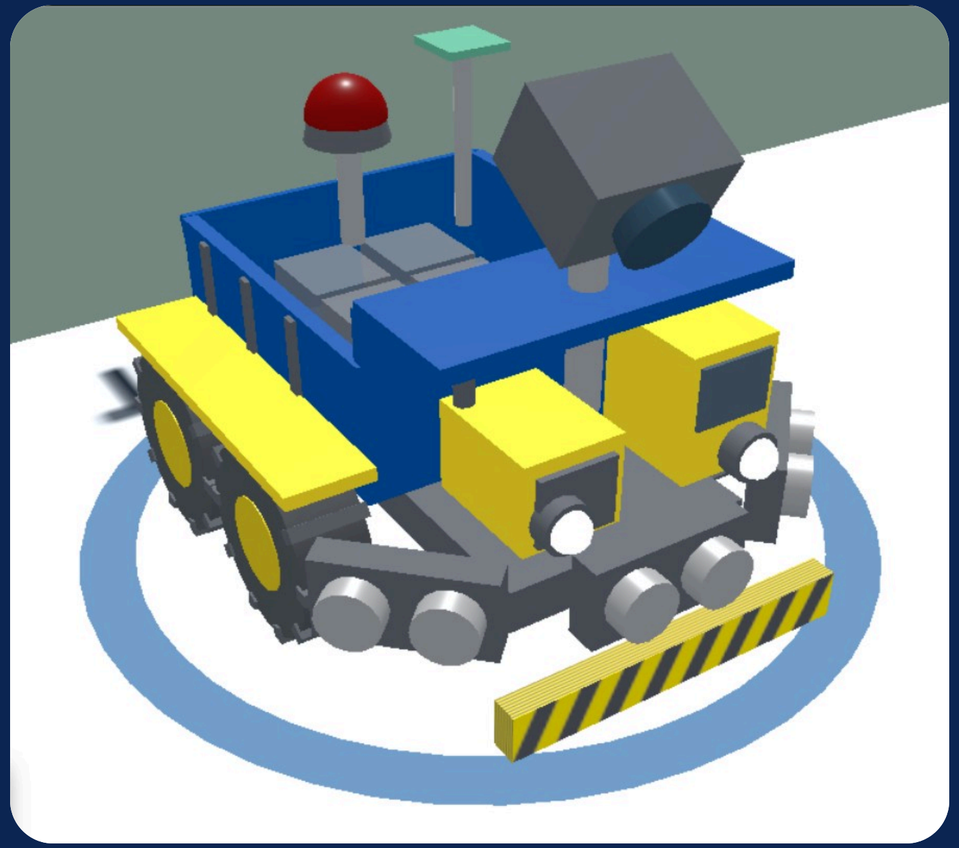
UNIVERSITY · PRESENTATION 6 · LEARNING WITH PYTHON

Pond by camera

Find the water pond in the camera image, mine only inside it, and show a blue LED there.

TheBotCup 2026 · AiBotCup Simulator · Advanced



RECAP

From objects to places

Session 14

Gold hunt

A colour test on every pixel finds gold.

Session 15

Oil or Rocks

The shape of a black blob: empty centre or ink.

Session 16 · last time

Line up and collect

Turn until the object is at the bottom centre, then drive.

Same camera, same 24 × 18 image, same **game1()** every 60 ms. New today: the camera also tells us **where we are**.

PART 1

The mission: Session 17

Mine 3 metals inside the water pond

Session 17 · Pond by camera
Mine 3 metals inside the water pond. Recognise the pond in the Camera image and use a BLUE LED there.
Pond metals are worth double. 3 mined: +150.
In the pond: 0/3
Camera tip: Pond pixels are about R < 60, G 140–200, B > 210. If several bottom-row pixels are pond blue, the object is in water: LED_B = 255, LED_R = LED_G = 0.

