

**EXPERIMENTAL EVALUATION OF HYBRID SOFTWARE ENGINEERING
METHODOLOGY FOR EMBEDDED FIRMWARE DEVELOPMENT ON
INTELLIGENT MOBILE ROBOT**

by

Dayang Norhayati Abang Jawawi, Safaai Bin Deris, Rosbi Bin Mamat, Radziah
Mohamed, Mohd Ridzuan Bin Ahmad, Ahmad Zariman Bin Abdul Majid and Ahmad
Ruzaimie Bin Abdul Rashid

Faculty of Computer Science and Information System
Universiti Teknologi Malaysia

2004

Hyperlink to the report contents, please click in the hyperlink to open the report.

CONTENTS	HYPERLINK
BORANG PENGESAHAN LAPORAN	Borang pengesahan
TITLE ABSTRACT ABSTRAK ACKNOWLEDGMENT CONTENTS	preface
CHAPTER 1 - Introduction	Chapter1
CHAPTER 2 - A Hybrid Software Engineering Methodology For Small-Scale Embedded Firmware Development	Chapter2
CHAPTER 3 - User Requirements Definition Phase for IMR71848 Intelligent Mobile Robot Software	Chapter3
CHAPTER 4 - Software Requirements Specification for IMR71848 Intelligent Mobile Robot Software	Chapter4
CHAPTER 5 - Evaluation of Hybrid Software Engineering Methodology for Development of Embedded Firmware For Intelligent Autonomous Mobile Robot	Chapter5
APPENDIX A - Intelligent Mobile Robot Software Structure and Behaviour	AppendixA

UNIVERSITI TEKNOLOGI MALAYSIA

BORANG PENGESAHAN LAPORAN AKHIR PENYELIDIKAN

TAJUK PROJEK :

EXPERIMENTAL EVALUATION OF HYBRID SOFTWARE ENGINEERING
METHODOLOGY FOR EMBEDDED FIRMWARE DEVELOPMENT ON
INTELLIGENT MOBILE ROBOT .

Saya DAYANG NORHAYATI ABANG JAWAWI .
(HURUF BESAR)

Mengaku membenarkan Laporan Akhir Penyelidikan ini disimpan di Perpustakaan Universiti Teknologi Malaysia dengan syarat-syarat kegunaan seperti berikut :

1. Laporan Akhir Penyelidikan ini adalah hakmilik Universiti Teknologi Malaysia
2. Perpustakaan Universiti Teknologi Malaysia dibenarkan membuat salinan untuk tujuan rujukan sahaja.
3. Perpustakaan dibenarkan membuat penjualan salinan Laporan Akhir Penyelidikan ini bagi kategori TIDAK TERHAD
4. * Sila tandakan (/)

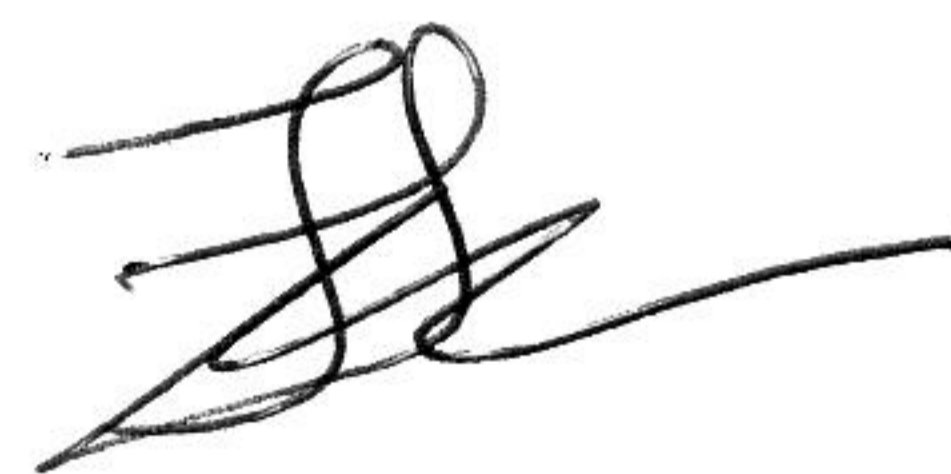
SULIT

(Mengandungi maklumat yang berdarjah keselamatan atau Kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

TERHAD

(Mengandungi maklumat TERHAD yang telah ditentukan oleh Organisasi/badan di mana penyelidikan dijalankan)

TIDAK
TERHAD



TANDATANGAN KETUA PENYELIDIK

DAYANG NORHAYATI ABANG JAWAWI
Ketua Projek Vot 71848
Fakulti Sains Komputer & Sistem Maklumat
UTM, Skopak, Johor

Tarikh : 24 Feb. 2004 .

EXPERIMENTAL EVALUATION OF HYBRID SOFTWARE ENGINEERING
METHODOLOGY FOR EMBEDDED FIRMWARE DEVELOPMENT ON
INTELLIGENT MOBILE ROBOT

(PENILAIAN SECARA UJIKAJI METADOLOGI KEJURUTERAN PERISIAN
UNTUK PEMBANGUNAN PERISIAN TERBENAM UNTUK ROBOT
BERGERAK PINTAR)

DAYANG NORHAYATI ABANG JAWAWI
SAFAAI BIN DERIS
ROSBI BIN MAMAT
RADZIAH MOHAMED
MOHD RIDZUAN BIN AHMAD
AHMAD ZARIMAN BIN ABDUL MAJID
AHMAD RUZAIMEE BIN ABDUL RASHID

FACULTY OF COMPUTER SCIENCE AND INFORMATION SYSTEM
UNIVERSITI TEKNOLOGI MALAYSIA

ABSTRACT

Software development for embedded real-time systems is very much different from the traditional data processing systems due to non-functional requirements such as dependability and the presence of timing constraints. Special tools and appropriate methodologies are therefore highly desirable for the development of embedded real-time software. Previous work on developing control firmware of Universiti Teknologi Malaysia wall climbing robot, have faced with the problem of adopting a single software engineering methodology for developing the robot control firmware, a methodology called hybrid methodology was proposed. This software engineering methodology was tested on a wall climbing robot system under hardware-in-the-loop simulation and was found to be effective and the software produced, conformed to the requirements. However, a few questions are still need to be answered and measured in order to fully test the effectiveness of the hybrid methodology on actual embedded system. The main objective of this research is to evaluate the hybrid software engineering methodology for developing control firmware on actual system. An intelligence mobile robot was developed to serve as embedded real-time system platform. Two main outputs of this research is evaluation results on strengths and weaknesses of the hybrid software engineering methodology and identification of the phases in the methodology, which require automation and can be automated.

ABSTRAK

Proses pembangunan perisian bagi sistem terbenam masa-nyata berbeza dengan pembangunan perisian bagi sistem pemprosesan data traditional, ini disebabkan oleh keperluan bukan fungsi seperti keboleharapan dan kekangan pemaasan. Oleh itu, peralatan istimewa dan metodologi yang bersesuaian sangat diperlukan untuk pembangunan perisian terbenam masa-nyata. Berdasarkan kerja pembangunan perisian kawalan bagi satu robot memanjat dinding, yang dibangunkan di Universiti Teknologi Malaysia, menghadapi masalah menggunakan hanya satu metodologi pembangunan perisian. Berdasarkan kajian tersebut, satu metodologi hibrid telah dicadangkan. Metodologi kejuruteraan perisian ini diuji ke atas robot memanjat dinding dengan menggunakan penyelakuan perkakasan-dalam-gelung dan hasil daripada pengujian menunjukkan keberkesanan penggunaan metodologi hibrid yang dapat menghasilkan satu perisian yang menepati keperluan. Walau bagaimanapun, beberapa persoalan masih perlu dijawab dan pengukuran lanjutan perlu dibuat untuk menguji keberkesanan metodologi hibrid tersebut, pada sistem terbenam yang sebenarnya. Sebuah robot bergerak pintar telah dibangunkan untuk dijadikan dasar kepada sistem masa-nyata terbenam. Dua output utama penyelidikan ini ialah keputusan penilaian kekuatan dan kelemahan metodologi kejuruteraan perisian tersebut dan fasa dalam metodologi yang telah dikenalpasti untuk diautomasikan.

ACKNOWLEDGEMENTS

We would like to extend our appreciation to Research Management Center, Universiti Teknologi Malaysia for funding this project

CONTENTS

TITLE	i
ABSTRACT	ii
ABSTRAK	iii
ACKNOWLEDGEMENTS	iv
CONTENTS	v
1. INTRODUCTION	1
Overview	1
General Problem Statement	2
Objectives	3
Scope of the Study	4
Report Outline	4
2. A HYBRID SOFTWARE ENGINEERING METHODOLOGY FOR SMALL-SCALE EMBEDDED FIRMWARE DEVELOPMENT	6
Title	6

Abstract	6
Introduction	6
Specification of Wall-Climbing Robot Systems	7
The Hybrid Methodology	8
WCR Requirement Analysis	9
The WCR Control Firmware Design	11
Conclusion	13
References	13
3. USER REQUIREMENTS DEFINITION PHASE FOR IMR71848 INTELLIGENT MOBILE ROBOT SOFTWARE	14
Title	15
Table of Contents	16
Introduction	16
General Description	19
Specific Requirements	25
List of User Requirements	32
List of User Requirements to be Confirmed	33
4. SOFTWARE REQUIREMENTS SPECIFICATION FOR IMR71848 INTELLIGENT MOBILE ROBOT SOFTWARE	34
Title	34
Table of Contents	35
Introduction	36
General Description	39
Specific Requirements	44

5. EVALUATION OF HYBRID SOFTWARE ENGINEERING METHODOLOGY FOR DEVELOPMENT OF EMBEDDED FIRMWARE FOR INTELLIGENT AUTONOMOUS MOBILE ROBOT	58
	58
Title	58
Abstract	58
Introduction	59
Intelligent Mobile Robot Specification	60
Hybrid Software Engineering Methodology	
Evaluation of The Hybrid Se Methodology For IMR71848	61
Firmware Analysis And Design	65
Conclusion	66
References	
APPENDIX A	67

CHAPTER I

INTRODUCTION

1.1 Overview

A mobile robot is an autonomous system capable of traversing a terrain, performs its designated tasks, senses its environment and intelligently reacts to it. As the complexity and functionality of the robot is increased, such as adding more sensors to the robot so as to increase its reactivity and intelligence, designing and developing control software for this type of robot can be very difficult and a challenging task.

Issues related to real-time control, embedded system and artificial intelligence are involved in the mobile robot software development process. This type of software must be developed with proper software methodology or well-defined development process. Typically, the software or firmware is embedded in the onboard controller. To provide intelligence and reactive action, the robot firmware must sense its environment with multiple sensors and process the information and taking actions in real-time.

In a real-time system such as a mobile robot, the correctness of the system depends not only on the logical results, but also on the time at which the results are produce. A mobile robot software system is inherently concurrent and multitasking since it has to react to and process numerous events simultaneously. Embedded system is a system, which contains microcomputer as a component to do the processing and control of its environment, but the user does not see the system as a computer. Most

embedded systems are real-time systems, in this report such systems are called Embedded Real-Time Systems (ERTS). Software development for ERT systems is very much different from the traditional data processing systems due to non-functional requirements such as dependability and the presence of hard timing constraints.

In order to increase the software productivity, maintainability and flexibility in developing ERT robot software, a proper software engineering needs to be considered. Proper Software Engineering (SE) methodology ensures the software development process is manageable, and that reliable and correct program is constructed.

The purpose of software engineering methodology is to promote a certain approach to solve software problems. A number of software engineering methodologies already exist, which targeted toward real-time systems such as Modular Approach to Software Construction (MASCOT), Design Approach for Real-Time Systems (DARTS), Structured Analysis and Design for Real-Time Systems (SDRTS), Unified Modeling Language for Real-Time (UML for Real-time) and Hard Real-Time Hierarchical Object Oriented Design (HRT-HOOD). Not all of these software engineering methodologies will support the software development for small-scale embedded system.

1.2 General Problem Statement

Our previous work on developing control firmware for UTM Wall Climbing Robot (WCR) have found that adopting of a single complete methodology for ERT firmware development is not beneficial to the designers due to the some deficiencies such as inadequately for presenting hard timing constraints, lacking in capability to represent control oriented systems, methodology complexity and the requirement of special support tool to implement a methodology. Faced with the problem of adopting a single methodology for developing small-scale ERT systems, a methodology called hybrid methodology was developed.

A hybrid methodology is proposed for the WCR control firmware development. In the proposed hybrid methodology, several suitable notations and diagrams taken from Ward-Mellor Structured Development for Real-Time Systems (Ward-Mellor), Unified Modeling Language for Real-Time (UML-RT) and Hard Real-Time Hierarchical Object Oriented Design (HRT-HOOD) methodologies were used to specify and design the robot control firmware. The idea of combining several notation and tools from different software engineering methodologies for developing real-time system is not new. The main advantage of adopting the hybrid method is that any appropriate notation and diagram can be chosen from different methodologies for functional and non-functional specification and design. The main disadvantage is that no CASE tool support is available to assist the use of the hybrid method. However, due to the scale of WCR project this can be handled manually.

This methodology was tested on UTM-WCR under hardware-in-the-loop simulation. From the test results it was found that by following the proposed hybrid method for developing the control firmware, a firmware that conformed to the requirements set could be developed successfully. The quality of the WCR control firmware reflects the effectiveness of the hybrid software engineering methodology and the software tools used in the methodology. However, a few questions are still need to be answered in the real-robot environment. In order to fully test the effectiveness of the hybrid methodology on actual system, measurement and experimental evaluation need to be performed to the outputs of the methodology.

1.3 Objectives

1. To test the previously developed Hybrid Software Engineering (SE) methodology for developing control firmware on real system.
2. To reveal the strengths and weaknesses of the hybrid SE methodology.
3. To suggest improvement and modification on the hybrid SE methodology.

1.4 Scope of the Study

The scope of this research was limited to the following;

1. The work on this project will be mainly on software engineering aspect of intelligence mobile robot. Robot building is not part of this project.
2. Software development process will be based on previously developed Hybrid SE methodology. Other methodology will not be evaluated in this study.

1.5 Report Outline

Chapter II discusses the hybrid software engineering methodology that was proposed for developing the WCR control firmware. In this Chapter, the modeling and the design of the WCR firmware using the hybrid real-time software methodology will be presented.

Chapter III describes the specification of the intelligent mobile robot used as the platform of the methodology evaluation. This specification was prepared based on a study on performed by the 71848 group members and documented as a research group's technical report titled "*Intelligent Mobile Robot Software Structure and Behaviour*". The report is enclosed in Appendix A. In order to define and describe the internal and external behaviour of the robot firmware, a *User Requirements Definition Phase for IMR71848 Intelligent Mobile Robot Software* (URD-IMR71848) document was prepared. The user requirement document follows the guidelines and format produced by the European Space Agency which relevant to IEEE/ANSI 830-1984 Standard. This requirement document is part of the output from the hybrid methodology.

In Chapter IV, the models and techniques used in the hybrid analysis were used to produce a detailed specification document for the IMR71848 robot firmware requirement analysis. This document, called *Software Requirements Specification for IMR71848 Intelligent Mobile Robot Software* (SRS-IMR71848). The specification document provides the analysis results of software requirements for the IMR71848 robot firmware. The software requirement document follows the guidelines and format produced by the European Space Agency which relevant to IEEE/ANSI 830-1984 Standard.

Finally in Chapter V will discuss the evaluation results of the hybrid software engineering methodology. There are two levels of evaluation, at analysis and design phase and at the implementation phase.

A HYBRID SOFTWARE ENGINEERING METHODOLOGY FOR SMALL-SCALE EMBEDDED FIRMWARE DEVELOPMENT

Dayang Norhayati Abang Jawawi,^a Radziah Mohamad^a, Safuai Deris^a, Rosbi Mamat^b

*^aFaculty of Computer Science and Information System, Universiti Teknologi Malaysia,
81310 Johor Bahru, Malaysia*

*^bDepartment of Mechatronics and Robotics Engineering, Faculty of Electrical Engineering, Universiti
Teknologi Malaysia, 81310 Johor Bahru, Malaysia.*

Abstract

Embedded real-time (ERT) systems require rather sophisticated software development, as the software is in a tight coupling with its physical environment and it must respond to real-time events under strict timing constraints. Special tools and appropriate software engineering methodologies are therefore highly desirable for the development of ERT software, in order to produce software that is not only satisfied the functional and performance requirements, but also reliable, easy to maintain, completed within the specified time frame and at a reasonable cost. A software engineering methodology provides notations, methods and tools to assist a software developer. Adoption of a single complete methodology for small-scale ERT firmware development is not beneficial to the designers due to the some deficiencies such as inadequately for presenting timing constraints, methodology complexity and the requirement of special support tool to implement a methodology. Faced with the problem of adopting a single methodology for developing small-scale ERT systems, a methodology called hybrid method is developed. A wall-climbing robot (WCR) system under development at Universiti Teknologi Malaysia (UTM) is an example of small-scale ERT system. This paper discussed the use of a hybrid software engineering methodology in developing firmware for the WCR control firmware.

Keywords: software engineering methodology, embedded systems and real-time system.

Introduction

Embedded real-time (ERT) system is a system, which contains microcomputer as a component to do the processing and control of its environment, but the user does not see the system as a computer. Software development for ERT systems is very much different from the traditional data processing systems due to non-functional requirements such as dependability and the presence of hard timing constraints. Therefore, special software tools and appropriate software engineering methodologies are therefore highly desirable for the development of ERT software.

The purpose of software engineering methodology is to promote a certain approach to solve software problems. A number of software engineering methodologies already exist, which targeted toward real-time systems such as Modular Approach to Software Construction (MASCOT), Design Approach for Real-Time Systems (DARTS),

Structured Analysis and Design for Real-Time Systems (SDRTS), Unified Modeling Language for Real-Time (UML for Real-time) and Hard Real-Time Hierarchical Object Oriented Design (HRT-HOOD). Not all of these software engineering methodologies will support the software development for small-scale embedded system.

Adoption of a single complete methodology for small-scale ERT firmware development is not beneficial to the designers due to the some deficiencies such as inadequately for presenting hard timing constraints, lacking in capability to represent control oriented systems, methodology complexity and the requirement of special support tool to implement a methodology. Faced with the problem of adopting a single methodology for developing small-scale ERT systems, a methodology called hybrid methodology is developed. In the developed hybrid methodology, several suitable notations and diagrams taken from Ward-Mellor Structured Development for Real-Time Systems [1], Unified Modeling Language for Real-Time [2] and Hard Real-Time Hierarchical Object Oriented Design [3] methodologies were used to specify and design a small-scale ERT firmware.

A wall-climbing robot (WCR) system under development at Universiti Teknologi Malaysia (UTM) is an example of small-scale ERT system. This paper discussed the use of a hybrid software engineering methodology in developing firmware for the WCR control firmware. This paper is organised as follows. Next Section presents the specifications of the WCR controller hardware and firmware. Some issues, which will influence the firmware development process, will be highlighted in the same Section. The analysis and the design modeling of the WCR firmware using the hybrid ERT software methodology will be presented in the following Section in detail. Finally, the paper will be concluded in Section 5.

Specification of Wall-Climbing Robot Systems

The UTM WCR consists of a body and four similar associated electronics, and others load. Each leg has three joints, which will give a three-degree of freedom movement for each leg. Each joint is move by a direct-current (DC) motor and position sensors measure the angles of movement for each joint. At the tips of each leg there is a suction pad, which will stick the robot on the wall. Pressure sensors monitor the pressure inside the suction pads. Proximity sensors and collision sensors detect obstacles around the robot. The robot can be moved forward and backward by making a sequence of predefined steps.

The block diagram of the embedded controller for the WCR is shown in Fig. 1. The embedded controller is based on Intel 801C88XL microcontroller with 64K EPROM and 64K RAM. The main function of the control firmware is to move the four legs of the robot with a predefined sequence during climbing operation and monitor its environment and react to it intelligently.

The environment of WCR must be monitored, typically, every half a second to detect the presence of obstacles using the collision and the proximity sensors during the forward

and reverse movement of the robot. At each sampling period, the control signals to the DC motors are calculated using the proportional-derivative (PD) control algorithm. The current position of each leg joint is sensed using the position sensors and fed back to the embedded controller. The computation of the control signal typically, must be completed within 100 milliseconds to ensure the correct movement of the legs. The embedded controller also communicates with a remote PC to receive commands and sending back information via a serial communication link.

The main functional operation of the control firmware can roughly be divided into four major tasks: high-level navigation, environment monitoring, motor control and serial communication with remote PC. To satisfy the multi-tasking requirements for these major tasks, a real-time kernel is used in the control firmware.

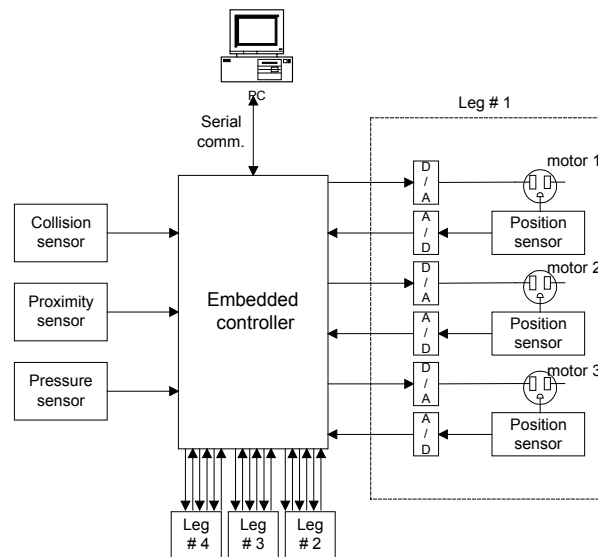


Fig. 1: Block diagram of the WCR controller.

To ensure the quality of WCR control software, the WCR development process should address the issues targeted toward the WCR system. Some issues, which will influence the software tools and software development process of the WCR firmware are; small-scale system limitations, concurrency and multitasking, hard real-time requirement, evolving nature of the WCR requirement and target hardware system.

The Hybrid Methodology

A hybrid methodology is proposed for the WCR control firmware development. In the proposed hybrid methodology, several suitable notations and diagrams taken from Ward-Mellor Structured Development for Real-Time Systems (Ward-Mellor), Unified Modeling Language for Real-Time (UML-RT) and Hard Real-Time Hierarchical Object Oriented Design (HRT-HOOD) methodologies were used to specify and design the robot control firmware. The models and techniques used in the hybrid analysis and design method are summarised in Table 1. In the specification stage the notations and diagrams

from Ward-Mellor and HRT-HOOD were used. In the design stage notations and diagrams from UML-RT and HRT-HOOD were used.

<i>No.</i>	<i>Modeling stage</i>	<i>Techniques and tools used</i>	<i>Methodology used</i>
1.	Environment model	Outside-in structuring	Ward-Mellor
		<i>Tools</i> Context diagram Event list table Timing estimation table	Ward-Mellor Ward-Mellor HRT-HOOD
2.	Behavioural model		
2.1.	First level decomposition	Structured-object model	HRT-HOOD
		<i>Tools</i> Data flow diagram Functional group table	Ward-Mellor HRT-HOOD
2.2.	Detail functions decomposition	<i>Tools</i> Data flow diagram	Ward-Mellor
2.3.	Non-functional decomposition	<i>Tools</i> Event-response table State transition diagram Sequence diagram	Ward-Mellor Ward-Mellor UML-RT
3.	Design		
3.1.	Tasks decomposition and task behavioural	<i>Tools</i> Statechart diagram	UML-RT
3.2.	Task communication and synchronisation	<i>Tools</i> Task diagram	-
3.3.	Timing Performance	<i>Tools</i> Priority table	HRT-HOOD

Table 1 : The models and techniques used in the WCR firmware specification analysis and design.

The idea of combining several notation and tools from different software engineering methodologies for developing real-time system is not new. The main advantage of adopting the hybrid method is that any appropriate notation and diagram can be chosen from different methodologies for functional and non-functional specification and design. The main disadvantage is that no CASE tool support is available to assist the use of the hybrid method. However, due to the scale of WCR project this can be handled manually.

The nature of the evolving WCR project make the iterative and incremental process model is more suitable to be used. Therefore in the development of the robot control firmware the incremental process model is adopted. Incremental process model combine elements of linear sequential model with the iterative prototyping, which enable software engineer to develop increasingly more complete version of the software [4].

