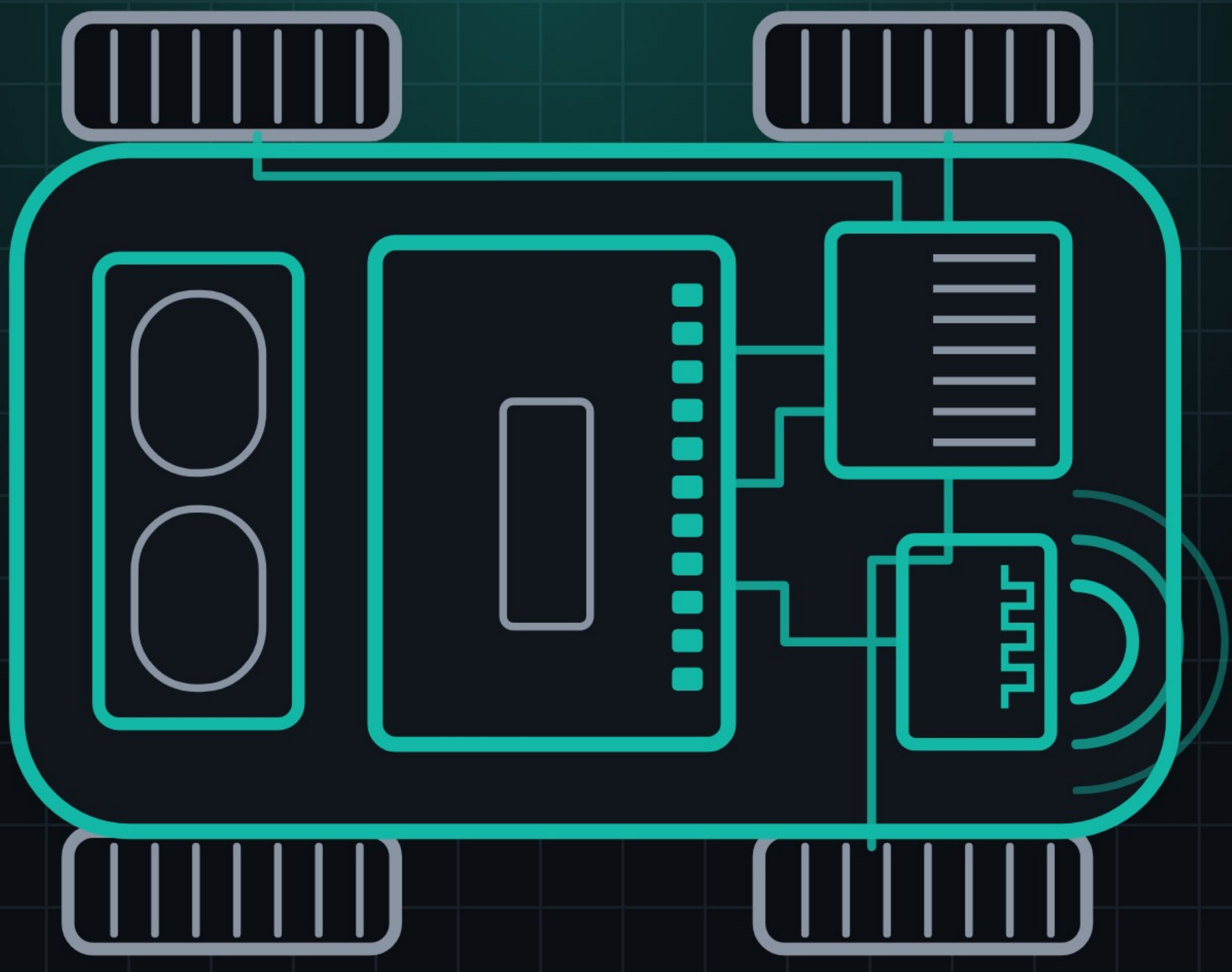
troniction

BUILD IT. UNDERSTAND IT.

SECOND EDITION



/// A PARENT + KID ARDUINO BUILD

Build a Car With Your Kid

A no-soldering Arduino robot car you drive from a phone.

HC-06 / HM-10 · 12+ WITH AN ADULT · KID + PARENT ROLES

/// TRONICTION

Build a Car With Your Kid

A No-Soldering Arduino Robot Car You Drive From a Phone

Troniction

Build a Car With Your Kid

A No-Soldering Arduino Robot Car You Drive From a Phone

© 2026 Troniction. All rights reserved.

Published by Troniction, troniction.com

`Version 2.0.1 · Color screen edition · built 2026-10-01`

Every printed link in this book is a short link, troniction.com/go/..., so it keeps working when the page behind it moves. Report an error: hello@troniction.com.

Contents

How to use this book 7

How this book was tested 9

Part 1 · Build it together 13

1 · Before you decide 14

2 · How to help without taking over 17

3 · The four boxes 22

4 · Unbox and find 27

5 · Safety: read this together 36

6 · Stage A · The board: put the sketch on it 39

7 · Stage B1 · Chassis and motors 45

8 · Stage B2 · Motor driver to the board 52

9 · Stage B3 · Power, and the wheels test 59

10 · Stage C · The Bluetooth module 70

11 · Stage D · The round trip, then drive 79

12 · If it doesn't move 87

13 · What next 93

Part 2 · Why it works 95

14 · Volts, amps and a shared ground 96

15 · The board 102

16 · The motor driver and the H-bridge 109

17 · Speed is a trick: PWM 116

18 · Motors, gears and skid steering 122

19 · Power: cells, Vin and brown-outs 128

20 · The Bluetooth module: how a letter reaches the board and back 135

Part 3 · Find the fault 142

21 · Four boxes and half-splitting 143

22 · Worked example: the speed that never changed 149

23 · Worked examples: spins on the spot, and connected but still 155

24 · Your Serial Monitor is a probe 161

25 · Ten broken cars 165

Part 4 · Program the car 173

26 · Reading a sketch: once, then forever 174

27 · The letter dispatcher 181

28 · Making it move: the motion code 188

29 · The speed ladder 193

30 · Sound, and why the car stops listening while it sings 200

31 · Find it: the bug since 2016 207

32 · Change it safely, then make it yours 212

Part 5 · For parents, clubs and teachers 216

33 · What this build teaches, honestly 217

34 · Running it with 6 to 12 children 221

35 · The one-week check 226

36 · Where next: extensions and resources 231

The full parts list 234

Glossary 237

Colophon and corrections 245

More from Troniction 246

Answers 247

Credits 257

How to use this book

This book is for two people at one table, and sometimes for a third who is only listening.

TABLE 1 Where to start, by who you are

YOU ARE	START WITH	THEN
The parent, before you buy	Chapter 1 (what it costs, what you need)	Chapter 2, before the first session: your job, and the prep list
The parent and child together	Chapters 3 to 5: the four boxes, the parts, safety	Part 1, one stage at a time. The child builds; you read and check
Listening to the Part 1 audio edition	The whole of Part 1, before you touch a wire	Build from this book, which has the pictures: troniction.com/go/car-book
A teacher or club leader	Chapter 33, then Chapter 34	Part 1 with the children in pairs
The child, after the drive	Part 2 (why it works) or Part 3 (find the fault)	Part 4: change the car's own code

What the marks on the page mean

- **Step cards** carry one action each, and a label for who does it: **Child**, **Parent** or **Together**. The child builds; the parent reads each step aloud and checks it against the picture. Only the parent handles the cells and installs the software.
- **Warnings** come before the step that needs them, and again at it.
- **Test boxes** end every stage: what you should see, what you may see instead, what it means and what to do. A stage isn't finished until its test passes, and every session stops after a test.
- **Talk prompts** are questions for the parent to ask. The small "Listen for" line tells you what a good answer sounds like, so you don't need to know it yourself.

- **Questions** end each chapter. Answer out loud first, then check: the answers are at the back of the book (in the audio edition, straight after the questions).
- **Two Bluetooth modules.** Where the HC-06 (Android phones only) and the HM-10 (Android or iPhone) differ, the steps are side by side. Follow your column only. Each column says how it was checked; the next page explains what that means.

Which pages to print

At the bench, the laptop is busy and the book is easier on paper. Print the **bench pack**: Part 1's step cards, test boxes, the pre-power checklist, the role card and the loop card, laid out for a home printer in black and white.

The bench pack, ready to print.

SCAN OR TYPE **troniction.com/go/car-bench**

If you print only a few pages, print the step cards for Stages A and B, the pre-power checklist (Chapter 9) and the loop card (Chapter 3).

The QR codes

Every QR code opens a short video of a hard step, a test page or a printable sheet, and its web address is printed under it, so you can type it instead. **Every page works with the phone off:** the video shows the same step again; it never replaces the page. Nothing in a step sends you to buy anything.

How this book was tested

I don't want you to find out at your kitchen table which parts of this book were tried and which were worked out on paper. So here is exactly what was built, what was checked another way, and what nobody has tested yet.

Every instruction in the book meets one rule: **a step you follow was either built and driven, or checked on a computer and against the part's datasheet.** Anything that meets neither is not a step. Where it appears at all, it is in a box marked "Untested idea".

Built and driven

The reference car is a real car, first built for the 2021 edition of this book and rewired in August 2026. It has a four-wheel chassis with four gear motors, an Arduino Uno board, an L298N motor driver, an HC-06 Bluetooth module, a buzzer and two 18650 cells in a holder. This is what it proves:

- **The motor wiring.** IN1 to pin 12, IN2 to 11, IN3 to 10, IN4 to 9; ENA to pin 5 and ENB to pin 6. The two speed wires were moved to pins 5 and 6 on 15 August 2026 (Chapter 22 tells that story). With the motor driver's two jumper caps off, speed 1 crawls and speed 9 runs (checked 16 August 2026).
- **The power wiring.** The battery's two leads feed the motor driver and the board's Vin and GND, so the board keeps running without the USB cable.
- **No soldering.** Every connection on the car is a screw terminal or a plug-in wire (confirmed 15 August 2026).
- **The HC-06, as it was wired then:** on the board's own serial pins, 0 and 1, driven from an Android phone, with the buzzer on pin 2.
- **The sketch's core:** the letters, the speed ladder, the horn and the tunes, and the bug Chapter 31 asks you to find.

Checked on a computer and against the datasheet, not yet built

These are new in this edition. Each one compiles for the Arduino Uno, its logic passes tests run on a computer that feed it letters and check what it does, and its wiring matches the part's datasheet or official documentation. None has been in the reference car yet.

- **The Bluetooth module on pins 2 and 3** (for both the HC-06 and the HM-10), and the buzzer on pin 4. This is what makes the Serial Monitor useful while the car is connected, and it is why this car has no upload trap (Chapter 32).
- **The two resistors** that protect the module's listening pin, for either module.
- **The HM-10 path:** connecting from Chrome on Android or the Bluefy browser on an iPhone.
- **What the new sketch adds:** `Car ready` and the board's L light staying on at start (Stage A's test), reading letters from both the laptop and the phone, showing each letter as it arrives, and answering `Car heard:` over the Bluetooth module (Stage D's round trip).
- **The switched battery holder and the mini breadboard**, from their listings and datasheets.
- **Every price**, from live US listings on the date printed beside it.
- **Part 4's experiments** were run against the sketch on a computer. The motor patterns are the 2021 car's; the new sketch hasn't driven the motors yet.

Where the two modules differ, the steps are side by side, and each column says which of these two lists it belongs to.

Worked out, not staged

- **The ten broken cars (Chapter 25) and the worked examples (Chapter 23)** were worked out from the circuit, the sketch and the datasheets. Nobody planted those faults on a car and watched what happened. Each answer says how sure it is.

- **The "Expected" lines in Part 2's experiments** are predictions from the datasheets and the sketch. Where your car disagrees, your car is right; tell me.

Not tested yet

- **How long it takes, and how many sessions.** No family has built from this edition yet. The book prints its stopping points instead of a time.
- **Ages 9 to 11**, with a parent reading and checking every step.
- **Running it with a group** (Chapter 34) and **the one-week check** (Chapter 35).
- **Numbers that need a meter or a thermometer on the car**, such as how warm the motor driver's heatsink gets or the voltage at which the board restarts. The book gives the datasheet value, says so, and tells you how to see it on your own car.
- **The upgrades** in Chapter 36. None has been built; none has steps in this edition.
- **The remote page** that drives the HM-10 car, on a real phone. It is built, and it was tested in a browser on a computer with a pretend module standing in for the radio: one letter when you press, `S` when you let go (sent again if the car doesn't confirm it), and the car's answer on the screen. It hasn't yet driven a real HM-10.
- **A reading test.** No parent outside Troniction has read this edition yet. The test is a parent with no electronics background answering "what would you do next?" at each stage. What it finds goes into the corrections.

If you find a mistake

You will be building parts of this before I have. If a step doesn't match your parts, a pin is wrong, a price has moved, or something worked differently, tell me: **troniction.com/go/car**, or email hello@troniction.com. Say which page and what you saw. Every correction is listed, with its date, in the Corrections at the back of the book and at troniction.com/go/car. Nothing is quietly fixed.

The book's home page: the current version, the corrections, and where to report a mistake.

SCAN OR TYPE **troniction.com/go/car**

1

/// PART 1

Build it together

In this part, the child builds the car and the parent reads each step aloud and checks it against the picture. The build happens in four stages. Each stage ends with a test, and the end of a test that passed is a good place to stop for the day.

The first two chapters are for the parent, before you buy anything and before the first session. Then you read three chapters together, and the child's hands go on the parts in Chapter 4.

/// CHAPTER

1

Before you decide

PART OF THE AUDIOBOOK

FOR THE PARENT, BEFORE YOU BUY

Read this chapter on your own. It takes five minutes, and it tells you what you need to know before you spend any money.

What you end up with

A four-wheel car that your child builds from parts and drives from a phone over Bluetooth. It has a horn and a reversing beep, and it can play *Happy Birthday*. Nothing in this book is a second half you have to pay for. The whole build is here, and so is everything about why it works and how to fix it.

Neither of you needs to know anything yet. The child builds. You read each step aloud and check it against the picture.

Before you decide

Cost. All the parts together cost about $100 to $110 at US prices, bought one by one. The full list, with prices and where to buy each part, is at troniction.com/go/car-parts, and Chapter 4 walks you through every part.

Time. The build happens in four stages, with a test at the end of each one. You can spread it over several sessions. Each one can end after a test that passed, so you never stop halfway through something.

Age. The builder should be 12 or older. A 9- to 11-year-old can build it if the parent reads and checks every single step with them. I haven't tested that age yet, so treat it as a reasonable guess, not a promise.

What you need to know first. Nothing. This book assumes neither of you has done any electronics before.

Soldering. None. Every connection in this car is a screw terminal or a wire that pushes onto a pin. The motors come with their wires already attached, so there is no soldering iron and no solder anywhere in the build. One warning about that, in Chapter 4: some motors are sold with bare metal tabs instead of wires, and those would need soldering. Buy the ones the parts list names.

A laptop. You need a Windows or Mac laptop that you're allowed to install software on. A Chromebook, a tablet, or a locked school or work laptop won't do, because the car's program goes onto the board from a free app that has to be installed.

A phone, and the choice of Bluetooth module. The phone drives the car through a small Bluetooth module, and there are two kinds you can buy. Which one depends on your phone.

TABLE 1.1 The two Bluetooth modules. Everything else in the car is the same

	HM-10	HC-06
Phones that can drive the car	iPhone (with the free Bluefy browser) and Android (with Chrome)	**Android only.** An iPhone can't connect to it at all
What you drive it with	The Troniction remote, a web page	A free Android app, Serial Bluetooth Terminal
How this book checked it	The program and its wiring are tested on a computer, and the module's behavior is from its datasheet. Not yet built into a car	The module drove the original 2021 car for years. Its new wiring in this book is tested on a computer

In short: if anyone in your house has an iPhone, buy the HM-10. It works with an iPhone and with an Android phone, and you drive it from a web page. If everyone uses Android, either module works. The HC-06 is the one the original car used, and you drive it from a free Android app.

Everything else in the car is the same, whichever module you choose: the same wiring, the same program, the same tests. Where the two differ, this book shows them side by side.

The one honest warning

One step is fiddly: which way the wheels turn. When you first test the car, the wheels on one side, or all four, might turn backwards. That isn't a mistake. It depends on how your motors were made and mounted, and nobody can tell from a picture. So the book makes it a test: you both guess, you watch, and if any wheels go the wrong way you swap two wires and test again. Nothing breaks.



The full parts list, with prices and where to buy each part.

SCAN OR TYPE **troniction.com/go/car-parts**

/// CHAPTER

2

How to help without taking over

PART OF THE AUDIOBOOK

FOR THE PARENT, BEFORE SESSION 1

Read this chapter on your own, then do the prep list at the end of it before your child touches anything.

Your job is to read and to check

In this build, your child is the **builder** and you are the **checker**. The builder does every wire, every screw and every push-fit. The builder plugs in the board, presses Upload, flips the battery switch, types the test letters and drives. The checker reads each step aloud, compares the result with the picture, ticks the checklist and asks the questions in the “Talk about it” boxes.

Some jobs are done together. The checklist before the power goes on is one: one of you reads each line and the other checks it on the car. The first drive is the other.

And a few jobs are yours alone: installing the software on the laptop, putting the battery cells in and taking them out, and charging them. That isn’t because a 12-year-old couldn’t. It’s because those are the steps where a mistake costs money or needs a grown-up’s judgment.

Every step in this book is marked with who does it: **Child**, **Parent**, or **Together**. Figure 2.1 shows the split.

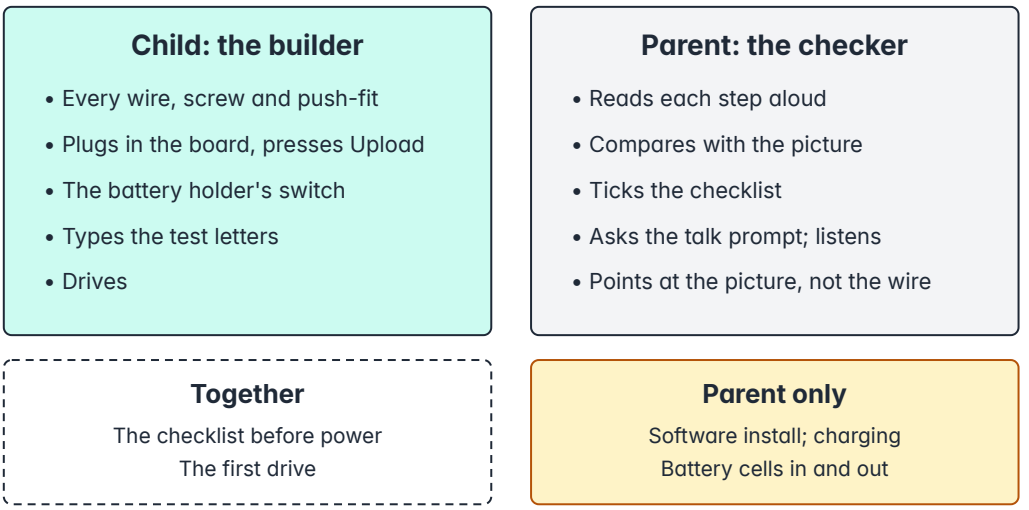


FIGURE 2.1 Who does what. The child builds; the parent reads, checks and owns the battery cells and the software install.

When a wire is wrong

Point at the picture, not at the wire. If your child puts a wire in the wrong hole, don't reach over and move it. Point at the figure in the book and ask which hole the picture shows. The child finds the mistake and fixes it. That moment, when they spot it themselves, is worth more than the minute it costs.

Three lines are worth having ready:

- "I don't know either. Let's check the picture."
- "Show me which pin the picture says."
- "What do you think will happen when you press F?"

The last one matters most. Before each test, the book asks you both to guess what will happen. Write the guess down. A wrong guess that you then check is how both of you learn how the car works.

The talk prompts

At every test, a "Talk about it" box gives you a question to ask. Under each one is a line starting **Listen for**. It tells you what a good answer sounds like, so you can judge the answer without knowing the electronics yourself. Any hone-

st guess is a good start. The point is to get your child saying what they think is happening.

The 10-minute rule

If you've both been stuck on the same thing for 10 minutes, stop. Don't keep trying random changes. Instead, run the debugging loop from Chapter 3: look at exactly what happens, decide which part of the car it belongs to, change one thing, and test again. If that doesn't find it, turn to Chapter 12, "If it doesn't move", or scan the code at that stage for the short video and the test page.

Why it's built this way

In studies of families, children whose parents support them in doing the work themselves, rather than doing it for them, tend to do better and feel better about the work (Vasquez, Patall and colleagues, 2016). And for children around this age, parents simply doing more of the helping has been linked to worse results, not better (Patall and colleagues, 2008). Both come from watching families, not from experiments, so treat them as a sensible direction rather than proof. This book follows it because it costs nothing, and the car gets built either way.

Where to stop: the session plan

You don't have to build the car in one go. There are five good places to stop, and each one comes right after a test that passed:

1. After the Stage A test, when the screen says `Car ready`.
2. After the horn test in the middle of Stage B, when the horn sounds from the laptop.
3. After the wheels test at the end of Stage B, when all four wheels go the right way.
4. After test C1, when the Bluetooth module's light is blinking.
5. After the first drive.

Stopping anywhere else means starting the next session halfway through a step. Stopping after a test means you start the next session knowing that everything so far works.

The prep list, before session 1

Do these on your own, before your child starts. Each one takes a few minutes, and doing them first means session 1 opens with your child's hands on the board, not with you clicking through installers.

1. **Charge the battery cells** in their charger until the charger shows they are full. They usually arrive only partly charged.
2. **Install the Arduino IDE** on the laptop, and check that the board shows up. Chapter 6 starts with the steps, in a section marked for the parent.
3. **Download the car's program**, the sketch, from troniction.com/go/car-sketch. Unzip it somewhere you'll find it again.
4. **Get the phone ready.** HM-10 on an iPhone: install the free Bluefy browser from the App Store. HM-10 on Android: you only need Chrome. HC-06 on Android: install the free Serial Bluetooth Terminal app from Google Play.
5. **Check the laptop** is one you can install software on (Chapter 1).
6. **Print the bench pack**, or at least the step cards for Stages A and B. It's free at troniction.com/go/car-bench, and it means the book can stay closed and clean at the table.



The car's program, the sketch, ready to open in the Arduino IDE.

SCAN OR TYPE **troniction.com/go/car-sketch**



A one-page card to print: who does what, every talk prompt with its "Listen for" line, the debugging loop and the checklist before power.

SCAN OR TYPE **troniction.com/go/car-roles**



The bench pack: every step card, test box and checklist from this part, laid out to print at home in black and white.

SCAN OR TYPE **troniction.com/go/car-bench**

CHECK YOURSELF

Pause here and answer out loud.

1. Your child has put a wire in the wrong pin. What do you do, and what do you say?
2. You've both been stuck for 10 minutes. What now?

Answers at the back of the book on page 247

/// CHAPTER

3

The four boxes

PART OF THE AUDIOBOOK

NOTE

Read this together. From here on, both of you read every chapter.

The journey of one letter

Here is the whole car in one idea. When you press Forward on the phone, the phone sends one letter: a capital F. That letter makes a journey through four boxes, and the car moves.

1. **The phone** sends the letter F through the air, over Bluetooth.
2. **The Bluetooth module**, a small board on the car, catches the letter and passes it along a wire.
3. **The board**, the Arduino Uno, reads the letter. Its program, called the sketch, says what to do with an F: make both sides of the car go forward.
4. **The wheels** turn. This box is two parts working together: the motor driver, which takes power from the battery and pushes it into the motors, and the four motors themselves.

Phone, Bluetooth module, board, wheels. Those four boxes are the whole car, and Figure 3.1 draws them.

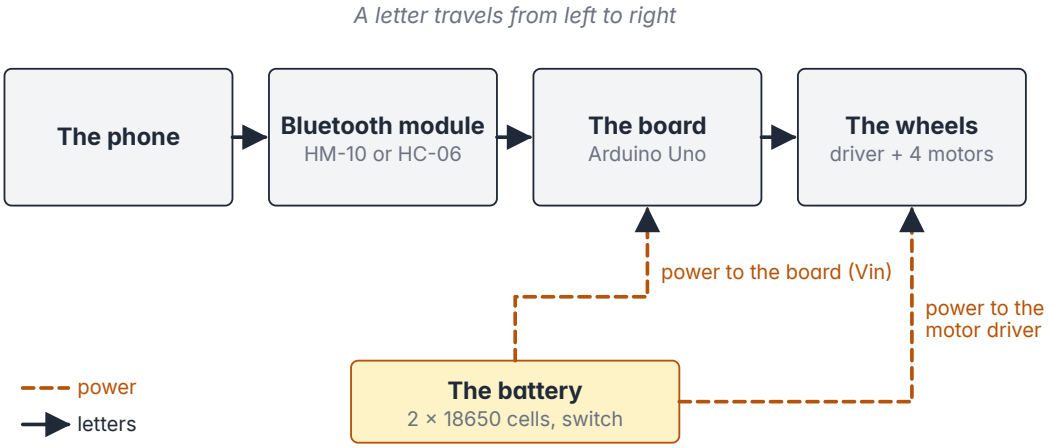


FIGURE 3.1 The four boxes. A letter travels from the phone to the Bluetooth module to the board to the wheels. The battery feeds the motor driver and the board on a separate power line.

There's one more thing in the picture that isn't a box: **the battery**. It doesn't carry letters. It carries power, along its own line, to two places: the motor driver and the board. Keep that in mind. A car can have every box working and still not move if the power line is broken.

The board decides; the motor driver pushes

People often call the board the car's brain. Like a brain, it decides. Unlike a brain, it only ever does exactly what the sketch says, nothing more, and it never guesses what you meant.

The board can't move the motors by itself. Its pins can **signal** but they can't **push**: a pin can send a tiny "on" or "off", but a motor needs far more power than that. So the board signals the motor driver, and the motor driver takes power straight from the battery and pushes it into the motors.

Being connected isn't the same as the letter arriving

One more idea, because it fools almost everyone. When the phone says *connected*, it means the phone and the Bluetooth module can hear each other. That is box 1 talking to box 2. It says nothing about whether the letter gets

from the module to the board, or whether the board does anything with it. “Connected” proves one link out of three. This book tests each link separately.

Four boxes, four stages

You build the car in four stages, and each stage brings one more box to life and then tests it. Figure 3.2 shows which stage proves which box.

- **Stage A** proves **the board**. You put the sketch on it, and it answers `Car ready`.
- **Stage B** proves **the wheels**. You build the chassis, wire the motor driver and the battery, and drive the wheels by typing letters on the laptop.
- **Stage C** proves **the Bluetooth module**. Its light comes on (test C1), and then the phone connects to it (test C2).
- **Stage D** proves the **whole journey**. A letter goes from the phone to the board, and the car’s answer comes back to the phone. Then you drive.

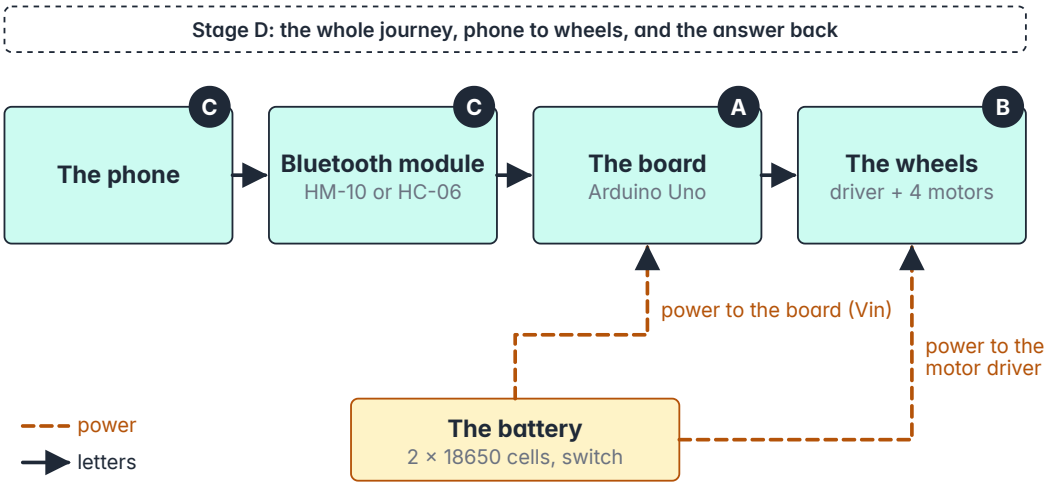


FIGURE 3.2 The four stages on the four boxes. Stage A lights the board, Stage B the wheels, Stage C the Bluetooth module and the phone, and Stage D the whole journey.

Because each stage has its own test, a test that fails tells you which box to look in. That’s the whole trick of fixing things, and it’s what Chapter 12 is built on.

When something doesn’t work: the loop

When something goes wrong, and something will, use this loop. It’s five short lines. Say them out loud.

1. **Look.** What exactly happens? Not “it doesn’t work”, but “the left wheels turn and the right ones don’t.”
2. **Which box?** Use the stage tests to find out which box the problem lives in.
3. **Change one thing.** Only one.
4. **Test again.** If that didn’t fix it, undo the change.
5. **Say what you found.** Out loud, to each other.

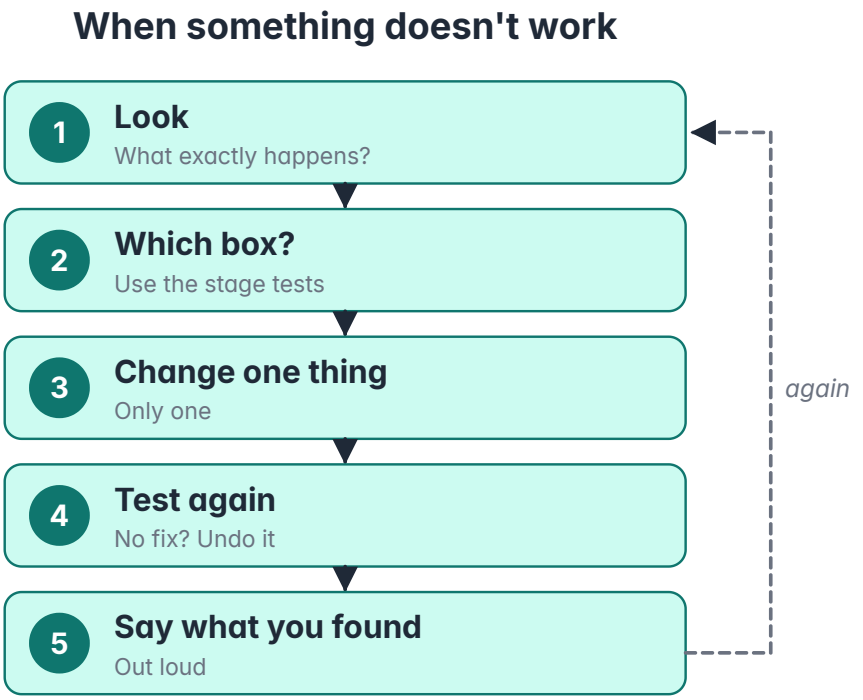


FIGURE 3.3 The debugging loop. Look, which box, change one thing, test again, say what you found.

The loop is on the printable card too, at troniction.com/go/car-roles.

TALK ABOUT IT

"Which box decides what to do? Which one does the pushing?"

PARENT: LISTEN FOR:

the board decides; the motor driver pushes, and the motors turn. Naming any box is a good start.

CHECK YOURSELF

Pause here and answer out loud.

1. The phone shows *connected*. Which boxes has that proved work, and which hasn't it?
2. Which two parts does the battery feed directly?

Answers at the back of the book on page 247

/// CHAPTER

4

Unbox and find

PART OF THE AUDIOBOOK

NOTE

Together. The child's first job. The child unpacks each part and finds every pin on it that the build uses. The parent reads each "find it" line aloud and ticks it when the child points to the right place.

How this chapter works

Lay a towel or a sheet of card on the table, so tiny parts don't roll away. Open one bag at a time. For each part, this chapter says what it does and then lists the pins to find on it. The child finds each one on the real board and points. The parent checks it against the figure and ticks it.

You'll meet every one of these pins again during the build. Finding them now, with nothing connected, means you'll recognise them later when your hands are full.

The full list of parts, with prices and where to buy them, is at the back of the book (**"The full parts list"**, page 234) and at troniction.com/go/car-parts.

TABLE 4.1 The parts. Prices and links are at troniction.com/go/car-parts

PART	HOW MANY	WHAT IT'S FOR
Board: Arduino Uno R3 (a clone is fine) and its USB cable	1	Runs the sketch; decides what the car does
Motor driver: L298N module, the screw-terminal kind	1	Takes battery power and pushes it into the motors
4WD chassis kit: plates, four wheels, four TT gear motors **with wires already attached**	1	The car's body and its wheels
Bluetooth module: **HM-10** (DSD TECH) **or HC-06** (Chapter 1 helps you choose)	1	Catches the letters from the phone
Battery holder for two 18650 cells, **with an on/off switch** and two wire leads	1	Holds the cells; the switch turns the power on and off
18650 rechargeable cells, button top	2	The car's power
18650 charger	1	Charges the cells
Passive buzzer module (a small board with a buzzer and three pins)	1	The horn, the reversing beep and the tune. Passive, not active: an active one only beeps
Mini breadboard, 170 holes	1	Joins the Bluetooth module's wires, without soldering
Resistors: one 1 kΩ and one 2 kΩ (2.2 kΩ also works)	1 each	Protect the Bluetooth module's listening pin
Jumper wires: male-to-female (at least 15) and male-to-male (at least 6)	21 or more	Every connection that isn't a screw terminal
Small Phillips screwdriver	1	The screw terminals and the chassis

Two buying traps

Motors without wires. The four motors must come with their two wires already attached, ending in small plastic plugs. Some motors that look almost the same arrive with two bare metal tabs instead, and those need soldering. The safe one is sold as Adafruit #3777. Adafruit's #3801 and #3802 look nearly identical and have no wires. If you buy a chassis kit, check the listing's photos show wires on the motors.

Bluetooth modules that only look right. Many cheap modules are sold as "HM-10" but run different software, and some behave differently. Buy the HM-

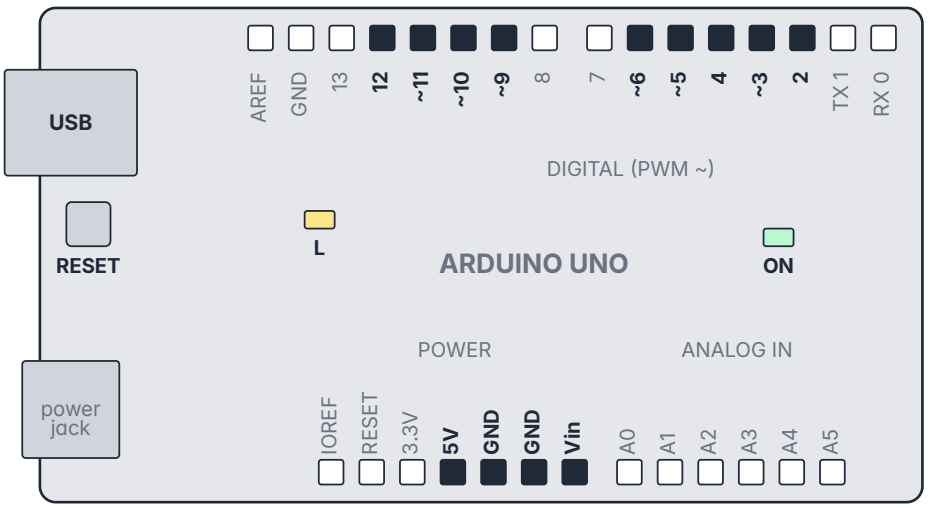
10 from DSD TECH, which is the one this book is written for. If you already have a module and aren't sure what it is, the lab at troniction.com/go/car-which-ble helps you tell.

One more: shops like Amazon US don't sell loose 18650 cells, so the cells usually come in a second order from a battery shop.

Which Bluetooth module do you have? A lab that helps you tell.
SCAN OR TYPE **troniction.com/go/car-which-ble**

The board

The board is the Arduino Uno: a small computer with no screen. You put a program on it once, over the USB cable, and it runs that program every time it has power. It keeps its program when you switch it off. What it doesn't keep is anything that happened while it was running, like the speed you last chose. Switch it on again and it starts the program from the beginning.



Black squares: the pins this car uses. The ~ mark means the pin can control speed.

FIGURE 4.1 The board, with every part this build uses labelled: the USB socket, the ON and L lights, pins 2, 3, 4, 5, 6 and 9 to 12, and the 5V, GND and Vin pins.

Find it (Figure 4.1). Parent reads; child points; parent ticks.

- [ ] The USB socket, where the cable plugs in.
- [ ] The ON light. It lights whenever the board has power.
- [ ] The L light. The sketch uses it to say it's running.
- [ ] Pins 2, 3 and 4, on the long row of holes marked DIGITAL.
- [ ] Pins 5 and 6. Look for a small wavy mark, `~`, printed beside them.
- [ ] Pins 9, 10, 11 and 12, further along the same row.
- [ ] The pins marked 5V, GND and Vin, on the shorter row on the other side.

The motor driver

The motor driver is the L298N. The board's pins can signal but can't push, so the motor driver does the pushing. The battery feeds it directly, and the board only tells it which way and how hard, using six thin wires that carry almost no power.

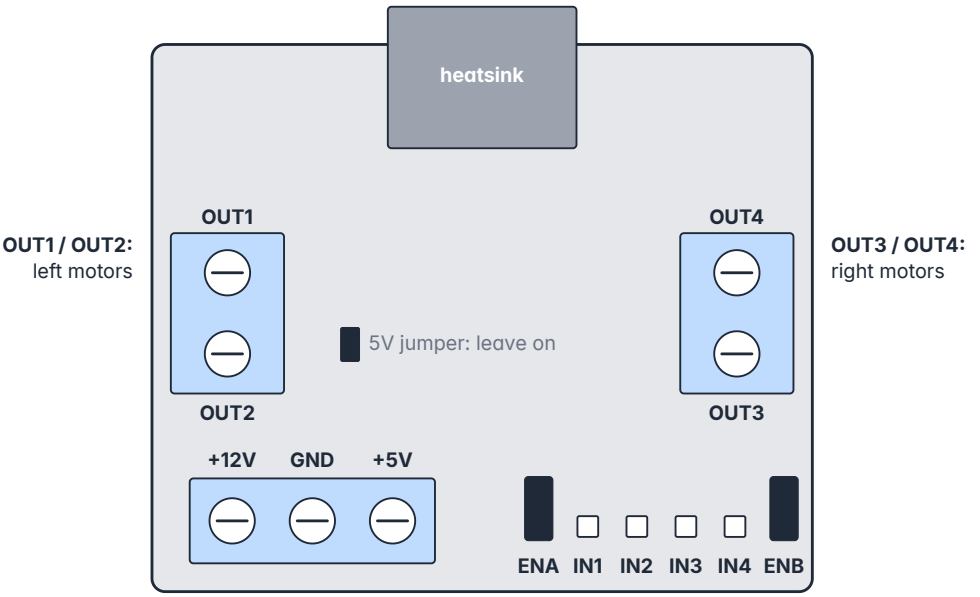


FIGURE 4.2 The motor driver. The four motor screws OUT1 to OUT4, the power screws marked +12V, GND and +5V, the six signal pins ENA, IN1, IN2, IN3, IN4 and ENB, and the two caps on ENA and ENB.

Find it (Figure 4.2).

- [ ] The two blue screw blocks for the motors, marked OUT1 and OUT2 on one side and OUT3 and OUT4 on the other.
- [ ] The three-screw block for power, marked +12V, GND and +5V. (Some boards print 12V or VMS instead of +12V; it's the screw at the end, furthest from +5V.)
- [ ] The row of six pins marked ENA, IN1, IN2, IN3, IN4 and ENB.
- [ ] The two small plastic caps sitting on the ENA and ENB pins. You'll take them off in Stage B.
- [ ] The black metal fin, the heatsink. It can get warm while the car drives.

The motors and the chassis

Four yellow gear motors, four wheels and two plates. They're **gear** motors because a bare motor spins far too fast and with almost no push. The gearbox trades that speed for the force needed to move a car carrying a battery.

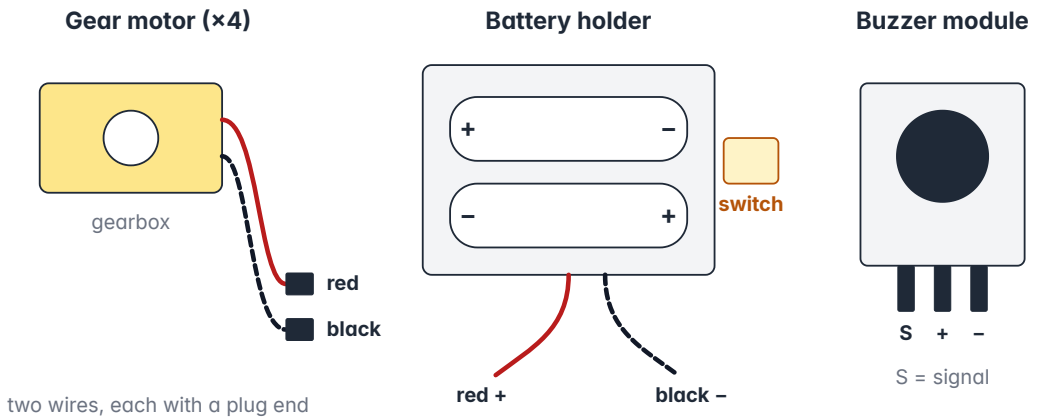


FIGURE 4.3 The small parts. A gear motor with its two wires, the battery holder with its switch and its red and black leads, and the buzzer module with its three pins.

Find it (Figure 4.3).

- [ ] On each of the four motors: two wires, one red and one black, each ending in a plug. If red and black look alike to you, the checker reads them out.
- [ ] On the battery holder: the on/off switch, and the two leads, red (+) and black (–).

- [ ] Inside the holder: the + and − marks that show which way each cell goes.
- [ ] On the buzzer module: three pins, marked something like S (signal), + and −.

The Bluetooth module

The Bluetooth module catches the letters the phone sends and passes each one along a wire to the board. Whichever one you bought, the HM-10 or the HC-06, it has the same four pins that matter: VCC and GND for power, TXD, which is the pin it talks on, and RXD, which is the pin it listens on. Some modules have six pins; the two at the ends aren't used in this car.

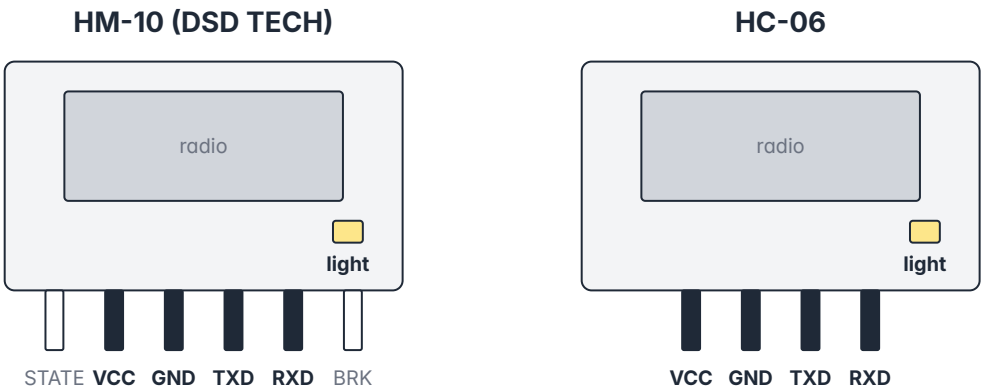


FIGURE 4.4 The two Bluetooth modules side by side. Both have VCC, GND, TXD and RXD, and a small status light.

Find it (Figure 4.4).

- [ ] VCC and GND, the power pins.
- [ ] TXD, the pin the module talks on.
- [ ] RXD, the pin the module listens on.
- [ ] The small status light on the module.

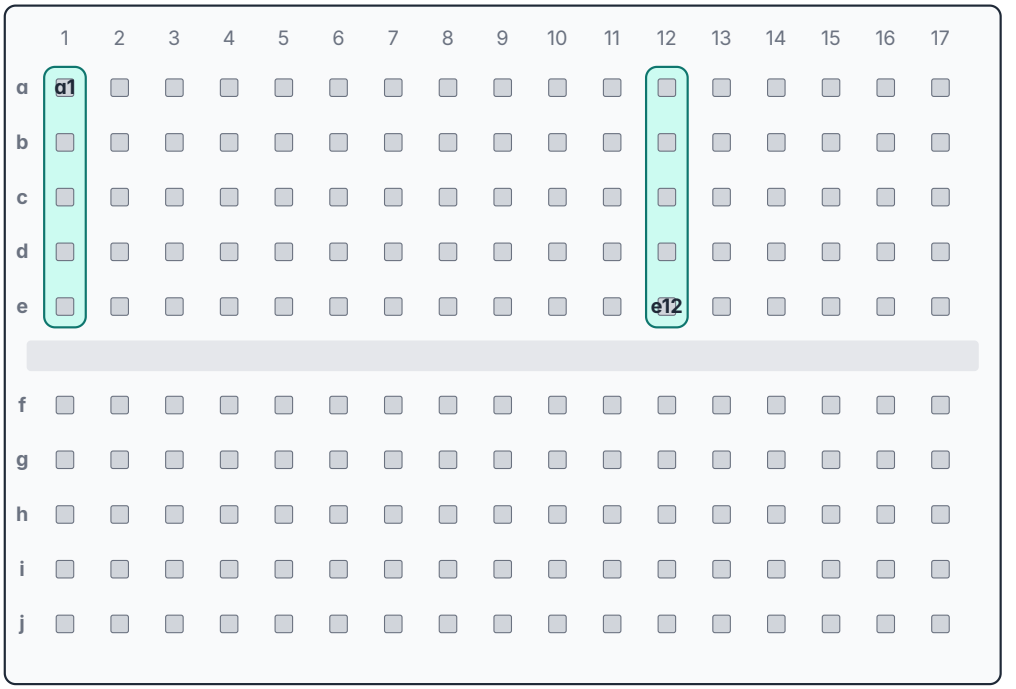
The breadboard and the two resistors

The breadboard is a little plastic block full of holes. Inside, metal strips join some of the holes together, so wires pushed into joined holes are connected to each other with no soldering. On this mini breadboard, each numbered column

of five holes, a to e, is joined. The five holes below the middle gap, f to j, make a separate group. Nothing joins one column to the next.

Every hole has a name: a letter for its row and a number for its column, like “a1” or “e12”. The steps in Stage C use those names, so you put each leg in an exact hole.

Each numbered column of five holes, a to e, is joined inside. f to j is a separate group.



Shaded: column 1 (holes a1 to e1) and column 12 (a12 to e12). Each is one joined strip.

FIGURE 4.5 The mini breadboard. Each column of five holes, a to e, is joined inside. Holes are named by row letter and column number, like a1.

The two resistors protect the Bluetooth module. The board’s pins send 5 volts, but the module’s listening pin is built for about 3.3. The two resistors together turn the board’s 5 volts down to about 3.3 before it reaches the module. Chapter 10 puts them in, and Part 2 explains how they do it.