Sessions | Restart session

SESSION 17 · POND BY CAMERA

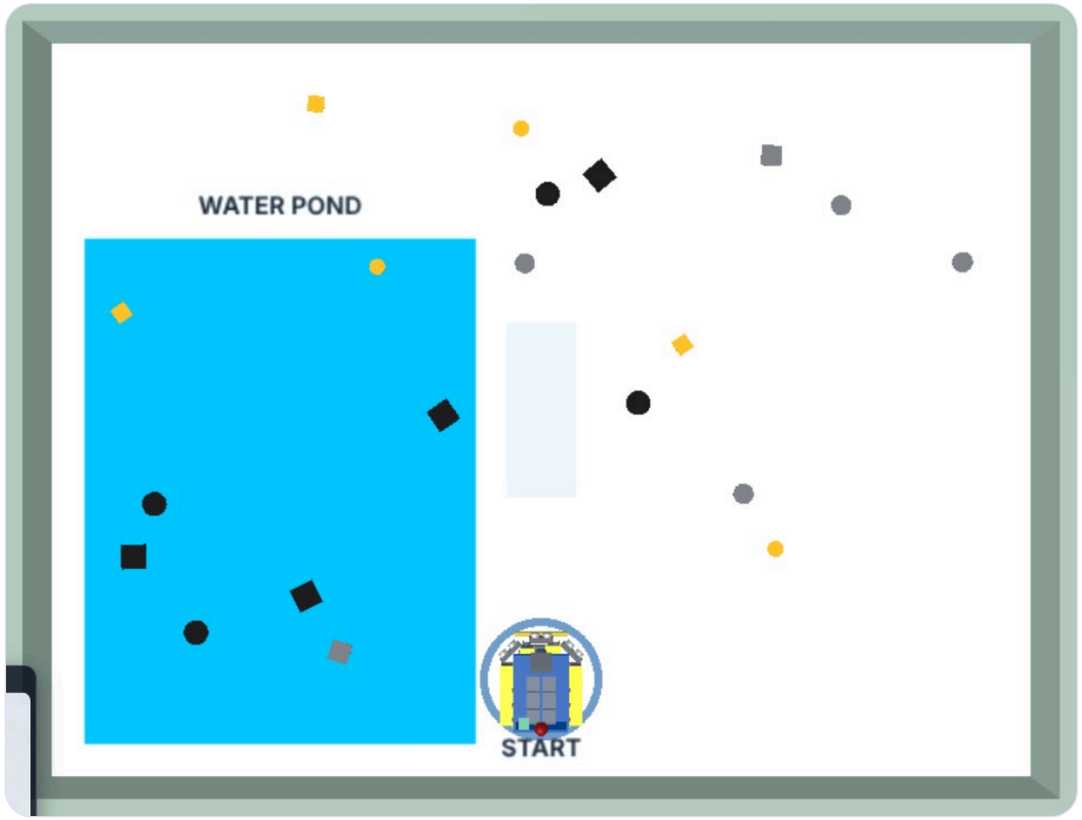
Mine in the water, and say so with blue

Only metals **inside the pond** count: a land metal is refused.

Pond metals are worth **double**: iron 20, silver 30, gold 40.

3 mined in the pond: **+150** · time limit **5:00**.

Collect in water with a **BLUE** LED: LED_R = 0, LED_G = 0, LED_B = 255.



THE FIELD

A pond on the left, a block in the middle

Pond: x 8–104, y 8–132 cm, with 8 metals in it.

Land: 12 more metals. Leave them.

A **block** right in front of START: (112–128, 70–110).

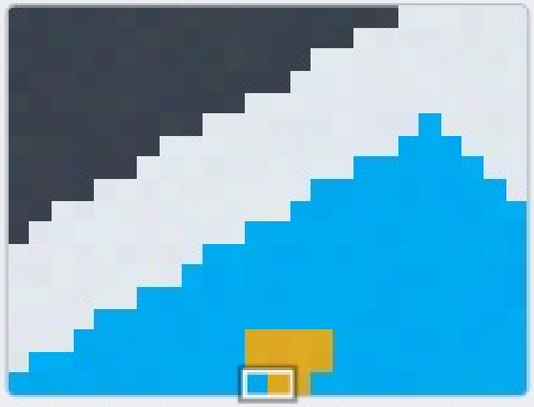
START at (120, 24), facing north (Compass 0).

PART 2

Seeing water

Three colour tests on one camera image

Camera `Camera[18][24]`



top = far · bottom = in front of the robot · white box = where collecting counts

THE CAMERA · A REMINDER

24 × 18 pixels of floor

Camera[row][col] = [R, G, B], each 0–255.

