Final draft - Hardware documentation

Introduction

Introduction to terminology:

- Main board - Small computer, in our case Raspberry PI 4, 4Gb
- IMU - Inertial Measurement Unit, in our case Bosch BNO055
- Camera - in our case SONY IMX500

The hardware system of the star tracker is designed for headless operation in setups:

- satellite integrated - no outside interfaces, just two-directional communication to another subsystem.
- workstation development environment - connected to WiFi, allowing access to development interface, data visualizations and command line interfaces (via the developer interface or secure shell).

If more information is required, please check out the BNO055 datasheet by Bosch.
Sadly we couldn't obtain access to the IMX500 datasheet by Sony. For more information contact them.

Hardware Specifications and main components List

The system is comprised of 3 main components. Camera, IMU and small main board computer.

Camera

- based on Sony IMX500 Intelligent Vision Sensor
  - Why is good: Big pixel count, fast operation and small form factor.
  - Problems: Big field of view, Manual focus (not motorized)

Specification	Details
Sensor	Sony IMX500
Resolution	12.3 megapixels
Sensor size	7.857 mm (type 1/2.3)
Pixel size	1.55 µm × 1.55 µm
Horizontal/vertical	4056 × 3040 pixels

Specification	Details
IR cut filter	Integrated
Autofocus system	Manual adjustable focus
Focus range	20 cm – ∞
Focal length	4.74 mm
Horizontal field of view	66.3 ±3 degrees
Vertical field of view	52.3 ±3 degrees
Focal ratio (F-stop)	F1.79
Infrared sensitive	No
Output	Image (Bayer RAW10), ISP output (YUV/RGB), ROI, metadata
Dimensions	25 × 24 × 11.9 mm
Cable connector	15 × 1 mm FPC or 22 × 0.5 mm FPC
Rated operating temperature	0°C to 50°C

IMU

- based on BOSCH BNO055 Sensor.
  - Why is good: Fast operation, simple interface, 9 axis, accurate.
  - Problems: closed algorithms

General

Feature	Details
Outputs	Quaternion, Euler angles, Rotation vector, Linear acceleration, Gravity, Heading
Sensors	16-bit triaxial gyroscope, 14-bit triaxial accelerometer, full-performance geomagnetic sensor
Package	LGA, 28 pins; Footprint: 3.8 × 5.2 mm^2; Height: 1.13 mm^2
Power Management	Intelligent Power Management: normal, low power, suspend modes
Voltage Supply	VDD: 2.4–3.6V; VDDIO: 1.7–3.6V
Digital Interface	I^2C, UART
Compliance	MSL1, RoHS compliant, halogen-free

Feature	Details
Operating Temperature	-40°C … +85°C

Accelerometer

Feature	Details
Acceleration Ranges	±2g, ±4g, ±8g, ±16g
Low-Pass Filter Bandwidth	1kHz – <8Hz
Operation Modes	Normal, Suspend, Low power, Standby, Deep suspend
On-chip Interrupts	Any-motion (slope), slow/no motion recognition, high-g detection

Gyroscope

Feature	Details
Ranges	±125°/s – ±2000°/s
Low-Pass Filter Bandwidth	523Hz – 12Hz
Operation Modes	Normal, Fast power up, Deep suspend, Suspend, Advanced power save
On-chip Interrupts	Any-motion (slope), high rate

Magnetometer

Feature	Details
Magnetic Field Range	±1300µT (x-, y-axis), ±2500µT (z-axis)
Magnetic Field Resolution	~0.3µT
Operating Modes	Low power, Regular, Enhanced regular, High Accuracy
Power Modes	Normal, Sleep, Suspend, Force

Main board

- Raspberry PI 4

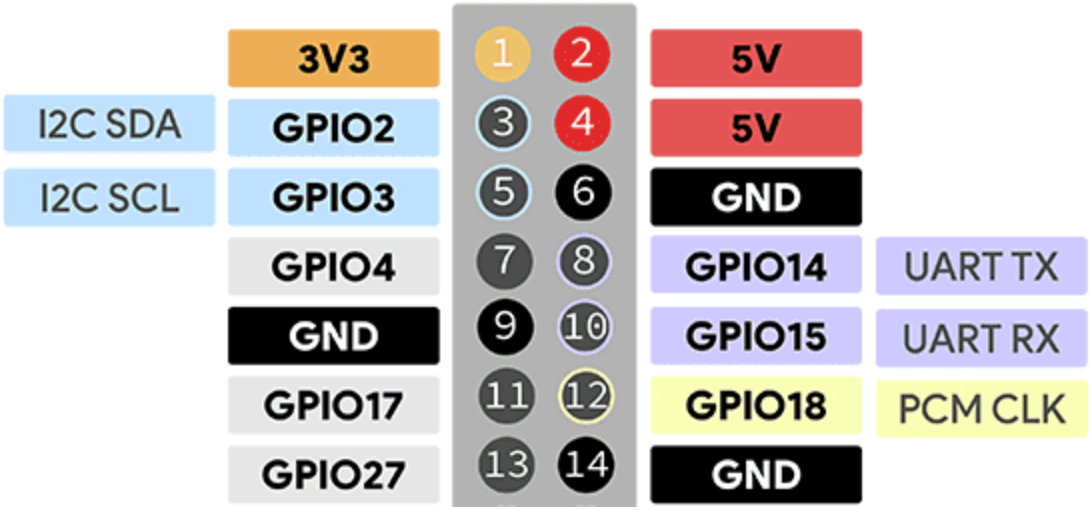
- Why is good: Enough computational power, low energy consumption, many interfaces, many libraries and different ways to integrate.
- Problems: Closed firmware. Too many unused features.

