates. These candidates are then subjected to a series of geometric tests, including filters for area, circularity, and solidity. These tests efficiently discard objects that do not have the typical shape of a star as seen by a sensor. Candidates that pass the geometric filtering then undergo a final photometric analysis to confirm they have the brightness profile of a star, a bright core that fades into the background. This is done by comparing the average brightness of the central region of the star to the surrounding ring. A real star should have a significantly brighter core. The stars that successfully pass all these filters are then sorted by their area, which acts as a proxy for brightness. The final output is the pixel coordinates of these identified stars.

4.3.4 Attitude Determination

The camera has a diagonal FOV of about 78.6 degrees. The images it'll produce will be almost identical to the gnomonic projection of the stars in our FOV. A pinhole model of a camera can then be assumed to find the angular distances between stars in the image without accounting for warping. The pixel coordinates of the image are transformed into unit vectors using the optical centre and the focal length as in the equations below.

$$x = \frac{u - c_x}{f_x} \tag{4.5}$$

$$y = \frac{v - c_y}{f_y} \tag{4.6}$$

$$z = 1 \tag{4.7}$$

The angular distance between each star can then be found by taking the arccosine of the dot product of the unit vectors of the stars. The image angular distances are used to search through Table 2 (Chapter 4.3.2). However, this method produces three problems: angles can be offset by over 1 degree, producing inaccurate matches. Candidates for the angle search are more than one due to tolerances needed for pixel mapping. Finally, even if there is exactly one candidate pair from Table 2, there is still uncertainty in which star in the image pair corresponds to which ID in the database pair. The solution to this problem was graph theory. Thinking of each star as a node on a graph, predictions can be made

of what other possible adjacent stars there can be from the star catalogue based on the angular distance. This is called graph traversal. More specifically, a Depth First Search (DFS) algorithm that checks the closest matches of different combinations. Bearing in mind, only three identified stars are needed for an attitude determination, thus there is no need for every star to have a match. The best matching assignment is finally determined by implementing a scoring system that depends on the angular distance difference and the amount of stars identified. A perfect match would have a score of zero.

The QUEST algorithm was used to get the final orientation in quaternion form. During the QUEST algorithm, a cost function based on Wahba's problem was formed, which aims to minimize the weighted sum of squared errors between the observed and predicted directions. The weighted attitude profile matrix B was built, which takes into account each vector pair's weighting. After this, the Davenport's K matrix was constructed and solved for it's maximum eigenvalue, giving us our quaternion. The most recent weightings all have an initial weight of one, this will be further refined in the next iteration.

