hardware

MDPI

Article

SKD-1: A Modular Skid-Steer Unmanned Ground Vehicle Platform for Robotics Research

Guido M. Sánchez [1,*], Agustín Capovilla [1], Marina Murillo [1], Hugo S. U. Hernández [1], Jesús E. Benavidez [1], Nestor Deniz [1,2] and Leonardo Giovanini [1]

[1] Research Institute for Signals, Systems and Computational Intelligence Sinc(i), FICH-UNL, CONICET, Ciudad Universitaria UNL, Santa Fe 3000, Argentina; acapovilla@sinc.unl.edu.ar (A.C.); mmurillo@sinc.unl.edu.ar (M.M.); hsuhernandez@sinc.unl.edu.ar (H.S.U.H.); ebenavidez@sinc.unl.edu.ar (J.E.B.); ndeniz@sinc.unl.edu.ar (N.D.); lgiovanini@sinc.unl.edu.ar (L.G.)
[2] Department of Engineering, Harper Adams University, Newport TF10 8NB, UK
* Correspondence: gsanchez@sinc.unl.edu.ar

Abstract

This work presents the design, construction and operation of the SKD-1, a modular skid-steer unmanned ground vehicle (UGV) developed as a low-cost research platform for mobile robotics applications. The platform integrates a differential skid-steer drive system, a Raspberry Pi-based onboard computer, and a microcontroller-based control layer implemented using an STM32 microcontroller. The sensing system includes light detection and ranging (LiDAR), global navigation satellite system (GNSS), and an inertial measurement unit (IMU), enabling experiments in localization, mapping, and autonomous navigation. The software architecture is based on the Robot Operating System (ROS) 2 framework, relying on standard ROS 2 packages for perception, mapping, and path planning, with the custom hardware-interface layer being the only non-standard software component. The mechanical and electronic subsystems were designed with a modular architecture that facilitates maintenance, sensor replacement, and hardware upgrades. The primary contribution of this work is the open-hardware design, integration, and documentation of a reproducible robotics testbed, motivated by the prohibitive cost of commercial platforms in resource-constrained research contexts. Indoor and outdoor experiments—covering velocity-tracking, SLAM, and waypoint-navigation trials—demonstrate the functional integration of the sensing, actuation, and computing subsystems.

Keywords: SKD-1; skid-steer UGV; unmanned ground vehicle; ROS 2; autonomous navigation; open hardware

Received: 5 May 2026
Revised: 3 August 2026
Accepted: 18 August 2026
Published: 1 September 2026

1. Introduction

The field of unmanned ground vehicles (UGVs) has experienced significant growth in recent years, driven by the increasing demand for autonomous systems across a wide range of industrial, agricultural, service, and research applications [1,2]. Continuous advances in sensing, embedded computing, and robotics software have enabled the development of increasingly capable mobile platforms for autonomous navigation in structured and unstructured environments [3,4]. Among the various locomotion configurations available for wheeled mobile robots, skid-steer is one of the most widely adopted approaches for all-terrain applications because of its mechanical simplicity, maneuverability, and ability to traverse uneven terrain [5].

A wide variety of research and commercial UGVs platforms are currently available. Commercial systems, including Clearpath Robotics Husky A300 and Jackal [6,7], AgileX Scout 2.0 and Scout Mini [8,9], and Husarion Panther and ROSbot [10,11], provide mature hardware and software solutions but are often prohibitively expensive for many research groups and offer limited flexibility for hardware modification. Conversely, open-source platforms such as TurtleBot [12], Linorobot [13], and academic prototypes including the Noah Robot [14,15], PiBot [16], the omnidirectional shock-absorbed robot [17], ROMR [18], and the Educational Prototype of a Differential Drive [19] substantially reduce acquisition costs while promoting reproducibility and customization. However, many of these platforms target educational or indoor applications and do not simultaneously provide the mechanical robustness, payload capacity, sensing capabilities, and modularity desirable for robotics research requiring operation in both indoor and outdoor environments.

To better position the proposed platform within the current landscape, Table 1 summarizes representative commercial and open-source UGVs according to characteristics relevant to research use, including size, weight, payload capacity, maximum speed, operating time, onboard sensing, and cost. Rather than serving only as a comparison of existing platforms, this analysis identifies the practical trade-offs between affordability, portability, modularity, and operational capability that guided the development of SKD-1.

Table 1. Comparison of SKD-1 with representative commercial and academic platforms. Commercial platform prices were obtained from Génération Robots https://www.generationrobots.com and RobotShop https://www.robotshop.com. All prices are approximate.

Platform	Size [mm]	Weight [kg]	Payload Capacity [kg]	Max. Speed [m/s]	Terrain	Operating Time [h]	IMU	LiDAR	GNSS	Price (USD)
Husky A300 [6]	990 × 698 × 381	80	50–100	1.0	All-terrain	4–12	*	*	*	27,000
Jackal [20]	508 × 430 × 250	17	10–20	2.0	Outdoor	2–8	✓	*	*	17,000
Agilex Scout 2.0 [8]	930 × 699 × 349	67	10–20	1.5	All-terrain	3.5–8	*	*	*	14,200
Agilex Scout Mini [9]	612 × 580 × 245	23	50	3.0	Outdoor	3–8	*	*	*	8300
Husarion Panther [10]	809 × 848 × 370	55	100	2.0	All-terrain	3.5–8	✓	*	*	19,000
Husarion ROSbot 3 [11]	200 × 233 × 197	2.84	5	1.0	Outdoor	1.5–5	✓	✓	×	3200
TurtleBot 3 [21]	281 × 306 × 141	1.8	15–30	0.26	Indoor	∼2.5	✓	✓	×	2200
TurtleBot 4 [7]	341 × 339 × 351	3.9	9–15	0.31	Indoor	2.5–4	✓	✓	×	2400
Linorobot [13]	–	–	–	–	Indoor	–	*	*	×	–
Noah Robot [14,15]	–	–	–	–	Indoor	–	✓	×	×	–
PiBot [16]	–	–	–	–	Indoor	–	×	×	×	<180
ROMR [18]	460 × 340 × 430	17.1	90	2.5	Outdoor	∼8	✓	✓	×	<1500
Omnidirectional shock-absorbed robot [17]	–	–	–	–	Indoor	–	×	×	×	∼600
Educational Prototype of a Differential Drive [19]	136 × 175 × 112	0.935	3	0.39	Indoor	–	×	×	×	∼330
SKD-1 (this work)	**410 × 260 × 370**	**6**	**5**	**0.8**	**Outdoor**	**1–2**	✓	✓	✓	**∼$1750** †

(✓) Included as standard, (×) Not available, (*) Available as an optional add-on, (–) Not reported. (†) Main components, see Table A1.

Guided by these design objectives, this work presents SKD-1, a modular skid-steer UGV developed as an affordable and reproducible research platform for mobile robotics. The platform integrates commercially available sensing, actuation, and computing components within a Robot Operating System (ROS) 2-based software architecture, combining a Raspberry Pi-based onboard computer with a microcontroller responsible for the low-level hardware interface.

Compatibility with ROS 2 [22]—specifically the "Jazzy Jalisco " release— is achieved through the implementation of the embedded firmware, the communication layer between

the microcontroller and the onboard computer, and the hardware abstraction interfaces required for the platform to operate as a standard ROS 2 robot. Consequently, the platform can directly leverage the extensive ROS 2 ecosystem, enabling the integration of established packages for localization, mapping, navigation, perception, and control while minimizing the effort required for hardware-specific software development.

The primary contribution of this work is the design, integration, and documentation of this modular UGV platform as a reproducible robotics testbed. Rather than proposing novel navigation, simultaneous localization and mapping (SLAM), or control algorithms, SKD-1 provides a standardized hardware and software foundation on which such methods can be readily implemented, evaluated, and compared under real operating conditions. By exposing standard ROS 2 interfaces and relying on widely adopted software components, the platform promotes software reusability, interoperability, and algorithm portability across ROS 2-compatible robotic systems, allowing researchers to focus on higher-level robotics problems instead of repeatedly developing and integrating custom hardware infrastructures.

2. Design

The comparison of representative commercial and open-source platforms presented in Table 1 reveals that existing solutions occupy different points in the design space, with trade-offs between cost, portability, modularity, sensing capabilities, and payload. Rather than maximizing a single performance metric, the objective of SKD-1 was to achieve a balanced platform suitable for robotics research while remaining affordable and reproducible. Based on this analysis, Table 2 provides a comprehensive list of the specific requirements that guided the design choices for our UGV.

Table 2. Design requirements.

Size and weight	
Handling	Manageable by 1 person (weight < 10 kg)
Transport	Fit the trunk of a medium-sized car
Speed and performance	
Working speed	Up to 1.0 m/s
Operating time	Up to 2 h
Terrain	Indoor and outdhoor (design target)
Turning radius	Turn in-place
Design	
Durability	Rugged and weatherproof
Sensors	Easily to interchange or replace
Accessibility	Easy access to batteries, cabling and electronics
Repairability	Easy to repair or replace parts
Operation	
Teleoperation	Manual control with standard gamepad
Ground Station	Telemeter vehicle states, parameters and mission data
Drivers and API	Easy to develop and integrated via ROS 2
Navigation	Integrated without major development
Components	
Onboard computer	Single board computer powered via 12 V battery and Ubuntu "Noble" 24.04 compatible
IMU	USB 3-axis accelerometer, gyroscope and magnetometer
GNSS	USB real-time kinematic (RTK) centimeter-level receiver
LiDAR	USB, 360° and up to 10 m distance sensing

To achieve these goals, we prioritized affordability, simplicity, and modularity in our component choices, while also minimizing the need for custom printed circuit boards (PCBs) fabrication and leveraging ROS 2 compatibility for efficient software implementation.

The electronic components were selected strategically, partly by recycling preexisting hardware in our laboratory and partly by choosing the remaining hardware carefully to fulfill the aforementioned requirements. A condensed Bill of Materials can be found in Table A1; however, all sensors can be easily substituted for different ones. By prioritizing compatibility with ROS 2, we reduce the time and effort required to get the UGV up and running, making it accessible to a wide range of users and applications. A simplified connection diagram showing the functional blocks and signal paths can be seen in Figure 1. For better visualization and fine details, the reader can refer to a wiring schematic provided in the Supplementary Materials.

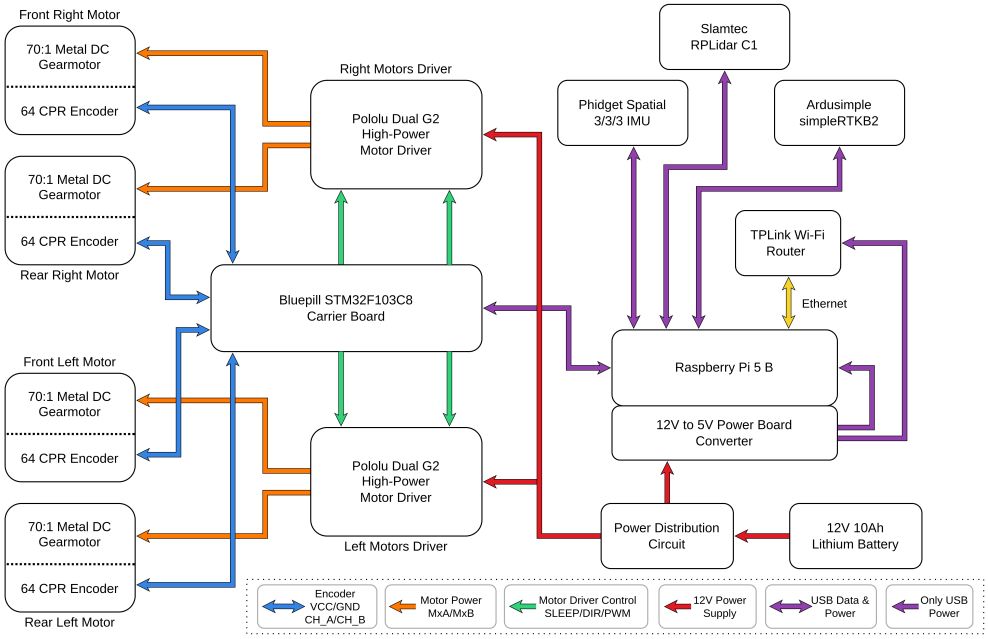


Figure 1. SKD-1 simplified connection diagram.

The SKD-1 is designed to support operation in both indoor and outdoor environments and moves using its four independent wheels. Since each motor is mounted directly on the chassis, it is really simple to build. It can be operated manually using a standard gamepad controller connected to either the UGV or to a computer acting as a ground control station (GCS). Furthermore, it can be configured to operate in autonomous mode using ROS 2 navigation algorithms. This brings flexibility and allows potential users to cover a wide range of use-cases. We want the UGV to be useful for diverse tasks, so we tried to meet the following requirements:

- Simple, durable and modular design.
- Lightweight, compact and easy to program.
- Capable of operating in indoor and outdoor environments.
- Sensors and hardware integrated with ROS 2 for out-of-the-box compatibility.