Feature	Specification
Processor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
Memory	4GB LPDDR4 with on-die ECC
Connectivity	• 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN • Bluetooth 5.0, BLE • Gigabit Ethernet • 2 × USB 3.0 ports • 2 × USB 2.0 ports
GPIO	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)
Video & Sound	• 2 × micro HDMI ports (up to 4Kp60 supported) • 2-lane MIPI DSI display port • 2-lane MIPI CSI camera port • 4-pole stereo audio and composite video port
Multimedia	H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES 3.0 graphics
SD Card Support	Micro SD card slot for loading operating system and data storage
Input Power	• 5V DC via USB-C connector (minimum 3A) • 5V DC via GPIO header (minimum 3A) • Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment	Operating temperature: 0–50ºC
MTBF	Ground Benign: 211,000 hours

Environmental Specifications

Connectors / Pinouts

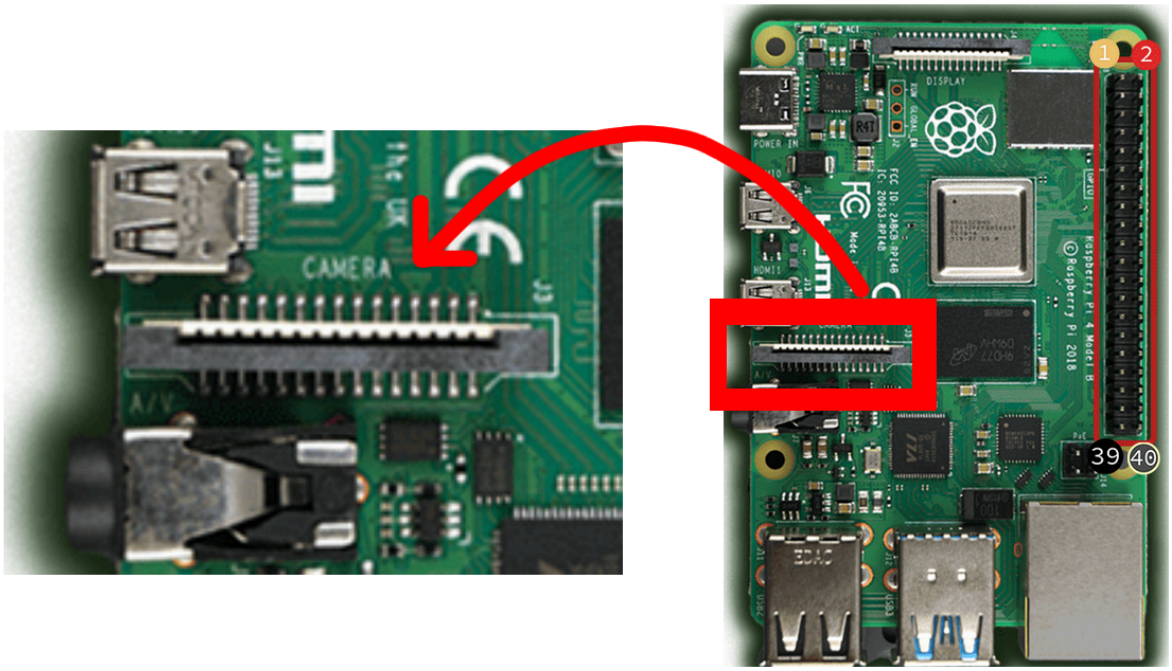
The product has Female Header (0.1-inch / 2.54 mm pitch) for connection with other subsystems and giving power to the device. The communication pins allow full-duplex UART communication . The power pins allow to power the device with +5 volts direct current.



Even though the main computer of the system has many connections, only few of them are used.

Pin name	alternative name	pin number	Used for
3v3		1	powering the IMU
GPIO2	I2C SDA	3	IMU data communication
GPIO3	I2C SCL	5	IMU clock sync
GPIO14	UART TX	8	Subsystem UART transmit
GPIO15	UART RX	10	Subsystem UART receive
GND		9	gounding to IMU

The Camera is connected directly to the Camera Serial Interface connector, via flat ribbon cable.



The current version uses perfboard (Stripboard or Veroboard) with connections on the bottom (similar to raspberry pi hats), allowing for easy assembly and better connection.

[picture here]

Power consumption

Interface Descriptions

- The IMU communicated with the main board via Inter-Integrated Circuit (I2C) communication, which allows for simple 2 wire interface.
- The camera communicated via MIPI (Mobile Industry Processor Interface) Camera Serial Interface standard, which allows high-bandwidth and low-latency data transfer.
- Other subsystems communicated via Full-duplex UART communication, which allows two directions communication at the same time. This interface can be expanded with the cubesat space protocl, like libscp.

  ==**!!! Important !!!** The communication protocol uses 3.3 volts due to requirements of the board and the specifications of the General Purpose Input/Output pins, even though the board is powered from 5 volts.
- Wifi is used to communicate with the developer interface and the secure shell protocol. This allows for updates, setup of the system and changes in the source code.

Firmware

Operating system

The operating system of the device is minimalist version of Linux Debian, made specifically for raspberry pi - **Raspberry Pi OS Lite 32-bit**. It runs in **headless** mode with minimal software and drivers, it provides good embedded environment for experimentation with simple integration and possibility for expansion.

Specification	Data
Operating system version	Debian version 12 (Bookworm)
Firmware version	24.10.0
Operating system bit	32-bits
Size	493 MB
Kernel version	6.12

System architecture

The code is written on python with many modules design for specific purpose:

- main - Combines all modules, runs specific parts in threads, cleans up after execution
- logger - Saves logs, displays information, displays logo, headers, organized output
- math - Computations and arithmetic functions
- flask_server - https server for live video feed and picture download (used for developer interface)
- web_sockets - tcp connection for command and live information transfer (used for developer interface)
- lost_in_space - main algorithms for calculating the attitude based on camera information.
- image_processing - support algorithms for lost_in_space. Identifying stars, etc.
- database and hashmap - all needed values for computation of the algorithms

External Device Compatibility

We need to consider that the connection with other subsystems require additional the voltage to be around 3.3 volts. The reason is the operational voltage of the main computer in the device. If you want to operate outside the scope of this, you risk to damage the internal computer.

