

Assignment 1: Basic Controllers

You will implement a series of simple, basic controllers using an e-puck robot. The main goals of this assignment are:

- To get comfortable with connecting to and programming software to remotely control an e-puck robot.
- Understand what an open-loop controller is and what the key limitations are.
- To understand the forms and limitations of simple kinematics for differential drive robots.

You will develop a very high level of comfort with programming differential drive robots to perform a range of desired high level commands. You will create toolkits that enable you to think on the level of what you want the robot to do, not how to manage actuators directly.

There is a lot of overhead on this assignment before you can really get anything working. You will need to take the time to get a connection to the robot, a development environment working, and live code running and communicating with the robot. Check out the online **e-puck robot primer** document and be sure you can successfully run the included sample program before you start. I recommend connecting to the robot via Bluetooth for this assignment as it is simpler and more robust than the WiFi connection.

epuck library

I have (written and...) provided a simplistic library for connecting to and communicating with the epuck robot. See the sample code in the primer package and the epucksample.py program. This library has minimal error handling and has had minimal debugging so you may have to delve into the robot and communications protocol if something breaks.

However, it should provide sufficient functions to connect to the robot, send commands, and read its sensors. Take time to read it over and be familiar with the functions and data provided.

Hint: helper functions

I provide line counts throughout the assignment. This is not a target, but a hint: if your code is significantly longer, you may be doing something wrong.

Put each task into its own file. I use AS1.2.py, AS1.3.py, etc.

I found it useful to start by implementing and testing the following helper functions. You will need to take measurements of your robot to do these. There is sufficient variation that I would not rely on spec sheets but rather use high quality calipers to measure if you can. I put these into a file called As1lib so I could use them across my tasks, which I put into separate files.

- `steps_delta(last, current): int`, calculates the difference in robot steps from the last position to the current, accounting for counter wraparound, and returns it as a signed integer. This one is tricky. Mine is 5 lines including def and return.
- `steps_to_rad(steps): float`, converts signed motor steps to signed radians, and returns that value, using your knowledge of the motor construction. 2 lines
- `rad_to_steps(rad): float`, converts signed radians to signed motor steps, and returns that value, using your knowledge of the motor construction. 2 lines
- `rad_to_mm(rad): float`, converts the given signed radians of wheel rotation into expected signed ground distance, and returns that value. 2 lines

