PRESENTATION 6 · LEARNING WITH C

Iron only

Light sensors, colour sensors, and a robot that finds black iron on the floor and collects it.

TheBotCup 2026 · AiBotCup Simulator



RECAP

From looking ahead to looking down

Sessions 1–2

Time sensor

Drive and turn at the right second.

Sessions 3–5

Ultrasonic

Measure distance to walls; follow a wall through a maze.

Session 6 · today

Colour sensors

Look at the floor and find what lies on it.

Same C rules as before: **Game1()** runs every 60 ms, and the **first true** branch of the if chain runs.

PART 1

The mission: Session 6

Collect 6 iron objects

SESSION 6 · IRON ONLY

Find the black ones and collect them

Each iron object: **+10**

All 6 iron: **+100** more · total **160**

Time limit **5:00**

6

Session 6 · Iron only

Collect 6 IRON objects (the black ones). Only iron is on the field.

Each iron object: +10. 6 iron: +100. Time limit 5:00.

Not done yet



THE FIELD

Open floor, nine iron, one block

240 × 180 cm, walls all round.

9 iron on the floor: you need any **6**.

A **block** in the middle: drive round it.

START bottom centre, facing up.

PART 2

Light sensors and colour sensors

How a robot sees the floor

WHAT IS A LIGHT SENSOR?

It measures how much light comes back

1 Shine

A small LED shines on the floor.

2 Bounce

White reflects a lot. Black swallows most light.

3 One number

A light detector gives one value: high = light, low = dark.

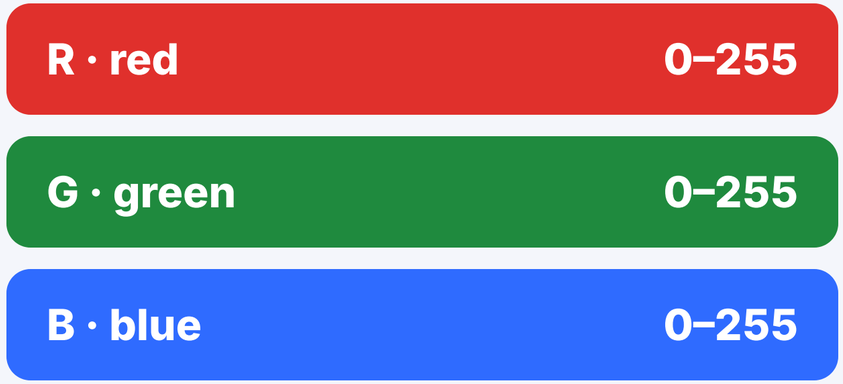
It answers one question: **how bright is the floor here?** Great for black lines and dark objects.

WHAT IS A COLOUR SENSOR?

It measures red, green and blue separately

A white LED lights the floor. The detector looks through **three colour filters**, so it measures three amounts of light.

Each one is a number from **0** (none) to **255** (a lot). Together they make the colour.



Like your eyes and every screen: all colours are mixes of **red, green and blue**.

THE DIFFERENCE

Light sensor or colour sensor?

	Light sensor	Colour sensor
Measures	How much light	How much red, green, blue
Gives	1 number	3 numbers (R, G, B)
Can tell	Light from dark	Light from dark and one colour from another
Good for	Black lines, dark objects	Sorting metals, finding water, traps
Simpler?	Yes: one check	More checks, more information

WHY COLOUR MATTERS

Same brightness, different colours



Gold

R 220 · G 168 · B 34

Brightness ≈ **141**



Water pond

R 0 · G 170 · B 240

Brightness ≈ **137**

A light sensor sees **141 and 137: almost the same**. A colour sensor sees lots of red in gold and **no red** in water.

