Proposed Solution For a DE2Bot to Park Semi-Autonomously in Perpendicular and Parallel Spaces and Fully Autonomously in Perpendicular Spaces

Jacob Campbell
Alex Cornila
Anina Mu
Moradeke Olumogba

Georgia Institute of Technology
School of Electrical and Computer Engineering

Submitted
April 7, 2017

Executive Summary

This document is a design proposal for the Spring 2017 ECE 2031 final project concerning the autonomous parking of a DE2Bot [1]. The design involves using assembly language to program the robot to respond to remote control commands, park itself in both perpendicular and parallel parking spaces, and move autonomously from a starting point outside a parking lot to fully park itself into any one of seven perpendicular parking spaces [1]. This proposal provides an overview of the types of parking to be implemented, detailed descriptions of the proposed algorithms for the robot to execute autonomous parking and respond to remote control commands, and justifications for selected implementation methods. Furthermore, information regarding the team's management and contingency plans are provided.

Proposed Solution For a DE2Bot to Park Semi-Autonomously in Perpendicular and Parallel Spaces and Fully Autonomously in Perpendicular Spaces

Introduction

Design Problem and Project Requirements

This document describes a solution for three types of autonomous parking to be performed by a DE2bot. Using the SCOMP assembly language, the robot will be programmed to perform semi-autonomous parking maneuvers in individual parallel and perpendicular parking spaces as well as fully autonomous maneuvers in a parking lot of perpendicular spaces [1].

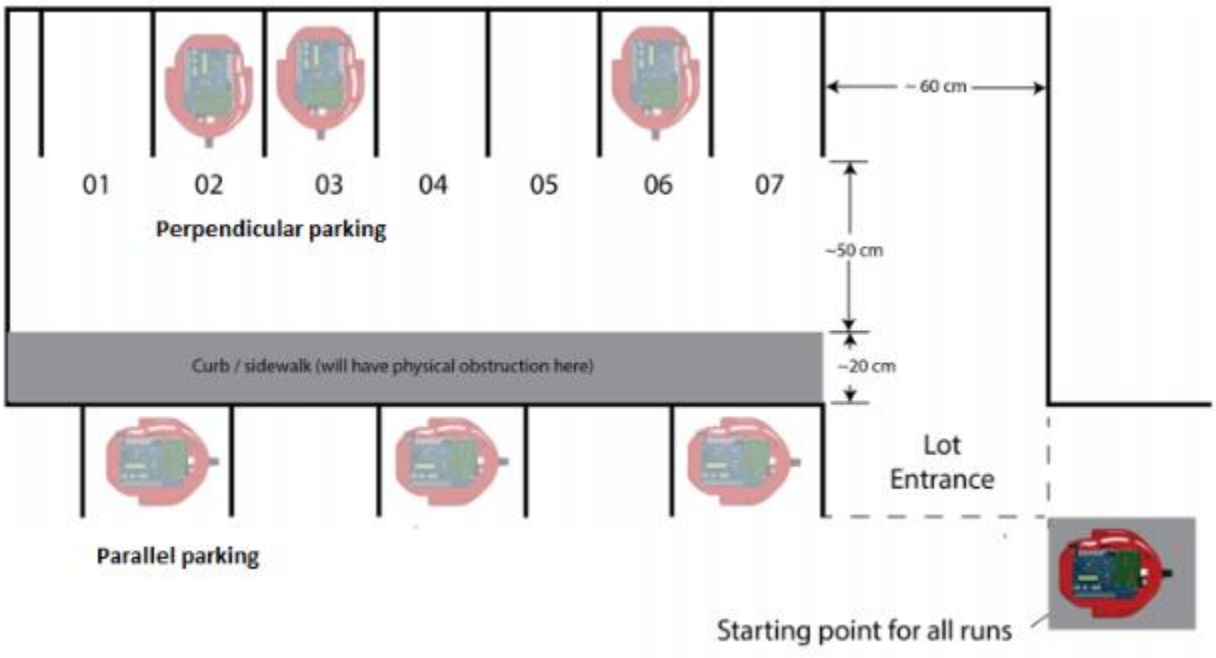


Figure 1. A visual representation of the spaces in which the robot will park seen in [1].

For all types of parking, the robot will park and align itself with the spaces in the orientations indicated by Figure 1, given that the robot in the image is facing forward at its starting point [1].

