Start here

Plato Pod puts small robots on a shared arena that students drive from a browser, in plain English, or from their own Python code. Four demo rooms are open to anyone with the codes below: no account, no install. A teacher who wants rooms of their own asks for an invite.

The four demo rooms

Each room is open all the time and keeps its own setup. Join with its code at arena.platopod.com/join, or follow its link. The courses are examples, from UNSW Canberra, of where each room fits: any class at any level can use any room.

Room	Example course	Code	What you do	Teacher needed?
Problem solving: first programs in Python	ZEIT1307 Computational Problem Solving	TLQGNR	Drive a pod in a square, find out why it misses, fix it with feedback	No
Control engineering: PID line following	ZEIT3222 Control Engineering	GFYFYJ	Follow tape with a line sensor; tune P, I and D against a real delay	No
Artificial intelligence: a swarm that searches	ZEIT4150 Fundamentals of Artificial Intelligence	UUDPYD	Five pods search for hidden targets: no plan against a plan	No
Surveillance: outwit the sensors in plain English	ZEIT8205 Fundamentals of Surveillance Technologies	UBACYD	Two teams: one hides sensors, the other writes orders to get past them	Yes, to start each round

The first three run from a Python notebook, at home or in class. The fourth is played live: a teacher at the console starts the rounds.

Try it in five minutes

All you need is a laptop with a browser and Python 3.10 or later, or a Google account for Colab.

1. Download the room's notebook (below) and open it in Jupyter, VS Code, or Google Colab (File, Upload notebook).
2. Put your name in the second code cell (`MY_NAME`). The room's code is already there.
3. Run the cells from the top. The first fetches the Plato Pod client from the server; the second joins the room, gives you a pod, and prints a link to watch it in 3D.

4. Open that link beside the notebook and watch your pod do what your code says.

Notebook	Room
first-programs.ipynb	Problem solving: first programs in Python
pid-line-following.ipynb	Control engineering: PID line following
swarm-search.ipynb	Artificial intelligence: a swarm that searches

Your name keeps your pod: run the notebook again tomorrow with the same name and you get the same pod back. A pod nobody has used for 20 minutes leaves the room, which frees its place. The demonstration rooms take six people at once (the swarm room three, as each brings five pods), so everyone who gets in has a pod that answers; a room of your own takes 12.

Without a notebook, you can still look in: join from arena.platopod.com/join with the code and a name, and the page links to your pod on the table and in 3D.

First programs in Python

Room **TLQGNR**. An example of where it fits: ZEIT1307 Computational Problem Solving, at UNSW Canberra. Notebook: first-programs.ipynb.
A first program that moves something real, then the first lesson in why programs that control things need feedback.

What students do

1. Drive straight for 3 s at 0.1 m/s and measure how far the pod went.
2. Turn a quarter turn by time: 0.8 rad/s for (π/2)/0.8 s.
3. A square by time: four sides, four turns. It does not close.
4. The same square by feedback: turn until the heading is right, drive until the distance is right.
5. Your turn: any polygon, a spiral, a star, and back to where you started.

Square	Missed its start by (three runs on the server, 2026-11-22)
By time	21 to 27 cm, heading off by 13° to 58°
By feedback on the pose	0.6 to 1.4 cm

The point for students: commands arrive late and unevenly and the pod speeds up and slows down, so a program that trusts the clock drifts, and one that looks at where the pod is does not.

Run it, start to end

You need a laptop with a browser and a steady connection: classroom or home Wi-Fi rather than a phone hotspot.

1. Download the notebook, first-programs.ipynb.
2. Open it in one of these:
   - **Google Colab**, nothing to install: go to colab.research.google.com, then **File**, **Upload notebook**, and choose the file.
   - **Jupyter** on your laptop, with Python 3.10 or later: in a terminal, in the folder you downloaded it to, run `python3 -m pip install notebook` once, then `jupyter notebook first-programs.ipynb`.
   - **VS Code**: open the file, accept the Python and Jupyter extensions it offers, and pick a Python 3.10 or later kernel at the top right.
3. In the second code cell, change `MY_NAME = "your name here"` to your name. The room's code is already there.
4. Run the first code cell. It prints `ready`: it has fetched the Plato Pod client from the server.
5. Run the second. It prints `You are pod` and a number, and a **Watch it** link. Open the link in a tab beside the notebook to see your pod in 3D.
6. Run the rest one cell at a time, from the top, reading each section before you run it.
7. When you are done, run the last cell, `arena.close()`. It gives your pod back to the room.

If the connection drops for a moment, the notebook connects again by itself and says so. If a cell stops with an error, run it again; if that fails too, choose **Restart** from the **Kernel** or **Runtime** menu and run from the top. Your name gets the same pod back. The room takes six people at a time.

PID line following

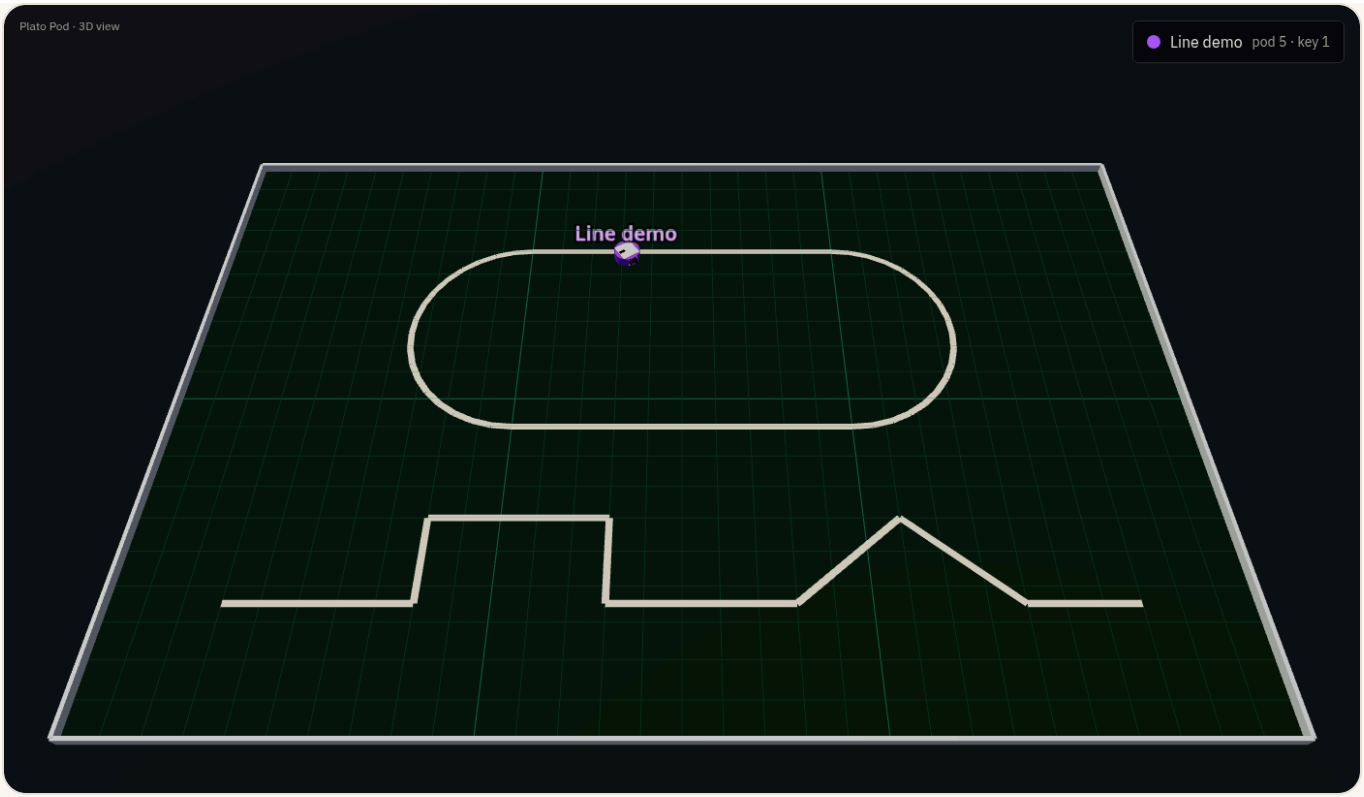
Room **GFYFYJ**. An example of where it fits: ZEIT3222 Control Engineering, at UNSW Canberra. Notebook: pid-line-following.ipynb.

