ANSWERS

5.3 Timers

Behaviours · Robot club · about 15 min

What this lesson is about

Two things at once, remembered ticks, timeouts.

- Two things at once
- A timer is a remembered tick
- Timeouts

Questions

7 marks in all

1. Why can a program not blink the LED with `wait(0.5)` while it drives to the zone? [1 mark]

- ☐ **A** `wait(0.5)` holds up the whole loop, so the driving stops being updated
- ☐ **B** `wait` turns the LED off
- ☐ **C** The LED cannot change while the robot moves
- ☐ **D** `wait` only accepts whole numbers

> **Answer:** A. In a behaviour nothing waits except the tick. Count ticks instead, and act on every fifth one.

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

```
on = False
changes = 0
for ticks in range(20):
    if ticks % 5 == 0:
        on = not on
        changes += 1
print(changes, on)
```

> **Answer:**
>
> ```
> 4 False
> ```
>
> Ticks 0, 5, 10 and 15 each flip the toggle. Four flips from False brings it back to False.

3. The loop waits 0.1 s a tick. To toggle the LED every half second, every how many ticks should it toggle? [1 mark]

> **Answer:** 5. Half a second is 5 ticks at ten ticks a second, so `ticks % 5 == 0`.

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

```
started = None
for ticks in range(40):
    if started is None and ticks == 20:
        started = ticks
        print("start", ticks)
    elif started is not None and ticks - started >= 10:
        print("stop", ticks)
        started = None
```

Answer:

```
start 20
stop 30
```

The timer remembers tick 20. `ticks - started` is the time since then, and it reaches 10 at tick 30, one second later.

5. In `started_turning = None`, what does `None` mean? [1 mark]

- ☐ **A** No timer is running
- ☐ **B** The timer has run for 0 ticks
- ☐ **C** The robot has stopped
- ☐ **D** The turn failed

Answer: A. `None` says there is no event to count from. When the turn starts, it holds the tick it started on.

6. A search spins while `ticks < 30` and gives up if nothing is found. Why is this timeout important? [1 mark]

- ☐ **A** Without it, a search for a tag that is not there would spin forever
- ☐ **B** The camera stops working after 3 seconds
- ☐ **C** It makes the tag easier to see
- ☐ **D** It saves the list of tags

Answer: A. The most important timer is the one that gives up, so the program can print a message and do something else.

7. What is the advantage of `clock() - started_at >= 1.0` over counting ticks? [1 mark]

- ☐ **A** It stays right even if a tick takes longer than planned
- ☐ **B** It works without a loop
- ☐ **C** It makes each tick shorter
- ☐ **D** It stops the robot automatically

Answer: A. Counting ticks assumes each is exactly 0.1 s. The real clock measures the time that actually passed.

The task: blink and drive

Drive into the green zone while the LED blinks green and off every half second, at least eight changes.

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

# the LED: green
led("green")
# drive forward at 60 for 70 cm, then stop
forward(60, distance=70)
# the LED: off
led("off")
```

The hint students can ask for: Drive to the green zone while the LED blinks green and off every half second. Two things at once means a tick loop and a tick counter, not two waits.

A solution

```
from bugbot import *
connect()
ticks = 0
on = False
while position()[1] < 70:
    forward(60)
    if ticks % 5 == 0:              # every 5 ticks = every half second
        on = not on
        led("green" if on else "off")
    ticks += 1
    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.