



Gurukul Technicals Institute

Computer education · Robotics · Science projects · Delhi

FREE EBOOK

ROBOTICS AND ELECTRONICS

Build a Bluetooth-Controlled Car

Drive a robot car from a phone with an HC-05 module and Arduino

INSIDE THIS BOOK

- How Bluetooth serial works
- The HC-05 module and its voltage divider
- Parts, tools and safety
- Building and wiring the car
- Pairing your phone and choosing an app
- ...and 5 more chapters, practice questions and a glossary

Intermediate

Students of class 8 to 12 and hobbyists

engineergurukul.com

+91 97116 99511

Build a Bluetooth-Controlled Car

Build a robot car and drive it from an Android phone over Bluetooth. You will learn how Bluetooth serial works, wire an HC-05 module safely with a voltage divider, pair a phone, design command characters, write the complete Arduino code with speed control, headlights and a horn, and fix common problems.

WHO IT IS FOR	Students of class 8 to 12 and hobbyists
LEVEL	Intermediate
SUBJECT AREA	Robotics and electronics
WHAT IS INSIDE	10 chapters, practice questions with answers, and a glossary

HOW TO USE IT

Read one chapter at a time and try the examples as you go: on a computer, at your workbench or in your notebook. The boxes give tips, safety points and short activities. When you finish, test yourself with the practice questions and check your answers at the end. Words you may not know are explained in the glossary.



Gurukul Technicals Institute

A-10, Chief Commissioner Colony, opposite Rana Pratap Bagh, Delhi 110007

Phone 011 2721 1512 · Mobile and WhatsApp +91 97116 99511

engineergurukul@yahoo.co.in · engineergurukul.com

Free ebook. You may read, print and share this ebook for learning. It may not be sold or changed.

© 2026 Gurukul Technicals Institute. First edition, October 2026. The information was checked when it was written; software, prices and rules change, so for anything official always check the latest source.

What you will find inside

1	How Bluetooth serial works	4
2	The HC-05 module and its voltage divider	6
3	Parts, tools and safety	8
4	Building and wiring the car	9
5	Pairing your phone and choosing an app	11
6	Command characters	13
7	The complete Arduino code	15
8	Speed control, lights and horn	19
9	Troubleshooting	21
10	Driving safely and presenting your car	23
	Practice questions	25
	Answers	26
	Glossary	27

How Bluetooth serial works

A Bluetooth car is a robot car that you drive from your phone. Tap a button on the phone, and a single letter, such as F for forward, flies through the air to the car. The car's Arduino reads the letter and runs its motors. To understand how, you need two ideas: serial communication and Bluetooth.

Serial communication

Serial means one after another. In serial communication, data travels along a wire one **bit** (a single 0 or 1) at a time. Every letter you send is stored as a number, and each of these numbers fits in 8 bits.

A serial connection needs two signal wires, one for each direction:

- **TX** (transmit) is the pin that sends data.
- **RX** (receive) is the pin that listens.

One device's TX always connects to the other device's RX. Think of a phone call: your mouth must reach the other person's ear, not their mouth.

Letters are numbers

Computers store each character as a number, using a standard code called **ASCII**. For example, capital F is 70, capital S is 83, and the digit character '0' is 48. When your phone sends "F", it really sends the number 70 as 8 bits.

Baud rate

Both ends must agree on how fast to send bits, or the message turns into rubbish. This speed is the **baud rate**, in bits per second. The most common rate for our module is 9600. Each character takes 10 bits (8 data bits plus a start bit and a stop bit), so 9600 baud carries about 960 characters a second. One command letter takes about a thousandth of a second.

Bluetooth: a wireless cable

Bluetooth is a short-range radio system that phones, earphones and speakers use. It works in the 2.4 GHz radio band, the same band that Wi-Fi uses. The HC-05 module uses a part of Bluetooth called the **Serial Port Profile**. It makes the radio link behave like an invisible serial cable: whatever the phone sends comes out of the module's TX pin, as if a wire joined them.

Step	What happens
1	You tap "Forward" in the phone app
2	The app sends the character F over Bluetooth
3	The HC-05 receives it and sends F out of its TXD pin, at 9600 baud
4	The Arduino reads F on its RX pin
5	The code sets both motors forward

REMEMBER

TX goes to RX, and RX goes to TX. Connecting TX to TX is the most common wiring mistake in serial projects.

DID YOU KNOW

The HC-05 uses the older "Classic" Bluetooth. Most Android phones can talk to it, but iPhones do not connect to it through ordinary apps. For an iPhone you would need a different type of module, which this book does not cover.

