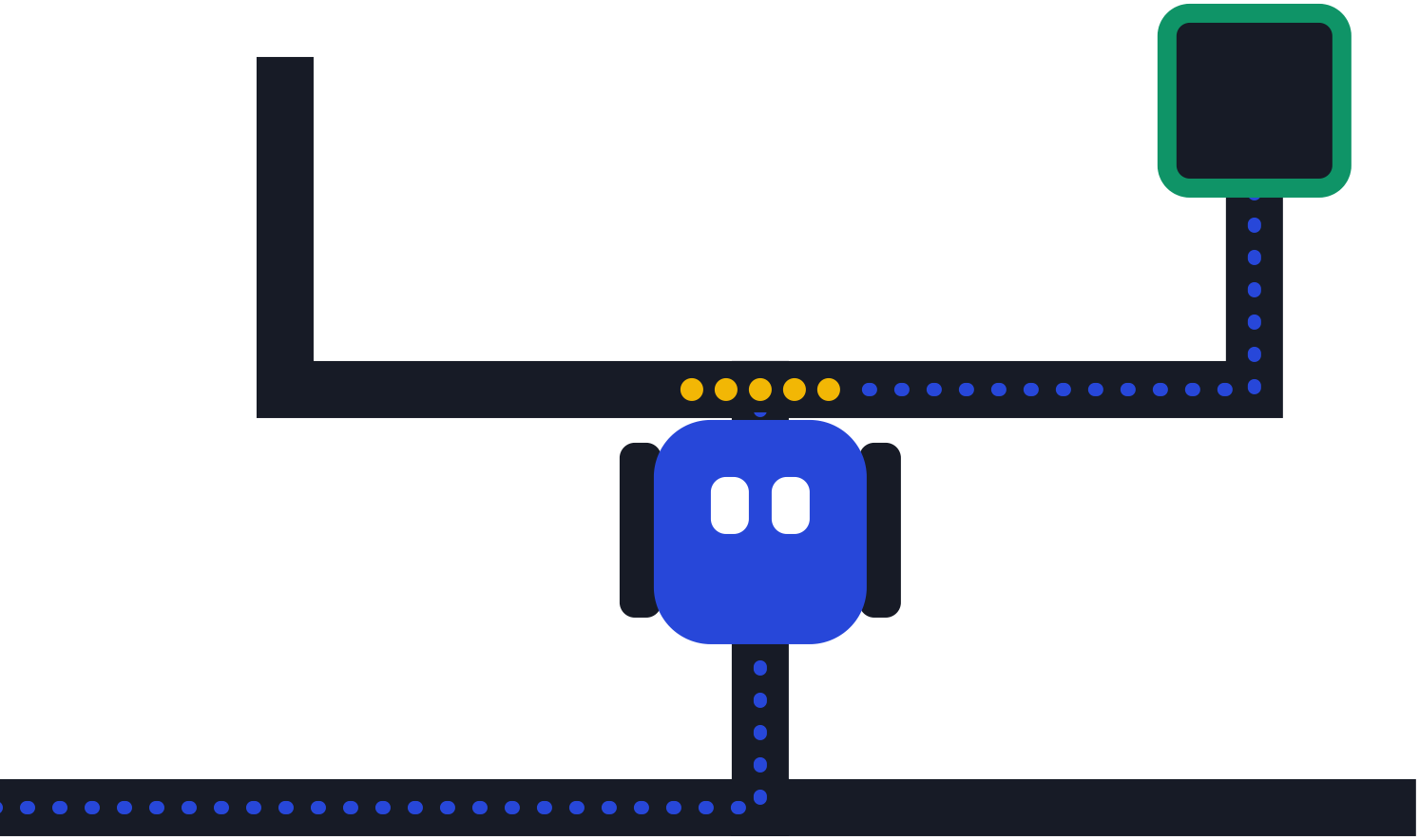


MAZE-SOLVING ROBOT · BUILD AND USER MANUAL

# Zouzou

Explores a maze it has never seen, proves which way out is shortest, drives home by itself, then races it. On tape or between walls.


