AIAA AVIATION Forum
8-12 June 2026, San Diego, CA
AIAA AVIATION 2026 Forum

10.2514/6.2026-4493

Biped: A Unified Embedded Hardware-Software Platform for Urban Air Mobility and Autonomous Applications

Simon Yu [*]
University of Illinois Urbana-Champaign, Urbana, IL 61801

Steven Zhu [†]
Multiped LLC, Urbana, IL 61802

Lui R. Sha [‡]
University of Illinois Urbana-Champaign, Urbana, IL 61801

Or D. Dantsker [§]
Indiana University, Bloomington, IN 47408

Autonomous systems have become a defining feature of modern technology, transforming transportation, logistics, and aerial mobility through intelligent perception and control. In emerging domains such as urban air mobility (UAM), where autonomous aerial vehicles are envisioned to operate within cities, reliable perception of the environment is fundamental to achieving safe and assured autonomy. Developing and validating these perception capabilities requires testbeds that are both realistic and accessible: low-cost, low-risk, and adaptable, especially for early-stage development. However, existing indoor and outdoor drone facilities are often expensive, risky, and time-intensive to operate, leaving a persistent sim-to-real gap between theoretical performance and physical deployment. This work presents *Biped*, a unified embedded hardware-software platform designed to accelerate the development and validation of perception pipelines for UAM scenarios as well as other autonomous applications. The Biped platform integrates sensing, computation, power management, and wireless communication into a single low-cost hardware module, simplifying assembly while enhancing hardware reliability and adaptability. Complemented by its modular firmware and desktop Biped Ground Station software stack, the Biped platform supports both onboard and remote workloads for various experimental setups. Biped is demonstrated on two configurations: a ground-based, two-wheeled self-balancing robot (TWSBR) and its integration with the Quanser helicopter, forming a low-risk and accessible aerial testbed. These capabilities establish Biped as a practical and extensible platform for developing and evaluating autonomy pipelines in safety-critical cyber-physical systems (CPS).

Nomenclature

AP	=	access point	*GPIO*	=	general-purpose input/output
BMS	=	battery management system	*GTSRB*	=	german traffic sign recognition benchmark
BSP	=	board support package	*HAL*	=	high-assurance layer
COTS	=	commercial off-the-shelf	*HPL*	=	high-performance layer
CPS	=	cyber-physical systems	*IMU*	=	inertial measurement unit
DAL	=	development assurance level	*ISR*	=	interrupt service routine
DAQ	=	data acquisition	*IoT*	=	internet of things
DMA	=	direct memory access	*IPC*	=	inter-process communication
DNN	=	deep neural network	*IR*	=	infrared
DoF	=	degree of freedom	*IRAM*	=	instruction random-access memory
DRAM	=	data random-access memory	*ISP*	=	image signal processor
EMI	=	electromagnetic interference	*JTAG*	=	joint test action group
ESD	=	electrostatic discharge	*LE*	=	low energy
ESP-IDF	=	Espressif IoT development framework	*LEDC*	=	LED controller

[*]Doctoral Candidate, Department of Electrical and Computer Engineering, jundayu2@illinois.edu
[†]Co-Founder, Division of Research and Development, stevenzhu@multiped.net
[‡]Professor, Siebel School of Computing and Data Science, lrs@illinois.edu
[§]Assistant Professor, Department of Intelligent Systems Engineering, AIAA Member, odantske@iu.edu

LiDAR	=	light detection and ranging	*TCP*	=	transmission control protocol
MCU	=	microcontroller unit	*TVS*	=	transient voltage suppressor
ML	=	machine learning	*TWSBR*	=	two-wheeled self-balancing robot
MPC	=	model-predictive control	*UAM*	=	urban air mobility
MPE	=	most probable explanation	*UART*	=	universal async receiver-transmitter
MTL	=	multi-task learning	*UDP*	=	user datagram protocol
MTU	=	maximum transmission unit	*VTOL*	=	vertical takeoff and landing
NASA	=	national aeronautics and space admin	*2S1P*	=	2-series 1-parallel
NPC	=	neural probabilistic circuit			
NSF	=	national science foundation	a_x, a_y, a_z	=	linear accelerations
PC	=	probabilistic circuit	e	=	controller error
PCB	=	printed circuit board	g	=	gravitational constant
PCLK	=	pixel clock	K_p, K_i, K_d	=	proportional, integral, and derivative gains
PD	=	power delivery	K_o	=	open-loop gain
PID	=	proportional-integral-derivative	r	=	controller reference
PSRAM	=	pseudo-static random-access memory	s	=	encoder steps
PWM	=	pulse-width modulation	t	=	time
RF	=	radio frequency	u	=	controller output
ROI	=	region of interest			
RR	=	round robin	β	=	low-pass filter smoothing factor
RTOS	=	real-time operating system	ω	=	angular velocity
SCCB	=	serial camera control bus	ω_f	=	filtered angular velocity
SoC	=	system-on-chip	ϕ, θ, ψ	=	roll, pitch, and yaw angles
SPI	=	serial peripheral interface	$\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$	=	roll, pitch, and yaw rates
STL	=	standard templated library			

I. Introduction

Autonomy has become a defining feature of modern intelligent systems, transforming sectors from transportation to logistics.[1–5] Advances in sensing, computing, and machine learning have enabled self-driving cars, autonomous aerial vehicles, and intelligent robots capable of operating with reduced or minimal human supervision.[6–9] These technologies have facilitated increasingly capable autonomy, yet their wider adoption in safety-critical applications remains constrained by the challenge of reliability and safety assurance under real-world operating conditions.[10,11] In recent years, a particularly promising domain of autonomy has been UAM, the envisioned network of on-demand, autonomous "air taxi" operations within metropolitan areas.[12–15] UAM aims to alleviate ground traffic congestion and shorten inter-city travel times; however, its success heavily depends on safe and assured autonomy: the ability of autonomous aircraft to accurately perceive their surroundings, make reliable decisions, and safely execute critical maneuvers such as takeoff and landing with minimal human oversight.[16–20] Perception during such scenarios is particularly challenging, requiring robust visual interpretation to accurately detect and recognize obstacles, as well as identify suitable landing zones in complex real-world environments. Failures in perception within such safety-critical contexts can directly undermine both operational safety and public and regulatory trust.

Therefore, ensuring safety and reliability in perception-driven autonomy requires systematic validation through controlled yet flexible experimental environments. Testbeds must be realistic yet accessible: low-cost, low-risk, and adaptable, especially for early-stage research. However, existing indoor and outdoor drone facilities are often costly, risky, and time-intensive to operate. Outdoor experiments demand weather-dependent flights, usually in large and remote fields, and carry the risk of hardware damage or loss,[21–23] while indoor environments depend on complex, expensive motion-capture or localization infrastructure and extensive safety equipment.[24–27] Consequently, many real-time and machine learning (ML) perception models and pipelines remain validated only in simulation or on datasets,[28–37] leaving a persistent sim-to-real gap between theoretical performance and physical deployment. To address this gap, this work

introduces *Biped*, a modular and embedded hardware-software platform enabling the rapid development and testing of perception pipelines. Biped integrates custom-designed hardware with a modular firmware and software stack to create a unified yet adaptable platform suitable for various ground and aerial autonomous applications and scenarios.

The Biped hardware builds upon an improved version of a custom-designed embedded module originally developed for an educational TWSBR chassis.[38] The latest design consolidates sensing, computation, power management, and wireless communication onto a single printed circuit board (PCB) hardware module. In addition to the integrated components, the hardware module features an expansion interface that provides power and data connectivity to additional hardware peripherals, such as external cameras, depth sensors, or light detection and ranging (LiDAR) modules. Beyond its hardware design, Biped incorporates a modular firmware and software stack that enables flexible operations of the hardware platform. The embedded firmware employs a layered design built on top of FreeRTOS,[39] providing reusable and expandable modules for control, sensing, actuation, and planning. These modules allow researchers to rapidly reconfigure or extend system functionality, for instance, by swapping controllers, adding new sensor drivers, or implementing additional modules, without altering the firmware core. Complementing the firmware is a desktop application called Biped Ground Station, which manages real-time telemetry, data visualization, and controller parameter tuning. The Biped Ground Station also serves as a backend interface for remote computation, allowing heavy workloads such as inference to be offloaded to a remote host or cloud.

In this work, the Biped platform is integrated with the Quanser helicopter,[40] a widely used experimental platform for education and control research. Biped replaces Quanser's traditional host-dependent interface to form an untethered, perception-enabled aerial testbed capable of performing onboard real-time tasks through its modular firmware while supporting remote workloads through the Biped Ground Station. The integrated testbed enables low-risk and accessible evaluation of perception pipelines in autonomous and safety-critical UAM scenarios. The proposed aerial testbed supports user experiments utilizing popular ML-based perception models, such as YOLO[41] and ResNet,[42] in the context of UAM vertical takeoff and landing (VTOL). Using both the onboard camera and depth sensors connected externally via the expansion interface provided by the testbed, the user experiments may adopt either image-only or multi-modal region of interest (ROI) perception paradigms for object detection and recognition during UAM VTOL. Although the Quanser helicopter serves as a representative chassis in this work, Biped is designed as a general-purpose platform that can be easily integrated with any other compatible aerial or ground chassis by conforming to its standardized power and data interfaces, as shown in Figure 1. With Biped's cross-platform adaptability, additional user experiments may be conducted on a ground-based TWSBR chassis, including tasks such as control balancing and LiDAR-based sensing.

The primary contributions of this work are as follows: 1) an embedded hardware architecture that integrates sensing, actuation, computation, power management, and communication within a unified, low-cost form factor; 2) a modular firmware architecture that provides a layered and extensible framework for sensing, control, planning, communication, interrupt servicing, and other functionalities; 3) the Biped Ground Station software framework that supports real-time telemetry, visualization, parameter tuning, and remote computational workloads; and 4) cross-platform integration that demonstrates the deployment and reconfiguration of the Biped platform across both aerial and ground-based chassis.

II. Related Work and Motivation

This section reviews relevant work in three particular areas that collectively motivate Biped's design: (A) Quanser helicopter, a representative chassis used for Biped's aerial integration, (B) TWSBRs, the foundation of Biped's hardware design, and (C) safety-critical perception pipelines, which serve as guidelines to Biped's functional requirements.

A. Quanser Helicopter

The Quanser 2-degree of freedom (DoF) and 3-DoF helicopters[40] are widely used academic platforms for education and control systems research. Most prior work involving the Quanser platforms focuses primarily on control theory and implementation using MATLAB/Simulink running on a host machine.[43–47] In this setup, the helicopter hardware functions solely as an actuator-sensor interface, connected to the host through Quanser's proprietary data acquisition (DAQ) boards[48] and powered by Quanser's bulky universal power modules,[49] as seen in Figure 2. While effective

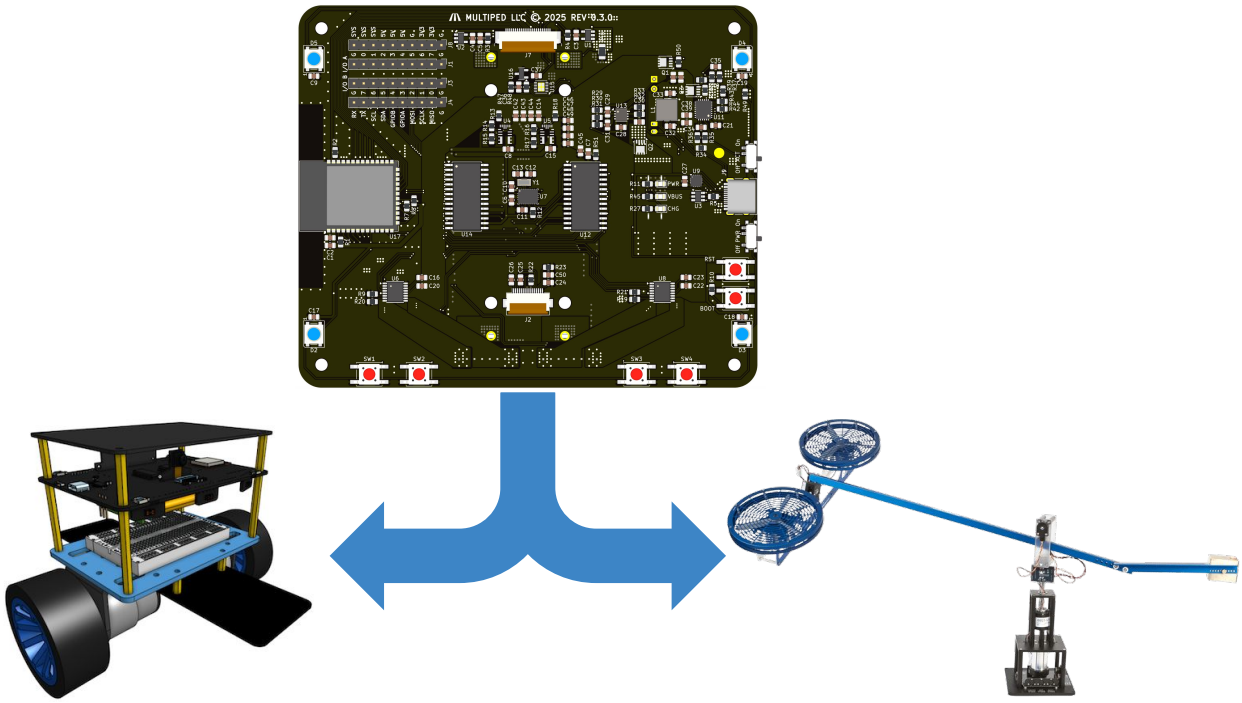

Figure 1. Biped hardware module interfacing with compatible aerial or ground chassis.

for traditional control studies, this architecture introduces several practical limitations: 1) all sensing and actuation computations rely on the host computer, preventing standalone or untethered operation, 2) sensing is restricted to encoders, with no integrated visual or spatial capabilities for perception-driven experiments, and 3) proprietary interfaces limit extensibility, making it difficult to integrate and adapt the platform for perception research.

To overcome these limitations, the Biped platform is integrated with the Quanser helicopter, forming an aerial testbed designed for perception research in aerial contexts. As shown in Figure 2, the Biped hardware module features a much more compact and self-contained design compared to Quanser's proprietary data and power interface. Equipped with onboard processing through its real-time firmware, wireless communication, and integrated sensors, including a smart inertial measurement unit (IMU)[51] and an onboard camera module, the Biped platform enables self-contained yet expandable operations, eliminating reliance on a MATLAB/Simulink host computer. While the Quanser helicopter serves as a *representative chassis* in this work, the Biped hardware design is chassis-agnostic and can be integrated into any other aerial or ground systems via its standardized power and data interfaces.

When coupled with additional visual or spatial sensors, such as LiDARs or infrared (IR) cameras, via Biped's hardware expansion interface, the integrated testbed supports experiments involving perception pipelines in aerial and UAM contexts. Operated indoors with minimal kinetic energy, the proposed integrated testbed provides a low-risk and easily reproducible environment for testing early-stage perception models and pipelines that would otherwise depend on risky, costly, and time-consuming free-flying drones. In contrast to indoor drone facilities, seen in Figure 2, with motion-capture arrays, safety equipment, and other specialized infrastructure,[24,25] the proposed testbed operates safely on a tabletop, offering a compact, accessible, and cost-effective alternative for early-stage perception research.

B. Two-Wheeled Self-Balancing Robot

The Biped hardware design originated from an educational TWSBR, developed as a compact, low-cost platform intended for education in control and embedded systems.[38] In contrast to Biped's unified architecture, most prior TWSBR designs rely on separate hardware modules for computation, actuation, and power management, often stacked vertically

Figure 2. Left: a typical indoor drone facility requiring a large room space and extensive infrastructure.[50] Right: comparison between the original Quanser helicopter power and data interface on the left side and the much more compact Biped hardware module with a generic, consumer-level power adapter on the right side.

and connected through complex wiring,[52–56] as seen in Figure 3. Such fragmented architectures complicate assembly, reduce flexibility, and often constrain each design to a specific mechanical chassis. In comparison, Biped unifies various components and functionalities onto a single PCB hardware module, simplifying interconnections and improving reliability. Biped's power and data interfaces allow the same module to be easily detached and reintegrated with any compatible ground or aerial chassis. This unified hardware approach enables Biped's cross-platform adaptability and establishes the foundation for its integration with the Quanser helicopter in this work.

In addition to its unified design, Biped introduces several practical hardware and software advantages over existing platforms. Many prior systems[52,54–56] rely on Arduino-based microcontroller units (MCUs) with limited processing capability and Bluetooth communication constrained by low bandwidth. In contrast, Biped employs a dual-core ESP32[57] MCU running FreeRTOS,[39] a popular and lightweight real-time operating system (RTOS) that supports concurrent and prioritized execution of sensing, control, and communication tasks. Furthermore, compared to Bluetooth, Biped features Wi-Fi connectivity that provides low-latency, high-bandwidth communication, allowing certain computationally intensive perception workloads to be offloaded to remote computational resources through the Biped Ground Station interface. Despite these capabilities, Biped is built entirely from commercial off-the-shelf (COTS) components and can be assembled for under $200, ensuring its affordability for research and educational use.

C. Safety-Critical Perception Pipelines

Recent publications in literature have emphasized safety-critical perception pipelines that not only detect and recognize objects but also ensure verifiable and timely responses under uncertainty. A key approach to achieving safety assurance in perception is through verification[32] and redundancy,[30,33] where layered perception architectures separate complex, learning-based mission capabilities from deterministic, verifiable safety capabilities that monitor or override any unsafe system outputs. These designs aim to guarantee collision-free operation through the safety layer even when mission-layer perception fails or becomes unavailable. The proposed Biped platform naturally supports such design philosophies. The verifiable safety layer, responsible for safety-critical obstacle detection and collision avoidance, can be integrated into Biped's modular firmware onboard. Meanwhile, the complex mission layer, which handles mission-level contextual and semantic recognition, can run remotely through the Biped Ground Station interface. This configuration closely mirrors the organization of these safety-critical perception frameworks, making Biped an effective experimental platform for evaluating similar pipelines in physical environments.

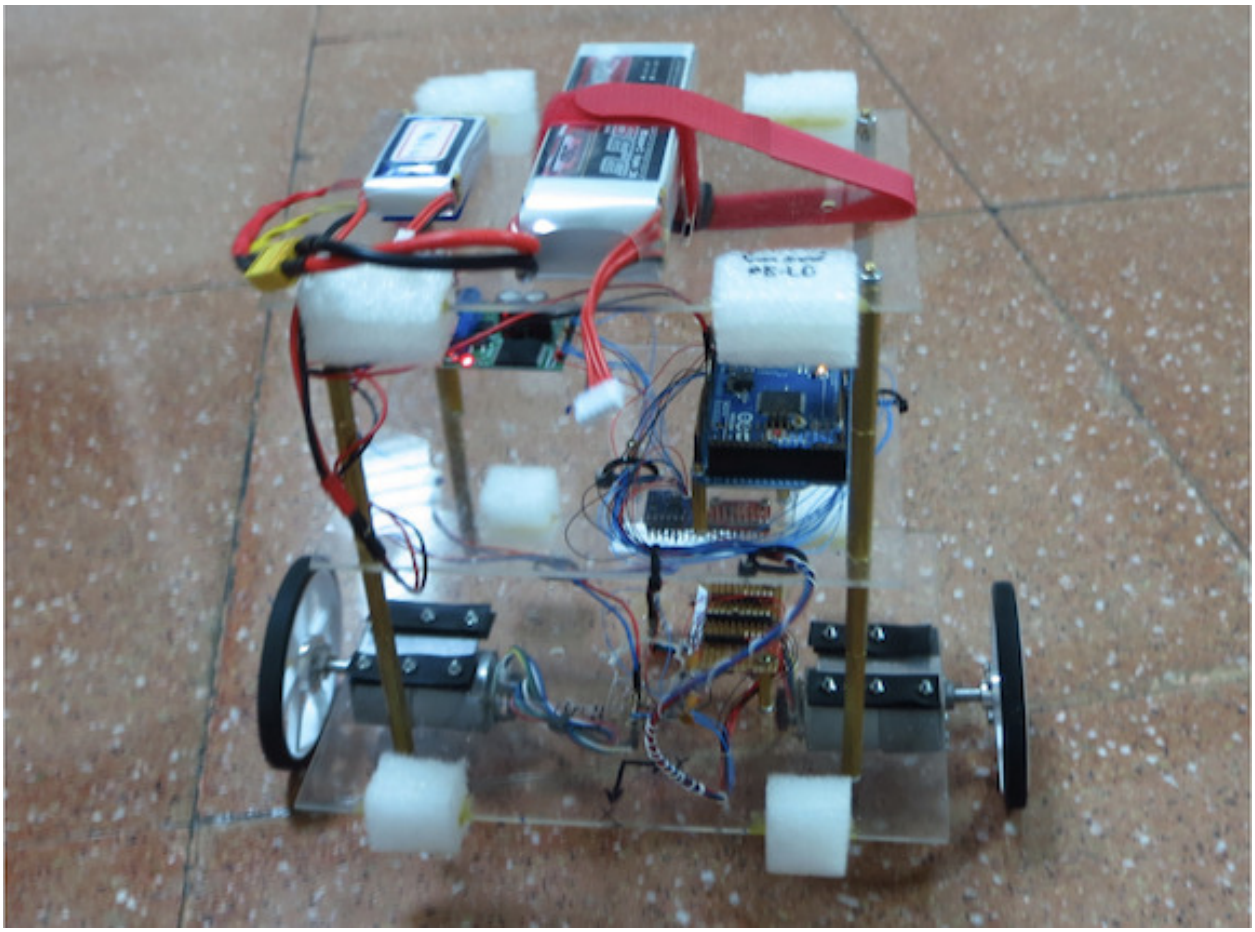

Figure 3. A typical TWSBR design with separate hardware modules connected through complex wiring.[52]

Another notable research direction addresses the problem of algorithmic priority inversion in safety-critical perception. Conventional vision pipelines process entire input frames uniformly, causing high-risk, nearby objects to be analyzed concurrently with, or even after, distant, low-risk ones. LiDAR-first perception approaches[31,34–36] mitigate this issue by using depth or LiDAR sensors to identify ROIs before camera-based recognition, assigning higher scheduling priority to closer or faster-moving objects. Lightweight, depth-based ROI detection algorithms, such as Depth Clustering,[58,59] have demonstrated real-time performance on embedded platforms, enabling onboard collision-avoidance decisions based purely on spatial occupancy. Recognition of prioritized ROIs, in contrast, can be offloaded as remote tasks through the Biped Ground Station, since object recognition contributes to mission-level awareness rather than immediate collision safety.[30,33] By combining onboard processing with remote workloads, the Biped platform is well-positioned as a viable testbed for validating pipelines in these categories.

