Design and Implementation of a Bluetooth-Based Remote Control Car

Abstract:

This work is on a Bluetooth control motor car. Bluetooth controlled car is battery powered model cars that can be controlled from a distance using a mobile app connected via Bluetooth. This controlled system is adopted in many vehicles like cars, boats, planes, and even helicopters and scale railway locomotives. In this work we will use a couple of ICs and a motor fixed to a chassis to make a remote control car. This work involve a brief idea is to transmit control signals through radio frequency and receive it through a receiver module in the car. We will have two switches in our remote control to power each motor of the car.

The objective of this project is to design a low cost remote control toy or robot. In this work we shall assemble the motors, circuit, and wheels on the chassis.

Introduction:

Remote control (RC) cars have been a popular toy for decades, providing users with a fun and engaging way to control the movement of a vehicle remotely. Here we use 1 Arduino Uno, 1 Motor Driver L298n, 4 Battery, 1 yellow motor car chassis Kit, 1 Buck converter with display, 1 Battery Holder Li-ion (4 cells), 1 Bluetooth Module HC-05 and Connecting wire all set. A controller app in our smartphone by which we control the car via Bluetooth.

Background of the Study:

Remote control car referred to as RC Car, RC is short for Radio control or Remote control. In Europe, Japan and Southeast Asian countries, the remote control model car race is a kind of sports competitions and a noble sport, even a high-tech hobby. This is a competition from age limit because enthusiasts who participate in this sport is aged from 8 to 80 years. This also suggests that the remote control cars are the same as remote control aircraft, whether adult or children can play. However, it has a certain degree of risk, because the remote control car is not a simple toy, its

principle is the same as the real car. So, I still recommend that children would better to accompany by their parents.

Electric remote control car is primarily provided power by an electric motor. Due to powered by a battery, it has higher overall operating efficiency and excellent acceleration performance of vehicles. Because of Clean Energy, the vehicle maintenance is very convenient, just pay attention to the maintenance of transmission parts. But because the electric motor remote control car need high requirements of motor performance and battery performance, so the motor and battery consumption is very high in the beginning, coupled with the electronic speed control, remote equipment costs.

Diagram of this project:

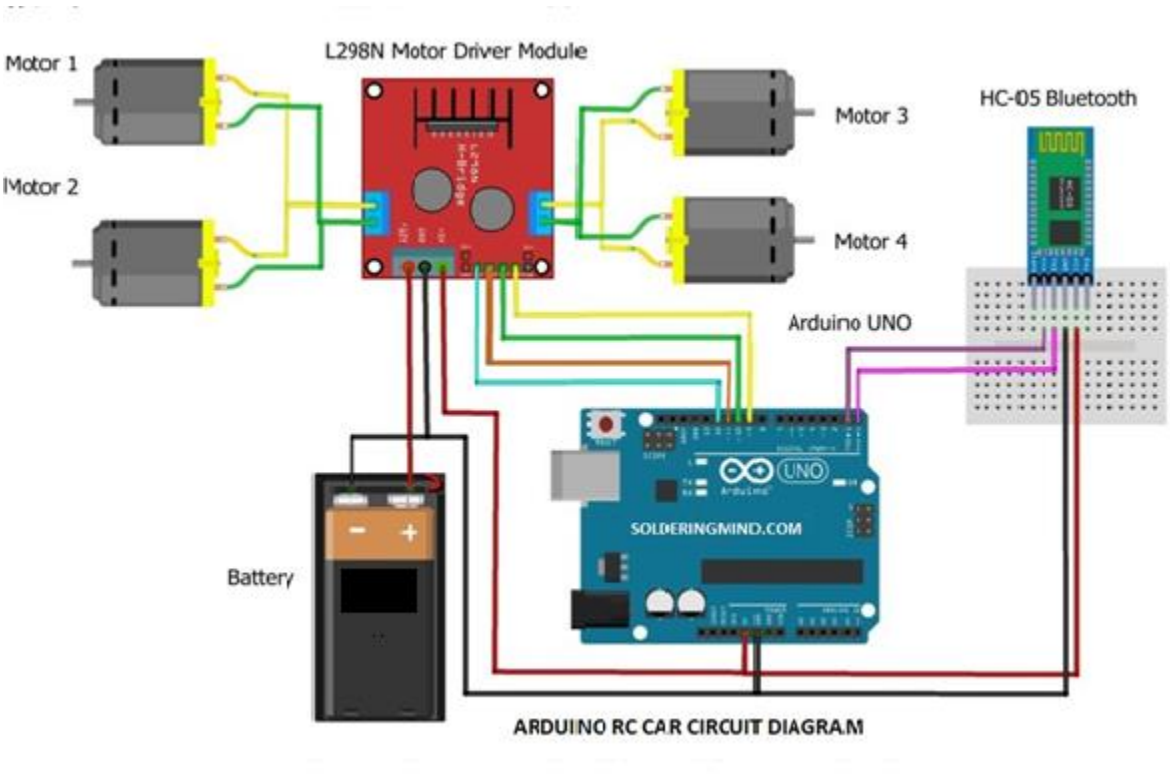

Fig_1: Diagram of Project

Arduino UNO:

Arduino UNO is a microcontroller board based on ATmega328P. It has 14 digital input/output pins (6 of which can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

Arduino UNO is flexible, and easy-to-use programmable open-source microcontroller board that can be integrated into a variety of electronic projects. This board can be interfaced with other Arduino boards, Arduino shield, raspberry pi board and can control relays, LEDs, servos and motors as outputs. This electronic platform includes interconnects, LEDs, and more. There are different types of Arduino boards in the market including Arduino UNO. Five key advantages of the Arduino programming language are easy to learn and use. Arduino programming language is based on C++, with a simple and straightforward syntax that is easy to pick up even for beginners.

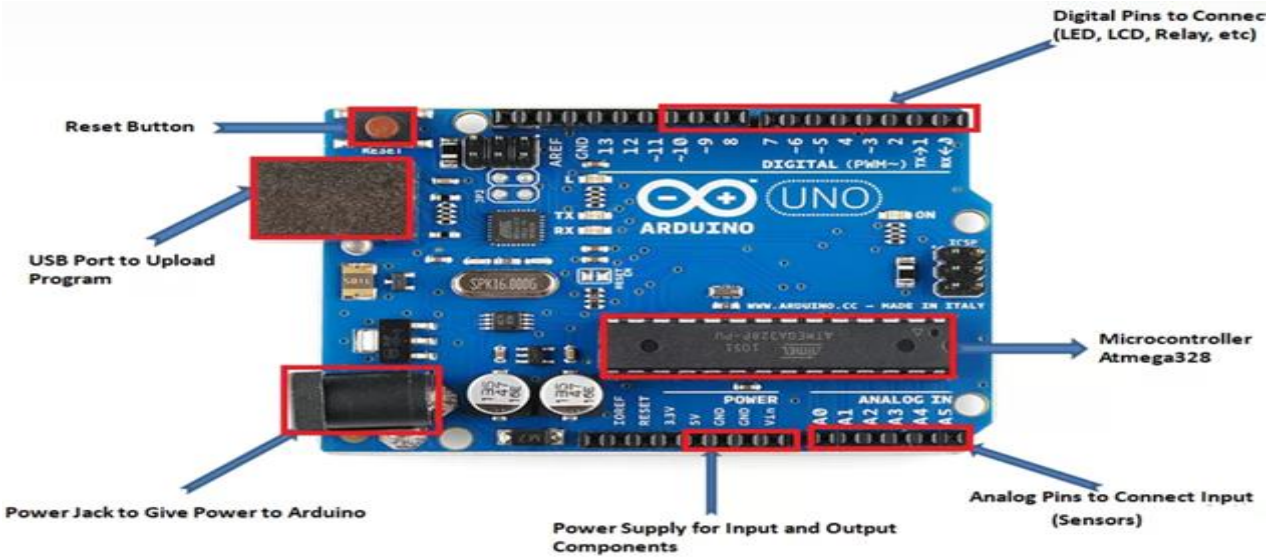

Fig 2: Diagram of Arduino Uno

Costing:

Name of product	Quantity	Price
Arduino Uno	1	950 tk
Motor Driver L298n	2	160*2 = 320 tk
Battery	4	80*4= 320 tk
Yellow motor Car chassis Kit	1	900 tk
Buck converter With display	1	230 tk
Battery Holder Li-io (4 cells)	1	230 tk
Bluetooth Module HC 05	1	250 tk
Connecting wire all set	1	210 tk
Li-ion charger	1	250 tk
		Total = TK.3480

Code:

```
#define ENA 5
#define IN1 6
#define IN2 7
#define IN3 8
#define IN4 9
#define ENB 10
int incomingByte=0; // for incoming serial data
int speed_min = 125; //the minimum "speed" the motors will turn - take it lower and motors don't turn
int speed_max = 255; //

int speed_left = speed_max; // set both motors to maximum speed
int speed_right = speed_max;

void left();
void right();
void forward();
void backward();
void forward_left();
```

```
void forward_right();
void back_left();
void back_right();


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  // put your setup code here, to run once:
pinMode(ENA,OUTPUT);
pinMode(IN1,OUTPUT);
pinMode(IN3,OUTPUT);
pinMode(IN4,OUTPUT);
pinMode(ENB,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
if (Serial.available() > 0) {
    incomingByte = Serial.read();
    }
  switch(incomingByte)
  {
     case 'S':
      {
        stopy();
       //Serial.println("Stop\n");
       incomingByte='*';}
     break;

     case 'B':

     {  backward();

      // Serial.println("Forward\n");
       incomingByte='*';}
     break;

      case 'F':
```

```
{   forward();
  // Serial.println("Backward\n");
   incomingByte='*';}
 break;

 case 'R':
 // turn right
  {
   right();
  // Serial.println("Rotate Right\n");
   incomingByte='*';}
 break;
   case 'L':
  {
   left();
   //Serial.println("Rotate Left\n");
   incomingByte='*';}
 break;
 case '1':

  { speed_left = 20;
   speed_right = 20;
   //Serial.println("Speed 1\n");
   incomingByte='*';}
 break;
case '2':
  {
    speed_left = 40;
   speed_right = 40;
   //Serial.println("Speed 2 \n");
   incomingByte='*';}
 break;
case '3':
  {
    speed_left = 60;
   speed_right = 60;
   //Serial.println("Speed 3 \n");
   incomingByte='*';}
```

```
break;
   case '4':
  {
   speed_left = 80;
  speed_right = 80;
  //Serial.println("Speed 4 \n");
  incomingByte='*';}
break;
   case '5':
  {
   speed_left = 100;
  speed_right = 100;
  //Serial.println("Speed 5 \n");
  incomingByte='*';}
break;
   case '6':
  {
   speed_left = 120;
  speed_right = 120;
  //Serial.println("Speed 6 \n");
  incomingByte='*';}
break;
   case '7':
  {
   speed_left = 140;
  speed_right = 140;
 // Serial.println("Speed 7 \n");
  incomingByte='*';}
break;
   case '8':
  {
   speed_left = 160;
  speed_right = 160;
  //Serial.println("Speed 8 \n");
  incomingByte='*';}
break;
   case '9':
  {
   speed_left = 200;
```

```
      speed_right = 200;
      //Serial.println("Speed 9 \n");
      incomingByte='*';}
    break;
      case 'q':
    {
      speed_left = 255;
    speed_right = 255;
    Serial.println("Speed full \n");
    incomingByte='*';}
    break;
      case 'J':
    {
    back_right();
    Serial.println("Speed full \n");
    incomingByte='*';}
    break;
    case 'H':
    {
    back_left();
    Serial.println("Speed full \n");
    incomingByte='*';}
    break;
    case 'I':
    {
    forward_right();
    Serial.println("Speed full \n");
    incomingByte='*';}
    break;
    case 'G':
    {
    forward_left();
    Serial.println("Speed full \n");
    incomingByte='*';}
    break;
  }

}
void left(){
```

```
  //left
  analogWrite(ENA,speed_left);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,speed_right);
  };
void right(){
   //right
  analogWrite(ENA,speed_left);
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  analogWrite(ENB,speed_right);

  };
void forward_left(){
  forward_left
  analogWrite(ENA,speed_left);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,speed_right);


  };
void forward_right(){
   //forward right
  analogWrite(ENA,speed_left);
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,speed_right);
```

```
  };
void back_right(){
   //backward right
  analogWrite(ENA,speed_left);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,speed_right);

  };
void back_left(){

  //backward left
  analogWrite(ENA,speed_left);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  analogWrite(ENB,speed_right);



  };
void forward(){
 //forward
  analogWrite(ENA,speed_left);
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,speed_right);


  };
void backward(){
 //backward
  analogWrite(ENA,speed_left);
  digitalWrite(IN1,LOW);
```

```
 digitalWrite(IN2,HIGH);
 digitalWrite(IN3,LOW);
 digitalWrite(IN4,HIGH);
 analogWrite(ENB,speed_right);
 };

void stopy(){
 //stop
 analogWrite(ENA,0);
 digitalWrite(IN1,LOW);
 digitalWrite(IN2,LOW);
 digitalWrite(IN3,LOW);
 digitalWrite(IN4,LOW);
 analogWrite(ENB,0);
 };
```