WCR Requirement Analysis

The first stage in software development work is to capture the requirements of the WCR application. The purpose of this requirement capture is to obtain a description of WCR control system, which implies a transfer of information and expertise of UTM four-legged WCR from UTM Mobile Robot Research Group (MRRG). Based on the captured

requirement, the analysis of the WCR control firmware will be modeled using environment model and behavioural model.

The environment model is used to build a clear model of the robot environment in this specification analysis phase. The notations used in the environment model consist of:

- Context diagram defines the external objects or external devices in the robot system environment, shown in Fig. 2.
- Event lists table aims to list all the possible objects and functions in the system for further analysis and in the design stage.
- Object timing estimation table, which the timing specification for each object in the environment was estimated, shown in Table 2.

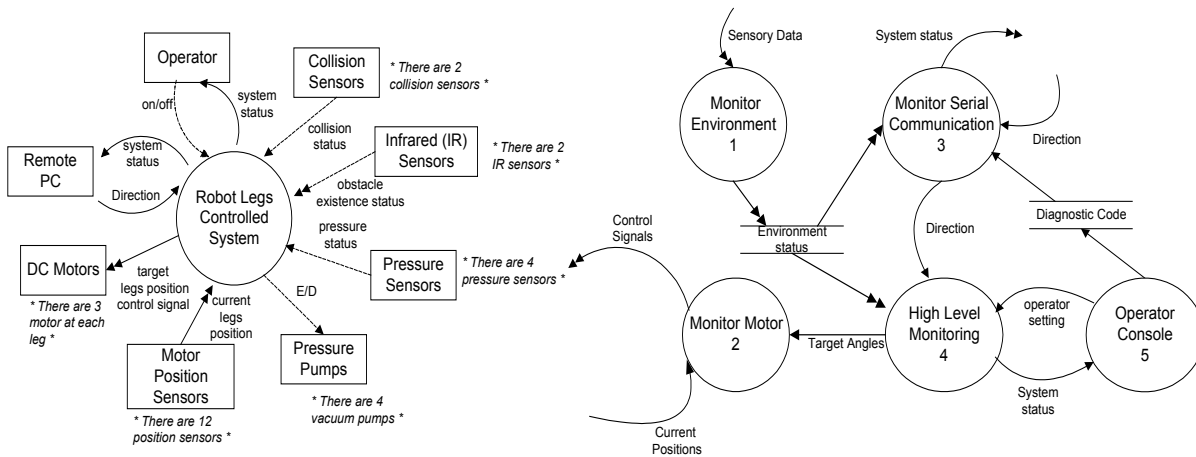


Fig. 2: WCR control system context diagram.

Fig.3: First levels DFD for the WCR control system.

Object	Min.-max. times
Collision Sensors – reading sensors	0.5 sec – 20 sec
Infrared (IR) Sensors – reading sensors	2 sec – 30 sec
Pressure Sensors – reading sensors	10 sec – 30 sec
Pressure Pumps – activate or deactivate suction pads	20 sec – 1 min
Motor Position Sensors – read sensors	5 millisecc - 20 millisecc
DC Motors – increase or decrease joint angles	50 millisecc – 100 millisecc
Remote PC – received or send data	1 sec – 10 sec
Operator – display system status	10-30sec

Table 2 : Minimum and maximum time for each object.

From the analysis of the environment model, the WCR control firmware can be divided into five main functional operations: *Monitor Environment*, *Monitor Motor*, *Monitor Serial Communication*, *High-level Monitoring* and *Operator Console*. The behaviour of the WCR transformation between the functions was presented using the Ward-Mellor first level data flow diagram (DFD) shown in Fig. 3. For the WCR analysis, data flow diagram (DFD) was used to make the analysis easier to understand. The software engineer and the system engineer can easily produce and understand this diagram and the correctness of this diagram is very important for the specification of the timing constraint in the next stage. The first level of the system behavior is then detailed by the Ward-Mellor’s second level decomposition of each module.

State transition diagram (STD) is used together with sequence diagram to model the behaviour of the robot system such as the control flow in the robot system. Example of the STD and the sequence diagram for event “Operator switch ON” with no error condition from the diagnostic test are shown in Fig. 4 and Fig. 5 respectively.

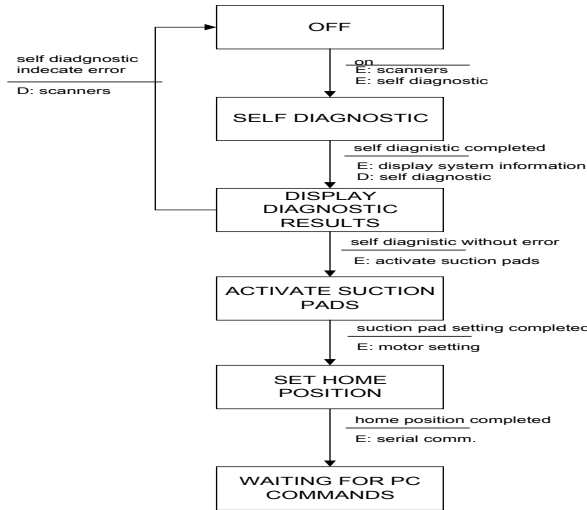


Fig. 4: Operator switch “on” STD.

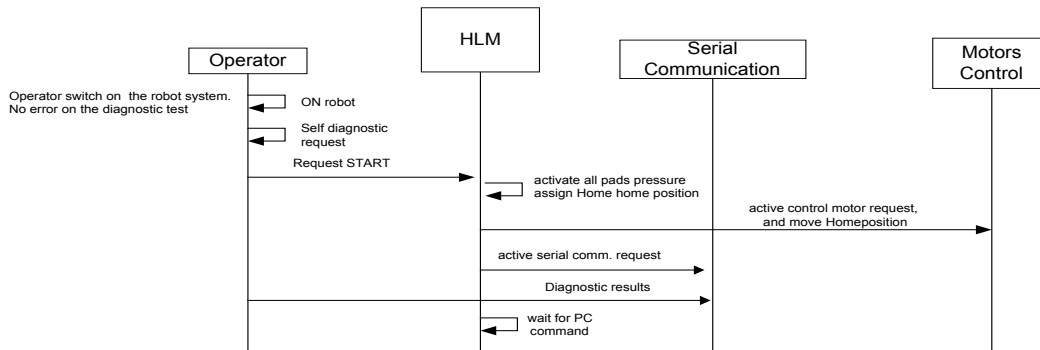


Fig. 5: Operator switch “on” sequence diagram.

The WCR Control Firmware Design

Based on the robot software requirement analysis, at this stage the robot behaviour and the robot constraints need to be presented in more detail, so that the design can be translated directly to coding pattern. The robot firmware was designed according to three design issues; tasks decomposition and task behavioural, task communication and synchronization and timing performance.

From the WCR environment and behavioural model, it was clear that concurrent module or processes presence in the operation of the robot. The five concurrent processes are *operator console, environment monitoring, PC communication, motor control and high level control*. From these concurrent modules, further tasks decomposition need to be performed in order to divide the processes into smaller and manageable tasks and

functions, consequently transferred to tasks code to be scheduled by a real-time kernel. Further task decomposition is performed, by detailing the WCR behavioural model using statechart. The statecharts for the WCR control firmware *environment monitoring* module are shown in Fig. 6.

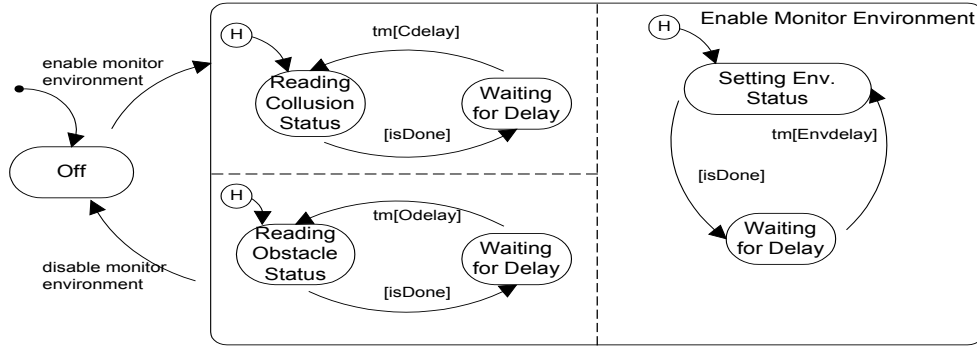


Fig. 6: Statechart for environment monitoring module.

Task diagram is used to show the tasks synchronisation, shared data or interfaces between tasks. From the task diagram Fig. 7, the design was divided based on the five main function groups derived from requirement analysis stage. In each function module the inner state are called *substate*. The basic state types for the climbing robot are cyclic (C), function (F) and protected (Pr). All states with function and protected type will be called by other state, the calling process is shown by arrow between the state. Only cyclic tasks will not be called, this is because after the creation of the cyclic task, they will cycle forever until the firmware is terminated.

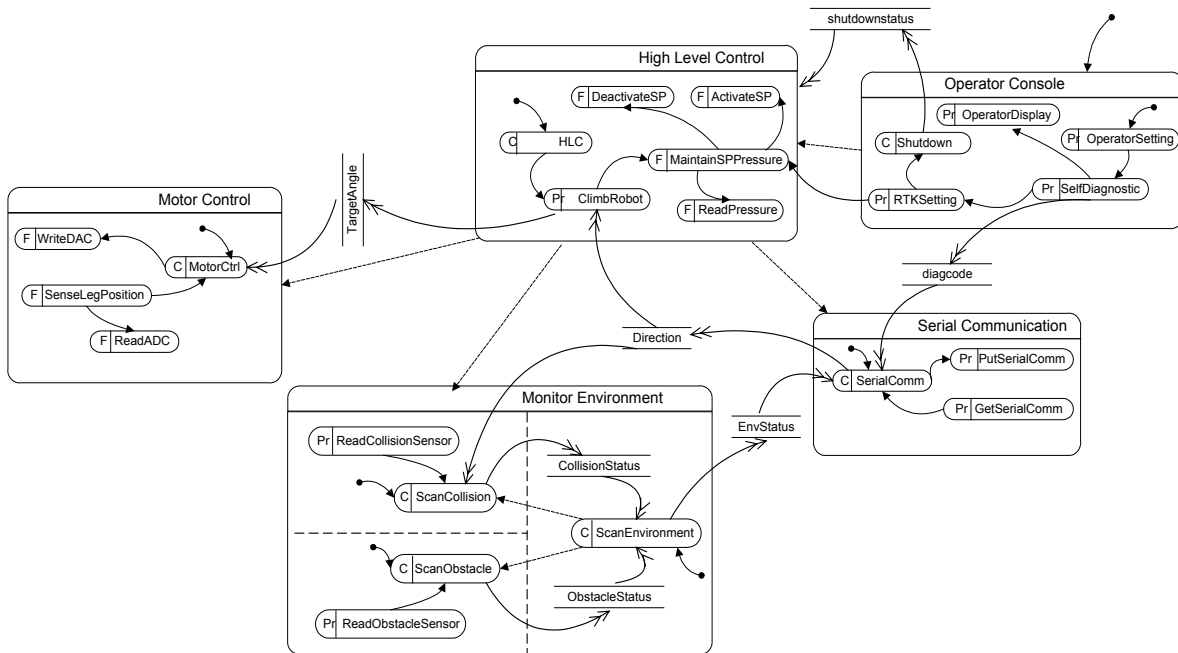


Fig. 7: Wall-climbing robot tasks diagram.

Each task is created in different module, dotted arrow shows the module create the task in other module, for example module *High Level Control* created cyclic tasks called *MotorCtrl* task, *ScanEnvironment* task and *SerialComm* task. Shared data between tasks and modules are shown using data store box and the flow of the data transformation is shown using double arrow. For example, *TargetAngle* data is generated by *ClimbRobot* function and read by *MotorCtrl* task.

Seven concurrent tasks were identified from *task diagram* in Fig. 7. In order to schedule the execution of the seven tasks using a preemptive real-time kernel, the priority of each task needs to be assigned. The timing constraints information derived from the WCR control firmware specification was used in the assigning tasks priority process. The timing of the seven tasks was analysed using *Rate Monotonic Scheduling (RMS)* technique to initialise priorities for each task. Basically, in RMS technique the tasks with the highest rate of execution are given the highest priority. Based on this, the priority for each task is assigned the priority.

CONCLUSION

A hybrid software engineering methodology was proposed for the WCR control firmware development. The WCR firmware hybrid analysis and design flow are presented in detail in this paper. Software engineering methods, models and techniques used in the WCR control firmware specification analysis and design were also discussed. Based on the WCR firmware design, the implementation stage is to gradually develop the firmware part by part by building more and more functionality and non-functionality of the WCR control firmware using software tools: Borland C/C++ 3.1 compiler, ROM locator for generating ROMable code and μ C/OS-II real-time kernel. A hardware-in-the-loop simulation method is used for testing the implemented firmware. From the test results it was found that by following the proposed hybrid method for developing the control firmware, a firmware that conformed to the requirements set could be developed successfully. The quality of the WCR control firmware reflects the effectiveness of the hybrid software engineering methodology and the software tools used in the methodology.

References

1. Ward, P. T. And Mellor, S. J. (1985). "Structured Development For Real-Time Systems", Volume 1-3, New York: Yourdon Press.
2. Douglass B. P. (1998). Real-Time UML Developing Efficient Object For Embedded Systems, USA: Addison Wesley.
3. Burns A., Wellings A. J. (1995). HRT-HOOD: A Structured Design Method For Hard Real-Time System, Volume 3, Elsevier.
4. Pressman, Roger S. (1997). "Software Engineering A Practitioner 'S Approach". Forth Edision. New York, U.S.A.: Mcgraw-Hill.

User Requirements Definition Phase for IMR71848 Intelligent Mobile Robot Software

Document ID : URD-IMR71848
First Issue – august 2003

Prepared: Dyg. Norhayati Abg. **Date:** 19 August 2003 **Signature**
Jawawi, FSKSM, UTM

Approved: Mohd Ridzuan Bin Ahmad **Date:** **Signature**

This document contains preliminary information. It subjects to revision and therefore does not represent a final report. The information in this document represents the view of the UTM Vot 71848 project research group.

TABLE OF CONTENTS

1 INTRODUCTION	3
1.1 PURPOSE.....	3
1.2 SCOPE.....	3
1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS	4
1.4 REFERENCES	4
1.5 OVERVIEW	5
2 GENERAL DESCRIPTION	6
2.1 PRODUCT PERSPECTIVE.....	6
2.2 GENERAL CAPABILITIES.....	6
2.3 GENERAL CONSTRAINTS	8
2.4 USER CHARACTERISTICS	9
2.5 OPERATIONAL ENVIRONMENT.....	9
2.6 ASSUMPTIONS AND DEPENDENCIES.....	11
3 SPECIFIC REQUIREMENTS	12
3.1 CRUISE	13
3.2 COMMUNICATION WITH EXTERNAL PC	14
3.3 SENSOR MONITORING	15
3.4 HIGH-LEVEL CONTROL.....	16
3.5 CAPABILITY REQUIREMENTS.....	17
3.6 CONSTRAINTS REQUIREMENTS.....	18
3.6.1 <i>Communication interfaces</i>	18
3.6.2 <i>Hardware Interfaces</i>	18
3.6.3 <i>Human-computer Interfaces (user interfaces)</i>	18
APPENDIX A: LIST OF USER REQUIREMENTS	19
APPENDIX B: LIST OF USER REQUIREMENTS TO BE CONFIRMED.....	20

User Requirements Definition Phase for Intelligent Mobile Robot Software

Abstract: This report describes a set of user requirements that apply to embedded software of a mobile Robot. This user requirement is prepared for the research Vot 71848 (Title - Experimental Evaluation of Hybrid Software Engineering Methodology for Embedded Firmware Development on Intelligence Mobile Robot) Research Group.

1 INTRODUCTION

1.1 Purpose

This document provides a definition of user requirements for the embedded software of an intelligent mobile robot. This software user requirement report is being prepared as part of a working group in UTM Vot 71848 Research Group (UTM71848RG).

Since these requirements are also connected to other user system requirements such as the robot hardware and mechanical system, some of the requirements may need to be adjusted, after review of the other work components. As such, it is an evolving document and is expected to undergo further review and refinement.

1.2 Scope

The software produced from this requirement will be called Intelligent Mobile Robot for 71848 research or called IMR71848 software. The IMR71848 software will be produce under the name of UTM71848RG. The software will be used to evaluate a hybrid software engineering methodology in intelligent mobile robot domain.

At this phase of the robot software development, there is no specific application targeted for the robot. The goal of this software is to control the movement the intelligent mobile robot in finding a passage and exiting through the passage. This

software requirement is mainly based on part of software robot requirement of a mobile robot, which is developed at UTM by [1] [2].

1.3 Definitions, Acronyms and Abbreviations

Priority - a mechanism used to order the request to run any task.

UTM – Universiti Teknologi Malaysia

UTM71848RG – UTM Vot 71848 Research Group

LCD - Liquid Crystal Display

LED – Light Emitting Diod

DC - Direct Current

IR - Infrared

I/O – Input and output

IMR - Intelligent Mobile Robot

ADC – Analog to Digital Converter

DAC – Digital to Analog Converter

UR – User Requirement

TBC – To Be Confirmed

1.4 References

[1] Mohd Ridzuan Bin Ahmad, “Development of Reactive-Decentralized Control Algorithm for Intelligent Multi-Agent Robotics System in Cooperative Task Achievement”, Master of Engineering thesis, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, August 2003.

[2] Dyg. Norhayati Abg. Jawawi, Ahmad Zariman Abd. Majid, Ahmad Ruzaimie Abd. Rashid, “Intelligent Mobile Robot Structure and Behaviour”, Technical Report VOT71848, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia.

[3] Nenad M. Kircanski, Mobile Robotics Systems, University of Toronto, CRC Press LLC, 2002.

- [4] Brooks R. A. (1986). "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, Vol. RA-2, No.1
- [5] Thomas Braunl, (2003). "Embedded Robotics – Mobile Robot Design and Applications with Embedded Systems", Springer

1.5 Overview

Kircanski defined mobile robot and intelligent mobile robot as followed [3]:

"A mobile robot is an autonomous system capable of traversing a terrain with natural or artificial obstacles."

"By the term intelligent in intelligent mobile robot means that the navigation is "task-oriented" and that it is based on dynamically sensing and modeling the external world."

Based on the definition the UTM71848RG proposed a wheeled mobile robot capable of traversing in an environment, which is surrounded by four walls. The goal or task of the robot is to find a passage and exiting through the passage. Therefore, the goal of this Intelligent Mobile Robot (IMR) software is to control the movement the robot in finding a passage and exiting through the passage.

The IMR71848 software requirements have been partitioned into the following categories:

- 1) General description and background of the IMR software
- 2) Specific requirements include the software functional capability, performance and constraints. The specific software requirements in discussed in the following sub-title:
 - (a) Cruise
 - (b) Monitor Sensor
 - (c) High-level navigation
 - (d) Communication

Each of the above areas is discussed in the remaining sections of this report.

2 GENERAL DESCRIPTION

2.1 Product Perspective

The software product is a new software product for this particular UTM robot implementation i.e. the software product is not to replace an existing system. The software product is not standalone and it has to communicate with other part of software product such as intelligence robot software. The software also has to interact with other system components such as embedded controller, remote PC and the environment through sensors.

2.2 General Capabilities

The IMR71848 software must capable to control the movement the robot in finding a passage and exiting through the passage. During the cruising process, the IMR71848 software must support the intelligent components of the robot in order to ensure that the robot can response to the conditions in the environment in archiving the robot goal.

The mechanical construction of the robot is shown in Figure 1 and the side view of the robot is shown in Figure 2. The IMR consists of a body and two pair of wheels. The robot's body is a three layers platform including a ground layer. The body carries the embedded controller and its associated electronics, and others load. The IMR71848 robot is a differential drive design robot. The cruising of the robot is supported a pair of drive wheels and a pair of castor wheels. Each drive wheels is move by a direct-current (DC) and the castor wheels are placed at the robot for stabilization of the IMR. The software is responsible to control the DC motors, in order control the movement of the IMR71848.

The software will be embedded into the AMD188ES embedded controller component of the robot. The embedded controller for the IMR can be represented in block diagram form as shown in Figure 2. The main function of the embedded digital controller is to move the wheels of the robot until the robot archive the goal. In order

to move the robot and archive the robot's goal, the software need to control the motor at each robot drive wheels, monitor the environments and navigate the robot. The embedded controller monitors its environment using four infrared (IR) proximity sensors, an infrared distance sensor and a digital camera. The embedded controller receives configuration commands and sends back information to a remote PC using wireless communication via Radio Frequency (RF) transceiver.

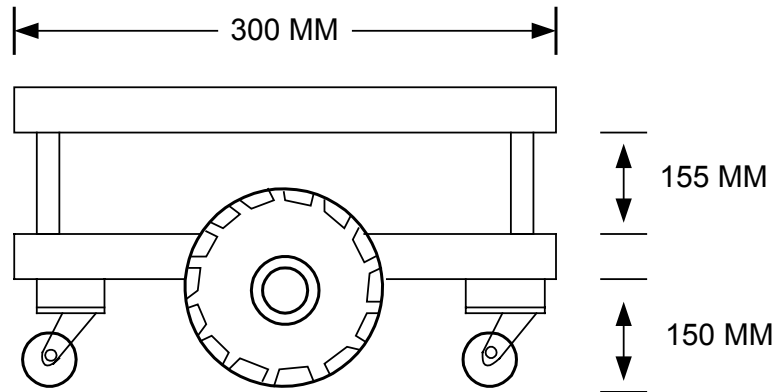


Figure 1: The Mechanical Construction Of The IMR.

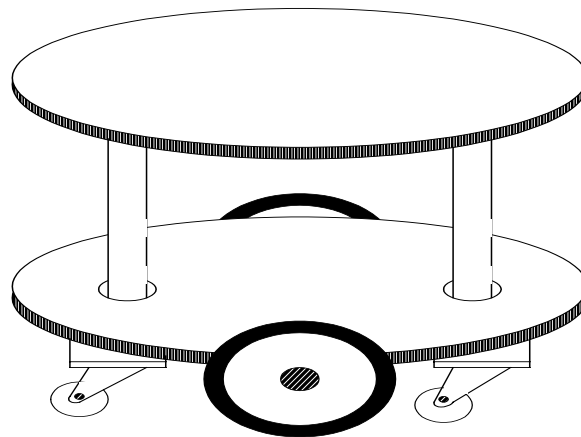


Figure 2: The side view of the IMR Robot.

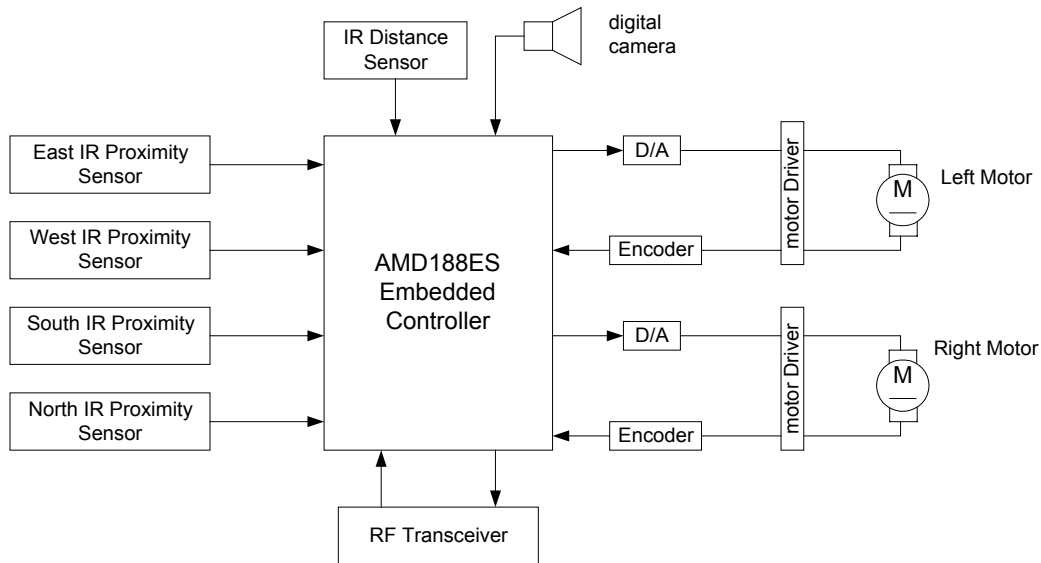


Figure 3 : Block diagram of IMR controller.