1 What you are building
2 Wiring
3 Power, safely
4 Shopping list
5 First power-up
6 The phone remote
7 Running a tape maze
8 Running a wall maze
9 How it thinks
10 A show for children
11 Tuning and troubleshooting
12 Reference card



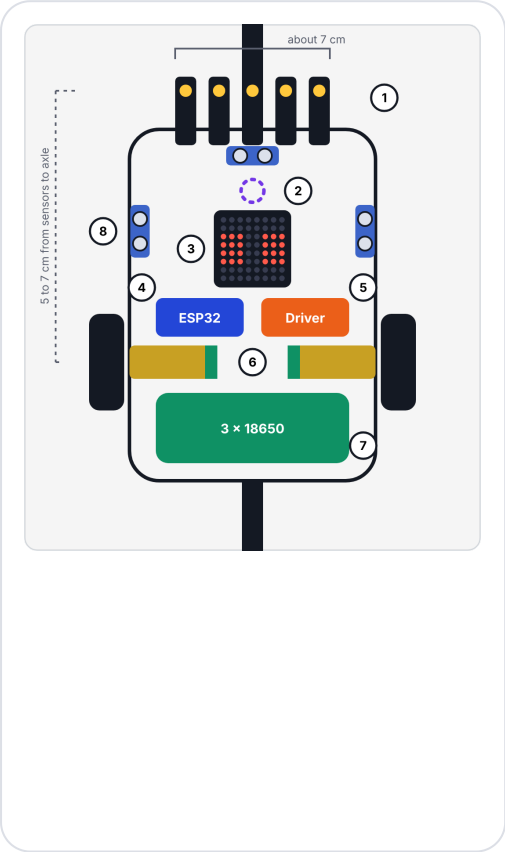
Elmehdi Essamit · Club Electro-Power · EHTP Casablanca

Firmware v3 · github.com/akeanti/zouzou-maze-robot

1 · WHAT YOU ARE BUILDING

# A two-wheeled robot with five eyes on the floor and three ears

Two geared motors drive it, a ball caster holds the nose up. Everything else is there to know where the robot is.




| # | PART | WHAT IT IS FOR |
|---|---|---|
| 1 | 5 × TCRT5000 module | See the black line. The middle three steer, the outer two notice side roads |
| 2 | Ball caster | Front support |
| 3 | MAX7219 8×8 matrix | The face |
| 4 | ESP32 DevKit, 30 pins | The brain, and the Wi-Fi for the phone |
| 5 | Motor driver | L298N, TB6612FNG or DRV8833 |
| 6 | 2 × N20 motor with encoder | Drive, and count 0.15 mm steps |
| 7 | 3 × 18650 + 5 V buck | Power |
| 8 | 3 × HC-SR04 | Hear walls and obstacles: front, left, right |
| | Passive buzzer | The voice |

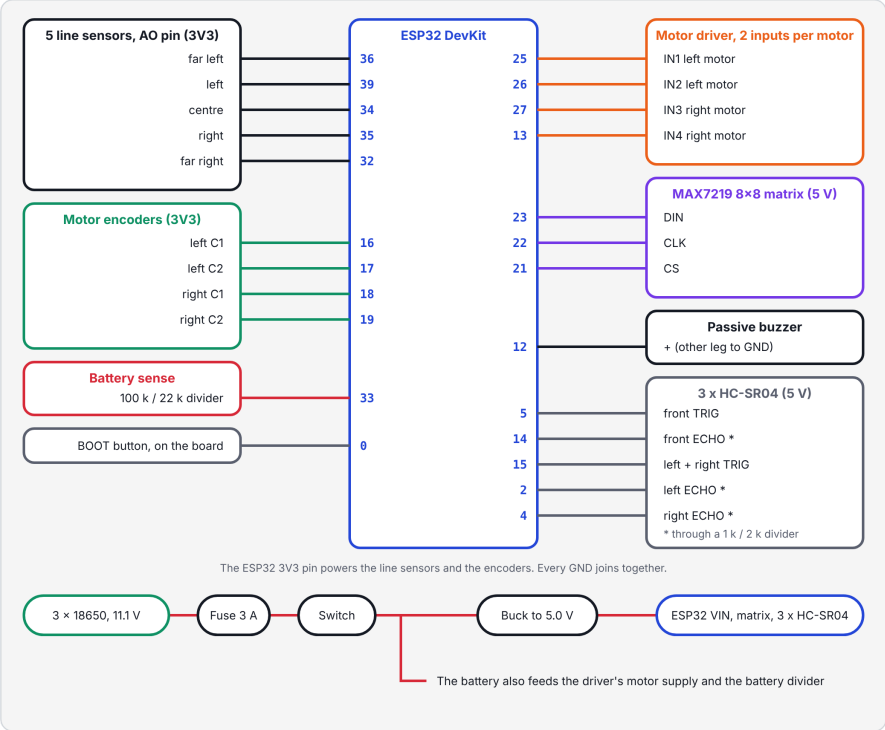
## Where things go

- **Line sensors:** one row, side by side, **3 to 8 mm above the floor**, all at the same height.
- **Sensor row to wheel axle: 5 to 7 cm.** Further forward steers more calmly; closer turns tighter.
- **Heavy things low and near the axle:** the battery sits behind it, between the wheels.
- **Front ultrasonic sensor** at the nose, looking straight ahead.
- **Side ultrasonic sensors** looking exactly sideways, 2 to 4 cm ahead of the axle, at least 3 cm above the floor.
- **Matrix** on top, facing up or forward, where the audience sees it.