Row **0** is far (≈ 38 cm ahead), row **17** is near (≈ 8 cm).

Column **0** is left, column **23** is right.

The **white box**: the two bottom-centre pixels, [17][11] and [17][12]. Collecting counts there.

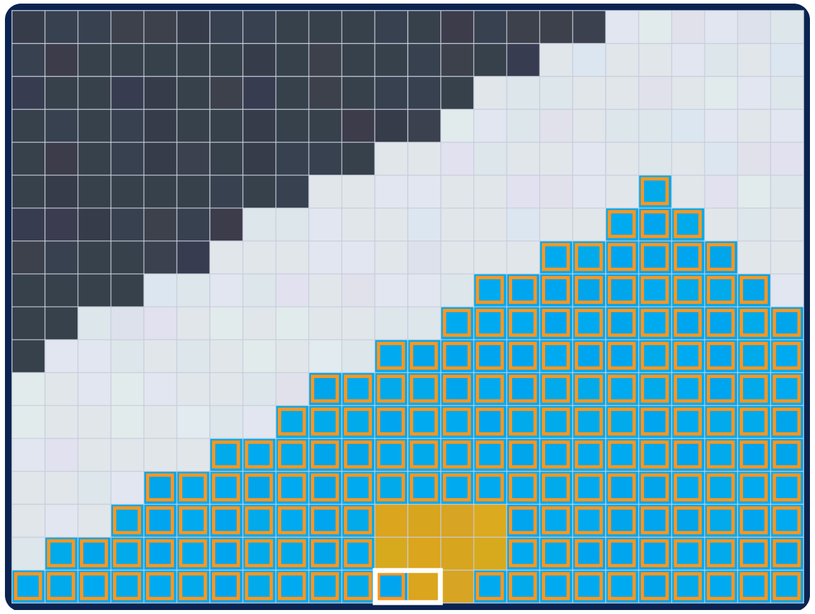
WHAT THE CAMERA READS

Water has its own colour

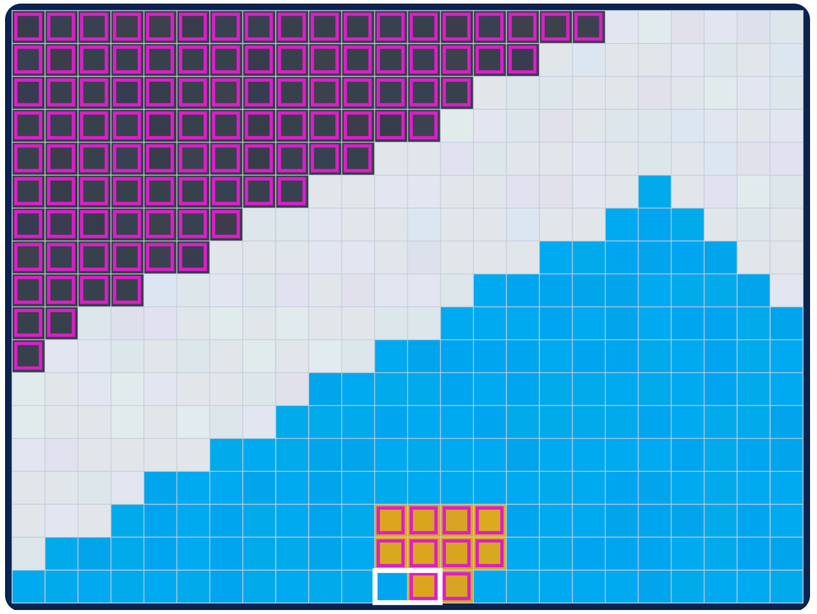
Surface	R, G, B (measured)	Test
Water pond	0, 171, 243	**R < 60 · G > 130 · B > 200**
White floor	226, 229, 239	all three high
Iron	25, 23, 25	all below 60
Silver	110, 113, 118	all near 110
Gold	220, 169, 34	R > 195 · B < 80
Wall or block top	112, 118, 126	grey, like silver

ONE IMAGE · TWO MASKS

Where is the water, where is the metal?



Pond pixels outlined: 170 of 432.



Metal-like pixels: gold at the bottom, wall at the top.

In the water? Pond pixels in the 3 bottom rows: 58 of 72.

