**Implementation of Semi-Autonomous Parallel Parking, Semi-Autonomous Perpendicular Parking, and Fully-Autonomous Perpendicular Parking for the DE2Bot**

Yotam Ghebre

Kevin Hilinski

Sebastian Jara Garay

Brian Mull

Joey Sterling

Georgia Institute of Technology

School of Electrical and Computer Engineering

Submitted

April 25, 2017

Table of Contents

[Abstract 3](#_Toc480849056)

[1.0 Introduction 4](#_Toc480849057)

[1.1 Problem Description 4](#_Toc480849058)

[1.2 Design Solution 5](#_Toc480849059)

[2.0 General Methodology 6](#_Toc480849060)

[2.1 Infrared Remote Decoding Method 6](#_Toc480849061)

[2.2 Manual Driving Control Design 7](#_Toc480849062)

[2.2.1 Justifications and Notable Features 9](#_Toc480849063)

[2.3 Semi-Autonomous Parking Mode 9](#_Toc480849064)

[2.3.1 Justifications and Notable Features 10](#_Toc480849065)

[2.4 Autonomous Perpendicular Parking Mode 11](#_Toc480849066)

[2.4.1 Justification and Notable Features 12](#_Toc480849067)

[2.5 Management Plan 13](#_Toc480849068)

[3.0 Technical Results 13](#_Toc480849069)

[4.0 Conclusion 14](#_Toc480849070)

[5.0 Works Cited 16](#_Toc480849071)

[Appendix A: Gantt Chart of Project Timeline 1](#_Toc480849072)

[Appendix B: Remote Reading Assembly Code 2](#_Toc480849073)

[Appendix C: Basic Movement Assembly Code 5](#_Toc480849074)

[Appendix D: Semi-Autonomous Parallel and Perpendicular Parking Assembly Code 8](#_Toc480849075)

[Appendix E: Fully-Autonomous Perpendicular Parking Assembly Code 11](#_Toc480849076)

[Appendix F: LogBook 17](#_Toc480849077)

# Abstract

The purpose of this paper is to describe a solution that autonomously and semi-autonomously parks a DE2 robot into parallel and perpendicular parking spaces using a universal infrared remote to control the robot’s movement and mode. An assembly code subroutine was written to decipher the output of the infrared remote. The control layout of the remote was chosen to be intuitive to the operator. An assembly code subroutine was written to manually control the robot, on which all autonomous motion was built. Yet another assembly code subroutine was used to determine which parking maneuver, either parallel or perpendicular, was to be performed. Lastly, special features, such as full space detection, autonomous pause and abort, and micro-movements, were added to the robot to increase safety and usability. During the final demonstration, each mode of parking was successfully shown, indicating all requirements of the initial design problem were met.

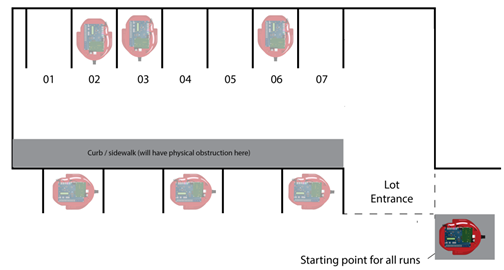
**Implementation of Semi-Autonomous Parallel Parking, Semi-Autonomous Perpendicular Parking, and Fully-Autonomous Perpendicular Parking for the DE2Bot**

# 1.0 Introduction

## Problem Description

The goal of this project was to implement semi-autonomous parallel and perpendicular parking, and fully-autonomous perpendicular parking in the DE2 robot. Specifically, the parameters of the assigned task were:

1. Semi-autonomous parallel parking
   1. The operator manually drives the robot near a parallel parking space (as determined by the design).
   2. The operator presses a button on a remote to signal the robot to park itself in the parking space desired.
2. Semi-autonomous perpendicular parking
   1. The operator manually drives the robot near a perpendicular parking space (as determined by the design).
   2. The operator presses a button on a remote to signal the robot to park itself in the parking space desired.
3. Fully-autonomous perpendicular parking
   1. The robot starts in the starting space outside of the parking lot as shown in Figure 1.
   2. The operator tells the robot in which parking space number to park.
   3. The robot then must move autonomously to that perpendicular parking space.

****

**Figure 1.** Diagram of the parking scenario to be used for the autonomous and semi-autonomous parking. [1]

## 1.2 Design Solution

To solve the design problem, four basic functions were created and implemented on the robot:

* moving forward
* turning left 90 degrees
* turning right 90 degrees
* stopping

All autonomous procedures were conducted using only these functions and the built-in odometer to measure distances travelled and angles turned by the robot. This design plan allowed for higher performance on the required complex maneuvers because it was easier to debug the smaller functions to reduce the amount of error that occurred in the smaller movements. In addressing the design problem requiring manual driving, three additional basic functions were implemented on the robot:

* moving backwards
* turning left 5 degrees
* turning right 5 degrees

The addition of these three basic functions allowed the manual driving to be more accurate and more robust to problems such as overshooting the desired parking space or drifting from the desired path.