Tell them apart by the label on the paper strip or bag they came in, not by their colors. The colored bands are a second check: the 1 kΩ reads brown, black, red; the 2 kΩ reads red, black, red; a 2.2 kΩ reads red, red, red. In some light

those colors are hard to tell apart, which is why the label comes first. Keep each one on its labelled strip until its step.

Find it (Figure 4.5 and Figure 4.6).

- [ ] The row letters a to e, and the column numbers along the edge.
- [ ] Hole a1 and hole e12. Put a finger on each.
- [ ] The 1 kΩ resistor, on its labelled strip.
- [ ] The 2 kΩ (or 2.2 kΩ) resistor, on its labelled strip.

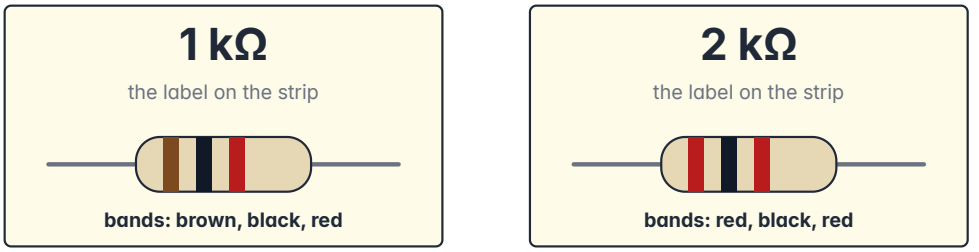


FIGURE 4.6 The two resistors, each on its labelled paper strip. The 1 kΩ has bands brown, black, red; the 2 kΩ has bands red, black, red.

The battery, the cells and the buzzer

The two cells are 18650s: lithium cells, bigger than an AA, and the only part of this car that needs real care. Chapter 5 is all about them. For now, the parent keeps them, still in their charger or their box. The child finds the holder's switch, because switching it off is the child's job before every wiring change.

The buzzer module is the car's horn. It also beeps while the car reverses, and it can play *Happy Birthday*.

TALK ABOUT IT

"Point to the module's two letter pins. Which one talks, and which one listens?"

PARENT: LISTEN FOR:

TXD talks and RXD listens. Pointing to the right two pins is enough; the names come back in Stage C.

CHECK YOURSELF

Pause here and answer out loud.

1. Your motors arrived with no wires, just two small metal tabs on each. What went wrong?
2. Guess now, and you'll find out in Stage B: which pins on the board can control the motors' speed? (This is a guess, not a test. Look at the board for a clue.)

Answers at the back of the book on page 247

/// CHAPTER

5

Safety: read this together

PART OF THE AUDIOBOOK

> NOTE
> **Read this page with your child, not for them.** It's short, and every rule on it comes back later, at the exact step where it matters.

The only dangerous thing in the box

The two battery cells are the only dangerous thing in this build, and they're dangerous in boring, avoidable ways.

- **18650 cells aren't AA batteries.** They don't forgive being shorted, crushed, or charged in the wrong charger. A shorted cell can get hot enough to burn you.
- **Check the polarity, which way round the cells and the leads go, before every power-up.** Getting the battery the wrong way round is the one mistake that destroys parts instead of just not working.
- **Charge the cells in their own charger, on a hard surface**, not on a bed or a sofa, and not overnight with nobody around.
- **If a cell is dented, swollen, leaking, or warm for no reason, stop using it.** Don't throw it in the bin. Take it to a battery recycling point; many hardware and electronics shops have one.

Nothing else here can hurt anyone. There's no mains electricity anywhere in this build and no soldering iron.

When does anything become live?

The car has two ways to cut the power, and they belong to different people.

The switch on the battery holder belongs to the child. Before every wiring change, the child switches it off. Every time.

The cells belong to the parent. Putting them in, taking them out and charging them is the parent's job. Not because a 12-year-old couldn't, but because this is where being wrong is expensive.

Nothing in the car is live until **both** are true: the cells are in, **and** the switch is on.

The board can also get power from the laptop, through the USB cable. So when this book says **"all power off"**, it always means two things: **the USB cable unplugged, and the battery switch off.**

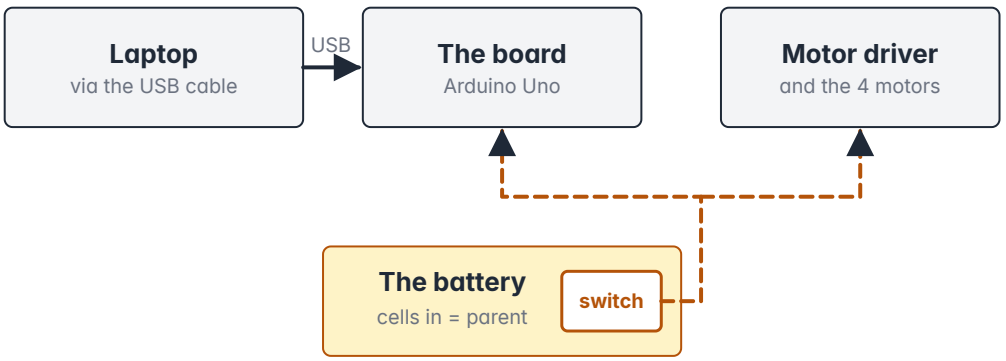


FIGURE 5.1 Where the power is. The battery feeds the motor driver and the board; the USB cable can also feed the board. All power off means the switch off and the USB cable out.

Warm is normal; hot is a stop

While the car drives, the motor driver's black metal fin, its heatsink, gets warm. That's normal: the motor driver turns some of the battery's power into heat, and the fin is there to get rid of it. Part 2 explains why.

But if anything is **too hot to keep touching**, or you smell something hot, or anything smokes: switch off at once, and the parent takes the cells out. Then look for the cause before you switch on again. The most likely one is a wire in the wrong place, which Chapter 12 helps you find.

The rules, in one list

1. The child switches off before every wiring change.
2. The parent puts the cells in and takes them out.
3. “All power off” means the switch off and the USB cable out.
4. The battery goes on last, after the checklist, with the car on a box.
5. Too hot to touch, a hot smell, or smoke: switch off, cells out.

These rules come back as warnings in the steps, just before the step where each one matters.



More about 18650 cells: what they are, how to charge them, and how to store them.

SCAN OR TYPE **troniction.com/go/car-batteries**

CHECK YOURSELF

Pause here and answer out loud.

1. Which two things must be true before anything in the car is live?
2. A cell is dented. What do you do with it?

Answers at the back of the book on page 247

/// CHAPTER

6

Stage A · The board: put the sketch on it

PART OF THE AUDIOBOOK

> ⓘ NOTE
> **This stage.** The child plugs the bare board into the laptop and presses Upload, and the board answers `Car ready`. That proves the first box, the board, and the laptop talking to it. Nothing else is connected yet. It's the shortest stage, and the end of its test is the first good place to stop.

Why the board comes first

The sketch goes onto the board before a single wire is connected. A bare board on a USB cable is the simplest the car will ever be, so if something goes wrong here, there are only three things it can be: the laptop, the cable, or the board. Once the board says `Car ready`, you know those three work, and you never have to wonder about them again while you build.

The parent does a short setup first, as part of the prep list. Then the child does the rest.

For the parent: install and set up

You need the Arduino IDE, the free app that puts sketches onto the board. Do these steps before session 1, then unplug the board so your child can plug it in.

There's a short video of these steps at troniction.com/go/car-a.

PARENT P1 Download the Arduino IDE from arduino.cc/en/software and install it. This book was written with version 2.3. A newer 2.x version works the same way.

PARENT P2 Plug the board into the laptop with its USB cable. The board's ON light comes on.

PARENT P3 Open the Arduino IDE. In the **Tools** menu, choose **Board**, then **Arduino AVR Boards**, then **Arduino Uno**. If the IDE offers to install "Arduino AVR Boards", say yes.

PARENT P4 In **Tools → Port**, choose the board. To be sure which one it is, unplug the board, look at the list, plug it back in, and pick the one that appeared.

⏱ TEST IT

YOU SHOULD SEE: a port appears when you plug the board in.

A INSTEAD: no new port appears, even though the ON light is lit.

MEANING: the laptop can't talk to the board. Either the cable only charges (some USB cables have no data wires inside), or the board is a clone that needs a driver.

FIX: try another USB cable first, one you know carries data. If that doesn't help and your board has a chip marked CH340 near the USB socket, install the CH340 driver from the chip maker, WCH. The link is on the Stage A test page, troniction.com/go/car-test-a.

PARENT P5 Unzip the sketch you downloaded from troniction.com/go/car-sketch. Keep the folder called `car` with both files inside it, `car.ino` and `pitches.h`.

PARENT P6 In the IDE, choose **File → Open** and open `car.ino`. The sketch opens in a window.

PARENT P7 Unplug the board. Leave the IDE open with the sketch in it.

For the child: put the sketch on the board

Session 1 starts here, with the child. You'll plug in the board, press Upload, and wait for the board to say it's ready. The steps are A1 to A5.

CHILD A1 Plug the USB cable into the board and into the laptop. Look for the ON light.

PARENT A2 Read the name at the top of the IDE window, beside the board picker. It should say **Arduino Uno**. If it doesn't, choose it again (step P3 and P4).

CHILD A3 Click **Upload**: the round button with an arrow pointing right, at the top left. Then wait. Don't unplug anything.

CHILD A4 Watch the message area at the bottom of the window. When the upload has worked, it says **Done uploading**. While it uploads, the lights on the board flicker. That's normal.

⏱ TEST IT

YOU SHOULD SEE:	"Done uploading".
A INSTEAD:	an error in orange or red text, such as "not in sync" or "port busy".
MEANING:	the IDE couldn't reach the board.
FIX:	check that **Tools → Board** says Arduino Uno and **Tools → Port** is the board's port. Close any other program that might be using the board. Unplug the cable, plug it back in, and press Upload again.

CHILD A5 Open the **Serial Monitor**: the magnifying-glass button at the top right of the window. A panel opens at the bottom. In its menu of speeds, choose **9600 baud**.



FIGURE 6.1 The Serial Monitor after the upload. It shows the words Car ready, and its speed is set to 9600 baud.

The Stage A test

The proof is the words `Car ready` in the Serial Monitor. The L light coming on and staying on is a second sign, but on its own it proves nothing.

Opening the Serial Monitor usually restarts the board, so `Car ready` appears even if you opened it a while after the upload. If it isn't there, press the board's RESET button. When the board restarts, the L light flickers quickly a few times. That's the board starting up. Then it comes on and stays on.

⏱ TEST IT	
YOU SHOULD SEE:	`Car ready` in the Serial Monitor. (The L light also comes on and stays on.)
A INSTEAD:	the L light blinks slowly, on and off, forever, and the Serial Monitor is empty.
MEANING:	the board is still running the sample sketch it came with. The upload didn't happen.
FIX:	press Upload again and wait for "Done uploading".

B INSTEAD:	strange symbols in the Serial Monitor instead of words.
MEANING:	the Serial Monitor's speed isn't 9600.
FIX:	choose 9600 baud. Then press the small button on the board marked RESET. It restarts the board, and `Car ready` prints again.
C INSTEAD:	the L light is on, but the Serial Monitor is empty.
MEANING:	nothing is proved yet. `Car ready` is the only proof that this sketch is on the board. The L light can be on for other reasons.
FIX:	check the speed is 9600, then press RESET on the board. If `Car ready` still doesn't appear, press Upload again (A3) and wait for "Done uploading".

TALK ABOUT IT

"The screen says `Car ready`. Which box have we just proved works?"

PARENT: LISTEN FOR:

the board, and the laptop talking to it. Not the wheels yet: nothing else is connected.

This is a stopping point. Unplug the USB cable. The sketch stays on the board, so next time the board starts with it already there.

Stuck for 10 minutes? Run the loop from Chapter 3, then go to the Stage A test page: troniction.com/go/car-test-a.

A short video of the upload, from plugging in to `Car ready`.

SCAN OR TYPE troniction.com/go/car-a

The Stage A test page: what you should see, what you might see instead, and the fix.

SCAN OR TYPE **troniction.com/go/car-test-a**

CHECK YOURSELF

Pause here and answer out loud.

1. The ON light is lit, but the laptop shows no new port. What's the most likely cause?
2. The L light blinks slowly forever and the Serial Monitor is empty. What does that tell you?

Answers at the back of the book on page 247

/// CHAPTER

7

Stage B1 · Chassis and motors

PART OF THE AUDIOBOOK

NOTE

This stage. The child builds the chassis, mounts the four motors and the other parts, and screws the motor wires into the motor driver. Stage B has three chapters, and its test comes at the end of the third, in Chapter 9. **This isn't a stopping point**: nothing new can be tested yet. If you must stop, stop at the end of a step and write down its number.

What you'll do

First the car gets its body: two plates, four motors and four wheels. Then the board, the motor driver, the breadboard and the battery holder go on top. Last, the four motors' wires go into the motor driver's screws.

The motors work in pairs. The two motors on the left side of the car always do the same thing, so they share one pair of screws on the motor driver, OUT1 and OUT2. The two on the right share OUT3 and OUT4. That's why the car turns without steering: to turn left, the left wheels go backwards while the right wheels go forwards, and the car spins on the spot.

Wire them straight, then test

Here's the fiddly part from Chapter 1. The motors on the left face the opposite way from the ones on the right. The sketch already allows for that. But

whether “forward” comes out as forward on your car depends on how your motors were made, and no picture can tell you.

So this book doesn’t guess. You wire **both sides the same way**, which this book calls “straight”: on each motor, the red wire goes to the first screw of its pair, and the black wire to the second. Then, in Chapter 9, you both predict which way the wheels will turn, test it with the wheels in the air, and if any wheels go the wrong way, you swap two wires and test again. Nothing can break.

Steps B1 to B17

The chassis steps follow your kit. There’s a short video of them at troniction.com/go/car-b1.

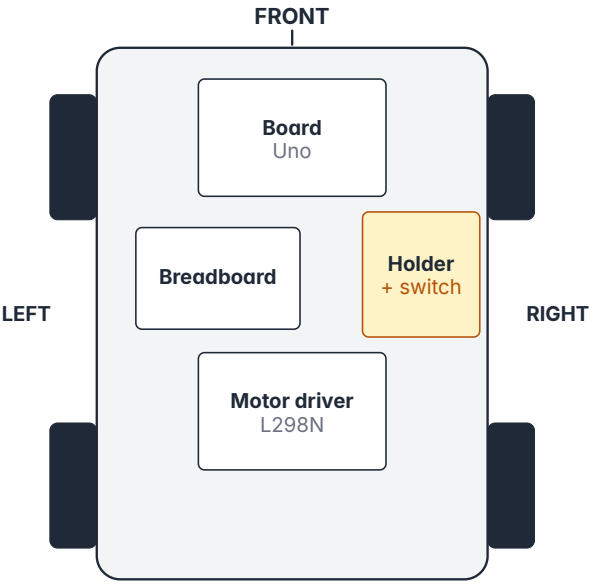
⚠ WARNING

Before you start: the battery cells stay out, with the parent, until Chapter 9. Switch the battery holder off now and leave it off.

TOGETHER B1 Open your chassis kit’s instruction sheet. It decides how the plates and motors fit together.

CHILD B2 Fix the four motors to the bottom plate the way the kit sheet shows. Point every motor’s wires toward the middle of the car.

CHILD B3 Push a wheel onto each motor’s shaft, all the way.



One possible layout. Your kit's holes decide the exact places.

FIGURE 7.1 The layout from above. The board sits near the front, the breadboard and the battery holder in the middle, and the motor driver near the back. The end marked FRONT is at the top.

CHILD B4 Put a small piece of tape on the front end and write FRONT. “Left” and “right” now mean the car’s left and right, as if you sat in it facing the front.

CHILD B5 Fix the motor driver, the board, the breadboard and the battery holder to the top plate, as in Figure 7.1. Use the kit’s screws, or double-sided tape. Put the holder where you can reach its switch.

CHILD B6 Fix the top plate onto the bottom plate. Bring all eight motor wires up so they reach the motor driver.

Screwing a wire into a terminal

Every motor wire ends in a small metal pin. It goes into a hole in the side of a screw block on the motor driver, and the screw on top clamps it. The same three moves work every time: loosen, push in, tighten. Then the tug test: pull the wire gently. If it comes out, it wasn’t clamped.

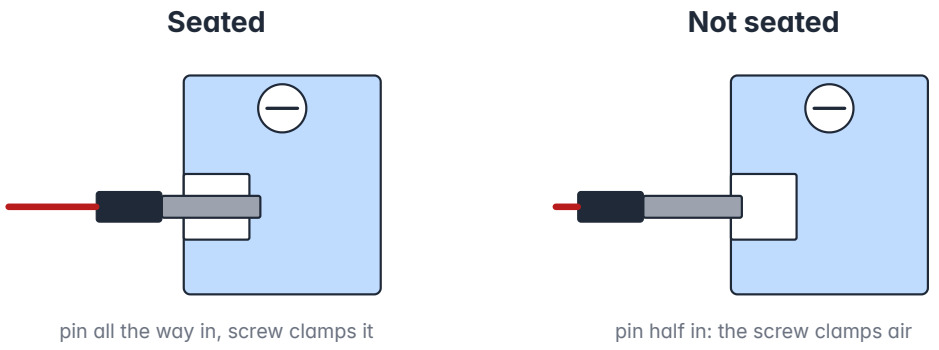


FIGURE 7.2 A wire in a screw terminal. Left: seated, with the metal pin all the way in and the screw clamping it. Right: not seated, with the pin only half in.

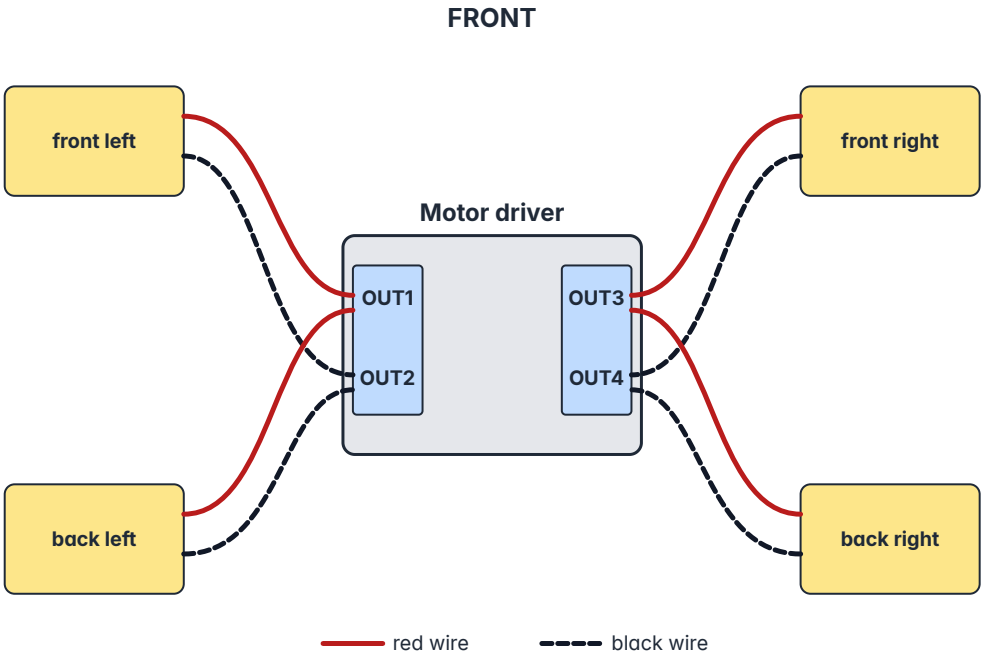


FIGURE 7.3 The motors to the motor driver, wired straight. Both left motors: red wire to OUT1, black to OUT2. Both right motors: red to OUT3, black to OUT4.

CHILD B7 Left side. Turn both screws of the block marked OUT1 and OUT2 to the left a few turns, so the holes open.

CHILD B8 Push the **red** wires of **both** left motors into **OUT1**, side by side.

CHILD B9 Push the **black** wires of both left motors into **OUT2**.

CHILD B10 Tighten both screws, turning to the right until they're firm.

CHILD B11 Tug each of the four wires gently. None should come out.

PARENT B12 Check against Figure 7.3: both left motors, red in OUT1, black in OUT2.

CHILD B13 Right side. Open both screws of the block marked OUT3 and OUT4.

CHILD B14 Push the **red** wires of both right motors into **OUT3**.

CHILD B15 Push the **black** wires of both right motors into **OUT4**.

CHILD B16 Tighten both screws. Tug each of the four wires.

PARENT B17 Check: both right motors, red in OUT3, black in OUT4 (the same figure as B12).

⏱ TEST IT

YOU SHOULD SEE: all eight motor wires stay in when you tug them.

A INSTEAD: a wire slides out.

MEANING: the screw was clamping nothing, or only the plastic.

FIX: loosen that screw, push the metal pin in all the way, so no metal shows outside the hole, and tighten again. Tug again.

What it will feel like

Two wires in one screw hole feels crowded. That's expected: push one pin in, then the other beside it, then tighten. If the kit's screws are tiny, the parent can hold the chassis steady while the child turns the screwdriver. That's still the child's step.

Stuck for 10 minutes? Run the loop from Chapter 3, then watch the short video at troniction.com/go/car-b1.



A short video of the motor wires going into the motor driver.

SCAN OR TYPE troniction.com/go/car-b1

CHECK YOURSELF

Pause here and answer out loud.

1. Why do the two left motors share one pair of screws?
2. You tug a wire and it comes out. What was wrong?

Answers at the back of the book on page 247

/// CHAPTER

8

Stage B2 · Motor driver to the board

PART OF THE AUDIOBOOK

NOTE

This stage. The child takes two caps off the motor driver, gives the breadboard power from the board, and wires the buzzer. Then a quick horn test on the laptop's power alone: **its end is a stopping point.** Last, six wires connect the motor driver to the board. Stage B's full test is in the next chapter.

Six wires: which way, and how hard

The board tells the motor driver two things for each side of the car: **which way** to turn, and **how hard**. Which way takes two wires per side, the four marked IN1 to IN4. How hard takes one wire per side, the two marked ENA and ENB. Six wires in all.

In words, the whole table is one sentence: **the two speed wires go to pins 5 and 6; the four direction wires go to pins 9 to 12, in reverse order.** ENA goes to 5 and ENB to 6. IN1 goes to 12, IN2 to 11, IN3 to 10 and IN4 to 9. It is all drawn in Figure 8.3.

The two caps come off first

Most motor drivers arrive with small plastic caps on the ENA and ENB pins. A cap joins its pin to a fixed "full power" signal, so with the caps on, the motor

driver ignores the board's speed wire and runs flat out. The car would drive, but every speed would be the same. So the caps come off before the wires go on.

Not every pin can do speed

MYTH

> "Any pin will do. A wire is a wire."

WRONG That's wrong for the speed wires.

To make a motor go slower, the board switches its pin on and off very fast, hundreds of times a second, and keeps it on for less of each moment. The motor feels that as "less push". Only six pins on the board have the hardware to switch that fast: 3, 5, 6, 9, 10 and 11, the ones with a `~` printed beside them. On any other pin, the sketch's "go at speed 150" just becomes "full on".

This isn't made up. The sketch this book's car runs was written in 2016 with its speed wires on pins 13 and 8. Neither can do speed. The car drove perfectly, always at full speed, and nobody noticed for years, because every speed button gave the same speed and nothing ever broke. In 2026 the two wires moved to pins 5 and 6, and speed 1 finally crawled. Part 3 tells the whole story as a detective case.

That's the answer to the guess in Chapter 4: only the `~` pins can control speed.

How four wires make a motor go both ways

Inside the motor driver, each side has four electronic switches arranged in a square, with the motor across the middle. Close one diagonal pair and the current runs through the motor one way. Close the other pair and it runs the other way, so the motor reverses. The IN wires choose which pair closes. Part 2 draws it.

Steps B18 to B35

The caps step has its own short video at troniction.com/go/car-caps.

> ⚠ WARNING
>
> **All power off before you start:** the USB cable unplugged and the battery switch off. The cells are still out.

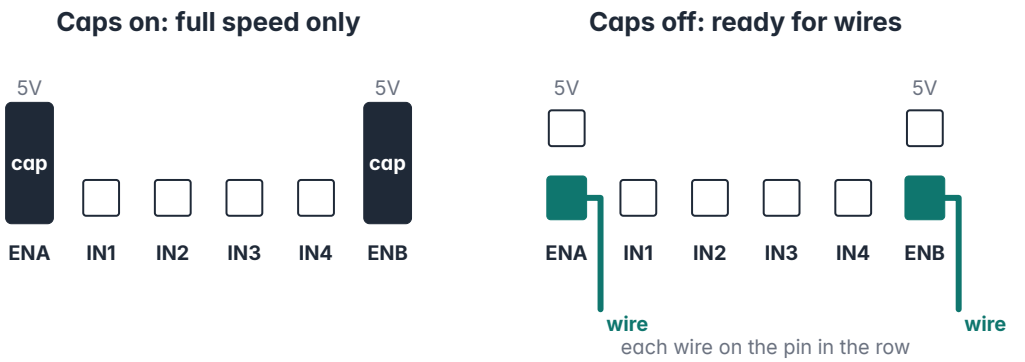


FIGURE 8.1 The ENA and ENB pins with their caps on, and with the caps off. With the caps off, each speed pin is free for a wire.

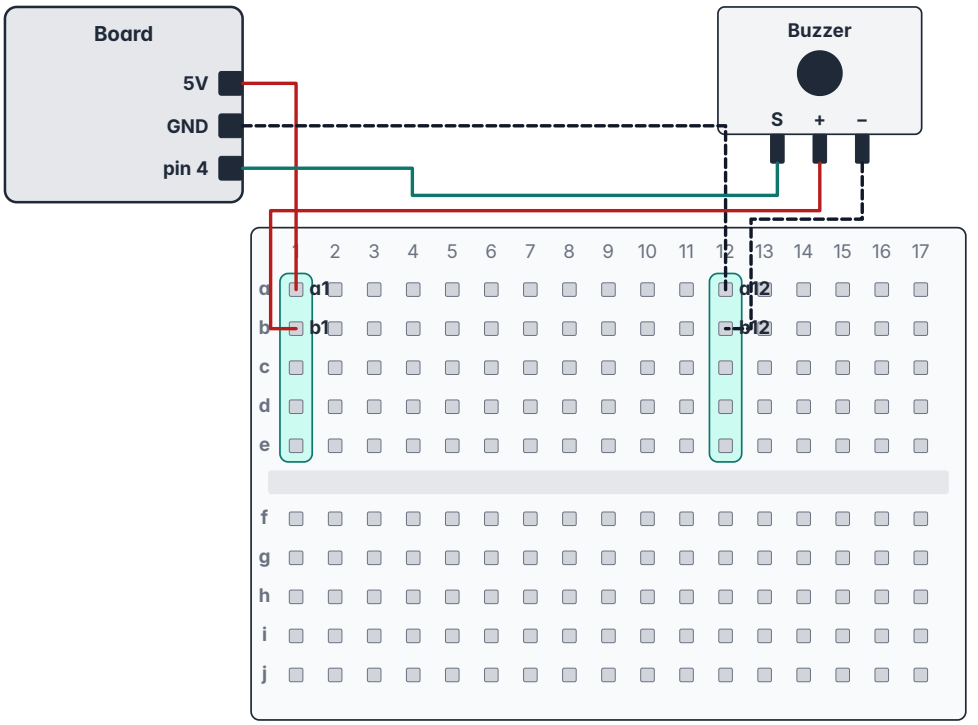
CHILD B18 Pull the cap off the **ENA** pins, straight up, with your fingers or small pliers. Put it in a bag so it doesn't get lost.

CHILD B19 Pull the cap off the **ENB** pins the same way. Leave the cap near the power screws where it is.

PARENT B20 Check: no cap on ENA or ENB. Each cap covered two pins: the ENA (or ENB) pin, in the row with IN1 to IN4, and a 5V pin behind it. The speed wire will go on the pin **in the row**.

The buzzer, and power for the breadboard

The buzzer needs three wires: one signal wire from the board's pin 4, and power from the board's 5V and GND pins. The board has only one 5V pin, and the Bluetooth module will need it too in Stage C, so 5V and GND go to the breadboard first. Each column of the breadboard is joined inside, so every wire in the 5V column gets 5V. The breadboard steps use the hole names from Chapter 4.



5V to column 1 (solid). GND to column 12 (dashed). Buzzer S to pin 4.

FIGURE 8.2 The buzzer and the breadboard's power. The board's 5V goes to column 1 and GND to column 12. The buzzer's plus goes to column 1, its minus to column 12, and its S pin to the board's pin 4.

CHILD B21 With a male-to-male jumper: the board's **5V** pin to breadboard hole **a1**. Column 1 is now the 5V column.

CHILD B22 With a male-to-male jumper: one of the board's **GND** pins, on the same short row as 5V, to hole **a12**. Column 12 is now the GND column.

CHILD B23 With a male-to-female jumper: the buzzer's **+** pin to hole **b1**.

CHILD B24 With a male-to-female jumper: the buzzer's **–** pin to hole **b12**.

CHILD B25 With a male-to-female jumper: the buzzer's **S** pin to **pin 4** on the board.

PARENT B26 Check against Figure 8.2: 5V in a1, GND in a12, buzzer + in b1, – in b12, S on pin 4. Column 1 holds only 5V wires; column 12 only GND wires.

> ⚠ WARNING
>
> **5V and GND must never touch.** Never put a 5V wire and a GND wire in the same column. That would short the board's power. If you find one, switch everything off before you move it.

The horn test

Before the six signal wires go on, a quick test on the laptop's power alone. Plug in the USB cable, open the Serial Monitor at 9600 with **No Line Ending**, and type a capital V: the horn sounds until you type S. It's the first time something you wired answers back, and it's a good place to stop.

CHILD B27 Plug in the USB cable. Open the Serial Monitor. Choose **9600 baud** and **No Line Ending**.

CHILD B28 Type **V** and press Enter. Then type **S** and press Enter.

⏱ TEST IT

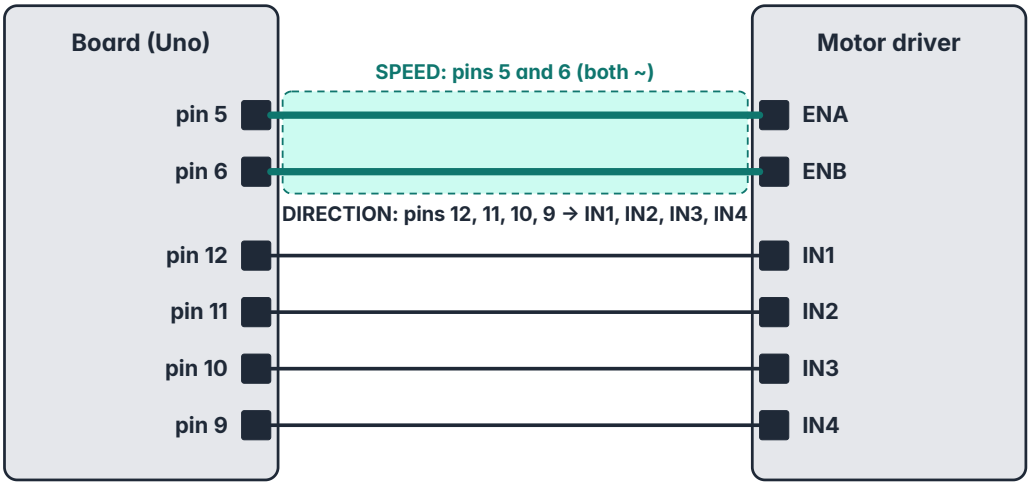
YOU SHOULD SEE:	`Got V from USB`, and the horn sounds until you type S.
A INSTEAD:	`Got V from USB`, but no sound.
MEANING:	the buzzer's wires. Its power goes through columns 1 and 12, and its S wire to pin 4.
FIX:	unplug the USB cable. Check B21 to B25 against Figure 8.2, then test again.
B INSTEAD:	one short beep, and `Got (newline) from USB.`
MEANING:	the Serial Monitor is adding a "new line" letter, and the car treats it as stop.
FIX:	choose No Line Ending.

This is a stopping point. Unplug the USB cable.

The six signal wires

Now the six wires. Each is a male-to-female jumper. The female end, the one with a hole, pushes onto the motor driver's pin. The male end, the one with a pin, goes into the board's socket.

> ⚠ WARNING
>
> **All power off:** the USB cable unplugged and the battery switch off.



Male end in the board's socket; female end on the motor driver's pin.

FIGURE 8.3 The motor driver to the board. Speed wires: ENA to pin 5 and ENB to pin 6. Direction wires: IN1 to pin 12, IN2 to 11, IN3 to 10 and IN4 to 9.

CHILD B29 ENA on the motor driver to **pin 5** on the board.

CHILD B30 ENB to **pin 6**.

CHILD B31 IN1 to **pin 12**.

CHILD B32 IN2 to **pin 11**.

CHILD B33 IN3 to **pin 10**.

CHILD B34 IN4 to **pin 9**.

TABLE 8.1 The six signal wires. Speed on 5 and 6; direction on 9 to 12, in reverse order

MOTOR DRIVER PIN	BOARD PIN	DOES
ENA	5	Left side's speed
ENB	6	Right side's speed
IN1	12	Left side's direction
IN2	11	Left side's direction
IN3	10	Right side's direction
IN4	9	Right side's direction

PARENT B35

Check all six against Table 8.1, one at a time, reading each pair aloud: "ENA, pin 5." The child points to both ends of the wire.

Stuck for 10 minutes? Run the loop from Chapter 3, then watch the caps video at troniction.com/go/car-caps.



A short video: taking the two caps off, and the six wires going on.

SCAN OR TYPE troniction.com/go/car-caps

CHECK YOURSELF

Pause here and answer out loud.

1. Every speed button gives the same speed. Name two causes.
2. Predict: you swap the wires on IN1 and IN2. What changes?

Answers at the back of the book on page 247

/// CHAPTER

9

Stage B3 · Power, and the wheels test

PART OF THE AUDIOBOOK

NOTE

This stage. The battery holder's two leads go to the motor driver **and** to the board. You check six things together, the parent puts the cells in, and the child switches on. Then, with the car on a box, the child drives the wheels by typing letters on the laptop. The last check: unplug the USB cable and the board stays on. **The end of this test is a stopping point.**

Why the board needs the battery too

So far the board has run on power from the laptop, through the USB cable. But a car can't drive around tied to a laptop. So the battery has to feed **two** places: the motor driver, which pushes power into the motors, and the board, which runs the sketch.

The battery holder has only two leads, red and black. Each one goes into a screw on the motor driver, and a short jumper wire goes from that same screw to the board. Red goes to the motor driver's +12V screw and on to the board's Vin pin. Black goes to the motor driver's GND screw and on to one of the board's GND pins.

That black wire does a second job. It joins the motor driver's ground and the board's ground, so the two agree on what "zero" is. Without a shared ground,

the six signal wires mean nothing to the motor driver, and the car can look perfectly wired while nothing moves.

Steps B36 to B45: the battery wires

There's a short video of the battery steps at troniction.com/go/car-vin.

> ⚠ WARNING
>
> **The battery cells stay out, and the switch stays off**, through all of these steps.

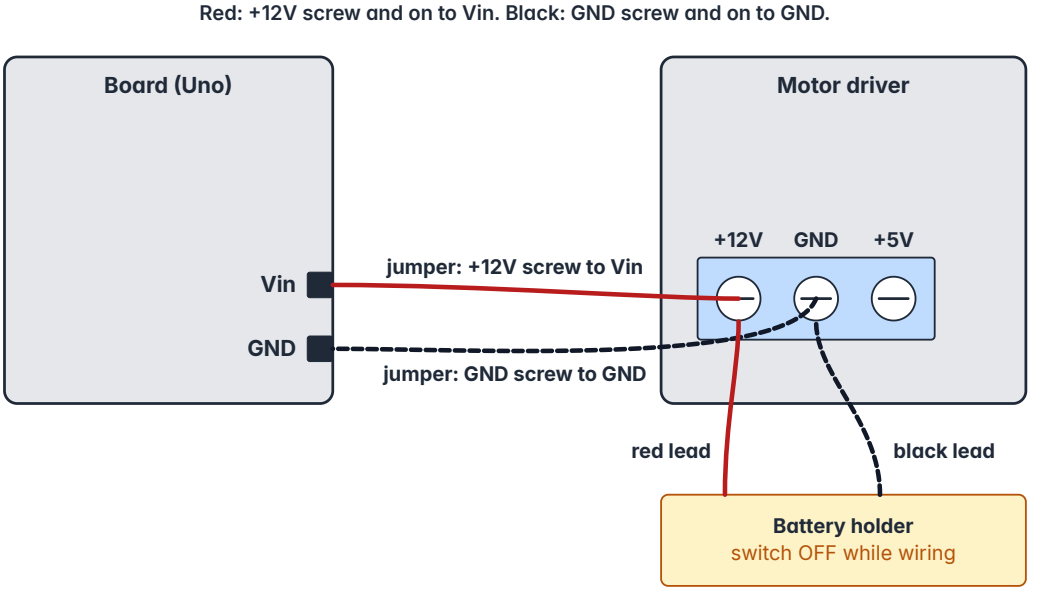


FIGURE 9.1 The battery to the motor driver and the board. Red: the holder's lead to the +12V screw, and a jumper from that screw to Vin. Black: the holder's lead to the GND screw, and a jumper from that screw to a GND pin.

CHILD B36 Check that the battery holder's switch is **off**.

CHILD B37 Loosen the motor driver's **+12V** screw a few turns.

CHILD B38 Push the holder's **red** lead and one end of a male-to-male jumper into it, side by side.

CHILD B39 Tighten the +12V screw.

CHILD B40 Push the other end of that jumper into the board's **Vin** pin.

CHILD B41 Loosen the motor driver's **GND** screw.

CHILD B42 Push the holder's **black** lead and one end of a second male-to-male jumper into it.

CHILD B43 Tighten the GND screw.

CHILD B44 Push the other end of that jumper into the board's free **GND** pin, next to Vin.

PARENT B45 Check: red lead and a jumper in +12V, the jumper on to Vin; black lead and a jumper in GND, the jumper on to GND. Then the child tugs each of the four ends gently. None should come out.

The checklist before power

Six checks. They prevent the only expensive mistakes in this book. One of you reads each line aloud; the other checks it on the car and says “yes”. Here they are.

The checklist before power. One reads, one checks. Tick each line.

- ☐ **Battery polarity:** red to +12V and Vin; black to GND.
- ☐ **No stray metal** touching anything.
- ☐ **Grounds joined:** the motor driver’s GND screw to the board’s GND.
- ☐ **Every motor wire seated** (tug each one).
- ☐ **Column 1 only 5V; column 12 only GND.**
- ☐ **The car is on a box,** all four wheels off the table.

Our guesses for F: left wheels □ forward □ backward · right wheels □ forward □ backward

TOGETHER B46 The checklist. One reads, one checks, every line, out loud. Tick each one.

TALK ABOUT IT

Before the cells go in: “Point to the wire that will carry the most power. Why that one?”

PARENT: LISTEN FOR:

a battery lead, the red or the black one at the holder. “Because it feeds everything” is right.

Cells in, switch on

This is the moment the car becomes live, so the parent does the cells, and the child does the switch. The USB cable stays out, so the board’s ON light can only come from the battery: this is the first check that the battery reaches the board.

⚠ WARNING

The parent puts the cells in. Check the switch is off first. Match each cell's + end to the + mark inside the holder. A cell in backwards can damage the motor driver and the board.

▲ PARENT B47 Check the switch is off. Put the two cells in, each matching the + mark in the holder.

▲ CHILD B48 Switch the holder **on**. Look at the board's ON light and the motor driver's small light. No wheel should move.

⏱ TEST IT

YOU SHOULD SEE:	the board's ON light and the motor driver's light both come on, and nothing gets hot.
A INSTEAD:	no light anywhere.
MEANING:	no power is reaching the motor driver or the board.
FIX:	switch off. The parent checks the cells are the right way round and pushed fully in. Then check the red and black leads are clamped in their screws (B38, B42).
B INSTEAD:	the motor driver's light is on, but the board's ON light is off.
MEANING:	the battery reaches the motor driver but not the board.
FIX:	switch off; the parent takes the cells out. Check B37 to B44: the jumper from +12V to Vin, and the jumper from GND to GND. Tug both ends of each.

C INSTEAD:	something feels hot, smells hot, or the lights flicker oddly.
MEANING:	something is wired wrongly, most likely the battery leads swapped.
FIX:	switch off at once. The parent takes the cells out. Go through the checklist again, starting at line 1.

The wheels test

Now the wheels box gets its test, driven from the laptop. Plug in the USB cable, open the Serial Monitor, choose 9600, and set the line ending to **No line ending**. Then type single letters and press Enter after each one. F for forward, B for back, L for left, R for right, S to stop. The car keeps doing the last thing you typed until you type something else, so S is your stop button.



FIGURE 9.2 The Serial Monitor's two settings for the wheels test: No Line Ending, and 9600 baud. You type in the box at the top and press Enter.

CHILD B49 Plug the USB cable into the board and the laptop. Open the Serial Monitor. Check it shows 9600 baud and **No Line Ending**.

Before the first letter, you both guess.

TALK ABOUT IT

“Which way will the left wheels turn when we type F? And the right wheels? Tick both guesses on the checklist.”

PARENT: LISTEN FOR:

any guess is fine. What matters is checking it. If any wheels go the wrong way, the test box says which wires to swap.

CHILD B50

Type **F** and press Enter. Watch all four wheels. Then type **S** and press Enter.

The Serial Monitor shows every letter the board receives, with where it came from: `Got F from USB`. That’s the board telling you the letter arrived. And one rule for this whole test: **after any change to the motor wires, go back to step B50 and type F again.**

TEST IT

YOU SHOULD SEE:	`Got F from USB`, and all four wheels turn forward.
A INSTEAD:	all four wheels turn backwards.
MEANING:	every motor is wired the other way round for this sketch. That’s normal. It depends on how your motors were made, and nothing is broken.
FIX:	type S, switch off. On all four motors, swap red and black at the motor driver. Switch on and type F again.
B INSTEAD:	one side’s two wheels turn backwards.
MEANING:	that side’s motors are the other way round.
FIX:	type S, switch off. On that side only, swap red and black on both motors. Switch on and type F again.

C INSTEAD:	one wheel turns backwards and the other wheel on its side turns forwards.
MEANING:	that one motor's wires are the other way round.
FIX:	type S, switch off, and swap red and black on the motor that turns backwards. Type F again.
D INSTEAD:	`Got F from USB` appears, but no wheel moves.
MEANING:	the board got the letter, so the problem is in the wheels box or its power.
FIX:	is the switch on? Are the cells in? Are the battery leads and both jumpers clamped (B37 to B44)? Are the six signal wires on the right pins (B29 to B34)? If the wheels twitch and stop and the Serial Monitor shows `Got (newline) from USB`, choose No Line Ending (Chapter 12).

CHILD B51 Type **B**, then **S**. All four wheels turn backwards, and the buzzer beeps while they do.

CHILD B52 Type **L**, then **S**. The left wheels turn backwards and the right wheels forwards: on the floor, the car would spin to the left. Then **R**, then **S**: the other way round.

⏱ TEST IT

YOU SHOULD SEE:	L spins the car to the left.
A INSTEAD:	F is right, but L spins the car to the right.
MEANING:	the left motors are on OUT3 and OUT4, and the right ones are on OUT1 and OUT2.
FIX:	type S, switch off. Move the left motors' wires to OUT1 and OUT2, and the right motors' to OUT3 and OUT4. Then go back to B50 and type F. All four wheels will probably turn backwards now; the first row of the F test fixes that.

CHILD B53 Type **F**. While the wheels turn, type **1**, then **9**, then **S**.

TEST IT

YOU SHOULD SEE:	speed 1 is clearly slower than speed 9.
A INSTEAD:	every speed is the same.
MEANING:	a cap is still on ENA or ENB, or a speed wire is on a pin without the ~ mark.
FIX:	switch off. Check B18 and B19 (caps off) and B29 and B30 (pins 5 and 6).

TALK ABOUT IT

After the speeds: "Speed 1 crawls and 9 runs. What is the pin doing differently?"

PARENT: LISTEN FOR:

"it's on for less of the time" or "it gives less push". A guess is fine. Part 2 has the answer.

The last check: unplug the USB cable

One more test, and it's the one that proves the battery really feeds the board. Type S, then the child unplugs the USB cable. The board's ON light must stay on.

CHILD B54 Type **S**. Then unplug the USB cable from the board. Look at the board's ON light.

⏱ TEST IT	
YOU SHOULD SEE:	the ON light stays on after the USB cable comes out.
A INSTEAD:	the ON light goes out.
MEANING:	the battery reaches the motor driver but not the board. The board was running on the laptop's power all along.
FIX:	switch off; the parent takes the cells out. Redo B37 to B44: the jumper from the +12V screw to Vin, and the jumper from the GND screw to GND. Tug both ends of each. Then the cells go back in and you test again.

This is a stopping point. Switch the holder off. The USB cable is already out.

Stuck for 10 minutes? Run the loop from Chapter 3, then go to the Stage B test page: troniction.com/go/car-test-b.

A short video of the battery wires: the two screws, the two jumpers, and the tug test.

SCAN OR TYPE **troniction.com/go/car-vin**

The wheels test page: what you should see, what you might see instead, and the fix.

SCAN OR TYPE **troniction.com/go/car-test-b**

CHECK YOURSELF

Pause here and answer out loud.

1. You type F and the car spins on the spot. Which stage does that belong to, and what do you change?
2. You unplug the USB cable and the ON light goes out. What's missing?
3. Why does the car sit on a box for this test?

Answers at the back of the book on page 248

/// CHAPTER

10

Stage C · The Bluetooth module

PART OF THE AUDIOBOOK

NOTE

This stage. With all power off, the child puts two resistors into the breadboard and wires the Bluetooth module. Then two tests. **Test C1:** switch on, and the module's light blinks. That's a stopping point. **Test C2:** the phone connects to the module. The HM-10 and the HC-06 connect differently, so C2 has a section for each.

What the Bluetooth module does

The Bluetooth module catches the letters the phone sends through the air and passes each one along a wire to the board's pin 2. It can also talk back: whatever the board sends on pin 3, the module sends to the phone. Stage D uses that to prove the whole journey.

The module needs four wires: two for power, VCC and GND, and two for the letters, TXD and RXD.

Why the wires cross

The module's TXD is the pin it talks on. The board **listens** on pin 2. So TXD goes to pin 2: the module's talking pin to the board's listening pin. The other way round, the board **talks** on pin 3, so pin 3 goes to the module's RXD, its listening pin. "Transmit" and "receive" are named from each part's own point of

view, which is why the wires cross. Get them the wrong way round and nothing breaks; the car just never hears the phone.

Why the two resistors

The board talks at 5 volts. The module's listening pin is built for about 3.3 volts. So the board's talking wire, from pin 3, goes through the two resistors, which share the 5 volts between them and hand the module about 3.3. That's called a voltage divider, and Part 2 explains how it works. The other wire, from the module to the board, needs nothing: the board hears the module's 3.3 volts as "on" without help.

