# AUTONOMOUS ROBOTIC WHEELCHAIR

# WITH COLLISION-AVOIDANCE NAVIGATION

A Thesis

by

PIN-CHUN HSIEH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2008

Major Subject: Mechanical Engineering

# AUTONOMOUS ROBOTIC WHEELCHAIR

# WITH COLLISION-AVOIDANCE NAVIGATION

A Thesis

by

PIN-CHUN HSIEH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

| | |
|---|---|
| Chair of Committee, | Won-jong Kim |
| Committee Members, | Chii-Der Suh |
| | Yoonsuck Choe |
| Head of Department, | Dennis O'Neal |

August 2008

Major Subject: Mechanical Engineering

# ABSTRACT

Autonomous Robotic Wheelchair with Collision-Avoidance Navigation.

(August 2008)

Pin-Chun Hsieh, B.A., National Tsing-Hua University

Chair of Advisory Committee: Won-jong Kim

The objective of this research is to demonstrate a robotic wheelchair moving in an unknown environment with collision-avoidance navigation. A real-time path-planning algorithm was implemented by detecting the range to obstacles and by tracking specific light sources used as beacons. Infrared sensors were used for range sensing, and light-sensitive resistors were used to track the lights.

To optimize the motion trajectory, it was necessary to modify the original motor controllers of the electrical wheelchair so that it could turn in a minimum turning radius of 28.75 cm around its middle point of axle. Then, with these kinematics, the real-time path planning algorithm of the robotic wheelchair is simplified. In combination with the newly developed wireless Internet-connection capability, the robotic wheelchair will be able to navigate in an unknown environment.

The experimental results presented in this thesis include the performance of the control system, the motion trajectory of the two driving wheels turning in a minimum radius, and

the motion trajectory of the real-time path-planning in a real-life testing environment. These experimental results verified that the robotic wheelchair could move successfully in an unknown environment with collision-avoidance navigation.

# ACKNOWLEDGMENTS

I would like to take this opportunity to express my sincere gratitude to my advisor, Dr. Won-jong Kim. I would also like to thank him for the invaluable time and guidance that I received from him throughout this thesis research.

# TABLE OF CONTENTS

Page

ABSTRACT……... iii

ACKNOWLEDGMENTS…….. v

TABLE OF CONTENTS…….….. vi

LIST OF FIGURES……..….. ix

LIST OF TABLES…….…. xv

CHAPTER

I INTRODUCTION…… 1

1.1 History…….……. 1
1.2 Objective…….. 2
1.3 Contributions…… 2
1.4 Thesis Organization……. 3

II LITERATURE REVIEW…….. 5

2.1 Modeling…….. 5
2.2 Sensor Implementation…… 6
2.3 Path Planning and Obstacle Avoidance……... 6

III ROBOTIC WHEELCHAIR DESIGN…….. 9

3.1 Step I – Main System of the Robotic Wheelchair……... 10
3.1.1 Wheelchair…….. 10
3.1.2 Laptop……..... 10
3.1.3 Data-Acquisition Card……..... 10
3.1.4 Interface Board……... 11
3.2 Step II – The Sensor System…… 12
3.2.1 Light-Sensitive Resister…….. 12
3.2.2 Distance-Measuring Sensor…… 12
3.2.3 Hall-Effect Sensors……..... 13
3.2.4 Interface Board on the Sensor System…… 14
3.2.5 Operational Amplifiers……... 14
3.2.6 Priority Encoders…… 15
3.2.7 Voltage Regulator……... 15
3.3 Stage III –Wireless Internet …… 15

CHAPTER Page

IV INTERFACING…… 16

4.1 The PCMDIO Data-Acquisition Card……..... 16
4.2 Interfacing the Motor Controllers…… 17
4.2.1 Speeding Control…… 18
4.2.2 Forward and Backward Control…….. 18
4.2.3 Movement Measurement…… 20
4.3 Interfacing the Sensor System…….…… 22
4.3.1 Interfacing the Light-Sensitive Resistors…… 22
4.3.2 Interfacing the Distance-Measuring Sensor…… 25

V KINEMATICS AND PATH PLANNING……. 26

5.1 Dynamics of the Wheelchair…… 26
5.2 Kinematics of the Wheelchair…….. 27
5.3 Algorithm of the Real-Time Path Planning Guided by Infrared Sensors……... 28
5.4 Light Tracking……..... 47

VI SOFTWARE DESIGN……... 49

6.1 Programming Language……... 49
6.1.1 PCMDRIVE Configuration Utility……..... 50
6.1.2 Performing Data Acquisition…….. 51
6.2 Hardware Control……..... 52
6.3 Operation of the Robotic Wheelchair..……..... 53
6.4 Remote Control...……..... 56

VII CONTROL SYSTEM DESIGN……..... 59

7.1 The Structure of the Control System…… 59
7.2 The Sensor in the Control System…… 60
7.3 Controlling the Wheel Speed……... 62

VIII OPERATION AND TESTING……... 68

8.1 Operation…….. 68
8.1.1 Autonomous Mode…….. 68
8.1.2 Manual Mode…….. 69
8.2 Experiments and Testing…….. 69
8.2.1 Recording the Motion Trajectory…… 69
8.2.2 Robotic Wheelchair Rotating around the Axle Middle Point……... 71

CHAPTER Page

8.2.3 Motion Trajectory of the Robotic Wheelchair in an Unknown Environment……... 73

IX CONCLUSIONS AND SUGGESTED FUTURE WORK…… 80

9.1 Conclusions…….. 80
9.2 Limitations……..……. 81
9.3 Suggested Future Work……..……..... 82

REFERENCES……..... 83

APPENDIX A OPERATING PROGRAM……..…….. 85

APPENDIX B CLIENT SIDE PROGRAM……..… 116

VITA……..….... 119

# LIST OF FIGURES


| FIGURE | | Page |
|---|---|---|
| 3.1 | Development of the robotic wheelchair. The sensor system is mounted at the front, and the laptop for controlling the robotic wheelchair is on the top……..... | 9 |
| 3.2 | Interface board and two motor controllers. Four relays and CD4066 chips are connected to the MC-7. The interface board connected to the PCMDIO card with the connectors that were modified from IPRV and PMLR……... | 11 |
| 3.3 | Seven light-sensitive resistors and five distance-measuring sensors are mounted on the sensor bracket with interface circuit board…….. | 13 |
| 3.4 | Interface board between the sensor bracket and the PCMDIO. The chips from top to bottom are the voltage regulators, operational-amplifiers, and priority encoder……..... | 14 |
| 4.1 | MC-7 motor controller and the interface board…… | 17 |
| 4.2 | Dervise MC-7 motor controller……..... | 19 |
| 4.3 | Circuit for speeding, forward, and backward control…… | 20 |
| 4.4 | Interface board that contains four relays, two CD4066 chips, one Darlington-array chip, and two 74HC191 counters…….. | 21 |
| 4.5 | Interface board for the sensor system…… | 22 |
| 4.6 | Voltage comparator…… | 23 |
| 4.7 | Circuit between photocells and PCMDIO……..... | 24 |
| 5.1 | The robotic wheelchair turns in an original point and the detecting range of five infrared sensors…… | 28 |
| 5.2 | The first condition. The robotic wheelchair detects the obstacle to the right. It turns counter-clockwise until it detects no obstacle…….. | 31 |
| 5.3 | The second condition.The robotic wheelchair detects the obstacle to the right and in the front. It turns counter-clockwise left until it | |

FIGURE Page


| | | |
|---|---|---|
| | detects no obstacle……. | 31 |
| 5.4 | The third condition. The robotic wheelchair detects the obstacle to the right and front. It turns counter-clockwise until it detects no obstacle……..... | 32 |
| 5.5. | The fourth condition. The robotic wheelchair detects the obstacle to the right and in the front. It turns counter-clockwise until it detects no obstacle……..... | 32 |
| 5.6 | The fifth condition. The robotic wheelchair detects the obstacle to the right. It turns counter-clockwise until it detects no obstacle……..... | 33 |
| 5.7 | The sixth condition. The robotic wheelchair detects the obstacle to the right and in the front. It turns counter-clockwise until it detects no obstacle……..... | 33 |
| 5.8 | The seventh condition. The robotic wheelchair detects the obstacle to the left. It turns clockwise until it detects no obstacle……..... | 34 |
| 5.9 | The eighth condition. The robotic wheelchair detects the obstacle to the left and in the front. It turns clockwise until it detects no obstacle……..... | 34 |
| 5.10 | The ninth condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle…….. | 35 |
| 5.11 | The tenth condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops……. | 35 |
| 5.12 | The eleventh condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle……..... | 36 |
| 5.13 | The twelfth condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops……. | 36 |
| 5.14 | The thirteenth condition. The robotic wheelchair detects the | |

FIGURE Page

obstacle to the left and right. It keeps moving forward until it detects an obstacle……..…… 37

5.15 The fourteenth condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops…….. 37

5.16 The fifteenth condition. The robotic wheelchair detects the obstacle to the left. It turns clockwise until it detects no obstacle… 38

5.17 The sixteenth condition. The robotic wheelchair detects the obstacle to the left and in the front. It turns clockwise until it detects no obstacle……..... 38

5.18 The seventeenth condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle……..... 39

5.19 The eighteenth condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops…….. 39

5.20 The nineteenth condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle……. 40

5.21 The twentieth condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops…….. 40

5.22 The twenty-first condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle……..... 41

5.23 The twenty-second condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops…….. 41

5.24 The twenty-third condition. The robotic wheelchair detects the obstacle to the left. It turns clockwise until it detects no obstacle…….. 42

5.25 The twenty-fourth condition. The robotic wheelchair detects the obstacle to the left and front. It turns clockwise until it detects no

FIGURE Page

obstacle……..... 42

5.26 The twenty-fifth condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle……..... 43

5.27 The twenty-sixth condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops…….. 43

5.28 The twenty-seventh condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle……. 44

5.29 The twenty-eighth condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops…….. 44

5.30 The twenty-ninth condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle……..... 45

5.31 The thirtieth condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops……. 45

5.32 The thirty-first condition. The robotic wheelchair detects no obstacle and it keeps moving forward……... 46

5.33 The thirty-second condition. The robotic wheelchair detects the obstacle in the front. This is an unpredictable situation and it stops……... 46

5.34 Sensing directions of the photocells are indicated as dashed lines……... 47

5.35 Steps of the robotic wheelchair track a specific light…… 48

6.1 PCMDRIVE configuration utility……..... 50

6.2 Operating interface to the user in Visual Basic program……... 54

6.3 Algorithm for the “autonomous” mode of the robotic wheelchair……..... 55

FIGURE Page


| | | |
|---|---|---|
| 6.4 | Remote control through Internet……...… | 57 |
| 6.5 | Schematic of remote control through the Internet……. | 58 |
| 6.6 | Interface of the client side program……... | 58 |
| 7.1 | Structure of the control system……..... | 60 |
| 7.2 | Interfacing infrared sensors with the control system……... | 61 |
| 7.3 | Interfacing photocells with the control system……..... | 61 |
| 7.4 | Interfacing Hall-effect sensors with the control system…… | 62 |
| 7.5 | Statistic chart of the control voltage and duty-ratio…….. | 65 |
| 7.6 | An experimental path of the robotic wheelchair moving two meters…… | 67 |
| 8.1 | Recording the motion trajectory…… | 70 |
| 8.2 | Recording the motion trajectory while one of the driving wheels are moving forward and the other are moving backward at time *i*.… | 72 |
| 8.3 | Motion trajectory of the two driving wheels……..... | 72 |
| 8.4 | The robotic wheelchair moves in different basis coordinates in the *xy*-plane……. | 74 |
| 8.5 | Motion trajectory of robotic wheelchair turned clockwise for approximately$140^{\circ}$ …… | 75 |
| 8.6 | Motion trajectory of the robotic wheelchair tracking a specific light…… | 75 |
| 8.7 | Motion trajectory of the robotic wheelchair recorded by long-term-exposure photography technique…….. | 76 |
| 8.8 | Motion trajectory of the robotic wheelchair moving in a real-life testing environment……..... | 77 |
| 8.9 | The robotic wheelchair starts at point (0,840)……... | 78 |
| 8.10 | The robotic wheelchair turned at point (148,842)……..... | 79 |

FIGURE Page

8.11 The robotic wheelchair turned at point (160,806)..... 79

# LIST OF TABLES


| TABLE | | Page |
|---|---|---|
| 5.1 | Response of the robotic wheelchair for each condition……..... | 30 |
| 5.2 | Response to the first condition……..... | 31 |
| 5.3 | Response to the second condition……... ……..... | 31 |
| 5.4 | Response to the third condition……... ……..... | 32 |
| 5.5 | Response to the fourth condition……..... | 32 |
| 5.6 | Response to the fifth condition…….. ……..... | 33 |
| 5.7 | Response to the sixth condition…….. ……..... | 33 |
| 5.8 | Response to the seventh condition……..... | 34 |
| 5.9 | Response to the eighth condition……..... | 34 |
| 5.10 | Response to the ninth condition……..... | 35 |
| 5.11 | Response to the tenth condition…….....……... | 35 |
| 5.12 | Response to the eleventh condition…….....…..… | 36 |
| 5.13 | Response to the twelfth condition…….....……..….. | 36 |
| 5.14 | Response to the thirteenth condition…….....…… | 37 |
| 5.15 | Response to the fourteenth condition…….……..... | 37 |
| 5.16 | Response to the fifteenth condition…….....…….. | 38 |
| 5.17 | Response to the sixteenth condition…….....…… | 38 |
| 5.18 | Response to the seventeenth condition..... | 39 |
| 5.19 | Response to the eighteenth condition…….....……..….. | 39 |
| 5.20 | Response to the nineteenth condition…….…....…... | 40 |
| 5.21 | Response to the twentieth condition…….....…… | 40 |

TABLE Page


| | | |
|---|---|---|
| 5.22 | Response to the twenty-first condition..... | 41 |
| 5.23 | Response to the twenty-second condition……..... | 41 |
| 5.24 | Response to the twenty-third condition..... | 42 |
| 5.25 | Response to the twenty-fourth condition……... | 42 |
| 5.26 | Response to the twenty-fifth condition..... | 43 |
| 5.27 | Response to the twenty-sixth condition……..... | 43 |
| 5.28 | Response to the twenty-seventh condition……..... | 44 |
| 5.29 | Response to the twenty-eighth condition……... | 44 |
| 5.30 | Response to the twenty-ninth condition……..... | 45 |
| 5.31 | Response to the thirtieth condition……..... | 45 |
| 5.32 | Response to the thirty-first condition……..... | 46 |
| 5.33 | Response to the thirty-second condition…… | 46 |
| 6.1 | PCMDIO channel configuration…… | 51 |
| 7.1 | Duty-ratio of the PWM signals generated from two MC-7 motor controllers……... | 64 |
| 7.2 | Experimental data for the two driving wheels…….…….. | 66 |

# CHAPTER I

# INTRODUCTION

## 1.1 History

This thesis is built upon previous research in the Precision Mechatronics Lab: Intelligent Pothole Repair Vehicle (IPRV) [1] and Precision Mechatronics Lab Robot (PMLR) [2]. Both of IPRV and PMLR are modified electrical wheelchairs, using a laptop with a data-acquisition card as the controller.

In the IPRV research, an electrical wheelchair was modified to be an autonomous road repair vehicle that would be used to fill potholes. The IPRV is capable of being maneuvered remotely over a wireless local-area network (LAN). The limitation of the IPRV was that it could only move straight during the autonomous mode.

The PMLR moved in a desired path with better accuracy. It demonstrated an ability to travel around 10 m with a combination of its dead-reckoning capability and position feedback by Hall-effect sensors. The limitation of PMLR was that it could only travel in a predetermined path and had a significantly larger turning radius.

Based on the existing development of the IPRV and PMLR, the modified wheelchair already has wireless remote control capability by LAN connection and is controlled with the feedback from the Hall-effect sensors. With these capabilities, adding other kinds of sensors and modifying the motor controller could make it possible for the wheelchair to be operated with real-time path planning and obstacle avoidance.

---

This thesis follows the style of *IEEE Transactions on Automatic Control*.

## 1.2 Objective

The objective of this research is to demonstrate a robotic wheelchair moving in an unknown environment with real-time path planning. The generation of a real-time map and a moving path can be implemented by detecting the range from the obstacles, and by tracking specific lights sources used as beacons. Infrared sensors are used to detect the range form the obstacles and the light-sensitive resistors are used to track the light.

To optimize the motion trajectory, it is necessary to modify the motor controller of the wheelchair so that it can turn in a minimum turning radius. Then, with these kinematics, the algorithm of the real-time path planning of the robotic wheelchair can be simplified. In combination with the newly developed wireless Internet-connection capability, the robotic wheelchair will be able to navigate in an unknown environment.

## 1.3 Contributions

As described above, this thesis is the advance of the previous research, IPRV and PMLR. The specific contributions of this thesis are:

1. Adding the sensor system to let the wheelchair have the ability to detect obstacles in an unknown environment.
2. Modifying the interface board between the PCMDIO 24-channel data-acquisition input/output (I/O) card and the motor controller to let the wheelchair rotate about its geometric center.
3. Having the wheelchair be capable of collision avoidance navigation and tracking a beacon.

4. Developing a real-time path-planning algorithm by the capability described above. With this real-time path-planning algorithm, the wheelchair can become an autonomous robot which can move in an unknown or partially known environment.

In this thesis, we continue to use the wheelchair from the IPRV and PMLR projects as the main frame. The setting of the PCMDIO data-acquisition input/output (I/O) card has been modified.

### 1.4 Thesis Organization

Chapter I describes the history of this thesis and its contribution.

Chapter II presents the relevant literature reviewed by the author. The literature review is divided into several categories, modeling, sensor implementation, path planning, and obstacle avoidance.

Chapter III describes in detail the design of the autonomous robotic wheelchair in three steps. The first step involved the design of the main system of the robotic wheelchair. The second step involved the sensor system. The final step involved the wireless LAN communication. The description of the mechanical design of the robotic wheelchair is organized according to the development steps mentioned above.

Chapter IV describes the details of the development of the interface boards. In order to operate and control the robotic wheelchair, two interface boards were developed between

the laptop and the electronic components of the robotic wheelchair. First, it describes the PCMDIO data-acquisition card which is used for all I/O data acquisition. Second, it describes the interface circuits between the motor controllers and the laptop. Then, it describes the interface circuits between the sensor system and the laptop.

Chapter V describes the dynamics and kinematics of the robotic wheelchair and the algorithm of real-time path planning. The dynamics and kinematics of the robotic wheelchair are described first. By the analysis of the kinematics, the design of the real-time path-planning algorithm is described. The light-tracking capability is described next. Then, by implementing this real-time path-planning algorithm and light-tracking capability, the robotic wheelchair can become an autonomous robot.

Chapter VI describes the control program of the autonomous robotic wheelchair including the real-time path-planning algorithm, hardware control, and networking. This chapter describes the software to control the hardware, the real-time path planning algorithm, and wireless networking connection.

Chapter VII describes how these designs are integrated together to make the autonomous robotic wheelchair move in an unknown environment with collision avoidance navigation. It describes a typical operation mode of the autonomous robot with experimental results.

Chapter VIII summarizes the achievements of this thesis. The future work towards further development of the autonomous robotic wheelchair is also given.

# CHAPTER II

# LITERATURE REVIEW

A usual electronic wheelchair, an assistive device for people with impaired mobility, has motor controllers with limited capabilities for perception of their environment. The present work related to the development of robotic wheelchairs' navigation includes dynamic and kinematic modeling, path planning, target tracking, obstacle avoidance, sensors implementation, and wireless remote control.

## 2.1 Modeling

Modeling and control of a fast moving, highly maneuverable wheelchair was demonstrated in [3]. This project considered a wheelchair with two independently driven front wheels and two castors at the rear, and showed that the system became unstable when driven at high speeds. A nonlinear control scheme was proposed to handle this problem.

The kinematics and coordinate systems of a robotic wheelchair were given in [4] and [5]. In [4], the authors supposed that the wheelchair is move on a planar surface inside a "corridor" formed by obstacles, which was approximated by two straight parallel walls. They further supposed that appropriate sensors mounted on the wheelchair could detect the distance to the walls and derived the non-holonomic constraint on the motion of the wheelchair. From this, the instantaneous speed lateral to the moving direction of the mobile platform had to be zero. Thus, the employed wheelchair was kinematically equivalent to the unicycle-type mobile robot.

### 2.2 Sensor Implementation

Although sensor technology is continually improving, the cost of sensors is often too high for the mass-production of robots. Sensors implemented in robotics systems include global positioning system (GPS) receivers, laser range finders, cameras for image processing, ultrasonic sensors, and infrared sensors. These sensors can be used for navigation and obstacle avoidance.

There are many sensor systems for mobile robots. Sonars, used for distance measurement in a preselected critical direction, and a panoramic camera, were equipped in [4]. The ultrasonic sensors, used for navigation, were equipped in [5]. In [6], sensors were arranged on a circular robot. That paper presented a high-performance ultrasonic sensing system for mobile robots. They describe how wide-angle ultrasonic transducers can be used to obtain substantial information of the environment. An actuated laser scanner mounted on an unmanned aerial vehicle (UAV) [7]. The scanner was mounted on a tilt actuator with an encoder. The ultrasonic sensors and stereo cameras were equipped in the multi-vehicle platform of [8]. The platform consisted of ten wireless networked robots.

### 2.3 Path Planning and Obstacle Avoidance

Robotic wheelchairs and mobile robots explore in an unknown environment require map generation of surrounding and path planning for obstacle-avoidance navigation. Since early 1980's, various algorithms and implementations have been developed and available for guiding robotic wheelchairs and mobile robots in a two-dimensional (2-D) environment.

In [4], the robotic wheelchair was capable of obstacle avoidance while moving in the middle of free space and following a specified moving target. By processing the color sequence of the image from a panoramic camera, the robotic wheelchair could determine the orientation of the target with respect to itself. The distance of the wheelchair from the target could be measured by several sonars. In order to have certain desired features of the control system, the motion-control laws of motion used the sensory data and took into account the non-holonomic kinematic constraints of the wheelchair.

An agent-based robotic wheelchair was developed in [5]. Its controller contains the functions of path planning, navigation, and obstacle avoidance. In that work, a fuzzy logic was used for obstacle avoidance and smooth wheelchair motion control, and the algorithm was used to develop the path planning. Autonomous exploration for UAV was presented in [7], In that article, the authors proposed an algorithm suitable for urban navigation by combining the model predictive control. The algorithm was based on obstacle avoidance with a local obstacle map, which was built by an onboard laser scanner. A real-time gradient-search-based optimization let the model-predictive control solve for a collision-avoidance trajectory. The tracking control was responsible for following through the given trajectory.

The multi-vehicle platform in [8] discussed several coordinated control algorithms. The authors implemented the algorithms on cooperative multi-vehicle testbed, with low-level robotics vehicles and combining them to generate high-level controllers. The cooperative multi-vehicle testbed are based on potential-field control. The authors added

motion-coordination algorithms to the library of team controllers, which include perimeter estimation and pattern formation, dynamic target tracking, deployment, and rendezvous. The authors also explored optimal formation shapes to improve the performance of existing motion-coordination algorithms.

Information consensus in multi-vehicle cooperative control was discussed in [9] to provide a tutorial overview. Theoretical results regarding consensus-seeking under dynamically changing communication topologies was described. This article also described several specific applications of consensus algorithms to multi-vehicle coordination.

# CHAPTER III

# ROBOTIC WHEELCHAIR DESIGN

The robotic wheelchair in Figure 3.1 was designed in three steps. The first step involved the design of the main system of the robotic wheelchair. The second step involved the sensor system. The final step involved the Wireless internet communication. The description of the mechanical design of the robotic wheelchair is organized according to these development steps.

