The Development of a Prototype Robotic Platform on which Awareness Strategies can be Implemented

Sean Kinsella
Department of Applied Physics and Instrumentation, Cork Institute of Technology, Cork, Ireland

Follow this and additional works at: https://sword.cit.ie/allthe

Part of the Robotics Commons

Recommended Citation
Kinsella, Sean, "The Development of a Prototype Robotic Platform on which Awareness Strategies can be Implemented" (2009). Theses [online]. Available at: https://sword.cit.ie/allthe/199

This Master Thesis is brought to you for free and open access by the Dissertations and Theses at SWORD - South West Open Research Deposit. It has been accepted for inclusion in Theses by an authorized administrator of SWORD - South West Open Research Deposit. For more information, please contact sword@cit.ie.
The Development of a Prototype Robotic Platform on which Awareness Strategies can be Implemented

Seán Kinsella

Cork Institute of Technology

M.Sc. 2009
The Development of a Prototype Robotic Platform on which Awareness Strategies can be Implemented

Mr. Seán Kinsella
September 2009

Submitted to Cork Institute of Technology in fulfilment of the requirements for the Degree of Master of Science by Research

Supervised by Mr. Harvey Makin CEng MSc MInstMC
Abstract

This thesis describes the design and development of the hardware and software for a prototype robotic platform on which awareness strategies can be developed. The mobile prototype platform designed over the course of work is capable of operating as a distributed or centralised system.

There is no singular definition of awareness for a robotic system and the development of awareness strategies is complex and requires expertise in many scientific disciplines. This paper describes the different types of awareness associated with robotic systems and the technology required to implement this awareness. A review of sensors critical to the development of awareness is conducted and their most important characteristics are defined.

The prototype platform developed in this work is controlled by a Motorola Dragonball™ VZ processor which runs the µCLinux operating system. Ultrasonic, infrared and compass sensors were integrated to the robotic platform. These sensors were selected to provide the platform with spatial and positional awareness. A flame detector and a light detector were integrated into the system to demonstrate environmental awareness capabilities. The choice of hardware and software ensures that the system is reliable, robust and easily expanded. The device driver software is written in C and supports sensors with pulse width modulated (PWM) outputs, proportional pulse width outputs, digital outputs and logic high-low outputs. Device driver software was also developed to support motor direction and speed control. The system is connected via Bluetooth™ to a desktop computer where a GUI displays a representation of the robot and its position in the environment.

Testing and experimentation shows that the prototype robotic system is suitable for the development of awareness strategies.
Acknowledgements

First and foremost I would like to thank: my research supervisor Mr. Harvey Makin for his guidance and commitment to this project; Dr. Liam McDonnell, Head of the Department of Applied Physics & Instrumentation, without whose perseverance, advice and crucial contribution this work would not have been completed. Harvey and Liam, I am grateful in every possible way.

I would like to thank the members of the post-graduate team in the department (Anthony, Conor, Michael, David) for support, advice and friendship and all of the staff at Cork Institute of Technology who assisted me in every way possible during my time there.

Furthermore, I am particularly grateful to Cork Institute of Technology for financial support of my research scholarship.

Finally, I would like to extend a very sincere thank you to my family (Pat, Mairead, Trisha) and Julia for their support and encouragement, without which this work would not have been possible.
Declaration
I hereby declare that, except where otherwise indicated, this document is entirely my own work and has not been submitted in whole or in part to any other university.
Table of Contents

ABSTRACT ................................................................................................................................. 2
ACKNOWLEDGEMENTS .............................................................................................................. 3
DECLARATION ............................................................................................................................ 4

CHAPTER 1 ROBOTS AND AWARENESS .................................................................................... 13

1.1 AWARENESS IN ROBOTS – INTRODUCTION ................................................................. 13
1.2 TYPES OF AWARENESS ..................................................................................................... 13
1.3 RESEARCH IN THE FIELD OF AWARENESS ................................................................. 16
1.3.1 RESEARCH CATEGORIES ............................................................................................. 17
1.4 AWARENESS STRATEGIES .............................................................................................. 21
1.4.1 FRONTIER BASED EXPLORATION ............................................................................ 23
1.4.2 POSITIONAL AWARENESS AND MAPPING .............................................................. 25
1.4.3 POSITIONAL AWARENESS AND NAVIGATION ......................................................... 30
1.4.4 POSITIONAL AWARENESS AND PATH PLANNING ................................................ 34
1.4.5 BEHAVIOUR BASED AWARENESS .......................................................................... 37
1.4.6 AWARENESS AND THE ANCHORING PROBLEM ..................................................... 38
1.4.7 HUMAN – ROBOT INTERACTION AND SITUATIONAL AWARENESS ...................... 41
1.4.8 IMPACT OF SENSOR POSITIONING ON AWARENESS ............................................ 44
1.5 DESIGN FOR RELIABILITY ............................................................................................... 46
1.5.1 DESIGNING RELIABLE HARDWARE ......................................................................... 47
1.5.2 DESIGNING RELIABLE SOFTWARE ........................................................................... 49
1.6 PREVENTING HARM TO HUMANS AND SITUATIONAL AWARENESS ....................... 52
1.7 CONCLUDING REMARKS .................................................................................................... 53

CHAPTER 2 SENSORS .................................................................................................................. 54

2.1 SENSORS FOR INTERNAL FEEDBACK ............................................................................ 54
2.1.1 VOLTAGE AND CURRENT SENSORS ........................................................................ 54
2.1.2 ACCELEROMETERS .................................................................................................... 55
2.1.3 LINEAR VARIABLE DIFFERENTIAL TRANSFORMERS (LVDT) .................................. 56
2.1.4 TEMPERATURE ............................................................................................................ 57
2.1.5 ENCODERS ................................................................................................................ 60
2.1.6 FORCE, PRESSURE AND TORQUE TRANSDUCERS ................................................ 62
2.2 SENSORS FOR EXTERNAL INTERACTION ...................................................................... 65
2.2.1 PROXIMITY DETECTION AND ULTRASONIC RANGERS .......................................... 65
2.2.2 OTHER NON-CONTACT PROXIMITY SENSORS AND RANGERS ............................ 73
2.2.3 TACTILE SENSORS ................................................................................................... 77
2.2.4 POSITION SENSORS ................................................................................................... 78
2.2.5 LIGHT SENSORS ........................................................................................................ 81
2.2.6 SOUND SENSORS ..................................................................................................... 82
2.2.7 VISION SENSORS ...................................................................................................... 83
2.2.8 ENVIRONMENTAL SENSORS .................................................................................... 85
2.3 IMPORTANT SENSOR CHARACTERISTICS ................................................................ 95
2.3.1 RESOLUTION .............................................................................................................. 95
2.3.2 SENSITIVITY .............................................................................................................. 95
2.3.3 LINEARITY ................................................................................................................ 95
2.3.4 RANGE ........................................................................................................................ 96

©
List of Tables

TABLE 1.1 AWARENESS DEFINITIONS ......................................................................................................... 14
TABLE 1.2 LEVELS OF AWARENESS BASED ON YOUNG'S SCALE .......................................................... 15
TABLE 1.3 FRONTIER BASED EXPLORATION CLASSIFICATION CRITERIA ............................................... 24
TABLE 2.1 MODERN ACCELEROMETER SENSORS .................................................................................... 56
TABLE 2.2 MODERN LVDT SENSORS ......................................................................................................... 57
TABLE 2.3 TEMPERATURE SENSORS .......................................................................................................... 59
TABLE 2.4 THERMAL IMAGING SENSORS ................................................................................................... 60
TABLE 2.5 MODERN ENCODER SENSORS .................................................................................................. 61
TABLE 2.6 MODERN FORCE/TORQUE SENSORS ......................................................................................... 62
TABLE 2.7 MODERN LASER TOF RANGING SYSTEMS ............................................................................... 76
TABLE 2.8 MODERN TOUCH SENSORS ..................................................................................................... 78
TABLE 2.9 MODERN POSITION SENSORS ................................................................................................ 81
TABLE 2.10 MODERN LIGHT SENSORS ..................................................................................................... 82
TABLE 2.11 MODERN SOUND SENSORS .................................................................................................... 82
TABLE 2.12 CCD VS. CMOS COMPARISON ............................................................................................... 84
TABLE 2.13 MODERN ENVIRONMENTAL SENSORS .................................................................................. 94
TABLE 3.1 COMMUNICATIONS INTERFACE OPTIONS ............................................................................... 109
TABLE 6.1 STANDARD ERROR CALCULATION BASED ON RECORDED VALUES ...................................... 170
TABLE 6.2 STANDARD DEVIATION AND STANDARD ERROR CALCULATIONS FOR SHARP IR SENSOR .... 180
List of Figures

FIGURE 1-1 IROBOT, AUTOMATED HARVESTER AND A TACTICAL SURVEILLANCE ROBOT...18
FIGURE 1-2 SONY ENTERTAINMENT ROBOTS Q RIO AND AIBO AND ASIMO THE HONDA HUMANOID...19
FIGURE 1-3 THE MARS EXPLORATION ROVER AS SHOWN ON THE NASA WEBSITE...20
FIGURE 1-4 A SIMPLE EXAMPLE OF FRONTIER EDGE DETECTION. THE ROBOTS LOCATION IS MARKED WITH A R. OCCUPIED CELLS ARE BLACK, UNOCCUPIED CELLS ARE WHITE AND UNKNOWN CELLS ARE GREY; CELLS MARKED WITH X ARE FRONTIER EDGE CELLS [33]...24
FIGURE 1-5 AN EVIDENCE GRID BUILT BY A REAL ROBOT IN A HALLWAY ADJACENT TO TWO OPEN DOORS [33]...24
FIGURE 1-6 GLOBAL MAPS OBTAINED USING SONAR (A) AND LASER (B) [36]...26
FIGURE 1-7 THE PATH TAKEN BY A SAHARAN ANT CATAGLYPHIS WHEN SEARCHING FOR FOOD. THE LARGE BLACK CIRCLE SIGNIFIES THE POINT WHERE THE FOOD IS FOUND; THE WHITE CIRCLE SIGNIFIES THE POINT WHERE THE ANTS NEST IS LOCATED...33
FIGURE 1-8 PATH PLANNING IN (A) REGULAR GRID REPRESENTATION, (B) QUADTREE REPRESENTATION AND (C) FRAMED-QUADTREE REPRESENTATION [43]...36
FIGURE 1-9 A SUCCESSFUL LONG TRAVERSE USING A FRAMED-QUAD-TREE DP ALGORITHM ON AN AUTONOMOUS JEEP [45]...37
FIGURE 1-10 DYNAMIC CAPABILITIES OF ANCHORING FRAMEWORK [43]...40
FIGURE 1-11 STEPS INVOLVED IN TRANSFERRING A "CHUNK" OF SPOKEN DATA INTO A ROBOT ACTION. [49]...42
FIGURE 1-12 EXAMPLE OF A FAULT TREE [66]...48
FIGURE 1-13 FEATURE DRIVEN DEVELOPMENT FRAMEWORK [67]...50
FIGURE 2-1 CONSTRUCTION OF LVDT [74]...57
FIGURE 2-2 THERMAL IMAGE OF A HUMAN...60
FIGURE 2-3 TYPICAL SINGLE CHANNEL ENCODER AND ABSOLUTE ENCODER [80]...61
FIGURE 2-4 A TYPICAL STRAIN GAUGE, ELECTRICAL RESISTANCE CHANGES AS FORCE IS EXERTED ON THE GAUGE AND A TYPICAL QUARTER BRIDGE STRAIN GAUGE CIRCUIT...63
FIGURE 2-5 A PRESSURE TRANSDUCER MEASURING A FLUID UNDER PRESSURE. AS THE PRESSURE DECREASES THE DIAPHRAGM WILL DEFLECT CHANGING THE RESISTANCE...63
FIGURE 2-6 A ROTARY TRANSFORMER TYPICAL CIRCUIT DIAGRAM AND BLOCK DIAGRAM...64
FIGURE 2-7 ULTRASONIC SENSOR OPERATION AND DEVANTECH SONAR RANGER [81]...66
FIGURE 2-8 SENSCOMP TRANSDUCER RANGE (LEFT TO RIGHT), 9000 SERIES, 600 SERIES AND 7000 SERIES [83]...68
FIGURE 2-9 SENSCOMP 6500 RANGING MODULE...68
FIGURE 2-10 TIMING DIAGRAMS FOR THE 6500-SERIES RANGING MODULE EXECUTING A MULTIPLE-ECHO-MODE CYCLE WITH BLANKING INPUT [85]...70
FIGURE 2-11 ROBART II [86] AND ROBART III [87]...71
FIGURE 2-12 DEVANTECH SRF04 ULTRASONIC RANGER...72
FIGURE 2-13 DEVANTECH SRF08 ULTRASONIC RANGER...73
FIGURE 2-14 SICK INC. LMS200 (LEFT) AND LMS220 (RIGHT), THE ONLY DIFFERENCE BETWEEN THE TWO SENSORS APART FROM THE PHYSICAL DIMENSIONS IS THE ENCLOSURE RATING. THE LMS200 HAS AN ENCLOSURE RATING OF IP65, WHILE THE LMS220 HAS AN ENCLOSURE RATING OF IP67 [89]...74
FIGURE 2-15 OPERATING PRINCIPLE OF LMS200 AND LMS220 [89]...75
FIGURE 2-16 DISTANCE COMPARISON CHART FOR SHARP RANGE OF IR SENSORS [85]...76
FIGURE 2-17 COMPARISON TABLE FOR SHARP RANGE OF IR SENSORS...76
FIGURE 2-18 DEVANTECH COMPASS SENSOR AND CALIBRATION CIRCUIT...79
FIGURE 2-19 GARMIN GPS 16/17/18 AND INTERFACE CIRCUIT...80

...
FIGURE 7-1 [A] ROBOTIC PROTOTYPE PLATFORM ON NIKKON REMOTE CONTROL CAR [B] ON MAX99 CHASSIS.
Chapter 1 Robots and awareness

This chapter introduces the concept of awareness in regard to robots and their operation. Firstly the different types of robotic awareness are reviewed. Secondly, the latest research in the field of robotic awareness is reviewed. This is followed by a discussion on strategies for design that provide high reliability. Finally methodologies that prevent robots from harming humans are described.

1.1 Awareness in robots – Introduction

To use a general definition, awareness is the understanding that objects have of each other in a shared environment. This section will discuss the different types of awareness associated with robots and the different elements of awareness that a robot must have to gain a better understanding of objects in its shared environment.

The development of awareness strategies is a complex problem to solve that requires expertise in many scientific disciplines. For a robotic system to "be aware", it must be equipped with sufficient sensors so that it can extract relevant data from its environment, and it must be equipped with enough intelligence to utilise this data. When we consider the multiple disciplines involved in making an autonomous robot one should realize how broad the topic is. We discuss here the important components of a fully aware system.

1.2 Types of awareness

There is no singular definition for awareness associated with robotics but there are many different types of awareness that one can implement on a robot. Unmanned autonomous systems for example must be mission aware. Mission awareness is a combination of the knowledge of mission objectives, internal self-situational awareness and external self-...
situational awareness [1]. Robots which are designed to interact with humans such as robots used in education, for rehabilitation or for entertainment must have social awareness [2]. Automated manipulators must be self-aware but must show robot-human awareness in order to prevent injury or death.

Scholtz et al. [3] provides a set of definitions which describes the types of awareness that humans have of robotic activities and the knowledge that robots have of the commands given to them by humans. This takes into account the cognitive element which a human will contribute when working with a robot and develops a framework around this. The key definitions of awareness which are outlined in [3] are shown in Table 1.1.

<table>
<thead>
<tr>
<th>Awareness Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awareness</td>
<td>An understanding of the activities of others, which provides a context for your own activities [4].</td>
</tr>
<tr>
<td>Concept/Task awareness</td>
<td>The participants' understanding of how their task will be completed [5].</td>
</tr>
<tr>
<td>Group-structural awareness</td>
<td>Knowledge about such things as people's roles and responsibilities, their position on an issue, their status and group processes [6].</td>
</tr>
<tr>
<td>Situational awareness</td>
<td>The perception of the elements in the environment within a volume of time, space, the comprehension of their meaning and the projection of their status in the near future [7].</td>
</tr>
<tr>
<td>Informal awareness</td>
<td>The general sense of who is around and what others are up to [6].</td>
</tr>
<tr>
<td>Social awareness</td>
<td>Information about the activities of people in a shared environment [8].</td>
</tr>
</tbody>
</table>

Table 1.1 Awareness definitions

Young [9] devised a scale which defines levels of awareness associated with humans in order of increasing complexity and this too can be applied to robots. Table 1.2 shows Young's scale in terms of its robotic equivalent.
<table>
<thead>
<tr>
<th>Level of Awareness</th>
<th>Term</th>
<th>Definition</th>
<th>Robotic Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Phenomenal</td>
<td>The experience of seeing, hearing, touching.</td>
<td>Sensory Inputs – Vision Cameras, Tactile Sensors, Microphones.</td>
</tr>
<tr>
<td>2</td>
<td>Access</td>
<td>Where stored information is brought to mind, such as when recognised an object or face, remembering when something has happened in the past.</td>
<td>Memory, Databases, complex algorithms.</td>
</tr>
<tr>
<td>3</td>
<td>Monitoring</td>
<td>Awareness of our own actions and their effects and monitoring perceptual information for discrepancies with current plans and hypothesis.</td>
<td>Complex Cognitive Algorithms.</td>
</tr>
<tr>
<td>4</td>
<td>Executive</td>
<td>Awareness of our goals and intentions</td>
<td>Combination of all types of awareness implemented in hardware and software – humanoid robot.</td>
</tr>
</tbody>
</table>

Table 1.2 Levels of awareness based on Young’s scale

Situational awareness as defined in Table 1.1 has been used as the basis for tactical-level reasoning in the creation of a navigation planner for an intelligent level vehicle by Rahul Sukthankar [10].

Cognitive psychologists believe that the tactical driving task which is taken for granted by humans is an extremely challenging information processing task where attention must be divided between the accelerator, the windscreen, the wing mirrors and the rear view mirror amongst other things in an environment which is constantly changing – other vehicles can over-take, turn off the road, turn onto the road, there can be cyclists or pedestrians on the road. Sukthankar attributes the expertise shown in tactical driving by humans to situational awareness whereby they can successfully decide on and execute maneuvers in real-time based on incomplete information about an environment which is rapidly changing. With the use of simulators he demonstrates that it is possible to safely control a vehicle using situational awareness as the basis for its algorithms.
Overall system health is a form of robotic self-awareness and Karl M. Reichard [11] has simulated the use of multiple intelligent sensor nodes to provide the robotic system with information regarding the health of its system components. The next section describes the research currently being undertaken in the field of robotics.

1.3 Research in the field of awareness

The need to execute repetitive or dangerous tasks has increased the need for robots to take the place of humans; at present there are a number of areas where robots are being used successfully to complete these tasks and with the success these robots are having in executing these tasks comes the want and need for further research.

The ultimate goal of every robotocist be it working for a company or as part of a research team at a university is to create an autonomous robot that is fully aware of its surroundings. In the past, research in the area of robotics was carried out predominately in the USA and Japan due to their significant research and development budgets, with Japan embracing the technology more readily. The latter being a key factor in their emergence as a global manufacturing power after the Second World War. Since the formation of the European Robotics Research Network [12] in 2000, robotics related research has grown in Europe; with Germany in particular to the fore with the overall goal of the European Robotics Research Network being a humanoid robot.

Awareness requires extensive arrays of sensors, powerful processing hardware and complex software algorithms and the key areas where this research is being carried out is discussed in the next section.
1.3.1 Research categories

Robotics can be divided rather crudely into the categories of service robots, security/defence robots, search and rescue robots, industrial robots, space robots and R&D/academic robots. Each one of these robots has a particular criterion which they must satisfy but they all must have an awareness of their surroundings in order to be autonomous.

Service robots are used for many daily tasks such as cleaning/housekeeping, agriculture/harvesting, pick and place and in the medical industry. There is currently a large amount of research ongoing in this sector and is mainly driven by industry which is trying to gain a dominant global position. The South Korean government announced in April 2009 that it is to invest $US750 million in its robotics industry in an attempt to accelerate economic growth. This investment is to concentrate mainly on the service robotics sector where they are expecting to be the global leader in 2018 [13].

The National Robotics Engineering Centre at Carnegie Mellon University [14] has developed a number of service robots that demonstrate awareness, e.g. their robotic harvester project [15] which has been designed to autonomously harvest a field with the use of vision perception and a global positioning system (GPS) receiver for navigation. They have also developed an automated dumping system [16] that uses laser detection and ranging (LADAR) sensors to locate the soil and the dump truck so as to automate the task of excavating material onto dump trucks.

The iRobot company [17], a $US307 million company founded by MIT roboticists Colin Angle and Helen Greiner currently leads the personal robots sector with their wide array of
robots which are designed to clean floors, swimming pools and gutters. It also comes with a development kit which encourages the development of new robotics applications.

Figure 1-1 iRobot, automated harvester and a tactical surveillance robot.

HONDA developed their first “walking” robot in 1986 with the goal of developing a robot capable of moving in any environment and over any terrain and by the year 2000 they had developed a robot whose name is an acronym for Advanced step in innovative mobility (ASIMO) [18].

The developers of ASIMO were predominately concerned with developing walking and balance techniques rather than developing Artificial Intelligence.

Sony Corporation is currently leading the way with their innovation for artificially intelligent humanoid robots. In 1993 they assembled a team of engineers to develop an “Artificially Intelligent Entertainment Robot (AIBO)” and they released the first generation AIBO in 1999. Sony are actively involved in RoboCup [19], an international research and education initiative designed with the goal of producing an artificially intelligent robot capable of playing soccer against the world's top players by 2050 and their most recent entertainment humanoid robot QRIO [20] is far ahead of its competitors (Figure 1-2).
Space robotics presents a unique challenge to engineers as they must consider zero gravity, radiation, the vacuum and extreme temperature swings within their designs. The German Aerospace Centre (DLR) [21], the Japan Aerospace Exploration Agency (JAXA) [22] and the Jet propulsion laboratory (JPL) at NASA [23] have teams of engineers working on designs for maintenance robots and researching new ways of exploring planets such as Mars and Europa and the other moons surrounding Jupiter. The exploration robots possess a multitude of sensors and scientific instruments in order to gather information about their environment. These instruments form two classes: direct sensing instruments which interact with the immediate vicinity of the robot such as radiation detection instruments; and remote-sensing instruments such as thermal imaging instruments [24]. These classes of instrument both provide the Mars rovers with an environmental awareness.

The Mars rovers, Spirit and Opportunity, which are part of NASA’s Mars exploration program landed in January 2004 to search for and characterise rocks and soils that hold clues to previous water activity on the planet [25]. These rovers were equipped with a panoramic camera to determine the structure of the local terrain, miniature thermal emission spectrometers to provide thermal profiles of the Martian atmosphere, microscopic
imagers for obtaining high resolution images of the rocks and soils and a rock abrasion tool to expose the inside of rocks. It was originally hoped that the rovers would travel a distance of 1km. However this was greatly exceeded with Spirit having travelled 7.7km and Opportunity having travelled 15.8km.

Future missions include an environmental survey of Mars using an autonomous powered airplane to capture images of the Martian landscape [26], sub surface exploration to search for water and information on the possibility of life on Mars and a mission to return samples of Martian rocks to Earth for laboratory analysis.

Figure 1-3 The Mars exploration rover as shown on the NASA website

One of the longest active research groups involved in the area of robotics and artificial intelligence (AI) is at the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. Founded in 1959, and currently involving 200 members the AI Lab at MIT believes that by studying vision, robotics and language they will begin to understand the awareness a robot must have to demonstrate intelligence. MIT currently has groups researching 47 different topics related to AI and robotics, ranging from the development of an efficient general purpose processor (Project Aries) to the building of an AI robot, (MARS project).

The iRobot company [17] is pioneering the research into security and defence robotics with the help of the US military which is funding a $US30million contract for their unmanned
ground vehicle (UGV) design which they use for mine disposal and reconnaissance missions.

Medical robots are perhaps the most advanced and aware non-autonomous robots developed at this moment in time. They are predominately used for surgery so there is no room for error therefore they must demonstrate awareness across a wide spectrum. Internal sensors must provide self-awareness whilst the sensors for human-robot/robot-human interaction must be precise. The most advanced medical robot is the Da Vinci surgical system from Intuitive Surgical [27] and is used predominately for minimally invasive surgery. It consists of 4 robotic arms that are used to make an incision in the abdomen into which rods are inserted; one of the rods contains two endoscopic cameras which are capable of providing a stereoscopic image whilst the other 3 rods contain surgical instruments. This precision robot has highly sensitive sensors and feedback systems which provides information to the surgeon in real-time and was originally designed by the US Defence forces and NASA.

1.4 Awareness strategies

There are very different levels of awareness available to robots in today’s world. Robot manipulators used in some manufacturing facilities may have little or no awareness of their surroundings. They are programmed to go to specific x-y-z co-ordinates and execute tasks such as removing plastic components from injection moulding machines, putting caps on soft drink bottles and placing components on a printed circuit board (PCB) during assembly. These robots will do exactly what the human programs them to do regardless of what is in the way. Thus a number of safety features such as fence beams, or protective guards must be in place when these robots are in operation. Advancements in sensor
technologies have reduced costs and allowed robot manufacturers to place small proximity sensors on the limbs of industrial robots, allowing them to detect foreign objects in their vicinity preventing damage to the robot itself and the foreign object. The placement of these sensors gives the robot a crude awareness of aspects of its surroundings.

At the other end of the spectrum, the Urban search and rescue robot (USAR) [28] developed by Idaho National Laboratory (INL) on a modified ATRVJr robot platform from iRobot [17] is the most advanced and aware robot of its class. In an international competition sponsored by the Defense Advanced Research Projects Agency (DARPA) and the American Association for Artificial Intelligence (AAAI), not only did the INL robot win first place in the competition, but it was able to enter areas inaccessible to other competitors and was able to reliably identify the “victims” ID tags, their location within the building and determine whether they were still “alive.” INL’s best run was more than double the score (based on speed, ground covered, and “victims” found) of the next highest competitor’s [29].

INL has four modes of operation [30]; tele-operation, safe mode, shared mode and full autonomy. In teleoperation mode the robot uses arrays of sensors to provide spatial awareness, but is ultimately controlled by a human. The user has full control of this robot via a joystick and is provided with information such as the angle of elevation and inertial effects via a graphical and textual interface. In safe mode, the user and the robot interact more readily with the robot demonstrating some situational awareness by making the decision on the user command based on its sensory information thus allowing the user to navigate the robot at a faster pace. The robot provides information on its surroundings from its sonar, laser, tilt and bump sensors to the user to identify obstacles and elevation angle...
allowing the user to react accordingly. The robot is given mission objectives by the user in shared control mode but the robot chooses its own path to execute these objectives. The robot requests input from the user at various intervals to guide it along its path to the next objective. Finally, in fully autonomous mode the robot demonstrates spatial and mission awareness. It performs its own path planning, selects its own routes and the only instruction it takes from the user are mission specific instructions such as ‘follow the target’ or ‘search the area’. It builds a map on the fly using frontier-based exploration and localization to help with performing searches over multiple rooms.