Why the module is on pins 2 and 3

The 2021 car had its module on pins 0 and 1, the USB cable's pins, which blocked every new upload. This edition moves it to pins 2 and 3. That wiring is new: tested on a computer, with the module's behavior from its datasheet, but not yet driven in a car.

Steps C1 to C8: the wiring

Each step names an exact breadboard hole. There's a short video of them at troniction.com/go/car-c.

⚠ WARNING

All power off before you start: type S, unplug the USB cable, and switch the battery holder off. The module's wires go near the board's 5V pin, and a slip with the power on can damage the module.

CHILD C1 Check: the USB cable is out, and the holder's switch is off.

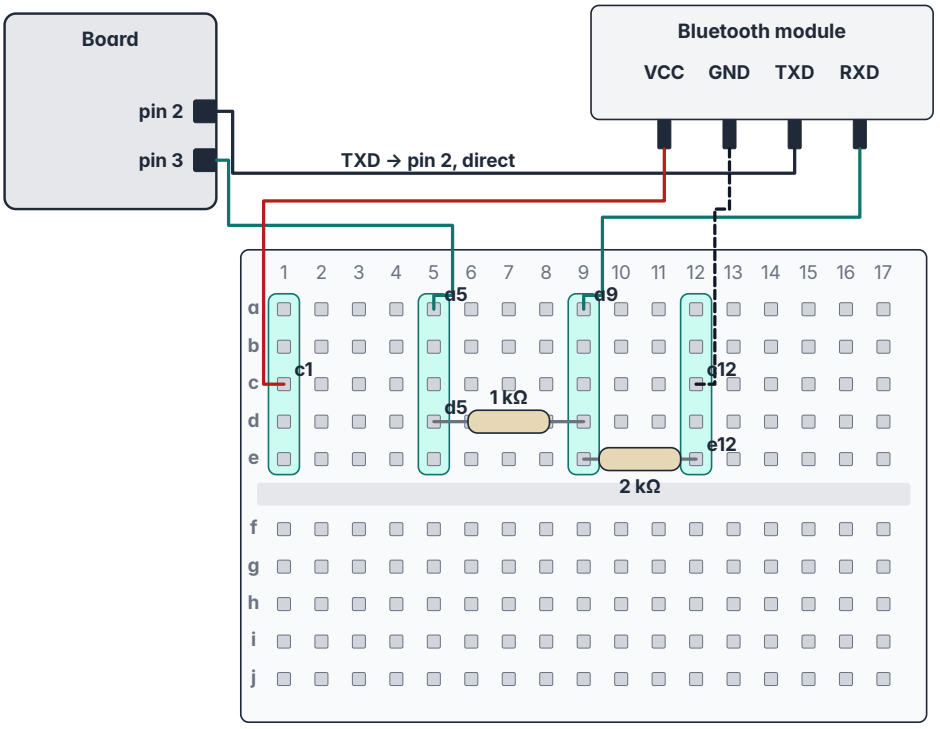


FIGURE 10.1 The Bluetooth module and the two resistors. Pin 3 goes to column 5, through the 1 kΩ resistor to column 9, where the module's RXD connects; the 2 kΩ resistor joins column 9 to GND in column 12. The module's TXD goes straight to pin 2. VCC goes to column 1 and GND to column 12.

CHILD C2 The **1 kΩ** resistor (check its label): one leg in hole **d5**, the other in hole **d9**.

CHILD C3 The **2 kΩ** resistor (or the 2.2 kΩ): one leg in hole **e9**, the other in hole **e12**, the GND column.

CHILD C4 With a male-to-male jumper: the board's **pin 3** to hole **a5**.

CHILD C5 With a male-to-female jumper: the module's **VCC** to hole **c1**, the 5V column.

CHILD C6 With a male-to-female jumper: the module's **GND** to hole **c12**, the GND column.

CHILD C7 With a male-to-female jumper: the module's **TXD** straight to the board's **pin 2**.

CHILD C8 With a male-to-female jumper: the module's **RXD** to hole **a9**, the column where the two resistors meet.

PARENT C9 Check, reading each line aloud while the child points (the figure is with C2 to C7):

- **The 2 kΩ from e9 to e12. This is the only check it gets: no test can see it.** The child points to both legs.
- 1 kΩ from d5 to d9.
- Pin 3 to a5.
- VCC to c1, GND to c12.
- TXD to pin 2.
- RXD to a9.
- Nothing from the module goes to pins 0 or 1.

Test C1: the module's light

Switch on, with the USB cable still out. The module's light should blink. A blinking light means the module has power and is waiting for a phone. Only the Bluetooth module's wiring can fail this test, because everything else already passed in Stage B.

▲ CHILD C10 Switch the holder **on**. Look at the module's light.

⏱ TEST IT	
YOU SHOULD SEE:	the module's light keeps blinking on and off (an HM-10 blinks about once a second).
A INSTEAD:	the light stays off.
MEANING:	the module has no power.
FIX:	switch off. Check the module's VCC is in column 1 and its GND in column 12 (C5, C6), and that the board's 5V jumper is still in a1 and its GND jumper in a12 (B21, B22). Then switch on again.

B INSTEAD:	the board's ON light goes out or flickers when you switch on.
MEANING:	a short: something joins 5V to GND.
FIX:	switch off at once. Look for a 5V wire in the GND column, or a GND wire in the 5V column.

This is a stopping point. Switch off if you're stopping here.

Test C2: the phone connects

Now the phone. This is where the two modules differ. In short: an HM-10 connects only from inside the Troniction remote, a web page, and never from the phone's Bluetooth settings. An HC-06 is paired in the phone's Bluetooth settings first, with a code, and then connected from the Serial Bluetooth Terminal app.

TABLE 10.1 Test C2 with each module. Only these rows differ

	HM-10	HC-06
Phone	Android with Chrome, or iPhone with Bluefy	Android only
Pair in the phone's Bluetooth settings?	**No.** Never	**Yes**, first, with the code 1234
Connect from	The remote: troniction.com/go/car-remote	The Serial Bluetooth Terminal app
How you know it's connected	The remote says *connected*; the module's light goes steady	The app says *Connected*; the module's light goes steady

IF YOU HAVE THE HM-10

Open the remote page in the right browser: Chrome on Android, or the Bluefy browser on an iPhone. Not Safari. Tap Connect, choose the module from the list, and the module's light goes steady.

CHILD C11 On the phone, open **Chrome** (Android) or **Bluefy** (iPhone). Go to **troniction.com/go/car-remote**.

CHILD C12 Tap **Connect**. If the phone asks to allow Bluetooth or nearby devices, the parent reads the question and says yes.

CHILD C13 A list of devices appears. Tap your module: it's usually called **HMSoft**. Some modules use another name, such as **DSD TECH**; the name is on the module's bag or listing.

MYTH

"You connect the car in your phone's Bluetooth settings."

WRONG **That's wrong for the HM-10.**

The remote page connects to it directly. Your phone may show the module in its Bluetooth settings, but don't pair it there. A phone that has paired it can keep hold of it, and then the remote can't find it. If you already paired it: open Settings, then the module, then **Forget**. Then try the remote again.

TEST IT

YOU SHOULD SEE:	the remote says *connected*, and the module's light goes from blinking to steady.
A INSTEAD:	on Android, the list says no devices found.
MEANING:	Chrome isn't allowed to look for Bluetooth devices.
FIX:	turn on the phone's **Location**, and in the phone's app settings, give Chrome the **Nearby devices** permission. Then tap Connect again.
B INSTEAD:	on an iPhone, nothing happens when you tap Connect, or the page says Bluetooth isn't supported.
MEANING:	you're in Safari, which can't use Bluetooth from a web page.
FIX:	open the same address in the **Bluefy** browser.

IF YOU HAVE THE HC-06

First pair the module in the phone's Bluetooth settings, using the code 1234. Then open the Serial Bluetooth Terminal app and connect to it there.

CHILD C11 On the Android phone, open **Settings → Bluetooth** and scan for new devices. Your module appears, usually as **HC-06** (some are called **linvor**). Tap it.

CHILD C12 When the phone asks for a PIN, type **1234**. (If 1234 is refused, try **0000**.) The module now shows as paired.

CHILD C13 Open the **Serial Bluetooth Terminal** app. Open its menu, choose **Devices**, then the **Bluetooth Classic** tab, and tap your module. The app says *Connected*.

⏱ TEST IT

YOU SHOULD SEE:	the app says *Connected*, and the module's light goes from blinking to steady.
A INSTEAD:	the module never appears when the phone scans.
MEANING:	the module has no power, or it's already connected to another phone.
FIX:	check that its light is blinking (test C1). Turn off Bluetooth on any other phone that connected to it before.
B INSTEAD:	the module is paired, but the app can't connect.
MEANING:	another app, or another phone, is holding the connection.
FIX:	close other Bluetooth apps, switch the car off and on, and connect again from the app.

Connected isn't the end

TALK ABOUT IT

"The light went steady. Does that mean the car will move? How could we check?"

PARENT: LISTEN FOR:

"not yet", or "we'd have to press a button and watch". Stage D checks it.

MYTH

> “The phone says connected, so the car should move.”

WRONG **That’s wrong.**

Connected means the phone and the Bluetooth module can hear each other: box 1 and box 2. Whether the letter gets from the module to the board, and whether the board does anything with it, are two separate questions. Stage D answers both.

The next stopping point is after the drive.

Stuck for 10 minutes? Run the loop from Chapter 3, then go to the Stage C test page: troniction.com/go/car-test-c.



A short video of the two resistors and the module’s four wires going in.

SCAN OR TYPE troniction.com/go/car-c



A short screen recording on both kinds of phone: opening the remote, allowing Bluetooth, and connecting.

SCAN OR TYPE troniction.com/go/car-connect



The Troniction remote, for the HM-10. Open it in Chrome on Android or Bluefy on an iPhone.

SCAN OR TYPE troniction.com/go/car-remote



Serial Bluetooth Terminal, the free Android app for the HC-06.

SCAN OR TYPE troniction.com/go/car-bt-terminal



The Stage C test page, with tests C1 and C2 for both modules.

SCAN OR TYPE **troniction.com/go/car-test-c**

CHECK YOURSELF

Pause here and answer out loud.

1. The module's light is off. Which wires do you check first, with the power off?
2. With an HM-10, an Android phone's remote page says no devices found. What do you check?

Answers at the back of the book on page 248

/// CHAPTER

11

Stage D · The round trip, then drive

PART OF THE AUDIOBOOK

NOTE

This stage. No new wires. With the car on its box, you watch a letter travel from the phone to the board, and the car's answer come back to the phone. Then the car goes on the floor, and the child drives. **The end of the first drive is the last stopping point of Part 1.**

Connected, arrived, answered

In Stage C, the phone said *connected*. That proved the phone and the Bluetooth module can hear each other. Stage D proves two more things, one at a time.

Arrived. When you press Forward, the letter F reaches the board. You see that in the Serial Monitor on the laptop: the board prints `Got F from phone`.

Answered. The board then answers the phone through the module, and the phone shows `Car heard: F`. That answer travels from pin 3, through the 1 kΩ resistor, into the module's listening pin. So if the answer arrives, the path through the 1 kΩ resistor is right. The 2 kΩ resistor, the one that protects the module, can't be tested this way: the answer gets through without it. That's why step C9 checks it by eye, hole by hole.

Connected, arrived, answered: three separate facts. When all three are true and the wheels turn, every box in the car works.

The round-trip test

Keep the car on its box for this. Plug in the USB cable and open the Serial Monitor at 9600, so the laptop is your window into the board. Switch on, and connect the phone as in test C2. Then send an F from the phone and watch both screens.

There's a short video of the round trip at troniction.com/go/car-d.

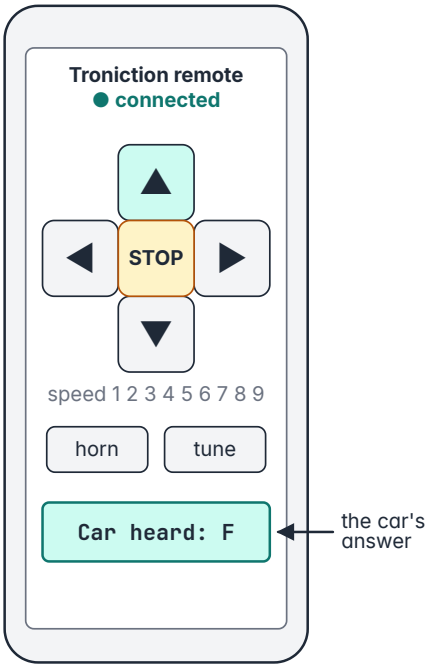
> ⚠ WARNING
>
> **The car stays on its box** until the round-trip test has passed. It will move when the letter arrives.

▲ CHILD D1 Car on its box. Switch on. Plug in the USB cable and open the Serial Monitor at **9600**.

CHILD D2 Connect the phone to the module, as in test C2. The module's light is steady.

WITH THE HM-10 AND THE REMOTE

CHILD D3 On the remote, press and hold **Forward** for a second, then let go.



Drawn mock-up of the planned remote

FIGURE 11.1 The remote after pressing Forward. Under the buttons, its reply line shows Car heard: F, the car's answer.

The remote sends F once when you press Forward, and S, stop, when you let go. So you see both letters arrive. If the car doesn't answer `Car heard: S`, the remote sends the S again, up to twice more, so a stop isn't lost.

WITH THE HC-06 AND SERIAL BLUETOOTH TERMINAL

The terminal app normally adds an invisible "new line" letter after everything you send, and to this car any letter it doesn't know means stop. So first, switch that off.

CHILD D3 In Serial Bluetooth Terminal, open **Settings**, then the **Send** section, and set **Newline** to **None**. Go back to the terminal.

CHILD D4 Type **F** in the box at the bottom and tap the send arrow. Then type **S** and send it.

WHAT YOU SHOULD SEE



FIGURE 11.2 The Serial Monitor during the round trip. Each letter from the phone is shown as it arrives: Got F from phone, then Got S from phone.

CHILD D5 Read both screens aloud: the Serial Monitor on the laptop, and the phone. The parent checks them against Figure 11.2.

⏱ TEST IT

YOU SHOULD SEE:	the Serial Monitor shows `Got F from phone`, the phone shows `Car heard: F`, and the wheels turn forward until the S arrives.
A INSTEAD:	`Got F from phone` appears, and the wheels turn, but the phone never shows `Car heard`.
MEANING:	the letter arrives, but the answer doesn't get back. The answer travels from pin 3, through the 1 kΩ resistor, to the module's RXD.
FIX:	all power off: unplug the USB cable and switch off. Check the 1 kΩ (C2), the pin 3 wire (C4) and the RXD wire (C8) against the figure in Chapter 10. Then test again.

B **INSTEAD:** the phone is connected, but nothing at all appears in the Serial Monitor.

MEANING: the letter doesn't reach the board. The wire from the module's TXD to pin 2 is missing or on the wrong pin, or TXD and RXD are swapped.

FIX: all power off. Check C7 (TXD to pin 2) and C8 (RXD to a9).

C **INSTEAD:** with an HM-10, the wires check out, but each press shows only a line with an odd code in brackets, such as `Got (code …) from phone`, and never `Got F`.

MEANING: the module talks at a different speed from the sketch. The sketch expects 9600, the factory speed of the HM-10. Some HM-10s with newer software (versions V700 to V709) are reported to start at 115200 instead.

FIX: near the top of the sketch, change the module's speed, `BT_BAUD`, from `9600` to `115200`, and upload it again (Chapter 6). If `Got F from phone` now appears, that was it. The car drives at that speed, but the board can miss a letter now and then. To set the module back to 9600 for good, it needs the setup command `AT+BAUD0` (and `AT+VERS?` reads its software version): *Make It Connect* has the bridge sketch and the steps, at troniction.com/go/car-mic.

D **INSTEAD:** the letters arrive and are answered, but the wheels don't turn.

MEANING: the phone, the module and the board all work. The problem is in the wheels box or its power.

FIX: is the holder switched on? Then redo the wheels test from Chapter 9.

TALK ABOUT IT
“The letter came back. Which boxes do we know work now?”

PARENT: LISTEN FOR:
all four: the phone, the Bluetooth module, the board and the wheels.

The letters

Every button sends one letter. Here’s the whole list, in words. F for forward, B for back, L for left, R for right. S for stop. The digits 1 to 9 set the speed, 1 the slowest and 9 nearly the fastest, and the small letter q is the very fastest. Capital V sounds the horn, and capital X plays *Happy Birthday*. Any letter the car doesn’t know stops it. The car keeps doing the last thing you told it until the next letter arrives.

TABLE 11.1 The letters the car understands

LETTER	WHAT THE CAR DOES
`F`	Forward
`B`	Backward, with a beep
`L`	Spin left
`R`	Spin right
`S`	Stop (any letter the car doesn’t know also stops it)
`1` to `9`	Speed, slowest to nearly fastest; it keeps moving at the new speed
`q`	Fastest
`V`	Horn, until the next letter
`X`	*Happy Birthday* (see the warning)

With the HC-06, you can put the letters on the app’s row of buttons along the bottom of the screen, so you don’t have to type: press and hold a button to edit it. Put S on the first one. Remember that a button sends its letter once, so the car keeps going until you press S.

⚠ WARNING

Stop the car before you play the tune. While *Happy Birthday* plays, about 16 seconds, the car keeps doing what it was doing, and it can't hear any letter you send. When the song ends, it plays again, until a new letter arrives. So: S first, then X, and S again to end the song.

The first drive

Now the car goes on the floor. Find a clear space, away from stairs and from anything that could fall.

CHILD D6 Send **S**, then unplug the USB cable. The holder stays switched on.

CHILD D7 Lift the car off its box and put it on the floor.

TOGETHER D8 Drive. Try speed 1 and speed 9, the horn, and a spin each way.

TALK ABOUT IT

"It drifts a little to one side on the floor. Is something wired wrong?"

PARENT: LISTEN FOR:

"probably not". No two motors are exactly the same, so one side runs a bit faster. If it *spins on the spot* when you press F, that's different, and the wheels test in Stage B would have caught it.

The first time it moves is the point of the whole book. Take a video. If you share it online, keep your child's face out of it: hands and the car are enough.

When you've finished, switch the holder off. The parent takes the cells out if the car won't be used again for a few days.

You can put a new sketch on the board any time, with the Bluetooth module still connected. Part 4 shows how.

That's Part 1 finished: a car built and driven. Stuck for 10 minutes on the way? Run the loop from Chapter 3, then go to Chapter 12 or the Stage D test page: troniction.com/go/car-test-d.



A short video of the round trip and the first drive.

SCAN OR TYPE **troniction.com/go/car-d**



The Stage D test page: connected, arrived, answered, and what to do when one of them fails.

SCAN OR TYPE **troniction.com/go/car-test-d**

CHECK YOURSELF

Pause here and answer out loud.

1. The Serial Monitor shows `Got F from phone` and the phone shows `Car heard: F`, but the wheels don't turn. Which boxes are fine?
2. Predict: you press X while the car is driving forward. What happens?

Answers at the back of the book on page 248

/// CHAPTER

12

If it doesn't move

PART OF THE AUDIOBOOK

Find the stage first

When the car doesn't do what it should, the question isn't "what's wrong with it?" It's "**which box** is it in?" And you already have the tool that answers that: the stage tests. Each one can fail only for a reason inside its own box.

So start at the beginning and run the tests in order, until one fails:

1. **Stage A.** Plug in the USB cable and open the Serial Monitor at 9600. Does the board say `Car ready`? (Press RESET if the screen is empty.)
2. **Stage B.** Car on its box, switch on, No line ending. Does F turn all four wheels forward? Does the ON light stay on when you unplug the USB cable?
3. **Test C1.** Does the Bluetooth module's light blink when you switch on?
4. **Test C2.** Does the phone connect, and the light go steady?
5. **Stage D.** Does `Got F from phone` appear, and `Car heard: F` come back?

The first test that fails tells you the box. Then the table in this chapter lists the causes for that test, the wiring first. It's almost never the sketch: the sketch is the same on every car, and it passed its tests before this book was printed.

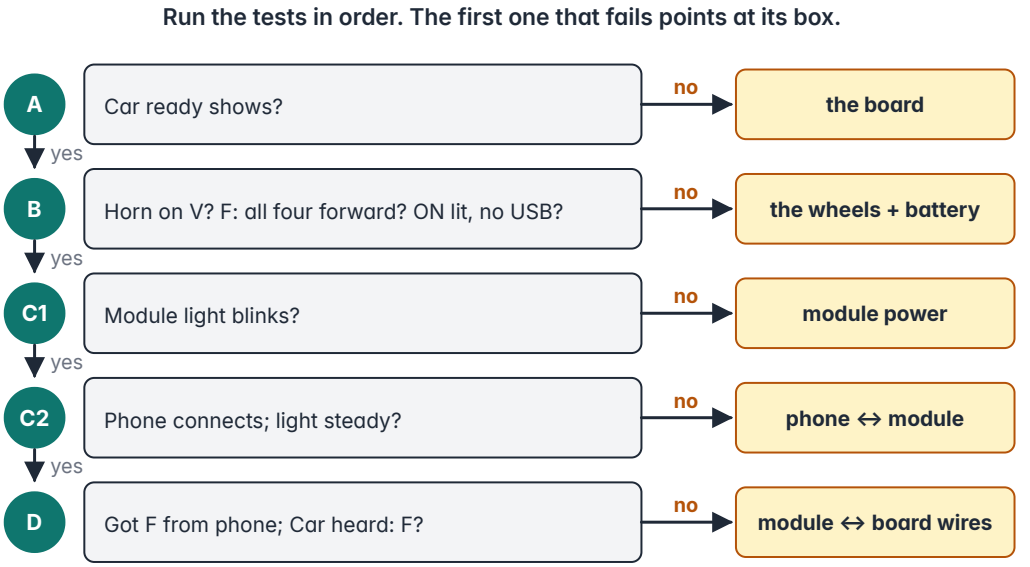


FIGURE 12.1 Finding the fault on the four boxes. Run the stage tests in order; the first one that fails points at its box.

And every time, use the loop from Chapter 3: look, which box, change one thing, test again, say what you found.

Three worked examples

"It goes backwards when I press Forward." Which test? The wheels test, Stage B: type F in the Serial Monitor with the car on its box, and watch all four wheels. Which box? The wheels. If all four turn backwards, switch off and swap red and black on all four motors. If only one side does, swap that side. Then type F again: after any change to the motor wires, you go back to F.

"The phone says connected, but nothing happens." Which test? Stage D. Plug in the USB cable, open the Serial Monitor and press Forward. If `Got F from phone` doesn't appear, the letter isn't reaching the board: check the wire from the module's TXD to pin 2. If it does appear and the wheels still don't turn, the board heard you, so the problem is in the wheels box: is the switch on? Then redo the wheels test.

"It drove yesterday. Today it stops and restarts when it starts moving." Which box? The battery's power line. Tired cells are the usual cause, exp-

lained in "What a tired battery looks like", page 89. Charge the cells and try again.

What a tired battery looks like

A battery doesn't go from full to dead in one step. As two 18650 cells run down, their voltage drops. The board needs about 7 volts at its Vin pin, and two cells give about 7.4 when they're neither full nor empty. That's the very bottom of what the board wants. So the board gives up before the cells are truly flat.

It shows up at the moment the motors start. Starting a motor takes a big gulp of power, the voltage dips for a moment, and the board restarts. The car stops dead, the L light flickers, and everything you sent before is forgotten. Then it works again for a bit, and does it again.

If that's what you see, it isn't a wiring fault. Switch off, and the parent charges the cells.

The full table

Each row of Table 12.1 names a symptom, the test that catches it, and the fix, with the wiring causes first. The same table is online at troniction.com/go/car-stuck.

TABLE 12.1 If it doesn't move: symptom, test, cause and fix, circuit causes first

SYMPTOM	TEST	MOST LIKELY CAUSE	FIX
ON light lit, no port on the laptop	A	Charge-only USB cable; clone needs the CH340 driver	Another cable; install the driver (Chapter 6)
L light blinks slowly forever, monitor empty	A	The sketch never uploaded	Upload again; wait for "Done uploading"
L light on, but no `Car ready` even after RESET	A	Nothing is proved: the L light can be on for other reasons	Upload again (A3)
Strange symbols in the Serial Monitor	A	Serial Monitor not at 9600	Choose 9600; press RESET
Upload error ("not in sync", "port busy")	A	Wrong board or port; another program using the port	Tools → Board: Arduino Uno; pick the right port; replug
A locked school or work laptop won't install the IDE	(prep)	No permission to install software	Use a laptop you're allowed to install on (Chapter 1)

SYMPTOM	TEST	MOST LIKELY CAUSE	FIX
No light anywhere when you switch on	B	Cells backwards or not seated; battery lead loose	Cells out (parent), check + marks; tug the leads
No horn on V, but `Got V from USB`	B	Buzzer wires	Check B21–B25
Motor driver's light on, board's ON light off (USB out)	B	The Vin or GND jumper is missing or loose	Check B37–B44; tug both jumpers
`Got F from USB`, but no wheel turns	B	Switch off; battery leads or jumpers loose; signal wires on wrong pins	Switch on; check B37–B44; check B29–B34
All four wheels turn backwards on F	B	Every motor is the other way round for this sketch (normal)	Swap red and black on all four; type F again
One side turns backwards; the car spins	B	That side's motors are the other way round	Swap that side's red and black (both motors); type F again
One wheel fights the other on its side	B	That one motor's wires are the other way round	Swap red and black on the one turning backwards; type F again
F is right, but L turns right	B	The left motors are on OUT3/OUT4, the right ones on OUT1/OUT2	Move the left motors to OUT1/OUT2 and the right to OUT3/OUT4; type F again (expect all four backwards, then the row above)
Every speed the same	B	Cap still on ENA or ENB; speed wire on a pin without `~`	Check B18–B19 (caps), B29–B30 (pins 5, 6)
Wheels twitch and stop; monitor shows `(newline)`	B	Line ending isn't "No line ending"	Choose No Line Ending
ON light goes out when the USB cable comes out	B	The jumper from +12V to Vin, or from GND to GND, is missing or loose	Redo B37–B44
Motors hum but don't turn on the floor	B / D	Speed too low for the floor, or tired cells	Try speed 9; charge the cells
Module light off	C1	Module VCC or GND not in its column; board's 5V/GND jumpers out	Check C5, C6, B21, B22
ON light dims or goes out when you switch on with the module	C1	A short: 5V joined to GND	Switch off at once; find the wire in the wrong column

SYMPTOM	TEST	MOST LIKELY CAUSE	FIX
HM-10: "no devices found" (Android)	C2	Location off, or Chrome lacks Nearby devices	Location on; allow Nearby devices for Chrome
HM-10: nothing happens on Connect (iPhone)	C2	You're in Safari	Open the remote in Bluefy
HM-10: the remote can't find it after pairing in Settings	C2	The phone is holding the module	Settings → the module → Forget
HC-06: never appears in the scan	C2	No power, or connected to another phone	Check C1; turn off the other phone's Bluetooth
HC-06: PIN refused	C2	The module's code isn't 1234	Try 0000
Connected, but no `Got F from phone`	D	TXD not on pin 2; TXD and RXD swapped	Check C7, C8
HM-10: each press shows `Got (code …)`, never `Got F`	D	The module talks at 115200 (some newer HM-10 software)	Set `BT_BAUD` to 115200 and upload (Chapter 11)
`Got F from phone`, but no `Car heard`	D	The 1 kΩ, the pin 3 wire or the RXD wire	Check C2, C4, C8 against the Chapter 10 figure. (The 2 kΩ can't show up in any test: C9 checks it by eye)
HC-06: the car jerks and stops after each letter	D	The app sends a newline	Settings → Send → Newline: None
Beeping won't stop	D	It's the reversing beep, or the horn	Send S
The car ignores you while the tune plays	D	*Happy Birthday* blocks everything for about 16 s, then repeats	Wait for the end, then send S. Stop before X next time
It stops and restarts when it starts moving	power	Tired cells (the board's power dips)	Charge the cells

Still stuck

If none of this finds it, do three things in this order. First, run the loop from Chapter 3 one more time, out loud, and write down exactly what you see at each stage test. Second, go to troniction.com/go/car-stuck, which has this table, the stage test pages and short videos. Third, write to me: the address is

on that page. Tell me which stage test fails and what you see, and I'll help. If it turns out the book was wrong, it gets a dated correction.

The symptom index online: symptom, stage, fix, with the stage test pages and videos.

SCAN OR TYPE **troniction.com/go/car-stuck**

CHECK YOURSELF

Pause here and answer out loud.

1. The car drove yesterday. Today it stops and restarts every time it starts moving. What changed?
2. Two wheels turn and two don't. Which stage test do you redo first, and why?
3. Which box? (a) The module's light is off. (b) `Got F from USB` appears, but the left wheels stay still. (c) The remote says *connected*, but the Serial Monitor shows nothing when you press Forward.

Answers at the back of the book on page 248

/// CHAPTER

13

What next

PART OF THE AUDIOBOOK

The car is a beginning

You built a car, and it drives. Your child put in every wire, and the two of you found out, test by test, that every box works. That's the hard part done.

The car isn't the end of it. It's a platform: a board with free pins, a sketch you can change, and a remote that sends letters you can give new meanings to. The rest of this book is about understanding it and making it yours.

The rest of this book

Part 2, Why it works, explains every part of the car in the order you built it: volts and a shared ground, the board, the motor driver's four switches, how "speed" is really a pin switching on and off very fast, the motors and their gears, the battery, and how a letter travels from the phone to the board and back. Every chapter ends with something to try on your own car, using only the Serial Monitor.

Part 3, Find the fault, teaches the method an expert uses on any broken machine, with real faults from real cars, including the speed bug from 2016 told as a detective story, and ten broken cars for you to diagnose.

Part 4, Program the car, opens the sketch. You'll read it, predict what it does, find a real bug that has been in it for years, find out why the car can't hear you while it sings, and add a command of your own.

Part 5, For parents, clubs and teachers, is for running the build with one child or with twelve, and for checking a week later what stuck.

Where to go from here

Two places, if your child wants more.

Make It Connect is Troniction's book about the Bluetooth link itself: every way it fails, why, and the fix. It covers both the HM-10 and the HC-06, and its car chapter uses the same pins as this car.

The weekend build adds new tricks to this same car. It's a short Troniction course, and its page says exactly what it adds and what it costs.

Both are at troniction.com/go/car-next.

The free illustrated book, in PDF and EPUB.

SCAN OR TYPE **troniction.com/go/car-book**

Where to go next: *Make It Connect*, the weekend build, and the rest of Troniction.

SCAN OR TYPE **troniction.com/go/car-next**

TALK ABOUT IT

"Which part of the car would you change first, and which of the four boxes does it belong to?"

PARENT: LISTEN FOR:

any part, as long as it comes with a box. There's no wrong answer here, and no answer at the back.

2

/// PART 2

Why it works

Your car drives. This part explains why, one part at a time, in the order you built it: the electricity, the board, the motor driver, the speed trick, the motors, the battery and the Bluetooth module.

The child leads here and the parent reads along. Every chapter starts from the car on your table, not from a textbook, and every chapter ends with something to try on that car. The experiments need nothing but the car, the laptop and the letters you already know from the Serial Monitor. Nobody edits any code until Part 4.

When a chapter uses a comparison (water in a pipe, a crowd at a door), it also tells you where the comparison stops being true. A comparison is a ladder: it gets you up to the idea, and then you step off it.

/// CHAPTER

14

Volts, amps and a shared ground

Look at the car on your table and count the wires that carry power. There are only a few: two from the battery holder to the motor driver, two more across to the board, and the four motor leads. Everything else, the six thin wires between the board and the motor driver and the two data wires to the Bluetooth module, carries almost no power at all. Those are **signal** wires. They carry messages, not muscle.

This chapter gives you just enough electricity to understand the difference, and to understand the one wire that beginners leave out because it seems to do nothing: the shared ground.

Three words: volts, amps, ohms

Electricity in a wire has three numbers that matter, and each has its own unit.

TABLE 14.1 The three numbers, each with two examples from your car

WORD	WHAT IT MEASURES	UNIT	TWO EXAMPLES ON YOUR CAR
Voltage	How hard the electricity is pushed	volts (V)	The USB cable gives the board 5 V. The battery gives about 7.4 V
Current	How much electricity flows past a point each second	amps (A), or milliamps (mA): 1,000 mA = 1 A	A board pin can safely give about 20 mA. One gear motor running freely takes about 150 mA
Resistance	How hard it is for electricity to get through	ohms (Ω)	The two resistors on the Bluetooth module's wire are 1,000 Ω and 2,000 Ω. A jumper wire is close to 0 Ω

They are tied together by one rule. **Current = voltage ÷ resistance.** Push harder and more flows; make it harder to get through and less flows.

Two worked examples, both from your car:

- **The resistors on the Bluetooth wire.** When the board's pin 3 is at 5 V, the current goes through 1,000 Ω and then 2,000 Ω to ground: 3,000 Ω in all. 5 ÷ 3,000 = 0.0017 A, which is 1.7 mA. That is a trickle, and that is the point: signal wires carry tiny currents.
- **A motor.** A gear motor's coil is a few ohms (that's a rough figure, from how much it draws when stalled, not a measurement). Put about 5 V across a few ohms and you get more than an amp. That is why a motor stuck against a chair leg draws over 1 A: the Adafruit page for this motor gives 1.1–1.5 A when it can't turn.

The water picture, and where it breaks

The best-known picture of electricity is water in pipes. It is worth using, because it gets two things right that are hard to see any other way.

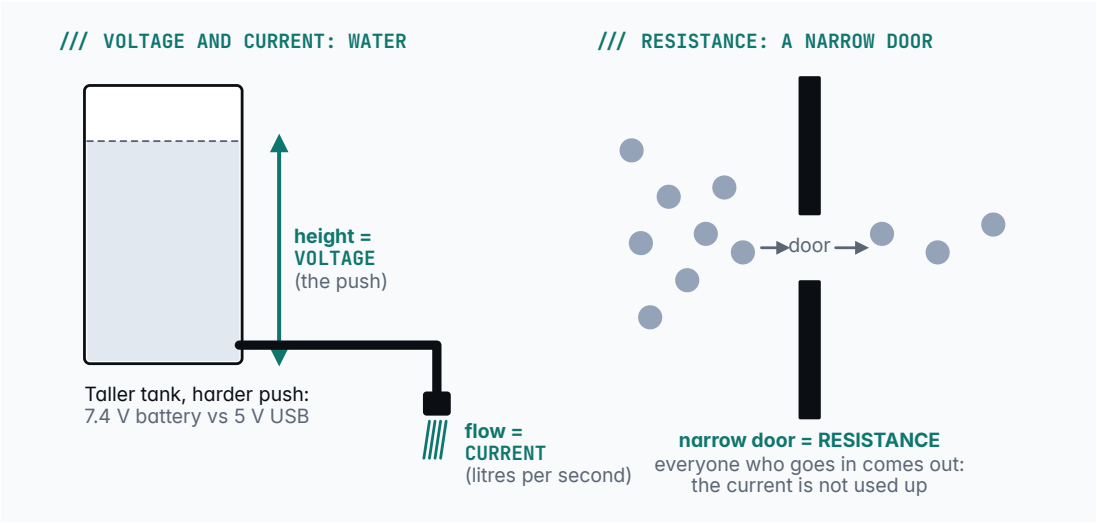


FIGURE 14.1 Two pictures of a circuit. On the left, a water tank: the height of the water is the voltage, the flow through the pipe is the current. On the right, a crowd going through a narrow door: the narrow door is the resistance, and everyone who goes in also comes out.

Voltage is like water pressure. A tall tank pushes water out of a tap harder than a short one. Your 7.4 V battery is a taller tank than the 5 V USB cable.

Pressure is always *between two places*: the top of the water and the tap. Remember that; it comes back at the end of this chapter.

Current is like the flow: how many litres pass the tap each second. More pressure, more flow.

Resistance is like a narrow door in a crowd. Picture a crowd leaving a stadium. A wide gate lets lots of people through each second; a single narrow door lets far fewer through, however hard the people at the back push. The 2,000 Ω resistor is a narrow door. The jumper wire is a wide-open gate.

> ⓘ NOTE
> **Where these pictures break.**
>
> - **Water pipes leak; wires don't.** Cut a water pipe and water pours out. Cut a wire and the electricity simply stops: nothing spills. Electricity only flows when there is a complete loop back to the battery.
> - **The water in a tank gets used up; the electricity in a wire doesn't.** In the crowd picture, every person who goes in through the door comes out the other side. Nobody is "used up" by the door. Electricity is the same: the current going *into* a motor equals the current coming *out* of it. What the motor uses is the battery's **energy**, not its current. Chapter 19 comes back to this, because it is the reason people misread a tired battery.

Signal wires and power wires

Now the first big idea of the car. **A board pin can signal, but it cannot push.**

The board's chip (the ATmega328P) is rated for 40 mA on one pin as an absolute maximum: a line you must not cross. The board's own documents recommend 20 mA. One gear motor running freely wants about 150 mA. Stalled, it wants over 1,000 mA.

So a pin is seven times too weak for a free-running motor and about fifty times too weak for a stuck one. Wire a motor straight to a pin and you don't get a slow car; you get a damaged board.

That is the whole reason the motor driver is in the car. The battery feeds the motor driver directly with the big current. The board only tells it *which way* and

how hard, over six signal wires that carry almost nothing. Chapter 16 opens the motor driver up.

Look at your own car: the battery leads are often thicker than the jumper wires. A thicker wire is a wider door. Wires that carry big currents are made thicker so they don't waste power warming up.

The shared ground

Here is the wire that seems to do nothing. The battery's black lead goes to the motor driver's `GND` terminal, and **also** to the board's `GND` pin. Why does the board's ground need to be joined to the motor driver's?

Go back to the water picture. Pressure is always measured *between two places*. A voltage is the same: it is always "this many volts above *something*". That something is called **ground**. When the board puts its pin 12 at 5 V, it means "5 V above the board's ground".

The motor driver reads its `IN1` pin and asks: is this above 2.3 V? (That is the number in the L298N datasheet: anything above 2.3 V counts as HIGH.) But 2.3 V above *what*? Above the motor driver's own ground. If the two grounds are joined, the board's "5 V above my ground" is also "5 V above the motor driver's ground", and the message arrives.

If the grounds are not joined, there is no loop for the signal. The signal current goes out along the `IN1` wire and has no way back. It is like phoning someone with only half a phone line. The motor driver sees nothing it can trust, and the wheels don't turn.

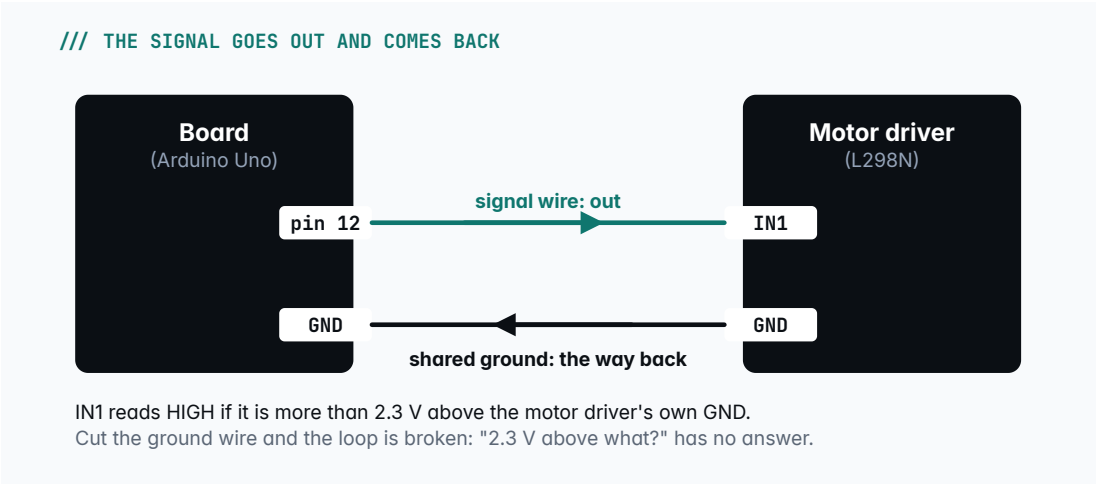


FIGURE 14.2 The signal from pin 12 to IN1 goes out on the signal wire and comes back on the shared ground wire. Cut the ground wire and the loop is broken.

Two examples on your car, both relying on the same shared ground:

- **The board talking to the motor driver**: pins 5, 6 and 9–12 to `ENA`, `ENB` and `IN1`–`IN4`. Every one of them returns through the ground wire you joined.
- **The Bluetooth module talking to the board**: the module's `TXD` to pin 2. The module's `GND` wire to the board's `GND` is its way back.

> ⓘ NOTE
> **Where the "ground is zero" idea breaks.** Ground is not a magic place where electricity disappears. It is just the wire everybody agrees to measure from. Two boards with separate grounds are like two people measuring the height of a table, one from the floor and one from the roof: both numbers are right, and they don't agree.

⏱ TEST IT

TRY THIS:

car on its box, wheels off the table, USB cable in, the battery switch **off**. In the Serial Monitor, send `F`. Then switch the battery on. Watch the lights on the board and on the motor driver.

YOU SHOULD SEE: with the switch off, the board's `ON` light is lit (the USB cable feeds it) but the motor driver's red light is off, and the wheels stay still, even though the Serial Monitor shows `Got F from USB`. When you switch on, the motor driver's light comes on and the wheels start turning.

A INSTEAD: the wheels don't start when you switch on.

MEANING: the board was holding the `F` all along; it only lacked a motor driver with power to pass it to. If nothing turns once the motor driver's light is on, a signal wire or the shared ground is loose: go back to the wheels test in Chapter 9. Send `S` to stop, then switch off.



How much current can an Arduino pin supply? The numbers, and what happens past them.

SCAN OR TYPE **troniction.com/go/car-pin-current**

CHECK YOURSELF

1. Why can't the board run a motor straight from one of its pins? Use the two numbers from this chapter.
2. Someone removes the black wire between the motor driver's `GND` and the board's `GND`, but leaves the battery connected to the motor driver. Predict what the motors do when you send `F`, and say why.
3. Use the rule current = voltage ÷ resistance: if pin 3 were at 5 V with only the 1,000 Ω resistor to ground, how much current would flow? Is that still a "signal-sized" current?

Answers at the back of the book on page 248

/// CHAPTER

15

The board

Pick the board up off the car (or just look at it where it sits). It is an **Arduino Uno**, and almost all of it is helpers: a USB socket, a power socket, a regulator that turns the battery's volts into a steady 5 V, a reset button, the rows of black sockets you pushed the wires into, and a little light marked `L`. The part that actually runs your sketch is the long black chip with 28 legs: an **ATmega328P**. That chip is the **microcontroller**: a whole small computer on one chip.

What a microcontroller does

It does three things, over and over, for as long as it has power:

1. **Read.** Has a letter arrived from the USB cable or from the Bluetooth module?
2. **Decide.** The sketch says what each letter means: `F` means forward, `5` means a speed.
3. **Set its pins.** Pins 9–12 go HIGH or LOW to choose direction; pins 5 and 6 set the speed.

People often call the board the car's brain. Like a brain, it decides. Unlike a brain, it only does exactly what the sketch says, and **one step at a time**. It is fast enough to seem to do several things at once (the chip does about 16 million simple steps a second), but it never does. Chapter 30 shows you the moment this matters: when the car plays *Happy Birthday*, it stops listening to the steering until the song ends.

Digital pins, and the six that can fake “half”

A **digital pin** has two states, and only two. **HIGH** is 5 V above ground, **LOW** is 0 V. There is no half-way. That is all the motor driver’s `IN1`–`IN4` pins need: each one is a yes/no question (“connect this side of the motor to the battery’s plus?”).

Speed is different. Speed needs “a bit”, “more”, “a lot”. Six of the Uno’s pins can fake that, by switching on and off very fast: **3, 5, 6, 9, 10 and 11**. They are marked with a `~` next to the number on most boards. The trick is called **PWM**, and Chapter 17 shows exactly how it works. For now, the rule: **speed wires must go on a PWM pin.** Your car’s speed wires, `ENA` and `ENB`, are on 5 and 6.

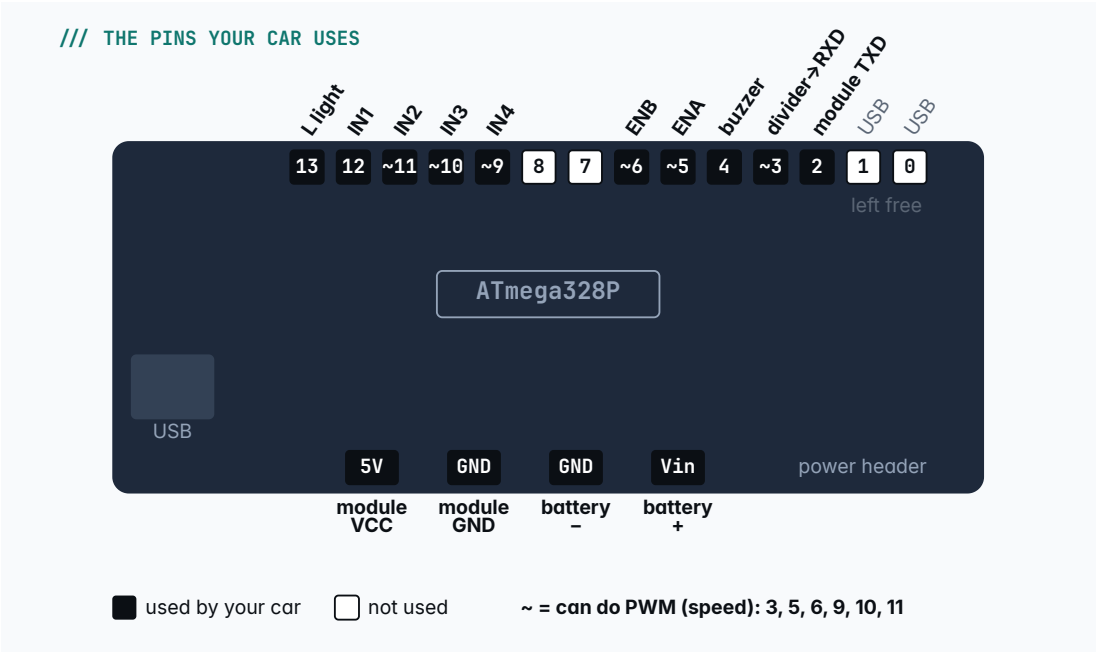


FIGURE 15.1 The Uno’s pins, with the ones your car uses highlighted. PWM pins carry a ~ mark. Pins 0 and 1 are the USB cable’s; your car leaves them free.