TRY THIS

Open the Arduino IDE's Serial Monitor with any sketch that prints text, and change the baud rate at the bottom to the wrong value. The text turns into strange symbols. That is what a baud-rate mismatch looks like.

The HC-05 module and its voltage divider

The **HC-05** is a small board, usually blue, with a Bluetooth module on it. It is cheap, popular in school projects, and easy to use, because to the Arduino it is just a serial device.

The pins

Most HC-05 boards have six pins, labelled on the back.

Pin	What it does	In this project
EN (or KEY)	Puts the module into a setup mode	Not connected
VCC	Power in	Arduino 5V
GND	Ground	Arduino GND
TXD	Sends data out of the module	Arduino pin 10
RXD	Receives data into the module	Arduino pin 11, through a voltage divider
STATE	Shows whether a phone is connected	Not connected

The board has a small regulator, so its VCC pin takes 5 V. But the Bluetooth chip itself works at 3.3 V, and its **RXD pin expects about 3.3 V signals**. An Arduino Uno's pins send 5 V signals, so connecting Arduino TX straight to RXD pushes too much voltage into the module and can damage it over time.

The other direction is fine. The HC-05's TXD sends 3.3 V signals, and 3.3 V is high enough for the Uno to read as HIGH. So only the RXD line needs care.

The voltage divider

A **voltage divider** is two resistors in a row that give out a fraction of the voltage put in.

1. Connect a 1 k Ω resistor from Arduino pin 11 to the HC-05's RXD pin.

2. Connect a 2 kΩ resistor from the HC-05's RXD pin to GND.

The RXD pin now sits at the joint between the two resistors. The formula is:

$$\text{Output voltage} = \text{input voltage} \times R2 \div (R1 + R2)$$

Here R1 is the resistor from the Arduino (1 kΩ) and R2 is the resistor to ground (2 kΩ).

EXAMPLE

Output = $5\text{ V} \times 2 \div (1 + 2) = 5 \times 2 \div 3 \approx 3.33\text{ V}$. That is just right for the HC-05.

If you do not have a 2 kΩ resistor, other pairs work too:

R1 (from Arduino)	R2 (to GND)	Output from 5 V
1 kΩ	2 kΩ	about 3.33 V
1 kΩ	2.2 kΩ	about 3.44 V
4.7 kΩ	10 kΩ	about 3.40 V
1 kΩ	two 1 kΩ in series	about 3.33 V

The LED tells you its state

The HC-05's LED usually blinks fast when it is waiting for a connection, and changes to a slower pattern (often two quick blinks, then a pause) once a phone has connected. The exact pattern varies between boards, so watch yours and learn it.

TIP

Read the resistor colour bands carefully. A 1 kΩ resistor is brown, black, red. A 2 kΩ resistor is red, black, red. Swap them and RXD gets only about 1.7 V, which may be too low for the module to read.

DID YOU KNOW

The HC-06 is a close cousin with four pins. It works with the same wiring and the same code for this car.

Parts, tools and safety

Collect everything before starting, and check it against this list.

Parts list

Part	Quantity	What it does
Arduino Uno with USB cable	1	The brain
HC-05 Bluetooth module	1	Receives commands from the phone
L298N motor driver module	1	Powers and controls the motors
BO motors (150 to 300 RPM) with wheels	2	Drive the car
Ball caster	1	Supports the front
2WD chassis plate with motor clamps and screws	1	The body
Battery holder for 2 × 18650 Li-ion cells, with cells, or a 6 × AA holder	1	Power
On/off switch	1	Cuts the battery safely
Resistors: 1 k Ω and 2 k Ω	1 each	The voltage divider
White LEDs and 220 Ω resistors	2 each	Headlights
Small 5 V active buzzer (under 20 mA)	1	The horn
Mini breadboard	1	Holds the divider, LEDs and shared power
Jumper wires	about 25	Connections
An Android phone	1	The remote control

A 4WD chassis with four motors also works. Connect the two left motors together to OUT1 and OUT2, and the two right motors together to OUT3 and OUT4. Four motors draw more current, so use strong batteries.

Tools

- Small screwdrivers, a wire stripper and pliers
- A multimeter
- A computer with the Arduino IDE (free from [arduino.cc](https://www.arduino.cc))

SAFETY

Every part of this car runs on batteries of 12 V or less. Never connect any part to mains. Li-ion cells need a proper Li-ion charger and an adult's supervision, and must never be shorted, crushed or left charging unattended. Younger students should use the 6 × AA holder. Soldering, hot glue and cutting need an adult.

