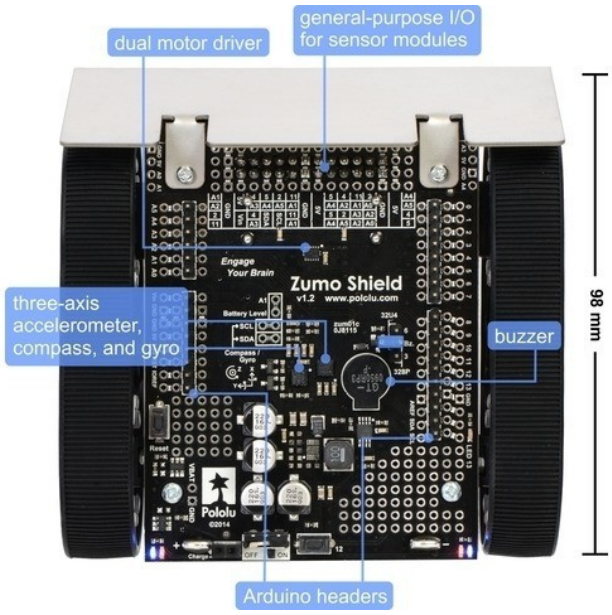
ENGR 207: Programming Robots and Sensors
Lab 4: Introduction to the Zumo 32U4 Robot: Open-Loop Control

Assigned: 27-March-2017
Due: 3-April-2017

Introduction

In the past three labs we have learned to assemble basic circuits by following electrical schematics and we have learned to use the Arduino micro-controller to read sensors and control actuators. These are fundamental skills that are required to design, construct and program robots. In today's lab we transition to working with a fully assembled robot that is also based on the Arduino architecture. The Zumo 32U4 is a small (10cm x 10 cm) mobile robot that is equipped with a variety of sensors including:

- three-axis accelerometer
- compass
- gyrsocope
- infrared proximity sensors
- reflectance sensors (for line following)



We will explore how to control the motion of this differential drive robot by sending open-loop motor commands. The Zumo32U4 library provides us with a variety of functions that we can use to interact with its sensors and actuators. In this lab we are primarily interested in using the `setMotorSpeeds` method of the `Zumo32U4Motors` class. To use this class we first declare an object of type `Zumo32U4Motors` at the beginning of the Arduino sketch:

```
Zumo32U4Motors motors;
```

Then we may call the `setSpeeds` method with two arguments that specify the rate at which the left and right pairs of wheels should be rotated:

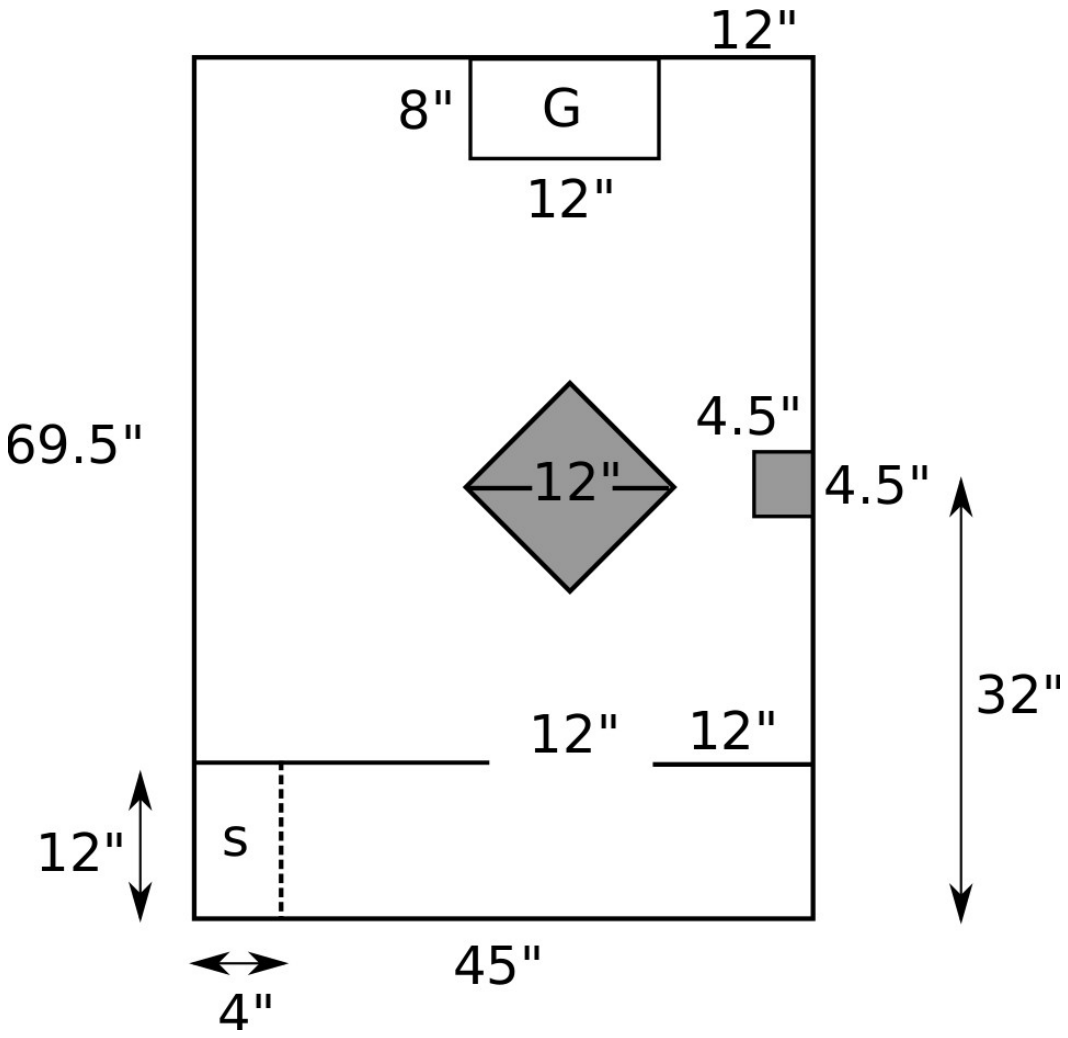
```
motor.setSpeeds( leftWheel, rightWheel)
```