TABLE 15.1 Every pin your car uses

PIN	WIRE GOES TO	JOB	KIND OF PIN
0, 1	nothing	The USB cable’s serial line	serial (left free)
2	Bluetooth module `TXD`	The board **listens** to the module	digital
3	divider → module `RXD`	The board **talks** to the module	digital (PWM-capable, used as digital)
4	buzzer	Horn, beeps, the song	digital
5	motor driver `ENA`	Speed, one side	**PWM**
6	motor driver `ENB`	Speed, other side	**PWM**
9, 10, 11, 12	`IN4`, `IN3`, `IN2`, `IN1`	Direction	digital (9–11 are PWM-capable, used as digital)
13	the `L` light on the board	On means “the car’s sketch is running”	digital
`Vin`, `GND`	battery	Power, and the shared ground	power
`5V`, `GND`	Bluetooth module `VCC`, `GND`	The module’s power	power

> NOTE
> **The 2021 car was different in one place.** It had its Bluetooth module (an HC-06) on pins 0 and 1, the USB cable’s own pins. That worked, but a module there blocks every upload, so you had to unplug it each time. The v2 sketch moves the module to 2 and 3, for both the HC-06 and the HM-10, and pins 0 and 1 stay free. Chapter 32 explains the trap this avoids.

Two examples of “the right kind of pin”:

- **Direction.** `IN1` on pin 12 only ever needs HIGH or LOW. Pin 12 is not a PWM pin, and it doesn’t need to be.
- **Speed.** `ENA` on pin 5 needs “59% on”, “98% on” and so on. Pin 5 can do that. Pin 13 can’t: if you ask a non-PWM pin for a speed, it quietly gives you full on (for any value from 128 up) or full off (below 128). The car still drives, at one speed only, and nothing tells you why. That is exactly what happened to this car’s sketch from 2016 until 2026; Chapter 22 tells the story.

⏱ TEST IT

TRY THIS:	find the ~ marks on your own board. Then find the two wires from `ENA` and `ENB`.
YOU SHOULD SEE:	both speed wires sit in sockets marked `~5` and `~6`.
A INSTEAD:	your board has no ~ marks (some copies leave them off).
MEANING:	nothing is wrong; use Figure 15.1 instead. The six PWM pins are 3, 5, 6, 9, 10 and 11 on every Uno.

What it remembers, and what it forgets

The chip has two kinds of memory that matter here.

TABLE 15.2 The board keeps its instructions, not what happened last time

MEMORY	SIZE ON THIS CHIP	WHAT YOUR CAR KEEPS THERE	SURVIVES POWER-OFF?
Flash	32 KB (32,256 bytes free for sketches; the rest holds the small loader that accepts uploads)	The sketch: 7,090 bytes of it	**Yes**
SRAM (working memory)	2 KB	The last letter received, and whether the car is reversing	**No**

So **the board keeps its instructions when it is switched off, but not what happened last time.** Every time it starts, it runs the sketch from the top: it prints `Car ready`, turns the `L` light on, holds the wheels still, and sets the speed to the slowest step. Whatever you told it before is gone.

A picture for it: flash is a recipe printed in a book; SRAM is a whiteboard where the cook writes “up to step 4”. Turn the lights off and on, and the book is still there; the whiteboard has been wiped.

> ⓘ NOTE
> **Where the recipe picture breaks.** A printed book wears out only slowly. Flash can be rewritten about 10,000 times before the chip's maker stops promising it will hold (that is the number in the ATmega328P datasheet). An upload is one rewrite, so this is not a limit you will meet. And a book can be read by anyone; the flash is only read by the chip, and only while it has power.

There is a third kind of memory, **EEPROM** (1 KB), which a sketch can use to remember things like a favourite speed across power-offs. Your car's sketch doesn't use it.

Ways the board restarts

The board starts the sketch from the top in three situations. Each shows the same rule.

- **Power on.** The battery switch goes on with no USB in, or the USB goes in with the switch off.
- **Opening the Serial Monitor.** On an Uno, opening the Serial Monitor resets the board. That is why you see `Car ready` again each time you open it. (It lets the laptop restart the board ready for an upload without you pressing the button.)
- **Pressing the reset button.**

And one situation where it does **not** restart: unplugging the USB while the battery switch is on. The board simply carries on, running from the battery through `Vin`. That is the test at the end of Chapter 9: the `ON` light must stay lit.

TEST IT

TRY THIS: car on its box, USB in, Serial Monitor open, battery switch on. Send `F`. The wheels turn. Now switch the battery off, unplug the USB, wait five seconds, plug the USB back in, and switch the battery on.

YOU SHOULD SEE: after the USB goes back in, the Serial Monitor (open it again if it closed) shows `Car ready`, and when the battery switch goes on the wheels stay still.

A INSTEAD: the wheels start turning again.

> **MEANING:** they shouldn't: the `F` lived in working memory and was wiped. If the wheels turn, the board never lost power (check that the switch really was off), or a phone is still connected and sending letters.

TEST IT

TRY THIS: the same again, but leave the battery switch **on** while you unplug the USB.

YOU SHOULD SEE: the wheels keep turning and the board's `ON` light stays lit. Nothing restarted.

A INSTEAD: the wheels stop and the `ON` light goes out.

> **MEANING:** the battery isn't reaching the board's `Vin`. Switch off and redo the power step in Chapter 9. Send `S` (or switch off) to stop the wheels.



The car's sketch, the one your board is running, free to download and read.

SCAN OR TYPE **troniction.com/go/car-sketch**

CHECK YOURSELF

1. You want to move `ENA` off pin 5. Which pins could it move to, and why not pin 13?
2. What survives a power-off: the program, or the speed you chose? What does the car do with its speed when it starts again?
3. Your car is driving on the floor from the phone, with no USB cable. You press the board's reset button. Predict what the car does, step by step.

Answers at the back of the book on page 249

/// CHAPTER

16

The motor driver and the H-bridge

The motor driver is the red board with the black finned block. The fins are a **heatsink**, and under it sits the chip that does the work: an **L298N**. Everything else on the red board is there to help that chip: screw terminals for the motors and the battery, a row of pins for the six signal wires, a small 5 V regulator, eight little diodes and a red power light.

This chapter answers three questions about it. How does it make a motor go backwards? What do the four `IN` pins actually do? And why does it get warm?

Reversing a motor, by hand first

You already reversed a motor once, in Chapter 9: if a side turned backwards, you swapped its two wires in the screw terminal. A DC motor turns one way when the current goes through it one way, and the other way when the current goes through it the other way. Swap the wires, and you swap the direction of the current.

Two examples of that rule:

- **A side that turned backwards in the wheels test.** You swapped its two leads at `OUT3` and `OUT4`, and it turned the right way.
- **A torch-battery experiment.** Touch a small motor's two wires to a battery's ends: it spins one way. Turn the battery round: it spins the other way. (You don't need to do this one; it's the same rule.)

The motor driver does exactly that swap, except electrically, in a fraction of a second, with no screwdriver. The way it does it is called an **H-bridge**.

The H-bridge: four switches

Picture four switches arranged in a square, with the motor across the middle. Drawn out, it looks like the letter H: the two sides of the square are the uprights, and the motor is the crossbar.

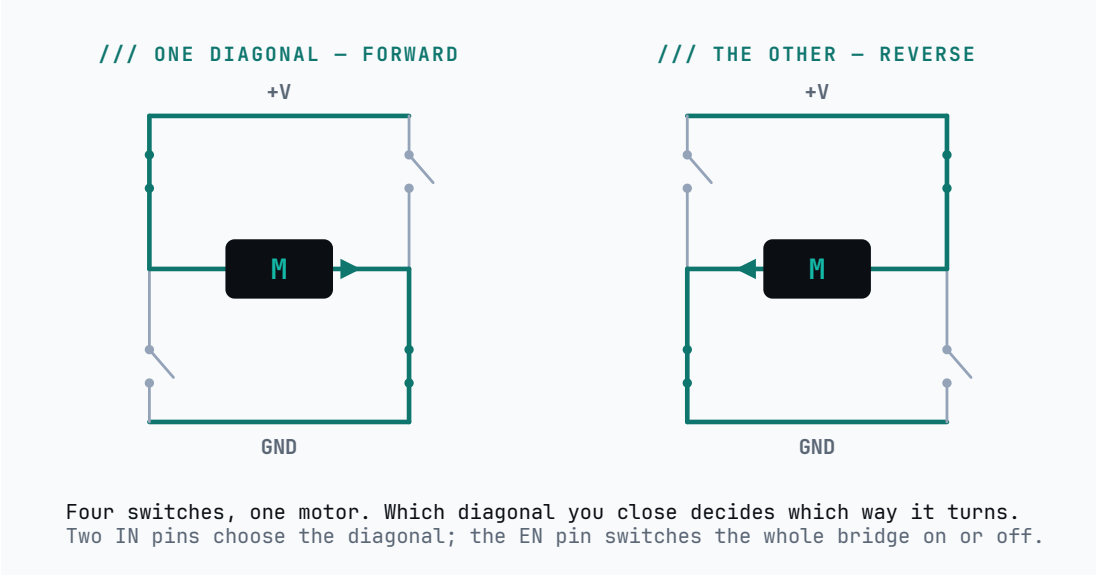


FIGURE 16.1 The same four switches, two different pairs closed. Closing one diagonal pair sends current through the motor one way; closing the other diagonal sends it the other way.

- Close the **top-left and bottom-right** switches, and current runs from the battery's plus, through the motor from left to right, down to ground. The motor turns one way.
- Close the **top-right and bottom-left** instead, and current crosses the motor from right to left. It turns the other way.

The motor itself never changes. Only the path through it does. The L298N has **two** H-bridges, one for each side of the car: bridge A drives the motors on `OUT1`/`OUT2`, bridge B the motors on `OUT3`/`OUT4`.

The switches are not little levers. They are **transistors**: parts that let current through or block it, controlled by a small signal. That is how the board can steer more than an amp with signals that each carry at most 0.1 mA into the chip (the datasheet's figure for an `IN` pin).

What the IN pins do

Here is the part v1 of this book got slightly wrong. You don't need four control wires per side. You need **three**: two `IN` pins, because the chip pairs the switches up for you, and one `EN` pin.

Each `IN` pin decides one end of the motor. `IN1` HIGH connects `OUT1` to the battery's plus; `IN1` LOW connects `OUT1` to ground. `IN2` does the same for `OUT2`. So:

- `IN1` HIGH, `IN2` LOW: `OUT1` is plus, `OUT2` is ground. Current flows one way. **Turn.**
- `IN1` LOW, `IN2` HIGH: the other way round. **Turn the other way.**
- `IN1` and `IN2` the **same**: both ends of the motor are tied to the same place. No push at all, and something more interesting, which is the next section.

The third wire for each side, `ENA` (or `ENB`), switches the whole bridge on or off. When `ENA` is off, all four switches open, whatever the `IN` pins say. So one side of the car uses three signal wires (`IN1`, `IN2`, `ENA`), and the whole car uses six.

TABLE 16.1 The L298N's truth table for one side (bridge B is the same with IN3, IN4 and ENB)

IN1	IN2	ENA	MOTOR A
HIGH	LOW	on	turns one way
LOW	HIGH	on	turns the other way
same	same	on	**brakes**: stops fast
any	any	off	**coasts**: freewheels to a stop

Which way is "forward" is not decided by the chip at all. It depends only on which way round the motor leads sit in the screw terminal, and that is why you tested and swapped in Chapter 9 rather than following a rule.

Now the car's own letters. These are the `IN` patterns your sketch sets, the same ones the 2021 car has always driven with:

TABLE 16.2 The car's letters as switch patterns

LETTER	IN1	IN2	IN3	IN4	SIDE A	SIDE B
`F`	LOW	HIGH	HIGH	LOW	one way	one way
`B`	HIGH	LOW	LOW	HIGH	reversed	reversed
`L`	HIGH	LOW	HIGH	LOW	reversed	one way
`R`	LOW	HIGH	LOW	HIGH	one way	reversed
`S`, or anything else	HIGH	HIGH	HIGH	HIGH	brake	brake

Look at `L` and `R`: one side forward, the other side backwards. That is how a car with no steering turns, and Chapter 18 explains it.

Brake or coast

Why does "both the same" stop the motor *fast*, rather than just stopping the push?

Because a spinning motor is also a small **generator**. Turn its shaft and it makes its own voltage. When both its ends are tied together through the switches, that home-made voltage drives a current round the short loop, and that current pushes *against* the spin. The motor brakes itself. When `ENA` is off instead, every switch is open, the home-made current has nowhere to go, and the motor just freewheels down.

Your car's stop is the brake: the sketch sets all four `IN` pins HIGH. The L298N datasheet calls this "fast motor stop", and "free running motor stop" for the coast.

⏱ **TEST IT**

TRY THIS:

car on its box, battery on. Send `9`, then `F`, then `S`. Watch how quickly the wheels stop. Then send `F` again, and this time switch the battery **off** while they spin.

YOU SHOULD SEE:	with `S` the wheels stop almost at once (brake). With the switch off they spin down more slowly, because the motor driver has lost its power and every switch has opened (coast).
A INSTEAD:	no clear difference.
MEANING:	the wheels have so little to keep them turning, off the ground, that both stops look quick. Watch one wheel closely, or try it at the fastest speed, `q`. On the floor, with the car's weight moving, the difference is bigger. Switch back on when you are done.

Why it gets warm

The transistors in an H-bridge are not perfect switches. When one is "closed", it still keeps a little voltage for itself. The current goes through **two** of them (one at the top, one at the bottom), so it pays twice. The L298N datasheet gives the total: **1.8 to 3.2 V lost at 1 A**, and up to 4.9 V at 2 A. The chart in the datasheet starts at 0.8 A, where the pair already costs a little over 2 V. Your motors draw less than that, so **about 2 V** is a fair working figure, not a reading.

So the motors never get the battery's full 7.4 V. They get about 7.4 − 2 = **5.4 V**, which sits inside their 3–6 V rating. (That is no accident: it is why the car uses two cells and not three.)

And the 2 V that didn't reach the motors has to go somewhere. **It turns into heat in the chip.** That is where the warmth comes from, and the energy for it comes from the battery, like everything else on the car.

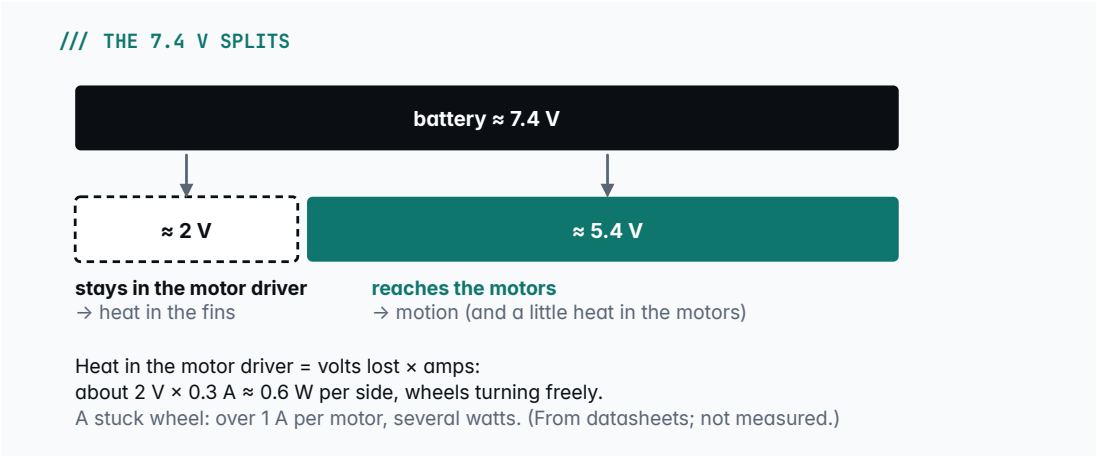


FIGURE 16.2 The battery's 7.4 V splits: about 2 V stays in the motor driver and becomes heat; about 5.4 V reaches the motors and becomes motion (and a little heat in the motors).

How much heat? Power (in watts) is volts times amps. With the wheels turning freely, each side takes roughly 0.3 A (two motors at about 150 mA each), so each bridge turns about 2 V × 0.3 A = 0.6 W into heat: about a watt for both. That is enough to make the fins feel warm, and it is what the fins are for: they spread the heat into the air.

It gets much worse when a motor can't turn. A stalled TT motor draws over an amp, and a side has two of them: well over 2 A, past the chip's 2 A rating. The chip then turns several watts into heat. The L298N has its own over-temperature protection and will shut its bridges off to save itself, but don't rely on that: **if the car is pushing against a wall or a chair leg, lift it off.**

⚠ WARNING

Don't hold a wheel still to "test" the motors. A stalled motor draws its biggest current, the motor driver gets hot, and the cells work hardest. To stop the wheels, send `S` or switch off.

⏱ TEST IT

TRY THIS:

car on its box. Send `F`, let the wheels run for a minute, then `B` for a minute. Send `S`, switch off, and touch the heatsink fins with one fingertip. Then touch the board's chip for comparison.

YOU SHOULD SEE:	the fins feel warmer than the board.
A INSTEAD:	you can't feel any difference.
MEANING:	with the wheels off the ground the motor driver works lightly, and a minute may not be enough to warm it. That fits the numbers: about a watt. The motor driver gets warmer driving on the floor, and warmest when the wheels are stuck. If the fins are ever too hot to keep a finger on, switch off and look for a wheel that can't turn. (This book has not measured how warm "normal" is on the reference car; that number will come from the next build.)

Flip the four switches yourself: the H-bridge lab shows the current's path and which way the motor turns.

SCAN OR TYPE **troniction.com/go/car-hbridge**

The L298N datasheet, explained, with the PDF and the drop chart.

SCAN OR TYPE **troniction.com/go/car-l298n-sheet**

CHECK YOURSELF

1. `IN1` and `IN2` are both HIGH. What does the motor do? Is that the same as `ENA` being off?
2. Why does the motor driver get warm, and where does that energy come from?
3. One side of your car turns the wrong way. You could swap its two motor leads in the screw terminal, or swap the `IN3` and `IN4` wires at the board. Explain why both fixes work. (Look back at Chapter 9.)

Answers at the back of the book on page 249

/// CHAPTER

17

Speed is a trick: PWM

You have already seen the trick work. In the wheels test, speed `1` made the wheels crawl and speed `9` made them race. But a digital pin has only two states, 5 V and 0 V. There is no “a bit”. So how does pin 5 make the motors go *slowly*?

It switches on and off, very fast, and changes **how long it stays on** each time.

On, off, on, off

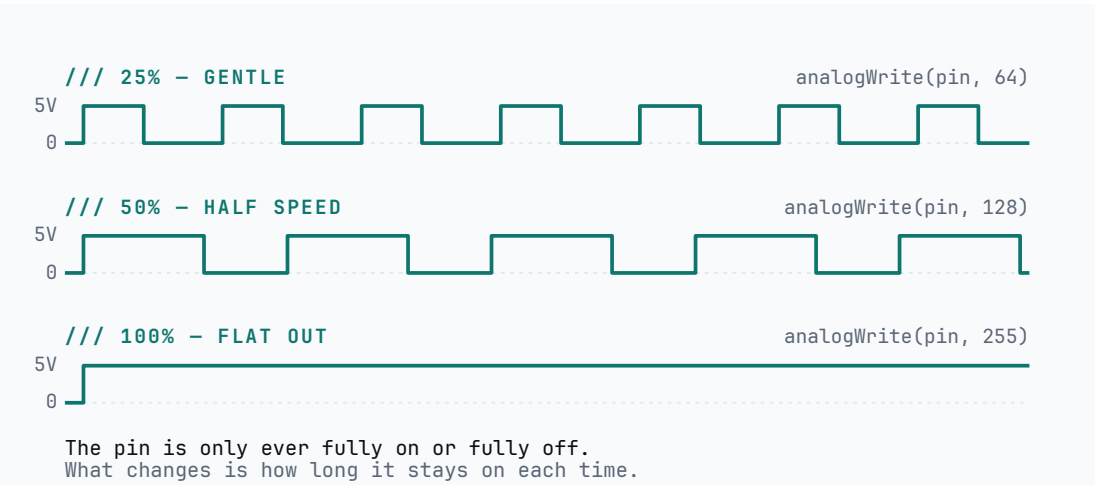


FIGURE 17.1 The same pin, three settings. At 25% it is on for a quarter of each cycle; at 50%, half; at 100%, all the time. The pin is only ever fully on or fully off.

The trick is called **PWM**, short for *pulse-width modulation*: the width of each pulse is what changes. The fraction of the time the pin is on is called the **duty**

cycle.

- On for a quarter of each cycle (25% duty): the motor behaves as if it got about a quarter of the push.
- On all the time (100%): full push.

On pins 5 and 6 of an Uno, each on-off cycle takes about one thousandth of a second. The pin repeats it about **980 times a second**. That is far too fast for a motor to follow each pulse, so what the motor feels is the *average*.

In the sketch, the speed is a number from 0 to 255. `analogWrite(5, 255)` means 100% on; `analogWrite(5, 128)` means about 50% on; `analogWrite(5, 0)` means off.

Why the motor averages

Two reasons, and you can feel both of them.

The car is heavy and the wheels are spinning. Something moving keeps moving for a moment after you stop pushing it. In one thousandth of a second the motor barely slows down before the next push arrives.

The motor's coil resists sudden changes in current. The coil is a long wire wound round and round, and a coil like that smooths out on-off pulses by itself, a little like a heavy door on a slow closer.

Two everyday examples of the same averaging:

- **A playground roundabout.** Push it for part of every turn and it settles at a steady speed. Push for a bigger part of every turn and it settles faster. Nobody pushes all the way round.
- **A dimmed light that flickers too fast to see.** Many LED lights are dimmed exactly this way. Your eye averages the flicker into "dim", as the motor averages the pulses into "slow".

> ⓘ NOTE
> **Where the roundabout picture breaks.** You push a roundabout a few times a minute and you can feel every push. Push a motor that rarely and it would jerk: on, stop, on, stop. PWM works only because the pulses are much faster than the motor can respond. They are not faster than your ears, though. About a thousand pulses a second is a note you can hear, so you may hear a faint whine from the motors at low speeds. That whine is the PWM itself.

The speed ladder

Your car doesn't use the whole 0–255 range. When the phone sends a digit, the sketch picks one step from a ladder:

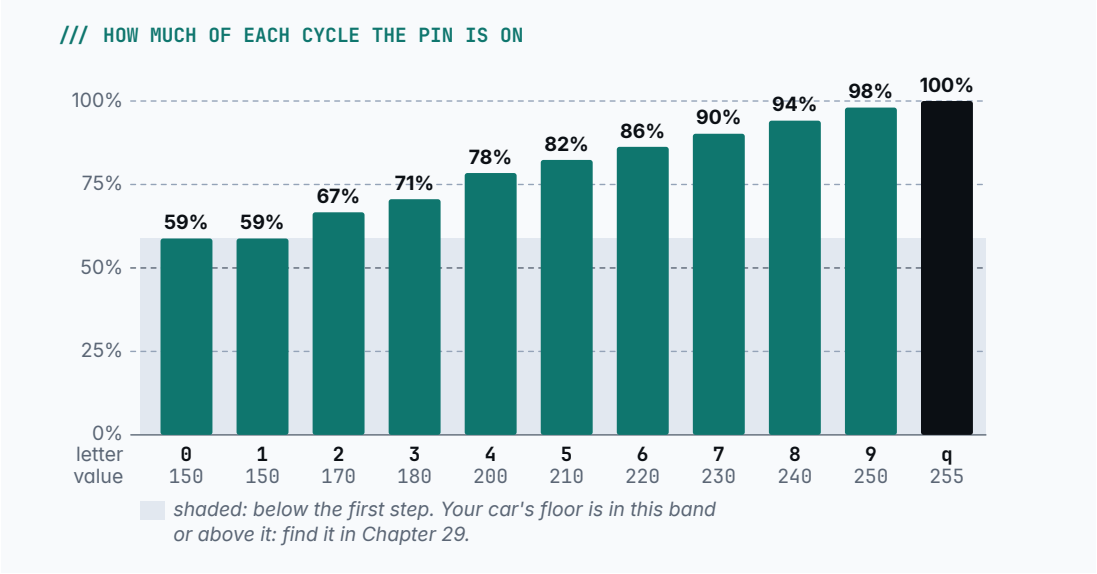


FIGURE 17.2 The ladder as bars. Every step is above half on. Below the first step is a shaded band: your car's floor is in this band or above it, and this book has not measured it.

TABLE 17.1 The car's speed ladder, from the sketch, with each step as a duty cycle

LETTER	0	1	2	3	4	5	6	7	8	9	Q
Value	150	150	170	180	200	210	220	230	240	250	255
Duty	59%	59%	67%	71%	78%	82%	86%	90%	94%	98%	100%

Three things stand out.

The ladder starts at 150, not 0. The 2021 author started it there because below about 150 the motors buzz but the car doesn't move. A motor has to push hard enough to get the gears and the car's weight going, and below some duty cycle it simply can't: it hums, warms up and stays still. Starting the ladder at that point means every step on the phone does something. That 150 is the author's choice for the 2021 car, not a measured number, and it will differ from car to car. Finding *your* car's real floor is a job for Chapter 29, where you change the ladder.

`0` **doesn't stop the car.** It gives the same speed as `1`. To stop, the car uses `S` (or any letter it doesn't know). Look at it this way: a digit only ever changes the *speed*; only a letter changes what the wheels are doing.

The ladder is uneven. The jump from `1` to `2` is 20 steps; the rest are 10 or so. There is no rule behind that: it is simply what the 2021 author chose. In Chapter 29 you can choose your own.

A rough calculation, to see why 150 is a sensible floor. The motors get about 5.4 V when they're fully on (Chapter 16). At 59% duty the average is about 0.59 × 5.4 ≈ 3.2 V. The motors are rated from 3 V up. So the bottom step sits just above the bottom of the motor's rating. That is a calculation, not a measurement, but it agrees with the 2021 author's 150.

TALK ABOUT IT

A pin can only be on or off. So how can it make a speed in between?

PARENT: LISTEN FOR:

"it switches on and off very fast", "it's on for longer" or "it pushes more of the time". Any answer about on and off is on the right track. "It gives more volts" is the common first guess: the pin always gives 5 V when it's on.

When PWM isn't there

Only six pins can do PWM. Ask a non-PWM pin for a speed and the Arduino software does the closest thing it can: for any value from 128 up it turns the pin fully on, and below 128 fully off. Every step on your ladder is 150 or more, so on a non-PWM pin every speed button would give full speed. The car would drive perfectly, at one speed, with no error anywhere.

That is not a made-up example. It is what this car's sketch did from 2016 until 2026, when its speed wires were found on pins 13 and 8. Chapter 22 walks through how it was found.

TEST IT

TRY THIS: car on the floor, driven from the phone (USB out). Send `1`, then `F`. Then try `1` again with the car on a rug, and then with its front wheels on a thick book, so it has to climb.

YOU SHOULD SEE: on a smooth floor, speed `1` moves the car slowly. On the rug or the climb it may crawl, hum and stall.

A INSTEAD: speed `1` doesn't move the car even on the smooth floor.

MEANING: the floor is where the push just beats the gears, the weight and the surface, so it changes with every surface. If `1` won't move your car on a smooth floor, try `2` and `3`: the first number that moves it is your car's floor today. Fresh cells move it lower down the ladder than tired ones. Write the number down; you'll use it in Chapter 29.

⚠ WARNING

Don't leave a stalled car humming. If the car hums but doesn't move, send `S` or a higher speed. A stalled motor draws its biggest current and warms the motor driver (Chapter 16).

Slide the duty cycle and watch the square wave and the average voltage change, with a marker for where a motor stalls.

SCAN OR TYPE **troniction.com/go/car-pwm**

CHECK YOURSELF

1. Chapter 15 said a non-PWM pin rounds every value of 128 or more to full on. Using the ladder, what would speed `0` do if `ENA` were on pin 13, and why?
2. Why not start the ladder at 0? What would the bottom few buttons do?
3. Your friend's car moves the same at every speed. Name two different faults that cause exactly this, one in the wiring and one on the motor driver. (Chapters 15 and 9 have one each.)

Answers at the back of the book on page 249

/// CHAPTER

18

Motors, gears and skid steering

Turn your car over. Under each corner is a yellow plastic box with a wheel on one side and two wires coming out of it. That box is a **TT gear motor**: a small motor and a gearbox in one case. Your car has four of them.

What's inside the motor

Inside the yellow case, at one end, is a small metal can: the **DC motor**. It has magnets round the outside and a coil of wire on the spinning part. When current flows through the coil, the coil is pushed by the magnets, and it turns. Reverse the current and the push reverses, which is exactly what the H-bridge does (Chapter 16).

On its own, that motor spins very fast and very weakly. Try to stop a bare motor's shaft with your fingers and it gives up at once. A car carrying a battery needs the opposite: slow and strong. That is the gearbox's job.

Gears: trading speed for strength

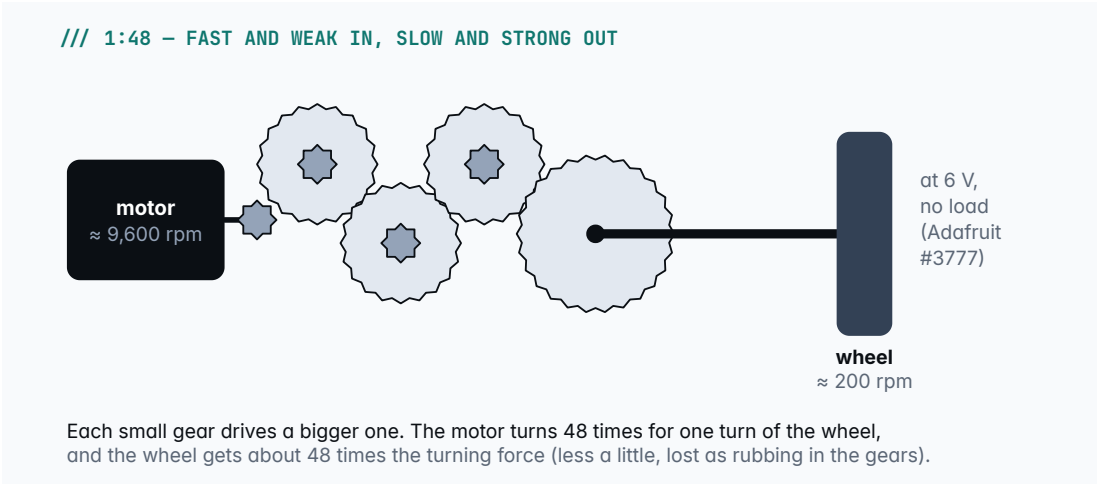


FIGURE 18.1 Inside the yellow case: a small fast motor drives a chain of plastic gears. Each small gear turns a bigger one, so the wheel shaft turns once for every 48 turns of the motor.

Your motors are **1:48**: the motor turns 48 times for every single turn of the wheel. In return, the wheel gets (roughly) 48 times the turning force, which engineers call **torque**.

Two examples you already know:

- **A bicycle's lowest gear.** Pedals go round fast, the wheel turns slowly, and you can climb a hill you couldn't climb in top gear.
- **A screwdriver with a fat handle.** Your hand goes round a big circle while the tip turns a small one, and the screw goes in with less effort.

The numbers, from the Adafruit page for this exact motor:

TABLE 18.1 The TT gear motor's ratings (Adafruit #3777)

	AT 3 V	AT 6 V
Wheel speed, no load	about 90 turns a minute	about 200 turns a minute
Turning force when stalled	0.4 kg·cm	0.8 kg·cm
Current when stalled	1.1 A	1.5 A
Current turning freely	150 mA	

And a calculation, to see what the gearbox buys. At 6 V the wheel turns about 200 times a minute, so the motor itself turns about 200 × 48 = **9,600 times a minute**. Without the gearbox, the wheel would turn that fast, with only 1/48 of the turning force: about 0.017 kg·cm, far too little to move the car at all.

> ⓘ NOTE
> **Where the bicycle picture breaks.** A bicycle lets you change gear. This gearbox is fixed at 1:48, forever. To change the car's speed you change the motor's push with PWM (Chapter 17), not the gear. And gears aren't free: plastic teeth rub, so some of the motor's work is lost as heat and noise before it reaches the wheel. They are also the weakest part of the car: push a wheel on by its hub, never by levering against the yellow case.

How fast should your car go?

One more calculation. Your wheels are 65 mm across, so one turn of a wheel moves the car about π × 65 mm ≈ 204 mm. At 200 turns a minute that is about 41 metres a minute, or **0.7 metres a second**.

That is the most it could be, and your car will be slower: the motors get about 5.4 V, not 6 V (Chapter 16), and "no load" means a wheel spinning in the air, not carrying a car.

⏱ TEST IT

TRY THIS:	mark a start line and a finish line 2 metres apart on a smooth floor. Drive the car over them at speed `9` (start a metre before the line so it's already up to speed). Time it with a phone's stopwatch, three times, and take the middle time. Speed = 2 ÷ time.
YOU SHOULD SEE:	a speed below 0.7 metres a second.
A INSTEAD:	the car curves off the line before the finish.
MEANING:	the two sides aren't quite equal. Motors from the same box still differ a little, and one side's pair is slightly faster. That's normal; steer it straight with short taps, or measure over a shorter distance.

Turning without steering

Your car has no steering wheel and no wheels that point. It turns the way a tank does: by driving its two sides differently. This is called **skid steering**.

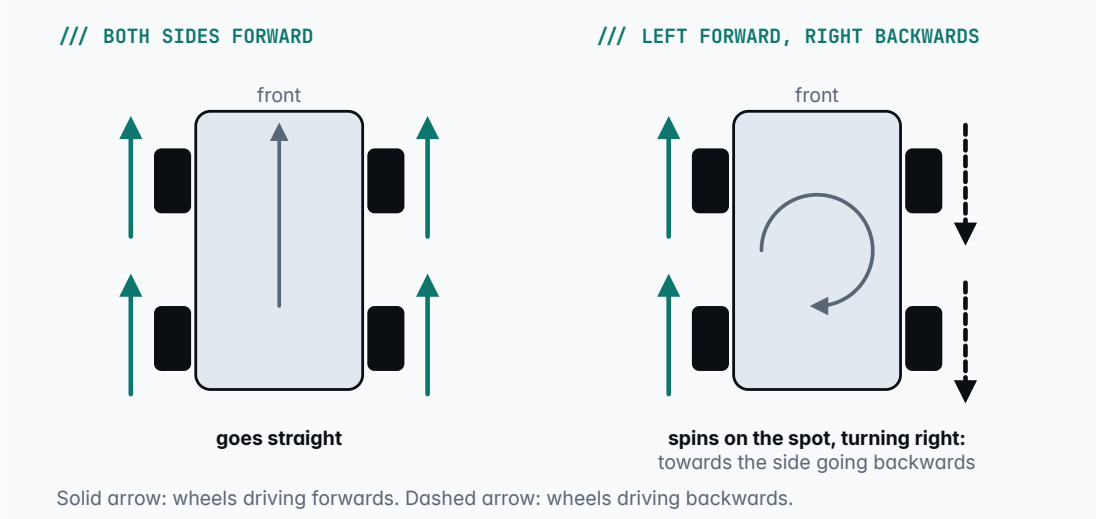


FIGURE 18.2 Seen from above. Both sides forward: the car goes straight. One side forward and the other backwards: the car spins on the spot, turning towards the side that goes backwards.

- **Both sides forward**: the car goes straight.
- **One side forward, the other backwards**: the car spins on the spot. It turns *towards* the side that is going backwards.

Look back at the letters in Chapter 16: `L` and `R` each run one side forward and the other side backwards. That is why your car turns on the spot rather than in a curve.

Two examples of the same idea:

- **A rowing boat** turns on the spot if one oar pulls forward and the other pushes back.
- **A digger or a tank** on tracks turns by running one track forwards and the other backwards.

This is also why both wheels on one side share one channel of the motor driver: they always do the same thing anyway. Two motors share `OUT1`/`OUT2`, the other two share `OUT3`/`OUT4`.

NOTE

Where the rowing boat breaks. A boat slides round easily on water. A car's tyres grip the floor, and to turn on the spot they have to be dragged sideways while they roll. That dragging takes extra push, wears the tyres, and is the "skid" in skid steering. On a grippy surface the turn needs more push than on a slippery one.

TALK ABOUT IT

Left side forward, right side backwards: which way will the car turn? Point before you test it.

PARENT: LISTEN FOR:

"right", towards the side going backwards. If the guess is "left", ask them to stand up and try it with their own feet: left foot steps forward, right foot steps back.

TEST IT

TRY THIS:	at speed `5`, send `L` for one second on a smooth floor, then on a rug. Count how far round the car gets each time.
YOU SHOULD SEE:	the car turns on both, by a different amount on each surface.
A INSTEAD:	on the rug the car hums and hardly turns.
MEANING:	the rug grips the tyres harder, so dragging them sideways takes more push than the motors have at that speed. Try a higher speed. Which surface turns better on *your* car is your answer to write down; this book hasn't measured it.

The chassis, wheels and gear motors: the ratings, and how to get all four wheels turning the same way.

SCAN OR TYPE troniction.com/go/car-motors

CHECK YOURSELF

1. A friend built a car from a different guide. `F` and `B` work, but `R` turns it left and `L` turns it right. Using Chapter 16's letter table, what is different about how it was built?
2. What would happen to the car if the motors had no gearbox and drove the wheels directly?
3. Speed `1` moves your car on the kitchen floor but not on the rug. Use Chapters 17 and 18 together to explain why.

Answers at the back of the book on page 249

/// CHAPTER

19

Power: cells, Vin and brown-outs

Your car has one battery: two **18650** lithium-ion cells in a holder with a switch. That one battery feeds everything. This chapter follows the power from the cells to the wheels and to the board, and explains the strangest thing a car like this does: **it restarts itself when it starts moving**, long before the battery is flat.

> ⚠ WARNING
>
> **The cells are the parent's job.** Taking cells in and out, charging them, and any measurement across the battery's leads is done by the adult. Never let anything metal touch both ends of a cell. The safety page in Chapter 5 has the full rules.

What "7.4 V" means

The number "18650" is just the cell's size: 18 mm across, 65 mm long. Each cell's voltage depends on how full it is. One cell maker's datasheet (Samsung's INR18650-25R, a common cell) gives these numbers; most 18650s are close:

TABLE 19.1 What a lithium-ion cell's voltage tells you

ONE CELL	TWO IN SERIES (YOUR BATTERY)	WHAT IT MEANS
4.2 V	8.4 V	Full, straight off the charger
3.6–3.7 V	7.2–7.4 V	The "nominal" voltage printed on the label: roughly the middle of a drive
3.0 V	6.0 V	Nearly empty
2.5 V	5.0 V	The maker's cut-off: below this you are damaging the cell

So "7.4 V" is a label, not a promise. A fresh battery gives more; a tired battery gives less. Your battery is two cells **in series**: one after the other, so their pushes add up (3.7 + 3.7 = 7.4).

The power path

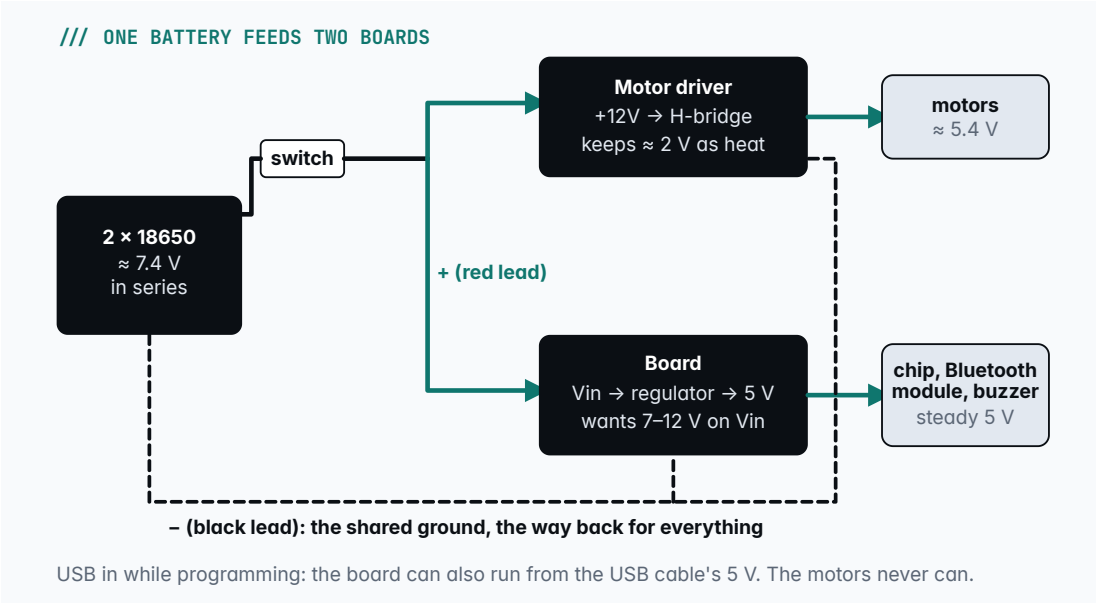


FIGURE 19.1 From the cells, through the switch, the power splits two ways. One way goes into the motor driver's +12V terminal and on to the motors. The other goes into the board's Vin pin, through the board's regulator, which makes a steady 5 V for the chip and the Bluetooth module. Every path comes back through the shared ground.

Follow the red lead in Figure 19.1:

1. **To the motor driver's `+12V` terminal.** Despite the label, it doesn't need 12 V: "12V" is just the most common supply people use, and the red board works on your 7.4 V. From there, through the H-bridge (minus its 2 V), to the motors.
2. **To the board's `Vin` pin.** The board has a **regulator** on it: a part that takes a higher, wobbly voltage and makes a steady 5 V out of it. The chip and the Bluetooth module run on that 5 V.

The board's own documents give its `Vin` limits as 6–20 V, and recommend **7–12 V**. The regulator needs some headroom above 5 V to do its job, and below about 7 V in, the 5 V it makes can start to sag.

Put that next to the cells' table and you see the problem. A full battery (8.4 V) is comfortably above 7 V. A battery at its "7.4 V" label is only just above it. A battery at 3.5 V per cell, with plenty of drive still left in it for the motors, is already at 7.0 V: the bottom of what the board wants. **This car lives at the bottom of the board's range.** It works, and has for years, but it means the board is the first thing to complain.

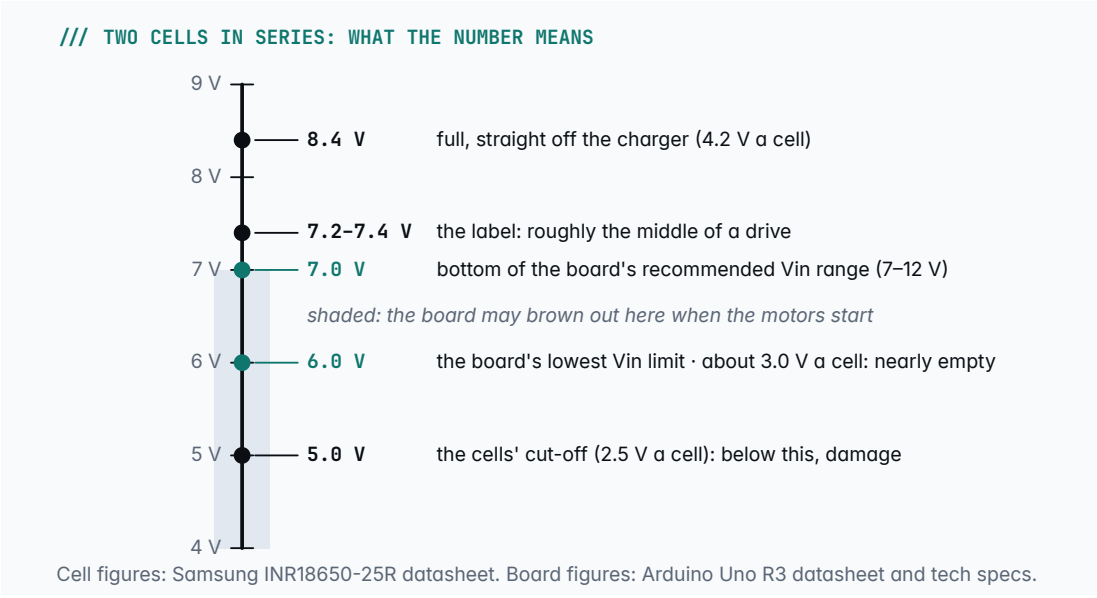


FIGURE 19.2 The battery's voltage as a ladder, with the board's lines marked: 8.4 V full; 7.4 V label; 7 V, the bottom of the board's recommended range; 6 V, the board's lowest limit; 5 V, the cells' cut-off.

Why the board restarts when the car moves

Here is the moment it happens. The car sits still. You press forward. Four motors start at once, and a motor starting from still draws far more current than one already turning: for a moment it can draw close to its stall current, which is over an amp.

Every battery has a little resistance inside it, and so do the holder's springs and the wires. When a big current flows through that resistance, some voltage is lost on the way out (current × resistance again, from Chapter 14). So **the battery's voltage dips while the motors start.** For a fraction of a second, a battery that read 7.2 V sitting still might give the board much less.

If that dip takes `Vin` low enough, the regulator can't hold 5 V, the 5 V line sags, and the chip stops working properly. The ATmega328P has a guard for exactly this, called the **brown-out detector**. On an Uno it is set to hold the chip in reset if its 5 V supply falls to about 2.7 V. Somewhere in that dip, the board restarts: the sketch starts again from the top, and the car stops.

That is a **brown-out**: the supply sagging too low without going away. No smoke, no heat, no error message. The name comes from old light bulbs that dimmed to a brownish glow when the mains sagged.

The tell is the pattern: **it works while it's standing still, and restarts when it starts to move.** And it gets worse as the cells run down, because the dip starts from a lower voltage.

⊗ MYTH

> "The motors use up the current, so there's less left over for the board."

WRONG **That is wrong: current isn't used up.**

Current goes round a loop and comes back to the battery; what goes into a motor comes out of it (Chapter 14's crowd at the door). The battery's current splits between the motor driver and the board, and each branch takes what it needs. The real problem is the **voltage**: the battery's push drops for everyone at once when the motors pull hard, and the board, which needs the most headroom, notices first.

MYTH

> "A battery gives a fixed amount of power until it's flat, so if the car is misbehaving, the battery must be dead."

WRONG That is wrong: a battery's voltage falls gradually, and how much it dips depends on what you ask of it.

A battery that restarts the board when the motors start can still read 7 V or more when you measure it at rest, and can still drive the motors. It isn't dead; it's tired, and at this car's voltage "tired" is enough to upset the board. Charge it, and the "fault" goes away. The site's advice is worth keeping: **if the car starts behaving strangely, charge it before you debug it.**

Why two cells, and not something else

The 7.4 V battery is a compromise between the motors and the board.

- **Three cells** would give 11.1 V. The board would be happy, but the motors would get about 9 V after the motor driver's 2 V, well over their 6 V rating.
- **Four AA batteries** give about 6 V when fresh (1.5 V each) and less as they drain. That is already below the board's recommended 7 V, the motors would get about 4 V, and ordinary AA cells sag badly at motor currents.
- **Two 18650s** give about 5.4 V to the motors (inside 3–6 V) and 7–8.4 V to the board (inside its range, at the bottom). Nothing else on the list does both.

TEST IT

TRY THIS: charge the cells (parent), then drive the car in circles on the floor until it stops by itself or starts restarting when you press forward. Note the time, and **switch off straight away**. Then let the parent measure the battery across the holder's leads with a multimeter set to volts DC, if you have one.