> WHY FIVE SENSORS
>
> The tape is 19 mm wide and a module is 14 mm wide, so side by side the sensors sit just under one tape width apart: the line is always seen by one and usually by two. And the ESP32 has exactly six analog inputs that work with Wi-Fi on: five sensors and the battery.

# Every pin has one job

All 23 usable pins of the board are taken. If you need to change one, change it in `config.h`, the only file that describes the hardware.




**3.3 V AND 5 V**

ESP32 pins do not survive 5 V. Line sensors and encoders are powered from **3V3**. The matrix and the HC-SR04 need **5 V**, and each HC-SR04 **ECHO** returns through a divider: 1 k from ECHO to the pin, 2 k from the pin to GND.

**DO NOT MOVE PINS 2, 12, 15**

They decide how the ESP32 starts. Pin 12 must be low at power-up: the buzzer holds it there. Anything else on pin 12 can stop the board from booting.

## Motor driver, four wires, any of three boards

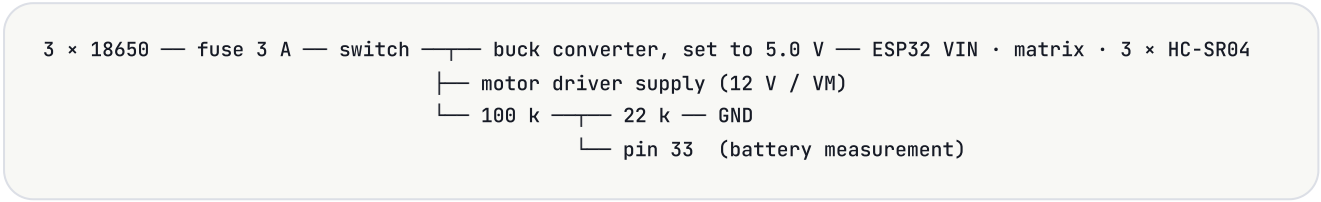
| BOARD | CONNECT 25, 26, 27, 13 TO | ALSO | 3 CELLS? |
|---|---|---|---|
| **L298N** | IN1 IN2 IN3 IN4 | leave the ENA and ENB jumpers on | yes |
| **TB6612FNG** | AIN1 AIN2 BIN1 BIN2 | PWMA, PWMB, STBY and VCC to 3V3 | yes, the best choice |
| **DRV8833** | IN1 IN2 IN3 IN4 | | **no**, it dies above 10.8 V |

Motor plug, six wires: `M1` `M2` to the driver output, `VCC` to 3V3, `GND` to GND, `C1` `C2` to the encoder pins.

3 · POWER, SAFELY

# Twelve volts in, five and three out

Three lithium cells hold enough energy to start a fire. Four habits keep that theoretical.




1. **A fuse right at the battery.**
The cells in a plain holder have no protection. A slipped screwdriver is a short circuit of tens of amps.

2. **Set the buck converter to 5.0 V before connecting anything to it.**
Out of the bag they are often set to 12 V or more. Use a multimeter.

3. **The battery never goes to VIN directly.**
The small regulator on the ESP32 board overheats at 12 V. And do not borrow the 5 V pin of the L298N: it sags when the motors start and the ESP32 resets.

4. **One ground.**
Battery minus, buck, ESP32, driver, sensors, matrix: all joined. A missing ground looks like random sensor readings.

## What the firmware does for you

### Protects the motors

N20 motors are usually rated 6 V. The firmware measures the battery and limits the PWM so they never get more on average. Set `MOTOR_VOLTS` in `config.h` to the voltage on your motors' listing (6 or 12).

Bonus: the robot drives the same on a full and a nearly empty battery.

### Protects the cells

Below **9.9 V** (3.3 V per cell) the robot shows a battery on its face and refuses to start a run. Unprotected cells discharged further than that lose capacity for good.

Take the cells out of the holder to charge them, in a proper 18650 charger.

CAPACITORS

470 to 1000 µF across the driver's motor supply and 470 µF across the 5 V rail. Without them the motors starting can reset the ESP32, and noise can scramble the matrix. A 100 nF ceramic across each motor's terminals keeps the encoders clean.

