

Wednesday, July 9, 2008

Mr. Patrick Leung
School of Engineering Science
Simon Fraser University
Burnaby, B.C. V5A 1S6

Re: ENSC 305/440 Post-Mortem for a Home Care Robot

Dear Mr. Leung,

PRAAlpha would like to present the post-mortem for our ENSC 305/440 project. This document outlines our design and development process, the implementation of our device, the budget and timeline, individual achievements and our team dynamics.

The objective of our project was to build a multi-functional home care robot platform which would be capable of manipulating objects and allow future add-ons and user-definable functionalities.

PRAAlpha Inc. was founded by three outstanding senior engineering students: Rick Wong, Jessica Sun and Feng Ye. If you have any concerns or questions, please feel free to contact us by phone (604)780-5392 or by email, pr-440@sfu.ca.

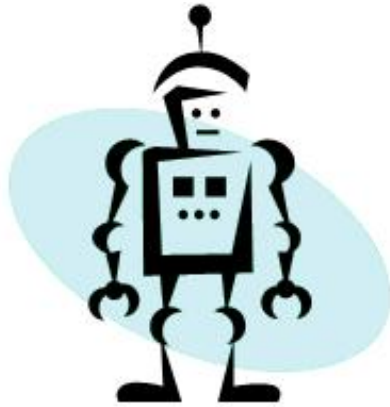
Sincerely,



Rick Wong (CEO),
PRAAlpha Robotics Inc.

Enclosure: *Post-Mortem for a Home Care Robot*

PRAAlpha Robotics Inc.



Post-Mortem for a Home Care Robot

Issue Date: July 9, 2008

Version : 1.0

Document ID : ETC-004

Project Team: Rick Wong
Jessica Sun
Feng Ye

Contract Person: Rick Wong
rickbow@sfu.ca

Submitted to: Steve Whitmore
Patrick Leung
Brad Oldham
Jason Lee

**PRAAlpha
Robotics**

Proprietary and Confidential

© Copyright 2007 PRAAlpha Robotics Inc.

All rights reserved. No part of this publication may be used or reproduced in any form by any means without prior written permission from PRAAlpha Robotics Inc.

Document Information

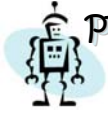
Document Name	Post-Mortem for a Home Care Robot
Document ID	ETC-004
Version	1.0
Issue Date	July 9, 2008

Revision History

Issue	Author(s)	Date	Description
0.1	Jessica Sun	June 25, 2008	<ul style="list-style-type: none"> ✧ Creation ✧ Introduction
0.2	Jessica Sun Rick Wong	June 30, 2008	<ul style="list-style-type: none"> ✧ Budget ✧ Conculsion ✧ Team Dynamic ✧ Future Improvment
0.3	Jessica Sun Feng Ye	July 3, 2008	<ul style="list-style-type: none"> ✧ Personal Reflection ✧ PC ✧ Encoutered Problem
0.4	Jessica Sun	July 5, 2008	<ul style="list-style-type: none"> ✧ Deviation of device ✧ Timeline
0.5	Rick Wong	July 6, 2008	<ul style="list-style-type: none"> ✧ Encoutered Problem ✧ Personal Reflection
0.6	Rick Wong	July 7, 2008	<ul style="list-style-type: none"> ✧ Current state of device
0.7	Jessica Sun	July 7, 2008	<ul style="list-style-type: none"> ✧ Current state of device ✧ What we would do differently

Table of Contents

Document Information.....	2
Revision History	2
Table of Contents	3
List of Figures	5
List of Tables.....	5
Acronyms	6
Glossary	7
1 Introduction.....	8
2 Current State of the Device.....	8
2.1 System Overview	8
2.2 Chassis.....	10
2.3 CMM	10
2.4 SCM	11
2.5 ACM.....	11
2.6 DBM.....	11
2.7 UDM.....	12
2.8 Pico.....	12
2.9 GUI Client	12
3 Device Deviations.....	14
3.1 Overall System	14
3.2 Sensor System	14
3.3 Motion System	14
3.4 Computer System	14
3.5 User Interface	15
3.6 Power System.....	15
4 Encountered Problems, Solutions and Limitations.....	15
4.1 IR Control.....	15
4.2 I2C BUS	15
4.3 DBM.....	16
4.4 USB Flash Drive	16
4.5 Video Delay.....	16
5 What We Would Do Differently and Future Improvements	17
5.1 What We Would Do Differently	17
5.2 Future Improvements	17



5.2.1	IR commands self-learning	17
5.2.2	Step motor with feedback.....	17
5.2.3	Smaller IC packaging	18
5.2.4	Microcontroller wireless controlling.....	18
5.2.5	Self positioning	18
5.2.6	Charging base.....	18
5.2.7	GUI.....	19
6	Budget and Timeline	19
6.1	Budget	19
6.2	Timeline.....	20
7	Inter-Personal and Technical Experiences	22
7.1	Rick Wong	22
7.2	Jessica Sun.....	23
7.3	Ye Feng.....	25
8	Team Dynamics	26
9	Conclusion	27
	References.....	28

List of Figures

Figure 1: Overall System Block Diagram	9
Figure 2: Hardware Layers of the System	10
Figure 3: Displaying error message in Debug Module	11
Figure 4: User Interface	13
Figure 5: Project Timeline	21

List of Tables

Table 1: Estimated and Actual Cost Comparison	19
Table 2: Unit Cost	20

Acronyms

Term	Definition
ACM	Action Control Module
ANSI	American National Standards Institute
CMM	Center Manager Module
DIP	Dual In-lin Package
DTAC	Dynamic TWI Address Configuration with pre-defined priority
I ² C	Inter-Integrated Circuit, a mutli-master serial BUS
IR	InfraRed
LUT	Look Up Table
MCU	Micro Conroller Unit
MLF	Micro Lead Frame, a type of scale microchip package
QFN	Quad Flat No leads Package, used with surface mounted PCB
PCB	Printed Circuit Board
PWM	Pulse-Width Modulation
SCM	Sensor Control Module
SSH	Secure Shell Handler
TQFP	Thin Quad Flat Pack, used in space-conscious application, PC cards
TWI	Two-Wire serial Interface, Atmel's standard of I ² C BUS
UDM	User Defined Module
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
USB	The Universal Serial Bus

