NEAR EAST UNIVERSITY

Faculty of Engineering

Depatment of Computer Engineering

COMPUTER – AIDED CONTROL OF MOBILE ROBOT

Graduation Project
COM – 400

Students : Boran Şekeroğlu (970065)

Hani Abuamer (970222)

Supervisor : Asst. Professor
Rahib Abiyev

Lefkoşa – 2001

TABLE OF CONTENTS

ACKNOWLEDGMENT
ABSTRACT i
INTRODUCTION ii
1. STATE OF ART UNDERSTANDING PROBLEMS OF MOBILE ROBOTS CONTROL 1
1.1 Control Problems of Mobile Robot 3
1.2 Robot Models and Control 3
1.3 Integrating High-Speed Obstacle Avoidance , Global Path Planning and Vision Sensing on a Mobile Robot 6
1.3.1 Obstacle Avoidance 11
1.3.2 Vector Field Histogram 13
1.3.3 Global Path Planning 15
1.3.4 Vision Sensing 18
1.3.5 The Algorithm 19
1.3.6 Localization 19
1.3.7 The Supervising Execution System 20
2. STRUCTURE OF INTERFACE AND SIGNAL TRANSMISSION 22
2.1 Signal Characteristics **24**
2.1.1 Analog Signals 24
2.1.2 Digital Signals 24
2.1.3 Some Common Terms 26
2.1.4 Problems of Using Voice Channels for Digital Transmission 27
2.2 Structure and Elements of Communication System 28
2.3 Channel Organizations 30
2.3.1 Characteristics of Communication Channels 33
2.3.2 Terminology 35
2.4 Interface by Parallel Port 38
2.4.1 Hardware Properties 41
2.4.2 Port Addresses 43
2.4.3 Software Registers – Standard Parallel Port (SPP) 45
2.4.4 Using the Parallel Port to Input 8 – bits 46
2.4.5 Parallel Port Modes in BIOS 49
2.5 Interface by Serial / RS232 Port 51
2.5.1 Hardware Properties 52
2.5.2 Flowchart 53
2.5.3 Connection of Two DTE Devices 56
2.6 DAC & ADC Conversion 58
3. NAVIGATION OF MOBILE ROBOT 61
3.1 Location Estimation of Mobile Robots **68**
3.2 Navigation of Mobile Robot Under Uncertainty Model 68
3.3 Real – Time Vision – Based Navigation and 3D Depth Estimation 70
3.3.1 Navigation 74
3.3.2 3D CAD Processing 74
3.3.3 Egomotion and Depth 76
76

4. **MOBILE ROBOT NAVIGATION USING FUZZY LOGIC** **80**
4.1 Fuzzy Logic Control 80
4.1.1 Fuzzy Expert Systems 81
4.2 Fuzzy Inference Process 81
4.2.1 Fuzzification 82
4.2.2 Inference Mechanism 83
4.2.3 Composition 84
4.2.4 Defuzzification 85
4.3 Software Implementation of the Fuzzy Control System 86
4.3.1 Fuzzy Control Module 87
4.4 Fuzzy Logic for Local Guidance of Mobile Robot 88
4.4.1 Local Guidance 89
4.4.2 Fuzzy System for Local Guidance 90
5. **COMPUTER – AIDED CONTROL OF MOBILE ROBOT** **99**
5.1 Structure of Control System of Mobile Robot 99
5.1.1 Transmitter 103
5.2 Technical Characteristics of Mobile Robot 104

CONCLUSION **108**
REFERENCES **109**

ACKNOWLEDGMENT

First of all, we would like to acknowledge as well our parents Sadan - Erdogan Sekeroglu & Najah - Abdulrahman Abu Amer. What ever we can write or say, we can not reserved them.

We would like to thank our supervisor Asst. Prof. Dr. Rahib Abiyev who gives us desire and corporation to present our project in this manner.

Also we would like to thank for all our staff of Faculty of Engineering.

Thanks to all our friends specially Boran Tolga Demir and Murat Küçük who helped us in the electronics parts of our project.

Finally ;

Special thanks for our darlings (Susen and Samah) that make us stronger and stronger to finish our university .Our best wishes for them.

What ever we do, we do it for you.....

Hani & Boran

We hope you will enjoy reading our project....

ABSTRACT

One of the actual field of application of Artificial Intelligence ideas is mobile robots , which allow us to solve complicated control problems in uncertainty of environment , fuzzyness of information . The project is devoted a computer aided control of mobile robot .

To solve this problem the state of application artificial intelligence ideas to mobile robot control is given .

Control structure of navigation of mobile robot , the function of its main blocks are described . The control algorithms of mobile robots navigation in uncertainty condition are described . The model of uncertainty in position and orientation is described .

Also the navigation of mobile robot by using fuzzy logic is given . Use of fuzzy logic provides smooth path and allows guidance,obstacle avoidance and docking using ultrasonic sensors. The fuzzy rules for control of steer motion , speed change , distance are described .

The structure of computer – aided control of mobile robot is given , the function of its main blocks are given . The program to transmits signal from computer to mobile robot is developed .

The obtained experimental results show the efficieny of used methodology .

INTRODUCTION

Mobile robots pose a unique challenge to artificial intelligence researchers . They are inherently autonomous and they force the researcher to deal with key issues such as uncertainty (in both sensing and action) , reliability, and real – time response . Mobile robots also require the integration of sensing , acting and planning within a single system . These are all hard problems , but ones that must be solved if we are to have truly autonomous , intelligent systems .

The navigation of a mobile robot through an environment with obstacles is based on the knowledge of the vehicle's current position . Then position estimation is a very important capability for a mobile robot in order to perform an useful mission .

The robot's odometric system provides fast positional information , but involves unavoidable errors that can increase positional uncertainty . Therefore , by using dead reckoning techniques and the vehicle's kinematic model it is possible to produce an estimation of the current position . This method has a high performance for short distances and can be computed very quickly , so it is used in the vehicle's real – time controller .

However , because these estimationed are not free of errors , a higher level system is necessary to periodically reduce the position uncertainty by using external sensors . Position uncertainty reduction techniques are very time – expensive , therefore they must only be used when the uncertainty has grown to dangerous levels .

Some of the most difficult problems or needs that arise when developing an autonomous mobile robot are navigation and perception of surroundings .

The first has been addressed for a long time and basically requires triangulation of detected references with known location and / or deadreckoning . Occupancy grids created using depth information can also be used for navigation by matching them with a theoretical 2D map of the surroundings and are a good basis for fusion of data provided by diverse sensor types .

Perceiving surroundings is by nature a complex task due to the broad amount of obtainable characterisics and focus will be on estimating depth . This has been addressed using sensors with precise range detection capabilities such as laser and infra – red , but with some drawbacks in cost and the need for mechanical 2D or 3D scanning . Video based solutions , although not as precise , offer an advantage in terms of electronically scanning the view in sight but must cope with extracting 3D information from the projection of illumination on a 2D plane thus increasing computational complexity . Although stereo vision techniques have been succesful in estimating depth , it has been shown that a single moving camera allows estimating relative depth , camera motion and the relative motion of the objects in the scene . Techniques using such an approach basically come in the flavors of feature correspondence and optic flow . Stereo solutions using these techniques have also been presented .

It is possible to guide a mobile by means of the information obtained from the road , when certain brightness requirements as well as road appearance ones are complying with . Two mobile guidance strategies are glimpsed , according to the fore lines . One of them , learning on following the on – road signed lines , allows relativley high speeds , up to 100 Km / h , to be achieved . The other one , basing on detecting the road edges by making use of texture or color differences , requires a complex processing . That is why it allows lower speeds .A system that tries to be robust must include , at least , both guidance strategies and decide which to use , according to the road state .

Basically , two approaches have been proposed by the different research worker groups . Some of them try to succeed in getting a Mobile Vehicle capable of following roads , without learning on previously known road nets , considering the fore approach is invalid . This knowledge , with the road information obtained by means of artificial vision , allows the Mobile Vehicle get its aims easily .

CHAPTER 1

STATE OF ART UNDERSTANDING PROBLEMS OF MOBLE ROBOTS CONTROL

1.1 Control Problems of Mobile Robot

In [1] a local guidance control method for wheeled mobile vehicles equipped with low – cost sensors is presented . This method uses fuzzy logic to provide a smooth path which is not necessarily optimal in distance and / or power consumption but allows guidance, obstacle avoidance and docking using few ultrasound sensors . The module that implements this method receives objectives from a Global Path Generation module as points that the robot should pass by at a specified orientation and speed . An important aspect of this method is that it does not require expensive sensors to create a good map of its surrounding because it relies only on range information provided by inexpensive sensors . Using only this range information also provides real – time performance at low cost .

In [2], it has been developed , built and tested an artificial vision based system to follow roads , which provides control signals , in a short time , by means of a joint of artificial neural nets . The image is segmented in " road " or " not road " . The obtained segmentation is the input for two neural nets , a classic architecture net (NN) , and a TDNN (Time Delay Neural Network) one . The outputs provide by both nets , with a trajectory estimation , are introduced to a decision – making block , which selects the alternative containing less error . The classifier parameters are updated according to the current segmentation .

A transputer based artificial vision system that allows estimating a mobile robot's absolute position (navigation) , it's motion parameters (egomotion) and a depth map of the sight of view is presented in [4]. Navigation is achieved by tracking expected features present in basic 3D CAD information of the environment . Egomotion and relative depth are estimated using optic flow calculated on closed contours of points in the sceme presenting high spatio – temporal gradients and require no a – priori knowledge .

In [5] ,the use of neural networks for robot control tasks constitutes an example of a very powerful approach to nonlinear

process control : the observation - or experience - based learning of the controllers , as opposed to the model – based design of classical controllers . Robot control is one of the fields where extensive research is being performed along this line , especially oriented towards achieving greater degrees of of autonomy in robot operation . One of the most important applications of neural networks in control is the identification of complex nonlinear systems , because of their ability to approximate broad classes of nonlinear functions . In the domain of robotics , the most relevant system identification problems are kinematics and dynamics . In its direct and inverse versions , the kinematics problems are concerned with the mapping between the joint coordinates of robot and the resulting end – effector position and orientation in the physical space . Similarly , the direct and inverse dynamics problems deal with the relationship between joint motor commands and the resulting response of the end – effector .

Neural Networks have also been shown to perform efficiently as controllers of nonlinear processes , alone or in conjunction with other type of controllers . Due to adaptivity properties of the neural learning models , they are especially useful for adaptive control .

Another broad area for the use of neural networks in control is sensorimotor control and , in particular , image – based control . Classical techniques for image processing and image – based positioning remain , however , severaly limited by the requirements of very specific and precise knowledge of the environment , the objects to be viewed , the camera and the robot characteristics . The flexibility and the leaning capability of neural networks can be efficiently exploited for tackling the problem of visual positioning , even in an unstructured or unknown environment .

In a distributed robot system , asynchronous and synchronous communication between the system components is necessary to guarantee problem solving capability in real time . The robot systems consist of several subcomponents , like manipulators , two hand – eye – cameras , one overhead camera and a mobile platform . To get better problem solving capability than the former centralized control architecture , these components are able to work together in teams (asynchronous communication) or special agents (synchronous communication) .

The neccesity for applying learning control arises in situations where a system must operate in conmditions of unceratinty , and when the available a priori information is so limits that it is impossible or impractical to design in advance system that has fixed properties and also performs sufficiently well .

In any closed-loop control system, sensors are used to provide the feedback information that represents the current status of the system and the environmental uncertainties. The sensors used in most control systems are considered to be passive elements that provide raw data to a central controller. The central controller computes the next command based on the required task and the sensor readings. The disadvantage of this scheme is that the central controller may become a bottleneck when the number of sensors increases which may lead to longer response time. In some applications the required response time may vary according to the required task and the environment status. For example, in an autonomous mobile robot with the task of reaching a destination position while avoiding unknown obstacles, the time to reach to the required position may not be important, however, the response time for avoiding obstacles is critical and requires fast response. Fast response can be achieved by allowing sensors to send commands directly to the physical system when quick attention is required.

In this work, several controllers (clients) are working in parallel, competing for the server. The server selects the command to be executed based on a dynamically configured priority scheme. Each of these clients has a certain task, and may use the sensor readings to achieve its goal. A special client with the task of avoiding obstacles is assigned the highest priority. The clients may also acquire the current state of the system and the command history to update their control strategy.

The logical sensor approach, which we used to model the sensory system, allows flexible and modular design of the controllers. It also provides several levels of data abstraction and tolerance analysis based on the sensor type and the required task. This approach is used to build high-level requests which may be used by the application programs.

Any sensory system can be viewed as a passive or dumb element which provides raw data. It can also be viewed as an intelligent element which returns ``analysed" information. Finally, it can be viewed as a commanding element which sends commands to the physical system. Each of these views is used in different situations and for different tasks.

Commanding sensors are an extension to the logical sensor approach in which a mapping from events to actions is added to the sensor model.

We propose a sensor-based distributed control scheme for mobile robots. The application of this scheme to control a real mobile robot is discussed. A server-client model is used to implement this scheme where the server is a process that carries out the commands to be executed, and each client is a process with a certain task. The logical sensor approach is used to model the sensory system that provides different levels of data representation with tolerance measures and analysis.

In some controls of mobile robots , a user recognition voice system and ultrasonic and sensor systems can be integrated . In this way , the mobile robot can move with oral commands and with the possibility of obstacle avoidance and downstairs or hole detection . The electronic control systems that built with user recognition voice system allows an easy and handy moving with only use of the human voice . The system moves itself through structured environments . System structure is in a way that is possible to add more features , easy and economically .

1.2 Robot Models and Control

A robot is a reprogrammable multifunctional mechanical manipulator designed to move materials , parts , tools , or special devices through planned trajectories for the performance of a variety of tasks . It is a computer controlled manipulator consisting of several relatively rigid links connected in series by revolute , spherical , or translational joints . One of these links is typically attached to a supporting base while another link has an end that is free and equipped with a tool known as the end – effector for manipulating objects or performing assembly tasks . Mechanically , the robot is composed of an arm , wrist subassembly and a tool . It is designed to reach a workpiece located within some distance or workspace determined by the maximum and mimimum elongations of the arm .

The dynamic equations of the robot are a set of highly nonlinear coupled differential equations containing a varying inertia term , a centrifugal and Coriolis term , a frictional term , and a gravity term . Movement of the end – effector in a particular trajectory at a particular valocity requires a complex set of torque functions to be applied by the joint actuators of the robot . The exact form of the required functions of actuator torque depend on the spatial and temporal attributes of the path taken by the end –

effector as well as the mass properties of the links and payload , friction in the joints , etc .

The non – linear dynamics governing robot motion presents a challenging control problem . A traditional linear controller control the motion of the robot . A controller based upon the theory of nonlinear control is better suited for the problems of robot manipulation . Unfortunately , nonlinear differential equations are plagued by substantial requirements for computation and have an incomplete theory of solution . Thus , most approaches to robot controller design have suffered due to the complications of nonlinear effects .

Because of these complications , fuzzy logic offers a very promising approach to robot controller design . Fuzzy logic offers a design rules that are relatively easy to use in a wide range of applications , including nonlinear robotic equations . Fuzzy logic also allows for design in cases where models are incomplete , unlike most design techniques . In addition , microprocessor – based fuzzy controllers have performed with data streams of eight bits or less to allow for a simple design .

The dynamic equation of a robotic manipulator can be described by the nonlinear differential equation (1) . For the purpose of simulation a two link planar robotic manipulator is considered (3) .

$$M(\theta)\,\ddot{\theta} + C(\theta,\dot{\theta}) + G(\theta) = \tau \qquad (1)$$

Where ; M is a multi – dimensional matrix of inertia terms
C is a vector of centrifugal and coriolis terms
G is a vector of gravity terms
τ is a vector of joint torques

θ , $\dot{\theta}$ and $\ddot{\theta}$ are the joint angular position , velocity and acceleration terms .

The mathematical expression for a two link robot is given by the following terms :

Inertia matrix :

$$M(\theta) = \begin{bmatrix} a_1 + a_2 \cos\theta_2 & a_3 + 0.5a_2 \cos\theta_2 \\ a_3 + 0.5\, a_2 \cos\theta_2 & a_3 \end{bmatrix} \qquad (2)$$

Coriolis and centrifugal torque vector :

$$C(\theta, \theta) = \begin{bmatrix} (a_2 \sin\theta_2)(\dot{\theta}_1\dot{\theta}_2 + 0.5\dot{\theta}_2^2) \\ (a_2 \sin\theta_2)\, 0.5\dot{\theta}_1^2 \end{bmatrix} \quad (3)$$

Gravity and loading vector :

$$G(\theta) = \begin{bmatrix} (a_4 \cos\theta_1) + a_5 \cos(\theta_1 + \theta_2) \\ a_5 \cos(\theta_1 + \theta_2) \end{bmatrix} \quad (4)$$

The parameters a1 , a2 , a3 , a4 and a5 account for the inertia and gravity that influences the motion nonlinear .

Figure 1. 1-2 shows the simulation results of comparison of fuzzy controller vs. the PD controller . A response to a step input is plotted against time . Figure 1.1 shows the fuzzy controller response and figure 1.2 shows the PD controller response . Simulation shows that fuzzy controller has a rise time of 0.3 seconds , whereas the PD controller has a rise time of 0.4 sec . Fuzzy controller is able to improve the response by reducing the overshoot and at the same time decreasing the rise time . It has been seen that if the gains of the PD controller is increased to improve the rise time , there is an inherent increase in the rise time . Also one set of gains of the gains of the PD controller do not provide the same performance in all the operating ranges .

Fuzzy logic attempts to bypass the difficulties that accompany the solutions to nonlinear differential equations . Fuzzy logic allows us to set – up a controller that is not entirely based on a complete description of the robot .

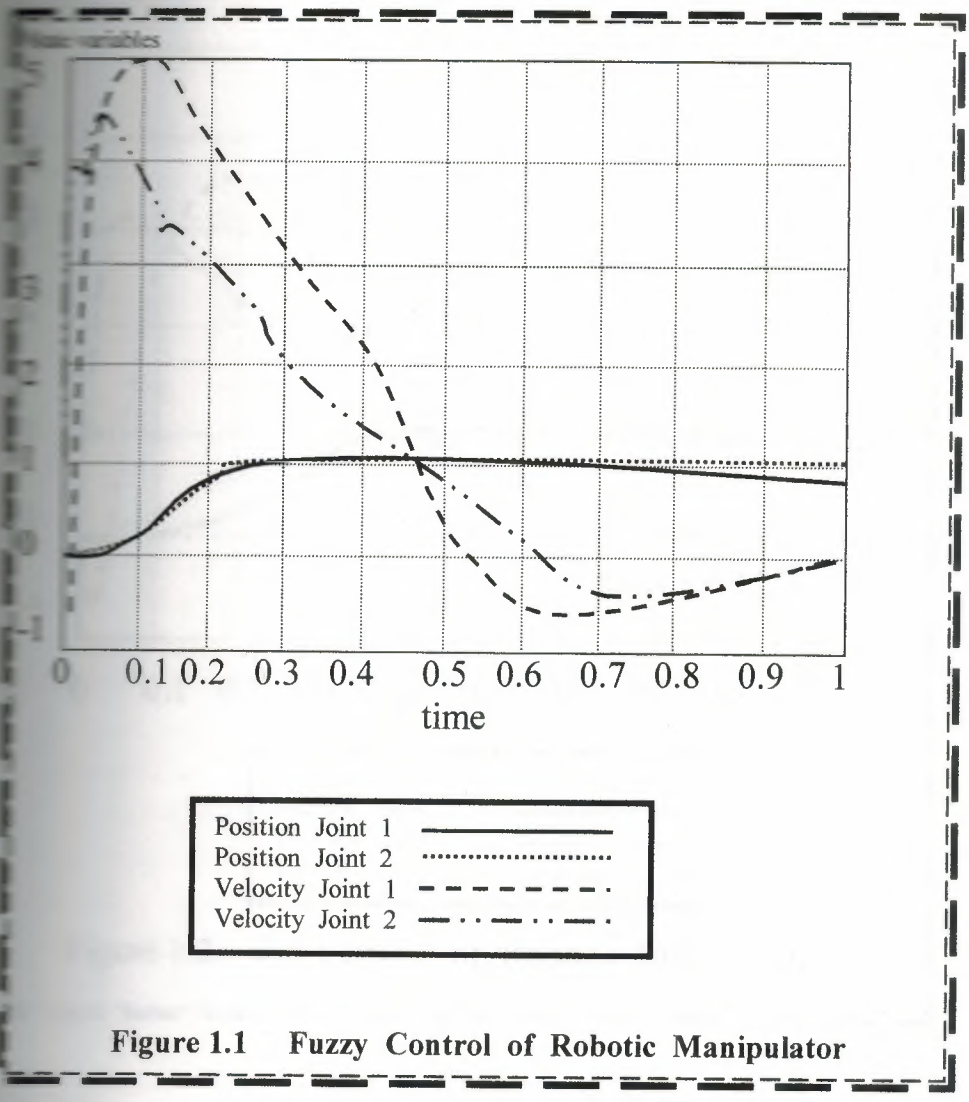


Figure 1.1 Fuzzy Control of Robotic Manipulator

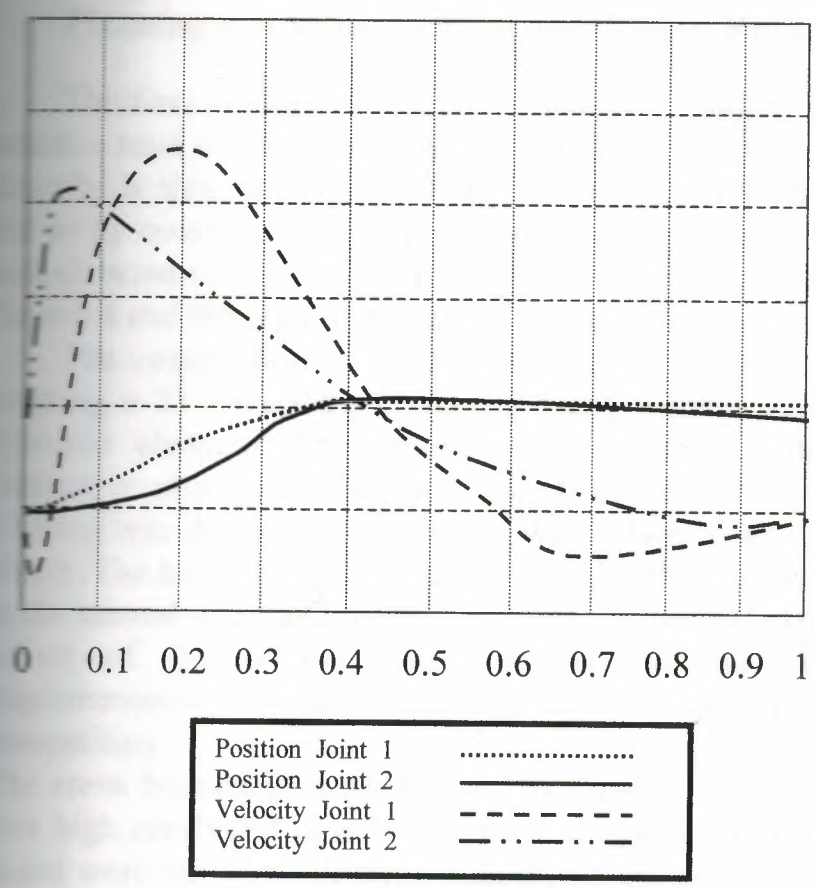


Figure 1.2 PD Control of Robotic Manipulator

1.3 Integrating High-Speed Obstacle Avoidance, Global Path Planning and Vision Sensing on a Mobile Robot

The first mobile robot Competition was held by AAAI in 1992, where a total of ten robots competed in a series of events. This robot describe in this chapter, CARMEL, won this inaugural competition. It did so by combining high-speed sonar-based obstacle avoidance with task-directed vision .in this case study, will give a detail description of the robot and the algorithm that led it to victory.

The competition considers three stages, the first stage required roaming a 22 meters by 22 meters arena while a voiding static and dynamic obstacle. The second stage involved searching for ten distinctive objects in the same arena and visiting for each object. Visiting was defined as moving to within two robot diameters of the object .The last stage was timed race to visit three of the object located in the second stage and return home. Since the first stage is primary a subset of the second stage requirement and the third stage implementation, this case study will focus on the second stage of the competition.

The arena boundaries were defined by three foot high walls and three foot high cardboard boxes were used as obstacles .The objects to be found were ten foot tall three inch diameters poles .To each pole we attached an omni directional barcode tag, which allowed our computer vision system to distinguish one pole from another. The objects could be seen 1.5 meters

CARMEL (computer-aided robotics for maintenance, emergency, and life support) is based on a commercially available Cyber motion K2A mobile robot platform. It is cylindrical robot a bout a meter diameter, standing a bit less than a meter high when equipped with a large hollow shell (for holding electronics and other equipment) on the top. It has top speed of approximately 80 millimeters per second and is driven by three synchronously driven wheels CARMEL's hexagonal top is decoupled from these, so that the orientation of the top is unchanged when the robot itself turns. Wheel encoders calculate the robot's displacement from a homed position; this calculation called dead reckoning. Errors accumulate in the dead-reckoned position because of wheel slippage; dealing with these errors is a major concern.

CARMEL is equipped with a ring of twenty-four ultrasonic sonar sensor evenly disrupted around the robot's torso, each wit a two-meter range and sensing cone of about thirty degrees. A grayscale CCD (change-couple device) camera was added to CARMEL to give

it victual capabilities. The camera on a routing tower, allowing it to turn360 degrees independent of the robot 's orientation.

CARMEL has three computers working cooperatively while the robot is turning. A motor control processor (Z80) receives motor and steering commands from the top-level computer and controls the robot's wheel speed and direction.

This processor also maintains the robot's dead-reckoning information. An IBM PC XT clone is dedicated to the sonar ring, controlling the firing sequence and filtering sonar cross talk and external noise fro the sonar data. Finally, an IBM pc clone running a 33-MHz, 80486-based processor performs the top level function of the robot, including obstacle avoidance, vision processing, planning, and localization. This computer communicates with the sonar and motor control processor s via RS-232 (9600 baud)

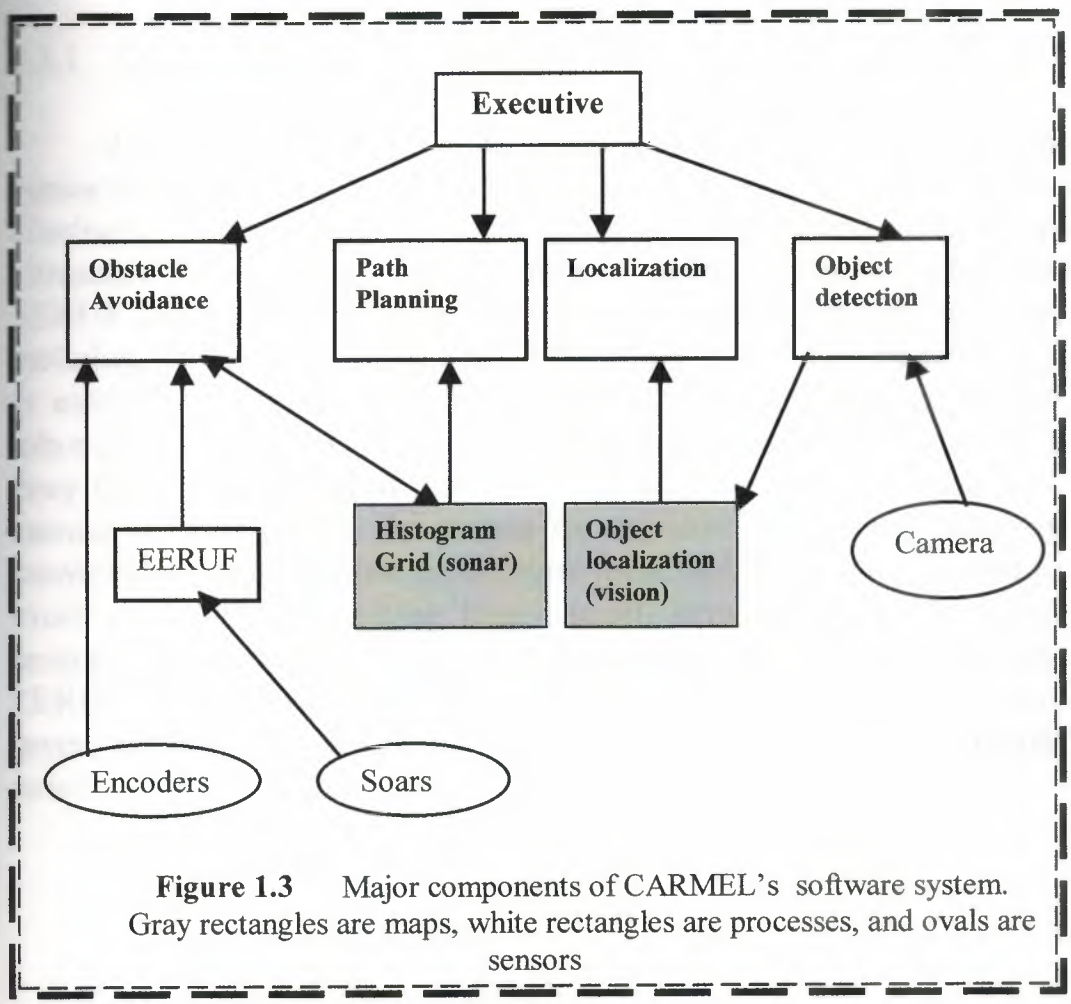


Figure 1.3 Major components of CARMEL's software system. Gray rectangles are maps, white rectangles are processes, and ovals are sensors

This processor also maintains the robot's dead-reckoning information. An IBM PC XT clone is dedicated to the sonar ring, controlling the firing sequence and filtering sonar cross talk and external noise fro the sonar data. Finally, IBM functions of the robot, including obstacle avoidance, vision processing, planning, and localization. This computer communicates with the sonar and motor control processor via RS232 (9600 baud).

CARMEL has hierarchical software structure. At the top level is planning system that decides when to call subordinate modules for movements, vision or recalibration of the robot's position. Each of the subordinate modules is responsible for doing low level error handling and must return control to the planner at the end of asset period of time, perhaps reporting failure at the figure 1.3 showing the major components of CARMEL's software system. Obstacle avoidance, path planning, vision sensing, and localization are described in the following four sections.

