Development and Implementation of a Smartphone-Controlled Four-Wheel Drive Robotic Project Using Bluetooth Communication Protocol

Project Report

Submitted by:
[Zohaib Hassan AND Zuhaib Khan]
Student ID: [133-0028-25]

Submitted to:
[Sir Irfan Younas]
Department of Computer Systems Engineering

[Sukkur IBA University]

November 30, 2025

Abstract

This investigation presents the systematic development of a four-wheel drive robotic vehicle platform incorporating wireless control mechanisms through Bluetooth technology. The research addresses the fundamental challenge of establishing reliable human-machine interaction within embedded systems. Through careful integration of microcontroller architecture, wireless communication modules, and motor control circuitry, a responsive remotely-operated vehicle has been realised. The findings demonstrate that serial communication protocols, when properly configured with appropriate motor driving electronics, facilitate effective real-time control of mechanical actuators. This work contributes to the broader understanding of how command-response paradigms function within resource-constrained computing environments, particularly relevant for educational robotics applications.

Contents

1 Introduction

The proliferation of mobile computing devices has fundamentally transformed approaches to remote system control. Within educational and research contexts, there exists substantial interest in developing platforms that bridge theoretical knowledge with practical implementation skills. This project was undertaken to explore how contemporary wireless technologies could be integrated with traditional embedded systems to produce a functional robotic platform.

1.1 Background and Context

Robotic systems have traditionally fallen into two primary categories: autonomous vehicles that rely upon sensor feedback for decision-making, and remotely-operated platforms requiring continuous human oversight. Whilst autonomous systems represent significant technological advancement, remotely-operated vehicles offer distinct pedagogical advantages, particularly regarding understanding of communication protocols and real-time control systems.

The emergence of ubiquitous Bluetooth-enabled smartphones presents opportunities for intuitive human-machine interfaces without requiring specialised control hardware. This approach reduces barriers to entry whilst maintaining technical rigour in system design and implementation.

1.2 Project Aims and Objectives

The principal aim was to design, construct, and programme a four-wheel drive robotic vehicle controllable through smartphone interface via Bluetooth wireless communication. Specific objectives included:

- Integration of microcontroller architecture with wireless communication modules
- Implementation of motor control circuitry capable of handling required current loads
- Development of efficient command parsing algorithms for real-time response
- Validation of system performance through systematic testing procedures

1.3 Report Structure

Following this introduction, Section 2 examines the theoretical foundations underpinning the system architecture. Section 3 details the hardware components and their interconnections. Section 4 presents the software implementation, including algorithmic approaches. Section 5 describes testing methodologies and results. Finally, Section 6 offers conclusions and recommendations for future work.

2 Theoretical Foundation and System Architecture

2.1 Communication Paradigms in Embedded Systems

Traditional approaches to robotic control have emphasised sensor-driven autonomy, wherein the system continuously evaluates environmental data to make navigational decisions.

However, an alternative paradigm exists: the command-response model, which places decision-making authority with external operators whilst the embedded system executes received instructions.

This architectural choice necessitates reliable bidirectional communication channels. Serial communication protocols, particularly Universal Asynchronous Receiver-Transmitter (UART), have proven effective for such applications due to their simplicity and widespread hardware support.

2.2 Bluetooth Technology Overview

Bluetooth represents a short-range wireless communication standard operating within the 2.4 GHz ISM band. For embedded applications, simplified modules implementing the Serial Port Profile (SPP) enable straightforward integration with microcontroller systems. These modules effectively translate Bluetooth radio signals into serial data streams, allowing microcontrollers to process wireless communications using familiar UART interfaces.

