**PROJECT REPORT**

**FALL - 2022**

**MOBILE CONTROL BLUETOOTH CAR**

***AN ADVANCE CAR***

***PREPARED BY: ISSMAH KHAN***

***COURSE: EMBEDDED SYSTEMS & CCN***

|  |  |
| --- | --- |
| S.NO | CONTENT |
| 1 | Table of Content |
| 2 | Introduction |
| 3 | CCN Concept Use in Project |
| 4 | Embedded Concept Use in Project |
| 5 | Feature Of Project |
| 6 | Topology |
| 7 | Code |
| 8 | Tools Component Used |
| 9 | Working |
| 10 | Results |
| 11 | Conclusion |
| 12 | References |

1. **INTRODUCTION**

Arduino Bluetooth control vehicle is a straightforward robot vehicle that can be constrained by your cell phone. This Smartphone gives a Bluetooth sign to the vehicle and from the sign, the vehicle works. For running the vehicle remotely we are utilizing the HC-05 Bluetooth module.

1. **CCN CONCEPTS USED IN PROJECT**

* Bluetooth module connected with Arduino uno, and networking can be done by connecting it to an application (Bluetooth RC car application).
* Communication Protocols: Depending on the complexity of the project, communication protocols like UART, I2C, or SPI may be used to facilitate communication between the Arduino and external modules or sensors.

<https://play.google.com/store/apps/details?id=braulio.calle.bluetoothRCcontroller&hl=en&gl=US&pli=1>

1. **EMBEDDED CONCEPTS USED IN PROJECT**

In an Arduino-based car project, various embedded concepts are utilized to create a functional and autonomous vehicle. Here are some of the key embedded concepts commonly employed:

* Microcontroller (Arduino): The heart of the project is the Arduino microcontroller, which acts as the brain of the car. It processes data from sensors and implements control algorithms to drive the motors, enabling the car to move, avoid obstacles, and perform other tasks.
* Sensors: Different types of sensors are integrated into the car to perceive its surroundings. For example, ultrasonic sensors detect obstacles, infrared sensors can be used for line following, and gyroscopes and accelerometers provide orientation and motion data.
* Actuators: Actuators are components responsible for carrying out physical actions. In this project, motors serve as actuators to control the car's wheels and allow it to move forward, backward, and turn.
* Motor Control: Embedded systems manage the speed and direction of the motors to achieve precise movements. Control algorithms are programmed on the Arduino to ensure smooth navigation and obstacle avoidance.
* PWM (Pulse Width Modulation): PWM signals are commonly used to control the speed of the motors. By varying the pulse width, the motors can be driven at different speeds, allowing the car to adjust its velocity and navigate through tight spaces.
* PID Control: Proportional-Integral-Derivative (PID) control is a feedback control mechanism used to maintain the car's stability and follow a desired path. It helps the car make corrections based on sensor feedback and minimizes error in positioning.
* Power Management: Embedded systems must efficiently manage power consumption to prolong battery life and ensure the car's longevity during operation. Proper power management techniques are essential for the car's sustained performance.

1. **FEATURES OF YOUR PROJECT**

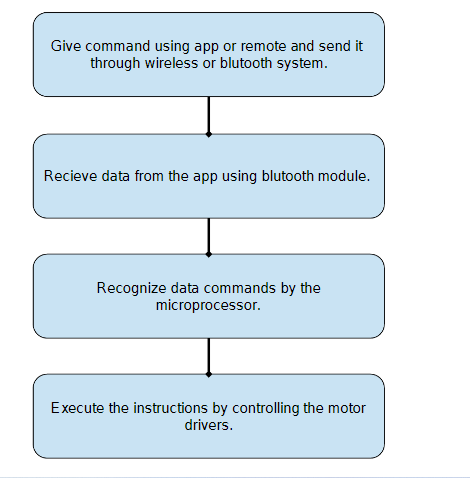
Bluetooth can connect with a device by having a stable and supported connection with an application. After a stable connection, an led will glow. Connection is secure. Backlight glows after connection.First, in the Android App, I have used 5 keys as Forward, Reverse, Left, Right and Stop. The corresponding data associated with each key is as follows:

