Modular Design of Assembly Parking Controller Implemented with Simple VHDL Computer for DE2Bot

Noah Roberts
Zachary Russell
Arjun Sabnis
Justin Vuong
Thomas Wyatt

Georgia Institute of Technology
School of Electrical and Computer Engineering

Submitted
April, 7, 2017

Executive Summary

This document examines the proposed solution to the parking of a DE2Bot in a designated space, using IR remote controls for manual movement, and the odometer built into the robot for automatic movement. This involves designing a modular controller on an assembly code with two major functionalities, manual controls automatic sequences, to be implemented on the DE2Bot. The manual controller, using IR remote buttons, will provide full coverage of basic DE2Bot movements, including forward and backward motion, braking, and complete right-angle turns on axis. The automatic perpendicular parking sequences are implemented using the on-board odometer. The positive X position of the DE2Bot, dependent on its forward movement, will be compared to a predetermined measured distance, to guide the bot towards a chosen parking spot. Sonar interrupts are also used during both parking sequences to make sure the parking space selected is clear of any obstacles.

Modular Design of Assembly Parking Controller Implemented with Simple VHDL Computer for DE2Bot

Introduction

This document proposes a solution to the parking project by using IR remote controls passed to the DE2Bot for manual control, and use of the DE2Bot's on-board odometer for automatic parking, coded into an assembly file. These will allow a user to manually guide and semi-autonomously park the robot in either a parallel or perpendicular parking space, or autonomously send the robot to park in one of seven possible perpendicular parking spaces. The manual functions will be coded as subroutines linked to buttons on an IR remote, which allow for basic movement, including forward and backward motion, turning in place, and parallel or perpendicular parking after manual driving. The automatic functions will utilize the odometer to recognize the robot's position and guide it in the completely automatic perpendicular parking sequence. In addition, the sonar sensors will be used to check if a spot is occupied, and will interrupt the parking routine if there is an obstacle in the way.

Technical Approach

<u>Problem Description</u>

The goal of this project is to design a control program for a DE2Bot that can perform the following tasks:

1. Parallel park after manually driving to a parking space.
2. Perpendicular park after manually driving to a parking space.
3. Navigate from the start point to a perpendicular parking spot and park autonomously.

The map of the parking lot is shown in Figure 1 on the next page. The bot will have obstacles such as a curb, other faux bots that occupy parking spaces, and a wall surrounding the perpendicular parking area of the parking lot.

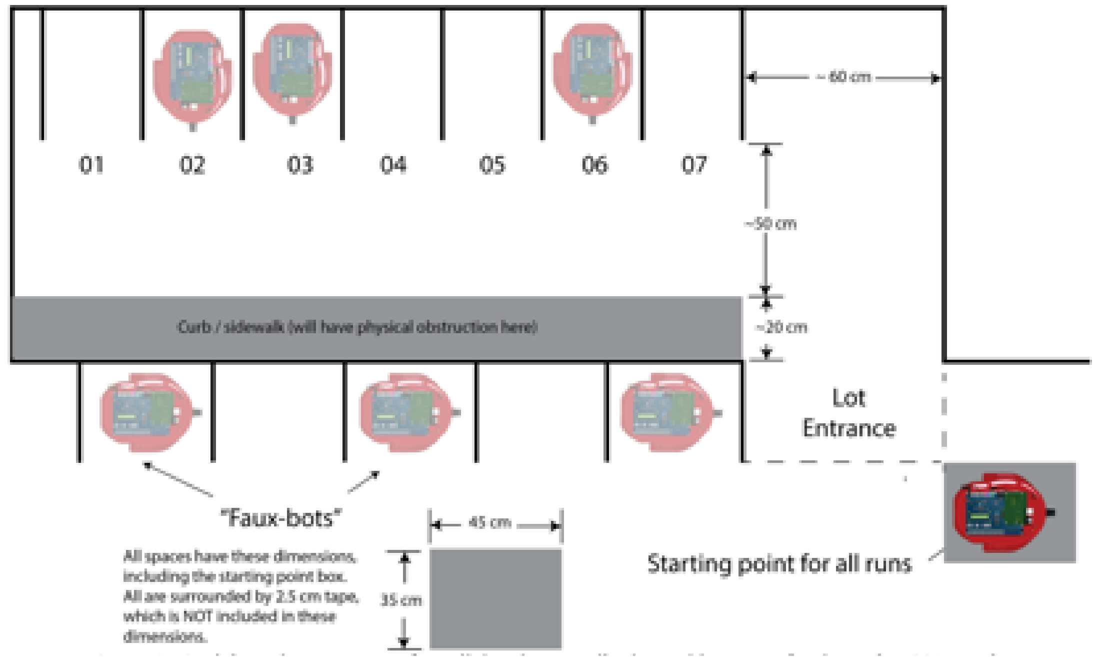
~60 cm
01
02
03
04
05
06
07
~50 cm
Curb / sidewalk (will have physical obstruction here)
~20 cm
Lot Entrance
"Faux-bots"
45 cm
All spaces have these dimensions, including the starting point box. All are surrounded by 2.5 cm tape, which is NOT included in these dimensions.
35 cm
Starting point for all runs