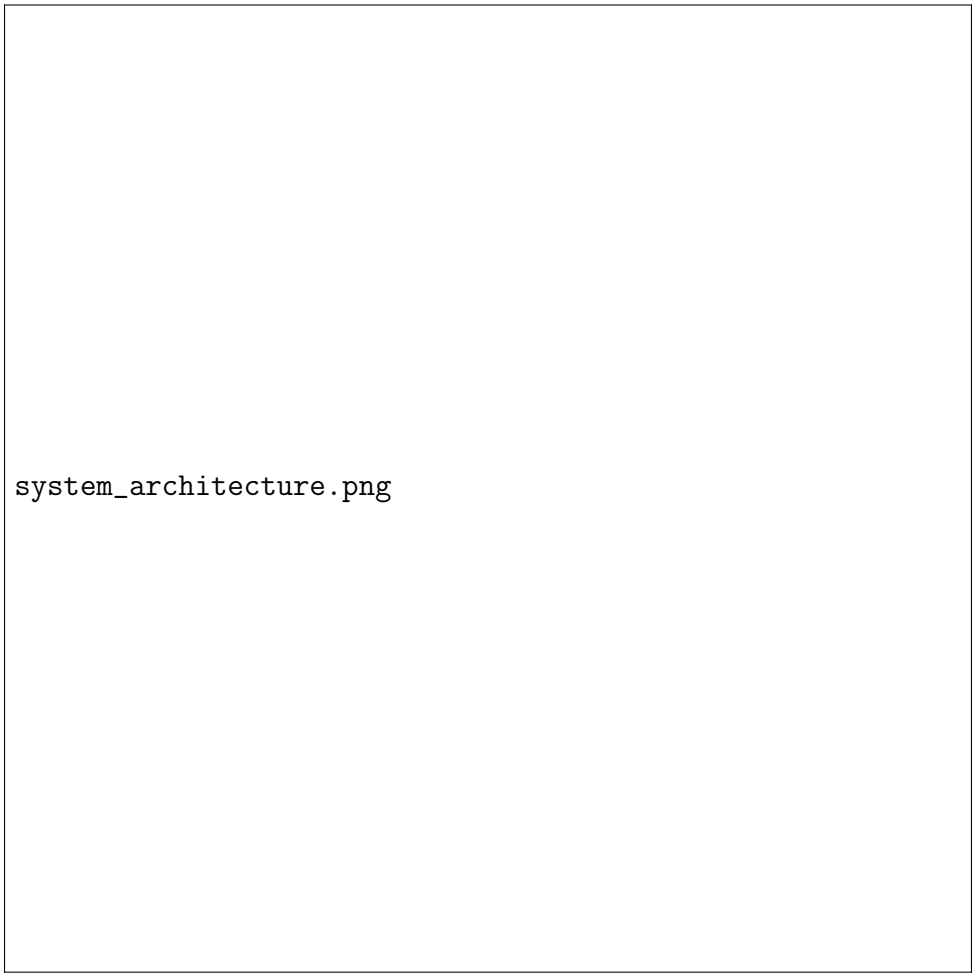

Figure 1: System architecture illustrating communication flow from user interface to mechanical actuators

2.3 Motor Control Principles

Direct current motors require current levels exceeding microcontroller output capabilities. Consequently, intermediate driver circuits become necessary. H-bridge configurations al-

low bidirectional current flow, enabling both forward and reverse motor operation. Pulse Width Modulation (PWM) techniques facilitate speed regulation by varying the effective voltage applied to motor terminals.

3 Hardware Components and Integration

This section examines the physical components comprising the robotic platform and their interconnections.

3.1 Microcontroller Selection and Configuration

The Arduino Uno, based upon the ATmega328P microcontroller, serves as the central processing unit. This selection was justified by several factors: widespread availability, comprehensive development environment support, and adequate processing capability for the intended application.

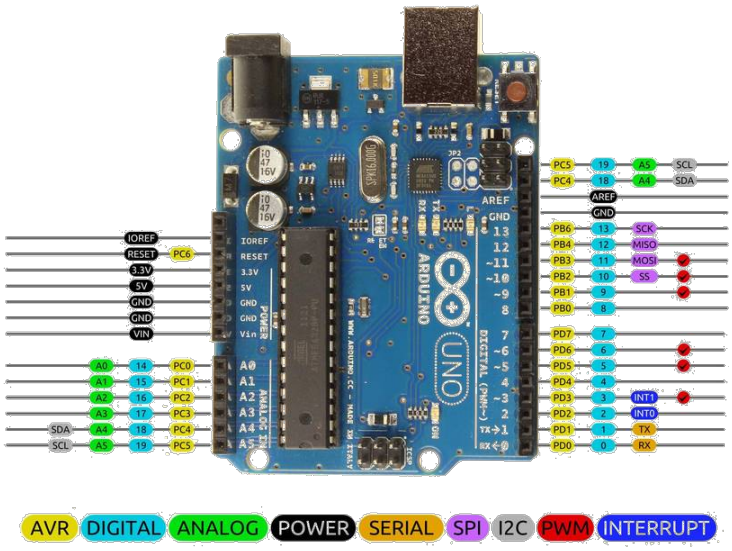

Figure 2: Arduino Uno microcontroller board highlighting relevant pin assignments

