ANSWERS

# U1.7 Project: write the data sheet

BugBotLab

The robot as a system · University · about 45 min

## What this lesson is about

Characterise the robot you were given and print the figures the rest of the course will use.

- What the data sheet holds
- How to write the program
- What comes next

## Questions

**6 marks in all**

**1.** Put the steps of one data sheet experiment in order. [1 mark]

Number the lines 1 to 5 to put them in the right order.

___ `Wait past the transient, about 3 tau`
___ `Stop, and let the robot settle`
___ `Take several readings and average them`
___ `Stop, and let it settle again before the next experiment`
___ `Apply the input`

**Answer:**

```
Stop, and let the robot settle
Apply the input
Wait past the transient, about 3 tau
Take several readings and average them
Stop, and let it settle again before the next experiment
```

The same discipline every time. Skipping the last step is the usual mistake.

**2.** Why must the robot settle again after each experiment before the next one starts? [1 mark]

☐ **A** Otherwise the next measurement is taken while the robot is still coasting, so it measures the previous experiment

☐ **B** The deadman needs half a second to reset

☐ **C** The flow sensor has to re-zero between experiments

☐ **D** The battery needs time to recover

**Answer:** A. A reading taken while the robot still carries motion from the last input is contaminated by that input.

3. Why does each student measure their own data sheet rather than use a shared one? [1 mark]

- ☐ **A** Each BugBot is built with its own gain on each axis, gyro bias and flow scale, fixed for the run
- ☐ **B** The values change continuously during a run, so a shared sheet would be out of date within seconds
- ☐ **C** The shared data sheet is in different units
- ☐ **D** The simulator only reports speeds that have been measured in the same run

> **Answer:** A. The figures belong to your robot and stay fixed for the run. That is why measuring beats looking up.

4. These are the settled readings from two experiments: forward speed from flow() and turn rate from imu(). What does this print? [1 mark]

```
readings = [19.4, 18.9, 19.6, 19.1, 19.0]
rates = [-112.0, -115.5, -113.0, -114.5, -112.5]
v_max = sum(readings) / len(readings)
w_max = sum(rates) / len(rates)
print("v_max:", round(v_max, 1))
print("w_max:", round(abs(w_max), 1))
```

> **Answer:**
>
> ```
> v_max: 19.2
> w_max: 113.5
> ```
>
> The speeds sum to 96.0, so the mean is 19.2. The rates sum to -567.5, a mean of -113.5, and abs() reports it as a size: 113.5.

5. A command is walked up in steps of 5, recording the settled flow() speed at each. Anything above 1 cm/s counts as moving. What does this print? [1 mark]

```
table = [(5, 0.0), (10, 0.1), (15, 2.4), (20, 3.6), (25, 4.9)]
for power, v in table:
    if v > 1.0:
        print("dead band:", power)
        break
```

> **Answer:**
>
> ```
> dead band: 15
> ```
>
> 0.1 cm/s at command 10 is noise, not motion. The first command that clearly moves the robot is 15.

6. The settled() helper waits 1.0 s after applying an input before reading. With a time constant of about 0.25 s, is that long enough? [1 mark]

- ☐ **A** Yes: 1.0 s is 4 tau, past the 3 tau at which the response is 95 percent settled
- ☐ **B** No: a response needs about 10 tau to settle
- ☐ **C** No: 1.0 s is shorter than one time constant
- ☐ **D** Yes, but only because the deadman stops the robot after half a second anyway

> **Answer:** A. 1.0 / 0.25 = 4 time constants, which leaves the response about 98 percent settled. A longer hold wastes the ninety second budget.

## The task: the data sheet

Measure all four and print them, one per line, exactly like this: The whole run has ninety seconds, which is plenty for four experiments and nowhere near enough to be careless.

```
from bugbot import *
connect()

# measure, do not assume
```

**The hint students can ask for:** Four measurements, one after another, each one a short drive and a read. Work up from a small command in steps to find where the robot first moves. Keep the whole run under ninety seconds.

### A solution

```
from bugbot import *
connect()

# top forward speed
forward(100)
wait(1.0)
fwd = [flow()[1] for i in range(10) if wait(0.1) is None]
stop()
wait(0.5)
v_max = sum(fwd) / len(fwd)

# top turn rate
drive(0, 0, 100)
wait(1.0)
turns = [imu()[1] for i in range(10) if wait(0.1) is None]
stop()
wait(0.5)
w_max = sum(turns) / len(turns)

# the dead band: work up until it actually moves
dead = 100
for power in range(5, 60, 5):
    forward(power)
    wait(0.8)
    if abs(flow()[1]) > 2.0:
        dead = power
        stop()
        break
    stop()
    wait(0.3)

ox, oy, oh = odometry()
tx, ty = position()
drift = ((ox - tx) ** 2 + (oy - ty) ** 2) ** 0.5

print("v_max:", round(v_max, 1))
print("w_max:", round(abs(w_max), 1))
print("dead band:", dead)
print("drift:", round(drift, 1))
```

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