YOU SHOULD SEE:	after some time of driving, the car starts to stop on its own when you press forward, or the Bluetooth module's light starts blinking again (it lost power and is waiting for the phone). The battery, measured at rest, still reads around 7 V or more.
A INSTEAD:	the car simply gets slower and slower, without restarting.
MEANING:	your cells hold their voltage well under load, so the board never dips far enough. Both results tell the same story: the battery's voltage falls during a drive, and the board is the first to feel it. Write down the time; it is your car's real "battery life".
FIX:	switch off as soon as the car starts restarting or crawling, and charge before the next session. A holder left switched on keeps draining the cells even when the car is still, and lithium cells are damaged by running flat.

ⓘ NOTE

Why a meter, with the switch off? At rest the battery has recovered from the dip: the voltage was never "used up". And a sketch can only measure the board's own 5 V, not the battery.



What a brown-out actually is, and why it looks exactly like a dead part.

SCAN OR TYPE **troniction.com/go/car-brownout**



18650 cells: real capacities, and the safety rules that are not optional.

SCAN OR TYPE **troniction.com/go/car-batteries**

CHECK YOURSELF

1. Why would four AA cells not work well in this car? Give two reasons.
2. The car resets when it starts moving. Explain it in one sentence.
3. The car stops by itself in the middle of the kitchen. The phone still says it's connected and the Serial Monitor shows the letters arriving. Which box from Chapter 3 do you check first, and what is the first thing you do? (Use Chapter 12 too.)

Answers at the back of the book on page 250

/// CHAPTER

20

The Bluetooth module: how a letter reaches the board and back

Press forward on the phone, and a moment later the wheels turn. In between, a single letter, `F`, has made a journey through four boxes: the phone, the Bluetooth module, the board, and the wheels. This chapter follows that letter, and then follows the answer the board sends back.

The letter's journey

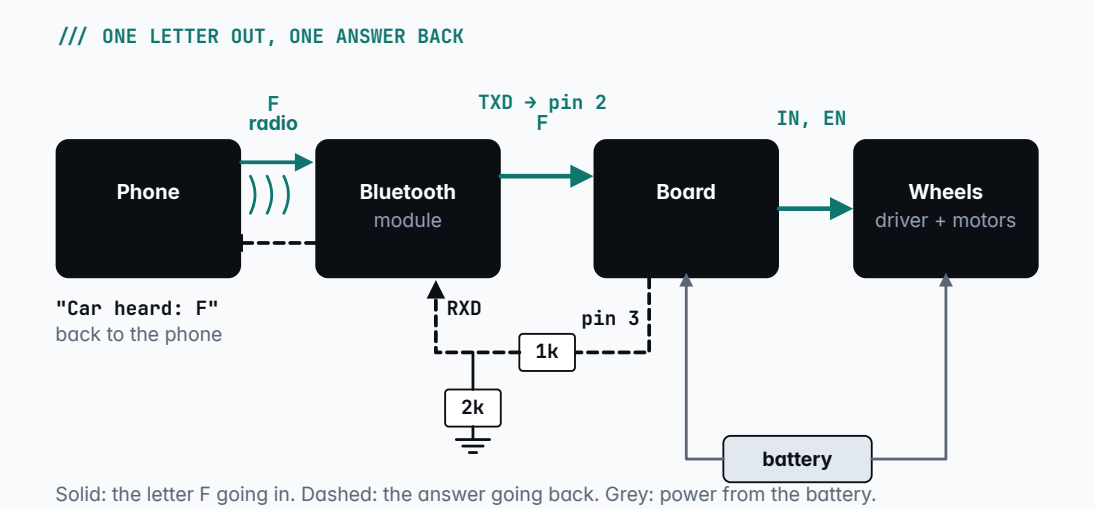


FIGURE 20.1 The letter F leaves the phone as a radio message, arrives at the Bluetooth module, travels down one wire into the board's pin 2, and the board's answer, "Car heard: F", goes back out of pin 3, through the two resistors, into the module, and over the radio to the phone.

1. **Phone to module: radio.** The phone sends `F` over Bluetooth, a radio link at about 2.4 GHz (2.4 billion waves a second), the same band as many home Wi-Fi networks.
2. **Module to board: one wire.** The module doesn't understand `F`. It just passes it on, down its `TXD` wire into the board's pin 2, as if someone had typed it.
3. **The board decides.** The sketch reads `F`, sets the four `IN` pins to the forward pattern, and the wheels turn.
4. **The answer goes back.** The sketch sends `Car heard: F` out of pin 3, through the two resistors, into the module's `RXD`, and the module sends it over the radio to the phone.

Two lines of the Serial Monitor tell you how far the letter got:

- `Got F from phone` means the letter made it through steps 1 and 2.
- `Car heard: F` on the phone's remote means the answer made it all the way back, through the resistors (if your remote or app shows what the car sends back; the book's web remote is designed to).

What "connected" proves

When the remote says *connected*, it has proved exactly one thing: **the phone can hear the module.** That's step 1. It says nothing about the wire to the board, or the board, or the wheels.

TABLE 20.1 What each sign proves, in the order the letter travels

WHAT YOU SEE	WHAT IT PROVES WORKS
The remote says *connected*	Phone and module (the radio)
`Got F from phone` in the Serial Monitor	...and the module's `TXD` wire into pin 2
The wheels turn	...and the board, the motor driver and the motors
`Car heard: F` on the phone	...and pin 3, the two resistors, and the module's `RXD`

That is why the Stage D test in Chapter 11 is a round trip and not just "it connected".

⏱ TEST IT	
TRY THIS:	car on its box, USB in, Serial Monitor open, battery off. Connect the phone. One of you presses forward about once a second while walking slowly away; the other watches the screen.
YOU SHOULD SEE:	`Got F from phone` lines keep arriving, then stop at some distance, or get patchy first.
A INSTEAD:	the lines stop almost at once, a step or two away.
MEANING:	bodies, walls and metal weaken a 2.4 GHz signal.
FIX:	try again with a clear line to the car. Where the lines stop is your car's real range indoors; write it down.

How a letter travels down one wire

Inside the board, `F` is just a number: 70. (Every letter has a number; that list is called ASCII.) In binary, 70 is `01000110`: eight ones and zeros, called **bits**.

The module sends those eight bits down its `TXD` wire **one after another**, as HIGH and LOW voltages, at an agreed speed. That speed is the **baud rate**: on your car, **9600** bits a second, so each bit lasts about 104 millionths of a second. This way of sending is called a **UART** (a *universal asynchronous receiver-transmitter*), or just **serial**.

But how does the board know when a letter starts? There is no clock wire telling it. The answer is in the frame:

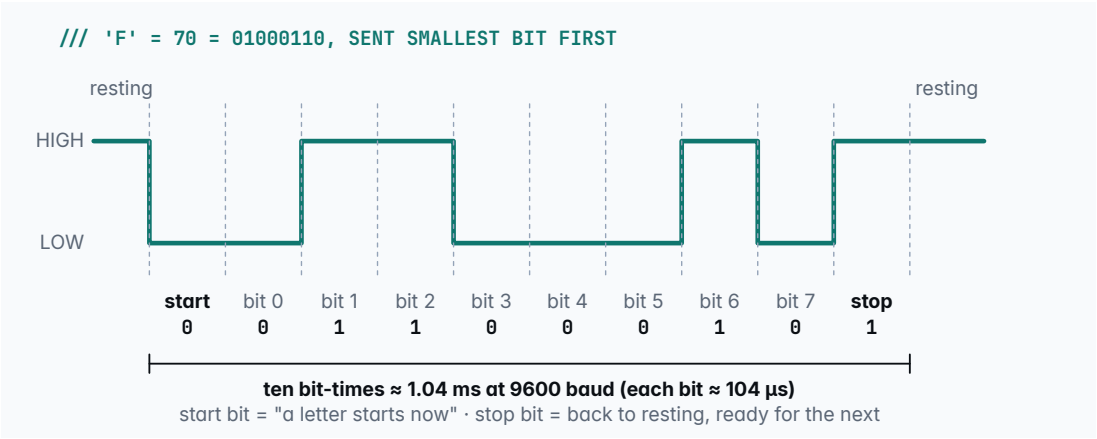


FIGURE 20.2 The letter F on the wire. The line rests HIGH. A start bit drops it LOW for one bit-time, which says "a letter is starting". Then come the eight bits of F, smallest first. Then a stop bit puts the line back HIGH.

- **The line rests HIGH.** When nothing is being sent, the wire sits at HIGH.
- **The start bit** pulls it LOW for exactly one bit-time. That fall is the signal: "a letter starts now". The board starts its own timer.
- **Eight data bits** follow, the smallest one first.
- **The stop bit** puts the line back HIGH for one bit-time, so the next start bit always has an edge to fall from.

Ten bits for one letter. This frame has a name, **8N1**: eight data bits, no parity bit, one stop bit. Your modules use it out of the box at 9600 baud. At ten bits a letter, 9600 baud carries at most 960 letters a second, and one letter takes about a thousandth of a second. Because the board times every bit from that one falling edge, both ends must agree on 9600; when they don't, the Serial Monitor fills with "garbage characters" (Chapter 24).

Why the wires cross

`TX` means "this pin talks". `RX` means "this pin listens". They are named from the point of view of **whoever owns the pin**. So one side's talking pin must go to the other side's listening pin:

- module `TXD` (talks) → board pin 2 (the board listens there);
- board pin 3 (the board talks there) → module `RXD` (listens).

Two examples of the same rule:

- **A phone call.** Your mouth goes to their ear, and their mouth goes to your ear. Mouth to mouth would be two people talking and nobody listening.
- **Your car.** The module's mouth (`TXD`) goes to pin 2, the board's ear. The board's mouth (pin 3) goes, through the resistors, to the module's ear (`RXD`).

> ⚠ WARNING
>
> **Never swap these two wires to "see what happens".** Wired talk-to-talk, the link is silent, for ever, with no error message. On this car it can also do harm: if the swap is at the board, pin 3's 5 V meets the module's 3.3 V `TXD` with no resistor between them. If you ever find the wires swapped, switch off (USB out, battery switch off) and swap them back before you test anything else.

> ⓘ NOTE
>
> **Where the phone-call picture breaks.** A call carries both directions on one line. Serial uses a wire for each direction, plus a **third**: the shared ground from Chapter 14. Because each direction has its own wire, both ends can talk at the same moment; nobody has to wait their turn.

Why the divider

The module's chip runs on 3.3 V, but the board's pins work at 5 V. So the two directions are different.

- **Module `TXD` → board pin 2: no resistors.** The board reads anything above 3.0 V as HIGH (that's 0.6 × 5 V, from the ATmega328P datasheet). The module's HIGH is 3.3 V. 3.3 clears 3.0.
- **Board pin 3 → module `RXD`: the divider.** The board's HIGH is a full 5 V, and the module's `RXD` goes straight to a 3.3 V chip. 5 V on every bit is past what that pin is built for.

The **divider** is two resistors that share the voltage between them. 1,000 Ω from pin 3 to the middle, 2,000 Ω from the middle to ground, and the module's `RXD` connects to the middle. The middle sits at two-thirds of the way up: 5 × 2,000 ÷ 3,000 = **3.33 V**.

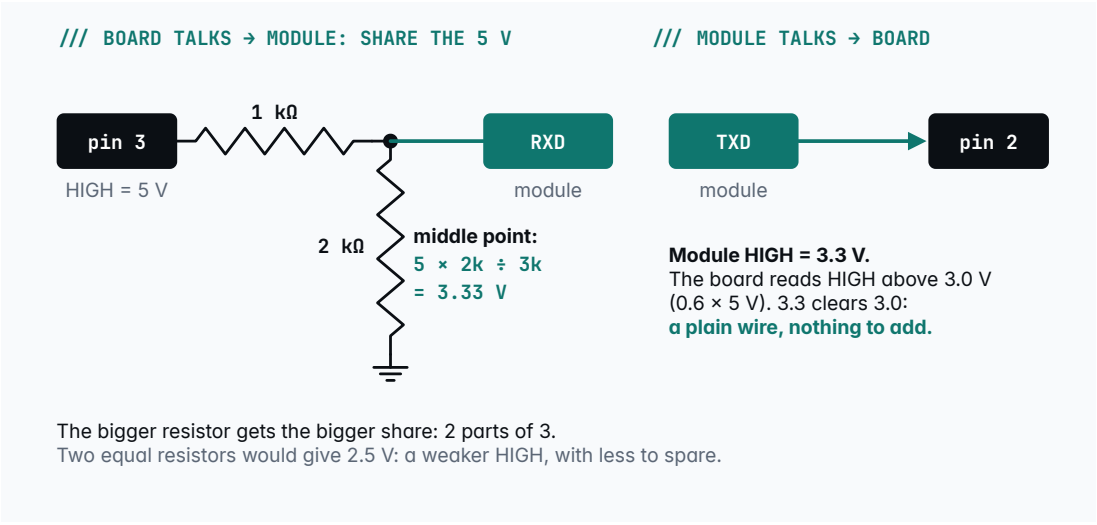


FIGURE 20.3 The divider. Pin 3's 5 V goes through 1 kΩ to the middle point, and 2 kΩ from the middle to ground. The module's RXD takes the middle: 3.3 V. The other wire, TXD to pin 2, needs nothing.

So why does the module's `VCC` go straight to 5 V, with no divider? Because `VCC` doesn't go to the chip. It goes to a small regulator on the module's own board, which makes the chip's 3.3 V from it. The power pin has a regulator; the data pin does not.

⊗ MYTH

"The module is a 3.3 V part, so its `VCC` goes on the board's 3.3V pin."

WRONG That is wrong: `VCC` goes on 5V.

The module's board carries its own 3.3 V regulator, built to take 5 V in. The Uno's 3.3V pin can give only 50 mA in total, and a Bluetooth module can draw nearly that while it's waiting for a phone to connect. On the 3.3V pin the supply sags at exactly the moment it matters, and the module appears in the phone's list and then vanishes.

Two modules, one car

This book supports two modules. On the board's side, they are the same four wires on the same pins, and the sketch can't tell them apart. The differences are all on the phone's side.

TABLE 20.2 The two modules side by side

	HC-06	HM-10
Kind of Bluetooth	Classic (version 2.0)	BLE, "Bluetooth Low Energy" (version 4.0)
Phones	Android only	Android (Chrome) and iPhone (the free Bluefy browser)
How the phone finds it	Pair it in the phone's Bluetooth settings first, then open the driving app	**Not** in Bluetooth settings: the remote page finds it
How it was verified	Drove the 2021 car for years, on pins 0 and 1. On pins 2 and 3: tested on a computer and from datasheets (S + D)	Tested on a computer and from datasheets (S + D). Not yet driven in this car

Why no HC-06 on an iPhone? iPhones don't let apps use Classic Bluetooth serial from uncertified devices like the HC-06; BLE is open to apps.

Build the frame for any letter and watch its ten bits on the wire.

SCAN OR TYPE **troniction.com/go/car-8n1**

Everything about the link: *Make It Connect* covers choosing a module, the AT commands (such as setting an HM-10 back to 9600), and every way the link fails.

SCAN OR TYPE **troniction.com/go/car-mic**

CHECK YOURSELF

1. On a friend's car, the module's `TXD` and `RXD` wires have been swapped at the board. Predict the symptom on the remote and in the Serial Monitor, and say what to do first.
2. Why does the module need 3.3 V on its `RXD` but not on its `VCC`?
3. The phone says *connected*, the Serial Monitor shows `Got F from phone`, but the wheels don't turn. Which boxes are proven, and which is left? (Chapter 16 has a cause that fits.)

Answers at the back of the book on page 250

3

/// PART 3

Find the fault

Every car breaks. A wire works loose, a jumper cap goes back on, a cell runs tired, or someone “tidies” the sketch. This part teaches the one skill that outlasts this car: finding a fault the way an expert does, by testing instead of guessing.

The method is the loop from Chapter 3 and the four boxes you proved one at a time in Part 1. The part teaches it in five chapters, and the help fades as you go. Chapter 22 talks through every thought on a real bug. Chapter 23 turns the thoughts into a form you fill in. Chapter 24 gives you an instrument for looking inside the board. Chapter 25 gives you ten broken cars with only the symptom, and you do the rest.

Read it as a pair if you like: one of you reads the symptom, the other plays the expert.

/// CHAPTER

21

Four boxes and half-splitting

Start from your own car

Your car is four boxes in a row and a power line. A letter starts in **the phone**, travels by radio to **the Bluetooth module**, goes down one wire into **the board**, and the board turns it into signals for **the wheels**: the motor driver and the motors. The battery is not on that road at all. It is a separate line that feeds the motor driver and the board, and through the board the Bluetooth module. Figure 21.1 shows the whole car this way.

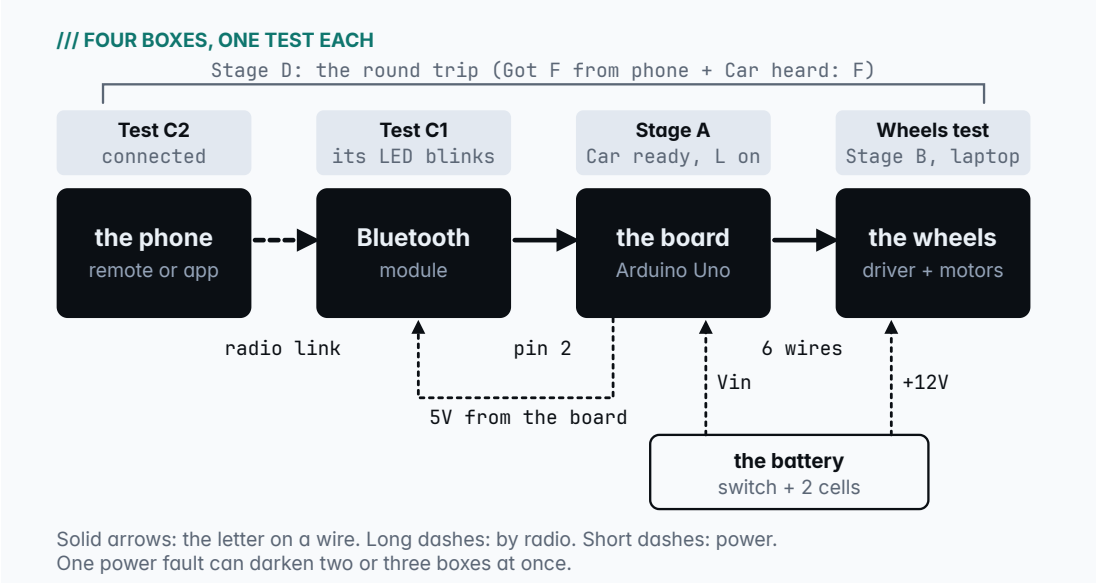


FIGURE 21.1 The car as a fault map. Each box has the stage test that proved it in Part 1. The power line feeds the motor driver and the board, and the board feeds the Bluetooth module.

In Part 1 you lit the boxes one at a time, and each stage ended with a test that only that stage could fail. Those tests did not stop being useful when the car first drove. **They are your instruments now.** When the car breaks next month, you will not need new tools to find out why. You will run the same tests and read the results differently: not "did I build this right?" but "which box stopped working?"

Table 21.1 is the key to reading them. The right-hand column matters most. Every test proves something, and every test leaves something unproved. Most wrong diagnoses come from believing a test proved more than it did.

TABLE 21.1 What each stage test proves, and what it leaves unproved

BOX	THE TEST	A PASS PROVES	A PASS DOES NOT PROVE
The board	Stage A: `Car ready` in the Serial Monitor, and the L light comes on and stays on	The board runs the car's sketch, and the laptop can talk to it	Anything about the wheels, the Bluetooth module or the battery
The wheels	The wheels test (Stage B): letters typed in the Serial Monitor drive each side the right way; the ON light stays lit when the USB comes out	The motors, the motor driver, the six signal wires, and the battery reaching both boards	Anything about Bluetooth
The Bluetooth module	Test C1: its LED blinks with the switch on	The module has power	That a letter can travel through it
The phone	Test C2: the remote (or, for the HC-06, the Android app) says *connected*, and the module's LED stops blinking	The phone and the module agreed to hold a link open	That a letter reaches the board
All four	Stage D: `Got F from phone` appears in the Serial Monitor **and** the phone shows `Car heard: F`	The whole round trip, both directions	—

The phone row is the one that fools people. *Connected* is a statement about two radios, not about your letter. A module with both data wires pulled out still connects. That is why Stage D exists, and why this part keeps coming back to it.

> ⓘ NOTE
> **HM-10 or HC-06?** Everything in this part works for both. The two modules use the same four wires to the same pins, run the same sketch and pass the same tests. Only the phone end differs: the HM-10 is driven from the Troniction remote in a browser, and the HC-06 from a free Android app. Where that changes a fault, the chapter says so.

The loop, again

Chapter 3, page 22 gave you five lines for when something doesn't work. Here they are again, with what an expert puts into each one.

1. **Look.** Say exactly what happens. "It doesn't work" is not a look. "It drives forward and back but won't turn left" is. Which letters, which wheels, which lights, which sounds? And always ask: **what changed since it last worked?** A car that drove yesterday and doesn't today has a short list of suspects: whatever was touched, moved, charged or uploaded since.
2. **Which box?** Use the stage tests to find the box. Don't open anything yet.
3. **Change one thing.** One wire, one setting, one line. Switch off first if it is a wire.
4. **Test again.** Run the same test as before, so the result means something. If the change didn't fix it, **undo it** before you try the next thing. Two changes at once and you can no longer tell which one did what, and a wrong change left in place is a second fault.
5. **Say what you found.** Out loud, or on paper: what was wrong, and how you knew. That sentence is what makes the next fault faster.

The undo step is the one people skip. Researchers who watched adults build an Arduino circuit saw fixes that didn't work pile up into new bugs (Booth and colleagues, 2016, a small lab study). The loop exists to stop that pile from starting.

Hardware first, but not hardware only

The same study is the best evidence for where to look first. Twenty adults, most of them experienced programmers, built a small Arduino circuit. Only 6 of the 20 finished. Faults in the circuit ended more attempts than anything else

(10 of the 20), and when something went wrong, people blamed their program about twice as often as their circuit. They looked in the wrong place first, and some of them never looked anywhere else.

So the habit to build is: **check the circuit before the sketch.** The stage tests are already in that order. They prove the board, the wires, the battery and the Bluetooth module before anyone asks what the sketch is doing.

But "hardware first" is not "hardware only". A setting in the Serial Monitor can make the car twitch instead of drive, and one changed line in a sketch can make a perfect circuit go silent. Two of the ten broken cars in Chapter 25, page 165 are exactly that. The rule is about **order**, not blame:

- a sketch you **did not change** since it last worked is the last suspect;
- a sketch or a setting you **did** change is one of the first, because it is "what changed".

⊗ MYTH

"If the car doesn't move, the code must be wrong."

> **WRONG** **This is wrong, and it is the most common wrong first guess in the subject.**
>
> The sketch is the one part of the car that cannot wear out, work loose or run flat. A sketch that drove the car yesterday is the same sketch today. Wires, cells, jumper caps and settings are what change between one drive and the next, so that is where faults usually live, and the stage tests look there first.

Half-splitting: test in the middle

Follow one letter from your thumb to the wheels and count the places it could be stopped: the phone, the radio link, the Bluetooth module, the wire into pin 2, the board and its sketch, the six signal wires, the motor driver, the motors. Eight places, plus the power line.

You could check them one at a time from the phone end. That is up to eight checks, and each one means unplugging and re-plugging things that were probably fine. Or you can test in the middle first.

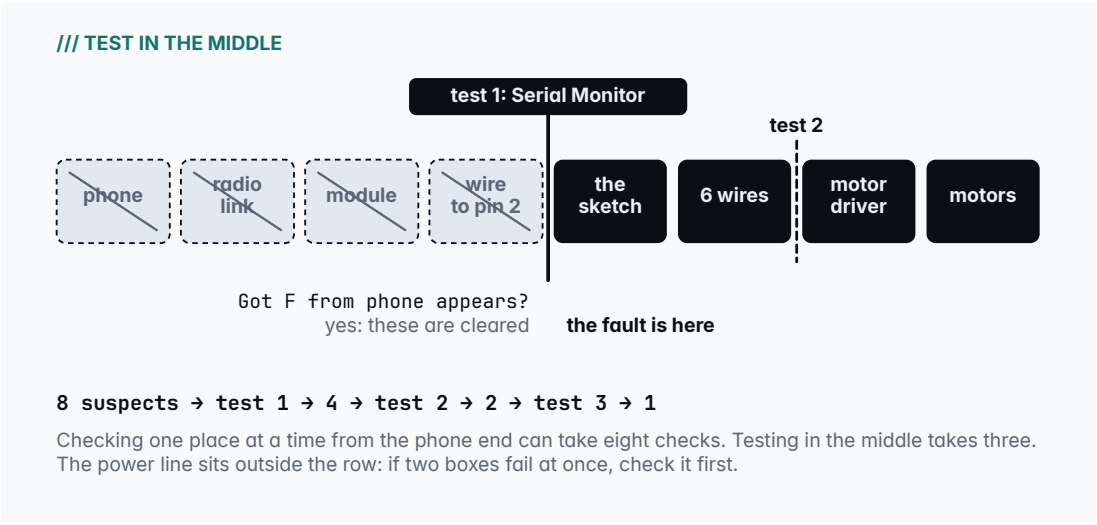


FIGURE 21.2 Half-splitting the car. One test in the middle, at the Serial Monitor, clears half the suspects. A second test splits the half that is left.

The middle of your car has a window in it: **the Serial Monitor shows every letter the board receives.** Press `F` on the remote and watch it.

- `Got F from phone` **appears.** The phone, the radio link, the Bluetooth module and the wire into the board all did their job. Cross off that half of Figure 21.2. The fault is after the board.
- **Nothing appears.** The board never got the letter. The wheels are not the problem yet. The fault is before the board.

One look, and eight suspects are four. Split the four that are left the same way and they are two, then one. Three tests instead of eight. That is half-splitting: **each test should cut the suspects in half, not cross off one at a time.** When the exact middle is awkward to test, split where a test is cheap and close to the middle. The stage tests are cheap splits you already own.

> ⓘ NOTE
> **How sure is this?** Half-splitting is how technicians have long found faults in long chains of parts, and it is sound reasoning: a test that can clear half the suspects beats a test that can clear one. But I have not found a study that tests it as a way of *teaching* children to fix things, so treat it as good practice, not as proven teaching.

The power line breaks the pattern

The boxes sit in a row, so most faults show up in one box and everything after it. The power line doesn't sit in the row. It feeds two boxes at once, and the board passes power on to the Bluetooth module. So **a power fault looks like two or three broken boxes at the same time**: the module's LED is out *and* the wheels won't turn from the laptop either.

Two boxes failing at once is very unlikely to be two separate faults. It is almost always the one thing they share. So the rule is: **when more than one box looks broken, check the power line before either box.** Is the holder's switch on? Are the cells charged? Is the lead to the board's Vin pin still in its terminal (the tug test from Stage B)?

TALK ABOUT IT

"We know the wheels work, because the laptop can drive them. Which one test tells us whether the phone's letter reaches the board?"

PARENT: LISTEN FOR:

watching the Serial Monitor while someone presses a button on the remote. Any answer that names the Serial Monitor is right.

CHECK YOURSELF

1. The phone shows *connected*, and the Serial Monitor shows each letter as you press it. Which half of the car is left to suspect, and which parts are now cleared?
2. Put the fix in order. The car won't drive from the phone. In what order would an expert do these? (a) Switch off, then check the Bluetooth module's TXD wire on pin 2. (b) Press `F` and watch the Serial Monitor. (c) Say what is happening: which lights are on, what the phone says. (d) Press `F` again and watch for `Got F from phone` and `Car heard: F`.
3. The Bluetooth module's LED is off, and the wheels don't turn from the laptop either. Which do you check first, and why? (This goes back to the battery line in Chapter 3, page 22.)

Answers at the back of the book on page 250

/// CHAPTER

22

Worked example: the speed that never changed

A fault nobody reported

The car this book is drawn from was built for the 2021 edition, and it runs a sketch first written in 2016. It drove forward, back, left and right. It beeped when it reversed and it played *Happy Birthday*. Nobody noticed anything wrong.

It had one anyway. **Its speed control had never worked.** In that sketch, from 2016 until August 2026, every speed button drove the car at full power, and nobody noticed for ten years.

This chapter walks through that fault the way an expert reasons about it, one thought at a time, using the loop from **Chapter 21**, page 143. Read the thinking, not just the answer. The answer is two wires; the thinking is what you will reuse on every fault after this one.

> NOTE
> **How it was really found.** Not at the bench. I found it in 2026 by *reading the sketch* while writing this book, then moved two wires on the real car and tested it. Reading can be a test, when what you read can only mean one thing. The steps below follow the order of the reasoning, which is the order you would use at the bench.

Look: what exactly happens?

The trouble with this fault is that the car looks fine. So the first step is to describe it precisely enough to see that something is wrong:

> "Speed `1` and speed `9` look the same. Every speed button gives the same speed, and it is fast."

That sentence is the whole symptom. Notice what it leaves out: the car still moves, still turns, still answers every letter. Only one feature is missing, and it is missing silently. No error light, no beep, nothing in any message.

What changed since it last worked? Here, nothing, and that is a clue too. A fault that has always been there was built in. Look at how the car was put together and programmed, not at what wore out.

Which box?

Go through the boxes.

- **The phone and the Bluetooth module** are fine: the car obeys every direction letter, so letters are arriving.
- **The board** is running the sketch: it obeys `F`, `B`, `L` and `R`.
- **The wheels** turn, both ways, on both sides.

Every box works, yet a feature is missing. So the fault is inside a box, in the part of it that handles *speed*. Speed lives in the wheels box: **Chapter 8**, page 52 split the six signal wires into four direction wires (IN1 to IN4) and two speed wires (ENA and ENB). Direction works, so the direction wires are fine. **That splits the wheels box in half, and only the speed half is left.**

Two suspects, cheapest first

What can make every speed the same? Two things, and an expert lists both before testing either:

1. **The jumper caps on ENA and ENB.** Many motor driver boards ship with a small cap on each speed header. A cap ties that speed input to full power, so the motor driver ignores the board's speed wire completely. That gives exactly this symptom.

2. **The speed wires are on pins that can't do speed.** Only some of the board's pins can make the fast on–off pattern that sets speed (Chapter 17, page 116 explains the pattern). On the board those are pins 3, 5, 6, 9, 10 and 11, marked with a `~` on the board.

Which first? The caps take five seconds to look at. Pins take a little longer: you read two lines of the sketch and follow two wires. Both are cheap, so check the caps first, then the pins. On this car the caps were off (the final test below proves it). That leaves the pins.

Read the pin, then predict

The 2016 sketch named its pins at the top, and two of the lines were these:

troniction-arduino-car.ino (2016, before the fix) LISTING 22.1

```
#define enA 13
#define enB 8
```

Pin 13 and pin 8. Neither has a `~`. Neither is on the list.

An expert's next move is not "change it and see". It is **predict**: if the pins are the fault, what exactly should the car do? To answer that you need one fact about the board. When a sketch asks a pin that can't do speed for a speed, the board doesn't refuse and doesn't warn. It rounds. Ask for anything under 128 (out of 255) and the pin is simply off. Ask for 128 or more and the pin is simply on, at full power.

Now look at the speeds the sketch asked for. The car's speed buttons send `0` to `9`, and the sketch turned them into 150, 150, 170, 180, 200, 210, 220, 230, 240 and 250. Every one of them is 128 or more.

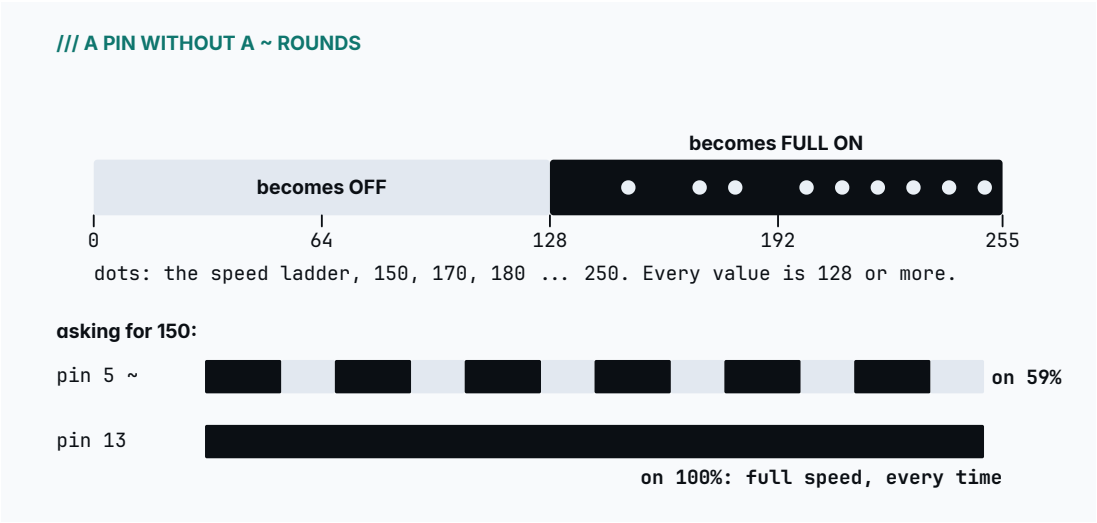


FIGURE 22.1 Why every speed was the same. On a pin that can't do speed, every value of 128 or more becomes full power, and the car's whole speed ladder sits above 128.

So the prediction is: **every speed button drives at full power.** That is exactly the symptom. A prediction that matches the symptom in every detail is strong evidence. It also explains why the car never crawled, and why nothing ever looked broken.

Change one thing, test again

Which pins to move to? The board's pins 2 and 8 to 13 were already in use on this car, so of the six speed pins only 3, 5 and 6 were free. The fix is one change, made in two places that must agree: the ENA wire moves from pin 13 to pin 5 and the ENB wire from pin 8 to pin 6, and the two lines of the sketch change to match. The four direction wires don't move.

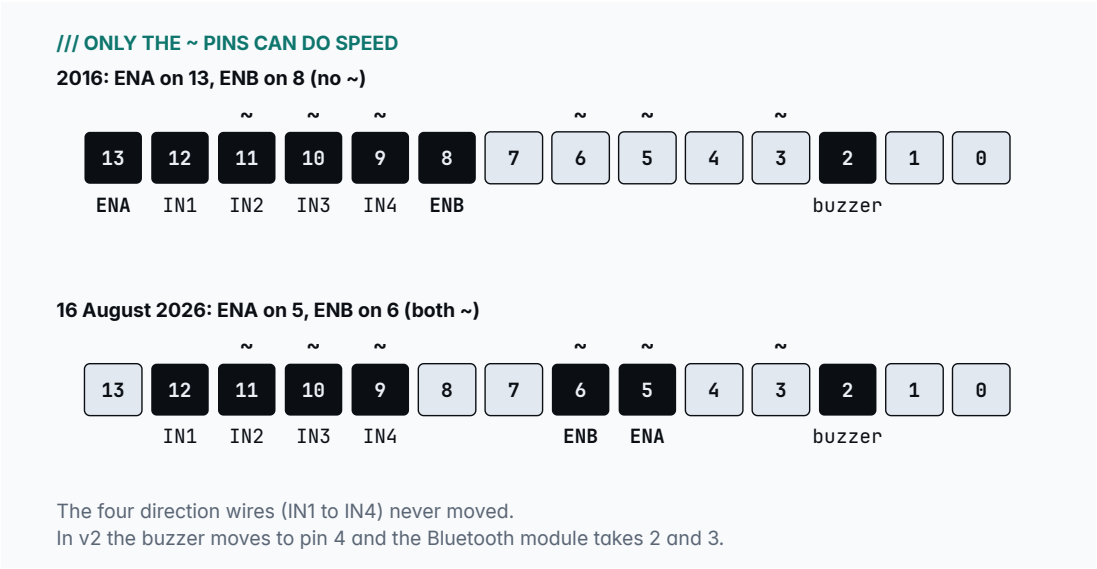


FIGURE 22.2 The two speed wires before and after the fix. Only the pins marked with a tilde can control speed.

Then the same test as the symptom: on the box, speed `1`, then speed `9`. On 16 August 2026 the real car passed it. **Speed 1 crawls and speed 9 runs.** That result also proved the caps were off, because with a cap on, speed 1 could not crawl.

Say what you found

The last step of the loop happened in the sketch itself. The fixed file starts with a note that says what was wrong, how it was found and what changed, so the next person to open it doesn't have to rediscover it. Your version of the same step can be one line in a notebook: *"Every speed the same → speed wire on a pin without a `~` → moved it."*

Why nobody noticed

Three reasons, and each one is worth knowing because it will hide other faults too.

- **It failed silently.** The board rounds instead of complaining. Nothing told anyone.

- **The car still worked.** People test what they expect a car to do: go, stop, turn. A car that does all three passes, even with a feature missing.
- **Full speed is a believable speed.** Nobody had seen the car crawl, so nobody missed it. You only notice a missing difference if you test for the difference.

That last point is why the wheels test in Chapter 9, page 59 asks for speed `1` and speed `9` one after the other. A test that compares two things catches faults that a test of one thing can't.

TALK ABOUT IT

"The car drove for ten years with this fault. What test would have caught it in the first week?"

PARENT: LISTEN FOR:

"try the slowest and the fastest speed and compare them". Any test that compares two speeds is right.

CHECK YOURSELF

1. Why did nobody notice the fault for ten years? Give two reasons.
2. Name a second fault with exactly the same symptom, "every speed is the same", that has nothing to do with the sketch. (It comes from Chapter 8, page 52.)
3. Find the fault in this pin list, and predict what the car does. A friend wired ENA to pin 5 and ENB to pin 7; the four direction wires are on 12, 11, 10 and 9, and the sketch matches the wires. What happens at speed `1`?
4. Put the expert's steps in order: (a) predict what the car should do if the pins are the fault; (b) list the two things that can make every speed the same; (c) check that the direction letters all work; (d) read the two speed-pin lines of the sketch; (e) look at the jumper caps on ENA and ENB.

Answers at the back of the book on page 250

/// CHAPTER

23

Worked examples: spins on the spot, and connected but still

From a story to a form

Chapter 22, page 149 talked through every thought. That is the right way to meet a method, but nobody fixes a car by writing an essay. Experts keep the same steps and drop the words. This chapter gives you the steps as a form, the **fault skeleton** in Table 23.1, and two faults to use it on. I fill in the first one. You fill in most of the second.

TABLE 23.1 The fault skeleton. One row per step of the loop

STEP	THE QUESTION IT ASKS
Look	What exactly happens? What changed since it last worked?
Which box?	Which stage test fails? Which boxes are cleared?
Split	Which one test cuts the remaining suspects in half? What did it show?
Suspects	What could cause exactly this? Cheapest to check first
Change one thing	What single change, with the power off if it's a wire?
Test again	Which test, and what result means it's fixed?
Say what you found	One sentence: the fault, and how you knew

It works on any fault in this book, and in any project after it.

Example 1: it spins on the spot

The car is new. The pair has reached the wheels test in Stage B: the car is on its box, the USB is in, and the child types `F` in the Serial Monitor.

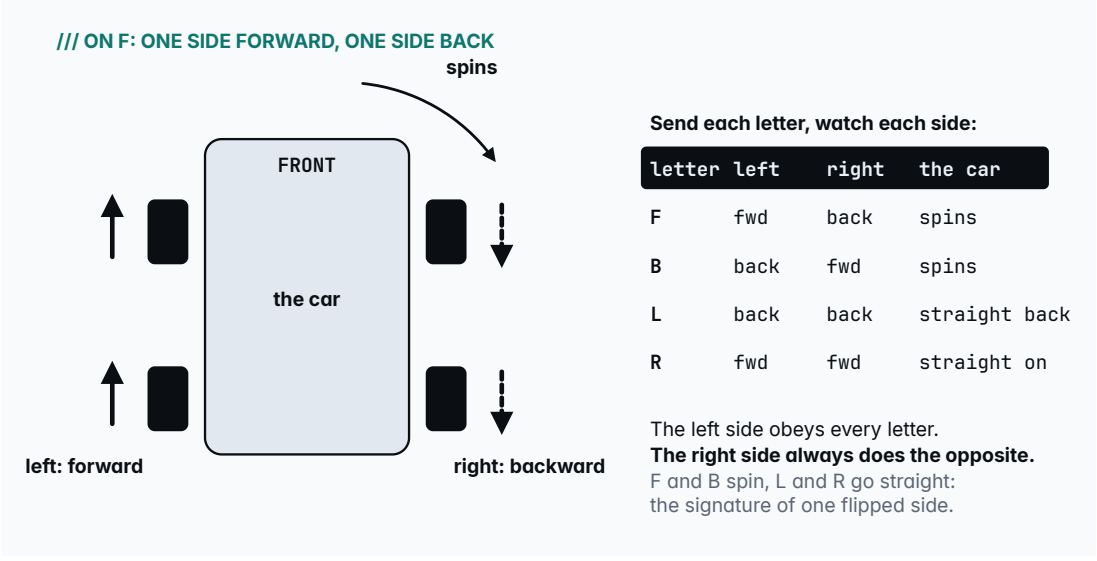


FIGURE 23.1 Example 1 seen from above. On F, the left wheels turn forward and the right wheels turn backward, so on the floor the car would spin on the spot.

TABLE 23.2 Example 1, the skeleton filled in

STEP	EXAMPLE 1, FILLED IN
Look	On `F`, the left wheels turn forward and the right wheels turn **backward** (Figure 23.1). On the floor this car would spin on the spot. The Serial Monitor says `Got F from USB`
Which box?	The wheels. The letter came from the laptop, so the phone and the Bluetooth module are not even involved. `Got F from USB` proves the board read it
Split	Is it one side or the whole car? Send `L` and `R` and watch each side. On `L` both sides now turn backward; on `R` both turn forward. **The left side obeys every letter; the right side always does the opposite of what it should.** The fault is in the right half of the wheels box
Suspects	The right motors' red and black wires the other way round at OUT3 and OUT4 (cheap: look at the terminal). IN3 and IN4 swapped at the board (cheap too: read two wires against the pin table). Both reverse the right side and nothing else
Change one thing	The IN wires match the pin table, so: switch off. At OUT3 and OUT4, swap the red and black wires of both right motors. Tighten both screws and tug each wire
Test again	Switch on, send `F`. All four wheels turn forward. Send `L` and `R`: the car turns each way on the spot, as it should
Say what you found	"The right side ran backwards on every letter, so its motor wires were the other way round. We swapped red and black at the motor driver."

Two things in that table are worth stopping on.

The split used letters, not tools. `L` and `R` cost nothing and cut the suspects in half: one side behaved, the other side didn't, so only the parts that belong to one side were left. The give-away was that `L` drove the car *straight*. A car turns only when its two sides turn differently, so "turn left" driving straight backward means one side has its direction flipped.

This isn't really a fault. Which way a motor turns depends on how it sits on your chassis, and nobody can tell from a picture. That's why Chapter 9, page 59 makes it a guess-then-test instead of a rule. Even the original car shows it. In its sketch, "forward" is a different pattern of signals on each side: one side gets low-then-high, the other high-then-low, because the two sides' motors face opposite ways. You won't find this fault by reading. You find it by testing.

TALK ABOUT IT

“The car spins when we press Forward. Before we touch anything: which side is going the wrong way, and how do we know?”

PARENT: LISTEN FOR:

“we watch each side while we send `F`”, and then naming the side whose wheels turn backward. A guess before watching is fine; the watching decides.

Example 2: connected, but still

This one is yours. The car passed the wheels test yesterday: it drove on its box from the laptop. Today the pair wired the Bluetooth module, and the phone connects. The module’s LED has stopped blinking. The phone says *connected*. The child presses Forward, and nothing happens.

Fill in the skeleton in Table 23.3 before you read on. I have started it for you. A pencil on the page is fine; so is saying each answer out loud.

TABLE 23.3 Example 2, for you to fill in

STEP	EXAMPLE 2: YOUR TURN
Look	LED steady, the phone says *connected*. Forward does nothing. Nothing new appears in the Serial Monitor, and the phone shows no `Car heard`. What changed since it last worked: the module and its four wires
Which box?	Which boxes does the wheels test from yesterday clear? Which does *connected* clear? What’s left?
Split	Type `F` in the Serial Monitor. What two things do you watch for, and what does each one tell you?
Suspects	What could give exactly what your split showed? Cheapest to check first
Change one thing	
Test again	
Say what you found	

Fill in Look, Which box and Split first. Then read on.

NOTE

When you've written your split. Here is what it shows on this car. Typed in the Serial Monitor, `F` prints `Got F from USB` and the wheels turn on the box. But the phone shows nothing: no `Car heard: F`. Now write your suspects. Then look at Figure 23.2, which shows where the four module wires actually go, and compare each one with steps C7 and C8 in Chapter 10, page 70 (turn back to it). One thing is wrong.

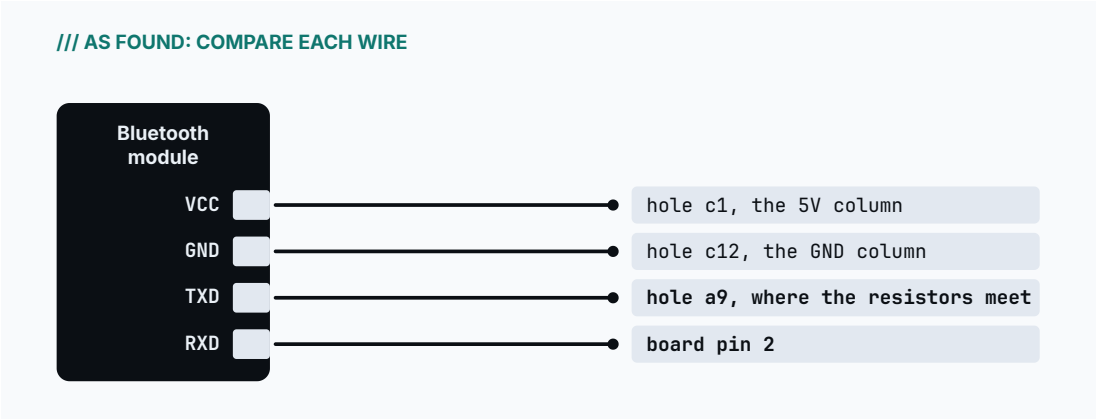


FIGURE 23.2 Example 2: where the module's four wires really go. Compare each wire with the wiring in Chapter 10, page 70. One thing is wrong.

NOTE

HM-10 or HC-06? The same. Both modules use the same four wires to the same pins, so the same fault gives the same symptom. Only the phone end differs: "connected" is the remote's status line for the HM-10, and the Android app's for the HC-06. On either one, `Car heard: F` is what comes back.

How this example was checked. Like every fault in this part, it was worked out from the circuit, the sketch and the datasheets, not staged on a car (Example 1 too). The sketch's side of it is tested on a computer.

When you have filled in every row, check your answer at the back. Then compare it with Example 1: the steps were the same, only the box was different. That is the point of the form.

Connected but nothing arrives: the five causes, in order. (Its examples use pins 10 and 11; your car uses 2 and 3.)

SCAN OR TYPE **troniction.com/go/car-no-data**

The symptom index: every symptom in this book, the stage it lives in, and the fix.

