Université Mohammed 6 Polytechnique
Digital Innovation Center of Excellence (DICE)

Specifications document

11 Feb 2025

Interactive Educational Robot

for Teaching Programming

Prepared By: Mohamed Ayman Ouchker

Supervised By: Mr. Said Amharech

Version: 2.0

Table of contents

1. Introduction

This document outlines the requirements for the development and design of an interactive educational robot aimed at teaching programming and basic robotics concepts to children. The purpose of this requirements document is to provide a clear, comprehensive, and agreed-upon description of the system's functionalities, performance targets, and technical constraints. It will serve as the roadmap for both the development and evaluation phases of the project.

2. Context and Pedagogical Objectives

Project Context:
The project is designed within an educational framework where the goal is to introduce children (aged 6 and above) to the fundamentals of programming and robotics. This interactive robot will be used in classroom settings and at home to stimulate logical thinking, creativity, and problem-solving skills. The robot's design will focus on simplicity and intuitiveness, ensuring it is accessible for young learners.

Overview:

- Build a simple, custom-designed chassis 3D printed.
- Develop a cross-platform Flutter app where kids can visually build a sequence of commands to define the robot's path and progress through increasingly complex programming levels.

Objectives:

- Teach foundational programming concepts (sequencing, debugging, modularity) in a fun, interactive way.
- Combine mechanical design, electronics, firmware development, and app programming into one interdisciplinary project.
- Provide an engaging tool to introduce children to coding and autonomous robotics.

3. Functional Requirements

The system shall meet the following functional requirements:

1. **Manual Command Execution:**
    - The robot shall respond to basic movement commands (Forward, Backward, Left, Right, Stop) issued from the mobile app.
    - The robot shall support a gripper command (Open/Close) to manipulate objects.
2. **Sequenced Command Processing:**
    - The robot's firmware shall parse sequenced commands received via Bluetooth or Wi-Fi from the app.
    - It shall execute sequences of commands (e.g., "F2000" for moving forward for 2000 ms) to enable path planning.
3. **Sensor Integration and Obstacle Detection:**
    - The robot shall use an ultrasonic sensor to measure distance to obstacles.
    - The robot shall integrate data from an IMU for orientation and acceleration to improve navigation.
    - If an obstacle is detected within a defined threshold (e.g., 15 cm), the robot shall execute an avoidance maneuver.
4. **Autonomous Navigation (Auto Mode):**
    - The robot shall support an autonomous mode where it continuously moves forward, checks sensor data, and performs obstacle avoidance maneuvers.
    - Autonomous navigation shall continue until a "stop" command is received from the app.
5. **Feedback and Telemetry:**
    - The robot shall continuously monitor its battery voltage via an analog input (using a voltage divider) and calculate the remaining battery percentage.
    - Telemetry data (battery level, sensor readings, movement status) shall be sent back to the mobile app for display.
6. **Motor Control with Feedback:**
    - The robot shall use DC motors with encoders to provide precise feedback.

    - The firmware shall implement PID (or PD) control loops using encoder data for accurate speed and direction adjustments.

7. **Communication:**
    - The system shall establish reliable communication via Bluetooth and/or Wi-Fi between the robot (ESP32) and the mobile app.
    - The app shall send commands (formatted as JSON, delimited strings, or similar) to the robot, and the robot shall respond with status updates.
8. **Mobile App Programming Interface:**
    - The Flutter-based app shall provide a drag-and-drop, block-based coding environment for creating robot control programs.
    - The app shall organize programming tasks into progressive levels:
        - **Level 1:** Basic movement.
        - **Level 2:** Path planning (sequencing basic commands).
        - **Level 3:** Sensor integration (conditional blocks using ultrasonic sensor data).
        - **Level 4:** Manipulation (gripper control).
        - **Level 5:** Auto mode (continuous autonomous navigation).
9. **Firmware Update (Optional):**
    - The system may support over-the-air (OTA) firmware updates to simplify maintenance and upgrades.

