

Application of Natural-Interfaced (Intelligent Adaptive) Agents to Domestic and Industrial Autonomous Robotics

This report is submitted in partial fulfilment of the requirements for the AMIETE Examination of The Institution of Electronics and Telecommunication Engineers

By

Abhishek Choudhary
SG-129546
(Computer Science and Engineering)

Under the guidance of
Shri Kaushik Muhuri, M.E.
Flat No. 501,
Sonajhuri Apartments,
22, Attapara Lane,
Calcutta – 700 050

Certificate from
Project Guide

Certificate from

to the

HOD

Acknowledgements

Any work that is the work of an individual certainly requires the efforts of many more people in keeping the original worker working. This project would not have reached its present form, had it not been for my parents' unending support, affection and inspiration. I must express my gratitude towards my employers and colleagues at Indian Cybernetics, who allowed me the use of the company's facilities and relaxed my work schedule such that I could allocate time for the project. I would next like to thank my guide Shri Kaushik Muhuri for his supervision and guidance. I am obliged to the Kolkata IETE centre for their consistent support. I must also thank Ms Sweta Karwa, an MBBS student, for helping me with the neuroanatomical and telemedicine aspects. Finally, and most importantly, I would like to thank the Almighty and my infant niece Saachi for providing me, in their own ways, with the much-needed spirit to complete this project. I apologise to everyone else who I should have, but could not name here.

I would like to dedicate this project to the victims of man-made catastrophes. I hope that in the future science shall yield only peace, health and prosperity.

Preface

Artificial Intelligence (AI) has established itself as one of the frontiers of scientific research. A pertinent application of AI has been in the field of autonomous robotics. An autonomous robot may be employed in various fields varying from domestic chores to industrial assembly line.

A major hindrance to a wider acceptance of legacy robotics has been its technically demanding interface, restricting its range of applications. A plausible solution is the incorporation of natural-interface in robotics design. This topic has become the focus of Human-Computer Interface (HCI) groups. The advent of Continuous Speech Recognition (CSR), along with speech synthesis, has made it possible to design robots that understand and make their responses in natural language. Incorporation of image recognition allows a robot to respond to visual stimuli.

A robot that simply carries out a programmed set of instructions is hardly considered intelligent. It needs to be able to detect and correct errors in its own plan and organize its actions independently. Such independent actions may, sometimes, have to be based on an incomplete set of instructions. This is the domain of machine learning (ML) and knowledge based systems (KBS). Lack of adaptability of a traditional robotic system can be solved using Genetic Algorithms (GAs). The AI research community has come to accept the role of Multi-Agent Systems (MAS) and sociological models in knowledge acquisition and ML.

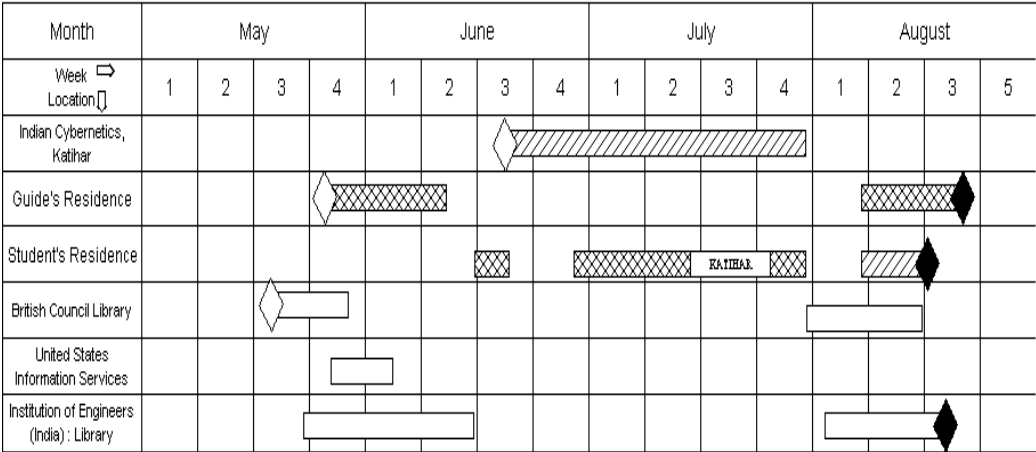
This project develops a Natural-Interfaced Intelligent Adaptive Agents (NI2A2) based system and designs applications for it in Domestic and Industrial Autonomous Robotics (DIARO). The problem domain of DIARO allows a proper environment to demonstrate the versatility of NI2A2 based systems, with obvious commercial applications.

Project planning, organisation and execution

The project work entitled "Application of Natural Interfaced (Intelligent Adaptive) Agents to Domestic and Industrial Robotics" has been undertaken in partial fulfilment of the requirements for the AMIETE examination. The project work has been completed over a period of three and a half months, and has been guided by Shri Kaushik Muhuri. The primary location for the project work has been the organisation where the student is employed. It has also been carried out in parts at the guide and student's residences. The literature survey has been done using the resources at British Council Library, USIS Library and IE (I) Library; all three in Calcutta. Books and technical papers from the student's own collection and the IETE Kolkata centre library have also been used. A timeline graph of the project is given on below.

Initially the NI2A2 theory was developed after a literature survey. Following that, the hardware design was undertaken. Finally, the software was designed using a spiral software development life-cycle model. The results from this project include, but are not limited to, the following:

1. Literature review of Artificial Intelligence and Autonomous Robotics.
2. Development of a theory for designing autonomous robots which can be controlled using a natural interface, can be trained in unrelated domains, can have their behaviour altered dynamically and have vision capabilities.
3. Development of a protocol for robot-computer interaction.
4. Development of a complete robotic platform, with a multiprocessor control system and an ultrasonic obstacle detector.
5. Development of an interface engine for the robot.
6. Control of a robot with voice commands.
7. Colour following robot.
8. Watchman robot.
9. Traffic light obeying robot
10. Housekeeper robot (simplified domain)
11. Industrial assistant robot (simplified domain)
12. Sign language understanding robot (simplified domain)
13. A proposal for a model vehicle collision prevention system
14. A proposal for a clinical decision support system reaching healthcare into remote villages.



Project timeline chart

Contents

<i>Preface</i>	v
<i>Project planning, organisation and execution</i>	vi
Chapter – 1:	
1.1 Introduction	1
1.2 Historical developments	2
1.3 Trends in AI and robotics	7
Chapter – 2:	
2.1 The Natural-Interfaced Intelligent Adaptive Agents	11
2.2 Natural-interface	12
2.3 Adaptive behaviour	13
2.4 Stimulus-Response agents	13
2.5 Intelligent behaviour	14
2.6 Top layer	15
2.7 Middle layer	15
2.8 Bottom layer	16
2.9 Point Events Driven Learner	17
2.10 Sensory areas	18
2.11 Association area	19
2.12 Coupling between different areas	19
2.13 An implementation of NI2A2	19
Chapter – 3:	
3.1 Overview of the system	45
3.2 Mechanical design of Angel-1	46
3.3 Control system design of Angel-1	46
3.4 Theory of operation of Angel-1 control system	47
3.5 Limitations of the system	49
Chapter – 4:	
4.1 Behavioural overview of Angel-1	83
4.2 A basic interface engine for Angel-1	83
4.3 Interfacing Angel-1's speech input to NI2A2	83
4.4 Interfacing Angel-1's vision to NI2A2	86
4.5 Components of the NI2A2:Angel-1 system	86
Chapter – 5:	
5.1 Overview of the experiments	99
5.2 A colour following robot	99
5.3 A robotic watchman	99
5.4 A traffic-light obeying robot	100
5.5 A sign language understanding robot	100
5.6 Housekeeping robot	101

5.7 Industrial assistant robot	101
5.8 A model Vehicle Collision Prevention System based on NI2A2	101
5.9 A Clinical Decision Support System based on NI2A2	102
Chapter – 6:	
6.1 Feasibility analysis	105
Conclusion	107
Appendix A: A short introduction to the PIC 16F84 microcontroller	109
References	117
CD-ROM	121

Chapter 1

1.1 Introduction

The dreams of the Greek philosophers to decipher mentation could, perhaps, have been fulfilled had they been bestowed with modern day computation power. The desire to resolve the mysteries that hold the key to the secrets about the cognitive abilities of man have never left the domain of active quest by humanity. Many strides have been made and many small steps taken. However, science, elucidated as man's adventure with his five senses, has been eluded by the inner space more than the outer. Man's conquest is now traversing the outer rims of the Solar system in the form of space probes, and he has seen far beyond - to a few fractions of a second before creation, lingering around the fringes of the universe. Still HAL9000, the truly "intelligent" machine from the Arthur C. Clarke classic "2001- A Space Odyssey", remains a piece of science fiction. The possibility remains that we are only coming full circle - the ancient Indian sages probably had all the answers we seek.

During his adventures in the world within man has designed many contraptions, trying to get them to mimic mental abilities or at least to extend his own. He has indeed succeeded in the later and the result is the modern day computer. The "modern day computer" refers not only to the familiar electronic devices but also cave-wall markings, the abacus, Napier's bones, Pascal's adder, Babbage's Analytical Engine, and the initial electromechanical devices that predated it. These have, however, been only tools; as in the words of Lady Ada Byron, Countess of Lovelace: "The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform." This pretext still holds true for our electronic computers. It was, hence, natural for man to shift his attention on the task of 'ordering' his contraptions to exhibit intelligent behaviour. This task was pioneered by the likes of Turing, Church, Chomsky, Minsky and Winograd among others. To 'en masse' their

achievement would be a diachronic effort, though in the history of civilisation a few decades stand out only as a speck of time.

Alan Turing's work on automatons showed that computing machines and the process of computation could be represented structurally. The Turing machine is capable of solving all "computable" problems. However, the fact that there exists a class of NP-hard problems which would require infinitesimal time for completion, alongside Kurt Godel's "Incompleteness Theorem", gave rise to the pretension that there is an unknown factor in the recipe of intelligence, one that man might never discover. A present day protagonist of this school of thought is Sir Roger Penrose. In his books 'Emperor's New Mind' and 'Shadows of The Mind' he holds out that it may be possible to synthesise intelligence but not with our present day science. Our standpoint through the course of this project shall be counter to Sir Penrose's, and in line with the beliefs of Noam Chomsky and Marvin Minsky. We shall first look into the historical factors and events that have lead upto the present day status in the related disciplines and then take a stock of the current technological developments. Following that we shall develop a theory to build a Natural Interfaced Intelligent Adaptive Agent (NI2A2) based robot. The next two steps would be to construct the robot hardware and model/program its behaviour. Finally, the adaptations for this system shall be designed and a feasibility analysis done.

1.2 Historical Developments

Since eternity man has wondered, "What is life?" He has indeed found many answers to this question, the simplest of which classifies living and non-living things and the most complex today deals with molecular biology and cytogenetics. He now knows the protoplasmic imprint that causes inheritance - the double helical DNA. He has even fiddled around with it resulting in "Dolly" - the first 'true' clone of a living organism. Intelligence and its mysteries, however, are still more than arms reach. Let us see how it all began.

The Indian scriptures deal elaborately on the topic of the cognitive abilities of man and the universe in general. "Isha vashyamidam..." is how the Ishavasya Upanishad begins. It literally translates to "All that exists, exists through the Paramataman". The individual Atman is like a segment, a part, of the Paramataman just like a part of space gets trapped in a vessel. It is both distinct, and indistinct, from the whole space at any given time. Hence all matter derives "Chetna", or consciousness, from the "Whole". Besides this philosophical dealing, the scriptures also provide elaborate theories of intelligence and consciousness. The current schools of thought in the fields of Artificial Intelligence (AI) however draw primarily from the Western schools of philosophy, which in turn are the direct descendants of Greek schools of thought. Hence, we begin our recollection of historical events starting with the Greek philosophers.

The 'Philosopher' had been a term primarily used for Aristotle for centuries. In his quest to fathom the depths of reality he promulgated many theories. He devised 'organon' (instrument) as a means of recording thought processes. Today's 'Logic' may be considered a direct descendant of 'organon'. He would say, "If I try to reflect upon reality using reality itself, I shall be intellectually blinded; hence I use language." His tool was language. He alongside Socrates started the analysis of language components, trying to understand its inner mechanisms. Those were the days before 'grammar'. (Although it may be argued that Panini's Astadhayayi, a Sanskrit grammar, predates Aristotle by centuries if not millennia, and has been described by Bloomfield as 'a monumental task of human intelligence'.) Aristotle and Socrates' work on analysing sentences into 'onamata' and 'rheamata', the parallel of 'subject' and 'predicate' of classical grammar, may be said to have seeded the development of the modern day grammar; and in turn the modern day AI, as we shall shortly see that Linguistics and AI have had a common lineage till very recently and still are somewhat interlinked.

In chronological order Descartes follows as the next, and the greatest, influencer of modern concepts of thought and mind. The efforts of St. Augustine at 'introspection' have also had a considerable impact, but we shall only focus on Descartes' promulgations. Descartes believed in the 'duality' of mind and body. He would doubt the existence of the material world and he believed that only thoughts were real - "Cogito ergo sum" - I think, so I am. In the Indian schools of thought, however, even the thoughts of the real world have been attributed to "Maya" and hence are not considered the reality. Moving on with Descartes we find the following consequences of his work on modern day AI:

- 1) By separating mind and body, the physical and mental processes could have their own laws, working in unison but independent of each other.
- 2) The mind and body having been separated, there was a need to find a link between the two, since their interaction is essential for human existence.

Following Descartes there was a slowdown in the development of the philosophy of mind and very few, if any, results were obtained during this period. However, this period saw the development of the first few automated machines. A wooden duck that mimicked the real duck's behaviour was designed by a French genius. Leonardo da Vinci gave the design for a mechanical man. However, these cannot be considered solid developments in the fields of AI and robotics. We shall now move on to the 18th and 19th centuries.

G.W. von Leibniz introduced the system of formal logic and Euler gave his analysis of the connectedness of the bridges of Konisberg, around the 18th century. Euler's work transcended into today's graph theory, the basis of our state-space graph, and the foundation on which the NI2A2 theory shall be based. From now on, the development of AI is parallel to the development of the "modern day computer".

Napier's had devised a set of bones for simplifying calculations, abacus has been in existence for millennia, Pascal designed a summing machine, however the truest computer was built only by the 'Father of Computers', Charles Babbage. His "difference engine" could solve many polynomial functions. Later, his "analytical engine" used stored program architecture. The latter could not be built in his lifetime owing to lack of technical expertise. IBM has in recent times developed and shown the validity of the design of the analytical engine.

In the years preceding "Sepoy Mutiny" in India, George Boole was developing the Boolean algebra. A discussion of the development of AI shall have a broken chain unless Boole's contributions are appreciated. Gottlob Frege developed what is now called the 'first-order predicate calculus' - a means of specifying propositions and giving their axiomatic meaning. Russell and Whitehead developed the formal system of specification of mathematical theorems. Alfred Tarski's "theory of reference" has a direct relationship to specifications for computing and programming languages. However, AI was to become a full-fledged scientific discipline only in the 20th century.

The hiatus between the world wars saw the race towards developing computing devices, primarily for secret messaging and deciphering intercepted enemy messages. Around this time Alan Turing, a British mathematician, devised what we now call the Universal Turing Machine (UTM). An UTM is only an abstract structure that can be programmed to solve "computable" problems. The 'Automata theory' also developed alongside. Another important work by Turing is the development of the Turing Test - a method of testing the "intelligence" of intelligent entities. It primarily is a test of testing resemblance through a question and answer session. The original test was designed such that the investigator had to decide which of his subjects was a female or a male. The current version decides which of the subjects is a human and which is a computer.

This was also the time that Bloomfield, an eminent linguist, came out with 'Immediate Constituent' (IC) analysis. IC analysis is a means for formally specifying grammars and analysing sentences. IC analysis formed the basis of the work of Noam Chomsky. Chomsky pioneered the field of "Syntactic Analysis". He developed what we now call the "generative grammars". Unlike classical grammars that can only analyse sentences, generative grammars can also 'generate' sentences. Chomsky desired to find a means of expressing the way language is represented in the mind. He developed the concept of "Deep Structure" and "Surface Structure". The utterances are the surface structure while the thought is the deep structure. A set of transformations converts between surface and deep structure. Beyond this, the era of modern AI started.

Marvin Minsky, an MIT scientist, devised a means of representing real world objects as "frames". His argument is that perception is based on the creation of "frames" from the perceived objects. The Object Oriented Programming (OOP) paradigm is based on this concept of frames and has now developed into what is known as the Universal Modelling Language (UML). He is also credited with conceptualising functional programming. Rosenblatt devised a "Threshold Logic Unit", called the perceptron. The perceptron along with McCulloch-Pitts neuron laid the foundations for the area of neural-networks. Widrow and Hoff devised the "adaptive control theory". There have been myriads of contributions around this period, however, another important contribution we consider is that of Winograd's SHRUDLU, a Natural Language Processing (NLP) system that conversed effectively about a world of "blocks". Before we move on to analyse the current state of AI and robotics, let us pay some attention to the latter.

The term "Robot" originates from the Czech word "Robota" meaning "work". This term was introduced into the English language by the playwright Karel Capek in his 1921 satirical drama R.U.R. (Rossum's Universal Robots). In this play the robots were humanlike machines that worked tirelessly and were intended to replace human workers, however in the end they turned against their

creators, annihilating the human race. The present day notion of robot's being endowed with individual personality draws from this work. Other important developments leading to initial developments of the notions of robotics were the design of the famous talking doll by Edison and the walking robot 'Electro' and his dog 'Sparko', displayed at the New York World's Fair in 1939. More recently, science fiction films like "Star Wars", "2001: A Space Odyssey" and the televised serial "Star Trek" have also added to 'robotic' folklore.

1.3 Trends in AI and Robotics related to the project

AI and robotics are very big and elaborate fields that combine many different disciplines. A few paragraphs cannot do justice to the length, breadth and depth of this field. We shall only consider those areas of AI and robotics that are of direct significance to this project.

We have so far skipped a formal definition of the term Artificial Intelligence. What is AI? The answer to this question is not simple. The term AI was coined at the 1956 Dartmouth College conference. It may be broadly defined as the discipline concerned with the design and development of intelligent artefacts, as well as the analysis of the intelligence of such artefacts. The areas of AI that are of significance to this project are formal systems, heuristic search, artificial neural networks (ANN), NLP, machine learning, knowledge acquisition, pattern recognition, visual image processing and expert systems. All these cover the aspects of robot-intelligence; the remaining areas of interest from robotics are robot-sensors, robot programming-languages, planning, scripting, navigation, and design. We now give a very brief description of the above topics.

Logic is a formal method of reasoning. Most concepts, which can be verbalised, can be translated into a symbolic representation. The formal methods of specifying and manipulating such symbolic representation are called "formal systems". First Order Predicate Logic (FOPL) is a formal system. FOPL allows

quantified variables to refer to the domain of discourse. It may be used for knowledge representation.

Heuristic is defined as "The study of the methods and rules of discovery and invention." It is based on the Greek root 'eurisco' meaning 'I discover'. Thus, when Archimedes famously shouted 'eureka' he meant 'I have found it.' So do the programs based on heuristic search. Only the constraints and rules are specified and the program 'discovers' the correct solution.

Hopfield proposed Artificial Neural Networks as a theory of memory. These are asynchronous, distributed, content addressable and fault tolerant connectionist structures. They are derived from models of TLU's such as the perceptron and the McCulloch-Pitts neuron. The TLU's are organised in a given topology with 'synaptic' connections. In effect ANN's are an effort towards modelling biological neural networks. Models based on ANN's are called connectionist models.

NLP is the term for processing humanlike languages by computers. These derive heavily on linguistic theories. The basic branches of linguistics - phonetics, phonology, morphology, syntax and semantics, form the basis of NLP. Syntactic analysis is based on generative grammars. An instance would be a sentence being broken up into Noun Phrase (NP) and Verb Phrase (VP). The NP and VP would further be subdivided in articles, nouns, adjectives etc.

Machine learning (ML) refers to a spectrum of activities from skill improvement to knowledge acquisition by computers. There are many prevalent theories and models of ML. Rote learning refers to learning through repetition. Learning from advice requires a trainer. Learning can also be achieved through trial and error. ML and knowledge acquisition are related topics as knowledge is acquired through learning.

Computers have been shown to recognise patterns very well. This is one successful area of AI. Some of the proponents of this field like Ray Kurzweil have devised applications decades ago, which could read out typed letters. Visual image processing, though not strictly an AI topic, is nevertheless indispensable. The tasks such as pattern recognition can only be accomplished when the computer has acquired a proper visual image. (Throughout this report we shall use the terms robot and computer interchangeably, as the autonomous robot is essentially a computer first.)

The final topic in AI we look at is the "Expert System". These systems try to model the human expertise in given domains. These are knowledge-based systems, with inference engines and very large database (VLDB) abilities. They can offer advice in their own problem domains. MYCIN was an expert system, which could identify the pathogens with a high accuracy. Deep Blue was another expert system, which beat the world chess champion at the game.

