PRESENTATION 0 · LEARNING WITH PYTHON

Welcome to the AiBotCup Simulator

Meet the robot, the mining field, the three ways to program and the simulator screen.

TheBotCup 2026 · Build. Code. Mine. Win.



WHAT IS THEBOTCUP?

An international robotics competition about mining

Teams program a mining robot to find metals, avoid danger and bring the resources home safely. The mission: **more resources for a better life.**

Build

Choose how your robot behaves.

Code

Turn your ideas into a program.

Mine

Collect iron, silver and gold.

Win

Score the most points in the match.

THE SIMULATOR

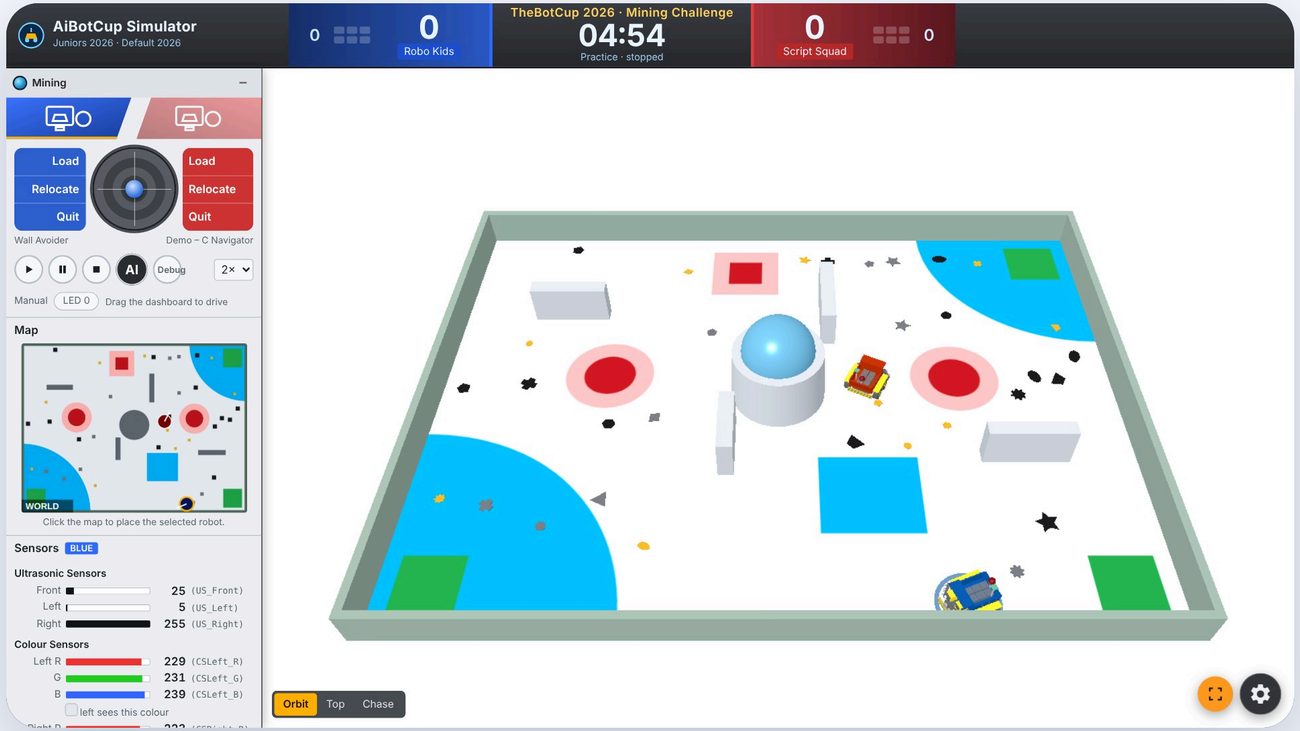
A robot you program in your web browser

No robot kit needed. A 3D mining robot drives on a 3D field.

You write the program, press Play, and the robot drives by itself.

It reads its sensors to find walls, colours and its position.

Two robots can play: BLUE against RED.





THE ROBOT

Meet the Miner-Bot and its sensors

Sensor	What it tells the robot
Ultrasonic · Front, Left, Right	How far away a wall is
Colour · Left, Right	The colour of the floor below
Compass	Which way the robot faces (0–360°)
Position · X, Y	Where the robot is on the field
Time	How long the game has run

It drives like a tank: **WheelLeft** and **WheelRight** each set one side's speed. The red **LED** can light up in any colour.