Tape is drawn on the floor: an oval and a course with square corners and sharp bends. Each pod carries a line sensor, five reflectance cells under its nose that read 0 to 1 with noise, and reports where the line is across the pod (`offset_m` , positive to the left). Students write the controller.

1. The pod drives itself onto the oval by feedback on its pose, at a place of its own so several students share the course.
2. P only at 5 cm/s, then the same at 8 cm/s.
3. D, smoothed: the textbook cure for snaking, and why it does not cure it here.
4. I, capped against windup: the steady error on the curves.
5. Your turn: the fastest lap, predicting through the delay, the course with sharp bends, the five raw cells.

Controller (measured on the server)	Mean error	Line lost
P, kp = 40, 5 cm/s	2.5 mm	never
P, kp = 40, 8 cm/s	7.3 mm	never
PD, kp = 40, kd = 0.5	2.5 mm	never
PI, kp = 40, ki = 10	2.8 mm	never

The lesson is a real one: a reading is a tenth of a second old when it arrives and commands cross the internet, so the delay, not the gains, limits the speed, and D and I need a better signal than this one to help. A teacher can draw new courses on the console's Mission sheet (Set up activities).



The room in 3D while a notebook's pod follows the oval: the link the notebook prints.

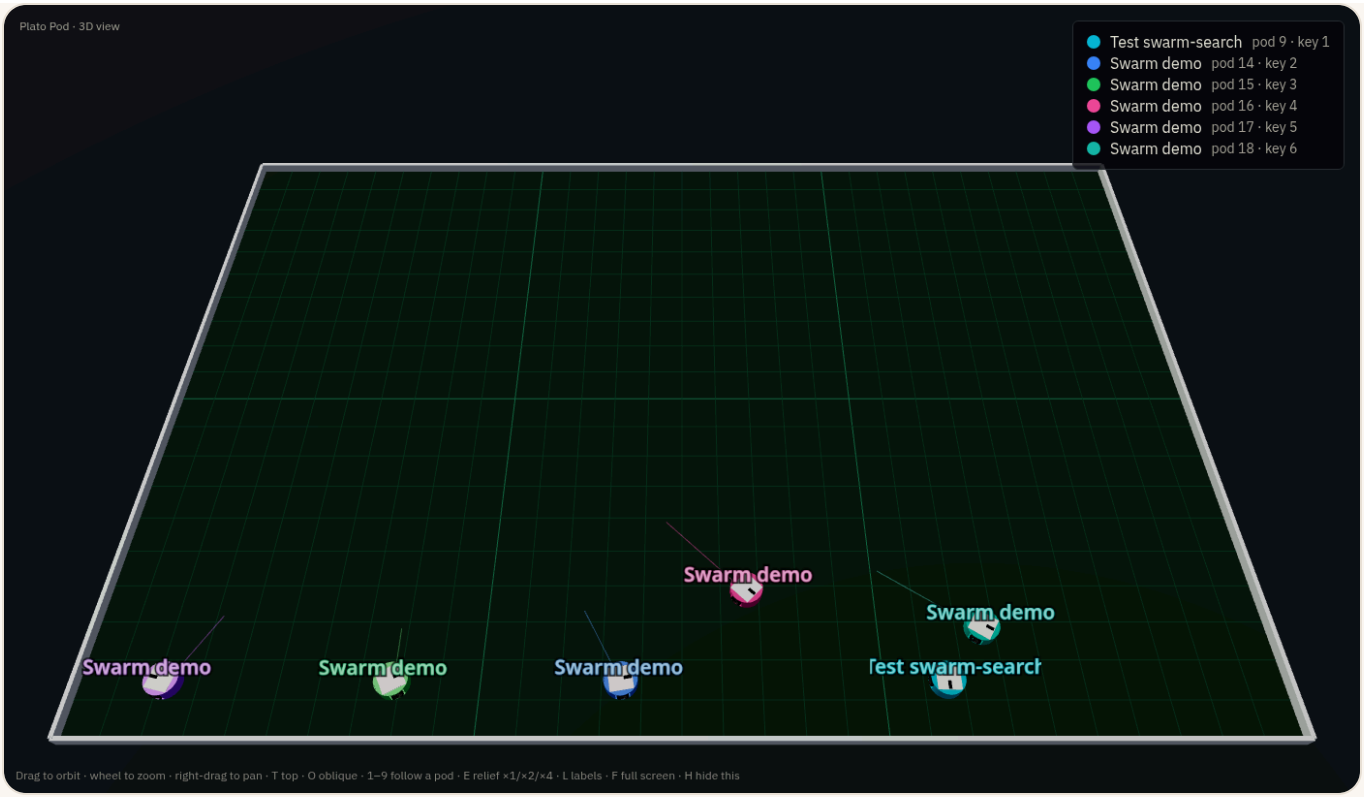
A swarm that searches

Room **UUDPYD**. An example of where it fits: ZEIT4150 Fundamentals of Artificial Intelligence, at UNSW Canberra. Notebook: swarm-search.ipynb.
Ten targets are hidden on a 3 m by 2 m floor; nobody, not even the console, shows where. One program commands five pods, each with a detector that reports a target within 15 cm. The swarm keeps one shared map of what it has found.

1. Join, and add four more pods to your swarm (`arena.spawn()`). They leave the room when your notebook closes.
2. No plan: each pod drives to random points for 90 s.
3. A plan: the floor split into one strip per pod, each swept in lanes 25 cm apart.
4. Your turn: compare the two, share out a strip when a pod fails, allocate strips by auction, and use what has been found to choose where to look next.

Search (measured on the server)	Targets found	Time
Random walk	4 of 10	90 s
Planned sweep	9 of 10	61 s

The sweep missed one target that sits where two pods' strips meet: a good first question for students.



One notebook's swarm of five early in its sweep, each pod starting its own strip. The targets are not shown anywhere.