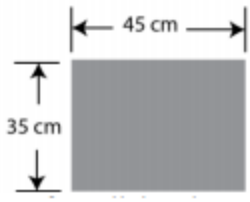


Figure 2. The dimensions the each space shown in the parking lot Figure 1 are 35x45 cm.

Proposed Solution to the Three Types of Parking

The proposed solution implements a remote control allowing the user to determine the movement of the robot. For the semi-autonomous perpendicular and parallel sequences, the remote will first be used to manually move the robot in front of the desired parking space, then park itself at the press of a button before returning control back to the user. The fully autonomous perpendicular parking will consist of the robot moving itself from the starting point and through the lot before parking into a space.

Performance Advantages of the Solution

The proposed implementation is effective and robust because it contains definite algorithms that depend on constant, predefined values such as distance and velocity, enabling the robot to move in a consistent, predictable manner.

Technical Approach

A State Machine to Determine Parking Maneuvers

This implementation features a state machine that enables control of the robot through the IR remote.

```
MControl:
    OUT     RESETPOS
    OUT     IR_LO
    LOADI   0
    STORE   DVel
    STORE   DTheta
    CALL    IRDisp      ; Display the current IR code
    IN      IR_LO
    STORE   RemoteKey
; Determine next state
    SUB     R0
    JZERO   Die         ; 0: die
    LOAD    RemoteKey
    SUB     R2          ; 2: move forward 1 second
    JZERO   MForward
    LOAD    RemoteKey
    SUB     R4          ; 4: turn left 90 degrees
    JZERO   MLeft
    LOAD    RemoteKey
    SUB     R5          ; 5: AutoSelect - choose parking type
    JZERO   AutoSelect
    LOAD    RemoteKey
    SUB     R6          ; 6: turn right 90 degrees
    JZERO   MRight
    LOAD    RemoteKey
    SUB     R8          ; 8: move backward 1 second
    JZERO   MBackward
    JUMP    MControl    ; loop in the control segment waiting for valid input
```

Figure 3. The MControl subroutine contains the functionality of the remote control with specific buttons representing different inputs.

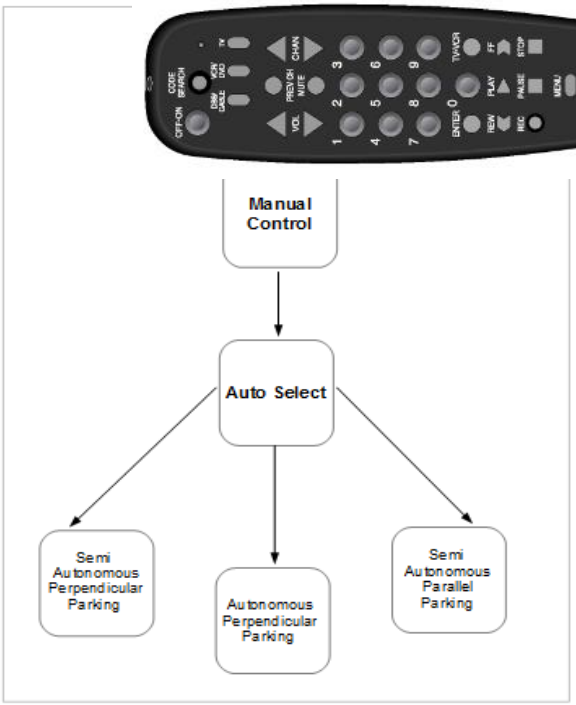
Use of an IR Remote to Enable Human Interaction

When the robot is reset, it is automatically brought to the MControl state. In MControl, the remote's numeric keypad is used as an intuitive controller to manually select the robot's movements. The user may move the robot forward with key 2, backward with key 8, left with key 4, and right with key 6. The forward and backward keys jump to subroutines that cause the robot to move in a straight line forward or backward for one second at a velocity of 33 cm/s. The left and right keys jump to subroutines that rotate the robot 90 degrees counterclockwise or clockwise, respectively. All four return the program to

MControl, allowing the user to retain manual control of the robot until otherwise specified. Additionally, the user may press MUTE on the remote to abort control of the robot and end any ongoing processes.

Figure 4. The IR remote used to process user input and control of the robot [2].

