جامعة سطيف 1
UNIVERSITY OF SETIF 1

Development of Modular Robotic Platforms

for Development, Research and Education

Presented by: Assil M. Ferahta
For the degree of **Masters** in **Embedded Systems Electronics**

University: Setif 1 Ferhat Abbas
Thesis Supervisor: Dr. Younes Terchi
Date: 2, 2025

Acknowledgments

Abstract

Modular robotics has emerged as a promising approach to address the limitations of traditional monolithic robotic systems in terms of adaptability, scalability, and maintenance. By enabling robots to be constructed from interchangeable functional units, modular robotic platforms offer the potential for rapid reconfiguration, fault tolerance, and task-specific customization. However, many existing modular robotic systems suffer from limited hardware interoperability, tightly coupled software architectures, and complex reconfiguration procedures that hinder practical deployment and scalability.

This thesis presents the design and implementation of a modular robotics platform that emphasizes both hardware and software modularity. The proposed platform is composed of standardized robotic modules that integrate actuation, sensing, control, and power interfaces, allowing modules to be physically and logically interconnected in multiple configurations. A layered software architecture is introduced to support dynamic module identification, inter-module communication, and distributed control, enabling plug-and-play functionality and reducing system integration effort.

The effectiveness of the proposed modular robotics platform is validated through a series of experimental evaluations focusing on reconfiguration capability, system scalability, and operational reliability. Experiments demonstrate that the platform can be reconfigured with minimal manual intervention while maintaining stable communication and control across different module arrangements. Performance metrics such as reconfiguration time, communication latency, and task execution success rate are used to assess system behavior under varying configurations.

The main contributions of this thesis include: (i) the design of a unified hardware interface that supports modular robotic assembly, (ii) a flexible software framework enabling dynamic reconfiguration and distributed control, and (iii) experimental validation of a modular robotic platform demonstrating practical feasibility. The results indicate that the proposed approach provides a viable foundation for scalable and adaptable modular robotic systems.

Contents

General Introduction

Robotics has emerged as a pivotal technology in fields ranging from industrial automation and research to education and personal innovation. Despite rapid technological advances, a gap persists in providing accessible, versatile, and modular platforms that can cater to a diverse audience, ranging from researchers and students to hobbyists and ordinary users with minimal technical background. This thesis addresses this gap by introducing a set of modular robotic platforms; Bulky, Thegill, and μtomba, Dronegaze, alongside controllers like iLite and software tools for development and tinkering and interfacing which are designed with adaptability and ease of use in mind.

At the heart of this work is the idea of modularity as a means to bridge diverse domains from rigorous academic research and hands-on educational experimentation to hobbyist tinkering and everyday problem solving. Bulky is designed as a precision rover that integrates a flexible sensor-ready framework, exemplifying how a platform can evolve by accommodating custom modules. Thegill challenges conventional boundaries by operating seamlessly across terrestrial and aquatic environments, pushing the envelope on how modular components can harmonize mechanical adaptability with sophisticated control strategies. Meanwhile, μtomba distills robotics to its essentials, replacing rigid circuit boards with a breadboard setup to foster an environment of learning, rapid prototyping, and creative exploration.

Together, these platforms are not merely products with predefined applications; each platform embodies a distinct set of features and design philosophies while adhering to a common vision: to democratize robotics by offering modular and upgradable solutions that can be customized, extended, and integrated into various applications. They are conceptual testbeds that encourage users to tap on robotics via open design and abstraction. This thesis systematically explores the theoretical foundations of modular design, the practical integration of mechanical, software and electronic systems, and the experimental validation of these concepts. By doing so, it aims to demonstrate that embracing modularity and abstraction can transform robotics into a universally accessible and continually evolving field, empowering researchers, students, enthusiasts, and even everyday users to craft adaptive solutions in parallel of what is being developed in the industrial world.

This thesis documents the philosophy, problematic, design, implementation, and evaluation of these platforms, demonstrating how a modular approach can address a

variety of challenges in robotics. It presents the theoretical foundations of modular design, details the system architecture, and evaluates the platforms through experimental testing. Ultimately, the work illustrates how such adaptable solutions can serve as universal products, empowering users to explore innovative applications in research, education, and beyond.

0.1 Overview of Robotics

Robotics is an interdisciplinary field that has evolved from simple automation to the development of complex systems with autonomous decision-making capabilities. This evolution is driven by:

- **Sensor and Actuator Advances:** Modern robots leverage cutting-edge sensors and actuators that provide high-resolution environmental data and precise control [4, 19].
- **Computational Power and Algorithms:** Enhanced processing capabilities and advanced algorithms, including machine learning and adaptive control, allow robots to perform real-time analysis and complex tasks.
- **Modularity and Open-Source Development:** The trend toward modular design enables scalable, reconfigurable platforms that can be tailored for research, education, and even consumer applications. This paradigm shift facilitates collaborative development and rapid innovation.

Academic studies often address challenges such as system integration, sensor fusion, and the development of robust control algorithms. Foundational works by Siciliano *et al.* [19] and Craig [4] have significantly influenced modern robotics by providing theoretical and practical insights that continue to shape the field.

0.2 Problematic

Despite extensive research in modular robotics, many existing platforms remain constrained by tightly coupled hardware designs, platform-specific software stacks, and limited interoperability between robotic systems. As a result, adapting a robot to new tasks often requires substantial redesign of both hardware and software, undermining the potential benefits of modularity.

Additionally, current solutions frequently target a single class of robots or applications, such as self-reconfigurable robots or educational kits, without addressing the challenge of maintaining a unified development framework across ground, amphibious,

and aerial robotic platforms. This fragmentation increases complexity for developers and limits the scalability of modular robotic ecosystems.

The core problem addressed in this thesis is the lack of a unified modular robotics framework that enables reusable hardware and software modules to be seamlessly deployed across multiple heterogeneous robotic platforms while maintaining simplicity, flexibility, and performance.

0.3 Objective

The primary objective of this thesis is to design, implement, and validate a modular robotics platform that supports hardware and software reuse across multiple robotic systems with different operational requirements.

The specific objectives are as follows:

Design modular robotic platforms capable of supporting interchangeable sensors, actuators, and peripherals.

Develop a software architecture that enables plug-and-play functionality, dynamic configuration, and simplified debugging.

Implement a unified controller and communication framework to abstract low-level hardware details.

Validate the proposed modular approach through real-world robotic platforms operating in different environments and applications.

