Control of Mobile Robotics

CDA4621

Fall 2017

Lab 1

Kinematics

Total: 100 points

Due Date: 9-25-17 by 8am

The assignment is organized according to the following sections: (A) Lab Requirements, (B) Task Description, and (C) Task Evaluation. Note that only the questions referred to in section (C) need to be answered. Nonetheless, you should be able to answer all questions in section (B) since some of them will come up in a quiz and possibly during a theory exam.

A. Lab Requirements

The lab requires use of the "Robobulls-2017" robot hardware provided at no charge for the duration of the semester. Required software can be downloaded free of charge from the web. All labs are to be done by teams of two students. Note that no diagrams or descriptions by hand will be accepted. Each group is required to submit its report through Canvas. Penalties will be applied for submitting late assignments (see syllabus). All submitted documentation needs to be in PDF.

A.1 Hardware Requirements

The "Robobulls-2017" (Figure 1) is the main robot hardware used for the course.

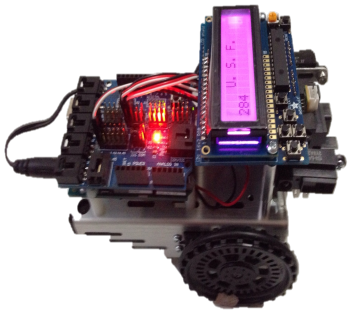

Figure 1: Robobulls-2017

A.2 Software Requirements

Arduino Software (Version 1.8.3 or later): https://www.arduino.cc/en/Main/Software

B. Task Description

The lab is divided into two parts: (1) "Robot Basics" where you will learn how to use the robot hardware, and (2) "Kinematics" where you will learn how to implement a software library and use it to move the robot along different paths.

B.1 Robot Basics

This section will cover basic usage of the hardware to gain an insight into what type of information can be found on a datasheet, how to properly use the robot hardware, and how to avoid typical errors made while working with the robot hardware.

B.1.1 Servos

Read the section "Communication Protocol" on the servos datasheet[1], and Arduino's reference about the function *writeMicroseconds*[2]. Make sure you are able to answer the following questions:

- What does the input parameter of *writeMicroseconds* control?
- Every how often does it make sense to call the function?
- What would happen in the following loop?

```
while(true) {
    servo.writeMicroseconds(1700)
    delay(20)
    servo.writeMicroseconds(1300)
    delay(20)
}
```

B.1.2 Encoders

In this section you will implement a library to use the wheel encoders. Encoders allow the robot to get feedback on motor rotations by counting the number of holes encountered by the encoder. Encoders are needed since a servo's response to the control signal varies due to several factors including floor friction, load on the motors, noise, etc. For this section, you must use interrupts[3] to implement the 4 functions on the header file "MyEncoders.h":

1) void resetCounts()
2) void getCounts(unsigned long counts[])
3) void getSpeeds(float speeds[])
4) void initEncoders()

- The first function should reset the tick count (number of holes counted) to zero.
- The second function should return the left and right tick counts since the last call to *resetCounts*, or since the start of the program (if there were no calls to *resetCounts*).
- The third function should return the instantaneous left and right wheel speeds (measured in revolutions per second).
- The fourth function should contain whatever code is necessary for initialization.

You are allowed to modify the header file but all those functions need to be implemented. Aspects to consider:

- Each encoder has 32 equidistant holes; thus, they are separated by 1/32 rotations.
- Sometimes, if the wheels shake, the values read by the encoders might oscillate making the tick count increase very fast. You should take this into account.
- The maximum speed of the servos is approximately 0.80 revolutions per second, so the shortest time interval between two holes is approximately 39ms. At half speed, the interval increases to 78ms, at one fourth it increases to 156ms, and so on. These intervals are very large when compared to the speed at which a processor can run. For this reason, it is possible to measure the speed when you haven't even counted 1 tick. This is especially true at slow speeds. If you haven't seen any new ticks since the last time you called *getSpeeds*, and you

[1] http://www.mantech.co.za/Datasheets/Products/900-00008-65S.pdf
[2] https://www.arduino.cc/en/Reference/ServoWriteMicroseconds
[3] https://github.com/GreyGnome/PinChangeInt

try to get a new measure what speed should you return? How do you know whether the robot is moving very slow or just standing still?

