

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS*

JUDUL:

REMOTED BALBOT

SESI PENGAJIAN: 2010/2011

Saya LEE KONG HAUR (871202-12-5025)
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/Sarjana/Doktor Falsafah)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

SULIT

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

TERHAD

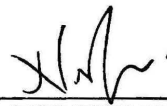
(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

TIDAK TERHAD

Disahkan oleh:



(TANDATANGAN PENULIS)



(TANDATANGAN PENYELIA)

Alamat Tetap:

MDLD 2777 TAMAN PERTAMA SATU
JALAN SILAM 91100 LAHAD DATU
PAHANG

NOR MANIHA BT. ABD GHANI
(Nama Penyelia)

Tarikh: **22 OCTOBER 2010**

Tarikh: : **22 OCTOBER 2010**

CATATAN:

- * Potong yang tidak berkenaan.
- ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
- ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

REMOTED BALBOT

LEE KONG HAUR

**This thesis is submitted in partial fulfilment of the
requirements for the award of the degree of
Bachelor of Electrical Engineering (Electronics)**

**Faculty of Electrical And Electronics Engineering
Universiti Malaysia Pahang**

OCTOBER 2010

“I hereby acknowledge that the scope and quality of this thesis is qualified for the award of the Bachelor Degree of Electrical Engineering (Electronics)”

Signature :  _____

Name : **NOR MANIHA BINTI ABDUL GHANI**

Date : **22 OCTOBER 2010**

“All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author.”

Signature

:  _____

Author

: LEE KONG HAUR

Date

: 22 OCTOBER 2010

Dedicate in thankful appreciation for support, encouragement and understandings to my beloved mother, father, lecturers and brothers.

ACKNOWLEDGEMENT

First of all, I would like to express my deepest gratitude to my supervisor, Pn. Nor Maniha Binti Abdul Ghani, for accept and giving me this wonderful and yet interesting final year project. Her supervision both aided me to channel and specify the discussed ideas and at the same time provided much appreciated freedom and support to explore new concepts. However, her endless drives for new and better results is highly appreciated, instead of willing to share her own experience and spend her free time to supervise me throughout the implementation of this project.

My appreciation also goes to my parents who bore immense difficulties to educate their children at the highest level. Their affection and love has brought happiness and joy to my life and therefore made this task becomes much easier to be implemented.

Nevertheless, my great appreciation dedicated to my friends, classmates and those whom involve directly or indirectly with this project. Finally, I hope that all the knowledge and experience gained through this project can be shared and bring benefit to all. Thank you.

ABSTRACT

In this age of fast changing technology, robots become part and parcel of our lives and yet it undergoes research and development in order to perform various of tasks. However, two-wheeled balancing robot nowadays could be considered as a challenging task for the robot hobbyists. For this project, Brain Board and Balance Board are the two most important elements for the two-wheeled balancing mobile robot system in order to make it has the capable of balancing upright of two wheels, as well as to interface or communicate effectively with the wireless PS2 controller via the aid of PSC28A I/O converter. Hence, the software of BrainBoard Code Editor provided can be used to develop our own navigation system which will then be loaded into the ATMEGA32 microcontroller on the Brain Board. Finally, the two-wheeled balancing robot can be navigated wirelessly by using the remote control of wireless PS2 controller.

ABSTRAK

Dalam era teknologi yang berubah dengan pantas, robot menjadi sebahagian yang tidak dapat dipisahkan daripada kehidupan kita dan malah masih di bawah kajian dan pembangunan untuk melakukan pelbagai tugas. Namun demikian, robot yang beroda dua pada masa kini boleh dianggap sebagai tugas yang mencabar bagi peminat robot. Dalam projek ini, Brain Board dan Balance Board merupakan dua elemen yang paling penting bagi robot beroda dua ini untuk menjadikannya memiliki kemampuan untuk mengimbangi tegak dua roda, serta dapat berkomunikasi secara berkesan dengan wayarles kawalan PS2 melalui bantuan PSC28A I/O penukar. Oleh itu, perisian BrainBoard Code Editor yang disediakan boleh digunakan untuk membina system navigasi kita sendiri yang kemudian akan dimuatkan ke dalam mikrokontroler ATMEGA32 di Brain Board. Akhirnya, robot yang beroda dua ini boleh dinavigasi tanpa kabel dengan menggunakan pengawal jauh kawalan PS2 wayarles.

TABLE OF CONTENTS

TITLE	PAGE
TITLE PAGE	i
DECLARATION	ii
DEDICATION	iv
ACKNOWLEDGEMENT	v
ABSTACT	vi
ABSTRAK	vii
TABLE OF CONTENTS	viii
LIST OF FIGURES	xi
LIST OF TABLES	xiii
LIST OF ABBREVIATIONS	xiv
LIST OF APPENDIXES	xv
CHAPTER 1	INTRODUCTION
1.1	Background of Project
1.2	Problem Statement
1.3	Objective of Project
1.3	Scope of Project
1.3	Project Summary
1.3	Flow of Project
CHAPTER 2	LITERATURE REVIEW
2.1	Introduction
2.2	Balancing Robot

CHAPTER 3	METHODOLOGY	19
3.1	Introduction	19
3.2	Balancing Mode	19
3.2.1	PID Controller	20
3.2.2	Analog GP2D12 IR Distance Sensor	21
3.3	Wireless Communication	23
3.3.1	Wireless PS2 Controller	23
3.3.2	PSC28A PS2 I/O Converter	26
3.4	Navigation System	29
CHAPTER 4	HARWARE SPECIFICATION	33
4.1	Introduction	33
4.2	Software Implementation	34
4.2.1	Software Installation	34
4.3	Hardware Implementation	36
4.3.1	Balance Board	39
4.3.1.1	Modes of Operation	40
4.3.2	Brain Board	41
4.3.3	Full-featured programmer	42
4.3.4	RS232 Serial Cable	43
4.3.5	Loading compiled program into the Brain Board	43
CHAPTER 5	RESULT AND DISCUSSION	47
5.1	Introduction	47
5.2	Hardware Testing	47
5.2.1	Testing on buttons	48
5.2.2	Testing on analog joystick	49
5.3	Balancing Mode	50
5.4	Navigation Mode	51
CHAPTER 6	CONCLUSION AND RECOMMENDATION	52
		52

6.1	Conclusion	
6.2	Recommendations for Future Work	54
6.3	Costing and Commercialization	55
REFERENCES		56
APPENDICES		58

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
1.1	Block Diagram for Mobile Robot's Navigation	5
1.2	Progress Flow Chart	7
2.1	JOE	11
2.2	nBot	12
2.3	Legway	12
2.4	Segway PT	13
2.5	Advanced Step in Innovative Mobility (ASIMO)	16
2.6	Two-wheeled autonomous balancing robot	16
2.7	Block diagram of PID control	17
2.8	Balancing a two-wheeled autonomous robot	17
2.9	PID control scheme for differential drive	18
3.1	Block diagram for balancing mode	20
3.2	Block diagram of PID controller	21
3.3	Sharp GP2D120 Infrared Sensor	22
3.4	Proper alignment surface being measured	22
3.5	Proper alignment to moving surfaces	22
3.6	Wireless PS2 Controller	24
3.7	Buttons of PS2 controller	25
3.8	Joystick direction output	25
3.9	Joystick PWM output	25
3.10	PSC28A PS2 I/O Converter	27
3.11	Flowchart for navigation system	30
3.12	Connection for the navigation system	31
3.13	Programming for navigation system	32
4.1	Block diagram of control system for two-wheeled	

	balancing robot	33
4.2	Icon for BalBot software	35
4.3	Steps for write and compile C code in BrainBoard Code Editor	36
4.4	ISP programmer	42
4.5	Serial Cable	43
4.6	Loading compiled program into the Brain Board	44
4.7	Steps for loading a compiled program	46
5.1	Connection for Up, Down, Left and Right buttons testing	48
5.2	Connection of RC Motors and PSC28A	49
5.3	Testing on analog joysticks	50

LIST OF TABLES

TABLE NO.	TITLE	PAGE
1.1	Final Year Project I Gantt Chart	8
1.2	Final Year Project II Gantt Chart	9
3.1	Absolute Maximum Rating for PSC28A	26
3.2	Parts Function Description	28
3.3	Navigation System	29
4.1	Specifications of two-wheeled balancing robot	37
4.2	Modes of Operation in Balance Board	40
6.1	Project Costing	55

LIST OF ABBREVIATIONS

TWIP	-	Two Wheeled Inverted Pendulum
BalBot	-	Balancing Robot
IR	-	Infrared
I2R	-	Inter Integrated IC
BPC	-	Balance Processor Chip
LQR	-	Linear Quadratic Regulator
GND	-	Ground
PID	-	Proportional-Integral-Derivative
CV	-	Controlled variable
SP	-	Set Point
PV	-	Processed variable
LED	-	Light Emitting Diode
MIPS	-	Microprocessor without Interlocked Pipeline Stages
DAC	-	Digital Analog Converter
PWM	-	Pulse Width Modulation

LIST OF APPENDICES

APPENDIX NO.	TITLE	PAGE
A	Programming in two-wheeled balancing robot	58
B	Sharp GP2D120 specifications	67
C1	Anatomy of Balbot	70
C2	Balance Board Overview	71
C3	Brain Board Overview	72
D	PSC28A I/O Converter and wireless PS2 receiver on Balbot	73

CHAPTER 1

INTRODUCTION

1.1 Background of Project

The research and development on two-wheeled balancing robot or commonly known as balancing robot or balbot has gained momentum over the last decade in a number of robotics laboratories around the world due to its inherent unstable dynamics of the system since it was built based on the inverted pendulum model. In brief, an inverted pendulum model is a pendulum which has its mass above its pivot point. Therefore, with the uniqueness of this model has drawn interest from many researchers as well as robot hobbyists around the world because of the unstable nature of the system. By then, it requires good control systems which capable to control itself in upright position dynamically balanced. Indeed, various type of controllers have been greatly implemented by the robotics researchers around the world currently such as Pole-placement controller, Linear Quadratic Regulator (LQR), Proportional-Integral-Derivative controller (PID) as well as Fuzzy Logic controller.

However, the main purpose of this project is to design and develop a two-wheeled balancing robot which has the capabilities to balance itself on flat terrain and hence navigated by the PS2 controller application. Briefly to say that this project would emphasis on the wireless communication between the two-wheeled balancing robot and the PS2 controller with the aid of PS2 I/O converter in order to interface effectively with the ATMEGA32 microcontroller of the Brain Board. However, BalBot Advanced is chosen as the model or platform to carry out this project as this

robot does not only solve the balancing problem but the robot could autonomously move around. In addition, the PID controller provided can be easily retuned with just modify the program in the balancing board, as such analysis can be done on it as well using MATLAB simulation.

However, recently many robot hobbyists would implement PS2 controller to control the navigation of the mobile robot built. Compared to other wireless communication application such as radio frequency (RF) and XBee, wireless PS2 controller would be a better choice as it could be navigated by the available joystick application. More importantly, vibrator motor available on the wireless PS2 controller would be an advantage for this project. However, the infrared sensors equipped in the mobile robot could able to detect the obstacles as getting close to them. Hence, a signal will be sent to the PS2 controller from the mobile robot to alarm the user not to move closer to that particular obstacle in order to protect the mobile robot. By then, the vibrate mode of the PS2 controller will be activated and the PS2 controller will be vibrated. It is cleared that this could be considered an extra function added compared to the previous two-wheeled balancing robot with remote control project which had been implemented.

In addition to that, the wireless PS2 controller does provide more buttons in order to expand more functions to be implemented on the two-wheeled balancing robot. However, the PS2 I/O converter of PSC28A in this project helps to convert the button and joystick information on PS2 controller into open collector output. As such, interfacing of the two-wheeled balancing robot with the wireless PS2 will be much more easy and faster. Apart from that, 40 pin PIC16F877A is provided for on-board programming purpose. Besides, each button pressed can be monitored via the LED indication on the PSC28A circuit board.

1.2 Problem Statement

The problem statement for this project is stated as below:

- i. To implement wireless hardware interfacing for the purpose of positioning control or navigation of Two Wheeled Balancing Robot.
- ii. To navigate the balbot wirelessly with retaining its balancing and stability

1.3 Objective of Project

The objectives of this project being carried out are stated as below:

- i. To interface the two-wheeled balancing robot with the remote control.
- ii. To control the navigation of the two-wheeled balancing robot in stable condition.

1.4 Scope of Project

The control system for the two-wheeled balancing robot emphasis on sensory applications and servo motor control function via the controller and navigated by the remote control. Briefly to say that this project covers the areas of research as below:

- i. Develop a stable self balancing two-wheeled balancing robot navigated by remote control.
- ii. To navigate wirelessly of the two-wheeled balancing robot with PS2 controller.

1.5 Project Summary

In brief, the project methodology can be divided into two parts which are hardware and software part. In this case, the hardware part refers to the Balance Board and Brain Board interfaced by I2C communication. On the other hand, the programming part written in C code would be the software part of this project which crucially in determining each button function as well as the navigation of the two-wheeled balancing robot. However, the software part for this project can be accomplished via the software “BrainBoard Code Editor” whereby the programme written in text form will then converted into hex file and finally burnt into the microcontroller using MEGA32ISP software.

The I2C communication can be defined as the Inter Integrated IC which commonly used to connect the microcontroller into the system. However, it is designed to be low cost and yet easy to implement with moderate speed up to 100Kb/s for standard bus and up to 400Kb/s for extended bus. Due to its good support for communication provided with various slow, on board peripherals devices that are accessed intermittently, therefore it become extremely modest in its hardware resource needs. Briefly to say that it is a multi master serial single ended computer bus that widely used in attaching low speed peripherals to a motherboard, embedded system or cell phone. Figure 1.1 illustrates the block diagram for wireless navigation in two-wheeled balancing robot.

An external system such as a microcontroller in Brain Board will be connected to the Balance Board expansion connector in order to control the Balance Board and to receive sensor information from the Balance Board. However, the data is exchange over the I2C bus. The balance processor chip (BCP) equipped in the Balance Board acts as an I2C slave. For communicating with it, a separate microcontroller would act as an I2C master. The microcontroller will initiate I2C read commands to receive data, and I2C write commands to transmit data to the BPC. While writing data, the master should always write 10 bytes. Meanwhile when reading data, it should always read 18 bytes [1].

The navigation of the two-wheeled robot is controlled by the wireless PS2 controller. However, as the buttons of the PS2 controller is pressed, hence the PSC28A will receive the signal wirelessly and finally received by the microcontroller of the Brain Board. By then, the microcontroller would process the data received before sending appropriate signal to the Balance Board whereby the BPC would use this data to drive the two DC motors so that it would keep the mobile robot in the center of gravity above the wheels all the time or, in the other word keep it balances instead of navigating according to the signal received from the wireless PS2 controller.

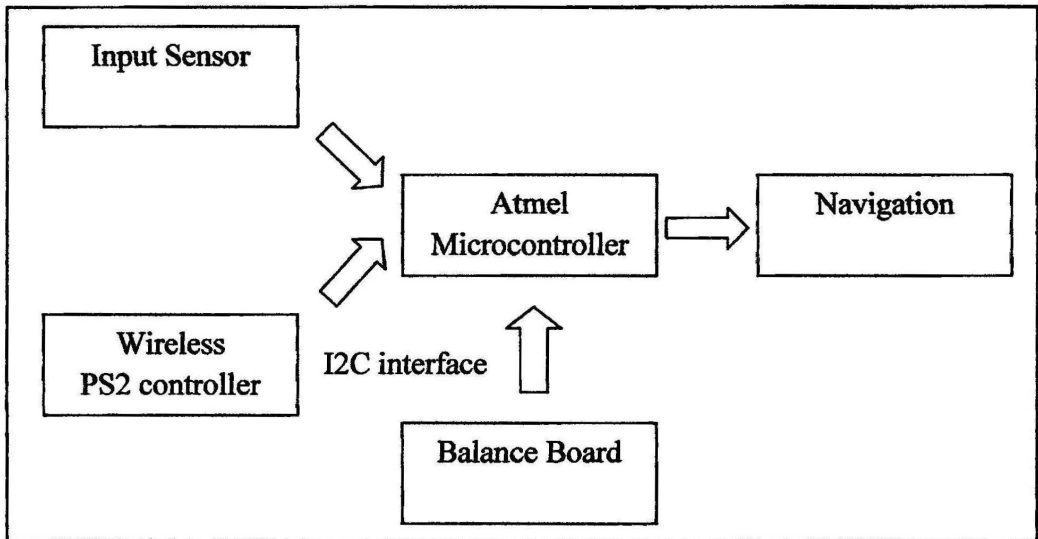


Figure 1.1: Block Diagram for Mobile Robot's Navigation

1.6 Flow of Project

The progress flow on the implementation of the project can be shown in the flow chart as illustrated in figure 1.2. It summarized all the work had been done throughout the implementation of the project on two-wheeled balancing robot with remote control from the project planning till the hardware interfacing. Furthermore, the Gantt charts in table 1.1 and table 1.2 show that the detailed of progress flow.

