PRESENTATION 8 · LEARNING WITH STATEMENTS

Two of each

Exactly two iron, two silver and two gold, counted by the simulator’s own built-in variables.

TheBotCup 2026 · AiBotCup Simulator



RECAP · PRESENTATION 7

We made our own counters

1 · Make

V_{ar} button → Iron, Silver, Gold = 0

2 · Read

Advanced Condition: **Iron < 2**

3 · Change

Advanced Action: **Iron = Iron + 1;**

Result: 2 + 2 + 2, **190 points in 1:15**. Today the same idea, with less work.

PART 1

The mission: Session 8

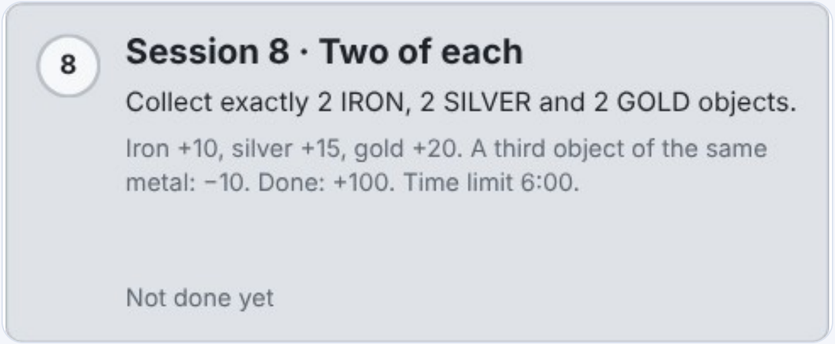
Exactly two of every metal

SESSION 8 · TWO OF EACH

2 iron, 2 silver, 2 gold. Not one more.

What happens	Points
Iron · silver · gold	+10 · +15 · +20
Two of each: done	+100
A third of the same metal	−10, refused

Time limit **6:00**.





THE FIELD

Six of each metal

- **6 iron** · black
- **6 silver** · grey
- **6 gold** · yellow

You need 2 of each: 12 objects must be **left behind**.

WE TESTED IT · NO COUNTING

Without counters: two penalties

What it did

It tried to collect a third of a metal **twice**. Both were refused: **−10, −10**.

Score

170 in 1:21, instead of 190. Each refused pickup also wasted about 3 seconds.

So we need counters again. This time we do not even have to make them.

PART 2

The simulator counts too

Built-in variables

BUILT-IN VARIABLES

Ready to use, no Var button needed

Name	Means	In the panel
LoadedObjects	objects on the robot now (0–6)	Load 3/6
LoadedIron	iron on the robot	Fe
LoadedSilver	silver on the robot	Ag
LoadedGold	gold on the robot	Au

Fe, Ag and Au are the chemistry symbols for iron, silver and gold. You can **read** these, but not change them: the simulator does that.

SENSORS PANEL · ROBOT SECTION

They count only what is really on board

Robot

Wheels L **0.0** · R **0.0** · LED **2**
Load **3**/6 · Fe 1 Ag 2 Au 0
On **collecting 1.3 / 3 s**

35.8 s: collecting the 2nd iron, still **Fe 1**.

Robot

Wheels L **0.0** · R **0.0** · LED **2**
Load **5**/6 · Fe 2 Ag 2 Au 1
On **collecting 1.5 / 3 s**

73.4 s: collecting the 2nd gold, still **Au 1**.

The number goes up only **after** the 3-second collect has finished.

OWN COUNTER OR BUILT-IN?

Same rule, different way of counting

	Iron (P7, own)	LoadedIron (built-in)
Made by	you, with V_{ar}	the simulator
Goes up	when Collect **starts**	when the object is **on board**
If a collect fails	would still go up	stays the same
Extra work	Advanced Action in 6 statements	none

Own variables are still important: use them for anything the simulator does **not** count for you.

