WORKSHEET

U1.4 Time, rate and latency

The robot as a system · University · about 25 min

Name _____ Class _____ Date _____

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

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

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

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

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")
```

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))
```

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()
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/u1-4-time-and-latency/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Raise the gain until the robot oscillates, then halve the loop rate. Does it take more gain or less to oscillate now?
2. Add an artificial delay: keep the last three errors in a list and control on the oldest one. What does it look like?
3. Time the loop above with `wait(0.03)` and with `wait(0.07)`. The simulator gives exactly 0.03 and 0.07. Add a line inside the loop that works something out for a long time, such as `sum(range(1000000))`. Does the period change in the simulator? Would it on the robot?