THE FIELD

Read the floor: every colour means something

Iron, silver, gold
Metal objects to collect

Green: deposit
Bring the metals here

Blue: water
Ponds on the field

Red: unsafe area
Stay away!

Grey: walls
And the tower in the middle

White: the floor
Free to drive on



PART 2

Three ways to program the same robot

Statements · Python · C

PROGRAMMING LANGUAGES

Three ways to program: we type Python

BOXES

Statements

No typing: fill in boxes. Each statement says WHEN and DO.

WE USE THIS ONE

Python

Real code that reads almost like English. Used in schools and by scientists all over the world.

BRACES

C

The language inside real robots and machines. Fast, exact and powerful.

PYTHON

One if = WHEN + DO

```python
# Wall Avoider

def game1():
    if US_Front <= 20:                # AvoidWall
        WheelLeft = 3; WheelRight = -3
        Duration = 400                # 0.4 s
    else:                             # MoveForward
        WheelLeft = 5; WheelRight = 5
```

WHEN · if

US_Front <= 20 means: a wall is close in front.

DO · the lines inside

WheelLeft 3, WheelRight −3 for **400 ms**: the robot turns on the spot.

A comment names each branch, like **AvoidWall**.

PYTHON

game1() runs every 60 ms, from the top

IF · AVOIDWALL

WHEN US_Front <= 20

DO turn: left 3, right −3, for 400 ms

ELSE · MOVEFORWARD

WHEN anything else (no conditions)

DO drive: left 5, right 5

The **first** branch whose condition is true is the one that runs. Put the most important one at the top.

SAME PROGRAM, OTHER LANGUAGES

The same Wall Avoider in Python and in C

Python

```python
# Wall Avoider · WORLD
# Team: Robo Kids
# game1() is called every 60 ms. Setting Duration (ms) holds the wheels
# and LED for that long before game1() runs again.


def game1():
    if 0 <= US_Front <= 20:  # AvoidWall
        WheelLeft = 3
        WheelRight = -3
        LED_1 = 0
        LED_R = 255
        LED_G = 0
        LED_B = 0
        Duration = 400
    else:  # MoveForward
        WheelLeft = 5
        WheelRight = 5
        LED_1 = 0
        LED_R = 255
        LED_G = 0
        LED_B = 0
        Duration = 60

```

C

```c
// Wall Avoider · WORLD
// Team: Robo Kids
// Called every 60 ms. Setting Duration (ms) holds the wheels and LED
// for that long before Game1() runs again.

void Game1()
{
    if (US_Front >= 0 && US_Front <= 20)  // AvoidWall
    {
        WheelLeft = 3; WheelRight = -3; LED_1 = 0; LED_R = 255; LED_G = 0; LED_B = 0;
        Duration = 400;
    }
    else  // MoveForward
    {
        WheelLeft = 5; WheelRight = 5; LED_1 = 0; LED_R = 255; LED_G = 0; LED_B = 0;
        Duration = 60;
    }
}

```

PART 3

Meet the simulator screen

Score bar · Control panel · 3D arena

THE SCREEN

Five places to know



1 **Score bar**: scores, cargo and the clock

2 **Control panel**: load, play and the AI button

3 **3D arena**: where the robots drive

4 **Cameras**: Orbit, Top and Chase

5 **Full screen** and settings

1 · SCORE BAR

Everything about the match, in one line



BLUE team

Score and the cargo slots: what your robot is carrying.