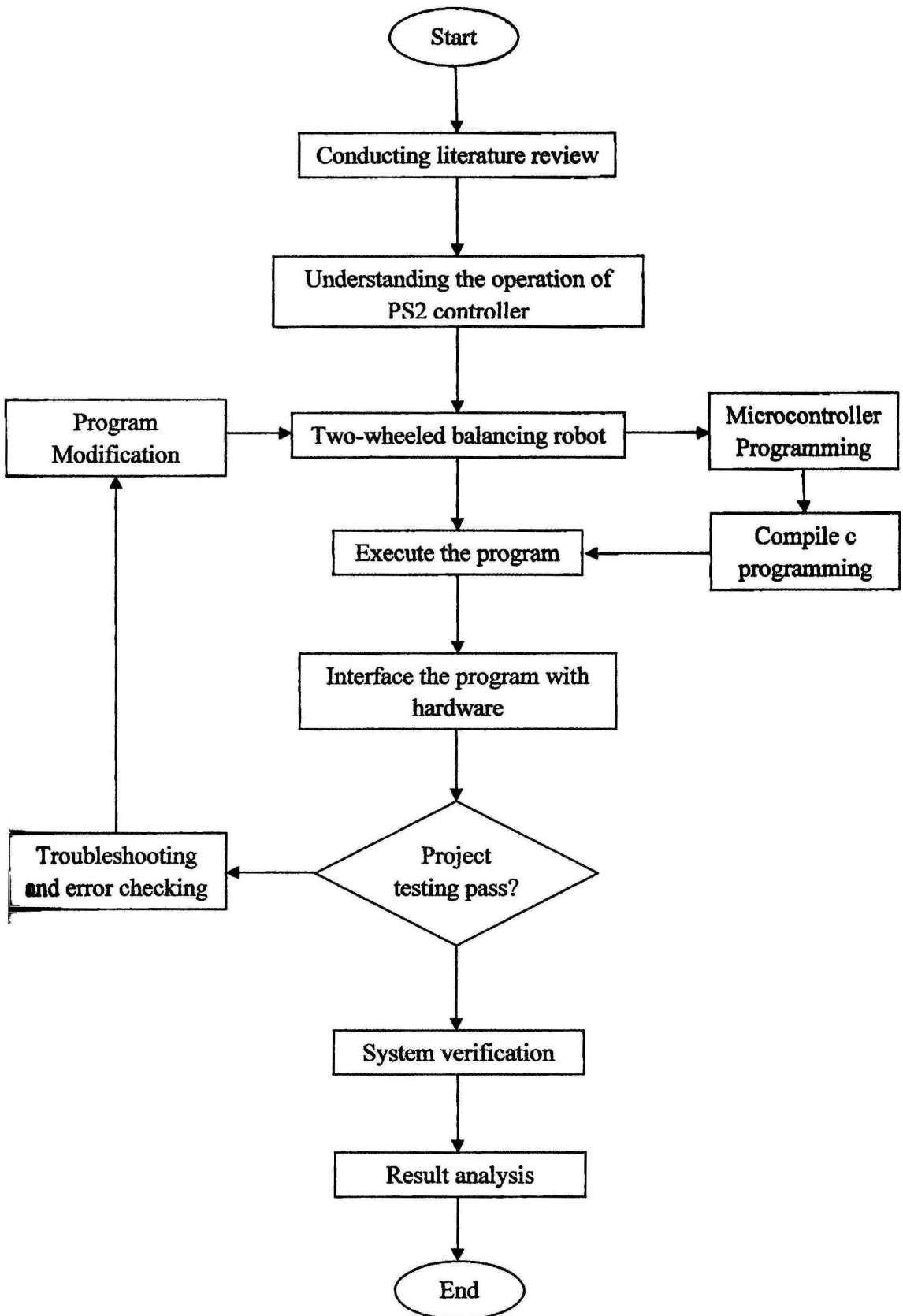


Figure 1.2: Progress Flow Chart

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Conducting literature review is considered crucially important especially for those who undertake research projects which will provide them with relevant information or detail regarding to technology available and methodologies that had been implemented or used by other researchers on related topics. However, this part would provide condensed summary of literature reviews on key topics which related on how to balance a two-wheeled mobile robot as well as application of wireless control with wireless PS2 controller are discussed.

2.2 Balancing Robots

The inverted pendulum is a pendulum which has its mass above its pivot point. However this problem is not uncommon in the field of control engineering nowadays. Meanwhile, with the uniqueness and wide application of technology derived from this unstable system has drawn interest from many researches and robot hobbyists around the world. Currently, the idea of a mobile inverted pendulum model is greatly applied by the researchers to various problems like designing balancing scooter, robotic wheelchairs as well as personal transport system.

The researchers at the Industrial Electronics Laboratory at the Swiss Federal Institute of Technology, Lausanne, Switzerland had successfully implemented a

scaled down prototype of a Digital Signal Processor (DSP) controlled two-wheeled vehicle based on the inverted pendulum model with weights attached to the system to simulate a human driver, namely JOE (figure 2.1). In brief, the control system used to achieve the stability of the system is based on two state-space controllers, utilizing sensory information from a gyroscope and two incremental encoders mounted on each dc motor [2].

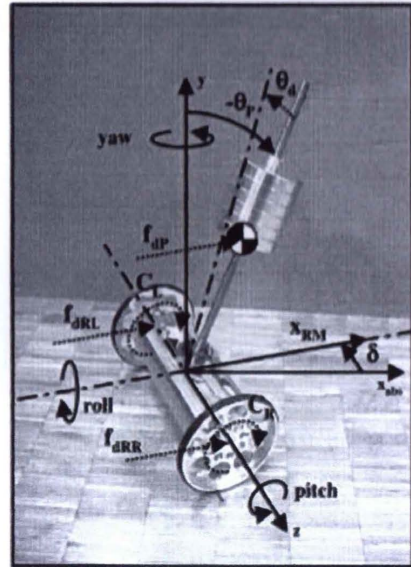


Figure 2.1: JOE
(<http://leiwww.efpl.ch/joe>)

And yet another two-wheeled balancing robot similar to JOE developed by David P. Anderson which makes use of commercially available inertial sensor and the position information from the motor encoder to balance the system, called nBot (figure 2). The robot would remain balanced as the system could able to drive the wheels in the direction that the upper part of the robot is falling. Four terms used to define the motion and position of the system which then summed and fed back to the platform as a motor voltage that is proportional to torque in order to balance and drive the robot [3]:

- i. The tilt angle
- ii. Its first derivative, the angle velocity
- iii. The platform position

iv. Its first derivative, the platform velocity

Apart from that, Steven Hassenplug who had successfully built a two-wheeled balancing robot called Legway (figure 3) using the LEGO Mindstorms robotics kit. However, two Electro-optical Proximity Detector (EOPD) sensors which based on the Infrared Proximity Detector (IRPD) circuit were used in order to provide the tilt angle of the robot to the controller and maintain a constant distance to the ground which programmed in BrickOS, a C/C++ like programming language specifically for LEGO Mindstorms [4].

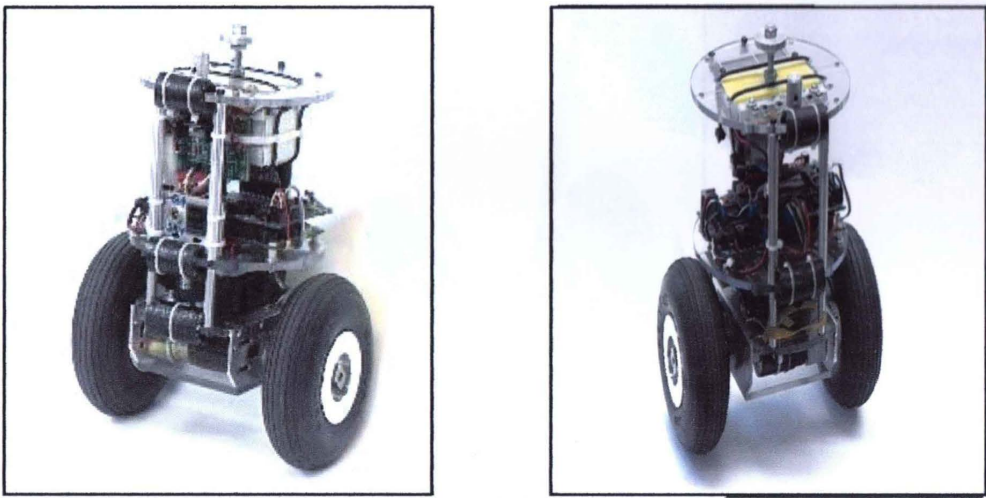


Figure 2.2: nBot



Figure 2.3: Legway

However, Segway PT (figure 4) is one of the examples of commercially available system which has been invented by Dean Kamen. This personal

transportation is able to balance a human standing on its platform while the user traverses the terrain with it. More importantly, the computers and motors in the base of the device keep it upright when powered on with balancing enabled. By then, the users lean forward to move forward and lean back to move backward, finally able to turn it by using a “Lean Steer” handlebar for leaning it left or right. Specifically, it balances with the help of dual computers running proprietary software, tilt sensors and five gyroscopes.



Figure 2.4: Segway PT

In the paper of Peter Miller on building a two wheeled balancing robot stated that some important elements in designing a stabilize two-wheeled robot system such as inertial sensors, shaft encoders, control algorithm and microcontroller as the brain of the robot to complete computations and process data by executing instruction or programming[5]. However, fuzzy logic controller is implemented to ensure the stability of the system. The Matlab is chosen as the simulation platform to work for stability and locomotion analysis.

Similarly, Zatil Hanan Binti Ahmad Lutpi from Universti Teknologi Malaysia who taken the research on position control of two wheeled inverted pendulum mobile robot. However, the paper described that the mobile robot would

balance on its own body and always place on center of gravity as both the values of analog reading GP2D120 IR distance sensor produce same value. Instead, the GP2D12 is chosen to work as the input for the path control algorithm[6].

Moreover, under the research on [7] of modelling and control of a balancing robot using digital state space approach defined that modelling is the process of identifying the principal physical dynamic effects to be considered in analyzing a system, writing the differential and algebraic equations from the conservation laws and property laws of the relevant discipline, and reducing the equations to a convenient differential equation model. In brief, the dynamic performance of a balancing robot depends on the efficiency of the control algorithms and the dynamic model of the system. Some assumptions and limitations have been mentioned in order to make the modelling of the balancing robot a reality which stated as below[7]:

- i. Motor inductance is neglected and the current through the winding is not considered in the equation of motion of the motor
- ii. The wheels of the robot will always stay in contact with the ground
- iii. There is no slip at the wheels
- iv. Cornering forces are also negligible

Shiroma et al. (1996) presented the ‘Cooperative Behaviour of a Wheeled Inverted Pendulum for Object Transportation’ by showing the interaction of forces between objects and the robot by taking into account the stability effects due to these forces[8]. This research highlights the possibility of cooperative transportation between two similar robots and between a robot and a human.

Chinichian (1990) design and analyze a controller for balancing one pendulum with two degrees of freedom, “spatial inverted pendulum”. The pendulum, with two degrees of freedom, has a three dimensional motion, and it will be more analogous to the design of a controller for attitude control during launching a rocket. A full state-variable feedback controller design for a state-space linear model of a three dimensional inverted cart/pendulum system is presented. This design was based on pole-placement technique. Alternative solutions to the simple pole-

placement technique were also proposed to exploit non-uniqueness of the feed-back gains for a certain closed-loop pole locations and the closed-loop system response was simulated on a digital computer.

2.3 Control System

The two-wheeled balancing robot would require a control algorithm in order to achieve the stability of dynamics of the system. Basically, it can be divided into two types of control systems, namely linear and non linear control system which widely implemented on inverted pendulum system such as proportional-integral-derivative (PID), linear quadratic regulator (LQR), pole-placement and fuzzy logic controller.

However, with the rapid increasing of the aged population in countries like Japan has prompted researchers to develop robotic wheelchairs to assist the infirm to move around (Takahashi et al. 2000). The control system for an inverted pendulum is applied when the wheelchair manoeuvres a small step or road curbs. Meanwhile on a higher level, Sugihara et al. (2002) modelled the walking motion of a human as an inverted pendulum in designing a real time motion generation method of a humanoid robot that able to control the centre of gravity by indirect manipulation of the Zero Moment Point (ZMP)[10]. The real time response of the method provides humanoid robots with high mobility. An example of humanoid robot is ASIMO as shown in figure 2.5.

Instead, Ong Yin Chee and Mohamad Shukri b. Zainal Abidin from Universiti Teknologi Malaysia who had undertake the research on design and development of two wheeled autonomous balancing robot (figure 2.5). The information on current position and tilt angle of the robot are acquired from the distance measuring sensors (Sharp GP2D12) mounted on an aluminum strip placed at the front and back of the robot. Hence, the analog voltage output from the sensor will converted into digital value via A/D converter of microcontroller then inputs it

into PID control algorithm as shown in the figure 2.7 to determine the speed and direction of the motors. Finally, the robot could stand it upright in stable condition and able to be navigated by RC remote control. In brief, these sensors are used to detect the current position and tilting angle of the mobile robot [11].

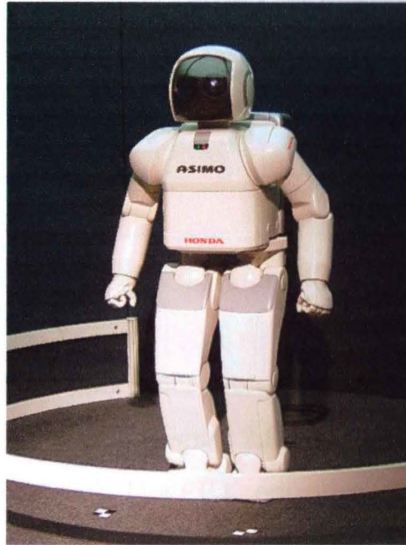


Figure 2.5: Advanced Step in Innovative Mobility (ASIMO)



Figure 2.6: Two-wheeled autonomous balancing robot

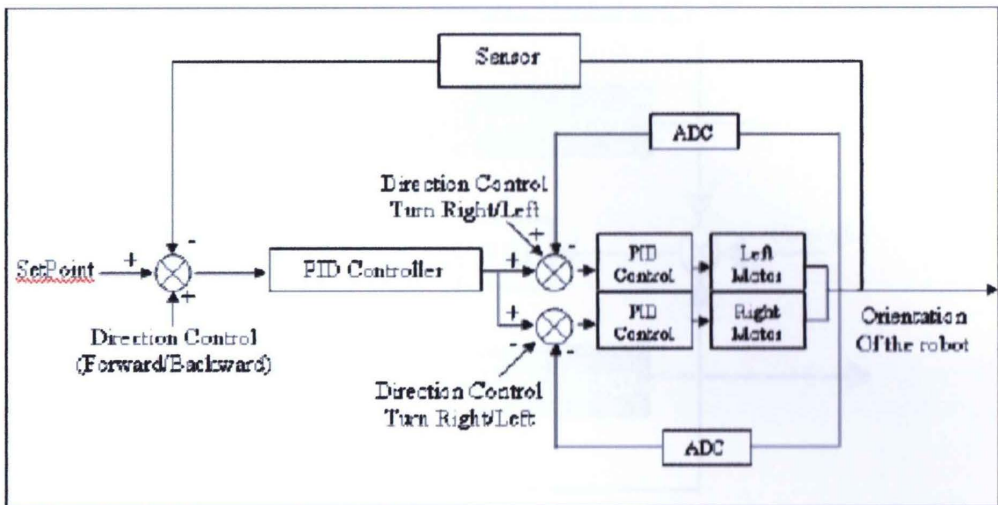


Figure 2.7: Block diagram of PID control

Rich Chi Ooi who undergoes research on balancing a two-wheeled autonomous robot as illustrated in the figure 2.8 described on how the PID controller working for the trajectory control. The PID controller which displayed in figure 2.9 is useful as it will ensure that the position errors between the wheels are minimized, so that the robot will move in a straight path. However, the PID controller used a set of tuning rules which is based on the Ziegler-Nichols method. The encoder reading difference is passed through the PID loop to obtain the required feedback to be added to the motor controller [12] which can be illustrated in the block diagram below.

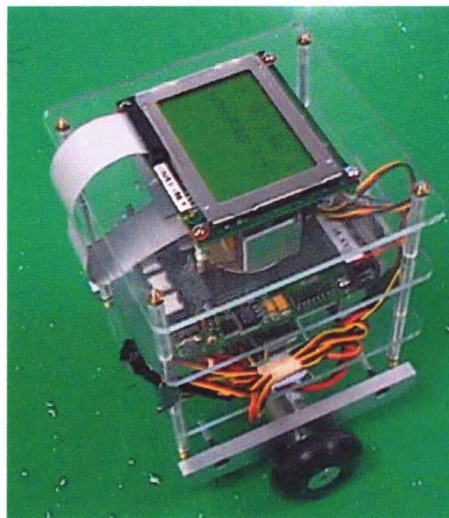


Figure 2.8: Balancing a two-wheeled autonomous robot

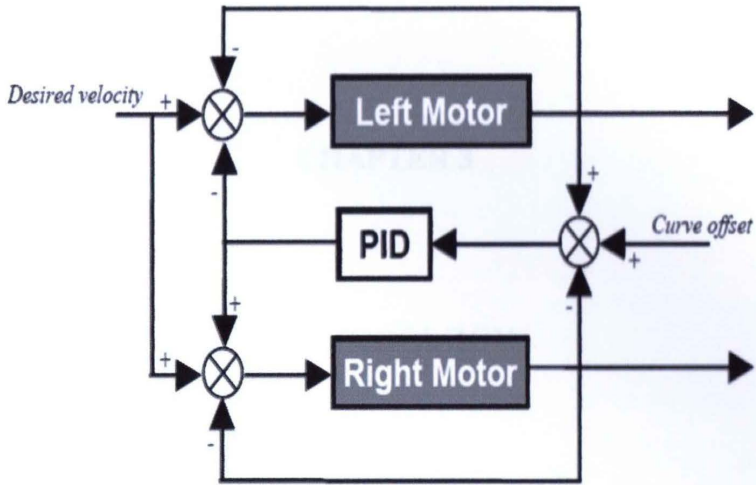


Figure 2.9: PID control scheme for differential drive

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter would emphasis on the balancing of the two-wheeled inverted pendulum mobile robot and the application of wireless remote control function on the mobile robot. Besides, interfacing between the wireless PS2 controller with the two-wheeled balancing robot will be greatly discussed, instead of explain testing procedures being carried out on the PSC28A circuit board which will then be applied on the mobile robot for wireless communication purposes.

