ANSWERS

3.4 Controlling speed

Control · Robot club · about 15 min

What this lesson is about

Speed from the remaining distance, the dead band, clamping.

- The bang-bang park
- Proportional speed
- The dead band
- The pattern

Questions

6 marks in all

1. The bang-bang park drives at 60 until `distance()` is 22 or less, then stops. Where does it end? [1 mark]

- ☐ **A** Closer than 22 cm, because it coasts
- ☐ **B** Exactly 22 cm
- ☐ **C** Further than 22 cm
- ☐ **D** Touching the wall every time

> **Answer:** A. Fast all the way and stop means a long slide. It is the overshoot from lesson 3.1, now driving forward.

2. Target 22, gain 4, and `distance()` reads 32. What speed does `speed = (distance() - target) * 4` ask for? [1 mark]

> **Answer:** 40. The error is 32 minus 22, which is 10, and 10 times 4 is 40.

3. The plain proportional park stalls a little short of the target. Why? [1 mark]

- ☐ **A** Near the target it asks for speeds below about 15, which do not move the robot
- ☐ **B** The gain is too high
- ☐ **C** The sensor cannot see closer than 25 cm
- ☐ **D** The loop runs out of ticks too early

> **Answer:** A. The vibration motors have a dead band. With 3 cm to go it asks for 12, which does nothing.

4. What does this program print? [1 mark]

```
for value in [4, 30, 250]:
    print(max(16, min(60, value)))
```

> **Answer:**
>
> ```
> 16
> 30
> 60
> ```
>
> Clamping keeps a number between two limits. 4 is raised to 16, 30 is fine, and 250 is capped at 60.

5. In the clamped park, the robot overshoots and `distance()` reads 18. What does it do? [1 mark]

```
error = distance() - target
speed = max(16, min(60, abs(error) * 4))
if error > 0:
    forward(speed)
else:
    backward(speed)
```

- ☐ **A** Backs up slowly, at speed 16
- ☐ **B** Stops, because it is past the target
- ☐ **C** Drives forward at speed 16
- ☐ **D** Backs up at speed 60

> **Answer:** A. The error is 18 minus 22, which is -4, so it goes backward. 4 times 4 is 16, which is exactly the lowest allowed speed.

6. Put the three lines of every P controller in order. [1 mark]

Number the lines 1 to 3 to put them in the right order.

___ `output = clamp(error * gain)`

___ `apply(output)`

___ `error = target - measured`

> **Answer:**
>
> ```
> error = target - measured
> output = clamp(error * gain)
> apply(output)
> ```
>
> Measure the error, work out how hard to push, then push. Steering and speed are the same loop with a different sensor and motor.

The task: park at twenty

Stop with the front of the robot 20 to 24 cm from the wall, the green band, without touching the wall.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

# as long as the thing ahead is further than 22 cm
while distance() > 22:
    # drive forward at 60 (keeps going until the next command)
    forward(60)
    # pause 0.1 s (the robot keeps doing what it was told)
    wait(0.1)
# all motors off
stop()
```

The hint students can ask for: Stop with the front of the robot 20 to 24 cm from the wall: the green band. Make the speed depend on how far you still have to go, and remember the motors do nothing below about 15.

A solution

```
from bugbot import *
connect()
target = 22
while True:
    error = distance() - target
    if abs(error) < 1:
        break
    speed = max(16, min(60, abs(error) * 4))
    if error > 0:
        forward(speed)
    else:
        backward(speed)
    wait(0.05)
stop()
```

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