PRESENTATION 10 · LEARNING WITH C

# Avoid the traps

See the warning band, back away, and find the way back to FINISH.

**TheBotCup 2026 · AiBotCup Simulator**



RECAP · COLOUR SENSORS

# Every colour on the floor means something

**Metals**

collect them (P6–P8)

**Green**

deposit there (P9)

**Pink**

FINISH or a target

**Red**

a trap: stay away! (today)

Today the colour sensors are used for **safety**, not for collecting.

PART 1

# The mission: Session 10

## START to FINISH, traps in the way

SESSION 10 · AVOID THE TRAPS

# Reach FINISH without touching a trap

| What happens | Points |
|---|---|
| Pass trap 1 without touching | +50 |
| Pass trap 2 without touching | +50 |
| Touch a trap | −20 |
| Reach FINISH | +100 |

Time limit **4:00**. Best score: **200**.

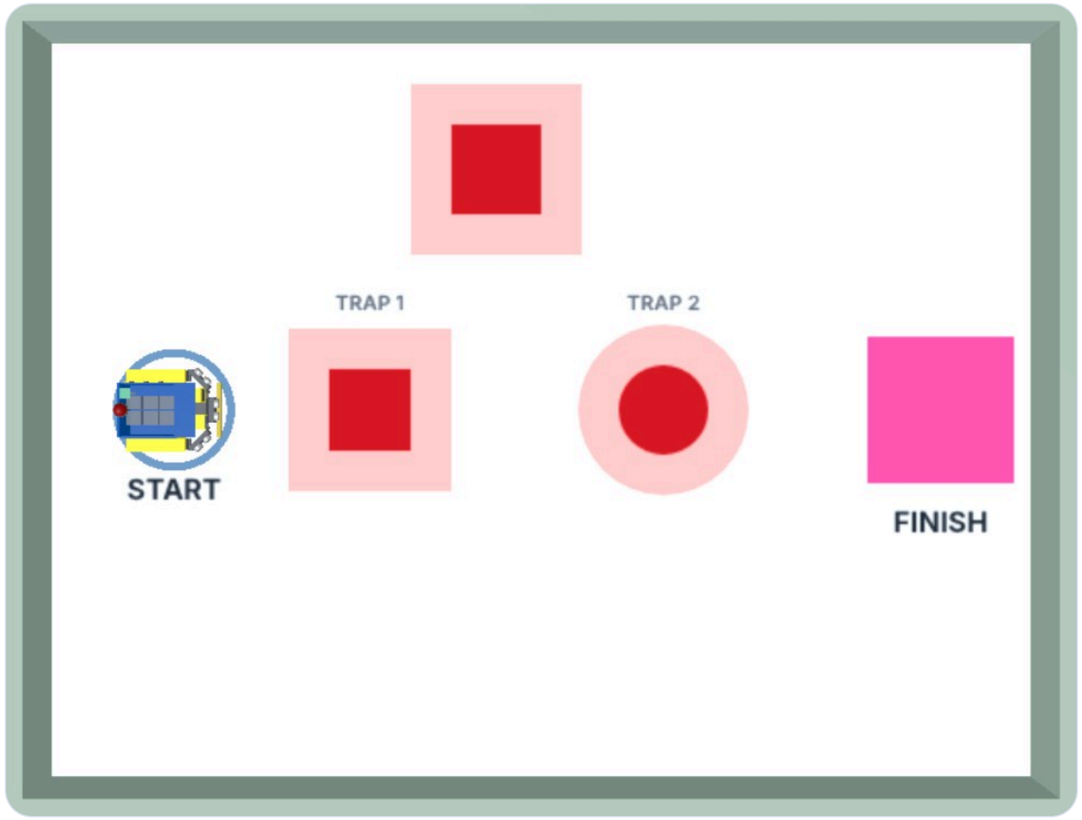
10

**Session 10 · Avoid the traps**

Drive from START to the pink FINISH pad. Two unsafe areas (traps) are on the straight route and a third is nearby: go around them.

Each of the two traps passed without touching a trap: +50. Touching a trap: −20. Reach FINISH: +100. Time limit 4:00.

Not done yet




THE FIELD

# Straight ahead is blocked twice

**START** on the left, facing **FINISH** on the right.

**Trap 1** (square) and **trap 2** (round) sit right on that line.

A **third trap** waits above: going round the top is risky.