---

4. Non-Functional Requirements

In addition to the functional requirements, the system must satisfy the following quality attributes:

1. **Safety and Child-Friendliness:**
    - The chassis design shall use durable, lightweight, and safe materials with rounded edges.
    - All components shall be securely mounted to prevent accidental damage or injury.
2. **Robustness and Durability:**
    - The hardware must withstand repeated use in a classroom or home environment.

    - The system should be designed for easy assembly, maintenance, and repair.
3. **User Experience:**
    - The Flutter app shall have an intuitive, visually engaging interface suitable for children (ages 7–12).
    - The block-based programming environment must be simple to use and provide clear visual feedback.
4. **Performance:**
    - Sensor readings and motor commands shall be processed in real time with minimal latency.
    - The autonomous navigation algorithm should respond quickly to obstacles to ensure smooth operation.
5. **Scalability and Modularity:**
    - The design shall be modular, allowing for the future integration of additional sensors or hardware features.
    - The software architecture (both firmware and app) must be maintainable and extensible.
6. **Cross-Platform Compatibility:**
    - The mobile app must run on Android, iOS, web, and desktop platforms using Flutter.
    - Communication protocols between the robot and the app should work seamlessly across all supported platforms.
7. **Cost-Effectiveness:**
    - Component selection should balance advanced functionality with overall cost, ensuring the project remains affordable for educational institutions.
8. **Documentation and Support:**
    - The project must include comprehensive documentation, including assembly instructions, software guides, and troubleshooting FAQs.
    - The design should be accessible to educators and children, facilitating an easy learning curve.

5. Hardware Specifications

Control & Processing:

- ESP32 Development Board:
  - Dual-core processor with integrated Wi-Fi and Bluetooth for robust connectivity.
  - Supports advanced algorithms (sensor fusion, PID control) while remaining accessible for educational purposes.

Custom Chassis:

- Chassis Design:
  - Custom-designed using CAD software (e.g., Fusion 360) to create a child-safe, visually engaging, and durable structure.
  - Fabricated using high-quality materials (aluminum, laser-cut acrylic, or high-strength polymers) for robustness and longevity.

Drive System:

- DC Motors with Encoders (×2):
  - High-quality motors equipped with encoders for precise odometry and closed-loop control.
  - Provide accurate feedback for implementing PID control and advanced movement algorithms.
- Motor Driver:
  - A robust dual H-bridge driver (e.g., Sabertooth, VNH5019) capable of handling higher currents and delivering fine speed/direction control.

Sensor Suit:

- Ultrasonic Sensor:
  - HC-SR04 or similar for short-range obstacle detection.
- Inertial Measurement Unit (IMU):
  - MPU9250 or BNO055 for accurate orientation, acceleration, and rotational data.

Manipulation Module:

- Gripper Mechanism:

- Controlled by a standard servo to open/close for object manipulation tasks.

Manipulation Module:

- Multi-Cell LiPo Battery Pack:
  - Selected based on voltage requirements (e.g., 2S or 3S configuration) to ensure adequate power for the ESP32 and motors.
- Battery Management System (BMS):
  - For safe charging/discharging, monitoring, and protection.
- Integrated Charging Module:
  - Dedicated charger circuit (or wireless charging option) to allow convenient recharging.
- Voltage Regulation:
  - Appropriate boost/step-down converters to deliver stable 3.3V for the ESP32 and 5V for peripherals.
- Voltage Divider Components:
  - Resistors (e.g., 100kΩ and 47kΩ) to safely monitor battery voltage through an analog input.