This design solution differs from the original proposed solution in that during the autonomous portions of the project, moving backwards is never used, and during the manual driving portions of the project, the 5-degree left and right turns have been added. This solution met the design specifications outlined above, resulting in successful parking jobs (i.e. the robot was completely within the desired parking space) for all three parking scenarios during the demonstration.

# 2.0 General Methodology

This design solution involved writing an assembly file to be run on a simple computer (SCOMP) that was implemented on the DE2Bot. An infrared (IR) remote was used to send signals from the operator to the DE2Bot while the assembly code deciphered the signals and implemented their corresponding functions.

## 2.1 Infrared Remote Decoding Method

For the operator to control the DE2Bot remotely, it was necessary to properly map the infrared remote control in such a way that allowed the use of several buttons. The way this was accomplished was through a loop in the assembly file that determined which button was pressed and then jumped to a corresponding subroutine that performed a task based on that button press. Appendix B shows the subroutine that determines the button pressed on the remote. The subroutine read in the IR signal from the remote, displayed it on the seven-segment display, and then XORd that value with the inverted values of each button signal. The XOR bitwise operator outputs zero when there is an exact match between the read IR code and the various button masks. This allowed a single JZERO to be called for each button and ensured any non-match would fall through. The program then jumps to an execute statement based on which button was pressed.

## 2.2 Manual Driving Control Design

To manually drive the robot, several subroutines were created in the main assembly file to control forward and backward motion, left and right turning, and stopping. These subroutines can be seen in Appendix C. The forward motion (MoveFWD) subroutine is controlled by the “2” button on the IR remote. Once the MoveFWD subroutine is called, the robot moves forward in a straight line until it is told to stop by pressing the “5” button on the remote, which will call the STOP subroutine. Similarly, the backward motion (MoveBWD) subroutine is called by pressing the “8” button on the remote and moves the DE2 bot backward until the STOP button is pressed. A subroutine was also created to display the total displacement from the starting point on the seven segment displays on the DE2 bot which can be used for debugging.



**Figure 2.** Highlight of the control groupings for the remote. Red shows the manual control, green shows the parking type for manual control, and blue shows the automatic mode control. [2]

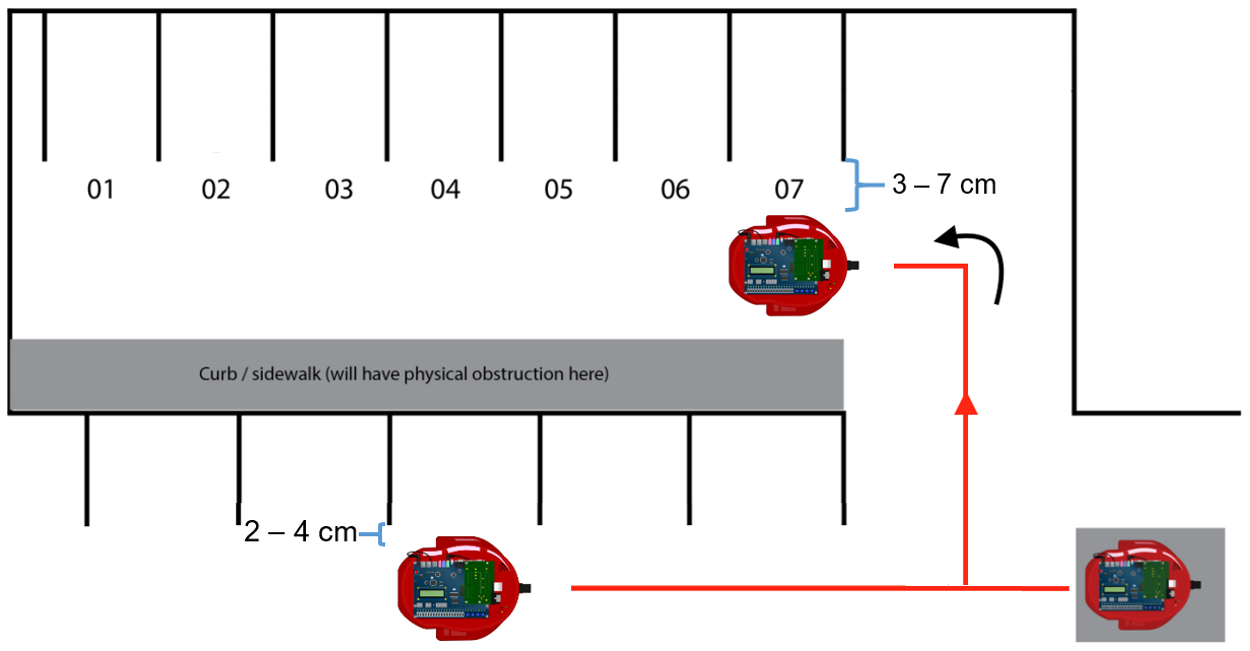
The 90-degree left (TurnLeft) and right (TurnRight) turning subroutines are called by the “4” and “6” buttons respectively on the remote. These subroutines will turn the robot 90 degrees to the left or right before stopping the turn. The micro-turns of 5 degrees left and right are called by the “1” and “3” buttons respectively on the remote. A highlighted picture of the remote can be seen in Figure 2. The red circles show which buttons will be used for control. This configuration reduces confusion and creates an intuitive interface for the operator.

