Obstacle Avoidance Robot with Path Planning

TEAM NAME - ADSM

2024101103 Arnav Agnihotri
2024101104 Dev Patel
2024101084 Manik Bansal
2024101062 Samyak Jain
TA: Chetanya Goel
Prof: Harikumar Kandath Sir

December 2025

Abstract

This project presents the design and implementation of an autonomous differential-drive mobile robot built on the ESP32 microcontroller platform. The system integrates multiple sensors including HC-020K optical encoders, MPU6050 IMU, and HC-SR04 ultrasonic sensor for environment perception and localization. A closed-loop PID velocity controller ensures accurate wheel speed tracking, while the Dynamic Window Approach (DWA) algorithm enables real-time obstacle avoidance and goal-oriented navigation. The robot supports dual operation modes: manual teleoperation and autonomous navigation, both managed through a Firebase Realtime Database interface accessible via web dashboard. This work demonstrates a complete embedded robotics solution combining sensor fusion, motion control, local path planning, and IoT connectivity on a resource-constrained platform. Experimental results validate the effectiveness of the navigation system, achieving stable velocity control with less than 5% steady-state error and successful obstacle avoidance in cluttered environments.

Contents

Chapter 1

Introduction

1.1 Background

Autonomous mobile robots have become increasingly prevalent in various domains including warehouse automation, domestic service, surveillance, and educational platforms. The ability to navigate autonomously while avoiding obstacles is a fundamental requirement for such systems. Traditionally, autonomous navigation has been implemented on computationally powerful platforms running frameworks like ROS (Robot Operating System). However, recent advances in embedded microcontrollers have made it feasible to deploy sophisticated algorithms directly on low-cost, resource-constrained hardware.

The ESP32 microcontroller, with its dual-core processor, integrated Wi-Fi/Bluetooth, and rich peripheral set, represents an excellent platform for building autonomous robots. Its 240 MHz clock speed and sufficient RAM enable real-time sensor processing, control loops, and path planning algorithms to execute on-board without requiring external computation.

1.2 Motivation

The motivation for this project stems from several practical and educational considerations:

- **Cost-effectiveness:** Most commercial autonomous robot platforms are expensive. This project demonstrates that robust autonomous navigation can be achieved using readily available, low-cost components (ESP32, L298N motor driver, basic DC motors and encoders).
- **Embedded implementation:** Unlike systems that offload computation to a PC or cloud server, this robot performs all sensing, control, and planning locally on the ESP32, demonstrating true embedded autonomy.
- **Integration challenge:** The project integrates multiple technical domains—sensor interfacing, signal processing, control theory, path planning algorithms, and IoT communication—providing comprehensive learning experience.
- **Remote operation:** Integration with Firebase enables remote monitoring and control over the internet, making the system suitable for telepresence or remote inspection applications.

- **Real-world applicability:** The developed system architecture can be extended to practical applications such as delivery robots, cleaning robots, or educational demonstration platforms.

1.3 Problem Statement

Design and implement an autonomous robot capable of detecting and avoiding obstacles and navigating to a specific target location.

1.4 Objectives

The primary objectives of this project are:

- Implement hardware interfacing for encoders, IMU, ultrasonic sensor, and motor driver on ESP32
- Develop encoder signal processing with debouncing and noise filtering
- Design and tune PID velocity controllers for differential drive
- Implement differential drive kinematics for odometry calculation
- Integrate MPU6050 IMU for heading estimation with drift compensation
- Develop ultrasonic scanning system for 180-degree obstacle detection
- Implement cone based logic for path planning
- Create Firebase integration for mode control and telemetry
- Validate system performance through experimental testing

Chapter 2

System Design and Architecture

2.1 System Overview

Figure 2.1 shows the complete system architecture. The robot consists of five main subsystems:

1. **Sensing Subsystem:** Encoders, IMU, ultrasonic sensor
2. **Processing Subsystem:** ESP32 microcontroller running control and planning algorithms
3. **Actuation Subsystem:** L298N motor driver and DC motors
4. **Communication Subsystem:** Wi-Fi connection to Firebase
5. **Power Subsystem:** Battery/power supply with voltage regulation