Once all the 'intelligence' is in place the robot needs to be physically designed. Let us briefly look at the robot design aspects. Primarily a robotic platform is required. It should provide the necessary degrees of freedom. This robotic platform should have sensors to provide feedback regarding the robot's environment. A robotic manipulator is a mechanism with which the robot can interact with its environment. This generally takes the form of a gripper or an arm. Robot programming has to be done with the native language of the robot's microprocessors. However some high level languages are available which greatly simplify the task. Planning refers to the organisation of the atomic units of task the robot has to perform. Scripting allows complex behaviour to be modelled out of discrete behaviour and events. Navigation literally means finding the proper way. This would include localisation, mapping and obstacle avoidance. Finally, before moving onto the description of the NI2A2 theory let us take a bird's eye view of the process of autonomous robotic design.

Autonomous robotic design can be easily categorised into design of the robot hardware and software – philosophically speaking, somatic and psychic design. The somatic design is essentially a task of mechatronics design. Hence, the mechanical and electronics components have to be developed in parallel. The control system design becomes much easier if the mechanical design is adaptable. Use of discrete control elements allows a simple control system analysis. Following this comes the design of the psychic or behavioural elements of the robot. This is essentially a software task. The software development model most suited for this purpose is the "Spiral Model", as the software is gradually improved. We shall now take a look at the NI2A2 theory.

Chapter 2

2.1 The Natural Interfaced Intelligent Adaptive Agents

We shall begin our discussion of the Natural Interfaced Intelligent Adaptive Agents (NI2A2) theory by first looking at the meaning of the individual constituents. Only when it is understood that what the constituents actually mean, shall we be able to define NI2A2 as a whole. During the course of this discussion our model shall be the human cognitive system, though at times we may incorporate features not available in humans, such as the ultrasonic guidance in bats. These shall only have a minor impact on the theory while increasing empirical appeal.

We begin with the meaning of the term natural-interface. It literally means the interface to nature. What is our natural interface? The input comes from the five external senses, and the output consists of a complex of speech, expression, locomotion, etc. If we are to take Descartes mind-body duality into account we shall also have to provide for the input sensations of proprioception and kinesis, that is sense within the muscles and the sense of motion. The only possible state when a 'living' human being may not have a natural interface of any kind is the comatose state. Another important fact of a natural-interface is its generality. All humans produce sounds within the audible spectrum while speaking. The audible spectrum for dolphins becomes ultrasonic for humans. Hence, we certainly do not have much of an interface with the dolphins. On the other hand dogs produce sounds within the audible range, and we can hear much of what they 'say', though we may not be able to interpret it. Besides speech, our senses include vision. Again there is a constraint on the visible spectra. Similarly for all our senses there are limitations. If we had the 'natural-interface' to RS-232, or perhaps RS-242 serial link, human computer interface groups would be left with very little to do. Indeed when electronic devices communicate using a given media and a protocol they are using a natural interface, though in an unintelligent

manner. Our primary requirement therefore is to provide our devices with a natural interface. Now let us consider the question of intelligence.

Intelligence is a very subjective topic. It may be recursively defined as "what intelligent artefacts do." However for our purpose it would suffice to consider intelligence as the sum of speech, learning, and knowledge acquisition. Speech is different from utterance, in that it has a definite structure and is intelligible. Learning and knowledge acquisition are the processes of remembering and recollecting, at a later time, what ever has been encountered with the natural interface. To adapt is to change as per new circumstances and changes in one's immediate environment.

Finally, lets us elucidate what we mean by an agent. In our context, an agent is a "reactive individual". There are two considerations here - reaction, and individuality. Reaction certainly is in response to some stimuli and individuality is a property of distinction of the agent. A more common term for such agents is 'stimulus-response' agents, or S-R agents. Let us now move on to the NI2A2 theory.

The NI2A2 theory is supposed to be a specification for the behavioural design and analysis of an "agent" accepting human "natural-interface" as stimulus and learning and acquiring knowledge from it, along with the ability to adapt. We shall base our philosophical thinking on the school of duality (mind and body as separate), as promulgated by Descartes. We shall follow the method of "Components Based Engineering" or CBE for our design purposes. Following is the design consideration of the constituents for NI2A2.

2.2 Natural Interface

Speech recognition and speech synthesis has been the focus area of active research for quite some time now. We already have continuous speech

recognition (CSR) systems where one can speak in a natural flow, unlike in discrete speech recognition where a pause is needed after every word. The problem with CSR systems as compared to discrete speech recognition is that the user base is limited. The CSR has to be individually trained for every user who uses it. However the benefits far exceed the cost. As they recognise complete sentences, it is easier to apply grammar rules. We shall therefore provide only for a natural-interface to text in the NI2A2 and make the system open ended such that any speech recognition engine may provide it with input.

Besides speech, the other natural interfaces do not have a formal means of definition. It would be nice to have a formal method for describing vision, but we know of none as yet. Hence, we shall need to take this fact into account in the NI2A2. Our proposition for natural-interface shall be clubbed with the proposition for intelligent behaviour. We shall therefore make intelligence the final consideration.

2.3 Adaptive Behaviour

All living beings represent adaptive behaviour. As we want our artefacts to exhibit adaptive behaviour we shall have to account for the basis of adaptation in organisms. Adaptation can be seen at two levels. An individual of a given generation adapts to his environment, for instance humans living in colder regions adapt to stay warm. Secondly, evolution is also a form of adaptation for survival. We shall incorporate both these types of adaptations into the NI2A2 theory.

Evolutionary adaptation may be achieved through the use of Genetic Algorithms (GA's). Individual adaptation shall have to be an intrinsic property of NI2A2 systems.

2.4 S-R Agents

S-R agents take sensory input from their surroundings, process the inputs and provide output. Generally the best examples of S-R agents are autonomous robots. However a lot of work is being done in the field of cellular automaton, which forms an important part of the discipline of machine intelligence. In India, we have some prominent researchers, like Prof. S. K. Pal from ISI, Kolkata. The 'Game of Life' developed by Conway, a biologist, was one of the first examples of cellular automata. We have developed as an example of cellular automaton a program for simulating multiple species based "Game of Life". It can be configured to include factors such as food, poison etc. into the cellular universe. The listing is given on page 23.

2.5 Intelligent Behaviour

Machine intelligence is one of the frontiers of scientific research. Initial developments included heuristic search, and other deterministic methods. However the limitations of such methods has long been demonstrated. Hence, we shall try and make the NI2A2 system intelligent by basing it on an intelligent 'role model'. Hence, we are not left with many choices, and we decide to use the human cognitive system for our role model. The modelling of human cognitive processes is another area of active research. There are many theories regarding how to model human cognition, and generally the more successful of these are the layered models. We shall follow a top-down approach for the analysis of the human cognitive system, till we finally have reached a level of implementable detail. This has an advantage over the bottom-up approach as there is no need for backtracking and the only hindrance comes from our limited understanding of the high-order functions of various parts of the *CNS* and the *cerebral cortex*. The available knowledge still is sufficient for the simplified version of our model.

The three layers are as follows:

I. Top Layer representing the higher order functions of cognition

II. Middle Layer which resolves the *Top Layer* into functional constituents

III. *Bottom Layer* which implements the components of *Middle Layer* using ANN's

2.6 Top Layer

The *Top Layer* is a black box representation of the cognitive processes for temporal knowledge acquisition and representation. It corresponds to the primary functional pathways of *stimulus-response (SR)* of the brain. Descartes' dualism proposed some non-material entity that was provided with the data from the sensory nerves, analysed it within itself, then responded with appropriate actions by acting on the motor nerves, but modern neurophysiology admits no path between the *SR* other than unbroken chains of neural connections. We can, therefore, safely assume that knowledge acquisition takes place as a precipitation from the *SR* pathways, which gets accumulated and concentrated in the brain. This acquired knowledge is represented in the *cerebral cortex* in the form of *synaptic weights* among *neuronal networks* in specialised areas. It is also recalled from these precipitation pathways. The specialised areas are resolved in the *Middle Layer* of the model. A schematic diagram for the top layer is given in Figure 1.

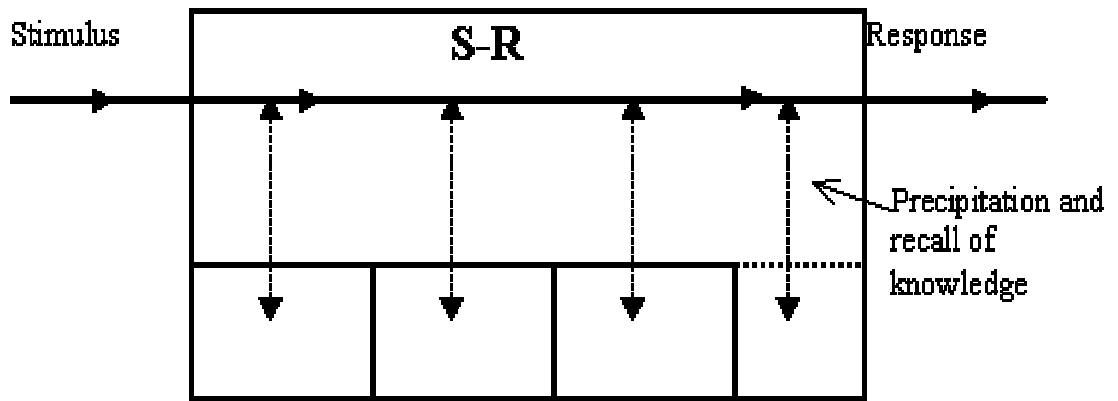


Figure 1: Top layer

2.7 Middle Layer

The *Middle Layer* resolves the *Top Layer* into constituent specialised areas. For our task of temporal knowledge acquisition we shall only consider the

sensory areas for *skin-sense*, *olfaction*, *vision*, *auditory-sense* and *gustatory-sense*. We do not neglect proprioception, as it is related to the domain of our applications, but as the required details would be specific for each implementation we do not consider it here. Hence the five areas representing the *somatosensory cortex (skin-sense)*, *primary auditory cortex (auditory-sense)*, *striate area (vision)*, *parietal operculum (taste)*, and *olfactory bulb (olfaction)* form the components of the *Middle Layer*, along with one generalised area representing the *association areas of frontal lobe, parietal lobe and temporal lobe*. These are implemented in the Bottom Layer of the model. A schematic diagram for the *Middle Layer* is given in Figure 2. The choice of the type, name and number of sensory areas is no critical. A method for representing the input, as a specific index, is given later.

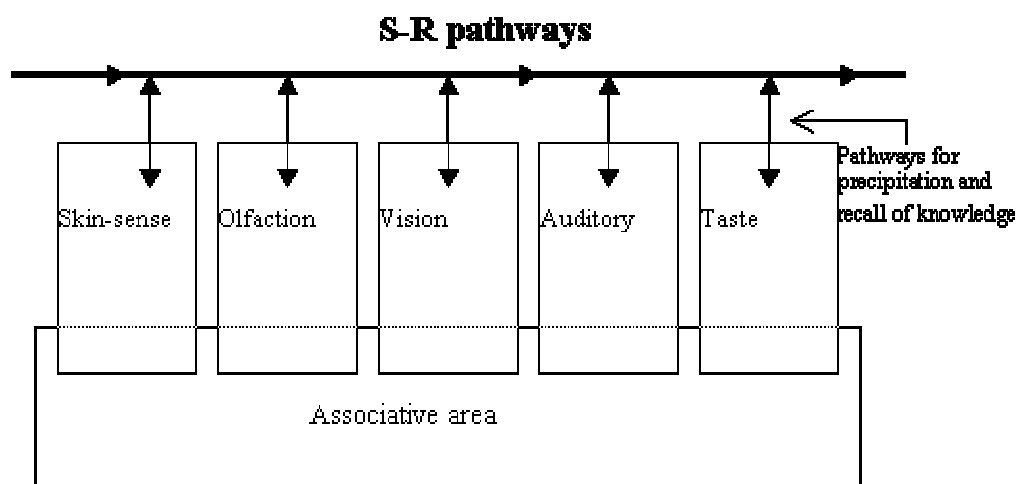


Figure 2 Middle Layer

2.8 Bottom Layer

The *Bottom Layer* resolves the components of the *Middle Layer* into easily implemented topologies of *ANN's*. The implementations of the *sensory areas*, *associative areas* are given below. However we before we can move on to these, we require an abstract 'learning' structure. We can use almost any learning structure such as Hopfield networks etc. here; however there may be some limitations. We choose to use a learning structure called "Point Events Driven

Learner", for the only reason that it does have a predefined number of states and can grow with increased knowledge.

2.9 Point Events Driven Learner

The *Point Events Driven Learner* is a learning automaton defined by the 6-tuple $(I, G, U, S, F, *)$, as below.

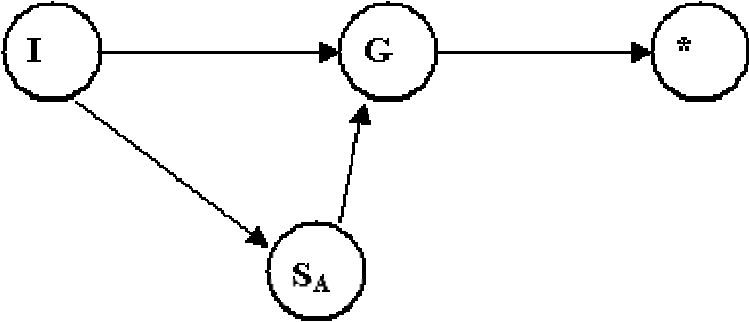
- I = Initial state
- G = *Governing function*, which determines state transitions and spawns new states.
- U = Set of all provided inputs
- S = Set of all attained states
- F = Set of functions mapping from domain U to S
- $*$ = Empirical class of all possible state-functions which the *Governing function* uses to spawn new states.

At the initial state I , *Pedler* has a non-deterministic behaviour and its state transitions are governed by the function G , which can accept all inputs possible in the given universe. All states call function G upon any interrupt. On receiving an input A , G checks to see whether an element F_A of the set F exists, if it does then it calls the function F_A , otherwise it spawns an instance S_A of the class $*$ and adds element S_A to set S , and defines the transition function F_A on domain U mapping into S , and adds element F_A to set F . Henceforth the automaton behaves in a deterministic way for the given input A . Therefore at any state, past the initial, the triple (S, F, U) , where S , F and U have the same meaning as above, defines a finite state automaton, and hence *Pedler* converges to a deterministic chaotic behaviour. In effect it learns to deal with the input A , while other inputs are handled by the *Governing function*. Therefore *Pedler* can be used to model evolutionary learning systems with a level of heuristics built in. The digraph for an initial state *Pedler* is given in Figure 3 and that after an input A at the initial state is given in Figure 4. The transition table is given in Table 1 below. For our model *Pedler* is used to dynamically generate the required topology using the

provided model of neuron (any stochastic model) as the state function. It may be stated here that the output of this model I given by the function F_A .



Pedler (initial state)
Figure 3



Pedler with state S_A spawned
Figure 4

G	A	~A
I	$F(A)$	$G(\sim A)$
S_A	$G(A)$	$G(\sim A)$

Table 1

2.10 Sensory areas

The five *Sensory Areas* from the *Middle Layer* are implemented using *Pedler* based topology of *ANN's* with a deterministic behaviour hard-wired into the *Governing function*, where the required states are already present and knowledge acquisition is entrusted to existing synapses. These areas are connected to *afferent* and *efferent* data lines in the system in which this cognitive model is to be implemented. A coding method for the sensory inputs is given below.

- i. Define a formal specification for the input class
- ii. Express the formal specification's language
- iii. Express an input formally as a string in the specification language
- iv. Find the Godel number for the input string.
- v. The Godel number gives the encoding for the sensory input

2.11 Association area

The *Association Area* accumulates and concentrates the temporal knowledge acquired by the *Sensory Areas*. It is also implemented using a *Pedler* based topology of *ANN's* but with a non-deterministic *Governing function*, which spawns new states as required by the input received from the *Sensory Areas*. The knowledge acquisition is of a dual nature here based on the connections generated and on individual synaptic weights.

2.12 Coupling between different areas

In any neural topology, including the biological ones, coupling between different areas is of primary importance. This is simplified by the incorporation of *Lateral Inhibition*, in the coupling stages. In the *Pedler* based topology of *ANN's* this is achieved by defining a single coupling structure, which is independent of the number of states spawned. This ensures that there is no blurring of inputs at any stage due to convergence or divergence in the number of spawned states, i.e. differences in the cardinality of the set S.

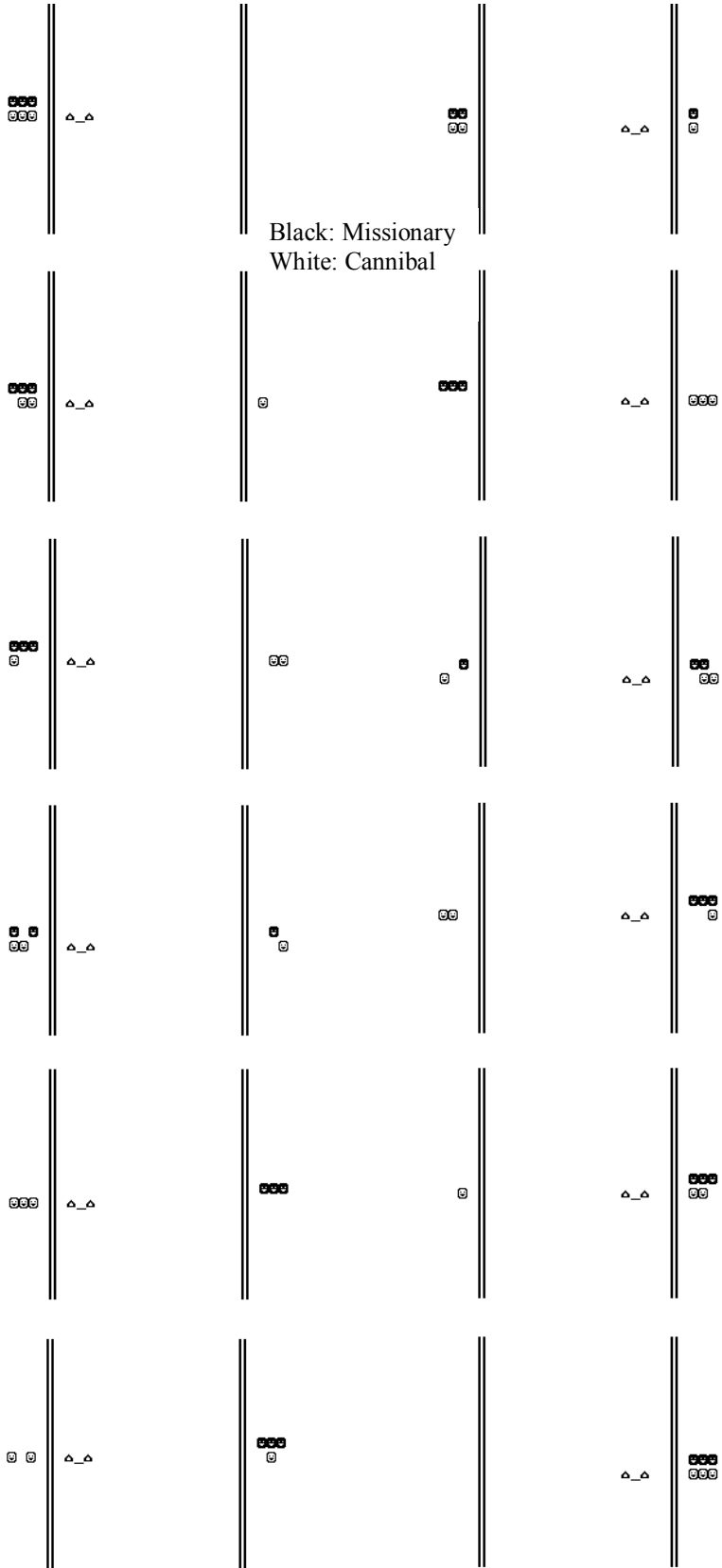
2.13 An implementation of NI2A2

Now that we have defined the NI2A2 theory let us test its applicability by developing an application based on it. However, first we shall develop a heuristic search program to be used as a control. We choose the missionaries and cannibals problem for this purpose. The program output is given on page 21, and the source code is listed on page 25. The task is to take a group of three

missionaries and three cannibals across a river. The constraints are that there is only one two seated boat, which needs at least one passenger to row it, second, if at any given point in time the number of cannibals exceeds the missionaries the result would be gory. Now let us move on to our NI2A2 implementation.

We use a tic-tac-toe (noughts and crosses) game for demonstrating the implementation of NI2A2. The choice has been affected by the factor that with an universe of only 19523 states the tic-tac-toe is a good candidate for heuristic search. Such implementations of tic-tac-toe can be found in many books on programming; one such being "OOP in Turbo C++" by Robert Lafore. We have used a Las Vegas simulation for the self-learning mode of the game. An elaborate discussion on this implementation of NI2A2 would not serve our purpose of applying NI2A2 to Domestic and Industrial Robotics. Hence, we give a brief description of it here. The listing for the program is given on page 28.

We shall call the program XOX for brevity. The sensory area of XOX is the 3x3 grid. The association areas are virtually implemented. The Pedler automaton generates a knowledgebase file as its number of states increases. The governing function has some amount of heuristics built into it. The input language is defined by the set {X, O, blank}, and every setup of the grid is specified as a string in this language. We have also designed programs to calculate the entropy for the knowledgebase as a ratio of complexions against the universe and to implement a GA for merging knowledgebases. The listings for these programs follow the main listing. This implementation defines the complete universe in the knowledgebase file, however for larger universes sparse matrix methods are necessary. For instance, the chess universe has around 2.25×10^{72} states, much of which is unreachable. Now that we have a theory for NI2A2 let us move on to the task to designing the hardware for the mobile robot.



