Comp3402 Calibrating the Robot

C.B.Price October 2021

Purpose	To learn how to drive the robot forward for a given distance in millimetres, and to realize the serious limitations of this approach.
Files Required	Arduino Sketchbook on the webpages.
ILO Contribution	LOs 1
Send to Me	nix
Homework	Read chapters 1 & 2

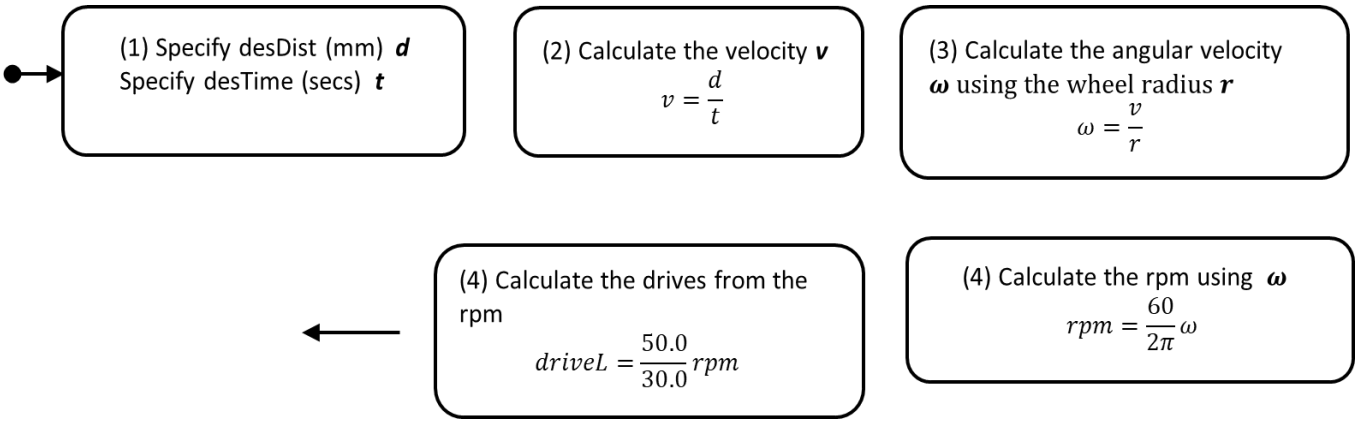
Activities

---

1 Moving the robot forwards on a straight line

We want to get our robot moving forward for a distance (in mm) we specify. We also need to specify the time (in secs) we want our robot to cover this distance. That's the input to the problem. The output from the problem (the solution) is to get values for **driveL** and **driveR** which we send to the servos.

So, we need to do some computations. The diagram below shows these, in order, a sort of flow-diagram. This refers to Chapter 2 (page 4).



The 50.0/30.0 comes from the servo graph relating **drive** (specified in the code) to wheel **rpm**, see Chapter 2 page 2). Here's a table that maps some maths symbols to variable names, used in the supplied code template

d	**desDist**
t	**desTime**
v	**desSpeed**
r	**wheelRad**
ω	**omega**
rpm	**rpm**
driveL	**driveL**

(a) Open up the sketch **CBP_3402_R_Forward_Calibrate.ino** and build up your code using the above information. The place where you should start is clearly indicated. Don't forget to drive both wheels. Choose a sensible desired distance, say 80 – 100 mm, and a reasonable time say not less than 1 second.

(b) Upload and run your sketch. Keep the robot attached to the Serial monitor using its USB umbilical so you can see the messages. ***Hold the robot above the bench top, and observe the wheels turning.*** Look at the Serial monitor printout.

(c) Now disconnect the umbilical and take the robot to the arena. Grab a piece of paper, pencil and rules. Draw a 'start line'. Switch the robot on, press reset, wait for the wheels to glitch then put the robot down. Measure how far it goes. *You will not be impressed.*

2 Calibrating the robot

So, we saw the calculation almost certainly gave you the wrong result. Correcting this is straightforward, we use the desired distance and the observed distance. We make a change to the **desTime**, e.g., if the robot has not moved far enough, we drive it for a little longer.

(a) Add the following line of code to change the **desTime**

desTime = desTime* desDist**/** actDist**;**

where actDist is the distance you measured and desDist your desired distance. You must make this change at the following location in the code (near the end of **loop()**).

```
desTime = desTime*desDist/actDist;
// Calculate the drive time in milliseconds
driveTime = (int)(desTime*1000);
cprintf("DriveTime = %d\n",driveTime);
delay(driveTime);
```

E.g. if your desired distance was 80 mm and the actual distance 70 mm, then you would code

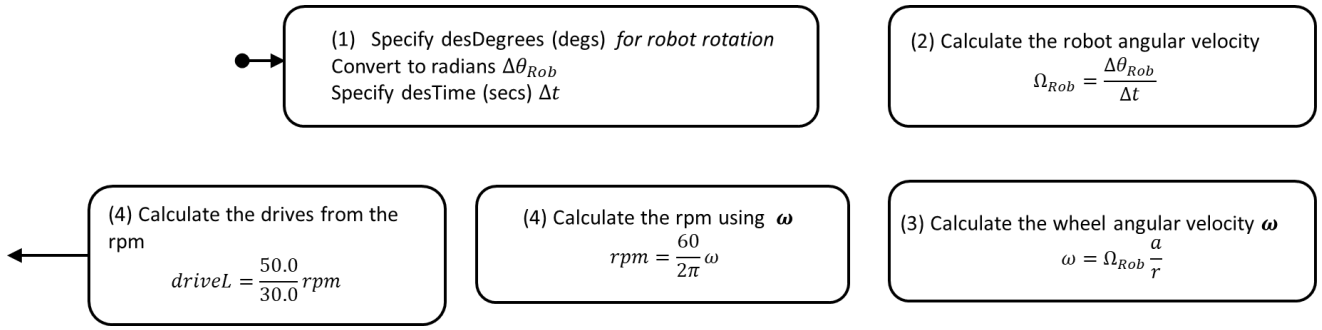
desTime = desTime80.0/70.0;**

You must divide using floats; if you wrote 80/70 then integer division may occur and give you 1. See if there is any improvement.

(b) Rerun and make some measurements. A real engineer would make several runs, note down the distances covered and make a scatter-plot of these and calculate the mean, and possibly standard deviation.

3 Getting the Robot to rotate about its centre

This is similar to the above, except we must give the robot angular speed around its centre, and no linear speed forwards. Here's the new flow diagram which starts different but ends the same.



and here's the table mapping maths symbols to variable names in code which have been declared for you.

	desDegrees
$\Delta\theta$	**desTheta**
Δt	**desTime**
Ω_{Rob}	**omegaRobot**
r	**wheelRad**
a	**(axleLen/2.0)**
ω	**omega**
rpm	**rpm**
driveL	**driveL**

(a) Open up the sketch **CBP_3402_R_Rotate_Calibrate.ino** and build up your code using the above information. The place where you should start is clearly indicated. Don't forget to drive both wheels. Choose a sensible desired angle, say 90 degrees, and a reasonable time say not less than 1 second. Remember to *drive the wheels in opposite directions!*

(b) Upload and run your sketch. Keep the robot attached to the Serial monitor using its USB umbilical so you can see the messages. ***Hold the robot above the bench top and observe the wheels turning.*** Look at the Serial monitor printout.

(c) Now disconnect the umbilical and take the robot to the arena. Grab a piece of paper, pencil and rules. Draw a 'start line'. Switch the robot on, press reset, wait for the wheels to glitch then put the robot down. Measure the angle turned

(d) Now apply a correction to **desTime** as in (2) and check that the robot rotates the desired angle.