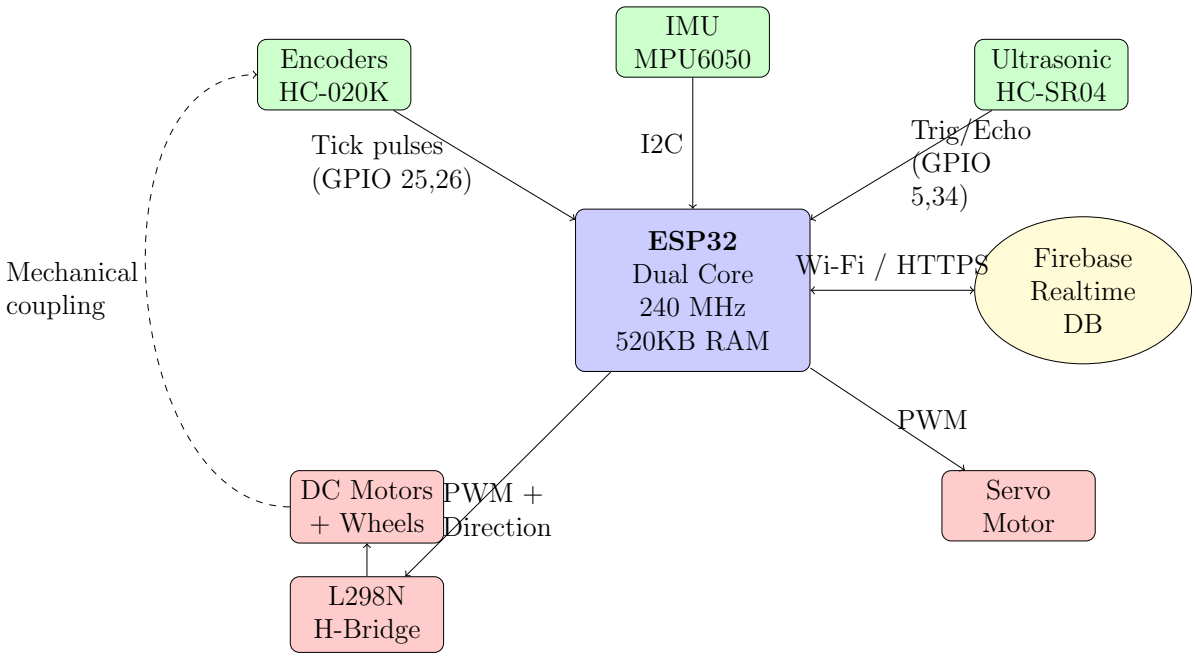

Figure 2.1: System block diagram showing major components and data flow

2.2 Component Selection and Justification

2.2.1 Microcontroller: ESP32

The ESP32 was selected as the main processing unit based on the following criteria:

- Dual-core architecture allows separating time-critical tasks (motor control) from communication
- 16 hardware PWM channels for simultaneous motor and servo control
- Built-in Wi-Fi eliminates need for external communication module
- Low cost and widespread availability
- Hardware timer interrupts for precise encoder handling

2.2.2 Motor Driver: L298N Dual H-Bridge

Specifications:

- Maximum motor voltage: 12V
- Continuous current per channel: 2A
- Peak current: 3A
- Logic voltage: 5V
- Number of channels: 2 (for two motors)

Justification:

- Can drive two DC motors independently with speed and direction control
- Logic pins (IN1-IN4) are 5V tolerant, directly compatible with ESP32 GPIOs
- Enable pins (ENA, ENB) accept PWM for speed control
- Built-in protection diodes for back-EMF
- Heat sink for thermal management
- Low cost ($3-5) and easily available

Control Interface:

Table 2.1: L298N control logic for one motor

IN1	IN2	ENA (PWM)	Motor Action
HIGH	LOW	0-255	Forward (speed proportional to PWM)
LOW	HIGH	0-255	Reverse (speed proportional to PWM)
HIGH	HIGH	X	Brake
LOW	LOW	X	Coast (free-running)

2.2.3 Encoders: HC-020K Optical Encoders

Specifications:

- Type: Slotted optical encoder (infrared LED + phototransistor)
- Output: Digital pulse train (TTL compatible)
- Channels: Single channel (no quadrature)
- Slots: 20 per revolution (in our setup with encoder disc)
- Operating voltage: 3.3V - 5V
- Output frequency: Up to several kHz