Brightness here = (R + G + B) ÷ 3

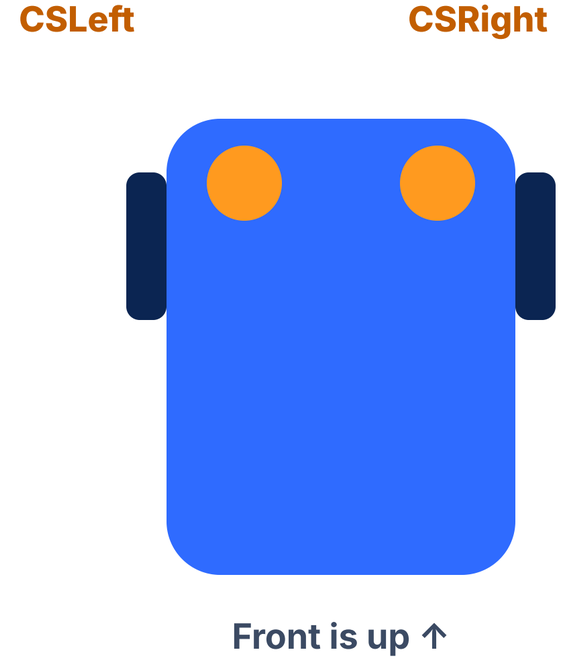
IN THE SIMULATOR

Two colour sensors under the front

One on the **left**, one on the **right**, 10 cm apart, just behind the front bumper, looking down.

Six values: **CSLeft_R, CSLeft_G, CSLeft_B** and **CSRight_R, CSRight_G, CSRight_B**.

Each 0–255, with a little noise (±3), like a real sensor.



THE COLOURS

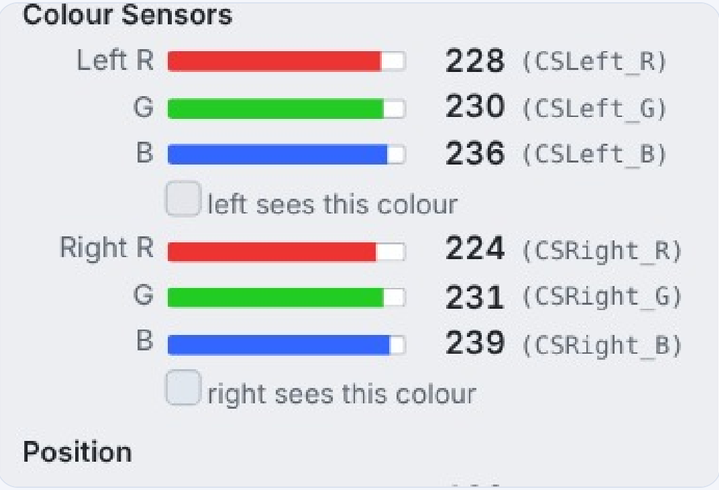
What the sensors see on the field

Today only the first two rows matter: **floor** is high in all three, **iron** is low in all three.

Surface	R	G	B	Bright
Floor	226	232	238	232
Iron · today	24	24	26	25
Silver	110	112	118	113
Gold	220	168	34	141
Water pond	0	170	240	137
Deposit area	30	160	70	87
Trap	185	20	30	78
Swamp	140	95	55	97

LIVE READINGS · SENSORS PANEL

Floor or iron? Look at the numbers



At START: both sensors on the floor, all values about **230**.



Over iron: the left sensor drops to **21, 22, 29**; the right still sees the floor.

PART 3

Step by step in C

One small program mines all the iron

STEP 1 · KNOW THE RULE

How does the robot collect?

Over it

A colour sensor must be on top of the iron.

Stop and flash

Wheels **0**, LED **flashing red**, held for about **3 s**.

Load it

The iron goes on board: **+10**. The robot holds up to 6.

In code we do it ourselves: **Duration = 3200** holds the stop and the flashing LED for 3.2 s, and Game1() waits until it ends.

STEP 2 · THE PLAN

Wander, and collect what you find

1. Left **or** right sensor sees iron (all dark) → **Collect**.
2. Wall or block close ahead → **turn left** for a moment.
3. Anything else → **drive forward**.

The robot bounces round the field like a ball and covers it, and picks up iron on the way.