3.2 Balancing Mode

The balancing mode for the two-wheeled balancing robot refers to the ability of the mobile robot balances on its body, or in the other words mean that the mobile robot could stand it upright in dynamically stable condition and hence navigated by wireless PS2 controller. However, the PID controller as well as the infrared distance sensor of GP2D120 plays the crucial role in achieving the stability of the two-wheeled inverted pendulum mobile robot in the manner that to ensure it stands upright on flat terrain while navigated, or keep the center of gravity of the mobile robot above the wheels all the times. Thus, two analog infrared distance sensor are used and located in front side and back side between the two DC motors on the two-wheeled mobile robot. Therefore, data obtained in analog form from the sensors will be read by the microcontroller in the Brain Board and then appropriate command

will be sent to Balance Board to drive the two DC motors that will ensure the mobile robot balance all the time. Figure 3.1 illustrates on how the balancing of the two-wheeled mobile robot is achieved or worked with PID controller and analog infrared sensor GP2D120.

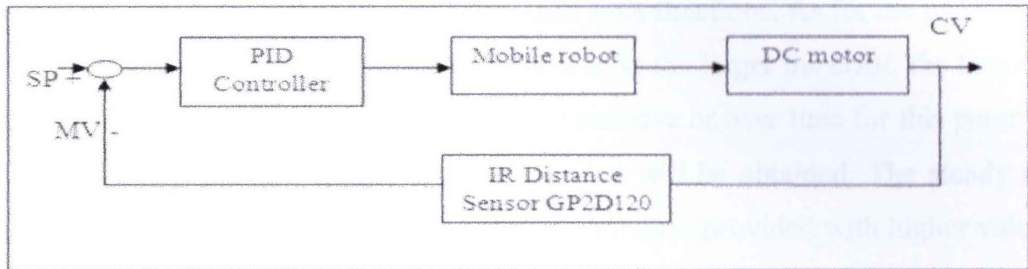


Figure 3.1: Block diagram for balancing mode

3.2.1 PID Controller

For our information, PID controller shown in figure 3.2 stands for Proportional-Integral-Derivative controller which can be considered as a linear and feedback type of controller that widely used in the robotics technology recently especially dealing with unstable inverted pendulum application, it is because of its simple implementation provided with three adjustable parameters of proportional K_p , integral K_i and derivative K_d . Instead, another three important terms for PID controller that needed to be considered are set point (SP), controlled variable (CV) and manipulated variable (MV).

However, the set point refers to the desired value set in the system. On the other hands, the manipulated variable can be defined as the actions taken by the PID

known as process variable is defined to as the measurable output of the system which will be used to compare to the set point of the system.

K_p , K_i and K_d which have been mentioned above which are the important parameters for this PID system and have their own functions. As for the proportional gain, larger values will give faster response due to the larger the error, the larger the proportional term compensation. However, excessive or over tune for this parameter would lead to process instability and oscillation will be obtained. The steady state errors will be eliminated more quickly as the system is provided with higher value of integral gain, but it will lead to large overshoot if tuned improperly. On the other hand, the derivative gain plays an important role in decreasing the overshoot, but slow down transient response of the system.

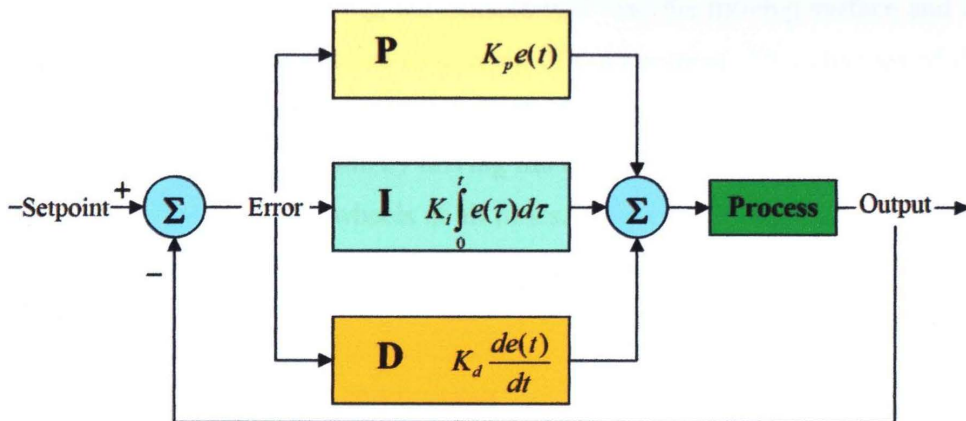


Figure 3.2: Block diagram of PID controller

3.2.2 Analog GP2D120 IR Distance Sensor

For the balancing system purposes, hence two analog IR distance sensor (Sharp GP2D120) is chosen to work as inputs to read the distance on floor surface and then it can balance afterward. This analog distance sensor could cover for the range of 4 to 30cm as shown in figure 3.3. However, by referring to appendix B is a datasheet of GP2D120 in terms of specifications which shows that this sensor

provides a non linear analog output voltage versus distance to reflective object. The object is detected by this sensor is a distance from the sensor to the flat floor surface.



Figure 3.3: Sharp GP2D120 Infrared Distance Sensor

Refer to figure 3.4 and 3.5 shows that a way of this IR sensor reads the flat floor surface as well as for moving surface. Each of them composed of an emitter and a receiver. The emitter will send the signal to the flat floor surface and the distance of surface is obtained and hence calculated that greatly depending on the reflected signal back to the receiver of the IR sensors. In the figure 3.5, it shows that a proper alignment when a mobile robot is moving, the sensors will read the moving surface and measures the different data of sensors due to altering of environment. This changes of data from the sensors will send to the microcontroller and gives a command to the Balance Board, so that adjustment will be done by driving the two motors in a manner that will keep the center of gravity above the wheels at all times. As such, the mobile robot could balance all the time especially during the navigation controlled by the wireless PS2 controller.

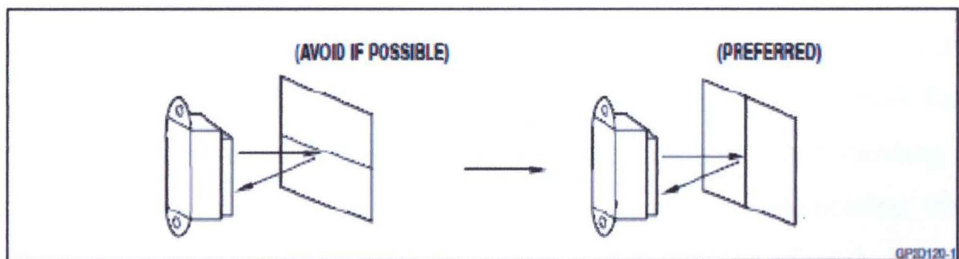


Figure 3.4: Proper alignment surface being measured

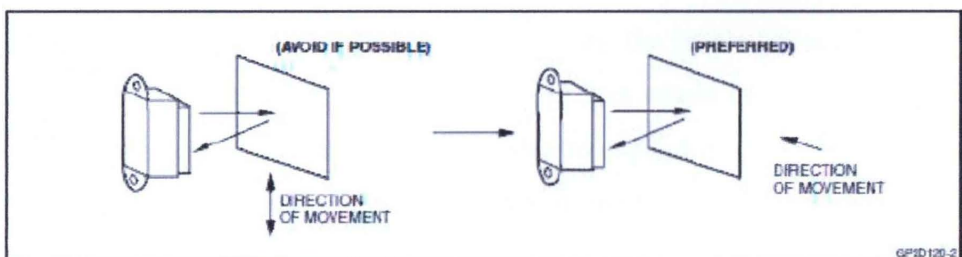


Figure 3.5: Proper alignment to moving surfaces

3.3 Wireless Communication

For the navigation of the two-wheeled mobile robot, wireless PS2 controller is chosen instead of using RC remote control and RF application due to its cost effective and easy to implement. Apart from that, it can be controlled in a wider range compared to the RF application. More importantly, with the two joysticks provided in the wireless PS2 controller and yet suitably used to navigate both of the DC motors respectively.

However, with the plenty of buttons provided in the wireless PS2 controller, able to expand the function or features of the two-wheeled balancing robot. In addition to that, a feature is added to this project that is fully utilized the function of the vibration found in the wireless PS2 controller.

3.3.1 Wireless PS2 Controller

In order to provide navigation for the mobile robot built, however more and more developer and robotics hobbyist are looking into implement existing joystick such as Play Station 2 (PS2) controller in their system. The major problems dealing with this application such as difficulties to obtain the connector socket for PS2 which is unique and difficult to source as well as require great understanding in the protocol to communicate with it. Instead, the process of implementing the PS2 controller protocol to acquire the status (digital and analog) of each button and analog stick are troublesome and time consuming.

With the problems arisen above, therefore the implementation of wireless PS2 controller as shown in figure 3.6 into the project would be considered as a great challenge. However, this application could able to expand more features or functions of the two-wheeled mobile robot to make it multifunctional. Compared to the previous two-wheeled mobile robot that navigated by the RC remote control, the

wireless PS2 controller provides plenty buttons and two joysticks with extra internal feature of vibration mode. Due to its cost effective, thus makes it a better choice for the robot hobbyist to apply it on their mobile robot compared to ordinary RF application. More importantly, stable data transmission of the wireless communication and thus make it easily interfaces between the wireless PS2 remote control with the two-wheeled mobile robot.



Figure 3.6: Wireless PS2 Controller

Figure 3.7 illustrate the labeled 14 buttons found the wireless PS2 controller. However, each button will control an output at PSC28A. Instead, another interesting feature offered by this PS2 controller would be the two analog joysticks which labeled with direction as shown in figure 3.8. Besides, the vibration features offered by this PS2 controller can be activated via sending active high signal to the pin MTR1 and MTR2. However, ensure that if these pins are connected and controlled by the microcontroller, the port connected to these pins must be set as output so that appropriate signal could be sent to them to activate the vibration motor of the PS2 controller.

Each joystick controls two PWM output. As such when the left joystick is at its origin location, which is at the center, both of the PWM output (J LX and J LY) will be zero. While the left joystick is being moved up or down from center, the J LY PWM output will alter from 0% to 100%. The value of 100% indicates that the

position at top most and bottom most. Similarly, the J_{LX} PWM will change from 0% to 100% once the left joystick is being moved left or right from center. Figure 3.9 illustrates the joystick PWM output.



Figure 3.7: Buttons of PS2 controller



Figure 3.8: Joystick direction output

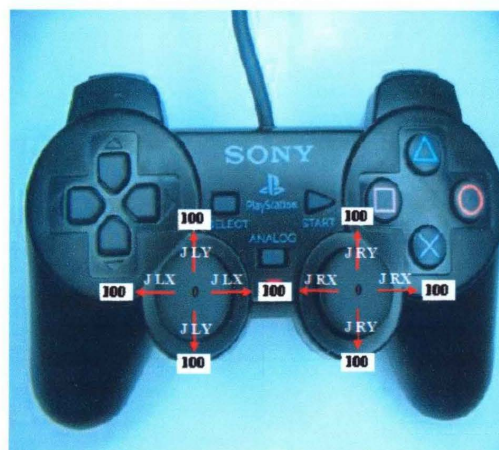


Figure 3.9: Joystick PWM output

3.3.2 PSC28A PS2 I/O Converter

PS2 I/O converter with model of PSC28A as shown in the figure 3.10, a product of Cytron Technologies is introduced to help to solve the problems mentioned above. More importantly, the PSC28A offers a standard connector socket for PS2 controller to plug-in and we can opt to interfacing with or without the microcontroller. Briefly to say that it provides fast and simple way to use the joystick as the input of each button would be converted directly into an output and yet its design is compact, reliable and low cost. The features for this PSC28A are concluded as below:

- i. 7-15V power input, current consumption less than 300mA at 12V
- ii. Using standard PS2 controller connector socket
- iii. Each button will be represented with an output pin
- iv. Easy to interface with different voltage system
- v. Vibrator motor is controllable
- vi. Fully compatible with wired or wireless PS2 controller
- vii. Joy-stick will only operate in analog mode
- viii. Indicator for each output
- ix. 4 PWM output to represent the analog

PSC28A does not only convert the buttons and joystick information on PS2 controller into open collector output with 14 open collector output for 14 buttons, four PWM output for four joystick axis and eight open collector output for 8 joystick directions. Instead, there are also two inputs to control the vibrator motor on the wireless PS2 controller. However, the product specifications and limitations for PSC28A are shown in Table 3.1 as well as part function description in Table 3.2.

Table 3.1: Absolute Maximum Rating for PSC28A

Parameter	Min	Typical	Max	Unit
Input voltage	7	12	15	V
PWR Out	-	Input voltage – 0.7	-	V

Current consumption	-	100	300	mA
---------------------	---	-----	-----	----

For our information, there are two methods of using PSC28A. The user can choose to interface the PSC28A with the microcontroller or directly connected into I/O device with ease of programming. As such if the user opts for use direct connection into I/O device, thus motor driver like L293D, L293B or L298 to connect with the PSC28A in order to control motor with controllable speed. Therefore, it helps to ease the programming parts that prone to error easily. Meanwhile, if the PSC28A is interfaced with the microcontroller, thus programming step is necessary as ports used must be defined as well as output port must be activated via program.

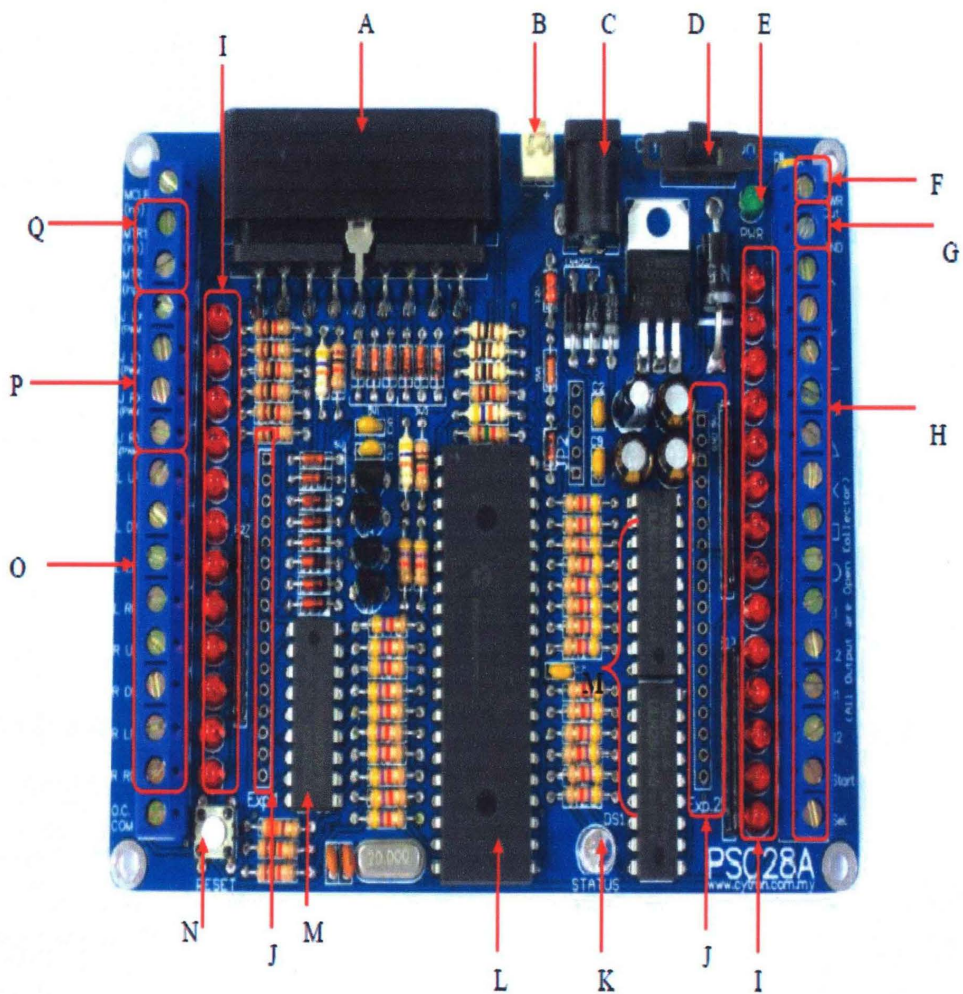


Figure 3.10: Board layout of PSC28A PS2 I/O Converter





Table 3.2: Parts Function Description

Label	Function
A	PS2 connector socket
B	Battery connector
C	DC power adaptor socket
D	Slide switch for main power supply
E	Power indicator LED
F	Power out terminal block
G	GND terminal block
H	PSC28A Output terminal (buttons)
I	Status indicator LED
J	Expansion 1 and Expansion 2
K	Status LED
L	PIC microcontroller
M	ULN2803AG
N	Reset button
O	PSC28A Output terminal (direction)
P	PSC28A PWM output terminal
Q	PSC28A Input terminal

3.4 Navigation System

Table 3.3 shows the designing of position control or navigation system for the two-wheeled balancing mobile robot using wireless PS2 controller. In brief, there are four states for the navigation system for the mobile robot which determine where its direction of movement either go forward, backward, left or right based on the button pressed on the wireless PS2 controller.