2.3 General Constraints

The embedded software will control the functionality of the hardware (e.g. sensors and motors). The software development will not consider the detail specification of the hardware, only the specification that related to interfacing of the software and the hardware will be considered.

Another limitation that will be considered in the software development process is the hardware capability and size of the system such as memory and processor capabilities. These limitations will influence the selection of tools and methods used in the software development. It also limits the code size of the software produce.

To enables the use of good quality and low cost PC based C compilers such as from Borland and Microsoft in the software development, the developments process will be done in PC and the real external world input and output will be simulate.

The IMR71848 software project in under research project vot 71848, therefore the software project development period and budget is limited under the grant funding. The software development process is limited until 15 September 2003. The financial

budget of the software development limited to RM13000. This budget includes software development tools and robot hardware.

2.4 User Characteristics

The operator the system are technician and engineer, the involvement of the operator is listed in Table 1.

	<i>Technician</i>	<i>Engineer</i>
Involvement phase	Operation, testing, maintenance	User requirements, operation, testing, maintenance
Education level	Minimum SPM	Minimum B.Sc.
Experience with such system	1-5 years	1-6 years
Language	Malay	Malay, English

Table 1: Operator the system

2.5 Operational Environment

The robot software is highly coupled to the external world. The relationship between the control software and the external devices is shown in Figure 2. Human operator starts or stops the robot through switch on/off button. Once the robot system is switched on the following operations with external devices need to be performed by the IMR71848 software:

1. **Cruise.** The robot movement needs to be controlled by sending commands to motor driver unit based on the environment monitoring.
2. **Sensors monitoring.** The environment must be monitored to detect the presence of obstacles using proximity sensors, distance sensor and digital camera.
3. **High-level control.** To navigate the robot intelligently toward the robot goal location and acknowledge the completion of the task.
4. **Communication.** The robot receives information from remote PC and sends robot information back to remote PC.

The experiment environment of the mobile robot is indoor environment. The robot's environment is surrounded with walls except for the doorway passage. Figure 3 shows the experiment environment of the IMR71848. The IMR71848 cruise around the

environment to find the passage. The robot needs to avoid obstacle exist within its environment.

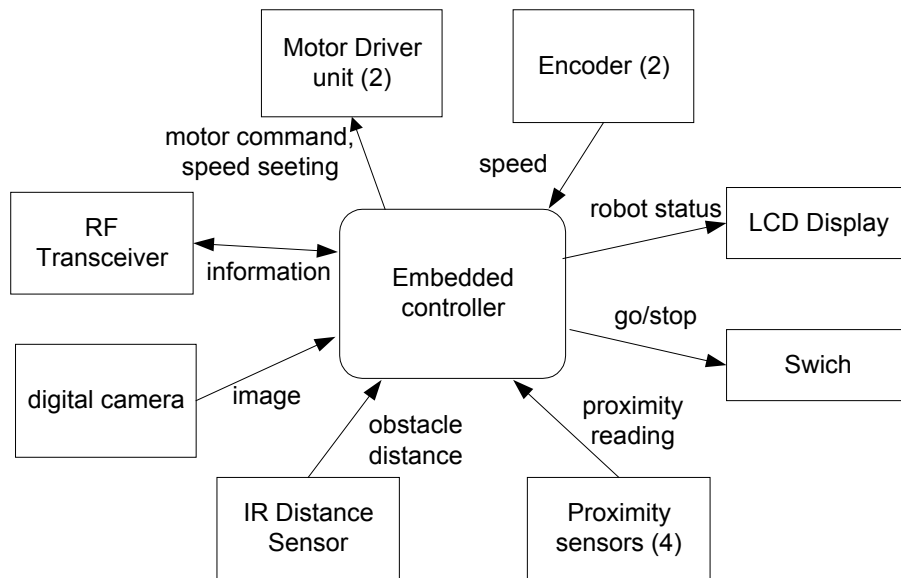


Figure 2 : Graph showing external devices.

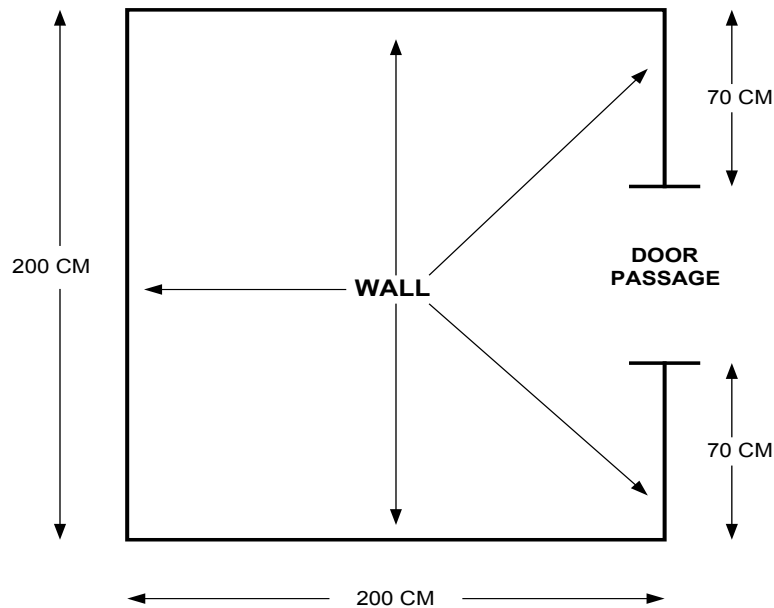


Figure 3 : Experiment environment of IMR71848.

2.6 Assumptions and Dependencies

It is assumed for these requirements that the requirements will be validated for the real controller system i.e. this requirement is applicable for software on embedded controller not only on PC. The end software product of the robot has to be tested at the embedded controller level. If the controller or hardware is not ready in the software-testing phase, a mechanism such as simulation of the hardware must be provided in this software development process. The requirements of the simulation software will not be considered in this requirement.

The success of meeting milestone dates set up at each development phase for this software depends on the following events:

- Non-availability of RTOS when the development has to be started
- Non-availability of hardware or robot controller when the implementation and testing of the embedded software need to be started.
- Lack of skill of the development team to new experience and new tools – RTOS, Real-time methodologies selected, C programming for microcontroller systems and hardware programming
- Major changes in the robot requirements during the process of software development

3 SPECIFIC REQUIREMENTS

The IMR7848 is a wheeled robot that capable to move around its environment in archiving the robot following goals:

- Finding a passage
- Exiting through the passage
- Avoid obstacles
- Maintain the robot at certain speed

The IMR7848 software is to move the wheels of the robot until the robot archive the robot's goals (**UR1**). In order to move the robot and archive the robot's goal, the software need to control the DC motors at each robot drive wheels, monitor the environments and navigate the robot.

The program starts to operate when the user switches on the robot and end when the user switches off the robot (**UR2**). Once the robot system is switched, the main operation of the robot software is divided into four main groups (**UR3**) as follows:

- i. Cruise. This software component enables the IMR to move around its environment. This module needs to send command to motor driver based on the movement commands received from high-level navigation module.
- ii. Monitor sensors. Based from the reading from the sensors, this component software presented the environment status for high-level navigation module.
- iii. High-level Control. In archiving the robot's goal, this module supports the intelligent component of the robot. Based on the environment status received from monitor sensors module, this module will decide the behaviour of the robot. Based on this behaviour the movement of the robot will be identified.
- iv. Communication. The robot configuration send to the robot through wireless communication, based on this configuration the robot will be configure before the robot journey started. The robot movement and sensor data will be monitored using the remote PC.

The summary of the three functional operation of the robot controller is presented in Figure 4.

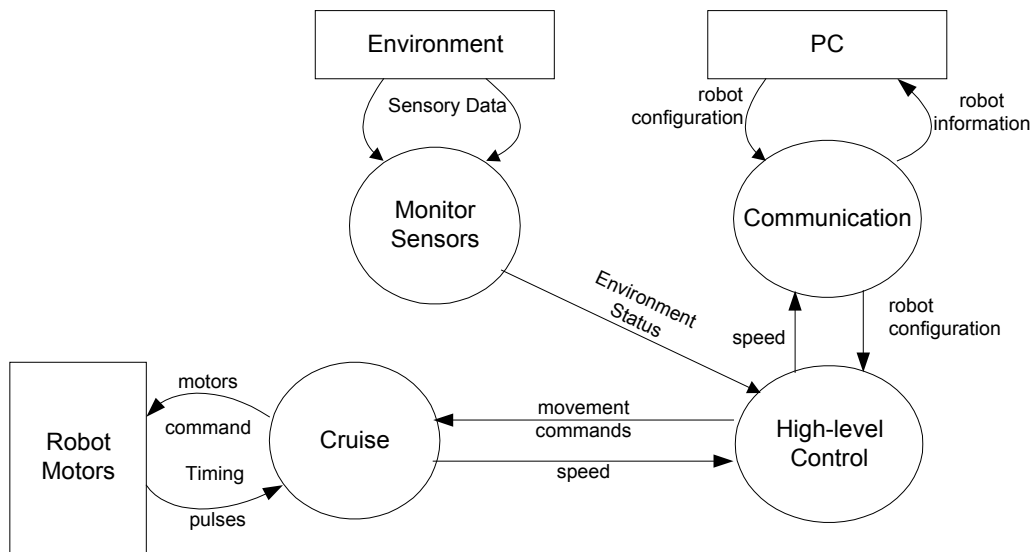


Figure 4: Interaction between the robot functional operation.

3.1 Cruise

Cruise software component control the IMR71848 movement around the robot environment (**UR4**). Based on movement commands that are received from High-level control module, the IMR71848 motor will be control in order to control the IMR71848 movement (**UR5**). Movement commands elements are (**UR6**):

1. **Status:** stop or go
2. **Direction:** straight or turning

The direction movement of the robot can be classified into two (**UR7**):

- Straight – two types of direction forward or backward.
- Turning – either left or right and each turning is 45 degree.

During straight line cruising, IMR7848 software needs to maintain the robot at constant speed level of **TBC01**. To make a right turn the speed of the left motor (**TBC02**) is faster than the right motor (**TBC02**). To make a left turn the speed of the right motor (**TBC03**) is faster than the left motor (**TBC03**).

This cruise module received tuning pulses from encoder in the robot motor. This module is responsible to change the tuning pulses reading to speed value of the motor **(UR8)**.

Based on the received movement command from high-level navigation module and the current speed of the motor, at each sampling period the control signals to the DC motors are calculated using the proportional-integral (PI) control algorithm **(UR9)**:

$$u(t) = K_p e(t) + K_i \int e(t) dt$$

Where,

$u(t)$ is the control signal at time t

$e(t)$ is the error between the target speed and the current speed at time t

K_p and K_i are constants for each motor.

The computation of the control signal must be completed within 100 milliseconds to ensure the constant speed of the robot **(UR10)**.

3.2 Communication with External PC

To monitor the movement of the IMR17848 robot, it is communicated to a remote PC. The IMR17848 communicates with a remote PC to receive configuration commands and sending back information via a Radio Frequency (RF) transceiver **(UR11)**. Configuration commands are the setting up of the robot before the robot starts its movement **(UR12)**. Information that sends to the remote PC is the robot speed and direction **(UR13)**.

In IMR17848 software design, the designer doesn't really care what software on the remote PC does. The designer just concern with the controller on the robot and how it interface. The remote PC software will be specified in other document.

3.3 Sensor monitoring

The embedded controller monitors its environment using some sensors. The sensors provide information for the IMR7848 software to enable the software to eliminate interferences due to the collision with obstacles and walls (**UR14**). Besides, the sensors help the robot to plan its movement (**UR15**). The robot environment is monitored using:

1. Four IR proximity sensors to detect the presence of obstacles during the forward and reverse movement of the robot (**UR16**).
2. An IR distance sensor to estimate the distance of obstacles and walls from the robot current location (**UR17**).
3. A digital camera to detect shape of objects in the robot's environment (**UR18**).

The implementation of IMR7848 software is divided into three main incremental phases based on the sensors and communication introduced to the robot. The phases suggested are as shown in Table 2.

Phase	Sensor added	Robot Capability
Basic	IR proximity	Find wall and follow the walls to detect passage.
Medium	IR distance, RF transceiver	Differentiate wall and other static obstacle.
Advanced	Digital camera	TBC

Table 2: The IMR71848 incremental phases.

The requirement specified in this document will cover basic and medium phase in detail, the implementation of the advanced phase will be specified in detail later (**TBC04**).

The sensors position and direction is shown in Figure 5. Four IR proximity sensors use to detect the existence of obstacle at four direction of the robot: front, back, left and right (**UR19**). An IR distance sensor will rotate 180 degree with 15 degree for each distance reading; therefore, with 180 degree rotation can record 7 different directions of distance reading (**UR20**).

The IMR7848 software needs to periodically get information from the environment through sensors, and the maximum period or cycle for reading the each sensor is listed in Table 3 (UR21). The reading of the sensors is called environment status.

Object	Min.-max. times
IR Proximity Sensors – reading sensors	0.5 sec – 1 sec
IR Distance Sensor – reading sensors	1 sec – 2 sec

Table 3: Sensors Timing Estimation

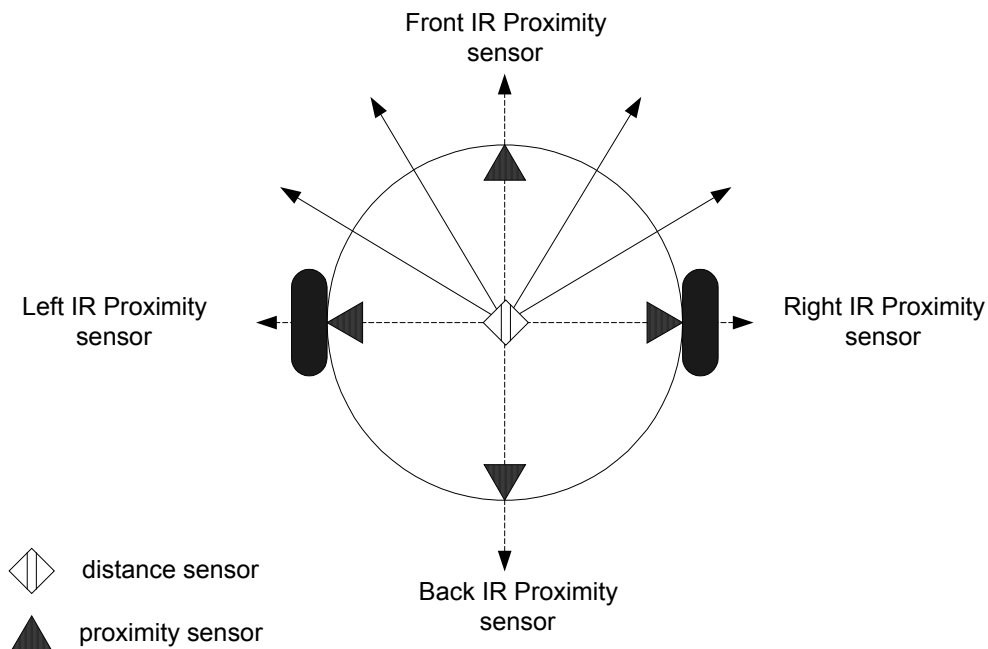


Figure 5: Sensors position and directions.

3.4 High-level Control

The robot can be moved forward, reverse, turn left and turn right. The robot moves at constant speed level during straight movement and this speed will be change during turning movement (UR22). The speed levels of the robot are depending on the distance of the robot from an obstacle (TBC05). The robot speed will be display on Liquid Crystal Display (LCD) display. Besides the robot's speed, the battery status will also be displayed on LCD (UR23).

During the cruising process, the IMR71848 software must support the intelligent components of the robot in order to ensure that the robot can response to the

conditions in the environment in archiving the robot goal (**UR24**). The intelligence of the IMR71848 robot is supported by subsumption architecture proposed by Professor Rodney Brooks from Mobile robot Group, MIT Artificial Intelligent Laboratory [4]. The implementation of this architecture toward IMR71848 is based on Mohd Ridzuan 2003 (**UR25**). Based on Mohd Ridzuan, IMR71848 robot basic behaviour are as followed (**UR26**):

1. Cruise – The ability of the robot to move around its environment.
2. Wall Detection – The ability of the robot to detect and follow the wall.
3. Obstacle avoidance – The ability of the robot to avoid collision with any obstacle.
4. Passage detection – The ability of the robot to reach the wall passage.
5. Task Completion – The ability of the robot to acknowledge the completion of the robot task once the robot manage to exit the passage.

In subsumption architecture all the above behaviours run in parallel, the higher-level behaviours have the power to suppress the lower-level behaviour. Each behaviour is assigned priority level as shown in Table 4 (**UR27**). The highest priority is given the lowest number.

Behaviour	Priority
Cruise	5
Wall Detection	4
Obstacle avoidance	3
Passage detection	2
Task Completion	1

Table 4: Behaviour priority level.

The navigation algorithm used by IMR71848 is wandering standpoint algorithm [4]. The algorithm try to reach from start in direct line, and when encounter an obstacle the robot will follow around the object until the goal direction is clear again (**UR28**).

3.5 Capability Requirements

Limitation that will be considered in the software development process is the hardware capability and size of the system such as memory and processor capabilities.

The hardware capability is 128K EPROM and 128 RAM. The size and the memory usage of the IMR7848 software must be with this capability.

The type and the timing specification of the information transfer to remote PC depending on the capability of the RF transceiver.

3.6 Constraints Requirements

3.6.1 Communication interfaces

The format of the data packet received and sends back to the external PC is yet to be defined (**TBC06**).

3.6.2 Hardware Interfaces

The software has to run on embedded processor (**UR29**). The main hardware requirements of the system are as follows (**UR30**):

- Controller - based on Intel AMD188ES microcontroller
- EPROM - 128 K
- RAM - 128 K
- Parallel I/O
- Serial I/O
- ADC/DAC
- Switches on/stop
- LCD Display

3.6.3 Human-computer Interfaces (user interfaces)

At this stage it is assume all input and output to the system done through remote PC and LCD display, LED display and switches (**UR31**).

Local user interfaces detail are as followed (**UR32**):

- 2 switches to start and stop robot
- a LCD to display robot status and diagnostic results
- LED display diagnostic results

Appendix A: List of User Requirements

<i>UR No.</i>	<i>User Requirements Description</i>
1.	The IMR7848 software is to move the wheels of the robot until the robot archive the robot's goals.
2.	The program starts to operate when the user switches on the robot and end when the user switches off the robot
3.	Once the robot system is switched, the main operation of the robot software is divided into four main groups
4.	Cruise software component control the IMR71848 movement around the robot environment
5.	Based on movement commands that are received from High-level control module, the IMR71848 motor will be control in order to control the IMR71848 movement
6.	Movement commands elements are status and direction.
7.	The direction movement of the robot can be classified into two: straight and turning.
8.	This cruise module is responsible to change the tuning pulses reading to speed value of the motor.
9.	Based on the received movement command from high-level navigation module and the current speed of the motor, at each sampling period the control signals to the DC motors are calculated using the proportional-integral (PI) control algorithm.
10.	The computation of the control signal must be completed within 100 milliseconds to ensure the constant speed of the robot.
11.	To monitor the movement of the IMR17848 robot, it is communicated to a remote PC. The IMR17848 communicates with a remote PC to receive configuration commands and sending back information via a Radio Frequency (RF) transceiver.
12.	Configuration commands are the setting up of the robot before the robot starts it movement.
13.	Information that sends to the remote PC is the robot speed and direction.
14.	The sensors provide information for the IMR7848 software to enable the software to eliminate interferences due to the collision with obstacles and walls.
15.	The sensors help the robot to plan its movement.
16.	Four IR proximity sensors to detect the presence of obstacles during the forward and reverse movement of the robot.
17.	An IR distance sensor to estimate the distance of obstacles and walls from the robot current location.
18.	A digital camera to detect shape of objects in the robot's environment.
19.	Four IR proximity sensors use to detect the existence of obstacle at four direction of the robot: front, back, left and right.
20.	An IR distance sensor will rotate 180 degree with 15 degree for each distance reading; therefore, with 180 degree rotation can record 7 different directions of distance reading.
21.	The IMR7848 software needs to periodically get information from the environment through sensors, and the maximum period or cycle for reading the each sensor is listed in Table 3.
22.	The robot moves at constant speed level during straight movement and this speed will be change during turning movement.
23.	The robot speed will be display on Liquid Crystal Display (LCD) display. Besides the robot's speed, the battery status will also be displayed on LCD.
24.	During the cruising process, the IMR71848 software must support the intelligent components of the robot in order to ensure that the robot can response to the conditions in the environment in archiving the robot goal.

25.	The implementation of subsumption architecture toward IMR71848 is based on Mohd Ridzuan 2003
26.	IMR71848 robot basic behaviour are cruise, wall detection, obstacle avoidance, passage detection and task completion.
27.	Each behaviour is assigned priority level as shown in Table 4.
28.	The navigation algorithm used by IMR71848 is wandering standpoint algorithm. The algorithm try to reach from start in direct line, and when encounter an obstacle the robot will follow around the object until the goal direction is clear again
29.	The software has to run on embedded processor
30.	The main hardware requirement of the system : Intel AMD188ES microcontroller, 128K EPROM and 128 RAM, Parallel I/O, ADC/DAC, Switches on/stop and LCD Display.
31.	At this stage it is assume all input or output to the system done through remote PC and LCD display.
32.	Local user interfaceS detail are 2 switches to start and stop robot, a LCD to display robot status and diagnostic results and LED display diagnostic results.

Appendix B: List of User Requirements to be confirmed

<i>UR No.</i>	<i>User Requirements Description</i>
1.	IMR7848 software needs to maintain the robot at constant speed level
2.	To make a right turn the speed of the left motor (TBC02) is faster than the right motor (TBC02).
3.	To make a left turn the speed of the right motor (TBC03) is faster than the left motor (TBC03).
4.	The requirement specified in this document will cover basic and medium phase in detail, the implementation of the advanced phase will be specified in detail later.
5.	The speed levels of the robot are depending on the distance of the robot from an obstacle.
6.	The format of the data packet received and sends back to the external PC is yet to be defined.

Software Requirements Specifications for IMR71848 Intelligent Mobile Robot Software

Document ID: SRS- IMR71848
First Issue - December 2003

Prepared: Dyg. Norhayati Abg. **Date:** 30 December **Signature**
Jawawi, FSKSM, UTM 2003

This document contains preliminary information. It subjects to revision and therefore does not represent a final report. The information in this document represents the view of the project research group.

TABLE OF CONTENTS

1	INTRODUCTION	3
1.1	PURPOSE.....	3
1.2	SCOPE.....	3
1.3	DEFINITIONS, ACRONYMS AND ABBREVIATIONS	4
1.4	REFERENCES	4
1.5	OVERVIEW	5
2	GENERAL DESCRIPTION	6
2.1	RELATIONS TO CURRENT PROJECTS	6
2.2	RELATION TO PREDECESSOR AND SUCCESSOR PROJECTS.....	6
2.3	FUNCTION AND PURPOSE.....	6
2.4	ENVIRONMENTAL CONSIDERATIONS	6
2.4.1	<i>Robot Hardware Environments.....</i>	<i>6</i>
2.4.2	<i>Robot Operating Environments.....</i>	<i>7</i>
2.4.3	<i>Operating Environments in the Development Systems.....</i>	<i>7</i>
2.5	RELATION TO OTHER SYSTEMS.....	8
2.6	GENERAL CONSTRAINTS	9
2.7	MODEL DESCRIPTION	9
3	SPECIFIC REQUIREMENTS	11
3.1	ENVIRONMENT MODEL	11
3.2	BEHAVIOURAL MODEL.....	13
3.2.1	<i>First level Functional Behaviour Analysis</i>	<i>13</i>
3.2.2	<i>Detail Functional Behaviour Analysis.....</i>	<i>15</i>
3.2.3	<i>Non-functional Analysis</i>	<i>19</i>

Software Requirements Specification for IMR71848 Intelligent Mobile Robot Software

Abstract: This report describes a set of software requirements that apply to an intelligent mobile robot system. This report is prepared for the research Vot 71848 (Title - Experimental Evaluation of Hybrid Software Engineering Methodology for Embedded Firmware Development on Intelligence Mobile Robot) Research Group. This document is an evolving document and is expected to undergo further review and refinement.