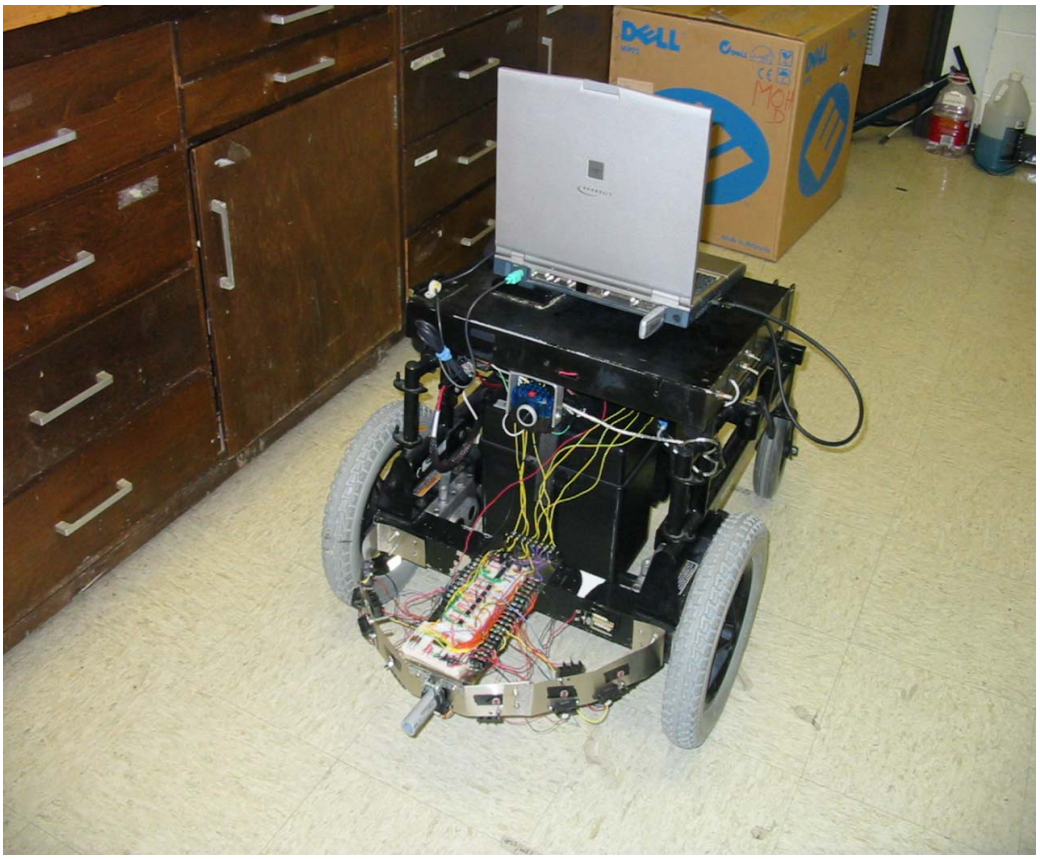



Figure 3.1. Development of the robotic wheelchair. The sensor system is mounted at the front, and the laptop for controlling the robotic wheelchair is on the top.

### 3.1 Step I – Main System of the Robotic Wheelchair

#### 3.1.1 Wheelchair

This robotic wheelchair is a take-over from the previous project in Precision Mechatronics Lab: IPRV and PMLR. It is built upon the base frame of an Invacare Ranger II™ electric powered wheelchair. The frame is 70-cm long, 48-cm wide, with a height of 55 cm. It is capable of supporting a weight of approximately 100 kg. This wheelchair is driven by two independent 12-V DC motors for the front wheels with a diameter of 31.75cm with built-in reduction gears that provide a maximum speed of 6 km/hr. Two Diverse Electronic Company's modular motor controllers are used for motion control and are mounted on the frame. Two 18-cm-diameter caster wheels in the rear provide support.

#### 3.1.2 Laptop

The main control program is operated by a Fujitsu Laptop with an AMD-K6 451-MHz processor and with 192 MB RAM. The main operation program is Visual Basic 6.0 in the Microsoft Windows XP operating system.

#### 3.1.3 Data-Acquisition Card

A Superlogics PCMDIO 24-channel digital I/O type II Personal Computer Memory Card International Association (PCMCIA) card is installed on the Fujitsu laptop and is used to perform all data-acquisition and control functions. A CP-1037 adapter cable is used to convert the PCMDIO's 33-pin 0.8-mm I/O connector to an industry standard D-37 connector. The PCMDIO has 24 transistor-transistor-logic (TTL) compatible buffered

digital-I/O channels individually programmable as either input or output.

### 3.1.4 Interface Board

The interface board shown in Figure.3.2 is between the PCMDIO and the Dervise MC-7 motor controllers. The logic signal from the PCMDIO is directly input to the two CD4066 switch chips, and the control voltage from a CD4066 chip is input to the motor controller for a certain speed of the wheelchair. A potentiometer is connected to one CD4066 chip to adjust the motor speed to ensure the wheelchair to move straight.

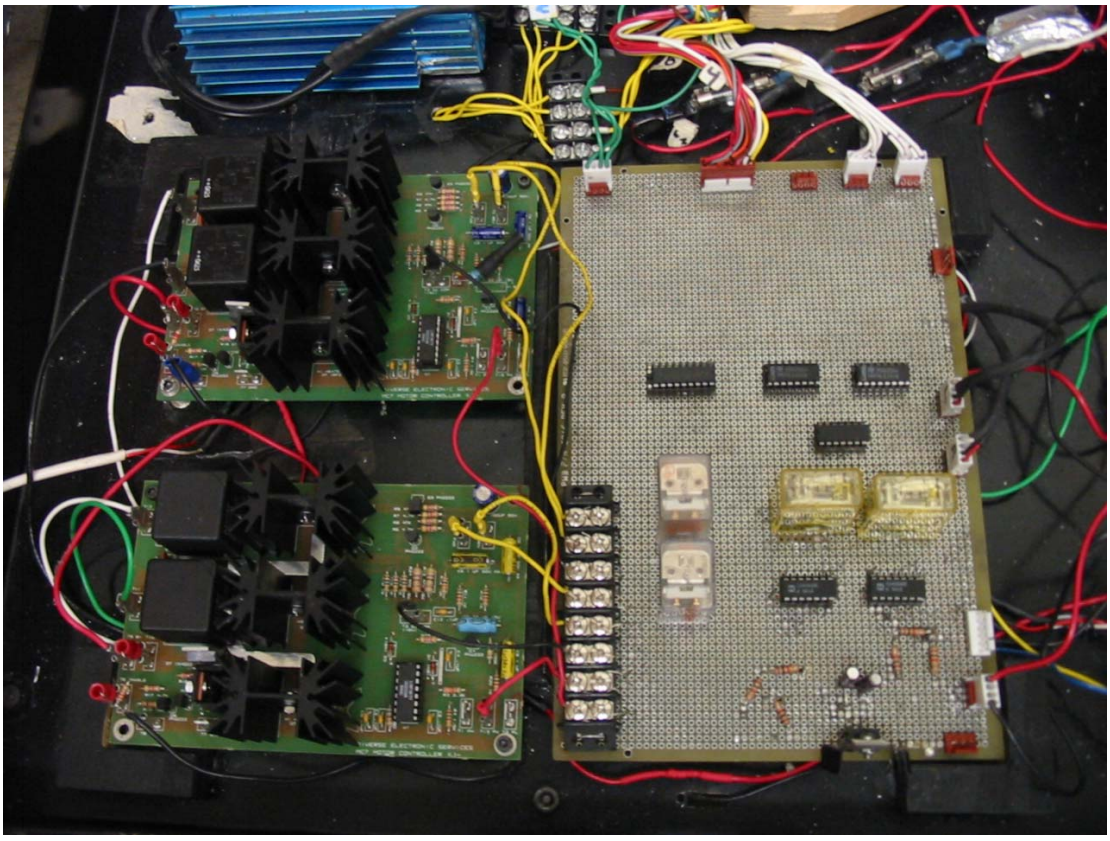


Figure 3.2. Interface board and two motor controllers. Four relays and CD4066 chips are connected to the MC-7. The interface board connected to the PCMDIO card with the connectors that were modified from IPRV and PMLR.

Two relays are connected between an ULN2803 Darlington array chip and two motor MC-7 controllers. The logic signal input to the Darlington array chip can select the forward mode or backward mode in the motor controllers. By this function the wheelchair can turn in a circle at original point. Two 74HC191 counter chips in PMLR are also rebuilt on the interface board for counting the pulses generated by the Hall-effect sensors.

## 3.2 Step II – The Sensor System

### 3.2.1 Light-Sensitive Resister

Seven CdS light-sensitive resistors are also referred to as photocells were assembled on the sensor bracket. The photocell is PDV-P5001 with a rise time of 55ms and with a typical resistance range of 8 kΩ to 16 kΩ at 10 lux at 2856K light. The photocell is connected to a 5-V power supply in series with a 1 kΩ resistor, and the voltage across the photocell's terminal is direct connected to an operational-amplifier comparator.

### 3.2.2 Distance-Measuring Sensor

Three GP2D15 and two GP2D12 infrared distance-measuring sensors manufactured by Sharp as shown in Figure3.3, are used to detect obstacles. The GP2D15 detects an obstacle at 24-cm range and the GP2D12, from 12 cm to 80 cm. The sensors generate the output voltage signals fed to the analog-to-digital converters on the interface board to the PCMDIO card.

### 3.2.3 Hall-Effect Sensors

Two Hall-effect sensors from IPRV and PMLR were mounted on the rear casing of both the motors. A pulse is generated by the Hall-effect sensors on every rotation of the motor shaft and fed to a circuit with a 74HC191 counter chip and input to the PCMDIO data-acquisition card installed on the laptop. When the wheelchair is moving in a path, the distance traversed by it is proportional to the number of rotations of the motor shaft. This resolution of the Hall-effect sensors was found to be approximately 1cm in the previous IPRV and PMLR research.

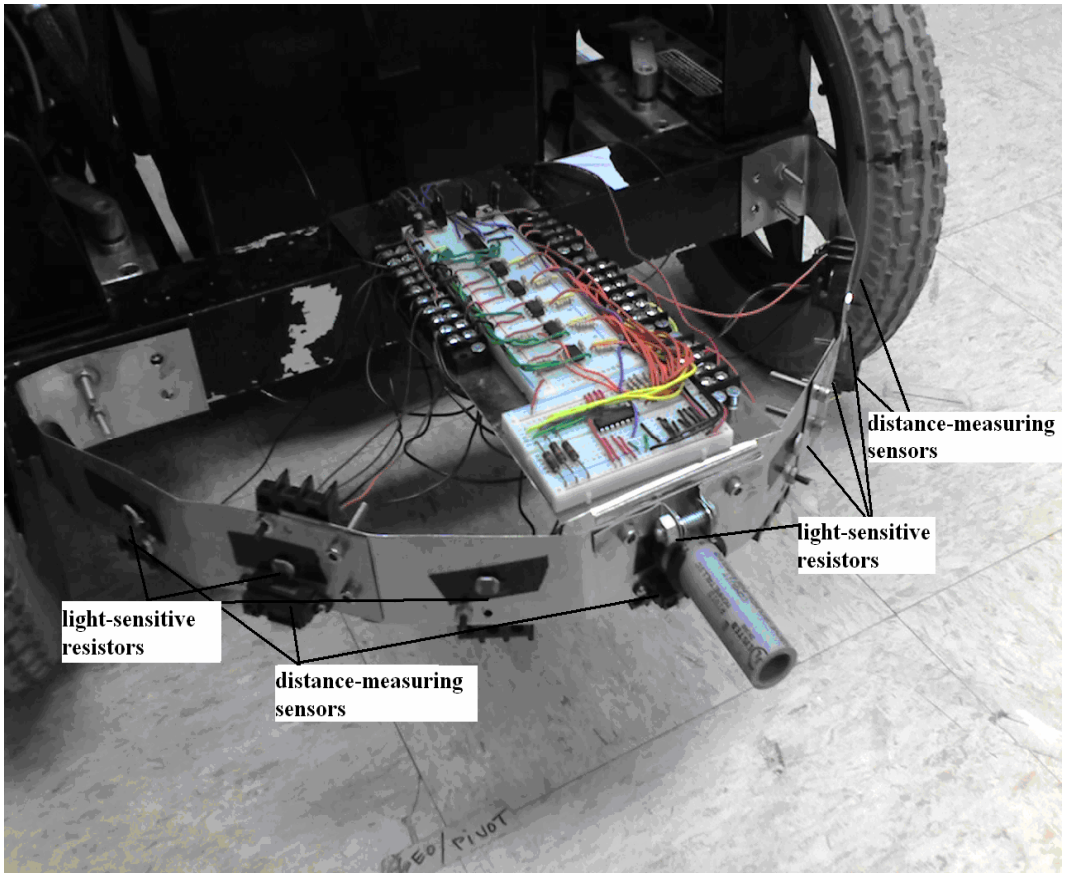



Figure 3.3. Seven light-sensitive resistors and five distance-measuring sensors are mounted on the sensor bracket with interface circuit board.

### 3.2.4 Interface Board on the Sensor System

An interface board connects between the sensor system and the PCMDIO card with the electronic components will be given in Section 4 for detailed description.

### 3.2.5 Operational Amplifiers

Four TL072ACP and three LM741 operational-amplifiers shown in Figure 3.4 are employed to compare the voltage signals from the photocells and the five Infrared sensors.

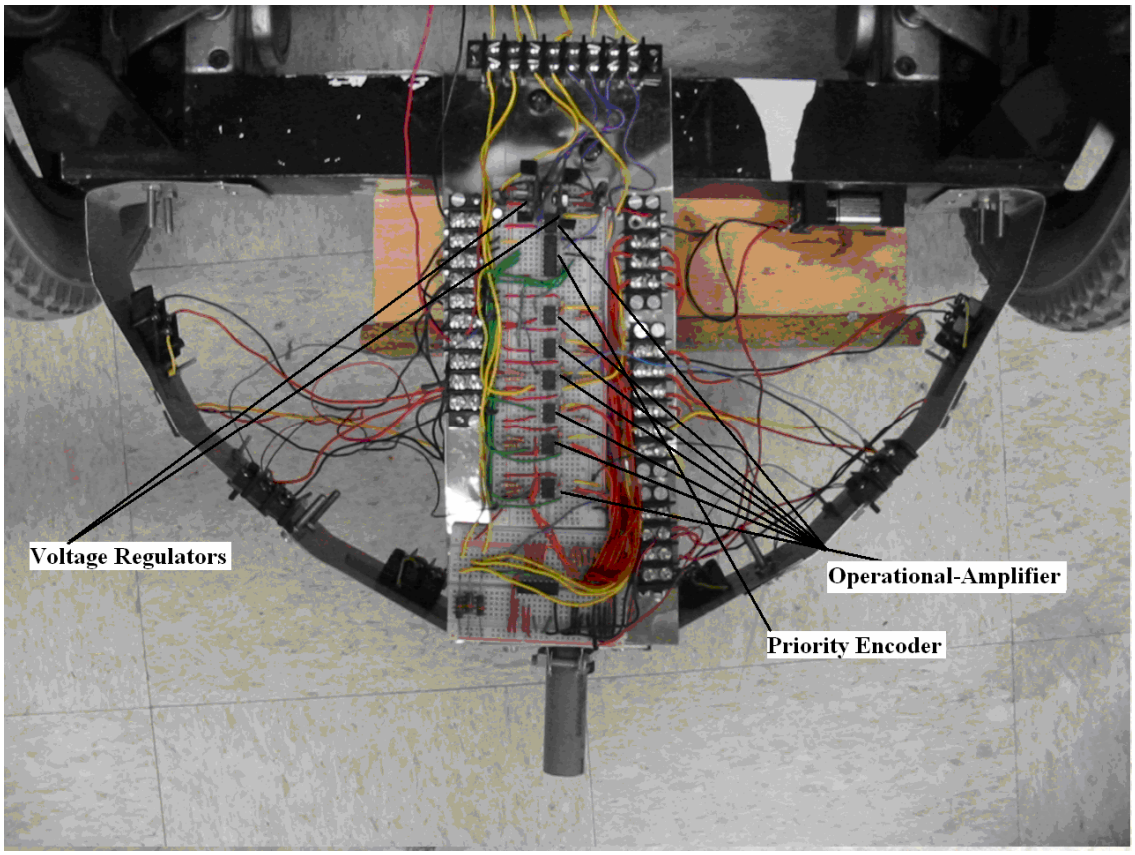



Figure 3.4. Interface board between the sensor bracket and the PCMDIO. The chips from top to bottom are the voltage regulators, operational-amplifiers, and priority encoder.

### 3.2.6 Priority Encoders

The output signals from operational-amplifier comparator are input to a 74LS148 priority encoder that generates a three-bit output signal and directly input to the PCMDIO data-acquisition card on the laptop.

### 3.2.7 Voltage Regulator

A KA7805 and a KA7905 voltage regulators are used to supply positive and negative 5 V to the whole electronic circuit.

## 3.3 Step III – Wireless Internet

Remote operability of the mobile robot is provided by use of a wireless LAN card installed on the laptop. This capability was developed in the IPRV and PMLR research. The wheelchair acts as a server and executes the server-side program. Any remote terminal executes the client-side program on the same LAN can be used to remotely operate the robot.

# CHAPTER IV

# INTERFACING

This chapter describes the details of the development of the interface boards. In order to operate and control the robotic wheelchair, two interface boards were developed between the laptop and the electronic components of the robotic wheelchair described in the last chapter. Section 4.1 describes the PCMDIO data-acquisition card from IPRV and PMLR which is used for all I/O data acquisition. Section 4.2 describes the interface circuits between the motor controllers and the laptop. Section 4.3 describes the interface circuits between the sensor system and the laptop.

## 4.1 The PCMDIO Data-Acquisition Card

A Superlogics PCMDIO 24-channel digital I/O type II Personal Computer Memory Card International Association (PCMCIA) card is installed on the Fujitsu laptop. It is used to perform all data acquisition and control functions. A CP-1037 adapter cable is used to convert the PCMDIO's 33-pin 0.8-mm I/O connector to an industry standard D-37 connector.

The PCMDIO has 24 TTL compatible buffered digital-I/O channels individually programmable as either input or output. These digital-I/O channels are grouped into several different ports with each port containing several channels. These ports are controlled via the Data Port A, Data Port B, and Data Port C control registers, respectively. In all three registers, each bit corresponds to one data line. The eight Port C I/O channels may also be configured as interrupt sources. The interrupts may be

configured in four ways: level-sensitive active-low interrupt, level-sensitive active-high interrupt, high-to-low transition-edge-sensitive interrupt, and low-to-high transition-edge-sensitive interrupt.

### 4.2 Interfacing the Motor Controllers

The interface board shown in Figure 4.1 is between the PCMDIO and the Dervise MC-7 motor controllers. The logic signal from the PCMDIO is directly input to the two CD4066 switch chips and two relays, and the control voltage from the relays input to the motor controller for certain speed of the wheelchair. A potentiometer is connected to one CD4066 chip and one relay to adjusting the motor speed for ensuring the wheelchair moving near straight without feedback.

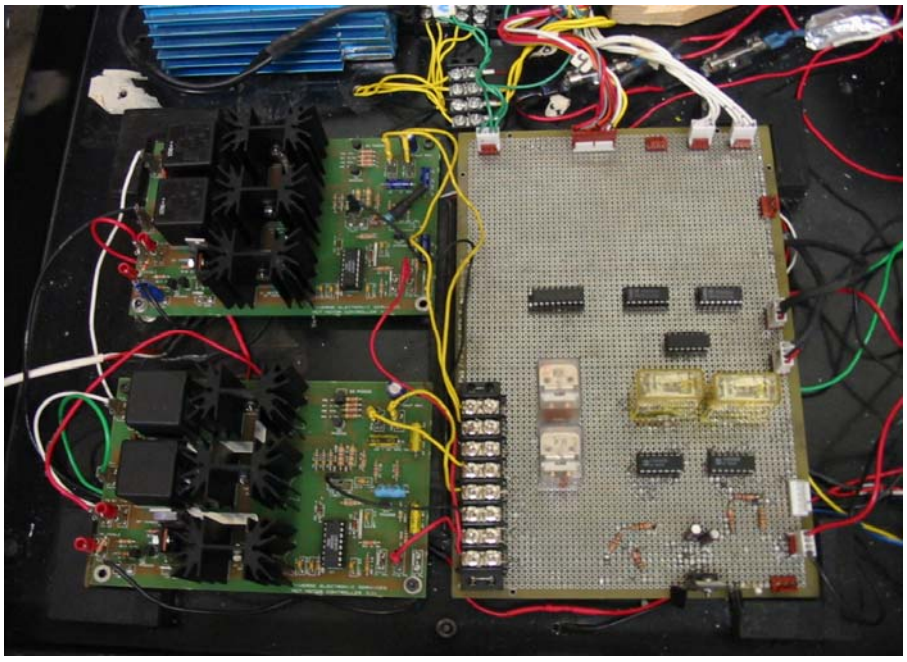


Figure 4.1. MC-7 motor controller and the interface board.

Two relays are connected between an ULN2803 Darlington array chip and two motor MC-7 controllers. The logic signal input to the Darlington array chips can select the forward mode or backward mode in the motor controllers. By this function the wheelchair can turn in a circle at original point. Two 74HC191 counter chips design in PMLR are also rebuilt on the interface board for counting the pulse generated by the Hall-effect sensors.

### 4.2.1 Speeding Control

The MC-7 motor controller is manufactured by Diverse Electronics and is used to power a DC motor by producing a pulse width modulation (PWM) power supply voltage. It has a power output range from 12 V to 36 V and can accept the control signal input. The range of voltage of the input signal is 1V to 3V, where 1V indicates the minimum speed and 3V indicates the maximum speed. In order to provide this control signal to the motor controller, we select the voltage as 1.66 V, which is easy to design the circuit and provides the proper speed for the robotic wheelchair. The circuit between the MC-7 motor controllers and the PCMDIO data acquisition card is shown in Figure 4.3. Two relays connect to pin T13 of the MC-7 motor controllers and ULN2803 Darlington array chip. This circuit provides the switch function as stop and start. Two CD 4066 switch chips provide the function of selecting speed if we need different speed.

### 4.2.2 Forward and Backward Control

The MC-7 controller will drive an electric motor in both the forward and backward directions. Two relays are connected between an ULN2803 Darlington-array chip and

two MC-7 motor controllers. The logic signal input to the Darlington-array chips can select the forward mode or backward mode in the motor controllers. By this function the wheelchair can turn in a circle at an original point with one wheel moving forward and the other moving backward. This tight-rotation function is very important in real-time path planning. The circuit for this function is shown in Figure 4.2.

