[ELE00118M] MSc Practical Robotics (PRAR)

[ELE00103M] Year 4 Robotics module

Lab Session 8 Handout

Lab 8: Robot Movement Control

Lecturer: Mark A Post (mark.post@york.ac.uk)

Technician: Mike Angus

1. Aims and Objectives

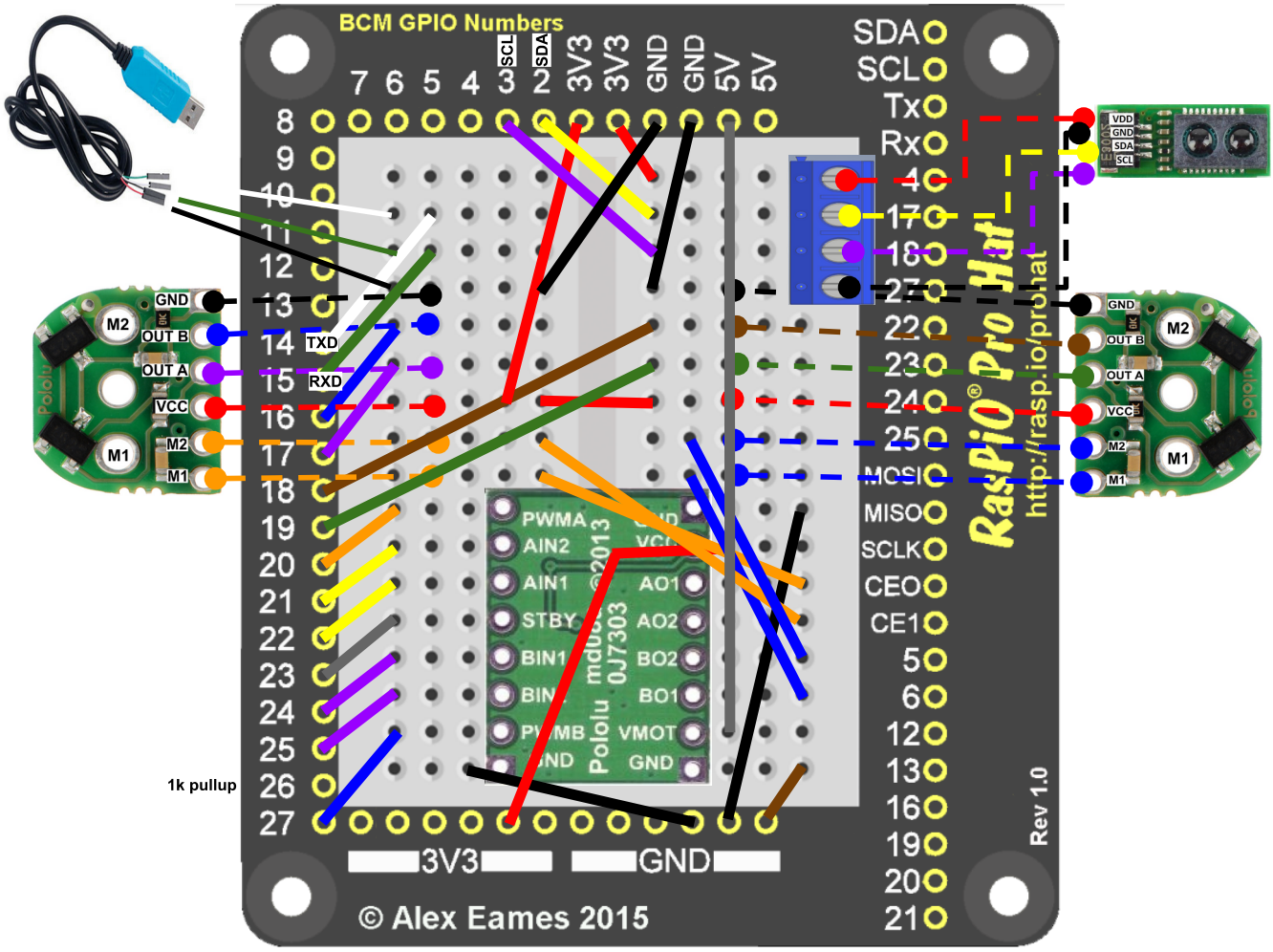
In this laboratory session, you will develop movement control code for your robot based on the previous labs.