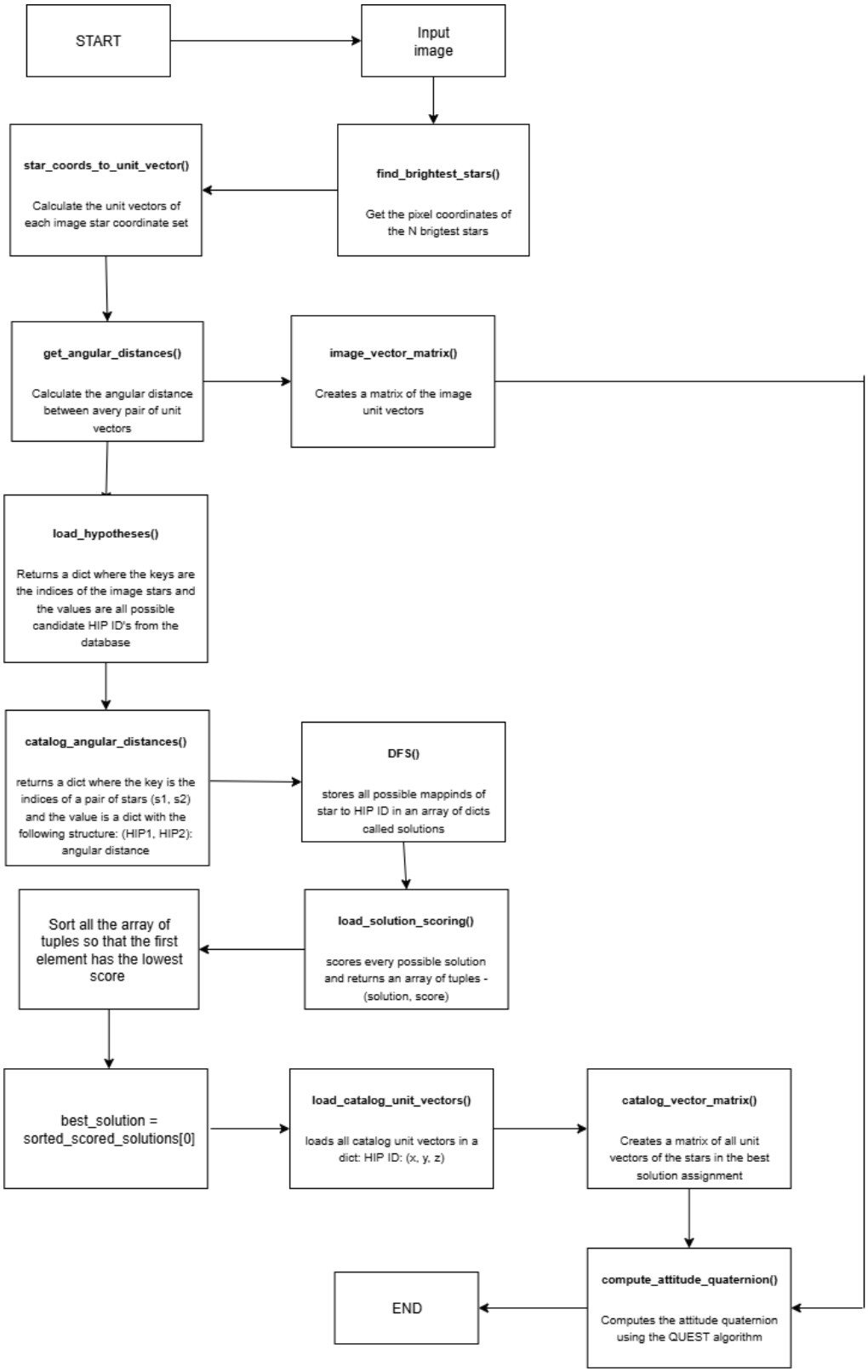

Figure 4.9: Lost in Space Algorithm

4.4 Mechanical Design

The physical container for the star tracker is 3D Printed in PolyTerra PLA enabling rapid prototyping and efficient implementation of design changes. The main purpose of the casing is to model how a star tracker module would fit inside new-generation modular CubeSats. In addition to the star tracker module, a prototype design for the frame of a 3U CubeSat will be made to demonstrate integration of the star tracker unit with other modules.

4.4.1 Mechanical Requirements

M.R.1: The star tracker module must fit within a $10\,\text{cm} \times 10\,\text{cm} \times 10\,\text{cm}$ volume.

M.R.2: The module must be able to be integrated with the rest of the satellite subsystems.

M.R.3: The material must have sufficiently high stress and temperature modulus to limit thermal expansion and compression, ensuring displacement error remains under $5\,\mu\text{m}$.

M.R.4: The star tracker camera should be oriented at an angle that maximizes deep space observation while maintaining the specific payload function.

Specialized material criteria:

M.R.5: Coefficient of Thermal Expansion (CTE) must be below $2\,\text{ppm}/^\circ\text{C}$.

M.R.6: Tensile strength must exceed $200\,\text{MPa}$.

M.R.7: Young's Modulus must be greater than $70\,\text{GPa}$ for the CubeSat frame.

M.R.8: Density must be less than $2800\,\text{kg}/\text{m}^3$.

M.R.9: Surface reflectivity must be under 5%.

4.4.2 Material Considerations

Star trackers are critical components of every satellite, crucial in determining their orientation and maintaining their function. Their high-precision requirements, environmental challenges in space, and miniaturization needs make the selection of materials highly complex. Although in this project, these material considerations cannot be implemented due to physical limitations, this section attempts to select materials based on stress analysis, temperature analysis, and cost.

Material selection will be prioritised on these properties: Coefficient of Thermal Expansion (CTE), Tensile Strength, Young's Modulus, Density, and Surface Reflectivity. This means the selection specifically selects materials with higher Young's Modulus and CTE to reduce

displacement errors of the star tracker due to compression and expansion of materials. Although factors like fracture resistance for protection against space debris and radiation resistance are important factors considered in many of today's satellites, for this project, it is assumed that improbable cases like space debris are impossible and that the orbit of the satellite is low enough such that the radiation is less than 10krad. Summarise all these factors, the Merit Index below is used for determining the optimal materials.

$$M = \frac{E \times \sigma_t \times S}{\alpha \times \rho} \tag{4.8}$$

Using the large material database MatWeb[13], materials with properties within the requirement are selected. For the housing unit of the star tracker, CFRP[21] is chosen for its very low CTE of -0.1 - 2 ppm/degree and high Young's modulus, which can be up to 250 GPa while being lighter (1550kg/m3) than materials such as titanium alloys (4430kg/m3). CFRP can also be easily machined depending on the layup. However, the downside is the cost at around 300 dollars per kg.

The baffle system has more specialised optical requirements, so Black Anodized Aluminium (7075-T6)[20] is selected. Apart from having physical properties within the mechanical requirements, it has desired optical and thermal properties. With a matte finish, it has high absorptivity and low reflectivity, minimising internal reflections which contaminate the optical signal, reducing reflectivity to under 3 percent. It has similar thermal expansion to optical lenses, which allows for higher compatibility with lenses. Along with high thermal conductivity, allowing heat to be distributed evenly. Aluminium can be machined easily and precisely, fitting for the precision needed in optics. In summary, choosing these materials enables not only a strong structure but also a structure which responds well to differences in temperature, crucial due to the involvement of optics. In reality, thermal subsystems and propulsion subsystems need to be used to regulate temperature ranges and also steer away from collisions, but the materials chosen are optimal in assisting the completion of the purpose of the star tracker.

