Control of Mobile Robotics

CDA4621

Spring 2017

Lab 2

Wall Following Task

Total: 100 points

Due Date: 2-27-17 by 8am

A. Lab Requirements

The lab requires use of the course robotic hardware ("Robobulls-2017") provided to students at no charge for the duration of the course. 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 student is required to submit his or her joint report through CANVAS. Penalties will apply to any individual student submitting a late assignment even if the partner has already submitted it. Accepted documentation file types are PDF, Word (doc, docx), and Powerpoint (ppt, pptx). You need to upload all your code together with the document as part of a "zip" or "rar" file.

- Hardware Requirements

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

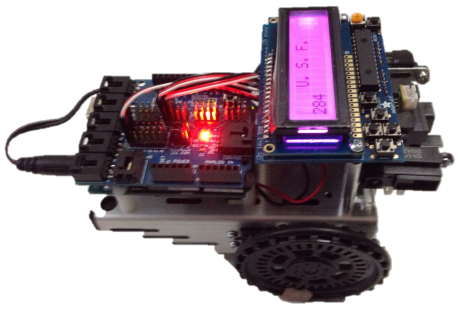

Figure 1: Robobulls-2017

- Software Requirements

Arduino Software (Version 1.8.1 or later)
https://www.arduino.cc/en/Main/Software

B. Task Evaluation

Each individual task is worth a specific number of points where these points are always split 50% between Task Execution and Task Report:

- Task Execution

The robot should execute the task correctly with a video clearly and completely showing the task execution (points will be taken for errors or missing aspects of task execution).

- Task Report

Each task report requires an accompanying document to be uploaded to Canvas together with ALL the files required to run the program in the robot. The task report needs to include ALL the following sections (points will be taken off if anything is missing):

1. Task description.
2. Solution describing the conceptual algorithm used to solve the task described in terms of flow charts to describe the logic of the program and block diagrams to describe the various robot components.
3. Video link to different task executions (you should split each task execution as a different video link most preferably in YouTube making sure the video is public to all). Provide at the beginning of each video your name and description of the task being performed.
4. Description of the code used to program your robot with explanations that clearly relate to the solution previously described.
5. Images taken from the actual robot task execution (at least one image per task).
6. Conclusions where you analyze any issues you encountered when running the task and how these could be improved.

C. Task Description

The goal of this lab is to perform close loop control using distance sensors and robot motor actuators. The lab consists of 3 tasks, each building on results from the previous task. Figure 2 presents a diagram for close loop control, with robot velocity proportional to its desired distance to a goal, $r(t)$, where the goal may be any object such as a wall or an obstacle. As the robot gets closer to the desired distance to the goal, the error $e(t)$ will modify the control signal $u(t)$ until such error becomes 0. In our case the control signal corresponds to robot motor velocity until it becomes 0 when the robot reaches a specific distance to the goal.

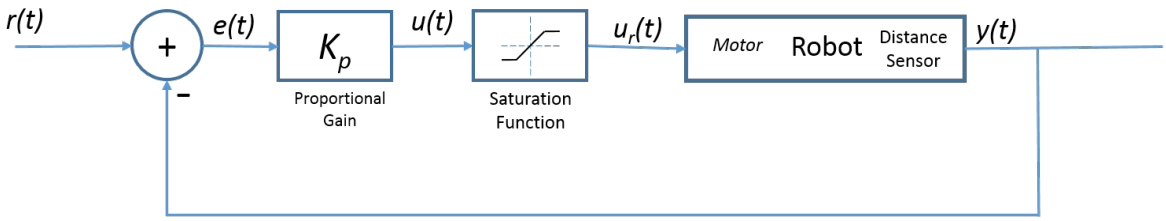

Figure 2: Close loop control of the robot velocity proportional to its distance to a goal.

Eq. 1-5 summarizes the diagram in Figure 1:

$$u(t) = K_p * e(t) \quad \text{(Eq. 1)}$$

$$u_r(t) = f_{Sat}(\, u(t)) \quad \text{(Eq. 2)}$$

$$u_r(t) = f_{Sat}(\, K_p * e(t)\,) \quad \text{(Eq. 3)}$$

$$e(t) = r(t) - y(t) \quad \text{(Eq. 4)}$$

$$u_r(t) = f_{Sat}(\, K_p \left(r(t) - y(t)\right)\,) \quad \text{(Eq. 5)}$$

where:

$r(t) = desired\ distance\ to\ the\ goal$
$y(t) = distance\ from\ robot\ to\ the\ goal$
$e(t) = distance\ error$
$K_p = proportional\ gain\ or\ correction\ error\ gain$
$u(t) = control\ signal\ corresponding\ to\ robot\ velocity$
$f_{sat} = Saturation\ Function\ (See\ Figure\ 3)$
$u_r(t) = control\ signal\ corresponding\ to\ saturated\ robot\ velocity$
$u_r(t)$ → RServo.write(u_r)

