Acknowledgement

There are many people whom I would like to acknowledge for their assistance and support in completing this work, both technical and moral.

First, I would like to thank my advisor Assis. Prof. Dr. Mürvet Kırcı Üçer, for her patient guidance and encouragement during the course of my graduate studies.

I would like to thank Ulaş Alaca and Cafer Koçkan for their support.

This thesis is dedicated to my parents, my sister and Ceylan for their endless support and encouragement.

Sinan ÖNCÜ
May 2005

Abstract

An automated parallel parking strategy for a car-like mobile robot is presented. The study considers general cases of parallel parking for a rectangular robot within a rectangular space. The system works in three phases.

In scanning phase, the environment is scanned by infrared sensors mounted on the robot and a parking position and maneveuring path is generated if sufficient parking space is detected. Then in the starting phase the robot moves to an approciate position which is a suitable place for beginning parking maneveurs. Finally in maneveuring phase the robot performs a series of maneveurs that result in the desired parking location.

Motion characteristics of this kind of robots are modeled, taking into account the nonholonomic constraints acting on the car-like robot.s On the basis of the characteristics, a collision free path is planned dynamically in reference to the surroundings.

The approach has been integrated into an automated parking system, and implemented in a modified car. The system is developed for an automated parking device to help vehicle drivers. The system is capable of safe parking in tight situations. It shows the potential to be integrated into automobiles.

CHAPTER 1

INTRODUCTION

The project presents the study of the parallel parking algorithm of an autonomous guided vehicle (AGV). The project aims developing an autonomous parking aid system that can be integrated on car-like vehicles.

1.1 Problem Definition : Parallel Parking

Parallel parking is a method of parking a vehicle in parallel to other parked cars. Cars parked in parallel are in one line, parallel to the curb, with the front bumper of each car facing the back bumper of an adjacent one.

Parallel parking is a challenging problem for human drivers, especially for beginners. Therefore this type of problem attracts great deal of attention from automobile manufacturers.

In this thesis implementation of parallel parking algorithm on a car-like robot will be presented. The parallel parking involves many problems such as recognition of driving circumstances, maneveuring and vehicle control. The system works in three phase. In the scanning phase the required parking space is scanned by infra-red sensors. It then goes to next phase, the starting phase where the car is moved to a suitable location where the parking maneveurs can begin. The required maneveurs for the parking phase are calculated dynamically by the MCU and the robot follows the path to the desires parking position.

1.2 Organization Of The Thesis

This thesis is divided into 5 sections. The present chapter introduces the scope of this thesis and gives a general description of the problem.

In Chapter 2 background materials for the research areas related to the topic of this thesis are explained. These include kinamatic modelling of the car-like vehicle, tracking control and path planning algorithms.

In Chapter 3 parallel parking algorithm and implementation on the car-like vehicle is discussed in detail.

Chapter 4 is devoted to the experimental studies on the subject including simulator program and hardware implementation. The modules that form the system and parts included in these madules are introduced.

In Chapter 5 results of the experimental studies and future work related to the problem are given.

CHAPTER 2

THEORY

This chapter provides the background materials for the main research areas related to the topic of this thesis: modelling of physical systems, kinematic model of a car-like vehicle, tracking control and path planning algorithms. First holonomic and non-holonomic system concepts are introduced. Differences between these two models are discussed. Non-holonomic constraints are the main focus of the discussion because this model best describes the problem at hand. Based on this discussion kinematic model of a car-like vehicle is introduced. Then, various tracking methods and their implementation on the current problem is described.

2.1 HOLONOMIC AND NONHOLONOMIC CONSTRAINTS

A mechanical system often consists of different components. In the case where the deformations of the body are insignificant compared to the motion of the body as a whole the body is called a rigid body. This idealization allows us to reduce the equations needed to describe a rigid body. [1]

In a mechnical system, rigid body components are usually connected to each other and there are various constraints on them. Because of these constraints, the Cartesian coordinates are not independent. There are various forms of constraints. Geometric constraints are such constraints in which restrictions are imposed on the possible geometrical positions of the individual parts of a system.

Let us consider a moving robot with geometrical position determined by n quantities q(1), q(2), ..., q(n) which represent the general coordinates. A certain variation of these generalized coordinates corresponds to a certain motion of the unconstrained system. If constraints are imposed on this system the variations of arbitrary coordinates will no longer correspond to the motion of the system as a whole. The variations of the generalized coordinates must now satisfy a number of conditions.

These constraints are geometric if they can be expressed in the form:

$$G(q(1), ..., q(n), t) = 0 \quad (2.1)$$

And they are said to be kinematic if the corresponding equations have the form:

$$G(q(1), ..., q(m), q'(1),....,q'(n), t) = 0 \quad (2.2)$$

If the corresponding system of differantial equations q'(n) is integrable, the kinematic constraints are also integrable and are usually geometric constraints. In the opposite case, they are non-integrable and they can not be reduced into geometric constraints. A mechanical system with non-integrable kinematic constraints that can not be reduced into geometric constraints is called a nonholonomic system.

In practical applications the main difference between a holonomic and nonholonomic system can be observed by reversing the motion operated on the system.In a holonomic system like a robot arm initial and final locations will be the same when a motion is applied and then reversed in the same order.

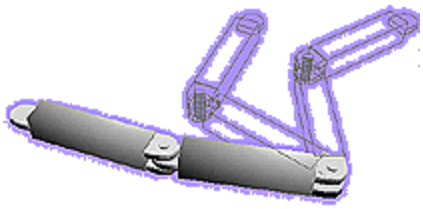


Figure 2.1

In a nonholonomic sytstem like a car-like robot initial and final positions will not be tthe same when a motion is performed and then reversed in the same order.

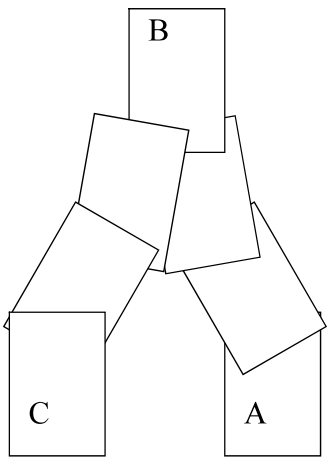


Figure 2.2

The sequence of maneveurs shown in figure 2.2 which are:
From initial location A:

1) Steer left
2) Move Forward
3) Steer Right
4) Move Forward results in location B

Performing the maneveur sequence in the reversed order:

1)Steer left
2)Move Backward
3)Steer right
4)Move Backward results in location C which is not the same location as A.

More generally, the system outcome for a nonholonomic system is path-dependent. [2]

Whereas holonomic kinematics can be expressed in terms of algebraic equations which constrain the internal, rotational coordinates of a robot to the absolute position/orientation of the body of interest, nonholonomic kinematics are expressible with differential relationships only.

As will be seen, this distinction has several implications for the implementation of a control system.

2.2 MODELLING OF A CAR-LIKE ROBOT

2.2.1 Kinematic Modelling

The robot is a four-wheeled car-like vehicle as shown in figure. The rear wheels are used for motion and front wheels are used for steering.