III. Hardware Architecture

The Biped hardware module, as shown in Figure 4, consolidates embedded computation, I/O, sensing, power management, actuation, and user interaction into a unified, single PCB hardware architecture. An embedded MCU forms the core of the system, with I/O expanders extending its connectivity to onboard and external peripherals. Onboard sensing is achieved through the camera module and IMU, which provide visual and inertial measurements for perception and control. The integrated power system manages both external and battery sources, distributing power rails throughout the module. The actuation complex delivers power and control to a range of compatible actuators, while the

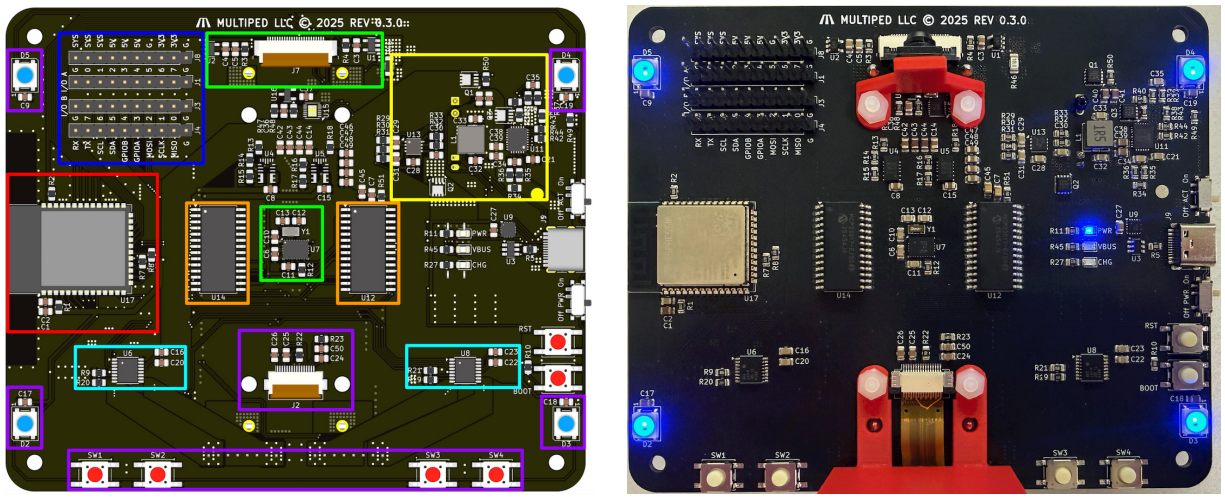

Figure 4. Left: front side of rendered Biped hardware module PCB with highlighted boxes indicating each subsystem complex. Red is the MCU. Orange are the I/O expanders. Yellow is the power complex. Green are the sensing components: camera connector (top) and IMU (center). Cyan are the actuation drivers. Blue is the expansion interface. Purple are the user interface components: NeoPixel RGB LEDs (corners), OLED display connector (lower center), push buttons (bottom). Right: front side of fabricated Biped hardware module PCB.

expansion interface supports additional external devices requiring data or power connections. Finally, the user interface combines the OLED display, status indicators, and push buttons to provide real-time feedback and interactive operation.

A. Microcontroller Unit

At the core of the Biped hardware module is the ESP32-S3[60] MCU, a low-cost and high-performance dual-core state of charge, system-on-chip (SoC) widely adopted for embedded and internet of things (IoT) applications.[57,61] The ESP32-S3 SoC integrates two 240-MHz Xtensa LX7 cores, providing onboard resources for concurrent real-time control and communications tasks. Additionally, the SoC features a built-in, full-speed USB 2.0 serial and joint test action group (JTAG) controller, which simplifies programming, debugging, and data logging compared to earlier ESP32 variants that required external USB universal asynchronous receiver-transmitter (UART) controllers.

Biped employs the ESP32-S3-WROOM-1[62] module, which integrates the ESP32-S3 SoC with 8 MB of serial peripheral interface (SPI) pseudo-static random-access memory (PSRAM), providing expanded non-volatile and volatile memory for task execution, dynamic memory allocation, and data-intensive workloads. The use of the pre-certified WROOM module also streamlines radio frequency (RF) integration with its factory-tuned antenna design, eliminating the need for impedance matching or certification testing of external antenna traces. Integrated Wi-Fi and Bluetooth low energy (LE) 5.0 connectivity enables high-bandwidth, low-latency telemetry and data transmission, supporting applications such as camera image streaming for remote perception workloads.

B. I/O Expander

While offering ample computational and wireless capability within a compact form factor, the ESP32-S3-WROOM-1 module exposes a limited number of general-purpose input/output (GPIO) pins due to constraints on the ESP32-S3 SoC package. To accommodate the wide range of components on the Biped hardware module, including motor drivers, camera module, power management, IMU, and communication buses, two MCP23017[63] digital I/O expanders are incorporated, as shown in Figure 5. Each I/O expander provides two ports of 8 programmable pins, totaling 16 bidirectional digital pins per I/O expander, all addressable via the I^2C bus. Between the two I/O expanders onboard the Biped hardware module, one is allocated as the system I/O expander for all onboard peripheral and sensor components, while the other is mapped to the expansion interface to support external peripherals.

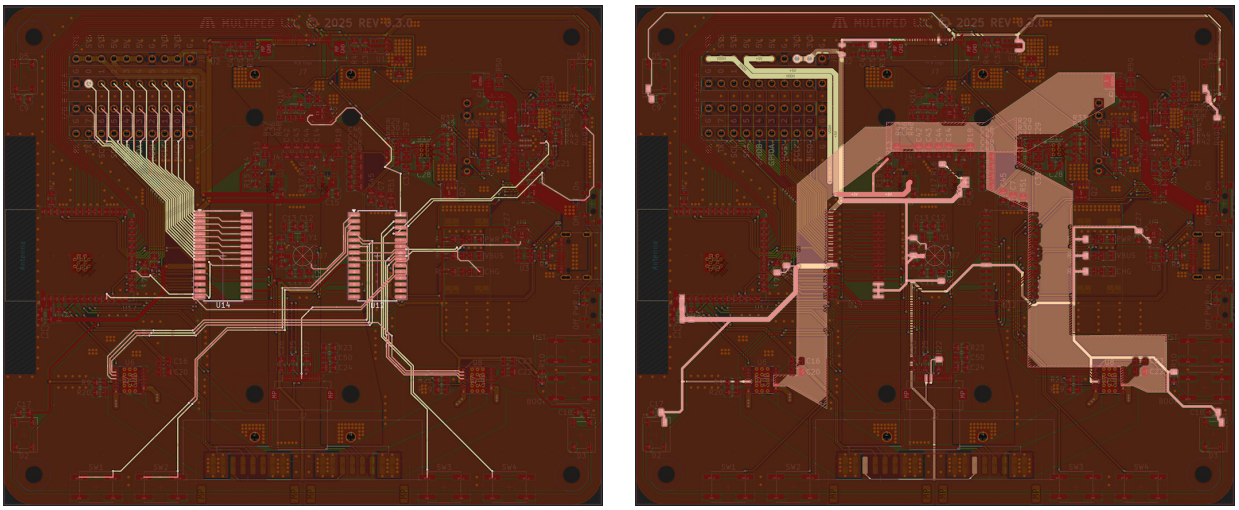

Figure 5. Left: I/O expanders and their connections. Right: power rails.

Due to the latency constraints of the I^2C protocol, the I/O expander pins are best suited for low-speed digital operations such as user inputs and outputs, discrete control signals, or interrupt-driven sensor signals that do not require high-speed timing. Each I/O expander supports interrupts on all of its 16 pins, with configurable interrupt-on-level or interrupt-on-change modes. When an interrupt condition occurs, the I/O expander asserts its interrupt line connected directly to the MCU, prompting the MCU to read its status registers over I^2C and handle the I/O expander interrupt. Based on the retrieved register data and the configured interrupt types for each I/O expander pin, the MCU firmware identifies the source I/O expander pin and invokes the corresponding interrupt service routine (ISR) attached to the pin.

C. Power

1. External Power Supply

The Biped hardware module supports USB power delivery (PD) through an onboard FUSB302B[64] USB PD transceiver. The ESP32-S3 MCU communicates with the USB PD transceiver over the I^2C bus, commanding it to negotiate appropriate voltage and current levels with the connected USB-C PD-compliant external power supply, such as a wall adapter or computer. The previous educational design[38] of the Biped hardware module relied on the standard USB 2.0 power interface, which is limited to only 2.5 W at 5 V. In contrast, the current design of the Biped hardware module supports a maximum power capacity of 60 W at 20 V through USB PD, providing both substantially faster battery charging and the additional power headroom required by high-power actuators, such as the Quanser helicopter motors.

To further improve system reliability over the USB-C connection, transient voltage suppressor (TVS) diodes are placed across the USB data lines. In the earlier design, static charge accumulation on the TWSBR chassis during its operations on the floor often led to electrostatic discharge (ESD) events when a USB cable was inserted. These discharges occasionally induced voltage spikes across the USB lines and power rails, triggering spontaneous MCU resets during flashing or serial communication. The addition of TVS diodes mitigates such transient surges, protecting the onboard electronics and ensuring reliable communication and power delivery over USB-C.

2. Battery Management System

The prior design of the Biped hardware module incorporated a generic, two-cell COTS battery pack, chosen primarily for safety and cost in classroom settings. The battery management system (BMS) from the pack was placed directly onto the module PCB as an independent component, with the battery cells rewired and connected through a JST connector. While the two soft polymer pouch cells provided sufficient energy for the low-power TWSBR chassis, the design lacked

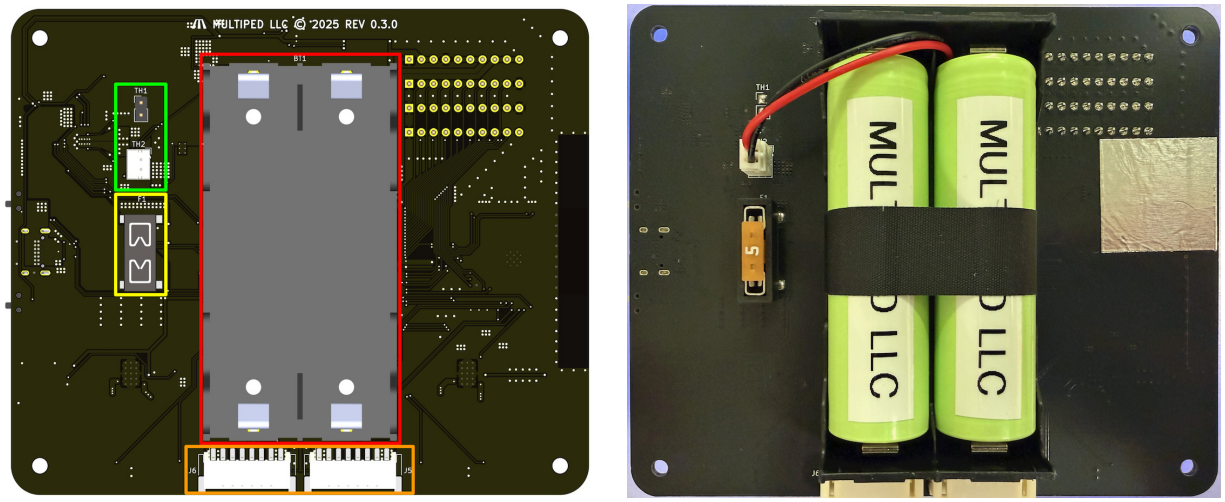

Figure 6. Left: back side of rendered Biped hardware module PCB with highlighted boxes indicating each subsystem complex. Red is the 18650 battery holder. Orange are the actuation interface connectors. Yellow is the fuse holder. Green are the thermistor connectors. Right: back side of fabricated Biped hardware module PCB with 18650 battery cells, fuse, and thermistors installed.

the flexibility and performance needed for higher-power applications. The BMS from the COTS battery pack operated as a closed system with no data interface or programmability, relying on the standard USB 2.0 power interface, which is limited to 2.5 W. Additionally, the BMS could only draw power from the battery output, as no power path existed to utilize power directly from the external power supply. This configuration constrained the overall power capacity to the combined discharge rate of the battery cells, which was adequate for the TWSBR platform but marginal for higher-power actuators, such as those on the Quanser helicopter.

The current design replaces the BMS from the COTS battery pack with a fully integrated, programmable BMS centered on the MP2760[65] buck-boost charger. The MP2760 supports charging currents up to 6 A and features built-in power-path management that enables the system to draw power directly from an external power supply while charging the batteries. The charger automatically switches between the external power supply and battery power as needed, providing seamless operation under varying power conditions. The power-path management greatly enhances system flexibility: for stationary configurations such as the Quanser helicopter, the Biped hardware module can operate entirely from the external power supply with the battery cells removed, whereas mobile configurations such as the TWSBR chassis can function solely on battery power. The charger also incorporates active safety features, including temperature protection and adaptive current throttling governed by two onboard thermistors, one monitoring ambient temperature and the other attached to the battery cells. In addition, the MP2760 communicates with the ESP32-S3 MCU via the I^2C interface, allowing the MCU firmware to configure power and charging parameters, monitor real-time power and temperature readings, and receive status or fault alerts.

The Biped hardware module is powered by two standard 18650 lithium-ion battery cells connected in a 2-series 1-parallel (2S1P) configuration. Standard 18650 cells offer high energy density, reliability, and broad compatibility, with a lower risk of swelling or puncture compared with soft polymer pouch cells. The 18650 cells are placed in a battery holder on the back side of the Biped hardware module, as shown in Figure 6. Therefore, the cells can be easily removed, replaced, or charged externally using standard COTS 18650 chargers, offering convenience during extended experiments. This design achieves a practical balance among energy capacity, reliability, and usability. To maintain voltage balance between the 2S1P 18650 cells, a BQ29209[66] cell balancer is integrated onto the module. The balancer equalizes cells and prevents over-voltage during charging, enhancing both safety and lifespan.

3. Distribution and Regulation

The Biped hardware module employs a structured multi-rail power distribution architecture, as illustrated in Figure 5, derived from the system power rail, i.e., the output of the BMS complex. The BMS power-path management derives the system power rail either directly from the battery, from the external power supply, or from both, depending on the current system power demand and the battery's state of charge. From the system power rail, two downstream DC-DC converters generate regulated 3.3 V and 5 V power rails. The 3.3 V power rail supplies all onboard electronics, such as the MCU, IMU, I/O expanders, and the camera module. The 5 V power rail, on the other hand, is routed through the expansion interface to power external hardware peripherals that require higher voltage levels. Meanwhile, the unregulated system power rail feeds directly into the actuation complex to maximize the actuator power delivery.

In its current configuration, the system power rail operates at the charging voltage of the 2S1P 18650 cells at approximately 8.5 V. This level provides sufficient power for both the TWSBR actuators and the Quanser helicopter platform, albeit operating near the lower bound of the Quanser helicopter's power requirements. Alternative designs involving boosting the system power rail to higher regulated voltage levels were considered; however, the present configuration was found to be adequate for stable operation while maintaining system simplicity and efficiency.

4. Monitoring and Control

The Biped hardware module integrates a MAX17261[67] Coulomb-counting fuel gauge to enable accurate estimation of the battery level. During operation, the connected actuators may draw high, pulsed currents; therefore, instantaneous voltage and current measurements are unreliable indicators of the remaining battery capacity. Instead, MAX17261 continuously measures charge and discharge currents to maintain a running record of cumulative energy usage, allowing the MCU firmware to accurately estimate the battery level and determine when recharging is required. The ESP32-S3 MCU communicates with the fuel gauge over the I^2C bus to access its data and control registers, while an interrupt line routed to the system I/O expander is used by the gauge to report status or fault conditions.

In addition to power monitoring, the module incorporates a dedicated power switch and three status indicator LEDs for its power system. The power switch controls the connection between the MP2760 charger output and the system power rail, allowing the module to be fully powered down to conserve energy when not in use while still permitting battery charging when an external power supply is connected. Two of the indicator LEDs provide hardware-level status feedback, showing whether system power is enabled and whether external power is detected at the USB-C port. The third LED is connected to the system I/O expander and can be controlled by the MCU firmware to indicate configurable power conditions, such as critical battery levels or charging status reported by the fuel gauge and charger.

D. Sensing

1. Camera

Biped employs an OmniVision camera sensor, specifically OV2640[68] or OV5640,[69] for onboard visual perception. The camera module features a 10-bit parallel data interface synchronized by a dedicated pixel clock (PCLK), which outputs a clock pulse to the MCU for every transmitted pixel. As seen in Figure 7, only 8 of the 10 data lines are utilized for the Biped design, as the ESP32-S3's camera hardware peripheral supports either 8-bit or 16-bit parallel data widths, and the remaining GPIO pins are reserved for other peripherals. This configuration strikes a balance between image dynamic range, bandwidth utilization, and available I/O. The ESP32-S3 MCU acts as a client receiver on this parallel interface, continuously capturing pixel data streamed from the camera. The ESP32-S3's built-in direct memory access (DMA) controller is configured to autonomously transfer image data streamed by the camera module directly into pre-allocated frame buffers in the MCU memory without consuming CPU cycles. The DMA enables continuous image data streaming with minimal processor overhead, allowing the firmware to dedicate computational resources to higher-level tasks, such as control, interrupts, or communication, while the camera operates in parallel. Camera configuration and control are managed via the serial camera control bus (SCCB),[70] a lightweight two-wire serial interface similar to I^2C. Since SCCB and I^2C are electrically compatible, they share the same signal lines on the Biped hardware module.

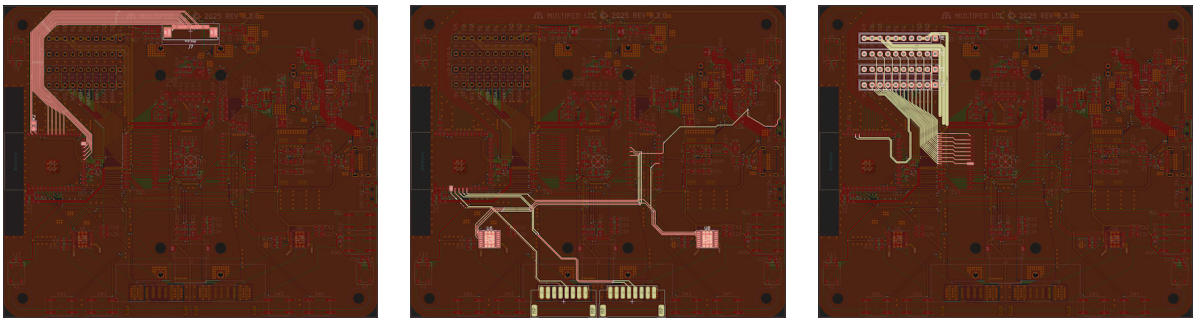

Figure 7. Left: camera connector. Center: actuation drivers and interface connectors. Right: Expansion interface.

2. *Inertial Measurement Unit*

The previous Biped design employed two discrete IMUs: the MPU6050,[71] a 6-axis IMU providing raw linear acceleration and angular velocity measurements, and Bosch BMX160,[72] a 9-axis IMU that was later added to provide compass readings with its integrated magnetometer. However, the raw compass output from the BMX160 proved highly sensitive to environmental interference and remained accurate only within a confined region even after initial calibrations. This limitation necessitated additional fusion and filtering on the MCU, which reduced the reliability and performance of attitude sensing during operation. The current Biped hardware module replaces both sensors with a single Bosch BNO055,[51] a smart 9-axis IMU integrating a high-efficiency 32-bit Arm Cortex-M0+ co-processor running Bosch's high-performance proprietary sensor-fusion firmware. The BNO055 provides the same raw accelerometer, gyroscope, and magnetometer data as usual, but performs continuous 9-DoF sensor fusion, self-calibration, and active compensation and filtering. Consequently, the BNO055 outputs attitude data via the I^2C bus directly in Euler angles or quaternions, eliminating the need for onboard sensor fusion or filtering algorithms on the MCU. This integration improves attitude sensing accuracy and real-time control performance for both ground and aerial configurations.

E. Actuation

The actuation complex of the Biped hardware module, shown in Figure 7, is designed to provide a powerful interface for driving various pulse-width modulation (PWM) compatible actuators. To support high-power actuators, such as the DC motors on the Quanser helicopter, the design replaces the single TB6612FNG dual H-bridge[73] used in the previous design with two DRV8874 high-performance H-bridge drivers.[74] While the TB6612FNG provides only 1.2 A of continuous output current and a limited voltage range, sufficient for the low-power TWSBR chassis, each DRV8874 supports up to 6 A of peak current output with substantially lower conduction losses. This improvement increases the available actuation power and torque capacity, enabling more efficient operation during spin-up and under high-speed PWM transitions. Each DRV8874 also integrates hardware-based current regulation for current-limited startup and stall protection, as well as spread-spectrum clocking to reduce electromagnetic interference (EMI).

DRV8874 drives the actuators by a combination of PWM and direction control signals. The ESP32-S3 MCU generates PWM waveforms corresponding to the desired actuation magnitude by modulating the given PWM control values into digital duty cycles, while direction control for each channel is managed through a single digital line routed through the system I/O expander. For improved usability and safety during experiments, a dedicated actuation enable switch is integrated to control the enable pin of the H-bridge drivers. In the previous design, experiments on the TWSBR chassis were often hindered by active or misbehaving actuators, while physically disconnecting their power connections or switching off the entire module was cumbersome. The addition of the actuation enable switch allows the MCU and other onboard electronics to remain powered and programmable while temporarily disabling the actuators, greatly simplifying testing and reducing the risk of hardware damage or operator injury.