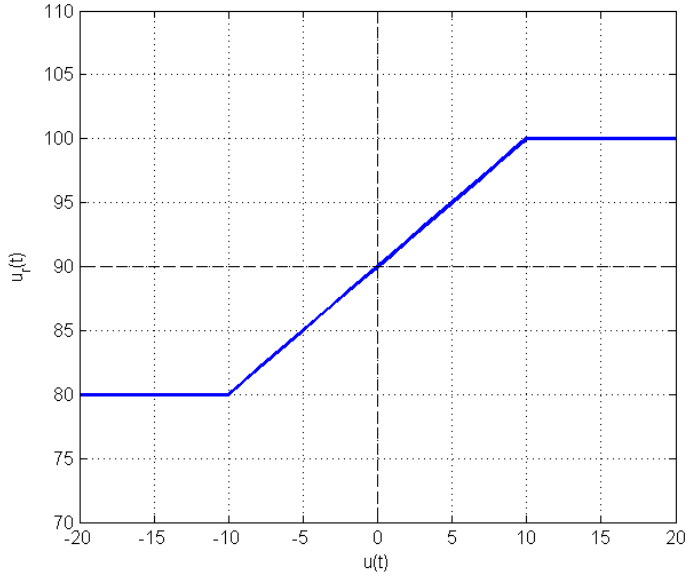

Figure 3: Saturation function showing cutoff for max positive motor velocity at $u(t)$ = 100, and min negative motor velocity at $u(t)$ = 80, where $u(t)$ = 90 corresponds to 0 velocity.

1. Wall Distance (10 points)

You need to program the above equations (1-5) in order to implement the close loop control program where the control signal velocity $u(t)$, corresponding to robot motor velocity, will move the robot from an initial distance of 10 inches to a desired distance of 5 inches from the wall, i.e. $r(t)=5$, as shown in Figure 4. The distance should be measured using the front short distance sensor. The task has to be performed using 5 different values of K_p(0.5, 1, 3, 5, and 20). For each value of K_p you will need to provide a graph showing Time vs. Distance from Goal for each individual K_p, in addition to all sections required in the task report.

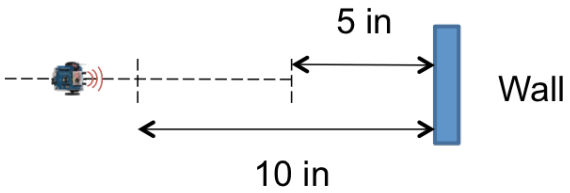

Figure 4: Wall Distance Task

2. Wall Following (40 points)

Task 2 requires the robot to move around the wall while keeping a min 5 inches of distance from the wall, as shown in Figure 5. Use the same close loop control implementation as in Task 1 with appropriate proportional gains to control front distance to the wall where robot needs to turn.