Figure 1. Map of parking lot. The DE2Bot must navigate through the parking lot, and parallel park and perpendicular park after manual driving. The bot must also perpendicular park without any manual driving after leaving the starting point.

<u>Manual Driving and Parking using IR Remote</u>

The manual driving portion of the demo will require the use of subroutines that need to be coded into the assembly file. The manual driving control will have 8 different subroutines:

1. WaitForIR - The DE2Bot waits until an IR signal is received from the remote
2. Stop - The DE2Bot does not move until a valid code is received
3. FW - moves forward continuously until a button is pressed
4. BW - moves backwards continuously until a button is pressed
5. LT90 - turns the DE2Bot 90 degrees counterclockwise
6. LT15 - turns the DE2Bot 15 degrees counterclockwise
7. RT90 - turns the DE2Bot 90 degrees clockwise
8. RT15 - turns the DE2Bot 15 degrees clockwise.

A state diagram displaying the functionality of manual controls can be seen in Figure 2 on the next page, and table 1 shows the button mapping for all functions that will be implemented in the robot.

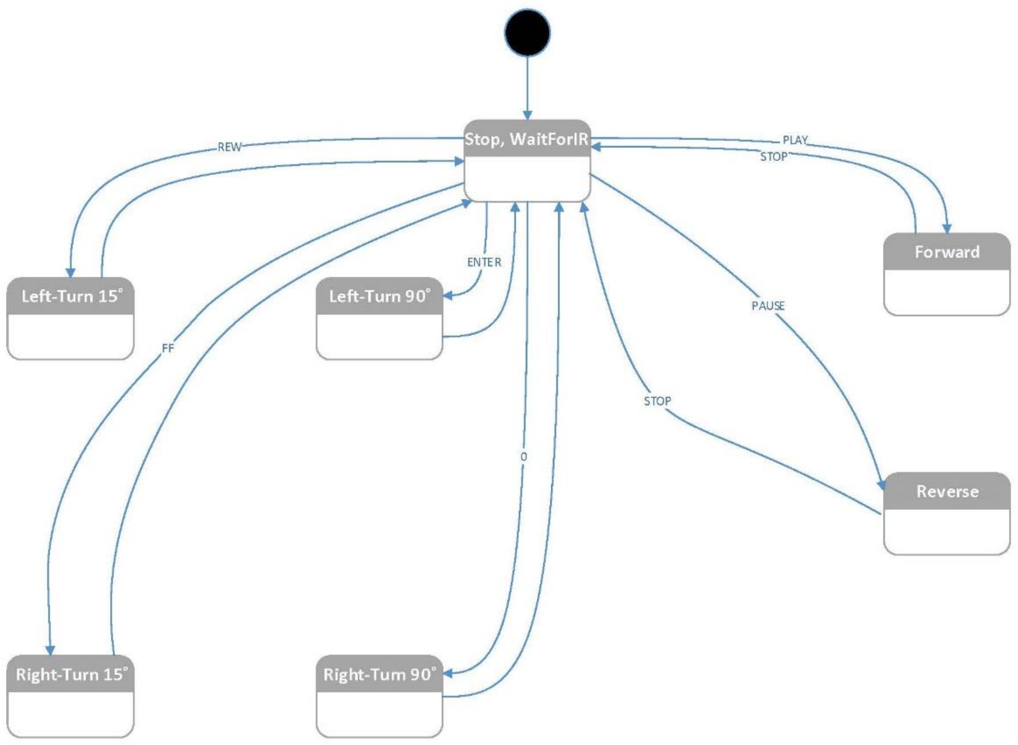

Figure 2. State diagram demonstrating manual controls for the DE2Bot. The robot will wait for an input button in the Stop state. The robot enters the turn states only briefly before returning to the Stop state, while the Forward and Backward states will wait for an input to return to the Stop state.

TABLE 1. BUTTON FUNCTIONALITY

Button	**Function**
PREV CHAN	Parallel park
MUTE	Perpendicular park
1	Park in space 1
2	Park in space 2
3	Park in space 3
4	Park in space 4
5	Park in space 5
6	Park in space 6
7	Park in space 7
0	RT90 (90° CW)
ENTER	LT90 (90° CCW)
REW	LT15 (15° CCW)
PLAY	FW (Forward)
FF	RT15 (10° CCW)
PAUSE	BW (Reverse)
STOP	STOP