Missionaries & Cannibals

Abhishek Choudhary

SG-129546

AMIETE (Computer Science and Engineering)

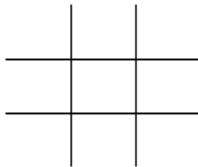
C-18: Project

Tic-Tac-Toe - An intelligent naughts and crosses game that learns how to play.
This program has been developed based on the PEDLER architecture.

Tic-Tac-Toe



Press: O



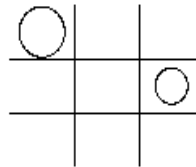
Press: X

Naught starts first.

Tic-Tac-Toe



Computer



Human

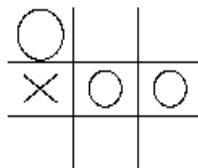
Use arrow keys. Press enter to select.

Using `scanf()`

Tic-Tac-Toe



Computer



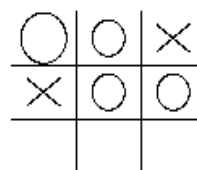
Human

Use arrow keys. Press enter to select.

Tic-Tac-Toe



Computer



Human

Use arrow keys. Press enter to select.

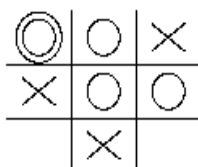
Using `scanf()`

Using `scanf()`

Tic-Tac-Toe



Computer



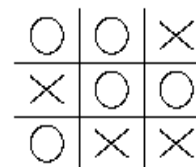
Human

Use arrow keys. Press enter to select.

Tic-Tac-Toe



Computer



Human

The match ends in a tie.

Using `scanf()`

Using `scanf()`

```
`Abhishek Choudhary
`SG-129546
`AMIETE (Computer Science and Engineering)
`Project C-18
`A program for multi-species game of life simulation
```

```
`$DYNAMIC
res = 70
RANDOMIZE TIMER
SCREEN 12
VIEW SCREEN (0, 0)-(639, 479)
WINDOW (0, 0)-(res + 1, res + 1)
DIM st(-1 TO res + 1, -1 TO res+1), dt(-1 TO res+1,-1 TO res+ 1)
DIM s(15)
```

```
`Only amale and afemale have been implemented
afemale = 1
amale = 2
food = 3
poison = 4
friendm = 5
friendf = 6
enemym = 7
enemyf = 8
pred1 = 9
ppred2 = 10
```

```
dt(26, 25) = 1
dt(26, 26) = 1
dt(26, 27) = 1
dt(27, 27) = 1
dt(28, 27) = 1
dt(28, 26) = 1
dt(28, 25) = 1
```

```
dt(46, 25) = 2
dt(46, 26) = 2
dt(46, 27) = 2
dt(47, 27) = 2
dt(48, 27) = 2
dt(48, 26) = 2
dt(48, 25) = 2
```

```
FOR x = 0 TO res
FOR y = 0 TO res
st(x, y) = dt(x, y)
LINE (x, y)-(x + 1, y + 1), st(x, y), BF
LINE (x, y)-(x + 1, y + 1), 0, B
NEXT y
NEXT x
```

```
FOR i = 1 TO 0'1600
x = CINT(res * RND)
y = CINT(res * RND)
st(x, y) = CINT(1 * RND) + 1
dt(x, y) = st(x, y)
LINE (x, y)-(x + 1, y + 1), dt(x, y), BF
LINE (x, y)-(x + 1, y + 1), 0, B
```

```

NEXT i
FOR i = 0 TO res
LINE (i, 0)-(i, res + 1), 0
LINE (0, i)-(res + 1, i), 0
NEXT i

DO
FOR x = 0 TO res
FOR y = 0 TO res
FOR i = 0 TO 15
s(i) = 0
NEXT i
s(st(x - 1, y - 1)) = s(st(x - 1, y - 1)) + 1
s(st(x, y - 1)) = s(st(x, y - 1)) + 1
s(st(x + 1, y - 1)) = s(st(x + 1, y - 1)) + 1
s(st(x - 1, y)) = s(st(x - 1, y)) + 1
s(st(x + 1, y)) = s(st(x + 1, y)) + 1

s(st(x - 1, y + 1)) = s(st(x - 1, y + 1)) + 1
s(st(x, y + 1)) = s(st(x, y + 1)) + 1
s(st(x + 1, y + 1)) = s(st(x + 1, y + 1)) + 1

SELECT CASE st(x, y)
CASE null
  IF s(amale) = 3 THEN dt(x, y) = amale
  IF s(afemale) = 3 THEN dt(x, y) = afemale
  IF s(afemale) + s(amale) = 3 THEN
    IF s(afemale) > s(amale) THEN dt(x, y) = afemale
    IF s(afemale) < s(amale) THEN dt(x, y) = amale
  END IF
  IF s(afemale) = s(amale) THEN dt(x, y) = null

CASE afemale
  IF s(afemale) < 2 OR s(afemale) + s(amale) > 3 THEN dt(x, y) = 0
CASE amale
  IF s(amale) < 2 OR s(afemale) + s(amale) > 3 THEN dt(x, y) = 0
CASE food
CASE poison
CASE friendm
CASE friendf
CASE enemym
CASE enemyf
CASE pred1
CASE pred2
END SELECT
'dt(x, y) = st(x, y)
NEXT y
NEXT x

FOR x = 0 TO res
FOR y = 0 TO res
st(x, y) = dt(x, y)
LINE (x, y)-(x + 1, y + 1), st(x, y), BF
LINE (x, y)-(x + 1, y + 1), 0, B
NEXT y
NEXT x

LOOP WHILE (INKEY$ = "")

```

```

//Abhishek Choudhary
//SG-129546
//AMIETE (Computer Science and Engineering)
//Missionaries and Cannibals : Heuristics based solution
//Compiler: Borland Turbo C++

#include <conio.h>
#include <iostream.h>

int crossed(char *m, char *c, char s);
void drawcurr(char *m, char *c, char b);
void ferry(char *m, char *c, char *b);
void ferry(char *m, char *b);

void main(){

    //declare variables
    int i,mn,cn;
    char m[3],c[3],b,s;

    //initialise arrays
    b='L';
    for(i=0;i<3;i++) {
        c[i]=m[i]=b;
    }

    //take them across
    s=b;
    drawcurr(m,c,b);while(!kbhit());getch();
    while(!crossed(m,c,s)) {
        mn=cn=0;
        for(i=0;i<3;i++) {
            if(m[i]==b) mn++;
            if(c[i]==b) cn++;
        }
        //if more miss on left then carry miss
        if((mn-2)>=cn && s==b) ferry(m,m,&b);
        else if(mn>cn && mn<3 && cn>1) ferry(c,c,&b);
        else if((mn==cn)) switch(mn) {
            case 1:
                ferry(m,&b);
                break;
            case 2:
                if(b==s)
                    else
                break;
            default:
                ferry(m,c,&b);
                break;
        }

        ferry(m,m,&b);

        ferry(m,c,&b);

        else if(cn>1 && s==b) ferry(c,c,&b);
        else ferry(c,&b);
        drawcurr(m,c,b);while(!kbhit());getch();
    }
}

```

```

} //end of main

void ferry(char *m, char *c, char *b){

    //declare local variables
    char t;
    int i;

    //ferry people
    if (*b=='L') {*b='R';t='L';}
    else {*b='L';t='R';}
    i=0;
    while(m[i]!=t) i++;
    m[i]=*b;
    i=0;
    while(c[i]!=t) i++;
    c[i]=*b;

} //end of ferry

void ferry(char *m, char *b){

    //declare local variables
    char t;
    int i;

    //ferry people
    if (*b=='L') {*b='R';t='L';}
    else {*b='L';t='R';}
    i=0;
    while(m[i]!=t) i++;
    m[i]=*b;
} //end of ferry

int crossed(char *m, char *c, char s){

    //declare local variables
    int i;

    //check if all have crossed
    for(i=0;i<3;i++){
        if (m[i]==s||c[i]==s) return(NULL);
    }
    return(!NULL);

} //end of crossed

void drawcurr(char *m, char *c, char b){
/*function for visual display*/
    //declare local variables
    int i;

    //clear screen
    clrscr();

    //draw river
    for(i=1;i<=25;i++){

```

```

        gotoxy(30,i);
        cout << (char)186;
        gotoxy(50,i);
        cout << (char)186;
    }

    //draw boat
    switch(b){
        case 'L':
            gotoxy(32,12);
            cout << (char)127 << "_" << (char)127;
            break;
        case 'R':
            gotoxy(45,12);
            cout << (char)127 << "_" << (char)127;
            break;
    }

    //draw miss & canni
    for(i=0;i<3;i++) {
        switch(m[i]){
            case 'L':
                gotoxy(26+i,11);
                cout << (char)2;
                break;
            case 'R':
                gotoxy(52+i,11);
                cout << (char)2;
                break;
        }
        switch(c[i]){
            case 'L':
                gotoxy(26+i,12);
                cout << (char)1;
                break;
            case 'R':
                gotoxy(52+i,12);
                cout << (char)1;
                break;
        }
    }
} //end of drawcurr

```

```

'Abhishek Choudhary
'SG-129546
'AMIETE (Computer Science and Engineering)
'C-18: Project
'Tic-Tac-Toe - An intelligent naughts and crosses game.
'This program has been developed based on the PEDLER
architecture.
'Compiler: Visual Basic for MS-DOS

'Declare subroutines and functions
DECLARE FUNCTION Analyse! (Checksum AS ANY)
DECLARE FUNCTION Antichecksum! ()
DECLARE FUNCTION CheckedMove! ()
DECLARE FUNCTION CheckMove! (Player AS ANY)
DECLARE FUNCTION Checksum! ()
DECLARE FUNCTION CheckWin! (Player AS ANY)
DECLARE FUNCTION ComputedInterpretation! ()
DECLARE FUNCTION HumanInterpretation! ()
DECLARE FUNCTION Results$ ()
DECLARE FUNCTION RookieMove! ()
DECLARE FUNCTION UserInput! ()
DECLARE FUNCTION Winstate! ()
DECLARE SUB ComputerMove ()
DECLARE SUB Configure ()
DECLARE SUB Cursor (X AS ANY, Y AS ANY)
DECLARE SUB Footnote (Message AS ANY)
DECLARE SUB GridUpdate ()
DECLARE SUB HumanMove ()
DECLARE SUB Initialize ()
DECLARE SUB MainSub ()
DECLARE SUB NewKnowledgeBase ()
DECLARE SUB PartingMessage ()
DECLARE SUB Uninitialize ()
DECLARE SUB UpdateKnowledgebaseFile (MoveNumber AS ANY)
DECLARE SUB UpdateKnowledgeBase ()
DECLARE SUB Welcome ()

'Define global constants
CONST Null = 0
CONST Tie = 6
CONST Over = 9
CONST Knaught = 1
CONST Cross = -1
CONST Left = 2
CONST Right = 3
CONST Up = 4
CONST Down = 5
CONST True = 1
CONST False = -1
CONST Linefeed = 13
CONST CursorRadius = .35
CONST PositionO = 7
CONST PositionX = 60
CONST Minimum = 0
CONST Maximum = 9761

'Declare global variables
DIM SHARED KnowledgeBaseFile AS STRING
DIM SHARED SelfTaught AS INTEGER, RandomStart AS INTEGER

```

```

DIM SHARED KnowledgeByte AS STRING * 1, Choice AS INTEGER
DIM SHARED ComputerChoice AS INTEGER, Cheat AS INTEGER
DIM SHARED MoveIndex AS INTEGER, FileChannel AS INTEGER
DIM SHARED Status, AnalysisA, AnalysisB

'Declare matrices for grid and MoveMades made
DIM SHARED Grid(2, 2), MoveMade(9, 1)

'Define memory for sprites
DIM SHARED SpriteO(1000), SpriteX(1000)

'Show welcome screen
CALL Welcome

'Initialize environment
CALL Initialize

'Play the game
CALL MainSub

'Give results and analyse game
CALL Footnote(Results)
SLEEP

'Update knowledge base
CALL UpdateKnowledgeBase

'Ask for another round
CALL Footnote("The more I play, the more I learn. Another round?
(Y/N)")
WHILE Response$ = ""
    Response$ = INKEY$
    SELECT CASE UCASE$(Response$)
        CASE "Y"
            RUN
        CASE "N"
            Response$ = "N"
        CASE ELSE
            Response$ = ""
    END SELECT
WEND

'Display parting screen
CALL PartingMessage

'Uninitialize
CALL Uninitialize

'End execution
END

FUNCTION Analyse (State)

    'Get the analysis from the knowledge base
    NeuronLocation = State + Maximum + 1
    GET FileChannel, NeuronLocation, KnowledgeByte

    'Exatract knowledge
    KnowledgeFull = ASC(KnowledgeByte)

```

```

    Status = KnowledgeFull \ 100
    AnalysisA = (KnowledgeFull MOD 100) \ 10
    AnalysisB = KnowledgeFull MOD 10

    'Provide analysis
    Analyse = AnalysisB

END FUNCTION

FUNCTION Antichecksum ()

    'Get unique antichecksum for current grid status
    Sum = 0
    FOR X = 0 TO 2
        FOR Y = 0 TO 2
            LinearPosition = X * 3 + Y
            LinearValue = 3 ^ LinearPosition
            LinearSigned = LinearValue * -Grid(X, Y)
            Sum = Sum + LinearSigned
        NEXT Y
    NEXT X
    Antichecksum = Sum

END FUNCTION

FUNCTION CheckedMove ()

    'Check if there is a positive move
    Positivity = CheckMove(ComputerChoice)
    IF Positivity <> False THEN
        CheckedMove = Positivity
        EXIT FUNCTION
    END IF

    'Check if there is a negative move
    Negativity = CheckMove(Choice)
    IF Negativity <> False THEN
        CheckedMove = Negativity
        EXIT FUNCTION
    END IF

    'When neutral
    CheckedMove = CINT(8 * RND)

END FUNCTION

FUNCTION CheckMove (Player AS INTEGER)

    'This code is a cheat to help the teacher!
    Cheat = Null

    'Check verticals
    FOR X = 0 TO 2
        IF Grid(X, 0) = Player AND Grid(X, 0) = Grid(X,
1) AND Grid(X, 2) = Null THEN CheckMove = X * 3 + 2: Cheat =
True: EXIT FUNCTION
        IF Grid(X, 0) = Player AND Grid(X, 0) = Grid(X,
2) AND Grid(X, 1) = Null THEN CheckMove = X * 3 + 1: Cheat =
True: EXIT FUNCTION
    
```

```

        IF Grid(X, 1) = Player AND Grid(X, 1) = Grid(X,
2) AND Grid(X, 0) = Null THEN CheckMove = X * 3 + 0: Cheat =
True: EXIT FUNCTION
    NEXT X

    'Check horizontals
    FOR Y = 0 TO 2
        IF Grid(0, Y) = Player AND Grid(0, Y) = Grid(1,
Y) AND Grid(2, Y) = Null THEN CheckMove = 2 * 3 + Y: Cheat =
True: EXIT FUNCTION
        IF Grid(0, Y) = Player AND Grid(0, Y) = Grid(2,
Y) AND Grid(1, Y) = Null THEN CheckMove = 1 * 3 + Y: Cheat =
True: EXIT FUNCTION
        IF Grid(1, Y) = Player AND Grid(1, Y) = Grid(2,
Y) AND Grid(0, Y) = Null THEN CheckMove = 0 * 3 + Y: Cheat =
True: EXIT FUNCTION
    NEXT Y

    'Check positive diagonal
    IF Grid(0, 0) = Player AND Grid(0, 0) = Grid(1, 1) AND
Grid(2, 2) = Null THEN CheckMove = 2 * 3 + 2: Cheat = True: EXIT
FUNCTION
    IF Grid(0, 0) = Player AND Grid(0, 0) = Grid(2, 2) AND
Grid(1, 1) = Null THEN CheckMove = 1 * 3 + 1: Cheat = True: EXIT
FUNCTION
    IF Grid(1, 1) = Player AND Grid(1, 1) = Grid(2, 2) AND
Grid(0, 0) = Null THEN CheckMove = 0 * 3 + 0: Cheat = True: EXIT
FUNCTION

    'Check negative diagonal
    IF Grid(2, 0) = Player AND Grid(2, 0) = Grid(1, 1) AND
Grid(0, 2) = Null THEN CheckMove = 0 * 3 + 2: Cheat = True: EXIT
FUNCTION
    IF Grid(2, 0) = Player AND Grid(2, 0) = Grid(0, 2) AND
Grid(1, 1) = Null THEN CheckMove = 1 * 3 + 1: Cheat = True: EXIT
FUNCTION
    IF Grid(1, 1) = Player AND Grid(1, 1) = Grid(0, 2) AND
Grid(2, 0) = Null THEN CheckMove = 2 * 3 + 0: Cheat = True: EXIT
FUNCTION

    'So checkmove fails
    CheckMove = False

END FUNCTION

FUNCTION Checksum ()

    'Get unique checksum for current grid status
    Sum = 0
    FOR X = 0 TO 2
        FOR Y = 0 TO 2
            LinearPosition = X * 3 + Y
            LinearValue = 3 ^ LinearPosition
            LinearSigned = LinearValue * Grid(X, Y)
            Sum = Sum + LinearSigned
        NEXT Y
    NEXT X
    Checksum = Sum

```

END FUNCTION

FUNCTION CheckWin (Player)

 'Initialise CheckWin to false
 CheckWin = Null

 'Check horizontal rows
 FOR X = 0 TO 2
 IF Grid(X, 0) = Grid(X, 1) AND Grid(X, 1) =
Grid(X, 2) AND Grid(X, 1) = Player THEN CheckWin = True
 NEXT X

 'Check vertical rows
 FOR Y = 0 TO 2
 IF Grid(0, Y) = Grid(1, Y) AND Grid(1, Y) =
Grid(2, Y) AND Grid(1, Y) = Player THEN CheckWin = True
 NEXT Y

 'Check diagonals
 IF Grid(0, 0) = Grid(1, 1) AND Grid(1, 1) = Grid(2, 2)
AND Grid(1, 1) = Player THEN CheckWin = True
 IF Grid(0, 2) = Grid(1, 1) AND Grid(1, 1) = Grid(2, 0)
AND Grid(1, 1) = Player THEN CheckWin = True

END FUNCTION

FUNCTION ComputedInterpretation ()

 'Ask the human to interpret the result
 CALL Footnote("Interpreting win state.")
 Response = Winstate
 SELECT CASE Response
 CASE Knaught
 ComputedInterpretation = 1
 CASE Cross
 ComputedInterpretation = 2
 CASE Tie
 ComputedInterpretation = 3
 CASE ELSE
 ComputedInterpretation = Null
 END SELECT

END FUNCTION

SUB ComputerMove ()

 'Ask the user to make a MoveMade
 CALL Footnote("Computer's move.")