WHAT IS A TRAP?

# A red danger zone with a warning band

The **dark red** middle is the trap. A colour sensor on it: **−20**.

Around it is a **10 cm light-red band**. Driving on the band is safe: it is a warning.

React to the **band**, and the robot never reaches the trap.

COLOUR SENSORS · LIGHT RED

# The band in numbers

| | R | G | B |
|---|---|---|---|
| Floor | about 226 | about 232 | about 238 |
| **Warning band** | 250 | 175 | 175 |
| Trap | 185 | 20 | 30 |
| **Range we use** | 230 – 255 | 145 – 205 | 145 – 205 |

Red as high as the floor, but green and blue clearly lower. The range comes from the simulator’s own tip.

PART 2

# The plan

## Back off, then get back on course

THE PLAN

# Three rules, most important first

1. A sensor sees the **warning band** → **back away** and turn away from it.
2. Off course → **curve back** towards FINISH (compass and position).
3. Anything else → **drive forward**.

RULE 2 · TWO SENSORS WE KNOW

# Which way? How high?

## Compass

**270** = facing FINISH (east). Below 270 = turned right, towards the bottom. More in Presentation 11.

## PositionY

START and FINISH are at about **Y = 90**. Below 80 = too low; above 100 = too high.



PART 3

# Step by step in C

## Six branches, at most 20 lines

STEP 1 · THE PLAN

# Safety, then course, then forward

| Branch | Condition | Wheels | Duration |
|---|---|---|---|
| **1 WarnLeft** | isWarn(left): R ≥ 230, G and B 145–205 | −1 / −4 | 500 ms |
| **2 WarnRight** | isWarn(right) | −4 / −1 | 500 ms |
| **3 Low** | PositionY <= 79 && Compass 240–299 | 3 / 4.5 | 60 ms |
| **4 High** | PositionY >= 101 && Compass 241–300 | 4.5 / 3 | 60 ms |
| **5 TooRight** | Compass 90–262 | 3.5 / 4.5 | 60 ms |
| **6 Forward** | else | 4.5 / 4.5 | 60 ms |

STEP 2 · LINES 3–13

# Back away, nose turning away from the band

```
3  int isWarn(int R, int G, int B) {
4      return R >= 230 && G >= 145 && G <= 205 && B >= 145 && B <= 205;
5  }
6
7  void Game1() {
8      if (isWarn(CSLeft_R, CSLeft_G, CSLeft_B)) {                    // WarnLeft
9          WheelLeft = -1; WheelRight = -4;
10         Duration = 500;
11     } else if (isWarn(CSRight_R, CSRight_G, CSRight_B)) {          // WarnRight
12         WheelLeft = -4; WheelRight = -1;
13         Duration = 500;
```

**isWarn**: R 230–255, G and B 145–205.

**WarnLeft**: −1 / −4, nose swings right.

**WarnRight**: −4 / −1, nose swings left.

STEP 3 · LINES 14–17

# Too low or too high: aim up or down a bit

```
14      } else if (PositionY <= 79 && Compass >= 240 && Compass <= 299) {   // Low
15          WheelLeft = 3; WheelRight = 4.5;
16      } else if (PositionY >= 101 && Compass >= 241 && Compass <= 300) {  // High
17          WheelLeft = 4.5; WheelRight = 3;
```

**Low**: Y below 80 && facing roughly east → curve left.

**High**: Y above 100 → curve right.

They stop at **299** / **241**, so the robot heads diagonally, not back into the traps.

STEP 4 · LINES 18–23

# Turned too far right? Curve back.

```
18      } else if (Compass >= 90 && Compass <= 262) {                    // TooRight
19          WheelLeft = 3.5; WheelRight = 4.5;
20      } else {                                                         // Forward
21          WheelLeft = 4.5; WheelRight = 4.5;
22      }
23  }
```