For parallel parking, the DE2Bot only needs to move forward before initiating the parking sequence. For the perpendicular parking, the DE2Bot needs to make two turns, a 90 degree right turn, and a 90 degree left turn, before it can approach the perpendicular spots to continue onto the parking sequence. Both parking subroutines will first check with the sonar sensors to see if the space is occupied. After the DE2Bot has reached the parallel parking spot, the bot will be facing the west. If the spot is not occupied, the bot will rotate 90 degrees clockwise in place, move forward into the parking space, and then rotate 90 degrees counterclockwise. We will be able to reuse similar code from the manual parking to streamline the automatic parking, however the parking routine itself will only initiate after the appropriate button is pressed on the remote. This mode of parallel parking takes advantage of the DE2Bot's zero turning radius, as opposed to the more complex, traditional approach to parallel parking. The perpendicular parking subroutine will be like that for parallel parking, with the only difference being that the bot does not have to turn once in the spot.

<u>**Automatic Parking Using Odometer**</u>

There are three different phases of the automatic parking. The first phase is always the same; the DE2Bot must drive forward, turn 90 degrees to the right, drive forward, and turn 90 degrees to the left. The second phase is the varying forward distance the bot must travel, depending on which space the DE2Bot is to park in. The third phase is to call the perpendicular parking subroutine used in manual parking. This sequence is highlighted in Figure 3. The approach we have chosen is to have seven different subroutines, mapped to buttons one through seven on the remote that denote which space to park in, as seen in Table 1. All seven subroutines will be the same except for the forward distance travelled after the first two turns. Figure 3 on the next page outlines the automatic parking routine.