1 INTRODUCTION

1.1 Purpose

This specification document provides a definition of software requirements for the embedded software of an intelligent mobile robot. It is based on requirements analysis of the user requirements (URS-IMR71848). This report will be used as the main reference for the detail design of the Intelligent Mobile Robot under Vot 71848 (IMR71848) software. Therefore the readers of this document are IMR71848 software designer, IMR71848 requirement analyser for next incremental stage or maintenance purposes and the IMR71848 verification team.

This software requirement report is being prepared as part of a working group in UTM VOT 71848 research member group (UTM71848RG).

1.2 Scope

The software will be produced under the name of UTM71848RG. The goal of this software project is to develop software for controlling the movement the intelligent mobile robot in finding a passage and exiting through the passage.

1.3 Definitions, Acronyms and Abbreviations

ADC – Analog to Digital Converter

DAC – Digital to Analog Converter

DC - Direct Current

I/O – Input and output

IR - infrared

LCD - Liquid Crystal Display

LED – Light Emitting Diode

PI - Proportional-Integral

RF - Radio Frequency transceiver

RTK – Real-time Kernel

RTOS – Real-time Operating Systems

SPM – Sijil Peperiksaan Malaysia

USR – User Requirement Specification

UTM – Universiti Teknologi Malaysia

UTM – Universiti Teknologi Malaysia

UTM71848RG – UTM Vot 71848 Research Group

WCR - Wall climbing Robot

1.4 References

- [1] Mohd Ridzuan Bin Ahmad, “Development of Reactive-Decentralized Control Algorithm for Intelligent Multi-Agent Robotics System in Cooperative Task Achievement”, Master of Engineering thesis, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, August 2003.
- [2] Dayang. Norhayati Abang. Jawawi, Ahmad Zariman Abd. Majid, Ahmad Ruzaimie Abd. Rashid, “Intelligent Mobile Robot Structure and Behaviour”, Technical Report VOT71848, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia.
- [3] Nenad M. Kircanski, Mobile Robotics Systems, University of Toronto, CRC Press LLC, 2002.

- [4] Brooks R. A. (1986). “A Robust Layered Control System for a Mobile Robot”, IEEE Journal of Robotics and Automation, Vol. RA-2, No.1
- [5] Thomas Braunl, (2003). “Embedded Robotics – Mobile Robot Design and Applications with Embedded Systems”, Springer
- [6] Dayang. Norhayati Abang. Jawawi, “User Requirements Definition Phase for IMR71848 Intelligent Mobile Robot Software”, Faculty of Computer Science and Information System, UTM, August 2003

1.5 Overview

The requirements have been partitioned into the following categories:

1. General description of the IMR71848 firmware project
2. IMR71848 formware specific requirement
 - 2.1.1. Environment Model
 - 2.1.2. Behavioural Model

Each of the above areas is discussed in the remaining sections of this report. Readers should understand the integration of various sections of this report.

2 GENERAL DESCRIPTION

2.1 Relations to Current Projects

This robot software project is being prepared as part of a working group in UTM71848RG. The parent project is funded under UTM grant, vot number 71848. The objective of the patent project is to evaluate experimentally a hybrid software engineering methodology for embedded firmware development on intelligent mobile robot. This software project develops a firmware of the UTM IMR71848 and used for the software methodology evaluation purposes. Previous work on developing firmware for UTM Wall Climbing Robot, has tested the software methodology under hardware-in-the-loop simulation and was found to be effective.

2.2 Relation to Predecessor and Successor Projects

This software prototype is mainly based on software methodology implemented on (Dayang, 2002) and the software intelligence of the robot is based on (Ridzuan, 2003).

2.3 Function and Purpose

The main function of the embedded digital controller is to move the wheels of the robot until the robot archive the goal. In order to move the robot and archive the robot's goal, the software need to control the motor at each robot drive wheels, monitor the environments and navigate the robot.

2.4 Environmental Considerations

2.4.1 Robot Hardware Environments

The software has to run on embedded processor. The main processor in the controller is based on AMD188ES microcontroller with 128K EPROM and 128K RAM, Parallel

I/O, Serial I/O, ADC/DAC, Switches on/stop and Status LED. The embedded controller monitors its environment using some sensors - four infrared proximity sensors, an infrared distance sensor and a digital camera. It also communicates with a remote PC via a Radio Frequency (RF) transceiver.

2.4.2 Robot Operating Environments

The experiment environment of the mobile robot is indoor environment. The robot's environment is surrounded with walls except for the doorway passage. Figure 1 shows the experiment environment of the IMR71848. The IMR71848 cruise around the environment to find the passage. The robot needs to avoid obstacle exist within its environment.

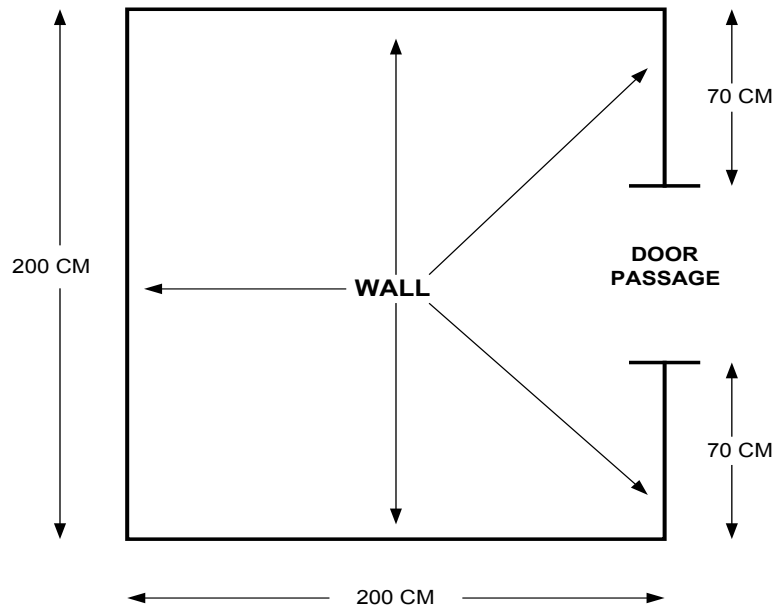


Figure 1: Experiment environment of IMR71848.

2.4.3 Operating Environments in the Development Systems

- The software development environment – Windows95 and DOS environments
- Software development tools – Borland C compiler version 3.1, Assembler
- MicroC/OS-II real-time operating system used for intertasks communication and memory management of the software.

- ROM locator for generating ROMable code

2.5 Relation to other Systems

The firmware is a subsystem of a complete IMR71848 system. Figure 2 shows the major component of the IMR71848 system and the interconnection of the IMR71848 firmware in the system.

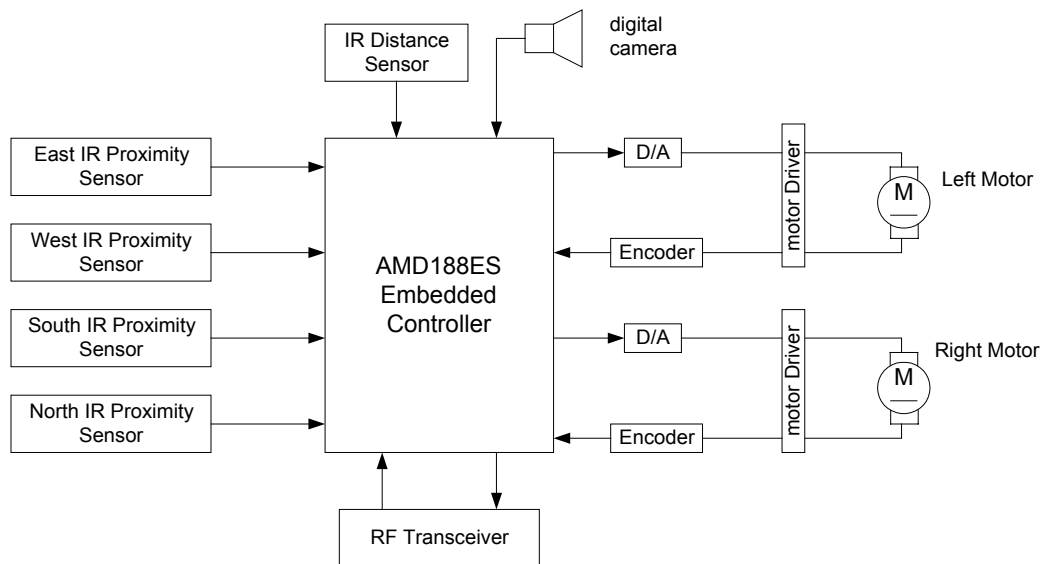


Figure 2: Block diagram of IMR controller.

Figure 1 shows that the robot controller system involves the following entities.

1. An operator – The operator has the ability to start and stop the robot. The operator is also provided with a display given information of the robot status and the robot movements.
2. IR proximity sensors – The sensors detect the presence of obstacles during the movement of the robot.
3. IR distance sensor – The sensor estimate the distance of obstacles and walls from the robot current location
4. Digital camera – The camera detect shape of objects in the robot's environment
5. Radio Frequency (RF) transceiver – To communicate to a remote PC.
6. DC motors – The robot is moved by actuating the DC motors.

7. Encoders – A feedback mechanism of the robot motor. The software will receive tuning pulses from encoder in the robot motor.

2.6 General Constraints

General constraints of the firmware development are:

1. Budget
2. Hardware
3. The used of the available tools and facilities in UTM only

2.7 Model Description

A hybrid methodology is proposed for the IMR71848 firmware analysis model. In the proposed hybrid methodology, several suitable notations and diagrams taken from Ward-Mellor Structured Development for Real-Time Systems (Ward-Mellor), Unified Modeling Language for Real-Time (UML-RT) and Hard Real-Time Hierarchical Object Oriented Design (HRT-HOOD) methodologies were used to specify and design the robot control firmware. The models and techniques used in the hybrid analysis are summarised in Table 1. In the specification stage the notations and diagrams from Ward-Mellor and HRT-HOOD were used. In the design stage notations and diagrams from UML-RT and HRT-HOOD were used.

Based on the captured requirement (Dayang, 2003), the analysis of the IMR71848 firmware will be modeled using environment model and behavioural model. The environment model is used to build a clear model of the robot environment in this specification analysis phase. The notations used in the environment model consist of:

- Context diagram defines the external objects or external devices in the robot system environment.
- Event lists table aims to list all the possible objects and functions in the system for further analysis and in the design stage.
- Object timing estimation table, which the timing specification for each object in the environment was estimated.

Table 1 : The models and techniques used in hybrid methodology.

<i>No.</i>	<i>Modeling stage</i>	<i>Techniques and tools used</i>	<i>Methodology used</i>
1.	Environment model	Outside-in structuring	Ward-Mellor
		<i>Tools</i> Context diagram Event list table Timing estimation table	Ward-Mellor Ward-Mellor HRT-HOOD
2.	Behavioural model		
2.1.	First level decomposition	Structured-object model	HRT-HOOD
		<i>Tools</i> Use-case diagram Data flow diagram Functional group table Package diagram	Ward-Mellor HRT-HOOD
2.2.	Detail functions decomposition	<i>Tools</i> Data flow diagram Sequence diagram	Ward-Mellor UML-RT
2.3.	Non-functional decomposition	<i>Tools</i> Event-response table Statechart diagram Sequence diagram	Ward-Mellor UML-RT UML-RT

From the analysis of the environment model, a group of the software behaviour will be identified. The behaviour of the IMR71848 transformation between the functions will be presented using the Ward-Mellor first level data flow diagram (DFD). The first level of the system behavior is then detailed by the Ward-Mellor's second level decomposition of each module. State transition diagram (STD) is used together with sequence diagram to model the behaviour of the robot system such as the control flow in the robot system.

3 SPECIFIC REQUIREMENTS

This section contains statements and discussions of requirements applicable to IMR71848 software. The discussion contains a statement of top-level requirements, refinements of requirements where there was agreement by the members of the working group, and an identification of issues that require further considerations.

3.1 Environment Model

The IMR71848 firmware environment model is defined using context diagram, (is shown in Figure 3), event list table (is shown in Table 2) and object timing estimation table (is shown in Table 3).

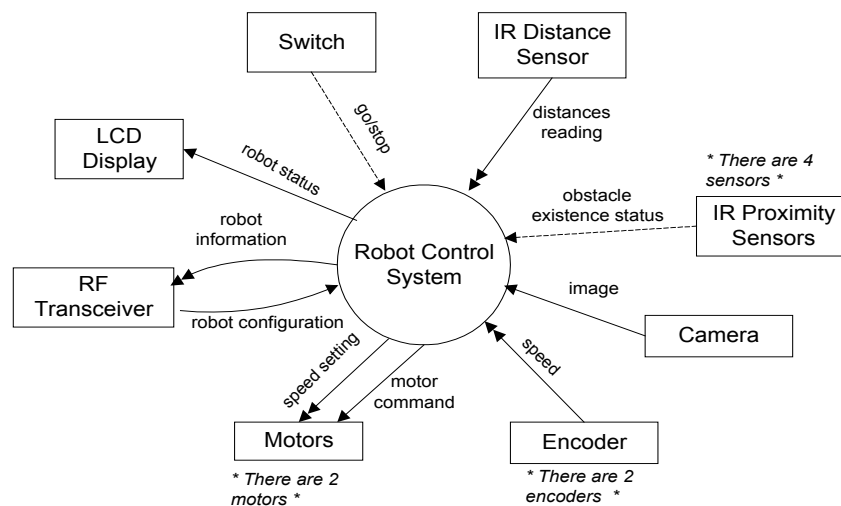


Figure 3: IMR71848 context diagram

From the context diagram the function of each object or device can be defined. Table 2 summarised the function of each object and lists of event that are connected to each function performed on each object. To perform the functions and react toward the listed events, each object was assigned with non-functional requirement, the timing specification for each object in the environment was estimated in Table 3.

Table 2: Event list table for robot control firmware

<i>Object</i>	<i>Function</i>	<i>Event list</i>
IR proximity sensors	To detect the presence of obstacles during the movement of the robot.	<ul style="list-style-type: none"> • Front IR proximity sensor indicate on • Back IR proximity sensor indicate on • Left IR proximity sensor indicate on • Right IR proximity sensor indicate on • Front IR proximity sensor indicate off • Back IR proximity sensor indicate off • Left IR proximity sensor indicate off • Right IR proximity sensor indicate off
IR distance sensor	To estimate the distance of obstacles and walls from the robot current location.	<ul style="list-style-type: none"> • No object in front • Object within 40-80 cm in front • Object within 20-39 cm in front • Exit found
Digital camera	To detect zone of objects in the robot's environment.	<ul style="list-style-type: none"> • Zone A • Zone B
Encoders	To change the tuning pulses reading to speed value of the motor.	<ul style="list-style-type: none"> • Speed > required speed • Speed < required speed
Motors	Actuating the DC motors according to the control instruction given to the motor.	<ul style="list-style-type: none"> • Increase the speed • Decrease the speed
RF transceiver	Received robot configuration commands from PC to remote control the robot and sends robot information to the PC to monitor robot sensor data.	<ul style="list-style-type: none"> • Received from PC on • Received from PC off • Transmit to PC on • Transmit to PC off
LCD display	Display information of the robot status for monitoring purposes.	<ul style="list-style-type: none"> • LCD display enable • LCD display disable
Switches	The operator has the ability to start and stop the robot.	<ul style="list-style-type: none"> • Operator switch on • Operator switch off • Operator switch go • Operator switch stop

Table 3: Minimum and maximum time for each object

Object	Min.-max. times
IR proximity sensors	0.5 sec – 1 sec
IR distance sensor	1 sec – 2 sec
Digital camera	50 sec – 1 min
Motors	50 millisecc – 100 millisecc
Encoders	50 millisecc – 100 millisecc
RF transeiver	1 sec – 10 sec
LCD display	10 sec – 30 sec
Switches	1 sec – 2 sec

3.2 Behavioural Model

3.2.1 First level Functional Behaviour Analysis

From the incoming and outgoing events analysis of the environment model, the IMR71848 firmware functional behaviour is presented in Figure 4 by capturing the robot systems environment. The eight objects from the context diagram are divided into three main group functions. The IMR71848 functions and the objects involve in each function is shown in Table 4.

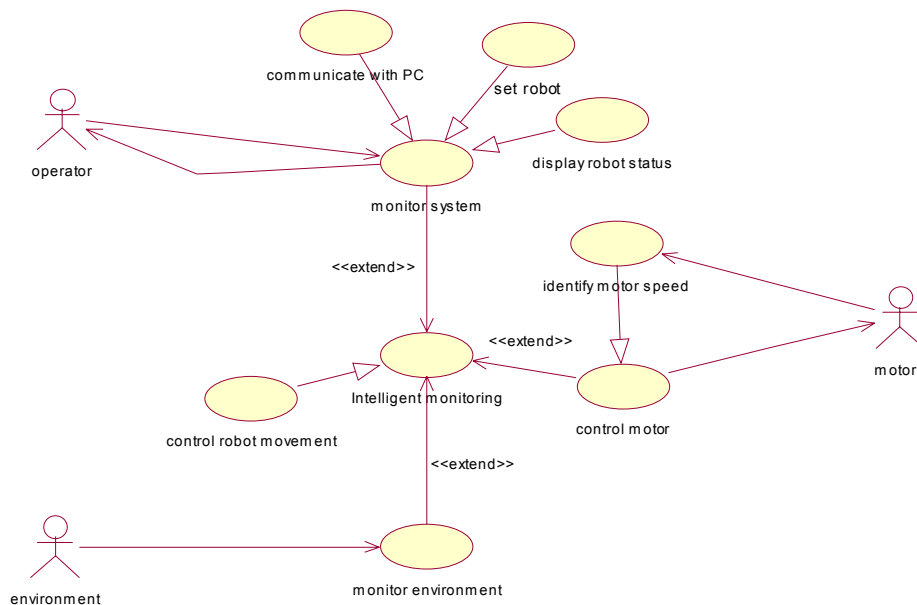
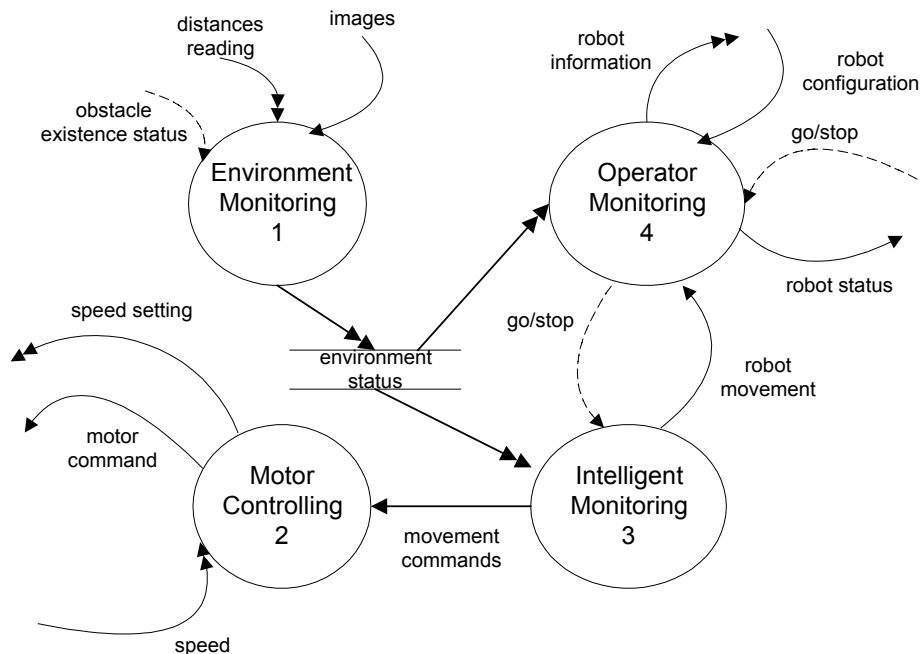
**Figure 4: IMR71848 use case diagram**

Table 4: Robot main function group and devices

Group function	Devices/Objects	Functions
i. Monitor Environment	IR proximity sensors, IR distance sensor and Digital camera	Monitor environment by information to the robot controller.
ii. Control Motor	Encoders and Motors	Actuating the DC motors based on the motor current speed and robot direction.
iii. Intelligent monitoring or also called Artificial Intelligence (AI)	none	High level and intelligent monitoring of the robot
iv. Monitor system or operator monitoring	RF transceiver, LCD display and Switches	Communicate with robot operators.

The IMR71848 first level functionality is shown using DFD in Figure 5. The first level DFD diagram, help to identify the essential abstractions of the IMR71848 domain, the domain diagram is presented using package diagram in Figure 6.

**Figure 5: The robot first levels DFD**

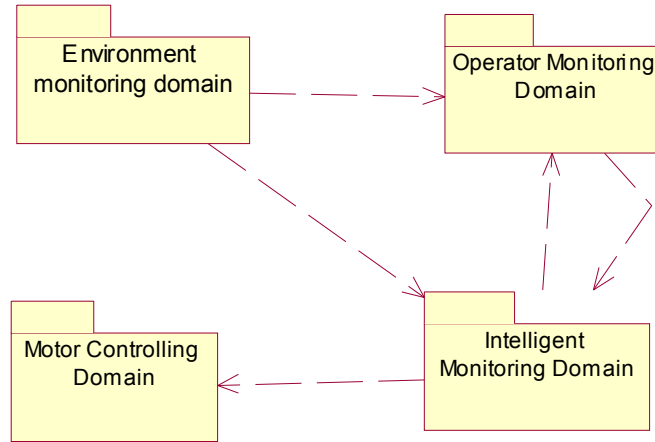


Figure 6: The IMR71848 Domain Diagram.

3.2.2 Detail Functional Behaviour Analysis

Each software group function, was decomposed into the following objects and sub-functions.