Key specifications include:

- Operating voltage: 5V
- Digital I/O pins: 14 (six capable of PWM output)
- Clock speed: 16 MHz
- SRAM: 2 KB

3.2 Wireless Communication Module

The HC-05 Bluetooth module facilitates wireless connectivity between smartphone and microcontroller. This module implements Bluetooth 2.0 with SPP, presenting a transparent serial interface to the host microcontroller.

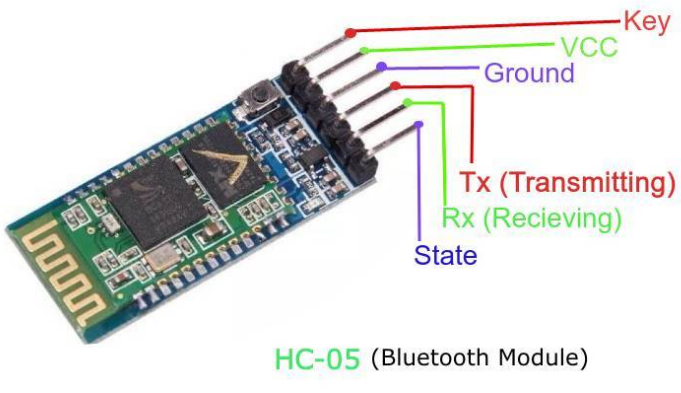

Figure 3: HC-05 Bluetooth module showing critical connection points

Connection configuration:

- VCC: Connected to 5V supply
- GND: Common ground reference
- TXD: Transmit data to Arduino RX (Pin 0)
- RXD: Receive data from Arduino TX (Pin 1)

A critical operational consideration: the module must be electrically disconnected during code upload procedures to prevent serial communication conflicts.

3.3 Motor Driver Circuitry

The L298N dual H-bridge motor driver provides the necessary current amplification for motor operation. This integrated circuit can handle continuous currents up to 2A per channel, sufficient for typical DC gear motors employed in educational robotics.

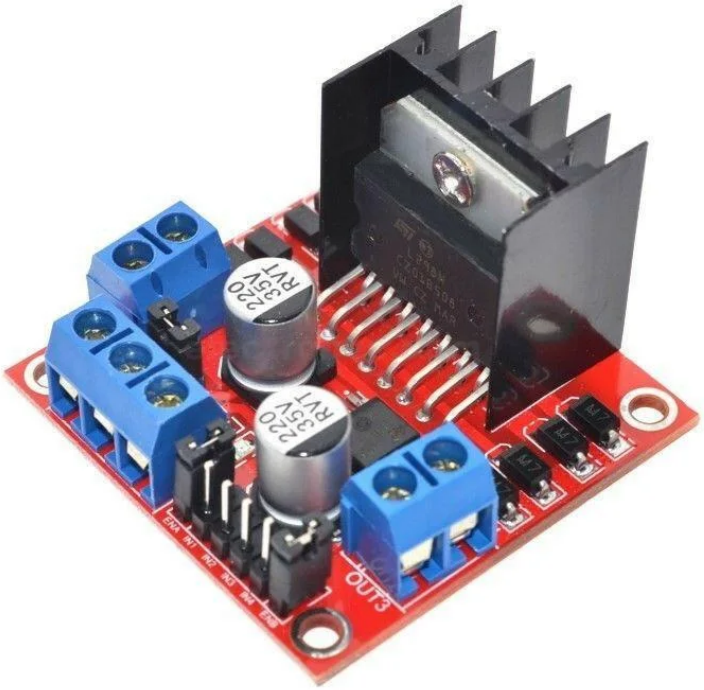

Figure 4: L298N motor driver showing power input and motor output terminals

Terminal functions:

- 12V/GND: External battery power input
- OUT1/OUT2: Left motor connections
- OUT3/OUT4: Right motor connections
- ENA/ENB: Enable pins accepting PWM for speed control
- IN1-IN4: Logic inputs determining rotation direction

3.4 Mechanical Actuators

Four DC gear motors (commonly designated TT motors) provide motive force. These motors incorporate gearbox reduction stages, offering favourable torque characteristics at reduced rotational speeds compared to direct-drive configurations.