The clock

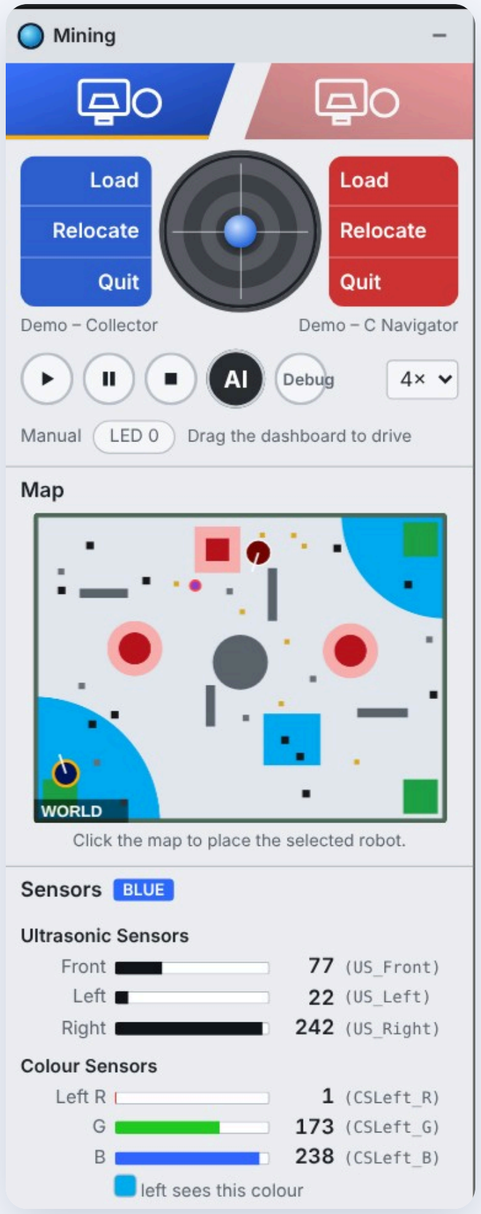
A match lasts **5:00**. The time counts down.

RED team

The other robot's score and cargo.

Event feed

Short messages: points won, deposits, penalties.



2 · CONTROL PANEL · PART 1

Robots and match buttons

BLUE / RED selector

Picks the robot that the joystick, the Sensors list and the Chase camera follow.

Load

Puts one of your built programs on that robot. Its name shows under the buttons.

Relocate

Moves a stuck robot to a nearby spot. In Competition: 3 per game, then frozen 10 s.

Quit

Takes the program off the robot. Not allowed during a competition match.

Joystick

Drag to drive by hand, or use the arrow keys / W A S D. Practice only.

▶ Play · ⏸ Pause · ■ Stop

Start the clock, freeze it, or stop and reset the field for a new try.

AI

Opens the AI Development Panel: write, build and save programs.

Debug · Speed

Debug opens a live graph of the sensors. Speed runs the match at 1×, 2×, 4× or 8×.

2 · CONTROL PANEL · PART 2

Manual tools, map, sensors and mode boxes

Manual · LED chip

While driving by hand, click LED: 0 off, 1 steady (deposit), 2 flashing (collect).

Map

A small view of the WORLD from above. In Practice, click it to place the selected robot.

Sensors

Live values with their names, like **US_Front**. Use these numbers in your conditions.

Debug window

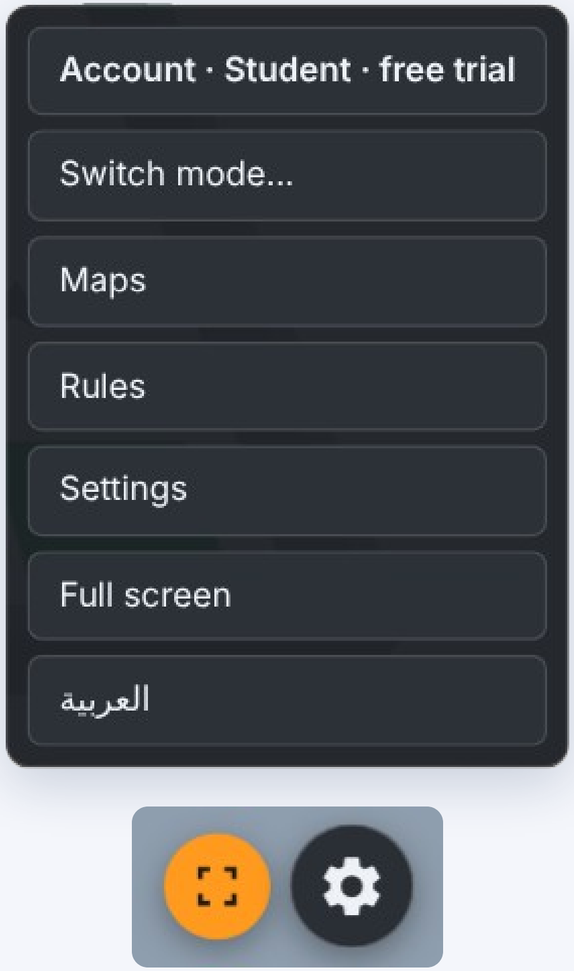
Graph of the last 30 s: Ultrasonic, Colour sensors, Compass & position, or Wheels.

Learning box

In Learning mode: the mission, **Sessions** to pick one, **Restart session**.

