DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DESIGN AND CONTROL OF BALANCING ROBOT

by
Ozan ENGİNOĞLU

July, 2012
İZMİR

DESIGN AND CONTROL OF BALANCING ROBOT

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the
Degree of Master of Mechatronics in
Mechatronics Engineering, Mechatronics Engineering Program

by
Ozan ENGİNOĞLU

July, 2012
İZMİR

M.Sc. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"DESIGN AND CONTROL OF BALANCING ROBOT"** completed by **OZAN ENGİNOĞLU** under supervision of **ASSIST. PROF. TOLGA SÜRGEVİL** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Assist. Prof. Tolga SÜRGEVİL

Supervisor

[illegible]

Assoc. Prof. Zeki KIRAL

(Jury Member)

[illegible]

Assist. Prof. Serkan GÜNEL

(Jury Member)

[illegible]

Prof.Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

First, I would like to express my sincere appreciation to my supervisor Assist. Prof. Dr. Tolga SÜRGEVİL for his support and guidance from the beginning to the very end of this study. Under his guidance I had the chance to learn and improve myself in various subjects which I believe will be of great importance in my future academic studies and career.

I also want to thank Sezai AKTÜRK and “BAST Saç İşleme ve Boya Sanayi Ticaret LTD. ŞTİ.” for providing me the laser cut aluminium sheets to be used in my first balancing robot design.

Finally, I owe my wife and family a big dept of gratitude for always being there for me and making me feel determined to complete my study.

Ozan ENGİNOĞLU

DESIGN AND CONTROL OF BALANCING ROBOT

ABSTRACT

This thesis is concentrated on the problem of balancing a two wheeled robot using the angle feedback. To calculate the pendulum angle, accelerometer and gyroscope sensors are used. The readings received from these sensors are processed using Kalman filtering to eliminate the noise caused by vibration and a clean noiseless signal is obtained. Then PID controller is used on the filtered angle feedback to balance the robot.

Calculations and the control of the robot are performed using a microcontroller, which processes sensor signals and generates logic signals to drive the motors of the robot.

The design used as a prototype is prepared in Solidworks. All parts are drawn in 3D and materials are assigned to these parts to obtain parameters like mass and inertia values of the pendulum and the wheels that are used in simulations.

In order to determine the PID controller parameters to be used in the system, Simulink is used to prepare the block diagram of the whole closed-loop system. For this purpose, mathematical equations of the system are obtained and these equations are implemented in Simulink as balancing robot system block. Also, other system parts like PID controller, PWM pulse generator and Kalman filtering blocks are added into simulation to construct a platform as close as possible to real system. After the simulation model is prepared, design optimization tool in Simulink is used to find PID parameters which give a feasible solution for the control of the balancing robot.

Finally, obtained PID parameters are experimented in the prototype design and the angle of the robot is logged and compared with the simulation results.

Keywords : Balancing robot, inverted pendulum, Kalman filter, PID controller

DENGE ROBOTUNUN TASARIM VE KONTROLÜ

ÖZ

Bu tez iki tekerlekli bir robotu açı geribeslemesi kullanarak dengede tutma problemi üzerine yoğunlaşmıştır. Sarkaç açısının hesaplanması için ivme ölçer ve jiroskop sensörleri kullanılmıştır. Bu sensörlerden gelen veriler, titreşim sonucu oluşan paraziti yok etmek için Kalman filtresi kullanılarak işlenmiş ve temiz, parazitsiz bir sinyal elde edilmiştir. PID denetleyicisi, robotu dengede tutmak için, filtre edilmiş açı geribeslemesi üzerinde kullanılmıştır.

Robot'un hesaplamaları ve kontrolü, motorları sürmek için sensör sinyallerini işleyen ve lojik sinyalleri üreten bir mikrodenetleyici kullanılarak gerçekleştirilmiştir.

Prototip olarak kullanılan tasarım Solidworks'de hazırlanmıştır. Bütün parçalar üç boyutlu olarak çizilmiş ve simulasyonlarda kullanılan sarkaç ve tekerlerin kütle ve atalet momentleri gibi parametrelerinin elde edilebilmesi için bu parçalara malzeme ataması yapılmıştır.

Sistemde kullanılacak olan PID denetleyicisi parametrelerini tespit edebilmek için, bütün kapalı-çevrim sistemin blok diyagramının hazırlanmasında Simulink kullanılmıştır. Bu amaçla sistemin matematiksel eşitlikleri çıkartılmış ve bu eşitlikler Simulink'de denge robotu sistem bloğu olarak uygulanmıştır. Ayrıca PID denetleyici, PWM pulse oluşturucusu ve Kalman filtresi blokları da gerçek sisteme mümkün olduğunca yakın bir platform oluşturmak için simulasyona eklenmiştir. Simulasyon modeli hazırlandıktan sonra, Simulink'deki tasarım optimizasyon aracı kullanılarak denge robotu'nun kontrolü için uygun çözümü veren PID parametreleri bulunmuştur.

Son olarak, elde edilen PID parametreleri tasarlanan prototip'de denenmiş ve robot'un sarkaç açısı simulasyon sonuçları ile karşılaştırılması için kayıt edilmiştir.

Anahtar Sözcükler : Denge robotu, ters sarkaç, Kalman filtresi, PID denetleyicisi

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

Two wheeled balancing robots have attracted many researchers because of its naturally unstable and nonlinear structure. Its properties provide a good environment for researchers to test various control theories on the subject and see how they behave in a complicated nonlinear system like this. Along with its scientific attraction, two wheeled robots have many advantages; they are small, simple and also good at climbing inclined planes, which makes them useful as personal transporting devices. Spot spin feature makes these robots usable in factories or in narrow places etc. (Karla, Dipesh & Stol, 2007).

In order to balance a two wheeled robot, different approaches can be applied on the system. Although the main objective of the system is simple; to stabilize the robot at its vertical position, various types of methods of calculating the pendulum angle of the robot and various filters to eliminate the signal noises and different control theories can be applied to this system. In the literature, various designs and methods can be found for two wheeled balancing robots. Some of them are listed below;

nBot - David P. Anderson from Southern Methodist University has achieved to develop a very stable and award winning balancing robot shown in Figure 1.1. This robot uses an Inertial Measurement Unit (IMU) board consisting of an accelerometer and a gyroscope. Kalman filter is used to filter the readings of these sensors. In addition to angle information, position angle, which is obtained using motor encoders, is also used as a feedback to stabilize the position of the robot in horizontal axis (Anderson, 2010).

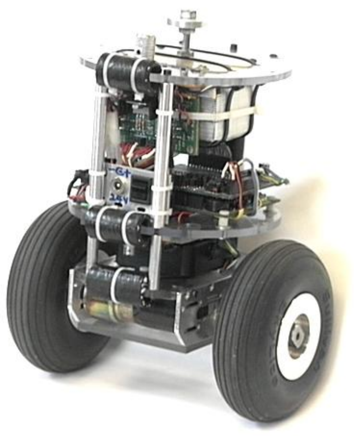


Figure 1.1 View of Nbot balancing robot (Anderson, 2010)

Joe - Researchers developed a two-wheeled balancing robot, shown in Figure 1.2, with weights on the top to simulate a driver standing on the robot. In the Joe balancing robot, encoders of the motors and gyroscope are used to calculate the inclination of the robot. As a controller pole placement method is used and in order to balance the robot, different pole placements are tested (Grasser, D'Arrigo, Colombi & Rufer, 2002).

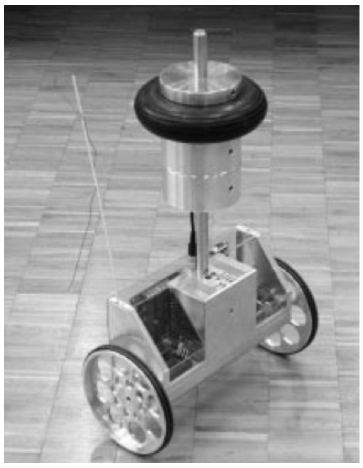


Figure 1.2 View of Joe balancing robot. (Grasser et al., 2002)

Segway - A commercial application of balancing robots has been started to be used with Segway personal transporter as shown in Figure 1.3. It was invented by Dean Kamen and announced in 2001. The design included a base where people would stand and a stick on the front to direct it by the user. Since Segway is designed to carry people, IMU board includes 5 gyroscopes where three are used and two are placed as backup. After the success of Segway, the company has presented different models. For instance adventure, golf, cargo or commuter Segways are among those. Segways can go up to 20 km/h and a range of 38 km. (Segway Personal Transporter Webpage).

Figure 1.3 Segway PT
(Segway Personal Transporter Webpage)

Other than classical control way of balancing; driving the motors according to the pendulum angle, different approaches to the problem are also available in the literature. Kalra et al. (2007) have researched the benefits of a reaction wheel they placed on the top of the robot, which can be seen in Figure 1.4. The aim of adding a reaction wheel to the system is to deliver the required balancing torque with this mechanism instead of base balancing wheels. Linear Quadratic Regulator (LQR) controller is used in this system for balancing the robot. Energy efficiency and balancing are compared with the system without a reaction wheel. As a result, using hybrid mechanism provided a better balancing and used %21 less energy (Kalra et al., 2007).

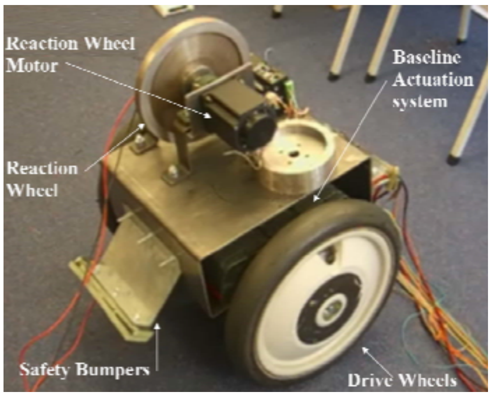


Figure 1.4 Balancing robot with a reaction wheel.
(Karla et al., 2007)

As stated previously, balancing robot is a suitable platform to test control theories. In the research of Lahdhiri, Carnal & Alouani (1994) a fuzzy rule based controller is developed for solving cart - pendulum balancing problem in real time. Similarly Muškinja & Tovornik (2000) have balanced an inverted pendulum on a rail system by using fuzzy logic controller. In their research, balancing starts from the downright position, which is pendulum's initial position and it is swung to turn it around and balance. Combination of control theories is also studied on the balancing robot such as the research of Sun & Gan (2010). In this research, LQR controller is combined with PID controller and they have accomplished to balance the robot within a larger inclination angle.

In this thesis, in order to find the pendulum angle of the robot, an IMU board including an accelerometer and a gyroscope sensor is used and for filtering the noisy signal, Kalman filter is applied. To balance the robot at the upright position, PID controller is implemented on the system in a closed-loop structure.

This thesis is divided into five chapters. In the second chapter, the nonlinear equations of the system are obtained by evaluating the pendulum, the wheels and the DC motors one by one and then combining their equations together. In the third chapter, the prototype design of the balancing robot, which is prepared using Solidworks 3D Cad software is shown. Components used on the balancing robot and also the structures of Kalman filtering and PID controller are explained. In the forth chapter, obtaining system parameters are explained. The open loop and the closed-loop system models and their results are given. The PID controller parameters are obtained using the design optimization tool and they are experimented on the prototype design. The results of the experiment are logged and real system results are compared with the simulation results. In the fifth chapter, conclusions and recommendations are given for future studies.

CHAPTER TWO
SYSTEM MODELING

In order to investigate the behavior of balancing robot through computer simulations, mathematical modeling of the system should be derived at the first step. The dynamic and linearized models based on simple inverted pendulum, which is applied to two wheel balancing robot, can be found in literature (Baik, 2008; Grasser et al., 2002; Jeong & Takahashi, 2008; Kadir, 2005; Ooi, 2003; Quintero, 2008). The modeling of the designed balancing robot based on the derivation of mathematical expressions is shown below. These equations were used to simulate the non-linear behavior of the robot. At the end of the chapter, linear equations are also shown which can be used to obtain linear state space modeling of the system.

The basic scheme of a two wheeled balancing robot is shown in Figure 2.1. In order to obtain these mathematical expressions, balancing robot will be considered in three parts: DC Motor, Wheels and Pendulum. Then combining the equations of these three parts, two nonlinear equations, which represent the whole system, are obtained.

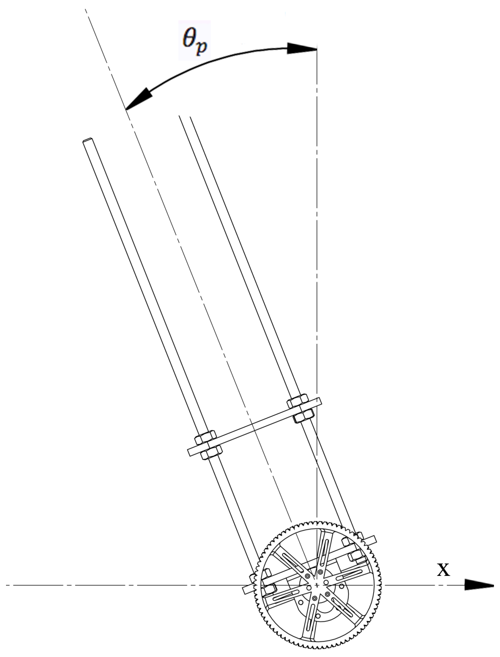


Figure 2.1 Balancing robot basic scheme.

2.1 Direct Current Motor

In the balancing robot two identical DC Motors with a metal gearbox, which has the ratio of 29:1, are used. The motors are very important during balancing and the performance of the balancing robot depends on the capability of the motors like ability to accelerate, backlash etc. Although two separate DC motors are used in the designed system, they can be considered to be a single motor driving a single shaft that delivers torque to the wheels (Kadir, 2005; Ooi, 2003).

The modeling of a single DC motor driving one wheel can be found in (Ooi, 2003) and its diagram is shown in Figure 2.2. When the armature voltage V_a is applied to the motor armature terminals, a current i_a is drawn by the armature (rotor) windings, which has a resistance of R and an inductance of L. With the flowing i_a current in the armature of the motor, a magnetic field, which interacts with the stator magnetic field, is produced. As a result, a torque τ_m is produced acting on the rotor and motor starts to turn with this torque. This torque is varying linearly with the armature current as follows:

$$\tau_m = k_m i_a \qquad (2.1)$$

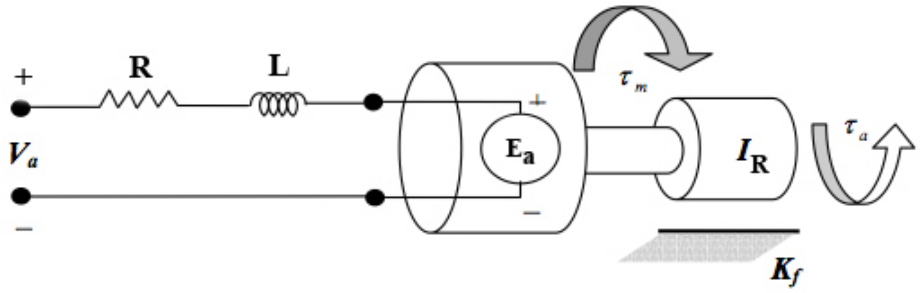


Figure 2.2 Diagram of a DC Motor (Ooi, 2003)

A back electromotive force voltage occurs due to the movement of coils through the magnetic field of the motor and this voltage is always proportional to the angular velocity of shaft as expressed as follows:

$$E_a = k_e \dot{\theta}_W \qquad (2.2)$$

Using *Kirchhoff's Voltage Law* with the motor's electrical circuit, which is shown in Figure 2.2, we obtain the following expression.

$$V_a - Ri_a - L\frac{di_a}{dt} - E_a = 0 \qquad (2.3)$$

The friction torque on the shaft of the motor is assumed to be varying linearly with the shaft angular velocity. Considering this, the sum of the torques acting on the shaft of the dc motor is equal to the product of angular acceleration of the rotor and rotor inertia, i.e,

$$\sum M = I_R \dot{\omega} \qquad (2.4)$$

$$\tau_m - k_f \omega - \tau_a = I_R \frac{d\omega}{dt} \qquad (2.5)$$

Substituting equation (2.1) into (2.5) gives us the following equation.

$$k_m i_a - k_f \omega - \tau_a = I_R \frac{d\omega}{dt} \qquad (2.6)$$

By rearranging (2.6) and leaving the derivative term alone, we find the angular acceleration of the shaft.

$$\frac{d\omega}{dt} = \frac{k_m i_a}{I_R} - \frac{k_f \omega}{I_R} - \frac{\tau_a}{I_R} \qquad (2.7)$$

Substituting (2.2) into (2.3) gives the following equation.

$$V_a - Ri_a - L\frac{di_a}{dt} - k_e \omega = 0 \qquad (2.8)$$

Derivative of the current can be obtained from the equation (2.8) as follows:

$$\frac{di_a}{dt} = \frac{V_a}{L} - \frac{Ri_a}{L} - \frac{k_e\omega}{L} \qquad (2.9)$$

Some given terms shown in DC motor circuit are so small, they can be neglected. In DC Motor circuit these negligible terms are inductance and motor friction constants. Until now they were considered as a part of the circuit.

By neglecting the motor friction in equation (2.7), new equation becomes as follows:

$$\frac{d\omega}{dt} = \frac{k_m i_a}{I_R} - \frac{\tau_a}{I_R} \qquad (2.10)$$

and after neglecting the inductance term equation (2.9) can be expressed in steady state as:

$$i_a = \frac{V_a}{R} - \frac{k_e\omega}{R} \qquad (2.11)$$

The current found in (2.11) is substituted in equation (2.10) to acquire a new equation with the parameters k_m (mechanical motor constant), k_e (electrical motor constant), V_a (armature voltage), ω (angular velocity of the shaft), I_R (Inertia of rotor), R (resistance of motor) and τ_a (torque generated).

$$\frac{d\omega}{dt} = \frac{k_m}{I_R}\left(\frac{V_a}{R} - \frac{k_e\omega}{R}\right) - \frac{\tau_a}{I_R} \qquad (2.12)$$

$$\frac{d\omega}{dt} = \frac{k_m V_a}{I_R R} - \frac{k_m k_e\omega}{I_R R} - \frac{\tau_a}{I_R} \qquad (2.13)$$

The angular displacement of the rotor is expressed as $\dot{\theta_W} = \omega$. This equation is used to form a convenient state-space model. This model represents the dynamics of DC Motors as shown below:

$$\frac{d}{dt}\begin{bmatrix}\theta_W\\ \omega\end{bmatrix} = \begin{bmatrix}\dot{\theta_W}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & -\frac{k_m k_e}{I_R R}\end{bmatrix}\begin{bmatrix}\theta_W\\ \omega\end{bmatrix} + \begin{bmatrix}0 & 0\\ \frac{k_m}{I_R R} & -\frac{1}{I_R}\end{bmatrix}\begin{bmatrix}V_a\\ \tau_a\end{bmatrix} \quad (2.14)$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\theta_W\\ \omega\end{bmatrix} + \begin{bmatrix}0 & 0\end{bmatrix}\begin{bmatrix}V_a\\ \tau_a\end{bmatrix} \quad (2.15)$$

2.2 Wheels

In the project two wheels are used and wheels with high friction surfaces are preferred on purpose. Thus, it is assumed that the wheels will not slip on the ground.

To integrate wheels into other parts of the robot, they will be studied alone using free body diagram method as shown in Figure 2.3 (Ooi, 2003). All the forces applied on the wheels will be shown and Newton's Law of Motion will be used to find the equations of motion. Since the used wheels are same, the equations will be found for just one wheel.