# Before you heat the soldering iron

The sensors, motors, ESP32, matrix and driver are the fun part. These are the parts that are easy to forget.

## Essential

- [ ] **Buck converter to 5 V** (MP1584EN or LM2596 module)
- [ ] **Power switch**
- [ ] **Fuse holder and 3 A fuse**, or a 3S protection board
- [ ] **18650 charger**
- [ ] **Two wheels for a 3 mm D-shaft**
- [ ] **Two N20 motor brackets**
- [ ] **Ball caster**
- [ ] **Resistors:** 100 k, 22 k, three 1 k, three 2 k
- [ ] **Capacitors:** 1000 µF 25 V, 470 µF 16 V
- [ ] **Female pin headers** to socket the ESP32 and the matrix
- [ ] **Perfboard** and **M3 standoffs**
- [ ] **Passive buzzer**
- [ ] **Black electrical tape, 19 mm**
- [ ] **White matte board** for the floor
- [ ] **Multimeter**

## Worth having

- [ ] **TB6612FNG** to replace the L298N: a quarter of the size, no heat
- [ ] **Two 100 nF ceramic capacitors**, one across each motor
- [ ] **74AHCT125 level shifter**, in case the matrix shows garbage
- [ ] **Spare 6-pin motor cables**: the pads on the encoder board tear off easily
- [ ] **Spare TCRT5000 modules**: single modules are not well matched
- [ ] **Foam board or cardboard** for wall mazes, 8 cm high

**SOLDER IN THIS ORDER**

Power first (battery, fuse, switch, buck), and test it. Then the header sockets. Then one group at a time: matrix, line sensors, driver and motors, encoders, ultrasonic sensors, buzzer. Test each group from the phone before starting the next. Finding a fault among three new wires is easy; among sixty it is an evening.

**NEVER SOLDER THE ESP32 ITSELF TO THE BOARD**

Use header sockets. If a pin is ever damaged you swap the module in ten seconds.

5 · FIRST POWER-UP

# Seven checks, in this order

Flash once, then everything is done from the phone. Each step proves one part of the wiring before the next depends on it.

```
Arduino IDE  ·  Board: ESP32 Dev Module, core esp32 3.x  ·  Libraries: none
Open firmware/zouzou_v3/zouzou_v3.ino and upload. Then join Wi-Fi ZOUZOU-ROBOT
(robot1234) and open http://192.168.4.1
```

1. **The F.**
   At power-up the matrix shows a letter F for a second. If it is sideways or mirrored, change `MATRIX_ROTATION` and `MATRIX_MIRROR` in `config.h` until it reads correctly.

2. **Battery.**
   The voltage at the top right of the remote should match your multimeter within 0.3 V. If it shows 11.1 V and never moves, the divider on pin 33 is not connected.

3. **Line sensors.**
   *Lab* tab. Slide a strip of tape under the row: each of the five bars rises in turn, left to right. A bar that stays flat is a wiring fault or a sensor mounted too high.

4. **Ultrasonic sensors.**
   Same tab. Hold a book in front of each sensor: the Left, Front and Right numbers follow in millimetres. A dash means no echo.

5. **Wheel test.**
   Wheels off the ground, press *Wheel test*. Each wheel spins forward for a moment and the remote tells you what to flip in `config.h` if a wheel or an encoder runs backwards.