Where is the pond? The average column of all pond pixels.

Nearest metal? The lowest row with a metal pixel.

FROM A COLUMN TO TWO WHEELS

Steer by the error from the centre

err = col − (CAM_W − 1) / 2
The centre is 11.5. err < 0: target on the left; err > 0: on the right.

WheelLeft = speed + 0.25 · err
WheelRight = speed − 0.25 · err
A target on the right speeds up the left wheel.

Target column	err	Wheels (speed 2)
4	−7.5	0.1 / 3.9 · turn left
12	0.5	2.1 / 1.9 · almost straight
20	8.5	4.1 / −0.1 · turn right

PART 3

Step by step in Python

One scan of the image, then one decision

THE PLAN

One scan, five rules in order

1. Wall or block ahead (US_Front < 15) → **turn** for 0.6 s.
2. Fewer than 8 water pixels under us → **steer to the pond** (or turn left to find it).
3. Metal at the bottom centre → **collect with a blue LED**.
4. Metal anywhere in view → **steer to it**.
5. Nothing in view → **explore** the pond on a gentle curve.

The first rule that is true wins, as in every Python if chain we have written.

STEP 1 · LINES 3–7

Two colour tests: pond() and thing()

```
3  def pond(p):                          # water-pond blue
4      return p[0] < 60 and p[1] > 130 and p[2] > 200
5
6  def thing(p):                         # iron, silver or gold
7      return max(p[0], p[1], p[2]) < 150 or (p[0] > 195 and p[2] < 80)
```

pond(p): little red, lots of green and blue.

thing(p): dark or grey (max < 150), or gold.

p is one pixel: p[0] red, p[1] green, p[2] blue.

STEP 2 · LINES 9–18

Two actions: collect() and steer()

```
 9  def collect():
10      WheelLeft = 0; WheelRight = 0
11      LED_1 = 2                                  # flashing
12      LED_R = 0; LED_G = 0; LED_B = 255          # BLUE: mining in water
13      Duration = 3200
14
15  def steer(col, speed):
16      err = col - (CAM_W - 1) / 2                # < 0: left of centre
17      WheelLeft = speed + err * 0.25
18      WheelRight = speed - err * 0.25
```

collect(): stop, flash (LED_1 = 2), **blue**, hold 3.2 s.

steer(col, speed): the error rule from Part 2.

STEP 3 · LINES 20–32

One loop over all 432 pixels

```
20  def game1():
21      LED_1 = 0
22      mid = CAM_W // 2
23      water = 0; wn = 0; wsum = 0; row = -1; col = 0
24      for y in range(CAM_H):
25          for x in range(CAM_W):
26              p = Camera[y][x]
27              if pond(p):
28                  wn += 1; wsum += x
29                  if y >= CAM_H - 3:
30                      water += 1
31              elif thing(p) and y > row:
32                  row = y; col = x
```

wn, wsum: how many pond pixels and their columns.

water: pond pixels in the 3 bottom rows.

row, col: the lowest metal pixel = nearest.

STEP 4 · LINES 33–45

The decision chain

```
33      if US_Front < 15:                                   # wall ahead
34          WheelLeft = -2; WheelRight = 2; Duration = 600
35      elif water < 8:                                     # not in the pond yet
36          if wn > 0:
37              steer(wsum / wn, 3)
38          else:
39              WheelLeft = -2; WheelRight = 2
40      elif thing(Camera[CAM_H - 1][mid - 1]) or thing(Camera[CAM_H - 1][mid]):
41          collect()
42      elif row >= 0:
43          steer(col, 2)
44      else:
45          WheelLeft = 3; WheelRight = 3.3                 # explore the pond
```

