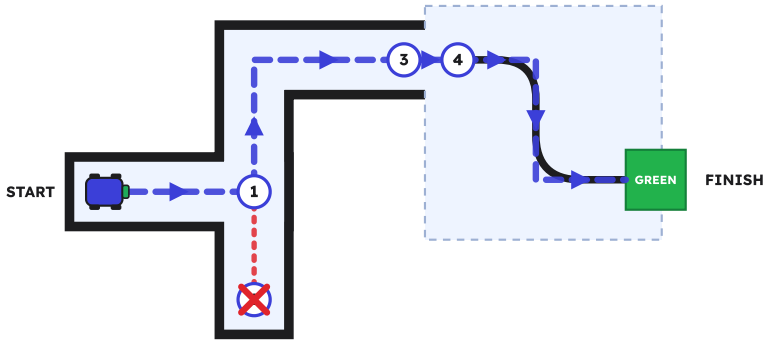
MISSION 7 · SOLUTION

The Grand Maze

There is more than one right answer. Yours may look different — that is fine if it works.

THE ROUTE



What the car senses

1. T-junction: wall ahead, right and left open → right first: into the dead end.
2. Dead end: everything blocked → turn around.
3. Corner: wall ahead, right open → turn right along the exit corridor.
4. Doorway: no wall ahead, so the car drives on until the belly sensor sees the **black start line** → switch to line following.
5. Follow the tape to the green tile. Stop. Sound.

FIRST

Make these My Blocks

turn right
motor D go to position 60° → move forward for 20 cm → motor D go to position 0° · Start values — tune until the car ends up facing sideways.

turn left
motor D go to position -60° → move forward for 20 cm → motor D go to position 0° · Mirror of turn right.

turn around
turn left → turn left · Two 90° turns make a U-turn inside the lane. Needs room in front to swing.

look around
motor C go to position 90° → wait 0.2 s → set right_open to distance sensor B is farther than 30 cm → motor C go to position -90° → wait 0.3 s → set left_open to distance sensor B is farther than 30 cm → motor C go to position 0° · Look right, look left, face ahead again. 30 cm = about one lane width — measure yours.

celebrate
set hub light to green → play beep 80 for 0.5 s → play beep 100 for 0.5 s · Sound + light on the tile.

THE PROGRAM

Word Blocks

- **when program starts**
- **set movement motors to A + E**
  use the ports from YOUR car
- **set movement speed to 30 %**
- **repeat until color sensor F sees black**
  - **start moving forward**
  - **wait until distance sensor B closer than 20 cm or color sensor F sees black**
    a wall ahead, or the finish
  - **stop moving**
  - **if not color sensor F sees black**
    - **look around**
      turret scans right then left, sets right_open and left_open
    - **if right_open**
      the right-hand rule: right first
      - **turn right**
        My Block
    - **else if left_open**
      - **turn left**
        My Block
    - **else**
      right and left and ahead are all blocked: a dead end
      - **turn around**
        My Block: two turns in a row
- **set movement speed to 25 %**
  slower for the line
- **start moving forward**
- **repeat until color sensor F sees green**
  - **motor D go to position (reflected light − 50) × 0.6**
    the Mission 6 rule
- **stop moving**
- **motor D go to position 0**
- **celebrate**
  My Block: sound + light