- Do the encoders used in the course provide information on the direction that the wheel has moved (forward / backward)?

B.2 Kinematics – Differential Navigation

In this section you will use kinematics to implement a library for differential navigation, and then you will perform several tasks using the library.

B.2.1 Differential Navigation Library

Implement the 5 functions in the header file "MyServos.h":

1) void setSpeeds(int microsLeft, int microsRight)
2) void calibrateSpeeds()
3) void setSpeedsRPS (float rpsLeft, float rpsRight)
4) void setSpeedsIPS(float ipsLeft, float ipsRight)
5) void setSpeedsvw(float v, float w)

- The first function should set the speed of the motors by giving the speeds in microseconds. You should write the code so that *microsLeft* and *microsRight* control the left and right speeds of the wheels respectively. Also, you should implement the function so that if both values are positive the robot will move forward, if both are negative it will move backwards, and if both are 0, it should not move.
- The second function should use the encoder functions implemented on section B.1.2 to create a mapping from the servos input (micro seconds) to the servos output (wheel speed in revolutions per second). In other words, measure the speeds of the wheels for different input values and store the results. You should do this for both wheels since the servo's output won't necessarily be the same for both wheels. This information will be used by the following functions.
- The third function should set the speed of the motors as in the first function, but this time, the input parameters indicate the speeds at which each wheel should spin.
- The fourth function is the same as the previous one, but this time, the speed is given in inches per second.
- The fifth function should set the speed of the robot, so that the robot will move with a linear speed given by the parameter 'v' (in inches per second), and with an angular velocity 'w' (given in radians per second). Positive angular velocities should make the robot spin counterclockwise.

Aspects to consider:

- The wheels' diameter is 2.61".
- The wheels are separated by 3.95" approximately, although this may vary from robot to robot since some may be bent.

B.2.2 Kinematics Tasks

Task 1 - Run the function *setSpeeds(0,100)*. Use the encoders to plot the *instantaneous speed*[4] of the right servo (in revolutions per second) vs time. Take measurements every 30 ms for

[4] https://en.wikipedia.org/wiki/Speed#Instantaneous_speed

approximately 2 seconds (about 67 measurements in total). Before taking any measurements wait 1 second to allow the motor to reach the desired speed. Aspects to think about:

- Is this value constant?
- If not, does the plot look random or does it have a pattern? Think of reasons that might explain the behavior.

Task 2 - With a fully charged set of batteries, in a single graph use the encoders to plot the speed of both wheels (in revolutions per second) vs input to the implemented function *setSpeeds*. Measure the speeds of the wheels from -200 micros to 200 micros in increments of 10. Repeat the experiment with low batteries. Provide both plots in the report. Things to think about:

- How similar do both servos react to the same input? Does the difference look like random noise? If not, what could be causing the difference? Why is it a good idea to use software calibration?
- Before measuring the speed, you should allow some time for the servo to stabilize its speed. You may want to use the average speed over a time window instead of the instantaneous speed.

Task 3 - Implement the program 'Forward.ino'. The program should make the robot stay still until the select button is pressed. After pressing select, the robot should move forward on a straight line for "X" inches. The robot should complete the movement in "Y" seconds. The robot should stop based on encoder readings (distance traversed), not on the time past. "X" and "Y" are parameters provided on "ForwardParams.h". The program should work for any values "X" and "Y" for which it is possible to complete the movement on the given time. If the robot can't complete the movement in the given amount of time, when pressing select, instead of moving forward, the robot should display an appropriate message on the LCD.

During the task presentation, the TA will assign different values for "X" and "Y".