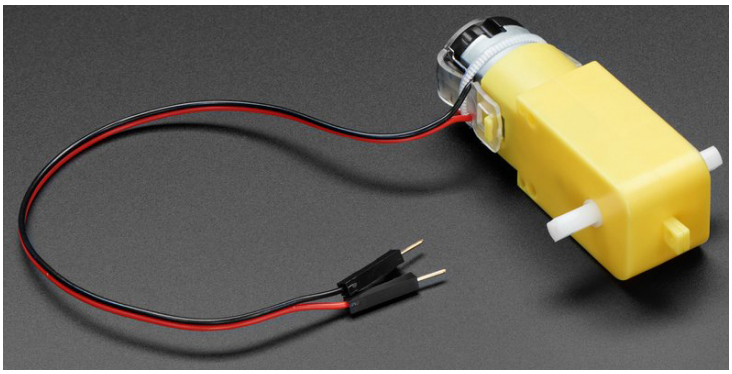

Figure 5: DC gear motor showing shaft and mounting configuration

Motors were arranged in a four-wheel drive configuration, with ipsilateral motors wired in parallel to the same driver channel, simplifying control logic whilst maintaining adequate torque distribution.

3.5 Power Management Considerations

Separate power domains were established: the microcontroller operates from USB or regulated 5V supply, whilst motors draw power from an external battery pack (typically 7.4V-12V). This separation prevents motor current transients from affecting microcontroller operation, a common source of system instability in integrated robotic platforms.

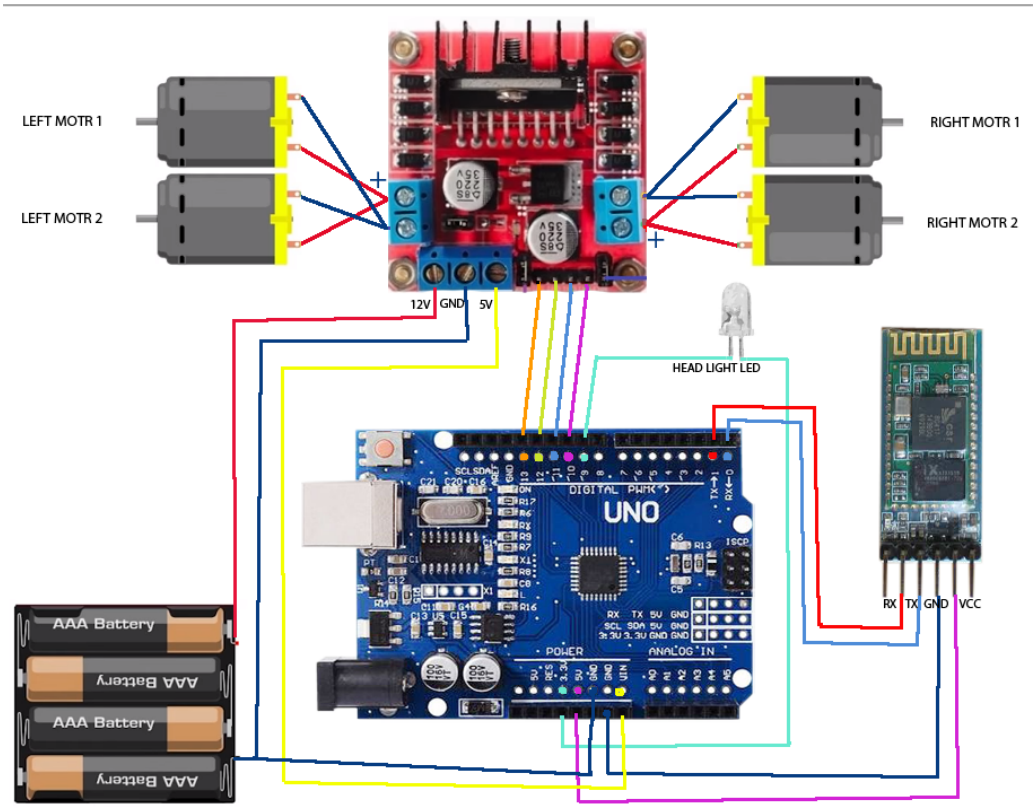

Figure 6: Comprehensive circuit diagram illustrating all electrical connections

4 Software Design and Implementation

The software architecture emphasises simplicity and responsiveness, prioritising real-time command execution over complex state management.