1.3.1 Obstacle Avoidance

A key to CARMEL success was its ability to deal with sonar sensor noise. CARMEL used a novel algorithm called EERUF (error-elimination ultrasonic firing) to allow for rapid firing and sampling of ultrasonic sonar sensor, which means faster obstacle avoidance. EERUF allows the robot to detect the end reject ultrasonic noise, including cross talk. The source of ultrasonic noise may be classified as external source, such as ultrasonic sensor used on another mobile robot operating in the same environment ;or internal sources, such as stray from other on-board ultrasonic sensors. The latter phenomenon, known as cross talk, is the reason for the slow firing rates in many conventional mobile robot application: most mobile robots designed to avoid cross talk by waiting long enough between firing individual sensors to allow each echo to dissipate before the next sensor is fired. EERUF,on the other hand ,is able to detect and reject a bout ninety-seven percent of all erroneous reading caused by external or internal noise

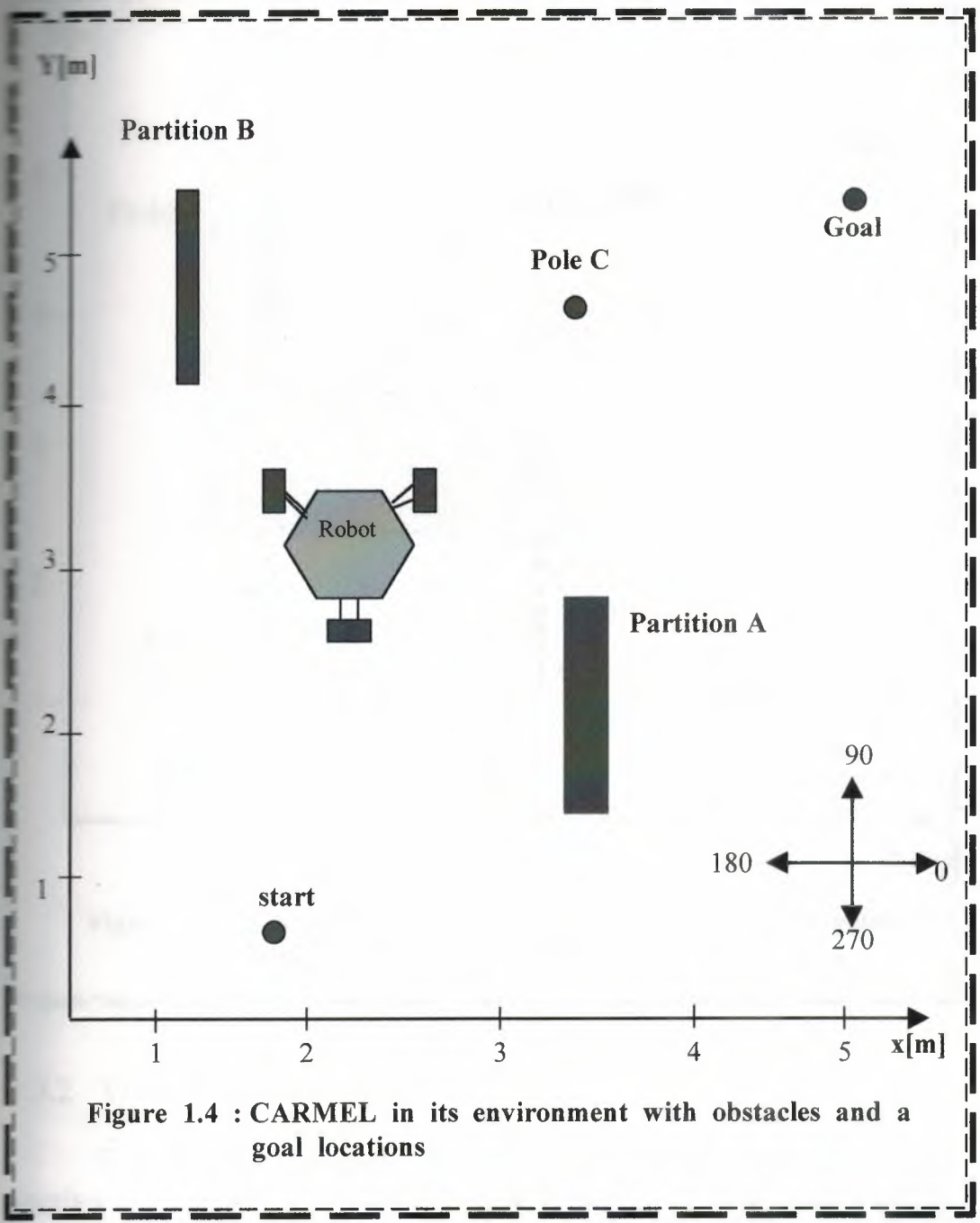


Figure 1.4 : CARMEL in its environment with obstacles and a goal locations

In general, external noise is random and can be detected simply by comparison of consecutive readings Cross talk, however mostly a systematic error Which may cause similar (albeit erroneous) consecutive errors. EERUF overcomes this problem by firing each sensor after individual, alternative delays that disrupt the repetitiveness of cross talk errors, rendering these errors detectable by the method of comparison of consecutive readings. Because EERUF practically eliminates the problem of cross talk, it allows for very fast firing rates.

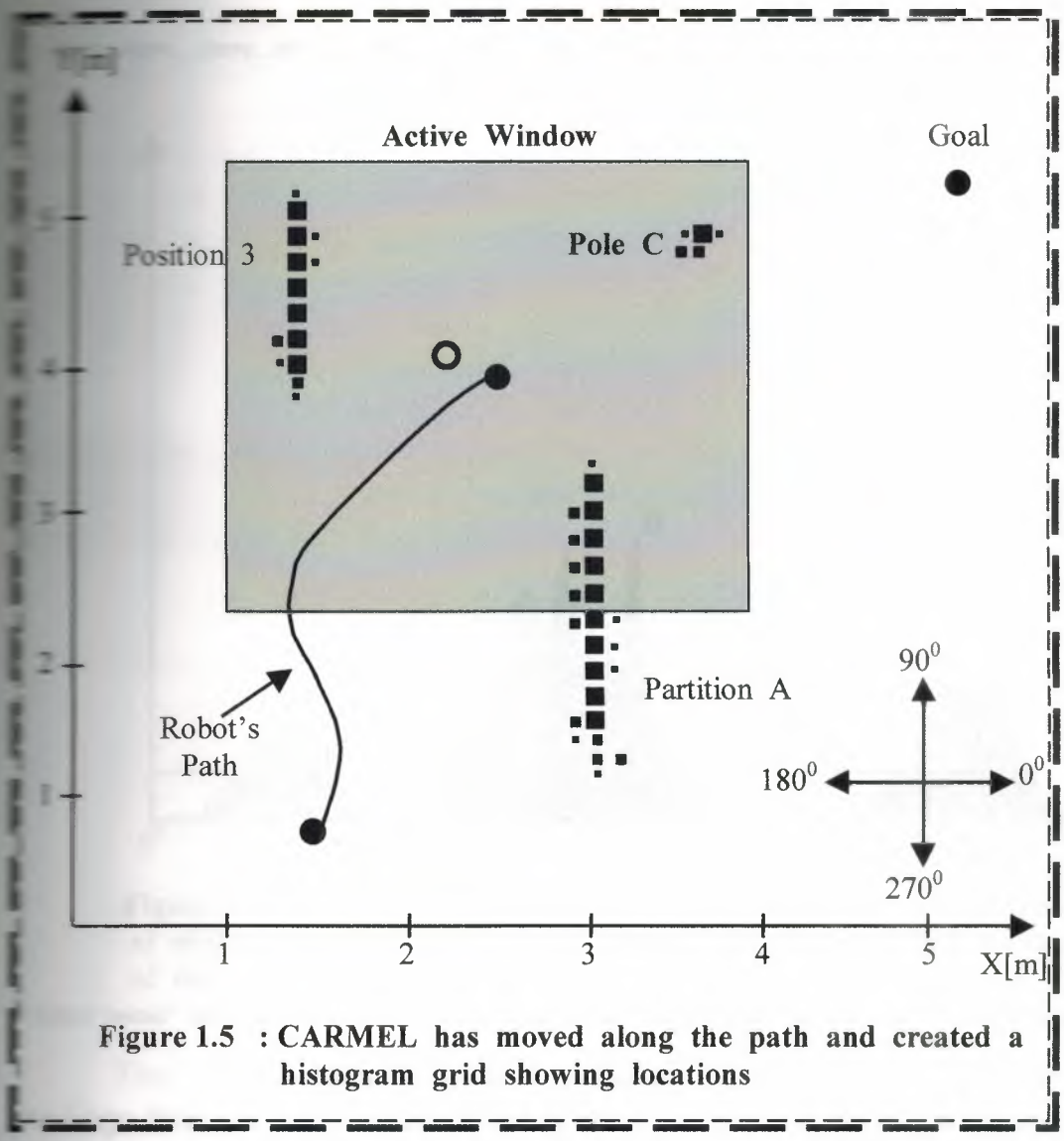


Figure 1.5 : CARMEL has moved along the path and created a histogram grid showing locations

1.3.2 Vector Field Histogram

To map the obstacle in the environment, CARMEL uses another innovative approach called a vector field histogram (VFH). The VFH method uses a two dimensional Cartesian grid, called the histogram grid, to represent data from ultrasonic range sensors. Each cell in the histogram grid holds a certainty value that represents the confidence of the algorithm in the existence of an obstacles and a goal location. The figure 1.4 shows CARMEL in a sample environment with obstacles and a goal action. The figure 1.5 shows the map that CARMEL creates as it moves towards the goal. This representation was derived from the certainty grid concept originally presented in Moravec and Elfes (1985).

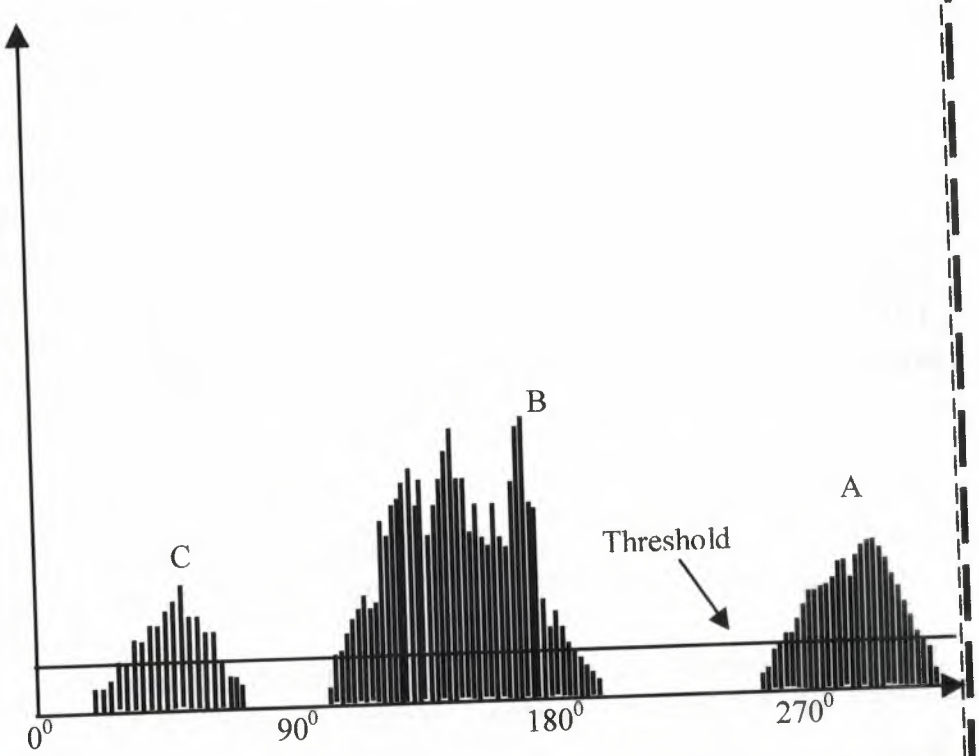


Figure 1.6 VFH's polar histogram has "mountains" in the direction of obstacles . CARMEL is steered toward a "valley" in the direction of the goal location .

The central idea behind a certainty grid is to fuse sonar readings over time to eliminate the error in individual sonar readings. In CARMEL's case, each cell of the grid array represents a ten-centimeter sequence, and the array covers the entire environment space. As the robot (CARMEL) travels through the environment, it's sonar sensors are continuously being fired and returning range readings to objects. Since the approximate location of the robot at any time knows through odometers and the direction of each sonar sensor is known, the location of object in the grid array can be estimated. Each time an object is detected at a particular cell location, the value of the cell is increased and the value of all the cells between the robot and this cell are decremented (since they must be empty). The cell has maximum and maximum values, which have been chosen arbitrarily for computational convenience.

To perform the actual obstacle avoidance using the histogram grid CARMEL creates an intermediate data representation called the polar histogram .The purpose of the polar histogram is to reduce the a

mount of data that needs to be handled for real-time analysis while at the same time retaining the statistical information of the histogram grid, which compensates for the inaccuracies of the ultrasonic sensors .In this way, the VFH algorithm produces sufficiently detailed spatial representation of the robot environment for travel among densely cluttered obstacle, without compromising the system's real-time performance. The spatial representation in the polar histogram can be visualized as a mountainous panorama around the robot, where the height and size of the peaks represent the proximity of obstacles, and the valleys represent possible travel directions (figure 1.6) . The VFH algorithm steers the robot in the direction of one of the valleys based on the direction of the target location .The details for CARMEL algorithm for creating and using the polar histogram .A high level description of the algorithm follows:

1. Collect current sonar sensor readings.
2. Create the histogram grid. For each sonar reading do the following:
    a) Given the direction of the sensor, the distance returned, and the robot's current location, determined the cell in the histogram grid in which the obstacle less. if the sonar did not detect on obstacle ,then determine the cell that lies at the maximum range of the sonar and skip to the next step.
    b) Increment the value in the cell by a fixed amount (we used three in our implementation) up to some maximum (we use fifteen).
    c) For each cell lying on straight line between the robot and the cell determined in (a) above, decrement its value by fixed a mount (we use one) down to minimum of zero.
3. Calculate the polar histogram as follows:
    a) Define an active window surrounding the center of the robot.
    b) For each direction around the robot, in five-degree increments, total the value in all the cells in that direction within the active window.
    c) Smooth the histogram by averaging neighboring direction.
4. Set the threshold such that a polar histogram value below that threshold means that direction is free once it is an above the threshold that means the direction is occupied this Threshold can very greatly between robots and even between environments;

5. Search the polar histogram for a free direction of travel as follows:

a) If the direction to the target if free and enough neighboring histogram direction is free then the target direction is the free direction.

b) Otherwise, begin checking to the left and right of the target direction in such a free segment can be return as the free direction.

c) If no free segment is found, the robot is trapped (i.e., surrounded on all sides by obstacles).

1.3.3 Global Path Planning

In addition to VFH, CARMEL, uses that searches the histogram grid created during obstacle avoidance and create a list of via points (intermediary goal points) that represent the shortest path to the goal .The algorithm simple and fast.

1. Initialize a planning grid of the same size and granularity as the histogram grid
2. Threshold the histogram grid to determine occupied cells for planning purpose mark these cells as occupied cells by the radius of the robot in the planning grid.
3. Expand each occupied cells by the radius of the robot in the planning grid;
4. If the start or target location fall inside on obstacle, move them to the nearest edge of the obstacle.
5. Create a linked list of structures of the type:

```
struct  NODE {          /*coordinates of cell grid*/
int x,y;                /*distance from the target*/
 struct NODE next;      /*forward link*/
};
```

6. Initialize the head of the linked list to the coordinates of the target cell and initialize the distance to zero.
7. While there is something in the list and the start cell has not yet been proceed:

a) Pop the head of the list.

b) Place the distance of item into its corresponding cell (x,y) in the planning grid.

c) Put all neighboring cells that are still 0 and are not obstacles at the end of the linked list with a distance equal to one plus the distance of popped item.

8. Starting at the start cell, following the minimum path (including diagonal elements) to the target cell. If all the neighbors of the start cell are zero then there are path.

9. Select those cells where the path changes and store their coordinates as via points that the robot should try to flows

if the path is not found ,it is possible to iterate the path planner using higher and higher threshold for occupied cells. This algorithm is very quick, taking less that one second to run on a grid of 256 by 256 cells (each cell represents 20 centimeters in the environment).

1.3.4 Vision Sensing

The ability to accurately detect and identify objects in the world was important for earning the maximum number of points, as well as for keeping position and orientation errors within tolerable limits. The competition rules allowed the objects (tall poles) to be tagged to allow for easier recognition. Various tagging schemes were considered, but a vision-based system had an important advantage in its potential for large-range sensing. A major concern was the inherently heavy computation generally required for image processing .By intelligently designing the object tags, we greatly reduce the required computation.

1.3.5 The Algorithm

The Vision Algorithm is composed primary of three routines: scanning a column of an image for objects, merging these objects with those objects found on pervious column, and heuristic for elimination invalid tags and merging multiply identified objects. The algorithm works on the raw image. We do not perform any filtering or preprocessing on the image before running the algorithm.

The works by scanning an image column by column, from the left side to the right side of an image in each column, the algorithm looks for the sharp intensity changes that marks the edges of the black and white bands. Because the tags start with a black band, the algorithm identifies the potential objects whenever it detects a transition from white to black .As the algorithm continues scan the

column /it calculates the high of the black and white potential bands for each potential object.

1.3.6 Localization

A robot internal (dead-reckoned) knowledge of where it is accumulates errors during movement due to wheels slippage, etc. therefore, triangulation known landmark position .The intention was update CARMEL's position and orientation using landmark whose position were recorded earlier in the navigation process, before dead-reckoned errors had accumulated .We implemented and experimented with our own landmark triangulation algorithm. This method is based on the geometric circle method, which largely used in literature.

In the figure 1.7 landmark 1, landmark 2, and the robot from one circle (circle A). Even though the robot location is not known, there are two-position circles because the angle between landmark as viewed by the robot is known. Landmark 2, landmark 3, and the robot also from another unique circle (B), from the information available (landmark locations and the circle angle $\propto and \beta$) the equation of the circle can be determined .the two circle can intersects at landmark 2 and the robot's location is the other interests is follows:

1. Properly order landmark:
   a) For the algorithm to work properly, both α and β must be less than 180 degrees. When this condition holds we can say that the handworks are "Ordered " properly. Properly ordered landmarks assure that the desired solution (out of two solution possible) is found. Properly ordered landmarks have the following two features: first, they are labeled consecutively (1,2,3) in counterclockwise faction; and second, the angle between hand marks 1 and 3(α) must be less than 180 degrees.
2. α =lo2-lo1. If α is too small or equal 90 degrees or 270 degrees return with error because division by zero is occurred.
3. β = lo3-lo2. if β is too small or equal to 90 degrees or 270 degrees return with error because division by zero is occurred.
4. ra = d12/2sin(α)
5. rb = d23/2sin(β)

6. la = d12/2tan(α)
7. lb = d12/2tan(β)

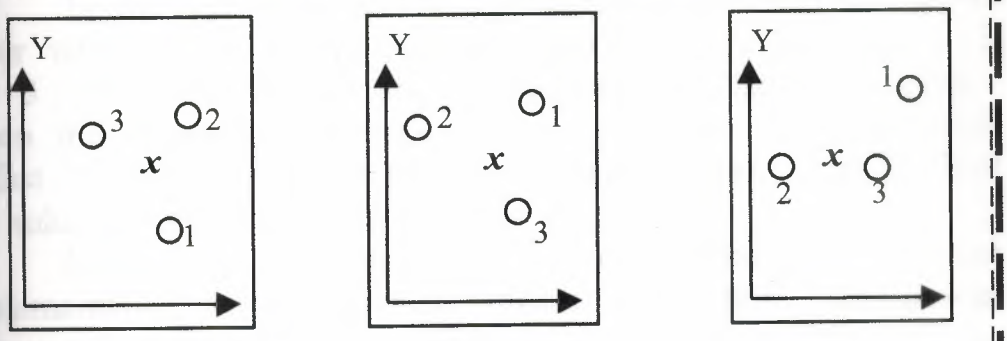


Figure 1.7 Landmarks must be properly ordered for geometric circle algorithm to work correctly ,only (a)shows properly ordered landmark; in (b) the landmarks are not labeled in a counterclockwise fashion . In (c) the angle between landmark 1 and landmark 2 is greater than 180 degrees .

8. Let v12x and v12y be the unit vector from landmark 1 to landmark 2 .
9. Let v23x and v23y be the unit vector from landmark 2 to landmark 3 .
10. cax = p12x – la (v12y)
11. cay = p12y + la (v12x)
12. cbx = p23x – lb (v23y)
13. cby = p23y + lb (v23x)
14. Return an error if the centers of the two circles are too close (we use 10 units) .
15. If y is very large – than return error .
16. Let cbax and cbay be the unit vector from the center of circle B to the center of circle of A .
17. d2r = 2rbsin (γ)
18. c2m = rbcos(γ)
19. Robot x position = Rx = 2mx – Lx2 + 0.5
20. Robot y position = Rx = 2my – Ly2 + 0.5

21. ϕ = arctan (Ly1 – Ry / Lx1 – Rx) , the heading of landmark 1 from the true robot position .
22. If $\phi > 0.0$ then robot orientation error = – (L01 - ϕ) else robot orientation error 360 degrees + ϕ - L01
23. Return with solution .

In the geometric circle intersection algorithm , the errors occur when circle A and circle B overlap ; i.e. , landmarks 1 , 2 and 3 , and the robot all lie on or close to the same circle . When this happens , the method cannot be used . This algorithm is fast , because of the simple vector manipulations used to find the solution , and failure detection is easy .

As a result of a large number of experiments the actual implementation of the geometric circle intersection method was discovered to require the headings of at least two of the landmarks to be greater than 90 degrees and the angular seperation between any pair of landmarks to be greater than 45 degrees . If this was not the case , the robot is moved to a location in which the landmarks fit these restrictions . With these restrictions , the algorithm proved very robust with respect to uncertainty in object headings and locations . Additionally , the result from triangulation can be compared with the current dead – reckoned position , and if the two are widely different then the triangulated value can be discarded .

1.3.7 The Supervising Execution System

The design of our execution system was motivated by several goals . First , and most importantly , we wanted to keep the executive very simple . The hierarchical design eliminated many problems , such as contention for resources between submodules and lack of coordination between submodules . All scheduling was done through the top – level planner . Second , we wanted the overall system to be extremely robust . Because the competition rules prohibited outside interference with the robot after they had started task , the executive needed to handle even catastrophic errors during run – time . Third , we wanted to use existing research as submodules . For example , the obstacle avoidance routines we used had been developed over many years as stand – alone processes . We wanted to integrate them within the framework of the task without substantial modifications . Finally , we wanted develop and test each submodule

independently of the others. This allowed several groups to work in parallel in developing the software system.

What emerged from these design goals was a strict hierarchy of modules with calls to each submodule and information exchange between each submodule being completely controlled by a supervising executive. No submodule was allowed to run by itself, and all submodules were guaranteed to return, even if they returned an error condition.

CHAPTER 2

STRUCTURE OF INTERFACE AND SIGNAL TRANSMISSION

2.1 Signal Characteristics

2.1.1 Analog Signals

The public dial-up service supports analogue signals. Analogue signals are what we encounter every day of our life. Speech is an analogue signal, and varies in amplitude (volume), frequency (pitch), and phase.

The three main characteristics of analogue signals are,

- **Amplitude**
  This is the strength of the signal. It can be expressed a number of different ways (as volts, decibels). The higher the amplitude, the stronger (louder) the signal. The **decibel** (named in honor of Alexander Graham Bell) is a popular measure of signal strength . Table 2.1

Sound Level	Type of Sound
40 db	Normal Speech
90 db	lawn mowers
110 db	shotgun blast
120 db	jet engine taking off
120 db +	rock concerts

Table 2.1

It has been discovered that exposure to sounds greater than 90db for a period exceeding 15 minutes causes permanent damage to hearing. Our ability to hear high notes is affected. As young babies, we have the ability to hear quite high frequencies. This ability reduces as the aging process occurs. It can also be affected by too much noise over sustained periods. **Ringing** in the ears after being exposed to loud noise is an indication that hearing loss may be occurring.

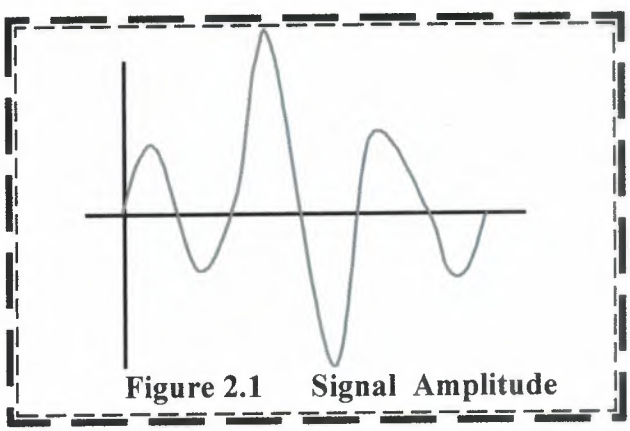


Figure 2.1 Signal Amplitude

The figure 2.1 shows a single signal of various amplitudes. The base line indicates a steady state, in this example, the signal amplitude rises both above and below the steady state.
The measurement of the two extremes is called the **peak to peak** measurement

- **Frequency**
  This is the rate of change the signal undergoes every second, expressed in Hertz (Hz), or cycles per second. A 30Hz signal changes thirty times a second. In speech, we also refer to it as the number of vibrations per second. As we speak, the air is forced out of our mouths, being vibrated by our voice box. Men, on average, tend to vibrate the air at a lower rate than women , thus tend to have deeper voices .

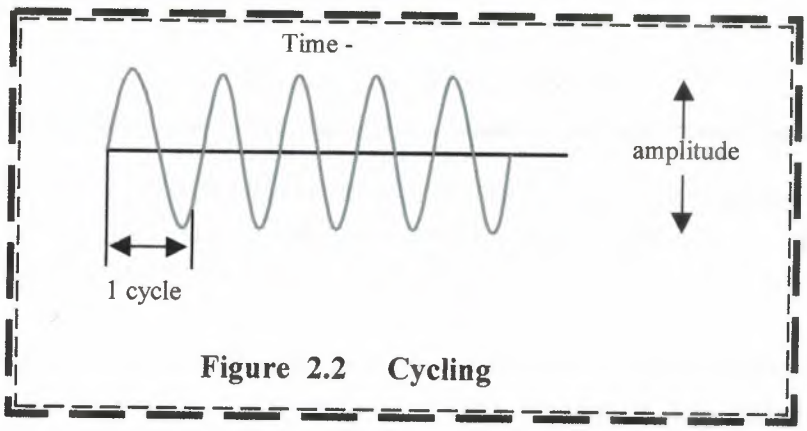


Figure 2.2 Cycling

A cycle is one complete movement of the wave, from its original start position and back to the same point again. The number of cycles (or waves) within a one second time interval is called cycles-per-second, or **Hertz.** (Figure 2.2)
An example is sitting on the beach, counting the waves as they come in on the shore. If we counted the number of waves that crashes on the beach over a minute period, then divided that number by 60 (because there are 60 seconds in one minute), we would have the number of waves per second... (i.e. frequency!).

Humans can hear from reasonably low frequency tones (about 100Hz) all the way up to about 12KHz. As we get older, our ability to hear the higher notes is lessened, due to the aging process making the bones in the ear harder and less able to vibrate. In addition, human speech contains a great deal of redundant information, and can be compacted within the range of 300Hz to 3400Hz whilst still retaining approximately 80% of its content.

- **Phase**
  This is the rate at which the signal changes its relationship to time, expressed as degrees. One complete cycle of a wave begins at a certain point, and continues till the same point is reached again. Phase shift occurs when the cycle does not complete, and a new cycle begins before the previous one has fully interconnected . (Figure 2.3)

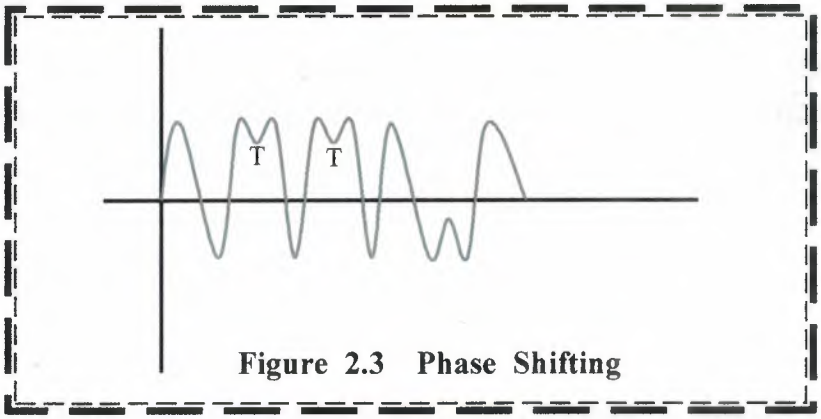


Figure 2.3 Phase Shifting

The human ear is insensitive to phase shift, but data signals are severely affected by it. Phase shift is caused by imperfections in cable media, such as joins and imperfect terminations.

In a practical sense, imagine you are in the bath. If you drop the soap, it forms ripples where the waves travel outwards towards the edge of the bath. When the wave reaches the edge, it hits the wall of the bath and bounces back. This is like phase shift, which is an abrupt change in the signals relationship.

Analogue signals are sent via the PTSN. Digital signals cannot be sent via the PTSN without being first converted to analogue.

2.1.2 Digital Signals

Digital signals are the language of modern day computers. Digital signals comprise only two states. These are expressed as ON or OFF, 1 or 0 respectively. Figure 2.4 Examples of devices having TWO states in the home are,

- Light Switches: Either ON or OFF
- Doors: Either OPEN or CLOSED

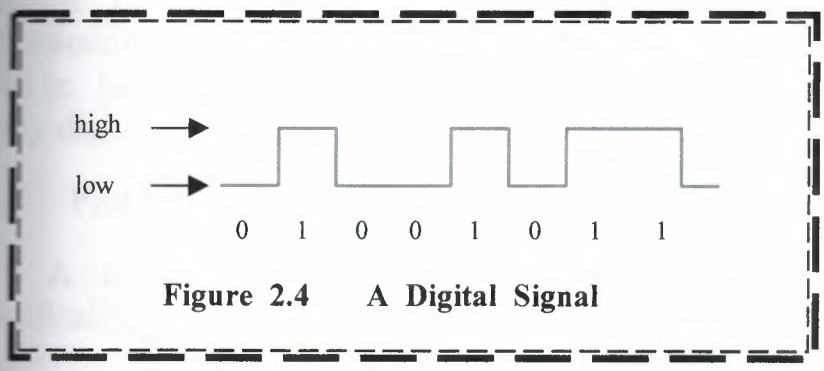


Figure 2.4 A Digital Signal

Digital signals require greater bandwidth capacity than analogue signals, thus are more expensive to communicate. The figure shows a digital signal.

2.1.3 Some Common Terms

- Baud Rate

Baud rate is the reciprocal of the shortest signal element (a measure of the number of line changes which occur every second). For a binary signal of 20Hz, this is equivalent to 20 baud (there are 20 changes per second). For telephone cables, the limiting factor in speed is the number of line changes per second. A line change is defined as switching from one state to another, for instance, switching from a 1 to a 0, or from a 0 to 1 for a digital signal. If the number of line changes per second are exceeded, errors occur and the signal at the receiving end cannot be reliably reconstructed.

- Bits Per Second

This is an expression of the number of bits per second. Where a binary signal is being used, this is the same as the baud rate. When the signal is changed to another form, it will not be equal to the baud rate, as each line change can represent more than one bit (either two or four bits).

Digital signals sent via the PSTN need to be converted to analogue first (by using a device called a modem). Digital signals can be sent via the ISDN unmodified.

- Bandwidth

Bandwidth is the frequency range of a channel, measured as the difference between the highest and lowest frequencies that the channel

supports. The maximum transmission speed is dependant upon the available bandwidth. The larger the bandwidth, the higher the transmission speed. A nominal voice channel has a bandwidth of 3.1KHz. In reality this equates to about 1200bps maximum for a binary digital signal.