Glossary

Term	Definition
Authenticate	The system confirm the user's identity by verifying the input password
Control Daemon	A multi-threaded program running in the background to provide TCP/IP connection of the GUI client. It transfers the commands from the GUI client to the Robot and also transfers messages from the Robot to the GUI client
GUI Client	The program with Graphic User Interface used to control the robot and display messages from the robot
Input Capture Channel	A method of dealing with input signals in an embedded system
IR Controller	Infrared sender used by the robot to control electrical devices
Java run-time libraries	A set of computer files and programs installed on a computer so that Java programs can run on it
Public computer	A computer provided in public locations like libraries, air port, hotels or Internet cafes
Stand-alone application	A computer program that can be executed alone without the help of other programs
Stream Server	A program that provides video stream through HTTP protocol
Ultrasonic	sound with a frequency greater than the upper limit of human hearing.
Web-based GUI	A Graphic User Interface run inside the web browser, it is connected to a web server and uses the services of that web server
Wi-Fi	Wireless local networking technology based on 802.11 standards
Wireshark	A free packet sniffer computer application for network analysis

1 Introduction

PRA Alpha Inc. was formed by Rick Wong, Feng Ye and Jessica Sun at the beginning of 2008 in Simon Fraser University to develop a remotely controlled home care robot, iBonni, in the purpose of offering a solution for handicaps or people who is away from home to monitor their house and control their home devices.

Through six months' struggling and many unsleeping nights, this idea finally comes to reality. We successfully implement the robot which has the functionalities listed in our functional specification [2].

This document reviews our design and develops, introduces the whole system and the implantations of our device. It also includes the encountered problems, actual budget and timeline, and personal reflections of each team member.

2 Current State of the Device

2.1 System Overview

As described in our project proposal [1], functional specification [2] and design specification [3], iBonni, can take care people's home when they are away for vacation. Users can control their home devices (TV, lights, pet feeder and potentially windows and doors) via the robot remotely; the robot reports the status of users' houses (temperature, blocks, gas leaking etc) and provides real-time videos to the remote users upon requests. The user interface is a Java based GUI that can be started on any remote computer (or PDA and Smart phones) by visiting the build-in server of the robot.

The below overall system block diagram illustrates the structure of the system, the modules and the communication protocols between them.

The whole system is divided into four main layers: remote, pico, on board and off board. The remote layer includes a Web GUI which can be started on any Java supported computers, potentially PDAs and Smart Phones. Users interface this layer with their favorite web browser. The remote layer sends command to and receives feedbacks from

the pico layer via internet. The pico layer includes a camera and the Pico-ITX embedded computer that mounted on the robot as the server. This layer interfaces with on board layer via UART. The on board layer includes an I²C BUS and the critical microcontroller modules such as CMM, SCM, UDM and ACM. This layer interacts with the lowest layer, off board, via several of protocols as shown in below Figure 1. The off board layer is simply formed by peripherals including motors and many kinds of sensors.