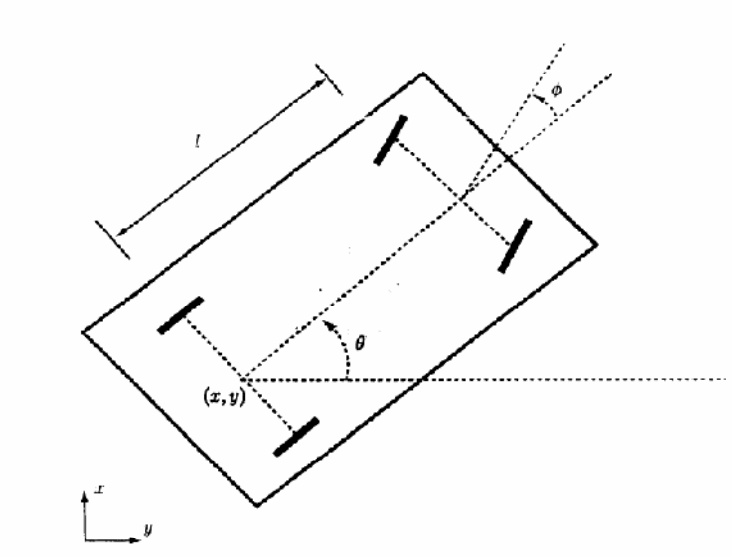


Figure 2.3

In the two dimensional coordinate space the position of the car can be described using parameters (x,y,θ) where x,y are the coordinates of the midpoint of the robot's back wheel axis. θ is the orientation of the car relative to its parallel position.

Given the initial coordinates (x,y,θ) and ϕ, ι where ϕ is the steering angle and ι is the distance between two wheel axis new coordinates of the robot moving with a velocity ν can be calculated using the following formulas: [3]

$$x' = x + dx/dt$$
$$y' = y + dy/dt \quad (2.3)$$
$$\theta' = \theta + d\theta/dt$$

where:

$$dx/dt = \nu\cos\phi\cos\theta$$
$$dy/dt = \nu\cos\phi\sin\theta \quad (2.4)$$
$$d\theta/dt = \nu * \sin\phi / \iota$$

These equations are valid for the robot moving in flat ground in low speed so that there is no significant slippage.

2.2.2 Coordinates of other critical points according to the reference point:

For our case corner points where a collision may most frequently occur are of significant importance. So the robot must know where these points will be located after executing a motion in order to avoid collisions. Coordinates of these points can be obtained using the following equations: [4]

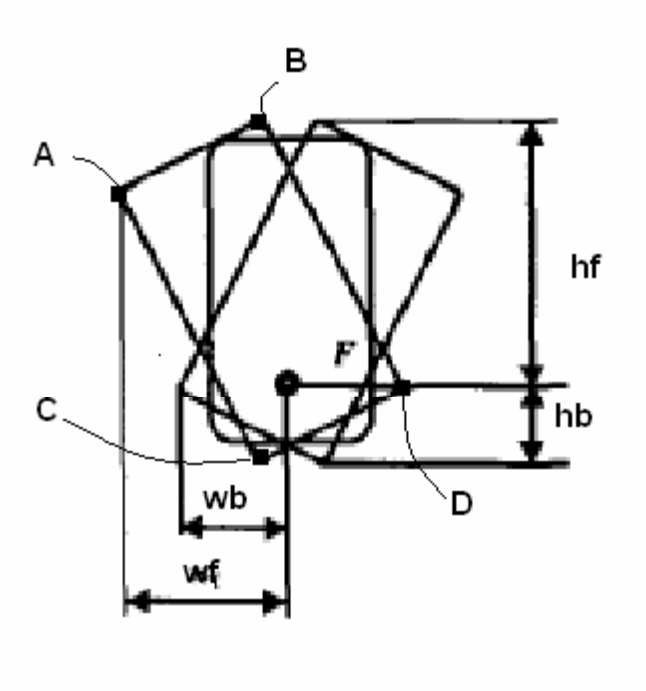


Figure 2.4

a: ½ times length of the car

b: ½ times width of the car

ι: distance between two wheel axis

F: reference point F (Fx,Fy)

θ: is the orientation of the car relative to its parallel position.

A(Ax,Ay)

B(Bx,By)

C(Cx,Cy)

D(Dx,Dy)

$$hf = b*\sin(\theta)+(a+\iota/2) * \cos(\theta)$$

$$hb = b*\sin(\theta)+(a-\iota/2) * \cos(\theta) \qquad (2.5)$$

$$wf = b*\cos(\theta)+(a+\iota/2) * \sin(\theta)$$

$$wb = b*\cos(\theta)+(a-\iota/2) * \sin(\theta)$$

Fy=fy

Fx=fx

Ay=fy-hf+sin(θ)*2*b

Ax=fx-wf

(2.6)

By=fy-hf

Bx=fx-wf+cos(θ)*2*b

Cy=fy+hb

Cx=fx-wf+2*a*cos(PI/2-θ)

Dy=fy+hb-2*b*sin(θ)

Dx=fx-wf+2*a*cos(PI/2-θ)+2*b*sin(PI/2-θ)

2.3 TRACKING PROBLEM CLASSIFICATIONS

Control algorithms are classified as trajectory tracking and path following. The two categories will be discussed in details.

2.3.1 Trajectory Tracking

In the trajectory tracking task, the robot must follow a desired path in Cartesian coordinates with a specified timing law. The tracking task must be executed with a low speed and no slipping effects so that nonholonomic constraints are satisfied. The robot tracks a path generated by a reference robot. The reference robot is assumed to be moving at a desired forward and angular velocities v_d and w_d. Let the desired state of a reference robot be described as:

$$P_d(t)= [x_d(t) ,y_d(t), \theta_d(t)]^T \quad (2.7)$$

Where $x_d(t),y_d(t)$ are coordinates of the desired path in the given time and $\theta_d(t)$ is the orientation angle of the reference robot at the given time and coordinates.

The actual state of the robot is described as:

$$P(t) = [x(t), y(t), \theta(t)]^T \quad (2.8)$$

The objective of the path tracking controller is to control the mobile robot to follow the desired path by controlling the robot forward and angular velocities so that the error between the desired state and the actual state converges to zero.

$$E_p(t) = P_d(t) - P(t) \quad (2.9)$$

As time approaches infinity, the error converges to zero.
The tracking control can be classifed as two-state and three-state control, depending on the number of the control variables (x, y, θ). If any two of these three control variables are tracked by the tracking controller it is called two-state control. If all three variables are tracked it is calles three-state control.

2.3.1 Path Following

Compared with the path tracking problem, path following is a simpler problem. A geometric description of the assigned Cartesian path is given to the controller in the path following task. Since only the geometric displacement between the robot and the path is concerned this task is time independent. The robot forward velocity is set to a constant value and the second input is the angular velocity used as an input for control.

2.4 OPEN LOOP AND CLOSED LOOP SYSTEMS

Automated systems operate under open or closed loop control. A program will control the system. The difference between open and closed loop control is that in open loop control the sequence of commands are carried out by the system irrespective of the consequences. In closed loop systems the controller obtains feedback information from the system and takes appropriate action. In our case closed loop control is used because of the need for dynamic adaptation to the environment. The feedback information is obtained from sensors and the controller decides the next action that shoul be performed.

2.5 PATH PLANNING USING SINUSOIDS

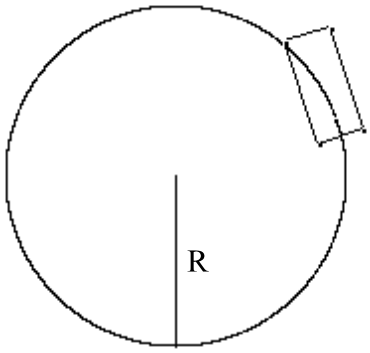


Figure 2.5

With a constant speed and a fixed steering angle planar motion of the robot forms a circle. The radius of the circle depends on the steering angle. The radius denoted by R has minimum value when the steering angle has its maximum.

$R_{min} = l/\tan(\phi_{max})$ [5] (2.10)

ϕ_{max}:maximum steering angle

These circles can be used to form a simple path generating algorithm as control parameters of the modelled car are easily calculated at every single point of the circle. Combining more than one identical circles can be used to generate a path that marks the desired position of the robot. In our case the parallel parking problem corresponds to a sideways displacement. So these circles must be combined to form a path that results in a sideways displacement of the robot body. In the following figure an example path that is obtained using this method is shown. [6]

Figure 2.6

These maneveurs result in a sideways displacement which is required for a parallel parking scenario. The locations of circle center are depending on the parking space and will be calculated by the MCU of the robot. The robot will mark its final position and then dynamically generate a collision free path it can follow in order to realise these maneveurs.

