Comp2403 Robot Kinematics - Wheel Encoders (1)

C.B.Price October 2021

Purpose	(i) To learn how wheel encoders work (ii) To get the robot to move on a straight line (iii) To get the robot to move on an arc
Files Required	Arduino Sketchbook and Octave scripts on the web-pages.
ILO Contribution	LOs 2
Send to Me	nix
Homework	Read chapter 1 (updated 09-10-21)

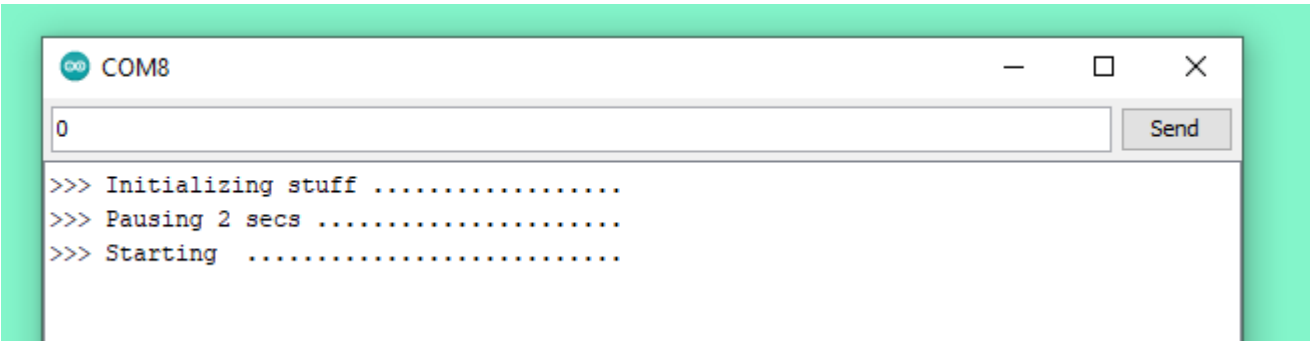
Activities

---

1 Zeroing the Servo Motors

Finally, here is the sketch we were all waiting for, to correct for any slight movement of the wheels when zero drive signals are applied. Open up the sketch **CBP_2403_R_Servo_Tweak.ino**

In what follows you will *input* values to the running program via the Serial Monitor. Here is how, see I've entered the number **0** and have pressed enter. This number will be passed to the sketch.



When you run the sketch both servos are detached.

(a) Type in **0** to attach the left servo.
(b) If it rotates, type in **+** a few times or **–** a few times until it stops. The value of **offset** which has been added to the servo drive is shown on the Serial Monitor. **Note this down.**
(c) Now enter ***** to detach the left servo
(d) Now enter **1** to attach the right servo and repeat the above steps.

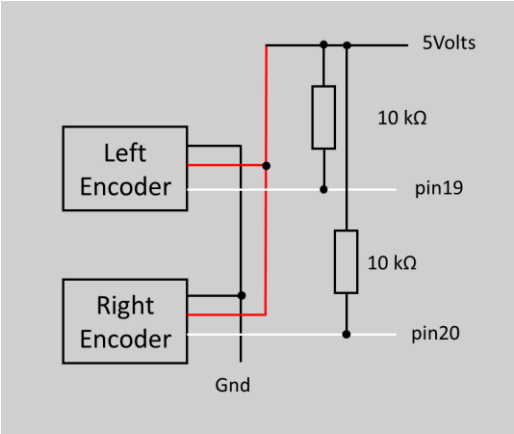
Now in all your subsequent code, you will modify the **driveServos(…)** function like this

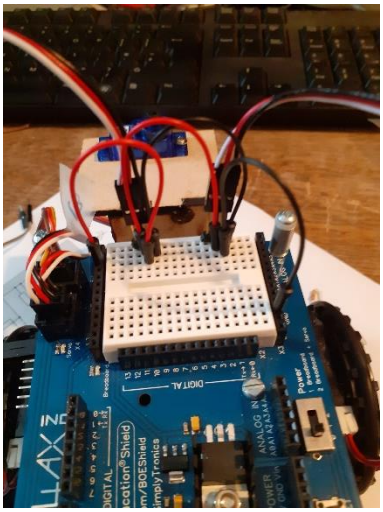
```
void driveServos(float _driveL, float _driveR) {
  servoL.writeMicroseconds(1500 + offsetL + (int)_driveL);
  servoR.writeMicroseconds(1500 + offsetR - (int)_driveR);
}
```

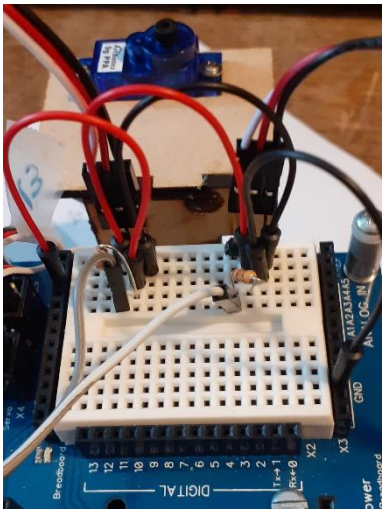
where **offsetL** and **offset** are the values you discovered from the tweak.

2 Wiring up the Encoders

Here's a circuit diagram of the encoders and some (not so useful) photos, heyho! The red, black, and white wires are the actual colours of the encoder wires. Red connected to 5Volts and Black to Gnd. Or else.