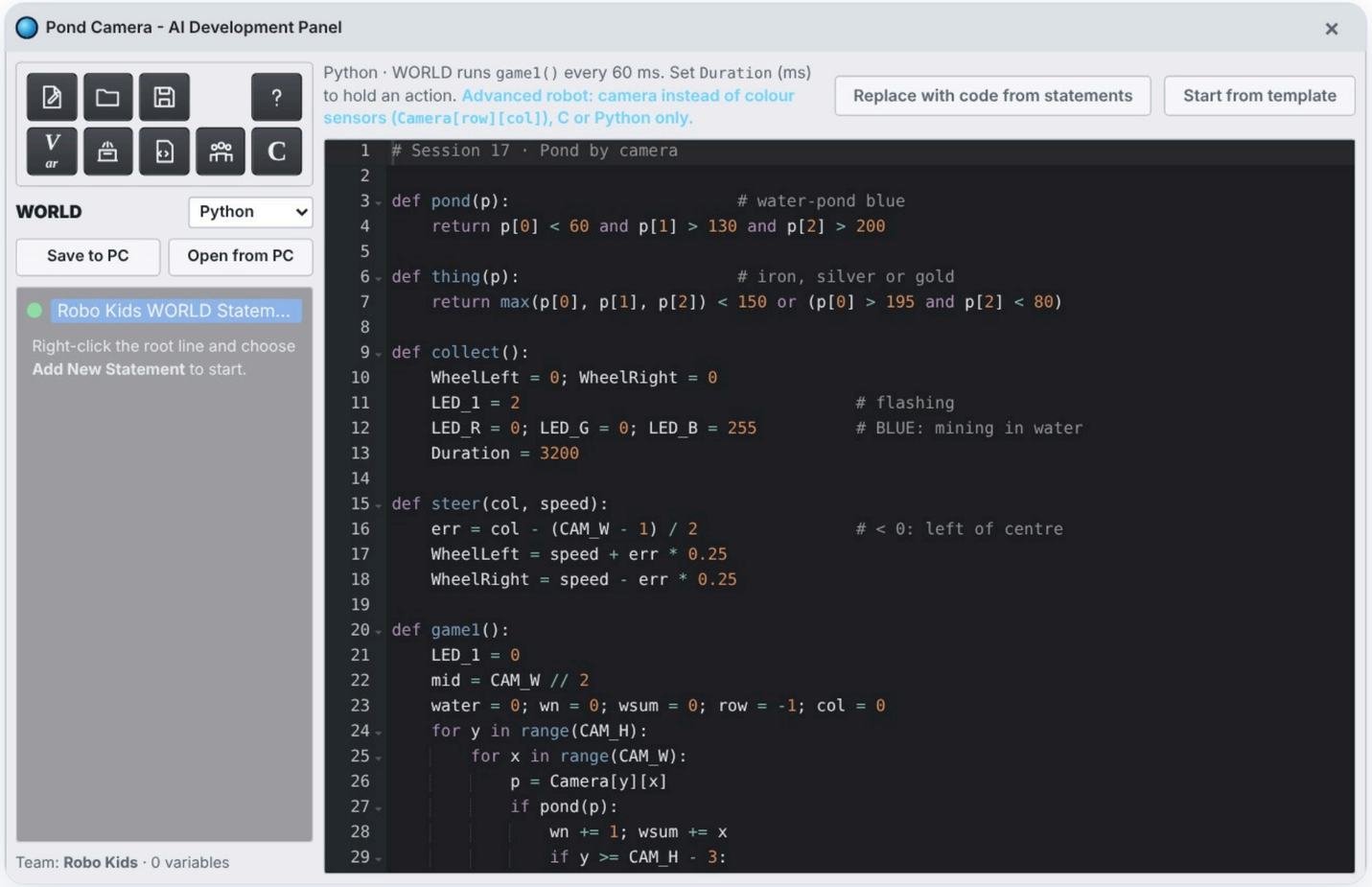
Not in the pond: **steer(wsum / wn, 3)** to the water.