4.4.3 Initial Prototype

The star tracker's structure was modeled using Autodesk Fusion 360[5] and then 3D printed with PLA filament[17]. The purpose of the initial prototype is to test the tolerance of the 3D printers to ensure optimal interface of the screw, camera front, USB ports, and the positioning of hardware. At the same time, the structure is used to fix the camera and the IMU in place in order for the test calibrations to be conducted. The initial prototype is also used for all initial testing of the interface between the Lost-in-Space program and the hardware. The image below shows the isometric CAD model as well as an image of the initial prototype.

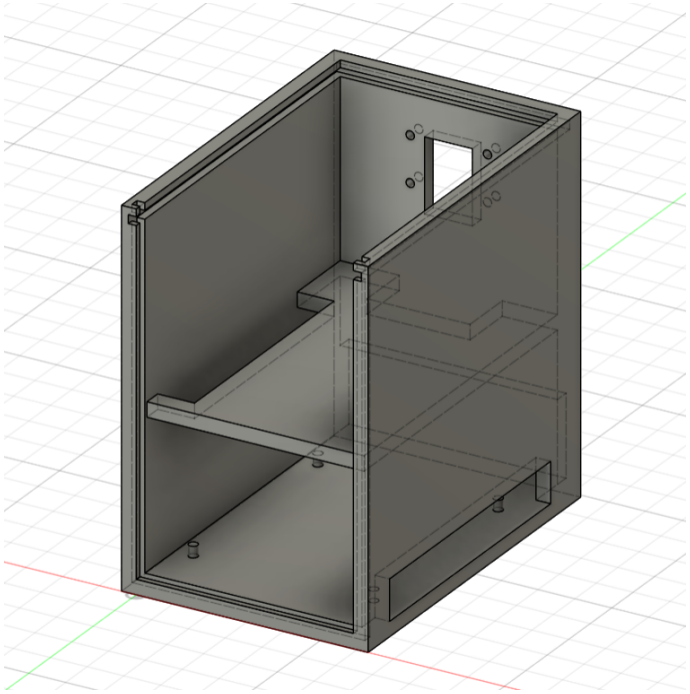

Figure 4.10: Initial Prototype of the Star Tracker's Structure

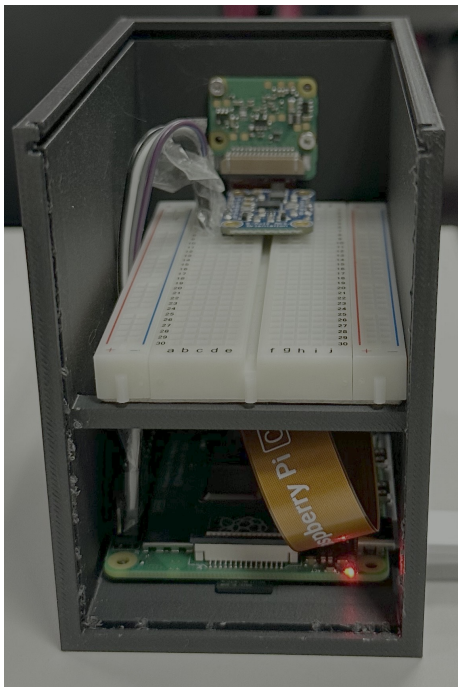

Figure 4.11: Integration of hardware components into initial Prototype

5 Testing

The process of testing is the most crucial step in any project. It is the first time you determine the accuracy of your theoretical model and any simulations. Testing is used to find any physical errors which could be overlooked in theory. Real-life examples include mispredicting the positioning of the sunshield on the Euclid telescope, which led to X-rays projecting an image of the body of the telescope onto the lens. Only after testing can the real product be manufactured.

5.1 General Testing of the Star Tracker

The tests conducted are centred around the system requirements. This can be split into four parts: determining the attitude, displaying orientation on the User Interface (UI), calibrating the IMU with the measured attitude (this allows the orientation to be tracked without the camera), and finally communicating with the rest of the system via Arduino. At the end, a test will be conducted where all four parts are integrated.

The attitude determination program test is conducted purely programmatically using algorithms heavily detailed in Chapter 4.3. Instead of a live camera image, the test uses a screenshot image randomly selected from a patch of sky in Stellarium. The image is processed through a filter, star identification, and angular identification using unit vectors. The first test measures the accuracy of star identification by comparing unit vectors recognised with known unit vectors of stars in the Stellarium image. After confirming the unit vectors, the quaternion can be determined using the QUEST algorithm, giving weighing to how each of the unit vectors contribute to the final attitude determination. This is then compared to the known image quaternion to complete the attitude test. The second test measures both the connection of the UI as well as attitude calculation using the camera. As detailed in Chapter 4.2.3, the UI contains a GUI showing the orientation of the module in Cartesian coordinates based on the quaternion calculated by the star image. The orientation of the module does not depend on the actual orientation of the box, but the positioning with respect to the image. For example, an image showing Polaris during the night will produce a GUI pointing towards the North at the inclination of Polaris, even if the module is pointing in another direction in real life. Knowing the rough attitude, in quaternions, of the Stellarium image vector, the orientation of the GUI can be compared to the expected orientation.