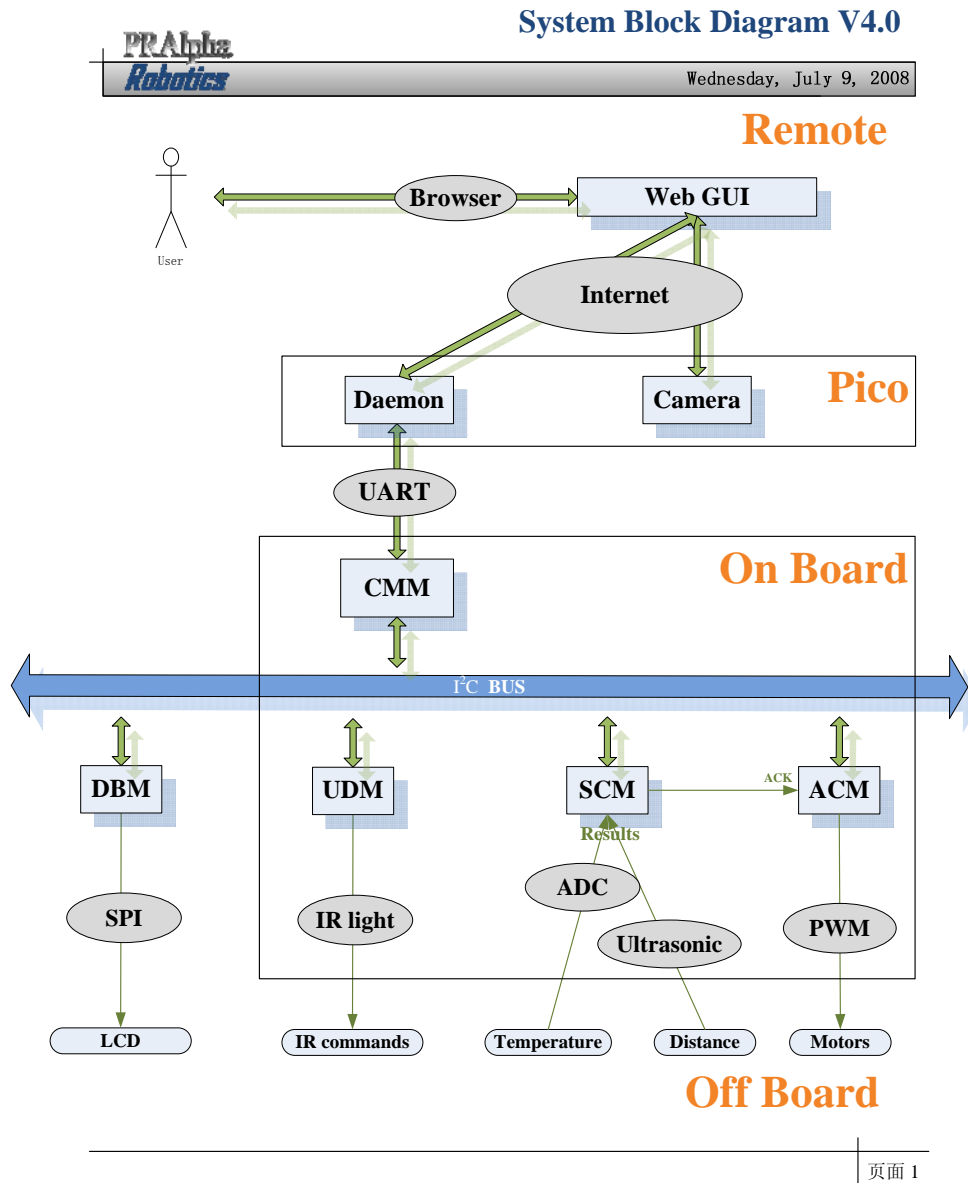


Figure 1: Overall System Block Diagram

2.2 Chassis

Figure 2 illustrates the hardware layers of the system. The battery packs for motors and computer are mounted in the bottom of the chassis. Layer 1 includes servo and DC motors, PCBs and on board modules. Layer 2 includes camera, IR LEDs and sensors in the front; Pico-ITX and power supply in the middle; USB hub, flash driver and wireless adapter in the rear.

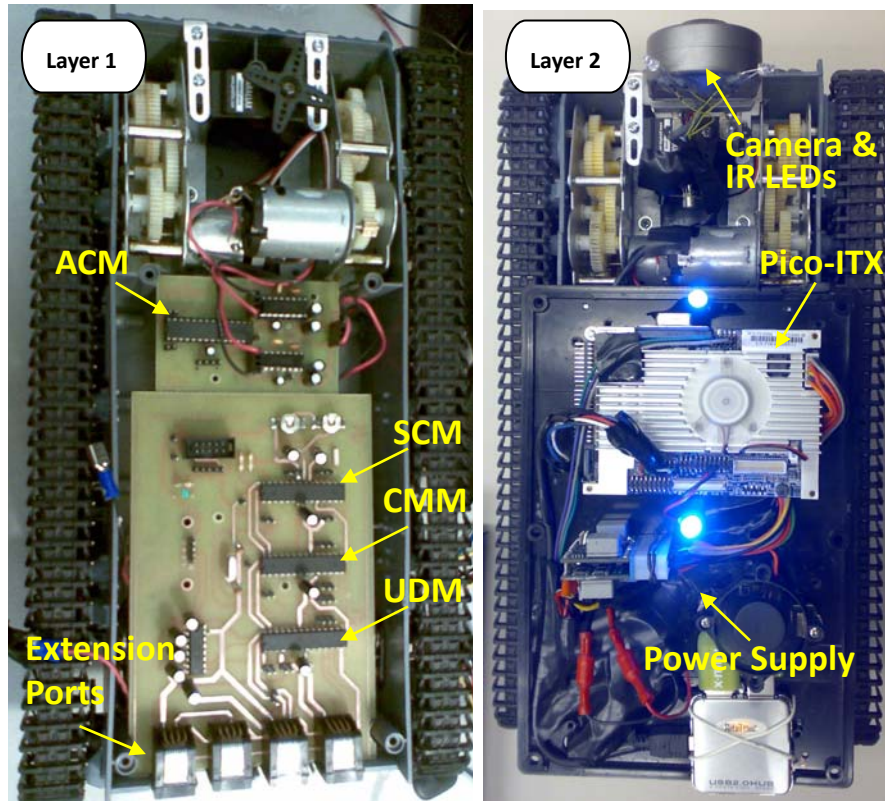


Figure 2: Hardware Layers of the System

2.3 CMM

The task manager, Center Manager Module, is the interface between the pico-ITX and other modules attached to the I²C BUS. It listens to both the USART port and I²C BUS, responds to services interrupts generated by either user or other modules. It also has the capability of resetting sub-modules and itself upon request.

Considering extensibility, we developed a DTAC protocol to support the plug-in-play function. Using this protocol, system recognizes a new device and assigns a TWI

address to it when the system is hot. The behaviors of the four extension ports shown in Figure 2 are pretty much like USB ports in PC. Besides the data BUS, the extension ports also provide standard 5V power and have a reset request pin which can be used to reset the attached devices.

2.4 SCM

In the current states, SCM, the Sensor Control Module, of the device is able to periodically collect data from temperature and distance sensors, and then report them to CMM. The camera and distance sensor are mounted on a servo that can be rotated upon user control in the range of -90 to 90 degrees.

2.5 ACM

The task of ACM is simple; it receives commands from the I²C BUS, decodes the commands, and controls motors. Users have freedom of defining the available motions of the robot such as moving forward, backward, turning or spinning left and right with three levels of speed.

2.6 DBM

In order to debug our system easier, we have designed the DBM, a Debug Module, which collects error messages with extra debug information from the system and displays them on LCD. This module is a standard alone device that can be attached to or dispatched from the robot any time using the previously mentioned DTAC protocol. Considering commercializing the robot, this device can be a very useful portable debug tool for technicians. Figure 3 shows how the error information is displayed on LCD.



Figure 3: Displaying error message in Debug Module

2.7 UDM

The User Defined Module, UDM, gives user the ability of controlling home devices which supports IR protocol. Acting like a universal IR remote controller, UDM translates the commands generated by users and transmits them via Infrared protocol. UDM uses the 32 bits of flash memory space to store the pre-defined IR control codes. We have mounted two IR LEDs at front of the robot to broadcast a series of IR beam bursts.

