Autonomous Mobile Robot (AMR) Design Document

1. Introduction

1.1 Purpose of the Document: This document provides comprehensive design specifications and architecture for the Autonomous Mobile Robot (AMR) being developed by Mission Mobile Medical. This AMR is intended to facilitate the automated swapping of modular medical clinic bodies on an Electric Vehicle (EV) platform at the "Mission Station." The development will occur on a scaled-down 48x48 inch version utilizing the same components as the final 9x9 foot AMR.

1.2 Project Goals: The primary goal is to create a reliable and efficient AMR system capable of autonomously transferring approximately 9x9x9 foot modular medical clinic units (CT scan, Mammogram, Dental, etc.) between storage bays at the Mission Station and the EV platform. This will enable rapid reconfiguration of the mobile medical units based on operational needs.

1.3 System Overview: The AMR is a wheeled robot featuring a lifting platform and a module transfer mechanism. It will navigate within the Mission Station using a 360-degree 3D lidar, precisely align with modules and the EV platform using laser distance sensors and QR code scanners, lift and level its platform based on inclinometer feedback, engage with the module via an electromagnet attached to a linear actuator, and then pull the module onto its chassis. The AMR will then transport the module and perform the reverse loading procedure. The system is controlled by a Raspberry Pi 5, communicating with motor controllers and absolute encoders over a CAN bus, with an inclinometer via RS-485, and with lidars and bumpers through digital inputs. A 7-inch touchscreen provides a user interface, and multiple E-stops ensure safety.

2. System Architecture

2.1 Overall System Diagram: (Diagram to be developed illustrating the AMR within the Mission Station, interacting with module storage, the EV platform, and showing the flow of data from sensors to the Raspberry Pi 5 and control signals to actuators)

2.2 AMR Subsystems:

2.2.1 Chassis: A base providing structural support for all other subsystems. Dimensions are 48x48 inches for the development platform and approximately 9x9 feet for the final product. Material and detailed structural design are to be defined based on load requirements and weight considerations.

2.2.2 Drive System: Employs four independent differential drive units, one at each corner. Each unit consists of two motor-driven wheels with integrated relative encoders (300,000

ticks per wheel rotation, 20.25-inch circumference), controlled by four dual motor controllers (CAN IDs 0x601-0x604) communicating via CANopen at 500 kbps. Corner angles relative to the chassis are measured by absolute encoders (CANopen, ID 0x188, 20,480 ticks per 360° rotation, up to 3.2 turns, 500 kbps).

2.2.3 Lifting Mechanism: A platform mounted on four independently controlled electric lift motors, each with an integrated relative encoder and two limit switches (digital input to Raspberry Pi 5) to detect the upper and lower limits of travel. Platform leveling is achieved using feedback from a dual-axis RS-485 digital inclinometer connected to the Raspberry Pi 5. The lift motors are controlled by single motor controllers (CAN IDs to be defined) via CANopen at 500 kbps.

2.2.4 Module Transfer Mechanism: A linear actuator mounted on the top platform equipped with an electromagnet. Specifications for the actuator (force, stroke length) and electromagnet (holding force) are to be defined based on the module's weight and transfer requirements.

2.2.5 Power System: (Specifications for the battery capacity, voltage levels for motors and electronics, power distribution network, and the integration of the emergency stop (E-stop) system to cut power to all drive and lift motors will be defined).

2.2.6 Sensor Suite:

Drive Wheel Encoders: Incremental encoders integrated with drive motors, 300,000 ticks per wheel rotation, read via CANopen from dual motor controllers.

Corner Angle Encoders: Absolute encoder (AMS AS5048A or equivalent), CANopen protocol, CAN ID 0x188, 20,480 ticks per 360° rotation, up to 3.2 turns, 500 kbps. Data interpretation involves reading two bytes for single-turn position (little-endian) and two bytes for turn count (little-endian). Absolute position is calculated as single_turn_pos + (turns_count

65536). Angle conversion: angle = (position % 20480) / 20480

360.0, turns = position / 20480, absolute_angle = turns

360.0 + angle. Zeroing can be done via hardware (yellow wire to ground) or software (SDO write 2B 03 60 00 00 00).

Lift Motor Encoders: Incremental encoders integrated with lift motors, read via CANopen from single motor controllers (IDs to be defined).