Boot cycle efficiency and optimizations

specific settings were set into place to ensure faster load time of the system.
This are the current settings of the `/tmp/firmware/config.txt` :

```
disable_splash=1
boot_delay=0
initial_turbo=30
dtparam=audio=off
dtparam=spi=off
display_auto_detect=0
```

There is problem, where the allocation of less than 32 Mb of memory for the graphical processing unit, don't allow the camera to initialize correctly.

```
gpu_mem=16
```

For now we will leave it at default value of 128 Mb to ensure stability.
This didn't impact the boot time.

the most improvement in boot speed came from

```
sudo systemctl disable NetworkManager-wait-online.service
```

This cut the boot time with 50%.

other services were removed to make the device faster and remove unnecessary scripts and programs running in the background

```
bluetooth
hciuart
avahi-daemon
triggerhappy
rpi-eeprom-update
keyboard-setup
```

Attempts to optimize the SD card, where the whole firmware was stored, were unsuccessful

```
(example)
dtparam=sd_overclock=100 - didn't work, not used
```

after this optimizations we only got down with just 8%

At the end, we got from around 11 seconds boot time to 5.5 seconds. Which is in the range of most cubesats boot time - around 10-60 seconds.

The shutdown time is around 3-4 seconds, which grants us around 15 second reboot time (some delays present) for the operating system.

The IMU has start-up time from off to configuration mode in 400 milliseconds typical time. and from reset to normal mode in 650 milliseconds mode.

System boot sequence

put graph of boot sequence that you showed to Simo

operation

flow charts
systems
architecture
data handling
whatever there is

Power consumption

The main board on the system is rated for direct current from power supply, which can deliver 3 amps at 5 volts. The documentation of the device rates typical bare-board active current consumption at 600 milliamps. The GPIO pins can deliver 3.3v at max 16 milliamps current.

step/process	Amps drawn (A)
Booting - max	0.85
Booting - avg.	0.7
Booting - min	0.6
Stress - max	1.25
Stress - average	1.2
Halt current (shutdown state)	0.023

Note: These measurements used a standard Raspberry Pi OS image (current as of 26 Feb 2016, or June 2019 for the Raspberry Pi 4), at room temperature, with the Raspberry Pi connected to a HDMI monitor, USB keyboard, and USB mouse, connected to Ethernet. Power consumption can easily exceed these measurements if multiple additional USB devices or a HAT are connected to the Raspberry Pi.

The IMU runs from 2.4V to 3.6V. Total supply current in normal mode with 9 degrees of freedom at 100 Hz output data rate is 12.3 milliamps. The IMU can run in few modes that can save

power (like "Low power"). We currently only work with normal operation with stopped fast calibration of the magnetometer.

Possible upgrades:

- Possible extensions on the system to detect current drop below 4.63V (±5%) can be added in the future.

Single event upsets (SEU)

In **Low Earth Orbit (LEO)**: SEFIs are expected about **once every ~100 days** per Pi (worst case). Actual rates may be 2–3× lower.
There is no built-in hardware mitigation. Because Raspberry PIs are designed for commercial on ground off-the-shelf deployment. They don't have any hardware integrations for error-correcting memory or hardened logic. This single event upsets can flip specific sectors, break calculations or even crash the system. There is no **Error-Correcting Code (ECC)** memory on the chip.

Possible problem can be:

- **bit errors:** crashes, corrupted variables, loops running out of bounds.
- **Single event function interupts:** Linux kernel panics, system lockups, or garbage communication output.
- **Permanent failures:** rare but possible from latchups or SD card corruption

Some software implementations can be introduced on the system, but they are not as reliable as hardware that is specifically space-graded. For this version we use most of the default configuration of the operating system, so many things like watch dogs are not activated.

Possible upgrades:

- Install protection mechanisms:
  - Automatic power-cycling if error is detected
  - Change SD card with more redundant/space-graded technology
  - External ECC memory modules
  - filesystems like **Btrfs** or **ZFS** to detect corruption and/or other application level checksum
  - hardware watchdogs timers (Raspberry pi have software watch dogs, but separate modules are mandatory for secure operation)
  - Structural shielding from particles

IMU Settings

We run the IMU in **NDOF_FMC_OFF** to stop recalibration after booting and loosing general direction. This allows us to not lose direction in case of rebooting the software ran on the star tracker. We stopped the heading calibration and load custom calibration from file. The system is designed for LEO missions, so the magnetometer can be used for calibration of the system. Here few design choices come into play when deciding if magnetometer recalibration on boot is required. In our case we choose to not do it and allow the lost in space algorithm to run the recalibration. This can be changed in future versions or depending on customer requirements.

The IMU settings allow for change in the axis mapping. This is done with a register *AXIS_MAP_CONFIG* and can help when the IMU is not places directly toward the camera. After that we do 45 degree offset to align it with the camera, software wise.

In our NDOF_FMC_OFF mode we allow for the outputs rates of:

Accel	**Mag**	**Gyro**	**Fusion data**
100 Hz	20 Hz	100 Hz	100 Hz

When booting in normal operation mode, by default we load this standard configuration mode

Sensor	**Parameter**	**Value**
Accelerometer	Power Mode	NORMAL
	Range	±4g
	Bandwidth	62.5 Hz
	Resolution	14 bits
Gyroscope	Power Mode	NORMAL
	Range	2000 °/s
	Bandwidth	32 Hz
	Resolution	16 bits
Magnetometer	Power Mode	FORCED
	ODR	20 Hz
	XY Repetition	15
	Z Repetition	16
	Resolution x/y/z	13/13/15 bits

Some of this values are not controllable directly from the developer, but are changed based on the fusion mode used in the chip.

