ANSWERS

U1.1 Sense, decide, act

The robot as a system · University · about 20 min

What this lesson is about

The control loop, its period, and the deadman that stops a robot whose program has gone quiet.

- A loop you can watch
- The period
- The deadman
- Blocking and non-blocking

Questions

7 marks in all

1. Put the body of the control loop in order, as it runs on every pass. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ `act(command)`
___ `wait(period)`
___ `command = decide(measurement)`
___ `measurement = sense()`

Answer:

```
measurement = sense()
command = decide(measurement)
act(command)
wait(period)
```

Read the sensors, work out what to do, tell the motors, then wait out the rest of the period. The rest of the course is about what goes inside decide.

2. A loop ends each pass with wait(0.1) and does nothing else that takes noticeable time. How often does it run, and roughly what is the fastest disturbance its measurements can represent properly? [1 mark]

- ☐ **A** 10 Hz, and disturbances slower than about 5 Hz
- ☐ **B** 10 Hz, and disturbances up to about 10 Hz
- ☐ **C** 0.1 Hz, and disturbances slower than about 0.05 Hz
- ☐ **D** 10 Hz, and disturbances up to about 20 Hz

Answer: A. A period of 0.1 s is 1 / 0.1 = 10 Hz. By the sampling theorem anything faster than about half the loop rate cannot be seen properly, and can look like a slower disturbance.

3. A loop has a period of 0.04 s. What is its rate in Hz? [1 mark]

Answer: 25. Rate is one over period: 1 / 0.04 = 25 Hz.

4. A loop calls wait(0.1), and the sensor reads and arithmetic in each pass take a further 0.03 s. This works out the rate it really runs at. What does it print? [1 mark]

```
work = 0.03
period = 0.1 + work
print("rate:", round(1 / period, 1), "Hz")
```

Answer:

```
rate: 7.7 Hz
```

The real period is 0.1 + 0.03 = 0.13 s, so the rate is 1 / 0.13 = 7.7 Hz. wait() is what is left over after the work, and the work is not free.

5. A program calls forward(60) once and then wait(2), with no other commands. What does the robot do? [1 mark]

- ☐ **A** Drives for about half a second, then stops itself because no new command arrived
- ☐ **B** Drives at 60 for the full 2 seconds, then stops
- ☐ **C** Does not move at all, because wait() blocks the command from being sent
- ☐ **D** Drives 60 cm and then stops

Answer: B. wait() counts as hearing from the program, so the deadman stays fed and the robot drives the full 2 seconds, about 19 cm at 60. The deadman is for a program that stops talking: one that crashes, gets stuck, or ends.

6. Why does every controller in the course use forward(60) followed by wait(0.1), rather than forward(60, distance=40)? [1 mark]

- ☐ **A** The non-blocking form returns at once, so the loop can change the command on the next pass
- ☐ **B** The blocking form is less accurate about the distance it drives
- ☐ **C** The deadman does not apply to blocking calls, which makes them unsafe
- ☐ **D** The non-blocking form drives the motors harder

Answer: A. A blocking call only comes back when the move is finished. A controller has to be able to change its mind half way, so it sets the motors, returns, and decides again next tick.

7. Which of these are set directly by the loop's period? [1 mark]

Tick every answer that is true.

- ☐ **A** How often the controller can react to what the world does
- ☐ **B** The fastest motion the measurements can represent properly
- ☐ **C** The robot's top speed
- ☐ **D** The size of the drive's dead band

Answer: A, B. The period decides how often you act and how often you measure. Top speed and dead band are properties of the hardware, whatever the loop does.

The task: run a loop at 10 Hz

Without driving, read the depth sensor thirty times at ten times a second, print each reading, and print how long the loop actually took on average, as `period: 0.1`.

```
from bugbot import *
connect()

# clock() is the seconds since the program started
start = clock()
```

The hint students can ask for: clock() gives the seconds since the program started. Read it before the loop and after it, and divide the difference by how many times round you went.

A solution

```
from bugbot import *
connect()

start = clock()
n = 30
for i in range(n):
    print(i, distance())
    wait(0.1)
print("period:", round((clock() - start) / n, 3))
```

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