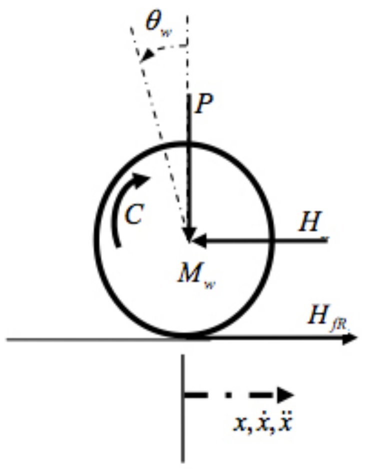


Figure 2.3 Free body diagram of the wheel. (Ooi, 2003)

The sum of all the forces along the x-axis is expressed as:

$$H_f - H = M_w\ddot{x} \quad (2.16)$$

Also, the torques about the center of the wheel should be equal to the inertia of the wheel multiplied by the angular acceleration. Equation of torque equilibrium can be stated as follows:

$$C - H_f r = I_w \ddot{\theta}_w \qquad (2.17)$$

There are two torques on the shaft of the motor. These are τ_m and τ_a and they are always in counter directions. The torque motor produces is the difference between these two. So it can be expressed as

$$C = \tau_m - \tau_a \qquad (2.18)$$

By using (2.5) with k_f is neglected, motor torque can be written as

$$\tau_m = I_R \frac{d\omega}{dt} + \tau_a \qquad (2.19)$$

Substituting equation (2.19) into (2.18) following statement is acquired.

$$C = I_R \frac{d\omega}{dt} \qquad (2.20)$$

τ_a is neglected in the system and substituting equation (2.13) into (2.20) the torque equation is obtained as follows:

$$C = \frac{k_m V_a}{R} - \frac{k_m k_e \omega}{R} - \underbrace{\tau_a}_{=0} \qquad (2.21)$$

Equation (2.21) is substituted into (2.17) and it yields to

$$\frac{k_m V_a}{R} - \frac{k_m k_e \dot{\theta}_w}{R} - H_f r = I_w \ddot{\theta}_w \qquad (2.22)$$

After arranging the equation (2.22), the friction force on the wheel in terms of system parameters is acquired.

$$H_f = \frac{k_m V_a}{Rr} - \frac{k_m k_e \dot{\theta}_w}{Rr} - \frac{I_w \ddot{\theta}_w}{r} \qquad (2.23)$$

Now we can replace the *H*$_f$ term used in (2.16) with the term we just found in (2.23).

$$M_w \ddot{x} = \frac{k_m V_a}{Rr} - \frac{k_m k_e \dot{\theta}_w}{Rr} - \frac{I_w \ddot{\theta}_w}{r} - H \qquad (2.24)$$

As wheels turns, rotational motion is transformed into linear motion. With the following two equations this transformation can be applied on motion equations.

$$\ddot{\theta}_w r = \ddot{x} \qquad (2.25)$$

$$\dot{\theta}_w r = \dot{x} \qquad (2.26)$$

Substituting equations (2.25) and (2.26) in (2.24), the motion equation was found for a wheel as follows:

$$M_w \ddot{x} = \frac{k_m V_a}{Rr} - \frac{k_m k_e \dot{x}}{Rr^2} - \frac{I_w \ddot{x}}{r^2} - H \qquad (2.27)$$

Until now, acquired equations were found considering just one wheel. Since balancing robot has two wheels, the equation (2.27) is multiplied by two and arranged as follows:

$$2M_w \ddot{x} = 2\frac{k_m V_a}{Rr} - 2\frac{k_m k_e \dot{x}}{Rr^2} - 2\frac{I_w \ddot{x}}{r^2} - 2H \qquad (2.28)$$

$$2\left(M_w + \frac{I_w}{r^2}\right)\ddot{x} = \frac{2k_m V_a}{Rr} - \frac{2k_m k_e \dot{x}}{Rr^2} - 2H \qquad (2.29)$$

2.3 Pendulum

As a third and final step towards obtaining the motion equations, pendulum of the balancing robot will be examined. Once it's motion equations are found, it'll also be integrated with the DC motor and wheel equations. The free body diagram of the pendulum is shown in Figure 2.4 (Ooi, 2003).

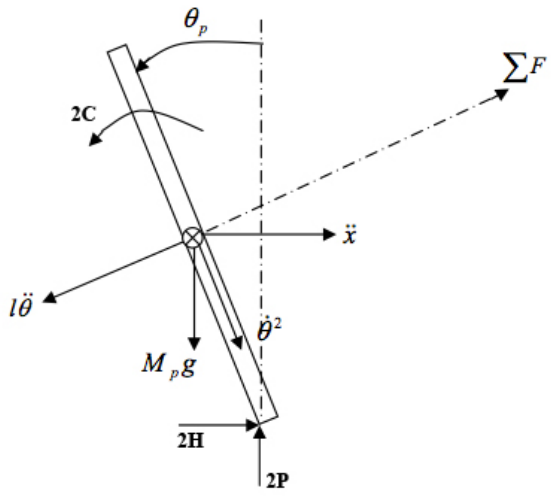


Figure 2.4 Free body Diagram of the pendulum. (Ooi, 2003)

Since the reaction forces of both wheels to the chassis are considered to be the same in direction and value, $H_L = H_R = H$ assumption is made. The sum of all forces along the x-axis is equal to the mass of pendulum multiplied by the acceleration of robot as follows:

$$\sum F_x = M_p\ddot{x} \tag{2.30}$$

$$2H - M_p l\ddot{\theta}_p \cos\theta_p + M_p l\dot{\theta}_p^{\,2} \sin\theta_p = M_p\ddot{x} \tag{2.31}$$

Reaction forces of both wheels on horizontal direction is

$$2H = M_p\ddot{x} + M_p l\ddot{\theta}_p \cos\theta_p - M_p l\dot{\theta}_p^{\,2} \sin\theta_p \tag{2.32}$$

The sum of all the forces perpendicular to the robot chassis are linearly proportional with mass of pendulum and acceleration on that direction.

$$2H\cos\theta_p + 2P\sin\theta_p - M_p g\sin\theta_p - M_p l\ddot{\theta}_p = M_p\ddot{x}\cos\theta_p \quad (2.33)$$

Sum of the moments around the COG of the pendulum is

$$\sum M = I_p\alpha \quad (2.34)$$

$$-2Hl\cos\theta_p - 2Pl\sin\theta_p - 2C = I_p\ddot{\theta}_p \quad (2.35)$$

C torque term was found previously in equation (2.21). After applying linear transformation functions (2.25) and (2.26), torque equation becomes

$$C = \frac{k_m V_a}{R} - \frac{k_m k_e \dot{x}}{Rr} \quad (2.36)$$

$$2C = \frac{2k_m V_a}{R} - \frac{2k_m k_e \dot{x}}{Rr} \quad (2.37)$$

Substituting (2.37) into (2.35), following statements are derived

$$-2Hl\cos\theta_p - 2Pl\sin\theta_p - \frac{2k_m V_a}{R} + \frac{2k_m k_e \dot{x}}{Rr} = I_p\ddot{\theta}_p \quad (2.38)$$

$$-2Hl\cos\theta_p - 2Pl\sin\theta_p = I_p\ddot{\theta}_p + \frac{2k_m V_a}{R} - \frac{2k_m k_e \dot{x}}{Rr} \quad (2.39)$$

By multiplying equation (2.33) with length of pendulum, we get the following equation

$$2Hl\cos\theta_p + 2Pl\sin\theta_p - M_p gl\sin\theta_p - M_p l^2\ddot{\theta}_p = M_p\ddot{x}l\cos\theta_p \quad (2.40)$$

To eliminate *P* and *H* terms from the equations (2.39) and (2.33), equations are summed.

$$\frac{2k_m k_e \dot{x}}{Rr} - \frac{2k_m V_a}{R} - I_p\ddot{\theta}_p - M_p gl\sin\theta_p - M_p l^2\ddot{\theta}_p = M_p\ddot{x}l\cos\theta_p \quad (2.41)$$

Substituting (2.29) into (2.32) also eliminates *H* term from the equation.

$$2\left(M_w + \frac{I_w}{r^2}\right)\ddot{x} = \frac{2k_m V_a}{Rr} - \frac{2k_m k_e \dot{x}}{Rr^2} - M_p\ddot{x} - M_p l\ddot{\theta}_p \cos\theta_p + M_p l\dot{\theta}_p^{\,2} \sin\theta_p \quad (2.42)$$

Rearranging equations (2.41) and (2.42),

$$\ddot{\theta}_p\left(I_p + M_p l^2\right) = \frac{2k_m k_e \dot{x}}{Rr} - \frac{2k_m V_a}{R} - M_p g l \sin\theta_p - M_p \ddot{x} l \cos\theta_p \quad (2.43)$$

$$\ddot{x}\left(\frac{2M_w r^2 + 2I_w + M_p r^2}{r^2}\right) = \frac{2k_m V_a}{Rr} - \frac{2k_m k_e \dot{x}}{Rr^2} - M_p l\ddot{\theta}_p \cos\theta_p + M_p l\dot{\theta}_p^{\,2} \sin\theta_p \quad (2.44)$$

and the nonlinear equations of the system in the final form obtained as follows:

$$\ddot{\theta}_p = \frac{2k_m k_e \dot{x}}{Rr\left(I_p + M_p l^2\right)} - \frac{2k_m V_a}{R\left(I_p + M_p l^2\right)} - \frac{M_p g l \sin\theta_p}{I_p + M_p l^2} - \frac{M_p \ddot{x} l \cos\theta_p}{I_p + M_p l^2} \quad (2.45)$$

$$\ddot{x} = \frac{\frac{2k_m V_a}{Rr} - \frac{2k_m k_e \dot{x}}{Rr^2} - M_p l\ddot{\theta}_p \cos\theta_p + M_p l\dot{\theta}_p^{\,2} \sin\theta_p}{\left(\frac{2M_w r^2 + 2I_w + M_p r^2}{r^2}\right)} \quad (2.46)$$

$$\dot{\theta}_p = \sqrt{\frac{\ddot{x}\left(\frac{2M_w r^2 + 2I_w + M_p r^2}{r^2}\right) - \frac{2k_m V_a}{Rr} + \frac{2k_m k_e \dot{x}}{Rr^2} + M_p l\ddot{\theta}_p \cos\theta_p}{M_p l \sin\theta_p}} \quad (2.47)$$

$$\dot{x} = \frac{Rr\left(\ddot{\theta}_p\left(I_p + M_p l^2\right) + \frac{2k_m V_a}{R} + M_p g l \sin\theta_p + M_p \ddot{x} l \cos\theta_p\right)}{2k_m k_e} \quad (2.48)$$

Equation (2.45), (2.46), (2.47) and (2.48) are the balancing robot's non-linear equations. By using (2.45) and (2.46) it is possible to simulate the behavior of the balancing robot system.

2.4 Linearization of Equations.

By simplifying non-linear equations given above, they will be turned into linear equations and state space model of the system will be obtained. In order to linearize these equations, some assumptions should be made. It is assumed that the pendulum is away from the upright position by an angle of ϕ, which is a very small angle. Hence,

$$\theta_p = \pi + \phi \qquad (2.49)$$

$$\cos\theta_p = -1 \qquad (2.50)$$

$$\sin\theta_p = -\phi \qquad (2.51)$$

$$\left(\frac{d\theta_p}{dt}\right)^2 = 0 \qquad (2.52)$$

By applying all given three assumptions, non-linear equations of balancing robot turns into two linear equations as follows:

$$\ddot{\phi} = \frac{M_p l}{(I_p + M_p l^2)}\ddot{x} + \frac{2k_m k_e}{Rr(I_p + M_p l^2)}\dot{x} - \frac{2k_m}{R(I_p + M_p l^2)}V_a + \frac{M_p g l}{(I_p + M_p l^2)}\phi \qquad (2.53)$$

$$\ddot{x} = \frac{2k_m}{Rr\left(2M_w + \frac{2I_w}{r^2} + M_p\right)}V_a - \frac{2k_m k_e}{Rr^2\left(2M_w + \frac{2I_w}{r^2} + M_p\right)}\dot{x} + \frac{M_p l}{\left(2M_w + \frac{2I_w}{r^2} + M_p\right)}\ddot{\phi} \qquad (2.54)$$

Linearized equations (2.53) and (2.54) can be expressed in state space model form. Since equations are complex, two new terms are given as α and β to simplify the equations.

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\phi}\\ \ddot{\phi}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \frac{2k_m k_e(M_p l r - I_p - M_p l^2)}{Rr^2\alpha} & \frac{{M_p}^2 g l^2}{\alpha} & 0\\ 0 & 0 & 0 & 1\\ 0 & \frac{2k_m k_e(r\beta - M_p l)}{Rr^2\alpha} & \frac{M_p g l \beta}{\alpha} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix} + \begin{bmatrix}0\\ \frac{2k_m(I_p - M_p l^2 - M_p l r)}{Rr\alpha}\\ 0\\ \frac{2k_m(M_p l - r\beta)}{Rr\alpha}\end{bmatrix}V_a \qquad (2.55)$$

where,

$$\alpha = \left[I_p\beta + 2M_p l^2\left(M_w + \frac{I_w}{r^2}\right)\right] \qquad (2.56)$$

$$\beta = \left(2M_w + \frac{2I_w}{r^2} + M_p\right) \qquad (2.57)$$

CHAPTER THREE
DESIGN OF BALANCING ROBOT SYSTEM

The balancing robot system consists of a lot of important parts, which should be carefully studied in order to understand how it works. Every part used in this project is carefully selected considering system requirements. In this chapter, the subsystems constituting the balancing robot and their working principles are explained. Also, the electrical connection diagrams of the electrical components used in the design of the robot are given.

3.1 Mechanical Design

Making a robot balance itself can be accomplished by a combination of a well-designed control system, right chosen sensors & actuators and a well-balanced mechanical design. A good mechanical design is important, because, even the subsystems are properly chosen, the robot will not be able to balance itself without it.

Design of the balancing robot has been performed using Solidworks 3D CAD Software. The rendered view of the robot drawing obtained from Solidworks is shown in Figure 3.1. This software is chosen because it is based directly on 3D modeling of the mechanical parts. Each mechanical part of the robot can be drawn and then these parts can be assembled in the software. Also, it is possible to assign materials to the created parts and acquire realistic technical properties of the whole assembly. It is also possible to make static and dynamic analysis of the robot but in this thesis Solidworks is used for drawing and modeling purposes. The inertia of the wheels and the chassis, which are used in simulation models, are obtained using this software. Otherwise, an experimental setup would be needed to calculate these inertia values of the robot.

For this balancing robot two different designs are manufactured. At the beginning, the design shown in Figure 3.1 has been made. But since this design consists of a lot of aluminum plates especially at the lower levels of the robot, COG of the robot was close to the rotational axis of the wheels.

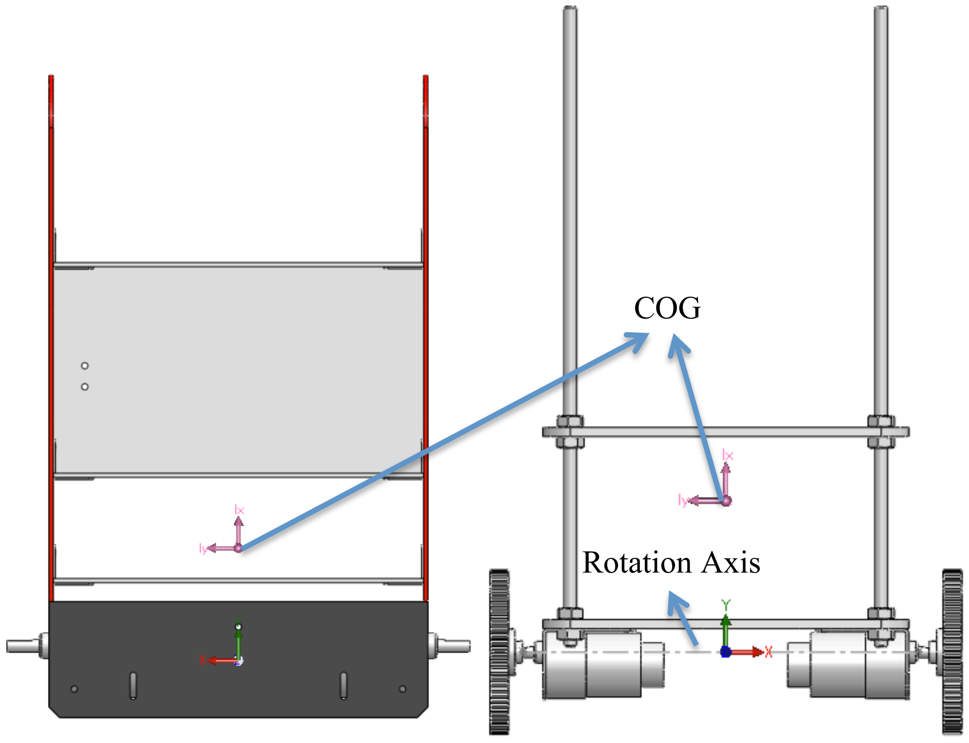


Figure 3.1 First (left) and second (right) manufactured balancing robot designs.

Since COG of the balancing the robot is an important parameter of the system and the higher COG the easier it balances, another design is manufactured with a higher COG. The comparison of COG of both balancing robots can be seen in Figure 3.1. In the new design, the length between the COG of the pendulum and the rotational axis is increased from 6 cm to 8.3 cm. The pendulum's total height is also increased from 30.5 cm to 35.5 cm. Pendulum weight of the robot has reduced from 1.150 kg to 1.086 kg. The system parameters obtained from new design's properties are listed in Table 4.1 in Section 4.

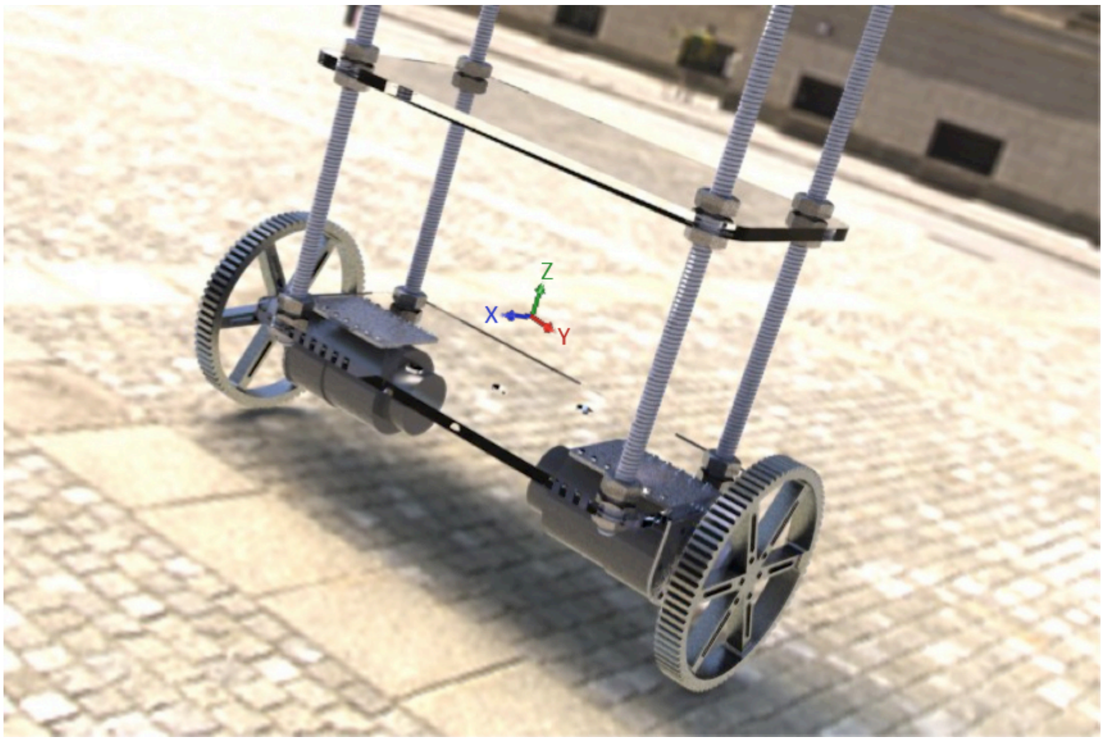


Figure 3.2 Rendered view of designed balancing robot in Solidworks

While designing a balancing robot, there are some important aspects that have to be considered. The most important one is center of gravity (COG) of the robot. Considering the coordinate center as shown in Figure 3.2 (Solidworks drawing showing COG and X,Y,Z coordinates), X and Y coordinates of COG should be as close as possible to the coordinate center (i.e., zero) because if these two coordinates are far from zero, the robot is going to lean in these directions. Since leaning of the robot by itself without any motor force is not intended, the COG should be optimally located on the center of X and Y coordinates. Also, Z coordinate of the COG should be as high as possible for a proper balancing. In order to move the Z coordinate of the COG upwards, components with more weight (battery, microcontroller etc.) should be placed at the top of robot.