2.1.4 Problems of using Voice Channels for Digital Transmission

A digital signal is comprised of a number of signals. Specifically, the signal is represented as follows,

signal = f + f3 + f5 +f7 +f9 +f11 +f13 ....f(infinity)

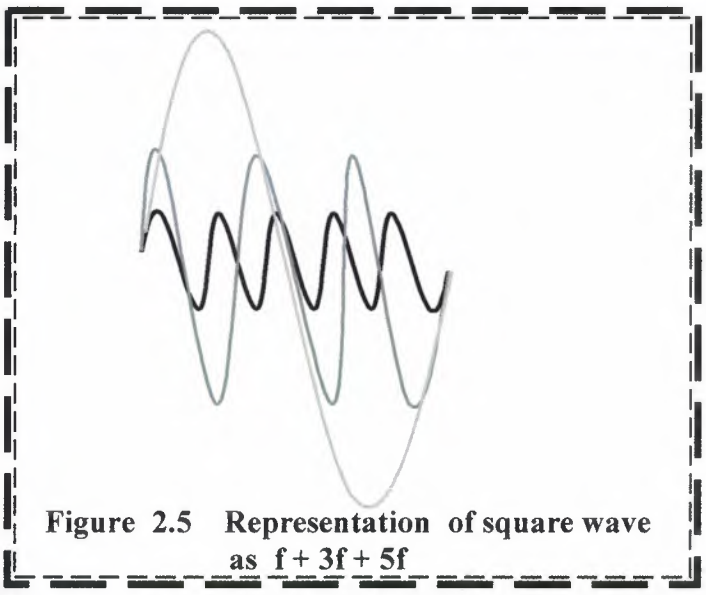


Figure 2.5 Representation of square wave as f + 3f + 5f

In figure 2.5 we can see that a digital signal has a base frequency, plus another at three times the base frequency, plus another at five times the base frequency etc. f3 is called the third harmonic, f5 the fifth harmonic and so on.

The third harmonic is one third of the amplitude of the base frequency (called the fundamental frequency), the fifth harmonic is one fifth the amplitude of the fundamental and so on.

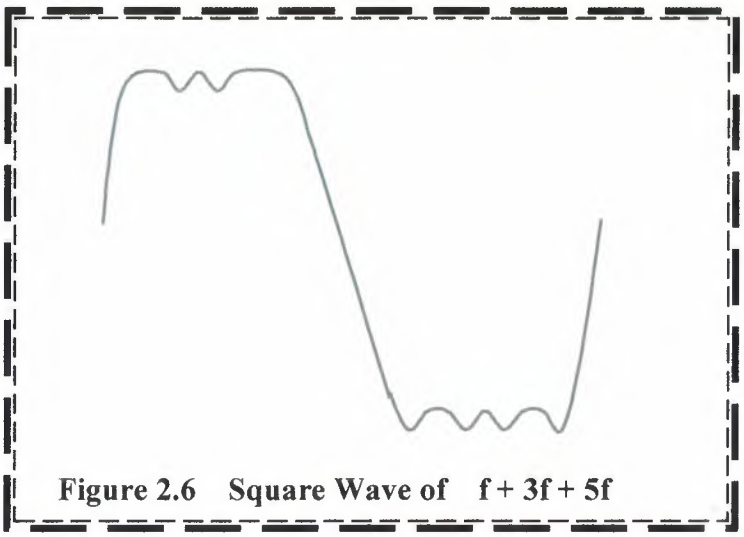


Figure 2.6 Square Wave of f + 3f + 5f

In order to send a digital signal across a voice channel, the bandwidth of the channel must allow the fundamental plus third and fifth harmonic to pass without affecting them too much. (figure 2.6)

As can be seen, this is what such a signal looks like, and is the minimum required to be correctly detected as a digital signal by the receiver. (figure 2.7)

Lets consider sending a 2400bps binary digital signal down a voice channel rated with a bandwidth of 3.1KHz. The base frequency of the digital signal is 1200Hz (it is always half the bit rate), so the fundamental frequency will pass through the channel relatively unaltered. The third harmonic is 3600Hz, which will suffer attenuation and arrive severely altered (if at all). The fifth harmonic has no chance of passing the channel.

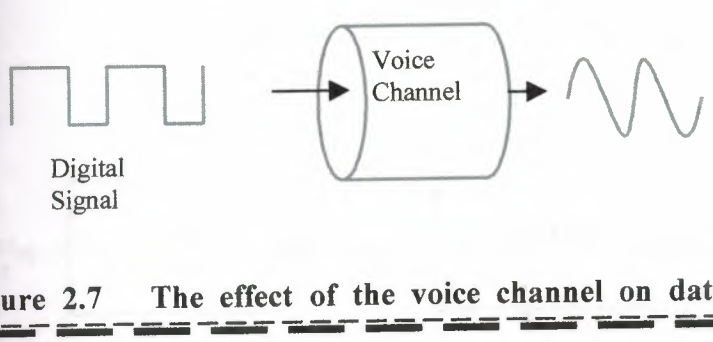


Figure 2.7 The effect of the voice channel on data signals

In this case it can be seen that only the base frequency will arrive at the end of the channel. This means the receiver will not be able to reconstruct the digital signal properly, as it will require f3 and f5 for proper reconstruction.
This results in errors in the detection process by the receiver.

Structure and Elements of Communication System

Electrical Communication Systems are designed to send messages or information from a source that generates the messages to one or more destinations . In general, a communication system can be represented by the functional block diagram shown in figure 2.8.

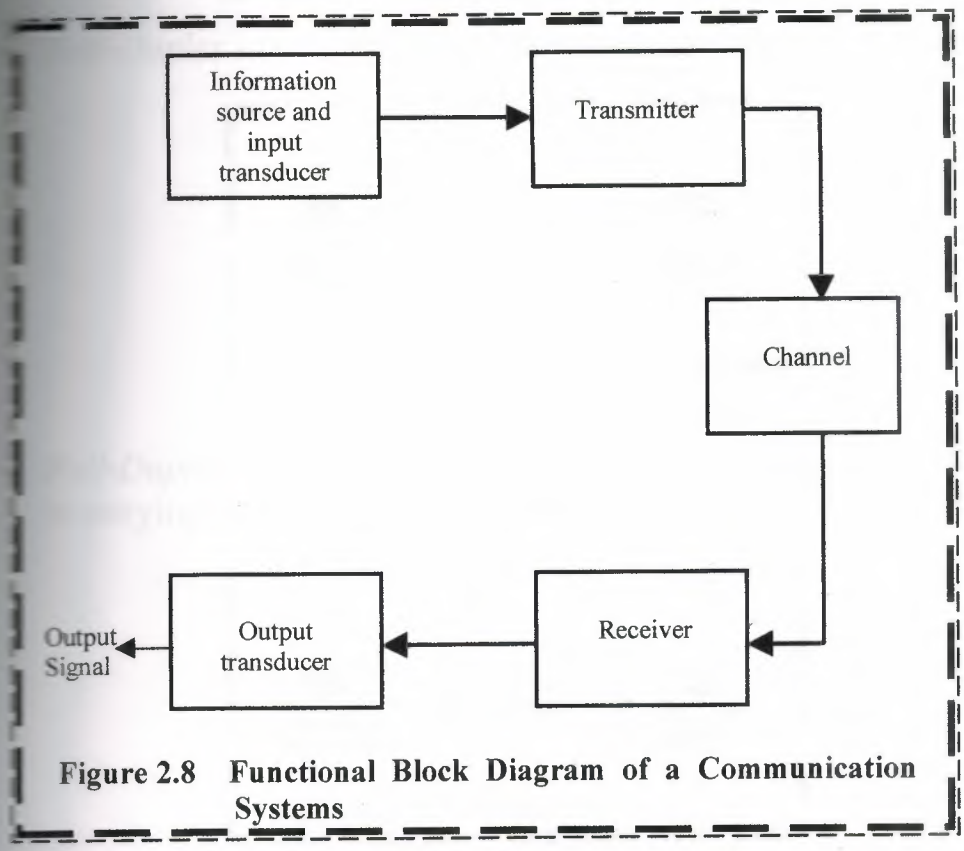


Figure 2.8 Functional Block Diagram of a Communication Systems

Transmitter : The transmitter converts the electrical signal into a form that is suitable for transmission through the physical channel or transmission medium . The transmitter must translate the information signal to be transmitted into the appropriate frequency range that matches the frequency allocation assigned the transmitter .

Transmission can be:

- *Simplex* - signals only in one direction; one station transmitter, other receiver; e.g. radio (figure 2.9)

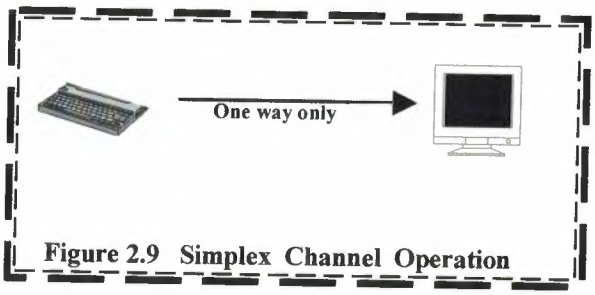


Figure 2.9 Simplex Channel Operation

- *Half-Duplex* - both stations can transmit, but only one at a time

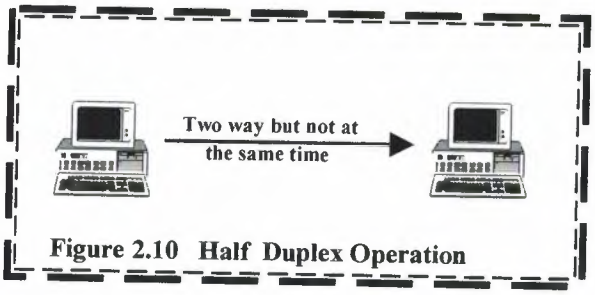


Figure 2.10 Half Duplex Operation

- *Full-Duplex* - both stations can transmit at same time i.e. medium is carrying signals in both directions at same time (figure 2.11)

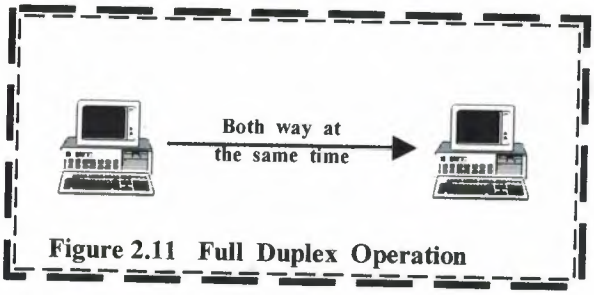


Figure 2.11 Full Duplex Operation

In general , the transmitter performs the matching of the message signal to the channel by a process called modulation .

Modulation: systematic variation of some attribute of carrier signal - e.g. amplitude, phase or frequency.

- *Amplitude Modulation (AM) :* It is sensitive to power fluctuations .For example In AM Radio broadcast , the information signal that is transmitted is contained in the amplitude variations of the sinusoidal carrier , which is the centre frequency in the frequency band allocated to the radio transmitting station . (figure 2.12)

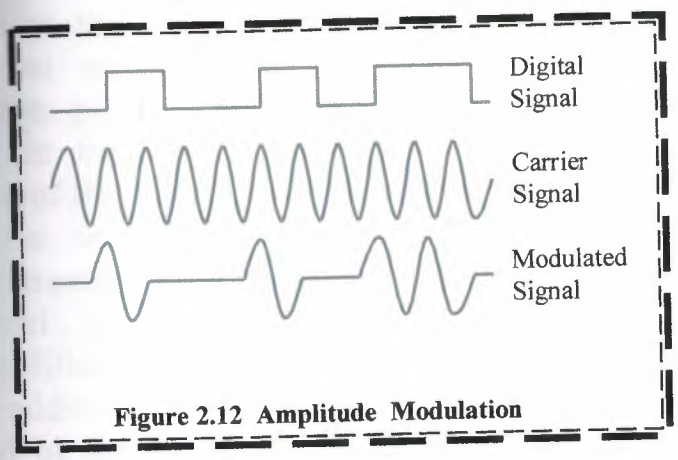


Figure 2.12 Amplitude Modulation

- *Frequency Modulation (FM) :* It allows *full duplex* communication . It needs four frequencies . In FM Radio broadcast , the information signal that is transmitted is contained in the frequency variations of the sinusoidal carrier . (figure 2.13)

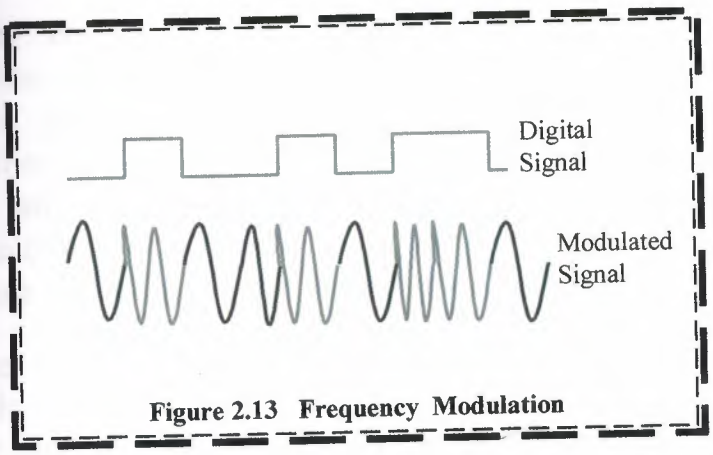


Figure 2.13 Frequency Modulation

- *Phase Modulation (PM)* : It is yet a third method for impressing the information signal on a sinusoidal carrier . It is more sophisticated . (Table 2.2)

Bit Pattern	Degrees Phase Shift
00	45^0
01	135^0
10	315^0
11	225^0

Table 2.2

In general , carrier modulation such as AM , FM and PM is performed at the transmitter to convert the information signal to a form that matches the characteristics of the channel . Thus through the process of modulation , the information signal is translated in frequency to match the allocation of the channel . The choice of the type of modulation is based on several factors , such as the amount of bandwidth allocated , the types of noise and interference that the signal encounters in transmission over the channel , and the electronic devices that are available for signal amplification prior to transmission .

In addition to modulation , other functions that are usually performed at the transmitter filtering of the information – bearing signal , amplification of the modulated signal , and in the case of wireless transmission , radiation of the signal by means of a transmitting antenna .

2.3 Channel Organizations

The communication channel provides the connection between the transmitter and the receiver . The physical channel may be a pair of wires that carry the electrical signal , or an optical fiber that carries the information on a modulated light beam , or an underwater ocean channel in which the information is transmitted acoustically , or free space over which the information – bearing signal is radiated by use of antenna.

One common problem in signal transmission through any channel is additive noise .In general , additive noise is generated internally by components such as resistors and solid – state devices used to implement the communication system . This is sometimes called thermal noise .Other sources of noise and interference may arise externally to the system , such as interference from other users of the channel .

The effects of noise may be minimized by increasing the power in the transmitted signal . However , equipment and other practical constraints limit the power level in the transmitted signal . Another basic limitation is the available channel bandwidth . A bandwidth constraint is usually due to the physical limitations of the medium and the electronic components used to implement the transmitter and the receiver . These two limitations result in constraining the amount of data that can be transmitted reliably over any communication channel .

Data may be transmitted between two points in two different ways. Lets consider sending 8 bits of digital data (1 byte). These bits may be sent all at once (*in parallel*), or one after the other (*serial*).

- **Parallel**
  Each bit uses a separate wire. If there is eight bits sent at a time, this will require 8 wires, one for each data bit. The organisation looks like,

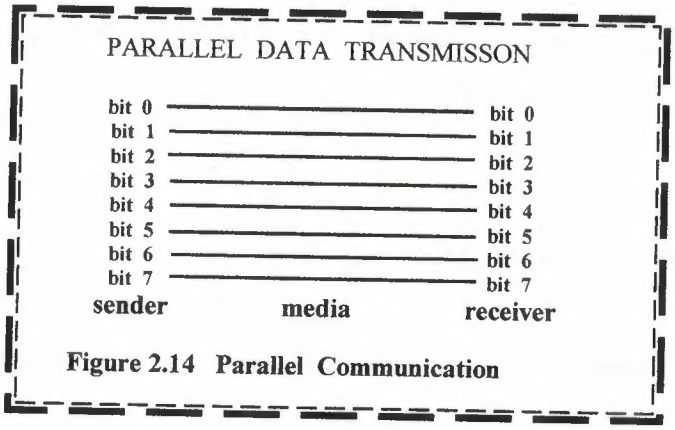


Figure 2.14 Parallel Communication

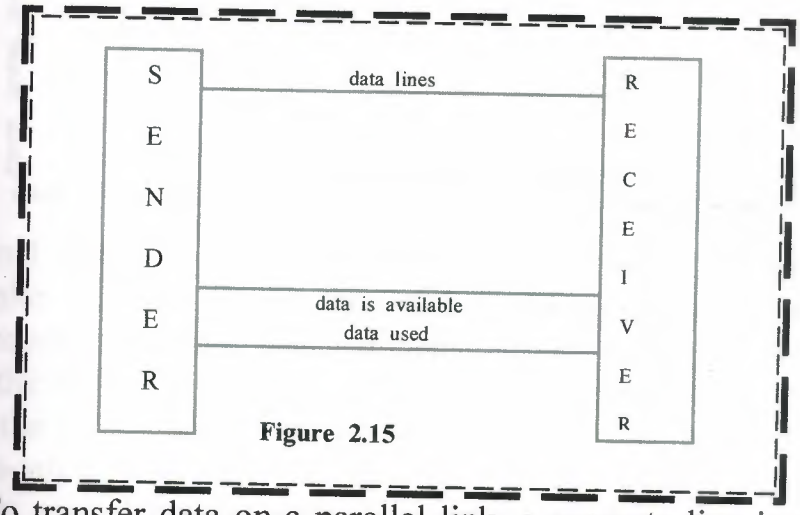


Figure 2.15

To transfer data on a parallel link, a separate line is used as a clock signal. This serves to inform the receiver when data is available. In addition, another line may be used by the receiver to inform the sender that the data has been used, and its ready for the next data.

In the figure 2.15, the sender places the data on the data lines, then signals the receiver that data is available by asserting (sending a pulse) on the **DA** (Data is available) line. After reading the state of the data lines, the receiver signals back to the sender that it has processed the data and is now ready for some more data by asserting the **DU** (Data used) line. The sender, upon getting the DU signal, removes the data and sends the next data element in the same manner.

This exchange of signals such as DA and DU between the sender and the receiver is called a **handshake**. These handshake signals allow the sender and receiver to keep synchronised (work on the same data at the same time in the proper sequence).
Parallel transmission is obviously faster than serial, because more than one bit is sent at a time. Parallel transmission is good only for short links, and examples are found in all computers. The address, data and control buses which interface the processor to other peripherals inside the computer are all parallel buses. In addition, most printers on PC's (LPT1/LPT2) use a parallel interface, commonly called the *Centronics interface*.

- **Serial**
  Each bit is sent over a single wire, one after the after. The organisation looks like,

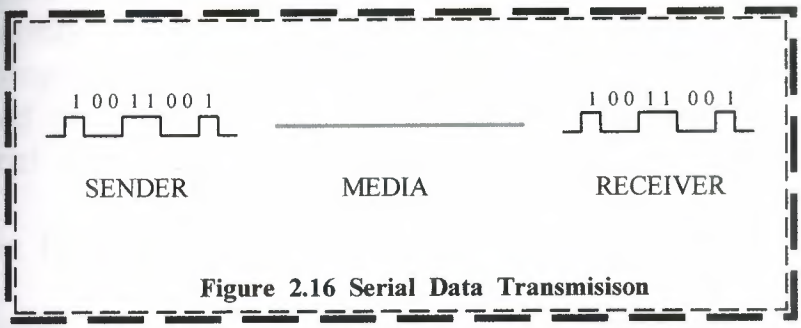


Figure 2.16 Serial Data Transmisison

No signal lines are used to convey clock (timing information) and handshake signals. There are two methods (asynchronous and synchronous) in which timing information is encoded with the signal so that the sender and receiver are synchronised (working on the same data at the same time). If no clock information was sent, the receiver can mis-interpret the arriving data (due to bits being lost, going too slow). In asynchronous, each character is synchronised using a start and stop signal. In synchronous, each group or block of characters is synchronised using a synchronise flag.

2.3.1 Characteristics of Communication Channels

Wireline Channels :
The telephone network makes extensive use of wire lines for voice signal transmissions , as well as data and video transmission . Twisted Pair & Coaxial Cable are basically guided electromagnetic channels which provide relatively modest bandwidth .

Signals transmitted through such channels are distoted in both amplitude and phase and further corrupted by additive noise . Twisted – pair wireline channels are also prone to crosstalk interference from physically adjacent channels . Because wireline channels carry a large percentage of our daily communications around the country and the world . You can see the figure of frequency range of guided wire channel in figure 2.17 .

Fiber Optic Channels :

Optical fibers offer the communications system designer a channel bandwidth that is several orders of magnitude larget than coaxial cable channels .
The transmitter or modulator in a fiber optic communication system is a light source , either a light – emmitting diode (LED) or a laser . Information is transmitted by varying (modulating) the intensity of the light source with the message signal . The light propogates through the fiber as a light wave and is amplified periodically along the transmission path to compensate for signal attenuation .

Wireless Electromagnetic Channels :

In wireless communication systems , electromagnetic energy is coupled to the propagation medium by an antenna which serves as the radiator . The physical size and the configuration of the antenna depend primarily on the frequency of operation . To obtain efficient radiation of electromagnetic energy , the antenna must be longer than 1 / 10 of the wavelength . In figure 2.18 you can see the various frequency bands of the electromagnetic spectrum . The mode of propogation of electromagnetic waves in atmosphere and in free space may be subdivided into three categories , namely , ground – wave propagation , sky – wave propagation and line-of-sight propagation (LOS) .

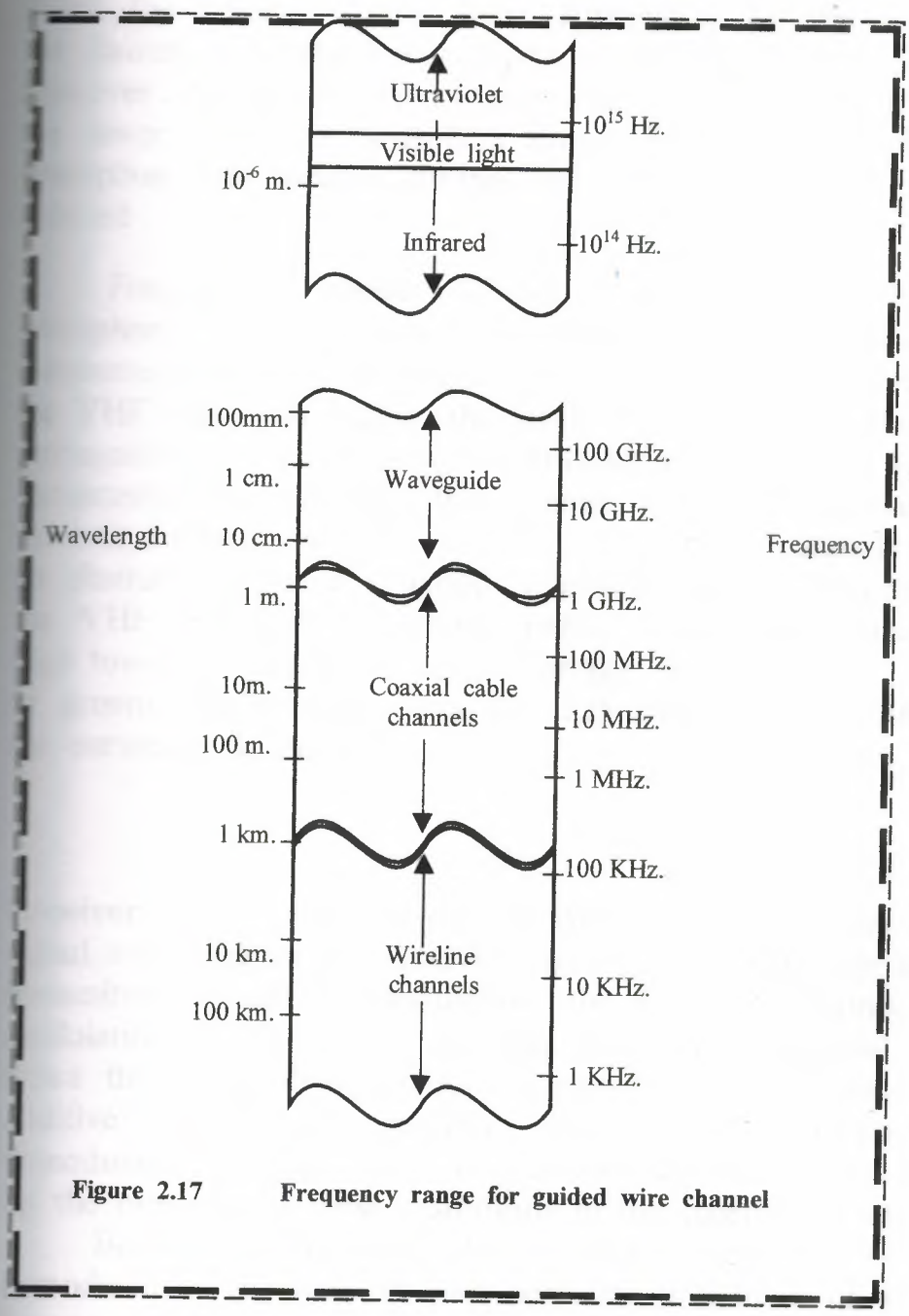


Figure 2.17 **Frequency range for guided wire channel**

Ground – wave propagation (figure 2.19) is the dominant mode of propagation for frequencies in the MF band (0.3 to 3 MHz.) . This is the frequency band used for AM broadcasting and maritime radio broadcasting .

Sky – wave propagation (figure 2.20) results from transmitted signals being reflected from the Ionospere , which consists of several layers of charged particles ranging in altitude from 30 to 250 miles above the surface of the earth . During the day – time

hours , the heating of the lower atmosphere by the sun causes the formation of the lower layers at altitudes below 75 miles . However , during the night – time hours , the electron density in the lower layers of ionosphere drops sharply and the frequency absorption that occurs during the day – time is significantly reduced .

Frequencies above 30 MHz. Propogate through the ionosphere with relatively little loss and make satellite and extraterresterial communications possible . Hence , at frequencies in the VHF band and higher , the dominant mode of electromagnetic propagation is line – of – sight (LOS) propogation . For terresterial communications systems , this means that the transmitter and receiver antennas must be in direct LOS with relatively little or no obstruction . For this reason , television stations transmitting in the VHF and UHF frequency bands mount their antennas on high tower to achieve a broad coverage area .
In general , the coverage area for LOS propagation is limited by the curvature of the earth .

Receiver : The function of the receiver is to recover the message signal contained in the received signal . If the message signal is transmitted by carrier modulation , the recevier performs carrier modulation to extract the message from the sinusoidal carrier . Since the signal demodulation is performed in the presence of additive noise and possibly other signal distortion , the demodulated message signal is generally degraded to some extent by the presence of these distortions in the received signal .

Besides performing the primary function of signal demodulation , the recevier also performs a number of peripheral functions , including signal filtering and noise suppression .

2.3.2 Terminology

Transmission Media: guided or unguided as electromagnetic waves

- *Guided :* guided along physical path, e.g. twisted pair, coaxial cable, optical fiber
- *Unguided Media :* transmit waves but do not guide them, e.g. propogation through air,vacuum, sea water

Data Transmission : Clock Drift dealt with by asynchronous transmission with stop & start bit; or synchronous can stay in synch for length of data

- Asynchronous : e.g. , ASCII character transmission between dumb terminal and host terminal . Asynchronous systems send data bytes between the sender and receiver by packaging the data in an envelope. This envelope helps transport the character across the transmission link that separates the sender and receiver. The transmitter creates the envelope, and the receiver uses the envelope to extract the data. Each character (data byte) the sender transmits is preceded with a start bit, and suffixed with a stop bit. These extra bits serve to synchronise the receiver with the sender.

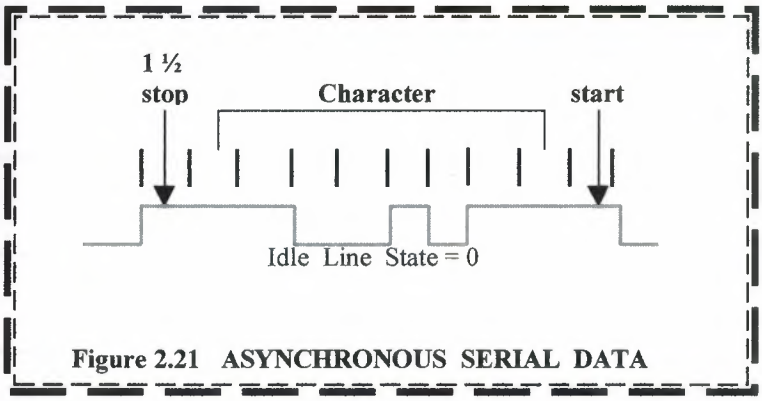


Figure 2.21 ASYNCHRONOUS SERIAL DATA

- Synchronous : "Byte – oriented scheme " ; e.g. , BISYNC

In asynchronous transmission, if there was no data to transmit, nothing was sent. We relied on the start bit to start the motor and thus begin the preparation to decode the incoming character. However, in synchronous transmission, because the start bit has been dropped, the receiver must be kept in a state of readiness. This is achieved by sending a special code by the transmitter whenever it has no data to send.