1.4.1 Frontier based exploration

INL builds its map on the fly is based on the detection of frontiers which are regions on the border between space known to be open and unexplored space [31] [32]. This technique is known as frontier based exploration (FBE). The robot is equipped with a combination of laser, infrared and sonar sensors to aide its spatial awareness and moves to a frontier where it can see into unexplored space before adding this information to its map. It then moves to the next frontier gaining awareness about its world until the map has been completed. Grids containing information on the probability of occupancy known as evidence grids are used as the spatial representation with the grid being updated every time a sensor is read. Once the evidence grid has been constructed, the cells in the grid are classified by comparing its occupancy probability to the initial probability assigned to the cells [33]. Once classified, the cells are categorised as open, unknown or occupied as shown in Table 1.3. The region between unknown and open space is identified by a process analogous to edge detection and region extraction in computer vision software [33].
An open cell adjacent to an unknown cell is considered to be a frontier cell. Cells adjacent to a frontier edge cell are classified as a frontier region. If the combined cells of the frontier region are larger than a pre-defined minimum size (usually the size of the robot) then this region is considered to be a frontier. Once a frontier has been detected the robot will try to navigate to the nearest unvisited frontier.

<table>
<thead>
<tr>
<th>Frontier detection class</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Occupancy probability &lt; Prior probability</td>
</tr>
<tr>
<td>Unknown</td>
<td>Occupancy probability = Prior probability</td>
</tr>
<tr>
<td>Occupied</td>
<td>Occupancy probability &gt; Prior probability</td>
</tr>
</tbody>
</table>

Table 1.3 Frontier based exploration classification criteria

Figure 1-4 A simple example of frontier edge detection. The robot's loon is marked with an R, occupied cells are black, unoccupied cells are white and unknown cells are grey; cells marked with X are frontier edge cells [33]

Figure 1-5 An evidence grid built by a real robot in a hallway adjacent to two open doors [33]
Figure 1-5 shows an evidence grid which was built by an actual robot. The left hand image shows the evidence grid, whilst the centre image shows the evidence grid with the frontier edges highlighted in red. The right hand image shows the regions where were extracted using blob analysis. Blob analysis is a vision systems technique for scaling, identifying and extracting groups of pixels of a certain pattern within an image.

1.4.2 Positional awareness and mapping

The research topic of “localisation” or positional awareness is an intensively researched and extremely important area. The ability for a mobile robot to know its position in the environment with respect to everything else is essential if they are truly become autonomous.

Numerous individuals and a multitude of research groups have tried anested ideas in a bid to develop the most efficient way of obtaining location information for their mobile robot. Monte Carlo localisation, a progression from Markov localisation, became a very popular framework for solving the “self-localisation” problem as discussed by Kretzschmar et al. [34].

Markov localisation estimates the position of the robot by computing the probability distribution over all possible positions in the environment [34]. The algorithm starts with an initial distribution \( Bel(l) \) that is refined by iteratively and incrementally carrying out two update steps. Firstly, by assuming that the new position depends only on the previous position and the movement of the robot (Markov property), a “motion model” \( P(l|l',m) \) is used to predict the robot being in position ‘l’, assuming that it executed motion ‘m’, and was previously in position ‘l’’ [34].

\[
Bel(l) = \int P(l|l',m)Bel(l')dl' \quad (1.1)
\]
The sensor data is then integrated to further refine and update the Belief function; for this step an "observation model" that models the likelihood of an observation 'o', given the robot is in position 'l', must be provided before the belief function is updated using Bayes Theorem, where $\alpha$ is the normalisation factor ensuring $\text{Bel}(l)$ integrates to 1.

$$\text{Bel}(l) = \alpha P(o | l') \text{Bel}(l) \quad (1.2)$$

Although the implementation of Markov localisation is easier to use than Kalman Filtering, its implementation is non-trivial [34]. Monte Carlo solves the implementation problem by representing the infinite probability distribution $\text{Bel}(l)$ by a set of $N$ samples

$$s = \{ s_i | i = 1, \ldots, n \} \quad (1.3)$$

where,

$$s_i = (l_i, p_i) \quad (1.4)$$

The location is represented by $l_i$, and $p_i$ corresponds to the likelihood of $l_i$ being the correct position of the robot, i.e.

$$p_i \approx \text{Bel}(l_i) \quad (1.5)$$

Similar to Markov localisation, two steps are carried out during the execution of the algorithm; A new sample set $S$ is generated from the previous set $S'$ by applying a motion model $P(l|l', m)$ as follows:

For each sample $(l', p') \in S'$ a new sample $(l, p')$ is added to $S$, where $l$ is randomly drawn from the density of $P(l|l', m)$. The motion model takes into account the robot properties of drift, translational errors and rotational errors.

The sensor inputs are then used to update the robot's belief about its own position; if distance sensors are being used and a map of the environment is given, "ideal" sensor
values can be computed \textit{a priori}. The observation model can then be obtained by "noisy-ing" the ideal sensor readings using a Gaussian noise distribution. Therefore given a sample \((l', p')\) the new weight \(p\) for this sample is given by:
\[
p = \alpha P(o \mid l') p'
\]
(1.6)
Thus, the new weights for the samples \(S'\) provide a probabil distribution over \(S'\) which is used to construct a new sample set \(S\).
\[
s_i = (l_i, p_i) \in S'(1.7)
\]
\[
Prob(s_i \in S) \approx p_i(1.8)
\]
Monte Carlo localisation can be implemented as an "online" algorithm or can be implemented dynamically as the size increases [34]. There are two scenarios where Monte Carlo localisation cannot be implemented; if there are no distance sensors available and if the data obtained from the distance sensors is unreliable, for example in a highly dynamic environment.

Kretzschmar and Enderle [34] outline a Vision Based location system that utilises the Monte Carlo localisation technique including experimental results with their Sparrow 99 robot. They utilise an omni-directional camera to select visual distance features and visual landmark features that is suitable for use in both the BoCup environment and in a real-life office environment. They show their results without their "self-localisation" algorithm and when they use the robot with the "self-location" algorithm. There is a considerable improvement when the self-localisation algorithm is used. They conclude by stating that with some more research into "vision based localisation" a cost effective alternative to laser scanners and sonar sensors can be found.
Demirli and Turksen [35] discuss a method of using sonar data, a global map and fuzzy triangulation to find the localisation of their robot.

Due to the inaccurate readings from odometers, especially when a robot travels for long periods and has to turn corners or has to avoid obstacles, Demirli and Turksen devised a way of identifying a robot’s location in a global map using a sonar sensor on a servo motor. The method is efficient and relatively trivial to implement.

Sonar sensors have high uncertainties associated with them with angular uncertainties and radial uncertainties being of most concern to scientists working with mobile robots. Demirli and Turksen [35] reduce angular uncertainty by characterising it in terms of a fuzzy set and creating a fuzzy membership function for the incidence angle. Radial uncertainty is also characterised and represented by a fuzzy set before being combined with the incidence angle information to compute the shortest distance to the object. Once the angular and radial uncertainties have been characterised, Demirli and Turksen used coarse localisation to provide starting co-ordinates that are passed to a detailed localisation function to compute an accurate “localisation” position of the robot with respect to the global map.

They also introduce an idea of removing “false” reflections from the sonar sensors by implementing a 3% filter i.e. if $d_1 - d_2 > 3\% \times \min\{d_1, d_2\}$ then do not proceed.

A majority of papers, including the two papers outlined, discuss their ideas with respect to a priori information such as global maps, and topological maps; however it is not always possible or feasible for a robot to have a reference map to compare with.

The problem of not having a priori information has opened a new research avenue where scientists have worked on the following problem: “Starting from an initial position a mobile robot should travel through a sequence of positions obtaining a set of sensor measurements...
at each position”, “The goal of the robot is to process this information, producing an estimate of its position while concurrently building a map of the environment [36]. This problem is commonly referred to as the concurrent mapping and localisation (CLM) problem or the simultaneous localisation and map building (SLAM) problem. Tardos, Newman, Neira and Leonard [36] discuss their attempt to find an efficient robust method to map and localise in indoor environments using only sonar data. Their findings contribute to the improvement in efficiency of mapping large scale environments, correct association of measurements and robust estimation of map and vehicle trajectory information.

The most common representation of a map for mobile robotics is grid based followed by topological, feature based and sequential Monte Carlo. Tardos et al. utilise feature based map representation and outline its considerable advantages over grid based mapping, they also use a Hough Transform instead of the commonly used triangulation based fusion. Their experimental results compare the feature based map generated by a laser scanner manufactured by SICK AG capable of gathering thousands of distance readings a second with a feature based map generated by a ring of 24 Polaroid Sonar Sensors. As can be seen from Figure 1-6 the differences are minimal.
Although solutions to the SLAM problem have come very close, there still is considerable room for improvement and it continues to be an area that is heavily invested in, both with time and money.

1.4.3 Positional awareness and navigation

The majority of navigational systems are based on selecting landmarks from the environment surrounding the robot using whatever sensory means are available. There have been many attempts to design and implement a formal, standard navigational system and research groups are continually getting closer to their goal. The predominant problem in the selection of landmarks is, "What constitutes a good landmark?" A landmark that humans would consider good, such as a door for example, may not be instantly recognisable by the robot or be distinguished from a corridor. This problem has encouraged some scientists to look for an algorithm that will select landmarks for the robot whilst other scientists looked for an alternative means of navigation.
One approach compares landmarks derived from sensor data collected at regular intervals with landmarks supplied by a human supervisor. There are fundamental problems with this approach. By pausing at time intervals, collecting sensor data and assigning this data to a landmark it is possible that the robot will miss "good" landmarks and end up with a large group of meaningless, unusable landmark data. With landmarks assigned by a human supervisor, the robot might not necessarily "see" the same landmark.

Marsland, Nehmzow and Duckett [37], have presented an idea that allows the autonomous selection of landmarks from a continuous stream of data without any human supervision. Traditionally landmark-based navigation systems use sensory perceptions e.g. light detection and radar (LIDAR) scans or ultrasonic scans at regular intervals to identify landmarks. Their novel idea selects landmarks when a sensory perception is unexpected, rather than trying to search for human obvious landmarks. Their method is also robot dependent rather than environment dependent, therefore in theory the method should work in any environment after some training. The execution of the method is carried out in two steps; first a general model of the environment is created during a training phase before it is used to predict future sensor readings based on current sensor values. If the predicted sensor values differ from the new sensor values by a certain threshold then the area is selected as a conspicuous landmark. The comparison between the predicted value and the new sensor value is implemented by two methods. The first detects peaks in the prediction error curve to select landmarks while the second utilises a Kalman filter to search for differences in the data. The algorithm was run on two robots, Nomad 200 and Rolland at the University of Manchester, and Bremen University respectively. Accuracy tests were carried out by generating the sum squared difference between prediction errors of different
traverses in the same environment and by checking the landmark consistency leveeach traverse. While the method proved to be successful at autonomously generating larks, Marsland et al. are continuing to research the area to see if the landmarks obtained from their algorithm are “useful” for aiding a robot in navigation of an area.

An alternative to generating a map and classifying landmarks has been suggested by Kidono, Miura and Shirai [38] who describe the generation of maps and classifying landmarks as a “tedious” exercise. They present a navigation strategy requiring minimal user assistance as a basis for further research. They propose that navigation be performed in two stages; map building and autonomous navigation. The robot is guided from its starting position to its goal using a joystick. As it is moved along it creates a map of the environment using a stereo vision system. During this stage the feature positions and viewpoints are recorded enabling the viewing directions to be selected in the autonomous navigation phase. Once the map is created, the robot is capable of computing the shortest path to the goal from the map, before proceeding to follow this path. This method has only been tested in a static environment, but it is envisaged that in a dynamic environment, obstacle avoidance sensors would be used to avoid any crashes. The outline formal framework for the plan execution problem [39] is discussed later.

A third and probably most interesting alternative to solely using landmarks is proposed by Lambrinos et al. [40] who study the Saharan Ant Cataglyphis and its methods of navigation. It has been shown in studies that the Saharan ant Cataglyphis when foraging for food can return to its nest in almost a straight line, and Lambrinos et al. [40] designed and built a robot based on its sensory attributes. It can be seen from Figure 1-7 that it vs a
random path until the food is found, the location represented by a black circle. Once the food is found, the ant will follow an almost direct path back to the nest.

![Diagram of ant path](image)

**Figure 1-7** The path taken by a Saharan Ant Cataglyphis when searching for food, the large black circle signifies the point where the food is found; the white circle signifies the point where the ants nest is located.

Sensors were created to correspond to different parts of the insect’s eye, including a panoramic camera, ambient light sensors and polarised light sensors. It has been discovered that the ant uses landmark navigation as well as the polarised light pattern of the sky to navigate so two models were created to implement this in the Saharabot II Robot.

A polarised light (POL) compass is used to reproduce the early processing stages of the insects’ polarisation vision system, and a model was derived to extract the relevant signals from the POL compass. It was also discovered that the ant takes a panoramic snapshot of its initial location before it leaves the nest and stores it in the retina. When it has found its food, it takes another panoramic snapshot of the area and computes a vector in the direction of home. The tests carried out by Lambrinos *et al.* shows that this method works in a real life environment.
An important contribution to mobile robot navigation has been carried out by Belker, Beetz and Cremers [39], where they formalise a solution to the *plan execution problem*. The task of navigation can be broadly categorised into two forms, global navigation and local navigation. Global navigation is commonly computed before the robot starts to move, it often analyses a map of the operating environment and computes the shortest path to the goal, before giving instructions to the robot telling it where to move. Local navigation occurs in the immediate vicinity of the robot for which there is no global map. It deals with obstacle avoidance and adjustments necessary to reach the final goal. Both global and local navigation planning are commonly used to complement each other; however this has lead to the formation of the plan execution problem.

The plan execution problem is the problem of how to link global with local, or how to “compute target points for the local navigation component from the global plan” [39]. This is a problem inherent with any system using global and local navigation and Belker *et al.* are a major contributor to the formalisation of the plan execution problem. They propose that a Markov decision process be used to select actions for navigation in the local domain based on the global map. The models needed to select these actions are automatically learned using simple multi-layer perceptrons and regression trees. The simulations carried out show the power and applicability of their solution for a formal framework for the plan execution problem.

### 1.4.4 Positional awareness and path planning

Path planning is a very important piece of the jigsaw for a fully aware robot. Consider for example a robot for use as a nurse’s aide in a hospital; if the nurse instructs it to go to the store room and pick up a bandage for a patient, the robot must be able to plan the most
optimal return trip path to the store room. The research group at the Field Robotics Center, CMU, headed by Anthony Stentz has developed an algorithm known as the “Framed Quadtree D*” algorithm for optimal path planning. Their approach has yielded the quickest execution and the most optimal path for roughly the same computational power.

Different solutions have been created to find the optimal path planning algorithm, however approaches to path planning for mobile robots can be broadly classified into two areas. Those that use exact information to represent the real world, and those that use discretised representations. The advantage of discretisation is that there is greater control over the computational complexity and it is easier to implement than the “exact information” equivalent.

Regular grids were originally used for path planning, but they are computationally expensive, represent space inefficiently and only allow for 8 angles of direction; therefore an optimal path is almost impossible to generate. There are smoothing algorithms that allow the path to travel off the centre of the grid [41], but this only offsets the optimal path even further.

An improvement was made to the computational cost with the introduction of quad-trees. Quad-Tree architecture is based on the sub-division of a region into 4 equally sized objects, with each region being recursively sub-divided until a sub-region free from obstacles is found, or the smallest grid cell is reached. The advantage of quad-trees is that a single large cell can be used to encode large empty regions, thus reducing the memory constraints. Paths generated by quad-trees are suboptimal, and it was with the introduction of framed quad-trees that an optimal solution to path planning was found.
Framed quad-trees are similar to quad-trees, in the way that they are recursively subdivided until a sub-region free from obstacles is found, or the smallest grid cell is reached. However, quad-trees allow for optimal paths by placing a frame of cells of the highest resolution around the perimeter of each quad-tree region [42]. Each border cell is then mapped to its neighbour, allowing for optimal path planning on the shortest traverse.

Two simulations were run to compare the regular grid D* algorithm with the framed quad-tree D* algorithm, one simulation was with full a priori knowledge while the other simulation was run in an unknown environment, depending on the vehicle sensors to detect obstacles. The algorithms were compared under the headings of traverse length, memory usage and execution time as shown in Figure 1-8.

The framed quad-tree D* algorithm was found to have a shorter traverse length, with a shorter execution time than the regular grid D* algorithm for both the known environment and the unknown environment. The regular grid D* algorithm required more memory for the unknown environment, but less memory for the known environment.

The framed quad-tree D* algorithm has also been tested on an autonomous jeep with obstacle avoidance capabilities. It was shown that the jeep moved 200m in 6 minutes.
detecting 80 obstacles on the way; it was also shown that this was the efficient path the robot could have taken to get to its goal (Figure 1-9).

![Figure 1-9 A successful long traverse using a framed-quad-tree D’rithm on an autonomous Jeep [43].](image)

1.4.5 Behaviour based awareness

Nature has been used as a basis to model technology on for many's; the Wright brothers for example spent a large portion of their time observing birdflight and used this information to build the first airplane and the lesser known George estral invented Velcro™ from modelling it on the hooks of burrs. Behaviour based robuses nature as a basis for modelling intelligent systems and the sensors which provide a bot’s with the required sensory systems for different types of awareness.

Some researchers are using creatures in the lower echelons of the animadom to model their robots on due to the fact that their movements are behavioural rather than cognitive. An experiment carried out by scientists in the 1990s showed that when severed the connection between the brain and spine of a frog they could still get it to move by stimulating certain areas of its spine thus proving that a majority of the's movements were reactive and not the result of the thought process [44]. Taking into account
researchers are designing their high-level systems based on layers of lower level behaviour based systems.

1.4.6 Awareness and the anchoring problem

Anchoring is defined as the process of creating and maintaining the correspondence between symbols and sensor data that refer to the same physical entity. The problem is how to perform anchoring in an artificial system.

The anchoring problem is an extremely difficult and still open problem that has been pondered on by philosophers and scientists for hundreds of years. It is commonly agreed by scientists that a practical solution must be found if robotics systems that include symbolic components are to be developed. However, it has only been addressed as a problem in the area of robotics and given the attention it requires since the early 2000s as outlined by Silvia Coradeschi and Alessandro Saffiotti [45].

Consider the following real life example; you are asked by a colleague to go to the kitchen and get your colleague’s mug. You begin the task by walking to the kitchen, you look on each shelf for your colleague’s mug, when you see the mug you reach out, grab it and bring it back to the office. Your colleague knows what he wants, but he cannot see his mug and you only have an incomplete description of the mug. Therefore in order to carry out this task or any other task for that matter, there must be a correspondence between the name of an object and the perceptual image of that object. In short, anchoring is the establishment and maintaining of this correspondence in an artificial system and hence enabling an awareness of the object. Anchors can be used for controlling motion, co-ordinating task execution, engaging in communicative actions and enabling human-robot interaction as outlined later in this chapter.
There are a number of challenges associated with the implementation of an anchoring framework. The anchor definition assumes that the available sensor data is capable of being segmented to isolate percepts (i.e. measurements assumed to originate from the same physical object) that correspond to individual objects, this is however a domain specific and difficult problem [46].

To aid with the anchoring problem, spurious data from sensors must be filtered to avoid errors, and uncertainty and ambiguity must be eliminated from the anchoring framework itself. The uncertainty referred to in the anchoring framework is inherent from the vagueness of symbolic descriptions. Also, one must find a formal way to refer to objects at symbolic level i.e. how would an anchoring system implement an indefinite description such as “Find me something to scratch my face with”.

A practical solution to the problem developed by Coradeschi and Saffiotti suggest that an anchoring framework similar to the one described in [45] [47] [48] be followed. This framework assumes that the anchoring process is performed on an intelligent system that comprises of a symbol system $\Sigma$ and a perceptual system $\Pi$. The predicate grounding relation $g$ is then used to connect individual symbols in $\Sigma$ to $\Pi$. To execute each anchoring step the framework suggests that three abstract functionalities: Find, Track and Reacquire are used to manage anchoring.

Tests have been carried out by Coradeschi and Saffiotti in different environments, outlining the flexibility of their approach. The first test outlined in [45] uses a Nomad 200 robot with a sonar sensor ring to execute commands given in terms of rooms and corridors. e.g. Follow(corr4) tells the robot that it should follow corridor 4. Once the command is given, the symbolic description of corr4 is placed in the local perceptual space (LPS) before the
descriptor is anchored to wall percepts using the Find function. The Track function is then used to keep the symbol anchored to new percepts.

The second test utilises an unmanned aerial vehicle (UAV) performing surveillance tasks in a simulated environment. The images contain two identical cars travelling in the same direction with the UAV tracking the car as outlined in Figure 1-10. The car passes underneath the bridge at $t_1$, with the car on top of the bridge approximately at the same position as the original car. The track function has access to the road topology and can decide that the top car is not the one that it was originally tracking. This initiates the Reacquire function to locate the original car, and the Reacquire function predicts where the car will next appear and directs the camera and UAV to this position. At $t_2$ the car reappears, a new percept is generated and normal tracking is resumed.

![Figure 1-10 Dynamic capabilities of anchoring framework](image)

The final experiment carried out by Coradeschi and Saffiotti is for anchoring an indefinite description. This test was carried out using a SONY AIBO entertainment robot on a RoboCup soccer playing area. The playing area contained ten identical red balls, and the task required the robot to “Score” the ten balls in turn. After a score the ball is removed from the playing area. They used the find function to select the first ball and anchor a
symbol to it before using track until the ball is moved into the goal. Reacquire is then used to locate a new ball and repeat the process until there are no balls left.

The anchoring problem straddles the fields of philosophy and linguistics and is still far from being understood. It is however extremely important for scientists to continue research in this topic as robot systems are required to operate in highly dynamic environments executing more complex tasks. It is suggested by Coradeschi and Saffiotti that "in the longer term, a research programme on anchoring should bring a deeper theoretical analysis for the anchoring problem, together with general practical solutions that can be re-used in different systems and domains" [45].

1.4.7 Human - robot interaction and situational awareness

If robots are to become commonplace in the home or in working environments significant developments are required in the field of human-robot and robot-human awareness. It is not possible for manufacturers of "Commercial" robots to program every single action, or possibility into a robot's core, so a probable solution to this problem is to allow the robot to learn in its environment by interaction with a human. It is however very important if robots are to be accepted that the means of communication must be non-intrusive, i.e. the user should not have to wear special clothing, special gloves or a specially constructed helmet.

Lauria et al. discuss a design of a practical solution to the programming of a robot using a "natural language" [49]. They suggest that learning by imitation or learning by reinforcement is a lengthy process that is impractical for "naive" home users, and that instruction based learning (IBL) using unconstrained speech has several advantages. The method discussed by Lauria et al. [49] reveals that "natural language uses symbols and syntactic rules and is well suited to interact with robot knowledge represented at the..."
symbolic level”. It has also been shown by Cangelosi [50] that learning in robots is more effective at the symbolic level.

An efficient and complete IBL requires several steps to transfer a spoken “chunk” into a robot action (Figure 1-11); the system firstly converts the “chunk” of speech to text before syntactic parsing and semantic analysis is executed. The system must then transform the “user sentence” at a functional level into internal symbols that the robot can understand. Once the user sentence has been converted, the robot can initiate sensors and motors as required.

Figure 1-11 Steps involved in transferring a "chunk" of spoken data into a robot action. [49]

The system devised [49] learns from a base of predefined knowledge, or a database of vocabulary used to describe certain situations that is linked to a list of primitive procedures that run sensors and motors. An experiment to gather a corpus of descriptions of route directions was carried out using 24 people, divided into three groups of 8. They were asked to describe particular routes for the robot to follow, using “free flow” speech. In total, descriptions for 144 routes was collected, helping in tuning the speech recognition system to the vocabulary used and establishing a list of primitive procedures that users refer to in
their instructions. It was discovered that on average one new procedure is defined every 38 route instructions. It was also discovered that users have an unrealistic reflection of the knowledge a robot has, they were producing complex primitives such as “go round the bend”, “cross the road” and “go round the roundabout”, which proved difficult to implement. It is important that the corpus of instructions is as wide as possible and capable of updating as new instructions are given.

Another important area of human robot interaction that is being actively researched is the study of emotion and ethology as a basis for implementation in mobile robots. Headed by Dr. Ronald Arkin and supported by the engineers at SONY, studies have been carried out on the behavioural patterns of animals and the implementation of these behaviours in software.

Ethology, founded by Lorenz and Tinbergen [51] in the early 1900s refers to the study of animals in their natural setting, “the work seeks to extract from observational behaviour suitable descriptions of animal activity that can be effectively mapped onto robotic systems to provide the appearance of life like activity” [52]. Scott and Fuller [53] provide a commonly used ethogram that spans the range of canine activities, suitable for mapping onto robotic systems.

While the ethological model provides a basis for the kind of behaviour associated with animals, an emotional basis is also required to allow the robot select an action in a given situation. Ideally a dog will not bite your hand if you are petting him even though you are in contact with the animal at all times; likewise a robot should not “lash out” if you come too close to change the batteries or to fix a wheel.
Six emotional states, happiness, anger, sadness, fear, surprise, disgust were proposed by Ekman and Davidson [54], however it has since been discovered that it is possible to use only 3 emotional states, pleasant, arousal and confident to generate an emotionally aware robot.

A robot capable of human-robot interaction (HRI) for domestic or office use would not be complete without sensors enabling the robot with the ability to distinguish humans from other objects in the environment. Authors Fritsch et al. discuss an idea for “multimodal anchoring” using a SICK laser scanner in conjunction with a SONY EVI-D31 camera running on a PeopleBot from ActivMedia [55]. The non intrusive nature of the SICK Laser scanner and the image capture techniques used in extracting information from the camera make this method ideal for use in domestic and office environments. By extending the work carried out by Coradeschi and Saffiotti [45] [47] they implemented an anchoring framework in C++ as well as adding a person tracking feature to the ISR software developed at the Center for Autonomous Systems at the KTH, Stockholm [56]. Instead of the traditional method of looking for only one colour which is subjected to variations in different light conditions, their algorithm is able to detect a human face by looking for multiple instances of “skin colour” in an image and “anchoring” these multiple instances of “skin colour” to a face object.

The methods discussed by Fritsch et al. is a basis for the development of more advanced HRI, and is currently being researched by different groups across the world.