2.8 Pico

We called the embedded computer system on the robot Pico because it runs on the Pico-ITX (10cm x 7.2cm) motherboard from VIA. The embedded computer has a VIA C7 CPU (Esther) running at 1.0 GHZ. It is equipped with 1 GB DDR II RAM, a USB wireless network adapter, a USB web cam and a 2.0GB high speed USB flash drive as the storage of the system. This Pico is mounted on the robot and able to access the Internet wirelessly.

Gentoo Linux OS with a web server, a video streaming server and our control daemon program is installed on the embedded computer. The system is accessible by SSH for command line interface or by NX client for graphical interface. Therefore, local user input and output peripherals are actually not necessary.

The HTTP web server provides a web start service for user to login and start the GUI client. The video streaming server converts the real time video from camera into a stream of flash video. The control daemon program is responsible for gathering commands from the GUI and sending them out to the serial port, at the same time it also gather data from the robot hardware and transfer it to the GUI.

2.9 GUI Client

The GUI client is implemented in Java, so it is considered platform independent and can be executed on any computer that has Internet access and Java virtual machine installed. It displays the real time video from the embedded computer streaming server. Thanks to flash video format, the video delay is quite small. The GUI has a set of buttons to drive the robot forwards, backwards or lefts and rights with three levels of speeds. User can subscribe to monitor some sensors on the robot and read the data from the GUI. These data is updated about 10 times per second. In addition, there are 12 user defined buttons for the user to control their home devices.

The figure below is the GUI window.

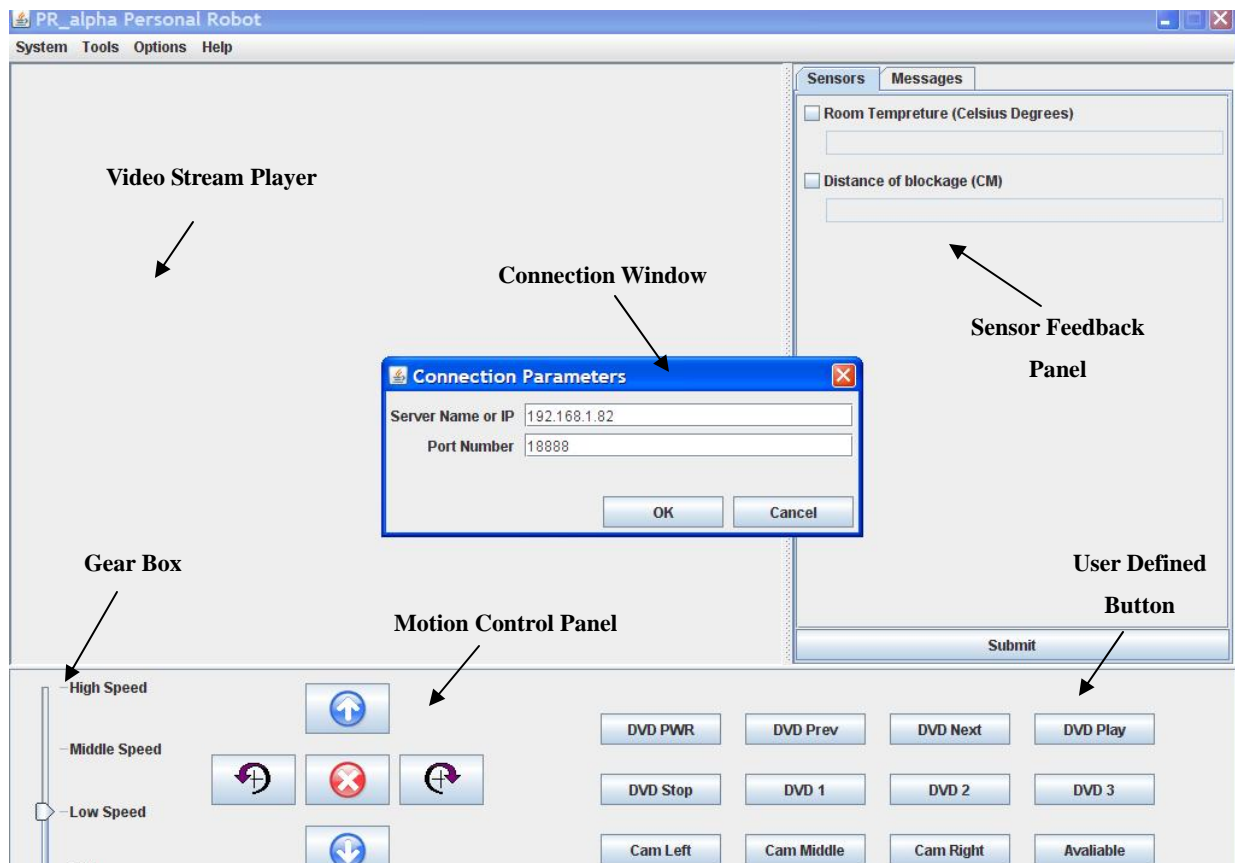


Figure 4: User Interface

3 Device Deviations

3.1 Overall System

In this project, we have achieved what we planned. The whole system matches most of the functional [2] and design specifications [3]. Due to time constraints, the following specified functionalities are not implemented:

“**[PR19-B]** Robot must send warning message whenever attention is needed, for instant, the power is low.” The device was only able to send warning message to user when the distance of the block is shorter than 20 centimeters. A new functionality we added was the robot was able to automatically stop when reached the minimum block distance.

“**[PR30-B]** The emergency response time refers to the time delay between the detection of emergency event by the sensor system to proper response. A proper action shall be generated within 100 microseconds.” Our actual response time is more than 100 microseconds because we send the emergency request from GUI. Therefore the Internet delay should be considered. This will be improved in a near future.

3.2 Sensor System

In the proposal, the sensor system should detect the room temperature, blocks, gas and water leaking and monitor the house situation. However, for the demonstration purpose, we only implement the functionalities of monitoring and reporting temperature and block distance; and we did not use a night vision camera for cost considerations.

3.3 Motion System

The motion system met the specified functionalities listed in our functional specification [2]. The design deviation is that we replaced the stepper motor with DC motor because we could not found a matched gear.