EXAMPLE

Vikram's first car ran on a single 9 V PP3 battery. It crawled, and the Bluetooth link kept dropping as the voltage sagged. With six AA cells it drove smoothly. A PP3 battery cannot supply the current motors need.

TIP

Write "1k" and "2k" on two small pieces of masking tape and stick them on the resistors' legs. You will thank yourself during wiring.

CHAPTER 4

Building and wiring the car

Building the chassis

1. Peel the protective paper off the acrylic plate.
2. Clamp the two BO motors in place with their long screws, shafts pointing outward.
3. If the motors have no wires, ask an adult to solder two wires to each.

4. Fit the caster at the other end, using spacers so the plate sits level.
5. Fix the battery holder low and near the middle.
6. Mount the Arduino on spacers, the L298N near the motors, and the mini breadboard where wires can reach it.
7. Put the switch in the battery's positive wire, and push on the wheels.

The caster end is the **front**. Fit the headlight LEDs there, facing forward.

Wiring table

Work with the battery off and the USB unplugged. Tick each row as you go.

From	To
HC-05 VCC	Uno 5V (through the breadboard rail)
HC-05 GND	Uno GND (through the breadboard rail)
HC-05 TXD	Uno pin 10
Uno pin 11	1 k Ω resistor, then HC-05 RXD
HC-05 RXD	2 k Ω resistor, then GND
L298N ENA (jumper cap removed)	Uno pin 5
L298N IN1, IN2	Uno pins 7, 8
L298N IN3, IN4	Uno pins 4, 3
L298N ENB (jumper cap removed)	Uno pin 6
Left motor	L298N OUT1 and OUT2
Right motor	L298N OUT3 and OUT4
Uno pin 12	Each headlight: 220 Ω resistor, LED long leg; LED short leg to GND
Uno A0	Buzzer positive leg; buzzer negative leg to GND
Battery positive	Switch, then L298N 12V terminal and Uno VIN
Battery negative	L298N GND terminal

From	To
Uno GND	L298N GND terminal

Leave the L298N's 5V terminal empty, and keep its regulator jumper beside the power terminals in place. Under each ENA and ENB cap, one pin is 5 V and the other is the real enable input: check the board's label, or measure each to GND with a multimeter and use the pin that does **not** show 5 V.

Why these pins?

- **Pins 10 and 11** carry the Bluetooth serial link, using a library called SoftwareSerial. This keeps pins 0 and 1 free for uploading code.
- **Pins 5 and 6** give PWM, for speed control of the two motors.
- **A0** works as a normal digital pin for the horn. All the Uno's analog pins can do this.
- The battery feeds **VIN**, so the Uno's own regulator makes a steady 5 V. VIN works best with 7 to 12 V, which a 2 × 18650 pack (7.4 V) or six AA cells give.

REMEMBER

Connect Uno GND to the L298N GND. This **common ground** lets the driver understand the Arduino's signals.

SAFETY

Check the battery polarity twice before switching on. A reversed battery can destroy the driver and the Arduino in a moment.

CHAPTER 5

Pairing your phone and choosing an app

Before the car can obey your phone, the two must be introduced. This takes two separate steps: pairing and connecting.

- **Pairing** happens once. The phone and the module exchange a PIN and remember each other.

- **Connecting** happens every time you want to drive. The app opens the link to the paired module.

Pairing step by step

1. Switch the car on. The HC-05's LED should blink fast.
2. On the phone, open Settings, then Bluetooth, and switch Bluetooth on.
3. Choose the option to pair a new device. (The exact words differ between phones.)
4. Wait for **HC-05** to appear in the list. At first it may show only a code of letters and numbers; wait a few seconds for the name.
5. Tap it and enter the PIN. On most modules it is **1234**; on some it is **0000**.
6. The phone now lists HC-05 as paired. The LED still blinks fast, because paired is not yet connected.

Choosing an app

You need an Android app that can send characters over Bluetooth to a module like the HC-05. Search the official app store for a "Bluetooth serial terminal" or a "Bluetooth RC car controller". There are many, and they change often, so this book does not name one. Look for these features:

Feature	Why it matters
Connects to Bluetooth Classic modules such as the HC-05	Some apps only work with newer Bluetooth Low Energy devices
Lets you send single characters	Our car listens for single letters
Buttons you can set to send a chosen character (controller apps)	So each button sends our command letters
A setting for line endings	Choose "none" if you can; our code ignores extra characters anyway

A **terminal app** shows a text box: type F and press Send. A **controller app** shows arrow buttons on screen. In most controller apps you can open the settings and choose which character each button sends. Some send the character once when you tap; others keep sending it while you hold the button. Find out which kind yours is.