300-400 lines
400-500 lines
500-600 lines
600-700 lines
700-800 lines
800-900 lines
900-1000 lines
1000-1100 lines
1100-1200 lines
1200-1300 lines
1300-1400 lines
1400-1500 lines
1500-1600 lines
1600-1700 lines
1700-1800 lines
1800-1900 lines
1900-2000 lines
2000-2100 lines
2100-2200 lines
2200-2300 lines
2300-2400 lines
2400-2500 lines
2500-2600 lines
2600-2700 lines
2700-2800 lines
2800-2900 lines
2900-3000 lines
3000-3100 lines
3100-3200 lines
3200-3300 lines
3300-3400 lines
3400-3500 lines
3500-3600 lines
3600-3700 lines
3700-3800 lines
3800-3900 lines
3900-4000 lines
4000-4100 lines
4100-4200 lines
4200-4300 lines
4300-4400 lines
4400-4500 lines
4500-4600 lines
4600-4700 lines
4700-4800 lines
4800-4900 lines
4900-5000 lines
5000-5100 lines
5100-5200 lines
5200-5300 lines
5300-5400 lines
5400-5500 lines
5500-5600 lines
5600-5700 lines
5700-5800 lines
5800-5900 lines
5900-6000 lines
6000-6100 lines
6100-6200 lines
6200-6300 lines
6300-6400 lines
6400-6500 lines
6500-6600 lines
6600-6700 lines
6700-6800 lines
6800-6900 lines
6900-7000 lines
7000-7100 lines
7100-7200 lines
7200-7300 lines
7300-7400 lines
7400-7500 lines
7500-7600 lines
7600-7700 lines
7700-7800 lines
7800-7900 lines
7900-8000 lines
8000-8100 lines
8100-8200 lines
8200-8300 lines
8300-8400 lines
8400-8500 lines
8500-8600 lines
8600-8700 lines
8700-8800 lines
8800-8900 lines
8900-9000 lines
9000-9100 lines
9100-9200 lines
9200-9300 lines
9300-9400 lines
9400-9500 lines
9500-9600 lines
9600-9700 lines
9700-9800 lines
9800-9900 lines
9900-10000 lines
10000-10100 lines
10100-10200 lines
10200-10300 lines
10300-10400 lines
10400-10500 lines
10500-10600 lines
10600-10700 lines
10700-10800 lines
10800-10900 lines
10900-11000 lines
11000-11100 lines
11100-11200 lines
11200-11300 lines
11300-11400 lines
11400-11500 lines
11500-11600 lines
11600-11700 lines
11700-11800 lines
11800-11900 lines
11900-12000 lines
12000-12100 lines
12100-12200 lines
12200-12300 lines
12300-12400 lines
12400-12500 lines
12500-12600 lines
12600-12700 lines
12700-12800 lines
12800-12900 lines
12900-13000 lines
13000-13100 lines
13100-13200 lines
13200-13300 lines
13300-13400 lines
13400-13500 lines
13500-13600 lines
13600-13700 lines
13700-13800 lines
13800-13900 lines
13900-14000 lines
14000-14100 lines
14100-14200 lines
14200-14300 lines
14300-14400 lines
14400-14500 lines
14500-14600 lines
14600-14700 lines
14700-14800 lines
14800-14900 lines
14900-15000 lines
15000-15100 lines
15100-15200 lines
15200-15300 lines
15300-15400 lines
15400-15500 lines
15500-15600 lines
15600-15700 lines
15700-15800 lines
15800-15900 lines
15900-16000 lines
16000-16100 lines
16100-16200 lines
16200-16300 lines
16300-16400 lines
16400-16500 lines
16500-16600 lines
16600-16700 lines
16700-16800 lines
16800-16900 lines
16900-17000 lines
17000-17100 lines
17100-17200 lines
17200-17300 lines
17300-17400 lines
17400-17500 lines
17500-17600 lines
17600-17700 lines
17700-17800 lines
17800-17900 lines
17900-18000 lines
18000-18100 lines
18100-18200 lines
18200-18300 lines
18300-18400 lines
18400-18500 lines
18500-18600 lines
18600-18700 lines
18700-18800 lines
18800-18900 lines
18900-19000 lines
19000-19100 lines
19100-19200 lines
19200-19300 lines
19300-19400 lines
19400-19500 lines
19500-19600 lines
19600-19700 lines
19700-19800 lines
19800-19900 lines
19900-20000 lines
20000-20100 lines
20100-20200 lines
20200-20300 lines
20300-20400 lines
20400-20500 lines
20500-20600 lines
20600-20700 lines
20700-20800 lines
20800-20900 lines
20900-21000 lines
21000-21100 lines
21100-21200 lines
21200-21300 lines
21300-21400 lines
21400-21500 lines
