AURA: Hardware Architecture, Dual-Rail Power Isolation, and Calibration Strategy for a Budget Autonomous Departmental Assistant Robot

Manthan Digambar Chavan[1] Pranali Murlidhar Meshram[2] Vedant Yuvraj Dhumal[3]
Dr. Jayashri Satre[4] Prof. Dhiraj Tikar[5]
[1,2,3,4,5]Department of Electrical Engineering
[1,2,3,4,5]Trinity College of Engineering and Research, Pune, India

Abstract — Practical deployment of mobile departmental service robots is regularly held back by steep component costs, motor-induced supply noise on embedded compute boards, and navigational safety hazards. This paper reviews the preliminary hardware architecture of AURA (Academic Utility Robotic Assistant), a planned voice-directed assistant constrained to a ₹45,000 prototype budget (alongside a documented ₹66,000–₹70,000 reference build). Physical assembly and sensor characterization remain pending; therefore, this work presents no fabricated experimental figures, focusing instead on system topology, electrical isolation boundaries, and empirical calibration protocols. The core architecture divides computation between an ARM Linux host (ROCK 5B+ 8GB baseline, with Axon Vichaarak evaluated as an alternative) for perception and an ESP32 microcontroller for deterministic low-level actuation. To protect digital logic against motor transients, a 3S 5200 mAh lithium-ion (Li-ion) logic supply is electrically isolated from an independent 3S 2200 mAh lithium-polymer (LiPo) motor pack. Safety relies on a hardwired emergency-stop (E-stop) circuit, mechanical bumpers, ultrasonic ranging, and 2D LiDAR. The paper also documents where the budget build gives up hardware margin and how calibration recovers it, fixes power-integrity, thermal and safety tests before assembly, and outlines how TypeSafe's Jev System One decision model operates above the Universal Serial Bus (USB) link to triage telemetry and gate navigation goals without touching low-level motion safety.

Keywords: AURA, Differential-Drive Robotics, Dual-Rail Power Isolation, ESP32 Real-Time Control, Hardware Safety Boundary, Motor Calibration, Jev by TypeSafe

I. INTRODUCTION

Departmental service robots can streamline recurring indoor routines, including student attendance sessions, visitor guidance, and official notice delivery. Industrial automated platforms, however, rely on expensive brushless actuators, high-resolution encoders, and custom power distribution hardware that exceed academic prototyping budgets. Conversely, naive budget prototypes frequently suffer compute resets from transient motor loads, heading drift caused by uncalibrated drives, and hazardous software-dependent emergency braking.

AURA (Academic Utility Robotic Assistant) targets this gap through a ₹45,000 hardware architecture sized for 5–6 daily trips across a single-floor Electrical Engineering facility. The design keeps core navigation autonomy by confining cost cuts to non-critical areas—brushed DC gearmotors, standard display panels and shop-fabricated chassis parts—while protecting compute, Light Detection and Ranging (LiDAR), encoder feedback, dual power rails and physical interlocks. Because physical assembly is pending, this review examines the planned electrical, structural and sensor subsystems alongside a benchtop calibration method designed to extract predictable kinematics from budget hardware.

This paper makes four contributions: (1) a compute split that leaves all timing-critical input/output on an ESP32 while the Linux host handles perception and planning; (2) a dual-rail power layout with a hardwired E-stop that works without any software; (3) a characterize–calibrate–compensate–validate method for getting repeatable motion out of commodity parts; and (4) validation tests and targets, fixed before assembly, together with a costed upgrade path. Section II reviews related hardware practice, Section III states the design approach, Sections IV–VI cover the architecture, power and safety, and calibration, and Sections VII–VIII deal with validation, limitations and upgrades.