Connecting

1. Open the app. When it asks for permission to find nearby devices, allow it, or it cannot see the module.
2. Choose HC-05 from the app's device list and tap Connect.
3. The LED changes to its slower "connected" pattern.

TIP

At an exhibition, several HC-05 modules may all be called "HC-05". Before the event, note the code of letters and numbers your phone shows for your module, so you connect to the right car. Many modules can also be renamed using special "AT commands", but the exact commands differ between module versions, so follow the datasheet for yours.

SAFETY

Only install apps from the official app store, and do not give an app permissions it does not need, such as your contacts.

CHAPTER 6

Command characters

The phone and the car need a shared language. Ours is a set of single characters. Each one stands for one action.

The command table

Character	Action
F	Drive forward
B	Drive backward
L	Spin left on the spot
R	Spin right on the spot
S	Stop

Character	Action
0 to 9	Set the speed level: 0 is slowest, 9 is full speed
T	Turn the headlights on, or off if they are on (T for torch)
H	Sound the horn for a short beep

Capital letters are the commands. The code converts small letters to capitals, so f works the same as F.

Why single characters?

- **Quick.** One character takes about a thousandth of a second at 9600 baud.
- **Simple.** The Arduino reads one character and acts. There is no word to collect and check.
- **Easy to type.** In a terminal app, you can drive by typing F, S, B.

What the speed digits do

The digit you send is a **level** from 0 to 9. The Arduino turns it into a PWM value between 90 and 255 using `map()`. PWM (pulse width modulation) switches the motor on and off very fast; a bigger number means more on-time and more speed. The lowest level is 90, not 0, because motors often hum without moving at very low values.

Level	PWM value
0	90
3	145
5	181
7	218
9	255

EXAMPLE

`map(5, 0, 9, 90, 255)` works out $(5 - 0) \times (255 - 90) \div (9 - 0) + 90 = 5 \times 165 \div 9 + 90$. The Arduino does whole-number division, so $825 \div 9$ gives 91, and $91 + 90 = 181$.

Setting up a controller app

If your controller app lets you choose characters, set:

Button	Sends
Up arrow	F
Down arrow	B
Left arrow	L
Right arrow	R
Middle or release	S
Extra buttons	T and H

If the app has a speed slider that sends digits 0 to 9, it will work as it is. If it sends something else, set spare buttons to send digits instead.

REMEMBER

The Arduino ignores any character that is not in the table. So newline characters from a terminal app, or extra letters from a controller, cannot confuse the car.

TRY THIS

Design your own extra command, such as D for a little dance: left, right, left, stop. Write it in your command table first, then add it to the code after the next chapter.

CHAPTER 7

The complete Arduino code

Here is the whole program, using **SoftwareSerial** for the Bluetooth link.

```

// Bluetooth car: HC-05, L298N and Arduino Uno
#include <SoftwareSerial.h>

SoftwareSerial bt(10, 11);  // RX, TX as seen by the Arduino

// ----- Pins -----
const int ENA = 5;  // left motor speed (PWM)
const int IN1 = 7;  // left motor direction
const int IN2 = 8;
const int ENB = 6;  // right motor speed (PWM)
const int IN3 = 4;  // right motor direction
const int IN4 = 3;
const int HEADLIGHTS = 12;
const int HORN = A0;

// ----- What the car is doing -----
int carSpeed = 181;  // PWM value, 0 to 255 (level 5)
char motion = 'S';  // F, B, L, R or S
bool lightsOn = false;
bool hornOn = false;
unsigned long hornStart = 0;
const unsigned long HORN_TIME = 300;  // beep length in ms

void setup() {
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(HEADLIGHTS, OUTPUT);
  pinMode(HORN, OUTPUT);
  drive(0, 0);
  Serial.begin(9600);  // USB link, for checking on the computer
  bt.begin(9600);  // HC-05 link
  Serial.println("Bluetooth car ready");
}

void loop() {
  if (bt.available() > 0) {
    char c = bt.read();
    Serial.print("Got: ");
    Serial.println(c);
    handleCommand(c);
  }
  // Switch the horn off when its time is up
  if (hornOn && millis() - hornStart >= HORN_TIME) {
    digitalWrite(HORN, LOW);
  }
}