21500-21600 lines
21600-21700 lines
21700-21800 lines
21800-21900 lines
21900-22000 lines
22000-22100 lines
22100-22200 lines
22200-22300 lines
22300-22400 lines
22400-22500 lines
22500-22600 lines
22600-22700 lines
22700-22800 lines
22800-22900 lines
22900-23000 lines
23000-23100 lines
23100-23200 lines
23200-23300 lines
23300-23400 lines
23400-23500 lines
23500-23600 lines
23600-23700 lines
23700-23800 lines
23800-23900 lines
23900-24000 lines
24000-24100 lines
24100-24200 lines
24200-24300 lines
24300-24400 lines
24400-24500 lines
24500-24600 lines
24600-24700 lines
24700-24800 lines
24800-24900 lines
24900-25000 lines
25000-25100 lines
25100-25200 lines
25200-25300 lines
25300-25400 lines
25400-25500 lines
25500-25600 lines
25600-25700 lines
25700-25800 lines
25800-25900 lines
25900-26000 lines
26000-26100 lines
26100-26200 lines
26200-26300 lines
26300-26400 lines
26400-26500 lines
26500-26600 lines
26600-26700 lines
26700-26800 lines
26800-26900 lines
26900-27000 lines
27000-27100 lines
27100-27200 lines
27200-27300 lines
27300-27400 lines
27400-27500 lines
27500-27600 lines
27600-27700 lines
27700-27800 lines
27800-27900 lines
27900-28000 lines
28000-28100 lines
28100-28200 lines
28200-28300 lines
28300-28400 lines
28400-28500 lines
28500-28600 lines
28600-28700 lines
28700-28800 lines
28800-28900 lines
28900-29000 lines
29000-29100 lines
29100-29200 lines
29200-29300 lines
29300-29400 lines
29400-29500 lines
29500-29600 lines
29600-29700 lines
29700-29800 lines
29800-29900 lines
29900-30000 lines
30000-30100 lines
30100-30200 lines
30200-30300 lines
30300-30400 lines
30400-30500 lines
30500-30600 lines
30600-30700 lines
30700-30800 lines
30800-30900 lines
30900-31000 lines
31000-31100 lines
31100-31200 lines
31200-31300 lines
31300-31400 lines
31400-31500 lines
31500-31600 lines
31600-31700 lines
31700-31800 lines
31800-31900 lines
31900-32000 lines
32000-32100 lines
32100-32200 lines
32200-32300 lines
32300-32400 lines
32400-32500 lines
32500-32600 lines
32600-32700 lines
32700-32800 lines
32800-32900 lines
32900-33000 lines
33000-33100 lines
33100-33200 lines
33200-33300 lines
33300-33400 lines
33400-33500 lines
33500-33600 lines
33600-33700 lines
33700-33800 lines
33800-33900 lines
33900-34000 lines
34000-34100 lines
34100-34200 lines
34200-34300 lines
34300-34400 lines
34400-34500 lines
34500-34600 lines
34600-34700 lines
34700-34800 lines
34800-34900 lines
34900-35000 lines
35000-35100 lines
35100-35200 lines
35200-35300 lines
35300-35400 lines
35400-35500 lines
35500-35600 lines
35600-35700 lines
35700-35800 lines
35800-35900 lines
35900-36000 lines
36000-36100 lines
36100-36200 lines
36200-36300 lines
36300-36400 lines
36400-36500 lines
36500-36600 lines
36600-36700 lines
36700-36800 lines
36800-36900 lines
36900-37000 lines
37000-37100 lines
37100-37200 lines
37200-37300 lines
37300-37400 lines
37400-37500 lines
37500-37600 lines
37600-37700 lines
37700-37800 lines
37800-37900 lines
37900-38000 lines
38000-38100 lines
38100-38200 lines
38200-38300 lines
38300-38400 lines
38400-38500 lines
38500-38600 lines
38600-38700 lines
38700-38800 lines
38800-38900 lines
38900-39000 lines
39000-39100 lines
39100-39200 lines
39200-39300 lines
39300-39400 lines
39400-39500 lines
39500-39600 lines
39600-39700 lines
39700-39800 lines
39800-39900 lines
39900-40000 lines
40000-40100 lines
40100-40200 lines
40200-40300 lines
40300-40400 lines
40400-40500 lines
40500-40600 lines
40600-40700 lines
40700-40800 lines
40800-40900 lines
40900-41000 lines
41000-41100 lines
41100-41200 lines
41200-41300 lines
41300-41400 lines
41400-41500 lines
41500-41600 lines
41600-41700 lines
