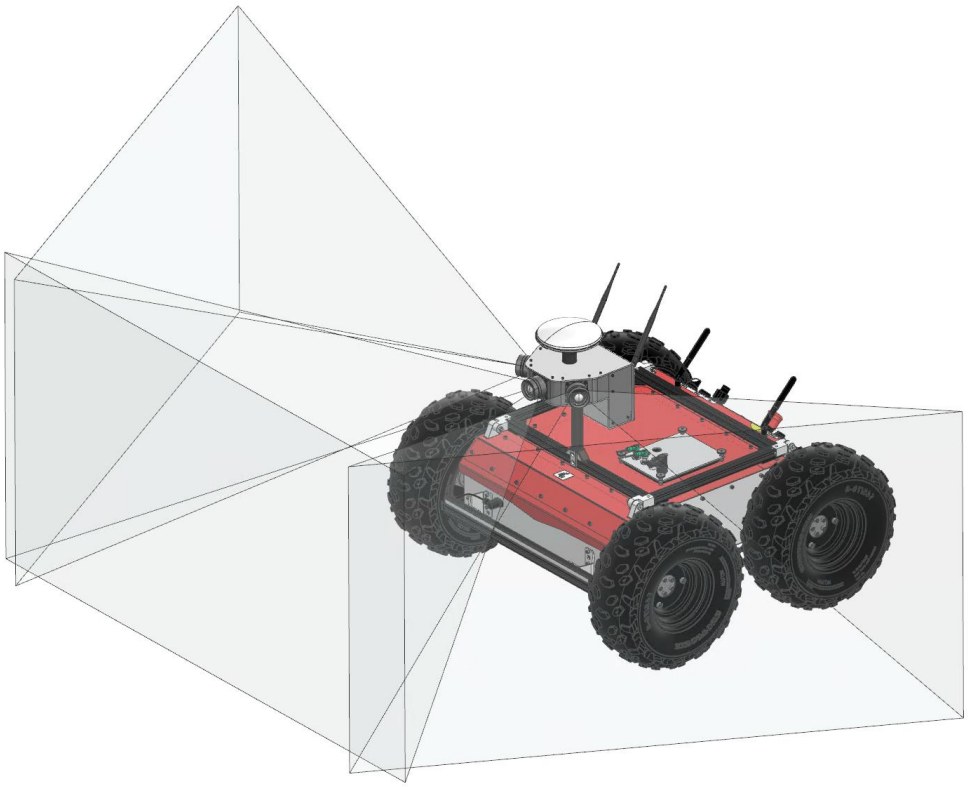
OST
Eastern Switzerland
University of Applied Sciences

Master Project 2

Design and Implementation of a Low-Cost GNSS-Synchronized Multi-Camera System for a UGV

Spring Semester 2026



Version: 1.0

Institute:	ISF Institute for Intelligent Systems and Smart Farming	**Student:**	Chris Stolz chris.stolz@ost.ch
Examinator:	Prof. Dr. Dejan Šeatović dejan.seatovic@ost.ch	**Matriculation Nr.:**	17-179-359
Supervisor:	Luis Meier luis.meier@ost.ch	**Degree Program:**	MSE Mechatronics & Automation

7 January 2026

OST
Eastern Switzerland
University of Applied Sciences

Contents

Abstract

Definition of Task:
This project presents the design, implementation, and evaluation of a low-cost synchronized multi-camera perception system for a mobile robotic application. The objective of the project was to develop a modular embedded sensor system capable of providing temporally coordinated image acquisition and GNSS-referenced timing using commercially available hardware components and a ROS2-based software architecture.

Fig. 1: UGV with Integrated Sensor Suite

Approach and Technology:
The developed system integrates three FLIR Firefly USB3 global-shutter RGB cameras, a GNSS/RTK receiver, and a Raspberry Pi 5 into a compact sensor enclosure mounted on a Husarion Panther unmanned ground vehicle. A custom ROS2-based acquisition framework implemented in C++ using the Teledyne FLIR Spinnaker SDK was developed to support deterministic camera management, synchronized frame grouping, calibration integration, and distributed sensor data publication. The project further includes the development of a complete intrinsic and extrinsic calibration pipeline based on OpenCV and ChArUco calibration targets.

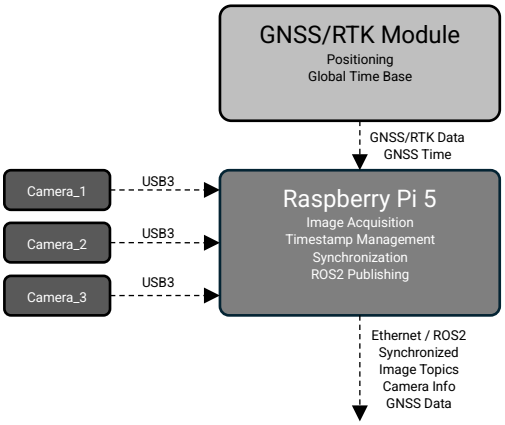


Fig. 2: High-Level System Architecture Overview

Results:
Experimental validation demonstrated sub-pixel intrinsic calibration accuracy together with physically consistent extrinsic calibration results for the evaluated camera configuration. The synchronization behaviour and temporal stability of the acquisition framework were evaluated under varying resolutions, frame rates, pixel formats, and recording workloads. The results show that stable synchronized acquisition at 30 Hz using full-resolution BayerBG8 image streams is achievable under nominal operating conditions with nearly no timing jitter and zero missing synchronized frame sets.

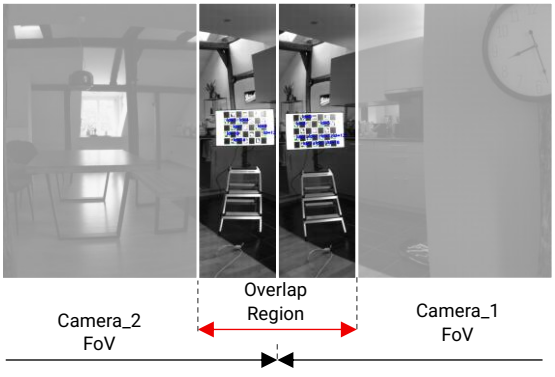


Fig. 3: Visualization of FoV Overlap

Although the current implementation relies on software-coordinated synchronization, the hardware and software architecture were intentionally designed to support future GNSS-triggered camera acquisition using a shared timing signal. Overall, the presented system establishes a reproducible and extensible foundation for future synchronized multi-camera perception and embedded robotic sensor integration.

Introduction

Problem Statement

Autonomous mobile robots require precise and reliable environmental perception to operate safely and robustly in complex and unstructured environments. The majority of robotic functionalities, such as localization, mapping, obstacle avoidance, 3D-point cloud colorization (Fig. 4) and sensor fusion, depend on spatially and temporally consistent sensor data. Inaccuracies in sensor alignment or timing can directly degrade downstream perception performance and lead to inconsistent environmental representations, resulting in mission failures, customer dissatisfaction, and ultimately safety risks when operating in environments where robots interact with humans.

Additionally, robots require sufficient computing capabilities, and the communication networks must provide adequate bandwidth to ensure that these spatial and temporal requirements can be fulfilled.

Fig. 4: Colourized Point Cloud [8]

Cutting-edge industrial-grade robotic perception systems typically address these requirements using industrial sensor suites with dedicated synchronization hardware, deterministic communication interfaces, powerful computing hardware and factory-calibrated sensor configurations. One of the most commonly known and widely applied approaches is the use of IEEE 1588 Precision Time Protocol (PTP) or hardware-triggered Ethernet cameras. While these solutions provide high synchronization accuracy and robust system integration, they significantly increase system cost, hardware complexity, and integration effort.

In contrast, low-cost and modular robotic platforms based on single-board computers and USB cameras face several practical limitations. USB camera systems usually do not provide deterministic timing behaviour or hardware-based synchronization. In addition, multiple high-bandwidth image streams must be processed on computationally limited embedded hardware, which can lead to acquisition delays, bandwidth bottlenecks, and inconsistent frame timing. Multi-camera calibration also becomes more difficult when compact mechanical designs reduce the overlap between neighbouring camera fields of view.

This project investigates whether synchronized multi-camera perception can be achieved on a low-cost embedded platform without relying on an industrial synchronization infrastructure.

The central engineering challenge is to design and implement a system that ensures:

- Temporally consistent image acquisition across multiple USB cameras using a shared time reference
- Reliable intrinsic and extrinsic calibration of the multi-camera setup despite constrained field-of-view overlap
- Robust integration into a ROS2-based distributed software architecture operating on resource-constrained embedded hardware

Objectives

The specific objectives of this project are:

- **System Architecture Design**
  Design and justify a modular hardware and software architecture for synchronized multi-camera perception on an embedded platform, considering trade-offs between cost, synchronization capability, bandwidth utilization, and system extensibility.
- **Temporal Synchronization and Validation**
  Implement and evaluate a GNSS-referenced synchronization strategy for multi-camera image acquisition, analyse temporal consistency under defined operating conditions, and investigate limitations of software-coordinated synchronization compared to hardware-triggered acquisition approaches.
- **Multi-Camera Calibration and Spatial Consistency**
  Develop and validate a calibration pipeline for intrinsic and extrinsic camera calibration under constrained field-of-view overlap conditions, and evaluate the resulting geometric consistency and spatial alignment of the perception system.
- **Reproducibility and Modularity**
  Provide a modular, extensible, and well-documented system architecture that supports reproducibility, future hardware-triggered synchronization integration, and further research-oriented system extensions.

Project Scope

The scope of the work is intentionally limited to the perception infrastructure layer, with particular emphasis on synchronization, calibration, system integration, and embedded multi-camera acquisition.

The implemented scope includes the following aspects:

- **Hardware Integration**
  Design and integration of a three-camera sensor suite, including mechanical mounting, enclosure design, power distribution, communication interfaces, and embedded platform integration within the robotic system.
- **Time Synchronization Implementation**
  Integration of a GNSS-referenced system time architecture and preparation of a hardware-triggered synchronization path for deterministic multi-camera acquisition. The current implementation focuses on software-coordinated synchronization using a shared time reference, while the system architecture remains extensible toward future hardware-triggered operation.
- **Calibration Pipeline**
  Implementation and validation of intrinsic camera calibration, as well as preparation and evaluation of extrinsic calibration procedures under constrained field-of-view overlap conditions. Particular focus is placed on the practical challenges and limitations of calibration in compact multi-camera configurations.
- **ROS2 Data Pipeline**
  Implementation of the software infrastructure required for synchronized image acquisition, timestamp management camera data publishing, and coordinate frame handling within a distributed ROS2 (Jazzy) environment.

- **System Validation**
  Evaluation of synchronization consistency, calibration quality, frame acquisition behaviour, bandwidth limitations, and overall system robustness under realistic operating conditions on an embedded computing platform.

The following aspects are explicitly excluded:

- **Advanced Perception Algorithms**
  The development of higher-level perception algorithms such as SLAM, object detection, visual odometry, or semantic scene understanding is intentionally excluded. This separation allows the work to focus specifically on the underlying perception infrastructure and sensor consistency required by such downstream applications.
- **Real-Time Sensor Fusion**
  Although the developed system is intended to support future sensor fusion applications, the implementation of full multi-sensor fusion pipelines is not part of this work. The focus remains on providing synchronized and geometrically consistent sensor data as a reliable foundation for future integration.
- **Full Colourized Point Cloud Generation**
  The generation of complete colourized 3D environmental representations is excluded, as this would require additional perception and fusion layers beyond the scope of the sensor infrastructure investigated in this project.
- **Custom Firmware or Low-Level Hardware Development**
  The project is based on commercially available hardware components and standard robotics software frameworks. While custom ROS2 software components and synchronization logic are developed as part of the system integration, low-level firmware modifications, kernel-level driver development, and custom embedded hardware design are intentionally excluded. The focus of this work remains on system-level architecture, synchronization, and perception infrastructure.
- **Optimization Beyond Proof-of-Concept**
  The work focuses on functional validation and architectural feasibility rather than full real-time optimization for production deployment. While system bottlenecks and performance limitations are analysed, extensive optimization for maximum throughput or industrial deployment is outside the scope of this project.

Documentation Structure

The documentation of this project is organized into three complementary components that together provide conceptual, implementation-level, and experimental traceability of the developed perception system.

- **Main Documentation**
  This document focuses on the system-level design, engineering methodology, implementation strategy, and experimental evaluation of the developed multi-camera perception platform. The document further provides the engineering rationale behind design decisions, discusses identified system limitations, and evaluates the resulting performance characteristics of the developed platform.

- **Project Repository**
  The complete software implementation is maintained within a version-controlled project repository. The repository contains the ROS2 packages, custom C++ acquisition nodes, launch configurations, calibration files, synchronization utilities, and supporting scripts developed during the project. Version control through Git enables reproducibility, implementation traceability, and structured development tracking throughout the project lifecycle.

  The repository further serves as the primary reference for implementation-specific details that extend beyond the scope of the main document.

The calibration and validation scripts used throughout the project are included within the repository under:

- tools/calibration
- tools/validation

These utilities include the intrinsic and extrinsic calibration workflows, synchronization validation tools, and supporting evaluation scripts used during experimental verification.

GitHub Link: https://github.com/CStolz95/husarion_ugv_sensor_suite

- **Supplementary Materials**
  Additional supporting materials are provided separately, including hardware datasheets, CAD models, MBOM, calibration datasets, validation data sets, measurement results, and experimental raw data. The datasets themselves are intentionally maintained separately from the public repository due to their size and archival purpose. However, all scripts and processing utilities required to reproduce the presented calibration procedures and validation results are included within the repository. These materials complement the thesis by providing detailed technical references and validation artifacts required for reproducibility and future system extension.

Cross-references throughout the document guide the reader to relevant implementation resources and supporting materials where appropriate.

Background

Robotic perception is a fundamental component of autonomous mobile systems, as it enables robots to interpret their surroundings and make decisions based on environmental information. Applications such as SLAM are used to create environmental maps, as described in the introduction section.

Traditionally, such maps are generated by combining heterogeneous sensors, most commonly LiDAR and cameras. However, multi-camera systems provide an attractive alternative or complementary approach when wide field of view, high-resolution visual coverage, and low-cost integration are required. By distributing several cameras across the platform, the system can observe a larger portion of the environment than a single conventional camera. A major advantage of RGB cameras compared to LiDAR sensors, particularly spinning LiDAR systems, is the significantly higher spatial measurement density. LiDAR point clouds are inherently sparse, with point density depending on the sensor specifications and measurement range. As the distance to the sensor increases, the spacing between individual points can reach several centimetres, which may limit the representation of fine structural details. In contrast, RGB cameras provide dense pixel-wise visual information across the entire image plane.

This benefit comes with additional system-level challenges. Multi-camera perception requires accurate intrinsic and extrinsic calibration to ensure geometric consistency between views. It also requires temporal synchronization to ensure that image data from different cameras corresponds to the same physical moment, especially when the robot or the environment is moving. These requirements become more difficult to satisfy on low-cost embedded platforms, where bandwidth, processing resources, and hardware synchronization options are limited.

This chapter introduces the technical background required to understand these challenges. It first discusses multi-sensor perception and multi-camera vision, followed by GNSS/RTK timing, synchronization methods, calibration principles, ROS2 integration, and the specific limitations of low-cost multi-camera systems.

Multi-Sensor Perception in Robotics

Modern robotic systems commonly rely on multiple complementary sensors to achieve robust and reliable perception of their environment. Multiple complementary sensors are required because of their differing characteristics and physical sensing capabilities, as well as the constantly changing environmental conditions. As a result, robotic platforms typically combine sensors such as LiDAR, RGB cameras, inertial measurement units (IMUs), and GNSS receivers to obtain both geometric and semantic information about their surroundings.

Fig. 5. ANYmal X (adapted from [14])

Each sensor system provides different strengths and limitations. LiDAR sensors deliver accurate geometric distance measurements and are largely independent of ambient lighting conditions, making them well suited for spatial mapping. In contrast, RGB cameras provide dense visual information such as colour, texture, and object appearance, which is essential for tasks such as scene understanding, object recognition, and visual localization. IMUs provide short-term motion information with high update rates, while GNSS systems contribute global positioning and timing information, particularly in outdoor environments

A representative example of such a multi-sensor architecture is the ANYmal inspection robot developed by ANYbotics (Fig. 5). The platform integrates LiDAR sensors, RGB cameras, depth cameras, and inertial measurement unit to achieve reliable perception in complex and unstructured environments.

Sensor fusion aims to combine these heterogeneous data sources into a unified and temporally consistent representation of the environment. This improves perception robustness, environmental understanding, and localization accuracy, particularly in dynamic real-world scenarios [12]. However, effective sensor fusion introduces significant system-level requirements.

First, all sensors must be spatially calibrated. For vision sensor this includes both intrinsic calibration, which determines sensor-specific parameters such as camera focal length, principal point, and lens distortion, and extrinsic calibration, which defines the relative positions and orientations of the sensors within a shared coordinate system. Even small calibration errors can introduce inconsistencies between sensor modalities and degrade downstream tasks such as visual odometry, mapping, feature matching, or point cloud colouring.

Second, sensor data must be temporally synchronized. Measurements acquired at different times may no longer correspond to the same physical scene, especially when either the robot or the environment is moving. Even millisecond-scale timing offsets between sensors can lead to spatial misalignment, distorted environmental representations, or reduced accuracy in multi-sensor fusion pipelines [15].

Multi-Camera Systems and Panoramic Vision

Multi-camera systems are widely used to extend the field of view beyond the limitations of a single conventional RGB camera. By arranging multiple cameras with partially overlapping fields of view (overlap region), a larger portion of the environment can be observed simultaneously from different perspectives. This is particularly advantageous in applications such as autonomous navigation, surveillance, and mobile robotics, where wide spatial coverage and environmental awareness are required.

Compared to alternative approaches such as fisheye or omnidirectional cameras, multi-camera systems provide several important advantages. First, each camera operates within a smaller angular field of view, resulting in reduced optical distortion and improved effective image resolution compared to ultra-wide-angle optics. This improves feature quality and geometric accuracy, which are critical for downstream tasks such as 3D reconstruction and visual odometry.

Second, multi-camera systems provide greater flexibility in system design. Individual cameras can be positioned and oriented according to the specific perception requirements of the robotic platform, allowing the sensing geometry to be optimized for coverage, overlap, or directional focus. However, this flexibility also increases system complexity, particularly with respect to calibration, synchronization, mechanical alignment, and data management.

A fundamental requirement for multi-camera perception is sufficient overlap between adjacent camera views. Overlapping regions are required to establish geometric correspondences during the extrinsic calibration process and to ensure consistent feature tracking between cameras. If the overlap region is too small, reliable feature matching and calibration become difficult or unstable, while excessive overlap reduces the total observable coverage and increases redundant image information.

As a result, the camera arrangement must balance panoramic coverage against calibration robustness and perception redundancy while minimizing blind zones within the observable environment. Poorly positioned cameras or insufficient overlap can produce regions with limited or no visual coverage, which may negatively affect feature tracking, obstacle detection, and environmental perception. Fig. 6 illustrates the developed multi-camera system, where overlapping fields of view (horizontal) are highlighted in red and the blind zones in front of the robot are highlighted in dark grey.

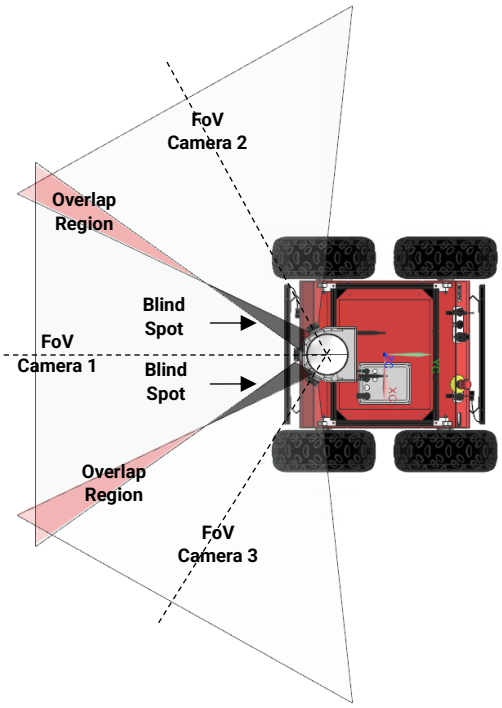


Fig. 6. Multi-Camera System Configuration With Three RGB Cameras and Partially Overlapping Fields of View

Another challenge in multi-camera systems is the acquisition and processing pipeline. Multiple high-resolution image streams create significant bandwidth and processing demands, especially on embedded platforms where cameras share communication and controller resources.

Consequently, system design requires careful balancing between image resolution, frame rate, compression strategy, and the number of simultaneously active cameras. Increasing image resolution improves geometric detail and feature quality, while higher frame rates improve temporal consistency during robot motion. However, both directly increase bandwidth utilization and processing load, potentially leading to frame drops, increased latency, or unstable acquisition behaviour on resource-constrained systems.

Previous work has shown that calibration inaccuracies and synchronization errors can significantly degrade the performance of multi-camera perception pipelines [13]. Accurate intrinsic calibration is required to compensate for lens distortion and projection errors, while precise extrinsic calibration ensures geometric consistency between viewpoints.

