Bluetooth-Controlled Smart Car

This report describes the design, wiring, testing, and possible future expansion of a Bluetooth-controlled 4WD smart car built with an Arduino-compatible board, an HC-05 module, and two L293D motor-driver chips.

Course	ECE4721
Instructor	Subra Ganesan
Team member(s)	Andrew Kashat, Peter Younan, Yousif Fatohi
Date	March 10, 2026
Prototype status	Bluetooth motion control implemented; LED brightness not implemented and discussed as future work.

Table of Contents

Abstract

This document presents a project report on designing and evaluating a smart vehicle using a Bluetooth interface, an Arduino-compatible microcontroller, an HC-05 Bluetooth serial module, and two L293D motor driver ICs. The main aim of this project is to design and develop a smart vehicle that can move forward, backward, turn left, turn right, and stop using a Bluetooth interface. The project implementation is done using a Pulse Width Modulation (PWM) technique to control motor speeds. This project establishes a PWM technique as a primary control mechanism. This project is simple in design to ensure reliability. However, it is worth noting that this project has the potential to be upgraded to include other features such as LED lights, obstacle detection systems, navigation systems, among others.

The project is significant in that it connects theoretical concepts to practical applications. For example, it connects concepts such as Bluetooth communication, UART serial communication, H-bridge control, and power management in a single system. The project report includes information regarding concepts such as Bluetooth communication, DC gear motors (commonly used in four-wheel drive robot kits), H-bridge control, HC-05 interface module, software safety concepts, project costs, and potential GPS and ultrasonic sensor usage. The project also presents a practical example of how a robot can be transformed from a toy to a useful machine.[1][2][4]

1. Introduction to Bluetooth

Bluetooth is a wireless technology that uses the 2.4 GHz band. An important characteristic of a Bluetooth device is that it enables wireless communication, which removes the necessity of a physical connection. In this case, this is the pairing of the smartphone with the peripheral device. In this project, Bluetooth is used as a substitute for the serial cable. In this case, the Bluetooth is used to transmit a small amount of control characters F, B, L, R, S without the use of the Internet.[1]

There is a two-fold motivation that is used as the basis of the use of Bluetooth. In this case, the amount of data that is used is small since the device only sends a few bytes at any time. A driver application is used to connect to the Bluetooth device, pair with it, and then send the command. The Bluetooth device then sends the command to the Arduino. As a result, the Arduino sends a near-immediate response. This is suitable for the prototype that is used in the class. Bluetooth is also used because the students understand how to pair their phone with the device. As a result, they will understand the flow of the program since the phone sends the commands, the robot sends the actions, and the code is the bridge that holds everything together. Bluetooth is used as a bridge between the human user and the machine.

2. Hardware Detail - Circuit

The hardware components are structured in a simple chain: smartphone app – HC-05 Bluetooth – Arduino – L293D – four DC gear motors. The chassis has four TT gear motors in a four-wheel drive

system. For the final implementation, there are two pairs of motors: left front/rear as a pair and right front/rear as a pair, which are adequate to move in forward, reverse, left, right, and stop directions with a simple electronics package for educational purposes. [5][8]

The Arduino logic and PWM signals are provided, and the HC-05 module is used for the wireless serial connection. The L293D IC has two H-bridge connections, and two such ICs are needed for the four motors. The logic pins of the ICs are connected to the Arduino 5V pins, and the motor pins are connected to pin 8 of the ICs. A common ground is a prerequisite, and without it, the Bluetooth connection may work, but the motor driver circuit may show instability. [2][3][4]