Limit Switches: Two per lift motor (mechanical or inductive), connected to digital input pins on the Raspberry Pi 5.

Obstacle Detection: Two 270-degree Lidars (e.g., Hokuyo UST-20LX), connected via digital inputs to the Raspberry Pi 5, providing planar scans for obstacle detection.

Navigation: 360-degree 3D Lidar (e.g., Velodyne Puck LITE), interface protocol to Raspberry Pi 5 to be defined (likely Ethernet).

Alignment: Multiple Laser Distance Sensors (e.g., Keyence LR-TB5000), interface protocol to Raspberry Pi 5 to be defined (potentially digital or analog outputs).

Positioning: Multiple QR Code Scanners (narrow field of view, e.g., Cognex DataMan 8050), interface protocol to Raspberry Pi 5 to be defined (potentially USB or serial).

Level Detection: Dual-Axis RS-485 Digital Inclinometer (e.g., SCA610), connected to Raspberry Pi 5 via RS-485, providing pitch and roll angles.

Collision Detection: Soft Bumpers with digital outputs (high when compressed), connected to digital input pins on the Raspberry Pi 5.

2.2.7 Control System: Raspberry Pi 5 will serve as the central control unit, running a real-time operating system (RTOS) to manage concurrent tasks. A custom circuit board connected to the 40-pin GPIO header will facilitate interfacing with CAN, RS-485, and digital I/O. A 7-inch touchscreen display connected via HDMI and USB will provide a graphical user interface for control, status monitoring, and debugging.

3. Mechanical Design

3.1 AMR Chassis

3.1.1 Dimensions:

Development Platform: 48 inches (length) x 48 inches (width). Height to be determined based on component integration.

Final Product: Approximately 108 inches (length) x 108 inches (width) to accommodate the 9x9 foot module. Height to be determined.

3.1.2 Material: High-strength aluminum alloy (e.g., 6061-T6) for both versions to balance strength and weight. Frame design will utilize a modular approach for scalability.

3.1.3 Structural Considerations: Finite Element Analysis (FEA) will be used to optimize the chassis design for load bearing (AMR weight + module weight), rigidity, and stress distribution during module transfer and movement. Mounting points for all subsystems will be integrated into the frame design.

3.2 Drive System

3.2.1 Wheel Configuration: Four independent differential drive units, each with two high-traction polyurethane wheels. Wheel diameter will be determined based on desired speed and ground clearance.

3.2.2 Motor Specifications: Brushless DC motors with high torque and efficiency, operating at 24V. Target maximum speed of 3000 RPM (at the motor shaft, gear reduction will affect wheel RPM). Torque requirements will be calculated based on maximum load, acceleration, and incline.

3.2.3 Encoder Specifications: Integrated magnetic rotary encoders with a resolution of 300,000 ticks per motor shaft revolution. Gear reduction ratio will be factored in to calculate wheel rotation resolution at the wheel.

3.2.4 Differential Drive Implementation: Each corner unit will have independent speed and direction control of its two motors. The Raspberry Pi 5 will implement control algorithms to coordinate the four units for precise omnidirectional movement, including Ackerman-like steering for smooth turns and crab-like motion for lateral movement. The absolute encoders on each corner unit will provide feedback on the wheel assembly's angle relative to the chassis, crucial for coordinated movements and accurate odometry.

3.3 Lifting Mechanism

3.3.1 Actuation Method: Four ball screw linear actuators driven by electric motors (24V). Each actuator will be independently controlled to allow for platform leveling.

3.3.2 Lifting Capacity: The lifting capacity of each actuator and the overall lifting system will be determined based on the estimated weight of the modular medical units. A safety factor will be included in the design.

3.3.3 Leveling System Requirements: The Raspberry Pi 5 will receive real-time pitch and roll data from the dual-axis RS-485 inclinometer. A PID (Proportional-Integral-Derivative) control loop will be implemented to adjust the extension of each of the four lift actuators to maintain a level platform within a specified tolerance.

3.3.4 Limit Switches: Two limit switches (one at the upper limit and one at the lower limit) will be integrated with each ball screw actuator to provide hardware-based safety mechanisms to prevent over-extension or retraction. These will be wired as digital inputs to the Raspberry Pi 5 for monitoring.

3.4 Module Transfer Mechanism

3.4.1 Actuation Method: A heavy-duty linear actuator (specifications for force and stroke to be determined) will be mounted centrally on the top platform.