1.4.8 Impact of sensor positioning on awareness

The problem of optimal sensor positioning is one that has troubled engineers for years, and has only recently evolved to the stage where genetic algorithms are becoming popular as a
method of deciding the optimal positioning of sensors. Throughout the years, numerous
techniques have been tried in a bid to find optimal sensor positioning; namely trial and
error, intuitive placement, heuristic recipes and systematic optimisation methods [57].
Ray and Mahajan [57] have developed a novel method of using genetic algorithms (GA) to
optimally position multiple ultrasonic receivers for a single ultrasonic transmitter. The GA
will work for any sensor that operates on the “time of flight” principle or uses wave theory.
The system devised by Ray et al. [57] is more accurate than other implementations such as
those implemented by Figueora and Mahajan [58] by making the transmitter independent
from the receiver and eliminating the need for a time delay which is common in ordinary
ultrasonic systems.
Even though the use of GAs is gaining popularity in terms of finding optimal solutions for
complex sensor/actuator problems, the area still remains an active area of research.
Ultrasonic sensors are widely used for mobile robotics, due to their ease of use and ease of
hardware interface. They are cheap, and they give relatively good distance values if used in
an array. Ultrasonic sensors do however have a number of problems associated with them,
the first being specular reflection. Specular reflection causes all sonar waves to be reflected
as if they are hitting a mirrored surface; this is due to the fact that most objects in a mobile
robot’s environment have a surface that has a smoother texture than that of the ultrasonic
waves. Secondly due to the wide beam angle of a sonar beam (typically a conic shape
between 15° and 30°) it is sometimes difficult to measure the exact location of an object
without a second or indeed third pair of ultrasonic sensors. Thirdly, the slow propagation
velocity leads to slow data acquisition.
Many attempts have been made to use the ultrasonic sensor for classification of objects; Kleeman and Kuc [59], Chung et al. [60] and Sabatini and Rocchi [61] all have made valuable contributions. Han and Hahn [62] have developed a new measurement scheme that uses two sets of ultrasonic sensors to determine the location and type of a target surface. They outline in [62] the theory and implementation of their measurement system that improves the accuracy and the need to take multiple samples of data to get accurate distance and location measurement. The sensing system is capable of classifying a target surface as a plane, a corner, a cylinder or an edge and is capable of localising its position. The method outlined by Han and Hahn [62] with some extension may be used as a cost-effective replacement for a laser scanner and could be used to classify objects and anchor symbols to them for robot learning.

1.5 Design for reliability

No human activity can enjoy zero risk, and no equipment a zero rate of failure, therefore a safety technology has grown for optimising risk by taking these failure rates into consideration. For reliability engineering purposes, reliability is defined as "the probability that a device, system or process will perform its intended function during a specified period of time under stated conditions [63]". Keeping this in mind it is known that it is easier to design reliability into a system rather than having to patch it in after.

There are a number of ways to design reliability into a system but the basic areas that one must keep in mind when designing a reliable system are complexity, duplication and derating. The less complex a design, the less components are involved and as a result the more reliable the design will be; critical components should be duplicated to account for the
failure of either one of these components and components should be derated to a level in excess of the level where it is anticipated the components will be used.

As reliability is based on statistical information, there are a number of techniques which can be used to design and analyse a reliable system. For the analysis of an electronic system, Weibull analysis is a very effective tool and when combined with Highly-accelerated life testing (HALT) or Highly-accelerated stress screening (HASS), one has a very good idea as to how long the design will last. From a software point of view the Unified modelling language (UML) and continuous integration creates a reliable platform for the development of software no matter what language one is coding in.

The core of a reliable system is based on well specified requirements; therefore the system will only be as reliable as these requirements. Typically the system will be divided into hardware and software requirements. Once the requirements have been specified and agreed upon for the hardware and software systems, they should be designed exactly to meet these requirements.

1.5.1 Designing reliable hardware

Every component on a hardware platform has a failure rate associated with it as determined by the manufacturer of the component and this information can be used to determine the reliability of the system. After the schematics have been designed to meet the hardware requirements an analysis of the system can be carried out using the reliability information provided by the component manufacturer.
1.5.1.1 Fault tree analysis

The fault tree analysis methodology was developed by Bell Labs in 1962 for the US air force and was later adopted and used extensively by Boeing [64]. A fault tree built using blocks representing gates and events is a top-down graphical representation that uses Boolean logic to represent combinations of events that lead to the overall system failure. The example in Figure 1-12 shows a fault tree diagram for the system failure by explosion. The manual analysis of each probability in complex systems is quite time consuming, so software programs such as those provided by Reisoft [65] can be used to generate accurate reliability data in a timely fashion.

![Fault tree diagram](image)

Figure 1-12 Example of a fault tree [66]

1.5.1.2 Design failure modes, effects and criticality analysis (DFMECA)

The Design failure mode, effects and criticality analysis (DFMECA) is a key document to author so as to ensure a reliable system. This document describes in detail what happens when any single component fails on the hardware platform from the smallest decoupling capacitor to the 64 bit microprocessor controlling the operation of the system.
Each component will be assigned a risk priority number based on its location in the design and a decision will be made on how best to eliminate this risk where possible.

### 1.5.1.3 Wiebull analysis

Once the design and the DFMECA has been finalized a Wiebull analysis can be carried out on the hardware system as a whole so as to determine how long the system is going to last. This analysis is implemented by gathering and processing the statistical information on the failure rates of the components involved in the circuit resulting in a number of results and plots including mean life, failure rate, and probability of failure given time and reliability Vs time. Typically this information is processed using a software package such as Reliasoft [65].

### 1.5.1.4 HALT/HASS

The last number of years has seen a move towards Highly-accelerated life testing (HALT) and Highly-accelerated stress screening (HASS) so as to supplement traditional environmental testing to determine the reliability and quality of a system. Unlike traditional environmental testing where the engineer will determine if their product meets a certain standard. HALT/HASS tries to break the device so as to determine the point at which the device is going to fail and hence expose the weak links in the design. Once the HALT/HASS has been completed the design can be fine tuned to ensure the utmost reliability.

### 1.5.2 Designing reliable software

As stated previously, the core of a reliable system is well defined requirements. Once the software requirements have been defined and the coding language agreed upon there are a
number of methods which can be used to help with reliability. One of the most popular development frameworks in the 2000’s own as Agile software development which was created out of necessity for developing very reliable software in a short amount of time. It is an iterative development framework that is based on the methods used in lean manufacturing and six sigma. Feature driven development is part of the Agile software development framework and when utilised with a modelling language such as the UML and rigorous testing one can develop high quality and highly reliable software.

1.5.2.1 FDD, the UML, testing and continuous integration

Feature driven development (FDD) is an iterative development framework that was proposed by Jeff De Luca in 1999 so his team of software engineers could quickly develop high quality software. It defines basic activities, develop overall model, build feature list, plan by feature, design by feature and build by feature.

![Feature Driven Development Framework](image)

The first activity is to develop a high level model of the system by combining the domain level models designed by each team of software engineers. Once the overall model has been created, a list of features is defined where each feature must be capable of being developed in a two week cycle. The complexity of the system is not reduced by doing this but the breaking down of the model into pieces of this size reduces the likelihood of errors and thus increases the reliability of the code. The development plan can then be created and tasks assigned to developers based on lists of features. Every two weeks a new set of
features is designed, developed, reviewed and tested before being promoted to an official release build.

The UML helps with the modelling phase of FDD and uses block diagrams to help the software engineer design his system to meet all of the software requirements defined in the software requirements document.

With the UML in place for each feature the next phase is the authoring of the software; in order to ensure that this software is robust it is important that it is tested thoroughly. This testing can take many forms but the most critical elements to ensure reliability are unit testing and system integration testing.

Unit testing should be carried out on every function that is written in software ensuring that the function does exactly what it is supposed to do.

System integration testing is the testing that is carried out as part of the overall system; i.e. how does the code interact with the rest of the system namely the electronic components such as sensors and motors.

Similar to the accelerated testing carried out as part of the hardware design, agitation can be carried out on the software to automate and accelerate the testing and expose possible weak links in the overall system software.

Continuous integration is a model for build and deployment of software and incorporates unit testing, agitation [68] and system integration into a single build. This philosophy ensures that coding and design errors are caught at an early stage.

Other tools such as bug reporter and project management tool JIRA [69] should be used to keep track of tasks and coding errors so that they can be followed to closure.
1.6 Preventing harm to humans and situational awareness

In the pursuit of developing a truly aware robot, the design engineer must ensure that his design has considered the possibility of harm being caused to the human which it is interacting with. Robots are designed to replace humans for repetitive or hazardous tasks but the closer the interaction between human and robot the greater the danger of death or serious injury and there has been a number of examples that demonstrate this. In 2005 there were 77 robot related incidents reported to the Health and Safety Executive in Britain [70]; in 2001 a man died from injuries sustained after a robotic arm pinned his head against the vertical post supporting the robot as it attempted to return to its home position [71] and in 2007 in spite of a safety zone in place around an industrial robot, a Swedish worker was injured when a robot which he was repairing grabbed his head and broke four of his ribs [72]. Reports and studies from Sweden and Japan where there is a large number of industrial robots show that many of these reported robot accidents occurred during the programming, maintenance, testing and setup of the robot where the engineer is temporarily within the robot’s working envelope [24].

These reported accidents are attributed to both mechanical failure and human error; for example mechanical failure in a pick and place robot could cause the robot to drop what it is carrying and thus seriously injure a person whilst an example of human error would be a robot service engineer neglecting to enable a safety switch on the robot’s panel before attempting to repair something within the robot’s working envelope.

An un-aware industrial robot designed with safety in mind should have a multitude of alarms such as sirens and flashing lights as well as conveniently located emergency stop buttons. A crude safety precaution required by health and safety authorities in most
countries removes power completely from the robot once the enclosure around the robot is breached; this safety net usually takes the form of a light fence or a cage.

Safety during maintenance is also something that must be considered when designing a robot; a deadman switch is the most common whereby the maintenance engineer must constantly apply a pressure to a safety switch when inside the working the robotic envelope; once the pressure is removed from this switch the robot is disabled immediately.

The same safety considerations must be taken for mobile robots as they become larger and the tools they carry become more hazardous. Mobile robots designed for defence style missions such as security patrols, mine disposal and environmental monitoring implement a similar type of deadman switch during maintenance. For example a mine disposal robot must not detonate the explosive before it is safe to do so and the relevant safety precautions must be in place to ensure this.

1.7 Concluding remarks

Chapter 1 provided an introduction to the concept of awareness in robotics applications by firstly introducing the types of awareness followed by a review of the most modern research projects that are currently being undertaken in the field of robotics and awareness. The discussion then proceeds to a description of the best methods of design for a robotic system describing the importance of a DFMEA document and HALT/HASS testing. Finally the some of the obstacles such as the SLAM problem and implications of increased human-robot interaction is discussed. Chapter 2 describes the sensors that are required to provide a robot with the awareness discussed in Chapter 1 and identifies the most important sensor characteristics when choosing a sensor for an aware robotic application.
Chapter 2 Sensors

The environment is full of different physical quantities waiting to be measured, and for every physical quantity, there is a sensor available to measure it. Arrays of sensors are critical to the creation of an effectively aware robot. Sensors are crucial for a number of reasons. Firstly they play a key role in the internal feedback systems that ensure accurate and precise robotic linear and rotary actuation and positioning. Secondly they provide information about the internal and external environment of the robot. Finally they provide information about the position of the robot with respect to its physical surroundings.

This chapter presents a review of the sensors that are critical to the creation of an aware system.

2.1 Sensors for internal feedback

Robotic systems that are used in industry have been using internal feedback systems for a long time and are a necessary part of the robotic system as they provide it with information regarding the status of its internal components. These internal feedback systems commonly take the form of accelerometers, linear variable differential transformers (LVDT), torque sensors, thermocouples/thermistors, internal voltage/current measurements and positional sensors.

2.1.1 Voltage and current sensors

Voltage and current measurements can be used to provide valuable feedback regarding the health of the robotic system e.g. the amount of charge left in a Lithium Ion battery is generally determined by electronics built in to the battery itself and the current drain on a system can be determined by measuring the voltage drop across a 49mΩ resistor and
feeding this information back to the DC/DC converter so it can increase or decrease the current output as required. Generally these measurements are not made with specific sensors but the voltage and current readings are routed to dedicated ICs from critical points in the system and fed back to the microcontroller for decision analysis.

2.1.2 Accelerometers

An accelerometer measures both static and dynamic acceleration forces and generally come in the form of piezoelectric or capacitive accelerometers. They have many uses in robotics from detecting the speed and angle of a robotic manipulator to the determination of the angle at which a search and robot rescue robot is operating at.

Capacitive accelerometers sense a change in electrical capacitance with respect to acceleration and are constructed with a diaphragm sandwiched between two fixed plates which in effect create two capacitors. The diaphragm flexes during acceleration causing the distance between it and the fixed plates to change hence changing the capacitance. The acceleration is then calculated by measuring the two capacitance values.

A piezoelectric accelerometer uses a crystal as the sensing device. The crystal is attached to a mass which compresses it when it is subjected to a force; the onboard electronics then sends out a signal to the microcontroller with this force information.

There are a number of electronics manufacturers which make accelerometers that can be easily integrated onto a PCB for use in a robotic system. Some of the most common are outlined in Table 2.1.
<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Manufacturer Part Number</th>
<th>Sensitivity</th>
<th>Typical Applications</th>
<th>Supply Voltage (V)</th>
<th>Number of Axes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freescale</td>
<td>MMA3201EG</td>
<td>40G</td>
<td>Virtual reality input devices, hard drive protection.</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>STMicroelectronics</td>
<td>LIS3LV02DQ</td>
<td>±2G/±6G</td>
<td>Free-fall detection, games controllers.</td>
<td>3.16</td>
<td>3</td>
</tr>
<tr>
<td>Analog Devices</td>
<td>ADIS16209CCCZ</td>
<td>±1.7G</td>
<td>Motion or position measurement, medical monitor devices.</td>
<td>3.6</td>
<td>2</td>
</tr>
<tr>
<td>Mesmic</td>
<td>MXA2300JV</td>
<td>±1.0G</td>
<td>Tamper detection, mouse input.</td>
<td>5.25</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2.1 Modern Accelerometer sensors

2.1.3 Linear variable differential transformers (LVDT)

Linear variable differential transformers (LVDT) are electromechanical transformers that are capable of measuring differing degrees of movement and converting them to an electrical signal. The LVDT operate in the same way as a transformer with the output equal to the differential output of the secondary windings. This value is based on the position of the core inside the LVDT. LVDTs are commonly used both as an internal feedback sensor where the LVDT provides the robotic system with information regarding the position of its manipulator limbs and as an input device where the LVDTs are used to sense the position of a human operators arm position and in turn direct the arm of the robotic manipulator [73]. Included in Table 2.2 is a list of some LVDTs’ and their typical applications.
Table 2.2 Modern LVDT sensors

2.1.4 Temperature

Temperature sensors are useful sensors for incorporating into the robot's design and can be grouped into contact and non-contact sensors. Contact sensors, such as thermistors or thermocouples, will generally be used to monitor the temperatures of the critical electronic components within the robot whilst the non-contact variety such as thermal imagers or radiation thermometers can be used to determine the environmental temperature and the temperature of objects in close vicinity to the robot.
2.1.4.1 Internal temperature

Most standard electronic components are rated to operate between -40°C and +85°C. Therefore to ensure the reliable operation of the robot it is essential that the temperature of the critical electronic components are monitored. This is particularly important if it is intended that the robot is to operate in a harsh environment where it is possible that the temperature will reach or exceed 60°C. In general microcontrollers, microprocessors and digital signal processor (DSP) integrated circuits (ICs) are equipped with an internal temperature monitor but if they do not it is a good idea to monitor the temperature of these devices with a thermocouple.

The thermocouple is one of the most readily available and easiest ways to measure the temperature of a component and is based on the Seebeck effect that occurs in electrical conductors that experience a temperature gradient along their length [75]. Their wide temperature range and their ease of interface make thermocouples the most popular form of temperature measurement in science and industry. The K type (chromel/chromel-alumel) thermocouple for example can measure between -200°C to 1350°C and can be interfaced easily with a microcontroller via a MAX 6675 IC [76].

Honeywell produce stable, almost linear low power platinum resistance thermometers (HEI-700). While thin film resistance temperature detectors (RTD) are less sensitive than thermistors, they do provide a very high base resistance and high device sensitivity capable of measuring between -200°C and 600°C [77]. Included in Table 2.3 is a list of modern temperature measurement sensors, their typical applications and temperature measuring range.
<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Manufacturer Part Number</th>
<th>Sensor Type</th>
<th>Typical Applications</th>
<th>Measuring Range (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABFACILITY</td>
<td>FM-J-U-152-5</td>
<td>J-Type thermocouple</td>
<td>Plastics machinery, injection moulding.</td>
<td>0°C - 450</td>
</tr>
<tr>
<td>LABFACILITY</td>
<td>011046TD</td>
<td>RTD.</td>
<td>Flowing fluids, process facilities.</td>
<td>-100 to +1100</td>
</tr>
<tr>
<td>Pico Technology</td>
<td>SE017</td>
<td>Platinum resistance thermometer.</td>
<td>Measuring air temperature.</td>
<td>-75 to +250</td>
</tr>
<tr>
<td>Vishay</td>
<td>2381 633 53104</td>
<td>NTC thermistors</td>
<td>Automotive systems, domestic appliances.</td>
<td>-40 to 200</td>
</tr>
</tbody>
</table>

Table 2.3 Temperature sensors

2.1.4.2 Environmental temperature

Non-contact sensors based on the thermal emission of radiation such as radiation thermometers, thermal imagers and optical pyrometers can be used to measure the temperature of objects in the immediate surroundings of the robot. This is particularly useful for search and rescue robots, where thermal imaging cameras and image recognition software can be used to locate the presence of a human.

Although only available to original equipment manufacturer (OEM) customers one such sensor that could be used is the THERMAL-EYE™ 3500AS [78] from L3 Communications. This sensor has a human detection range of 100m and outputs images at 30 frames per second. It has a universal serial bus (USB) interface so it is possible to integrate it with any microcontroller with USB capabilities where additional processing can be carried out.

Table 2.4 compares the key characteristics of the THERMAL-EYE™ 3500AS, the THERMAL-EYE™ 4506AS and the MCT1500/3000 from FLIR.
Table 2.4 Thermal imaging sensors

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Part Number</th>
<th>Frame Rate, Output</th>
<th>Typical Applications</th>
<th>Communications Interface</th>
<th>Measuring Range to Humans (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3 Communications</td>
<td>THERMAL-EYE 4506AS</td>
<td>30fps, NTSC/PAL</td>
<td>Security, search &amp; rescue.</td>
<td>RS232</td>
<td>1860</td>
</tr>
<tr>
<td>L3 Communications</td>
<td>THERMAL-EYE 3500AS</td>
<td>30fps, NTSC/PAL</td>
<td>Security, search &amp; rescue.</td>
<td>USB</td>
<td>100</td>
</tr>
<tr>
<td>FLIR</td>
<td>MCT1500/3000</td>
<td>30fps, NTSC/PAL</td>
<td>Marine, automotive thermal navigation.</td>
<td>RS232 or TCP/IP</td>
<td>6300</td>
</tr>
</tbody>
</table>

2.1.5 Encoders

Incremental encoders are one of the most common sensors used in the determination of angular position and direction of a motor in the field of robotics [79]. These devices output transistor-transistor logic (TTL) pulses based on the position of the code track disk which is attached to the motor driving the movement of the robot. The code track disk has a pattern of opaque and transparent sectors which corresponds to the number of cycles the encoder can output and is read by a photodetector which in turn generates the required TTL signal. There are three types of encoders; single channel, quad channel and absolute encoders,
examples of which are given in Table 2.5. The single channel encoder, more commonly known as a tachometer uses a single photodetector to indicate rotational speed of the motor but not direction.

![Photo Sensor Diagram]

Figure 2-3 Typical single channel encoder and absolute encoder [80]

Quad encoders use two photodetectors offset by 90° from each other that are capable of measuring the speed and direction of the motor and will generally be implemented in pairs, one to the motor and one to the payload to provide additional feedback to the robot.

Absolute encoders are used in robots where it is necessary for it to know the position of its motor at startup; these encoders are typically used in industrial pick and place robots.

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Manufacturer Number</th>
<th>Part Number</th>
<th>Encoder Type</th>
<th>Typical Applications</th>
<th>Speed (min^-1)</th>
<th>Load Capacity (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KUBLER</td>
<td>05.2400.1122.0050</td>
<td>Incremental</td>
<td>Motor feedback</td>
<td>12000</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Vishay</td>
<td>DE2-1-1-32</td>
<td>Incremental</td>
<td>Motor feedback</td>
<td>200</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Omron</td>
<td>E6A2CW3C-200</td>
<td>Rotary</td>
<td>Robotic manipulators</td>
<td>5000</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Avago Technologies</td>
<td>AEAS-7000-1GSG0</td>
<td>Absolute</td>
<td>Rotary applications</td>
<td>16MHz</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.5 Modern encoder sensors
2.1.6 Force, pressure and torque transducers

Force, pressure and torque transducers can be used by robots to sense force and movement. These transducers, more commonly known as load cells, strain gauges and torque cells often use a Wheatstone bridge to convert small changes in force, pressure and torque into an electrical signal which can be used by the robot’s microcontroller.

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Manufacturer Number</th>
<th>Part</th>
<th>Sensor Type</th>
<th>Typical Applications</th>
<th>Sensing Range (N)</th>
<th>Sensitivity (mV/g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHMIDT</td>
<td>BS 15.024</td>
<td></td>
<td>Force</td>
<td>Automotive occupant safety.</td>
<td>0 – 15.6</td>
<td>0.1</td>
</tr>
<tr>
<td>DYTRAN</td>
<td>1203V</td>
<td></td>
<td>Force</td>
<td>Automated machinery, process monitoring.</td>
<td>0 – 444.8</td>
<td>0.1</td>
</tr>
<tr>
<td>SCAMIE</td>
<td>K1613</td>
<td></td>
<td>Force</td>
<td>Aeronautics, safety systems.</td>
<td>100</td>
<td>0.0014</td>
</tr>
<tr>
<td>Honeywell</td>
<td>FSG15N1A</td>
<td></td>
<td>Force</td>
<td>Robotic end effectors, occlusion detection.</td>
<td>0-14718</td>
<td>0.2</td>
</tr>
<tr>
<td>ATI</td>
<td>Nano, Mini, Gamma, Delta</td>
<td>Force, Torque</td>
<td>Robotic hand research, robotic surgery research</td>
<td>0 – 40000 with resolution of .0002N for Force and .00001Nm for Torque.</td>
<td>This range of sensors from ATI measures both force and torque in a single sensor.</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.6 Modern force/torque sensors

Force transducers operate on the principle that the electrical characteristics of a material changes as force is exerted on it. Typically the resistance of the material will be used as the measurement medium as it can be easily integrated into a Wheatstone bridge circuit (Figure 1-1Figure 2-4). The Wheatstone bridge circuit becomes unbalanced as the force on the gauge changes which results in a change in output voltage. Therefore the configuration is capable of detecting small changes in force.
Quarter-bridge strain gauge circuit

Tension causes resistance increase

Gauge insensitive to lateral forces

Compression causes resistance decrease

Figure 2-4 A typical strain gauge, electrical resistance changes as force is exerted on the gauge and a typical quarter bridge strain gauge circuit.

Most modern pressure transducers operate on the same principle of piezoresistance. These transducers will typically use resistors, strain gauges or semiconductors as the resistive media and measure the pressure by calculating the difference in pressure between two sides of a diaphragm (Figure 2-5). Pressure transducers are categorised into absolute, gauge and differential transducers based on the reference point used for the calculation. A transducer is an absolute transducer if the reference point is a vacuum, a gauge transducer if the reference point is atmospheric pressure and a differential transducer if it is measuring the pressure at two different points of a system.

Figure 2-5 A pressure transducer measuring a fluid under pressure. As the pressure decreases the diaphragm will deflect changing the resistance.
Torque is defined as the measure of forces required to cause an object to rotate and can be divided into static torque and dynamic torque. Although both static and dynamic torque will be present to some degree in a robotic application, it is the measurement of dynamic torque that is of most interest.

The most common and inexpensive method of measuring dynamic torque is to connect the rotating sensor to an electronic system via a slipring. In its simplest form, the slipring system comprises of a metal brush which rubs off a metal ring as it rotates. This action will result in electrical current being conducted and it is from this that the torque calculation is made. This system is adequate for low speeds, however as the rotational speed increases, the electrical noise from the brush system increases. An alternative to the slipring is the rotary transformer.

A rotary transformer provides a non-contact means of transferring signals to and from the rotating structure. They operate on the same principle as a conventional transformer with the primary and secondary coils positioned so that one can be rotated with respect to the other. Dynamic torque sensors will typically use two rotary transformers. One will transmit the excitation voltage to the strain gauge bridge whilst the other transformer will transfer the torque measurement signal to the non-rotating part of the transducer.

Figure 2-6 A rotary transformer typical circuit diagram and block diagram.
The rotary transformer transducer is also susceptible to noise, is quite fragile and requires bearings for support. It also requires specialised signal conditioning electronics which adds to the cost.

An alternative type of non-contact torque sensor is the infrared (IR) torque sensor. This sensor consists of a strain gauge, a rotating sensor and an optical transmission ring. The strain gauge which is powered by a rotary transformer measures the torque. This torque measurement is then converted from a voltage to a frequency modulated signal. The signal is then transmitted via IR LED's to a stationary receiver which checks the signal for errors and converts it into an analog voltage which can then be processed by a microprocessor. The IR torque sensor is more expensive than the rotary transformer but it is highly reliable and suitable for the measurement of high speeds.

2.2 Sensors for external interaction

The sensors discussed in the previous section are capable of making the robotic system self-aware by providing data about its internal systems and sub-systems, however in order to develop a robotic system that is also aware of its surroundings there are a number of sensors that can be used.

2.2.1 Proximity detection and ultrasonic rangers

Proximity sensors are used to determine the presence (or lack of presence) of nearby objects thus providing a crude type of obstacle avoidance sensor.

2.2.1.1 Principles of proximity detection

The original ultrasonic proximity sensor was a reflective sensor consisting of two transducers with the receiver response being a function of the amplitude of the return
energy. The original ultrasonic proximity sensor has been replaced with a proximity sensor that can return an accurate distance to a detected object. These proximity sensors are also known as ultrasonic rangers, and operate on the principle of \textit{time of flight (TOF)}.

The ultrasonic proximity sensor shown in Figure 2-7 illustrates the principle of TOF operation. The sensor electronics measure the round-trip time required for a pulse of emitted energy to travel to a reflecting object where it is echoed back to a receiver. The sonic transducer shown in Figure 2-7 is capable of transmitting and receiving alternately. It first emits a burst of sonic waves before switching to receive mode to wait for a return signal. If the transmitted sonic burst hits an object, it will be reflected back to the transducer and the distance to the object can be worked out using basic physics. If the transmitted sonic burst fails to hit an object within a certain amount of time there will be no signal returned.

![Figure 2-7 Ultrasonic sensor operation and Devantech sonar ranger](81]

Ultrasonic proximity sensors are used extensively in the automation industry and are at present preferred to photoelectric sensors as a method of position measurement due to their immunity to harsh environmental conditions such as dust, smoke and steam. They are also used extensively in the automotive industry as reversing sensors.