```

```

    hornOn = false;
}
}

void handleCommand(char c) {
    c = toupper(c); // f works like F
    if (c == 'F' || c == 'B' || c == 'L' || c == 'R' || c == 'S') {
        motion = c;
        applyMotion();
    } else if (c >= '0' && c <= '9') {
        carSpeed = map(c - '0', 0, 9, 90, 255);
        applyMotion(); // use the new speed now
        bt.print("Speed ");
        bt.println(carSpeed);
    } else if (c == 'T') {
        lightsOn = !lightsOn;
        digitalWrite(HEADLIGHTS, lightsOn ? HIGH : LOW);
    } else if (c == 'H') {
        digitalWrite(HORN, HIGH);
        hornOn = true;
        hornStart = millis();
    }
    // Any other character is ignored
}

void applyMotion() {
    switch (motion) {
        case 'F': drive(carSpeed, carSpeed); break;
        case 'B': drive(-carSpeed, -carSpeed); break;
        case 'L': drive(-carSpeed, carSpeed); break;
        case 'R': drive(carSpeed, -carSpeed); break;
        default: drive(0, 0); break;
    }
}

// Positive speed = forward, negative = backward, 0 = stop
void drive(int leftSpeed, int rightSpeed) {
    setMotor(ENA, IN1, IN2, leftSpeed);
    setMotor(ENB, IN3, IN4, rightSpeed);
}

void setMotor(int enPin, int inA, int inB, int speed) {
    if (speed > 0) {
        digitalWrite(inA, HIGH);
        digitalWrite(inB, LOW);
    } else if (speed < 0) {
        digitalWrite(inA, LOW);
        digitalWrite(inB, HIGH);
        speed = -speed;
    } else {

```

```

    digitalWrite(inA, LOW);
    digitalWrite(inB, LOW);
  }
  analogWrite(enPin, constrain(speed, 0, 255));
}

```

How the code works

- **SoftwareSerial bt(10, 11);** makes a serial port on pins 10 (receive) and 11 (transmit), named **bt**. The real serial port, **Serial**, stays connected to the USB cable.
- **setup()** sets the pins, stops the motors and starts both serial links at 9600 baud.
- **loop()** checks **bt.available()**, which tells how many characters are waiting. If one has arrived, **bt.read()** takes it, the Serial Monitor shows it, and **handleCommand()** acts on it.
- **handleCommand()** first makes the letter a capital with **toupper()**. Movement letters are stored in **motion**. A digit changes **carSpeed**; **c - '0'** turns the character '5' into the number 5, because the ASCII code for '5' is 5 more than the code for '0'. **T** flips the lights; **H** starts the horn.
- **applyMotion()** uses **switch** to pick the right **drive()** call for the stored motion. Calling it again after a speed change makes the car speed up or slow down at once.
- **drive()** and **setMotor()** set each wheel's direction and PWM speed.

TIP

Keep the USB cable connected during your first test, with the wheels lifted. Open the Serial Monitor at 9600 baud. Every command from your phone appears as "Got: F" and so on. If nothing appears, the problem is in the Bluetooth link, not the motors.

Using hardware serial instead

The Uno has one built-in serial port, on pins 0 (RX) and 1 (TX), called **Serial**. You can connect the HC-05 there instead:

1. Wire HC-05 TXD to pin 0, and pin 1 through the voltage divider to HC-05 RXD.
2. In the code, delete the **#include <SoftwareSerial.h>** line and the **SoftwareSerial bt(10, 11);** line.

3. Change every `bt.` to `Serial.`, and delete the line `bt.begin(9600);` so that `Serial.begin(9600);` is the only begin.

There is one big catch. Pins 0 and 1 are also used by the USB connection to upload your code. **Before every upload, unplug the wires from pins 0 and 1**, and plug them back after uploading. If you forget, the upload fails with an error. The "Got:" messages will now go to the phone instead of the Serial Monitor.

	SoftwareSerial (pins 10 and 11)	Hardware serial (pins 0 and 1)
Uploading	Works with the HC-05 connected	Unplug the HC-05 wires every time
Serial Monitor for checking	Free to use	Shared with the HC-05
Reliability at 9600 baud	Good	Very good

CHAPTER 8

Speed control, lights and horn

Your car already has all three features in its code. This chapter explains how each one works, so you can change them with confidence.