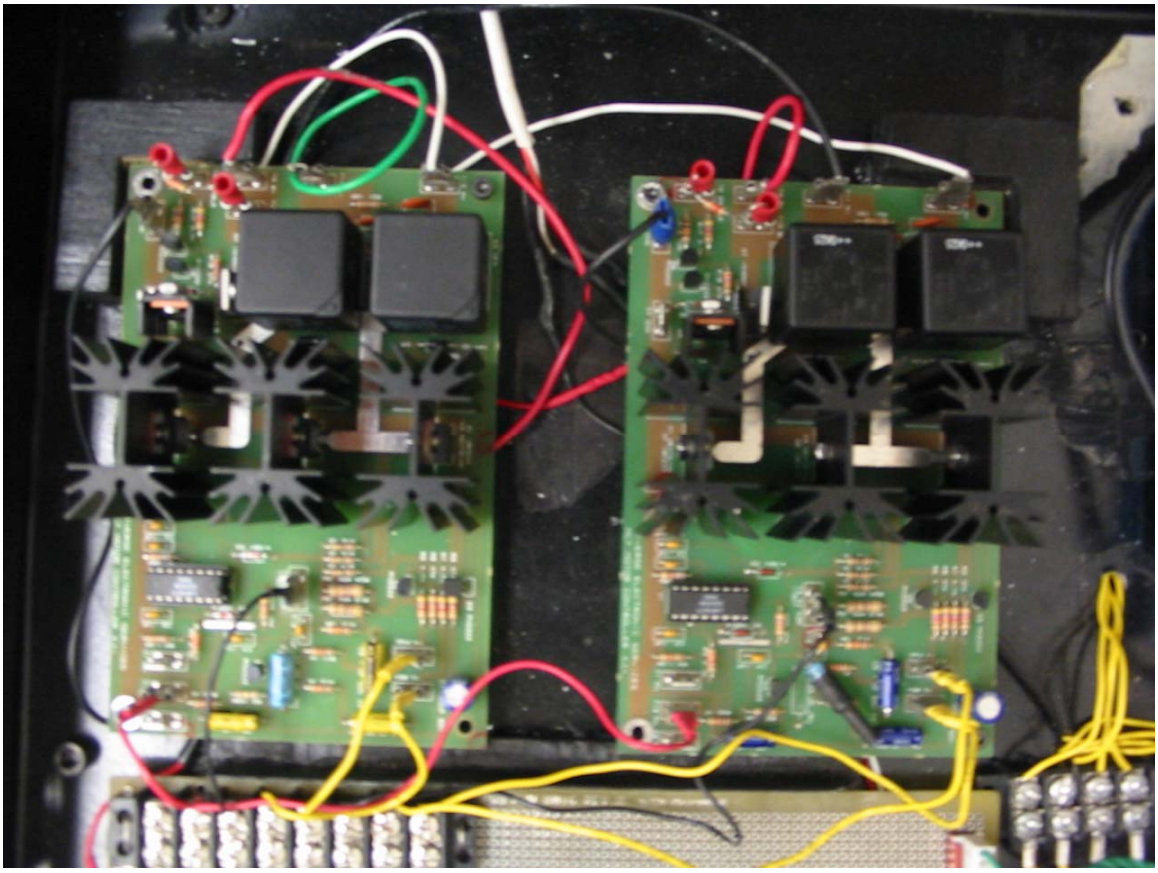


Figure 4.2. Dervise MC-7 motor controllers.

In Figure 4.3, we can see that two relays connect to pin T13 of the MC-7 motor controller,

which provide the function of the speeding control. Two relays connect to pins T3, T4, and T5 of the MC-7 motor controller, which provide the forward and backward control ability.

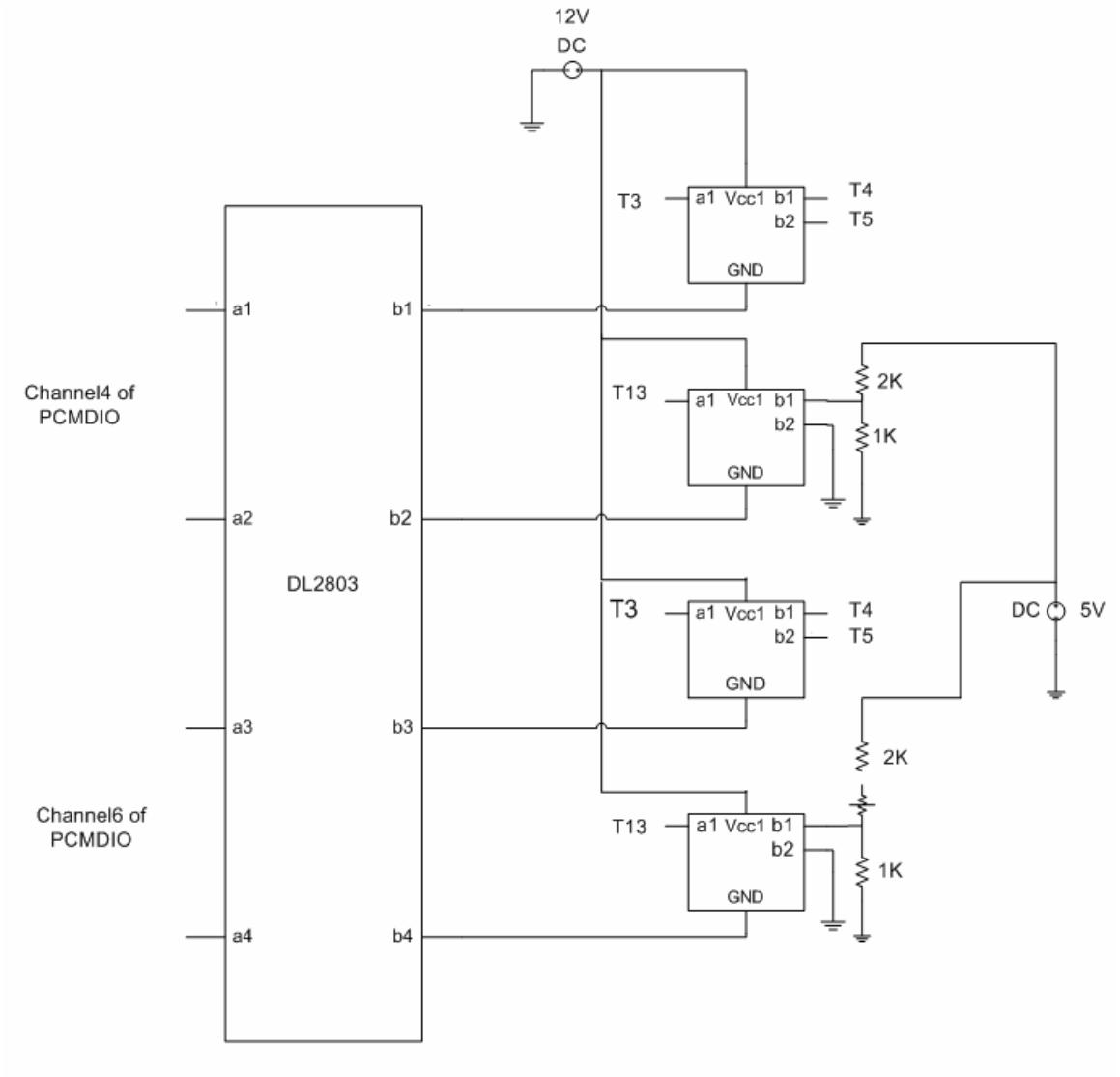



Figure 4.3.Circuit for speeding, forward, and backward control.

### 4.2.3 Movement Measurement

Two 74HC191 decade counter chips shown in Figure 4.4 from IPRV and PMLR are also

included on the interface board to count the pulses generated by the Hall-effect sensors. By this function, it has the ability to measure the moving distance of the robotic wheelchair. Referred to [10] for the details.

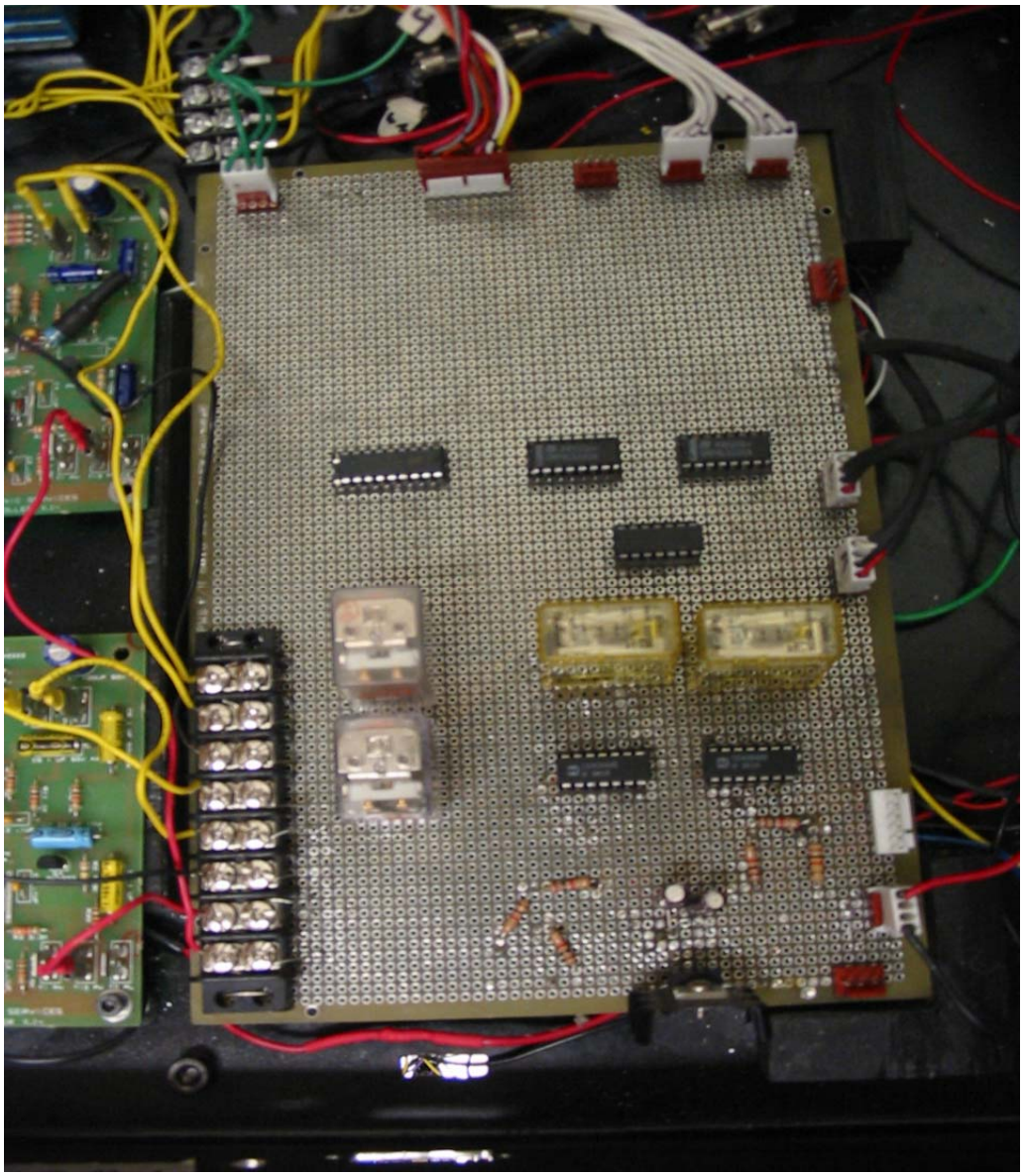


Figure 4.4. Interface board that contains four relays, two CD4066 chips, one Darlington-array chip, and two 74HC191 counters.

### 4.3 Interfacing the Sensor System

The PCMDIO is a TTL-comparable I/O card. It can also take the digital input signal. It is necessary to build an interface circuit as an ADC function between the sensor system and the PCMDIO. Interface board for the sensor system is shown in Figure 4.5.

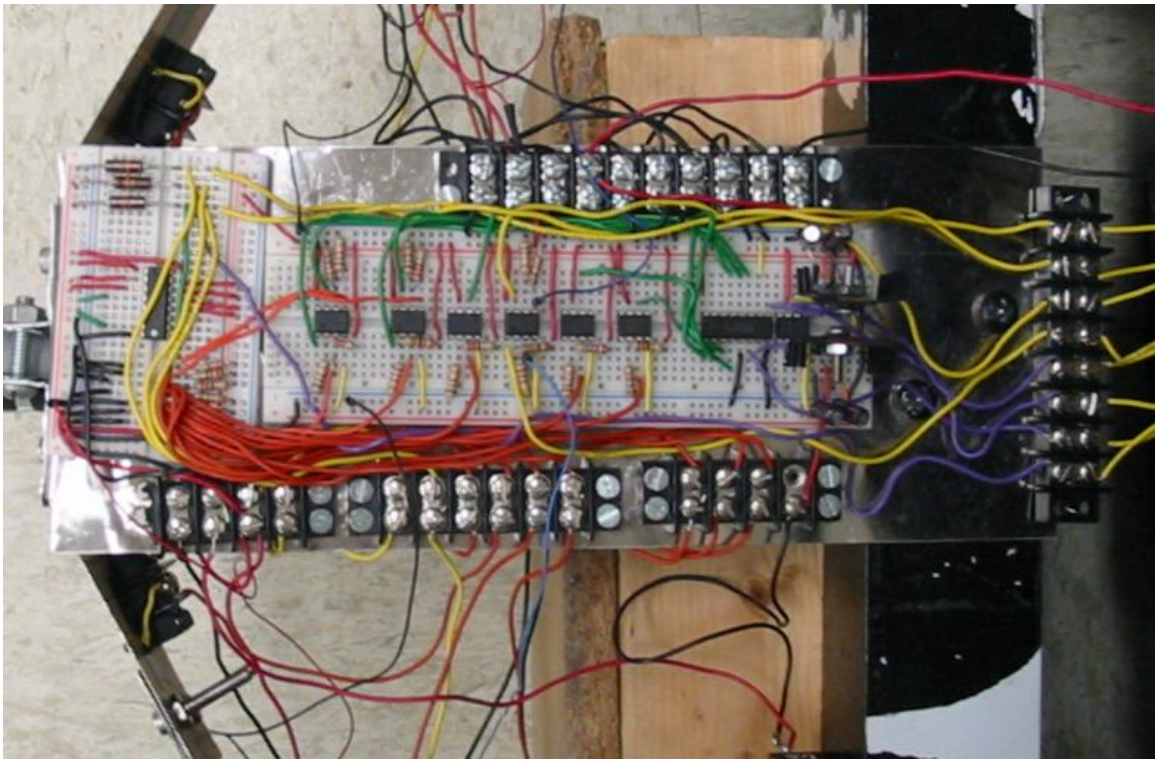


Figure 4.5. Interface board for the sensor system.

#### 4.3.1 Interfacing the Light-Sensitive Resistors

Seven CdS light-sensitive resistors are also known as photocells were assembled on the sensor bracket. The photocell is PDV-P5001 with a rise time of 55 ms and with a typical resistance range of 8 kΩ to 16 kΩ at 10 lux at 2856K light. The photocell is connected to a 5-V power supply in series with a 1 kΩ resistor, and the voltage across the photocell's

terminal is direct connected to an operational-amplifier comparator. The operational-amplifier is design as voltage comparators. By selecting the reference voltage, the comparator provides the 5-V output while the light is darker than the desired brightness, and provides 0-V output while the light is brighter than the desired brightness. This function can also refer as an ADC the PCMDIO. For describing this function, we analysis the voltage comparator designed by operational- amplifier shown in Figure 4.6.

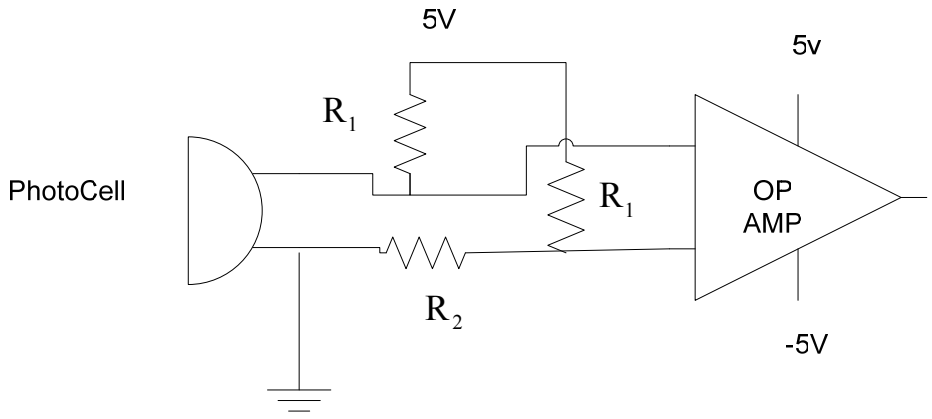



Figure 4.6. Voltage comparator.

The reference voltage is set as $V_{ref} = 5 \frac{R_1}{R_1+R_2}$. In this project, we select $R_1 = 220\ \Omega$ and $R_2 = 1\ k\Omega$. Then, if the voltage generated by the photocell drops below 0.41 V, the output voltage of the operational amplifier becomes 0-V. This can also be referred as a logic-low level signal to the 74LS148 priority encoder. Figure 4.7 shows the entire circuits between the seven photocells and the PCMDIO card.

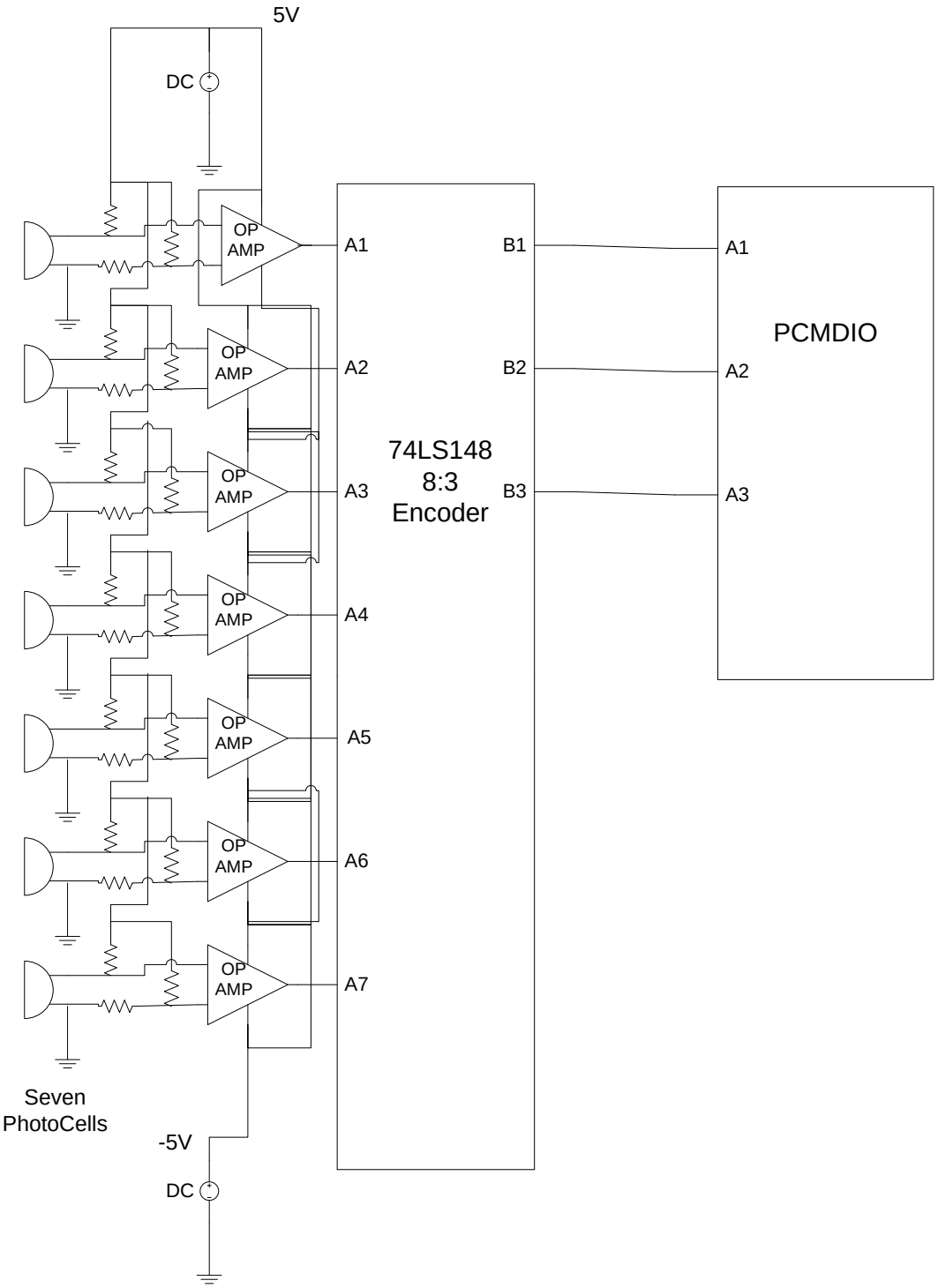



Figure 4.7. Circuit between photocells and PCMDIO.

### 4.3.2 Interfacing the Distance-Measuring Sensor

Three Sharp GP2D15 and two GP2D12 distance-measuring sensors also known as infrared sensors are used to detect obstacles. The GP2D15 detects an obstacle at a 24 cm range and the GP2D12 detects an obstacle at the range from 12 cm to 80 cm. From the datasheet of the GP2D15, for generating a 5-V output signal, it is necessary to connect to a 12 kΩ resister. The GP2D12 is an analog sensor, we can set the ADC outputs 5-V signal to the PCMDIO while the GP2D12 detects the obstacle at the range of 70 cm.

# CHAPTER V

# KINEMATICS AND PATH PLANNING

This chapter describes the kinematics and dynamics of the robotic wheelchair and the algorithm of the real-time path planning. The analysis of the kinematics and dynamics for the robotic wheelchair is described in Section 5.1 and 5.2. By the analysis of the kinematics for the robotic wheelchair, the design of the real-time path planning algorithm is described in Section 5.3. The light tracking capability is described in Section 5.4. In Section 5.5 by implementing this real-time path planning algorithm and light tracking ability, the robotic wheelchair can become as an autonomous robot.

## 5.1 Dynamics of the Wheelchair

The dynamics of the two driving wheel vehicle can be written as an equation below:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{V}_1 \\ \dot{V}_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(V_1 + V_2)\cos\theta \\ \frac{1}{2}(V_1 + V_2)\sin\theta \\ \frac{1}{2}(V_1 - V_2) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2 , \tag{5.1}$$

where $V_1$ and $V_2$ are the velocities of two driving wheels, $x$ and $y$ are the positions of the axle middle point of two driving wheels in the two dimension reference frame, and $\theta$ is the turning angle of the vehicle [11].

By setting

$$V = \frac{1}{2}(V_1 + V_2)$$
$$\omega = \frac{1}{l}(V_1 - V_2) \tag{5.2}$$

We can get

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} V + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega \tag{5.3}$$

From the dynamics equation (5.3) and [11], we can see that the middle point of axle can be fixed at an original point if $V_1 = -V_2$. Thus, for designing the path-planning algorithm of the two-driving-wheel vehicle, we do not have to consider the turning radius of the axle middle point of the vehicle, but consider the size of the vehicle.

### 5.2 Kinematics of the Wheelchair

By experiments and measurement, we can see that if we let the wheelchair turning at an original point, which is the vertical axis on the center of the wheelchair axle. The maximum turning radius $\overline{OA}$ of the wheelchair to prevent collision is approximately 60 cm. Figure 5.1 shows the robotic wheelchair turns in an original point and the detecting range of five infrared sensors. We select the detecting range of the two GP2D12 infrared sensors as 70 cm.

In Figure 5.1, we can see that the circle is the turning trajectory of the wheelchair turning at an original point. The detecting range of two GP2D12 and three GP2D15 infrared

sensors are indicated as dash lines. The robotics wheelchair could avoid any obstacles outside the turning trajectory, and it will be much easier to design a real-time path-planning. The algorithm of the real-time path-planning can be simplified if the motion trajectory of the robotic wheelchair turning around the axle middle point is a circle.