II. REVIEW OF RELATED HARDWARE ARCHITECTURES

A. Heterogeneous Compute Split (SBC and Microcontroller)

Running Robot Operating System 2 (ROS 2) navigation, Simultaneous Localization and Mapping (SLAM) and local Artificial Intelligence (AI) alongside microsecond motor loops on an unpatched Linux kernel introduces scheduling jitter into encoder sampling and Pulse-Width Modulation (PWM) timing [1], [2]. Moving motion control to an external microcontroller removes this bottleneck. Dedicating an ESP32 to Proportional-Integral-Derivative (PID) control, Inertial Measurement Unit (IMU) polling and hardware interrupts frees the ARM Single-Board Computer (SBC) for perception and keeps motor timing deterministic.

B. Odometry Error and Budget Component Compensation

Differential-drive dead reckoning accumulates trajectory error from unequal wheel diameters, track-width tolerances and actuator mismatch, made worse by floor slip [3]. On platforms using budget geared direct-current (DC) motors, assuming symmetric actuator gains causes severe rotational drift. Borenstein and Feng showed that the two dominant systematic errors, unequal wheel diameters and an uncertain wheelbase, can be measured and corrected with a simple square-path test known as UMBmark [4]. Reducing this error further requires PWM-versus-RPM (revolutions per minute) profiling of both drives, closed-loop encoder feedback, rate-gyro integration and 2D LiDAR scan matching within the navigation stack [3].

C. Layered Safety in Low-Speed Indoor Robots

Safety on small indoor robots is usually built in layers, so that no single sensor or program carries the whole burden. The innermost layer has to be a stop function that does not depend on software; ISO 13850 describes the emergency-stop function as one started by a single human action that brings the machine to a safe state ahead of its other functions [5].

All rights reserved by www.ijsrd.com

Around it sit contact sensing (bumpers), short-range proximity sensing (ultrasonic) and, outermost, map-based obstacle avoidance from LiDAR costmaps [2]. Each layer covers blind spots of the next, which matters a great deal when the individual sensors are inexpensive.

D. Decoupling System-One AI Decisions from Physical Actuation

Connecting learning models to a mobile base needs a strict boundary, so that inference latency or a network stall can never delay motion gating. TypeSafe's Jev System One model fits this role: it evaluates typed questions against a system state and returns typed Choice, Score and Noul answers with probabilities instead of generated text [6], [7]. Keeping Jev on the Linux host—above the USB link to the ESP32—allows quick classification of navigation goals and hardware health flags without bypassing any low-level actuator interlock.

III. Design Approach and Budget Philosophy

The V1 build is a function-preserving prototype: no core feature is removed to meet the ₹45,000 ceiling. The savings come from hardware margin instead—motor consistency, mechanical finish, camera and audio quality, display quality and thermal headroom. The working rule is that the complete system must be predictable, measurable and safe, not that every part must be premium. Every budget component therefore goes through the same four steps: characterize it, calibrate it, compensate in software where the physics allows, and validate the result before the configuration is locked.

The budget protects six items first: the compute board, the LiDAR, encoder-equipped motors, the motor driver, the E-stop and fuse, and the two separate power domains. A premium microphone array, a better display and an industrial chassis are deliberately left out, since none of them affects autonomy or safety in a controlled indoor setting.

IV. Proposed AURA V1 Hardware Architecture

A. Compute Selection

High-level perception, ROS 2 Jazzy/Nav2, FastAPI and local AI run on the ROCK 5B+ 8GB (RK3588 System-on-Chip [SoC]: 4× Cortex-A76 + 4× Cortex-A55 cores, Mali-G610 GPU, 6-TOPS Neural Processing Unit [NPU] and LPDDR5 memory), which is the preferred platform. An Axon Vichaarak board is kept as an alternative, but only after it passes a compatibility matrix covering Linux/ARM support, ROS 2 Jazzy installation, USB bandwidth, the RPLiDAR, camera, audio and display interfaces, NVMe storage, the serial link to the ESP32, sustained CPU/GPU/NPU load, thermal behavior and power draw. Its full specification and ROS 2 support were not documented at the time of writing, so the rest of the design is deliberately not built around it.