Infiltration in plain English

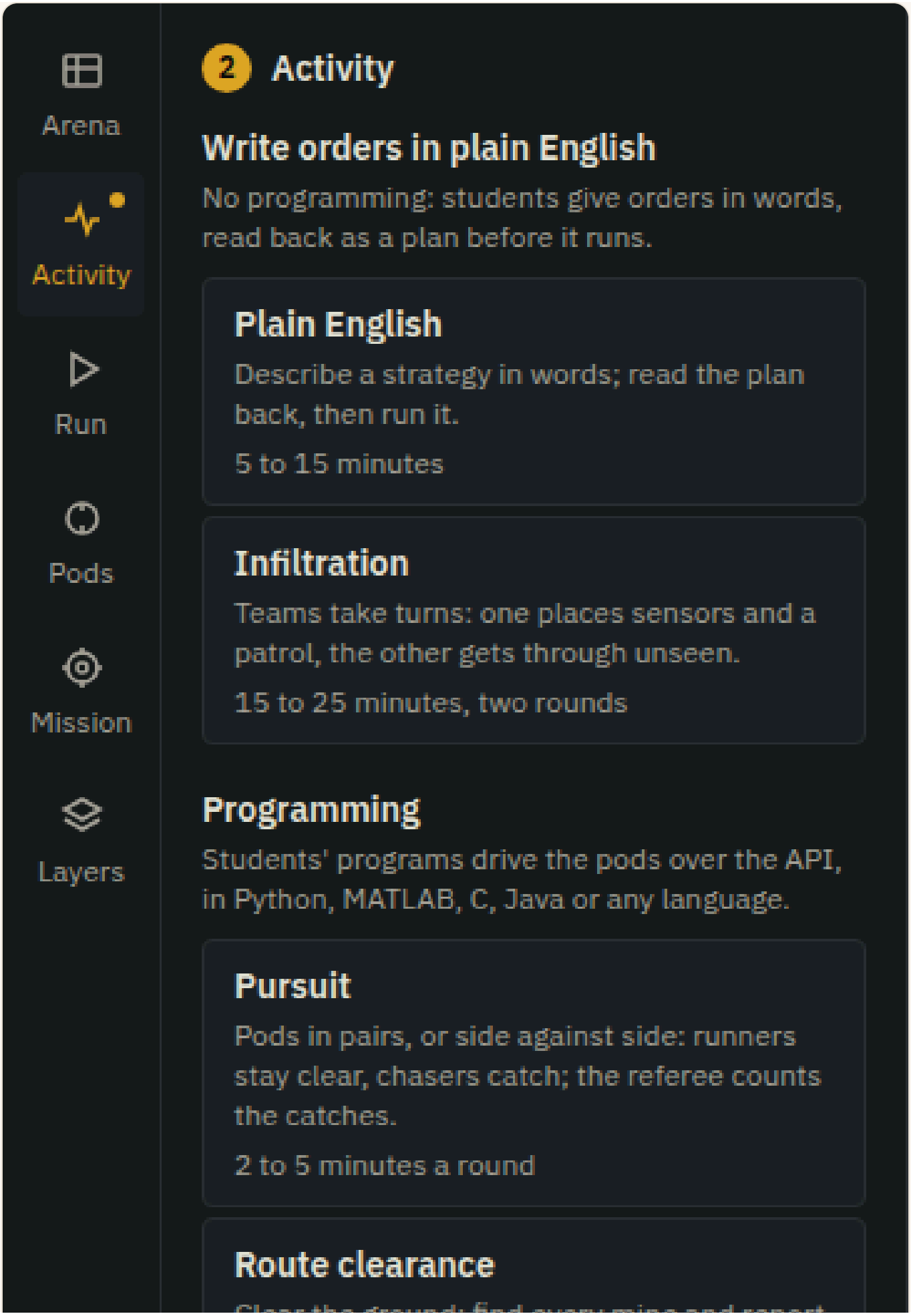
Room **UBACYD**. An example of where it fits: ZEIT8205 Fundamentals of Surveillance Technologies, at UNSW Canberra.

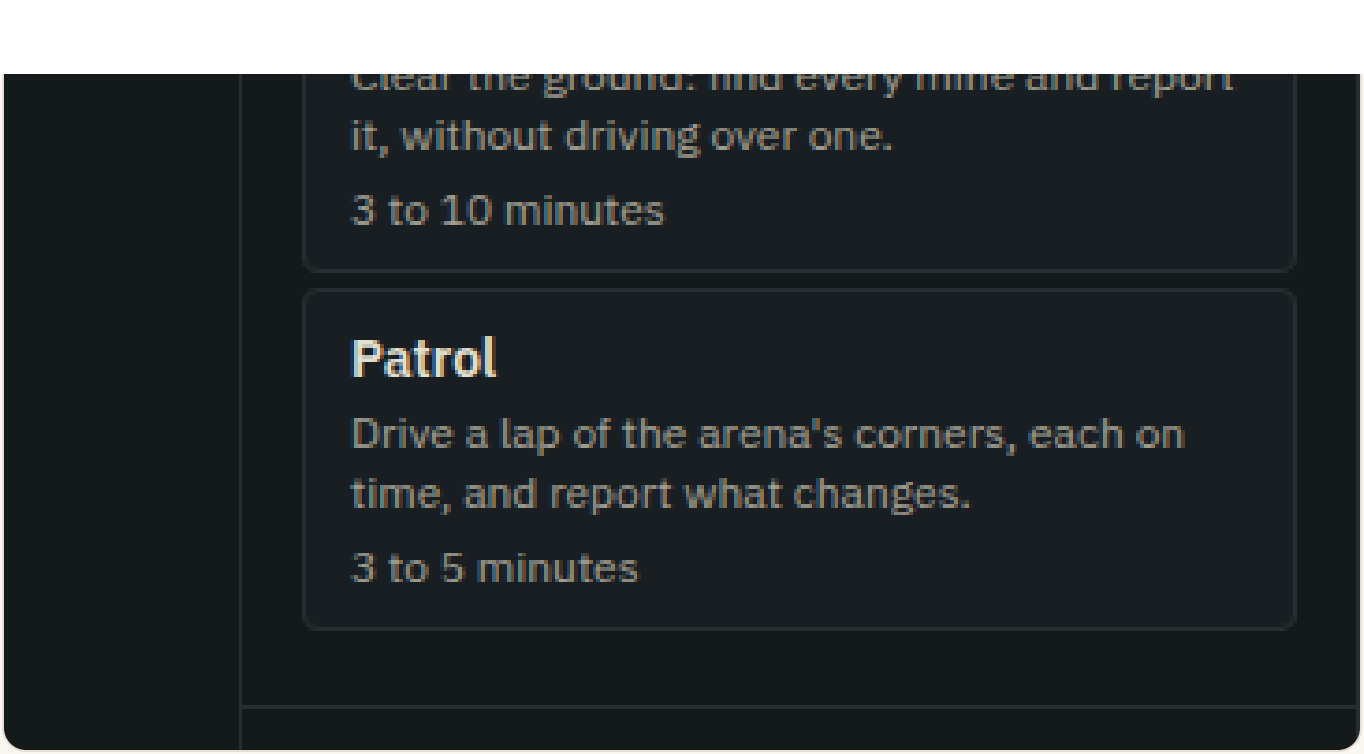
One team hides sensors and patrols; the other tries to get its pods to the objective unseen. Two rounds of up to 120 s, roles swapped; about 15 minutes in all. Orders are typed in plain English and turned into a plan.

It is played live, with a teacher at the console: the activity is already chosen in the surveillance demo room, where the presenter runs the console. In your own room, steps 1 and 2 choose it. The steps and pictures below are a two-team match from start to report. For a class that uses ATAK, the same match shows on ATAK devices: see ATAK.

