CSC3322 Coursework — Line Measuring

Ken Pierce, March 2018

A. Aims

The aim of this coursework is to give you further experience in supervisory controller design in a control loop context, and in how designs and design trade-offs can be explored. These skills will be useful if you work in an embedded-systems or cyber-physical context in the future.

B. Submission

The coursework is to be submitted by 23:59:59 on Friday 29th April (Week 9). You must submit:

- A zip file of your INTO-CPS project.
- A zip file of your Overture project.
- A Word / PDF document containing written answers for Task 4.

The coursework is marked out of 14 and contributes 14% to your assessment for CSC3322. One mark may be deducted for poor style, commenting and readability.

To export your VDM model from Overture, select *File* > *Export...* then under *General* select *Archive File*. Select the checkbox next to your project on the following screen, set the file name then click *Finish*. For the INTO-CPS project, simply zip up the project folder.

C. Scenario

This coursework builds on the line-following robot that you worked with in Exercises 3 and 4. The robot model has an encoder on each wheel, which like the Torsion Bar from Exercise 2 measure how far the wheel has rotated. The encoders produce a count, where 44 counts is one revolution. This information can be used to calculate how far the wheel has travelled, i.e. after one rotation (44 counts) the wheel has travelled a distance equal its circumference. The average of the two encoders gives an estimate of how far the centre of the robot has travelled, and if measurements are taken at the start and end of the line being followed, then the line's length can be estimated. The closer the centre of the robot follows the line, the more accurate the measurement will be. Your goal is to add line-measuring functionality to the line-following robot to measure a line.

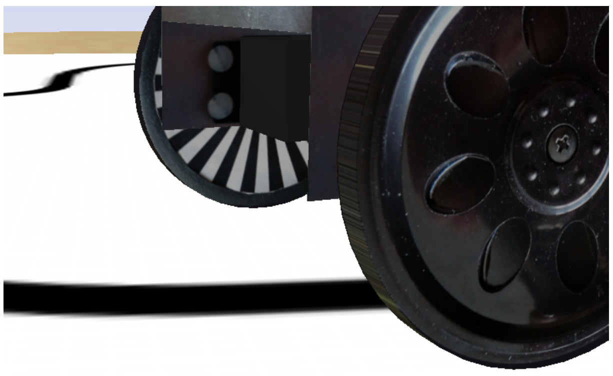

Figure 1: The encoder pattern producing 44 counts per revolution

D. Tasks

To begin, download *LMControllerFMU.zip* from Blackboard and import the project into Overture. To do this, use *File > Import...*, then select *Existing Projects into Workspace* and click *Next*. Check *Select archive file*, browse for *LMControllerFMU.zip* and click *Finish*. Also download *linemeasurer.zip*, extract it and open it in the INTO-CPS Application.

Task 1 — Encoders

The 20-sim model includes encoders for each wheel. The readings arrive at the `hwi.leftEncoder` and `hwi.rightEncoder` ports respectively. Create a class called `Encoder` with the interface below. Note that as with the right servo, the right encoder value must be negated (because the wheel turns the opposite direction to go forward), hence the **`bool`** parameter of the constructor. This class will therefore have similarities to both the `IRSensor` and `Servo` classes.

The class should hold a `RealPort` as an instance variable and the `getReading` operation should read the port (in encoder counts) and return a distance (in metres). There are 44 counts per revolution, so the conversion is `distance = counts * PI * DIAMETER / RESOLUTION`.

```
class Encoder

operations

public Encoder: RealPort * bool ==> Encoder
public getReading: () ==> real

values

PI = 3.14159
DIAMETER = 0.0665
RESOLUTION = 44

end Encoder
```

Instantiate and deploy two objects of this class in the *System* class, passing `hwi.leftEncoder` and `hwi.rightEncoder`, respectively. **[2 marks]**

Task 2 — Measurement Class

Create a class called `Measurement` with the interface below. The class should hold two `Encoder` instance variables. The `startReading` and `stopReading` operations should store the value of the encoders when called, then the `getMeasurement` operation should compute the difference between the start and end for each encoder (distance travelled), then return their average.

```
class Measurement

operations

public Measurement: Encoder * Encoder ==> Measurement
public startReading: () ==> ()
public stopReading: () ==> ()
public getMeasurement: () ==> real

end Measurement
```

Instantiate and deploy a measurement object, passing it the two encoder objects. **[2 marks]**

Task 3 — Line Measuring

Create a `Controller` class that can detect, follow and measure a line. You can base this on your model from *Exercise 3*. You should pass your `Measurement` object to the controller (along with the `Servo` objects and the five `IRSensor` objects). Your controller should drive forward until it detects a line, then start the measurement. It should then follow the line and stop at the end. After it stops it should print out the measurement.

The extra sensors could be used for line detection, e.g. dedicating a sensor to detecting the line, or for improving line-following accuracy. You can position the sensors using the `lf_position_x` and `lf_position_y` parameters of the sensor FMUs. These positions are in relation to the centre of the robot body, which is (0,0). Note that the wheels are 0.0185m (1.85cm) behind this centre. For reference, the positions from Exercise 4 were (-0.01, 0.065) and (0.01, 0.065).

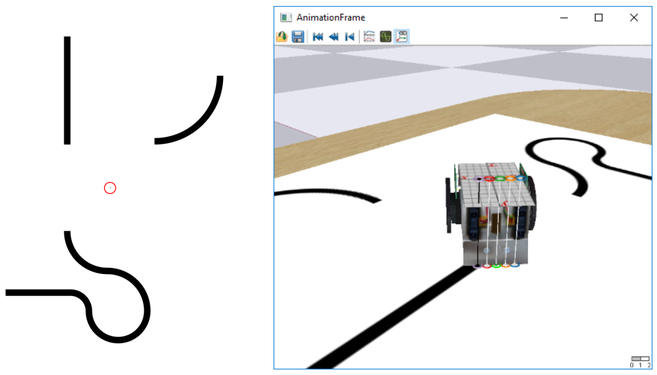

Figure 2: Straight, curved and unknown tracks with start position marked, and 3D output

There are three lines on the map: two test lines of known length (both 0.25m or 25cm) and a line of unknown length. You can point the robot at these lines using the `initial_position[2]` parameter of the body FMU. See the *Setting Parameters for Line-Measuring Robot.pdf* on Blackboard for instructions on setting the sensor positions and robot start angle.

[5 marks]

Task 4 — Reflection

Add the following to your Word / PDF document:

(a) Your design's measurement of the unknown line. Do you think this is close to the real answer, an underestimate or an overestimate? Why?

(b) A short piece of writing *reflecting* on your experience of the task, and covering the points below. Reflective writing is about your response to experiences. In reflective writing it is common to use the first person – 'I' — when necessary. See `http://www.uefap.com/writing/genre/reflect.htm` for guidance.

- How did you approach your design?
- What was your experience of using the INTO-CPS technologies?
- What would you do differently next time?

(400 words max.)

[4 marks]

Remember to submit your completed Word / PDF file along with your INTO-CPS project and exported VDM model to NESS by 23:59:59 on Friday 27th April.