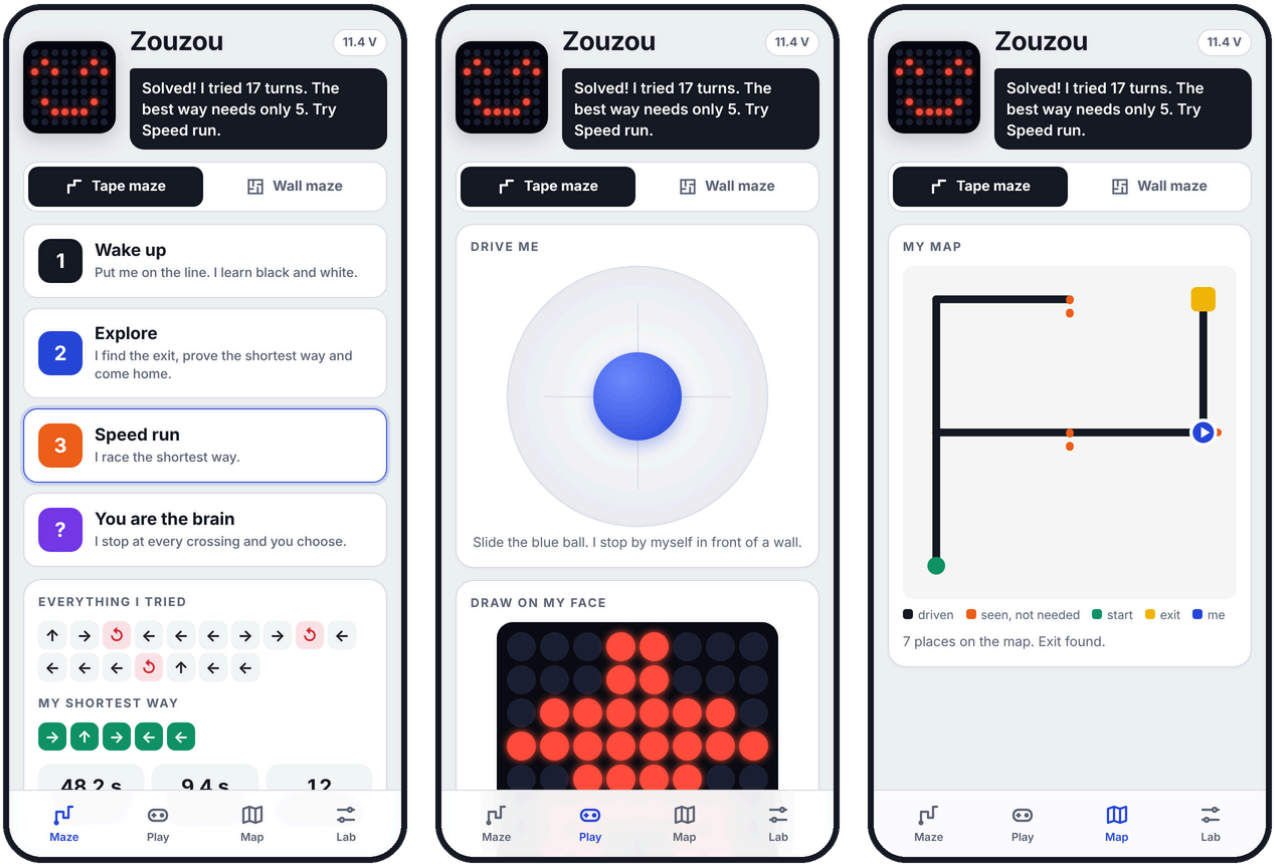
6. **The ruler.**
   Press *Zero the ruler*, push the robot 500 mm along a ruler, and change *Micrometres per tick* until *Driven* reads 500. Then turn it one full turn by hand and change *Track width* until *Turned* reads 360.

7. **Two measurements with a real ruler.**
   Enter *Line sensors to axle*. For wall mazes also *Axle to front sensor* and *Side sensors ahead of axle*. Press *Save tuning*.

> YOU NEVER NEED TO KNOW THE GEAR RATIO
>
> The starting value assumes a 1:50 gearbox and 34 mm wheels: 3.1416 × 34 mm ÷ (7 pulses × 2 edges × 50) = 0.153 mm per tick. Step 6 replaces that guess with a measurement of your own robot.

# No app. The robot is the website

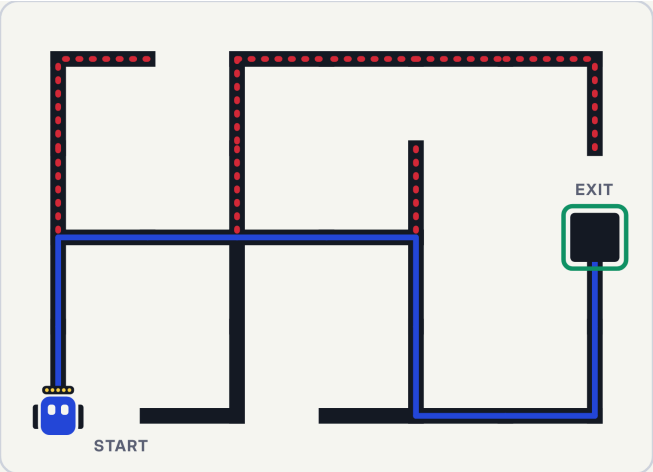
Zouzou makes its own Wi-Fi network and serves one page. Any phone works, with no internet.




| WHERE | WHAT IT DOES |
|---|---|
| **Top of every tab** | The robot's face, mirrored live. What it is saying. Battery voltage. The **Tape maze / Wall maze** switch. A red **STOP** appears whenever the robot is moving. |
| MAZE | **1 Wake up**, **2 Explore**, **3 Speed run**: the next step is outlined in blue. **You are the brain** lets the children choose at each crossing. Below: every turn tried, the shortest way, both times. |
| PLAY | Joystick, a drawing pad that is the robot's face, scrolling messages, piano, dance, sing. |
| MAP | The maze as the robot has drawn it so far. Black roads it drove, orange dots roads it saw and did not need. |
| LAB | Sensor readings, the ruler, the wheel test and every setting. Nothing here is needed during a show. |

> NO PHONE
>
> The BOOT button on the ESP32 does the essentials. Hold it: wake up. Press it: explore. Press it again once solved: speed run. Any press during a run: stop.