Speed control

Send a digit, and the speed changes immediately, even while the car is moving. Start at a low level such as 3 for your first drives indoors. Spin turns use the same speed as straight driving, which can feel very fast. To make turns gentler, change the two turn lines in `applyMotion()` to use `carSpeed * 2 / 3`, for example `drive(-carSpeed * 2 / 3, carSpeed * 2 / 3);`.

EXAMPLE

At level 9, `carSpeed` is 255. Two-thirds of that is $255 \times 2 \div 3 = 170$, so the car spins at about two-thirds power while still driving straight at full power.

Headlights

Each white LED has its own $220\ \Omega$ resistor to limit the current. Without a resistor, an LED draws too much current and burns out, and can damage the Arduino pin.

EXAMPLE

A white LED drops about 3 V. The resistor gets the rest: $5 - 3 = 2$ V. By Ohm's law, $\text{current} = \text{voltage} \div \text{resistance} = 2 \div 220 \approx 0.009$ A, or about 9 mA. Two LEDs on pin 12 draw about 18 mA together, which is within the 20 mA that an Uno pin can comfortably give.

The LED's long leg (positive) goes towards the Arduino pin, through the resistor. The short leg goes to GND. A backwards LED simply stays dark.

The horn

The **active buzzer** makes its own tone when powered, so the Arduino only switches it on and off. Choose a small 5 V buzzer that draws less than 20 mA; a louder one needs a transistor to switch it.

The interesting part is how the code times the beep. It does **not** use `delay(300)`. A delay would freeze the whole program for 300 ms, and the car could not obey a Stop command during that time. Instead the code uses `millis()`, which counts the milliseconds since the Arduino started.

1. When H arrives, the code switches the buzzer on and stores the time in `hornStart`.
2. Every pass through `loop()`, it checks: has 300 ms passed since `hornStart`?
3. When the answer is yes, it switches the buzzer off.

EXAMPLE

H arrives when `millis()` is 12,000, so `hornStart` becomes 12,000. At 12,150, the difference is 150, less than 300, so the horn stays on. At 12,300 the difference reaches 300 and the horn switches off. Meanwhile the car has kept reading commands.

An automatic stop

If your controller app keeps sending a letter while you hold a button, you can make the car stop by itself when the commands stop coming, for example when you let go or the connection drops. Add these two lines above `setup()`:

CPP

```
unsigned long lastCommand = 0;
const unsigned long AUTO_STOP_MS = 600;
```

Inside `loop()`, straight after `char c = bt.read();`, add `lastCommand = millis();`. Then add this at the end of `loop()`:

CPP

```
if (motion != 'S' && millis() - lastCommand > AUTO_STOP_MS) {
    motion = 'S';
    applyMotion();
}
```

Do not use this with an app that sends each letter only once: the car would stop 0.6 seconds after every tap.

TRY THIS

Add red tail lights on pin A1 that switch on whenever `motion` is B. Put the `digitalWrite` for them inside `applyMotion()`.

CHAPTER 9

Troubleshooting

When the car misbehaves, test one part at a time. Keep the wheels lifted and the Serial Monitor open: the "Got:" lines show at once whether the Bluetooth part is working.

Bluetooth problems

Problem	Likely cause	What to do
HC-05 LED is off	No power	Check VCC and GND wires and the battery switch
HC-05 not in the phone's list	Already connected to another phone, or too far away	Disconnect the other phone; come closer
Phone is an iPhone	HC-05 uses Classic Bluetooth serial	Use an Android phone
Pairing fails	Wrong PIN	Try 1234, then 0000
App cannot find the module	Permission refused, or wrong type of app	Allow nearby-devices permission; use an app for Classic modules
Connected, but no "Got:" lines	TXD and RXD swapped	HC-05 TXD must go to pin 10
"Got:" shows strange symbols	Baud-rate mismatch	Try <code>bt.begin(38400);</code> in case your module uses that rate
"Got:" shows other letters	App sends different characters	Change the app's button settings

Car problems

Problem	Likely cause	What to do
F makes the car go backward	Motor wires reversed	Swap both motors' wires at the L298N
L and R are swapped	Left and right motors swapped	Swap the OUT1/OUT2 pair with the OUT3/OUT4 pair
Car spins when it should go straight	One motor reversed	Swap that motor's two wires
Motors hum but do not move	Speed level too low or weak battery	Send a higher digit; charge the battery

Problem	Likely cause	What to do
Speed digits have no effect	ENA or ENB jumper caps still on	Remove the caps and wire the enable pins
Car resets or link drops when starting	Battery voltage dipping (a brown-out)	Charge the batteries; tighten wires
Headlight stays dark	LED backwards	Long leg towards pin 12
Upload fails (hardware serial)	HC-05 still on pins 0 and 1	Unplug those wires during upload

A brown-out

When motors start, they draw a surge of current. A weak battery's voltage dips, and the Arduino or the HC-05 may restart. You will see "Bluetooth car ready" again in the Serial Monitor, and the phone may lose its connection. Use fresh or charged batteries, keep power wires short, and never use a 9 V PP3 battery.