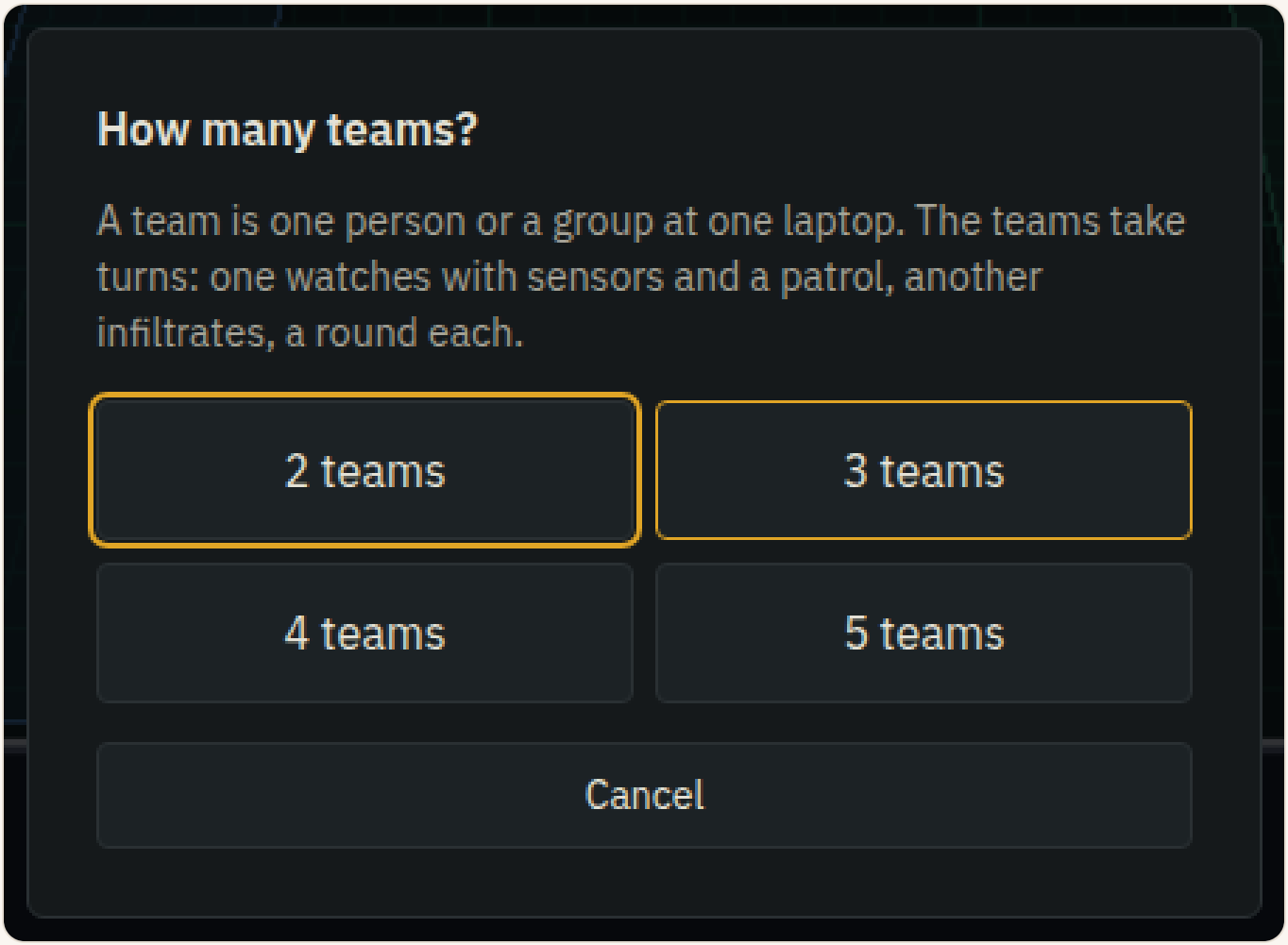
Teacher (big screen)

1. Click **Activity** on the left rail, choose **Infiltration**. A box asks how many teams: choose **2** teams.
2. Click **Run**, then **Put pods out** if the pods are not out yet (on an arena already drawn, choosing the teams puts them out). The amber button is always the next step. Four pods: A-patrol for the watchers, B1 to B3 for the infiltrators (in round 2, B-patrol and A1 to A3).

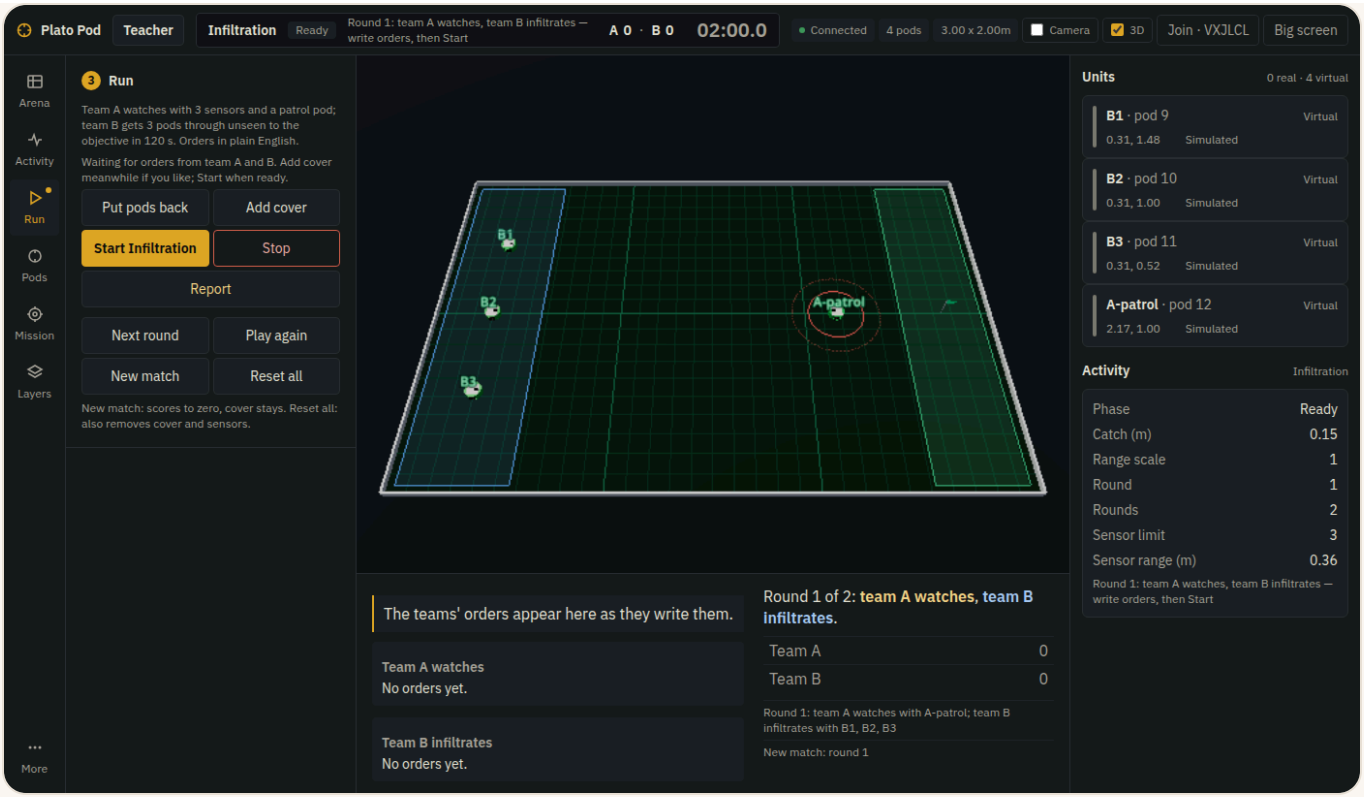

Arena
Activity
Run
Pods
Mission
Layers
2 Activity
Write orders in plain English
No programming: students give orders in words, read back as a plan before it runs.
Plain English
Describe a strategy in words; read the plan back, then run it.
5 to 15 minutes
Infiltration
Teams take turns: one places sensors and a patrol, the other gets through unseen.
15 to 25 minutes, two rounds
Programming
Students' programs drive the pods over the API, in Python, MATLAB, C, Java or any language.
Pursuit
Pods in pairs, or side against side: runners stay clear, chasers catch; the referee counts the catches.
2 to 5 minutes a round
Route clearance



Step 1: the Activity sheet. Infiltration is under Write orders in plain English.



