Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Design and Parametric Control of Co-Axes Driven Two-Wheeled Balancing Robot

A thesis presented in partial fulfilment of
the requirements for the degree of

Master of Engineering
In
Mechatronics

at
Massey University
School of Engineering and Advanced Technology
Albany
New Zealand

Hamid Reza Memarbashi
2010

To my dearest mother, father and Lisa for their encouragement and support

Abstract

Nowadays robots can be seen in our daily life. Recently, robotic applications and their wide range of functionalities have drawn many engineers' attentions. Two-wheeled balancing robots are typical example of unstable dynamic system. Understanding the classical theory of inverted pendulum and its dynamic system are initial steps for developing a two-wheeled balancing robot. A balancing robot's structure has two different sections. The first section contains the moving parts or wheels and the second section contains the rigid parts or chassis. An initial physical structure was designed and built and robot's specifications were measured for developing the mathematical model of two-wheeled balancing robot. Existing energies of dynamic model were observed separately and substituted into Lagrangian equation to generate the mathematical model of balancing robot. Mathematical model was generated to observe the behaviour of the model. State-space model of robot was developed and a controller was designed according to state-space model. Tilt sensor and gyroscope provide the feedbacks of closed-loop system. Two-wheeled balancing robot has some key parameters that are directly engaged with system's performance and responses. Parametric studies were done and system responses were observed by variation of key parameters. Observed results from parametric studies were applied into physical model to improve the robot performance. Kalman filter was implemented for fusing the gyroscope angular rate and raw tilt angle. A proportional-integral-derivative (PID) controller was designed to generate the required input for motor controllers to control the rotation of wheels based on the Kalman filter's output.

Acknowledgements

I would like to thank Dr. Jen-Yuan (James) Chang for the supervision of my work and guidance along the way.

Table of Contents

Abstract i

Acknowledgements ii

List of Figures vii

List of Tables xi

1. Overview

1.1. Introduction 1

1.2. How Does Balancing Robot Balance Itself? 2

1.3. Literature Review 4

1.4. Conclusion 9

2. System Configuration

2.1. Introduction 10

2.2. Components 11

2.2.1. Microcontroller 11

2.2.2. Gyroscope 11

2.2.3. Accelerometer 12

2.2.4. Motor Controller 14

2.2.5. Gear-Head DC Motor 14

2.2.6. Batteries 15

2.3. Robot's Structure 16

2.4. Calibration 18

2.4.1. Tilt Sensor Calibration 18

2.4.2. Gyroscope Calibration 19

2.4.3. DC Motor and Motor Controller Calibration 20

2.5. Conclusion 22

3. Mathematical Modelling

3.1. Introduction 23

3.2. Mathematical Modelling of Two-wheeled Balancing Robot 24

3.3. Lagrangian Dynamic Analyses 25

3.3.1. Total Kinetic Energy of System 26

3.3.2. Total Potential Energy of System 29

3.3.3. Total Dissipation Energy of System 29

3.4. Robot's Specifications 33

3.4.1. Weights 33

3.4.2. Location of Centre of Gravity 33

3.4.3. Rotation Inertia of Chassis 34

3.4.4. Rotation Inertia of Wheel 34

3.5. State-Space Modelling 36

3.6. Conclusion 43

4. Controller Design (MATLAB & SIMULINK Implementations)

4.1. Introduction 44

4.2. Open-Loop Control System 45

4.2.1. Open-Loop Impulse Response 45

4.2.2. Open-Loop System Poles 46

4.3. Controllability and Observability 48

4.4. Closed-loop Control System 49

4.5. Conclusion 57

5. Parametric Studies

5.1. Introduction 58

5.2. Variation of μ_0 59

5.3. Variation of l 62

5.4. Variation of R 64

5.5. Conclusion 65

6. Results and Implementations

6.1. Introduction 66

6.2. Optimised Closed-loop Impulse Responses 68

6.3. Kalman Filter 69

6.4. PID Controller 73

6.5. Robot Performance 75

6.6. Conclusion 82

7. Overall Conclusion 83

References 84

Appendices

Appendix 1 87

Appendix 1.1 - iBot Functionalities 87

Appendix 1.2 - Hitachi Balancing Robot 88

Appendix 2 89

Appendix 2.1 - Arduino Duemilanove Specifications 89

Appendix 2.2 - Gyroscope CRS03 90

Appendix 2.3 - Memsic 2125 Dual Axis Accelerometer 91

Appendix 2.4 - HB-25 Motor Controller 92

Appendix 2.5 - EMG30 DC Motor 94

Appendix 2.6 - Physical Model 95

Appendix 3 96

Appendix 3.1 - Bench Scale 96
Appendix 3.2 - Rotation Inertia of Robot's Chassis 97
Appendix 3.3 - Rotation Inertia of Robot's Wheel 98
Appendix 3.4 - Algebraic calculations for observing $\dot{X}_3$ and $\dot{X}_4$ 99

Appendix 4 103

Appendix 4.1 - m File 103
Appendix 4.2 - Eigenvalues of matrix A 108
Appendix 4.3 - Observing the C_M matrix 110
Appendix 4.4 - Observing the O_M matrix 111

Appendix 5 112

List of Figures

Figure 1.1 - Three different modes of balancing robot 2

Figure 1.2 - Balancing robot flow chart 3

Figure 1.3 - Segway i180 series (left) and iBot 4000 (right) 4

Figure 1.4 - EMIEW (left) and EMIEW2 (right) 5

Figure 1.5 - QA (left) and QB (right) 6

Figure 1.6 - nBot 7

Figure 1.7 - JOE 8

Figure 1.8 - Two wheeled platform with reaction wheel actuator 8

Figure 2.1 - Arduino Duemilanove USB module 11

Figure 2.2 - CRS03, single axis angular rate sensor 12

Figure 2.3 - Accelerometer heated gas chamber 13

Figure 2.4 - Memsic 2125 pulse output 13

Figure 2.5 - HB-25 input signal 14

Figure 2.6 - Developed two-wheeled balancing robot 16

Figure 2.7 - Schematic diagram of two-wheeled balancing robot 17

Figure 2.8 - Tilt sensor calibration jig 18

Figure 2.9 - Tilt sensor calibration graph 19

Figure 2.10 - Gyroscope full scale output 20

Figure 2.11 - DC motor calibration graph 21

Figure 3.1 - Dynamic model of two-wheeled balancing robot 24

Figure 3.2 - Observing the location of centre of gravity 33

Figure 3.3 - Block diagram of state-space equations 37

Figure 3.4 - State-space model block diagram 42

Figure 4.1 - Block diagram of open-loop system 45

Figure 4.2 - Open-loop impulse response 46

Figure 4.3 - Open-loop system poles 47

Figure 4.4 - Block diagram of closed-loop system 49

Figure 4.5 - Block diagram of full-state feedback control 49

Figure 4.6 - Closed-loop system poles 51

Figure 4.7 - Impulse response of robot displacement 52

Figure 4.8 - Impulse response of robot pitch angle 52

Figure 4.9 - Closed-loop SIMULINK model of balancing robot 53

Figure 4.10 - Closed-loop impulse response of robot displacement 54

Figure 4.11 - Closed-loop impulse response of robot pitch angle 54

Figure 4.12 - Closed-loop impulse response of robot velocity 55

Figure 4.13 - Closed-loop impulse response of robot angular velocity 55

Figure 5.1 - Controller gains vs. coefficient of friction (μ_0) 59

Figure 5.2 - Displacement impulse response vs. coefficient of friction (μ_0) 60

Figure 5.3 - Pitch angle impulse response vs. coefficient of friction 60

Figure 5.4 - Displacement impulse response vs. l 62

Figure 5.5 - Pitch angle impulse response vs. *l* 63

Figure 5.6 - Displacement overshoot vs. *R* 64

Figure 6.1 - Final design of two-wheeled balancing robot 66

Figure 6.2 - Displacement impulse response 68

Figure 6.3 – Pitch angle impulse response 68

Figure 6.4 - Estimated angle by Kalman filter 70

Figure 6.5 - Kalman filter's output in balance mode 70

Figure 6.6 - PID controller's block diagram 73

Figure 6.7 - Balancing robot in action 75

Figure 6.8 - Pitch angle vs. time 79

Figure 6.9 - Displacement vs. time 80

Figure 6.10 - Estimated angle vs. disturbance force 81

Figure A1.1 - iBot functionalities 87

Figure A1.2 - Hitachi balancing robots 88

Figure A2.1 - CRS03 pin definition 90

Figure A2.2 - Memsic 2125 dual axis thermal accelerometer 91

Figure A2.3 - Memsic 2125 pin definitions 91

Figure A2.4 - HB-25 motor controller 92

Figure A2.5 - HB-25 pin definitions 93

Figure A2.6 - EMG30 gear-head DC motor 94

Figure A2.7 - Dimension of plastic levels in millimetres 95

Figure A2.8 - Developed structure for two-wheeled balancing robot 95

Figure A3.1 - SJ-HS bench scale 96

Figure A3.2 - Robot’s chassis 97

Figure A3.3 – Robot’s wheel 98

List of Tables

Table 4.1 - Specifications of closed-loop impulse response 56

Table 6.1 - Raw and estimated angle 71

Table A2.1 - Arduino Duemilanove USB module specifications 89

Table A2.2 - CRS03 pin definition 90

Table A2.3 - HB-25 pin definitions 92

Table A2.4 - HB-25 specifications 93

Table A2.5 - EMG30 gear-head DC motor specifications 94

Chapter 1

Overview

MASSEY UNIVERSITY
School of Engineering & Advanced Technology

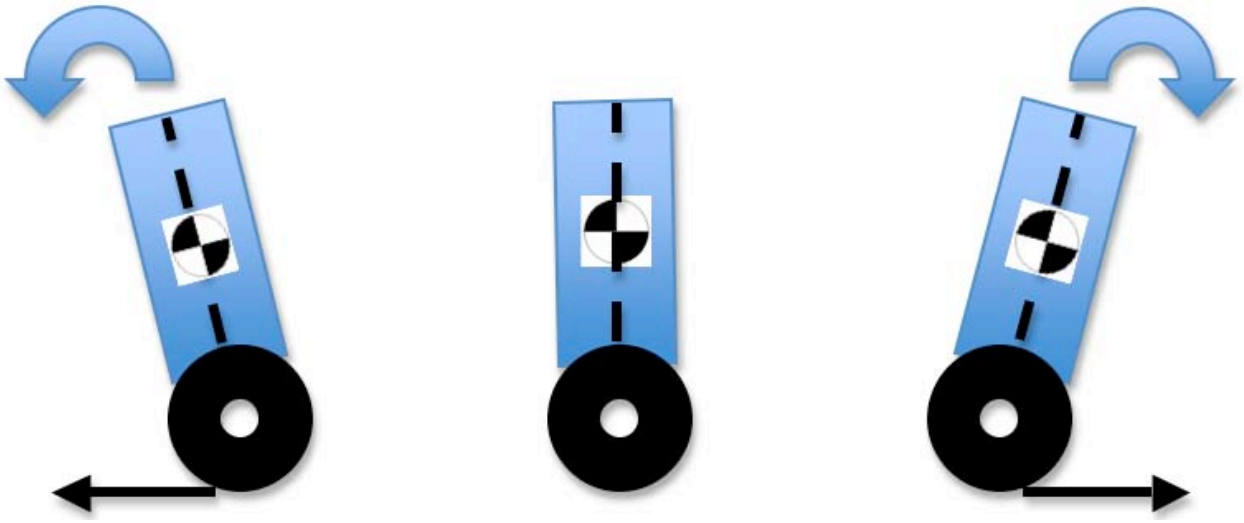
1.1. Introduction

Nowadays robots can be seen in our daily life. Robots are employed on production lines in factories, appeared as intelligent machines to do a specific task or used as commercial products. Recently, robotic applications and their wide range of functionalities have drawn many engineers' attentions. Two-wheeled balancing robots are typical example of unstable dynamic system.

Two-wheeled balancing robots can be used in several applications with different perspectives such as an intelligent gardener in agricultural fields (e.g. vineyards), an autonomous trolley in hospitals, shopping malls, offices, airports, healthcare applications or an intelligent robot to guide blind or disable people.

1.2. How Does Balancing Robot Balance Itself?

Two-wheeled balancing robot is an unstable dynamic system. It means that robot is free to fall forward or backward without any applied forces. Robot is balance when its centre of gravity and wheels are located on an identical imaginary vertical line. Otherwise, wheels should follow the robot's falls until robot balance itself. Using two wheels only for two-wheeled balancing robot provide lighter weight and smoother manoeuvre. Figure 1.1 illustrates the basic functions of balancing robot.



(a) Tilting backward (b) Balanced (c) Tilting forward

Figure 1.1 - Three different modes of balancing robot

Figure 1.2 shows the basic flow chart of balancing robot.

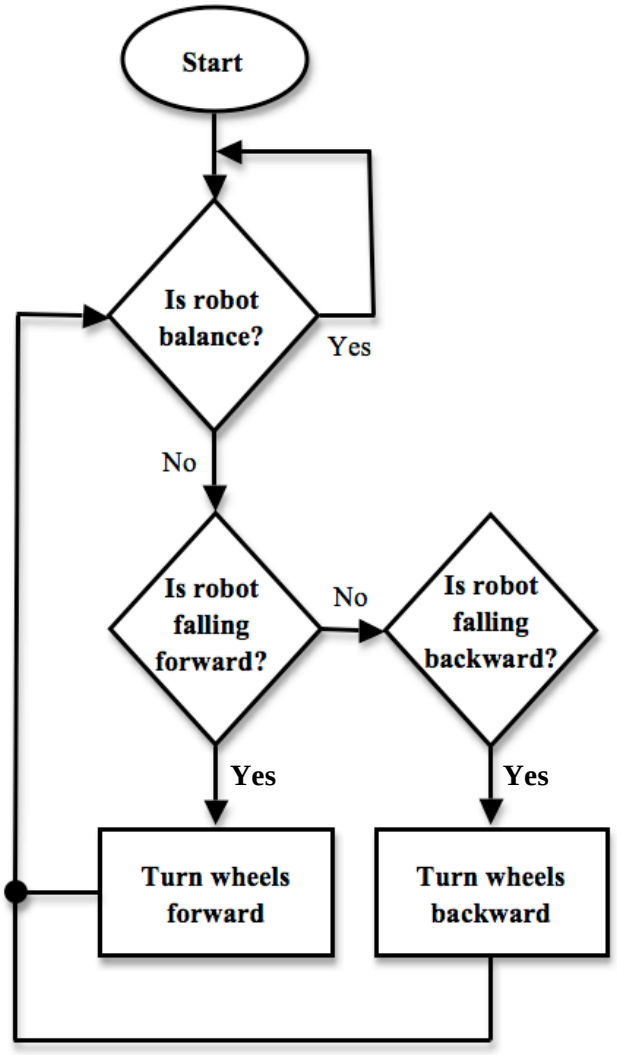


Figure 1.2 - Balancing robot flow chart

1.3. Literature Review

There are several successful commercial products and many other research prototypes around the world. Following paragraphs review some of products, researches and prototypes that built based on two-wheeled balancing robot basics.

Segway is a well-known self balancing personal transporter, which was produced by Segway Inc. of New Hampshire, USA [22]. It was invented by Dean Kamen in 2001. Segway is driven by servo motors and it can speed up to 20 km/h. Several tilt sensor and gyroscopes are employed in Segway to detect the angular motion of robot. Segway rider is able to accelerate and decelerate by leaning forward or backward. In 2006 iBot was another product that developed by Kamen [10]. iBot is a powered mobile wheelchair. It climbs the stairs and balances itself on two wheels. iBot increases its height when it balances on two wheels. Increasing the height provides an eye level vision for a disable person to communicate with other people. Figure A1.1 (Appendix 1.1) illustrates iBot functionalities.

Figure 1.3 - Segway i180 series (left) and iBot 4000 (right)

EMIEW is a humanoid balancing robot developed by Hitachi research group [8]. EMIEW stands for Excellent Mobility and Interactive Existence as Workmate. This robot is a guide or surveillance robot in a real world environment such as offices, factories or hospitals. Hitachi has developed two models of EMIEW. These two models can avoid obstacles and have a top speed of 6 km/h. Emiew2 was given an enhanced mobility by deploying a suspension system for each leg and voice recognition feature. Figure A1.2 (Appendix 1.2) illustrates more pictures of EMIEWs.

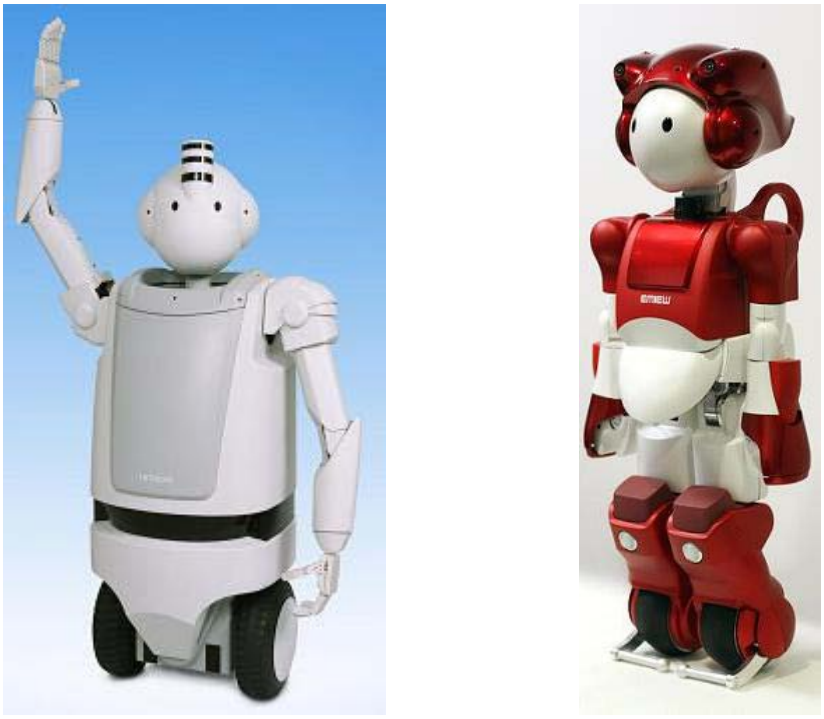


Figure 1.4 - EMIEW (left) and EMIEW2 (right)

Anybots is a robotic company founded by Trevor Blackwell in 2001 [2]. QA and QB are two mobile telepresence robots developed by Anybots that balance on two wheels and manoeuvre smoothly. These robots provide an easy communication via Internet for a user, who cannot attend at the second place. QB is the newest version of Anybot's telepresence robot. It can rotate around its vertical axis and drive at 5.6 km/h. Figures 1.5 illustrates QA and QB respectively.

Figure 1.5 - QA (left) and QB (right)

David P. Anderson has developed a prototype of two-wheeled balancing robot in his workshop [1]. Anderson was chosen four essential measurements that define the motion and position of nBot. These measurements are:

- Position of robot
- Velocity of robot
- Angular position of wheels
- Angular velocity of wheels

These measurements were summed as a feedback to microcontroller. Then microcontroller generates the required signal as a motor voltage, which is proportional to motor's torque.

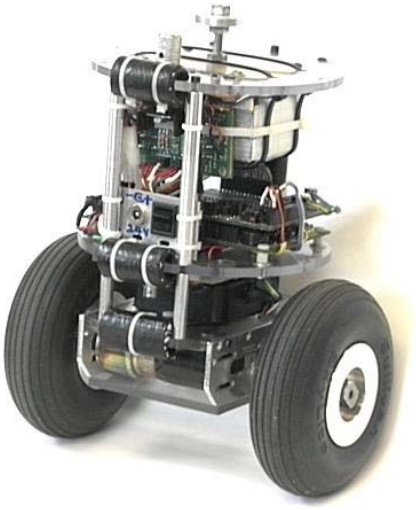


Figure 1.6 - nBot

The Industrial Electronics Laboratory at the Swiss Federal Institute of Technology in Switzerland has built a truly mobile and autonomous two-wheeled balancing robot (JOE) [6]. Researchers have developed two state-space controllers. One controller controls rotation around lateral axis (pitch) and the second one control the dynamics around its vertical axis (yaw). Each controller produce required torque for right and left motors.

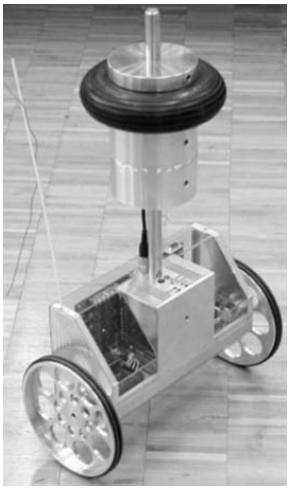


Figure 1.7 – JOE

All the mentioned projects and products (figures 1.3 - 1.7) use back and forth motion of wheels to balance the robot. There are some projects that use other methods to balance the structure. Department of mechanical engineering at university of Auckland in New Zealand used a reaction wheel on top of the robot to generate an external force to balance the robot [12]. The reaction wheel provides rotation inertia to the opposite direction of robot's fall and stabilise the system. Figure 1.8 illustrates the two wheeled platform with reaction wheel actuator.

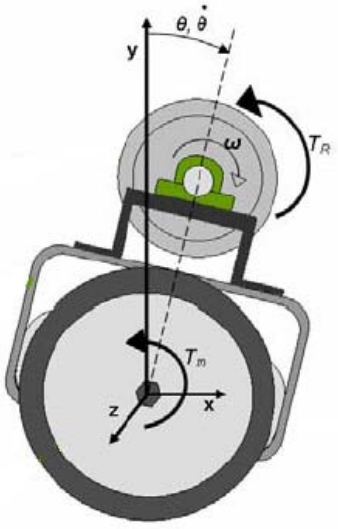


Figure 1.8 - Two wheeled platform with reaction wheel actuator

1.4 Conclusion

Two-wheeled robots provide higher level of mobility and manoeuvrability. The application of two-wheeled balancing robot varies with different environment and requirements. Understanding the classical theory of inverted pendulum and its dynamic system are initial steps for developing a two-wheeled balancing robot. Safety and robustness of two-wheeled products are so important and these aspects will be achieved by employing accurate sensors and designing a precise control unit.

Chapter 2

System Configuration

MASSEY UNIVERSITY
School of Engineering & Advanced Technology

2.1. Introduction

Two-wheeled balancing robot is a clear example of a natural unstable dynamic system. It means that robot falls forward or backward freely without having a control unit. Basically, it follows the inverted pendulum theory. A balancing robot's structure has two different sections. The first section contains the moving parts or wheels and the second section contains the rigid parts or chassis. An initial physical structure was designed and built. Then, robot's specifications were measured for developing the mathematical model of two-wheeled balancing robot. This chapter explains the system configuration it shows the robot's specifications.

2.2. Components

2.2.1. Microcontroller

An Arduino Duemilanove USB module was used as balancing robot's processor. This module is based on ATmega328 IC. It has 14 digital input/output pins, 6 analogue inputs and a 16MHz crystal oscillator. The module can be powered via the USB connection or an external power supply. The Arduino module selects the power source automatically and can be programmed with the Arduino software. Figure 2.1 illustrates the Arduino module.

Figure 2.1 - Arduino Duemilanove USB module

Table A2.1 (Appendix 2.1) illustrates the specifications of Arduino module [3].

2.2.2. Gyroscope

CRS03, single axis angular rate sensor was used to detect the angular velocity of robot. This sensor is manufactured by Silicon Sensing System Ltd [23]. It provides an analogue output, which is proportional to the angular rate about its sensing axis. The analogue output is between 0 to 5 volts. The output voltage is 2.5V (plus offset) at zero angular rate and 4.5V or 0.5V at full scale angular rate depending on direction of rotation. CRS03

sensor utilizes silicon micromachining and other solid state fabrication techniques and employs the Coriolis effect to measure angular rate. Figure 2.2 illustrates the CRS03 sensor.