Installation & Setup

Most of this should be expanded, because in current state we haven't prepared shipping and packaging. Current development may change the form factor. This is general overview for future documentation

Unboxing Instructions

The packaging materials are not ready, but this is the planed system:

Open the box you received. Remove the top packing material. Carefully take the Star tracker out of the box. In the box you will also find additional manuals, wires, connector, adapters, testing stand, spare bolts, cleaning materials, stickers, etc..
No special protection or equipment is needed if you want to use it in desk mode, to learn and experiment.

Protective covers

The star tracker relies on camera that need to be clean and ready to take picture at any time. For this reason the device comes with protective cap on top of the camera. Its purpose is to stop dust, humidity or any outside particles and environment variables to interact and lower the efficiency of the star tracker. Remove this cap for testing and before flight. Remember to clean the camera with the provided cleaning materials and/or according to the instructions.

Cleaning the Star Tracker

1. **Power Off:** Make sure the device is turned off and unplugged from all power sources.
2. **Dust Removal:** Use a can of clean, dry compressed air to blow away any dust from the lens, connectors, and vents. Hold the can upright and spray in short bursts to avoid condensation.
3. **Lens Cleaning:**
    - If there are smudges on the lens, use a lens cleaning pen or a microfiber cloth made for camera optics.
    - Do not press hard — wipe gently in a circular motion.
    - If needed, add a drop of optical lens cleaning solution to the cloth (never directly on the lens).
4. **Body Cleaning:** Wipe the housing with a soft, dry cloth. Slightly damp cloths are fine for stubborn marks. Never let liquid enter any opening.

Protective Covers

- **Lens Cap:** Always put the supplied lens cap on when the Star Tracker is not in use. This prevents dust, scratches, and accidental fingerprints.
- **Connector Covers:** If your model includes rubber or plastic covers for USB-C, UART, or power connectors, place them after each use to keep out dust and debris.
- **Transport Case:** For storage or transport, place the Star Tracker back into its original foam-lined box or into a padded protective case to protect against shock and vibration.
  Important:
- Never use household cleaning sprays, alcohol, or abrasive materials on the lens or housing.
- Avoid storing the unit in direct sunlight, humid areas, or near strong magnetic/electrical fields.

Physical Installation / Mounting Instructions

The star tracker is design to install in CubeSat with sizes of 1U to 3U, or any system that can support the design [please expands]. The system also can be installed on desk or workstation, connected via usb-c for power.
All bolts in the design are m2 screws with specific length and star head end, coutner sunk or some shit, idk, Michael your turn

Electrical Connections

The raspberry pi can be connected via usb-c for desk/workstation operation
When you need to install the system in the satellite, connect the UART and power pins based on the provided diagram.

Initial Power-Up Procedure

Power the device with the power requirements described in section [somewhere]

Before powering the device make sure all connection are secure and there is no visible damage on the device. Ensure Ethernet cable is connected so that you can connect to via secure shell or the developer interface and control the software.
When the device received power, the booting process starts.
On first boot it enters into standard desk operation. In this way you can connect to it and change the system parameters.

Future upgrades:

- make pin for selecting the operational based if it connected to ground or not

Configuration and Calibration

The device comes with small white circle with small pins to calibrate the focus of the camera. The current setup requires to use external monitor for testing (16:9 aspect ration), positioning the camera module to be able to capture the whole monitor. After positioning you can rotate the camera lens, by positioning the calibration tool on top of the camera lens, by aligning the hole in the middle. Look at the developer software to see the change of focus. In this way you can find the optimal focus length for your test bench. The provided stand is for 23 inch monitor with 16:9 aspect ratio.
You can test the functionality from the command line interface or the buttons provided. The secure shell connection doesn't allow for command to be send when we are in ssh session with booted and displayed Star Tracker Software. In that case you control the system via the Developer interface. The ssh is used for direct control and access of the source code.

Future upgrades:

- Stand with ability to manually change position, based on the size of screen.
- Better secure shell CLI interfacing for direct control without the GUI.

Maintenance & Troubleshooting

- Routine Maintenance Guidelines
- Troubleshooting Guide (symptoms → causes → fixes)

Firmware Update Procedure

if you want to update the firmware you can download and flash an iso by contacting customer support from the link on our website or download the source code from Github and install it manually on the device via the scp.
https://github.com/IvaylooTs/StarTracker
Note: Save your settings by exporting them manually. You can easily override them and lose valuable configuration settings.

Future upgrade:

- Simple export and import for settings via the developer interface and ssh

Replacement Parts / Consumables

The raspberry pi 4Gb, sd card, AI camera and IMU can be found from many distributors and this makes the current version of the system - accessible and easily reparable. For the pcb design, structure or any other part, contact us via our customer email or our website.

Developer Interface

The developer interface is designed specifically to run on computer, connected on the same network as the main computer. It provides overview and visualization of the most important information. It is programmed with html, css and javascript like a normal website. Libraries used are Three.js for 3d visual visualization and Graph.js for quaternion data visualization in graphs