GNSS/RTK

Global Navigation Satellite Systems (GNSS) provide positioning and timing information on a global scale. In robotic systems, GNSS is used for localization in outdoor environments.

Standard GNSS positioning typically provides meter-level accuracy, which may not be sufficient for applications that require precise localization. To improve accuracy, Real-Time Kinematic (RTK) techniques are used. RTK enhances GNSS measurements by incorporating correction data from a base station, allowing position accuracy to reach the centimetre level under favourable conditions. This makes RTK suitable for applications such as mapping, surveying, and autonomous navigation. Fig. 7 illustrates the principle of RTK-based positioning, where correction data from a base station is used to improve the positioning accuracy of the rover.

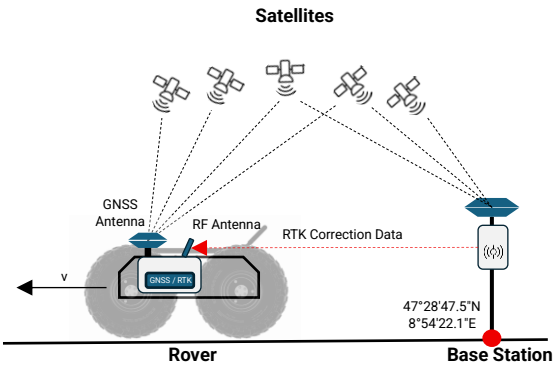


Fig. 7: Principle of RTK-Based GNSS Positioning

Beyond positioning, GNSS receivers offer precise timing signals, most commonly in the form of a Pulse Per Second (PPS) output. The PPS signal provides a periodic electrical pulse aligned with GNSS time, typically with sub-microsecond accuracy. This signal can be used to synchronize system clocks or to provide a common timing reference for multiple sensors. Compared to software-based synchronization methods, GNSS-based timing offers high stability and independence from network conditions.

However, GNSS-based timing depends on reliable satellite reception. In environments with poor or unavailable GNSS coverage, such as indoor areas, urban canyons, or tunnels, the PPS signal may become unstable or unavailable entirely, leaving the system without an external absolute time reference unless a fallback synchronization mechanism is provided.

Synchronization

As we already discussed, accurate time synchronization is a fundamental requirement in multi-sensor robotic perception systems. In multi-camera systems, synchronization errors can lead to visible misalignment between overlapping image regions, degraded feature correspondence, and reduced accuracy in downstream tasks such as visual odometry, image stitching, and 3D reconstruction. The severity of these effects increases with robot velocity, scene dynamics, and camera baseline distance.

Several synchronization approaches are commonly used in robotic and distributed sensing systems. These methods differ significantly with respect to timing accuracy, determinism, infrastructure requirements, and implementation complexity. A comparison of the synchronization methods most relevant to this work is shown in Table 1.

Characteristics	Software-Based Synchronization	IEEE 1588 PTP	GNSS PPS / TPS / Hardware Trigger
Principle	Acquisition coordinated through software timestamps and system time	Network-based synchronization of distributed system clocks	Shared hardware trigger signal derived from GNSS PPS / TPS
Typical Accuracy	Millisecond-level	Microsecond-level or better	Microsecond-level or better
Determinism	Low	Medium	High
Infrastructure Requirements	No additional hardware required	PTP-capable network hardware and supported devices	GNSS receiver, trigger wiring, hardware trigger support
Advantages	Simple integration, low cost, flexible architecture	High clock synchronization accuracy, scalable for distributed systems	Deterministic synchronized acquisition, globally referenced timing
Limitations	Affected by OS scheduling, USB, latency, and buffering	Increased network complexity, does not guarantee simultaneous sensor exposure	Additional hardware integration effort and trigger distribution complexity
Suitability of Low-Cost Embedded Systems	High	Medium	Medium
Synchronization of Sensor Exposure	No	Indirect only	Yes
Dependency on Network Infrastructure	None	High	None

Table 1. Comparison of Synchronization Approaches for Multi-Camera Systems

A widely used synchronization method is the IEEE 1588 Precision Time Protocol (PTP), which synchronizes distributed system clocks over a communication network. In contrast to hardware triggering, PTP synchronizes clocks rather than the physical exposure events of the image sensors. As a result, acquisition timing may still be influenced by internal sensor latency, buffering behaviour, or operating system scheduling.

An alternative approach is GNSS-based timing, where a GNSS receiver provides a globally referenced Pulse Per Second (PPS) and/or Time Pulse Signal (TPS). These signals can discipline system clocks or directly trigger synchronized sensor acquisition using a shared hardware trigger. Since image acquisition is initiated directly at the sensor level, this approach provides highly deterministic timing with minimal jitter. Unlike PTP, GNSS-based synchronization does not depend on network infrastructure, making it well suited for distributed or mobile robotic platforms operating outdoors with reliable GNSS reception. However, the achievable synchronization accuracy still depends on factors such as trigger signal distribution, cable propagation delays, internal camera trigger latency, and sensor exposure behaviour. As a result, GNSS-triggered synchronization is widely used in industrial vision systems and other high-precision multi-camera.

As shown in Table 1, software-based synchronization provides low integration complexity and high flexibility but suffers from non-deterministic timing behaviour. Hardware-triggered synchronization offers the highest level of temporal consistency but requires additional hardware integration and trigger infrastructure. PTP represents a compromise between these approaches but introduces network and hardware requirements that are often difficult to satisfy in low-cost embedded systems.

Calibration

Camera calibration is a fundamental requirement for accurate visual perception in robotic systems. It establishes the mathematical relationship between the three-dimensional (3D) environment and the two-dimensional (2D) image observations captured by a camera. In multi-camera systems, calibration is particularly important because all cameras must operate within a geometrically consistent shared coordinate system.

Accurate calibration is essential for reliable downstream perception. Calibration errors can result in inaccurate image projections, inconsistent feature matching, and geometric misalignment between different camera views. These issues become particularly severe in multi-camera systems and when the robotic platform is in motion.

Camera calibration is typically divided into two stages: intrinsic and extrinsic calibration.

- Intrinsic calibration determines the internal optical parameters of an individual camera, including focal length, principal point, and lens distortion coefficients. These parameters describe how 3D scene points are projected onto the image plane and are required to compensate for optical distortion and perspective-related projection errors. Accurate intrinsic calibration is particularly important when using wide-angle lenses, where distortion effects become more pronounced toward the image boundaries.
- Extrinsic calibration determines the relative position and orientation between multiple cameras or between cameras and other sensors. It defines the rigid-body transformation between coordinate frames and enables sensor data from different viewpoints to be represented consistently within a shared spatial reference frame. In multi-camera systems, accurate extrinsic calibration is essential for maintaining geometric consistency across overlapping image regions.

Several established calibration approaches exist in robotic vision systems. A widely used intrinsic calibration method is the approach proposed by Zhang [12], which uses planar calibration targets such as checkerboards to estimate camera parameters from multiple viewpoints. The method is widely adopted due to its robustness, relatively simple setup requirements, and compatibility with standard computer vision frameworks.

For multi-camera systems, calibration frameworks such as Kalibr [15] support joint estimation of spatial and temporal sensor parameters across multiple devices. In addition, marker systems such as AprilTag [16] improve feature detection robustness under varying illumination conditions, partial occlusions, or challenging viewing angles, making them well suited for practical robotic calibration tasks.

Despite the availability of mature calibration methods, reliable multi-camera calibration remains challenging in practical embedded robotic systems. A key requirement is sufficient overlap between adjacent camera views to observe common calibration features simultaneously. Limited overlap, large camera baselines, or restricted mounting geometries can significantly reduce the stability and accuracy of extrinsic parameter estimation.

Calibration quality is also strongly influenced by image quality and acquisition consistency. Motion blur, image noise, poor illumination, or inconsistent feature detection directly reduce calibration robustness and increase reprojection error.

ROS2

ROS2 is used as the central software integration framework for the developed multi-camera perception system. It provides a modular software architecture in which individual functions, such as camera image acquisition, GNSS data handling, timestamp management, and data publishing, can be implemented as separate nodes with clearly defined interfaces.

This modularity is important for the proposed system because the perception pipeline combines multiple sensors, high-bandwidth image streams, and timing information on a resource-constrained embedded platform. By using standardized ROS2 message types, topics, and launch configurations, the system remains extensible and can be integrated within the existing ROS2 software stack of the Husarion Panther UGV.

A key aspect of ROS2 is its use of the Data Distribution Service (DDS) as the communication middleware. DDS enables distributed communication between nodes and devices and supports configurable Quality of Service policies. This is particularly relevant for camera-based perception, where large image streams may require different reliability and latency settings than lower-bandwidth sensor data such as GNSS messages.

ROS2 also provides the time and transform infrastructure required for synchronized perception. Timestamped messages allow sensor data to be aligned temporally, while the TF framework defines

the spatial relationships between the robot, cameras, and other sensors. These mechanisms are essential for combining multi-camera data within a consistent spatial and temporal reference frame.

Challenges in Low-Cost Multi-Camera Systems

While the fundamental principles of synchronization and calibration are well established, their implementation on low-cost embedded platforms introduces significant practical constraints. In contrast to industrial perception systems with dedicated synchronization hardware and high-performance computing resources, low-cost robotic platforms must balance timing accuracy, calibration robustness, bandwidth limitations, processing load, mechanical stability, and system complexity simultaneously.

These constraints are not independent but strongly coupled at the system level. For example, bandwidth limitations may require reduced frame rates or image resolution, which can affect feature quality and synchronization consistency. Similarly, mechanical instability or synchronization errors can directly degrade calibration accuracy and downstream perception performance.

The combination of these challenges highlights the gap between theoretical multi-camera perception methods and their practical implementation on low-cost embedded robotic platforms. Addressing these limitations while maintaining modularity, reproducibility, and cost efficiency forms the central engineering objective of this work.

System Requirements & Architecture

System Overview

This section provides a high-level overview of the developed multi-camera perception system. The system is designed as a compact and modular sensor suite for a mobile ground robot.

Main Components:
The main hardware components of the system are:

- a Raspberry Pi 5 as central processing unit,
- three USB3 machine vision cameras,
- a GNSS/RTK receiver providing positioning and time reference,
- and supporting components such as a USB hub and power distribution.

A detailed overview of the physical system integration and interconnections is shown in Fig. 15 on page 23.

Requirements

The system requirements define the functional capabilities, performance objectives, environmental conditions, and design constraints that guided the development of the synchronized multi-camera perception platform. A complete and formal specification is documented separately in the System Requirements Document (SRD).

Functional Requirements
The primary functional requirements are:

- Acquisition of synchronized image data from three RGB cameras
- Integration of GNSS/RTK positioning and timing information
- Provision of globally referenced timestamps derived from GNSS PPS synchronization
- Publishing of synchronized sensor data within a ROS2-based middleware architecture
- Support for intrinsic and extrinsic multi-camera calibration
- Provision of synchronized image sets for downstream perception and sensor fusion pipelines
- Support for Ethernet-based integration into distributed robotic systems
- Recording of synchronized sensor datasets, calibration files, and configuration data for reproducibility and offline analysis
- Modular and maintainable sensor subsystem integration for simplified testing and debugging

Performance Requirements
Several performance-related requirements significantly influenced the system architecture and synchronization design:

- Inter-camera synchronization accuracy target of ≤ 100 µs
- Global system time alignment to GNSS reference within ≤ 10 µs to PPS
- Continuous image acquisition at ≥ 30 FPS per camera
- End-to-end image transport latency ≤ 100 ms
- Frame drop rate ≤ 0.1 % under nominal operating conditions
- Deterministic synchronization behaviour during continuous acquisition
- Stable operation under shared USB bandwidth and embedded processing limitations

Environmental Requirements
The system is intended for limited outdoor operations. The environmental requirements include:

- Continuous operation within a temperature range of −5 °C to 25 °C
- Resistance against mechanical shock and vibration during field operation
- Ingress protection level of IP42 or higher for the enclosure and exposed interfaces
- Stable mechanical sensor mounting to minimize calibration drift during operation

System Constraints
The final system architecture was additionally shaped by practical technical and project-related constraints:

- Overall project budget limited to 2000 CHF
- Use of commercially available low-cost hardware components
- Limited USB bandwidth for simultaneous multi-camera image transport
- Limited computational resources of the embedded Raspberry Pi 5 platform
- Physical mounting constraints on the robotic platform
- Required overlap between adjacent camera fields of view for calibration and feature association needs to be ~10°
- Temporary unavailability of dedicated hardware trigger cabling during development
- Manufacturing constraints favouring 3D printing and sheet-metal fabrication

System Architecture

System Overview

This section provides a high-level overview of the developed multi-camera perception system and introduces the main architectural components and data flows.

The perception system is implemented around a Raspberry Pi 5 running Ubuntu 24.04 and ROS2 Jazzy. Three USB3 global-shutter cameras provide synchronized image acquisition, while a GNSS/RTK receiver supplies both positioning information and a globally referenced timing source. The use of global-shutter image sensors is particularly important for synchronized multi-camera perception, as it prevents motion-induced image distortion during robot movement. A high-level overview of the system architecture is shown in Fig. 8.

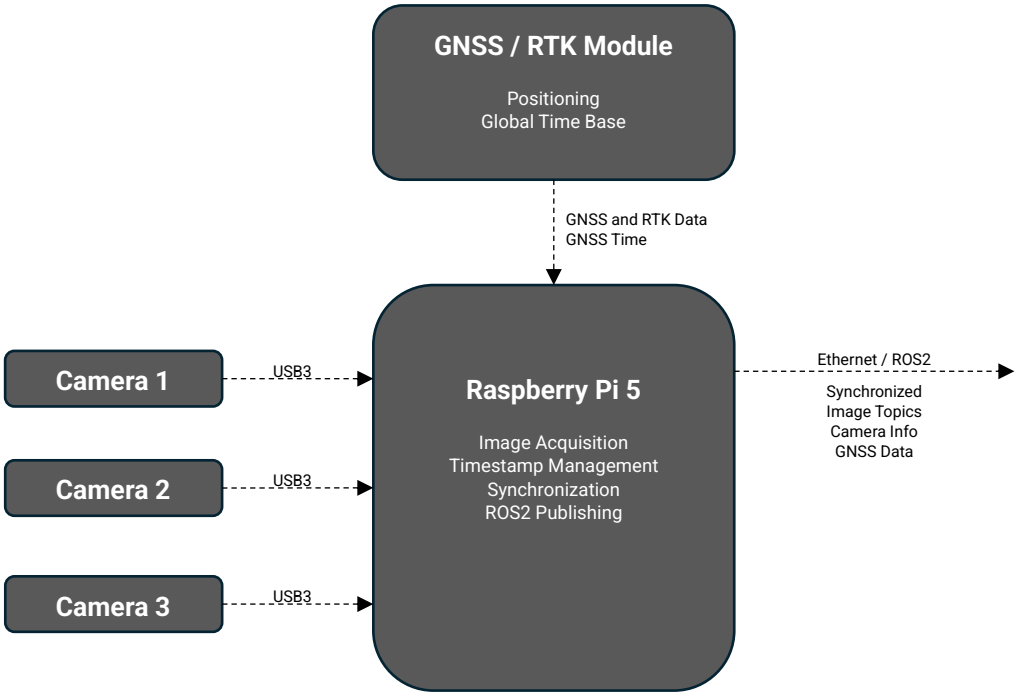


Fig. 8. High-Level System Architecture Overview

From a data-flow perspective, the cameras continuously send image streams to the embedded processing unit via USB3. The Raspberry Pi mainly acts as a synchronized sensor interface and as a bridge between the sensor suite and the higher-level robotic software stack. Its main tasks include coordinated multi-camera acquisition, timestamp management, integration of calibration data, and publishing synchronized image streams together with camera information and GNSS/RTK data through ROS2.

Instead of running computationally intensive perception algorithms locally, the embedded platform focuses on reliable sensor communication and temporally consistent data distribution. This separation of tasks reduces the computational load on the embedded platform and allows downstream applications such as SLAM, mapping, and sensor fusion to run on more powerful computing hardware within the robotic system.

In parallel, the GNSS receiver provides positioning data and GNSS-referenced timing information, establishing a consistent global time base across all sensor data. The architecture is designed to support both the current software-coordinated synchronization strategy and future hardware-triggered image acquisition. This enables the system to remain modular and extensible while allowing a transition toward deterministic synchronization with minimal architectural modifications.

A key design objective of the system is to balance synchronization accuracy, bandwidth limitations, processing performance, modularity, and overall system cost. These considerations directly influenced

the hardware selection, synchronization strategy, and software architecture presented in the following sections.

Hardware Architecture

Hardware Design Considerations

The hardware architecture of the developed perception system is mainly driven by the requirement to achieve synchronized multi-camera perception on a low-cost embedded robotic platform while maintaining robustness, modularity, and extensibility.

Design Considerations
Several factors directly influenced the hardware design:

- **Bandwidth and Processing Constraints**
  Simultaneous acquisition of multiple high-resolution image streams generates substantial bandwidth and processing demands on the embedded platform. Since all cameras share common USB3 and processing resources, image resolution, frame rate, and data throughput must be carefully balanced to maintain stable acquisition behaviour. These limitations motivated the use of configurable camera operating parameters, enabling the system to adapt image quality and acquisition rate according to the available processing and communication resources

- **Mechanical Stability**
  Accurate multi-camera calibration requires stable relative camera poses over time. Even small mechanical shifts between cameras can alter the estimated extrinsic parameters and reduce geometric consistency between views. The hardware design therefore prioritizes rigid sensor mounting and mechanically stable integration of the camera assemblies in order to preserve calibration accuracy during mobile robotic operation.

- **Field-of-View and Overlap Constraints**
  The selected wide-angle camera configuration is designed to maximize environmental coverage while minimizing blind spots around the robotic platform. However, increasing panoramic coverage reduces the overlap between adjacent camera views, which can negatively affect feature matching and extrinsic calibration stability. The final camera arrangement therefore represents a compromise between wide-area perception coverage and sufficient overlap for reliable multi-camera calibration.

- **Synchronization Requirements**
  Temporal consistency between camera streams is essential for geometrically consistent multi-camera perception. The architecture therefore integrates a GNSS-referenced timing concept to establish a common global time base across all sensor data. At the same time, the system is designed to remain compatible with future hardware-triggered synchronization. The required trigger routing and synchronization interfaces are already integrated into the hardware architecture, enabling future deterministic image acquisition with minimal structural modifications.

- **Embedded Platform Constraints**
  The Raspberry Pi 5 provides a compact, low-cost, and ROS2-compatible embedded processing platform which is capable of managing the data acquisition and timestamp management.

  Manufacturability and Serviceability
  In addition to perception performance, the hardware architecture was designed to simplify assembly, maintenance, and troubleshooting. The sensor enclosure therefore follows a modular design approach that allows the complete sensor suite to operate independently from the robotic platform during development and testing. All critical service interfaces, including the Raspberry Pi USB-C connection, HDMI interface, and USB hub power connection, remain accessible through a removable side cover without requiring complete system

disassembly. This significantly simplifies debugging, software deployment, and hardware maintenance during system integration and experimental evaluation. The enclosure design also minimizes cable routing complexity by reducing internal cable lengths and limiting unnecessary interconnections between subsystems. Besides improving assembly simplicity and maintainability, this reduces potential electromagnetic interference (EMI/EMC) issues and improves overall integration robustness.

Design Trade-offs
The final hardware architecture reflects several engineering trade-offs between performance, complexity, scalability, manufacturability, and cost.

- **Cost vs. Performance**
  The use of low-cost commercial off-the-shelf components significantly reduces overall system cost and improves reproducibility. However, this introduces additional challenges related to synchronization precision, bandwidth limitations, and embedded processing constraints compared to industrial perception systems.

- **Manufacturing Cost. Vs Mechanical Robustness**
  To remain within the project budget constraints, the sensor enclosure was primarily manufactured using 3D-printed components and bent sheet-metal elements rather than precision-machined industrial components. This significantly reduced manufacturing cost and enabled rapid prototyping and iterative mechanical design modifications. However, this low-cost manufacturing approach introduces limitations in mechanical rigidity, long-term durability, and mounting precision compared to industrial-grade constructions. Since the accuracy of multi-camera calibration depends strongly on maintaining stable relative camera poses, reduced structural rigidity can increase sensitivity to vibration, mechanical stress, and long-term calibration drift. In addition, the use of 3D-printed parts results in a less robust system, as the plastic material is susceptible to degradation over time due to heat exposure and UV radiation. The final mechanical design therefore represents a compromise between manufacturing cost, assembly simplicity, structural robustness, and calibration stability.

- **Compactness vs. Calibration Quality**
  A compact sensor arrangement simplifies integration within the robotic platform and reduces mechanical complexity. At the same time, limited physical separation and reduced overlap between camera views increase the difficulty of stable extrinsic calibration.

- **Flexibility vs. Determinism**
  Software-based synchronization enables flexible system integration and compatibility with standard USB3 machine-vision cameras without requiring dedicated trigger infrastructure. However, deterministic hardware-triggered acquisition provides superior temporal consistency and lower synchronization jitter. The current architecture therefore prioritizes flexibility and modularity while remaining extensible toward future hardware-triggered synchronization.

Overall, the developed hardware architecture represents a practical compromise between synchronization capability, calibration robustness, embedded processing limitations, manufacturability, modularity, and overall system cost.