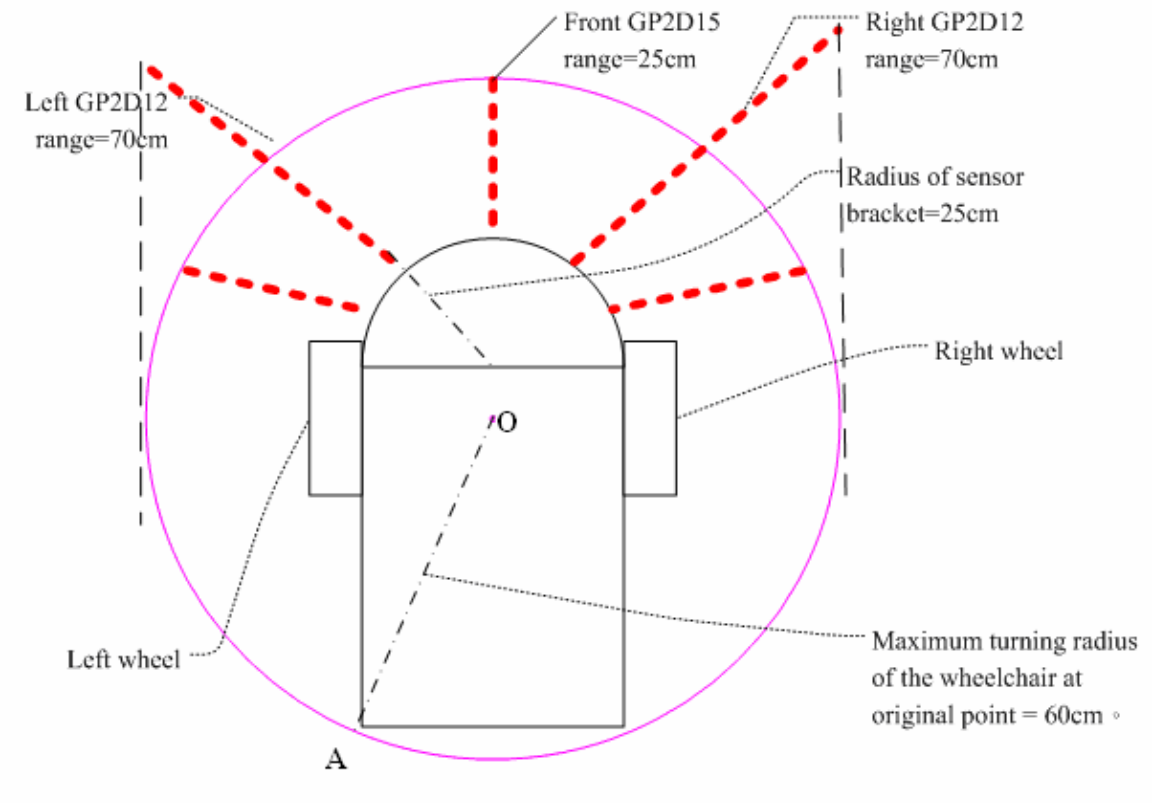



Figure 5.1. The robotic wheelchair turns in an original point and the detecting range of five infrared sensors.

### 5.3 Algorithm of the Real-Time Path Planning Guided by Infrared Sensors

To design the algorithm of the real-time path-planning, we can set the reaction of the robotic wheelchair to let it turns left, turns right, moving forward, or stop, according to

each condition of the infrared sensors' signal. From some experiments and testing, we set the reaction of the robotic wheelchair in Table 5.1.

There are five infrared sensors assembled on the sensor bracket. Two Sharp GP2D12 infrared sensors assembled on the right and left sides as Figure 5.1. We set the detecting range as 70 cm. Three GP2D15 sensors have the detecting range at 25 cm. One GP2D15 infrared sensor was mounted at the front. Two GP2D15 infrared sensors assembled on the left and right side just behind the GP2D12 sensors. These two GP2D15 sensors are used to eliminate the dead zone. Since the GP2D12 and GP2D15 sensors detect different distance to obstacles, it can prevents some dead zone and detects the parallel obstacles.

With these sensors arrangement, there are 32 conditions distinct of the signal from these five sensors. In Table 5.1, the H means the infrared sensor generates logic-high level signal to the PCMDIO, and L means a logic-low level signal.

While the robotic wheelchair running in autonomous mode, it will keep moving forward if there is no signal from any of the five infrared sensors. If there are signals from those five infrared sensors, it will response according to each of these 32 conditions. It may turn right, turn left, keep moving forward, or stop. The schematic diagrams of the response of the robotic wheelchair in the autonomous mode after it detects the obstacles are given in Tables 5.1 to 5.33 and Figures 5.2 to 5.33. The details of the autonomous mode algorithm will be described in Chapter VI.

Table 5.1. Response of the robotic wheelchair for each condition.

| Logic Signal from Infrared Sensors | | | | Response of Robot | |
|---|---|---|---|---|---|
| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L | Front GP2D15's signal is H |
| L | L | L | H | Left turn | Left turn |
| L | L | H | L | Left turn | Left turn |
| L | L | H | H | Left turn | Left turn |
| L | H | L | L | Right turn | Right turn |
| L | H | L | H | Forward | Stop |
| L | H | H | L | Forward | Stop |
| L | H | H | H | Left turn | Stop |
| H | L | L | L | Right turn | Right turn |
| H | L | L | H | Forward | Stop |
| H | L | H | L | Forward | Stop |
| H | L | H | H | Forward | Stop |
| H | H | L | L | Right turn | Right turn |
| H | H | L | H | Forward | Stop |
| H | H | H | L | Right turn | Stop |
| H | H | H | H | Forward | Stop |
| L | L | L | L | Forward | Stop |

Table 5.2. Response to the first condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| L | L | L | H | Left turn |

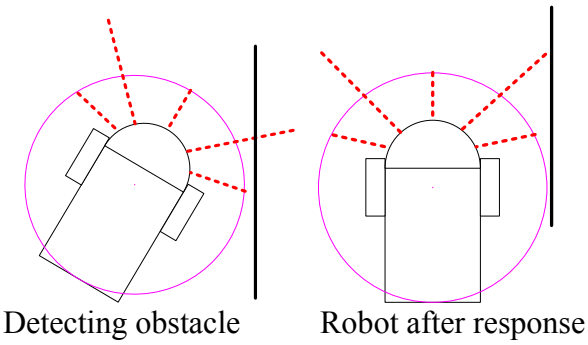



Figure 5.2. The first condition. The robotic wheelchair detects the obstacle to the right. It turns counter-clockwise until it detects no obstacle.

Table 5.3. Response to the second condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| L | L | L | H | Left turn |

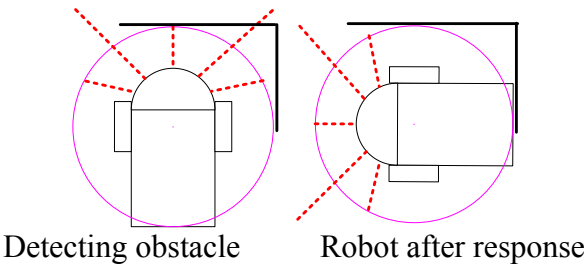



Figure 5.3. The second condition. The robotic wheelchair detects the obstacle to the right and in the front. It turns counter-clockwise left until it detects no obstacle.

Table 5.4. Response to the third condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| L | L | H | L | Left turn |

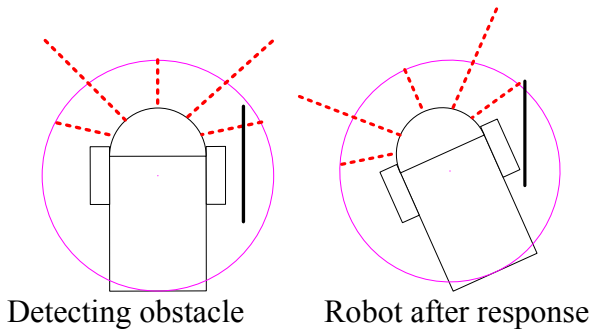



Figure 5.4. The third condition The robotic wheelchair detects the obstacle to the right and front. It turns counter-clockwise until it detects no obstacle.

Table 5.5. Response to the fourth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| L | L | H | L | Left turn |

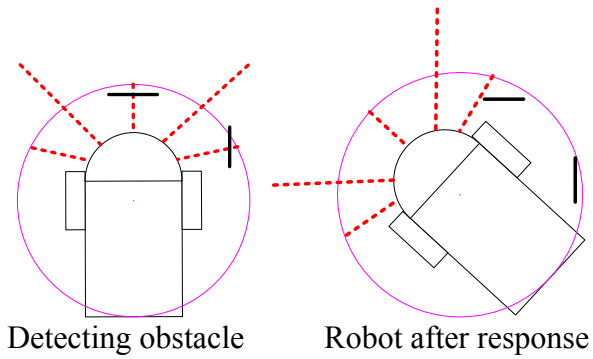



Figure 5.5. The fourth condition. The robotic wheelchair detects the obstacle to the right and in the front. It turns counter-clockwise until it detects no obstacle.

Table 5.6. Response to the fifth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| L | L | H | H | Left turn |

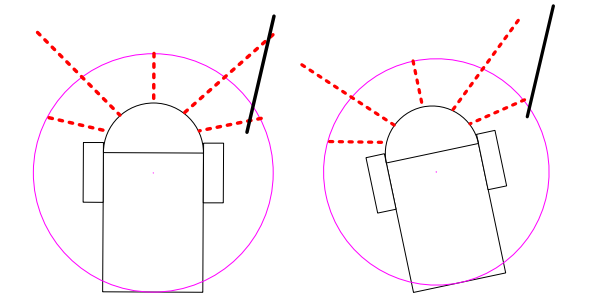



Figure 5.6. The fifth condition. The robotic wheelchair detects the obstacle to the right. It turns counter-clockwise until it detects no obstacle.

Table 5.7. Response to the sixth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| L | L | H | H | Left turn |

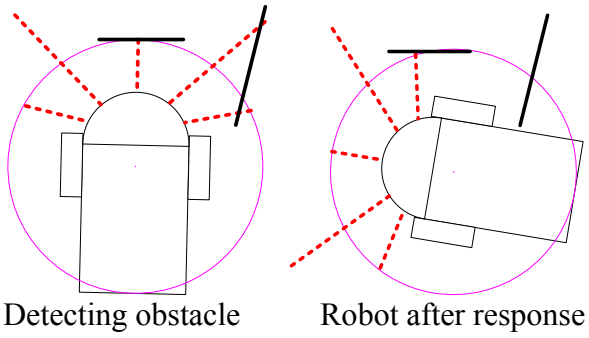



Figure 5.7. The sixth condition. The robotic wheelchair detects the obstacle to the right and in the front. It turns counter-clockwise until it detects no obstacle.

Table 5.8. Response to the seventh condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| L | H | L | L | Right turn |

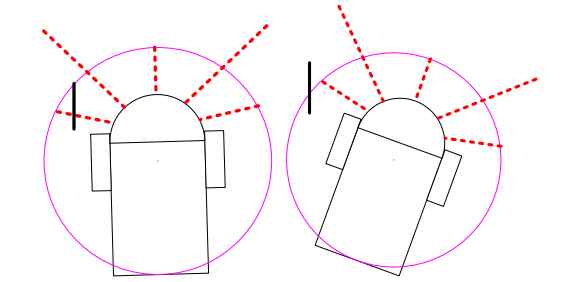



Figure 5.8. The seventh condition. The robotic wheelchair detects the obstacle to the left. It turns clockwise until it detects no obstacle.

Table 5.9. Response to the eighth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| L | H | L | L | Right turn |

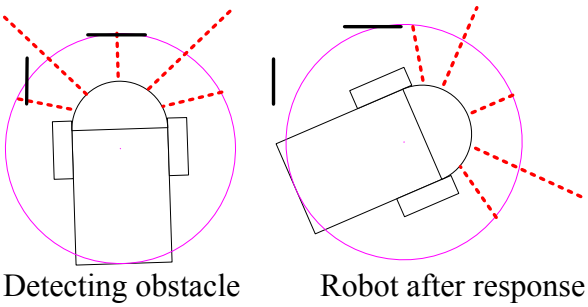
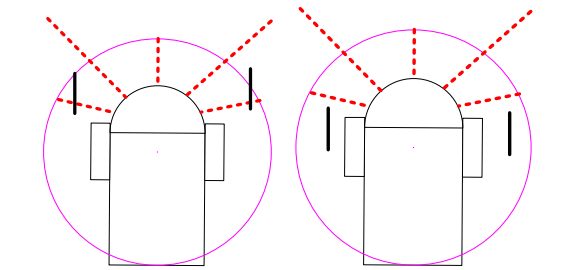



Figure 5.9. The eighth condition. The robotic wheelchair detects the obstacle to the left and in the front. It turns clockwise until it detects no obstacle.

Table 5.10. Response to the ninth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| L | H | L | H | Forward |

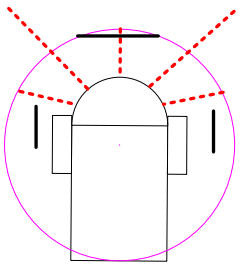


Detecting obstacle Robot after response

Figure 5.10. The ninth condition.The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle.

Table 5.11. Response to the tenth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| L | H | L | H | Stop |



Unpredictable situation, stop.

Figure 5.11. The tenth condition The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops.

Table 5.12. Response to the eleventh condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| L | H | H | L | Forward |

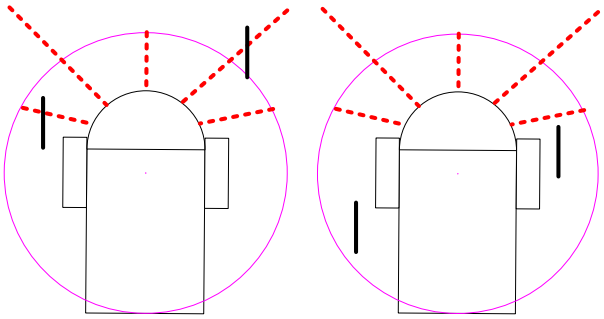



Figure 5.12. The eleventh condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle.

Table 5.13. Response to the twelfth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| L | H | H | L | Stop |

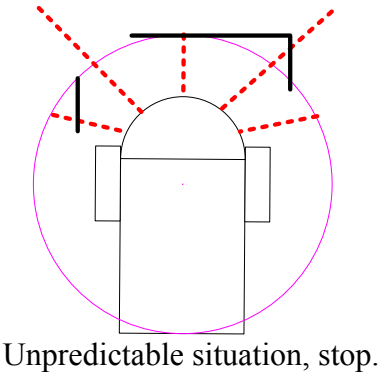



Figure 5.13. The twelfth condition The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops.

Table 5.14. Response to the thirteenth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| L | H | H | H | Left turn |

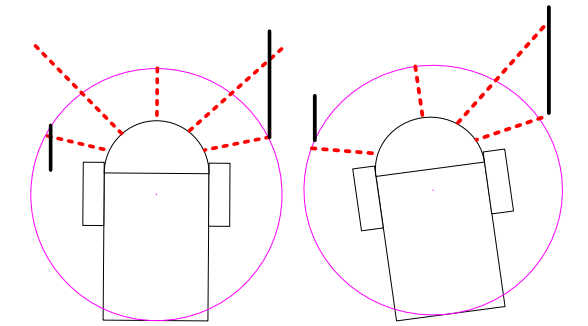



Figure 5.14. The thirteenth condition The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle.

Table 5.15. Response to the fourteenth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| L | H | H | H | Stop |

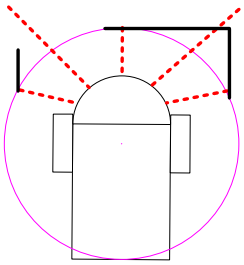



Figure 5.15. The fourteenth condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops.

Table 5.16. Response to the fifteenth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| H | L | L | L | Right turn |

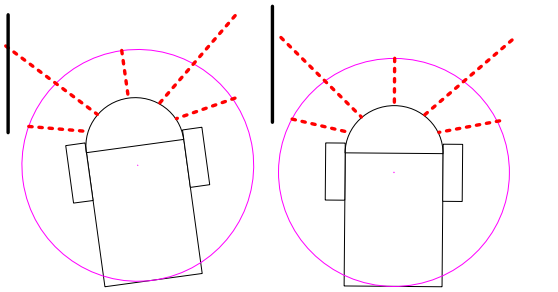



Figure 5.16. The fifteenth condition.The robotic wheelchair detects the obstacle to the left. It turns clockwise until it detects no obstacle.

Table 5.17. Response to the sixteenth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| H | L | L | L | Right turn |

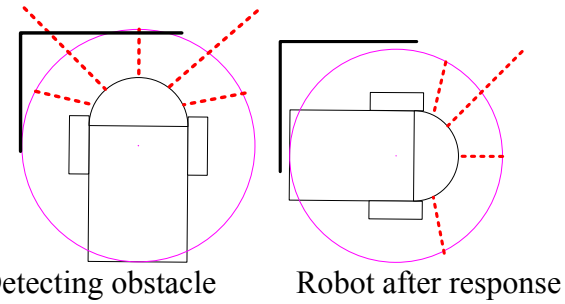
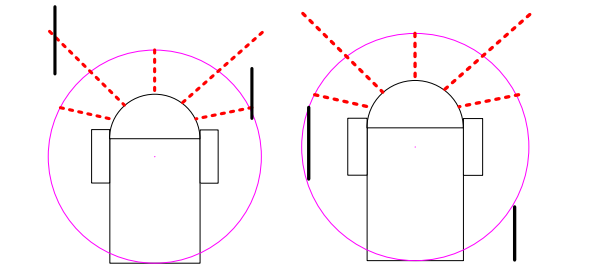



Figure 5.17. The sixteenth condition. The robotic wheelchair detects the obstacle to the left and in the front. It turns clockwise until it detects no obstacle.

Table 5.18. Response to the seventeenth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| H | L | L | H | Forward |

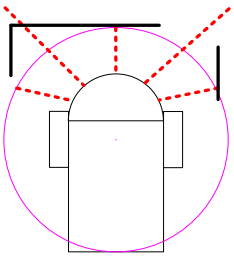


Detecting obstacle Robot after response

Figure 5.18. The seventeenth condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle.

Table 5.19. Response to the eighteenth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| H | L | L | H | Stop |

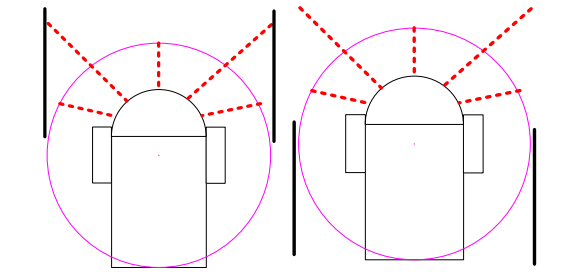


Unpredictable situation, stop.

Figure 5.19. The eighteenth condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops.

Table 5.20. Response to the nineteenth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| H | L | H | L | Forward |

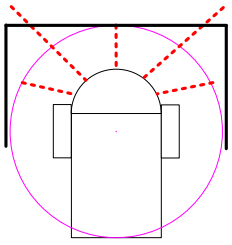


Detecting obstacle Robot after response

Figure 5.20. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle.

Table 5.21. Response to the twentieth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| H | L | H | L | Stop |



Unpredictable situation, stop.

Figure 5.21. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops.

Table 5.22. Response to the twenty-first condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| H | L | H | H | Forward |

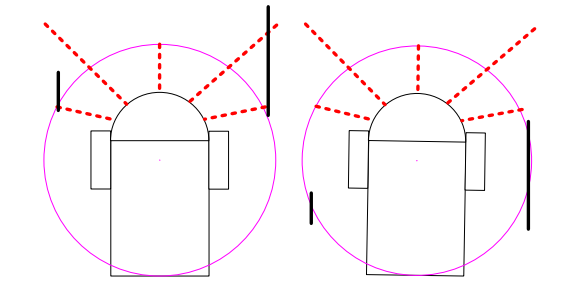



Figure 5.22. The twenty-first condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle.

Table 5.23. Response to the twenty-second condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| H | L | H | H | Stop |

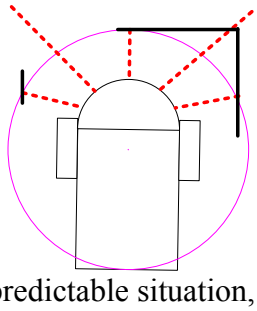
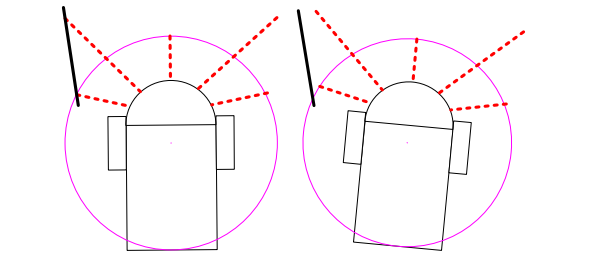



Figure 5.23. The twenty-second condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops.

Table 5.24. Response to the twenty-third condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| H | H | L | L | Right turn |

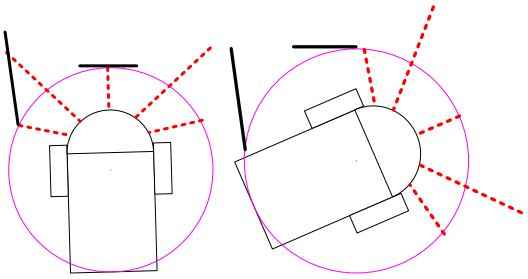


Detecting obstacle Robot after response

Figure 5.24. The twenty-third condition. The robotic wheelchair detects the obstacle to the left. It turns clockwise until it detects no obstacle.

Table 5.25. Response to the twenty-fourth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| H | H | L | L | Right turn |



Unpredictable situation, stop.

Figure 5.25. The twenty-fourth condition. The robotic wheelchair detects the obstacle to the left and front. It turns clockwise until it detects no obstacle.

Table 5.26. Response to the twenty-fifth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| H | H | L | H | Forward |

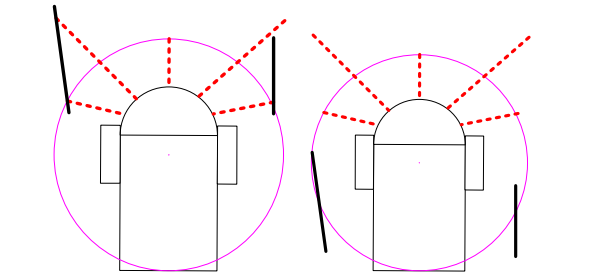



Figure 5.26. The twenty-fifth condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle.

Table 5.27. Response to the twenty-sixth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| H | H | L | H | Stop |

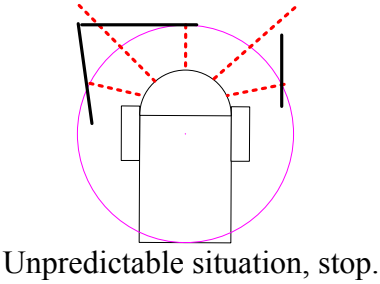



Figure 5.27. The twenty-sixth condition.The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops.

Table 5.28. Response to the twenty-seventh condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| H | H | H | L | Forward |

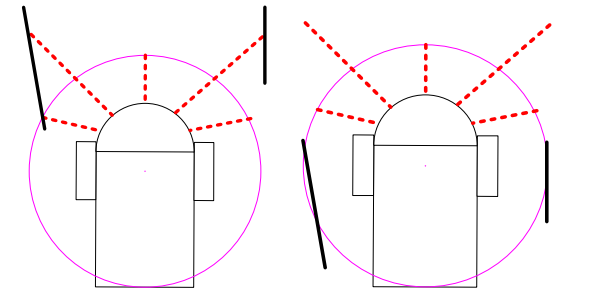



Figure 5.28. The twenty-seventh condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle.

Table 5.29. Response to the twenty-eighth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| H | H | H | L | Stop |

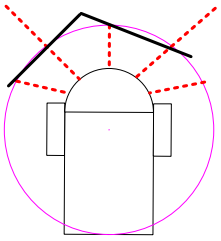
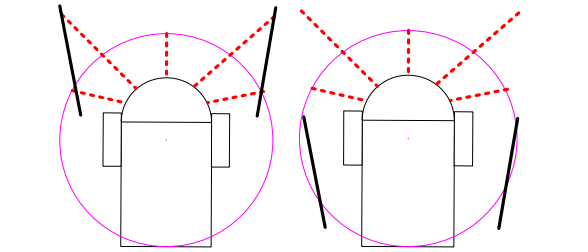



Figure 5.29. The twenty-seventh condition.The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops.

