WORKSHEET

U12.7 Capstone: the whole robot

BugBotLab

Learning, and the capstone · University · about 90 min

Name _____ Class _____ Date _____

What this lesson is about

Calibrate, estimate, localise, plan, follow, arrive, and report. One run, everything in it.

- What it takes
- The structure
- The fix
- What a good submission looks like
- How it would be assessed
- Where to go next

Questions

6 marks in all

1. Tag 21 is at (100, 195) and tag 22 at (40, 195). The robot measures r1 = 46.1 cm to tag 21 and r2 = 67.3 cm to tag 22. What does this print? [1 mark]

```
import math
x1, x2, wall = 100.0, 40.0, 195.0
r1, r2 = 46.1, 67.3
x = (x1 ** 2 - x2 ** 2 - r1 ** 2 + r2 ** 2) / (2 * (x1 - x2))
y = wall - math.sqrt(r1 ** 2 - (x - x1) ** 2)
print(round(x, 1), round(y, 1))
```

2. In the two tag fix, why take the negative square root for y? [1 mark]

- ☐ **A** The robot is below the wall the tags are on, so it is at 195 minus the distance, not plus
- ☐ **B** Square roots in Python are negative by default
- ☐ **C** Heading 0 points down the mat
- ☐ **D** The tags face away from the robot

3. After calibration a gyro bias of 0.2 degrees per second is left uncorrected. How many degrees of heading error does that build up over 30 s of driving? [1 mark]

4. A tag fix puts the robot 40 cm from where its estimate says it is. What should the program do? [1 mark]

- ☐ **A** Treat it as a probable misread and gate it out, and print the disagreement
- ☐ **B** Jump the estimate to the fix, since the tags are ground truth
- ☐ **C** Average the fix and the estimate equally
- ☐ **D** Stop the run, because the estimator has failed

5. Only one of the two tags is in view for a frame. How should that frame be used for the fix? [1 mark]

- ☐ **A** As no fix at all
- ☐ **B** As half a fix, weighted by 0.5
- ☐ **C** By assuming the missing range equals the one seen
- ☐ **D** By using the last range of the missing tag

6. Which belong in a submission that would earn a good mark? [1 mark]

Tick every answer that is true.

- ☐ **A** An estimator that runs every tick, not dead reckoning with a jump at the end
- ☐ **B** A stated frame, with every reported number in it
- ☐ **C** The run repeated a few times with the spread reported
- ☐ **D** The final estimate with how far out you believe it is
- ☐ **E** The single best run, with the others left out

The task: the capstone

Reach the green target through the gap, without touching anything, and print `gyro bias:`, `my x:` and `my y:` from your own estimate. Plot `x` and `y` as you go. No `position()`.

```
from bugbot import *
import math
connect()

DT = 0.1
V_MAX, V_LAT = 20.0, 15.0
START = (30.0, 25.0)
TAGS = {21: (100.0, 195.0), 22: (40.0, 195.0)}
```

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

Do it on the robot

www.bugbotlab.com/learn/u12-7-capstone/

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

Challenges

1. Add a gate on the fix: reject one that disagrees with the estimate by more than three times your uncertainty, and count the rejections.

2. Run it ten times and report the mean and worst final error. Which stage contributes most of it?
3. Remove the tag fix and see how far dead reckoning alone gets. Then work out what heading accuracy would have been needed to make up the difference.