41700-41800 lines
41800-41900 lines
41900-42000 lines
42000-42100 lines
42100-42200 lines
42200-42300 lines
42300-42400 lines
42400-42500 lines
42500-42600 lines
42600-42700 lines
42700-42800 lines
42800-42900 lines
42900-43000 lines
43000-43100 lines
43100-43200 lines
43200-43300 lines
43300-43400 lines
43400-43500 lines
43500-43600 lines
43600-43700 lines
43700-43800 lines
43800-43900 lines
43900-44000 lines
44000-44100 lines
44100-44200 lines
44200-44300 lines
44300-44400 lines
44400-44500 lines
44500-44600 lines
44600-44700 lines
44700-44800 lines
44800-44900 lines
44900-45000 lines
45000-45100 lines
45100-45200 lines
45200-45300 lines
45300-45400 lines
45400-45500 lines
45500-45600 lines
45600-45700 lines
45700-45800 lines
45800-45900 lines
45900-46000 lines
46000-46100 lines
46100-46200 lines
46200-46300 lines
46300-46400 lines
46400-46500 lines
46500-46600 lines
46600-46700 lines
46700-46800 lines
46800-46900 lines
46900-47000 lines
47000-47100 lines
47100-47200 lines
47200-47300 lines
47300-47400 lines
47400-47500 lines
47500-47600 lines
47600-47700 lines
47700-47800 lines
47800-47900 lines
47900-48000 lines
48000-48100 lines
48100-48200 lines
48200-48300 lines
48300-48400 lines
48400-48500 lines
48500-48600 lines
48600-48700 lines
48700-48800 lines
48800-48900 lines
48900-49000 lines
49000-49100 lines
49100-49200 lines
49200-49300 lines
49300-49400 lines
49400-49500 lines
49500-49600 lines
49600-49700 lines
49700-49800 lines
49800-49900 lines
49900-50000 lines
50000-50100 lines
50100-50200 lines
50200-50300 lines
50300-50400 lines
50400-50500 lines
50500-50600 lines
50600-50700 lines
50700-50800 lines
50800-50900 lines
50900-51000 lines
51000-51100 lines
51100-51200 lines
51200-51300 lines
51300-51400 lines
51400-51500 lines
51500-51600 lines
51600-51700 lines
51700-51800 lines
51800-51900 lines
51900-52000 lines
52000-52100 lines
52100-52200 lines
52200-52300 lines
52300-52400 lines
52400-52500 lines
52500-52600 lines
52600-52700 lines
52700-52800 lines
52800-52900 lines
52900-53000 lines
53000-53100 lines
53100-53200 lines
53200-53300 lines
53300-53400 lines
53400-53500 lines
53500-53600 lines
53600-53700 lines
53700-53800 lines
53800-53900 lines
53900-54000 lines
54000-54100 lines
54100-54200 lines
54200-54300 lines
54300-54400 lines
54400-54500 lines
54500-54600 lines
54600-54700 lines
54700-54800 lines
54800-54900 lines
54900-55000 lines
55000-55100 lines
55100-55200 lines
55200-55300 lines
55300-55400 lines
55400-55500 lines
55500-55600 lines
55600-55700 lines
55700-55800 lines
55800-55900 lines
55900-56000 lines
56000-56100 lines
56100-56200 lines
56200-56300 lines
56300-56400 lines
56400-56500 lines
56500-56600 lines
56600-56700 lines
56700-56800 lines
56800-56900 lines
56900-57000 lines
57000-57100 lines
57100-57200 lines
57200-57300 lines
57300-57400 lines
57400-57500 lines
57500-57600 lines
57600-57700 lines
57700-57800 lines
57800-57900 lines
57900-58000 lines
58000-58100 lines
58100-58200 lines
58200-58300 lines
58300-58400 lines
58400-58500 lines
58500-58600 lines
58600-58700 lines
58700-58800 lines
58800-58900 lines
58900-59000 lines
59000-59100 lines
59100-59200 lines
59200-59300 lines
59300-59400 lines
59400-59500 lines
59500-59600 lines
59600-59700 lines
59700-59800 lines
59800-59900 lines
59900-60000 lines
60000-60100 lines
60100-60200 lines
60200-60300 lines
60300-60400 lines
60400-60500 lines
60500-60600 lines
60600-60700 lines
60700-60800 lines
60800-60900 lines
60900-61000 lines
61000-61100 lines
61100-61200 lines
61200-61300 lines
61300-61400 lines
61400-61500 lines
61500-61600 lines
61600-61700 lines
61700-61800 lines