In Zumo syntax the two arguments are integers that can range from -400 (full speed in reverse) to 400 (full speed forward). Of course, a command of `motor.setSpeeds(0,0)` will cause both wheels to remain stationary.

Recall from our study of locomotion that a differential drive robot can either turn in place by rotating its wheels (on the left and on the right) in opposite directions, or it can turn while moving by rotating one set of wheels more slowly than the other.

In this lab you will program the robot to travel from the start region labeled “S” in the diagram below, to the goal region labeled “G” while avoiding all obstacles (include the boundaries of the stage and the shaded regions). The stage is designed with a white-board-like material thus you are permitted to mark up the area (if you find this helpful).

The robot will execute the programmed commands in a “open-loop” fashion meaning that it will not use its sensors to provide feedback that accounts for modeling errors or disturbances.



Lab Instructions:

1) Follow the instructions provided in "L4_Installing_Zumo_Libraries.pdf" to install and test the Zumo 32U4 libraries.

2) Download the ZumoLab4.ino sketch from Blackboard and move it into your working directory.

3) Open the sketch in the Arduino IDE.

4) Upload the sketch to the Zumo to see different ways in which the Zumo can move.
   - go forward then back (slowly)
   - go forward then back (fast)
   - spin in place (to the left, slowly)
   - spin in place (to the right, fast)

   * Pay careful attention to how the motor.setSpeeds(,) command is used in each case.

5) Rather than specifying low level motor speed commands we can write functions that have more intuitive inputs (such as distance to travel, or degrees to turn). The `goStraight` function is used to drive the robot in a straight line and has the following syntax:

```
void goStraight( Zumo32U4Motors motor, float inches,
                 int leftWheelBias,
                 int rightWheelBias, int pauseTime)
```

   The inputs are:

motor: the object referring to the Zumo32U4Motors
inches: desired length the robot should travel along a straight line
leftWheelBias: a speed offset for the left wheel
rightWheelBias: a speed offset for the right wheel
pauseTime: time (in milliseconds) the robot should pause after the command

6) A related function is (partially) provided for turning in place:

```
void turnInPlace(Zumo32U4Motors motor, float deg,
                 int dir, int leftWheelBias,
                 int rightWheelBias, int pauseTime)
```

motor: the object referring to the Zumo32U4Motors
deg: desired number of degrees to turn
dir: (+1) indicates a clockwise (left) turn, while (-1) indicates a counter-clockwise (right) turn
leftWheelBias: a speed offset of the left wheel
rightWheelBias: a speed offset of the right wheel
pauseTime: time (in milliseconds) the robot should pause after the command

7) Write an appropriate motor command on line 45 to ensure the robot turns as desired.

Before attempting to complete the obstacle course we will "tune" the goStraight and turnInPlace functions:

8) Use the `goStraight` function to command the robot to go forward some distance (e.g., 24 inches). Measure the distance the robot actually traveled. Adjust the parameter `dist2delay` on line 24 so that the command is executed for a longer or shorter period (as desired).
9) Use the `turnInPlace` function to command the robot to rotate 360 degrees. Based on your judgment of how well the robot executed the command, adjust the `deg2delay` parameter on line 43.

Proceed once you are satisfied that the robot can, with reasonable accuracy, travel a specified distance and turn a specified angle.

10) Write a sequence of commands that allow the robot to reach the goal region without hitting any obstacles or the boundaries of the stage. This may take a few dozen iterations.

11) Now we wish to complete the obstacle course (again) without a new turn function:

```
void turnWhileMoving(Zumo32U4Motors motor, float deg,
                     int dir, int leftWheelBias,
                     int rightWheelBias, int pauseTime)
```

Write this function to allow the robot to turn while moving. Use a similar "tuning" test as in 9).

12) Repeat Step 10) with the new turning function.

Required Submission

To demonstrate that you have completed the lab you must send an email to wolek@cua.edu with the following (include the names of all your group members):

- (10 pts total) Two (2) videos of the robot completing the course, using the "turn in place" approach and the "turn while moving" approach. Alternately, you can demonstrate your robot completing the course during the lab session or before/after another class prior to the due date.
- (10 pts total) Two (2) Arduino codes corresponding to the above demonstrations. Grading will be based on quality of comments and clarity of the code.
- (10 pts total) One (1) group document that describes:
  - (5 pts) How your group solve the problem and what you learned. Discuss any difficulties that were encountered and how they were overcome.
  - (5 pts) Provide the total duration of each of your codes (in milliseconds or seconds)
- You must promptly return the Zumo robot, USB cable, and tape measure (if applicable) to the Instructor when the lab assignment is complete.

Bonus Points: Members of the group that completes the course with the fastest average time (for the two approaches) will receive 5 bonus points that will automatically be added to their lowest exam or homework grade. To receive the bonus points a video is required to verify the claimed time.