– Same thing in Python

```
from hub import port, sound, light_matrix
import runloop, motor, motor_pair, distance_sensor, color_sensor, color

DRIVE      = motor_pair.PAIR_1
STEER      = port.D            # large motor, steers the back
TURRET     = port.C            # large motor that swings the distance sensor
EYES       = port.B            # distance sensor (rides on the turret)
BELLY      = port.F            # color sensor pointing down
STOP_MM    = 200               # wall closer than 20 cm ahead = stop and look
OPEN_MM    = 300               # farther than this = OPEN (about one lane width - measure yours!)
LOOK_RIGHT = 90                # turret position that looks right (tune for your gearing)
LOOK_LEFT  = -90               # turret position that looks left
TARGET_REFLECT = 50          # halfway between white and black readings
```

```
K_LINE     = 0.6              # line-following strength
DEG_PER_CM = 20               # wheel degrees per cm: tune
TURN_CM    = 20               # roll while steering: tune

def wall_closer_than(mm):
    d = distance_sensor.distance(EYES)
    return 0 < d < mm

def on_color(c):
    return color_sensor.color(BELLY) == c

async def drive_cm(cm, speed=300):
    await motor_pair.move_for_degrees(DRIVE, int(cm * DEG_PER_CM), 0, velocity=speed)

async def turn_right():
    await motor.run_to_absolute_position(STEER, 60, 500)
    await drive_cm(TURN_CM)
    await motor.run_to_absolute_position(STEER, 0, 500)

async def turn_left():
    await motor.run_to_absolute_position(STEER, -60, 500)
    await drive_cm(TURN_CM)
    await motor.run_to_absolute_position(STEER, 0, 500)

async def turn_around():
    await turn_left()          # two 90-degree turns = a U-turn inside the lane
    await turn_left()

async def celebrate():
    light_matrix.show_image(light_matrix.IMAGE_HAPPY)
    await sound.beep(880, 300)
    await sound.beep(1175, 500)

async def look_open(angle):
    """Swing the turret to 'angle' and say whether that way is OPEN."""
    await motor.run_to_absolute_position(TURRET, angle, 700)
    await runloop.sleep_ms(200)                 # let the sensor settle
    d = distance_sensor.distance(EYES)          # -1 = nothing seen = wide open
    return d == -1 or d > OPEN_MM

async def look_around():
    right_open = await look_open(LOOK_RIGHT)
    left_open = await look_open(LOOK_LEFT)
    await motor.run_to_absolute_position(TURRET, 0, 700)   # face ahead again
    return right_open, left_open

async def drive_until_wall_or(stop_color):
    """Drive until a wall is close ahead, or stop_color is under the car."""
    motor_pair.move(DRIVE, 0, velocity=300)
    await runloop.until(lambda: wall_closer_than(STOP_MM) or on_color(stop_color))
    motor_pair.stop(DRIVE)

async def hug_right_wall_until(stop_color):
    """Stop-and-look maze solver (right-hand rule). Returns (route, dead_ends)."""
    route = []
    dead_ends = 0
    while not on_color(stop_color):
        await drive_until_wall_or(stop_color)
        if on_color(stop_color):
            break
        right_open, left_open = await look_around()
        if right_open:
            await turn_right()
            route.append("R")
        elif left_open:
            await turn_left()
            route.append("L")
        else:                               # all sides blocked: a dead end
            await turn_around()
            dead_ends += 1
            route.append("U")
    return route, dead_ends

async def follow_line_until_green():
    motor_pair.move(DRIVE, 0, velocity=250)
    while not on_color(color.GREEN):
        error = color_sensor.reflection(BELLY) - TARGET_REFLECT
        angle = max(-60, min(60, int(error * K_LINE)))
        await motor.run_to_absolute_position(STEER, angle, 1000)
    motor_pair.stop(DRIVE)
    await motor.run_to_absolute_position(STEER, 0, 800)

async def main():
    motor_pair.pair(DRIVE, port.A, port.E)      # your drive motor ports
    await hug_right_wall_until(color.BLACK)
    await follow_line_until_green()
```

```
    await celebrate()

runloop.run(main())
```

Starting values only. Steering angle, roll distances, the OPEN limit and the line target depend on *your* car and maze. Measure, then tune one at a time. I have not run these on a car — expect to adjust.

LEVEL UP

Ready for more?

- Time your run. Can you beat 45 seconds?
- Build the maze mirrored, with the dead end on the other side.
- Design your own Mission 8 and swap with a friend.

Keep going: Hints → `mission-7-hints.pdf` | All missions → `index.pdf`
The same content is in the matching .html files.