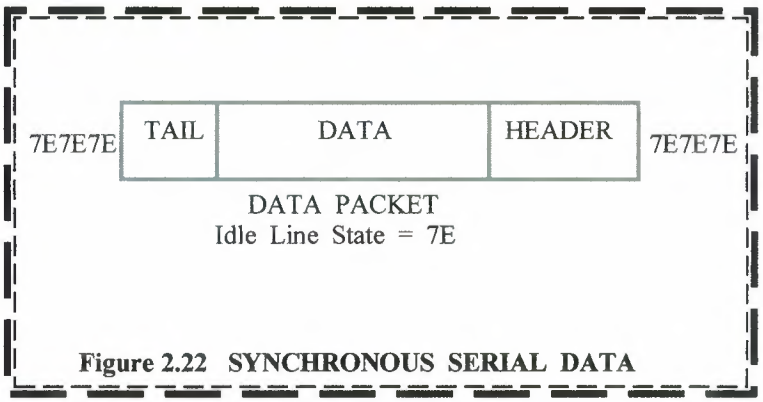


Figure 2.22 SYNCHRONOUS SERIAL DATA

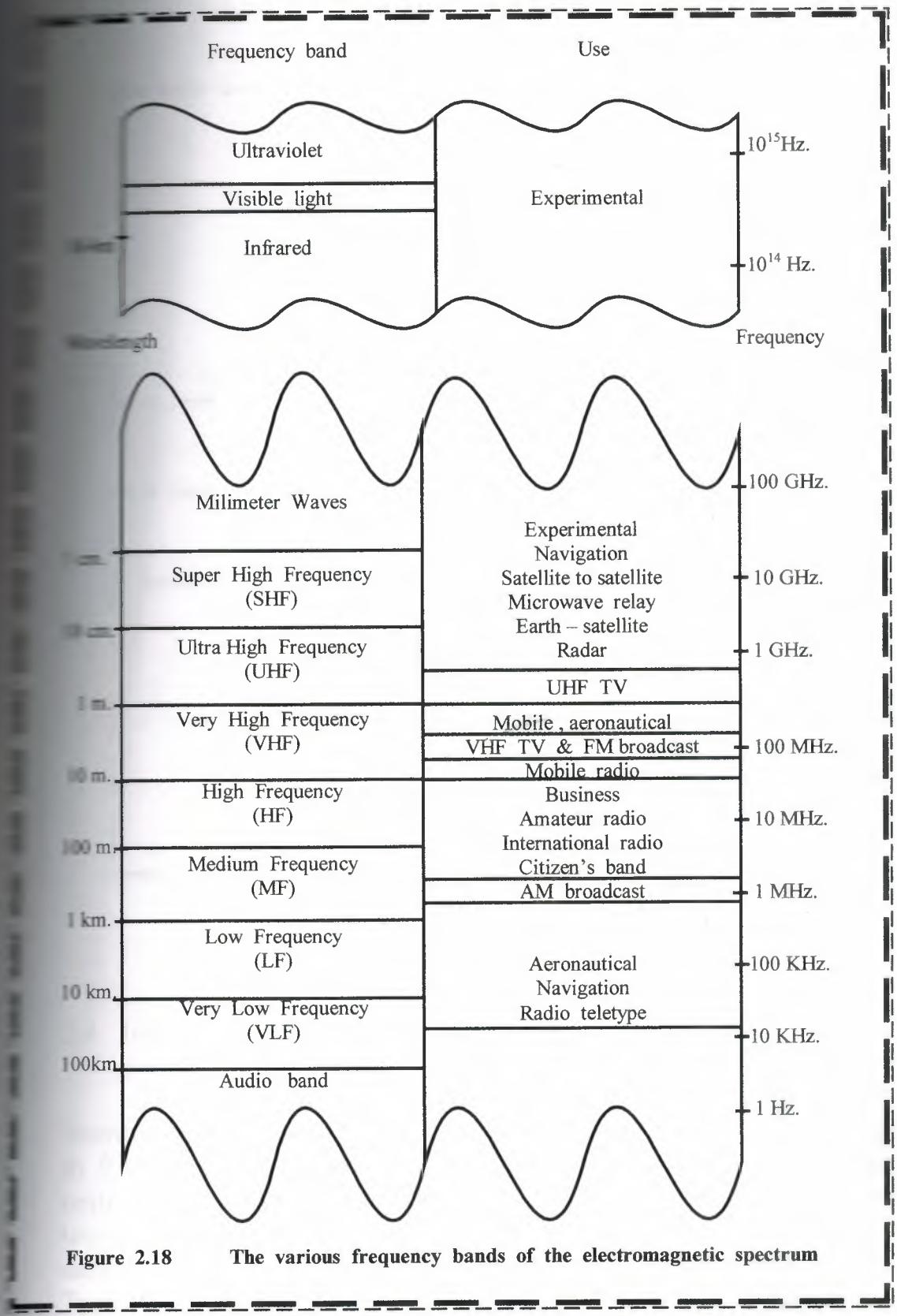


Figure 2.18 **The various frequency bands of the electromagnetic spectrum**

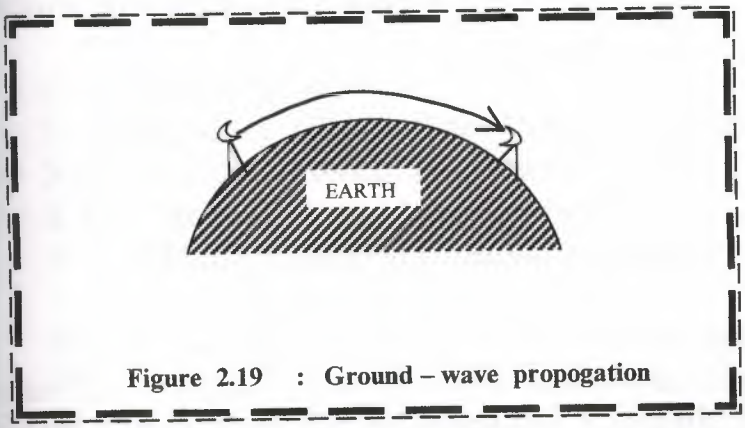


Figure 2.19 : Ground – wave propogation

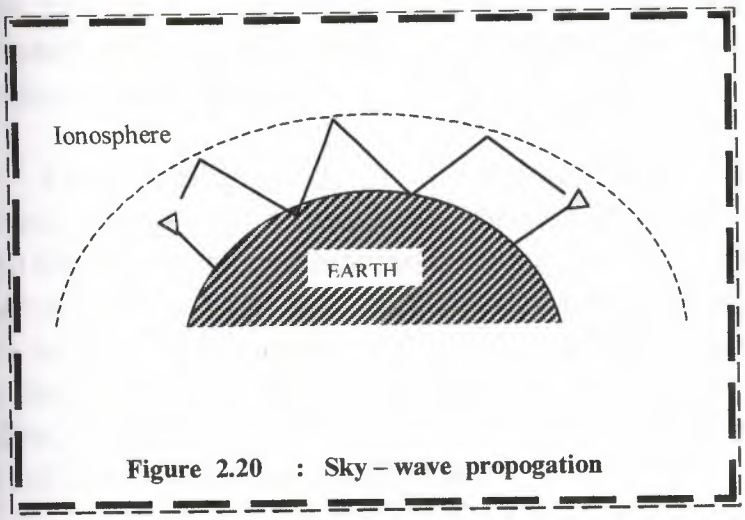


Figure 2.20 : Sky – wave propogation

2.4 Interface by Parallel Ports

The Parallel Port is the most commonly used port for interfacing home made projects. This port will allow the input of up to 9 bits or the output of 12 bits at any one given time , thus requiring minimal external circuitry to implement many simpler tasks. The port is composed of 4 control lines, 5 status lines and 8 data lines . It's found commonly on the back of your PC as a D-Type 25 Pin female connector. There may also be a D-Type 25 pin male connector. This will be a serial RS-232 port and thus, is a totally incompatible port.

Newer Parallel Port's are standardised under the IEEE 1284 standard first released in 1994. This standard defines 5 modes of operation which are as follows,

1. Compatibility Mode.
2. Nibble Mode. (Protocol not Described in this Document)
3. Byte Mode. (Protocol not Described in this Document)
4. EPP Mode (Enhanced Parallel Port).
5. ECP Mode (Extended Capabilities Mode).

The aim was to design new drivers and devices which were compatible with each other and also backwards compatible with the Standard Parallel Port (SPP). Compatibility, Nibble & Byte modes use just the standard hardware available on the original Parallel Port cards while EPP & ECP modes require additional hardware which can run at faster speeds, while still being downwards compatible with the Standard Parallel Port.

Compatibility mode or "Centronics Mode" as it is commonly known, can only send data in the forward direction at a typical speed of 50 kbytes per second but can be as high as 150+ kbytes a second. In order to receive data, you must change the mode to either Nibble or Byte mode. Nibble mode can input a nibble (4 bits) in the reverse direction. E.g. from device to computer. Byte mode uses the Parallel's bi-directional feature (found only on some cards) to input a byte (8 bits) of data in the reverse direction.

Extended and Enhanced Parallel Ports use additional hardware to generate and manage handshaking. To output a byte to a printer (or anything in that matter) using compatibility mode, the software must,

1. Write the byte to the Data Port.
2. Check to see is the printer is busy. If the printer is busy, it will not accept any data, thus any data which is written will be lost.
3. Take the Strobe (Pin 1) low. This tells the printer that there is the correct data on the data lines. (Pins 2-9)
4. Put the strobe high again after waiting approximately 5 microseconds after putting the strobe low. (Step 3)

This limits the speed at which the port can run at. The EPP & ECP ports get around this by letting the hardware check to see if the printer is busy and generate a strobe and /or appropriate handshaking. This means only one I/O instruction need to be performed, thus increasing the speed. These ports can output at around 1-2 megabytes per second. The ECP port also has the advantage of using DMA channels and FIFO buffers, thus data can be shifted around without using I/O instructions.

2.4.1 Hardware Properties

Below is a table of the "Pin Outs" of the D-Type 25 Pin connector and the Centronics 34 Pin connector. The D-Type 25 pin connector is the most common connector found on the Parallel Port of the computer, while the Centronics Connector is commonly found on printers. The IEEE 1284 standard however specifies 3 different connectors for use with the Parallel Port. The first one, 1284 Type A is the D-Type 25 connector found on the back of most computers. The 2nd is the 1284 Type B, which is the 36 pin Centronics Connector, found on most printers.

IEEE 1284 Type C however, is a 36 conductor connector like the Centronics, but smaller. This connector is claimed to have a better clip latch, better electrical properties and is easier to assemble. It also contains two more pins for signals, which can be used to see whether the other device connected, has power. 1284 Type C connectors are recommended for new designs, so we can look forward on seeing these new connectors in the near future.

The table 2.3 table uses "n" in front of the signal name to denote that the signal is active low. e.g. nError. If the printer has occurred an error then this line is low. This line normally is high, should the printer be functioning correctly. The "Hardware Inverted" means the signal is inverted by the Parallel card's hardware. Such an example is the Busy line. If +5v (Logic 1) was applied to this pin and the status register read, it would return back a 0 in Bit 7 of the Status Register.

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/out	Register	Hard ware Inver ted
1	1	nStrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out / Paper-End	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	nInitialize	In/Out	Control	
17	36	nSelect-Printer / nSelect-In	In/Out	Control	Yes
18 - 25	19-30	Ground	Gnd		

Table 2.3 Pin Assignments of the D-Type 25 pin Parallel Port Connector.

The output of the Parallel Port is normally TTL logic levels. The voltage levels are the easy part. The current you can sink and source varies from port to port. Most Parallel Ports implemented in ASIC, can sink and source around 12mA. However these are just some of the figures taken from Data sheets, Sink/Source 6mA, Source 12mA/Sink 20mA, Sink 16mA/Source 4mA, Sink/Source 12mA. As

you can see they vary quite a bit. The best bet is to use a buffer, so the least current is drawn from the Parallel Port.

2.4.2 Port Addresses

The Parallel Port has three commonly used base addresses. These are listed in table 2, below. The 3BCh base address was originally introduced used for Parallel Ports on early Video Cards. This address then disappeared for a while, when Parallel Ports were later removed from Video Cards. They has now reappeared as an option for Parallel Ports integrated onto motherboards, upon which their configuration can be changed using BIOS.

LPT1 is normally assigned base address 378h, while LPT2 is assigned 278h. However this may not always be the case as explained later. 378h & 278h have always been commonly used for Parallel Ports. The lower case h denotes that it is in hexadecimal. These addresses may change from machine to machine.

Address	Notes:
3BCh - 3BFh	Used for Parallel Ports which were incorporated on to Video Cards - Doesn't support ECP addresses
378h - 37Fh	Usual Address For LPT 1
278h - 27Fh	Usual Address For LPT 2

Table 2.4 Port Addresses

When the computer is first turned on, BIOS (Basic Input/Output System) will determine the number of ports you have and assign device labels LPT1, LPT2 & LPT3 to them. BIOS first looks at address 3BCh. If a Parallel Port is found here, it is assigned as LPT1, then it searches at location 378h. If a Parallel card is found there, it is assigned the next free device label. This would be LPT1 if a card wasn't found at 3BCh or LPT2 if a card was found at 3BCh. The last port of call, is 278h and follows the same procedure than the other two ports. Therefore it is possible to have a LPT2 which is at 378h and not at the expected address 278h.

What can make this even confusing, is that some manufacturers of Parallel Port Cards, have jumpers which allow you to set your Port

LPT1, LPT2, LPT3. Now what address is LPT1? - On the majority of cards LPT1 is 378h, and LPT2, 278h, but some will use 3BCh as LPT1, 378h as LPT1 and 278h as LPT2. Life wasn't meant to be easy. The assigned devices LPT1, LPT2 & LPT3 should not be a worry to people wishing to interface devices to their PC's. Most of the time the base address is used to interface the port rather than LPT1 etc. However should you want to find the address of LPT1 or any of the Line PrinTer Devices, you can use a lookup table provided by BIOS. When BIOS assigns addresses to your printer devices, it stores the address at specific locations in memory, so we can find them.

Start Address	Function
0000:0408	LPT1's Base Address
0000:040A	LPT2's Base Address
0000:040C	LPT3's Base Address
0000:040E	LPT4's Base Address (Note 1)

Table 2.5 - LPT Addresses in the BIOS Data Area;

Note 1 : Address 0000:040E in the BIOS Data Area may be used as the Extended Bios Data Area in PS/2 and newer Bioses.

The above table, table 2.5, shows the address at which we can find the Printer Port's addresses in the BIOS Data Area. Each address will take up 2 bytes.

2.4.3 Software Registers - Standard Parallel Port (SPP)

Offset	Name	Read/Write	Bit No.	Properties
Base + 0	Data Port	Write (Note-1)	Bit 7	Data 7
			Bit 6	Data 6
			Bit 5	Data 5
			Bit 4	Data 4
			Bit 3	Data 3
			Bit 2	Data 2
			Bit 1	Data 1
			Bit 0	Data 0

Table 2.6 Data Port

Note 1 : If the Port is Bi-Directional then Read and Write Operations can be performed on the Data Register.

The base address usually called the Data Port or Data Register is simply used for outputting data on the Parallel Port's data lines (Pins 2-9). This register is normally a write only port. If you read from the port, you should get the last byte sent. However if your port is bi-directional, you can receive data on this address. See Bi-directional Ports for more detail.

Offset	Name	Read/Write	Bit No.	Properties
Base + 1	Status Port	Read Only	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reserved
			Bit 0	Reserved

Table 2.7 Status Port

The Status Port (base address + 1) is a read only port. Any data written to this port will be ignored. The Status Port is made up of 5 input lines (Pins 10,11,12,13 & 15), a IRQ status register and two reserved bits. Please note that Bit 7 (Busy) is a active low input. E.g. If bit 7 happens to show a logic 0, this means that there is +5v at pin 11. Likewise with Bit 2. (nIRQ) If this bit shows a '1' then an interrupt has **not** occurred.

Offset	Name	Read/Write	Bit No.	Properties
Base + 2	Control Port	Read/Write	Bit 7	Unused
			Bit 6	Unused
			Bit 5	Enable Bi-Directional Port
			Bit 4	Enable IRQ Via Ack Line
			Bit 3	Select Printer
			Bit 2	Initialize Printer (Reset)
			Bit 1	Auto Linefeed
			Bit 0	Strobe

Table 2.8 Control Port

The Control Port (base address + 2) was intended as a write only port. When a printer is attached to the Parallel Port, four "controls" are used. These are Strobe, Auto Linefeed, Initialise and Select Printer, all of which are inverted except Initialise.

The printer would not send a signal to initialise the computer, nor would it tell the computer to use auto linefeed. However these four outputs can also be used for inputs. If the computer has placed a pin high (e.g. +5v) and your device wanted to take it low, you would effectively short out the port, causing a conflict on that pin. Therefore these lines are "open collector" outputs (or open drain for CMOS devices). This means that it has two states. A low state (0v) and a high impedance state (open circuit).

Normally the Printer Card will have internal pull-up resistors, but as you would expect, not all will. Some may just have open collector outputs, while others may even have normal totem pole outputs. In order to make your device work correctly on as many Printer Ports as possible, you can use an external resistor as well. Should you already have an internal resistor, then it will act in Parallel with it, or if you have Totem pole outputs, the resistor will act as a load.

An external 4.7k resistor can be used to pull the pin high. I wouldn't use anything lower, just in case you do have an internal pull up resistor, as the external resistor would act in parallel giving effectively, a lower value pull up resistor. When in high impedance state the pin on the Parallel Port is high (+5v). When in this state, your external device can pull the pin low and have the control port change

read a different value. This way the 4 pins of the Control Port can be used for bi-directional data transfer. However the Control Port must be set to xxxx0100 to be able to read data, that is all pins to be +5v at the port so that you can pull it down to GND (logic 0).

Bits 4 & 5 are internal controls. Bit four will enable the IRQ (See Using the Parallel Ports IRQ) and Bit 5 will enable the bi-directional port meaning that you can input 8 bits using (DATA0-7). This mode is only possible if your card supports it. Bits 6 & 7 are reserved. Any writes to these two bits will be ignored.

2.4.4 Using The Parallel Port to Input 8 Bits.

If your Parallel Port doesn't support bi-directional mode, don't despair. You can input a maximum of 9 bits at any one given time. To do this you can use the 5 input lines of the Status Port and the 4 inputs (open collector) lines of the Control Port.

The inputs to the Parallel Port has be chosen as such, to make life easier for us. Busy just happens to be the MSB (Bit 7) of the Status Port, then in ascending order comes Ack, Paper Out and Select, making up the most significant nibble of the Control Port. The Bars are used to represent which inputs are Hardware inverted, i.e. +5v will read 0 from the register, while GND will read 1. The Status Port only has one inverted input.

The Control port is used to read the least significant nibble. As described before, the control port has open collector outputs, i.e. two possible states, high impedance and GND. If we connect our inputs directly to the port (For example an ADC0804 with totem pole outputs), a conflict will result if the input is high and the port is trying to pull it down. Therefore we use open collector inverters.

However this is not always entirely necessary. If we were connecting single pole switches to the port with a pull up resistor, then there is no need to bother with this protection. Also if your software initialises the control port with xxxx0100 so that all the pins on the control port are high, then it may be unnecessary. If however you don't bother and your device is connected to the Parallel Port before your software has a chance to initialise then you may encounter problems.
Another problem to be aware of is the pull up resistors on the control port. The average pull-up resistor is 4.7k. In order to pull the line low, your device will need to sink 1mA, which some low powered devices may struggle to do. Now what happens if I suggest that some ports have 1K pull up resistors? Yes, there are such cards. Your device now has to sink 5mA. More reason to use the open collector inverters.

Open collector inverters were chosen over open collector buffers as they are more popular, and thus easier to obtain. There is no reason, however why you can't use them. Another possibility is to use transistors.

The input, D3 is connected via the inverter to Select Printer. Select Printer just happens to be bit 3 of the control port. D2, D1 & D0 are connected to Init, Auto linefeed and strobe, respectively to make up the lower nibble. Now this is done, all we have to do is assemble the byte using software. The first thing we must do is to write xxxx0100 to the Control Port. This places all the control port lines high, so they can be pulled down to input data.

```
outportb(CONTROL, inportb(CONTROL) & 0xF0 | 0x04);
```

Now that this is done, we can read the most significant nibble. This just happens to be the most significant nibble of the status port. As we are only interested in the MSnibble we will AND the results with 0xF0, so that the LSnibble is clear. Busy is hardware inverted, but we won't worry about it now. Once the two bytes are constructed, we can kill two birds with one stone by toggling Busy and Init at the same time.

```
a = (inportb(STATUS) & 0xF0); /* Read MSnibble */
```

We can now read the LSnibble. This just happens to be LSnibble of the control port - How convenient! This time we are not interested with the MSnibble of the port, thus we AND the result with 0x0F to clear the MSnibble. Once this is done, it is time to combine the two bytes together. This is done by OR'ing the two bytes. This now leaves us with one byte, however we are not finished yet. Bits 2 and 7 are inverted. This is overcome by XOR'ing the byte with 0x84, which toggles the two bits.

```
a = a |(inportb(CONTROL) & 0x0F); /* Read LSnibble */
a = a ^ 0x84; /* Toggle Bit 2 & 7 */
```

Note: Some control ports are not open collector, but have totem pole outputs. This is also the case with EPP and ECP Ports. Normally when you place a Parallel Port in ECP or EPP mode, the control port becomes totem pole outputs only. Now what happens if you connect your device to the Parallel Port in this mode? Therefore, in the interest of portability I recommend using the next circuit, reading a nibble at a time.

2.4.5 Parallel Port Modes in BIOS

Today, most Parallel Ports are multimode ports. They are normally software configurable to one of many modes from BIOS. The typical modes are,

Printer Mode (Sometimes called Default or Normal Modes)
Standard & Bi-directional (SPP) Mode
EPP1.7 and SPP Mode
EPP1.9 and SPP Mode
ECP Mode
ECP and EPP1.7 Mode
ECP and EPP1.9 Mode

Parallel Port Modes and the ECP's Extended Control Register

It is better to set the Parallel Port to ECP and EPP1.9 Mode and use the ECP's Extended Control Register to select different modes of operation. The ECP Registers are standardised under Microsoft's Extended Capabilities Port Protocol and ISA Interface Standard, thus we don't have that problem of every vendor having their own register set.

When set to ECP Mode, a new set of registers become available at Base + 0x400h. Here we are only interested in the Extended Control Register (ECR) which is mapped at Base + 0x402h. It should be stated that the ECP's registers are not available for port's with a base address of 0x3BCh.

The table 2.9 is of the Extended Control Register. We are only interested in the three MSB of the Extended Control Register which selects the mode of operation. There are 7 possible modes of operation, but not all ports will support all modes. The EPP mode is one such example, not being available on some ports.

Bit	Function	
7:5	Selects Current Mode of Operation	
	000	Standard Mode
	001	Byte Mode
	010	Parallel Port FIFO Mode
	011	ECP FIFO Mode
	100	EPP Mode
	101	Reserved
	110	FIFO Test Mode
	111	Configuration Mode
4	ECP Interrupt Bit	
3	DMA Enable Bit	
2	ECP Service Bit	
1	FIFO Full	
0	FIFO Empty	

Table 2.9 - Extended Control Register (ECR)

2.5 Interface by Serial / RS232 Port

The Serial Port is harder to interface than the Parallel Port. In most cases, any device you connect to the serial port will need the serial transmission converted back to parallel so that it can be used. This can be done using a UART. On the software side of things, there are many more registers that you have to attend to than on a Standard Parallel Port. (SPP)

Advantages of using serial data transfer rather than parallel :

Serial Cables can be longer than Parallel cables. The serial port transmits a '1' as -3 to -25 volts and a '0' as +3 to +25 volts where as a parallel port transmits a '0' as 0v and a '1' as 5v. Therefore the serial port can have a maximum swing of 50V compared to the parallel port which has a maximum swing of 5 Volts. Therefore cable loss is not going to be as much of a problem for serial cables than they are for parallel.

You don't need as many wires than parallel transmission. If your device needs to be mounted a far distance away from the computer then 3 core cable (Null Modem Configuration) is going to be a lot cheaper that running 19 or 25 core cable. However you must take into account the cost of the interfacing at each end.

Infra Red devices have proven quite popular recently. You may of seen many electronic diaries and palm top computers which have infra red capabilities build in. However could you imagine transmitting 8 bits of data at the one time across the room and being able to (from the devices point of view) decipher which bits are which? Therefore serial transmission is used where one bit is sent at a time. IrDA-1 (The first infra red specifications) was capable of 115.2k baud and was interfaced into a UART. The pulse length however was cut down to $3/16^{th}$ of a RS232 bit length to conserve power considering these devices are mainly used on diaries, laptops and palmtops.

Microcontroller's have also proven to be quite popular recently. Many of these have in built SCI (Serial Communications Interfaces) which can be used to talk to the outside world. Serial Communication reduces the pin count of these MPU's. Only two pins are commonly used, Transmit Data (TXD) and Receive Data (RXD) compared with at least 8 pins if you use a 8 bit Parallel method (You may also require a Strobe).

2.5.1 Hardware Properties

Devices which use serial cables for their communication are split into two categories. These are DCE (Data Communications Equipment) and DTE (Data Terminal Equipment.) Data Communications Equipment are devices such as your modem, TA adaptor, plotter etc while Data Terminal Equipment is your Computer or Terminal .

The electrical specifications of the serial port is contained in the EIA (Electronics Industry Association) RS232C standard. It states many parameters such as –

1. A "Space" (logic 0) will be between +3 and +25 Volts.
2. A "Mark" (Logic 1) will be between -3 and -25 Volts.
3. The region between +3 and -3 volts is undefined.
4. An open circuit voltage should never exceed 25 volts. (In Reference to GND)
5. A short circuit current should not exceed 500mA. The driver should be able to handle this without damage. (Take note of this one!)

Above is no where near a complete list of the EIA standard. Line Capacitance, Maximum Baud Rates etc are also included. For more information please consult the EIA RS232-C standard. It is interesting to note however, that the RS232C standard specifies a maximum baud

rate of 20,000 BPS!, which is rather slow by today's standards. A new standard, RS-232D has been recently released.
Serial Ports come in two "sizes", There are the D-Type 25 pin connector and the D-Type 9 pin connector both of which are male on the back of the PC, thus you will require a female connector on your device. Below is a table of pin connections for the 9 pin and 25 pin D-Type connectors.

The EIA RS232-C Standard

Specifies a 25 pin connector as the standard interface in data communication networks, with lettering pin designations for ground, data, control and timing circuits. The table 2.9 shows the designations for each of the 25 pins of the standard.

INTERCHANGE	CIRCUIT NO.	PIN NO.	DESCRIPTION
AA	101	1	Projective Ground
BA	103	2	Transmit Data
BB	104	3	Receive Data
CA	105	4	Request to Send
CB	106	5	Clear to Send
CC	107	6	Data set ready
AB	102	7	Signal Ground
CF	109	8	Receive Line Signal
			Detect / Carrier Detect
---	---	9	Reserved
---	---	10	Reserved
---	---	11	Unassigned
SCF	122	12	Secondary RLSD
SCB	121	13	Secondary CTS
SBA	118	14	Secondary TD
DB	114	15	Transmitter Signal
			Element Timing
SBB	119	16	Secondary RD
DD	115	17	Receiver Signal
			Timing Element
---	---	18	Unassigned
SCA	120	19	Secondary RTS
CD	108.2	20	Data Terminal Ready
CG	110	21	Signal Quality
			Detector
CE	125	22	Ring Indicator
CH / CI	111 / 112	23	Data Signal Rate
			Detector
DA	113	24	Transmit Signal
			Element Timing
---	---	25	Unassigned

DTE / DCE Speeds

Table 2.9

We have already talked briefly about DTE & DCE. A typical Data Terminal Device is a computer and a typical Data Communications

Device is a Modem. Often people will talk about DTE to DCE or DCE to DCE speeds. DTE to DCE is the speed between your modem and computer, sometimes referred to as your terminal speed. This should run at faster speeds than the DCE to DCE speed. DCE to DCE is the link between modems, sometimes called the line speed.

Most people today will have 28.8K or 33.6K modems. Therefore we should expect the DCE to DCE speed to be either 28.8K or 33.6K. Considering the high speed of the modem we should expect the DTE to DCE speed to be about 115,200 BPS.(Maximum Speed of the 16550a UART) This is where some people often fall into a trap. The communications program which they use have settings for DCE to DTE speeds. However they see 9.6 KBPS, 14.4 KBPS etc.

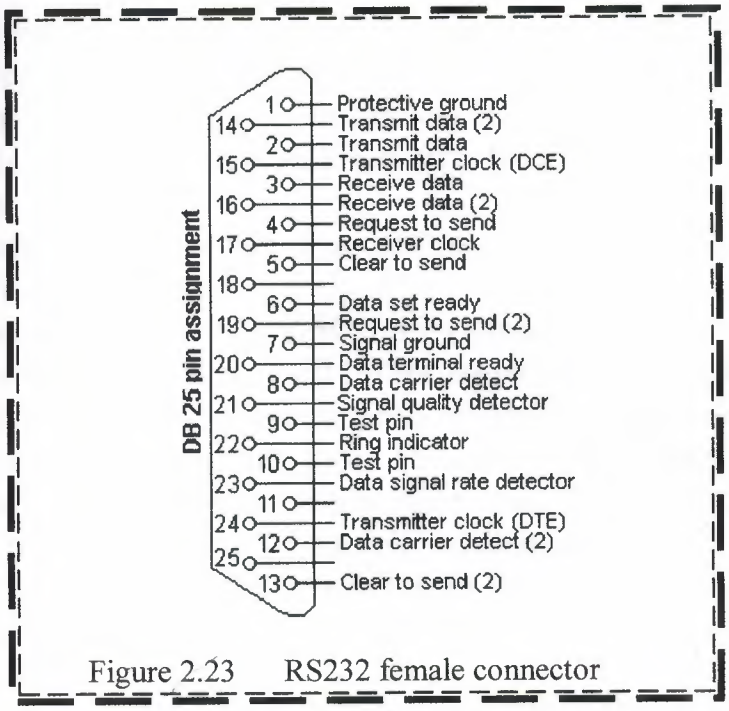


Figure 2.23 RS232 female connector

Today's Modems should have Data Compression build into them. This is very much like PK-ZIP but the software in your modem compresses and decompresses the data. When set up correctly you can expect compression ratios of 1:4 or even higher. 1 to 4 compression would be typical of a text file. If we were transferring that text file at 28.8K (DCE-DCE), then when the modem compresses it you are actually transferring 115.2 KBPS between computers and thus have a DCE-DTE speed of 115.2 KBPS. Thus this is why the DCE-DTE should be much higher than your modem's connection speed.

Some modem manufacturers quote a maximum compression ratio as 1:8. Lets say for example its on a new 33.6 KBPS modem then we may get a maximum 268,800 BPS transfer between modem and

UART. If you only have a 16550a which can do 115,200 BPS tops, then you would be missing out on a extra bit of performance. Buying a 16C650 should fix your problem with a maximum transfer rate of 230,400 BPS.

These are MAXIMUM compression ratios. In some instances if you try to send a already compressed file, your modem can spend more time trying the compress it, thus you get a transmission speed less than your modem's connection speed. If this occurs try turning off your data compression. This should be fixed on newer modems. Some files compress easier than others thus any file which compresses easier is naturally going to have a higher compression ratio.

2.5.2 Flow Control

If our DTE to DCE speed is several times faster than our DCE to DCE speed the PC can send data to your modem at 115,200 BPS. Sooner or later data is going to get lost as buffers overflow, thus flow control is used. Flow control has two basic varieties, Hardware or Software.

Software flow control, sometimes expressed as Xon/Xoff uses two characters Xon and Xoff. Xon is normally indicated by the ASCII 17 character where as the ASCII 19 character is used for Xoff. The modem will only have a small buffer so when the computer fills it up the modem sends a Xoff character to tell the computer to stop sending data. Once the modem has room for more data it then sends a Xon character and the computer sends more data. This type of flow control has the advantage that it doesn't require any more wires as the characters are sent via the TD/RD lines. However on slow links each character requires 10 bits which can slow communications down.

Hardware flow control is also known as RTS/CTS flow control. It uses two wires in your serial cable rather than extra characters transmitted in your data lines. Thus hardware flow control will not slow down transmission times like Xon-Xoff does. When the computer wishes to send data it takes active the Request to Send line. If the modem has room for this data, then the modem will reply by taking active the Clear to Send line and the computer starts sending data. If the modem does not have the room then it will not send a Clear to Send.

RS-232 Signal Descriptions

The interface transfers data between the computer and the modem via the TD and RD lines. The other signals are essentially

used for FLOW CONTROL, in that they either grant or deny requests for the transfer of information between a DTE and a DCE. Data cannot be transferred unless the appropriate flow control line are first asserted.

The interface can send data either way(DTE to DCE, or, DCE to DTE) independently at the same time. This is called FULL DUPLEX operation. Some systems may utilize software codes so that information may only be transmitted in one direction at a time (HALF DUPLEX), and requires software codes to switch from one direction to another (i.e., from a transmit to receive state).
The following is a list of common RS232 signals.

- Request To Send (RTS): This signal line is asserted by the computer (DTE) to inform the modem (DCE) that it wants to transmit data. If the modem decides this is okay, it will assert the CTS line. Typically, once the computer asserts RTS, it will wait for the modem to assert CTS. When CTS is asserted by the modem, the computer will begin to transmit data.
- Clear To Send (CTS): Asserted by the modem after receiving a RTS signal, indicating that the computer can now transmit.
- Data Terminal Ready (DTR): This signal line is asserted by the computer, and informs the modem that the computer is ready to receive data.
- Data Set Ready (DSR): This signal line is asserted by the modem in response to a DTR signal from the computer. The computer will monitor the state of this line after asserting DTR to detect if the modem is turned on.
- Receive Signal Line Detect (RSLD): This control line is asserted by the modem, informing the computer that it has established a physical connection to another modem. It is sometimes known as Carrier Detect (CD). It would be pointless a computer transmitting information to a modem if this signal line was not asserted. If the physical connection is broken, this signal line will change state.
- Transmit Data (TD): is the line where the data is transmitted, a bit at a time.
- Receive Data (RD): is the line where data is received, a bit at a time.