Collapse –

Hides the panel for a bigger arena. **Show control panel** brings it back.



5 · THE GEAR BUTTON ⚙

The menu for everything else

Account · your name, license type and days left, **Request teacher account...**, Log out.

Switch mode... · back to School / University, category and mode.

Maps · choose the arena map, save it, or open a map file from your judges.

Rules · the rulebook for your category.

Settings · rule set, game length, graphics, theme and more (next slide).

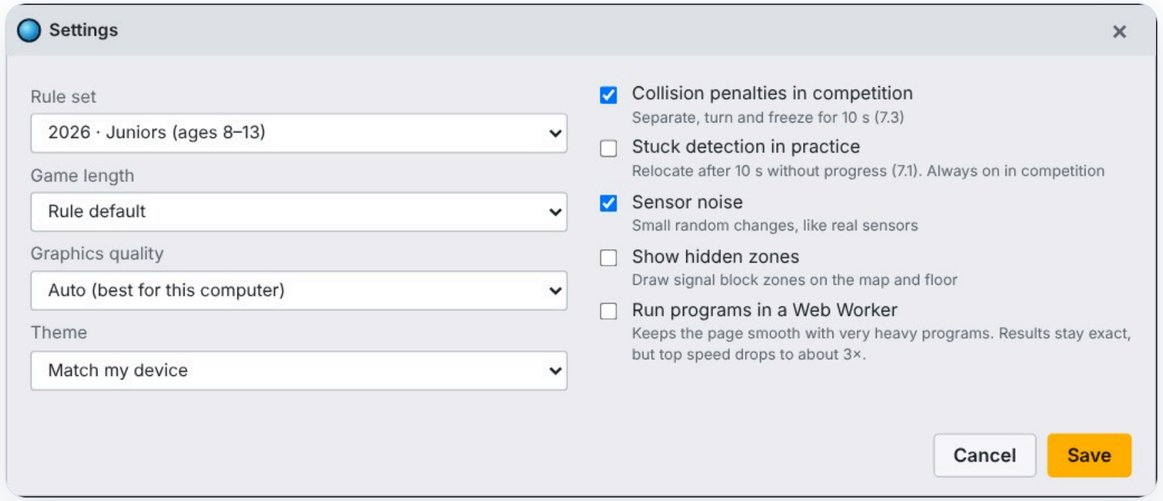
Full screen · same as the orange button next to the gear.

العربية / English · switch the language.

Teachers and judges also see Leaderboards and Standings.

⚙ GEAR → SETTINGS

Settings, one by one



Rule set · Juniors, Seniors, Beginners or Advanced rules.

Game length · rule default (5:00), or 6:00 / 8:00 for practice.

Graphics quality · Auto, High, Medium, or Low for older computers.

Theme · match my device, Dark or Light.

Collision penalties

In Competition: separate, turn, freeze 10 s.

Stuck detection

In Practice: relocate after 10 s with no progress.

Sensor noise

Small random changes, like real sensors.

Hidden zones

Shows the signal block zones where X and Y read 0.

Web Worker

Smoother page for very heavy programs (top speed about 3×).