### 2.2.1 Justifications and Notable Features

1. The configuration of buttons for the motor control were chosen to give the operator an intuitive way to control the DE2 bot. As seen in Figure 2 by the red, circled buttons, the main control layout mimics the arrow keys on a keyboard, also mimicking the actual directions and movements that the buttons cause the DE2 bot to execute (i.e. the forward button being on top, the turn left button being on the left, etc.).
2. The micro-turns of 5 degrees left and right were intuitively placed next to their accompanying 90 degree turns, with the 5-degree left turn being on the left side of the IR remote, and the 5-degree right turn being on the right side of the IR remote.
3. The DE2 bot does not execute an arcing turn. If a turn is requested while the DE2 bot is driving, the bot will stop driving before executing the turn. This is quite acceptable because the layout of the parking scenario never requires an arcing turn.

## 2.3 Semi-Autonomous Parking Mode

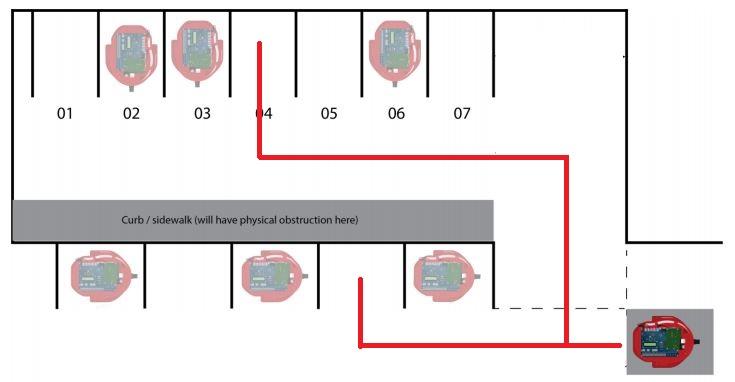
In the semi-autonomous parking mode, the operator drives the DE2 bot to the front and center of the desired parking spot, stops the robot, and then selects whether to perform a parallel or perpendicular parking maneuver. The desired position outside both parallel and perpendicular parking spaces can be seen in Figure 3. Once in the desired position, the “7” button, seen circled in green in Figure 2, will initiate the perpendicular parking where the robot will rotate 90 degrees to the right and then pull straight into the parking spot 47.42 cm. The “9” button, also circled in green in Figure 2, initiates the parallel parking wherein the robot will rotate 90 degrees to the right, pull into the spot for 32.86 cm, and then turn 90 degrees to the left, leaving the DE2 bot in the center of the parking spot. Given that there are no constraints on how the robot must move, this is the simplest way to have the robot enter the space. Also, the dimensions of the robot are such that this maneuver is possible without striking the sides of the parking space. Sample directions can be seen in Figure 4. All the semi-autonomous assembly code can be found in Appendix D for reference.



**Figure 3.** Diagram showing the exact specifications of the desired position outside of both parallel and perpendicular parking spaces, prior to executing the parking maneuver. [1]

### 2.3.1 Justifications and Notable Features

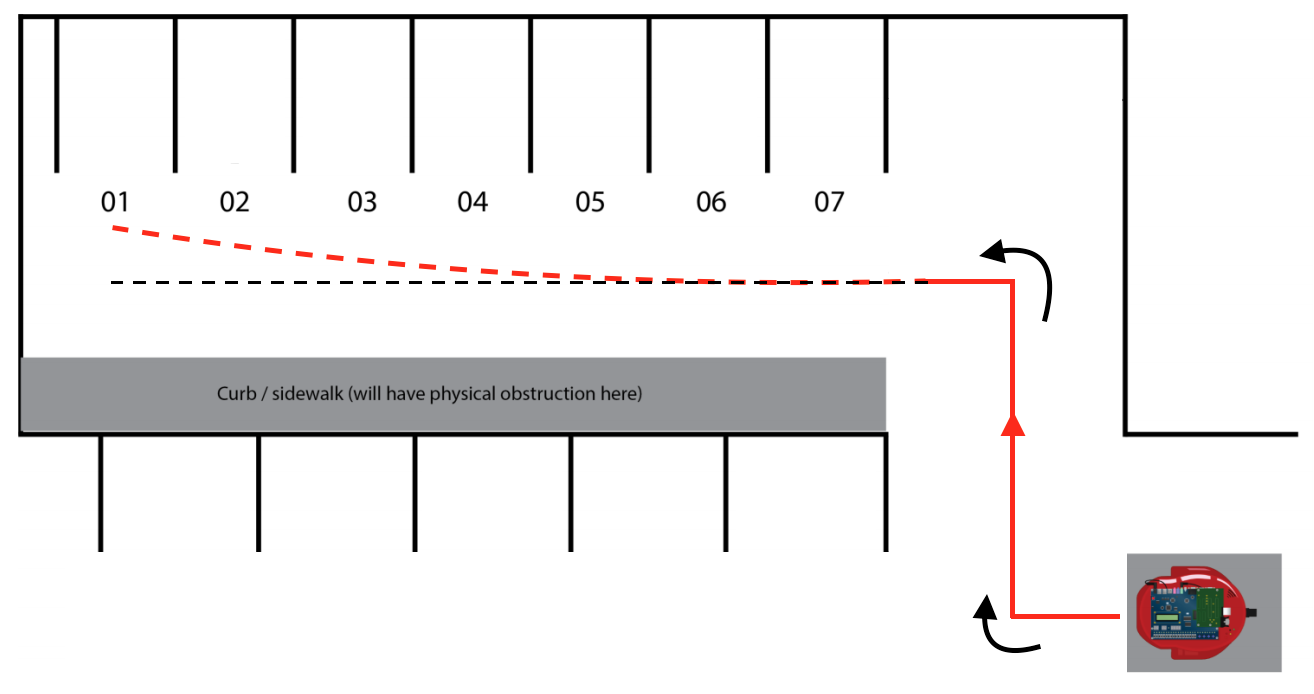
1. The robot will be in the manual driving with semi-autonomous parking mode by default. This is to make the design like a normal car, where when a car is turned on, it is driven as normal until a special parking mode is selected.
2. The distances inside each parking space were measured to determine the appropriate displacement for the DE2 bot to travel inside each parking spot.
3. The DE2 bot’s position is reset each time a parking maneuver is called. This ensures that if the bot is in the desired position shown in Figure 3, the distance measured for the displacement into the parking space measured will be accurate.
4. Before the DE2 bot begins parking itself, it uses sonar to detect if there is already a car in the selected space and will abort the parking maneuver if a car, or another hazard, is detected. This prevents the DE2 bot from crashing if the operator mistakenly tries to select a space that is not empty. This differs from real-world parking assist features in that it checks the availability of the parking space after it has been selected as opposed to checking for availability before offering the spot as a suitable selection.