0.4 Research Approach

The methodology of this thesis is built on the principles of abstraction and modularity. Rather than developing fixed-function systems, the focus is on creating versatile platforms with the following core components:

- **Abstraction:** By decoupling hardware from software, the design allows for independent evolution and easy integration of new technologies.
- **Modularity:** Systems are conceived as collections of interchangeable modules, enabling rapid prototyping, iterative testing, and scalability. This approach is supported by recent research on modular self-reconfigurable robots [22].

- **Theory-Practice Integration:** Theoretical models, such as the kinematic formulations presented earlier, are used to inform design choices and are validated through extensive simulations and experiments.

0.5 Introduction to Modularity in Robotic Systems

Modularity is a design philosophy that emphasizes constructing complex systems from smaller, self-contained components—commonly referred to as modules[1]. In the context of robotic platforms, this approach involves decomposing the system into discrete parts, each encapsulating specific functionalities such as sensing, actuation, communication, or processing, thereby simplifying design and troubleshooting.

At its core, modularity offers several significant benefits:

- **Reusability**[2]: Successfully developed and tested modules can be reused across different configurations or projects.
- **Flexibility**[3]: Standardized interfaces[4] and communication protocols enable modules to be interchanged or reconfigured, a feature particularly valuable in rapid prototyping and iterative design.
- **Scalability**[5]: New modules can be integrated without necessitating a complete redesign, making it straightforward to expand functionality or incorporate emerging technologies.
- **Ease of Maintenance**[6]: When a single module fails or becomes outdated, it can be replaced or upgraded without affecting the entire system.

In educational environments, modular robotic platforms serve as excellent teaching tools. They allow students to visualize and understand the intricate relationships between individual system components and the overall architecture. By experimenting with different module combinations, learners gain practical insights into system integration, problem-solving, and the inherent trade-offs in design decisions.

[1] Modules are discrete units of functionality that can be independently developed, tested, replaced, or upgraded.

[2] Reusability refers to the ability to use developed modules in various configurations or projects, reducing overall development time and costs.

[3] Flexibility is the capacity to interchange or reconfigure modules to adapt to new requirements or experimental setups.

[4] Interfaces are predefined methods for communication between modules, ensuring compatibility and ease of integration.

[5] Scalability describes the ease of integrating additional modules into an existing system as project requirements evolve.

[6] Ease of Maintenance refers to the capacity to replace or upgrade individual modules independently, thereby minimizing disruptions to the overall system.

From a development perspective, modularity supports a more agile workflow. It enables parallel development, where different teams or individuals work simultaneously on separate modules. Upon integration, the overall system benefits from collective innovations, ensuring a robust and versatile platform. This design strategy also opens avenues for further innovation, as new modules can be developed to enhance system capabilities without overhauling the existing architecture.

Ultimately, adopting a modular approach in robotic platforms aligns perfectly with the goals of advancing research, fostering educational experiences, and enabling rapid prototyping. By leveraging the principles of modularity, developers can create adaptable and cost-effective platforms that meet the dynamic challenges of modern robotics.

0.6 Notion of platforms

Platforms can be considered a workspace that provides mechanical, electronic, software and power support for a project be it experimental or practical, but at the heart of this concept is "Abstraction", you don't need to understand how semiconductor or hardware works to work with an arduino for an instance, all you need is to be familiar with algorithms and programming; high level programming or even human language in our times of AI, no assembly or machine code anymore, but if you need to or want to go that low, it's still possible for you to do so, the arduino platform allows for all therefore a *platform* is a deliberately stable core together with explicit contracts that enable independent parties to create, combine, and evolve extensions and applications over time without modifying the core.

0.6.1 Platform vs. Product, Toolkit, Framework, Ecosystem

The magic behind ***ASCE*** platforms is that aside from being platforms, they may also behave as products, toolkits frameworks, and an ecosystem (Since they allow for cross platform and emergent behaviour).

Table 1: Platform vs. related constructs

Construct	Core intent	Who extends it?	Interface stability	Typical deliverables
Product	Solve a specific use case	Vendor	n/a	Device/app with fixed features
Toolkit	Provide parts to assemble	End user	Low–Medium	Components, examples
Framework	Invert control; fill hooks	Developer	Medium–High	Base code + callbacks
Ecosystem	Community of complements	Many parties	Mixed	Marketplaces, standards
Platform	**Stable core + contracts for many products**	**First&third parties**	**High (versioned)**	**Core, interfaces, SDKs, docs, governance**

0.6.2 Modular Robotic Platform (MRP)

Since we are developing platforms meant for versatility and ease of use, It is useful to "objectify" platforms and give them rules and features and compatibility constrains , let's call this object an "MRP" standing for "Modular Robotic Platform".

An MRP exposes mechanical, electrical, and software contracts for safe composition of actuators, sensors, power, and behaviours, allowing the users the creation of "Assemblies" and conceptualizing them as fits their uses or ordering them from fabricators (Which is what our startup is about as will be shown later)

MRPs also allow us to keep track of information such as weight of platforms, cost, and gather statistical data and feedback for further improvements and maturation of assemblies and their constituents.

With all this said, a *Modular Robotic Platform* (MRP) exposes mechanical, electrical, and software contracts for safe composition of actuators, sensors, power, and behaviors so that a wide range of robots can be built, upgraded, and repurposed without modifying the core.

0.6.3 Space-Time Modularity

"ASCE" platforms; what this thesis is about; follows an analogy: "if modularity is a universe, then our platforms allow freedom in both **space** and **time**"

Space is *discipline* (mechanical, electronics, software, logic/UX). Time is *user maturity* (consumer, learner, researcher, professional). A high-quality MRP provides variation points across space and growth paths across time as shown in the following table.

Table 2: ASCE Space–Time matrix of modularity

Time ↓ / Space →	**Mechanical**	**Electronics**	**Software**	**Logic & UX**
Consumer	Prebuilt mounts; safe geometry	Pre-wired modules	App presets	One-click modes, arming rules
Learner	Parametric parts	Labeled harnesses; rails	Config files; examples	Guided labs, wizards
Researcher/Dev	Custom frames; CoG guidance	Swappable drivers; EMI budget	Ports/adapters; SDK	Tests, CI, telemetry
Professional	Optimized rigs; stiffness specs	Power tiers; fusing; connectors	RT guarantees; schedulers	Toolchains, HIL, certification hooks

according to this analogy, ASCE platforms essentially behaves as a platform for users and developers, we may keep expanding and building on top of solid concepts from these platforms, and users may use them to tackle their own problematic.