HOW WE WORK

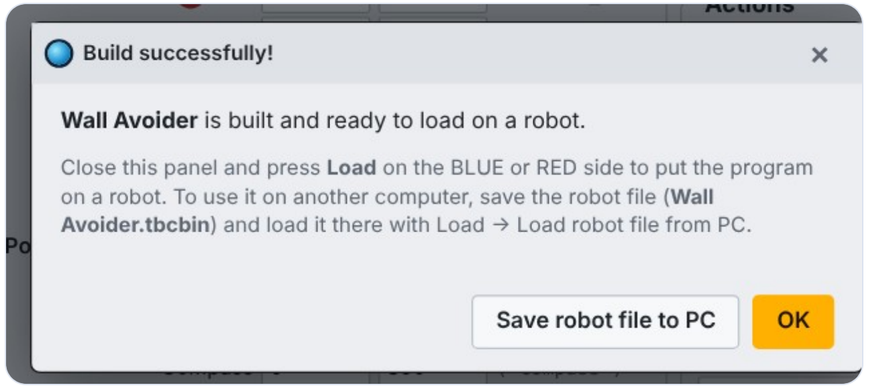
Write, Build, Load, Play

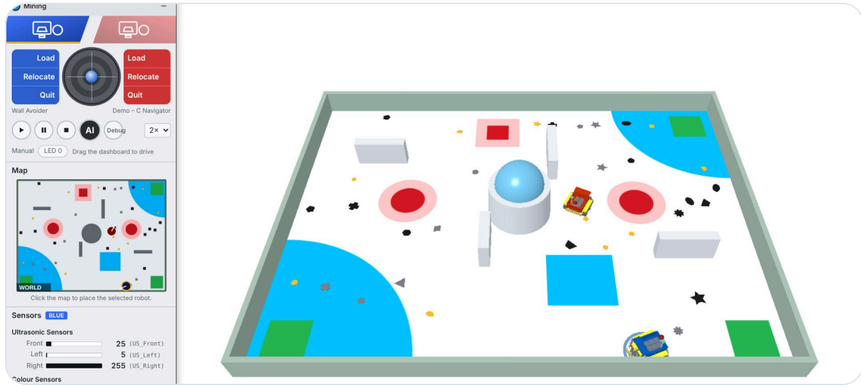
1
Write
Type your Python code in the AI panel.

2
Build
Wait for "Build successfully!"

3
Load
Load the program on the BLUE robot.

4
Play ▶
Watch, then fix and try again.

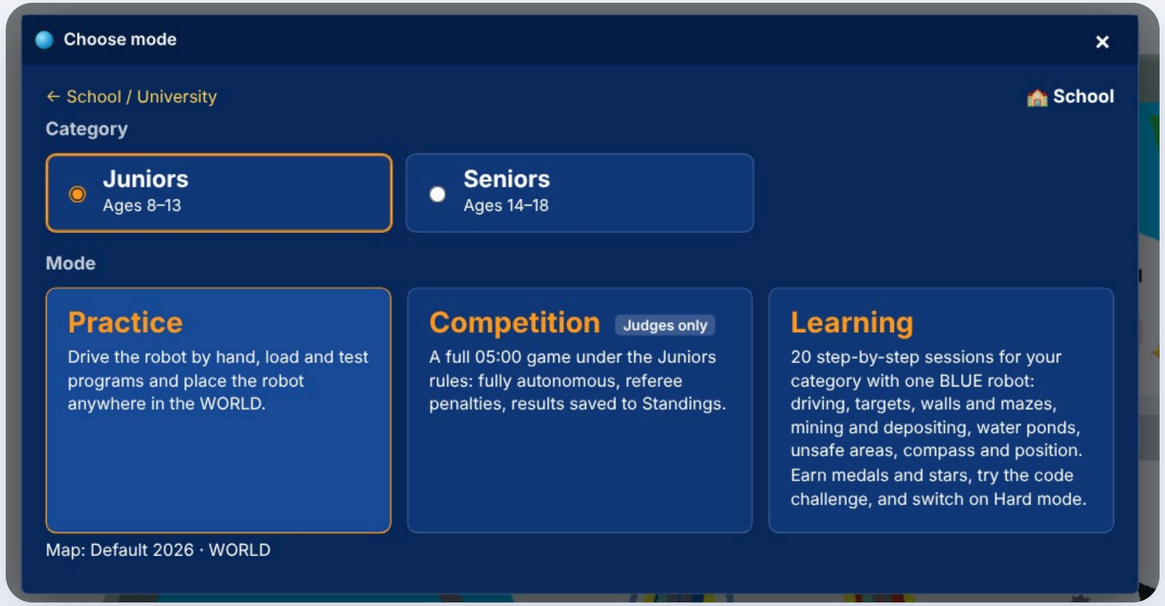




Did not work?
Change one thing, **build again**, and test.

CHOOSING A MODE

Pick your level, then how you play



School: Juniors 8–13 · Seniors 14–18. Or **University**.

Practice: free play, try anything.

Learning: 20 guided sessions. This course.