As detailed in prior chapters, the BNO055 automatically computes the orientation using sensor fusion of the accelerometer, gyroscope, and magnetometer []. As detailed in Chapter 4.2.3, the calibrated IMU (by the camera image) generates an offset from its reference point. The third test attempts to demonstrate the orientation displayed by the GUI post calibration. Solely depending on the IMU, the star tracker module is rotated. The success of this test will depend on how well the GUI model follows the star tracker's perturbations in reality. The second part of this test will show the periodic calibration of the program

in parallel with IMU dependency and the ability of the program to mitigate drift in orientation. The final test is the interface between the RaspberryPi with the Arduino. As in Chapter 4.2.3, they use the Universal Asynchronous Receiver/Transceiver in conjunction with the Octo bus transceiver with 3-state outputs to change the transmission voltage. The transmission of data will simply be confirmed in the Arduino console which will display 'message received'.

5.1.1 Final Experimental Setup

All aspects of general testing will be double-checked in a final experimental setup, combining all systems. This will be exhibited in the final presentation. Similar to the general test, the process will consist of determining the attitude quaternion from multiple Stellarium images, comparing its GUI orientation with the known orientation of the image, calibration of the IMU, and finally interfacing with the Arduino. However, to simulate conditions of space, instead of using a virtual image, a real image will be displayed on a screen, which can then be viewed by the star tracker in lightless conditions. To achieve this, a dark space is created using four spray-painted planes to surround the camera, with a drape to cover the opening for the star tracker itself.

6 Future Work

6.1 Prototype Development

Future work on the star tracker prototype can be divided into three major areas of improvement.

Adjustments to the modular physical structure should ensure compliance with standard CubeSat dimensional constraints. Optimization efforts must focus on component layout, as well as thermal and electrical isolation of the camera and IMU systems, to reduce signal interference and enhance accuracy. Additionally, the design should address thermal deformation, which may cause angular offsets. This can be mitigated by using space-grade thermal foils or thermally stable materials. The integration of standard interfaces, such as *SpaceWire*, will facilitate seamless communication with other CubeSat subsystems.

Refinement of the star pattern recognition algorithm and enhancement of the sensor fusion logic should lead to better precision in attitude determination. Leveraging onboard AI capabilities—such as dynamic noise filtering and adaptive star field identification under varying lighting conditions—will be key. Future iterations should also include support for remote software updates, allowing in-orbit performance tuning and algorithm optimization.

To increase system versatility, the optical platform can be extended to include Earth observation capabilities. This would involve software routines that allow the star tracker to switch between celestial navigation and terrestrial imaging modes. Integration of onboard AI to autonomously classify and prioritize imagery for downlink can improve mission adaptability and maximize scientific output.

6.2 Business Plan

CASSI is a compact, modular star tracker system tailored for CubeSats. It delivers precise attitude determination while maintaining low cost and ease of integration, particularly for educational and early-stage space missions. With its compatibility for CubeSat platforms and low-power operation, *CASSI* enables real-time orientation determination via onboard star recognition algorithms—without the complexity of conventional tracking systems.

The core software will be made available to students and academic institutions at accessible prices, supporting both educational initiatives and sustainable revenue models. This access empowers users to engage with real-world navigation algorithms and develop hands-on experience in areas such as sensor fusion, computer vision, and aerospace systems.

Comprehensive documentation, tutorials, and example projects will assist students and researchers in setting up the tracker, customizing its software, and deploying it in ground-based or near-space applications such as stratospheric balloon missions.

Beyond education, *CASSI*'s modular design encourages collaboration with commercial partners to co-develop custom star tracking units for specific mission profiles. This flexibility expands its reach into both commercial and scientific use cases, supporting a wide range of CubeSat-based applications.

In addition, downlinking data from satellites is extremely costly, primarily because of the high price of payload components with the star tracker being the most expensive. Our goal is to make star tracker manufacturing more affordable by using readily available and easy-to-use components. This reduction in cost will directly lower the expenses associated with data downlinking. By developing a modular star tracker that can be easily manufactured and integrated into satellites. This approach will accelerate innovation in the space industry and open the door for small to medium-sized companies to participate by launching reliable, cost-effective satellites. In turn, this will positively impact the entire space industry.

Bibliography

[1] AHRS Project. *Davenport'sq-Method filter documentation*. Documentation of Davenport's quaternion-based attitude estimator. 2025. URL: https://ahrs.readthedocs.io/en/latest/filters/davenport.html (visited on 08/01/2025).

[2] AHRS Project. *QUEST filter documentation*. Documentation on the QUEST (Quaternion ESTimator) attitude estimator in the AHRS Python library. 2025. URL: https://ahrs.readthedocs.io/en/latest/filters/quest.html (visited on 08/01/2025).

[3] Arduino. *Arduino Nano hardware documentation*. Official documentation for Arduino Nano board. 2025. URL: https://docs.arduino.cc/hardware/nano/ (visited on 08/01/2025).