Each actuation channel is accessed through a pair of dedicated connectors that serve as a standardized power and data interface for external chassis integration. Each connector provides high-power output pairs from the actuator

drivers, along with the power output from the regulated 3.3 V power rail and a pair of MCU digital pins for integrated sensor connections, such as rotary encoders on the TWSBR chassis actuators. This standardized interface simplifies adaptation to different robotic platforms. Users can integrate the Biped hardware module into any compatible chassis by conforming their actuator and sensor wiring to the actuation interface, together with any additional peripherals through the expansion interface. This design approach enables the cross-platform adaptability of the Biped hardware module, enabling rapid integration across diverse hardware configurations.

F. Expansion Interface

To enhance expandability and support additional hardware peripherals, the Biped hardware module integrates a versatile expansion interface, as seen in Figure 7. One of the two onboard I/O expanders is dedicated to this interface, providing 16 additional digital GPIO and interrupt pins. These additional I/Os enable users to connect external devices and components, such as buttons, relays, or digital sensors, without consuming the primary high-speed MCU pins. In addition to the GPIO pins, the interface exposes the I^2C and UART buses for advanced peripherals and reserves two direct MCU GPIO pins for high-speed, low-latency operations. An additional SPI bus is also integrated into the expansion interface to further extend compatibility with high-bandwidth peripherals. Furthermore, the regulated 3.3 V and 5 V power rails, along with the system power rail, are also available through the same expansion header, supporting a broad range of external devices with varying power demands. Through its combination of data and power connectivity, the expansion interface enables seamless integration of external modules, making the Biped hardware module a versatile platform for control and perception research. In the Quanser helicopter configuration, the two direct MCU GPIO pins and the 5 V power rail are used to interface with the third-axis rotary encoder. On the TWSBR chassis, a rotary 2D LiDAR sensor[75] is connected via the UART serial bus and 5 V power rail.

G. User Interface

An SH1106G[76] OLED display is integrated into the Biped hardware module to provide onboard logging, debugging, and system status monitoring. Before its integration, all diagnostic and status outputs had to be performed through serial communication, which was cumbersome during standalone operation. With the OLED display, essential information can be presented immediately upon power-up, eliminating the need for a host computer connection. In the previous design, the OLED module communicated via the shared I^2C bus with other onboard peripheral components such as the IMU and I/O expanders. While minimizing pin usage and electrical complexity, this configuration also increased I^2C bus traffic during display refreshes, which consumes moderate data bandwidth, occasionally causing higher-priority tasks to preempt OLED I^2C transmissions, resulting in intermittent graphical glitches.

In the current design, the SH1106G OLED display interface is connected through a dedicated SPI connection, which provides substantially higher data throughput and independent bus arbitration. The SH1106G is also driven by a dedicated DMA channel on the ESP32-S3 MCU, allowing graphical updates to occur in parallel without consuming MCU RTOS scheduling budgets or I^2C bandwidth. As previously mentioned, the SPI interface is also exposed through the expansion interface. By routing SPI control signals such as chip select through the expansion GPIO pins, the expansion interface supports an arbitrary number of external SPI devices.

In addition to the OLED display, an array of four addressable RGB NeoPixel LEDs[77] provides programmable visual feedback for system runtime status and events. Through color coding and patterns, the LEDs indicate states such as power-on, controller activation, and fault conditions. Compared to the OLED display, the NeoPixels offer instant and more readily apparent status indication of the MCU's runtime state, signaling firmware operational transitions or error conditions. Finally, a set of push buttons enables direct user input for functions such as soft reset, MCU boot mode selection, or triggering custom firmware operations. Four of these buttons are connected to the system I/O expander, which provides both digital state readings and interrupts, allowing custom user operations triggered via ISRs.

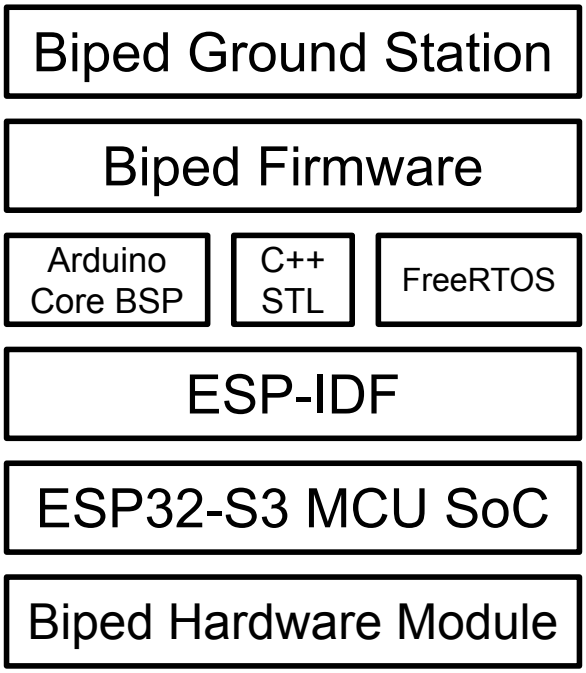

Figure 8. Biped firmware and software stack.

IV. Firmware Architecture

The Biped firmware is built on a modular software stack layered over the Biped hardware module and the ESP32-S3 SoC, as illustrated in Figure 8. Above the hardware, the Espressif IoT development framework (ESP-IDF)[78] provides peripheral drivers, system libraries, and toolchain utilities for firmware development. To enhance compatibility and leverage a broad ecosystem of community-supported hardware drivers, the firmware employs the Arduino core board support package (BSP)[79] for ESP32, which wraps the ESP-IDF APIs without severely compromising performance. Development is performed entirely in modern C++, utilizing its standard templated library (STL), and compiled using the Xtensa GCC toolchain. The build process is managed using CMake,[80,81] allowing the firmware to be organized as a standard CMake-based C++ project while linking against both Arduino and ESP-IDF libraries. The Biped firmware's core functionalities, sensing, actuation, control, planning, communications, task scheduling, interrupts, and user interface, are implemented as independent modules executed by FreeRTOS, ensuring deterministic real-time performance and platform-agnostic extensibility.

A. Sensing

The sensor module in the Biped firmware forms the foundation of its sensing and perception pipeline. The module provides both translational and attitude measurements, enabling downstream control and planning tasks for various hardware configurations, including the TWSBR and the Quanser helicopter. Each sensing component is abstracted through a dedicated driver class implemented under the platform modules and initialized by the sensor module at startup. This layered abstraction promotes modularity, allowing additional sensors to be integrated by implementing new platform modules without modifying the core logic within the sensor module. The current firmware implementation supports sensors sufficient for both the TWSBR and Quanser helicopter configurations, i.e., the onboard IMU, rotary encoders connected through the actuation or expansion interfaces, and the camera module.

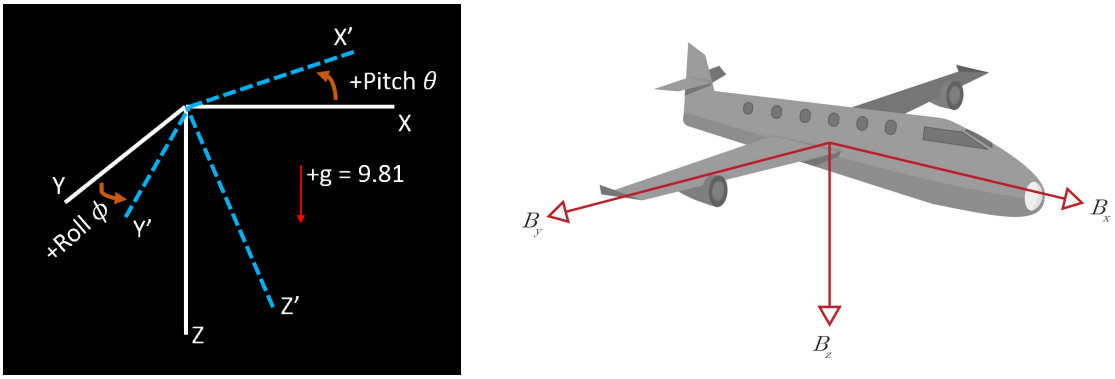

Figure 9. Left: rotation of body frame with respect to inertial frame.[85] Right: standard body reference frame.[83]

1. *Inertial Measurement Unit*

The mounting orientation of the physical IMU sensor hardware may vary between hardware designs. Therefore, the sensor module performs IMU axis remapping to conform all IMU measurements to the standard body reference frame,[82,83] as shown in Figure 9, where the X-axis points forward along the nose, the Y-axis points out of the starboard side, and the Z-axis points down toward the ground. Translational remapping is performed experimentally by aligning each side of the Biped hardware module with the gravity vector to determine the corresponding axis directions. Angular remapping then follows by applying the rotational right-hand rule[84] around each translational axis to establish the correct rotational directions. This remapping ensures consistent sign conventions across all firmware modules.

Earlier versions of the IMU platform module supported both the MPU6050[71] and the Bosch BMX160[72] sensors. The IMUs communicate with the MCU via the I^2C bus, providing 3-axis measurements of acceleration, angular velocity, and magnetic field strength. The sensor module polls these measurements through the read routine of the IMU platform module, retrieves acceleration, gyroscope, and compass data, and initializes their entries within the sensor data structure. However, due to instability in the BMX160's compass readings and the need for manual sensor fusion on the MCU to obtain attitude estimates from both devices, the latest firmware consolidates IMU support around the more advanced Bosch BNO055.[51] Nevertheless, to illustrate the evolution from low-level, MCU-side fusion to high-level, sensor-side fusion, the MCU-side manual fusion process is demonstrated below as an example in the context of TWSBR.

Once raw IMU measurements are remapped to the standard body frame, the sensor module computes the roll and pitch angles, as illustrated in Figure 9, by deriving the instantaneous rotation of the body frame with respect to the inertial frame,[82,85] whose attitude aligns with the gravity vector:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 & -\sin\theta \\ \sin\theta\sin\phi & \cos\phi & \cos\theta\sin\phi \\ \sin\theta\cos\phi & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{1}$$

where a_x, a_y, a_z are the translational accelerations along the inertial frame axes measured by the IMUs, ϕ is the roll angle, θ is the pitch angle, and g is the gravitational constant. Note that the yaw angle ψ is unobservable from the accelerometers, as the yaw or XY plane within the inertial frame is orthogonal to the gravity vector, which lies along the X-axis in the inertial frame. Therefore, ψ depends on measurement through the IMU compass.

Equation 1 yields the following system of equations:

$$a_x = -g\sin\theta \tag{2}$$

$$a_y = g\cos\theta\sin\phi \tag{3}$$

$$a_z = g\cos\theta\cos\phi \tag{4}$$

which further yields:

$$\phi = \tan^{-1}\left(\frac{a_y}{a_z}\right) \tag{5}$$

$$\theta = \sin^{-1}\left(-\frac{a_x}{g}\right) \tag{6}$$

To obtain a stable attitude estimate, the sensor module fuses the roll and pitch angles derived from the accelerometer, as shown in Equation 5 and 6, with the corresponding roll and pitch rates, $\dot{\phi}$ and $\dot{\theta}$, measured by the gyroscope using Kalman filters.[86] The Kalman filtering, computed on the MCU, combines the derived attitude angles with their measured derivatives, i.e., the attitude rates, to produce a temporally consistent and robust attitude estimate suitable for downstream control tasks. This manual fusion method is effective for TWSBR basic balancing control but assumes negligible translational acceleration within the inertial XY plane. When the chassis accelerates due to motion, the resulting acceleration components skew the accelerometer readings, thereby biasing the estimated roll and pitch angles. This assumption highlights a key limitation of raw IMU-based estimation without complete multi-axis fusion.

Therefore, the latest Biped hardware module design incorporates the Bosch BNO055, a smart IMU featuring an onboard ARM Cortex-M0+ coprocessor running Bosch's proprietary fusion firmware. The BNO055 performs continuous 9-DoF fusion, self-calibration, and active compensation and filtering, producing attitude estimations directly in Euler angles and quaternions accessible through the I^2C interface. This integration eliminates the need for the manual attitude computation and the associated real-time Kalman filtering on the MCU, freeing CPU cycles for other tasks while delivering more accurate attitude measurements with minimal drift. These improvements directly enhance the stability and robustness of downstream closed-loop control requiring IMU-based attitude measurements.

2. *Rotary Encoder*

Rotary encoders provide direct and precise measurements of angular displacement along wheel, motor, or other rotational axes. A magnetic rotary encoder consists of a magnetic disk coupled to the rotational axis being measured and two Hall-effect sensors positioned around the axis. The two sensor channels output square-wave signals 90 degrees out of phase during rotation. Optical rotary encoders operate under the same principle, except that the phase-shifted signals are detected optically rather than magnetically.

By observing which channel leads the other, as illustrated in Figure 10, the encoder platform module determines the direction of rotation and observes the rising and falling edges, i.e., steps, corresponding to the angular displacement. In addition to step measurement, the encoder platform module estimates angular velocity by differentiating the encoder position within the sense function of the sensor module, which is invoked periodically by the real-time task with a fixed execution period. Let Δs denote the change in encoder steps since the previous period, and Δt denote the period of the real-time task. The instantaneous angular velocity is computed as:

$$\omega = c \cdot \frac{\Delta s}{\Delta t} \tag{7}$$

where c is a scaling constant determined through the type and parameters of a given chassis. For example, in the context of the Quanser helicopter, c converts encoder steps to angular displacements in degrees, yielding an angular velocity used for differential control to dampen overshoot and improve stability during attitude maneuvers. For TWSBR, on the other hand, c converts encoder steps to linear displacement of the wheels in meters along the body frame X-axis, providing a translational velocity used for position and velocity control.

To mitigate quantization and noise arising from small step changes over short sampling intervals, the encoder platform module applies a low-pass filter in the form of a difference equation through discrete time sampling:

$$\omega_f(t) = \beta \cdot \omega_f(t-1) + (1-\beta) \cdot \omega \tag{8}$$

where $\omega_f(\cdot)$ is the filtered encoder velocity at a given sampling time, ω is the most recent instantaneous encoder velocity, and $\beta \in [0,1]$ is the filter smoothing factor. The filtered encoder velocity provides a stable estimate of translational or angular velocity for both ground and aerial configurations.

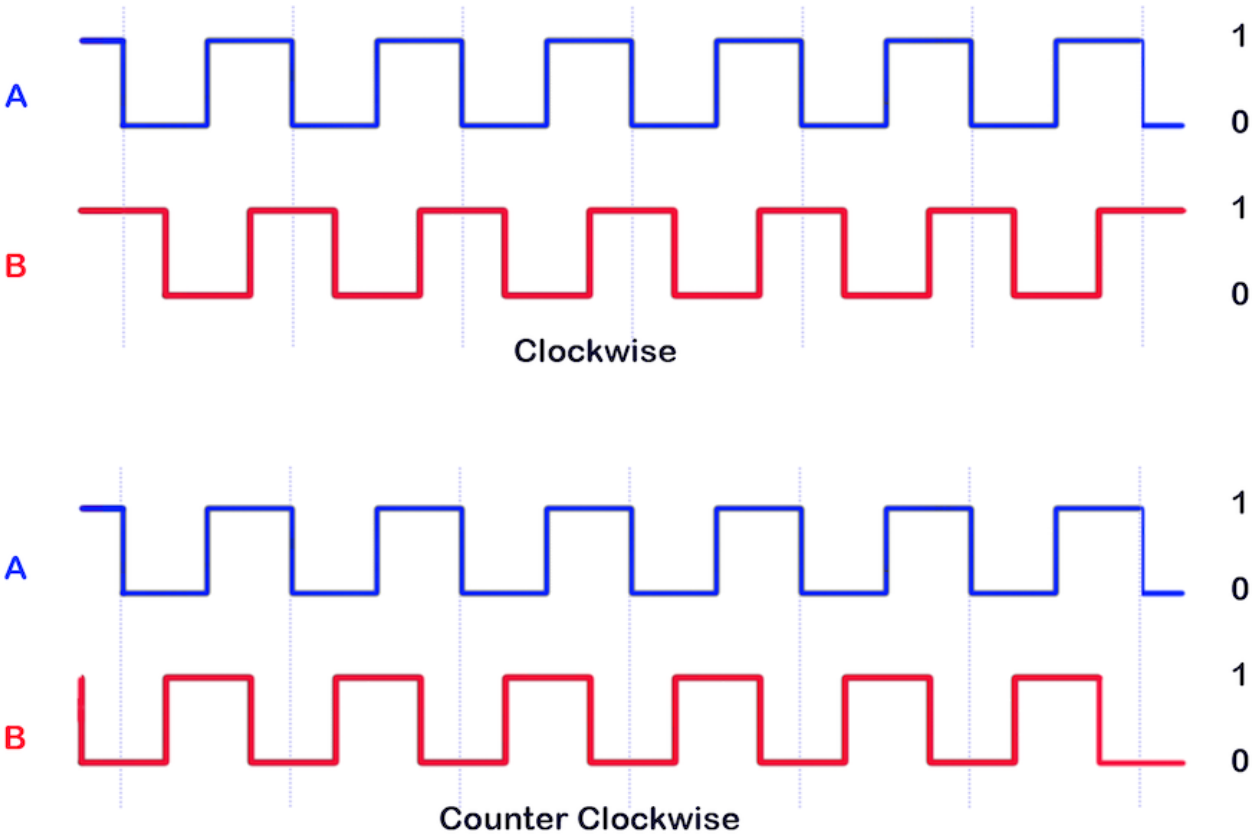

Figure 10. Rotary encoder sensor signals.[87]

3. *Camera*

The Biped firmware includes a camera platform module for configuration, initialization, and image frame acquisition and transmission. Camera configuration and control are managed via the SCCB,[70] a lightweight two-wire serial interface similar to the I^2C protocol. Since SCCB and I^2C are electrically compatible, they share the same signal lines on the Biped hardware module. Therefore, the firmware ensures that the camera hardware is configured and initialized via the SCCB first before initializing the I^2C bus. Within the camera platform module, the image resolution can be configured through the SCCB. Depending on the camera hardware, supported resolution ranges from QVGA to UXGA, allowing users to balance image quality and resource utilization. Additionally, the image signal processor (ISP) onboard the camera hardware performs real-time JPEG compression, enabling each frame streamed from the camera hardware to be pre-compressed without consuming CPU cycles, thereby reducing bandwidth and improving latency. Furthermore, the ESP32 camera driver allows users to configure the frame buffer location, either in internal data random-access memory (DRAM) for maximum access speed or in external PSRAM for larger buffers. Both the JPEG quality and the number of frame buffers are configurable, allowing users to control frame rate, image quality, and memory utilization.

Once configured and initialized, the camera platform module handles the acquisition and transmission of image frames over the network. Each image frame is accessed through the ESP32 camera driver by a pointer to the most recent frame buffer in memory. The retrieval of image frames is effectively instantaneous, as the image data is written directly into the MCU memory by the camera hardware through the DMA controller. Depending on the camera hardware, the retrieved frame may already be JPEG-compressed by the onboard ISP. Otherwise, the firmware invokes a function on the MCU side to perform JPEG compression before transmission. Prior to sending each frame, the firmware transmits a frame boundary to delineate the separation between consecutive frames within the network packet stream. This

boundary is represented by a long and unique sequence of characters, such as:

12345678900000000000000987654321

which is sufficiently distinctive and thus highly unlikely to occur naturally within any image payload. The use of frame boundaries is necessary due to a behavior within the ESP32 network stack, where oversize network packets exceeding the maximum transmission unit (MTU) are fragmented but not reassembled on the receiving side. Since each image frame may be split across multiple packets, the receiving endpoint relies on these explicit frame boundaries to reconstruct the complete image frames reliably. While exceedingly rare, collisions between frame boundary sequences and actual image data can cause minor and transient frame corruptions, an acceptable limitation given the inherently lossy and non-reliable nature of image frame transmissions.

B. Actuation

The actuation module within the Biped firmware provides configuration and control for the onboard two-channel actuation interface that drives the connected chassis actuators, as well as any external actuation interfaces or actuators connected via the expansion interface. Each actuation channel corresponds to a dedicated PWM control pin on the MCU, initialized during startup. The firmware configures these PWM pins as outputs to the H-bridge drivers and designates the actuation enable and direction pins through the system I/O expander. In addition to software control, the actuation enable pin can be overridden by a physical hardware switch on the Biped hardware module. This dual mechanism ensures both operational safety and convenience: the operator can physically disable all actuators while leaving the rest of the system powered for debugging or calibration, while the firmware retains shared control of the actuators for software-level control and testing.

During operation, an upstream actuation command data structure is passed to the actuation module, containing the actuation enable flag, channel directions, and PWM control values. Based on the given actuation commands, the actuation module sets the actuation enable and direction pins through the system I/O expander and generates the PWM control signals corresponding to the actuation magnitudes. PWM generation is handled by the LED controller (LEDC) peripheral within the ESP32-S3 MCU. Initially designed for LED brightness control, the LEDC provides precise PWM generation at configurable frequencies and duty cycles without CPU overhead. By default, the firmware configures the LEDC to produce PWM signals with an 8-bit resolution, although higher resolutions can be selected depending on the timer configuration. An 8-bit resolution corresponds to PWM control values ranging from 0 to 255. A value of 255 represents a full-duty (100%) output, while 0 corresponds to no actuation.

C. Controller

The controller module within the Biped firmware serves as the intermediary between the sensing and actuation modules, generating real-time actuation commands for the actuation module based on the plant states observed by the sensor module. The module implements a custom and configurable controller cascade composed of modular controller building blocks, enabling flexible design and construction of control schemes tailored to different chassis configurations. The current firmware offers two representative controller building blocks: a proportional-integral-derivative (PID) controller and an open-loop controller, which can be combined to implement cascaded control architectures for both the ground-based TWSBR chassis and the air-based Quanser helicopter. The adaptable design of the controller module further enables implementation and integration of new and more advanced controller types, such as state-feedback control,[88] model-predictive control (MPC),[89] or L1 adaptive control,[90] by simply conforming to the same base interface.

1. PID Controller

The PID controller within the controller module serves as a core building block implementing the PID control law defined in the time domain as:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt} \tag{9}$$

where t is time, $u(t)$ is the controller output, $e(t)$ is the error between the controller reference and the observed plant state, and K_p, K_i, K_d are the proportional, integral, and derivative gains, respectively. Intuitively, the proportional term acts as a reflex, generating an immediate response proportional to the current error. The integral term functions as a piggy bank, compensating for steady-state offsets by accumulating the error over time. The derivative term behaves as a damper, counteracting overshoots and oscillations by limiting response based on the rate of change of the error.