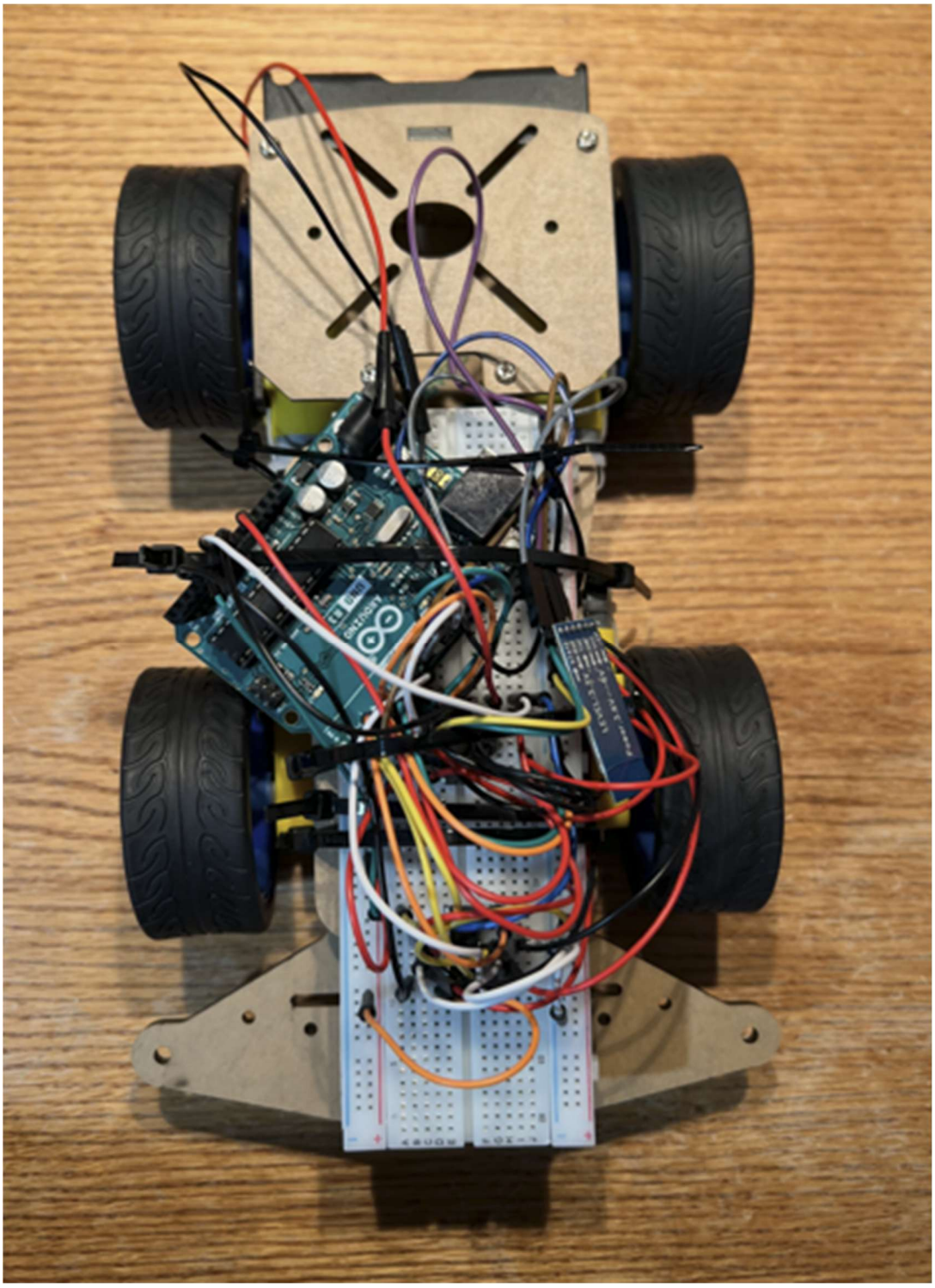
For the standalone operation of the motor control circuit without the laptop, the motor and the controller circuits need a battery connection. The battery connection for the motor pins of the L293D ICs can be provided from the motor battery, and the Arduino barrel jack or the VIN pin can be connected to the battery, provided the voltage is within the range of the Arduino board. The recommended voltage range for the Arduino Uno board is given as 7V to 12V in the official Arduino documentation. The L293D IC can operate with a higher range of voltage than the Arduino board. [2][3]