AutoSelect: Determining a Parking Maneuver to Execute




```
AutoSelect:
    IN      IR_LO
    CALL    IRDisp
    STORE   RemoteKey
    OUT     IR_LO
    LOAD    RemoteKey
    SUB     RMUTE
    JZERO   MControl    ;Exit and return to MControl
    LOAD    RemoteKey
    SUB     R1          ; 1:Semi Autonomous perpendicular
    JZERO   Perp
    LOAD    RemoteKey
    SUB     R2          ; 2:Autonomous perp
    JZERO   SetLotMoveTime
    LOAD    RemoteKey
    SUB     R3          ; 3:Semi Autonomous parallel
    JZERO   Parallel
    JUMP    AutoSelect  ; loop until valid input
```

Figure 5. Graphical representation of the AutoSelect state that enables the user to select between the different parking modes and the corresponding assembly code implementation.

In MControl, the user may additionally press key 5 to enter another state called AutoSelect. AutoSelect

reads in user input from the remote to determine which of the three parking maneuvers to perform. Key 1 jumps to the subroutine that performs semi-autonomous perpendicular parking while key 3 jumps to a different subroutine that performs parallel parking, both of which restore user control by returning to MControl upon successful completion of the parking operation. Key 2 indicates that the robot should perform a fully autonomous perpendicular parking job. This brings the program to another state that waits for the user to input a specific parking space number with numeric keys 1-7 corresponding to the seven spaces in the parking lot shown in Figure 1. Once the user has selected a valid space number, the robot will immediately move itself from the starting point to the location and park itself completely.

Semi-Autonomous Perpendicular Parking

Once the robot has been manually moved in front of the parking space in MControl, it will be faced 90 degrees counterclockwise to its final parked position. To begin the autonomous motion, the user will press key 5 to enter the AutoSelect state then key 1 to execute the perpendicular parking subroutine. This process will cause the robot to rotate 90 degrees clockwise to align itself with the space and move forward until reaching a position completely within the space. At this point, the robot will halt and the program will return to MControl.

Semi-Autonomous Parallel Parking

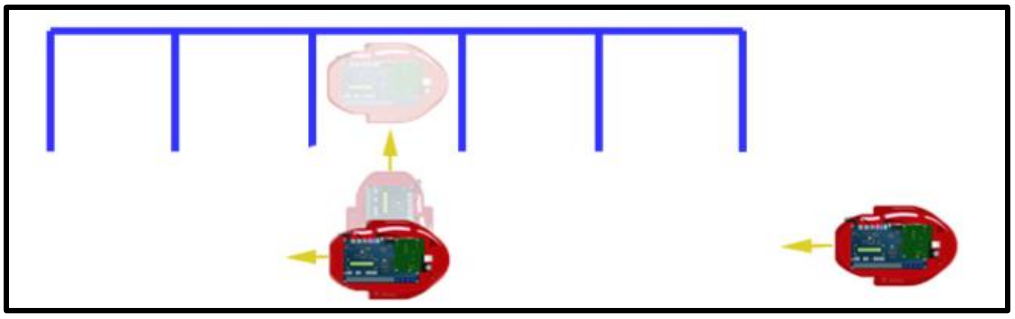


Figure 6. Step-by-step depiction of the parallel parking movement.

Similar to the semi-autonomous perpendicular parking, the robot will be manually driven to the space using the remote control. For this maneuver, however, the robot will initially be faced parallel to its final orientation, thus requiring an extra rotation. Once again, key 5 will enter AutoSelect, and key 3 will execute the parallel parking subroutine. The robot will rotate 90 degrees clockwise and move into the space in the same manner it would move into a perpendicular space. Additionally, the robot will rotate 90 degrees counterclockwise to align itself with the space before it stops and the program returns to MControl. This parallel parking solution proves most convenient given the difficulty to pivot at different angles and the robot's unique capability to rotate in place with ease.

