PRESENTATION 15 · LEARNING WITH STATEMENTS

One of each

One iron, one silver, one gold, and never a second of the same: check what you carry before collecting.

TheBotCup 2026 · AiBotCup Simulator



RECAP · PRESENTATION 14

LoadedObjects counts everything

LoadedObjects < 3

collect while not full

But…

iron + iron + silver is also 3

It counts objects, not **which metal**. Today that matters.

PART 1

The mission: Session 15

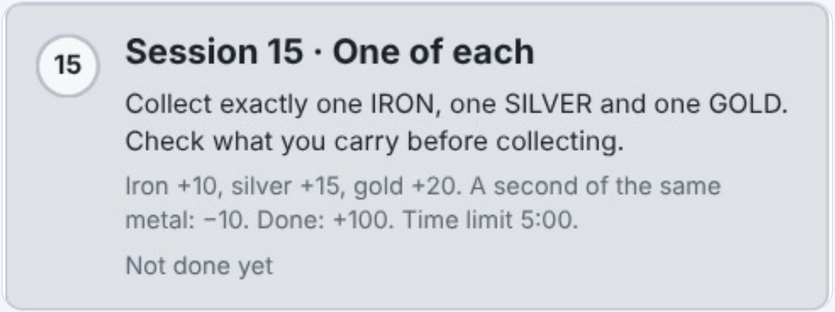
Exactly one of each metal

SESSION 15 · ONE OF EACH

One IRON, one SILVER, one GOLD

Iron **+10**, silver **+15**, gold **+20**. A second of the same metal: **−10**. Done: **+100**.

Time limit **5:00**. At most **12** statements.





THE FIELD

Plenty of every metal

START at the bottom centre.

Many irons, silvers and golds.

The robot will drive over metals it **must not** take.

PART 2

One counter per metal

LoadedIron, LoadedSilver, LoadedGold

THE IDEA · BUILT-IN VARIABLES

The robot counts each metal

LoadedIron

irons on board

LoadedSilver

silvers on board

LoadedGold

golds on board

Like LoadedObjects: no Var button, they count only once a metal is **really on board**, and a deposit sets them back to 0.

HOW TO USE THEM

Each collect checks its own metal

Collect statements	Advanced Condition	Means
IronLeft, IronRight	LoadedIron < 1	no iron yet
SilverLeft, SilverRight	LoadedSilver < 1	no silver yet
GoldLeft, GoldRight	LoadedGold < 1	no gold yet

Once a metal is on board, its two statements **switch off**: the robot drives over that metal.

PART 3

Step by step in statements

Eight statements

STEP 1 · THE WHOLE PROGRAM

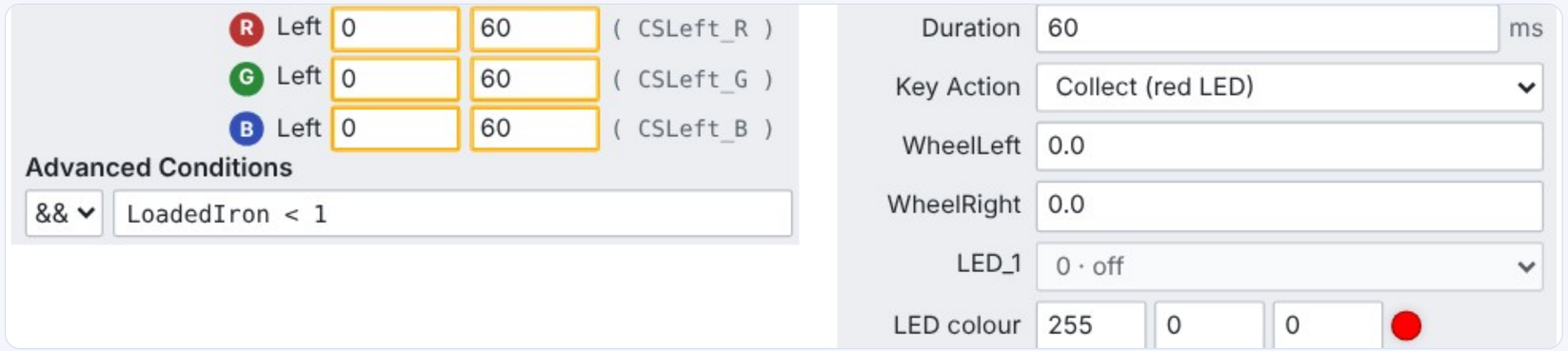
Six careful collects, two drivers

Statement	Condition	Advanced	Action
1 IronLeft	left iron	LoadedIron < 1	Collect · 0 / 0
2 IronRight	right iron	LoadedIron < 1	Collect · 0 / 0
3 SilverLeft	left silver	LoadedSilver < 1	Collect · 0 / 0
4 SilverRight	right silver	LoadedSilver < 1	Collect · 0 / 0
5 GoldLeft	left gold	LoadedGold < 1	Collect · 0 / 0
6 GoldRight	right gold	LoadedGold < 1	Collect · 0 / 0
7 Ahead	US_Front 0–20	none	−2 / 2 · 700 ms
8 Forward	none	none	4.5 / 4.5