 'Try knowledgebase
 Analysis = Analyse(Checksum)
 SELECT CASE Status
 CASE 0
 IF SelfTaught = True THEN
 MakeMove = RookieMove
 Y = MakeMove MOD 3
 X = MakeMove \ 3

```

        ELSE
            X = Null
            Y = Null
            CALL Cursor(X, Y)
        END IF
    CASE 1
        MakeMove = Analysis
        Y = MakeMove MOD 3
        X = MakeMove \ 3
        IF Grid(X, Y) <> Null THEN
            MakeMove = RookieMove
            Y = MakeMove MOD 3
            X = MakeMove \ 3
        END IF
    CASE 2
        EXIT SUB
END SELECT

'Do a first move
IF MoveIndex = 0 THEN
    IF RandomStart = True THEN
        MakeMove = CINT(8 * RND)
        Y = MakeMove MOD 3
        X = MakeMove \ 3
    ELSE
        MakeMove = Analysis
        Y = MakeMove MOD 3
        X = MakeMove \ 3
    END IF
END IF

'Update grid
Grid(X, Y) = ComputerChoice
MoveMade(MoveIndex, 1) = X * 3 + Y
MoveMade(MoveIndex + 1, 0) = Checksum
MoveIndex = MoveIndex + 1
CALL GridUpdate

END SUB

SUB Configure ()

    'Check if configuration file exists
    Channel = FREEFILE
    OPEN "X-O-X.INI" FOR RANDOM AS Channel
    FileExists = SGN(LOF(Channel))
    CLOSE Channel

    'Take appropriate action and configure environment
    SELECT CASE FileExists
        CASE 0
            KnowledgeBaseFile = "KBFile.KB"
            RandomStart = False
            SelfTaught = True
            OPEN "X-O-X.INI" FOR OUTPUT AS Channel
                PRINT #Channel,
"KnowledgeBaseFile=KBFile.KB"
                PRINT #Channel,
"RandomStart=True"

```

```

        PRINT #Channel, "SelfTaught=True"
        PRINT #Channel, "#Please do not
write any comments above this line."
        PRINT #Channel, "#Do not change
the order of variables above."
        PRINT #Channel,
"#KnowledgeBaseFile accepts any filename in DOS(FILENAME.EXT)
format."
        PRINT #Channel, "#RandomStart and
SelfTaught accept True or False"
        PRINT #Channel, "#If there is
problem starting Tic-Tac-Toe delete this file."
        PRINT #Channel, "#The program
automatically generates a new file."
        CLOSE Channel
    CASE 1
        DIM IniLine(3) AS STRING
        OPEN "X-O-X.INI" FOR INPUT AS Channel
        FOR Counter = 1 TO 3
            LINE INPUT #Channel,
IniLine(Counter)
            IniLine(Counter) =
LTRIM$(RIGHT$(IniLine(Counter), LEN(IniLine(Counter)) -
INSTR(IniLine(Counter), "=")))
            PRINT IniLine(Counter)
        NEXT Counter
        CLOSE Channel
        KnowledgeBaseFile = IniLine(1)
        IF UCASE$(IniLine(2)) = "FALSE" THEN
            RandomStart = False
        ELSE
            RandomStart = True
        END IF
        IF UCASE$(IniLine(3)) = "FALSE" THEN
            SelfTaught = False
        ELSE
            SelfTaught = True
        END IF
    END SELECT
END SUB

SUB Cursor (X, Y)
    'Perform cursor routine
    Movement = Null
    CIRCLE (X + .5, Y + .5), CursorRadius
    WHILE Movement <> Linefeed
        Movement = UserInput
        SELECT CASE Movement
            CASE Left
                CIRCLE (X + .5, Y + .5),
CursorRadius, 0
                X = (X - 1)
                IF X < 0 THEN X = X + 3
            CASE Right
                CIRCLE (X + .5, Y + .5),
CursorRadius, 0
                X = (X + 1)
        END SELECT
    END WHILE
END SUB

```

```

                IF X > 2 THEN X = X - 3
CASE Up
    CIRCLE (X + .5, Y + .5),
CursorRadius, 0
    Y = (Y - 1)
    IF Y < 0 THEN Y = Y + 3
CASE Down
    CIRCLE (X + .5, Y + .5),
CursorRadius, 0
    Y = (Y + 1)
    IF Y > 2 THEN Y = Y - 3
CASE Linefeed
    IF Grid(X, Y) = Null THEN
        CIRCLE (X + .5, Y + .5),
CursorRadius, 0
    ELSE
        Movement = Null
    END IF
CASE ELSE
    Movement = Null
END SELECT
IF Movement <> Linefeed THEN CIRCLE (X + .5, Y +
.5), CursorRadius
WEND

```

END SUB

SUB Footnote (Message AS STRING)

```

    'Print message as centered footnote
    Length = LEN(Message)
    LOCATE 23, 1: PRINT SPACE$(80)
    LOCATE 23, 36 - Length \ 2: PRINT Message

```

END SUB

SUB GridUpdate ()

```

    'Redraw grid
    FOR X = 0 TO 2
        FOR Y = 0 TO 2
            SELECT CASE Grid(X, Y)
                CASE Cross
                    PUT (X, Y), SpriteX, OR
                CASE Knaught
                    PUT (X, Y), Sprite0, OR
            END SELECT
        NEXT Y
    NEXT X

```

END SUB

SUB HumanMove ()

```

    'Ask the user to make a MoveMade
    CALL Footnote("Use arrow keys. Press enter to select.")

    'Provide the human an interface
    X = Null
    Y = Null

```

```

CALL Cursor(X, Y)

'Update grid
Grid(X, Y) = Choice
MoveMade(MoveIndex, 1) = X * 3 + Y
MoveMade(MoveIndex + 1, 0) = Checksum
MoveIndex = MoveIndex + 1
CALL GridUpdate

END SUB

SUB Initialize ()

    'Set environment
    CALL Configure

    'Set graphics mode and screen resolution
    SCREEN 12
    CLS
    VIEW (0, 0)-(639, 479)
    WINDOW SCREEN (-2, -2)-(6, 6)

    'Print title and foot note
    LOCATE 4, 4: PRINT "Tic-Tac-Toe"
    LOCATE 5, 4: PRINT "~~~~~"
    CALL Footnote("Naught starts first.")

    'Draw grid
    LINE (1, 0)-(1, 3)
    LINE (2, 0)-(2, 3)
    LINE (0, 1)-(3, 1)
    LINE (0, 2)-(3, 2)

    'Draw knaught and cross and label them
    CIRCLE (-1, 0), .25
    LINE (4, -.25)-(4.5, .25)
    LINE (4.5, -.25)-(4, .25)
    LOCATE 11, Position0: PRINT "Press: O"
    LOCATE 11, PositionX: PRINT "Press: X"

    'Get sprites
    GET (-1.5, -.5)-(-.5, .5), Sprite0
    GET (3.75, -.5)-(4.75, .5), SpriteX

    'Get user option
    Choice = Null
    WHILE Choice <> Knaught AND Choice <> Cross
        Choice = UserInput
    WEND
    ComputerChoice = (NOT Choice) + 1

    'Initialize grid matrix
    FOR X = 0 TO 2
        FOR Y = 0 TO 2
            Grid(X, Y) = Null
        NEXT Y
    NEXT X
    CALL GridUpdate

```

```

'Modify screen
HumanLabel = PositionO
ComputerLabel = PositionX
IF Choice = Cross THEN
    SWAP HumanLabel, ComputerLabel
END IF
LOCATE 11, HumanLabel: PRINT " Human "
LOCATE 11, ComputerLabel: PRINT "Computer"

'Startup knowledge-base
FileChannel = FREEFILE
LOCATE 28, 1: PRINT "Using "; KnowledgeBaseFile
OPEN KnowledgeBaseFile FOR RANDOM AS FileChannel LEN =
LEN(KnowledgeByte)
IF LOF(FileChannel) = 0 THEN CALL NewKnowledgeBase

'Setup random sequence generator
RANDOMIZE TIMER

END SUB

SUB MainSub ()

'Get grid checksum and analyse
Analysis = Analyse(Checksum)

'Play the game
MoveIndex = 0
IF Choice = Knaught THEN
    WHILE NOT (Status = 2 OR MoveIndex >= Over)
        IF Winstate THEN EXIT SUB
        IF MoveIndex MOD 2 THEN
            ComputerMove
        ELSE
            HumanMove
        END IF
        Analysis = Analyse(Checksum)
    WEND
ELSE
    WHILE NOT (Status = 2 OR MoveIndex = Over)
        IF Winstate THEN EXIT SUB
        IF MoveIndex MOD 2 THEN
            HumanMove
        ELSE
            ComputerMove
        END IF
        Analysis = Analyse(Checksum)
    WEND
END IF

END SUB

SUB NewKnowledgeBase ()

'Generate structure for new knowledge-base
CALL Footnote("Generating knowledge base. Please wait.")
KnowledgeByte = CHR$(Null)
FOR Location = 1 TO 2 * Maximum + 1

```

```

                PUT FileChannel, Location, KnowledgeByte
            NEXT Location

END SUB

SUB PartingMessage ()

    'Print the program information and hold screen
    CLS
    LOCATE 1, 1
    PRINT "Abhishek Choudhary"
    PRINT "SG-129546"
    PRINT "AMIETE (Computer Science and Engineering)"
    PRINT "C-18: Project"
    PRINT "Tic-Tac-Toe - An intelligent naughts and crosses
game that learns how to play."
    PRINT "This program has been developed based on the
PEDLER architecture."
    SLEEP

END SUB

FUNCTION Results$ ()

    'Decide results
    Decision = Analyse(Checksum)
    IF Status <> 2 THEN Decision = ComputedInterpretation

    'Return results
    MoveMade(9, 0) = Checksum
    SELECT CASE Decision
        CASE 1
            MoveMade(9, 1) = 1
            IF Choice = Knaught THEN
                Results$ = "Congratulations. You
win."
            ELSE
                Results$ = "I win. Try next
time."
            END IF
        CASE 2
            MoveMade(9, 1) = 2
            IF Choice = Knaught THEN
                Results$ = "I win. Try next
time."
            ELSE
                Results$ = "Congratulations. You
win."
            END IF
        CASE 3
            MoveMade(9, 1) = 3
            Results$ = "The match ends in a tie."
    END SELECT

END FUNCTION

FUNCTION RookieMove ()

    'Check if antichecksum exists

```

```

MakeMove = Analyse(Antichecksum)
IF Status AND MoveIndex THEN
    Y = MakeMove MOD 3
    X = MakeMove \ 3
    IF Grid(X, Y) = Null THEN
        RookieMove = MakeMove
        EXIT FUNCTION
    END IF
END IF

'When all fails generate a randomized MoveMade
CALL GridUpdate
MakeMove = False
WHILE MakeMove < 0
    MakeMove = CheckedMove
    Y = MakeMove MOD 3
    X = MakeMove \ 3
    IF Grid(X, Y) <> Null THEN
        MakeMove = -1
    END IF
WEND

'Set rookie MoveMade
RookieMove = MakeMove

END FUNCTION

SUB Uninitialize ()

    'Close all files
    CLOSE

    'Set video mode to text and clear screen
    SCREEN 0, 0, 0
    CLS

END SUB

SUB UpdateKnowledgeBase ()

    'Decide outcome
    Decision = MoveMade(9, 1)

    'Make appropriate updates
    SELECT CASE Decision
        CASE 1
            FOR UpdateMove = 0 TO MoveIndex - 1 STEP
2
                CALL
UpdateKnowledgebaseFile(UpdateMove)
                NEXT UpdateMove
            CASE 2
                FOR UpdateMove = 1 TO MoveIndex - 1 STEP
2
                CALL
UpdateKnowledgebaseFile(UpdateMove)
                NEXT UpdateMove
            CASE 3

```

```

FOR UpdateMove = 0 TO MoveIndex - 1 STEP
1
    CALL
UpdateKnowledgebaseFile(UpdateMove)
    NEXT UpdateMove
    END SELECT

    'Update result
    KnowledgeByte = CHR$(200 + MoveMade(9, 1))
    PUT FileChannel, MoveMade(9, 0) + Maximum + 1,
KnowledgeByte

END SUB

SUB UpdateKnowledgebaseFile (MoveNumber)

    'Analyse status
    Analysis = Analyse(MoveMade(MoveNumber, 0))

    'Format information
    SELECT CASE Status
        CASE 0
            Information = 100 + MoveIndex * 10 +
MoveMade(MoveNumber, 1)
        CASE 1
            IF AnalysisA > MoveIndex THEN
                Information = 100 + MoveIndex *
10 + MoveMade(MoveNumber, 1)
            ELSE
                EXIT SUB
            END IF
        CASE 2
            EXIT SUB
    END SELECT
    KnowledgeByte = CHR$(Information)

    'Calculate neuron location
    NeuronLocation = MoveMade(MoveNumber, 0) + Maximum + 1

    'Update neuron at MoveMade number
    PUT FileChannel, NeuronLocation, KnowledgeByte

END SUB

STATIC FUNCTION UserInput ()

    'Clear buffer
    WHILE INKEY$ <> "": WEND

    'Pool for user response
    DIM Response AS STRING
    Response = ""
    WHILE Response = ""
        Response = UCASE$(INKEY$)
    WEND

    'Generate and return response code
    SELECT CASE Response
        CASE CHR$(0) + "K"

```

```

        UserInput = Left
CASE CHR$(0) + "M"
        UserInput = Right
CASE CHR$(0) + "H"
        UserInput = Up
CASE CHR$(0) + "P"
        UserInput = Down
CASE CHR$(13)
        UserInput = Linefeed
CASE "Y"
        UserInput = True
CASE "N"
        UserInput = False
CASE "O"
        UserInput = Knaught
CASE "X"
        UserInput = Cross
CASE "T"
        UserInput = Tie
CASE ELSE
        UserInput = Null
END SELECT

END FUNCTION

SUB Welcome ()

    'Same as parting message
    SCREEN 12
    CALL PartingMessage

END SUB

FUNCTION Winstate ()

    'Check if any knaught or cross won or its a tie
    Winstate = Null
    IF MoveIndex = 9 THEN Winstate = Tie
    IF CheckWin(Knaught) THEN Winstate = Knaught
    IF CheckWin(Cross) THEN Winstate = Cross

END FUNCTION

```

```

'Abhishek Choudhary
'SG-129546
'AMIETE (Computer Science and Engineering)
'C-18: Project
'Entropy - Finds the entropy of XOX knowledgebases against the
XOX universe.
'This program has been developed based on the PEDLER
architecture.
'Compiler: Visual Basic for MS-DOS

'Get identity
F$ = COMMAND$
IF F$ = "" THEN INPUT "Enter identity(filename): ", F$
IF F$ = "" THEN
    BEEP
    PRINT "Entropy for ambiguous identity(non-existent
filename) cannot be determined."
    END
END IF

'Calculate entropy
DIM Cell AS STRING * 1
Maximum = 19523                                'Universe Value
OPEN F$ FOR RANDOM AS #1 LEN = LEN(Cell)
FOR Counter = 1 TO LOF(1)
    GET #1, Counter, Cell
    IF Cell <> CHR$(0) THEN Complexions = Complexions + 1
NEXT Counter
Entropy = (Complexions / Maximum) * 100        'Entropy as a
measure of
CLOSE                                          'complexions

'Display entropy
PRINT "Entropy of "; F$
PRINT STRING$(40, "°"); " 100 "; "Maximum"
PRINT STRING$(CINT(.4 * Entropy), "°"); CINT(Entropy);
SELECT CASE Entropy
    CASE IS < 1
        PRINT "Nil"
    CASE 1 TO 33
        PRINT "Low"
    CASE 34 TO 67
        PRINT "Medium"
    CASE 67 TO 99
        PRINT "High"
    CASE IS > 99
        PRINT "Maximum"
END SELECT

END

```

```

'Abhishek Choudhary
'SG-129546
'AMIETE (Computer Science and Engineering)
'C-18: Project
'Genesis - A simple GA based merger of XOX knowledgebase
'This program has been developed based on the PEDLER
architecture.
'Compiler: Visual Basic for MS-DOS

'Get the parental knowledgebase identity
INPUT "Enter identity(filename) of first initiator: ", F1$
INPUT "Enter identity(filename) of second initiator: ", F2$

'Christen the neo-knowledgebase
INPUT "Enter identity(filename) of progeny: ", O$

'Check ambiguity
IF F1$ = "" OR F2$ = "" OR O$ = "" THEN
    BEEP
    PRINT "Genesis cannot work with ambiguous genotypes(non-
existent filenames)."
    END
END IF

'Create media
DIM KnowA AS STRING * 1
DIM KnowB AS STRING * 1
DIM KnowO AS STRING * 1

'Start synthesis
OPEN F1$ FOR RANDOM AS #1 LEN = LEN(KnowA)
OPEN F2$ FOR RANDOM AS #2 LEN = LEN(KnowB)
OPEN O$ FOR RANDOM AS #3 LEN = LEN(KnowO)

'Check for paradox
PRINT "Checking paradox..."
IF LOF(1) = 0 OR LOF(2) = 0 THEN
    BEEP
    PRINT "Both initiators are required."
    END
END IF
IF LOF(3) > 0 THEN
    BEEP
    PRINT "The progeny requires a new identity(filename)."
    END
END IF

'Check compatibility
IF LOF(1) <> LOF(2) THEN
    BEEP
    PRINT "Parents are incompatible."
    END
END IF

'Apply Darwin's theorem: The fittest survives.
PRINT "Applying Darwin's theorem. May the fittest survive..."
PRINT "(PseudoGenetic Algorithm)"
FOR i = 1 TO LOF(1)
GET #1, i, KnowA

```

```

GET #2, i, KnowB
SELECT CASE ASC(KnowA)
  CASE IS = 0
    KnowO = KnowB
  CASE 100 TO 199
    SELECT CASE ASC(KnowB)
      CASE 0
        KnowO = KnowA
      CASE 100 TO 199
        IF KnowA < KnowB THEN
          KnowO = KnowA
        ELSE
          KnowO = KnowB
        END IF
      CASE IS >= 200
        KnowO = KnowA
    END SELECT
  CASE IS >= 200
    KnowO = KnowA
END SELECT
PUT #3, i, KnowO
NEXT i

'Complete genesis
PRINT "Genesis completed."
CLOSE

'Analyse
SHELL "Entropy " + F1$
SHELL "Entropy " + F2$
SHELL "Entropy " + O$

END

```

Chapter 3

3.1 Overview of the system

We shall begin the design details of the mobile robot developed during the course of this project with an overview of the system attributes before moving onto the mechatronics design. The mobile robot was not procured as a kit but rather constructed from discrete components as far as feasible. This essentially was a mechatronics design task. The features of the robot are given below.

Sensory inputs:

- I. Stereo vision (Logitech Quickcam Express x 2)*
- II. Obstacle detection (ultrasonic sensor)*
- III. Audio input (FM microphone)*

Outputs:

- I. Translation (movement)*
- II. Rotation (steering)*
- III. Audio output*

Computer Interface:

- I. Centronics parallel port (non-typical)*
- II. Parity checking on control word*
- III. Real-time response (parity error, busy and obstacle)*
- IV. Separate vision interface (dual USB port)*

The robot shall henceforth be called Angel-1. Angel is an abbreviation for "Autonomous Natural-interfaced General-purpose Learner" and the numeral '1' specifies that this is an original concept and not built upon an existing platform. Further enhancements may be termed Angel-2, Angel-3 and so on. Two photographs of Angel-1 and its control circuitry are given on page 51.

A complete description of the mechanical design of Angel-1 would not further our point. We shall rather take a look at its major components such as the steering and drive motor assemblies, the ultrasonic-transducer tower assembly, the gripper assembly and a brief description of the construction method. However, the control circuit needs to be specified in detail as it is directly related to the software designed for the robot.

3.2 Mechanical design of Angel-1

The gripper mechanism and ultra-sonic sensor positioning mechanism were independently developed from scratch. Complete assemblies have been used for steering and drive motor. The drive motor assembly consists of gearings for torque amplification. The steering utilises an Ackerman steering model. The chassis has been designed using "Mechanix" construction pieces. The gripper mechanism is illustrated in the drawing on page ___. The ultra-sonic sensor positioning mechanism utilises a stepper motor. For this purpose we have used a 50-step motor. An opto-slot has been provided for detection of zero-displacement position. The illustration is on page ____.

3.3 Control system design of Angel-1

The control system of Angel-1 is based on a master-slave configuration of four PIC16F84A microcontrollers. The system has been kept modular in design, so as to achieve an open-ended architecture. It interfaces to a PC using the Centronics parallel port, used in a non-typical configuration. The schematic for the control circuit is given on page ___. The overall system organisation is illustrated on page ___. We shall first describe each of the ten modules of the control system before moving onto the theory of operation of the system.

The first five modules consist of the interface and microcontrollers, while the next five modules consist of the Angel-1's power supply and motor drivers.

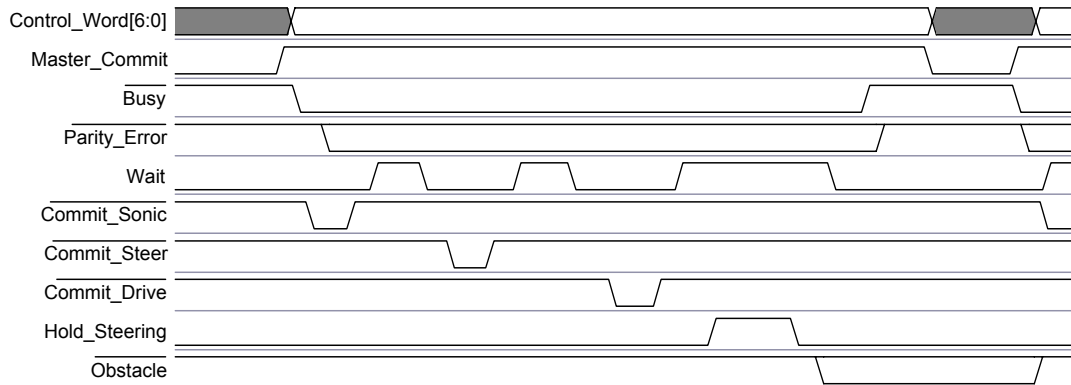
The system requires a DC supply of 12 volts, for which we have used a PC-AT power supply. In module-1 the output from the parallel port is buffered to provide the required fan-out using a 74LS245. Module-2 consists of the master PIC 16F84, a 74LS139 1:4 demultiplexer for decoding the commit signal from the master mcu, and a 4-wide 2-AOI gate for combining the 'Wait' signal. Modules-3, 4 and 5 each have a PIC 16F84 and control the stepper, steering/gripper, and the drive motors, respectively. All the PIC's have been clocked using 4Mhz crystals. The listing of the firmware source-code for modules-2, 3, 4 and 5 along with the flowcharts is given starting on page _____. All the above modules use a 78'05 voltage regulator. Module-6 is the Angel-1 power supply, providing outputs at 5v, 9v and 12v. Modules-7, 8 and 9 control a motor each. They consist of a DPDT reed relay (5v), forming an H-bridge around the motor. The supply current is switched using a 2N3904 NPN transistor and a TIP31 power transistor in Darlington pair configuration. This allows for PWM of the supply to the motors. Module-10 drives the stepper motor and consists of four Darlington pairs; identical to the one's described above. Heat sinks, resistors and capacitors have been used as appropriate. The ultrasonic sensor was purchased as a kit and assembled. It uses an interference-based technique for movement detection. An obstacle "moving" into its range is detected; hence it works only while Angel-1 is in motion.