B. ESP32 Real-Time Controller

Across the USB/serial link, the Linux host sets navigational intent by publishing /cmd_vel, while firmware on the ESP32 checks each command against live sensor states. The microcontroller manages PWM duty cycles, encoder counting, PID velocity control, MPU6050 IMU polling, three HC-SR04 ultrasonic rangefinders, bumper interrupts, E-stop sensing, a software watchdog and the Bluetooth Low Energy (BLE) attendance beacon; its UART, SPI, I²C and pulse-counting peripherals cover all of these. Put simply, the host decides what should happen and the ESP32 decides whether it can be done safely right now. Fig. 1 shows how sensors and actuators are shared between the two boards.

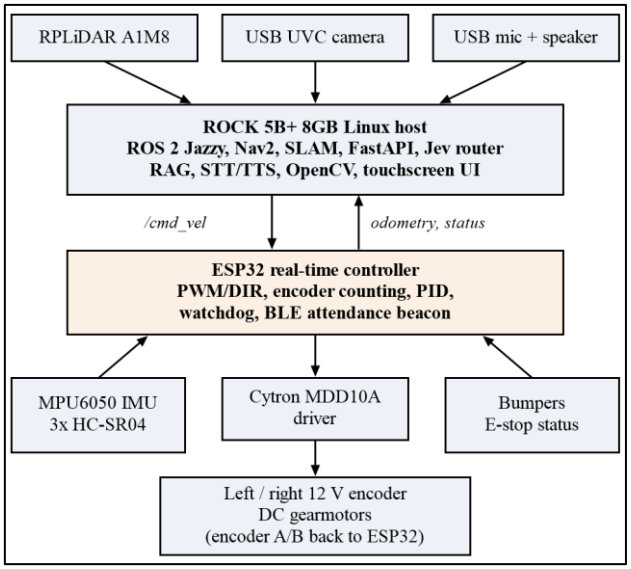


Fig. 1: Proposed compute split between the Linux host and the ESP32 real-time controller.

C. Drive Train

AURA uses two 12 V geared DC motors with quadrature encoders in a differential-drive layout, with a rear caster. The ESP32 turns velocity commands into PWM and direction signals for a Cytron MDD10A-class dual driver and reads both encoders back to close the loop. Open-loop driving—applying a fixed duty cycle and hoping the robot runs straight—is ruled out from the start. Before commissioning, the driver's supply voltage, peak current, motor stall current, connector ratings, heat dissipation and common-ground arrangement are all checked against the actual motors.

D. Sensing Layers

The RPLiDAR A1M8 is the primary navigation sensor and feeds /scan to SLAM, localization and the Nav2 costmaps. The MPU6050 adds gyroscope and accelerometer data; it is mounted rigidly, and its axes are checked against the robot frame before it is trusted in localization. Three HC-SR04 sensors (left, front and right) form a secondary close-range layer for objects the LiDAR plane can miss, such as low obstacles near the front; their blind zones and false triggers will be measured during testing. Two or more bumper switches give contact detection and are wired fail-safe, so that a broken lead reads as a fault rather than as a healthy switch. Printed ArUco markers [8] at HOME, ATTENDANCE_POINT, the Head of Department (HOD) office and the laboratories can later be added as visual landmarks for precise stopping and recovery; they cost almost nothing and use the existing camera.

E. V1 Component List and Budget Allocation

Table I details the proposed Bill of Materials (BOM), totaling ₹43,100–₹44,500 within the ₹45,000 ceiling. Any price

All rights reserved by www.ijsrd.com

increase must be absorbed elsewhere; safety-critical hardware is never removed to make the numbers work.