[4] arthur_dent. *Star Recognition Using Computer Vision (OpenCV)*. Instructables project detailing HAAR-cascade star pattern recognition. 2020. URL: https://www.instructables.com/Star-Recognition-Using-Computer-Vision-OpenCV/ (visited on 08/01/2025).

[5] Autodesk. *Fusion 360 Overview*. Comprehensive cloud-based CAD, CAM, CAE, and PCB platform for product design and manufacturing. 2025. URL: https://www.autodesk.com/products/fusion-360/overview (visited on 08/01/2025).

[6] Bosch Sensortec / Adafruit. *BNO055 Intelligent 9-Axis Absolute Orientation Sensor Datasheet*. Bosch Sensortec datasheet published by Adafruit; fetched from official PDF. 2014. URL: https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf (visited on 08/01/2025).

[7] European Space Agency. *ESA CubeSats – Discovery Preparation Programme*. ESA Enabling Support page on CubeSat mission concept studies and programs. 2023. URL: https://www.esa.int/Enabling_Support/Preparing_for_the_Future/Discovery_and_Preparation/CubeSats (visited on 08/01/2025).

[8] European Space Agency. *Gaia Archive and Legacy ESA Science Archive (GEA / ESAC)*. ESA's Gaia/ESAC science data archive portal. 2025. URL: https://gea.esac.esa.int/archive/ (visited on 08/01/2025).

[9] European Space Agency. *Hipparcos and Tycho Catalogues Overview*. ESA Cosmos portal page describing the Hipparcos Tycho catalogues. 2025. URL: https://www.cosmos.esa.int/web/hipparcos/catalogues (visited on 08/01/2025).

[10] European Space Agency. *Hipparcos overview*. ESA official overview of the Hipparcos space astrometry mission. 2025. URL: https://www.esa.int/Science_Exploration/Space_Science/Hipparcos_overview (visited on 08/01/2025).

[11] J. Hegel. *Miniaturized Star Trackers for Small Satellites*. NASA CubeSat Symposium Presentation. Accessed 2025-08-01. 2018. URL: https://science.gsfc.nasa.gov/attic/cubesats.gsfc.nasa.gov/symposiums/2018/presentations/Day2/1115_Hegel.pdf (visited on 08/01/2025).

[12] Leonardo Electronics. *AA-STR / A-STR Medium Field of View Star Tracker*. AA-STR and A-STR are medium Field of View star trackers, radiation-hardened design. 2025. URL: https://electronics.leonardo.com/en/products/aastr (visited on 08/01/2025).

[13] MatWeb. *MatWeb: Online Materials Information Resource*. Comprehensive searchable database of material properties including metals, plastics, composites. 2025. URL: https://www.matweb.com/ (visited on 08/01/2025).

[14] NASA HEASARC. *TYCHO2 – Tycho-2 Catalogue of the 2.5 Million Brightest Stars*. Overview of the Tycho-2 Catalogue at NASA HEASARC. 2025. URL: https://heasarc.gsfc.nasa.gov/W3Browse/all/tycho2.html (visited on 08/01/2025).

[15] Raspberry Pi Foundation. *Official Raspberry Pi Camera Module documentation*. Camera hardware and software documentation from RaspberryPi website. 2025. URL: https://www.raspberrypi.com/documentation/accessories/camera.html (visited on 08/01/2025).

[16] Raspberry Pi Foundation. *Raspberry Pi 4 Model B specifications*. Product brief and specifications from official RaspberryPi site. 2025. URL: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (visited on 08/01/2025).

[17] RE3D. *PLA Filament Collection*. High-quality PLA filament for 3D printing, available in various colors and finishes. 2025. URL: https://re3d.eu/product-category/filament/pla/ (visited on 08/01/2025).

[18] Sodern. *HYDRA Star Tracker Datasheet*. Accessed 2025-08-01. 2017. URL: https://sodern.ariane.group/wp-content/uploads/sites/4/2018/04/Datasheet_HYDRA_2017.pdf (visited on 08/01/2025).

[19] Unknown. "Star Recognition Using Computer Vision". In: *IEEE Conference Proceedings*. Star-pattern recognition using OpenCV, IEEE article. 2017. DOI: 10.1109/7914959. URL: https://ieeexplore.ieee.org/abstract/document/7914959 (visited on 08/01/2025).

[20] Wikipedia contributors. *7075 aluminium alloy*. Technical summary of AA7075, its composition and aerospace applications. 2025. URL: https://en.wikipedia.org/wiki/7075_aluminium_alloy (visited on 08/01/2025).

[21] Wikipedia contributors. *Carbon-fiber-reinforced polymer*. Wikipedia overview of structure, properties and applications of CFRP. 2025. URL: https://en.wikipedia.org/wiki/Carbon-fiber_reinforced_polymer (visited on 08/01/2025).