3.4 Computer System

The mainboard was changed from Nano-ITX to Pico-ITX which is smaller, lighter and cheaper with the similar functionalities. The hard drive also replaced by USB drive to reduce the cost and increase the system stability. The power consumption is about half of before these two changes are made.

3.5 User Interface

The only unimplemented functionality of our device is “[PR63-B] Sound from the vision system shall also be transmitted and played back.” In order to easily controlling the robot, we are not going to import audio stream until the video delay is further improved.

3.6 Power System

The power system fits all the expected functional specifications [2]. In design part, we use Ni-NH battery instead of Li-ion battery to provide a safer solution.

4 Encountered Problems, Solutions and Limitations

During the development, integration and testing phases, we have encountered uncountable problems. We lists some important ones which either changed our design or taught us lessons below.

4.1 IR Control

- a) Issue: We spent 16 hours in the lab and finished coding for UDM to send IR control signals. The code was tested on the same day and uploaded into our version control system. After one week, we were going to integrate the UDM with other modules but unable to control DVD player with the UDM anymore.
- b) Solution: It ended up that none of our code or UDM had any problems, the IR LED was burned; however, human eyes cannot see IR lights (and we did not know that our camera can see it). In addition, we did not even realize the IR LED could be burned in normal operations until wasted much time on it.
- c) Experience: We cannot always see the truth, physically and mentally.

4.2 I2C BUS

- a) Issue: Our code for I2C BUS worked perfectly in ISIS simulation but no data could be transmitted on actual hardware. Since we have to implement both the transmitter and the receiver, we could not tell which caused problems.
- b) Solution: Patrick taught us how to debug the BUS using digital analyzer and we finally could analyze the transmitter and the receiver separately. After debugging, we solved this problem which was caused by timing issue.

-
- c) Experience: There is a huge gap between simulation and actual hardware.

4.3 DBM

- a) Issue: We designed a debug module on breadboard which displays error information received from other sub-modules on the I2C BUS. Unfortunately, our LCD did not display anything with the code. Then, we spent our code to the manufactory of our LCD and it magically worked on their site.
- b) Solution: After discussing with the manufactory, we found that the issue was caused by the unstable connections of the components in breadboard. Then we had decided to make our own PCBs and the problem was solved.
- c) Experience: We started to make our own PCBs for all the circuits. It might take some time to design and fabricate, but PCB has many advantages such as stable connection, small size and easy for mass production.

4.4 USB Flash Drive

- a) Issue: The USB flash drive behaves as a hard drive with lower writing speed. Since we used the USB flash drive as the storage for the embedded computer, the program encountered some strange behaviors and crashed irregularly from time to time
- b) Solution: We setup a similar environment on another workstation, compiled programs on it and then built the flash drive synchronously. In order to speed up the build process, we even used multiple workstations to share the compiling jobs. Finally, we found out that this issue is caused by the compiling workstations had not have X86 i686 class processors.
- c) Limitation: The compiling workstations must have X86 i686 class processors.
- d) Experience: System compatibility is quite important for hardware design.

4.5 Video Delay

- a) Issue: The video streaming server and the video player in the GUI did not seem to be synchronized very well. After about 20 seconds of video playback, the delay is relatively long and it was getting longer and longer.
- b) Solution: We noticed that this was due to the small frame rate on the player side, so we setup a timer on the player in every 10 seconds for the player to skip 10 frames. That means that the player would catch up with the streaming server and playback the most latest video frames in every 10 seconds.

- c) Limitation: It works fine right now but we need more experiments to figure out how many frames to skip can achieve better video quality.
- d) Experience: Problems sometimes cause by minutia.

5 What We Would Do Differently and Future Improvements

5.1 What We Would Do Differently

There are some drawbacks of the device that we do not have time and chance to correct them in this project. If we are going to build a similar device, we will do something differently.

First of all is the pico-ITX computer. Around half of the unit cost is spent on the computer. We found out later that there is a new technology can directly communicate the microcontroller and the network protocol. So we are going to implement is technology to reduce the product cost. Secondly, the chassis is not suitable for the battery pack since we changed our power supply solution. If we will do the project again, we will change the chassis to fit the currently used battery pack. Thirdly, the project box we use right now is made in plastic which is not good for heat radiation. We are going to use a metal project box to achieve better heat disperse quality.

5.2 Future Improvements

iBonni has great potential for future research and improvement. During the development and final testing phase, we summarized the main issues which can be solved in the future as below:

5.2.1 IR commands self-learning

Currently, the robot only can send pre-defined IR command or otherwise the users have to manual input the IR codec of their devices. This process is inefficient and not easy for most of the users. It can be solved by implementing IR record and playback functions. Users could easily set the robot into learning mode and then press the buttons on the remote controller that comes with their home devices. The robot will detect the received codec and store it into an EPROM for later use.

5.2.2 Step motor with feedback

Our original motion system design was using step motors because they have both high

accuracy and enough torque. However, lacking the gears that fit both existing step motors and tank base belt, we have to use DC motors instead. The main disadvantage of DC motors is that their speed much depends on the supply current. When using a battery, it is extremely difficult to maintain a constant and linear speed. If we can find the gears that match our robot, we can go back and use step motors with feedbacks, which will enable the accuracy control of iBonni's speed.

5.2.3 Smaller IC packaging

For IC packaging, we choose DIP whenever it is possible. The advantage of DIP package is that it can be easily plugged into breadboard or soldered into both prototyping board and PCB. However, DIP package occupies much bigger space comparing to other packages such as QFP.

5.2.4 Microcontroller wireless controlling

The sub-modules of iBonni communicate with each other via TWI BUS; the CMM communicate with PC via UART; the PC then communicates with the remote users via Internet. This design limits the bottom line of our cost from the beginning. We should have design the system in such a way that microcontroller communicates with user via Zigbee (a low cost protocol that enable the wireless ability of microcontrollers) directly. By removing PC, we will cut approximately half of the cost while saving at least 70% of the battery power. This solution will be researched in the following four months.