The remainder of this section will describe the design and construction of the chassis, the wiring of the power module, motors, sensors, communication system, and software.

2.1. Chassis

The chassis is built using a single laser-cut 400 mm × 250 mm × 5 mm acrylic sheet. The side and top covers are made of 4 mm laser-cut and folded acrylic, each consisting of a single piece. These covers are optional, but provide the SKD-1 with the top-plate for mounting sensors and offer some dust and weather protection. Acrylic sheets were chosen for the base and cover pieces primarily for practical manufacturing considerations rather

than rigorous structural optimization: acrylic is electrically insulating and easy to handle and machine, allowing rapid prototyping, and is low-cost, with sheets already available from a previous platform prototype [23], without sacrificing reliability. Users seeking higher structural strength or durability may consider substituting the acrylic base with an aluminum base of the same shape. All the mounting parts are 3D-printed in polyethylene terephthalate glycol (PETG) using fused deposition modeling (FDM) technology. If FDM is unavailable or if reducing the total cost is necessary, these parts can be replaced by anchors directly attached to the acrylic base. This combination of acrylic and 3D-printed parts makes the platform easy and inexpensive to manufacture, construct and assemble.

2.2. Power

The SKD-1 is powered by a 12 V lithium sealed battery pack. The battery has a nominal voltage of 12 V (operating range 10.3–12.6 V), a nominal capacity of 10 Ah (120 Wh), a maximum discharge current of 10 A, and a rated cycle life of 2000 cycles. The exact internal cell configuration is not disclosed by the manufacturer. A typical operating time of approximately 2 h can be achieved on a full charge. As a baseline reference, the platform drew approximately 2 A with all sensors active and the drive motors commanded at maximum speed while the vehicle was elevated with the wheels free of ground contact; this figure reflects electronics and unloaded-motor consumption only and does not include the additional current drawn under normal driving load and traction.

The motor drivers draw power directly from the 12 V battery and regulate the voltage supplied to the motors via pulse-width modulation (PWM). The onboard computer is powered through a Yahboom Power Supply Expansion Board, which converts the battery voltage to a regulated 5 V supply (up to 5 A, 25 W) via USB-C Power Delivery, satisfying the Raspberry Pi 5 power requirements. Most sensors and the low-level controller use USB connections for both power delivery and data communication with the onboard computer. The onboard router is powered from the auxiliary 5 V output of the expansion board.

The power system also includes an external power switch and an emergency push-button located on the top panel for easy access, together with a voltage display mounted on the rear panel. Both panels can be 3D printed, and the battery is secured to the chassis using custom brackets that facilitate installation and replacement.

2.3. Drive System

The drive system is composed of four Pololu 37D metal gearmotors driven by two Pololu 18V18 Dual G2 high power motor drivers mounted on the acrylic base. Each of these gearmotors is a 12 V brushed direct current (DC) motor with a 70:1 metal gearbox and an integrated 64 CPR quadrature encoder for feedback. The four independent motors are mounted using stamped aluminum brackets with a complementary 3D-printed enclosure that provide weatherproofing and shock protection. Each motor is coupled to a 120 mm all-terrain DAGU Wild Thumper wheel through a hexagonal adapter compatible with the motor shaft. Each motor driver controls two motors on the same side of the vehicle. Motor speed and direction of rotation are regulated using a combination of PWM and digital signals.

2.4. Low-Level Controller

A Blue Pill development board (STM32F103C8T6) implements closed-loop speed control of the gearmotors by processing feedback from the quadrature encoders and generating the corresponding control signals for the motor driver modules. Each encoder outputs two quadrature signals (`ENC_A` and `ENC_B`), and each channel of the Pololu Dual G2 High-Power Motor Driver is controlled through three signals: a `PWM` signal that sets

motor speed, a digital `DIR` signal that selects the direction of rotation, and a `SLEEP` signal that enables or disables the driver.

All eight quadrature channels are handled via general purpose input-output (GPIO) external interrupts, with position decoding performed entirely in software using a 4×-resolution lookup table. The eight encoder pins were deliberately assigned so that each occupies a distinct external interrupt line number (`EXTI8` through `EXTI15`), which prevents line-sharing conflicts on this microcontroller family and allows all four encoders to be serviced independently and concurrently. A separate hardware timer (`TIM2`) is reserved exclusively for a 50 Hz periodic interrupt that executes four independent control loops sequentially. The control-loop function processes the encoder measurements and the wheel-speed reference to compute the corresponding PWM duty cycle required to minimize the tracking error using a proportional-integral-derivative (PID) control algorithm.

The Blue Pill provides a USB serial communication interface for receiving wheel-speed reference commands and reporting encoder position and velocity measurements. The pin mapping between the Blue Pill, the encoders, and the motor drivers is shown in Figure 2. This low-level control architecture builds on our earlier work on ROS-compatible skid-steer platforms [23,24], and the full firmware source is available in the Supplementary Material.

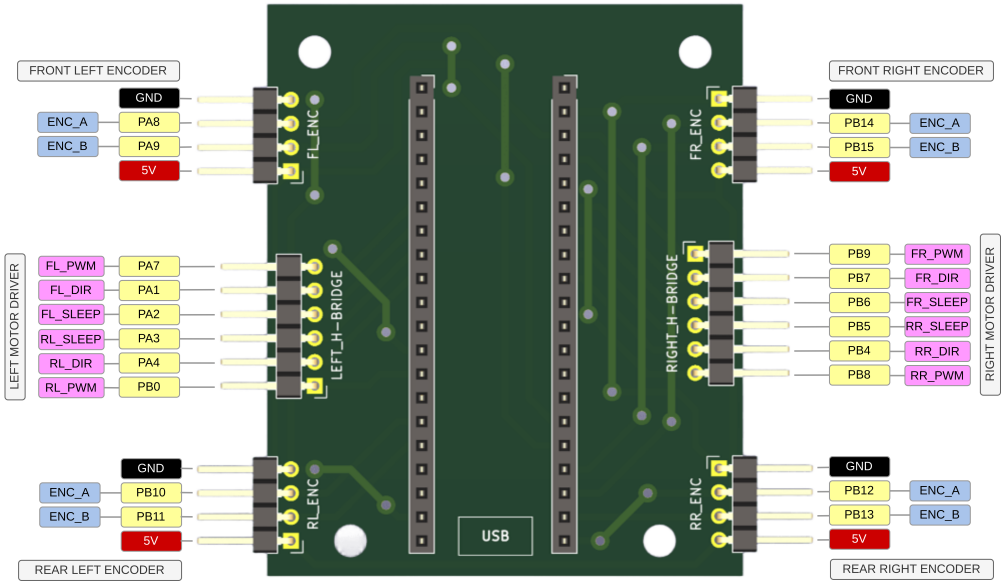


Figure 2. Low-level controller board pinout.

2.5. Wiring

The wiring was designed to ensure modularity and ease of maintenance while minimizing assembly cost. Signal connection between the quadrature encoders, motor drivers, and the low level controller are implemented using standard 0.1″ (2.54 mm) DuPont connectors. Power connections are handled separately using screw terminals. Each motor is connected to its corresponding motor driver through screw terminal block, while the motor drivers receive power via 2.5 mm^2 cables to support the required current levels. The power routing is implemented using aerial terminal blocks that form a centralized power bus. A voltmeter is connected to this bus to monitor the system supply voltage during operations. A power module board with DC-DC step-down converter is used to derive regulated 5 V from the main (unregulated) 12 V supply. This board powers the onboard computer and peripheral devices, which are connected through USB interfaces. The system includes a main power switch and a secondary selector that enables switching between normal operation and battery charging configurations. An external charger can be connected through a 2.1 mm barrel jack to recharge the battery. Additionally, a latching

emergency stop button is incorporated to allow immediate disconnection of the system power in case of fault conditions. Sensors are powered at 5 V and interfaced primarily via USB connections. USB hubs are used to expand the available ports of the single-board computer, facilitating integration and future upgrades (x86-based Mini-ITX, for example).

2.6. Computer and Communications

The system middleware runs ROS 2 and is executed in a Raspberry Pi 5 single-board computer located on the mobile platform. The single-board computer has a quad-core 64-bit Arm Cortex-A76 CPU running at 2.4 GHz, 8 GB of RAM, dual-band 802.11ac Wi-Fi, and four USB ports. The Raspberry Pi 5 runs Ubuntu 24.04 and is installed on an NVMe drive using a Pimoroni NVMe Base. This provides dramatically faster storage than micro-SD cards, cutting boot and load times to seconds. The single-board computer receives information about the position of the vehicle from the global navigation satellite system (GNSS) receptor, the neighborhood of the vehicle from the light detection and ranging (LiDAR) and the attitude of the vehicle from the inertial measurement unit (IMU). The GNSS receptor, the IMU and the LiDAR are connected with the single-board computer through USB ports.

A 2.4 GHz Wi-Fi router is used to transmit telemetry data from the robotic platform to the ground station computer. The platform can also be manually operated using a standard joystick connected to the base station. The Wi-Fi connection is further employed to send `Twist` commands (linear and angular velocities) to the onboard computer, allowing the navigation and guidance systems of the mobile platform to be overridden in critical situations.

If the ROS 2 navigation stack is properly configured, the vehicle is capable of receiving indoor and outdoor waypoints from the ground station to the robotic platform and traverse through them autonomously.

2.7. Sensors

Mobile robotic platforms need to be located in space, providing information about their position and attitude, as well as acquiring information about their surroundings. These goals are achieved by combining different sensors. In order to provide out-of-the-box indoor and outdoor autonomous operation using ROS 2 packages, we needed at least a global navigation satellite system receptor, an inertial measurement unit and a LiDAR. We chose the sensors that provide the best performance while preserving the cost-effective nature of the UGV.

Position estimation in outdoor environments is performed using an ArduSimple simpleRTK2B GNSS receiver based on the u-blox ZED-F9P, together with a u-blox multi-band GNSS antenna. The ZED-F9P supports dual-band (L1/L2, including E5b) multi-constellation tracking and provides a configurable navigation update rate of up to 10 Hz. When operated in RTK mode with external correction data, the receiver achieves centimeter-level positioning accuracy and interfaces directly with the onboard computer via USB.

An IMU was incorporated to estimate the attitude and motion of the mobile platform by providing three-axis measurements of linear acceleration, angular velocity, and magnetic field intensity. A PhidgetSpatial Precision 3/3/3 sensor was selected due to its favorable cost-to-performance ratio and suitability for integration. This IMU integrates a high-resolution 3-axis accelerometer (±16 g), a 3-axis gyroscope (±2000 °/s), and a 3-axis magnetometer (±8 G), and features an internal temperature stabilization circuit to mitigate thermal effects on sensor readings. The device has built-in support for AHRS and IMU algorithms. Native ROS 2 support is available, facilitating system integration, and the sensor interfaces directly with the onboard computer via USB.

Distance sensors not only aid to locate the robot in space but also warn about the presence of obstacles. The mobile platform employs a LiDAR to gather the distances to objects (or obstacles) that surround it, build maps and navigate the vehicle. An RPLiDAR C1 was chosen because of its low cost and specifications: It can perform 360-degree scans within a 12 m range with 0.2 cm distance resolution, and it can be configured for a 2–10 Hz scanning rate (that's for an entire rotation) with 1-degree angular resolution and it can also be configured for 4000–8000 Hz sampling frequency (that is the number of measurements), and it can work in all kinds of indoor environments without direct sunlight. It can power and communicate directly over the serial pins, but we use a USB cable and connect it to a USB-serial converter.

Finally, the robotic platform includes one quadrature encoder for each wheel. It provides a resolution of 64 counts per revolution of the motor shaft, which corresponds to 4480 counts per revolution of the gearbox's output shaft.

2.8. Software Architecture

To make the SKD-1 platform compatible with the ROS 2 ecosystem, the software architecture was designed around the standard integration point provided by the `ros2_control` framework [25]. Rather than developing a custom control architecture, the SKD-1 platform exposes its low-level controller through a `hardware_interface`, which provides the standardized command and state interfaces expected by `ros2_control`, abstracting the underlying communication protocol or embedded controller implementation. This allows the platform to appear to the remainder of the ROS 2 ecosystem as a standard mobile robot. Consequently, the only software component developed specifically for the SKD-1 platform is its hardware interface named `SKDHardware`. The high-level software stack is then built by configuring existing ROS 2 packages for the platform.