Table 5.30. Response to the twenty-ninth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| H | H | H | H | Forward |

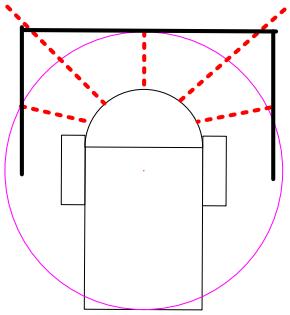


Detecting obstacle Robot after response

Figure 5.30. The twenty-ninth condition. The robotic wheelchair detects the obstacle to the left and right. It keeps moving forward until it detects an obstacle.

Table 5.31. Response to the thirtieth condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| H | H | H | H | Stop |



Unpredictable situation, stop.

Figure 5.31. The thirtieth condition. The robotic wheelchair detects the obstacle to the left, right, and in the front. This is an unpredictable situation and it stops.

Table 5.32. Response to the thirty-first condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is L |
|---|---|---|---|---|
| L | L | L | L | Forward |

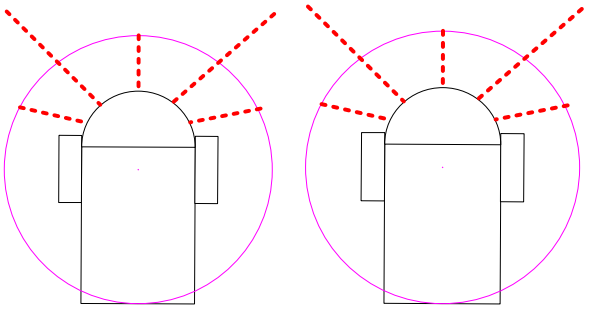



Figure 5.32. The thirty-first condition.The robotic wheelchair detects no obstacle and it keeps moving forward.

Table 5.33. Response to the thirty-second condition.

| Left GP2D12 | Left GP2D15 | Right GP2D12 | Right GP2D15 | Front GP2D15's signal is H |
|---|---|---|---|---|
| H | H | H | H | Stop |

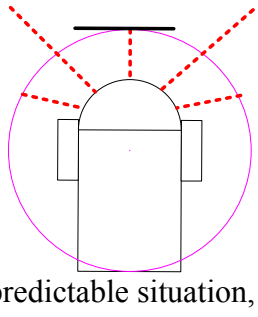



Figure 5.33. The thirty-second condition. The robotic wheelchair detects the obstacle in the front. This is an unpredictable situation and it stops.

## 5.4 Light Tracking

The robotic wheelchair has the capability of tracking a motion trajectory defined with a light with the seven photocells mounted on the sensor bracket. The algorithm of this tracking capability is: If any of the three photocells on the left detects the light, then the wheelchair turns counter-clockwise until the front photocell detects the light. If any of the three photocells on the right detect the light, the wheelchair turns clockwise until the front photocells detect the light. The diagrams to illustrate this light-tracking capability are shown in Figure 5.34 and 5.35.

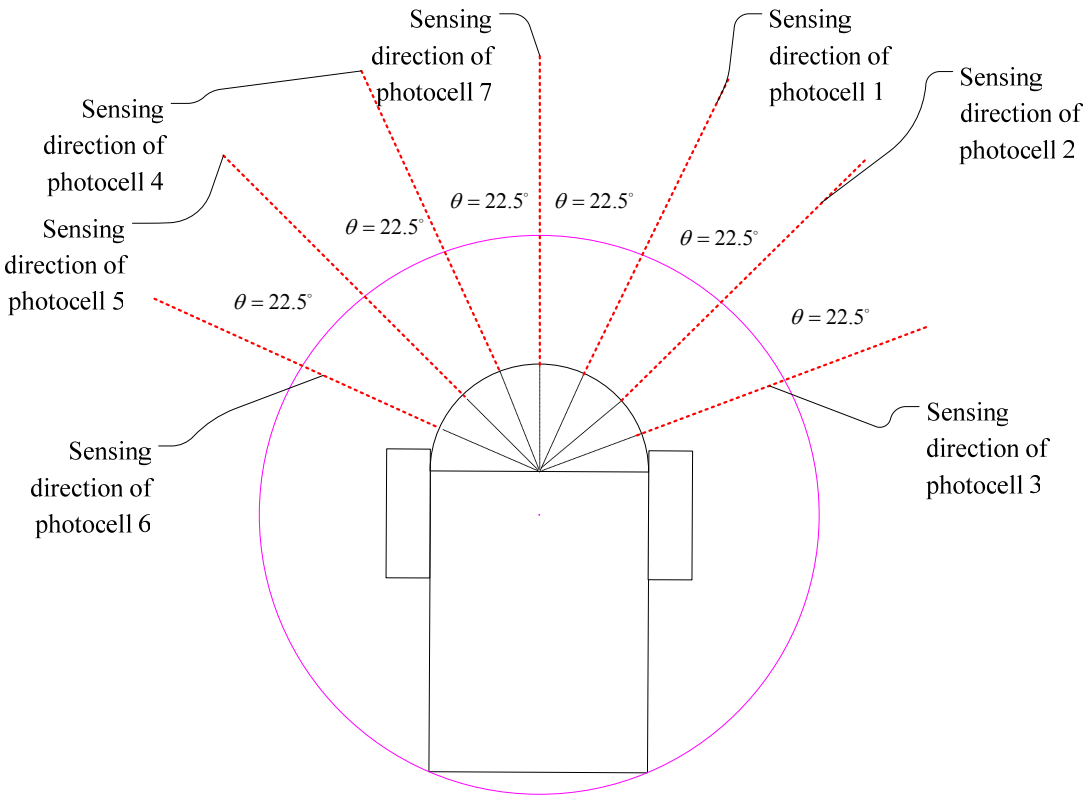



Figure 5.34. Sensing directions of the photocells are indicated as dashed lines.

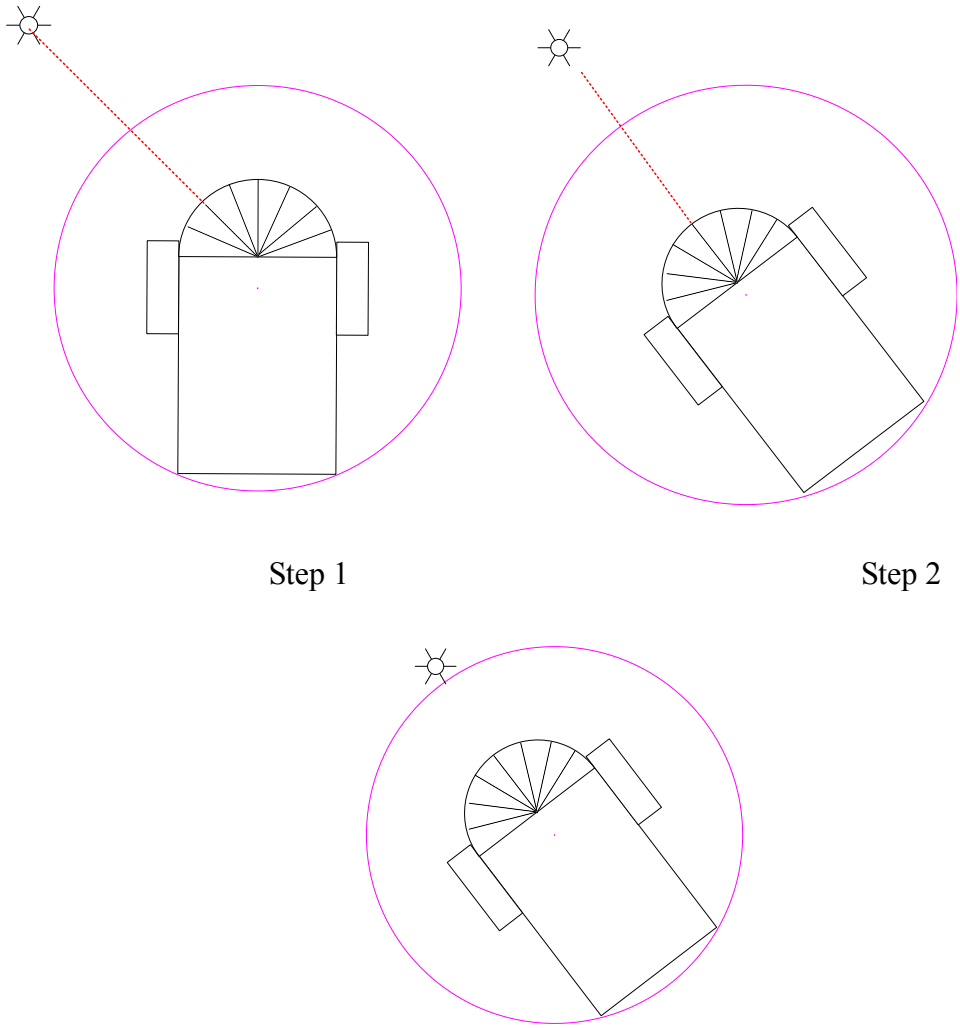



Figure 5.35. Steps of the robotic wheelchair track a specific light.

Step 1: The left photocells detect the light.

Step 2: The wheelchair turns to the counter-clockwise until the front photocell detects the light.

Step 3: After the front photocell detects the light, the wheelchair moves to the target and stops at 25 cm away.

# CHAPTER VI

# SOFTWARE DESIGN

The control program of the autonomous robotic wheelchair includes real-time path planning, hardware control, and networking. This chapter describes each program. Section 6.1, it describes the programming and software being used. Section 6.2 describes the software to control the hardware. Section 6.3 describes the real-time path planning algorithm. Section 6.4 describes the wireless networking connection.

## 6.1 Programming Language

The Microsoft® Windows® Visual Basic 6.0 for Windows development environment is being used for the programming requirements of the robotic wheelchair. It provided a single platform to write programs for all the applications of the robotic wheelchair. In the IPRV development, the Microsoft Windows application programming interface (API) was utilized to develop the application to control the PCMDIO digital-I/O card [1]. It use of the Windows API provides direct access to the dynamic-link-library (DLL) files to operate the PCMDIO card. This development is also need in this thesis research.

The PCMDIO digital-I/O data-acquisition card is used for all data acquisition and output control signal to operate the robotic wheelchair. The vendor of the PCMDIO card also provides the PCMDRIVE® data acquisition software. The software includes the following components. For the details of the PCMDRIVE software can be found in [12] and [13].

### 6.1.1 PCMDRIVE Configuration Utility

This software was specifically designed to support the PCMDIO data acquisition adapter function. It is easy to use the application that allows the user to graphically acquire and display real-time data. This software is used to edit the PCMDIO hardware configuration file. This file contains the setup of the 24 individual I/O channels of the PCMDIO card into logical channels. Using the configuration software, each logical channel can be set as single-bit or multiple-bit channels. Once all the logical channels have been set, each channel may be configured as an input channel or an output channel. The PCMDRIVE configuration utility with the 24 data I/O lines is shown in Figure 6.1. For the autonomous robotic wheelchair, the PCMDIO was configured to have 8 logical channels because of some limitation from the PMLR's development. The detail of the channel configuration is shown in Table 6.1.

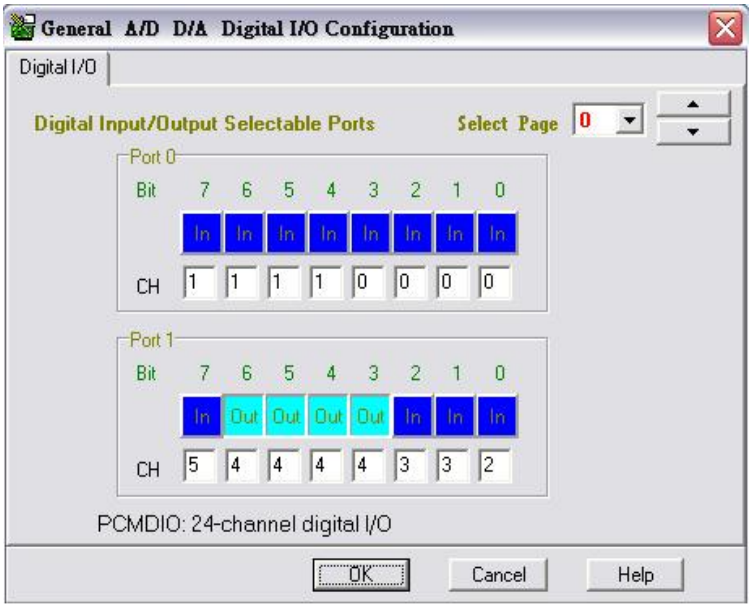



Figure 6.1. PCMDRIVE configuration utility [13].

Table 6.1. PCMDIO channel configuration.

| Logical Channel | Number of bits | Channel type | Function |
|---|---|---|---|
| CH 0 | 4 | Input | Signal from left hall effect sensor. |
| CH 1 | 4 | Input | Signal from right hall effect sensor. |
| CH 2 | 1 | Input | Signal from left GP2D15. |
| CH 3 | 2 | Input | Signal from right GP2D15 and GP2D12. |
| CH 4 | 4 | Output | Control the left wheel. |
| CH 5 | 1 | Input | Signal from left GP2D12. |
| CH 6 | 4 | Output | Control the right wheel. |
| CH 7 | 1 | Input | Signal from front GP2D15 |
| CH 8 | 3 | Input | Signal from 74LS148 encoder. |

### 6.1.2 Performing Data Acquisition

PCMDRIVE uses a data-defined interface, and each data-acquisition operation is defined by a series of configuration parameters. These parameters are contained in a data structure and are collectively referred to as a request or a request structure. From the IPRV development , in order to perform an input or output operation using the PCMDIO, it requires the following sequence of steps [1]:

1. Define the hardware configuration.

2. Open the hardware device.

3. Allocate the request structure and data buffers.

4. Define the request structure and data buffers.

5. Request the operation.

6. Write data to the locked data buffer.

7. Arm the request.

8. Trigger the request.

9. Wait for completion.

10. Read data from the locked data buffer.

11. Release the configuration.

12. Close the hardware device.

There are five functions were specially created in order to simplify the use of the PCMDIO for digital I/O operations. The functions are

1. Function openDevice

2. Function singleDigitalInput

3. Function multipleDigitalInput

4. Function singleDigitalOutput

5. Function multipleDigitalOutput.

The detail of those twelve sequences and five functions can be found in [1].

### 6.2 Hardware Control

To equip the autonomous robotic wheelchair with feedback control ability, we set the logical channels, input and output data lines, and the bits number per logic channel of the PCMDIO I/O card as Table 6.1. The six input channels take 16 bits of the input data lines.

These are for the signals from two GP2D12 infrared sensors, three GP2D15 infrared sensors, the 71HC191 counters count the signal from the right and the left Hall-effect sensors, and the 74LS148 priority encoder to encode the signals from seven photocells. In thisr research, these two channels are used to control the speed of the wheelchair and to generate the forward and backward motion.

### 6.3 Operation of the Robotic Wheelchair

The real-time path-planning algorithm was described in Section 5.3 and 5.4. To program this algorithm in Visual Basic 6.0, the operation interface with the user was designed in Visual Basic 6.0 as Figure 6.2.

The interface for the user includes the “manual mode” and “autonomous mode” for the user operating the robotic wheelchair. If the user presses the manual mode button, the wheelchair can be controlled by the user manually. This function includes front, right, left, stop, and back motions of the wheelchair. If the user presses the autonomous mode button, the program will run the algorithm shown in Figure 6.3.

In this autonomous mode, we set the sampling time interval from the input data lines of the PCMDIO as 100 ms, and the output control signals are 100 ms interval.

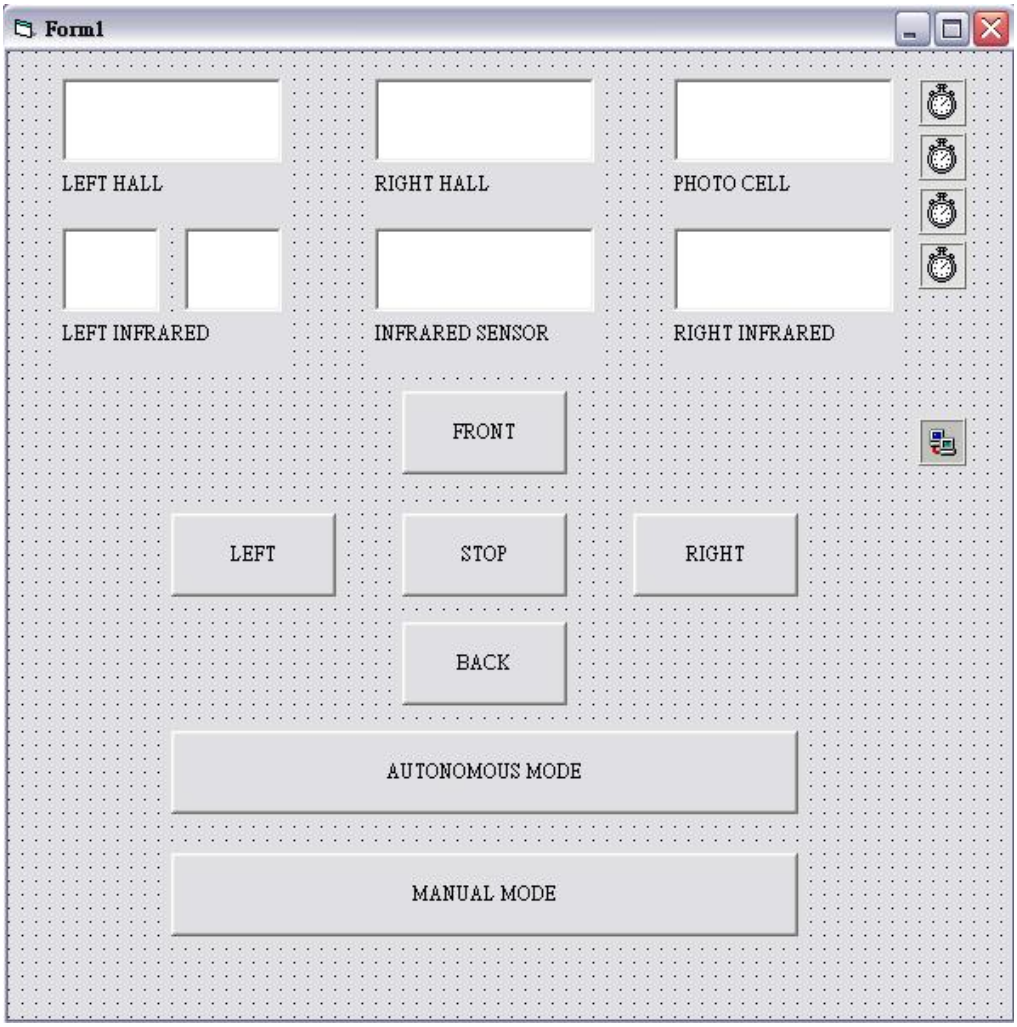



Figure 6.2 Operating interface with the user in Visual Basic 6.0.

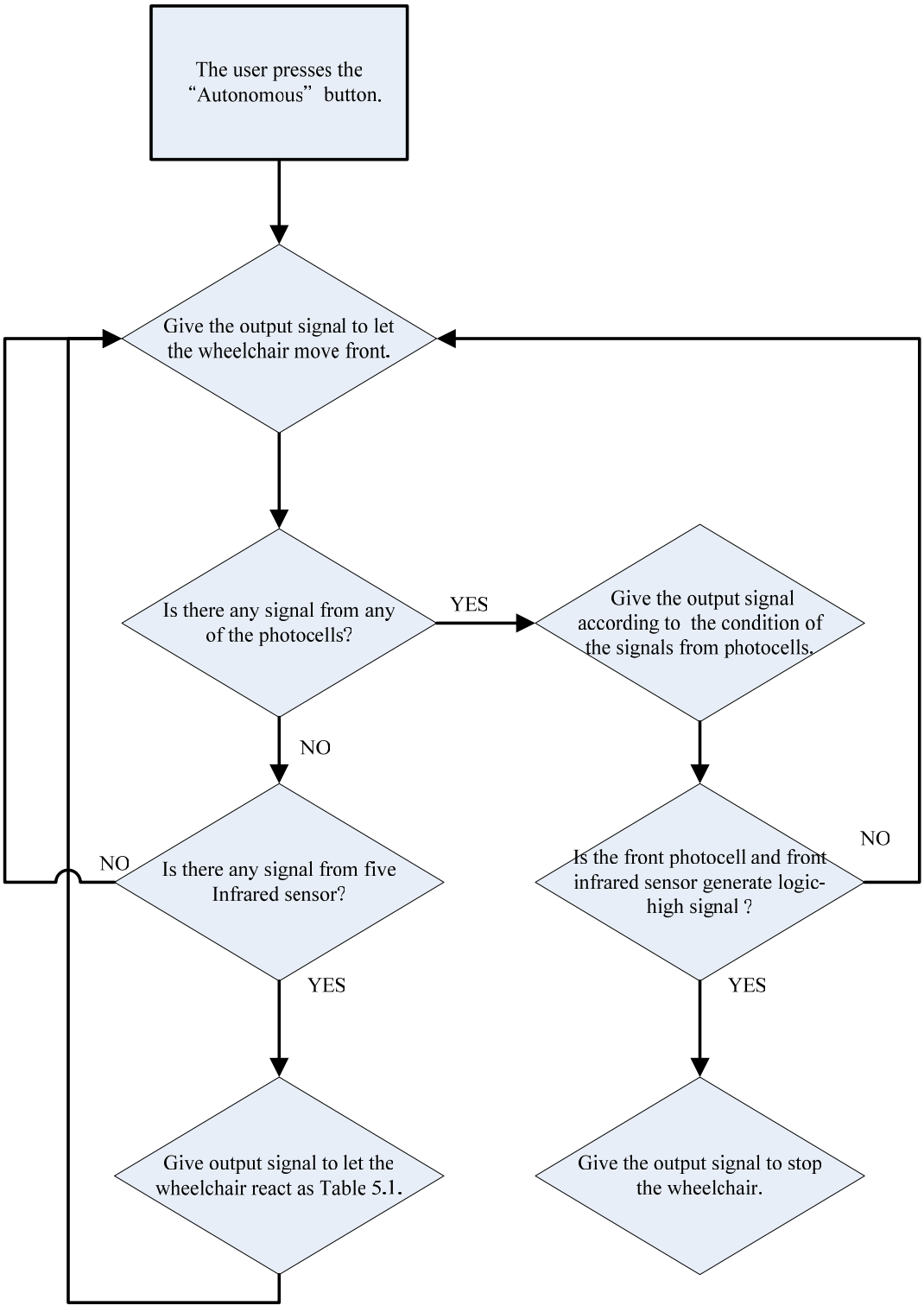



Figure 6.3 Algorithm for the "autonomous" mode of the robotic wheelchair.

### 6.4 Remote Control

Remote operability of the robotic wheelchair is provided by interfacing with a LAN using a wireless USB LAN card installed on the laptop. In the development of the PMLR, it used an “ad-hoc” technique to equip the PMLR with remote control ability [2]. This ad-hoc technique can only be controlled by the client computer in the same network, and there was only a 10-meter effective range to control the PMLR.

With the newly developed technique by Cheng-Yeh Hsu in Precision Mechatronics Lab, the autonomous robotic wheelchair could be controlled by any computer connected to the Internet while the robotic wheelchair moving in the environment with a Wi-Fi access. We used the Tamulink system, which is a Wi-Fi access provided by Texas A&M University almost everywhere on its campus.

The transport layer protocol used for sending and receiving data is the Transmission Control Protocol (TCP). The Microsoft Winsock Control 6.0 ActiveX control is used for the implementation of the TCP sockets within Visual Basic 6.0 [Appendix B]. While the client computer has the IP address of the laptop on the wheelchair, the user on the client-side computer could control the robotic wheelchair with the client-side program.

A schematic of the control system is shown in Figure 6.4 and 6.5. It can be seen that the commands from the client computer send through the Tamulink wireless Internet system. The controller will have response according to the commands from client computer and the sensor system. The interface of the client-side program is shown in Figure 6.6.