The easiest way to explain the concept of moving the COG to a higher position is by imagining a person trying to balance a small and then a tall stick on his/her finger. It is easier to balance a tall stick rather than small one because of the rotational inertia. Inertia increases proportionally with the mass and square of length between

COG and rotational axis. When the rotational inertia is high, it flips over slower and gives the robot some more time to be balanced.

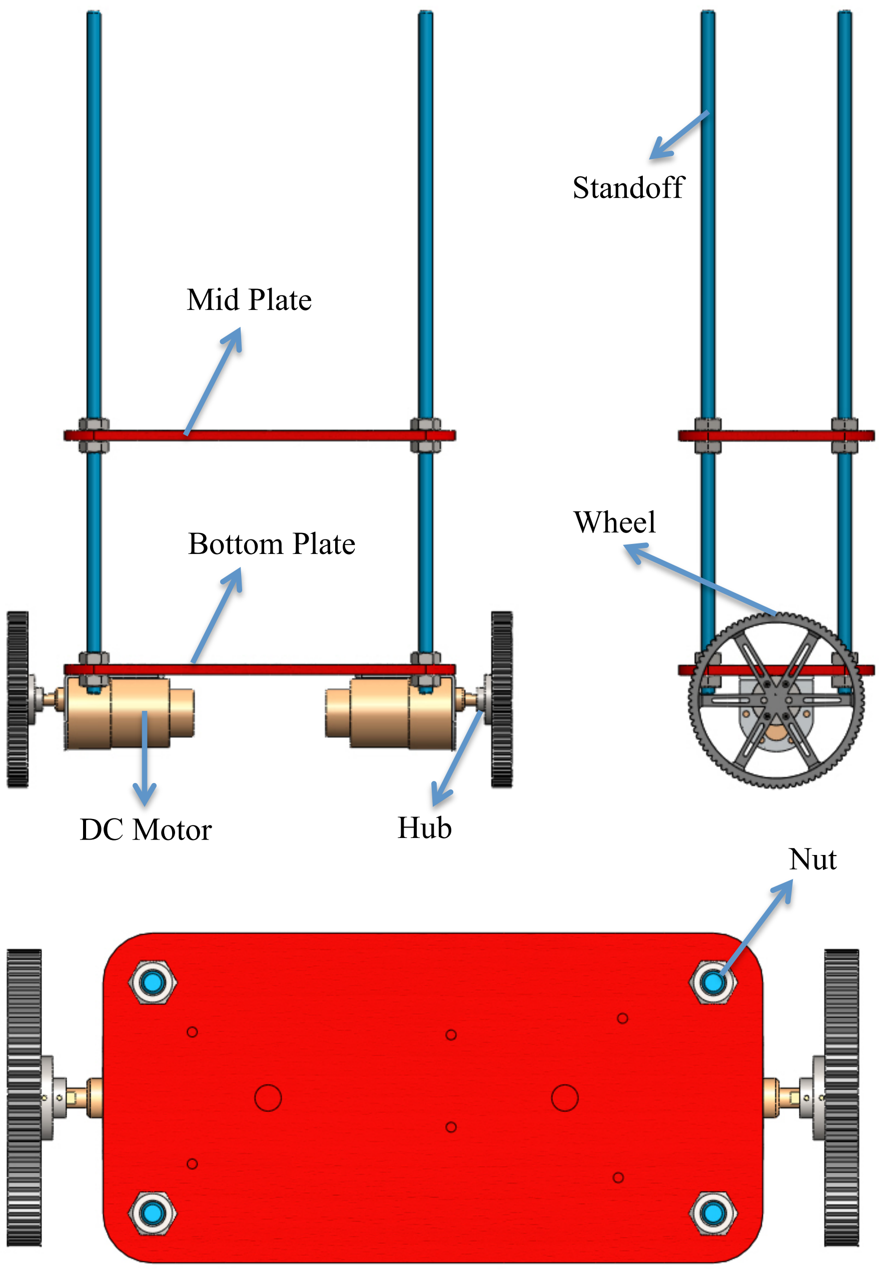


Figure 3.3 Front and left and top views of the balancing robot.

In Figure 3.3, front, left and top views of the robot obtained from Solidworks drawings are given. The list of the parts shown in Figure 3.3 is as follows:

- Bottom plate (made of light wood)
- Middle plate (made of light wood)
- Standoffs (made of aluminum)
- Nuts (made of steel)
- DC Motors (Pololu 12 V Motors with encoders)
- Hub (Used to connect wheels to the shaft of the motor)
- Motor connector (used to connect DC motor to the plate)
- Wheel (Pololu wheels with high friction)

During the design of the robot, sensors (accelerometer and gyroscope), motor driver and microcontroller boards are not modeled since they are light so they don't have a significant effect on calculation of inertia. The masses of these parts are added while assigning materials to the plates to which they are attached in Solidworks model of the robot.

In the design of the robot, an easy assembly of the sensors, motor driver, DC motors, and microcontroller were considered. Accelerometer - Gyroscope combo board and motors are mounted on the bottom plate, while microcontroller and motor driver boards are mounted on the middle plate. Since COG is important, a symmetrical placement of these components is necessary. Also, the most critical part of the system; accelerometer – gyroscope combo board is located close to the rotational axis of DC motors. The reason it is placed close to the rotational axis is to avoid centrifugal and inertial forces acting on the transducers.

During the placement of the components, it was necessary to fix them to their places properly. The vibration of the parts may cause an error in sensor readings, which are acquired from Acc-Gyro board. Especially, accelerometer is very sensitive to vibration. That is why the accelerometer is used together with the gyroscope module in the combo board. The output of the Acc-Gyro combo board is filtered to obtain a noiseless signal.

3.2 Electrical Design

Main electrical and electronic components used in the control of the robot will be introduced here. These components are as follows;

- Microcontroller Board (Arduino Uno)
- DC Motor (Pololu DC Motor with 29:1 metal gearbox)
- DC Motor Driver (Dual VNH3SP30 Motor Driver Carrier MD03A)
- Acc-Gyro Combo Board (IMU 5 Degrees of freedom IDG500/ADXL335)

3.2.1 Microcontroller Board

Arduino Uno microcontroller board, shown in Figure 3.4, is an open source electronics prototyping platform designed with ATmega328 microcontroller. It is designed for easy use and compatibility with different platforms. Since it is provided by a USB interface, it is possible to supply power to the board and upload programs to the microcontroller. This facility makes programming much easier to the user.

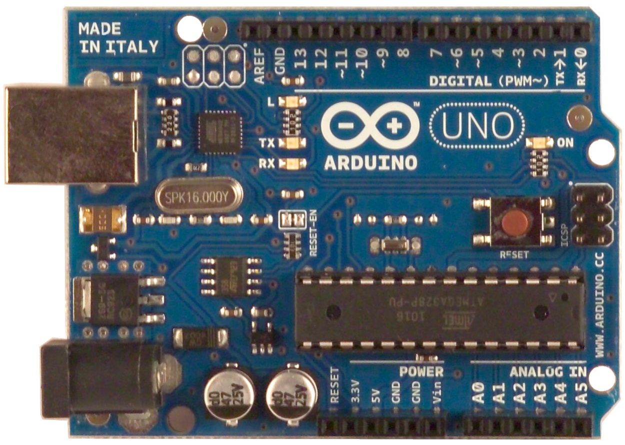


Figure 3.4 Arduino Uno front view (Arduino Uno Webpage)

ATmega328 microcontroller can be programmed using C programming language provided in Arduino development environment. The uploaded programs on the Arduino can be run standalone or, if desired, by means of serial communication which enables to get data from microcontroller to PC simultaneously using a USB cable. All data that has been used to draw graphs are acquired via serial port connection of Arduino board to PC. Some of the features of the Arduino Uno microcontroller board are 14 digital I/O pins (6 of them provide PWM output), 6 analog input pins, 32 KB flash memory, and 10-bit ADC. The detailed specifications of Arduino UNO microcontroller board can be found in (Arduino Uno Webpage). In this system 3 analog input pins are used to get data from Acc-Gyro combo board, 4 digital output pins are used to send direction data to the motor driver for each motor, and 2 PWM output pins are used to set the speeds of the DC motors.

Filtering of the input signals, PWM waveform generation, and closed-loop control algorithm for balancing the robot were implemented for the Arduino microcontroller using development software.

3.2.2 DC Motor

A right chosen DC motor is of great importance for this project because even the drive signal sent from the microcontroller to DC motor driver is correct, if the DC motors are not capable of working with the desired RPM (revolutions per minute), it would be impossible to balance the robot. In Figure 3.5 the isometric view of the DC motors is given. The specifications of the DC motors used in this thesis are given in Table 3.1 (Pololu DC Motor Webpage).

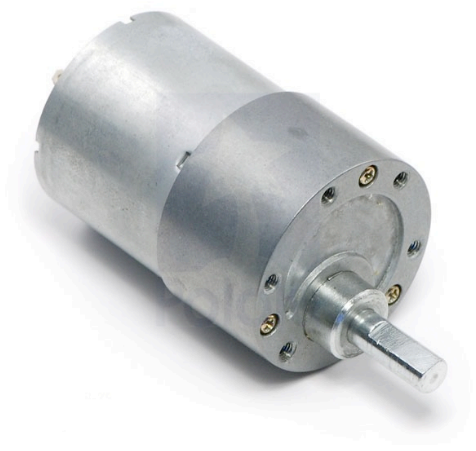


Figure 3.5 Pololu DC Motor isometric view
(Pololu DC Motor Webpage)

Table 3.1 Specifications of Pololu DC Motor
(Pololu DC Motor Webpage)

Gear Ratio	29:1
Free Run Speed (at 6 Volts)	175 RPM
Free Run Current (at 6 Volts)	250 mA
Stall Current (at 6 Volts)	2500 mA
Stall Torque (at 6 Volts)	55 oz.in
Free Run Speed (at 12 Volts)	350 RPM
Free Run Current (at 12 Volts)	300 mA
Stall Current (at 12 Volts)	5000 mA
Stall Torque (at 12 Volts)	110 oz.in

With the new design, the DC motors have been also changed. The old DC motor used with the old design was a Faulhaber 2342012CR (Faulhaber DC Motor Datasheet). This motor had a gearbox with a high reduction ratio (69:1) and free run speed of 117 RPM, which increases the torque but decreases the speed of the shaft. Maximum speed of used DC motors are important especially when the pendulum angle of the balancing robot gets away from the upright position because of a force input from outside or irregular ground surface where motors need to run at full speed to get under the pendulum to balance itself again. If the motor speed is not enough, falling pendulum angle can not be corrected and balancing becomes impossible.

3.2.3 DC Motor Driver

In order to drive two DC motors, the VNH3SP30 Dual DC motor driver shown in Figure 3.6 is used. DC motors that have been used for this project are supplied with 12 Volts and each draw approximately 5 A at stall state.

Figure 3.6 VNH3SP30 Dual DC motor driver
(Pololu DC Motor Driver Webpage)

Since the output pins of the microcontroller unit (Arduino UNO) can supply up to 40 mA, a motor driver unit is necessary to drive these motors. After considering several DC motor drivers, Pololu VNH3SP30 Dual DC motor driver is chosen. Some of the features of VNH3SP30 DC motor driver are 10 kHz of maximum PWM frequency and operating supply voltage up to 36 V. The detailed specifications of the driver are given in (VNH3SP30 Datasheet). This driver can drive motors up to current of 20 A and since the motors used in the project draw up to 5 A current each, this driver will be enough for the purpose.

Motor driver is connected to the microcontroller from its IN_A (clockwise input), IN_B (counter clockwise input) and PWM (pulse width modulation input) pins and motors' positive and negative terminals are connected to the driver's OUT_A and OUT_B pins (shown in Figure 3.14). By using these pins with the Arduino development environment it is possible to rotate the motor in a desired rotation, specify its speed and make the motor break its speed.

In the Table 3.2 truth diagram of the motor driver is given (VNH3SP30 Datasheet). So, for instance, if a connected motor is going to be turned in clockwise direction, a HIGH signal (+5 Volts) needs to be send to IN_A pin and a LOW signal (0 Volts) to the IN_B pin. These two pins are used to define the rotation direction of the motors. An additional PWM pin on the motor driver is used to give the motor its speed. EN_A and EN_B pins shown in Figure 3.4 are statuses of high-side and low-side switches and they need to be externally pulled high for functions to actualize. In case of fault detection these two pins are pulled low by the device.

Table 3.2 Truth table of the motor driver.
(VNH3SP30 Datasheet)

IN_A	IN_B	EN_A	EN_B	OUT_A	OUT_B	Comment
1	1	1	1	HIGH	HIGH	Break to V_{CC}
1	0	1	1	HIGH	LOW	Clockwise
0	1	1	1	LOW	HIGH	Counter Clockwise
0	0	1	1	LOW	LOW	Break to GND

The working principle of a DC motor drive can be understood from the given schematic in Figure 3.7. H-bridge schematic and its connections with the microcontroller are given below.

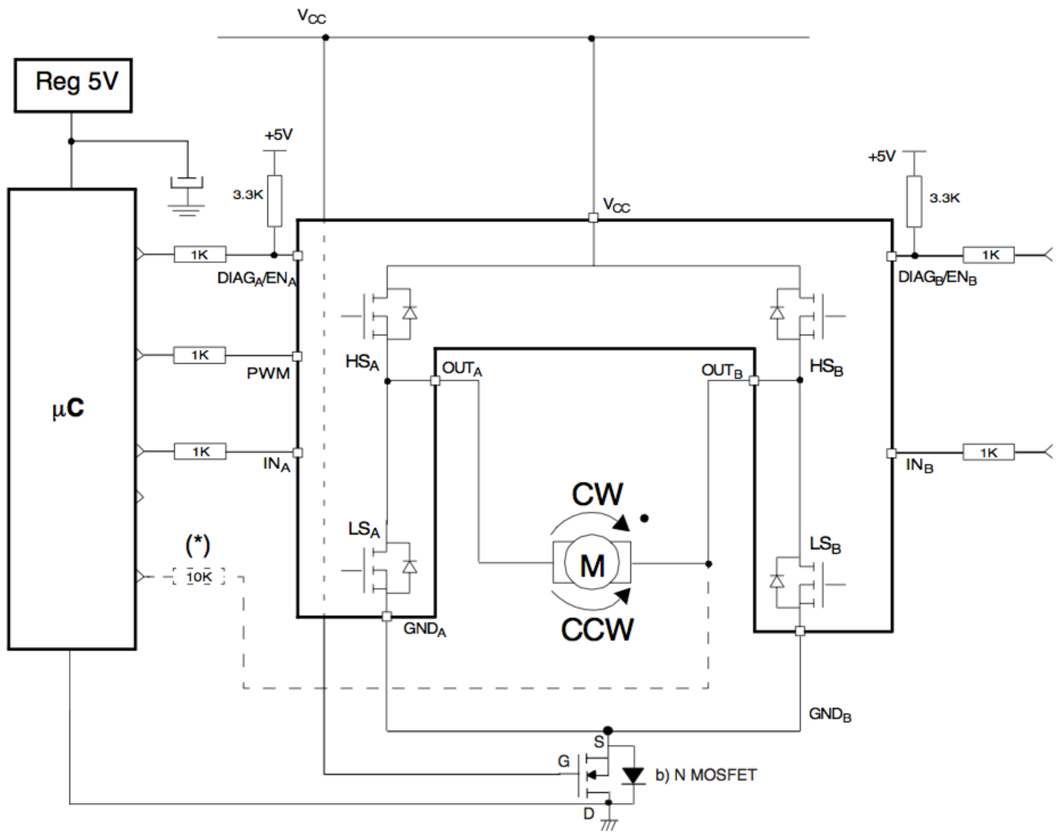


Figure 3.7 H-Bridge circuit of the dc motor driver.
(VNH3SP30 Datasheet)

According to the Table 3.2, if the DC motor is going to rotate clockwise, IN_A pin is set HIGH and IN_B pin is set LOW. In this case V_{CC} voltage (12 Volts motor supplying voltage) connects to the ground over HS_A and LS_B transistors. Conversely, if the motor is going to rotate counter clockwise, IN_A pin is set as LOW and IN_B pin is set as HIGH. This pin configuration makes the supplying voltage go over HS_B and LS_A transistors to the ground and gives the motor a reverse motion.

3.2.4 IMU Analog Combo Board

The Combo IMU board that has been used in the system has two sensors on it. One of them is the accelerometer and the other one is gyroscope. Using the measurements of these two sensors, a filtered and clean signal is acquired which gives inclination of the robot relative to the ground plane. The basic reason of combining the measurements of accelerometer and gyroscope is that both units have flaws if they are used separately. In Figure 3.8 IMU combo board is shown.

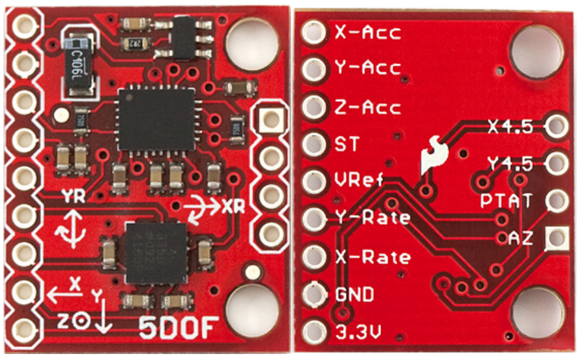


Figure 3.8 Top and bottom layer view of combo board with IDG500 Gyroscope and ADXL335 Accelerometer. (Sparkfun IMU Board Webpage)

Accelerometers are very sensitive to noises like motor vibration. Since two DC motors are used in the system and IMU board is placed very close to them, the vibration noise caused by the motors is a big problem while reading realistic results from the sensor.

Gyroscope sensor doesn't easily get affected from the vibration noise like accelerometer does but it has drifting problem due to the integration while calculating the angular velocity. That means, the readings from the gyroscope will linearly drift over time with the ratio stated in its datasheet.

Fortunately it is possible to combine these two sensors, eliminate the flaws and acquire accurate information from them using sensor fusion methods (such as Kalman filtering, Bayesian and Dempster-Shafer inference methods) (Elmenreich, 2002). Before getting into sensor fusion, these individual sensors will be explained in detail in the following sections.

3.2.4.1 Accelerometer

General specifications of the ADXL335 accelerometer are shown in Table 3.3 (ADXL335 Datasheet). When using this sensor, data from ACC_X and ACC_Z pins are used in order to calculate the angle (θ_p) between gravitation force vector and Z coordinate which indicates leaning of the robot through the forward / backward direction.