PART 3

Step by step in statements

Eight statements, zero own variables

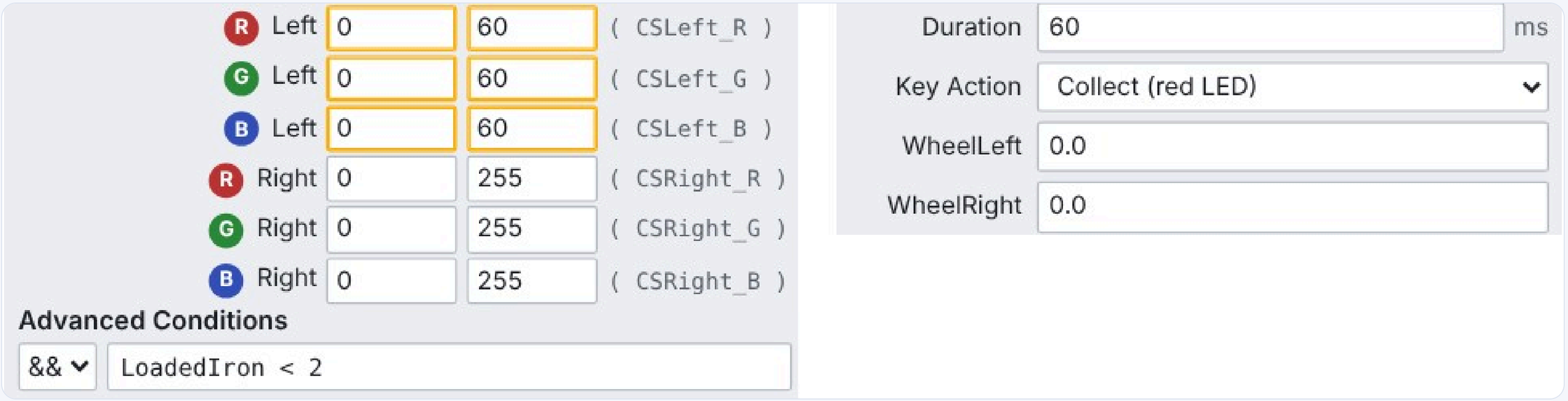
STEP 1 · THE WHOLE PROGRAM

Same shape as P7, built-in counters

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

STEP 2 · STATEMENTS 1 AND 2 · IRON

IronLeft: one Advanced Condition, no Advanced Action



Type **LoadedIron < 2** exactly: capital L and I. **IronRight** is the same with the 0–60 ranges on CSRight.

STEP 3 · STATEMENTS 3 TO 6 · SILVER AND GOLD

LoadedSilver and LoadedGold

3 SilverLeft
LoadedSilver < 2

R Left	90	135	(CSLeft_R)
G Left	92	137	(CSLeft_G)
B Left	98	142	(CSLeft_B)
R Right	0	255	(CSRight_R)
G Right	0	255	(CSRight_G)
B Right	0	255	(CSRight_B)

Advanced Conditions

&& LoadedSilver < 2

Duration	60 ms
Key Action	Collect (red LED)
WheelLeft	0.0
WheelRight	0.0

5 GoldLeft
LoadedGold < 2

R Left	195	255	(CSLeft_R)
G Left	135	200	(CSLeft_G)
B Left	0	80	(CSLeft_B)
R Right	0	255	(CSRight_R)
G Right	0	255	(CSRight_G)
B Right	0	255	(CSRight_B)