Figure 3 summarizes the resulting ROS 2 node and topic graph for the manual teleoperation configuration, with auxiliary nodes omitted for clarity. Multiple command-velocity sources, including joystick teleoperation (`/cmd_vel_joy`), keyboard teleoperation (`/cmd_vel_key`), generic velocity commands (`/cmd_vel`), and autonomous navigation (`/cmd_vel_nav`), publish to independent topics that are arbitrated by a `twist_mux` instance configured for the platform. The selected command is forwarded to `/skd1_base_controller/cmd_vel`, where it is processed by the `diff_drive_controller` instance named `skd1_base_controller`. Platform-specific parameters, including wheel radius, wheel separation, joint names, controller update rates, velocity limits, and odometry parameters, are provided through the controller configuration files. The controller converts the commanded body velocity into left and right wheel velocity references. Through the platform-specific controller configuration, these references are assigned to the front and rear wheels on each side and transmitted to the `SKDHardware` hardware interface through the standard `CommandInterface`. The hardware interface forwards these wheel velocity references to the low-level controller. The `skd1_base_controller` also publishes wheel odometry, while the executed velocity command is republished on `/skd1_base_controller/cmd_vel_out` for monitoring and debugging. Conversely, wheel encoder positions and velocities received from the low-level controller are exposed through the `StateInterface`. These interfaces are consumed by the `joint_state_broadcaster`, which publishes `/joint_states` and `/dynamic_joint_states`. These topics are then used by `robot_state_publisher`, together with the robot description defined in the URDF model, to publish the complete kinematic tree on `/tf`.

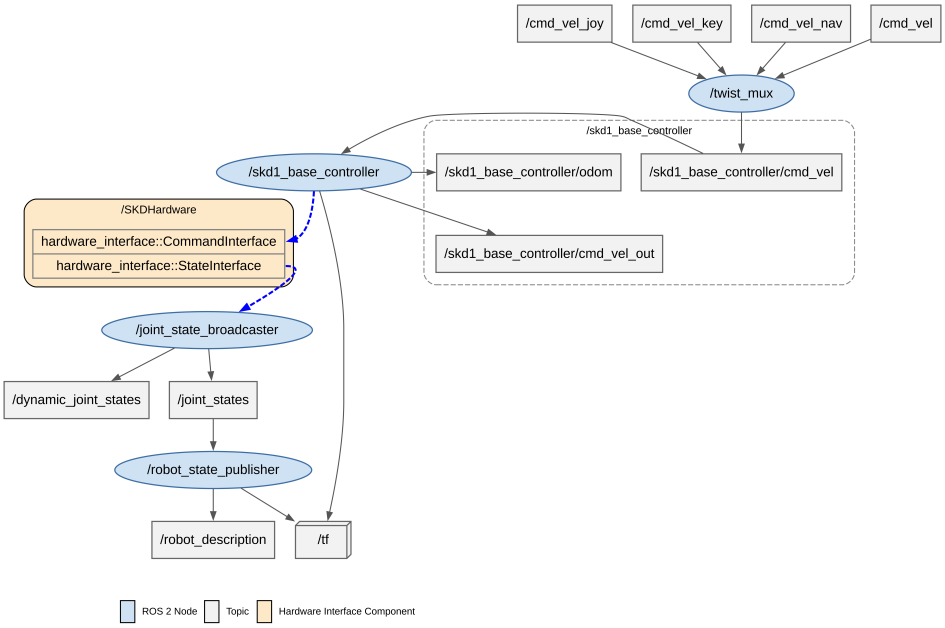


Figure 3. Software architecture of the SKD-1, showing the ROS 2 node graph. Arrows indicate the direction of data flow. The SKDHardware closes the loop using ROS 2 Control interfaces.

The implementation of `SKDHardware` and the corresponding Blue Pill firmware builds upon our previous work on ROS-compatible skid-steer platforms [23,24]. Communication between both components is performed over a USB serial link at 115200 baud using an HDLC-style framing protocol with byte stuffing and a CRC-CCITT checksum for error detection. A compact set of command codes is used to exchange data between the two layers, summarized in Table 3. As a safety measure, if no wheel-velocity command is received for 500 ms, all four PID controllers are automatically reset and the motors are stopped, preventing runaway motion in the event of a communication failure between the onboard computer and the microcontroller.

Table 3. Serial command codes used by the host-microcontroller HDLC protocol.

Code	Function
`0x0004`	Read wheel position and velocity for all four wheels
`0x0005`	Set the commanded wheel velocities
`0x0006`	Reset odometry: clears encoder position and PID controller state
`0x0010`	Set PID gains for one wheel or all wheels
`0x0105`	Set raw PWM output directly (open-loop, bypassing the PID controller)
`0x00AA`	Echo/loopback frame, used for communication link testing

Higher-level functionality is provided entirely by standard ROS 2 packages. In the SKD-1 platform, the `robot_localization` package [26] is configured to fuse wheel odometry with IMU measurements for state estimation, `slam_toolbox` [27] is used for indoor mapping and localization, and the *Navigation 2* (Nav2) [28] stack provides autonomous waypoint navigation. Since these packages interact exclusively through the standard ROS 2 interfaces exposed by the platform, they remain independent of the underlying hardware implementation. Their deployment and operation are described in Section 4.

3. Build Instructions

The mechanical structure was designed as a modular assembly, allowing each component to be manufactured independently and facilitating straightforward replacement

when required. Consequently, the assembly procedure is not strictly sequential and may be performed in any convenient order.

The assembly begins with the main structural module. The primary acrylic base serves as the mechanical support for the drive system and other components, including the Raspberry Pi single-board computer. Aluminum L-brackets used for mounting the DC motors are fixed to the underside of the acrylic base using M3 Allen screws and corresponding nuts. Each gearmotor is then secured within its respective bracket, ensuring correct alignment with the wheel axis. Motor protectors are subsequently installed to provide mechanical protection during operation, as shown in Figure 4.

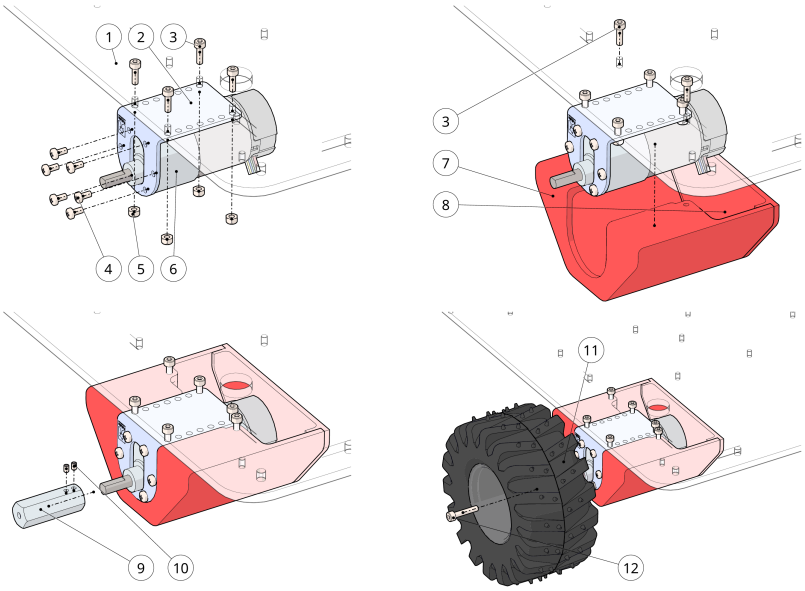


Figure 4. Assembly of the main structural module, showing the acrylic base (**1**) with aluminum L-brackets (**2**) secured using M3 × 10 mm socket head cap screws (**3**) and M3 nylon lock nuts (**5**). DC gearmotors (**6**) are mounted to the brackets with M3 × 5 mm screws (**4**), and motor protectors (**7**,**8**) are fixed to the base using additional screws (**3**). A 12 mm hex wheel adapter (**9**) is attached to the motor shaft with M3 × 4 mm set screws (**10**), and coupled to the wheel (**11**) using an M4 × 8 mm screw (**12**).

After the motor mounts are secured, the wheels are attached to the motor shafts using 12 mm hexagonal wheel adapters designed for 6 mm shafts. Proper wheel alignment is verified to minimize mechanical friction and ensure smooth translational motion. Figure 5 shows the completed assembly of the SKD-1 base.

Once the base assembly is completed, the protective structure is installed. This structure consists of corner mounting elements and protective chassis walls. The corner components are fixed to the acrylic base and serve as mechanical supports for the vertical protective walls. The chassis walls are then mounted onto the corner elements, forming a rigid enclosure around the platform. This configuration provides mechanical protection for the internal components while maintaining access to the base structure. The assembled protective structure is shown in Figure 6.

The rear panel is mounted as part of the protective enclosure and serves as the mechanical support for the external power interface components. The panel is fixed to the previously installed chassis structure, completing the rear side of the enclosure. A battery voltage monitor, an operational selector switch, a protection fuse, and a DC input connector for battery charging are mounted on the rear panel. The switch allows the operating mode to be selected between the off state, battery charging mode, and battery connected mode. The assembled rear panel with the mounted components is shown in Figure 7.

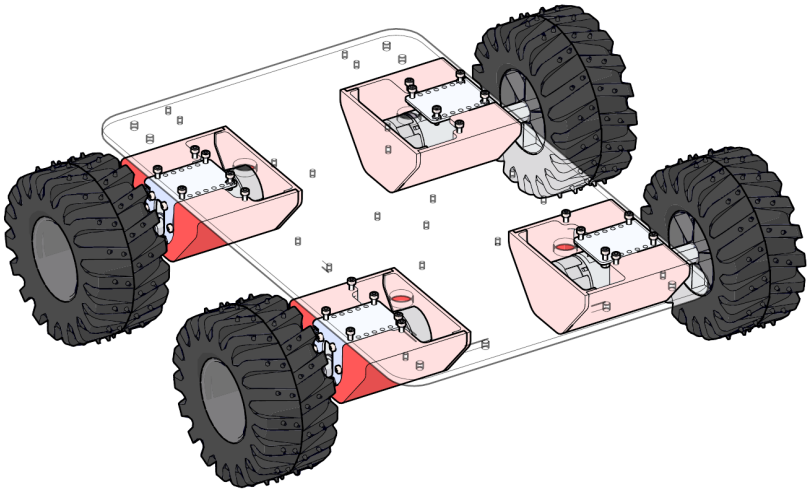


Figure 5. Complete assembly of the SKD-1 base.

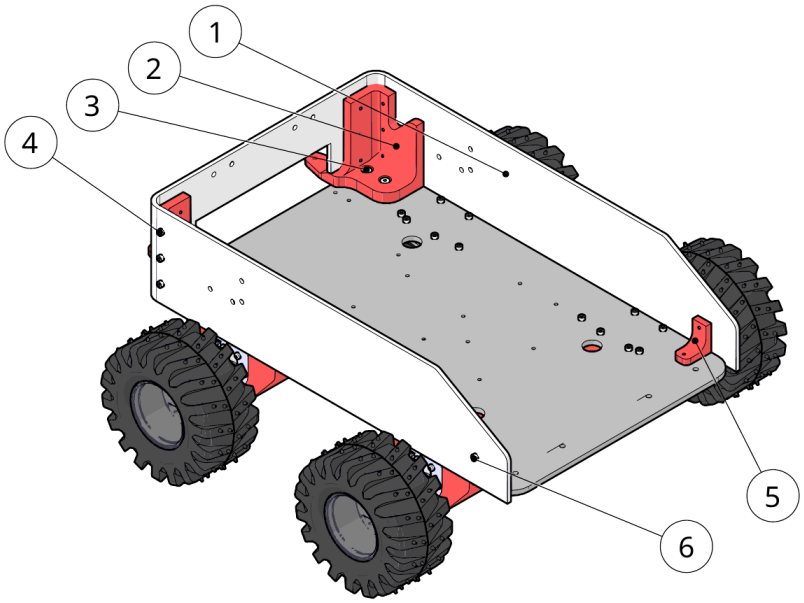


Figure 6. Assembly of the protective structure (**1**), showing the corner mounting elements (**2**) fixed to the acrylic base using M4 × 20 mm socket countersunk head screws and M4 nylon lock nuts (**3**). The side protective panel is attached to the corner mounts with M3 × 12 mm socket head cap screws (**4**). Front clamps (**5**) are used to reinforce the structure, being secured to both the panel and the base with M3 × 10 mm socket head cap screws (**6**).

The on–off switch and the emergency stop switch are mounted on the upper rear panel, which is installed as part of the protective enclosure and is mechanically independent from the rear panel. These components provide direct user access for system power control and emergency interruption. Their placement on the upper rear panel ensures adequate visibility and accessibility during operation. The assembled configuration is shown in Figure 8.

Figure 9 shows the electrical connection diagram of the back panel, including the battery input, charging connector, operational switches, protection fuse, voltage monitor, and the 12 V power distribution bus. The back panel circuit provides the main power management and operating mode selection for the platform. A selector switch allows the system to operate in two modes: charge mode and battery connected for normal operation mode. In charge mode, the battery is connected only to the external DC input, allowing safe charging without powering the vehicle electronics. When the selector is set to normal

operation mode, the battery is connected to the system, which becomes ready for operation. In this mode the vehicle must then be powered on using the main switch located on the upper rear panel.