Task 4 - Implement the program 'SShape.ino'. The program should make the robot move in two semicircles of radius "R1" and "R2" respectively, as shown in Figure 2. The robot should stay still until the "select" button is pressed. When "select" is pressed, the robot should perform the first semicircle in a clockwise movement. After completing the first semicircle, the robot should wait for the "select" button to be pressed again, and then perform the second semicircle in a counterclockwise manner. The whole movement must be completed in "Y" seconds moving at the same constant speed for both halves. The robot should stop based on encoder readings (distance traversed), and not based on a given time.

"R1", "R2" and "Y" will be provided on "SShapeParams.h". The program must work for any values of "R1", "R2" and "Y". If the motion cannot be completed in the given time, after pressing "select" for the first time, the robot should display an appropriate message on the LCD.

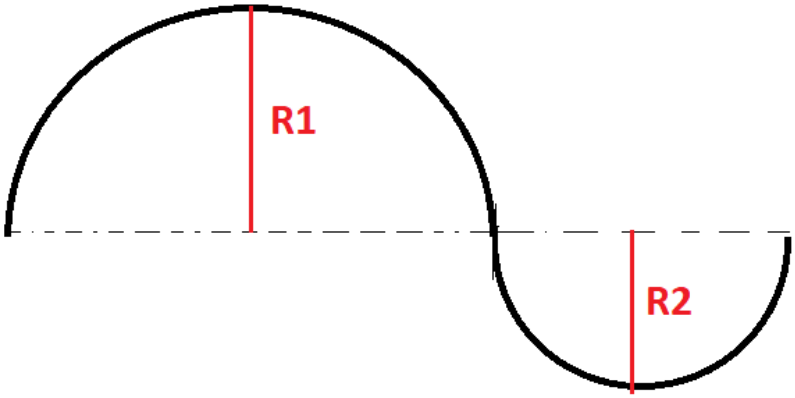

Figure 2. Path to be followed by the robot.

- Observe that, independently of the trajectory, the robot moves by fixing the speeds of the wheels for a given amount of time (as in the "S" shape shown in Figure 2). Taking this into consideration, can the robot move in any kind of trajectory? Consider an ellipse as an example.

During the task presentation, the TA will assign different values for "R1", "R2" and "Y".

Task 5 - Implement the program "Ellipse.ino". The program should make the robot move in an ellipse given by the formula $\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1$ (as seen in Figure 3). The robot should stay still until the "select" button is pressed. When the button is pressed, the robot should move on an ellipse at constant linear speed "V". The parameters "a", "b", and "V" will be provided on "EllipseParams.h". The program should work for any assigned values. On this part, you do not need to worry on whether the robot can traverse the ellipse at the given speed, but you have to include in the report what limits the possible values of "V". The robot should start on the leftmost point of the ellipse, corresponding to position (-a, 0) facing the positive y-axis direction. The robot should move in clockwise manner.

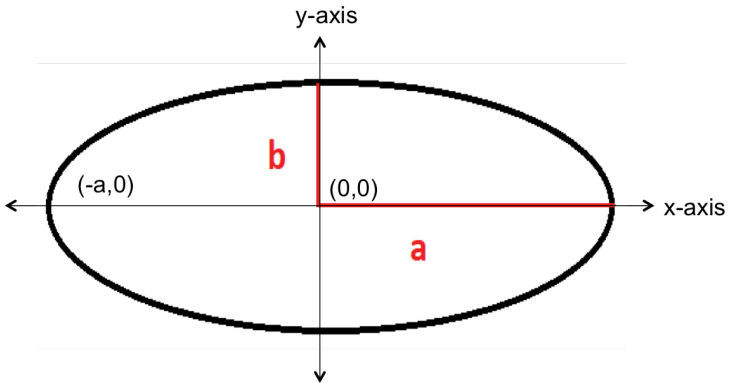

Figure 3. Path to be followed by the robot.

During the task presentation, the TA will assign different values to "a", "b" and "v".

Hints and notes:

- The objective of this task is for you to understand the difficulties of doing a non-trivial curve. You should strive to draw the ellipse the best you can but don't expect results to be perfect.

- Use the material seen in class to calculate curvature of an ellipse as a function of the position. Use the formula $\omega = kv$.