7 Appendix

7.1 Product Specifications

GENERAL DESCRIPTION	
OPTICAL HEAD (OH)	
(Potential) Baffle protection for direct Sun and Earth illumination	
1 Optical Head may be connected to 1 Electronic Unit with up to 2 m length cable	
12-megapixel CMOS image sensor with on-board inferencing acceleration	
Integrated IR cut filter	
Manual adjustable focus (range: 20 cm – ∞)	
ELECTRONIC UNIT (EU)	
Embedded software processing OH's data and computing the attitude	
Embedded Star Catalog and Algorithms	
Lifetime of the EU can range between 1 and 5 years and can be enhanced with additional shielding for space missions	
Processor: Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz	
Connectivity includes dual-band Wi-Fi (2.4/5 GHz), Bluetooth 5.0/BLE, Gigabit Ethernet, 2× USB 3.0, and 2× USB 2.0 ports.	
Memory: 4 GB LPDDR4 with on-die ECC	
TECHNICAL SPECIFICATIONS	
ENVIRONMENTAL CHARACTERISTICS	
Operating temperature range (EU) (°C)	0–50ºC
Operating temperature range (OH) (°C)	0°C to 50°C
OH size (mm)	25 × 24 × 11.9 mm (heigh)
EU size (mm)	85 x 56 x 17 mm (heigh)
OH mass (kg)	0.006 kg
EU mass (kg)	0.075 kg
ELECTRICAL INTERFACES	
OH Power supply (V)	Supplied by EU
EU Power supply (V)	5V
OH Power consumption (W, typ/max)	1.0−1.25 W / 1.5−1.75 W
EU Power consumption (W, typ/max)	2.7W / 6.4W
EU Output data	Quaternion rotation
Output rate (Hz)	∼0.3 Hz
PERFORMANCES AND ROBUSTNESS	
Time from lost-in-space (typ)	7-10 s
Error (worst case)	2 deg
Image Processing	0.09 s
Full Moon in the Field of View	No performance degradation

Figure 7.1: Product Specifications

7.2 Bill of Materials

Item Description	QTY	Supplier	Function
Raspberry Pi AI Cameras	1	Raspberry Pi	system sensor
IMU Fusion Breakout - BNO055	1	Adafruit	system sensor
Raspberry PI 4	1	Raspberry Pi	Main computing unit
3D Printed Structure	1		mounting structure
M2x10 Countersunk Screws	4	Fastenright	camera mounting
M2 Hexagon Nuts	4	Fastenright	camera mounting