TIP

Check the simple things first: is the battery charged, is the switch on, is the app connected? Most "big" problems turn out to be one of these.

REMEMBER

If the phone disconnects while the car is moving, the car keeps doing its last command. Always be ready to switch the car off by hand.

CHAPTER 10

Driving safely and presenting your car

A remote-controlled car is great fun, and fun is safest with a few rules.

Safety rules

- Drive indoors or in a closed space, never on a road or near stairs.
- Start at a low speed level until you are used to the controls.

- Never drive at people's feet, at pets, or under furniture where you cannot reach the car.
- Remember that the car keeps moving if the link drops. Stay close enough to grab it or switch it off.
- Use batteries of 12 V or less, and never connect anything to mains power.
- Switch the car off when you are not driving, and before changing any wire.
- If the motors or driver get too hot to touch, or anything smells hot, switch off and let it cool.
- Charge Li-ion cells only with a proper charger, under an adult's eye.

EXAMPLE

At a school fair in Patna, Harpreet's team let visitors drive their car around a cardboard track for ! 10 a lap, collecting money for the school library. After 85 laps they had raised ! 850, and learnt to charge a spare battery pack while the other one was in use.

Presenting your car

Make a chart showing the signal path as a row of boxes: phone, Bluetooth, HC-05, Arduino, L298N, motors. Add your command table and the voltage divider calculation. Then practise a one-minute explanation:

"I drive this car from my phone. The app sends single letters, such as F for forward, over Bluetooth. The HC-05 module receives them and passes them to the Arduino through a serial connection at 9600 baud. The Arduino reads each letter and controls the motors through the L298N driver. Digits change the speed using PWM, T switches the headlights and H sounds the horn."

Viva questions and model answers

- 1. What is serial communication?** Sending data one bit at a time, along one wire for each direction.
- 2. Why does the HC-05's TXD go to the Arduino's receive pin?** One device's transmit must reach the other's receive, like a mouth speaking into an ear.
- 3. Why is there a voltage divider on RXD?** The HC-05's RXD expects about 3.3 V. The 1 k Ω and 2 k Ω resistors bring the Arduino's 5 V signal down to about 3.33 V.

4. **What does 9600 baud mean?** 9600 bits per second, which is about 960 characters per second.
5. **What is the difference between pairing and connecting?** Pairing is a one-time introduction with a PIN; connecting opens the link each time you drive.
6. **Why use SoftwareSerial?** It keeps pins 0 and 1 free, so you can upload code without unplugging the module.
7. ****Why does the horn use `millis()` instead of `delay()` ?**** A delay would freeze the program, so the car could not obey a Stop command during the beep.
8. **Why can't you use an iPhone?** The HC-05 uses Classic Bluetooth serial, which iPhones do not offer to ordinary apps.
9. **How does the car turn?** The wheels run in opposite directions, so it spins on the spot.

REMEMBER

When you explain, show the Serial Monitor printing "Got: F" as you tap the phone. Seeing the letters arrive makes the whole idea clear to any visitor.

CHECK YOURSELF

Practice questions

1. What does "serial" mean in serial communication?
2. What do TX and RX stand for, and how are they connected between two devices?
3. What is the ASCII code of capital F?
4. At 9600 baud, about how many characters can be sent each second? Show why.
5. Which Bluetooth profile makes the HC-05 behave like a serial cable?
6. Name the six pins of a typical HC-05 board.
7. Why must the HC-05's RXD pin not receive 5 V signals directly?
8. Why can the HC-05's TXD connect straight to the Arduino?
9. Calculate the output of a divider with $R_1 = 1\text{ k}\Omega$ and $R_2 = 2\text{ k}\Omega$ from 5 V.