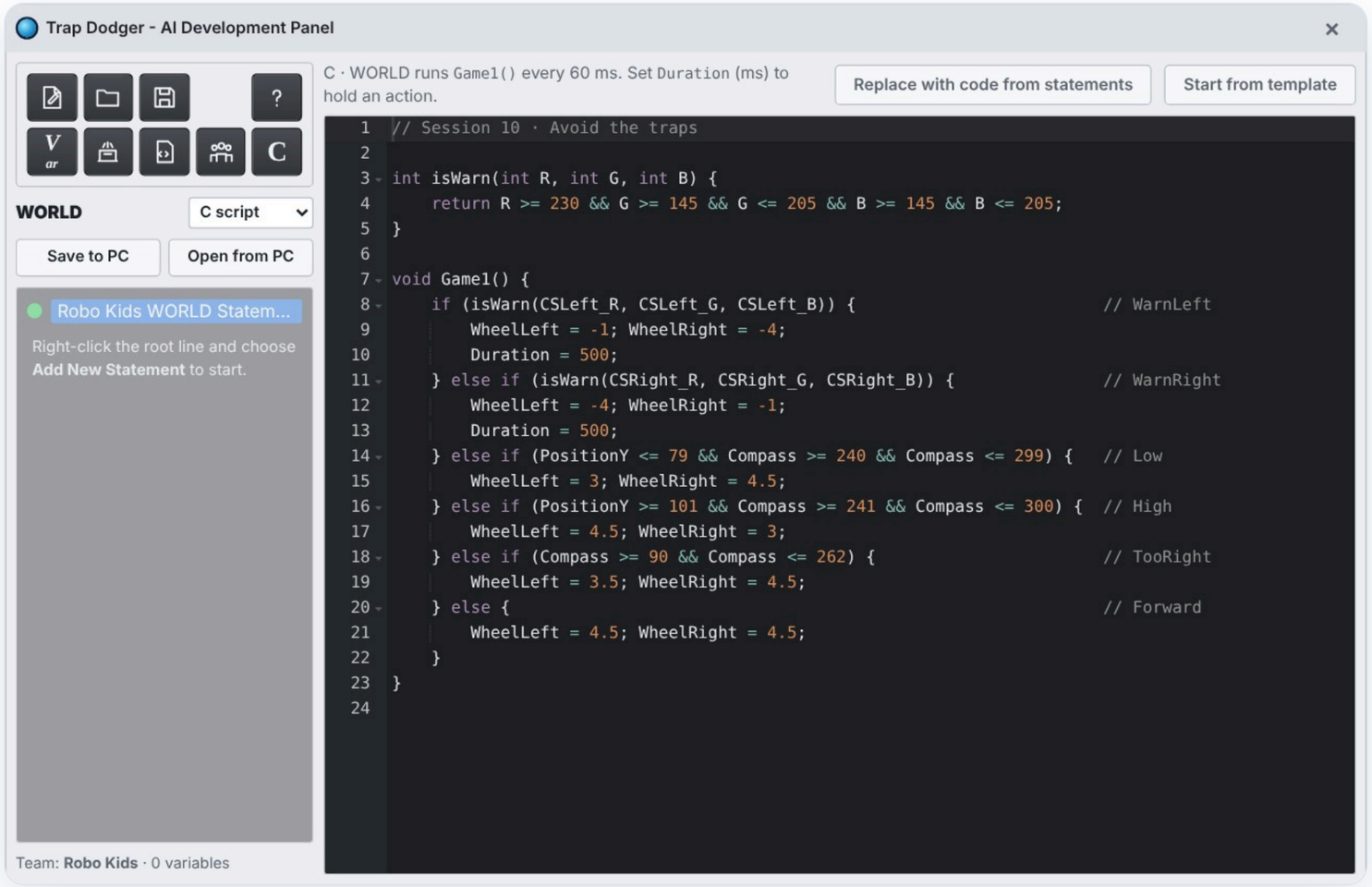
**TooRight**: Compass 90–262 → curve left, 3.5 / 4.5.