There are other non-contact proximity sensors that could be readily used with robotics applications, but because of their expensive nature they are not always feasible. A majority
of proximity sensors available for the automation industry that could be used for collision avoidance in mobile robotics are either TOF ultrasonic based or laser triangulation sensors.

2.2.1.2 SensComp ultrasonic ranging modules

The SensComp [82] ranging module is a very simple, active TOF device that was originally developed for automatic camera focusing [83] and is the most consistently used ultrasonic ranging module that is on the market, due to its low cost and availability. SensComp currently manufacture two types of ultrasonic transducers (Figure 2-8), electrostatic transducers and piezo transducers as well as a ranging module that is capable of driving all electrostatic transducers.

The electrostatic transducers, the 600, Instrument and 7000 series, operate in a similar fashion to an electrical capacitor. Each electrostatic transducer consists of one fixed aluminium plate, and one movable aluminium plate coated in a film of Kapton™ and gold. When a 50 kHz drive signal is placed between the plates the film on the moveable plate is attracted to the aluminium, thus displacing air and generating a burst of ultrasonic sound. This technology creates an extremely sensitive device that is capable of detecting small objects such as the stem of a flower at a distance of 10.7 metres [84].

The Piezo transducer, the 9000 series, consists of a crystal or ceramic Piezo material bonded to a metal case or a cone [83]. As the Piezo material is excited by a 40 kHz drive signal it expands and contracts, displacing air around the metal case or cone generating a pulse of ultrasonic sound. Piezo transducers are less sensitive than electrostatic transducers and were designed to operate in severe environments and comply with the SAE specification J1455 for heavy duty trucks [83].
The 6500 sonar ranging module (Figure 2-9) contains all the circuitry required to drive all of SensComp electrostatic transducers and can operate in two modes, single-echo mode and multiple-echo mode.

A typical cycle of operation is as follows [83] and the timing diagram can be seen in Figure 2-10.

1) The control circuitry fires an ultrasonic burst that moves at the speed of sound from the transducer at a frequency of 50 kHz. The transducer then waits an indication that transmission has begun.

2) The receiver is blanked for a short period of time to prevent false detection.

3) Returning signals hitting the transducer create a voltage that is in turn fed to a stepped-gain amplifier. Since the signal decreases in strength with distance at an inverse squared proportion, the gain of the amplifier is increased exponentially.
4) Returning echoes that exceed a fixed threshold value are recorded and the
associated distances are calculated from elapsed time.

As can be seen on the timing diagram in Figure 2-10, once $V_{cc}$ is applied a minimum of 5
ms must elapse before the INIT input can be taken high; this is to allow the internal
circuitry to reset and the internal oscillator to stabilise. When INIT is taken high, sixteen
pulses at 49.4 kHz and 400V amplitude excite the transducer with the dc bias of 200V
remaining on the transducer. To eliminate ringing in the transducer, and thus preventing the
ringing being mistaken as the return signal, the receive input is blocked for 2.38ms after the
INIT signal. The receive input can be forced back on at any time during this “blanking
period” by bringing the BINH signal high. After this stage has been completed, single-echo
mode and multiple echo mode differ slightly in their operation.

In single-echo mode, after the “blanking period”, the ranger waits for the returning signal
which is amplified once received before being represented as an echo pulse with a pulse
width proportional to the distance of the object from the transducer. The process can be
repeated by clocking the INIT pin for one cycle.

If there is more than one target, like a cluttered office environment the chances of multiple
echoes are increased. After the “blanking period” the ranger awaits a return signal, once the
first return signal arrives and causes the ECHO output to go high, the blanking (BLNK)
input must be from low to high to low in order to reset the ECHO output before the next
return signal. This blanking reset must be at least 0.44ms in duration to allow for internal
delay times.
Figure 2-10 Timing diagrams for the 6500-Series ranging module executing a multiple-echo-mode cycle with blanking input [85]

The ROBART series of robot prototypes started in 1980 was designed in support of the MDARS program by the US Navy. The initial prototype ROBART did not use SensComp ultrasonic transducers for obstacle avoidance, but the preceding prototypes ROBART II and ROBART III do. ROBART II (Figure 2-11) uses a fixed array of 11 transducers on the front of its body to provide distance information, with a ring of 24 additional transducers at a spacing of 15° mounted below the robot’s head to gather range information for position estimation. ROBART II has a final ranging unit located on the rotating head assembly allowing for distance measurements to be made in different directions. ROBART III (Figure 2-11) is equipped with a video camera, allowing the number of SensComp transducers to be reduced. It uses additional transducers to help with the control of its weapons.
The SensComp range of transducers and ultrasonic rangers do have a few shortcomings [87], the maximum range of 10.7 metres and minimum of 26cm is not really useful for use with indoor mobile robots that are required to work in tight spaces. The transducer is also quite large, it consumes 2.5 Amps during an ultrasonic burst and it has a quiescent current of 150mA which is extremely high for such a small device.

2.2.1.1 Devantech ultrasonic ranger

The Devantech SRF04 shown in Figure 2-12 was designed to improve on the shortcomings of the SensComp transducer discussed in the last section whilst the SRF08 was designed as a progression from the SRF04. Both of these sensors are used very frequently in mobile robot applications to provide spatial awareness.

The SRF04 is just as easy to use as the SensComp ultrasonic transducer. It is also a more compact package that can be triggered with a short pulse returning an echo pulse on pin 2.
The SRF08 shown in Figure 2-13 was developed to improve the features of the SRF04 and incorporates a light sensor, provides a 6 metre detection range which accommodates the receipt of multiple echoes and uses the I^2C standard interface [88].

While the SRF04 is limited to a 3 metre detection range so as to prevent the on-board op-amps from saturating when the sensor is operating at close range, the SRF08 improves on this limitation by adding a digital potentiometer capable of varying the gain of the op-amp as the range increases. If the op-amps saturate it is almost impossible for the electronics to distinguish a real echo from cross-coupling between the transmitter and receiver [88]. The addition of the digital potentiometer allows a higher overall gain and as a result a better range; during prototyping of the SRF08 the designers noted that a typical range of 6 metres was the most common, with the SRF08 detecting large objects at up to 11 metres [88]. The sensor was detecting “small close by anomalies in the floor” [88] at a resolution of 11 metres, so the final SRF08 has a maximum range of 6 metres.

The processor used on the SRF04, the PIC12C508 is limited by a single timer and as a result a 36 millisecond timeout for non-detection was imposed. For the SRF08 the processor has been upgraded to the PIC16F872, an 8-bit CMOS FLASH microcontroller, a 10-bit A/D, 3 timers and a capture/compare PWM module [88].

...
The SRF08 uses the I²C bus interface allowing multiple sensors to be controlled using 2 I/O lines, unlike the SRF04 where each SRF04 sensor requires 2 I/O lines on the controlling processor. That means that if there is a requirement to use an array of 6 sensors on a mobile robot, the SRF04 needs 12 I/O lines while the SRF08 only requires 2 I/O lines.

The current consumption of the SRF08 is greatly improved from that of the SRF04. The SRF04 requires a maximum current of 50 milliamps and even though this is an improvement on the SensComp sensor it is still quite high. The SRF08 has a quiescent current of 3mA, with a normal current of 15mA, and automatically goes into standby after it has completed its ranging.

![Image of Devantech SRF08 ultrasonic ranger](image)

**Figure 2-13 Devantech SRF08 ultrasonic ranger**

The SRF08 is suitable for connection to any microcontroller with I²C bus capability; the OOPic, Basic Stamp BS2p and a host of other microcontrollers.

### 2.2.2 Other non-contact proximity sensors and rangers

Along with the ultrasonic sensors discussed in the last section there are two other sensor products that have been used in the past for proximity detection and range finding; the Sharp IR range of range finders and proximity detectors, and the SICK laser range finder.
2.2.2.1 Laser based TOF systems

Laser based TOF ranging systems, known as light detection and radar (LIDAR) or laser radar, first appeared in work performed at the JPL California in the 1970s. Since then they have become a popular sensor to prevent collisions, to measure the size of objects, as a road vehicle classifier and have almost replaced radar as a form of speed detection by the authorities. A table of common LIDAR devices is included in Table 2.7.

Laser range finders work by transmitting short pulses of light via an infrared laser beam in the direction of an object. The time required for a given pulse to reflect off the object and return to the receiver is measured, and the distance to the object is calculated using 
\[ d = \frac{vt}{2} \]

where \( v \) is the speed of light in m/s, \( t \) is the time taken to reflect from the object, and \( d \) is the distance to the object.

SICK Inc. has developed two Laser Measurement Systems that have been used as a distance measurement sensor for mobile robots; the LMS200 designed for indoor use, and the LMS220 designed for outdoor use.

Figure 2-14 SICK Inc. LMS200 (left) and LMS220 (right), the only difference between the two sensors apart from the physical dimensions is the enclosure rating. The LMS200 has an enclosure rating of IP65, while the LMS220 has an enclosure rating of IP67 [89].

Apart from their enclosure ratings both range finders have the same operating principle. A pulsed laser beam is deflected by an internal rotating mirror so as to create a fan shaped
scan as shown in Figure 2-15. An angular sensor on the internal rotating mirror gives the precise direction of the laser pulse.

If the scan meets an object it is reflected back and collected by the sensors receiver. The distance to the object from the scanner is directly proportional to the time between the emission and reception of the pulse, with the contour of the target area being determined from the sequence of pulses received. This data is transferred in real time over a serial interface (either RS232 or RS422) so the surrounding areas can be analysed.

A laser measurement interface LMI200 can be purchased to allow all of the features of the LMS200/220 to be easily used. It includes an onboard CPU, 2 RS422 LMS sensor interface, 1 configurable RS232/RS422 interface, a RS232 interface with an internal CPU debug interface and 1 RS485 interface with 1 CAN bus.

Figure 2-15 Operating principle of LMS200 and LMS220 [89]

The Robots and Artificial Intelligence (RIA) group at the Laboratory analysis and architecture of systems (LAAS), Toulouse, France have used the SICK LMS200 for obstacle avoidance [90].
Table 2.7 Modern Laser TOF Ranging Systems

2.2.2.2 SHARP infra-red range finders

There are currently seven types of Sharp IR rangefinders available from Acroname [85]; the GP2D10, GP2D02, GP2D05, GP2Y0D02YK, GP2Y0A02YK, GP2D12 and GP2D15, and while they all operate using triangulation to measure distance, their ranges and outputs are different as can be seen in the visual comparison in Figure 2-16 and Figure 2-17.

Figure 2-16 Distance comparison chart for Sharp range of IR sensors [85]

<table>
<thead>
<tr>
<th>IR Ranger</th>
<th>Output</th>
<th>Min. Distance (cm)</th>
<th>Max. Distance (cm)</th>
<th>On Current (mA)</th>
<th>Off Current (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP2D02</td>
<td>Serial</td>
<td>10</td>
<td>80</td>
<td>~22</td>
<td>~3</td>
</tr>
<tr>
<td>GP2D05</td>
<td>Digital</td>
<td>-</td>
<td>24 (Fixed)</td>
<td>~10</td>
<td>~3</td>
</tr>
<tr>
<td>GP2D12</td>
<td>Analog</td>
<td>10</td>
<td>80</td>
<td>~33</td>
<td>Always On</td>
</tr>
<tr>
<td>GP2D15</td>
<td>Digital</td>
<td>-</td>
<td>24±3 (Fixed)</td>
<td>~33</td>
<td>Always On</td>
</tr>
<tr>
<td>GP2D120</td>
<td>Analog</td>
<td>4</td>
<td>30</td>
<td>~33</td>
<td>Always On</td>
</tr>
<tr>
<td>GP2Y0A02YK</td>
<td>Analog</td>
<td>20</td>
<td>150</td>
<td>~33</td>
<td>Always On</td>
</tr>
</tbody>
</table>

Figure 2-17 Comparison table for Sharp range of IR sensors
Because of their low cost, availability and ease of integration to projects, the Sharp IR ranging sensor has been used in many applications for proximity detection. Students at the University of Connecticut [91] have used the GP2D12 in conjunction with the GP2Y0A02YK to create a collision avoidance prototype suitable for motorised vehicles, Roberto Montane at the Intelligent Machine Design Lab, University of Florida [92] has used the GP2D12 for collision avoidance in conjunction with the GP2D120 for wall following in his autonomous vacuum cleaner prototype, while Justin Osborn [93] and Carnegie Mellon University have developed a Palm pilot robot that can be interfaced to the GP2D12 IR ranger. There are no other sensors on the market that come close to the Sharp range of sensors in terms of cost, accuracy and compatibility.

2.2.3 Tactile sensors

Tactile sensors are designed to determine if there is contact between an object and the sensor and this feedback is very useful for robotic manipulators. They can be used to detect objects that are in the way of the robot or as an input device for human-robot interaction.

The most basic form of tactile sensor is a switch connected to the digital input port of a microcontroller. When the switch is pressed the microcontroller will see a binary 0 or 1 depending on the surrounding circuit and react accordingly. Tactile sensors are also designed using capacitive, resistive and optical technologies, with capacitive being the most common touch technology being used in consumer electronics and no doubt in the future in the field of robotics. Table 2.8 shows a comparison of some touch sensors and their typical applications.
Table 2.8 Modern touch sensors

2.2.4 Position sensors

In order for an autonomous robot to self-localise it needs to keep track of its current location and orientation. Table 2.9 lists the most common type of position sensor and their key characteristics.

For robots whose environment is predominately indoors the compass is the most effective device for providing the robotic system with this information. The CMPS01 compass from Devantech [94] uses a Philips KMZ51 magnetic field sensor that is sensitive enough to detect the earth’s magnetic field; it has a resolution of 0.1° and an accuracy of 3° to 4° once calibrated [95]. Due to its low power consumption, typically in the order of 20mA at 5V, it is easily interfaced with most microcontroller hardware. The sensor also has a choice of either a pulse width modulated (PWM) output or an output via an FC interface. It should be calibrated before use with an accurate orienteering compass and the circuit shown in Figure 2-18.
Figure 2-18 Deventech compass sensor and calibration circuit

The next generation of digital compass similar to those manufactured by PNI Sensor corporation [96] which are based on magneto-inductive technology remove the need for complex circuitry and are capable of sensing differences in the earth’s magnetic field accurately by removing magnetic fields from other sources [97] [98]. They have an accuracy of less than 2°, a resolution of 0.1° and have an inbuilt processor. The CompassPoint Prime digital compass has been used successfully in iRobot’s explosive ordnance disposal robot to navigate reliably in extreme environments. Its low cost, RS232 interface and wide range of operating temperatures make it an ideal sensor for such robots.

For outdoor autonomous robots, a global positioning system (GPS) can easily be used to provide location information to the robot via a serial communication interface. The GPS consists of a radio-navigation system formed by a constellation of 24 satellites located 10,900 nautical miles from earth which orbit the earth once every 12 hours and five ground stations which track the satellites.

GPS devices are commonplace in today’s world but the GPS receiver chips are heavily integrated into the ‘SatNav’ as we know them and not easily modified for use in robotics. There are however a number of readily available GPS receivers that can easily be integrated
into a robotic system to provide the robot with real-time location and elevation information. One such group of suitable sensors come from the GARMIN GPS 16/17/18 [99] family. These receivers are ideal for use as a navigation sensor for outdoor autonomous robotic systems for a number of reasons;

- They are IPx7 rated, meaning that there will not be any water ingress to the unit even if immersed in water.
- They have a wide range of operating voltages from 6V to 40V unregulated DC meaning that they can be accommodated in all robotic systems regardless of scale.
- They have an acquisition time of 45 seconds on first turn-on, with reacquisition of 2 seconds if the satellite coverage is lost for any reason.
- They are accurate to within 15 metres worst case; but typically 3 metres.
- The interface is RS232 with a user selectable baud rate meaning that it can be integrated with most microcontrollers/microprocessors.

![Figure 2-19 GARMIN GPS 16/17/18 and interface circuit](image)

The software interface to the GPS device is based on the National marine electronics association (NMEA) 0183 ASCII interface specification and all of the detailed protocol information is outlined in the standards which can be obtained from NMEA [100]. By
default the GARMIN GPS will output seven strings, GPRMC, GPGGA, GPGSA, GPGSV, PGRME, PBRMB, PGRMM and PGRMT. However in reality only GPRMC which contains configuration information such as the type of GPS to use and baud rate and GPGGA which contains information such as UTC time, Longitude, Latitude, GPS quality and dilution of precision are required to calculate an accurate position.

There is a major drawback however with using GPS in that it requires line of sight to at least 4 satellites to determine its location, orientation and direction so it struggles to operate reliably in street canyons or in areas with heavy foliage and although the prototype system that is described in the course of this work is not intended to operate in an outdoor environment these limitations must be considered for future designs.

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Manufacturer Part Number</th>
<th>Sensor Type</th>
<th>Typical Applications</th>
<th>Communications Interface</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honeywell</td>
<td>HMR3300</td>
<td>3-Axis Compass</td>
<td>Navigation, guidance systems.</td>
<td>UART, SPI</td>
<td>1 RMS</td>
</tr>
<tr>
<td>Honeywell</td>
<td>HMR3200</td>
<td>2-Axis Compass</td>
<td>Navigation, guidance systems.</td>
<td>UART, SPI</td>
<td>1 RMS</td>
</tr>
<tr>
<td>Reneseas</td>
<td>HM55B</td>
<td>2-Axis Compass</td>
<td>Mobile robotics, automotive electronic compass.</td>
<td>Asynchronous</td>
<td>N/A</td>
</tr>
<tr>
<td>Navsync</td>
<td>CW20</td>
<td>GPS Receiver</td>
<td>Navigation, asset tracking, security.</td>
<td>UART</td>
<td>95% Confidence of 2m</td>
</tr>
<tr>
<td>Garmin</td>
<td>GPS17X</td>
<td>GPS Receiver</td>
<td>Marine navigation.</td>
<td>RS422</td>
<td>1 RMS</td>
</tr>
</tbody>
</table>

Table 2.9 Modern position sensors

### 2.2.5 Light sensors

The most basic form of light sensor is the light dependent resistor (LDR) whose resistance depends on the quantity of light falling on it. It can be used to great effect in autonomous robots for avoiding dark areas in a room and has been used successfully by teams in
robotics competitions where tasks range from following a line to searching for a ball which is emitting infra-red light. Some common light sensors, their key characteristics and typical application are listed in Table 2.10.

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Manufacturer Part Number</th>
<th>Sensor Type</th>
<th>Typical Applications</th>
<th>Peak Sensitivity (nm)</th>
<th>Luminance Range (lx)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHARP</td>
<td>GAI A2S100LY</td>
<td>Ambient Light Sensor</td>
<td>Office automation, home appliances.</td>
<td>500</td>
<td>10 – 10000</td>
</tr>
<tr>
<td>TAOS</td>
<td>S3931</td>
<td>Light to Digital Converter</td>
<td>Electronic appliances.</td>
<td>640</td>
<td>10 – 10000</td>
</tr>
<tr>
<td>SILONEX</td>
<td>NSL 4960</td>
<td>LDR</td>
<td>Mobile robotics, automotive electronic compass.</td>
<td>515 – 690</td>
<td>N/A</td>
</tr>
<tr>
<td>Avago</td>
<td>APDS-9007</td>
<td>Ambient Light Photo sensor</td>
<td>Consumer devices, sunlight harvesting.</td>
<td>560</td>
<td>3 – 70000</td>
</tr>
</tbody>
</table>

Table 2.10 Modern light sensors

2.2.6 Sound sensors

Sound can be captured and analysed using a microphone with some complementary hardware and software for human-robot interaction (HRI) or for locating distress calls when used with a search and rescue robot. Table 2.11 lists some sound sensors that can easily be integrated into a robotic system capable of detecting sound from voice levels (30dB – 70dB) to higher levels such as from an airplane (70dB – 110dB).

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Manufacturer Number</th>
<th>Part</th>
<th>Sensor Type</th>
<th>Typical Applications</th>
<th>Range</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fourier</td>
<td>DT320</td>
<td></td>
<td>Sound Level Sensor</td>
<td>Investigating environmental noise.</td>
<td>45dB – 110dB</td>
<td>±3dB</td>
</tr>
<tr>
<td>Pasco</td>
<td>PS-2109</td>
<td></td>
<td>Sound Level Sensor</td>
<td>Measuring environmental noise.</td>
<td>30dB – 110dB</td>
<td>±2dB</td>
</tr>
<tr>
<td>Rapid</td>
<td>85-2612</td>
<td></td>
<td>Sound Level Sensor</td>
<td>Measuring noise.</td>
<td>50dB – 100dB</td>
<td>±5dB</td>
</tr>
</tbody>
</table>

Table 2.11 Modern sound sensors
2.2.7 Vision sensors

The two most common form of image capture sensors are charge coupled device (CCD) and complementary metal oxide semiconductor (CMOS) sensors each with their own strengths and weaknesses thus giving advantages in different applications [101]. Both CMOS and CCD sensors convert light into an electrical charge before converting this charge into an electrical signal; CMOS sensors convert the charge to digital signals whilst CCD sensors convert the charges to analogue signals.

![Combined diagram showing CCD sensor and integration of additional circuitry in CMOS sensor](image)

CCD and CMOS sensors were both invented the late 1960s and early 1970s. However it was the CCD type sensor that achieved early success as lithography technology was new and CCD sensors could produce higher quality images. It was not until the 1990s following major advancements lithography technology that the CMOS sensor caught up with the CCD sensor. Early predictions as to the future of CMOS sensors indicated that it would be the superior technology. However this has not proved to be the case. e.g. it was expected that CMOS sensors could be designed with all electronics integrated on a single chip, but this has only materialised in some higher level sensors due to the high development costs.

...
The CMOS sensor and CCD sensor both have their advantages and disadvantages (a comparison is included in Table 2.12) but it is really only engineers and scientists who are specialising in the area of vision systems that must carefully consider these when choosing the type of sensor required; for engineers who are involved in developing an aware robotic system they have the luxury of considering the off the shelf product based on the product specifications.

<table>
<thead>
<tr>
<th>Feature</th>
<th>CCD Sensor</th>
<th>CMOS Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal out of Chip</td>
<td>Voltage (Analog)</td>
<td>Bits (Digital)</td>
</tr>
<tr>
<td>System Complexity</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Sensor Complexity</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Camera Components</td>
<td>Sensor &amp; Multiple Support Chips &amp; Lens</td>
<td>Sensor + Lens is possible, but usually require support chips.</td>
</tr>
<tr>
<td>R&amp;D Cost</td>
<td>Lower</td>
<td>Higher</td>
</tr>
<tr>
<td>Uniformity</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Dynamic Range</td>
<td>High</td>
<td>Moderate</td>
</tr>
<tr>
<td>Responsivity</td>
<td>Moderate</td>
<td>Better than CCD</td>
</tr>
</tbody>
</table>

Table 2.12 CCD vs. CMOS comparison

The CMUcam2 from Carnegie Mellon University (CMU) [102] integrates a CMOS sensor with a maximum resolution of 160 x 255 pixels on a printed circuit board assembly with a microcontroller which contains pre-programmed firmware capable of tracking motion, user defined blobs and controlling 5 servo motors. This camera operates in a voltage range of 6v to 15V DC, drawing a maximum of 200mA interfacing with the microcontroller/microprocessor via an RS232 serial interface. CMU are also leading a project CMUcam3 [103] whose goal is to create an embedded colour vision platform based on CMUcam3 which is a 352x288 resolution camera which is programmable using C and includes basic image manipulation algorithms.
Both CMOS and CCD sensors have been integrated into cameras, webcams and mobile phones and are ideal for use in robotics due to their small size and ease of interface with external hardware. Most of them come with USB or RS232 serial interface. However it is the harnessing of these images and converting them into useful information that is proving the most challenging and this is why an entire branch of study is dedicated to this topic.

2.2.8 Environmental sensors

The major advantage of designing a robot is that it can be placed in a situation and environment where it is not possible to put a human, thus the integration of sensors capable of detecting substances that a human is unable to or substances that are hazardous to humans is important. There are a large number of sensors that are used in industry that can be implemented on an autonomous robot, sensors that detect combustible gases such as hydrogen or propane, toxic gases such as carbon monoxide and ammonia, or indoor pollutants such as carbon dioxide. The ammonia sensor in Figure 2-21 shows how easily it could be integrated into an environmentally aware robot.

![Figure 2-21 Figaro TGS 826 Ammonia sensor and interface diagram](104)

A majority of these environmental sensors are not readily available for research and development purposes as the manufacturers are more intent on mass production for OEM
buyers, but with advances in sensor research moving rapidly it will not be long until they are.

2.2.8.1 Humidity sensors

Honeywell produce a range of low power, high accuracy relative humidity and temperature sensors (Figure 2-22) that can be integrated with any controller but are only sold to OEM manufacturers.

Figure 2-22 HIH-3610 Honeywell humidity sensor (a) and a HEL-700 platinum RTD temperature sensor

The relative humidity sensors made by Honeywell (HIH-3610 and HIH-3602) use a thermoset polymer, a three layer capacitance structure with platinum electrodes and an integrated voltage output signal conditioner [105]. The sensor detects when water vapour equilibrates with the surrounding gas in the capacitor’s dielectric layer and outputs a linear voltage accordingly. The porous platinum layer and the polymer layer protect the sensor from external influences and contaminants [105]. The sensor is affected by ambient temperature but this can be offset with a platinum temperature sensor and some external electronics [105].
2.2.8.2 Gaseous sensors

A majority of chemical and biological sensors are designed for use within a military environment and although "kits" and "devices" containing these sensors can be purchased commercially, it is extremely difficult to obtain the sensor alone. These "stand-alone" chemical and biological sensors would add significantly to the "usefulness" of mobile robots. Sensors to detect potentially fatal gases in mines, sewers and pharmaceutical plants could be attached to a mobile robot that is programmed to cause an alert if there are dangerous levels of chemical or biological gases present in the environment hence giving the robotic system enhanced awareness of the environment.

Applied Sensor [106] based in Sweden, Germany and the USA have designed and built a number of very useful sensors that could be easily integrated into a mobile robot. They have divided their products into field effect (FE) sensors, metal oxide semiconductor (MOS) sensors and quartz microbalance (QMB) sensors.

Applied Sensor manufacture two hydrogen detection sensors based on the field effect; the AS-FHH-400 high sensitivity sensor and the AS-FHH-450, a less sensitive device (Figure 2-24).
Hydrogen is an odourless, tasteless, colourless and non poisonous gas \[106\] under normal environmental conditions making it impossible for a human to detect it. Hydrogen is a highly flammable gas with a high flammability range and it only takes a minimal amount of energy to ignite it and if a situation arises where there is a build up of hydrogen gas it could prove fatal. Hydrogen is used to produce fuel cells so the effective early detection of hydrogen gas leaks and the build up of hydrogen gas pockets is critical. By utilising such sensors (AS-FHH-400 and AS-FHH-450), the formation of a hydrogen gas pocket could be detected thus preventing the possibility of an explosion.