# Black tape on a white board




## Rules of the maze

- White **matte** floor. Shiny boards blind the sensors.
- Black tape, 19 mm. Turns are always 90°.
- At least **20 cm** between two junctions or corners.
- **Start:** the free end of a line.
- **Exit:** a solid black square, 15 × 15 cm.
- Loops are allowed.

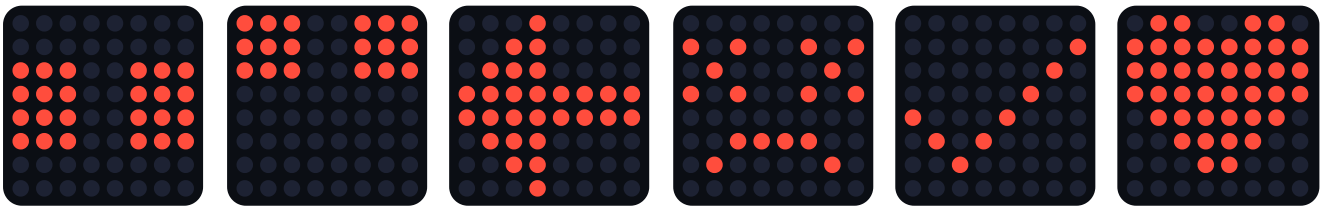
> **DRAW IT ON A GRID**
> If every road is a multiple of one square (say 25 cm), enter the square in *Lab*. Measurement errors then cannot add up.

1. **Wake up.**
   Put the robot on the line, wheels near the start end. It sweeps left and right to learn this floor's white and this tape's black. Redo it when the lighting changes.

2. **Explore.**
   3, 2, 1. It follows the line, tries roads left first, and draws its map. At the exit it shows a tick, then keeps going until no untried road could be shorter. Then it drives back to the start and turns around.

3. **Speed run.**
   It replays the shortest route, fast on the straights, slowing 12 cm before each junction it knows is coming.

## What the face is telling you



waiting | thinking | this way | dead end | exit found | finished

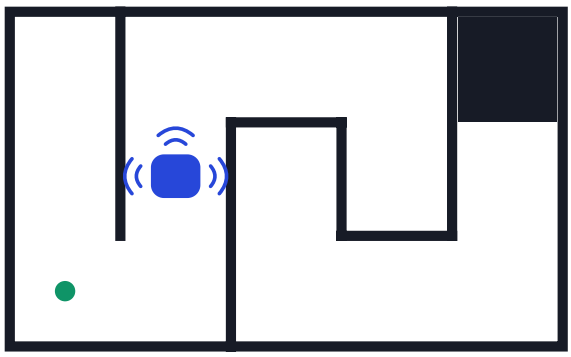
If something is on the line, the robot stops 12 cm before it, goes cross-eyed and honks until the way is clear.

# The same brain, listening instead of looking

Flip the switch to **Wall maze**. The three ultrasonic sensors replace the line; the map, the proof and the race stay the same.

## Rules of the maze

- Corridors **25 cm** wide, wall centre to wall centre, the same everywhere.
- Walls at least **8 cm** high, flat and hard: foam board, cardboard, wood.
- Right angles only, walls on the grid lines.
- **Start:** a dead-end cell. Robot in the middle, facing out.
- **Exit:** a dead-end cell with a **black floor**.
- Loops are allowed.