The parallel port lines are used as given below:

- | | | |
|------|--------|-----------------------|
| i. | D0-D5: | Control word (input) |
| ii. | D6: | Parity (input) |
| iii. | D7: | MCommit (input) |
| iv. | Busy: | Busy (output) |
| v. | 'PE: | 'Obstacle (output) |
| vi. | Error: | Parity error (output) |

3.4 Theory of operation of Angel-1 control system

The diagram illustrating the system configuration on page ___ and the flowcharts for the firmware source-code make the operation of the system obvious. However for the sake of continuity we shall state the theory of operation in brief here. For this let us consider the timing diagram for the system given below.



Angel-1 Timing Schematic

The control word is 6-bits wide (D0-D6). The bit designations are given below:

- i. D0: Gripper Open/'Close
- ii. D1: Left
- iii. D2: Right
- iv. D3: 'Forward/Backward
- v. D4: Walk/'Run
- vi. D5: Sonar On/'Off

The control word becomes valid only when Master-Commit (D7: MCommit) is high. On a MCommit, the master mcu raises the Busy signal and checks for even parity of control word on D6. If there is no parity error it executes the commit loop for the three slave-microcontrollers, otherwise it raises Parity_error signal. The commit loop consists of raising the commit for a slave mcu, and waiting for the 'Wait' signal to go high and then down. The slave mcu's perform their task during the commit cycle. The exception is the steering/gripper mcu, which completes the commit cycle by releasing "Wait" but does not steer

unless it receives an HStr (Hold Steering) signal from the drive mcu. When the drive mcu has completed the commit cycle and released the "Wait" signal, the master mcu releases the "Busy" signal and starts accepting commands. In case of an obstacle being detected the drive mcu "brakes" Angel-1 and signals "Obstacle" on 'PE, bypassing the master mcu. In brief it may be stated here that this model of execution has been designed to mimic the local and central reflexes in living organisms. This is illustrated on page _____. The set of valid control words along with mnemonics is given on page _____.

3.5 Limitations of the system

Before we move onto the behavioural (software) design of Angel-1 let us consider a few of its identified physical limitations. These are listed below.

- Radius of turning is very large and a complex set of movements is needed for turning in a small area. Differential steering/drive would have solved this problem.
- Localisation abilities are very limited. Sonar ranging may be implemented using the existing ultrasonic transducer.
- Attachment to the mother computer using an umbilical restricts it to a fixed area, and lays severe restraints on its abilities to move. Wireless linking is an utter requirement.

Blank Page

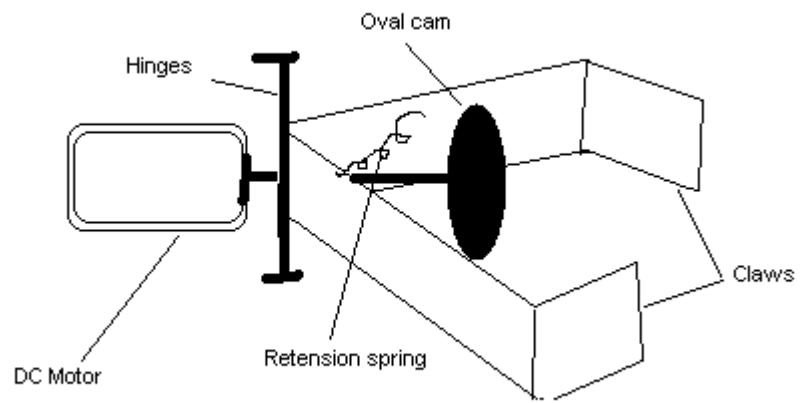


Angel-1 Autonomous Mobile Robot

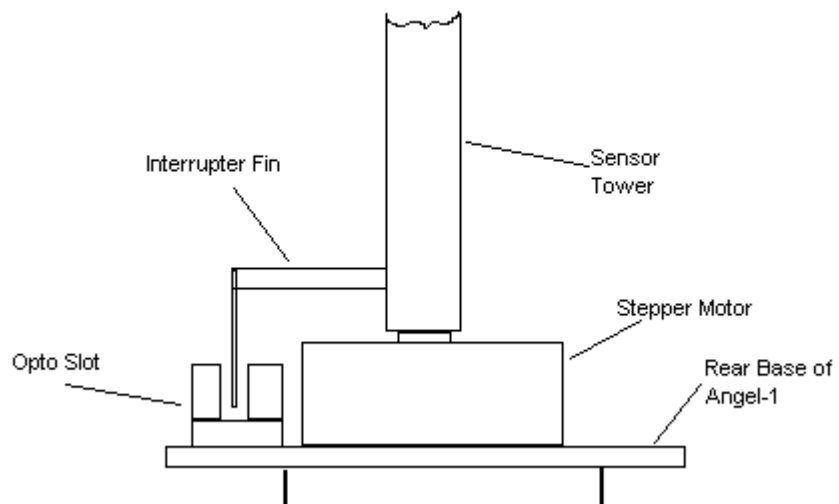


Angel-1 Controller Circuit

Blank Page

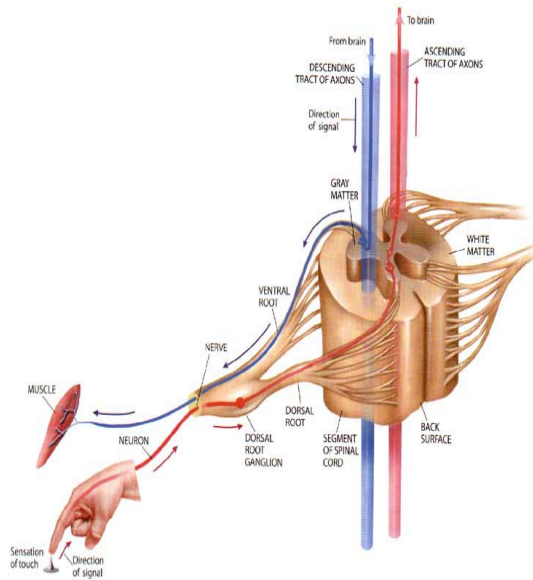
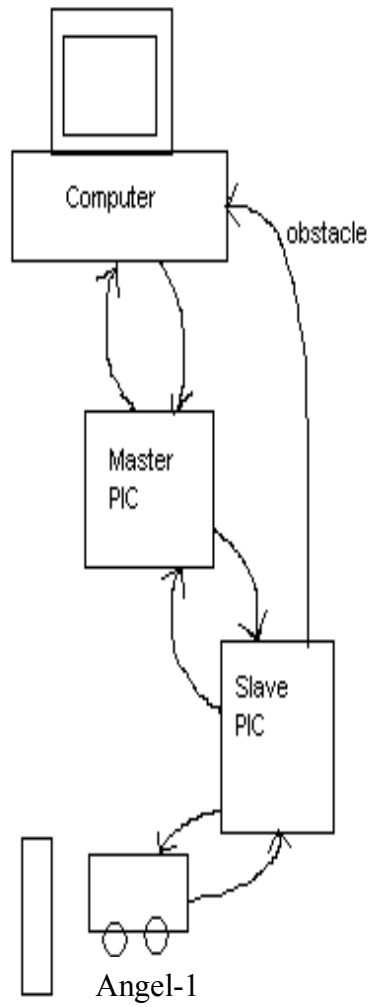


Gripper Mechanism



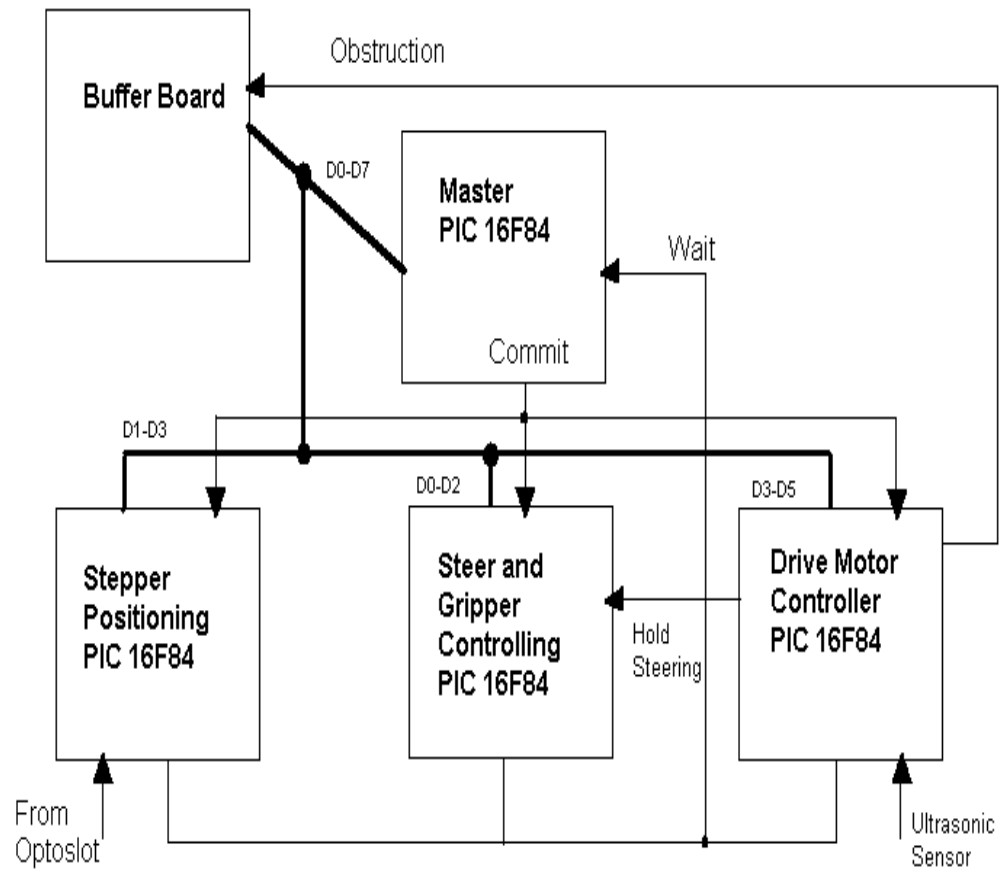
Opto-slot for zero displacement positioning of stepper

Blank Page



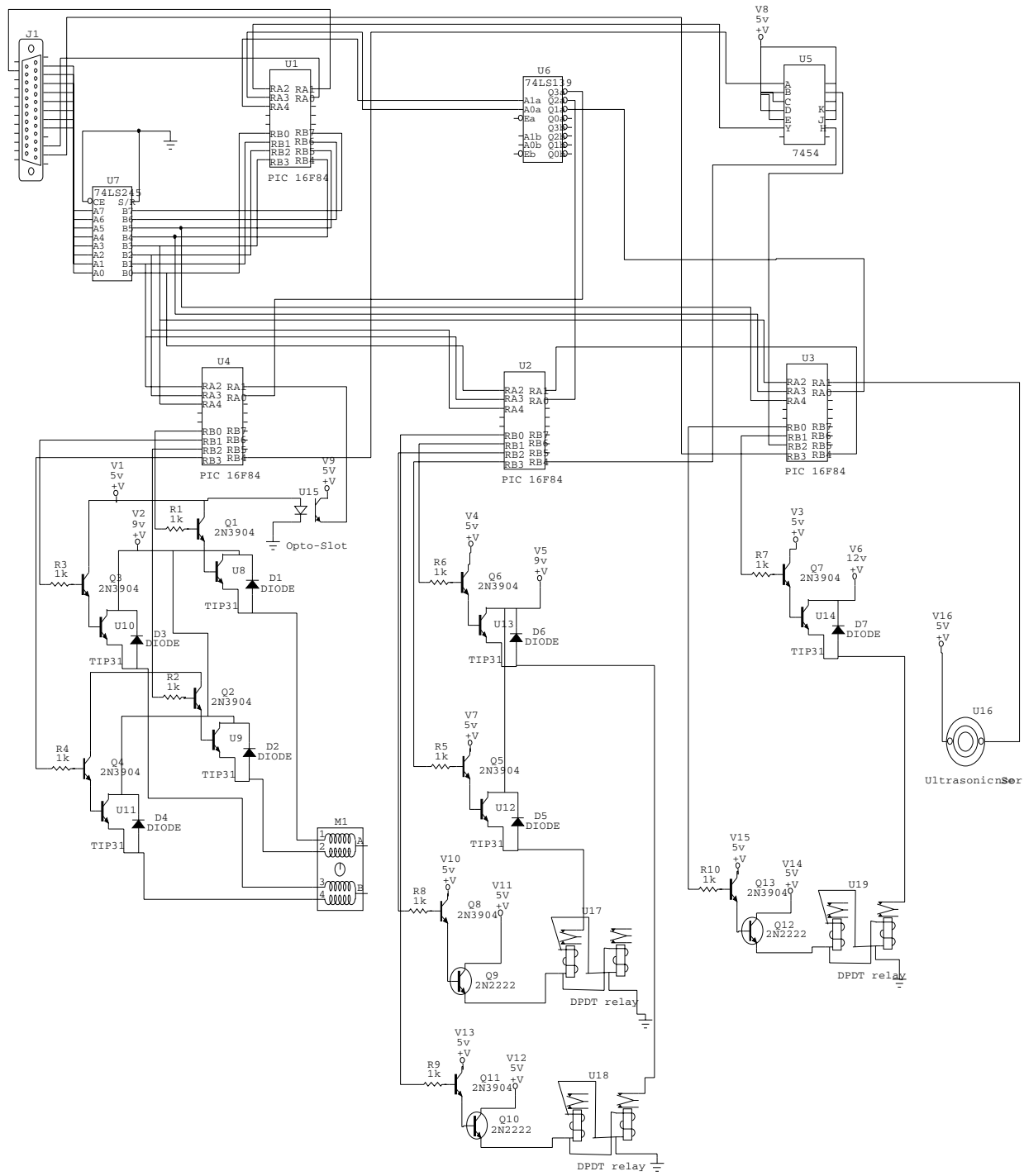
Comparison of Angel-1 and Human Reflex and Central Stimulus Response

Blank Page



System Configuration of Angel-1

Blank Page



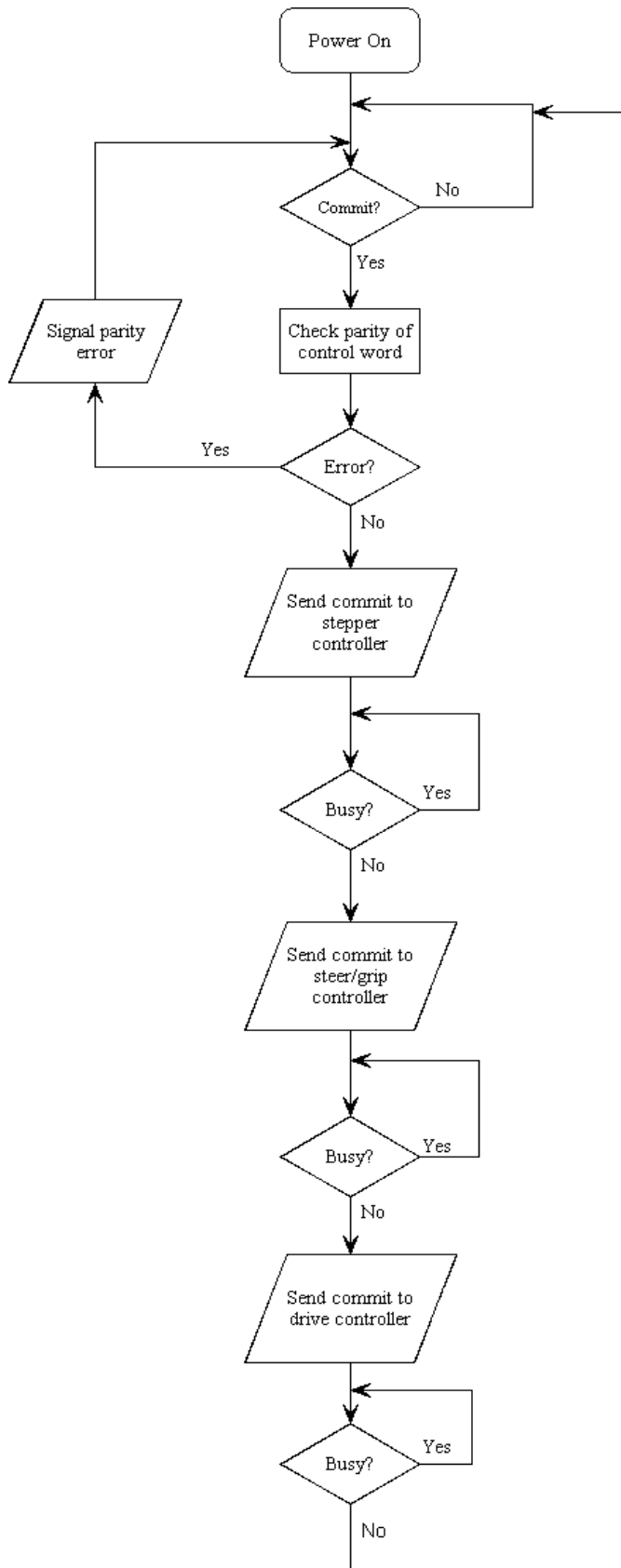
Schematic of Angel-1 Control System

Blank Page

Angel-1 Control Words and Mnemonics

Parity	D5	D4	D3	D2	D1	D0	cw	cwp	high	mnemonics				
1	0	0	0	0	0	0	0	64	192	gripper close	centre	forward	sonar off	run
0	1	0	0	0	0	0	32	32	160	gripper close	centre	forward	sonar on	run
0	0	1	0	0	0	0	16	16	144	gripper close	centre	forward	sonar off	walk
1	1	1	0	0	0	0	48	112	240	gripper close	centre	forward	sonar on	walk
0	0	0	1	0	0	0	8	8	136	gripper close	centre	backward	sonar off	run
1	1	0	1	0	0	0	40	104	232	gripper close	centre	backward	sonar on	run
1	0	1	1	0	0	0	24	88	216	gripper close	centre	backward	sonar off	walk
0	1	1	1	0	0	0	56	56	184	gripper close	centre	backward	sonar on	walk
0	0	0	0	1	0	0	4	4	132	gripper close	right	forward	sonar off	run
1	1	0	0	1	0	0	36	100	228	gripper close	right	forward	sonar on	run
1	0	1	0	1	0	0	20	84	212	gripper close	right	forward	sonar off	walk
0	1	1	0	1	0	0	52	52	180	gripper close	right	forward	sonar on	walk
1	0	0	1	1	0	0	12	76	204	gripper close	right	backward	sonar off	run
0	1	0	1	1	0	0	44	44	172	gripper close	right	backward	sonar on	run
0	0	1	1	1	0	0	28	28	156	gripper close	right	backward	sonar off	walk
1	1	1	1	1	0	0	60	124	252	gripper close	right	backward	sonar on	walk
0	0	0	0	0	1	0	2	2	130	gripper close	left	forward	sonar off	run
1	1	0	0	0	1	0	34	98	226	gripper close	left	forward	sonar on	run
1	0	1	0	0	1	0	18	82	210	gripper close	left	forward	sonar off	walk
0	1	1	0	0	1	0	50	50	178	gripper close	left	forward	sonar on	walk
1	0	0	1	0	1	0	10	74	202	gripper close	left	backward	sonar off	run
0	1	0	1	0	1	0	42	42	170	gripper close	left	backward	sonar on	run
0	0	1	1	0	1	0	26	26	154	gripper close	left	backward	sonar off	walk
1	1	1	1	0	1	0	58	122	250	gripper close	left	backward	sonar on	walk
1	0	0	0	1	1	0	6	70	198	gripper close	invalid	forward	sonar off	run
0	1	0	0	1	1	0	38	38	166	gripper close	invalid	forward	sonar on	run
0	0	1	0	1	1	0	22	22	150	gripper close	invalid	forward	sonar off	walk
1	1	1	0	1	1	0	54	118	246	gripper close	invalid	forward	sonar on	walk
0	0	0	1	1	1	0	14	14	142	gripper close	invalid	backward	sonar off	run
1	1	0	1	1	1	0	46	110	238	gripper close	invalid	backward	sonar on	run
1	0	1	1	1	1	0	30	94	222	gripper close	invalid	backward	sonar off	walk
0	1	1	1	1	1	0	62	62	190	gripper close	invalid	backward	sonar on	walk
0	0	0	0	0	0	1	1	1	129	gripper open	centre	forward	sonar off	run
1	1	0	0	0	0	1	33	97	225	gripper open	centre	forward	sonar on	run
1	0	1	0	0	0	1	17	81	209	gripper open	centre	forward	sonar off	walk
0	1	1	0	0	0	1	49	49	177	gripper open	centre	forward	sonar on	walk
1	0	0	1	0	0	1	9	73	201	gripper open	centre	backward	sonar off	run
0	1	0	1	0	0	1	41	41	169	gripper open	centre	backward	sonar on	run
0	0	1	1	0	0	1	25	25	153	gripper open	centre	backward	sonar off	walk
1	1	1	1	0	0	1	57	121	249	gripper open	centre	backward	sonar on	walk
1	0	0	0	1	0	1	5	69	197	gripper open	right	forward	sonar off	run
0	1	0	0	1	0	1	37	37	165	gripper open	right	forward	sonar on	run
0	0	1	0	1	0	1	21	21	149	gripper open	right	forward	sonar off	walk
1	1	1	0	1	0	1	53	117	245	gripper open	right	forward	sonar on	walk
0	0	0	1	1	0	1	13	13	141	gripper open	right	backward	sonar off	run
1	1	0	1	1	0	1	45	109	237	gripper open	right	backward	sonar on	run
1	0	1	1	1	0	1	29	93	221	gripper open	right	backward	sonar off	walk
0	1	1	1	1	0	1	61	61	189	gripper open	right	backward	sonar on	walk
1	0	0	0	0	1	1	3	67	195	gripper open	left	forward	sonar off	run
0	1	0	0	0	1	1	35	35	163	gripper open	left	forward	sonar on	run
0	0	1	0	0	1	1	19	19	147	gripper open	left	forward	sonar off	walk
1	1	1	0	0	1	1	51	115	243	gripper open	left	forward	sonar on	walk
0	0	0	1	0	1	1	11	11	139	gripper open	left	backward	sonar off	run
1	1	0	1	0	1	1	43	107	235	gripper open	left	backward	sonar on	run
1	0	1	1	0	1	1	27	91	219	gripper open	left	backward	sonar off	walk
0	1	1	1	0	1	1	59	59	187	gripper open	left	backward	sonar on	walk
0	0	0	0	1	1	1	7	7	135	gripper open	invalid	forward	sonar off	run
1	1	0	0	1	1	1	39	103	231	gripper open	invalid	forward	sonar on	run
1	0	1	0	1	1	1	23	87	215	gripper open	invalid	forward	sonar off	walk
0	1	1	0	1	1	1	55	55	183	gripper open	invalid	forward	sonar on	walk
1	0	0	1	1	1	1	15	79	207	gripper open	invalid	backward	sonar off	run
0	1	0	1	1	1	1	47	47	175	gripper open	invalid	backward	sonar on	run
0	0	1	1	1	1	1	31	31	159	gripper open	invalid	backward	sonar off	walk
1	1	1	1	1	1	1	63	127	255	gripper open	invalid	backward	sonar on	walk