**Figure 4.** Diagram showing the potential paths of a parallel and perpendicular manual park. [1]

## 2.4 Autonomous Perpendicular Parking Mode

The operator puts the DE2 bot in autonomous perpendicular parking mode by pressing the “0” button, highlighted in blue in Figure 2. Once in autonomous mode, the operator can select a spot in which to park based on which button is pressed (as defined in Figure 4, “1” parks in spot 01, “2” in spot 02, etc.). This method will reduce confusion about which space the operator is directing the robot. The team measured the distances for the first two legs of the displacement. However, an algorithm was constructed that calculated the distance to travel to the directed spot instead of hard-coding each linear distance. When the operator selects a space, the DE2 bot will automatically maneuver to that space and park itself. The parking maneuver can be stopped by pressing the “5” button on the IR remote, which stops the DE2 bot and has it wait for further commands. Pressing “5” again will abort the parking altogether, while pressing “0” will continue the parking from the current spot. All the assembly code for the autonomous parking maneuver can be seen in Appendix E



**Figure 5.** Drift (red) experienced away from desired course heading (black) as bot moves to decreasing spot numbers. [1]

### 2.4.1 Justification and Notable Features

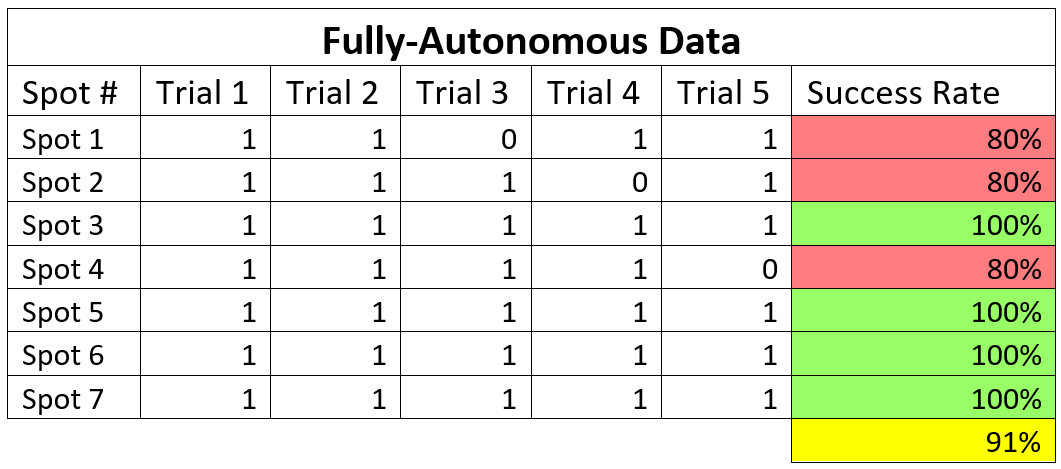
1. The “5” button is used for pausing the autonomous parking maneuver to keep the theme of an intuitive button layout. The “5” button has previously been established as the DE2 bot’s brake, so pressing it to stop the maneuver seems natural to the operator. The “0” button has been established as the autonomous parking button, so pressing “0” to continue parking autonomously also seems natural to the operator.
2. Spot 1 had its own distance to drive into the parking space to account for error due to drifting as shown in Figure 5. There was also a distance set for spots 2, 3, and 4, along with a third distance for spots 5, 6, and 7. These distances yielded the best results for their parking spaces.
3. Error from drifting has been reduced by saving the DE2 bot’s desired course heading (i.e. 0 degrees, 90 degrees, etc.), and then continuously checking, comparing, and correcting the bot’s heading in an infinite loop as it drives.
4. As in the semi-autonomous parking modes, the DE2 bot uses sonar when it is outside of the desired parking space to ensure the operator has not selected an occupied space.