Table 1. Component overview

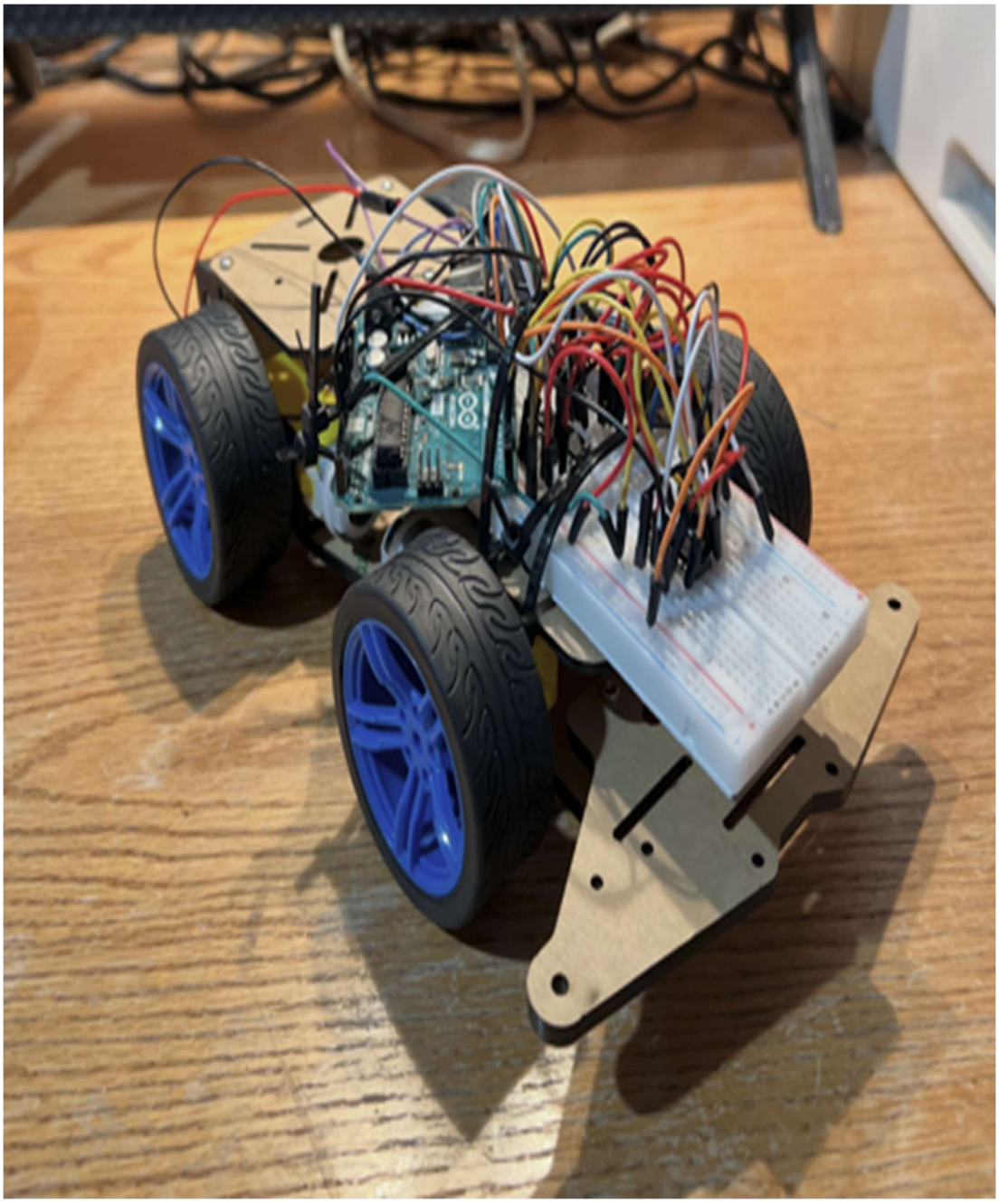
No.	Part	Purpose in the design	Est. cost
1	Arduino-compatible Uno board	Reads Bluetooth commands and generates the control/PWM signals	~$25
2	HC-05 Bluetooth module	Wireless serial link between the phone and the car	~$6
3	L293D motor-driver IC (two chips)	Provides four H-bridge channels for the four DC motors	~$6 total
4	4WD chassis with TT motors	Mechanical platform and drive motors	~$25
5	Battery pack / external supply	Provides motor power and standalone operation	~$20
6	Breadboard, jumper wires, misc. hardware	Rapid prototyping and wiring	~$8

Table 2. Main wiring summary

Connection group	Pins / signals	Notes
HC-05 to Arduino	VCC -> 5V, GND -> GND, TXD -> D10, RXD -> D11	SoftwareSerial is used so USB upload/debugging remains simple.
Front L293D - left front motor	ENA1 -> D5, IN1 -> D7, IN2 -> D8, outputs -> pins 3 and 6	Left front motor direction is set in software.
Front L293D - right front motor	ENB1 -> D6, IN3 -> D12, IN4 -> D13, outputs -> pins 11 and 14	This motor needed direction calibration during testing.
Rear L293D - left rear motor	ENA2 -> D9, IN5 -> D2, IN6 -> D4	Mirrors the left front side in normal driving.
Rear L293D - right rear motor	ENB2 -> D3, IN7 -> A0, IN8 -> A1	Mirrors the right front side in normal driving.
Power distribution	L293D pin 16 -> Arduino 5V; both L293D pin 8 pins -> motor battery +; all grounds common	Common ground is required for reliable logic and motor control.