3.4.2 Attachment Method: A high-power electromagnet will be attached to the end of the linear actuator. A corresponding ferromagnetic plate will be designed and integrated into the structure of the modular medical units.

3.4.3 Pulling Force Requirements: The linear actuator must exert sufficient force to overcome the static friction and rolling resistance of the module as it is pulled onto the AMR platform. Calculations will consider the maximum expected weight of a module and the surface conditions.

3.4.4 Alignment Considerations: Precision in the horizontal alignment of the AMR with the module and the vertical alignment (height matching achieved by the lift mechanism) will be critical for a successful and safe transfer. Laser distance sensors will provide feedback to the control system for fine-tuning the AMR's position before engaging the electromagnet and initiating the pull.

4. Electrical Design

4.1 Power Distribution: A 48V (or suitable voltage based on motor requirements) battery pack will power the AMR. A robust power distribution system will be designed with appropriate fusing and circuit protection for all motors, sensors, and the Raspberry Pi 5. The emergency stop (E-stop) circuit will be hardwired to interrupt power to the drive and lift motors immediately upon activation of any of the multiple E-stop buttons located on the AMR.

4.2 Motor Control System:

Drive Motors: Four dual-channel motor controllers, each controlling two drive motors. Communication will be over CANopen at 500 kbps with CAN IDs 0x601, 0x602, 0x603, and 0x604. Both Position Control Mode (2F 60 60 00 01 00 00 00) and Speed Control Mode (default) will be utilized. Position setting command: 23 7A 60 00 [4-byte position value] (little-endian). Speed setting command: 23 81 60 00 [4-byte speed value] (little-endian). Start movement sequence: 2B 40 60 00 2F 00 00 00 followed by 2B 40 60 00 3F 00 00 00. Stop movement command: 2B 40 60 00 0F 00 00 00.

Lift Motors: Four single-channel motor controllers communicating via CANopen at 500 kbps with assigned CAN IDs (to be determined). These will primarily operate in position control mode based on feedback from the integrated relative encoders.

Linear Actuator: Control will likely be through a dedicated motor driver (specifications to be determined) receiving PWM or digital signals from the Raspberry Pi 5 to control extension and retraction.

Electromagnet: Controlled by a relay module (rated for the electromagnet's current and voltage) connected to a digital output pin on the Raspberry Pi 5.

4.3 Encoder Interface:

Drive Motors: Relative encoder data (300,000 ticks/rotation) will be read from the dual motor controllers via CANopen.

Corner Angle Encoders: Absolute encoder data (20,480 ticks/360°, up to 3.2 turns) will be read from the encoder (CAN ID 0x188) via CANopen. Zeroing will be implemented in software using the SDO write command 2B 03 60 00 00 00.

Lift Motors: Relative encoder data will be read from the single motor controllers via CANopen.

4.4 Lift Motor Control: The Raspberry Pi 5 will implement a closed-loop control system for the lift motors, using the relative encoder feedback for precise positioning and the inclinometer data for leveling adjustments. The limit switches will be monitored as digital inputs to prevent movement beyond the physical limits of the actuators.

4.5 Linear Actuator Control: The Raspberry Pi 5 will control the linear actuator based on the module transfer sequence, extending to engage and retract to pull the module. Feedback on the actuator's position (if available on the chosen actuator) may be used for more precise control.

4.6 Electromagnet Control: The Raspberry Pi 5 will energize the electromagnet at the appropriate stage of the module transfer sequence and de-energize it once the module is securely positioned on the AMR or the EV platform.

4.7 Sensor Integration:

Lidars: The digital output signals from the two 270-degree lidars will be continuously monitored by the Raspberry Pi 5 to detect obstacles in the immediate vicinity of the AMR.

3D Lidar: Data from the 360-degree 3D lidar will be processed by the Raspberry Pi 5 for autonomous navigation and mapping of the Mission Station environment.

Laser Distance Sensors: The output signals from the laser distance sensors will be read by the Raspberry Pi 5 to determine the precise distance and angle to the target module or EV platform, enabling accurate alignment.

QR Code Scanners: The Raspberry Pi 5 will receive decoded QR code data from the scanners, allowing it to identify specific modules and loading positions.

Inclinometer: The RS-485 interface will be used to continuously read the pitch and roll angles of the AMR's platform.