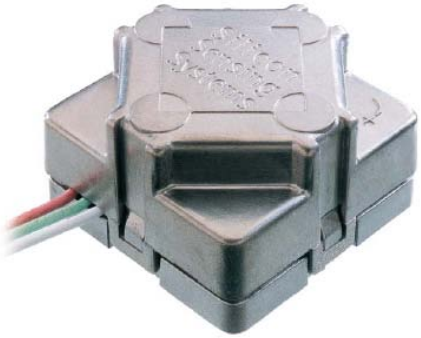


Figure 2.2 - CRS03, single axis angular rate sensor

CRS03 has a ratiometric output. It means that the sensor's output is a function of angular rate and supply voltage. Equation 2.1 shows the typical rate output of the sensor.

$$V_O = \frac{1}{2} \times V_{dd} + \left(R_a \times SF \times \frac{V_{dd}}{5} \right) \tag{2.1}$$

Where, V_O is rate output (V); V_{dd} is supply voltage (V); R_a is applied rate (°/s) and SF is scale factor (V/°/s). Table A2.2 (Appendix 2.2) illustrates the sensor's pin definitions.

2.2.3. Accelerometer

Memsic 2125, a dual axis thermal accelerometer was used to measure the tilt angle. This sensor is manufactured by Parallax, Inc and it is able to measure dynamic acceleration (vibration) and static acceleration (gravity) with a range of ±2 g [18]. Memsic 2125 has

two outputs that show the pulse output of g-force for X and Y axis. There is a small heater inside a chamber of gas that warms a bubble of air within the device (fig. 2.3). This bubble moves, when gravitational forces act on it. This movement is detected by four temperature sensors that are located around the chamber and onboard electronics convert the position of bubble into pulse output for the X and Y axis. The pulse outputs from the Memsic 2125 are set to a 50% duty cycle at 0 g. Duty cycle of pulse changes in proportion to acceleration of robot. Figure A2.2 and A2.3 (Appendix 2.3) illustrates the Memsic 2125 and its pin definitions.

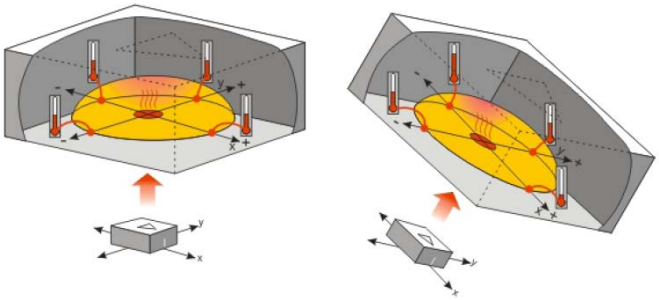


Figure 2.3 - Accelerometer heated gas chamber

Figure 2.4 illustrates the pulse output of Memsic 2125 and equation 2.2 illustrates the required formula for observing the g force [23].

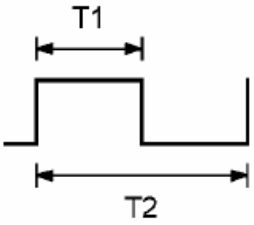


Figure 2.4 - Memsic 2125 pulse output

$$A(g) = ((T1/T2) - 0.5)/12.5\% \quad (2.2)$$

2.2.4. Motor Controller

Two HB-25 motor controllers were used to control the speed and direction of DC motors' rotation. This type of motor controller was controlled by a PWM signal. Figure 2.5 illustrates a typical form of input signal for HB-25 [19].

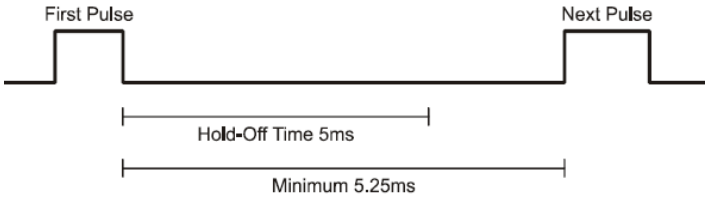


Figure 2.5 - HB-25 input signal

Figure 2.5 illustrates that there is a hold-off time of 5 ms and HB-25 ignores incoming pulses during this period. Therefore, the unit should not be refreshed more frequently than about 5.25 ms plus pulse time. Pulse time can be anywhere between 0.8 ms to 2.2 ms. HB-25 and its features are illustrated in appendix 2.4.

2.2.5. Gear-Head DC Motor

Two EMG30 gear-head DC motors were used for initial design. EMG30 is a 12V motor with gearbox 30:1. It produces the rate torque of 1.5 Kg/cm with 170 rpm. Appendix 2.5 illustrates the specification of EMG30 DC motor.

2.2.6. Batteries

A 12V-2.3AH battery was used for motor controllers and four AA batteries were used to power up the microcontroller.

2.3. Robot's Structure

Inverted pendulum theorem was followed to design and develop the two-wheeled balancing robot. In this project chassis was developed by four hard plastic levels (Figure A2.7) that were separated by M6 bolts and nuts (Figure A2.8) and heaviest parts (batteries) were located at the highest level of robot. Two co-axis DC motors were mounted under the chassis and other parts were distributed on levels. Figure 2.6 illustrates the initial design of balancing robot.

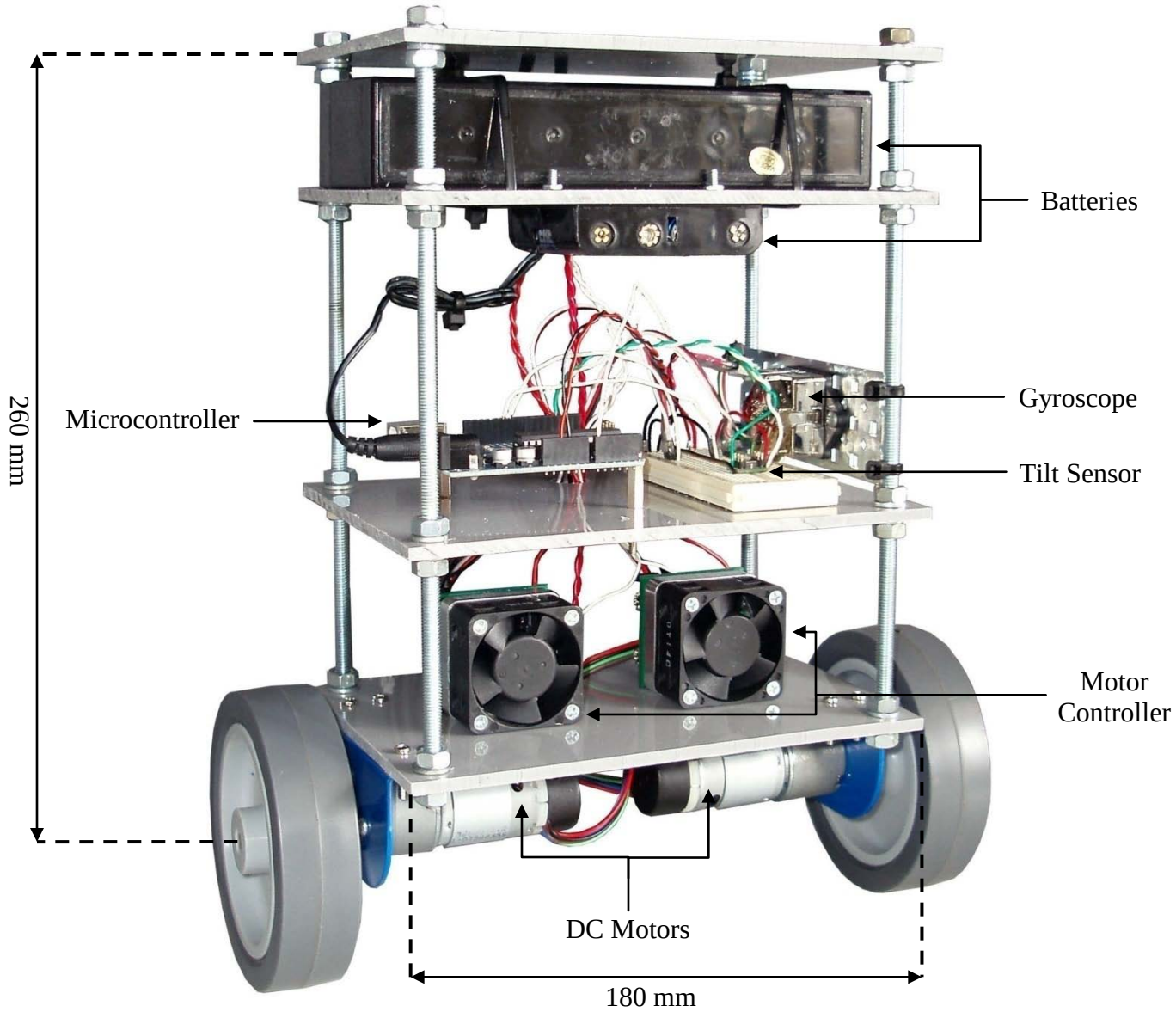


Figure 2.6 - Developed two-wheeled balancing robot

Figure 2.7 illustrates the schematic diagram of two-wheeled balancing robot and it shows how the different components are related together.

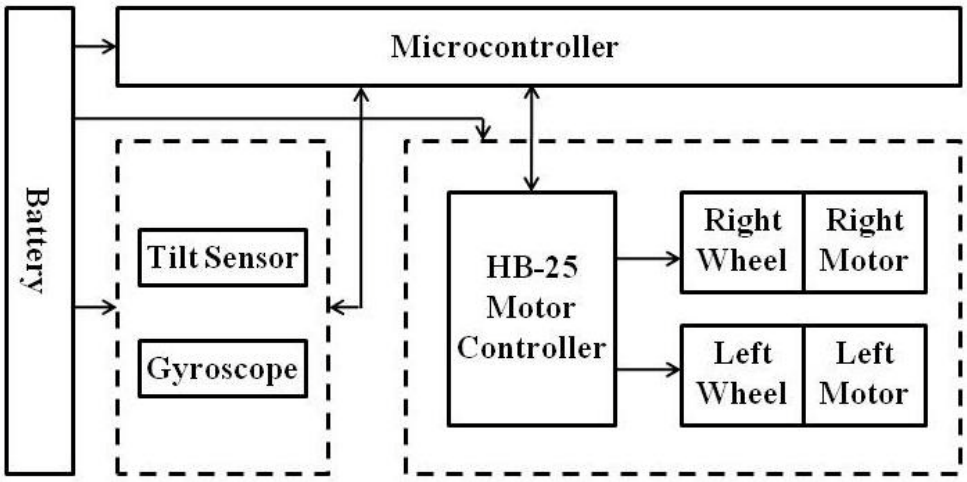


Figure 2.7 - Schematic diagram of two-wheeled balancing robot

2.4. Calibration

Tilt sensor, gyroscope and DC motor were calibrated before integrating with physical model. Calibration of these three parts provided better understanding of their outputs. Calibration results were used in programming the microcontroller.

2.4.1. Tilt Sensor Calibration

A home-made calibration jig was developed for calibrating the tilt sensor. Tilt sensor was located at different angles and sensor's output was observed as its pulse output of X axis g-force. Figure 2.8 and 2.9 illustrate the developed calibration jig and tilt sensor calibration's graph respectively.

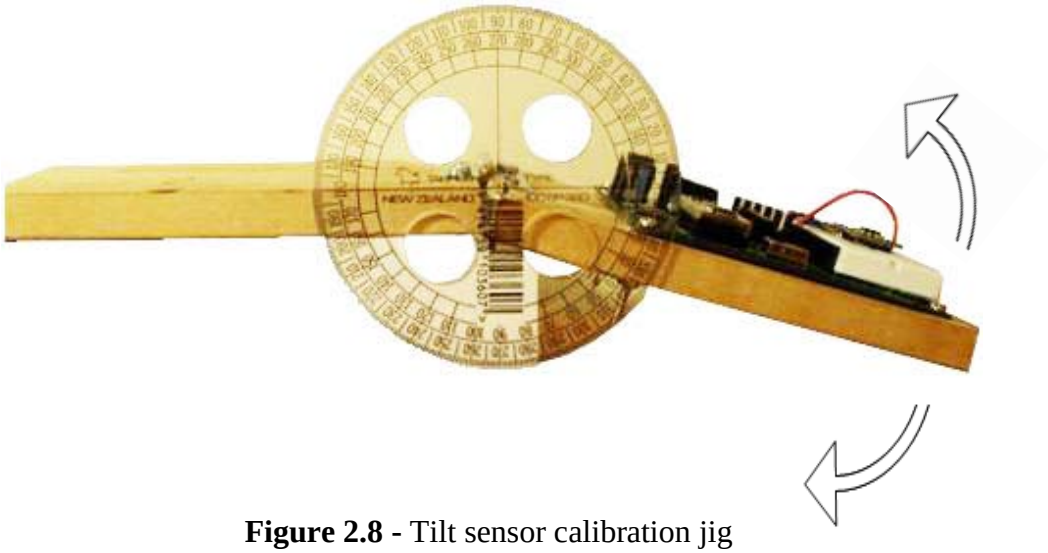


Figure 2.8 - Tilt sensor calibration jig

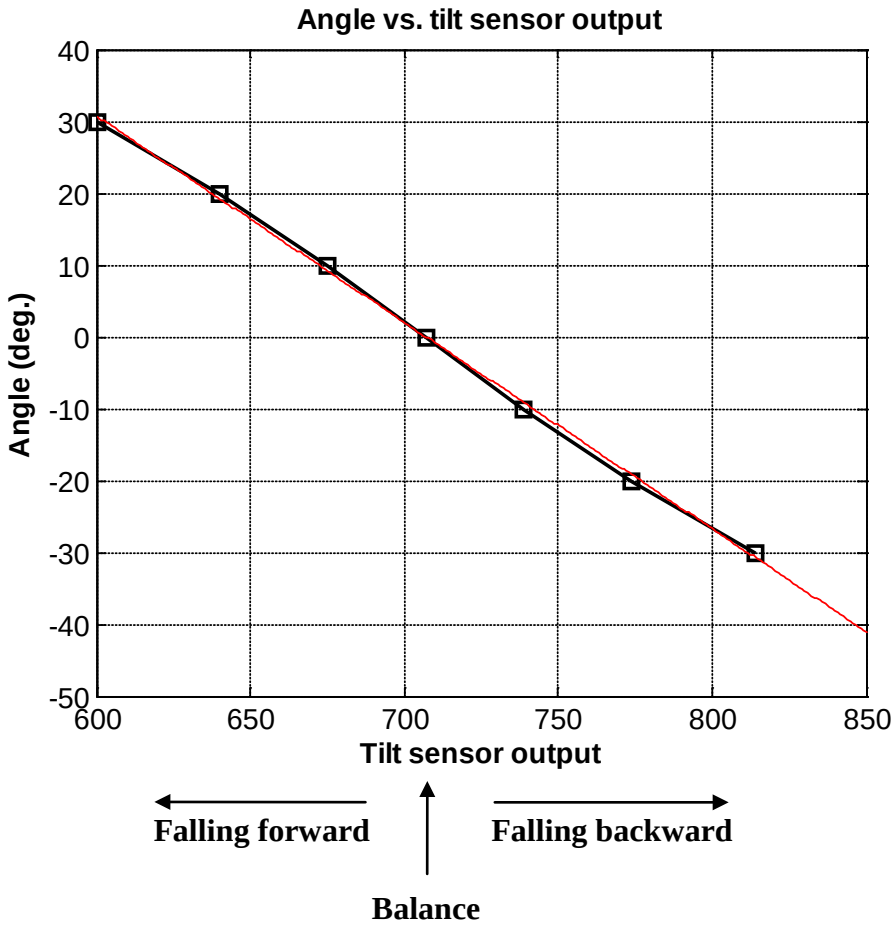


Figure 2.9 - Tilt sensor calibration graph

Equation 2.3 shows the equation of linear fitted line on tilt sensor's calibration graph.

$$y = -0.29x + 200 \tag{2.3}$$

2.4.2. Gyroscope Calibration

Full-scale output graph of gyroscope and its linear equation were observed. Figure 2.10 and equation 2.4 illustrates the full-scale output graph and its related equation respectively.

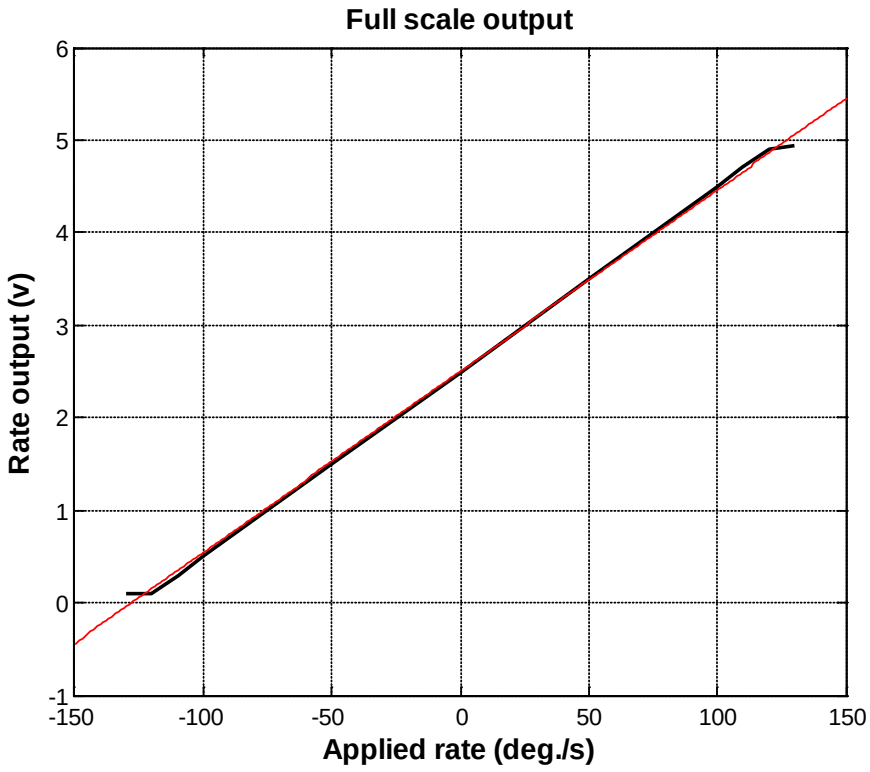


Figure 2.10 - Gyroscope full scale output

$$y = 0.02\,x + 2.5 \tag{2.4}$$

2.4.3. DC Motor and Motor Controller Calibration

An experiment was done to determine the corresponding duty cycle of PWM signal for desired rotation per minute (RPM) of DC motors. Figure 2.11 illustrates the DC motor calibration graph.

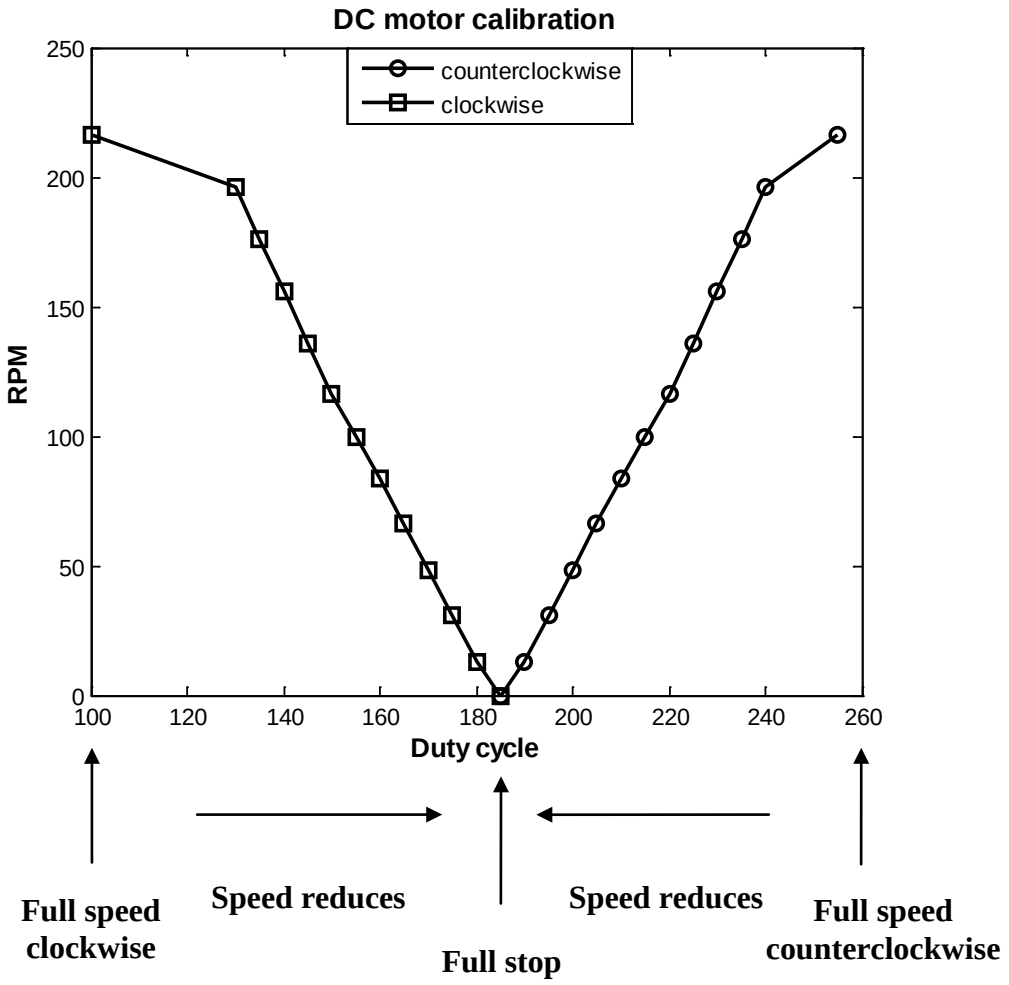


Figure 2.11 - DC motor calibration graph

2.5. Conclusion

Initial structure of two-wheeled balancing robot was developed. Symmetrical weight distribution on robot's chassis was an important step for designing the structure. Robot's weight could be reduced by using the lighter materials. Best location of centre of gravity and sensors were determined after mathematical simulation.

Chapter 3

Mathematical Modelling

MASSEY UNIVERSITY
School of Engineering & Advanced Technology

3.1. Introduction

Mathematical model of two-wheeled balancing robot was generated to observe the behaviour of model such as maximum overshoot, settling time and etc. Robot's specifications were observed and substituted into mathematical model. In this chapter existing energies of dynamic model were observed separately and substituted into Lagrangian equation to generate the mathematical model. Then, state-space model of system was design after developing the differential equations of system.

3.2. Mathematical Modelling of Two-wheeled Balancing Robot and Its Dynamic System

It is necessary to understand the dynamics of two-wheeled balancing robot for modelling the system. Figure 3.1 illustrates the dynamic model of two-wheeled balancing robot.