SCAN OR TYPE **troniction.com/go/car-stuck**

CHECK YOURSELF

1. Finish Table 23.3: fill in every empty row for Example 2.
2. Find the fault in Figure 23.2. Which two wires are wrong, and is anything damaged?
3. Predict, for Example 1 before the fix: you send `B`. What does each side do, and what does the car do on the floor? (This uses the patterns from Example 1's split.)
4. Put the fix for Example 1 in order: (a) send `F` and watch all four wheels; (b) switch off; (c) tighten both screws and tug each wire; (d) at OUT3 and OUT4, swap the red and black wires of both right motors.

Answers at the back of the book on page 251

/// CHAPTER

24

Your Serial Monitor is a probe

A window in the middle of the car

An electrician has a meter. You have something better for this car: the Serial Monitor. The car's sketch writes a line for every letter the board receives, so the Serial Monitor shows you the inside of the board, at the middle of the car, while it runs. That makes it the half-splitting test from **Chapter 21**, page 143: it tells you which side of the board a fault is on.

Plug in the USB, open the Serial Monitor, and set its two menus at the bottom: **9600 baud** and **No line ending**. Figure 24.1 shows what a short drive looks like.

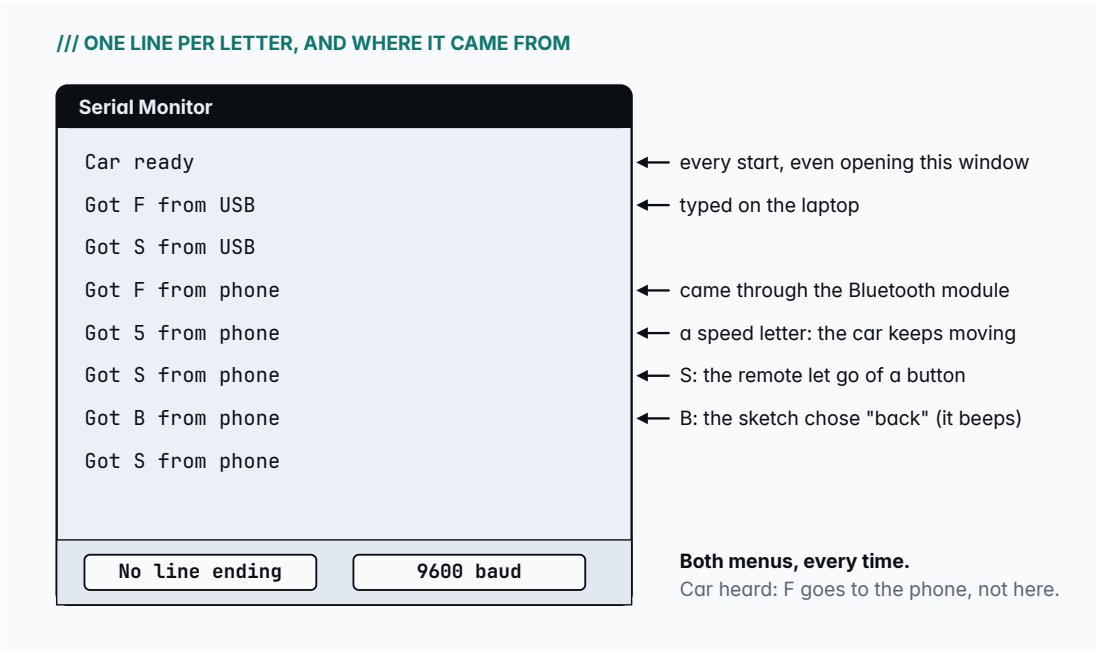


FIGURE 24.1 The Serial Monitor during a short drive, with each kind of line labelled. Every letter the board receives gets one line, saying where it came from.

Five questions it answers

1. Is the sketch running? Every time the board starts, it prints `Car ready` and turns the L light on, and the light stays on. Opening the Serial Monitor restarts the board, so you will see the L light flicker quickly a few times (that's the board's own start-up, before the sketch runs), then come on and stay on, and `Car ready` appear. That's normal. No `Car ready`, and an L light that blinks slowly forever, means the board is still running the sample sketch it came with: upload the car's sketch again (Chapter 6, page 39).

2. Did the letter arrive, and from where? Every letter gets a line: `Got F from USB` when you typed it in the Serial Monitor, `Got F from phone` when it came through the Bluetooth module. If you press Forward on the phone and no `Got F from phone` appears, the letter never reached the board. Everything after the board is cleared from suspicion, and everything before it is suspect.

3. Did it arrive the way you meant? The sketch names letters you can't see. `Got (newline)` and `Got (return)` are the invisible "Enter" characters some apps and settings add after every letter; `Got (space)` is a space. The car treats every one of them as "anything else", which means **stop**. So a

`(newline)` line straight after every letter tells you why the car barely twitches instead of driving (the fix is the **No line ending** menu, or the same setting in an app). And a run of `(code …)` lines, or odd symbols, means the two ends of a link disagree about speed: the bytes are arriving, but at the wrong rate.

4. Did the answer get back? For every letter, the board also answers through the Bluetooth module: `Car heard: F`. That answer goes to **the phone**, not to the Serial Monitor. Here's a trick an expert uses: type a letter **in the Serial Monitor**, and the answer still goes out to the phone. If the phone shows `Car heard: F` for a letter you typed on the laptop, the return path works on its own (board → pin 3 → the two resistors → the module → the phone), and you have tested one direction of the link without the other.

5. Did the board restart? A `Car ready` you didn't cause, one that isn't just after you opened the Serial Monitor or pressed upload, means the board restarted by itself. A board that restarts on its own is nearly always being starved of power. Look at the power line before anything else (**Chapter 19**, page 128).

What it can't show

The Serial Monitor shows what **arrived**. It doesn't show what the sketch **decided** to do with it: for that, watch the wheels and listen. The reversing beep is the sketch telling you it chose "back". In Part 4 you will learn to make the sketch print its decisions too (**Chapter 27**, page 181).

It also needs the USB cable, so the car is tied to the laptop. Use it with the car on its box, or on the floor within reach of the cable. And the USB powers the board, so a fault in the battery's feed to the board can hide while the cable is in. That's why the wheels test ends by unplugging the USB and checking that the board's ON light stays lit.

> TIP
> **Settings first, every time.** Before you believe anything the Serial Monitor shows, check its two menus: 9600 baud and No line ending. Most "impossible" readings are one of those two menus.

TABLE 24.1 The probe with each module. The sketch and the wiring are the same; only the phone end differs

MODULE	WHAT THE SERIAL MONITOR SHOWS
HM-10, driven from the remote	`Got F from phone` for each button; the remote shows `Car heard: F`
HC-06, driven from the Android app	The same lines. In a terminal app you type the letters yourself, so it shows exactly what you typed: lower-case letters, spaces and Enter included



Odd symbols in the Serial Monitor: why they mean the link works, and the one number to change.

SCAN OR TYPE **troniction.com/go/car-garbage**

CHECK YOURSELF

1. You open the Serial Monitor and see a line of odd symbols instead of `Car ready`. Which setting do you check, and what should it say?
2. Predict: the line-ending menu says *Newline*. You type `B` and press Enter. What two lines appear, and what does the car do?
3. Find the fault in this Serial Monitor. A child with the HC-06 build types in the Android app and says "it won't go". The Serial Monitor shows `Got f from phone`. What's wrong, and what is the fix?
4. Put the probe checks in order. The car won't drive from the phone. (a) Press Forward and look for `Got F from phone`. (b) Check the two menus: 9600 baud and No line ending. (c) Look for `Car ready` and the steady L light. (d) Type `F` in the Serial Monitor and watch the phone for `Car heard: F`.

Answers at the back of the book on page 251

/// CHAPTER

25

Ten broken cars

How to use the ten cars

Here are ten cars, each with one fault. You get what an owner would see and nothing else. The reasoning is yours now.

For each car, name **the box**, **the part**, and **the one change** that fixes it. Use the loop and the skeleton from **Chapter 23**, page 155. If you're stuck, there are two hints per car at the end of the chapter: the first points at the box, the second at the test. The answers, with the reasoning, are at the back of the book. As a pair, one of you reads the car out and the other plays the expert; swap for the next car.

A few of these faults aren't wires. One is a setting and one is a change to the sketch, because real faults are sometimes exactly that. They are mixed in with the others on purpose.

> ⓘ NOTE
> **How these cars were checked.** These ten faults were reasoned out, not staged: none of them was set up on a real car for this edition. The motor wiring and the sketch's letters are the original car's, which drove for years; the module wiring is checked by compiling and testing the sketch on a computer; each symptom is worked out from the circuit, the sketch and the parts' datasheets. Each answer says how sure it is. If your car shows something different, I want to hear about it: troniction.com/go/car has the address.

TABLE 25.1 The ten broken cars at a glance

CAR	WHAT THE OWNER SAYS
1	“Every speed is the same speed.”
2	“The right side is dead.”
3	“Forward goes backward, and it beeps going forward.”
4	“It jerks and stops when I type a letter.”
5	“It works with the laptop, and dies on the floor.”
6	“Yesterday speed 1 crawled. Today it just hums.”
7	“It drives, but it never answers.”
8	“The phone can’t drive it, but the laptop can.”
9	“I updated the sketch and now it ignores the phone.”
10	“My friend’s iPhone can’t find it.”

Car 1 · Every speed the same

Applies to: both modules.

A friend borrowed the car for a weekend. It came back driving fine, but now speeds `1` to `9` all run at the same fast speed. Every direction works. The Serial Monitor shows `Got 1 from phone`, `Got 9 from phone` and so on, exactly as it should. Figure 25.1 shows the motor driver as it came back.

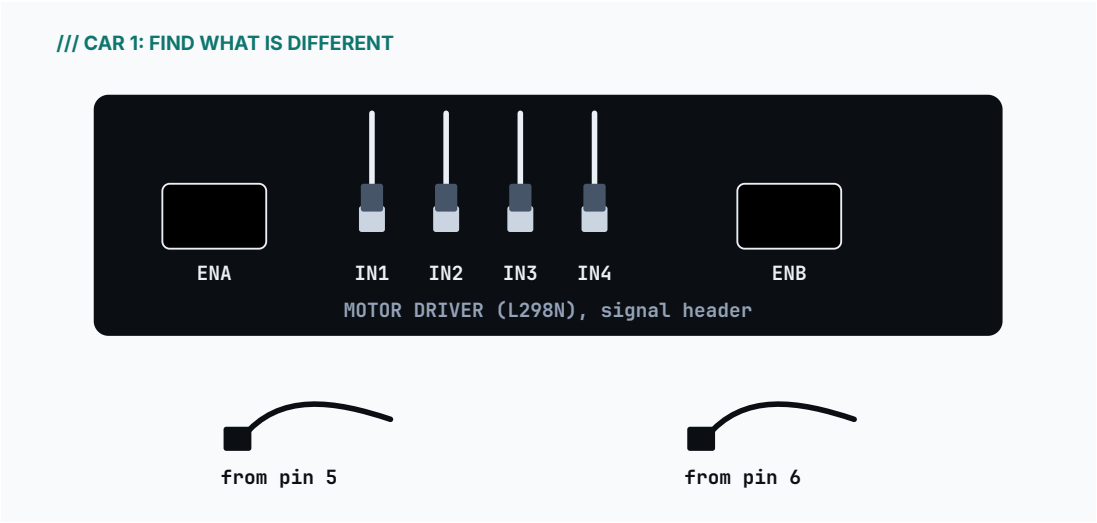


FIGURE 25.1 Car 1. The corner of the motor driver with the two speed headers, ENA and ENB, as the car came back. Find what is different from your own car.

Car 2 · The right side is dead

Applies to: both modules.

On the box, from the laptop: `F` turns the left wheels forward and the right wheels not at all. `B`, `L` and `R` do the same: the left side obeys every letter, the right side never moves. Every letter appears in the Serial Monitor. The car was dropped off a table yesterday.

Which parts does **only** the right side use? List them, then choose the one to check first.

Car 3 · Backwards, and beeping

Applies to: both modules.

A new car, just through Stage B. Forward drives it backward, silently. Back drives it forward, **beeping**. Left turns it right and right turns it left.

Is the board confused, or the wires? The beep tells you. Then decide what to change, and predict what the car will do after your **first** change, before the second.

Car 4 · Jerks and stops

Applies to: both modules.

The wheels test. The child types `F` in the Serial Monitor and presses Enter. The wheels give, at most, a tiny jerk and stop. `B`, `L` and `R` do the same. The car never really moves. The motor driver, the wires and the battery have all been checked twice.

Before you touch a wire: what would you look at, and what would you expect to see there?

Car 5 · Fine with the laptop, dead on the floor

Applies to: both modules.

On the box, with the USB in, the car passed every test, including driving from the phone. The pair skipped the last step of the wheels test. Now the USB is out and the car is on the floor. The holder's switch is on and the cells are freshly charged. Nothing responds. The board's ON light is out, and so is the Bluetooth module's light.

Two boxes look dead at once. What does that tell you?

Car 6 · It used to crawl

Applies to: both modules.

Yesterday speed `1` crawled and speed `9` ran. Today, after a long afternoon of driving, speeds `1`, `2` and `3` only make the motors hum; the wheels don't turn. From speed `4` the car moves, but slowly. Every letter arrives in the Serial Monitor, and nobody has touched a wire.

What changed since it last worked?

Car 7 · It drives, but never answers

Applies to: both modules.

The phone connects. Forward drives the car, and the Serial Monitor shows `Got F from phone`. But the phone never shows `Car heard: F`. Typing `F` in the

Serial Monitor turns the wheels, and the phone still shows nothing. Figure 25.2 shows the two resistors on the breadboard.

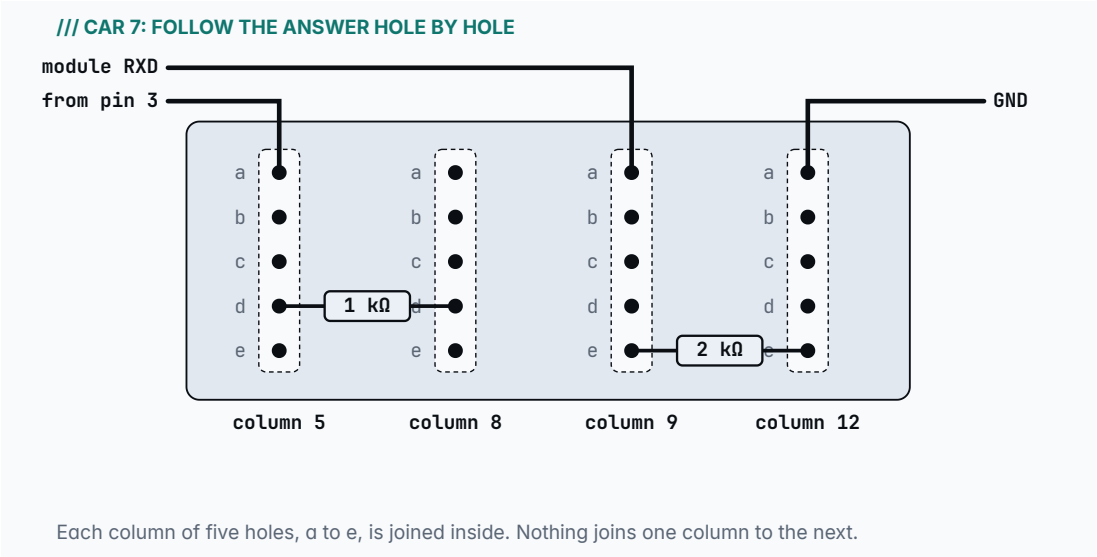


FIGURE 25.2 Car 7. The two resistors and the module's listening wire on the breadboard, as found. Each column of five holes is joined inside; nothing joins one column to the next.

Car 8 · The laptop can, the phone can't

Applies to: both modules.

The phone connects, and the module's light goes steady. Forward does nothing, and no `Got F from phone` appears. But type `F` in the Serial Monitor and two things happen: the wheels turn, **and the phone shows** `Car heard: F`.

The second half of that is the clue. Which parts has it just cleared?

Car 9 · "I updated the sketch"

Applies to: both modules.

The car drove last week. Since then, someone read a Bluetooth tutorial, changed one line of the sketch and uploaded it. Now the phone connects, but every press of Forward stops the car instead of driving it, and the Serial Monitor fills with lines like `Got (code 128) from phone`: one or more for every press, and never `Got F`. Typing `F` in the Serial Monitor still drives the wheels. Listing 25.1 is the changed line.

car.ino (Car 9, after the update) LISTING 25.1

```
#define BT_BAUD 38400      // the factory speed of the HC-06 and the HM-10
```

The hardware has already told you it's fine. How?

Car 10 · The iPhone that can't find it

Applies to: the HM-10 build. (An iPhone can't connect to an HC-06 at all; that's why the book offers the HM-10.)

The car drives perfectly from an Android phone with Chrome. A visiting friend opens the remote page on her iPhone, in Chrome, and presses Connect. No car appears in the list, or the page says Bluetooth isn't available (the exact words depend on the browser). The car hasn't changed.

Which box is this, and what does the friend need?

Hints

Read only the hint for the car you're on. The first hint points at the box; the second at the test or the part.

1. **Car 1.** (a) Direction works and the letters arrive: only the speed half of the wheels box is left. (b) Two things can make every speed the same. One of them is visible in Figure 25.1.
2. **Car 2.** (a) One side obeys every letter, so the board and the letters are fine. (b) The right side alone uses: the ENB wire, the IN3 and IN4 wires, the motor wires in OUT3 and OUT4, and the two right motors. Which can a fall loosen, and which is quickest to check?
3. **Car 3.** (a) The beep only sounds when the sketch has chosen "back". (b) Compare with Example 1 in Chapter 23, page 155: there, one side was flipped. What if both are?
4. **Car 4.** (a) Chapter 24, page 161, question 3 of the five. (b) Look at the line straight after `Got F from USB`.
5. **Car 5.** (a) Chapter 21, page 143: when more than one box looks broken, check the line they share. (b) The battery feeds the board through two leads. The wheels test's last step checks them.

6. **Car 6.** (a) Nothing was touched; something was used. (b) Speed is the share of time the motors get the battery's push. What happens to the push itself as cells run down?
7. **Car 7.** (a) Letters arrive; answers don't. Which direction is broken? (b) The answer's path is pin 3 → column 5 → the 1 kΩ → column 9 → the module's RXD. Follow it hole by hole in **Figure 25.2**.
8. **Car 8.** (a) The answer reached the phone, so the return path, the module's power and the radio link all work. (b) Only one wire is used by phone-to-board letters and by nothing else.
9. **Car 9.** (a) A line appears for every press: the letters are arriving, so the wires and the radio link work. (b) A line arrives for every press, but never the letter you sent. Which setting has to match at both ends, and what does the HC-06 or the HM-10 use out of the box?
10. **Car 10.** (a) The car, the module and the remote page all work on Android. (b) On an iPhone, which engine is Chrome built on?



The ten broken cars online, with tap-to-reveal hints.

SCAN OR TYPE **troniction.com/go/car-faults**

Ten cars, and every box of the car has broken at least once. When a fault is in the link between the phone and the board and you want to go deeper, *Make It Connect* (troniction.com/go/mic) gives its whole Part 5 to link faults.

CHECK YOURSELF

For each car, name the box, the part and the one change that fixes it.

1. Car 1 · Every speed the same
2. Car 2 · The right side is dead
3. Car 3 · Backwards, and beeping
4. Car 4 · Jerks and stops
5. Car 5 · Fine with the laptop, dead on the floor
6. Car 6 · It used to crawl
7. Car 7 · It drives, but never answers
8. Car 8 · The laptop can, the phone can't
9. Car 9 · "I updated the sketch"
10. Car 10 · The iPhone that can't find it

Answers at the back of the book on page 251

4

/// PART 4

Program the car

Your car runs a sketch: about 290 lines that decide what every letter from the phone does. In this part you read it, predict what it will do before you run it, and then change it. You will find a real bug that has been in the car's sketch since 2016, and by the last chapter you will have added a letter of your own.

You need the car on its box with the wheels off the table, the USB cable, and the laptop with the Arduino IDE from Chapter 6. Every experiment here works from the Serial Monitor; the phone is optional. Each chapter follows the same pattern: **predict** what will happen, **run** it, **look inside** to see why, then **change** something and test it.

/// CHAPTER

26

Reading a sketch: once, then forever

You uploaded this sketch in Chapter 6 and it has driven the car ever since. You haven't needed to read a line of it. This chapter shows you how: what the parts of the file are, which part runs once and which part runs forever, and how to say what the car will do *before* you try it.

Set up the bench

Every experiment in this part uses the same set-up. Do it once now.

> ⚠ WARNING
>
> **Wheels off the table before any letter.** In this part you send driving letters from the laptop. Put the car on its box with the wheels in the air, as you did for the wheels test in Chapter 9, before you open the Serial Monitor. A car that drives off a desk with the USB cable attached pulls the laptop with it.

CHILD 1 Put the car on its box, wheels in the air. Plug the USB cable into the board and the laptop.

CHILD 2 Open `car.ino` and the Serial Monitor. Set the Serial Monitor to **9600 baud** and **No line ending** (the two menus at the bottom of its window).

▲ CHILD 3 Switch the battery on only when an experiment needs the wheels, and off after it; your parent checks the wheels are in the air first.

The board itself runs from the USB cable, so the Serial Monitor works with the battery off.

Predict first

Before you look at the sketch, write down your answers to these two questions. Guessing is fine; the point is to have something to check.

1. When the board starts, what will the Serial Monitor show?
2. You type `F` and press Enter. What will the Serial Monitor show, and what will the wheels do?

Now run it: press the board's reset button, then type `F` and press Enter.

⏱ TEST IT

YOU SHOULD SEE: `Car ready` appears when the board starts. After `F`, the line `Got F from USB` appears, and with the battery on, the wheels turn forwards and keep turning.

A INSTEAD: `Got F from USB` followed straight away by `Got (newline) from USB`, and the wheels barely twitch or don't move.

MEANING: the Serial Monitor is adding an invisible "end of line" letter after yours, and the car obeys it. Chapter 27 explains exactly why.

FIX: set the line-ending menu to **No line ending** and send `F` again.

Were your predictions right? Most people get the first one and miss the second: they expect the car to move for a moment and stop. It doesn't stop. It keeps going until it is told something else. By the end of this chapter you will be able to point at the line that makes that happen.

The map of the file

A sketch is read from top to bottom, but it doesn't *run* from top to bottom. Figure 26.1 shows the six parts of `car.ino` in the order they appear in the file.

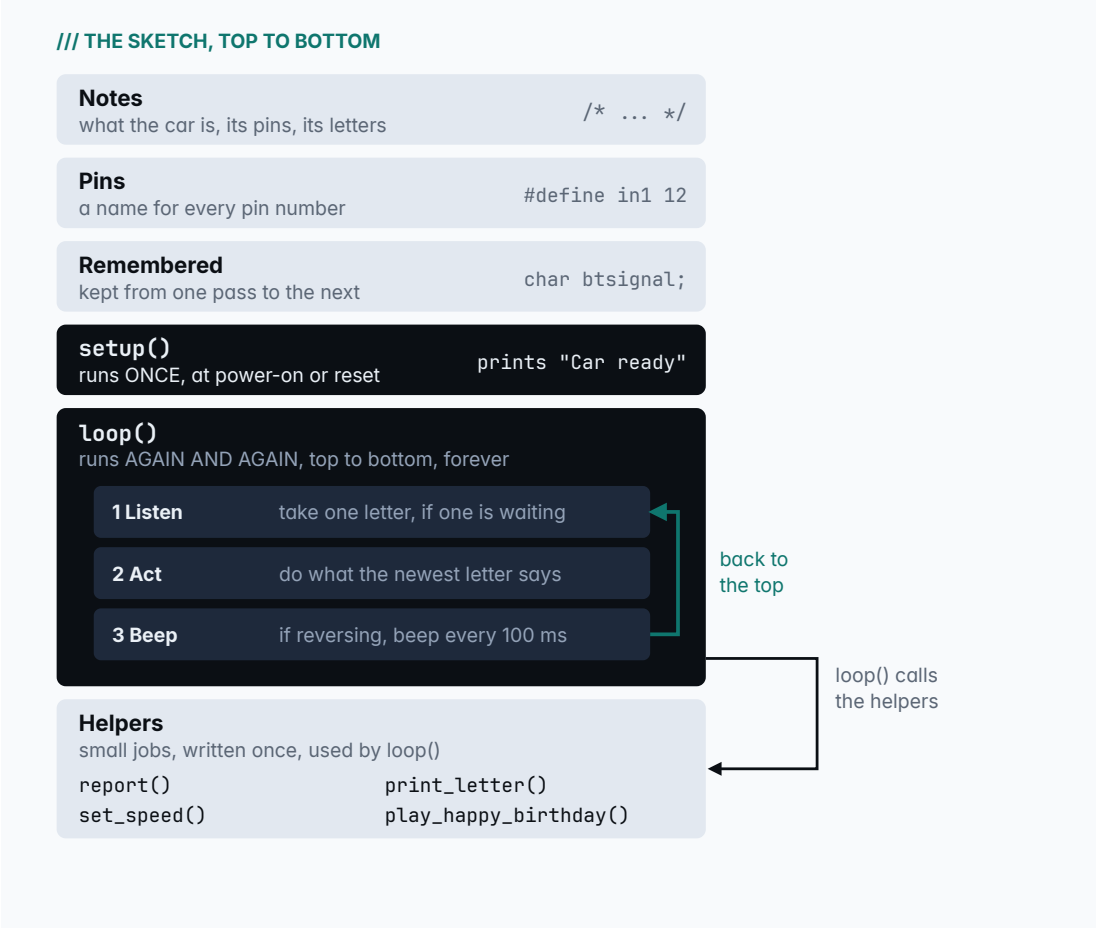


FIGURE 26.1 The six parts of the car's sketch. `setup()` runs once; `loop()` runs again and again, and calls the helpers below it.

The notes at the top, between `/*` and `*/`, are for people. The board never sees them. They list the pins and the letters, so they are the first thing to read when you open a sketch you didn't write.

The pins. Lines like `#define in1 12` give a pin number a name. From then on, the sketch says `in1` and the board uses pin 12. If you ever move a wire to a different pin, you change one number here, not every line that uses it.

Remembered. Four lines make boxes that hold a value from one pass of `loop()` to the next. The important one is `btsignal`, which holds the newest letter.

`setup()` runs once, when the board powers up or restarts. `loop()` runs after it, again and again, for as long as the board has power. **The helpers** at the bottom are small jobs with names, like `set_speed()`, written once and used from `loop()`.

Once: `setup()`

Here is the end of `setup()`, the part you can see happen.

car.ino LISTING 26.1

```
  set_speed(150);

  digitalWrite(in1, HIGH);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, HIGH);

  Serial.begin(9600);
  bluetooth.begin(BT_BAUD);

  Serial.println("Car ready");
  digitalWrite(readylight, HIGH);   // L stays on: this sketch is running
}
```

Read it as a list of orders, one per line, done in order:

- set the speed to 150 (the slowest step of the speed ladder, Chapter 29);
- set all four motor-direction pins HIGH, which holds the wheels still (Chapter 28 says why);
- start talking to the laptop, and to the Bluetooth module, both at 9600;
- print `Car ready`;
- switch on the L light.

That is why the Serial Monitor says `Car ready` every time the board restarts, and why opening the Serial Monitor makes it appear again: opening it restarts the board, and `setup()` runs once more.

Forever: `loop()`

`loop()` is the whole car. Strip out the details and it has three steps, each marked in the sketch by a comment line that starts with `// ----`:

1. **Listen:** if a letter is waiting, from the USB cable or from the module, take it.
2. **Act:** do what the newest letter says.
3. **Beep:** if the car is reversing and 100 milliseconds have passed, beep.

When the board reaches the end of step 3 it goes straight back to step 1. It never waits for you. Most passes find no new letter at all; they just do step 2 again with the letter they already have.

That is the answer to the prediction. Look at the Listen step:

car.ino LISTING 26.2

```
// ---- Listen: take one letter, from the USB cable or the module ----
if (Serial.available()) {
  btsignal = Serial.read();
  report(btsignal, "USB");
}
else if (bluetooth.available()) {
  btsignal = bluetooth.read();
  report(btsignal, "phone");
}
```

`btsignal` changes **only** when a letter arrives. On every other pass it keeps the old one, and the Act step obeys the old one again. You sent `F` once; the car does `F` on every pass until a new letter replaces it. The letter is “sticky”.

MYTH

> "`loop()` does several things at the same time: it listens, drives and beeps all at once."

WRONG **This is wrong. The board does one line at a time, in order, and never two at once.**

It *looks* simultaneous because each pass is short and the board goes round again straight away, so listening, driving and beeping take turns many times a second. Chapter 30 shows what happens when one step takes a long time: everything else waits for it, which could never happen if the steps really ran together.

Investigate: change one letter, predict, run

Use Table 26.1. For each row, write your prediction in the middle column, then send the letters (battery on, wheels in the air) and fill in what happened.

TABLE 26.1 Predict, then run

SEND, IN ORDER	WHAT I PREDICT THE WHEELS DO	WHAT THEY DID
`F`		
`F`, then `B`		
`F`, then `Z`		
`F`, then `5`		

The last two rows usually surprise someone. `Z` stops the car, though nothing in the notes mentions `Z`. `5` changes the speed but the car keeps going forwards. Both answers are in the Act step, which is the next chapter.

TALK ABOUT IT

"The car is still going forwards and nobody is pressing anything. Which line in the sketch is keeping it going?"

PARENT: LISTEN FOR:

the letter is remembered in `btsignal`, and `loop()` keeps obeying it. A good answer points at `btsignal = Serial.read();` and says "it only changes when a new letter comes".



The whole sketch, `car.ino`, to download or read on a phone beside the book.

SCAN OR TYPE **troniction.com/go/car-sketch**



A short primer on Arduino sketches: `setup()`, `loop()`, pins and serial, with small examples.

SCAN OR TYPE **troniction.com/go/car-primer**

CHECK YOURSELF

1. You unplug the USB cable and plug it back in. Does the car remember it was driving forwards? Use `setup()` to explain your answer.
2. Is `loop()` doing two things at once when the car drives and beeps? What does it actually do?
3. Put these in the order the board does them after you press reset: (a) `Got F from USB` appears; (b) the L light comes on; (c) `Car ready` appears; (d) you send `F`; (e) the wheels turn.

Answers at the back of the book on page 253

/// CHAPTER

27

The letter dispatcher

Every letter the car receives goes down one long chain of questions: is it `F`? Is it `B`? Is it `L`? The first question that says yes decides what the car does. This chapter follows one letter down that chain, shows why every letter you didn't plan for stops the car, and solves the twitch you may have met in Chapter 26.

Predict

Write down your answers before you read on.

1. You send `F`, then `f` (small f). What do the wheels do after the `f`?
2. You send `F`, then `7`. What do the wheels do after the `7`?
3. The phone sends `F`. What appears in the Serial Monitor, and what goes back to the phone?

Run: watch the letters arrive

Bench set-up as in **"Set up the bench"**, page 174, battery on, wheels in the air. Send `F`, then `f`. Then `F`, then `7`.

⏱ TEST IT

YOU SHOULD SEE:	after `f` the wheels stop. After `7` they keep turning forwards, a little faster. Each letter adds one line to the Serial Monitor: `Got f from USB`, `Got 7 from USB`.
A INSTEAD:	no line at all when you send a letter.
MEANING:	the letter never reached the board. Either the Serial Monitor's speed is not 9600, or it is connected to a different port.
FIX:	check the speed menu says 9600 and the port is the board's (Chapter 6), then send again.

If a phone is connected, letters from the phone show up as `Got F from phone`, and the sketch sends `Car heard: F` back to the phone. The remote is designed to show that reply, so the phone knows the car received the letter.

Investigate: the line that reports

Every letter, from either place, goes to `report()` first:

car.ino LISTING 27.1

```
// ---- Report: show the letter, and answer the phone ------------------
void report(char letter, const char *from) {
  Serial.print("Got ");
  print_letter(Serial, letter);
  Serial.print(" from ");
  Serial.println(from);

  bluetooth.print("Car heard: ");
  print_letter(bluetooth, letter);
  bluetooth.println();
}
```

The first four lines write to the Serial Monitor: `Got`, the letter, `from`, and where it came from (`USB` or `phone`). The last three send `Car heard:` and the letter back out through the Bluetooth module. `print_letter()` prints the letter, except for the ones you can't see, which it prints by name: `(newline)`, `(return)`, `(space)`.

So the Serial Monitor is a window into the car. Chapter 24 used it as a probe: if `Got F from phone` appears, the radio, the module and its wires to pin 2 are all working, whatever the wheels do.

Investigate: the chain of questions

The Act step is a chain of `if` and `else if`. Figure 27.1 draws it.

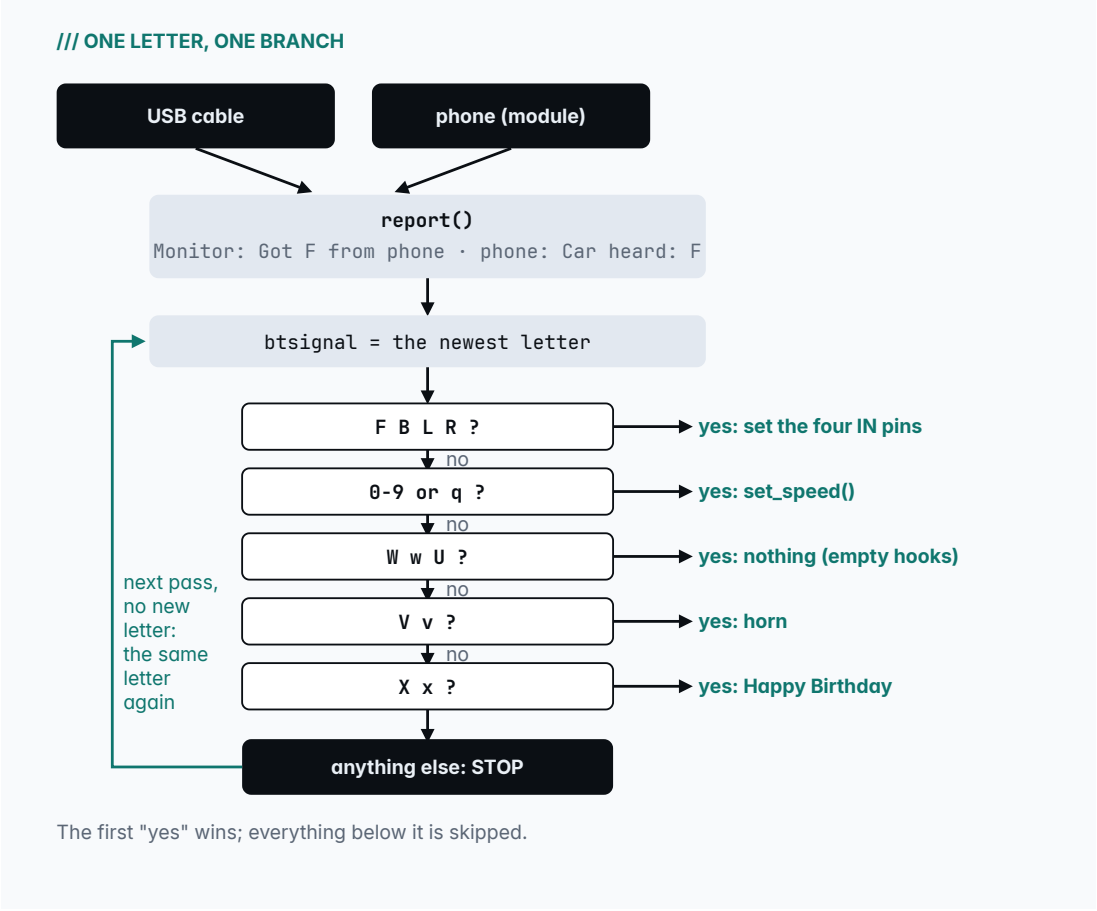


FIGURE 27.1 The dispatcher. A letter is reported, stored, then asked a chain of questions; the first yes wins, and a letter that matches nothing stops the car.

Here are the first two links and the last one:

car.ino LISTING 27.2

```
// ---- Act: the newest letter decides what the car does -------------
if (btsignal == 'F') {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}

else if (btsignal == 'B') {
  reversing = true;
  // ...the four pins for backwards (Chapter 28)
}

// ...L, R, the speeds, W, w, V, v, U, X, x...

else {
  reversing = false;
  digitalWrite(in1, HIGH);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, HIGH);
}
```

Three rules explain everything the chain does:

1. `==` **asks a question;** `=` **gives an order.** `btsignal == 'F'` asks "is the newest letter F?". The single `=` in `btsignal = Serial.read()` (Chapter 26) puts a letter into the box.
2. **The first yes wins.** Once one question matches, every `else` below it is skipped. Only one branch runs per pass.
3. `else` **with no question catches everything left over.** Any letter that matched nothing, from `S` to `Z` to a stray symbol, ends up here, and this branch stops the car.

Letters are exact. `'F'` and `'f'` are different letters to the board (they are different codes), so `f` goes all the way down the chain, matches nothing, and stops the car. That answers prediction 1.

The digits have their own branches, and they only call `set_speed()`. They don't touch the four direction pins, so the car keeps doing whatever it was doing, faster or slower. That answers prediction 2.

> TIP
> **Why "anything else stops" is a good design.** Radio links sometimes deliver a damaged letter. With this chain, most damaged letters match nothing and stop the car. A damaged letter can only do something new if it happens to become another real letter. Stopping is the safe thing to do when you don't understand an order.

The remote sends `S` when you let go (Chapter 11). There is no `S` branch: when you type `S` in the Serial Monitor, it stops the car because it is "anything else".

The twitch, solved

In Chapter 26 you may have seen `Got F from USB` and then `Got (newline) from USB`, with wheels that barely moved. Now you can explain it.

The Serial Monitor's line-ending menu adds invisible letters after whatever you type:

TABLE 27.1 What the line-ending menu adds

MENU SETTING	WHAT IS SENT WHEN YOU TYPE `F` AND PRESS ENTER
No line ending	`F`
Newline	`F`, then the newline letter
Carriage return	`F`, then the return letter
Both NL & CR	`F`, then return, then newline

The newline letter arrives about a thousandth of a second after the `F` (at 9600 baud each letter takes roughly a millisecond to travel). The board reads `F` and sets the wheels forwards. On the very next pass it reads the newline, which matches nothing in the chain, and the `else` stops the car. The motors are switched on for about a millisecond: a twitch at most, and often nothing you can see.

The car isn't broken and the sketch isn't wrong. The Serial Monitor sent a second letter, and the car obeyed it. The remote is designed to send only the letters, which is why the car drives normally from the phone with the same sketch.

Pick a line-ending setting and see the exact bytes it sends down the wire.

SCAN OR TYPE **troniction.com/go/car-line-ending**

Modify: one letter, two ways

No changes to the sketch yet; that starts in Chapter 28. Instead, try these from the Serial Monitor and check each against the chain in Figure 27.1 before you send it:

- Set the menu to **Both NL & CR** and send `F`. How many `Got` lines appear? (Predict first.)
- Back on **No line ending**, type `FB` on one line and press Enter. Two letters arrive in one go. Which one does the car end up obeying, and why?

TALK ABOUT IT

"You found that `f` stops the car. Is that a bug, or did the person who wrote this mean it?"

PARENT: LISTEN FOR:

either answer, with a reason. "It's on purpose: anything it doesn't know stops it, which is safe" is a good one. "It's a bug because I meant forwards" is also fair, and Chapter 32 shows how to add `f` as a letter of your own.

CHECK YOURSELF

1. The Serial Monitor is set to **Newline**. You send `F`. Predict exactly which two lines appear in the Serial Monitor and what the wheels do.
2. You want to add a letter `H`. You haven't written a branch for it yet. What does the car do today when it receives `H`, and which branch decides that?

3 Find the fault. A friend's Serial Monitor shows this, and their car won't drive from the laptop:

LISTING 27.3

```
Car ready
Got F from USB
Got (return) from USB
Got (newline) from USB
```

What setting do they change, and to what?

4 Put the chain in order so that `F` drives forwards, `B` drives backwards and every other letter stops the car. One piece doesn't belong: leave it out and say why.

1 `else { ...all four direction pins HIGH... };`

2 `if (btsignal == 'F') { ...forwards... };`

3 `else if (btsignal == 'B') { ...backwards... };`

4 `else if (btsignal == 'F') { ...forwards... }.`

Answers at the back of the book on page 253

/// CHAPTER

28

Making it move: the motion code

Four lines of the sketch move the car: one `digitalWrite()` for each of the motor driver's four direction pins. In Chapter 16 you met the H-bridge and its truth table. This chapter puts that table next to the sketch, so you can read any motion branch and say which way each side turns, and then write one yourself.

Predict

Here is the branch for `F`:

car.ino LISTING 28.1

```
if (btsignal == 'F') {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
```

`in1` and `in2` control one side of the car (call it side A, on the motor driver's ENA side). `in3` and `in4` control the other (side B). Predict: if you swapped the first two lines' values, so `in1` is HIGH and `in2` is LOW, what would `F` do?

Run: one side at a time

Bench set-up as in **"Set up the bench"**, page 174, battery on, wheels in the air. Send each of `F`, `B`, `L`, `R`, with `S` between them, and watch which way each side's wheels turn. Write it down; you'll compare it with the sketch in a moment.

⏱ TEST IT	
YOU SHOULD SEE:	`F` turns both sides forwards, `B` both backwards. `L` and `R` turn the two sides opposite ways. `S` stops all four wheels.
A INSTEAD:	`F` turns one side forwards and the other backwards.
MEANING:	one motor pair is wired the other way round. It is the same fault Chapter 9's wheels test catches.
FIX:	switch the battery off. On that side, swap the red and black wires of both motors at the motor driver's screw terminal (Chapter 9). Switch on and send `F` again: after any change to the motor wires, you go back to `F`. You don't need to change the sketch.

Investigate: the table in the sketch

Each side of the car has two direction pins. The motor driver reads them as a pair:

- one LOW and one HIGH: the motor turns one way;
- the other way round: it turns the other way;
- both the same: the motor is held still (the L298N's datasheet calls this a "fast motor stop", the brake).

Figure 28.1 shows what the sketch writes for every letter. Fill in Table 28.1 from your own run beside it.

/// LETTER → FOUR PINS → TWO SIDES

Letter	IN1	IN2	side A	IN3	IN4	side B	the car
F	L	H	forward	H	L	forward	forward
B	H	L	backward	L	H	backward	backward
L	H	L	backward	H	L	forward	spins to side A
R	L	H	forward	L	H	backward	spins to side B
else	H	H	brake	H	H	brake	stops

H = HIGH, L = LOW. ENA (pin 5) and ENB (pin 6) carry the speed for each side.
Which physical side is A depends on how your motors were wired (Chapter 9).

FIGURE 28.1 What each letter writes to the four direction pins, and what each side and the car do as a result.

TABLE 28.1 My car, letter by letter

LETTER	SIDE A DID	SIDE B DID	THE CAR
F			
B			
L			
R			
S			

Three things to notice in the figure:

1. **Forward isn't the same pattern on both sides.** Side A goes forwards on LOW, HIGH; side B on HIGH, LOW. That's how the motors were wired when this sketch was written in 2016: the two sides face opposite ways, and the wiring makes up for it. The sketch matches the wires, not a rule.
2. **Turning is driving the sides in opposite directions.** `L` runs side A backwards and side B forwards, so the car spins on the spot towards side A. There is no steering; Chapter 18 calls it skid steering.
3. **Stop is a brake, not "off".** The `else` branch sets all four pins HIGH, so both pairs are "the same" and both sides brake. The speed pins (ENA, ENB) stay

on. Some cars stop by setting all four LOW instead; that also brakes on the L298N, and either works.

Now answer your prediction. Swapping `in1` and `in2` in the `F` branch reverses side A only. On `F`, side A goes backwards and side B forwards: the car spins, exactly like `L`.

⊗ MYTH

"HIGH means forwards and LOW means backwards."

WRONG **This is wrong. No single pin has a direction.**

A motor turns because current flows through it, and current needs a difference between its two ends. The motor driver makes one end of the motor positive and the other negative depending on *which* of the two pins is HIGH. Only the pair means anything, and which way "forwards" is depends on how the motor's two wires were screwed in.

Modify: fill in `R`

Cover the `R` row of Figure 28.1 with your hand. You know:

- `L` is side A backwards, side B forwards;
- `R` should be the mirror: side A forwards, side B backwards.

Side A's forward pair is in the `F` listing at the start of this chapter. Side B's backward pair is in the figure's `B` row.

Fill in the four blanks:

car.ino (fill in) LISTING 28.2

```
  else if (btsignal == 'R') {
    digitalWrite(in1, ____);
    digitalWrite(in2, ____);
    digitalWrite(in3, ____);
    digitalWrite(in4, ____);
  }
```

Then open `car.ino`, find the real `R` branch, and check your answer against it.

TALK ABOUT IT

"You want `L` and `R` to swap over on your car. What would you change: the wires or the sketch? Why?"

PARENT: LISTEN FOR:

"the sketch: swap what the `L` and `R` branches write." Swapping one side's motor wires would break `F` too, because forwards would change on that side; after any change to the motor wires, you go back to `F` and check.

Four switches and a motor: flip IN1 and IN2 and watch which way the current flows.

SCAN OR TYPE troniction.com/go/car-hbridge

CHECK YOURSELF

1. In the `B` branch, `in3` is LOW and `in4` is HIGH. What does side B do, and why?
2. Predict: you swap the values on `in3` and `in4` in **every** branch. Which letters now do something different, and what?
3. Put the fix in order. `F` makes the car spin instead of going forwards: (a) send `F` again on the box; (b) switch the battery off; (c) see which side turns backwards; (d) swap the red and black wires of that side's motors at the screw terminal; (e) switch the battery on.

Answers at the back of the book on page 253

/// CHAPTER

29

The speed ladder

The digits `0` to `9` and the letter `q` set the car's speed. Each one sends a number between 0 and 255 to the two speed pins. This chapter shows where those numbers live, why the ladder starts at 150 instead of 0, and how to find the right bottom step for *your* car, then change it.

Predict

1. Sending `0` stops the car. True or false?
2. The car is going backwards and you send `9`. What happens?

Run

Bench set-up as in **"Set up the bench"**, page 174, battery on, wheels in the air. Send `F`, then `1`, then `9`, then `0`. Watch and listen to the wheels after each one.

⏱ TEST IT

YOU SHOULD SEE:	`1` slows the wheels, `9` speeds them up, and `0` gives the same slow speed as `1`. The car never stops on a digit.
A INSTEAD:	every digit gives the same speed.
MEANING:	the speed pins aren't doing their job. Either a jumper cap is still on the motor driver's ENA or ENB header, or a speed wire is not on pin 5 or 6. This is the fault the real car's sketch had from 2016 until 2026 (Chapter 22).
FIX:	battery off; take off the jumper caps and check ENA goes to pin 5 and ENB to pin 6 (Chapter 8). Test again.

So the answer to prediction 1 is **false**: `0` is the slowest speed, not stop. Chapter 31 has more surprises like this one.