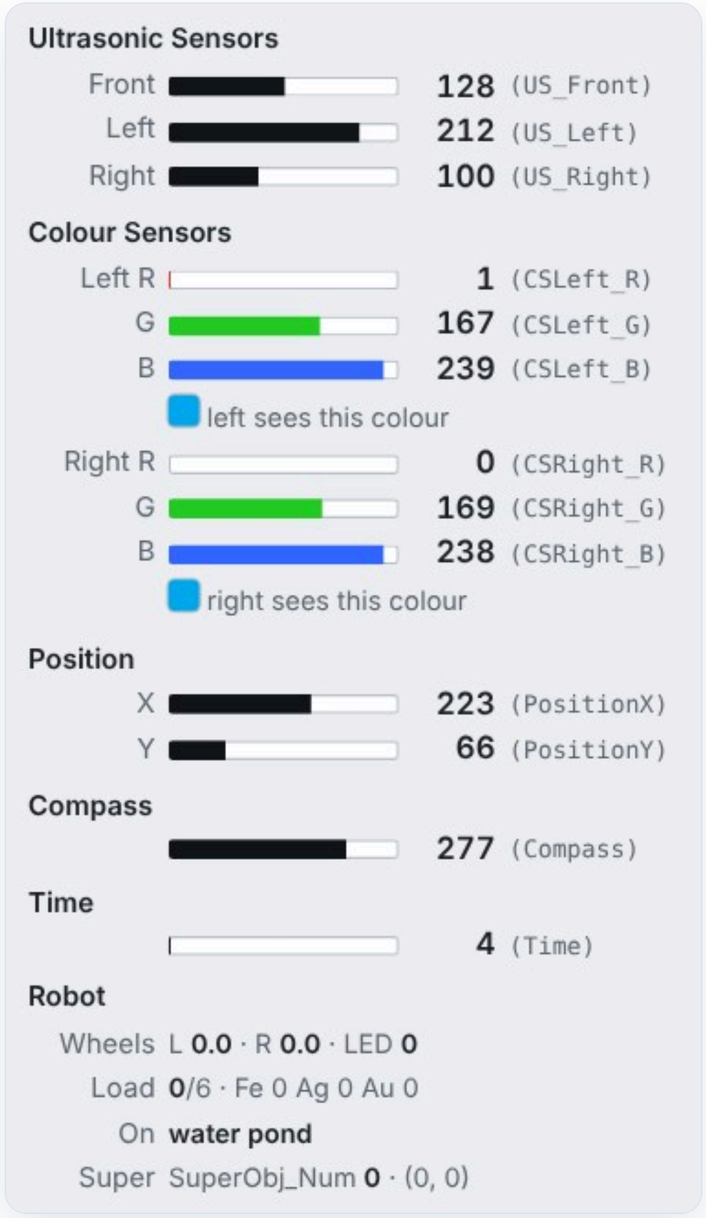
Competition: official matches, run by judges.

AGE GROUPS

Four categories, two tracks

	SCHOOL Juniors	SCHOOL Seniors	UNIVERSITY Beginners	UNIVERSITY Advanced
Ages	8–13	14–18	18–19	19–22
Team	2 or more	1 or more	1 or more	1 or more
WORLD edge	**Outer wall**	No wall	No wall	No wall
Swamps (20 % speed)	No	**Yes**	**Yes**	**Yes**
Floor sensing	2 colour sensors	2 colour sensors	2 colour sensors	**1 camera**
Programming	Statements, Python, C	Statements, Python, C	Statements, Python, C	**Python or C only**

All categories: one WORLD, **5-minute** games, BLUE vs RED. No wall means: leave the field and you are frozen 10 s. Age is counted on 1 July 2026. Beginners play the Seniors rules; Advanced also has **O (Oil)** to collect and **R (Rocks)** to leave.



SENSORS · JUNIORS, SENIORS, BEGINNERS

The Miner-Bot with colour sensors

3 ultrasonic · **US_Front**, **US_Left**, **US_Right**: distance to a wall or obstacle, 0–255 cm.

2 colour sensors under the robot · **CSLeft_R/G/B**, **CSRight_R/G/B**, each 0–255: find metals, ponds, traps, swamps, deposit areas.

Position · **PositionX**, **PositionY** in cm (0 inside hidden zones).

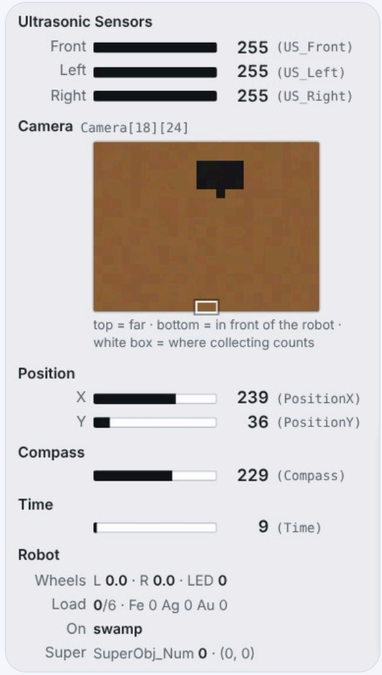
Compass 0–359° (0 = north, 90 = west) · **Time** in seconds.