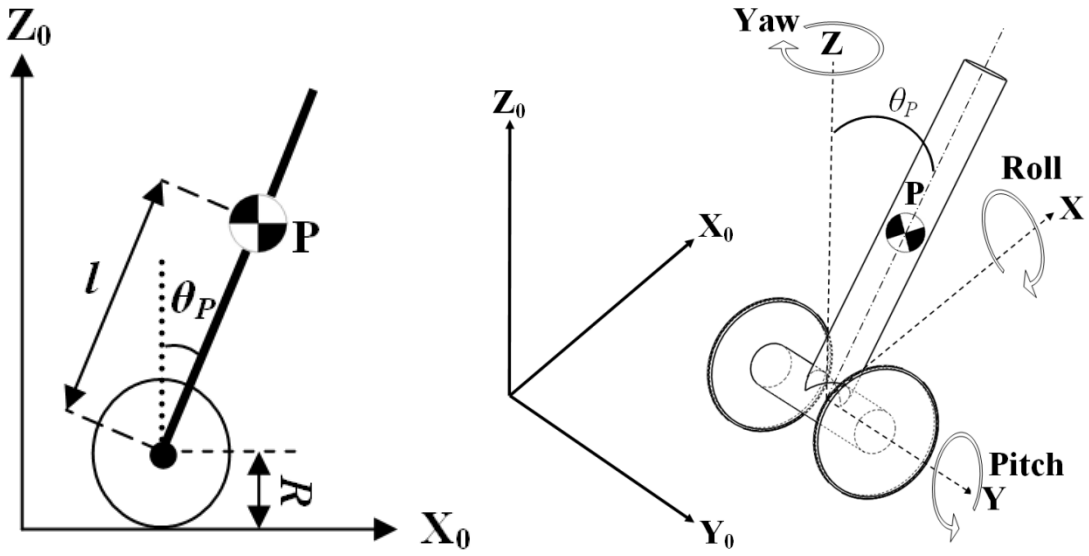


Figure 3.1 - Dynamic model of two-wheeled balancing robot

Two-wheeled balancing robot has three degrees of freedom (3DOF). Degrees of freedom are described by three types of robot's rotations around X, Y and Z axes, which are called roll, pitch and yaw respectively. In this project the main focus was on the rotation around Y axes (pitch). It assumed that centre of gravity of robot is located at point *P* and θ_P represents the pitch angle. The coordinate of point *P* will change if robot moves away from its initial location along X_0 axis. Therefore, equations 3.1 and 3.2 illustrate the displacement of point *P* along X_0 and Z_0 axes [11].

$$X_P = x + l \sin\theta_P \tag{3.1}$$

$$Z_P = l \cos\theta_P \tag{3.2}$$

3.3. Lagrangian Dynamic Analyses

Lagrangian approach was used to generate the ordinary differential equations (ODE) of system. Equation 3.3 shows the general form of Lagrangian equation [13].

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_r}\right) - \frac{\partial L}{\partial q_r} = Q_r \tag{3.3}$$

Where,

- $L=T-V$, is called the Lagrange function where, T is total Kinetic energy of the system from all inertia elements and V is total potential energy of the system from all stiffness elements and inertia elements due to gravitational effect.
- q_r $(r = 1, 2, 3, ..., n)$ are generalised coordinate/variable or displacement variable such as x_1, x_2, θ_1, θ_2 and etc.
- Q_r is sum of non-conservative forces projected on direction of generalised coordinate such as applied forces.

The first step for developing the Lagrangian equation of balancing robot was to observe the existing energies in the system. There are three types of energy for chassis and wheels. These energies are:

- Kinetic energy
  - Translational kinetic energy of chassis
  - Rotational kinetic energy of wheel
- Potential energy
- Dissipation energy

Equation 3.4 illustrates the Lagrangian equation of two-wheeled balancing robot where, E is kinetic energy, U is potential energy, F is dissipation energy and τ_r is required torque for left and right wheels.

$$\frac{d}{dt}\left(\frac{\partial E}{\partial \dot{q}_r}\right)-\frac{\partial E}{\partial q_r}+\frac{\partial F}{\partial \dot{q}_r}+\frac{\partial U}{\partial q_r}=\tau_r \tag{3.4}$$

3.3.1. Total Kinetic Energy of System

Translational kinetic energy of chassis (E_{TKC}) with constant mass and centre of gravity located at point P was observed by classical equation of kinetic energy shown on equation 3.5 [16].

$$E_{TKC}=\frac{1}{2}mv^2=\frac{1}{2}m\dot{X}_P{}^2=\frac{1}{2}m\left(\dot{X}_P{}^2+\dot{Z}_P{}^2\right) \tag{3.5}$$

Where, m is mass of robot's chassis and v is travelling speed of chassis. Equation 3.6 was developed by substituting equations 3.1 and 3.2 into equation 3.5.

$$E_{TKC}=\frac{1}{2}m\left[\left(v+l\omega_P\cos\theta_P\right)^2+\left(-l\omega_P\sin\theta_P\right)^2\right] \tag{3.6}$$

Equation 3.6 was simplified by following algebraic equations.

$$E_{TKC} = \frac{1}{2} m\left(v^2 + 2vl\omega_P \cos\theta_P + l^2 {\omega_P}^2 \cos^2\theta_P + l^2 {\omega_P}^2 \sin^2\theta_P\right)$$

$$E_{TKC} = \frac{1}{2} m\left(v^2 + l^2 {\omega_P}^2 + 2vl\omega_P \cos\theta_P\right)$$

Therefore, translational kinetic energy of chassis was defined by equation 3.7.

$$E_{TKC} = \frac{1}{2} m\left(v^2 + l^2 {\omega_P}^2\right) + mvl\omega_P \cos\theta_P \tag{3.7}$$

Rotational kinetic energy of chassis (E_{RKC}) was observed by equation 3.8.

$$E_{RKC} = \frac{1}{2} J_C {\omega_P}^2 \tag{3.8}$$

Where, J_C is rotation inertia of chassis and ω_P is angular velocity of point P. Total kinetic energy of chassis (E_{KC}) was observed by adding the translational and rotational kinetic energy of chassis.

$$E_{KC} = E_{TKC} + E_{RKC}$$

$$E_{KC} = \frac{1}{2} m\left(v^2 + l^2 {\omega_p}^2\right) + mvl\omega_P \cos\theta_P + \frac{1}{2} J_C {\omega_P}^2 \tag{3.9}$$

Translational and rotational kinetic energy of two wheels were observed by equations 3.10 and 3.11 respectively.

$$E_{TKW} = 2 \times \frac{1}{2} M v^2 = M v^2 \tag{3.10}$$

$$E_{RKW} = 2 \times \frac{1}{2} J_W \omega_W{}^2 = J_W \frac{v^2}{R^2} \tag{3.11}$$

Where, M and J_W are mass and rotation inertia of one wheel respectively. Total kinetic energy of wheels (E_{KW}) was observed by adding the translational and rotational kinetic energy of wheels.

$$E_{KW} = E_{TKW} + E_{RKW}$$

$$E_{KW} = M v^2 + J_W \frac{v^2}{R^2} \tag{3.12}$$

Total kinetic energy of two-wheeled balancing robot was observed by adding equations 3.9 and 3.12.

$$E = E_{KC} + E_{KW}$$

$$E = \frac{1}{2} m \left(v^2 + l^2 \omega_p{}^2 \right) + mvl \cos\theta_P \omega_P + \frac{1}{2} J_C \omega_P{}^2 + M v^2 + J_W \frac{v^2}{R^2}$$

$$E = \frac{1}{2}mv^2 + \frac{1}{2}ml^2\omega_p{}^2 + mvl\cos\theta_P\omega_P + \frac{1}{2}J_C\omega_P{}^2 + Mv^2 + J_W\frac{v^2}{R^2} \quad (3.13)$$

3.3.2. Total Potential Energy of System

There are no changes in potential energy of wheels while robot moves. Therefore, total potential energy of system (*U*) was defined by equation 3.14 [16].

$$U = mgl\cos\theta_P \quad (3.14)$$

Where, *m* is mass of chassis, *g* is gravity and *l* is distance between centre of gravity and rotational axis of wheels.

3.3.3. Total Dissipation Energy of System

Total dissipation energy of system (*F*) was defined by equation 3.15 [16].

$$F = \mu_0 v^2 + \mu_1\omega_P{}^2 \quad (3.15)$$

Where, μ_0 is coefficient of friction between wheel and ground and μ_1 is coefficient of friction between chassis and rotational axis of wheels. It was assumed that $\mu_0=0.1$ and $\mu_1=0$.

Displacement of balancing robot was selected as the first generalised variable (q_1) in Lagrangian equation of system.

$$q_1 = x \tag{3.16}$$

Following equations show the expansion of Lagrangian equation of system When $q_1 = x$.

$$\frac{\partial E}{\partial \dot{q}_1} = \frac{\partial E}{\partial \dot{x}} = \frac{\partial E}{\partial v} = mv + ml\cos\theta_P \omega_P + 2Mv + 2J_W \frac{v}{R^2}$$

$$\frac{d}{dt}\left(\frac{\partial E}{\partial v}\right) = m\ddot{x} + ml\left(\cos\theta_P\right)\ddot{\theta}_P + 2M\ddot{x} + 2J_W \frac{\ddot{x}}{R^2}$$

$$\frac{d}{dt}\left(\frac{\partial E}{\partial v}\right) = \left(m + 2M + \frac{2J_W}{R^2}\right)\ddot{x} + ml\left(\cos\theta_P\right)\ddot{\theta}_P \tag{3.17}$$

$$\frac{\partial E}{\partial q_1} = \frac{\partial E}{\partial x} = 0 \tag{3.18}$$

$$\frac{\partial F}{\partial \dot{q}_1} = \frac{\partial E}{\partial \dot{x}} = \frac{\partial F}{\partial v} = 2\mu_0 v \tag{3.19}$$

$$\frac{\partial U}{\partial q_1} = \frac{\partial E}{\partial x} = 0 \tag{3.20}$$

The first differential equation of system (eq. 3.21) was observed by adding the equations 3.17 to 3.20.

$$\left(m + 2M + \frac{2J_W}{R^2}\right)\ddot{x} + ml\left(\cos\theta_P\right)\ddot{\theta}_P + 2\mu_0 v = 0 \tag{3.21}$$

Pitch angle of balancing robot was selected as the second generalised variable (q_2) in Lagrangian equation of system.

$$q_2 = \theta_P \tag{3.22}$$

Following equations show the expansion of Lagrangian equation of system when $q_2 = \theta$.

$$\frac{\partial E}{\partial \dot{q}_2} = \frac{\partial E}{\partial \dot{\theta}_P} = \frac{\partial E}{\partial \omega_P} = ml^2\omega_P + mvl(\cos\theta_P) + J_C\omega_P$$

$$\frac{d}{dt}\left(\frac{\partial E}{\partial \omega_P}\right) = ml^2\ddot{\theta}_P + ml(\cos\theta_P)\ddot{x} + J_C\ddot{\theta}_P$$

$$\frac{d}{dt}\left(\frac{\partial E}{\partial \omega_P}\right) = (ml^2 + J_C)\ddot{\theta}_P + ml(\cos\theta_P)\ddot{x} \tag{3.23}$$

$$\frac{\partial E}{\partial q_2} = \frac{\partial E}{\partial \theta_P} = -mvl(\sin\theta_P)\omega_P \tag{3.24}$$

$$\frac{\partial F}{\partial \dot{q}_2} = \frac{\partial F}{\partial \omega_P} = 2\mu_1\omega_P \tag{3.25}$$

$$\frac{\partial U}{\partial q_2} = \frac{\partial U}{\partial \theta_P} = -mgl(\sin\theta_P) \tag{3.26}$$

The second differential equation of system (eq. 3.27) was observed by adding the equations 3.23 to 3.26.

$$ml\left(\cos\theta_P\right)\ddot{x}+\left(ml^2+J_C\right)\ddot{\theta}_P+mvl\left(\sin\theta_P\right)\omega_P+2\mu_1\omega_P-mgl\left(\sin\theta_P\right)=\tau_L+\tau_R$$

$$ml\left(\cos\theta_P\right)\ddot{x}+\left(ml^2+J_C\right)\ddot{\theta}_P+\left(mv\omega_P-mg\right)l\left(\sin\theta_P\right)+2\mu_1\omega_P=\tau_L+\tau_R \quad (3.27)$$

Equations 3.21 and 3.27 are two non-linear equations of two-wheeled balancing robot. Before any further analysis, differential equations of system were linearised. There is only one equilibrium point for balancing robot and it is when robot is balanced or up-right. Therefore, it was assumed that $\theta_P = 0$. So,

$$\sin\theta_P \approx \theta_P \quad (3.28)$$

$$\cos\theta_P = 1 \quad (3.29)$$

Linear equations of two-wheeled balancing robot were observed by substituting equations 3.28 and 3.29 into non-linear equations. Equations 3.30 and 3.31 show the linear equations of system.

$$\left(m+2M+\frac{2J_W}{R^2}\right)\ddot{x}+ml\ddot{\theta}_P+2\mu_0 v=0 \quad (3.30)$$

$$ml\ddot{x}+\left(ml^2+J_C\right)\ddot{\theta}_P-mgl\theta_P+2\mu_1\omega_P=\tau_L+\tau_R \quad (3.31)$$

3.4. Robot's Specifications

Linear equations of two-wheeled balancing robot contain different constants such as mass of a wheel (M), mass of chassis (m), radius of a wheel (R) and etc. Each constant was measured separately. Then, observed specifications were used for mathematical modelling and simulating the system.

3.4.1. Weights

Weights of chassis and wheels were observed by SJ-HS bench scale (Figure A3.1).

3.4.2. Location of Centre of Gravity

Location of centre of gravity was observed by hanging the robot horizontally. Then, distance between the location of centre of gravity and rotational axis (motor's shafts) was measured. Figure 3.2 illustrates the way of observing the location of centre of gravity.

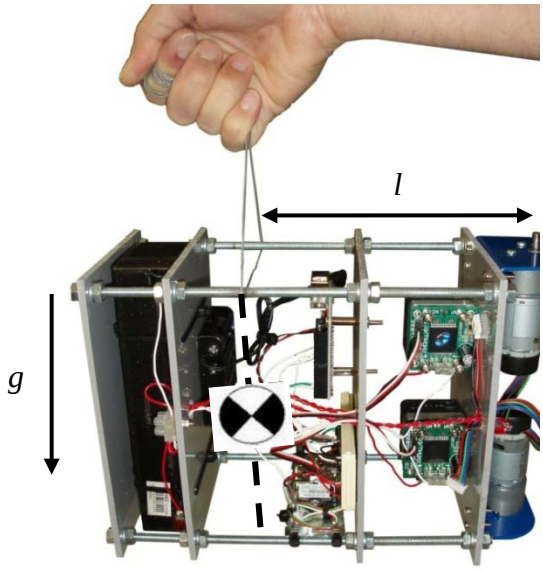


Figure 3.2 - Observing the location of centre of gravity

3.4.3. Rotation Inertia of Chassis

The chassis was assumed as a rectangular cube in order to simplify the calculation. Equation 3.32 was used to calculate the rotation inertia of chassis [16].

$$I_C = \frac{1}{12} m \left(h^2 + d^2 \right) \tag{3.32}$$

Where, *h* and *d* are height and width of the chassis respectively (Appendix 3.2).

3.4.4. Rotation Inertia of Wheel

It was assumed the robot's wheel as a cylinder. Equation 3.33 was used for calculating the rotation inertia of wheel [16].

$$I_W = \frac{1}{2} MR^2 \tag{3.33}$$

Where, *M* and *R* are mass and radius of the wheel respectively (Appendix 3.3).

Therefore, the following are physical parameters of two-wheeled balancing robot.

g = 9.81	[m/sec^2]	:	Gravity acceleration
m = 2.63	[kg]	:	Chassis weight
J_C = 0.018	[kgm^2]	:	Rotation inertia of chassis

$R = 0.05$	$[m]$	:	Wheel radius
$J_W = 1.75\times10^{-4}$	$[kgm^2]$	:	Rotation inertia of a wheel
$M = 0.14$	[kg]	:	Wheel weight
$W = 0.18$	$[m]$	:	Chassis width
$D = 0.12$	$[m]$	:	Chassis depth
$H = 0.24$	$[m]$	:	Chassis height
$l = 0.18$	$[m]$	:	Distance of the centre of mass from the wheel axle
$\mu_0 = 0.1$		:	Coefficient of friction between wheel and ground
$\mu_1 = 0$		:	Coefficient of friction between chassis and wheel

3.5. State-Space Modelling

State-space model of a dynamic system is a mathematic model that includes a set of input, output and state variables of system. The state of a dynamic system is described by a set of state variables ([$x_1(t)$, $x_2(t)$,..., $x_n(t)$]). The state variables determine the future behaviour of system when the present state of the system and input signals are known [4]. The minimum number of state variables is equal to order of system's differential equations. State-space model of a linear system contains four matrices, which are:

- *A* matrix is known as state matrix
- *B* matrix is known as input matrix
- *C* matrix is known as output matrix
- *D* matrix is known as direct transmission matrix

Equations 3.34 and 3.35 are two equations of general state-space model for a dynamic system, which are called state and output equations respectively.

$$\dot{X} = AX + Bu \tag{3.34}$$

$$Y = CX + Du \tag{3.35}$$

Figure 3.3 illustrates the related block diagram of the state-space equations.

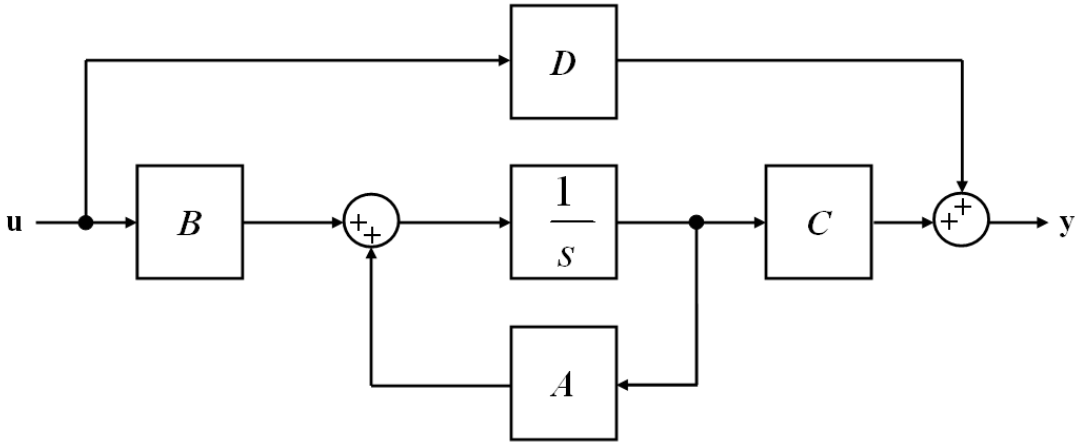


Figure 3.3 - Block diagram of state-space equations

Four state variables were chosen for dynamic system of two-wheeled balancing robot. Equation 3.36 shows the state vector of dynamic system (X). Four elements of state vectors are:

- Chassis displacement, x
- Chassis pitch angle, θ_P
- Chassis velocity, v
- Chassis angular velocity, ω_P

$$X = \begin{bmatrix} x & \theta_P & v & \omega_P \end{bmatrix}^T \tag{3.36}$$

State variables of velocity and angular velocity are derivative of displacement and pitch angle respectively. Therefore,

$$X_1 = x \tag{3.37}$$

$$X_2 = \theta_P \tag{3.38}$$

$$X_3 = v = \dot{X}_1 \tag{3.39}$$

$$X_4 = \omega_P = \dot{X}_2 \tag{3.40}$$

According to state equation (3.34), state variables were written in first derivative format. Therefore, equivalent of first derivative of each state variable were observed. Target is to find an equivalent for $\dot{X}_3$ and $\dot{X}_4$.

$$\dot{X}_1 = X_3$$

$$\dot{X}_2 = X_4$$

$$\dot{X}_3 = ?$$

$$\dot{X}_4 = ?$$

Appendix 3.4 shows all the algebraic calculations for observing $\dot{X}_3$ and $\dot{X}_4$. Therefore, $\dot{X}_3$ and $\dot{X}_4$ are:

$$\dot{X}_3 = \left[-\frac{(ml)^2 g}{den} \right] X_2 + \left[-\frac{2\mu_0 \left(ml^2 + J_C \right)}{den} \right] X_3 + \left[\frac{2ml\mu_1}{den} \right] X_4 + \left[-\frac{ml}{den} \right] (\tau_L + \tau_R)$$

$$\dot{X}_4 = \left[\frac{\left(m + 2M + 2\frac{J_W}{R^2} \right) mgl}{den} \right] X_2 + \left(\frac{2\mu_0 ml}{den} \right) X_3 + \left[-\frac{2\mu_1 \left(m + 2M + 2\frac{J_W}{R^2} \right)}{den} \right] X_4 + \left[\frac{m + 2M + 2\frac{J_W}{R^2}}{den} \right] (\tau_L + \tau_R)$$

Where,

$$den = \left(m + 2M + 2\frac{J_W}{R^2} \right)\left(ml^2 + J_C\right) - (ml)^2$$

and τ_L and τ_R are required torques for left and right wheels. Equation 3.41 shows the expanded version of state equation.

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \\ \dot{X}_4 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \times \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix} \times u \tag{3.41}$$

Each elements of *A* and *B* matrices were observed after developing the derivative form of state variables.

$$\dot{X}_1 = A_{11}X_1 + A_{12}X_2 + A_{13}X_3 + A_{14}X_4$$

$$\dot{X}_2 = A_{21}X_1 + A_{22}X_2 + A_{23}X_3 + A_{24}X_4$$

$$\dot{X}_3 = A_{31}X_1 + A_{32}X_2 + A_{33}X_3 + A_{34}X_4 + B_3\left(\tau_L + \tau_R\right)$$

$$\dot{X}_4 = A_{41}X_1 + A_{42}X_2 + A_{43}X_3 + A_{44}X_4 + B_4\left(\tau_L + \tau_R\right)$$

Therefore, *A*, *B*, *C* and *D* matrices are shown below.

$$A=\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\dfrac{(ml)^2 g}{den} & -\dfrac{2\mu_0\left(ml^2+J_C\right)}{den} & \dfrac{2ml\mu_1}{den} \\ 0 & \dfrac{\left(m+2M+2\dfrac{J_W}{R^2}\right)mgl}{den} & \dfrac{2\mu_0 ml}{den} & -\dfrac{2\mu_1\left(m+2M+2\dfrac{J_W}{R^2}\right)}{den} \end{bmatrix} \tag{3.42}$$

$$B=\begin{bmatrix} 0 \\ 0 \\ -\dfrac{ml}{den} \\ \dfrac{m+2M+2\dfrac{J_W}{R^2}}{den} \end{bmatrix} \tag{3.43}$$

$$C=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.44}$$

$$D=\begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.45}$$

Numeric state-space model of two-wheeled balancing robot was observed by substituting the constant values (shown in table 3.1) into equations 3.42 and 3.43. Equations 3.46 and 3.47 show the state and output equations of state-space model.

$$\dot{X}=\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -24.2421 & -0.2276 & 0 \\ 0 & 156.1861 & 1.044 & 0 \end{bmatrix}\times\begin{bmatrix} x \\ \theta_P \\ v \\ \omega_P \end{bmatrix}+\begin{bmatrix} 0 \\ 0 \\ -5.22 \\ 33.63 \end{bmatrix}u \tag{3.46}$$

$$Y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x \\ \theta_P \\ v \\ \omega_P \end{bmatrix} \tag{3.47}$$

State-space matrices (*A, B, C* and *D*) were used to observe the further analysis and simulation with MATLAB and SIMULINK. Figure 3.4 illustrates the designed block diagram of state-space model.