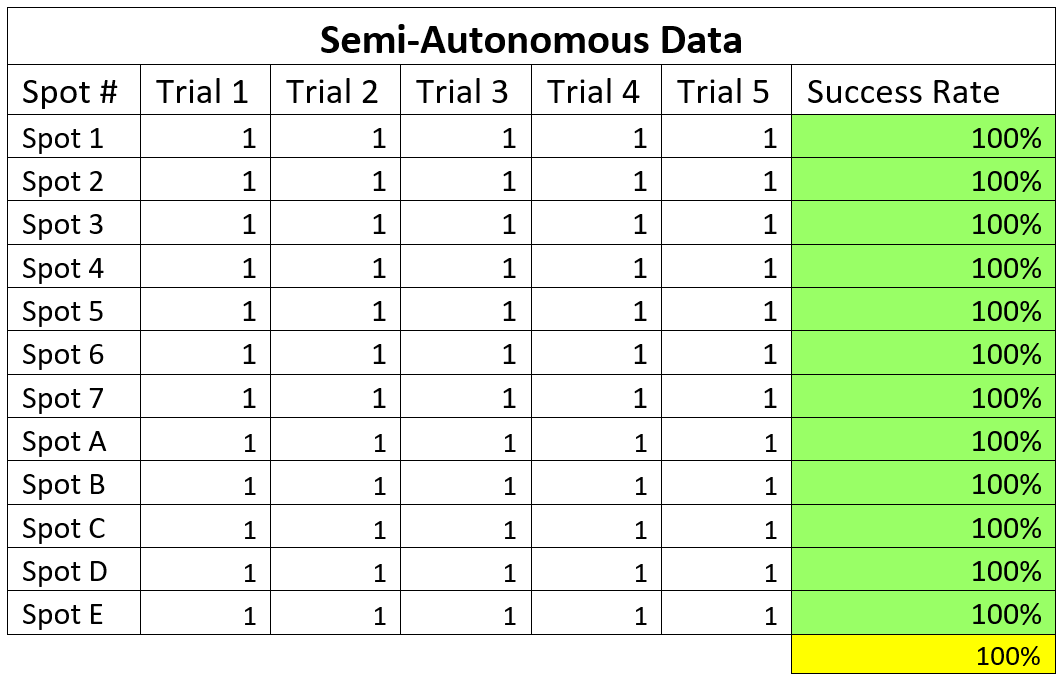
# 2.5 Management Plan

The management plan for this project can be seen in a Gantt chart in Appendix A at the end of this report. It should be noted that the development of the IR remote reading system was of highest priority and was completed first. The manual parallel and perpendicular and autonomous perpendicular parking modes were developed concurrently.

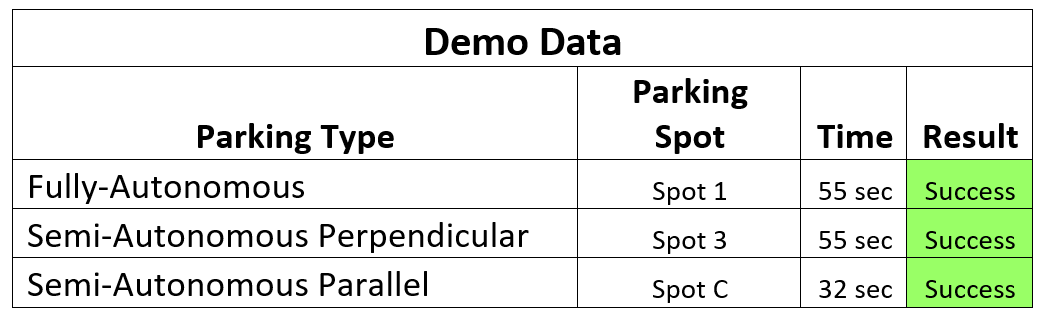
# 3.0 Technical Results

Prior to the demonstration, the fully-autonomous perpendicular parking feature was tested through a series of 35 trials (5 trials for each parking space). As shown in Table 1, this solution to the design problem was very successful in the testing phase, having a success rate of 91% with the results being more successful as the parking spaces became closer to the starting position. The semi-autonomous features were also tested 5 times per parking space, and these trials yielded a 100% success rate due to the reset position technique discussed in section 2.3.1. These results are depicted in Table 2.

**Table 1.** Trial runs for fully autonomous parking

**Table 2.** Trial runs of semi-autonomous parking

The results from the demonstration supported the data gathered in the pre-demonstration testing. All three parking scenarios were successfully executed (i.e. the DE2 bot was entirely within the lines of the parking space), despite the fully-autonomous parking space being spot 1, one of the spots with the lowest success rate during the pre-demonstration trials. These results are in Table 3.