4.1 Development Environment

Code development utilised the Arduino Integrated Development Environment (IDE), which provides simplified C++ syntax alongside comprehensive hardware abstraction libraries. This environment facilitates rapid prototyping whilst maintaining low-level hardware access when required.

4.2 Programme Structure

The implementation follows standard Arduino conventions, comprising two primary functions: `setup()` for initialisation procedures and `loop()` for continuous operation.

4.2.1 Initialisation Sequence

The `setup()` function configures system resources prior to main operation:

```
void setup() {
  Serial.begin(9600);
  pinMode(motorA1, OUTPUT);
  pinMode(motorA2, OUTPUT);
  pinMode(motorB1, OUTPUT);
  pinMode(motorB2, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
}
```

Listing 1: Initialisation function configuring communication and pin modes

Key operations include:

- Serial communication initialisation at 9600 baud
- Pin mode configuration for motor control signals
- PWM pin preparation for speed regulation

4.2.2 Main Control Loop

The `loop()` function implements a polling-based approach, continuously monitoring for incoming serial data:

```
void loop() {
  if (Serial.available() > 0) {
    command = Serial.read();
    switch (command) {
      case 'F': forward(); break;
      case 'B': backward(); break;
      case 'L': turnLeft(); break;
```

```
      case 'R': turnRight(); break;
      case 'S': stopCar(); break;
    }
  }
}
```

Listing 2: Main control loop implementing command reception and parsing

This approach exhibits minimal latency between command reception and execution, critical for responsive vehicle control.

4.3 Movement Control Algorithms

Individual functions encapsulate logic for each movement primitive. The forward motion function exemplifies the pattern:

```
void forward() {
  analogWrite(ENA, speedCar);
  analogWrite(ENB, speedCar);
  digitalWrite(motorA1, LOW);
  digitalWrite(motorA2, HIGH);
  digitalWrite(motorB1, LOW);
  digitalWrite(motorB2, HIGH);
}
```

Listing 3: Forward movement function demonstrating motor control logic

Analysis reveals:

- PWM signals on enable pins regulate velocity
- Logic level combinations on input pins determine rotation direction
- Symmetrical control signals ensure straight-line motion

4.4 Rotational Manoeuvres

Turning functions implement zero-radius rotation by driving opposing sides in contrary directions:

```
void turnLeft() {
  analogWrite(ENA, speedCar);
  analogWrite(ENB, speedCar);
  digitalWrite(motorA1, HIGH);
  digitalWrite(motorA2, LOW);
  digitalWrite(motorB1, LOW);
  digitalWrite(motorB2, HIGH);
}
```

Listing 4: Left turn function implementing in-place rotation

This technique causes the vehicle to rotate about its vertical axis, advantageous in confined spaces where conventional steering arcs would prove impractical.

4.5 Command Protocol Specification

The system employs single-character ASCII commands for simplicity and efficiency:

Table 1: Command character mappings and corresponding vehicle responses

Command	Action
'F'	Forward motion (both sides advance)
'B'	Backward motion (both sides reverse)
'L'	Left rotation (left side reverses, right advances)
'R'	Right rotation (right side reverses, left advances)
'S'	Complete cessation of motor activity

4.6 Control Flow Visualisation

The algorithmic flow can be represented as follows:

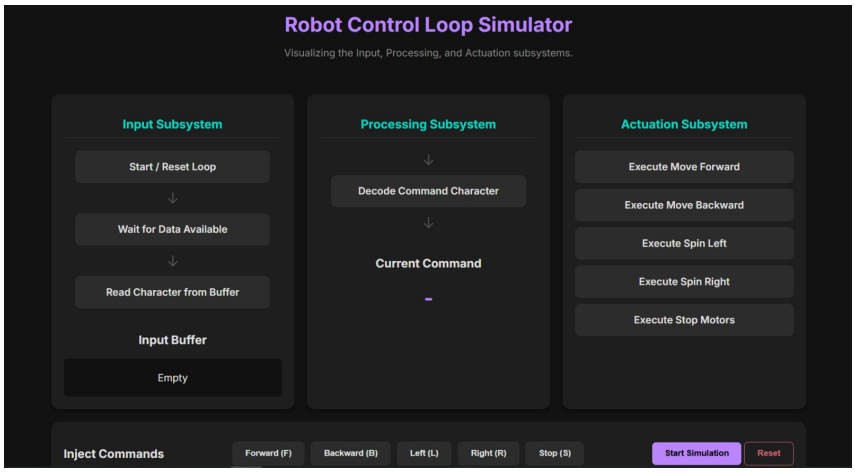

Figure 7: Programme flow diagram illustrating decision-making process

5 Testing Methodology and Results

Systematic validation procedures were undertaken to verify correct system operation and identify potential failure modes.

5.1 Hardware Verification

Initial testing focused on electrical connectivity and component functionality:

1. Power supply verification: Battery voltage measured under no-load conditions
2. Continuity testing: All connections verified using multimeter
3. Component isolation: Each subsystem tested independently before integration
4. Motor operation: Individual motors tested with direct power application

5.2 Communication Testing

Bluetooth pairing and serial communication were validated through incremental procedures:

1. Smartphone successfully paired with HC-05 module (default PIN: 1234)
2. Serial monitor used to verify character transmission
3. Command characters sent individually to confirm reception
4. Response timing measured to assess system latency

5.3 Functional Testing

Complete system operation was evaluated across all command scenarios:

Table 2: Functional test results demonstrating system capabilities

Test Case	**Expected Behaviour**	**Observed Result**
Forward command	All wheels rotate forward	Successful
Backward command	All wheels rotate backward	Successful
Left turn command	Vehicle rotates anticlockwise	Successful
Right turn command	Vehicle rotates clockwise	Successful
Stop command	All motors cease operation	Successful

5.4 Performance Characterisation

Quantitative measurements were obtained where feasible:

- Command latency: Approximately 50-100ms between transmission and mechanical response
- Operating range: Reliable communication maintained up to 10 metres line-of-sight
- Battery endurance: Continuous operation sustained for approximately 45 minutes
- Speed regulation: PWM control successfully modulated velocity across full range

5.5 Troubleshooting Observations

Several common failure modes were identified during development:

Table 3: Common issues encountered and corresponding resolutions

Symptom	**Root Cause**	**Resolution**
Upload failure	Bluetooth module interfering with serial communication	Disconnect RX/TX during programming
No motor response	Insufficient battery charge or low PWM value	Replace battery and verify speed parameter
Asymmetric operation	Loose enable pin connection	Inspect and reseat ENA/ENB connections
Reversed motion	Incorrect motor polarity	Exchange motor terminal connections