Both the AS-FHH-400 and the AS-FHH-450 are quite sensitive to hydrogen in air and are capable of detecting 10ppm to 2000ppm and 500ppm\(^1\) to 5% hydrogen in air respectively. They are insensitive to N\(_2\), Ar, C\(_3\)H\(_8\), CO, CO\(_2\), and NO\(_x\) when measuring hydrogen and relative humidity has no effect on the sensor accuracy. They have an operating temperature of 140°C at an ambient temperature of between -40 °C to 120 °C. They both have a typical operation of 2.2 V and a power consumption of 80mW and have diode like characteristics. Both the heater and the sensor should be run with a constant current source as the output

\[^1\] 1 ppm = 1 mg/l in gases and 1 mg/kg in solids
response is measured as the voltage drop over the device operated at a constant current [107] (Figure 2-25).

Figure 2-25 Application circuit for hydrogen detection sensor from Applied Sensor

A field effect is generated by gases interacting between MOSFET devices and catalytic metals and Applied Sensor utilise this effect to create their hydrogen sensor. By using a catalytic metal such as silicon carbide rather than silicon allows the sensor to operate at a higher temperature thus increasing the response time even in highly contaminated environments. The electronic equivalent of the FE sensor can be seen in Figure 2-26. The current through the MOSFET is controlled by the gate voltage, which is in turn controlled by the gas molecules interacting with the catalytic metal. The level of gas molecules will affect the gate voltage and therefore affect the current flowing through the MOSFET device, with the gas response being recorded as the voltage change in the sensor signal (Figure 2-26).
The MOS range of sensors manufactured by Applied Sensor (Figure 2-29) measure the resistance in a sensing layer that comprises of an alumina substrate and a micromachined silicon substrate. The resistance of this sensing layer changes when gas molecules react with the surface; the resistance increases for oxidising gases such as NO$_3$ and O$_3$, and decreases for reducing gases such as H$_2$ and CO. The electronic equivalent and a typical response curve can be seen in Figure 2-27.

Applied Sensor currently manufacture 3 MOS based sensors, the AS-MLK, AS-MLN and the AS-MLC capable of detecting methane, nitrogen dioxide and carbon monoxide.
respectively, potentially three of the most dangerous gases to humans. The advantage of these sensors is that they can be easily integrated with any controller using a simple voltage divider circuit. The properties of these gases, make dangerous levels of these gases almost undetectable to humans, so if the MOS based sensors were to be implemented on a mobile robot, it would make it very useful indeed.

Carbon monoxide is a highly toxic gas, and if inhaled it binds itself to the haemoglobin in the blood thus reducing the oxygen carrying capability of the blood. It is a colourless, tasteless, odourless gas with a flammable range of 12.5% to 74%. With varying degrees of exposure to carbon monoxide, health effects on a human range from a temporal headache with two to three hours exposure at a 400ppm concentration to death in two hours of exposure at a 1600ppm concentration with deafness being reported after severe carbon monoxide intoxication [106] (Figure 2-28). With an acceptable level of carbon monoxide suggested to be in the region of 25ppm, the AS-MLC with a sensitivity of 0.5ppm to 500ppm could be attached to a mobile robot and programmed to create a signal if carbon monoxide levels increase.

![Figure 2-28 Effects of exposure to carbon monoxide](108)
Nitrogen dioxide has been connected with death resulting from fluid build-up in the lungs. It is an insoluble red-brown gas allowing it to penetrate to the lower respiratory tract where it can cause death and it has a pungent, acrid odour. It is detectable at 0.04ppm to 5ppm concentration with irritation to the nose and throat, however if concentration is increased above 80ppm it is highly dangerous. It is produced by diesel exhausts or by burning nitrate explosives. The AS-MLN has a sensitivity of 0.1ppm to 2ppm (the acceptable maximum concentration of nitrogen dioxide in a working environment) so it could be implemented on a mobile robot to monitor nitrogen dioxide levels or in fact detect nitrogen dioxide leaks.

Methane is a colourless but flammable gas and occurs as free gas in coal beds and its presence is of particular concern in underground coal operations where its concentration and sources of ignition must be controlled. There have been many explosions in coal mines over the years killing hundreds of workers. A mobile robot could be used to monitor levels of methane in different parts of the mine if equipped with an AS-MLK sensor.

![Figure 2-29 Applied Sensor MOS sensors (a) MOS micro sensor and (b) a thick-film sensor](image)

The final type of sensor produced by Applied Sensor is designed to detect volatile organic compounds by measuring the frequency of polymer coated quartz crystals with the sensitivity and selectivity determined by the type of polymer coating. The sensors from the
quartz microbalance (QMB) range (Figure 2-30) are capable of detecting a large range of gases, with a high reproducibility and low power consumption. However, it would not really be suitable for use with a mobile robot as it is susceptible to relative humidity and temperature fluctuations.

![Figure 2-30 Applied Sensor QMB Sensor](image)

**2.2.8.3 Flame detection sensors**

The UVTron bulb (Figure 2-31) is a very simple yet very sensitive device and when used in conjunction with the Hamamatsu drive circuit (Figure 2-31) it can detect a cigarette lighter flame from 5 metres.

![Figure 2-31 UVTron bulb from Hamamatsu](image)
When the cathode is exposed to ultraviolet rays from the flame source, photoelectrons are emitted from the cathode by the photoelectric effect and then accelerated towards the anode by the electric field. As the applied voltage increases the electric field becomes stronger and the kinetic energy of the electrons becomes large enough to ionise the molecules of the gas enclosed in the tube by collision. The electrons generated by the ionisation are accelerated, enabling them to ionise other molecules before reaching the anode. Positive ions are accelerated toward the anode and collide with it creating secondary electrons. This avalanche process causes a large current between the electrodes and discharge takes place. Once the tube is discharged, it is refilled with electrons and ions with the voltage drop between the cathode and anode being reduced considerably. This state will remain without ever lowering the anode voltage to below a saturation point. Table 2.13 lists some state of the art environmental sensors that could be used in a robotics application to provide environmental awareness.

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Manufacturer Part Number</th>
<th>Sensor Type</th>
<th>Typical Applications</th>
<th>Output</th>
<th>Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applied Sensor</td>
<td>AS-MLN</td>
<td>Nitrogen Dioxide</td>
<td>Leak detection.</td>
<td>Resistance changes with Nitrogen concentration.</td>
<td>0.1 – 2ppm</td>
</tr>
<tr>
<td>Applied Sensor</td>
<td>AS-MLC</td>
<td>Carbon Monoxide</td>
<td>Leak detection.</td>
<td>Resistance changes with carbon monoxide concentration.</td>
<td>0.5 – 5ppm</td>
</tr>
<tr>
<td>Applied Sensor</td>
<td>HSS-440</td>
<td>Hydrogen</td>
<td>Leak detection.</td>
<td>CAN BUS interface.</td>
<td>0 – 4.4 &amp; Hydrogen in Air.</td>
</tr>
<tr>
<td>Honeywell</td>
<td>HIH-3610</td>
<td>Humidity Sensor</td>
<td>Battery powered systems, refrigeration.</td>
<td>Analog output.</td>
<td>±0.5%</td>
</tr>
<tr>
<td>Figaro</td>
<td>TGS826</td>
<td>Ammonia</td>
<td>Ventilation control, leak detection.</td>
<td>Resistance changes with Ammonia concentration.</td>
<td>30 – 300ppm</td>
</tr>
</tbody>
</table>

Table 2.13 Modern environmental sensors
2.3 **Important sensor characteristics**

When deciding on a sensor for integration into a robotic system, there are a number of key characteristics which must be taken into account; how accurate does the robot need it to be? How quickly does the robot need this information? What size does the sensor need to be? What interface does it have? This section discusses the key characteristics of sensors which must be taken into account when deciding on a sensor for a robotic application.

### 2.3.1 Resolution

The resolution of a sensor is the smallest measurement that it can reliably measure and as one can understand is one of the most important sensor characteristics.

Medical robots that are used in surgery for example have touch sensors with a resolution equivalent to human touch, typically 40 micrometres [109], whilst resistive touch sensors used in commercial touch screen interfaces do not have or do not require a device with such a high resolution, typically 90 micrometres [110].

### 2.3.2 Sensitivity

The sensitivity of a sensor refers to how much the output of the sensor changes with respect to the object which it is measuring, for example the sensitivity of a K type thermocouple is typically 41µV/°C. This characteristic must be considered carefully when choosing the sensor for the robot as it will impact the level of awareness a robot will have.

### 2.3.3 Linearity

The linearity of a sensor refers to the extent to which the plot of the actual measured values from the sensor differs from the ideal plot and will generally take into account external environmental factors such as electrical noise, temperature and vibration.
It is important to choose a sensor whose measured values are as close to the ideal curve under the expected normal operation conditions. Consider for example a robot that was designed to be used in an environment where 125 kHz RFID tags were being used. If the sensor was 50% non linear when subject to RF noise with a frequency of 125 kHz then the information that the sensor is feeding back to the microcontroller will not be accurate.

2.3.4 Range

The range of a sensor is the difference between the maximum value which a sensor can measure and the minimum value which a sensor can measure. It should also be considered when choosing a sensor. There is not much point in choosing an ultrasonic sensor with a range of 15m when the design specifications require a range of 20m or a T type thermocouple when the normal operating conditions are going to be 500°C.

2.3.5 Response time

The sensor’s response time defines the speed at which a sensor responds to a change in the properties of the object it is measuring. It is extremely important for safety critical sensors such as light curtains around an automated manipulator; if the light curtain is broken the information needs to get back to the microcontroller or a safety relay in real time so the manipulator can be shut down. When using non mission-critical sensors where the parameters are unlikely to change rapidly the response time of the sensor is not critical.

2.3.6 Reliability

Sensor manufacturers should be able to provide reliability information on their sensors if it is not contained on their datasheets. They will generally run a number of extended accelerated tests as discussed in Chapter 1 of this document to determine the mean time...
between failures (MTBF) of their sensors. This characteristic is extremely important when designing robots that are not easily maintained such as the space exploration robots or robots that are critical to life such as medical robots the type of which are discussed in Chapter 1.

2.3.7 Size

The sensor’s size is an important factor to consider when choosing a sensor for a robot. It must be large enough to detect and measure its intended target but small enough not to increase the size of the robot to a point where it is un-useable.

2.3.8 Accuracy and repeatability

The accuracy of a sensor refers to the accuracy of the values which a sensor returns to the robotic system once it is calibrated and is dependent on the construction of the sensor and the accuracy of the calibration equipment. There are two types of errors that affect accuracy; fixed errors and random errors. Fixed errors can be introduced to a system by restrictions on the interfacing hardware, for example a microcontroller might only have 1 byte available for storing temperature so the temperature will always be a whole number and thus the accuracy will be ±0.5°C. This might be acceptable for measuring the temperature of the air, but not if an accurate measurement of the temperature of a human body is required. Random errors are inconsistent errors that happen over the course of time, examples of which are electrical noise and thermal drift of electronic components. Repeatability refers to the accuracy of the sensor output when a given input condition is applied to the sensors a number of times over a given time period and is usually determined by calibration of the sensor using known repeated inputs.
2.3.9 Hardware – software interface

The hardware and software interface to the sensor is equally important as the sensor properties previously discussed. The sensor should have an interface that is easily integrated into the robotic system: Serial, USB, PC or some other form of recognised interface are best as they prevent the need for the implementation of a new protocol which can sometimes prove problematic from both a hardware and software point of view.

2.4 Concluding remarks

This chapter described in detail the most modern and applicable sensors required for developing an aware robotic test platform from cost effective ultrasonic rangers to advanced gaseous sensors. It provides a comparison of the sensors that are most readily available to a system designer and discusses the most important characteristics to take into consideration when designing such an aware system. Taking this into account, Chapter 3 describes the design options available for the development of an aware robotic prototype platform.
Chapter 3 Design options

This chapter discusses the different general design options for implementing awareness strategies and the implications thereof. The specific design option chosen for the prototype platform constructed and evaluated in the work is presented.

3.1 Mobile vs. Manipulator

Without a movable chassis the system could not be considered as a robot, merely a sensory system with a powerful information processor. This resulted in two options being considered, firstly a robotic arm or manipulator designed for pick and place and secondly a mobile robot. Whilst robotic arms are generally non-aware (as discussed in Chapter 1), relying on safety barriers and procedures to avoid harm to humans there is scope for adding awareness. Such scope is limited and it was decided that the design of the robotic prototype platform should be completed on a mobile robot as such a platform would provide a greater opportunity for the development of awareness strategies.

3.2 Distributed vs. Centralised system

As discussed in Chapter 1 the information processing required in developing a robot capable of demonstrating awareness is quite intensive. The requirement for this processing power impacts the design options available to a robotic prototype platform. Research groups for the most part divide their attention between distributed systems e.g. Dollarhide et al. [111], Yasuda [112], Heung-Soo et al. [113] and Premvuti et al. [114] and centralised systems e.g. Bekey et al. [115] Selekwa et al. [116] and Lee et al. [117].
3.2.1 Distributed system

A distributed robotic system consists of multiple autonomous robots each capable of executing tasks whilst communicating with each other constantly over a communications network. The advantage of this type of system is that each robot can be configured to complete a different task hence increasing the efficiency of the system. e.g. in a search and rescue context a team of 6 robots could be released into an environment where three robots are dedicated to locating humans and three robots are dedicated to determining if there are any dangers to human life present in the environment.

The drawback of this type of system is that the awareness algorithms are processed for the most part on the mobile robot hardware. This means that the requirement for increase in processing power will dictate the size of the robot, both in software and hardware which in turn places a constraint on the level of awareness a robot in this system is capable of.

3.2.2 Centralised system

On the contrary, in a centralised system a majority of the information processing is done at a central point. The advantage of this type of system is twofold. Firstly it allows the robot’s design to be rudimentary with only very limited processing capabilities. e.g. the robot hardware could be limited to an array of sensors and a communications interface. The robot software could be designed to simply read the sensors, send it over the communications interface and wait for instruction from the central processing unit.

Secondly, by utilising a central processing unit in this manner the system can be expanded without the size constraints that must be considered in a distributed system.
3.2.3 Proposed system design

The proposed system design is one which is a combination of a distributed and centralised system. It is envisaged that the system consist of mobile robots capable of maintaining autonomy if communication is lost and a central processing unit on which more complex awareness strategies can be developed. The mobile robots are to remain autonomous if communication is lost by equipping them with enough processing power for implementing data mining and obstacle avoidance algorithms. The proposed system diagram is shown in Figure 3-1.

Figure 3-1 Proposed system diagram
3.3 Hardware design options

Keeping the proposed system in mind there are a multitude of hardware options available for the design of a mobile prototype platform on which awareness strategies can be implemented with the main constraint being the budget. Therefore it was important to investigate all available electronics options in this project, before choosing a prototype to evaluate. Over the course of the project, numerous designs and ideas were considered and tested. A subset of which was found to be useful for the development of the aware robot. This section will discuss the design options considered for the hardware and software systems.

3.3.1 The robot prototype controller

The choice of controller for an aware system is critical as the controller is the heart and brains of the robot; so in order for the robotic prototype platform to evolve a number of design options were considered. The controller needs to be powerful enough to run complex algorithms, have enough I/O to accommodate sensors and flexible enough to evolve in the future without a major re-design.

3.3.1.1 Lego Mindstorms

One option which was considered in the early stages of the project for the prototype platform was the RCX from Lego Mindstorms. The LEGO Mindstorms kit from the LEGO group which fulfils the requirements of the system design contains an 8 bit microcontroller within a traditional LEGO brick. The robotics invention system (RIS) was purchased with the aim of using the on-board microcontroller to control the robot prototype platform. The RIS contains the RIS software, LEGO bricks, 3 sensors, 2 motors and a programmable

...
brick known as the RCX. The RCX comes equipped with 3 inputs and 3 outputs and this can be expanded by using a multiplexer. However, individual control and monitoring of the sensors attached to the multiplexer would prove difficult without the development of an additional PCB containing its own microcontroller, microprocessor or DSP.

![RCX programmable hardware](image)

The LEGO Group are a major sponsor of the Media Lab Group at the Massachusetts Institute of Technology (MIT), and as a result the design of the RCX was inspired by the MIT programmable brick. The two devices (The RCX and the MIT programmable Brick) use different CPUs that are running different software operating systems.

The CPU (Figure 3-3) in the RCX is a Renesas H8/300 [118] 8 bit single chip microcomputer. It is a 64 pin package with 8, 16 bit general purpose registers (or 16, 8-bit registers depending on what configuration is used) and it runs at 16MHz. The H8/300 CPU is contained on the H8/3292 IC that has 512 Bytes of RAM, 16kBytes of ROM, and an on-chip register field of 120 bytes.
The RCX can be programmed in many different languages ranging from the specially packaged RIS software, to C, C++ and Java, with Java becoming the most popular over the last few years with the advent of the LeJos firmware [119].

Combined with the CPU are a few vital components to get the RCX to function as a robot (Figure 3-4). The outlined components are as follows

1. **NEC 28 kB RAM module**
2. **Flip-Flops**
3. **Renesas H8/3292 Single Chip Microcomputer**
4. **NAND gates**
5. **Motor Driver Chips**
The communication between the RCX and the personal computer (PC) takes place over an infra-red (IR) link. On the PC side the IR tower connects to the USB port. The RCX communicates with the PC via the IR receiver at the front of the RCX. This allows new firmware and software to be downloaded to the RCX as required.

Although this system meets the requirements of the overall system it was determined that the 6 I/Os and issues with the communications between the PC and the robot would severely limit the evolution of a useful prototype platform so other options were considered at this stage.
3.3.1.2 Programmable logic controller (PLC)

The second option considered was to use a programmable logic controller (PLC) similar to the Siemens S5 or S7 as the controller. PLCs also fulfil the proposed system requirements and are used extensively in the manufacturing and pharmaceutical industry to automate processes based on sensory information provided to its I/O cards. These I/O cards come in the form of both analog and digital I/O and it was considered that this system which is predominately used for process automation could be adapted to create an aware robot. The Siemens PLC system comes equipped with a fully integrated development environment (IDE) and can be programmed in either ladder logic or statement list form. This PLC controlled robotic system could potentially work well for a stationary robot powered from AC, but for a mobile robot however the PLC and its associated I/O cards are quite large and heavy and would require a bulky 24V battery in order for the robot to be mobile, thus adding to its size and weight.

3.3.1.3 IC controllers

With this in mind the design option narrowed to using a microcontroller, microprocessor, digital signal processor (DSP), a field programmable gate array (FPGA) or application specific integrated circuit (ASIC). These are very powerful devices that could provide a platform for the prototype platform to evolve, thus giving the project a long life span. They are programmed in high level languages, such as C, and are designed for low power applications such as mobile phones, PDAs and laptop computers. A datasheet comparison of a number of microcontrollers, microprocessors and DSPs was carried out and it was decided that a Motorola VZ Dragonball (M68VZ328) MMU-less
microprocessor be used to control the operation of the robot based on its low cost, availability and range of hardware interface options.

FPGAs and ASICs were ruled out for this prototype system as the integration of such a device would require a substantial design effort in both hardware and software. The effort required is a project in itself and should be considered in the future due to the flexibility that FPGAs and ASICs give a system.

3.3.1.4 Robotic prototype microprocessor – Motorola Dragonball VZ

This Motorola Dragonball VZ microprocessor contains many features, but the main reasons for selection were:

- 76 general purpose input/output lines;
  - This provided adequate I/O resources for reading sensors and controlling motors.
  - 5 of these I/O lines are programmable interrupt lines, thus providing a point of entry to the software for safety critical sensors.
- Low Power;
  - The microprocessor's maximum current is 40mA; therefore the battery will not drain quickly from a heavily used processor.
- Standard Communications interfaces;
  - The microprocessor has two UART ports and two SPI ports.
- 2 PWM ports which can be used to control a motor/servo;
- LCD Driver ports which allows the robot to expand and use an LCD in the future;
• Glueless interface to SRAM, DRAM, EEPROM and Flash memory which allows the robotic system to expand easily if more memory was required for software, data storage or processing.

In addition to these features, the microprocessor was readily available on a development board with all its associated hardware thereby removing the need for the design, development, testing and fabrication of a PCB.

3.4 Sensors

As discussed in detail in Chapter 2, an aware system must be equipped with enough sensors to give it information about the surrounding environment. For the prototype system sensors were selected for the three aspects of awareness that were deemed essential for a mobile robot operating in a security role.

The first layer of sensors considered were distance rangers so as to provide the robot with a means of being spatially aware. It was decided that the SRF04 ultrasonic ranger from Devantech and the GP2D02 IR Ranger from Sharp were the most suitable. These sensors were chosen due to their low cost, low power consumption and accurate readings. They are to be used to give the system spatial awareness. Other rangers such as the SICK Laser scanner were considered but their high cost, typically between €3500 and €5000, prevented them from being purchased.

Orientation and positioning is very important for the development of awareness and a CMPS02 compass sensor from Devantech was considered to be the best option due to its accurate readings and low power operation. A GPS unit was also considered for future development of an outdoor robot. These sensors were considered to provide the robot with positional awareness.
In order to make the system useful and stand-alone it is important to include sensors that will detect physical elements that humans cannot readily detect. A sensor such as the flame-detector from Hamamatsu was considered as well as temperature, light, humidity and pressure sensors. Other low cost sensors included in the design were chemical sensors that sense Ammonia, Nitrogen Oxide, Methane and hydrogen. This group of sensors are intended to give the robot awareness of its environment.

3.5 Communications interface

In order to implement a system which is capable of evolving through combinations of distributed and centralised designs the communications interface needs to have wide enough bandwidth to transmit and receive information between the central processing unit and the robot. As the system is intended to be mobile the communications interface must be wireless.

Amongst the communications interfaces considered for this project were wireless Ethernet (Wi-Fi), Bluetooth™ communications and cellular network communications. Table 3.1 compares the communications interfaces considered for the project.

<table>
<thead>
<tr>
<th>Communications interface</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wireless Ethernet (IEEE 802.11)</td>
<td>- Protocol is defined.</td>
<td>- Limited range ~35m</td>
</tr>
<tr>
<td></td>
<td>- High speed ~11Mb/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Wide bandwidth</td>
<td></td>
</tr>
<tr>
<td>Bluetooth™</td>
<td>- Protocol is defined.</td>
<td>- Operational area limited to 100m from central processing unit.</td>
</tr>
<tr>
<td></td>
<td>- Wide range ~100m</td>
<td>- Low transfer rate ~2Mb/s (max)</td>
</tr>
<tr>
<td>Cellular Networks (GSM, GPRS, 3G, HSDPA)</td>
<td>- High speed.</td>
<td>- Charges from cellular networks for data transmission.</td>
</tr>
<tr>
<td></td>
<td>- Protocol is defined.</td>
<td>- Lack of coverage in certain areas/buildings.</td>
</tr>
<tr>
<td></td>
<td>- No limit on operational area.</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1 Communications interface options
It was decided that Bluetooth™ was the best fit for the system requirements; it has a defined protocol, a relatively wide reach and a reasonable transfer rate. Bluetooth™ modules which utilise the RS232 port on existing hardware are also readily available.

3.6 Associated supplementary electronics

The entire prototype platform system is integrated using supplementary electronics; analog, digital and mixed signal electronics. As described in Chapter 2 and later in Chapter 4 the sensors chosen for the project have their own on-board circuitry in order to digitise their outputs. This reduces greatly the work required in developing the design of the robotic prototype platform as the information which the Dragonball microprocessor has to deal with is all in the digital domain. The interface of the sensors to the microprocessor is described in more detail in Chapter 4.

The uCevolution board used for the development of the prototype platform is also described in Chapter 4 and has the required protection circuitry at its I/O ports and communication interface ports preventing the need for the design of a PCB and the validation and verification would be necessary.

For analog output sensors that do not have on-board electronics, such as the gaseous detectors from Applied Sensor described in Chapter 2 additional electronics must be designed in order to interface with the microprocessor. The sampling rate at which these sensors are read will not have to be very fast as the levels at which the gas is present in the atmosphere will not change rapidly. The best option for these sensors would be to route them through an external A/D converter or through the A/D converter which is built into the Dragonball VZ.
3.7 The robot prototype chassis

The original research into the chassis for the robot uncovered three options that hold true for any robotics application: one is to design and custom-build a chassis; two is to buy a kit such as Lego Mindstorms [120] and use its chassis as the "prototype platform"; and finally an off-the-shelf purpose built chassis can be purchased. There are drawbacks to each of these options, but a correct selection of the "chassis" can save time in the long run allowing more development time for awareness related issues such as sensor information processing and decision making.

There are numerous advantages in designing and building the chassis yourself with the most important being that it can be customised to suit the specific requirements of the project. However, what is gained in customisation is lost in time. The building of a chassis requires a large amount of time and mechanical skill and for these reasons it was decided that a custom built chassis would not be attempted for this project. The chassis systems evaluated over the course of this work is shown in Figure 3-6 and discussed below.

Real Robots [121] magazine supplied a ready built chassis for their particular robot, and this was readily available for use as a chassis. It is a rear wheel drive, three wheel chassis with motors and gears included. The plastics for the gearings were poorly made causing the wheels to get stuck intermittently so it was decided that this was not fit for purpose.

Robotics kits, such as Lego Mindstorms contain the pieces required to build a chassis, and because a Lego Mindstorms kit was readily available, the possibility of using it as the robot chassis was investigated. After some testing however it was decided that the Lego Mindstorms chassis was not large enough, and its motors not powerful enough to enable the robotic prototype platform to evolve. Another design option which was considered was the
use of a remote controlled car with a wide wheel base, but testing showed that this was incapable of carrying any weight heavier than 1kg.

In order for the robotic prototype platform to develop into a system on which new hardware can be added, the design requires motors capable of moving a payload greater than the 1kg i.e. without this capability, additional sensors required to make the system more aware cannot be added and larger power supplies necessary for extending the autonomous life of the robot system cannot be used. The discovery resulted in additional headaches as far as the design was concerned; e.g. power consumption will increase with larger motors therefore reducing battery life and electro-magnetic interference (EMI), both conducted and radiated, will increase thus increasing the number of filtering components and hence increasing size. This is why an off the shelf chassis was purchased from Active Robots [122].

Figure 3-6 DFD representation of chassis systems evaluated over the course of work.
It is possible to purchase an "all in one" package that includes the chassis, motors, motor drivers, sensors and sensor drivers, amongst other items, thus allowing the focus and effort to be on research and development of awareness strategies rather than spending time on the development and test of a chassis; however one must have a large budget in order to consider this and this option was not feasible.

3.8 Software model

When designing an overall system model, hardware and software must be taken into account. A good hardware package may not always have the development environment to match and vice-versa. So after the hardware was chosen for its power and portability there were only realistically three or four software options left for consideration.

Just as poor hardware design can affect the operation of a device as discussed in Chapter 1, poor software design can result in hours of needless debugging and troubleshooting. In this case the software was carefully designed before any code was written in order to make it user friendly and portable.

Another advantage of good software design is that software related problems can be quickly eliminated enabling hardware related problems to be identified and located.

The software is to be well commented, well structured and conform to coding standards. As a result, the task of writing new software to control the robot and read the sensors should be relatively straight forward.
3.8.1 Software design options and the choice of C

There are many computer languages that could be used in this project but a choice must be made that will benefit the project long term. The language must be powerful, yet easy to learn with a good standard structure in place.