Table 3.3: Navigation System

Navigation Button	Output Desired Path
	Forward
	Backward
	Turn Right
	Turn Left

Once the navigation buttons on the wireless PS2 controller are pressed, the PSC28A will process the signal input obtained via the PS2 receiver plugged into it, hence will then be received by the ATMEGA32 microcontroller in the Brain Board. By then, the microcontroller will process the signal input received and identified which button is being pressed, so that appropriate action in the form of signal will be sent to the DC motors for further direction navigation of the mobile robot. However, the navigation system for the mobile robot can be expressed in flowchart which shown in the figure 3.11.

In addition to that, additional feature adds on for this two-wheeled balancing mobile robot in order to notify the user from hitting the obstacles detected in front by activating the vibration motor found in the wireless PS2 controller. Therefore, the infrared distance sensors of GP2D120 play an important role for detecting the

obstacles in front. Hence, the two-wheeled balancing mobile robot is equipped with two infrared sensors with located in the left and right of it.

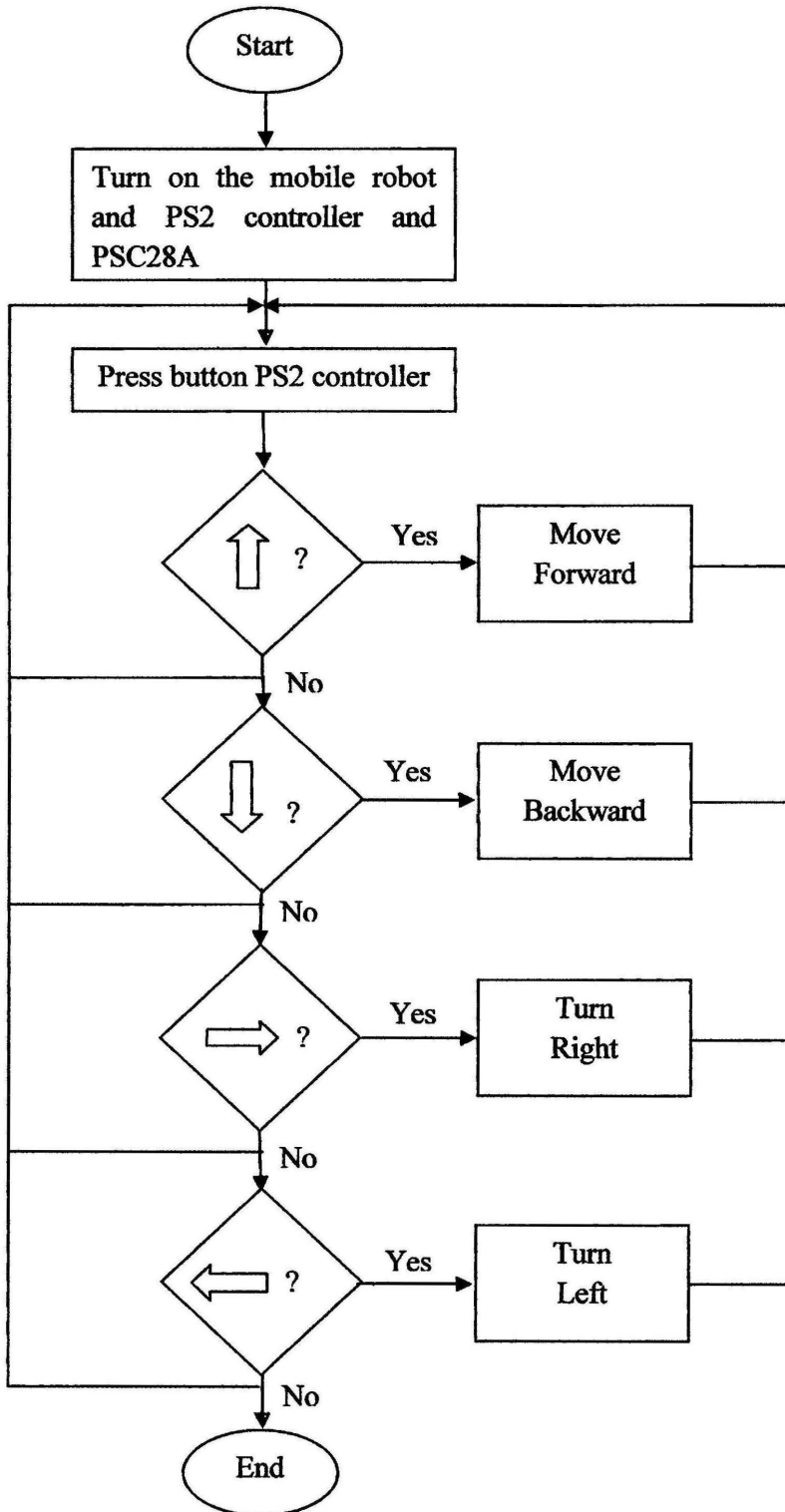


Figure 3.11: Flowchart for navigation system

Figure 3.12 shows the connection between the ATMEGA32 microcontroller in the Brain Board and PSC28A I/O converter in order to interface with the wireless PS2 controller. Port B is another available port which is further defined to use for navigation system. Therefore, pin 0 until pin 3 of the port B are available to use for this purpose. As such, modification must be done for the programming part of the microcontroller in order to achieve this objective. The pin B0 is connected to Up button, pin B1 is connected to Down button; meanwhile the pin B2 is connected to Left button and pin B3 is connected to Right button.

Instead, there are two modes available in the two-wheeled balancing mobile robot: balancing and explore modes. However, the programming is modified as well so that the mobile robot only will be navigated while in the explore mode. Moreover, two important variables that determine the navigation system, namely `bal_proc_wr.fwd_rev` and `bal_proc_wr.steer`. The positive value for `bal_proc_wr.fwd_rev` indicates it will move forward and move backward if this variable is set to negative value. On the other hand, the positive value for `bal_proc_wr.steer` indicates it will turn right; else it will turn left if the value is set to be negative. The programming for the navigation system is shown in the figure 3.13.

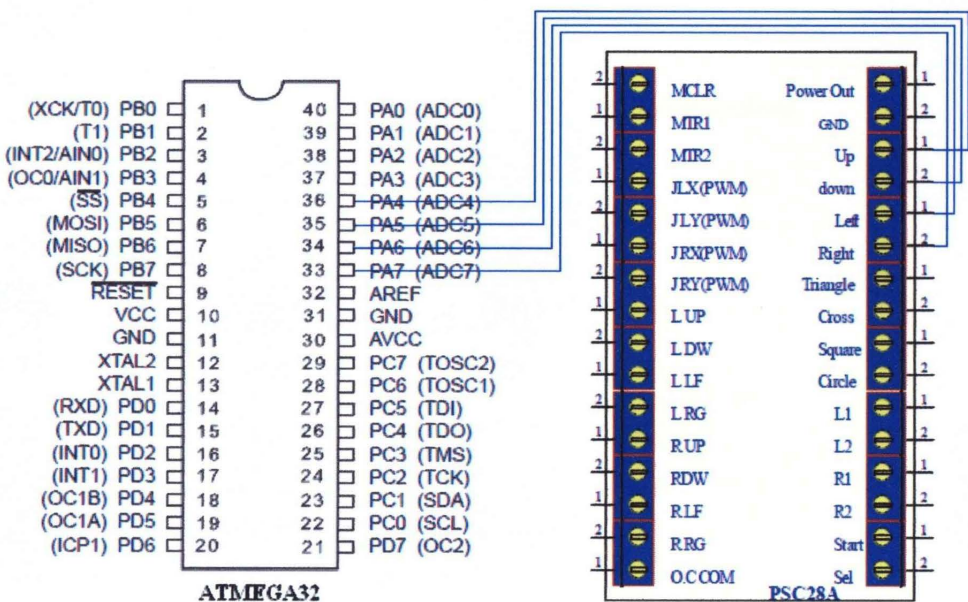


Figure 3.12: Connection for the navigation system

```
if(explore)                // If we're told to explore
{
  DDRA  &=0x00;
  PORTA |=0xFF;
  if((PINA&0x10)==0)
  {
    bal_proc_wr.fwd_rev=50;
    bal_proc_wr.steer=0;
  }
  else if((PINA&0x20)==0)
  {
    bal_proc_wr.fwd_rev=-50;
    bal_proc_wr.steer=0;
  }
  else if((PINA&0x40)==0)
  {
    bal_proc_wr.fwd_rev=50;
    bal_proc_wr.steer=-60;
  }
  else if((PINA&0x08)==0)
  {
    bal_proc_wr.fwd_rev=50;
    bal_proc_wr.steer=60;
  }
}
else
{
  bal_proc_wr.fwd_rev = 0;
  bal_proc_wr.steer = 0;
}
```

Figure 3.13: Programming for navigation system

CHAPTER 4

HARDWARE SPECIFICATION

4.1 Introduction

In this project, it can be divided into two important parts, namely Brain Board and Balance Board. However, each of them has its own function or contribution towards this two-wheeled balancing robot system. The ATMEGA32 microcontroller located in the Brain Board will process the data or information obtained by the Balance Board in order to perform real-time balancing control. The figure 4.1 shows the block diagram of two-wheeled balancing robot control system whereby it is a closed-loop system with the controlled of the desired velocity and altering of tilt angle as the input of the system [1].

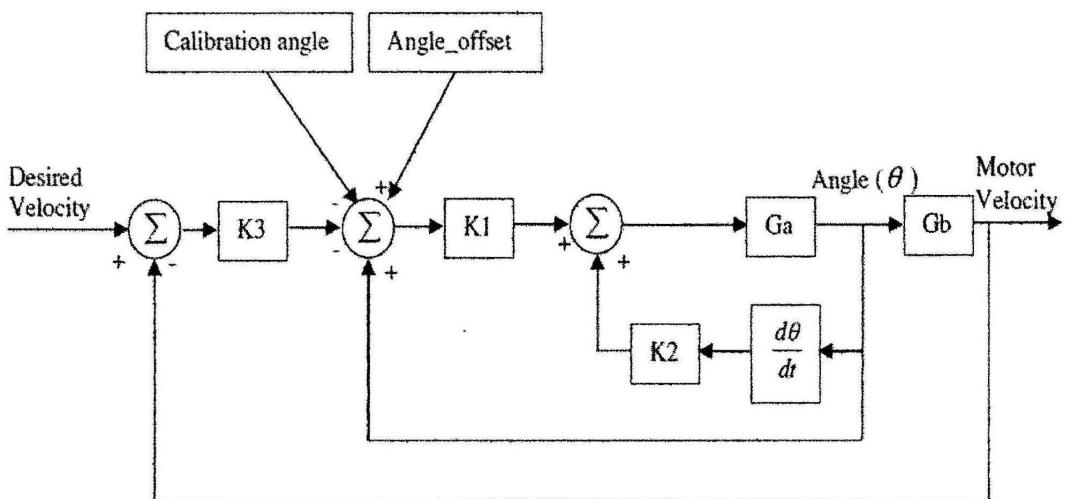


Figure 4.1: Block diagram of control system for two-wheeled balancing robot

While the mobile robot navigates according to the desired velocity, one thing that will take into consideration is the balancing problem of the mobile robot

especially dealing high speed of navigation. As such, from the diagram of control system above, the desired velocity is compensated by the K3 controller. Moreover, the calibration angle refers to the deviation or comparison between the correct angle and the measured angle. On the other hand, the angle offset can be defined as the compensated angle done to correct the measured angle close to the desired angle of the system which controlled by K1 and K2 controllers.

Appendix C1 displays the construction of the two-wheeled balancing robot with each part of the mobile robot is clearly stated and labeled. However, this chapter would explain in detail on the components used in the two-wheeled balancing robot as well as the software implementation.

4.2 Software Implementation

This section would explain in detail on the software used to program the microcontroller of the Brain Board. Instead, the software installation and ways to operate the software installed in order to deal with the work of compiling and burning the program written to our ATMEGA32 microcontroller located in the Brain Board.

4.2.1 Software Installation

Once the software is completely installed, a new shortcut group called “BalBot” will appear under the start menu which shown in the figure 4.2. However, the Programmers Notepad 2.0.5.32 which comprised of AVR-GCC 3.4.1, hence make it a user-friendly and easy to use with especially for the new user. Besides, its all-in-one features hence make it a convenient and useful tool to dealing with the programming of the microcontroller. It is because it does not only provide notepad

for the user to write the C code, but also provide compiling function that able to convert the C code into hex file to be burned in the microcontroller.

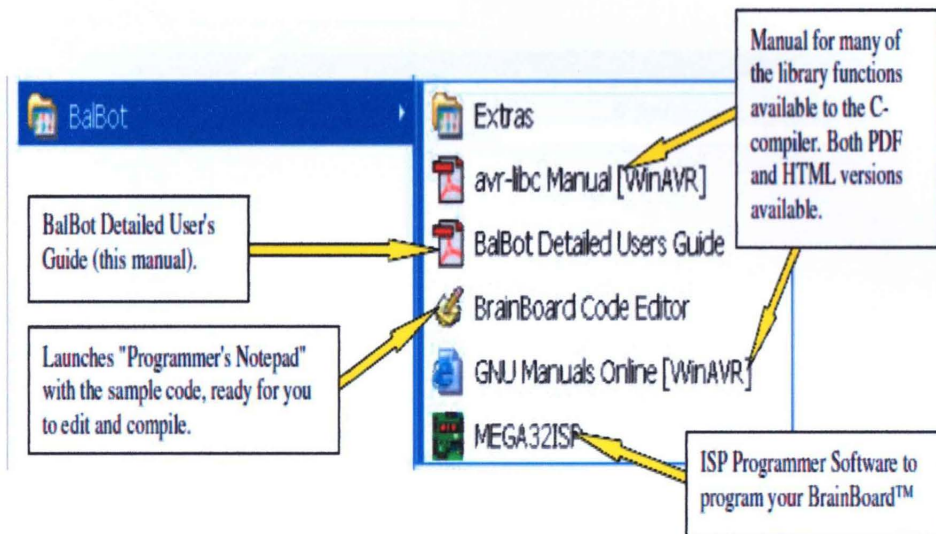


Figure 4.2: Icon for BalBot software

Below are the steps or procedures on how to write a C code and compile the C code written into hex file in order to be burned into the microcontroller using MEGA32ISP [1] which expressed clearly in the figure 4.3.

- i. The icon "BrainBoard Code Editor" is clicked.
- ii. Programmer's notepad should be opened up with the sample project automatically loaded. This sample project is consisted of a makefile, several *.c files and several *.h header files. However by default, the files of this software will be installed in the following directory:
C:\BalBot\BrainBoard_Code\.
- iii. The file balbot.c is ensured recently being displayed in the Programmer's Notepad main window.
- iv. In order to compile the C code written in the notepad, select "Tools", followed by "[WinAVR] Make All". However, in the window at the bottom of the screen, it will be observed that the C code will be compiled by the AVR-GCC compiler and displays "> **Process Exit Code: 0**".

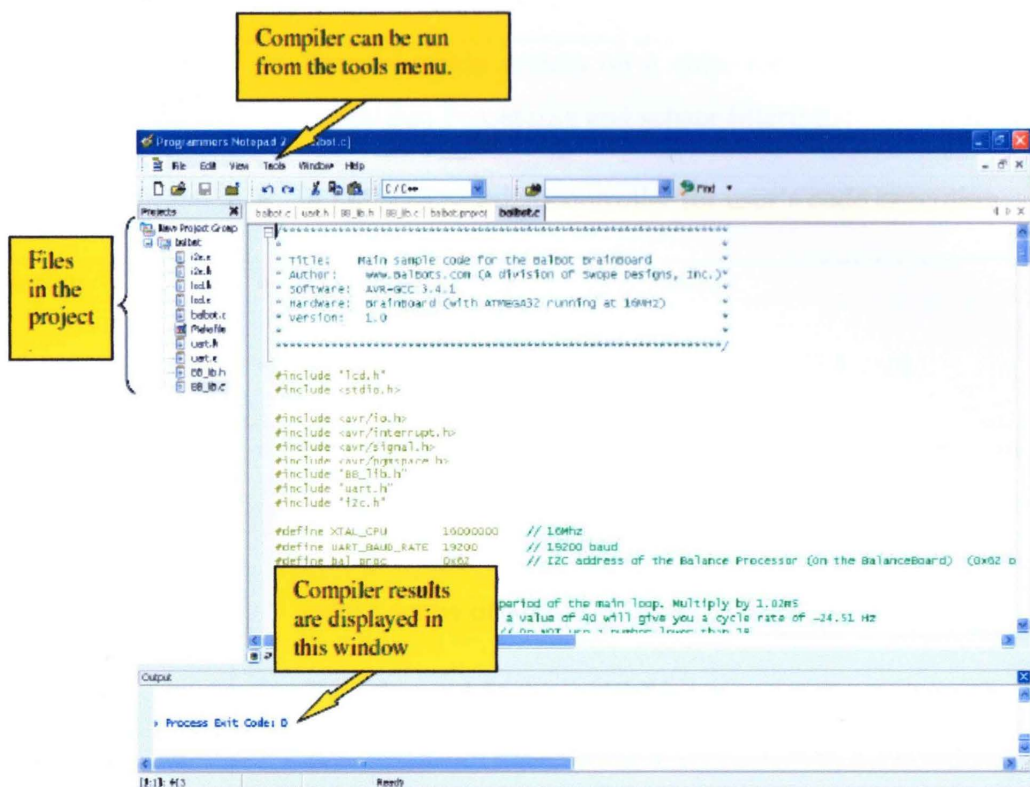


Figure 4.3: Steps for write and compile C code in BrainBoard Code Editor

4.3 Hardware Implementation

This subtopic would discuss about the elements or components part in the anatomy of two-wheeled balancing robot that have been implemented including Expandable Frame System, Brain Board, Balance Board, DC gear motors, dual ground sensors, full featured programmer as well as the serial cable for the communication with the personal computer. The specification for the two-wheeled balancing robot is described in detail in the Table 4.1.