0.6.4 The ASCE Platform Family

It is difficult to start with a single build (MRP) that is versatile enough to do everything in the field of robotics; be highly flexible and easy to use, operate across all environments (air,land,water), and offer manipulation means such as robotic arms and similar actuators, this could become a realizable target if we have multiple separate builds (MRPs) and unify them after their maturation under one platform or two, therefore initially we will go with the following list to cover all said domains:
-Dronegaze: A drone test jig allowing the developement and testing aerial builds.
-Bulky: A compact rover equipped with sensors for spatial interpretation.
-The'gill: an amphebian build able to move both in terrain and water surfaces.
-ţTomba: a pedagogic breadboard based build suitable for learning and teaching.
-ILITE: a universal controller, which is the most important element as it can be used to control any arbitrary configuration within our platform families or beyond.

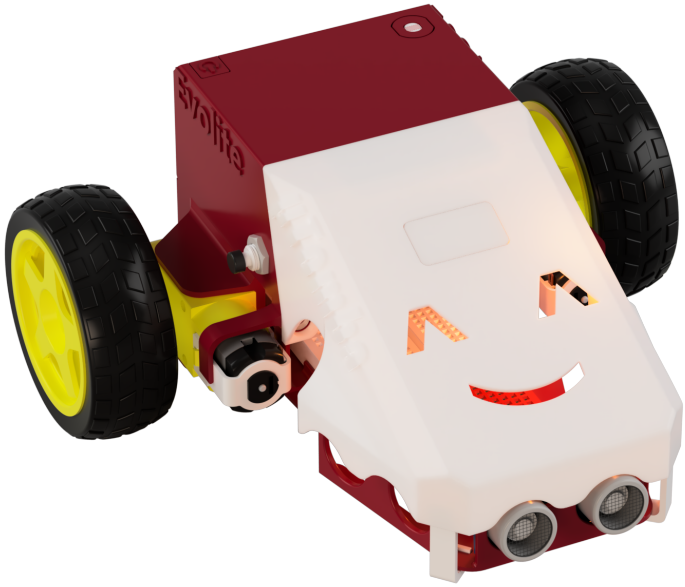

Figure 1: a 3D render of ţTomba

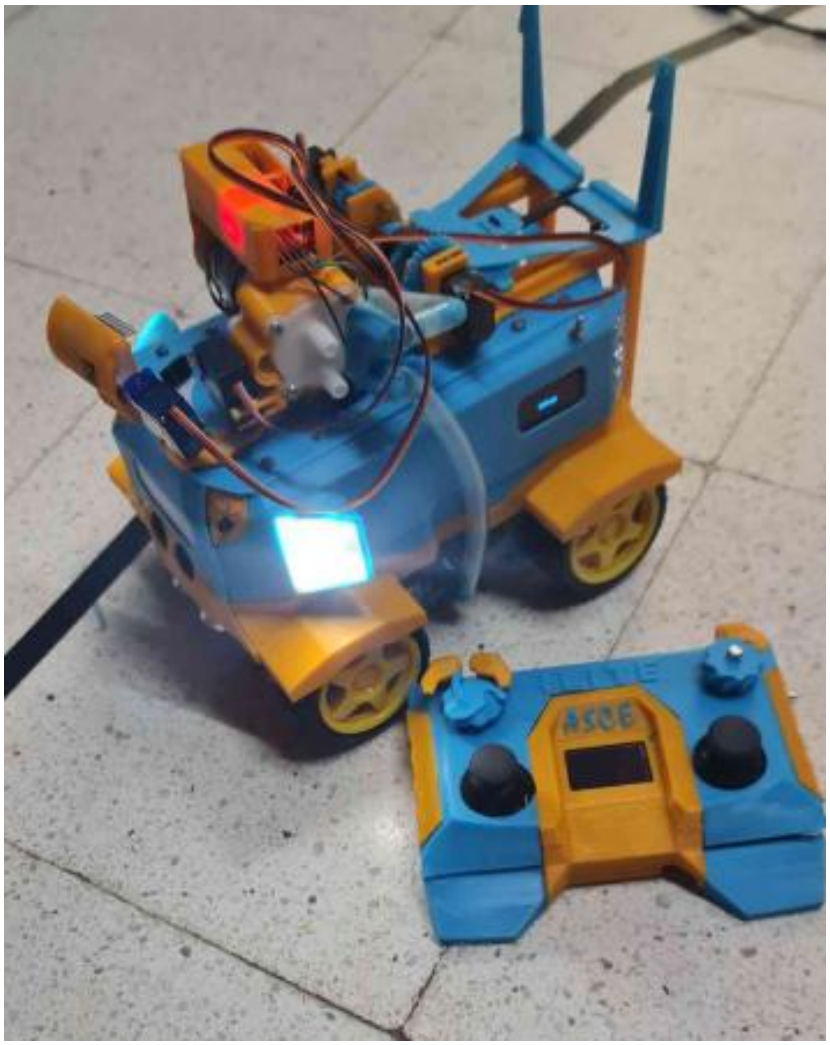

Figure 2: Bulky alongside ILITE

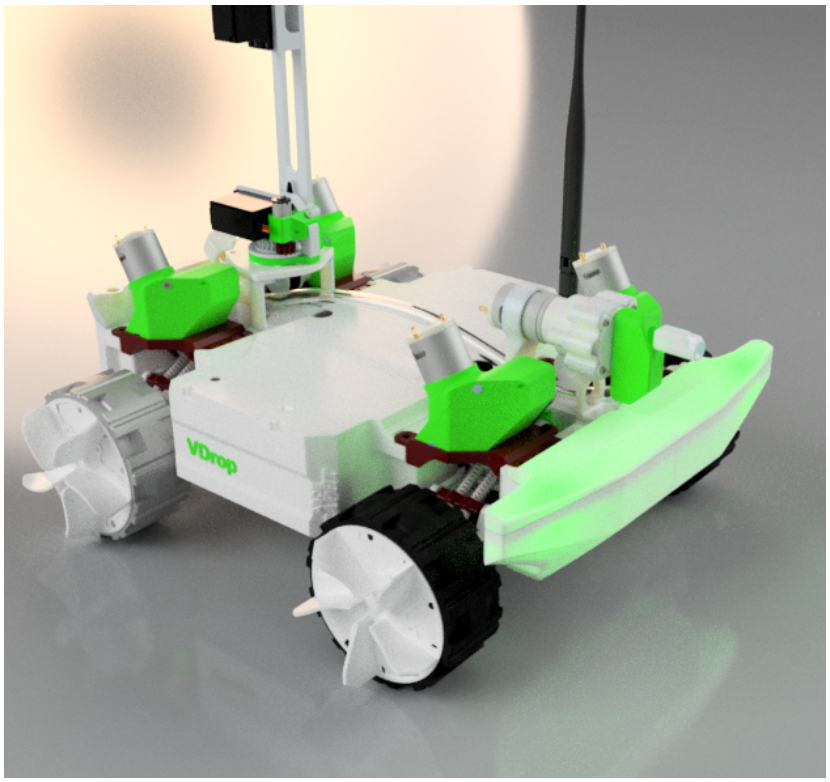

Figure 3: a 3D render of The'Gill

Figure 4: A 3D render of ILITE

Figure 5: A 3D render of Dronegaze test jig

0.6.5 Minimum Viable Platform (MVP)

We define an "MVP" any valid MRP that is enough to serve as a platform for a specific use case; for instance a user may need a purely mobile platform to test out their movement and localisation algorithms for example, they don't need every other module or peripheral available, this would make an MVP. Less than an MVP would basically remain a set of components and modules. an MVP is essentially a MRP but without saturating **its module capacity**.

0.6.6 Platform Layers for Robotics

Robotic systems integrate diverse fields, each contributing unique methodologies and insights.

0.6.7 Mechanical Engineering

Mechanical engineering underpins the design and fabrication of the physical structure. Key considerations include:

- **Structural and Material Analysis:** Techniques such as finite element analysis (FEA) assess the durability and performance of robot components.

- **Mechanism Design:** Analyzing kinematic chains, joint configurations, and mobility mechanisms is essential for ensuring reliable operation under various conditions.

Classical texts, such as those by Spong *et al.* [20], provide deep insights into these principles.

0.6.8 Electrical and Electronics Engineering

This discipline focuses on:

- **Circuit and Sensor Integration:** Designing custom PCBs, interfacing diverse sensors, and ensuring efficient power management are critical, especially for platforms requiring long battery life.
- **Signal Processing:** Filtering and interpreting sensor data to provide accurate feedback for control algorithms.

These topics are discussed extensively in foundational works like [4].

0.6.9 Computer Science and Control Theory

Software and control algorithms are the brain of robotic systems:

- **Control Strategies:** The implementation of PID controllers, model predictive control (MPC), and adaptive algorithms enables stable and responsive behavior.
- **Software Architectures:** Developing modular, scalable, and real-time software platforms is essential for integrating various subsystems. Open-source frameworks like ROS have revolutionized this integration.

For a comprehensive treatment, refer to [19].

0.6.10 Systems Engineering

Systems engineering ensures that all components operate harmoniously:

- **Integration and Testing:** Systematic approaches for validating interoperability and performance are vital.
- **Lifecycle Management:** Structured methodologies help manage the complexity of designing, developing, and deploying advanced robotic systems.

This holistic perspective is crucial for the successful implementation of modular and adaptable platforms.

To have a valid MVP/MRP, abstracting the mentioned fields into layers is optimal. We need a ready mechanical build, alongside a control and power circuit, then software, therefore we have mostly 3 critical layers:

-Mechanical layer.

-Electronic Layer.

-Software layer.

