Overview

This project involved the design and construction of a remote-controlled (RC) car at 1:32 scale, focusing on electrical integration, microcontroller programming, and compact power management. The goal was to create a functional system capable of steering and motor control using an Xbox controller as a wireless interface.

Design and Planning

The project began with a series of hand-drawn sketches detailing the car's internal layout, dimensions, and component placement. This initial planning stage helped determine spatial constraints and guided mechanical integration decisions.
Early prototypes were developed using an Arduino Uno and large motor drivers to understand motor control fundamentals such as reversing polarity and varying speed through pulse-width modulation (PWM). These prototypes served as proof-of-concept builds before transitioning to a more compact and efficient design.

Hardware and Components

- **Microcontrollers:** Arduino Uno (initial testing), ESP-32 (final build)
- **Motor Driver:** L298N TB66 (initial testing), Motor Driver (final build)
- **Power Management:** Buck converter, capacitors, and flyback diodes
- **Motors:** DC motor (driver) and servo motor (steering)
- **Control Interface:** Xbox controller via Bluetooth connection
- **Lighting:** Functional brake lights and high beams (LEDS)

The final configuration utilized the ESP-32 for its Bluetooth capability and smaller footprint, reducing overall space usage while maintaining functionality.

System Integration and Challenges

A major challenge in this project was creating a **single power system** that could drive both motors and the microcontroller safely. Research into motor current draw and voltage requirements led to the use of a buck converter to regulate voltage and protect sensitive components.

Capacitors were used to smooth voltage spikes, and flyback diodes were integrated to prevent back EMF from damaging the microcontroller.

After transitioning to the ESP-32, PWM interference became an issue; the board could only send one stable PWM signal without crosstalk. To resolve this within budget constraints, I optimized the control logic to prioritize steering and direction rather than speed variation. This limitation ultimately improved my understanding of signal management and hardware communication.

Programming and Control

All control logic was written in **C++** using the Arduino IDE.
The program handled:

- Servo angle control via PWM for steering
- Direction control for the DC motor through the TB66 driver
- Bluetooth input mapping for Xbox controller joystick and buttons

This stage involved iterative testing and debugging, emphasizing code efficiency, PWM timing, and microcontroller-to-driver communication.

Construction and Assembly

Mechanical integration required drilling and modifying the car chassis to fit all components securely. Non-conductive adhesive was used to mount electrical hardware safely, and all soldering was planned ahead to maximize internal space and maintain structural balance.

The final assembly achieved:

- Fully wireless Bluetooth control up to ~30 feet
- Responsive steering via servo motor
- Reversible drive motor control
- Integrated headlights and brake lights

Results and Reflection

The completed RC vehicle successfully demonstrated the integration of electrical, mechanical, and programming concepts. While limited by PWM channel interference, the system achieved full motion control, stable power regulation, and reliable wireless communication.

Skills and Knowledge Gained

- **Microcontroller Programming:** Writing and debugging C++ code for Arduino and ESP-32 platforms
- **Circuit Design & Power Management:** Designing safe single-source power systems using regulators, capacitors, and protection diodes
- **Mechanical Integration:** Planning and fitting components within spatial and structural constraints
- **Iterative Design Process:** Prototyping, testing, identifying system limitations, and refining solutions
- **Bluetooth Communication:** Implementing wireless control interfaces between hardware and external devices

---

Future Improvements

- Utilize an ESP-32 variant with additional PWM channels to enable speed control.
- Design a custom PCB to reduce wiring complexity.
- Integrate a rechargeable Li-ion battery system with power monitoring.