Sensor Subsystem

Multi-Camera Configuration

The visual perception subsystem consists of three Teledyne FLIR Firefly USB3 machine-vision cameras (FFY-U3-16S2) (Fig. 9) arranged in a synchronized multi-camera configuration. The cameras are based on the Sony IMX296 CMOS sensor with global-shutter and provide a resolution of 1440 × 1080 pixels. The selected machine-vision camera supports external triggering via a dedicated GPIO interface, enabling future deterministic hardware-triggered image acquisition.

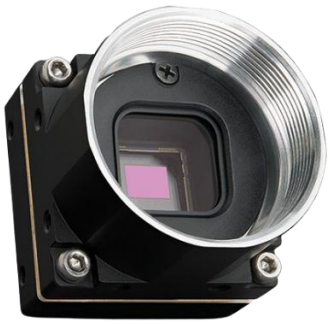


Fig. 9. Firefly S 1.6 MP Camera [1]

The preferred image format is 8-bit Bayer because it preserves raw colour sensor data while minimizing bandwidth usage. Demosaicing can then be performed by downstream perception applications. RGB8 output is avoided because it triples the required bandwidth without providing additional raw sensor information. At a resolution of 1440 × 1080 pixels and 30 FPS, the three-camera system requires approximately 140 MB/s of sustained USB bandwidth when using Bayer8 image formats. A USB3 hub can support this data rate, but all cameras share the same upstream USB3 connection to the Raspberry Pi. Therefore, sufficient bandwidth margin, hub quality, power delivery, and processing performance must be ensured.

Pixel Format	Per Camera (1440 x 1080 x 30 x bytes per pixel)	Per Camera (1440 x 1080 x 30 x bytes per pixel x 3)
8-bit mono/Bayer	~46.7 MB/s (~0.37 Gbit/s)	~140 MB/s (~1.12 Gbit/s)
16-bit mono/Bayer	~93.3 MB/s (~0.75 Gbit/s)	~280 MB/s (~2.24 Gbit/s)
RGB8, 3 byetes/pixel	~140 MB/s (~1.12 Gbit/s)	~420 MB/s (~3.36 Gbit/s)

Table 2. Estimated Required USB Bandwidth Depending on Image Format

Optical System

The camera subsystem is equipped with Kowa LM4NCL C-mount lenses (Fig. 11) featuring a focal length of 3.5 mm. The selected focal length provides a wide field of view, allowing the multi-camera arrangement to maximize environmental coverage while minimizing blind spots around the robotic platform.

Fig. 10. Anti-Reflection Window [10]

Fig. 11. C-Mount Industrial Lens [2]

Wide-angle optics are particularly beneficial for mobile robotic perception because they increase scene visibility and allow visual features to remain observable across multiple image frames. This improves feature tracking robustness and benefits downstream applications.

The lenses provide a maximum aperture of F1.8, allowing sufficient light collection under varying illumination conditions. In the implemented system, the aperture is manually adjusted and fixed prior to calibration and operation in order to maintain consistent imaging characteristics across all cameras. From a mechanical perspective, the selected lenses include locking mechanisms for focus and iris adjustment. This helps preserve stable optical parameters during robot operation and reduces the risk of calibration drift caused by vibration or unintended mechanical movement.

The optical assemblies are additionally protected by anti-reflective optical windows (Fig. 10) integrated into the sensor enclosure. Besides providing environmental protection against dust, moisture, and mechanical impact, the optical windows reduce unwanted reflections and imaging artifacts that could otherwise degrade image contrast and feature extraction reliability.

GNSS/RTK Integration

The sensor subsystem additionally integrates an ArduSimple simpleRTK3B Pro GNSS receiver (Fig. 12) based on the Septentrio mosaic-X5 chipset. The receiver supports multi-band and multi-constellation GNSS operation together with Real-Time Kinematic (RTK) positioning capability.

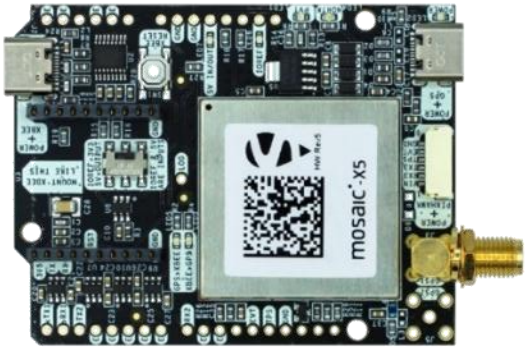


Fig. 12. simpleRTK3B Pro [11]

Within the developed architecture, the GNSS subsystem fulfils both temporal and spatial reference functions. In addition to providing positioning information for future localization and sensor fusion applications, the receiver supplies a globally referenced timing source used for synchronization of the perception system.

GNSS-derived time information is used to establish a common global time base across all sensor data streams. This improves temporal consistency between acquired measurements and enables synchronization independent of local system clock drift.

The receiver additionally provides a single configurable Time Pulse Signal (TPS/PPS) output intended for future hardware-triggered camera synchronization. Since only one hardware timing output is available, the system architecture prioritizes deterministic camera triggering over direct PPS-based Linux clock disciplining. The required trigger routing infrastructure is already integrated into the sensor suite, allowing future deterministic image acquisition with minimal hardware modifications.

Embedded Processing Subsystem

The embedded processing subsystem is centred around a Raspberry Pi 5 (Fig. 13) and serves as the primary acquisition, synchronization, and communication platform of the perception system.

Fig. 13. Raspberry Pi 5 [5]

The Raspberry Pi 5 was selected because it provides a compact, low-cost, and energy-efficient Linux-based platform with native support for USB3, Gigabit Ethernet, and ROS2. These interfaces are essential for simultaneous multi-camera acquisition, GNSS integration, and network-based data distribution within the robotic platform.

Within the developed system architecture, the Raspberry Pi receives image streams from the multi-camera subsystem together with GNSS timing and positioning data. The platform is responsible for coordinated sensor acquisition, timestamp handling, synchronization management, integration of calibration information, and ROS2 publication of synchronized sensor streams.

The separation between sensor-side acquisition and downstream perception processing provides several architectural advantages. First, it reduces computational load on the embedded platform, improving acquisition stability under high-bandwidth operation. Second, it allows advanced perception algorithms to execute on more powerful external computing hardware without modifying the sensor subsystem architecture. Finally, this modular separation simplifies debugging and system validation because the sensor suite can operate independently from the rest of the robotic software stack.

Despite these advantages, the use of a compact embedded platform also introduces limitations. Simultaneous acquisition of multiple USB3 image streams generates substantial memory transfer and processing load, particularly when operating at higher image resolutions or frame rates. These constraints directly influence operating parameters such as frame rate, image resolution, and synchronization strategy.

To support stable multi-camera operation, the cameras are connected through an externally powered USB3 hub (Fig. 14), which aggregates all image streams and forwards them to the Raspberry Pi via USB3. The use of an externally powered hub is particularly important because the combined power consumption of multiple machine-vision cameras exceeds the power that can be reliably supplied directly through the Raspberry Pi USB interfaces alone.

Fig. 14. USB Hub [7]

In addition to improving power stability, the USB hub simplifies cable management and reduces the number of direct connections required at the embedded platform. This improves maintainability and supports future scalability of the sensor subsystem.

Communication and Synchronization Infrastructure

A detailed overview of the physical interconnections power and signal paths within the system is shown in Fig. 15.

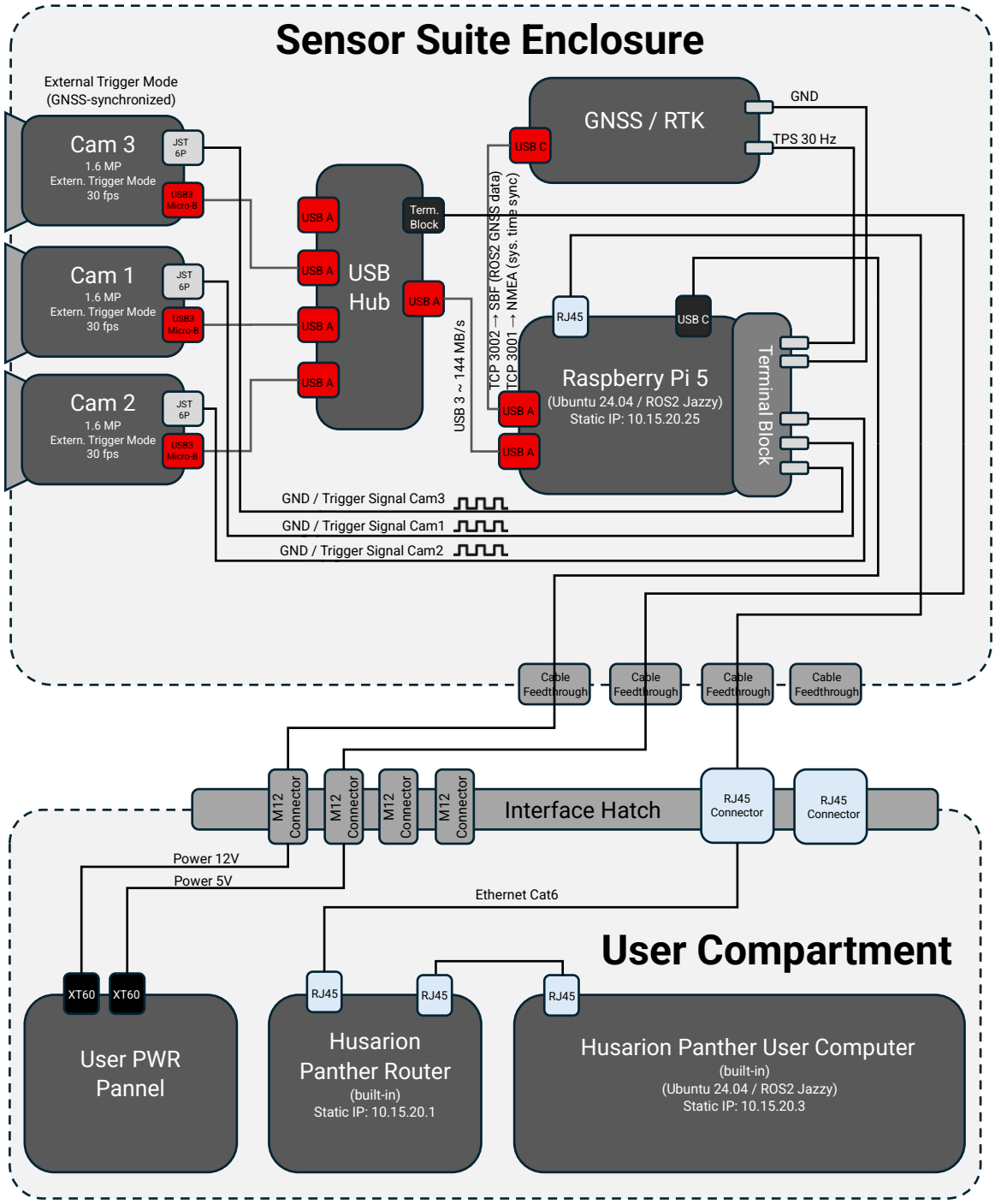


Fig. 15. Interconnection Diagram With Hardware Triggering Concept

USB Sensor Communication

Image data from the three machine-vision cameras is transmitted to the embedded processing subsystem via USB3 interfaces. All cameras are connected through an externally powered USB3 hub, which aggregates the individual image streams and forwards them to the Raspberry Pi through a single high-speed connection.

Cable routing within the sensor enclosure was additionally designed to minimize cable length and reduce wiring complexity. Besides simplifying assembly and maintenance, shorter communication paths reduce susceptibility to electromagnetic interference (EMI/EMC) and improve overall integration robustness.

Ethernet-Based Communication
The Raspberry Pi distributes synchronized sensor data to the robotic platform through a Gigabit Ethernet connection. This network interface enables transfer of synchronized camera streams, calibration information, and GNSS data to the more powerful user computer of the robotic platform for downstream perception processing.

The use of Ethernet-based communication additionally improves modularity and simplifies integration into existing ROS2-based robotic systems.

GNSS-Referenced Synchronization
The GNSS time is transmitted via USB rather than through a dedicated PPS signal cable, as shown in Fig. 15.

Trigger Signal Routing and Synchronization Infrastructure (Concept)
The hardware architecture as shown in Fig. 15. includes a dedicated trigger designed for deterministic camera synchronization. The GNSS-generated trigger signal is routed through the terminal block which is mounted on the Raspberry Pi 5 and distributed to the trigger inputs of all three cameras.

To minimize propagation differences between cameras, trigger cables need to be designed with equal cable lengths and minimized routing distances wherever possible. In addition, all cameras, the Raspberry Pi, and the GNSS subsystem must share a common electrical ground reference to ensure signal integrity and reduce synchronization inconsistencies caused by electrical potential differences.

At the current stage of development, the hardware-trigger routing infrastructure is fully prepared within the system architecture but not yet physically connected due to the unavailability of suitable trigger cabling. Nevertheless, the communication and synchronization architecture is already designed to support deterministic hardware-triggered image acquisition with minimal future modifications.

Mechanical and Electrical Integration

The mechanical and electrical integration of the perception system is designed to provide stable sensor operation, simplified maintenance, modular deployment, and reliable integration within the Husarion Panther robotic platform.

Particular emphasis was placed on maintaining stable camera alignment for calibration robustness while simultaneously minimizing manufacturing cost, assembly complexity, and cable routing overhead.

Mechanical Integration and Enclosure Design
The sensor subsystem is integrated within a dedicated enclosure mounted on the robotic platform (Fig. 16). Its elevated and symmetric position on the platform ensures an unobstructed field of view and improves overall environmental coverage. In addition, this placement allows the future integration of additional sensors, such as a LiDAR unit, in a central position below the sensor suite without introducing field-of-view obstructions.

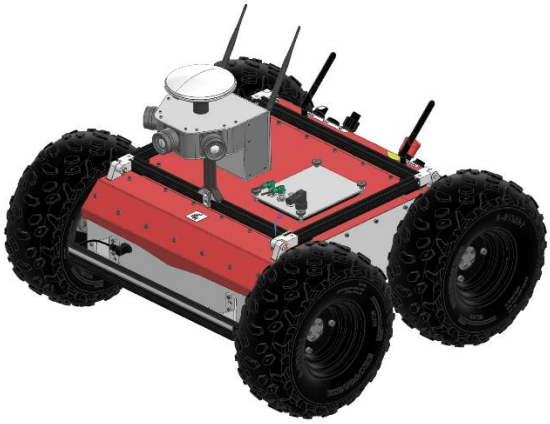


Fig. 16. Sensor Suite Mounted on the Robotics Platform

Modularity and Serviceability
A major design objective of the mechanical integration was to simplify assembly, maintenance, and troubleshooting throughout system development and operation.

The perception subsystem was therefore designed as a largely self-contained modular sensor unit that can operate independently from the remaining robotic platform during testing and debugging. This separation significantly simplifies subsystem validation and allows sensor-side problems to be isolated without requiring the complete robotic software stack to be operational.

To support maintainability, all critical service interfaces remain accessible through a removable side cover (Fig. 17) integrated into the enclosure.

These interfaces include:
- the Raspberry Pi USB-C interface,
- HDMI access,
- and the external USB hub power connection.

This arrangement enables debugging and hardware maintenance without requiring complete enclosure disassembly, thereby reducing maintenance effort and improving development efficiency.

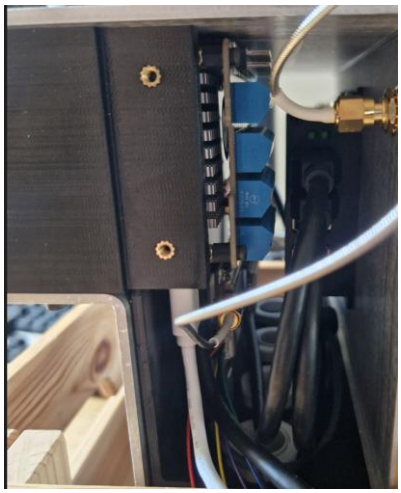


Fig. 17. Sensor Suite With Removed Side Covers

Electrical Integration and Power Distribution
The electrical integration is designed to ensure stable operation of all sensor and processing components while minimizing wiring complexity within the robotic platform.

The Raspberry Pi is supplied directly from the regulated 5 V power output of the robotic platform, while the USB3 hub is powered independently using a dedicated 12 V supply. Providing dedicated

power to the hub improves overall power stability and reduces the risk of voltage drops or unstable camera operation during simultaneous multi-camera acquisition.

The GNSS/RTK receiver is powered directly via USB from the Raspberry Pi because of its comparatively low power consumption, eliminating the need for additional dedicated power infrastructure.

Power and communication interfaces between the sensor enclosure and the robotic platform are routed through accessible external connectors integrated into the maintenance interface of the robot. This improves maintainability and enables simplified subsystem installation and replacement.

Cable Routing and Integration Robustness
Cable routing within the enclosure was designed to minimize cable length, reduce wiring complexity, and simplify system integration.

Reducing internal cable lengths improves assembly clarity and maintainability while also reducing susceptibility to electromagnetic interference (EMI/EMC) and signal integrity issues. This is particularly important in synchronized multi-camera systems where timing consistency and reliable high-bandwidth communication are essential.

In addition, minimizing cable complexity reduces mechanical strain on connectors and simplifies future system modifications or maintenance procedures.

Software Architecture

Software Architecture Overview

The software architecture of the developed perception system is designed to support synchronized multi-sensor acquisition and distributed sensor integration within a ROS2-based robotic framework.

Particular emphasis is placed on:

- temporally consistent multi-camera acquisition,
- deterministic sensor handling,
- scalable data distribution,
- and separation between sensor acquisition and downstream perception processing.

The software system follows a distributed node-based architecture, where individual functional components are separated into dedicated processing nodes. This architectural approach improves maintainability and enables flexible integration of additional sensors or perception components without requiring fundamental modifications to the existing system structure. A high-level overview of the software architecture is shown in Fig. 18.

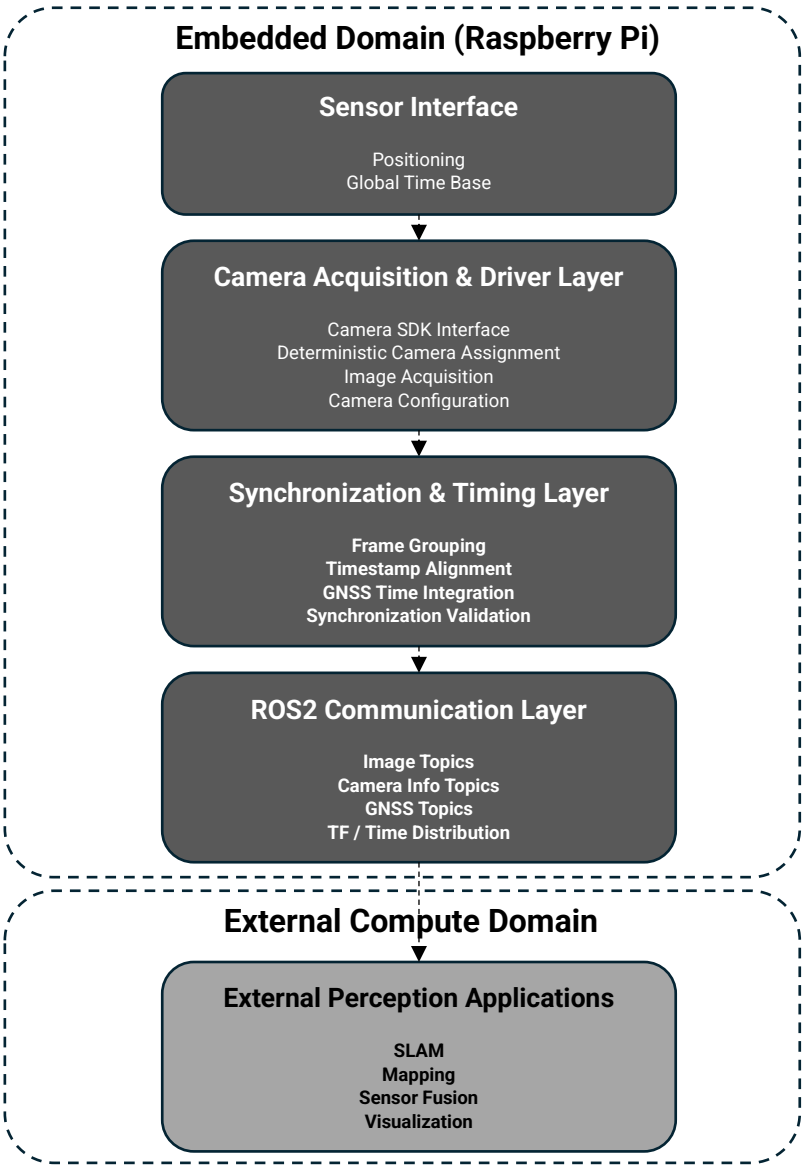


Fig. 18. High-Level Software Architecture

The software architecture consists of four main functional layers:

- a camera acquisition and driver layer responsible for image acquisition and camera management,
- a GNSS integration layer responsible for positioning and time synchronization,
- a synchronization layer responsible for temporal consistency across sensor streams,
- and a ROS2 communication layer responsible for distributed sensor data exchange.