In the next chapter implementation of this path generation algorithm on the paralel parking problem will be explained in detail.

CHAPTER 3

PARALLEL PARKING PROBLEM

In this chapter parallel parking problem and implementations of methods explained in the previous chapter on this problem are explained in detail.

3.1 GENERAL PARALLEL PARKING PROCEDURE

Parallel parking is a method of parking a vehicle in parallel to other parked cars. Cars parked in parallel are in one line, parallel to the curb, with the front bumper of each car facing the back bumper of an adjacent one. [7]

The parallel parking procedure can be divided into three sub phases which are:

1. Scanning Phase
2. Starting Phase
3. Parking Maneveurs

3.1.1 Scanning Phase

In the scanning phase the car moves along a path in which potential parking spaces may exist. The task is to find a suitable parking space which is determined based on dimensions of the car. In our case general case of parallel parking will be considered where a suitable parking space is a rectangular space having the legth of 1.5 times the longtidual distance of the car. Both right and left sides are scanned for a suitable parking space. The default heading of the car is the forward direction. When the car detects an object in the forward direction the search continues in the backward direction. The right side has priority to the left side which means that the car will prefer the right side for parking if parking spaces exist in both sides.

3.1.2 Starting Phase

When a suitable parking space is detected the car enters this phase. In this phase the car first nears the target space with a low steering angle if neccessary. This helps both avoiding collision with the cars residing in the opposite side of the road and positioning the car in a approciate location where parking maneveurs can begin. After the approciate position is reached parking phase begins.

3.1.3 Parking Phase

After reaching the desired location where parking maneveurs can begin, for parking on the right side of the road: First, the steering wheels are turned all the way right and the car begins moving in reverse direction. The car is backed up so that rear right bumper is first to enter the parking spot. Once the vehicle is at 45^{o} to the parking spot the steering wheels are turned all the way to the left. The car is backed up until the car lies parallel to the curb.

3.2 PARALLEL PARKING A CAR-LIKE ROBOT USING SINUSOIDAL ARCS METHOD

In this section a generalized algorithm for parallel parking a car-like robot will be presented. The background materials of the algorithm were provided in the previous chapter. Here implementation of these concepts on the parallel parking problem will be explained. Hardware implementation of the various modules will be explained in chapter 5.

3.2.1 The Car Model

The car-like robot is a rectangular body as shown in figure 2.3 whose control parameters are (x,y,θ). The robot is stepper motor driven so that displacement is time independent. The variation of this parameters can be calculated using formulas given in (2.3) and (2.4) where ν can be replaced by the unit displacement value 1 which corresponds to a 1 cm turn of the rear wheels which provide the motion of the robot. After each motion cycle of the robot new coordinates of the reference point (fx,fy)

are calculated. The corner points of the robot are recalculated according to the formulas explained in chapter 2.2.2 using the formulas (2.5) and (2.6).
The controller obtains the environmental data from the sensors and decides the next action that the robot will perform. Based on the decision of the controller control signals are sent to the motion module and after execution of the task new coordinates are recalculated.

3.2.2 Parking Area Search

During parking area search a closed loop control is used to find a suitable parking space.The initial orientation of the robot is parallel to the y-axis. Execution of each motion corresponds to a 1 cm displacement in the +y direction with a zero displacement in the x direction and a zero degrees of variation in the orientation angle as the steering angle is also zero during this phase.

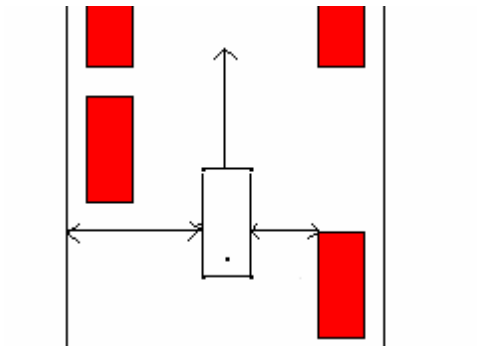
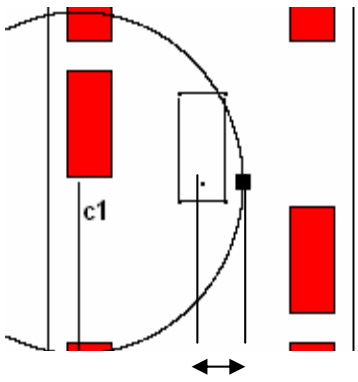


Figure 3.1

The default direction of the car is the forward direction and the car searchs for a parking space in this direction until an object is detected or the operation is stopped manually. The forward sensor data is used to avoid collisions that might occur during the park search.



Δx

After each cycle which corresponds to a 1 cm displacement in the y-direction the parking space variables are updated. The distance between the robot and parked cars is also recorded that will be used in the generation of a dynamic path to the desired location.
If the variable that holds the corresponding parking space data exceeds a limit value that spot is marked as a potential desired location.The center of the circle c1 is horizontally perpendicular to this point.

This point is where fx,fy will be located if the parking space is decided to be of sufficient length. The coordinates of the circle c1 center are :

$$c1x = fx + \Delta x - R \qquad (3.1)$$
$$c1y = fy$$

where Δx is the amount of sideways displacement of the location with reference to the initial x coordinate of the robot.

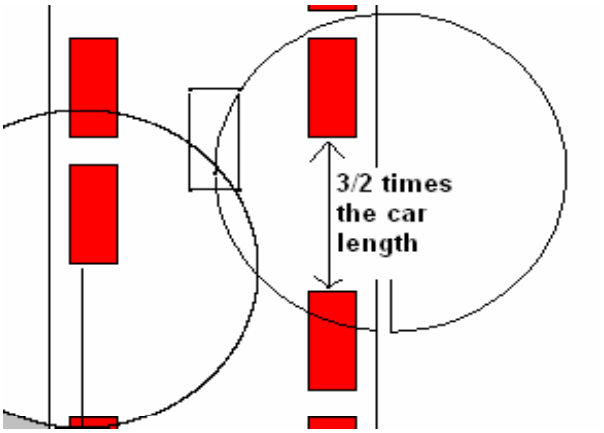


Figure 3.3

If the parking space is of sufficient length the robot enters the starting phase.

3.2.2 Starting Phase

When sufficient parking space is detected the robot begins the starting phase. In the starting phase the robot moves to an approciate position where parking maneveurs can begin. The task in this phase is to reach the desired location that was marked during the scanning phase by using the path following algorithm using the sinusoidal arcs method.

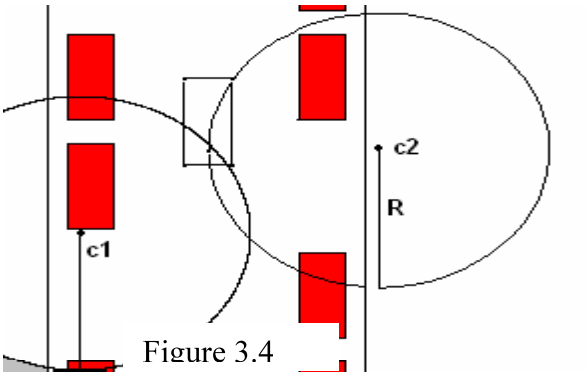


Figure 3.4

First, another circle is formed that represents the reachable points by the car with the determined steering angle. The center of this circle is calculated using the formula:

c2x = fx + R

c2y = fy

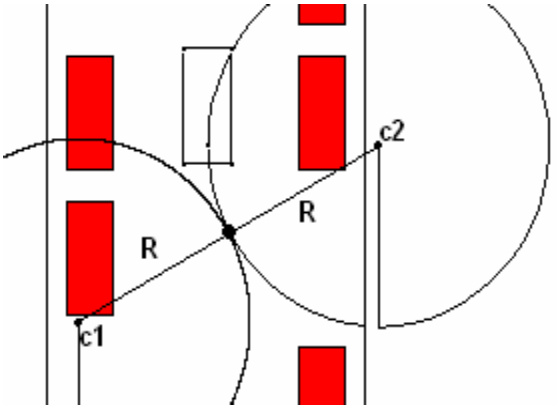


Figure 3.5

The car moves forward until two circles intersect only at one point which is the tangent point. At the tangent point the distance between c1 center and c2 center equals 2*R. Using this property tangent point can easily be located using the formula:

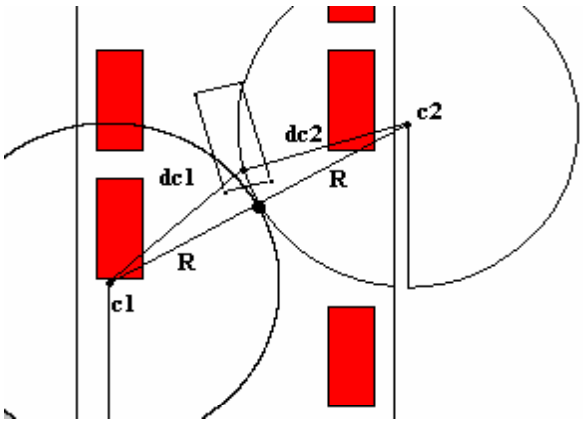


Figure 3.6

Then the car is backed up with a determined steering angle to the right so that it reaches the tangent point. At the tangent point

$$dc1 + dc2 = 2 * R \qquad (3.3)$$

dc1: distance between c1 center and car reference point (fx, fy)

dc2 :distance between c2 center and car reference point (fx, fy)

$$dc1 = \text{sqrt}\,[(fx-c1x)^2 + (fy-c1y^2)] \quad (3.4)$$

$$dc2 = \text{sqrt}\,[(fx-c2x)^2 + (fy-c2y^2)] \quad (3.5)$$

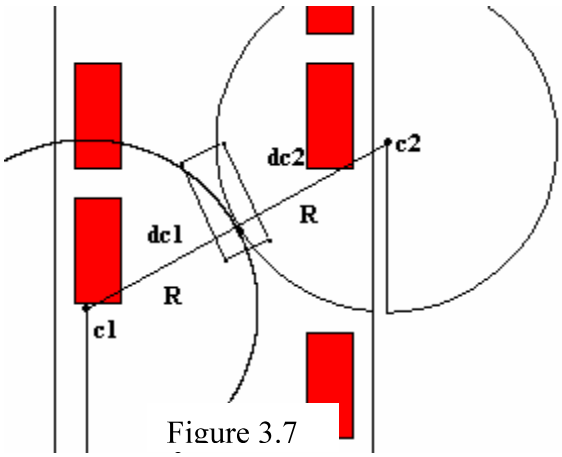
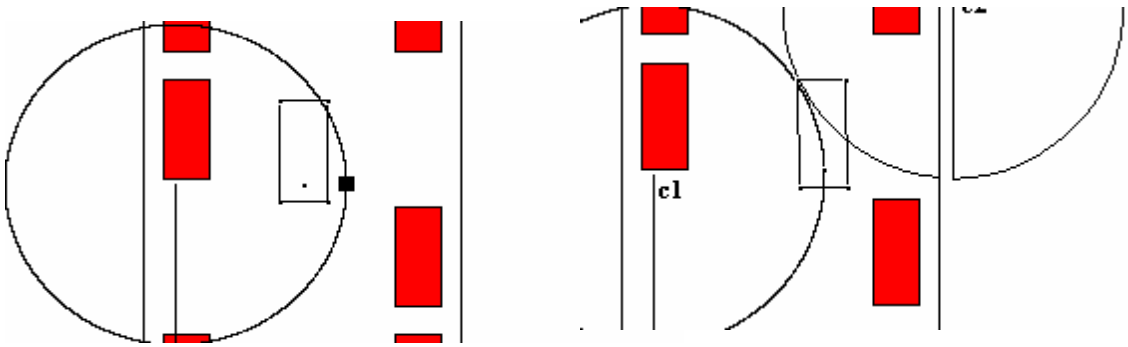


Figure 3.7

When the tangent point is reached c1, c2 and F (fx, fy) lie on the same line. Now the steering wheels are turned to the opposite side with the same steering angle and the car begins to follow the arc generated by the circle c1.

Finally the car is backed up with the determined steering angle until its orientation angle is again parallel to the y-plane. The sideways displacement obtained can be observed by comparing the given figures 3.8.a and 3.8.b.



a) b)

Figure 3.8

3.2.3 Parking Maneveurs

In the parking maneveurs the same approach is used that was explained in the starting phase with a sideways displacement resulting in the final location where the car is desired to be located after the parking is completed. Different from the starting phase greater steering angle values are used during parking maneveurs to increase maneveuring capability of the car.

The parking maneveurs for the given scenario are shown in the following figure:

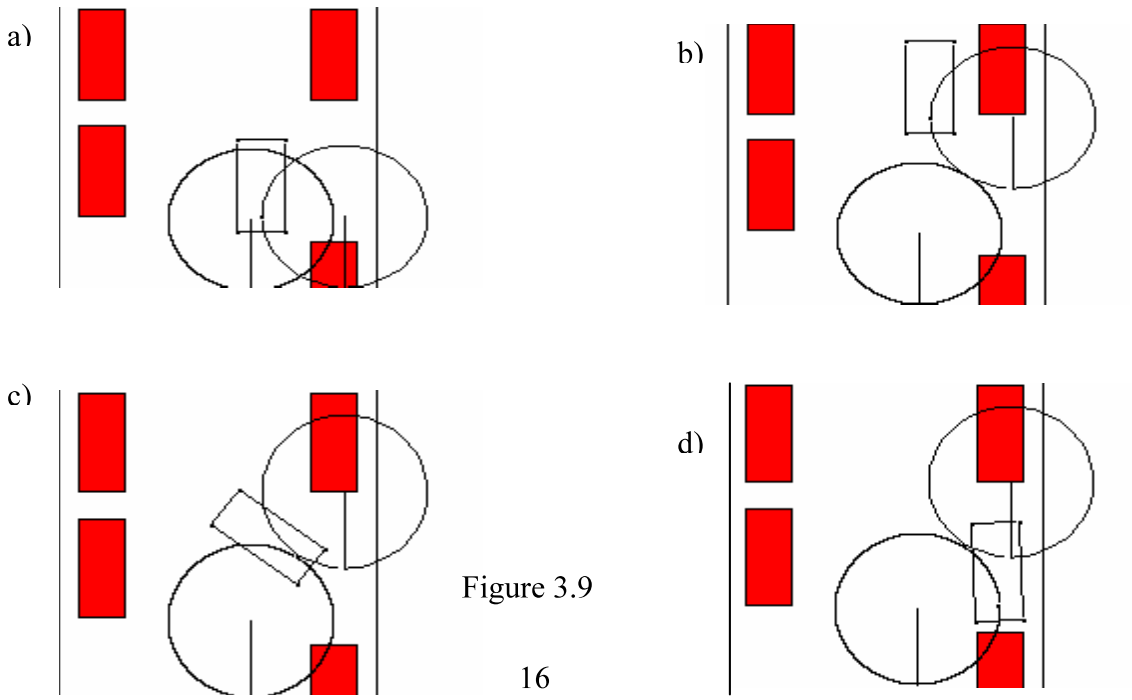


Figure 3.9

CHAPTER 4

EXPERIMENTAL STUDIES

In this chapter practical application of the parallel parking algorithm on an car like robot is explained. The application consists of two steps. First, a visual computer program running in PC environment is developed in order to simulate the approach which was discussed in the previous chapter. The program allows the user to generate random scenarios for parallel parking. In the first section of this chapter the simulation program is explained in detail. In the second part, hardware implementation of the parallel parking algorithm on a car like robot and modules that are used are explained.