61800-61900 lines
61900-62000 lines
62000-62100 lines
62100-62200 lines
62200-62300 lines
62300-62400 lines
62400-62500 lines
62500-62600 lines
62600-62700 lines
62700-62800 lines
62800-62900 lines
62900-63000 lines
63000-63100 lines
63100-63200 lines
63200-63300 lines
63300-63400 lines
63400-63500 lines
63500-63600 lines
63600-63700 lines
63700-63800 lines
63800-63900 lines
63900-64000 lines
64000-64100 lines
64100-64200 lines
64200-64300 lines
64300-64400 lines
64400-64500 lines
64500-64600 lines
64600-64700 lines
64700-64800 lines
64800-64900 lines
64900-65000 lines
65000-65100 lines
65100-65200 lines
65200-65300 lines
65300-65400 lines
65400-65500 lines
65500-65600 lines
65600-65700 lines
65700-65800 lines
65800-65900 lines
65900-66000 lines
66000-66100 lines
66100-66200 lines
66200-66300 lines
66300-66400 lines
66400-66500 lines
66500-66600 lines
66600-66700 lines
66700-66800 lines
66800-66900 lines
66900-67000 lines
67000-67100 lines
67100-67200 lines
67200-67300 lines
67300-67400 lines
67400-67500 lines
67500-67600 lines
67600-67700 lines
67700-67800 lines
67800-67900 lines
67900-68000 lines
68000-68100 lines
68100-68200 lines
68200-68300 lines
68300-68400 lines
68400-68500 lines
68500-68600 lines
68600-68700 lines
68700-68800 lines
68800-68900 lines
68900-69000 lines
69000-69100 lines
69100-69200 lines
69200-69300 lines
69300-69400 lines
69400-69500 lines
69500-69600 lines
69600-69700 lines
69700-69800 lines
69800-69900 lines
69900-70000 lines
70000-70100 lines
70100-70200 lines
70200-70300 lines
70300-70400 lines
70400-70500 lines
70500-70600 lines
70600-70700 lines
70700-70800 lines
70800-70900 lines
70900-71000 lines
71000-71100 lines
71100-71200 lines
71200-71300 lines
71300-71400 lines
71400-71500 lines
71500-71600 lines
71600-71700 lines
71700-71800 lines
71800-71900 lines
71900-72000 lines
72000-72100 lines
72100-72200 lines
72200-72300 lines
72300-72400 lines
72400-72500 lines
72500-72600 lines
72600-72700 lines
72700-72800 lines
72800-72900 lines
72900-73000 lines
73000-73100 lines
73100-73200 lines
73200-73300 lines
73300-73400 lines
73400-73500 lines
73500-73600 lines
73600-73700 lines
73700-73800 lines
73800-73900 lines
73900-74000 lines
74000-74100 lines
74100-74200 lines
74200-74300 lines
74300-74400 lines
74400-74500 lines
74500-74600 lines
74600-74700 lines
74700-74800 lines
74800-74900 lines
74900-75000 lines
75000-75100 lines
75100-75200 lines
75200-75300 lines
75300-75400 lines
75400-75500 lines
75500-75600 lines
75600-75700 lines
75700-75800 lines
75800-75900 lines
75900-76000 lines
76000-76100 lines
76100-76200 lines
76200-76300 lines
76300-76400 lines
76400-76500 lines
76500-76600 lines
76600-76700 lines
76700-76800 lines
76800-76900 lines
76900-77000 lines
77000-77100 lines
77100-77200 lines
77200-77300 lines
77300-77400 lines
77400-77500 lines
77500-77600 lines
77600-77700 lines
77700-77800 lines
77800-77900 lines
77900-78000 lines
78000-78100 lines
78100-78200 lines
78200-78300 lines
78300-78400 lines
78400-78500 lines
78500-78600 lines
78600-78700 lines
78700-78800 lines
78800-78900 lines
78900-79000 lines
79000-79100 lines
79100-79200 lines
79200-79300 lines
79300-79400 lines
79400-79500 lines
79500-79600 lines
79600-79700 lines
79700-79800 lines
79800-79900 lines
79900-80000 lines
80000-80100 lines
80100-80200 lines
80200-80300 lines
80300-80400 lines
80400-80500 lines
80500-80600 lines
80600-80700 lines
80700-80800 lines
80800-80900 lines
80900-81000 lines
81000-81100 lines
81100-81200 lines
81200-81300 lines
81300-81400 lines
81400-81500 lines
81500-81600 lines
81600-81700 lines
81700-81800 lines
81800-8190