The firmware implements the PID controller as an object-oriented class that exposes getter and setter functions for all key parameters and internal states. These functions enable the configuration of the control reference, observed state, externally measured error derivative if available, control period, and gain parameters, as well as saturation limits for both the controller input and output. The implementation automatically resets the integrated error whenever the reference, period, or gains are updated, preventing residual accumulation from previous configurations. The integral term is computed as a discrete-time summation of the error using the configured control period, with an anti-windup saturation that constrains the integrated error to a predefined bound to prevent integral runaway. The differential term may be estimated internally or provided externally as an observed derivative, such as an angular or linear velocity obtained from the sensor module. During each control cycle, the controller evaluates the proportional, integral, and derivative terms, sums them, and applies output saturation to the sum as the final output from the PID controller.

2. *Open-Loop Controller*

The open-loop controller implements a simple, feedforward control law, which is useful when reliable feedback is unavailable or unnecessary. The controller shares the same interface as the PID controller but omits feedback computation. Instead, the output is computed directly as a gain-scaled reference:

$$u(t) = K_o \cdot r(t) \tag{10}$$

where $r(t)$ is the controller reference and K_o is the open-loop gain. As with the PID controller, saturation is applied to both the input and output to ensure that the produced controller output remains within a reasonable limit. The open-loop controller is particularly useful for non-critical control axes where feedback sensors are unreliable or absent, such as the yaw control in earlier Biped designs prior to the adoption of the BNO055 smart IMU. The implementation of the open-loop controller also highlights the firmware's extensibility, demonstrating how additional controller types can be implemented and integrated with minimal modification to the firmware architecture.

3. *Controller Cascades*

The main controller class within the controller module composes multiple controller instances, either PID or open-loop, into a series of custom, configurable controller cascades. Each cascade instance maintains dedicated parameter and reference structures that specify controller gains, saturation limits, activation thresholds, and control references for each control axis. Once configured, the controller cascade receives the plant states collected by the sensor module, updates each controller building block with its corresponding state and reference, and computes the cascaded control outputs that form the actuation commands. These commands are then passed to the actuation module, which drives the connected actuators through the actuation interface. Safety mechanisms are integrated to prevent unstable behavior during control. For instance, the controller module monitors the pitch angle of the TWSBR and the Quanser helicopter, automatically disabling actuation when the observed pitch exceeds the defined activation threshold to prevent unintended runaway motion. Upon deactivation, the controller module resets all integral terms and sets all actuation outputs to zero, thereby ensuring a safe control idle state and stable reactivation.

Within the current implementation, the controller cascade for the TWSBR chassis, as seen in Figure 11, employs three independent control paths for pitch, translational position, and yaw. The pitch controller maintains vertical stability using the pitch angle and pitch rate measured by the IMU. The position controller regulates forward motion using position and velocity feedback derived from the rotary encoders. The yaw controller operates in open loop, producing a fixed bias between the left and right actuation commands for turning. With the integration of the BNO055 IMU, the open-loop yaw controller can be easily replaced by a PID controller within the cascade, enabling closed-loop yaw

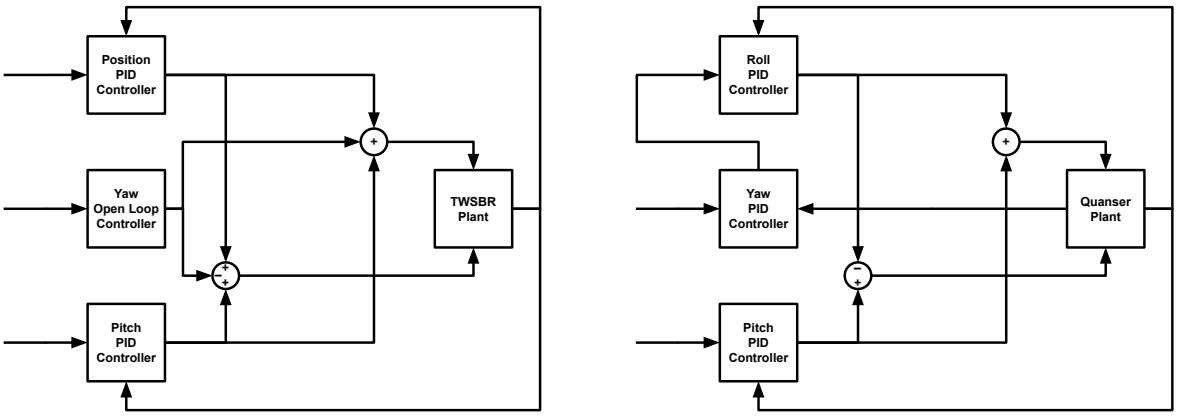

Figure 11. Left: controller cascade for TWSBR with open loop yaw. Right: controller cascade for Quanser helicopter.

control using the yaw angle measured by the IMU. Within the cascade, the left and right controller outputs are computed to form the actuation commands:

$$u_{left} = u_{pitch} + u_{position} + u_{yaw} \tag{11}$$

$$u_{right} = u_{pitch} + u_{position} - u_{yaw} \tag{12}$$

This controller cascade achieves closed-loop balance and translational control while enabling basic heading control through the open-loop yaw controller.

In contrast, the Quanser helicopter controller cascade implements a 3-axis closed-loop PID control managing all three attitude angles. Each attitude controller receives its corresponding angle and angular velocity state feedback derived from the rotary encoders. In this configuration, the yaw controller output is cascaded as the reference input to the roll controller, reflecting the coupled dynamics between the two axes: roll motion produces the lateral thrust necessary for yaw rotation. Meanwhile, the pitch controller remains an independent control path. Therefore, the controller outputs are computed as:

$$u_{left} = u_{pitch} + u_{roll} \tag{13}$$

$$u_{right} = u_{pitch} - u_{roll} \tag{14}$$

As illustrated in Figure 11, this controller cascade enables stable multi-axis attitude control of the Quanser helicopter in a fully closed-loop configuration.

The above controller cascades demonstrate the modular and adaptable aspects of the Biped firmware architecture. New control strategies can be implemented by deriving from the same base interface, while existing cascades can be reconfigured entirely through parameter or configuration updates without altering the underlying firmware structure. This unified and layered design enables rapid adaptation to new hardware platforms and control schemes, allowing the same controller framework to be reused across diverse chassis configurations with minimal development effort.

D. Planner

The planner module within the Biped firmware defines the high-level decision logic responsible for generating controller references for the controller module, based on either time-driven or condition-driven transitions. The module provides an abstract interface that supports arbitrary planner implementations, enabling different planning strategies to be created and interchanged without modifying the firmware structure. Each planner inherits from a shared abstract base class that defines two core functions. The start function resets the internal planner state and initializes the planner sequence, which is typically invoked once upon startup or on demand in response to user input, such as a push-button interrupt. The run function is executed periodically by the real-time task, evaluating the current plan status, verifying time-based

or state-based transition conditions, and issuing updated controller references to the controller module based on the current plan. This design encapsulates initialization and periodic execution behaviors within a consistent interface, enabling new planner designs to be incorporated by simply overloading the interface, thus maintaining compatibility with the existing firmware pipeline.

1. Waypoint Planner

The waypoint planner implements a time-based sequencing mechanism that advances through a predefined list of controller references, or waypoints, according to their specified time durations. Each waypoint is represented as a node in a linked list structure containing its controller reference, duration, and a pointer to the next waypoint in the list. Upon start, the waypoint planner resets to the first waypoint in the predefined plan. During execution, the run function periodically checks the elapsed time of the active waypoint using the system timer. Once the duration of the current waypoint expires, the planner advances to the next node and applies its controller reference to the controller module. This process continues until the final waypoint is reached, at which point the plan is marked complete. The waypoint planner, therefore, operates as a finite-state machine driven solely by elapsed time.

2. Maneuver Planner

The maneuver planner generalizes the waypoint-based approach into a condition-driven planning framework inspired by prior work on flight maneuver automation,[91,92] as illustrated in Algorithm 1. Each maneuver specifies both a behavior, such as driving forward, reversing, or turning, and a transition condition that determines when to advance to the next maneuver. Transitions may depend on elapsed time, as in the waypoint planner, or on plant states observed by the sensor module, enabling the planner to react dynamically based on the current states of the plant. Upon start, the maneuver planner resets to the beginning of the maneuver sequence and generates the initial controller reference based on the maneuver type. During each periodic call to the run function, the planner monitors the transition condition of the active maneuver, comparing observed sensor data or elapsed time against its specified threshold. When the condition is satisfied, the planner advances to the next maneuver and regenerates the controller reference accordingly. For example, the planner generates a controller reference to hold the current position during a park maneuver or commands forward motion with a yaw offset during a turning maneuver.

E. Communications

The communication module within the Biped firmware forms the networking backbone that enables wireless data exchange between the Biped hardware module and external network peers, such as the Biped Ground Station interface running on a remote host. The module integrates several software components: a Wi-Fi platform module that establishes and manages wireless connectivity, a user datagram protocol (UDP) interface that provides connectionless and efficient data transmission, and a serialization framework that encodes structured data into transferable byte streams. Together, these components form a lightweight, asynchronous communication pipeline supporting application-level features such as remote telemetry, controller parameter tuning, and image frame transmission.

The Wi-Fi platform module manages wireless network configuration by initializing the wireless modem onboard the ESP32-S3 MCU, connecting to the designated wireless access point (AP), and reporting the assigned IP address. Built on top of this module, the UDP interface implements the connectionless mechanism layer used for all data transmissions. Compared to the transmission control protocol (TCP), UDP is sufficient and more suitable for the Biped firmware, where data exchanges are non-critical and do not impact core firmware operations. The TCP connection handshakes, acknowledgments, and retransmission procedures introduce unnecessary latency and overhead, making UDP preferable for tasks such as remote telemetry and control. The UDP interface provides functions for transmitting and receiving either raw byte buffers or strings to and from a specified IP address and port. To prevent packets from unwanted sources, the UDP receive routine filters incoming datagrams by checking the source IP address and port against the expected peer, while outgoing packets embed the destination directly in each datagram.

Algorithm 1: Maneuver planner

```
function start()
    if plan.completed then
        plan.completed ← false
        plan.started ← false
        maneuver.current ← plan.head
        maneuver.started ← false
function run()
    if plan.completed then
        return
    if plan.started ∧ maneuver.current = ∅ then
        plan.completed ← true
        plan.started ← false
        return
    if ¬ plan.started then
        if maneuver.current = ∅ then
            return
        plan.started ← true
    if ¬ maneuver.started then
        controller.reference ← createReference(maneuver.current.type)
        maneuver.started ← true
    else if checkTransition(maneuver.current.transition) then
        maneuver.current ← maneuver.current.next
        maneuver.started ← false
task bestEffort()
    while true do
        run()
plan.completed ← true
createTask(bestEffort)
start()
```

All data transmitted across the network is defined through shared structures in the global type header. This header includes common structures such as IMU data, encoder data, controller parameters, controller references, and actuation commands. Each structure implements a member function that conforms to a serialization interface defined by a lightweight header-only serialization library,[93] enabling consistent serialization and deserialization of complex data without external dependencies. A top-level structure called `BipedMessage` aggregates all relevant data structures. This unified message format serves as the standardized communication interface between the Biped hardware module and external network peers, ensuring that both ends of the network share an identical data schema.

F. FreeRTOS Tasks

The Biped firmware runs on the FreeRTOS,[39] which provides deterministic task scheduling and concurrent execution across multiple functional modules within the firmware. By default, FreeRTOS employs a fixed-priority preemptive scheduling policy,[94] where higher-priority tasks preempt lower-priority ones, and tasks of equal priority execute in a round robin (RR), time-sliced fashion. To mitigate real-time priority inversion, FreeRTOS supports mechanisms such as the priority inheritance protocol,[95] which temporarily elevates the priority of a blocked task to guarantee bounded blocking during shared resource access. Shown in Figure 12, FreeRTOS enables the Biped firmware to ensure real-time performance for sensing, control, and actuation, while executing non-critical operations, such as communication, planning, and user interface updates, on a low-priority, best-effort basis.

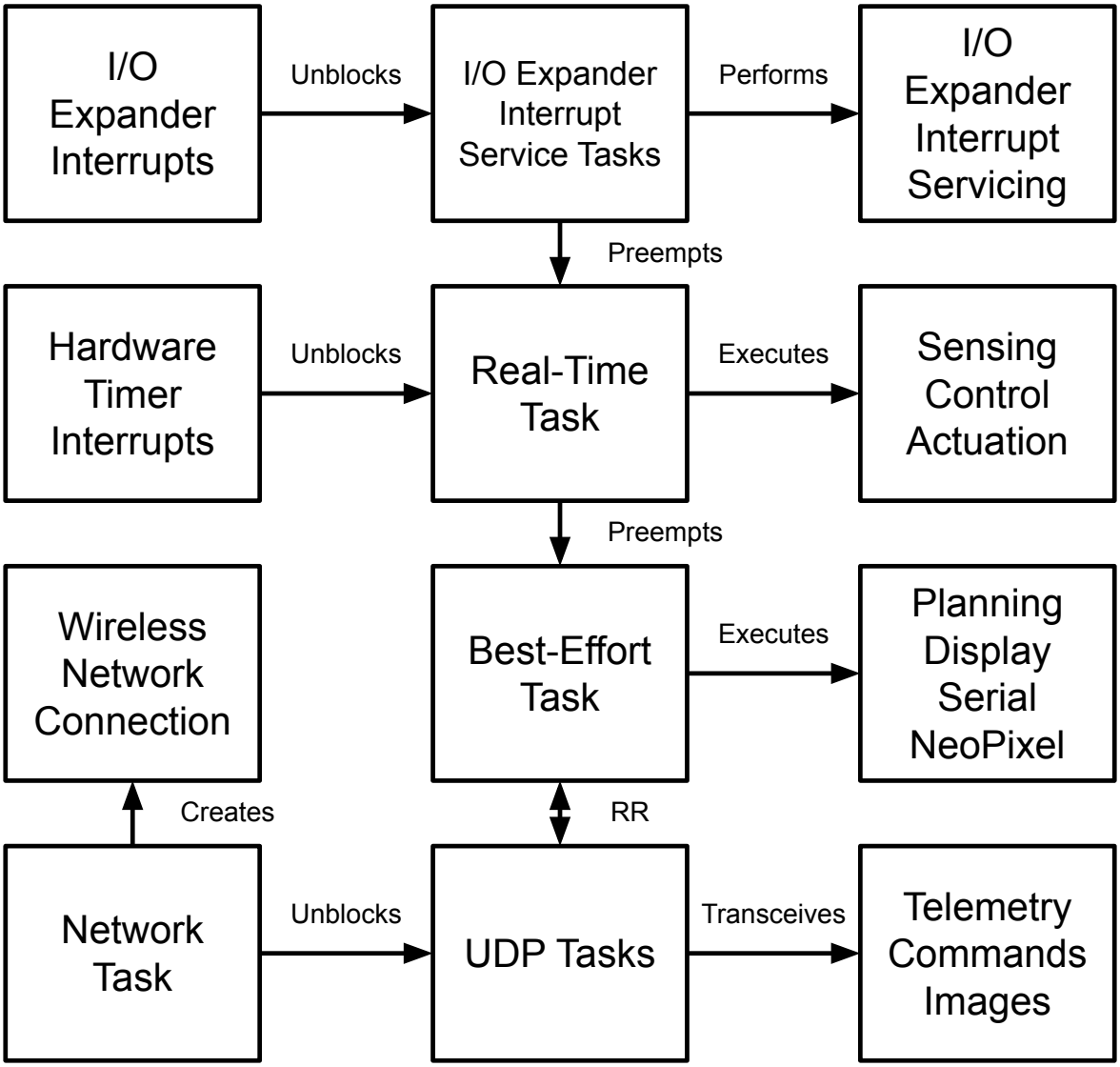

Figure 12. Overview of FreeRTOS tasks. RR denotes the round-robin execution of equal priority tasks.

1. Real-Time Task

The real-time task forms the core execution loop of the Biped firmware. The task operates at a fixed period synchronized by a hardware timer to ensure consistent execution timing required by the controller and sensor modules. The timer platform module provides the interface for configuring one of the ESP32-S3's hardware timers, supporting features such as auto-reload, prescaling, and interrupt generation. The module also implements an interrupt service framework that allows a timer interrupt handler to be attached to the configured timer. Upon timer expiration, the timer interrupt handler releases a binary semaphore, which unblocks the real-time task and triggers its next iteration. This synchronization mechanism guarantees that the sensing, control, and actuation routines execute exactly once per timer period, maintaining consistent sampling and update rates. Within each iteration, the task performs sensor data acquisition, controller computation, and actuation output in sequence. If the cumulative execution time of the sensing-control-actuation pipeline exceeds the timer period, users should either optimize their implementation or increase the real-time task period to avoid deadline violations.

2. *I/O Expander Interrupt Service Tasks*

The firmware employs two high-priority interrupt service tasks: one dedicated to the system I/O expander and the other to the expansion interface I/O expander. Both tasks execute at the highest priority within the firmware, preempting even the real-time task to emulate a software-level interrupt response. Each task remains blocked on a semaphore until the MCU receives a hardware interrupt from its corresponding I/O expander. Upon detecting an interrupt from one of its pins, the I/O expander asserts its interrupt line, which is directly connected to an MCU GPIO pin. The I/O expander interrupt handler on the MCU then triggers and releases the semaphore, waking the associated interrupt service task. This design delegates the I^2C transactions required to read the I/O expander status registers to the high-priority I/O expander tasks rather than executing them directly within the interrupt context, thereby avoiding potential deadlocks or watchdog timeouts. Once awakened, the I/O expander interrupt service tasks use the interface provided by the I/O expander platform module to service the pending interrupt before returning to their blocked state.

3. *Best-Effort Task*

The best-effort task handles non-critical operations that do not require deterministic timing. The task runs with the lowest priority and executes user interface routines such as NeoPixel updates, OLED display rendering, and serial logging, along with the periodic execution of the planner module. Since they operate at the decision-making level to generate controller references, planners can tolerate occasional missed cycles without compromising control stability or safety. This scheduling approach prioritized computational resources for real-time and interrupt-driven workloads, while maintaining a responsive user interface and ensuring the operation of non-critical modules.

4. *Network and UDP Tasks*

Network operations are executed as low-priority background tasks together with the best-effort task in an RR fashion. At startup, the network task initializes the wireless connection through the Wi-Fi platform module and configures all UDP interfaces. Each UDP-related task remains blocked on a binary semaphore until the network task successfully connects to the specified wireless AP and releases the semaphore, ensuring that periodic UDP operations begin only after the network is available. Once initialization is complete, the network task terminates, leaving the UDP tasks to operate autonomously in the background. Two dedicated UDP tasks handle the transmission of serialized `BipedMessage` packets. The read task listens for incoming datagrams, deserializes valid messages, and updates the system state, such as applying controller parameters or control references received from the Biped Ground Station. The write task periodically collects sensor data, controller states, and actuation commands into the `BipedMessage` format, and then serializes and transmits the packet over UDP. This symmetric read-write mechanism enables continuous, bidirectional communication for remote monitoring and control. A third task, the UDP camera write task, streams compressed image frames retrieved from the camera platform module.

G. Interrupts

The interrupt framework within the Biped firmware coordinates all event-driven operations, providing a high-priority, low-latency response to asynchronous hardware signals. Interrupts in the Biped firmware fall into two primary categories: 1) MCU-based interrupts, which occur directly on the native MCU GPIO pins and are serviced within standard ISRs, and 2) I/O-expander-based interrupts, which originate on the I/O expanders and are propagated to the MCU through their dedicated interrupt lines. This hierarchical design enables the firmware to handle both high-frequency peripherals, such as rotary encoders, and slower, multiplexed peripherals connected via the I/O expanders, without consuming the MCU's limited GPIO pins. All ISRs are compiled into the instruction random-access memory (IRAM) by assigning them the IRAM attribute. This attribute instructs the firmware to place interrupt code in a low-latency on-chip memory region specifically optimized for instruction execution rather than in flash, DRAM, or PSRAM, thereby minimizing instruction fetch latency within interrupt contexts.

Algorithm 2: Rotary encoder interrupt service framework

```
ISR handleEncoderA()
    a ← readMCUPin(mcu.pin.encoder.a)
    b ← readMCUPin(mcu.pin.encoder.b)
    if a ≠ b then
        encoder.steps ← encoder.steps + 1
    else
        encoder.steps ← encoder.steps − 1
ISR handleEncoderB()
    a ← readMCUPin(mcu.pin.encoder.a)
    b ← readMCUPin(mcu.pin.encoder.b)
    if a = b then
        encoder.steps ← encoder.steps + 1
    else
        encoder.steps ← encoder.steps − 1
registerMCUISR(mcu.pin.encoder.a, handleEncoderA, {rising, falling})
registerMCUISR(mcu.pin.encoder.b, handleEncoderB, {rising, falling})
```


Within the encoder platform module, a pair of ISRs, shown in Algorithm 2, is registered for each encoder channel. Each ISR is triggered on both the rising and falling edges of the encoder signal, allowing the firmware to capture every transition and thus achieve full quadrature resolution. Upon each edge event, the corresponding ISR is invoked to update the encoder's step count according to the detected rotation direction. This interrupt-driven design eliminates the need for polling and maintains precise step tracking that remains tightly synchronized with the physical encoder motion. Each encoder channel is connected directly to a dedicated MCU GPIO pin configured for dual-edge interrupts. Connecting the encoder channels directly to native MCU GPIO pins, rather than through I/O expanders, reduces delay by avoiding the I^2C transactions otherwise required to poll expander registers and resolve interrupt sources. This design choice enables fast and reliable encoder interrupt servicing even at high rotational speeds.

Beyond MCU-level interrupts, the firmware incorporates a dedicated I/O expander platform module that implements a hierarchical interrupt service framework for handling I/O-expander-level interrupts. The module provides unified interrupt configuration, registration, and servicing mechanisms for all 16 pins across the two I/O expander ports, each containing 8 pins. Each pin can independently attach or detach an ISR and specify its interrupt triggering mode. The interrupt modes are either rising edge, falling edge, level-high, or level-low, which correspond to edge-triggered or level-triggered interrupts. These modes are configured by writing to the MCP23017[63] I/O expander's interrupt control and default value registers, which determine whether a pin's current logic state is compared against its previous state, i.e., edge-triggered interrupts, or a fixed default value, i.e., level-triggered interrupts. Each registered ISR, along with its arguments and interrupt mode, is stored in an internal ISR vector indexed by I/O expander pin number, allowing for efficient lookup during interrupt servicing.