C. Task Evaluation

Task evaluation involves: (1) a task presentation of robot navigation with the TA, (2) the accompanying code and task report to be uploaded to Canvas, and (3) an online quiz. All uploaded documents will be scanned through copy detection software.

C.1 Task Presentation (70 points)

The task presentation needs to be scheduled with the TA. Close to the project due date, a time table will be made available online from which groups will be able to select a schedule on a first come first serve basis. All team members must be present at their scheduled presentation time. On the presentation day, questions related to the project will be asked, and the robot's task performance will be evaluated. To do so, the submitted code will be downloaded from Canvas and uploaded to the robot.

C.2 Task Report (25 Points)

The accompanying task report needs to be uploaded to Canvas as a PDF file together with ALL the files required to run the robot navigation. Upload all files into a single "zip" file. The task report should include ALL of the following (points will be taken off if anything is missing):

1. List of all code files uploaded to canvas. All requested code must be included in the list. A one-line description for each file to which task it relates to must be added.
2. Equations used to calculate the speed of the wheels in the encoder library, and the equations used in functions *setSpeedIPS* and *setSpeedsvw*. A brief description should be provided for each equation.
3. The plots for kinematics "Task 1" (*instantaneous speed* [5] *of the right servo vs time*) and "Task 2" (*speed of both wheels vs input: for fully charged batteries & low batteries*) referred on section B.2.2 (3 plots in total). All plots must include title, axis names, units, sufficient tick marks and legend (if plotting more than one graph). Also, each data point should clearly be marked with a symbol. Plots of a variable vs time should always place time on the x-axis. See sample plot in Figure 4.

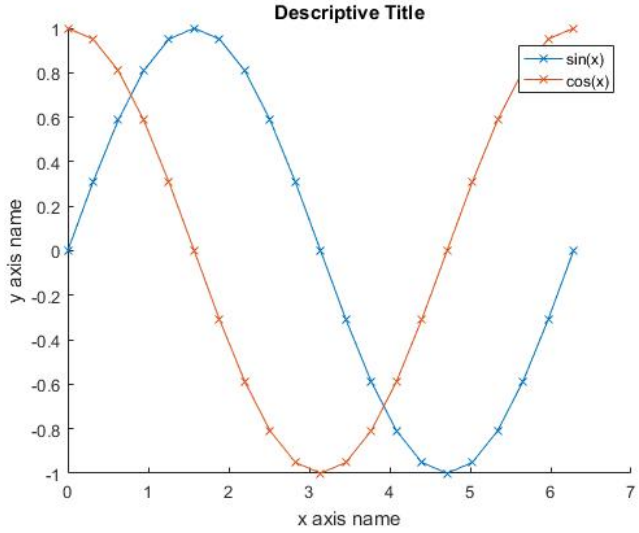

Figure 4. Sample Plot

[5] https://en.wikipedia.org/wiki/Speed#Instantaneous_speed

4. Mathematical solutions for kinematics “Task 3”, “Task 4”, and “Task 5” referred on section B.2.2.
5. Answer to the question on kinematics “Task 4” referred on section B.2.2. You need to back up your answer with an argument; otherwise it will be marked as incorrect.
6. Conclusions where you analyze any issues you encountered when running the tasks and how these could be improved. Conclusions need to show an insight of what the group has learnt (if anything) during the project. Phrases such as “everything worked as expected” or “I enjoyed the project” will not count as conclusions.

C.3 Online Quiz (5 Points)

An online quiz based on section “Robotic Basics” will be due Thu Sept 7thth 8am.

C.4 Suggested Submission Schedule

Week 1 – “Robotic Basics”, turn in the code for section B.1.2 due Thu Sept 7thth 8am.
Week 2 – Turn in the code and plots for differential Navigation and Kinematics, Tasks 1 and 2, due Mon Sept 11th 8am.
Week 3 – Turn in the code for Kinematics, Tasks 3 and 4, and their mathematical solutions Mon Sept 18th 8am.
Week 4 – Turn in the code for Kinematics Task 5 and the full report Mon Sept 25th 8am.