Advanced Conditions

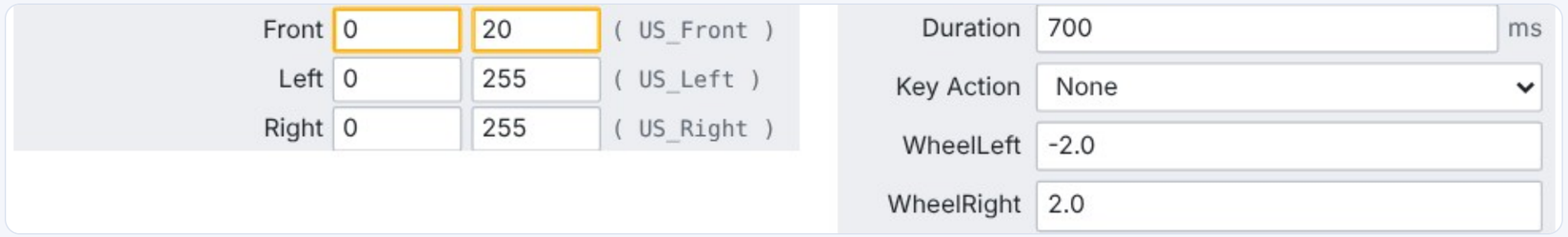
&& LoadedGold < 2

Duration	60 ms
Key Action	Collect (red LED)
WheelLeft	0.0
WheelRight	0.0

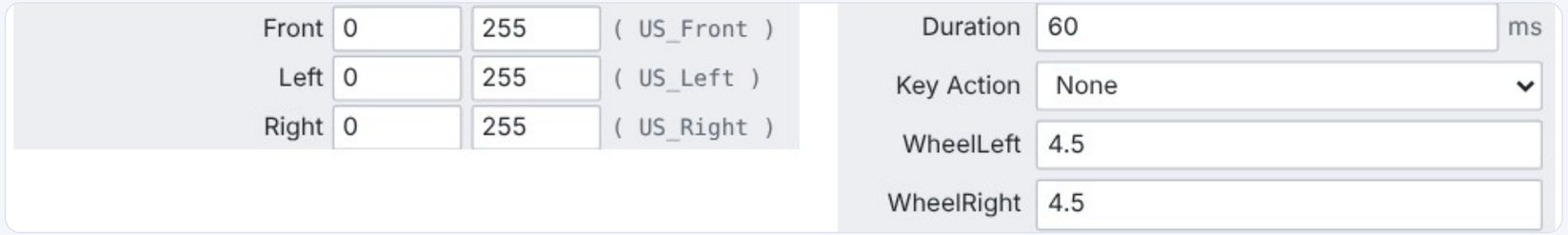
STEP 4 · STATEMENTS 7 AND 8

Ahead and Forward, unchanged

7 Ahead
US_Front 0–20



8 Forward
no conditions



A metal that is already full matches no collect statement, so **Forward drives over it**.



STEP 5 · RUN

The 2nd iron at 35.8 s

Colour Sensors

Left R	23	(CSLeft_R)
G	22	(CSLeft_G)
B	28	(CSLeft_B)

left sees this colour

Right R	229	(CSRight_R)
G	230	(CSRight_G)
B	240	(CSRight_B)

right sees this colour

Position

X	130	(PositionX)
Y	152	(PositionY)

Compass

111 (Compass)

Time

35 (Time)