Table 3.3 Specifications of ADXL335 accelerometer.
(ADXL335 Datasheet)

Measurement Rate	$\pm$3.6 g
Sensitivity at $X_{OUT}, Y_{OUT}, Z_{OUT}$	300 $\frac{mV}{g}$
Zero g Voltage at $X_{OUT}, Y_{OUT}, Z_{OUT}$	1.5 V
Supply Current	350 μA
Reference voltage (V_{ref})	3.3 V

To understand the working principles of this unit an imaginary cube shaped box with a ball inside will be imagined. It is assumed that the box is in a no gravitation area that can not affect the position of the ball. So the ball will float in the center of the box if no forces are applied which means the readings of the accelerometer in this case will be zero g for each axis (Starlino IMU Guide Webpage).

Each wall of the accelerometer box represents an axis. It is assumed these walls are pressure sensitive. If the box is placed on earth, due to the gravitation force, the ball hits the ground wall as shown in Figure 3.9. In this case the readings of the accelerometer for the Z axis shows -1g and 0g for X and Y axes.

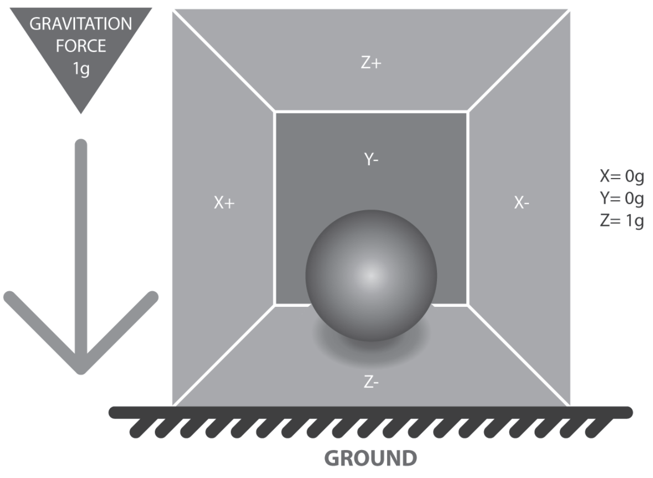


Figure 3.9 Gravitation force effect on the ball.

Previously a single axis accelerometer output is analyzed but in case triaxial accelerometer is used it is possible to detect inertial forces on three walls at the same time. Assume that the box is placed on earth again but this time with an angle of 45 degrees relative to the ground surface. In this case the ball will touch 2 walls as in Figure 3.10.

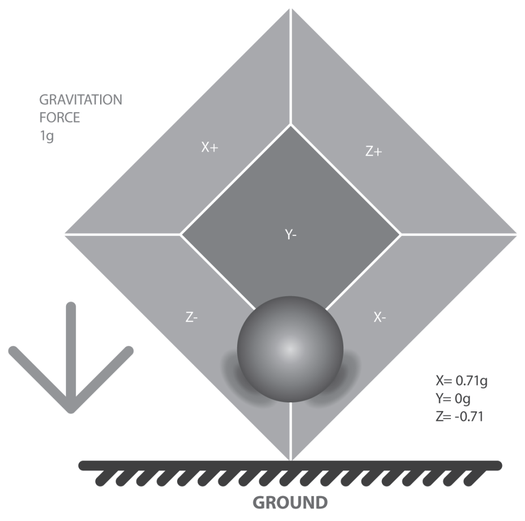


Figure 3.10 Gravitation force effecting on two axes.

The readings of the accelerometer will be -0.71g for X and Z axes and 0g for Y axis. The magnitude on the walls reduces because the pressure the ball puts on the walls are shared on two walls.

It is possible to use vectorial representation of the imaginary box model as shown in Figure 3.11.

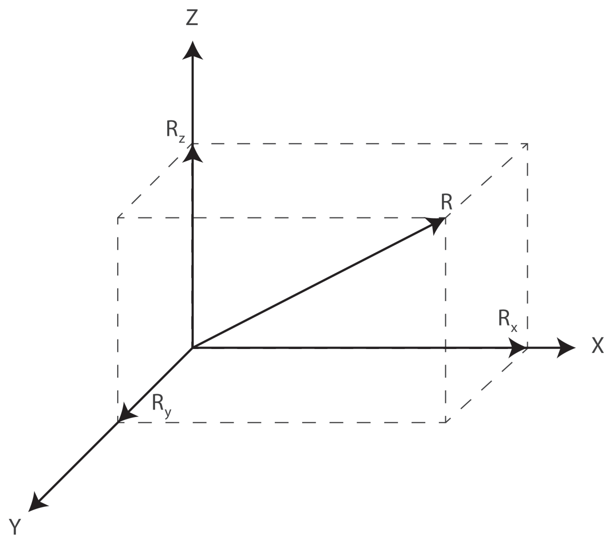


Figure 3.11 R vector representing inertial forces affecting on the cube model.

The shown vector R is the measured acceleration force and R_x, R_y, R_z are projections of the R vector on X, Y, Z planes. By using 3D Pythagorean theorem, the following equation can be easily obtained;

$$R^2 = R_x^2 + R_y^2 + R_z^2 \qquad (3.1)$$

These three vectorial values are actually the readings of the accelerometer. Accelerometers calculate inertial forces on each axis and return them as analog values to be used in microcontroller unit. Since the used IMU board is analog, the output of the accelerometer will be in Volts and they need to be converted into digital values in order to be used in the system. For this purpose ADCs (analog digital converter) are used. On Arduino UNO there are ADCs with 10-bit resolution. That means the converted values will be between 0 and 1023.

To be able to make all these calculations in one step (especially for easy calculation in programming) we may use the following formula (Starlino IMU Guide Webpage);

$$R_{x,y,z} = \frac{\left(\frac{ADC_{R_{x,y,z}}\ V_{ref}}{ADC_{resolution}} - Volts_{ZeroG}\right)}{Sensitivity} \left[\frac{m}{s^2}\right] \qquad (3.2)$$

By calculating R_x, R_y, R_z inertial force vectors, if there are no other forces other than the gravitation, it can be assumed that the obtained vector is the gravitational force vector, which can be used to acquire the inclination angle of the robot.

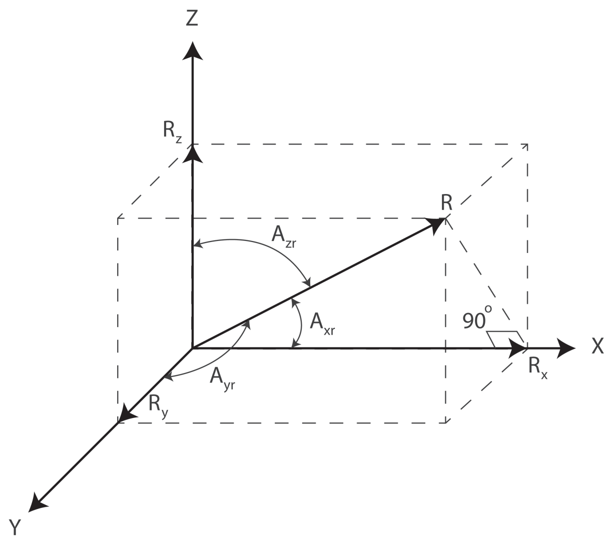


Figure 3.12 Angles between R vector and axes gives the leaning of the robot.

Figure 3.12 shows angles between gravity vector and axes. These angles are the leaning angles of the robot and they can be calculated using the following formulas (Starlino IMU Guide Webpage);

$$A_{xr} = cos^{-1}\frac{R_x}{R} \tag{3.3}$$

$$A_{yr} = cos^{-1}\frac{R_y}{R} \tag{3.4}$$

$$A_{zr} = cos^{-1}\frac{R_z}{R} \tag{3.5}$$

In this project, accelerometer's X axis is the driving direction and Z axis is the upward direction of the robot. Although the accelerometer can calculate gravitation force for three axes, Y axis readings are not evaluated in programming stage since it gives information on the leaning of the robot to the left or right while the robot is balanced by rotating itself around Y axis using information on the leaning of the robot about X and Z axes. Hence, the inclination angle of the robot can be calculated from $A_{zr} = tan^{-1}\frac{R_x}{R_z}$.

3.2.4.2 Gyroscope

Gyroscope working principles can be easily introduced using a similar model that has been used for accelerometer. In Figure 3.13 this similar representation is shown and gyroscope's specifications are shown in Table 3.4 (IDG500 Datasheet).

Table 3.4 Specifications of IDG500 gyroscope.
(IDG500 Datasheet)

Full-scale Range	$\pm 500\ deg/s$
Sensitivity	$2.0\ \frac{mV}{deg/s}$
Zero-G voltage ($Volts_{zeroG}$)	$1.35\ Volts$
Reference voltage (V_{ref})	3.3 V
Power Supply (V_{DD})	$3 \mp 0.3\ Volts$

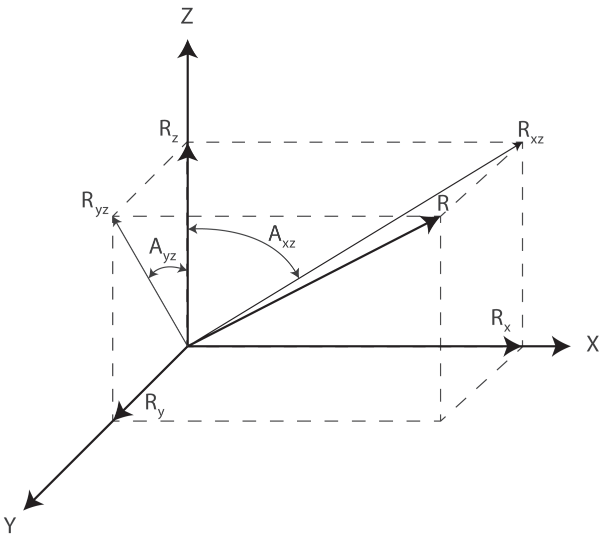


Figure 3.13 Representative model for gyroscope.

Gyroscope measures the rotation rates around the axes. For instance in Figure 3.13 rate of changes of A_{xz} and A_{yz} angles are returned (rate of A_{xy} angle is not returned since the gyroscope used in the IMU is dual-axis) from the sensor in analog format which then needs to be converted to digital format by ADC. Same as accelerometer sensor, in order to get gyroscope's measurements in $\frac{deg}{s}$ units, the following ADC convertion formula needs to be used to the readings of the sensor (Starlino IMU Guide Webpage).

$$Rate_{A_{xz}} = \frac{\left(\frac{ADC_{Gyro_{xz}} V_{ref}}{ADC_{Resolution}} - Volts_{zeroG}\right)}{Sensitivity} \left[\frac{deg}{s}\right] \tag{3.6}$$

3.3 Electrical Connections

All the parts that have been used in this project send or receive data from each other. For instance, motor driver receive information data about speed and direction from Arduino UNO for each DC motor and it sends output data to DC motors according to the received information. Also Acc-Gyro board sends the calculated readings to Arduino board. Since Arduino is the board carrying microcontroller unit, it receives inputs from the sensors, evaluates them and sends an output as a result. It is possible to call it the brain of the balancing robot.

In order to make connections between the parts, the following wirings are used. Boards shown in Figure 3.14 are as follows;

- Arduino UNO
- DC Motor Driver
- Acc-Gyro Board

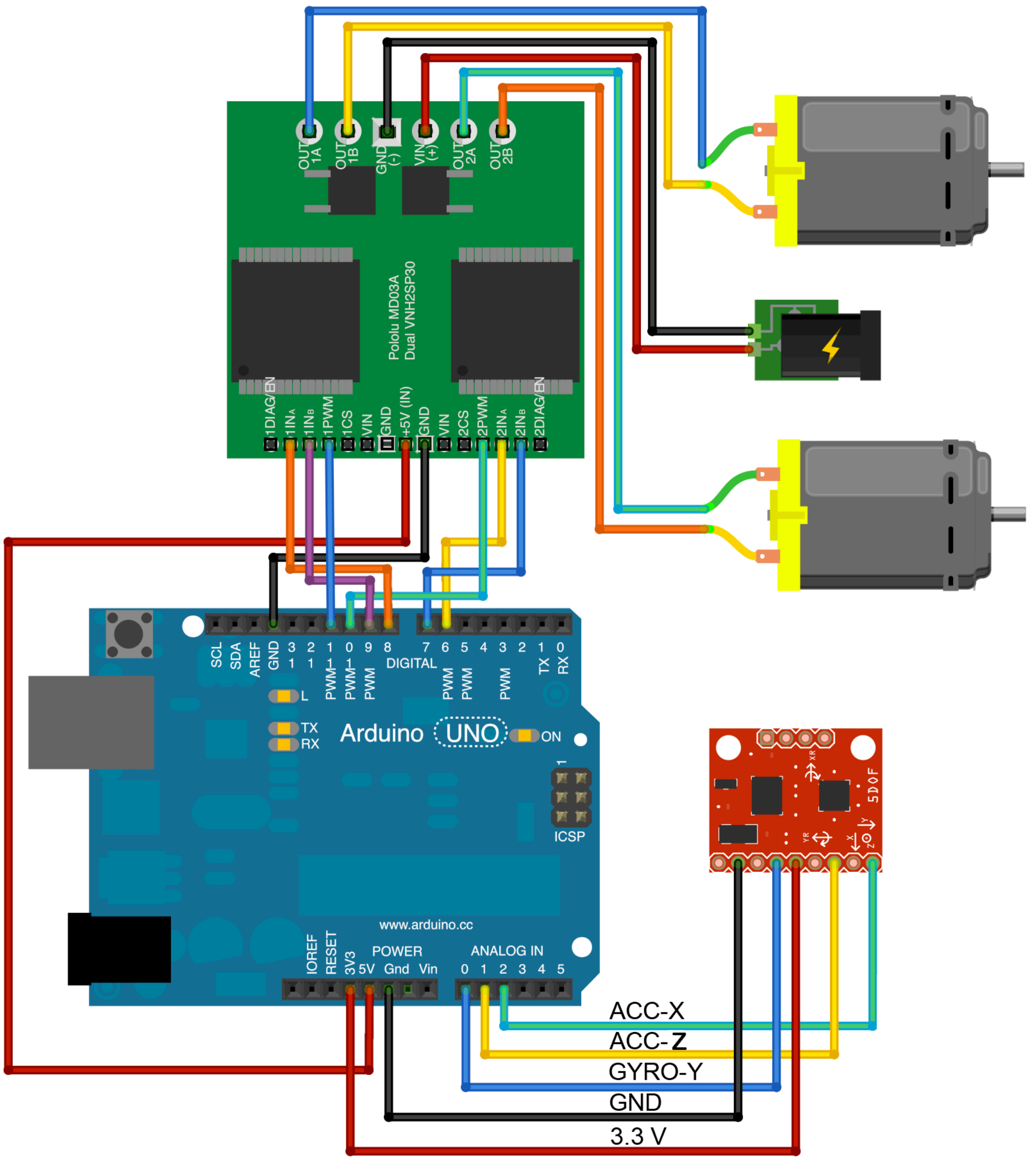


Figure 3.14 Connections between parts of the balancing robot.

The software used to draw wirings is called Fritzing (Fritzing Webpage). It is an open-source application, which can also be used for obtaining schematic and PCB drawings of the components.

3.4 Implementation of Control Algorithms

3.4.1 Sensor Fusion

Sensor Fusion is a way to integrate two or more separate sensor data to obtain a more accurate and realistic data (Elmenreich, 2002). In this thesis two sensors are used. These are accelerometer and gyroscope sensors, which are located together on the Combo IMU Board. These sensors' data are used to obtain the inclination angle of the robot. Since inclination is the only source that is used to balance this robot, it is necessary to be sure of its accuracy. The main problem of accelerometers is that they are easily effected from vibration. In this system accelerometer is placed near the center of axis of rotation where two dc motors are also mounted. That is why the sensing data acquired from the accelerometer will include a lot of noise. In order to reduce this noise, gyroscope data is integrated with the accelerometer data using Kalman filtering method (Åström, 2002). Unlike accelerometer, gyroscopes are reliable in sense of vibration. They don't get effected from the vibration but their problem is that they drift from their initial reference in time. To overcome these sensor problems, measurements of the accelerometer and gyroscope are used together using Kalman filtering method described in the following section.

3.4.1.1. Kalman Filter

Kalman filter is a way to estimate past, present and future states of a system with a set of equations using least squares method. In order to use this filter on a system, first, it should be able to linearized and written in state space form.

A linear system can be mathematically expressed with a state equation (3.7) and a measurement equation (3.8).

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{3.7}$$

$$z_k = Hx_k + v_k \tag{3.8}$$

In these two equations k is used as the index to determine the instant of sampling time, $A(\text{state matrix}), B(control\ input)\ and\ H(measurement\ state)$ are matrices, u is a known input and x_k is the vector that represents present state of the system. w_k and v_k are process and measurement noises, respectively. Sensor measurements that include v_k measurement noise is represented with z_k.

In order to use Kalman filter in a system these two requirements should be met:

1. The noises should be white with normal probability.
2. Noises should be independent from each other. So there should be no correlation between them.

Defining $\hat{x}_k^-$ as priori state estimate at time k and $\hat{x}_k$ as posteriori state estimate at time k, estimate errors can be written as

$$e_k^- = x_k - \hat{x}_k^- \tag{3.9}$$
$$e_k = x_k - \hat{x}_k \tag{3.10}$$

which make the priori and posteriori estimate error coveariances as

$$P_k^- = E[e_k^- e_k^{-T}] \tag{3.11}$$
$$P_k = E[e_k e_k^T] \tag{3.12}$$

Kalman Filtering equations can be separated into two parts. One part is time update equations and the other one is measurement update equations. These equations are recursive and they complete each other. Kalman recursive behaviour between these groups is shown in Figure 3.15.

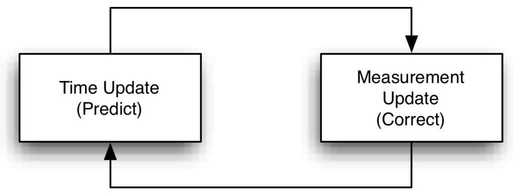


Figure 3.15 Kalman Filtering recursive behaviour.

Time update equations take initial estimation inputs (posteriori state estimation $\hat{x}_{k-1}$ and error covariance P_{k-1}) and make a prediction for the next step in time as follows:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \qquad (3.13)$$

$$P_k^- = AP_{k-1}A^T + Q \qquad (3.14)$$

Using equation (3.13) and (3.14), priori state estimation and priori error covariance are obtained, where Q is defined as process noise covariance.

After time update equations are applied to the initial estimation inputs and predictions are made for the next step, these predictions are corrected using measurement update equations.

To find the posteriori (corrected) state estimate $\hat{x}_k$. The following equation is used:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \qquad (3.15)$$

where K_k is known as blending factor and the difference $(z_k - H\hat{x}_k^-)$ is called measurement innovation. The blending factor is given as

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R} \qquad (3.16)$$

where R is the measurement noise covariance matrix.

Blending factor determines the trust degree of the Kalman filter. If the R measurement noise value approaches zero, blending factor K_k weights the measurement innovation heavily, which also means that the sensor measurements are trusted more and more as blending factor approaches zero. On the other hand if R measurement noise gets bigger, sensor measurements are trusted less.

As a last step of measurement update stage, P_k error covariance is corrected using equation (3.17).

$$P_k = (1 - K_k H)P_k^- \qquad (3.17)$$

The whole recursive system with already given equations is shown clearly in Figure 3.16.

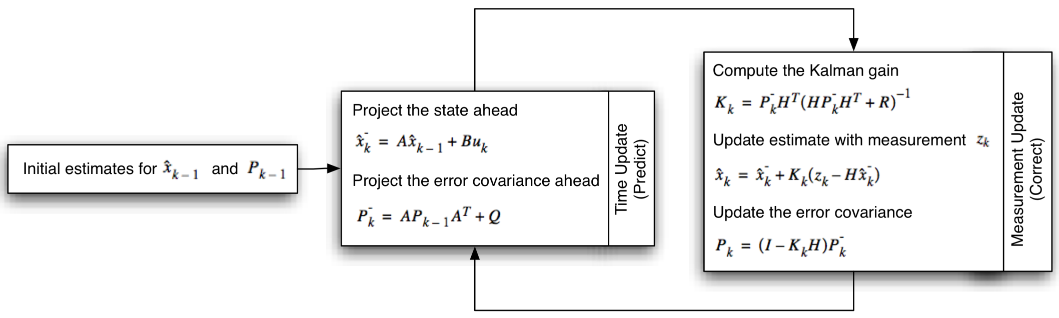

Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}
Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$
Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Time Update (Predict)
Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
Update the error covariance
$P_k = (I - K_k H)P_k^-$
Measurement Update (Correct)

Figure 3.16 Kalman Filtering equations shown in two groups.