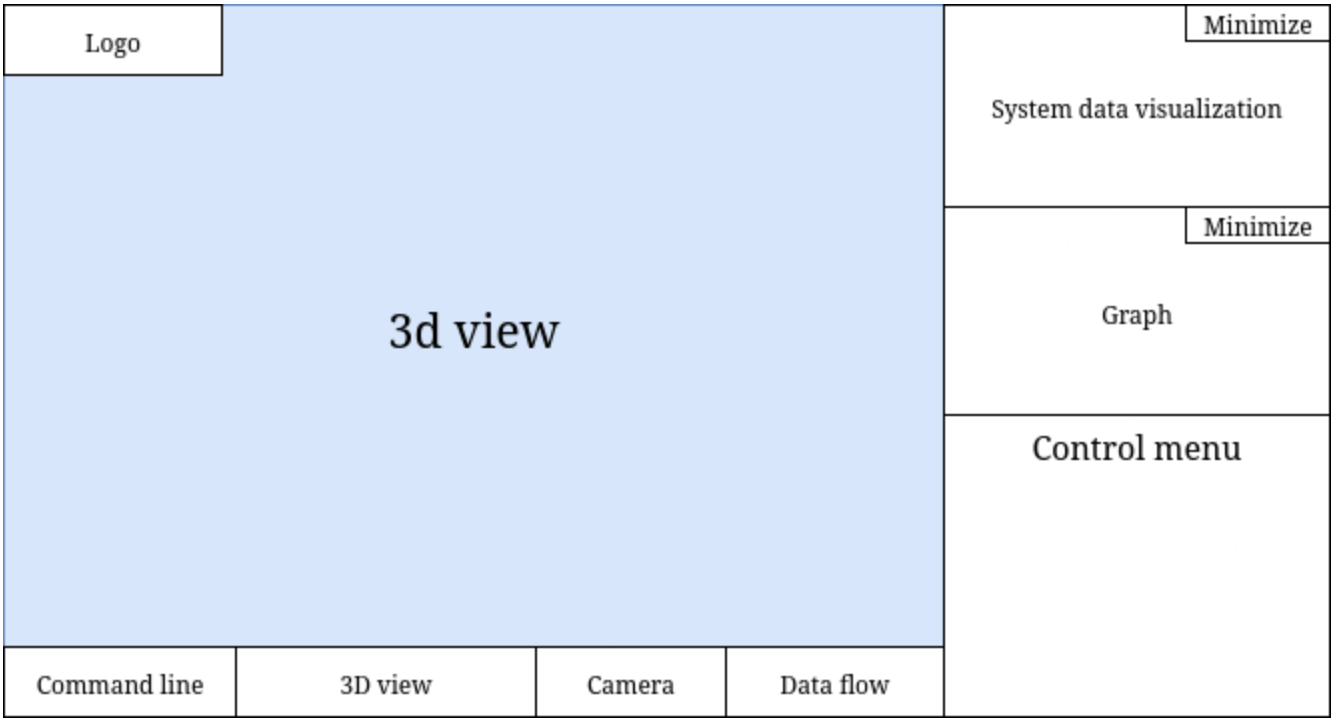
It connects via websocket on port 6789 to transmit data in two directional. The data is transfer in json format. The main board sends constantly data about its quaternion rotation, temperature, cpu and ram loads. Any other information is manually requested with command in the command line interface (or just console) inside the developer interface. Few command can be send via buttons for faster call:

- save and download current video feed image as picture on the user computer
- activate test with image on board
- activate lost in space with camera view
- add custom quaternion value
- calibrate IMU to last calibration
  On the left of the terminal is present live data feed and on the right are last command and acknowledgements made from the raspberry pi and the system.
  If connection is lost or not made, websocket tries to reconnect. Some problem are recorded that it can take some time to establish connection. It can range between instant connection to 10-15 seconds wait. Refreshing the interface, rebooting the software or cleaning the cached page data seem to help for now. (this will be fixed in future developments)

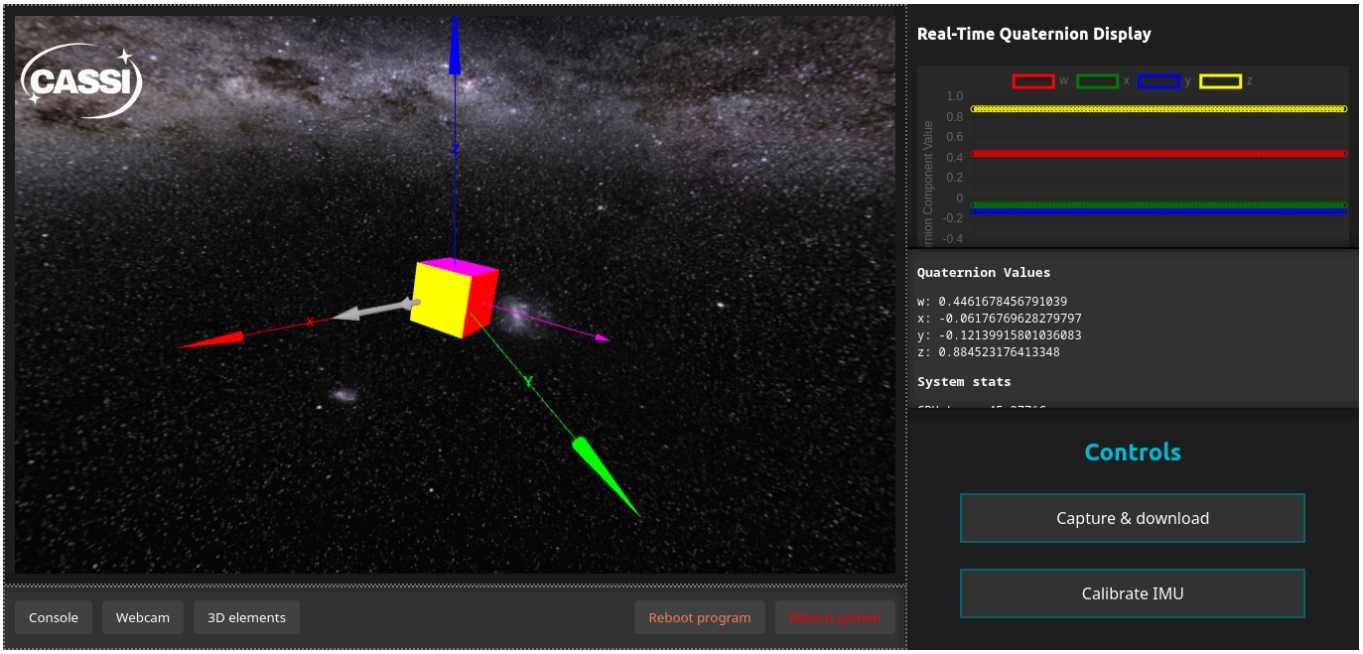
Flask server is loaded with live feed from the camera. It can be accessed on port 5000 from any browser in with URL "https://{ip}:5000/video_feed". It is also displayed in small window in the developer interface. We use https, because of cross site resource management and display. The problem occurs, because the video feed is hosted on the raspberry pi and the visualization happens on our web interface on another computer. We use locally generated certifications and we need to connect to the link above to accept "the risks" with connecting to this system.