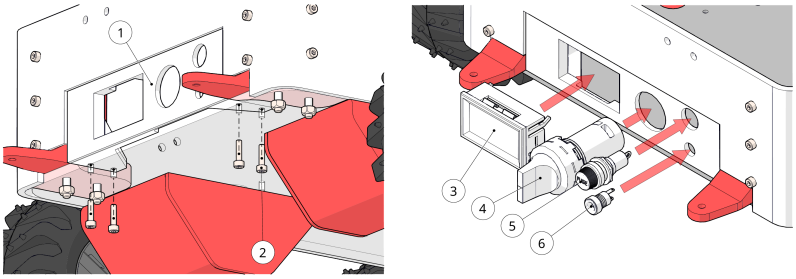


Figure 7. Rear panel assembly (**1**) mounted to the acrylic base using M3 × 12 mm socket head cap screws (**2**), showing the installation of the battery voltage monitor (**3**), operational mode selector switch (**4**), protection fuse (**5**), and DC input connector for battery charging (**6**).

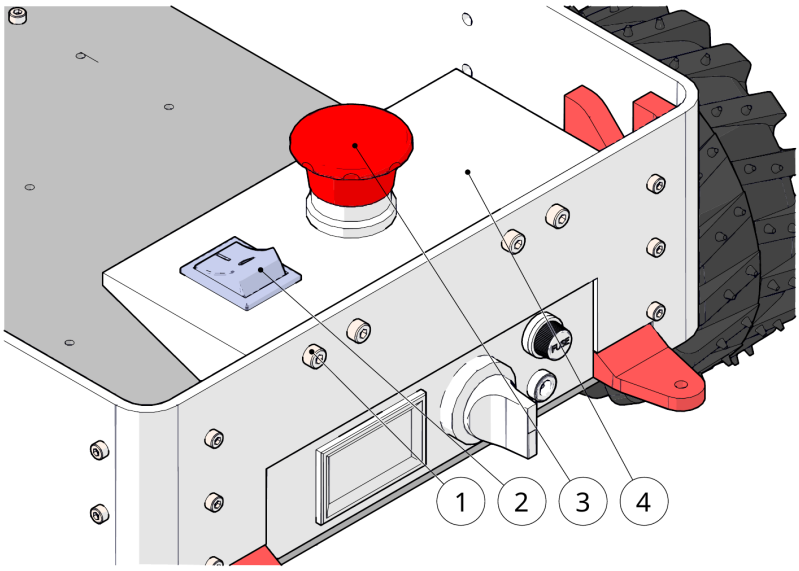


Figure 8. Installation of the upper rear panel (**4**), mounted to the protective structure using M4 × 12 mm socket head cap screws (**1**), showing the integration of the on–off switch (**2**) and the emergency stop switch (**3**).

Figure 10 presents the main electronic components used in the platform. The figure is composed of four subfigures illustrating the individual modules prior to integration into the system. These components include the STM32 Blue Pill microcontroller board with its custom printed circuit board, the motor driver modules, the lithium battery used as the power source, and the Raspberry Pi single-board computer. Each module is shown separately to clearly identify the hardware elements that constitute the electronic subsystem of the platform.

The top plate is the uppermost structural element of the enclosure. It provides mechanical support for the platform's external communication and positioning devices.

A real-time kinematic global navigation satellite system (RTK-GNSS) antenna, a radio-link module, and a Wi-Fi router are installed on the top plate. The RTK-GNSS antenna sits on the upper surface to ensure unobstructed signal reception. The radio-link module and the Wi-Fi router are mounted on the underside, providing RTK corrections and wireless communication capabilities. The assembled top plate with the installed devices is shown in Figure 11.

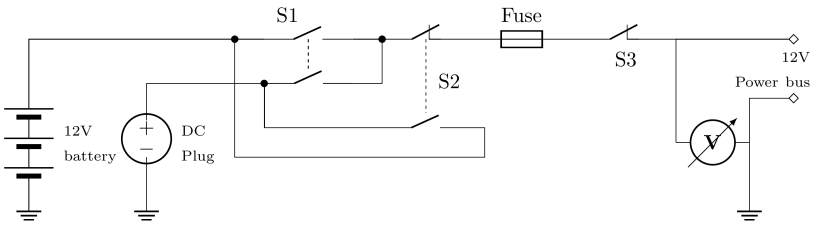
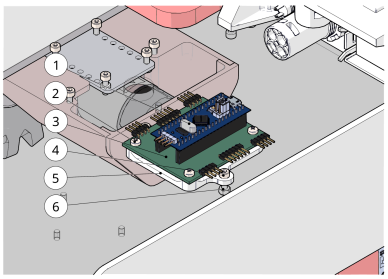
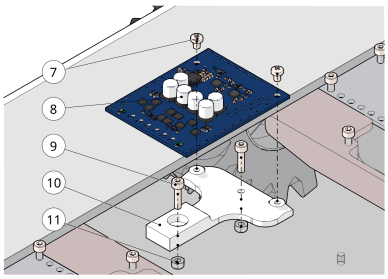


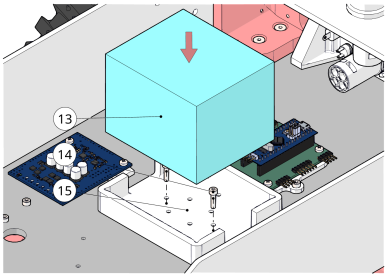
Figure 9. Electrical connection diagram showing the 12 V power distribution system. The **12 V battery** and **DC input plug** are connected to a DPDT main switch **S1**, whose output feeds the normally closed contact of a selector switch **S2**. The normally open contact of **S2** provides a bypass between the battery and the DC input, enabling a charging mode that isolates the rest of the system, while the normal position allows standard operation. The circuit then passes through a protection **fuse** and a normally closed emergency stop switch **S3**. A voltmeter is connected in parallel to monitor the supply voltage, and the resulting output provides the 12 V power bus.



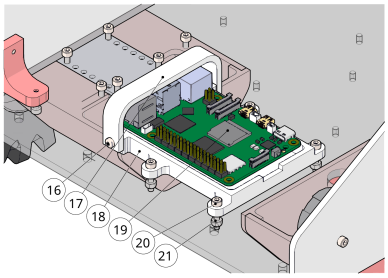
(a) STM32 Blue Pill microcontroller mounted on its printed circuit board



(b) H-bridge motor driver module



(c) Lithium battery



(d) Raspberry Pi single-board computer

Figure 10. Electronic components assembly showing: (**a**) the STM32 Blue Pill microcontroller (**1**) mounted on a signal distribution PCB (**3**), which is fixed to a 3D–printed support (**5**) using M3 × 5 mm screws (**2**), and the support attached to the acrylic base with M3 × 12 mm socket head cap screws (**4**) and M3 nylon lock nuts (**6**); (**b**) the motor driver (**8**) mounted on its base (**10**) with M3 × 5 mm screws (**7**), and the base fixed to the acrylic with M3 × 12 mm socket head cap screws (**9**) and M3 nylon lock nuts (**11**); (**c**) the lithium battery (**13**) placed on a 3D–printed bracket (**15**), which is secured to the acrylic base with M3 × 12 mm socket head cap screws (**14**); and (**d**) the Raspberry Pi (**19**) mounted on its base (**18**) and retained by a bracket (**17**) fixed with M3 × 5 mm screws (**16**), with the base attached to the acrylic using M3 × 16 mm socket head cap screws (**20**) and M3 nylon lock nuts (**21**).

Figure 12 shows the installation of the IMU, the LiDAR sensor, and the top plate of the platform. The figure also illustrates the mounting of the hinges that allow the top cover to be opened for access to the internal components of the vehicle. These hinges provide mechanical support while enabling convenient maintenance and inspection of the internal electronics.

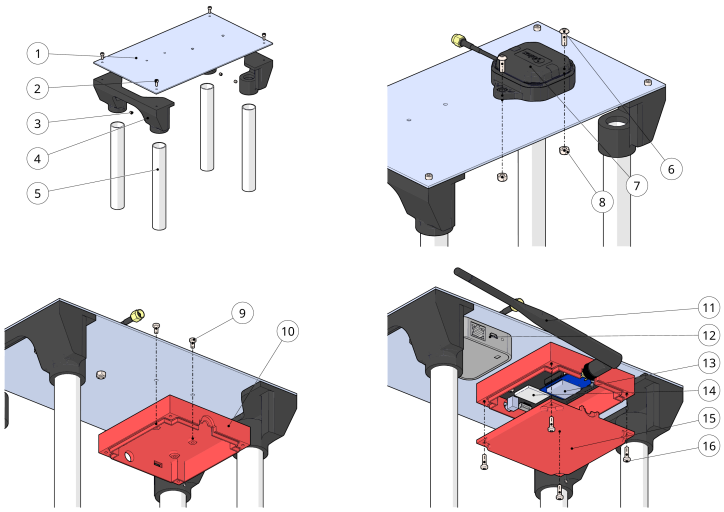


Figure 11. Top plate assembly showing an aluminum ground plane (**1**) mounted on 3D–printed supports (**4**) using M3 × 10 mm socket head cap screws (**2**), with PVC tubes (**5**) secured by M4 × 4 mm set screws (**3**). The RTK-GNSS antenna (**7**) is fixed to the ground plane with M4 × 12 mm button head cap screws (**6**) and M4 nylon lock nuts (**8**). On the underside, the RTK enclosure (**10**) is attached using M3 × 5 mm screws (**9**), housing the RTK module (**14**) and connected to an RF antenna (**11**) via the long–range module (**13**). A Wi–Fi router (**12**) is also mounted underneath. The enclosure is completed with a top cover (**15**) secured by M3 × 10 mm screws (**16**).

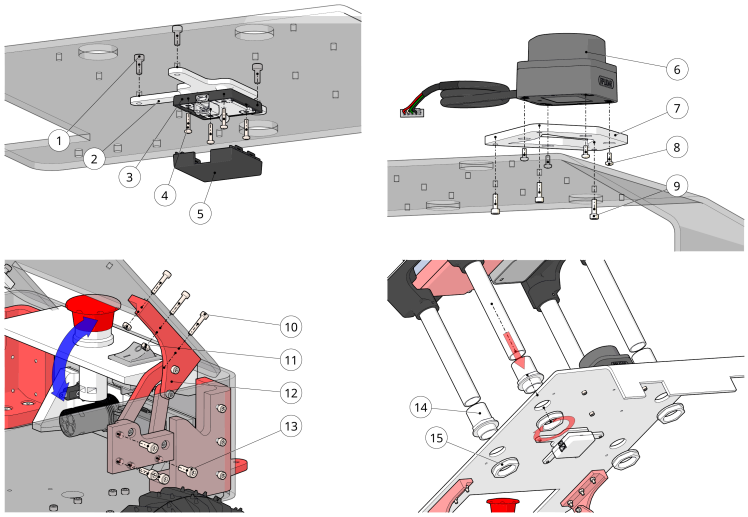


Figure 12. Installation of the sensing and upper structural assembly, showing the IMU mounted on its support (**1**–**5**) and the LiDAR sensor (**6**–**9**) fixed to the upper protective panel. The IMU assembly consists of a base (**3**) mounted to the support adapter (**2**) using M2 × 10 mm screws (**4**) and enclosed by a snap–fit cover (**5**), with the adapter fixed to the top panel using M3 × 10 mm socket head cap screws (**1**). The LiDAR sensor (**6**) is attached to its mounting base (**7**) using M2.5 × 5 mm screws (**8**), and the base is secured to the top panel with M3 × 10 mm socket head cap screws (**9**). The upper protective panel is connected to the hinge assembly (**12**) using M3 × 16 mm screws (**10**) and M3 nylon lock nuts (**11**), while the hinges are fixed to the side structure with M4 × 12 mm socket head cap screws (**13**). Finally, the top plate support tubes are press–fitted into electrical connectors (**14**) and secured to the panel with retaining nuts (**15**).

Figure 13 shows the final assembly of the platform, presenting the completed SKD-1 mobile robot after all mechanical and electronic components have been installed.

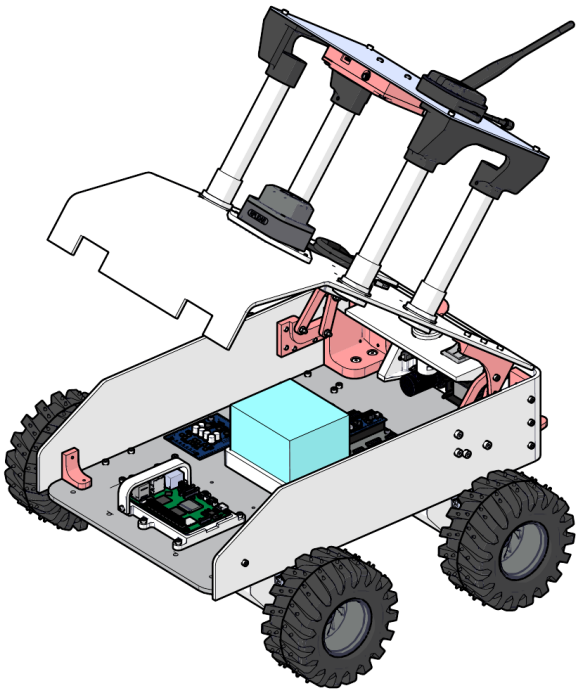


Figure 13. Final assembly of the SKD-1 platform showing the fully integrated mobile robot.