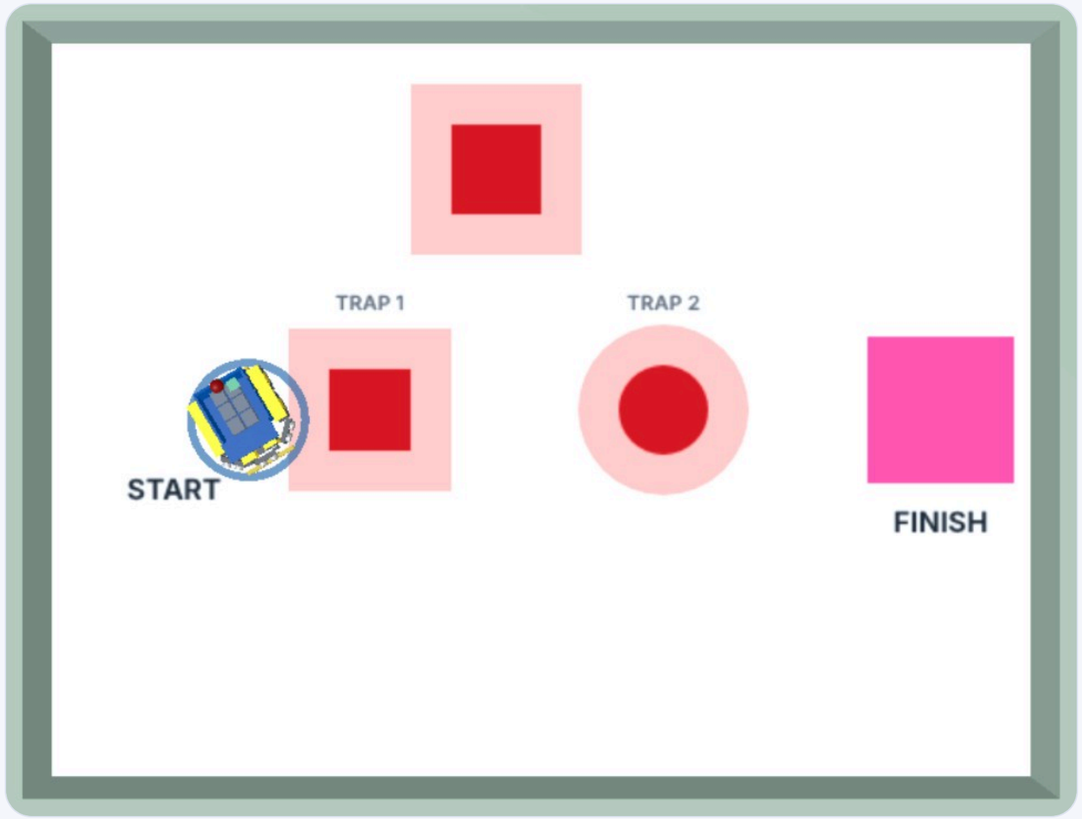
**Forward**: the else, always last.

STEP 5 · BUILD AND LOAD

# The whole program, built

1. WORLD menu: **C script**.
2. Type the program, name it **Trap Dodger**.
3. Press **Build**, then **OK**.
4. Press **Load** on the BLUE side.