Note: The Websocket and Flask server are not active in Satellite mode.

Initial concept design:

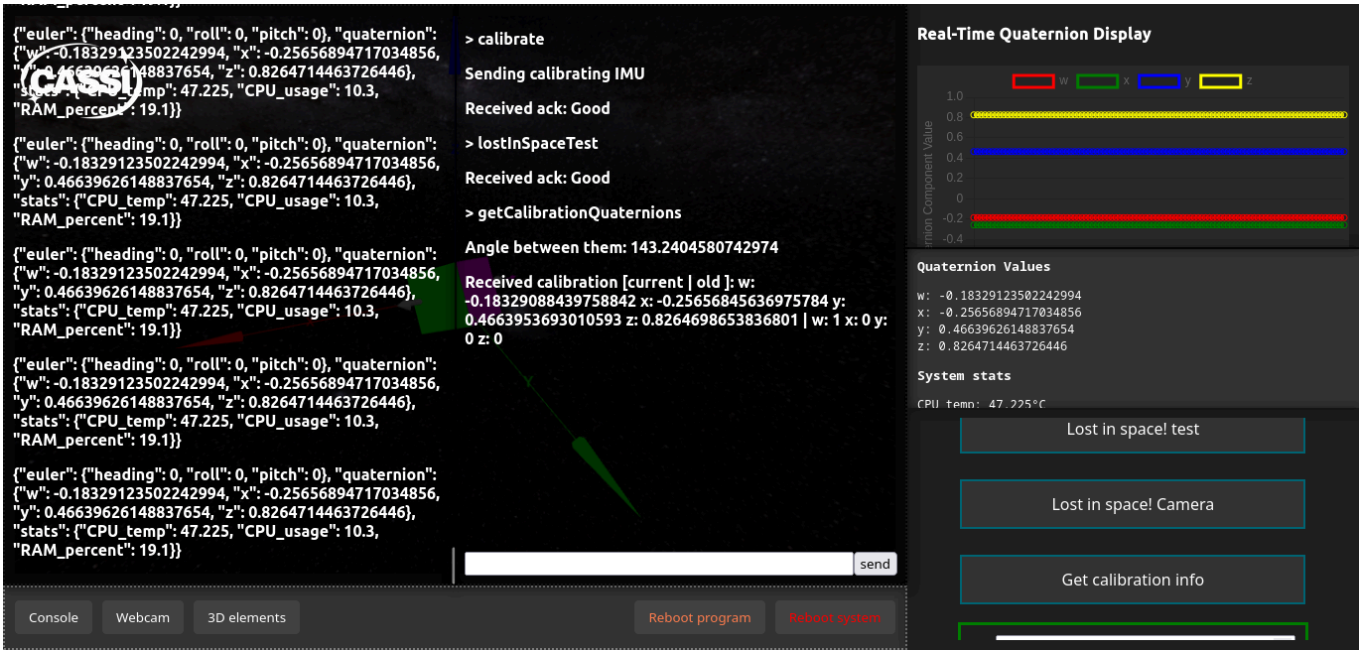


Current version:

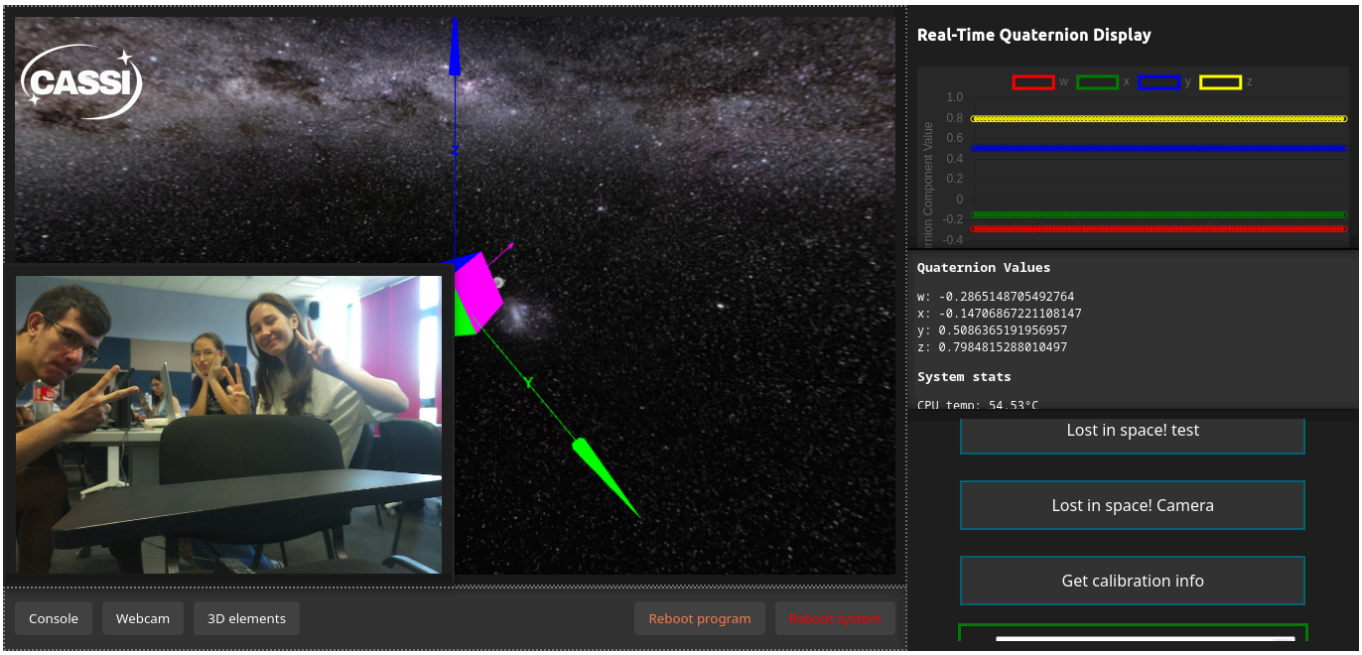


Most of the things are the same as the concept. We can see that the the graph and system data visualization are switched. on the panel on the right you can scroll down to see more control buttons for testing (in controls panel), and system stats (in the visualized data).