Robot

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

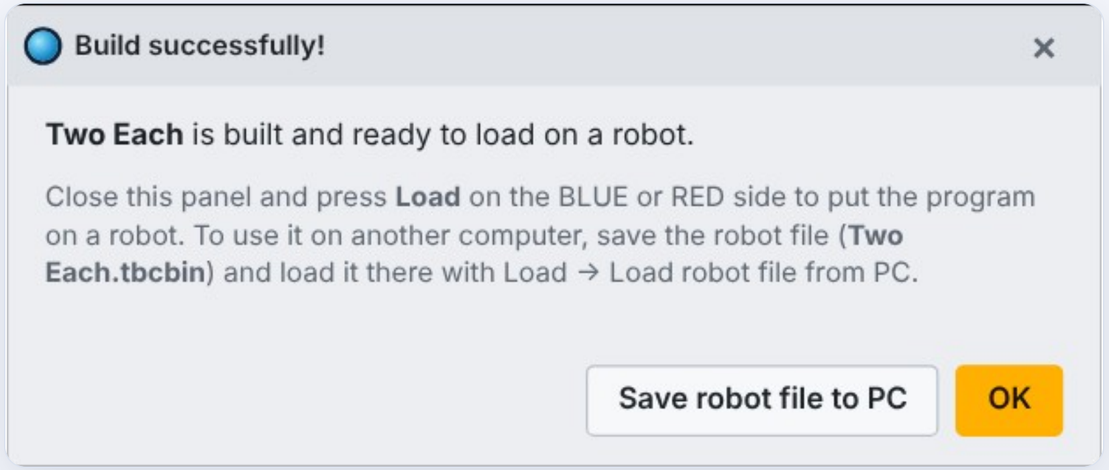
Load **3**/6 · Fe 1 Ag 2 Au 0

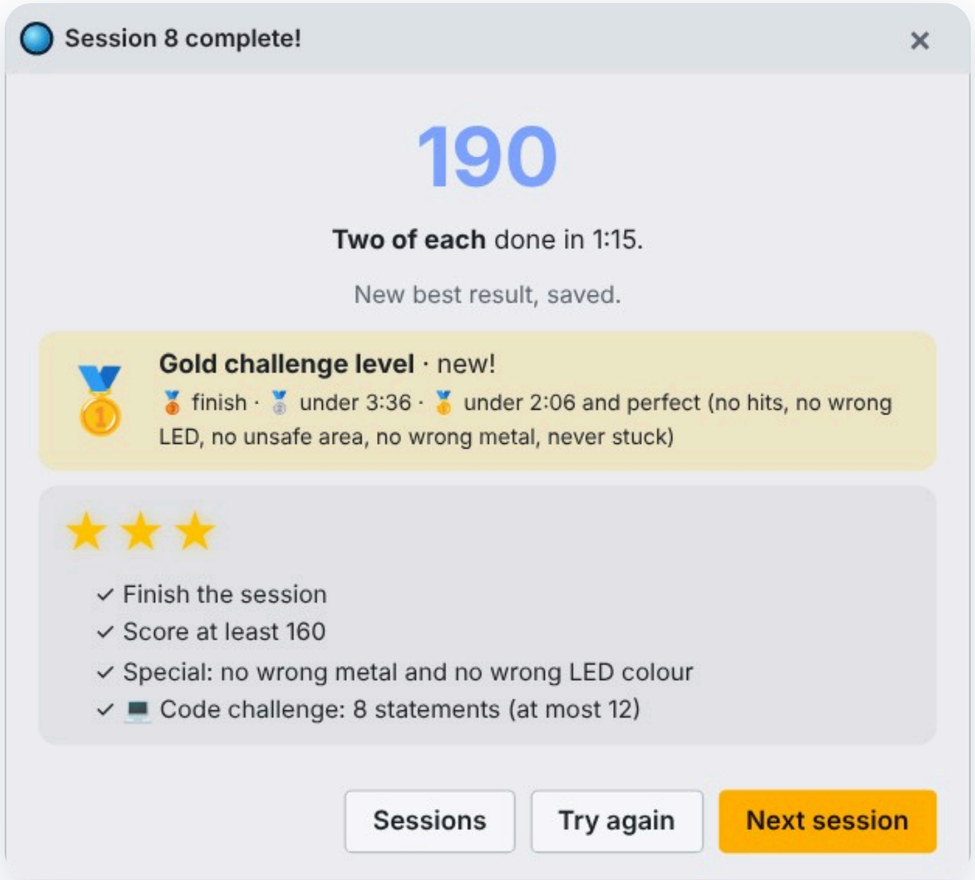
On **collecting 1.3 / 3 s**

BUILD AND LOAD

Name it, build it, load it

1. Name the program **Two Each**.
2. Check the 8 statements are in order.
3. Press **Build**, then **OK**.
4. Press **Load** on the BLUE side.