5.2.5 Self positioning

After employing Zigbee, the robot will be able to locate itself in the room by communicating with the preset positioning spots. Because the cost of employing Zigbee is relatively low (\$3 for each chip), the solution will be affordable for standard families. Similar positioning project has been researched by many companies such as Atmel. In addition, the robot will be able to find a relatively optimized path to its destination by using pre-inputted room map.

5.2.6 Charging base

With the abilities of self positioning, we could then design an auto charging base that can charge the robot whenever the battery is low. The robot will be able to locate itself and the charging base, drive itself to the charging base and starts to charge when it detects low battery situations. Archiving this, iBonni will finally be a fully functional robot that takes care of your home unpiloted.

5.2.7 GUI

The GUI is started from the web, so all configurations should be loaded from the server and then stored back to the server. And because now we are using a trial version of a commercial flash player, it only works on Windows, developing a native Java flash player would make the GUI really cross-platform.

6 Budget and Timeline

6.1 Budget

The following table compares the estimated cost and the actual development cost of our project incurred to demo date.

Items	Estimated Cost	Actual Cost	Revenues
PC	\$646.06	\$1078.53	ESSEF: \$680 School of Engineering Science: \$50 per group
Chips	\$90.90	\$143.97	
Sensors	\$250.00	\$152.33	
Chassis & Motor	\$113.22	\$92.48	
Power Supply	\$200.00	\$374.02	
PCB	--	\$152.86	
Development Fee	--	\$244.97	
Miscellaneous	\$50.00	\$316.99	
Shipping & Handling Fee	--	\$129.74	
Unpredictable	% 15	--	
Total	\$1553.86	\$2685.71	

Table 1: Estimated and Actual Cost Comparison

The actual cost in the above table includes the items we broke, the items we brought for backup and the items we didn't use on the final device. Through comparing the estimated cost and the actual cost, we could find that the development cost of the project

was under-estimated by 57%. Many actual development costs of the items are much higher than their estimated costs because we were keeping doing research and seeking for better solutions during the whole product development phase. Therefore, although some of the actual cost increased, the unit cost is reduced. The following table lists the final unit cost of our product.

Category	Items	Sub	Category Cost
PC	Pico-ITX Mainboard	\$258.72	\$340.44
	RAM, USB drive, Wireless adapter, etc	\$81.72	
Sensors	Ultrasonic Sensor	\$41.84	\$94.11
	Temperature Sensor	\$1.88	
	USB Camera	\$50.39	
Chassis	Tank base, Project box & Servo motor, etc	\$73.88	\$73.88
Chips	Microcontroller, Motor driver, UART chip, etc	\$26.71	\$26.71
PCB	Pre-sensitized PCB, Components, etc	\$45.80	\$45.80
Power Supply	Eneloop Batteries	\$50.80	\$115.74
	Battery Charger	\$39.95	
	PC Power Supply	\$24.99	
Total		\$696.6	\$696.69

Table 2: Unit Cost

The table above shows that the unit cost of our product is \$696.69. Looking into the category cost column, the cost of PC is around half of the total product unit cost, so the PC is the most expensive part of our product.

6.2 Timeline

The following figure presents the Gantt chart with estimated timeline in shadow and actual timeline in blue (for the first deadline) and green (for the second deadline) of our project.

As shown in Figure 5, the progress of our project did not meet the expected schedule

because we could not find the right gears that fit the step motors. After spending a few weeks of searching, we eventually decided to switch to DC motor. Due to this, we had to make research on DC motors and rewrite the programs of motion module. Having the relatively complicated project, we only have three man powers and were unable to catch up with the original schedule. Therefore, we requested the first extension at the beginning of April. Unfortunately, we had to postpone the project on May 20th one more time because the hardware issues we experienced a week before demo.

Besides finishing our original design, we have additionally improved the functionalities and eliminated the potential issues we found during the extension. During the first extension, we added debug module and improved the power solution. During the second extension, we used USB instead the hard drive, found a better charger solution and improved video quality.