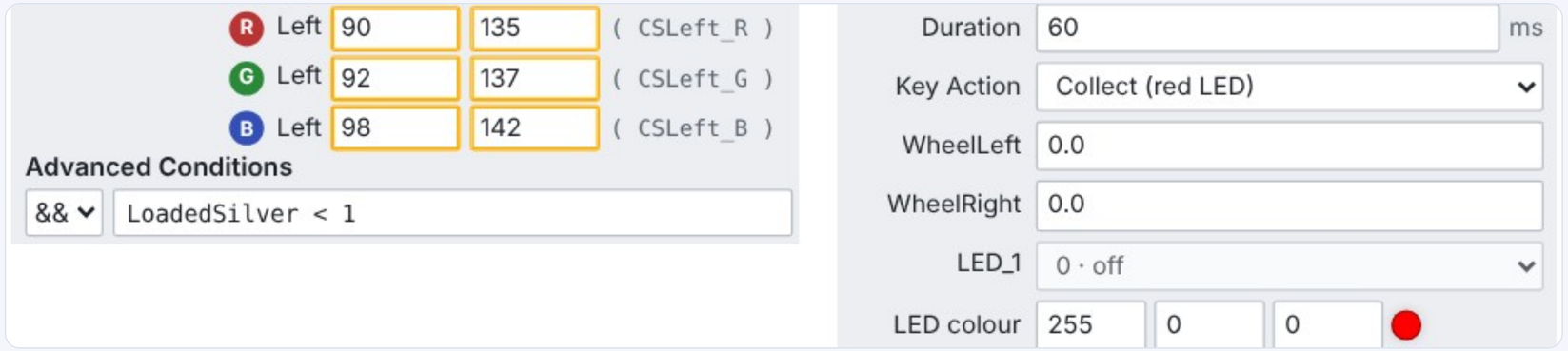
STEP 2 · STATEMENTS 1 TO 4

Iron and silver: only if we have none

1 IronLeft
LoadedIron < 1



3 SilverLeft
LoadedSilver < 1

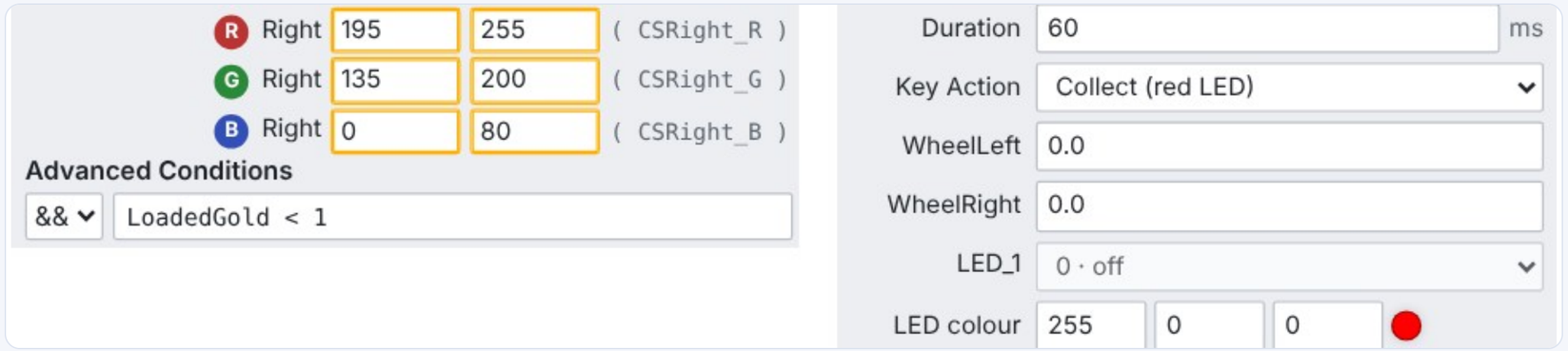


IronRight and SilverRight: the same with the **right** sensor.