The left photo shows the wires connected, and the second shows the resistors and the connexions to Arduino pins 19 and 20. You will find flying leads connected to these.

3 Getting to know the encoders

As explained in class (and in Chapter 2), the encoder electronics provides hardware interrupts to the Arduino. The hardware receives these and immediately interrupts the running code in the **loop()** function, and transfers processing to one of two **Interrupt Service Routines (ISRs)**

(a) Open up the sketch **CBP_2403_R_EncodeTest.ino** and look for the ISRs, one for the left wheel encoder interrupt and one for the right. Make sure you understand what these functions do.

(b) Now look for the code which *attaches the interrupt* to a physical Arduino pin, and also to the correct ISR. Good.

(c) Remove the commenting-out in the ISRs, mine are at lines 70 and 75, so you will see that the ISRs are being called. Upload and fire-up the Serial Monitor.

(d) You should see some text telling you that the executing code is "In loop". Now rotate each wheel individually by hand, and you should get a message from the ISRs telling you that the executing has left the loop and is in the ISR. This really is magic and is a powerful way to connect hardware and software.

(e) Now comment out the **cprintf**'s in the ISRs and uncomment-out the **cprintf(…)** statement in **loop()** This will let you see that the values of **countL** and **countR** incremented by the ISRs make their way to the may program loop.

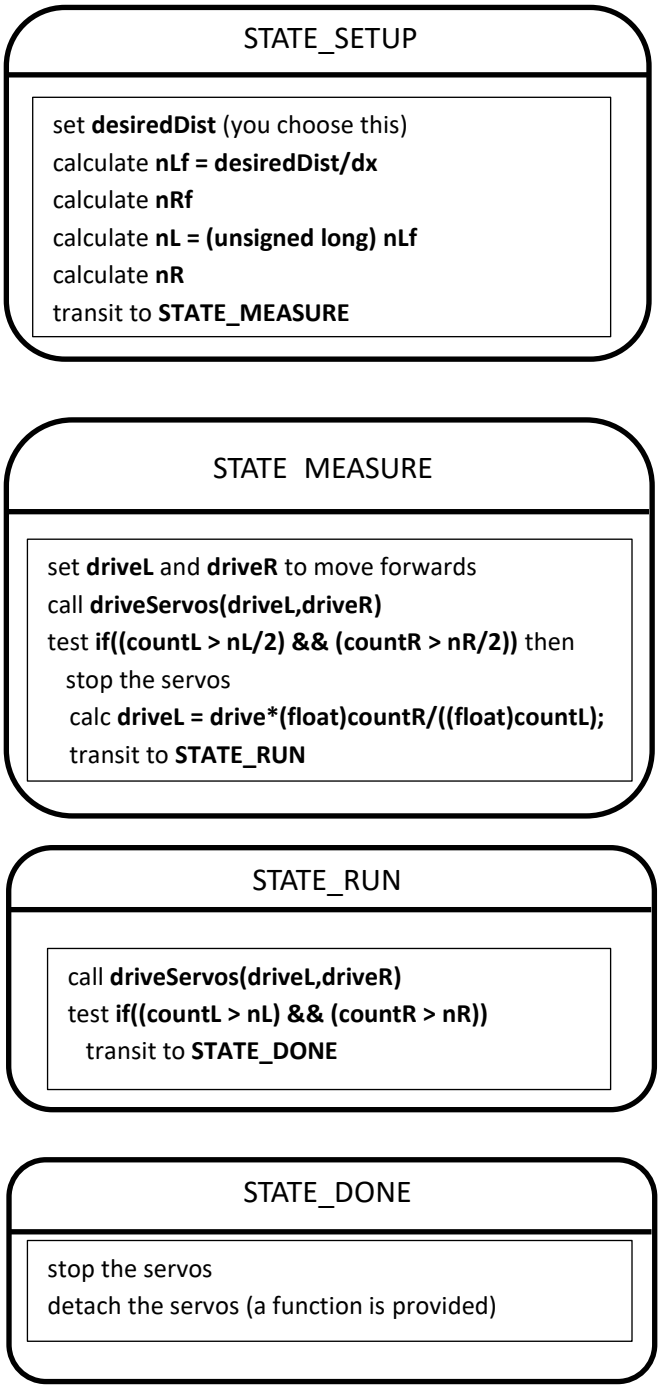
(f) Finally, start the sketch again, and manually take one wheel through one complete revolution. Look at the reported value of count. It should be close to 64.

4 Using encoders to make the robot move in a straight line – First Attempt using a FSM

As explained in class, and in Chapter 1, the whole idea of using encoders is to work out how many steps each motor should take, **nL** and **nR**. Then we count the steps each motor actually takes, **countL** and **countR**, and we apply some correction when **countL** differs from **nL** and **countR** differs from **nR**. One way of doing this is using our good old faithful FSM. The idea is covered in class and Chapter 1.

(a) Open up sketch **CBP_2403_R_StLine_FSM.ino** and you will see a template for the FSM corresponding to the state diagram presented below.

(b) Follow the guidance in the state diagram and in the code template, and code up the FSM.



(c) Compile, upload and run. Open the Serial Monitor and keep the serial umbilical connected so you can see the DEBUG messages. Hold down reset and then move the switch into position 2. Hold the robot off the ground for the first run.

(d) Now place the robot in the arena. First a poor straight line, then a perfect one?