**Table 3.** Results of in class demonstration

# 4.0 Conclusion

The results of the final demonstration displayed a successful parking maneuver of the DE2 robot for both semi-autonomous perpendicular and parallel parking and for the autonomous perpendicular parking.  The four simple motions, on which all the instructions are built, increased the ease with which the maneuvers were debugged. This simple coding method will continue to future iterations of this project to allow for more complex parking scenarios and maneuvers. The addition of the empty space detection and the micro-movements increased the safety and usability of the parking modes by removing the possibility of autonomously parking in a non-empty space and adding additional error control to the manual driving. Also, the autonomous pause and abort feature increases safety should moving obstacles be introduced to the parking environment.

Though this approach was successful, it is possible to correct for the errors experienced with the lower numbered parking spots. For future work, it is recommended to use sonar as opposed to hardcoded special distances for each parking spot. The use of sonar would eliminate any discrepancies caused by uncontrollable drift due to wheel wear, carpet slippage, or improper semi-autonomous alignment. Similarly, sonar could be used to prevent an operator from running into a wall or another robot while manually driving. This could be done by having the sonar override the IR input if it detects an obstacle.

# 5.0 Works Cited

[1] Dr. Tom Collins. *Final Design Project: Self Parking*. [Online] Available: http://powersof2.gatech.edu /2031/Project/Spring2017Project\_SelfParking.pdf

[2] *RCA Universal Remote Manual* [Online] Avaiable: [http://powersof2.gatech.edu/resource /RCU300T\_DOC\_OM.pdf](http://powersof2.gatech.edu/resource%20/RCU300T_DOC_OM.pdf)

# Appendix A: Gantt Chart of Project Timeline

The following Gantt chart shows the final timeline and completion periods for the project. The period time is in days. Each item was completed on time or early.

# Appendix B: Remote Reading Assembly Code

ReadRemote:

   CALL    UpdatePosition

   CALL    DisplayPosition

   IN      IR\_LO

   STORE   ButtonPressed

   XOR     Remote1

   JZERO   Execute1            ; 1 is pressed

   LOAD    ButtonPressed

   XOR     Remote2

   JZERO   Execute2            ; 2 is pressed

   LOAD    ButtonPressed

   XOR     Remote3

   JZERO   Execute3            ; 3 is pressed

   LOAD    ButtonPressed

   XOR     Remote4

   JZERO   Execute4            ; 4 is pressed

   LOAD    ButtonPressed

   XOR     Remote5

   JZERO   Execute5            ; 5 is pressed

   LOAD    ButtonPressed

   XOR     Remote6

   JZERO   Execute6            ; 6 is pressed

   LOAD    ButtonPressed

   XOR     Remote7

   JZERO   Execute7            ; 7 is pressed

   LOAD    ButtonPressed

   XOR     Remote8

   JZERO   Execute8            ; 8 is pressed

   LOAD    ButtonPressed

   XOR     Remote9

   JZERO   Execute9            ; 9 is pressed

   LOAD    ButtonPressed

   XOR     Remote0

   JZERO   Execute0            ; 0 is pressed

   JUMP    ReadRemote          ; jump to return if no button pressed

Execute1:

   CALL    SmallLeft

   JUMP    EndReadRemote

Execute2:

   CALL    MoveFWD

   JUMP    EndReadRemote

Execute3:

   CALL    SmallRight

   JUMP    EndReadRemote

Execute4:

   CALL    TurnLeft

   JUMP    EndReadRemote

Execute5:

   CALL    Stop

   JUMP    EndReadRemote

Execute6:

   CALL    TurnRight

   JUMP    EndReadRemote

Execute7:

   CALL    CheckPerp

   JUMP    EndReadRemote

Execute8:

   CALL    MoveBWD

   JUMP    EndReadRemote

Execute9:

   CALL    CheckPara

   JUMP    EndReadRemote

Execute0:

   CALL    AutomaticParking

   JUMP    EndReadRemote

EndReadRemote:

   OUT     IR\_LO

   RETURN

Remote1:        DW  &H20DF

Remote2:        DW  &HA05F

Remote3:        DW  &H609F

Remote4:        DW  &HE01F

Remote5:        DW  &H30CF

Remote6:        DW  &HB04F

Remote7:        DW  &H708F

Remote8:        DW  &HF00F

Remote9:        DW  &H38C7

Remote0:        DW  &HB847

ButtonPressed:  DW  0

# Appendix C: Basic Movement Assembly Code

SmallLeft:

   CALL    Stop

   IN      THETA    ; taking in the current angle of the robot

   ADDI    5       ; we desire to turn 90 degrees to the left

   CALL    Mod360   ; mod'ing the angle by 360 to get right theta

   STORE   CurrentAngle  ; we want the angle to change to this angle

   CALL    TestAngle

   RETURN

SmallRight:

   CALL    Stop

   IN      THETA   ; taking in the current angle of the robot

   ADDI    -5      ; we desire to turn 90 degrees to the left

   CALL    Mod360  ; mod'ing the angle by 360 to get right theta

   STORE   CurrentAngle  ; we want the angle to change to this angle

   CALL    TestAngle

   RETURN

TurnLeft:

   CALL    Stop

   IN      THETA   ; taking in the current angle of the robot

   ADDI    90      ; we desire to turn 90 degrees to the left

   CALL    Mod360  ; mod'ing the angle by 360 to get right theta

   STORE   CurrentAngle  ; we want the angle to change to this angle

   CALL    TestAngle

   RETURN

TurnRight:

   CALL    Stop

   IN      THETA   ; taking in the current angle of the robot

   ADDI    -90     ; we desire to turn 90 degrees to the right

   CALL    Mod360  ; mod'ing the angle by 360 to get right theta

   STORE   CurrentAngle  ; we want the angle to change to this angle

   CALL    TestAngle

   RETURN

TestAngle:

   ; wait for robot to turn completely to the desired angle

   CALL UpdatePosition

   CALL    GetThetaErr

   CALL    Abs

   JPOS    TestAngle

   RETURN

MoveFWD:

   LOAD    FSlow     ;load in FWD

   STORE   WheelSpeed

   IN THETA

   STORE CurrentAngle

   RETURN

MoveBWD:

   LOAD    RSlow        ;load in BWD

   STORE   WheelSpeed

   IN THETA

   STORE CurrentAngle

   RETURN

Stop:

   LOADI   0

   STORE   WheelSpeed

   IN THETA

   STORE DTheta

CheckStop:

   CALL UpdatePosition

   IN      LVel

   STORE   LeftVelocity

   IN      RVel

   ADD     LeftVelocity

   JZERO   EndStop

   JUMP    CheckStop

   LeftVelocity:   DW  0

EndStop:

   RETURN

WheelSpeed:      DW   0

CurrentAngle: DW 0

UpdatePosition:

   LOAD    WheelSpeed

   STORE DVel

   LOAD CurrentAngle

   STORE DTheta

   CALL ControlMovement

   RETURN

# Appendix D: Semi-Autonomous Parallel and Perpendicular Parking Assembly Code

CheckPara:

LOADI &B00100000

OUT SONAREN

CALL Wait1

CheckSpotPara:

IN DIST5

OUT SSEG1

IN DIST5

ADDI -250

JNEG SpotTaken

LOADI &B00000000

OUT SONAREN

AutoParallel:

OUT      RESETPOS

CALL     TurnRight

    CALL     MoveFWD

CheckParallelDist:

   CALL     UpdatePosition

   CALL     DisplayPosition

   IN       YPos         ; read yPos

   ADDI 316

   JNEG     FinishParallel

   JUMP     CheckParallelDist

FinishParallel:

   CALL TurnLeft

   RETURN

CheckPerp:

   LOADI &B00100000

   OUT SONAREN

   CALL Wait1

CheckSpotPerp:

   IN DIST5

   OUT SSEG1

   IN DIST5

   ADDI -300

   JNEG SpotTaken

   LOADI &B00000000

   OUT SONAREN

TurnPerp:

   OUT RESETPOS

   CALL     Stop

   CALL     TurnRight

   CALL     MoveFWD

CheckPerpDist:

   CALL     UpdatePosition

   CALL     DisplayPosition

   IN YPos

   ADDI 456

   JNEG     FinishPerp

   JUMP     CheckPerpDist

FinishPerp:

   CALL Stop

   RETURN

# Appendix E: Fully-Autonomous Perpendicular Parking Assembly Code

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Automatic Perpendicular Parking Algorithm

;       Completely autonomous parking from start to finish

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AutomaticParking:

   OUT     RESETPOS ; reset the position

   IN      IR\_LO

   STORE   SpotSelected

   XOR     Remote1

   JZERO   Spot1           ; 1 is pressed

   LOAD    SpotSelected

   XOR     Remote2

   JZERO   Spot2           ; 2 is pressed

   LOAD    SpotSelected

   XOR     Remote3

   JZERO   Spot3           ; 3 is pressed

   LOAD    SpotSelected

   XOR     Remote4

   JZERO   Spot4           ; 4 is pressed

   LOAD    SpotSelected

   XOR     Remote5

   JZERO   Spot5           ; 5 is pressed

   LOAD    SpotSelected

   XOR     Remote6

   JZERO   Spot6           ; 6 is pressed

   LOAD    SpotSelected

   XOR     Remote7

   JZERO   Spot7           ; 7 is pressed

   JUMP    AutomaticParking

   SpotSelected:   DW  0

Spot1:

   LOAD    One

   STORE   SpotSelected    ; storing value of 1 into SpotSelected

   JUMP    AutomaticManeuver

Spot2:

   LOAD    Two

   STORE   SpotSelected    ; storing value of 2 into SpotSelected

   JUMP    AutomaticManeuver

Spot3:

   LOAD    Three

   STORE   SpotSelected    ; storing value of 3 into SpotSelected

   JUMP    AutomaticManeuver

Spot4:

   LOAD    Four

   STORE   SpotSelected    ; storing value of 4 into SpotSelected

   JUMP    AutomaticManeuver

Spot5:

   LOAD    Five

   STORE   SpotSelected    ; storing value of 5 into SpotSelected

   JUMP    AutomaticManeuver

Spot6:

   LOAD    Six

   STORE   SpotSelected    ; storing value of 6 into SpotSelected

   JUMP    AutomaticManeuver

Spot7:

   LOAD    Seven

   STORE   SpotSelected    ; storing value of 7 into SpotSelected

   JUMP    AutomaticManeuver

AutomaticManeuver:

   OUT IR\_LO

   LOAD    SpotSelected

   OUT     LCD         ; showing spot selected on the LCD screen

   OUT     RESETPOS    ; resetting the position of the robot

   CALL    MoveFWD     ; moving forward

CheckTurn1:

   CALL CheckEmergencyStop

   LOAD AbortMission

   JPOS Finished

   CALL    UpdatePosition

   CALL    DisplayPosition

   IN      XPos

   SUB     FirstStraight ; checking to see if it has gone 55 cm yet

   JPOS    Turn1        ; if we have travelled 55cm, ready to turn

   JUMP    CheckTurn1   ; else, check the distance travelled

Turn1:

   CALL    Stop        ; stopping the robot

   CALL    TurnRight   ; turning the robot 90 degrees to the right

   CALL    MoveFWD     ; moving forward

CheckTurn2:

   CALL CheckEmergencyStop

   LOAD AbortMission

   JPOS Finished

   CALL    UpdatePosition

   CALL    DisplayPosition

   IN      YPos         ; reading in the position of the robot

   ADD     SecondStraight ; checking to see if it went 105 cm yet

   JNEG    Turn2        ; if we have travelled 1010cm,ready to turn

   JUMP    CheckTurn2  ; else, keep checking the distance travelled

Turn2:

   CALL    Stop        ; stopping the robot

   CALL    TurnLeft    ; turning the robot 90 degrees to the left

   CALL    MoveFWD     ; moving forward

   LOADI   7

   SUB     SpotSelected

   OUT     SSEG2       ; display on sseg1

   STORE   m16sA

   LOADI   366         ; width of each parking spot

   STORE   m16sB

   CALL    Mult16s

CheckStopPoint:

   CALL CheckEmergencyStop

   LOAD AbortMission

   JPOS Finished

   CALL    UpdatePosition

   CALL    DisplayPosition

   IN      XPos        ; reading in the position of the robot

   SUB     mres16sL

   SUB     FirstStraight

   SUB     FirstStraight

   JPOS    CheckSpot

   JUMP    CheckStopPoint

CheckSpot:

   CALL Stop

   LOADI &B00100000

   OUT SONAREN

   CALL Wait1

CheckSpotContinue:

   IN DIST5

   OUT SSEG1

   IN DIST5

   ADDI -390

   JNEG SpotTaken

   LOADI &B00000000

   OUT SONAREN

Turn3:

   CALL    TurnRight

   CALL    MoveFWD

CheckFinalStop:

   CALL CheckEmergencyStop

   LOAD AbortMission

   JPOS Finished

   CALL    UpdatePosition

   CALL    DisplayPosition

   LOAD    SpotSelected

   ADDI   -1

   JZERO   Specail1

   LOAD SpotSelected

   ADDI   -4

   JPOS   Special567

   IN      YPos        ; read xPos

   ADD     SecondStraight

   ADDI    456        ; distance bot should drive into each spot

   JNEG    Finished

   JUMP    CheckFinalStop

Specail1:

   IN      YPos        ; read xPos

   ADD     SecondStraight

   ADDI    426        ; distance bot should drive into each spot

   JNEG    Finished

   JUMP    CheckFinalStop

Special567:

   IN      YPos        ; read xPos

   ADD     SecondStraight

   ADDI    490        ; distance bot should drive into each spot

   JNEG    Finished

   JUMP    CheckFinalStop

Finished:

   CALL    Stop

   RETURN

FirstStraight:  DW  &H1EC

SecondStraight: DW  &H3AA

CheckEmergencyStop:

   LOADI 0

   STORE AbortMission

   IN      IR\_LO

   XOR     Remote5

   JZERO   EmergencyStop ; 5 is pressed

   RETURN ; no emergency stop

EmergencyStop:

   OUT IR\_LO

   LOAD WheelSpeed

   STORE BeforeStopSpeed

   LOAD CurrentAngle

   STORE BeforeStopAngle

   CALL Stop

   BeforeStopSpeed: DW 0

   BeforeStopAngle: DW 0

EmergencyStop2:

   IN      IR\_LO

   STORE AbortButtonPressed

   XOR     Remote5

   JZERO   Abort            ; 5 is pressed

   LOAD AbortButtonPressed

   XOR Remote0

   JZERO Continue ; 0 is pressed

   JUMP EmergencyStop2

   AbortButtonPressed: DW 0

Abort:

   LOADI 1

   STORE AbortMission

   RETURN

Continue:

   LOAD BeforeStopSpeed

   STORE WheelSpeed

   LOAD BeforeStopAngle

   STORE CurrentAngle

   RETURN

SpotTaken:

   LOADI &B00000000

   OUT SONAREN

   JUMP Stop

AbortMission: DW 0

# Appendix F: LogBook