2. Learning outcomes

- Creation of movement control functions for the robot in software
- Movement control of the robot using its sensors and motors
- Reproduction and analysis of prescribed patterns of movement

3. Software and hardware

You will be building on all of the software and hardware that you have constructed in the previous labs. By now you should have a completely constructed robot with a networking set up, a USB-serial link, the IR sensor calibrated and working, and the motors connected and moving reliably with feedback from the wheel encoders. Below is the recommended wiring diagram for all of this hardware on your robot's RasPiO Pro Hat. Make sure that all the parts of your robot are working as expected and ask for help if something is still not working as you would expect.



4. Pre-Lab Preparation

- Review the lecture materials on movement control of differential-drive robots
- Plan out how you would organize functions in your C/Python program to perform simple movement patterns with your robot
- Make a space in the lab or at home with at least a square meter of empty floor or table space available, preferably on a hard, flat surface.

5. Tasks

TASK 1: CREATE A MOVEMENT CONTROL PROGRAM

You are now at the point where you will need to make a more complex program to control your robot based on what you have learned in previous labs.

Starting with your code in C or Python from Labs 3, 4, and 5, create three new functions that can be called within your program:

```
move_forward(distance)
turn_right();
void turn_left();
```

The `move_forward()` will make the robot move forward for the number of centimeters given by distance. To create this you should first use the PWM control statements that you used for motor control in Lab 4. Note that for forward movement the two motors must turn in opposite directions as they are facing opposite each other (e.g. one CCW, one CW).

`motors.forward(float speed)` library function. You will need to experiment to choosing an appropriate speed and add functionality to your method so that it stops the robot after the correct amount of time. Test your robot over a range of distances to see how well it works over both short and long distances. Use the `motors.turn(float speed)` library function to create the `turn_right()` and `turn_left()` functions, that should make the robot turn 90 degrees in either direction. If you wish, you can create a generic turn function which will turn a given number of degrees, then make your turn_right and turn_left functions use this. In the `user_code_loop()` function of your code, make the robot move forward 50cm, turn right, then repeat 4 times and stop (so the robot returns to where it started).

Experiment with different motor speeds. You might find it is tricky to get accurate distances at higher speeds, and that the motors also don't work reliably at very low speeds – the motor needs to run at a high enough speed to get over its stalling torque. You may also notice that your robot doesn't move in a perfectly straight line when the left- and right- motor are set to the same speed; if so, consider how to compensate for this your code. It should be clear that using the sensors for navigation would help a lot with position accuracy.

You may also want to set up the light sensor as an "emergency stop" that stops the motors of your robot and places it into a stationary state when it is covered by your hand (or even when the robot flips over potentially?). Emergency stop switches are mandatory as a safety feature on any robot strong enough or heavy enough to pose a hazard to human safety.

TASK 2: SET UP AND TEST REPEATED ROBOT MOTION

To test the repeatability of your robot's motions using the space you have available on a hard, flat surface, place paper markers on the surface at the corners of a 50cm square. Then program your robot to consistently navigate around these four markers using feedback from the encoders to measure distance.

Now, try some more complex motions. Write a program that receives as input one of the following four shapes and performs the desired movement:

1. Move the robot in a straight line.
2. Move the robot in a rectangle (not equal sides).
3. Move the robot in a circle.
4. Move the robot in a figure-eight.

TASK 3: IMPLEMENT OBSTACLE DETECTION

Now, use the code you have developed for the IR sensor to detect if an obstacle is in front of your robot and stop when it gets within 10cm of the object. Take a solid, reflective object and place it in your robot's path. Does the robot stop reliably and accurately? Modify your code if needed to improve the reliability and accuracy of stopping.

Now, instead of stopping try to program your robot to move around the object and then continue with its original path. Use the IR sensor's calibrated distance measurement to try and maintain a fixed distance from obstacles. If necessary you can stop and rotate the robot to check where obstacles are also. Consider how far you are from the obstacles and try to improve the repeatability of motion.

TASK 4: ERROR DATA COLLECTION & ANALYSIS

Write a program to collect the following data:

1. Design two ways of measuring the error when your robot is moving in straight line. At least one of them must use input from both the IR sensor and encoders. Compare the two methods. What do you find?
2. Design two ways of measuring the error when the robot is rotating. At least one of them must use input from both kinds of sensors. Compare the two methods. What do you find?
3. How does error accumulate in rotation and linear movements of your robot, as a function of the power applied to the robot motors? Write down your answers.

Use these methods to collect data on the performance of your robot and try to make its movement as accurate as possible.