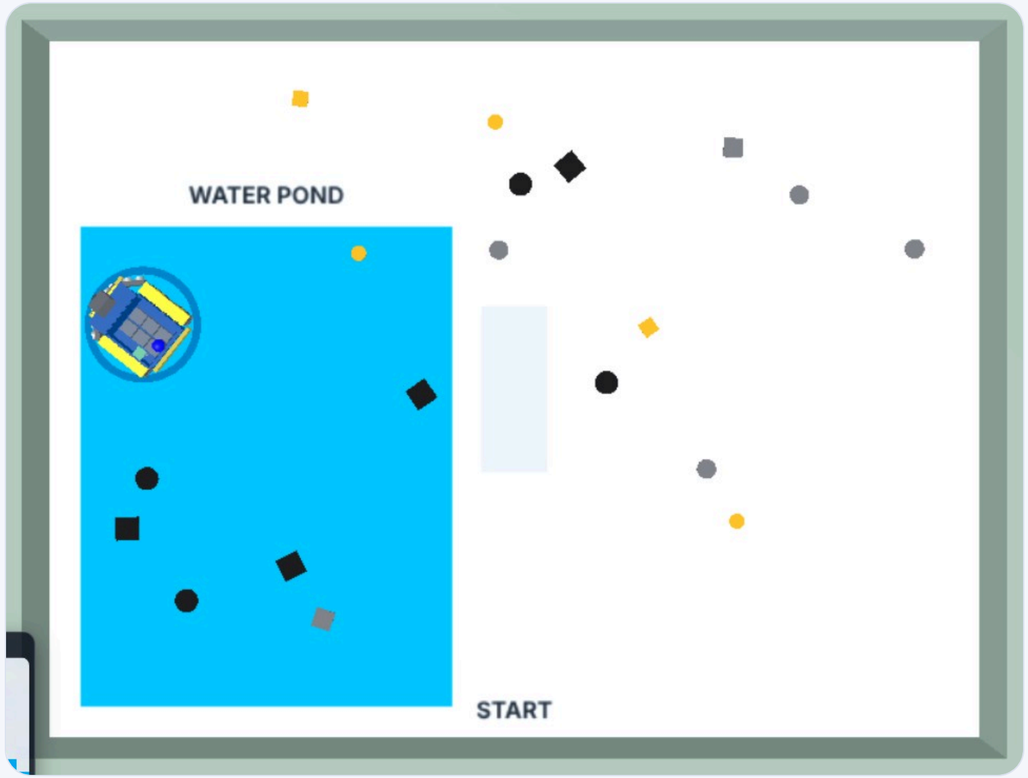
The bottom-centre test uses row **CAM_H − 1**, columns **mid − 1** and **mid**.

BUILD AND LOAD

The whole program, built

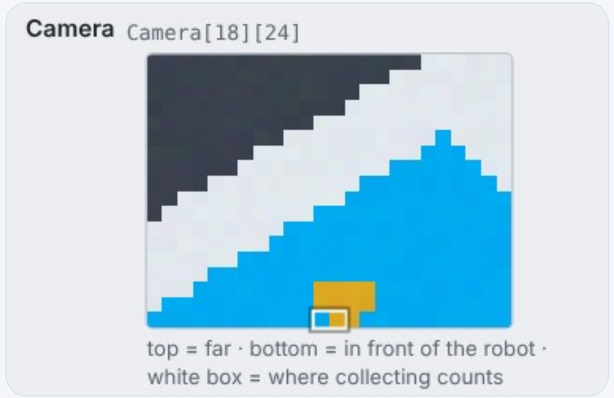
1. WORLD menu: **Python**.
2. Type the program, name it **Pond Camera**.
3. Press **Build**, then **OK**.
4. Press **Load** on the BLUE side.