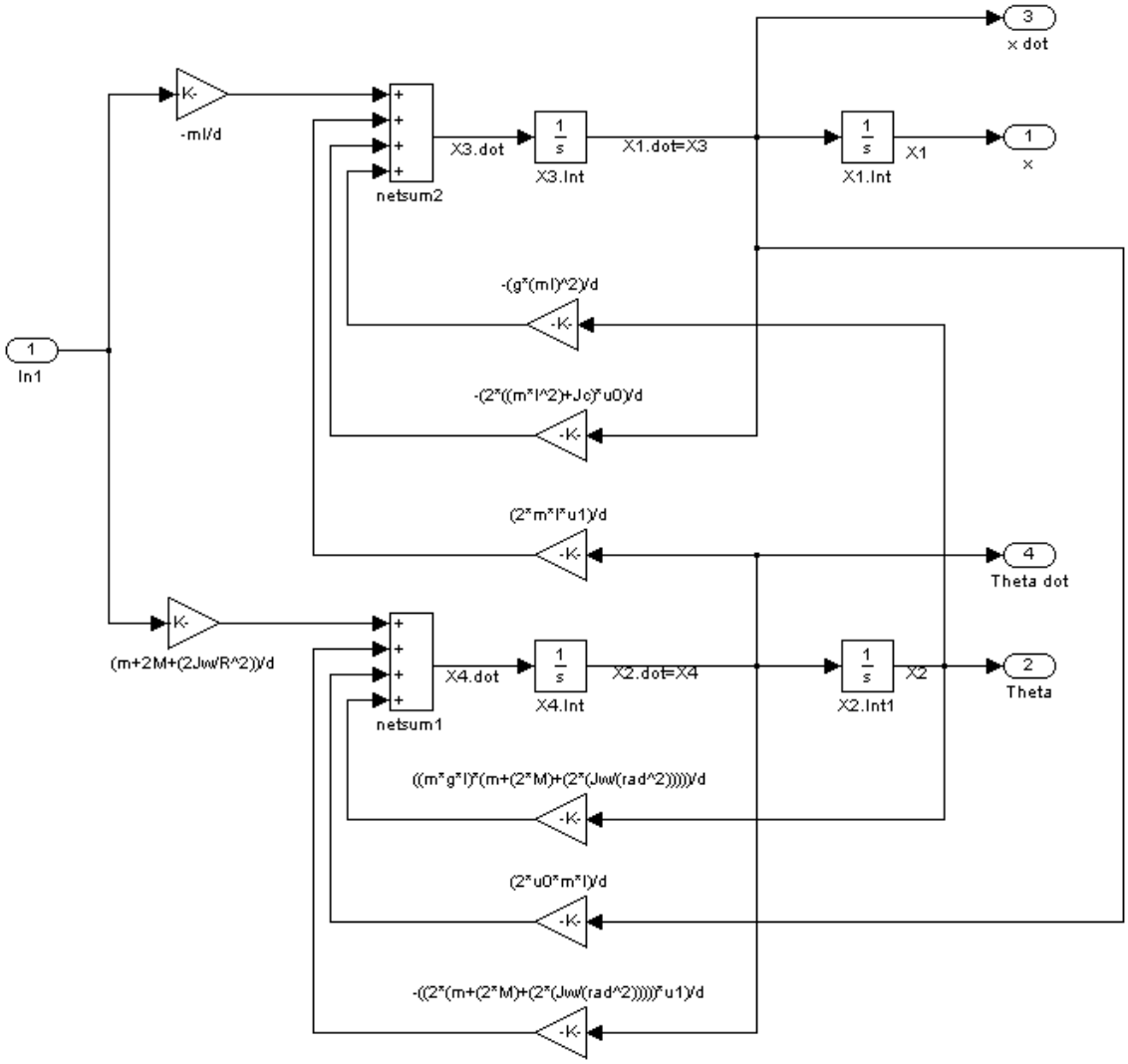


Figure 3.4 - State-space model block diagram

3.6. Conclusion

In this chapter differential equations of system were observed and existing constants of equations were measured. State-space model of system were designed for further simulation and analyses.

Chapter 4

Controller Design

(MATLAB & SIMULINK Implementations)

MASSEY UNIVERSITY

School of Engineering & Advanced Technology

4.1. Introduction

MATLAB and SIMULINK were used in this chapter to observe the system responses and its performance. Designing a full-state feedback controller was required to stabilise the unstable balancing robot. Set point or reference point of the system is when robot stands upright.

4.2. Open-Loop Control System

Those systems in which the output has no effect on the control action are called open-loop control systems [17]. Figure 4.1 illustrates the block diagram of open-loop system.

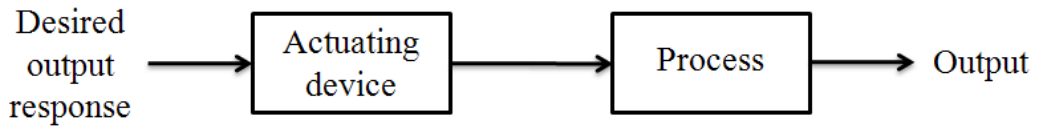


Figure 4.1 - Block diagram of open-loop system

Transfer function of a system is a mathematical relation between input and output of a linear system. This function can be observed by taking the Laplace transforms of differential equations of system. In this project MATLAB was used to generate the transfer functions of two-wheeled balancing robot (Appendix 4.1).

There are two proofs that show the two-wheeled balancing robot is an unstable dynamic system. These proofs are:

4.2.1. Open-Loop Impulse Response

Figure 4.2 illustrates the open-loop impulse response of displacement (x), pitch angle (θ_P), velocity (v) and angular velocity (ω_P) of balancing robot.

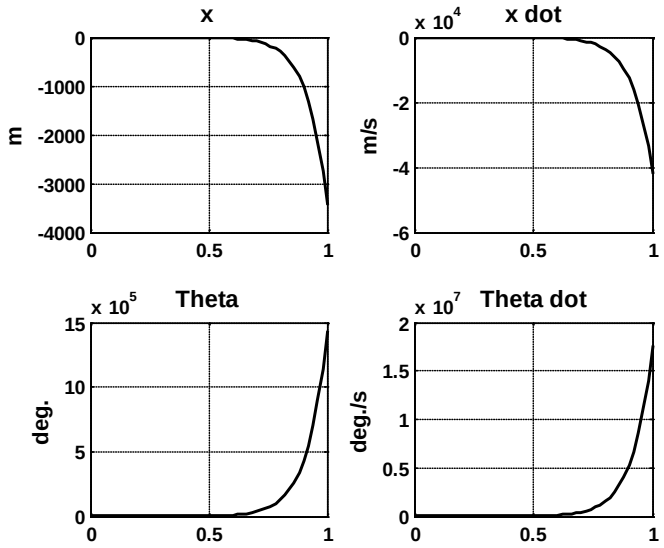


Figure 4.2 - Open-loop impulse response

Open-loop impulse response of balancing robot was proved that open-loop system is an unstable system. Figure 4.2 shows that there is no settling point for impulse responses.

4.2.2. Open-Loop System Poles

The system poles were found by observing the eigenvalues of matrix *A*, defined as the solutions λ in equation 4.1[21].

$$\left|A-\lambda I\right|=0 \tag{4.1}$$

Where, *I* is the identity matrix and the symbol '| |' refers to the matrix determinant. Appendix 4.2 shows the algebraic calculations for observing the eigenvalues of matrix *A*. Equation 4.2 illustrates the eigenvalues of matrix *A*, which are open-loop system poles.

$$Poles = \begin{bmatrix} 0 & -12.5792 & 12.4171 & -0.0656 \end{bmatrix} \tag{4.2}$$

As equation 4.2 shows, there is one positive pole located on right-hand plane of pole-zero plot and it means open-loop system is unstable. Figure 4.3 illustrates the location of open-loop system poles.

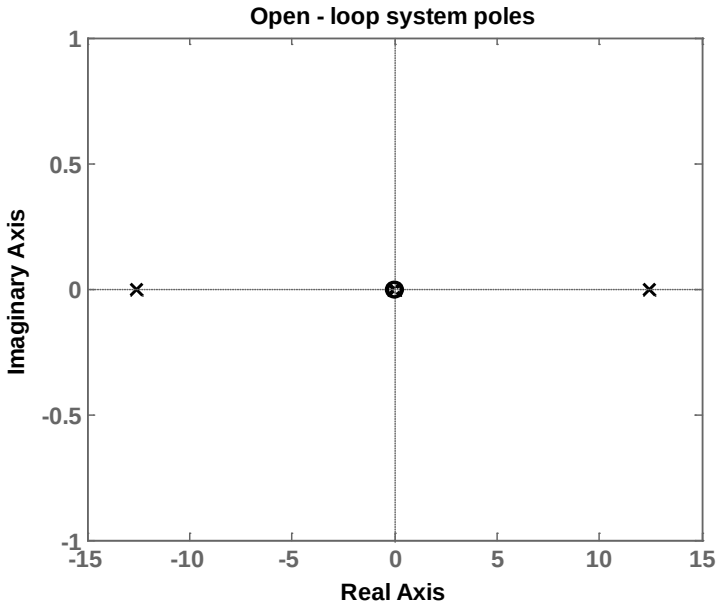


Figure 4.3 - Open-loop system poles

Therefore, full-state feedback control was designed to stabilise the unstable dynamic system.

4.3. Controllability and Observability

Full-state feedback design commonly relies on pole-placement techniques [4]. It was necessary to note that the dynamic system is completely controllable and observable. The observed state-space matrices in section 3 were used to check the controllability and observability of system.

- A system is completely controllable if C_M matrix is of rank *n*. Where *n* is number of state variables.

$$C_M = \left[B \mid AB \mid A^2B \mid A^3B \right] \tag{4.3}$$

Appendix 4.3 illustrates more details of generating the C_M matrix. Rank of matrix C_M was observed and it is equal to number of state variables. Therefore, dynamic system of balancing robot is completely controllable.

$$Rank\left(C_M\right) = 4$$

- A system is completely observable if O_M matrix is of rank *n*. Where *n* is number of state variables.

$$O_M = \left[C^T \mid A^TC^T \mid \left(A^T\right)^2C^T \mid \left(A^T\right)^3C^T \right] \tag{4.4}$$

Appendix 4.4 illustrates more details of generating the O_M matrix. Rank of matrix O_M was observed and it is equal to number of state variables. Therefore, dynamic system of balancing robot is completely observable.

$$Rank\left(O_M\right) = 4$$

4.4. Closed-loop Control System

Figure 4.4 illustrates general block diagram of closed-loop system. Tilt sensor and gyroscope measurements were used as controller feedbacks.

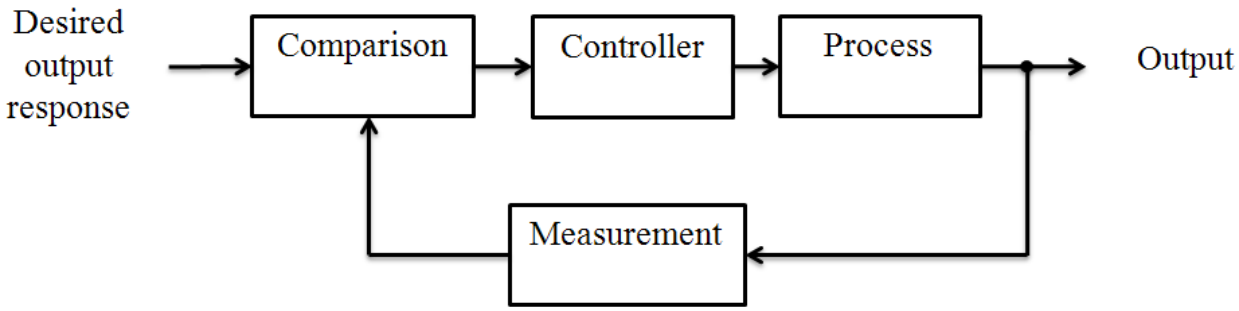


Figure 4.4 - Block diagram of closed-loop system

A full-state feedback controller was designed to locate all the closed-loop system poles on left-hand plane of pole-zero plot. Figure 4.5 illustrates the block diagram of full-state feedback control.

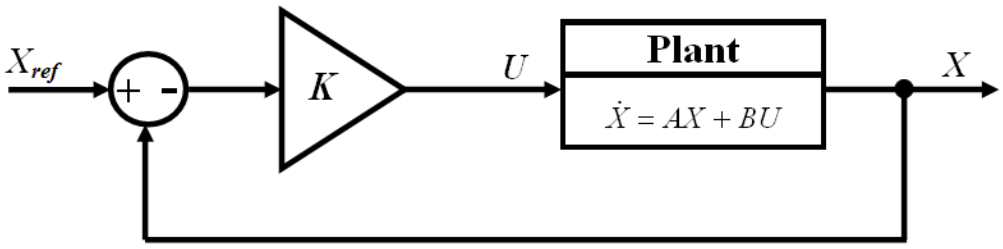


Figure 4.5 - Block diagram of full-state feedback control

Where, X_{ref} is reference of system and K matrix contains negative feedback gains that were used to make the input of system proportional to the given state.

$$U = -KX = -\left(k_1 x_1 + k_2 x_2 + k_3 x_3 + k_4 x_4 \right)$$

Equation 4.5 describes the full-state feedback control shown on figure 4.5.

$$u(t) = -K\left(x(t) - x_{ref}\right) \tag{4.5}$$

State matrix of closed-loop control system were defined as $A_c = A - BK$. Therefore state equation of balancing robot system was described as equation 4.6.

$$\dot{x}(t) = \left(A - BK\right)\ x(t) + BK\ x_{ref} \tag{4.6}$$

Linear-quadratic regulator (LQR) method was used to generate the K matrix. Appendix 4.1 shows required MATLAB commands to generate the K matrix. Equation 4.7 illustrates the closed-loop system gains in K matrix.

$$K = \begin{bmatrix} -0.006 & 82.2429 & 75.6809 & 13.9637 \end{bmatrix} \tag{4.7}$$

Equation 4.8 shows the eigenvalues of matrix A_c, which are the controlled closed-loop system poles.

$$Poles = \begin{bmatrix} -0.0649 & -37.3627 + 35.2102i & +37.3627 + 35.2102i & 0 \end{bmatrix} \quad (4.8)$$

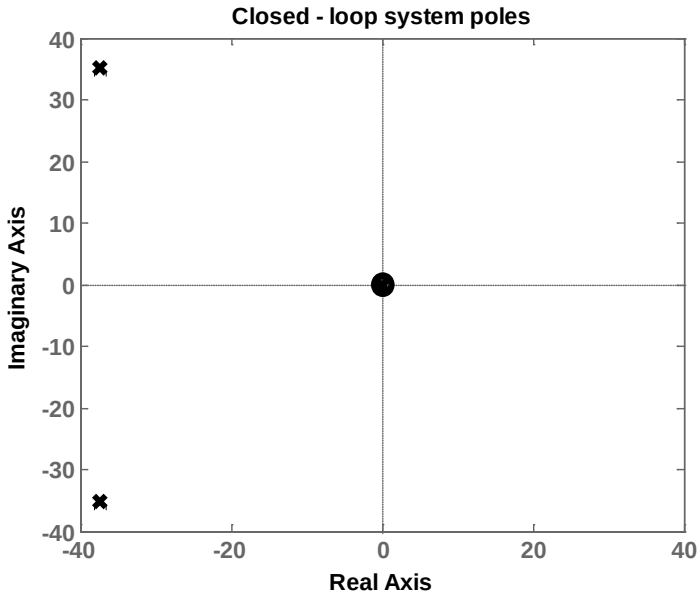


Figure 4.6 - Closed-loop system poles

Figure 4.7 and 4.8 illustrates the closed-loop impulse response of displacement and pitch angle of system respectively.

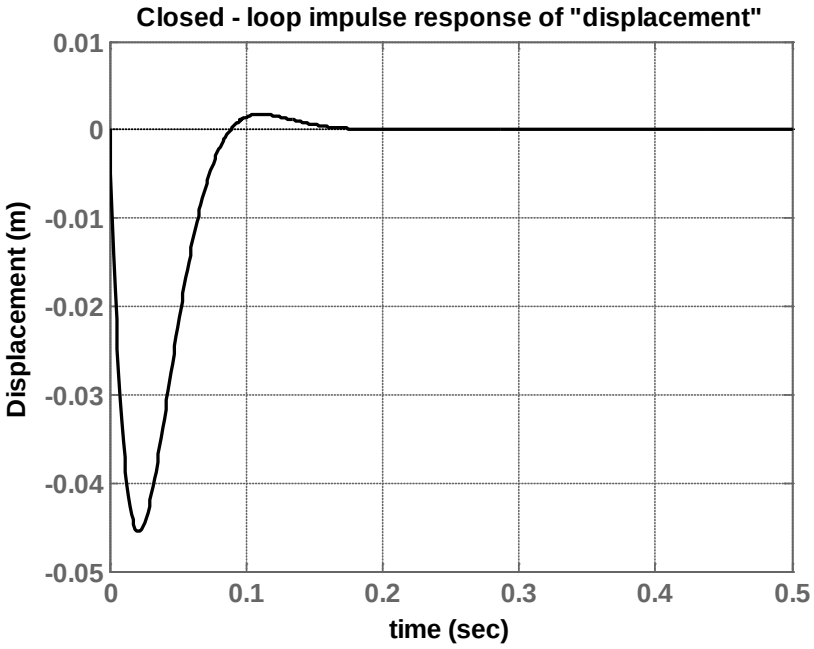


Figure 4.7 - Impulse response of robot displacement

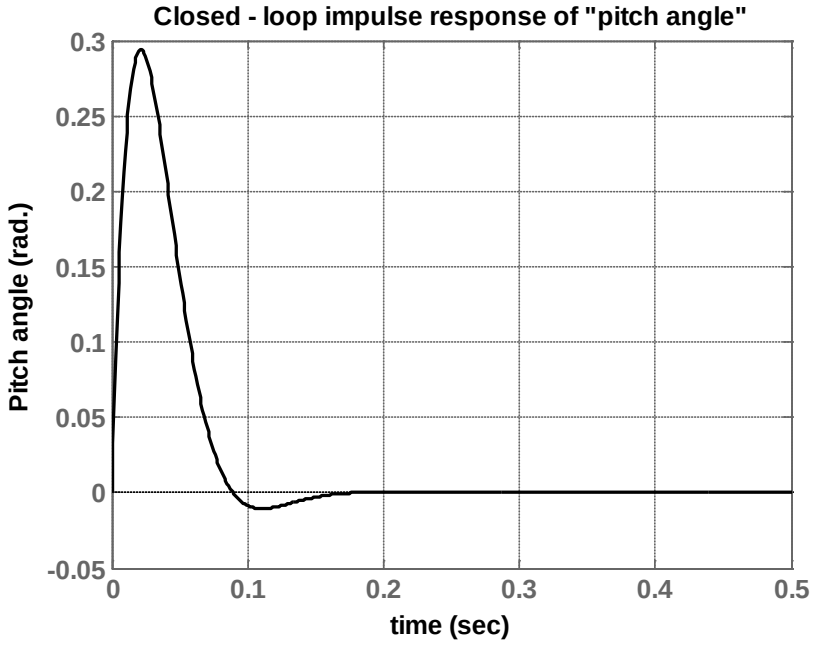


Figure 4.8 – Impulse response of robot pitch angle

A closed-loop SIMULINK model was designed for dynamic system of two-wheeled balancing robot. In this model, initial pitch angle of robot was set at 4.5 degree and behaviour of robot was observed. Following graphs show robot moved forward and balanced itself after 0.2 seconds. Figure 4.9 illustrates the closed-loop SIMULINK model and following plots show the robot impulse responses respectively.