STEP 3 · LINES 3–9

A function that collects: collect()

```
3  void collect() {
4      WheelLeft = 0;
5      WheelRight = 0;
6      LED_1 = 2;                          // flashing
7      LED_R = 255; LED_G = 0; LED_B = 0;  // red
8      Duration = 3200;                    // hold 3.2 seconds
9  }
```

Wheels 0: stop on the iron.

LED_1 = 2: flashing, in **red** (255, 0, 0).

Duration = 3200: hold it 3.2 s.

STEP 4 · LINES 11–16

IronLeft and IronRight: R, G and B all ≤ 60

```
11  void Game1() {
12      LED_1 = 0;
13      if (CSLeft_R <= 60 && CSLeft_G <= 60 && CSLeft_B <= 60) {           // IronLeft
14          collect();
15      } else if (CSRight_R <= 60 && CSRight_G <= 60 && CSRight_B <= 60) { // IronRight
16          collect();
```

LED_1 = 0 first: the LED is off unless we collect.

&&: all three must be true. Floor is about 230, so it never matches by mistake.

STEP 5 · LINES 17–23

Ahead turns away, Forward drives

```
17      } else if (US_Front <= 20) {                                    // Ahead
18          WheelLeft = -2; WheelRight = 2;
19          Duration = 700;
20      } else {                                                        // Forward
21          WheelLeft = 4.5; WheelRight = 4.5;
22      }
23  }
```

Ahead: −2 / 2 for 700 ms ≈ a 64° turn, so each bounce sends the robot a new way.

Forward: the last else, 4.5 / 4.5, about 27 cm/s.

WHY THIS ORDER?

The most important branch goes first

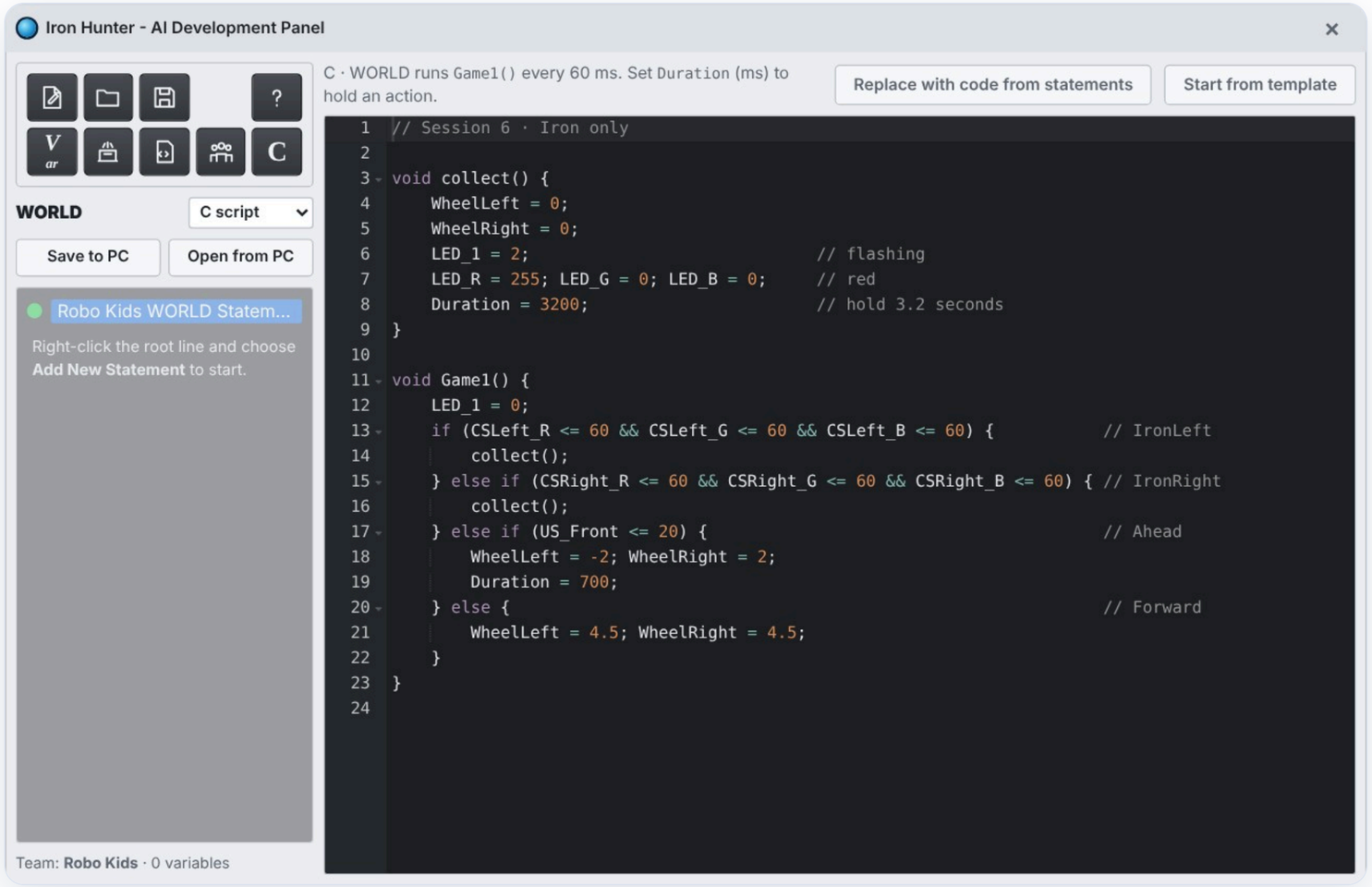
#	Branch	Why here
1–2	**IronLeft, IronRight**	Iron next to a wall is still collected
3	**Ahead**	Safety before driving on
4	**Forward**	The else: runs only when nothing above is true