1. Environment Monitoring

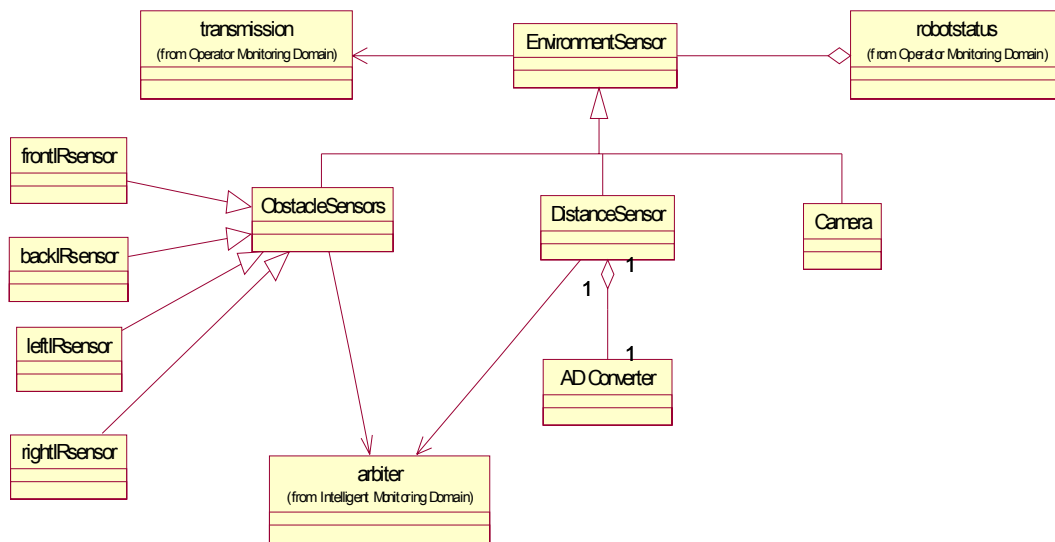


Figure 7: Environment monitoring class diagram

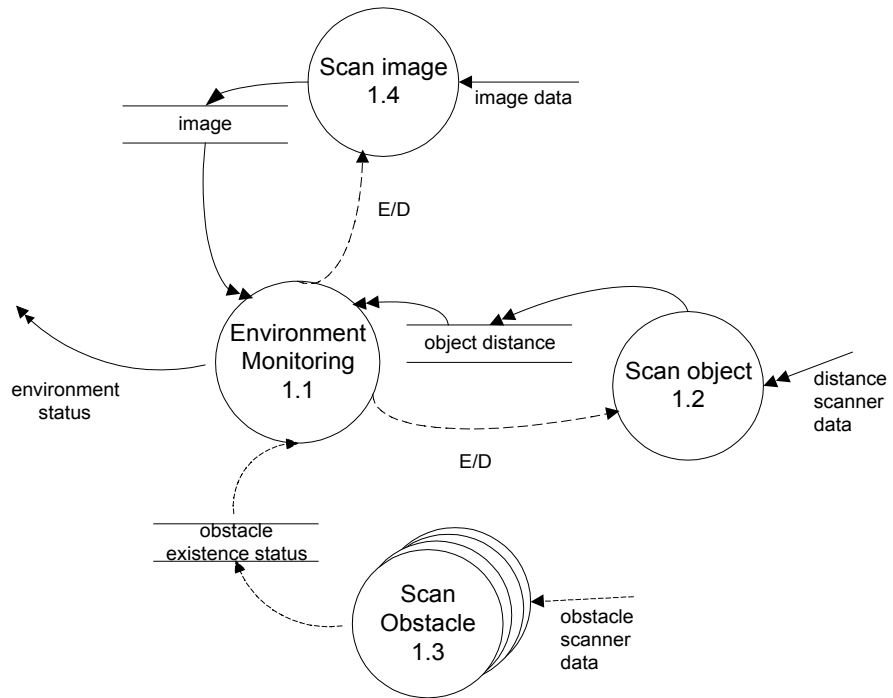


Figure 8: Environment monitoring DFD

2. Motor Controlling

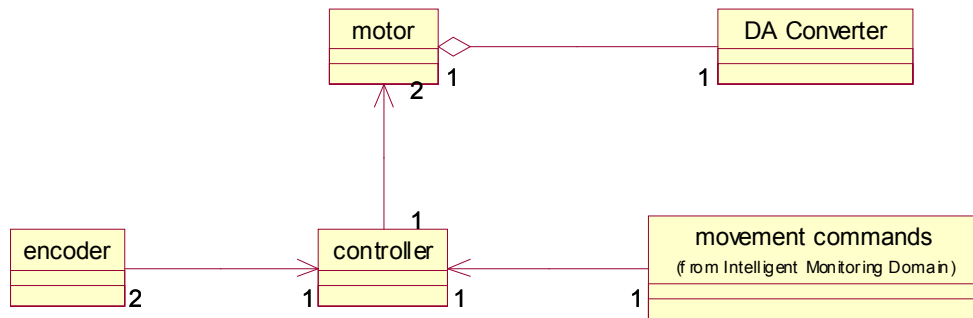


Figure 9: Motor controlling class diagram

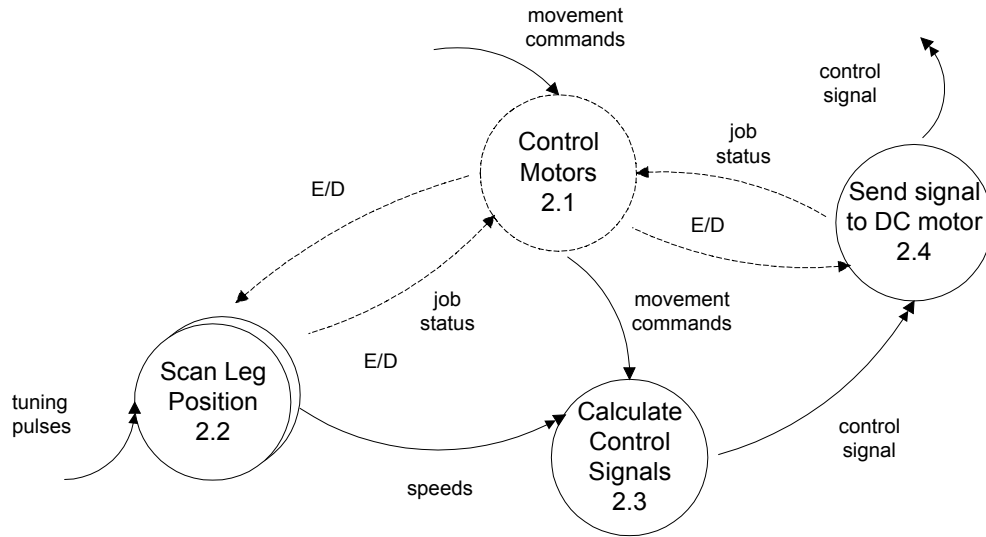


Figure 10: Motor controlling DFD

3. Operator Monitoring

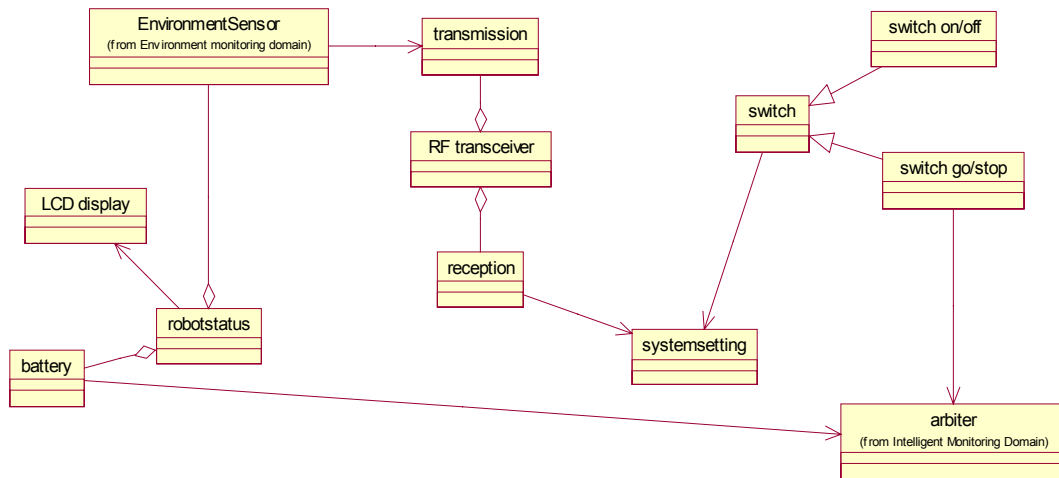


Figure 11: Operator monitoring class diagram

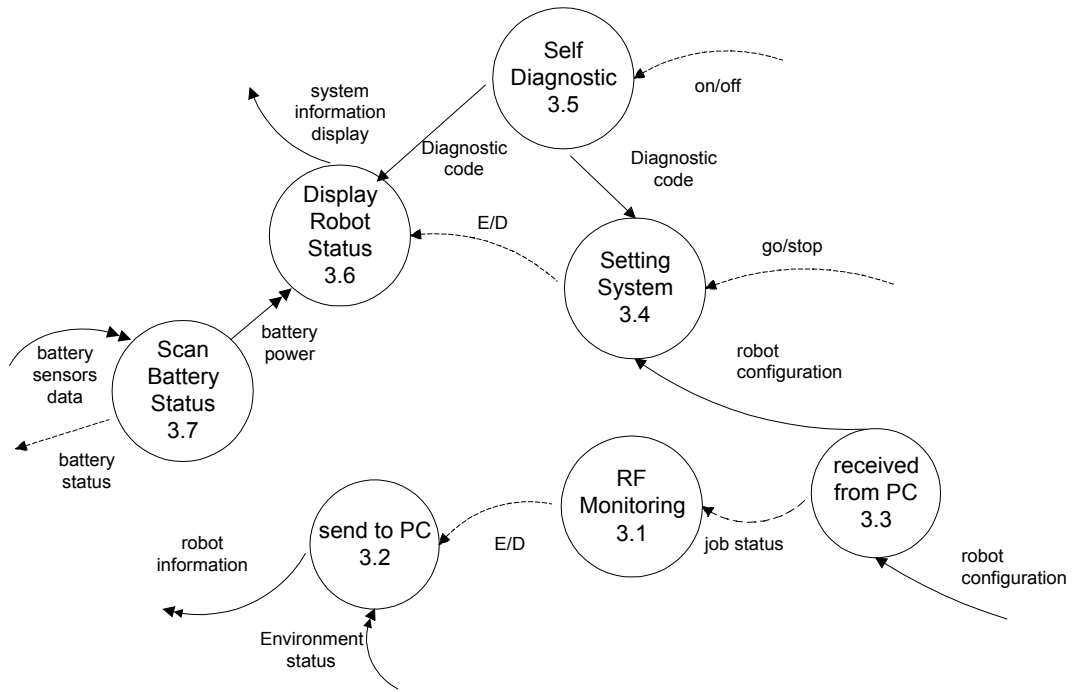


Figure 12: Operator monitoring DFD

4. Intelligent Monitoring

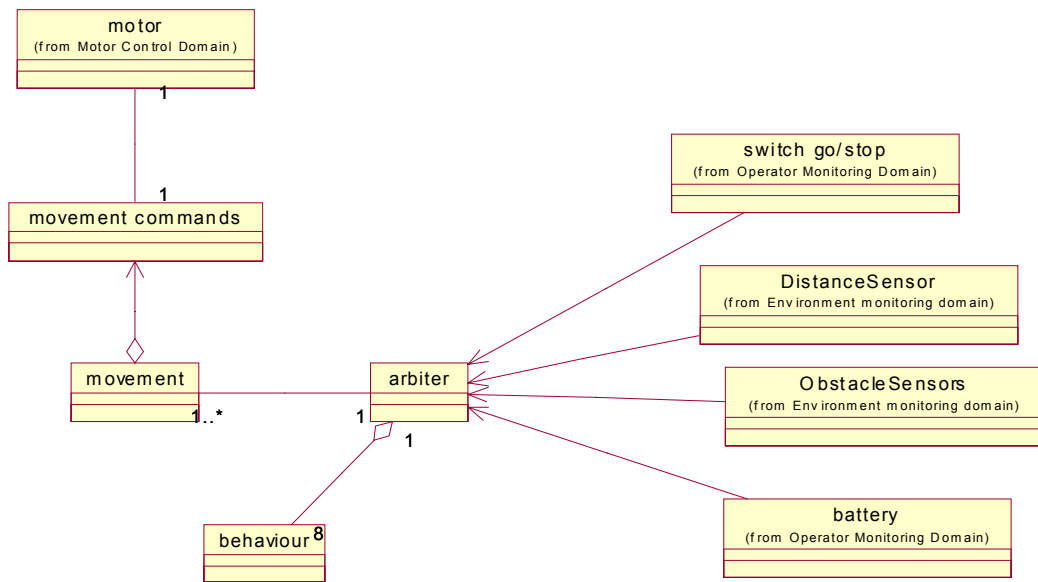


Figure 13: Intelligent Monitoring class diagram

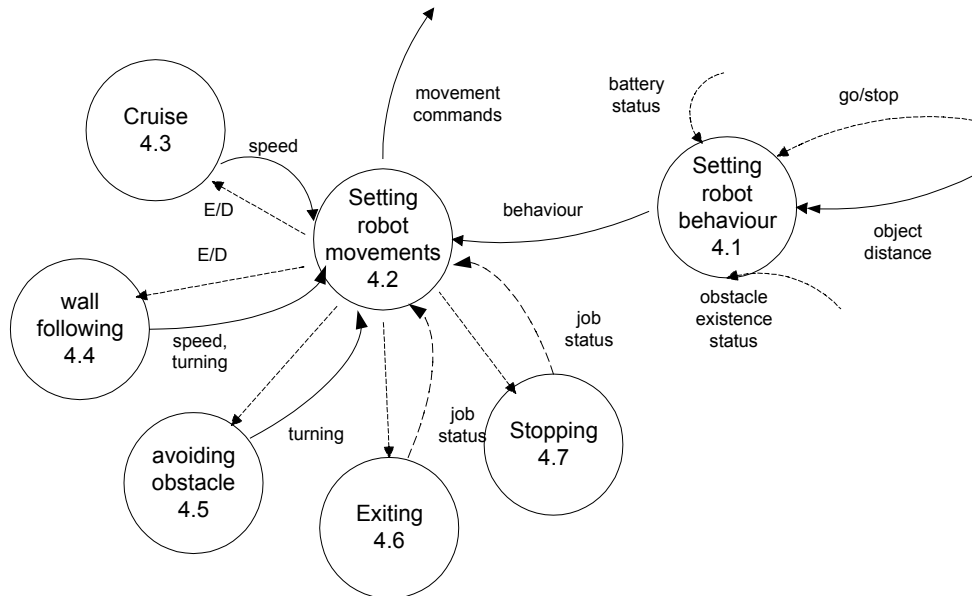


Figure 14: Intelligent Monitoring DFD

3.2.3 Non-functional Analysis

Based on the event lists from Table 2, each event are detailed to show the responses of the robot software toward the environments transformation. The response to each active event listed is shown in Table 5.

Statechart and sequence diagram were used to define the dynamic behaviour of the robot firmware. The analysis of the IMR71848 dynamic behaviour was analysed based on the event lists and responses from Table 4. Using the following sub-section the IMR71848 operation and the IMR71848 firmware dynamic behaviour was analysed.

Table 5: Events list and responses

	Event lists	Responses
	Environment Monitoring	
1.	Front IR proximity sensor indicate on	<ul style="list-style-type: none"> • stop • turn right • follow object
2.	Back IR proximity sensor indicate on	<ul style="list-style-type: none"> • forward
3.	Left IR proximity sensor indicate on	<ul style="list-style-type: none"> • follow object
4.	Right IR proximity sensor indicate on	<ul style="list-style-type: none"> • forward
5.	Front IR proximity sensor indicate off	<ul style="list-style-type: none"> • forward
6.	Back IR proximity sensor indicate off	<ul style="list-style-type: none"> • forward
7.	Left IR proximity sensor indicate off	<ul style="list-style-type: none"> • forward
8.	Right IR proximity sensor indicate off	<ul style="list-style-type: none"> • forward
9.	No object in front	<ul style="list-style-type: none"> • forward
10.	Object within 40-80 cm in front	<ul style="list-style-type: none"> • increase speed
11.	Object within 20-39 cm in front	<ul style="list-style-type: none"> • decrease speed
12.	Exit found	<ul style="list-style-type: none"> • turn left • forward • stop
13.	Zone A	<ul style="list-style-type: none"> • update zone
14.	Zone B	<ul style="list-style-type: none"> • update zone
	Motor Controlling	
15.	Speed > required speed	<ul style="list-style-type: none"> • decrease speed
16.	Speed < required speed	<ul style="list-style-type: none"> • decrease speed
17.	Increase the speed	<ul style="list-style-type: none"> • update PD calculation
18.	Decrease the speed	<ul style="list-style-type: none"> • update PD calculation
	Operator Monitoring	
19.	Received from PC on	<ul style="list-style-type: none"> • read data
20.	Received from PC off	<ul style="list-style-type: none"> • check latest data
21.	Transmit to PC on	<ul style="list-style-type: none"> • write data
22.	Transmit to PC off	<ul style="list-style-type: none"> • update data
23.	LCD display enable	<ul style="list-style-type: none"> • write display
24.	LCD display disable	<ul style="list-style-type: none"> • update robot status
25.	Operator switch on	<ul style="list-style-type: none"> • self diagnostic test • display robot status • wait for on
26.	Operator switch off	<ul style="list-style-type: none"> • disable the firmware
27.	Operator switch go	<ul style="list-style-type: none"> • enable arbiter
28.	Operator switch stop	<ul style="list-style-type: none"> • disable arbiter

1. Environment Monitoring

Event related with environment are IR sensors on or off, distance sensor range and zone identified by the camera.

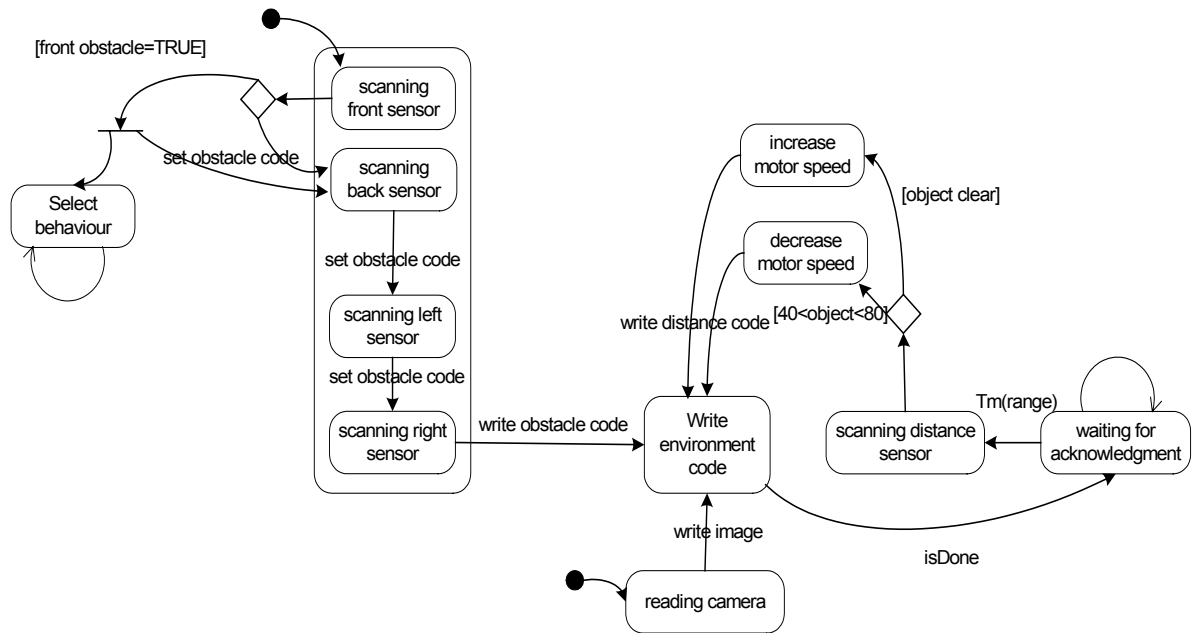


Figure 15: Environment behaviour.

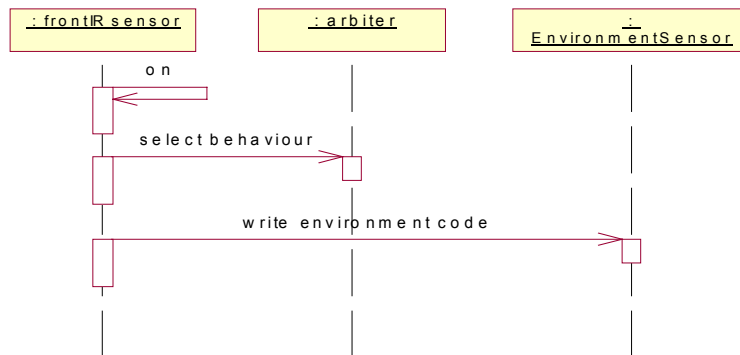


Figure 16: Front sensor on sequence behaviour.

2. Motor Controlling

Control motors

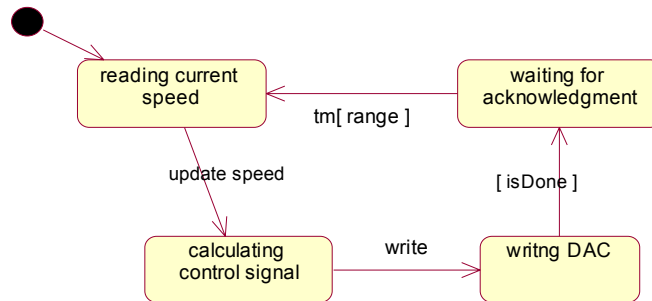


Figure 17: Control motor behaviour.

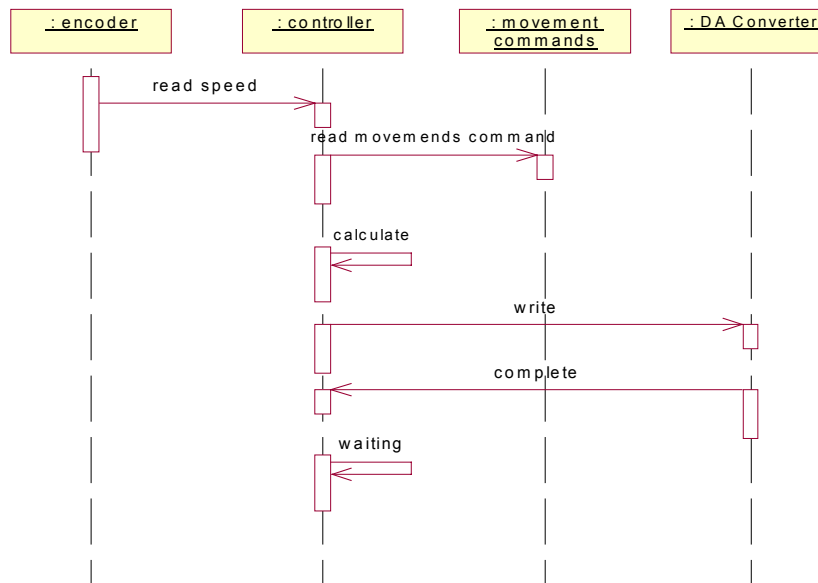


Figure 18: Control motor sequence behaviour.

3. Operator Monitoring

Event – Operator Switch on

The program starts to operate when the user switches on the robot and end when the user switches off the robot. This function includes: self diagnostic test and display system information the operator.

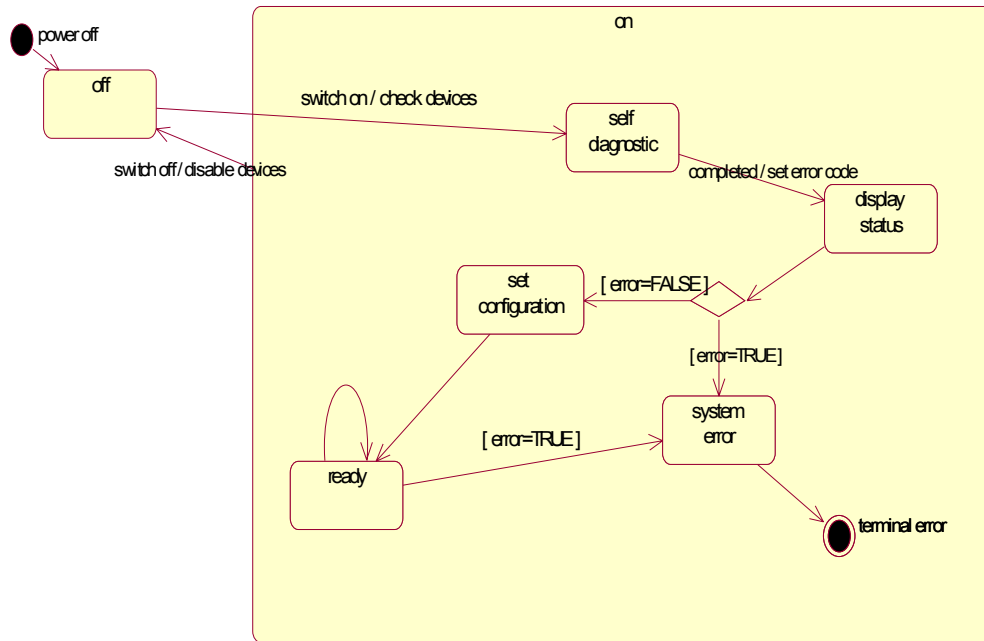


Figure 19: Operator switch “on”