4.1 SIMULATION

A simulation is an abstraction of a physical phenomenon. Simulation programs are used for imitating or estimating how events might occur in a real situation. In our case simulating the approach that was discussed has great importance since the car-like robot will be controlled using an embedded software in the controller. It must be verified that both the mathematical, kinematic model and the parallel parking method satisfy the laws of the real world.

In the simulation program, physical model of a car-like vehicle is generated first. The coordinates of the reference point F (fx,fy) is the main control point. The corner points of the vehicle are generated with referencing to the reference point using the formulas (2.5) and (2.6). The kinematic model explained in chapter 2.2.1 is implemented on this model. After each cycle of the main program loop,coordinate parameters (x, y ,θ) of the reference point are recalculated depending on the previous values of (x, y ,θ) and the steering angle ϕ.

On this car model four sensors were modelled which obtain the environmental data from the simulation screen by checking color changes which is an abstraction of the real world operation of sensors.

A random scenario generator program has been developed in order to simulate the environment.

For decision making tasks a program simulating the controller is generated which obtains the environmental data, makes decissions and sends the control signals to the motion module simulator.

In figure 4.1 a screen shot from the simulator running in PC environment is seen.

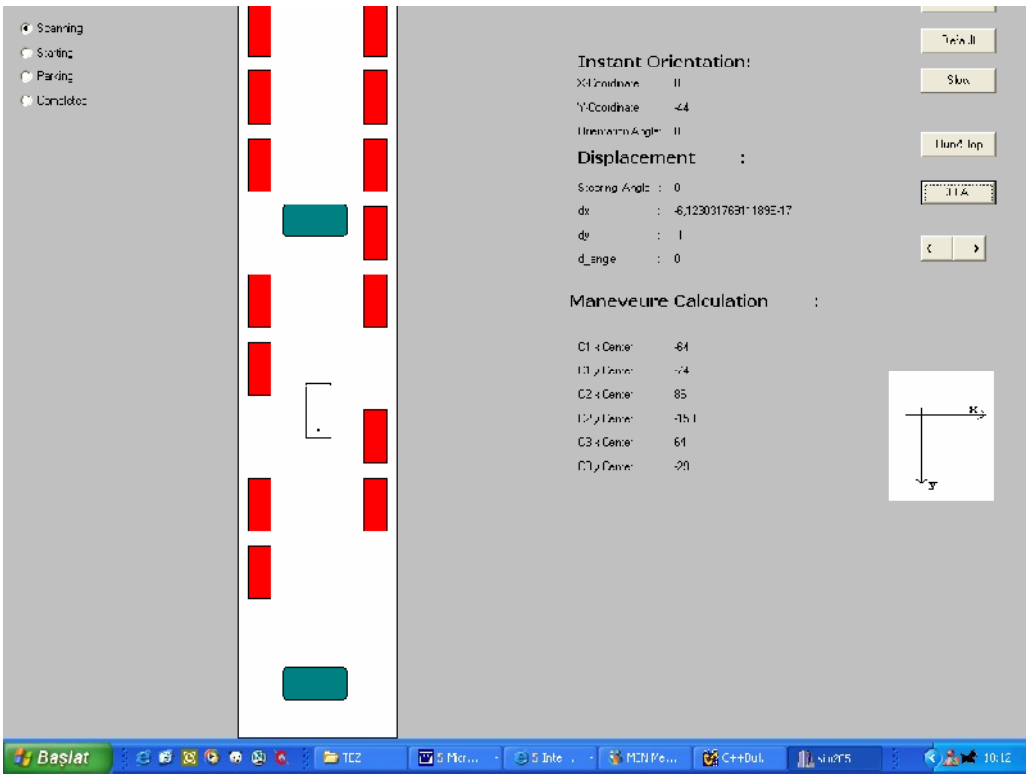


Figure 4.1

The flow chart and the source code of the simulator program written in Borland C++ Builder development environment can be seen in AppendixA.

4.2 HARDWARE

Realizing the parallel parking algorithm on a car like robot requires obtaining environmental data, processing data, making decisions and driving the car to perform the task. This requirements are satisfied using three modules of hardware:

1) The Sensor Module
2) The Microcontroller Module
3) The Motion Module

In the following sections these modules and their integration will be explained in detail.

4.2.1 Sensor Module

Sensors are devices that convert a physical parameter such as distance to an object, room temperature, blood pressure or wind speed into a signal that can be measured electrically. Once the physical parameter has been converted to an electrical equivalent it is easily input into a computer or microprocessor for manipulating, analyzing and displaying. For the parallel parking of the car-like robot Sharp GP2D120 disteance measuring infra-red sensors were used to obtain environmental data. These IR sensors operate between 4 ~ 30 cm distances which is sufficient for our 20x50 cm robot.

4.2.1.1 Theory of Operation

Sharp GP2DXX rangers all use triangulation method together with a small linear charge coupled device (CCD) array to compute the distance and/or presence of objects in the field of view. The basic idea is this: a pulse of IR light is emitted by the emitter side of the sensor. This light travels out in the field of view and either hits an object or just keeps on going. In the case of no object, the light is never reflected and the reading shows no object. If the light reflects off an object, it returns to the detector and creates a triangle between the point of reflection, the emitter, and the detector as shown in the figure: [8]

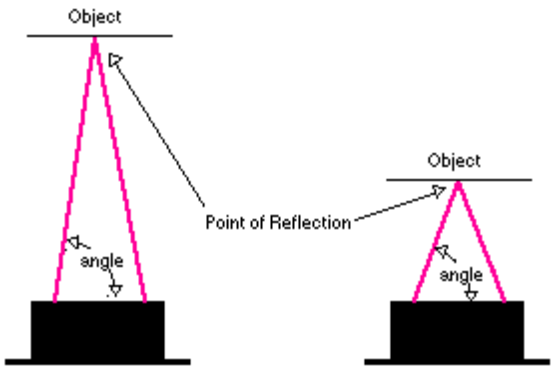


Figure 4.2

The angles in this triangle vary based on the distance to the object. The receiver side of the detectors is actually a precision lens that transmits the reflected light onto

various portions of the enclosed linear CCD array based on the angle of the triangle described above. The CCD array can then determine what angle the reflected light came back at and therefore, it can calculate the distance to the object. So corresponding analog voltage level can be measured at the output which varies between 0.3 ~ 3.1 V. This new method of ranging is almost immune to interference from ambient light and offers indifference to the color of object being detected. Detecting a black wall in full sunlight is possible.

4.2.1.2 Linearization of the Output

Although this new method of ranging offers many advantages the output of these new detectors is non-linear with respect to the distance being measured because of some basic trigonometry within the triangle from the emitter to reflection spot to receiver. The following graph obtained from Sharp GP2D120 datasheet shows a typical output from these sensors: [9]

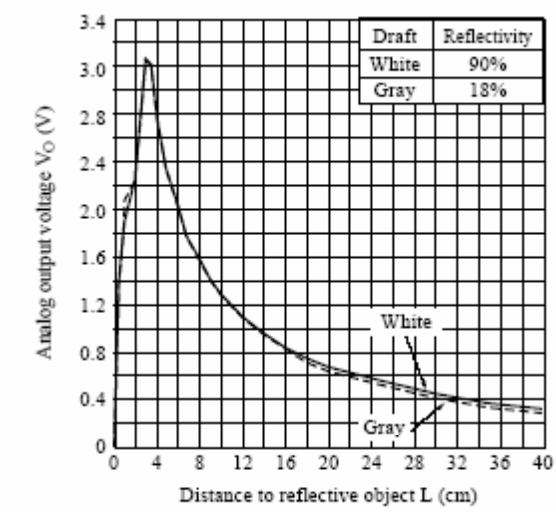


Figure 4.3

As shown in the graph on figure 4.3 the output of the detectors within the stated range (4 cm - 30 cm) is not linear. This curve has to be normalized with a lookup table or parameterized function. In this way distance information can be obtained by matching voltage levels to metric values.