When an I/O expander interrupt occurs, the expander first asserts its dedicated hardware interrupt line connected directly to an MCU GPIO pin. The corresponding MCU ISR executes with minimal overhead. The ISR temporarily deregisters itself from the MCU GPIO interrupt service to prevent reentrancy, then signals a binary semaphore to unblock the high-priority I/O expander interrupt service task. Once awakened, the I/O expander interrupt service task invokes the interrupt service framework provided by the I/O expander platform module, as outlined in Algorithm 3. Within this framework, the firmware first reads the interrupt flag and capture registers from the I/O expander over the I^2C bus. The flag register identifies which I/O expander pins generated interrupts, while the capture register records the logic level (high or low) of each pin at the precise moment the interrupt occurred. For the MCP23017 I/O expander, reading the capture register automatically clears the asserted interrupt.

Then, the interrupt service framework iterates over all I/O expander pins, checking each pin's corresponding flag bit to determine whether it was the source of the interrupt. For every triggered pin, the framework retrieves the corresponding entry from the internal ISR vector and evaluates its interrupt mode to determine whether its ISR should

Algorithm 3: I/O expander interrupt service framework

```
function registerExpanderISR(pin, isr, args, mode)
    expander.isr[pin].invoke ← isr
    expander.isr[pin].args ← args
    expander.isr[pin].mode ← mode
    if mode = rising ∨ mode = falling then
        writeExpanderRegister(expander.register.control, pin, false)
    if mode = low then
        writeExpanderRegister(expander.register.control, pin, true)
        writeExpanderRegister(expander.register.value, pin, true)
    if mode = high then
        writeExpanderRegister(expander.register.control, pin, true)
        writeExpanderRegister(expander.register.value, pin, false)
    writeExpanderRegister(expander.register.enable, pin, true)
function serviceExpanderInterrupt()
    flag ← readExpanderRegister(expander.register.flag)
    capture ← readExpanderRegister(expander.register.capture)
    foreach pin ∈ expander.pin do
        if flag[pin] then
            switch expander.isr[pin].mode do
                case rising do
                    if capture[pin] then
                        expander.isr[pin].invoke(expander.isr[pin].args)
                case falling do
                    if ¬ capture[pin] then
                        expander.isr[pin].invoke(expander.isr[pin].args)
                otherwise do
                    expander.isr[pin].invoke(expander.isr[pin].args)
ISR handleExpander()
    deregisterMCUISR(mcu.pin.expander)
    giveSemaphore(expander.semaphore)
task expanderInterruptService()
    while true do
        takeSemaphore(expander.semaphore)
        serviceExpanderInterrupt()
        registerMCUISR(mcu.pin.expander, handleExpander, {high})
createTask(expanderInterruptService)
registerMCUISR(mcu.pin.expander, handleExpander, {high})
```

be invoked. In edge-triggered modes, the capture register indicates whether the interrupt was triggered by a rising or falling transition. In level-triggered modes, the ISR is invoked directly, as the previously configured control and default value registers ensure the interrupt was raised by either a high or low signal level. This mechanism supports mixed-mode operation, allowing different pins on the same I/O expander to employ distinct interrupt modes and independent ISR implementations concurrently. After servicing all pending interrupts, the I/O expander interrupt service task reattaches the MCU ISR for the I/O expander's interrupt line, restoring its readiness for subsequent interrupts. Algorithm 3 summarizes the complete interrupt service framework implemented by the I/O expander platform module, including ISR registration, MCU interrupt handling, semaphore signaling, and task-level servicing logic.

H. User Interface

The user interface module within the Biped firmware provides embedded visual and diagnostic feedback to both the operator and developer through three platform modules: the OLED display, serial, and NeoPixel platform modules. The display platform module presents structured textual information directly on the onboard OLED display hardware, the serial platform module manages serial diagnostic logging and debugging through a host computer, and the NeoPixel platform module provides immediate visual cues indicating system state and fault conditions during runtime. Together, these modules enable convenient onboard monitoring, debugging, and real-time visual indications.

1. OLED Display

The display platform module provides a streamlined abstraction over the SH1106G[76] monochrome OLED display hardware, enabling formatted text output through a convenient interface. Unlike low-level display drivers that require manual pixel addressing and framing, this module enables developers to print directly to the display hardware, much like writing to a standard output stream. Each display operation is executed through a transient display object, which is instantiated with a target line index and an optional raw printing mode. Users can stream variables, strings, or formatted data into the object using the stream insertion operator ($<<$), along with supported stream manipulators such as newline, clear line, and carriage return. When the display object goes out of scope, its destructor automatically flushes the internal string stream buffer to the display hardware while ensuring atomic and thread-safe updates via a mutex-protected lock. The module's initialization routine configures the display hardware, establishes text and cursor properties, and applies additional parameters such as rotation and brightness. During runtime, the module also adapts the display orientation to chassis orientation by reading the Z-axis acceleration from the sensor module.

2. Serial

The serial platform module provides a structured logging and diagnostic framework based on a configurable log-level hierarchy. The module supports 6 severity levels: trace, debug, info, warn, error, and fatal, allowing developers to adjust logging verbosity at runtime. Similar to the display platform module, each log entry is created by instantiating a transient serial object with a specified log level and streaming data into the object using the stream insertion operator ($<<$). When it goes out of scope, the object's destructor automatically flushes the accumulated string stream buffer to the serial hardware. The module maintains two global states: a configurable maximum log level that filters logged messages, and a worst log level that tracks the most severe event recorded since startup. These states are shared across the firmware, enabling other modules, such as NeoPixel, to indicate the firmware's runtime status and fault conditions.

3. NeoPixel

The NeoPixel platform module provides an immediate, low-latency visual indicator for the status of the hardware and firmware. The module interfaces with 4 NeoPixel LEDs, configured via shared frame buffers representing RGB color values for each LED. Two predefined frames are maintained: a normal operating frame rendered in blue and an error frame rendered in red. During each update cycle, the show function, invoked by the best-effort task, selects the active frame based on the current worst log level reported by the serial platform module, switching to red when an error or fatal condition is detected. This mechanism enables at-a-glance fault indication, particularly useful during autonomous operation, where maintaining a wired serial connection or observing the OLED display is impractical. The NeoPixel colors can also be updated during runtime by other firmware modules. For instance, the controller module sets the LEDs to green when active and reverts them to blue upon deactivation, providing immediate feedback on controller status without requiring serial or close observation of the OLED display.

V. Biped Ground Station

The Biped Ground Station is a desktop application developed in Qt and C++ that provides an integrated software environment for interacting with the Biped hardware module. Through its modular architecture, the Biped Ground Station supports wireless telemetry, real-time controller parameter tuning, and live camera streaming, while also enabling the logging of telemetry and image data for offline analysis. Built on an event-driven framework of background daemons and asynchronous signal-slot interactions, the Biped Ground Station maintains a responsive and extensible runtime that not only facilitates user interaction but also functions as a middleware for integrating external perception and planning algorithms on the remote host, allowing the Biped hardware module to offload computationally intensive workloads to higher-performance resources over the network.

A. User Interface

The Biped Ground Station provides an integrated desktop user interface module for real-time visualization, configuration, and control of the Biped hardware module. The graphical layout of the interface is organized into 6 primary tabs: controller, planner, data, camera, parameters, and settings, each dedicated to a specific aspect of system functionality and operation. Together, these components consolidate control, monitoring, and configuration into a unified environment. Through real-time telemetry and online parameter adjustment, the Biped Ground Station enables efficient controller tuning without the need to repeatedly modify, rebuild, or reflash firmware. This streamlined workflow forms a tight feedback loop between the hardware, firmware, and the user, substantially improving efficiency and usability.

1. Controller

The controller tab enables users to monitor controller performance and tune parameters in real-time while the system is operating. At the top are the control response plots for each controller instance. Figure 13 shows an example setup for the Quanser helicopter, corresponding to its roll, pitch, and yaw controllers. Each plot displays two curves: a green curve representing the current controller reference, and a red curve showing the plant state measured by the firmware sensor module. These plots update continuously as telemetry is received from the Biped hardware module over the network, providing an intuitive visualization of the controller's real-time tracking performance. If the curves exceed the current Y-axis range, the plots automatically zoom out to maintain visibility. A zoom-to-fit button beneath each plot allows manual rescaling when needed, ensuring the operator retains a consistent sense of scale even during rapid transients or abrupt controller responses.

Below the control response plots is the controller parameter widget, which lists the parameters for each controller, such as proportional, integral, and derivative gains, as well as optional saturation and integral-limit terms. The table is organized into three columns: the parameter name, the current value received from telemetry, and an editable input field for user adjustments. Users can modify the input fields and press the apply button to immediately transmit the updated parameters to the Biped hardware module wirelessly over the network. The revert button restores the editable fields to the most recent values received from the hardware module, while the save button stores the current controller parameters into the parameters tab for later reuse. Together, these interfaces enable real-time and online controller monitoring and tuning, eliminating the need to repeatedly modify, rebuild, and reflash firmware during experiments.

2. Planner

The planner tab provides both monitoring and manual control over the Biped firmware's onboard planner module, as well as the ability to monitor and override the controller reference values generated by the planner. As shown in Figure 13, the planner control widget allows users to monitor the current status and stage of the onboard planners, and to start or stop the planners using the provided controls. Additionally, users can create or modify the existing maneuver or waypoint sequence of the onboard planner, enabling fine-tuning of the sequence without requiring repeated compilation or flashing of the firmware. Within the planner parameter widget, users can monitor or directly specify desired controller references, such as target roll, pitch, or yaw values, and transmit them to the Biped hardware module in real time.

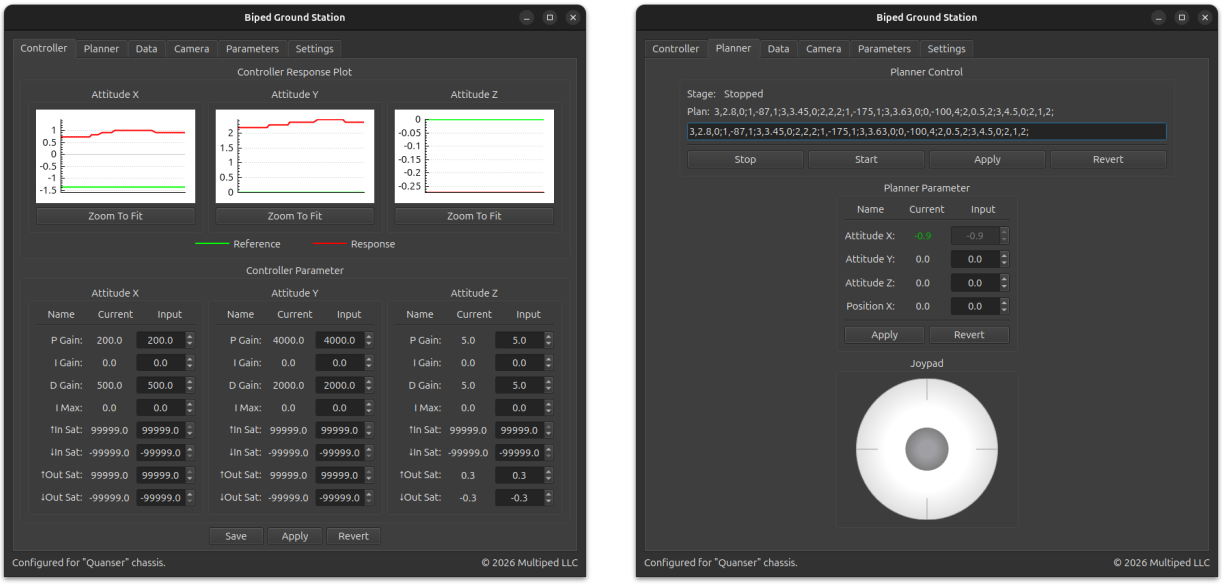

Figure 13. Left: Biped Ground Station controller tab. Right: Biped Ground Station planner tab.

This capability is particularly valuable for testing dynamic controller responses or manually commanding the plant without relying on onboard planners. The tab also includes a custom joypad widget designed for analog input along two axes. When deflected, the joypad updates the control references in real time, either through the software interface or a hardware joystick connected to the remote host, thereby enabling manual teleoperation of the chassis connected to the Biped hardware module. While the current planner tab primarily supports control of the onboard planner module, the same interface can be extended to incorporate a backend remote planner executing advanced trajectory generation or motion planning algorithms on the remote host computer, which may otherwise be too computationally intensive to run directly on the Biped hardware module.

3. Data

The data tab aggregates and visualizes raw telemetry data streamed from the Biped hardware module. The tab displays sensor readings, controller references, actuator commands, and other low-level system variables in real-time, updating continuously as new data arrives over the network. Illustrated by Figure 14, the tab provides users and developers with a direct and comprehensive view of the system performance in terms of raw values, allowing them to verify sensor performance, observe controller references, and confirm actuation commands. By exposing detailed information from the sensing, control, and actuation pipeline within the Biped firmware, the data tab serves as a crucial tool for debugging, validation, and performance analysis during development and experimentation.

4. Camera

The camera tab, shown in Figure 14, displays the live video feed transmitted by the camera module onboard the Biped hardware module. A dedicated rendering widget receives image frames over the network, decodes them, and renders the image data in real-time using OpenGL. This camera feed allows users to observe the scene as perceived by the Biped hardware module, providing useful visual feedback during operation and experiments. In addition, the camera tab supports overlays and visualizations for additional metadata, such as detection or recognition results from the ML-based perception models integrated with the Biped Ground Station running on the remote host. These visualizations enable users to view live inference results, providing immediate insight into the performance of vision models during runtime.

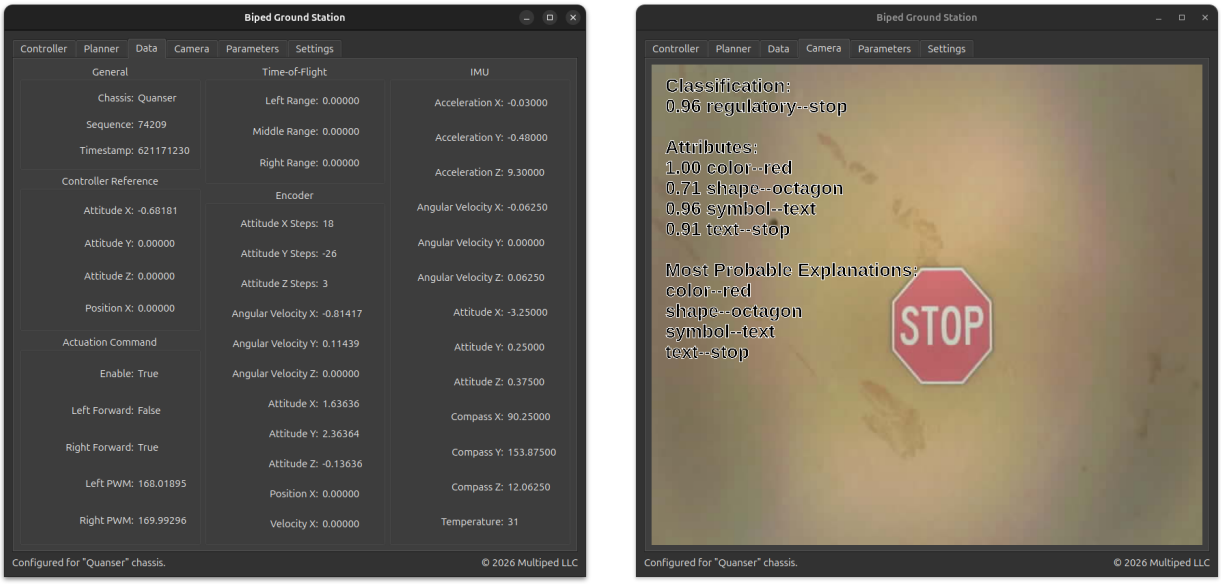

Figure 14. Left: Biped Ground Station data tab. Right: Biped Ground Station camera tab.

5. Parameters

The parameters tab serves as a library for storing, organizing, and reusing sets of controller parameters, as shown in Figure 15. Users can save multiple revisions of the controller parameters, assign descriptive names, pin frequently used sets to the top for quick access, and reload them into the controller tab when needed. The interface also allows users to attach notes documenting the conditions, effects, or observations associated with each saved parameter set, providing valuable context during iterative tuning. When a saved set is loaded, the parameter values populate the controller tab fields but are not immediately applied. Changes only take effect when the user explicitly presses the apply button. This design prevents accidental overwrites of live controller parameters while enabling efficient comparison between different tuning profiles. With the parameter tab, users can maintain a well-organized archive of tested controller parameters, streamlining the process of optimizing controller performance and ensuring reproducibility across experiments.

6. Settings

The settings tab, illustrated in Figure 15, manages system configuration and utility functions. Here, the user specifies the IP address of the Biped hardware module used by the backend to establish network communications. The tab also provides controls for data logging, enabling users to record all incoming telemetry data and camera frames, as visualized by the data and camera tabs, into timestamped and structured files for offline review or analysis. Logging can be toggled at any time without interrupting ongoing communication or interface responsiveness. Furthermore, the settings tab offers optional theme customization, allowing users to adjust interface colors or load personalized visual profiles. While primarily cosmetic, this feature improves usability in shared lab environments and enhances accessibility for users with color-vision impairments or other visual sensitivities.

B. Communications

The communication module of the Biped Ground Station serves as the counterpart to the communication module implemented in the Biped firmware. Both modules share an identical communication interface, message formats, and serialization mechanisms, enabling seamless bidirectional data exchange between the Biped hardware module and the Biped Ground Station running on a remote host. Both the firmware and the Biped Ground Station employ the same lightweight header-only serialization library[93] to encode and decode structured data into transferable byte streams.

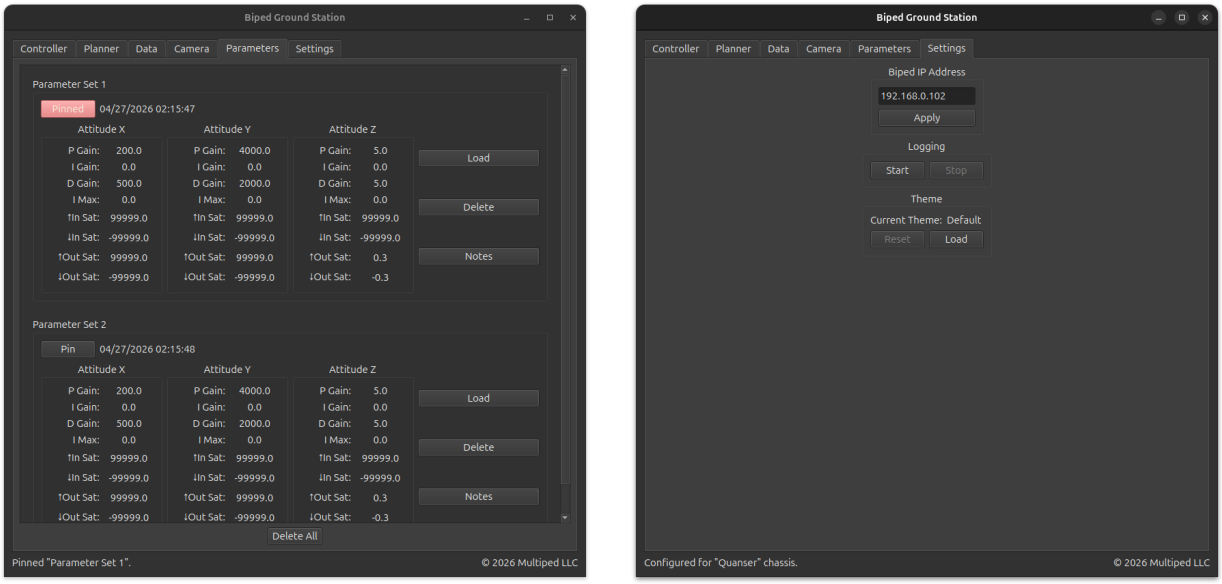

Figure 15. Left: Biped Ground Station parameters tab. Right: Biped Ground Station settings tab.

This unified approach ensures complete compatibility and guarantees that any message generated by one side can be accurately reconstructed by the other. At the core of the Biped Ground Station's communication module is the UDP interface, which mirrors the structure and functionality of its firmware counterpart, differing only in the underlying networking backend. While the firmware relies on the ESP-IDF networking stack on the ESP32-S3 MCU, the Biped Ground Station utilizes the C++ Boost.Asio library to implement the same asynchronous, cross-platform UDP interface. As in the firmware, the UDP interface provides high-level functions for transmitting and receiving either raw data buffers or string payloads, forming the foundation of all network communication between the two sides.

Consistent with the firmware design, all transmitted data structures within the Biped Ground Station are defined in a shared global type header. This file is an identical copy of the type definitions used in the Biped firmware, including the top-level `BipedMessage` structure that aggregates sensor data, controller parameters, and actuation commands. Complementing the UDP interface are two Biped Ground Station background daemons that handle continuous data exchange. The inbound daemon monitors the UDP socket, receives packets from the Biped hardware module, deserializes them into structured `BipedMessage` objects, and publishes the resulting data to other modules within the Biped Ground Station through software signals. The outbound daemon performs the reverse process, accepting data prepared by the user interface or internal modules, converting it into a serialized `BipedMessage`, and transmitting it to the hardware module over the network. Together, these communication daemons maintain an asynchronous, non-blocking network pipeline that keeps the Biped Ground Station interface responsive while providing reliable telemetry, parameter updates, and image streaming between the remote host and the Biped hardware module. Aside from the Biped Ground Station, any external software framework or third-party application can directly interface with the Biped hardware module by conforming to the same message format and serialization mechanics.