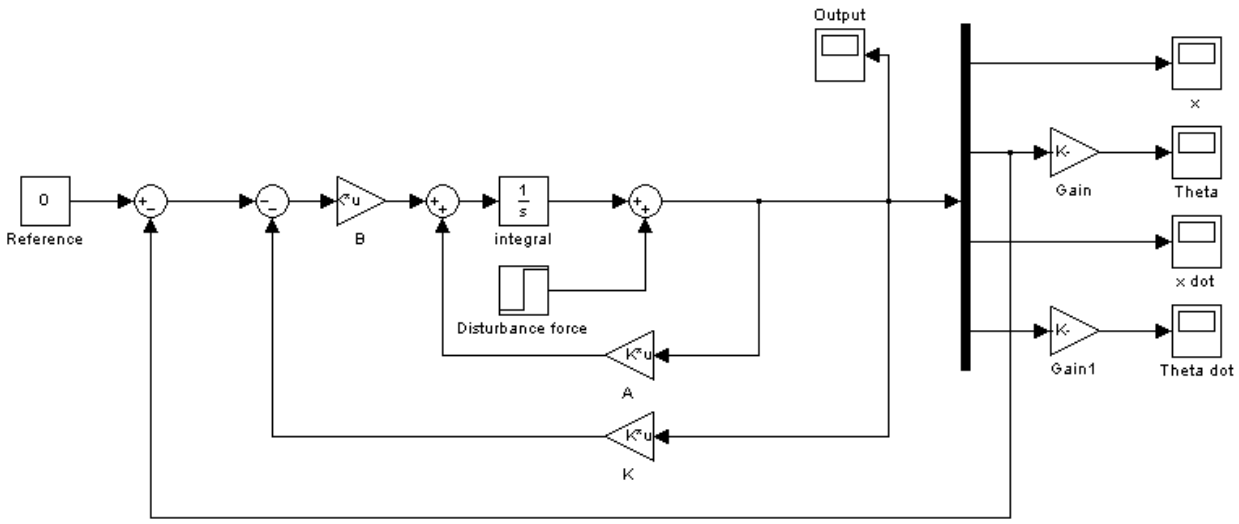


Figure 4.9 - Closed-loop SIMULINK model of balancing robot

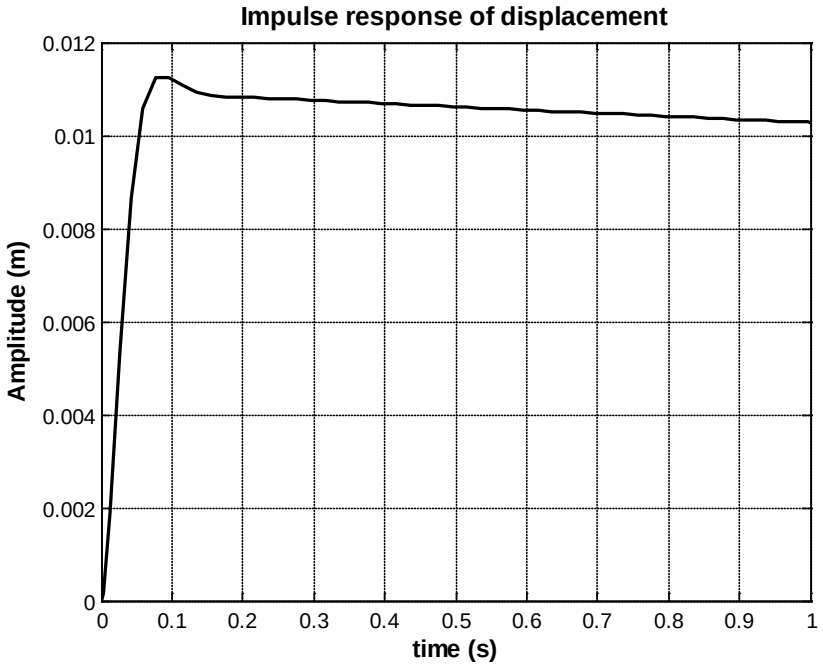


Figure 4.10 - Closed-loop impulse response of robot displacement

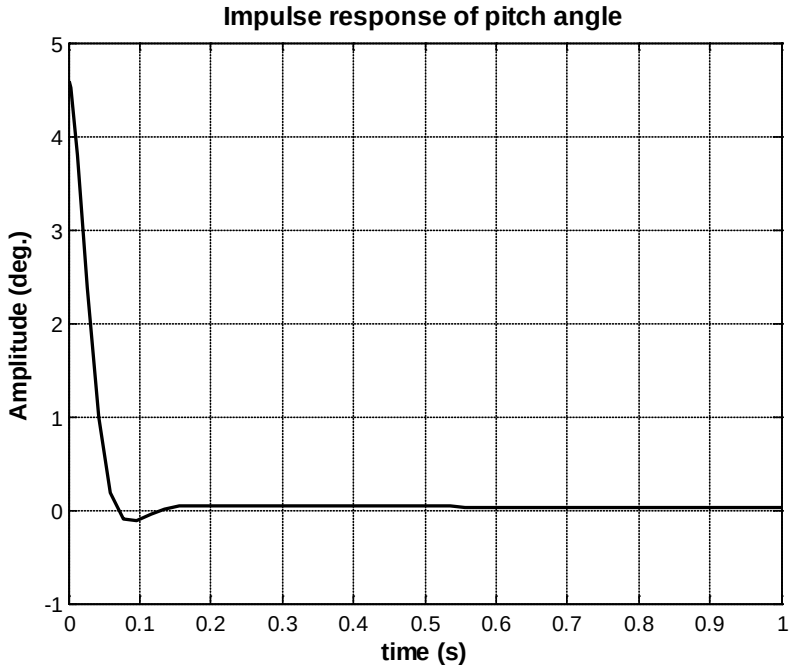


Figure 4.11 - Closed-loop impulse response of robot pitch angle

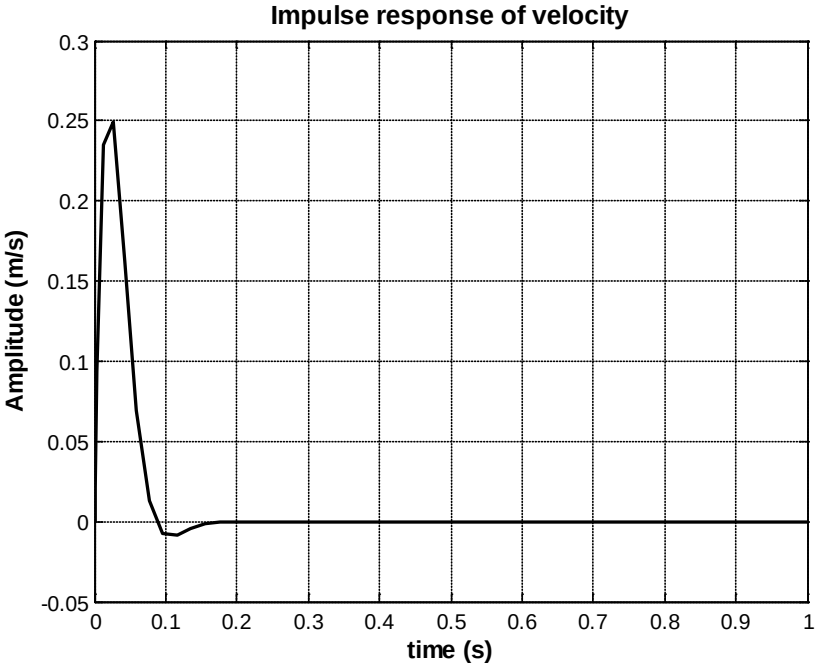


Figure 4.12 - Closed-loop impulse response of robot velocity

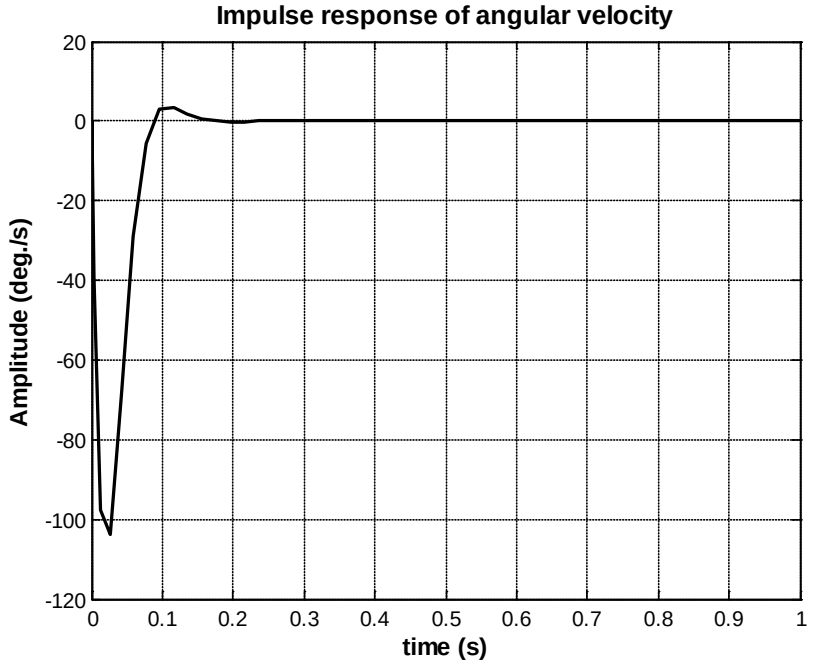


Figure 4.13 - Closed-loop impulse response of robot angular velocity

Table 4.1 illustrates the specifications of pitch angle closed-loop impulse response.

Table 4.1 - Specifications of closed-loop impulse response

	Closed-loop impulse response of Pitch angle (θ_P)
Rise time (t_r)	0.05 s
Delay time (t_d)	0.03 s
Peak time (t_p)	0.09 s
Settling time (t_s)	0.15 s

4.5. Conclusion

Full-state feedback control was designed successfully to control the unstable two-wheeled balancing robot. Tilt sensor and gyroscope provide the closed-loop system feedback. Reference point was compared with generated feedback and microcontroller was sent the right command to motor controllers.

Chapter 5

Parametric Studies

MASSEY UNIVERSITY
School of Engineering & Advanced Technology

5.1. Introduction

Two-wheeled balancing robot has some key parameters that are directly engaged with system's performance and responses. The robot's movement was optimised by choosing the suitable parameters and specifications for physical model. These parameters are:

1. Coefficient of friction (CoF) between wheels and ground (μ_0).
2. Location of centre of gravity (l).
3. Radius of the wheel (R).

In this chapter parametric studies were done and system responses were observed by variation of key parameters based on designed closed-loop SIMULINK model in chapter 4 (Figure 4.8).

5.2. Variation of μ_0

Variation of μ_0 affects on some of controller gains and closed-loop impulse response of displacement. Figure 5.1 illustrates the changes of K_1, K_2, K_3 and K_4 that are the controller gains of x, θ_P, v and ω_P respectively.

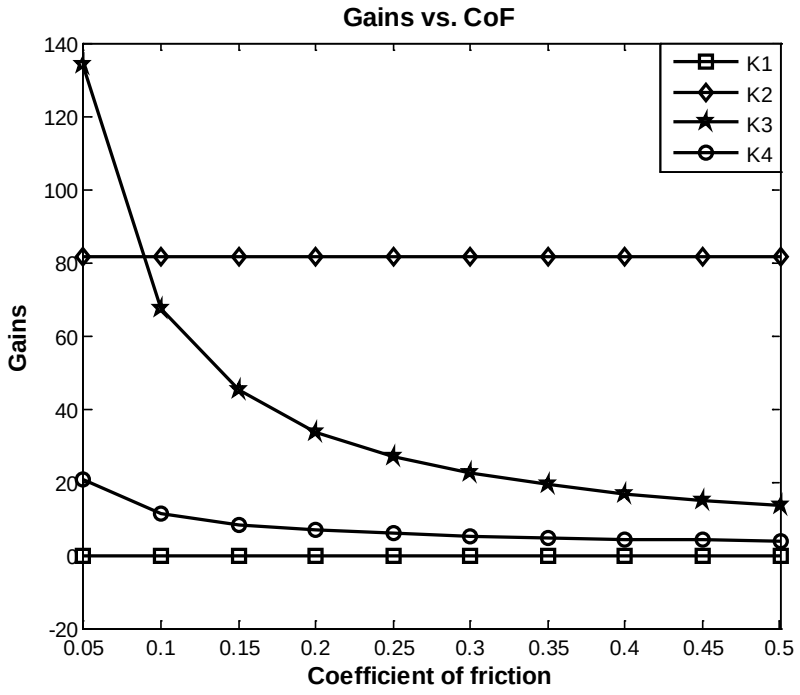


Figure 5.1 - Controller gains vs. coefficient of friction (μ_0)

Figure 5.1 illustrates that variation of μ_0 has no affect on controller gains of displacement (K_1) and pitch angle (K_2). As figure 5.1 shows, controller gain of velocity (K_3) is more sensitive to variation of μ_0. Therefore robot velocity is the most important factor that was controlled precisely for balancing the robot.

Variation of μ_0 affects on impulse response of robot displacement too. Figure 5.2 illustrates that displacement overshoot decreases slightly by increasing μ_0.

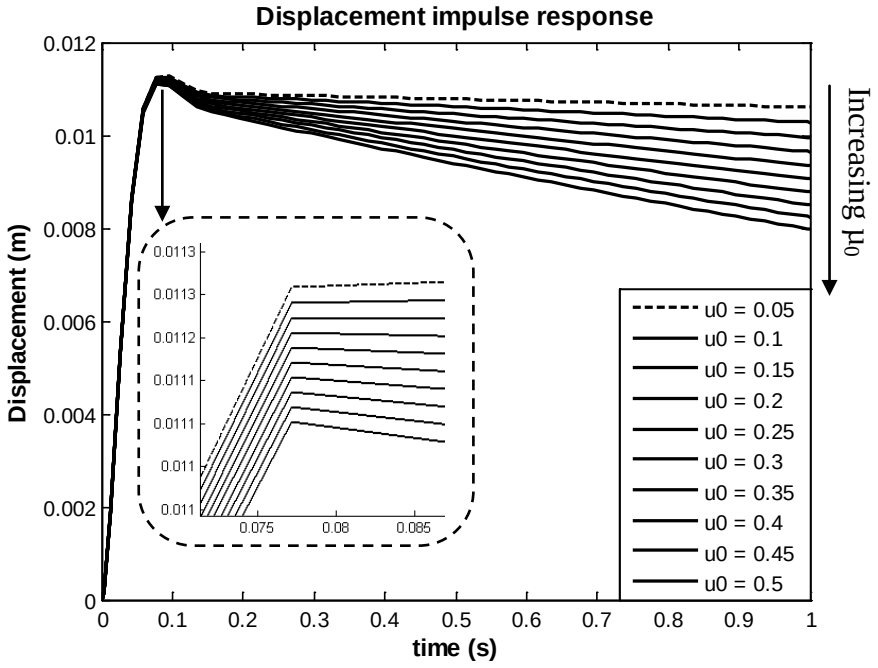


Figure 5.2 - Displacement impulse response vs. coefficient of friction (μ_0)

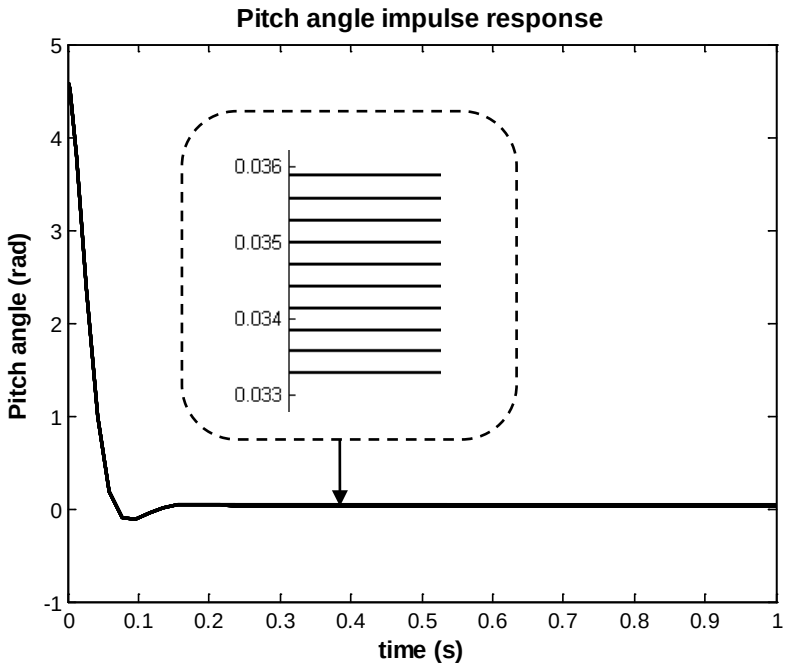


Figure 5.3 - Pitch angle impulse response vs. coefficient of friction

Figure 5.3 illustrates that variation of μ_0 has minor affect on impulse response of robot's pitch angle.

5.3. Variation of *l*

Location of centre of gravity is another important factor that affects the performance of two-wheeled balancing robot. Figure 5.3 illustrates the sensitivity of impulse response overshoot of displacement with variation of *l*.

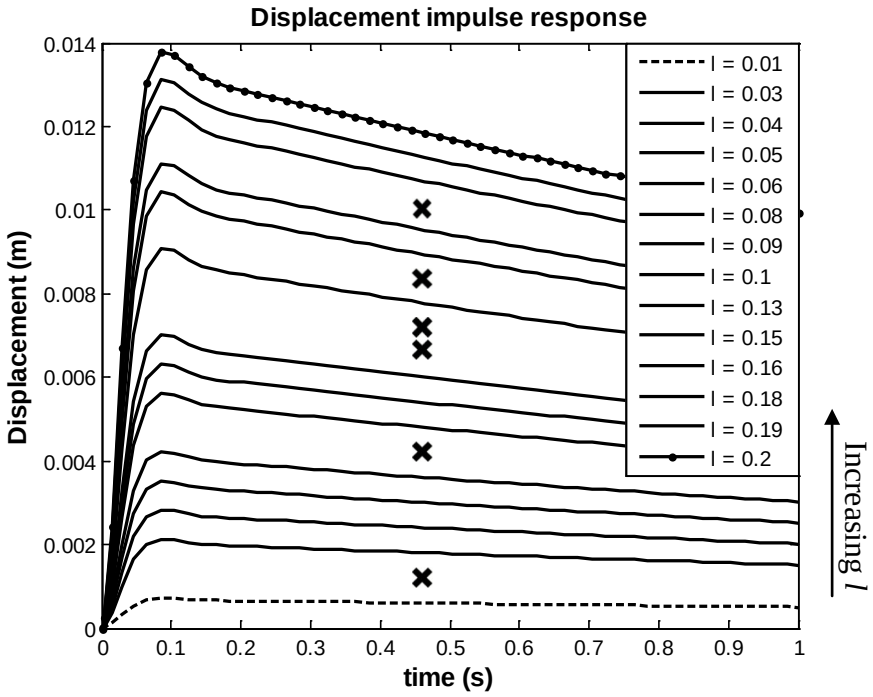


Figure 5.4 - Displacement impulse response vs. *l*

Figure 5.4 illustrates that displacement overshoot increases with increasing the location of centre of gravity. Also this fact was observed by several experiments that robot was tended to fall faster when centre of gravity located at higher point of robot because of existence the higher rotation inertia. There are some singular points for *l* that are shown by crosses on figure 5.4. These points were not chosen as location of centre of gravity of physical model. Figure 5.5 illustrates that variation of *l* has a minor affect on impulse response of robot's pitch angle.

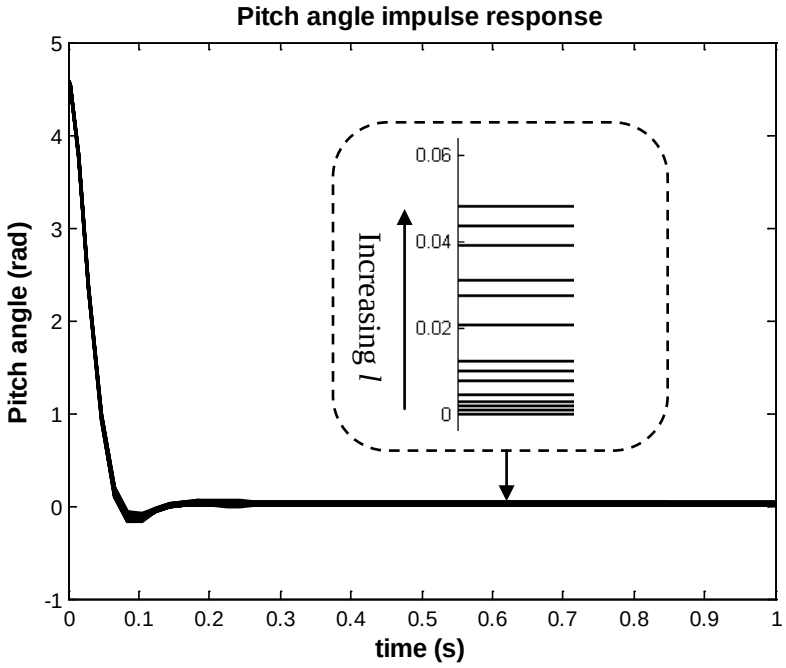


Figure 5.5 - Pitch angle impulse response vs. *l*

5.4. Variation of *R*

Size of the wheel is another important factor that affects the performance of balancing robot. Bigger wheels increase the stability of robot, because it takes shorter time to travel from point A to point B. Figure 5.6 illustrates that displacement overshoot increases by increasing the radius of wheel.

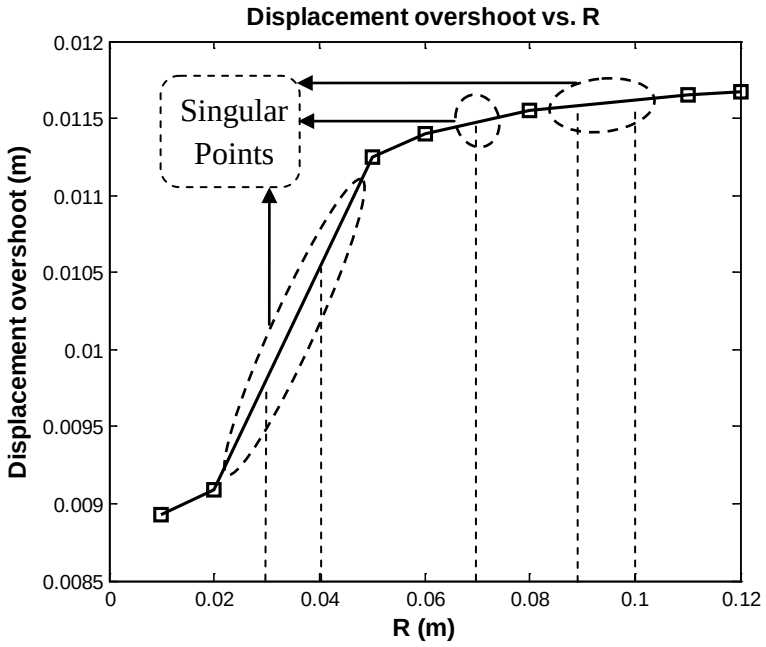


Figure 5.6 - Displacement overshoot vs. *R*

There were some singular points in simulation that were not chosen as size of robot's wheel in physical model.

5.5. Conclusion

According to observed results in this chapter, ideal constants were chosen for location of centre of gravity (l) and radius of robot's wheel (R) to improve the robot's performance. The observed plots in this chapter show that variations of key parameters were caused small changes on robot's performance. But several experiments were proved that deploying the observed constants on physical model had a visible improvement in robot's motion.

Chapter 6

Results and Implementation

MASSEY UNIVERSITY
School of Engineering & Advanced Technology

6.1. Introduction

Observed results from parametric studies were applied into physical model to improve the robot performance. Figure 6.1 illustrates the final physical model of balancing robot.

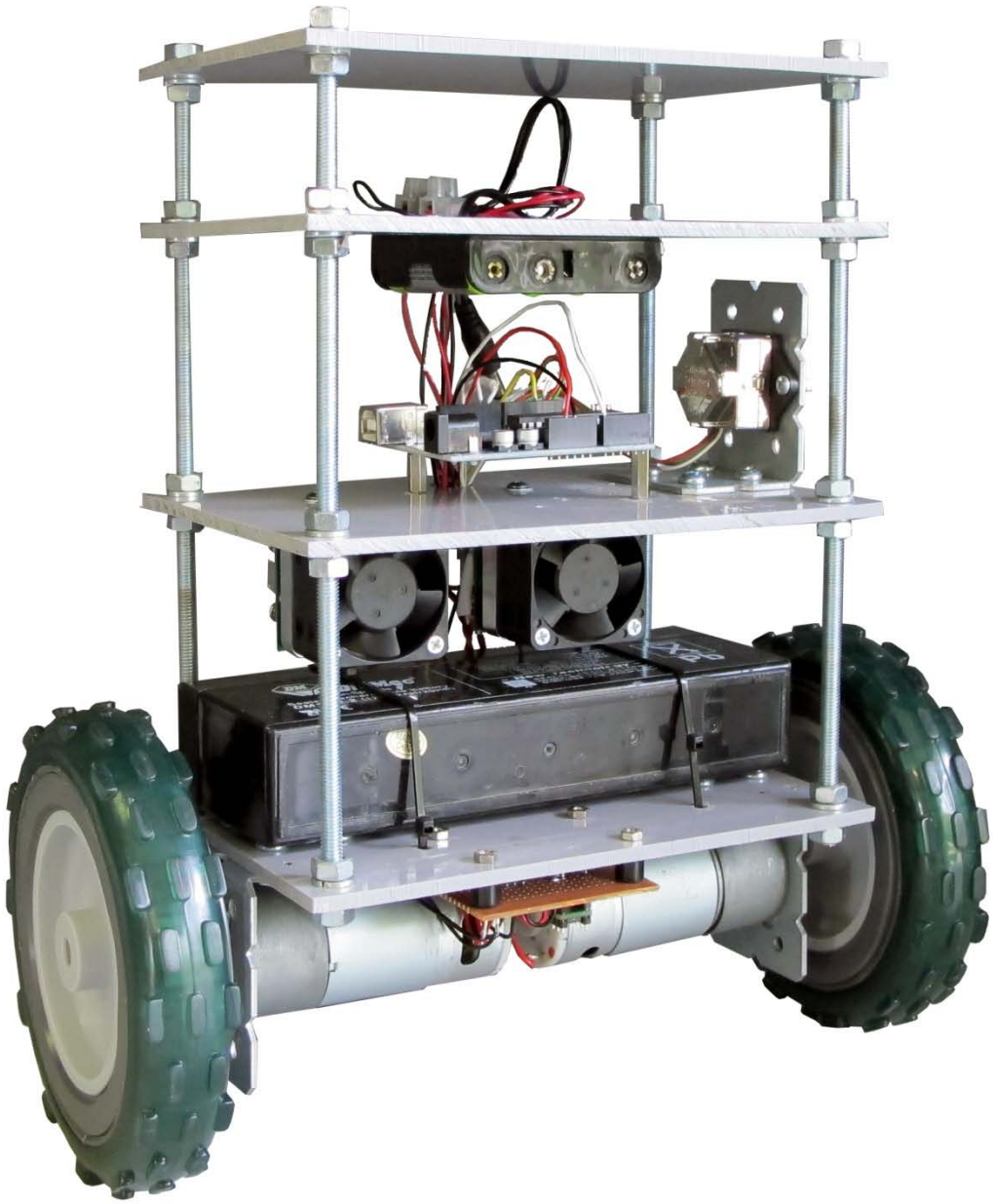


Figure 6.1 - Final design of two-wheeled balancing robot

Few changes were done in final design of balancing robot to improve the performance of robot. These changes are:

- Location of centre of gravity (l) was pulled down and located on lower level of chassis. This change provided less rotation inertia and increased the stability of robot.
- Size of the wheels (R) was increased.
- Tilt sensor was located parallel to rotation axis of wheels. Therefore sensor performs more accurate and detects small changes of pitch angle.
- Initial DC motors were replaced with higher torque DC motor to increase the stability of robot.