Subsystem	Proposed V1 Component	Qty	Est. (₹)
Compute	ROCK 5B+ / 5B 8GB	1	19,000
Storage	NVMe 256 GB	1	1,550
Controller	ESP32 microcontroller	1	450
Driver	Cytron MDD10A-class	1	1,800
Drive	12 V ~100 RPM encoder DC	2	2,160
Navigation	RPLiDAR A1M8 2D scanner	1	7,880
IMU	MPU6050 gyro/accel	1	155
Safety	HC-SR04 ultrasonic	3	210
Safety	Mechanical bumper switch	2	100
Safety	Physical E-stop + fuse	1 set	1,070
HMI	7-inch HDMI touchscreen	1	2,000
Vision	USB UVC camera	1	1,000
Audio	USB mic + Class-D + spkr	1 set	1,000
Logic Power	3S ~5200 mAh Li-ion	1	2,561
Motor Power	3S ~2200 mAh LiPo	1	1,465
Mechanical	2WD chassis + caster	1 set	2,099
Total	Estimated upper bound		44,500

Table I: Proposed Aura V1 Hardware Bill of Materials (₹45,000 Ceiling)

V. Power, Safety, and Mechanical Design

A. Dual-Domain Power Rail Isolation

Brushed DC motors generate inductive current spikes, voltage sags and regenerative transients during reversal. Powering compute and drive from one shared battery risks brownout resets of the SBC. AURA therefore uses two electrically isolated battery rails: a 3S ~5200 mAh lithium-ion (Li-ion) logic pack, regulated for the ROCK 5B+, the 7-inch High-Definition Multimedia Interface (HDMI) display, the ESP32 and the sensors, and an independent 3S ~2200 mAh high-discharge Lithium Polymer (LiPo) motor pack feeding the Cytron MDD10A driver. The choice of chemistry follows the loads. The logic side draws a steady, moderate current for long periods, which suits the higher energy density and longer cycle life of Li-ion cells, while the motor side sees short, heavy current bursts at start-up and reversal, which a high-discharge LiPo pack handles better. Each load gets its own DC/DC regulation, sized from measured current rather than nominal ratings. The two grounds meet at a single star point to avoid ground loops. Separate packs do not, by themselves, make every signal safe to connect: the PWM and direction lines from the ESP32 to the driver still need that shared reference, so the grounding is planned on paper rather than left to whatever wiring is convenient. Fig. 2 shows both domains and the one place where they meet.

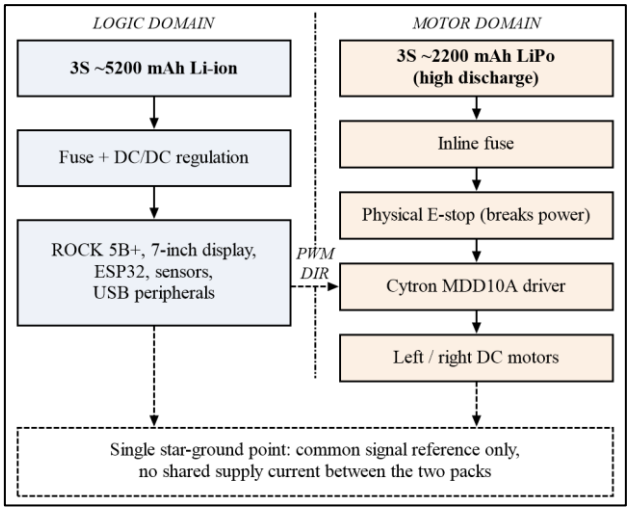


Fig. 2: Proposed dual-rail power architecture with a hardwired E-stop in the motor domain.

B. Mandatory Hardware E-Stop and Layered Safety

The primary E-stop switch and inline fuse are wired directly between the motor pack and the MDD10A driver, so they break motor current mechanically, independent of Linux, ROS 2, Python, Wi-Fi, FastAPI, any language model or any cloud service. This follows the general design principles for emergency-stop functions in ISO 13850 [5]. Releasing or resetting the switch must not restart motion by itself; the robot stays still until a fresh command arrives. Secondary protection comes from fail-safe bumpers triggering high-priority ESP32 stop interrupts, the three forward HC-SR04 sensors guarding close blind zones, and the RPLiDAR A1M8 providing 360° obstacle scanning for the Nav2 costmaps [2].