As discussed in the previous section, a Siemens PLC could have been used to program the mobile robot. It comes equipped with the Siemens S7 software package allowing one to program the PLC to read inputs from devices, and send signals to output devices. It is a powerful package, is user friendly and it is easy to put an efficient, well commented structure in place. However, the benefits of using a small microcontroller rather than using a large PLC, which would have required a large power supply, outweighed the usability of its software package.

Another software package that was considered was LabView from National Instruments, designed for interface with National Instruments data acquisition cards. LabView provides an interface to National Instruments data acquisition cards as well as some other peripheral devices. However, the powerful LabView software package did not have the hardware to match.

Embedded Java began to surface about the time that the software platform was being considered. Java can be used for Embedded Linux and Lego Mindstorms and it is free to download for development from their website. However it had not been optimised for small processors and research showed that the processing power required and the ease of implementation would make it prohibitive compared to more established languages such as C or C++.
Assembly Language is by far the most powerful language that can be used, and many purists do not believe in writing code in any other language. It is ideal for low level real-time control of a microcontroller if it is coded efficiently and has been successfully used in other mobile robotics projects in the past. It is however extremely complex to structure effectively and for future software developers to become involved in the project at a later date, assembly language is not really ideal. However, in most languages that are used to program embedded systems, there is often a need to include some assembly language in their code in order to execute instructions at a higher speed.

The use of C with embedded assembly language is the preferred choice of embedded software engineers, giving them the added bonus of power with flexibility. It has been tried and tested for millions of combined man hours, and it has a very helpful and active community. It can be compiled at the command line using the free GNU C compiler (GCC) and the free EMACS from Linux. It is very powerful and a majority of microprocessors, DSPs and microcontrollers support it. The added advantage of using C is that an open source operating system such as uClinux can be used to develop the software, thus preventing the additional cost of licence purchase.

It was decided to develop the GUI in Visual Basic (VB) as it is a language that allows the rapid development of a graphical front end. Java and Visual C++ were both considered but the integrated communications interface MSCOMM which is part of VB made it the option which was most fit for purpose.
3.9 Concluding remarks

During the design phase of the project it was envisaged that the prototype system be capable of:

- Demonstrating awareness based on sensor information,
- Navigation with obstacle avoidance,
- Wireless operation,
- Evolving easily in hardware and software in the future,
- Being controlled as a centralised or distributed system.

When implementing these goals other requirements surfaced that were not initially taken into account e.g. the chassis will be required to carry a large payload as the project evolves in the future as more hardware is added.

The implementation of this functionality was somewhat restricted by the cost of hardware and software systems and resulted in relatively inexpensive hardware and open source software being used. At the time of development, single board computers (SBC) with the necessary hardware were retailing at €1500, whilst the integrated development environment (IDE) required to develop the firmware for the SBC cost €2000 plus an annual license fee of €500. Sensors were also quite expensive with the SICK laser scanner described in chapters 1 and 2 costing between €3500 and €5000 each. The chemical sensors discussed in Chapter 2 were in their infancy and only available to OEM customers. Commercially made research robots were not readily available either. K-Team in Switzerland produced a robot called Kephera which cost in excess of €4000. In order to control this robot a proprietary software kit had to be purchased. Robotics simulation systems were quite rare with licences ranging from €800 to €2000. These prohibitive costs greatly influenced the choice of
system and subsystem components with open source software and inexpensive hardware being favoured to proprietary software and complex hardware.

The final system design consists of a microprocessor running an embedded Linux operating system capable of processing sensor information, controlling motors and sending information back via a Bluetooth™ link to a PC for additional control and processing. This design allows the system to operate as a centralised or distributed system. The software is designed so that sensors and additional robots can be added and removed with ease. The chassis is capable of carrying up to 14kg which allows for the addition of some larger and more complex sensors. The final system design is shown in Figure 3-7. Chapter 4 and 5 discuss the implementation of the hardware and software for this system.
Chapter 4 Hardware implementation

This chapter describes the prototype hardware integrated into the awareness platform developed during the course of this work. The flow diagram in Figure 4-1 shows the sequence of work completed during the development of the prototype platform.

Figure 4-1 Flow control map of hardware and software integration
4.1 Microprocessor hardware – uCevo1ution

The uCEvolution board is an off the shelf development platform from Arcturus Networks which couples the Motorola Dragonball™ VZ microprocessor with a specially designed host platform which allows access to all of the microprocessors features via a small outline dual inline memory module (soDIMM) interface, thereby removing the need to design, develop and validate a printed circuit board (PCB). The benefit of a soDIMM interface is that the robot is not tied to a single processor as it is possible to upgrade the processor at any time as long as it adheres to the soDIMM single slot 144 pin formation. The soDIMM shown in Figure 4-2 provides all the required system memory, RS232 line drivers, Ethernet and high speed serial and parallel I/O. The fact that it is in an industry standard 144 pin soDIMM makes it an ideal development platform that can run for many years. It also eliminates the need to design an operating system as it already has the uClinux flavour of embedded Linux running on it.

![Figure 4-2 uC68VZ328 (uCDIMM/soDIMM)](image)

The uCdimm consists of 3 functional blocks; the Ethernet controller, the system memory and the MCU core (Figure 4-3). The low power consuming 33MHz Dragonball™ VZ MCU provides a FLX68000 CPU core and all the bus control logic, including the SDRAM controller, UART, SPI, LCD controller, timer/PWM and parallel I/O.
The uCdimm can be purchased with 2Mb of FLASH ROM and 8Mb of SDRAM or alternatively the 4Mb version of uCdimm with 8Mb of SDRAM can be purchased. The flash ROM device is the Atmel AT49BV1614, and all system calls are managed by the included bootloader. The program and erase of the flash ROM is also handled by calls into the bootloader and direct programming of the hardware should be avoided.

**Figure 4-3 uCdimm architecture**

The 8Mb SDRAM device is refreshed by the Dragonball™ VZ MCU and is configured by the bootloader or the uClinux kernel. The first 128kb block of SDRAM contains the global environment variables default fault handlers and debug stubs.

The uCdimm Ethernet controller is a CrystalLan CS8900A, implementing a 10BaseT Ethernet port with no external components, although the driver code must be provided by the OS running on the module.

The 2 RS232 ports are capable of running to 230400bps, and with integrated RS232 line drivers there is no need for any external components. The RS232 line drivers are capable of detecting if there is an RS232 device connected by checking the voltage on the RXD pin, if
there are no RS232 devices connected, the line drivers go into low power shutdown automatically. The default setting is 1900, 8, N, 1 on port 0.

There are two standard Motorola SPI serial masters (one master/slave) included on the uCdimm capable of running at up to 4MBps and up to 16bit word lengths. The SPI’s allow a connection to a large range of peripherals and other SPI slaves without additional components.

A single 8 bit PWM with 5 sample FIFO and a single bit PWM with 16 sample FIFO are provided, both capable of between 4ks/s and 32ks/s. There are two identical 16 bit hardware timers with 60ns resolution packaged with the PWM.

The uCdimm connects to a single panel monochrome LCD with up to 5 levels of grey out of a palette of 16 and is suited to common 640 x 512 panels. A frame buffer is provided to allow the LCD to be programmed with ease.

4.2 Hardware – sensors

Each of the sensors used in the project requires external circuitry so as to interface with the controller. The sensors used in the project, their hardware interface and their contribution to the awareness of the system are described in this section

4.2.1 Ultrasonic sensors – Devantech SRF04

Ultrasonic ranging sensors, and the Devantech SRF04 was chosen to provide the robot with relative spatial awareness to allow it to both navigate through an environment whilst avoiding obstacles and plotting a map of its surroundings for future reference.
4.2.1.1 Devantech SRF04 hardware

The SRF04 is controlled by the external microcontroller, the Dragonball™ VZ, by sending the Trigger Pulse Input pin high for approximately 10μs. After the init signal has been sent by the external microcontroller the Transmit transducer sends out an 8 cycle burst of ultrasound at a frequency of 40 kHz before waiting for a return signal.

The SRF04 has a timeout threshold of 36 ms due to the limitations of the timer on-board the PIC12C508; this means that if an echo is not detected within 36ms, the Echo line will go low and the sensor is ready to be initiated again as shown in the timing diagram (Figure 4-4). If an echo is returned within this 36ms threshold, the echo pulse will go high and be visible on the Echo Pulse Output pin with a pulse-width proportional to the distance of the object detected; this pulse width will be between 100μs and 18ms which must be timed with an external microcontroller before interpreting the signal and converting it to a meaningful value in software.

![SRF04 Timing Diagram](image)

Figure 4-4 Timing diagram for SRF04 ultrasonic sensor

The circuit (Figure 4-5) uses a PIC12C508 to control two standard 40 kHz Piezo Transducers. The transmitter transducer is driven by a MAX323 IC commonly used for RS232 communications and capable of providing 16 V of drive; the receiver circuit involves a two stage op-amp circuit also driven by the MAX323 IC.
Each gain stage is set to 24, which is close to the maximum gain of 25 available using the LM1458, so the op-amp circuit is set up for a gain of "576ish" [94]. The amplifier output is then fed to a LM311 comparator to generate a clean stable output.

The mechanical makeup of the piezo transducer means that it will resonate for a small period of time after the drive signal has been removed. This means that the chances of the receiver picking up coupling from the transmitter are quite high.

The returning echo from an object is much larger in magnitude than the resonating signal from the piezo in the transmitting transducer, so the SRF04 uses this fact to detect objects at close quarters. The detection threshold for a return signal is increased by charging a 100nF capacitor to -6V during the sonic burst so only the returning echo is detected, this allows the SRF04 to have a minimum detection range of 3cm. This 100nF capacitor discharges slowly enough through a 10k resistor for the receiver to detect objects that are close and quickly enough so the sensor can detect objects that are far away [123].

The echo can also be affected by high frequency noise generated by ICs. The MAX232 IC used to provide the op-amp and comparator with the negative voltage is shut down while the sensor is listening for the echo.

Once the discrete electronics have been taken care of the PIC12C508 waits until it receives an active low trigger pulse. It then generates a burst of 8 sonic pulses at a frequency of 40 kHz. The echo line is then raised in conjunction with shutting down the MAX232 IC to signal to the external microcontroller that it should start timing until the echo line goes low.

If an echo is detected within the threshold time of 36ms the echo line will go low and signal the external microcontroller to stop timing. The width of the echo signal is proportional to
the distance from the sensor to the object it has just detected. Because the MAX232 IC has been shut down the process must then wait at least 10ms so the ±10v can recharge

![Schematic of Deventech SRF04 sonic range finder](image)

**Figure 4-5 Schematic of Deventech SRF04 sonic range finder**

Since the integration of the sensors to this project there have been improvements made to the SRF04. The op-amp is now an LMC6032, the comparator is now an LP311, the 10µF capacitor has been increased to 22µF and some resistors have been tweaked. The pin formally named the "do not connect" pin can now be connected to ground (0v) to allow the sensor to be connected to slower controllers. If the pin is connected to ground, a 300µs delay is added between the trigger pulse and the transmitting sonic burst; if the pin remains unconnected the SRF04 will work exactly as outlined in the course of this section.

### 4.2.1.2 Physical connection of sonar sensors to uCevolution

The connection to the uCevolution board is as follows: VCC and GND are taken from the development board and routed to a 5V regulator circuit, which in turn provides power to the
Devantech SRF04 sonar sensor. The trigger input to the Devantech SRF04 sonar sensor is taken directly from GPIO Port A1 of the processor at pin 63 of the uCevolution development board and the Echo Pulse from the Devantech SRF04 is wired directly to GPIO Port A2 of the processor at pin 64 of the development board (Figure 4-6).

**Figure 4-6 Circuit Diagram of connection between uCevolution development board and Devantech SRF04 sonar sensor**

### 4.2.2 IR sensors

The IR ranger sensor was chosen to complement the ultrasonic sensor by providing the robot with shorter range spatial information.

#### 4.2.2.1 SHARP GPD02 hardware

The main reason for choosing the Sharp GP2D02 was for its easily useable output. It returns a byte of information containing the distance information, making it easier to interface the sensor to the Motorola Dragonball™ VZ.
The GP2D02 sends out a ping when its input line is driven low for 70ms. This ping returns a value that is proportional to the distance of the object that it reflects off and stores it in memory on the on-board PIC as a byte. In order to get the distance information from the sensor, the sensor must first be enabled before ‘clocking out’ the detector 8 times so as to get the byte of information.

The Sharp range of infra-red detectors come in a small package and require very little current to operate. This is an important feature of a sensor when making a mobile robot that is mobile. The first Sharp IR sensors used a Sharp metal can IR receiver along with a driving circuit that had one or more LEDs. The sensor worked but had limited range, was susceptible to ambient light interference, and required a drive circuit along with LEDs and a receiver.

A new approach that gives object detection along with a longer range and range information is used for the next generation IR sensor from Sharp. These IR sensors are almost immune to ambient lighting conditions. The current IR sensors use triangulation and a small linear CCD array to calculate the presence (or lack of presence) of an object in its field of view. The GP2D02 and the GP2D12 are capable of returning a ‘distance’ value if an object is detected in its field of view. The output of the sharp sensors is non-linear with respect to the distance being measured. (Figure 4-7) shows a typical output from the Sharp IR sensor.
4.2.2.2 Physical connection of IR sensors to uCevolution board

The Sharp GP2D02 IR Ranging sensor comes complete with everything necessary to integrate it to any available I/O pins on a microcontroller, including a JST connector with four wires and a small signal diode to prevent any damage occurring to the microcontroller.

The input pin is connected GPIO port 43 on the uCevolution board and the output is connected to GPIO pin 45 on the uCevolution board.

![Figure 4-8 SHARP GP2D02 hardware interface schematic](image-url)
Distance readings are taken from the Sharp GP2D02 sensor in the same sequence as for the Devantech SRF04 sonar sensor. The actual work of triggering the sensor and “clocking out” the return distance value are completed in the “sensors module” discussed in the next chapter.

4.2.3 Compass sensors

In every robotics project, localisation is a very important aspect of design and the Devantech compass sensor was chosen for this project so as to give the robotic test-bed the awareness capabilities of knowing absolutely where it is, and where it has come from. As discussed in Chapter 2 the Devantech CMPS01 compass sensor (Figure 4-9) uses a Philips KMZ51 magnetic field sensor that is sensitive enough to detect the earth’s magnetic field.

4.2.3.1 Devantech CMPS01 hardware

The CMPS01 is powered by a 5V power supply capable of providing a nominal 15mA. There are two ways to get the bearing from the compass; a PWM signal is available on pin 4 with a positive pulse width representing the angle; and an I²C interface is provided on pins 2 and 3. The PWM signal is used for this project with the pulse width varying from 1mS (0°) to 36.99mS (359.9°). There is also a 65mS low signal generated between pulses to distinguish between bearings.
Figure 4-9 Deventech compass sensor

The sensor was calibrated using an orienteering compass and a calibration switch circuit as shown in Figure 4-10. The orienteering compass was placed at magnetic north with the top of the CMPS01 facing in the same direction and the switch was closed and re-opened. The orienteering compass was then placed at magnetic east with the CMPS01 being rotated to this point before the switch was closed and re-opened. This procedure was carried out for the south and west directions.

Figure 4-10 Calibration circuit setup CMPS01
4.2.3.2 Physical connection of compass sensor to uCevolution board

The CMPS01 from Devantech can be connected to a microcontroller via an I²C bus or via a PWM output, where the pulse width represents the orientation of the compass. Although the MC68VZ328 microprocessor has the I²C hardware capabilities, the uClinux 2.4 kernel did not have the required software support and therefore the orientation of the compass had to be read via the PWM port on the MC68VZ328.

In order to activate PWM output from the CMPS02 it is necessary to put a 470kΩ resistor between pin two and ground and pin 3 and ground. The PWM signal is available on pin four of the CMPS02 and it is processed from the PWM port at pin 75 of the uCevolution development board by the software. To calibrate the compass, it is necessary to place a normally open switch between pin 5 and ground, pressing it at north, south, east and west, pulling pin 5 to ground and thereby calibrating the compass. It was shown during the testing and characterisation of the compass that the measurements calculated in software were within 2° of the actual measured value. This is outlined in greater detail in Chapter 6.

Figure 4-11 Circuit diagram of connection between uCevolution development board and Devantech CMPS01

...
4.2.4 Flame detector sensor

The UVTron bulb and flame detector circuit as discussed in Chapter 2 is a very simple, yet very sensitive, device that is capable of detecting a small flame from 5 metres. This sensor was chosen for this project to allow the robot prototype platform evolve into a search and rescue robot capable of detecting fire.

4.2.4.1 Hamamatsu flame detector hardware

The Hamamatsu driver creates the required voltage difference in the tube to allow the avalanche process to take place whenever ultraviolet rays are present. The circuit then watches the output current from the tube and when the avalanche process occurs, the circuit will allow the bulb to reset whilst sending out a pulse.

There are 3 output modes on the Hamamatsu driver (Figure 4-12): signal output, signal output inverted; and open collector output. For this project it was decided to use the inverted signal output for ease of software interface.

![Hamamatsu driver circuit diagram](image)

Figure 4-12 Hamamatsu driver circuit
The unadjusted output is a CMOS compatible high signal, dropping low for 10ms on each tube discharge. Because it takes 10ms for an instruction to execute on the MC68VZ328 processor, software could not be written to execute quickly enough to catch the 10ms pulse. A minor adjustment was made to the board by putting a 100μF capacitor onto the output of the Cx pin thus delaying the pulse in the low state to 1s.

Figure 4-13 Outputs on UVTron driver output connections

The software interface to the Motorola Dragonball™ VZ is explained in the next chapter.

4.2.5 Physical connection of light sensors to uCevolution board

There is a pair of light dependent resistors (LDRs) acting as sensors on-board the mobile robot. One LDR is set up to detect when the environment is dark, the other is set up to detect when the environment is bright. The first LDR operates as a pull-up resistor to its voltage divider circuit while the second LDR operates as a pull-down resistor to its voltage divider circuit.
4.3 Hardware – motors

The initial solution which was proposed for control of the robot was to use a port on the uCevolution development board to control the RCX motors via a CMOS Quad Switch and the 3 inputs available. Although this worked in theory, and worked to a certain extent in practice, the inability to regulate the power applied to each motor rendered the solution unfit for purpose as discussed in the next section.
4.3.1 Mindstorms – uCevolution interface via CMOS quad switch

As was discussed in Chapter 3 there were numerous options considered for the chassis of the prototype system including the use of the uCevolution development board to control the Lego Mindstorms RCX brick. The proposed solution was to implement the decision making on the uCevolution development board before sending a signal to the RCX, which would then in turn control the motors sending it in whatever direction required. A number of experiments were carried out using this set-up and it was shown that it was possible to send signals from the uCevolution development board to the RCX brick and get software on the RCX brick to perform motor control algorithms.

This solution was discounted after experimentation showed that it was not possible to distribute the power evenly to each motor and that the chassis was incapable of carrying a payload of greater than 2kg.

These restrictions prompted an investigation into alternative solutions such as to the use of the motors and chassis of Real Robots, and the motors and chassis of a Nikkon radio controlled car.

4.3.2 Real Robots – uCevolution interface via PWM port

De Agostini, the magazine manufacturer released a publication called “Real Robots” that allowed one to build an autonomous robot from the parts. A motor driver board came included with the motors for the “Real Robots” publication and following testing it was determined that the motor driver could be used to drive the motors in a forward and reverse direction, but not to control the speed of the motors. The pins on the motor driver circuit that controlled the direction of the motors were located, and software was written to drive the motors in any direction required. Software to give control over the PWM output was
written and tested extensively by implementing hardware and software experiments. A
switching circuit was then designed to provide control over the direction of the motors, and
this was tested. The switching circuit which controlled was a major obstacle. Although the
correct signals from were being generated from software it was discovered that the CMOS
switch never gives the open circuit that is required to drive the motors just as a real switch
would. The principle of the idea was shown to work by using a 5V regulated circuit and a
pushbutton switch connected to the pins of the supplied motor driver. However the
electrical characteristics of the CMOS switch means that there is always an electrical path
between the gates and hence it can never be in a truly open state.

4.3.3 Nikkon radio controlled car – uCevolution interface via PWM port

An alternative to using the “Real Robots” chassis was to replace the electronics of a Nikkon
radio controlled car with electronics controlled by the uCevolution development board.
After some research into different methods for bi-directional motor control it was decided
to use the L293D motor driver IC from STM to run the motors. The L293D is a quadruple
high-current half-H driver designed to provide currents to 600mA per channel at voltages
from 4.5V to 36v. The circuit was set up to provide a full H bridge reversible drive suitable
for bi-directional drive of the motors as well as control over the power supplied to each
motor.

The tests carried out using this motor driver circuit were controlled using 3 outputs from
the uCdimm. The SPI controller outputs the bit pattern needed to control the motors while
the SPI Clock pin clocks the bit pattern into the Shift Register. The speed of the motors was
controlled from the PWM output. It was decided to integrate the motor driver circuit into
the chassis of a remote control car and use the motors, wheels and battery supplied with the
car as the base for the robot. The control software and the motor driver circuit was tested extensively with the motors from the RC car and it was proven that the software and motor driver circuit were working correctly, however the current required to power the motors at startup and when subjected to a payload exceeded the specification of the L293D IC. A circuit using Darlington transistors was constructed but after some testing and analysis it was decided that the setup would not work due to the necessity to drive the motors in both directions. At this stage a decision was made to purchase a chassis with a motor driver supplied allowing speed and bi-directional control.

4.4 Hardware – chassis

When discussing the design in Chapter 3 and the details of the hardware implementation in the previous section it was revealed that there were a number of chassis evaluated over the course of the work but they were discounted mainly due to size, or lack thereof and their inability to evolve to carry a greater payload. As a result of this experimentation a chassis from Active Robots as shown in Figure 4-15 was purchased and implemented in the final design.

Figure 4-15 Active Robots chassis and shown from underneath with motors controlling the two large wheels
4.5 Hardware – communications interface

As discussed in Chapter 3 the decision was made to implement a Bluetooth™ interface as the means of communication for the system. It is possible to connect a PC and the uCevolution development board via an RS232 cable so by replacing the “real” RS232 with a “wireless” RS232 connection the uCevolution development board is made mobile. This mobility is required if the robotic prototype platform is to explore environments outside the immediate radius of the PC which it is being controlled from.

4.5.1 BlueWAVE industrial wireless cable hardware

The BlueWAVE wireless cable shown in Figure 4-16 is designed to provide a standard Bluetooth™ interface conforming to the Bluetooth™ v1.1 standard [124] and was purchased to make the uCevolution development board mobile. The wireless cable consists of a Bluetooth™ master module and a Bluetooth™ slave module that when used together provide a “virtual” RS232 cable. However it is not necessary to know which is the master and which is the slave if the units are being used as two ends of a wireless cable. The master and slave units are powered by a 7.2 V power supply and allow the connection to an RS232 device using the 9 way D-type connectors. One unit is supplied with a male 9 way D connector whilst the other unit has a female 9 way D connector.

Figure 4-16 BlueWAVE industrial wireless cable
The operating range of Bluetooth™ is specified as 100m, but this can be diminished by environmental variables such as walls and large objects.

The power supply to the units is regulated on-board to 3.3 V but the power supply used must be capable of delivering 150mA minimum. If hardware flow control is required it must be enabled on the host device. If hardware flow control is not required the RTS and CTS lines of the cable must be connected together.

The BlueWAVE cables are considered to be two ends of a virtual cable so in theory anywhere there is a real RS232 cable it can be replaced with the BlueWAVE “wireless” cable regardless of operating system software. The BlueWAVE units support hardware flow control that must be enabled on the host device if required. The uCevolution development board does not support hardware flow control so it is suggested in the manual that the RTS and CTS lines must be joined together. A header was constructed to connect the BlueWAVE module to the uCevolution development board via a 9-pin D-type gender changer. Pins 7 and 8 were removed from both sides as shown in Figure 4-17 and pins 2 and 3 were soldered together.

![Figure 4-17 Diagram of 9-Pin D-type female gender changer with adjustments for use with BlueWAVE wireless unit](image)

Each BlueWAVE unit comes equipped with a regulated 7.2V power supply; however this was not a suitable means of powering a unit that travels with the mobile platform.
A 7.2V regulator circuit powered from two 9v batteries was built in order to supply the BlueWAVE unit with the 7.2V required to drive it. Experiments which were carried out show that the BlueWAVE units draw 150mA when they attempt to locate each other and the drain on the batteries is increased as the space between the transmitter and receiver increases. The BlueWAVE units also draw 150mA as information is exchanged between the units. These experiments showed that the current requirement drains the batteries in a number of minutes. A single 9v battery was connected with a series resistor to a 2.1mm plug to give the BlueWAVE unit the current it needed and this is capable of running for 15 to 20 minutes depending on the quantity of information exchanged between the two devices. Although the BlueWAVE units work, the high current drain does not make the units suitable for use in small scale mobile platforms.

In larger scale developments which do not have a weight restriction the option of using larger and heavier battery packs can be considered. A circuit was constructed for the Max 99 robot base that used a 1900 mA/h 7.2V battery pack which gives about 15 hours continuous communication between the uCevo evolution development board and the PC (Figure 4-18).

Figure 4-18 Picture of BlueWAVE mobility circuit using a 1900 mA/h 7.2 v battery pack
4.6 **Hardware - power supplies**

Supplying power to each component for adequate periods of time is very important for mobile robotics. Each separate group of components on the board has a different power requirement. For example, the microcontroller requires 12V, the motors require a 12V supply capable of delivering up to 1.5A, and each of the sensors require 5V. A decision was made at this stage that a central power supply board should be built, with regulated outputs of 12V, 7.5V, and 5V for the microcontroller, the Bluetooth™ transmitter/receiver and the sensors respectively. It was decided that due to the large current requirement of the motors, a separate battery and power circuit should be built with an interface to the microcontroller.

4.7 **Concluding remarks**

This chapter discussed in detail the hardware used in the development of the robotic prototype platform. The details of the Dragonball™ VZ microprocessor was discussed and the interface of the sensors and motors to this processor as well as the details of the Bluetooth™ wireless communications interface to the PC. The prototype platform setup with annotated picture can be seen in Figure 4-19. Chapter 5 discusses the software interface to this hardware including detail on the structure of the software modules, the software modules integration into the Linux kernel and the development of the GUI with Visual Basic.
Prototype System Hardware Setup
Block Diagram

uCevolution Processor Board

- IRMFR Ranger
- SRTP4 Ultrasonic Sensor
- UVTRON Plane Sensor
- Light Sensor
- Bluewave Wireless Module
- Motor 1
- Motor 2
- Bluewave Wireless Module
- Laptop Processing Data from Robot

Annotation of sensor positioning shown here. Refer to the colour coding on block diagram above.