In our case the output analog voltage has to be converted to digital data using analog digital converter (ADC) ports of the PIC16F877 microcontroller which will be explained in detail in 4.2.2.4. For now, the linearization of the binary 8-bit data received from the ADC will be explained. The following figure shows the test circuit used for obtaining 8-bit binary data from the sensor output.

Figure 4.4

In the test circuit LM7805 voltage regulator integrated circuits are used to provide the circuit with a constant regulated voltage level. The sensor input voltage is +5V and sensor ground is the common ground of the circuit. The ADC of the 16F877 is configured to operate with a reference voltage of +5V. The resulting data is displayed on the leds connected to PORTC of the 16F877. A low-pass filter is implemented to the output of the sensor to reduce noise effect on the measurements.

By using the test circuit shown in figure 4.4 various distance measurements were made. At each distance ranging from 4 to 30 cm 100 conversions were made and the average value displayed was recorded. (For a detailed explanation on the usage of the A/D Converter look at 4.2.2.4)

Table 1

Metric Distance (cm)	ADC Decimal Value
4	173
5	148
6	125
7	108
8	95
9	84
10	75
11	68
12	64
13	57
14	52
15	50
16	45
17	43
18	40
19	38
20	35
21	33
22	32
23	30
24	29
25	28
26	27
27	24
28	23
29	22
30	12
>30	<12

Putting these values into a graph figure 4.5 is obtained:

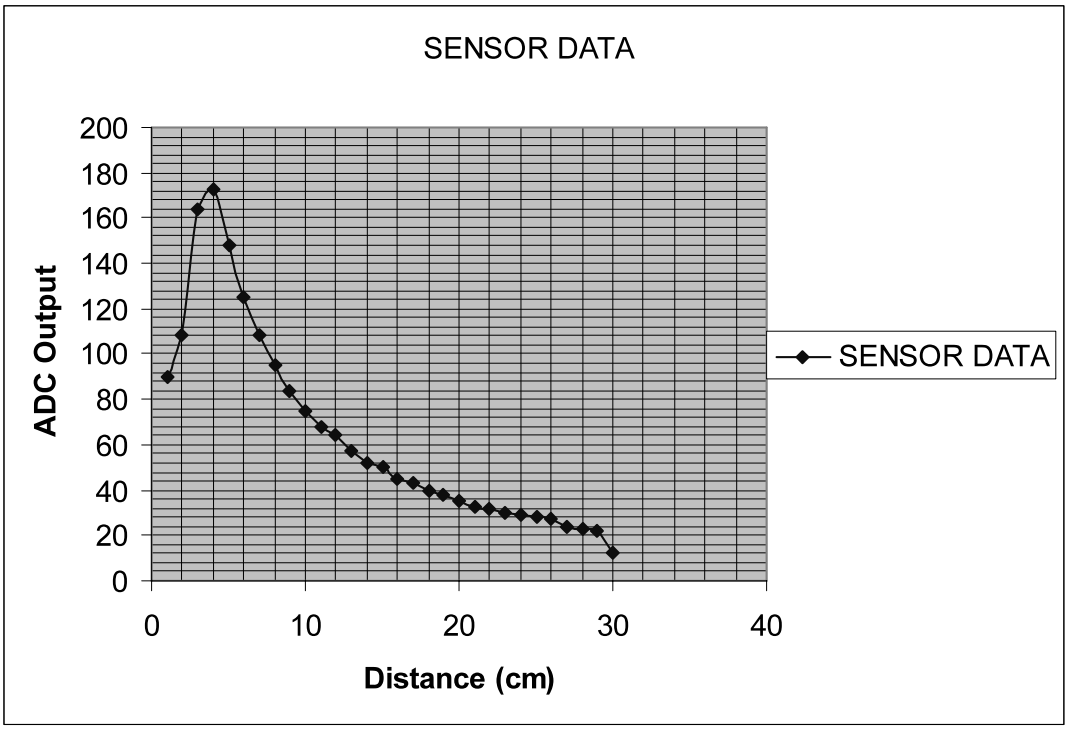


Figure 4.5

It is seen that the graph in figure 4.5 looks similar to the graph shown in figure 4.3 from the datasheet.

4.2.1.3 Conclusions

The experiment showed that the sensor data resolution is sufficient in the interval 4-20 cm which covers the critical distances for the parallel parking problem of a car-like robot with dimensions 20x50 cm. The second thing to notice in the graph is that the output looks like a longer range reading when the object lies nearer than 4 cm. Additional low range sensors can be used to overcome this problem.

4.2.2 The Microcontroller Module

The microcontroller module is used for processing of information obtained from the sensor module, making decisions and sending approciate control signals to the motion module.

4.2.2.1 Microprocessors and Microcontrollers

A microprocessor is a silicon chip that contains a central processing unit (CPU). A microprocessor needs additional hardware components in order to operate in a system. These components can vary according to different needs but RAM, I/O and data bus are the main parts of a microprocessor system that can not be omitted. Microcontrollers are highly integrated chips that contain all the components comprising a controller. There are many different microcontrollers which vary according to their memory size, operating speed, number of ports, additional modules like analog/digital converter, USART etc... In this project Microchip PIC16F877 microcontrollers were chosen because of their large memory size, built-in ADC, low price and incircuit programming capability.

4.2.2.2 Microchip PIC16F877

Microchip PIC16F877 is a powerful (200 nanosecond instruction execution) yet easy-to-program (only 35 single word instructions) CMOS FLASH-based 8-bit microcontroller packs Microchip's powerful PIC® architecture into an 40- or 44-pin package and is upwards compatible with the PIC16C5X, PIC12CXXX and PIC16C7X devices. PIC16F877 features 256 bytes of EEPROM data memory, self programming, an ICD, 8 channels of 10-bit Analog-to-Digital (A/D) converter, 2 additional timers, 2 capture/compare/PWM functions, the synchronous serial port can be configured as either 3-wire Serial Peripheral Interface (SPI™) or the 2-wire Inter-Integrated Circuit (I²C™) bus and a Universal Asynchronous Receiver Transmitter (USART). All of these features make it ideal for more advanced level A/D applications in automotive, industrial, appliances and consumer applications.[10]

4.2.2.3 PIC16F877 Pin Diagram

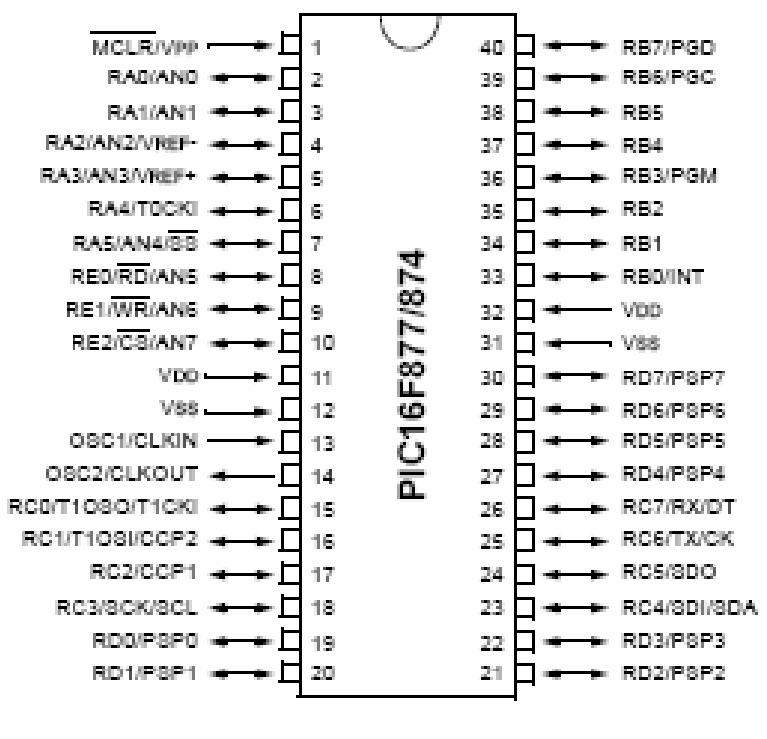


Figure 4.6

In figure 4.6 pin diagram of the PIC16F877 is shown. In this section short descriptions of these pins will be given. Detailed information can be seen PIC 16F877 datasheet available in www.microchip.com. Pins of PIC16F877 are multi-purpose pins. Here their operating modes in the project are explained.