C. Battery Handling

Both lithium packs are treated as a hazard in their own right: a charger matched to each chemistry (with balance charging for the LiPo pack), fire-safe storage, rated connectors, fusing and physical protection are all required, and swollen or damaged packs are never used. The first powered runs are done at low speed while battery voltage, driver temperature, regulator temperature and compute stability are logged.

D. Mechanical Target and Peripheral Integration

The proposed chassis is 400–450 mm wide and 350–400 mm deep, with 90–100 mm wheels and a rear caster. Heavy batteries sit low for stability, while the RPLiDAR is raised for an unobstructed 360° view and the display is placed at a comfortable height for users. Exact dimensions will be frozen in the digital twin rather than guessed. Mechanical faults such as a misaligned wheel or a loose motor mount are measured and fixed before any PID or Nav2 tuning, because a robot that physically pulls to one side cannot always be corrected in software. A USB Video Class (UVC) camera handles Quick Response (QR) verification and recognition of a small registered faculty set. Speech uses a ₹1,000–₹1,500 audio package—a USB microphone, a Class-D amplifier and a 3–5 W speaker—with the microphone kept away from the motors and the speaker mechanically decoupled from chassis vibration.

All rights reserved by www.ijsrd.com

VI. BUDGET COMPENSATION AND CALIBRATION METHODOLOGY

A. Motor and Differential-Drive Characterization

Under the proposed protocol, each commodity motor is profiled on its own rather than assumed identical to its partner. As listed in Table II, bench tests measure the PWM dead-band, loaded RPM response, direction symmetry and encoder stability before the ESP32 PID gains are tuned [9]. The kinematic model behind this tuning is the standard one for a differential-drive base [10]. With right and left wheel speeds v_R and v_L and track width b, the body velocities are

$$v = (v_R + v_L) / 2, \quad \omega = (v_R - v_L) / b. \tag{1}$$

The ESP32 uses the same relation in reverse, turning a /cmd_vel request (v, ω) into two wheel set-points,

$$v_R = v + \omega b / 2, \quad v_L = v - \omega b / 2, \tag{2}$$

and the distance covered by each wheel is estimated from encoder counts as

$$\Delta s = \pi D \, \Delta n / C, \tag{3}$$

where D is the effective wheel diameter, Δn the counts in one control period and C the counts per wheel revolution. Since D and b only exist as nominal values on a datasheet, both are treated as calibration variables rather than constants. PID gains are first tuned on measured wheel speed alone. Effective wheel diameter and track width are then refined through repeated 1 m straight-line runs and 90°/180°/360° rotation trials, in the spirit of UMBmark [4], until the remaining error is repeatable and can be traced to geometry, motor mismatch or controller tuning.

Test	Measured Parameter	Calibration Purpose
PWM sweep	RPM across PWM levels	Feed-forward control curve
Startup	Minimum start PWM	Dead-zone compensation
Loaded test	RPM under chassis load	Realistic PID tuning
Direction	Forward/reverse symmetry	Sign and scaling parity
Encoder	Counts/rev and jitter	Accurate odometry config

Table II: Motor Characterization and Calibration Protocol

B. Odometry Fusion

Wheel odometry alone is not trusted. To limit drift along corridors, localization fuses encoder displacement, MPU6050 yaw rate and RPLiDAR scan matching within ROS 2, using an extended Kalman filter of the kind provided by the robot_localization package [11] together with SLAM Toolbox for mapping [12]. All of these parameters are first exercised on a Unified Robot Description Format (URDF) model in Gazebo [13], so the chassis dimensions can be frozen before any metal is cut. The navigation speed starts at 0.15–0.25 m/s and is raised only after braking, obstacle avoidance, odometry and motor-temperature tests pass.