Table 4.1: Specifications of two-wheeled balancing robot

Processor	<ul style="list-style-type: none"> • Programmable system on a chip, for real-time Balance Algorithm Processing and sensor filtering. • ATMEGA32 microcontroller for user's code execution.
Motion	<ul style="list-style-type: none"> • Powerful DC brushed gear motors. • Two current-limiting motor controllers (3A each).
Mechanics	<ul style="list-style-type: none"> • Expandable Frame System (EFS), made primarily of aircraft aluminum. Allows unlimited form factor changes, sensor additions, board additions, etc., with only a screw driver.
Sensors	<ul style="list-style-type: none"> • Two forward looking, 10-80 cm range distance-sensing infrared sensors are included. • Two infrared ground sensors, for tilt-angle sensing. • Back-EMF sensing on each motor, to detect wheel velocity.
Upgrade-ability	<ul style="list-style-type: none"> • Additional sensors can easily be added (up to 8 total analog sensors, 6 digital devices and multiple I2C devices). • "Plug and Play" which is no soldering required for additions devices • Sensors and circuit boards can be easily added with only a screw driver.
Balance Algorithms	<ul style="list-style-type: none"> • Digital Cascaded Control Loop is standard (built into Balance Processor Chip)

	<ul style="list-style-type: none"> • Balance Control Loop parameters can be modified by using an add-on microcontroller such as the Brain Board. • Balance Control Loop can be by-passed altogether to allow user's completely custom control loop.
Control	<ul style="list-style-type: none"> • All balance algorithms performed on-board. The Brain Board (included) only has to send simple commands to tell it whether to move forward/reverse or left/right, and how fast.
Development Environment	<ul style="list-style-type: none"> • Complete, professional-quality development environment included: <ul style="list-style-type: none"> — Full featured ISP device programmer for the ATMEGA32, for programming the Brain Board In-system — AVR-GCC open-source C-compiler — Programmer's Notepad IDE — Brain Board sample C code, with basic routines included to read sensors, write to LCD, communicate with serial port and read navigation buttons.
Balance Modes	<ul style="list-style-type: none"> • Modes of operation selectable with momentary switch and LED to display status and over communication bus.
Communications	<ul style="list-style-type: none"> • An I2C bus (100KHz) links the Balance Board to the Brain Board • A serial port and cable link the Balance Board to your personal computer.

Power Source	<ul style="list-style-type: none"> • 12 x AA Batteries. Alkaline or NiMH that able to supply voltage up to 18V.
Power Conversion	<ul style="list-style-type: none"> • Switching DC-DC converter provides efficiently converted power to on-board electronics as well as add-on 5V electronics.
Dimensions	<ul style="list-style-type: none"> • 27" high x 9" wide x 5.4" deep
Weight	<ul style="list-style-type: none"> • Approximate 3.2 Lbs.

4.3.1 Balance Board

For our information, the Balance Board is the board whereby it includes of Balance Processor Chip (BPC), motor drivers, BEMF motor velocity sensing, switching power converter and expansion port as depicted in the appendix C2. However, it plays a vital role in ensuring the two-wheeled mobile robot always in dynamically stable condition especially during navigation in greater speed of movement.

The Balance Processor Chip (BPC) which can be considered as the main component in the Balance Board. This is due to the programmable system-on-a-chip contains analog and digital circuitry, as well as a microprocessor, RAM and FLASH memory that able to store the calibration data. The calibration data refers to the measurement output with the desired or set value in the system. Thus, it is responsible or function to determine each motor's velocity, each ground sensor's distance and to accordingly perform real-time control of the motors to keep the mobile robot in the upright position dynamically balanced. Apart from that, this chip can also accepts commands from an external source (over an I2C bus) in order to

move the mobile robot forward, reverse, turn left or right. More importantly, it will report key information such as motor velocities, robot angle and battery voltage [1].

4.3.1.1 Modes of Operation

Basically, the Balance Board consists of modes of operation which summarized in the table 4.2. For our acknowledge, the mode 0 until mode 2 can be easily used without programming. However, the mode 3 until mode 5 are similar to mode 0 until mode 2, but the only differentiate between them is that they allow custom control variables (K1, K2 and K3) to be used. Briefly to say that all these modes (0-5), the Balance Processor Chip (BPC) perform the real-time control which lead the two-wheeled mobile robot to successfully balance.

Mode 6 is considered as a very unique mode for the system. It is due to the BPC which does not try to balance the mobile robot under this mode of operation. Instead, it only reads and reports the sensors, and it controls motors according to what it is told to do over the I2C interface. Moreover, the mobile robot can only balance if the external controller has its own real-time control system under this mode. However, it is suitably for the user who wishes to develop their balancing control system.

Table 4.2: Modes of Operation in Balance Board

Mode	Name	Control variables used	Description	LED status
0	Normal Balancing	Default	Normal balance, free moving. (DEFAULT at power-on)	Solid
1	Sleep Mode	Default	Motors tri-sated	1 blink

2	Nap Mode	Default	Motors shut down until unit is bumped. Then switch to mode 0.	2 blink
3	Normal Balancing	Custom	Custom control variables are used, free moving	3 blink
4	Sleep Mode	Custom	Motor tri-sated (custom variables are loaded)	4 blink
5	Nap Mode	Custom	Motors shut down until unit bumped. Then switch to mode 3.	5 blinks
6	Custom control Algorithm mode	N/A	Dummy slave mode. Control algorithm is bypassed. I2C definitions are different.	6 blinks

4.3.2 Brain Board

Brain Board which shown in the appendix C3 is another important component that plays a crucial role in process the data obtained in the Balance Board in order to control the balancing of the mobile robot. However, this board consists of a microcontroller of ATMEGA32 which can be programmed in order to customize the two-wheeled balancing robot according to the user's need. Briefly to say that the Atmel ATMEGA32 microcontroller is considered the brain of the two-wheeled mobile robot system as it offers a lot of power to control it. In addition to that, additional features found in this board will be explained in further section of this chapter. The key features of this chip are listed below:

- i. 16 MIPS (with included 16 MHz resonator)
- ii. 32K bytes FLASH
- iii. 1K EEPROM
- iv. 2K internal SRAM

- v. 8 channel, 10-bit DAC
- vi. I2C interface
- vii. 4 PWM channels
- viii. 3 timer/counters

4.3.3 Full-featured programmer

The main function for the full-featured programmer is to load the program into the Atmel ATMEGA32 microcontroller on the Brain Board via the ISP programmer connector as displayed in the figure 4.4, that consists of a small circuit board which connected to a 25-pin parallel-port connector to connect with the PC as well as a 10-pin connector on the other end to connect with the Brain Board [1]. However, this stuff must be work together with the software of MEGA32ISP in order to perform loading program to the microcontroller.

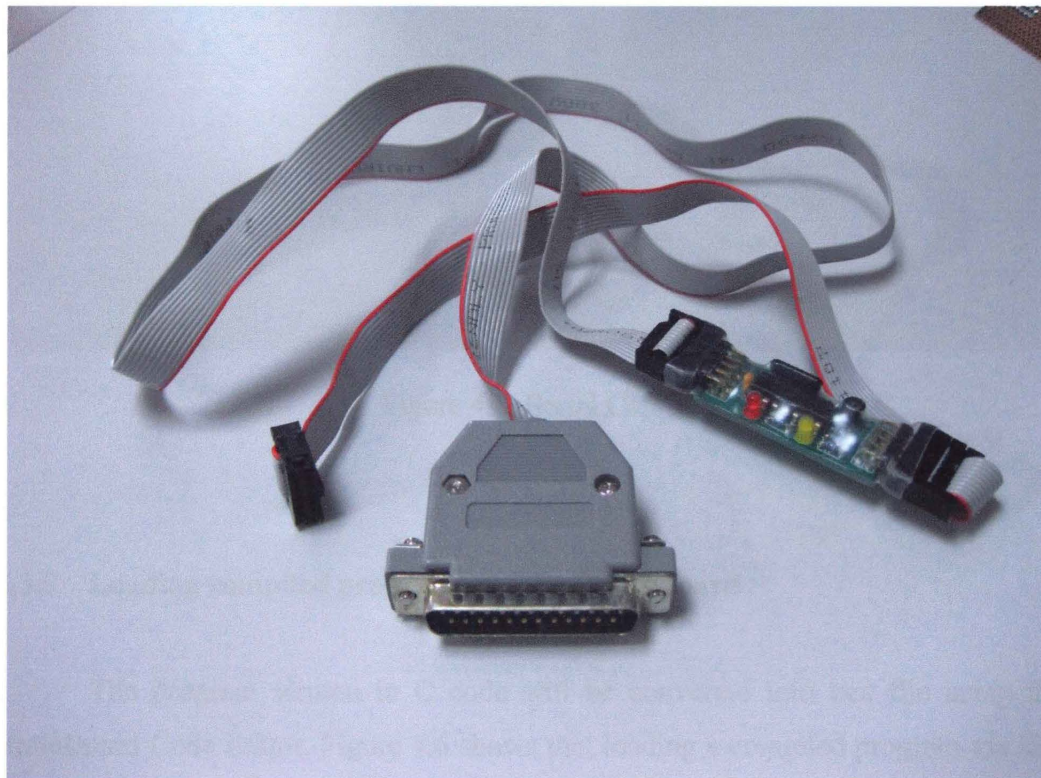


Figure 4.4: ISP programmer

4.3.4 RS232 Serial Cable

One of the features offered by the Brain Board is the serial communication to interface the Brain Board with the PC through the RS232 serial cable which shown in the figure 4.5. However, a simple terminal program such as HyperTerminal will be used to analysis the data of the two-wheeled balancing robot in terms of stability or balancing of it during the balancing mode as well as exploring mode for navigation purposes.



Figure 4.5: Serial Cable

4.3.5 Loading compiled program into the Brain Board

The program written in C code will be converted into hex file using the BrainBoard Code Editor. Figure 4.6 shows that loading a compiled program via ISP programmer. For our information, this ISP programming port uses PB5, PB6 and

PB7 on the ATMEGA32 microcontroller. Therefore, since these pins are already reserved for ISP programming, thus the users could not use these pins for other purposes. The MEGA32ISP is required in order to load the compiled program into the Brain Board.

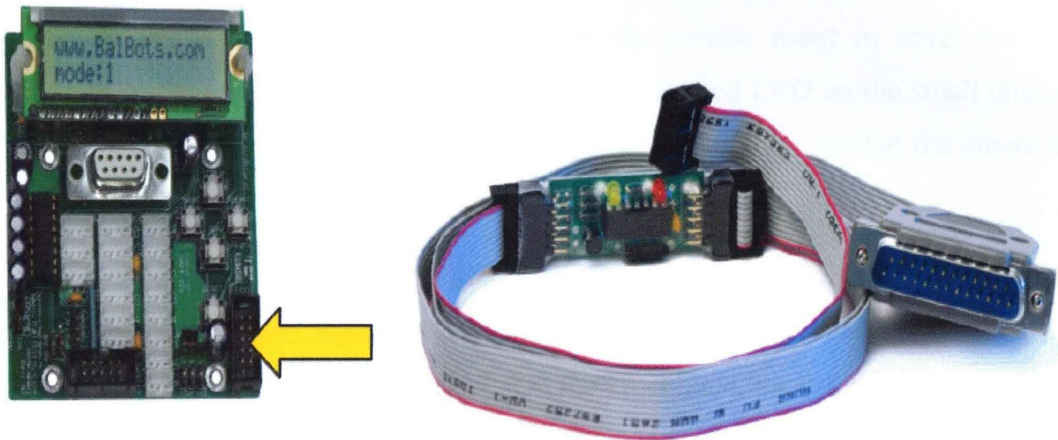


Figure 4.6: Loading compiled program into the Brain Board

Below are the procedures on how to program the ATMEGA32 on the Brain Board [1] which clearly illustrated in figure 4.7:

- i. The icon “MEGA32ISP” is clicked in order to open the programming software. However, this icon should be located under the “BalBot” programming group. Else, the program mega32isp.exe. is opted for the users to run.
- ii. The software should be opened up and look similar to the figure 4.11.
- iii. The port that the ISP programmer is chosen and is connected to the PC. By then, the number of ports available on the computer will be automatically found and usually is LPT1.
- iv. The download or loading speed can be adjusted or changed if necessary. For newer computers, the speed is recommended to be reduced.

- v. Followed by the memory area in which the code is to be downloaded is chosen. Usually this will be FLASH memory, where the program is stored. The EEPROM is typically used to store data values and other set-up parameter.
- vi. The power switch of the mobile robot is turned on and, hence the Brain Board must be received power from the mobile robot in order for the programming operation to work. However, the red LED on the small circuit board on the ISP programming cable will be lit to indicate that the power is received.
- vii. There are modes available to send or load the program to the ATMEGA32 microcontroller: Auto and Manual. As for Auto mode, several steps will be executed in a row automatically. On the other hand, for the Manual mode, the user is required to select the hex file to be downloaded to the microcontroller. The “Open Files” button is clicked and the balbot.hex file will be located, which was created earlier after being compiled. However, the default location for this file is: C:\BalBot\BrainBoard_Code\balbot.hex
- viii. Followed by erasing the memory in the microcontroller by clicking the “Chip Erase” button operated under the Manual section.
- ix. “Blank?” button is clicked in order to check or to make sure that the memory is really erased.
- x. The “program” button is clicked in order to download or load the program compiled in hex file to the ATMEGA32 microcontroller. During the program is being transferred, the yellow LED on the ISP programmer will light on. At the same time, the progress bar on the lower left of the window will also indicate the status of the transfer. Thus, check this to make sure program is transferred completely.
- xi. The contents of the microcontroller can be checked against the contents of the PC’s buffer once the message “programming complete” appeared. In order to do so, the “Verify” button is clicked.

- xii. The 10-conductor ISP ribbon cable from the Brain Board now can be unplugged. Thus, either the “reset” button on the Brain Board is pressed or turned off and the power from the Balance Board is turned on.
- xiii. The new program should be running.

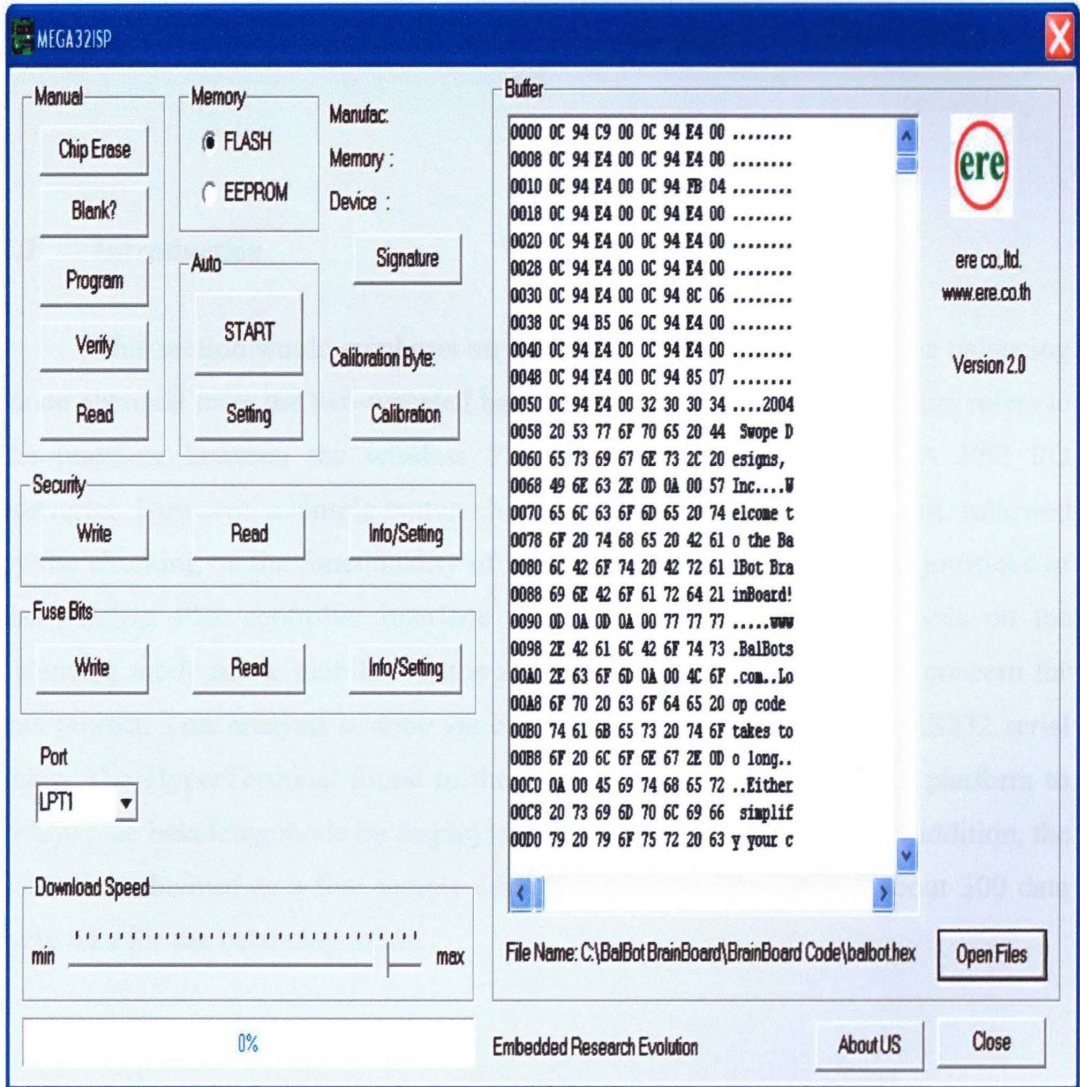


Figure 4.7: Steps for loading a compiled program

CHAPTER 5

RESULT AND DISCUSSION

5.1 Introduction

This section would emphasize on the hardware testing as well as the balancing mode obtained from the two-wheeled balancing robot. The hardware testing refers to the interface between the wireless PS2 controller with the PSC28A PS2 I/O converter. However, a simple button checked procedures to be carried out, followed by the checking on the functionality of the both left and right of analog joysticks of the wireless PS2 controller interface with RC motors. Instead, analysis on the balancing mode of the mobile robot is also part and parcel of our result concern for this project. This analysis is done via collecting the data by using the RS232 serial cable. The HyperTerminal found in the window will be chosen as the platform to analyse the balancing mode by displaying the data in term of figures. In addition, the data were obtained as a five sample data for every one second and about 300 data were take for the balancing mode.