Table 7.1: Bill of materials for the Initial Prototype

7.3 Engineering Drawings

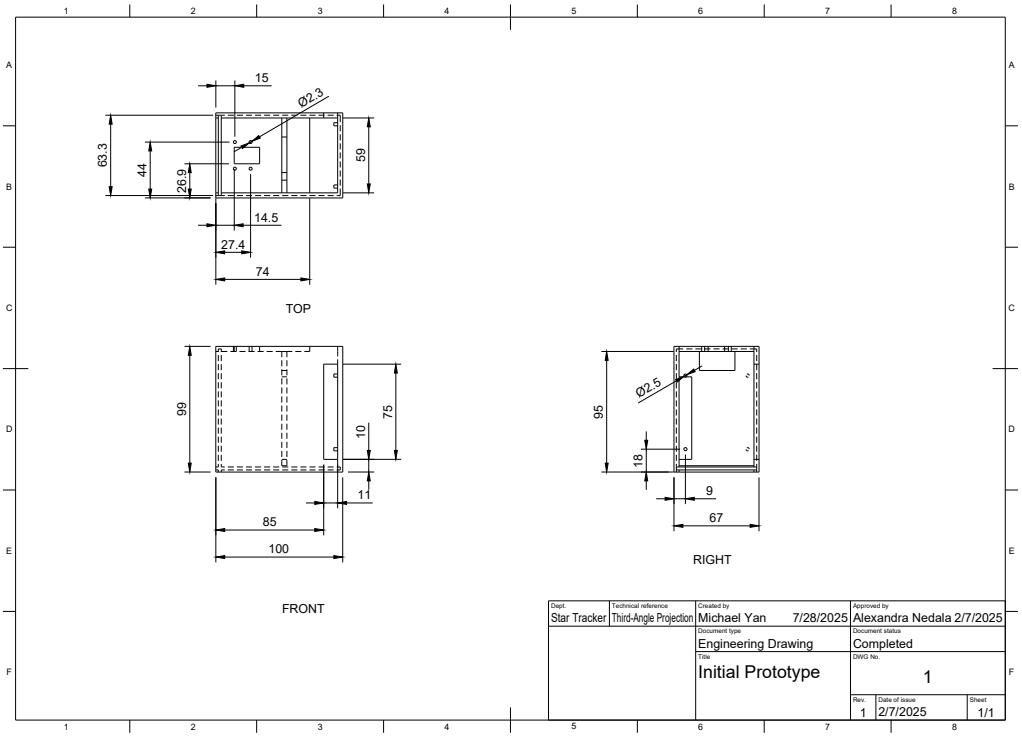

Figure 7.2: Initial Prototype Drawing

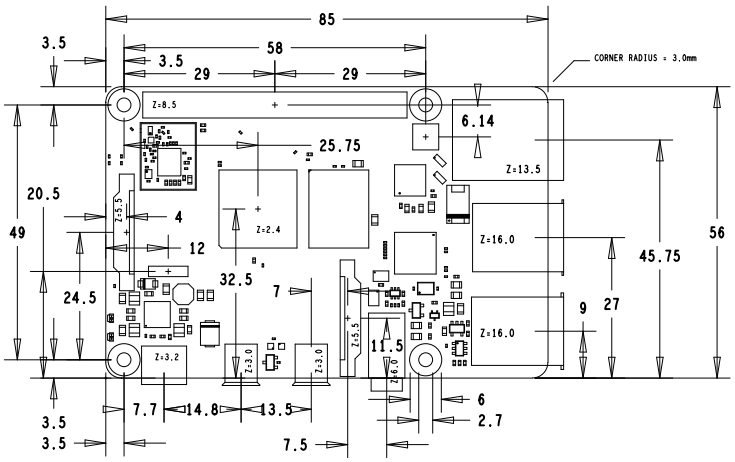

Figure 7.3: raspberry pi 4 mechanical-drawing