---

6. Software Specifications

Firmware (ESP32)

- **Development Environment:**
  - Written in C/C++ using the Arduino framework (or PlatformIO) on the ESP32.
- **Core Functions:**
  - **Motor Control:** Implements precise control using encoder feedback and PID algorithms.
  - **Sensor Integration:** Reads data from ultrasonic sensors and the IMU, enabling obstacle detection and improved orientation.
  - **Autonomous Navigation:** Features an “Auto Mode” routine that continuously drives forward with real-time obstacle avoidance until a stop command is issued.
  - **Communication:** Utilizes Bluetooth and/or Wi-Fi for receiving command sequences from the Flutter app and sending telemetry (e.g., battery level, sensor data).

3.2 Mobile/Desktop/Web App (Flutter)

- **Development Environment:**
  - Built using the Flutter SDK (Dart language) to ensure cross-platform compatibility across Android, iOS, desktop, and web.
- **User Interface:**
  - Block-Based Coding Interface:
- **Connectivity & Data Exchange:**
  - Implements Bluetooth and/or Wi-Fi to send command sequences (formatted as JSON, delimited strings, or similar) to the ESP32.
  - Receives telemetry data (battery level, sensor status) from the robot.
- **Gamification & Feedback:**
  - Progress tracking, badges, and real-time sensor data visualization to enhance the learning experience.

---

7. Rationale for Material Choices

- **Advanced Processing (ESP32):** Provides the necessary processing power and connectivity (Bluetooth/Wi-Fi) for advanced sensor fusion and autonomous navigation while being cost-effective and widely supported.
- **High-Quality Motors with Encoders:** Ensure precise control and reliable feedback, which are essential for implementing advanced control algorithms and accurate odometry.
- **Custom Chassis:** A bespoke design allows for optimized sensor placement, modularity, and a robust, child-friendly structure.
- **Enhanced Sensor Suite:** Incorporating ultrasonic sensors and an IMU allows for real-time obstacle detection and improved navigation, bridging the gap between simple educational projects and real-world robotics.
- **Smart Power System:** A multi-cell LiPo battery with a BMS and integrated charging circuit ensures longer runtime, safety, and convenient recharging.
- **Cross-Platform Flutter App:** Enables the educational programming interface to be accessible on all major platforms, maximizing reach and engagement for children.

---

8. Programming Levels & Features

Level 1: Basic Movement

- **Objective:** Introduce simple movement commands.
- **Blocks Available:** Forward, Backward, Left, Right, Stop.
- **Focus:** Understanding sequencing and immediate motor responses.

Level 2: Path Planning

- **Objective:** Teach kids to design and sequence multiple moves to form a complete path.
- **Blocks Available:** Basic movement blocks arranged to form a route.
- **Focus:** Spatial reasoning, planning, and visualizing the robot's path.

Level 3: Sensor Integration (Ultrasonic Sensor)

- **Objective:** Incorporate obstacle detection.
- **New Blocks:** Sensor blocks that read distance values; conditional blocks (if/else) to handle obstacle detection.
- **Focus:** Understanding how sensor input influences decisions and integrating simple logic.

Level 4: Manipulation (Gripper Integration)

- **Objective:** Expand functionality to include object manipulation.
- **New Blocks:** Gripper controls (Open, Close).
- **Focus:** Combining movement, sensor input, and manipulation to perform tasks (e.g., picking up and delivering objects).

Level 5: Auto Mode (Autonomous Navigation)

- **Objective:** Enable the robot to memorize and autonomously navigate a path without direct sequential commands.
- **New Blocks:**
  – **Auto Mode Block:** Initiates an autonomous routine using preprogrammed navigation algorithms.
  – **Path Memory & Correction Blocks:** Allow the robot to store a learned path and adjust based on real-time sensor data (from ultrasonic sensors or line sensors).
- **Features:**

– The robot uses previously learned commands and sensor feedback to autonomously follow a path and avoid obstacles.
– Introduces more advanced algorithms like PID control or even simple path memorization techniques inspired by research.

- **Focus:**

  – Teaching autonomous decision-making and error correction.

  – Allowing the robot to operate with minimal intervention, enhancing its real-world application potential.