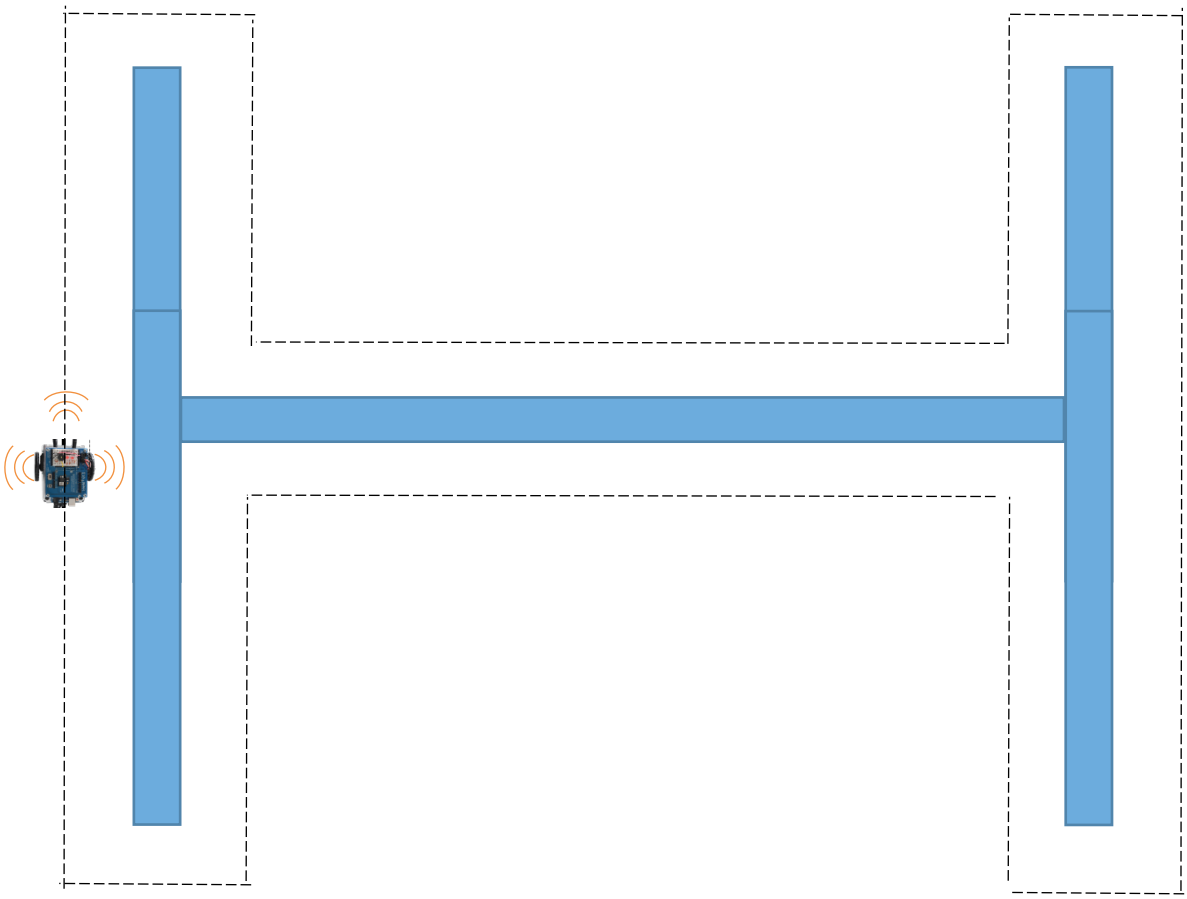

Figure 5: Wall following task

3. Color Classification (10 points)

Task 3 requires the robot to be able to discriminate between three arbitrary colors. One color will correspond to the floor color, whereas the other two colors will be used in the next task to locate start and end position, as shown in Figure 6. When the robot is put on top of each of the colors, it should print the color name on the LCD screen.

4. Corridor Navigation (40 points)

Task 4 requires the robot to navigate inside a corridor, while keeping the same distance between walls, as shown in Figure 6. Use the same close loop control implementation as in Task 1 with appropriate proportional gains when reading distances to front walls where robot needs to turn. Avoid crashing to any of the walls. Two patches of the previously chosen colors in Task 3 will be laid out on the floor as shown in Figure 6. The robot will use these patches to detect when it

has completed an entire loop. The robot should stop upon detecting the same color for the second time, after it has looped around the maze.

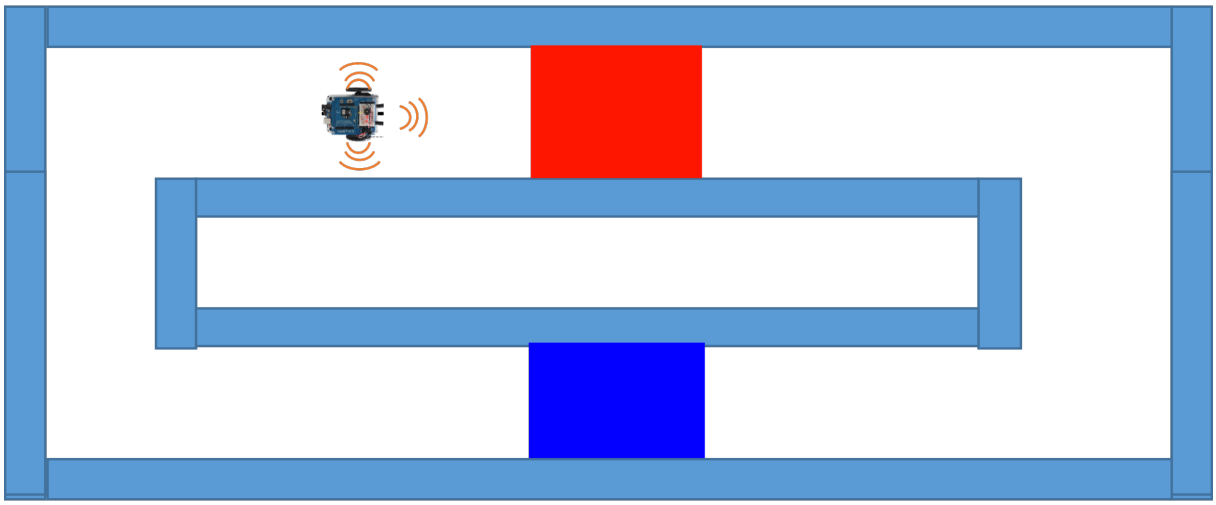

Figure 6. Corridor navigation task