Pin # 1: $\overline{\text{MCLR}}$: Is an active-low input used for resetting the controller. During operation this pin is connected to +Vdd. For programming +13 V is applied to this pin (for high level programming) which indicates the controller that program is written to the flash memory.

Pin # 2 - 7: PORTA : PORTA is a 6-bit wide, bi-directional port. The corresponding data direction register is TRISA. PORTA can be configured for digital I/O or analog input. The operation of each bit is selected by setting/clearing the corresponding bits in the TRISA, ADCON0, ADCON1 registers. ADC operation of this port will be explained in detail in section 4.2.2.4.

Pin #8–10: PORTE: PORTE has three pins (RE0/RD/AN5, RE1/WR/AN6,and RE2/CS/AN7) which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers. PORTE pins are also multiplexed with analog inputs that can be used as ADC. ADC operation of this port will be explained in detail in section 4.2.2.4.

Pin # 11&32: Vdd: The 16F877 best operates at a +5V regulated voltage level. Although the PIC can operate at a wide range of voltage levels (2 ~6 V) best performance / low power consumption is with +5 V.

Pin #12&31:GND: This pin is connected to the common ground of the circuit.

Pin # 13&14: OSC1 & OSC2: The PIC requires an impulse clock generator to operate. Different clock generator can be used.

RC: Resistor / Capacity pairs are used for timing tolerant applications. The frequency can vary in a +-20% of the calculated frequency between 0~4 MHz.

XT: Crystal clock generators are used in applications where accurate timing is needed between (0~4 MHz)

HS: High Speed crystal oscillators are used when high speed and accurate timing is necessary between 4~20 MHz.

Pin #15-18 & Pin 23-26: PORTC: PORTC is an 8-bit wide, bi-directional digital I/O port. The corresponding data direction register is TRISC. In this project PORTC is used as digital output for display purposes.

Pin #19-22 & Pin 27-30: PORTD: PORTD is an 8-bit wide, bi-directional digital I/O port. The corresponding data direction register is TRISD. In this project first two bits are used as digital input that obtain feedback information from the steering wheels of the robot.

Pin #33-40 : PORTB: PORTB is an 8-bit wide, bi-directional digital I/O port. The corresponding data direction register is TRISB. In this project PORTB is used as digital output for generating the proper sequences to drive both steering and driving stepper motors.

4.2.2.4 A/D Convertor

The PIC16F877 has 8 analog inputs. The analog input charges a sample and hold capacitor. The output of this capacitor is used as the input to the converter. The converter then generates a digital result corresponding to this analog level. The converted digital data is 10 bits wide which is stored in ADRESH and ADRESL registers. The A/D convertor has 4 registers which define the operation:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register0 (ADCON0)
- A/D Control Register1 (ADCON1)

The ADCON0 register controls the operation of the A/D Convertor and ADCON1 register controls the operation of the port pins. Detailed information on configuring these registers can be seen in PIC 16F877 datasheet available in www.microchip.com.

The steps followed for doing an A/D Conversion in this project is explained :

1. Configure the A/D module:
- Configure analog pins/voltage reference and digital I/O (ADCON1)

```
ADCON1=0X00;          // All 8 pins are configured as analog inputs
                      //  with  Vref+=Vdd and Vref-= Vss
```

- Select A/D input channel (ADCON0)
- Select A/D conversion clock (ADCON0)
- Turn on A/D module (ADCON0)

```
ADCON0=0X41;          // AN0 is selected
                      // fosc/32 is selected as A/D conversion clock
                      // A/D module is turned on
```

2. Configure A/D interrupt (if desired):
- Clear ADIF bit
- Set ADIE bit

• Set PEIE bit

• Set GIE bit

In this project A/D module is used in polling mode without using interrupts.

3. Wait the required acquisition time.

```
delay_short(1);                    //generates a 20 us delay
```

4. Start conversion:

• Set GO/DONE bit (ADCON0)

```
ADCON0=0X45;                    //Sets GO/DONE bit.
```

5. Wait for A/D conversion to complete, by either:

• Polling for the GO/DONE bit to be cleared (with interrupts disabled) OR Waiting for the A/D interrupt

```
while((ADCON0 & 4)!= 0);  //checks if the GO/DONE bit is cleared
```

6. Read A/D result register pair (ADRESH:ADRESL), clear bit ADIF if required.

7. Go to step 1 or 2 for next conversion

For obtaining proper data from the A/D convertor module acquisition times and minimum wait times required before the next acquisition starts must be considered. For the timing diagrams look at PIC16F877 datasheet.

4.2.3 Motion Module

The motion module consists of two stepper motors and their driver chips, the ULN2003A. Stepper motors are used for both steering and the driving of the robot because accurate control on the motion of the robot is needed for the controller software to predict the displacements of the robot body.

4.2.3.1 Stepper Motors

Stepper motors are electrical motors without commutators. Different from a DC motor, stepper motors offer opportunities for precise positioning. The motors and

their controllers are designed so that the motor can be held in any fixed position as well as being rotated one way or the other. A stepper motor moves one step when the direction of current flow in the field coil(s) changes, reversing the magnetic field of the stator poles.

Stepping motors come in two varieties, permanent magnet and variable reluctance. Stepper motors can be distinguished between the two varieties with an ohmmeter. Variable reluctance motors usually have three (sometimes four) windings, with a common return, while permanent magnet motors usually have two independent windings, with or without center taps. Center-tapped windings are used in unipolar permanent magnet motors.

Another important parameter of the stepper motor is the angular resolution. High resolution permanent magnet motors are commonly able to handle 1.8 or even 0.72 degrees per step. In this project stepper motors with 7.5 degrees per step angular resolution are used.

Another classification for stepper motors is made based on the attribute of the drive and relates to whether the current can flow in one (unipolar) or two (bipolar) directions.

4.2.3.1.1 Unipolar Stepper Motors

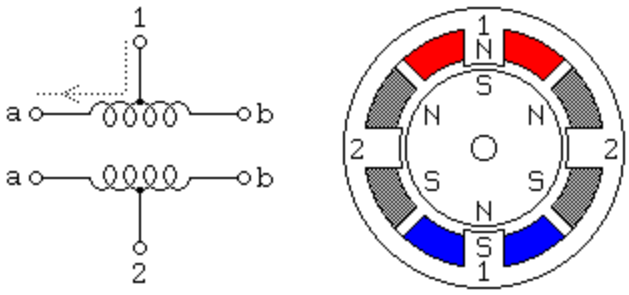


Figure 4.7

Unipolar stepping motors, both Permanent magnet and hybrid stepping motors with 5 or 6 wires are usually wired as shown in the schematic in Figure 4.7, with a center tap on each of two windings. In use, the center taps of the windings are typically

wired to the positive supply, and the two ends of each winding are alternately grounded to reverse the direction of the field provided by that winding.