Camera Acquisition and Driver Architecture

To achieve reliable and reproducible multi-camera operation, the system uses a dedicated custom ROS2 driver node that interfaces directly with the underlying camera SDK. This approach provides full control over camera configuration, acquisition timing, synchronization handling, and data publication while remaining compatible with standardized ROS2 interfaces.

A major design objective of the driver architecture is deterministic multi-camera management (Fig. 19). In multi-camera USB systems, device enumeration order can vary between system restarts, potentially leading to inconsistent camera assignment and incorrect spatial interpretation of image streams. To prevent this issue, cameras are identified and assigned deterministically using unique hardware identifiers (FLIR serial numbers). This ensures reproducible mapping between physical camera positions and software-defined camera instances, which is essential for stable calibration, coordinate-frame consistency, and repeatable system behaviour.

The driver architecture additionally implements coordinated multi-camera acquisition where each camera operates independently in its own acquisition thread and continuously captures images at the configured frame rate. The newest image from each camera is stored in a shared buffer together with a frame counter. A separate synchronization timer running at 30 Hz periodically checks whether all cameras have produced a new frame since the last publication cycle. The camera acquisition frame rate was intentionally configured equal to the synchronization timer frequency to ensure deterministic one-to-one correspondence between acquisition cycles and publication events. This minimizes unnecessary frame buffering, reduces CPU and USB bandwidth utilization, and simplifies the software synchronization logic by ensuring that ideally one fresh frame per camera is available during each synchronization cycle. If new frames are available from all cameras, the grouped image set is published using a common ROS2 timestamp representing the software synchronization event. This creates temporally aligned image sets at the software level.

However, this method does not guarantee that the cameras expose their image sensors at the exact same physical instant. Since the cameras are free-running USB devices, each camera exposes its image independently based on its own internal clock. As a result, the real acquisition times of the images may differ by several milliseconds due to USB transfer delays, operating system scheduling jitter, and asynchronous camera timing. The current implementation therefore performs post-acquisition synchronization by grouping the most recent frames from all cameras into a temporally aligned frame set. The maximum temporal offset between cameras is bounded by the camera exposure timing, USB transfer latency, operating system scheduling jitter, and the synchronization polling interval.

A hardware-triggered synchronization approach solves this limitation by forcing all cameras to start image exposure from the same electrical trigger signal. In such a system, an external trigger pulse generated by dedicated hardware or a synchronized GPIO signal is distributed simultaneously to all cameras. Because all cameras receive the same trigger edge at nearly the same time, image exposure becomes physically synchronized rather than only software-aligned afterward. This reduces inter-camera timing offsets from the millisecond range typical of software synchronization to the microsecond range determined primarily by trigger signal propagation and camera internal response latency.

This hardware-trigger concept is also aligned with the synchronization requirements defined in the system requirements document, where the Raspberry Pi 5 is intended to generate hardware-timed trigger pulses referenced to a GNSS-disciplined clock in order to achieve globally synchronized image acquisition across all cameras.

The GNSS receiver provides a single configurable timing output (PPS/TPS), which is intentionally reserved for future hardware-triggered camera synchronization rather than direct Linux kernel clock disciplining. This design decision prioritizes deterministic image exposure timing across all cameras, since hardware-triggered acquisition provides a significantly greater improvement in multi-camera temporal consistency than further increasing system clock accuracy within the current software-based synchronization architecture.

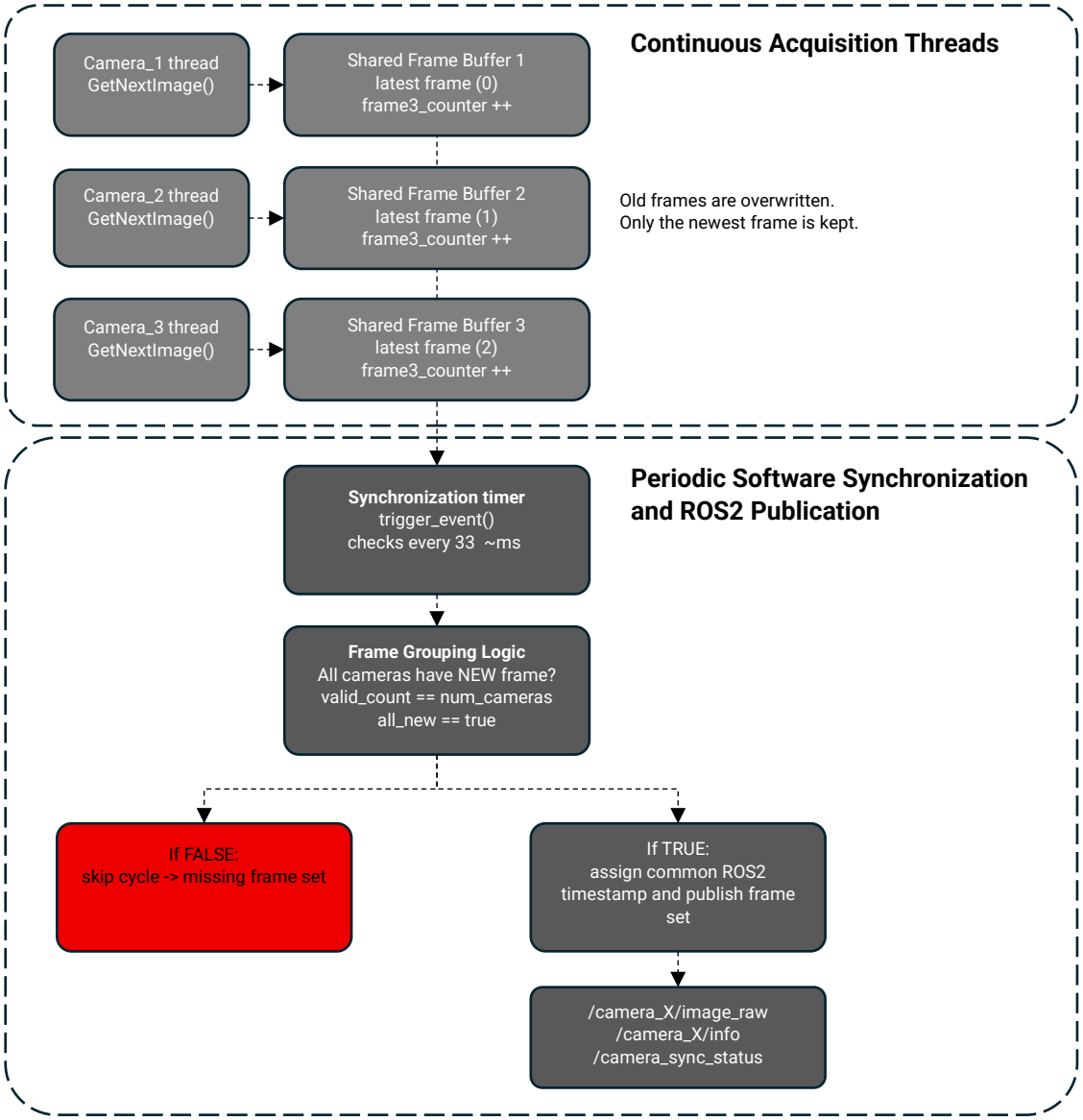


Fig. 19. Software-Based Multi-Camera Acquisition and Synchronization

The software architecture is intentionally designed to remain compatible with future hardware-triggered synchronization. Since synchronization handling is abstracted within the driver architecture, hardware-triggered acquisition can later be integrated with minimal modification to higher-level software components.

In addition to image acquisition, the driver node manages camera configuration parameters. Centralized parameter management simplifies reproducibility and ensures consistent operating conditions across all cameras.

The driver additionally publishes intrinsic calibration information through standardized ROS2 camera interfaces. Maintaining direct association between image streams and calibration data is essential for downstream perception tasks that rely on geometrically consistent image measurements.

Communication with the remaining software architecture is realized through standardized ROS2 message interfaces, enabling modular integration of visualization, calibration, mapping, and perception components without requiring modifications to the acquisition framework.

GNSS and Time Integration

As discussed previously, synchronized timestamps are essential for maintaining temporal consistency between distributed sensor measurements. The current implementation uses GNSS-provided time information transferred via USB communication. Although this provides a globally referenced time source, the absence of a dedicated PPS synchronization signal limits timing precision due to communication latency and operating-system scheduling effects.

The synchronization approach reduces long-term system clock drift and enables all sensor data streams to share a common global time reference within the ROS2 framework.

After system startup, the GNSS receiver requires an initial cold-start phase before valid GNSS time becomes available. During evaluation, approximately 60 seconds were required before the Raspberry Pi system clock and system date were synchronized to GNSS time. Until synchronization is established, the system temporarily operates using the local unsynchronized clock.

GNSS-derived positioning data is published independently and can be used for future localization and sensor fusion applications. A high-level overview of the GNSS-referenced timestamp integration is shown in Fig. 20.

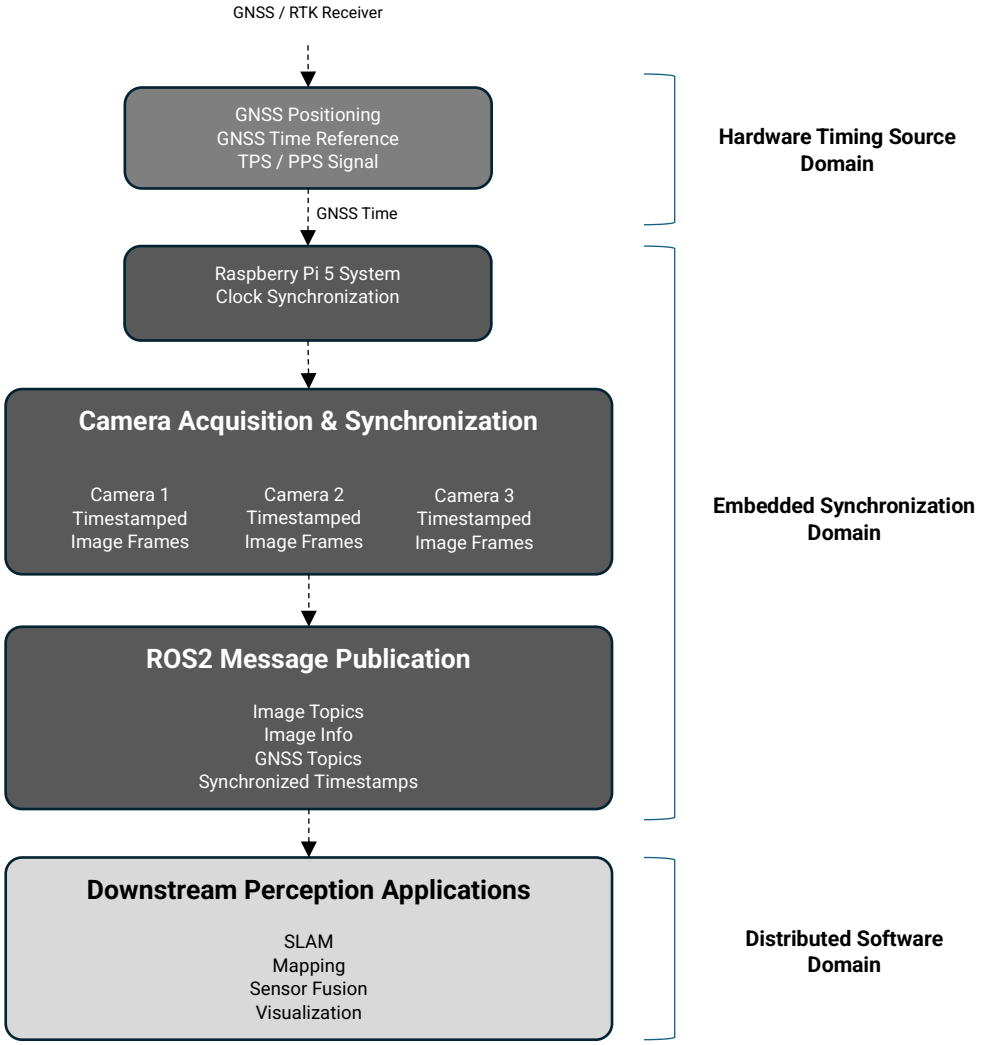


Fig. 20. GNSS-Referenced Timestamp Integration Within Software Architecture

Communication and Data Flow

The communication architecture ensures consistent propagation of image data, calibration parameters, and synchronized timestamps throughout the processing pipeline. To achieve this, the camera acquisition layer publishes synchronized image streams together with the corresponding camera calibration information using standardized ROS2 interfaces.

As discussed previously the embedded platform primarily operates as a synchronized acquisition and data distribution layer, while computationally intensive perception tasks are executed on external computing hardware connected through the robotic network infrastructure.

Within the communication architecture, timestamps derived from the GNSS-referenced system clock are propagated consistently across all sensor messages. This enables downstream software components to temporally associate image streams, positioning data, and future sensor inputs using a common global time base.

The use of standardized ROS2 communication interfaces additionally improves interoperability with existing ROS2 tools and perception frameworks while simplifying future system extensions.

System Implementation

The implementation chapter describes the practical realization of the hardware and software architecture introduced in the previous chapter. While the architecture chapter defines the system structure and design rationale, this chapter focuses on how the sensor suite was physically assembled, configured, and integrated into the ROS2-based robotic platform.

The implementation is organized into mechanical integration, electrical integration, embedded system setup, GNSS time synchronization, camera driver implementation, software synchronization, and ROS2 integration. This structure separates the physical realization of the sensor suite from the software components required for synchronized data acquisition and distribution.

Particular emphasis is placed on implementation decisions that affect synchronization accuracy, calibration stability, data throughput, maintainability, and reproducibility. Practical limitations encountered during integration, such as bandwidth constraints, software compatibility issues, and incomplete hardware-trigger cabling, are explicitly discussed where relevant.

Mechanical Integration

Sensor Layout and Field of View

The spatial arrangement of the camera system is designed to maximize environmental coverage in the primary driving direction of the robot while maintaining sufficient overlap between adjacent cameras for reliable multi-camera calibration and feature association. Fig. 21 illustrates the implemented field-of-view configuration of the three-camera setup.

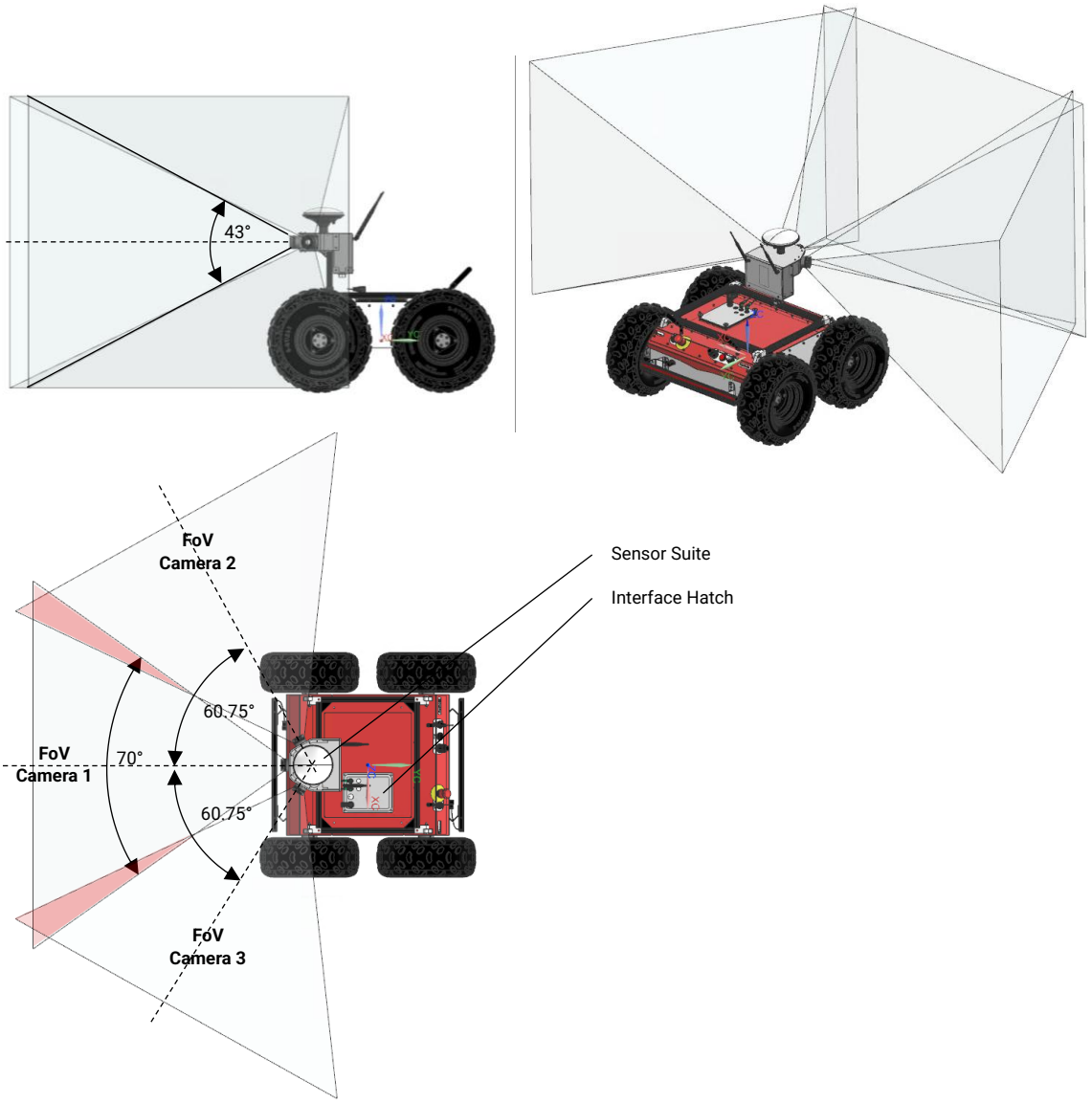


Fig. 21. Field-of-View Visualization of the Implemented Three-Camera Configuration With Approximately 10° Overlap Between Adjacent Camera Views

The cameras are arranged in a forward-facing configuration with defined angular offsets, resulting in a wide combined horizontal field of view of approximately 190° around the front of the robotic platform.

A deliberate overlap region of approximately 10° is maintained between adjacent camera views. As discussed previously, this overlap is required to establish geometric correspondences during extrinsic calibration and to support robust feature matching across camera boundaries. At the same time, the overlap must remain limited in order to maximize the total observable coverage of the environment.

The selected configuration therefore represents a compromise between panoramic coverage and calibration robustness.

Sensor Enclosure and Component Integration

The sensor suite enclosure integrates the three RGB cameras, the Raspberry Pi 5, the GNSS/RTK receiver, and the externally powered USB hub into a compact and modular subsystem mounted on the Husarion Panther platform.

Fig. 22 shows the assembled sensor suite and the integration of the main hardware components within the enclosure.

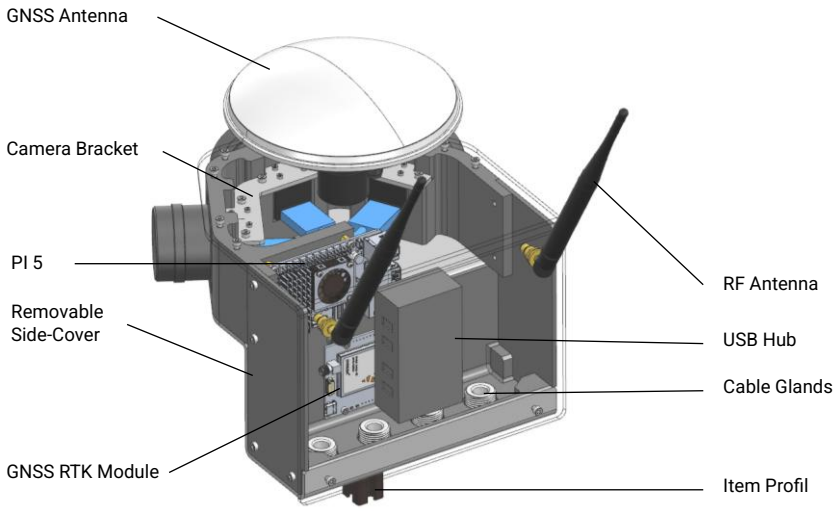


Fig. 22. Sensor Suite Assembly

The enclosure follows a cost-optimized hybrid construction combining sheet metal elements with 3D-printed structural components. Sheet metal parts provide the primary structural rigidity of the enclosure, while additive-manufactured components enable flexible integration of camera mounts, cable routing elements, and internal fixtures without requiring complex machining processes.

This manufacturing approach significantly reduces production cost and simplifies iterative prototyping, but introduces limitations in environmental robustness and long-term mechanical rigidity compared to industrial enclosure solutions. The resulting enclosure achieves an approximate protection level of IP42 and is therefore not intended for prolonged operation in heavy rain or highly dusty environments.

Cable glands integrated at the bottom of the enclosure provide controlled routing of external connections while also acting as strain relief elements. In addition to improving cable organization, this reduces mechanical stress on internal connectors and contributes to basic environmental sealing of the enclosure.

The complete sensor suite is mounted on a single 20×20 Item profile, simplifying installation on the Husarion Panther platform (Fig. 22). The use of the Item profile additionally allows flexible adjustment of the sensor suite height. This enables fine-tuning of the maximum field of view (FoV) while preventing obstruction by the Panther's wheels and simultaneously minimizing the overall height of the robot.