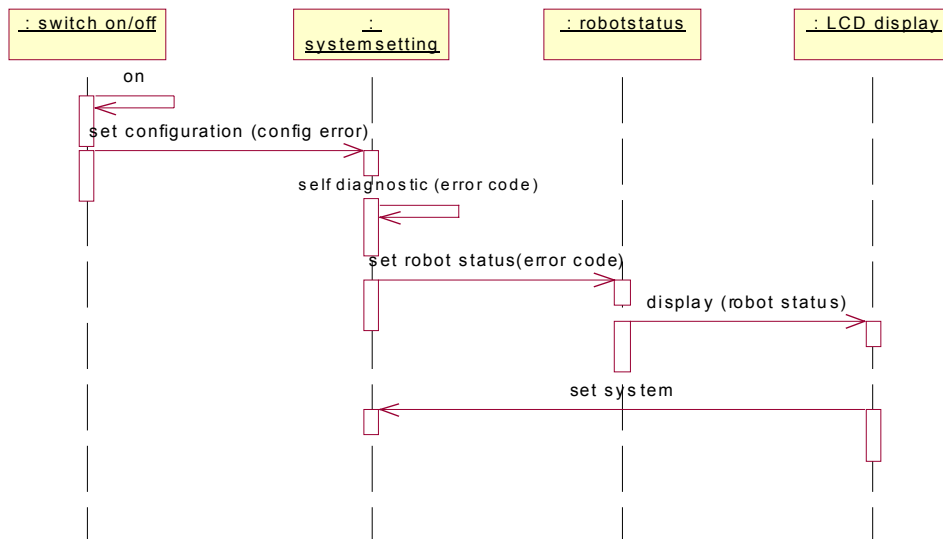


Figure 20: Operator switch “on” sequence diagram

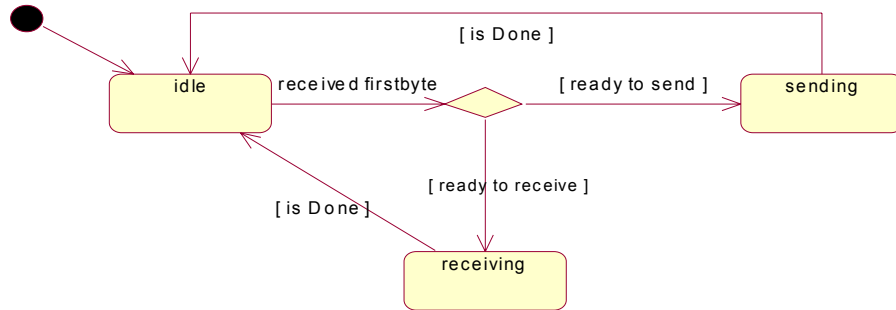


Figure 21: TR communication behaviour

4. Intelligent Monitoring

Event 1 – Operator Switch go

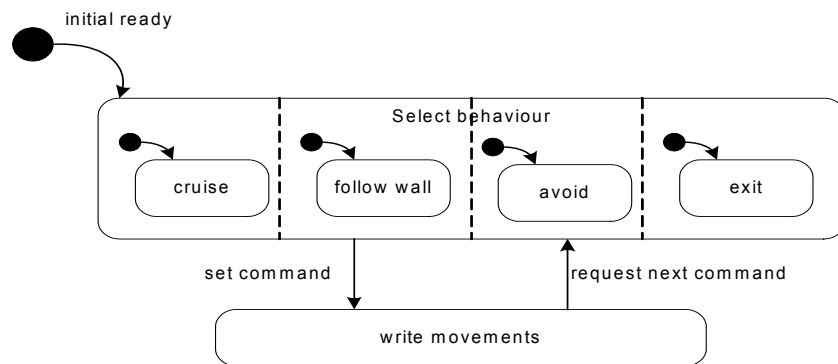


Figure 22: Operator switch “go”

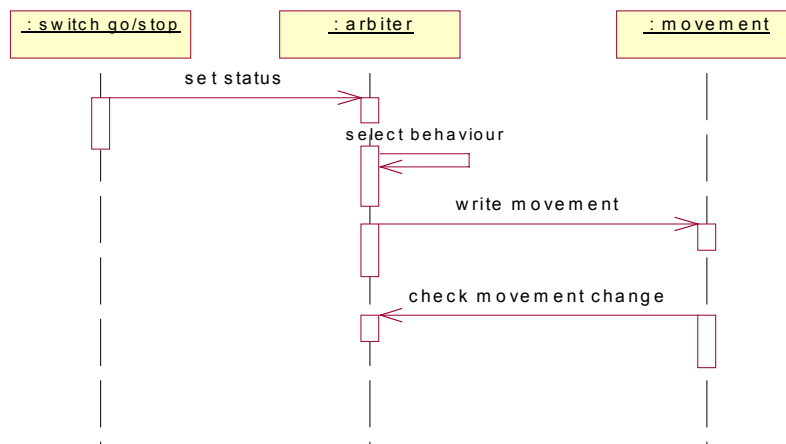


Figure 23: Operator switch “go” sequence diagram

EVALUATION OF HYBRID SOFTWARE ENGINEERING METHODOLOGY FOR DEVELOPMENT OF EMBEDDED FIRMWARE FOR INTELLIGENT AUTONOMOUS MOBILE ROBOT

Dayang Norhayati Abg. Jawawi^a, Ahmad Zariman Abd. Majid^a, Ahmad Ruzaimie Abd. Rashid^a, Dr. Safaai Deris^a, Dr. Rosbi Mamat^b, Radziah Mohamad^a
E-mail: dayang@fsksm.utm.my

^aDepartment of Software Engineering, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia

^bDepartment of Mechatronics and Robotics Engineering, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia

ABSTRACT

Software development for embedded real-time systems is very much different from the traditional data processing systems due to non-functional requirements such as dependability and the presence of hard timing constraints. Special tools and appropriate methodologies are therefore highly desirable for the development of embedded real-time software. Previous work on developing control firmware for Universiti Teknologi Malaysia-Wall Climbing Robot have faced with the problem of adopting a single methodology for developing the robot control firmware, a methodology called hybrid methodology was proposed. This methodology was tested on UTM Wall Climbing Robot under hardware-in-the-loop simulation and was found to be effective. However, a few questions are still to be answered and measured in order to fully test the effectiveness of the hybrid methodology on actual embedded system. The main objective of this paper is to evaluate hybrid Software Engineering methodology for developing control firmware at analysis and design phase. An intelligence mobile robot is developed to serve as embedded real-time system platform.

Keywords : *Software Engineering, Embedded Firmware, Intelligent Mobile Robot*

INTRODUCTION

A mobile robot is an autonomous system capable of traversing a terrain, performs its designated tasks, senses its environment and intelligently reacts to it. As the complexity and functionality of the robot is increased, such as adding more sensors to the robot so as to increase its reactivity and intelligence, designing and developing control software for this type of robot can be very difficult and a challenging task.

Issues related to real-time control, embedded system and artificial intelligence are involved in the mobile robot software development process. This type of software must be developed with proper software methodology or well-defined development process. Typically, the software or firmware is embedded in the onboard controller. To provide intelligence and reactive action, the robot firmware must sense its environment with multiple sensors and process the information and taking actions in real-time.

In a real-time system such as a mobile robot, the correctness of the system depends not only on the logical results, but also on the time at which the results are

produce. A mobile robot software system is inherently concurrent and multitasking since it has to react to and process numerous events simultaneously. Furthermore, this real-time software differs from the traditional data processing software in that it is constrained by non-functional requirements such as timing and dependability. In order to increase the software productivity, maintainability and flexibility in developing real-time robot software, proper software engineering need to be considered. Proper Software Engineering (SE) methodology ensures the software development process is manageable, and that reliable and correct program is constructed [2].

Previous work on developing control firmware for Universiti Teknologi Malaysia (UTM) Wall Climbing Robot [1] have found that Unified Modeling Language Real-Time SE methodology is still lacking in capability for control oriented hard real-time systems, Hard Real-Time HOOD SE methodology requires the support of ADA language, and Ward-Mellor SE methodology is inadequate for representing hard timing constraints. Faced with the problem of adopting a single methodology for developing the robot control firmware, a methodology called hybrid methodology was proposed. This methodology was tested on UTM Wall Climbing Robot software under hardware-in-the-loop simulation and was found to be effective. The aim of this paper is to evaluate hybrid SE methodology for developing control firmware at analysis and design phase. An intelligent mobile robot is developed to serve as embedded real-time system platform.

INTELLIGENT MOBILE ROBOT SPECIFICATION

The Intelligent Mobile Robot developed in this work is called IMR71848. The robot is a differential drive wheeled mobile robot, capable of traversing in an environment, which is surrounded by four walls. The task of the robot is to find a passage and exiting through the passage. Therefore, the goal of the IMR71848 software is to control the movement the robot in finding a passage and exiting through the passage.

The IMR71848 consists of a body and two pair of wheels. The body carries the embedded controller and its associated electronics, and others load. The cruising of the robot is supported a pair of drive wheels and a pair of castor wheels. Each drive wheels is move by a direct-current (DC) motor. The software controls the speed of DC motors, in order to move the robot.

The software will be embedded into the AMD188ES on board microcontroller The embedded controller for the IMR71848 with 128K EPROM and 128K RAM can be represented in block diagram form as shown in FIGURE 1. In order to move the robot and achieve the robot's goal, the software need to control the motor at each robot drive wheels, monitor the environments and navigate the robot intelligently.

During the cruising process, the IMR71848 software must support the intelligent components of the robot in order to ensure that the robot can response to the conditions in the environment in achieving the goal. The intelligent behaviour of the IMR71848 robot is supported by subsumption [3]. The navigation algorithm used by IMR71848 is wandering standpoint algorithm [4]. The algorithm try to reach from start in direct line, and when encounter an obstacle the robot will follow around the object until the goal direction is clear again.

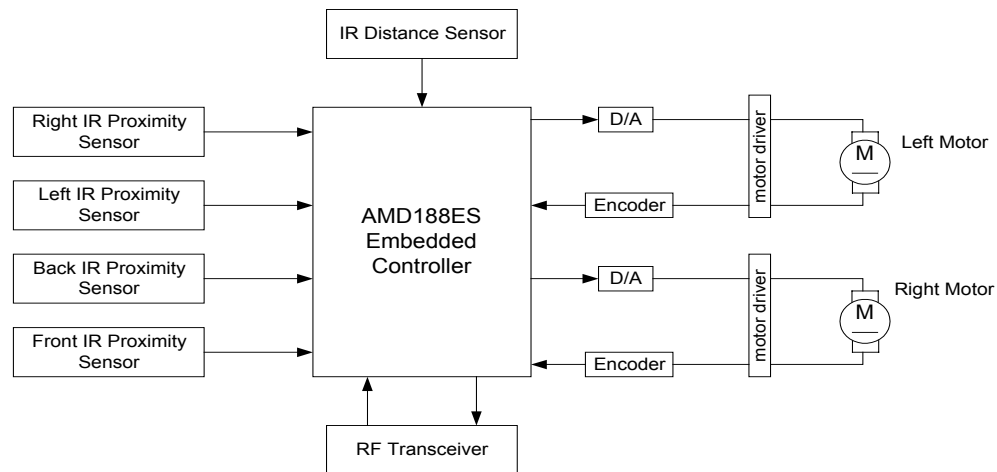


FIGURE 1: Block Diagram of IMR71848 Embedded Controller and Interfaces

The embedded controller monitors its environment using four infrared (IR) proximity sensor and a IR distance sensor. During the movement of the robot, the environment must be monitored, typically, every half a second to detect the presence of obstacles using the IR proximity sensors and measuring distance to wall every second using IR distance sensor. At each sampling period, the control signals to the DC motors are calculated using the proportional-integral (PI) control algorithm. The speed of the motor is sensed using the encoders and fed back to the embedded controller. The computation of the control signal typically, must be completed within 100 milliseconds to ensure the correct speed of the robot. The embedded controller also communicates with a remote PC to receive configuration commands and sending back information via a Radio Frequency (RF) transceiver.

The main functional operation of the robot controller can roughly be divided into four major tasks: high-level control, monitor environment, communication and cruise. To satisfy the timing requirements for these major tasks, a real-time kernel is used to achieve multi-tasking in the control firmware.

HYBRID SOFTWARE ENGINEERING METHODOLOGY

A hybrid methodology was proposed for the UTM Wall Climbing Robot control firmware development [1]. In the proposed hybrid methodology, several suitable notations and diagrams taken from Ward-Mellor Structured Development for Real-Time Systems (Ward-Mellor) [5], Unified Modeling Language for Real-Time (UML-RT) [6] and Hard Real-Time Hierarchical Object Oriented Design (HRT-HOOD) [7] methodologies were used to specify and design the robot control firmware. The models and techniques used in the hybrid analysis and design method are summarised in TABLE 1. In the specification stage the notations and diagrams from Ward-Mellor and HRT-HOOD were used. In the design stage notations and diagrams from UML-RT and HRT-HOOD were used.

The idea of combining several notation and tools from different software engineering methodologies for developing real-time system is not new in software engineering practice. The main advantage of adopting the hybrid method is that any appropriate notation and diagram can be chosen from different methodologies for functional and non-functional specification and design. The main disadvantage is that

no CASE tool support is available to assist the use of the hybrid method. However, due to the scale of IMR71848 project this can be handled manually.

EVALUATION OF THE HYBRID SE METHODOLOGY FOR IMR71848 FIRMWARE ANALYSIS AND DESIGN

To ensure the quality of IMR71848 firmware, the mobile robot development process should address the issues targeted toward the embedded IMR71848 system. Some issues, which will influence the software tools and software development process of the robot firmware are; small-scale system limitations, concurrency and multitasking, real-time requirement, evolving nature of the IMR71848 requirement and target hardware system.

<i>No.</i>	<i>Modeling stage</i>	<i>Techniques and tools used</i>	<i>Methodology used</i>
1.	Environment model	Outside-in structuring	Ward-Mellor
		<i>Tools</i> Context diagram Event list table Timing estimation table	Ward-Mellor Ward-Mellor HRT-HOOD
2.	Behavioural model		
2.1.	First level decomposition	Structured-object model	HRT-HOOD
		<i>Tools</i> Data flow diagram Functional group table	Ward-Mellor HRT-HOOD
2.2.	Detail functions decomposition	<i>Tools</i> Data flow diagram	Ward-Mellor
2.3.	Non-functional decomposition	<i>Tools</i> Event-response table State transition diagram Sequence diagram	Ward-Mellor Ward-Mellor UML-RT
3.	Design		
3.1.	Tasks decomposition and task behavioural	<i>Tools</i> Statechart diagram	UML-RT
3.2.	Task communication and synchronisation	<i>Tools</i> Task diagram	-
3.3.	Timing Performance	<i>Tools</i> Priority table	HRT-HOOD

TABLE 1: The Models and Techniques Used in the Hybrid Method Specification Analysis and Design

To effectively analyse the suitability of the hybrid method for the IMR71848 firmware development, some desirable criteria were set. These desirable criteria were derived from the IMR71848 firmware specification and development issues discussed above. The criteria include; (1) software development life cycle, (2) concurrency and multitasking model, (3) timing analysis, (4) maintainability and (5) others including simplicity, training, case tools and documentation.

Software Development Life-Cycle

The mobile robot system is still in developmental stage; as such the firmware will evolve in functionality and complexity. When more functionality is added, for example extending sensing capability of the robot, the firmware will change and the methodology used should be able to cope with these incremental changes. The nature

of the evolving IMR71848 project make the iterative and incremental process model is more suitable to be used. Therefore in the development of the robot firmware the incremental process model is adopted. Incremental process model combine elements of linear sequential model with the iterative prototyping, which enable software engineer to develop increasingly more complete version of the software [2].

The software life cycle for each increment model of the firmware development is summarised in FIGURE 2. Each phase of the life cycle is characterised by specific activities and the product produced by those activities. The products of each phase are shown in FIGURE 2 in italic besides each transition arrow.

Based on the firmware design, the implementation stage is to gradually develop the firmware part by part by building more and more functionality and non-functionality of the IMR71848 firmware using software tools: Borland C/C++ 3.1 compiler, ROM locator for generating ROMable code and μ C/OS-II real-time kernel.

Concurrency and Multitasking Model

In the development of the IMR71848 software, a real-time kernel will be used to provide multitasking and concurrency facilities. Software design with a real-time kernel will be much easier if tasks are properly structured. The hybrid methodology can provides task structuring in the design stage, and provides facilities for identifying and portioning the structured tasks into independent tasks group that will handle the concurrent activities.

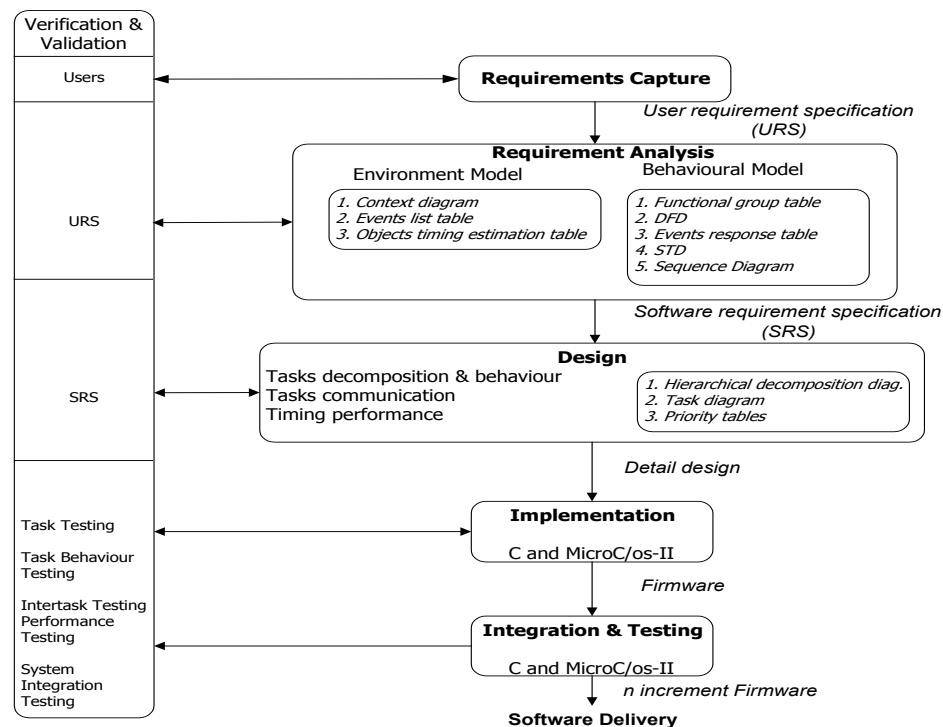


FIGURE 2: Software Life Cycle Of The Hybrid Methodology

In hybrid methodology, a system is structured into functions and the interfaces between them are defined in the environment and behaviour model described in

TABLE 1. This procedure is called *outside-in functional structuring*. The functions then will be grouped into concurrent tasks. The IMR71848 context diagram from environment model and the first levels data flow diagram (DFD) from behavioural model of the IMR71848 is shown in FIGURE 3 and FIGURE 4 respectively.

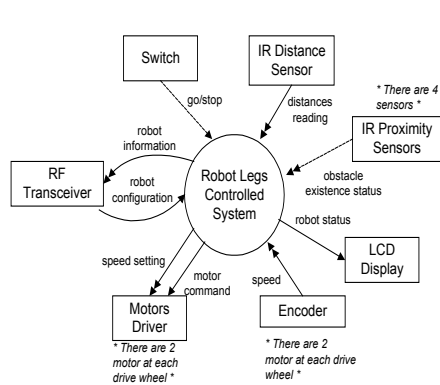


FIGURE 3: IMR71848 Context Diagram

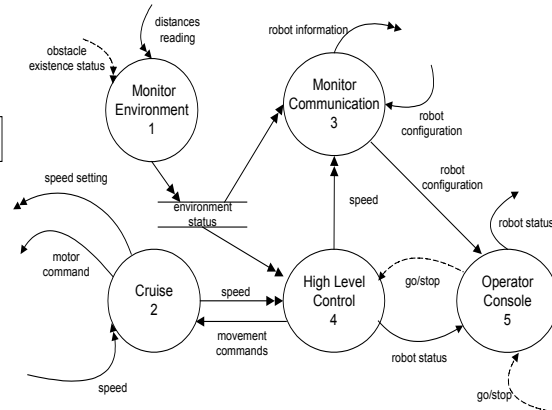


FIGURE 4: First Levels Data Flow Diagram for the IMR71848

The functionality of the robot is not the only type of requirement in the robot system. To perform the functions and react toward the events, each event was assigned with non-functional requirement. With the assistance of the functional requirements from context diagram and DFD the event-driven and time-driven requirements can be specified correctly using events tables, object timing estimation, state transition diagram (STD) and sequence diagram.

Three important design issues, related to the mobile robot firmware presented using hybrid method are tasks structuring or task decomposition, task communication and synchronization and timing. Further task decomposition is performed during design phase by detailing the IMR71848 behavioural model using statechart. The information offered by behavioural model were used to derive the statechart for the IMR71848 firmware, an example of the IMR71848 statechart is shown in FIGURE 5. To make sure that mutual exclusion between shared data is obtained in this concurrent system, a task diagram is introduced in the hybrid method [1].

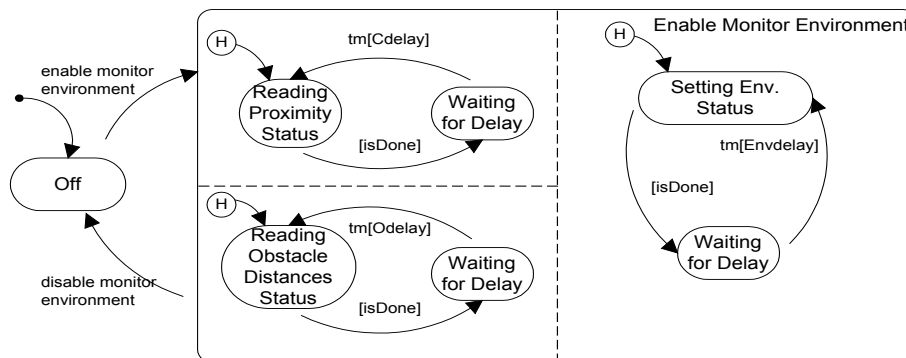


FIGURE 5: The IMR71848 Statechart for Environment Monitoring

The hybrid method proposes a functional-based task structuring with the assistance of sequence diagram and statechart diagram from object-based structuring. Object and functional tasks structuring seem to be natural for the IMR71848 software, with sensor and actuator objects, and control objects between them. It also can support concurrency and task structuring in a clear and visible manner.

Timing Analysis

The hybrid methodology address timing constraints during the analysis and design phase. During analysis, the response time specification is developed. The timing specification for each object in the environment was estimated at this stage and is presented in TABLE 2.

Object	Min.-max. times
IR Proximity Sensors – reading sensors	0.5 sec – 1 sec
IR Distance Sensor – reading sensors	1 sec – 2 sec
DC Motors – increase or decrease speed	100 millisecc
Encoders – read speed	100 millisecc
Communication – received or send data	1 sec – 2 sec

TABLE 2: Object Timing Estimation

Hybrid method classifies tasks according to temporal nature such as cyclic, function and protected task in design stage, this information is specified in task diagram. In order to schedule the execution of tasks using a preemptive real-time kernel, the priority of each task needs to be assigned. The timing constraints information derived from the IMR71848 firmware specification was used in the assigning tasks priority process. The timing was analysed using Rate Monotonic Scheduling technique [7] and based on the object timing estimation in TABLE 2, in order to initialise priorities for each task.

Sequence diagram is used to analyse the system timing by presenting the messages sequence and time constraints for particular message patterns between objects. It aimed to represent the timing constraints at different level and different objects. The sequence diagram, if event “IR Distance and Proximity Sensors indicate obstacle” occur, is shown in FIGURE 6.

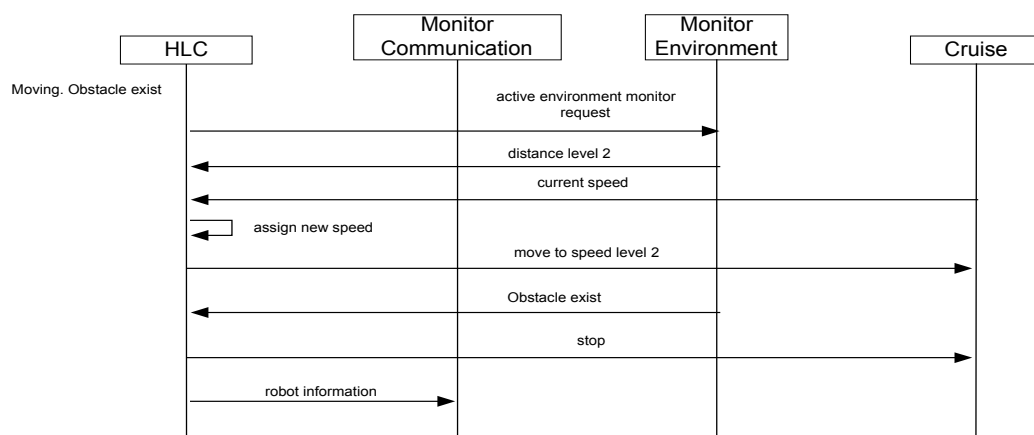


FIGURE 6: Sequence Diagram for Event “IR Distance and Proximity Sensors indicate obstacle”

This timing analysis approach supported by hybrid method is suitable for the specification of the embedded robot firmware as the timing constraints, which will affect the stability of the robot, can be specified much earlier.

Maintainability

The evolving nature of the IMR71848 system requires the firmware to be easy to maintain and modify. Traceability and adaptability of the IMR71848 specification and design is important in order to make the firmware easy to maintain and modify. The methodology should be traceable in term of technique used in derivation of requirement and design.