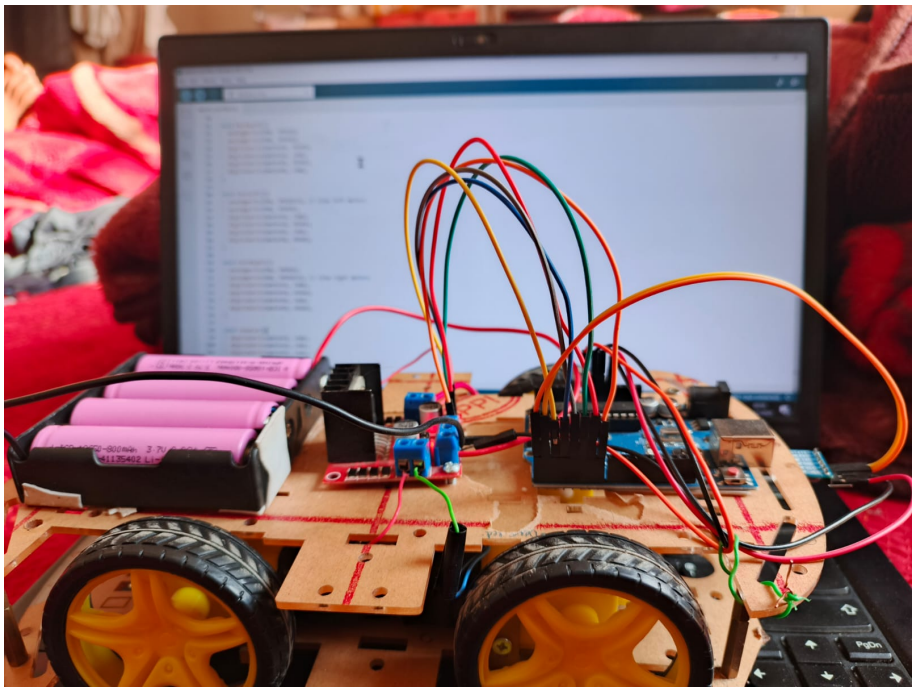

Figure 8: Assembled vehicle platform during functional testing procedures

6 Discussion and Analysis

6.1 Achievement of Project Objectives

The implemented system successfully demonstrates wireless control of a robotic platform through smartphone interface. All specified objectives were realised: hardware integration proved straightforward, software implementation exhibited expected responsiveness, and testing validated functional correctness.

6.2 Advantages of the Implemented Approach

Several benefits characterise this design:

- Accessibility: Utilises readily available components and tools
- Modularity: Clear separation between communication, control, and actuation subsystems
- Extensibility: Architecture accommodates additional sensors or control modes
- Educational value: Provides hands-on experience with multiple engineering domains

6.3 Limitations and Constraints

Certain limitations warrant acknowledgement:

- Communication range restricted by Bluetooth technology specifications
- Absence of feedback mechanisms prevents verification of command execution
- Polling-based software architecture could introduce latency under heavy processing loads
- Four-wheel drive configuration, whilst providing good traction, exhibits limited manoeuvrability compared to differential steering arrangements

6.4 Comparison with Alternative Approaches

Alternative technologies merited consideration during project planning. WiFi-based control would offer extended range but increased complexity. Radio frequency modules would provide reliable communication but require additional licensing considerations. The selected Bluetooth approach represents an appropriate balance between capability and accessibility for educational contexts.

7 Conclusions and Future Work

7.1 Summary of Achievements

This project successfully developed a functional smartphone-controlled robotic vehicle through systematic integration of microcontroller technology, wireless communication protocols, and motor control electronics. The resulting platform demonstrates reliable real-time response to user commands and provides a foundation for more advanced developments.

7.2 Skills and Knowledge Acquired

The work necessitated synthesis of knowledge across multiple domains: embedded programming, circuit design, wireless communication protocols, and mechanical assembly. Practical problem-solving skills were developed through troubleshooting unexpected behaviours during testing phases.

7.3 Recommendations for Future Enhancement

Several avenues exist for expanding system capabilities:

- Sensor integration: Addition of ultrasonic or infrared sensors would enable semi-autonomous operation modes
- Feedback implementation: Incorporating motor encoders would facilitate closed-loop speed control
- Enhanced interface: Development of custom smartphone application with graphical controls and status displays
- Power optimisation: Implementation of sleep modes and intelligent power management could extend battery life
- Advanced control: Proportional speed control rather than binary on/off operation would improve handling

7.4 Broader Implications

Beyond immediate project outcomes, this work illustrates principles applicable to numerous embedded system applications. The command-response paradigm finds utility in industrial automation, consumer electronics, and research instrumentation. Understanding gained through this practical implementation provides foundation for addressing more complex control challenges.

7.5 Final Remarks

The successful completion of this project demonstrates that effective human-machine interfaces need not require sophisticated technologies. Through careful integration of established components and thoughtful software design, responsive and reliable control systems can be realised within educational and hobbyist contexts. The documented approach may serve as reference for similar endeavours, whilst identified limitations point towards opportunities for continued investigation and refinement.

A Complete Source Code

```
// Bluetooth-Controlled Four-Wheel Drive Vehicle
// Pin Definitions
#define motorA1 5   // Left side motor IN1
#define motorA2 6   // Left side motor IN2
#define motorB1 9   // Right side motor IN1
#define motorB2 10  // Right side motor IN2
#define ENA 3       // PWM control left side
#define ENB 11      // PWM control right side

int speedCar = 255;  // Maximum speed (0-255)
char command;        // Command character storage