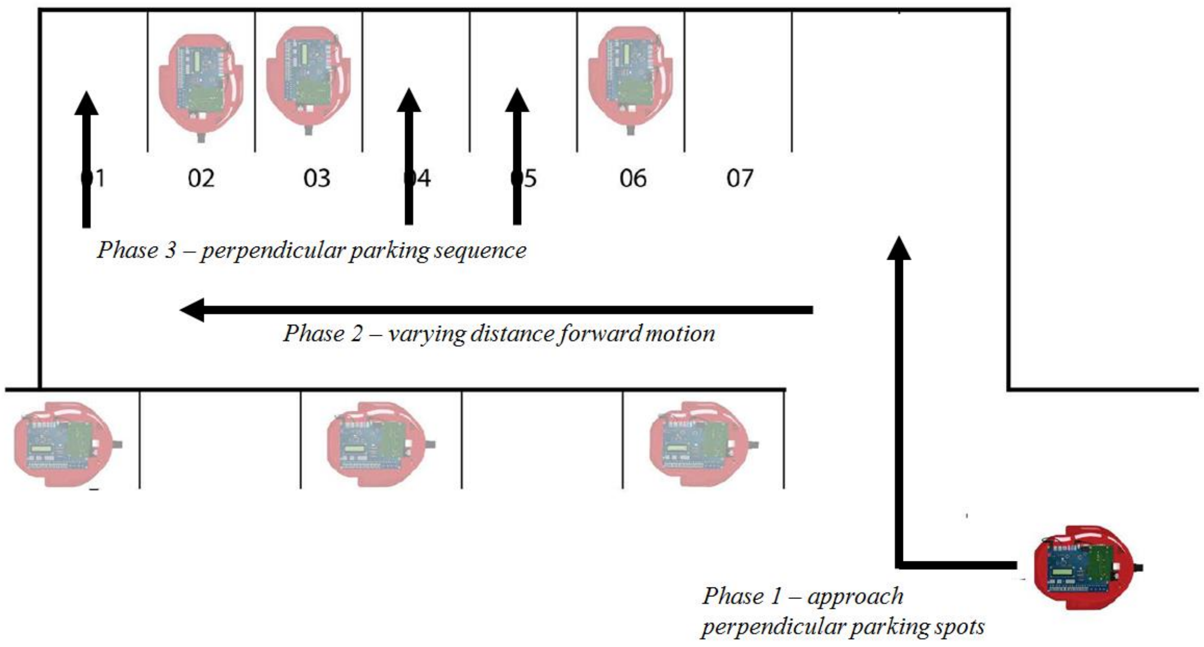
01
02
03
04
05
06
07
Phase 3 – perpendicular parking sequence
Phase 2 – varying distance forward motion
Phase 1 – approach perpendicular parking spots

Figure 3. Parking lot map demonstrating the automatic parking routine. The bot moves forward, turns right 90 degrees, moves forward, turns left 90 degrees, moves a varying distance depending on which spot to park in, and then calls the perpendicular parking subroutine.

Automatic perpendicular parking will be demonstrated with the odometer, rather than sonar, due to the latter's inaccuracy at short distances. The navigation will rely on reading the X position of the robot from its odometer, and comparing the received values to a predetermined distance the robot must cover.

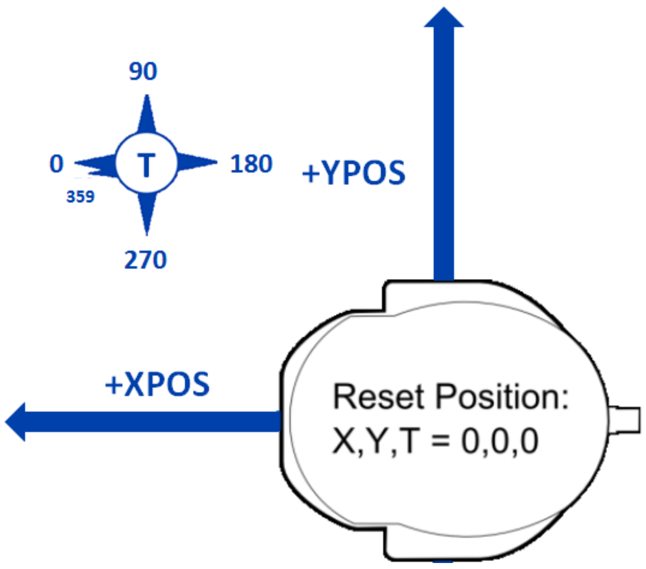

Figure 4. Coordinate system used by the robot's odometer to determine distance travelled.