As shown in the figure, the current flowing from the center tap of winding 1 to terminal a causes the top stator pole to be a north pole while the bottom stator pole is a south pole. This attracts the rotor into the position shown. If the power to winding 1 is removed and winding 2 is energised, the rotor will turn one step. To rotate the motor continuously, power is applied to the two windings in sequence. Stepper motors can be driven in full-step mode or half-step modes. In the full step mode each control nibble (4 bits) sent to the motor will cause the motor to rotate for one step (7.5 degrees). Driving in half-step mode offers more control on the position of the motor with the motor stopping alternately at the positions indicated by one or the other sequence.

The following table shows both the full-step and half-step mode driving sequences:

Table 2

Full Step			
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

Half-Step			
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	0	0
1	1	0	0
1	0	0	0
1	0	0	1

In this project unipolar stepper motors were driven in full-step mode with a angular resolution of 7.5 degrees. The sequences were generated by the microcontroller module.

4.2.3.1.2 Bipolar Stepper Motors

Figure 4.8

Bipolar permanent magnet and hybrid motors are constructed with exactly the same mechanism as is used on unipolar motors, but the two windings are wired more simply, with no center taps. The only difference is that the current driving these motors is flowing in two directions as the name bipolar implies. Because of this the driving circuit is more complicated for bipolar stepper motors. The driving sequences are the same as shown in Table 2.

4.2.3.2 Stepper Motor Driver Circuit

Microprocessors are not capable of supplying the required current by the stepper motors. For this reason additional circuit is necessary for driving the stepper motors. For applications where each motor winding draws under 500 milliamps, the ULN200X family of darlington arrays can be used.

Figure 4.9

4.2.4 Autonomous Parallel Parking Robot

A car-like robot has been developed with 20x50 cm dimensions. This is a four wheeled rear-driven car with front-steering wheels. It is equipped with 2 stepper motors. One of the stepper motors is used for driving the car and the other is used for controlling the steering system. 4 infra-red sensors are placed on the robot as shown in the figure.

Figure 4.10

The sensors shown in figure 4.10 are used to obtain environmental data. The FW and BW sensors are used to prevent collisions during parking area search. The left and right sensors scan both sides of the road in order to detect parking spaces. These

sensors are connected to ADC pins of the microcontroller. The data obtained is recorded and decisions are made according to this information by the microcontroller through software. Then approciate control signals are sent to the stepper motors.
In figure 4.10 the resulting circuit is shown which consists of:

1 x PIC16F877
4 x 6V Power Supply
1 x 7805 Voltage Regulator
2 x ULN2003A stepper motor driver Ics
1 x 20 MHz Crystal Clock Generator
4 x Sharp GP2D120 IR Sensors
1 x 7447 7 Segment display driver
2 x 7 Segment displays
2 x Stepper Motors

Figure 4.11

The circuit shown in figure 4.11 was printed on a PCB for ease of implementation. The following figure shows the bottom copper layer of this PCB created with ARES.

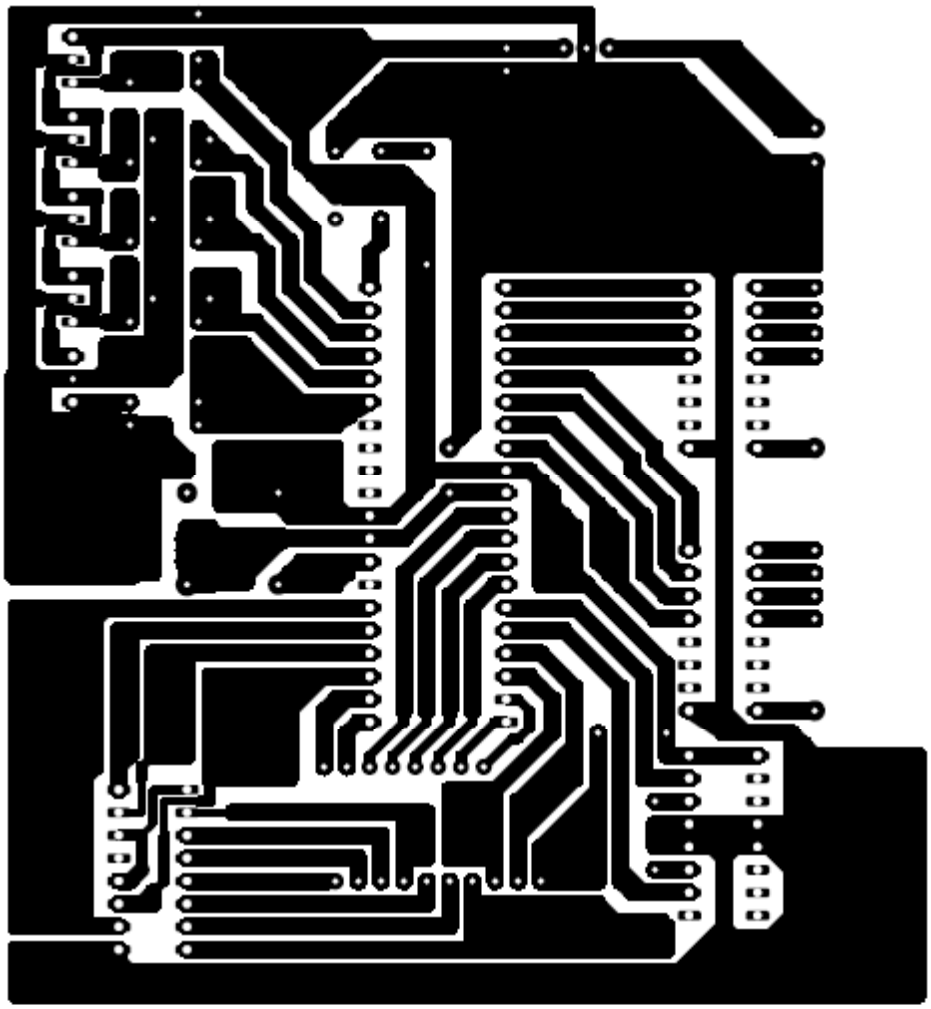


Figure 4.12

The car model explained in Chapter 2 and the parallel parking algorithm explained in Chapter 3 has been implemented on the robot through software. The source code of the program running in the microcontroller is shown in Appendix B.

CHAPTER 5

CONCLUSIONS

The parallel parking algorithm introduced in Chapter 3 has been realized on the car-like robot using closed loop control and path following methods. The approach was verified with experimental studies. The method can be used in different sized cars by modifying relevant parameters (car length, width etc..) in the controller software.

The approach solves the parallel parking problem for a general case. The approach can be implemented on more complex scenarios which include smaller parking spaces with the use of hybrid sensor systems for a better modelling of the environment.

References

[1] L. Pars, “A Treatise on Anaytical Dynamics”, Heinemann, London, 1965

[2] Non-holonomic Kinematics
http://www.nd.edu/NDInfo/Research/sskaar/NonholonomicKinematics.html

[3] Laugier C., Fraichard Th., Paromtchik I. E. and Garnier Ph., 1998. Sensor-Based Control Architecture for a Car-Like Vehicle, *IEEE International Conference on Intelligent Robots and Systems,* Victoria, B.C., Canada, October 1998.

[4] Jiang K. and Seneviratne L.D., 1999. A Sensor Guided Autonomous Parking System For Nonholonomic Mobile Robots, *IEEE International Conference on Robotics & Automation*, Detroit, Michigan,USA, May 1999.

[5] La Valle, M.S., 1999. Planning Algorithms, University of Illinois, Illinois.

[6] Lo Y. K., Rad A. B., Wong C.W. and Ho M. L., 2003. Automatic Parallel Parking, *IEEE Proceedings on Intelligent Transportation Systems*

[7] Wikipedia, The Free Encyclopedia,
http://en.wikipedia.org/wiki/Parallel_parking

[8] Sharp General Application Note: Distance Measuring Sensors
www.sharp.com

[9] Sharp GP2D120 Datasheet
www.sharp.com

[10] Microchip PIC16F877 Datasheet
www.microchip.com