A lot of signals work in pairs. Some signals are generated by the DTE, and some signals are generated by the DCE. If you were measuring the signals on a computer which was NOT connected to a modem, you could only expect to see those signals that the DTE can generate.

Exchanging information between a DCE and DTE :

Now, lets look at the sequence that occurs when data is transferred between a DTE and a DCE. The data can only be transferred after the correct sequence of signals is followed, for instance, there is no point sending data if the modem is turned off. Lets go through each of the steps involved (i.e., signal line assertions required) to transmit and receive characters across the RS232 interface.

TRANSMITTING DATA (DTE to DCE)

- 1: Assert DTR and RTS
- 2: Wait for DSR
- 3: Wait for CTS
- 4: Transmit the data

Step 1 and 2 are essential to ensure that the modem is on-line and connected to another modem. Waiting for DSR checks that the modem is on-line.

RECEIVING DATA (DCE to DTE)

- 1: Assert DTR
- 2: Wait for DSR
- 3: Receive the data

2.5.3 Connection of Two DTE Devices

Often, two DTE devices need to be connected together using a serial link. This is for file transfer or printer access. The problem is that DTE devices expect to talk directly to DCE devices, not another device of the same type. DTE's cannot generate signals like DSR and CTS, so connecting two DTE's together will result in neither getting permission to send, and thinking that the modem is off-line (by not receiving DSR).

To allow the interconnection of two DTE devices without using DCE's, a special type of cable must be used. This is called a Null Modem Cable, which fools the DTE into thinking that it is connected to a DCE device. In this case, modems are not used, so the connection looks like :

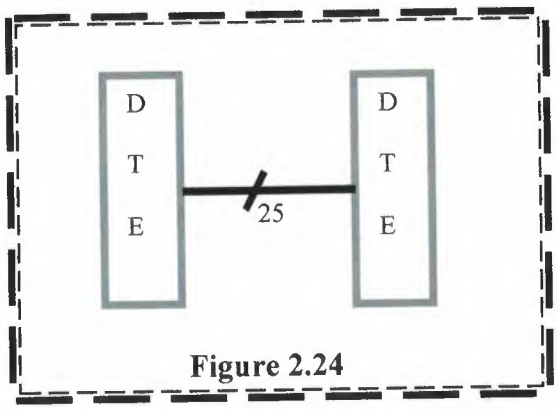


Figure 2.24

RS232D (9 pin Connector)

The following table illustrates the 9 pin serial connector as found on most PC's today. This has all but replaced the previous 25 pin connector found on earlier PC's.

SIGNAL	PIN No.
Carrier Detect	1
Receive Data	2
Transmit Data	3
Data Terminal Ready	4
Signal Ground	5
Data Set Ready	6
Request To Send	7
Clear To Send	8
Ring Indicator	9

Table 2.10 Pin Descriptions of DB9

2.5.4 Serial Port's Registers (PC's)

Port Addresses & IRQ's

Name	Address	IRQ
COM 1	3F8	4
COM 2	2F8	3
COM 3	3E8	4
COM 4	2E8	3

Table 2.11 Standard Port Addresses

Above is the standard port addresses. These should work for most P.C's. If you just happen to be lucky enough to own a IBM P/S2 which has a micro-channel bus, then expect a different set of addresses and IRQ's. Just like the LPT ports, the base addresses for the COM ports can be read from the BIOS Data Area.

Start Address	Function
0000:0400	COM1's Base Address
0000:0402	COM2's Base Address
0000:0404	COM3's Base Address
0000:0406	COM4's Base Address

Table 2.12 COM Port Addresses in the BIOS Data Area

First In / First Out Control Register (FCR)

The FIFO register is a write only register. This register is used to control the FIFO (First In / First Out) buffers which are found on 16550's and higher.

Bit 0 enables the operation of the receive and transmit FIFO's. Writing a '0' to this bit will disable the operation of transmit and receive FIFO's, thus you will loose all data stored in these FIFO buffers.

Bit's 1 and 2 control the clearing of the transmit or receive FIFO's. Bit 1 is responsible for the receive buffer while bit 2 is responsible for the transmit buffer. Setting these bits to 1 will only clear the contents of the FIFO and will not affect the shift registers. These two bits are self resetting, thus you don't need to set the bits to '0' when finished.

Bit	Notes		
Bits 6 and 7	Bit 7	Bit 6	Interrupt Trigger Level
	0	0	1 Byte
	0	1	4 Bytes
	1	0	8 Bytes
	1	1	14 Bytes
Bit 5	Enable 64 Byte FIFO (16750 only)		
Bit 4	Reserved		
Bit 3	DMA Mode Select. Change status of RXRDY & TXRDY pins from mode 1 to mode 2.		
Bit 2	Clear Transmit FIFO		
Bit 1	Clear Receive FIFO		
Bit 0	Enable FIFO's		

Table 2.13 : FIFO Control Register

Bit 3 enables the DMA mode select which is found on 16550 UARTs and higher. More on this later. Bits 4 and 5 are those easy type again, Reserved.

Bits 6 and 7 are used to set the triggering level on the Receive FIFO. For example if bit 7 was set to '1' and bit 6 was set to '0' then

the trigger level is set to 8 bytes. When there is 8 bytes of data in the receive FIFO then the Received Data Available interrupt is set.

The line status register is a read only register. Bit 7 is the error in received FIFO bit. This bit is high when at least one break, parity or framing error has occurred on a byte which is contained in the FIFO.

When bit 6 is set, both the transmitter holding register and the shift register are empty. The UART's holding register holds the next byte of data to be sent in parallel fashion. The shift register is used to convert the byte to serial, so that it can be transmitted over one line. When bit 5 is set, only the transmitter holding register is empty. So what's the difference between the two? When bit 6, the transmitter holding and shift registers are empty, no serial conversions are taking place so there should be no activity on the transmit data line. When bit 5 is set, the transmitter holding register is empty, thus another byte can be sent to the data port, but a serial conversion using the shift register may be taking place.

The break interrupt (Bit 4) occurs when the received data line is held in a logic state '0' (Space) for more than the time it takes to send a full word. That includes the time for the start bit, data bits, parity bits and stop bits.

A framing error (Bit 3) occurs when the last bit is not a stop bit. This may occur due to a timing error. You will most commonly encounter a framing error when using a null modem linking two computers or a protocol analyser when the speed at which the data is being sent is different to that of what you have the UART set to receive it at.

A overrun error normally occurs when your program can't read from the port fast enough. If you don't get an incoming byte out of the register fast enough, and another byte just happens to be received, then the last byte will be lost and a overrun error will result.

Bit 0 shows data ready, which means that a byte has been received by the UART and is at the receiver buffer ready to be read.

2.6 DAC & ADC Conversion

We have a certain number of bits whose logic levels are characterised with electric levels, say logic zero = 0 volts and logic one = V volts. This bit set can be interpreted as a number comprised between zero and 2n - 1, and so it can represent the levels of an analog signal. The analog signal can be extracted directly from the digital signals if the electric levels of all of the bits are summed up with certain weights. The most significant bit is not weighted (that is, it has weight = 1), the bit next to it has a weight of 1/2, the following bit has

a weight of 1/4 and so on. The least significant bit will have a weight of 1/2n - 1. This way the resulting output signal is analog (because it is not restricted to two values, 0 or V volts) and its level is proportional to the logic value represented by the bit set.

In the practice the DAC is applied to the parallel port of the PC. In this case we have 8 bits (but it could be more if we use not only the data pins, but also the control pins) whose electric levels are 0 volts (for logic zero) and approximately +5 volts (for logic one). The summing and weighting of the bits are made through resistors. The simplest way is to connect the pin of the bit number j with a resistor whose value is 2j x R. This would require 8 resistors, each twice the previous one, something which is difficult to achieve in practice. That's why a more elaborated circuit was designed. In the figure you can see that the DAC in made with a chain of
resistors which only needs two different kinds of resistors. This makes the circuit stabler and easier to build in practice. The output signal will be approximately comprised between zero and +5 volts with 28 = 256 different levels, that is, the resolution of the DAC is 8 bits.

The main problem was in the design of a simple threshold detector. It has to be sensible to one step of the DAC, which is 5/256 = 20 mV. The conversion of a small step of voltage into a logic signal (for example, 5 volts if the step exceeded threshold and 0 volts otherwise) is typically done through a chain of transistors, so that they convert a step of 20 mV into a step of 5 V (gain = 5/20x10-3 = 250). As long as I wanted to keep the design as simple as possible I tried to use only one transistor, then the transistor has to be carefully chosen and polarized. My knowledge of transistors was not enough to do it this way, so I tried another approach.

The specifications of the logic levels say basically that there are 3 situations when an input signal is applied. If the signal is below some level, say 1 volt, it will be interpreted as a logic 0. If the signal is above a certain level, say 4 volts, it will be interpreted as a logic 1. If the signal is between both level then nothing can be said, the manufacturer of the equipment cannot predict the behaviour, it will be randomly interpreted as a logic 0 or a logic 1. My bet was that the level in which the
interpretation of the input signal changes from logic 0 to logic 1 is fixed and stable enough, though it will change randomly from computer to computer.

The input signal to be digitalised is added to the DAC signal and applied to the base of a transistor. The collector of the transistor is connected to an input of the parallel port and the emitter to a variable resistor. The ends of the variable resistor are connected to ground and to a negative voltage source. I took the serial port as a source for this negative voltage, but any other source can be used. The serial port uses two electric levels, -12 and +12 volts approximately. There should not be risk of damage because the power consumption is low, besides, by definition of the RS-232C standard, any pin can be short-circuited with any other pin (including ground) without damage, otherwise your port is not a real RS-232C. When the signal in the base of the transistor grows, the voltage of the collector decreases, that's how the input of the parallel port is set from 1 to 0. The variable resistor has to be adjusted so that in absence of signal the digitalisation results in the level 128 of 256. The volume control of the sound source (microphone, cassette, radio, CD, etc.) has to be adjusted so that the extreme levels (0 and 255) are almost never reached (to avoid saturation).

After the first trials I was surprised with the stability of the device. My bet was right. Once the variable resistor has been correctly set, the fluctuation is +/- 1 LSB, that is, like professional devices!

According to Nyquist's theorem, the maximum frequency that can be successfully digitalised is half of the sampling frequency. With my 386/40 MHz running a C program I scanned at a rate of 14 kHz. This sets the limit in 7 kHz, which is enough for human voice but not for hi-fi music. With a faster computer and programming directly in assembler the limit should be raised. To eliminate frequencies incorrectly digitalised, so they appear as noise, I added the low pass filter that can be found in the design. If your computer is faster than mine maybe you'll have to change the filter's constant to be optimum with your setup.

The digitalisation process requires a lot of memory, at a sampling rate of 14 kHz, the data flow is 14 kb per second. As long as I don't know how to manage extended memory, my particular solution was to create a RAM disk and use it to store temporarily the data.

The design is very experimental and is intended to serve as an introduction to basic electronics, interfacing PCs with your own devices and its programming (you have to take into account that now really good sound cards have become quite cheap). I'm currently working on other devices so I don't plan to improve this design, but it can be done in many ways. Try it. The filter can be better, the extra

source (for which I took the serial port) might be removed, the manual switch that changes from ADC mode to DAC mode can be substituted by an automatic switch (transistor, relay) controlled with the parallel port itself. Also the software is very naive, I didn't care about timing, so now one have to try several times to play the data at the right speed. More efficient code could be written to use extended memory, to make some sound enhancement, data compression, etc.

As a closing remark, it is interesting to notice that there are several sound cards which use the parallel port, like Disney Sound. They are basically a DAC as described here. Therefore it is very likely that you can play the example found in the preceding page with the software there, in any of these cards. Conversely it's also true, you may use the design of this page with any program that allows sound cards attached to the parallel port.

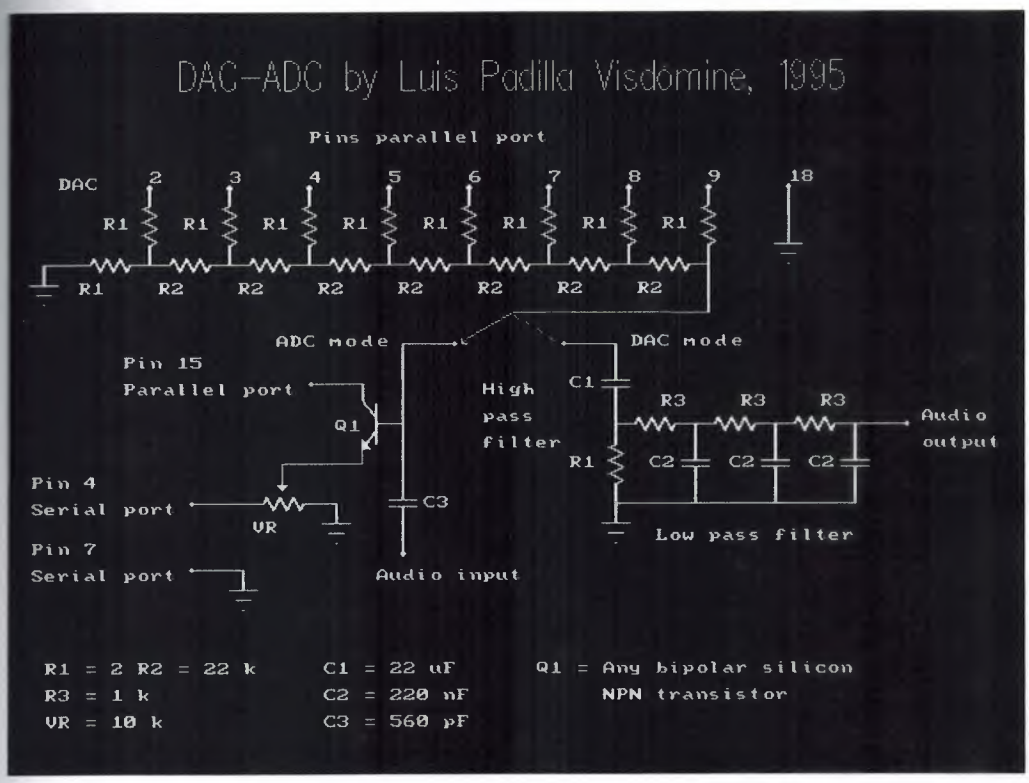


Figure 2.25

The analog-to-digital converter (ADC) is used to convert an analog voltage to a digital number (figure 2.26).

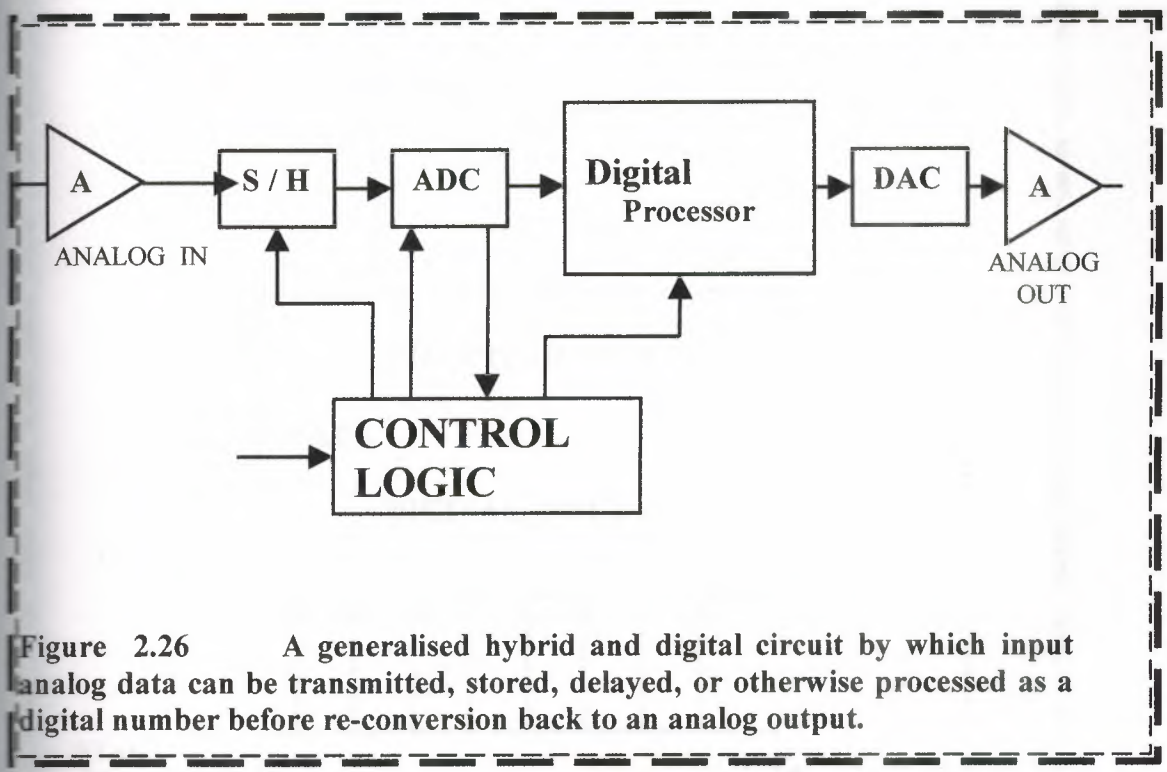


Figure 2.26 A generalised hybrid and digital circuit by which input analog data can be transmitted, stored, delayed, or otherwise processed as a digital number before re-conversion back to an analog output.

The parallel-encoding or flash ADC design provides the fastest operation at the expense of high component count and high cost (figure).

The resistor network sets discrete thresholds for a number of comparators. All comparators with thresholds above the input signal go false while those below go true. Then digital encoding logic converts the result to a digital number.

The successive-approximation ADC is the most commonly used design (figure). This design requires only a single comparator and will be only as good as the DAC used in the circuit.

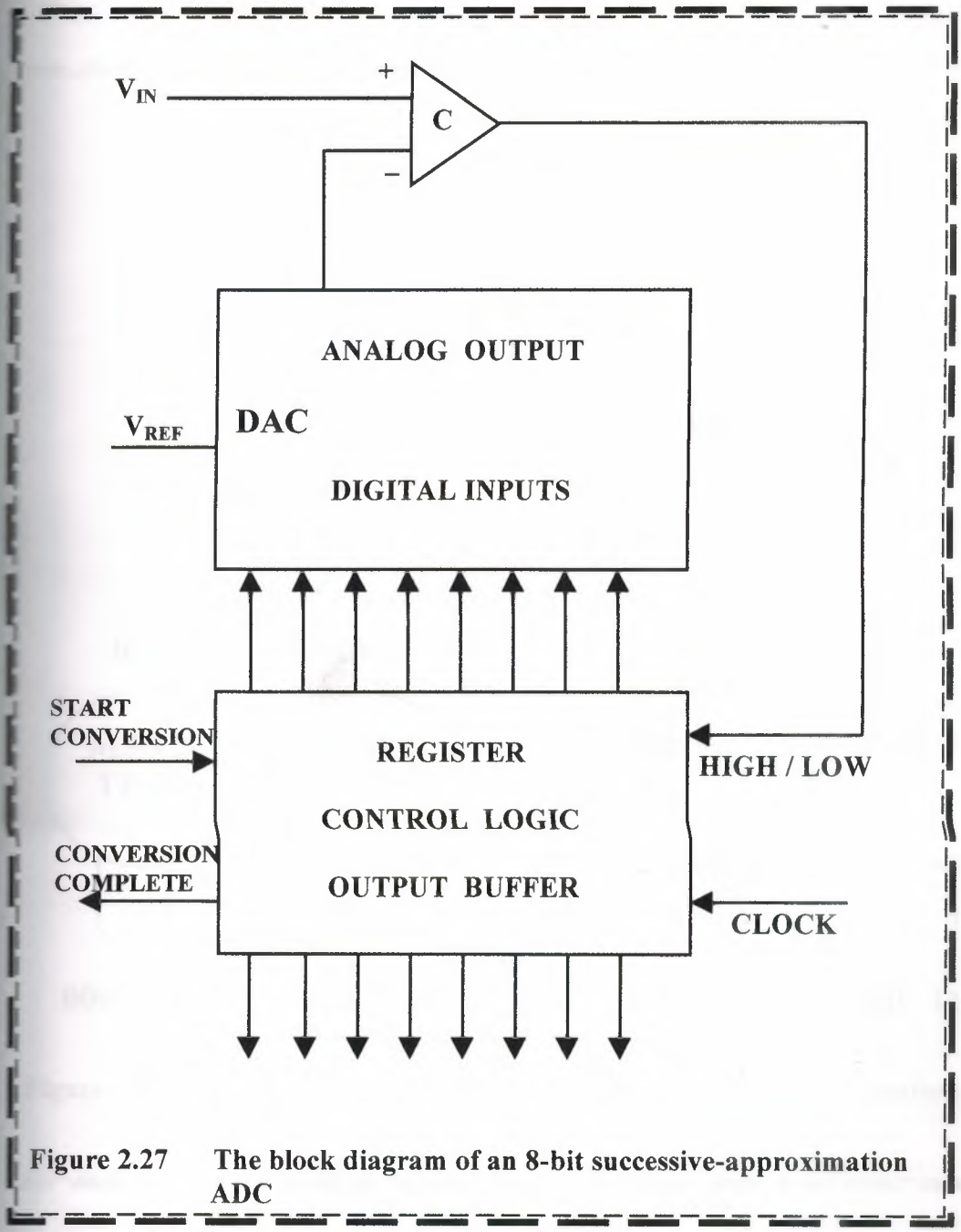


Figure 2.27 **The block diagram of an 8-bit successive-approximation ADC**

The analog output of a high-speed DAC is compared against the analog input signal. The digital result of the comparison is used to control the contents of a digital buffer that both drives the DAC and provides the digital output word.

The successive-approximation ADC uses fast control logic which requires only *n* comparisons for an *n*-bit binary result (figure 2.28).

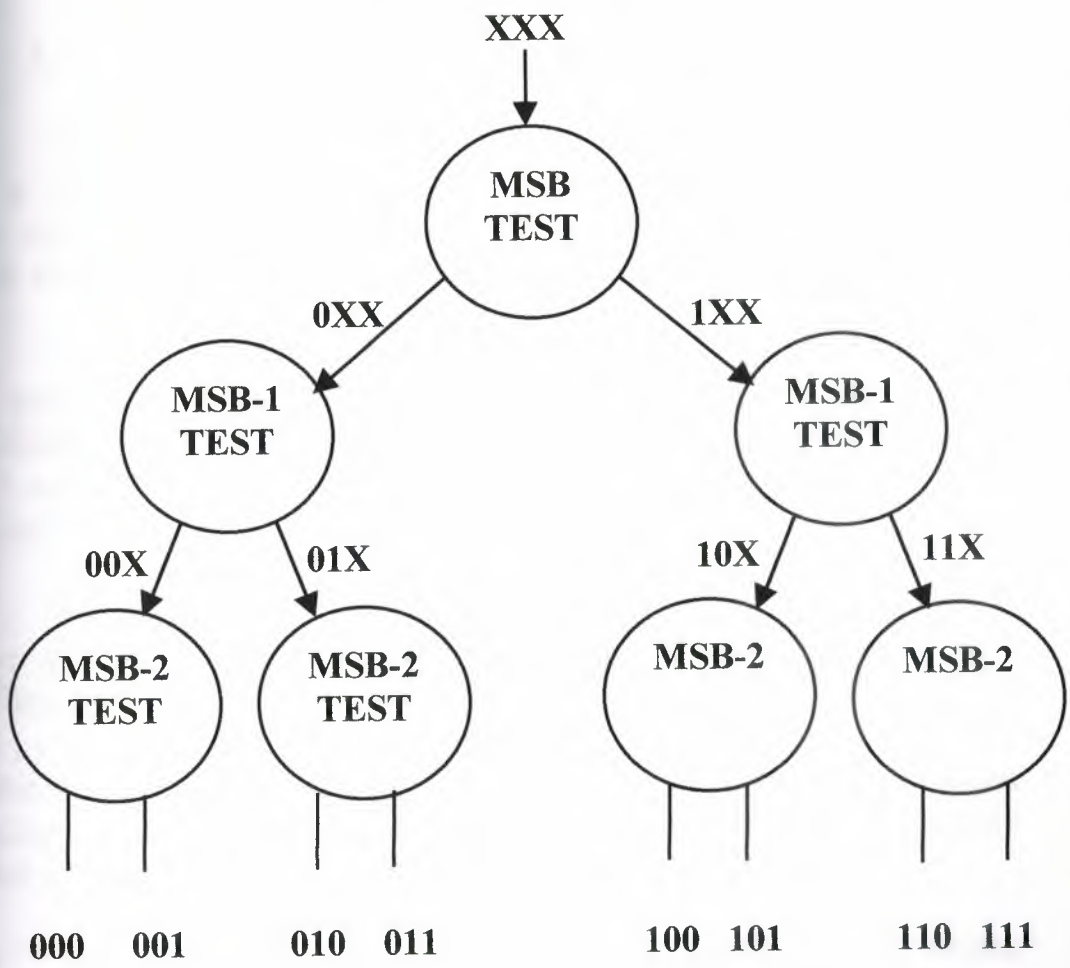


Figure 2.28 The bit-testing sequence used in the successive approximation method.

CHAPTER 3

NAVIGATION OF MOBILE ROBOTS

3.1 Location Estimation of Mobile Robots

The mobile robot has four wheels located in the vertices of a rhomb . The front and rear wheels in the longitudinal axis are steered at the same time by a DC motor with a rigid link . The two parallel wheels are driven dependently by DC motor .

The center of the coordinate frame attached to the vehicle is placed in the center of rhomb . The X – axis is on the transverse axis of the vehicle and points to the right , while the Y – axis is on the longitudinal axis on points to the front (see figure 3.1) .

For path tracking (Ollero and others , 1994) the front and rear wheels are steered , and the parallel wheels are used with differential drive . There are many ways to computer the position (x , y) , orientation (θ) , curvature (κ) and velocity (v) of the vehicle the in the global coordinates framework . Data from multiple sensors must be merged to produce one accurate positioning information .

The distance travelled by the right wheel (ds_r) and by the left wheel (ds_l) are computed through two incremental encoders located in the motors (Kanayama and others , 1988) . The steering angle (α) of the front and rear wheels is calculated by means of an incremental encoders , which is very accurate , placed in one of the extremes of the rigid link , and by an absolute encoder located at the other end of the link . The absolute encoder is used to reset the count of the incremental encoder , when the vehicle is booted up .

Each one of the traction motors has a techno – generator that allows to compute the velocity of the right (V_r) and left (V_l) wheels . A gyroscope mounted on the vehicle measures the angular velocity (w) .

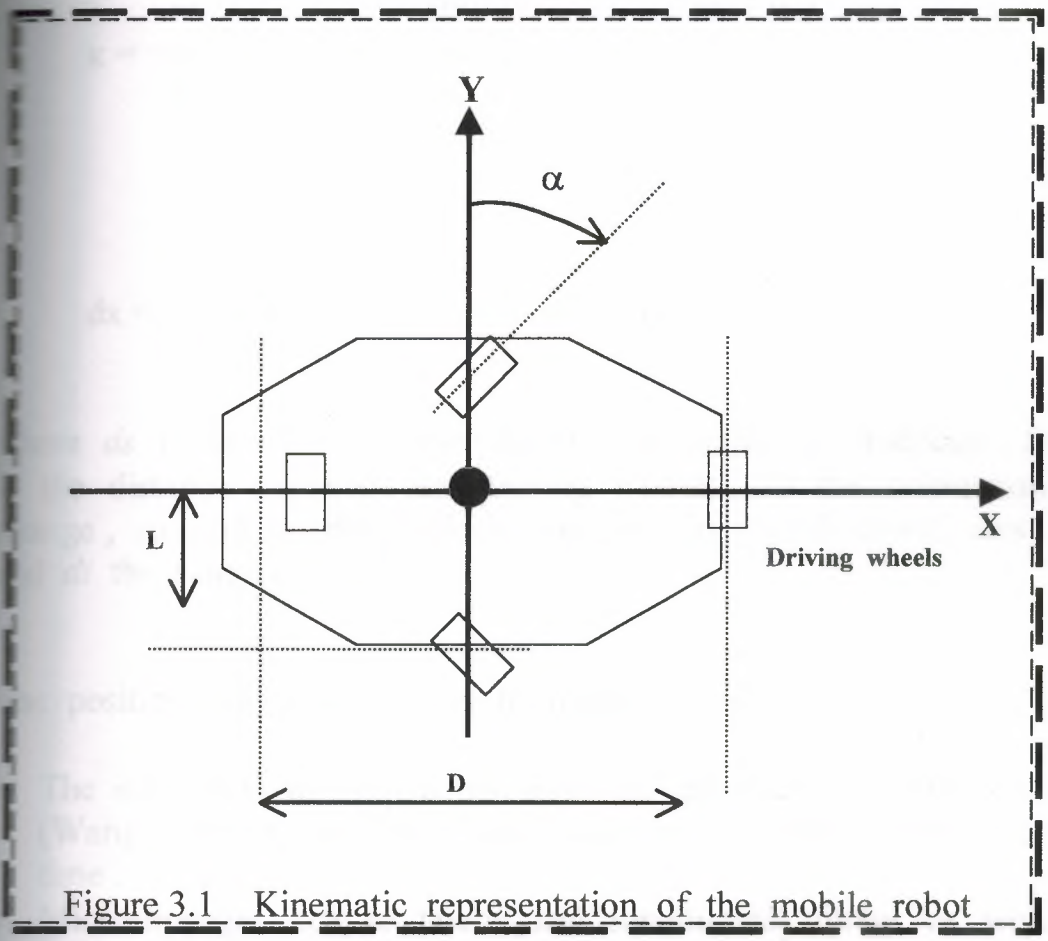


Figure 3.1 Kinematic representation of the mobile robot

Depending on the source of the data used , the location information can be computed in various ways :

$$ds = \frac{ds_l + ds_r}{2} \qquad ds = v\ dt \qquad d\theta = \kappa\ ds$$

$$v = \frac{ds}{dt} \qquad v = \frac{v_r + v_l}{2} \qquad v = \frac{w}{\kappa} \tag{1}$$

$$d\theta = \frac{ds_r - ds_l}{D} \qquad d\theta = \frac{v_r - v_l}{D}\ dt$$

$$\kappa = \frac{w}{v} \qquad \kappa = \frac{d\theta}{ds} \qquad \kappa = \frac{\tan(\alpha)}{L}$$

$$dx = -\sin(\theta)\ ds \qquad dy = \cos(\theta)\ ds$$

where *ds* is the distance travelled by the vehicle's midpoint , *D* is the distance between the steering wheels , $d\theta$ the orientation change , *dx* and *dy* the position variation in global coordinates , and *dt* the sampling period .

The position uncertainty arises from the following facts :

- The odometric measurements from the encoders are not exact (Wang , 1988) , and they are made at discrete intervals of time .
- Microslippages due to lateral and longitudinal forces as well as lack of friction with the ground .
- The vehicle's initial location is not free of errors .
- The gyroscope produces noisy data .

3.2 Navigation of Mobile Robot under Uncertainty Model

When the vehicle follows a path , position uncertainty grows , due to the factors described above . This is an important fact for safe path planning and generation (figure 3.2) .