STEP 6 · RUN · THE BAND

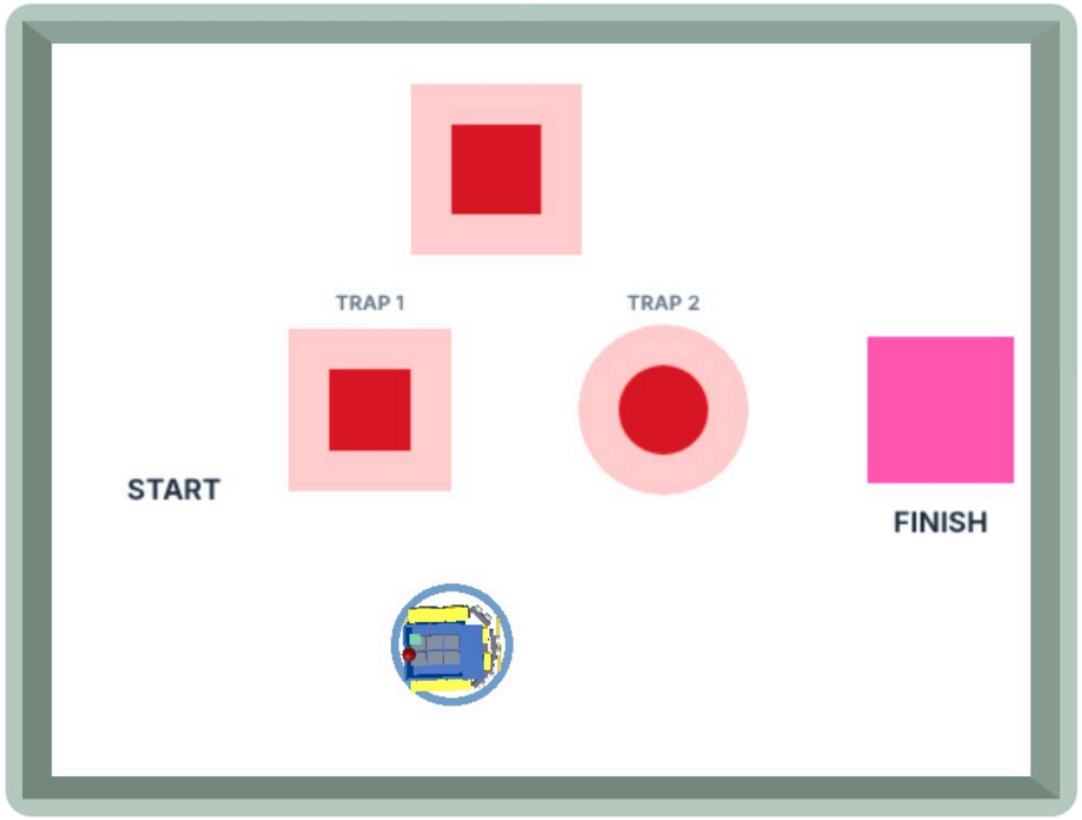
# Band ahead: WarnLeft

After about 1 s the sensors reach the band of trap 1.

**WarnLeft** backs away, and again if needed.

At 2.8 s the compass reads **196**: turned towards the bottom.

The trap itself is **never touched**.



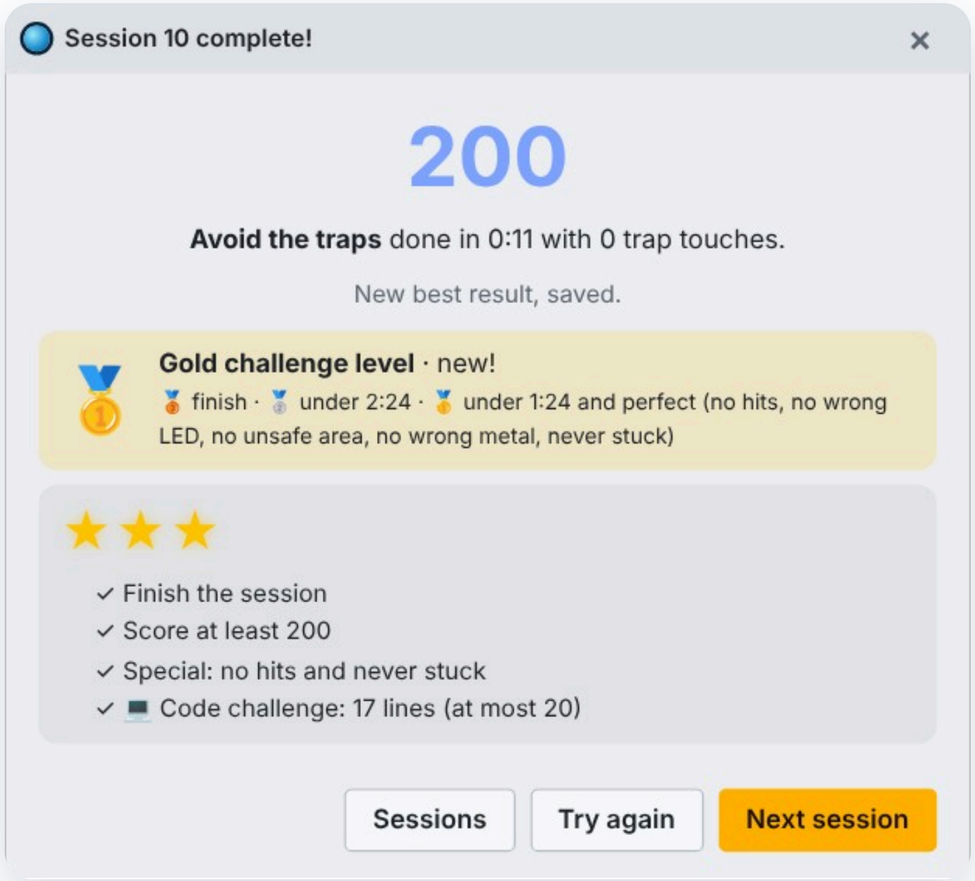

STEP 7 · RUN · BACK ON COURSE

# TooRight, then Low, then Forward

TooRight first curves the robot back towards east.