Our platform should provide these 3 layers as well as allow the customization and interchangeability of these elements when possible.

0.7 Conclusion

This chapter has established a superficial robotics foundation for the thesis. By exploring, the interdisciplinary framework that supports tinkering and development, the groundwork has been laid for the subsequent chapters. The next sections will build on these concepts by detailing the design, implementation, and experimental validation of the modular robotic platforms, thereby contributing to the evolution of accessible and adaptable robotics.

1 Mechanical foundations

1.1 Role of Mechanical Design in Modular Robotic Platforms

In a Modular Robotic Platform (MRP), mechanical design plays a fundamental role in enabling reusability, adaptability, and scalability. Unlike monolithic robotic systems, where mechanical components are tightly coupled to a single application, an MRP must support frequent reconfiguration while maintaining structural integrity and functional reliability.

The mechanical layer provides the physical substrate upon which electronic and software modules operate. Therefore, mechanical interfaces must be standardized, robust, and flexible enough to accommodate heterogeneous components such as sensors, actuators, power systems, and auxiliary structures. In the context of the ASCE platform family, mechanical modularity is treated as a first-class design constraint rather than a secondary optimization.

1.2 Design Principles for Mechanical Modularity

The mechanical foundations of the proposed platforms are guided by the following principles:

1.2.1 Reusability

Mechanical parts are designed to be reused across multiple robotic platforms whenever possible. Mounting patterns, enclosure dimensions, and structural elements follow repeatable geometries, allowing components to migrate between different MRPs without redesign.

1.2.2 Standardized Interfaces

Standard mechanical interfaces, such as mounting holes, brackets, and rails, enable consistent attachment of modules. This standardization reduces integration time and minimizes the risk of incompatibility during reconfiguration.

1.2.3 Structural Safety and Load Distribution

While modularity favors flexibility, mechanical safety remains critical. Load-bearing elements are designed to withstand worst-case operating conditions, and mass distribution is carefully considered to preserve stability and maneuverability across different configurations.

1.2.4 Scalability

Mechanical designs allow incremental expansion. Additional modules can be mounted without altering the core structure, ensuring that the platform can evolve alongside user requirements.

1.3 Mechanical Interfaces and Mounting Strategy

To support modular assembly, the platforms adopt a layered mechanical interface strategy:

Primary structural frame: Provides rigidity and defines the robots main geometry.

Secondary mounting zones: Dedicated areas for sensors, actuators, and peripherals.

Payload regions: Reserved volumes for experimental or user-defined modules.

Mounting strategies emphasize accessibility and reversibility, enabling users to assemble, disassemble, and modify configurations with minimal tools. This approach aligns with the MRP objective of rapid prototyping and iterative development.

1.4 Platform-Specific Mechanical Implementations

Although each platform targets different operational environments, all share the same modular mechanical philosophy.

1.4.1 uTomba

uTomba adopts a minimalistic mechanical structure designed to prioritize accessibility and learning. The use of a breadboard-based layout eliminates rigid enclosure constraints, allowing users to physically interact with components and explore alternative configurations. This mechanical openness supports rapid experimentation and educational use while still maintaining a coherent platform identity.

1.4.2 Bulky

Bulky is designed as a compact terrestrial rover with an emphasis on sensor and actuator expandability. Its chassis integrates multiple mounting points for ultrasonic arrays,

cameras, manipulators, and auxiliary modules. The mechanical design balances compactness with flexibility, enabling Bulky to operate both as a standalone system and as a peripheral-driven slave platform controlled by external computation units.

