ANSWERS

U1.4 Time, rate and latency

The robot as a system · University · about 25 min

What this lesson is about

How often the loop runs, how late the answer arrives, and what both do to a controller.

- Three different times
- The same controller at two rates
- Why lateness turns into instability
- Measuring your own loop

Questions

6 marks in all

1. This drive takes about a quarter of a second to get most of the way to a new speed. Which of the three times in the lesson is that? [1 mark]

- ☐ **A** The time constant
- ☐ **B** The period
- ☐ **C** The latency
- ☐ **D** The deadman timeout

> **Answer:** A. The time constant is how long the machine itself takes to react. The period is how often your loop runs, and latency is how old a measurement is when you act on it.

2. The drive's time constant is about 0.25 s. Which loop rate does the lesson's rule of thumb favour? [1 mark]

- ☐ **A** 10 Hz
- ☐ **B** 4 Hz, one pass per time constant
- ☐ **C** 2 Hz
- ☐ **D** 1 Hz

> **Answer:** A. Sample well faster than the thing you are controlling. At 10 Hz the loop runs several times within one time constant; at 2 Hz the error grows for half a second before anything is done about it.

3. Why can too much delay make a feedback loop unstable? [1 mark]

- ☐ **A** The loop pushes against an error that has already moved on, and with enough delay it pushes the wrong way, so negative feedback becomes positive
- ☐ **B** Late measurements carry more noise than fresh ones
- ☐ **C** A slow loop always saturates the motors
- ☐ **D** Delay adds a constant bias to the measured heading

> **Answer:** A. Feedback only works if it pushes against the current error. Delay turns negative feedback into positive feedback, which is the whole story of instability in one sentence.

4. A loop oscillates because its information is late. Which of these are fixes from the lesson? [1 mark]

Tick every answer that is true.

- ☐ **A** Run the loop faster
- ☐ **B** Turn the gain down
- ☐ **C** Predict where the error is going rather than where it was
- ☐ **D** Turn the gain up, so the correction is finished before the information goes stale

> **Answer:** A, B, C. Fresher information, gentler pushes, or pushing against a prediction. More gain makes a late push do more harm, not less.

5. A loop asks for wait(0.05), and each pass also spends 0.012 s in distance() and 0.018 s in scan(). What does this print? [1 mark]

```
work = 0.012 + 0.018
requested = 0.05
period = requested + work
print("period:", round(period, 3), "s")
print("rate:", round(1 / period, 1), "Hz")
```

> **Answer:**
>
> ```
> period: 0.08 s
> rate: 12.5 Hz
> ```
>
> The pass costs 0.05 + 0.03 = 0.08 s, so the loop runs at 1 / 0.08 = 12.5 Hz, not the 20 Hz the wait suggests.

6. The heading-hold loop works out its error with wrapped(). With clockwise-positive heading, what does this print? [1 mark]

```
def wrapped(a):
    return (a + 180) % 360 - 180

print(wrapped(0 - 350), wrapped(0 - 10), wrapped(190))
```

> **Answer:**
>
> ```
> 10 -10 -170
> ```
>
> A robot at heading 350 is 10 degrees anticlockwise of 0, so the error is +10 (turn clockwise), not -350. At heading 10 it is -10, and 190 wraps to -170.

The task: hold the heading

Drive into the green zone, and from three seconds onwards stay within 8 degrees of heading 0. This robot pulls hard to one side, so the loop has to be doing real work all the way.

```
from bugbot import *
connect()

def wrapped(a):
    return (a + 180) % 360 - 180

# drive, correcting as you go, and keep correcting once you are there
forward(70, distance=130)
stop()
```

The hint students can ask for: This robot pulls to one side. Correct with the rotation term of drive(), in proportion to the heading error. A slower loop needs a gentler gain, or it will swing about.

A solution

```
from bugbot import *
connect()

def wrapped(a):
    return (a + 180) % 360 - 180

while position()[1] < 130:
    error = wrapped(0 - heading())
    drive(70, 0, error * 3)
    wait(0.1)
stop()
for i in range(12):
    error = wrapped(0 - heading())
    drive(0, 0, error * 3)
    wait(0.1)
stop()
```

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