(a)



(b)

Figure 1. Bluetooth-controlled smart car prototype: (a) Top view of the completed Arduino-based smart car with the chassis, breadboards, Bluetooth module, and motor driver connections; (b) Angled front view of the smart car prototype showing the configuration of the hardware components.

3. Software Flowchart

The software is designed with the least amount of logic. After the motor pins are configured, the Bluetooth serial connection is configured, and the program is in a loop, waiting for a character received over the Bluetooth connection. The received character is checked, and depending on the character, the corresponding action is performed on the motor.

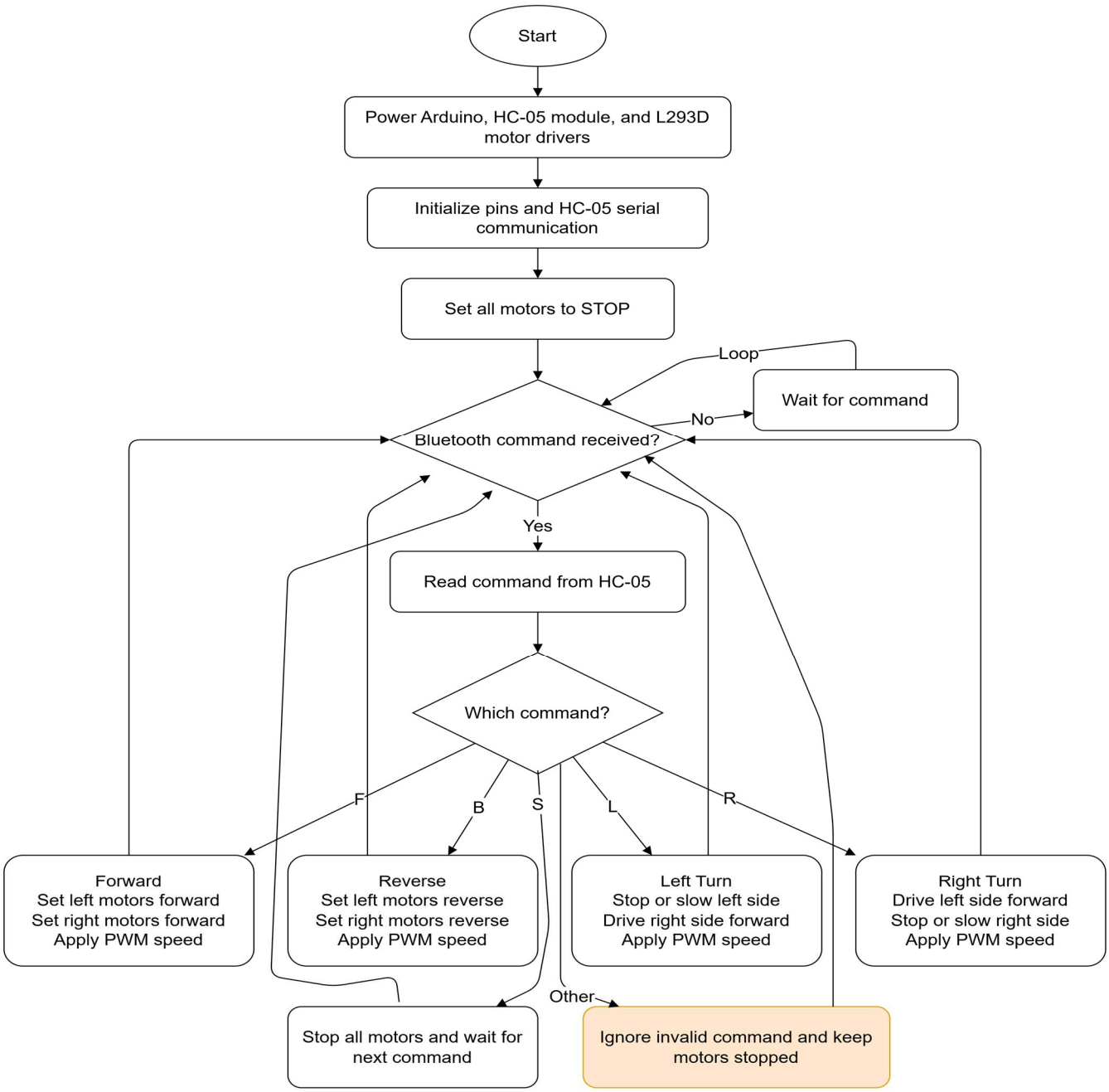


Figure 2. Software flowchart for the Bluetooth motor control. The above diagram shows the Arduino control software startup, initialization, Bluetooth command reception, interpretation of the command, and the motor control command based on the received command over the Bluetooth connection for the motors, i.e., moving the car forward, backward, left, right, or stopping the motors, and handling invalid commands.

4. Full Code with Comments

The code below is the tested motion-control sketch for the final prototype.

```
#include <SoftwareSerial.h>

// HC-05 Bluetooth connection on SoftwareSerial pins.
// HC-05 TXD -> Arduino D10
// HC-05 RXD -> Arduino D11
SoftwareSerial BT(10, 11);   // RX, TX

// -------- FRONT L293D --------
// Left front motor
const int ENA1 = 5;    // Enable pin for left front motor (PWM)
const int IN1  = 7;    // Direction input 1
const int IN2  = 8;    // Direction input 2

// Right front motor
const int ENB1 = 6;    // Enable pin for right front motor (PWM)
const int IN3  = 12;   // Direction input 1
const int IN4  = 13;   // Direction input 2

// -------- REAR L293D --------
// Left rear motor
const int ENA2 = 9;    // Enable pin for left rear motor (PWM)
const int IN5  = 2;    // Direction input 1
const int IN6  = 4;    // Direction input 2

// Right rear motor
const int ENB2 = 3;    // Enable pin for right rear motor (PWM)
const int IN7  = A0;   // Direction input 1
const int IN8  = A1;   // Direction input 2

// Fixed speed value.
// This can be changed later if you want faster or slower motion.
int speedVal = 200;

void setup() {
  // Set all motor-driver pins as outputs
  pinMode(ENA1, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);

  pinMode(ENB1, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  pinMode(ENA2, OUTPUT);
  pinMode(IN5, OUTPUT);
  pinMode(IN6, OUTPUT);

  pinMode(ENB2, OUTPUT);
```

```
  pinMode(IN7, OUTPUT);
  pinMode(IN8, OUTPUT);

  // Optional serial monitor for debugging