Fully Autonomous Perpendicular Parking

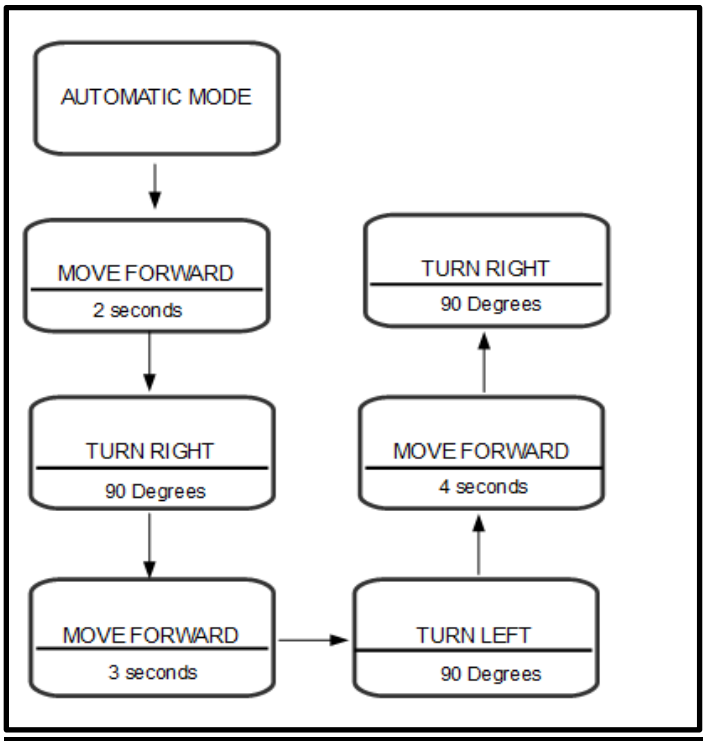


Figure 7. Flow chart representing an autonomous perpendicular parking process to a space with a four

second travel distance from the beginning of the lot.

When the robot is at its stationary starting point, key 5 brings the program to AutoSelect and, then key 2 brings the program to the fully autonomous perpendicular parking state. At this point, another remote key should be pressed to indicate the desired space in which to park. The IR code of the key pressed is immediately translated to a space number and stored as a variable to be used later. First, the robot will move forward, turn 90 degrees clockwise to align with the lot entrance, move forward into the lot, turn 90 degrees counterclockwise to travel to the parking space, and move forward for a specified time interval determined by its speed of 33 cm/s and the travel distance to the indicated space, determined by the width of the spaces and the space number that was earlier stored in memory. Upon reaching the space, the robot will immediately execute the perpendicular parking subroutine described above.

Performance Advantages of this Solution

This solution is effective because it conserves memory, requires no hardware modifications to the existing system, and provides a user-friendly control interface.

Memory is saved by recycling code through the state machine and subroutines. Furthermore, for the fully autonomous sequence, this implementation uses the predetermined values to determine the movement of the robot from the entrance of the lot to the space. The algorithm based on known constants is stronger than one that may be based on unreliable values such as sensor measurements. This requires fewer resources at compile time by avoiding code complexities and less computational power at run time by reducing dependencies on dynamic parameters. Minimal resources are required to successfully use this program, as it only depends on the direction, angle, and speed speed parameters of the bot and the onboard timer.

Another benefit of this solution is that no hardware modifications to SCOMP and the DE2Bot are

required. Taking advantage of a previously tested system promotes robustness and portability, allows swifter debugging during program development, and prevents unexpected runtime errors that may arise due to potentially dangerous third-party additions and modifications to the hardware.

Lastly, this design provides a simple, intuitive user experience. The controls of the remote, as described in the Technical Approach, correspond to arrow keys of standard direction controls, and the different modes expand the number of possible key inputs.

Management Plan

The Gantt chart in Appendix A shows the major tasks to be completed, time estimates for each phase of the project, and an overall project timeline. Initially, it was necessary to devote time to determine the initial starting point of the provided system by writing simple programs with the SCOMP assembly language, reviewing the robot and remote manuals, and examining previously implemented functionalities. The next step was to brainstorm ideas to develop an optimal solution for this particular problem.

The first major task was to develop a robust algorithm for semi-autonomous perpendicular parking and derive the other algorithms from it. The next task was to implement manual controls and integrate the remote with the program. Throughout the development period, considerable time was spent debugging code, identifying weak points, and adjusting the strategies as needed. Additionally, gathering and organizing material for the project proposal remained a priority throughout the process.

Contingency Plan

The straightforward nature of the implementation may be a tradeoff for precision. Limitations may arise

from from the dependencies on predetermined measurements, requiring the robot to be placed at exactly the same starting location before all runs of the fully automatic parking sequence. In addition, imperfect odometry may cause the robot to stray from a straight line of motion when moving over long distances to the farthest parking spaces. Possible alterations to accommodate such situations may be to double check the robot's displacement using its x and y positions, program the robot to pause and reset its orientation while moving, or return to MControl in response to remote key controls.