1.4.3 Thegill

Thegill extends mechanical modularity into amphibious environments. Its structure is engineered to support both terrestrial locomotion and aquatic propulsion while maintaining compatibility with additional systems such as robotic arms. Special consideration is given to sealing, buoyancy, and power routing, ensuring that modular components can be integrated without compromising environmental resilience.

1.4.4 Dronegaze

Dronegaze is conceived as a modular aerial test platform rather than a conventional flight-ready drone. Its mechanical design prioritizes safety and accessibility, functioning as a test jig that allows controlled experimentation with propellers, frames, and control parameters. This setup enables rapid validation of aerodynamic and control hypotheses while preserving reusability of components.

1.4.5 ILITE

ILITE's mechanical design is meant to serve as a drop resistant ergonomic model, easy on the hands and satisfying controls and feedback, with buttons and encoders accessible and long lasting, and for the least; aesthetically sound with ability to create multiple colour and style variants.

1.5 Mechanical Modularity and MRP Validation

Across all platforms, mechanical modularity enables the creation of multiple valid configurations that satisfy the criteria of an MRP or MVP. By reusing structural elements, adapters, mounting standards, and design principles, the platforms demonstrate that mechanical abstraction can coexist with domain-specific optimization.

This approach validates the mechanical layer of the MRP concept by showing that diverse robotic systems; terrestrial, amphibious, aerial, and educational; can be derived from unique or shared mechanical foundations without sacrificing functionality or safety.

1.6 Conclusion

This chapter presented the mechanical foundations of the proposed modular robotic platforms. By outlining key design principles, standardized interfaces, and platform-specific implementations, it has been shown that mechanical modularity provides a robust basis for reconfigurable robotic systems. These foundations enable the seamless integration of electronic and software layers, which are explored in the following chapter.

2 Electronic Foundation

2.1 Role of Electronics in a Modular Robotic Platform

In a Modular Robotic Platform (MRP), the electronic layer acts as the interface between mechanical structures and software logic. It is responsible for power distribution, signal acquisition, actuation, communication, and user interaction. In contrast to monolithic robotic systems, modular platforms must tolerate frequent reconfiguration, heterogeneous peripherals, and evolving performance requirements without requiring redesign of the core electronics.

The electronic architecture presented in this thesis emphasizes abstraction, reuse, and robustness. By relying on widely available components, standardized signaling protocols, and scalable microcontroller architectures, the platforms achieve electronic modularity across diverse robotic systems.

2.2 Power Architecture and Energy Storage

2.2.1 Battery Technologies

Mobile robotic platforms require compact, high-energy-density power sources. Two battery chemistries are employed:

- **Lithium-Ion (Li-Ion):** Used where mechanical robustness, stable discharge, and long cycle life are required.
- **Lithium Polymer (LiPo):** Selected for applications requiring high discharge rates and flexible form factors, such as mobile and aerial platforms.

Both battery types are widely adopted in robotics and embedded systems due to their favorable energy-to-weight ratios [11].

2.2.2 Voltage Regulation

Robotic systems integrate components operating at different voltage levels. Step-down (buck) converters are therefore used to regulate battery voltages to stable logic and peripheral supply rails. Popular and well-documented modules such as Mini360 and

regulators based on the LM2596 family are employed due to their high efficiency, thermal stability, and widespread availability [12].

Distributed regulation improves fault tolerance and enables independent scaling of subsystems.

2.3 Signal Integrity and Noise Management

Robotic electronics often combine low-level analog signals with high-current switching devices. To ensure reliable operation, the following practices are applied:

- Local decoupling capacitors near IC power pins
- Separation of high-current and low-signal paths
- Short signal traces for high-speed or sensitive signals
- Logical grounding strategies to minimize noise coupling

These techniques are well established in embedded and mixed-signal system design [21].

2.4 Input and Output Expansion

2.4.1 Digital Expansion

Shift registers are used to expand digital outputs while minimizing GPIO usage. By serially shifting bit patterns, multiple outputs can be controlled using a small number of microcontroller pins. This approach is effective for driving LEDs, relays, and generating software-based PWM signals [14].

2.4.2 Analog Expansion

Analog multiplexers (MUX) allow multiple analog sensors to share a single ADC channel. This technique is especially useful when integrating numerous sensors without increasing microcontroller complexity or pin count.

2.5 Actuation and Signal Amplification

Microcontroller GPIO pins are insufficient for directly driving motors, solenoids, or high-power loads. Therefore, signal amplification stages are introduced:

- **Bipolar transistors** for low-power switching

- **MOSFETs** for high-current applications due to low on-resistance
- Push-pull and H-bridge configurations for bidirectional control

These circuits form the basis of motor drivers and power switching stages across all platforms [11].

2.6 Sensors and Peripheral Modules

The platforms integrate a wide range of sensors and peripherals using standardized interfaces such as I^2C, SPI, and UART. Common peripherals include:

- Inertial Measurement Units (IMUs) for orientation and motion estimation; MPU6040
- Distance and proximity sensors (ultrasonic, infrared)
- Audio indicators such as buzzers and LEDs for feedback and debugging

Standardized interfaces ensure compatibility, reusability, and ease of integration.

2.7 Microcontroller Architectures and Selection

Microcontroller choice significantly impacts system scalability and performance.

2.7.1 8-bit Microcontrollers

Platforms such as ţTomba use 8-bit AVR microcontrollers (e.g., Arduino Nano) for educational purposes. These devices offer simplicity and accessibility but are constrained by limited RAM, flash memory, and processing capability [14].

2.7.2 32-bit Microcontrollers

More advanced platforms employ 32-bit microcontrollers, including the ESP32 and ESP32-S3 families. These devices, based on the Xtensa architecture, provide:

- 32-bit word size for faster computation
- Larger addressable memory spaces
- Integrated communication peripherals

ESP32-based controllers offer hundreds of kilobytes of SRAM and multiple megabytes of flash, enabling complex firmware, graphical interfaces, and communication stacks to coexist [6].

2.8 Implementation Across Platforms

The electronic design philosophy is applied consistently across all robotic systems:

- **μTomba and Dronegaze:** Breadboard-based circuits for rapid prototyping and education
- **Bulky:** Custom PCB (BULKER32) for robustness and signal integrity
- **Thegill:** GillerS3 (ESP32-S3) and GillerDUO (ESP32 + RP2040) for high-performance and distributed control

This progression from breadboard to custom PCB reflects the MVP-to-MRP maturation process.