- `mm_to_rad(mm): float`, converts the given mm distance into expected radians of wheel rotation, and returns that value. 2 lines
- `steps_to_mm(steps): float`, converts motor steps into expected ground distance, and returns that value. 2 lines
- `mm_to_steps(mm): float`, converts expected ground distance into motor steps, and returns that value. 2 lines
- `print_pose ((x_mm, y_mm, theta_rad))`, prints x,y,theta while converting theta to degrees. 3 lines
-

Task 1: Simple open-loop¹ controller (20%)

You will first develop a simple control loop that uses the built-in robot odometry to hit a target movement distance. Start by implementing the following function:

```
move_steps(epuckcomm, l_speed_steps_s, r_speed_steps_s, l_target_steps,
r_target_steps, Hz=10): (left_steps_moved, right_steps_moved)
```

This function sets the robot's left and right wheel speed as given, and then starts a control loop that monitors the robot odometry readings to see how far (in motor steps) the robot has gone. The loop should stop after *both* the left and right targets were met (note that one may overshoot). Use `time.sleep()` to control the loop speed. After the targets are hit, make the robot stop moving and the function return. Return a tuple representing the actual left and right steps moved. Mine is 31 lines.

- You can test this with a few simple cases.
 - `move_steps(epuckcomm, R_MAX_SPEED, R_MAX_SPEED, 1000, 1000)` should move the robot forward about 13cm
 - `move_steps(epuckcomm, -R_MAX_SPEED, -R_MAX_SPEED, 1000, 1000)` should move the robot backward about 13cm
 - `move_steps(epuckcomm, R_MAX_SPEED, -R_MAX_SPEED, 1290, 1290)` should move the robot on the spot, clockwise, about one rotation
 - `move_steps(epuckcomm, R_MAX_SPEED, 0, 2580, 0)` should make the robot turn right, pivoting on the right wheel, about one rotation

Next, we will make a wrapper that lets you work in mm instead of steps.

```
move_straight (epuckcomm, distance_mm, Hz=10)
```

which commands the robot to move a given distance then stop. A negative distance indicates backward. The function returns when the motion is complete, returning the actual robot distance moved based on odometry readings. This should call `move_steps` to get the work done.

The following two lines should move the robot forward by 1m, then move back roughly to the same spot. Tune your math and measurements until you are reasonably accurate.

```
move_straight (epuckcomm, 1000)
move_straight (epuckcomm, -1000)
```

¹ Note that, in using the wheel's built in step counter, it appears as if you are in fact using a closed loop controller on the wheel motor. However, since the motor doesn't provide any actual feedback this is a guess. Further, it is open loop with respect to the robot's location within the environment as we are not having any feedback on our location, we can only guess based on what directions we gave.


Deliverables

- **Demonstration /5**

Task 2: Open-loop forward kinematics (15%)

In this task you will implement forward kinematics for the robot to convert a given robot movement into an expected real-world outcome.

Write the following function which takes in the current robot pose as a tuple (x, y, θ) and number of steps that each of the wheels have turned, and returns a tuple representing the robot's configuration after the given movement. This function does not move the robot at all but only does calculations.

 `diff_drive_forward_kin((r_x, r_y, r_theta), left_steps, right_steps):` (newx, newy, newtheta), Takes current robot pose and returns the new pose as a tuple after the left and right wheel movement in steps

Note that θ of zero means the robot is looking straight down the x axis. Tips: Using numpy with vectors and matrices makes this much easier. Mine is 31 lines.

You can test this with a few simple cases. Note that due to floating point and math errors, and your particular robot measurements, you may not have exact answers to those shown.

```
o print_pose(diff_drive_forward_kin( (0, 0, 0), 0, 0)) # should give: (0, 0, 0)
o print_pose(diff_drive_forward_kin( (10, 20, 0), 1290, 1290)) # should give:
  (178, 20, 0)
o print_pose(diff_drive_forward_kin( (10, 20, np.pi/2), 1290, 1290)) # should
  give: (10, 188, 90)
o print_pose(diff_drive_forward_kin( (0, 0, 0), -1290, 1290)) #should give (0, 0,
  0)
o print_pose(diff_drive_forward_kin( (0, 0, np.pi/2), 1290, -1290)) #should give
  (0, 0, 90)
o print_pose(diff_drive_forward_kin( (0, 0, 0), 2580, 0)) #should give (0, 0, 0)
o print_pose(diff_drive_forward_kin( (1000, 1000, np.pi/2), 1290, -1290)) #should
  give (1000, 1000, 90)
o print_pose(diff_drive_forward_kin( (0, 0, np.pi/2), 1290, 100)) #should give
  (62, 7, 283)
o print_pose(diff_drive_forward_kin( (0, 0, 0), 1991, 2075)) #should give (263,
  27, 12)
o print_pose(diff_drive_forward_kin( (0, 0, 0), 189, 2422)) #should give (-23,
  10, 312)
o print_pose(diff_drive_forward_kin( (0, 0, 0), 1249, 2598)) #should give (-11,
  152, 188)
```