Blank Page



Master Microcontroller Firmware Flowchart

```
;*****
;Abhishek Choudhary
;SG-129546
;AMIETE(CSE)-Project C18
;Master controller ANGEL-1 mobile robot
;18th August, 2003
;4MHz crystal; XT mode; WDT off; PUT off
;*****
```

PROCESSOR 16F84A

```
;Register label equates
```

```
PORTA      EQU      05
PORTB      EQU      06
Count      EQU      0C
State      EQU      0D
TestB      EQU      0E
Count2     EQU      0F
Badpar     EQU      10
Temp       EQU      11
```

```
;PortA register bit labels
```

```
Parerr     EQU      0
Busy       EQU      1
Wait       EQU      2
Com0       EQU      3
Com1       EQU      4
```

```
;PortB register bit labels
```

```
CWGr       EQU      0
CWS0       EQU      1
CWS1       EQU      2
CWM0       EQU      3
CWM1       EQU      4
CWM2       EQU      5
Parity     EQU      6
Commit     EQU      7
```

```
;Start program *****
```

```
;Initialise .....
```

```
MOVLW b'11100100'      ;Select I/O status of pins
TRIS PORTA
MOVLW b'11111111'
TRIS PORTB
CLRF PORTA
CLRF PORTB
CLRF Count
CLRF State
CLRF TestB
CLRF Count2
CLRF Badpar
CLRF Temp
GOTO main              ;Delay=14us
```

```
;Delay subroutine .....
```

```
delay      MOVLW      0FF
           MOVWF      Count
down       DECFSZ     Count
```

```

        GOTO  down
        RETURN

;Parity-check subroutine .....
chkpar  NOP
        MOVF  PORTB,W
        MOVWF TestB
        MOVLW 006
        MOVWF Count2
        MOVLW 001
        MOVWF State
        MOVLW 001
prloop  BTFSC  TestB,0
        XORWF State
        RRF      TestB
        DECFSZ   Count2
        GOTO  prloop
        CLRW
        BTFSC  PORTB,Parity
        MOVLW 001
        XORWF State
        INCF  State
        RETURN

;Parity check failed .....
prfail  NOP
        BCF      PORTA,Parerr
        CALL  delay
        CALL  delay
        MOVLW 001
        MOVWF Badpar
        RETURN

;Clear commit subroutine .....
clrcom  NOP
        MOVF  PORTA,W
        MOVWF Temp
        BCF      Temp,Com0
        BCF      Temp,Com1
        MOVF  Temp,W
        MOVWF PORTA
        RETURN

;Ultrasonic sensor tower commit subroutine .....
soncom  NOP
        CALL  clrcom
        CALL  delay
        CALL  delay
        BSF      PORTA,Com0
        CALL  delay
        CALL  delay
        CALL  delay
        CALL  delay
        CALL  delay
        CALL  delay
        CALL  delay
        CALL  delay
        BCF      PORTA,Com0
        CALL  delay
        CALL  delay

```

```

                RETURN

;Steer/Grip commit subroutine
grpcom          NOP
                CALL  clrcom
                CALL  delay
                CALL  delay
                BSF   PORTA, Com1
                CALL  delay
                CALL  delay
                CALL  delay
                CALL  delay
                CALL  delay
                CALL  delay
                BCF   PORTA, Com1
                CALL  delay
                CALL  delay
                RETURN

;Drive commit subroutine .....
drvcom          NOP
                CALL  clrcom
                CALL  delay
                CALL  delay
                MOVF  PORTA, W
                MOVWF Temp
                BSF   Temp, Com0
                BSF   Temp, Com1
                MOVF  Temp, W
                MOVWF PORTA
                CALL  delay
                CALL  delay
                CALL  delay
                CALL  delay
                CALL  delay
                CALL  delay
                CALL  delay
                CALL  clrcom
                CALL  delay
                CALL  delay
                CALL  delay
                RETURN

;Main program loop .....
main           NOP
                CLRF  Badpar
loop0          BTFSC PORTB, Commit
                GOTO  loop0
                BSF   PORTA, Parerr
                BSF   PORTA, Busy
loopa          NOP
                BTFSS PORTB, Commit
                GOTO  loopa
                BCF   PORTA, Busy                ;Min delay=24us
                CALL  chkpar
                DECFSZ State
                CALL  prfail                ;Min delay=73us
                DECFSZ Badpar
                GOTO  caryon
                GOTO  main

```

```
caryon      NOP

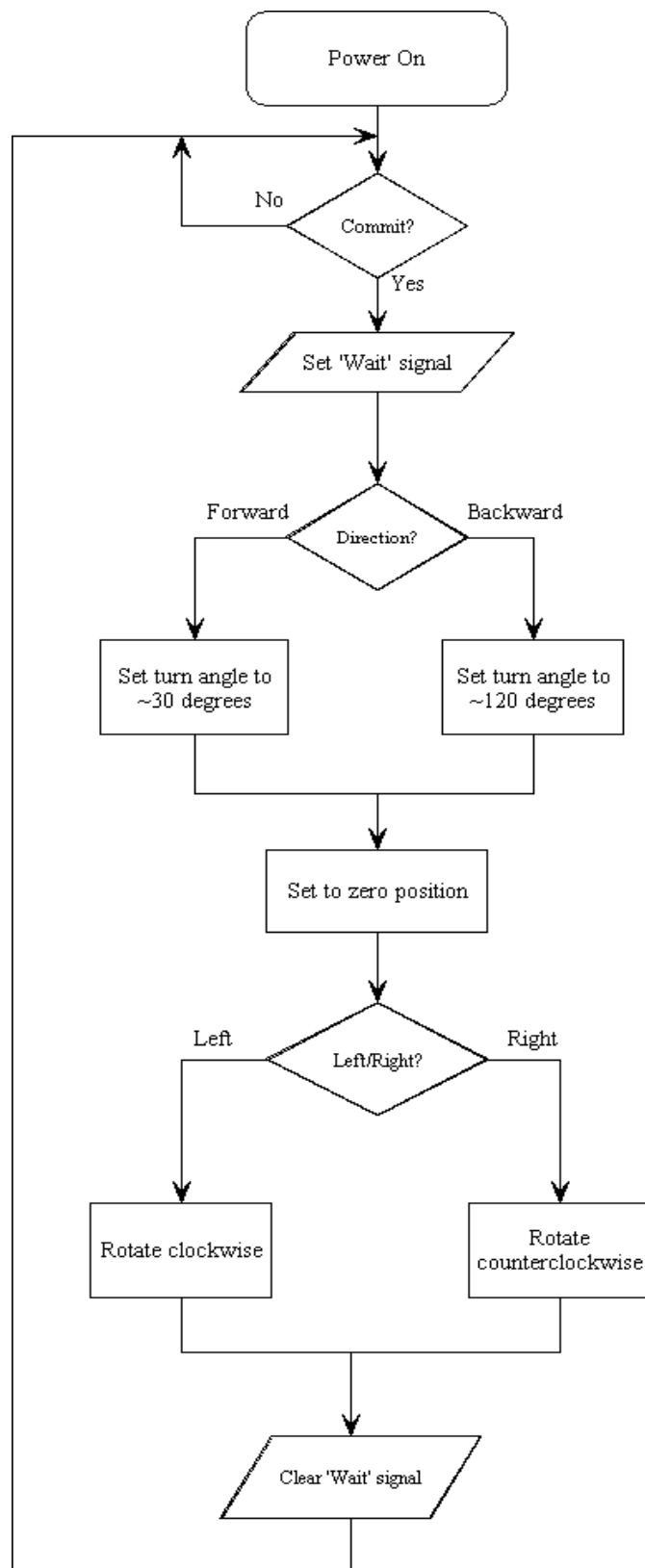
            CALL  soncom                      ;Min delay=77us
loopsa      BTFSC PORTA,Wait
            GOTO  loopsa
loopsb      BTFSS PORTA,Wait
            GOTO  loopsb

            CALL  grpcom
loopga      BTFSC PORTA,Wait
            GOTO  loopga
loopgb      BTFSS PORTA,Wait
            GOTO  loopgb

            CALL  drvcom
loopda      BTFSC PORTA,Wait
            GOTO  loopda
loopdb      BTFSS PORTA,Wait
            GOTO  loopdb

            GOTO  main

            END
```



Ultrasonic-sensor Stepper Controller Flowchart

```

;*****
;Abhishek Choudhary
;SG-129546
;AMIETE(CSE)-Project C18
;Stepper controller for ultrasonic sensor tower
;17th August, 2003
;*****

```

PROCESSOR 16F84A

```

;Register label equates
PORTA EQU      05
PORTB EQU      06
Count EQU      0C
Count2 EQU     0D
Temp EQU       0E
Dirset EQU     0F
Last EQU       10
Rptcnt EQU    11
Test EQU       12
First EQU      13

```

```

;PortA register bit labels
Commit EQU     0
Opto EQU       1
CWS0 EQU       2
CWS1 EQU       3
CWM0 EQU       4

```

```

;PortB register bit labels
StepA EQU      0
StepB EQU      1
StepC EQU      2
StepD EQU      3
Busy EQU       4

```

```

;Start program *****

```

```

;Initialise .....
        MOVLW b'00011111'
        TRIS  PORTA
        MOVLW b'11100000'
        TRIS  PORTB
        CLRF  PORTA
        CLRF  PORTB
        CLRF  Count
        CLRF  Count2
        CLRF  Dirset
        CLRF  Temp
        CLRF  Last
        CLRF  Rptcnt
        CLRF  Test
        CLRF  First
        GOTO  main

```

```

;Delay subroutine .....
delay MOVLW 0FF
        MOVWF Count
down  MOVLW 00C

```

```

        MOVWF Count2
down2  DECFSZ    Count2
        GOTO    down2
        DECFSZ    Count
        GOTO    down
        RETURN

;Step CCW subroutine .....
ccw    BCF      PORTB,StepD
        BSF      PORTB,StepA
        CALL    delay
        CALL    delay
        BCF      PORTB,StepA
        BSF      PORTB,StepB
        CALL    delay
        CALL    delay
        BCF      PORTB,StepB
        BSF      PORTB,StepC
        CALL    delay
        CALL    delay
        BCF      PORTB,StepC
        BSF      PORTB,StepD
        CALL    delay
        CALL    delay
        RETURN

;Step CW subroutine .....
cw    BCF      PORTB,StepA
        BSF      PORTB,StepD
        CALL    delay
        CALL    delay
        BCF      PORTB,StepD
        BSF      PORTB,StepC
        CALL    delay
        CALL    delay
        BCF      PORTB,StepC
        BSF      PORTB,StepB
        CALL    delay
        CALL    delay
        BCF      PORTB,StepB
        BSF      PORTB,StepA
        CALL    delay
        CALL    delay
        RETURN

;Check-last subroutine .....
chklst  NOP
        CLRF    Test
        BTFSC  PORTA,CWS0
        BSF     Test,CWS0
        BTFSC  PORTA,CWS1
        BSF     Test,CWS1
        BTFSC  PORTA,CWM0
        BSF     Test,CWM0
        MOVF   Last,W
        XORWF  Test
        INCF   Test
        RETURN

```

```

;Update-last subroutine .....
updlst    NOP
          CLRf    Last
          BTFSC   PORTA,CWS0
          BSF     Last,CWS0
          BTFSC   PORTA,CWS1
          BSF     Last,CWS1
          BTFSC   PORTA,CWM0
          BSF     Last,CWM0
          RETURN

;Main program loop .....
main     NOP
          CALL    delay
          CALL    delay

loop0    NOP
          BTFSS   PORTA,Commit
          GOTO    loop0

          CLRf    PORTB

loopa    NOP
          BTFSC   PORTA,Commit
          GOTO    loopa
          BSF     PORTB,Busy

          BTFSS   First,0
          GOTO    frsttm
          CALL    chklst
          DECFSZ   Test
          GOTO    fresh
          GOTO    repeat

frsttm   NOP
          INCF    First
          GOTO    fresh

repeat   NOP
          DECFSZ   Rptcnt
          GOTO    rpt0
          GOTO    fresh

rpt0     CALL    delay
          CALL    delay
          GOTO    main

fresh    NOP
          CALL    updlst
          MOVLW   010
          MOVWF   Rptcnt

          BTFSS   Dirset,0
          GOTO    ccset

          MOVLW   019
          MOVWF   Temp

setcw0   BTFSS   PORTA,Opto
          GOTO    set0
          CALL    cw

```

```

        DECFSZ  Temp
        GOTO   setcw0
        MOVLW  032
        MOVWF  Temp
setcw1  BTFSS   PORTA, Opto
        GOTO   setend
        CALL  ccw
        DECFSZ  Temp
        GOTO   setcw1

ccset  NOP

        MOVLW  019
        MOVWF  Temp
setcc0 BTFSS   PORTA, Opto
        GOTO   setend
        CALL  ccw
        DECFSZ  Temp
        GOTO   setcc0
        MOVLW  032
        MOVWF  Temp
setcc1 BTFSS   PORTA, Opto
        GOTO   set0
        CALL  cw
        DECFSZ  Temp
        GOTO   setcc1

set0   BTFSC   PORTA, Opto
        GOTO   set0a
        CALL  cw
        GOTO   set0
set0a  BTFSS  PORTA, Opto
        GOTO   setend
        CALL  ccw
        GOTO   set0a

setend  NOP

        BTFSC  PORTA, CWM0
        GOTO   forwrđ

bkward  NOP
        BTFSS  PORTA, CWS0
        GOTO   bkrt

bklf   NOP

        BTFSC  PORTA, CWS1
        GOTO   bkerr
        BCF    Dirset, 0
        MOVLW  013
        MOVWF  Temp
loop1  CALL   cw
        DECFSZ  Temp
        GOTO   loop1
        GOTO   main

bkrt   NOP

        BTFSS  PORTA, CWS1
        GOTO   bkcntř
        BSF    Dirset, 0

```

```

                                MOVLW 013
                                MOVWF Temp
loop2 CALL ccw
                                DECFSZ      Temp
                                GOTO  loop2
                                GOTO  main

bkerr NOP
                                GOTO main

bkcntr NOP
                                GOTO main

forwrdrd NOP
                                BTFSS PORTA,CWS0
                                GOTO  fdrt

fdlfrd NOP
                                BTFSC PORTA,CWS1
                                GOTO  fderr
                                BCF      Dirset,0
                                MOVLW 006
                                MOVWF Temp
loop3 CALL ccw
                                DECFSZ      Temp
                                GOTO  loop3
                                GOTO  main

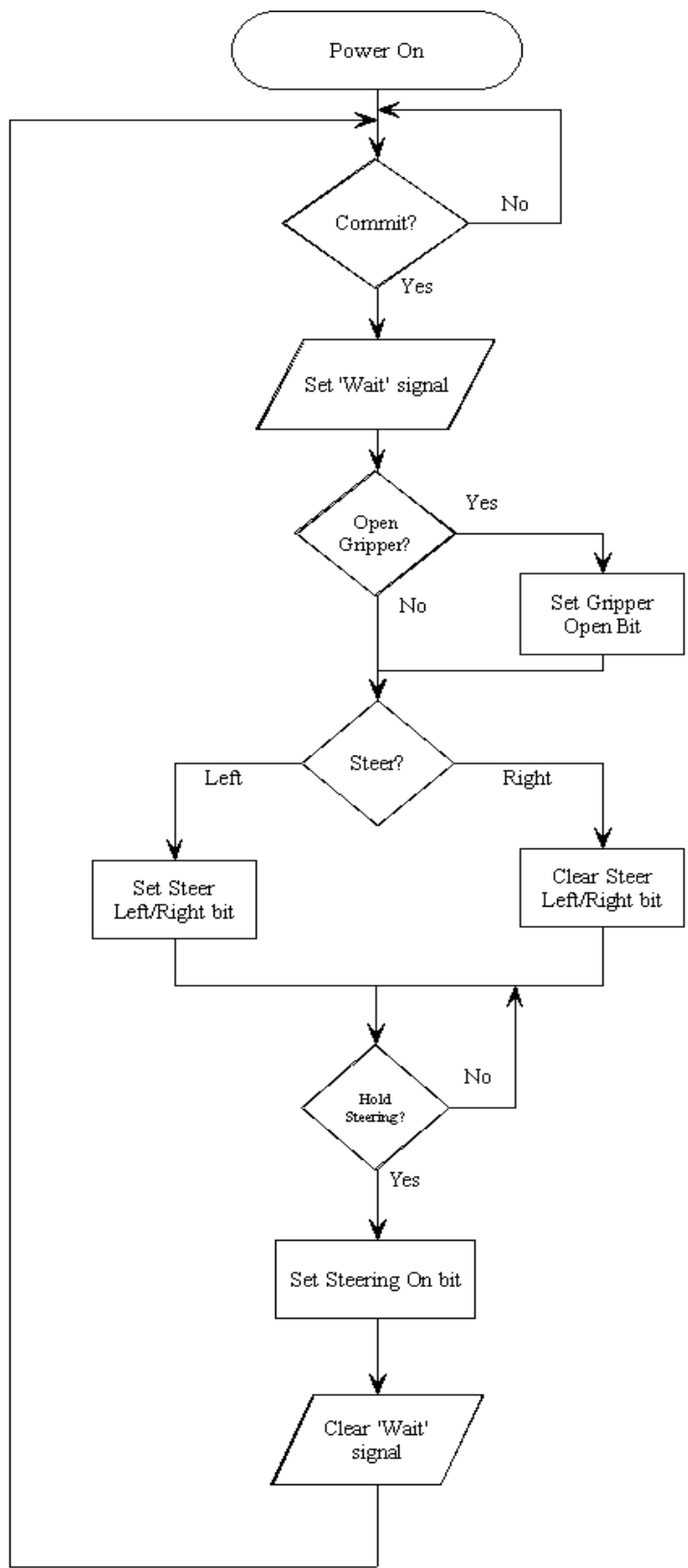
fdrt NOP
                                BTFSS PORTA,CWS1
                                GOTO  fdcntr
                                BSF      Dirset,0
                                MOVLW 006
                                MOVWF Temp
loop4 CALL ccw
                                DECFSZ      Temp
                                GOTO  loop4
                                GOTO  main

fderr NOP
                                GOTO main

fdcntr NOP
                                GOTO main

                                END

```



Steering/Gripper Microcontroller Flowchart

```

;*****
;Abhishek Choudhary
;SG-129546
;AMIETE(CSE)-Project C18
;Steering and gripper controller
;18th August, 2003
;*****

```

PROCESSOR 16F84A

```

;Register label equates
PORTA EQU      05
PORTB EQU      06
Count EQU      0C
Count2      EQU      0D
StrCh EQU      0E

```

```

;PortA register bit labels
Commit      EQU      0
HStr EQU      1
CWGr EQU      2
CWS0 EQU      3
CWS1 EQU      4

```

```

;PortB register bit labels
GrpRL EQU      0
GrpON EQU      1
StrRL EQU      2
StrON EQU      3
Busy EQU      4

```

```

;Start program *****

```

```

;Initialise .....
        MOVLW b'00011111'
        TRIS  PORTA
        MOVLW b'11100000'
        TRIS  PORTB
        CLRF  PORTA
        CLRF  PORTB
        CLRF  Count
        CLRF  Count2
        CLRF  StrCh
        GOTO  main

```

```

;Delay subroutine .....
delay MOVLW 0FF
        MOVWF Count
down  MOVLW 00C
        MOVWF Count2
down2 DECFSZ  Count2
        GOTO  down2
        DECFSZ  Count
        GOTO  down
        RETURN