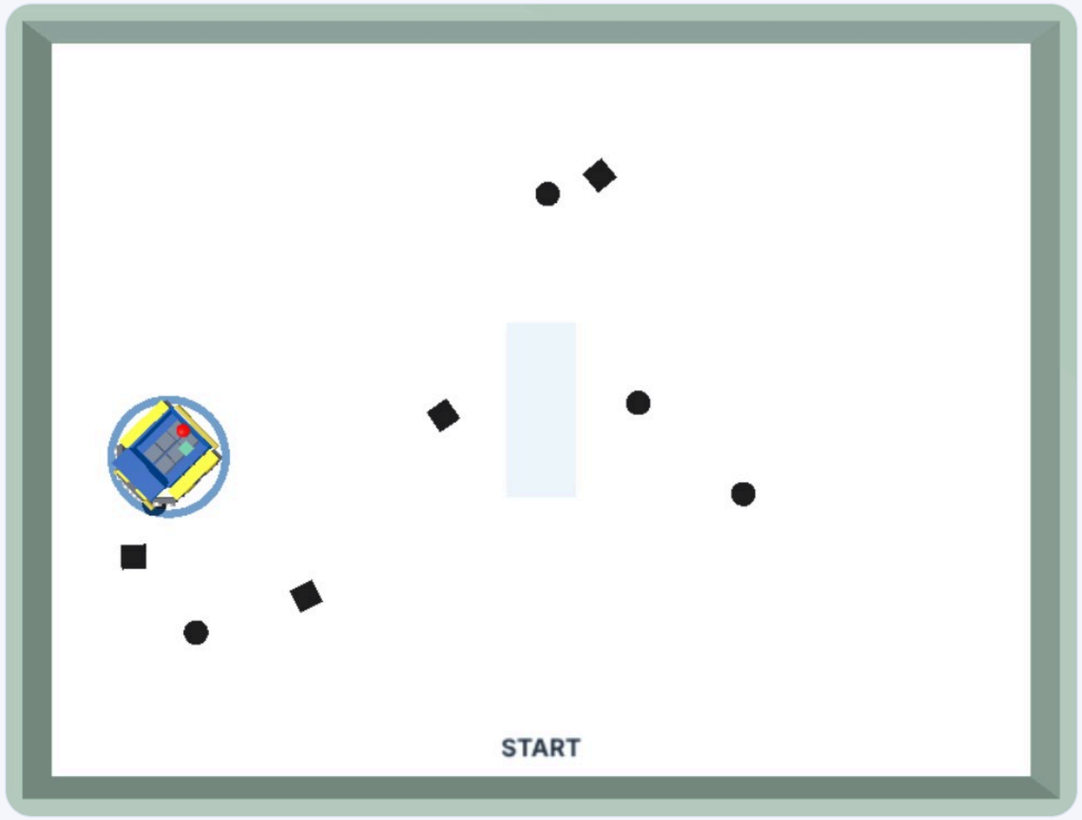
Put Forward first and the robot would **never** collect or turn.