10. Calculate the output of a divider with $R1 = 4.7\text{ k}\Omega$ and $R2 = 10\text{ k}\Omega$ from 5 V.
11. What is the difference between pairing and connecting?
12. What is the usual pairing PIN for an HC-05?
13. What does the character T do in our command table?
14. What PWM value does speed level 0 give, and why not 0?
15. What does `bt.available()` tell the program?
16. What does `c - '0'` do when c is the character '7'?
17. Why does the code call `applyMotion()` after a speed change?
18. Using hardware serial, what must you do before every upload, and why?
19. Calculate the current through a white LED with a $220\text{ }\Omega$ resistor from 5 V, if the LED drops 3 V.
20. Why does the horn code use `millis()` rather than `delay()`?
21. The Serial Monitor shows strange symbols for each command. What is the likely cause?
22. What happens if the phone disconnects while the car is moving, and how can you guard against it?

ANSWER KEY

Answers

1. One after another: the bits travel one at a time.
2. Transmit and receive. One device's TX connects to the other's RX, in both directions.
3. 70.
4. About 960. Each character takes 10 bits, and $9600 \div 10 = 960$.
5. The Serial Port Profile.
6. EN (or KEY), VCC, GND, TXD, RXD and STATE.
7. The module's chip works at 3.3 V, so 5 V signals can damage it over time.
8. Its 3.3 V signals are high enough for the Uno to read as HIGH.
9. $5 \times 2 \div (1 + 2) \approx 3.33\text{ V}$.

10. $5 \times 10 \div (4.7 + 10) = 50 \div 14.7 \approx 3.40 \text{ V}$.
11. Pairing is a one-time introduction with a PIN; connecting opens the link each time you use it.
12. 1234 on most modules, or 0000 on some.
13. It turns the headlights on, or off if they are already on.
14. 90, because motors often hum without moving at very low PWM values.
15. How many characters have arrived and are waiting to be read.
16. It turns the character '7' (ASCII 55) into the number 7, because '0' is 48.
17. So the car changes to the new speed at once while it keeps moving.
18. Unplug the wires from pins 0 and 1, because the USB upload uses those pins and the HC-05 would interfere.
19. $(5 - 3) \div 220 \approx 0.009 \text{ A}$, about 9 mA.
20. A delay would freeze the program, so the car could not respond to a Stop during the beep.
21. A baud-rate mismatch between the code and the module.
22. The car keeps doing its last command. Drive at low speed in a closed space, stay ready to switch off, or use the automatic stop with an app that keeps sending commands.

WORD LIST

Glossary

Active buzzer

A buzzer that makes its own tone when power is applied.

ASCII

A standard code that stores each character as a number.

Baud rate

The speed of a serial link, in bits per second.

Bit

The smallest piece of data: a single 0 or 1.

Bluetooth

A short-range radio system for linking devices without wires.

Brown-out

A dip in battery voltage that makes a board restart.

Common ground

A shared GND connection between the boards and the battery.

Connecting

Opening the Bluetooth link to a paired device each time you use it.

HC-05

A Bluetooth module that acts as a wireless serial link.

Line ending

Extra invisible characters some apps add after each message.

millis()

An Arduino function giving the milliseconds since the board started.

Pairing

The one-time introduction of two Bluetooth devices, usually with a PIN.

PWM

Pulse width modulation: switching power on and off fast to control speed.

RX

The receive pin of a serial connection.

Serial communication

Sending data one bit at a time.

Serial Port Profile

The part of Bluetooth that makes a wireless link act like a serial cable.

SoftwareSerial

An Arduino library that makes a serial port on ordinary pins.

TX

The transmit pin of a serial connection.

Voltage divider

Two resistors that give out a fixed fraction of the input voltage.



Gurukul Technicals Institute

Computer education institute in Delhi, with authorised member centres, robotics classes and science projects, working models and DIY kits.

COURSES

Computer & Office Skills · Accounting & Business Software · Programming & Web Development · Design & Digital Skills · Hardware & Networking · Robotics & Coding · Other Programmes

PROJECTS AND KITS

Ready-made school projects · DIY and STEM kits · Robotics and coding · Components and spares · Engineering projects

CONTACT

Head office: A-10, Chief Commissioner Colony, opposite Rana Pratap Bagh, Delhi 110007

Phone: 011 2721 1512 · **Mobile and WhatsApp:** +91 97116 99511

Email: engineergurukul@yahoo.co.in

Website: engineergurukul.com · **Instagram:** [@science_projects_club](https://www.instagram.com/science_projects_club)

More free ebooks for computer courses, robotics and science projects: engineergurukul.com/resources/ebooks/

Free to read, print and share. Not for sale.