Context diagram gives the scope definition of system and environment, in which devices used to interact with environment such as sensors and actuator are considered external events, is shown in TABLE 2. This makes context diagram analysis in detail, which lead to unclear boundary between requirements and design. Therefore, there is the tendency to make design decision during the specification phase particularly if the specification gets detailed. This probably have some disadvantages to the changing specification of the developmental IMR71848 software. But considering the current complexity of the IMR71848 software, the context diagram of hybrid method is sufficient to model the system environment.

Simplicity and Good Documentation

The notation used in hybrid method is much simpler and some of the constraints are represented in textual notation. The advantage is that users can easily understand it and minimum training is needed. This is important in the development of IMR71848 system as the software designer are usually electrical engineers with little training in software development methodology. Because of the simple notation used, the non-functional requirements cannot fully be represented diagrammatically.

Half of the notation and tools used in hybrid method is adopted from Ward-Mellor methodology, which was introduced almost twenty years ago, so it is stable and many references, documentation and reviews of this methodology is available. Because of its simplicity and stability, the Ward-Mellor methodology is widely used in industry. The tools from UML-RT and HRT-HOOD methodology are mainly used to support non-functional nature of the embedded firmware.

CONCLUSION

The hybrid methodology, which combined several suitable notations and diagrams from three different methodologies, was used for the IMR71848 firmware analysis and design. The hybrid method is an enhance version of the Ward-Mellor method which better support for behavioural structuring, non-functional behaviour and timing specification in the analysis phase, and better support in design phase with emphasis on task decomposition, task synchronisation and task timing performance.

The suitability of the hybrid method for the IMR71848 firmware analysis and design was evaluated using some criteria: concurrency and multitasking model; timing analysis; maintainability and others including simplicity, training, case tools and documentation. It was found that the hybrid method flows of analysis and design can translate the mobile robot problem to implementation more naturally by considering the IMR71848 firmware development issues.

The hybrid method has been shown in this paper is suitable and practical for developing small-scale real-time embedded control software manually. Further work need to be done in order to use some of the CASE tools available assisting specification and design of the embedded real-time firmware. Currently our group is testing the firmware on actual IMR71848 system by measuring the exact performance of the IMR71848 firmware.

ACKNOWLEDGEMENT

The authors would like to thank Malaysia Ministry of Science, Technology and Environment and UTM Research Management Center (RMC) for supporting this work under funding VOT 71848.

REFERENCES

- [1] Dayang Norhayati Abang Jawawi (2000). "The Development of Real-time Control Firmware for A Wall Climbing Robot - A Small-scale Embedded Hard Real-time System." Faculty of Computer Science and Information Systems. Universiti Teknologi Malaysia: Master's Thesis.
- [2] Pressman, Roger S. (1997). "Software Engineering A Practitioner 'S Approach". Forth Edision. New York, U.S.A.: Mcgraw-Hill.
- [3] Brooks R. A. (1986). "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, Vol. RA-2, No.1.
- [4] Thomas Brauml, (2003). "Embedded Robotics – Mobile Robot Design and Applications with Embedded Systems", Springer.
- [5] Ward, P. T. And Mellor, S. J. (1985). "Structured Development For Real-Time Systems", Volume 1-3, New York: Yourdon Press.
- [6] Douglass B. P. (1998). Real-Time UML Developing Efficient Object For Embedded Systems, USA: Addison Wesley.
- [7] Burns, A., and Wellings, A. J. (1996). "Real-time Systems and Programming Languages." Second Edition. UK: Addison Wesley.

APPENDIX A

Intelligent Mobile Robot Software Structure and Behaviour

Compiled by:

Dyg. Norhayati Abg. Jawawi, Ahmad Zariman Abd. Majid, Ahmad
Ruzaimie Abd. Rashid



Department of Software Engineering
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
81310 Johor Bahru, Malaysia

<i>Abstract</i>	1
1.0 Definition	2
1.1 ISO Definition on Industrial Robot	2
1.2 ISO Definition on mobile robot (not official)	2
1.3 Definition Of Intelligent Mobile Robot	2
2.0 Structure	3
2.1 Goal Oriented	3
2.2 Task Orinted	3
2.3 Action Selection and Arbitration	4
2.4 Computational Paradigm	5
2.4.1 The Composite Local Model	6
2.4.2 The Sensor Model	7
2.4.3 Update Local Model	8
3.0 Behavior	8
3.1 Recognition	9
3.1.1 Recognition techniques	9
3.2 Manipulation	10
3.3 Navigation	11
3.3.1 Find Path	12
3.3.2 The Stanford Cart and the C-MU Rover	12
3.3.3 Hilare	13
3.3.4 Comment	13
4.0 IMR Project	14
4.1 Mobile Robot System to aid the daily life for physically handicapped - www.stakes.fi/tidecong/622taka.htm	14
4.2 MIT ARTIFICIAL INTELLIGENCE LABORATORY - www.ai.mit.edu	16
4.2.1 The Ants : A Community of Microrobots. - www.ai.mit.edu/projects/ants	16
4.3 ActivMedia Robotics - www.activmedia.com	18
4.3.1 POWERBOT - www.activrobots.com/ROBOTS/power.html	19
4.3.2 PIONEER 3-AT - www.activrobots.com/ROBOTS/p2at.html	20
4.3.3 PEOPLEBOT-	22

ww.activrobots.com/ROBOTS/peoplebot.html	
4.3.4 P3-DX8 - www.activrobots.com/ROBOTS/p2dx.html	23
4.4 Amigobot - www.amigobot.com/amigo/robots.html	25
4.5 CYBERBOTICS - www.cyberbotics.com	26
4.5.1 HEMISSON –	
www.cyberbotics.com/products/robots/hemisson.html	26
4.5.2 KHEPERA II –	
www.cyberbotics.com/products/robots/khepera.html	26
4.5.3 KOALA -	
www.cyberbotics.com/products/robots/koala.html	27
4.6 Angelus Research Corp. - www.angelusresearch.com	27
4.6.1 WHISKER - www.angelusresearch.com/Whiskers.htm	28
4.6.2 ADVANCE WHISKER –	
www.angelusresearch.com/advwhrs.htm	29

Abstract

Intelligence for robot to grow and evolve can be observed both through growth in computational power, and through the accumulation of knowledge of how to sense, decide and act in a complex and dynamically changing world. There are four elements of intelligence: sensory processing, world modeling, behavior generation and value judgment. Input to, and output from, intelligent system are via sensors and actuators. Recently, intelligent systems have been discussed in knowledge engineering, computer science, mechatronics and robotics. Various methodologies about intelligence have been successfully developed. As the scale and complexity of robot software increases, the successful construction of integrated robot systems depends less on the performance of any one particular algorithm and more on system architecture and decision processes in each element of the system. This paper is trying to find the true definition of Intelligent Mobile Robot by research through existing projects whether from research prospect or commercial prospect.

1.0 Definitions

1.1 ISO Definition on Industrial Robot

An *industrial robot* is an automatic, servo-controlled, freely programmable, multipurpose *manipulator*, with several axes, for the handling of workpieces, tools, or special devices. Variably programmed operation make possible the execution of a multiplicity of tasks.

1.2 ISO Definiton on mobile robot (*not official*)

A mobile robot is an autonomous system capable of traversing a terrain with natural or artificial obstacles. Its chassis is equipped with wheels/tracks or legs, and, possibly, a manipulator setup mounted on the chassis for handling of work pieces, tools, or special devices. Various preplanned operations are executed based on a preprogrammed navigation strategy taking into account the current status of the environment.

1.3 Definition Of Intelligent Mobile Robot

By the term "intelligent" in Intelligent Mobile Robot we mean that the navigation is "task-oriented" and that it is based on dynamically sensing and modeling the external world. The Intelligent Mobile Platform (IMP) is designed to respond to commands of the form "Go To <place>" where <place> is a pre-learned location in a network of "learned places". The IMP is able to use its network of places to plan a path to <place>. It is then able to use its sensing, modeling and navigation abilities to execute this plan and to modify the plan dynamically in reaction to unexpected events. The IMP is to serve as a foundation for household, business, and factory robots which require intelligent navigation. [1]

2.0 Structure

2.1 Goal Oriented

Navigation in indoor environment can be considered a goal-oriented task. In fact, normally some kind of a priori knowledge about environment is assumed and one of the main task main task of a mobile robot must be able to planning a path from its current position to the goal. Then the vehicle must be able to verify, continuously, if it is following the planned path and to detected (and successfully to avoid) unknown obstacles along the path. In literature, many references about goal oriented navigation can be found, using several kind of sensors: vision, odometers, gyro, laser and so on. Most of techniques can be grouped into two classes, dead-reckoning and external references based approaches.

Dead reckoning techniques determine present location of vehicles by advancing some previous position through known course and velocity data over known period of time. The most known implementation of dead reckoning is the odometry. The incremental estimation of position is affected by an incremental estimation of position is affected by the incremental growth of the error. Practical application on AGV, generally control error growth by integrating inertial navigation techniques (gyro based but are too much expensive) to dead reckoning or using external beacons to determine the correct current position and so positioning uncertainty can be reset.

External references based approaches use sensors placed on the vehicle to detect known landmarks in the environment in order to determine the robot position. Landmarks can be active or passive. Active landmarks are emitting beacons placed at known locations in the environment and the vehicles is equipped idoneous sensors able to detect them. Passive landmarks are detected, in most cases, by vision based techniques and they consist of natural landmarks.[2]

2.2 Task Oriented

a large application domain for multi-robot teams involve task-oriented missions, in which potentially heterogeneous robot must solve several distinct task. Consider the following problem: a team of heterogeneous mobile robot is required to perform a task oriented mission. Each robot on the team is programmed with the task capabilities necessary to perform a subset of task required by the current mission. In order to reduce the effect of bottlenecks and single point of failure, the robot are

designed to overlap in the task they are able to accomplish, although they may demonstrate different level of performance in accomplishing the same task due to robot heterogeneity. The capabilities of the robot in such a mission may change overtime, due either to robot subsystem failure or perhaps due to robot action learning. [3]

2.3 Action Selection and Arbitration

A robot operates by selecting actions that will achieve tasks and goals. Maes describes this as the action selection problem. All architectures must provide a solution for this problem. Identifying the action selection problem places the focus on building software that provides resources and mechanisms for action selection, thus while approaches may vary, action selection or more generally action arbitration is the bottom line for intelligent activity. Arbitration is not limited to the actuators of the system. It can be applied at various levels within a system to build up resources for action selection (e.g., a sensor fusion process), allocate resources (e.g., task/goal arbitration or planning), and control actuators (motor action arbitration). Figure 1.1, adapted from Bagchi, shows a generalized architecture that uses action selection to build up a robot abstraction for the tasks and goals of the system as well as to handle task and goal selection.

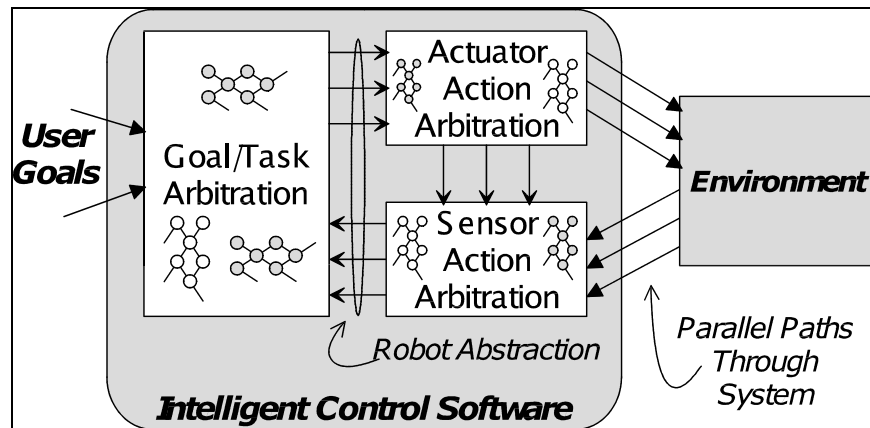


Figure 1.1 Fundamental Decision Process

Arbitration mechanisms form an important element of intelligent system behavior at many levels. In sensors processing, action selection results in sensor fusion. Sensor arbitration combines sensor data into logical sensors, much like those developed by Luo [9], to be used by other modules. In the motion control domain,

arbitration combines influences on motion (e.g., goal points, obstacles) to yield an overall motion that simultaneously meets several goals for the robot, similar to motor schema. For planning and sequencing, the arbitration mechanism predicts or activates a sequence of operations as shown by Bagchi. Thus action selection or arbitration is pervasive in the design of intelligent robots that include complex motion control, sensor fusion, behavior sequencing and task planning. The type of action selection mechanism used depends on the flexibility and structure of the system architecture. Rigid architectures typically allow only a single mechanism, while more flexible architectures combine several arbitration mechanisms at different points in the system. [4]

2.4 Computational Paradigm

In common usage, a paradigm is an example which serves as a model. A computational paradigm is a framework of data structures and processes which perform some task. The vision system of the IMP is based on the computational paradigm illustrated in figure 1.2

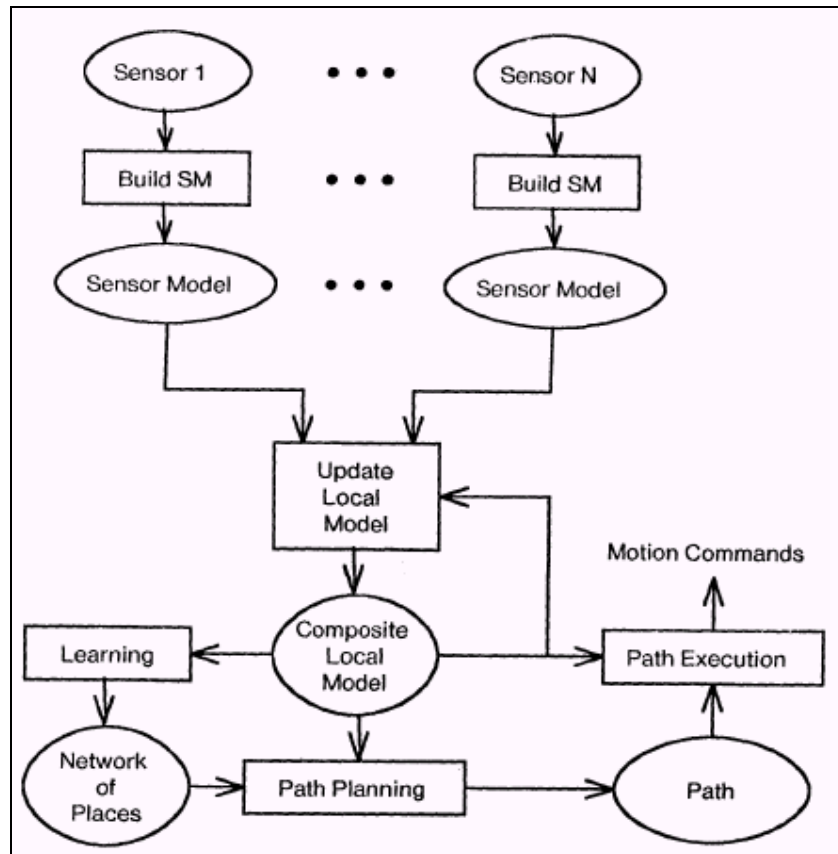


Figure 1.2 Framework for intelligent mobile platform

The navigation system of the IMP is based on maintaining a dynamic internal model of the local environment of the robot. An inexpensive rotating depth sensor continuously provides information about the external world. Differences between the sensor information and the internal model are used to indicate errors in the estimate of the position of the robot. The information from the sensor is then used to update the state of the internal model. This internal model also plays a crucial role in integrating sensor information and in providing reliable information for path planning, obstacle avoidance, learning, and path execution.

2.4.1 The Composite Local Model

At the core of this computational paradigm is a dynamic model of the surfaces and obstacles in the immediate environment of the IMP. This model is called "The Composite Local Model". "Local" refers to the fact that only information in the local environment of the robot is represented. "Composite" refers to the fact that this

model is composed of information obtained over time from multiple sensors and from many views. The Composite Local Model plays two fundamental roles in this computational framework.

- It is the structure in which potentially conflicting information from diverse sensors is integrated with recently observed information and information recalled from long term storage (in the case of the IMP: the global model).
- It is the structure on which processes for local path planning, path execution, learning, object tracking, object recognition, and other "higher level" processes are based.

Because of the nature of the navigation task and the sensors that are employed, the Composite Local Model in the IMP is implemented with a relatively simple 2-D representation. The IMP models the world and plans paths in a 2-D "flat-land" universe. Surfaces and obstacles are represented as connected sequences of line segments. Thus a table and a wall have the same structure; both appear as a barrier with an infinite (or unknown) extent in vertical dimension. The Composite Local Model must include the ability to represent the uncertainty of information. In the IMP, this ability is provided by a state transition mechanism. The line segments which compose the composite local model include a "state" attribute which represents both their source and varying degrees of uncertainty. Consistent line segments are reinforced and extended while inconsistent line segments are decayed and eventually removed from the model.

2.4.2 The Sensor Models

Sensors typically produce large amounts of information. Before the information from a sensor can be integrated into the Composite Local Model, surface information must be abstracted from it. This abstraction is performed by the module labeled BuildSM which produces a structure called the Sensor Model. The Sensor Model may be viewed as a form of "Logical Sensor" which provides the sensor information in a standard form which may be integrated into the Composite Local Model. In the first version of the IMP, the sensors are a set of contact sensors on a skirt and the rotating sonar sensor. In each case, BuildSM abstracts information in the form of line segments representing obstacles in the real world.

2.4.3 Update Local Model

The module labeled Update Local Model integrates the information from the Sensor Models with the current Composite Local Model. This module consists of two parts. The first part is a matching process which establishes the correspondence between the segments in the Sensor Models and the Composite Local Model as each line is produced by BuildSM. One of the side effects of this correspondence matching is an average error vector for the orientation and the position of the robot, This error vector tells the difference between the IMP'S estimated orientation and position and its actual orientation and position. Special procedures also exist for detecting and tracking moving objects. The second part is an integration step in which-the position, size, connectivity and confidence of the segments in the Composite Local Model are adjusted to reflect the results of correspondence matching. This second stage also removes segments for which the confidence is low or for which the distance is too far. The process does not remove nearby surfaces which are not currently visible. The problems of reconciling conflicting information and of representing uncertain information in the Composite Local Model involve interesting scientific issues. These problems are intimately related to the representation of the Composite Local Model. [5]

3.0 Behavior

Intelligent autonomous robots perform tasks according to different behaviors. We can assume that in general, each robot will act according to a set of behaviors, either fully or partially pre-defined. Suppose now that an agent has the ability to observe a robot and to autonomously identify which behavior the robot is performing. Generally Intelligent Robots consist these three sets of behavior.

Robot behaviors :

- Recognition
- Manipulation

- Navigation

3.1 Recognition

Recognition behavior explain how the vision subsystem is interfaced to the rest of the robot. This sub system is also used to locate suitable object from a distance identify it or interact with the object. Developing computational models for visual recognition and vision-based robotic control are some of the most fundamental problems in Cybernetics. In spite of strong interdependency between these functions in biological organisms, computational techniques for addressing these problems have typically been mutually exclusive. Generally, recognition techniques, neither seek to address the problem of vision-based control, nor do they attempt to formulate a framework linking recognition to purposive motion. Vision-based robotics on the other hand, either assumes the recognition problem to be solved or involves manual feature selection and correspondence as an essential part of the technique, thus implicitly addressing the recognition issues. Thus, while many recognition and vision based control techniques are available, there is need for research that studies and exploits the interaction of these problems.

3.1.1 Recognition Techniques :

Function based object recognition - page 272

A function-based recognition system reasons about observed object shape in order to determine what function the object might serve then classifies the object accordingly. Thus the recognition performed by a function –based system is inherently more generic than that performed by a system which matches an observed objects against a predefined set of geometric models

Symmetry exploration in 3d object recognition – page 257

A number of popular 3D object recognition paradigms are considered, including interpretation tree search, hypothesis and test, invariant feature indexing of interpretation tables, pose clustering and evidence based techniques. It is shown that

symmetries can be used to avoid the generation of equivalent recognition result, or to identify and filter out equivalent recognition result after they have been generated. The necessity and usefulness of symmetry exploration in object recognition leads to our believe that symmetry extraction algorithm should be integral part of the vision model preprocessor of such a system. [6]

3.2 Manipulation

The Control system for robot. The set of behaviors develop here allow the robot to acquire and retrieve object. (minimalist mobile robot – chapter 3) This control system demonstrates two important principles.

- Robot trajectory toward object is guided by the environment itself, rather than some plan developed from an internal world.
- Even a system composed of independent local agent can exhibit globally directed behavior. – overcome get caught in loops or local minima by removing central supervisor instead it is replace with another agent which monitor some sensory variable indicative of the overall process of the robot.

During the past years, significant efforts have been contributed to the research on the control design of a robot to perform the mechanical contact task on the environment. Contact task generally require the simultaneous consideration of both the robot's position and the interactions force. Three fundamental approaches have been identified.

1. Hybrid control suggest an approach to divide the robot's motion space into the position control subspace and the force control subspace.
2. Impedance control focuses on the design of a robot's mechanical impedance as seen from the environment. By utilizing the force feedback compensation, the robot mechanical impedance can be adjusted naturally as soon as it interact with the environment.

3. Model matching control fundamentally controls the robot 's position and pays attention to the frequency bandwidth of the robot control system with respect to the environmental dynamics. In order to adapt to the dynamical environment, a reference model of the robot position control loop is selected and accordingly a force feedback compensator is design to adjust the system bandwidth without using the hardware or software switches. [7]

3.3 Navigations

There are two major control strategies in the field of mobile control, in which actuators are controlled so that their output are regulated at the desired values determined from a given reference path and actuator-sensor configuration. The other is mobile robot guidance control in which the relative position of the mobile robot from reference is maintained as desired. In mobile robot guidance control, position and posture of the mobile robot must be measured or estimated based on the measurements of the relative position of the reference. Such information is also necessary for navigation. [8]

A number of interesting research results have been obtained on problems which are relevant to mobile robot navigation. A quick review of the salient systems provides a picture of the current state of the scientific art.

3.3.1 Find-Path

Planning a path based on a model is a problem that is fundamental to intelligent control of robot arms as well as mobile robots. Lozano-Perez has developed a formal version of the general path planning problem. This formalization is referred to as the "find-path" problem . In its most general form, the goal of find-path is to determine a continuous path for an object from an initial location to a goal location without colliding with an obstacle. Lozano-Perez provided a mathematical treatment of the find-path problem using the "configuration space" approach. The idea is to find those parts of free space which the object at particular orientations may occupy without colliding with an obstacle. Obstacles are "expanded" by the shape of an object at a set of orientations, while the object to be moved is shrunk to a point. The shortest path for the object, including rotations, is computed as the shortest connected path through the expanded obstacles. The shortest path through obstacles generally leads

through a sequence of points which are adjacent to the expanded obstacles. If there is position error in the control of the path execution, 3 such points can possibly result in a collision. Brooks has recently proposed a new approach to the find-path problem based on modeling free space . Brooks' solution was developed in a two dimensional plane. Brooks fit two dimensional "generalized cylinders" to the space between obstacles to obtain pathways in which the object may freely travel on a plane. The technique was extended to the third dimension by stacking planes.

3.3.2 The Stanford Cart and the C-MU Rover

Moravec developed a navigation system based on sensory signals using the Stanford cart. This cart sensed its environment using a set of 9 stereo images obtained from a sliding camera. A set of candidate points were obtained in each image with an "interest" operator. Small local correlations were then made at multiple resolutions to arrive at a depth estimate for the points. The matched points were plotted on a two dimensional grid and then expanded to a circle. A best path from the current location to a goal was then chosen as the shortest sequence of line segments which were tangent to the circles. The cart would advance by 3 feet and then repeat the sensing and planning process. Stereo matching was also performed between the images taken at different steps to obtain confirming and additional depth information. A new vehicle, called the C-MU Rover has recently been constructed by Moravec to support these techniques.

3.3.3 Hilare

A team under the direction of George Giralt at the LAAS laboratory in Toulouse has been investigating the design and control of mobile robots since 1977. They have developed a mobile robot named Hilare. Chatila developed a navigation system for Hilare which is based on dividing a pre-learned floor plan into convex regions [2]. Convex regions were formed by connecting nearest vertices to form areas called C-Cells. Laumond, at the LAAS in Toulouse, extended this idea by developing hierarchies of C-Cells to represent rooms and parts of a known domain [8].