An issue that may arise for the user could be confusion over the current state of the robot. For example, the user must press three keystrokes to execute a fully automatic parking sequence. To appease this issue, button 0 has been programmed to return to MControl and Mute to enter the Die subroutine that immediately halts the robot. To go one step further, indicators of the robot's current state could be displayed on the LCD screen.

Another problem may arise from errors in the approaching the desired angle from the origin that result from non-uniform motor rotation. The turning angle is not guaranteed to be exact, which can result in an accumulation of errors that could noticeably alter the robot's path from the desired path. To account for this issue, a second movement solution is currently being implemented. This solution will allow the DE2Bot to move across a cartesian coordinate system to a desired (x, y) point. The robot should follow a straight path, which represents the hypotenuse of a triangle formed by the values of the target point in relation to the robot's most recent reset position of (0, 0) [4].

Works Cited

[1] Dr Thomas Collins. ECE 2031 Digital Design Laboratory (2017) *Final Design Project:Self Parking.*
[Online] Available: http://powersof2.gatech.edu/2031/Project/Spring2017Project_SelfParking.pdf
Accessed on March 20th.

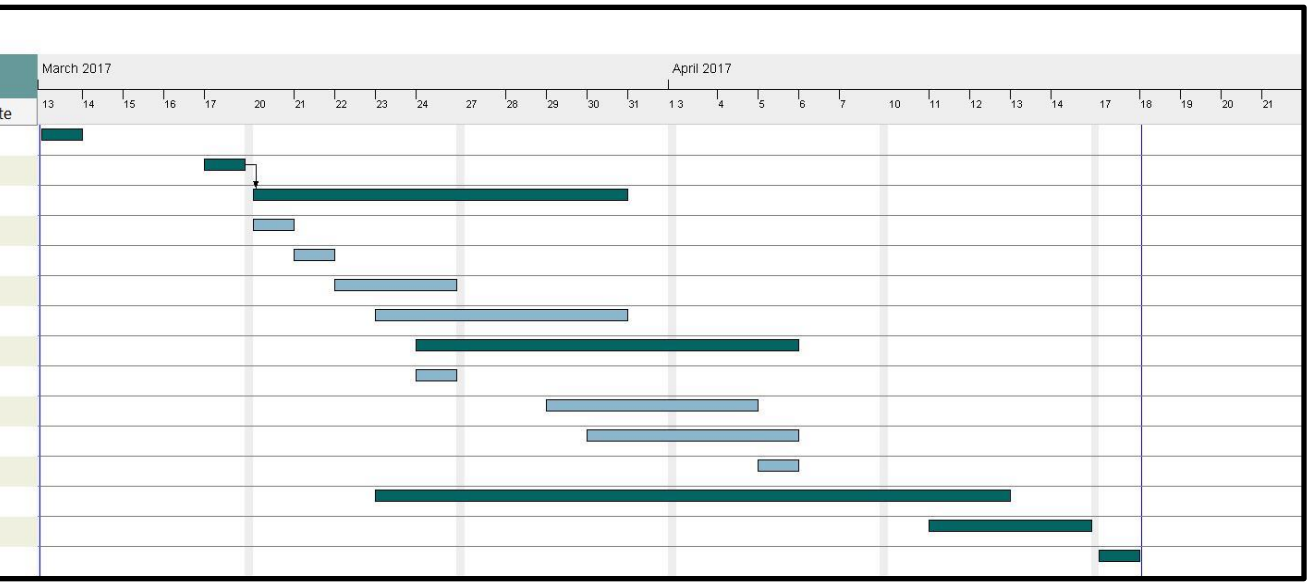
[2] Universal RCA Remote Manual [Online]
Available: http://powersof2.gatech.edu/resources/RCU300T_DOC_OM.pdf
Accessed on March 21st.

[3] DE2Bot User's Manual. [Online]
Available: http://powersof2.gatech.edu/resources/DE2Bot/DE2BotUsersManual_v5.pdf
Accessed on March 21st.

[4] Dr Magnus Egerstedt. Control of Mobile Robots [Online]
Available: https://www.coursera.org/learn/mobile-robot

Appendix A: Gantt Chart of Project Timeline

The Gantt Chart estimates the time frame of individual tasks throughout the project development period and enumerates the sequence and duration of major tasks to be completed.


March 2017
April 2017
13 14 15 16 17 20 21 22 23 24 27 28 29 30 31 1 3 4 5 6 7 10 11 12 13 14 17 18 19 20 21