## How it drives

| JOB | HOW |
|---|---|
| **Stay centred** | Steers toward equal distance to the left and right walls. With one wall only, it holds the distance it learned; with none, it keeps both wheels turning the same. |
| **Notice a choice** | A side reading jumps past three quarters of a corridor, twice in a row: a wall just ended. |
| **Stop mid-cell** | Half a cell past the wall end. If there is a wall ahead, the front sensor decides instead. |
| **Turn** | 90° or 180° measured by the encoders. |
| **Finish** | The line sensors see the black floor. |

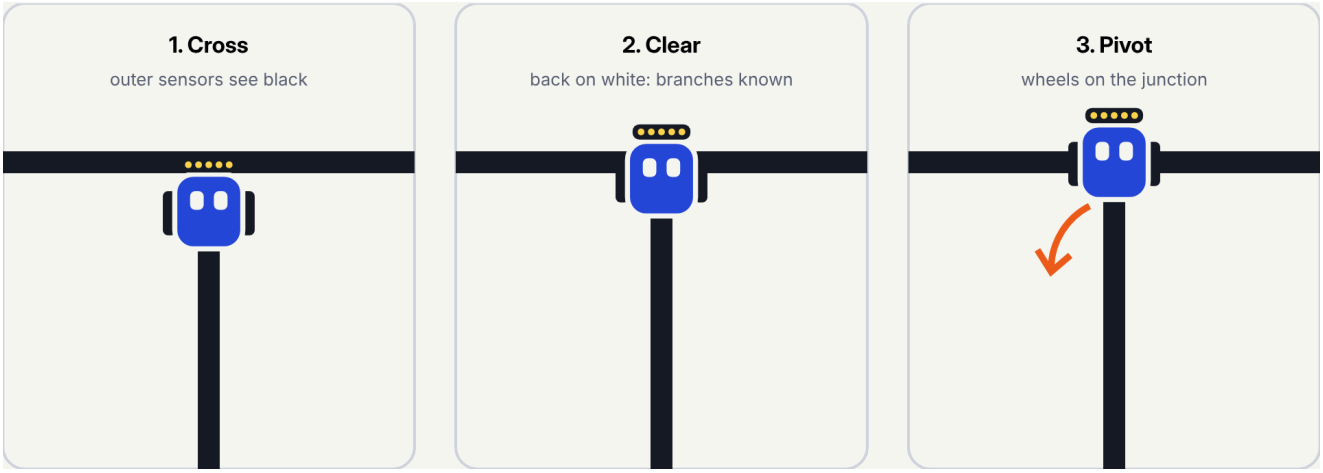
1. **Wake up**
   with the robot half on the black finish floor, so it learns black from white.
2. **Explore**
   from the middle of the start cell.
   **Speed run**
   when it is back.

NOT YET PROVEN ON THE ROBOT

Wall mode is written and checked on a PC but has not run on the finished chassis. Plan an afternoon with the tuning table on page 11.

# Turns are not enough. Position is

A robot that only remembers "left, right, left" cannot tell new ground from a place it has been. One that knows where it is can.




## 1 · Follow

Five sensors become one number, the line position, from −2000 to +2000. 250 times a second: `correction = Kp × position + Kd × change`, added to one wheel and taken from the other.

## 2 · Measure

The encoders count 0.15 mm steps. Every turn is 90°, so the robot always faces one of four ways. Length plus heading gives each junction a coordinate.

## 3 · Recognise

Arriving at a coordinate that is already on the map means "I have been here". That one test is what makes loops harmless.

## 4 · Explore

Untried road here? Take it, left first. None? Drive to the nearest junction that still has one, using the map as a shortcut.

## 5 · Know when to stop

Roads run along two axes, so from any point the exit is never closer than `|dx| + |dy|`. If reaching a junction plus that is not better than the best route, nothing behind it can help. When no road is left, the route is **proven** shortest.

## 6 · Choose and race

Dijkstra's algorithm, where a turn costs as much as 80 mm of road. The route is stored with every road length, so the robot knows when to brake.

> CHECKED 3000 TIMES
>
> The solver has no hardware in it, so it runs on a PC. A simulated robot with realistic measurement errors drove 3000 random mazes with loops. Every stored route was the true shortest one.

# Five acts, about fifteen minutes

The robot does not need the theatre to solve the maze. The children need it to see what is happening.

| | ACT | PRESS | THEY SEE | THEY LEARN |
|---|---|---|---|---|
| 1 | **Meet the robot** | *Say your name*, then *Play* | It scrolls its name. A child draws a heart and it appears on the face | A screen is dots someone switches on |
| 2 | **You are the brain** | *You are the brain* | It stops at each crossing, arrows blink, they shout a direction | Guessing is slow. Is there a rule? |
| 3 | **The robot's turn** | *Explore*, *Map* on a big screen | Dead ends, dizzy eyes, and the maze drawing itself | A robot cannot see the maze. It must go and find out, and remember |
| 4 | **Did it learn?** | nothing | It drives home by itself | It is using what it remembered |
| 5 | **The race** | *Speed run* | Straight to the exit. Fireworks, confetti | Remembering made it fast. Compare the times |

> THE QUESTION THAT MAKES ACT 3 LAND
>
> After act 2, ask: "What would you tell the robot to *always* do?" Someone says "always go left". That is the left-hand rule, and they invented it.