5.2 Hardware Testing

This section would discuss in detail on how to test our hardware purchased to ensure it is in good condition before implement or apply them to our project. The hardware testing consists of wireless PS2 controller, robot chassis equipped with RC motors and PSC28A PS2 I/O converter. In brief, this hardware testing procedures begin with the simple testing on the selected buttons to be used to for our project

later on. Followed by the testing on the both left and right sides of analog joysticks of the wireless PS2 controller with the robot chassis that equipped with RC motors.

5.2.1 Testing on buttons

The stage of testing includes of wireless PS2 controller, PSC28A, LEDs and resistors. For our information, the PS2 controller consists of 14 buttons. However, only four selected buttons will be test for this stage as it will be used for our project, namely Up, Down, Left and Right buttons as shown in the figure 5.1. Thus, remember that the outputs of the PSC28A are commonly active high (+5V) and will become active low (0V) once the button is pressed on the wireless PS2 controller.

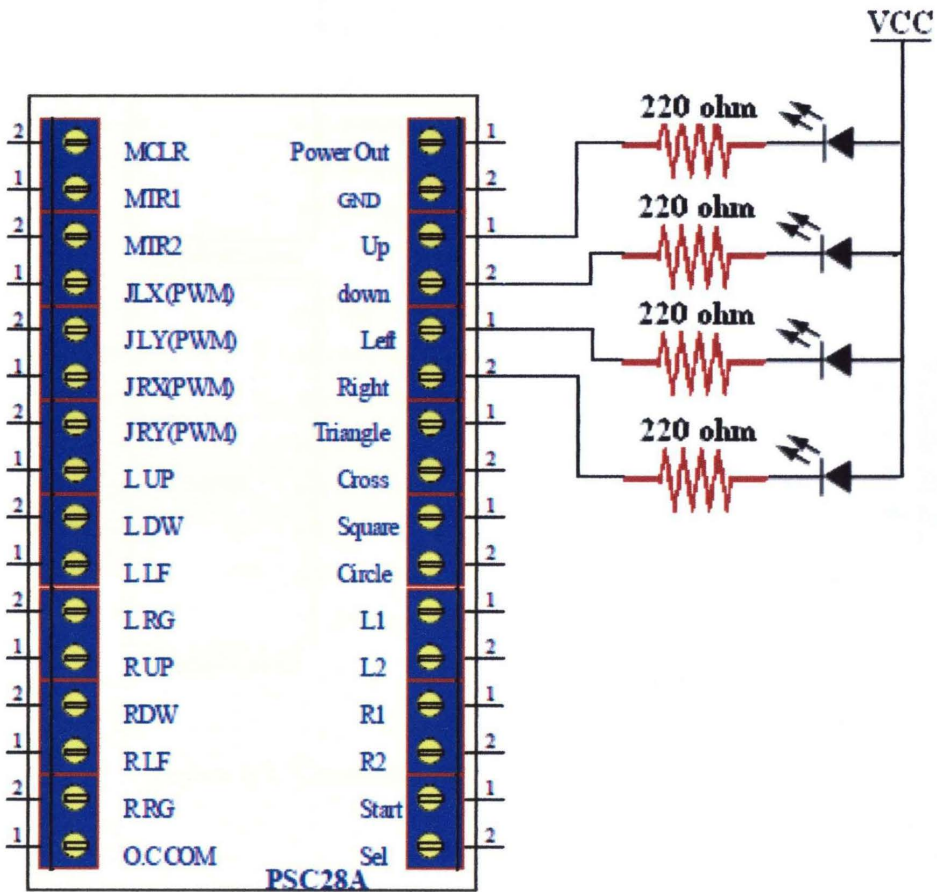


Figure 5.1: Connection for Up, Down, Left and Right buttons testing

5.2.2 Testing on analog joysticks

The testing on the both sides of analog joysticks will be explained in detail in this section. However, two motor drivers are required instead of PSC28A and wireless PS2 controller. Each motor will be connected to the PSC28A to drive one side of RC motor. Other components are needed for running this test comprises of resistors and capacitors. However, each analog joystick has two axes: X axis and Y axis. The figure 5.2 shows the schematic on the connection between the RC motor and PSC28A, while the result of implementation testing on analog joysticks is displayed in figure 5.3.

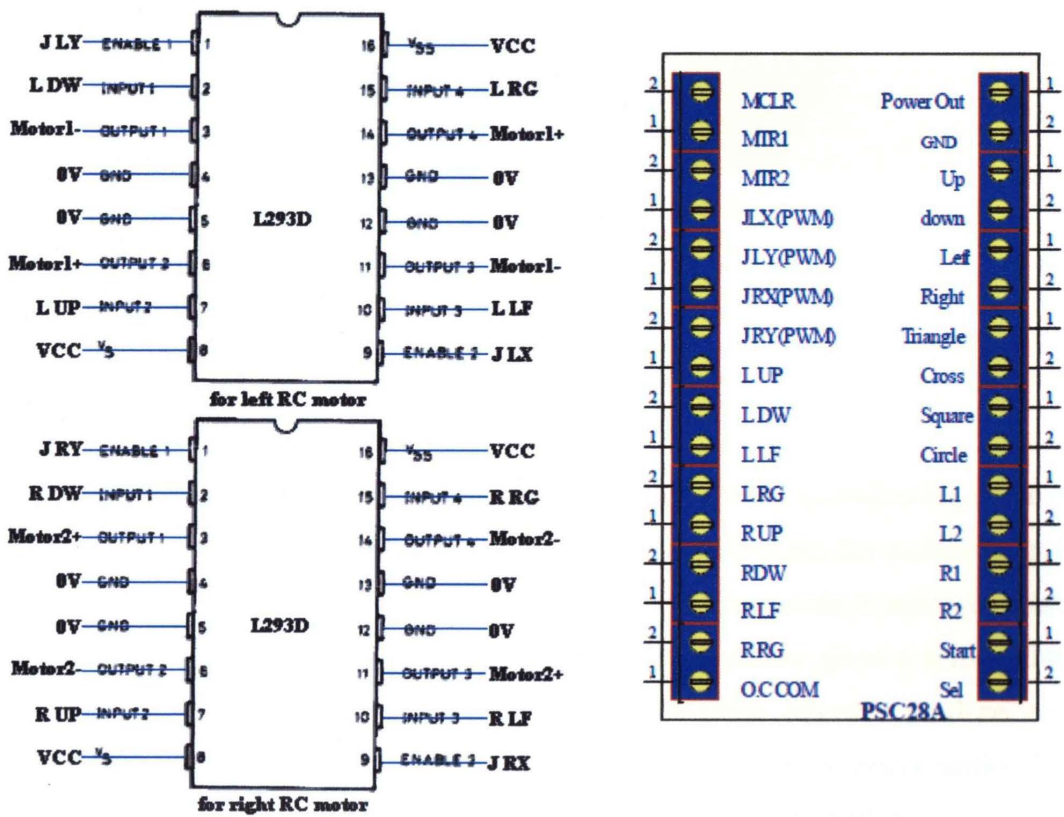


Figure 5.2: Connection of RC Motors and PSC28A

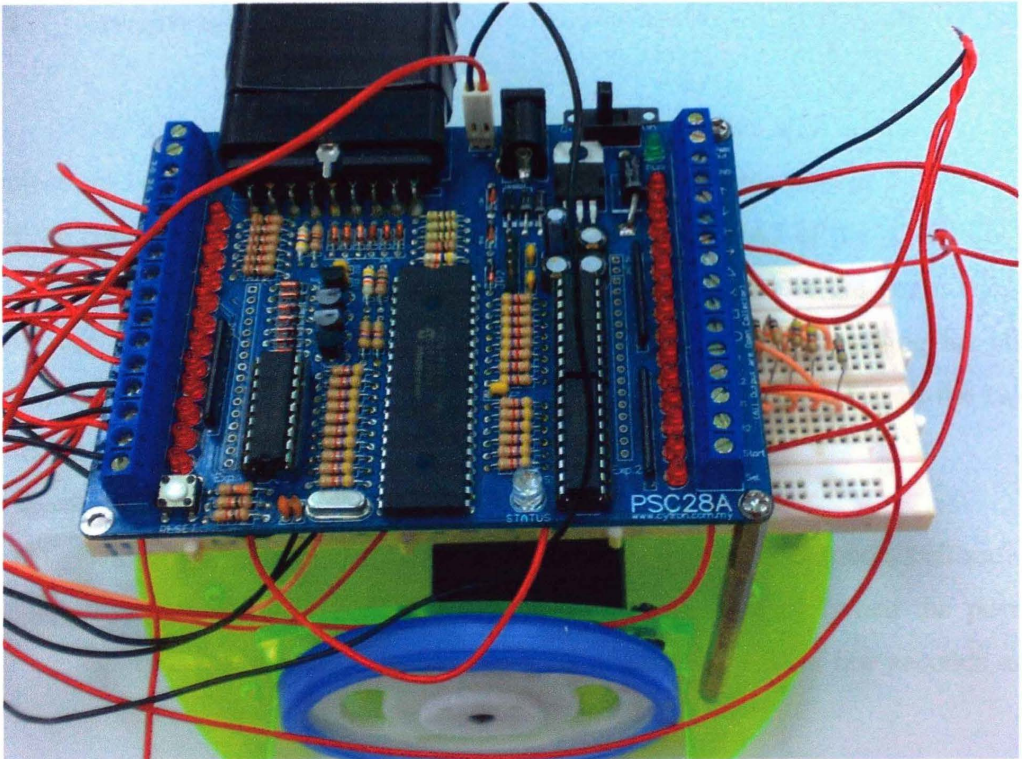


Figure 5.3: Testing on analog joysticks

5.3 Balancing Mode

This section would discuss on the result obtained from the balancing mode of the two-wheeled balancing robot. This is the mode initially when the mobile robot is turned on. However, the stability of the system in balancing mode is achieved when both the analog reading of GP2D120 infrared distance sensor gives a same value. Hence, these two values will be subtracted at each other whereby produce zero reading in distance (cm) indicates that the system is stabilized. In this condition, the two-wheeled mobile robot balances on its own body and always in place on center of gravity. In order to do this analysis, HyperTerminal is required as well as RS232 serial cable. The data obtained in terms of figures will be displayed in the HyperTerminal screen and the RS232 serial cable will be used to connect the mobile robot with the PC.

5.4 Navigation Mode

The four buttons of the wireless PS2 controller that will be used for this project are Up, Down, Left and Right buttons. However, these pins of the PSC28A will be connected to the PORT A of PA4, PA5, PA6 and PA7 of the ATMEGA32 microcontroller via connect them to the Digital/Analog inputs of the Brain Board. However, bear in mind that there are must be sharing a same grounding for both PSC28A and Brain Board. Below are the steps to communicate between the two-wheeled balancing robot and the wireless PS2 controller:

- i. Before we started to navigate the two-wheeled mobile robot, calibration procedure must be carried out in order it will be stable when the power supply is on. It is done by putting it on a flat terrain for a few seconds and calibrate under balancing mode.
- ii. Turn on the PSC28A device and the Rx LED of the wireless PS2 receiver will be blinking indicates it is searching for PS2 controller. While the power of the wireless PS2 controller is being turned on, the Rx LED of the receiver will become static indicates the PSC28A device is communicating with the wireless PS2 controller.
- iii. Followed by turn on the power supply of the two-wheeled balancing robot. Switch the mode from balancing mode into explore mode for navigation purposes.
- iv. By then, the user could navigate the mobile robot wirelessly through the wireless PS2 controller by pressing the Up, Down, Left and Right buttons.

CHAPTER 6

CONCLUSION AND RECOMMENDATION

6.1 Conclusion

In a nutshell, this project has successfully introduced or applied the concepts of the two-wheeled balancing robot in terms of the problem, research requirement as well as related current technology. Briefly to say that this project emphasis on the wireless communication of the two-wheeled mobile robot via remote control as well as the balancing control of the robot. More importantly, the project does include a comprehensive and up-to-date literature review.

Chapter 1 summarized that the background of this project and give a brief idea on how it works or functions. Instead, it described in detail regarding to the problem statement, objectives and scopes of the project. More importantly, this chapter presents on how is the project to be carried out throughout the year from the planning stage towards the implementation work.

Chapter 2 discusses on the previous project regarding to the two-wheeled balancing mobile robot that have been done. This however gives us the basic idea on how the balancing of the two-wheeled robots work and provide relevant information on control systems and mathematical modelling and simulation that have been implemented.

Chapter 3 explains on the methodology of the project which touch on how is the project being carried out as well as introduction to the external hardware that will be applied for this project including explanation and related schematics of the

hardware. As such, wireless PS2 controller and PSC28A PS2 I/O converter will be included under this section as well as how the wireless communication to be done.

Chapter 4 presents the hardware specification of the two-wheeled balancing robot. However, this chapter would explain the detailed information about the basic operation of the mobile robot and the roles played by the Brain Board and Balance Board throughout the system. Besides, it also includes the software implementation in terms of installation, C-code compiling and loading the compiled program to the Atmel ATMEGA32 microcontroller using ISP programmer and MEGAISP32 software.

Chapter 5 touches on the result of the two-wheeled balancing robot in term of balancing via the balance mode. However, it further explains for the navigation system for the two-wheeled mobile robot in terms of how to interface the wireless PS2 controller with the robot as well as how operate them.

As an overall conclusion, this project represents the development and design of the navigation system for the two-wheeled balancing robot that able to communicate wirelessly using remote control. A successfully two-wheeled balancing robot system should include of having ability to handle the balancing problem and effective wireless communication with the remote control. In brief, the results obtained also meet the requirement of the PID controller in order to support the performance of analysis whereby the two-wheeled robot can achieve dynamically stable. Finally, the robot can be navigated well using the remote control.

6.2 Recommendations for Future Work

There are some recommendations and suggestions that can be applied in order to improve the performance of the two-wheeled mobile robot as well as adding extra features to improve the overall system:

- i. Develop own controller to provide greater stability
- ii. Utilize the function of vibrator motor of the wireless PS2 controller and the infrared distance sensor detect the obstacles under certain distance to notify the user not to move forward
- iii. Add more safety protection to avoid the sensor from being broken
- iv. Use wireless network for collect the real time data for analysis purpose especially while the mobile robot navigated wirelessly using remote control
- v. Utilize the analog joystick to take over the function of Up, Down, Left and Right buttons to control the navigation of the two-wheeled balancing robot
- vi. Make use of the available buttons in the wireless PS2 controller to expand the functions of the two-wheeled balancing robot

6.3 Costing and Commercialization

The costing for this project includes of PSC28A PS2 I/O converter and the wireless PS2 controller which shown in the table 6.1. However, the balbot is provided by supervisor. This project is generally can be implemented suitably on subject at the university for learning purposes due to its cost and the project itself greatly related to control system development.

Table 6.1: Project Costing

No.	Product	Cost
1	PSC28A	RM 168.00
2	Wireless PS2 Controller	RM 80.00
	Total	RM 248.00

As this project emphasis on developing a navigation system that able to control the balbot wireless via remote control in some distance, therefore it is a good experience if implemented on subject related to control system at the university. As such, variety of control system such as LQR, fuzzy logic and pole-placement controllers can be realized into this balbot.

The target of commercialization for this project is focus on the robot hobbyists as well as industrial purposes in order to perform dangerous activities in the industrial plan navigated wirelessly via remote control. Instead, this project can be commercialized to other universities for learning and research purposes.