Only the X position is required, since the robot's odometer will be reset after every movement, and the only motion will be forward. Using Figure 4 as a reference, the robot must move in the positive X direction to begin the automatic parking sequence. The X position is continuously read in from the odometer until the robot reaches the desired position. Additionally, the robot will travel in a straight line within ±5 degrees of its set heading, defined by T for the odometer. The heading is changed to allow the robot to turn 90 degrees clockwise in place. This same method of navigation is used to direct the robot towards the designated perpendicular parking spot, through the sequence of events detailed in the previous paragraph.

Management Plan

The management plan follows the Gantt Chart seen in Appendix A. The project was divided into three major milestones from the introduction of the assignment to the class, namely being the project introduction, function implementation, and testing and execution. These divisions allowed the group to easily shift focus between the key tasks required for completion. The Gantt Chart also displays the subtasks that made up the major milestones; these included brainstorming and logistics to aid groupwork, manual control implementation based on IR remote inputs, implementation of the odometer-based automatic parking sequence, and the sonar interrupt failsafes.

Contingency Plan

Relying on the comparison between staging area measurements and the odometer to determine the remaining distance to a parking spot could lead to errors from wheel slippage or friction, which would cause the automatic parking sequence to be inaccurate. In that case, the team is also prepared to adapt the sonar subroutine to create an alternate routine for automatic perpendicular parking. The sonars will read in the remaining distance to the end wall. This value can be compared to the measurements of the staging area, to relay the distance to be covered to a selected parking spot. In case this is too hard to implement or is inaccurate as well, the measured distance values to be compared with the odometer values can be modified till the sequence is accurate.

Appendix A: Gantt Chart for Project Management Plan

Task	Duration	Start	Finish
Project introduction	15 days	Monday 3/16/2017	Thursday 3/30/2017
Implement initial test functions - forward and backward arcs, in-place turns	1 day	Monday 3/16/2017	Monday 3/16/2017
Create Github repository, Google Drive for central file sharing	1 day	Tuesday 3/17/2017	Tuesday 3/17/2017
Go over DE2Bot Manual, ASM file and flesh out brainstorming ideas	7 days	Monday 3/20/2017	Sunday 3/26/2017
Discuss and decide most effective implementation of parking sequence	4 days	Monday 3/27/2017	Thursday 3/30/2017
Compile list of required functions, IR button codes, test area dimensions	1 day	Thursday 3/30/2017	Thursday 3/30/2017
Function Implementations	15 days	Thursday 3/30/2017	Thursday 4/13/2017
Write pseudocode for required manual and automatic parking functions	4 days	Thursday 3/30/2017	Monday 4/3/2017
Code manual driving controls into ASM file	4 days	Monday 4/3/2017	Thursday 4/6/2017
Code odometry-based automatic parking functions into ASM file	7 days	Thursday 4/6/2017	Thursday 4/13/2017
Debug manual controls	1 day	Thursday 4/6/2017	Thursday 4/6/2017
Code sonar interrupts for parking failsafe	4 days	Monday 4/10/2017	Thursday 4/13/2017
Debug automatic sequence, sonar interrupts	1 day	Thursday 4/13/2017	Thursday 4/13/2017
Testing and Execution	8 days	Thursday 4/13/2017	Thursday 4/20/2017
Test and confirm full functionality of manual and automatic sequences	7 days	Thursday 4/13/2017	Wednesday 4/19/2017
Project final demo	1 day	Thursday 4/20/2017	Thursday 4/20/2017