Therefore, new physical parameters of two-wheeled balancing robot after few changes in physical model are:

$m = 3.18$	$[kg]$	:	Chassis Weight
$J_C = 0.022$	$[kgm^2]$	:	Rotation inertia of chassis
$R = 0.06$	$[m]$	:	Wheel radius
$J_W = 5.44\times10^{-4}$	$[kgm^2]$	:	Rotation inertia of a wheel
$M = 0.25$	[kg]	:	Wheel weight
$l = 0.183$	$[m]$	:	Distance of the centre of mass from the wheel axle

6.2. Optimised Closed-loop Impulse Responses

Figure 6.2 and 6.3 illustrates the optimised closed-loop impulse response of displacement and pitch angle of two-wheeled balancing robot. The overshoot of optimised closed-loop impulse responses were decreased compare with initial impulse responses (figures 4.7 and 4.8).

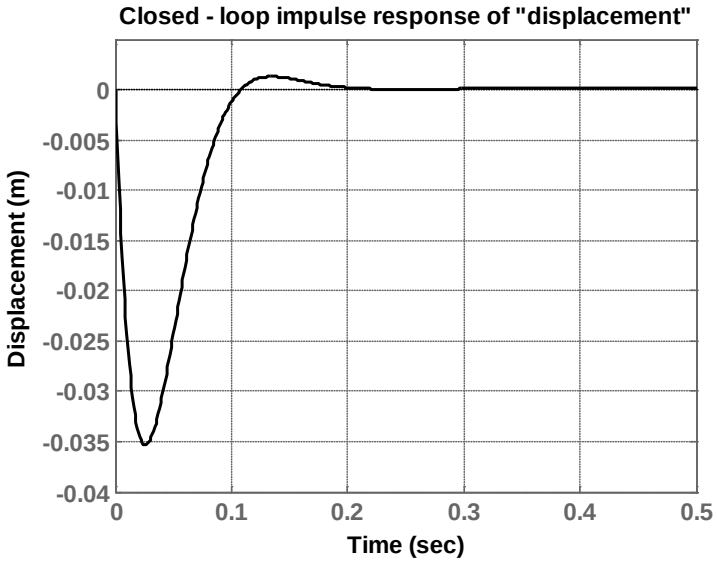


Figure 6.2 - Displacement impulse response

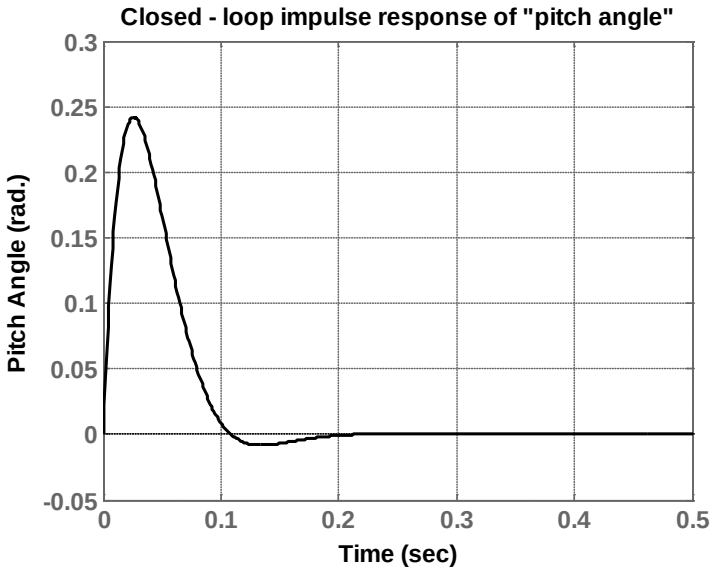


Figure 6.3 – Pitch angle impulse response

6.3. Kalman Filter

Kalman filter was implemented for gyroscope rate and raw tilt angle fusion. The Kalman filter is the minimum variance state estimator for linear dynamic systems with Gaussian noise [20]. Kalman filter is an iterative filter that requires two inputs or states. In this project tilt sensor and gyroscope provide two inputs of Kalman filter. A linear model is required for Kalman filter. Equation 6.1 and 6.2 illustrate the general form of a linear model using the gyroscope data respectively. Including the time (*dt*) between two measurements are essential.

$$x_{k+1} = A.x_k + B.u_k \tag{6.1}$$

$$\begin{bmatrix} \theta_P \\ bias \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_P \\ bias \end{bmatrix}_k + \begin{bmatrix} dt \\ 0 \end{bmatrix} u_k \tag{6.2}$$

The output of tilt sensor and gyroscope were fused together by Kalman filter. Following lines show the simple pseudo code of Kalman filter [9].

- Read the value of the last measurement
- Update the state *X* of model every *dt* with a biased gyro measurement
- Read the angle from tilt sensor
- Calculate the difference between the measured value and predicted value
- Calculate the covariance matrix
- Calculate the Kalman gain
- Update the covariance matrix
- Calculate the error

Appendix 5 illustrates the implemented Kalman codes. For observing the performance of developed Kalman filter's algorithm, robot was tilted back and forth randomly and figure 6.4 was generated and this figure shows that Kalman filter was implemented correctly and estimated angle followed the raw angle smoothly. Figure 6.5 illustrates the Kalman filter's output while robot was balanced.

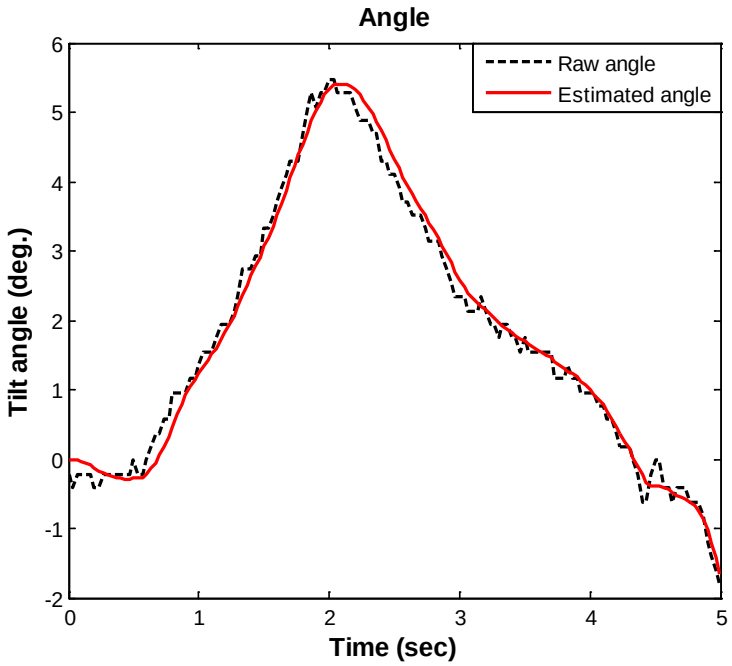


Figure 6.4 - Estimated angle by Kalman filter

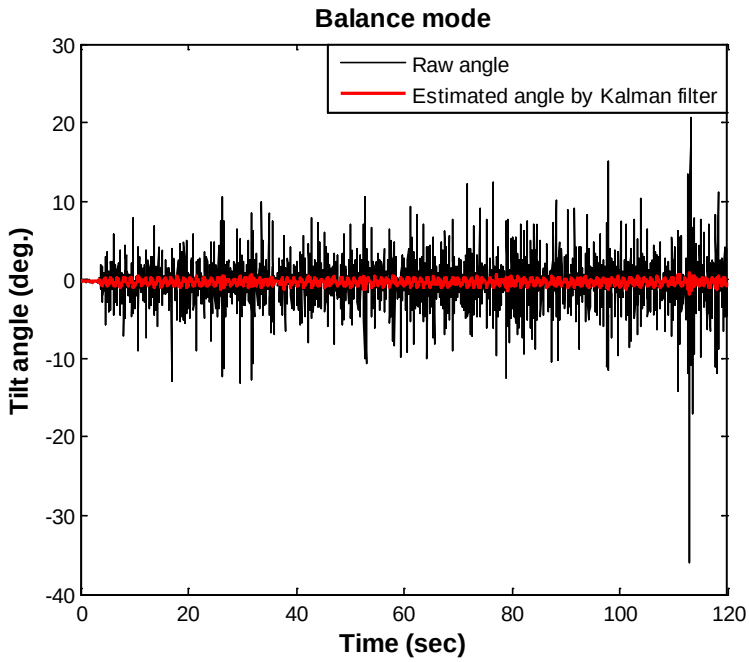


Figure 6.5 - Kalman filter's output in balance mode

Table 6.1 shows the sample of numeric values of raw angle and estimated angle shown on figure 6.5, when robot was balance for 2 seconds.

Table 6.1 - Raw and estimated angle

Raw angle	Estimated angle
-2.2	-0.73
-2.8	-0.84
-0.22	-0.75
-0.22	-0.64
-1.21	-0.51
-1.21	-0.35
-2.4	-0.26
-3.99	-0.35
-2.6	-0.44
-1.41	-0.57
1.16	-0.52
-0.22	-0.53
-1.21	-0.52
0.17	-0.38
-1.81	-0.33
-0.81	-0.24
-0.02	-0.1
0.37	0.05
0.57	0.25
-3.79	0.15
-1.21	0.06
2.34	0.08
0.37	-0.13
2.34	-0.13
1.55	-0.12
1.55	-0.04
0.37	0
-0.02	0.05
-0.62	0.01
-1.21	-0.14
1.55	-0.18
1.36	-0.22
-0.62	-0.36
-0.42	-0.46

-1.01	-0.62
-0.81	-0.8
5.86	-0.47
-2.6	-0.59
-4.39	-0.78
-0.22	-0.71
-0.22	-0.65
-2.4	-0.69
-0.62	-0.57
-1.01	-0.5
-2.2	-0.49
-1.61	-0.47
-2.8	-0.58
1.95	-0.4
-2.2	-0.48
-0.62	-0.44
-0.02	-0.35
-0.02	-0.24
-0.22	-0.15
0.96	0.05
-0.02	0.2
-1.21	0.26
-3.2	0.08
1.36	0.08
1.95	0.02
0.57	-0.1
1.75	-0.1

6.4. PID Controller

A proportional-integral-derivative (PID) controller was designed to generate the required input for motor controllers to control the rotation of wheels based on the Kalman filter's output. Figure 6.6 illustrates the general block diagram of PID controller [17].

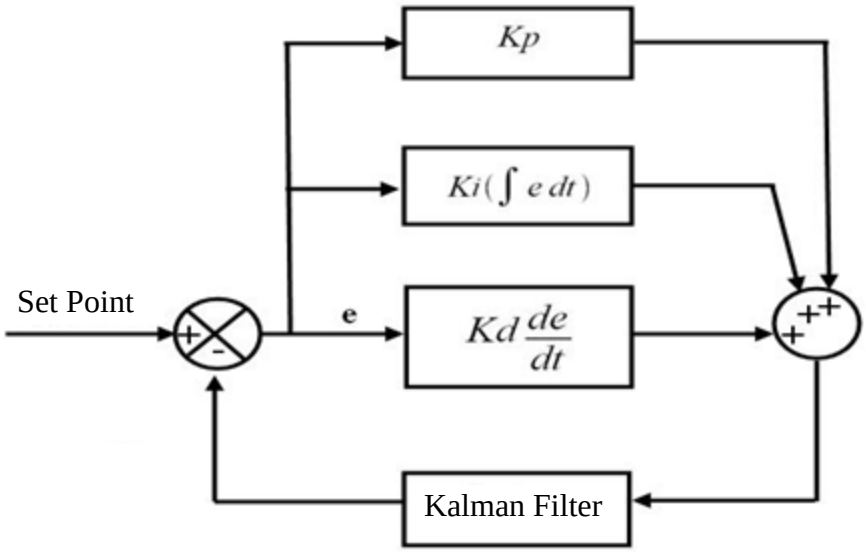


Figure 6.6 - PID controller's block diagram

A PID controller consists of three types of controllers, which are proportional, integral and derivative controller. Each of these controllers generates the corresponding P, I and D values. Basically error (*e*) in PID controller is the difference between set point and the measured value. There are three types of error, which are used to generate the PID values:

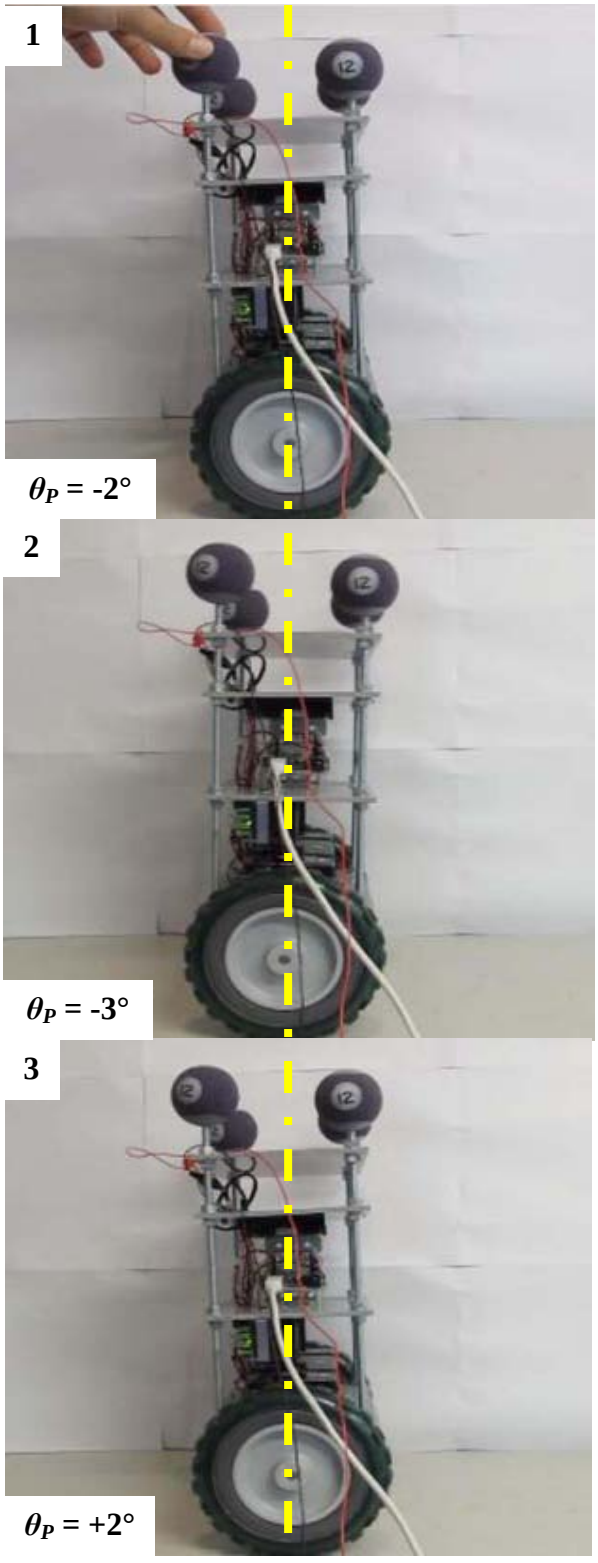
1. Current error = set point - Kalman output
   $P = K_p$ x current error
2. Accumulator error = accumulator error - current error
   $I = K_i$ x accumulator error

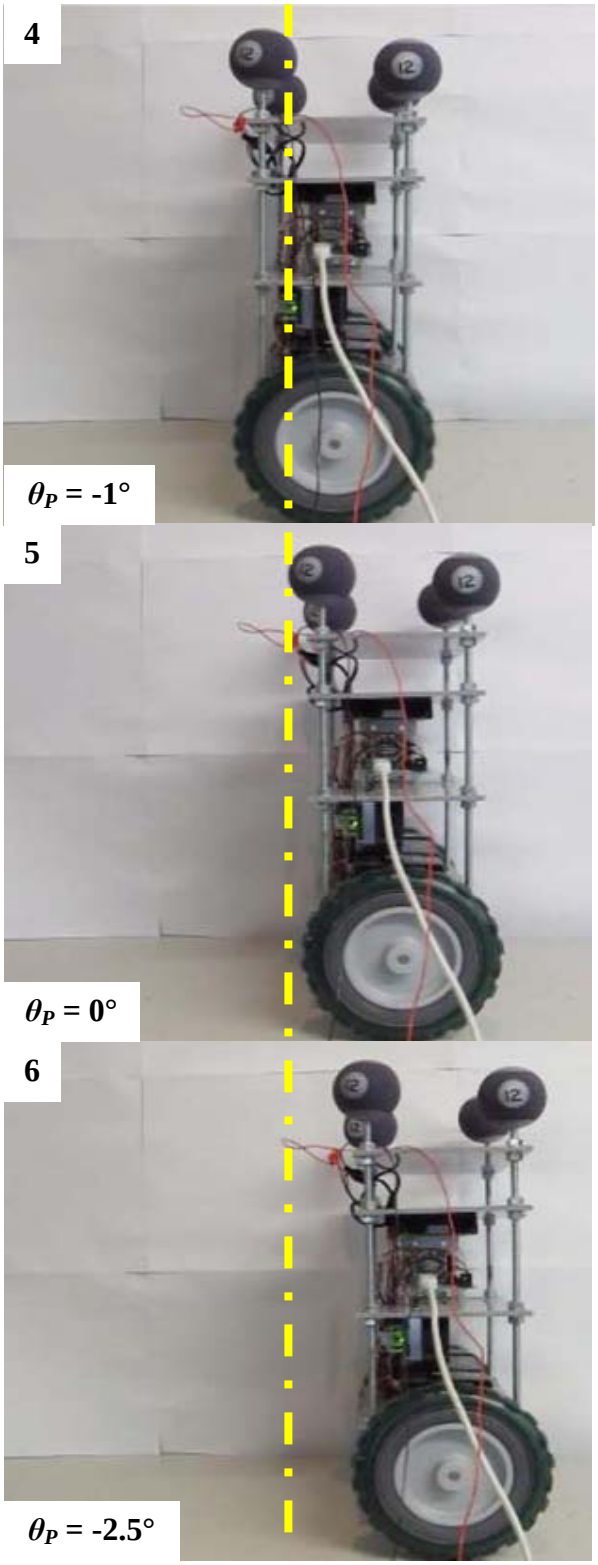
3. Delta error = current error – previous error

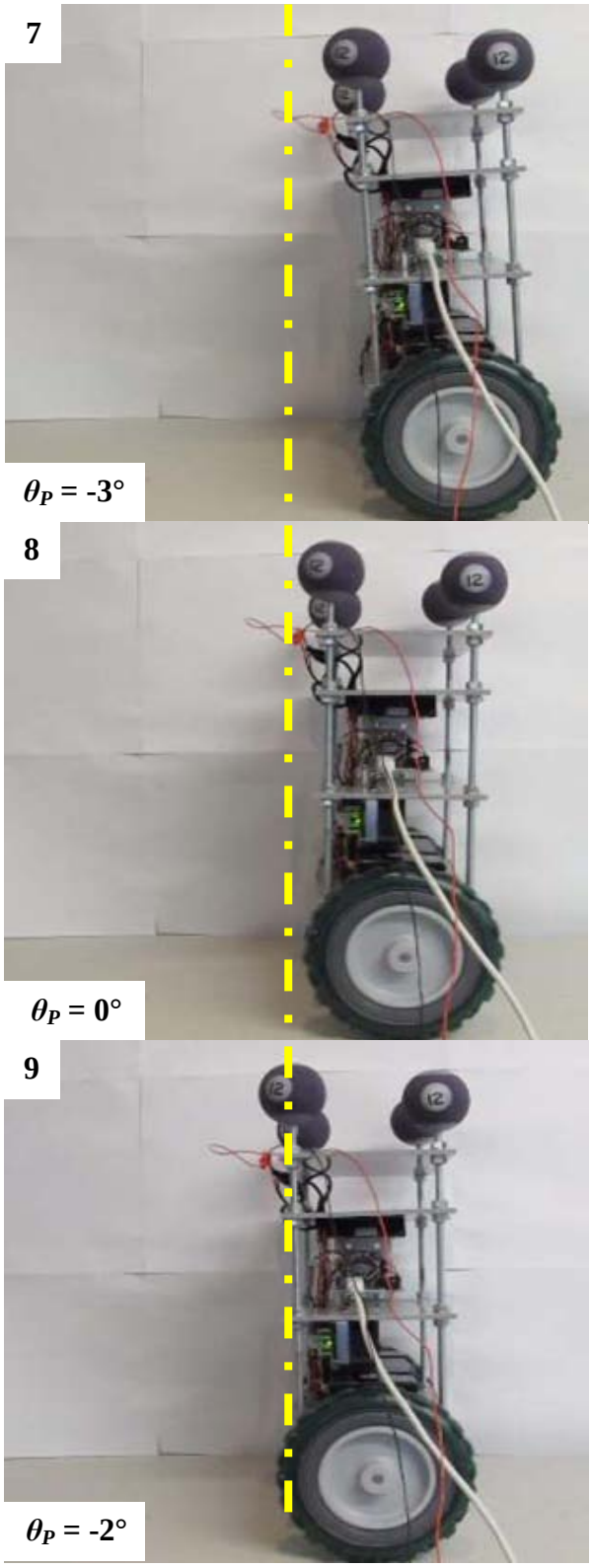
   D = K_d x delta error

The sum of P, I and D generate the required input for HB-25 motor controller.

6.5. Robot Performance




4
θ_P = -1°
5
θ_P = 0°
6
θ_P = -2.5°


7
θ_P = -3°
8
θ_P = 0°
9
θ_P = -2°

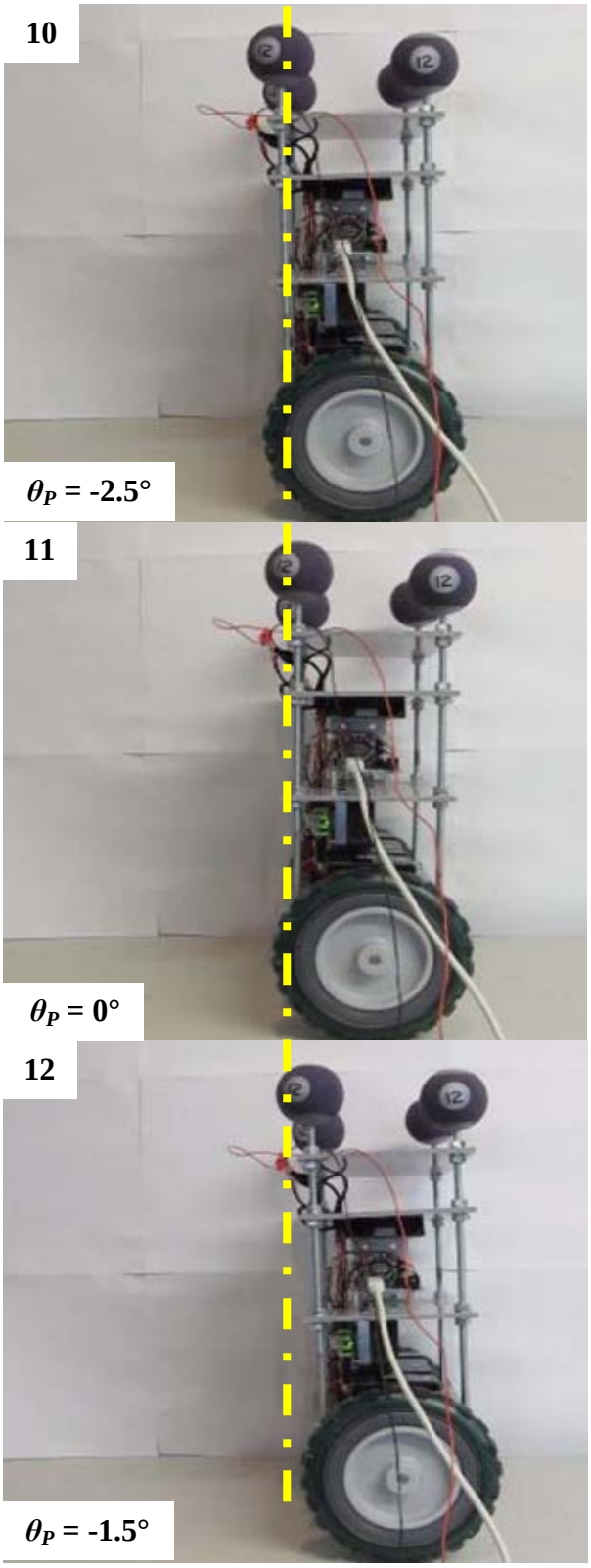


Figure 6.7 - Balancing robot in action

Figure 6.7 illustrates the performance of balancing robot. The sequential pictures in figure 6.7 show that motors turned the wheels to direction of chassis fall and kept the robot upright. Figure 6.8 and 6.9 show the measured pitch angle and displacement of balancing robot respectively while robot was balanced. Figure 6.8 shows that linear fitted line of all collected pitch angles is located at zero degree, where robot stands vertically.