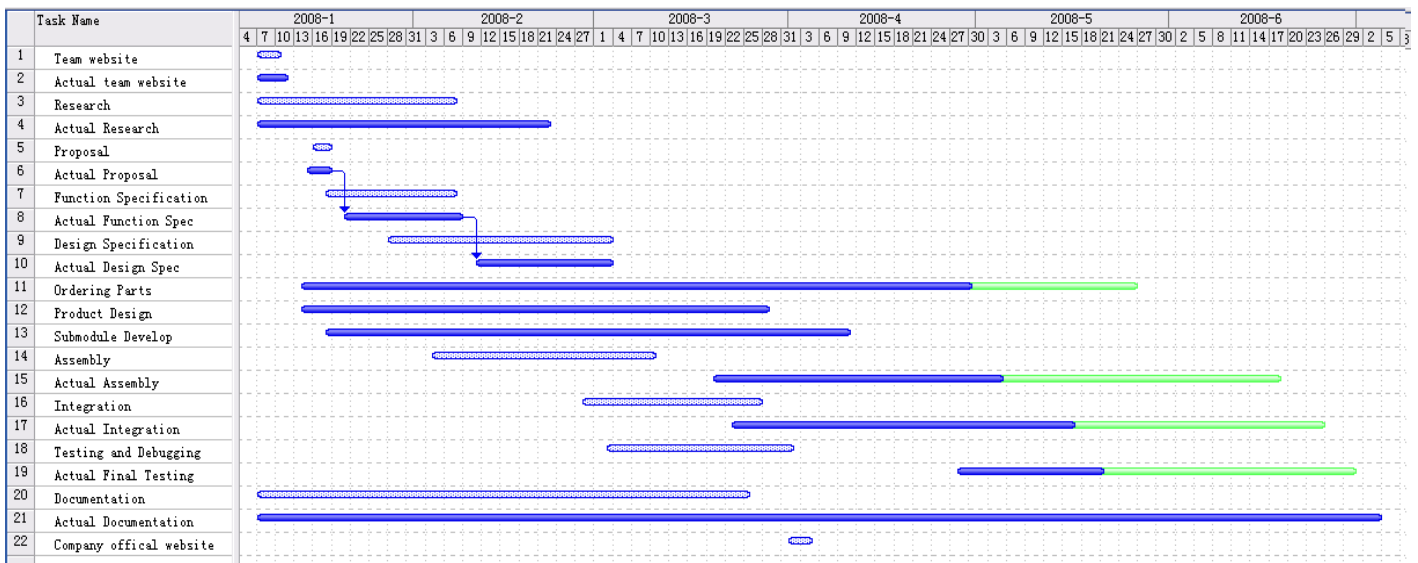


Figure 5: Project Timeline

7 Inter-Personal and Technical Experiences

7.1 Rick Wong

As a team lead, I am very proud of having the best team in the class. With the minimum number of the manpower, we have accomplished a complicated project. Each one of us is both great team player and able to work individually. I believe that maintain an excellent team dynamic is our key to success. Besides communicating in our team page and having weekly meetings, we always discuss the encountered problems and share the experiences during our team dinners.

By answering other's questions and providing advices, I have gained much knowledge from my team members. Feng is a computing science student who has strong background in both operating system and programming. Working together with Feng, I have significantly improved my coding skills and learned many useful skills about how to handle the low level layer of an operating system. For example, in terms of Linux, I have never used Gentoo before this project. The main difference between Gentoo and other Linux distributions is that Gentoo is highly customized and does not come with prebuilt packages. It took us more than a day just to install and compile the operating system on the robot, but as a pay back, we got a highly embedded operating system which is specified for our robotic applications. Walking though with Feng during the setup of the OS platform is a cheerful journey and solving the various issues about the Pico-ITX is an unforgettable experience. Jessica is a fellow engineering student who has knowledgeable engineering background and patience to details. As the only female in our group, she can always give us alternative suggestions and hints to the solutions. Working together on design and fabricating PCB, modifying the chassis and coding the microcontrollers, I was amazed by her professioness. She has assisted me among my researching, coding and debugging, and I really feel thankful for her effort which saved me a lot time. I also thank her for being a challenger who has corrected me from going the wrong directions more than once. I consider this project as an unrepeatable team-work experience that enhanced my leadership skills.

Mentioning hardware experiences, I have attained many handy skills that I could never learn from lectures. In order to modify the robot chassis, I have learned how to use blade saw and drill. In order to connect components from board to board, I have learned how to make special cables, sockets and headers. In order to make a reliable PCB, I have learned how to cut, expose, develop and etch. And special thanks to Jamal, one of

my best engineering friends who have taught me amazing soldering skills on double sided PCBs. I would like to take this chance and share some experiences on making PCBs. Most of the PCB software is expected to be used to design manufactory PCB instead of the ones that we make by our own hands. Therefore, the default values are not recommended. For example, the default trace width and distance are 0.3mm, but if we use this default value, part of the trace will be washed away during developing stage. The minimum trace widths we used are 0.5mm for standard trace and 1.5mm for power and critical path. For cutting a pre-sensitized PCB, I would suggest one to employ two layers of plywood holding the PCB in between and feed this “sandwich” to the PCB cutter. In this way, the damage caused to sensitized layer during cutting will be minimized. In addition, using a shape blade to score the boarder will also help a lot.

In terms of software experience, I have learned many useful tools that can be used in my future engineering career. I used ICCAVR and WinAVR for coding the microcontrollers. Personally I prefer the former one simply because the interface is friendlier for me; in fact, it is very close to vim, the editor I used the most in Linux. To simulate my code, I have to use both AVR Studio and Proteus ISIS. AVR Studio, the official simulator released by Atmel, is an excellent tool to debug the AVR microcontrollers. However, it can only simulate the internal environment of the microcontroller but not the external communications among all the modules, therefore, I use ISIS from Proteus to do this job.

Last but not the least, I would thank Patrick who gave us help and advices where they are needed; thank Steve who taught us the importance of maintain a fine team dynamic; and thank Brad and Jason who gave us useful suggestions and put the effort on marking our documents.

7.2 Jessica Sun

Working on this project is the most interesting and worthwhile experience I have ever had in my undergraduate study at Simon Fraser University. Working on an interested topic with wonderful teammates sometimes made me even forget it was a course and hope it would never end.

Besides mainly responsible for SCM and ACM design, I was also well involved in components ordering, PCB design and fabrication, circuit design and soldering, assembling, testing and documentation.

In terms of technical experience, I gained various new skills on both software and

hardware. In software part, I learned to use Protus, AVR Studio and ICC AVR for microcontroller programming and debugging, Altium Designer for PCB design, Office Project and Visio for documentation. In hardware part, I learned to use real time embedded system, sensors, motors, PCB, PC, power supply, remote controller and wireless solution. I am more familiar with the principles of how these systems work and how to control these systems..

As expected, we encountered numerous problems during all the project development phases. I would like to share many precious experiences here. A tip for researching is finding some relatively professional and comprehensive websites or BBS could save much of your seeking hours. Usually, you can find almost everything you need and all the encountered problems on a well developed website and BBS. Some strategies for design include, first of all, always trying to think as many corner cases as possible and design for the recoverability. In our case, almost all the possible problems we thought happened later and considering ahead helped us a lot. Unexpected problems usually cost much more effort to recover. Secondly, do not forget the extension ability and reliability of the device. Installing a switch on important connections, the power supply wire for example, might save our device in emergency. There are also some hints for ordering. Sometimes the cost of ordering components from abroad like China and Tailand within shipping fee is even cheaper than the cost of ordering locally. Delivery the order from US to Point Roberts, a US small town which is 20 minutes drive from Richmond, and drive to pick it up sometimes will be able to save some money. An advice for fabrication is the more you pay attention in each produce steps, the more reliable your device will be. A lesson I learned for testing is a full-scale and practical test plan is important. My suggestion is always test from small to large, from bottom to top, from inside to outside and please do remember to keep your old working version codes.