```

```

;Main program loop .....
main  NOP
        CALL  delay

```

```

                CALL    delay
loop0 NOP
                BTFSS  PORTA,Commit
                GOTO   loop0

                CLRF   PORTB

loopa NOP
                BTFSC  PORTA,Commit
                GOTO   loopa
                BSF    PORTB,Busy
                CALL   delay
                CALL   delay

choose        BTFSC  PORTA,CWS1
                GOTO   right

left  BTFSS  PORTA,CWS0
                GOTO   center
                BCF    PORTB,StrRL
                CLRF  StrCh
                CALL   delay
                CALL   delay
                GOTO  grptst

right BTFSC  PORTA,CWS0
                GOTO   strerr
                BSF    PORTB,StrRL
                CLRF  StrCh
                CALL   delay
                CALL   delay
                GOTO  grptst

strerr        NOP
                GOTO  center

center        NOP
                CLRF  StrCh
                INCF  StrCh
                GOTO  grptst

grptst        NOP
                BTFSC  PORTA,CWGr
                GOTO   hstst1
                BSF    PORTB,GrpON
                CALL   delay
                CALL   delay
                CALL   delay
                BSF    PORTB,GrpRL
                CALL   delay
                CALL   delay
                BCF    PORTB,GrpRL
                CALL   delay
                CALL   delay

hstst1        BCF    PORTB,Busy

hstest        BTFSS  PORTA,HStr

```

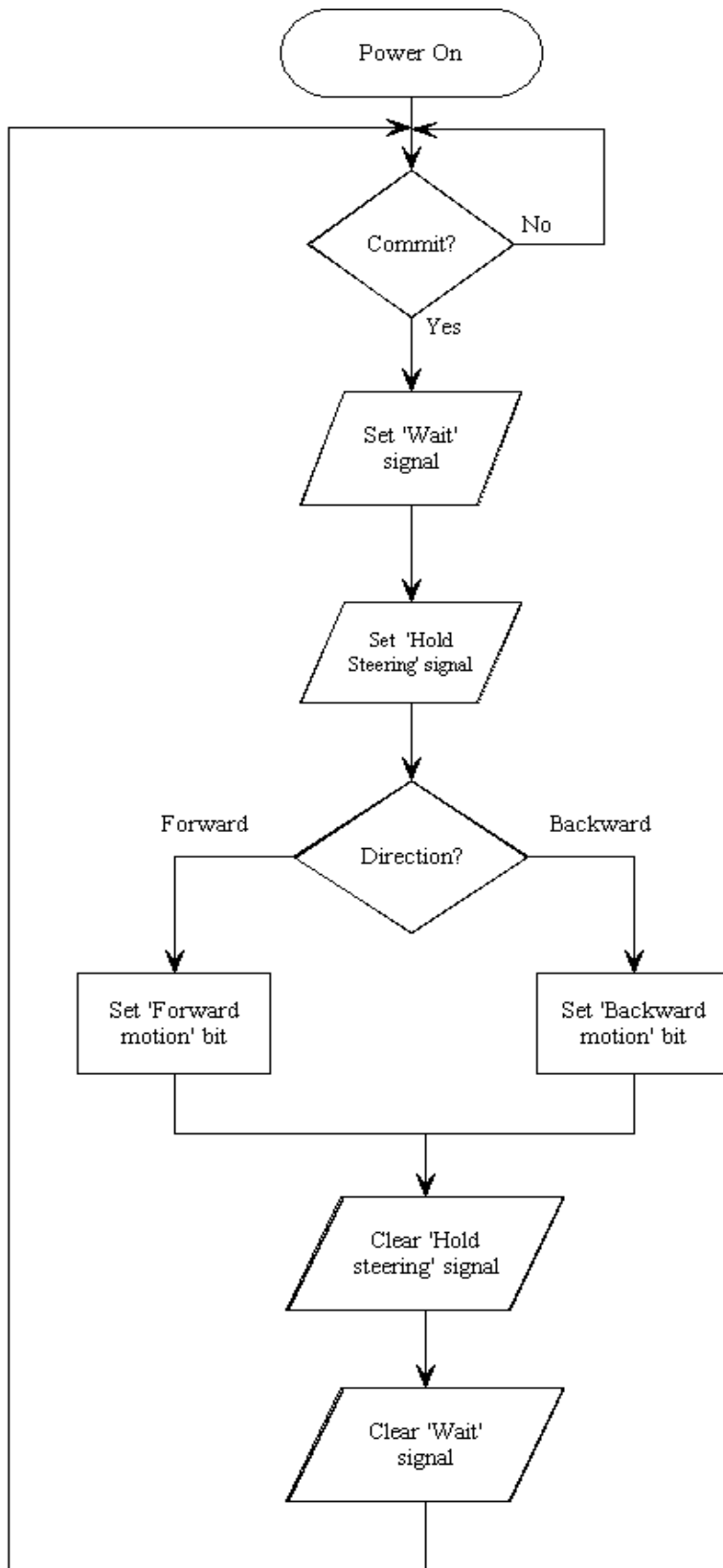
```
GOTO hstest

DECFSZ      StrCh
BSF         PORTB, StrON

hstst0     BTFSC PORTA, HStr
           GOTO hstst0

           GOTO main

           END
```



Drive Microcontroller Flowchart

```

;*****
;Abhishek Choudhary
;SG-129546
;AMIETE(CSE)-Project C18
;Drive controller
;18th August, 2003
;*****

PROCESSOR 16F84A

;Register label equates
PORTA EQU      05
PORTB EQU      06
Count EQU      0C
Temp EQU       0D
Count2 EQU     0E
RLen EQU       0F

;PortA register bit labels
Commit EQU     0
Sonar EQU     1
CWM0 EQU     2
CWM1 EQU     3
CWM2 EQU     4

;PortB register bit labels
DrvBF EQU     0
DrvON EQU     1
Busy EQU     2
Obst EQU     3
HStr EQU     4

;Start program *****

;Initialise .....
        MOVLW b'00011111'
        TRIS  PORTA
        MOVLW b'11100000'
        TRIS  PORTB
        CLRF  PORTA
        CLRF  PORTB
        CLRF  Count
        CLRF  Temp
        CLRF  Count2
        CLRF  RLen
        GOTO  main

;Delay subroutine .....
delay MOVLW 0FF
        MOVWF Count
down  MOVLW 00C
        MOVWF Count2
down2 DECFSZ Count2
        GOTO down2
        DECFSZ Count
        GOTO down
        RETURN

;Main program loop .....

```

```

main NOP
    CALL delay
    CALL delay

loop0 NOP
    BTFSS PORTA,Commit
    GOTO loop0

    BCF PORTB,DrvON
    BCF PORTB,DrvBF
    BCF PORTB,HStr
    BCF PORTB,Busy

loopa NOP
    BTFSC PORTA,Commit
    GOTO loopa

    BSF PORTB,Obst
    BSF PORTB,Busy

    CLRf RLen
    CLRw
    MOVLW 011
    BTFSS PORTA,CWM1
    ADDLW 030
    MOVWF RLen

    BTFSC PORTA,CWM0
    BSF PORTB,DrvBF

    BSF PORTB,DrvON
    CALL delay
    CALL delay
    CALL delay
    CALL delay
    BSF PORTB,HStr
    CALL delay
    CALL delay

rloop BTFSS PORTA,CWM2
    GOTO noobst
    BTFSS PORTA,Sonar
    GOTO noobst
    BTFSS PORTA,CWM0
    BSF PORTB,DrvBF
    CALL delay
    CALL delay
    CALL delay
    CALL delay
    CALL delay
    CALL delay
    CALL delay
    CALL delay
    CALL delay
    CALL delay
    CALL delay
    CALL delay
    BTFSC PORTA,CWM0
    BSF PORTB,DrvBF
    CALL delay
    CALL delay

```

```

CALL delay
BTFSS PORTA,CWM0
BSF PORTB,DrvBF
CALL delay
BCF PORTB,DrvON
BCF PORTB,Obst ;signal obstacle
CALL delay
CALL delay
GOTO obst
noobst CALL delay
CALL delay
CALL delay
DECFSZ RLen
GOTO rloop
obst NOP

BCF PORTB,HStr
CALL delay
CALL delay
GOTO main

END

```

Blank Page

Chapter 4

4.1 Behavioural overview of Angel-1

Since the beginning of this project work our objective has been to apply NI2A2 theory for Angel-1 control, such that we have an existential proof of the applicability of NI2A2 to autonomous robotics. The senses of Angel-1 shall need to be interfaced to the NI2A2 model and the NI2A2 model's output shall provide the control words for Angel-1. To achieve this we need to define a formal language to represent the sensory input of Angel-1. Before that we shall develop a menu-driven "natural" user-interface for Angel-1

4.2 A basic interface engine for Angel-1

We have written a basic interface engine for Angel-1. The listing for this interface is given on page 88 and a screen shot is shown on page 87. The interface is intuitive. The only technical requirement is the knowledge of the port number to which Angel-1 is connected. For most cases it shall default to 378H. Let us move on and derive a formal specification language for Angel-1's speech input. Henceforth, we shall only provide pseudocode, as much of the implementation detail is system dependent, while both NI2A2 and Angel-1 are system independent. The source code and binary executable shall be provided on the accompanying CD-ROM.

4.3 Interfacing Angel-1's speech input to NI2A2

Interfacing Angel-1's speech input to NI2A2 is essentially an effort at NLP. Hence we shall first look at another possible method, that of syntactic analysis (using lexical analysers and parsers). For now we neglect the fact that the NI2A2 implementation can learn new words while the syntactic method cannot.

To implement NLP using syntactic analysis we first have to build a lexical analyser to generate tokens from the input and then build a parser to apply the tokens to the productions of a grammar. Given below are the specifications for lexical analysis and parsing. These may be directly mapped into input files for "lex" and "yacc", which then may be used to generate a compiler for the natural input of Angel-1.

Specification for lexical analysis:

Input	Token
gripper	command
steer	command
forward	attribute
backward	attribute
left	attribute
right	attribute
centre	attribute
on	attribute
off	attribute
close	attribute
open	attribute
sonar	command
walk	commit
run	commit

Specification for parsing:

Sentence := fullcmd
 | partcmd commit
 | partcmd Sentence

fullcmd := partcmd commit
 | commit

partcmd := command attribute

Now let us interface Angel-1's speech input to NI2A2. It is unnecessary to use the English alphabet as the language. If we consider Angel-1's mnemonics we find that being mutually exclusive they form a language in themselves. We assign values to the mnemonics as shown below

Mnemonic	Value
gripper	1
steer	2
forward	3
backward	4
left	5
right	6
centre	7
on	8
off	9
walk	10
run	11
sonar	12
close	13
open	14

Now a Godel number can be found for this language and used as input to the NI2A2 model. On comparing both the implementations of NLP we find the difference to be trivial. Hence, NI2A2 based NLP is at least as good as that based on syntactic analysis. Once trained the time complexity of NI2A2 is exponentially reduced, while no such benefit is obtained with the syntactic method. Besides NI2A2 can learn new words, and its language can be dynamically updated; for the syntactic method, though, extending the language requires complete redefinition of the grammar.

Implementations of both types of NLP are provided on the CD-ROM.

4.4 Interfacing Angel-1's vision to NI2A2

Computer vision is generally an implementation specific task. A given method, which works optimally for a problem, may not be very economical for another. We shall, therefore, develop two very generalised models of vision; both shall be used later in the project.

In the first, we divide the image into cells and average each cell into a few colours. These colours form the alphabets of the input language. For instance, if we divide the image into a 3x3 grid and average the cells into 3 colours we get a language of radix -3 with maximum sentence length of 9. This interface is impracticable for high resolutions.

The second interface identifies patterns. We use scan lines at regular interval vertically and horizontally on the averaged image to find zero-crossovers. Another set of scan lines generates the run-length of the initial colour. We then find the trends in these characteristics. These trends form the input language. For instance, the trends may be increasing, decreasing or static. A Godel number can now be generated.

4.5 Components of the NI2A2: Angel-1 system

Before we leave this discussion let us identify the components we have already designed for the NI2A2: Angel-1 system. These include NLP and vision. The remaining senses from the obstacle detector shall have to be heuristically accounted for during this project. This can be done through the governing function. Now let us move on the application of NI2A2 to industrial and domestic robotics, and a few other experiments.



Angell Basic Interface Engine

`Abhishek Choudhary
`SG-129546
`Basic Interface Engine for Angel-1
`Compiler: Visual Basic for MS-DOS

Version 1.00

```
BEGIN Form Form1
  AutoRedraw      = 0
  BackColor       = QBColor(7)
  BorderStyle     = 3
  Caption         = "ANGEL-1 Interface Engine"
  ControlBox     = -1
  Enabled         = -1
  ForeColor       = QBColor(0)
  Height          = Char(15)
  Left            = Char(9)
  MaxButton       = -1
  MinButton       = -1
  MousePointer    = 0
  Tag             = ""
  Top             = Char(3)
  Visible         = -1
  Width           = Char(63)
  WindowState     = 0
BEGIN Frame Frame2
  BackColor       = QBColor(7)
  Caption         = "Sonic Sensor"
  DragMode        = 0
  Enabled         = -1
  ForeColor       = QBColor(0)
  Height          = Char(4)
  Left            = Char(2)
  MousePointer    = 0
  TabIndex        = 1
  Tag             = ""
  Top             = Char(6)
  Visible         = -1
  Width           = Char(17)
BEGIN OptionButton Option3
  BackColor       = QBColor(7)
  Caption         = "Disable"
  DragMode        = 0
  Enabled         = -1
  ForeColor       = QBColor(0)
  Height          = Char(1)
  Index           = 0
  Left            = Char(1)
  MousePointer    = 0
  TabIndex        = 19
  TabStop         = -1
  Tag             = ""
  Top             = Char(1)
  Value           = -1
  Visible         = -1
  Width           = Char(12)
END
BEGIN OptionButton Option3
  BackColor       = QBColor(7)
  Caption         = "Enable"
```

```

        DragMode      = 0
        Enabled       = -1
        ForeColor     = QBColor(0)
        Height        = Char(1)
        Index         = 1
        Left          = Char(1)
        MousePointer  = 0
        TabIndex      = 20
        TabStop       = 0
        Tag           = ""
        Top           = Char(0)
        Value         = 0
        Visible       = -1
        Width         = Char(12)
    END
END
BEGIN Frame Frame3
    BackColor      = QBColor(7)
    Caption        = "Steer"
    DragMode       = 0
    Enabled        = -1
    ForeColor      = QBColor(0)
    Height         = Char(5)
    Left           = Char(22)
    MousePointer   = 0
    TabIndex       = 2
    Tag            = ""
    Top            = Char(1)
    Visible        = -1
    Width          = Char(18)
    BEGIN OptionButton Option2
        BackColor   = QBColor(7)
        Caption     = "Left"
        DragMode    = 0
        Enabled     = -1
        ForeColor   = QBColor(0)
        Height      = Char(1)
        Index       = 1
        Left        = Char(1)
        MousePointer = 0
        TabIndex    = 17
        TabStop     = 0
        Tag         = ""
        Top         = Char(0)
        Value       = 0
        Visible     = -1
        Width       = Char(14)
    END
    BEGIN OptionButton Option2
        BackColor   = QBColor(7)
        Caption     = "Right"
        DragMode    = 0
        Enabled     = -1
        ForeColor   = QBColor(0)
        Height      = Char(1)
        Index       = 2
        Left        = Char(1)
        MousePointer = 0
        TabIndex    = 18
    END

```

```

        TabStop      = 0
        Tag          = " "
        Top         = Char(1)
        Value       = 0
        Visible     = -1
        Width      = Char(14)
    END
    BEGIN OptionButton Option2
        BackColor   = QBColor(7)
        Caption     = "Center"
        DragMode    = 0
        Enabled     = -1
        ForeColor   = QBColor(0)
        Height     = Char(1)
        Index      = 0
        Left       = Char(1)
        MousePointer = 0
        TabIndex   = 16
        TabStop    = -1
        Tag        = " "
        Top       = Char(2)
        Value     = -1
        Visible   = -1
        Width    = Char(14)
    END
END
BEGIN Frame Frame5
    BackColor   = QBColor(7)
    Caption     = "Mode"
    DragMode    = 0
    Enabled     = -1
    ForeColor   = QBColor(0)
    Height     = Char(4)
    Left       = Char(42)
    MousePointer = 0
    TabIndex   = 4
    Tag        = " "
    Top       = Char(6)
    Visible   = -1
    Width    = Char(17)
    BEGIN OptionButton Option5
        BackColor   = QBColor(7)
        Caption     = "Walk"
        DragMode    = 0
        Enabled     = -1
        ForeColor   = QBColor(0)
        Height     = Char(1)
        Index      = 1
        Left       = Char(1)
        MousePointer = 0
        TabIndex   = 24
        TabStop    = 0
        Tag        = " "
        Top       = Char(1)
        Value     = 0
        Visible   = -1
        Width    = Char(13)
    END
END
BEGIN OptionButton Option5

```

```

        BackColor      = QBColor(7)
        Caption        = "Run"
        DragMode        = 0
        Enabled         = -1
        ForeColor       = QBColor(0)
        Height         = Char(1)
        Index           = 0
        Left            = Char(1)
        MousePointer    = 0
        TabIndex        = 23
        TabStop         = -1
        Tag             = ""
        Top             = Char(0)
        Value           = -1
        Visible         = -1
        Width           = Char(13)
    END
END
BEGIN CommandButton Command1
    BackColor      = QBColor(7)
    Cancel         = 0
    Caption        = "Commit"
    Default        = 0
    DragMode        = 0
    Enabled         = -1
    Height         = Char(3)
    Left           = Char(22)
    MousePointer    = 0
    TabIndex        = 5
    TabStop         = -1
    Tag             = ""
    Top            = Char(6)
    Visible         = -1
    Width           = Char(18)
END
BEGIN Timer Timer1
    Enabled         = -1
    Interval        = 1
    Left           = Char(52)
    Tag             = ""
    Top            = Char(10)
END
BEGIN Label Label1
    Alignment      = 0
    AutoSize       = 0
    BackColor      = QBColor(7)
    BorderStyle    = 0
    Caption        = "Control word:"
    DragMode        = 0
    Enabled         = -1
    ForeColor       = QBColor(0)
    Height         = Char(1)
    Left           = Char(22)
    MousePointer    = 0
    TabIndex        = 9
    Tag             = ""
    Top            = Char(9)
    Visible         = -1
    Width           = Char(13)

```

```

END
BEGIN Label Label2
  Alignment      = 0
  AutoSize      = 0
  BackColor     = QBColor(7)
  BorderStyle   = 0
  Caption       = "xxx"
  DragMode      = 0
  Enabled       = -1
  ForeColor     = QBColor(0)
  Height        = Char(1)
  Left          = Char(35)
  MousePointer  = 0
  TabIndex     = 10
  Tag           = ""
  Top           = Char(9)
  Visible      = -1
  Width        = Char(4)
END
BEGIN CommandButton Command2
  BackColor     = QBColor(7)
  Cancel        = 0
  Caption       = "Exit"
  Default       = 0
  DragMode      = 0
  Enabled       = -1
  Height        = Char(3)
  Left          = Char(44)
  MousePointer  = 0
  TabIndex     = 11
  TabStop       = -1
  Tag           = ""
  Top           = Char(10)
  Visible      = -1
  Width        = Char(14)
END
BEGIN Label Label3
  Alignment      = 0
  AutoSize      = 0
  BackColor     = QBColor(7)
  BorderStyle   = 0
  Caption       = "Port:"
  DragMode      = 0
  Enabled       = -1
  ForeColor     = QBColor(0)
  Height        = Char(1)
  Left          = Char(24)
  MousePointer  = 0
  TabIndex     = 12
  Tag           = ""
  Top           = Char(11)
  Visible      = -1
  Width        = Char(5)
END
BEGIN TextBox Text1
  BackColor     = QBColor(7)
  BorderStyle   = 1
  DragMode      = 0
  Enabled       = -1

```

```

        ForeColor      = QColor(0)
        Height         = Char(3)
        Left           = Char(29)
        MousePointer   = 0
        MultiLine      = 0
        ScrollBars     = 0
        TabIndex       = 13
        TabStop        = -1
        Tag            = ""
        Text           = "&h378"
        Top            = Char(10)
        Visible        = -1
        Width          = Char(8)
END
BEGIN Frame Frame1
    BackColor         = QColor(7)
    Caption           = "Gripper"
    DragMode          = 0
    Enabled           = -1
    ForeColor         = QColor(0)
    Height            = Char(4)
    Left              = Char(2)
    MousePointer      = 0
    TabIndex          = 0
    Tag               = ""
    Top               = Char(1)
    Visible           = -1
    Width             = Char(17)
    BEGIN OptionButton Option1
        BackColor     = QColor(7)
        Caption       = "Open"
        DragMode      = 0
        Enabled       = -1
        ForeColor     = QColor(0)
        Height        = Char(1)
        Index         = 0
        Left          = Char(1)
        MousePointer  = 0
        TabIndex      = 14
        TabStop       = 0
        Tag           = ""
        Top           = Char(0)
        Value         = 0
        Visible       = -1
        Width         = Char(14)
    END
    BEGIN OptionButton Option1
        BackColor     = QColor(7)
        Caption       = "Close"
        DragMode      = 0
        Enabled       = -1
        ForeColor     = QColor(0)
        Height        = Char(1)
        Index         = 1
        Left          = Char(1)
        MousePointer  = 0
        TabIndex      = 15
        TabStop       = -1
        Tag           = ""
    END