STEP 6 · BUILD AND LOAD

The whole program, built

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





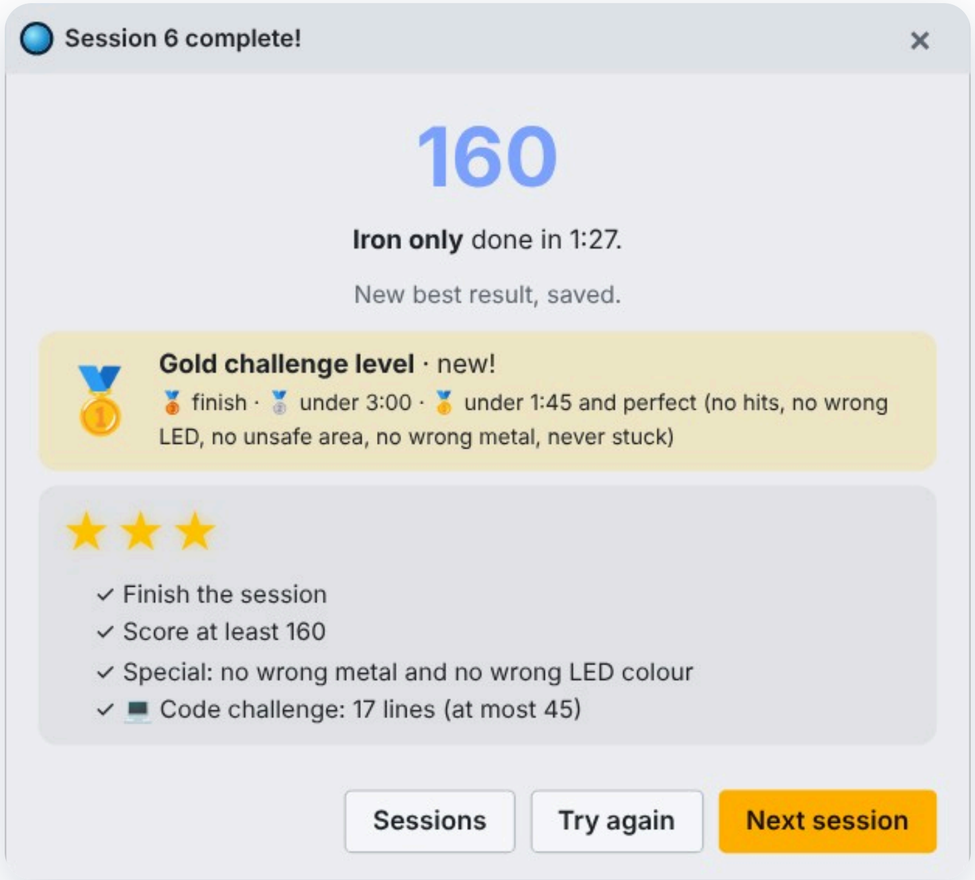
STEP 7 · RUN

Play, and watch the first catch

After **7.5 s** the left sensor is on iron: CSLeft **24, 21, 25**.

IronLeft is true → collect(): the robot stops and the red LED flashes.

Then it drives on to look for the next one.