Short Algorithm:

1. Initialize the pins and serial communication in the setup function.
2. In the loop function, read incoming serial commands.
3. Depending on the command, call the corresponding function to control the motor.
4. If the electronic braking system is enabled, read the button state and activate the brakes if the button is pressed.
5. Repeat steps 2-4 until new serial commands are received.

Problem Statement:

Sometime the collection is lost and when it's being used for a long time then the battery become hot and run out of charge. The charging process of these batteries are also a big problem as we don't have a 4 battery combined charger. The body of the car is not durable, for this we always have to be very careful, it's a pain for all.

Betterment:

Instead of using Bluetooth if we use Wi-Fi connector then the range will be much more or if we use an antenna with the Bluetooth module then the range may increase.

Project Demonstration:

Fig_3 shows the prototype of the Bluetooth-controlled RC car. The car is powered by DC motors mounted on a chassis and controlled through a mobile application connected via a Bluetooth module. When the user presses directional buttons in the app, control signals are transmitted to the car's microcontroller, which drives the motors accordingly. This allows forward, backward, left, and right movement with real-time response. The system demonstrates how wireless communication and motor driver circuits can be combined to create a low-cost robotic vehicle. Such a design can be further extended for educational robotics and smart vehicle applications.

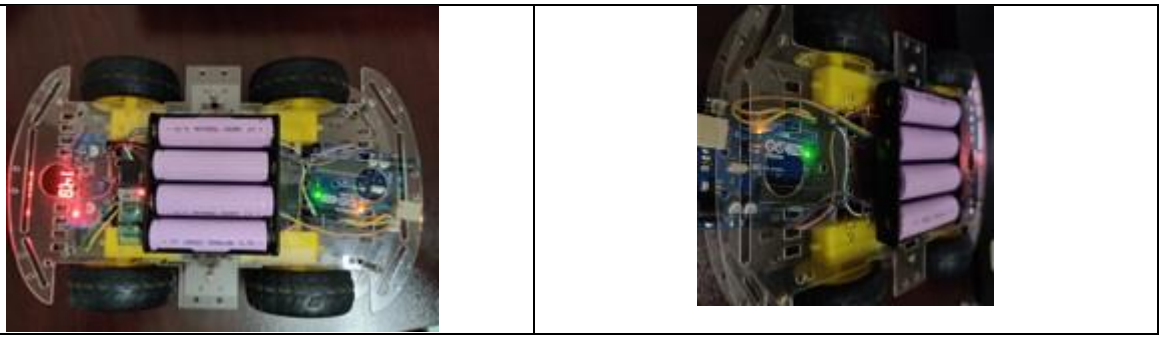

Fig_3: Prototype of RC car

Conclusion:

The Bluetooth-controlled RC car project successfully demonstrates how wireless communication and embedded systems can be applied to build a low-cost robotic vehicle. By using a mobile application and a Bluetooth module, the car can be controlled remotely with reliable response in all directions. The integration of motor driver ICs, chassis, and DC motors provided a practical understanding of circuit design and hardware assembly. This project highlights the potential of Bluetooth-based control systems in developing educational robotics, smart toys, and scalable prototypes for automation.