2.8.1 BULKER32 design

2.8.2 GillerDUO design

2.8.3 GillerS3 design

2.8.4 Breadboard based designs

Since the primary reason this option exists is for pedagogic and teaching applications, it is helpful to include component kits and guides with the MRPs explaining the mechanism of breadboards and how to use them as well as circuit examples and templates as can be shown in the figures below.

with these 3 circuits, we have multiple options to realize MVPs that serve as customizable MRPs

2.9 ILITE Controller: Electronic Design Rationale

ILITE serves as a unifying control interface across the proposed Modular Robotic Platforms (MRPs). Its electronic design prioritizes robustness, scalability, and usability while maintaining compatibility with heterogeneous robotic configurations. The controller integrates human input devices, sensing, display, and processing within a compact and power-efficient architecture.

2.9.1 Human Interface Components

ILITE integrates multiple input modalities to support intuitive and reliable user interaction.

Analog Joysticks and potentiometers

Two analog joysticks are used to provide continuous two-dimensional control inputs. These devices typically operate within a 03.3 V or 05 V range and draw negligible current (on the order of microamperes), making them well-suited for low-power embedded systems. Analog joysticks interface directly with the microcontrollers ADC, enabling high-resolution input sampling and smooth control behavior [11].

Rotary Encoders

Incremental rotary encoders are employed for precise user input and navigation. Encoders offer advantages over potentiometers in terms of durability and repeatability, as they are not subject to mechanical wear of resistive elements. Their digital quadrature outputs allow accurate position and direction detection with minimal processing overhead [14].

Buttons with Hardware Debouncing

Push buttons are interfaced using a Schmitt triggerbased conditioning circuit implemented with a NOT gate, resistors, capacitors, and diodes. This approach introduces hysteresis and RC filtering to suppress contact bounce and noise at the hardware level. Hardware debouncing reduces software complexity and ensures reliable state transitions, particularly in noisy environments or during fast user interaction [21].

2.9.2 Visual Feedback Interface

An SH1106-based OLED display connected via the I^2C bus is used to provide graphical feedback. OLED technology was selected due to its low power consumption, high contrast ratio, and wide viewing angle. The SH1106 controller supports resolutions sufficient for menus, telemetry, and debugging information while consuming only a few tens of milliamperes during active operation [18].

The use of I^2C minimizes pin usage and allows the display to coexist with other peripherals on the same bus, reinforcing the modular philosophy of ILITE.

2.9.3 Microcontroller Architecture and Word Size

ILITE is designed around 32-bit microcontroller architectures, specifically the ESP32 and ESP32-S3 families. These devices are based on the Xtensa LX6/LX7 architecture and provide significant advantages over traditional 8-bit microcontrollers such as AVR-based devices.

Compared to 8-bit architectures, 32-bit microcontrollers offer:

- Wider word size, enabling faster arithmetic and control computations
- Larger addressable memory spaces
- Native support for advanced peripherals and communication protocols

ESP32-class devices typically provide hundreds of kilobytes of SRAM and several megabytes of external flash, allowing complex firmware, graphical interfaces, and communication stacks to coexist within a single system [6].

In contrast, 8-bit AVR microcontrollers, such as those used in Arduino Nanobased systems, offer limited RAM and flash memory, making them more suitable for educational platforms such as ţTomba but less appropriate for feature-rich controllers like ILITE [14].

2.9.4 Power and Current Considerations

The electronic components of ILITE are selected to operate within safe and efficient current limits. Input devices and displays draw relatively low currents, while the microcontroller represents the primary power consumer, particularly during wireless communication or intensive computation.

ESP32-based systems are capable of dynamic power scaling and low-power modes, allowing ILITE to balance performance and energy efficiency. Voltage regulation and decoupling strategies discussed earlier ensure stable operation even under varying load conditions.

2.9.5 Design Justification

The combination of robust input conditioning, efficient display technology, and a 32-bit microcontroller architecture enables ILITE to function as a scalable and reusable controller across multiple robotic platforms. By abstracting hardware complexity while retaining performance headroom, ILITE embodies the electronic principles of the Modular Robotic Platform concept.

2.10 Chapter Summary

This chapter presented the electronic foundations of the proposed modular robotic platforms. Through careful selection of power architectures, signaling techniques, microcontroller families, and implementation strategies, the electronic layer supports scalability, robustness, and reuse. These foundations enable the software abstractions and communication frameworks discussed in the following chapter.

3 Software Foundations

3.1 Overview

In a Modular Robotic Platform (MRP), the software layer is responsible for transforming heterogeneous hardware and assemblies into a consistent and reusable programming model. The key challenge is maintaining portability and development velocity across different robots (ground, amphibious, aerial test rigs) and different electronics (breadboard prototypes vs. custom PCBs), while still enabling deterministic behavior where needed.

This thesis adopts a layered software strategy: (i) a toolchain and framework layer for portability and developer productivity, (ii) an ASCE abstraction layer that standardizes access to hardware assemblies and platform circuits, and (iii) a real-time execution model for concurrency and responsiveness on supported microcontrollers.

3.2 Toolchain: C++ and PlatformIO

All firmware is developed in C++ due to its balance between low-level control, performance, and support for reusable abstractions (classes, templates, modules). The project is built using PlatformIO as the development environment manager. PlatformIO provides a unified build system, dependency/library management, and multi-platform/multi-architecture workflows, which is aligned with the needs of an MRP that targets different boards over time [16, 17, 15].

From an engineering perspective, the use of PlatformIO improves reproducibility and maintainability by making compiler settings, board targets, and dependencies explicit at the project level (e.g., via configuration files), rather than relying on IDE-specific state.

3.3 Arduino Framework as a Portability Layer

Firmware is based on the Arduino framework to benefit from its high-level API surface and portability across multiple microcontroller families. Arduino's programming model provides stable interfaces for common embedded operations such as GPIO, ADC,

PWM, serial communication, and timing, enabling code reuse across boards with minimal changes [3].

For ESP32-class platforms specifically, the Arduino core is implemented on top of Espressif's ESP-IDF stack, which provides access to vendor drivers and system services while preserving the Arduino programming experience [5, 1, 2]. This approach supports rapid iteration and cross-platform compatibility, while still allowing advanced features to be accessed when required.