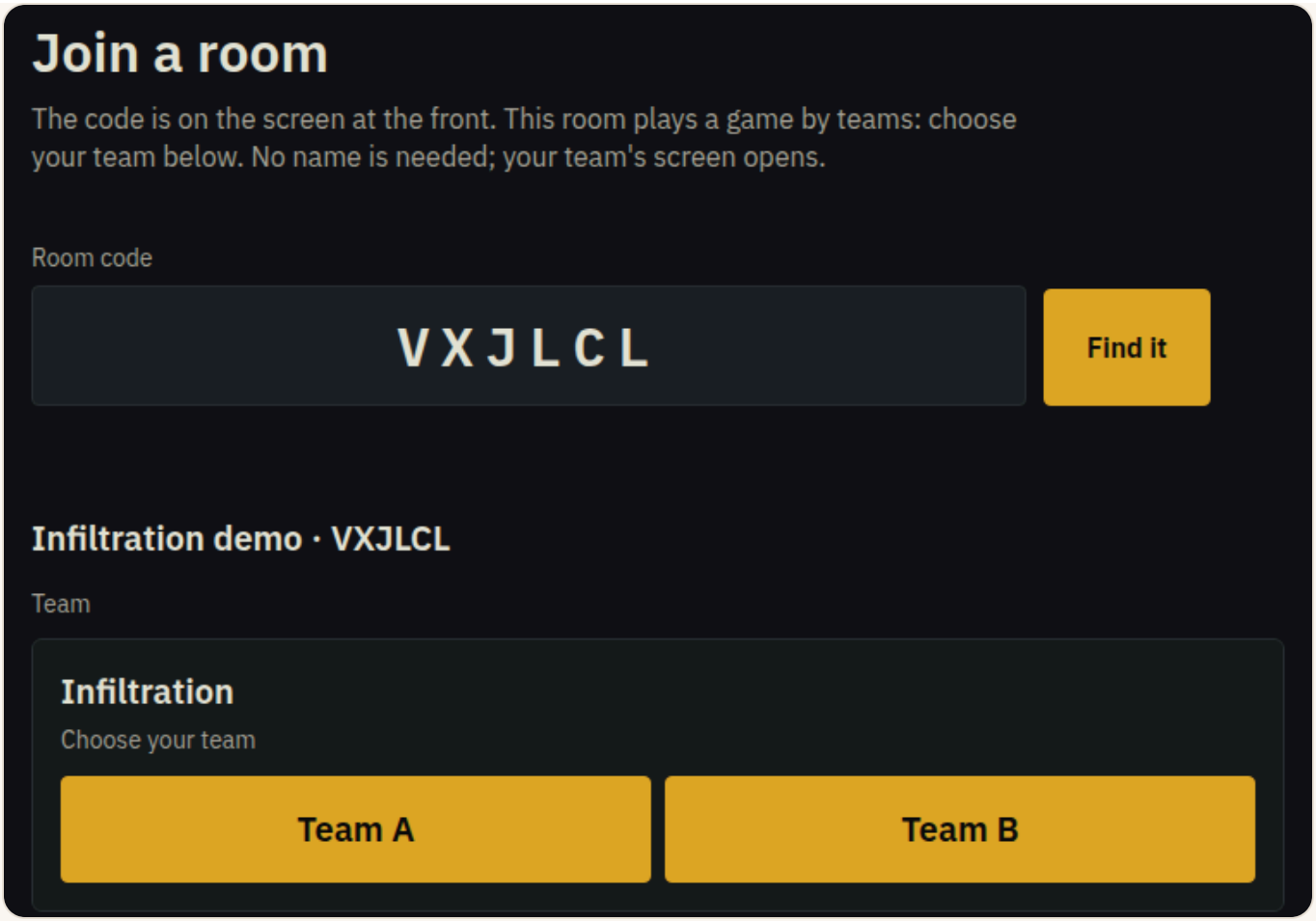
Step 1: choose 2 teams.



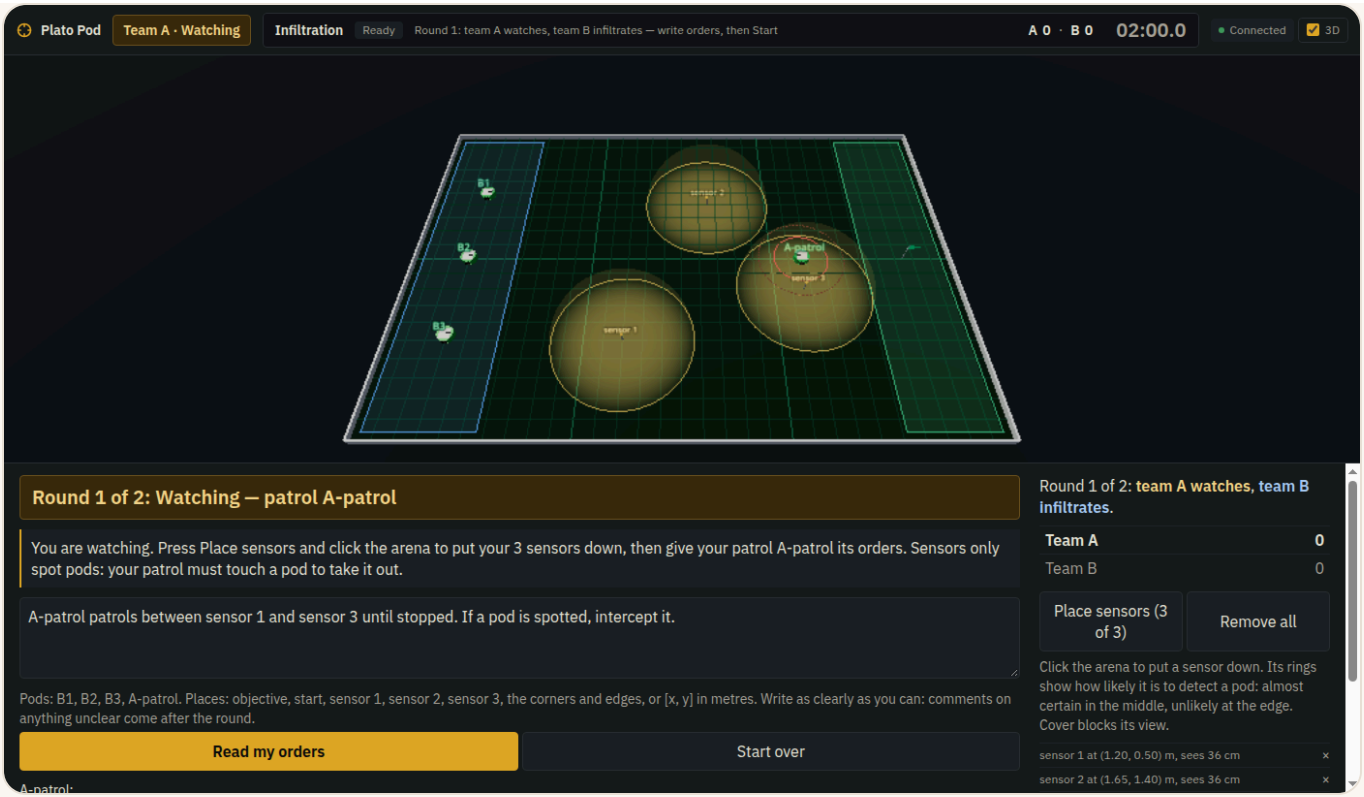
Step 2: the Run sheet with the pods out. The amber button is the next step, here **Start Infiltration**; after a round it moves to **Next round**. Infiltrators start in the blue strip on the left; the objective is the green strip on the right.

Each team (own laptop)

3. Open arena.platopod.com/join (or scan the QR code on the big screen), type the room code, click **Find it**, then **Team A** or **Team B**. The team's console opens.
4. The watching team clicks **Place sensors**, clicks up to three spots on the ground, then **Place sensors** again to finish. The other team cannot see them.
5. Both teams type their orders and click **Read my orders**. Check the read-back says what was meant. Example orders:
   - Watchers: "A-patrol patrols between sensor 1 and sensor 3 until stopped. If a pod is spotted, intercept it."
   - Infiltrators: "B1 goes along the bottom edge to the objective. B2 goes to the objective. B3 goes along the top edge to the objective."