STEP 3 · STATEMENTS 5 TO 8

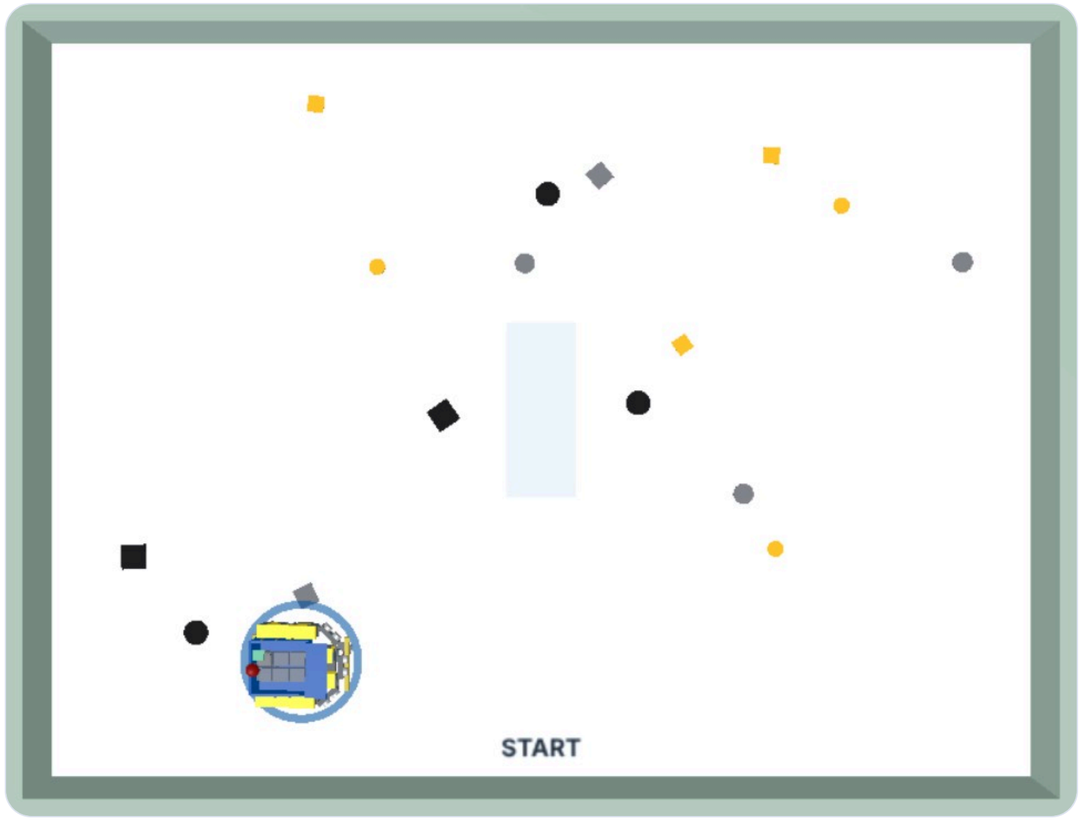
Gold, then the two drivers

6 GoldRight
LoadedGold < 1



7 Ahead
US_Front 0–20 → −2 / 2, 700 ms

8 Forward
no conditions → 4.5 / 4.5



Colour Sensors

Left R	112	(CSLeft_R)
G	111	(CSLeft_G)
B	116	(CSLeft_B)

left sees this colour

Right R	226	(CSRight_R)
G	232	(CSRight_G)
B	236	(CSRight_B)