4. Operating Instructions

4.1. Pre-Deployment

- Charge the batteries
  - Mobile platform lithium battery to 12.5 V
  - Ground station laptop battery
  - Gamepad battery
  - RTK base powerbank (if using outdoor)
- Secure fully charged battery into the mobile platform
- Check all wires connections
  - Battery connectors
  - Blue Pill wires
  - Motor driver wires
  - Motor wires
  - Encoder wires
  - Sensors USB cables
  - Gamepad dongle connection
  - WiFi router ethernet and power cables
  - RTK-GNSS antenna (if needed)
- Check all mechanical parts are properly fastened
  - Wheels
  - Top cover
  - Sensor mounts

4.2. Ground Station Connection Check

- Turn on the SKD-1
- Check that the LiDAR is powered on
- Wait for Wi-Fi network and connect
- Open a terminal on the ground station laptop

- Configure the ROS 2 environment
  - On the SKD-1
    1. Log in via SSH
    2. Preferably launch `tmux`
    3. Run `source /opt/ros/jazzy/setup.bash`
    4. Source the SKD-1 workspace
    5. Execute:
       - For manual operation:
         `ros2 launch skd1_bringup bringup_joy_teleop_launch.yaml`
       - For indoor navigation:
         `ros2 launch skd1_bringup skd1_nav2_indoor_launch.py`
       - For outdoor navigation:
         `ros2 launch skd1_bringup skd1_nav2_outdoor_launch.py`
  - On the ground station laptop
    1. Run `source /opt/ros/jazzy/setup.bash`
    2. Confirm remote execution of `Twist` commands using the wireless gamepad and checking the `cmd_vel` topic
    3. Run the corresponding RViz launch file from the `skd1_visualization` package, enabling control, topic inspection and data visualization

4.3. Manual Operation

In manual operation mode, the user must launch `bringup_joy_teleop_launch.yaml` which is part of the `skd1_bringup` package. After successful launch, the SKD-1 UGV is teleoperated with velocity commands generated by a standard wireless gamepad. All commands are sent through the USB connected wireless gamepad receiver. The current configuration uses the R2 button as a safety enable button that must remain pressed in order for commands to be sent to the vehicle. Then, the user can choose to use either digital or the analogue stick to send the desired combination of linear and angular velocities. This allows the mobile platform to be directed to the front, back, turn right and turn left.

4.4. Autonomous Operation

In autonomous operation mode, the user must launch `skd1_nav2_indoor_launch.py` or `skd1_nav2_outdoor_launch.py`, located in the `skd1_bringup` package. Both launch files contain all the required ROS 2 components and configuration for proper operation. The SKD-1 moves based on closed-loop navigation algorithms available in the ROS Nav2 stack, which is configured to employ `Navfn` as the global planner and `RegulatedPurePursuit` [29] as the local controller. The navigation system computes the linear and angular velocities required to move the platform through space based on the waypoints, position, and attitude. Waypoint goals can be sent using RViz on the ground station or by sending a Pose message to the `/goal_pose` topic. When working indoors, the navigation system computes the position and attitude of the platform using the IMU, the LiDAR, and wheel odometry. This configuration employs `slam_toolbox` with a dual `robot_localization` setup to estimate the position and attitude of the platform and to fuse sensor measurements, respectively. Meanwhile, when working outdoors, the navigation system computes the position and attitude of the platform using the IMU, the GNSS, and wheel odometry. This configuration employs `navsat_transform` with a dual `robot_localization` setup to estimate the position and attitude of the platform and to fuse sensor measurements, respectively.

5. Validation

To verify the proper integration and operation of all system components of the SKD-1, a series of experiments were conducted. The validation focused on three main aspects: (i) wheel velocity tracking, (ii) odometry estimation, and (iii) integration with the standard ROS 2 navigation stack. The velocity tracking and odometry experiments were carried out in a controlled indoor environment using custom ROS 2 Python scripts that generated repeatable sequences of `Twist` commands. Wheel encoder feedback was analyzed to verify accurate tracking of the commanded wheel velocities, while the estimated odometry was compared against physically measured displacements to evaluate the accuracy of the platform kinematic model.

To demonstrate the compatibility of the SKD-1 platform with the ROS 2 ecosystem, additional indoor and outdoor experiments were conducted using the standard `slam_toolbox`, `navsat_transform`, `robot_localization`, and Nav2 packages. These experiments verified the correct integration of the onboard sensors, odometry estimation, localization, and motion control interfaces required for autonomous navigation, rather than evaluating the performance of the employed SLAM and navigation algorithms themselves.

5.1. SKD-1 Platform Specifications

Before presenting the experimental results, the main specifications of the SKD-1 platform are summarized in Table 4. These values correspond to the final assembled platform and were used to configure the robot description and motion control packages. The maximum linear and angular velocities reported were determined from the motor specifications and subsequently verified experimentally. Conservative limits were then adopted for the `skd1_base_controller` to ensure reliable operation during navigation experiments. The reported operating time reflects repeated operational use of the platform during indoor and outdoor testing with the complete sensor suite installed. Under these conditions, the platform consistently operated for approximately 1–2 h before battery recharging was required, depending on the usage conditions and payload.

Table 4. SKD-1 Technical Specifications.

Mechanical characteristics	
Dimensions (L × W × H)	410 mm × 260 mm × 370 mm
Total weight	6 kg
Wheel diameter	120 mm
Wheel separation	340 mm
Wheel base	210 mm
Ground clearance	88 mm
Speed and performance	
Maximum linear speed	0.8 m/s
Maximum angular speed	4.0 rad/s
Estimated payload	5 kg
Battery and Power System	
Battery	12 V, 10 Ah, Li-ion battery
Operating time	1 to 2 h (load dependent)
Motor operating voltage	12 V
Computer and sensors	
Onboard computer	Raspberry Pi 5 (8 GB RAM, 250 GB NVMe SSD)
IMU	PhidgetSpatial Precision 3/3/3
LiDAR	Slamtec RPLiDAR C1
GNSS	Ardusimple simpleRTK2B (u-blox ZED-F9P)
Software	Ubuntu 24.04, ROS 2 "*Jazzy Jalisco*"

5.2. Velocity Tracking and Odometry Calibration

To evaluate the performance of the motion control architecture, four experiments were conducted. The first two experiments assessed linear and angular velocity tracking, whereas the remaining two were designed to calibrate the platform kinematic model and evaluate the resulting odometry accuracy.

For the velocity tracking experiments, custom ROS 2 Python scripts continuously published `Twist` commands, updating the commanded velocity every 5 s according to predefined velocity profiles. For the linear velocity experiment, the commanded velocity was increased from 0 to 1.0 m/s in increments of 0.25 m/s. After reaching the maximum commanded velocity, the platform was brought to a complete stop before repeating the sequence symmetrically down to -1.0 m/s. Since the maximum linear velocity configured in the `skd1_base_controller` was 0.75 m/s, commands exceeding this value were automatically saturated. An analogous experiment was performed for angular velocity, where the commanded angular velocity was varied from 0 to ± 4.0 rad/s in increments of 1.0 rad/s. Likewise, the platform was commanded to stop before changing the direction of rotation. The controller saturated these commands according to the configured limit of 3.14 rad/s.

Figures 14 and 15 show the commanded velocities, the saturated commands generated by the `skd1_base_controller`, and the corresponding odometry estimates. In both experiments, the estimated linear and angular velocities closely followed the commanded references within the configured controller limits, confirming the correct operation of the complete motion control pipeline.

Figures 16 and 17 present the commanded and measured angular velocities of the four wheels during the linear and angular velocity experiments, respectively. Encoder feedback closely matched the commanded wheel velocities, indicating accurate actuator control with low steady-state error.

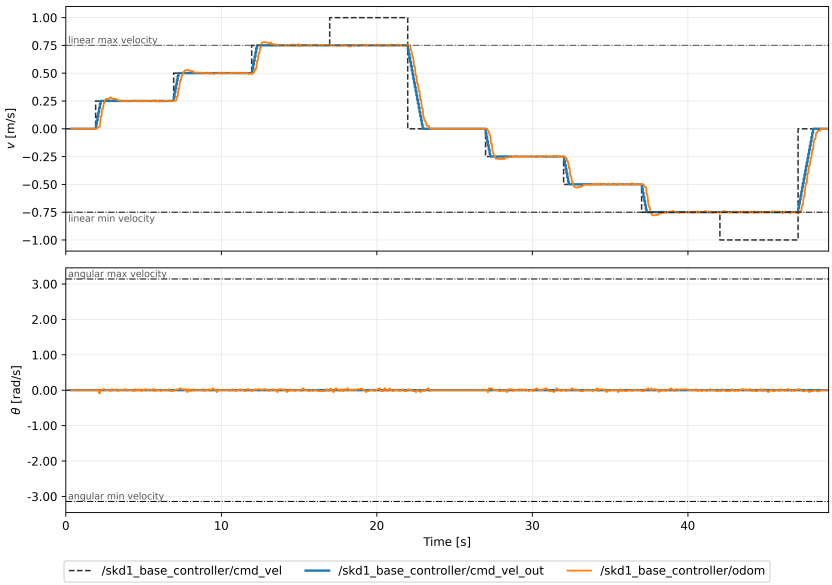


Figure 14. Linear velocity tracking during the predefined velocity profile experiment. The figure compares the commanded linear velocity (`skd1_base_controller/cmd_vel`), the saturated command generated by the `skd1_base_controller` (`skd1_base_controller/cmd_vel_out`), and the velocity estimated from wheel odometry (`skd1_base_controller/odom`).

The remaining two experiments were conducted to calibrate and evaluate the odometry of the SKD-1 platform. A custom ROS 2 Python script commanded the mobile platform to move at a constant linear velocity of 0.3 m/s for 9 s, resulting in a theoretical travel

distance of 2.7 m. The experiment was performed on a flat indoor surface to minimize wheel slip and other external disturbances. Physical markers were placed at the initial and final positions of the platform, and the traveled distance was measured manually using a measuring tape. This reference distance was compared with the displacement estimated by the `skd1_base_controller` odometry to quantify the odometry error.

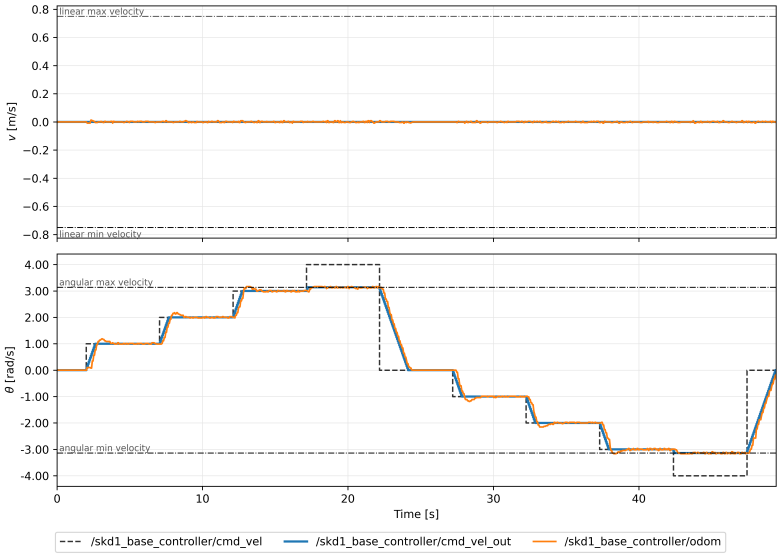


Figure 15. Angular velocity tracking during the predefined velocity profile experiment. The figure compares the commanded angular velocity (`skd1_base_controller/cmd_vel`), the saturated command generated by the `skd1_base_controller` (`skd1_base_controller/cmd_vel_out`), and the angular velocity estimated from wheel odometry (`skd1_base_controller/odom`).

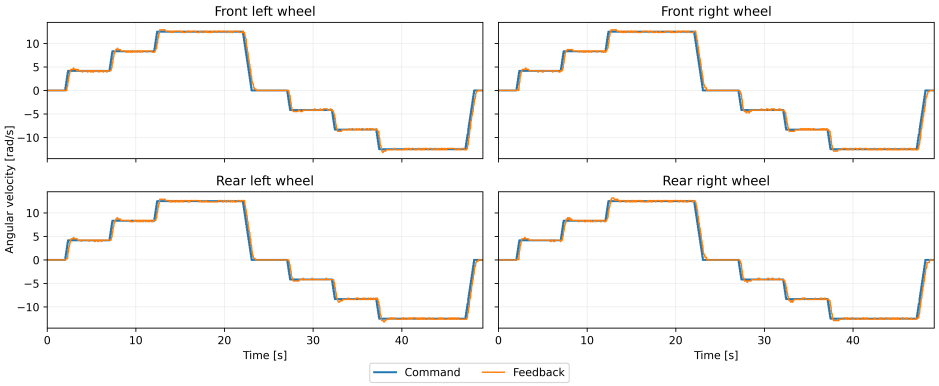


Figure 16. Commanded and measured wheel angular velocities during the linear velocity experiment. Encoder feedback is compared with the wheel velocity references for each of the four drive wheels.