Deliverables

- **Writeup: Discuss the limitations with this approach and the accuracy of the function you made. Half page. /5 (30%)**
- **Demonstration /5 (70%)**

Task 3: Simple Robot Teleoperation with Odometry (20%)

Using your differential drive forward kinematic solver from Task 2, we can now implement odometry based on the commands you give the robot. What you will do is assume that the robot starts at coordinate $x=0, y=0, \theta=0$ (looking down the x axis), create a simple program to let the user drive the robot, and along the way calculate where in the real world the robot ends up. Here are some tips

- Getting keyboard requires hooks into your OS/windowing system. I used pynput but feel free to use whatever you know. I setup events that set global variable states for moving left, right, forward, backward, for WASD, where an on_press event set the global variable to true, while on_release set to false. You don't read key states in your control loop, just read those variables.
- I created a simple control loop (e.g., like in task1) that set the robot wheel speed based on keyboard input (reading your global state variables). Every loop tick (I run at 10Hz) read the motor steps from the robot and use the forward kinematic solver to estimate the new robot location.
- Make some reasonable mapping from keys/key combinations to robot wheel speeds, nothing is prescribed but you should have flexible control of the robot.
- Print the new robot pose (position in the world) every second (10 ticks)

Once this interactive controller works, set your robot at a clearly marked start location and drive forward and backward, turn, etc., to observe how accurate the odometry approach is. My entire As1.3.py file (excluding As1lib.py) is 94 lines.

Deliverables

- **Writeup: Discuss the limitations with using odometry, including the sources of errors. How would you improve it? Half page /5 (20%)**
- **Demonstration /5 (80%)**

Task 4: Inverse Kinematics (15%)

Inverse kinematics has a well-earned reputation of being a difficult problem. Especially for non-holonomic configurations like a differential drive, many configurations may not be solvable with a simple motion – you may need a planner. For example, in this case to move to a target configuration, you should probably first turn in the right direction, drive there, then turn toward the goal orientation. We'll do this later. Instead we'll implement a much simpler form.

Write a function to calculate what would make the robot move a given distance at a given speed with a given delta turn. This is a solvable problem. For example, you could tell the robot to go 500mm at 100mm/s while turning a total of $\pi/2$ radians.

`def diff_drive_inverse_kin(distance_mm, speed_mm_s, omega_rad):` (left_steps_s, right_steps_s, left_steps, right_steps). The `distance_mm` is the absolute distance to be travelled, with `speed_mm` being the signed speed (negative means moving backward). The first two returned items represent the left and right wheel speeds in steps/s. The second two represent just how many steps each wheel should move overall (positive, unsigned). This does not move the robot at all, just performs calculations. Note the special case of turning on the spot (`distance_mm == 0`). Mine is 25 lines.

Test cases, note that your results may be slightly different based on your robot measurements.

```
o print(diff_drive_inverse_kin(130, 10, 0)) # should give (75, 75, 978, 978)
o print(diff_drive_inverse_kin(130, -10, 0)) # should give (-75, -75, 978, 978)
o print(diff_drive_inverse_kin(300, 50, 0)) # should give (376, 376, 2257, 2257)
o print(diff_drive_inverse_kin(200, 70, np.pi/4)) # should give (472, 582, 1348, 1661)
o print(diff_drive_inverse_kin(-200, 70, np.pi/4)) # should give (472, 582, 1348, 1661)
o print(diff_drive_inverse_kin(300, -40, -np.pi*2)) # should give (-134, -468, 1005, 3510)
```

```
o print(diff_drive_inverse_kin(0, 100, -np.pi*2)) # should give (753, -753, 1253, -1253)
o print(diff_drive_inverse_kin(0, 50, np.pi/2)) # should give (-376, 376, -313, 313)
o print(diff_drive_inverse_kin(0, -50, np.pi/2)) # should give (-376, 376, -313, 313)
```