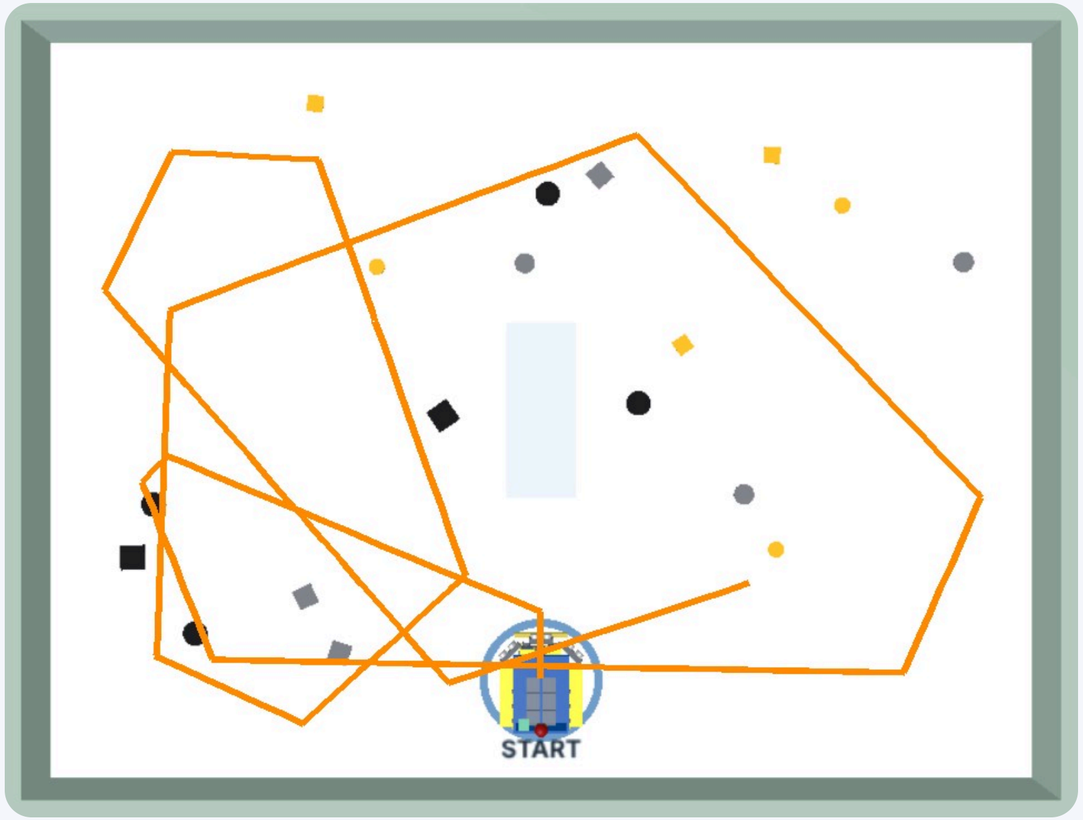
RESULT · TESTED

190 points in 1:15 · Gold

Score · 2×10 + 2×15 + 2×20 = 90, plus 100 = **190**. No penalties.

Medals · Silver: under 3:36 · Gold: under 2:06 and perfect.

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



THE PATH · RECORDED

Same bounces, smarter choices

Order in our run: iron, silver, silver, iron, gold, gold.

Compared with no counting: **+20 points** and 6 s faster.

Built-in counters: **less code, always right**.

DID NOT WORK?

Check first, then tune carefully

What you see	What to do
Build error in a statement	Spelling: **LoadedIron**, **LoadedSilver**, **LoadedGold**
A third of one metal collected	That statement’s Advanced Condition is missing or uses the wrong metal
Never collects silver or gold	Check the R, G, B ranges against the colour table
Still have the P7 variables	That is fine, but they are not needed here
Old program still on the robot	**Build** again, then **Load** on BLUE

NEXT TIME

Presentation 9 · Collect and deposit

Fill the robot with **LoadedObjects**, then empty it in the green deposit area.