Outputs · **WheelLeft**, **WheelRight** −5 to 5 · **LED_1** 0/1/2 · **LED_R/G/B** colour.

SENSORS · UNIVERSITY

Beginners: colour sensors · Advanced: a camera



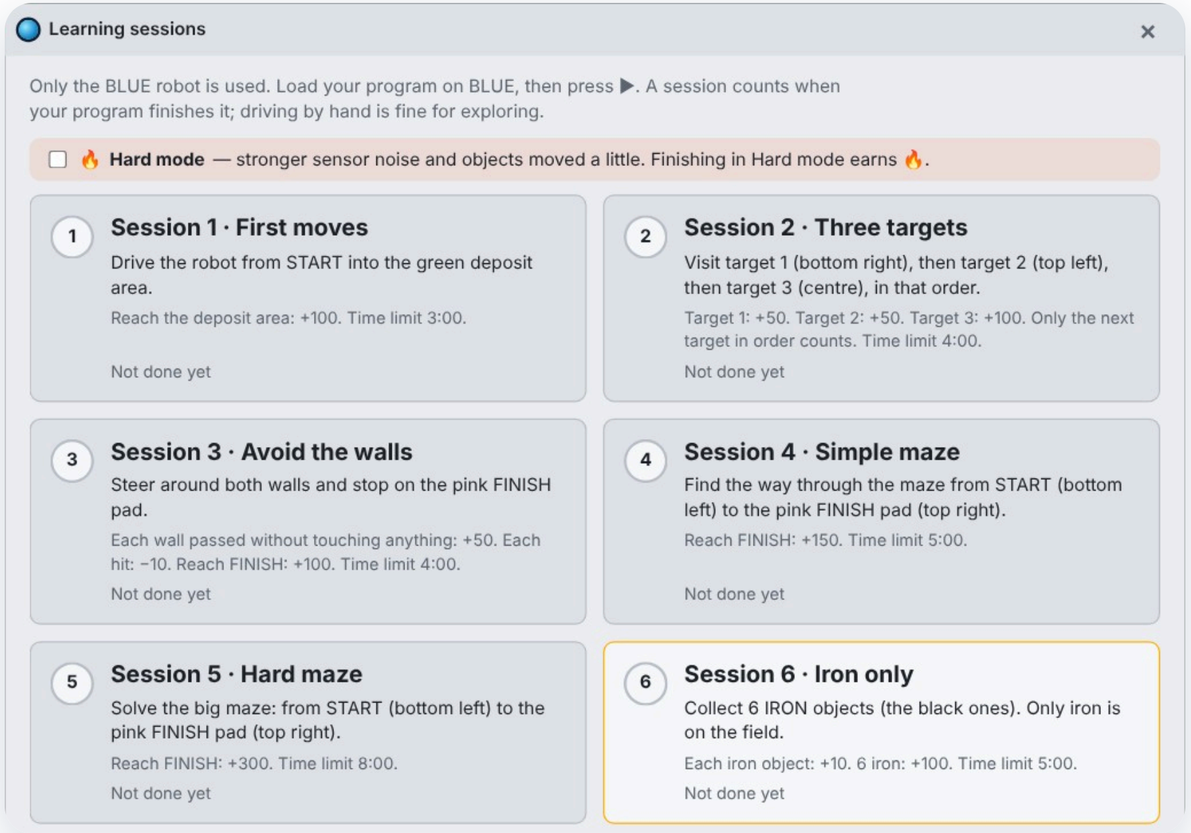


Beginners (18–19) · the same robot as Seniors: 2 colour sensors.

Advanced (19–22) · no colour sensors. One **camera**: **Camera[row][col] = [R,G,B]**, 24 × 18 pixels.

Row 0 = far, row 17 = nearest. The point that counts is the **bottom centre**, 8 cm in front.

Same 3 ultrasonic, position, compass, time, wheels and LED.



LEARNING MODE

20 sessions, one step at a time

- You program the **BLUE** robot only.
- Win a medal and stars in each session.
- Done? Try it again in **Hard mode**.

NEXT TIME

Presentation 1 · First moves

Drive the robot from START into the green deposit area: **+100 points** in 3:00.

Log in → **School** → **Juniors** → **Learning · Session 1**