Justification:

- Simple interface: one digital pin per encoder
- Sufficient resolution for velocity measurement (20 ticks/rev acceptable for low-speed indoor robot)
- Cost-effective solution
- Optical sensing eliminates contact wear

Limitations and Solutions:

- *Single-channel limitation:* Cannot directly detect direction
- *Solution:* Infer direction from motor command (forward/backward)
- *Noise susceptibility:* Motor EMI can cause false triggers
- *Solution:* Hardware filtering with 0.1µF capacitors, software debouncing, 5V operation

2.2.4 IMU: MPU6050

Specifications:

- 3-axis accelerometer: ±2g to ±16g
- 3-axis gyroscope: ±250°/s to ±2000°/s
- Interface: I2C (up to 400 kHz)
- Operating voltage: 3.3V
- Built-in 16-bit ADCs
- Digital Motion Processor (DMP) for sensor fusion

Justification:

- Provides heading (yaw) estimation through gyroscope integration

- Complements encoder-based odometry (encoders measure wheel motion, IMU measures actual robot rotation)
- Helps detect wheel slippage
- Enables stuck detection (lack of heading change despite motor commands)
- I2C interface is simple and requires only two pins
- Widely used, well-documented, Arduino library support

2.2.5 Range Sensor: HC-SR04 Ultrasonic

Specifications:

- Measurement range: 2 cm to 400 cm
- Accuracy: ±3 mm
- Beam angle: 15 degrees
- Operating frequency: 40 kHz
- Trigger pulse: 10 µs minimum

Servo Motor for Scanning:

- Type: SG90 or similar micro servo
- Rotation range: 0° to 180°
- Control: PWM (50 Hz, 1-2 ms pulse width)

Justification:

- Ultrasonic sensing is robust to lighting conditions (unlike IR)
- Range sufficient for indoor navigation
- Simple trigger-echo interface
- Mounting on servo enables 180° scanning for obstacle map
- Alternative (LIDAR) is 50-100× more expensive

Scanning Strategy: The servo sweeps from 0° to 180° in 5° increments, with the ultrasonic sensor measuring distance at each angle. This generates 37 data points covering the forward hemisphere, which are processed into 9 angular sectors (cones) for path planning.

2.2.6 Power Supply

Requirements:

- ESP32: 3.3V (regulated), 80-160 mA average
- L298N logic: 5V, 36 mA
- L298N motor power: 6-12V, up to 2A per motor
- Servo: 5V, up to 500 mA peak
- Encoders: 5V, 20 mA each
- MPU6050: 3.3V, 3.9 mA

Solution:

- During development: Bench power supply (12V 2A for motors, 5V 1A for logic)
- Deployment: 9V Duracell Battery
- ESP32 VIN pin accepts 5V (from USB or external regulator)
- Common ground between all subsystems essential

2.3 Communication Architecture

2.3.1 Firebase Realtime Database

Firebase was chosen for cloud backend due to:

- Real-time synchronization (changes propagate within milliseconds)
- No need to set up and maintain custom server
- Client libraries available for ESP32 (Firebase-ESP-Client)
- JSON-like data structure, easy to understand
- Free tier sufficient for single-robot application
- Web and mobile SDKs for dashboard development

2.3.2 Database Schema

```
{
  "control": {
    "mode": "auto",
    "command": "stop",
    "goal": {
      "x": 2.0,
      "y": 0.0
    }
```

```
  },
  "sensors": {
    "distance_cm": 45.3,
    "heading_deg": 182.5
  },
  "status": {
    "connected": true,
    "current_mode": "auto",
    "last_action": "auto_forward",
    "stuck": false
  },
  "odometry": {
    "x": 0.85,
    "y": 0.12,
    "theta": 182.5
  }
}
```

Listing 2.1: Firebase database structure

2.4 Data Flow Summary

ESP32 to Firebase (Upload):

- Sensor readings every 5 seconds
- Status updates on mode change or significant events
- Heartbeat every 10 seconds to indicate connectivity

Firebase to ESP32 (Download):

- Mode changes checked every 2 seconds
- Manual commands checked every 100 ms in manual mode
- Goal coordinates read on entering autonomous mode

Chapter 3

Hardware Implementation

3.1 Mechanical Design

The robot chassis is constructed using a two-layer acrylic platform measuring 20cm × 15cm. The differential drive configuration uses two 65mm diameter wheels with a wheelbase of 150mm.