Since Kalman filtering is recursive, corrected estimations become new previous values to be used for calculating future estimations.

In order to show the effects of Kalman filtering, a test platform is set and raw data of the accelerometer and output of the Kalman filtering process are shown together in Figure 3.17.

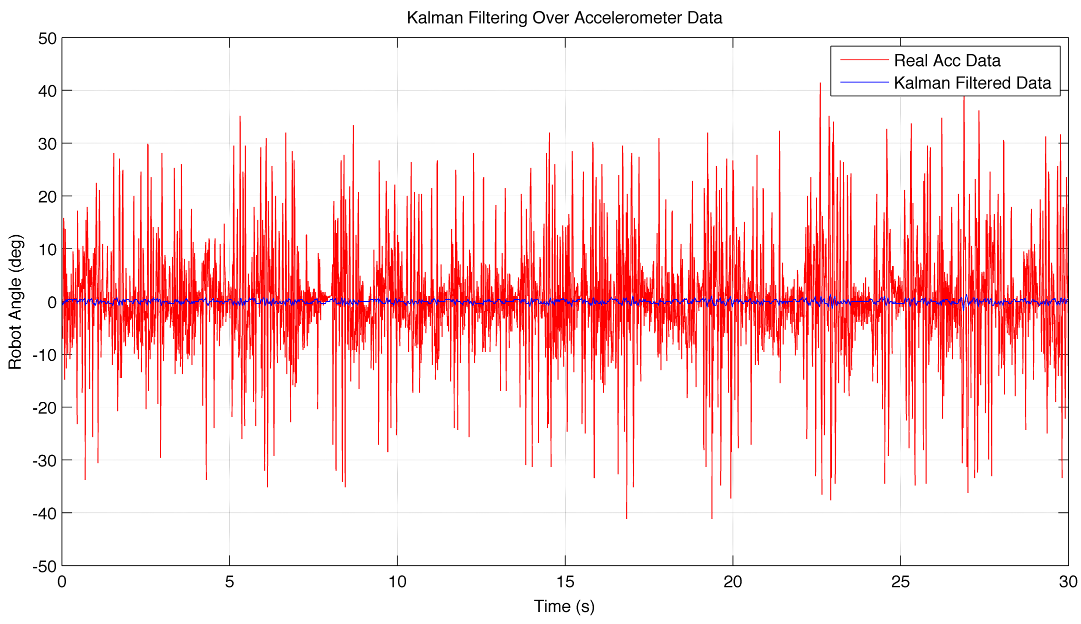


Figure 3.17 Accelerometer raw data and Kalman filtering implementation on the data

As it can be easily seen from the Figure, raw accelerometer data noise is significantly reduced with Kalman filter implementation. The output data have neither vibration nor drifting problems. Without filtering process it would be very hard to balance the robot using the raw data but after the filtering, a smooth data is acquired which can be used for balancing.

Since matrix calculation in microcontrollers consume a lot of processing time, Kalman filter's matrix equations are implemented in algebraic form in programming (Yong & Kwong, 2011; Baik, 2008) . In Figure 3.18, implementation of Kalman filter in software is given as a flowchart.

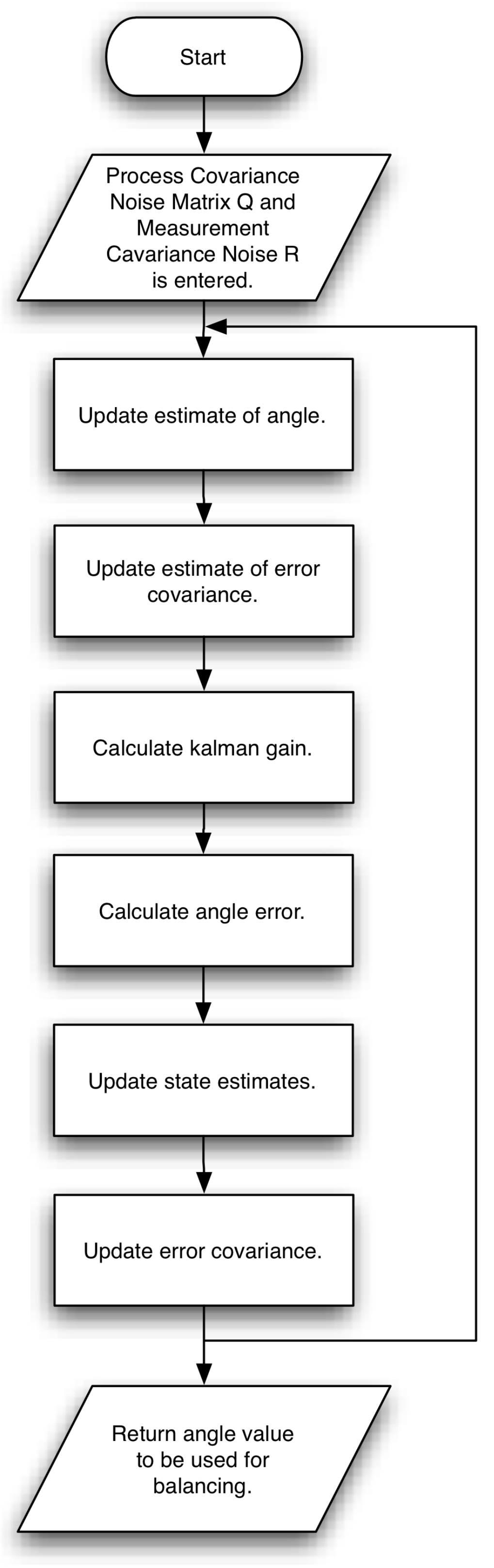

Start
Process Covariance Noise Matrix Q and Measurement Cavariance Noise R is entered.
Update estimate of angle.
Update estimate of error covariance.
Calculate kalman gain.
Calculate angle error.
Update state estimates.
Update error covariance.
Return angle value to be used for balancing.

Figure 3.18 Kalman Filter flowchart.

3.4.2 PID Control

The Proportional Integral Derivative (PID) controller is the most common and widely used generic feedback mechanism. After the emerge of process control in 1940s, PID controllers are started to be used as standart tools in control and they are still the most widely used controllers in the industry (Åström, 2002).

The main working principle of a PID controller is simple; it calculates the error between the desired setpoint and the measured process output and tries to minimize this error by changing P, I and D process control parameters. In this balancing robot thesis, the desired setpoint is set as the upright position of the robot. So PID controller will try to return the robot to the upright position if it somehow with any effects leans in any direction

Control error e is given in equation (3.18) as the difference between reference setpoint (upright position) and measured process variable (measured leaning of the robot).

$$e_{(t)} = y_{SP(t)} - y_{(t)} \qquad (3.18)$$

PID controller algorithm is embedded in the Arduino microcontroller. The processing unit of the Arduino is able to calculate the error approximately ten times in a second and sends PWM signal to motor drivers to adjust the robot to balance itself.

PID algorithm is given in equation (3.19) and (3.20) (Åström, 2002).

$$u_{(t)} = K\left(e_{(t)} + \frac{1}{T_i}\int_0^t e_{(\tau)}d\tau + T_d\frac{de_{(t)}}{dt}\right) \qquad (3.19)$$

$$u_{(t)} = \underbrace{Ke_{(t)}}_{P-term} + \underbrace{\frac{K}{T_i}\int_0^t e_{(\tau)}d\tau}_{I-term} + \underbrace{KT_d\frac{de_{(t)}}{dt}}_{D-term} \qquad (3.20)$$

where y is the measured process variable, e is the control error and u is the control signal. Controller parameters are given as K proportional gain, T_i integral time constant and T_d derivative time constant.

Control signal u has three separate constant parameters; P, I and D. These parameters can be interpreted as past, present and future; P term is proportional to the present error, I term is proportional to the accumulation of past errors and D term is the prediction of future errors. Control signal is calculated by the sum of these P, I, D parameters which can be seen in Figure 3.19, which is constructed in MATLAB Simulink.

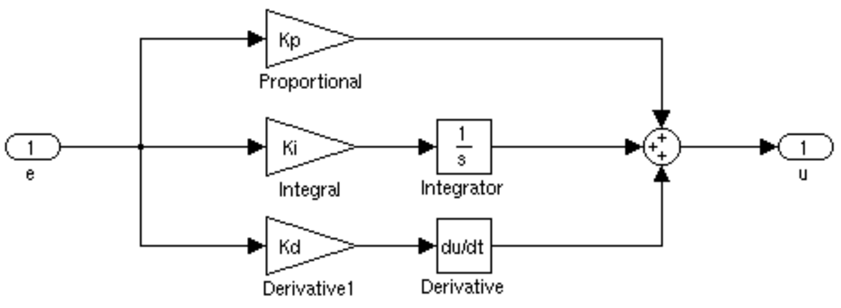


Figure 3.19 PID Controller

3.4.2.1 Effects of PID Controller Parameters

Although PID control is pretty straightforward, since P, I and D terms are depended on each other, effects of each parameter on other parameters should be put into consideration. Each term and their effects on the system will be investigated for a better understanding of PID control (Åström, 2002).

For the examination of the terms, the following process transfer function is taken as an example and the setpoint is set as 1.

$$G_{(s)} = \frac{1}{(s+1)^3} \quad (3.21)$$

To understand proportional term's behavior on the given process, $T_i = \infty$ and $T_d = 0$ assumptions are made and K parameter's value is changed. The simulation results for each K parameter are shown in Figure 3.20.

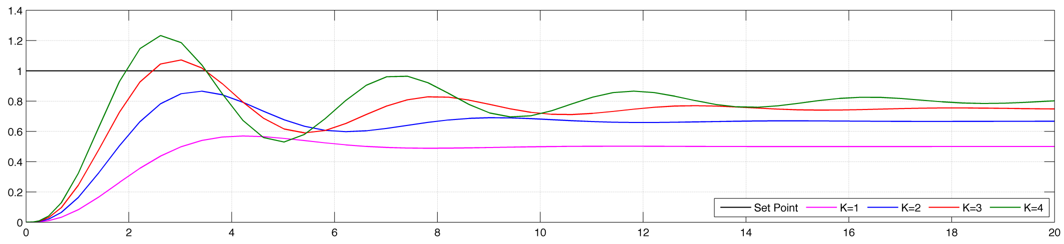


Figure 3.20 Proportional control of the process with various K values.

As it can be easily observed from the simulation results, when K gain is increased, $y_{(t)}$ measured control variable is getting closer to the setpoint value which means $e_{(t)}$ control error is getting smaller but at the same time oscillation increases when K gain is increased. It is also important to see that when just proportional term is used during control, there will always be a steady state error. If elimination of steady state error is desired, integral term has to be used.

Integral term's effects on the process given in equation 3.21 are shown in Figure 3.21.

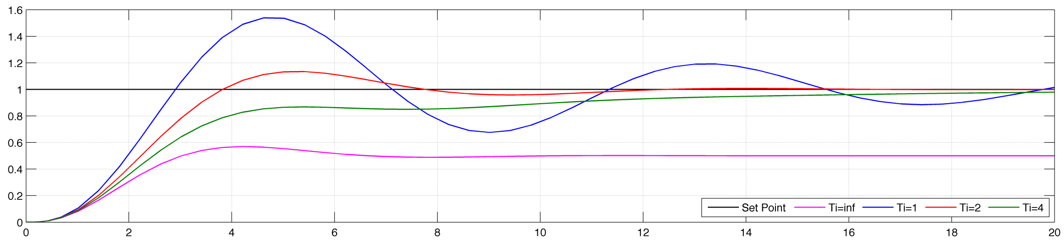


Figure 3.21 Proportional - Integral (PI) Control simulation results with various T_i parameters. (Controller gain K = 1)

The effect of integral term increases when integral time T_i decreases. It is also clear from the simulation results that when integral term is used steady state error disappears. Oscillation tendency and settling time are also increased when integral term is used together with the proportional controller.

Effects of derivative term can be seen in Figure 3.22.

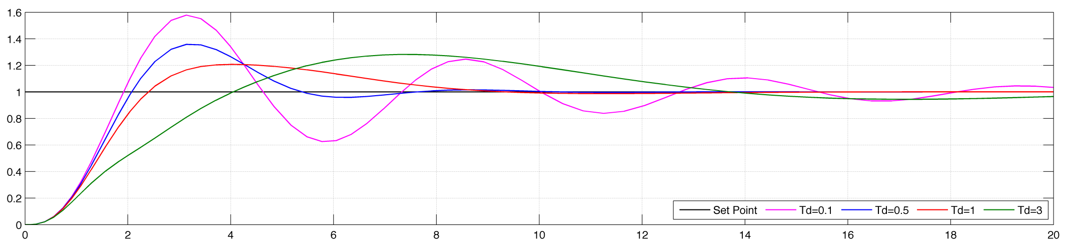


Figure 3.22 Proportional - Integral - Derivative (PID) Control simulation results with various T_d parameters. (An oscillatory system with controller gain K = 3 and integral time $T_i = 2$ is chosen)

Derivative term has a damping effect on the system. As T_d derivative time increases, it's effect increases and it reduces the overshoot caused by proportional and integral terms and it also reduces the settling time of the system. But as the derivative time gets too large, damping effect starts to decrease.

Close relations between PID control parameters can make the tuning process quite hard, especially in complex and nonlinear systems like a balancing robot system. To find control parameters of the PID controller, Ziegler - Nichols frequency response method is experienced using Table 3.5 (Åström, 2002) but since this is not an optimal method for all systems, it is not used in the balancing robot system. Instead of using this method, Simulink Design Optimization Toolbox is used. In the next section this method will be explained in detail.

Table 3.5 Ziegler - Nichols frequency response method controller parameters (Åström, 2002)

Controller	K	T_i	T_d
P	$0.5K_u$	-	-
PI	$0.4K_u$	$0.8T_u$	-
PID	$0.6K_u$	$0.5T_u$	$0.125T_u$

After calculating the PID control parameters using design optimization toolbox, these parameters are used to calculate control output in microcontroller as shown in the flowchart in Figure 3.23.

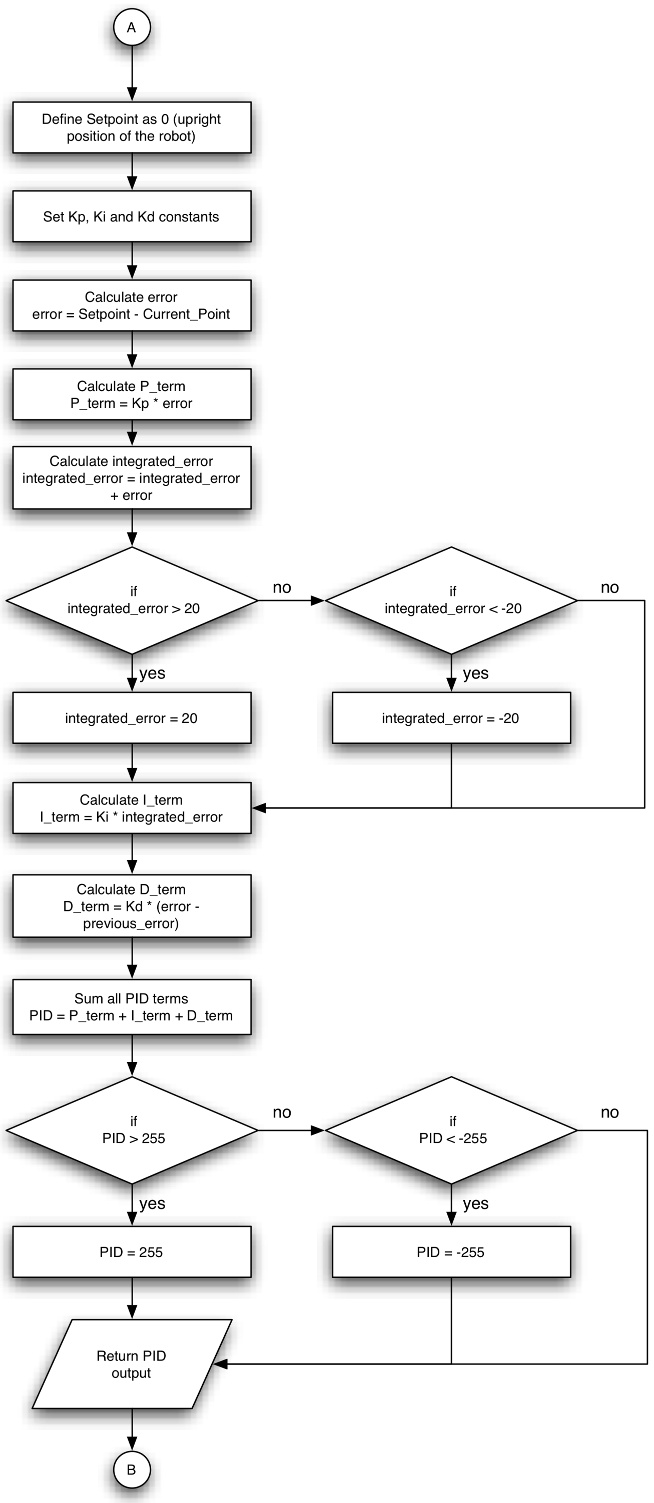

A
Define Setpoint as 0 (upright position of the robot)
Set Kp, Ki and Kd constants
Calculate error error = Setpoint - Current_Point
Calculate P_term P_term = Kp * error
Calculate integrated_error integrated_error = integrated_error + error
if integrated_error > 20
no
if integrated_error < -20
no
yes
yes
integrated_error = 20
integrated_error = -20
Calculate I_term I_term = Ki * integrated_error
Calculate D_term D_term = Kd * (error - previous_error)
Sum all PID terms PID = P_term + I_term + D_term
if PID > 255
no
if PID < -255
no
yes
yes
PID = 255
PID = -255
Return PID output
B

Figure 3.23 PID control algorithm flowchart.

CHAPTER FOUR
COMPARISON OF SIMULATION AND EXPERIMENTAL RESULTS

Simulation is imitating real world processes using softwares, which are specialized for this purpose. Making a simulation of a process is a necessary stage, especially for the projects that have complex and unpredictable behaviors. Simulations make it possible to virtualize how the process is going to behave in time and the effects of changing system parameters on the system can be easily observed. The approximation of the simulations to the real world processes are proportional to the details integrated into the simulations which means the more real world conditions are integrated, the more realistic simulations are acquired.

In this thesis, Matlab numerical computer environment is used. The whole nonlinear equations of the balancing robot system are constructed using Simulink. The main reason for using Simulink was that it provides model based design using elementary blocks for dynamic systems and makes observation of the whole system much easier when compared to code based simulations.

While constructing the model of the balancing robot, both non-linear equations given in (2.45) and (2.46) and linearized equations given in (2.53) and (2.54) were used. Linearized equations, as explained in Section 2.4, are useful when the angle of the pendulum is very close to the linearization point. But, if the pendulum angle gets bigger, linearized equations start to give results with error. This error increases exponentially when the pendulum angle gets bigger. This situation is shown in Figure 4.1, which gives the response of the pendulum angle after an impulse input. As can be seen clearly, when the pendulum angle is calculated using linearized equations, after pendulum starts to get away from the upright position, calculated angle value changes rapidly and goes to infinity.