* Forward
* Reverse
* Left
* Right
* Stop

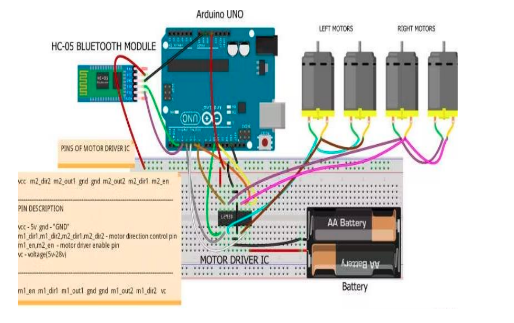
When a key is pressed, the corresponding data is transmitted to the Bluetooth Module from the Phone over Bluetooth Communication.

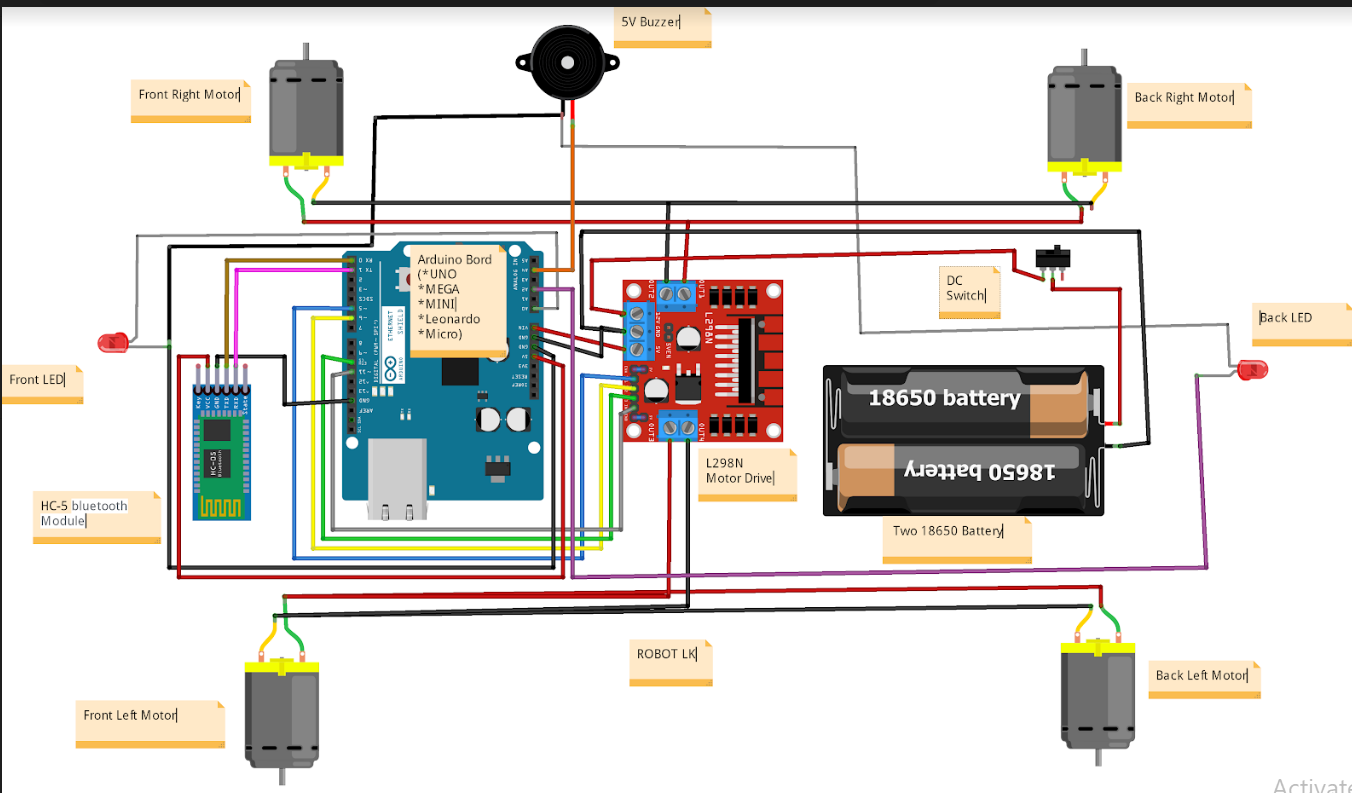
1. **DIAGRAMS**

**BLOCK DIAGRAM / FLOWCHART:**



**PIN DIAGRAM WITH CONNECTIONS & TOPOLOGY**





1. **CODE**

**#define in1 5 //L298n Motor Driver pins.**

**#define in2 6**

**#define in3 10**

**#define in4 11**

**#define light\_FR 14 //LED Front Right pin A0 for Arduino Uno**

**#define light\_FL 15 //LED Front Left pin A1 for Arduino Uno**

**#define light\_BR 16 //LED Back Right pin A2 for Arduino Uno**

**#define light\_BL 17 //LED Back Left pin A3 for Arduino Uno**

**#define horn\_Buzz 18 //Horn Buzzer pin A4 for Arduino Uno**

**int command; //Int to store app command state.**

**int Speed = 204; // 0 - 255.**

**int Speedsec;**

**int buttonState = 0;**

**int lastButtonState = 0;**

**int Turnradius = 0; //Set the radius of a turn, 0 - 255 Note:the robot will malfunction if this is higher than int Speed.**