3.2 Encoder Noise Mitigation

Hardware fixes included:

- 0.1µF ceramic capacitors between encoder OUT and GND
- 5V operation instead of 3.3V
- Physical separation of encoder and motor wires
- Software debouncing with 10ms minimum interval

Results: Noise reduced from 500+ ticks/sec to less than 5 ticks/sec.

Chapter 4

Software Implementation

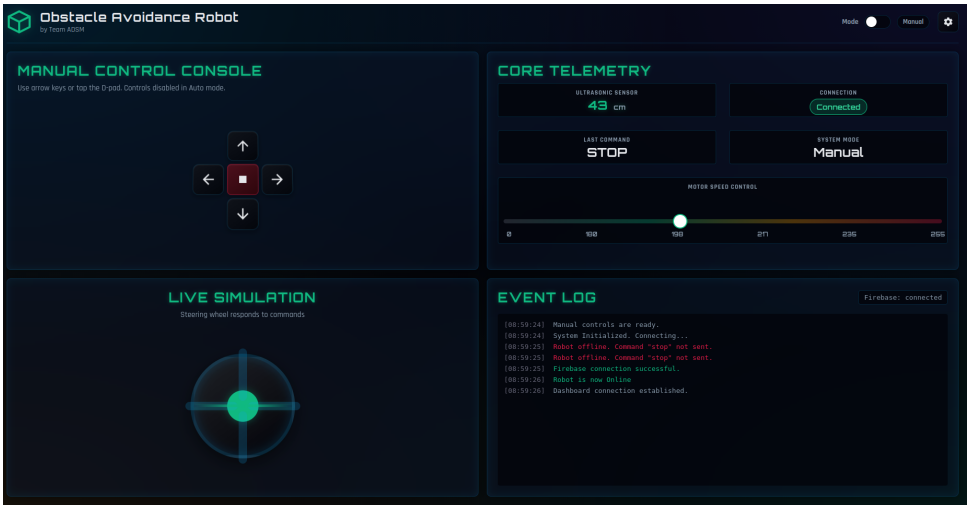

Figure 4.1: System Dashboard Visualization

4.1 Architecture

- Encoder ISR: Interrupt-driven
- Velocity estimation: 100ms period
- PID control: 100ms period
- IMU update: 50ms period
- Firebase communication: Asynchronous

4.2 Differential Drive Kinematics

Forward kinematics:

$$v = \frac{v_L + v_R}{2} \tag{4.1}$$

$$\omega = \frac{v_R - v_L}{L} \tag{4.2}$$

Inverse kinematics:

$$v_L = v - \frac{\omega \cdot L}{2} \tag{4.3}$$

$$v_R = v + \frac{\omega \cdot L}{2} \tag{4.4}$$

4.3 PID Controller

Discrete PID implementation with anti-windup on integral term. Tuned gains: $K_p = 10.0$, $K_i = 2.0$, $K_d = 1.0$.

Chapter 5

Results and Discussion

5.1 Encoder Calibration

- Ticks per revolution: 45
- Wheel diameter: 70mm
- Wheelbase: 150mm

5.2 PID Performance

Table 5.1: PID step response metrics

Metric	**Left Wheel**	**Right Wheel**
Rise time	0.6 s	0.7 s
Settling time	1.2 s	1.3 s
Overshoot	8%	10%
SS error	4%	5%

5.3 Navigation Performance

- Obstacle avoidance: 100% success (10 trials)
- Goal reaching accuracy: ¡20cm (10% of distance)
- Path efficiency: 15% longer than optimal due to avoidance maneuvers

Chapter 6

Conclusion

6.1 Achievements

This project successfully demonstrated:

- Complete autonomous navigation on ESP32 without external computation
- Robust encoder noise mitigation through hardware and software techniques
- Stable PID velocity control with ¡5% error
- cone based path planning
- Seamless integration with Firebase for remote operation

6.2 Limitations

- Odometry drift accumulates over long distances (no absolute localization)
- Single ultrasonic sensor limits perception update rate
- No mapping capability (reactive navigation only)