Figure 4-19 Block diagram of hardware implementation and annotated picture
Chapter 5 Software implementation

The last chapter described the different options tested during the development of the hardware prototype platform ranging from the use of the RCX hardware in the early stages of the project to the use of a powerful development board in the final design and as the hardware components of the robotic prototype platform evolved, so too did the software implementation. This chapter will discuss the software implementation from its infancy to a final design which is capable of integrating new hardware components easily in order to develop a wide range of awareness strategies.

5.1 RCX software

As discussed in Chapter 4, the RCX hardware from Lego contains a powerful Renesas H8 microcontroller capable of executing many tasks with the correct software implementation. The Mindstorms product comes complete with a software development kit that allows the user access to only a small portion of this processor's capabilities. Therefore in order to explore the full capabilities of the system it was necessary to implement open source solutions.

5.1.1 NQC

Not Quite C [125] (NQC) developed by Dave Baum is a programming language with C-like syntax designed specifically for programming the RCX hardware and is free to use under the Mozilla Public Licence (MPL). It comes with a command line tool which is used to program the RCX hardware. The RCX firmware must be replaced with NQC firmware. This is implemented by connecting to the RCX from the command line via the
communications port (USB or RS232) and downloading executable software to the RCX hardware. This language was used to evaluate the capabilities, limitations and characteristics of the RCX as a development prototype platform by executing code to perform certain tasks, such as follow lines and avoid obstacles.

5.1.2 BrickOS (LegOS)

BrickOS [126] (formally LegOS) is a low level OS developed on Linux capable of realising the full power of the Renesas H8 microprocessor. It is written in C and requires a full install of Cygwin in order to run in a Windows environment. In contrast to NQC and the software supplied with the Mindstorms kit, BrickOS allows software to be written for the RCX hardware with little or no limitations. The application software can be written in C or C++ with the benefit of access to all 32kb of memory, complete LCD control, priority pre-emptive multi-tasking, floating point calculations and the ability to run multiple programs.

In theory this was an ideal solution for using the RCX hardware as a development platform. However in practice the operating system had not really developed at the same pace as the RCX hardware and a considerable amount of time was spent troubleshooting unsupported hardware issues rather than developing application software. For example the universal serial bus (USB) infra-red (IR) tower was not supported by the maintainers of the BrickOS replacement firmware and because the USB IR tower was unsupported by BrickOS, it required an investigation of IR and USB protocols so as to implement them into BrickOS and in turn program the robot. A considerable amount of time was invested in researching Lego network protocol (LNP) and USB protocol, and their operation.
5.2 μCLinux

The micro Linux (μCLinux) distribution has been optimised to have a very small code size whilst still retaining all the major advantages of the Linux operating system. It was initially designed for the Motorola 68328 chip, but was subsequently ported to many more including the Motorola Dragonball™, Motorola Coldfire, ARM7TDMI, Intel x86, ADI Blackfin and the 2500, 3000 and 4000 series Cisco routers. It was considered to be the best and most portable operating system available for the development of a robotic prototype platform due to the wide range of microcontrollers it supported.

5.2.1 μCLinux – background

The version of μCLinux that was supplied by Arcturus Networks with the μCevolution development board was μCLinux 2.0.x, however it was later upgraded to 2.4.x. The 2.4.x distribution is run under Red Hat Linux 9.0 in conjunction with the m68k elf toolset to compile a downloadable kernel image for the MC68VZ328 Dragonball™. The installation and testing of which is discussed in greater detail later in the chapter.

One of the main reasons for the successful and rapid progression of Linux and subsequently μCLinux is due to its development on the open source platform using GNU general public licence (GNU GPL). Therefore, by allowing thousands of programmers to study existing implementations, including source code and kernel modifications, these programmers were able to bring these platforms to the next level in a short space of time.

In the same way that Linux has different “flavours”, embedded Linux has a number of different vendors that have developed “development kits” for their own flavour of embedded Linux. Metrowerks, an independently operating subsidiary of Motorola sell an embedded Linux software development package that is compatible with most 32-bit
processors, FSM Labs have a similar package called RTLinux, whilst MontaVista have
developed a software package that supports ARM, PowerPC, x86, MIPS,
StrongARM/XScale, SH and Xtensa [127].
These packages support both microprocessors with and without a memory management
unit (MMU). With more of an emphasis on portability and low power consumption in the
last few years and the fact that of the 2 billion microprocessors fabricated each year, 80
percent go into embedded systems with highly specialised functions [128], a majority of
embedded engineers opt for “MMU-less” processors running the \( \mu \)CLinux flavour of
Embedded Linux.

5.2.2 \( \mu \)CLinux – A portable software model
By using the C programming language and the \( \mu \)CLinux flavour of embedded Linux it is
possible to build a portable code system using the standard format that is already in place.
The code is designed in such a way that it can be compiled for any other microcontroller
that is running the \( \mu \)CLinux flavour of embedded Linux. The location of the I/O ports, the
timers and the registers are the only aspects that have to be changed in order for it to
completely work.
The software is designed, written in C and tested in blocks before being integrated into the
\( \mu \)CLinux kernel (Figure 5-1).
Each module is in effect a device driver that defines the operation of each device. The sensor module defines the operation of each sensor and can be accessed by "user applications" to obtain data from the sensors. The operation of each sensor is coded in the module software and the module is designed to return a "raw" reading each time it is queried by an application. This "raw" data reading makes the module software very portable as it allows users to do what they will with the data reading. The module is capable of handling up to 32 sensors. The motor module defines the operation of each motor, with the capability to add up to 4 more motors if required. Once testing is completed on each module they can be integrated into the μCLinux distribution as discussed later in the document.
5.2.3 μCLinux – module structure

The structure of the each function follows the same model as seen in Figure 5-2, with any peripherals being controlled by a device driver inserted into the kernel, and being accessed by user applications also embedded in the μCLinux kernel. This structure allows future hardware developers to add new sensors if and when they require and future software developers to write C code to access these sensors and use the information to develop awareness strategies.

Figure 5-2 Software flow for a sensor read

The structure as seen in Figure 5-3 shows that from a “main” program a switch-case control loop determines the functions that will be executed in software. This allows a developer to
add new functions to complete different tasks at the highest level possible without ever having to worry about how exactly each sensor works.

**Figure 5-3 Software flow control from main program**

Once a function is called from the main loop, it can do one of two things; it can access the sensors and motors that are already present in the kernel or it can execute some other function. Figure 5-4, Figure 5-5 and Figure 5-6 shows how a sensor read should take place after being called from the main program. It was decided that a call to any module should be controlled by the IOCTL interface supplied by the µCLinux framework.
Figure 5-4 In-depth execution flow of software from main program
Figure 5-5 In-depth execution flow of software from irApp.c program

Figure 5-6 In-depth execution flow of IOCTL call from irApp.c program
5.2.3.1 Software implementation of Devantech sensor module

As was outlined previously in this chapter, the control of the Devantech SRF04 sonar sensor takes place in the "sensors module" within the µCLinux kernel. The sensors module is inserted when the kernel is booted initialising the ports that are to be used by the Devantech SRF04 sensor, as well as the sequence of events necessary to return a distance measurement from the sensor. A sensor reading is obtained by executing an application that queries the sensor module using an IOCTL interface. The actual code is written in C and block diagram description can be seen in Figure 5-7.

![Block diagram explanation of software interface between Devantech SRF04 sonar sensor and the µCLinux kernel.](image)

Figure 5-7 Block diagram explanation of software interface between Devantech SRF04 sonar sensor and the µCLinux kernel.
5.2.3.2 Software implementation of SHARP GP2D02 sensor module

Distance readings are taken from the Sharp GP2D02 sensor in the same sequence as for the Devantech SRF04 sonar sensor. The actual work of triggering the sensor and "clocking out" the return distance value is completed in the "sensors module". When the sensors module has completed the measurement cycle, it returns a raw distance value to the calling application. The reason a raw distance value is returned to the calling application, rather than a SI distance is so make the software as portable as possible. The raw distance value is entered into a small distance calculation algorithm to give a distance in millimetres, centimetres or metres depending on the resolution required (Figure 5-8).

![Figure 5-8 Block diagram explanation of software interface between Sharp GP2D02 and the μCLinux kernel](image-url)
5.2.3.3 Software implementation of CMPS02 sensor module

The software takes the same structure as the IR ranger and the ultrasonic sensor for the integration of the CMPS02. The main program makes a call to the /dev/sensors module and asks for the CMPS02 to return an orientation value. This orientation value is a positive integer between 0 and 359. North is calibrated to be 0, east is a calibrated to be 90, south is calibrated to be 180 and west is calibrated to be 270.

Figure 5-9 Block diagram explanation of software interface between CMPS02 compass and the µCLinux kernel
5.2.3.4 Software implementation of LDR sensor module

The software control sequence is the same for this part as it has been for the other sensors that were integrated. Controlled from the main program, a function is called to test if the environment is dark or bright, with logic 1 being returned from one sensor if the environment is bright while the other will return logic 1 if the environment is dark. The sensors if used in this fashion can be used as a security mechanism along the following lines; if there is an object detected by the IR sensor and the environment is dark, turn on lights and take a picture using a camera.

5.2.3.5 Software implementation of flame detection sensor module

The software is limited by the fact the Hamamatsu driver board is only capable of indicating the presence of the flame, with no additional information. As a result, the software will signal if there is a flame detected by the Flame Detector allowing a function to search the immediate area using the other on-board sensors to be implemented.

5.2.4 μCLinux – adding new sensors

The well structured code makes it very easy to add as many sensors as is required for a given application. The code can be tested independently as a standalone application before integrating it into the sensor module. It is then a matter of using the IOCTL interface to obtain information from the sensor. The necessary steps required to integrate a new sensor to the sensors module are discussed below.
5.2.4.1 Sensor device drivers

Device driver software should be tested thoroughly as a standalone application before integrating into the Linux kernel; the reasons being two fold. Firstly the kernel compile takes a long time, and secondly the device driver could potentially lock-up the kernel if there are embedded software bugs.

The stand alone application should initialise any ports, timers and registers required before performing instructions on these ports, timers and registers to test the code. It is best to develop the software in such a way that it can be easily ported to a module once tested thoroughly.

Once tested the sensor device driver can be added to the kernel as follows:

In the defining header file, in the case of this project sensors.h (Figure 5-10):

- Add a magic number to represent the new sensor.
- Define the IOCTL call as Read Only, Read/Write or Write only depending on the hardware specifications of the sensor.
- Define the functions required by the sensor.

The software will compile into the kernel at this stage, but until the tested application has been ported to the sensors.c source file no tasks will be executed.
Figure 5-10 Additions to be made to the sensors.h file in order to add a new sensor

In the sensors.c source file, an area should be sectioned off for the new sensor (Figure 5-11), to aide future troubleshooting before adding the sensor to the IOCTL interface (Figure 5-12).

Once the sensor has been added to the /dev/sensors all the user space application has to do in order to access the sensor information is to call this module and request a return value.
Figure 5-11 Assign a section to the sensors.c file to accommodate the new sensor.

Figure 5-12 Additions to be made to the IOCTL function in sensors.c
5.2.5 μCLinux – adding modules and applications

Once the device driver modules and applications have been tested thoroughly, it is necessary to add them to the μCLinux distribution kernel so they become part of the firmware. In order to add the device driver modules and applications to the kernel, adjustments to a number of files within the μCLinux distribution is required. This section describes a generic example of how to add an application and a device driver to the kernel.

5.2.5.1 Adding applications

It is necessary to add the application to the kernel so it starts when the hardware has powered up and to do this it is necessary to copy the required source code to the μCLinux-dist/usr directory and edit the Makefile in the usr directory and the Configuration files in the config directory.

The following edits must be made:

- Add the following line to /usr/Makefile:
  ```
  dir $(CONFIG_USER_MYAPPLICATION)+= myApplication
  ```

- Add a brief description of the application to /config/Configure.help. (Figure 5-13)
  - The text must be indented by two spaces.
  - There must be no empty lines.
  - Lines should be less than 70 characters long.
CONFIG_USER_MYAPPLICATION

This is an example application, that does X, Y and Z.

Figure 5-13 Configure.help example

- Edit the /config/config.in file to add the menu location where it is appropriate to give the option of selecting and deselecting the application as part of the "make menuconfig" step of the kernel compile.
  - bool 'user ex application' CONFIG_USER_MYAPPLICATION

5.2.5.2 Adding modules

To add a module to a μCLinux distribution the module directory should be added to the ../linux-2.4.x/drivers tree and a reference to it included in the drivers Makefile.

The global Makefile should be modified to contain the line subdir-m += myModule

where myModule is the name of the hardware module. An individual Makefile is required to compile the module as part of the kernel and should look similar to Figure 5-14.

Once these steps have been followed the integration of the module into the kernel can be tested by compiling the module using the "make modules" command or a complete re-compile of the kernel.
5.2.5.3 Software algorithms

As discussed in the previous section, once all the hardware initialisation is completed in a software device driver integrated into the µCLinux kernel then software algorithms can be processed on either the µCevolution development board or a PC communicating with the board.
Any software algorithm run as a user application can query the `/dev/sensors` module to get sensor data, and then call the `/dev/motors` module to drive the motors in whatever direction is necessary.

![Diagram of software algorithm function accessing `/dev/sensors` and `/dev/motors`]

**Figure 5-15 Software algorithm function accessing `/dev/sensors` and `/dev/motors`**

5.3 *Graphical user interface – spatial and location awareness*

As discussed in Chapter 3 the development of a GUI interface was required so as to give a visual indication as to the level of awareness the mobile robot has when exploring the environment and to give the system greater flexibility in evolving as an aware system. The information is sent from the mobile robot to a computer via the Bluetooth™ link which allows the on-board obstacle avoidance and sensor measurement to take place on the
mobile robot but the complex processing will take place on a computer which is not limited by hardware restraints.

5.3.1 GUI – basic operation

Visual Basic was chosen as the development platform due to its utilisation of the MSCOMM utility of Windows. The MSCOMM utilities allows one to access, and gather information from the RS232 port, so as a result the $\mu$Cevolution board can be connected to the RS232 port and information can be read at the port through Visual Basic.

The Visual Basic program reads the data from the RS232 port on a PC, before displaying the sensory information on a GUI. It is capable of keeping track of the robot’s movements and in turn lets the robot know where it is in the environment as well as where it has been.

The communications protocol between the robot and the PC is outlined in Figure 5-16.

<table>
<thead>
<tr>
<th>Header</th>
<th>Compass Orientation °</th>
<th>Valid Signal</th>
<th>Distance in cm</th>
<th>Other Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>~#d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>&amp; or !</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 5-16 Communications protocol

For example if the data received by the PC application is “~#d090&123” then this signifies that there is an object 123 cm away 90° to the robot.

5.3.2 GUI – plotting the sensor information

The robot’s position is plotted relative to the sensor information it receives, i.e. it is designed to be used in a structured environment (N x N) and plot the outer-skirts of the environment. For example, when a distance value is read from the RS232 port it is plotted on the screen at a distance that is proportional to the representation of the robot. Once a distance is measured, it is stored until the next reading where it is compared with the new...
reading; if the new reading is different the robot representation will move to represent this. The GUI is only a representation of the robot’s position in relation to its environment after an initial scan; it is up to future developers to implement an accurate representation of where the robot is.

The GUI provides access to the software running on the uCevolution development board and to visualise the position of the robot in the environment from distance information provided by the ultrasonic sensors and the IR sensors and orientation information is provided by the CMPS02 compass. It is possible to access the uClinux kernel running on the uCevolution development board from the Windows environment, but the uClinux kernel must still be programmed from within Red Hat 9.0.

The initial screen (Figure 5-17) that is seen when the program is run is the communication settings screen; the defaults are also the defaults for accessing the uCevolution board, so there should be no need to change these.

![Communication settings screen on Bigmac application](image)

Once the “Apply” button is pressed the main screen is displayed and it is at this screen that a majority of the control will be done. If the uCevolution development board is connected
to the PC, a screen similar to Figure 5-18 should be displayed; the bootloader screen can be seen at the far left, with a command window and a display area towards the centre.

![Figure 5-18 Startup Screen with uCevolution board attached](image)

To boot the uCevolution board type “go” in the command window and when the system has booted successfully the login screen will be displayed (Figure 5-19); the login name is “train” and the password is bypassed.
A number of experiments were carried out using distance and orientation values from the IR sensor and compass sensor respectively, and in spite of hardware issues regarding the motor driver circuitry the robot was physically moved to show that the GUI and robot representation worked as designed.

This was demonstrated by placing an obstacle in front of the robot at varying distances from 10cm to 50cm. It could be clearly seen that the GUI was updated in real-time as the obstacle was moved thus creating a visual representation of the spatial awareness associated with the prototype robot platform. In addition to this in order to verify that the compass bearing information was being displayed correctly, the robot was rotated in 10° increments and a reading from the orienteering compass was taken and compared with that on the GUI interface. It could be seen from the GUI that as the robot was rotated the bearing information was updated in real time on the GUI hence giving a visual representation of the positional awareness of the robot. The GUI output can be seen in Figure 5-20 for the initial measurement in a 60cm wide by 40cm long box.
The GUI gives a representation of where the robot is with the data received from its sensors and the robot prototype platform is capable of communicating with the uClinux kernel and running any programs and modules that are present thus giving the potential of adding more sensors and making it more than just spatially aware.

5.4 Concluding remarks

This chapter described the implementation of embedded Linux on the Dragonball™ microprocessor and the integration of new software modules into the Linux kernel. The chapter discussed in detail the steps required for the addition of new hardware to the system. It was shown that the prototype platform is capable of communicating via the Bluetooth™ link with a central compute and displaying the sensory information in real-time on the GUI.
Chapter 6 describes the steps taken to characterise the sensors. The experiments were carried out to determine if the sensor data that was being received was providing the robotic system with an accurate representation of its environment, i.e. could it be said that the robotic system had a spatial and environmental awareness based on the sensor information it was receiving.
Chapter 6 Sensor characterisation

The previous Chapters 4 and 5 have described the hardware and software required to develop a robotic prototype platform on which awareness strategies can be explored. In order for this to happen the sensors must be characterised to ascertain their accuracy when integrated into the robotic system. This chapter describes the characterisation of the sensors when integrated into the robotic prototype platform.

6.1 Sensor characterisation – Devantech SRF04

In order to get an appreciation of the range of view of the Devantech SRF04 ultrasonic ranger a test was carried out to find out how far the measured distance deviated from the actual distance for different angles of incidence in steps of 10°, from 0° through 30°. It can be clearly seen from Figure 6-2 that the accuracy of range of the Devantech SRF04 ultrasonic ranger deteriorated as distance and angle of incidence increased.
This inaccuracy is unavoidable when using the MC68VZ328 as the processor, as the timer used to measure the return signal from the Deventech SRF04 ultrasonic ranger is not resolute enough. However, in order to get the most accurate, real-time reading from the Deventech SRF04 ultrasonic ranger the average of ten readings is returned to the calling application whenever it requires a reading. As a consequence the central limit theorem, by increasing the number of readings, the mean will tend to become closer to the true value. The value of 10 readings was chosen as this gives a more accurate result without affecting the rapidity of the measurement. It can be seen from Table 6.1 that the standard error on measurement is very small at 0°, 10° and 20° but increases at 30°.
<table>
<thead>
<tr>
<th>Actual Measurement</th>
<th>25cm</th>
<th>50cm</th>
<th>75cm</th>
<th>100cm</th>
<th>125cm</th>
<th>150cm</th>
<th>175cm</th>
<th>200cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean 0 Degrees</td>
<td>24.9</td>
<td>51.7</td>
<td>68.5</td>
<td>95.2</td>
<td>120.4</td>
<td>134.3</td>
<td>166.4</td>
<td>189.9</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>0.316228</td>
<td>4.164666</td>
<td>0.707107</td>
<td>3.155243</td>
<td>0.516398</td>
<td>4.191261</td>
<td>0.699206</td>
<td>0.316228</td>
</tr>
<tr>
<td>Standard Error</td>
<td>0.1</td>
<td>1.316983</td>
<td>0.223607</td>
<td>0.997775</td>
<td>0.163299</td>
<td>1.325383</td>
<td>0.221108</td>
<td>0.1</td>
</tr>
<tr>
<td>Mean 10 Degrees</td>
<td>25.2</td>
<td>48.7</td>
<td>70.4</td>
<td>96.9</td>
<td>120.5</td>
<td>143</td>
<td>162.3</td>
<td>188.9</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>0.421637</td>
<td>0.483046</td>
<td>1.349897</td>
<td>0.316228</td>
<td>0.527046</td>
<td>0.471405</td>
<td>0.483046</td>
<td>0.316228</td>
</tr>
<tr>
<td>Standard Error</td>
<td>0.133333</td>
<td>0.152753</td>
<td>0.426875</td>
<td>0.1</td>
<td>0.166667</td>
<td>0.149071</td>
<td>0.152753</td>
<td>0.1</td>
</tr>
<tr>
<td>Mean 20 Degrees</td>
<td>25</td>
<td>47.7</td>
<td>70.2</td>
<td>87.3</td>
<td>116.9</td>
<td>140.9</td>
<td>164</td>
<td>185.3</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>0</td>
<td>0.483046</td>
<td>1.75119</td>
<td>4.029061</td>
<td>0.316228</td>
<td>0.316228</td>
<td>0</td>
<td>0.483046</td>
</tr>
<tr>
<td>Standard Error</td>
<td>0</td>
<td>0.152753</td>
<td>0.553775</td>
<td>1.274101</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
<td>0.152753</td>
</tr>
<tr>
<td>Mean 30 Degrees</td>
<td>43.6</td>
<td>44.1</td>
<td>69.8</td>
<td>82.6</td>
<td>116.8</td>
<td>140.1</td>
<td>166</td>
<td>182.7</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>33.82701</td>
<td>0.567646</td>
<td>2.097618</td>
<td>11.09755</td>
<td>0.421637</td>
<td>1.370322</td>
<td>0</td>
<td>0.483046</td>
</tr>
<tr>
<td>Standard Error</td>
<td>10.69704</td>
<td>0.179505</td>
<td>0.663325</td>
<td>3.509353</td>
<td>0.133333</td>
<td>0.433333</td>
<td>0</td>
<td>0.152753</td>
</tr>
</tbody>
</table>

Table 6.1 Standard Error calculation based on recorded values

Based on this information the Deventech SRF04 ultrasonic ranger, if used with a MC68VZ328 processor should not be used for applications that require a great deal of long-distance accuracy such as mapping. It could perhaps be used on two levels, one as a "coarse" obstacle avoidance mechanism at distances from 50cm to 300cm and then as a local navigation sensor at distances from 10 to 50cm. This configuration would give a great deal of flexibility to a mobile robot allowing it to detect the presence of objects in the 50cm to 300cm realm, before categorising the object on closer inspection of 10cm to 50cm.
Figure 6-2 Plot of distance calculated from sonar vs. actual measured distance. Error bars can be seen on the measurement at an angle of incidence of 30°. The standard error at the other angles is too small to be seen on this plot. The ideal plot is shown in blue on this figure.

6.2 Sensor characterisation – SHARP GP2D02 IR Sensor

The GP2D02 SHARP IR sensor returns “raw data” that decreases exponentially with distance after a measurement (Figure 6-3). This attribute of the sensor means that it makes it more difficult to calculate an accurate reading without performing some mathematics on the raw data. Matthias Schöldgen [129] discovered that by dividing the returned raw data by 1500 an approximate reading for the sensor was returned with an error of ±1cm on each reading.
It was decided at this point to search for the best representation of the data in order to get a more accurate reading from the sensor.

An exponential approximation does not approximate to the actual data very well, so it was not considered. A third and fourth order polynomial were also tested, but the results returned by the third order polynomial and the fourth order polynomial were considered to be inaccurate and computationally expensive approximations (Figure 6-4, Figure 6-5). Lookup tables were considered for these calculations however, their use was discounted as the memory space required to store the tables was considered to be too large.
Figure 6-4 Plot of third order polynomial on raw distance vs. Measured distance (cm) plot

Figure 6-5 Plot of fourth order polynomial on raw distance vs. Measured distance (cm) plot
The second attempt was to compute the natural log of the raw sensor reading, and plot it against \( \frac{1}{\text{Measured distance (cm)}} \) resulting in the graph shown in Figure 6-6. It can be seen from this plot that the best approximation is a third order polynomial:

\[
y = 1235.4x^3 - 374.69x^2 + 46.564x + 2.929.
\]

This polynomial would require six calculations to convert the raw data and return it as an accurate distance reading rather than a divide by 1500, so what was gained in accuracy, would be lost in execution time.

![Figure 6-6 Plot of ln(raw distance) Vs 1/Measured distance(cm)](image)

After some more tests, and gathering of raw data results from the sensor it was discovered that a third order polynomial approximation on the plot of actual distance (cm) Vs ln(raw data) provided the most accurate distance measurement (Figure 6-7). It requires four instructions to calculate the distance returning a distance value in 40ms. The third order polynomial can be adjusted to return a distance in m or mm, but the default measurement is cm. The code is executed in the following steps (Figure 6-8); the raw data is clocked out of
the sensor, where the natural log is obtained i.e. $x$ is calculated. The natural log value is cubed i.e. $x^3$ is calculated and stored in a variable. Next the natural log value is squared i.e. $x^2$ is calculated and stored in a variable before $x, x^2, \text{ and } x^3$ are passed to the "calculate polynomial" function. This function returns a value that is as accurate as possible to the actual distance value.

Figure 6-7 Plot of measured distance (cm) vs. ln(raw data) with third order polynomial approximation
The code was analysed with MS Excel using real raw data readings from the sensor and comparing them to ideal theoretical values and it was shown that the polynomial calculation is accurate to ±0.1 cm up to distances of 40 cm (Figure 6-9).
In implementation where the calculated values were compared with values that were measured with a calibrated ruler it was shown that the calculations are accurate to 50cm with an error of ±0.4cm.

As can be seen from the plot Figure 6-10, the accuracy tails off after 50cm where there is quite a substantial difference between the actual measured distance and the distance obtained from the Sharp IR sensor. These results make the Sharp IR Sensor ideal for close range mapping due to the highly accurate distance measurements it is capable of.

It has also been shown that the method used to calculate the distance here is more accurate than any others used to date, a comparison between the method used here and the method devised by Schöldgen is shown in Figure 6-11.
Figure 6-10 Plot of calculated distance (cm) Vs Measured distance (cm)

Figure 6-11 Comparison of method utilising a polynomial and the method that calculates distance from 1500/raw distance data
In a similar fashion to the Devantech SRF04 sonar sensor, the standard error on the distance measurement is very low when detecting objects close to the prototype platform. However this error increases as the measurement distance moves beyond 30cm as can be seen in Figure 6-12 and Table 6.2.

Figure 6-12 Plot of mean of 10 IR sensor measurements at distances ranging from 10cm to 60cm. It can be seen that the standard error increases significantly beyond 30cm.
### Table 6.2 Standard deviation and standard error calculations for Sharp IR sensor.