M12 cable connectors integrated into the interface hatch (Fig. 22) allow quick installation and removal of the complete sensor suite.

Camera Module Design

Fig. 23 illustrates the cross-sectional implementation of the camera module integrated within the sensor suite enclosure. Each camera is mounted to a common aluminium bracket designed to provide stable optical alignment and reproducible positioning of the optical components. Compared to the surrounding 3D-printed structural components, the aluminium bracket provides improved dimensional stability under temperature variations. This reduces thermally induced relative motion between the cameras and helps maintain calibration consistency during extended operation.

In addition, the comparatively large thermal mass and high thermal conductivity of the aluminium structure allow it to function as a passive heat sink for the camera modules, supporting thermal stability during continuous operation.

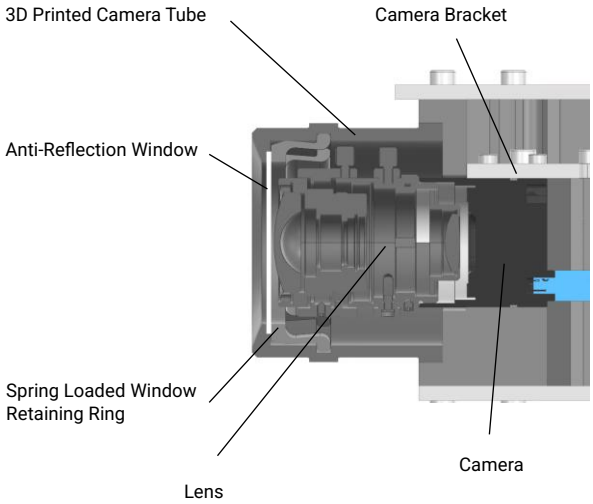


Fig. 23. Cross-Section View Camera with Lens Tube

The C-mount lenses are directly attached to the cameras. Due to the aperture and focus adjustment screws located on the lenses, installation must be performed from the outside of the enclosure. To protect the sensitive optics and electronics from environmental ingress, 3D-printed lens tubes incorporating anti-reflective coated optical windows are used. Each lens tube is designed as an independent sub-assembly that can be installed or removed from the outside, simplifying maintenance, cleaning, and lens replacement. Particular attention is placed on minimizing optical disturbances introduced by the protective window. The anti-reflective coating reduces unwanted reflections and ghosting artifacts that could otherwise degrade image contrast and negatively affect feature detection during calibration and downstream perception tasks.

The use of 3D-printed components simplifies fabrication and enables rapid modification of the mechanical design throughout system development while still providing sufficient structural rigidity for the intended research application. However, such an approach would generally not be suitable for an industrial-grade product due to the limited long-term durability, reduced structural stability, and less favourable thermal characteristics of typical 3D-printed materials compared to industrial manufacturing methods and materials.

Electrical Integration

The electrical integration of the sensor suite follows the architecture introduced previously (see Fig. 15 on page 23) and focuses on stable operation of the embedded multi-camera system under practical robotic operating conditions.

A dedicated trigger-signal wiring path between the GNSS subsystem and the cameras is already prepared within the enclosure to support future hardware-triggered synchronization. Although the trigger signal is not yet electrically connected in the current system configuration, the required signal-routing infrastructure has already been integrated into the hardware design through the use of a terminal block mounted on the Raspberry Pi 5 (Fig. 24).

The terminal block provides a robust and maintainable solution for distributing the GNSS TPS signal to the individual cameras while additionally simplifying common grounding between the cameras, GNSS subsystem, and embedded platform. This preparation enables future implementation of deterministic hardware-triggered image acquisition without requiring major modifications to the existing enclosure or internal wiring layout. It should be noted that the Raspberry Pi itself is not intended to generate the synchronization signal directly. Instead, the GNSS subsystem serves as the primary timing source, while the terminal block is used exclusively for signal distribution and electrical integration.

Fig. 24. Terminal Block for PPS Signal Distribution [18]

Embedded System Setup

This section describes the software and operating-system-level setup of the Raspberry Pi 5 used as the embedded processing platform of the sensor suite.

The implementation includes:

- installation and configuration of the operating system and ROS2 environment,
- integration of the camera SDK and required software dependencies,
- system-level configuration for stable USB3 device operation,
- and network integration into the Husarion Panther platform.

The following subsections describe the implemented embedded software environment and the practical considerations encountered during system setup.

Raspberry Pi Setup

Operating System and ROS2 Environment
The Raspberry Pi 5 operates using Ubuntu 24.04 together with ROS2 Jazzy. A ROS2 workspace is used to organize the custom camera driver, launch descriptions, configuration files, and supporting packages required for operation of the perception system. Detailed installation procedures and dependency management are documented within the project repository.

Camera SDK Integration
A major practical challenge during implementation was integration of the Teledyne FLIR Spinnaker SDK required for interfacing with the Firefly USB3 cameras. At the time of development, the SDK was not officially supported on Ubuntu 24.04 for ARM64-based platforms.

To overcome this limitation, the Ubuntu 22.04 ARM64 version of the SDK was integrated manually and validated on the target system. Although this configuration represents a compatibility workaround, stable operation of all three cameras was achieved during continuous multi-camera acquisition. Detailed instructions on how to install the SDK are documented within the project repository.

Networking
The Raspberry Pi is integrated into the Husarion Panther network infrastructure using a static Gigabit Ethernet configuration to ensure deterministic communication and reproducible network access between the sensor suite and external processing components.

System Configuration
The Raspberry Pi is configured to support stable operation of multiple high-bandwidth USB3 devices during continuous multi-camera acquisition. This includes the use of an externally powered USB hub together with system-level USB and device-management configuration to ensure reliable camera detection and communication stability.

The complete software implementation, configuration structure, and deployment workflow are maintained within the project repository, enabling reproducible system setup and version-controlled development.

GNSS Time Synchronization Implementation

To separate positioning and synchronization functionality, the receiver is configured to provide two independent TCP data streams:

- TCP port 3002: SBF stream for positioning and status data,
- TCP port 3001: NMEA stream for GNSS time synchronization.

This separation simplifies independent handling of positioning and timing information within the ROS2-based software architecture

GNSS Receiver Configuration
The GNSS receiver is configured using the integrated web-based user interface (Fig. 25) provided by the manufacturer. The interface is accessible through the local network connection of the receiver and allows configuration of communication interfaces, GNSS output streams, and timing settings.

Within the implemented system configuration, separate TCP streams are configured for positioning and time synchronization data. The resulting configuration is stored directly within the receiver boot configuration to ensure persistent and reproducible behavior across system restarts.

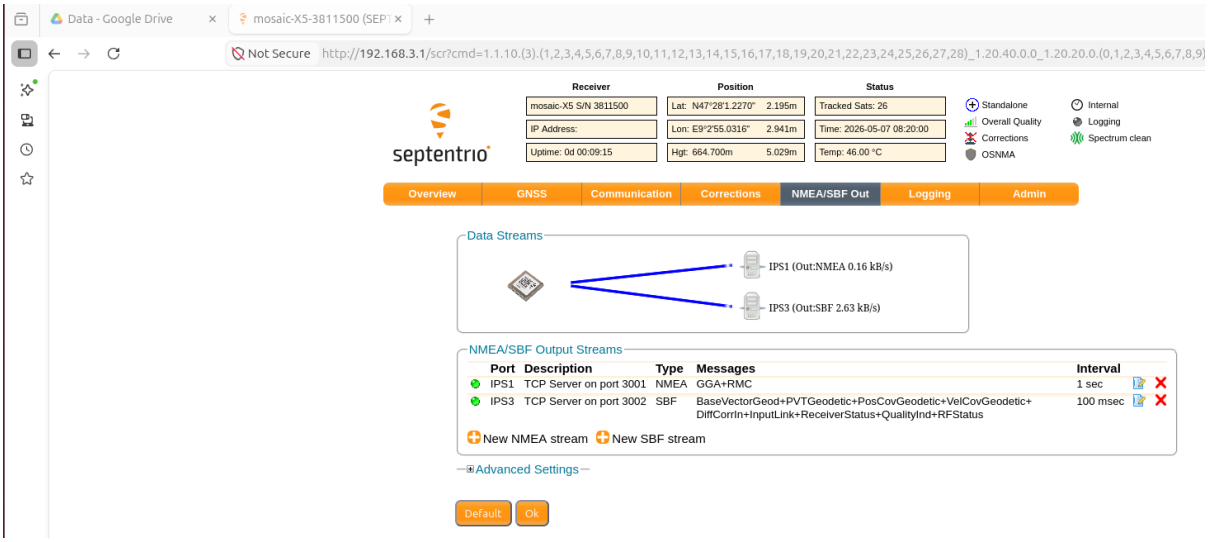


Fig. 25. GNSS Receiver Configuration Interface Used for Persistent Stream Configuration

Automatic GNSS receiver configuration through the ROS2 software stack is intentionally disabled. This avoids unintended runtime configuration changes and ensures that the receiver operates with a consistent and validated configuration during system operation.

System Time Synchronization
The Raspberry Pi system clock is synchronized to GNSS time using the NMEA data stream provided by the receiver.

Time synchronization is implemented using the following software chain:
GNSS receiver → NMEA (TCP 3001) → gpsd → chrony → system clock

Within this configuration, the gpsd service receives and interprets the GNSS time data, while chrony disciplines the Linux system clock accordingly. As a result, all timestamps generated within the ROS2 framework are referenced to a common GNSS-derived global time base.

Compared to direct PPS-disciplined synchronization, the presented NMEA-based synchronization approach introduces additional latency and timing jitter caused by network transport, software processing, and operating-system scheduling. However, the achieved synchronization accuracy is sufficient for the current software-based multi-camera synchronization approach.

To verify correct GNSS communication and time availability, the GNSS data stream was inspected using the cgps monitoring utility provided by the gpsd framework. The output shown in Fig. 26 confirms that the receiver provides valid positioning information together with a stable GNSS-derived timestamp and active 3D satellite fix status. This demonstrates that GNSS time data is successfully received through the NMEA stream and made available to the Linux time synchronization framework.

Detailed configuration procedures, deployment scripts, and synchronization utilities are documented within the project repository.

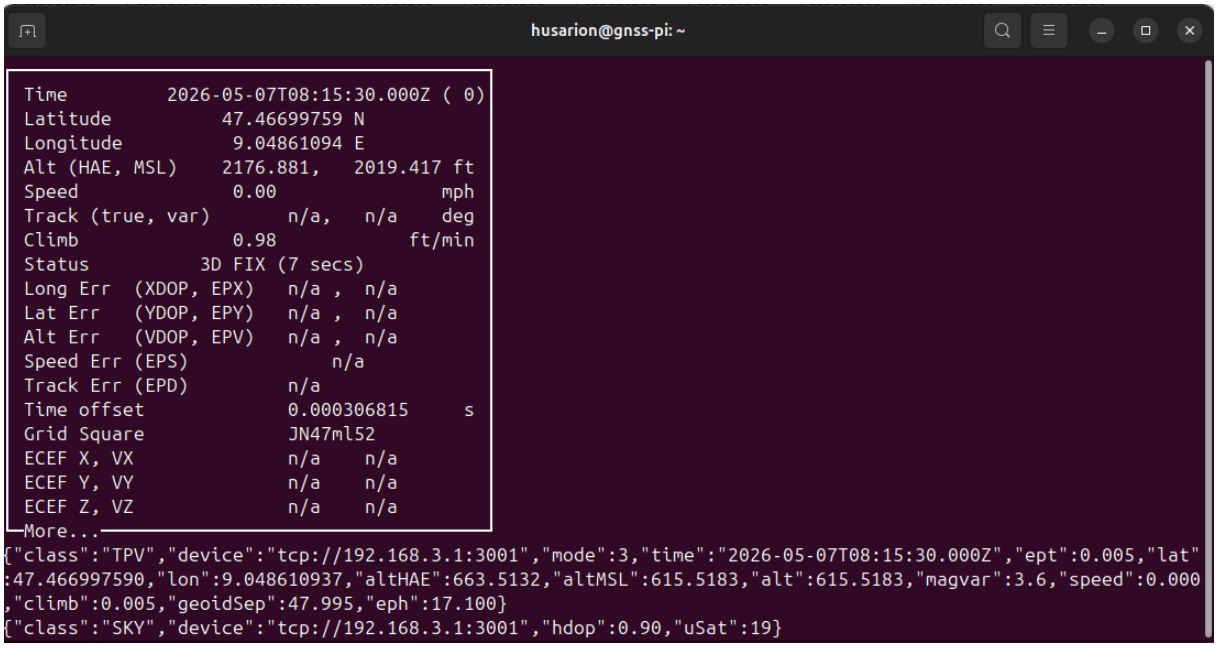


Fig. 26. GNSS Status and Time Information Obtained via cgps

Positioning Data Integration
Positioning and status information are provided through the SBF data stream on TCP port 3002. The data is integrated into the ROS2 environment and made available for localization, mapping, and future sensor fusion applications.

The GNSS receiver configuration is managed externally using the manufacturer-provided configuration interface (Fig. 25).

Camera Driver Implementation

Driver Overview
The camera driver is implemented as a custom ROS2 node in C++ using the rclcpp API and the Teledyne FLIR Spinnaker SDK. The driver is responsible for camera initialization, image acquisition, synchronization handling, and ROS2-based data publication.

A custom implementation was chosen in order to achieve deterministic multi-camera handling and application-level synchronization tailored to the requirements of the developed perception system. In contrast to independent single-camera nodes, the implemented driver manages all cameras within a shared process. This reduces communication overhead and simplifies coordinated handling of synchronized multi-camera acquisition.

The driver implementation depends on ROS2 rclcpp, sensor_msgs, camera_info_manager, and the Teledyne FLIR Spinnaker SDK for low-level camera interfacing and calibration management.

Configuration Management
Driver configuration is implemented using a YAML-based parameter structure defining camera identifiers, acquisition parameters, synchronization settings, and calibration file locations.

A major configuration objective is reproducible and deterministic multi-camera operation. Camera serial numbers are therefore explicitly assigned within the configuration to maintain a fixed mapping between physical camera positions and logical ROS2 camera instances.

Intrinsic calibration data is loaded automatically during driver initialization using resolution-specific calibration files associated with each camera stream. This ensures consistent propagation of calibration parameters throughout the ROS2 perception pipeline.

The main configurable driver parameters are summarized in Table 3.

Parameter	Description	Implemented Value (Default)
num_cameras	Number of synchronized cameras	3
trigger_rate	Acquisition and publication rate	30 Hz
exposure_time_us	Manual exposure time	2000 µs
image_width	Image width	1440 px
image_height	Image height	1080 px
pixel_format	Spinnaker pixel format	BayerBG8
ros_encoding	ROS2 image encoding	Bayer_bggr8
camera_serials	Deterministic camera mapping	Camera specific

Table 3. Main Camera Driver Configuration Parameters

Detailed configuration files, launch descriptions, and deployment procedures are documented within the project repository.

Image Acquisition
Image acquisition is implemented using the Teledyne FLIR Spinnaker SDK, which provides low-level access to camera configuration and image-stream handling. During initialization, each camera is configured according to the system parameters defined in the YAML-file, including image resolution, frame rate, pixel format, and exposure settings.

The implemented acquisition configuration operates at a frame rate of 30 Hz using the full sensor resolution of 1440 × 1080 pixels. The selected frame rate represents a compromise between temporal resolution, USB bandwidth limitations, and stable multi-camera operation on the embedded platform.

Exposure time is configured manually at 2000 µs in order to maintain consistent image brightness and avoid automatic exposure variations between cameras. The chosen exposure time reduces motion blur during robot movement while maintaining sufficient image brightness under typical indoor operating conditions.

The cameras operate using the BayerBG8 pixel format, which allows transmission of raw Bayer image data with reduced bandwidth compared to fully debayered RGB images. Within ROS2, the corresponding bayer_bggr8 image encoding is used to maintain compatibility with standard ROS2 image-processing pipelines.

To support stable parallel operation of multiple cameras, the driver uses a dedicated acquisition thread for each camera stream. This improves robustness against temporary USB transfer latency and reduces blocking between independently operating camera streams.

To reduce acquisition latency and minimize propagation of outdated image data, the camera stream buffer handling mode is configured to NewestOnly. This ensures that the most recently acquired image is processed preferentially under high system load, at the cost of potentially discarding intermediate frames if acquisition temporarily exceeds processing throughput.

Data Publication
The driver publishes image streams and associated calibration data using standardized ROS2 message interfaces. For each camera, both raw image data and the corresponding camera calibration information are published, enabling direct integration with existing ROS2 perception frameworks and image-processing pipelines.

ROS2 Integration

ROS2 Topics
The camera driver publishes image data and corresponding calibration information for each camera using standard ROS2 message types. Each image topic is accompanied by a matching camera_info topic, ensuring that intrinsic calibration parameters remain directly associated with the corresponding image stream.

All images belonging to the same synchronized frame set share a common timestamp generated from the GNSS-disciplined system clock. This enables temporal association between camera streams, GNSS data, and future sensor inputs.

The main ROS2 interfaces are summarized in Table 4.

Topic	Message Type	Description
/camera_X/image_raw	sensor_msgs/Image	Raw image stream for camera X
/camera_X/camera_info	sensor_msgs/CameraInfo	Intrinsic calibration data for camera X
/camera_X/sync_status	std_msgs/String	Diagnostic information about frame grouping and synchronization state

Table 4. Main Camera Driver Topics

TF tree

In addition to sensor data publication, the system implements a hierarchical coordinate-frame structure using the ROS2 TF framework. The TF tree defines the spatial relationships between the robot platform, the sensor suite, and all integrated sensors.

The main reference frame of the robotic platform is defined by the base_link frame. All sensor-related coordinate frames are attached relative to the intermediate sensor_suite_link frame, which represents the mechanical mounting reference of the complete sensor enclosure.

Each camera is assigned an individual optical coordinate frame following standard ROS optical-frame conventions. The center camera acts as the primary optical reference frame of the multi-camera system, while the left and right camera frames are defined relative to this central camera. This hierarchy provides a consistent basis for multi-camera calibration and downstream sensor-fusion applications.

The GNSS antennas are represented using dedicated coordinate frames that define their spatial relationship to the sensor suite and robot platform. This enables consistent integration of positioning information into the overall perception framework.

All transformations within the implemented system are currently published as static transforms. The transform values are derived from CAD measurements, mechanical design data, and calibration measurements obtained during system integration.

The TF structure provides the spatial reference framework required for downstream applications such as multi-camera calibration, visual odometry, mapping, localization, and future multi-sensor fusion approaches. An overview of the implemented TF tree is shown in Fig. 27.

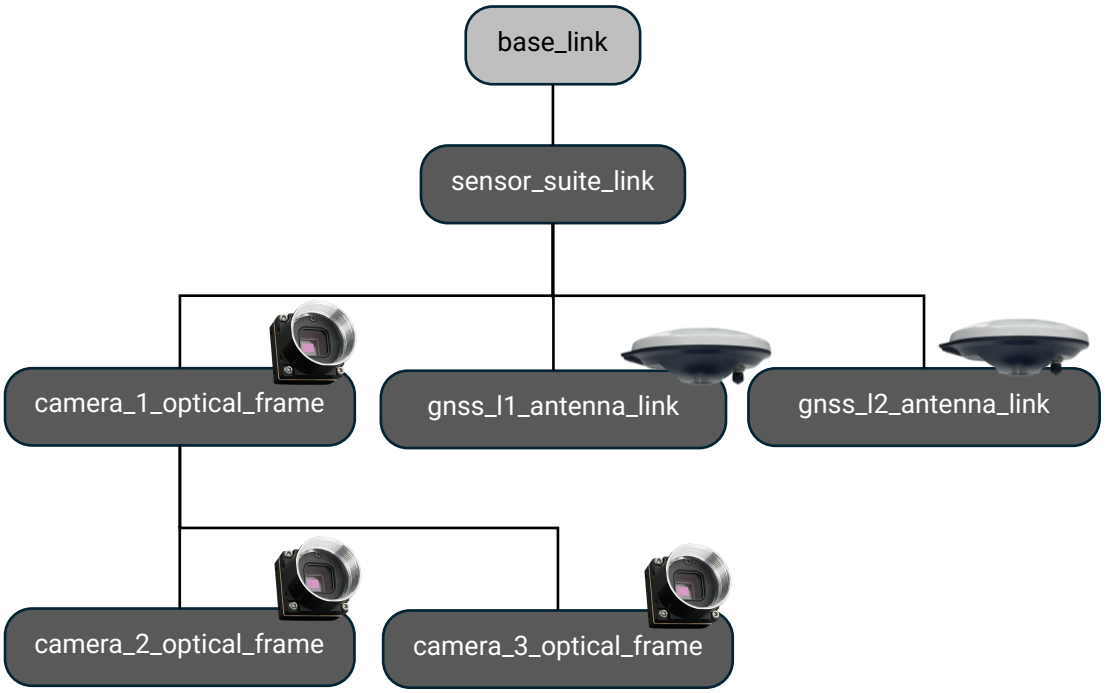


Fig. 27. TF tree Structure

RGB Camera Calibration

The complete calibration workflow is reproducible using the provided scripts, configuration files, and recorded datasets. All calibration parameters, generated calibration files, and evaluation artifacts are documented within the project repository and supplementary materials to ensure reproducibility and traceability of the presented results.