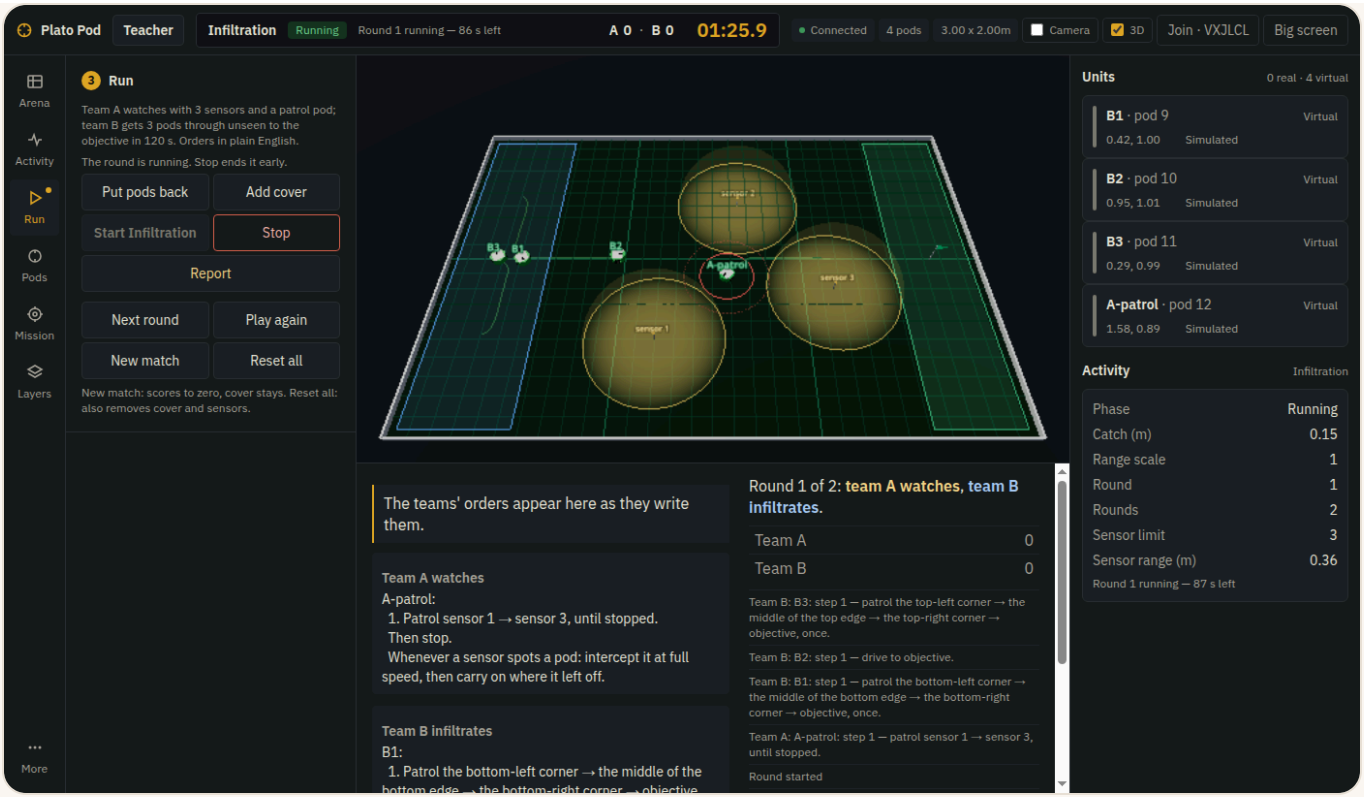
Step 3: the join page for a team game. No name is needed; each team clicks its own button.



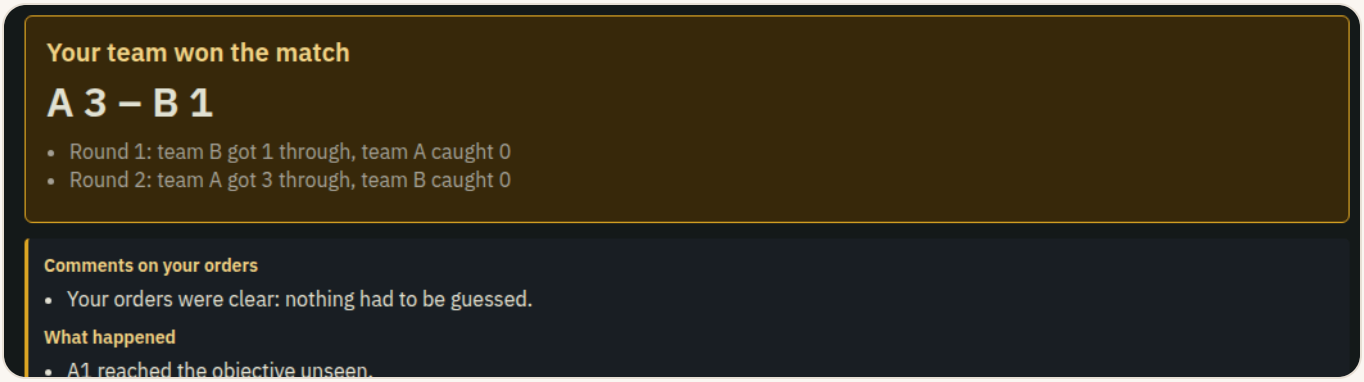
Steps 4 and 5, the watching team: three sensors placed (the yellow circles), orders typed and read back. The pod names and places the orders may use are listed under the box.

Teacher

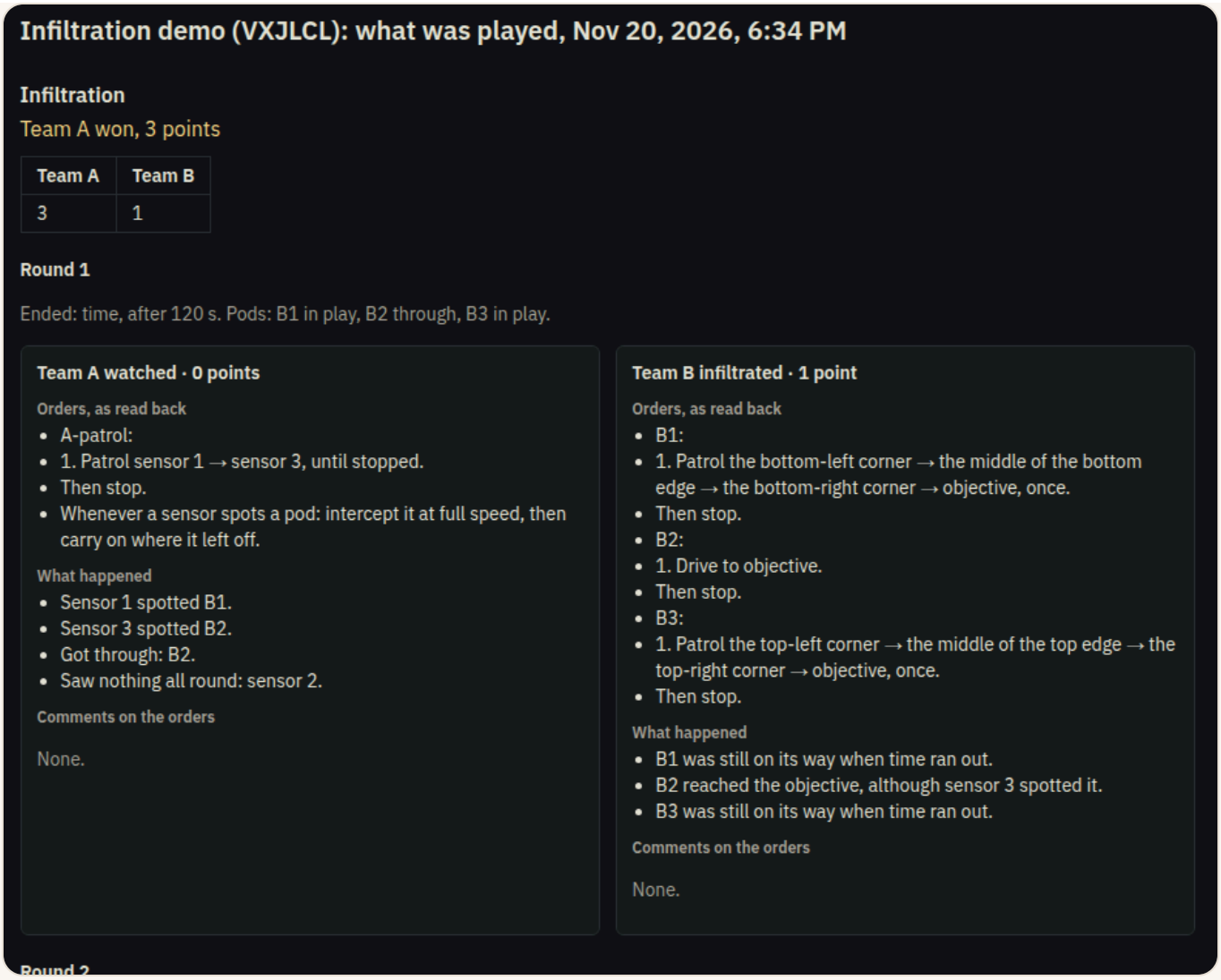
6. Click **Start** Infiltration once both teams' orders show in the middle panel. If a team has sent none, the console asks before it starts. The round ends by itself at 120 s, or sooner once every pod is through or caught; the clock then stops on the time that was left.
7. Click **Next round**: roles swap and the pods go back out. Repeat steps 4 to 6.
8. After round 2 every screen shows the result. **Report** opens the match write-up (orders, rounds, scores) to print or save.
9. **New match** to play again, or **Reset all** to clear.