Investigate: from letter to number

Each digit has a branch that calls `set_speed()` with one number:

car.ino LISTING 29.1

```
  else if (btsignal == '0') {
    set_speed(150);
  }

  else if (btsignal == '1') {
    set_speed(150);
  }

  else if (btsignal == '2') {
    set_speed(170);
  }
  // ...3 to 9: 180, 200, 210, 220, 230, 240, 250
  // ...and q: 255
```

and `set_speed()` sends the same number to both sides:

car.ino LISTING 29.2

```
// ---- Speed: the same number to both sides ---------------------------
void set_speed(int speed) {
  analogWrite(enA, speed);
  analogWrite(enB, speed);
}
```

`analogWrite()` doesn't set a voltage. It switches the pin on and off very fast (about a thousand times a second on pins 5 and 6) and the number says what share of the time it is on: 255 is on all the time, 128 about half, 0 never. Chapter 17 explains why a motor treats that as "less power". Figure 29.1 shows the whole ladder and the share of time each step is on.

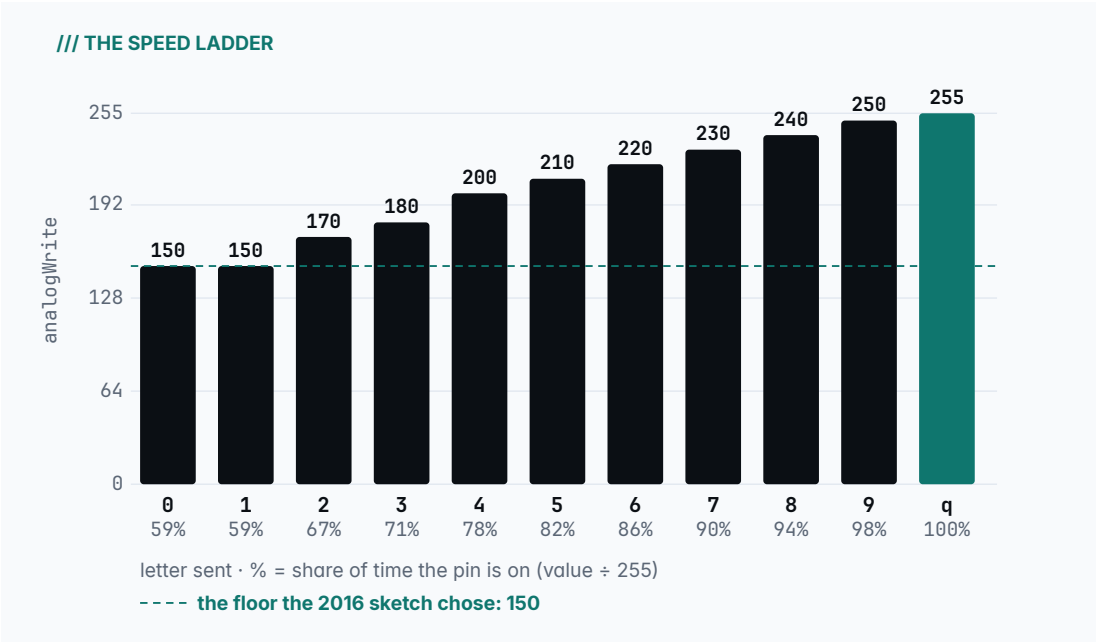


FIGURE 29.1 The speed ladder: the number each letter sends, and the share of time the speed pins are on.

The direction pins aren't touched in any of these branches. That's prediction 2: going backwards, `9` makes the car go backwards faster.

Why the bottom step is 150

A gear motor needs a certain push before it turns at all: enough to beat the friction in its gearbox and, on the floor, the weight of the car. Below that, the

motor hums and the wheels sit still. The person who wrote this sketch in 2016 put the bottom of the ladder at 150, so that speed `1` always moves the car.

That number was chosen for one car, with its motors, its battery and its floor. Yours may start moving lower, or need more. You can find out, by changing the sketch.

The safe-change routine

This is the first time in the book you change the sketch. Every change, however small, goes the same way: five steps, in this order. You'll use them again in Chapters 30, 31 and 32.

CHILD 1 **Keep the original.** File → Save As, with a new name for this change (`car-floor`, say).

CHILD 2 **Change one thing:** one branch, one number, or one line.

CHILD 3 **Verify.** Press the tick button, and if red errors appear, read only the first one.

▲ CHILD 4 **Switch the battery off, then upload.** Your parent checks the switch before you press Upload.

TOGETHER 5 **Test on the box,** wheels in the air: the letters your change affects, then `F` and `S`.

Why each step:

1. The original `car.ino` stays untouched, so you can always get back to a car that works. If you lose it, download it again (troniction.com/go/car-sketch).
2. If something goes wrong, you know which change did it.
3. Verify checks the sketch without uploading. The errors after the first one usually follow from it.
4. While the board restarts to take the new sketch, its pins let go of the motor driver for a moment. With the battery off, the wheels can't move while that happens.
5. `F` and `S` check you didn't break driving.

If the test fails and you can't see why, open the original `car.ino` and upload it. The car is back to how it was, and you can try again.

TEST IT

YOU SHOULD SEE:	after the upload, `Car ready` appears in the Serial Monitor, and `F` and `S` still drive and stop the wheels.
A INSTEAD:	the upload never finishes (“Uploading...” and then a timeout).
MEANING:	the laptop can’t reach the board. On this car the Bluetooth module isn’t the cause (Chapter 32 explains why), so check the port, the cable and the board choice (Chapter 6).
FIX:	choose the port again under Tools → Port, and try once more. If it still fails, Chapter 12’s “no port” rows apply.

Find your car’s floor

WARNING

This experiment runs on the floor. Clear a space with nothing to hit, and keep the USB cable long and slack, or do it from the phone. The car goes forwards until you send a stop letter; the battery switch stops it at any moment.

CHILD 6 — Follow the safe-change routine (“The safe-change routine”, page 196). Your one change: `set_speed(150);` becomes `set_speed(120);` in the branch for `'1'`.

▲ CHILD 7 — With the battery off, put the car on the floor. Then switch the battery on.

TOGETHER 8 — Send `1`, then `F`, and watch for two seconds before you send `S`.

TEST IT

YOU SHOULD SEE:	the car either crawls forwards or doesn't move.
A INSTEAD:	it doesn't move, but you can hear the motors hum.
MEANING:	120 is below your car's floor. The motors get power, but not enough to start.
FIX:	it isn't a fault. Try a bigger number (130, then 140) until the car just moves. That number is your car's floor.

Repeat with different numbers until you find the lowest one that still moves the car on the floor. Write it down. Then do the same on a rug: is the floor higher or lower?

> TIP
> **A weak battery raises the floor.** If `1` moved the car yesterday and doesn't today, and nothing was touched, charge the cells before anything else (Chapter 19).

Modify: your own ladder

You know your floor, and you know the top is 255. Spread the nine steps between them. One way:

TABLE 29.1 A ladder with even steps of 15, for a floor of 120

DIGIT	1	2	3	4	5	6	7	8	9
If your floor is 120	120	135	150	165	180	195	210	225	240

Change the nine numbers, upload, and drive. Does the car feel smoother between steps than before? Keep `q` at 255 so there's always a full-power letter.

> TALK ABOUT IT
> **"Speed `1` crawls and `9` runs. What is the pin doing differently?"**
>
> PARENT: LISTEN FOR:
> it is on for a bigger share of the time at `9`; the pin is never "half on". If the answer is "more volts", ask what the pin can do besides on and off.

A slider for the number you send, the on–off pattern it makes, and the average the motor feels.

SCAN OR TYPE **troniction.com/go/car-pwm**

CHECK YOURSELF

1. Sending `0` doesn't stop the car. Which branch would you change to make `0` stop it, and what number would you give `set_speed()`?
2. You change the `'1'` branch to `set_speed(40)`. What do you expect on the floor, and why?
3. Every speed letter gives the same speed. Name two causes, and say which one you would check first.
4. Put the floor-finding steps in order. One step doesn't belong: leave it out and say why. (a) send `1`, then `F`, and watch; (b) Save As a new name; (c) change `150` to `120` in the `'1'` branch; (d) switch the battery off and upload; (e) change all ten speed numbers at once, to save time; (f) Verify.

Answers at the back of the book on page 253

/// CHAPTER

30

Sound, and why the car stops listening while it sings

The car makes three sounds: a horn (`V`), a beep while it reverses (`B`), and *Happy Birthday* (`X`). Two of them leave the car free to drive. The third takes over the whole board for sixteen seconds, and while it plays, the car can't hear you. This chapter shows why, using the two ways the sketch waits: one that blocks everything and one that doesn't. Then you rewrite the song so it doesn't.

Predict

1. You send `V` once. How long does the horn sound?
2. The car is driving forwards. You send `X`, then two seconds later `S`. When does the car stop?

> ⚠ WARNING
>
> **Do this one on the box, not the floor.** Prediction 2 is about a car that can't be stopped from the keyboard for several seconds. Keep the wheels in the air. The battery switch stops the wheels at any moment.

Run

Bench set-up as in **"Set up the bench"**, page 174, battery on, wheels in the air.

1. Send `V`. Count how long the horn lasts. Then send `S`.

2. Send `B`. Listen. Then send `S`.
3. Send `F`. Then send `X`. Two seconds into the song, send `S`. Watch the wheels and the Serial Monitor.

⏱ **TEST IT**

YOU SHOULD SEE:	the horn sounds until you send another letter. The reversing beep goes on and off quickly while the car reverses. After `X`, the wheels **keep turning** through the song, the `Got S from USB` line appears only when the song ends, and only then do the wheels stop.
A INSTEAD:	no sound at all.
MEANING:	the buzzer module's signal pin (S) isn't on pin 4, or its + and – pins aren't getting 5V and GND.
FIX:	battery off; check the buzzer's three wires against the wiring figure in Part 1, then test again.

Prediction 2 surprises nearly everyone. The `S` was sent at two seconds, but the car didn't read it until the song finished, about sixteen seconds after it started.

Investigate: two ways to wait

A sound needs time: a note has to last a while before the next one. The sketch waits in two different ways.

The reversing beep waits with `millis()`. `millis()` is a clock: the number of milliseconds since the board started. The Beep step looks at the clock on every pass:

car.ino LISTING 30.1

```
// ---- Beep while reversing, every 100 ms --------------------------
unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= interval) {
  previousMillis = currentMillis;

  if (reversing == true) {
    tone(buzzerpin, 5000, interval);
  }
}
```

Read it as: "Has it been at least 100 ms since the last time? If not, carry on with `loop()`. If so, note the time and, if reversing, start a 5000 Hz beep lasting 100 ms." `tone()` starts the sound and returns at once; the buzzer keeps going by itself. So `loop()` never stops to wait. It checks the clock, finds it isn't time yet, and goes round again to listen for letters.

The song waits with `delay()`. Here is the inside of `play_happy_birthday()`:

car.ino LISTING 30.2

```
for (int i = 0; i < sizeMelody; i++) {

  noteDuration = 1000 / timeHappy[i];

  tone(buzzerpin, melodyHappy[i], noteDuration);

  pauseBetweenNotes = noteDuration * 1.30;

  delay(pauseBetweenNotes);

}
```

`delay()` means "do nothing at all for this many milliseconds". The board doesn't go back to the loop, doesn't look for letters, doesn't touch the wheels. It sits on this line. And the `for` loop does that 26 times, once per note. Add up the pauses and one play-through takes 16.25 seconds.

Figure 30.1 puts the two side by side.

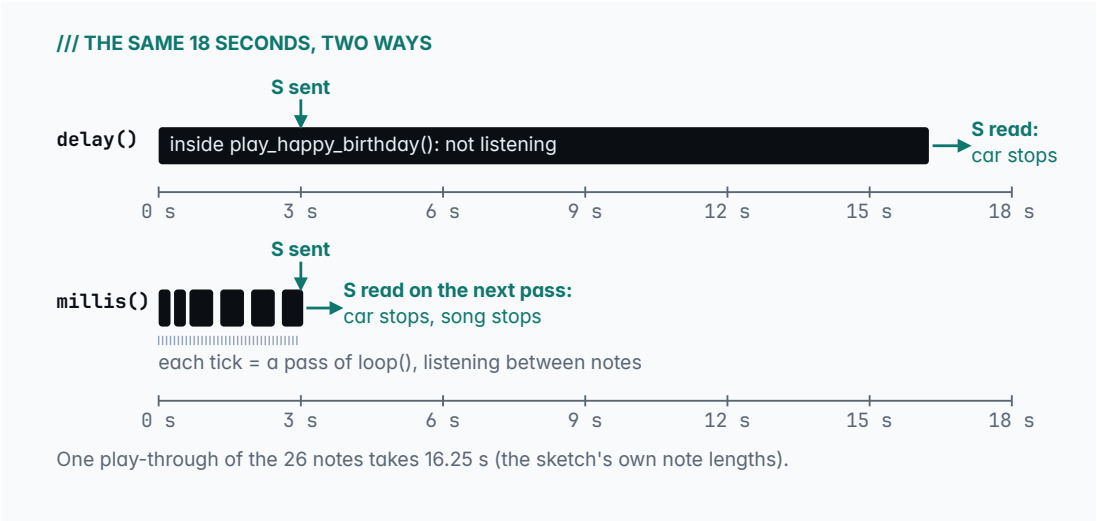


FIGURE 30.1 While `delay()` plays the song, the board can't read the `S`. With `millis()`, it reads it on the next pass.

Now the three surprises make sense:

- **The wheels keep turning during the song.** The `X` branch never touches the direction pins, so they stay as `F` left them.
- **The `S` waits.** Letters that arrive during the song wait in a small queue (64 letters) until the board next reaches the Listen step.
- **The song plays again if nobody sends anything.** `X` is sticky, like every letter. When the song ends and no new letter is waiting, the next pass obeys `X` again.

The horn is sticky too: every pass of `loop()` calls `tone()` again for another second, so the horn never runs out while `V` is the newest letter. That answers prediction 1.

⊗ MYTH

> "The board was busy, so it ignored my `S`."

WRONG **This is wrong: the `S` wasn't ignored, it was kept.**

The Bluetooth module and the USB chip pass every letter to the board, and the board stores it in a queue even while `delay()` is running. Nothing reads the queue until `loop()` comes round again. The `S` arrived on time and waited in line, which is why the car does stop in the end, without you sending it twice.

Modify: one note at a time

To keep the car listening, the song has to work like the beep: play **one** note, go back to the loop, and come back when it is time for the next note. That needs three things remembered between passes: which note is next, when the last one started, and how long it needs.

This is a completion problem. The new function is below with three blanks. Use the beep in Listing 30.1 as your pattern.

car-yours.ino (fill in) LISTING 30.3

```
int nextNote = 0;                  // which note plays next
unsigned long noteStarted = 0;     // when the last note started
unsigned long pauseBetweenNotes = 0; // how long that note needs

void play_happy_birthday_step() {

  if (millis() - ________ < pauseBetweenNotes) {
    return;                        // not time yet: go back and listen
  }

  int noteDuration = 1000 / timeHappy[nextNote];
  tone(buzzerpin, melodyHappy[nextNote], noteDuration);

  noteStarted = ________;
  pauseBetweenNotes = noteDuration * 1.30;

  nextNote = nextNote + 1;
  if (nextNote == sizeMelody) {
    nextNote = ________;           // start the tune again
  }
}
```

To use it, three more changes:

1. move the two lists `melodyHappy` and `timeHappy`, and `sizeMelody`, out of the old function to the top level, so the new function can see them;
2. in the `X` and `x` branches, call `play_happy_birthday_step();` instead of `play_happy_birthday();`;
3. delete the old function.

The finished version is `car-yours.ino` in the sketch download. Compare your blanks with it, then follow the safe-change routine ("The safe-change routine", page 196) to upload it, and run the same test as before: `F`, `X`, and `S` two seconds in.

TEST IT

YOU SHOULD SEE:	the song plays as before, and when you send `S`, the wheels stop and the song stops within a moment (the note that is playing finishes).
A INSTEAD:	the song plays every note at once as a blur, or stops after the first note.
MEANING:	the first blank is wrong. If it isn't `noteStarted`, the time check never waits for the right moment.
FIX:	compare your line with the beep's `currentMillis - previousMillis` pattern, change it, and upload again with the routine.

TIP

`return` **inside a function** means "stop this function here and go back to whoever called it". Here, that's `loop()`, which goes on to listen for letters.

TALK ABOUT IT

"The song and the wheels seem to happen at the same time now. Are they?"

PARENT: LISTEN FOR:
no; the board still does one thing at a time. It plays a note, then goes round the loop many times checking letters and the clock, then plays the next note. Chapter 26's myth again.

CHECK YOURSELF

1 The horn (`V`) never blocks `loop()`, but it never stops by itself either. Why doesn't it stop after one second, when `tone()` was told 1000 ms?

2 Predict: in the original sketch, you send `X`, then type `F` five times in the Serial Monitor during the song. What happens when the song ends?

3 In your new version, you send `X`, then `S`, then `X` again. Does the song start from the beginning? Look at `nextNote` to decide.

4 Put these lines of `play_happy_birthday_step()` in order. One line doesn't belong: leave it out and say why. (a) `noteStarted = millis();`; (b) `if (millis() - noteStarted < pauseBetweenNotes) { return; }`; (c) `nextNote = nextNote + 1;`; (d) `tone(buzzerpin, melodyHappy[nextNote], noteDuration);`; (e) `delay(pauseBetweenNotes);`.

Answers at the back of the book on page 254

/// CHAPTER

31

Find it: the bug since 2016

There is a bug in your car's sketch. It has been there for ten years, since the sketch was written in 2016. It was in the car that drove in 2021, and I left it in on purpose so you could find it. Nobody invented it for this book. This chapter gives you the part of the sketch it hides in, a way to catch it in the act, and then the fix.

Find it by reading

The bug is somewhere in these lines, the end of the Act step's chain. Read them slowly. One branch can never run, however many letters you send. Which one?

car.ino LISTING 31.1

```
  else if (btsignal == 'W') {
    //frontlight = true;
  }

  else if (btsignal == 'w') {
    //frontlight = false;
  }

  else if (btsignal == 'V') {
    tone(buzzerpin, 2000, 1000);
  }

  else if (btsignal == 'v') {
    tone(buzzerpin, 2000, 1000);
  }

  else if (btsignal == 'U') {
    //backlight = true;
  }

  else if (btsignal == 'U') {
    //backlight = false;
  }
```

TIP

A hint, if you need one. The comments say what each pair was *meant* to do: a light on, then the same light off. Compare each pair's two letters.

Write down your answer before you read on.

Catch it in the act

Found it? The two `U` branches. The second one asks the same question as the first. The first always answers yes first, so the second can never run. It was meant to be `'u'`, small u: back light off, to match `w` for front light off.

A branch that never runs sounds harmless, especially when the light was never wired. But a bug usually has a second effect, somewhere you aren't looking. Predict before you run this:

> The car is driving forwards. It receives `u`. What does the car do?

Bench set-up as in **“Set up the bench”**, page 174, battery on, wheels in the air. Send `F`, then `U`. Then `F`, then `u`.

⏱ TEST IT

YOU SHOULD SEE:	after `U`, the wheels keep turning (an empty hook does nothing). After `u`, the wheels **stop**.
A INSTEAD:	the wheels stop after `U` too.
MEANING:	the line-ending menu is adding a letter after yours (Chapter 27).
FIX:	set it to **No line ending** and repeat.

Here is why. Nothing in the chain asks about `'u'`, so `u` goes all the way down to the final `else`, and the `else` stops the car. The typo didn’t just break a light that doesn’t exist. It turned the back-light-off button into a stop button. On a car with lights, switching the back light off would have stopped the car dead every time.

Fix it

The fix is one character. Follow the safe-change routine (**“The safe-change routine”**, page 196):

CHILD 1 **Save a copy first:** File → Save As, name it `car-fixed`. Keep the original as it was.

CHILD 2 In the second `U` branch, change `'U'` to `'u'`. Change nothing else.

▲ CHILD 3 Verify, switch the battery off, and upload.

TOGETHER 4 Battery on, wheels in the air. Send `F`, then `u`.

⏱ TEST IT

YOU SHOULD SEE:	the wheels keep turning after `u`. `Got u from USB` appears in the Serial Monitor.
A INSTEAD:	the upload stops with red errors. The first one says `missing terminating ' character`, and the ones after it say things like `'report' was not declared in this scope.`
MEANING:	a quote mark was deleted along with the letter. Each letter in a branch needs a quote on both sides: `'u'`. Only the **first** error matters; the others are the compiler getting confused by it.
FIX:	put the missing quote back, upload again.

One more, harder

There's a second thing in this sketch that behaves oddly. It isn't a typo, and you'll only notice it if you skip the stop.

Send `B` (listen to the reversing beep), then send `F` straight away, with no `S` in between. What do you hear? Find the line in the sketch that switches the beep on, and the only line that switches it off. Which branches would need `reversing = false;` to fix it?

The answer, and the fixed version, are in `car-yours.ino` in the sketch download.

TALK ABOUT IT

"This bug was in the car for ten years and nobody noticed. Why not?"

PARENT: LISTEN FOR:

the light was never built, so nobody had a reason to switch it off. A bug you never trigger stays hidden. Chapter 22's speed bug hid the same way, from 2016 until 2026, behind a car that still drove.

CHECK YOURSELF

1. Which branch can never run, and why?
2. Before the fix, what does the car do when it receives `u` while driving? Which branch decides that?
3. Put the fix in order: (a) send `F`, then `u`, and watch the wheels; (b) change the second `'U'` to `'u'`; (c) Save As a new name; (d) upload; (e) switch the battery off.

Answers at the back of the book on page 254

/// CHAPTER

32

Change it safely, then make it yours

You've used the safe-change routine three times now: for the floor (Chapter 29), the song (Chapter 30) and the bug (Chapter 31). This chapter makes it a habit, then you design and add a letter of your own. Last, it explains why this car never gets stuck on upload, when many Bluetooth cars do.

The routine, from memory

Before you look at the card below, cover it and say the five steps out loud, in order. Your parent holds the book and checks. Then uncover it.

TABLE 32.1 The safe-change routine (Chapter 29)

	THE STEP	WHY
1	Keep the original: File → Save As	You can always get back to a car that works
2	Change one thing	If it goes wrong, you know which change did it
3	Verify, and read only the first error	It checks the sketch without uploading
4	Switch the battery off, then upload	The wheels can't move while the board restarts
5	Test on the box: your letters, then `F` and `S`	Check you didn't break driving

Which one did you forget? That's the one to say first next time. If a test fails and you can't see why, open the original `car.ino` and upload it: the car is back to how it was.

Design your own letter

A new command is a new branch in the chain. Here is a worked one: `T` for turbo, forwards at full speed.

car-yours.ino LISTING 32.1

```
  else if (btsignal == 'T') {          // my new letter: turbo
    set_speed(255);
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
  }

  else {
    reversing = false;
    // ...the stop branch, unchanged
```

Four rules, each one something you have already seen:

1. **Pick a letter nobody uses.** Check the letters list in the notes at the top of the sketch. Letters are exact: `T` and `t` are different (Chapter 27).
2. **Put it before the final `else`.** The `else` must be the last link in the chain: a branch after it won't even compile (`'else' without a previous 'if'`). And use a letter no other branch asks about. If two branches ask about the same letter, only the first one ever runs: the trap of the second `U` (Chapter 31).
3. **Remember it repeats.** Your branch runs on *every* pass while your letter is the newest one (Chapter 26). That's right for driving and speed. For a sound or a light that should happen once, it isn't: a `delay()` in your branch would freeze the car again and again (Chapter 30).
4. **Remember what it leaves behind.** Turbo sets the speed to 255, and the speed stays 255 after it. Send a digit to come back down. Decide whether you want that.

Now design yours. Fill in this card before you change the sketch:

TABLE 32.2 My new command

My letter
What the car does
Which pins or helpers it uses (`in1`–`in4`, `set_speed()`, `tone()`)
What it leaves behind (speed? direction? a sound?)
The test: what I send, what I expect

Some ideas that use only what the car already has: a slow crawl forwards (`set_speed()` and the `F` pattern in one branch); spin on the spot at full speed; a "find me" horn at a different pitch; `f` as a second forwards letter. Then follow the safe-change routine (Table 32.1) to add it.

> TIP
> **Making a letter happen once.** End the branch by changing `btsignal` to a letter that does nothing, such as `'W'`. On the next pass the chain finds `W`, does nothing, and the car carries on with whatever it was doing. For example, a single short honk: `else if (btsignal == 'H') { tone(buzzerpin, 1000, 200); btsignal = 'W'; }`

Why this car has no upload trap

Uploading uses two of the board's pins, 0 and 1: they carry the new sketch from the USB cable. On many Bluetooth cars, including this one in 2021, the module was wired to those same two pins. Then the module and the laptop both talk on one wire during the upload, the board hears noise, and the upload hangs. Those cars need the module's wires pulled out before every upload, and plugged back in after.

In this book the module is on pins 2 and 3 instead, with the `SoftwareSerial` library making those two pins work as a second serial port. Pins 0 and 1 belong to the USB cable alone, so you can upload with the module connected, and the Serial Monitor can show you what the phone sends at the same time. That second advantage is the one you have used in every chapter of this part.

The cost is small: `SoftwareSerial` can only do one thing at a time on its pins, so a letter arriving at the very moment the car is sending `Car heard` back can be missed. That is why the remote sends each letter once instead of a stream,

and sends `S` again if the car doesn't answer `Car heard: S`. If any other press is ever missed, press again.



The upload that hangs with Bluetooth connected: the error messages, why pins 0 and 1, and the fix.

SCAN OR TYPE **troniction.com/go/car-upload-trap**



`car.ino` and `car-yours.ino` (every change from this part), to download.

SCAN OR TYPE **troniction.com/go/car-sketch**

TALK ABOUT IT

"Show me your new letter working. What did you expect, what happened, and what would you change next?"

PARENT: LISTEN FOR:

a prediction made before the test, and a difference between expected and actual that they can explain. Getting a surprise and explaining it is the whole of this part.

Where next: the Troniction Academy's Foundations stage teaches Arduino programming from the beginning, through to a non-blocking sketch of your own design, like the song you rewrote in Chapter 30.

CHECK YOURSELF

1. Why can this car take a new sketch with the Bluetooth module still connected, when some cars can't?
2. You add `else if (btsignal == 'T')` after the final `else`. What does `T` do, and why?
3. Design a letter `C` that makes the car crawl forwards at speed 150. Write the branch, and say what the car's speed will be after you send `C` and then `F`.

Answers at the back of the book on page 254

5

/// PART 5

For parents, clubs and teachers

This part is for the adults. Nothing in it is a step for the child.

Read Chapter 33 before you start, whether you are building one car at the kitchen table or twelve in a classroom: it says what a build like this can honestly be expected to teach, and how you would know. Chapter 34 runs the build with a group. Chapter 35 is the check you do a week after the first drive, which tells you more than anything you see on the day. Chapter 36 is where a keen child can go next.

/// CHAPTER

33

What this build teaches, honestly

Most books like this one end with a promise about your child: future skills, an engineer's mind, a head start. This one doesn't, because the evidence doesn't support it, and you deserve to know what it does support before you spend your weekends on it.

What the evidence says a build like this does

Children and teachers say they enjoy it. When teachers and pupils in England were interviewed about a small programmable board (the BBC micro:bit), they talked about how easy it was to use, how creative it felt, the fact that it was a real object in their hands, and what they had learned about programming (Sentance and colleagues, 2017 `[study]`). That is a study of what children *said*: it shows interest, not measured learning.

It has not been shown to teach programming better than a screen does. Physical builds make programming more engaging and more concrete. The research does not show that they make it easier to learn. So this book never claims that building the car will make your child a better programmer, a better student or "smarter". "Learning to code makes you smarter" is on the list of claims the research does not support: the evidence does not show that skill in one subject spreads into general cleverness.

What it can teach, and what you can check, is narrower and more useful. The book is built to teach three things:

1. **A picture of the whole car.** The four boxes (the phone, the Bluetooth module, the board, the wheels), with the battery feeding the motor driver

and the board. A child who has this picture can say where a fault lives.

2. **A way to find a fault.** The loop from Chapter 3: look, pick a box, change one thing, test again, say what you found. In one small classroom study, about ten minutes of teaching a loop like this one helped 15- and 16-year-olds fix more bugs (Michaeli and Romeike, 2019 `[study]`: two classes, not randomised, measured soon after).
3. **A few real mechanisms.** Why only some pins can control speed, how four switches reverse a motor, why the module's wires cross, why the board needs the battery too.

None of the three is "how to follow instructions". A child who can follow the steps has built a car. A child who can find a fault a week later has learned something.

Why the book tests the circuit first

In a lab study, 20 adults who could already program built a small Arduino circuit on a breadboard. Only 6 of the 20 finished. Wiring faults ended more attempts than anything else, yet the adults blamed their program twice as often as their circuit (66% against 33%) (Booth and colleagues, 2016 `[study]`). These were experienced programmers, not children, and it was one task. But it is the closest study I have found to what happens at your table, and it is why every stage in Part 1 ends with a test of the wiring before anything else, and why Chapter 12 never starts with "it's the code".

Why the roles are what they are

The roles in Part 1 (the child builds, you read and check) come from research on parents helping with homework, not with robot cars. Treat it as a strong hint, not a proof.

- **How you help matters more than how much.** Help that leaves the child in charge, focuses on the task, and stays calm is linked to better outcomes. Help that takes over, or makes the child feel judged, is not (Pomerantz and colleagues, 2007 `[review]`).
- **At this age, more homework help goes with lower marks** (Patall and colleagues, 2008 `[meta]`). That finding is a correlation: children who are

struggling may simply get more help. It is still a reason not to take the screwdriver.

- **A parent's own nerves can pass to the child through the help.** In one study, maths-anxious parents who helped often with homework had children who learned less maths over the year (Maloney and colleagues, 2015 `[study]`). That was maths, in the first two years of school. For electronics it is a hypothesis. The book's answer is the line "I don't know either. Let's check the picture.": it is honest, and it keeps you and your child on the same side of the problem.
- **Explain the *why* to every child.** In one museum study, parents were three times as likely to explain science to their sons as to their daughters, though they talked to both just as much (Crowley and colleagues, 2001 `[study]`). That was one museum, with young children, in 2001. It costs nothing to act on: every talk prompt in this book is written for any child, and the *why* is in the main text, not in a box for the keen ones.

What nobody has tested yet

A parent and a 12-year-old building from a written guide is almost unstudied. This book is designed from the nearest evidence, and the honest page at the front ("How this book was tested") lists what was built, what was checked by computer and datasheet, and what wasn't. In particular:

- **The time it takes** and **the number of sessions** have not been measured with a family. The book prints its stopping points instead of a promise.
- **The age band** (12 and up; 9 to 11 if you read and check every step) has not been tested with a 10-year-old.
- **Running it with a group** (Chapter 34) has not been tried in a club.

If you build it, you are part of the test. What you find is worth more to the next reader than anything in this chapter: tell me at troniction.com/go/car.

How you will know

Not from how the day felt. "They loved it" and "it was so clear" are pleasant and tell you very little: how easy something feels while you learn it is a poor guide to what stays (Soderstrom and Bjork, 2015 `[review]`). The test that tells

you something is a week later, without the book: can your child say which box a fault is in, and fix a fault you planted? Chapter 35 is that test.

CHECK YOURSELF

1. Before you start, write one sentence: what would you accept, a week after the first drive, as evidence that your child learned something? Keep it, and compare it with Chapter 35.
2. Your child's friend's parent says the build "will make him a better programmer". What can you honestly say instead?

Answers at the back of the book on page 254

/// CHAPTER

34

Running it with 6 to 12 children

> ⓘ NOTE
> **Untested in a group.** This chapter is built from Part 1 and from research on working in pairs. No club has run this build yet. If you are the first, tell me how it went at troniction.com/go/car.

With one child, you are the checker. With twelve, you can't be. The design that scales is Part 1's, moved one level down: **two children per car, one builds and one checks**, and the adults keep the two jobs no child does, the cells and the software install.

Pairs, roles and the swap

Give each pair one car, one tray of parts and one printed copy of the stage they are on.

TABLE 34.1 The roles in a club: Part 1's roles, with a second child as the checker

ROLE	WHO	DOES	NEVER
Builder	One child	Every wire, screw and push-fit; presses Upload; the holder's switch; types the test letters	Cells, charger, software install
Checker	The other child	Reads each step aloud; compares the wire with the picture; ticks the checklist; asks the talk prompt	Takes the part out of the builder's hands
Adult	You	The prep, the cells, the install; walks the room; answers "which box?" with a question	Fixes the car

Swap the roles at every stage test, so both children build and both check. Pairs did better than students working alone in university programming cours-

es (Umapathy and Ritzhaupt, 2017 [meta]); that is adults writing code, so treat it as a reason to try pairs, not a promise. The main problems were scheduling and partners who didn't get on (Hanks and colleagues, 2011 [review]), so choose the pairs yourself. **Three in a group** works if you must: the third holds the tray and reads ahead, and all three rotate.

Before the day

Several stalls happen before anyone touches a wire: a locked laptop, a charge-only cable, cells never charged. Chapter 2's prep list becomes this, once per car:

1. **Every laptop:** install the Arduino IDE and see the port appear with a bare board (Chapter 6), and download the sketch (troniction.com/go/car-sketch). A locked school laptop may refuse the install: ask whoever manages it a week before.
2. **Every phone:** Chrome on Android or Bluefy on an iPhone for the HM-10; Android only for the HC-06. Buy one kind of module for every car.
3. **Charge every cell** (Chapter 5's rules) and label each pair with its car's number.
4. **Pack one numbered tray per car** and run Chapter 4's find-it list on each.
5. **Print** the step cards, the pre-power checklist, the role card and the loop card for each pair: they are all in the club pack at the end of this chapter.
6. **Spares:** jumper wires, the two resistors, one known-good USB data cable and, if you can, a spare board and a spare Bluetooth module: a failed one ends a pair's session.

The session plan

A club session ends at a stopping point (after the Stage A test, the wheels test, test C1 or the drive) or before Stage A starts, never in the middle of a stage.

TABLE 34.2 A four-session plan. Nobody has timed it with a group yet: time your first run and write it in

SESSION	READ AND BUILD	ENDS WHEN	ADULT JOBS	YOUR TIME
1	Chapters 3–5 together; unbox and find; Stage A (Chapter 6)	`Car ready` in the Serial Monitor	Walk the room at the upload	____ min
2	Stage B (Chapters 7–9)	The wheels test passes and the board stays on without the USB	Cells in, at each table, after the checklist	____ min
3	Stage C (Chapter 10)	Test C1; C2 if there is time	All power off before the module goes on	____ min
4	Stage D (Chapter 11)	The car drives off the box	The runway; cells out and counted	____ min

Time each session on your first run, from the first page read to the test that passed: it is your best guide to the next group, and the next edition wants it. A pair that finishes early does the "Try this" at the end of a Part 2 chapter; a pair a stage behind starts the next session at its own stopping point.

Batteries in a group

The cells are the most dangerous thing in the box. Chapter 5's rules hold; a group adds four:

1. **The cells live in one box, held by an adult,** and come out at a table only for the power step in Chapter 9, after the pair has read the pre-power checklist aloud. With more than about four cars, a second adult at the cell station keeps pairs from waiting (untested).
2. **"Switches off" is a room-wide call.** Before any wiring change the builder switches off and the checker says so out loud. "All power off" means the USB unplugged **and** the switch off.
3. **At the end of every session the cells come out and are counted back in.** A dented, swollen or oddly warm cell goes aside and is recycled.
4. **No charging during the session,** and never unattended afterwards.

The room

Figure 34.1 shows one way to lay out the four things that matter.

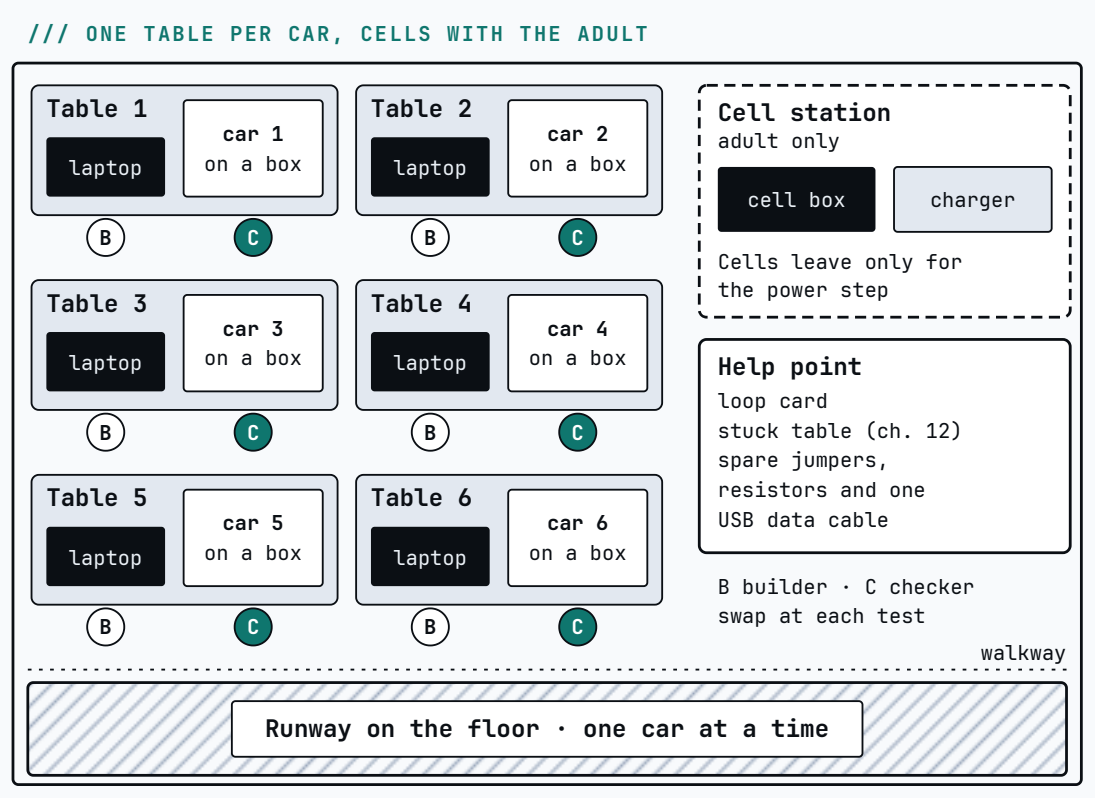


FIGURE 34.1 A room for six cars. Pairs work at their own tables with the car on a box. The cells stay at the adult's station; the test runway is a marked floor area away from the tables.

1. **Each car stays on its box** until Stage D, so a wheels test can't drive it off the table.
2. **The cells and charger have their own station,** with an adult, on a hard surface.
3. **The runway is on the floor, away from the tables.** One car at a time.
4. **A help point** with the loop card and Chapter 12's table. A stuck pair goes there first.

Many identical modules. Every module of one kind shows up under the same name, so connect one car at a time and press the horn: the car that beeps is

yours. Then keep each phone with its car. (Renaming the modules, which *Make It Connect* covers, is an adult job before the day.)

When a pair is stuck

The 10-minute rule becomes **"run the loop, then raise the card"**. When you get there, ask before you touch anything: "What exactly happens? Which stage test still passes? What did you change last?" Then point at the picture, not the wire. Keep a tally of where pairs got stuck, by stage: it shows the step the book explains badly, which is exactly what I want to hear about.

Photos: hands and cars, not faces, and your club's consent rules. When in doubt, film from above.

The club pack: the session plan, role cards, every chapter's questions by part, and the one-week check, ready to print.

SCAN OR TYPE **troniction.com/go/car-club**

CHECK YOURSELF

1. Two pairs reach the power step at the same time, and you are the only adult in the room. What do you do?
2. A pair has been stuck for ten minutes in Stage C and the builder is pulling wires out to start again. What do you say first?
3. You are planning the club's first session. It is 45 minutes long. Where should it end?

Answers at the back of the book on page 254

/// CHAPTER

35

The one-week check

On the day of the first drive, everything looks like learning. The car moves, your child explains the H-bridge, the room is loud. A week later is when you find out what stayed. How easy something feels while you learn it is a poor guide to what you keep; what you can still do later, without help, is the honest measure (Chapter 33).

This check takes about twenty minutes, a week after the first drive, **without the book open**. It has two parts: one fault you plant in the car for your child to find and fix, and three questions your child answers out loud. Then you score both on the rubric in Table 35.3, and the rubric tells you what to do next.

> ⓘ NOTE
> **This is a check, not a test.** Tell your child that: "I want to see whether the book did its job." If something isn't there, that is information about the book and about what to practise, not about your child.

Part 1: the planted fault

> ⚠ WARNING
> **You plant the fault with all power off: the USB unplugged and the holder's switch off.** The cells stay where they are; you don't need to touch them. Any wire you take off a pin goes where its bare end touches nothing: into an empty row of the breadboard that no other wire uses.

Choose **one** fault. Plant it while your child is out of the room.

TABLE 35.1 The planted faults. Each lives in one box, and each is undone by putting back what you moved

FAULT	WHAT YOU DO, POWER OFF	WHAT YOUR CHILD WILL SEE
A. One side reversed	Swap the two motor wires of **one** side in their screw terminals (OUT1 and OUT2, or OUT3 and OUT4)	Forward makes the car spin on the spot
B. A speed cap back on	Pull the board's speed wire off ENA and push its end into an empty breadboard row. Then push a jumper cap onto ENA	That side always runs at full speed; the speed letters change only the other side
C. A data wire off	Pull the Bluetooth module's TXD wire off pin 2 on the board and push its end into an empty breadboard row	The phone connects, but the car ignores every button

Then hand over the car: "Something is wrong with the car. Find it and fix it." Start a timer.

The rules for you:

- **Stay quiet.** Don't hint, don't point, don't look at the right place.
- **Your child may ask you to read any page of the book aloud**, but not to open it at the right page. Finding the right page is part of the task.
- **Safety still holds:** the switch goes off before any wire is touched, and the cells stay yours.
- **Stop at 15 minutes.** If the fault isn't fixed by then, say "Let's finish this one together with the book," and do. Fifteen minutes is the target the book was designed to; it is not a norm, and missing it tells you what to practise.

Write down, as it happens: what your child looked at first; whether they named a box; whether they ran a stage test before changing anything; whether they changed one thing at a time; and how long the fix took.

Part 2: three questions, out loud

These six questions are new: none of them is in the chapters. Each one asks your child to use the four boxes on a car or a situation they haven't seen. **Always ask question 1. Then ask the two that go with the fault you planted**, so no question gives the fault away:

TABLE 35.2 Which questions to ask. Each set mixes boxes and keeps away from the fault you planted

YOU PLANTED	ASK
A (one side reversed)	1, 2 and 6
B (a speed cap)	1, 3 and 5
C (a data wire off)	1, 3 and 4

Read the question; don't help; write down what your child says, in their words. The answers are at the back of the book, under Chapter 35, and in the club pack.

1. **Fine on the box, not on the floor.** A car's wheels all turn when it is on the box. On the floor, every time it starts to move, it stops, and the board's L light goes out for a moment and comes back on. What is happening, and what do you try first?
2. **The speed letters do nothing.** On a friend's car, the phone says *connected* and `F`, `B`, `L` and `R` all work, from the phone and from the Serial Monitor. But `1` and `9` give exactly the same speed, on both sides. Which boxes do you know work? Where would you look?
3. **Backwards on both sides.** A car goes back when you press Forward, and forward when you press Back. Left turns it right, and Right turns it left. What is wrong, and what is the smallest fix?
4. **One side flat out.** On someone else's car, one side always runs at full speed and the other side's speed changes with the speed letters. Name two different faults that would do this, and how you would tell them apart without a meter.
5. **Silent car.** A car drives from the phone, forward, back and both turns, but the horn makes no sound and neither does the reversing beep. Which box is it in, and what do you check first?
6. **Never connects.** On a car with an HM-10, the module's light keeps blinking, and the remote never says *connected*, however many times you press Connect. Which box is it, and what are two things to check?

The rubric

Score each row from your notes. The point of the rubric is the last column: each score comes with the next thing to do.

TABLE 35.3 The one-week check rubric. Each score points at the next step

WHAT YOU LOOK FOR	NOT YET	GETTING THERE	SECURE	NEXT STEP
Names a box	"It's broken" or "it's the code"	Names a part, but not the box it belongs to	Names the box, and says which stage test would prove it	Not yet: reread Chapter 3 together and redo its talk prompt on the real car
Tests before changing	Starts pulling wires out	Changes something, then tests	Runs a stage test first, then changes one thing	Not yet or Getting there: do Chapter 21 together
One change at a time	Changes several things at once	Changes one thing but doesn't undo it when it didn't help	Changes one thing, tests, undoes it if it didn't help	Use the loop card at the next fault, out loud, one line at a time
Says what was found	Can't say why it works now	Says what they did ("I swapped the wires")	Says what was wrong and why the fix works ("this side was backwards, so the car spun")	Ask "why did that fix it?" after every fix from now on
Safety	Touches wiring with the switch on	Switches off when reminded	Switches off before touching anything, unprompted	Not yet: stop and reread Chapter 5 together before the car is used again
The three questions	0 or 1 answer names the right box	2 answers name the right box (the book's target)	3 of 3, each with a reason	Fewer than 2: try the Part 3 puzzles (Chapter 25) together, one a day

The book's targets: at least two of the three questions answered with the right box, and the planted fault found and fixed within 15 minutes. They are the targets this book was designed around, not norms for children of any age. A child who misses them hasn't failed anything: the book, or the practice, has a gap, and the "Next step" column says where.

NOTE

In a club. One child at a time takes one adult about twenty minutes each, which is hours for a room. Instead, plant one fault per car (A, B or C, mixed across the room), and have each pair swap to another pair's car. The builder finds and fixes; the checker ticks the record sheet's five rows; the adult only times and checks that the switch goes off first. The three questions are answered on paper, one sheet each, without the book: score "names the right box" from the writing. Untested in a group, like the rest of Chapter 34. A record sheet for this check, one per child, is in the club pack at troniction.com/go/car-club.

CHECK YOURSELF

1. Your child fixed planted fault A in four minutes, but when you ask why it works now, says "I just tried swapping them". Which rubric rows can you score from that, and what is the next step?

2. Why does this check happen a week later, rather than right after the first drive?

Answers at the back of the book on page 255

/// CHAPTER

36

Where next: extensions and resources

Some children drive the car for an afternoon and move on. That is a good outcome: they built a working machine and found out how it works. Others come back the next day with “can it…?”. This chapter is for them, and for you, so you know where each “can it…?” leads before you agree to it.

The list runs roughly from the smallest step to the largest. Each one starts from the car you already have.

Things the car’s own sketch is waiting for

The sketch already has letters that do nothing yet: `W` and `w`, and `U` and `u`, were meant for lights at the front and the back (Part 4 shows them, and the bug hiding in one of them). Wiring an LED and a resistor to a free pin, then filling in one of those empty branches, is the smallest real upgrade: one new part, a few lines of code, and a clear test (the light comes on when you press the button).