<table>
<thead>
<tr>
<th>Actual Distance (cm)</th>
<th>Mean (10 cm)</th>
<th>Standard Deviation</th>
<th>Standard Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10.1930811</td>
<td>0.31714411</td>
<td>0.100289868</td>
</tr>
<tr>
<td>11</td>
<td>11.43794291</td>
<td>0.37805412</td>
<td>0.119551217</td>
</tr>
<tr>
<td>12</td>
<td>12.73177611</td>
<td>0.469197969</td>
<td>0.148373426</td>
</tr>
<tr>
<td>13</td>
<td>13.89827183</td>
<td>0.570712673</td>
<td>0.180475194</td>
</tr>
<tr>
<td>14</td>
<td>14.77796404</td>
<td>0.409149384</td>
<td>0.129384396</td>
</tr>
<tr>
<td>15</td>
<td>15.69144133</td>
<td>0.314812876</td>
<td>0.099552572</td>
</tr>
<tr>
<td>16</td>
<td>16.75015184</td>
<td>0.603019396</td>
<td>0.190691476</td>
</tr>
<tr>
<td>17</td>
<td>17.6893085</td>
<td>0.543811045</td>
<td>0.171968152</td>
</tr>
<tr>
<td>18</td>
<td>18.6605088</td>
<td>0.437365174</td>
<td>0.138307012</td>
</tr>
<tr>
<td>19</td>
<td>19.71986894</td>
<td>0.490421118</td>
<td>0.155084775</td>
</tr>
<tr>
<td>20</td>
<td>20.67424482</td>
<td>0.73554232</td>
<td>0.232598905</td>
</tr>
<tr>
<td>21</td>
<td>21.54046434</td>
<td>0.590449713</td>
<td>0.186716594</td>
</tr>
<tr>
<td>22</td>
<td>22.52164731</td>
<td>0.813883568</td>
<td>0.257372582</td>
</tr>
<tr>
<td>23</td>
<td>23.71546228</td>
<td>1.365370771</td>
<td>0.431768149</td>
</tr>
<tr>
<td>24</td>
<td>24.73395867</td>
<td>1.800295253</td>
<td>0.56903346</td>
</tr>
<tr>
<td>25</td>
<td>26.26378348</td>
<td>2.077637538</td>
<td>0.657006677</td>
</tr>
<tr>
<td>26</td>
<td>27.80273707</td>
<td>3.129247102</td>
<td>0.98955482</td>
</tr>
<tr>
<td>27</td>
<td>28.89194071</td>
<td>3.179341148</td>
<td>1.005395949</td>
</tr>
<tr>
<td>30</td>
<td>33.7967537</td>
<td>5.449017256</td>
<td>1.723130554</td>
</tr>
<tr>
<td>40</td>
<td>38.45651054</td>
<td>5.804504027</td>
<td>1.835545341</td>
</tr>
<tr>
<td>50</td>
<td>46.92597794</td>
<td>5.665028156</td>
<td>1.791439198</td>
</tr>
<tr>
<td>60</td>
<td>58.38448082</td>
<td>10.80952701</td>
<td>3.418272579</td>
</tr>
</tbody>
</table>

6.3 Sensor characterisation – compass CMPS02

As discussed in an earlier chapter, the Compass was set up so that it used a PWM output with a pulse width proportional to the orientation of the compass (Figure 6-13). The software picks up the positive pulse width using the PWM input pin 75 and performs a few minor sanity checks before returning the orientation detected. The overall accuracy of the system is excellent; however there is an intermittent problem in that sometimes the processor is unable to catch the falling edge of the pulse in time resulting in a bogus reading.
There are a number of checks in the software with the intention of minimising this error as much as possible, including an automatic re-read compass action if the compass does not return a value between 1 and 370.

Although the compass works, the timer capture system on-board the MC68VZ328 is not really equipped for detecting pulses shorter than 10ms and the software was adjusted to maximise the measurement potential of the MC68VZ328.

The measurements returned from the compass were compared with an orienteering compass and the results that are returned are almost identical to the ones returned by the compass with a deviation of 1° to 2° from the norm.

Figure 6-13 Plot of compass output, with positive pulse width proportional to the orientation of the compass
6.4 Sensor characterisation – Humanatsu flame detector

The Humanatsu Flame Detector is a very sensitive device capable of detecting a flame from up to 5 metres. It has been engineered with the placement of a capacitor that when a flame source is detected, it drives its output pin low until the flame is not detected any more. The software running on the MC68VZ328 is capable of detecting this signal but because the signal does not vary as distance to and from the flame increases and decreases, it is impossible to detect the position of this flame with the Humanatsu Flame Detector alone. A scanning algorithm is implemented as soon as the flame is detected.

---

Figure 6-14 Plot of Humanatsu flame detector output on a digital oscilloscope
6.5 Concluding remarks

This chapter discussed the characterisation of the sensors in an effort to determine how much they contributed to the awareness of the system. It can be seen from the experiments that the Devantech ultrasonic sensors are accurate up to 50 cm. Based on this information it is concluded that these sensors can be used for obstacle avoidance and provide the robotic system with spatial awareness. The chapter also discussed the characterisation of the Sharp IR sensors and it was shown that the method devised and implemented in the prototype system is more accurate than any existing method. The plot shown in Figure 6-11 shows that there is a substantial difference between the method devised in the course of this work and the previous methods. This improvement was made without affecting the execution time of the software or the processing capabilities of the hardware. These results make the Sharp IR Sensor ideal for close range mapping due to the highly accurate distance measurements it is capable of.

Chapter 7 summarises the work that was carried out as part of the project and discusses a future direction for the work.
Chapter 7 Summary and experience gained

Over the course of the design of the robot prototype platform a number of ideas were trialed and valuable experience was gained. This chapter summarises the work carried out, the contribution this project has made and possible paths for the evolution of the robotic prototype platform into a fully aware system.

7.1 Context

The overall aim of this work was to develop a prototype platform on which awareness strategies could be developed and evaluated.

7.2 Summary

The uCLinux controlled robotic prototype platform developed during this project is an ideal system to develop future projects on. It consists of a powerful processor running stable software, a robust chassis capable of carrying a payload of 14kg, a wireless communications interface and a number of proximity and environmental sensors. This section summarises the work carried out and describes the contribution made with respect to each section.

7.2.1 Review of the research in robotic awareness

The field of robotics awareness research encapsulates many disciplines yet there is no singular definition of awareness for a robotics system. During the course of this work a detailed investigation was carried out to define awareness in the context of robotics systems and to review the state of the art research which is currently being undertaken at various institutions world-wide. The implication of increased human-robot interaction (HRI) was...
investigated with the determination that as HRI becomes more commonplace the need for robotic systems with heightened awareness becomes greater. A review of best practices for system design was carried out which described key methods for the design of a reliable system.

Best practice methods for hardware design were identified such as;

- The authoring of a DFMEA/DFMECA where the failure mode of every component in the hardware system is evaluated,
- HASS/HALT testing where the system is put through an accelerated test in order to identify the weakest components in the system,
- Wiebull analysis of the system where the expected life of the system is calculated.

Methods for the design of reliable software such as the adaptation of the FDD methodology and continuous integration were also discussed. These items were identified as critical for the design of a system that is required to operate autonomously and reliably without human supervision.

This research was necessary in order to clearly understand the scope of the work and define the requirements for a robotic prototype platform on which awareness strategies can be implemented. This work is detailed in Chapter 1 and can be used as an accurate reference for future work where an understanding of existing research and awareness in terms of a robotic system is required.

7.2.2 Review of the sensor technology

A review of existing sensor technology was completed and the most applicable sensors for the development of an aware robotic prototype platform were identified. The review is divided into internal feedback sensors and external interaction sensors both of which...
contribute to the overall awareness of a robotic system. The sensors reviewed range from thermocouples to LVDTs to sensors capable of detecting toxic gases such as carbon monoxide and ammonia. A comparison of these sensors from different manufacturers was also completed and this information is included in Chapter 2. This review contributes to the goal of developing an aware robotic system by defining the sensors essential characteristics and describing the most technologically advanced sensors that can be integrated into a robotic system for the development of awareness strategies.

7.2.3 Design of prototype robotic platform

Taking into consideration the review of existing research, best practice design methods and sensor technologies a number of design options were considered for the development of the prototype platform. The final system design combines the benefits of a centralised and a distributed system by using a powerful microprocessor running a stable software operating system. It is capable of processing sensor information and transmitting this information over a wireless communications interface to a computer for further processing. The software is designed so that additional hardware can be added or removed with ease. The microprocessor has 76 GPIO ports that can be used for sensor hardware and the mobile robot chassis is capable of carrying up to 14kg allowing for the addition of larger and more complex sensors. The detail of the design is included in Chapter 3 but the key contribution from this work is the:

- Design of a robotic system that is capable of operating as a distributed and/or a centralised system,

- Design of a robotic hardware system that has the potential to evolve without major re-design,
• Design of a framework within embedded Linux whereby sensory hardware can be easily integrated into the robotic system for future development.

7.2.4 Hardware

The main goals of the hardware development were to:

• Develop a hardware system on which awareness strategies can be developed,
• Develop a hardware system which is capable of expanding for future work,
• Develop a hardware system that can be reproduced easily,
• Develop a hardware system that is reliable and robust.

To achieve these goals the hardware components tested and evaluated were predominately 'off the shelf' components. This allowed the rapid development and evaluation of potential platforms. Chapter 4 describes the different hardware that was evaluated and implemented over the course of the work.

7.2.5 Software

The software was developed in tandem with the hardware and was required to change to facilitate changes in hardware. Over the course of the project the software operating system (OS) changed from RCX firmware to LegOS and finally µCLinux and although the OS changed the OS requirements did not. The OS is required to be portable, robust, reliable and expandable for future work. The software for the µCLinux platform was developed in C and Chapter 5 describes the steps required in software to add hardware components to this platform.

The system design outlines a requirement for the system to be capable of operating as a distributed or centralised system. To fulfil this requirement the development of a Windows
based application capable of interfacing with the \( \mu \)CLinux platform was necessary. The detail of the software evaluated and implemented over the course of the work is discussed in Chapter 5.

**7.2.6 Development of prototype robotic platform**

The development of the hardware and software for the prototype robotic platform was carried out in parallel with the main goals being the,

- Implementation of a stable, robust and portable robotic hardware system on which awareness strategies can be developed,
- Implementation of a hardware system that has the capabilities to expand and evolve in future projects,
- Implementation of a stable, robust and portable robotic software system on which awareness strategies can be developed,
- Implementation of a software system that is capable of evolving without the necessity for cumbersome redesign in future work.

![Robotic prototype platform](image)

*Figure 7-1 [a] Robotic prototype platform on Nikkon remote control car [b] on Max99 chassis.*
7.2.7 Testing of prototype robotic platform

In order to determine if the system met the criteria of a robotic platform on which awareness strategies could be developed testing was carried out. This testing included the characterisation of the sensors used to determine their level of contribution to the platform as a whole. Chapter 6 describes the experiments carried out and outlines the differences in sensor accuracy. One of the major developments of this experimentation was the discovery of a more accurate method of manipulating the raw data from the Sharp IR GP2D02 sensor. It was shown that there is a substantial improvement in distance measurement calculation from the methods previously employed. It was shown that the new method has an error of ±0.1cm at 10cm compared with ±0.4cm with the old method. The difference is more noticeable as the distance increases with the new method having an error of ±2cm at 60cm whilst the old method has an error of ±20cm at 60cm. This finding enables the Sharp IR sensor to be used as a close range mapping device due to its highly accurate distance measurements.

7.2.8 Demonstrating awareness

It can be seen from the experiments that the robot platform is capable of spatial, positional and environmental awareness based on the existing hardware and software.

This was demonstrated by running the robot in different enclosed environments such as enclosure discussed in Chapter 5 and the laboratory. The prototype platform is capable of navigating through these environments avoiding obstacles hence showing spatial awareness. The LED bank on the uCevolution board was used to demonstrate the platforms reaction to the environmental sensors. It was shown that the robot platform was capable of reacting to certain environmental conditions such as darkness or the presence of a flame in
the environment. The software can be expanded to instruct the robot to raise an alarm or to investigate further if it detects changes to the environment. The robot is also capable of positional awareness and this was demonstrated by comparing the robots heading with an orienteering compass.

7.2.9 Design limitations

There is one major design limitation that should be considered for future design. The Bluetooth™ wireless system is limited to a 100m radius from base-station. Therefore an alternative wireless solution should be considered for development of a system that is required to operate in a greater envelope.

7.3 Critique

In industry, system designs will typically go through a number of iterations before being released. These iterations are necessary to address issues identified during the testing phase so that the end product is of a high standard. Once the system is released, work based on the experiences gained during the initial development is begun so as to improve the released system.

The delivered system fulfils the original design requirements and it was shown during the testing phase that it can be used for the development of awareness strategies. It contains a powerful processor, a wireless communications interface and a stable software system. The system as a whole is easily reproducible and it is possible to expand the hardware and software as required. However there were some issues identified during the development that should be addressed in order to refine the system.
From a hardware point of view the interface electronics for the sensors and motors was implemented on copper strip-board and connected to the microprocessor with single core wire. The issue with this is twofold. Firstly this requires the building of a board for each hardware component that is added and secondly the possibility of intermittent hardware disconnects caused by vibration during the movement of the robot is increased. With some foresight these issues can be rectified by the design and development of an interface PCB which provides a reliable connection to the microprocessor ports and a means of adding new hardware. Secondly the mounting of the sensors on the prototype platform is not ideal and although it is adequate for the purpose of evaluating the system, the system would be improved by the development of sensor mounting brackets for the chassis. The power supply for the prototype system was another issue identified during the course of the work. In order to improve the system the possibility of using rechargeable lithium-ion batteries should be considered and the implementation of a dedicated power supply PCB should be investigated.

From a software point of view the system is easily expanded, reliable and robust. However the GUI should be refined to be more aesthetically pleasing.

These improvements should be implemented as part of the future work discussed in the next section.

7.4 Future work

Future work lies in several areas. First and foremost further algorithms need to be implemented on the hardware to explore the depth of awareness that can be realised in the existing system. As new algorithms are developed additional sensors can be integrated to the robotic system. As discussed in the last section, a PCB which maps to the • • •
microprocessors GPIO ports should be designed to accommodate new sensory hardware as it is required.

Other possible future projects are,

- The development of software to integrate the liquid crystal display (LCD) into the robotic system for HRI,
- The integration of a GPS system for outdoor operation and development of positional awareness algorithms,
- An investigation into cellular communications as an alternative to Bluetooth™ as a communications interface,
- The software implementation of frontier based exploration algorithms,
- The exploration of a distributed system by the development of a robot team,
- The integration of rechargeable lithium-ion batteries as a means of powering the mobile robot,
- The development of a vision system and implementation of image compression algorithms on the embedded Linux platform,
- The development of mapping algorithms as part of a distributed system,
- The integration of gaseous sensors discussed in Chapter 2.

With these possible areas of future work defined, this area of research should result in a mobile robot that is aware of its mission, aware of what is around it and aware of its position in the world, finally evolving into a system with executive awareness – the ultimate goal of every robotocist.
7.5 Conclusions

The aim of the project was to develop a proto robotic platform on which awareness strategies could be developed. The combination key system components discussed in this chapter and the detail contained in this thesis that the prototype robotic system fulfils this requirement. The core of the platform consists of a powerful microprocessor running a leading edge operating system. The platform has enough memory to run awareness algorithms locally and has a wire communications interface to run more computationally intensive algorithms on a computer. This allows the system to operate as a distributed or centralised system. It can be added easily to the robotic platform in hardware and software allowing them to be used for the evaluation of different awareness strategies. A more accurate method of calculating the distance measurement obtained from the Sharp IR sensor was discovered during the sensor characterisation. This improvement was achieved without affecting the execution time of the software or impacting the processing capability of the hardware. It was also shown during the characterisation of the sensors that it is capable of spatial, positional and environmental awareness with a high degree of accuracy and it is hoped that this system is used to develop some if not all of the ped projects outlined in this chapter in the future.
## Appendix A – Datasheet Comparison of microcontrollers, microprocessors and Digital Signal Processors

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>MICROCHIP</th>
<th>MICROCHIP</th>
<th>MICROCHIP</th>
<th>MICROCHIP</th>
<th>MICROCHIP</th>
<th>FREESCALE</th>
<th>MICROCHIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoHS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Description</td>
<td>MCU, 8bit, 7K FLASH, 28S025</td>
<td>MCU, 8bit, 7K FLASH, 28S025</td>
<td>MCU, 8bit, 14K FLASH, 28S025</td>
<td>MCU, 14K FLASH, 28S025</td>
<td>MCU, 14K FLASH, 28S025</td>
<td>MCU, 14K FLASH, 28S025</td>
<td>MCU, 14K FLASH, 28S025</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>Flash Memory Size: 7KB</td>
<td>Flash Memory Size: 14KB</td>
<td>Flash Memory Size: 3KB</td>
<td>Flash Memory Size: 3KB</td>
<td>Flash Memory Size: 3KB</td>
<td>Flash Memory Size: 3KB</td>
<td>Flash Memory Size: 3KB</td>
</tr>
<tr>
<td>EEPROM Memory</td>
<td>EEPROM Memory Size: 256KB</td>
<td>EEPROM Memory Size: 256KB</td>
<td>EEPROM Memory Size: 256KB</td>
<td>EEPROM Memory Size: 256KB</td>
<td>EEPROM Memory Size: 256KB</td>
<td>EEPROM Memory Size: 256KB</td>
<td>EEPROM Memory Size: 256KB</td>
</tr>
<tr>
<td>RAM Memory</td>
<td>RAM Memory Size: 256KB</td>
<td>RAM Memory Size: 512KB</td>
<td>RAM Memory Size: 512KB</td>
<td>RAM Memory Size: 512KB</td>
<td>RAM Memory Size: 512KB</td>
<td>RAM Memory Size: 512KB</td>
<td>RAM Memory Size: 512KB</td>
</tr>
<tr>
<td>I/O Lines</td>
<td>No. of I/O Lines: 25</td>
<td>No. of I/O Lines: 25</td>
<td>No. of I/O Lines: 25</td>
<td>No. of I/O Lines: 25</td>
<td>No. of I/O Lines: 25</td>
<td>No. of I/O Lines: 25</td>
<td>No. of I/O Lines: 25</td>
</tr>
<tr>
<td>ADC</td>
<td>No. of ADC Inputs: 11</td>
<td>No. of ADC Inputs: 11</td>
<td>No. of ADC Inputs: 11</td>
<td>No. of ADC Inputs: 11</td>
<td>No. of ADC Inputs: 11</td>
<td>No. of ADC Inputs: 11</td>
<td>No. of ADC Inputs: 11</td>
</tr>
<tr>
<td>Timers</td>
<td>No. of Timers: 5</td>
<td>No. of Timers: 5</td>
<td>No. of Timers: 5</td>
<td>No. of Timers: 5</td>
<td>No. of Timers: 5</td>
<td>No. of Timers: 5</td>
<td>No. of Timers: 5</td>
</tr>
<tr>
<td>PWM Channels</td>
<td>No. of PWM Channels: 5</td>
<td>No. of PWM Channels: 5</td>
<td>No. of PWM Channels: 5</td>
<td>No. of PWM Channels: 5</td>
<td>No. of PWM Channels: 5</td>
<td>No. of PWM Channels: 5</td>
<td>No. of PWM Channels: 5</td>
</tr>
<tr>
<td>Clock Frequency</td>
<td>Clock Frequency: 32MHz</td>
<td>Clock Frequency: 32MHz</td>
<td>Clock Frequency: 32MHz</td>
<td>Clock Frequency: 32MHz</td>
<td>Clock Frequency: 32MHz</td>
<td>Clock Frequency: 32MHz</td>
<td>Clock Frequency: 32MHz</td>
</tr>
<tr>
<td>Interface</td>
<td>Type: EUSART, I2C, SPI</td>
<td>Type: EUSART, I2C, SPI</td>
<td>Type: EUSART, I2C, SPI</td>
<td>Type: EUSART, I2C, SPI</td>
<td>Type: EUSART, I2C, SPI</td>
<td>Type: EUSART, I2C, SPI</td>
<td>Type: EUSART, I2C, SPI</td>
</tr>
<tr>
<td>Voltage Supply (max)</td>
<td>Max Supply Voltage: 5.5V</td>
<td>Max Supply Voltage: 5.5V</td>
<td>Max Supply Voltage: 5.5V</td>
<td>Max Supply Voltage: 5.5V</td>
<td>Max Supply Voltage: 5.5V</td>
<td>Max Supply Voltage: 5.5V</td>
<td>Max Supply Voltage: 5.5V</td>
</tr>
<tr>
<td>Termination</td>
<td>Termination Type: SMD</td>
<td>Termination Type: SMD</td>
<td>Termination Type: SMD</td>
<td>Termination Type: SMD</td>
<td>Termination Type: SMD</td>
<td>Termination Type: SMD</td>
<td>Termination Type: SMD</td>
</tr>
<tr>
<td>Case</td>
<td>Case Style: S01C</td>
<td>Case Style: S01C</td>
<td>Case Style: S01C</td>
<td>Case Style: S01C</td>
<td>Case Style: S01C</td>
<td>Case Style: S01C</td>
<td>Case Style: S01C</td>
</tr>
</tbody>
</table>

### Microcontrollers Specifications:
- **PIC16F1933-1/40**: 8-bit Microcontroller, 7K FLASH, 28S025
- **PIC16F1933-1/44**: 8-bit Microcontroller, 7K FLASH, 28S025
- **PIC16F1936-1/45**: 8-bit Microcontroller, 14K FLASH, 28S025
- **PIC16F505-1/20**: 8-bit Microcontroller, 14K FLASH, 28S025
- **PIC16F628-20/40**: 8-bit Microcontroller, 14K FLASH, 28S025
- **MCF52277CV160**: 8-bit Microcontroller, 28S025
- **PIC18C452-1/4**: 8-bit Microcontroller, 28S025

### Memory Specifications:
- **Flash Memory**: 7KB, 14KB, 3KB
- **EEPROM Memory**: 256KB
- **RAM Memory**: 256KB, 512KB

### Interface Specifications:
- **Type**: EUSART, I2C, SPI

### Voltage Supply Specifications:
- **Min Supply Voltage**: 1.8V, 2V, 3V, 1.4V, 4.2V
- **Max Supply Voltage**: 5.5V, 5.5V, 5.5V, 5.5V, 5.5V

### Termination Type:
- **SMD**

### Operating Temperature Range:
- **-40°C to +85°C**
<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Freescale Semiconductor</th>
<th>FREESCALE SEMICONDUCTOR</th>
<th>FREESCALE SEMICONDUCTOR</th>
<th>FREESCALE SEMICONDUCTOR</th>
<th>FREESCALE SEMICONDUCTOR</th>
<th>FREESCALE SEMICONDUCTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part Number</td>
<td>MC68V2328</td>
<td>MC68378EVRANG</td>
<td>MC68343EVRAGDB</td>
<td>MC68020EH16E</td>
<td>MC68020RC2SE</td>
<td>MC68020RC33E</td>
</tr>
<tr>
<td>RoHS</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Description</td>
<td>Integrated Processor</td>
<td>Embedded Networking</td>
<td>Embedded Networking</td>
<td>Microprocessor</td>
<td>Microprocessor</td>
<td>32-Bit Microprocessor</td>
</tr>
<tr>
<td></td>
<td>Processor IC</td>
<td>Processor IC</td>
<td>Processor IC</td>
<td>Processor Type: M680X0</td>
<td>Processor Type: M680X0</td>
<td>Processor Type: M680X0</td>
</tr>
<tr>
<td>Bits</td>
<td>16</td>
<td>No. of Bits:32</td>
<td>No. of Bits:32</td>
<td>No. of Bits:32</td>
<td>No. of Bits:32</td>
<td>No. of Bits:32</td>
</tr>
<tr>
<td>Frequency</td>
<td>33MHz</td>
<td>Frequency Typ:800MHz</td>
<td>Frequency Typ:400MHz</td>
<td>Frequency Typ:33MHz</td>
<td>Frequency Typ:33MHz</td>
<td>Frequency Typ:33MHz</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>3.3V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manufacturer</td>
<td>ANALOG DEVICES</td>
<td>ANALOG DEVICES</td>
<td>ANALOG DEVICES</td>
<td>TEXAS INSTRUMENTS</td>
<td>TEXAS INSTRUMENTS</td>
<td>ANALOG DEVICES</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Part Number</td>
<td>ADSP-2101BPZ-100</td>
<td>ADSP-2184BSTZ-160</td>
<td>ADSP-BF531BSTZ2400</td>
<td>TMS320LF2403APAGA</td>
<td>TMS320VC33PGE-120</td>
<td>ADSP-BF537KB6Z-68V</td>
</tr>
<tr>
<td>RoHS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Description</td>
<td>MICROCOMPUTER, 16BIT 25MIPS, SMD</td>
<td>MICROCOMPUTER, 16BIT 40MIPS, SMD</td>
<td>16BIT DSP BLACKFIN, 531BSTZ400</td>
<td>IC, SM 16-BIT DSP W/FLASH, WAFLE80</td>
<td>DSP PROCESSOR, 320VC33, LQFP144</td>
<td>Digital Signal Processor IC</td>
</tr>
<tr>
<td>Type</td>
<td>DSP Type:Fixed Point</td>
<td>DSP Type:Core</td>
<td>DSP Type:Fixed Point</td>
<td>DSP Type:Fixed Point</td>
<td>DSP Type:Fixed Point</td>
<td>DSP Type:Fixed Point</td>
</tr>
<tr>
<td>Frequency</td>
<td>Type Core Frequency:25MHz</td>
<td>Type Core Frequency:40MHz</td>
<td>Type Core Frequency:400MHz</td>
<td>Type Core Frequency:60MHz</td>
<td>Type Core Frequency:600MHz</td>
<td>Type Core Frequency:600MHz</td>
</tr>
<tr>
<td>RAM Memory</td>
<td>RAM Memory Size:20KB</td>
<td>RAM Memory Size:14KB</td>
<td>RAM Memory Size:1KB</td>
<td>RAM Memory Size:136KB</td>
<td>Memory Size, RAM:1,032MB</td>
<td>Memory Size, RAM:1,032MB</td>
</tr>
<tr>
<td>External Supported Memory</td>
<td>External Supported Memory:EPROM</td>
<td>External Supported Memory:SDRAM, SRAM, FLASH, ROM</td>
<td>External Supported Memory:SDRAM, SRAM, FLASH, ROM</td>
<td>External Supported Memory:SDRAM, SRAM, FLASH, ROM</td>
<td>External Supported Memory:SDRAM, SRAM, FLASH, ROM</td>
<td>External Supported Memory:SDRAM, SRAM, FLASH, ROM</td>
</tr>
<tr>
<td>Interface</td>
<td>Interface Type:Serial, UART</td>
<td>Interface Type:SPI, UART</td>
<td>Interface Type:CAN, SCI, SPI</td>
<td>Interface Type:UART</td>
<td>Interface:SPI</td>
<td>Interface:SPI</td>
</tr>
</tbody>
</table>
Bibliography


[79] Kurfess, Thomas R. ROBOTICS AND AUTOMATION HANDBOOK.

http://www.cmucam.org/.
http://www.real-robots.co.uk.


http://bricxcc.sourceforge.net/nqc/.


http://www.schoeldgen.de.