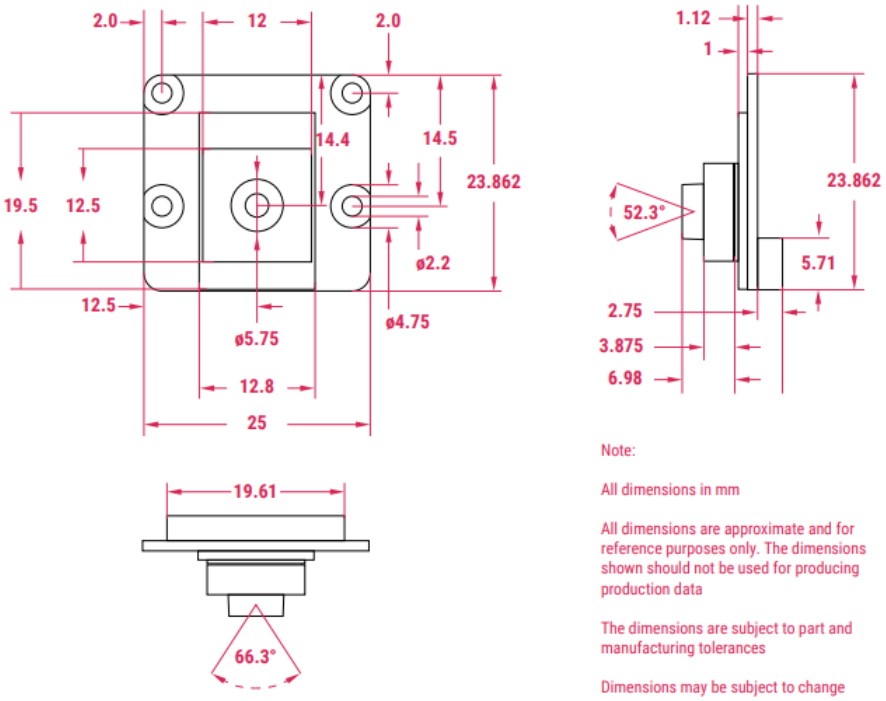

Figure 7.4: Raspberry Pi Camera mechanical-drawing

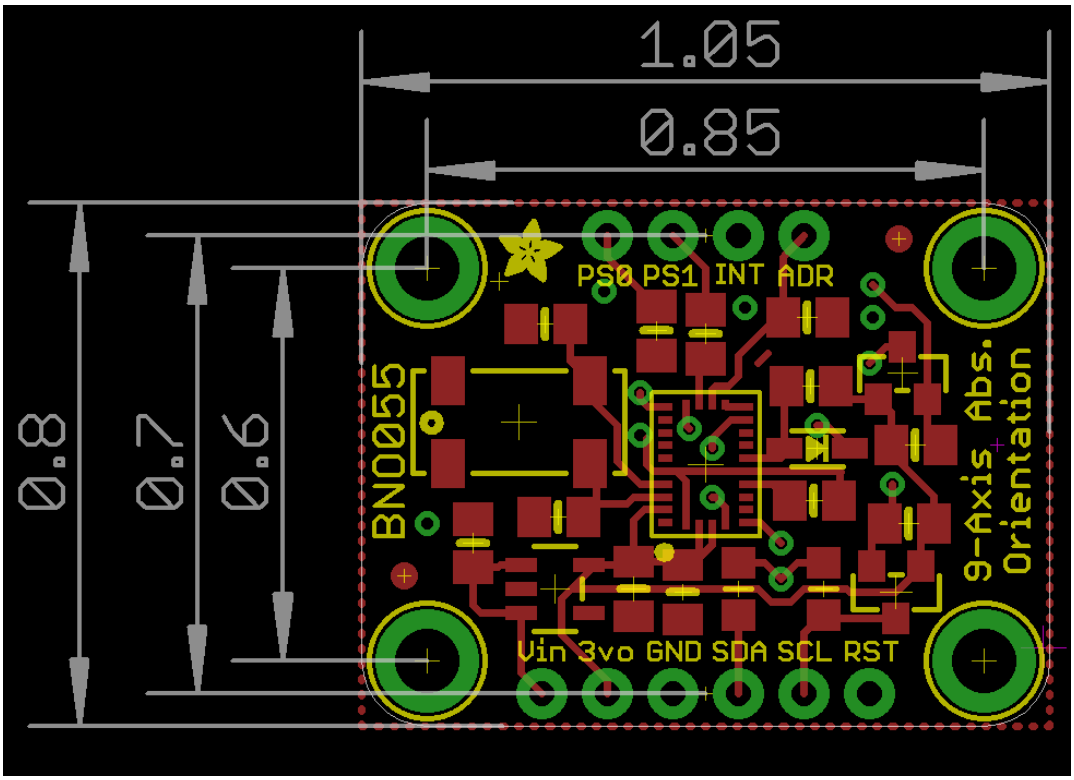

Figure 7.5: IMU BNO055 mechanical-drawing

7.4 Hardware Configuration

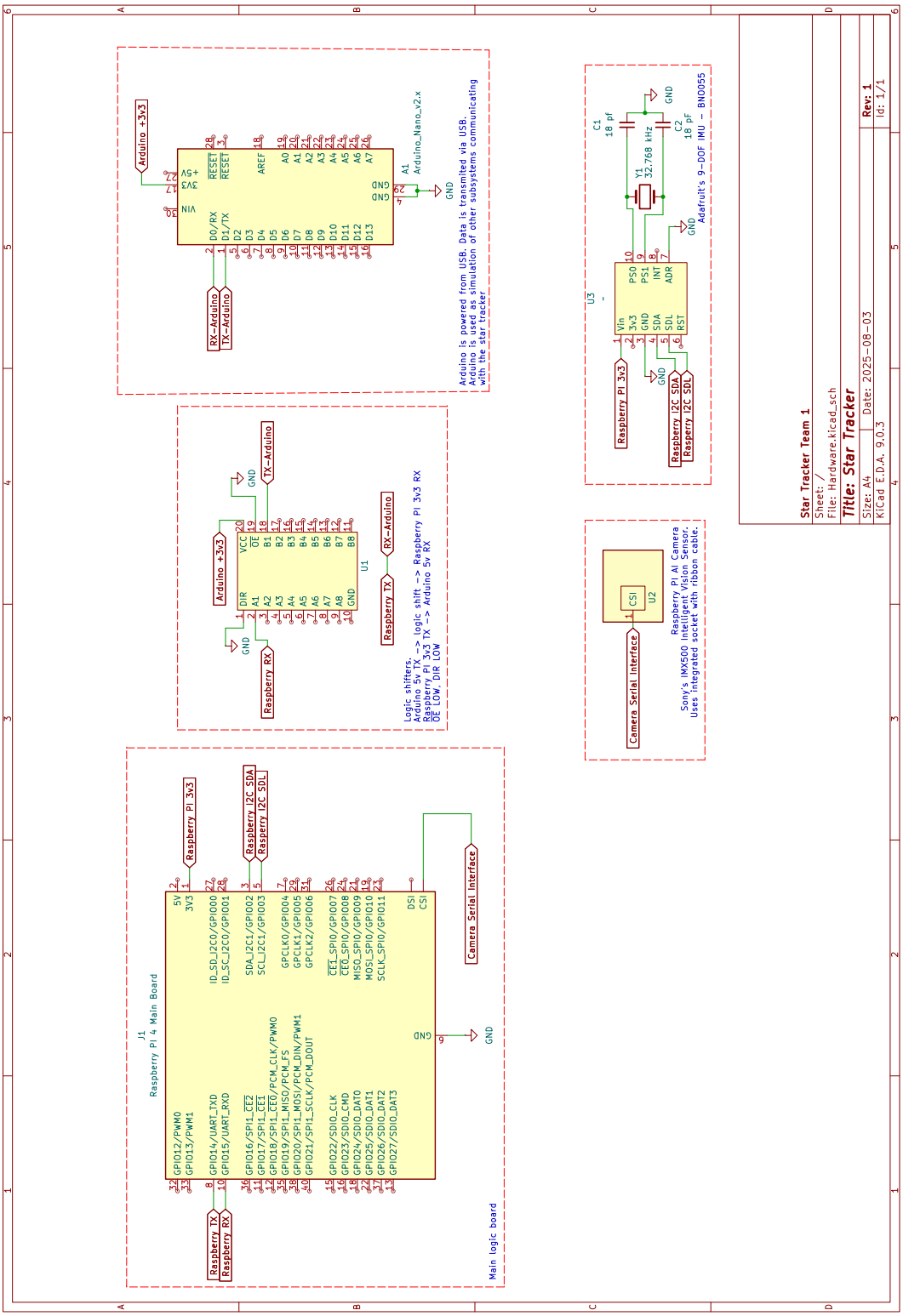

Figure 7.6: Hardware Configuration