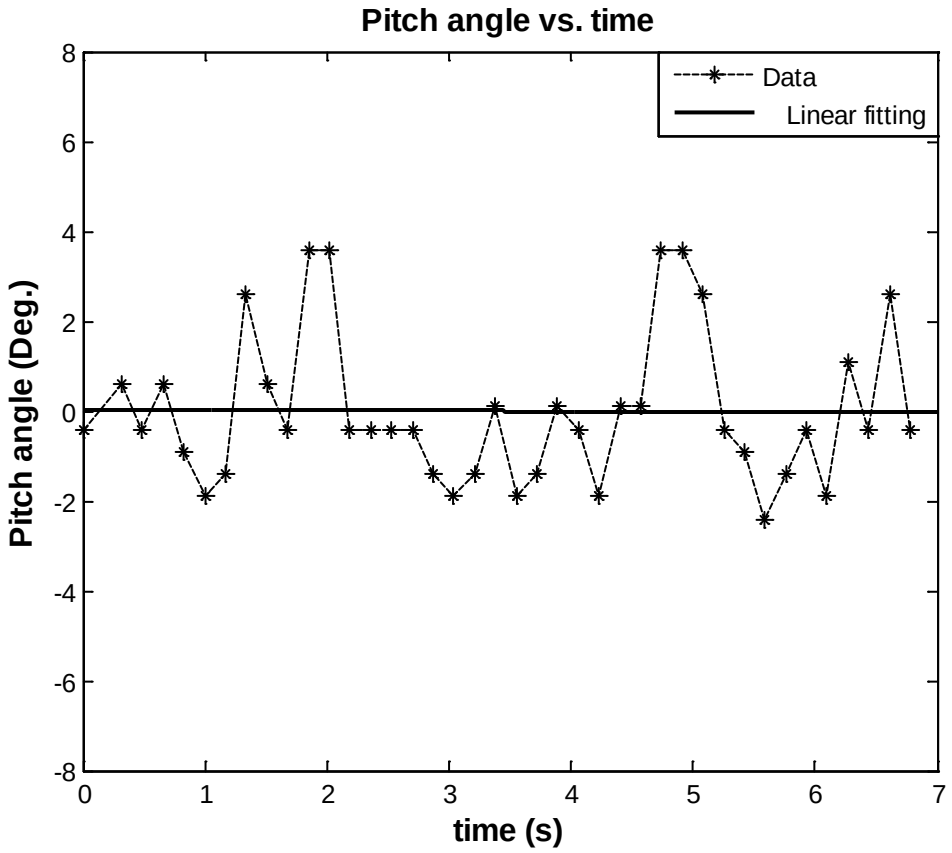


Figure 6.8 - Pitch angle vs. time

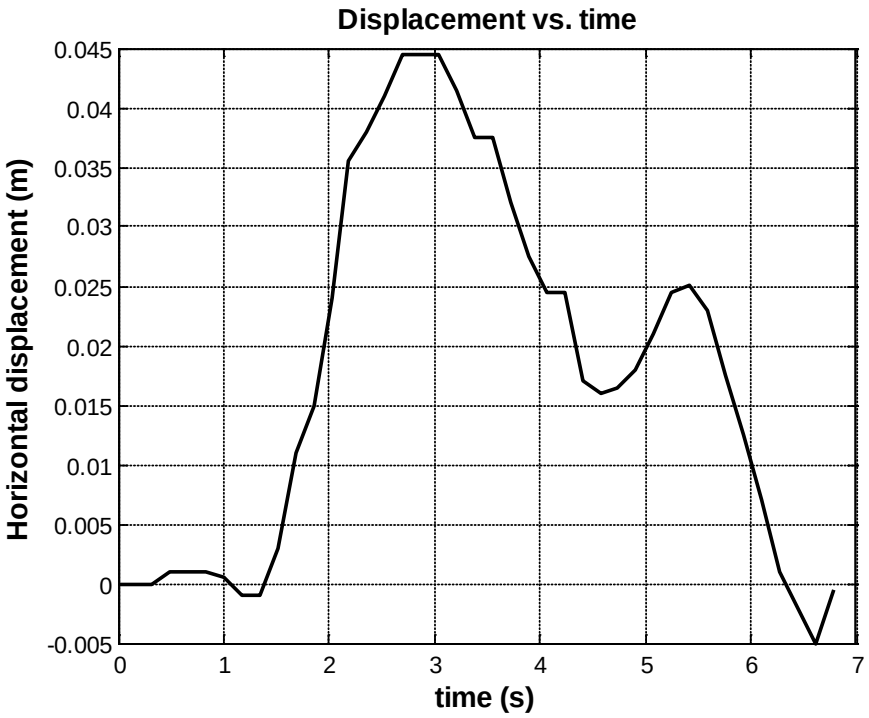


Figure 6.9 - Displacement vs. time

Figure 6.9 illustrates that robot's travelling distance is longer when pitch angle is higher (as shown on first peak) for keeping the robot balance. Figure 6.10 shows the Kalman filter's output in balance mode when a disturbance force disturbed the robot's balancing.

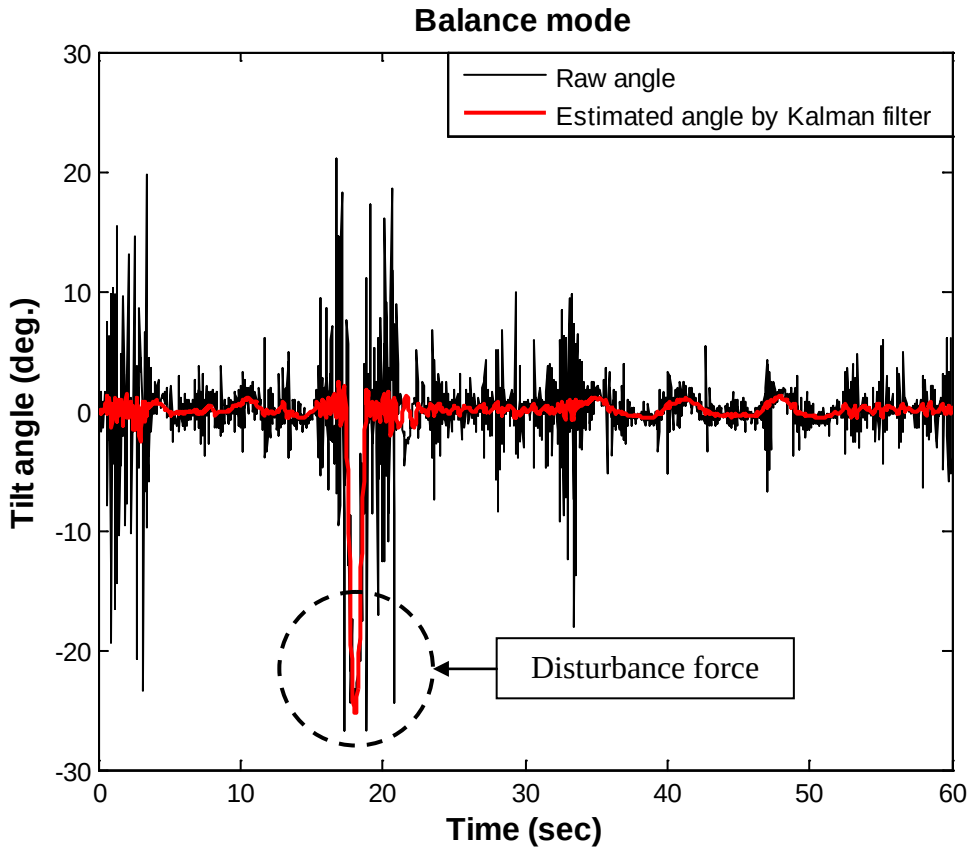


Figure 6.10 - Estimated angle vs. disturbance force

6.6 Conclusion

A closed-loop control unit was implemented and desired performance of robot was observed. Fusion of tilt sensor's output and gyroscope rate by Kalman filter was an essential step for developing a two-wheeled balancing robot that increase the smoothness of robot's manoeuvre. Increasing the number of gyroscope and tilt sensor will improve the performance of balancing robot as well as deploying two encoders and better DC motors with higher torque.

Chapter 7

Overall Conclusion

MASSEY UNIVERSITY
School of Engineering & Advanced Technology

Overall Conclusion

Two-wheeled robots provide higher level of mobility and manoeuvrability. The application of two-wheeled balancing robot varies with different environment and requirements. Understanding the classical theory of inverted pendulum and its dynamic system are initial steps for developing a two-wheeled balancing robot. Safety and robustness of two-wheeled products are so important and these aspects will be achieved by employing accurate sensors and designing a precise control unit. Initial structure of two-wheeled balancing robot was developed. Symmetrical weight distribution on robot's chassis was an important step for designing the structure. Best location of centre of gravity and sensors were determined after mathematical simulation. Differential equations of system were observed and existing constants of equations were measured. State-space model of system were designed for further simulation and analyses. Full-state feedback control was designed successfully to control the unstable two-wheeled balancing robot. Tilt sensor and gyroscope provide the closed-loop system feedback. Reference point was compared with generated feedback and microcontroller was sent the right command to motor controllers. According to observed results of parametric studies, ideal constants were chosen for location of centre of gravity (l) and radius of robot's wheel (R) to improve the robot's performance. A closed-loop control unit was implemented and desired performance of robot was observed. Fusion of tilt sensor's output and gyroscope rate by Kalman filter was an essential step for developing a two-wheeled balancing robot that increase the smoothness of robot's manoeuvre. Increasing the number of gyroscope and tilt sensor will improve the performance of balancing robot as well as deploying two encoders and better DC motors with higher torque.

References

[1] Anderson, D. P. *nbot, a two-wheeled Balancing Robot.* Retrieved February 25, 2010 from http://www.geology.smu.edu/~dpa-www/robo/nbot/

[2] Anybots, Retrieved February 25, 2010 from http://www.anybots.com/

[3] Arduino, *Arduino Duemilanove*, Retrieved May 20, 2010 from http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove

[4] Dorf, R. C., & Bishop, R. H. (2005). *Modern Control Systems* (10th ed.). New Jersey: Pearson Educational International.

[5] Franklin G. F., Powell J. D. & Emami-Naeini A. (2009). *Feedback Control of Dynamic Systems.* New Jersey: Pearson Education Inc.

[6] Grasser, F., D'Arrigo, A., Colombi S. & Rufer A. C. (2002). JOE: A Mobile, Inverted Pendulum. *IEEE Transactions on Industrial Electronics*, Vol. 49, No. 1.

[7] Grewal M. S. & Andrews A. P. (2008). *Kalman Filtering: Theory and Practice Using Matlab* (3rd ed.). New Jersey: A John Wiley & Sons, Inc.

[8] HITACHI. *Research & Development.* Retrieved February 25, 2010 from http://www.hitachi.com/rd/research/robotics/emiew2_01.html

[9] Hudson T. (2003). *tilt.c.* Retrieved September 10, 2010 from http://www.rotomotion.com/downloads/tilt.c

[10] iBot, *iBot 4000*, Retrieved February 25, 2010 from http://www.ibotnow.com/

[11] Jian-hai H., Shu-shang Z., Ji-shun L. & Hang L. (2008). Research on Developed Parallel Two-wheeled Robot and Its Control System. *Proceedings of the IEEE International Conference on Automation and Logistics*, pp. 2471-2475.

[12] Kalra S., Patel D., & Stol K. *Design and Hybrid Control of a Two Wheeled Robotic Platform*. Department of Mechanical Engineering, University of Auckland, New Zealand.

[13] Krey, U. & Owen A. (2007). *Basic Theoretical Physics*. New York: Springer.

[14] Meirovitch L. (1994). *Methods of Analytical Dynamics* (International ed.). New York: McGraw-Hill.

[15] Memarbashi, H. R., Chang J. Y., Saegusa S. (2010). Modelling and Control of an Up-Right Two-Wheeled Intelligent Machine, *ASME-Information Storage and Processing System*, BI-A3:274-276.

[16] Meriam, J. L. & Kraige, L. G. (1997). *Engineering Mechanics: Dynamics* (4th ed.). New York: John wiley & Sons, INC.

[17] Ogata, K. (2002). *Modern Control Engineering* (4th ed.). New Jersey : Prentice Hall.

[18] PARALLAX. *Memsic 2125 Dual-axis Accelerometer*. Retrieved January 15, 2010 from http://www.parallax.com/StoreSearchResults/tabid/768/List/0/SortField/4/ProductID/93/Default.aspx?txtSearch=Dual+Axis+Accelerometer

[19] PARALLAX. *HB-25 Motor Controller*. Retrieved January 15, 2010 from http://www.parallax.com/StoreSearchResults/tabid/768/txtSearch/HB25/List/0/SortField/4/ProductID/64/Default.aspx

[20] Rhodes I. B. (1971). A Tutorial Introduction to Estimate and Filtering. *IEEE Transactions on Automatic Control*, AC-16, (6), pp. 688-706.

[21] Robinson, J. C. (2004). *An Introduction to Ordinary Differential Equations*, United Kingdom: Cambridge University Press.

[22] SEGWAY, Retrieved February 25, 2010 from http://www.segway.com

[23] SILICON SENSING. *CRS03, Angular Rate Sensor*. Retrieved January 15, 2010 from http://www.siliconsensing.com

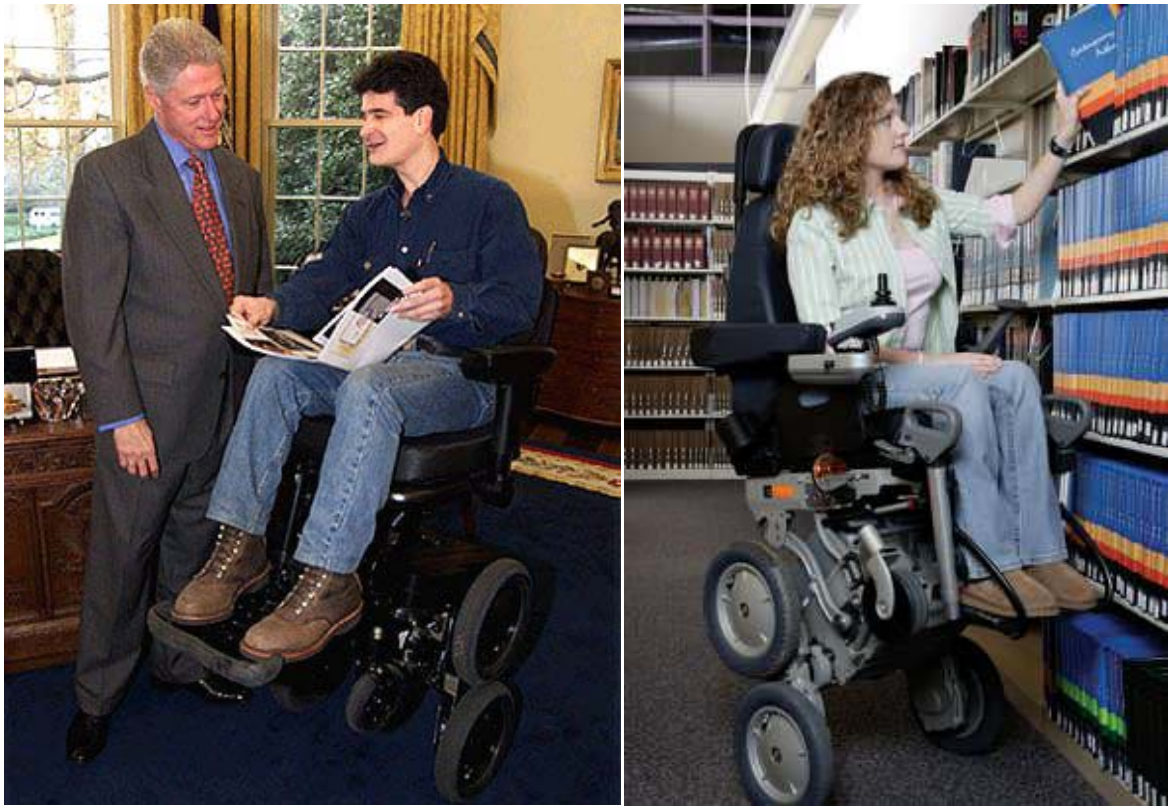
[24] Sorenson H. W. (1970). Least-squares estimation: from Gauss to Kalman. *IEEE spectrum*, pp. 63-68.

Appendices

Appendix 1

Appendix 1.1 - iBot Functionalities

Figure A1.1a illustrates that iBot provides eyes contact for disable person and figure A1.1b illustrates that iBot provides higher level seat for disable person to reach an item located at higher level.



(a) (b)

Figure A1.1 - iBot functionalities

Appendix 1.2 - Hitachi Balancing Robot

Figure A1.2 shows the EMIEWs at office. EMIEW is 70 kg with 130 cm height and EMIEW2 is 13 kg with 80 cm height.

(a) EMIEW (b) EMIEW2

Figure A1.2 - Hitachi balancing robots

Appendix 2

Appendix 2.1 - Arduino Duemilanove Specifications

Table A2.1 - Arduino duemilanove USB module specifications

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analogue Input Pins	6
DC Current per I/O Pin	40mA
DC Current for 3.3V Pin	50mA
Flash Memory	32KB (of which 2KB used by boot loader)
SRAM	2KB
EEPROM	1KB
Clock Speed	16 MHz

Appendix 2.2 - Gyroscope CRS03

Table A2.2 - CRS03 pin definition

Pin Number	Function
1	+VDC
2	Ground (0V)
3	Rate Output

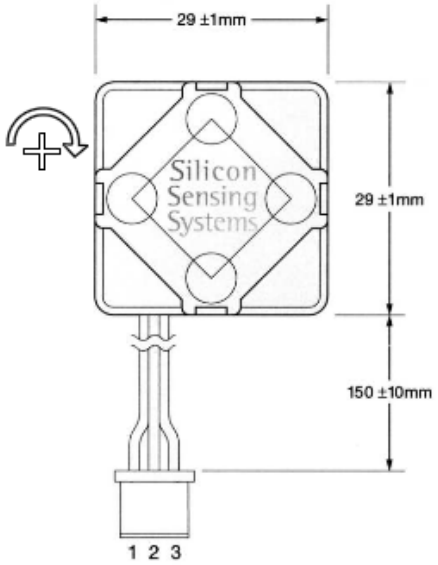


Figure A2.1 - CRS03 pin definition

Appendix 2.3 - Memsic 2125 Dual Axis Accelerometer

Figure A2.2 - Memsic 2125 dual axis thermal accelerometer

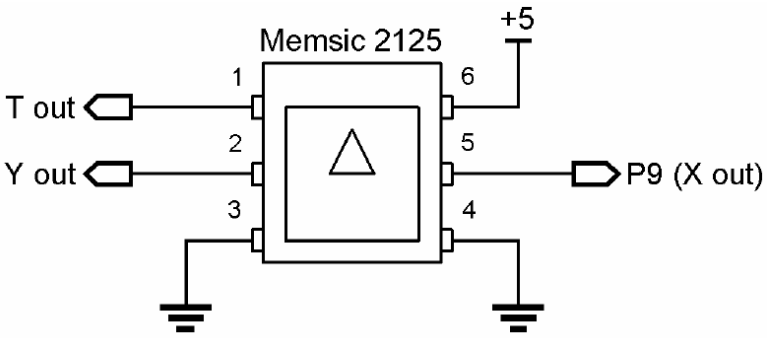


Figure A2.3 - Memsic 2125 pin definitions

Appendix 2.4 - HB-25 Motor Controller

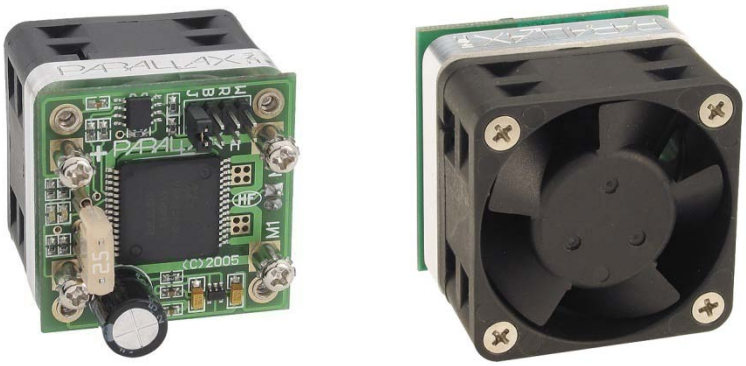


Figure A2.4 - HB-25 motor controller

Table A2.3 - HB-25 pin definitions

Vin	Power input terminal, +6 to +16 VDC
GND	Negative side of battery terminal
M1 & M2	Motor connectors (Polarity reverses direction)
W	Servo pulse input
R	Not connected
B	Servo ground
J	Mode jumper pins

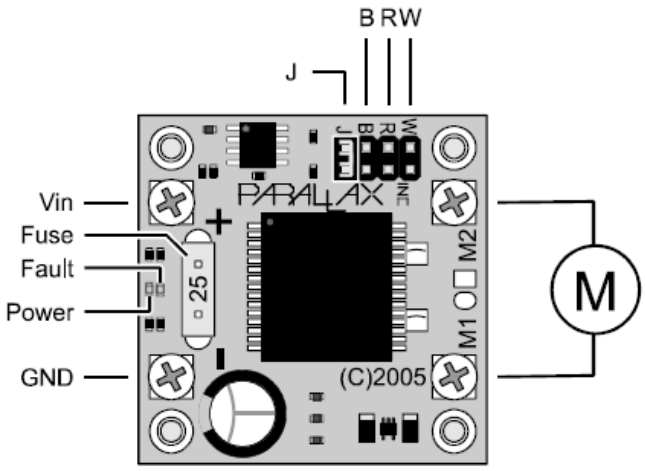


Figure A2.5 - HB-25 pin definitions

Table A2.4 - HB-25 specifications

Motor Supply	6.0 VDC - 16.0 VDC
Load Current	25A Continuous 35A Surge (13.8 V)
Standby Current	50 mA @ 6 V, 80 mA @ 13.8 V (fan on)
PWM Frequency	9.2 KHz
Pulse Input	1.0 ms Full Reverse 1.5 ms Neutral (off) 2.0 ms Full Forward
Modes	Single/Dual Motor Control
Weight	71 grams
Size	40.6 mm × 40.6 mm × 50.0 mm

Appendix 2.5 - EMG30 DC Motor

Table A2.5 - EMG30 gear-head DC motor specifications

Rated Voltage	12 V
Rated Torque	1.5 Kg/cm
Rated Speed	170 rpm
Rated Current	530 mA
No Load Speed	216
No Load Current	150 mA
Stall Current	2.5 A
Rated Output	4.22 W