**int brakeTime = 45;**

**int brkonoff = 1; //1 for the electronic braking system, 0 for normal.**

**boolean lightFront = false;**

**boolean lightBack = false;**

**boolean horn = false;**

**void setup() {**

**pinMode(in1, OUTPUT);**

**pinMode(in2, OUTPUT);**

**pinMode(in3, OUTPUT);**

**pinMode(in4, OUTPUT);**

**pinMode(light\_FR, OUTPUT);**

**pinMode(light\_FL, OUTPUT);**

**pinMode(light\_BR, OUTPUT);**

**pinMode(light\_BL, OUTPUT);**

**pinMode(horn\_Buzz, OUTPUT);**

**Serial.begin(9600); //Set the baud rate to your Bluetooth module.**

**}**

**void loop() {**

**if (Serial.available() > 0) {**

**command = Serial.read();**

**Stop(); //Initialize with motors stoped.**

**if (lightFront) {digitalWrite(light\_FR, HIGH); digitalWrite(light\_FL, HIGH);}**

**if (!lightFront) {digitalWrite(light\_FR, LOW); digitalWrite(light\_FL, LOW);}**

**if (lightBack) {digitalWrite(light\_BR, HIGH); digitalWrite(light\_BL, HIGH);}**

**if (!lightBack) {digitalWrite(light\_BR, LOW); digitalWrite(light\_BL, LOW);}**

**if (horn) {digitalWrite(horn\_Buzz, HIGH);}**

**if (!horn) {digitalWrite(horn\_Buzz, LOW);}**

**switch (command) {**

**case 'F':**

**forward();**

**break;**

**case 'B':**

**back();**

**break;**

**case 'L':**

**left();**

**break;**

**case 'R':**

**right();**

**break;**

**case 'G':**

**forwardleft();**

**break;**

**case 'I':**

**forwardright();**

**break;**

**case 'H':**

**backleft();**

**break;**

**case 'J':**

**backright();**

**break;**

**case '0':**

**Speed = 100;**

**break;**

**case '1':**

**Speed = 140;**

**break;**

**case '2':**

**Speed = 153;**

**break;**

**case '3':**

**Speed = 165;**

**break;**

**case '4':**

**Speed = 178;**

**break;**

**case '5':**

**Speed = 191;**

**break;**

**case '6':**

**Speed = 204;**

**break;**

**case '7':**

**Speed = 216;**

**break;**

**case '8':**

**Speed = 229;**

**break;**

**case '9':**

**Speed = 242;**

**break;**

**case 'q':**

**Speed = 255;**

**break;**

**case 'W':lightFront = true;break;**

**case 'w':lightFront = false;break;**

**case 'U':lightBack = true;break;**

**case 'u':lightBack = false;break;**

**case 'V':horn = true;break;**

**case 'v':horn = false;break;**

**}**

**Speedsec = Turnradius;**

**if (brkonoff == 1) {**

**brakeOn();**

**} else {**

**brakeOff();**

**}**

**}**

**}**

**void forward() {**

**analogWrite(in1, Speed);**

**analogWrite(in3, Speed);**

**}**

**void back() {**

**analogWrite(in2, Speed);**

**analogWrite(in4, Speed);**

**}**

**void left() {**

**analogWrite(in3, Speed);**

**analogWrite(in2, Speed);**

**}**

**void right() {**

**analogWrite(in4, Speed);**

**analogWrite(in1, Speed);**

**}**

**void forwardleft() {**

**analogWrite(in1, Speedsec);**

**analogWrite(in3, Speed);**

**}**

**void forwardright() {**

**analogWrite(in1, Speed);**

**analogWrite(in3, Speedsec);**

**}**

**void backright() {**

**analogWrite(in2, Speed);**

**analogWrite(in4, Speedsec);**

**}**

**void backleft() {**

**analogWrite(in2, Speedsec);**

**analogWrite(in4, Speed);**

**}**

**void Stop() {**

**analogWrite(in1, 0);**

**analogWrite(in2, 0);**

**analogWrite(in3, 0);**

**analogWrite(in4, 0);**

**}**

**void brakeOn() {**

**//Here's the future use: an electronic braking system!