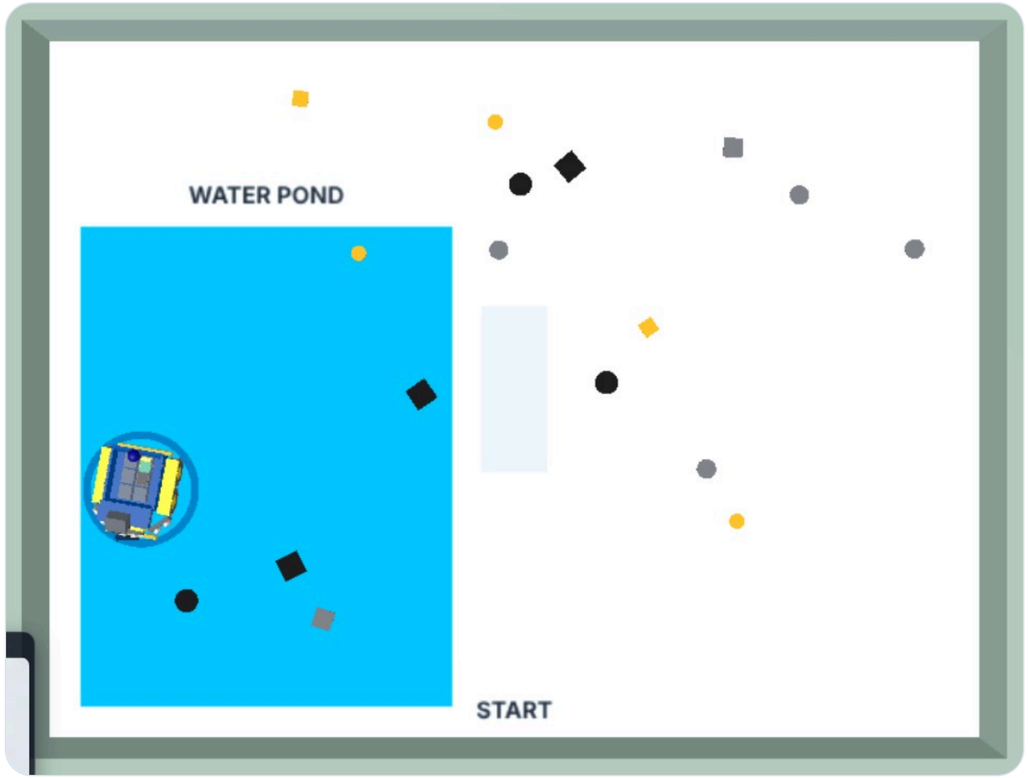
RUN · 9.5 S

The first catch: gold in the water



At (**24, 107**), Compass **52**: **58** water pixels in the bottom rows.

Bottom centre: **0, 168, 240** and **222, 168, 32** (gold) → collect(), blue LED.



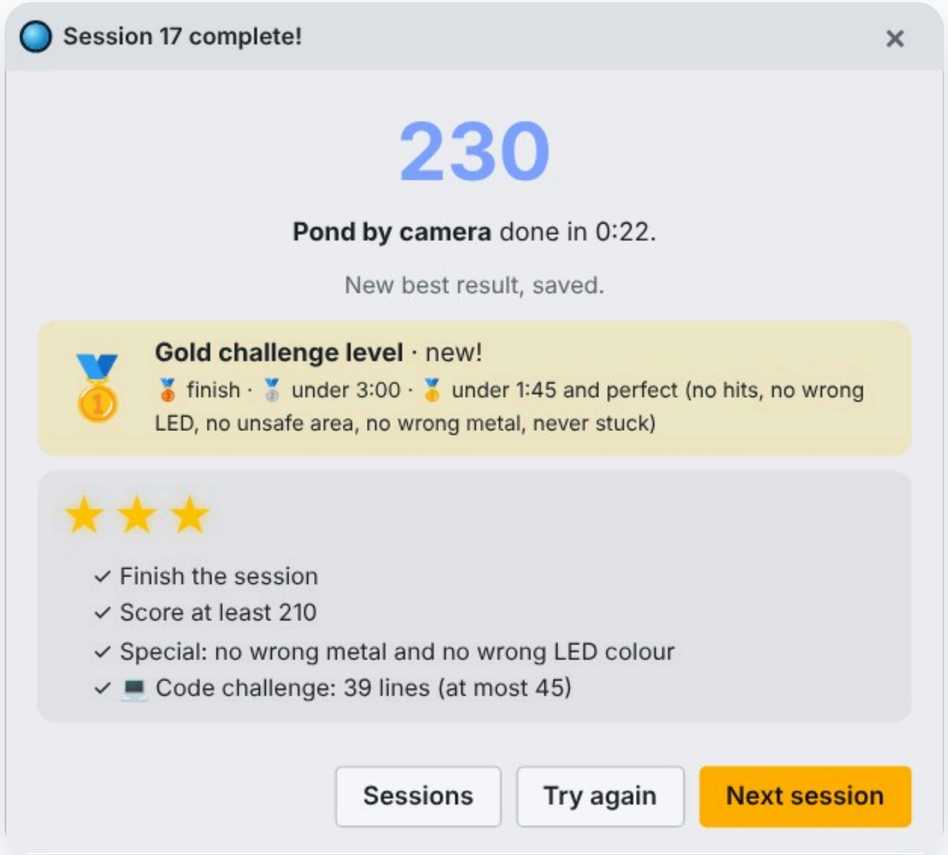
RUN · 21.9 S

Third metal, mission done



Two on board, an iron at the box: **23, 23, 24**.