C. Daemons

The Biped Ground Station operates on an event-driven software architecture centered around background daemons and signal-slot interactions. Each daemon runs in its own dedicated thread, continuously processing data in the background while emitting or responding to asynchronous events that coordinate communication between software modules within the Biped Ground Station. The user interface module follows the same event-driven paradigm. Every button, text box, and widget emits signals that trigger corresponding slots in the module logic, enabling responsive and non-blocking interaction between the user interface and backend modules. Network communication between the Biped Ground

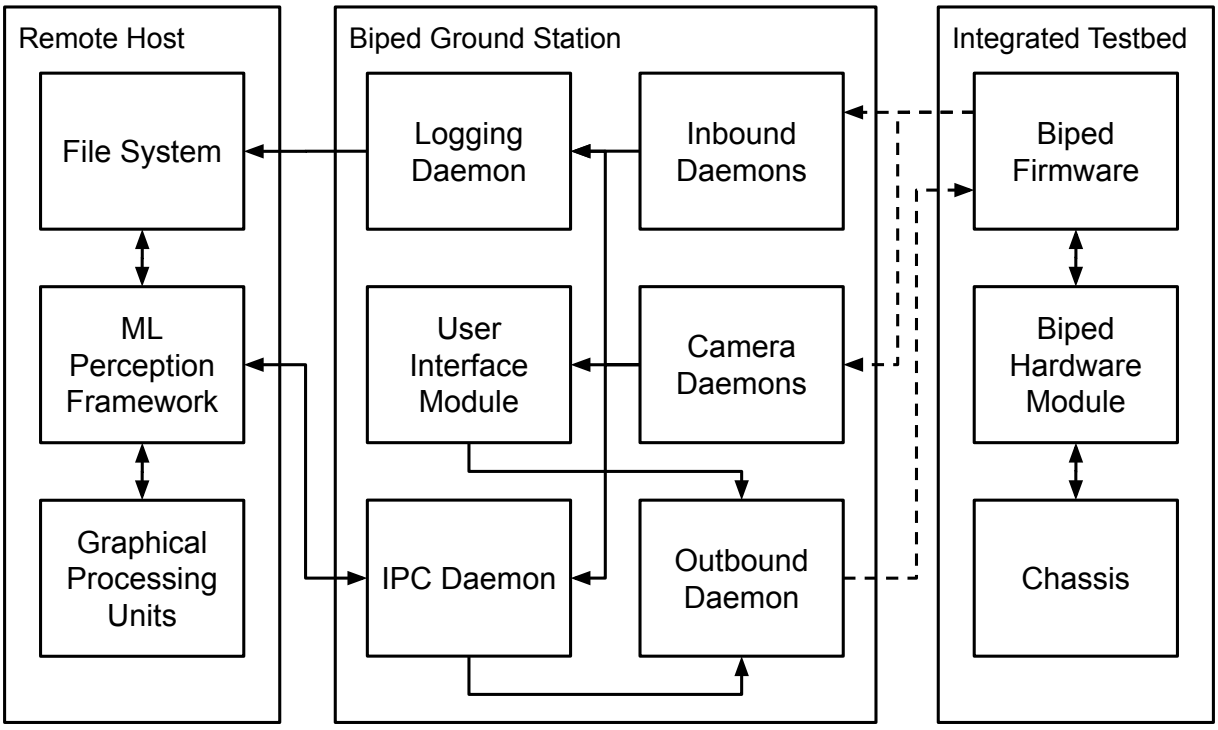

Figure 16. Events among the Biped Ground Station, its remote host, and the integrated testbed. Dashed arrows indicate wireless connections.

Station and the Biped hardware module is managed by the inbound and outbound daemons, as shown in Figure 16. The inbound daemon runs as a background thread that continuously polls the UDP interface, deserializes incoming packets into `BipedMessage`, and emits a signal carrying the deserialized message as its payload. Any component connected to this signal, such as the controller or data tabs within the user interface module, receives the `BipedMessage` and processes it according to its own slot implementation. Rather than emitting signals, the outbound daemon defines a slot for other modules to signal and send an outgoing `BipedMessage` to the Biped hardware module. When signaled, the outbound daemon serializes the supplied `BipedMessage` and transmits it via UDP.

The camera daemon operates under the same event-driven principle. Running in its own dedicated thread, the daemon continuously receives camera frame packets from the UDP interface. The camera daemon distinguishes fragmented camera frame packets using the frame boundaries transmitted between consecutive frames, reassembles the fragments into complete images, and emits a signal containing each reconstructed frame. Any module connected to this signal, most notably the camera tab in the user interface module, can then update its view asynchronously. The logging daemon subscribes to both the inbound daemon's message signal and the camera daemon's frame signal. When logging is activated in the settings tab of the user interface, the daemon records every received `BipedMessage` and camera frame to timestamped files on the file system. A separate joypad daemon provides optional integration with physical joystick hardware. The daemon binds USB joystick input events to the software joypad interface within the planner tab, allowing hardware control inputs to mirror the on-screen software joystick.

Beyond these core daemons, an additional daemon is included to interface with local inference models through inter-process communications (IPCs), more specifically, shared memory. As illustrated in Figure 16, this IPC daemon publishes incoming camera frames and `BipedMessage` telemetry data over the IPC, enabling Python-based perception frameworks implemented in PyTorch to perform online inference on the published data using GPU resources provided by the remote host. The resulting detection or recognition outputs are also returned through the IPC daemon, which distributes the results via signals to the Biped Ground Station modules. These inference outputs are visualized directly within the camera tab or transmitted back to the Biped hardware module for further onboard decision-making. This extensible design enables the Biped Ground Station to function not only as a graphical and logging interface, but also as

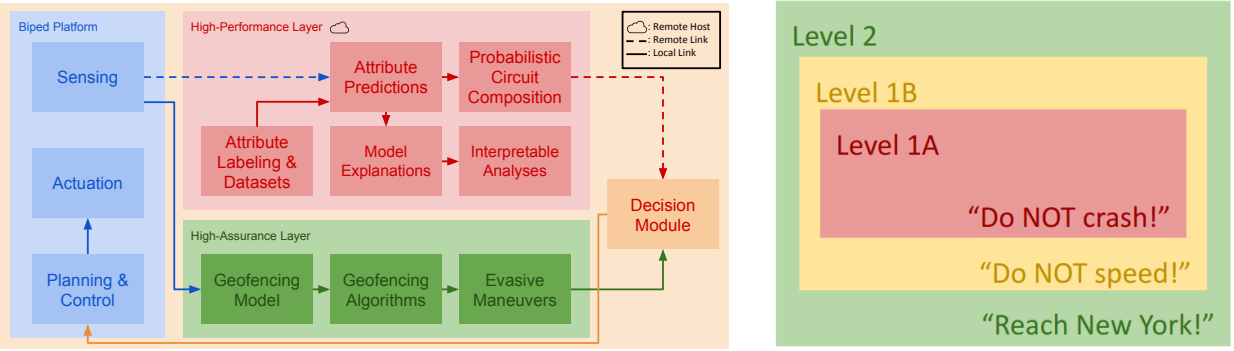

Figure 17. Left: safety-critical CPS architecture adopted in the experiments. Right: corresponding CPS DALs segregating system functionality by criticality.

a versatile middleware platform for integrating external perception and planning algorithms that otherwise exceed the computational capacity of the Biped hardware module.

VI. Cross-Platform Integration

A. Overview

This section presents the cross-platform integration and rapid deployment of the proposed Biped hardware, firmware, and software modules across heterogeneous chassis platforms through end-to-end, scenario-based experiments. To provide a structured context for integration, a layered CPS architecture is adopted, as shown in Figure 17, comprising a high-assurance layer (HAL), a high-performance layer (HPL), and the Biped platform. This architecture follows CPS development assurance levels (DALs), inspired by the DO-178C[96] standard, where system functionality is organized and segregated by criticality. At the highest DAL, i.e., Level 1A, the HAL enforces physical safety constraints independently of other subsystems. In this work, a position-based geofencing module is employed in the ground-based scenario to prevent entry into restricted regions. At Level 1B, the HPL provides perception-driven functionality for regulatory compliance through a robust and interpretable ML-based vision model, the Neural Probabilistic Circuit (NPC),[28,29] which performs real-time environmental recognition and interpretation. At the lowest DAL, i.e., Level 2, the Biped platform provides sensing, actuation, control, planning, and communication, and executes the nominal mission.

Within this layered structure, system behavior is governed hierarchically, with higher-criticality components overriding lower-criticality functions when necessary. This hierarchy enables safe interaction among perception, control, and mission execution while maintaining runtime safety guarantees. Using this architecture as the reference framework, the Biped platform is integrated with two distinct chassis: a ground-based TWSBR and an aerial Quanser helicopter. These platforms differ significantly in dynamics, actuation, and operational constraints. The TWSBR enables evaluation of perception-driven behaviors and safety enforcement through geofencing, while the aerial platform demonstrates hardware and control integration in a 3-DoF system and focuses on perception-driven behaviors in a UAM VTOL context. The objective of this section is to demonstrate the functionality and adaptability of Biped across platforms, enabling consistent deployment, operation, and experimentation. Through this cross-platform integration, the Biped platform establishes a practical foundation for evaluating autonomy pipelines in diverse CPS settings.

VII. Scenario

Two experimental scenarios are constructed to evaluate the integrated system across both the ground-based TWSBR and aerial Quanser helicopter platforms. These scenarios are designed to exercise the interaction among perception, control, and safety mechanisms within the layered CPS architecture, and to demonstrate system functionality and performance across both platforms. The ground-based experiment, shown in Figure 18, is implemented on the TWSBR

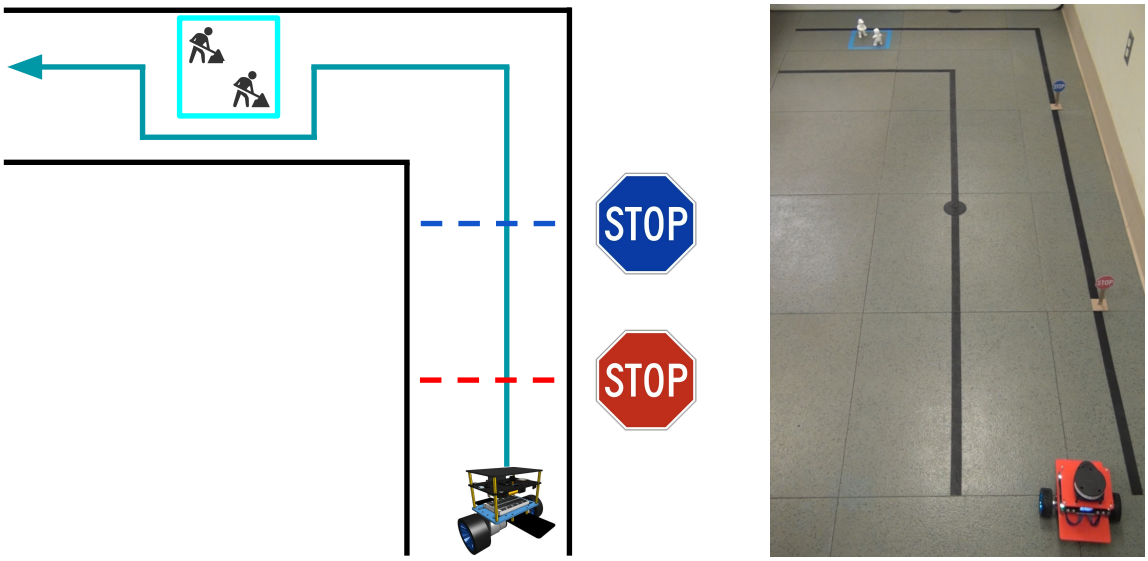

Figure 18. Left: illustration of the TWSBR experimental scenario. Right: corresponding physical implementation.

chassis and follows a predefined trajectory marked by lane boundaries. Along the first segment, a series of stop signs is placed along the path to evaluate regulatory compliance enforced by the HPL. The onboard camera provides real-time image input to the NPC, which performs recognition and interpretation of the stop signs. Upon recognition, the nominal mission is overridden, and a halt command is issued. To evaluate robustness, both standard and color-corrupted stop signs are included, including extreme variations such as blue stop signs. After stopping and proceeding past the signs, the system resumes the nominal mission and executes a turning maneuver into a second segment. In this segment, a position-based geofence representing a construction zone is introduced. As the platform approaches this region, the geofencing module within the HAL overrides the nominal mission and injects evasive maneuvers. This ground scenario demonstrates both HPL-driven perception and HAL-based runtime safety enforcement within an end-to-end system.

The aerial experiment, shown in Figure 19, is implemented on the Quanser helicopter platform and follows a simplified perception-driven task without geofencing. Instead of traveling along marked lanes, the platform performs controlled yaw motion, sweeping its field of view across the environment. The Biped module is mounted beneath the propeller assembly, with the onboard camera oriented downward. During operation, the platform rotates about its yaw axis while maintaining stable hover. A set of stop signs is placed beneath the platform within the camera's field of view. As the platform yaws and a stop sign enters view, the NPC performs real-time recognition using the downward-facing camera. Upon successful recognition, the yaw maneuver is halted, and the platform maintains a stationary hover for a short duration. After this pause, the yaw maneuver resumes until the next sign is encountered. This aerial scenario mirrors the perception-driven behavior of the ground experiment, framed in a UAM VTOL context. Together, the two scenarios provide complementary validation of the integrated system across distinct platforms and operating conditions.

A. Platform

The Biped hardware, firmware, and software modules are integrated with two distinct physical platforms: a ground-based TWSBR chassis and an aerial Quanser helicopter. Due to the streamlined design of the expansion and actuation interfaces, electrical integration between the Biped module and the two platforms, as illustrated in Figure 20, is straightforward. Through this integration, the same Biped hardware-software stack provides sensing, actuation, computation, and communication across both platforms, despite differences in dynamics, actuator characteristics, and experimental objectives. These two chassis serve as representative platforms for evaluating cross-platform integration

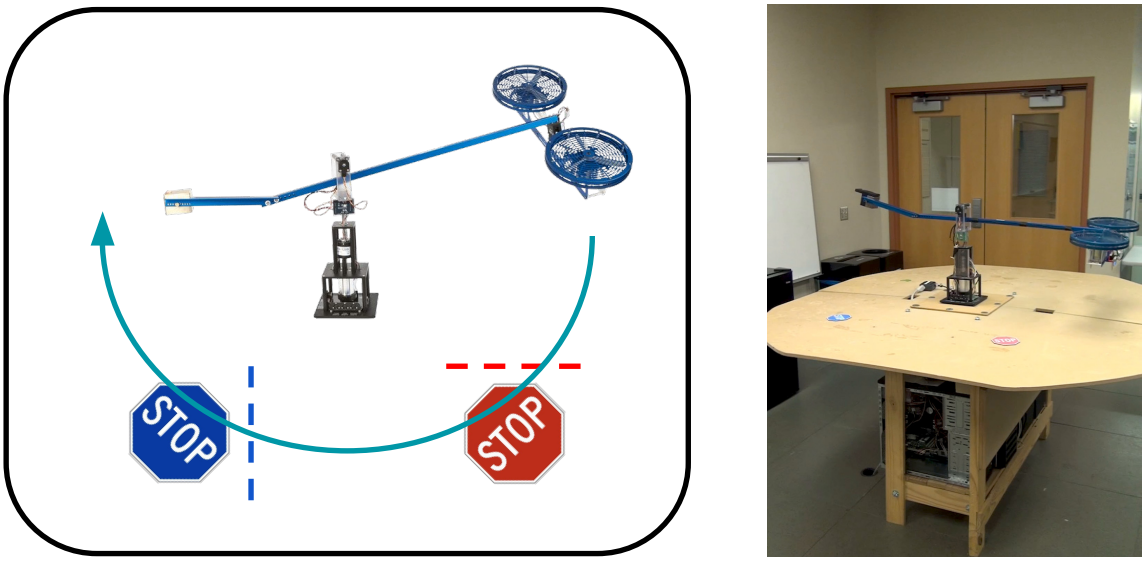

Figure 19. Left: illustration of the Quanser helicopter experimental scenario. Right: corresponding physical implementation.

of Biped in this work. The TWSBR supports closed-loop balancing, perception-driven recognition, and position-based geofencing, while the Quanser helicopter supports closed-loop attitude control and perception-driven recognition.

1. Quanser Helicopter

The Quanser helicopter exposes five electrical connectors at its base: three optical rotary encoder connectors for 3-axis attitude sensing and two propeller motor connectors for actuation. In its stock configuration, as shown in Figure 2, these connectors interface with Quanser's proprietary hardware stack, consisting of two bulky universal power modules,[49] one for each propeller motor, a proprietary DAQ board,[48] and a separate I/O card installed in a desktop workstation. Host-side software, such as MATLAB/Simulink or LabVIEW, communicates with the I/O card to read sensor data and drive the motors, thereby making the helicopter entirely dependent on a tethered workstation for operation. This configuration requires a significant physical footprint, involves complex wiring, and relies on proprietary hardware and software, resulting in a closed ecosystem that is difficult to extend or adapt for perception research. As shown in Figure 2, the Biped module replaces this entire hardware-software stack with a single compact hardware module running its modular firmware. The Biped hardware module integrates computation, sensing, actuation, and communication into a unified form factor powered by a standard USB-C adapter. With Biped, the integrated testbed operates untethered and host-independent for basic functions, simplifying integration for more advanced perception-based experiments.

Each Quanser encoder connector provides two channels along with 5 V and ground, while each propeller motor connector exposes a two-pin terminal. These motor connectors map directly to the actuator driver outputs within the actuation interface connectors on the Biped hardware module, enabling bidirectional PWM control through the onboard actuator drivers. The roll and pitch encoders are routed to the actuation interfaces, which expose two direct MCU GPIO pins originally intended for the TWSBR motor encoders. Since the Quanser encoders require a 5 V supply rather than the 3.3 V rail provided by the actuation interface, their power lines are connected to the 5 V rail exposed by the expansion interface. The yaw encoder is connected through the expansion interface, which provides an additional pair of MCU GPIO pins and the same 5 V rail. On the firmware side, all three encoders are read using the rotary encoder interrupt service framework, as detailed in Algorithm 2, enabling high-resolution sensing of roll, pitch, and yaw angles.

During initial testing, intermittent drift was observed in the roll and pitch encoder readings despite correct hardware and firmware configurations. Oscilloscope measurements confirmed that the encoders themselves were functioning correctly. The issue was traced to the multi-conductor slip ring on the Quanser helicopter chassis, which routes electrical

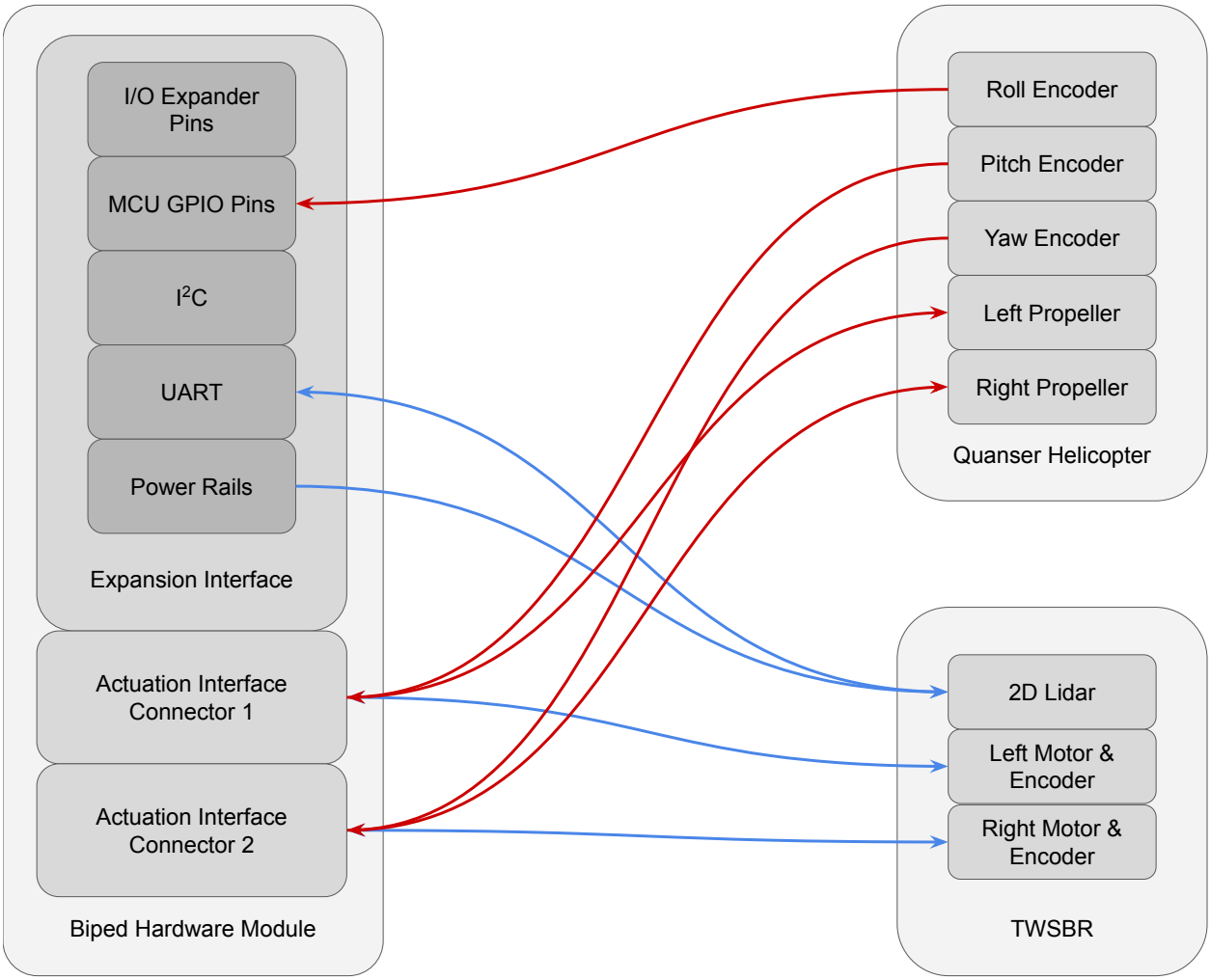

Figure 20. Hardware integration of the Biped hardware module with the Quanser helicopter and the TWSBR chassis.

connections through the yaw axis. Over time, the conductive surfaces within the slip ring accumulate debris and wear, leading to brief losses of electrical contact as the platform rotates. These contact losses result in missed encoder pulses, which manifest as gradual drift in the measured encoder steps during operation. To eliminate this issue, the roll and pitch encoders were wired directly to the Biped hardware module interfaces, bypassing the slip ring entirely. The yaw encoder does not traverse the slip ring and therefore does not require modification. The motor lines remain routed through the slip ring, since occasional contact irregularities only introduce minor perturbations in the PWM duty cycle and do not affect actuation performance, as the motors behave as mechanical low-pass filters.