Step 6: a round running on the big screen. The clock is top centre; the event log on the right says what each pod and sensor did.



Step 8: what each team sees after round 2: the result, each round's score, and comments on their orders.



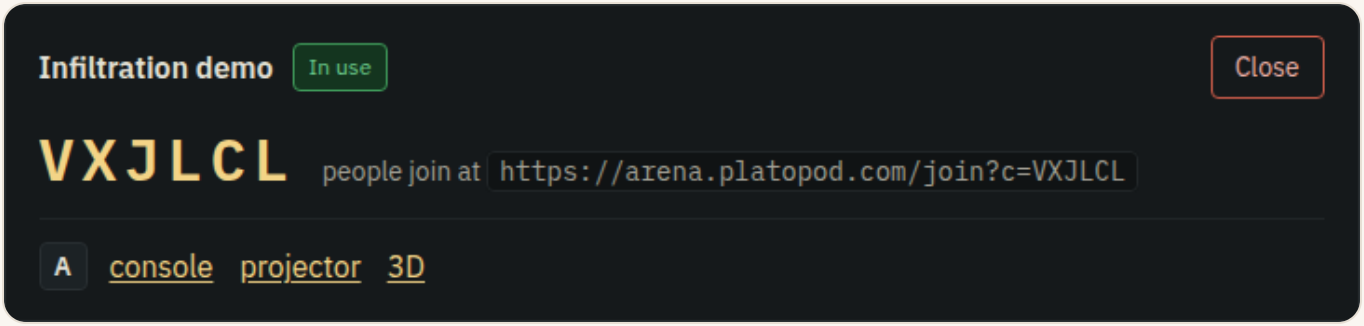
Step 8: the report, with **Print or save as PDF** and **Download the scores (CSV)** at the top.

Run your own room

While the service is new, rooms of your own are by invitation: write to a.lenskiyunsw.edu.au for an invite code, and say if your class will use ATAK. A room is yours alone, holds one arena per group of up to five students, and closes after 7 days nobody uses it.

1. Sign up at arena.platopod.com with the invite code. **My rooms** opens.
2. Type a name, choose how many arenas (one per group), and click **Make a room**.
3. On the room card, note the six-letter **room code**. Students type it on the join page, or in a notebook.
4. On the big-screen laptop, click **console** on the card. It opens signed in as the teacher; never share this link. Its **Join** button puts the code, the join address and a QR code on the screen.
5. Optional: **projector** gives a clean full-screen view for a second screen; **3D** the arena in three dimensions.

Then choose an activity, or set one up: see Set up activities.



The room card: the code players type, and the **console** link for the big screen.



The console's **Join** card, for the projector while players join. Phones can scan the QR code.

Real pods under your camera

With a camera over a table and real pods, your laptop is the link: it watches the table, drives the pods over Bluetooth and sends everything to your room's arena. There are no corner markers. One pod drives a small circle to measure the table, and then you draw the arena on the camera's picture.

What the laptop needs

- **Docker**, and a copy of the Plato Pod folder. The camera program runs in the platform's container, so nothing else is installed; the first start fetches the container, about 5 GB.
- **Linux**, for the camera and Bluetooth inside the container.
- **The pods' settings.** Pods built differently drive differently. Copy `config/agent/pods.example.yaml` to `.platopod/pods.yaml` in your home folder and keep the pods you have: each pod is listed by its tag, with its own motors and wheels. A pod not listed takes the blue pod's settings.

Before the class

1. Make a room (see Run your own room) and open its **console** on the laptop with the camera.
2. Fix the camera so it cannot move, looking down at the table. It can be tilted. If the camera or the table moves later, the arena has to be measured again.
3. Switch the pods on, and put one near the middle of the picture with room around it.

4. Start the camera program on the laptop, in a terminal, from the Plato Pod folder. The room's page has the command for each arena: open **camera laptop** under the arena and press **Copy**. It looks like this:
   - `tools/agent.sh --server https://a.arena.platopod.com --pin <PIN> --camera "Global Shutter" --pods 6 --radio ble`
   - `--server` is your arena's address, the one the console opens on.
   - `--pin` is the room's teacher PIN, shown on the room's page.
   - `--camera` takes the camera's name, or part of it (`"Global Shutter"` for the IR camera), or its number (`0`).
   - `--pods` lists the tag number on each pod, for example `--pods 5 6` .
   - Stop it with Ctrl+C.
5. In the console, the top bar says **No arena yet**, and the pods appear where the camera sees them.

Measure the table

1. Open the **Arena** sheet. It says *The table has not been measured*.
2. Press **Calibrate**. The pod drives a small circle. The sheet shows *Measuring the table...* and how far it has got.
3. When it finishes, the sheet says *Table measured*.

The lap also fits the camera's lens to the pod's wheels, so the measurements get better each time.

Draw the arena

1. In the Arena sheet, press **Draw the arena**. In 3D, the camera's picture is laid on the table seen from above.

2. Choose **Rectangle** and click two opposite corners, or **Any shape** and click each corner in turn, then **Close**.
    - Every corner must stay inside the dashed outline: that is what the camera sees. A rectangle turns red when a corner leaves it.
3. Press **Confirm**. The top bar shows the arena's size, and the pods are kept inside it.

To start again, press **Draw a new arena**. This removes the arena and stops the pods until you draw the new one.

The next time

The laptop keeps the table's measurement and the arena in the `.platopod` folder in your home folder. At the next start, the Arena sheet asks *Use the previous arena?*:

- **Use it** only if the camera and the table have not moved. It is checked as soon as a pod's tag is seen.
- **Discard it** and press **Calibrate** if anything has moved.

Leave out `--calibration` when you start the camera program: a different calibration file given there takes over from the one the last lap kept.

Light

A dark table hides the tags. In the **Layers** sheet, set **Camera brightness** to *Set by hand*:

- **Exposure**: shorter keeps moving pods sharp.
- **Gain**: in a dim room, raise the gain rather than the exposure.

You can also give the light at the start, with `--exposure` and `--gain`.