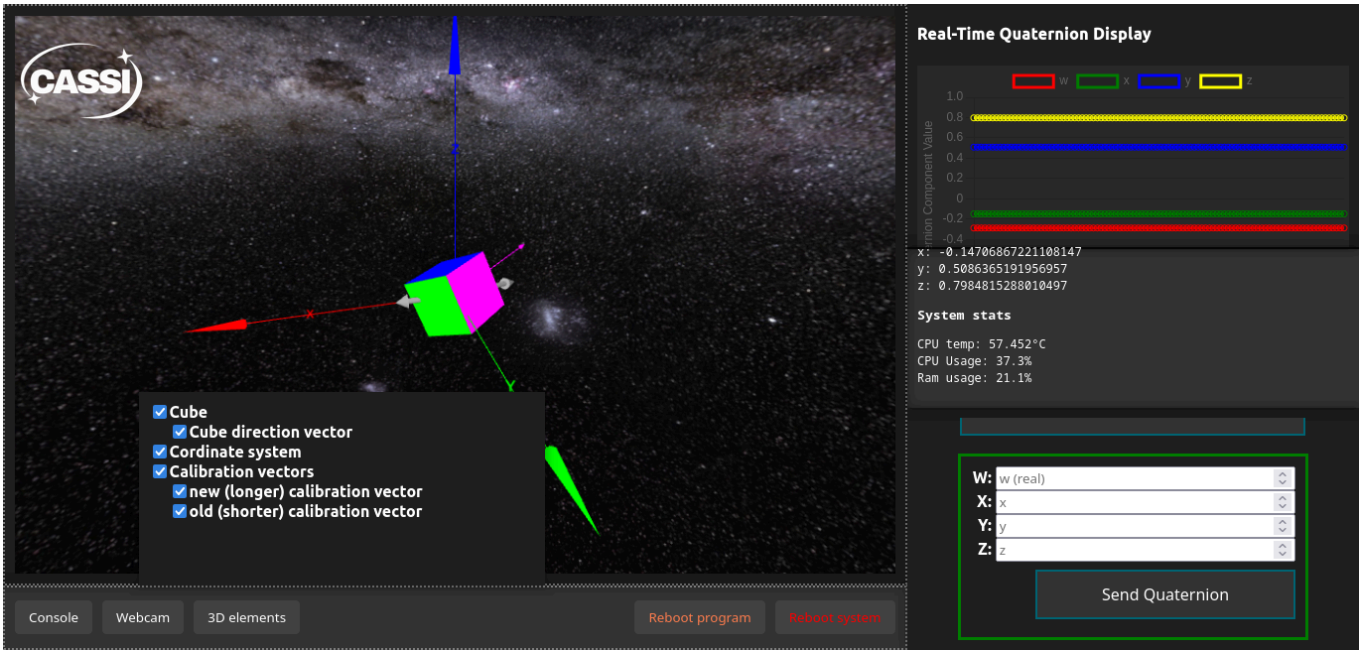
Console visualized:



Camera window:



Visualization toggle:



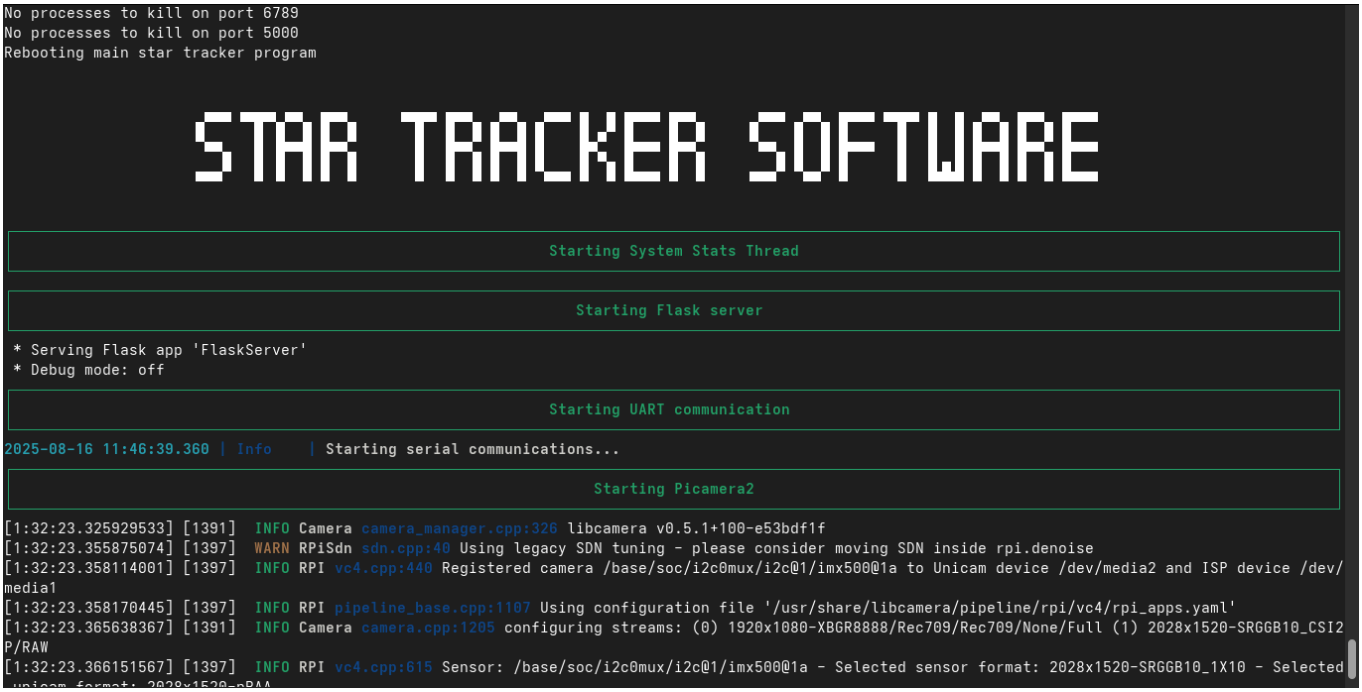
Note: The star visualization on the 3D visualization is only visible when the python script 'start.py' is started and we are connected to the local ip. We cannot load the image the image because of some html problems and resource management.