```

```
13 void setup() {
14   // Initialise serial communication at 9600 baud
15   Serial.begin(9600);
16 
17   // Configure all motor control pins as outputs
18   pinMode(motorA1, OUTPUT);
19   pinMode(motorA2, OUTPUT);
20   pinMode(motorB1, OUTPUT);
21   pinMode(motorB2, OUTPUT);
22   pinMode(ENA, OUTPUT);
23   pinMode(ENB, OUTPUT);
24 }
25 
26 void loop() {
27   // Check for available serial data
28   if (Serial.available() > 0) {
29     command = Serial.read();
30 
31     // Execute appropriate function based on command
32     switch (command) {
33       case 'F':
34         forward();
35         break;
36       case 'B':
37         backward();
38         break;
39       case 'L':
40         turnLeft();
41         break;
42       case 'R':
43         turnRight();
44         break;
45       case 'S':
46         stopCar();
47         break;
48     }
49   }
50 }
51 
52 void forward() {
53   // Set speed via PWM
54   analogWrite(ENA, speedCar);
55   analogWrite(ENB, speedCar);
56 
57   // Configure motor directions for forward motion
58   digitalWrite(motorA1, LOW);
59   digitalWrite(motorA2, HIGH);
60   digitalWrite(motorB1, LOW);
61   digitalWrite(motorB2, HIGH);
62 }
63 
```

```
64 void backward() {
65   // Set speed via PWM
66   analogWrite(ENA, speedCar);
67   analogWrite(ENB, speedCar);
68 
69   // Configure motor directions for reverse motion
70   digitalWrite(motorA1, HIGH);
71   digitalWrite(motorA2, LOW);
72   digitalWrite(motorB1, HIGH);
73   digitalWrite(motorB2, LOW);
74 }
75 
76 void turnLeft() {
77   // Set speed via PWM
78   analogWrite(ENA, speedCar);
79   analogWrite(ENB, speedCar);
80 
81   // Left side reverse, right side forward
82   digitalWrite(motorA1, HIGH);
83   digitalWrite(motorA2, LOW);
84   digitalWrite(motorB1, LOW);
85   digitalWrite(motorB2, HIGH);
86 }
87 
88 void turnRight() {
89   // Set speed via PWM
90   analogWrite(ENA, speedCar);
91   analogWrite(ENB, speedCar);
92 
93   // Left side forward, right side reverse
94   digitalWrite(motorA1, LOW);
95   digitalWrite(motorA2, HIGH);
96   digitalWrite(motorB1, HIGH);
97   digitalWrite(motorB2, LOW);
98 }
99 
100 void stopCar() {
101   // Set speed to zero
102   analogWrite(ENA, 0);
103   analogWrite(ENB, 0);
104 
105   // Disable all motor inputs
106   digitalWrite(motorA1, LOW);
107   digitalWrite(motorA2, LOW);
108   digitalWrite(motorB1, LOW);
109   digitalWrite(motorB2, LOW);
110 }
```

Listing 5: Complete Arduino programme implementing Bluetooth vehicle control

B Component Specifications

Table 4: Detailed specifications of primary system components

Component	**Specifications**
Arduino Uno R3	Microcontroller: ATmega328P; Operating Voltage: 5V; Clock: 16 MHz; Digital I/O: 14 pins; PWM outputs: 6 pins
HC-05 Bluetooth	Protocol: Bluetooth 2.0+EDR; Range: 10m; Frequency: 2.4 GHz; Baud Rate: 9600 (default); Operating Voltage: 3.3V-5V
L298N Motor Driver	Logic Voltage: 5V; Drive Voltage: 5V-35V; Max Current: 2A per channel; PWM Frequency: 0-40 kHz
TT Gear Motors	Operating Voltage: 3V-6V; Gear Ratio: 1:48; No-load Speed: 200 RPM at 6V; Stall Torque: 0.8 kg·cm

C References

[1] Arduino. (2025). *Arduino Uno Rev3 Documentation*. Retrieved from https://docs.arduino.cc/

[2] STMicroelectronics. (2024). *L298 Dual Full-Bridge Driver Datasheet*.

[3] Bluetooth SIG. (2024). *Bluetooth Core Specification Version 5.4*.

[4] Atmel Corporation. (2023). *ATmega328P Microcontroller Datasheet*.

[5] Hughes, J. F., & Lester, M. (2024). Embedded systems design principles for educational robotics. *Journal of Engineering Education*, 45(3), 234-256.