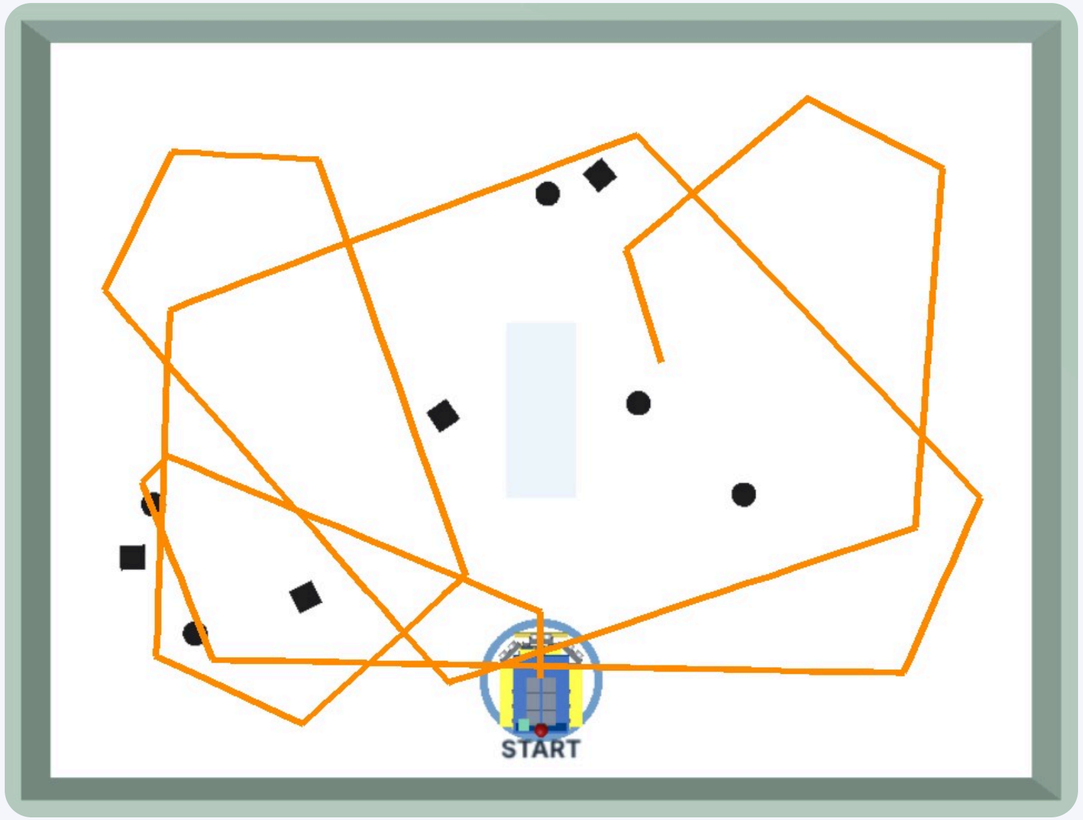
RESULT · TESTED

160 points in 1:27 · Gold

Medals · Bronze: finish · Silver: under 3:00 · Gold: under 1:45 and perfect.

Stars · finish · score 160 · no wrong metal and no wrong LED colour.

Code challenge · at most 45 lines. We used 17.



THE PATH · RECORDED

Bounce, bounce, collect

Straight lines, a turn at every wall: the field gets covered.

The robot never knew where the iron was. It found 6 of the 9.

Simple rules + sensors = **smart behaviour**.

DID NOT WORK?

Check first, then tune carefully

What you see	What to do
Drives over iron, no stop	Check all three values are **<= 60** and joined with **&&**
Stops but nothing is loaded	collect(): **LED_1 = 2**, red **255, 0, 0**, **Duration = 3200**
Build error with a line number	Check the semicolons ; and the brackets { }
Keeps hitting walls	Ahead: **US_Front <= 20**, Duration 700
Old program still on the robot	**Build** again, then **Load** on BLUE

NEXT TIME

Presentation 7 · One of each

Iron, silver and gold on one field: now the robot needs **all three colours** to tell them apart.