Calibration Overview

The calibration workflow is divided into two stages: intrinsic calibration and extrinsic calibration, as illustrated in Fig. 28. First, intrinsic calibration is performed individually for each camera to estimate the internal camera parameters and lens distortion characteristics required for image rectification. Second, extrinsic calibration is performed to estimate the rigid spatial transformations between the cameras, enabling all camera streams to operate within a common reference frame.

The resulting calibration parameters are integrated into the ROS2 framework and form the basis for synchronized and spatially consistent multi-camera perception.

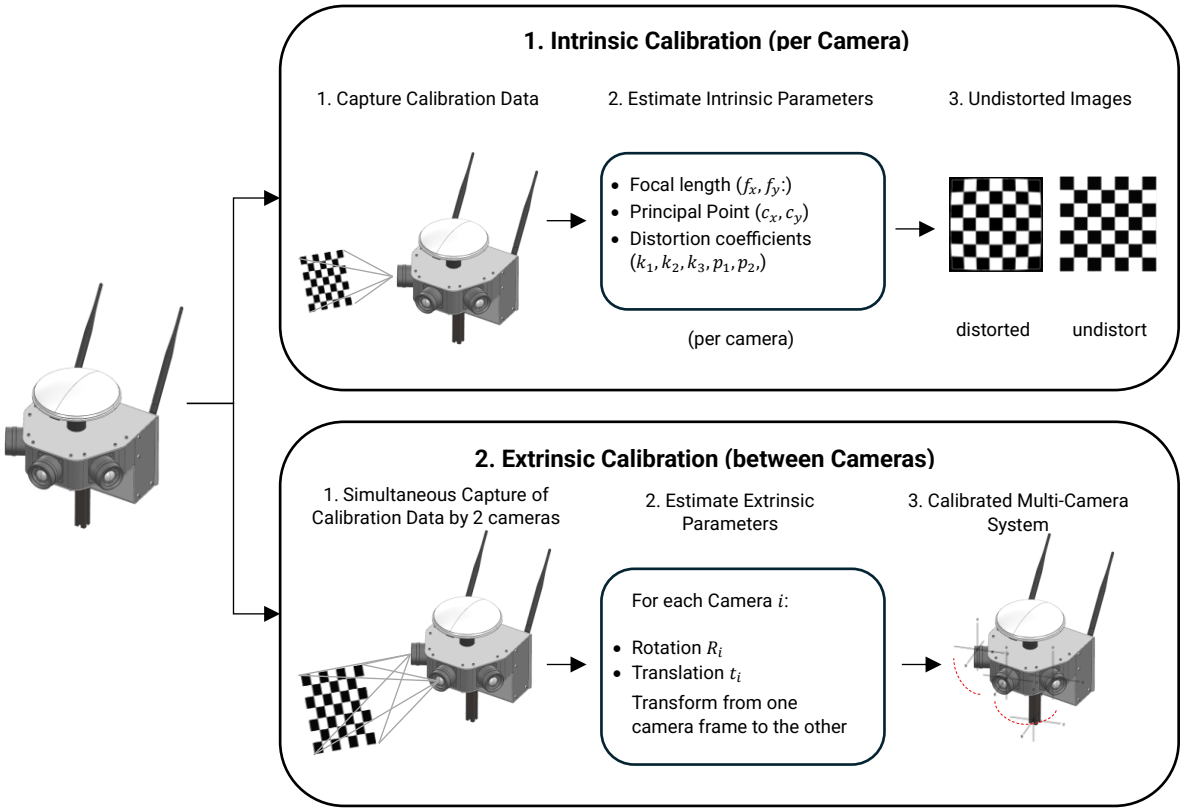


Fig. 28. Camera Calibration Pipeline

Lense Setup

Before performing intrinsic calibration, the optical parameters of each camera lens were manually adjusted and fixed using the integrated set screws shown in Fig. 29. Maintaining a fixed optical configuration is essential because intrinsic calibration parameters are only valid for a constant lens configuration.

The aperture of each lens was adjusted to a fixed mid-range position in order to achieve a balance between light sensitivity and depth of field. Focus was adjusted individually for each camera using a target positioned at a distance of 3 m, corresponding to the expected operational working range of the

system. All cameras were configured using identical aperture settings and a comparable focus distance to ensure consistent imaging behavior across the multi-camera setup.

Changing lens focus or aperture after calibration alters the optical characteristics of the camera system and invalidates the intrinsic calibration parameters. For this reason, the adjusted lens settings were mechanically fixed using the lens set screws.

In mobile robotic applications, it is generally recommended to additionally secure the lens configuration using a low-strength thread-locking compound in order to prevent unintended parameter changes caused by vibration or mechanical stress. In the current development stage, thread locking was intentionally omitted because further modifications to the hardware configuration and recalibration procedures are expected.

The selected optical configuration provides sufficient depth of field while maintaining stable and repeatable calibration conditions for all cameras.

Fig. 29. Lens With Set Screws for Aperture and Focus Adjustment [2]

Intrinsic Calibration

Camera Model
The calibration uses the standard OpenCV pinhole camera model with radial and tangential lens distortion. The intrinsic camera matrix is defined as:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where f_x and f_y denote the focal lengths, (c_x, c_y) the principal point, and sthe skew coefficient. Lens distortion is modeled using radial distortion coefficients (k_1, k_2, k_3) and tangential distortion coefficients (p_1, p_2) following the standard OpenCV distortion model.

Calibration Target

A ChArUco board was used as the calibration target due to its robust marker detection and accurate corner localization capabilities. The board configuration is shown in Fig. 30.

The calibration board was defined with the following parameters:

- Number of squares: 9 × 6
- Square length: 0.025434 m
- Marker length: 0.016532 m
- Dictionary: DICT_4X4_50

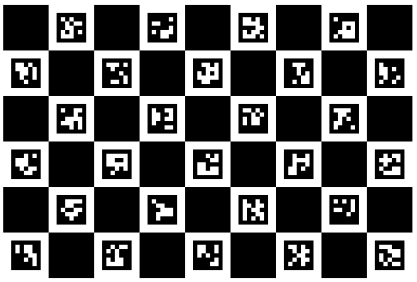


Fig. 30. 9 x 6 ChArUco Calibration Board

The board was displayed digitally on a laptop screen in order to avoid geometric inaccuracies caused by printing distortions or paper deformation. The selected board size provides sufficient feature density while allowing broad coverage of the image plane during calibration

Camera Characteristics

The intrinsic calibration is based on the physical and imaging properties of the employed cameras. The following parameters are particularly relevant:

- Resolution: 1440 × 1080 pixels
- Sensor: Sony IMX296 CMOS
- Pixel size: 3.45 µm
- Shutter type: Global shutter

The resolution defines the image coordinate system in which the intrinsic parameters are expressed. The pixel size, in combination with the focal length, determines the effective field of view and scaling between physical and image coordinates.

Calibration Configuration and Data Acquisition

Calibration was performed using the operational camera configuration described previously in the implementation chapter.

All automatic image-processing features influencing image appearance were disabled, including:

- auto exposure,
- auto gain,
- auto white balance,
- gamma correction,
- sharpening,
- and denoising.

During calibration, the following parameters were kept constant:

- Resolution: 1440×1080
- Exposure time: 20000 µs
- Fixed gain
- Full sensor readout without ROI cropping

To improve calibration consistency, the calibration process was performed under stable environmental conditions. The ambient temperature during calibration was approximately $21°C$, and all cameras were operated continuously for approximately 20 minutes prior to calibration in order to reach a thermally stable operating condition. This reduces the influence of temperature-dependent mechanical or optical variations during parameter estimation.

Initial calibration experiments used continuous image acquisition, which resulted in a large amount of redundant image data and unnecessarily long calibration processing times. To improve the workflow, a dedicated script was developed to enable manual acquisition of calibration images. The script is available within the project repository.

Each camera was calibrated individually using multiple images of the calibration board acquired from different positions and orientations. During acquisition, the board was intentionally moved across the entire image plane, including the image borders, to improve estimation of lens distortion parameters.

The calibration process was performed under stable lighting conditions while avoiding motion blur and screen reflections that could negatively affect feature detection accuracy.

Results
The intrinsic calibration process resulted in accurate camera models for all installed sensors. The interpretation of the results below can be found in the chapter “Intrinsic Calibration Validation” on page 50. Fig. 37 and Fig. 38 on page 51 illustrate the comparison before and after intrinsic calibration.

Camera 1 (Center):

- Serial: 25473579
- Resolution: 1440 × 1080
- RMS reprojection error: 0.114 px

$$K = \begin{bmatrix} 1059.70 & 0 & 720.70 \\ 0 & 1059.92 & 485.59 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D = [-0.3741,\ 0.2585,\ -0.00015,\ 0.00004,\ -0.17897]$$

Camera 2 (Left):

- Serial: 25473576
- Resolution: 1440 × 1080
- RMS reprojection error: 0.090 px

$$K = \begin{bmatrix} 1062.53 & 0 & 732.02 \\ 0 & 1061.79 & 494.41 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D = [-0.3585,\ 0.1839,\ -0.00033,\ -0.00112,\ -0.03360]$$

Extrinsic Calibration

This section describes the extrinsic calibration of the multi-camera system. The objective of extrinsic calibration is to estimate the rigid spatial transformation between the individual cameras and establish a common geometric reference frame for synchronized multi-camera perception.

Calibration Target

The calibration board (Fig. 31). was defined using the following parameters:

- Number of squares: 7 × 5
- Square length: 0.063747 m
- Marker length: 0.047635 m
- Dictionary: DICT_4X4_50

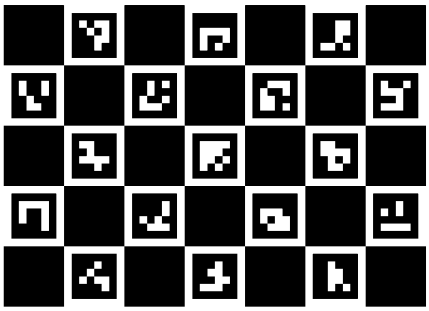


Fig. 31. 7 x 5 ChArUco Calibration Board

The board was displayed digitally on a monitor with a resolution of 2560 x 1440 pixels and visible dimensions of approximately 600 mm x 335 mm. The board occupied approximately 95% of the screen area while maintaining a small margin to avoid clipping effects.

Using a digitally displayed calibration target avoids geometric distortions caused by printing inaccuracies or paper deformation while additionally providing high contrast and repeatable calibration conditions.

Geometric Constraints of the Camera System

The design of the extrinsic calibration setup and choice of the calibration baord (Type and Dimension) was strongly influenced by the geometric configuration of the camera system. As described previously in the hardware integration chapter, the cameras are mounted with an approximate relative viewing angle of 60°, resulting in a narrow overlapping field of view of approximately 10°.

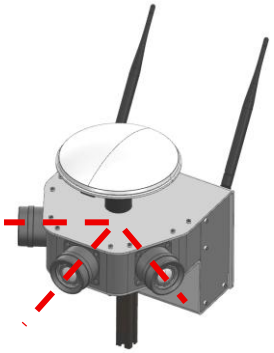


Fig. 32. Geometric Configuration of Cameras (Sensor Suite)

This limited overlap significantly constrains the calibration process. The calibration target must be sufficiently large to remain simultaneously visible in both cameras, while also remaining small enough to fit completely inside the overlap region. In addition, the calibration was performed at an observation distance of approximately 3 – 4 m, corresponding to the expected operational working range of the system. At this distance, the apparent marker size becomes relatively small, requiring a physically large calibration target to maintain reliable marker detection and accurate corner refinement.

The selected 7 x 5 board configuration therefore represents a compromise between:

- sufficient detectable feature density,
- compatibility with the limited overlap region,
- and reliable detection at larger working distances.

These geometric constraints additionally explain the relatively small number of valid stereo image pairs obtained during calibration.

Fig. 33 and Fig. 34 illustrate the setup for extrinsic calibration and also show the minimal overlap region of the FoV.

Fig. 33. Extrinsic Calibration View (Camera_2)

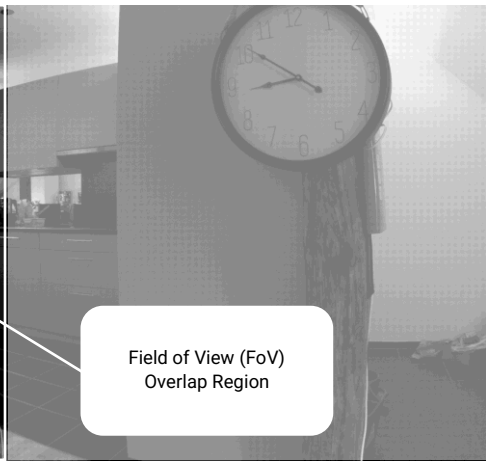


Fig. 34. Extrinsic Calibration View (Camera_1)

Data Acquisition and Calibration Procedure
Calibration images were acquired using the custom multi-camera acquisition script capture_multicam.py. The script records synchronized image pairs from both cameras together with consistent timestamps.

Although hardware-triggered synchronization was not available during calibration, both cameras operated using fixed exposure settings and controlled acquisition timing. Since the calibration target remained static during each acquisition, small temporal offsets between cameras did not significantly influence calibration accuracy.

During data collection, the cameras remained rigidly mounted within the sensor enclosure while the calibration board was manually positioned within the overlapping field of view.

Multiple board poses were recorded with variations in:

- observation distance,
- yaw and pitch orientation,
- slight roll rotation,
- and position inside the overlap region.

These variations improve geometric diversity and support stable estimation of the relative camera transformation.

A total of 46 stereo image pairs were captured. For each image pair, ChArUco features were detected independently in both images using the OpenCV ArUco module. Fig. 39 and Fig. 40 on page 52 illustrate the feature detection.

The detection pipeline consists of:

- marker detection,
- ChArUco corner interpolation,
- and subpixel corner refinement.

Only image pairs satisfying the following quality criteria were retained:

- successful ChArUco detection in both images,
- and at least eight common corners visible in both views.

Image pairs failing these conditions were automatically discarded.

Extrinsic calibration was performed using the OpenCV function cv::stereoCalibrate(). The optimization was executed using fixed intrinsic parameters and distortion coefficients obtained from the intrinsic calibration stage. The stereo optimization minimizes the reprojection error over all valid image pairs and estimates a single rigid transformation between the two cameras.

Results
Out of the 46 captured stereo image pairs, 12 satisfied the required calibration quality criteria and were retained for optimization.

The relatively low number of valid pairs is consistent with the limited overlap region of the camera configuration. Nevertheless, the retained image pairs provided sufficient geometric diversity and feature correspondences for stable pose estimation.

The resulting stereo reprojection error is:

$$\text{RMS reprojection error} = 0.889\ \text{px}$$

A detailed interpretation of this result is available in the chapter “Extrinsic Calibration Validation” on page 51.

The corresponding Euler angles are:
- Roll: $-1.35°$
- Pitch: $-60.74°$
- Yaw: $0.04°$

The resulting homogeneous transformation matrix is:

$$T_{\text{camera\_2}\rightarrow\text{camera\_1}} = \begin{bmatrix} 0.4888 & 0.0198 & -0.8722 & -0.0834 \\ 0.0003 & 0.9997 & 0.0229 & -0.0016 \\ 0.8724 & -0.0115 & 0.4886 & -0.0793 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Testing and Validation

Testing Overview

This chapter evaluates the functionality and performance of the developed multi-camera perception system under pre-defined operating conditions. The validation focuses on both geometric consistency and temporal synchronization performance of the integrated hardware and software architecture.

The evaluation is divided into two main parts:
- geometric validation of the intrinsic and extrinsic camera calibration,
- and temporal performance analysis of the synchronized multi-camera acquisition system.

Geometric validation assesses the accuracy and physical consistency of the estimated camera parameters, ensuring that the system provides geometrically reliable image data for downstream perception tasks.

Temporal validation evaluates the stability of synchronized image acquisition under varying operating conditions, including different resolutions, frame rates, pixel formats, and recording workloads.

All evaluations were performed using the implemented embedded sensor platform and acquisition architecture. Due to the modular system design, validation could be performed independently from the complete robotic platform, allowing isolated evaluation of the sensor suite and synchronization pipeline under controlled conditions. During validation, only two of the three camera sensors were available. Therefore, all experimental results are based on the dual-camera configuration described throughout this chapter.

Experimental Setup

The evaluation was performed using recorded datasets acquired with the implemented dual-camera sensor platform under multiple operating configurations.

For geometric validation, the calibration datasets and resulting intrinsic and extrinsic calibration parameters obtained in the previous chapter were analyzed with respect to reprojection accuracy and physical consistency with the CAD-based sensor geometry.

For temporal performance evaluation, synchronized multi-camera image streams were recorded using the ROS2 acquisition framework under varying resolutions, frame rates, pixel formats, and recording conditions. Depending on the test configuration, image data was either processed directly or additionally recorded using rosbag to evaluate the influence of storage overhead on synchronization stability.

The recorded datasets were analyzed offline using custom Python evaluation scripts based on ROS2 timestamps and synchronization diagnostics generated by the acquisition framework

To investigate the influence of system load on acquisition stability, temporal test configurations were defined with varying image resolution, frame rate, pixel format, and recording conditions. An overview of these temporal performance test cases is provided in Table 5.

ID	Name	Resolution	Format	FPS	Rosbag	Primary Purpose
A	Baseline	720 x 540	Mono8	30	Yes	Reference Performance
B	Full Resolution Mono	1440 x 1080	Mono8	30	Yes	Resolution Stress
C	Low-Rate ROI	720 x 540	Mono8	15	Yes	Low-rate Stability
D	Full Resolution Low Rate	1440 x 1080	Mono8	15	Yes	Resolution vs Rate
E	Colour ROI	720 x 540	BayerBG8	30	Yes	Pixel Format Impact
F	Colour Full Resolution	1440 x 1080	BayerBG8	30	Yes	Worst-case load
G	Colour Full Resolution Low Rate	1440 x 1080	BayerBG8	15	Yes	Recovery Behaviour
H	No recording	720 x 540	Mono8	30	No	Acquisition Baseline
I	With Recording	720 x 540	Mono8	30	Yes	Recording Overhead
J	Full Res No Recording	1440 x 1080	BayerBG8	30	No	Max System Capability

Table 5. Test Case Configurations

All measurements were performed directly on the target embedded platform using the final system configuration described in the implementation chapter. This ensures that the measured performance reflects realistic operating conditions of the deployed perception system.

Evaluation Metrics

Geometric Calibration Metrics

Intrinsic Reprojection Error

The primary metric used to evaluate intrinsic calibration accuracy is the root mean square (RMS) reprojection error. As illustrated in Fig. 35, the reprojection error corresponds to the distance between the observed image point and the point projected back onto the image plane using the estimated camera model. This error is commonly expressed in pixels and serves as a direct indicator of calibration quality.

Reprojection errors within the sub-pixel range are generally considered indicative of a high-quality camera calibration, as they imply that the average deviation between observed and projected feature points is smaller than one image pixel [19].

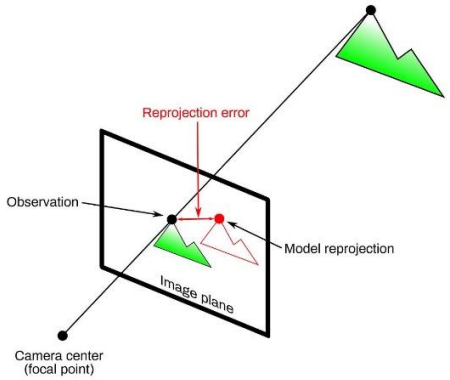


Fig. 35. Schematic of Reprojection Error [19]

Stereo Reprojection Error

For extrinsic calibration, the stereo reprojection error is also used to evaluate the geometric consistency of the estimated spatial transformation between multiple cameras.

In a multi-camera system, the final reprojection error is typically computed as the global root mean square (RMS) reprojection error across all camera pairs and calibration images. As illustrated in Fig. 36, the reprojection error is determined independently for each camera view by comparing the observed image point with the corresponding model reprojection on the image plane. Low stereo reprojection errors indicate that the estimated relative camera poses are geometrically consistent and that the stereo geometry accurately aligns corresponding feature points across all camera views.

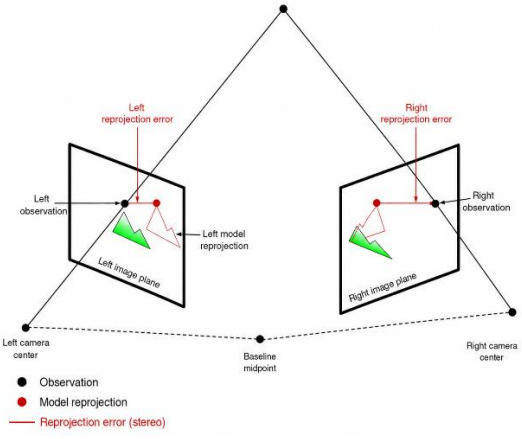


Fig. 36. Schematic of Stereo Reprojection Error Adopted From [19]

Parameter Consistency and Physical Plausibility

In addition to reprojection accuracy, the estimated extrinsic parameters are evaluated for consistency with the known mechanical geometry of the sensor system.

This includes comparison of:

- estimated rotation angles,
- translation magnitudes,
- and relative camera alignment

with the corresponding CAD-based reference geometry of the sensor enclosure.