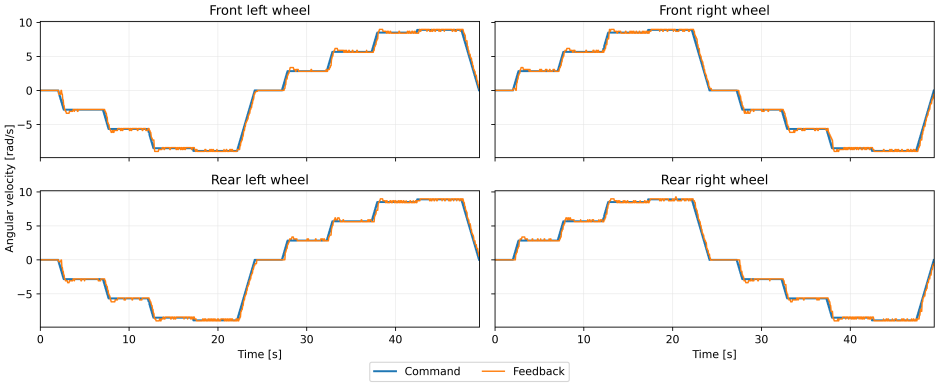


Figure 17. Commanded and measured wheel angular velocities during the angular velocity experiment. Encoder feedback is compared with the wheel velocity references for each of the four drive wheels.

The experiment was repeated ten times in the forward direction and ten times in the backward direction. For each trial, the relative distance error (e_d) was computed by comparing the measured displacement ($d_{measured}$) with the odometry estimate (d_{odom}). The mean relative error ($\overline{e_d}$) and the corresponding standard deviation (σ_d) were then calculated to evaluate the repeatability of the odometry estimation. The relative error, mean error, and standard deviation were computed according to Equations (1)–(3), respectively.

$$e_d = \frac{d_{odom} - d_{measured}}{d_{measured}} \tag{1}$$

$$\overline{e_d} = \frac{1}{n}\sum e_d \tag{2}$$

$$\sigma_d = \sqrt{\frac{\sum(e_d - \overline{e_d})^2}{n-1}} \tag{3}$$

Table 5 summarizes the results obtained from the forward and backward motion experiments. From these data, the mean relative error for forward motion is $\overline{e_{d,f}} = -3.05$ % with a standard deviation of $\sigma_{d,f} = 0.11\%$, whereas the backward trials resulted in a mean relative error of $\overline{e_{d,b}} = -0.43\%$ and a standard deviation of $\sigma_{d,b} = 0.17\%$. The relatively small standard deviations observed in both directions indicate good repeatability of the experiments, while the systematic difference between the forward and backward mean errors suggests the presence of a direction-dependent bias in the estimated traveled distance. These results were subsequently used to estimate the wheel radius correction factor of the platform.

Table 5. Measured distance ($d_{measured}$), odometry-estimated distance (d_{odom}), and relative error (e_d) for the forward and backward motion experiments.

Sample	Forward			Backward		
	$d_{measured}$ [m]	d_{odom} [m]	e_d (%)	$d_{measured}$ [m]	d_{odom} [m]	e_d (%)
1	2.792	2.707	−3.04	−2.712	−2.705	−0.25
2	2.790	2.706	−3.01	−2.721	−2.709	−0.42
3	2.787	2.702	−3.04	−2.721	−2.709	−0.42
4	2.798	2.711	−3.11	−2.724	−2.712	−0.44
5	2.793	2.707	−3.07	−2.715	−2.709	−0.21
6	2.788	2.707	−2.91	−2.716	−2.706	−0.38
7	2.792	2.707	−3.06	−2.704	−2.682	−0.82
8	2.790	2.702	−3.17	−2.714	−2.701	−0.46
9	2.790	2.710	−2.85	−2.711	−2.702	−0.32
10	2.797	2.707	−3.23	−2.714	−2.698	−0.57

The experimental results were used to estimate the `left_wheel_radius_multiplier` and `right_wheel_radius_multiplier` parameters of the `skd1_base_controller` package. A correction factor of 1.03 was adopted for both parameters. Three additional verification experiments were then conducted using the calibrated parameters, resulting in a mean relative error of -0.64% and a standard deviation of 0.0029%.

The fourth and last experiment was conducted to calibrate the rotational odometry. A custom ROS 2 Python script commanded the platform to rotate in place at a constant angular velocity of 0.785 rad/s for 8 s, resulting in a theoretical rotation of 360°. The final orientation of the platform was measured with respect to its initial heading, and the relative angular error (e_θ) was computed by comparing the measured rotation ($\theta_{measured}$) with the odometry estimated (θ_{odom}) by the `skd1_base_controller`. The experiment was repeated five times for both clockwise and counterclockwise rotations. The mean relative error and the corresponding standard deviation were then computed according to Equations (4), (5), and (6), respectively.

$$e_\theta = \frac{\theta_{odom} - \theta_{measured}}{\theta_{measured}} \tag{4}$$

$$\overline{e_\theta} = \frac{1}{n}\sum e_\theta \tag{5}$$

$$\sigma_\theta = \sqrt{\frac{\sum(e_\theta - \overline{e_\theta})^2}{n-1}} \tag{6}$$

Table 6 summarizes the results obtained from the clockwise and counterclockwise rotation experiments. Based on these data, the mean relative error for the clockwise turn is $\overline{e_{\theta,\mathrm{cw}}} = 38.81\%$ with a standard deviation of $\sigma_{\theta,\mathrm{cw}} = 0.0085\%$, whereas the counterclockwise trials resulted in a mean relative error of $\overline{e_{\theta,\mathrm{ccw}}} = 39.8\%$ and a standard deviation of $\sigma_{\theta,\mathrm{ccw}} = 0.0034\%$. The similar mean errors and the low standard deviations observed in both directions indicate a systematic error in the rotational kinematic model, which was subsequently used to estimate the wheel separation correction factor of the platform.

Table 6. Measured rotation angle ($\theta_{measured}$), odometry-estimated rotation angle (θ_{odom}), and relative error (e_θ) for the clockwise and counterclockwise rotation experiments.

	Clockwise			**Counterclockwise**		
Sample	$\theta_{measured}$ [°]	θ_{odom} [°]	e_θ (%)	$\theta_{measured}$ [°]	θ_{odom} [°]	e_θ (%)
1	257	360.6	40.30	257	359.0	39.71
2	260	359.6	38.30	256	358.9	40.18
3	260	359.8	38.40	257	359.5	39.90
4	260	359.7	38.35	257	359.7	39.98
5	259	359.2	38.68	257	357.9	39.27

The experimental results were used to estimate the `wheel_separation_multiplier` parameter of the `skd1_base_controller` package. A correction factor of 1.39 was adopted. Three additional verification experiments were then conducted using the calibrated parameter, resulting in a mean relative error of -0.62% and a standard deviation of 0.0038%.

5.3. Indoor SLAM and Waypoint Navigation

To evaluate the SLAM and navigation, a series of indoor experiments were conducted in an office environment using the `slam_toolbox` and Nav2 packages. Initially, the platform was commanded to follow multiple nearby waypoints to incrementally build a map of the environment. Subsequently, a final waypoint located approximately 25 m away was issued to evaluate long-range navigation, including global path planning, obstacle avoidance, and path-following performance.

Figure 18 shows the occupancy grid map generated during the experiment together with the platform's estimated trajectory and the commanded waypoints. The resulting map captures the main structural features of the office environment, while the overlaid trajectory illustrates the vehicle motion during map construction and the subsequent long-range navigation task.

Figure 19 presents the distance error between the estimated platform position and the active navigation goal throughout the experiment. The waypoint transitions are indicated by the labels W1–W6, allowing the navigation performance to be related to each commanded destination. The distance error decreases as the platform approaches each waypoint, reaching values close to zero before the next waypoint is commanded.

A synchronized recording of the experiment, combining a video of the platform operation, an RViz screen recording of the navigation process, and a Foxglove [30] playback of the recorded ROS 2 bag, is available at https://youtu.be/Ky_n3-d5krc (accessed on 17 August 2026) and in the Supplementary Material.

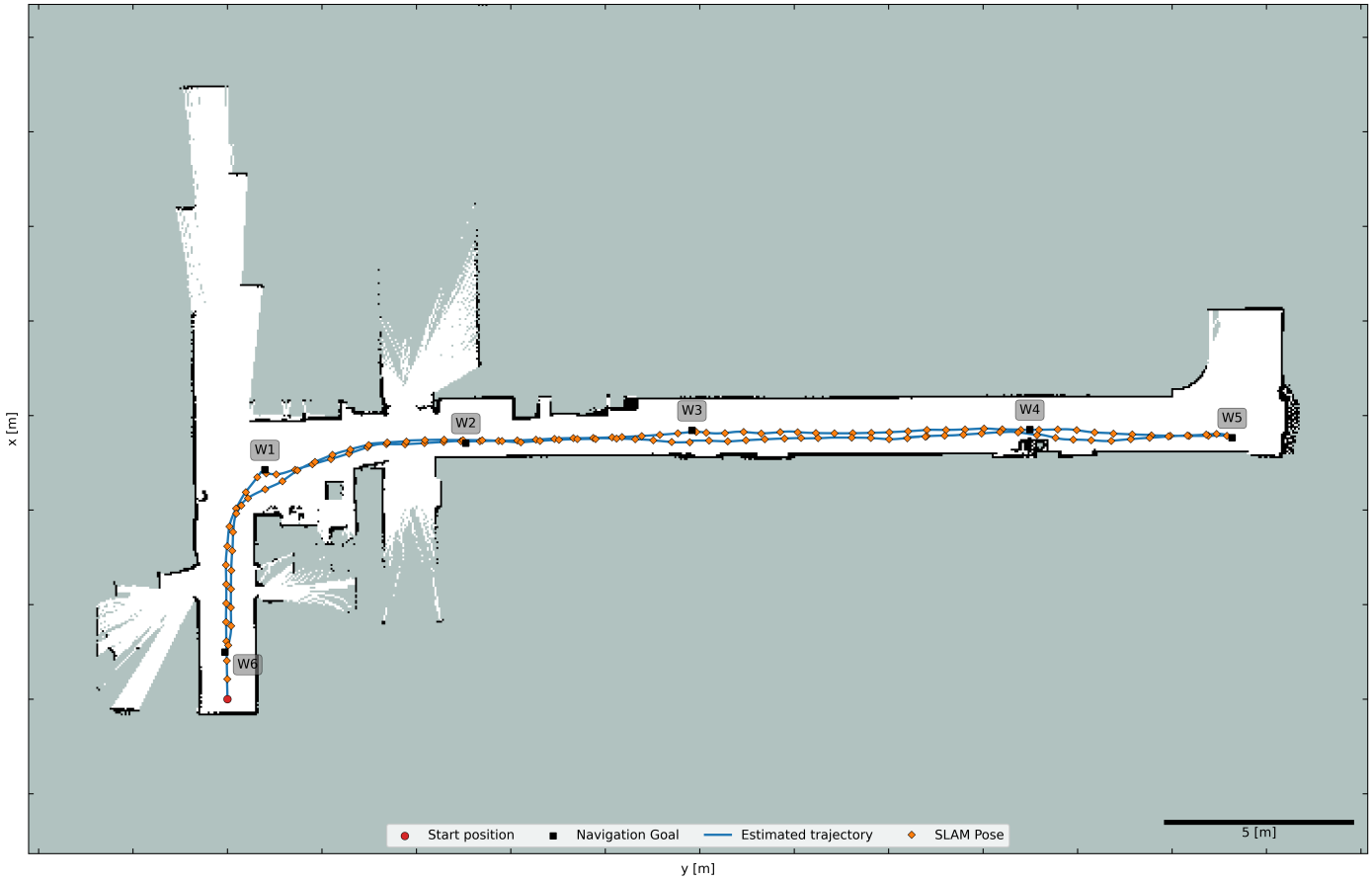


Figure 18. Occupancy grid map generated during the indoor navigation experiment using `slam_toolbox` in `online_async` mode. The figure also includes the commanded navigation goals, the platform trajectory, and the corresponding SLAM pose estimates.

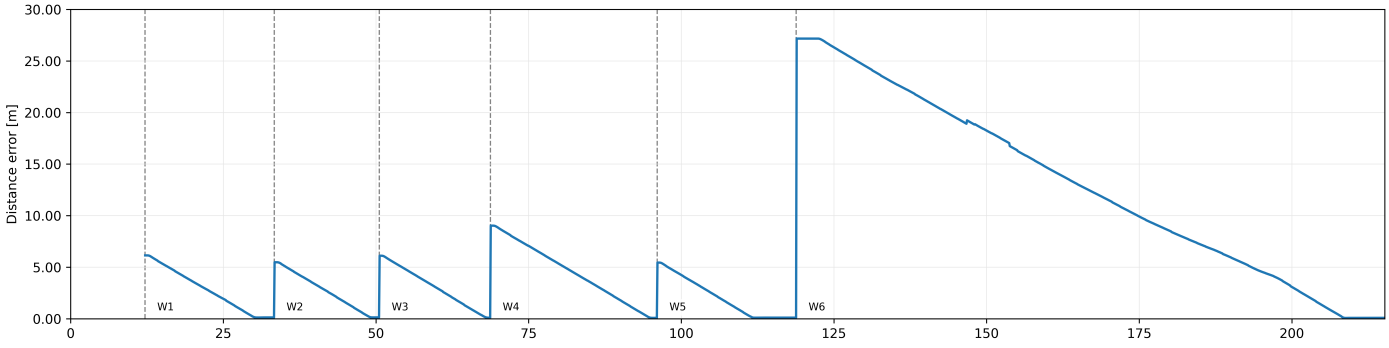


Figure 19. Distance error with respect to the active navigation goal during the indoor waypoint navigation experiment. Waypoint transitions are indicated by W1–W6.