## For the queue

- **Drive me:** a joystick. It will not drive forward into a wall, so let them try.
- **Draw on my face:** stamps for a fast start: heart, star, ghost, cat, rocket.
- **Make me talk:** their name scrolls across the face.
- **Piano:** eight keys. *Frère Jacques* fits.

## Before they arrive

- ☐ Battery above 11 V
- ☐ Maze taped flat, exit in place
- ☐ *Wake up* done on this floor, in this light
- ☐ One full *Explore* and *Speed run* rehearsed

# When it does not do what you expect

## Tape maze

| WHAT HAPPENS | CHANGE THIS, IN LAB |
|---|---|
| Wobbles left and right along a straight | lower **Kp**, or raise **Kd** |
| Slow to return to the line, cuts corners | raise **Kp** |
| Fine when slow, shakes when fast | raise **Kd**; move the sensor row further forward |
| Turns end beside the line | lower **Turn speed** |
| Pivots happen before or after the junction | correct **Line sensors to axle** |
| "I cannot see the line well" | sensors too high or too low (3 to 8 mm), or the floor is shiny |
| "I lost my place on the map" | redo the ruler; put junctions further apart; enter the grid square |
| Takes the exit for a junction, or the reverse | the exit square must be at least 15 cm; `FINISH_MM` in `config.h` |

## Wall maze

| WHAT HAPPENS | CHANGE THIS, IN LAB |
|---|---|
| Zigzags down the corridor | lower **Wall steering strength** |
| Drifts onto a wall | raise it |
| Stops short of the cell middle at an opening / overshoots | raise / lower **Side sensors ahead of axle** |
| Stops too close to or too far from the wall ahead | correct **Axle to front sensor** |
| Turns too far or not far enough | correct **Track width** with the ruler |
| Misses openings | lower **Speed between walls** |

## Electrical

| WHAT HAPPENS | LOOK AT |
|---|---|
| Resets when the motors start | capacitor on the motor supply; ESP32 fed from the buck, not the L298N |
| Matrix blank, all on, or garbage | 5 V and GND at the matrix; shorter wires; a 74AHCT125 on DIN, CLK, CS |
| Does not boot with everything plugged in | something other than the buzzer on pin 12 |
| Upload fails | unplug the left ultrasonic ECHO from pin 2, or hold BOOT while uploading |
| Encoder counts jump while standing still | 100 nF across the motor terminals; encoder VCC on 3V3 |

# Everything on one page

## Pins

| | |
|---|---|
| Line sensors, left to right | `36 39 34 35 32` |
| Battery divider | `33` |
| Left motor IN1, IN2 | `25 26` |
| Right motor IN3, IN4 | `27 13` |
| Left encoder C1, C2 | `16 17` |
| Right encoder C1, C2 | `18 19` |
| Matrix DIN, CLK, CS | `23 22 21` |
| Front sonar TRIG, ECHO | `5 14` |
| Side sonars TRIG | `15` |
| Left ECHO, right ECHO | `2 4` |
| Buzzer | `12` |
| Button (BOOT) | `0` |

## Connect

| | |
|---|---|
| Wi-Fi | `ZOUZOU-ROBOT` |
| Password | `robot1234` |
| Page | `192.168.4.1` |
| Serial monitor | `115200` |

## Starting values

| | |
|---|---|
| Kp / Kd | `0.07 / 0.90` |
| Explore / speed-run / turn speed | `140 / 200 / 140` |
| Line sensors to axle | `60 mm` |
| Micrometres per tick | `153` |
| Track width | `95 mm` |
| Corridor width | `250 mm` |
| Axle to front sensor | `70 mm` |
| Side sensors ahead of axle | `30 mm` |
| Speed between walls | `110` |
| Wall steering strength | `0.60` |
| Low-battery stop | `9.9 V` |

## BOOT button

| | |
|---|---|
| Hold | wake up (calibrate) |
| Press | explore; speed run once solved |
| Press during a run | stop |

## In `config.h`

| | |
|---|---|
| `HAS_ENCODERS` | false: timed moves, turn list only |
| `HAS_SONAR` | false: no ultrasonic sensors |
| `MOTOR_VOLTS` | rated voltage of your motors |
| `DRIVER_DROP_VOLTS` | 1.8 for L298N, 0.2 otherwise |
| `*_SIGN` | flip a wheel or an encoder |

MORE

Design journal, algorithm notes, hardware guide and the simulator:
`github.com/akeanti/zouzou-maze-robot`