Agreement between estimated and expected geometry confirms that the calibration results are not only numerically stable but also physically meaningful.

Temporal Performance Metrics

Effective Frame Rate

The effective frame rate describes the actual rate at which synchronized image sets are published by the acquisition system. It is computed from the timestamps of consecutive synchronized frame sets.

Unlike the configured acquisition frame rate, the effective frame rate reflects the combined influence of image acquisition, synchronization, processing load, USB transfer behavior, and optional recording overhead

Jitter

Jitter quantifies the temporal variation between consecutive synchronized frame sets. It is defined as the deviation of the measured inter-frame interval from the nominal acquisition interval.

Low jitter indicates stable and deterministic timing behavior, while increased jitter reflects temporal instability caused by system load, communication latency, or processing delays.

Missing Frame Sets

Missing frame sets represent instances where synchronized image sets are not successfully produced by the system.

This occurs when one or more camera streams fail to provide valid image data within the synchronization window, resulting in incomplete frame sets that are discarded by the synchronization logic.

This metric is used to evaluate the robustness and reliability of the acquisition pipeline under varying operating conditions.

Synchronization Validity

Synchronization validity evaluates whether the acquisition framework successfully produces complete and temporally aligned frame sets across all cameras.

Two synchronization indicators are used:

- valid ratio: proportion of frame sets containing valid data from all cameras,
- all_new ratio: proportion of frame sets in which all cameras provide newly acquired frames.

Together, these metrics assess both completeness and temporal consistency of the synchronized multi-camera acquisition process.

Intrinsic Calibration Validation

The intrinsic calibration results are evaluated with respect to geometric accuracy, consistency between cameras, and practical suitability for downstream perception tasks.

Quantitative Validation of Intrinsic Calibration

The primary quantitative validation metric is the RMS reprojection error obtained during intrinsic calibration.

The calibrated cameras achieved reprojection errors of:

- Camera_1: 0.114 px
- Camera_2: 0.090 px

Both values are well below one pixel and indicate high geometric accuracy of the estimated camera models.

Considering the image resolution of 1440 x 1080 pixels, the resulting errors correspond to sub-pixel calibration accuracy. This confirms that the intrinsic parameters accurately describe the imaging geometry and lens distortion characteristics of the camera system.

Qualitative Validation of Intrinsic Calibration

In addition to numerical evaluation, the intrinsic calibration was validated visually by comparing original distorted images with the corresponding undistorted outputs generated using the estimated calibration parameters.

After undistortion, geometric structures such as monitor edges and straight structrual beam features appear visually straight across the entire image plane. No significant residual barrel or pincushion distortion is observable, including near the image borders.

These observations confirm that the implemented distortion model successfully compensates for the dominant lens distortions present in the camera system.

Representative examples of distorted and undistorted images are shown in Fig. 37 and Fig. 38.

Fig. 37. Image With Distortion (Camera_2)

Fig. 38. Image Without Distortion (Camera_2)

Extrinsic Calibration Validation

Dataset and Geometric Constraints

A total of 46 stereo image pairs were captured for extrinsic calibration. Due to the limited overlap region between adjacent cameras, only 12 image pairs satisfied the required calibration quality criteria and were retained for optimization.

The relatively small number of valid stereo pairs is consistent with the narrow overlap region of approximately 10° between adjacent cameras. Nevertheless, the retained image pairs provided sufficient geometric diversity and feature correspondences for stable pose estimation.

Marker Detection and Stereo Correspondence Validation

Fig. 39 and Fig. 40 show representative examples of successful ChArUco marker detection in images (zoomed) acquired from camera_1 and camera_2 during extrinsic calibration. Each detected marker is assigned a unique identifier (blue numbers), while the overlays (green corners) indicate the detected marker boundaries and corner locations used for stereo correspondence estimation.

The visualizations confirm that reliable marker detection and feature localization are achievable despite the limited overlap region and strong viewpoint differences between the cameras. In the shown example, only a subset of the calibration board is simultaneously visible in both camera views, illustrating the geometric constraints imposed by the approximately 10° overlap region.

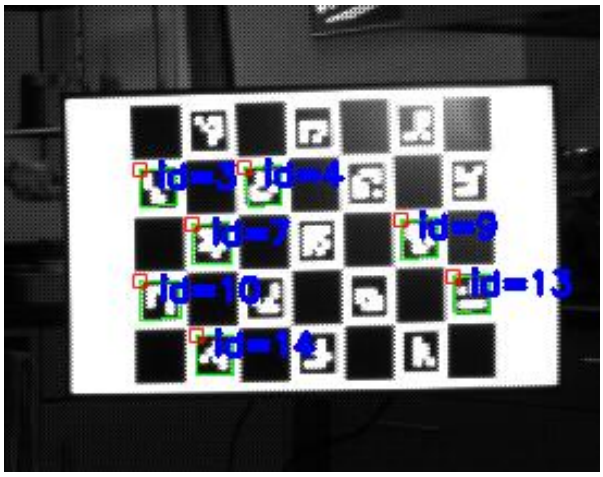


Fig. 39. ChArUco Marker Detection During Extrinsic Calibration (Camera_2)

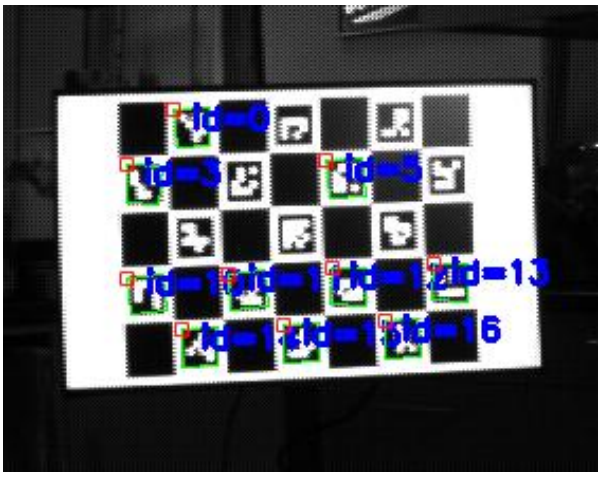


Fig. 40. ChArUco Marker Detection During Extrinsic Calibration (Camera_1)

The detected marker IDs are particularly important for stereo calibration, as they establish consistent correspondences between the 2D image observations and the known 3D locations of the markers on the ChArUco board. Only markers that are successfully detected in both camera images contribute to the set of common feature points used during stereo optimization.

Reprojection Error Evaluation

The resulting stereo reprojection error obtained during optimization is:

$$\text{RMS reprojection error} = 0.889 \text{ px}$$

A reprojection error below one pixel indicates good internal geometric consistency between the estimated stereo model and the observed feature correspondences. Despite the constrained overlap geometry and reduced dataset size, the achieved reprojection accuracy confirms that the estimated camera transformation is numerically stable and suitable for multi-camera perception applications.

Consistency with CAD Geometry

To evaluate the physical plausibility of the estimated transformation, the calibration results were compared with the nominal camera geometry derived from the CAD model of the sensor enclosure.

The estimated relative orientation between camera_1 and camera_2 is:

- Roll: $-1.35°$
- Pitch: $-60.74°$
- Yaw: $0.04°$

The dominant rotation component closely matches the expected mounting angle of 60.75° defined by the mechanical sensor configuration. The deviation between estimated and nominal geometry is below 0.1°.

Temporal Performance Validation

Acquisition Performance without Recording

To establish the baseline capability of the multi-camera system, the acquisition performance was evaluated without rosbag recording. This configuration isolates the image acquisition and synchronization pipeline from additional I/O load and allows assessment of the intrinsic system performance.

Two representative test cases were considered:

- Test H: 720 × 540, Mono8, 30 Hz
- Test J: 1440 × 1080, BayerBG8, 30 Hz

These configurations cover both a reduced-resolution scenario and a maximum-load scenario in terms of resolution and pixel format.

Performance Results
The measured performance metrics for both configurations are summarized as follows:

- Test H (720 × 540, Mono8, 30 Hz)
  - Effective frame rate: 30.303 Hz
  - Jitter: 0.280 ms
  - Missing frame sets: 0

- Test J (1440 × 1080, BayerBG8, 30 Hz)
  - Effective frame rate: 30.305 Hz
  - Jitter: 0.421 ms
  - Missing frame sets: 0

Interpretation
For both configurations, the effective frame rate closely matches the configured acquisition frequency of 30 Hz. No synchronized frame sets were lost during acquisition.

The measured jitter remains below 0.5 ms in both cases, indicating highly stable temporal behavior even under full-resolution Bayer acquisition.

These results confirm stable synchronized acquisition at the target frame rate without observable frame loss.

Impact of Recording Overhead

To evaluate the influence of rosbag recording on system performance, acquisition behavior with and without recording was compared under otherwise identical operating conditions.

The following test pairs were analyzed:

- Test H vs. Test I:
  720 × 540, Mono8, 30 Hz (baseline vs. recording)
- Test J vs. Test F:
  1440 × 1080, BayerBG8, 30 Hz (maximum load vs. recording)

Performance Results

The impact of rosbag recording is summarized in Table 6, which compares system performance with and without recording for both low-resolution and full-resolution configurations.

Configuration	Resoltion	Format	Rosbag	Rate (Hz)	Jitter (ms)	Missing Frames
Test H	720 x 540	Mono8	No	30.303	0.280	0
Test I	720 x 540	Mono8	Yes	29.200	29.954	64
Test J	1440 x 1080	BayerBG8	No	30.305	0.421	0
Test F	1440 x 1080	BayerBG8	Yes	26.701	61.848	206

Table 6. Performance Comparison With and Without rosbag Recording

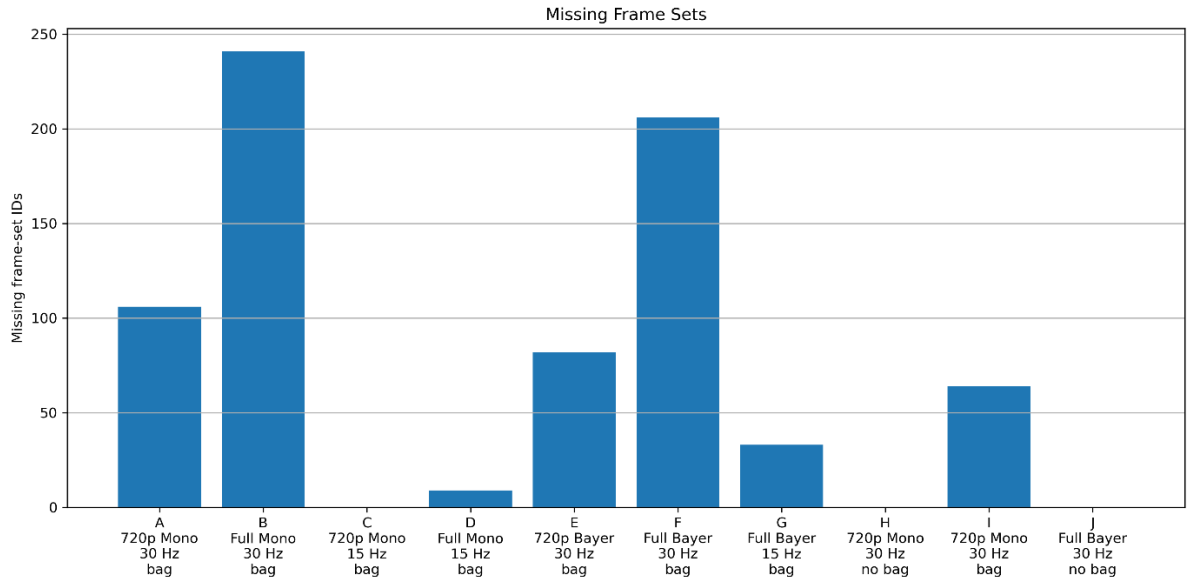


Fig. 41. Missing Synchronized Frame Sets Across Evaluated Test Configurations

Interpretation

Rosbag recording introduces a significant increase in system load and directly affects acquisition stability. Even at reduced resolution, recording causes a measurable decrease in effective frame rate together with substantially increased timing jitter and frame loss. Under full-resolution Bayer acquisition, the impact becomes considerably more pronounced, resulting in reduced synchronization stability and a large number of missing frame sets.

The influence of recording overhead on acquisition reliability is additionally illustrated in Fig. 41. Configurations without rosbag recording exhibit no missing synchronized frame sets, whereas recording scenarios show substantial frame loss, particularly under full-resolution acquisition conditions.

This behavior indicates that the primary bottleneck is not the synchronized acquisition framework itself, but rather the additional storage bandwidth and I/O overhead introduced by high-throughput rosbag recording

Pixel Format Influence

To evaluate the influence of pixel format on system performance, acquisition behavior was compared between monochrome and Bayer-encoded image streams under identical resolution, frame rate, and recording conditions.

The comparison is based on the following test cases:

- Test A: 720 × 540, Mono8, 30 Hz, with recording
- Test E: 720 × 540, BayerBG8, 30 Hz, with recording

This configuration isolates the effect of pixel format while keeping all other parameters constant.

Performance Results
The impact of pixel format on system performance is summarized in Table 7.

Configuration	Resoltion	Format	Rosbag	Rate (Hz)	Jitter (ms)	Missing Frames
Test A	720 x 540	Mono8	Yes	28.474	39.443	106
Test E	720 x 540	BayerBG8	Yes	28.851	32.511	82

Table 7. Impact of Pixel Format on System Performance

Interpretation
The results show only minor performance differences between monochrome and Bayer image encoding.

Effective frame rate, jitter, and frame loss remain within a comparable range for both configurations, indicating that the selected pixel format does not significantly influence overall system stability.

Impact of Image Resolution

To assess the influence of image resolution on temporal performance, reduced-resolution and full-resolution acquisition configurations were compared under identical operating conditions without recording overhead.

The following configurations were analyzed:

- Test H: 720 × 540, Mono8, 30 Hz, without recording
- Test J: 1440 × 1080, BayerBG8, 30 Hz, without recording

Although the two configurations differ in pixel format, the results presented in previous section “ Pixel Format Influence” show that the impact of pixel format on system performance is negligible. Therefore, the comparison can be used to evaluate the effect of resolution without introducing significant bias.

Performance Results
Both configurations achieve stable soft real-time performance at the target frame rate of 30 Hz. The measured jitter remains below 0.5 ms, and no missing frame sets are observed in either case.

Configuration	Resoltion	Format	Rosbag	Rate (Hz)	Jitter (ms)	Missing Frames
Test H	720 x 540	Mono8	No	30.303	0.280	0
Test J	1440 x 1080	BayerBG8	No	30.305	0.421	0

Table 8. Impact of Resolution on System Performance (Without Recording)

Although full-resolution acquisition increases the total data throughput substantially, the resulting increase in jitter remains small and does not significantly affect synchronization stability.

These results demonstrate that the implemented acquisition and synchronization framework is capable of handling full-resolution multi-camera acquisition under nominal operating conditions.

Frame Rate Sensitivity

To evaluate the influence of acquisition frequency on system stability, configurations with identical resolution and pixel format were compared at different frame rates under recording conditions.

The following test cases were considered:

- Test A: 720 × 540, Mono8, 30 Hz, with recording
- Test C: 720 × 540, Mono8, 15 Hz, with recording
- Test F: 1440 × 1080, BayerBG8, 30 Hz, with recording
- Test G: 1440 × 1080, BayerBG8, 15 Hz, with recording

These pairs allow a direct comparison of high and reduced frame rates under both moderate and high system load.

Performance Results
The impact of frame rate on system performance is summarized in Table 9.

Configuration	Resoltion	Format	FPS	Rate (Hz)	Jitter (ms)	Missing Frames
Test A	720 x 540	Mono8	30	28.474	39.443	106
Test C	720 x 540	Mono8	15	15.152	0.912	0
Test F	1440 x 1080	BayerBG8	30	26.701	61.848	206
Test G	1440 x 1080	BayerBG8	15	14.584	42.167	33

Table 9. Impact of Frame Rate on System Performance (With Recording)

The influence of frame rate reduction is additionally visible in Fig. 42, which summarizes the measured synchronization jitter across all evaluated configurations.

Reducing the acquisition frame rate from 30 Hz to 15 Hz significantly improves temporal stability in both the reduced-resolution and full-resolution configurations. This effect is particularly evident when comparing Test A with Test C and Test F with Test G.

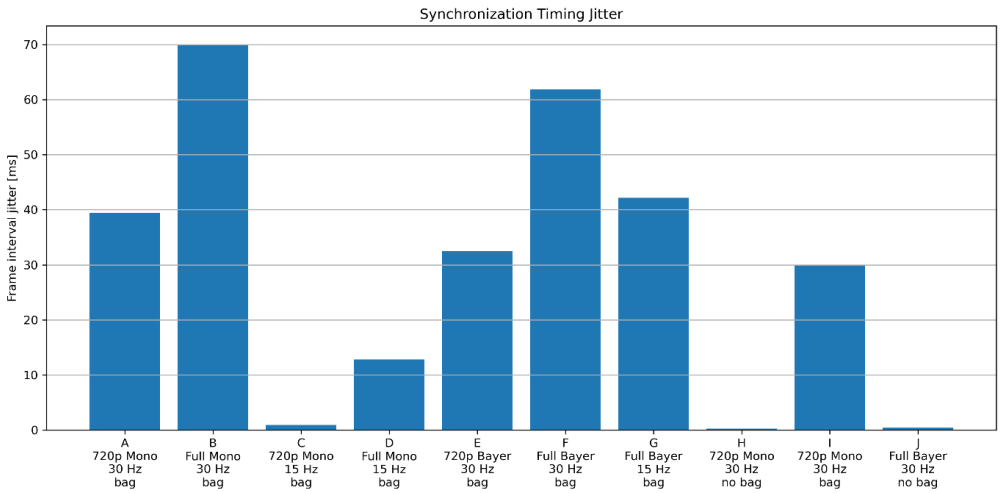


Fig. 42. Synchronization Jitter Across All Temporal Performance Test Cases

Interpretation
Reducing the acquisition frame rate significantly improves temporal stability across all evaluated configurations.

At reduced frame rates, the system exhibits lower jitter and substantially fewer missing synchronized frame sets. In the reduced-resolution configuration, lowering the acquisition rate to 15 Hz eliminates frame loss entirely while maintaining stable timing behavior.

Even under full-resolution Bayer acquisition, reduced frame rates noticeably improve synchronization stability, although some residual frame loss remains due to the continued high data throughput.

These results confirm that acquisition frame rate is one of the dominant parameters influencing overall system stability under high-load operating conditions.

Multi-Camera Synchronization Performance

While the previous sections focused on the temporal performance of the system under varying conditions, the following analysis evaluates the correctness of the multi-camera synchronization mechanism itself.

To evaluate the correctness and reliability of the multi-camera synchronization mechanism, synchronization-specific metrics were analyzed across all test scenarios (A-J). In contrast to previous sections, the focus here is not on performance under varying parameters, but on the ability of the system to consistently produce complete and temporally aligned frame sets.

The embedded platform clock was synchronized using the GNSS receiver via the chrony time synchronization service. Fig. 43 shows the runtime synchronization status of the Linux system clock, where the GNSS receiver is selected as the active timing reference source.

The reported offset remains within the microsecond range, confirming stable GNSS-referenced system time synchronization throughout the evaluation process.

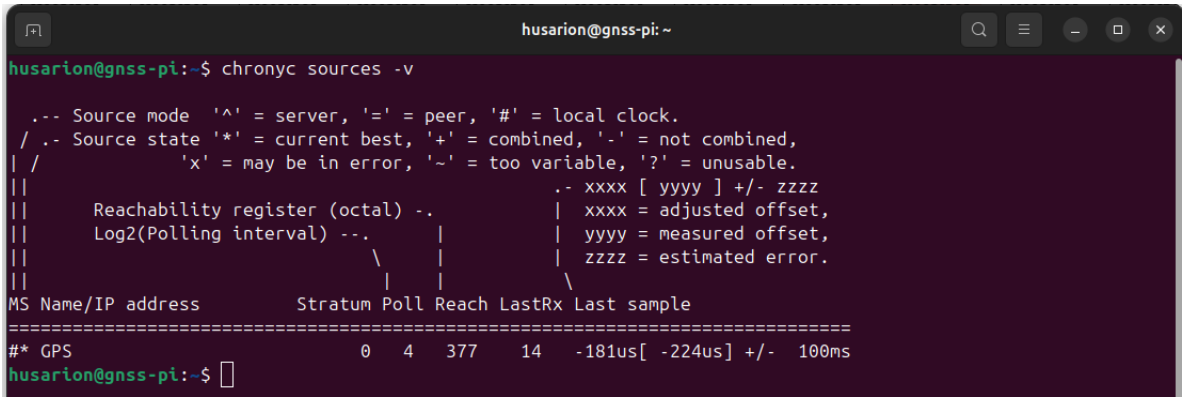


Fig. 43. Verification of GNSS-Disciplined System Time Using chronyc sources -v

Synchronization Metrics
Synchronization quality was evaluated using:

- valid ratio:
  proportion of synchronized frame sets containing valid data from all cameras
- all_new ratio:
  proportion of frame sets in which all cameras contribute newly acquired frames

Performance Results
The aggregated synchronization metrics over the evaluation runs are summarized below:

- Valid ratio: 100.0 %
- All_new ratio: 97.672 %
- all_new = false occurrences: 41

Interpretation
The valid ratio of 100% confirms that the synchronization framework consistently prevents publication of incomplete frame sets.