Now that this works, you can connect it to your `move_steps` function from Task 1. Simply call the inverse kinematics function and use `move_steps` to drive the robot. For example:

```
move_steps(epuckcomm, *diff_drive_inverse_kin(300, -40, -np.pi*2), Hz=30)
```

Deliverables

- **Writeup: When does this chain, (diff_drive_inverse_kin to move_steps) , break? Think about the extreme values. Where should this be checked and fixed? Half page. /5 (40%)**
- **Demonstration /5 (60%)**

Task 5: Measuring Open-loop trajectories (20%)

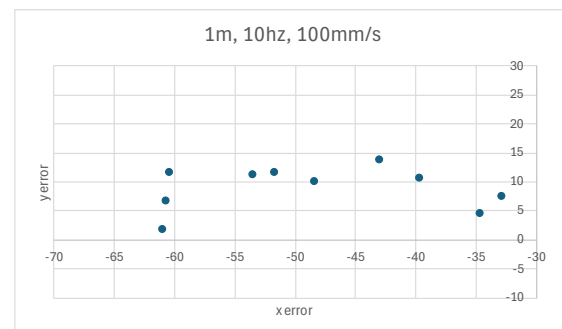
You will do a series of very simple experiments to learn about error in open-loop trajectories. For this part you will need to setup an experimental test bed that needs to be consistent throughout all your experiments.

- You need a clean, large hard surface like a table. Carpet or floors with bumps (e.g., tile) will not work. It needs to be at least 1.2m long and .6m wide, preferably bigger.
- You need to setup a mechanism for starting the robot from the exact same starting spot every time. I used electrical tape on my table, and placed a pencil mark on the robot and table.
- You need a mechanism for measuring coordinates on the table, the Δx , Δy from the robot. X is lateral and y is forward distance. Note that this is different from the robot's frame. I used a protractor and string to measure the polar coordinates which you can then convert into cartesian. Easier to do with 2 people.
- You should measure all distances from the same spot on the robot. In my case I measure the center of the front at start position vs. end position as shown.



Trials:

- Run your loop at 10Hz. Place your robot on the starting position and use `move_straight` to make the robot move forward 1000mm, at 100mm/s. Do this 10 times and record the resulting locations.
- Calculate the two components of the error, `errX` and `errY`, by subtracting your measured end point from the optimal end point. For each, calculate the average error (for X, and Y) and standard deviation (for X, and Y). Plot the error using a scatter plot. Hint: use Excel for all this. E.g., such as the plot on the right
- Repeat the above process, except process, except at 1Hz and 30Hz.



- Pick the best result (least error), and do another ten trials at 130mm/s, and take all measurements again.

Deliverables

- **Writeup: A clear lab report showing your findings including your error statistics and graphs. No analysis just results. 1-2 pages. /5**
- **Writeup: What does controller Hz do to the accuracy of odometry? Why? Half page. /5**
- **Writeup: What does robot speed do to the accuracy of odometry? Why? Half page /5**
- **Writeup: In a point form fashion, two sentences each, list all significant sources of error that contribute to your final recorded robot position error, and explain why it happens and how it can be mitigated. /5**

Task 6: More Complex Behaviors (5%)

This is an easy task, and more to highlight a problem with what we've been doing.

Code a robot behavior that (should) first move forward 0.5m, turns around, returns to its original location, then turns again to face forward – it ends in the same configuration where it started. Don't re-tune your calculations or measurements beyond what you did earlier, or you may have to re-do your experiments! Fix bugs of course, if you find them.

Next, imagine we want to make the robot to go forward for a meter, but stop immediately if it detects an object in front of it. How would you implement this using your current code structure? Write a paragraph explaining what you would do, and then discuss why this is a problem.

Deliverables

- **Writeup: What does this highlight about using odometry for more complex behaviors? Half page /5**
- **Writeup: Imagine you wanted to do this behavior, but have the robot stop immediately if it detects an object in front of it. How would you implement this using your current code structure? Discuss why this is a problem. Half page /5**

Code (5%)

Deliverables

Code: / 5