REFERENCES

- [1] Swope Design. Detailed User's Guide for BalBot Basic and BalBot Advanced. Sacramento, CA: *Swope Design, Inc.*. 6-40; 2004
- [2] Grasser, F., D'Arrigo, S., Colombi, S., and Rufer, A. Joe: A mobile Inverted Pendulum. Proceedings of the 2002 IEEE Trans. Electronics. Vol.49, no 1. February. Lausanne, Switzerland: IEEE, 2002. 107-104
- [3] nBot Balancing Robot. Available at:
<http://www.geology.smu.edu/~dpa-www/robo/nbot/>
- [4] 17th JAN 2003, Steve's Legway URL
<http://www.teamhassenplug.org/robots/legway/>
- [5] Peter Miller (2008). Building a Two Wheeled Balancing Robot. University of Southern Queensland: Thesis B. Engineering (Electrical and Electronics)
- [6] Zatil Hanan Binti Ahmad Lutpi (2009). Position Control of Two Wheeled Inverted Pendulum. Universiti Teknologi Malaysia: Thesis B. Electrical Engineering (Control & Instrumentation)
- [7] Herdawatie binti Abdul Kadir (Nov 2005). Modelling and Control of a Balancing Robot using Digital State Space Approach, Master Thesis, Universiti Teknologi Malaysia.
- [8] Shiroma, N., Matsumoto, O., Kajita, S. & Tani, K, 1996 'Cooperative Behaviour of a Wheeled Inverted Pendulum for Object Transportation', *Proceedings of the*

1996 IEEE/RSJ International Conference on Intelligent Robots and Systems '96, IROS 96, Volume: 2, 4-8 Nov. 1996 Page(s): 396 -401 vol.

- [9]New, A.A, Aung, W.P., and Myint, Y.M. Software Implementation of Obstacle Detection and Avoidance System for Wheeled Mobile Robot. *Proceedings of the 2008 World Academy of Science, Engineering and Technology*. August 15. Mandalay, Myanmar: IEEE. 2008. 572-577.
- [10]Sugihara, T, T Nakamura & Hirochika I., 2002 'Realtime Humanoid Motion Generation Through ZMP Manipulation based on Inverted Pendulum Control', *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, Washington D.C.
- [11]Ong Yin Chee, Mohamad Shukri b. Zainal Abidin(2006). Design and Development of Two Wheeled Autonomous Balancing Robot. Universiti Teknologi Malaysia: Thesis B. Electrical & Electronics Engineering
- [12]Ooi, Rich Chi (2003). Balancing a Two-Wheeled Autonomous Robot. University of Western Australia: Thesis B. Mechatronics Engineering.

APPENDIX A

Programming in two-wheeled balancing robot

Program in Atmel Microcontroller for 'Two-Wheeled Balancing Robot With Remote Control'.

```

/*****
*
* Title:          Main sample code for the BalBot BrainBoard
* Author:         www.BalBots.com (A division of Swope Designs, Inc.)
* Edited by:     Lee Kong Haur
* Software:      AVR-GCC 3.4.1
* Hardware:      BrainBoard (with ATMEGA32 running at 16MHz)
* Version:       1.0
*
*****/

#include "lcd.h"
#include <stdio.h>

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <avr/pgmspace.h>
#include "BB_lib.h"
#include "uart.h"
#include "i2c.h"

#define XTAL_CPU          16000000    // 16Mhz
#define UART_BAUD_RATE   19200      // 19200 baud
#define bal_proc          0x62       // I2C address of the Balance
                                     Processor (On the
                                     BalanceBoard) (0x62 or
                                     decimal 98)

// The following determines the period of the main loop. Multiply by 1.02mS
// to get actual loop period. So a value of 40 will give you a cycle rate of ~24.51 Hz
#define LOOP_PERIOD_mS 40            // Do NOT use a number lower than 28.
                                     // That could over-run the BPC I2C
                                     buffer.

unsigned char i, explore;

int left_sensor_distance, right_sensor_distance;

```

```

//***** Global Variables *****
volatile unsigned char timer2_count, sufficient_flag;

#define NUMBER_OF_LCD_INPUTS    3    // The number of different variables
                                     the user is allowed to change
#define TOTAL_NUMBER_OF_LCD_VARIABLES    11
// The first variables below are considered INPUTs (that is, the user can change them
  using buttons)
// The remainder can still be seen on the LCD, but the user cannot change them.
const char *LCD_variable_names[TOTAL_NUMBER_OF_LCD_VARIABLES] =
{ "setting", "Explore speed", "Steer speed", "L sensor", "R sensor", "Angle", "L
RPM", "R RPM", "voltage", "tmp_steer", "tmp_fwd"};

volatile signed int LCD_variables[TOTAL_NUMBER_OF_LCD_VARIABLES],
LCD_variable_index;

volatile signed int tmp_fwd, tmp_steer, bot_stuck,tmp_fwd_last, fwd_der;

FILE *uart;
FILE *lcd;

int
main (void)
{
    unsigned int c;

        // Initialize variables:
        LCD_variables[0] = 0;                // Default to normal Balancing setting
        LCD_variables[1] = 60;              // Explore speed (0 to 100)
        LCD_variables[2] = 70;              // Explore steer speed (0 to 100)
        DDRD  &= 0xBF;                      // Set port D, pin 6 as input (cycle input from
                                           BalanceBoard (1 = output, 0 = input)

        timer2_count=0;

        // Initialize timer2: (This is used to pace the main cycle rate)
        TCCR2 = 0x84;                       // Normal mode, clk/64 prescaler(4 uS / click)
        TIMSK |= 0x40;                      // Enable Timer2 Overflow Interrupt

        uart = fdevopen(uart_putc, uart_getc, 0);
        lcd = fdevopen(lcd_putc, 0, 0);

        // Initialize I2C
        DDRB = 0x3F;                        // Make I2C lines input
        TWI_first_init();

        // Initialize servos
        servo_A(0);
        servo_B(0);

```

```

servos_init();

// Initialize UART library, pass baudrate and avr cpu clock
uart_init( UART_BAUD_SELECT(UART_BAUD_RATE,XTAL_CPU) );
// (Frame format is: asynchronous, 8data, no parity, 1stop bit)

// now enable interrupt, since UART library is interrupt controlled
sei();

// Transmit strings from program memory to UART:
uart_putc('\r');
uart_putc('\n');
uart_puts_P("2004 Swope Designs, Inc.\r\n");
uart_puts_P("Welcome to the BalBot BrainBoard!\r\n\r\n");

// ***** LCD DISPLAY *****
lcd_init(LCD_DISP_ON);    // initialize display, cursor off
//lcd_init(LCD_DISP_ON_CURSOR);    // Optional: Turn cursor on.
//lcd_init(LCD_DISP_ON_CURSOR_BLINK);    // Optional: Make cursor blink.
lcd_clrscr();    // clear display and home cursor
// put string to display (line 1) with linefeed

lcd_puts_P("www.BalBots.com\n");

// move cursor to position 8 on line 2
lcd_gotoxy(7,1);

// 1 = output, 0 = input
DDRB &= 0x0F;    // Configure B7:B4 as inputs (Momentary Switches)
PORTB |= 0xF0;    // Enable pull-ups on B7:B4

//*****Main Loop. This will repeat indefinitely at a rate determined*****
//*****by LOOP_PERIOD_mS *****
while(1) {
// timer2_count increments at the rate of 1.02mS between increments
// so a value of 32 will give you cycle rate of ~30.64 Hz
    while(timer2_count < LOOP_PERIOD_mS) {    // Wait for start flag
        sufficient_flag = 1;    // Raise flag to say that we actually had to
                                // wait at least a little, which means the loop
                                // code doesn't take too long
    }
    if(!sufficient_flag) {    // This will be true if your main loop code takes
                              // longer to run than the cycle rate you're trying to run
                              // at.
uart_puts_P("Loop code takes too long.\r\n");

```

```

uart_puts_P("Either simplify your code or increase LOOP_PERIOD_mS.\r\n");
}
sufficient_flag = 0;
timer2_count = 0;    // Reset flag. The Timer2 ISR will increment this so
                    // we will know how long we've been in the main loop.

// The following will check whether any buttons are
// pressed and then act accordingly.
// Caution! Because of shared programming pins, if the
// programmer is left plugged in, it will appear
// as though the left button is being held down.
// To avoid this, always unplug programmer from
// the BrainBoard after programming the board.
switch( check_buttons() ) {
    case 'L':                // Left held down
        uart_putc('-');
    case 'l':                // left pressed
        uart_putc('l');
        if(LCD_variable_index < 1)
            LCD_variable_index = (TOTAL_NUMBER_OF_LCD_VARIABLES - 1);
        else
            LCD_variable_index--;
        break;

    case 'R':                // Right held down
        uart_putc('-');
    case 'r':                // right pressed
        uart_putc('r');
        LCD_variable_index++;
        if(LCD_variable_index >= TOTAL_NUMBER_OF_LCD_VARIABLES)
            LCD_variable_index = 0;
        break;

    case 'U':                // Up held down
        uart_putc('-');
    case 'u':                // up pressed
        uart_putc('u');
        if(LCD_variable_index < NUMBER_OF_LCD_INPUTS)
            LCD_variables[LCD_variable_index] ++;
        break;

    case 'D':                // Down held down
        uart_putc('-');
    case 'd':                // down pressed
        uart_putc('d');
        if(LCD_variable_index < NUMBER_OF_LCD_INPUTS)
            LCD_variables[LCD_variable_index]--;
        break;
}

```

```

    default:                                // Nothing pressed
        //uart_putc(' '); // Put a space in place of the letter
        break;

}

// Transfer report variables from I2C buffer to LCD screen.
LCD_variables[3] = left_sensor_distance;    // Left forward-looking sensor
LCD_variables[4] = right_sensor_distance;  // Right forward-looking sensor
LCD_variables[5] = bal_proc_rd.angle;
LCD_variables[6] = bal_proc_rd.L_mtr_RPM/31; // Divide by 31 (gear ratio)
                                                to get wheel speed
                                                instead of motor speed
LCD_variables[7] = bal_proc_rd.R_mtr_RPM/31; // Divide by 31 (gear ratio)
                                                to get wheel speed
                                                instead of motor speed
LCD_variables[8] = bal_proc_rd.battery;     // Battery Voltage (in
                                                increments of 0.1V)

switch( LCD_variable_index ) {
    case 0:                                // if we're on the first variable (setting)
        lcd_gotoxy(0,0);                  // Goto Beginning of first line on LCD
        lcd_puts_P("www.BalBots.com\n");
        if(LCD_variables[LCD_variable_index] < 1) { //Balance
            LCD_variables[LCD_variable_index] = 0;
            bal_proc_wr.mode = 0;
            explore = 0;
            lcd_gotoxy(0,1);
            lcd_puts_P("Balancing... \n");// Beginning of second line
        } else if(LCD_variables[LCD_variable_index] == 1) { // Explore
            bal_proc_wr.mode = 0;
            explore = 1;
            lcd_gotoxy(0,1);
            lcd_puts_P("Exploring... \n");// Beginning of second line
        } else if(LCD_variables[LCD_variable_index] > 1) { // Sleep
            LCD_variables[LCD_variable_index] = 2;
            bal_proc_wr.mode = 1;
            lcd_gotoxy(0,1);
            lcd_puts_P("Sleeping... \n");// Beginning of second line
        }
        break;
    default:
        lcd_gotoxy(0,0);                  // Goto Beginning of first line on LCD
        lcd_puts_P("www.BalBots.com\n"); // Stores string in program memory
        lcd_gotoxy(0,1);                  // Beginning of second line
        fprintf(lcd,LCD_variable_names[LCD_variable_index]);
        // Display name of variable

```

```

fprintf(lcd,"%d      ",LCD_variables[LCD_variable_index]);
// Display value of variable in Decimal
break;
}

//***** Write to the Balance Processor Chip (BPC) using I2C *****
I2C_write(bal_proc,(unsigned char *)&bal_proc_wr.mode,10);

//***** Read from the Balance Processor Chip (BPC) using I2C *****
I2C_read(bal_proc,(unsigned char *)&bal_proc_rd.mode,18);

// Limit the explore speed setting to 0-100
if(LCD_variables[1] < 0) {
    LCD_variables[1] = 0;
} else if(LCD_variables[1] > 100) {
    LCD_variables[1] = 100;
}
tmp_fwd *= LCD_variables[1];           // Take into account the explore speed
                                     // setting (0-100)
tmp_fwd /= 100;

// Limit the explore steer speed setting to 0-100
if(LCD_variables[2] < 0) {
    LCD_variables[2] = 0;
} else if(LCD_variables[2] > 100) {
    LCD_variables[2] = 100;
}
tmp_steer *= LCD_variables[2]; // Take into account the Explore Steer speed
                               // setting (0-100)
tmp_steer /=100;

if(tmp_fwd > 127 )
    tmp_fwd = 127;
if(tmp_fwd < -128)
    tmp_fwd = 128;
if(tmp_steer > 127 )
    tmp_steer = 127;
if(tmp_steer < -128)
    tmp_steer = 128;

LCD_variables[9] = tmp_steer;           // Display this variable to the LCD
                                       // (when user selects it)
LCD_variables[10] = tmp_fwd;           // Display this variable to the
                                       // LCD (when user selects it)

// Format and display the following variables on a PC through the serial port

```

```
fprintf(uart,"%5d %5d %5d %5d %5d\r\n",left_sensor_distance,
right_sensor_distance, fwd_der, tmp_steer, tmp_fwd);
```

```
if(explore) // If we're told to explore
{
    DDRA &=0x00;
    PORTA |=0xF0;
    if((PINA&0x10)==0)
    {
        bal_proc_wr.fwd_rev=50;
        bal_proc_wr.steer=0;
    }
    else if((PINA&0x20)==0)
    {
        bal_proc_wr.fwd_rev=-50;
        bal_proc_wr.steer=0;
    }
    else if((PINA&0x40)==0)
    {
        bal_proc_wr.fwd_rev=50;
        bal_proc_wr.steer=-60;
    }
    else if((PINA&0x80)==0)
    {
        bal_proc_wr.fwd_rev=50;
        bal_proc_wr.steer=60;
    }
}
else
{
    bal_proc_wr.fwd_rev = 0;
    bal_proc_wr.steer = 0;
}
```

```
// ***** Operate Servos *****
// This is silly code to move a servo towards the direction
// the bot wants to go to. This way, if you mount the
// servo on top of the bot and put a little toy head on
// it, it will have a bit of a personality.
// The two servo ports will be driven exactly the opposite
// of each other, so you can choose
//
// The input to the servo functions is a value from 0 to 2000
servo_B(1000 + (tmp_steer * 7) ); // Drive servo B according to which way
// we are steering to.
```



```

servo_A(1000 - (tmp_steer * 7)); // Drive servo A according to which way
                                we are steering to, opposite of servo B

// *****
// ***** Get character from UART *****
// ***** Then place it in corner of LCD screen *****

    c = uart_getc();
    if ( c & UART_NO_DATA ) {
// no data available from UART
    } else {
// new data available from UART
// check for Frame or Overrun error
if ( c & UART_FRAME_ERROR ) {
// Framing Error detected, i.e no stop bit detected
    uart_puts_P("UART Frame Error: ");
}
if ( c & UART_OVERRUN_ERROR ) {
// Overrun, a character already present in the UART UDR register was
// not read by the interrupt handler before the next character arrived,
// one or more received characters have been dropped
    uart_puts_P("UART Overrun Error: ");
}
if ( c & UART_BUFFER_OVERFLOW ) {
// We are not reading the receive buffer fast enough,
// one or more received character have been dropped
    uart_puts_P("Buffer overflow error: ");
}
// send received character back through UART
    uart_putc( (unsigned char)c );
// move cursor to position 16 on line 1
    lcd_gotoxy(15,0);
// Print the character on the LCD
    lcd_putc((unsigned char)c );
}

} //*****End of Main Loop *****

//*****

} // End of Main

//*****
//*****                               Interrupt Service Routines                               *****

```

/**** Choose from the following possible Interrupt Vectors *****
Then use this name when declaring the ISR below**

!!Make sure you do not misspell these, as compiler will not flag an error!!

**SIG_INTERRUPT0
SIG_INTERRUPT1
SIG_INTERRUPT2
SIG_OUTPUT_COMPARE2
SIG_OVERFLOW2
SIG_INPUT_CAPTURE1
SIG_OUTPUT_COMPARE1A
SIG_OUTPUT_COMPARE1B
SIG_OVERFLOW1
SIG_OUTPUT_COMPARE0
SIG_OVERFLOW0
SIG_SPI
SIG_UART_RECV
SIG_UART_DATA
SIG_UART_TRANS
SIG_ADC
SIG_EEPROM_READY
SIG_COMPARATOR
SIG_2WIRE_SERIAL
SIG_SPM_READY**

***/**

// *** Timer2 Interrupt *******

// This timer is used to pace the main cycle. This executes every 1.02 mS.