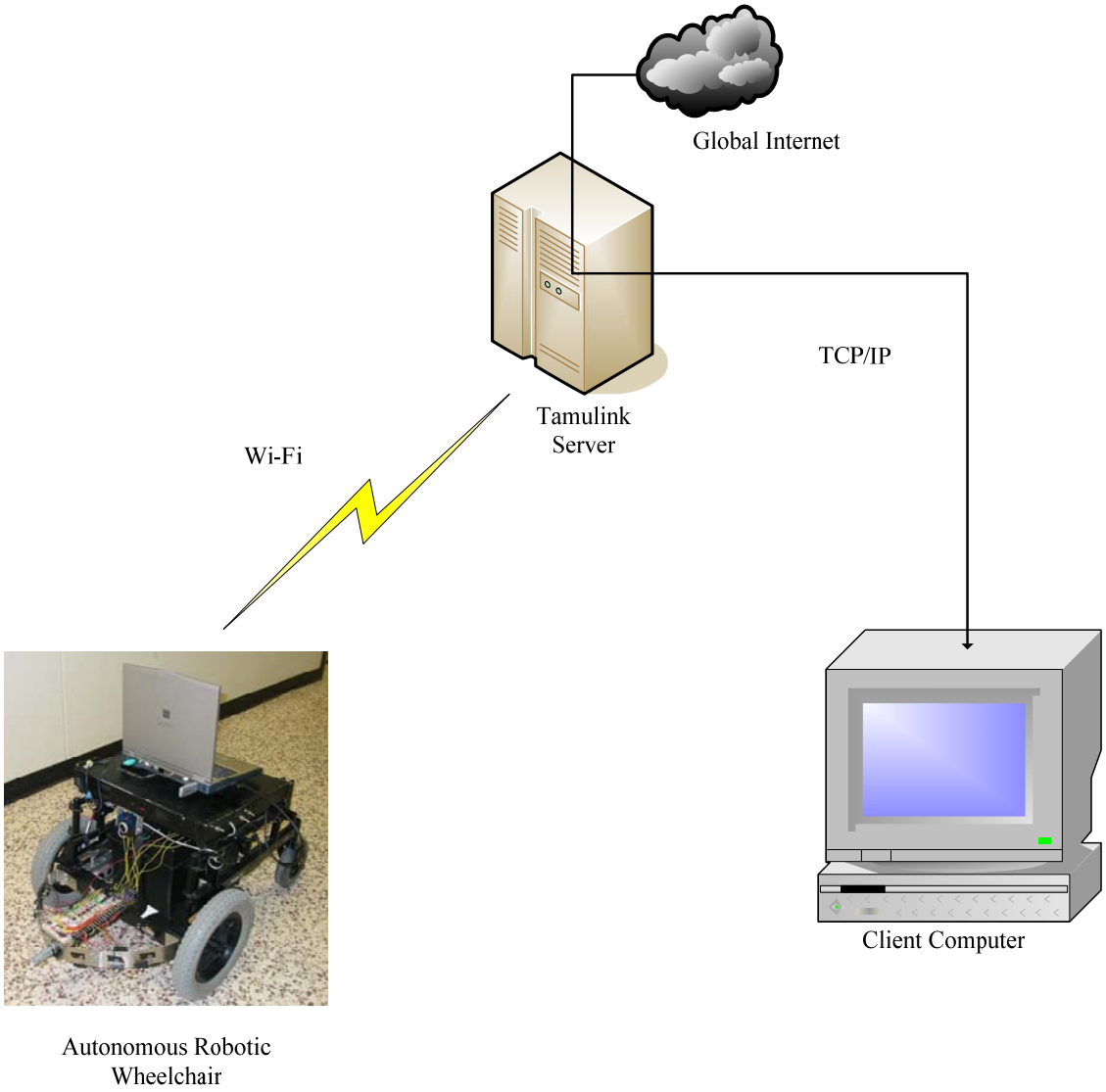



Figure 6.4 Remote control through Internet.

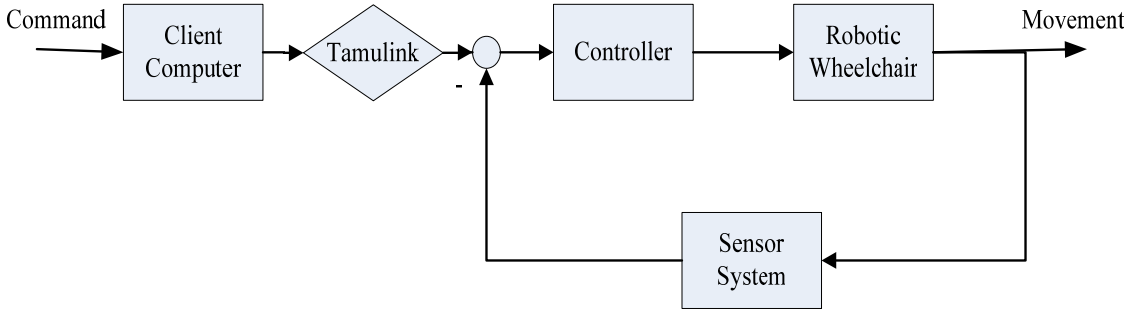



Figure 6.5 Schematic of remote control through the Internet.

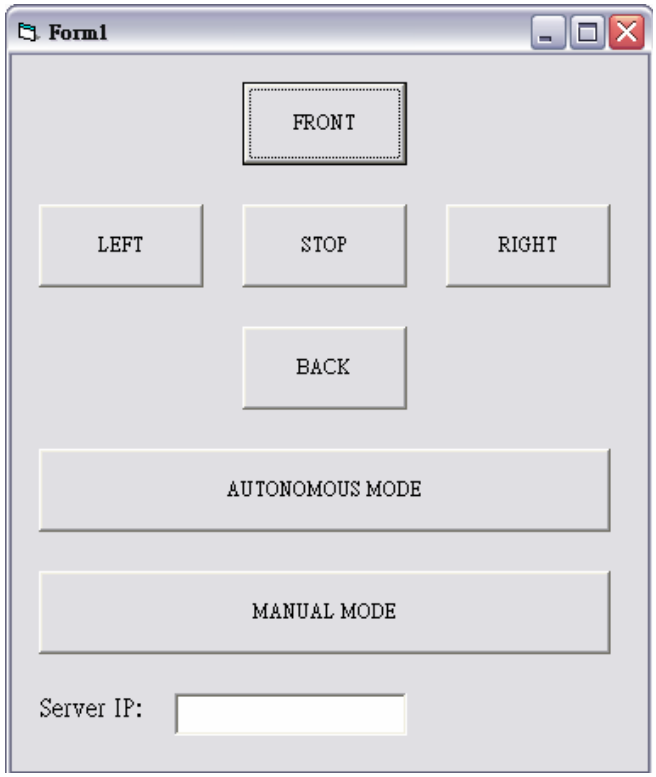



Figure 6.6. Interface of the client side program.

# CHAPTER VII

# CONTROL SYSTEM DESIGN

A key remaining issue of the autonomous robotic wheelchair is developing the control system. The design of the control system includes the sensor system and the speeds of two motors. Section 7.1 describes the main structure of the control system. Section 7.2 describes how the sensor system interacts with the control system. In Section 7.3, the speed control of the speeds of two motors is described.

## 7.1 The Structure of the Control System

The structure of the control system is shown in Figure 7.1. The client-side computer sends the command signals to the control program running on the laptop on the robotic wheelchair. The signals from the seven photocells and the five infrared sensors are input to the laptop through the six input channels of the PCMDIO data-acquisition card with a 10 Hz sampling frequency. The control program generates the output signals to the MC-7 motor controllers through the output channels of the PCMDIO data-acquisition card and the interface circuits. The MC-7 motor controllers generate pulse-width-modulation (PWM) signals to the left-side and right-side motors. The Hall-effect sensors generate the pulses by the rotations of the two motors, which are feedback to the control program through the input channels of PCMDIO data-acquisition card.

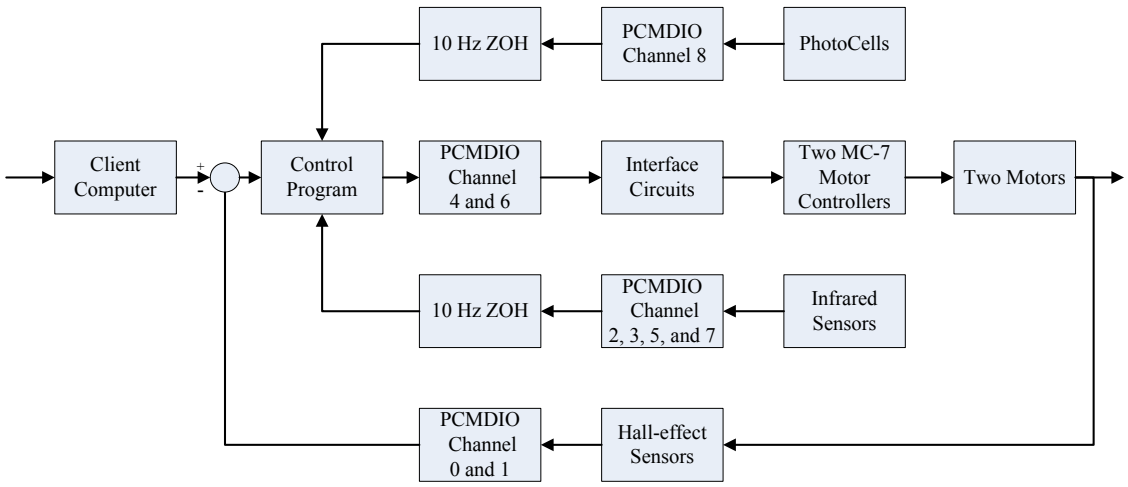



Figure 7.1. Structure of the control system.

## 7.2 The Sensor in the Control System

As described in section 7.1, there are three kinds of sensors in the sensor system: infrared sensors, photocells, and Hall-effect sensors. How these sensors are interfeced with the control system is described below.

Figure 7.2 shows the infrared sensors in the control system. The signals from the five infrared sensors are input to the control program through the input channels of the PCMDIO data-acquisition card. The sample time interval for the signals from the infrared sensors in the control program was set as 100 ms. It can also be referred as a 10 Hz zero-order holder (ZOH) to the control system.

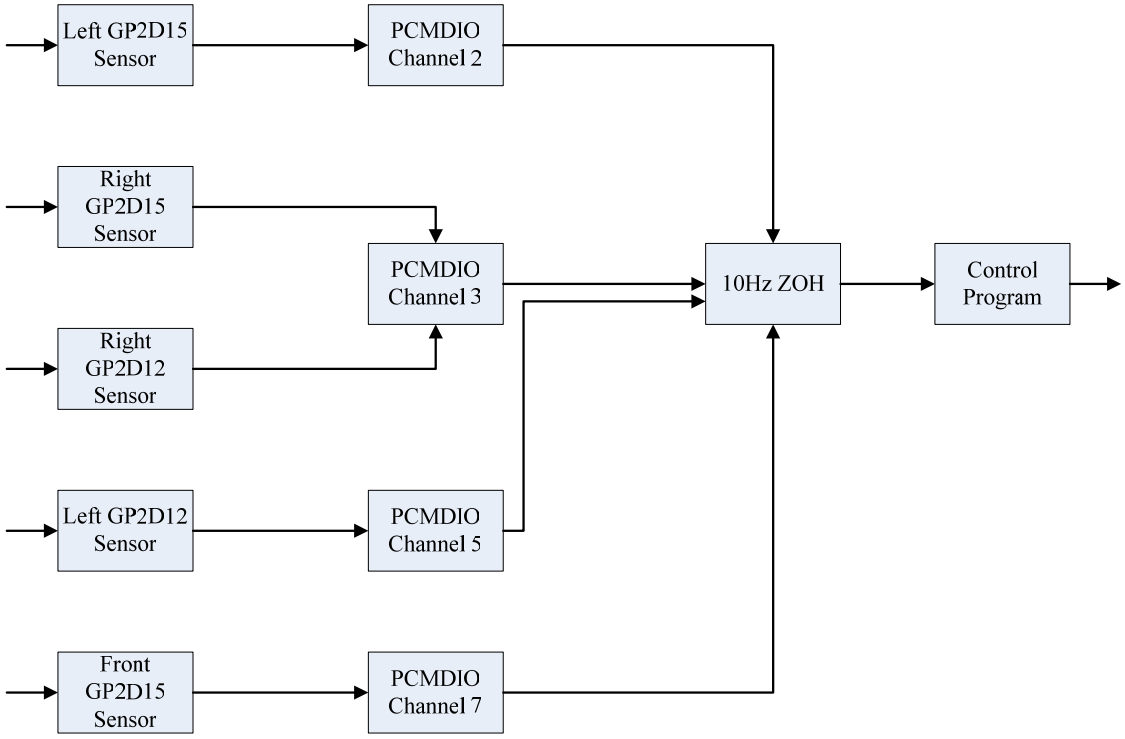



Figure 7.2. Interfacing infrared sensors with the control system.

Figure 7.3 shows how the photocells are interfaced with the control system. The signals from the seven photocells are encoded to three-bit data by a 74LS148 priority encoder. The signal from this priority encoder is input to the control program through channel 8 of the PCMDIO data-acquisition card. The sampling interval for the signals for the photocells is set as 100 ms as well.

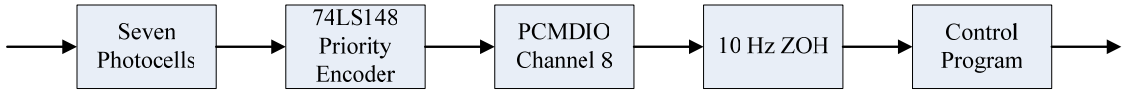



Figure 7.3. Interfacing photocells with the control system.

Figure 7.4 shows how the Hall-effect sensors are interfaced with the control system. The control program generates the output signals to the left- and right-side MC-7 motor controllers through the output channels of the PCMDIO data-acquisition card and the interface circuits. The MC-7 motor controllers generate the PWM signals to the left and right side motors. The Hall-effect sensors mounted on the left- and right-side motors generate the pulses by the rotations of the two motors. The pulses are input to the control program through the input channels of the PCMDIO data-acquisition card. The pulses from the two Hall-effect sensors can be used to record the motion path of the robotic wheelchair and to adjust the speeds of the two driving wheels.

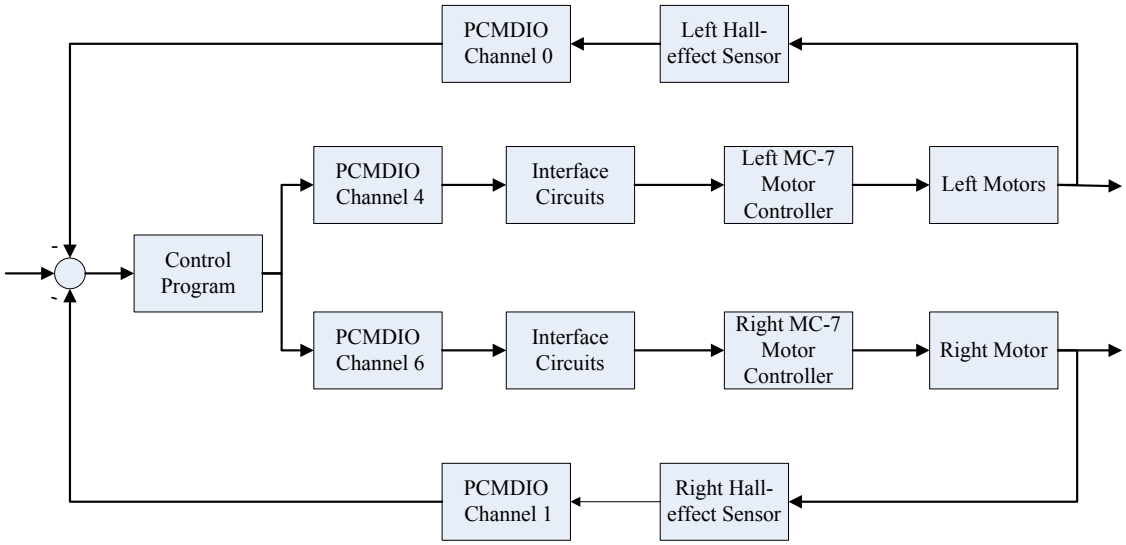



Figure 7.4 Interfacing Hall-effect sensors with the control system.

### 7.3 Controlling the Wheel Speed

To control the speeds of the two driving wheels of the robotic wheelchair, the MC-7

motor controllers are used to generate the PWM signals to the two motors. However, the PWM signals from the MC-7 motor controllers are not exactly the same while the same control voltage connected to the pins T13 of the MC-7 controllers. The distance of the two driving wheels is 57.5 cm and the diameter of the wheel is 31.75 cm. Although the speed difference of the two driving wheels is only 5%, the wheelchair will move approximately $16^{\circ}$ to one side while it only moving for three meters. The $16^{\circ}$ error is too large to implement the real-time path-planning. It is necessary to make the robotic wheelchair move as straight as possible. The development in the PMLR is used the Hall-effect sensors' signals to design a feedback controller to let the wheelchair move near straight. However, the resolution of the Hall-effect sensor is $120^{\circ}$. Although feedback controller in the PMLR can work, the wheelchair would have a significant vibration and the moving trajectory in not smooth [2].

To adjust the speed of the two driving wheels, it is necessary to measure the duty-ratio of the PWM signals generated from the two MC-7 motor controllers with various control voltage to pins T13. Figure 7.5 and Table 7.1 show the results of the duty-ratio of the PWM signals generated by the two MC-7 motor controllers measured by an oscilloscope.

From Table 7.1 and Figure 7.5, we can see that the difference in the duty-ratios of the two MC-7 motor controllers is approximately 1.2%. For the better resolution in the duty-ratio, it can be seen that the control voltage should be adjusted below 0.01 V. However, it is practically impossible to adjust the control voltage below 0.01 V precisely.

Table 7.1. Duty-ratio of the PWM signals generated from two MC-7 motor controllers.

| Control voltage to pin T13 of MC-7 motor controllers | Duty-ration of the left MC-7 motor controller | Duty-ration of the right MC-7 motor controller |
|---|---|---|
| 1.55V | 24.9% | 23.7% |
| 1.56V | 25.2% | 24.1% |
| 1.57V | 25.6% | 24.4% |
| 1.58V | 25.9% | 24.7% |
| 1.59V | 26.1% | 25.2% |
| 1.60V | 26.7% | 25.4% |
| 1.61V | 27.0% | 25.9% |
| 1.62V | 27.4% | 26.1% |
| 1.63V | 27.9% | 26.5% |
| 1.64V | 28.3% | 26.9% |
| 1.65V | 28.6% | 27.3% |
| 1.66V | 29.1% | 27.7% |
| 1.67V | 29.5% | 28.1% |
| 1.68V | 29.9% | 28.5% |
| 1.69V | 30.4% | 28.9% |
| 1.70V | 30.9% | 29.3% |
| 1.71V | 31.2% | 29.6% |
| 1.72V | 31.7% | 30.1% |
| 1.73V | 32.1% | 30.5% |
| 1.74V | 32.5% | 30.9% |
| 1.75V | 33.0% | 31.2% |

The method to making the autonomous robotic wheelchair move in a near straight and

smooth path is adding a 10Ω~200Ω potentiometer on the interface board. This potentiometer can adjust the control voltage to the-left side MC-7 motor controller from 1.60 V to 1.69 V. From the experiments result, by adjusting the left control voltage to 1.62 V, the difference of the two wheels can be reduced to approximately 1%. The experimental data are shown in Table 7.2.

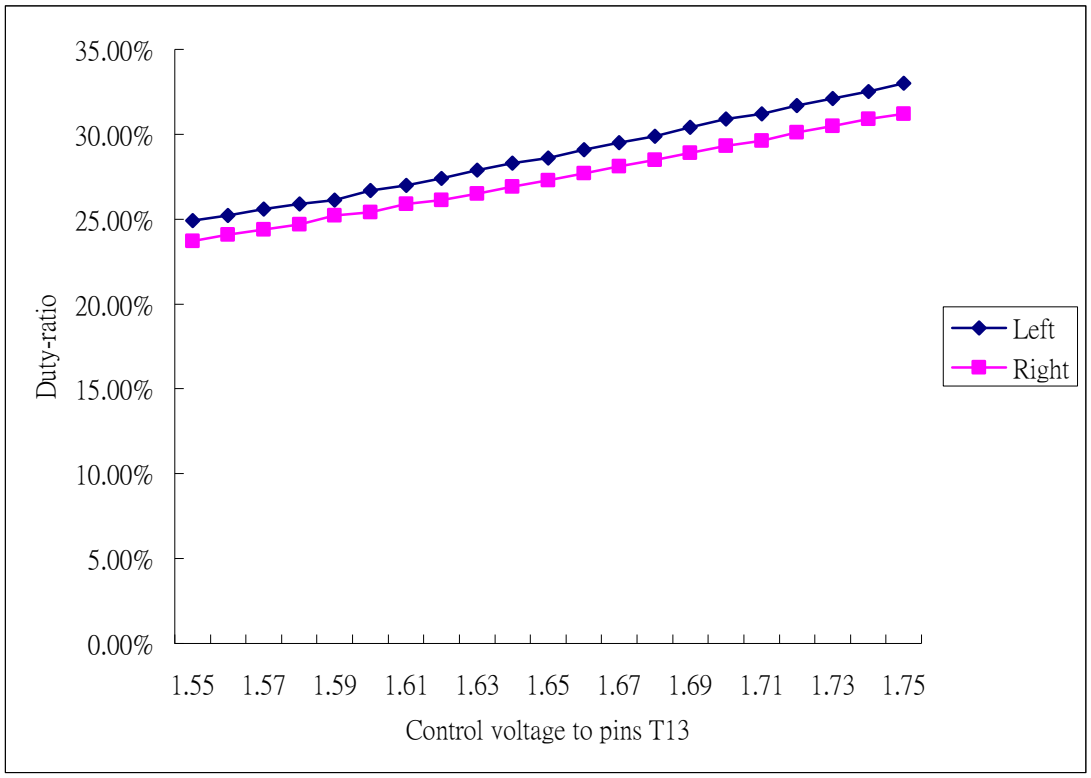



Figure 7.5. Statistic chart of the control voltage and duty-ratio.

Table 7.2. Experimental data for the two driving wheels.

| Control voltage to pin T13 of the right MC-7 motor controller | Control voltage to pin T13 of the left MC-7 motor controllers | Pulses counted by the right Hall-effect sensor in one minute | Pulse counted by the left Hall-effect sensor in one minute |
|---|---|---|---|
| 1.66 V | 1.61 V | 1228 | 1202 |
| 1.66 V | 1.62 V | 1284 | 1302 |
| 1.66 V | 1.63 V | 1324 | 1380 |
| 1.66V | 1.64V | 1412 | 1482 |

In this thesis research, the control voltage to the left wheel was set as 1.62 V and the right wheel is set as 1.66 V. From Table 7.2, the velocity *V* of the center between the two driving wheels can be obtained as $\frac{1284+1302}{2}=1293$ cm/min = 0.21 m/s from Table 7.2. An experimental path of the robotic wheelchair moving for two meters recorded by the pulses counted by the Hall-effect sensors is shown in Figure 7.6. It can be seen that the robotic wheelchair were moving in a near straight path. The experimental method will be described in the next chapter.