The all_new ratio of approximately 97.7% indicates that nearly all synchronized frame sets contain newly acquired images from every camera stream.

The small number of cases where the all_new condition is not satisfied can primarily be attributed to temporary acquisition delays under high system load, particularly during rosbag recording.

Importantly, these occurrences do not produce incomplete frame sets but only indicate minor temporal inconsistencies between individual camera streams that are handled by the synchronization logic

Validation Summary

The performed validation demonstrates that the developed perception platform provides geometrically accurate and temporally consistent multi-camera acquisition under practical operating conditions.

Intrinsic calibration achieved sub-pixel reprojection accuracy, while the estimated extrinsic parameters closely matched the expected mechanical sensor geometry derived from the CAD model.

Temporal evaluation confirmed stable synchronized acquisition at 30 Hz under nominal operating conditions, including full-resolution Bayer acquisition. Performance degradation was observed primarily during high-bandwidth rosbag recording, identifying storage and I/O overhead as the dominant system limitation of the current embedded implementation.

The synchronization analysis further demonstrated that the implemented software-based synchronization framework reliably produces complete and temporally aligned multi-camera frame sets across all evaluated configurations.

Overall, the developed platform provides a stable and reproducible foundation for future multi-camera perception and sensor-fusion applications.

Discussion

The developed system demonstrates that a low-cost ROS2-based multi-camera perception platform can be implemented using commercially available USB3 cameras, a Raspberry Pi 5, and GNSS-referenced timing. The platform provides synchronized image streams, calibration data, and globally referenced timestamps within a modular embedded architecture.

At the same time, the validation highlights the practical limitations of achieving deterministic multi-camera perception on low-cost embedded hardware. The current implementation provides software-coordinated synchronization using frame grouping and GNSS-disciplined system time, while the hardware-triggered synchronization path is prepared but not yet physically implemented. Therefore, the system should be interpreted as a functional and extensible perception platform rather than a fully deterministic hardware-synchronized acquisition system.

Hardware Design Discussion

The hardware design successfully integrates the camera subsystem, GNSS receiver, Raspberry Pi, USB hub, power distribution, and trigger-routing infrastructure into a compact modular sensor suite. The modular enclosure design proved useful because the sensor suite could be tested independently from the complete robotic platform. This simplified debugging, validation, and subsystem-level development.

A major strength of the mechanical design is the rigid camera mounting concept. The use of an aluminium camera bracket improves relative camera stability compared to a purely 3D-printed structure and supports calibration consistency during operation. This is particularly relevant because extrinsic calibration depends directly on fixed relative camera poses.

The selected manufacturing approach, based on 3D-printed components and sheet-metal elements, provides a cost-efficient and flexible solution for prototyping. However, it also introduces limitations in long-term rigidity, thermal stability, and environmental robustness compared to industrial-grade machined enclosures. The design therefore represents a practical compromise between cost, manufacturability, modularity, and calibration stability.

The electrical integration also supports future system extension. Although hardware triggering was not yet connected during validation, the trigger-signal routing path and terminal block integration already prepare the system for deterministic GNSS-based camera triggering. This is an important architectural outcome, even though the current implementation still relies on software-level synchronization.

Synchronization and Timing Discussion

The synchronization concept is one of the central engineering aspects of the project. The SRD defines a target of highly accurate inter-camera synchronization and explicitly links the synchronization concept to PPS and trigger-based acquisition. The current implementation does not yet fully satisfy this final hardware-triggered target, because image exposure is not physically triggered by a shared electrical signal.

Instead, the implemented system uses software-coordinated frame grouping. Each camera runs independently, and the driver publishes a synchronized frame set only when new images from all cameras are available. This ensures that incomplete frame sets are not propagated through the ROS2 pipeline and provides a practical level of temporal consistency for proof-of-concept operation.

The validation confirms that this approach is robust under nominal acquisition conditions. However, software synchronization cannot guarantee simultaneous exposure timing. The actual image acquisition times remain affected by camera internal timing, USB transfer latency, operating-system

scheduling, and buffering behavior. Therefore, the current system improves temporal consistency at the software level but does not yet achieve deterministic exposure synchronization.

The use of GNSS-disciplined system time is still valuable because it provides a common global timestamp base for all ROS2 messages. This reduces long-term clock drift and supports future integration with GNSS, LiDAR, IMU, or other sensor data. However, NMEA-based time synchronization through gpsd and chrony cannot replace PPS-based kernel-level clock disciplining or direct hardware triggering when microsecond-level exposure alignment is required.

The design decision to reserve the single configurable GNSS timing output for future camera triggering rather than direct Linux clock disciplining is reasonable. Hardware-triggered exposure synchronization has a larger impact on multi-camera temporal consistency than further improving the system clock while cameras remain free-running.

System Performance Discussion

The temporal validation showed stable synchronized acquisition under nominal operating conditions. Performance degradation mainly occurred during high-bandwidth rosbag recording, where additional storage and I/O overhead introduced increased jitter and occasional missing frame sets. However, this limitation is less critical for the intended system architecture, since the Raspberry Pi primarily acts as a sensor interface and data bridge rather than a platform for long-term storage or computationally intensive processing. Its primary role is to manage image acquisition, synchronization, and data streaming to external processing hardware.

The pixel format evaluation further showed that BayerBG8 acquisition does not significantly reduce performance compared to Mono8 under the tested conditions, supporting the selected use of raw Bayer image data for efficient color image transmission.

Calibration Discussion

The intrinsic calibration results demonstrate that the selected camera and lens configuration can be calibrated accurately. The achieved reprojection errors of 0.114 px and 0.090 px indicate sub-pixel calibration accuracy and confirm that the generated camera models are suitable for image rectification and downstream perception.

The calibration setup was well controlled through fixed lens settings, stable temperature conditions, disabled automatic image processing, and manual image acquisition. These choices improved feature detection consistency and reduced unnecessary dataset size.

Extrinsic calibration was more challenging due to the panoramic camera arrangement. The limited overlap between adjacent camera views increases environmental coverage but reduces the number of common features available for stereo calibration. This explains why only 12 of 46 stereo image pairs were suitable for optimization.

Despite this constraint, the estimated extrinsic transformation is physically plausible. The dominant rotation angle closely matches the expected mechanical camera orientation, and the translation magnitude is consistent with the physical camera spacing. This supports the conclusion that the calibration pipeline is valid for the implemented dual-camera configuration.

However, the calibration results should not be overgeneralized to the full three-camera system. Complete three-camera extrinsic calibration remains future work once all sensors and the final hardware-triggered synchronization path are available.

System Design Trade-Offs

The project clearly demonstrates several system-level trade-offs.

The first trade-off is synchronization accuracy versus implementation complexity. Software synchronization is easier to implement and works with standard USB cameras, but cannot guarantee simultaneous exposure. Hardware triggering requires additional wiring and configuration effort, but is necessary for deterministic acquisition.

The second trade-off is panoramic coverage versus calibration robustness. Wider camera spacing improves environmental coverage and reduces blind spots, but decreases overlap and makes extrinsic calibration more difficult.

The third trade-off is image quality versus throughput. Full-resolution Bayer acquisition provides high-quality image data for perception and calibration, but increases USB bandwidth, memory transfer, ROS2 transport load, and recording overhead.

The fourth trade-off is low-cost modular design versus industrial robustness. The Raspberry Pi and 3D-printed enclosure approach makes the platform affordable and reproducible, but limits throughput, mechanical robustness, and environmental durability.

Limitations of the Current System

The most important limitation is that deterministic hardware-triggered synchronization was not completed and validated. The current system therefore does not yet fully realize the intended synchronization concept defined in the SRD.

A second limitation is that validation was performed using a dual-camera configuration because only two of the three intended camera sensors were available during the experimental phase. As a result, full three-camera extrinsic calibration and panoramic synchronization validation remain open topics.

A third limitation is that validation focused primarily on the perception infrastructure rather than downstream perception performance. The system was not evaluated using higher-level applications such as SLAM, visual odometry, colorized point cloud generation, or complete multi-sensor fusion pipelines.

A fourth limitation is that environmental and long-term robustness were not comprehensively evaluated. Mechanical stability under vibration, outdoor temperature variation, UV exposure, and long-term calibration drift remain future validation topics.

Finally, the embedded platform still exhibits bandwidth and I/O limitations under high-load recording conditions. Although the Raspberry Pi 5 is sufficient for synchronized acquisition and streaming under nominal operation, high-bandwidth rosbag recording significantly affects timing stability.

Despite these limitations, the developed platform establishes a reproducible and extensible foundation for future research on synchronized multi-camera perception and GNSS-referenced sensor integration on low-cost embedded robotic systems

Conclusion and Recommendations

Conclusion

This thesis presented the design, implementation, and evaluation of a modular GNSS-referenced multi-camera perception platform for mobile robotic applications based on low-cost embedded hardware and ROS2.

The developed system integrates synchronized RGB cameras, GNSS/RTK positioning, embedded image acquisition, ROS2-based communication, and a complete camera calibration pipeline within a compact sensor enclosure designed for the Husarion Panther UGV. Particular emphasis was placed on synchronization architecture, deterministic camera management, and modular system integration.

The implementation demonstrated that synchronized multi-camera acquisition using commercially available USB3 machine-vision cameras is feasible on a Raspberry Pi 5–based embedded platform using software-coordinated synchronization and GNSS-referenced system timing. The developed acquisition framework provides synchronized image streams, calibration data, and globally referenced timestamps within a reproducible ROS2 architecture.

Experimental validation showed stable synchronized acquisition under nominal operating conditions, while also highlighting the practical limitations of low-cost embedded perception systems. The results demonstrated that system stability is influenced primarily by recording overhead and embedded platform throughput rather than by the camera hardware itself.

The calibration results confirmed sub-pixel intrinsic calibration accuracy and physically plausible extrinsic transformations consistent with the mechanical sensor design. At the same time, the project highlighted the challenges of extrinsic calibration in panoramic multi-camera configurations with limited overlap regions.

Although deterministic hardware-triggered synchronization was not fully implemented during the project, the developed hardware and software architecture already prepares the required infrastructure for future trigger-based acquisition.

Overall, the developed platform establishes a modular and extensible foundation for future research on synchronized multi-camera perception and embedded robotic sensing systems under realistic engineering and cost constraints.

Recommendations / Future Work

The most important next step is the implementation and validation of deterministic hardware-triggered synchronization using the GNSS-derived timing signal. This would enable physically synchronized image exposure and allow direct comparison between software-based and hardware-triggered acquisition.

Another important extension is the validation of the complete three-camera configuration, including full panoramic extrinsic calibration and synchronization analysis once all sensors are available. Future work should additionally evaluate the developed platform using downstream perception applications such as visual SLAM, visual odometry, panoramic image stitching, or multi-sensor fusion in order to assess the practical impact of synchronization accuracy and calibration quality on higher-level perception tasks.

The project also revealed throughput limitations of embedded USB3 acquisition systems during high-bandwidth recording. Future improvements could therefore investigate optimized recording pipelines, hardware-assisted compression, distributed acquisition architectures, or more powerful embedded processing hardware.
Finally, the modular ROS2-based architecture provides a strong foundation for future integration of additional sensing modalities such as LiDAR, IMUs, radar, or thermal cameras within a unified GNSS-referenced perception framework.

Authorship Declaration

I hereby declare that this work has been carried out independently and without any unauthorized assistance. I have used only the sources and tools that I have explicitly cited. All ideas, data, or text taken from external sources—whether directly or indirectly—have been clearly acknowledged. This work has not been submitted, either in whole or in part, to any other examination authority, institution, or publisher, and has not been published previously. I am aware of and have adhered to the ethical standards applicable to academic research and writing.

Place, Date: Wil, 10th May 2026

Signature: 

Chris Stolz

Appendix

References

[1] Teledyne Vision Solutions, “Firefly S (Model FFY-U3-16S2C-C),” [Online]. Available: Firefly S product page . [Accessed: Mar. 27, 2026].
[2] Kowa Lenses, “LM4NCL 3.5mm C-Mount wide angle lens,” [Online]. Available: https://www.kowa-lenses.com/LM4NCL-1-1.8-3.5mm-C-Mount-wide-angle-lens/10096. [Accessed: Mar. 27, 2026]
[3] “SimpleRTK3B-X5,” ArduSimple. [Online]. Available: https://www.ardusimple.com/product/simplertk3b-x5/ (accessed Nov. 30, 2025).
[4] “AS-ANT2B-SUR-L1L2-25SMA-00,” ArduSimple (Mouser). [Online]. Available: https://www.mouser.ch/ProductDetail/ArduSimple/AS-ANT2B-SUR-L1L2-25SMA-00?qs=vHuUswq2%252BswBsit1qMDm7Q%3D%3D … (accessed Nov. 30, 2025).
[5] “Raspberry Pi 5 8GB Entwicklungsboard Kit,” Digitec. [Online]. Available: https://www.digitec.ch/de/s1/product/raspberry-pi-5-8gb-entwicklungsboard-kit-38955607 (accessed Dec. 1, 2025).
[6] “Radio Module – Extra Long Range,” ArduSimple. [Online]. Available: https://www.ardusimple.com/product/radio-module-extra-long-range/ (accessed Nov. 30, 2025).
[7] WaveShare, “4-Ch USB 3.2 Gen1 HUB Industrielle (USB-A, 4 Ports),” *Digitec*, [Online]. Available: https://www.digitec.ch/de/s1/product/waveshare-4-ch-usb-32-gen1-hub-industrielle-usb-a-4-ports-dockingstation-usb-hub-49026700. [Accessed: Mar. 30, 2026].
[8] FARO Technologies, “How do you colourise a point cloud,” *Resource Library*, [Online]. Available: https://www.faro.com/de-DE/Resource-Library/Article/How-do-you-colourise-a-point-cloud. [Accessed: Apr. 1, 2026].
[9] Husarion, “Panther-5 robot,” Husarion. [Online]. Available: https://husarion.com/assets/images/panther-5-medium-0187b508c426f99ab3af82150c89dd75.png. Accessed: Sep. 10, 2025.
[10] Edmund Optics, “38.1mm Dia. x 1mm High Efficiency Window,” *Edmund Optics*. [Online]. Available: https://www.edmundoptics.com/p/381mm-dia-x-1mm-high-efficiency-window/40085/. [Accessed: Apr. 7, 2026]
[11] “SimpleRTK3B-X5,” ArduSimple. [Online]. Available: https://www.ardusimple.com/product/simplertk3b-x5/ (accessed Nov. 30, 2025).
[12] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot.: Sci. Syst. (RSS)*, Berkeley, CA, USA, 2014.
[13] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easily calibrating omnidirectional cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Beijing, China, 2006, pp. 5695–5701.
[14] Microwatt, "ANYmal X - The ultimate legged robot for hazardous environments," [Online]. Available: https://microwatt.com/products/anymal-x/. Accessed: Apr. 7, 2026.
[15] P. Furgale, J. Rehder, and R. Siegwart, “Unified Temporal and Spatial Calibration for Multi-Sensor Systems,” in *IEEE/RSJ IROS*, 2013.
[16] E. Olson, “AprilTag: A robust and flexible visual fiducial system,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.
[17] “Team,” ISF Institute for Intelligent Systems and Smart Farming, OST. [Online]. Available: https://www.ost.ch/en/research-and-consulting-services/technology/system-technology/isf-institute-for-intelligent-systems-and-smart-farming/team (accessed Dec. 1, 2025).
[18] Great-IT, "KF301-2P 301-3P 5.0mm Pitch Screw Terminal Block Connector," AliExpress. [Online]. Available: https://de.aliexpress.com/item/4001315420216.html. [Accessed: May 8, 2026].
[19] Camcalib.io, “What is the Reprojection Error?,” Camcalib Blog. [Online]. Available: https://www.camcalib.io/post/what-is-the-reprojection-error. [Accessed: 08-May-2026].

Tools used

The following tools were used in the preparation, configuration, and editing of this project: Visual Studio Code, NX Siemens, OpenAI ChatGPT , GitHub, GitLab, Microsoft Teams, Microsoft Word, Septentrio configuration tool, Spinnaker SDK

Remark on AI assistance:

- This thesis involved the use of generative AI tools, specifically ChatGPT (OpenAI), to assist in both software development and documentation. The tool was used for generating and refining code snippets, explaining programming concepts, and supporting the drafting of written content.

  In the preparation of the documentation, ChatGPT was primarily used to improve grammar, enhance clarity, and help restructure explanations. It was not used to produce large sections of text without critical revision, nor to develop core academic arguments.

  All AI-generated material, including code and text, was carefully reviewed, edited, and validated by the author. The author takes full responsibility for the accuracy, originality, and integrity of the final work. No content was included without appropriate human oversight and substantial modification where necessary.

Project Management

Stakeholder Overview

The following section provides an overview of the project's stakeholders and their responsibilities and interests.

Fig. 44: Prof. Dr. Dejan Šeatović [17]

Contact & Personal Information:

Full name:	Prof. Dr. Dejan Šeatović
Position / Responsibility:	Head of ISF / Advisor
Phone:	+41 58 257 47 04
Mail:	dejan.seatovic@ost.ch

Responsibilities:

- Project sponsor
  - Definition of scope and deliverables
- Examination

Interests:

- Successful project according to scope and time line definition.

Fig. 45: Luis Meier [17]

Contact & Personal Information:

Full Name:	Luis Meier
Position:	Sr. Development Engineer at ISF
Phone:	+41 58 257 42 14
Mail:	luis.meier@ost.ch

Responsibilities:

- Review / Approval / Release
- Consulting / Support
- Supervision
- Examination

Interests:

- Enable MSE Student

Fig. 46: Chris Stolz

Contact & Personal Information:

Full Name:	Chris Stolz
Position:	MSE Student
Phone:	+41 78 622 26 54
Mail:	chris.stolz@ost.ch

Responsibilities:

- Project lead
- Development and implementation
- Verification
- Documentation

Interests:

- On-time delivery of project deliverables
- Knowledge acquisition in the field of RGB Camera
- Knowledge acquisition in the field of data acquisition and time synchronization.
- Knowledge acquisition in systems engineering

Project Timeline and Milestones:

The following figure shows the project plan with the most relevant milestones. In order to successfully develop, implement and verify the individual components.

The so called V-Model was applied for this project. The V-Model can be used for guidance and ensures that all system requirements are verified.

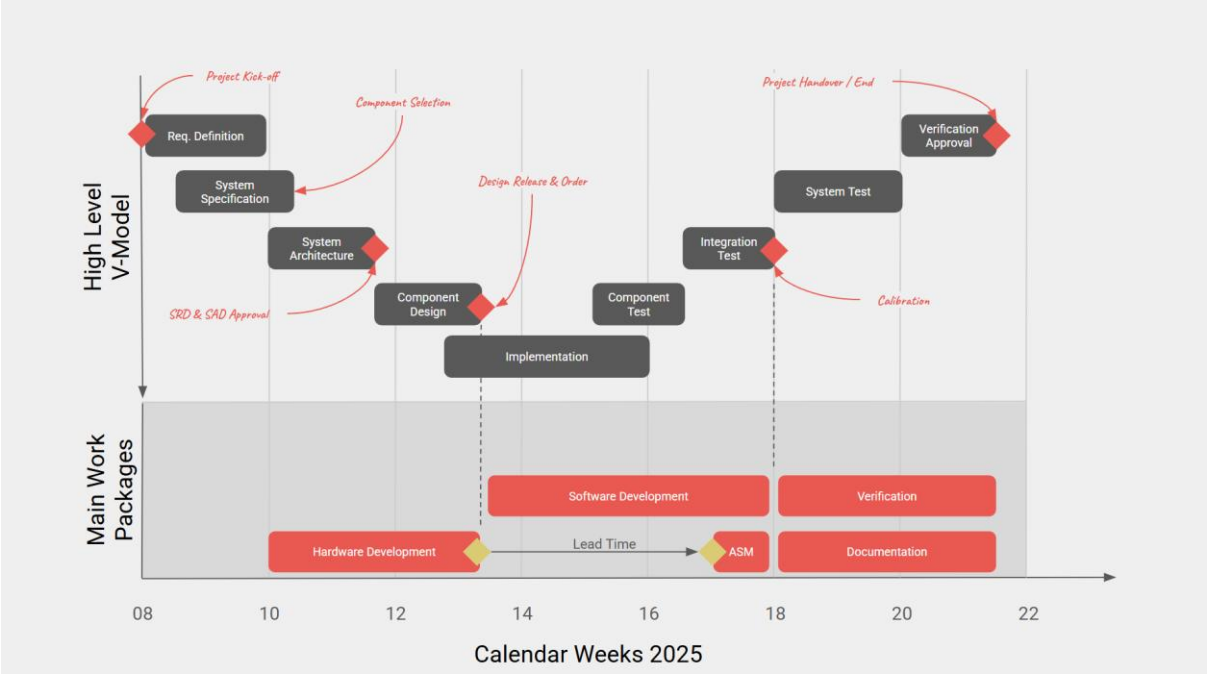


Fig. 47: Project Plan

Appendix Folder

1. System Requirements Document
2. CAD
3. Data Sheets
4. Calibration Data
5. Validation Data