SIGNAL(SIG_OVERFLOW2)

```
{  
    timer2_count ++;  
}
```

APPENDIX B

Sharp GP2D120 specifications

SHARP
GP2D120

GP2D120

General Purpose Type Distance Measuring Sensors

■ Features

1. Less influence on the color of reflective objects, reflectivity
2. Line-up of distance output/distance judgement type
Distance output type (analog voltage) : GP2D120
Detecting distance : 4 to 30cm
3. External control circuit is unnecessary

■ Applications

1. TVs
2. Personal computers
3. Amusement equipment
4. Copiers

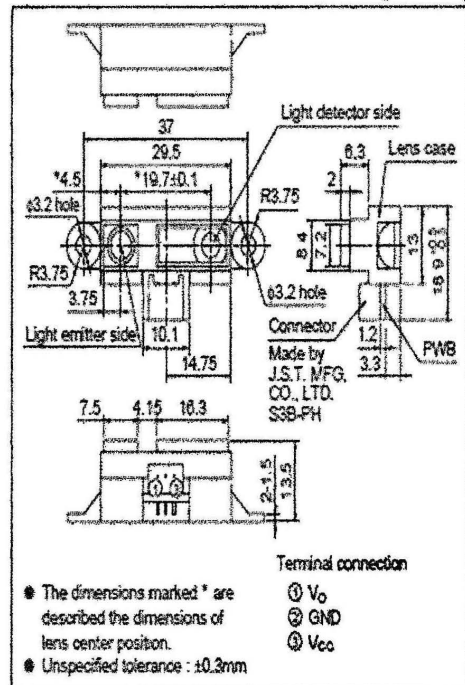
■ Absolute Maximum Ratings

(Ta=25°C, Vcc=5V)

Parameter	Symbol	Rating	Unit
Supply voltage	V _{CC}	-0.3 to +7	V
Output terminal voltage	V _O	-0.3 to V _{CC} +0.3	V
Operating temperature	T _{opr}	-10 to +60	°C
Storage temperature	T _{stg}	-40 to +70	°C

■ Outline Dimensions

(Unit: mm)



Notice In the absence of confirmation by device specification sheets, SHARP takes no responsibility for any defects that may occur in equipment using any SHARP devices shown in catalogs, data books, etc. Contact SHARP in order to obtain the latest device specification sheets before using any SHARP device.
Internet Internet address for Electronic Components Group <http://www.sharp.co.jp/ecg/>

■ Recommended Operating Conditions

Parameter	Symbol	Rating	Unit
Operating supply voltage	V _{CC}	4.5 to +5.5	V

■ Electro-optical Characteristics

(T_a=25°C, V_{CC}=5V)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Distance measuring range	ΔL	*1 *2	4	-	30	cm
Output terminal voltage	V _O	L=30cm ^{*1}	0.25	0.4	0.55	V
Difference of output voltage	ΔV _O	Output change at L=30cm to 4cm ^{*1}	1.95	2.25	2.55	V
Average Dissipation current	I _{CC}	L=30cm ^{*1}	-	33	50	mA

Note) L : Distance to reflective object.

*1 Using reflective object : White paper (Made by Kodak Co. Ltd. gray cards R-27 - white face, reflective ratio ; 90%)

*2 Distance measuring range of the optical sensor system

Fig.1 Internal Block Diagram

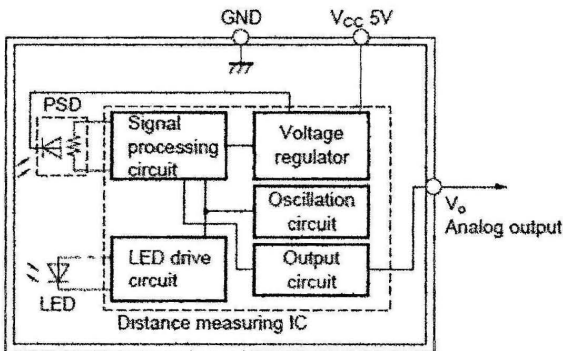


Fig.2 Timing Chart

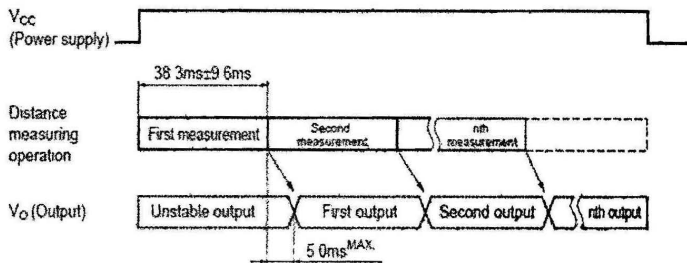


Fig.3 Analog Output Voltage vs. Surface Illuminance of Reflective Object

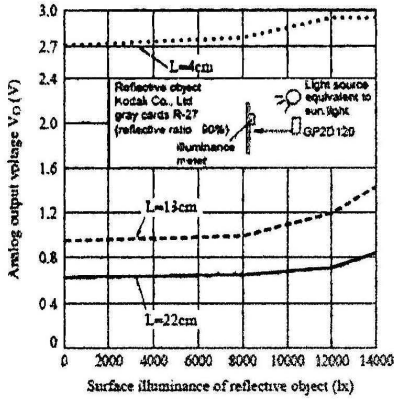


Fig.4 Analog Output Voltage vs. Distance to Reflective Object

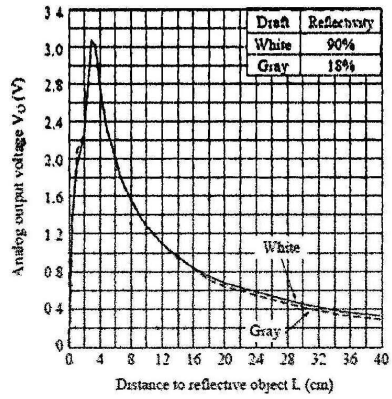


Fig.5 Analog Output Voltage vs. Ambient Temperature

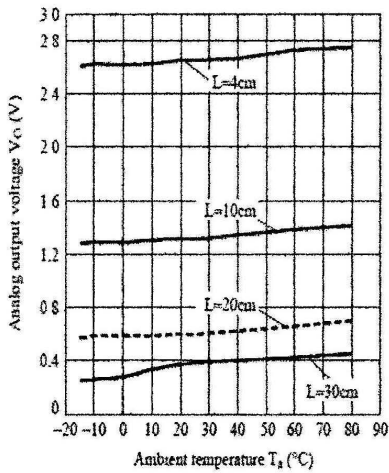
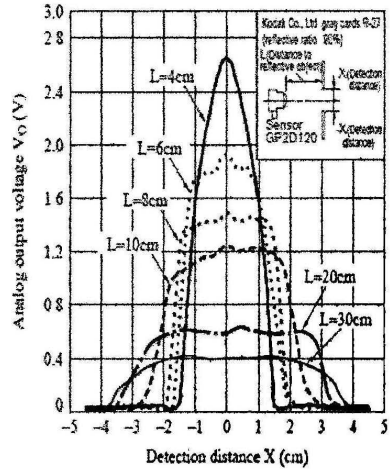
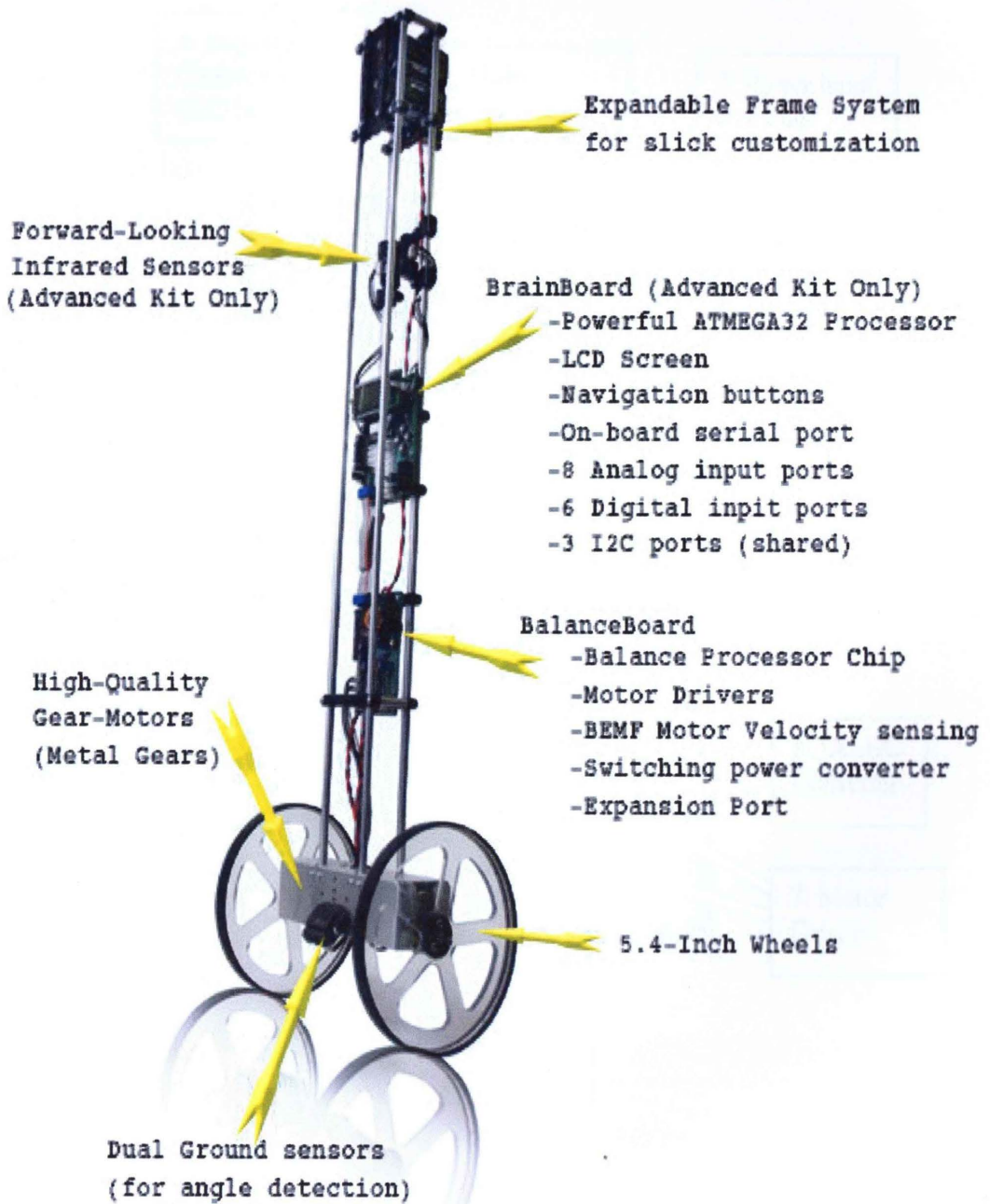


Fig.6 Analog Output Voltage vs. Detection Distance



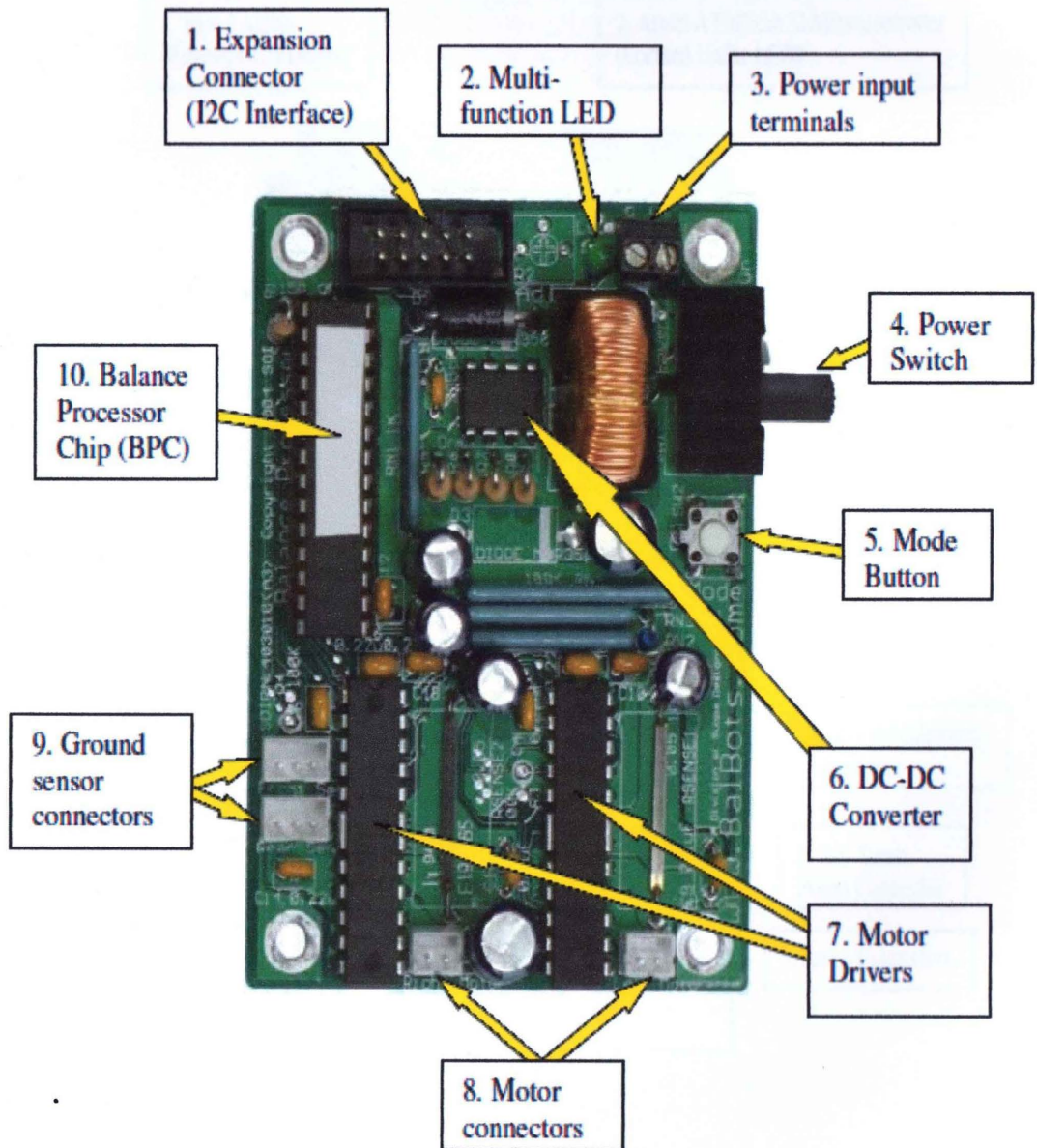
APPENDIX C1

Anatomy of Balbot



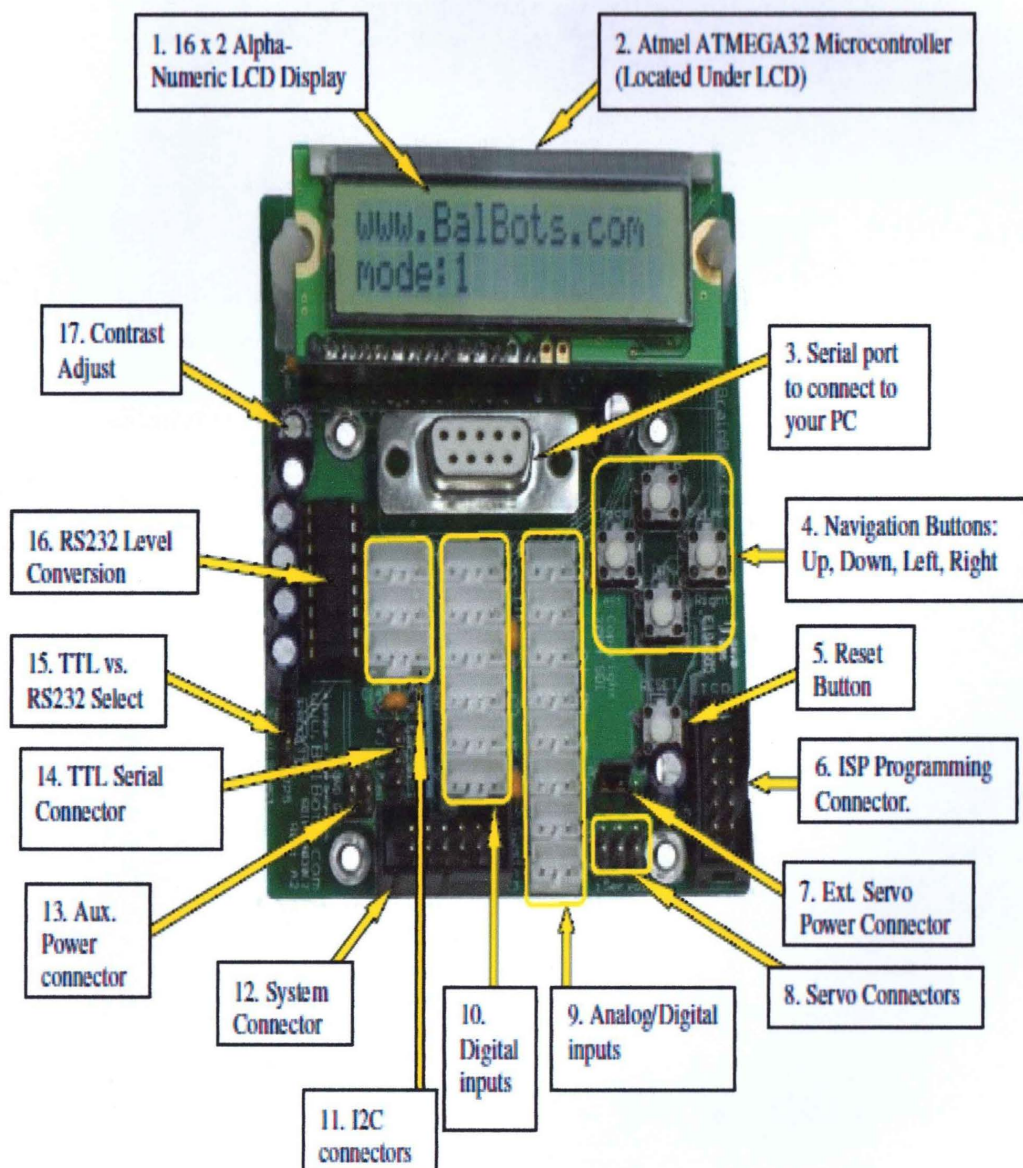
APPENDIX C2

Balance Board Overview



APPENDIX C3

Brain Board Overview



APPENDIX D

PSC28A I/O Converter and wireless PS2 receiver on Balbot