3.4 ASCE Framework Layer

On top of Arduino, this thesis introduces ASCE's own framework as a reusable library layer. The goal of this layer is to formalize modularity in software by:

- Abstracting platform circuits (e.g., ILITE, BULKER32, GillerS3, GillerDUO) behind stable interfaces.
- Encapsulating recurring hardware patterns (sensor buses, actuator drivers, input devices, screen rendering) into reusable components.
- Hiding platform-specific details (pin maps, voltage assumptions, peripheral quirks) behind configuration structures, pre-definitions, and well-scoped APIs.

This design enables developers to implement robot-specific behaviors while minimizing the need to manage low-level technicalities repeatedly. It also supports the MVP-to-MRP maturation path: early breadboard builds can share the same programming model as later PCB-based builds, with differences handled internally by platform definitions.

3.5 Execution Model: FreeRTOS on ESP32 Platforms

Several target platforms are based on ESP32-class microcontrollers, where FreeRTOS support is a natural fit. ESP-IDF includes a FreeRTOS-based execution environment, including implementations that support dual-core scheduling on compatible ESP targets [8, 7].

Using FreeRTOS provides practical benefits for modular robotic systems:

- **Concurrency:** decomposition of firmware into tasks (e.g., sensing, control, communication, UI).
- **Responsiveness:** preemptive scheduling with priorities, allowing time-sensitive tasks to run reliably [10].

- **Structured communication:** queues, semaphores, and mutexes to coordinate subsystems safely [9, 13].
- **Scalability:** ability to add new modules as separate tasks without rewriting the entire control loop.

In the context of this thesis, FreeRTOS is used as an enabling mechanism for modular software composition. It supports separating robot behavior from low-level services (e.g., telemetry, user interface, communication framing), and it reduces coupling between subsystems.

3.6 Open-Ended Integration Points for This Thesis

The remainder of this thesis will ground the above foundations through concrete implementations, including:

- Platform drivers (breadboard assemblies vs. PCB-based platforms) and their configuration strategy.
- Task partitioning and scheduling approach on ESP32-based platforms.
- Examples of ASCE APIs that expose stable "contracts" for sensors, actuators, and user interaction.
- Communication patterns and debugging facilities used across the platform family.

These examples will demonstrate how portability (Arduino + PlatformIO), modular abstraction (ASCE framework), and concurrency (FreeRTOS) jointly support the MRP goals of reuse, flexibility, and scalability.

4 Realization of a Modular Robotic Platform

4.1 From Principles to Practical Implementation

The preceding chapters established the mechanical, electronic, and software foundations required to support a Modular Robotic Platform (MRP). This chapter demonstrates how these principles are applied in practice to realize complete, configurable robotic systems. Rather than treating modularity as a purely conceptual goal, the MRP framework introduced in this thesis is operationalized through formal component descriptions, compatibility criteria, and supporting software tools.

The objective of this chapter is to show how MRPs can be constructed, customized, deployed, and evolved using the concepts introduced earlier.

4.2 Component Registration and Attribute Modeling

At the core of the MRP concept is the notion of explicitly registered components. Each hardware or software component is described using a set of attributes that characterize its capabilities and constraints. Typical attributes include:

- Mechanical properties (dimensions, weight, mounting interface)
- Electrical characteristics (operating voltage, current consumption)
- Interface requirements (digital I/O, analog inputs, communication protocols)
- Software dependencies (drivers, services, configuration parameters)

By modeling components in this structured manner, it becomes possible to reason about compatibility and system composition without relying on ad-hoc integration decisions.

4.3 Compatibility Criteria and System Composition

Compatibility between components is determined by a set of explicit criteria. These criteria include:

- **Mechanical compatibility:** availability of mounting adapters, load limits, and spatial constraints.
- **Electrical compatibility:** power budget, voltage levels, and connector availability.
- **Interface compatibility:** sufficient digital or analog I/O, or support for required communication protocols.
- **Software compatibility:** availability of drivers and framework-level abstractions.

Using these criteria, a platform configuration can be validated as either a Minimum Viable Platform (MVP) or a complete Modular Robotic Platform (MRP). This validation process ensures that assembled systems are functional, safe, and aligned with user requirements.

4.4 Platform Builder System

The formalization of components and compatibility rules enabled the development of a *platform builder* system. This system allows users to configure custom robotic platforms by selecting components within defined constraints.

The platform builder was later implemented as part of an online shop website, enabling users to:

- Assemble custom platforms starting from minimal MVP configurations.
- Select from predefined MRP configurations optimized for common use cases.
- Customize platforms based on payload, sensing, actuation, and control needs.

This approach bridges engineering design with practical deployment, making modular robotics accessible to users without requiring deep technical expertise.

4.5 Software Tooling for Platform Interaction

Beyond hardware composition, software criteria play a crucial role in realizing MRPs. Dedicated software tools were developed to assist users in interfacing with their platforms, configuring behavior, and tuning control systems.

Examples include:

- An inverse kinematics solver for the Mech'Iane robotic arm, enabling intuitive motion control.

- A telemetry and tuning interface for the Dronegaze platform, allowing users to visualize system states and adjust PID parameters in real time.

These tools were implemented using high-level environments such as Processing, emphasizing rapid development and user accessibility while interfacing with the underlying embedded systems.

4.6 Feedback-Driven Platform Evolution

MVP and MRP configurations generated through the platform builder and user deployments serve as valuable feedback data. By analyzing component usage, configuration patterns, and user behavior, insights can be gathered regarding demand, common use cases, and platform limitations.

This data-driven approach supports efficient platform maturation by:

- Guiding hardware revisions and documentation efforts.
- Identifying high-value modules and underutilized components.
- Informing future software and framework improvements.

Such feedback loops are essential for evolving modular platforms in a controlled and sustainable manner.

4.7 User-Driven Platform Creation with ILITE

The ILITE controller plays a central role in enabling user-driven MRP creation. By exposing stable software interfaces through the ASCE framework, ILITE allows users to define custom platforms without managing low-level communication, scheduling, or hardware intricacies.

This capability empowers ILITE owners to:

- Create new MRPs tailored to specific applications.
- Extend existing platforms with additional modules.
- Experiment with novel configurations while relying on a stable backend.

In this way, ILITE functions not merely as a controller, but as a gateway to modular platform development.

4.8 Chapter Summary

This chapter demonstrated how the theoretical concepts introduced throughout the thesis are applied to realize practical Modular Robotic Platforms. By formalizing components, defining compatibility criteria, and supporting customization through software tools and services, the MRP concept is shown to be viable in real-world deployment. The resulting ecosystem enables scalable, user-driven platform creation and continuous evolution, setting the stage for the concluding discussion of contributions and future work.