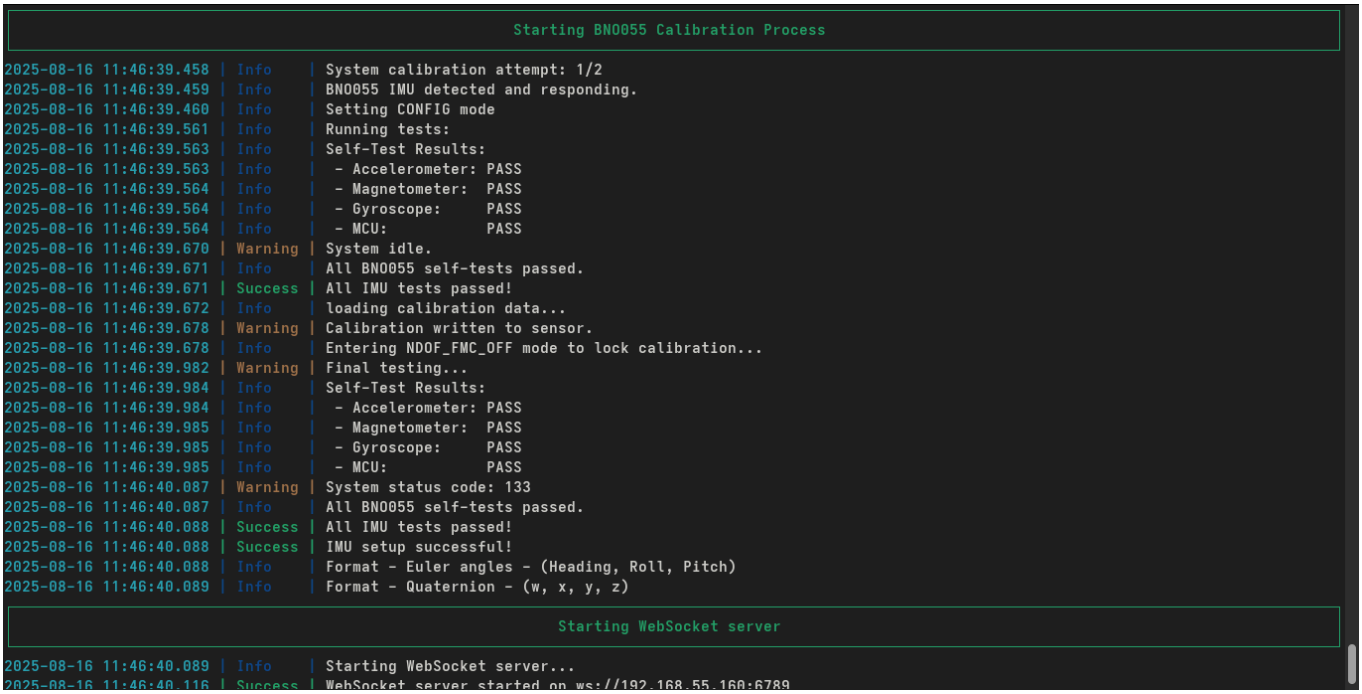
Future upgrades:

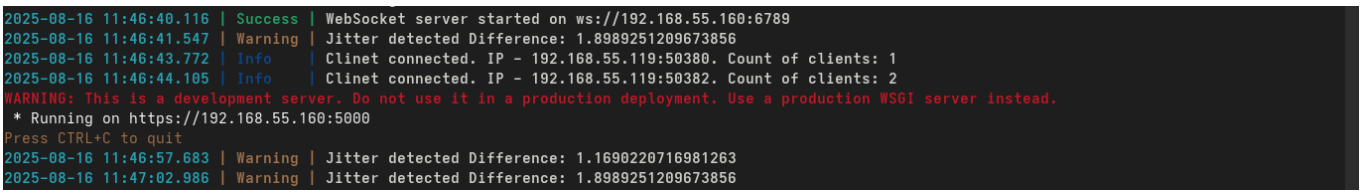
- more control buttons
- better visualization
- movable windwos
- more graphs and information

On board CLI

When you boot into the raspberry pi and run the script manually, you see this visualization







This helps you display debugging information

The problem is that you cannot write CLI commands while you are in this interface.
For future development we can add this function.

Testing & Validation of hardware

- Factory Test Procedures
- Quality Control Checklist
- Environmental Stress Tests
- Electrical and Functional Testing

Diagrams & Schematics

- Block Diagrams
- Circuit Schematics
- PCB Layouts
- Wiring Diagrams

Developer interface

Appendices

- Glossary of Terms
- Reference Documents / Standards
- Revision History / Changelog
- Contact Information / Support

https://randomnerdtutorials.com/raspberry-pi-pinout-gpios/

https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#power-supply
https://nepp.nasa.gov/docs/papers/2021-Guertin-Raspberry-Pi-Guideline-CL-21-5641.pdf