After completing the slip ring bypass and wiring, the Biped firmware was rapidly reconfigured to support the integrated Quanser helicopter chassis. Each encoder channel pair was mapped to its corresponding MCU pins, the actuation channels were assigned to the two H-bridge drivers, and the controller cascade designed for the Quanser helicopter, as shown in Figure 11, was instantiated within the firmware controller module. Using real-time telemetry and online controller parameter tuning provided by the Biped Ground Station, all controller gains were tuned within minutes. For ease of development, initial testing was performed with the Biped hardware module placed externally on a table and connected to the Quanser helicopter through wiring harnesses, as shown in Figure 21. This configuration enabled rapid validation of sensing, actuation, and control, and was used to establish stable closed-loop attitude control using the unified Biped hardware-software stack.

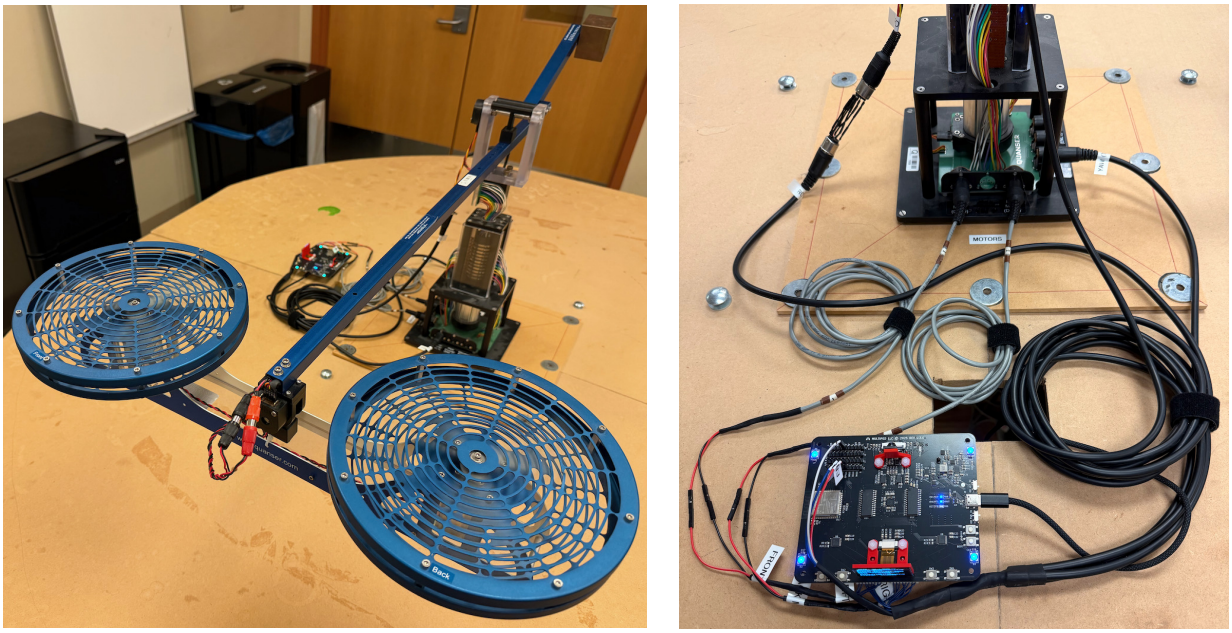

Figure 21. Left: Quanser helicopter under closed-loop control driven solely by the Biped hardware module in the table-top configuration. Right: corresponding table-top electrical connections between the Biped hardware module and the Quanser helicopter.

Following initial validation, the system was transitioned to a fully integrated configuration, in which the Biped hardware module is mounted directly beneath the roll encoder assembly between the two propeller motors, as shown in Figure 22. In this configuration, wiring is routed and secured within the chassis, and the Biped hardware module operates as an onboard embedded system. This integration enables perception tasks using the camera, now oriented downward, such as UAM VTOL scenarios involving recognition and hover-based interaction with ground objects.

2. *Two-Wheeled Self-Balancing Robot*

Common TWSBR constructions consist of separate hardware modules, as shown in Figure 3. These designs[52–56] often exhibit questionable reliability, are difficult to reproduce, and are not easily transferable across different chassis without substantial rewiring and redesign, making them unsuitable as general-purpose research platforms. In contrast, the Biped hardware module, designed as a self-contained and adaptable embedded platform, can be easily detached from one chassis and reintegrated into another. The previous educational version of the Biped hardware module, shown in Figure 23, illustrates the expected functionality for the TWSBR configuration. The TWSBR chassis employs a pair of motors mounted at the base, each with an extended rear axle coupled to a magnetic rotary encoder. Electrically, the educational Biped hardware module connects to the TWSBR chassis through two actuation interface connectors, each supplying both actuator connections and encoder input signals, with no additional wiring required. With these connections, the Biped firmware receives encoder readings and controls the TWSBR chassis by actuating the motors.

Integration with the latest Biped hardware module follows the same interface setup, with the fully constructed new TWSBR chassis shown in Figure 23. While the previous educational chassis relied on a layered assembly of fiberglass balancing pedals, a separate metal base plate, and independent motor mounts, the updated design introduces a one-piece 3D-printed frame that reduces weight and part count, simplifies assembly, and improves reproducibility. This new chassis integrates the motor mounts, balancing paddle, and base plate into a single structure that accepts standard standoffs for mounting the Biped hardware module. The updated chassis also includes a redesigned 3D-printed top plate that replaces the previous fiberglass construction and incorporates mounting points for a 2D rotating LiDAR sensor,[75] as shown in Figure 23. The LiDAR operates from a 5 V supply and communicates over a UART interface, both provided through the Biped expansion interface. By connecting the LiDAR directly to this interface, the system can acquire dense 2D range measurements for navigation experiments. Although fully integrated and operational, the

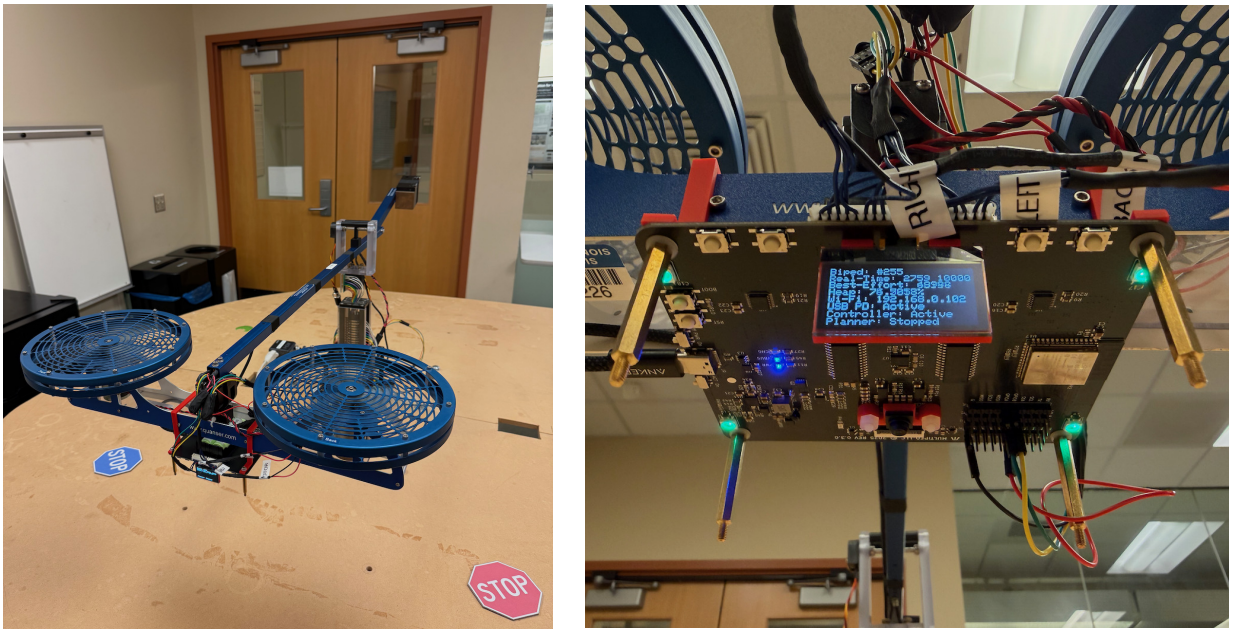

Figure 22. Left: Quanser helicopter under closed-loop control driven solely by the chassis-mounted Biped hardware module in a fully integrated aerial testbed configuration. Right: corresponding chassis-mounted electrical connections between the Biped hardware module and the Quanser helicopter.

LiDAR is not utilized in the current experimental setup. Nevertheless, the LiDAR integration enables future research on LiDAR-first safety-critical perception pipelines.[31,34–36]

The integrated TWSBR platform demonstrates complete end-to-end functionality across sensing, control, and communication. During testing, the Biped hardware module achieved stable closed-loop balancing, forward and backward motion, and turning maneuvers. All onboard sensors, including the IMU, rotary encoders, and camera, operated reliably throughout these tasks. Controller tuning for the TWSBR chassis was performed through the Biped Ground Station with real-time gain adjustment, without requiring repeated firmware modification.

3. Biped Ground Station

A key component of the integrated system is the interaction between the integrated embedded platform and the HPL running on a remote host. During operation, the camera module onboard the Biped hardware module streams camera frames to the Biped Ground Station, which serves as the interface between the Biped hardware module and the computational resources on the remote host running the Biped Ground Station. As shown in Figure 16, the Biped Ground Station publishes incoming camera frames, along with telemetry data, through its IPC daemon via shared memory. A separate Python program on the remote host retrieves the camera frames directly from shared memory and performs inference using the NPC. The NPC processes each frame to produce real-time classifications, attribute predictions, and model explanations, specifically most probable explanation (MPE), which identify the attribute combinations contributing most to each classification. These inference results are written back to shared memory and received by the Biped Ground Station. The outputs are visualized in real time by overlaying predictions and explanations onto the camera frames within the camera tab, as shown in Figure 14. Simultaneously, the results are transmitted back to the Biped hardware module, where they are used by the onboard decision-making and planning modules. For example, when a stop sign is detected, the system overrides the nominal mission and issues a halt command. More generally, inference outputs can influence control behavior depending on the scenario.

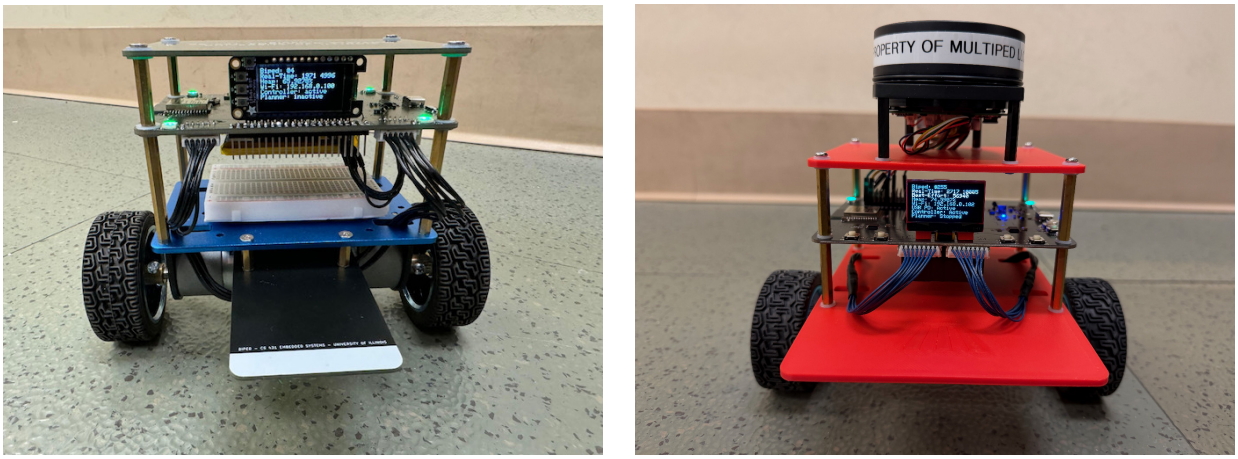

Figure 23. Left: educational TWSBR chassis with the prior Biped hardware module. Right: redesigned TWSBR chassis with the latest Biped hardware module as an integrated ground-based testbed.

B. Results

The experimental scenarios shown in Figures 18 and 19 were constructed in an indoor laboratory setting and evaluated using the integrated system. As described in Section VII, the setup includes a series of stop signs along the yaw path for the aerial scenario, and for the ground scenario, along the first segment, and a position-based geofence representing a construction zone in the second segment. Photographs of the constructed scenarios are shown in Figures 18 and 19, depicting the physical layouts used during evaluation. The system was tested over repeated runs to assess the interaction among nominal mission, perception-driven recognition, and safety enforcement within the closed-loop architecture.

Within the HPL, the NPC used in this experiment was trained on the German Traffic Sign Recognition Benchmark (GTSRB) dataset[97] and deployed directly within the integrated system without transfer learning or fine-tuning. Despite this, the NPC demonstrated consistent and reliable recognition performance in real-time operation across both ground and aerial scenarios. In all configurations, the system successfully recognized stop signs and issued halt commands, overriding the nominal mission as expected. The system stopped reliably regardless of the position of the stop sign, indicating the robust performance of the NPC under varying conditions. To further evaluate robustness under distribution shift, stop signs with extreme color variations, as shown in Figure 24, were introduced, including blue, green, and black variants in addition to the standard red sign. Across all cases, the system maintained correct behavior, consistently recognizing the stop sign and triggering the appropriate response. This observation is consistent with the NPC attribute selection ablation study,[28] which indicates that color is not a dominant attribute on the GTSRB dataset.

More importantly, the correctness of these decisions is not inferred solely from classification outputs but is supported by the interpretable nature of the NPC. During operation, the model produces MPE in real time, as shown in Figure 25, identifying the attribute combinations that contribute most to each classification. Inspection of these explanations shows that, when color is degraded or incorrect, the model relies on other attributes, specifically the octagonal shape and the presence of the "STOP" text, to reach the correct classification. This provides direct evidence that the model's behavior aligns with the intended semantic structure of the task based on the human definition of a stop sign. This capability is particularly important in safety-critical CPS for auditability and certifiability. Unlike conventional black-box models, which provide no insight into internal reasoning, the NPC enables real-time inspection of its decision and inference processes. In cases of misclassification or uncertainty, these explanations reveal which attributes are missing or incorrectly inferred, and whether correcting them would yield the correct classification, providing a principled basis for diagnosis and improvement. As a result, this experiment demonstrates that the interpretability of the NPC is not merely a theoretical property confined to datasets, but a practical feature that enhances the reliability, auditability, and trustworthiness of perception-driven autonomy.

Figure 24. Stop signs with extreme color variations used in the experiments.

During the experiment, an important limitation of the NPC's multi-task learning (MTL) attribute recognition model was observed. Specifically, the shared-encoder MTL used to jointly predict multiple attributes, i.e., color, shape, symbol, and text, introduced spurious correlations among them. Although the NPC attribute selection ablation study[28] identifies color as a non-dominant and independent attribute on the GTSRB dataset, perturbations to color in this physical setting, i.e., modifying a stop sign from red to blue, resulted in unintended degradation in the prediction of shape, symbol, and text, despite their semantic independence. This behavior is undesirable in both safety-critical and performance settings, where robustness to variations in non-dominant attributes is essential. To mitigate this issue, the original NPC pipeline[28,29] was restructured to decouple color prediction from the remaining attributes. Two inference passes were introduced: one operating on the original color input for color prediction, and a second operating on a grayscale version of the same input for shape, symbol, and text prediction. The outputs of these passes were then combined into a unified attribute prediction and provided as input to the probabilistic circuit (PC) for final reasoning. This modification leverages the interpretable and compositional nature of the NPC framework and does not require retraining of the underlying models. Empirically, the decoupled inference strategy significantly improved robustness to color perturbations. Variations in color no longer affected the prediction of shape and symbol attributes, confirming that the original MTL pipeline introduced unintended attribute coupling. This result highlights both a limitation of shared-encoder MTL in safety-critical perception and the advantages of the NPC framework: (1) its interpretability enables identification of such attribute-level spurious correlations, and (2) its modularity enables straightforward mitigation through simple architectural modification without retraining.

In the second segment of the ground-based TWSBR experiment, the system was evaluated on its ability to enforce safety constraints through geofencing via the HAL. As the platform approached the predefined construction zone, represented by the blue square in Figure 18, the HAL successfully overrode the nominal mission and injected evasive maneuvers to avoid the geofence. Across repeated runs, the system consistently avoided the geofence and continued along a safe trajectory, demonstrating effective runtime enforcement of safety constraints. Within this experiment, a simplified position-based geofencing algorithm was used to accommodate the kinematic characteristics of the TWSBR chassis. The objective of this scenario is to validate the safety enforcement mechanism and control authority of the HAL, rather than the performance of a specific geofencing algorithm. The results confirm that the HAL operates independently

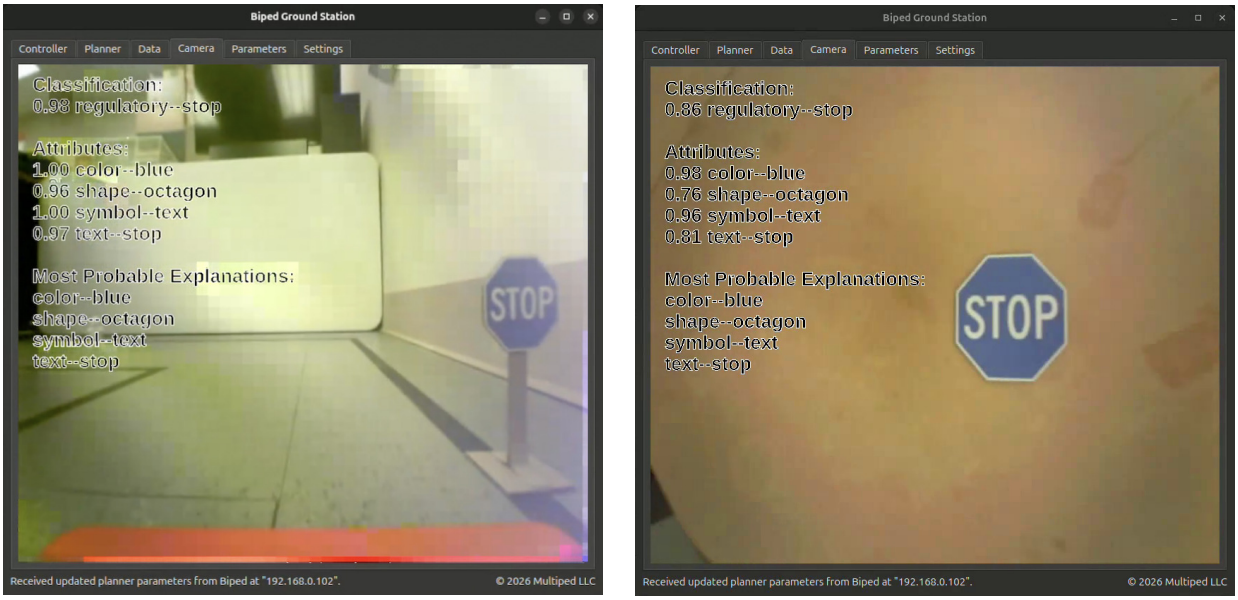

Figure 25. Left: Biped Ground Station camera tab showing NPC inference and explanation results in the ground scenario. Right: corresponding view for the aerial scenario.

of the nominal mission and retains high-criticality control authority within the system. When activated, the geofencing module overrides the nominal mission through the maneuver planner framework, as detailed in Algorithm 1 and in,[91] ensuring that safety constraints are satisfied. This behavior demonstrates the intended hierarchical structure of the proposed architecture, where Level 1A physical safety supersedes Level 1B regulatory safety and Level 2 nominal mission. Furthermore, the experiment shows that the system architecture is inherently algorithm-agnostic. Although a simplified geofencing strategy is used in this scenario, the same interface can support more advanced algorithms, such as the Beta Trajectory model,[98–100] without modification.

VIII. Conclusion and Future Work

A. Conclusion

This work presented Biped, a modular embedded hardware-software platform for developing and evaluating autonomy and perception pipelines in CPS and UAM. The proposed system integrates sensing, computation, actuation, power management, and communication within a unified hardware module, enabling compact and extensible deployment across both ground-based and aerial chassis. A layered firmware architecture supports structured implementation of sensing, control, planning, communication, and interrupt handling, while maintaining modularity and real-time performance. Complementing the hardware and firmware, the Biped Ground Station provides a flexible interface for real-time telemetry, visualization, parameter tuning, and integration with external computational resources. The platform was validated through cross-platform integration with both ground-based and aerial systems under the constructed experimental scenarios. On the ground-based TWSBR, the system demonstrated closed-loop interaction between perception-driven behavior and safety enforcement, where the NPC-based HPL enabled reliable stop-sign recognition under distribution shift, and the HAL enforced runtime safety through geofencing by overriding the nominal mission. On the aerial Quanser helicopter, the system demonstrated stable closed-loop attitude control together with perception-driven behavior in a UAM VTOL context. Across both platforms, the system operated consistently in real time, with interpretable NPC outputs providing insight into decision-making and enabling diagnosis of model behavior under perturbations. These results demonstrate that the Biped platform enables unified deployment of perception, control, and safety mechanisms across heterogeneous systems, establishing a practical foundation for rapid development and experimental validation of autonomy pipelines in safety-critical CPS environments.

B. Future Work

Several directions can further demonstrate the capabilities of the Biped platform for advanced autonomy applications. A key extension is the integration of additional safety-critical perception pipelines, as discussed in Section II, within the HAL. In particular, multi-modal LiDAR-first pipelines[31,34–36] can be incorporated to provide robust and verifiable obstacle detection at Level 1A using the 2D rotating LiDAR sensor[75] available on the TWSBR chassis. Such approaches enable detection-first safety mechanisms that operate independently of semantic recognition, further strengthening system reliability. At the HPL, the framework also supports integration of alternative perception and reasoning models. Beyond the NPC, other concept-based and probabilistic models[101–108] can be incorporated as complementary or redundant pipelines, enabling comparative evaluation and improving robustness through diversity. These evaluations can be further extended by incorporating standardized performance metrics, such as accuracy, precision, and recall, to quantify perception effectiveness under real-world conditions.