Soft Bumpers: The digital input signals from the soft bumpers will be monitored by the Raspberry Pi 5 to detect physical contact with obstacles, triggering an immediate stop if necessary.

4.8 Communication System: The primary communication will be over a CAN bus (500 kbps) for motor control and absolute encoder data. RS-485 will be used for the inclinometer. Digital I/O pins on the Raspberry Pi 5 will interface with the limit switches and soft bumpers. The interface protocols for the 3D lidar, laser distance sensors, and QR code scanners will need to be defined based on the specific hardware chosen (likely Ethernet for the lidar, and potentially USB or serial for the others).

5. Software Design

5.1 Main Robot Program Architecture: The software on the Raspberry Pi 5 will be structured using a modular and event-driven architecture, potentially utilizing ROS (Robot Operating System) or a similar framework to manage the various subsystems and sensors. Real-time performance will be critical, especially for motion control and safety functions, potentially requiring the use of a real-time operating system (RTOS) or careful scheduling of tasks.

5.2 Motion Planning and Control:

Low-Level Control: PID control loops will be implemented for each drive motor to achieve precise speed and position control based on feedback from the relative encoders. The absolute encoders on the corner units will provide feedback for the overall orientation of the AMR and will be integrated into the odometry calculations.

High-Level Planning: The 360-degree 3D lidar data will be used for mapping the Mission Station environment and for path planning to navigate between module storage locations and the EV platform. Algorithms such as A

or similar will be employed.

Local Navigation and Obstacle Avoidance: Data from the two 270-degree lidars will be used for real-time obstacle detection and avoidance, potentially using techniques like the Vector Field Histogram (VFH) or similar reactive algorithms to modify the planned path.

Precise Positioning and Alignment: Data from the laser distance sensors and QR code scanners will be used for final, precise alignment with the module and the EV platform

before initiating the module transfer. Control algorithms will use this feedback to make fine adjustments to the AMR's position and orientation.

Control Modes: The software will implement both Position Control Mode (for accurate turns and calibrated movements) and Speed Control Mode (for general navigation) for the drive motors, as defined by the motor controller command reference. The start and stop movement sequences will be implemented as defined.

5.3 Sensor Data Processing:

Encoder Data: Raw encoder counts from both relative (drive and lift motors) and absolute (corner angles) encoders will be read and processed to determine velocity, position, and orientation of the AMR and its components.

Lidar Data: Point cloud data from the 270-degree lidars will be processed to identify obstacles, calculate their distance and size, and trigger avoidance maneuvers. The 3D lidar data will be used for map building, localization, and high-level path planning.

Laser Distance Sensor Data: Analog or digital readings from the laser distance sensors will be converted to distance measurements and used for precise alignment calculations.

QR Code Data: Decoded data from the QR code scanners will be used to identify the target module and its exact position relative to the AMR.

Inclinometer Data: Pitch and roll angles from the inclinometer will be read and used in a closed-loop control system to level the AMR's platform by adjusting the lift motors.

Limit Switch and Bumper Data: Digital inputs from the limit switches and soft bumpers will be monitored for safety and to detect the state of the lifting mechanism and potential collisions.

5.4 Lifting and Leveling Control: The software will continuously read the pitch and roll angles from the inclinometer. Based on these readings, it will calculate the necessary adjustments to the target positions of each of the four lift motors (using PID control with feedback from the lift motor relative encoders) to bring the platform to a level state. The limit switch inputs will be used to constrain the range of motion of the lift motors.

5.5 Module Transfer Control: This software module will orchestrate the sequence of actions for module transfer:

Approaching the module and achieving precise alignment using laser distance sensors and QR code data.

Lowering the platform to the correct height (based on sensor data).

Extending the linear actuator and energizing the electromagnet to engage with the module.

Retracting the linear actuator to pull the module onto the AMR platform.

Securing the module on the platform (potentially with additional mechanisms).

Reversing the process for loading onto the EV platform.

5.6 Safety System: The software will implement several safety features:

Emergency Stop (E-stop) Handling: Upon receiving a signal from any of the E-stop buttons (monitored via digital inputs), the software will immediately cease all motor движения and disable the electromagnet.

Timeout Mechanisms: Timers will be implemented for critical operations (e.g., motor movements, actuator extension) to prevent the AMR from being stuck in a potentially unsafe state.