5.4. Outdoor RTK-GNSS Waypoint Navigation

To provide an outdoor functional validation of the platform, an autonomous waypoint-following experiment was conducted using the onboard RTK-GNSS receiver in an open, obstacle-free environment. Since the onboard LiDAR is not rated for operation under direct sunlight, the obstacle layers of both the global and local Nav2 costmaps were disabled for this experiment. Consequently, obstacle avoidance was unavailable, and the trial was

conducted under continuous operator supervision, with manual intervention available at any time through the `twist_mux` velocity override.

A sequence of six geographic waypoints was defined from RTK-GNSS coordinates previously recorded with the onboard receiver while manually driving the platform along the desired route. These waypoints were subsequently provided to the ROS 2 navigation stack, which autonomously generated and executed the corresponding navigation plan. Figure 20 presents a representative trial, showing the commanded waypoints overlaid on a satellite image together with the RTK-GNSS-estimated trajectory followed by the platform.

The platform successfully completed the waypoint sequence during the reported trial, autonomously reaching all commanded waypoints. Figure 21 shows the distance to the active waypoint throughout the mission, illustrating the progressive reduction of the navigation error as each waypoint was approached before the navigation stack advanced to the next target.

Although performed in a controlled obstacle-free environment, this experiment provides a functional validation of the platform's outdoor autonomous navigation capability using RTK-GNSS waypoint following. A synchronized recording of the outdoor experiment, including the ROS 2 bag playback in Foxglove alongside the video captured during the trial, is available at https://youtu.be/exNnlu0ab7w (accessed on 17 August 2026) and in the Supplementary Material.

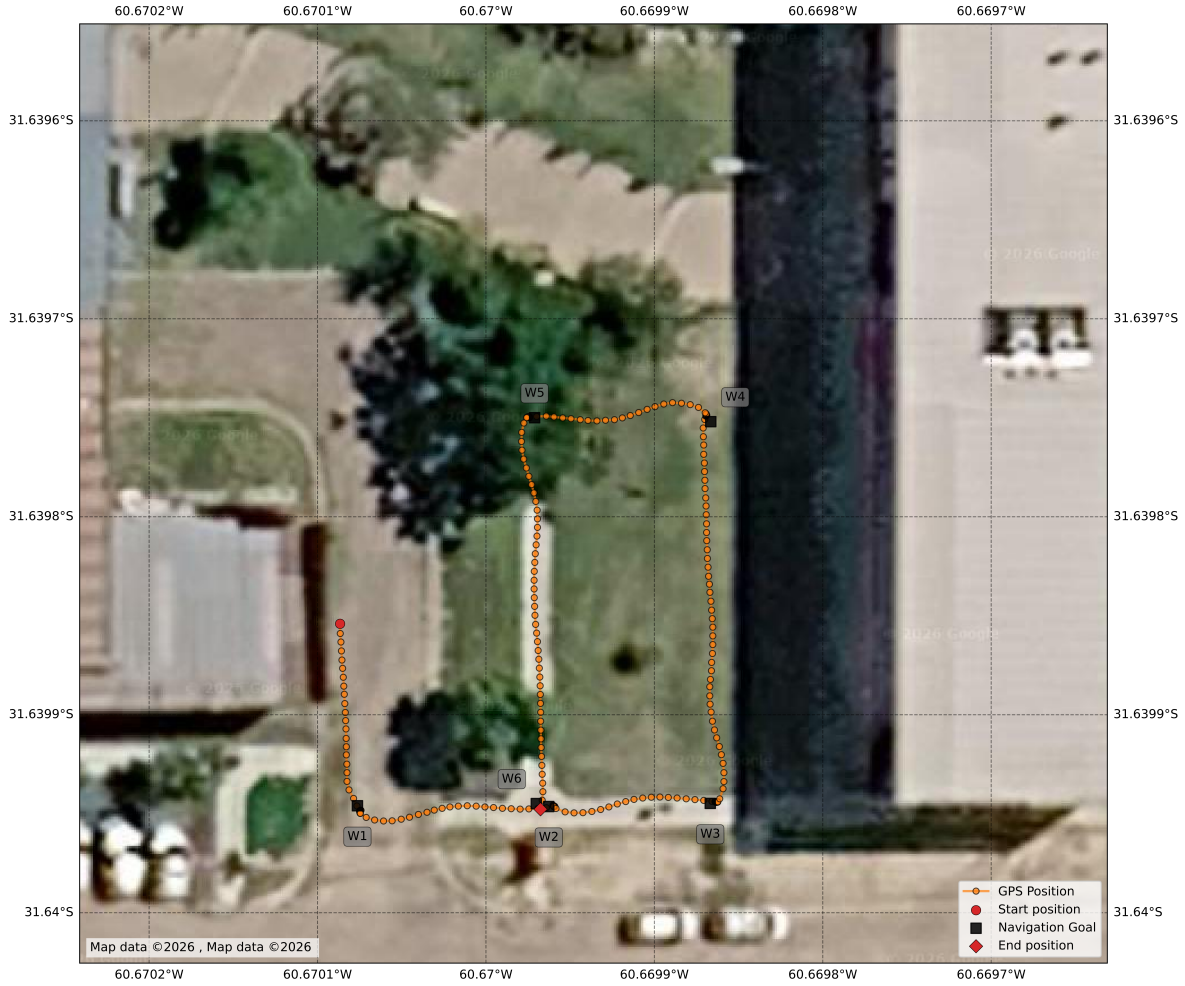


Figure 20. Satellite image showing the location where the outdoor experiments were conducted. The figure also includes the commanded navigation goals and the RTK-GNSS-estimated trajectory.

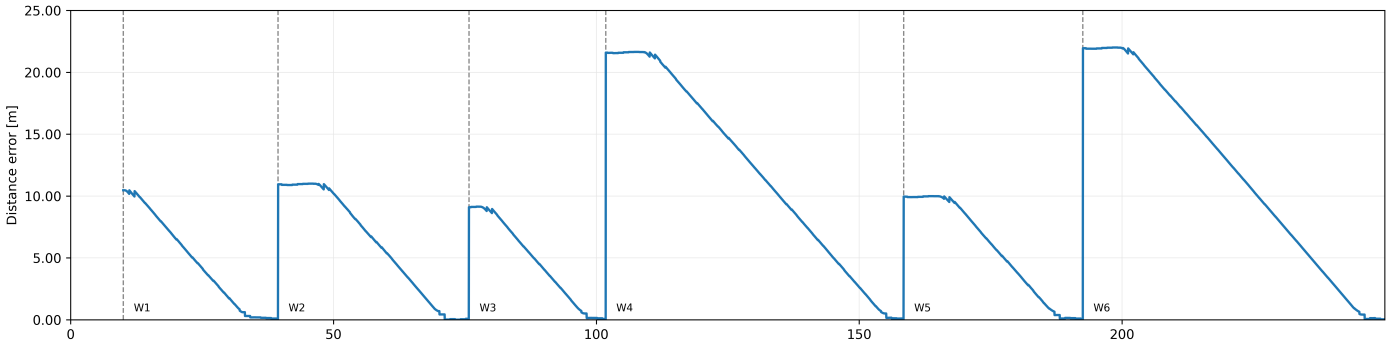


Figure 21. Distance error with respect to the active navigation goal during the outdoor waypoint navigation experiment. Waypoint transitions are indicated by W1–W6

6. Conclusions

This work presented the design of the SKD-1 skid-steer UGV, a modular robotic platform intended for research and experimentation in mobile robotics. The system integrates a differential skid-steer drive, onboard computing based on a Raspberry Pi single-board computer, and a microcontroller-based control layer implemented using an STM32 platform. The vehicle incorporates multiple sensing modalities, including LiDAR, GNSS, and an IMU, selected to support operation in both indoor and outdoor environments.

A key objective of the design was to create a modular and reproducible hardware platform. The mechanical structure was developed using a layered assembly that allows individual components to be easily manufactured, replaced, or modified. The electronic architecture integrates motor drivers, power management, sensing, and communication subsystems while maintaining a compact and serviceable configuration. Detailed build instructions and hardware descriptions were provided to facilitate replication of the platform.

The software architecture is based on the ROS 2 framework and relies, with the exception of the custom hardware-interface layer, entirely on standard ROS 2 packages for localization, mapping, and navigation. The velocity tracking and odometry calibration experiments confirmed the correct operation of the motion control pipeline and identified systematic biases in the platform's kinematic model: while the linear distance calibration revealed only a small direction-dependent bias, corrected with a wheel radius multiplier of 1.03, the rotational calibration exposed a much larger systematic error of approximately 39%, addressed through a wheel separation multiplier of 1.39. These findings underscore the value of empirical odometry calibration in ensuring accurate pose estimation for navigation. Verification experiments using the calibrated parameters showed a reduction in mean relative error to -0.64% (linear) and -0.62% (rotational), confirming the effectiveness of the calibration procedure. Indoor experiments further demonstrated correct integration of the sensing, actuation, and computing subsystems, including velocity tracking, SLAM-based mapping, and closed-loop waypoint navigation in an office environment. Outdoor experiments similarly demonstrated correct integration of the RTK-GNSS receiver, localization, and motion control subsystems, enabling autonomous waypoint-following navigation in an open, obstacle-free environment.

The SKD-1 platform provides a flexible open-hardware foundation for future research in autonomous ground vehicles, particularly for studies involving navigation, perception, and control in real-world environments. Together with the experimental results presented in this work, these findings demonstrate the proper functioning of the platform in both indoor and outdoor operating scenarios, while the limitations associated with outdoor perception and future work are discussed in the following Section.

7. Discussion

The SKD-1 platform was designed with the objective of providing a flexible and reproducible testbed for mobile robotics experimentation in both indoor and outdoor environments. The mechanical configuration is simple, modular, robust, and easy to build. Its dimensions and weight allow it to be handled by one or two operators at various experimental sites without the need for specialized equipment. The platform can be loaded and unloaded by a single person and transported in a medium-sized vehicle.

The modular architecture adopted for the mechanical and electronic subsystems proved advantageous during the development process, as it allows individual components to be assembled, replaced, or upgraded independently. This architecture also facilitates maintenance and simplifies platform modifications, which is particularly valuable in research and educational contexts where hardware configurations evolve in response to emerging technologies and new technical challenges.

The separation between the high-level computing layer and the low-level motor control layer further enhances system flexibility, as it relies on standard interfaces and a modular integration approach. The Raspberry Pi provides sufficient computational capability to run the ROS 2 software stack and handle common perception and navigation tasks, while the STM32-based controller manages time-critical motor control functions. Additionally, all sensing devices are connected via USB interfaces, establishing a standardized communication interface between subsystems and components. This design enables the platform to leverage widely available software frameworks and commercial sensors with minimal integration effort, thereby accelerating development processes. It also allows components to be replaced or upgraded without modifying the overall system architecture. Furthermore, the computing unit can be upgraded to a more powerful embedded system if additional processing capability is required.

From a sensing perspective, the integration of LiDAR, GNSS, and inertial measurements provides a comprehensive sensor suite that supports a wide range of navigation and localization algorithms. The inclusion of an RTK-capable GNSS receiver extends the operational capabilities of the platform to outdoor environments requiring high-precision positioning. This combination of sensors makes the system suitable for experiments involving autonomous navigation, mapping, and sensor fusion. Moreover, the sensor suite is not fixed; the modular architecture, combined with standardized interfaces, enables straightforward integration of additional sensors to address emerging challenges in autonomous robotics.

The open-hardware philosophy adopted in this work also plays an important role in the potential impact of the platform. By providing access to mechanical design files, electronic schematics, and software repositories, the system can be replicated and adapted by other researchers. This openness promotes reproducibility and facilitates the collaborative development of new functionalities built upon the presented platform.

Despite these advantages, certain limitations should be acknowledged. The skid-steer configuration was deliberately chosen for its mechanical simplicity—steering is resolved entirely in software, with no additional mechanical steering linkage—and for the improved traction it offers across mixed and unstructured terrain compared to a two-wheel configuration. Wheel slip during turning is an inherent and accepted characteristic of this drive configuration rather than an uncharacterized defect; a detailed kinematic and slip characterization was considered outside the scope of this work, which focuses on platform integration rather than locomotion modeling. Additionally, the current platform configuration is primarily focused on navigation and perception experiments; therefore, applications requiring heavier payloads or specialized manipulation capabilities may necessitate structural modifications.