3.3.4 Comment

A few other efforts towards developing autonomous mobile robots have also been reported. In many cases the efforts focus on engineering problems and pay little attention to the issues of world modeling or path planning. Other groups have become bogged down on the vision problem, often spending their efforts on general solutions to the problems of low level vision. We believe that the most important

problems to be addressed now are sensor interpretation, navigation, and system organization. Toward this end, we have developed a computational paradigm for intelligent robotic systems. This computational paradigm provides a framework for the processes involved in sensor interpretation, path planning, and path execution.[5]

4.0 IMR Project

4.1. Mobile Robot System to aid the daily life for physically handicapped - www.stakes.fi/tidecong/622taka.htm

Purpose: The purpose of this system is to bring daily using objects and putting them somewhere indoors semi automatically.

2 main concepts:

(1)Telecommunication using a computer network

Recently, a computer network became easy to use. If we can use computer network to control the mobile robot system, the mobile robot can controlled by any place, and also can transfer volumes of informations by LAN.

(2)Graphical and Interactive communication

The operator must realize safe robot motion in human working space, However, the operator of this robot is physically handicapped, he is not always engineer and professional to robot. Therefore, it is necessary to use graphical and interactive interface to operate the robot.

System Structure of Mobile Robot

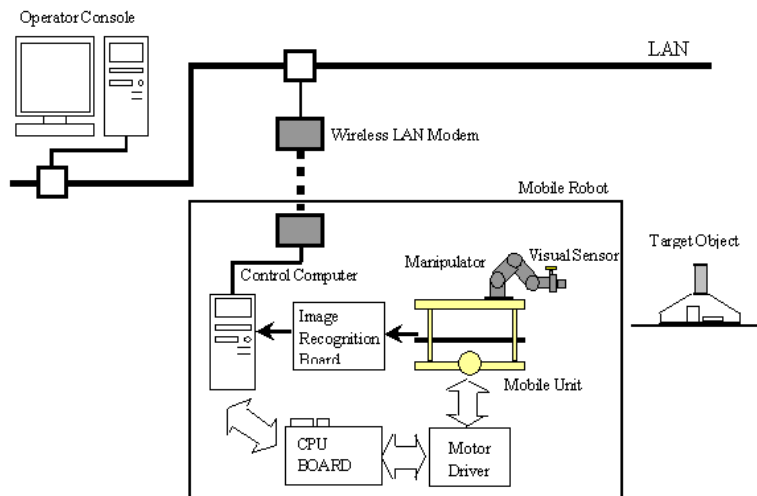


Figure1.3 System Structure of Mobile Robot

Interface System

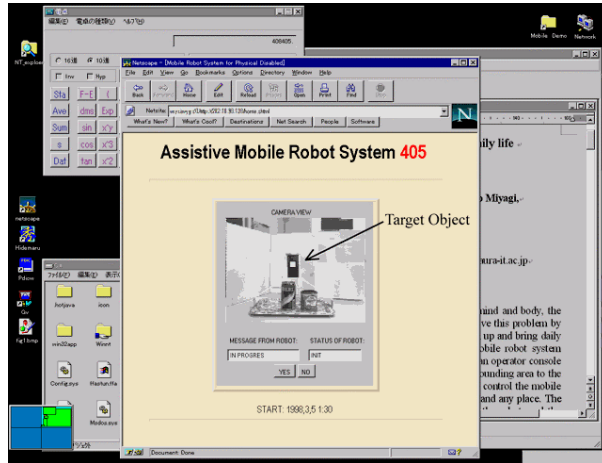


Figure.1.4 Image of Operator Console

- It has easy and interactive interface between human and robot.
- For example, it must have user-friendly input method, to show the real status of the robot and etc. for non-professional operator.
- The motion environment for this robot is that the target objects puts somewhere indoors, not fixed position, the operator must communicate to the robot through the operator console to look for the object, and also the robot must show the motion condition with easy understanding method.

4.2 MIT ARTIFICIAL INTELLIGENCE LABORATORY - www.ai.mit.edu

Introduction

The Artificial Intelligence Laboratory has been an active entity at MIT in one form or another since at least 1959. Our goal is to understand the nature of intelligence and to engineer systems that exhibit intelligence. We are an interdisciplinary laboratory of over 200 people that spans several academic departments and has active projects ongoing with members of every academic school at MIT. Our intellectual goal is to understand how the human mind works. We believe that vision, robotics, and language are the keys to understanding intelligence, and as such our laboratory is much more heavily biased in these directions than many other Artificial Intelligence laboratories. Our mode of operation is to attack theoretical issues and application areas at the same time. Even for theory however, we like to build experimental systems to test out ideas.

4.2.1 The Ants : A Community of Microrobots. - www.ai.mit.edu/projects/ants

Introduction : The ants are a community of cubic-inch microrobots at the MIT Artificial Intelligence Lab. There are two main goal of this project :

1. Push the limits of microrobotics by integrating many sensors and actuators into small package.
2. Form a structured robotic community from the interactions of many simple individuals.

Social Behavior

1. Clustering Around Food

Once the robot in the middle detest the food, she emits the “I found food” IR signal. Any robot within about 12 inches of her can detect the signal and head towards her. When a robot receives the “I found food” signal it heads towards the robot with the food while transmitting “I see an Ant with food”. Any robot within range of the second robot receives the “I see an Ant with food” signal, heads towards the second robot, and transmit “I see an Ant that sees an Ant with food”. It is like a robotic relay team.

2. Tag

The objectives of this behavior is for the single robot to seek out and tag (bump into) any of “Not It” robots. The “It” robot heads for the “Not It” that the other robots are transmitting from their IR beacons. When the “It” robot bumps into anything, it transmits “Tag” from its tag emitter. If the object that was bumped into is a wall or anything else boring, the “It” robots does not get the return signal and continues with whatever it was doing. If, however, the “It” robot bumped into a “Not It” robot, the “Not It” robot transmits “I got tagged” and then changes its mood from “Not It” to “It”. When the former “It” robot receives the “I got tagged” signal then its changes it’s mood to “Not It”.

3. Manhunt

Manhunt is like tag with teams. There is a red team and a green team. The object is for all the members of each team to tag all the members of the opposing team. When a robot is tagged, it changes teams.

The Software

The software for the Ants is written using programming style called Subsumption Architecture, developed by Prof. Rodney Brooks. The software on each robot is made up of many little programs, or behaviors. Each behavior monitors a few of the robot's sensors and outputs a motor command based on those sensor's readings. These commands are then sent to the motors based on a hierarchy; the outputs of more important behaviors override, or subsume, the outputs of less important ones.

4.3 ActivMedia Robotics - www.activmedia.com

Introduction

ActivMedia Robotics launched the Pioneer robot with Saphira API software in 1995 in collaboration with inventor Dr. Kurt Konolige and with iRobot. Since breaking with iRobot in 1997, ActivMedia Robotics systems have become the most popular intelligent mobile robotics development platform in the world. Its Pioneer robots are three-time winners of the World RoboCup Soccer Championship as well as many American Association of Artificial Intelligence contests. The company has collaborated on grants from DARPA and NIH. ActivMedia's AmigoBot™ has been chosen as test platform for the new Intel low-power XScale board, designed specifically for portable and wearable applications. ActivMedia robots have appeared on Discovery Channel, Scientific American Frontiers and other popular venues.

In 2001, the company shipped its 1,000th robot. In February, 2002, the company released its revolutionary new ActivMedia Robotics Interface for Applications (ARIA)

with Saphira 8+. ARIA with Saphira 8+ expands the former capabilities of Saphira to provide developers with a complete software development solution from robot OS to gradient navigation and localization to integrated accessories such as grippers, cameras and integrated I/O bus for custom accessories.

In April, 2002 the company announced its first general purpose application, LaserPlans™. In May, the PatrolBot™ surveillance & monitoring system was released. In October, 2002, it announced its third commercial product; the PowerBot AGV for flexible manufacturing and lab automation.

4.3.1 POWERBOT - www.activrobots.com/ROBOTS/power.html

Introduction

POWERBOT™ is an amazing high-payload high-speed highly maneuverable platform with all the intelligence of our smaller platforms. PowerBot™ moves up to 6 kph with a payload up to 100kg. It is built on the same core client-server model as all ActivMedia robots, making it software compatible. PowerBot™ offers a full-sized PC computer option, opening the way for onboard vision processing, Ethernet-based communications, laser, DGPS, and other autonomous functions. The PowerBot™ stores up to 2100 watt-hours of swappable batteries. Fourteen forward and 14 rear sonar sense obstacles from 15 cm to 7 m. PowerBot's powerful motors and two monster wheels on steel frame with suspension is designed for higher speeds with good response. The PowerBot™ uses 500 tick motor encoders. Its sensing extends far beyond the ordinary with laser-based navigation options, GPS, bumpers, 6 dof DC arm, vision, and a rapidly growing suite of other options.

The PowerBot has the ability to :

1. Wander randomly, avoiding obstacles
2. Drive controlled by keys or joystick
3. Communicate sensor & control information including sonar, motor encoder, motor controls, user I/O and battery charge data
4. Run C/C++ programs created with or without ARIA or our other software developments environments.

With Laser Mapping and Navigation option, PowerBot can :

1. MAP rooms, labs or buildings in minutes.
2. Plan path to goal.
3. Avoid obstacles along the way.
4. Navigate tight spaces for delivery or docking.

PowerBot is designed for industrial, commercial and research use :

1. Handling
2. Delivery
3. Mapping
4. Navigation
5. Monitoring
6. Reconnaissance
7. Vision
8. Cooperation

4.3.2 PIONEER 3-AT - www.activrobots.com/ROBOTS/p2at.html

Introduction

PIONEER 3-AT is a highly versatile all-terrain robotic platform, software-compatible with all ActivMedia robots, chosen by many DARPA grantees and others requiring a high-performance robot with plenty of real estate for customization. Powerful, yet easy to use; reliable, yet flexible, P3-AT is a popular team performer for outdoor or rough-terrain projects.

P3-AT offers an embedded computer option, opening the way for onboard vision processing, Ethernet-based communications, laser, DGPS, and other autonomous functions. The P3-AT stores up to 252 watt-hours of hot-swappable batteries. Optional 8 forward and 8 rear sonar sense obstacles from 15 cm to 7 m. P3-AT's powerful motors and four monster wheels can reach speeds of .8 meters per second and carry a payload of up to 30 kg. The P3-AT uses 100 tick encoders with inertial correction recommended for dead reckoning to compensate for skid steering. Its sensing extends far beyond the ordinary with laser-based navigation options, integrated inertial correction to compensate for slippage, GPS, bumpers, gripper, vision, stereo rangefinders, compass and a rapidly growing suite of other options.

The bare P3-AT base with included ARIA and Saphira8+ software has the ability to:

1. WANDER randomly
2. DRIVE controlled by keys or joystick
3. PLAN PATHS with gradient navigation
4. DISPLAY a map of its sonar and/or laser readings
5. LOCALIZE using sonar (with optional laser upgrade)
6. COMMUNICATE SENSOR & CONTROL information relating sonar, motor encoder, motor controls, user I/O, and battery charge data
7. TEST ACTIVITIES QUICKLY with ARIA API from C++ programs
8. SIMULATE BEHAVIORS OFFLINE with the simulator that accompanies each development environment

The Pioneer 3-AT is an all-purpose outdoor base, used for research and prototyping applications involving:

1. Mapping
2. Navigation
3. Monitoring
4. Reconnaissance
5. Vision
6. Manipulation
7. Cooperation

P3-AT's are made for use outdoors; they run on many earth, stone or paved surfaces. Unencumbered, they can climb steep 45% grades. To operate on carpet, select Indoor wheels, which may be swapped as needed. P3-AT's are not water-proof.

4.3.3 PEOPLEBOT – www.activrobots.com/ROBOTS/peoplebot.html

Introduction

PeopleBot™ provides a base for service or performance robots. The handsome black and silver PeopleBot™ offers a gripper, table-sensing IR's and precise pan-tilt-camera with ActivMedia Color-Tracking Software (ACTS) for sensing and grasping objects on tables. Included demo uses state machines to recognize a colored object, fetch it from one table and set it on another.

Peoplebot has the ability to :

1. Play sound files or synthesized speech
2. Listen for phrase or sounds it recognizes
3. respond to requests or conditions it senses
4. Navigate without running over toes or into furniture
5. Find and Fetch objects it recognizes
6. Follow colors
7. Transmit video images to surveillance monitors
8. Communicate with other robots
9. Connect to PC's via the internet or LAN
10. Run autonomously

At 112 cm (45 in), the PeopleBot™ stands midriff to chest height on most adults.

Designed for use by seasoned professionals, the PeopleBot™ can be programmed in C or C++. PeopleBot™ is ideal for prototyping, research or applications such as:

1. Tour guides
2. Waiters
3. Messengers
4. Monitors and guards
5. Trade shows
6. Exhibition
7. performances
8. Education
9. Research
10. Cooperative tasks

PeopleBot™ runs indoors on flat floors. It can traverse low sills and household power cords. With upper and lower sensing, the PeopleBot™ will turn away from nearly all obstacles. Performance PeopleBot™ also has the ability to sense tabletops and move its gripper into place for picking up objects. PeopleBot™ can run five days a week for six hours a day without maintenance for years. More intensive use may require regular factory maintenance, which is available by contract.

PeopleBot™ bases may communicate with each other via Ethernet and cooperate in teams.

4.3.4 P3-DX8 - www.activrobots.com/ROBOTS/p2dx.html

Introduction

PIONEER 3-DX8 is an agile, versatile intelligent mobile robotic platform updated to carry loads more robustly and to traverse sills more surely with high-performance current management to provide power when it's needed. Built on the same core client-server model as all ActivMedia robots, the P3-DX8 offers an embedded computer option, opening the way for onboard vision processing, Ethernet-based communications, laser, DGPS, and other autonomous functions. The P3-DX8 stores up to 252 watt-hours of hot-swappable batteries. It arrives with a ring of 8 forward sonar and with an optional 8 rear sonar ring. 3-DX8's powerful motors and 19cm wheels can reach speeds of 1.6 meters per second and carry a payload of up to 23 kg. In order to maintain accurate dead reckoning data at these speeds, the Pioneer uses 500 tick encoders. Its sensing moves far beyond the ordinary with laser-based navigation options, GPS, bumpers, gripper, vision, stereo rangefinders, compass and a rapidly growing suite of other options.

The bare P3-DX8 base with included ARIA and Saphira software has the ability to:

1. Wander randomly
2. Drive controlled by keys or joystick
3. Plans paths with gradient navigation
4. Display a map of its sonar and/or laser readings

5. Localize using sonar (with optional laser upgrade)
6. Communicate sensor & control information relating sonar, motor encoder, motor controls, user I/O, and battery charge data
7. Run C/C++ programs created with or without Saphira or other software developments environments
8. Test activities quickly with Saphira's Colbert real-time programming language
9. Simulate behavior offline with the simulator that accompanies each development environment.

With ActivMedia Robotics Basic Suite software, this robot can travel point-n-click to a location on your map. Basic Suite also allows the robot to be easily programmed for demos or instructional purposes. With Laser Mapping & Navigation System, your robot can map buildings and constantly update its position within a few cm while traveling within mapped areas.

The Pioneer 3-DX8 is an all-purpose base, used for research and applications involving:

1. Mapping
2. Teleoperation
3. localization
4. monitoring
5. reconnaissance
6. vision
7. manipulation
8. Cooperation

P3 -DX8's run best on hard surfaces. They can traverse low sills and household power cords and climb most wheelchair ramps.

4.4 Amigobot - www.amigobot.com/amigo/robots.html

This robot has the ability to :

1. sense their environment and respond to it
2. operate autonomously
3. protect themselves and let you know their current status
4. allow you to modify their behaviors
5. can be operated over the Internet
6. serve as videoconferencing devices

AMIGOBOT TMePRESENCE is the choice for those who want to see and hear what's happening around the robot, whether the robot is in the next room or on the other side of the world. AmigoBot TM ePresence includes all the hardware and software you need to turn your desktop PC into an Internet robot operation and chat center. You can:

1. drive the robot locally or online, up to 300 feet from your PC
2. create maps for the robot to navigate
3. send the robot to make deliveries, point-n-click, avoiding obstacles along the way
4. see and hear from the robot's point of view
5. take snapshots of what the robot sees
6. see the robot's 8 sonar display
7. read the robot's action status messages
8. program new behaviors
9. modify demo behaviors
10. make the robot talk
11. download your own sounds to the robot
12. control who shares the robot online
13. chat with people sharing the robot

4.5 CYBERBOTICS - www.cyberbotics.com

Introduction

Cyberbotics was founded in 1998 by Olivier Michel, as a spin off company from the MicroComputing and Interface Lab (LAMI) of the Swiss Federal Institute of Technology, Lausanne (EPFL). Cyberbotics is developing Webots, a 3D mobile robot simulator for research and education. Cyberbotics is also developing custom 3D mobile robot simulators for a number of companies and universities.

Products :

4.5.1 HEMISSON -

www.cyberbotics.com/products/robots/hemisson.html

Features :

- Cost effective
- Designed for education
- Works in remote control or autonomously.
- Capabilities: obstacle avoidance, line following, wall following, dance (LED flashing, buzzer), drawings, etc.
- Applications: teaching, programming contests.
- Includes BotStudio programming software (DeLuxe only)
- Includes Webots simulation software (Deluxe only)
- Supported platforms: Windows and Linux (Deluxe only)
- Works in remote control or autonomously

4.5.2 KHEPERA II -

www.cyberbotics.com/products/robots/khepera.html

Features :

- Compact
- Easy to Use
- Affordable
- Powerful Microcontroller
- Many Sensor and Actuator Extensions
- Many Software, Curriculum, Papers
- Applications: navigation, artificial intelligence, multi-agents systems, control, collective behavior, real-time programming.

4.5.3 KOALA - www.cyberbotics.com/products/robots/koala.html

Features :

- Compact size
- Highly modular with a wide variety of expansion options
- Powerful computational capabilities
- Full Khepera compatibility
- All-terrain indoor experiments
- Applications: telemanipulation, path planning, object research, recognition and transport, surveillance, tour guide, Automatic vacuum cleaner

4.6 Angelus Research Corp. - www.angelusresearch.com

Introduction

the worlds largest supplier of intelligent robots with over 1000 robots in use around the world. Our Educational line of Whiskers Robots teach the world's children about the next technology wave beyond the Internet: Intelligent Robots and Machines. Our Military Robots like Intruder and ART are used to make the most dangerous job in the world: Military Operations, safer. Our military line of robots are dual use, also providing Law Enforcement activities support in life threatening situations. Piper the Robot inspects pipes protecting the environment by detecting problems before they become Ecological disasters.

Mission :

- To bring intelligent machines and robots into our society for the betterment of mankind.
- To educate young people in this new and exciting technology that will touch all of our lives.

Products

4.6.1 WHISKER - www.angelusresearch.com/Whiskers.htm

Whiskers is a highly intelligent robot which can be programmed in an English like language. It is highly motivational and educational for kids from Elementary School through High Schools. This is the most popular intelligent robot in Technical Education today. Students explore computer science careers in less than ten days without any prior programming experience. Due to Whiskers unique capabilities, many Universities use this robot as well.

Whiskerstm emulates the three levels of the human brain in real-time.

- Easy to program, no experience required
- User can teach the robot new commands
- Durable all heavy gauge aluminum

The Software Architecture is based on the three levels of intelligence found in the Human Brain :

1. **Cerebral Cortex** - Physically the outer layer of the brain, which is characterized by the folds just under the skull. Functions include: Decision making, analysis, and dreaming. This is called the Goal Level in the intelligent operating system.
2. **Limbic System** - The gray matter found in the center of the brain controls human behavior such as breathing, hunger, etc. This is called Behavior Level. Real-time decisions are made and simple or complex actions are triggered.
3. **Brain Stem** - The base of the brain connected to the spinal cord and nervous system. This level controls our critical responses and instinctive behavior giving the machine common sense. This is called Instinct Level. Motor/sensor fusion at this level allows the machine to instantly react to its environment. The Behavior and Goal levels can change the way this level reacts at any time.

Consider what happens when you are cooking and you touch something hot. Your skin feels the heat and your muscles immediately pull your hand away (Instinct Level). A message is sent to your brain (pain) which causes your brain to make decisions on what actions are to be taken next (Behavior Level). After you have taken care of your burned finger, you resume your original task(Goal Level).

Touching - Whiskers give the robot the ability to touch objects around him.

Seeing - Four independent optical sensors use Light Emitting Diodes (LED's) and photo-transistor pairs. They give Whiskers the ability to see objects around him.

Feeling - Wheel load or drag is measured continuously to give the robot a sense of the terrain. It allows Whiskers to sense objects the other sensors did not see.

Speaking - The speaker and software that controls it, gives Whiskers the ability to make sounds just like other animals use for communication.

4.6.2 ADVANCE WHISKER - www.angelusresearch.com/advwhrs.htm

This robot is an advanced version of Whiskers. It adds a head subsystem including sonar ranging and four additional optical sensors. It uses two onboard computers networked together so students can experiment with intelligent robots that can navigate about their environment.

The Software Architecture is based on the three levels of intelligence found in the human brain :

- 1. Cerebral Cortex-** Physically the outer layer of the brain, which is characterized by the folds just under the skull. Functions include: Decision making, analysis, and dreaming. This is called the Goal Level in the Triune Operating System.

- 2. Limbic System** - The gray matter found in the center of the brain, controls human behavior such as breathing, hunger, etc. This is called Behavior Level. Real-time decisions are made when simple or complex actions are triggered.

- 3 Brain Stem** - The base of the brain is connected to the spinal cord and nervous system. This controls our critical responses and instinctive behaviors. It is analogous to the Instinct Level which gives the machine common sense. Motor/sensor fusion allows the machine to instantly react to its environment. The Behavior and Goal levels can alter the Instinct's reaction at any time.

Consider what happens when a person touches something hot. The nerve endings in the skin detects the heat and causes an immediate muscular response (Instinct Level). Additionally, a message (pain) is sent to the brains' Limbic System that activates a higher level behavior or set of actions based on programmed behaviors or learned experiences. This behavior or actions pre-empt the Cerebral Cortex (Goal Level) while the behavior is executing. When the action is finished, the Cerebral Cortex (Goal Level) regains control and continue where it left off or it may decide to change strategies or goals.

Moving - Two independent DC motors provide locomotion using an advanced pulse width modulation motor speed control. Speeds can be controlled from one to one hundred percent in one percent increments.

Touching - Two whiskers on the base section are used for tactile sensors.

Seeing - Four independent optical sensors are mounted on the base using Light Emitting Diodes (LED) and phototransistors pairs. A proprietary narrow beam sonar system is mounted in the panning head section for navigation and long range sensor scans. The sonar can detect object distances to one eighth of an inch. A single Visible Red LED sensor located in this section has the capability to see about three to four feet. Three optical sensor arrays are located in the non-moving collar section for additional object detection.

Feeling - A force feedback system is used to monitor wheel load. Force is measured continuously to monitor the surface type or load. It is sensitive enough to determine whether the robot is operating on carpet or hard flooring.

Thinking - The processor in the base section performs real-time collision avoidance while the head processor navigates and scans the environment, simultaneously. The two computers are networked together. This allows them to cooperate in solving the navigation problem.

Learning - The language used for programming the onboard computers is English. No prior programming experience is necessary to create new commands (words) for this robot. However, the very tools used to create this easy to use and powerful language is always available to the user. The user words actually become part of language. The potential for this robot is limited only by the users imagination.

References

- [1] Nenad M. Kircanski, Mobile Robotics Systems, University of Toronto,2-4
- [2] E. Stella, Goal Oriented Mobile Robot Navigation Using an Odour Sensors, University di Lecce, 147
- [3] Lynne E. Parker, Task Oriented Multi Robot Learning in Behaviour Based System, Center For Engineering Systems Advanced Research, 1478
- [4] Pacman,wilkes,kawamura, A Software Architecture for Integrated Service Robot Development, Center for Intelligent Systems Vanderbilt University Nashville, 1
- [5] James L. Crowley, Navigation for an Intelligent Mobile Robot, The Laboratory for Household Robots The Robotics Institute Carnegie-Mellon University of Pittsburgh, Pennsylvania, 5-11
- [6] Book – modeling and planning for sensor based intelligent robot system,
- [7] Zhi Wei Luo, Multiple Robot Manipulators Cooperative Compliant Manipulation on Dynamical Environment, Department of Information and Computer Science Toyohashi University of Technology, 1927
- [8] N. Matsumoto, Mobile Robot Guidance Control with Nonlinear Observer Based State Estimation, Research Laboratories Nippondenso Company Limited,2264