The path planner builds a path by using an expanded version of the environment . So , a critical parameter is the expansion factor of the environment , which is a function of a safety distance computed with the robot uncertainty model . So , it is necessary to replan the path when the position uncertainty area intersects a polygon that models a real obstacle .

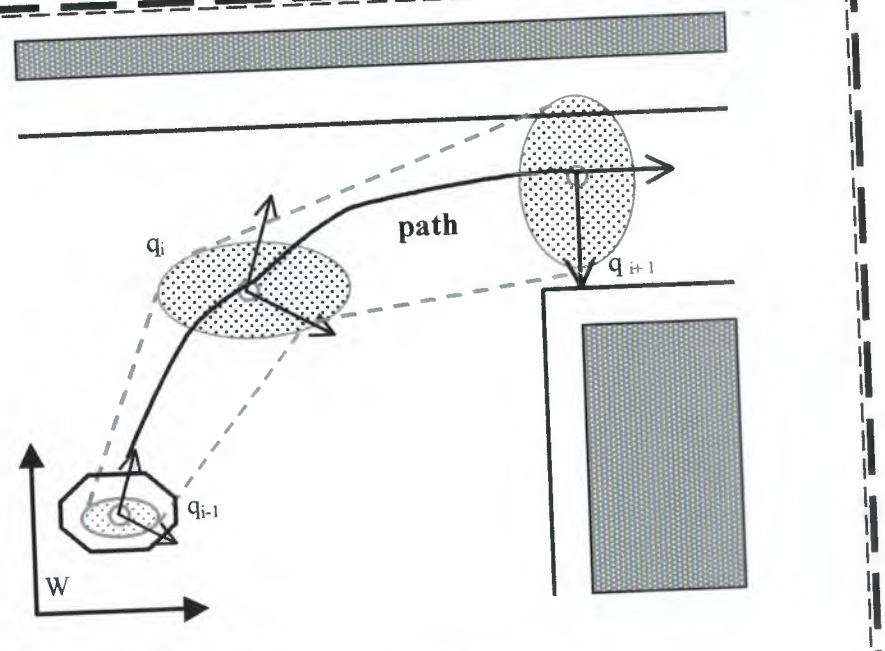


Figure 3.2 The growth of position uncertainty

Therefore , if the next posture provided by the path is given by the vector $q_{i+1} = (x_{i+1} , y_{i+1} , \theta_{i+1})$ with respect to current reference frame at posture ${}^{w}q_i = ({}^{w}x_i , {}^{w}y_i , {}^{w}\theta_i)$ and this last frame is defined with respect to the global world system W , then the expresion of ${}^{w}q_{i+1}$ is given by :

$$
\begin{aligned}
{}^{w}x_{i+1} &= {}^{w}x_i + \cos ({}^{w}\theta_i) \; x_{i+1} - \sin ({}^{w}\theta_i) \; y_{i+1} \\
{}^{w}y_{i+1} &= {}^{w}y_i + \sin ({}^{w}\theta_i) \; x_{i+1} - \cos ({}^{w}\theta_i) \; y_{i+1} \qquad (2) \\
{}^{w}\theta_{i+1} &= {}^{w}\theta_i + {}^{w}\theta i_{+1}
\end{aligned}
$$

However , both vectors q_{i+1} and ${}^{w}q_i$ are not exact . Assume that these components have means provided by ${}^{w}\hat{q}_i = ({}^{w}\hat{x}_i , {}^{w}\hat{y}_i , {}^{w}\hat{\theta}_i)$ and $\hat{}\,{}^{w}q_{i+1} = ({}^{w}\hat{x}_{i+1} , {}^{w}\hat{y}_{i+1} , {}^{w}\hat{\theta}_{i+1})$ and a dispersion vector defined by ${}^{w}\Delta q_i = ({}^{w}\Delta x_i , {}^{w}\Delta y_i , {}^{w}\Delta \theta_i)$ and ${}^{w}\Delta q_{i+1} = ({}^{w}\Delta x_{i+1} , {}^{w}\Delta y_{i+1} , {}^{w}\Delta \theta_{i+1})$. The dispersion vectors represent a set of normal distributed random variables with zero mean . The covarience matrix is defined as follows :

$$ {}^{w}C_i = E\left[{}^{w}\Delta q^{i}{}_{T} \quad {}_{w}\Delta q_i \right] $$

$$ C_{i+1} = E\left[\Delta q^{T}{}_{i+1} \quad \Delta q_{i+1} \right] \qquad (3) $$

where E represents the mathematical exceptation , and T the transpose operation .

The calculation of ${}^{w}\hat{q}_{i+1} = ({}^{w}\hat{x}_{i+1} , {}^{w}\hat{y}_{i+1} , {}^{w}\hat{\theta}_{i+1})$ and ${}^{w}\Delta q_{i+1} = (\Delta x_{i+1} , \Delta y_{i+1} , \Delta\theta_{i+1})$ is necessary for computing the real value of ${}^{w}q_{i+1} = ({}^{w}x_{i+1} , {}^{w}y_{i+1} , {}^{w}\theta_{i+1})$. These vectors are approximated by using Taylor expansion of the expression (2) evaluated at the mean values (Smith and Cheeseman , 1986) :

$$ {}^{w}\hat{x}_{i+1} \approx {}^{w}\hat{x}_i + \cos ({}^{w}\hat{\theta}_i) \hat{x}_{i+1} - \sin ({}^{w}\hat{\theta}_i) \hat{y}_{i+1} $$

$$ {}^{w}\hat{y}_{i+1} \approx {}^{w}\hat{y}_i + \sin ({}^{w}\hat{\theta}_i) \hat{x}_{i+1} - \cos ({}^{w}\hat{\theta}_i) \hat{y}_{i+1} $$

$$ {}^{w}\hat{\theta}_{i+1} = {}^{w}\hat{\theta}_i + {}^{w}\hat{\theta}i_{+1} $$

$$ \Delta q_{i+1} \approx J \, (\Delta q_i \, , \Delta q_{i+1}) \qquad (4) $$

where J is the Jacobian matrix :

$$ J = \begin{pmatrix} 1 & 0 & {}^{w}\hat{y}_i - {}^{w}\hat{y}_{i+1} & \cos ({}^{w}\hat{\theta}_i) & - \sin ({}^{w}\hat{\theta}_i) & 0 \\ 0 & 1 & {}^{w}\hat{x}_{i+1} - {}^{w}\hat{x}_i & \sin ({}^{w}\hat{\theta}_i) & \cos ({}^{w}\hat{\theta}_i) & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \qquad (5) $$

If the errors Δq_{i+1} and ${}^{w}\Delta q_i$ are independent then the covariance matrix ${}^{w}C_{i+1}$ can be :

$$ {}^{w}C_{i+1} \approx J \begin{pmatrix} {}^{w}C_i & 0 \\ 0 & C_{i+1} \end{pmatrix} J^{T} \qquad (6) $$

The ellipse that defines the uncertainty area centered around (x_i, y_i) attached to a covariance matrix ${}^{w}C_{i+1}$ with a threshold of 90 % is given by the following expression (Figure 3.3) :

$$4.6 = \frac{{}^{w}\sigma y^2_{i+1}\ x^2 + {}^{w}\sigma x^2_{i+1}\ y^2 - 2\ {}^{w}\sigma xy_{i+1}\ x\ y}{{}^{w}\sigma x^2_{i+1}\ {}^{w}\sigma y^2_{i+1} - {}^{w}\sigma xy^2_{i+1}} \qquad (7)$$

where ${}^{w}\sigma x^2_{i+1}$, ${}^{w}\sigma y^2_{i+1}$ and ${}^{w}\sigma xy_{i+1}$ are the elements (1 , 1), (2 , 2) and (1 , 2) of ${}^{w}C_{i+1}$ respectively .

The graphical parameters necessary to represent the ellipse consist of the angle α_{i+1} used to align the minor and major radius with the global axis :

$$\alpha_{i+1} = \frac{1}{2} \operatorname{atan}\left[\frac{2\ {}^{w}\sigma xy_{i+1}}{{}^{w}\sigma y^2_{i+1} - {}^{w}\sigma x^2_{i+1}}\right] \qquad (8)$$

and the length of both radius :

$$Rx_{i+1} = \sqrt{\frac{9.21\ ({}^{w}\sigma x^2_{i+1}\ {}^{w}\sigma y^2_{i+1} - {}^{w}\sigma xy^2_{i+1})}{{}^{w}\sigma x^2_{i+1} + {}^{w}\sigma y^2_{i+1} + T}} \qquad (9)$$

$$Ry_{i+1} = \sqrt{\frac{9.21\ ({}^{w}\sigma x^2_{i+1}\ {}^{w}\sigma y^2_{i+1} - {}^{w}\sigma xy^2_{i+1})}{{}^{w}\sigma x^2_{i+1} + {}^{w}\sigma y^2_{i+1} - T}}$$

where T is defined by :

$$T = \sqrt{({}^{w}\sigma y^2_{i+1} - {}^{w}\sigma x^2_{i+1})^2 + 4\ {}^{w}\sigma xy^2_{i+1}} \qquad (10)$$

From the path tracking point of view , the above methodology to propogate the position uncertainty and the set of closed from to describe the uncertainty area on a plane are the same , the only difference lies on considering q_{i+1} as the vehicle's actual position estimation with respect ${}^{w}q_i$, which is the previous position of the robot in world coordinates .

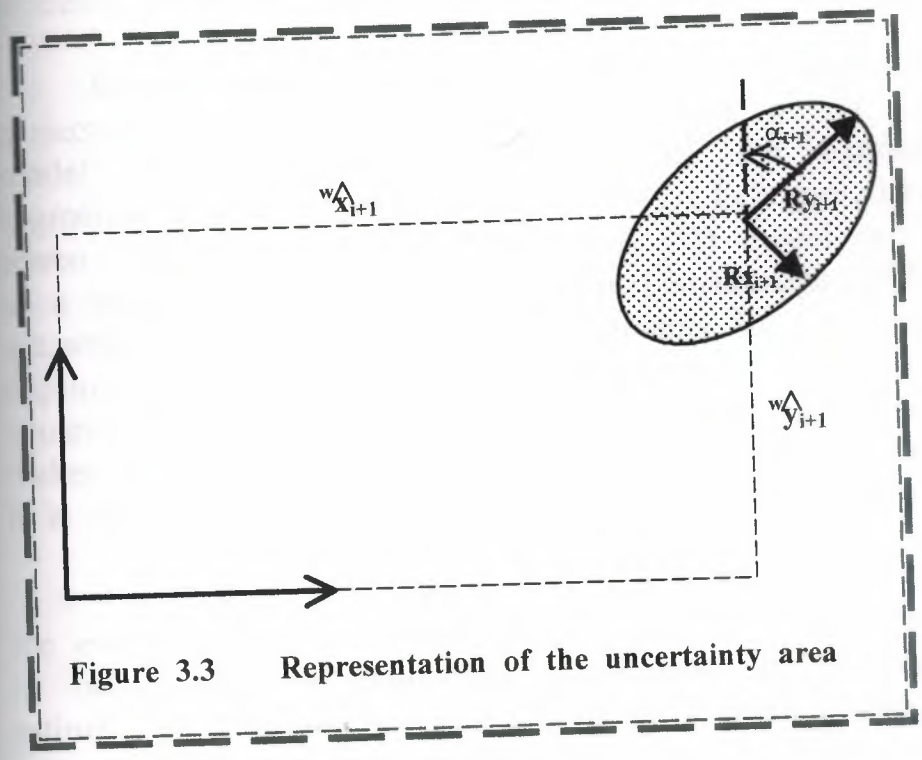


Figure 3.3 Representation of the uncertainty area

3.3 Real – Time Vision – Based Navigation and 3D Depth Estimation

3.3.1 Navigation

The problem of estimating the position of mobile robots using artificial vision is being widely studied . Focusing our attention on systems that use a single fixed camera , some authors have determined the mobile robot's position by tracking landmarks (Leonard , 1991) or geometrical beacons (Atiya , 1993) or visual cues explicitly placed in the environment (Baumgartner , 1994) .

Other authors exploit properties and restrictions of structured environments in which the robot navigates . Indoor scenes including offices , rooms and corridors . Usually have a

rectangular structure with three prominent orientations for 3D line segments : one vertical and two horizontal orientations perpendicular to each other . There are not many lines in any other orientations than these three . We can also assume that major structures such as walls and doorways remain static , and that floor and walls are approximately flat and even . It is possible to take advantage of this " a priori " knowledge when processing indoor images .

Recent papers apply these and other assumptions and projection geometry properties to determine a 3D parametric model of structured environments from the extracted line segments in monocular (Straforini , 1992 ; Lebegue , 1993) and stereo (Christensen , 1994) images . This 3D model can be used as a basis for mobile robot navigation (Shakunaga , 1992) . The extraction of the significant line segments in the image often requires a great deal of low level processing , and a mimimum squares fitting process of pixels into lines . This complexity makes it difficult to implement these algorithms in real time if it is not by means of extensive use of custom hardware .

Conversly , if we have a previously stored 3D model of the environment and under certain bounds or restrictions referred to actual position estimation uncertainty . It is possible to estimate the position and location of the camera (mobile robot) respect to the environment . A recent paper (Christensen , 1994) employs the line segment extracted from the image in a minimumsquares matching process with the 2D pin – hole projection of the 3D model from the expected point of view , to update robot position and orientation.

The main purpose of the vision based location system is to correct the robot position (X , Y) and orientation (H) estimated by odometer in real time and to increase its level of confidence . It will also be assumed that the approximate initial position of the robot is known . This may have been obtained by an initial condition or as a result of a previous iteration of the location system .

A position implies an uncertainty level that represents a probable locations ellipsoid with main axis (dX , dY , dH) . A deterministic searching algorithm is performed inside the volume of this ellipsoid ellipsoid . In order to obtain the position and

orientation in which a fitting function reaches a maximum . In each test point , the fitting function takes as inputs the gradient image and the 3D map projected from the test point , and returns a normalized value indicating how good is the matching between both inputs .

3.3.2 3D CAD Processing

In order to increase the speed , a parallel process that selects the interesting features in the 3D CAD map is used . It mainly consists in a clipping and hiding process followed by a line segment weighting that measures the relative impoertance or confidence of such feature in the scene (sockles and doorway frameworks are more important and visible than other line segments). This 3D CAD map processing is of major importance : a high number of features not only represents a higher computational requirement but , due to perspective effects , it can result in a number of view points which return a near maximum matching . While a low number of features reduces the algorithm robustness to unmodelled occlusions , poor lightning , etc . It has been found that a number within ten and twenty line segments in the local CAD 3D map is usually a good solution .

The resulting local 3D map contains only the segment lines that must be matched with the gradient image . In the maximum matching searching algorithm , this local 3D map is projected from view points relatively close to one other (X and Y increments of 1 cm and Heading increments of 0.05 degrees) , so this transformations has been linearized to optimize its performance .

3.3.3 Egomotion and Depth

Optic flow techniques estimate the projection of relative motion on the image plane to construct a 2D flow field . In a discrete case , optic flow is measured on a set of points that could form a uniform grid or any other type of configuration of points for which displacement through time should be found . Many techniques exist , each differing in the type of property that is considered conserved , but with the common computational restriction of having to use information that is spatio – temporally local to each point (aperture problem) .

This restriciton implies that optic flow cannot be calculated with the same confidence in all points , creating situations where estimations in some regions must be ignored . This is the case of uniform regions for which the lack of textural information provokes bad measurements in all directions and the case of regions with a single strong gradient edge for which good estimations can only be made in the direction normal to the edge . On the other hand , corners allow high confidence estimations .

Low confidence estimates can be improved by imposing neighborhood constraints (smoothing , velocity propogation , etc.) . In any case , it is computationally wasteful to calculate estimates that will be ignored because their confidence is too low . Furthermore , improvement of confidence measures usually implies an iterative procedure for which convergence will be slow if initial estimates are bad .

Therefore , it would be desirable to estimate optic flow only on points which are a – priori good candidates for having a high confidence measurement . This of course calls for algorithms that do not require a dense uniform grid for their computations .

An optic flow field may be used to estimate egomotion and relative depth . Most existing algorithms have assumed however a dense uniform flow field , situation that is typically inadequate because of variable confidence regions .

A recent paper by Hammel (1993) shows that it suffices to use the vorticity of the velocity field on a set of closed paths in the image to estimate rotational motion parameters (flow circulation algorithm) and that sparse measurements can be used to estimate the focus of expansion (FOE) and therefore , translational motion parameters and relative depth . Furthermore , Hammel used a global data field approach in calculating these parameters instead of local measurements , leading to very robust results . Hammel showed that if a certain set of conditions hold , the curl of the flow field at any point in the image is a linear combination of the rotational parameters (figure 3.4) .

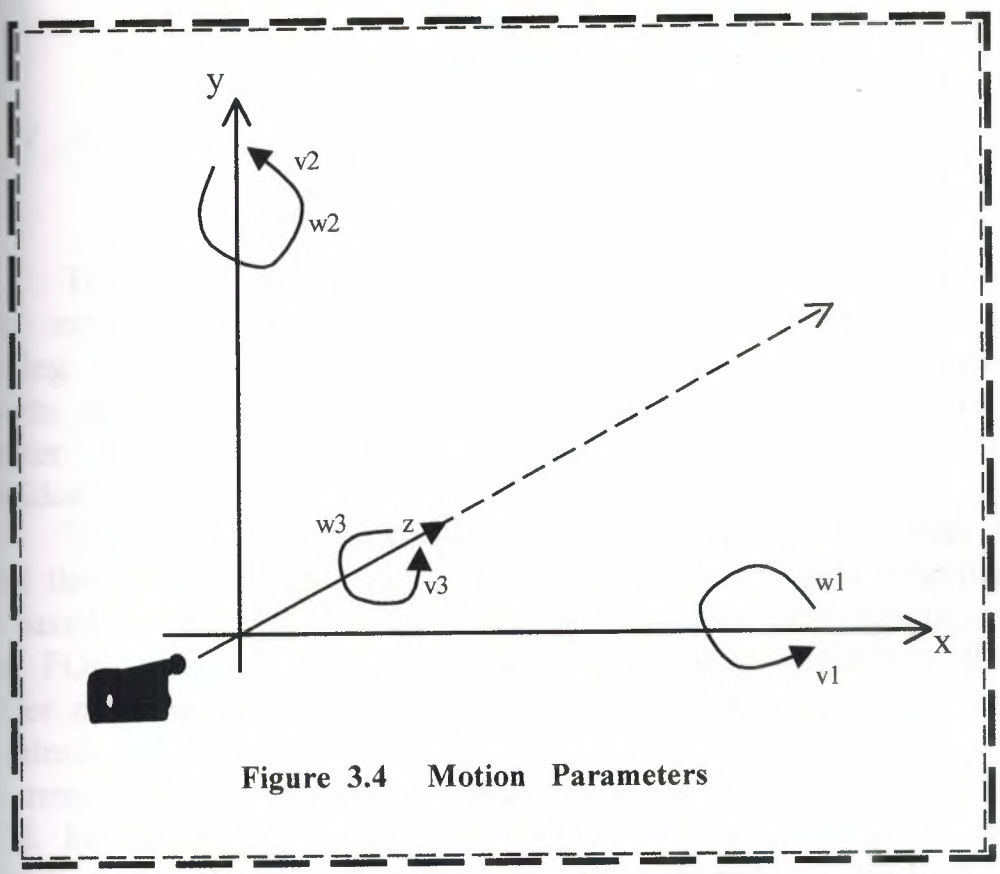


Figure 3.4 Motion Parameters

$$\text{Curl}(v) = \nabla x V(x,y) \approx \frac{-1}{f}(xw1 + yw2 + 2fw3) \qquad (1)$$

The average curl in a region D is then :

$$\frac{1}{|D|}\iint_D \text{curl}\,(V)\,dxdy \approx \frac{-1}{f}\,(x_0w1 + y_0w2 + 2fw3) \qquad (2)$$

Where x_0 and y_0 are the coordinates of the centroid of the region D . This average also forms a linear function with the rotational parameters . If it is calculated for a set of regions , linear regression can be done to fit the resulting set of linear equations to obtain a good estimate of the motion parameters . Furthermore , Stoke's theorem states that the total curl of a field in a region is equal to the circulation (vorticity) of the field on the boudary that surrounds the region . So that :

$$\oint_{\partial D} V \cdot ds = \iint_D \text{curl}\,(V)\, dxdy$$

This result is the basis for the flow circulation algorithm and means that motion parameters can be estimated without having to calculate the curl of the field , using only boundary points and without any loss of information or precission . As a matter of fact , noise due to calculation of the curl is being avoided .

The FOE search algorithm (Heeger et al . 1992) tries to find the focus of expansion (FOE) of the image . This algorithm is based on the fact that the circular component flow function at the FOE is precisely a quadratic polynomial . Algorithm then differ on how to find the point where the function is exactly a quadratic polynomial .

Hummel (1993) introduced three methods for calculating the FOE based on this criteria , two of which do not require any search and just use the values of this function on any set of points to calculate where the condition will be satisfied . Once the FOE is known , relative depth can be estimated using the FOE and Kalman filtering .

CHAPTER 4

MOBILE ROBOT NAVIGATION USING FUZZY LOGIC

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth - truth values between "completely true" and "completely false".

The guiding principle of soft computing is the combination is the combination of neural networks , Fuzzy Logic (FL) and Genetic Algorithm (GAs) in hybrid systems in order to benefit from the advantages of each methodology . About thirty years ago Lotfi Zadeh founded the theory of fuzzy sets , by extending the classical concept of a set . Unlike classical logic , in which elements either do or do not belong to a set , the degree of membership for elements of a fuzzy set can take on any value of the interval [0 , 1] . Fuzzy logic offers a framework for representing imprecise , uncertain knowledge . Similar to the way in which human beings make their decisions fuzzy systems are using a mode of approximate reasoning , which allows them to deal with vagueness and incomplete information . Fuzzy Control (FC) provides a flexible method to a model the relationship among input information and control output . Fuzzy Logic Controllers (FLCs) prove their robustness with regard to noise and variations of system parameters .

Genetic Algorithm (GAs) are optimisation methods inspired by the principles of natural evolution and genetics . Gas have been succesfully applied to solve a variety of difficult theoretical and practical problems by imitating the underlying processes of evolution such as selection , recombination and mutation . Their capability of learning enables a GA to adapt a system to deal with any desired task , that can be described by a scalar objective function . In contrast to other specialised optimisation methods GAs work well across a broad spectrum of problems and require no problem specific knowledge . Following the guideline of soft computing GAs have been widely used for automatic design of FLCs . The basic idea is to learn the rules of a knowledge based system by artificial evolution.

4.1 Fuzzy Logic Control

Fuzzy logic provides a means to deal with nonlinear functions . A fuzzy controller was designed to simulate the performance of the model of two – link Robotic manipulator . The

membership functions were developed for the effect of position errors and velocity parameters for two links of the robot . A membership function for the output of the controller i.e. the joint torque was also defined .

4.1.1 Fuzzy Expert Systems

Put as simply as possible, a fuzzy expert system is an expert system that uses fuzzy logic instead of Boolean logic. In other words, a fuzzy expert system is a collection of membership functions and rules that are used to reason about data. Unlike conventional expert systems, which are mainly symbolic reasoning engines, fuzzy expert systems are oriented toward numerical processing.

The rules in a fuzzy expert system are usually of a form similar to the following:

If x is low and y is high then z = medium

Where x and y are input variables (names for know data values), z is an output variable (a name for a data value to be computed), low is a membership function (fuzzy subset) defined on x, high is a membership function defined on y, and medium is a membership function defined on z. The part of the rule between the "if" and "then" is the rule's premise or antecedent. This is a fuzzy logic expression that describes to what degree the rule is applicable. The part of the rule following the "then" is the rule's conclusion or consequent. This part of the rule assigns a membership function to each of one or more output variables. Most tools for working with fuzzy expert systems allow more than one conclusion per rule.

A typical Fuzzy Expert System has more than one rule. The entire group of rules is collectively known as a rulebase or knowledge base.

4.2 Fuzzy Inference Process

With the definition of the rules and membership functions in hand, we now need to know how to apply this knowledge to specific values of the input variables to compute the values of the output variables. This process is referred to as inferencing. In a fuzzy expert system, the inference process is a combination of four subprocesses :

Fuzzification, Inference, Composition, and Defuzzification. The defuzzification subprocess is optional.

For the sake of example in the following discussion, assume that the variables x, y, and z all take on values in the interval [0, 10], and that we have the following membership functions and rules defined.

low(t) = 1 - t / 10
high(t) = t / 10

rule 1: if x is low and y is low then z is high
rule 2: if x is low and y is high then z is low
rule 3: if x is high and y is low then z is low
rule 4: if x is high and y is high then z is high

Notice that instead of assigning a single value to the output variable z, each rule assigns an entire fuzzy subset (low or high).

Notes:

1.In this example, low(t)+high(t)=1.0 for all t. This is not required, but it is fairly common.
2.The value of t at which low (t) is maximum is the same as the value of t at which high (t) is minimum, and vice-versa. This is also not required, but fairly common.
3.The same membership functions are used for all variables. This isn't required, and is also *not* common.

4.2.1 Fuzzification

In the fuzzification subprocess, the membership functions defined on the input variables are applied to their actual values, to determine the degree of truth for each rule premise. The degree of truth for a rule's premise is sometimes referred to as its alpha. If a rule's premise has a non-zero degree of truth (if the rule applies at all...) then the rule is said to fire .

For example:

x	y	Low (x)	High (x)	Low (y)	High (y)	Alpha 1	Alpha 2	Alpha 3	Alpha 4
0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0
0.0	3.2	1.0	0.0	0.68	0.32	0.68	0.32	0.0	0.0
0.0	6.1	1.0	0.0	0.39	0.61	0.39	0.61	0.0	0.0
0.0	10.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
3.2	0.0	0.68	0.32	1.0	0.0	0.68	0.0	0.32	0.0
6.1	0.0	0.39	0.61	1.0	0.0	0.39	0.0	0.61	0.0
10.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0
3.2	3.1	0.68	0.32	0.69	0.31	0.68	0.31	0.32	0.32
3.2	3.3	0.68	0.32	0.67	0.33	0.67	0.33	0.32	0.32
10.0	10.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0

4.2.2 Inference Mechanism

In the inference subprocess, the truth value for the premise of each rule is computed, and applied to the conclusion part of each rule. This results in one fuzzy subset to be assigned to each output variable for each rule.

I've only seen two inference methods or inference rules: MIN and PRODUCT. In MIN inferencing, the output membership function is clipped off at a height corresponding to the rule premise's computed degree of truth. This corresponds to the traditional interpretation of the fuzzy logic AND operation. In PRODUCT inferencing, the output membership function is scaled by the rule premise's computed degree of truth.

Due to the limitations of posting this as raw ASCII, I can't draw you a decent diagram of the results of these methods. Therefore I'll give the example results in the same functional notation I used for the membership functions above.

For example, let's look at rule 1 for x = 0.0 and y = 3.2. As shown in the table above, the premise degree of truth works out to 0.68. For this rule, MIN inferencing will assign z the fuzzy subset defined by the membership function:

rule1(z) = { z / 10, if z <= 6.8
0.68, if z >= 6.8 }

For the same conditions, PRODUCT inferencing will assign z the fuzzy subset defined by the membership function:

rule1(z) = 0.68 * high(z)
= 0.068 * z

For the same conditions, PRODUCT inferencing will assign z the fuzzy subset defined by the membership function:

rule1(z) = 0.68 * high(z)
= 0.068 * z

4.2.3 Composition

In the composition subprocess, all of the fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable.

We are familiar with two composition rules: MAX composition and SUM composition. In MAX composition, the combined output fuzzy subset is constructed by taking the pointwise maximum over all of the fuzzy subsets assigned to the output variable by the inference rule. In SUM composition the combined output fuzzy subset is constructed by taking the pointwise sum over all of the fuzzy subsets assigned to the output variable by the inference rule. Note that this can result in truth values greater than one! For this reason, SUM composition is only used when it will be followed by a defuzzification method, such as the CENTROID method, that doesn't have a problem with this odd case.

For example, assume x = 0.0 and y = 3.2. MIN inferencing would assign the following four fuzzy subsets to z:

rule1(z) = { z / 10, if z <= 6.8
0.68, if z >= 6.8 }

rule2(z) = { 0.32, if z <= 6.8
1 - z / 10, if z >= 6.8 }

rule3(z) = 0.0

rule4(z) = 0.0

MAX composition would result in the fuzzy subset:

fuzzy(z) = { 0.32, if z <= 3.2
z / 10, if 3.2 <= z <= 6.8
0.68, if z >= 6.8 }

PRODUCT inferencing would assign the following four fuzzy subsets to z:

rule1(z) = 0.068 * z
rule2(z) = 0.32 - 0.032 * z
rule3(z) = 0.0
rule4(z) = 0.0

SUM composition would result in the fuzzy subset:

fuzzy(z) = 0.32 + 0.036 * z

4.2.4 Defuzzification

Sometimes it is useful to just examine the fuzzy subsets that are the result of the composition process, but more often, this _fuzzy value_ needs to be converted to a single number - a crisp value. This is what the defuzzification subprocess does.

There are more defuzzification methods than you can shake a stick at. A couple of years ago, Mizumoto did a short paper that compared roughly thirty defuzzification methods. Two of the more common techniques are the CENTROID and MAXIMUM methods. In the CENTROID method, the crisp value of the output variable is computed by finding the variable value of the centre of gravity of the membership function for the fuzzy value. In the MAXIMUM method, one of the variable values at which the fuzzy subset has its maximum truth value is chosen as the crisp value for the output variable. There are several variations of the MAXIMUM method that differ only in what they do when there is more than one variable value at which this maximum truth value occurs. One of these, the AVERAGE-OF-MAXIMA method, returns the average of the variable values at which the maximum truth value occurs.

For example, go back to our previous examples. Using MAX-MIN inferencing and AVERAGE-OF-MAXIMA defuzzification results in a crisp value of 8.4 for z.

Using PRODUCT-SUM inferencing and CENTROID defuzzification results in a crisp value of 6.7 for z.

Note: sometimes the composition and defuzzification processes are combined, taking advantage of mathematical relationships that simplify the process of computing
the final output variable values.

4.3 Software Implementation Of The Fuzzy Control System

The controller software was divided into two primary modules written in the C language . The two modules are the fuzzy control module and the communications module .