Acknowledgments

The material presented in this work is based upon work supported by the National Science Foundation (NSF) under Grant ECCS-2311085 and the National Aeronautics and Space Administration (NASA) under Grant 80NSSC22M0070. Any opinions, findings, conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1]Eskandarian, A., Wu, C., and Sun, C., "Research advances and challenges of autonomous and connected ground vehicles," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 22, No. 2, 2019, pp. 683–711.

[2]Zhao, J., Wu, Y., Deng, R., Xu, S., Gao, J., and Burke, A., "A survey of autonomous driving from a deep learning perspective," *ACM Computing Surveys*, Vol. 57, No. 10, 2025, pp. 1–60.

[3]Perez-Grau, F. J., Martinez-de Dios, J. R., Paneque, J. L., Acevedo, J. J., Torres-González, A., Viguria, A., Astorga, J. R., and Ollero, A., "Introducing autonomous aerial robots in industrial manufacturing," *Journal of Manufacturing Systems*, Vol. 60, 2021, pp. 312–324.

[4]Saunders, J., Saeedi, S., and Li, W., "Autonomous aerial robotics for package delivery: A technical review," *Journal of Field Robotics*, Vol. 41, No. 1, 2024, pp. 3–49.

[5]Kanellakis, C., Fresk, E., Mansouri, S. S., Kominiak, D., and Nikolakopoulos, G., "Autonomous visual inspection of large-scale infrastructures using aerial robots," *arXiv preprint arXiv:1901.05510*, 2019.

[6]Xu, J., Kendrick, K., and Bowers, A. R., "Update on Experiences of a Driver with Vision Impairment when Using a Tesla Car—Full Self-driving (Beta) in City Driving," *Optometry and Vision Science*, Vol. 100, No. 6, 2023, pp. 351–353.

[7]Bridgelall, R., Tolliver, D., et al., "Autonomous Aircraft: Challenges and Opportunities," 2023.

[8]Jung, S. and Ariyur, K. B., "Enabling operational autonomy for unmanned aerial vehicles with scalability," *Journal of Aerospace Information Systems*, Vol. 10, No. 11, 2013, pp. 517–529.

[9]Johnson, E. N., Proctor, A. A., Ha, J., and Tannenbaum, A. R., "Development and test of highly autonomous unmanned aerial vehicles," *Journal of Aerospace Computing, Information, and Communication*, Vol. 1, No. 12, 2004, pp. 486–501.

[10]Araujo, H., Mousavi, M. R., and Varshosaz, M., "Testing, validation, and verification of robotic and autonomous systems: a systematic review," *ACM Transactions on Software Engineering and Methodology*, Vol. 32, No. 2, 2023, pp. 1–61.

[11]Tan, Y. et al., "Paving the Way for Self-driving Cars-Software Testing for Safety-critical Systems Based on Machine Learning: A Systematic Mapping Study and a Survey," 2017.

[12]Garrow, L. A., German, B. J., and Leonard, C. E., "Urban air mobility: A comprehensive review and comparative analysis with autonomous and electric ground transportation for informing future research," *Transportation Research Part C: Emerging Technologies*, Vol. 132, 2021, pp. 103377.

[13]Thipphavong, D. P., Apaza, R., Barmore, B., Battiste, V., Burian, B., Dao, Q., Feary, M., Go, S., Goodrich, K. H., Homola, J., et al., "Urban air mobility airspace integration concepts and considerations," *2018 aviation technology, integration, and operations conference*, 2018, p. 3676.

[14]Bilgin, Z., Bronz, M., and Yavrucuk, I., "Experimental evaluation of robustness of panel-method-based path planning for urban air mobility," *AIAA AVIATION 2022 Forum*, 2022, p. 3509.

[15]Brown, A. and Harris, W. L., "Vehicle design and optimization model for urban air mobility," *Journal of Aircraft*, Vol. 57, No. 6, 2020, pp. 1003–1013.

[16]Song, K. and Yeo, H., "Development of optimal scheduling strategy and approach control model of multicopter VTOL aircraft for urban air mobility (UAM) operation," *Transportation research Part C: emerging technologies*, Vol. 128, 2021, pp. 103181.

[17]Park, J., Kim, I., Suk, J., and Kim, S., "Trajectory optimization for takeoff and landing phase of UAM considering energy and safety," *Aerospace Science and Technology*, Vol. 140, 2023, pp. 108489.

[18]Veneruso, P., Opromolla, R., Tiana, C., Gentile, G., and Fasano, G., "Sensing requirements and vision-aided navigation algorithms for vertical landing in good and low visibility UAM scenarios," *Remote Sensing*, Vol. 14, No. 15, 2022, pp. 3764.

[19]You, D. I., Jung, Y. D., Cho, S. W., Shin, H. M., Lee, S. H., and Shim, D. H., "A guidance and control law design for precision automatic take-off and landing of fixed-wing UAVs," *AIAA guidance, navigation, and control conference*, 2012, p. 4674.

[20]Kügler, M. E., Heller, M., and Holzapfel, F., "Automatic take-off and landing on the maiden flight of a novel fixed-wing UAV," *2018 Flight Testing Conference*, 2018, p. 4275.

[21]Smith, B., Stark, B., Zhao, T., and Chen, Y., "An outdoor scientific data drone ground truthing test site," *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2015, pp. 436–443.

[22]Sandino, J., Maire, F., Caccetta, P., Sanderson, C., and Gonzalez, F., "Drone-based autonomous motion planning system for outdoor environments under object detection uncertainty," *Remote Sensing*, Vol. 13, No. 21, 2021, pp. 4481.

[23]Brunswicker, S., Goppert, J., Gough, E., Lercel, D., Hwang, I., Kong, N., Sribunma, W., Deng, C., Shreekumar, J., Zoltowski, M., et al., "AIrTonomy: An Experimental Infrastructure for Testing Next-Generation Autonomous Aerial Vehicles," *AIAA AVIATION FORUM AND ASCEND 2025*, 2025, p. 3047.

[24]Cheng, S., Kim, M., Song, L., Yang, C., Jin, Y., Wang, S., and Hovakimyan, N., "Difftune: Auto-tuning through auto-differentiation," *IEEE Transactions on Robotics*, 2024.

[25]How, J. P., Behihke, B., Frank, A., Dale, D., and Vian, J., "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, Vol. 28, No. 2, 2008, pp. 51–64.

[26]Valenti, M., Bethke, B., Fiore, G., How, J., and Feron, E., "Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery," *AIAA guidance, navigation, and control conference and exhibit*, 2006, p. 6200.

[27]Cramer, N. B., Seruge, C., Keller, G., Grabbe, S. R., and Rieffel, E., "Enhanced UAS Availability via Vehicle to Vehicle Routing Scaled Experiments," *AIAA AVIATION 2020 FORUM*, 2020, p. 2850.

[28]Chen, W., Yu, S., Shao, H., Sha, L., and Zhao, H., "Neural probabilistic circuits: Enabling compositional and interpretable predictions through logical reasoning," *arXiv preprint arXiv:2501.07021*, 2025.

[29]Chen, W., Yu, S., Shao, H., Sha, L., and Zhao, H., "Neural Probabilistic Circuits: An Overview," *Eighth Workshop on Tractable Probabilistic Modeling*, 2025.

[30]Bansal, A., Kim, H., Yu, S., Li, B., Hovakimyan, N., Caccamo, M., and Sha, L., "Perception simplex: Verifiable collision avoidance in autonomous vehicles amidst obstacle detection faults," *Software Testing, Verification and Reliability*, Vol. 34, No. 6, 2024, pp. e1879.

[31]Liu, S., Yao, S., Fu, X., Tabish, R., Yu, S., Bansal, A., Yun, H., Sha, L., and Abdelzaher, T., "Taming algorithmic priority inversion in mission-critical perception pipelines," *Communications of the ACM*, Vol. 67, No. 2, 2024, pp. 110–117.

[32]Bansal, A., Kim, H., Yu, S., Li, B., Hovakimyan, N., Caccamo, M., and Sha, L., "Verifiable obstacle detection," *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, 2022, pp. 61–72.

[33]Bansal, A., Yu, S., Kim, H., Li, B., Hovakimyan, N., Caccamo, M., and Sha, L., "Synergistic redundancy: Towards verifiable safety for autonomous vehicles," *arXiv preprint arXiv:2209.01710*, 2022.

[34]Chen, J., Yu, S., Tabish, R., Bansal, A., Liu, S., Abdelzaher, T., and Sha, L., "Lidar cluster first and camera inference later: A new perspective towards autonomous driving," *arXiv preprint arXiv:2111.09799*, 2021.

[35]Liu, S., Yao, S., Fu, X., Shao, H., Tabish, R., Yu, S., Bansal, A., Yun, H., Sha, L., and Abdelzaher, T., "Real-time task scheduling for machine perception in intelligent cyber-physical systems," *IEEE Transactions on Computers*, Vol. 71, No. 8, 2021, pp. 1770–1783.

[36]Liu, S., Yao, S., Fu, X., Tabish, R., Yu, S., Bansal, A., Yun, H., Sha, L., and Abdelzaher, T., "On removing algorithmic priority inversion from mission-critical machine inference pipelines," *2020 IEEE Real-Time Systems Symposium (RTSS)*, IEEE, 2020, pp. 319–332.

[37]Kawamura, E., Dolph, C., Kannan, K., Lombaerts, T., and Ippolito, C. A., "Simulated vision-based approach and landing system for advanced air mobility," *AIAA SciTech 2023 Forum*, 2023, p. 2195.

[38]Multiped, "Biped," 2025.

[39]Barry, R. et al., "FreeRTOS," *Internet, Oct*, Vol. 4, 2008, pp. 18.

[40]Lee, T., "The Quanser Platform for Control Systems Research Validation," .

[41]Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You only look once: Unified, real-time object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[42]He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[43]Luo, B., Wu, H.-N., and Huang, T., "Optimal output regulation for model-free quanser helicopter with multistep Q-learning," *IEEE Transactions on Industrial Electronics*, Vol. 65, No. 6, 2017, pp. 4953–4961.

[44]Mansor, H., Esa, M., Gunawan, T. S., and Janin, Z., "Design of travel angle control of quanser bench-top helicopter using mamdani-based fuzzy logic controller," *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 17, No. 2, 2020, pp. 815–825.

[45]Hairon, A. M., Mansor, H., Gunawan, T. S., and Khan, S., "Travel angle control of quanser bench-top helicopter based on Quantitative Feedback Theory technique," *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 1, No. 2, 2016, pp. 310–318.

[46]Jafri, M., Mansor, H., and Gunawan, T., "Development of fuzzy logic controller for quanser bench-top helicopter," *IOP Conference Series: Materials Science and Engineering*, Vol. 260, No. 1, 2017, pp. 012015.

[47]Dyvik, M., Fjereide, D. E., and Rotondo, D., "Modeling and identification of the Quanser Aero using a detailed description of friction and centripetal forces," *Scandinavian Simulation Society*, 2023, pp. 246–253.

[48]Chen, F., "Research on Speed Control System Based on Quanser Intelligent Data Acquisition Board," *2023 3rd International Symposium on Computer Technology and Information Science (ISCTIS)*, IEEE, 2023, pp. 896–899.

[49]Ionete, C., "LQG/LTR controller design for flexible link quanser real-time experiment," *Proc. of Int. Symp. SINTES11, Craiova, Romania*, Vol. 1, 2003, pp. 49–54.

[50]Purdue University, "Facilities / Current Assets," 2025.

[51]Thavitchasri, P., Maneetham, D., and Crisnapati, P. N., "Intelligent surface recognition for autonomous tractors using ensemble learning with BNO055 IMU sensor data," *Agriculture*, Vol. 14, No. 9, 2024, pp. 1557.

[52]Juang, H.-S. and Lum, K.-Y., "Design and control of a two-wheel self-balancing robot using the arduino microcontroller board," *2013 10th IEEE International Conference on Control and Automation (ICCA)*, IEEE, 2013, pp. 634–639.

[53]Iwendi, C., Alqarni, M. A., Anajemba, J. H., Alfakeeh, A. S., Zhang, Z., and Bashir, A. K., "Robust navigational control of a two-wheeled self-balancing robot in a sensed environment," *IEEE Access*, Vol. 7, 2019, pp. 82337–82348.

[54]Shekhawat, A. S. and Rohilla, Y., "Design and control of two-wheeled self-balancing robot using Arduino," *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, IEEE, 2020, pp. 1025–1030.

[55]Abdelgawad, A., Shohdy, T., and Nada, A., "Model-and data-based control of self-balancing robots: Practical educational approach with labview and arduino," *IFAC-PapersOnLine*, Vol. 58, No. 9, 2024, pp. 217–222.

[56]Hellman, H. and Sunnerman, H., "Two-Wheeled Self-Balancing Robot: Design and control based on the concept of an inverted pendulum," 2015.

[57]Cameron, N., "ESP32 microcontroller," *ESP32 Formats and Communication: Application of Communication Protocols with ESP32 Microcontroller*, Springer, 2023, pp. 1–54.

[58]Bogoslavskyi, I. and Stachniss, C., "Fast range image-based segmentation of sparse 3D laser scans for online operation," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 163–169.

[59]Bogoslavskyi, I. and Stachniss, C., "Efficient online segmentation for sparse 3D laser scans," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, Vol. 85, No. 1, 2017, pp. 41–52.

[60]Espressif Systems, *ESP32-S3 Series Datasheet*, 2025.

[61]Litayem, N., "Scalable smart home management with ESP32-S3: A low-cost solution for accessible home automation," *2024 International Conference on Computer and Applications (ICCA)*, IEEE, 2024, pp. 1–7.

[62]Espressif Systems, *ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U Datasheet*, 2025.

[63]Gay, W., "MCP23017 I/O Port Extender," *Custom Raspberry Pi Interfaces: Design and build hardware interfaces for the Raspberry Pi*, Springer, 2017, pp. 153–168.

[64]Semiconductor Components Industries, *FUSB302B Programmable USB Type-C Controller w/ PD*, 2021.

[65]Monolithic Power Systems, *MP2760 I2C-Controlled, 1-Cell to 4-Cell Buck-Boost Charger with NVDC Power Path Management*, 2023.

[66]Huertas, C. J. O., *New Design of an Electronic Power System for the Cosmic X-Ray Background Nanosatellite-3*, Master's thesis, Morehead State University, 2019.

[67]Lopez, D., Posada, D., and Henderson, T., "EagleCam Lunar Mission-Power Subsystem Analysis and Lessons Learned," *2024 IEEE Aerospace Conference*, IEEE, 2024, pp. 1–12.

[68]Dietz, H., Abney, D., Eberhart, P., Santini, N., Davis, W., Wilson, E., and McKenzie, M., "ESP32-Cam as a programmable camera research platform," *Imaging*, Vol. 232, No. 2, 2022, pp. 10–2352.

[69]Lu, Y., Tao, J., and Wang, K., "Light field sensor and real-time panorama imaging multi-camera system and the design of data acquisition," *7th International Symposium on Advanced Optical Manufacturing and Testing Technologies: Optical Test and Measurement Technology and Equipment*, Vol. 9282, SPIE, 2014, pp. 571–577.

[70]OmniVision Technologies, *OmniVision Serial Camera Control Bus (SCCB) Functional Specification*, 2007.

[71]Jian, H., "Design of angle detection system based on MPU6050," *7th International Conference on Education, Management, Information and Computer Science (ICEMC 2017)*, Atlantis Press, 2016, pp. 6–8.

[72]Biggs, H., Starr, A., Smith, B., de Lima, S., Sykes, J., Haddadchi, A., Smart, G., and Hicks, M., "Kinematic loggers—development of rugged sensors and recovery systems for field measurements of stone rolling dynamics and impact accelerations during floods," *Sensors*, Vol. 22, No. 3, 2022, pp. 1013.

[73]Yao, F., Xue, Z., Ou, H.-c., Gao, W.-f., and Li, N., "Study on the method to improve the driving stability of intelligent regulator motor," *International Field Exploration and Development Conference*, Springer, 2023, pp. 1563–1572.

[74]Kabala, T. and Weremczuk, J., "Smart wireless low power valve positioner dedicated for use in aviation laboratories," *Aviation*, Vol. 29, No. 2, 2025, pp. 95–103.

[75]Ikhtiar, A., Qurban, D., and Samo, S., "Design and Control of a Self-Driving Vehicle using RP Lidar," *2023 International Conference on Robotics and Automation in Industry (ICRAI)*, IEEE, 2023, pp. 1–5.

[76]Huang, Z., Tan, H., Peng, Y., and Lei, L., "Design of intelligent fire detection and alarm system," *International Conference on Electronics, Electrical and Information Engineering (ICEEIE 2024)*, Vol. 13445, SPIE, 2024, pp. 706–714.

[77]Mujić, M., Cogo, E., and Bešić, I., "Embedded Real-Time Analogue NeoPixel Clock," *2024 23rd International Symposium INFOTEH-JAHORINA (INFOTEH)*, IEEE, 2024, pp. 1–5.

[78]Espressif Systems, "ESP IoT Development Framework," 2025.

[79]Espressif Systems, "Arduino core for the ESP32, ESP32-C3, ESP32-C5, ESP32-C6, ESP32-H2, ESP32-P4, ESP32-S2 and ESP32-S3." 2025.

[80]Clemencic, M. and Mato, P., "A CMake-based build and configuration framework," *Journal of Physics: Conference Series*, Vol. 396, IOP Publishing, 2012, p. 052021.

[81]a9183756-gh, "Arduino CMake Toolchain," 2020.

[82]Premerlani, W. and Bizard, P., "Direction cosine matrix imu: Theory," *Diy Drone: Usa*, Vol. 1, 2009.

[83]VectorNav Technologies, "2.1 Reference Frames," 2025.

[84]Sands, W. A., "Angular kinematics," *The Science of Gymnastics*, Routledge, 2017, pp. 98–106.

[85]Wrona, M., "Roll and Pitch Angles From Accelerometer Sensors," 2020.

[86]Welch, G., Bishop, G., et al., "An introduction to the Kalman filter," 1995.

[87].NET nanoFramework, "Quadrature Rotary Encoder," 2025.

[88]Yue, D., Han, Q.-L., and Peng, C., "State feedback controller design of networked control systems," *Proceedings of the 2004 IEEE International Conference on Control Applications, 2004.*, Vol. 1, IEEE, 2004, pp. 242–247.

[89]Garcia, C. E., Prett, D. M., and Morari, M., "Model predictive control: Theory and practice—A survey," *Automatica*, Vol. 25, No. 3, 1989, pp. 335–348.

[90]Cao, C. and Hovakimyan, N., "Design and analysis of a novel l1 adaptive controller, part i: control signal and asymptotic stability," *2006 American Control Conference*, IEEE, 2006, pp. 3397–3402.

[91]Yu, S., "Flight maneuver automation for system analysis of small fixed-wing UAVs," *IDEALS*, 2019.

[92]Dantsker, O. D., Yu, S., Vahora, M., and Caccamo, M., "Flight testing automation to parameterize unmanned aircraft dynamics," *AIAA Aviation 2019 Forum*, 2019, p. 3230.

[93]eyalz800, "zpp serializer," 2021.

[94]FreeRTOS, "FreeRTOS scheduling (single-core, AMP and SMP)," 2025.

[95]Sha, L., Rajkumar, R., and Lehoczky, J. P., "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Transactions on computers*, Vol. 39, No. 9, 2002, pp. 1175–1185.

[96]Rierson, L., *Developing safety-critical software: a practical guide for aviation software and DO-178C compliance*, CRC Press, 2017.

[97]Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C., "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, Vol. 32, 2012, pp. 323–332.

[98]Yu, S., "Geofencing trajectory estimation for small-scale fixed-wing UAVs," *Illinois Digital Environment for Access to Learning and Scholarship (IDEALS)*, 2026.

[99]Theile, M., Yu, S., Dantsker, O. D., and Caccamo, M., "Trajectory estimation for geo-fencing applications on small-size fixed-wing UAVs," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 1971–1977.

[100]Theile, M. and Yu, S., "Kinematic Model for Fixed-Wing Aircraft with Constrained Roll-Rate," *Illinois Digital Environment for Access to Learning and Scholarship (IDEALS)*, 2018.

[101]Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P., "Concept bottleneck models," *International conference on machine learning*, PMLR, 2020, pp. 5338–5348.

[102]Espinosa Zarlenga, M., Barbiero, P., Ciravegna, G., Marra, G., Giannini, F., Diligenti, M., Shams, Z., Precioso, F., Melacci, S., Weller, A., et al., "Concept embedding models: Beyond the accuracy-explainability trade-off," *Advances in neural information processing systems*, Vol. 35, 2022, pp. 21400–21413.

[103]Kim, E., Jung, D., Park, S., Kim, S., and Yoon, S., "Probabilistic concept bottleneck models," *arXiv preprint arXiv:2306.01574*, 2023.

[104]Yeh, C.-K., Kim, B., Arik, S., Li, C.-L., Pfister, T., and Ravikumar, P., "On completeness-aware concept-based explanations in deep neural networks," *Advances in neural information processing systems*, Vol. 33, 2020, pp. 20554–20565.

[105]Kazhdan, D., Dimanov, B., Jamnik, M., Liò, P., and Weller, A., "Now you see me (cme): concept-based model extraction," *arXiv preprint arXiv:2010.13233*, 2020.

[106]Barbiero, P., Ciravegna, G., Giannini, F., Zarlenga, M. E., Magister, L. C., Tonda, A., Lió, P., Precioso, F., Jamnik, M., and Marra, G., "Interpretable neural-symbolic concept reasoning," *International Conference on Machine Learning*, PMLR, 2023, pp. 1801–1825.

[107]Rodriguez, D. M., Cuellar, M. P., and Morales, D. P., "Concept logic trees: enabling user interaction for transparent image classification and human-in-the-loop learning: DM Rodríguez et al." *Applied Intelligence*, Vol. 54, No. 5, 2024, pp. 3667–3679.

[108]Ciravegna, G., Barbiero, P., Giannini, F., Gori, M., Lió, P., Maggini, M., and Melacci, S., "Logic explained networks," *Artificial Intelligence*, Vol. 314, 2023, pp. 103822.