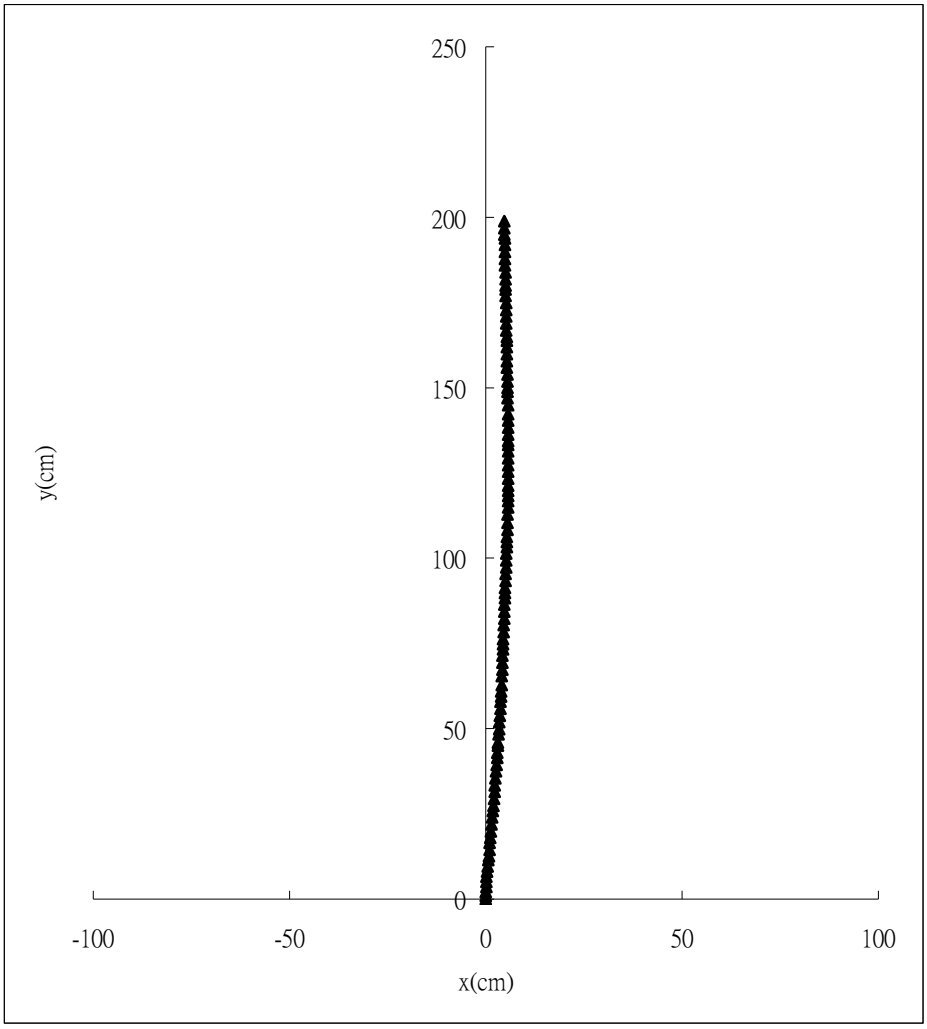



Figure 7.6. An experimental path of the robotic wheelchair moving two meters.

# CHAPTER VIII

# OPERATION AND TESTING

This autonomous robotic wheelchair research contains three major design components: hardware design, interface design, and real-time path-planning algorithm design. This chapter describes how these three design components are combined together to make the autonomous robotic wheelchair move in an unknown environment with collision-avoidance navigation. Section 8.1 describes the typical autonomous and manual operation modes. Section 8.2 describes the experiments and testing results of the motion trajectory with real-time path-planning.

## 8.1 Operation

As described in Chapter VI, the control software provides two modes (autonomous and manual) of the robotic wheelchair. When the user turns on the robotic wheelchair, it runs in the autonomous mode by default unless the user switches to the manual mode.

### 8.1.1 Autonomous Mode

While the robotic wheelchair running in the autonomous mode, it keeps moving forward until an obstacle is detected by the any of five infrared sensors or a specific light is detected by any of the seven photocells. The algorithm of the autonomous mode can be referred to Figure 6.3. If there is any obstacle detected by any of the infrared sensors, the robotic wheelchair will react according to the Table 5.1. This function allows the robotic wheelchair to perform the collision-avoidance navigation. If the robotic wheelchair detects the specific light by any of the seven photocells, it moves toward the specific light,

and stops there. The specific light is considered as the final target. According to the real time path-planning algorithm, all the trajectories are generated in real time by the path-planning algorithm described in Figure 6.3 without any predefined route.

### 8.1.2 Manual Mode

The manual mode allows the user to control the robotic wheelchair manually. It provides five functions: front, back, right, left, and stop. Those functions allows the user to control the robotic wheelchair to move forward, move backward, turn left (counter-clockwise), turn right (clockwise), and stop any time. The manual mode also allows the user to control the robotic wheelchair manually when it stops in a dead zone. While the user operating the robotic wheelchair in the manual mode, the autonomous mode is disabled.

## 8.2 Experiments and Testing

To implement the real-time path-planning algorithm, it is necessary to ensure that the motion trajectory of the robotic wheelchair turning around the middle point of its axle be a perfect circle. This motion trajectory can be recorded by the Hall-effect sensors and converted to a two-dimensional trajectory in the *xy*-plane.

### 8.2.1 Recording the Motion Trajectory

The motion trajectory can be recorded by the Hall-effect sensors and converted to an *xy*-plane coordinate system. The gear-ratio of the driving wheel is found as 32:1 and the resolution of the Hall-effect sensor is $120^{\circ}$, so that there are 96 pulses for one revolution. The circumference of the driving wheel is approximately 100 cm, and one pulse

represents closely 1 cm of the wheel moving on the ground if there is no skid. The difference of pulses counted by the left- and right-side Hall-effect sensors represents closely $1^\circ$ of the turning angle of the robotic wheelchair in PMLR's research [2].

Assume that the robotic wheelchair starts at the $(x_0, y_0) = (0,0)$ point in the *xy*-plane. Then, set the sampling interval as 100 ms for the pulses counted by the Hall-effect sensors. Defined the pulse counted by the left-side Hall-effect sensor at the sampling period $i$ is $LH_i$, and that counted by the right-side Hall-effect sensor is $RH_i$, where $i = 1,.....,n$. The displacement $d_i$ of the robotic wheelchair from $(x_{i-1}, y_{i-1})$ to $(x_i, y_i)$ is $\frac{LH_i + RH_i}{2} - \frac{LH_{i-1} + RH_{i-1}}{2}$ cm and the turning angle is $\theta_i = (LH_i - RH_i)^\circ$. In the *xy*-plane, the position of the robotic wheelchair is $x_i = x_{i-1} + d_i \sin\theta_i$, $y_i = y_{i-1} + d_i \cos\theta_i$. An illustration of the motion path recording method is shown in Figure 8.1.

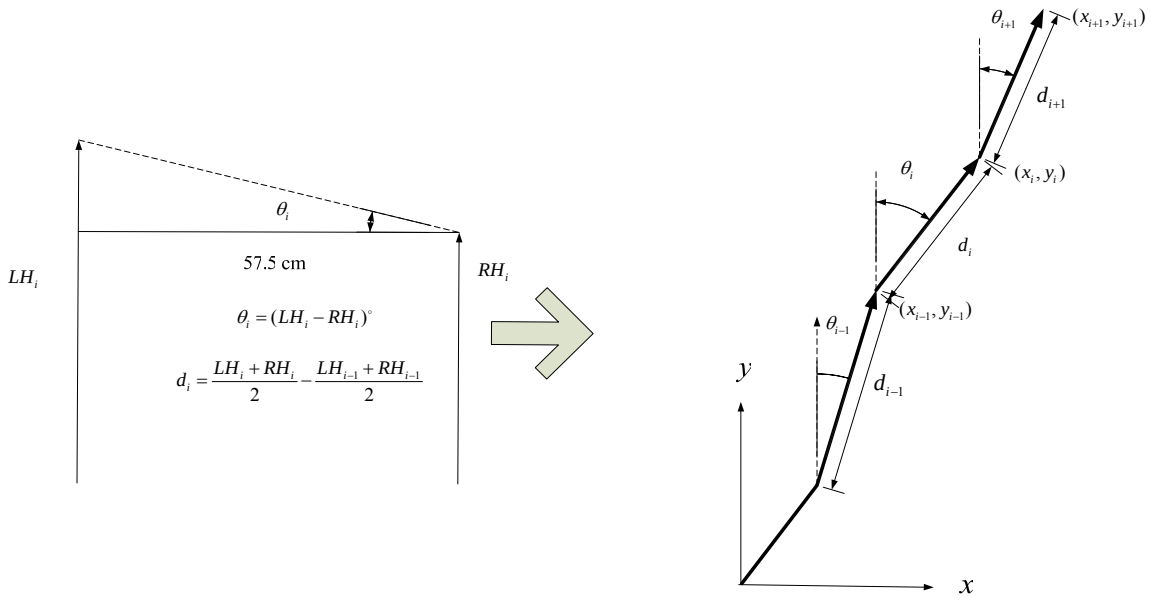



Figure 8.1. Recording the motion trajectory.

### 8.2.2 Robotic Wheelchair Rotating around the Axle Middle Point

Theoretically, if one of the driving wheels moves forward and the other moves backwards at the same speed, the robotic wheelchair will turn around the middle point of its axle. The motion trajectory is a circle when it turns $360^\circ$, and the diameter of this circle is the distance of the two wheels, which is 57.5 cm. The circumference of the circle is $57.5 \times 2\pi = 361.28$ cm. However, the speeds of the two driving wheels are not exactly the same and the driving wheels may skid on the ground, therefore the position of the middle point of the axel will not be fixed and the motion trajectory is not a perfect circle.

To record the motion trajectory, the Hall-effect sensors can be used. Set the sampling interval as 100 ms for the pulses counted by the Hall-effect sensors. In Figure 8.2, the position of the axle middle point $O_i(x_{i,} y_i)$ at the sampling period $i$ in *xy*-plane can be found by the method described in Section 8.2.1. Notice that $d_i = \frac{-LH_i + RH_i}{2} - \frac{-LH_{i-1} + RH_{i-1}}{2}$ if it turns counter-clockwise and $d_i = \frac{LH_i - RH_i}{2} - \frac{LH_{i-1} - RH_{i-1}}{2}$ if it turns clockwise. At the sampling period *i,* while one of the driving wheels is moving forward and the other moving backward, the turning angle $\alpha_i$ from the wheel to the axle middle point can also be found by the pulse $LH_i$ and $RH_i$. While the robotic wheelchair is turning $360^\circ$ around the middle point of the axle, the circumference of the motion trajectory circle is 361.28 cm. Since one pulse represents 1 cm of the motion of the wheel, the turning angle can be represented as $\alpha_i = (LH_i + RH_i)^\circ$, and the radius $r = 28.75$ (cm) is the distance from the driving wheel

to the axle middle point. The point $(\overline{x}_i, \overline{y}_i)$ represents the position of the driving wheel at the sampling period $i$ may be defined as $\overline{x}_i = x_i + r\cos\alpha_i$, $\overline{y}_i = y_i + r\sin\alpha_i$.

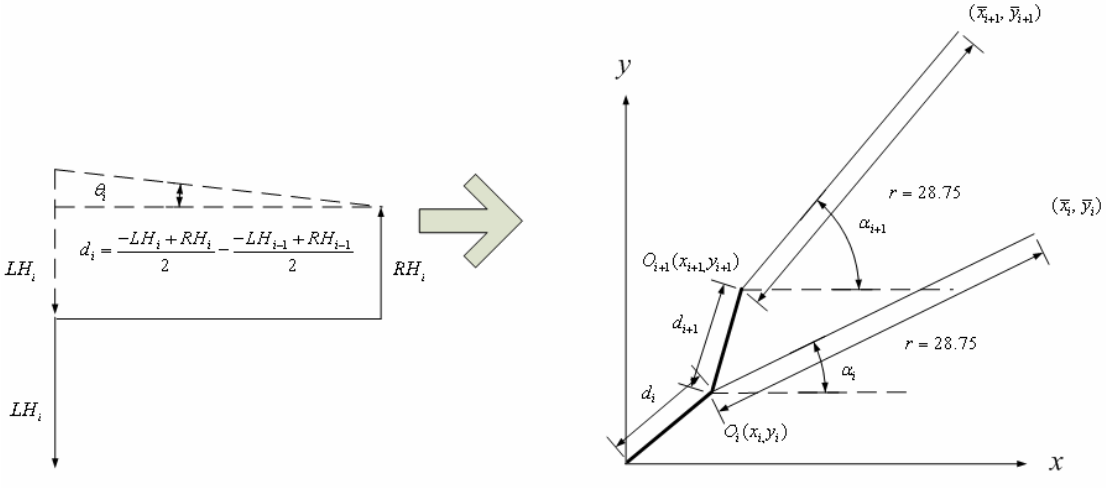



Figure 8.2. Recording the motion trajectory while one of the driving wheels are moving forward and the other are moving backward at time *i*.

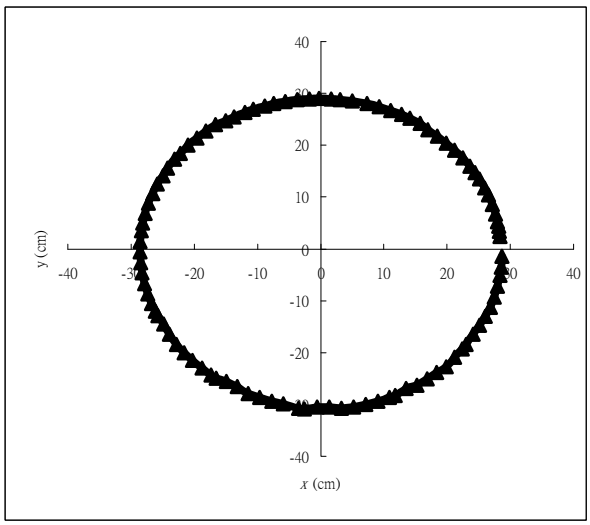



Figure 8.3. Motion trajectory of the two driving wheels.

The experimental measurement of the motion trajectory of the two driving wheels while the robotic wheelchair is turning $360^{\circ}$ is shown in Figure 8.3. It can be seen that the motion trajectory of the two driving wheels is nearly a circle. Therefore it is approximated that the robotic wheelchair turns around the middle point of its axle.

### 8.2.3 Motion Trajectory of the Robotic Wheelchair in an Unknown Environment

The robotic wheelchair can move in an unknown environment with real-time path-planning with collision avoidance navigation. While the robotic wheelchair is turning clockwise or counter-clockwise by an angle $\alpha$ around the middle point of its axle, the body-fixed coordinate system in the *xy*-plane also rotates by an angle $\alpha$. Figure 8.4 shows the robotic wheelchair moving to point $(x_{n,0}, y_{n,0})$, turning clockwise for an angle $\alpha_1$, the basis coordinate rotates to $(\tilde{x}_1, \tilde{y}_1)$, the sampling period $i$ resets to 0, and moves to the point $(x_{i,1}, y_{i,1})$. At the sampling period $i$, the point $(x_{i,1}, y_{i,1})$ relation to the basis coordinate $(\tilde{x}_0, \tilde{y}_0)$ can be found as $\begin{cases} x_{i,1} = x_{i-1,1} + d_{i,1}\sin(\theta_{1,0} + \alpha_1 + \theta_{i,1}) \\ y_{i,1} = y_{i-1,1} + d_{i,1}\cos(\theta_{1,0} + \alpha_1 + \theta_{i,1}) \end{cases}$,

$\begin{cases} x_{n,0} = x_{0,1} \\ y_{n,0} = x_{0,1} \end{cases}$ where $d_{i,1} = \dfrac{LH_{i,1} + RH_{i,1}}{2} - \dfrac{LH_{i-1,1} + RH_{i-1,1}}{2}$, $\theta_{i,1} = (LH_{i,1} - RH_{i,1})^{\circ}$,

$i = 1, ....., n$, $i \in N$. Furthermore, if the robotic wheelchair moves to the basis coordinate $(\tilde{x}_j, \tilde{y}_j)$, the position $(x_{i,j}, y_{i,j})$ can be found as

$\begin{cases} x_{i,j} = x_{i-1,j} + d_{i,j}\sin(\sum_{p=0}^{j-1}\theta_{n,p} + \sum_{q=0}^{j}\alpha_q + \theta_{i,j}) \\ y_{i,j} = y_{i-1,j} + d_{i,j}\cos(\sum_{p=0}^{j-1}\theta_{n,p} + \sum_{q=0}^{j}\alpha_q + \theta_{i,j}) \end{cases}$, $\begin{cases} x_{n,j-1} = x_{0,j} \\ y_{n,j-1} = x_{0,j} \end{cases}$, $\theta_{i,j} = (LH_{i,j} - RH_{i,j})^{\circ}$,

$d_{i,j} = \frac{LH_{i,j} + RH_{i,j}}{2} - \frac{LH_{i-1,j} + RH_{i-1,j}}{2}$, $i = 1, ....., n$, $j = 0, ....., m$, $i, j \in N$. Where $(x_{i,j}, y_{i,j})$ represents the position to the basis coordinate $(\tilde{x}_0, \tilde{y}_0)$. At the sampling period $i$ in the basis coordinate $(\tilde{x}_j, \tilde{y}_j)$, $\theta_{i,j}$, $d_{i,j}$, $LH_{i,j}$, $RH_{i,j}$ represent the small turning angle, small displacement from the last position, and pulses counted from the Hall-effect sensors in the basis coordinate $(\tilde{x}_j, \tilde{y}_j)$. $(x_{n,j-1}, y_{n,j-1})$ represents the last position in the basis coordinate $(\tilde{x}_{j-1}, \tilde{y}_{j-1})$, $\theta_{n,j-1}$ represents $\tan^{-1}\frac{x_{n,j-1}}{y_{n,j-1}}$, and $\alpha_j$ represents the total turning angle around its axle middle point at $(x_{n,j-1}, y_{n,j-1})$, $\alpha_j$ can be measured by the method described in Section 8.2.2. The illustration of this motion path recording method is shown in Figure 8.4.

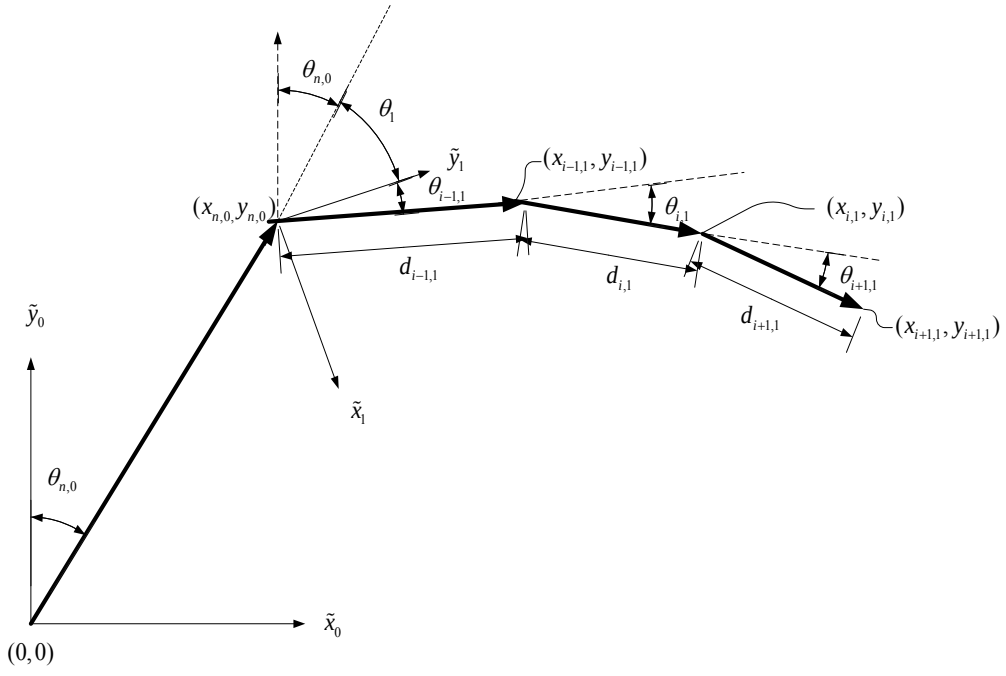



Figure 8.4. The robotic wheelchair moves in different basis coordinates in the *xy*-plane.

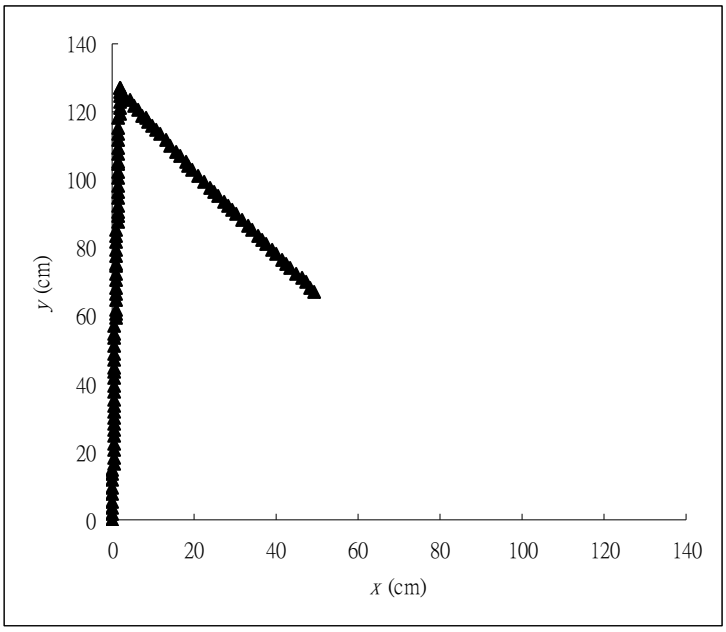



Figure 8.5. Motion trajectory of robotic wheelchair turned clockwise for approximately $140^{\circ}$.

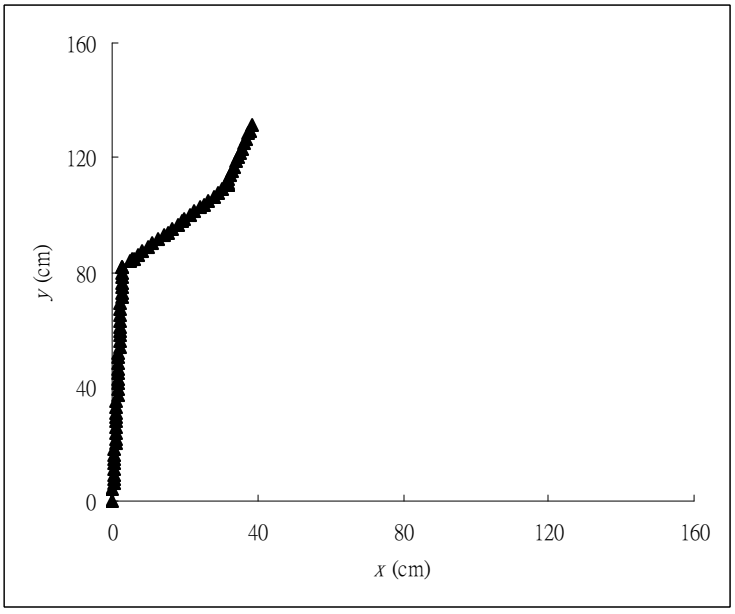



Figure 8.6. Motion trajectory of the robotic wheelchair tracking a specific light.

By this motion trajectory recording method, the real-time path-planning can be recorded as Figures 8.5, 8.6. Figure 8.5 shows the motion trajectory of robotic wheelchair moved forward for approximately 120 cm and turned clockwise for approximately $140^{\circ}$. Figure 8.6 shows the motion trajectory while it tracking a specific light which was described in Section 5.4. The robotic wheelchair detected the light on the right side, it turned clockwise to the right, and turned counter-clockwise after approximately 45 cm to correct the error automatically. Figure 8.7 shows the motion trajectory of the robotic wheelchair moving in real-life testing environment with collision-avoidance navigation recorded by the long-term-exposure photography technique. A lamp was mounted on the robotic wheelchair. Figure 8.8 shows the same motion trajectory of the robotic wheelchair as Figure 8.7.

Figure 8.7. Motion trajectory of the robotic wheelchair recorded by the long-term-exposure photography technique.