```

```

        Top          = Char(1)
        Value        = -1
        Visible      = -1
        Width        = Char(14)
    END
END
BEGIN CheckBox Check1
    BackColor      = QBColor(7)
    Caption        = "Busy"
    DragMode       = 0
    Enabled        = 0
    ForeColor      = QBColor(0)
    Height         = Char(1)
    Left          = Char(3)
    MousePointer   = 0
    TabIndex       = 6
    TabStop        = -1
    Tag            = ""
    Top           = Char(12)
    Value          = 0
    Visible        = -1
    Width         = Char(8)
END
BEGIN CheckBox Check3
    BackColor      = QBColor(7)
    Caption        = "Obstacle"
    DragMode       = 0
    Enabled        = 0
    ForeColor      = QBColor(0)
    Height         = Char(1)
    Left          = Char(3)
    MousePointer   = 0
    TabIndex       = 8
    TabStop        = -1
    Tag            = ""
    Top           = Char(11)
    Value          = 0
    Visible        = -1
    Width         = Char(13)
END
BEGIN CheckBox Check2
    BackColor      = QBColor(7)
    Caption        = "Parity err"
    DragMode       = 0
    Enabled        = 0
    ForeColor      = QBColor(0)
    Height         = Char(1)
    Left          = Char(3)
    MousePointer   = 0
    TabIndex       = 7
    TabStop        = -1
    Tag            = ""
    Top           = Char(10)
    Value          = 0
    Visible        = -1
    Width         = Char(15)
END
BEGIN Frame Frame4
    BackColor      = QBColor(7)

```

```

Caption      = "Direction"
DragMode     = 0
Enabled      = -1
ForeColor    = QBColor(0)
Height       = Char(4)
Left         = Char(42)
MousePointer = 0
TabIndex     = 3
Tag          = ""
Top          = Char(1)
Visible      = -1
Width        = Char(17)
BEGIN OptionButton Option4
  BackColor  = QBColor(7)
  Caption    = "Backward"
  DragMode   = 0
  Enabled    = -1
  ForeColor  = QBColor(0)
  Height     = Char(1)
  Index      = 0
  Left       = Char(1)
  MousePointer = 0
  TabIndex   = 21
  TabStop    = 0
  Tag        = ""
  Top        = Char(1)
  Value      = 0
  Visible    = -1
  Width      = Char(12)
END
BEGIN OptionButton Option4
  BackColor  = QBColor(7)
  Caption    = "Forward"
  DragMode   = 0
  Enabled    = -1
  ForeColor  = QBColor(0)
  Height     = Char(1)
  Index      = 1
  Left       = Char(1)
  MousePointer = 0
  TabIndex   = 22
  TabStop    = -1
  Tag        = ""
  Top        = Char(0)
  Value      = -1
  Visible    = -1
  Width      = Char(12)
END
END
END
DIM SHARED GrpVal AS INTEGER
DIM SHARED StrVal AS INTEGER
DIM SHARED SonVal AS INTEGER
DIM SHARED DirVal AS INTEGER
DIM SHARED ModVal AS INTEGER
DIM SHARED InpVal AS INTEGER
DIM SHARED OutVal AS INTEGER
DIM SHARED InpPort AS INTEGER
DIM SHARED OutPort AS INTEGER

```

```

SUB Command1_Click ()

    OutVal = 0: Parity = 0

    OutVal = OutVal + GrpVal * 2 ^ 0
    Parity = Parity + GrpVal

    OutVal = OutVal + DirVal * 2 ^ 3
    Parity = Parity + DirVal

    OutVal = OutVal + ModVal * 2 ^ 4
    Parity = Parity + ModVal

    OutVal = OutVal + SonVal * 2 ^ 5
    Parity = Parity + SonVal

    SELECT CASE (StrVal)
        CASE 0:
            OutVal = OutVal + 2 ^ 1 * 0
            OutVal = OutVal + 2 ^ 2 * 0
        CASE 1:
            OutVal = OutVal + 2 ^ 1 * 1
            OutVal = OutVal + 2 ^ 2 * 0
        CASE 2:
            OutVal = OutVal + 2 ^ 1 * 0
            OutVal = OutVal + 2 ^ 2 * 1
    END SELECT
    Parity = Parity + SGN(StrVal)

    OutVal = OutVal + ((Parity MOD 2) XOR 1) * 2 ^ 6
    Label2.Caption = STR$(OutVal)

    OUT OutPort, OutVal + 128
    t = TIMER
    WHILE TIMER - t < .01: WEND

END SUB

SUB Command2_Click ()

    OUT OutPort, 0
    END

END SUB

SUB Form_Load ()

    OutVal = 0
    OutVal = OutVal + GrpVal * 2 ^ 0
    OutVal = OutVal + DirVal * 2 ^ 3
    OutVal = OutVal + ModVal * 2 ^ 4
    OutVal = OutVal + SonVal * 2 ^ 5
    SELECT CASE (StrVal)
        CASE 0:
            OutVal = OutVal + 2 ^ 1 * 0
            OutVal = OutVal + 2 ^ 2 * 0
        CASE 1:
            OutVal = OutVal + 2 ^ 1 * 1

```

```

        OutVal = OutVal + 2 ^ 2 * 0
    CASE 2:
        OutVal = OutVal + 2 ^ 1 * 0
        OutVal = OutVal + 2 ^ 2 * 1
    END SELECT
    Label2.Caption = STR$(OutVal)
    OutPort = &H378
    InpPort = &H379
    OUT OutPort, 0

END SUB

SUB Option1_Click (Index AS INTEGER)

    GrpVal = Index

END SUB

SUB Option2_Click (Index AS INTEGER)

    StrVal = Index

END SUB

SUB Option3_Click (Index AS INTEGER)

    SonVal = Index

END SUB

SUB Option4_Click (Index AS INTEGER)

    DirVal = Index

END SUB

SUB Option5_Click (Index AS INTEGER)

    ModVal = Index

END SUB

SUB Text1_Change ()

    OutPort = VAL(Text1.Text)
    InpPort = VAL(Text1.Text) + 1
    Label2.Caption = STR$(OutPort)

END SUB

SUB Timer1_Timer ()

    OUT OutPort, OutVal
    InpVal = INP(InpPort)

    IF (InpVal AND 2 ^ 7) THEN
        Check1.Value = 1
    ELSE
        Check1.Value = 0
    END IF

END SUB

```

```
END IF
IF (InpVal AND 2 ^ 5) THEN
    Check3.Value = 0
ELSE
    Check3.Value = 1
END IF
IF (InpVal AND 2 ^ 3) THEN
    Check2.Value = 0
ELSE
    Check2.Value = 1
END IF
END SUB
```

Chapter 5

5.1 Overview of the experiments

We shall summarise here the experiments carried out to show the versatility of the NI2A2 system. Only the design and procedure of the experiments shall be mentioned. The actual implementation files have been included on the CD-ROM. Before moving on to the industrial and domestic applications let us consider a few simple but interesting experiments. We shall utilise the following design pattern for the experiments.

I. Specify objective

II. Specify method

III Specify formal language used for NI2A2 input

5.2 A colour following robot

Objective:

To design a robot, which shall follow a colour, it hears

Method:

First, an NI2A2 model is set up. We then identify the three basic colours - blue, green and red as the input language and train Angel-1 to identify those colours.

Specification language:

The specification language is $L = \{\text{Red, Green, Blue}\}$, mapped to $M = \{0,1,2\}$

5.3 A robotic watchman

Objective:

To design a robotic watchman

Method:

An NI2A2 model is set up. Motion detection is identified as the input state; i.e. motion or no motion. Angel-1 is trained to respond to motion.

Specification language:

The specification language is $L=\{\text{Motion, No motion}\}$ mapped to $M=\{0,1\}$

5.4 A traffic light obeying robot

Objective:

To make a robot which responds appropriately to the traffic lights.

Methods:

Similar to light following robot, only the alphabets change. The alphabets are identified as red, green and yellow - the colours of the traffic light. Angel-1 is trained to stop on red, run on green and slow-down on yellow. As the sequence is important the minimum word length is 2.

Specification language:

The specification language $L=\{\text{Red, Green, Yellow}\}$ mapped to $M=\{0,1,2\}$

5.5 A sign language understanding robot

Objective:

To make a robot understand gestures

Method:

The gestures are made with a shiny rod. The two alphabets are horizontal and vertical. Searching for the rod has been made trivial.

Specification language:

The specification language $L=\{\text{Horizontal, Vertical}\}$ mapped to $M=\{0,1\}$

5.6 Housekeeping robot

Objective:

To make angel behave as a housekeeper

Method:

Use various shapes to identify different rooms. Train Angel-1 to perform a few tasks. Get Angel-1 to identify proper rooms. Get Angel-1 to perform specific tasks in specific rooms as specified.

Specification language:

$L=\{\text{set of shapes and command}\}$

5.7 Industrial assistant robot

Objective:

To make an industrial assistant robot

Method:

Use shapes to identify different areas as above. Train Angel-1 for a few tasks. Get it to perform the tasks properly such as delivering goods into bins.

Specification language:

$L=\{\text{set of shapes and command}\}$

5.8 A model Vehicle Collision Prevention System (VCOPS) based on NI2A2

Objective:

To make a model VCOPS

Method:

Train Angel-1 to identify a road, oncoming traffic and distance. Form a language with these. Train Angel-1 to avoid collision situations.

Specification language:

$L = \{\text{Road, traffic, distance, etc.}\}$

The implementations for all the above experiments are included on the CD-ROM.

5.9 A clinical decision support system based on NI2A2

Clinical Decision Support can be broadly defined as the use of information to help a clinician diagnose or treat a patient's health problem. The field of medicine changes constantly. Each day there are headlines about new ways to diagnose medical conditions, breakthroughs in treatments, and newly discovered ways to prevent health problems. It is not possible for any person to keep track of these developments mentally; this is where Clinical Decision Support becomes helpful by providing suggestions and hints. A Health Monitoring System (HMS) with a learning algorithm that helps develop a knowledgebase automatically can prove a great boon for a country like India where rural healthcare is minimal.

Such a system can allow telemedicine to be administered in an effective way. This is how we propose it may take place. We shall define an extensive input language for an NI2A2 model (an HMS not a robot in this case) based on a classification of diseases, such as ICD-CM, and the model shall take as input a patients condition; during the first few cases it shall learn the Doctor's response, which should be part of the alphabet, and later make similar suggestions. NI2A2

would allow the doctor and patient to use natural language. Such health care "kiosks" may be set up in village at minimal cost.

Before we end this discussion, and now that we are nearing the end of this project, we would like to mention that justice could not be done to the above two topics of VCOPS and CDSS owing to time restraints for the project. Let us now move on to a small feasibility analysis. Finally we shall summarise the project.

Blank Page

Chapter 6

6.1 Feasibility analysis

The remaining scope of the project allows for a very brief feasibility analysis to be performed. We shall consider the metric called "Mean-time-between-assist" or MTBA, which is quite analogous to MTBF. MTBA is the arithmetic or stochastic mean of the time elapsed before the autonomous robot needs to be assisted after an initial human assistance. Mean Time To Assist or MTTA is an assistance interval after which the robot regains performance. Efficiency is obtained from MTBA using the following relation

$$\text{Efficiency} = \text{MTBA} * 100 / (\text{MTBA} + \text{MTTA})$$

In the experiments we have carried out we were able to obtain an efficiency of about 60%, however this result is based on a very small test data and requires further validation, after which it is most likely to improve.

Blank Page

Conclusion

Finally, we have reached the end of our humble effort at devising a means of taking technology closer to non-technical people. At the end of this project we have looked at the history of artificial intelligence and robotics, taken stock of their current position, developed a theory for designing autonomous robots which can be controlled using a natural interface, can be trained in unrelated domains, can have their behaviour altered dynamically and have vision capabilities, designed a protocol for robot-computer interaction, constructed a complete robotic platform, with a multiprocessor control system and an ultrasonic obstacle detector, written an interface engine for it, and have taught it to understand voice commands, follow colours and traffic rules, work as a watchman, do housekeeping, understand gestures and serve as an industrial assistant. We also have short proposals for a vehicle collision prevention system and a clinical decision support. However, our effort shall fail to be worthwhile if these do not find any real world application. We end this project here wishing that the topics that have been skimmed through in this project might be dealt with in depth in the future.

Blank Page

Appendix A

A short introduction to the PIC 16F84 Microcontroller

The PIC family of microcontroller units from Microchip Technology, Inc. is an upcoming industry standard. These wonderful devices pack a lot more of power in themselves than is obvious at a first glance. The Harvard architecture, coupled with a RISC design, makes it a tremendous performer. These have more features than what is generally expected of microcontrollers. Adding an icing on the top is the fact that Microchip releases MPLAB IDE, an environment for assembling and simulating PIC software, free of cost. It may be downloaded over the Internet from the Microchip website.

<http://www.microchip.com>

The PIC 16F84 is a midrange microcontroller. It offers two ports with a total of 13 I/O pins. The I/O functions of the individual pins may be selected from within the software. It provides a timer and a few interrupt sources, however there are no A/D or PWM pins. This is compensated by the fact that the PIC 16F84 may be interfaced with most common logic families. The wide electrical specification allows for high tolerance designs. With only 35 instructions, it is easy to learn and master for anyone familiar with microprocessor technology. The data-sheet may be downloaded from the manufacturers website. We are providing some excerpts from the data sheet here for quick reference.

Visit <http://www.microchip.com> for more information on
the PIC16F84 and other microcontrollers.

Visit <http://www.microchip.com> for more information on
the PIC16F84 and other microcontrollers.

Visit <http://www.microchip.com> for more information on
the PIC16F84 and other microcontrollers.

Visit <http://www.microchip.com> for more information on
the PIC16F84 and other microcontrollers.

Visit <http://www.microchip.com> for more information on
the PIC16F84 and other microcontrollers.

Visit <http://www.microchip.com> for more information on
the PIC16F84 and other microcontrollers.

Visit <http://www.microchip.com> for more information on
the PIC16F84 and other microcontrollers.

Reference

- [ARB95] Arbib, M. A. ed. (1995). *The Handbook of Brain Theory and Neural Networks*. MIT Press, Cambridge, M. A.
- [BAT00] Bates, M. (2000). *Introduction to Microelectronic Systems: The PIC 16F84 Microcontroller*. Arnold, London.
- [BRE97] Brey, B. B. (1997). *The Intel Microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, and Pentium Pro Processor : Architecture, Programming, and Interfacing*. 4th Edition. Prentice-Hall, New Delhi.
- [CAR00] Carpenter, M. B. *Human Neuro-anatomy*. William and Winkins, Baltimore.
- [CAR90] Carpenter, R. H, S. (1990). *Neurophysiology*. Edward Arnold, London.
- [CAR01] Carr, J. J. and Brown J. M. (2001). *Introduction to Biomedical Equipment Technology*. 4th Edition. Prentice-Hall, New Jersey.
- [COM95] Comer, D. E. (1995). *Internetworking with TCP/IP-Volume 1 : Principles, Protocols, and Architecture*. 3rd Edition. Prentice-Hall, N. J.
- [CRI72] Critchley, M., O'heary, J. L. and Jennet, B. eds. (1972). *Scientific Foundation of Neurology*. Hienmann, London.
- [DHA99] Dhamdhare, D. M. (1999). *Systems Programming and Operating Systems*. 2nd Revised Edition. Tata McGraw-Hill, New Delhi.
- [DIC91] Dick, R. S. and Steen, E. B. eds. (1991). *The Computer-Based Patient Record : An essential Technology for Healthcare*. National Academy Press, Washington DC.
- [FLA98] Flanagan, D. (1998). *Javascript : The Definitive Guide*. O'Reilly, CA.
- [FU87] Fu, K. S., Gonzalez, R. C. and Lee, C. S. G. (1987). *Robotics : Control, Sensing, Vision, and Intelligence*. McGraw-Hill, Singapore.
- [GAO00] Gaonkar, R. S. (2000). *Microprocessor Architecture, Programming, and Applications with the 8085*. 4th Edition. Penram International Publishing, Mumbai.
- [GAR91] Gardner, R. M. et al. (1991). *Real Time Data Acquisition : Experience with the Medical Information Bus (MIB)*. Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care, Vol. 15, pp. 813-817.

- [GOL00] Goldberg, D. E. (2000). *Genetic Algorithms in Search, Optimization & Machine Learning*. Pearson Education, Bangalore.
- [GOU74] Gould, P. and White, R. E. (1974). *Mental Maps*. Penguin, Harrowdsworth.
- [HAY89] Hayes, T. C. and Horowitz, P. (1989). *The Art of Electronics (Student Manual)*. University Press, Cambridge.
- [HOR89] Horowitz, P. and Hill, W. (1989). *The Art of Electronics*. 2nd Edition. University Press, Cambridge.
- [HOR98] Horowitz, E., Sahni, S. and Rajasekaran, S. (1998). *Fundamentals of Computer Algorithms*. W. H. Freeman & Co. (Galgotia, New Delhi).
- [HUD00] Hudak, P. (2000). *The Haskell School of Expression : Learning Functional Programming through Multimedia*. Cambridge University Press, Cambridge.
- [LAN95] Langton, C. G. ed. (1995). *Artificial Life : An Overview*. MIT Press, Cambridge M.A.
- [LAT98] Lathi, B. P. (1998). *Modern Digital and Analog Communication Systems*. 3rd Edition. Oxford University Press, New York.
- [LUG02] Luger, G. F. (2002). *Artificial Intelligence : Structures and Strategies for Complex Problem Solving*. 4th Edition. Pearson Education Ltd., Delhi
- [MAN93] Mano M. M. (1993). *Computer System Architecture*. 3rd Edition. Prentice-Hall, N. J. (New Delhi).
- [MIL87] Millman, J. and Grabel, A. (1987). *Microelectronics*. 2nd Edition. McGraw-Hill, Singapore.
- [MIT00] *MIT Encyclopedia of Cognitive Science*. <http://www.mit.edu>
- [MOS01] Moss, S. and Davidson, P. eds. (2001). *Multi-Agent-Based Simulation*. 2nd International Workshop : Revised and Additional Papers / MABS 2000, Boston, MA, USA, July. Springer-Verlag, Berlin.
- [MOU99] Moulton, C. and Yates, D. (1999). *Emergency Medicine*. 2nd Edition. Blackwell Science, London.
- [MUK94] Mukherjee, J. B. (1994). *Forensic Medicine and Toxicology*. 2nd Edition. Arnold Associates, New Delhi.
- [NEH00] Nehmzow, U. (2000). *Mobile Robotics : A Practical Introduction*. Springer-Verlag, London.

- [NEL96] Nelson, M. and Gailly, J. -L. (1996). *The Data Compression Book*. 2nd Edition. M & T Books, U.S.A. (BPB Publications, New Delhi).
- [NIE00] Nielsen, M. A. and Chuang, I. L. (2000). *Quantum Computation and Quantum Information*. University Press, Cambridge.
- [NOO00] Noorgaard, M., Ravn, O., Poulsen, N. K., and Hansen, L. K. (2000). *Neural Networks for Modelling and Control of Dynamic Systems: A Practitioner's Handbook (Advanced textbooks in control and signal processing)*. Springer-Verlag, London.
- [OPP97] Oppenheim, A. V., Willsky, A. S. and Nawab, S. H. (1997). *Signals & Systems*. 2nd Edition. Prentice-Hall, New Delhi.
- [PIA91] Piatetsky-Shapiro, G. and Frawley, W. J. eds. (1991). *Knowledge Discovery in Databases*. The MIT Press, Menlo Park, CA.
- [PRE92] Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992). *Numerical Recipes in C : The Art of Scientific Computing*. 2nd Edition. Cambridge University Press.
- [RAM00] Ramakrishnan, R. and Gehrke, J. (2000). *Database Management Systems*. 2nd Edition. McGraw-Hill, Singapore.
- [RED94] Reddy, K. R., Badami, S. B. and Balasubramanian, V. (1994). *Oscillations and Waves*. Universities Press, Hyderabad.
- [RIC91] Rich, E. and Knight, K. (1991). *Artificial Intelligence*. 2nd Edition. McGraw-Hill, Inc., New York.
- [ROM97] Romp, G. (1997). *Game Theory : Introduction and Applications*. Oxford University Press, New York.
- [SHE97] Shetty, D. and Kolk, R. A. (1997). *Mechatronics System Design*. Thomson Asia, Singapore.
- [STA00] Stallings, W. (2000). *Data & Computer Communications*. 6th Edition. Pearson Education, Inc. (A. W. Longman, Delhi).
- [SUD99] Sudha, G. F. (1999). *Digital Stethoscope*. IETE Journal of Education, Vol. 40 nos. 3 & 4, Dec.,pp. 83-86.
- [SWA02] Swash, M. ed. (2002). *Hutchison's Clinical Methods*. 21st Edition. W. B. Saunders, Edinburgh.
- [TOM95] Tompkins, W. J. ed. (1995). *Biomedical Digital Signal Processing*. Prentice-Hall, New Jersey.

[VAL74] Valkenburg, M. E. V. (1974). *Network Analysis*. 3rd Edition. Prentice-Hall, N. J. (New Delhi).

[WEB74] Webos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD Thesis. Harvard University, Cambridge, MA

[WEI48] Weiner, N. (1948). *Cybernetics*. John Wiley and Sons, New York.

[WHE01] Whelan, P. F. and Molloy, D. (2001). *Machine Vision Algorithms in Java : Techniques and Implementation*. Springer-Verlag, London.