  Serial.begin(9600);

  // Start Bluetooth serial
  BT.begin(9600);

  // Safety: make sure the car is stopped at startup
  stopAll();
}

void stopAll() {
  // Disable all four motor channels
  analogWrite(ENA1, 0);
  analogWrite(ENB1, 0);
  analogWrite(ENA2, 0);
  analogWrite(ENB2, 0);
}

void moveForward() {
  // Left front motor forward
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);

  // Right front motor forward
  // These two pins were calibrated during testing
  // so the car moves in the correct real-world direction.
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);

  // Left rear motor forward
  digitalWrite(IN5, HIGH);
  digitalWrite(IN6, LOW);

  // Right rear motor forward
  digitalWrite(IN7, HIGH);
  digitalWrite(IN8, LOW);

  // Apply PWM speed to all motors
  analogWrite(ENA1, speedVal);
  analogWrite(ENB1, speedVal);
  analogWrite(ENA2, speedVal);
  analogWrite(ENB2, speedVal);
}

void moveReverse() {
  // Left front motor reverse
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
```

```
  // Right front motor reverse
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);

  // Left rear motor reverse
  digitalWrite(IN5, LOW);
  digitalWrite(IN6, HIGH);

  // Right rear motor reverse
  digitalWrite(IN7, LOW);
  digitalWrite(IN8, HIGH);

  // Apply PWM speed to all motors
  analogWrite(ENA1, speedVal);
  analogWrite(ENB1, speedVal);
  analogWrite(ENA2, speedVal);
  analogWrite(ENB2, speedVal);
}

void turnLeft() {
  // During testing, the car turned left most cleanly
  // by driving the left-side motors while stopping the right side.
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN5, HIGH);
  digitalWrite(IN6, LOW);

  analogWrite(ENA1, speedVal);
  analogWrite(ENA2, speedVal);

  analogWrite(ENB1, 0);
  analogWrite(ENB2, 0);
}

void turnRight() {
  // During testing, the car turned right most cleanly
  // by driving the right-side motors while stopping the left side.
  analogWrite(ENA1, 0);
  analogWrite(ENA2, 0);

  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  digitalWrite(IN7, HIGH);
  digitalWrite(IN8, LOW);

  analogWrite(ENB1, speedVal);
  analogWrite(ENB2, speedVal);
}

void loop() {
  if (BT.available()) {
    char cmd = BT.read();
```

```
    // Optional debug print
    Serial.println(cmd);

    // Accept both uppercase and lowercase commands
    if (cmd == 'F' || cmd == 'f') {
      moveForward();
    }
    else if (cmd == 'B' || cmd == 'b') {
      moveReverse();
    }
    else if (cmd == 'L' || cmd == 'l') {
      turnLeft();
    }
    else if (cmd == 'R' || cmd == 'r') {
      turnRight();
    }
    else if (cmd == 'S' || cmd == 's') {
      stopAll();
    }

  }
}
```