Figure A2.6 - EMG30 gear-head DC motor

Appendix 2.6 - Physical Model

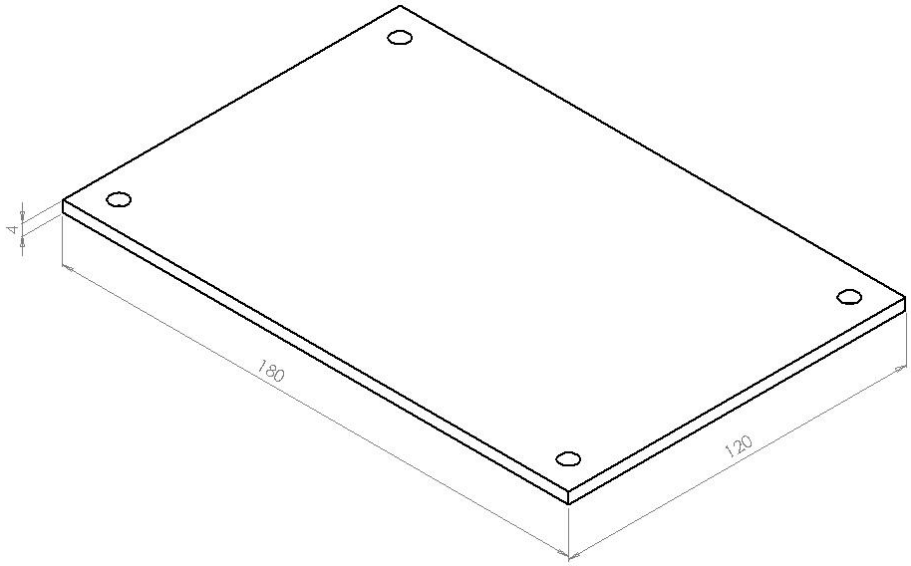


Figure A2.7 - Dimension of plastic levels in millimetres

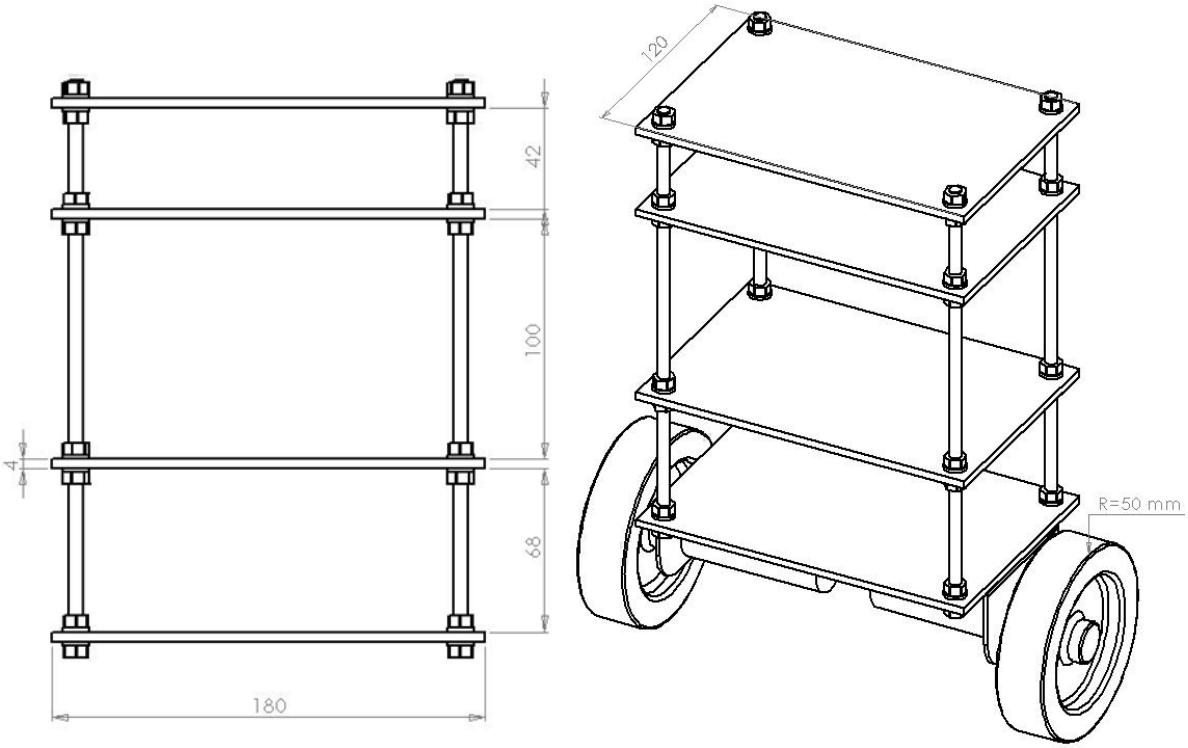


Figure A2.8 - Developed structure for two-wheeled balancing robot

Appendix 3

Appendix 3.1 - Bench Scale

Figure A3.1 - SJ-HS bench scale

Appendix 3.2 - Rotation Inertia of Robot's Chassis

$$I_C = \frac{1}{12} m \left(h^2 + d^2\right)$$

Where,

$h = 0.26\,\text{m}$
$d = 0.12\,\text{m}$

Then,

$$I_C = \frac{1}{12} \times (2.63\,\text{Kg}) \times \left((0.26\,\text{m})^2 + (0.12\,\text{m})^2\right)$$

$$I_C = 0.018\,\text{Kgm}^2$$

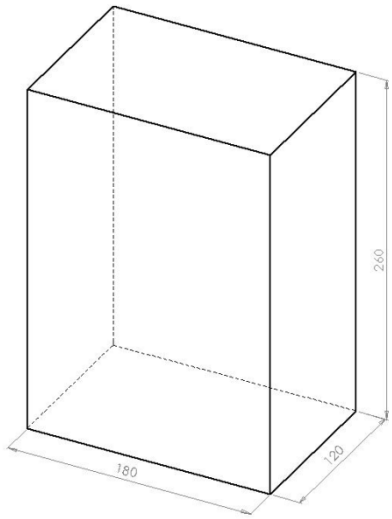


Figure A3.2 - Robot's chassis

Appendix 3.3 - Rotation Inertia of Robot's Wheel

$$I_W = \frac{1}{2} MR^2$$

Where,

$M = 0.14\,\text{Kg}$
$R = 0.05\,\text{m}$

Then,

$$I_W = \frac{1}{2} \times (0.14\,\text{Kg}) \times (0.05\,\text{m})^2$$

$$I_W = 1.75 \times 10^{-4}\ \text{Kgm}^2$$

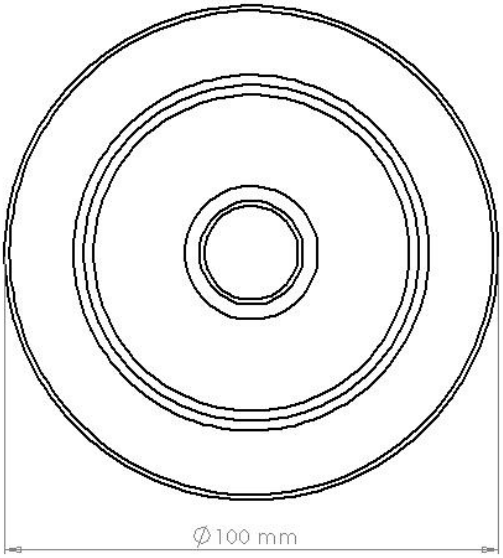


Figure A3.3 – Robot's wheel

Appendix 3.4 - Algebraic calculations for observing $\dot{X}_3$ and $\dot{X}_4$

First step is to cancel the $\dot{X}_4$ term and observe the $\dot{X}_3$.

$$-\frac{ml}{ml^2+J_C}\begin{cases}\left(m+2M+2\dfrac{J_W}{R^2}\right)\dot{X}_3+ml\,\dot{X}_4+2\mu_0\,X_3=0\\ ml\,\dot{X}_3+\left(ml^2+J_C\right)\dot{X}_4-mgl\,X_2+2\mu_1\,X_4=\tau_L+\tau_R\end{cases}$$

$$\begin{cases}\left(m+2M+2\dfrac{J_W}{R^2}\right)\dot{X}_3+ml'\dot{X}_4+2\mu_0\,X_3=0\\ -\dfrac{m^2l^2}{ml^2+J_C}\dot{X}_3-ml'\dot{X}_4+\dfrac{m^2l^2g}{ml^2+J_C}X_2-\dfrac{2ml\mu_1}{ml^2+J_C}X_4=-\dfrac{ml}{ml^2+J_C}\left(\tau_L+\tau_R\right)\end{cases}$$

Equivalent terms of above equations were added together. Therefore,

$$\left[\left(m+2M+2\frac{J_W}{R^2}\right)-\frac{m^2l^2}{ml^2+J_C}\right]\dot{X}_3+\frac{m^2l^2g}{ml^2+J_C}X_2+2\mu_0\,X_3-\frac{2ml\mu_1}{ml^2+J_C}X_4=-\frac{ml}{ml^2+J_C}\left(\tau_L+\tau_R\right)$$

$$\left[\frac{\left(m+2M+2\frac{J_W}{R^2}\right)\left(ml^2+J_C\right)-(ml)^2}{ml^2+J_C}\right]\dot{X}_3+\frac{(ml)^2 g}{ml^2+J_C}X_2+2\mu_0 X_3-\frac{2ml\mu_1}{ml^2+J_C}X_4$$

$$=-\frac{ml}{ml^2+J_C}\left(\tau_L+\tau_R\right) \qquad \text{(A3.1)}$$

The long term in equation A3.1 was replaced with *den* to shorten the equation.

$$den=\left(m+2M+2\frac{J_W}{R^2}\right)\left(ml^2+J_C\right)-(ml)^2$$

Therefore,

$$\left(\frac{den}{ml^2+J_C}\right)\dot{X}_3=-\frac{(ml)^2 g}{ml^2+J_C}X_2-2\mu_0 X_3+\frac{2ml\mu_1}{ml^2+J_C}X_4-\frac{ml}{ml^2+J_C}\left(\tau_L+\tau_R\right)$$

$$\dot{X}_3=-\frac{\frac{(ml)^2 g}{ml^2+J_C}}{\frac{den}{ml^2+J_C}}X_2-\frac{\frac{2\mu_0}{1}}{\frac{den}{ml^2+J_C}}X_3+\frac{\frac{2ml\mu_1}{ml^2+J_C}}{\frac{den}{ml^2+J_C}}X_4-\frac{\frac{ml}{ml^2+J_C}}{\frac{den}{ml^2+J_C}}\left(\tau_L+\tau_R\right)$$

Equation A3.2 shows the equivalent expression of $\dot{X}_3$.

$$\dot{X}_3 = \left[-\frac{(ml)^2 g}{den} \right] X_2 + \left[-\frac{2\mu_0 \left(ml^2 + J_C \right)}{den} \right] X_3 + \left[\frac{2ml\mu_1}{den} \right] X_4 + \left[-\frac{ml}{den} \right] (\tau_L + \tau_R) \qquad \text{(A3.2)}$$

Next step is to cancel the $\dot{X}_3$ term and observe $\dot{X}_4$.

$$-\frac{\left(m + 2M + 2\frac{J_W}{R^2} \right)}{ml} \begin{cases} \left(m + 2M + 2\frac{J_W}{R^2} \right) \dot{X}_3 + ml\dot{X}_4 + 2\mu_0 X_3 = 0 \\ ml\dot{X}_3 + \left(ml^2 + J_C \right) \dot{X}_4 - mglX_2 + 2\mu_1 X_4 = \tau_L + \tau_R \end{cases}$$

$$\begin{cases} \left(m + 2M + 2\frac{J_W}{R^2} \right) \dot{X}_3 + ml\dot{X}_4 + 2\mu_0 X_3 = 0 \\ -\left(m + 2M + 2\frac{J_W}{R^2} \right) \dot{X}_3 - \left[\frac{\left(m + 2M + 2\frac{J_W}{R^2} \right) \left(ml^2 + J_C \right)}{ml} \right] \dot{X}_4 + \left[\left(m + 2M + 2\frac{J_W}{R^2} \right) g \right] X_2 - \left[\frac{2\mu_1 \left(m + 2M + 2\frac{J_W}{R^2} \right)}{ml} \right] X_4 = \left(-\frac{m + 2M + 2\frac{J_W}{R^2}}{ml} \right) (\tau_L + \tau_R) \end{cases}$$

$$\left[\frac{(ml)^2 - \left(m + 2M + 2\frac{J_W}{R^2} \right) \left(ml^2 + J_C \right)}{ml} \right] \dot{X}_4 + \left[\left(m + 2M + 2\frac{J_W}{R^2} \right) g \right] X_2 + 2\mu_0 X_3 - \left[\frac{2\mu_1 \left(m + 2M + 2\frac{J_W}{R^2} \right)}{ml} \right] X_4 = \left(-\frac{m + 2M + 2\frac{J_W}{R^2}}{ml} \right) (\tau_L + \tau_R)$$

$$-\frac{den}{ml}\dot{X}_4=\left[-\left(m+2M+2\frac{J_W}{R^2}\right)g\right]X_2-2\mu_0X_3+\left[\frac{2\mu_1\left(m+2M+2\frac{J_W}{R^2}\right)}{ml}\right]X_4+\left[-\frac{m+2M+2\frac{J_W}{R^2}}{ml}\right](\tau_L+\tau_R)$$

Equation A3.3 shows the equivalent expression of $\dot{X}_4$.

$$\dot{X}_4=\left[\frac{\left(m+2M+2\frac{J_W}{R^2}\right)mgl}{den}\right]X_2+\left(\frac{2\mu_0ml}{den}\right)X_3+\left[-\frac{2\mu_1\left(m+2M+2\frac{J_W}{R^2}\right)}{den}\right]X_4+\left[\frac{m+2M+2\frac{J_W}{R^2}}{den}\right](\tau_L+\tau_R) \qquad \text{(A3.3)}$$

Appendix 4

Appendix 4.1 - Following m file was written to generate the open- and closed-loop transfer functions of displacement and pitch angle. Also the relevant impulse responses were plotted by MATLAB.

```
% close all the open boxes and clear the workspace

clear all;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% defining the constants of linear differential equations of system

g=9.81;          % gravity acceleration (m/s^2)
rad=0.05;        % wheel radius (m)
M=0.14;          % wheel mass (kg)
m=2.63;          % chassis mass (kg)
Jw=1.75e-4;      % rotation inertia of wheel (kg*m^2)
Jc=0.018;        % rotation inertia of body (kg*m^2)
l=0.18;          % distance of the center of mass from the wheel axle (m)
u0=0.1;          % coefficient of friction between wheel and ground
u1=0;            % coefficient of friction between chassis and wheel
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

Appendix 4.2 - Eigenvalues of matrix *A*

$$|A-\lambda I|=0 \tag{A4.1}$$

$$\left|\begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 0 & -24.2421 & -0.2276 & 0\\ 0 & 156.1861 & 1.044 & 0\end{bmatrix}-\lambda\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\right|=0$$

$$\left|\begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 0 & -24.2421 & -0.2276 & 0\\ 0 & 156.1861 & 1.044 & 0\end{bmatrix}-\begin{bmatrix}\lambda & 0 & 0 & 0\\ 0 & \lambda & 0 & 0\\ 0 & 0 & \lambda & 0\\ 0 & 0 & 0 & \lambda\end{bmatrix}\right|=0$$

$$\begin{vmatrix}-\lambda & 0 & 1 & 0\\ 0 & -\lambda & 0 & 1\\ 0 & -24.2421 & -0.2276-\lambda & 0\\ 0 & 156.1861 & 1.044 & -\lambda\end{vmatrix}=0$$

$$-\lambda\begin{vmatrix}-\lambda & 0 & 1\\ -24.2421 & -0.2276-\lambda & 0\\ 156.1861 & 1.044 & -\lambda\end{vmatrix}+1\begin{vmatrix}0 & -\lambda & 1\\ 0 & -24.2421 & 0\\ 0 & 156.1861 & -\lambda\end{vmatrix}=0$$

$$-\lambda\left[-\lambda\begin{vmatrix}-0.2276-\lambda & 0\\ 1.044 & -\lambda\end{vmatrix}+1\begin{vmatrix}-24.2421 & -0.2276-\lambda\\ 156.1861 & 1.044\end{vmatrix}\right]+1\left[\lambda\begin{vmatrix}0 & 0\\ 0 & -\lambda\end{vmatrix}+1\begin{vmatrix}0 & -24.2421\\ 0 & 156.1861\end{vmatrix}\right]=0$$

$$-\lambda\left[-\lambda\left(0.2276\lambda+\lambda^2\right)+\left(-25.3087524+35.54795636+156.1861\lambda\right)\right]=0$$

$$-\lambda\left(-0.2276\lambda^2-\lambda^3+10.23920396+156.1861\lambda\right)=0$$

$$\lambda^4+0.23\lambda^3-156.19\lambda^2-10.24\lambda=0 \qquad \text{(A4.2)}$$

The roots of equation A4.2 are the open-loop system poles.

Appendix 4.3 - Observing the C_M matrix

$$C_M = \left[B \;\middle|\; AB \;\middle|\; A^2B \;\middle|\; A^3B \right]$$

$$C_M = \left[\begin{array}{c|c|c|c} 0 & -5.22 & 0.0012 & -0.8156 \\ 0 & 33.6314 & -0.0054 & 5.254 \\ -5.22 & 1.1882 & -0.8156 & 0.3178 \\ 33.6314 & -5.4498 & 5.254 & -1.7026 \end{array}\right]$$

$$Rank\left(C_M\right) = 4$$

Appendix 4.4 - Observing the O_M matrix

$$O_M = \left[C^T \;\middle|\; A^T C^T \;\middle|\; \left(A^T\right)^2 C^T \;\middle|\; \left(A^T\right)^3 C^T \right]$$

$$O_M = \left[\begin{array}{cc|cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -24.2421 & 156.1861 & 5.5175 & -25.3088 \\ 0 & 0 & 1 & 0 & -0.2276 & 1.044 & 0.0518 & -0.2376 \\ 0 & 0 & 0 & 1 & 0 & 0 & -24.2421 & 156.1861 \end{array}\right]$$

$$Rank\left(O_M\right) = 4$$

Appendix 5

```
#include <math.h>
//*****************************************
//Defining the Constants
//*****************************************
#define   NumReadings     5       // number of reading from gyroscope
#define   GyroPin         0       // assign the gyroscope pin
#define   xPin            2       // tilt sensor output pin for X axis
#define   yPin            3       // tilt sensor output pin for Y axis
#define   HB25R           9       // assign a pin for right motor controller
#define   HB25L           10      // assign a pin for left motor controller
#define   SF              0.02    // a constant used for rounding the gyro's
                                  // output
const     int             Current=0;
const     int             Accumulator=1;
const     int             Previous=2;
const     int             Delta=3;
```

```
//*****************************************
//Defining the variables
//*****************************************
float       error[4];          // define the error matrix with 4 elements
float       oldAngle=0;        // assign the oldAngle as zero when robot is
                               // balance
float       P=0;               // initialise the P, I and D value for PID
                               // controller
float       I=0;
float       D=0;
float       Kp=20;             // assign the Kp, Ki and Kd values of PID
                               // controller
float       Ki=0.2;
float       Kd=1;
float       motors=0;          // initialise the command value for motor
                               // controller
float       dt=0.6;
float       angle=0;           // assign the current angle
float       q_bias=0;          // initialise the gyro bias
float       rate=0;            // assign the unbiased angular rate
float       q_m=0;
```

```
float           R_angle=100;            // assign the covariance noise matrix, which
                                        // is a 1x1 matrix
int             readings[NumReadings];  // define the gyroscope output as a matrix
int             index=0;
int             total=0;                // initialise the total and average of angle
                                        // value
int             average=0;
int             mydt=30;
int             pulseX,pulseY;          // define the required variables for tilt sensor
int             accelerationX,accelerationY;
int             ax_m=0;
int             ay_m=0;
int             cnt=0;


unsigned long  lastread=0;
unsigned long  startmillis=0;


static float        p[2][2]={{1,0},{0,1},};  // assign the covariance matrix of Kalman
                                             // filter
static const float  Q_angle=0.001;
static const float  Q_gyro=0.003;
```

```
//*****************************************
void setup()
{
    Serial.begin(9600);
    pinMode(xPin, INPUT);                    // define the tilt sensor pins as input pins
    pinMode(yPin, INPUT);

    for(int i=0; i<NumReadings; i++)
    readings[i]=0;                           // initialise all the gyroscope readings to zero
    startmillis=millis();
    P=0;
    I=0;
    D=0;
  }
//*****************************************
void loop()
{
    int delta=millis()-lastread;
    if(delta>=mydt)                          // sample gyroscope output every dt ms
    {
```

```
lastread=millis();
total-=readings[index];            // subtract the last gyro reading from total value
readings[index]=analogRead(GyroPin);                    //read gyro's output
readings[index]=((readings[index]/204.8)/SF)-125;       //centring  the gyro's
                                                        // output
total+=readings[index];        // add every gyro reading to total
index=index+1;                 // increase the index of reading

if (index>=NumReadings)     // reset the index if index is greater than 5
index=0;
average=total/NumReadings;                      // averaging the gyro's output
q_m=((float)average)*(1500.0/1024.0)*PI/180;
const   float    q=q_m-q_bias;
const   float    pdot[2*2] =    {
    Q_angle-p[0][1]-p[1][0],
    -p[1][1],
    -p[1][1],
    Q_gyro
};
```

```
rate=q;                          // store the unbiased gyro estimate
angle+=q*dt;                     // update the angle estimate


p[0][0] += pdot[0] * dt;                    // update the covariance
                                            // matrix
p[0][1] += pdot[1] * dt;
p[1][0] += pdot[2] * dt;
p[1][1] += pdot[3] * dt;
pulseX = pulseIn(xPin,HIGH);                // read the tilt sensor's output
ax_m = (((pulseX / 10) - 500) * 8)-7.125;   // X acceleration
float   angle_m=atan2(-4000,ax_m);
angle_m = (angle_m*100)+146.10;             // centring the tilt sensor's
                                            // output


const   float       angle_err=angle_m-angle;    // observing error
const   float       c_0=1;
const   float       PCt_0 = c_0 * p[0][0];
const   float       PCt_1 = c_0 * p[1][0];
const   float       E =R_angle+ c_0 * PCt_0;
const   float       K_0 = PCt_0 / E;
const   float       K_1 = PCt_1 / E;
```

```
const    float          t_0 = PCt_0;

const    float          t_1 = c_0 * p[0][1];


p[0][0] -= K_0 * t_0;

p[0][1] -= K_0 * t_1;

p[1][0] -= K_1 * t_0;

p[1][1] -= K_1 * t_1;

angle += K_0 * angle_err;


q_bias += K_1 * angle_err;

error[Current] = angle-myAngle;


// another method to generate the PID values.

/*P=Kp*error[Current];

error[Accumulator] = error[Accumulator] + error[Current];

I =Ki*error[Accumulator];

error[Delta] =  error[Current]-error[Previous];

D = Kd * error[Delta];

error[Previous]=error[Current];*/
```

```
float     Index=0;
float     myAngle=angle+Index;

P=Kp*myAngle;                    // observing the PID values according to
                                 // Kalman filter's output
I =Ki*myAngle;
D = Kd * myAngle;

if ((angle<-20) || (angle>20))   // stop the motors when robot falls over
{
        motors=185;
} else   {
        motors=P+I+D+194;        // generating the required command for
                                 // motor controllers, based on its calibration
                                 // graph
}

if (motors>=250) motors = 250.0; // defined boundaries for motor's value
if (motors<=110) motors = 110.0;
analogWrite(HB25L,motors-1.2);   // send signal to motor controllers
analogWrite(HB25R,motors);
```

```
                delay(1.5);


                analogWrite(HB25L,0);
                analogWrite(HB25R,0);


                delay(6.0);


                oldAngle=myAngle;
                Serial.print(angle_m);
                Serial.print("\t");
                Serial.print(angle);
                Serial.print("\t");
                Serial.println(ax_m);
        }
}
```