In the end, I would like to thank my teammates. I appreciate it a lot about their accomplishment, hardworking and consideration. It is my fortune to be their project teammate and friend. I will never forget how they comforted me when I was frustrated by burning component. I will never forget how we worked tirelessly together overnight to meet the deadline. I will never forget how we share our cheerfulness with each other when the device was working properly. At last, I wish we have chance to work together again.

7.3 Ye Feng

After more than 6 months of hardworking on our Home Robot project with our small team, I have experienced the whole process of how to develop an engineering project from researching, designing, implementing to documenting. As a software engineer, I got a chance to write programs that are closely working with hardware, and I also learned how to program, test and debug microcontrollers.

I had a very good experience of an excellent team work, although we had many issues during the development, we could help each other to overcome all of them.

First of all, I think we did a right thing at the beginning of the project. We discussed carefully and divided the whole system into 3 layers, and modules are also well defined within layers. In this way, every one of us could work independently. Developing and testing of each module could be done without other modules, and then integrate them all together. It proves that clean modulation is the first step of a successful implementation.

Although each of us was assigned to be responsible to some modules, we could still help each other. We often shared what we find interesting during the development and also shared the problems we met. Many times, Rick and Jessica could give me very useful ideas on the software part. The scenario is often like this: I just explained my problem to them and asked them to look at my code, for most of the cases, they just asked me why I did this, or why didn't I do that. Usually, those questions magically guided me to a new idea of a solution. And I also believe that my "stupid" questions also helped them more or less. Communication is the most important in team work, team members need to understand and be able to cover areas of other team members.

Although I was not responsible for the hardware, I also learned a lot on microcontrollers and sensors. Thanks to Rick and Jessica, they explained many details to me, and showed me how to program the microcontroller and how to run and debug the program on a simulator and then write the program to the chip through the programmer. Finally, I found my own favorite tools to develop programs for microcontroller under Linux with GCC. As a fact, now all of our team members know, there are different libraries and macros for different compiler. The C code of microcontroller, at least for AVR family is not portable among different compilers, or different developing environments. So it is important that the whole team would use the same developing environment for developing programs on microcontrollers. I personally prefer avr-gcc for AVR microcontrollers because it is closer to ANSI C.

One lesson I have learned is that the interface between hardware and software needs more attention. When I developed the control program which communicates with the microcontroller through UART, at the time when the hardware is not ready for testing, I tested my IO commands with a NULL modem cable connecting to another computer, and it worked fine, but when it is connected to the hardware, it could not send out anything. Finally, we found problem, I enabled hardware flow control as usual, but when it is connected to the microcontroller, only 3 pins were used, no hardware flow control signal at all. There were many other times, when we tested on a simulator, it worked fine, but when we integrate everything together, it stopped working. Most of the cases were just because the hardware's default setting did not match the expected settings of the software, the solution is simple, checking the manual carefully. And as a programming tip, those configuration variables should be easily changed in the program, better define them in the macro.

One successful experience is that we used a version control system to store and manage our code at the early phase of our project. It is easy for everybody in the team to share the code, and to work on the same piece of code simultaneously. It is also useful when the new changes break something, and you can easily rewind back to any old versions. After this project, I think none of our team members would doubt the importance of regular backup. All of us had a need of a backup copy several times during the development of this project. Take one minute to do a backup, and it might save work of weeks.

8 Team Dynamics

This project is very valuable for all of our team members. We gain much knowledge and experience on both of software and hardware. Since we are a small group doing complex project, everyone has to play almost all the roles involved in the whole product developing cycle like research, design, testing, documentation, etc, therefore what we practiced and learned is comprehensive.

Our project was divided into portions depending on every member's strongpoint. Rick is mainly responsible for communication solution and motor, Jessica is for the sensor part and Feng is for the PC part. We discussed and helped each other frequently. Each of the members in our team agrees that everyone in our team is irreplaceable.

We all involved with the prototype testing, PCB making and final assembly where we

learned schematics, prototyping, PCB design, fabrication, soldering and testing skills. We learned and implemented various functions of the microcontroller to achieve different functionalities. We also built our small PC and created the communication between demon and GUI by C and Java.

We wrote all the required documentation together. Each member wrote a specific portion that fell under his expertise and one of the members was responsible for a final proofreading and formatting. In this aspect, we all gained experience in writing documents for developing a product from its conceptualization to its prototype stage.

Our team dynamic skills are remarkable. In order to better communicating between team members, besides every week's meeting, we build our own BBS, pralpha.info, to share research resources and ideas, report progress and discuss encountered problems. To better organize our code, we use a CVS version control system, RapidSVN, to manager our code version.

Most importantly, through making many various decisions in each development phase, we improved our team work skills like how to listen and respect other's ideas, how to share your own opinions, how to skillfully persuade your teammates and how to encourage each other when facing difficult situation. All of our team members agree to stay in our team to improve our product after finish ENSC 305/440.

9 Conclusion

Through doing this project, we were involved in a whole product development process and successfully developed a practical product. We all agree that the Capstone Project Course is the most useful courses in our undergraduate engineering study. We have an opportunity to implement the knowledge we learned on textbooks and design an interested product.

We plan to keep improving our product and would like to commercialize it in the future.

Our recommendation to other groups is like Samuel Smiles said, practical wisdom is only to be learned in the school of experience. Our suggestion to school is one semester is not enough for this project course and one project course is not enough for the undergraduate engineering study.

References

- [1]. **PRAAlpha, Inc.** Proposal for the Home Care Robot. Burnaby : Simon Fraser University, January, 2008.
- [2]. **PRAAlpha, Inc.** Functional Specification for the Home Care Robot. Burnaby : Simon Fraser University, February, 2008.
- [3]. **PRAAlpha, Inc.** Design Specification for the Home Care Robot. Burnaby : Simon Fraser University, March, 2008.