NOTE

Untested idea. Lights, a stop-before-the-wall sensor, line following and a fail-safe are the next edition’s Part 5. None of them has been built on this book’s car yet, so this edition gives no wiring or code for them. Before you start one, check which pins are free on *your* car: 2 and 3 belong to the Bluetooth module, 4 to the buzzer, and 5, 6 and 9 to 12 to the motor driver.

Bigger upgrades, and what each one teaches

TABLE 36.1 Upgrades, roughly in order of difficulty. None is built in this edition

UPGRADE	WHAT IT ADDS	WHAT IT TEACHES	STATUS IN THIS BOOK
Headlights and reversing lights	Four LEDs on the sketch's unused letters	Why an LED needs a resistor; free pins; filling in a branch	Planned for v2.1; not built
Stop before the wall	An ultrasonic distance sensor at the front	Timing a sound echo; a car that decides for itself	Planned for v2.1; not built
Follow a line	Two line sensors and a taped track	Turning sensor readings into a rule table	Planned for v2.1; not built
The fail-safe	The car stops by itself if the phone goes quiet	Why "no news" must mean "stop"; timing without `delay()`	Planned for v2.1; not built
Your own remote page	A web page of your own that drives the car (HM-10 cars)	HTML, JavaScript and Web Bluetooth	Not covered here; see *Make It Connect*

The fail-safe is the one I would do first, even though it is not the most fun. Part 4 shows that the car keeps doing the last thing it was told. If the phone's battery dies while the car is driving, it keeps driving. A car that stops by itself when the letters stop arriving is a lesson in designing for failure, and it is what makes the others safe to try on the floor.

When the car isn't enough

Your child wants to understand the Bluetooth link properly. That is the subject of *Make It Connect*, my book about why an Arduino's Bluetooth link fails and how to fix it, from the first wire to every module this car can use. It covers the HC-06 and the HM-10 in depth, renaming a module, and the iPhone question.

Your child wants to write programs, not just change this one. The Troniction Academy's first stage, Foundations, starts from never having written code and ends with a sketch of their own design. It is the next step after Part 4 for a child who enjoyed predicting and changing the code.

Your child wants voice control or a car that drives itself. The weekend build course does that on this same car.

Free, and not mine. Arduino's own documentation (docs.arduino.cc) has the official reference for every function in the sketch, and a getting-started guide for the board. It is written for adults, but a parent and child can read it together. A local makerspace or school robotics club is often the best next step of all: other people building, and someone to show your child what you can't.

None of these is needed to finish this book. Everything the car needs is in its pages, and it will still be here next weekend.

Where to go from here: the current versions of everything on this page, including the upgrades as they are built and tested.

SCAN OR TYPE **troniction.com/go/car-next**

CHECK YOURSELF

1. Your child wants to add the stop-before-the-wall sensor next weekend. This edition has no steps for it. What two things do you check before you start?
2. Why might the fail-safe be the right first upgrade, even though it is the least exciting?

Answers at the back of the book on page 255

The full parts list

Prices are US list prices, single quantities, on the dates shown. They move; the current list, with links, is at troniction.com/go/car-parts. None of the links there earn me anything.

Every car needs these

TABLE 1 Parts every car needs, whichever Bluetooth module you choose

PART	QTY	ABOUT (CHECKED)	NOTES
The board: Arduino Uno R3 or a clone, with a USB cable	1	$10.99 (Aug 2026)	A data cable, not charge-only. **Not the Uno R4**
The motor driver: L298N, screw-terminal	1	$6.98–9.99 (Aug 2026)	Packs of four about $11.50 (Sept 2026)
Four-wheel chassis kit with four TT gear motors **with wires attached**	1	$20.99–26.99 (Aug 2026)	Or a chassis and four Adafruit #3777 ($2.95 each)
The buzzer: a **passive** buzzer **module** (three pins: S, +, –)	1	not priced yet	Passive, so it can play the tune; an active one only beeps. A bare buzzer needs soldering
The battery holder: 2 × 18650, **with a switch** and leads	1	$7 (Sept 2026)	Check it is wired for 7.4 V
The cells: 18650, button-top	2	about $10 (estimate)	Not sold loose on Amazon US: a battery specialist
18650 charger	1	$13.99 (Aug 2026)	
Jumper wires: male-to-female (at least 15) and male-to-male (at least 6)	21 or more	$3.99 for 40 (Aug 2026)	Female-to-female ones aren't used
The breadboard: mini, 170 holes (400 also works)	1	400: $6.25 (Sept 2026); 170 not priced	Holds the resistors and the buzzer's power
Resistors: one 1 kΩ, one 2 kΩ (2.2 kΩ works)	1 each	$10 kit (Sept 2026), about $1 loose (estimate)	Protect the module's listening pin
Small Phillips screwdriver	1	—	Probably in the house

Then choose one Bluetooth module

TABLE 2 The two module options. Everything else on the list is the same

	HC-06	HM-10
Phones that can drive it	**Android only**	Android (Chrome) or iPhone (the free Bluefy browser)
The module	HC-06: $9.47–9.99 (16 Aug 2026)	**DSD TECH HM-10**: $11 (Sept 2026). Name this exact one: look-alikes share the chip but not the behaviour
How it was checked	The module and an Android phone: built and driven on the 2021 car, on pins 0 and 1. On this edition's pins, 2 and 3: checked on a computer and against the datasheet	Checked on a computer and against the datasheet; not yet built

The whole list comes to about $100 to $110, bought one item at a time, with either module (an estimate from August and September 2026 listings; the figure Chapter 1 prints).

You also need

- **A laptop you can install software on:** Windows or Mac. Not a Chromebook, a tablet, or a locked school or work laptop.
- **A phone:** for the HM-10, an Android phone with Chrome or an iPhone with the free Bluefy browser; for the HC-06, an Android phone.
- **A box or a stack of books** to stand the car on, wheels off the table, for the tests.

The two buying mistakes

1. **Motors with no wires.** Adafruit sells near-identical TT motors with no wires (#3801 and #3802). Those need a soldering iron. Buy #3777, or a chassis kit whose photos show wires on the motors.
2. **The wrong Bluetooth module.** An HC-05 or HC-06 looks almost exactly like an HM-10, and cheap HM-10 look-alikes behave differently. If you want to drive from an iPhone, you need a genuine HM-10. The free page at troniction.com/go/car-which-ble shows how to tell them apart in a minute.

Glossary

Each concept in this book has one name, and the book uses only that name. Other books and websites use other words for the same things; they are listed after “Elsewhere”, so you can match them up. The number at the end of each entry is the chapter where the term is first explained.

The parts

The battery

The two cells in their holder: the car’s only source of power once the USB cable is out. It feeds the motor driver and the board. *Chapter 3.*

The battery holder

The plastic case that holds the two cells, with a switch and two wire leads. The switch is the child’s; the cells are the parent’s. *Elsewhere:* battery box, battery pack. *Chapter 4.*

The Bluetooth module

The small board that lets the phone talk to the board without wires. This book uses one of two: the **HC-06** or the **HM-10**. *Elsewhere:* BT module, radio, Bluetooth adapter, transceiver. *Chapter 3.*

The board

The Arduino Uno: the part that runs the sketch and decides what the car does. *Elsewhere:* Arduino, Uno, microcontroller board, controller, “the brain”. *Chapter 3.*

The breadboard

A small plastic block with rows of holes joined underneath, which holds the two resistors and the buzzer’s power wires without soldering. *Elsewhere:* solderless breadboard, protoboard. *Chapter 4.*

The buzzer

A small module that beeps when the board tells it to: the horn, the reversing beep and the tunes. *Elsewhere:* piezo, beeper, sounder. *Chapter 4.*

The cells

The two rechargeable 18650 lithium-ion cells that go in the battery holder. Only the parent puts them in, takes them out and charges them. *Elsewhere:* batteries, 18650s, Li-ion cells. *Chapter 4.*

The chassis

The car's frame, with its four wheels and the mounts for everything else. *Elsewhere:* car kit, robot base, frame. *Chapter 7.*

Jumper cap

A tiny plastic-covered clip that joins two neighbouring pins. The motor driver comes with two, on ENA and ENB; this car needs them off. Not the same thing as a jumper wire. *Elsewhere:* jumper, shunt. *Chapter 8.*

Jumper wire

A short wire with a plug at each end. A **male** end has a pin; a **female** end has a socket. *Elsewhere:* Dupont wire, hookup wire, jumper lead. *Chapter 4.*

The motor driver

The L298N: the board that takes the battery's power and pushes it through the motors, in the direction and at the speed the board's pins ask for. *Elsewhere:* L298N, driver board, H-bridge module, motor controller. *Chapter 3.*

The motors

The four TT gear motors that turn the wheels: a small electric motor and a plastic gearbox in one yellow case. *Elsewhere:* gearmotors, DC motors, TT motors. *Chapter 4.*

The phone

The phone that drives the car: an Android phone, or an iPhone with the HM-10. *Chapter 3.*

The remote

What the phone uses to drive the car: the remote page in Chrome or Bluefy (HM-10 cars), or the Serial Bluetooth Terminal app on Android (HC-06 cars).

Elsewhere: controller, app, controller app. *Chapter 10.*

Resistor
A part that limits how much current flows. The car uses two, a 1 kΩ and a 2 kΩ, as a voltage divider (with either module). *Chapter 4.*

Screw terminal
A metal clamp with a screw on top: the bare end of a wire goes in, the screw holds it. The motor driver's motor and power connections are screw terminals. *Elsewhere:* terminal block. *Chapter 7.*

The sketch
The program that runs on the board. *Elsewhere:* code, program, firmware, `.ino` file. *Chapter 3.*

Power and signals

5V
The board's own steady 5-volt supply, and the pin it comes out of. *Chapter 10.*

Brown-out
The board restarting because its supply voltage dipped too low for a moment, usually because tired cells sag when the motors start. *Chapter 19.*

Current
How much electricity flows, measured in amps (A) or thousandths of an amp (mA). *Chapter 14.*

GND (ground)
The return path for every circuit on the car, and the zero that every voltage is measured from. *Chapter 9.*

Shared ground
The board's GND and the motor driver's GND joined by a wire, so a signal from the board means the same thing to the motor driver. *Chapter 9.*

Vin
The board's input for power from the battery. With the battery connected here, the board keeps running when the USB cable comes out. *Chapter 9.*

Voltage
The push that makes current flow, measured in volts (V). *Chapter 14.*

Voltage divider
Two resistors in a row, which turn the board's 5 V signal into about 3.3 V for the Bluetooth module's listening pin. *Elsewhere:* level shifter (a different part that does a similar job). *Chapter 10.*

How the car works

Direction pins (IN1 to IN4)
The four motor-driver pins that choose which way each side's motors turn. IN1 and IN2 are one side; IN3 and IN4 the other. *Chapter 8.*

H-bridge
Four switches around a motor, inside the motor driver, that can send current through it either way. That is how a motor runs backwards. *Chapter 16.*

Heatsink
The ridged metal block on the motor driver that carries away heat. It gets warm in use. *Chapter 5.*

PWM (pulse-width modulation)
Switching a pin on and off very fast and changing how long it stays on, so a motor gets an average of less than full power. It is how the car has speeds. Only pins 3, 5, 6, 9, 10 and 11 on the Uno can do it. *Chapter 17.*

Speed pins (ENA and ENB)
The two motor-driver pins that set each side's speed, connected to the board's pins 5 and 6. *Elsewhere:* enable pins. *Chapter 8.*

Speed ladder
The list in the sketch that turns the speed letters `0` to `9` and `q` into PWM values from 150 to 255. *Chapter 29.*

The link

Baud rate
How fast the letters are sent, in symbols per second. This car uses 9600 everywhere. *Chapter 6.*

BLE (Bluetooth Low Energy)
The kind of Bluetooth the HM-10 uses. iPhones and web pages can talk to it. *Chapter 20.*

Bluetooth Classic
The older kind of Bluetooth the HC-06 uses. Android phones can talk to it; iPhones can't. *Chapter 20.*

Bluefy
A free web browser for the iPhone that can talk to BLE modules, which Safari can't. *Chapter 10.*

Connected
The phone and the Bluetooth module can talk. It does not prove that letters reach the board, or that the board answers. *Chapter 3.*

Letter
One character the phone or the laptop sends to drive the car, such as `F` for forward. *Elsewhere:* command, byte, character. *Chapter 11.*

Line ending
An extra, invisible character the Serial Monitor can add after what you type. This car needs **No line ending**. *Chapter 9.*

TX and RX (TXD and RXD)
Transmit and receive: the pin that sends and the pin that listens. One side's TX goes to the other side's RX, which is why the wires cross. *Chapter 10.*

UART
The name of the simple serial link between the board and the Bluetooth module. *Elsewhere:* serial, TTL serial. *Chapter 20.*

Web Bluetooth
The feature that lets a web page in Chrome or Bluefy talk to a BLE module. *Chapter 20.*

On the laptop

Arduino IDE
The free program on the laptop where you open the sketch and upload it. *Chapter 6.*

`Car heard:`
What the v2 sketch sends back over the Bluetooth module for every letter it receives, such as `Car heard: F`. It proves the round trip. *Chapter 11.*

`Car ready`
What the v2 sketch prints in the Serial Monitor when it starts, while the board's L light comes on and stays on. Together they are Stage A's test. *Chapter 6.*

L light
The small light on the board joined to pin 13. The v2 sketch turns it on at start and leaves it on. *Chapter 6.*

Port
The name the laptop gives the board when it is plugged in by a data cable. *Chapter 6.*

Serial Monitor
The window in the Arduino IDE that shows what the board prints, and sends what you type to the board. *Chapter 6.*

SoftwareSerial
The part of the sketch that lets the board talk to the Bluetooth module on pins 2 and 3 instead of its own serial pins 0 and 1. *Chapter 26.*

Upload
Copying the sketch from the laptop into the board. *Elsewhere:* flash, program, burn. *Chapter 6.*

Upload trap
An upload that fails because a Bluetooth module is on the board's pins 0 and 1, which the USB cable also uses. This book's car avoids it by using pins 2 and 3. *Chapter 32.*

USB driver (CH340)
A small program some clone boards need on the laptop before a port appears. Not to be confused with the motor driver. *Elsewhere:* the driver. *Chapter 6.*

In the code

`analogWrite()`
The instruction that sets a PWM value on a pin. On a pin that can't do PWM, it can only switch fully on or off. *Chapter 17.*

`delay()`
An instruction that makes the board wait, doing nothing else, for a number of milliseconds. *Chapter 30.*

Dispatcher
The part of the sketch that reads one letter and picks the action that matches it. *Chapter 27.*

`loop()`
The part of the sketch the board runs again and again, for as long as it has power. It does one thing at a time. *Chapter 26.*

`setup()`
The part of the sketch the board runs once, when it starts. *Chapter 26.*

`tone()`
The instruction that makes the buzzer play a note. *Chapter 30.*

The method

Builder, checker
The two roles. The builder (the child) does every physical step except the cells and the install; the checker (the parent) reads each step aloud and checks it against the picture. *Chapter 2.*

The four boxes
The whole car as one picture: the phone → the Bluetooth module → the board → the wheels (the motor driver and the motors), with the battery feeding the motor driver and the board. *Chapter 3.*

Half-splitting

Finding a fault by testing the middle of the path first, so each test rules out half of what is left. *Chapter 21.*

The loop

Five steps for when something doesn't work: look, which box, change one thing, test again, say what you found. *Elsewhere:* debugging loop. *Chapter 3.*

Planted fault

A fault an adult puts into a working car on purpose, for the child to find. *Chapter 35.*

Stage

One part of the build that ends with its own test: A (the board), B (the wheels), C (the Bluetooth module) and D (the round trip and the drive). *Chapter 3.*

Stage test

The test at the end of a stage. Only that stage can fail it. *Chapter 3.*

Stopping point

A place where a session can safely end: after a stage test that passed. *Chapter 2.*

Colophon and corrections

Build a Car With Your Kid, second edition. Written by Troniction.

The car in this book is a real car, first built for the 2021 edition and rewired in August 2026. What was built and driven, and what was checked another way, is listed on “How this book was tested” at the front. The drawings are drawn from the car’s wiring tables and the parts’ datasheets. Any generated picture is listed as generated in the credits.

The book is written in Markdown and typeset with pandoc and Paged.js. The type is Inter for the text, Space Grotesk for headings and JetBrains Mono for code and labels.

Corrections

Errors get a dated correction note here, and in the online copy at troniction.com/go/car. Nothing is quietly fixed.

None yet in this edition. If you find one, tell me at troniction.com/go/car.

More from Troniction

This book is free, and the whole build is in it. If your child wants more, these are the next steps I make. Prices change and offers come and go, so this page prints none: the current ones, with what each includes, are at troniction.com/go/car-next.

Make It Connect (book). Why an Arduino's Bluetooth link fails and how to fix it: every module this car can use, the iPhone question, and a troubleshooting section for when the phone says connected and nothing happens. PDF and EPUB direct; also on Kindle and in paperback.

The weekend build (course). Eight short lessons on this same car, ending with voice control and a car that drives itself. Every sketch compiled; a priced parts list.

The Troniction Academy (course). Two stages, from never having written code to programs of your own design, with graded work and a certificate for each stage.

No soldering. No prior electronics. A named parts list you can price before you commit.



Everything on this page, at today's prices.

SCAN OR TYPE **troniction.com/go/car-next**

Answers

Chapter 2 · How to help without taking over

1. Don't touch the wire. Point at the figure and ask, "Which pin does the picture say?" Your child finds the difference and moves the wire (**"When a wire is wrong"**, page 18).
2. Stop trying random changes. Run the debugging loop from Chapter 3, then turn to Chapter 12 or scan the stage's code for its video and test page (**"The 10-minute rule"**, page 19).

Chapter 3 · The four boxes

1. Only the phone and the Bluetooth module, and the link between them. It hasn't proved that the letter reaches the board, or that the wheels turn (**"Being connected isn't the same as the letter arriving"**, page 23).
2. The motor driver and the board (**"The journey of one letter"**, page 22).

Chapter 4 · Unbox and find

1. They're the unwired version of the motor, which needs soldering. Return them and buy motors with wires attached, such as Adafruit #3777 (**"Two buying traps"**, page 28).
2. Only pins 3, 5, 6, 9, 10 and 11, the ones with the `~` mark. This car uses 5 and 6 for speed. Chapter 8 tells the story of a car that got this wrong for years.

Chapter 5 · Safety: read this together

1. The cells are in the holder, and the holder's switch is on. (And remember the board can also be fed by the USB cable, so "all power off" means the switch off and the cable out.) (**"When does anything become live?"**, page 37)
2. Stop using it. Don't bin it: take it to a battery recycling point (**"The only dangerous thing in the box"**, page 36).

Chapter 6 · Stage A · The board: put the sketch on it

1. The USB cable only charges and has no data wires. Try another cable. If the board is a clone with a CH340 chip, it may also need its driver (**"For the parent: install and set up"**, page 39).
2. The sketch never reached the board. The board is still running the blink program it came with. Upload again and wait for "Done uploading" (**"The Stage A test"**, page 42).

Chapter 7 · Stage B1 · Chassis and motors

1. Because the two motors on one side always do the same thing. To go forward both turn forward; to turn, one whole side goes backwards (**"What you'll do"**, page 45).
2. The screw wasn't clamping the metal pin: the pin wasn't pushed in far enough, or the screw clamped the plastic. Loosen, push in fully, tighten, tug again (**"Screwing a wire into a terminal"**, page 47).

Chapter 8 · Stage B2 · Motor driver to the board

1. A cap is still on ENA or ENB, or a speed wire is on a pin without the `~` mark, which can't do speed (**"The two caps come off**

first", page 52, "Not every pin can do speed", page 53).

2. The left side turns the other way, for every letter: forward becomes backward for those two wheels. The right side doesn't change ("How four wires make a motor go both ways", page 53).

Chapter 9 · Stage B3 · Power, and the wheels test

1. Stage B, the wheels. One side is turning backwards. Switch off, and on that side swap the red and black wires of both motors. Then type F again, because after any change to the motor wires you go back to F ("The wheels test", page 64).
2. The jumper from the motor driver's +12V screw to the board's Vin pin, or the one from GND to GND, or one isn't clamped. The battery reaches the motor driver but not the board ("The last check: unplug the USB cable", page 67). In the four boxes of Chapter 3, it's the battery's power line to the board that's broken.
3. So the wheels can spin without the car driving off the table, and you can watch all four at once ("The checklist before power", page 62).

Chapter 10 · Stage C · The Bluetooth module

1. The module's power: VCC in column 1 and GND in column 12, and the board's 5V and GND jumpers that feed those columns ("Test C1: the module's light", page 73).
2. That the phone's Location is on and Chrome has the Nearby devices permission. And that the module isn't paired in the phone's Bluetooth settings; if it is, Forget it there ("If you have the HM-10", page 74).

Chapter 11 · Stage D · The round trip, then drive

1. The phone, the Bluetooth module and the board are all fine. The letter arrived and was answered. Look in the wheels box and its power: is the holder switched on? Then redo the wheels test ("The round-trip test", page 80).
2. The car keeps driving forward while the song plays, about 16 seconds, and it ignores any letter you send until the song ends. Then the song starts again, until a new letter arrives. Part 4 explains why and shows how to fix it ("The letters", page 84).

Chapter 12 · If it doesn't move

1. Most likely the cells have run down. When the motors start, the voltage dips below what the board needs, and the board restarts. Charge the cells ("What a tired battery looks like", page 89).
2. The wheels test from Stage B, with the car on its box and the USB cable in. It's the test that only the wheels box can fail. If `Got F from USB` appears and two wheels still don't turn, check those motors' wires in their screws and the six signal wires, starting with the side that doesn't move ("Find the stage first", page 87).
3. (a) Test C1, the module's power: VCC, GND and the board's 5V and GND jumpers. (b) The wheels box: the left motors' wires, or ENA, IN1 and IN2. (c) Stage D: the letter isn't reaching the board, so check TXD to pin 2 ("Find the stage first", page 87).

Chapter 14 · Volts, amps and a shared ground

1. A pin is rated for 20 mA (40 mA is the absolute maximum), and a free-running gear motor wants about 150 mA, over 1,000 mA when stalled. The pin can signal, but it cannot push; the motor driver does the pushing.
2. They most likely do nothing. The motor driver has power (its red light is on), but the signals from the board have no way back to the board, so the motor driver can't tell HIGH from LOW. That is the site's "nothing moves; the red power LED is on" row, and the first thing to check is the shared ground.

3. 5 ÷ 1,000 = 0.005 A = 5 mA. That is still a signal-sized current, well under the pin's 20 mA.

Chapter 15 · The board

1. Only to a PWM pin: 3, 6, 9, 10 or 11. Pin 6 is `ENB` and 9, 10, 11 are taken by direction wires, so you would have to move a direction wire to a free non-PWM pin first. Pin 3 is the board's talking pin to the module. (One more catch you'll meet in Part 4: while the buzzer plays, pins 3 and 11 can't do PWM, because the sound uses the same timer.) Not 13, because 13 can't do PWM: every speed would become full on, which is the 2021 bug.
2. The program survives (it is in flash). The speed doesn't (it was in working memory). On every start the sketch sets the slowest speed again and holds the wheels still.
3. The car stops: the board restarts, runs the sketch from the top, and holds the wheels still. The `L` light goes out for a moment and comes back on. It will drive again when the phone sends the next letter. If you were at speed 9, you are back at the slowest speed.

Chapter 16 · The motor driver and the H-bridge

1. It brakes: both ends of the motor are tied to plus, so the motor's own generated current loops round and stops it fast. That is not the same as `ENA` off, which opens every switch and lets the motor coast. The car's `S` uses the brake.
2. Each closed transistor keeps a little voltage, about 2 V for the pair at these currents, and that lost voltage times the current flowing is heat in the chip. The energy comes from the battery.
3. Swapping the motor leads reverses the current through that motor for the same `IN` pattern. Swapping `IN3` and `IN4` reverses the `IN` pattern itself, so the bridge sends the current the other way. Either way, the current through the motor is reversed, and so is the motor.

Chapter 17 · Speed is a trick: PWM

1. Full speed. On this car `0` sets the value 150 (the same as `1`), and 150 is more than 128, so pin 13 would simply switch fully on. Every step on the ladder would do the same: that is the 2016–2026 bug.
2. Below the floor the motors can't overcome the gears and the car's weight, so the bottom buttons would only make the motors hum and warm up. Starting at 150 means every button moves the car (on the 2021 car, at least: that is the author's choice, not a measurement).
3. A speed wire on a non-PWM pin (for example pin 13): every value from 128 up becomes full on. Or a jumper cap still on `ENA` or `ENB` on the motor driver: it ties the enable to 5 V, so the board's PWM never reaches the bridge. Same symptom, two causes.

Chapter 18 · Motors, gears and skid steering

1. Its two sides are on each other's screw terminals: the left motors are on `OUT3`/`OUT4` and the right ones on `OUT1`/`OUT2`. `F` and `B` drive both sides the same way, so they can't show it; `L` and `R` drive the sides differently, so they swap. Chapter 9's wheels test is built to catch exactly this.
2. The wheels would try to spin about 48 times faster with about 1/48 of the turning force, which is far too little to move a car carrying a battery. It would most likely sit still and hum, or barely creep.
3. Speed `1` is a 59% duty cycle, which gives the motors a fairly small average push (Chapter 17). On the smooth floor that push is just enough to roll the wheels; the rug grips and drags the tyres, so the same push isn't enough. The floor speed depends on the surface.

Chapter 19 · Power: cells, Vin and brown-outs

1. They give about 6 V fresh (less as they drain), which is below the board's recommended 7 V, and leaves the motors only about 4 V after the motor driver's 2 V. And ordinary AA cells sag badly at the currents motors draw.
2. Starting four motors draws a big current, the battery's voltage dips under it, and the board's supply sags low enough that the chip restarts: a brown-out.
3. The wheels box (motor driver and motors), and its power: the letters prove the phone, the module and the board all work. Before anything else, switch off and charge the cells, then try again.

Chapter 20 · The Bluetooth module: how a letter reaches the board and back

1. The remote connects (the radio is fine), but nothing arrives: no `Got F from phone`, no `Car heard`, no movement. Talk is wired to talk, so nobody listens. Swapped at the board, this can also strain the module: pin 3's 5 V reaches its `TXD` with no resistor. Switch off and swap them back. (Chapter 23's swap is at the module end, where the 1 kΩ resistor limits the current.)
2. `RXD` goes straight to the module's 3.3 V chip, so it needs the divider. `VCC` goes to the module's own regulator, which is built to take 5 V and make 3.3 V.
3. The phone, the module and the module's wire to the board are proven (and the board is running). What's left is the board's signals to the wheels: the motor driver, its power and the motors. A jumper cap left on, a loose signal wire, a missing shared ground or the battery switch are all candidates; and "connected" never covered any of them.

Chapter 21 · Four boxes and half-splitting

1. The half after the board: the sketch's decision, the six signal wires, the motor driver, the motors, and the battery's feed to the motor driver. The phone, the radio link, the Bluetooth module and its wire into pin 2 are cleared, because the letter could not have reached the Serial Monitor without them.
2. (c), (b), (a), (d). Look first; then the test in the middle, which says whether the letter reaches the board; then change one thing, with the power off; then the same test again, this time watching for both directions.
3. The power line: the switch, the cells, and the lead to the board's Vin pin. Two boxes failing at once almost always share one cause, and the only thing these two share is power.

Chapter 22 · Worked example: the speed that never changed

1. Any two of: it failed silently (the board rounds instead of giving an error); the car still did everything people tested it for (go, stop, turn); and nobody had ever seen it crawl, so nobody missed it. A fault is only noticed when a test compares the right two things.
2. A jumper cap left on ENA and ENB. The cap ties the speed input to full power, so the motor driver ignores the board's speed wire. That is why the wheels test checks speed `1` against speed `9`.
3. Pin 7 has no `~`, so it can't do speed. ENB drives the right side in this book's wiring (OUT3 and OUT4), so the right side gets 150 or more, which rounds to full power, at every speed; the left side, on pin 5, really slows down. At speed `1` the left side crawls and the right side runs at full power, so the car curves round towards its slow side. At speed `9` both sides are near full power and it runs nearly straight.
4. (c), (b), (e), (d), (a). Clear the boxes (direction works, so letters arrive and the board runs); list both suspects; look at the caps, which takes seconds; read the two

pin lines; then predict, and compare the prediction with the symptom.

Chapter 23 · Worked examples: spins on the spot, and connected but still

1. **Which box:** the wheels test clears the wheels, the board and the battery. *Connected* clears the phone and the module's power and radio. Left: the wires between the module and the board. **Split:** typing `F` in the Serial Monitor sends the board's answer, `Car heard: F`, out through pin 3 to the module and the phone. It came out as `Got F from USB` with turning wheels, but no `Car heard` on the phone: so the board→module direction is broken too, not only the module→board direction. **Suspects:** a fault that breaks both directions while power and radio work is the data pair, TXD and RXD, swapped or both unseated. **Change one thing:** USB out, switch off; swap the two data wires at the module, so TXD goes to pin 2 and RXD to hole a9, where the two resistors meet (steps C7 and C8). **Test again:** reconnect, press Forward: the Serial Monitor shows `Got F from phone` and the phone shows `Car heard: F`. **Say what you found:** "Connected, but no letters either way: the module's TXD and RXD wires were swapped."
2. The two data wires are swapped at the module end: TXD goes to hole a9, where the two resistors meet, and RXD goes to pin 2. Both ends are talking into each other's mouths and neither is listening, so nothing gets through in either direction. Swapped at this end, nothing should be damaged: pin 3's 5 V meets the module's 3.3 V `TXD` only through the 1 kΩ resistor, which keeps the current under 2 mA. *How sure:* reasoned from the circuit, not staged. Swapped at the board end there is no resistor in the way, so never leave it like that. Swap the two wires back with the power off.
3. `B` drives the left side backward. The right side, flipped, turns forward. The sides turn opposite ways, so the car spins on the spot, the other way from `F`. (`F` and `B` spin; `L` and `R` drive straight. That pattern is the signature of one side flipped.)
4. (b), (d), (c), (a). Power off before any wire moves, then the one change, then the tug test, then the same test as the symptom.

Chapter 24 · Your Serial Monitor is a probe

1. The baud menu at the bottom of the Serial Monitor. It must say 9600, the speed in the car's sketch. Odd symbols mean bytes are arriving at the wrong speed, so the link and the wiring are fine.
2. `Got B from USB`, then `Got (newline) from USB`. The newline arrives about a thousandth of a second after the `B` and counts as "anything else", so it stops the car almost before it starts: at most a twitch backward, and usually no beep, because the beep only gets a turn every tenth of a second. Set the menu to No line ending.
3. A lower-case `f`. The sketch only knows a capital `F`, so `f` counts as "anything else" and stops the car. The letter arrived perfectly; it was the wrong letter. Type a capital `F` (or set up the app's buttons to send capitals).
4. (b), (c), (a), (d). Settings first, or nothing else you read can be trusted; then is the sketch running; then did the letter arrive; then, if it did not, does the other direction work on its own.

Chapter 25 · Ten broken cars

1. **The wheels box, the speed half: the jumper caps are back on ENA and ENB.** Someone thought they had gone missing and put them back, and the two speed wires were left hanging. A cap ties the speed input to full power, so the motor driver ignores the board. Fix: switch off, pull both caps off, push the speed wires back onto ENA and ENB, and test speed `1` against speed `9`. *How sure:* bench-proven that speed varies with the caps off (the original car, August

2026); the caps-on behaviour is from the motor driver's documentation.

2. **The wheels box, the right side: the right motors' wires have pulled out of OUT3 and OUT4.** Everything that only the right side uses is a suspect: the ENB, IN3 and IN4 wires and the motor wires in OUT3 and OUT4. A fall loosens screw terminals more easily than push-on wires, and a tug test checks them in seconds, so check the terminals first. Fix: switch off, push the wires back in, screw down, tug each one, test. *How sure:* reasoned from the circuit; the other suspects give the same symptom, which is why the tug test comes first and the wires next.
3. **The wheels box, both sides: both motor pairs are the other way round.** The beep sounds on Back, so the sketch chose "back" correctly; the wheels did the opposite. The board is right and the wires are wrong. Fix: switch off, swap red and black on one side (both motors), and test: the car now **spins** on `F`, which is progress: it is the signature of one flipped side from Chapter 23, page 155. Then swap the other side and test again. *How sure:* the beep-on-`B` is in the sketch (tested on a computer); the rest is reasoned from how a motor reverses.
4. **Not a box: the Serial Monitor's line-ending menu.** The Serial Monitor shows `Got (newline) from USB` straight after every letter. The car obeys the letter for about a thousandth of a second, then the newline counts as "anything else" and stops it. Fix: set the menu to No line ending. *How sure:* tested on a computer; the size of the jerk is not measured.
5. **The power line: the jumper from the motor driver's +12V screw to the board's Vin pin** (or its partner, GND to GND). Two boxes dark at once means the one thing they share, and the board's ON light being out says the board has no power; the module gets its power from the board. With the USB in, the laptop fed the board and hid the fault. Fix: switch off; the parent takes the cells out; redo steps B26 and B27 and tug both ends; cells in, switch on; unplug the USB and check the ON light stays lit (the last check of the wheels test). *How sure:* the original car needed exactly this lead to run without the USB.
6. **The power line: the cells are tired.** What changed was an afternoon of driving. As the cells run down they give less push, so each speed setting delivers less to the motors, and the slowest settings no longer overcome the car's weight and friction. Don't leave the motors humming: send `S`. Then switch off; the parent charges the cells, following the rules in Chapter 5, page 36. If it carries on, Chapter 19, page 128 explains what a tired battery does next. *How sure:* reasoned from the circuit and the cells' behaviour; how many speeds stop working depends on your cells and is not measured.
7. **The Bluetooth module's listening wire: the 1 kΩ resistor's leg is in the wrong column.** Letters arrive, answers don't, even for a typed letter: the board-to-module direction is broken and nothing else. In Figure 25.2, the 1 kΩ's second leg should be in d9, in column 9 with the 2 kΩ and the RXD wire. It is in d8, a column that nothing else uses, so the board's answer never reaches the module. Fix: USB out, switch off; move that leg to d9 (step C2 in Chapter 10, page 70); test with a typed `F`. *How sure:* reasoned from the circuit, with the resistors' values and the sketch's answer tested on a computer.
8. **The Bluetooth module's talking wire: TXD has slipped off pin 2.** The `Car heard: F` on the phone proves the board → resistors → module → radio → phone path, and that the module has power. The only part that phone-to-board letters use and nothing else does is the wire from the module's TXD to pin 2. Fix: USB out, switch off; push it back onto pin 2 (step C7); test from the phone. *How sure:* reasoned from the wiring

and the sketch, which is tested on a computer.

9. **The sketch: `BT_BAUD` is 38400, and the module talks at 9600.** A line appears for every press, so something arrives each time and the wires and radio link work. But it is never the letter that was sent: the two ends disagree about speed. What changed was the sketch. The HC-06 and the DSD TECH HM-10 both talk at 9600 out of the box (their datasheets say so); 38400 is the speed an HC-05, a different module, uses for its setup commands, and tutorials about it are full of that number. Fix: change it back to 9600, upload, test. *How sure:* the sketch's side is tested on a computer and the module speeds are from the datasheets; the exact codes you'd see depend on the letter and are not predicted. (Some HM-10s with newer firmware are reported to start at 115200 instead: the same symptom, on a car nobody has edited. That is a module setting, and *Make It Connect* covers changing it.)
10. **The phone box: the browser.** On an iPhone, Chrome is built on the same engine as Safari, and that engine has no Web Bluetooth, the part of a browser the remote page needs to reach the car. Fix: install the free Bluefy browser and open the remote there. *How sure:* from the browsers' published support (checked August 2026 for *Make It Connect*); recheck when Apple or Google change it.

Chapter 26 · Reading a sketch: once, then forever

1. No. A restart runs `setup()` again, which sets all four direction pins HIGH (wheels held still) and starts with no letter in `btsignal`, so the car sits still until it gets a new letter (**"Once: setup()"**, page 177).
2. No. It does one line at a time: listen, act, beep, then round again, fast enough that it looks like all three at once (**"Forever: loop()"**, page 178).
3. (c), (b), (d), (a), (e). `Car ready` is printed just before the L light is switched on, both in `setup()`. The wheels turn in the Act step, after `report()` has printed the line.

Chapter 27 · The letter dispatcher

1. `Got F from USB` and then `Got (newline) from USB`. The wheels twitch at most, because the newline reaches the `else` branch on the next pass and stops the car (**"The twitch, solved"**, page 185).
2. It stops. `H` matches none of the questions, so it falls to the final `else`, which sets all four direction pins HIGH (**"Investigate: the chain of questions"**, page 183).
3. The line-ending menu is on **Both NL & CR**. Set it to **No line ending** (Table 27.1).
4. (b), (c), (a). The chain starts with `if` and the catch-all `else` goes last. (d) doesn't belong: a second `F` question could never run, because (b) always answers yes first. Chapter 31 finds exactly this mistake in the car's own sketch.

Chapter 28 · Making it move: the motion code

1. It turns backwards. LOW, HIGH is the reverse of side B's forward pattern (HIGH, LOW) (Figure 28.1).
2. Side B reverses in every motion. `F` and `B` now spin the car; `L` now drives it backwards and `R` forwards. Stop is unchanged, because HIGH, HIGH swapped is still HIGH, HIGH.
3. (c), (b), (d), (e), (a). Find the side first, with the car running on the box; the battery goes off before you touch a wire; and after any change to the motor wires you go back to `F`.

Chapter 29 · The speed ladder

1. The `'0'` branch: `set_speed(0)` means the speed pins are never on, so the motors get no power (**"Investigate: from letter to number"**, page 194). The direction pins don't change, so a later digit starts the car again in the same direction.

2. The motors hum, or stay silent, and the car doesn't move: 40 is far below a floor of around 120–150 ("Why the bottom step is 150", page 195).
3. A jumper cap left on ENA/ENB, or a speed wire on a pin that can't do the fast on–off (only 3, 5, 6, 9, 10 and 11 can on an Uno). Check the jumper caps first: it is a look, not a rewire (Chapter 22).
4. (b), (c), (f), (d), (a). (e) doesn't belong: change one thing at a time, or you can't tell which change made the difference ("The safe-change routine", page 196).

Chapter 30 · Sound, and why the car stops listening while it sings

1. Because `V` is sticky: every pass of `loop()` calls `tone()` again and starts a new second ("Investigate: two ways to wait", page 201).
2. The five `F`s wait in the queue. When the song ends the board reads them one per pass, the car goes forwards, and the song doesn't repeat, because the newest letter is now `F`.
3. No. `nextNote` still holds where the song stopped, so it carries on from that note. To restart it, set `nextNote = 0;` in the branch that stops the car.
4. (b), (d), (a), (c). Check the clock first, play the note, note the time, move on to the next note. (e) doesn't belong: `delay()` is the line that froze the car in the first place ("Investigate: two ways to wait", page 201).

Chapter 31 · Find it: the bug since 2016

1. The second `else if (btsignal == 'U')`. The first `U` branch matches first and every `else` below it is skipped ("Catch it in the act", page 208, and Chapter 27's "first yes wins").
2. It stops. No branch asks about `'u'`, so it reaches the final `else`, which sets all four direction pins HIGH.
3. (c), (b), (e), (d), (a). Saving a copy comes before any change; the battery goes off before the upload.

Chapter 32 · Change it safely, then make it yours

1. Its module is on pins 2 and 3. Uploads use pins 0 and 1, so nothing else is talking on the upload wires ("Why this car has no upload trap", page 214).
2. Nothing: the sketch won't compile. The compiler stops with `'else' without a previous 'if'`, because an `else` can't follow the final `else`. The branch must go before it ("Design your own letter", page 213).
3. `else if (btsignal == 'C') { set_speed(150); digitalWrite(in1, LOW); digitalWrite(in2, HIGH); digitalWrite(in3, HIGH); digitalWrite(in4, LOW); }`, placed before the final `else`. After `C` then `F`, the speed stays 150: `F` doesn't change the speed.

Chapter 33 · What this build teaches, honestly

1. There is no single right answer. A good one names something your child *does* without the book: "she can tell me which box a fault is in", or "he can fix a swapped motor wire on his own". "He enjoyed it" is worth having, but it is not evidence of learning ("How you will know", page 219).
2. That children and teachers say they enjoy builds like this, and that the research does not show it teaches programming better than a screen does. What it can teach is a picture of the whole car and a way to find a fault, and you can check both a week later ("What the evidence says a build like this does", page 217).

Chapter 34 · Running it with 6 to 12 children

1. Do one table at a time. The other pair reads the pre-power checklist aloud, with the switch off, until you arrive. The cells stay in

throws away what the stage tests proved. If the wheels test still passes, the board and the wheels are fine, and the fault is in the Bluetooth module's wiring or the phone (**"When a pair is stuck"**, page 225).

3. At a stopping point: after the Stage A test, if it is the first session. Nobody knows yet whether a group reaches it in 45 minutes, so have the pairs read Chapters 3 to 5 before the session, and if the uploads haven't started by about half-way, end after the find-it list instead. Spare time goes on the "Try this" experiments, never on starting Stage B (**"The session plan"**, page 222).

Chapter 35 · The one-week check

The six one-week questions:

1. **Fine on the box, not on the floor.** The board is restarting: the L light goes out because the board resets, and comes back on when the sketch starts again. On the floor the motors work much harder and pull more from the cells; tired or half-charged cells can't keep the voltage up, and the board's supply dips. That is the battery, feeding the board. Charge the cells first. Then, switch off, and tug-test the battery leads at the board's Vin and GND.
2. **The speed letters do nothing.** The phone, the module and the board all work: letters arrive from both and the car obeys the direction letters. The fault is in how speed reaches the motor driver: both jumper caps still on ENA and ENB, or both speed wires on pins that can't control speed (only 3, 5, 6, 9, 10 and 11 can). Look at the motor driver's caps first: it needs no tools.
3. **Backwards on both sides.** Both sides are wired the same way round, but that way is the opposite of what the sketch calls forward. Left and Right swap for the same reason: each side turns the opposite way to the one the sketch asks for. The smallest fix is to swap the two motor wires on **both** sides, with the switch off. Swapping one side would make it spin.
4. **One side flat out.** A jumper cap left on (or put back on) that side's ENA or ENB, or that side's speed wire on a pin that can't control speed. Look at the motor driver first: a cap is visible. If there is none, follow that side's speed wire to the board and check its pin number against the pin table.
5. **Silent car.** Not the phone, the Bluetooth module or the link into the board: the car drives, so letters arrive and the board acts on them. The fault is in the board's output to the buzzer: its three wires. Switch off, then check the buzzer's S wire is on pin 4, and its + and – are in the breadboard columns the board's 5V and GND go to (Chapter 8).
6. **Never connects.** The Bluetooth module's box, and the phone's side of it: the blinking light says the module has power but no connection. Check that you opened the remote page in Chrome on Android or Bluefy on an iPhone (not the phone's Bluetooth settings, and not Safari); and on Android, that Chrome is allowed to find nearby devices.

Check yourself:

1. From that alone, only "Says what was found", and it is Getting there. The other rows need your notes from the timer, because a fast fix can still be a guess. The next step is to ask "why did that fix it?" after every fix (**Table 35.3**).
2. Because how easy something felt on the day is a poor guide to what was learned. What your child can still do a week later, without the book, is the honest measure (**"How you will know"**, page 219).

Chapter 36 · Where next: extensions and resources

1. Which pins are free on your car (2 and 3 are the Bluetooth module's, 4 the buzzer's, 5, 6 and 9 to 12 the motor driver's), and whether there is a tested guide for it yet:

the next edition's Part 5, or the page at troniction.com/go/car-next (**"Things the car's own sketch is waiting for"**, page 231).

2. Because the car keeps doing the last thing it was told. If the phone stops sending, it keeps driving. A fail-safe makes every later upgrade safer to test on the floor (**"Bigger upgrades, and what each one teaches"**, page 232).

Credits

Image credits

IMAGE	SOURCE	AUTHOR	LICENSE	CHANGES
figures/p4-sketch-map.svg	own work	Troniction	own work	none (drawn for v2)
figures/p4-dispatcher.svg	own work	Troniction	own work	none (drawn for v2)
figures/p4-hbridge-code.svg	own work	Troniction	own work	none (drawn for v2)
figures/p4-ladder.svg	own work	Troniction	own work	none (drawn for v2)
figures/p4-timeline.svg	own work	Troniction	own work	none (drawn for v2)
figures/p2-01-analogies.svg	own work	Troniction	own work	none (drawn for v2)
figures/p2-01-ground.svg	own work	Troniction	own work	none (drawn for v2)
figures/p2-02-uno-pins.svg	own work	Troniction	own work	none (drawn for v2)
figures/p2-03-hbridge.svg	own work	Troniction	own work	v1 figure; last line corrected (two IN pins and one EN, not four)
figures/p2-03-heat.svg	own work	Troniction	own work	none (drawn for v2)
figures/p2-04-pwm.svg	own work	Troniction	own work	v1 figure; unchanged
figures/p2-04-ladder.svg	own work	Troniction	own work	none (drawn for v2)
figures/p2-05-gearbox.svg	own work	Troniction	own work	none (drawn for v2)
figures/p2-05-turning.svg	own work	Troniction	own work	none (drawn for v2)
figures/p2-06-power-path.svg	own work	Troniction	own work	none (drawn for v2)
figures/p2-06-ladder.svg	own work	Troniction	own work	none (drawn for v2)
figures/p2-07-journey.svg	own work	Troniction	own work	none (drawn for v2)

IMAGE	SOURCE	AUTHOR	LICENSE	CHANGES
figures/p2-07-frame.svg	own work	Troniction	own work	none (drawn for v2)
figures/p2-07-divider.svg	own work	Troniction	own work	none (drawn for v2)
figures/p5-teach-club-room.svg	own work	Troniction	own work	none (drawn for v2)
figures/p3-fault-map.svg	own work	Troniction	own work	none (drawn for v2)
figures/p3-half-split.svg	own work	Troniction	own work	none (drawn for v2)
figures/p3-analogwrite.svg	own work	Troniction	own work	none (drawn for v2)
figures/p3-pin-map.svg	own work	Troniction	own work	none (drawn for v2)
figures/p3-spin.svg	own work	Troniction	own work	none (drawn for v2)
figures/p3-module-swap.svg	own work	Troniction	own work	none (drawn for v2)
figures/p3-monitor.svg	own work	Troniction	own work	none (drawn for v2)
figures/p3-car1-driver.svg	own work	Troniction	own work	none (drawn for v2)
figures/p3-car7-divider.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-breadboard.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-buzzer.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-caps.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-car-ready.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-driver-board.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-fault-flow.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-four-boxes.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-l298n.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-layout.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-loop.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-module-wiring.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-module.svg	own work	Troniction	own work	none (drawn for v2)

IMAGE	SOURCE	AUTHOR	LICENSE	CHANGES
figures/p1-monitor-phone.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-monitor-settings.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-motors.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-power-where.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-power.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-remote-heard.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-resistors.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-roles.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-small-parts.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-stages.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-terminal.svg	own work	Troniction	own work	none (drawn for v2)
figures/p1-uno.svg	own work	Troniction	own work	none (drawn for v2)