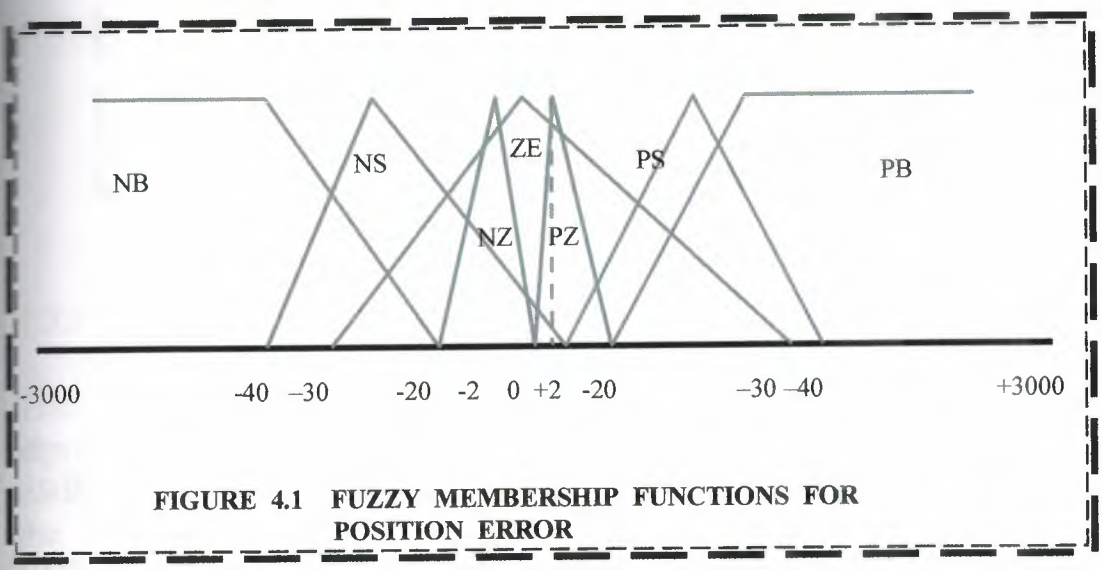


FIGURE 4.1 FUZZY MEMBERSHIP FUNCTIONS FOR POSITION ERROR

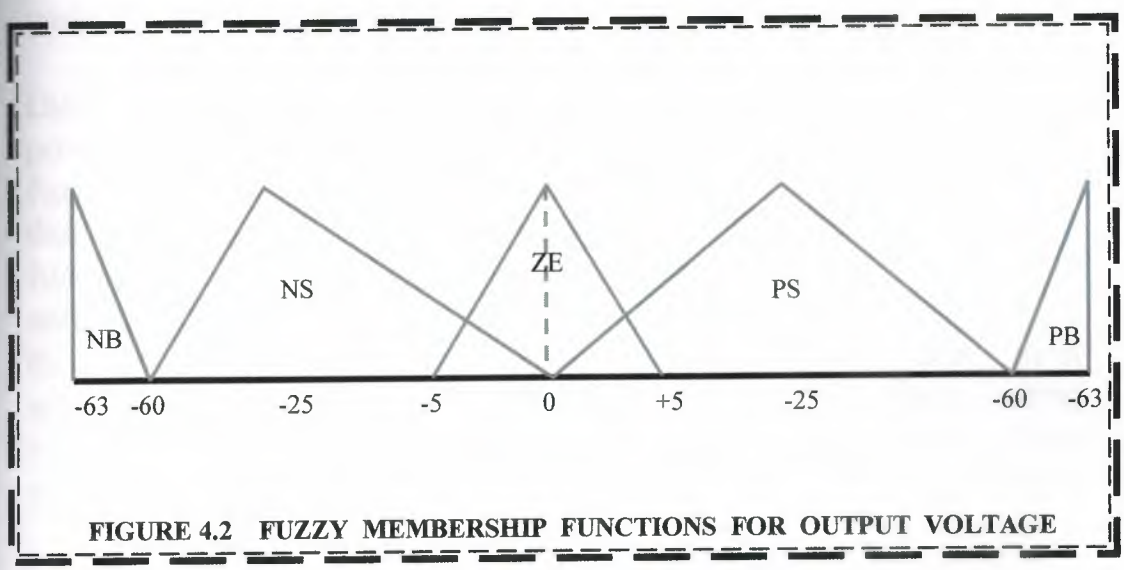


FIGURE 4.2 FUZZY MEMBERSHIP FUNCTIONS FOR OUTPUT VOLTAGE

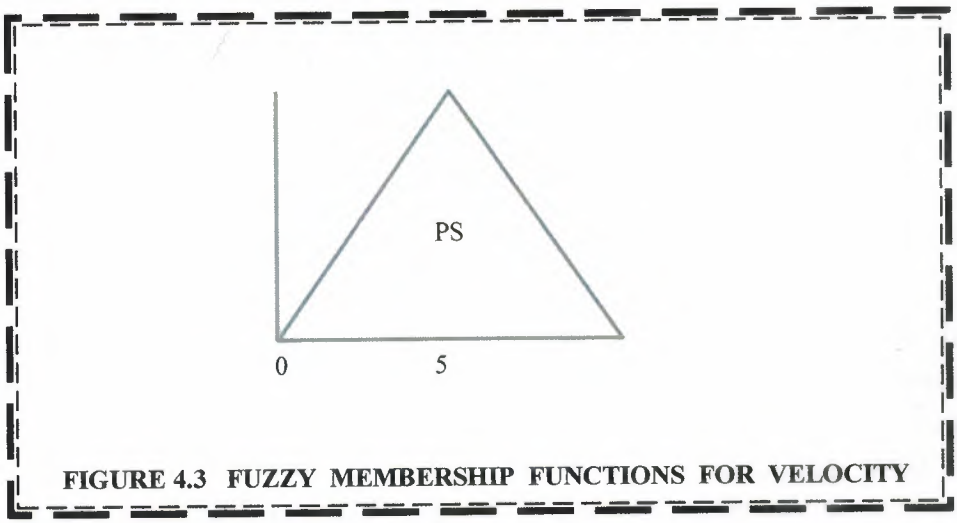


FIGURE 4.3 FUZZY MEMBERSHIP FUNCTIONS FOR VELOCITY

4.3.1 Fuzzy Control Module

The first step in fuzzy controller design was the selection of input and output variables . The designated input variables were ERROR (robot link position error) and VELOCITY (velocity of the robot link) . The output variable was chosen to be VOLTAGE (drive voltage output by the power supply) . Each variable was accompanied with a set of membership functions .

During actual operation , the computer would read in the robot position data and then compute position error and velocity . The fuzzy controller fuzzified the input quantities through algorithms that operated on the data as specified by the membership functions . Next , the fuzzified input quantities passed through a series of fuzzy controller routinely assessed the current state of the robot and determined which control action was most appropriate . Defuzzification was applied using the voltage output variable and a control action was selected . The action was then passed through the interface to the external power supply driver .

4.4 Fuzzy Logic For Local Guidance of Mobile Robot

A method that tries to make a mobile robot reach a certain position with a specified bearing and speed while detouring obstacles is presented . This action is on a reactive level and will be referred to as local guidance . The module that implements it forms part of a four level hierarchy consisting of different levels of control :

- Mission Planning : Determines the tasks to be done and works in terms of hallways , rooms , docking zones , etc.
- Global Path Planning : Calculates a set of control points through which the vehicle should try to pass when executing a path . These control points include desired position , bearing , speed and a type of precision (passing or docking point) . User desired or historically good paths are strongly considered . These control points will be referred to as poses .
- Local Guidance : Takes each pose fed by the Global Path Planning module and tries to reach it as desired . When a pose is surpassed , unreachable or missed , it prompts the Global Path Planning module fo a new one .
- Servo Control : Offers a virtual mobile base accepting orders regarding desired bearing and speed . These orders are issued by the Local Guidance module .

Since there may be failure reaching a pose (maybe there is an obstacle on top of or near the point) , the Global Path Planning module may have to recalculate the next point .

This structure is similar to many previous detailed – map – based approaches but a significant difference exists in that order approaches assume that the Global Path Planning module can provide a large set of control points to succesfully cross a room while this approach will suffice with just the final destination point. In this sense , the method uses information if it is available and manages without it if not . Therefore it can take advantage of existing path planning techniques (Munoz et al 1993 ; Shin et al 1992) .

4.4.1 Local Guidance

The Local Guidance module repeatedly takes a pose from the Global Path Planning module and issues steer and speed orders to the Servo Control module until that pose is reached or missed , until it is interrupted or when the pose is determined unreachable (bail – out) . It is then given a new desired pose .

In this action , current position , bearing and speed are compared with desired final values in a way that the system will react slowly when far away but will progressively increase necessary actions when close . This allows it to avoid drastic maneuvers unless strictly necessary and thus provides smooth actions that save energy , increase stability and provide better navigation . Range information is also used in order to determine the possibility of colliding in a set of directions and thus maneuver to avoid obstacles .

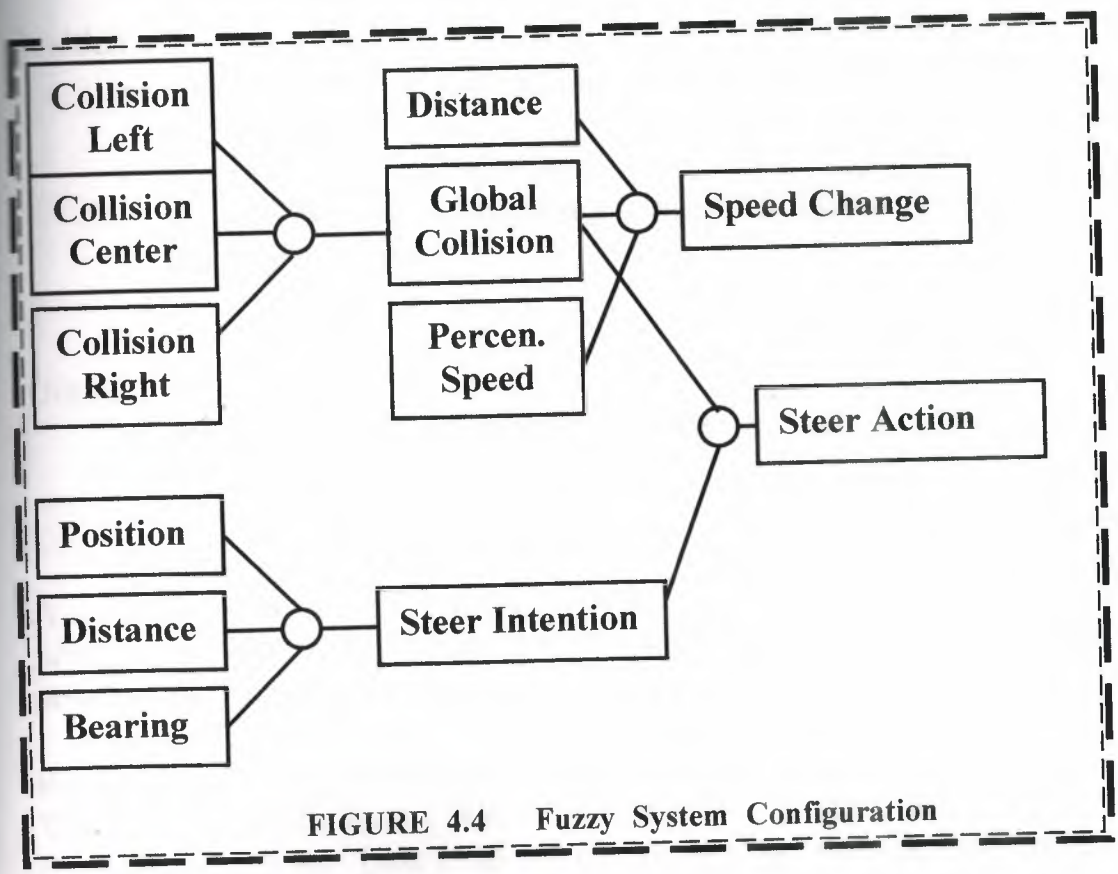


FIGURE 4.4 Fuzzy System Configuration

4.4.2 Fuzzy System For Local Guidance

Fuzzy Sets :

A fuzzy system has been designed to implement this method using the configuration shown in Fig 4.4 . It uses the following input , hidden and output sets :

Input :

- Bearing : Robot's bearing with respect to the desired bearing .
- Position : Robot's position with respect to the desired position and bearing .
- Distance : Distance from the final position .
- Speed : Robot's speed with respect to the desired speed .
- Possibility of Collision : At left , right or center .

Hidden :

- Global Collision : Integrates possibilities of collision in all directions .
- Steer Action : Determines a position – oriented steer action (without considering collision) that is relative to the target point .

Output :

- Steer : Absolute change in bearing .
- New Speed : Change in speed .

A coordinate system (Fig 4.5) is set up so that the desired point's position is the center and it's direction is north , the true position and bearing are transformed to such system .

Regarding bearing , the vehicle may be directed South West (SW) , West (W) , North West (NW) , North (N) , North East (NE) , East (E) or Sout East (SE) with repect to the desired final bearing . (fig 4.5 , fig 4. 6 – 10)

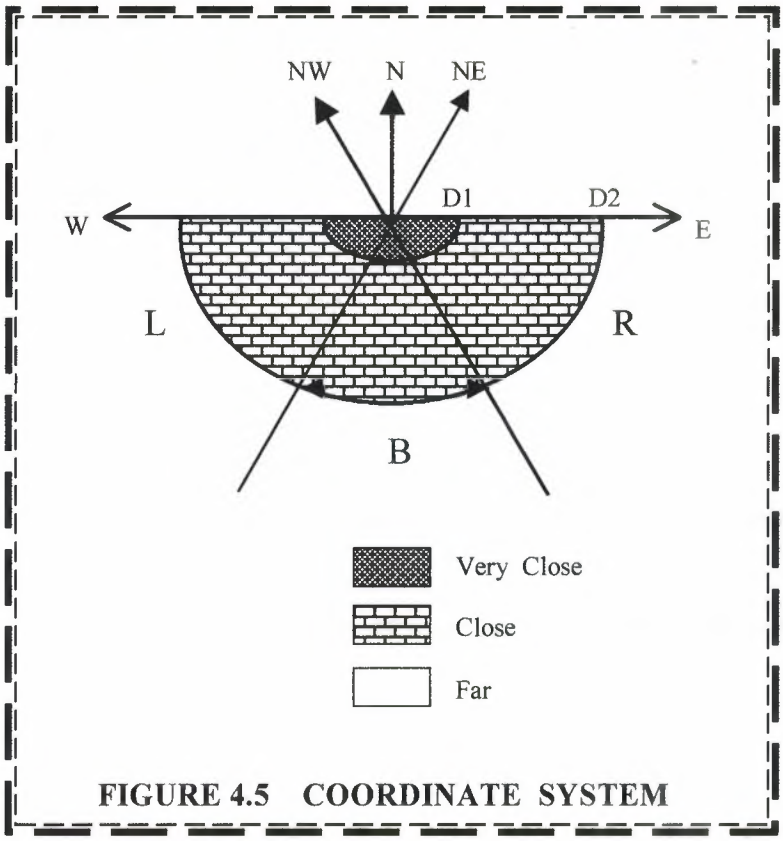


FIGURE 4.5 COORDINATE SYSTEM

The vehicle's position may be left (L) of the final position , Centered (C) , Right (R) of the final position . (fig 4.5 , fig 4. 6-10) . This final case terminates the guidance action and triggers a request for a new position .

Distance to the objective can be considered Very Close (VC) , Close (C) , and Far (F) (fig 4.5 , fig 4. 6-10) . The values of the angle B , and distances D1 and D2 (fig 4.5) must take into account the kinematic and dynamic properties of the robot and also the environment where the vehicle will move .

INPUT FUZZY SETS :

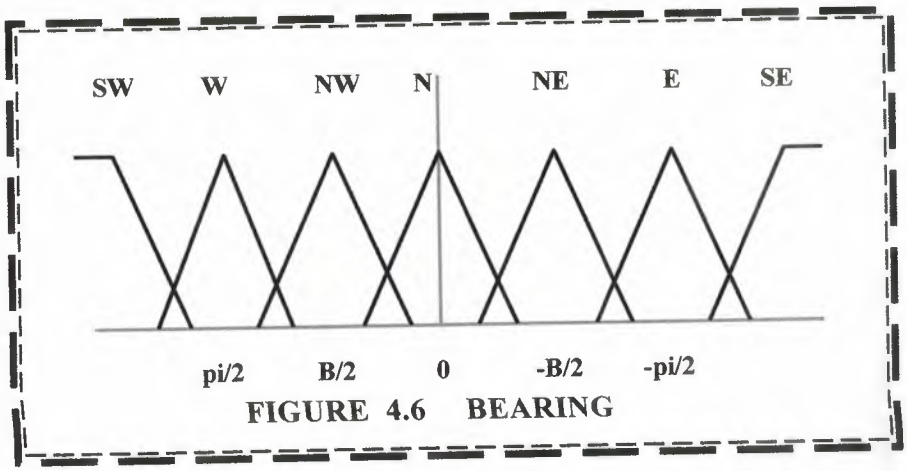


FIGURE 4.6 BEARING

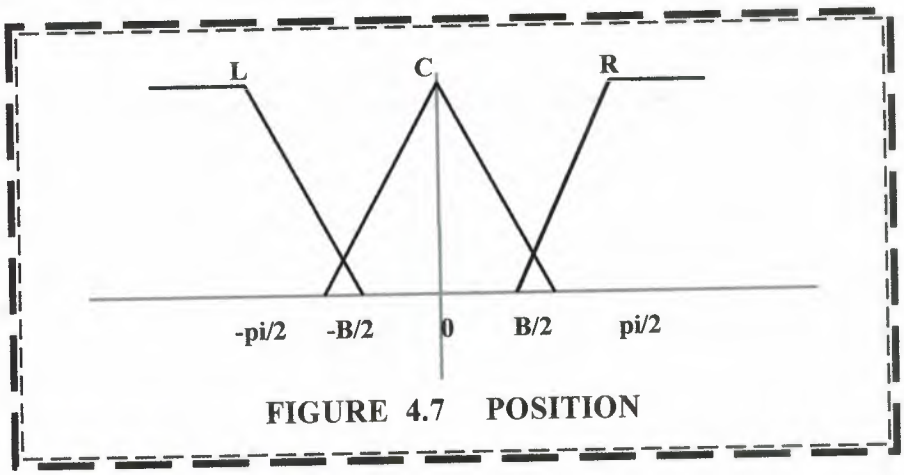


FIGURE 4.7 POSITION

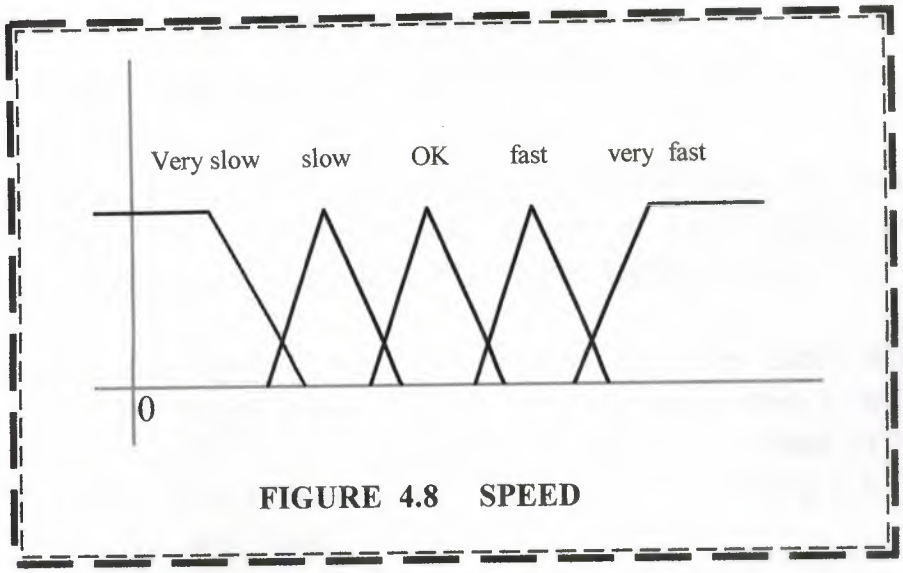


FIGURE 4.8 SPEED

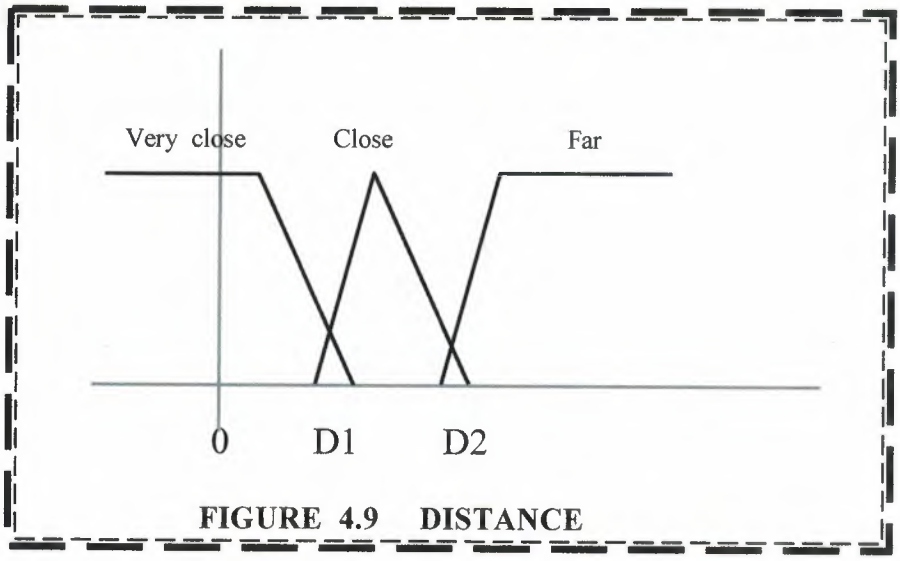


FIGURE 4.9 DISTANCE

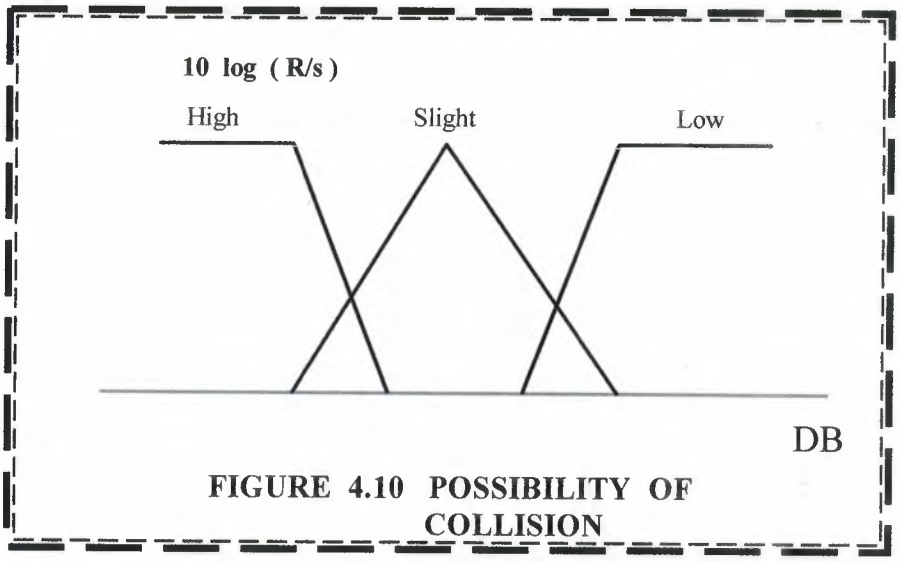


FIGURE 4.10 POSSIBILITY OF COLLISION

Collision (fig 4.11 , 4.12) summarizes the possibilities of collision at left , center and right . It can take values of Left Center (LC) , Left (L) , Low (Low) , Right (R) , Right Center (RC) .

Steer intention (fig 4. 11-12, fig 4.13 - 19) specifies steer actions relative to the target point (north and distance zero) without considering possibility of collision . It can take values of Steer Sideways East (SSE) , Steer East (SE) , Steer to Bearing (STB) , Steer West (SW), and Steer Sideways West (SSW) .

HIDDEN FUZZY SETS :

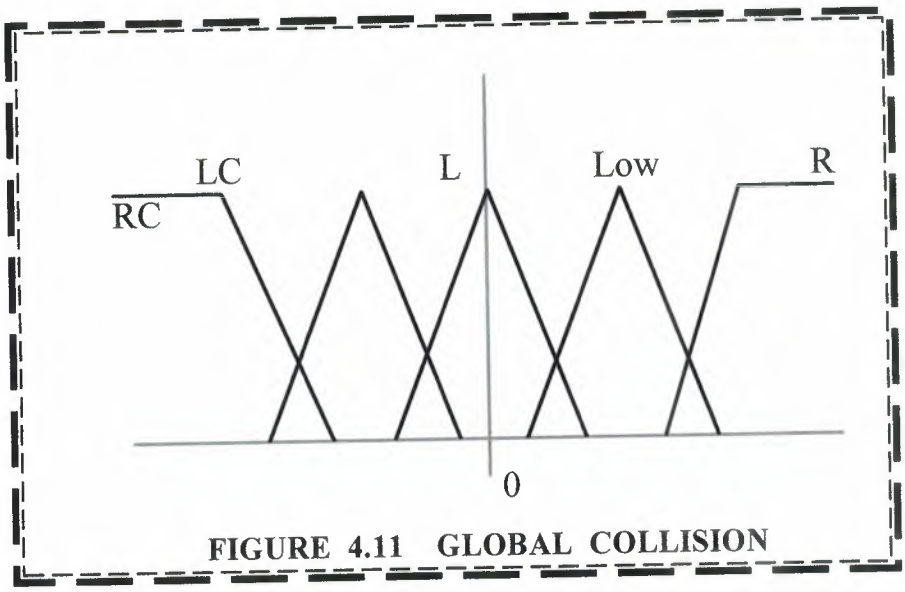


FIGURE 4.11 GLOBAL COLLISION

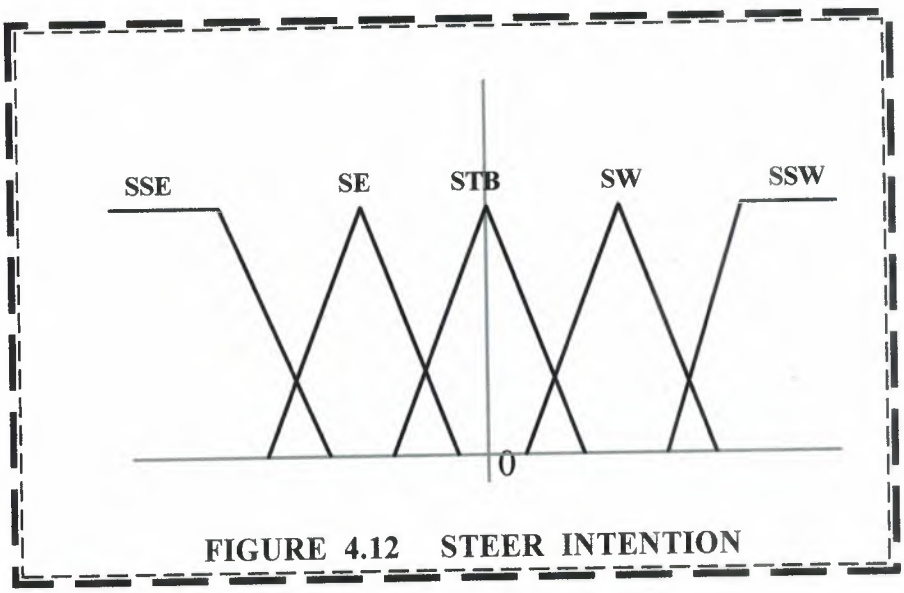


FIGURE 4.12 STEER INTENTION

De – fuzzification is done using the center of gravity method .

Fuzzy Rules

Rules can be divided in those that contribute to steering and those that contribute to change in speed .

Strategy used for steering :

- Drive straight towards the objective if far away .
- When close , and if not centered , make a side movement until in the centered zone and then turn back towards target .

When Very Close , steer to obtain correct bearing even if not centered .

Strategy used for speed :

- Make small changes towards the final speed when far away and increase them when close .
- Decrease Speed drastically when a frontal collision is possible .

The first set of rules (fig 4. 13 - 19) use Bearing , Position and Distance to determine Steer Intention .

Fuzzy Rules for

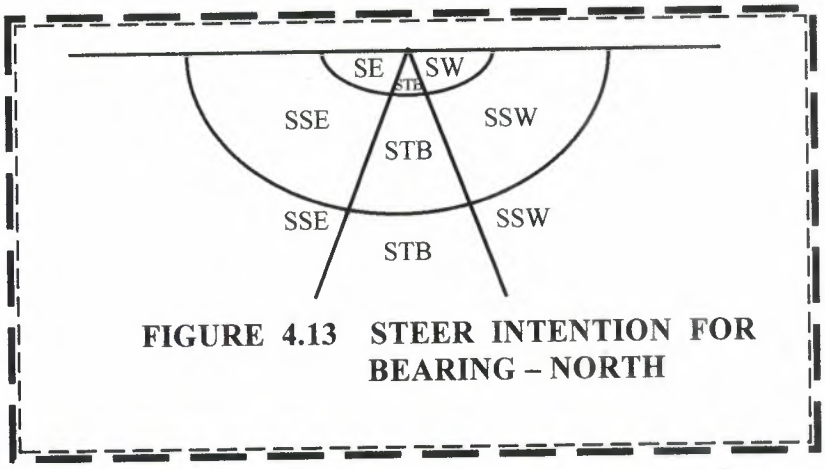


FIGURE 4.13 STEER INTENTION FOR BEARING – NORTH

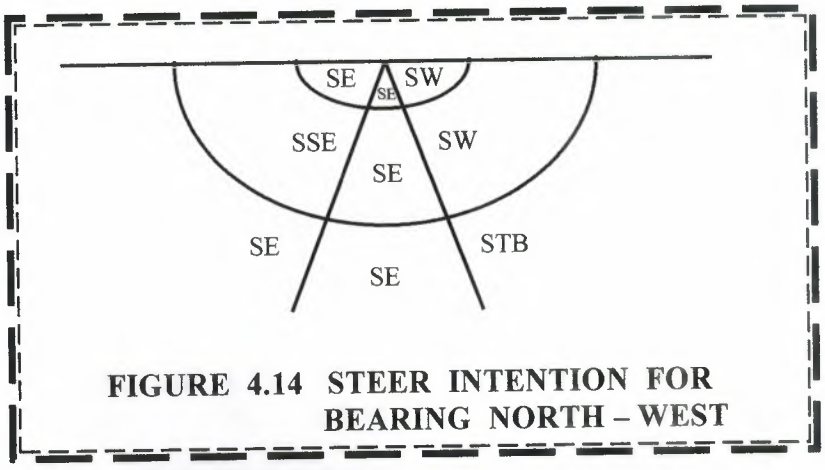


FIGURE 4.14 STEER INTENTION FOR BEARING NORTH – WEST

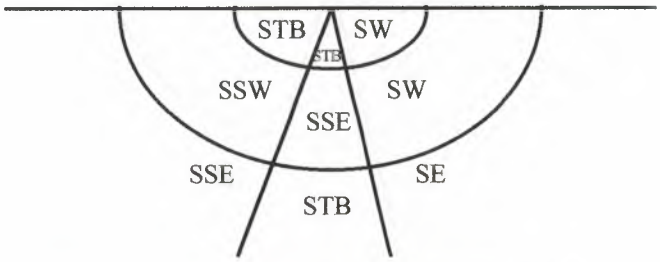


FIGURE 4.15 STEER INTENTION FOR BEARING – WEST

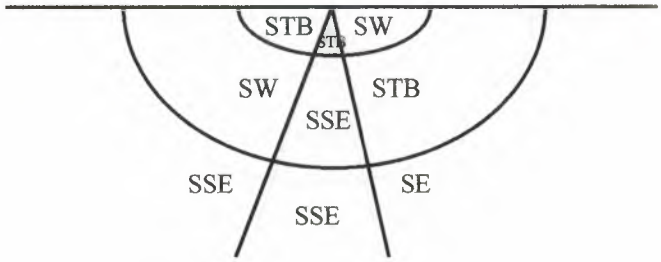


FIGURE 4.16 STEER INTENTION FOR BEARING – SOUTH WEST

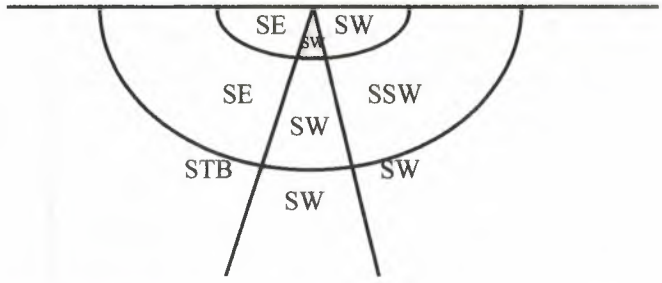


FIGURE 4.17 STEER INTENTION FOR BEARING – NORTH EAST

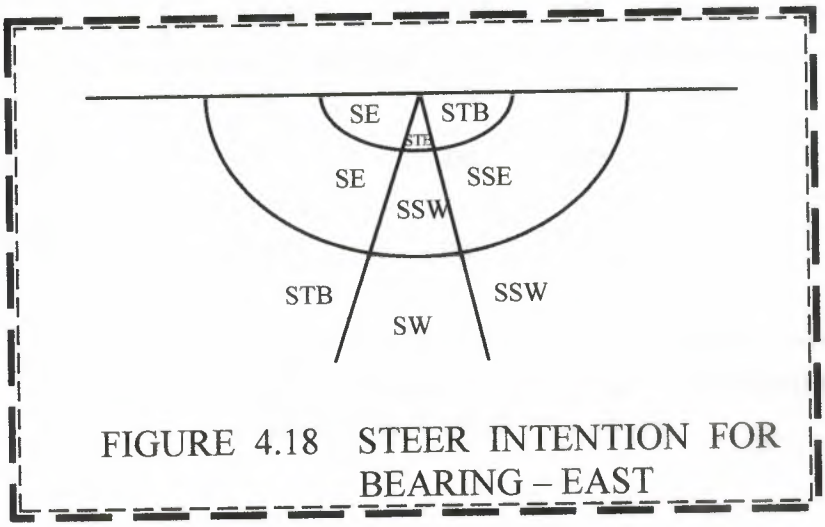


FIGURE 4.18 STEER INTENTION FOR BEARING – EAST

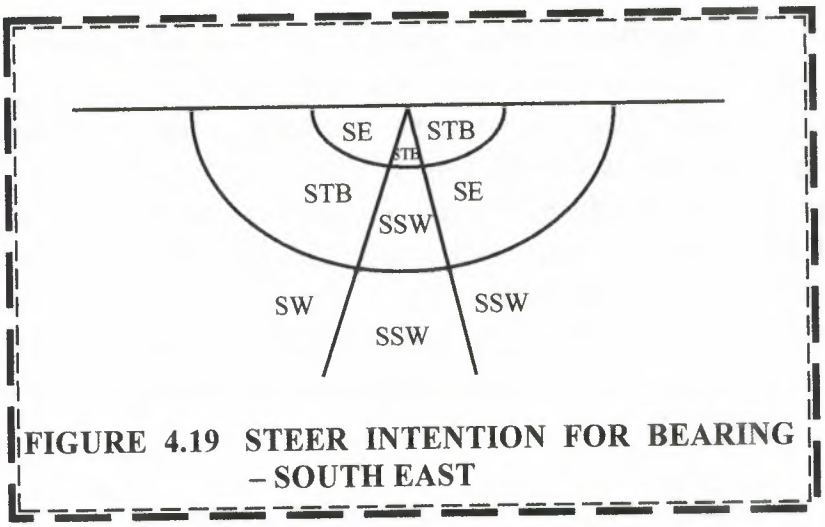


FIGURE 4.19 STEER INTENTION FOR BEARING – SOUTH EAST

The other set of rules use Distance , Possibility of Collision at Center and Speed to determine New Speed . These can be seen in table 4.1 and table 4.2 .