**

**// read the pushbutton input pin:**

**buttonState = command;**

**// compare the buttonState to its previous state**

**if (buttonState != lastButtonState) {**

**// if the state has changed, increment the counter**

**if (buttonState == 'S') {**

**if (lastButtonState != buttonState) {**

**digitalWrite(in1, HIGH);**

**digitalWrite(in2, HIGH);**

**digitalWrite(in3, HIGH);**

**digitalWrite(in4, HIGH);**

**delay(brakeTime);**

**Stop();**

**}**

**}**

**// save the current state as the last state,**

**//for next time through the loop**

**lastButtonState = buttonState;**

**}**

**}**

**void brakeOff() {**

**}**

**8. TOOLS / COMPONENTS Used:**

1) Arduino Uno

2) L298N Motor Drive

3) Wheels (4x)

4) TT Gear Motor (4x)

5) HC-05 or HC-06 Bluetooth Module

7) 3x18650 Battery Holder

8) Male and Female Jumper wire/TT Wire

9) Two LED (Front and Back Light)

10) 5V Buzzer (car horn)

**9. WORKING ( Description of Technique / Algorithm used in project):**

Assemble the robot, make the necessary connections and upload the code to Arduino. If you understood the HC-05 Bluetooth Module tutorial, then understanding the Bluetooth Controlled Robot project is very easy.

First, in the Android App, I have used 5 keys as Forward, Reverse, Left, Right and Stop. The corresponding data associated with each key is as follows:

* Forward
* Reverse
* Left
* Right
* Stop

When a key is pressed, the corresponding data is transmitted to the Bluetooth Module from the Phone over Bluetooth Communication.

In the Arduino code, the Arduino UNO receives any of this data from the Bluetooth Module (as per the key pressed) and performs a simple switch case operation, where each case associated with appropriate instructions to the Motor Driver Input Pins.

For example, if ‘Forward’ key is pressed in the Android Phone, then ‘1’ is transmitted. Arduino will then make IN1 and IN3 as HIGH and IN2 and IN4 as LOW to achieve a forward motion.

Similarly, other keys correspond to appropriate setting of IN1 – IN4 pins.

#### **10. RESULT (Output / Graph / Display):**



A video description is provided in the link below to see the car’s operation:

#### <https://drive.google.com/file/d/1Ghxb3jVM4nGBgRd438W9Kl4hkEwe34IP/view?usp=sharing>

#### 

#### **11. CONCLUSION:**

This project consists of a basic prototype of a Bluetooth control car above stated. The prototype car can recognize commands from users and can turn the car left, right, and stop with great accuracy. It can be further improved by using different sensors like ultrasonic or infrared and with various levels of coding.

Our Bluetooth control car has a range of 10-20 meter with the mobile Bluetooth controlling system. The range mostly depends on the receivers transmission level.

#### **12. REFERENCES:**

<https://projecthub.arduino.cc/samanfern/c71cd04b-79fd-4d0a-8a4b-b1dacc2f7725>

<https://www.electronicshub.org/bluetooth-controlled-robot-using-arduino/>

<https://www.researchgate.net/publication/361312318_Bluetooth_Control_Car_with_Arduino>