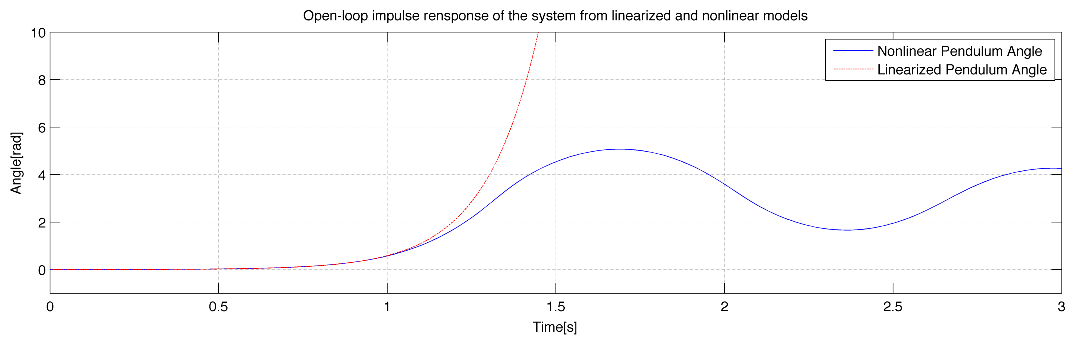


Figure 4.1 Open-loop response of linear and nonlinear models to $V_a = 1V$ impulse input.

In this thesis, non-linear equations are used to observe the behavior of the system, because there is always an error when linearized equations are used (even for small angles) and when linear models are used it is not possible to see the whole behavior of the system.

4.1 System Modeling and Parameters

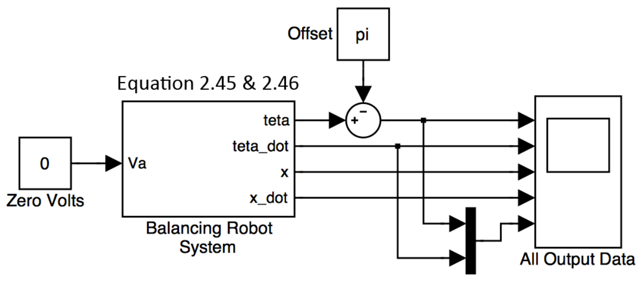


Figure 4.2 Open-loop model of the balancing robot.

First, an open-loop model shown in Figure 4.2 is constructed using Simulink. Balancing robot block consist of nonlinear equations that are given in equations (2.45) and (2.46). It has one input and four states which are also system outputs. So it is a SIMO (single input and multiple output) system. Its input is dc motor armature terminal voltage (V_a) and outputs are pendulum angle (θ_p), pendulum angular velocity $(\dot{\theta}_p)$, distance (x) and velocity $(\dot{x})$. The equations implemented inside the balancing robot system are shown in Figure 4.3. Since upright position of the balancing robot equals to π radians according to the nonlinear equations, to increase the readability of the angle graph, an offset block with a value of π is used in the model, which subtracts that value from the angle. So the zero value that is seen on the scope corresponds to the robot's upright position.

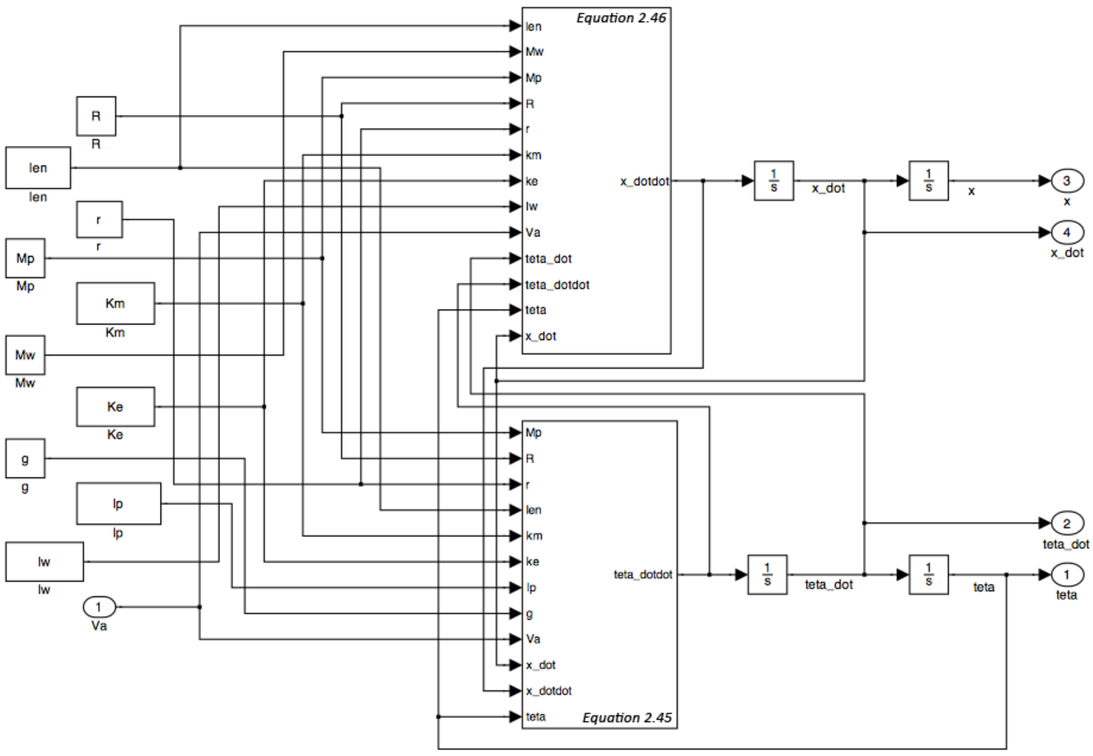


Figure 4.3 Balancing robot system block, which consist of nonlinear system.

In order to simulate the system and see how it behaves, first, it's system parameters should be measured and entered into the simulation. System parameters of the balancing robot which are also given in the non-linear system equations (2.45) and (2.46), are listed in Table 4.1.

Table 4.1 System Parameters

Parameters	Description	Value
R	DC motor armature resistance	3.8 Ohm
r	Radius of the wheel	0.045 m
M_w	Mass of the wheel (including hub)	0.034 kg
M_p	Mass of the pendulum	1.086 kg
K_e	Electrical constant of the motor	0.1344962 $\frac{Vs}{rad}$

Table 4.1 Continues

K_m	Torque constant of the motor	$0.1344962\ \frac{Nm}{A}$
I_p	Inertia of the pendulum	$0.02042542\ \frac{kg}{m^2}$
I_w	Inertia of the wheel	$0.00003061\ \frac{kg}{m^2}$
l	Lenght between the rotational axis of the motors and the COG of the pendulum.	$0.083\ m$
g	Gravity force	$9.81\ \frac{m}{s^2}$

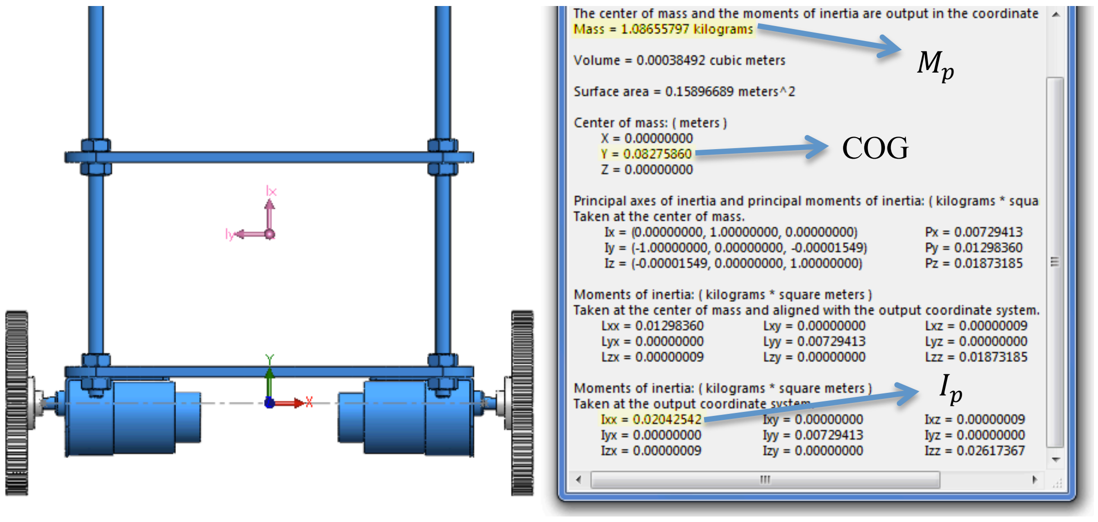
Parameters given in Table 4.1 are obtained using different methods;

DC motor armature resistance (R) is measured with a multimeter. The armature inductance, and as a result electrical transients are neglected in the model. Radius of the wheel (r) and mass of the wheel (M_w) are taken from the manufacturer's web site (Pololu Wheels Webpage). Mass of the pendulum (M_p) is calculated in Solidworks after drawing parts in the program and assigning materials to them matching their mass. All of the parts forming the pendulum are then selected and sum of their mass is calculated. Back emf constant (K_e) and torque constant (K_m) of DC motor are measured experimentally. In order to acquire back emf constant Equation (2.2) is used.

DC motor's shaft is fixed to another motor and its shaft is turned with a constant speed of 142 RPM. During this rotation, terminal voltage is measured with a multimeter as 2 Volts. Since there is no armature current, the terminal voltage (V_a) is equal to back emf (E_a). The calculation of K_e and K_m with conversion ratio of gearbox included, is obtained as;

$$k_e = \frac{E_a}{\dot{\theta_w}} = \frac{2}{142\frac{2\pi}{60}} = 0.1344962\ \frac{Vs}{rad} \qquad (4.1)$$

Inertia of the pendulum (I_p), inertia of the wheel (I_w), length between COG of the pendulum and the rotation axis (l) and mass are calculated automatically in Solidworks' mass properties tool. In Figure 4.4, calculation results for l, I_p and mass of the pendulum (M_p) are shown.



```
The center of mass and the moments of inertia are output in the coordinate
Mass = 1.08655797 kilograms

Volume = 0.00038492 cubic meters

Surface area = 0.15896689 meters^2

Center of mass: ( meters )
	X = 0.00000000
	Y = 0.08275860
	Z = 0.00000000

Principal axes of inertia and principal moments of inertia: ( kilograms * squa
Taken at the center of mass.
	 Ix = (0.00000000, 1.00000000, 0.00000000)	 	Px = 0.00729413
	 Iy = (-1.00000000, 0.00000000, -0.00001549)	 	Py = 0.01298360
	 Iz = (-0.00001549, 0.00000000, 1.00000000)	 	Pz = 0.01873185

Moments of inertia: ( kilograms * square meters )
Taken at the center of mass and aligned with the output coordinate system.
	Lxx = 0.01298360	Lxy = 0.00000000	Lxz = 0.00000009
	Lyx = 0.00000000	Lyy = 0.00729413	Lyz = 0.00000000
	Lzx = 0.00000009	Lzy = 0.00000000	Lzz = 0.01873185

Moments of inertia: ( kilograms * square meters )
Taken at the output coordinate system.
	Ixx = 0.02042542	Ixy = 0.00000000	Ixz = 0.00000009
	Iyx = 0.00000000	Iyy = 0.00729413	Iyz = 0.00000000
	Izx = 0.00000009	Izy = 0.00000000	Izz = 0.02617367
```

Figure 4.4 Calculating the parameters in Solidworks.

In Solidworks, in order to calculate properties of parts, first, they are selected together. Selected parts are shown in blue color. Parts that construct pendulum of the balancing robot are selected and then mass properties tool is opened for Solidworks to calculate properties of the selected parts. To be used in inertia or COG calculations, a reference coordinate system is needed for this tool to function properly. Colored coordinate system shown in Figure 4.4 is selected in Solidworks to be used in calculations.

In the properties window shown in Figure 4.4, calculated total weight of the pendulum is given in kilograms at the top of the properties window. l is given under center of mass category as Y coordinate and I_p is shown under Moments of Inertia: Taken at the output coordinate system category shown as I_{xx}. In order to find the inertia of the wheel (I_w) and mass of the wheel (M_w), wheel and hub is selected and mass properties window is opened for these parameters to be calculated by Solidworks.

After all the parameters are found, these parameters are entered in the Balancing Robot Block shown in Figure 4.3.

When the open-loop system simulation is run for 7 seconds, the variations of outputs ($\theta_p, \dot{\theta_p}, x\ and\ \dot{x}$) with respect to time are shown in Figure 4.5. In this simulation no motor force is used (zero volts is given as input) but a very small initial pendulum angle of 0.01 rad ($0.573\ degrees$) leaning is given from within the balancing robot subblock. When the angle graph is examined, it is obvious that the leaning angle θ_p does not approch to zero (upright position), instead, it oscillates and dampens about 3.14 (π). This means, since there is no feedback, it does not balance and due to the gravity force effecting on the robot it falls down and stops at downright position. In this model, it is considered that the pendulum can freely rotate $360°$ about the rotational axis of the wheels.

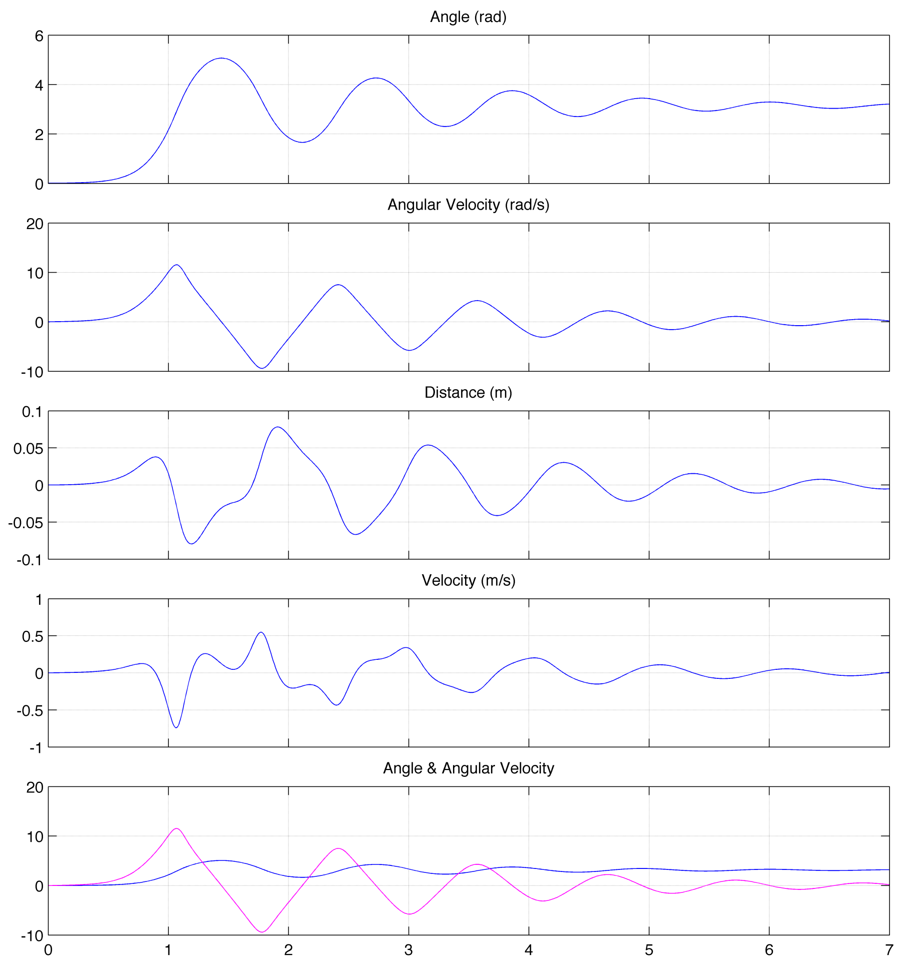


Figure 4.5 Open-loop behavior of the balancing robot.

As stated earlier, unlike linear equations, non-linear equations give correct system status even the pendulum angle is far from the upright position.

4.2 Modeling of Closed-loop Control System

Open-loop controller does not use any feedback in order to correct the control error. It is just a representation of the balancing robot calculated by using the current state of the system. However, since the aim of the balancing robot is to bring the robot to upward position, thus make the control error equals to zero, a PID controller is used in this thesis. Balancing robot system with a PID controller is constructed using Simulink as shown in Figure 4.6. Since the DC motors are controlled by pulse width modulated signals in the real system, the actual terminal voltage applied to the motors is not a pure dc waveform. So, PWM pulse generator block is also included in the model shown in Figure 4.6.

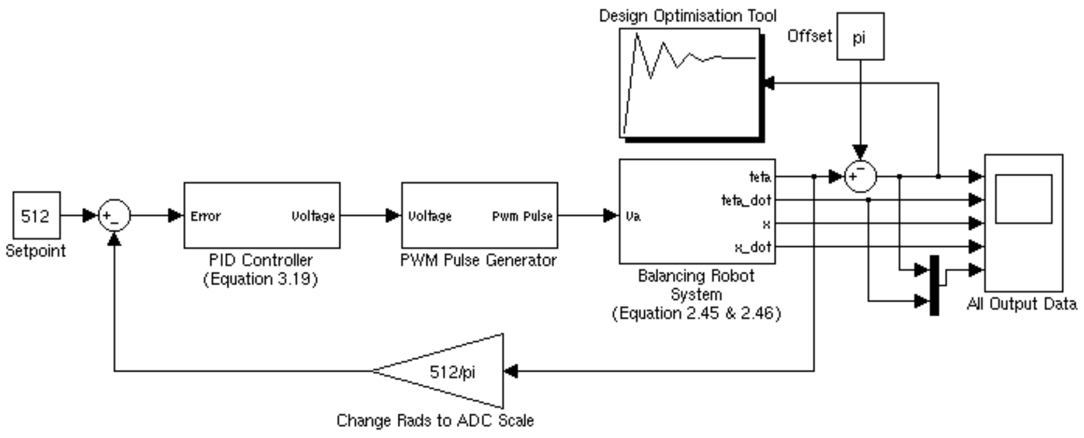


Figure 4.6 PID controlled balancing robot process.

As explained earlier in Section 3.4.2, PID controller tries to eliminate the control error using P, I, D controller parameters. This control error is calculated by taking the difference between the reference setpoint and the angle feedback. Setpoint block, which is set to 512, indicates upward position of the robot in terms of converted number scale of ADC, which is between 0 and 1024. In order to covert the radian to ADC number scale, pendulum angle feedback is multiplied by a gain block $(2\pi\ [rad] = 1024)$. The voltage to be send to the motor driver is calculated using

PID algorithm in the microcontroller and then it is send to the motor driver using PWM (Pulse Width Modulation).

In order to simulate PWM pulses, PWM Pulse Creator block is placed after the controller. This block gets the analog voltage output as an input and transforms this value to digital 0, +12 or -12 Volts with a frequency of 3921.56 Hz. The inner structure of the PWM Pulse Generator block is shown in Figure 4.7.

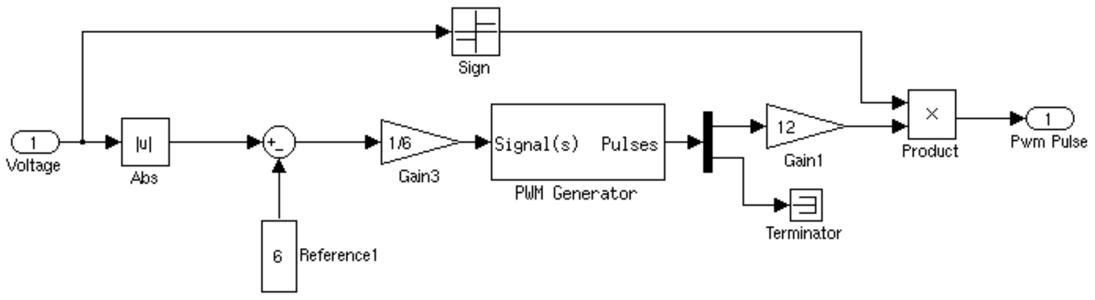


Figure 4.7 Inside of the PWM Pulse Generator Block.