	V.SLOW	SLOW	OK	FAST	V.FAST
V. CLOSE	IS	IS	NA	DS	DS
CLOSE	IS	SIS	NA	SDS	DS
FAR	SIS	NA	NA	NA	SDS

Table 4.1 Speed Change as a function of Distance and Speed

	V.SLOW	SLOW	OK	FAST	V.FAST
Low	NA	NA	NA	NA	NA
Sight	NA	SDS	SDS	DS	DS
High	SDS	DS	DS	DS	DS

Table 4.2 Speed Change as a function of Possibility of Collision at Center and speed

CONCLUSION

This method allows real – time local guidance at low cost regarding both sensors and computation . Although this method can manage without complex sensors , it will always take good advantage of more advanced (and costly) sensors because in this case it will just provide a smooth path for an obstacle free path , given by the Global Path Planning module , while being able to manage situations with unexpected obstacles . Also noteworthy is that the method not only tries to reach points but does so at certain orientation and speed .

CHAPTER 5

COMPUTER – AIDED CONTROL OF MOBILE ROBOT

5.1 Structure of Control System of Mobile Robot

Transmitter Block is used to transmit data from the keyboard when we push the one of the up – down – left – right buttons to remote control . Our program read the command that we send and gives it to the parallel port as a 5V . When 5V. Transmitted from parallel port 10 kΩ resistors pulls down the current and the transistors with these resistors acts as a switch to control the circuit . Then the integrated circuit in the transmitter control process the command and by understanding its frequency . Then transmitter antenna sends this command through the air (analog transmission) to the receiver of the mobile robot (car) .

The Mobile Robot receive the command through its receiver (receiver antenna) . It tooks the command and process frequency to understand which direction it will go . Then , according to the command given by the user , it gives 5 V. To stepper motors and start moving . Now , we can see the processes in details .
(Fig 5.1)
Control Program of Mobile Robot :

Our program is written in C ++ to control the mobile robot's movement through the parallel port by using the ASCII codes of the forward – backward – left – right buttons .
When we push any of these buttons , it sends a command through parallel port until the next button pushed . It transmits commands with the command OUTPORT (address of parallel port , pin number of the parallel port) .

OUTPORT (888 , 1)

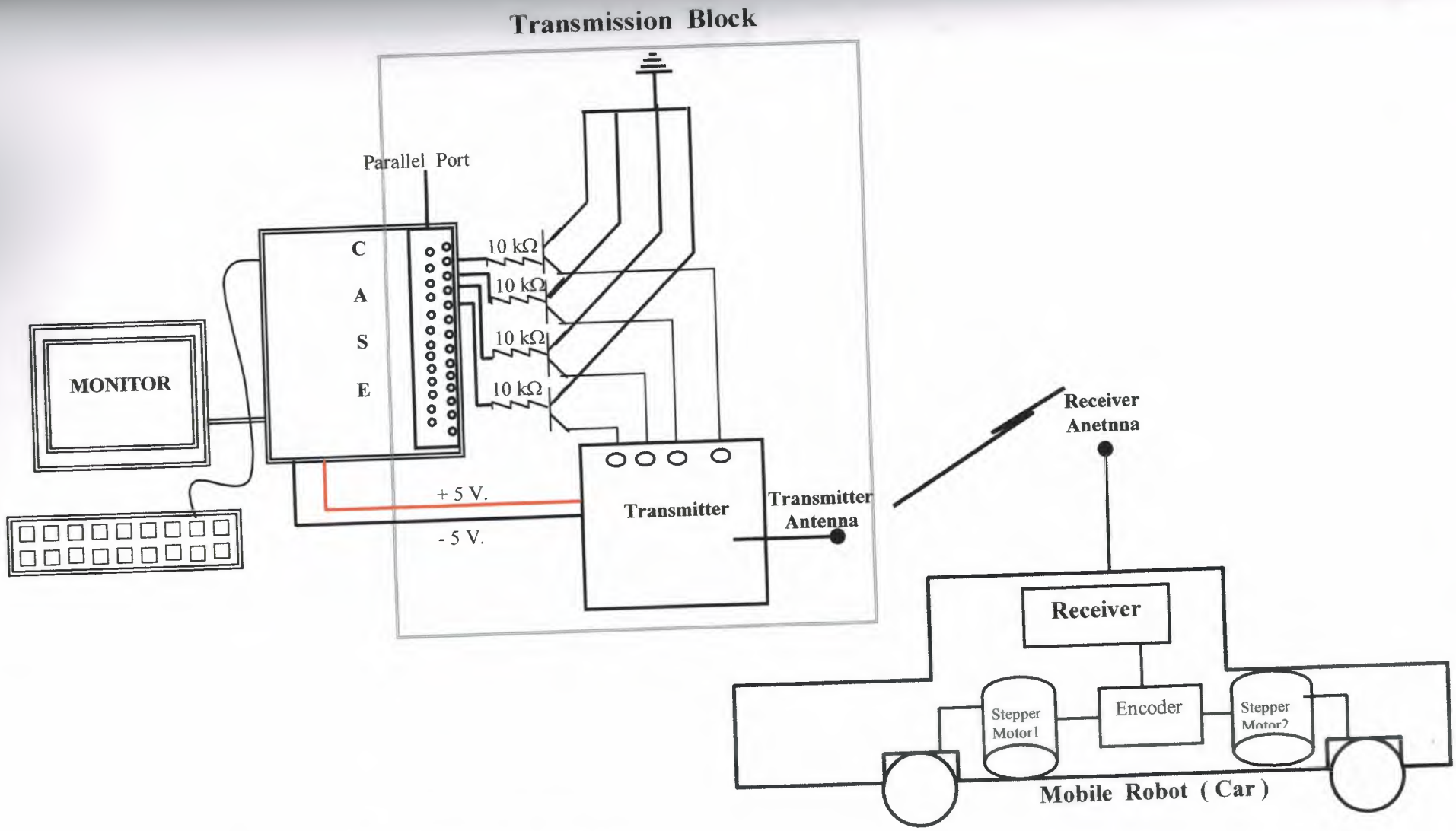


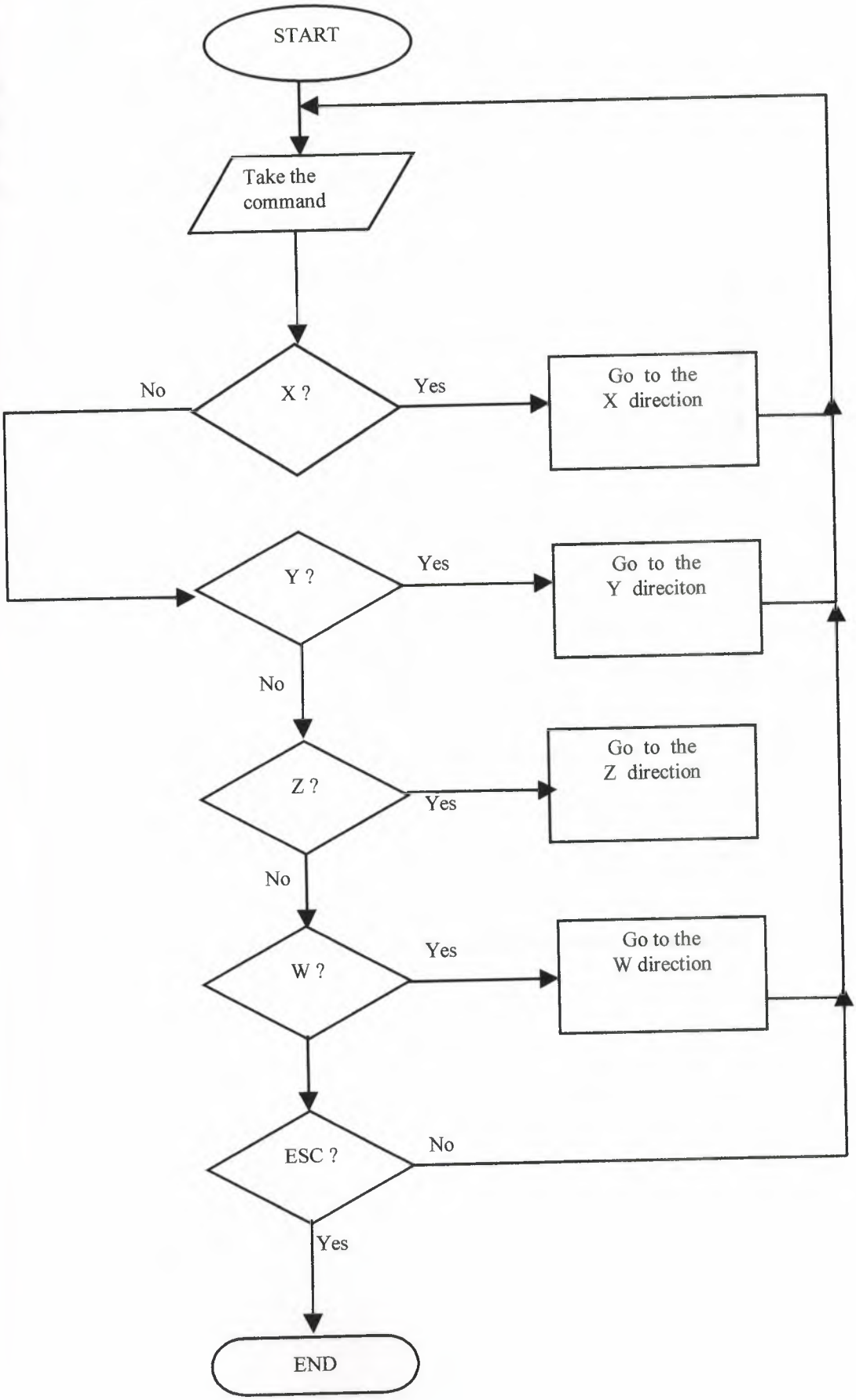
Figure 5.1 **Block Diagram of our project**

Algorithm of Control Program of Mobile Robot :

1. START
2. Take the command /*program will wait the command from the user */
3. if (x) then outport (888 ,a) /*program sends this command and mobile robot starts to move according to the given command */
4. else if (y) then outport (888 ,b) /*program cuts the previous command and send this command to mobile robot to move this direction */
5. else if (z) then outport (888 , c) /*program cuts the previous command and send this command to mobile robot to move this direction */
6. else if (w) then outport (888 , d) /*program cuts the previous command and send this command to mobile robot to move this direction */
7. Continue until terminating (ESC) .
8. END.

Where x,y,z and w are the direction and the a,b,c, and d are the number of the pin of the parallel port that transmits data .

FLOWCHART OF CONTROL PROGRAM OF MOBILE ROBOT :



5.1.1 Transmitter

Our transmitter (Fig. 5.2) has an encoder to encode the commands that given by the user through the parallel port . And it transmits the command to the transmitter antenna . And transmitter antenna sends the command to the Mobile Robot .

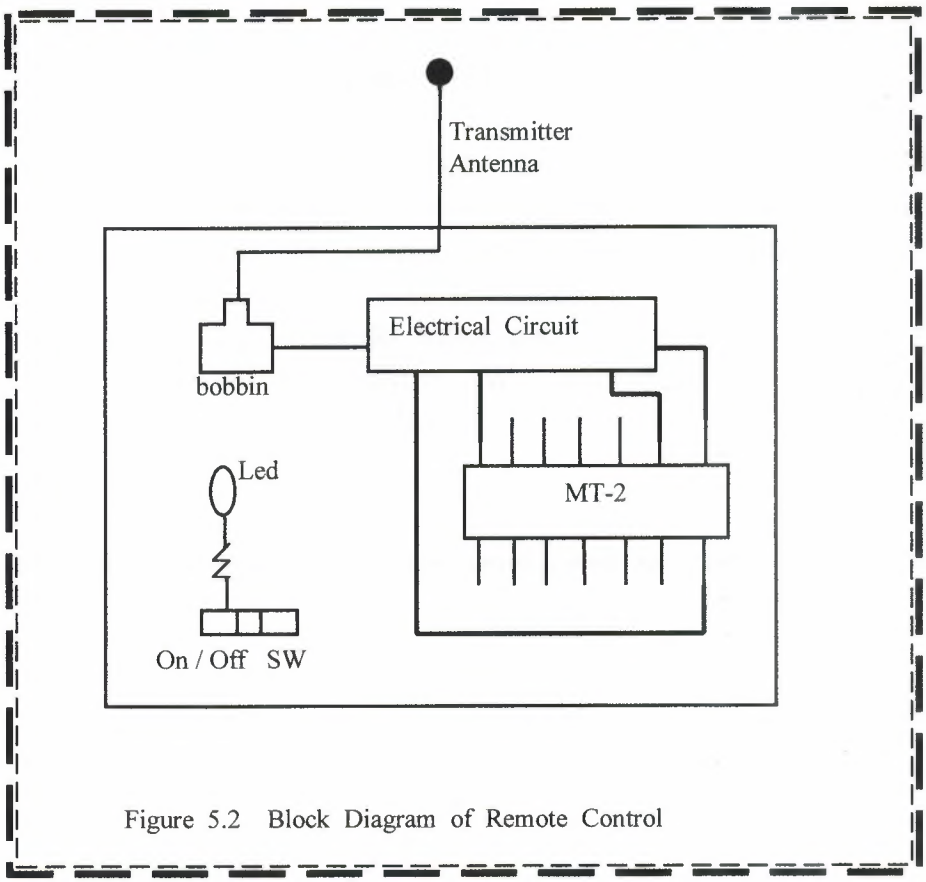


Figure 5.2 Block Diagram of Remote Control

5.2 Technical Characteristics of Mobile Robot

Mobile Robot that used in this project has four wheels . Two of them is at the back which controlled by a single stepper motor to move forward and backward . And the other two wheels also controlled by a stepper motor to determine the directions left or right . The stepper motors work by 9V (6 x 1.5 V battery) .
It receives a command from the transmitter which located in the remote control of control block by receiver antenna set above the mobile robot .

Behaviour of the used Mobile Robot

1. Speed : Mobile robot can go maximum 5 km / h when the batteries of our mobile robot is full . And when we calculate the average speed , we have seen that it can go 3.5 km /h . (Figure)
2. Rotation : Mobile robot can rotate 30^0 to left or right as seen in the figure . We take the 0 point at the center of the mobile robot .

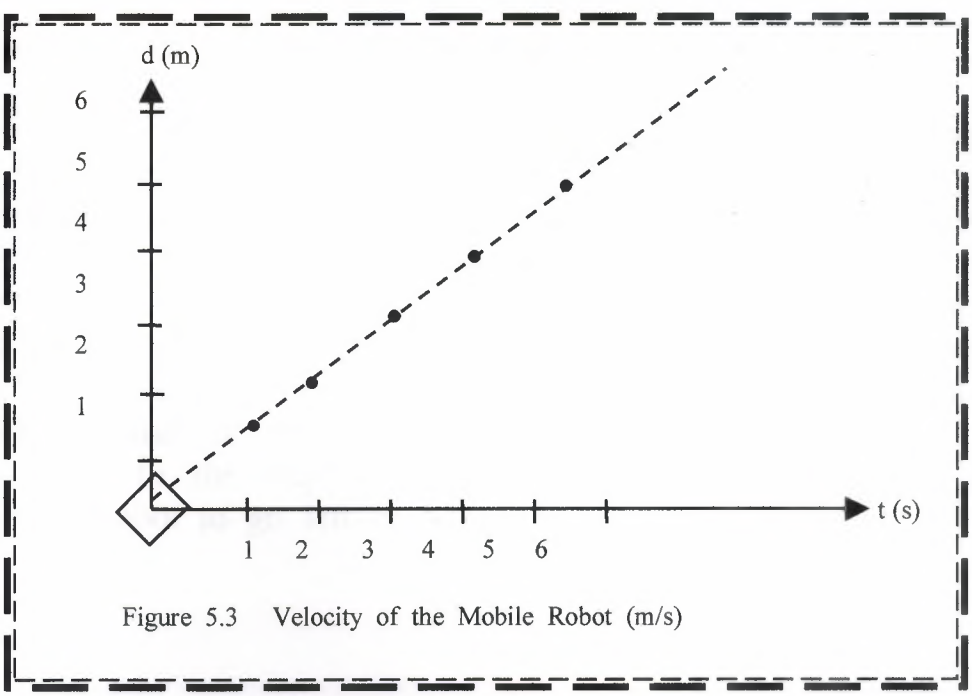


Figure 5.3 Velocity of the Mobile Robot (m/s)

Where d is the distance (m) and t is the time (sec) .

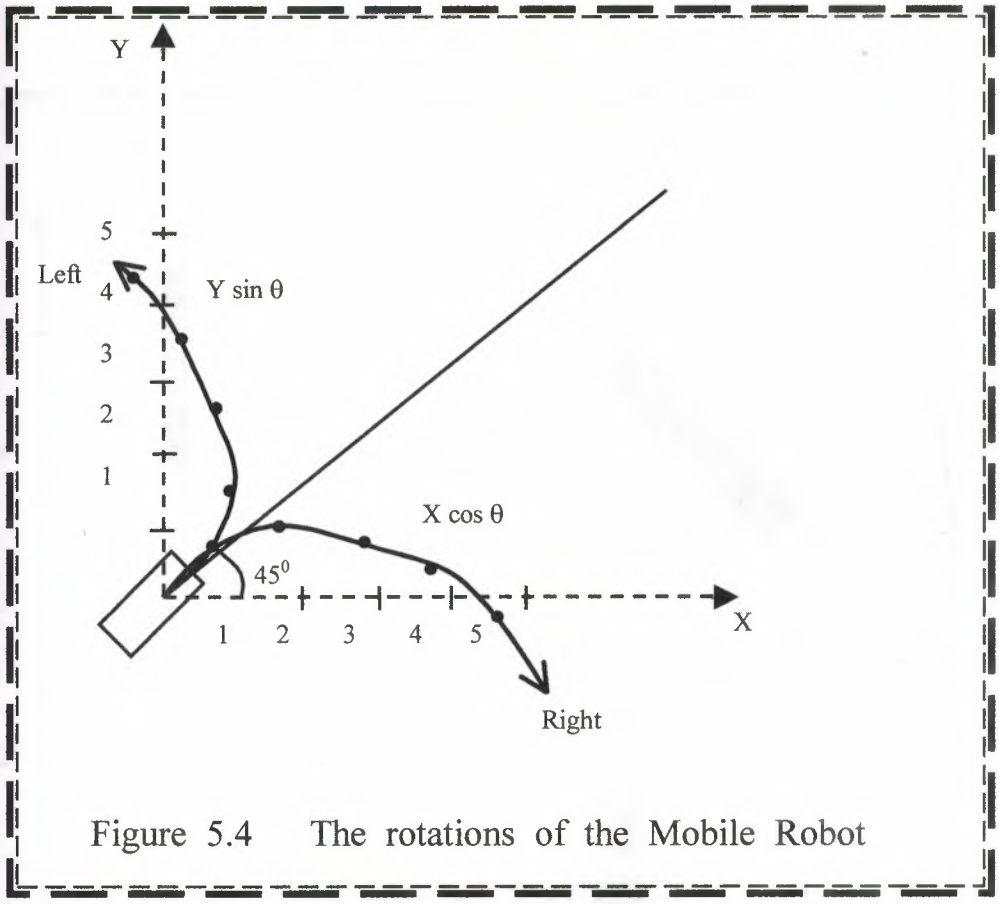
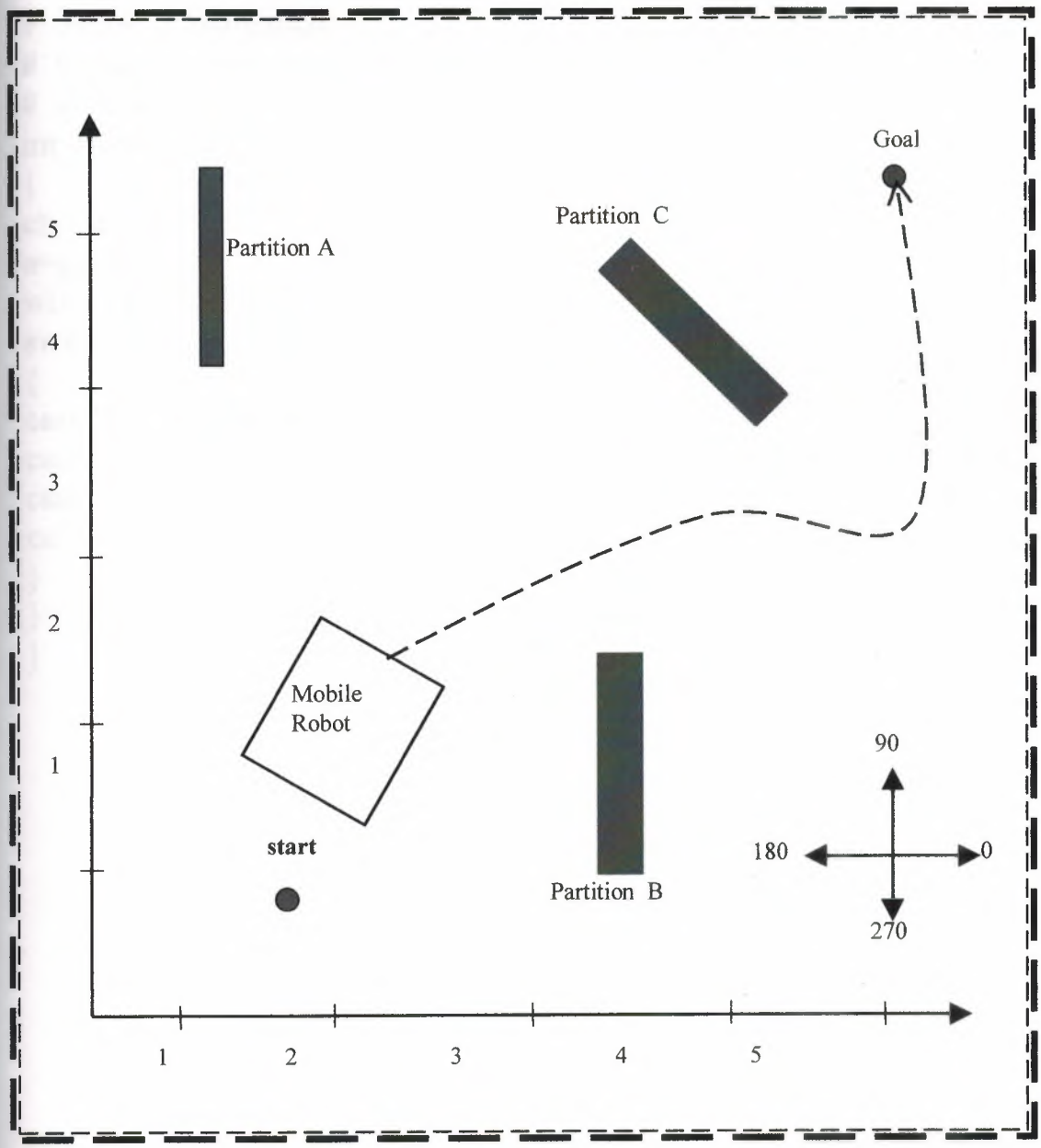


Figure 5.4 The rotations of the Mobile Robot

How does the Mobile Robot work ?

We said that , when we send a command from keyboard through parallel port , transmitter block transmits analog signal in air with transmitter antenna . The transmitted signal is received by the Mobile Robots receiver antenna . And receiver gives it to the encoder to understand what is the command . And encoder gives commands to the stepper motors . If the command is forward or backward , it gives command to stepper motor to go forward or backward and if the command is turn right or left , it gives command to the stepper motor 2 that rotates the wheels of the Mobile Robot to go left or right .


Goal
Partition A
Partition C
5
4
3
2
1
Mobile
Robot
start
Partition B
90
180
0
270
1
2
3
4
5

FRAGMENT PROGRAM :

```
# include <iostream.h>
# include <conio.h>
# include <stdio.h>
int main (void)
{
char a;
a=getch();
while (a=! ESC)
switch(a)
{
case 'f ' : outport(888,1); break;
case 'g ' : outport(888,2); break;
case 'h' : outport(888,3); break;
case 'j' : outport(888,4); break;
}
}
}
```

CONCLUSION

Analysis of navigation of mobile robots show that , they are characterized with uncertainty of environment functioning . In these condition to solve control problem of mobile robots it is necessary to use the ideas of Artificial Intelligence .

In the project , the navigation system of mobile robot in the condition of uncertainty is developed . The structure and control algorithm of mobile robot is presented . Also by using fuzzy logic , the development of control system of mobile robot is carried out . The structure of fuzzy actual system of robot is given out . The control rules of speed , steer action is described .

The fuzzy processing maechanism is based max – min composition . The structure of computer – aided control control of mobile robot and the functions of its main blocks is described . The interface between computer and mobile robot is developed . Software realisation of system is realized by using C++ language . The obtained simulation results show the efficieny of applied approach .

References:

[1] A Local Guidance Method for Low – Cost Mobile Robots using Fuzzy Logic
F. VAZQUEZ , E. GARCIA
ARTIFICIAL INTELLIGENCE IN REAL – TIME CONTROL 1994

[2] Road following by Artificial Vision using Neural Network
M.MAZO , F. J. RODRIGUEZ , E.SANTISO , M. A. SOTELO
ARTIFICIAL INTELLIGENCE IN REAL – TIME CONTROL 1994

[3] Navigation with Uncertain Position Estimation in the RAM – 1 Mobile Robot
V. MUNOZ , J. L. MARTINEZ , A. OLLERO
ARTIFICIAL INTELLIGENCE IN REAL – TIME CONTROL 1994

[4] Real – Time Vision – Based Navigation and the 3D Depth Estimation for an Indoor Autonomous Mobile Robot
F. VAZQUEZ , E. PAZ , R . MARIN
ARTIFICIAL INTELLIGENCE IN REAL – TIME CONTROL 1994

[5] Neural Networks for Robot Control
G. CEMBRANO , C. TORRAS , G. WELLS
ARTIFICIAL INTELLIGENCE IN REAL – TIME CONTROL 1994

[6] Decentralized Control of Distributed Intelligent Robot and Subsystems
Th. LAENGLE , T.C. LUETH
ARTIFICIAL INTELLIGENCE IN REAL – TIME CONTROL 1994

[7] Integrated Acquisitoin , Execution , Evaluation and Tuning of Elementary Skills for Intelligent Robots
M.KAISER , A.GIORDANA , M.NUTTIN
ARTIFICIAL INTELLIGENCE IN REAL – TIME CONTROL 1994

[8] Learning Task Applied to Identification of a Marine Vehicle
R. FERREIRO GARCIA , J. VIDAL PAZ
ARTIFICIAL INTELLIGENCE IN REAL – TIME CONTROL 1994

[9] Training Neurofuzzy Systems
D.J. MILLS , M.BROWN , C.J. HARRIS

[10] Increasing a Knowledge Representation for FMS Control with Fault Detection and Error Recovery Capabilities
ARTIFICIAL INTELLIGENCE IN REAL – TIME CONTROL 1994

[11] Artificial Intelligence and Mobile Robots
DAVID KORTENKAMP , R. PETER BONASSO , ROBIN MURPHY

[12] Telecommunications
PROF. DR. FAHRETTEN SADIGOGLU

[13] Communication Systems Engineering
JOHN G. PROAKIS , MASOUD SALEHI
Prentice Hall

[14] Real – Time Controller Implemented on Parallel Architecture
Z. HANZALEK
ARTIFICIAL INTELLIGENCE IN REAL – TIME CONTROL 1994

[15] www.ieee.com

[16] www.britanica.com

[17] http://www.lvr.com/parport.htm

[18] http://dynamik.fb10.tu-berlin.de/manuals/hardware/pp-AMP-36.html

[19] http://www.stokely.com/unix.serial.port.resources/A-B-Ycablepinout.html

[20] http://www.doc.ic.ac.uk/~ih/doc/par/

[21] http://margo.student.utwente.nl/stefan/hwb/menu_Cable.html

[22] http://www.hut.fi/Misc/Electronics/circuits/power_from_pc.html

[23] http://www.lammertbies.nl/comm/cable/parallel.html

[24] http://cyclone.parad.ru/hwb/ca_ParallelPortLoopbackCheckIt.html