A further limitation concerns the lack of quantitative characterization of certain operational parameters, such as current draw under normal driving load, effective payload capacity, and traction force, as their measurement during operation would require dedicated instrumentation (e.g., current sensors, load cells, or dynamometers) that was not available at the time of testing. A baseline no-load current measurement is reported in Section 2.2 for reference, but current draw under representative driving conditions was not characterized. The reported payload and operating time values are therefore based on component specifications and qualitative experience rather than direct measurement. Incorporating such instrumentation to quantitatively characterize these parameters is left for future work.

Future work will focus on expanding the sensing suite and further improving system integration. Potential directions include the incorporation of additional perception sensors—such as RGB-D cameras or LiDAR sensors rated for direct sunlight exposure—, enhancements to the power management subsystem, and the development of advanced autonomy algorithms that leverage the available sensing and computational resources. Moreover, further experimental validation in more complex environments will be conducted to assess the platform's performance in real-world robotic applications.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/hardware4030017/s1.

Name	Type	Description
skd1-main	Package (.zip)	Main packages
skd1_sensors-main	Package (.zip)	Sensor packages
skd1_visualization-main	Package (.zip)	Visualization packages using RViz
skd1_simulation-main	Package (.zip)	Simulation packages using Gazebo
skd1_extra-main	Package (.zip)	Extra packages, containing design files, BoM and the ROS 2 bag files recorded during the navigation experiments.
hibachi_firmware-main	Package (.zip)	Firmware for the STM32 Blue Pill low level controller
skd1-connection-diagram	PDF (.pdf)	SKD-1 connection diagram with vector graphics (recommended for printing and zooming)
SKD-1_Indoor	MP4 (.mp4)	Video recording of the indoor experiment
SKD-1_Outdoor	MP4 (.mp4)	Video recording of the outdoor experiment

Author Contributions: Conceptualization, G.M.S., M.M., A.C. and L.G.; methodology, G.M.S. and A.C.; software, G.M.S. and A.C.; validation, H.S.U.H., J.E.B. and N.D.; formal analysis, G.M.S. and A.C.; investigation, G.M.S. and A.C.; resources, G.M.S. and L.G.; data curation, A.C., H.S.U.H. and J.E.B.; writing—original draft preparation, G.M.S., A.C. and M.M.; writing—review and editing, G.M.S., A.C., H.S.U.H., N.D. and L.G.; visualization, H.S.U.H., J.E.B. and N.D.; supervision, G.M.S.; project administration, G.M.S. and A.C.; funding acquisition, G.M.S. and L.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been granted by the Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET). We would like to thank the Ministerio de Producción, Ciencia y Tecnología, Santa Fe (PEIC I+D 2023-231), and Universidad Nacional del Litoral (CAI+D 2020 50620190100080LI and CAI+D 2024 85520240100081LI) for their support for this work.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article and supplementary material. Further inquiries can be directed to the corresponding authors.

Acknowledgments: During the preparation of this manuscript, the authors used ChatGPT (OpenAI, GPT-5.3) for language editing and formatting assistance. The authors reviewed and edited all generated text and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CPU	Central Processing Unit
DC	Direct Current
FDM	Fused Deposition Modeling
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input-Output
HDLC	High-Level Data Link Control
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
PCB	Printed Circuit Board
PETG	Polyethylene Terephthalate Glycol
PID	Proportional-Integral-Derivative
PVC	Polyvinyl Chloride
PWM	Pulse-Width Modulation
RGB-D	Red, Green, Blue & Depth
ROS	Robot Operating System
RTK	Real-Time Kinematic
SLAM	Simultaneous Localization and Mapping
UGV	Unmanned Ground Vehicle
URDF	Unified Robot Description Format
USB	Universal Serial Bus

Appendix A

Table A1. Condensed Bill of Materials.

Qty.	Component	Source of Materials	Material Type	Cost (Unit)
1	Platform base 400 × 250 × 5 mm	Local store	Acrylic (Plastic)	20.00
1	Protective structure and cover	Local store	Acrylic (Plastic)	75.00
1	Creality PETG White 3D Printer Filament 1.75 mm 1 kg	Amazon	PETG (Plastic)	21.99
1	KCD4 Rocker Switch DPST 2-Position	Amazon	Electromechanical	4.84
1	Mini Digital Voltmeter 0.56” LED Display Panel	Amazon	Electronic Component	2.15
1	Push Button Switch, Twist Release	Amazon	Electromechanical	16.71
1	Panel Mount 5 × 20 mm Fuse Holder	Amazon	Electrical Component	1.76
1	Panel Mount 5.5 × 2.1 mm DC Power Jack	Amazon	Electrical Connector	1.25
1	Antenna ground-plane	Local store	Aluminum	4.39
4	70:1 Metal Gearmotor 37D × 70L mm 12V with 64 CPR Encoder (Helical Pinion)	https://www.pololu.com/product/4754 (accessed on 17 August 2026)	Actuator	60.95
2	Stamped L-Bracket Pair for 37D mm Metal Gearmotors	https://www.pololu.com/product/1084 (accessed on 17 August 2026)	Mechanical	11.95

Table A1. *Cont.*

Qty.	Component	Source of Materials	Material Type	Cost (Unit)
2	Dagu Wild Thumper Wheel 120 × 60 mm	https://www.pololu.com/product/1556 (accessed on 17 August 2026)	Mechanical	26.24
2	12mm Hex Wheel Adapter for 6mm Shaft, Extended (2-Pack)	https://www.pololu.com/product/2687 (accessed on 17 August 2026)	Mechanical	8.95
2	Dual G2 High-Power Motor Driver 18v18 Shield for Arduino	https://www.pololu.com/product/2515 (accessed on 17 August 2026)	Power Electronics	74.95
1	WeAct Studio Blue Pill	https://es.aliexpress.com/item/1005004918334754.html (accessed on 17 August 2026)	Microcontroller	6.40
1	12V 10Ah Li-Ion Battery	https://productosintegra.com/producto/bateria-recargable-litio-12v-10a (accessed on 17 August 2026)	Power Supply	250.00
1	Raspberry Pi 5 B+ with 8 GB RAM	https://www.adafruit.com/product/5813 (accessed on 17 August 2026)	Single Board Computer	200.00
1	Power supply expansion board for Raspberry Pi 5	https://category.yahboom.net/products/power-board-pi5 (accessed on 17 August 2026)	Power Management	12.00
1	NVMe Base for Raspberry Pi 5	https://shop.pimoroni.com/products/nvme-base?variant=41219587178579 (accessed on 17 August 2026)	Storage Expansion	14.96
1	Crucial P2 250GB M.2 2280 NVMe SSD		Storage	115.00
1	TP-Link TL-MR3020 (or similar)	https://www.tp-link.com/ar/service-provider/lte/tl-mr3020 (accessed on 17 August 2026)	Networking	29.99
1	M2, M2.5, M3 and M4 fasteners	Local Store		30.00
			Total without sensors	1276.52
1	simpleRTK2B—Basic Starter Kit	https://www.ardusimple.com/product/simplertk2b-basic-starter-kit-ip65 (accessed on 17 August 2026)	Sensor/GNSS	263.75
1	Phidget Spatial 3/3/3	https://www.phidgets.com/?prodid=1205 (accessed on 17 August 2026)	Sensor/IMU	99.75
1	Slamtec RPLiDAR C1	https://www.dfrobot.com/product-2803.html (accessed on 17 August 2026)	Sensor/LiDAR	69.00
			Total (main components listed above)	1709.02

Note: the total above sums only the main components listed in this condensed BOM; it excludes wiring, connectors, and the custom PCB.

References

1. La Regina, R.; Genel, Ö.E.; Pappalardo, C.M.; Guida, D. A Comprehensive Review of Theoretical Advances, Practical Developments, and Modern Challenges of Autonomous Unmanned Ground Vehicles. *Machines* **2025**, *13*, 1071. [CrossRef]
2. Niu, H.; Chen, Y. The Unmanned Ground Vehicles (UGVs) for Digital Agriculture. In *Smart Big Data in Digital Agriculture Applications: Acquisition, Advanced Analytics, and Plant Physiology-Informed Artificial Intelligence*; Agriculture Automation and Control; Springer Nature: Cham, Switzerland, 2024; pp. 99–109. [CrossRef]
3. Nahavandi, S.; Alizadehsani, R.; Nahavandi, D.; Mohamed, S.; Mohajer, N.; Rokonuzzaman, M.; Hossain, I. A Comprehensive Review on Autonomous Navigation. *ACM Comput. Surv.* **2025**, *57*, 1–67. [CrossRef]
4. Weerakoon, K.; Sathyamoorthy, A.J.; Elnoor, M.; Manocha, D. AdVENTR: Autonomous Robot Navigation in Complex Outdoor Environments. In *Proceedings of the Experimental Robotics*; Ang, M.H., Jr., Khatib, O., Eds.; Springer: Cham, Switzerland, 2024; Volume 30; pp. 219–228. [CrossRef]
5. Khan, R.; Malik, F.M.; Raza, A.; Mazhar, N. Comprehensive study of skid-steer wheeled mobile robots: development and challenges. *Ind. Robot. Int. J. Robot. Res. Appl.* **2020**, *48*, 142–156. [CrossRef]
6. Clearpath Robotics Husky A300. Available online: https://clearpathrobotics.com/husky-a300-unmanned-ground-vehicle-robot/ (accessed on 7 July 2026).
7. Clearpath Robotics Turtlebot 4. Available online: https://clearpathrobotics.com/turtlebot-4/ (accessed on 7 July 2026).
8. AgileX Scout 2.0. Available online: https://global.agilex.ai/products/scout-2-0 (accessed on 8 March 2026).
9. AgileX Scout Mini. https://global.agilex.ai/products/scout-mini (accessed on 8 March 2026).
10. Husarion Panther. Available online: https://husarion.com/manuals/panther/ (accessed on 8 March 2026).
11. Husarion ROSbot. Available online: https://husarion.com/manuals/rosbot/ (accessed on 8 March 2026).
12. TurtleBot. Available online: https://www.turtlebot.com/ (accessed on 8 March 2026).
13. Linorobot. Available online: https://linorobot.org/ (accessed on 8 March 2026).

14. Noah Robot (Hardware). Available online: https://github.com/GonzaCerv/noah-hardware (accessed on 8 March 2026).
15. Noah Robot (Software). Available online: https://github.com/GonzaCerv/noah-software (accessed on 8 March 2026).
16. Vega, J.; Cañas, J.M. PiBot: An Open Low-Cost Robotic Platform with Camera for STEM Education. *Electronics* **2018**, *7*, 430. [CrossRef]
17. Betancur-Vásquez, D.; Mejia-Herrera, M.; Botero-Valencia, J. Open source and open hardware mobile robot for developing applications in education and research. *HardwareX* **2021**, *10*, e00217. [CrossRef]
18. Nwankwo, L.; Fritze, C.; Bartsch, K.; Rueckert, E. ROMR: A ROS-based open-source mobile robot. *HardwareX* **2023**, *14*, e00426. [CrossRef]
19. Márquez-Sánchez, C.; Sandoval-Gutiérrez, J.; Martínez-Vázquez, D.L. Construction of an Educational Prototype of a Differential Wheeled Mobile Robot. *Hardware* **2026**, *4*, 2. [CrossRef]
20. Clearpath Robotics Jackal. Available online: https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/ (accessed on 7 July 2026).
21. Turtlebot 3. Available online: https://www.turtlebot.com/turtlebot3/ (accessed on 29 July 2026).
22. Robot Operating System (ROS). Available online: https://www.ros.org/ (accessed on 8 March 2026).
23. Capovilla, A.; Sánchez, G.; Murillo, M.; Giovanini, L. Diseño y construcción de un vehículo skid-steer compatible con ROS. In Proceedings of the XI Jornadas Argentinas de Robótica, San Carlos de Bariloche, San Carlos de Bariloche, Argentina, 9–11 March 2022.
24. Sánchez, G.M.; Murillo, M.H.; Benavidez, J.E.; Deniz, N.N.; Giovanini, L.L. Hibachi: A ROS compatible skid-steer vehicle. In Proceedings of the 2021 XIX Workshop on Information Processing and Control (RPIC), San Juan, Argentina, 3–5 November 2021; pp. 1–6. [CrossRef]
25. ROS 2 Control. Available online: https://control.ros.org/jazzy/index.html (accessed on 27 July 2026).
26. Moore, T.; Stouch, D. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13), Padua, Italy, 15–18 July 2014.
27. Macenski, S.; Jambrecic, I. SLAM Toolbox: SLAM for the dynamic world. *J. Open Source Softw.* **2021**, *6*, 2783. [CrossRef]
28. Macenski, S.; Martin, F.; White, R.; Ginés Clavero, J. The Marathon 2: A Navigation System. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2019–24 January 2020.
29. Macenski, S.; Singh, S.; Martin, F.; Gines, J. Regulated Pure Pursuit for Robot Path Tracking. *Auton. Robot.* **2023**, *47*, *685–694*. [CrossRef]
30. Foxglove. Available online: https://foxglove.dev (accessed on 29 July 2026).