In order to control the system with a PID controller, first, parameters of the controller should be obtained. Design Optimization Tool of Matlab is used to obtain them. K_P, K_I and K_D parameters are estimated with numerical optimization by numerous simulations using the Simulink model. In order to use this tool, several conditions need to be entered into the system. So, optimization tool tries to find a feasible solution, which satisfies given conditions.

For this system rise time is required to be less than 0.5 s, overshoot is to be at most 0.03 rad, settling time is to be at most 3 s and Steady-State error is to be 0.01 rad. These conditions are entered into Design Optimization Tool as shown in Figure 4.8.

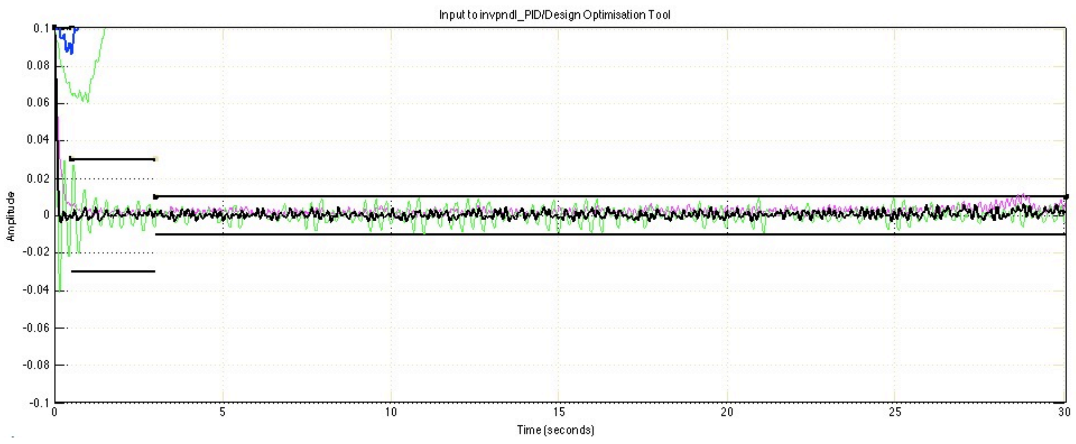
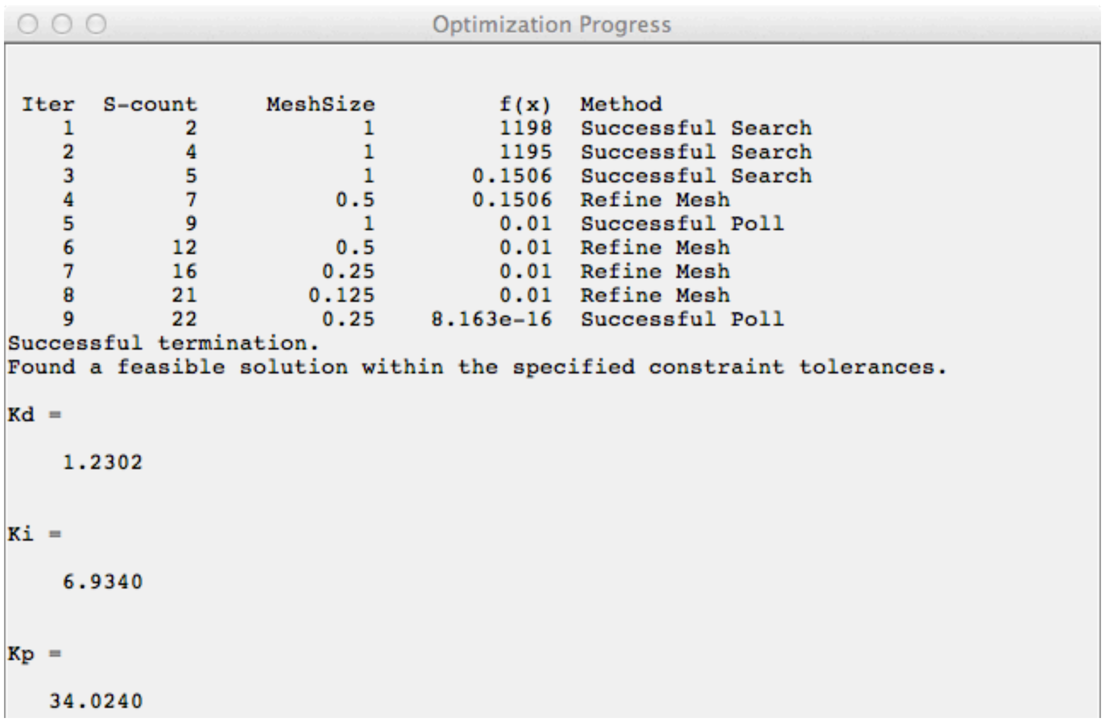


Figure 4.8 Tuning of PID parameters using design optimization tool.

When optimization is run for a 30 second long simulation using Pattern Search optimization method, after several unsuccessful attempts, Matlab found a feasible solution (shown with black signal in Figure 4.8) as $K_p = 34.0240, K_i = 6.9340\ and\ K_d = 1.2302$. Optimization result screen is shown in Figure 4.9.



```
 Iter   S-count     MeshSize         f(x)  Method
    1         2            1         1198  Successful Search
    2         4            1         1195  Successful Search
    3         5            1       0.1506  Successful Search
    4         7          0.5       0.1506  Refine Mesh
    5         9            1         0.01  Successful Poll
    6        12          0.5         0.01  Refine Mesh
    7        16         0.25         0.01  Refine Mesh
    8        21        0.125         0.01  Refine Mesh
    9        22         0.25    8.163e-16  Successful Poll
Successful termination.
Found a feasible solution within the specified constraint tolerances.

Kd =

    1.2302


Ki =

    6.9340


Kp =

   34.0240
```

Figure 4.9 Design Optimization Result Screen.

4.3 Real System Implementation and Control

Arduino microcontroller's default PWM frequency is about 490 Hz (Arduino PWM Frequency Webpage) but it is increased to 3921.56 Hz using timers of the microcontroller. The details of the calculation of the PWM frequency is given in Appendix. Balancing robot moves according to the given motor force and tries to balance itself. It's angle is fed as feedback to be used to calculate the new error and this cycle goes on over and over again in an infinite loop to balance the robot. The flowchart of this infinite loop of the whole balancing robot system is given in 4.10.

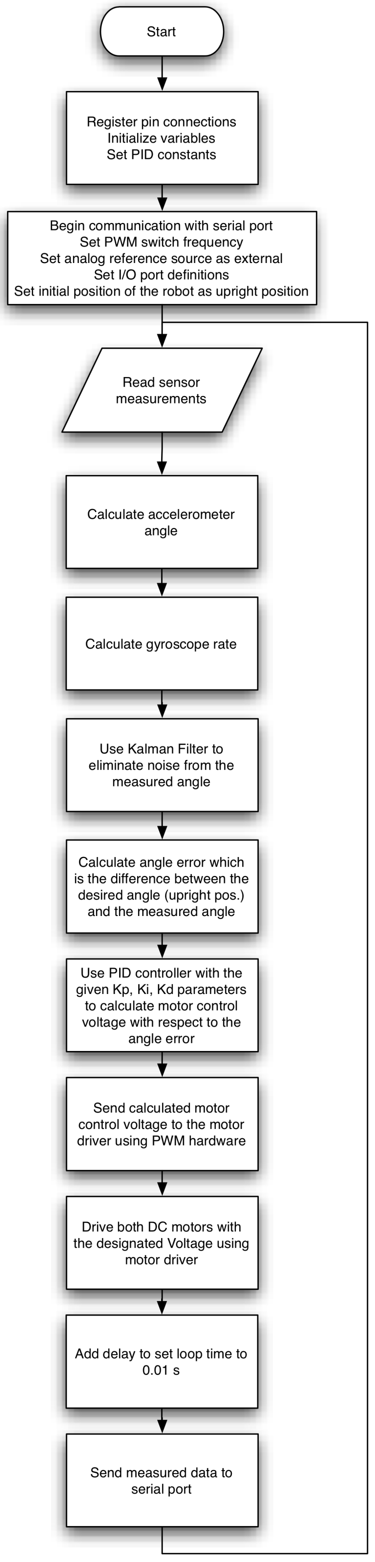

Start
Register pin connections
Initialize variables
Set PID constants
Begin communication with serial port
Set PWM switch frequency
Set analog reference source as external
Set I/O port definitions
Set initial position of the robot as upright position
Read sensor measurements
Calculate accelerometer angle
Calculate gyroscope rate
Use Kalman Filter to eliminate noise from the measured angle
Calculate angle error which is the difference between the desired angle (upright pos.) and the measured angle
Use PID controller with the given Kp, Ki, Kd parameters to calculate motor control voltage with respect to the angle error
Send calculated motor control voltage to the motor driver using PWM hardware
Drive both DC motors with the designated Voltage using motor driver
Add delay to set loop time to 0.01 s
Send measured data to serial port

Figure 4.10 Flowchart of the complete balancing robot system.

When the simulation is run for 10 seconds with the found PID parameters, the following scope results were obtained as shown in Figure 4.11.

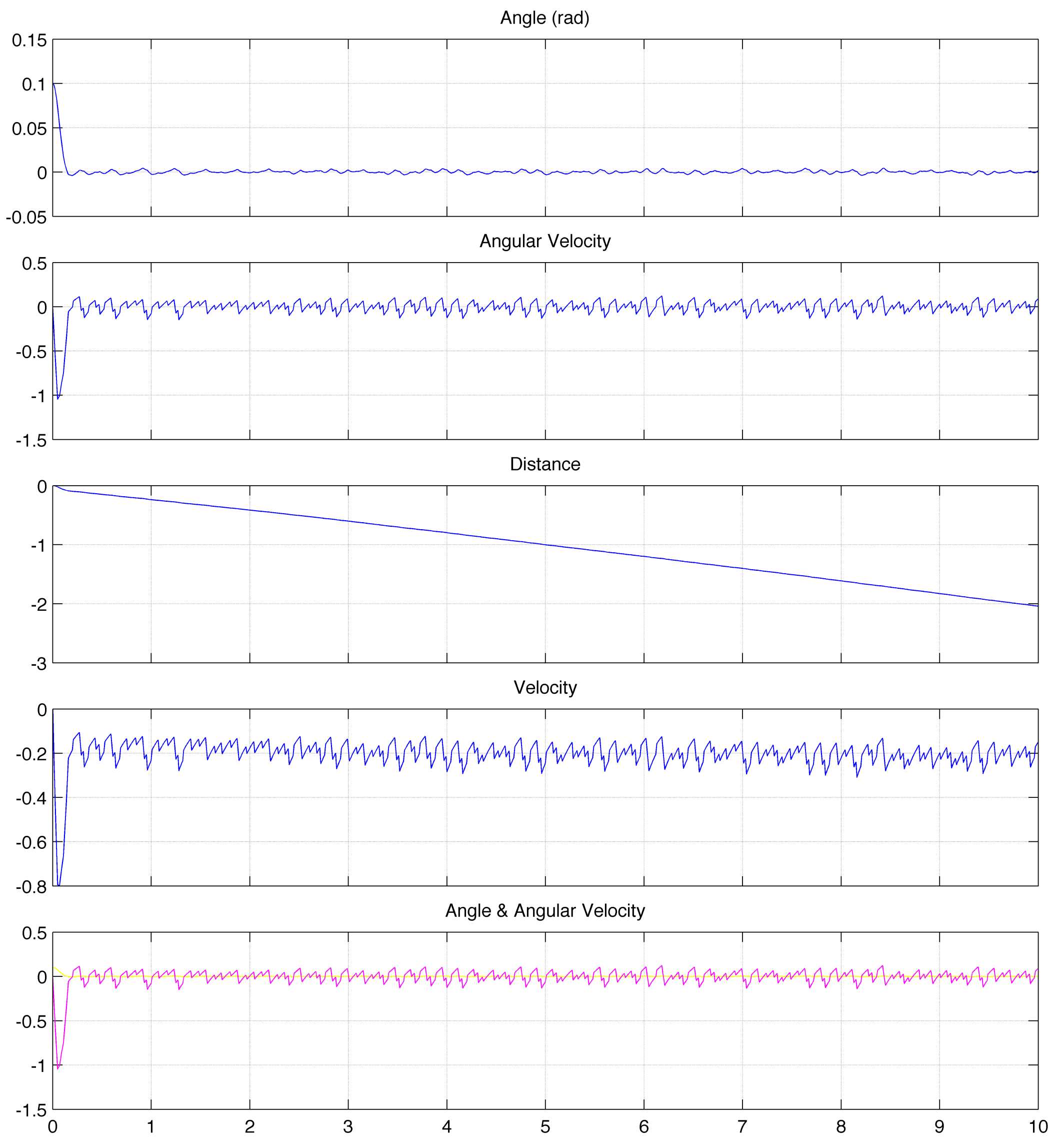


Figure 4.11 PID controlled balancing robot behavior.

As it can be seen from the figure 4.11, an initial angle of 0.1 rad is given to test the balancing ability of the robot and with the PID parameters, which are found using Design Optimization Tool, balancing is successfully accomplished.

After finding a desired solution for the balancing problem with the help of the simulations, the PID parameters obtained from simulations have been tested on the real robot. In the Figure 4.12, the pendulum angle variation is given. In this figure both filtered and unfiltered data are also shown to show the effects of Kalman filter. Since kalman filtered data shown in Figure 4.12 is not easy to read because of the scale, a zoomed view of this data is given in Figure 4.13.

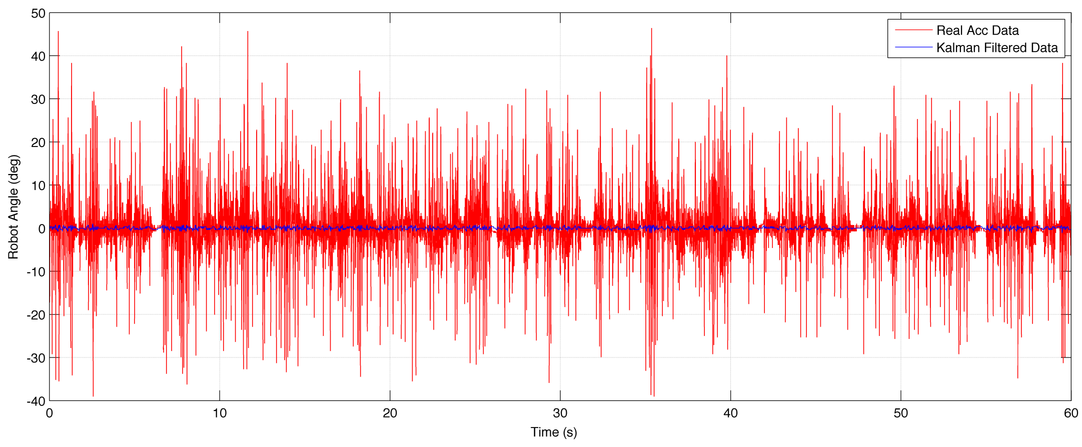


Figure 4.12 PID controlled balancing robot’s angle variation in time (experimental results).

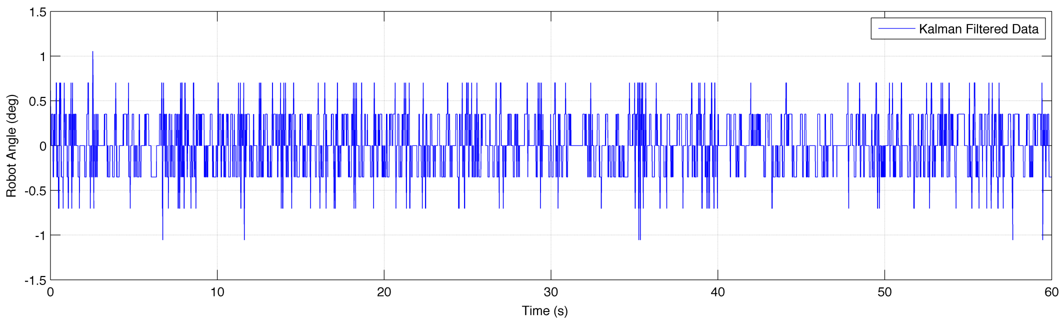


Figure 4.13 Zoomed view of Figure 4.12 (just Kalman filtered angle is shown).

As it can be seen in Figure 4.12, accelerometer data is effected from environment a lot. Kalman filtering is used to acquire realistic data using accelerometer and gyroscope sensor readings together. As a result, Kalman filtered angle data, which shows the leaning of the robot, oscillates around upright position with a small angle error. In Figure 4.13, it can be seen that this error is around ∓ 0.5 degree.

While simulating the system, balancing robot system block angle output always gives noiseless signal. In order to simulate measurement noise of the accelerometer, a Noise Generator block with an average noise variance of 0.1 rad is added to the angle feedback in Simulink to test if the system would stay stable if Kalman filtering is not applied. In Figure 4.14, a noise signal with a variance of 0.1 rad is shown.

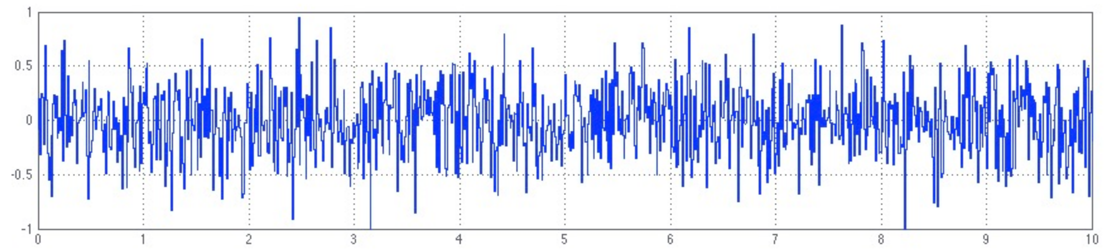


Figure 4.14 Signal noise added to the angle feedback in order to simulate effect of noisy feedback signal on system stability.

When the simulation is run, the robot could not balance itself and fall off to downright position just like an openloop system as shown in Figure 4.15. As a result, it is proved that without implementing Kalman filter, it is difficult to balance the system with noisy angle feedback.

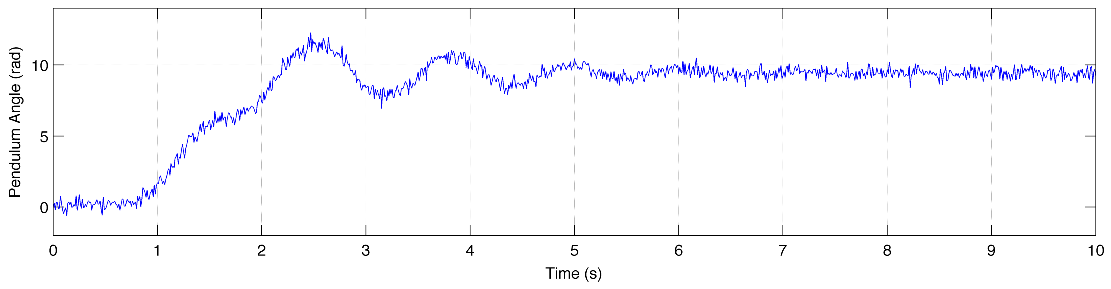


Figure 4.15 System response without applying Kalman filter on the noisy feedback signal.

To see the effects of filtering on the system, Kalman filter block is added to the constructed simulation model as shown in Figure 4.16.