At 7.3 s: PositionY **34**, Compass **287**: **Low** curves it up.

Then Forward runs diagonally up to FINISH, below trap 2.



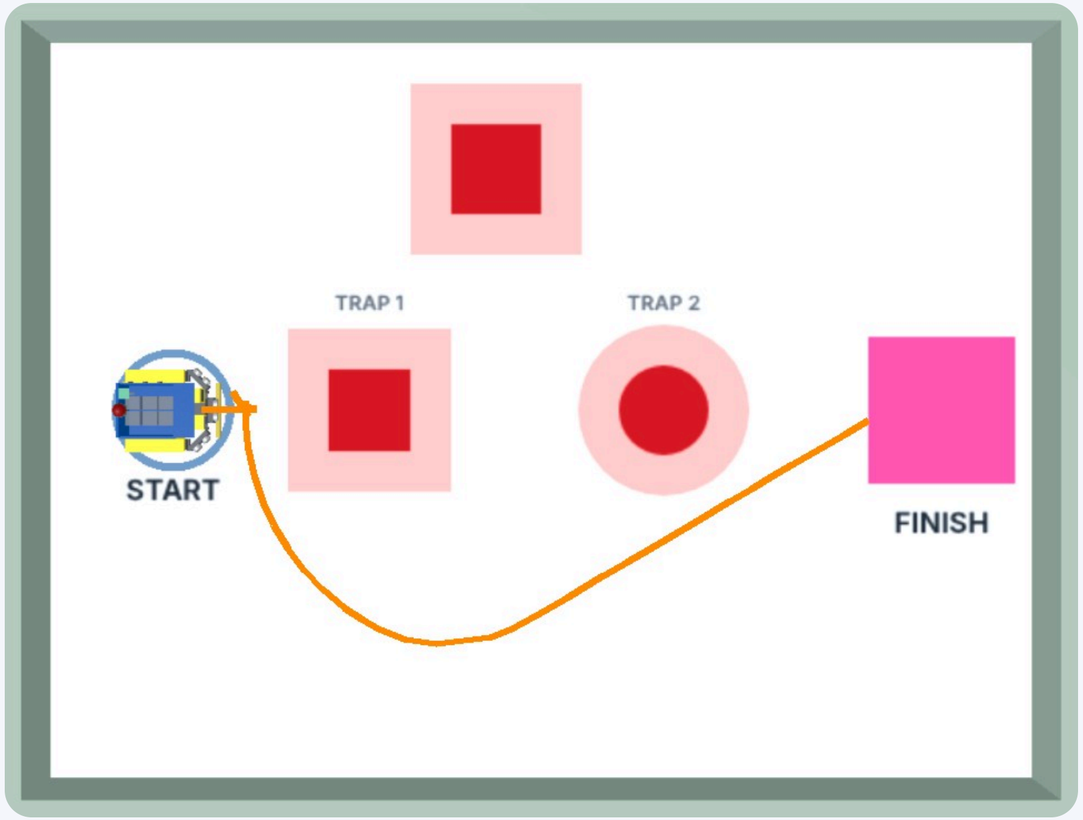

RESULT · TESTED

# 200 points in 0:11 · Gold

**Score** · trap 1 +50 · trap 2 +50 · FINISH +100. 0 trap touches.

**Medals** · Silver: under 2:24 · Gold: under 1:24 and perfect.

**Code challenge** · 17 lines, at most 20.




THE PATH · RECORDED

# Round the bottom, then a diagonal

One bounce off the band of trap 1. Then the course rules did the rest.

React to the **warning**, not the danger.

DID NOT WORK?

# Check first, then tune carefully

| What you see | What to do |
|---|---|
| Build error with a line number | ; at the end of lines, { } matched |
| Drives into the trap (−20) | isWarn ranges: R ≥ 230, G and B 145–205 |
| Spins on the band for a long time | Back away (both wheels negative), don’t spin on the spot |
| Ends in a corner | Low and TooRight missing, or Compass ranges wrong |
| Misses FINISH by a little | Check Low: PositionY <= 79 && Compass 240–299 |
| Old program still on the robot | Build again, then Load on BLUE |

NEXT TIME

# Presentation 11 · Compass targets

What a compass is, how it works, and how to turn to **any heading** in C.