Water all round: **57** of 72 bottom pixels.



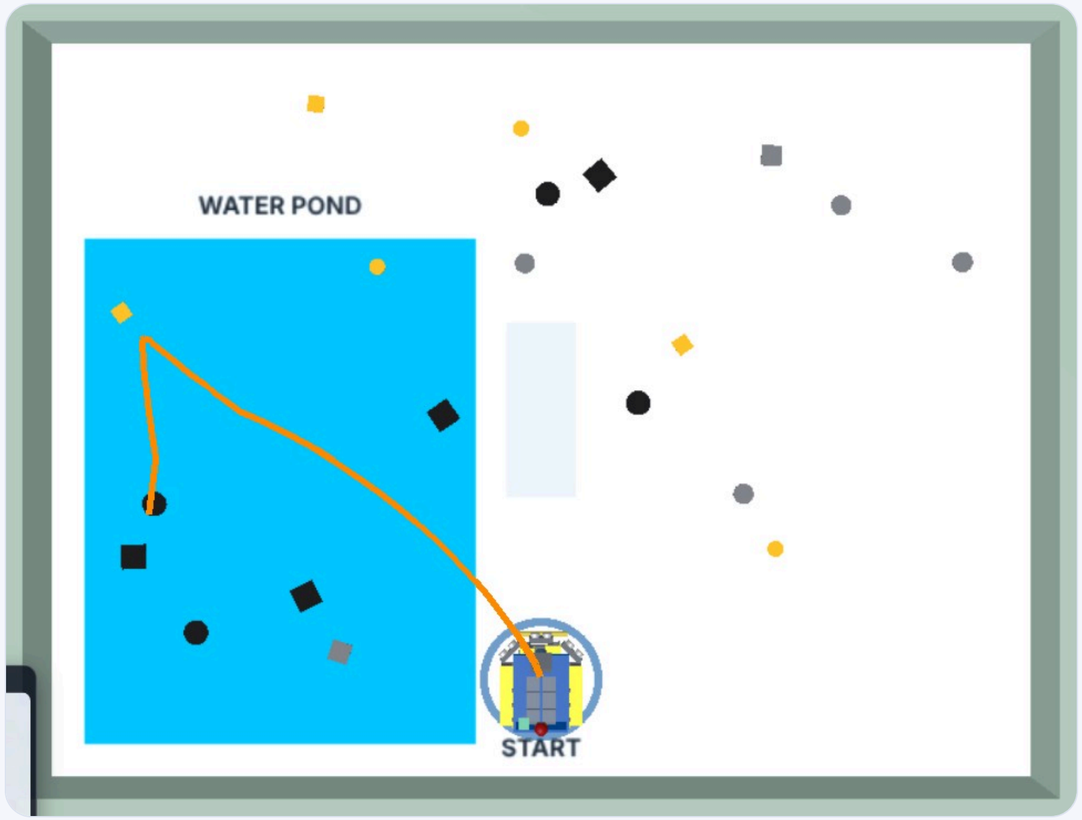
RESULT · TESTED

230 points in 0:22 · Gold

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

Stars · finish · score 210 · no wrong metal and no wrong LED colour.

Code challenge · at most 45 lines. We used 39.



THE PATH · RECORDED

Straight to the water, then metal to metal

No searching on land: the camera saw the pond from START.

Inside the pond the robot steered from one metal to the next.

A colour mask turns a picture into a **place**: in the water or not.

DID NOT WORK?

Check first, then tune carefully

What you see	What to do
Collects on land, “wrong metal”	The water test needs **8 or more** bottom pond pixels before collecting
Red LED in the water	collect(): **LED_R = 0, LED_G = 0, LED_B = 255**
Drives round the block forever	Steer to the pond: use **wsum / wn**, the average water column
Chases the grey wall in the pond	US_Front < 15 turns it away; the next metal takes over
Build error with a line number	Check the colons, the indentation and the [ ] of Camera[y][x]]

NEXT TIME

University Presentation 7 · Avoid by camera

Five unsafe areas between START and FINISH: see their **red warning bands** and steer around them.