Collision Avoidance: The software will continuously process lidar data to detect obstacles and autonomously plan and execute avoidance maneuvers. If a collision is imminent or detected by the soft bumpers, the AMR will stop immediately.

Battery Monitoring: (If battery voltage and current sensors are implemented), the software will monitor the battery status and potentially trigger warnings or safe shutdown procedures if critical levels are reached.

5.7 Communication Interface: The software will handle communication over the CAN bus using a suitable library (e.g., python-can). This will involve sending commands to the motor controllers (setting mode, speed, position, start/stop) and reading status and encoder data. It will also handle RS-485 communication with the inclinometer, reading digital inputs for lidars and bumpers, and manage the interface protocols for the 3D lidar, laser distance sensors, and QR code scanners based on the chosen hardware.

5.8 User Interface: The 7-inch touchscreen display will run a graphical user interface (GUI) built using a framework like PyQt or Tkinter. This interface will provide:

Real-time status information for all subsystems and sensors.

Manual control options for testing individual motors and actuators.

Visualization of sensor data (e.g., lidar scans, inclinometer readings).

Diagnostic tools and error logging.

Potentially a simplified interface for initiating autonomous module transfer sequences.

The web-based interface described earlier (joystick controls, QR code access, mobile-friendly design, automatic start, blue title #44a4db, duration control slider) will also be implemented and accessible over the network.

6. Testing and Validation

6.1 Component Testing: Individual testing of motors, encoders (results to be documented), limit switches, lidars, laser distance sensors, QR code scanners, inclinometer, linear actuator, and electromagnet will be performed to verify their functionality and specifications.

6.2 Subsystem Integration Testing: Integrated testing of the drive system (including motor control and encoder feedback), lifting mechanism (including motor control, encoder feedback, and leveling with the inclinometer), and module transfer mechanism (actuator and electromagnet control) will be conducted.

6.3 Full System Testing: Comprehensive testing of the complete AMR system performing the module swapping procedure (navigation to module, alignment, lifting, transfer, loading onto EV platform) will be conducted in a simulated and then a real-world environment at the Mission Station.

6.4 Performance Metrics: Key performance indicators (KPIs) will be defined and measured during testing, including:

Navigation accuracy and speed.

Module transfer cycle time.

Leveling accuracy and stability.

Alignment precision.

Obstacle avoidance reliability.

Battery life and power consumption.

6.5 Safety Testing: Thorough testing of all safety features, including the emergency stop system, limit switches, collision detection (bumpers and lidars), and software safety interlocks, will be performed under various conditions to ensure reliable and safe operation.

7. Future Considerations

7.1 Potential Enhancements:

Integration of visual servoing using cameras for more precise alignment.

Advanced path planning algorithms for optimized navigation.

Predictive maintenance capabilities based on sensor data.

Wireless charging capabilities.

Remote monitoring and control.

7.2 Scalability: Design considerations for scaling the AMR to handle different module sizes or weights in the future.

7.3 Maintenance: A detailed maintenance plan will be developed, including:

Preventative Maintenance Schedules: Regular checks and maintenance tasks for mechanical components (wheels, actuators, bearings), electrical systems (wiring, connections, motors), and sensors (calibration, cleaning).

Component Replacement Procedures: Clear guidelines and procedures for replacing wear items and faulty components.

Software Updates and Management: A process for deploying software updates and managing different software versions on the Raspberry Pi 5.

Troubleshooting Guides: Documentation to assist in diagnosing and resolving common issues.

Data Logging and Diagnostics: Implementation of robust data logging to track system performance and facilitate diagnostics.

8. Appendices

8.1 Bill of Materials (BOM): A comprehensive list of all hardware components used in the AMR, including manufacturer part numbers and specifications.

8.2 Electrical Schematics: Detailed wiring diagrams for the power distribution, motor control, and sensor integration.

8.3 Software Architecture Diagrams: Visual representations of the software modules and their interactions on the Raspberry Pi 5.

8.4 Communication Protocols: Detailed specifications for CANopen, RS-485, and other communication protocols used.

8.5 Safety Analysis: A thorough risk assessment and safety analysis of the AMR's operation, including potential hazards and mitigation strategies.

8.6 Test Procedures: Detailed procedures for component, subsystem, and full system testing.

This concludes the current version of the Autonomous Mobile Robot (AMR) Design Document based on the information provided. Further details will be added to each section as the design and development progress.