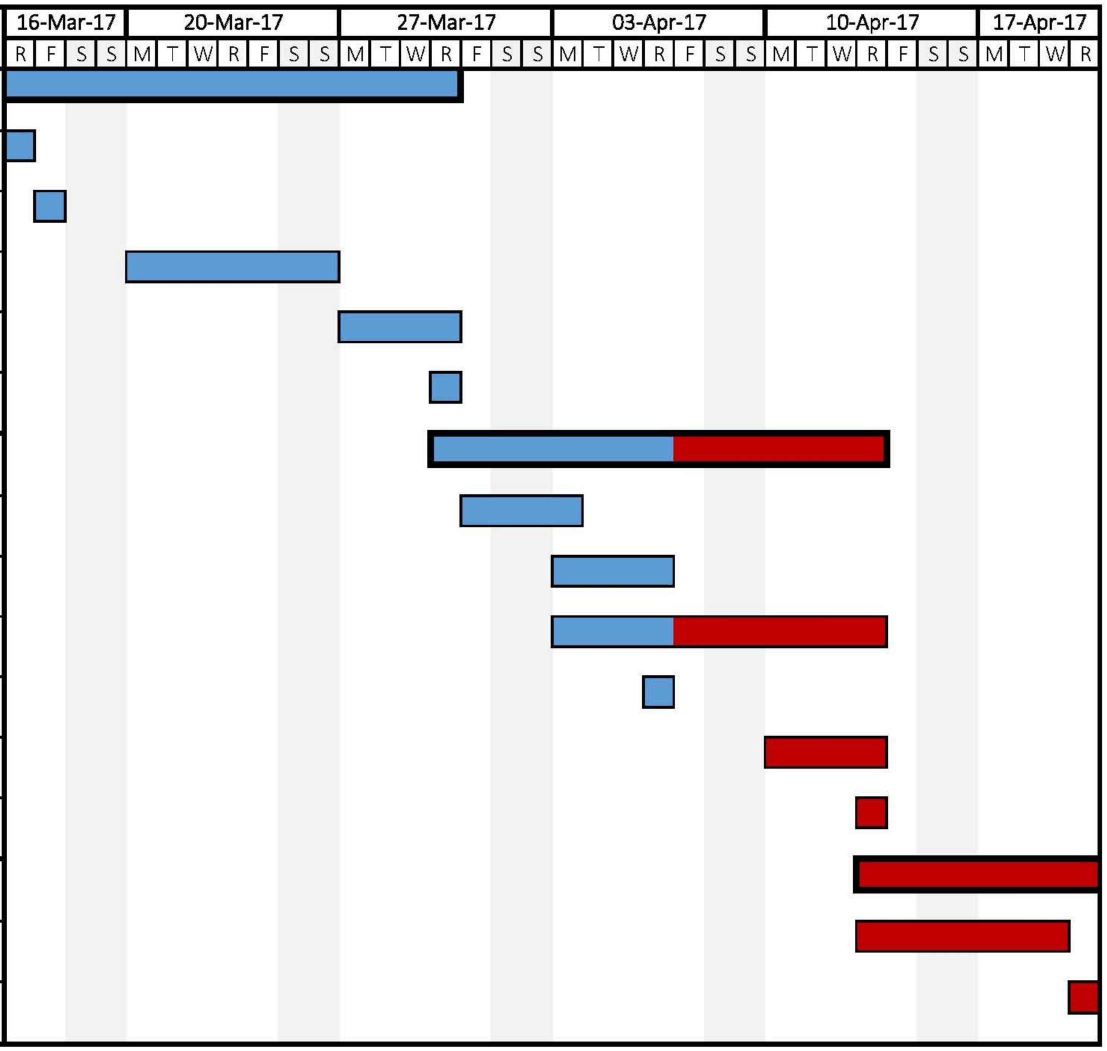

Figure 1. Gantt Chart displaying timeline of major events within the management plan. Currently completed tasks in the timeline are shaded blue, while estimated future tasks are shaded red.