right sees this colour

Robot

Wheels L **0.0** · R **0.0** · LED **2**

Load **1**/6 · Fe 1 Ag 0 Au 0

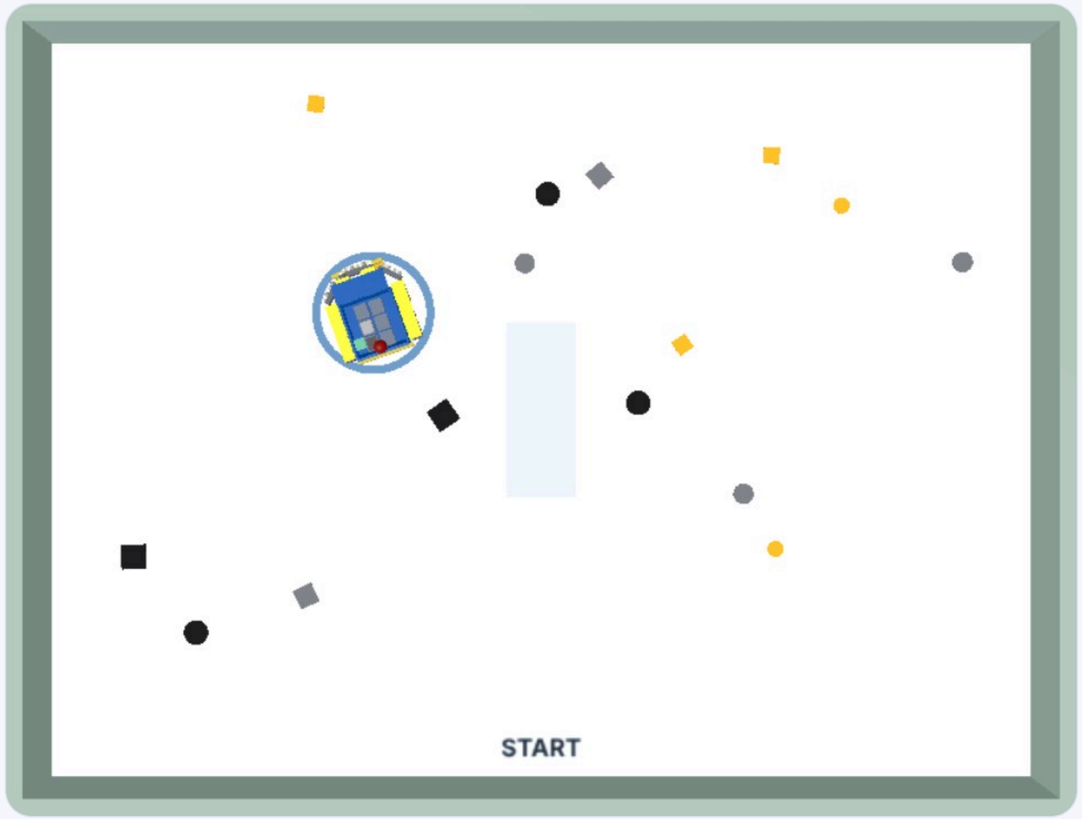
On **collecting 1.8 / 3 s**

STEP 4 · RUN · 0:15

Iron done, now silver

Load: **Fe 1**. Left sees silver, LoadedSilver is 0: **SilverLeft**.

Iron statements are now off.



Colour Sensors

Left R	225	(CSLeft_R)
G	233	(CSLeft_G)
B	238	(CSLeft_B)

left sees this colour

Right R	220	(CSRight_R)
G	167	(CSRight_G)
B	36	(CSRight_B)

right sees this colour

Robot

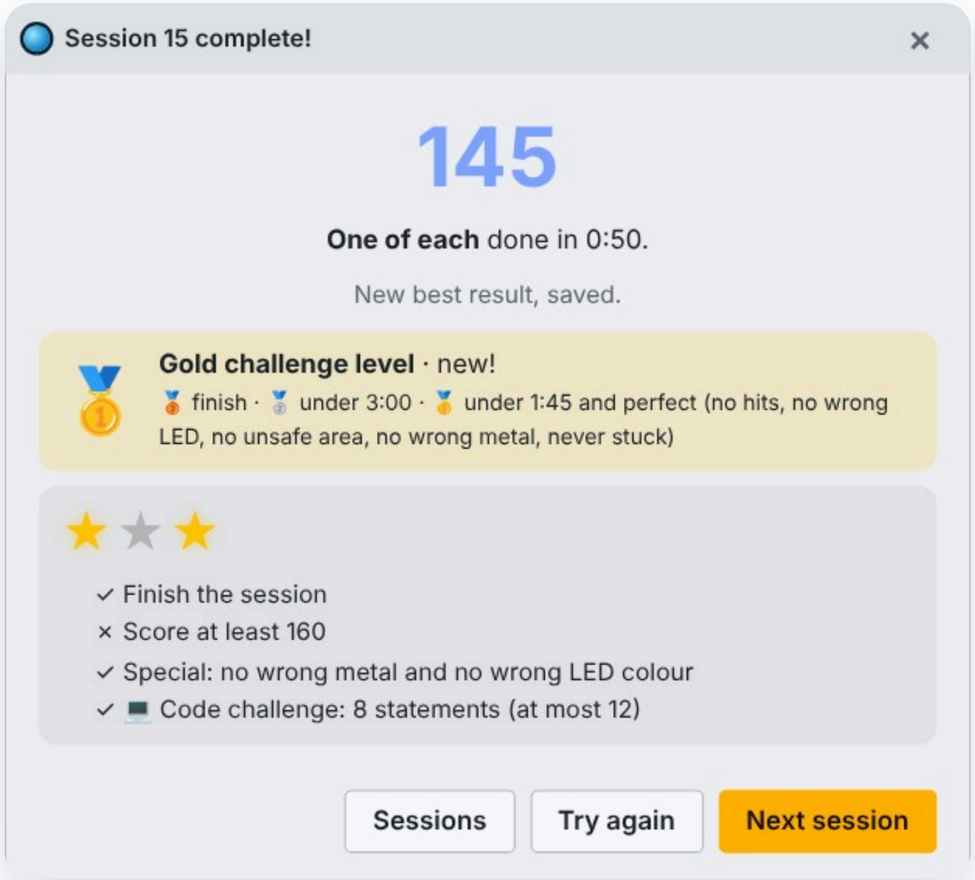
Wheels	L **0.0** · R **0.0** · LED **2**
Load	**2**/6 · Fe 1 Ag 1 Au 0
On	**collecting 1.8 / 3 s**