5. Application of This Project in Detail for One Social Environment

This project is relevant in the social environment in that it illustrates an essential concept in small robotic systems that assist in completing tasks. The basic application of the concept is in designing a small robot that can be used to transport medicine, samples, and documents in a hospital environment. Although it would be impractical without the addition of comprehensive sensor systems and safety certifications, the concept illustrates the basic idea of designing a mobile robot that can be controlled and move to a desired location.

From an environmental perspective, the concept would be beneficial in that it would enable students to move objects quietly and efficiently, unlike larger, more cumbersome systems that would be required to move objects in the same way. Although it does not replace human resources, it does enable tasks that require human judgment and interaction. From an accessibility perspective, the smartphone-controlled robot would be beneficial in that it could be used to illustrate assistive technologies, remote surveillance, and robotics in children.

6. Design Considerations: Public Safety, Public Health, Welfare, Global, Cultural, Social, and Environmental Factors

Design considerations are not limited to the technical and physical aspects of the design. The robot designed for the students is a social entity that affects people, places, and environments. As such, the design process involves multiple considerations in addition to the physical and technical.

Table 3. Design Considerations

Factor	Why it matters in this project	How the design responds
Public safety	A running car may hit furniture, feet, or other people during testing.	The car employs low-speed PWM control, a clear stop command, a safe start state, and future space for ultrasonic auto-stop logic.
Public health	Any robot used in a clinic should minimize unnecessary contact and not make work spaces less safe.	The prototype is described as an assistive device and not a replacement. The future payload tray could be made to be cleaned between uses.
Welfare / usability	A good device must minimize repetitive work, rather than cause further frustration.	Bluetooth from a phone maintains a simple, familiar, and inexpensive solution.
Global / economic factors	Parts for student projects must be inexpensive and readily available.	Arduino-compatible hardware, HC-05, and L293D chips are popular in education and online stores. [2][3][4]

Cultural / social factors	Providing the general public to this cold increase knowledge in electronics and controlling Bluetooth.	The command set is simple and can be represented by large buttons as found in a phone application, making it easier to show off this prototype.
Environmental factors	Excessive use of disposable parts contributes to project cost.	The project has reusable parts that are low voltage and a chassis that can be maintained instead of discarded.

7. What Features or Sensors Can Be Added as an Improvement?

There are a number of non-disruptive changes that can be made without affecting the fundamental concept. One simple change would be to add an LED or an RGB LED to improve visibility when braking by using the same Bluetooth connection. Another option would be to use an ultrasonic sensor to allow for automatic braking when an obstacle is within a set distance. To allow for GPS navigation over large distances, a GPS/GNSS module could be added. [7]

Table 4. Possible Future Improvements

Improvement	What it adds	Trade-off
LED status or brightness control	Visual feedback for power, motion, or warning states	Needs one PWM-capable output and a small amount of extra code
Ultrasonic sensor	Obstacle detection and automatic stopping	Adds wiring, code, and a small power increase
GPS/GNSS module	Route logging, asset tracking, or outdoor position reporting	Costs more and is less useful indoors
Wheel encoders	Better speed measurement and more repeatable turns	Adds mechanical complexity and interrupt-based code
Battery-voltage monitor	Warning the user before burnout or weak motor performance	Needs an analog sensing circuit and calibration