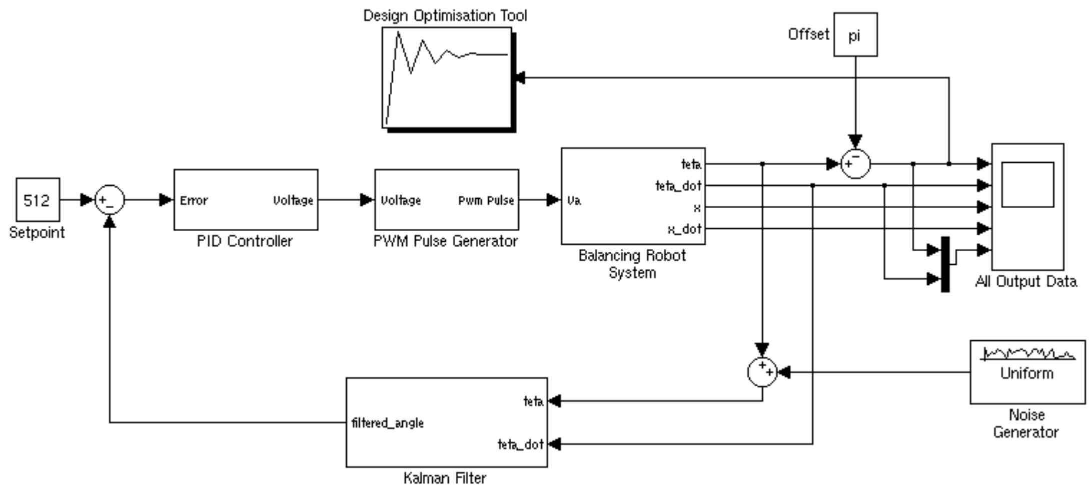


Figure 4.16 Block diagram of the PID controlled balancing robot with Kalman filter implementation.

As stated previously in Section 3.4.1, Kalman filter process angle and angular velocity signals together to eliminate the noise and obtain a clear signal as an output (Ooi, 2003). Since angle signal is measured by accelerometer and it is known that it is effected from vibrations, a noise generator with a bound of ± 0.1 rad and sampling rate of 0.01 s is added to the angle feedback in order to simulate the noise caused by the vibrations.

When the simulation is run for 10 seconds, system response of the closed-loop system is given in Figure 4.17.

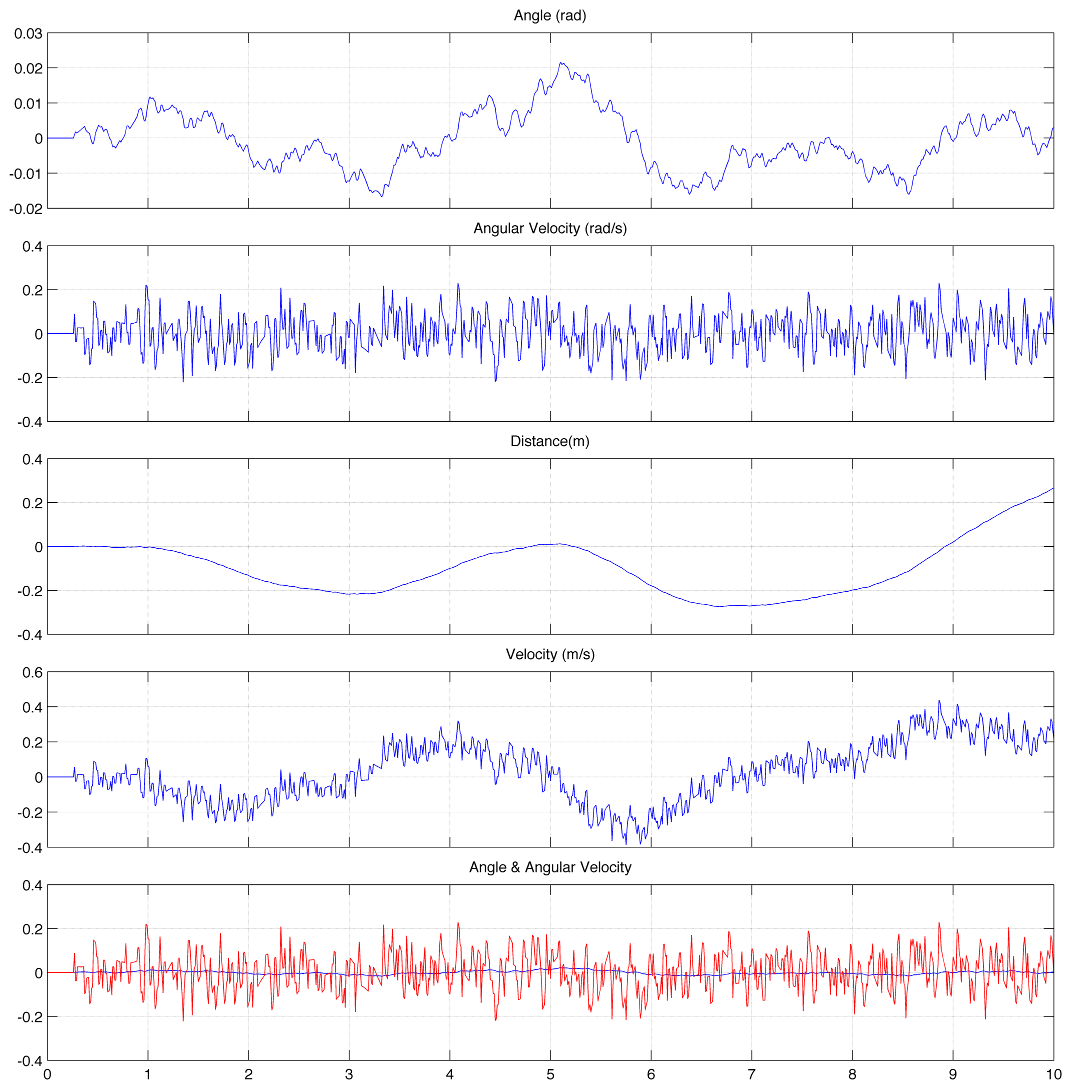


Figure 4.17 Simulation results of the PID controlled balancing robot behavior (with Kalman filter)

As it can be seen from the Figure 4.16, the robot successfully balances itself with a ripple of about 0.01 rad. Kalman filter's ability to eliminate noise can be seen in Figure 4.18.

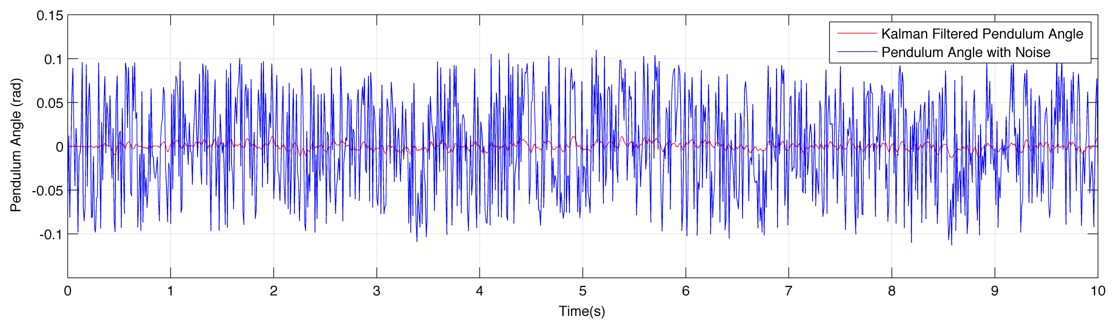


Figure 4.18 Simulation result of Kalman filter output with noisy angle feedback.

When the simulation and experimental results are compared, it is seen that experimental results agree with the simulation results with the PID parameters determined from simulation models. Angle error of the robot is constrained to the predetermined limits. The stability of the robot is maintained in the real system and the experimental results approve this condition.

CHAPTER FIVE
CONCLUSIONS

In this thesis, the objectives of this study are accomplished successfully. A two wheeled self balancing robot prototype is designed and its control for balancing around the upright position is accomplished.

For the design of the robot, each part forming the balancing robot is designed in 3D using Solidworks software. Materials are assigned to these parts and inertia, mass, COG properties of the pendulum and the wheels are calculated using this software.

In order to perform balancing of the two wheeled robot, pendulum angle which is obtained from the Acc-Gyro IMU board is used as a feedback signal. This feedback signal is acquired using Accelerometer and Gyroscope sensors. Accelerometer signal, which contains noise and Gyroscope signal, which drifts over time are combined using Kalman filter to obtain a noiseless angle signal. Then, this angle feedback is used for calculating angle error and hence applying PID control. PID parameters, which are related to each other, are obtained with simulations that are constructed in Simulink. Design Optimization Tool, a numerical optimization tool build in Simulink, is used to find a feasible solution for PID parameters. After these parameters are found, they are experimented in manufactured balancing robot and it has seen that found PID parameters balances the robot as desired. To check it's integrity precisely, experimental results are logged and they are compared with the simulation results. It has been observed that the ripple range in the measured inclination angle agrees with the simulations.

Although balancing of the robot is successfully achieved, there are several points, which can be improved for better balancing as a further study.

1. Backlash of the DC motors' gearbox is one of the reasons of the ripples. Since balancing progress requires instant rotation changes of DC motor to

balance itself at the upright position, effects of the backlash are visible. A gear mechanism without backlash, like a harmonic drive mechanism, can improve the balancing performance.

2. PID controller, which is commonly used for controlling linear systems, is used for controlling the nonlinear system. Although it has successfully balanced the robot, using other control schemes are possible, for a better balancing.

3. During mathematical modeling of the system, it’s assumed that a single DC motor is driving a single shaft. So, only forward and backward balancing movements are considered for this balancing system. For future research, a new mathematical model can be constructed including left and right wheels of the robot individually. Their effects of rotation on balancing can be investigated and experimented.

4. In this balancing robot pendulum angle is used as a feedback and position of the robot in the horizontal direction is not controlled. For position control, encoders located at the back of the DC motors can be used and robot’s position can also be stabilized just like pendulum angle as suggested in the literature.

5. The main goal of this balancing robot was to balance itself where it is left off. In addition, a control scheme can be added to the system to control the motion of the robot in horizontal direction (route tracking).

REFERENCES

ADXL335 Datasheet. (n.d.). Retrieved April, 2010, from http://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf.

Anderson, D. P. (2010). *nBot, a two wheel balancing robot.* Retrieved March, 2010, from http://www.geology.smu.edu/~dpa-www/robo/nbot/.

Arduino Uno Webpage. (n.d.). Retrieved April, 2010, from http://arduino.cc/en/Main/ArduinoBoardUno.

Arduino PWM Frequency Webpage. (n.d.). Retrieved August, 2011, from http://arduino.cc/en/Reference/analogWrite.

Åström, K. J. (2002). *Feedback systems: An introduction for scientists and engineers.* Retrieved April, 2012, from http://www.cds.caltech.edu/~murray/amwiki/index.php/Main_Page.

A*Tmega328 Microcontroller Datasheet.* (n.d.). Retrieved April, 2010, from http://www.atmel.com/Images/doc8161.pdf.

Baik, K. (2008). *BoRam: Balancing robot using arduino and lego*. Retrieved February, 2012, from http://boram.wdfiles.com/local--files/start/PROJECT_REPORT_BORAM2-final.pdf.

Elmenreich, W. (2002). *Sensor fusion in time-triggered systems.* Retrieved March, 2012, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.69.9299.

Faulhaber DC Motor Datasheet. (n.d.). Retrieved February, 2010, from http://www.faulhaber.com/uploadpk/EN_2342_CR_DFF.pdf.

Fritzing Webpage. (n.d.). Retrieved February, 2011, from http://www.fritzing.org.

Grasser, F., D’Arrigo, A., Colombi, S., & Rufer, A. C. (2002). Joe: A mobile, inverted pendulum. *IEEE Transactions on Industrial Electronics*, 49 (1), 107-114.

IDG500 Datasheet. (n.d.). Retrieved April, 2010, from http://www.sparkfun.com/datasheets/Components/SMD/Datasheet_IDG500.pdf.

Jeong, S. & Takahashi, T. (2008). *Wheeled inverted pendulum type assistant robot: Design concept and mobile robot*. Intel Serv Robotics, 1, 313-320. Retrieved May, 2012, from SpringerLink database.

Kadir, H. B. (2005). *Modeling and control of a balancing robot using digital state space approch*. Retrieved May, 2011, from http://eprints.utm.my/2721/.

Kalra, S., Dipesh, P. & Stol, K. (2007). *Design and hybrid control of a two wheeled robotic platform.* Retrieved July 2010, from http://www.araa.asn.au/acra/acra2007/papers/paper186final.pdf.

Lahdhiri, T., Carnal, C. L. & Alouani, A.T. (1994). Cart-pendulum balancing problem using fuzzy logic control. *Southeastcon '94 - Proceedings of the 1994 IEEE*, 393-397.

Muškinja, N. & Tovornik, B. (2000). *Controlling of real inverted pendulum by fuzzy logic.* Retrieved May, 2012, from http://www.apca.pt/~apca_docs/CONTROLO2000/Papers/C2000_M02.pdf.

Ooi, R. C. (2003). *Balancing a two-wheeled autonomous robot*. Retrieved May, 2011, from http://robotics.ee.uwa.edu.au/theses/.

Pololu DC Motor Driver Webpage. (n.d.). Retrieved April, 2010, from http://www.pololu.com/catalog/product/707.

Pololu DC Motor Webpage. (n.d.). Retrieved April, 2010, from http://www.pololu.com/catalog/product/1443.

Pololu Wheels Webpage. (n.d.). Retrieved April, 2010, from http://www.pololu.com/catalog/product/1435/specs.

Quintero, S. A. P. (2008). *Controlling the inverted pendulum*. Retrieved June, 2011, from http://www.ece.ucsb.edu/~roy/student_projects/.

Segway Personal Transporter Webpage. (n.d.). Retrieved May, 2012, from http://www.segway.com/individual/model.

Sparkfun IMU Board Webpage. (n.d.). Retrieved April, 2010, from http://www.sparkfun.com/products/9268.

Starlino IMU Guide Webpage. (n.d.). Retrieved March, 2010, from http://www.starlino.com/imu_guide.html.

Sun, L. & Gan, J. (2010). Researching of two-wheeled self-balancing robot base on LQR Combined with PID. *Intelligent systems and applications (ISA), 2010 2th international workshop*, 1-5.

VNH3SP30 Datasheet. (n.d.). Retrieved July, 2011, from http://www.pololu.com/file/0J51/vnh3sp30.pdf.

Yong, C. & Kwong, C. F. (2011). Wireless controlled two wheel balancing robot. *International Journal of Network and Mobile Technologies*, 2 (2), 88-109. Retrieved May, 2012, from http://www.ijnmt.com/JournalPapers/Vol2No2/.

NOMENCLATURE

τ_m - DC motor torque (Nm)

τ_a - Applied torque to the shaft of the DC motor (Nm)

k_m - DC motor torque constant (Nm/A)

k_e - DC motor back-emf constant (Vs/rad)

k_f - Friction constant (Nms/rad)

i_a - DC motor armature current (A)

E_a - DC motor back-emf voltage (V)

V_a - Applied terminal voltage to DC motor (V)

θ_w- Rotation angle of the wheels (rad)

θ_p - Angular position of the pendulum (rad)

ϕ - Very small angle which is used for linearization (rad)

ω - DC motor shaft angular velocity (rad/s)

I_R - DC motor shaft moment of inertia (kgm^2)

I_W - Moment of inertia of one wheel (kgm^2)

I_p - Moment of inertia of the pendulum (kgm^2)

M_w - Mass of one wheel (kg)

M_p - Mass of the pendulum (kg)

R - DC motor armature resistance (Ohms)

L - DC motor armature inductance (H)

$P = H$ - Reaction forces between one wheel and robot chassis (N)

H_f - Friction force between the ground and the wheels of the robot (N)

r - Radius of the wheels (m)

l - Lenght between the rotational axis of the wheels and center of gravity (COG) of the pendulum (m)

g - Acceleration of gravity (m/s^2)

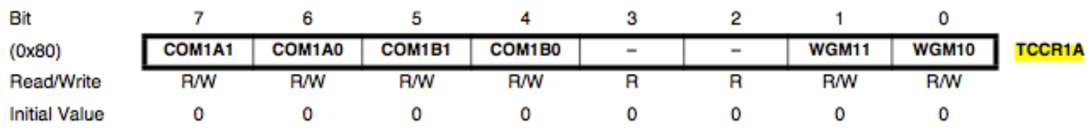
APPENDIX

Configuring Timers to Set PWM Frequency

Arduino use Atmel ATmega328 microcontroller and by using its timers, it is possible to change Arduino's default PWM frequency. In this thesis in order to speed control two DC motors, two PWM pins are used which are connected to 10th and 11th pins of Arduino. 10th pin is connected to Timer1 and 11th pin is connected to Timer2. So, it is necessary to configure both of these timers in order to send PWM pulses to DC motors with the same frequency.

Each timer has it's control registers which can be configured. Timer1's control registers are TCCR1A, TCCR1B and TCCR1C. Similarly, Timer2's control registers are TCCR2A, TCCR2B and TCCR2C. TCCR1C and TCCR2C control registers are used only when a non-PWM mode is active. So, they won't be configured in this study. Also Timer1 and Timer2's configurations are exactly same with each other, hence just Timer1's configurations will be shown.

TCCR1A control register is shown below. Bits are selected according to the needs from the given tables in the ATmega328 datasheet (ATmega328 Microcontroller Datasheet).



Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

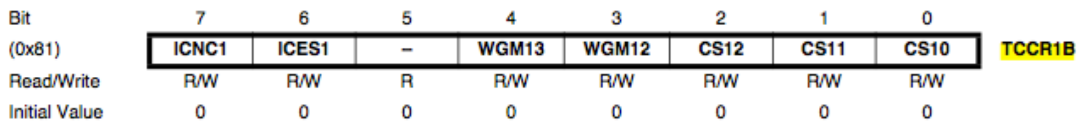
TCCR1A register's bit descriptions, selections and bit values according to the selections are given in the table below.

Bit No	Description	Bit Name	Bit Value	Selection
Bit 7:6	Compare Output Mode for Channel A	COM1A1	0	Normal Port Operation
		COM1A0	0	

Bit 5:4	Compare Output Mode for Channel B	COM1B1	0	Normal Port Operation
		COM1B0	0	
Bit 3:2	Reserved Bit	N/A	0	No selection is possible.
		N/A	0	
Bit 1:0	Waveform Generation Mode	WGM11	0	PWM, Phase Correct, 8-bit mode
		WGM10	1	

So, TCCR1A register configuration can be written in bit format as B00000001.

As a second control register: TCCR1B is shown below.



Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ATmega328 datasheet (http://www.atmel.com/Images/doc8161.pdf)

TCCR1B register's bit descriptions, selections and bit values according to the selections are given in the table below.

Bit No	**Description**	**Bit Name**	**Bit Value**	**Selection**
Bit 7	Input Capture Noise Canceler	ICNC1	0	Not activated
Bit 6	Input Capture Edge Select	ICES1	0	Not activated
Bit 5	Reserved Bit	N/A	0	No selection is possible.
Bit 4:3	Waveform Generation Mode	WGM13	0	PWM, Phase Correct, 8-bit mode
		WGM12	0	
Bit 2:0	Clock Select	CS12	0	$clk_{I/O}/8$ (from prescaler)
		CS11	1	
		CS10	0	

So, TCCR1B register configuration can be written in bit format as B00000010.

In order to calculate the PWM frequency of the configured microcontroller, the following formula is used which is given in ATmega328 datasheet (ATmega328 Microcontroller Datasheet) for the Phase Correct PWM mode.

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N.510}$$

where, N represents the prescaler factor (1, 8, 32, 64, 128, 256 or 1024) and $f_{clk\_I/O}$ represents Arduino's clock speed which is given in Table 3.1.

As a result, PWM frequency is found as;

$$f_{OCnxPCPWM} = \frac{16000000}{8\,.510} = 3921.56\,Hz$$