ANSWERS

U1.2 A command is a request

The robot as a system · University · about 25 min

What this lesson is about

Open loop on real hardware: gain that is not 1, a dead band, a lag, and a robot that curves.

- What actually happens
- Measuring instead of assuming
- Why this matters

Questions

6 marks in all

1. What does forward(60) actually ask for? [1 mark]

- ☐ **A** The motors driven at 60 percent
- ☐ **B** A speed of 60 cm/s
- ☐ **C** Exactly 60 percent of the catalogue top speed of 20 cm/s, which is 12 cm/s
- ☐ **D** A move of 60 cm

> **Answer:** A. A command is a request to the motors. The speed that results is a fact about this robot's gain, dead band, lag and coupling, and has to be measured.

2. Asked to drive 60 cm forward, the robot finishes to one side of where it started and a few degrees off heading. Which of the four effects in the lesson explains that? [1 mark]

- ☐ **A** Coupling
- ☐ **B** Dead band
- ☐ **C** Lag
- ☐ **D** Gain

> **Answer:** A. Coupling means driving one axis leaks into the others: a little sideways and a little rotation, so the robot curves. Gain, dead band and lag change how fast, not which way.

3. A robot's measured forward speed is 7 cm/s at command 40 and 17 cm/s at command 80. Using the straight line through those two points, what command gives 12 cm/s? [1 mark]

> **Answer:** 60. The slope is (17 - 7) / (80 - 40) = 0.25 cm/s per percent. 12 cm/s is 5 cm/s above 7, which is 5 / 0.25 = 20 percent above 40, so 60.

4. For the same line (7 cm/s at command 40, 17 cm/s at command 80), at what command does the line predict zero speed? [1 mark]

> **Answer:** 12. Speed = 0.25 × command - 3, which is zero at 3 / 0.25 = 12. That is only where the straight line crosses zero. Below the dead band, about 15 on this drive, the real robot gives nothing at all, so the line is only good for commands above it.

5. The measurement program waits 1.0 s after each forward() before it starts averaging flow(). Why? [1 mark]

- ☐ **A** The speed rises over about a quarter of a second, so readings taken during the rise would underestimate the settled speed
- ☐ **B** flow() only produces a new reading once a second
- ☐ **C** The deadman needs a second to reset before readings are valid
- ☐ **D** It lets the battery voltage recover after the previous step

> **Answer:** A. That is the lag. Speed does not step, it rises, so you wait past the transient and then measure the level.

6. Which description of open loop control matches the lesson? [1 mark]

- ☐ **A** Acting on the command and hoping, which only works on a stiff, repeatable, unloaded machine
- ☐ **B** Measuring the result and correcting as you go
- ☐ **C** Keeping an estimate of the state rather than a commanded position
- ☐ **D** Running the loop without any wait() so it goes as fast as possible

> **Answer:** A. Open loop trusts the command to be the result. Measuring, feedback and state estimation all exist because on a real machine it is not.

The task: how fast is this robot?

Measure this robot's top forward speed in centimetres per second and print it as `top speed: 18.4`. Measure it, do not look it up.

```
from bugbot import *
connect()

forward(100)
wait(1.0)
```

The hint students can ask for: It takes about a quarter of a second to reach full speed, so ignore the beginning. Either read flow() several times and average, or time a known distance with position().

A solution

```
from bugbot import *
connect()

forward(100)
wait(1.0)
readings = []
for i in range(20):
    readings.append(flow()[1])
    wait(0.1)
stop()
print("top speed:", round(sum(readings) / len(readings), 1))
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.