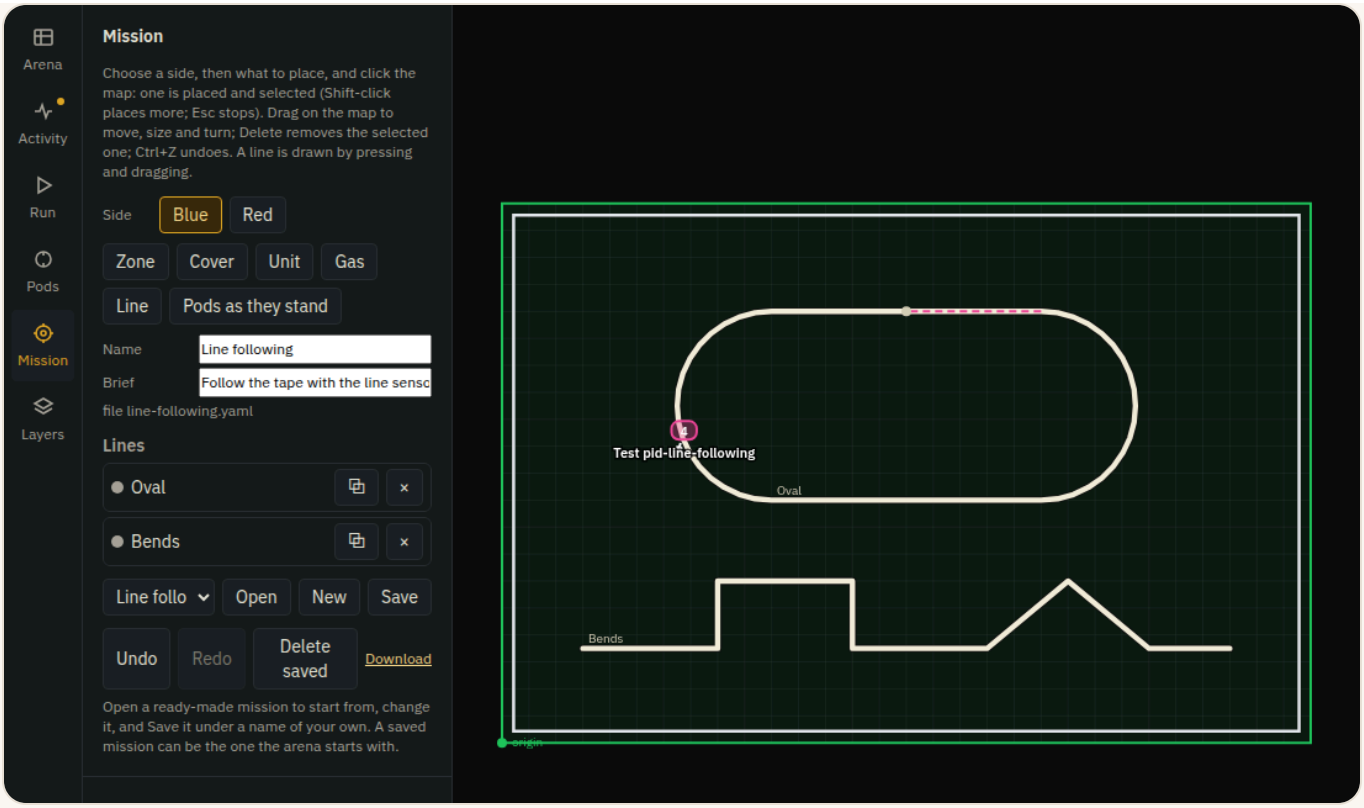
When it goes wrong

Problem	Fix
The top bar says **Camera lost: check its cable**	The laptop has had no picture for 2 seconds, and the pods' places are withdrawn. Plug the camera back in, in any port: the program finds it by name within a few seconds.
The top bar says **Camera laptop off the line**	The laptop changed network, or the Wi-Fi dropped. Wait: the program reconnects by itself within about 15 seconds.
Calibration failed: lost sight of the pod	Keep the pod in the picture and nothing over it, then press **Calibrate** again.
Calibration failed: the pod drives the wrong way from its tag	The lid is on the wrong way round. Turn it, then press **Calibrate** again.
Calibration failed: another pod moved, or the pod is not moving	Check each pod's tag matches the number in `--pods`, that it is switched on, and that `pods.yaml` lists its settings.
A pod drives the wrong way, or curves	Its settings are another pod's: list it in `pods.yaml` under its tag.
Arena stopped: the camera or table may have moved	Look at the picture. If the dashed outline is still on the table, press **It is still right**; otherwise **Measure again**.
A pod is not seen	Make the table brighter (see Light), and keep the tag flat and in view.
The arena sits a little off the table	Raised things lean away from a tilted camera. Draw the corners where the table's edge is in the picture.
No Bluetooth on Linux	Run `rfkill unblock bluetooth` and start the program again.

Set up activities

Everything the demo rooms do can be set up in your own room from the console.

- **An activity:** the **Activity** sheet lists them, each with how long it takes: Plain English and Infiltration (orders in plain English), and Pursuit, Route clearance and Patrol (programming). Choose one, then **Run**.
- **A mission:** the **Mission** sheet places zones to hold or take, cover, where each unit starts, gas, and lines on the floor. **Open** loads a ready-made one; **Save** keeps yours as a mission of your own.
- **Lines for the line sensor:** in the Mission sheet choose **Line**, then press and drag on the map. The line's card sets the tape's width. The ready-made **Line following** mission has the oval and the bends of the PID room.
- **Hidden targets:** the ready-made **Swarm search** mission hides the ten targets of the swarm room.
- **A swarm:** in the **Arena** sheet, set how many pods a student's program may drive (up to 8). Back to one when the room closes.
- **ATAK:** see ATAK.



The Mission sheet with the line-following room's two courses. Choose **Line** and drag on the map to draw another.

ATAK

For classes that train with ATAK: an arena can show its units, zones and engagements on ATAK devices, through a TAK server beside the arenas. Students drive or command the pods in Plato Pod; ATAK shows the same picture on the map, as a staff officer would see it.

What you need

- A room whose arena has the ATAK link. Ask for it with your invite: one arena per TAK server can show on ATAK at a time, so it is set up for your room on request.
- ATAK on an Android device (ATAK-CIV is free on Google Play), on the internet.

In the console

In the **Arena** sheet, tick **Show this arena on ATAK**. It starts off, and it goes off again when the room closes, so ATAK shows nothing between classes.

In ATAK

Add a TAK server connection in ATAK's network settings:

Setting	Value
Address	tak.arena.platopod.com
Port	8087
Protocol	TCP (streaming)
Authentication	none, no certificate

Within a few seconds the arena's units appear: each pod as a unit marker with its call sign, the mission's zones as shapes, and engagements as they happen. The arena sits at a fixed place on the map, scaled up so the table becomes a training area.

Limits

- One arena per TAK server shows on ATAK at a time.
- The connection is plain TCP with no password: anyone who knows the address can watch the units. That is fine for training with virtual units; never use it for real positions or sensitive scenarios.

Students and privacy

- **What students give:** a name, nothing else; a nickname is fine. Students have no accounts.
- **How long a name stays:** in the demo rooms a name and its pod are cleared after 20 minutes unused; in your own room, until you clear it or the room closes.
- **The language model:** orders written in plain English are sent to Claude, Anthropic's model, to be turned into a plan, which is read back before anything moves. Keep personal details out of orders.
- **What is kept:** a closed room's match reports (orders and results) are kept 90 days for its teacher, then deleted.
- **Teachers:** an account holds an email address, a name and an organisation. Rooms are reached by their codes and links; nobody else sees a teacher's rooms.
- **Devices:** any laptop or tablet with a browser; phones work but are cramped. Notebooks need Python 3.10 or later, or Colab. The hosted rooms are all virtual; the same code drives real pods on a classroom box with a camera.

Troubleshooting

Problem	Fix
Join page says the room isn't found	Check the code on the room card; the room may have closed after 7 idle days. Make a new room.
A notebook says the room is full	The demo rooms take six people at once (the swarm room three). Try again in a few minutes: names nobody uses leave after 20 minutes. Or ask for a room of your own.
A notebook cannot reach the server, or cannot install websockets	A school network may block it: run the notebook in Google Colab instead.
A sensor click didn't land	Click the spot again. On a raised wall a sensor sits on the wall's top.
Start asks: a team has no orders yet	Click Cancel and have that team click **Read my orders**, then Start again.
The read-back is not what was meant	Rewrite the orders more plainly, naming the pod and the place, and send again.
ATAK shows nothing	Check **Show this arena on ATAK** is ticked in the Arena sheet, and the server is tak.arena.platopod.com, port 8087, TCP. Only an arena with the ATAK link has the switch.
A screen looks stuck	Reload the page; it reconnects and keeps the team.
Everything is wrong	**Reset all**, then start again from Activity.

If none of this helps, write to a.lenskiyunsw.edu.au.