STEP 5 · RUN · 0:49

The last one: gold

Load: **Fe 1 · Ag 1**. Right sees gold: **GoldRight**.

On the way it passed irons and silvers **without stopping**.



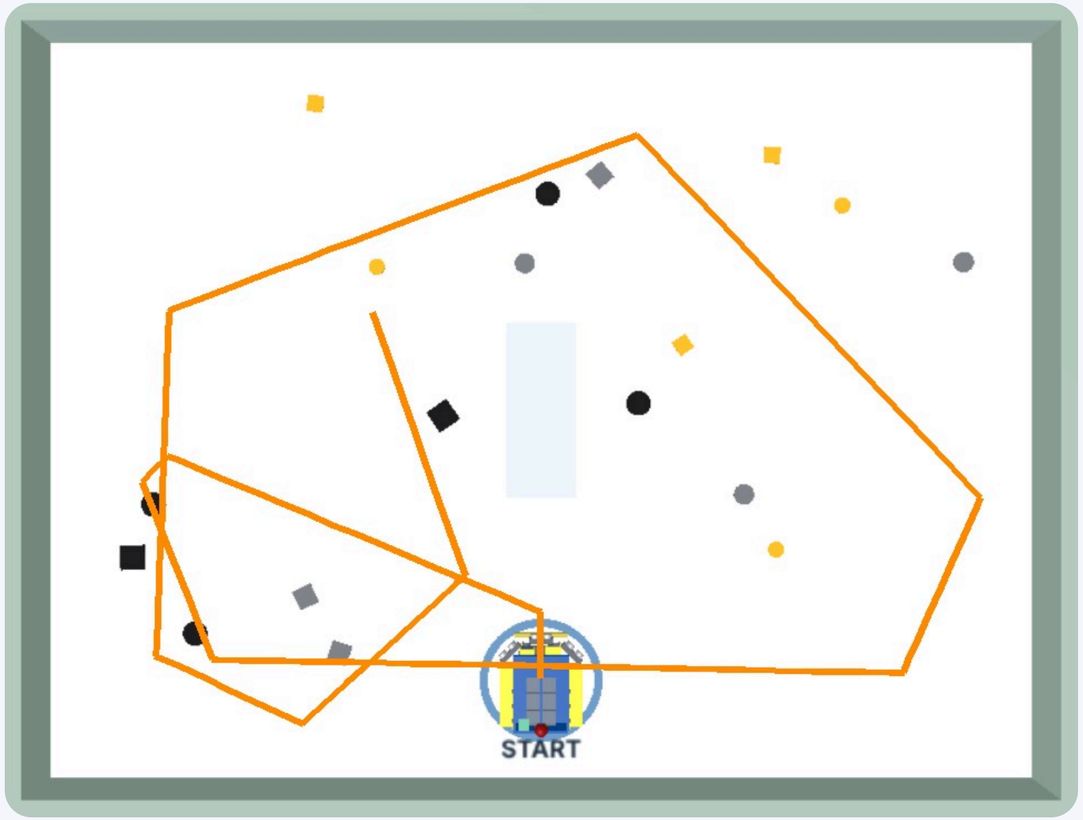
RESULT · TESTED

145 points in 0:50 · Gold

Score · 10 + 15 + 20 + 100, no −10.

Special · no wrong metal, no wrong LED. The “score 160” star was not reached.

Code challenge · at most 12 statements. We used 8.



THE PATH · RECORDED

Driving over, on purpose

The line crosses metals the robot **already had**.

Three counters, three switches: each metal turns its own statements **off**.

DID NOT WORK?

Check first, then tune carefully

What you see	What to do
A second iron, −10	Both iron statements need **LoadedIron < 1**
Never collects silver	Silver statements check **LoadedSilver**, not LoadedIron
Typing error in the condition	Spelling and capitals: **LoadedGold** exactly
Old program still on the robot	**Build** again, then **Load** on BLUE

NEXT TIME

Presentation 16 · Around the flags

A route around obstacles using **position and compass**.