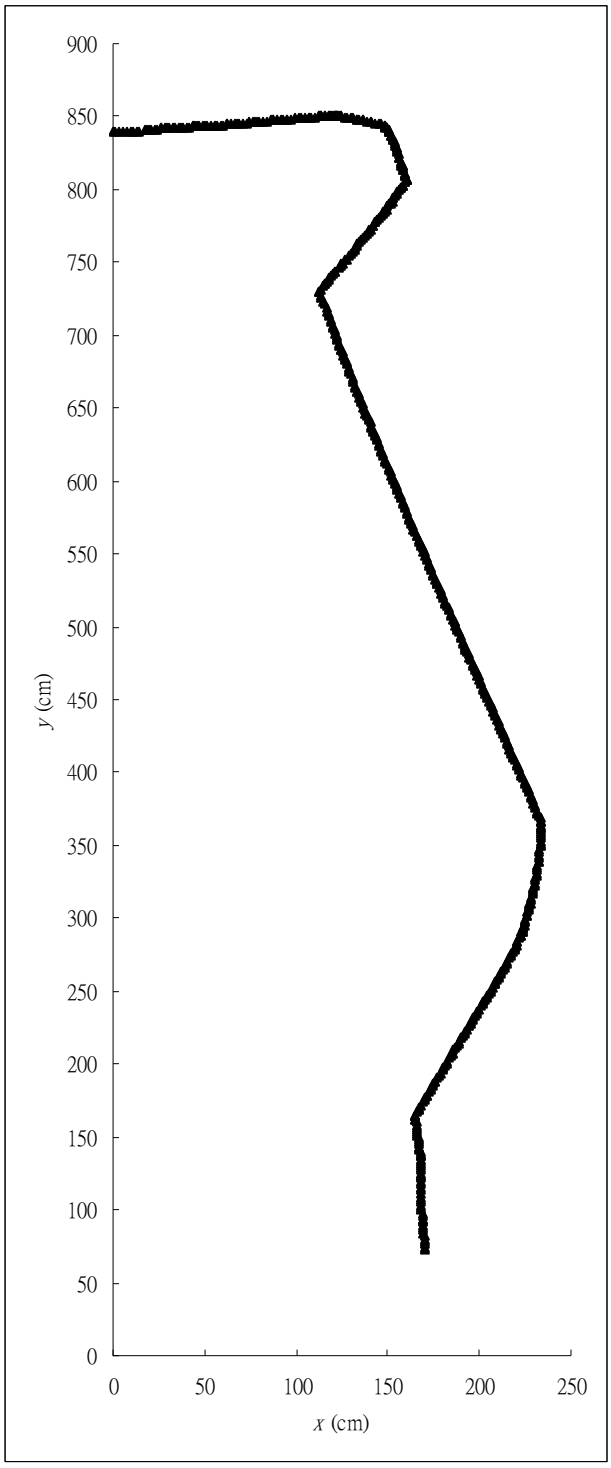



Figure 8.8. Motion trajectory of the robotic wheelchair moving in a real-life testing environment.

From Figure 8.7 and 8.8, it can be seen that the motion trajectory recorded by the Hall-effect sensor is very close to the long-term-exposure photograph. It can be seen that the motion trajectory on Figure 8.7 is a smoother path. A possible reason is that the speeds of the two driving wheels were not exactly the same and there was skidding.

The testing environment for this research is in the ground-floor hallway and Precision Mechatronics Lab inside the Zachery Engineering Center of Texas A&M University. This testing result demonstrates that the robotic wheelchair can move in an unknown environment located in a normal building. Figures 8.9-8.11 shows sequence photos of the robotic wheelchair during testing.

Figure 8.9. The robotic wheelchair starts at point (0,840).

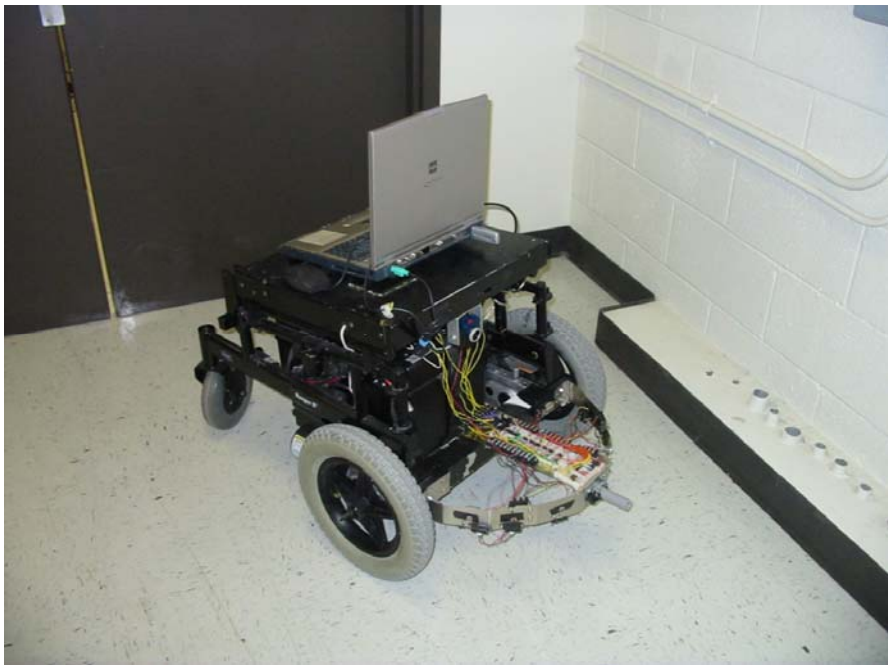


Figure 8.10. The robotic wheelchair turned at point (148,842).

Figure 8.11. The robotic wheelchair turned at point (160,806).

# CHAPTER IX

# CONCLUSIONS AND SUGGESTED FUTURE WORK

The autonomous robotic wheelchair was successfully constructed and met the objective. Section 9.1 summarizes the accomplishments of the thesis. Section 9.2 discusses the current limitations of the autonomous robotic wheelchair. In Section 9.3, future work is proposed to enhance the functionality of the autonomous robotic wheelchair and overcome the current limitations.

## 9.1 Conclusions

The autonomous robotic wheelchair has met the objectives. The robotic wheelchair could move in an unknown environment with real-time path planning. The generation of a real-time path was implemented by detecting the range from the obstacles, and by tracking specific lights sources which is used as a beacon. The infrared sensors were used to detect the distance to the obstacles, and the light-variance resistors were used to track the specific light source.

To optimize the motion trajectory, the circuits to the motor controller were modified to ensure the wheelchair can turns in a minimum turning radius. The robotic wheelchair could turn around the center point of the axle. The algorithm of the real-time path planning of the robotic wheelchair was simplified. Combined with the newly developed of Internet-connection capability, the robotic wheelchair could move in an unknown environment with collision avoidance navigation.

## 9.2 Limitations

The autonomous robotic wheelchair in its current form has the following limitations.

1. The speeds of two driving wheels are not exactly the same and the autonomous robotic wheelchair cannot move in a straight line. Even using the feedback controller by the pulses from the Hall-effect sensors, it is impossible to adjust the control voltage to the motor controllers precisely.

2. The main limitation of the robotic wheelchair is that the PCMDIO data-acquisition card has digital I/O capability alone. All signals from the sensors need to be converted to digital signal through ADCs. The GP2D12 infrared sensor generates different analog signals according to the distance from the obstacle. Without the analog I/O capability, the robotic wheelchair can only detect the obstacles in a fixed range. It cannot measure the precise distance from the obstacles. The control program and real-time path-planning algorithm can only be designed by this digital input signals. Other control laws such as optimal controller to ensure the robotic wheelchair to move in an optimal path cannot be implemented.

3. The laptop tends to overheat, which causes it unstable. The wireless adapter does not have good performance in receiving Wi-Fi signal.

4. The robotic wheelchair has no sensor at backside, while it moving to the dead zone it

can only set to stop and cannot moving backward.

### 9.3 Suggested Future Work

The following are proposed as future work to enhance the functionality of the autonomous robotic wheelchair and overcome the current limitations.

1. Use of the controller with analog I/O capability, such as digital-signal-processor (DSP) board. With the analog I/O capability, we could implement other real-time path-planning algorithms which might have better performance. The PWM signals could be directly generated from the DSP, and we could adjust the duty-ratio at the same.

2. Adding the optical encoders on the two driving wheels instead of the Hall-effect sensors. The optical encoders have much better resolution than the Hall-effect sensors. By the signal from the optical encoders, a better feedback controller can be designed to ensure the robotic wheelchair to move in a straight line.

# REFERENCES

[1] R. Homji, *Intelligent Pothole Repair Vehicle*, M.S. thesis, Texas A&M University, 2005.

[2] A. Rogers, *Precision Mechatronics Lab Robot Development*, M.S. thesis, Texas A&M University, 2007.

[3] T. J. A. de Vries, C. v. Heteren, and L. Huttenhuis, "Modeling and Control of a Fast Moving, Highly Maneuverable Wheelchair," in *Proceedings of the International Biomechatronics Workshop,* pp. 110−115, Apr. 1999.

[4] A. Argyros, P. Georgiadis, P. Trahanias, and D. Tsakiris, "Semi-Autonomous Navigation of a Robotic Wheelchair," *Journal of Intelligent and Robotic Systems,* vol. 34, no. 3, pp. 315−329, 2002.

[5] C. H. Kuo, H. L. Huang, and M. Y. Lee, "Development of Agent-Based Autonomous Robotic Wheelchair Control Systems," *Journal of Biomedical Engineering - Applications, Basis, Communications,* vol. 15, no. 6, pp. 12−23, Dec. 2003.

[6] D. Bank, "A High-Performance Ultrasonic Sensing System for Mobile Robots," in *ROBOTIK 2002: Leistungsstand, Anwendungen, Visionen, Trends. VDI-Berichte Nr. 1679,* pp. 557−564, Jun. 2002.

[7] D. H. Shim, H. Chung, and S. S. Sastry, "Conflict-Free Navigation in Unknown Urban Environments," *IEEE Robotics & Automation Magazine,* vol. 13, pp. 27−33, Sep. 2006.

[8] D. Cruz, J. McClintock, B. Perteet, O. A. A. Orqueda, Y. Cao, and R. Fierro, "Decentralized Cooperative Control - A Multivehicle Platform for Research in

Networked Embedded Systems," *IEEE Control Systems Magazine,* vol. 27, no. 3, pp. 58−78, Jun. 2007.

[9] W. Ren, R. W. Beard, and E. M. Atkins, "Information Consensus in Multivehicle Cooperative Control," *IEEE Control Systems Magazine,* vol. 27, no. 2, pp. 71−82, Apr. 2007.

[10] H. Şahin and L. Gűvenc, "Household Robotics: Autonomous Devices for Vacuuming and Lawn Mowing," *IEEE Control Systems Magazine,* vol. 27, no. 2, pp. 20−96, Apr. 2007.

[11] J. Laumond, "Robot Motion Planning and Control," *Lecture Notes in Control and Information Science 229*. Berlin: Springer.

[12] *Superlogics PCMDIO Users Manual*. Available at SuperLogics, Inc. 300 Third Avenue, Waltham, MA 02451, USA.

[13] *Superlogics PCMDRIVE® Data Acquisition Software User's Manual*. Available at SuperLogics, Inc. 300 Third Avenue, Waltham, MA 02451, USA.

# APPENDIX A

# OPERATING PROGRAM

```
Dim IFR As Byte

Dim LIFR As Byte

Dim RIFR As Byte

Dim LIFRS As Byte

Dim PCL As Byte

Dim mintStatus As Integer


Dim bytRippleLeft As Integer

Dim bytRippleRight As Integer

Dim bytHallLeft As Byte

Dim bytHallRight As Byte

Dim blnRipCntLeft As Boolean

Dim blnRipCntRight As Boolean

Dim mintStatus1 As Integer

Dim mintStatus2 As Integer

Dim PrePulseR As Boolean

Dim PrePulseL As Boolean

Dim ActDist As Single

Dim StopTime As Single

Dim CarryDeg As Integer
```

```
Dim NeutralMC As Byte



Dim R_Hall As Integer
Dim L_Hall As Integer
Dim i As Integer
Dim blnRun As Boolean

Public Sub Initialize()
  Timer1.Enabled = True
  Timer2.Enabled = True
  Timer3.Enabled = True
  Timer4.Enabled = False
    blnRun = True
    StopTime = 10000000

    intStatus = 0
    OldDist = 0
    RbytRipple = 0
    RbytRipple = 0
    Text1.Text = 0
    Text2.Text = 0
```

```
End Sub



Public Sub HallsRead()
    Do
    DoEvents

      mintStatus1 = singleDigitalInput(gintlogicaldevice, 0, bytHallLeft)
      If mintStatus1 <> 0 Then
            Call errorMessage(mintStatus1)
            Call PCMCloseDeviceVB(gintlogicaldevice)
            End
        End If

        mintStatus2 = singleDigitalInput(gintlogicaldevice, 1, bytHallRight)
        If mintStatus2 <> 0 Then
            Call errorMessage(mintStatus2)
            Call PCMCloseDeviceVB(gintlogicaldevice)
            End
        End If

    If bytHallLeft > 0 And bytHallLeft < 8 Then
      If blnRipCntLeft = True Then
```

```
            bytRippleLeft = bytRippleLeft + 1
            End If
            blnRipCntLeft = False
            Else
            blnRipCntLeft = True
        End If

            If bytHallRight > 0 And bytHallRight < 8 Then
            If blnRipCntRight = True Then
                bytRippleRight = bytRippleRight + 1
            End If
            blnRipCntRight = False
        Else
            blnRipCntRight = True
        End If

        Text1.Text = bytRippleLeft * 15
        Text2.Text = bytRippleRight * 15
Loop
End Sub

Public Sub SensorRead()
```

```
mintStatus = singleDigitalInput(gintlogicaldevice, 7, IFR)
        If mintStatus <> 0 Then
            Call errorMessage(mintStatus)
            Call PCMCloseDeviceVB(gintlogicaldevice)
            End
        End If
Text3.Text = IFR

mintStatus = singleDigitalInput(gintlogicaldevice, 8, PCL)
        If mintStatus <> 0 Then
            Call errorMessage(mintStatus)
            Call PCMCloseDeviceVB(gintlogicaldevice)
            End
        End If

Text4.Text = PCL

mintStatus = singleDigitalInput(gintlogicaldevice, 2, LIFR)
        If mintStatus <> 0 Then
            Call errorMessage(mintStatus)
            Call PCMCloseDeviceVB(gintlogicaldevice)
            End
        End If
```

```
Text5.Text = LIFR


mintStatus = singleDigitalInput(gintlogicaldevice, 3, RIFR)
        If mintStatus <> 0 Then
            Call errorMessage(mintStatus)
            Call PCMCloseDeviceVB(gintlogicaldevice)
            End
        End If
Text6.Text = RIFR


mintStatus = singleDigitalInput(gintlogicaldevice, 5, LIFRS)
        If mintStatus <> 0 Then
            Call errorMessage(mintStatus)
            Call PCMCloseDeviceVB(gintlogicaldevice)
            End
        End If
Text7.Text = LIFRS



End Sub


Private Sub AUTO_Click(Index As Integer)
```

```
Timer1.Enabled = True

Timer2.Enabled = True

Call SensorRead

Call HallsRead

End Sub


Private Sub MANUAL_Click(Index As Integer)

Timer1.Enabled = False

Timer3.Enabled = False

Timer4.Enabled = False

blnRun = False

          intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)

                If intStatus <> 0 Then

                     Call pcmdioError(gintlogicaldevice, intStatus)

                End If

        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)

                If intStatus <> 0 Then

                     Call pcmdioError(gintlogicaldevice, intStatus)

                End If

End Sub


Private Sub front_Click(Index As Integer)

Timer1.Enabled = False
```

```
Timer3.Enabled = False
Timer4.Enabled = False
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If


If IFR = 1 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If
Call HallsRead
Call SensorRead
```

```
End Sub



Private Sub left_Click(Index As Integer)
  Timer1.Enabled = False
  Timer3.Enabled = False
  Timer4.Enabled = False
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 5)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
 If IFR = 1 Then
          intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
                If intStatus <> 0 Then
```

```
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If
Call HallsRead
Call SensorRead
End Sub


Private Sub right_Click(Index As Integer)
Timer1.Enabled = False
Timer3.Enabled = False
Timer4.Enabled = False
    intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
    intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
If IFR = 1 Then
```

```
            intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
                    If intStatus <> 0 Then
                        Call pcmdioError(gintlogicaldevice, intStatus)
                    End If
            intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
                    If intStatus <> 0 Then
                        Call pcmdioError(gintlogicaldevice, intStatus)
                    End If
End If



Call HallsRead
End Sub

Private Sub back_Click()
Timer1.Enabled = False
Timer3.Enabled = False
Timer4.Enabled = False
            intStatus = singleDigitalOutput(gintlogicaldevice, 4, 5)
                    If intStatus <> 0 Then
                        Call pcmdioError(gintlogicaldevice, intStatus)
                    End If
            intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
```

```
        If intStatus <> 0 Then
            Call pcmdioError(gintlogicaldevice, intStatus)
        End If

 Call HallsRead

End Sub
Private Sub stop_Click(Index As Integer)
Timer1.Enabled = False
Timer3.Enabled = False
Timer4.Enabled = False
blnRun = False
Call Neutral

End Sub



Private Sub Form_Load()
Close #1
gintlogicaldevice = openDevice()
    Call Initialize
    Call SensorRead
```

```
End Sub
Private Sub Form_Unload(Cancel As Integer)
        Call Neutral
        waitTime (100)
        Close #1
          End
  End Sub


Public Sub Neutral()
    intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
    If intStatus <> 0 Then
        Call pcmdioError(gintlogicaldevice, intStatus)
    End If

    intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
    If intStatus <> 0 Then
        Call pcmdioError(gintlogicaldevice, intStatus)
    End If

End Sub


Private Sub Timer1_Timer()
If LIFR = 0 And LIFRS = 0 And RIFR = 0 And IFR = 0 And PCL = 7 Then
```

```
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 0 And LIFRS = 0 And RIFR = 2 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 5)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 0 And LIFRS = 0 And RIFR = 1 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 5)
```

```
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 0 And LIFRS = 0 And RIFR = 3 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 5)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 0 And LIFRS = 1 And RIFR = 0 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
                If intStatus <> 0 Then
```

```
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 0 And LIFRS = 0 And RIFR = 2 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 0 And LIFRS = 1 And RIFR = 1 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
```

```
            End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 0 And LIFRS = 1 And RIFR = 3 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 1 And LIFRS = 0 And RIFR = 0 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
```

```
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 1 And LIFRS = 0 And RIFR = 2 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 1 And LIFRS = 0 And RIFR = 1 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
```

```
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 1 And LIFRS = 0 And RIFR = 3 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 5)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 1 And LIFRS = 1 And RIFR = 0 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
            If intStatus <> 0 Then
```

```
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 1 And LIFRS = 1 And RIFR = 2 And IFR = 0 And PCL = 7 Then
    intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
    intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 1 And LIFRS = 1 And RIFR = 1 And IFR = 0 And PCL = 7 Then
    intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
    intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
```

```
                End If
End If

If LIFR = 1 And LIFRS = 1 And RIFR = 3 And IFR = 0 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 0 And LIFRS = 0 And RIFR = 0 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
```

```
End If

If LIFR = 0 And LIFRS = 0 And RIFR = 2 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 5)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 0 And LIFRS = 0 And RIFR = 1 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 5)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If
```

```
If LIFR = 0 And LIFRS = 0 And RIFR = 3 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 5)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 0 And LIFRS = 1 And RIFR = 0 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If
```

```
If LIFR = 0 And LIFRS = 0 And RIFR = 2 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 0 And LIFRS = 1 And RIFR = 1 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 0 And LIFRS = 1 And RIFR = 3 And IFR = 1 And PCL = 7 Then
```

```
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 1 And LIFRS = 0 And RIFR = 0 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 1 And LIFRS = 0 And RIFR = 2 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
```

```
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 1 And LIFRS = 0 And RIFR = 1 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If LIFR = 1 And LIFRS = 0 And RIFR = 3 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
                If intStatus <> 0 Then
```

```
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 1 And LIFRS = 1 And RIFR = 0 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 1 And LIFRS = 1 And RIFR = 2 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
```

```
            End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 1 And LIFRS = 1 And RIFR = 1 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If LIFR = 1 And LIFRS = 1 And RIFR = 3 And IFR = 1 And PCL = 7 Then
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
```

```
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If



End Sub

Private Sub Timer2_Timer()
Call SensorRead
End Sub

Private Sub Timer3_Timer()
If PCL = 0 Then
Timer1.Enabled = False

        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
                If intStatus <> 0 Then
```

```
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If PCL = 1 Or PCL = 2 Or PCL = 3 Then
Timer1.Enabled = False
    intStatus = singleDigitalOutput(gintlogicaldevice, 4, 5)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
    intStatus = singleDigitalOutput(gintlogicaldevice, 6, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
End If

If PCL = 4 Or PCL = 5 Or PCL = 6 Then
Timer1.Enabled = False
    intStatus = singleDigitalOutput(gintlogicaldevice, 4, 6)
            If intStatus <> 0 Then
                Call pcmdioError(gintlogicaldevice, intStatus)
            End If
    intStatus = singleDigitalOutput(gintlogicaldevice, 6, 5)
```

```
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If PCL = 0 And IFR = 1 Then
Timer1.Enabled = False
        intStatus = singleDigitalOutput(gintlogicaldevice, 4, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
        intStatus = singleDigitalOutput(gintlogicaldevice, 6, 0)
                If intStatus <> 0 Then
                    Call pcmdioError(gintlogicaldevice, intStatus)
                End If
End If

If PCL = 7 Then
Timer1.Enabled = True
End If

End Sub
```

# APPENDIX B

## CLIENT SIDE PROGRAM

This Program is a development from Cheng-Yeh Hsu who is a member in Precision Mechatronics Lab

```
Dim sendData As String
Private Sub AUTO_Click(Index As Integer)
    sock.RemoteHost = txtIP.Text
    sock.RemotePort = "4400"
    sock.Connect
    sendData = "auto"
    sock.sendData sendData
    sock.Close
End Sub

Private Sub MANUAL_Click(Index As Integer)
    sock.RemoteHost = txtIP.Text
    sock.RemotePort = "4400"
    sock.Connect
    sendData = "manual"
    sock.sendData sendData
    sock.Close
End Sub
```

```
Private Sub front_Click(Index As Integer)
    sock.RemoteHost = txtIP.Text
    sock.RemotePort = "4400"
    sock.Connect
    sendData = "auto"
    sock.sendData sendData
    sock.Close
End Sub

Private Sub left_Click(Index As Integer)
    sock.RemoteHost = txtIP.Text
    sock.RemotePort = "4400"
    sock.Connect
    sendData = "left"
    sock.sendData sendData
    sock.Close
End Sub

Private Sub right_Click(Index As Integer)
    sock.RemoteHost = txtIP.Text
    sock.RemotePort = "4400"
    sock.Connect
```

```
        sendData = "right"
        sock.sendData sendData
        sock.Close
End Sub

Private Sub back_Click()
        sock.RemoteHost = txtIP.Text
        sock.RemotePort = "4400"
        sock.Connect
        sendData = "back"
        sock.sendData sendData
        sock.Close
End Sub
Private Sub stop_Click(Index As Integer)
        sock.RemoteHost = txtIP.Text
        sock.RemotePort = "4400"
        sock.Connect
        sendData = "stop"
        sock.sendData sendData
        sock.Close
End Sub
```

# VITA

**Name**: Pin-Chun Hsieh

**Address**: 1F 8-1 Ln 26 Gangqian Rd, Taipei, Taiwan

**Email**: ryanhsieh@seed.net.tw

**Education:**

B.S: Power Mechanical Engineering, National Tsing-Hua University, 2004

M.S: Mechanical Engineering, Texas A&M University, 2008