C. Where the Budget Build Gives Up Margin

Table III lists, area by area, what the ₹45,000 build gives up compared with the reference build and how the design compensates. Writing these trade-offs down in advance makes it clear which test results will reflect the hardware and which will reflect the calibration effort.

Area	₹45K V1 Effect	Compensation
Motors	More mismatch and backlash	Per-motor profiling + encoder PID
Mechanical	More alignment variation	Careful fabrication + geometry calibration
Camera	Less low-light margin	Controlled lighting, registered faculty set
Audio	Less far-field margin	Short distance, VAD, isolated speaker
NVMe	Lower endurance	Health monitoring, sensible logging
Power	Basic protection	Correct fusing, isolation, thermal checks
Cooling	Less thermal headroom	Active cooling + sustained-load test
Display	Lower visual quality	Used as a functional UI only

Table III: V1 Hardware Compromises and Compensation

D. Jev at the Host Decision Boundary

Above the navigation stack, TypeSafe's Jev model sits in the FastAPI Application Programming Interface (API) router and serves two functions: (1) a Choice question maps a validated request to a fixed coordinate goal, such as HOD_OFFICE or ATTENDANCE_POINT, before any /cmd_vel is sent, and (2) Score and Noul questions triage logged telemetry anomalies—thermal throttling, battery sag, repeated bumper stops—for operator alerts. Jev is reached over a hosted HTTP API [7], so a network loss can only delay a goal or an alert; it can never command motion or bypass the ESP32 safety loops, and the router falls back to fixed waypoint rules when the API is unavailable.

VII. VALIDATION PLAN AND TARGETS

A. Bring-Up Order

Integration follows a fixed order—electronics, ESP32 firmware, mechanics, ROS simulation, navigation, backend, RAG/AI, voice, mobile app and, last, full integration—and each layer must expose a stable interface before the next one depends on it. The first month is software-first: the digital twin, FastAPI, a dashboard and a basic RAG service run before any final hardware is assembled.

B. Power-Integrity and Thermal Tests

For power integrity, motor start-up, both motors under load, rapid stops, direction reversals and repeated acceleration are run while the logic-rail voltage and the host's stability are logged. The test passes only if no motor event resets or crashes the ROCK 5B+. For thermal behavior, the full workload—ROS 2, Nav2, LiDAR, camera, FastAPI, database, speech-to-text and text-to-speech, AI and the UI—runs in progressively longer sessions while CPU temperature, clock behavior, RAM use and storage temperature are recorded. If throttling or instability appears, cooling and workload scheduling are fixed before any new feature is added.

All rights reserved by www.ijsrd.com

C. Engineering Targets

Table IV summarizes the target validation metrics defined for future testing. None of them has been measured yet.

Parameter	Proposed Engineering Target
Normal speed	0.15–0.25 m/s controlled indoor motion
Daily duty cycle	5–6 controlled departmental trips/day
Route completion	≥ 95% in mapped single-floor runs
Safety stop	100% in controlled obstacle/E-stop tests
Logic rail stability	Zero SBC resets under peak motor load
Thermal soak	Full workload with no throttling-induced faults

Table IV: Proposed V1 Hardware Validation Targets (Unmeasured)

VIII. Limitations and Upgrade Path

Known V1 hardware limitations include gearhead backlash, basic battery protection on both packs, an indoor-only 2D LiDAR, consumer-grade sensors and BLE Received Signal Strength Indicator (RSSI) distance variance [14]. The V1 is nevertheless designed as an upgrade-ready platform rather than disposable hardware. A ₹66,000–₹70,000 reference design, costed in Table V, mainly buys margin: better motor consistency, mechanics, audio, camera, display, power hardware and cooling.