General conclusion

This thesis explored the design, implementation, and practical realization of modular robotic platforms with the objective of improving adaptability, scalability, and accessibility in robotics development. By combining mechanical, electronic, and software abstractions into a unified framework, the work demonstrates that modularity can be effectively applied beyond theoretical constructs and realized in functional, deployable systems.

The primary outcome of this work is the definition and validation of the Modular Robotic Platform (MRP) concept. Through the development of multiple robotic platforms; ranging from educational and breadboard-based systems to mechanically and electronically sophisticated robotsthe thesis shows that modularity can be achieved consistently across heterogeneous domains.

On the mechanical level, standardized mounting strategies and reusable structural elements enabled platforms to be reconfigured and extended without redesigning the core structure. Electronically, the use of widely available components, scalable power architectures, and flexible input/output expansion techniques supported both rapid prototyping and robust operation. On the software side, a layered architecture built on C++, PlatformIO, the Arduino framework, and FreeRTOS enabled portability, abstraction, and concurrency while maintaining accessibility for users with varying levels of expertise.

The introduction of the ASCE framework and the ILITE controller further unified these layers by abstracting platform-specific details and exposing stable interfaces for application development. This allowed users to focus on behavior and experimentation rather than low-level implementation, reinforcing the MVP-to-MRP maturation path.

Beyond technical implementation, the thesis demonstrated practical deployment through a platform builder system and associated software tools. These systems enabled users to customize and order platforms, interact with them through dedicated interfaces, and provide feedback that informed iterative platform improvement. Together, these results validate the feasibility and usefulness of the proposed modular

robotics approach.

Despite the progress achieved, several limitations remain. The modularity introduced in this thesis relies on well-defined contracts and abstractions, which inherently introduce some overhead compared to highly optimized, single-purpose robotic systems. In performance-critical applications, this abstraction cost may limit achievable efficiency.

Additionally, while the platforms support a wide range of configurations, mechanical and electronic compatibility still depends on the availability of suitable adapters and interface modules. Expanding the ecosystem of standardized modules remains an ongoing challenge.

From a software perspective, reliance on high-level frameworks such as Arduino simplifies development but can obscure low-level behavior and limit fine-grained control in certain scenarios. Although access to lower layers is possible, it requires additional expertise and careful design to avoid breaking abstraction boundaries.

Finally, experimental validation in this work focused on functional feasibility and integration rather than exhaustive quantitative benchmarking. More systematic performance evaluations would be required to compare the proposed platforms directly against industrial or research-grade systems.

Several directions for future work naturally emerge from this thesis. Mechanically, further standardization of interfaces and the development of additional modular adapters would enhance interoperability and ease of assembly. Electrically, integrating more advanced power management and monitoring features could improve efficiency and reliability in complex configurations.

On the software side, extending the ASCE framework with additional services, formal configuration schemas, and automated validation tools would strengthen platform robustness. Deeper integration of simulation and digital twin approaches could also facilitate design exploration and testing prior to physical deployment.

From a systems perspective, expanding data collection from MVP and MRP deployments offers opportunities for data-driven platform evolution. Analyzing usage patterns, failure modes, and user behavior can guide future design decisions and documentation efforts.

Ultimately, the concepts presented in this thesis lay the groundwork for a continu-

ously evolving modular robotics ecosystem. By lowering barriers to entry while preserving extensibility and rigor, the proposed approach has the potential to support research, education, and experimentation in parallel with industrial developments.

This work demonstrates that modularity, when treated as a first-class design principle and supported by appropriate abstractions, can transform how robotic systems are developed and used. The Modular Robotic Platform concept presented in this thesis represents a step toward more adaptable, user-driven, and sustainable robotics, providing a foundation upon which future innovation can build.

Bibliography

[1] *Arduino as an ESP-IDF Component*. https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/esp-idf_component.html. Accessed 2025-12-17. Espressif Systems.

[2] *Arduino Core for the ESP32 (GitHub Repository)*. https://github.com/espressif/arduino-esp32. Accessed 2025-12-17. Espressif Systems.

[3] *Arduino Documentation*. https://docs.arduino.cc/. Accessed 2025-12-17. Arduino.

[4] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson Education, 2005.

[5] *ESP32 Arduino Core Documentation*. https://docs.espressif.com/projects/arduino-esp32/. Accessed 2025-12-17. Espressif Systems.

[6] *ESP32 Series Datasheet*. Espressif Systems. 2023.

[7] *FreeRTOS (IDF) — ESP-IDF Programming Guide*. https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/freertos_idf.html. Accessed 2025-12-17. Espressif Systems.

[8] *FreeRTOS Overview (ESP32) — ESP-IDF Programming Guide*. https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/freertos.html. Accessed 2025-12-17. Espressif Systems.

[9] *FreeRTOS Queues*. https://freertos.org/Embedded-RTOS-Queues.html. Accessed 2025-12-17. FreeRTOS.

[10] *FreeRTOS Task Scheduling*. https://www.freertos.org/Documentation/02-Kernel/02-Kernel-features/01-Tasks-and-co-routines/04-Task-scheduling. Accessed 2025-12-17. FreeRTOS.

[11] Paul Horowitz and Winfield Hill. *The Art of Electronics*. 3rd. Cambridge University Press, 2015.

[12] *LM2596 SIMPLE SWITCHER Power Converter Datasheet*. Texas Instruments. 2016.

[13] *Mastering the FreeRTOS™ Real Time Kernel*. Accessed 2025-12-17. FreeRTOS.

[14] Paul Monk. *Practical Electronics for Inventors*. 4th. McGraw-Hill Education, 2017.

[15] *PlatformIO as an ESP-IDF Third-Party Tool*. `https://documentation.espressif.com/projects/esp-idf/en/latest/esp32c61/third-party-tools/platformio.html`. Accessed 2025-12-17. Espressif Systems.

[16] *PlatformIO Documentation*. `https://docs.platformio.org/en/latest/`. Accessed 2025-12-17. PlatformIO.

[17] *PlatformIO: Your Gateway to Embedded Software Development*. `https://platformio.org/`. Accessed 2025-12-17. PlatformIO.

[18] *SH1106 OLED Controller Datasheet*. Sinowealth. 2016.

[19] Bruno Siciliano et al. *Robotics: Modelling, Planning and Control*. Springer, 2009.

[20] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2006.

[21] David Williams. *The Art of Digital Design*. McGraw-Hill, 2004.

[22] Mark Yim et al. "Modular Self-Reconfigurable Robot Systems". In: *IEEE Robotics & Automation Magazine* 14.1 (2007), pp. 43–52.