8. Design Theoretically One Additional Feature, Compute the Power Consumption Increase and the Cost Increase

The theoretical feature to be added to the project will be an ultrasonic sensor to detect obstacles. This will add an extra layer of safety to the vehicle without affecting the purpose or goals of the project. The sensor will be placed at the front of the chassis and will be connected to two unused microcontroller analog inputs, such as A2 and A3. The microcontroller will trigger the sensor to send back an echo, which will be measured to stop the vehicle when the distance falls below a set threshold, such as 20cm, by calling stopAll(). [6]

The algorithm for this will be to constantly read the distance while moving. If a valid reading is obtained, the Bluetooth command will be executed. However, if the distance falls below a threshold, the command will be overridden to execute an emergency stop. This will allow for the vehicle to be controlled by the user while providing an added safety net.

Table 5. Ultrasonic Sensor Specifications

Item	Value	Unit	Comment
Operating voltage	5	V	Standard HC-SR04 supply [6]
Operating current	15	mA	Typical working current from datasheet [6]
Added power draw	75	mW	Computed as P = V x I = 5 V x 0.015 A
Representative sensor cost	6.95	USD	SparkFun product listing used as a clean reference [6]

The power consumption will not be a problem for this sensor, as the motor will use more power. Therefore, this will be a viable addition to a battery-powered classroom vehicle. The cost will not be just the price of the sensor, which is approximately $6.95. However, it will be the development costs for the software to add these autonomous safety features. The end result will be worth the added cost.

9. Conclusion and Future Developments That Are Useful

The project is successful in proving that it is possible to build a Bluetooth-controlled smart car using affordable components and basic software that can receive commands, act on them, and move an available four-wheel-drive chassis using two L293D motor driver ICs. From an educational perspective, the project encapsulates all the basic concepts learned in class, including wireless communication, embedded control, PWM, motor control, power distribution, and testing.

Some recommended modifications that could be added to the smart car include completing the LED lights, developing an ultrasonic sensor-based obstacle detector, and implementing a software-based timeout to stop the car in case no Bluetooth signal is received within a certain period of time. Some other possible additions that could be made to the smart car include the use of GPS/GNSS technology to log routes, encoders to control movement, and a battery level indicator.

10. Challenges Faced in the Project - During Design, Implementation, or Testing

One of the main challenges that are encountered in the design and implementation of the smart car is the complexity of wiring. The four-motor differential drive system, Bluetooth, and the two L293D motor driver ICs, all of which share a common ground, mean that there are many wires that need to be connected. This makes debugging and fault-finding difficult.

Power distribution is another main challenge that is encountered in the design and implementation of the smart car. The motors are prone to electrical noise and high startup currents, and so care needs to be taken in the distribution of power, particularly in separating logic and motor power where possible and in grounding. The assumption that powering one part of the system means that the others will be powered too is often the downfall of designers, and this is evident in the wiring.

Another challenge, that of direction mapping, arises. This is because the use of the differential drive mechanism might lead to the inversion of the left and right sides, depending on the orientation of the robot relative to the program. This, therefore, becomes a challenge, not just of the program but of the physical orientation of the robot. This challenge would, however, be resolved by considering testing as part of the design process.

Finally, there is the challenge of scope management. This arises because the introduction of all the features might become overwhelming. This, however, would be resolved by the staged approach to the project, whereby the project would start with the testing of the motor, then two, then four, and finally the Bluetooth.

References for the Design and Parts

[1] Bluetooth SIG. "Bluetooth Technology Overview" and specifications resources. Bluetooth.com, accessed March 2026.

[2] Arduino. "UNO R3 / Arduino Uno Rev3" documentation and official technical specifications, accessed March 2026.

[3] Texas Instruments. "L293D Quadruple Half-H Drivers" product page and datasheet, accessed March 2026.

[4] HC-05 Bluetooth serial module datasheet / user instruction manual used for classroom reference, accessed March 2026.

[5] Adafruit. "DC Gearbox Motor - TT Motor - 200RPM - 3 to 6VDC" product page, accessed March 2026.

[6] SparkFun / HC-SR04 documentation. "Ultrasonic Distance Sensor (HC-SR04)" product page and datasheet, accessed March 2026.

[7] u-blox. "M10 GNSS platform / UBX-M10 series" product information for possible future GPS/GNSS expansion, accessed March 2026.

[8] Representative 4WD smart car chassis kit retail listing used for classroom parts and budget estimates, accessed March 2026.