Subsystem	Allowance (₹)
ROCK compute + storage	21,500
ESP32 + motor driver	2,600
Higher-quality encoder motors	4,500
Navigation + sensor layer	8,500
Better touchscreen	6,000
Better camera	2,500
Better audio capture/output	5,000
Dual batteries + power protection	5,100
Chassis, wheels, mounting, cooling	8,200
Hardware subtotal	63,900
Fabrication / contingency reserve	2,000–6,000
Reference total	≈ 66,000–70,000

Table V: ₹66,000–₹70,000 Upgraded Reference Allowances

Beyond that, the planned path runs through a V2 with a custom Printed Circuit Board (PCB), better audio and mechanics, and a V3 with stronger AI, vision and sensors, toward an industrial version. That version would add a smart Battery Management System (BMS) with fuel gauging and telemetry, Brushless DC (BLDC) drives over Controller Area Network (CAN), a Computer Numerical Control (CNC) aluminium frame, 16–32 GB industrial compute, and a safety chain with dual-channel E-stop, a motor power contactor, safety-rated sensors and a hardware watchdog.

IX. Conclusion

This review has presented AURA's proposed ₹45,000 V1 hardware architecture and shown how SBC/ESP32 partitioning, dual-rail Li-ion/LiPo isolation, hardwired E-stop wiring and per-motor characterization can deliver predictable indoor autonomy from accessible components. The compromises of the budget build are listed openly, together with the calibration step that answers each one. Adding TypeSafe's Jev System One model at the host API router improves waypoint gating and telemetry triage without putting AI latency anywhere near motor actuation. Future work will validate physical dimensions in the Gazebo Harmonic digital twin, then run the power-integrity, thermal and safety tests on the assembled chassis against the targets in Table IV.

Acknowledgment

The authors thank Dr. Jayashri Satre and Prof. Dhiraj Tikar, Department of Electrical Engineering, Trinity College of Engineering and Research, Pune, for their guidance throughout this hardware review.

References

[1] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, May 2022.

[2] S. Macenski, F. Martín, R. White, and J. Ginés Clavero, "The Marathon 2: A navigation system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 2718–2725.

[3] J. Borenstein, H. R. Everett, and L. Feng, "Where am I? Sensors and methods for mobile robot positioning," Univ. Michigan, Ann Arbor, MI, USA, Tech. Rep. UM-MEAM-94-21, 1996.

[4] J. Borenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots," *IEEE Trans. Robot. Autom.*, vol. 12, no. 6, pp. 869–880, Dec. 1996.

[5] *Safety of Machinery — Emergency Stop Function — Principles for Design*, ISO 13850:2015, International Organization for Standardization, Geneva, 2015.

[6] TypeSafe AI, "Introduction," *TypeSafe Documentation*, 2026. [Online]. Available: https://docs.typesafe.ai/introduction (accessed Nov. 17, 2026).

[7] TypeSafe AI, "System One," *TypeSafe Documentation*, 2026. [Online]. Available: https://docs.typesafe.ai/concepts/system-one (accessed Nov. 17, 2026).

[8] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognit.*, vol. 47, no. 6, pp. 2280–2292, 2014.

[9] K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, 2nd ed. Research Triangle Park, NC: Instrument Society of America, 1995.

[10] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. Cambridge, MA: MIT Press, 2011.

[11] T. Moore and D. Stouch, "A generalized extended Kalman filter implementation for the Robot Operating System," in *Intelligent Autonomous Systems 13*, Advances in Intelligent Systems and Computing, vol. 302. Cham: Springer, 2016, pp. 335–348.

All rights reserved by www.ijsrd.com

[12] S. Macenski and I. Jambrecic, "SLAM Toolbox: SLAM for the dynamic world," *J. Open-Source Softw.*, vol. 6, no. 61, p. 2783, 2021.
[13] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2004, pp. 2149–2154.
[14] R. Faragher and R. Harle, "Location fingerprinting with Bluetooth Low Energy beacons," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 11, pp. 2418–2428, Nov. 2015.

All rights reserved by *www.ijsrd.com*