Sudan University of Science and Technology
College of Engineering
School of Electrical and Nuclear Engineering

DESIGN AND IMPLEMENTATION OF TWO WHEELED SELF BALANCING ROBOT USING PID CONTROLLER

A Project Submitted In Partial Fulfillment for the Requirements of the Degree of B.Sc. (Honor) In Electrical Engineering (Control)

Prepared By:

1. Osman Mohammed Elamin Khidir Ahmed
2. El-Amin Omer Mohammed El-Amin
3. Asaad Salem Abdurraheem Salem
4. Ahmed Mohammed Osman Mohammed

Supervised By:

Dr. Awadalla Taifour Ali

October 2017

الآية

قال تعالى :

ٱللَّهُ لَآ إِلَٰهَ إِلَّا هُوَ ٱلْحَىُّ ٱلْقَيُّومُ ۚ لَا تَأْخُذُهُۥ
سِنَةٌ وَلَا نَوْمٌ ۚ لَّهُۥ مَا فِى ٱلسَّمَٰوَٰتِ وَمَا فِى ٱلْأَرْضِ ۗ
مَن ذَا ٱلَّذِى يَشْفَعُ عِندَهُۥٓ إِلَّا بِإِذْنِهِۦ ۚ يَعْلَمُ مَا بَيْنَ
أَيْدِيهِمْ وَمَا خَلْفَهُمْ ۖ وَلَا يُحِيطُونَ بِشَىْءٍ مِّنْ عِلْمِهِۦٓ
إِلَّا بِمَا شَآءَ ۚ وَسِعَ كُرْسِيُّهُ ٱلسَّمَٰوَٰتِ وَٱلْأَرْضَ ۖ
وَلَا يَـُٔودُهُۥ حِفْظُهُمَا ۚ وَهُوَ ٱلْعَلِىُّ ٱلْعَظِيمُ ﴿٢٥٥﴾

سورة البقرة

DEDICATIONS

This study is lovingly dedicated to our parents for their emotional and financial support, our brothers, our sisters and our friends whose has been constant source of inspiration for us. They have given us the drive and discipline to tackle any task with enthusiasm and determination. Without their love and support this project would not have been made possible.

ACKNOWLEDGEMENT

We wish to express our profound gratitude to our Supervisor assistant professor **Awadalla Taifour Ali** for his valuable guidance, continues encouragement, worthwhile suggestions and constructive ideas throughout this project. His support, pragmatic analysis and understanding made this study a success and knowledgeable experience for us.

Abstract

The dynamics of the two-wheeled self-balancing robot system is similar to that of the inverted pendulum, which is unstable and prone to tip over. The inverted pendulum is a classic problem in dynamics and control theory, and is used as a benchmark for testing control strategies and algorithms. The system includes an angle sensor, two geared DC motors, two motor drivers and an Arduino. The basic idea is to use the torque generated by the motors to maintain the structure's vertical equilibrium state.

In this project the model is obtained by employing physical laws. This model despite its simplicity can represent a typical real system such as Segway Personal Transporter. The system is nonlinear and unstable, but can be linearized for small angles.

The Arduino microcontroller acquires the readings from the angle sensor and compare it with the desired angle. A PID algorithm computes the exact pulse width modulated signal which is fed to the motors.

The instant response of the system has been plotted using MATLAB software, by interfacing Arduino with computer. System response to different values of controller parameters were obtained and the parameters that gave the best performance of the system were chosen in the final implementation. The results showed that the model is not yet reliable. The reasons for this are discussed and recommendations for future development are list.

المستخلص

الروبوت ذاتي الإتزان مماثل من ناحية ديناميكية للبندول المقلوب أي أنه غير مستقر و معرض للسقوط. البندول المقلوب يمثل مشكلة تقليدية في مجال الديناميكا ونظرية التحكم ويستخدم كمعيار لاختبار استراتيجيات التحكم المختلفة. يتكون النظام من محساس لقياس الزاوية، و محركي تيار مستمر، ومتحكم. الفكرة الاساسية تقوم على استخدام العزم المنتج بواسطة المحركات للحفاظ على حالة التوازن الرأسي. في هذا المشروع تم التحصل على النموذج الرياضي باستخدام القوانين الفيزيائية. رغم بساطة هذا النموذج إلا أنه قادر على تمثيل أنظمة عملية من الحياة اليومية. النظام غير خطي وغير مستقر لكن يمكن جعله خطيا عند الزوايا الصغيرة. متحكم الاردوينو يتحصل على القراءات من محساس الزاوية ويقارنها مع الزاوية المطلوبة.المتحكم التناسبي التكاملي التفاضلي يحسب الإشارة المضمنة بعرض النبضة لتوجيه المحركات. تم إيجاد ورسم الاستجابة اللحظية للنظام باستخدام برنامج ماتلاب، عن طريق ربط متحكم الاردوينو بجهاز الحاسوب. حيث تم التحصل على إستجابة النظام لقيم مختفة لمعاملات المتحكم. وذلك للتحصل على أفضل إستجابة ممكنة. بينت النتائج أن النظام لا يستجيب بالصورة المطلوبة ووضعت بعض الطرق والأساليب لمزيد من التطوير في التوصيات.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Kp	Proportional gain
Ki	Integral gain
Kd	Derivative gain
e	Signal Error
u	Control Signal
F_v	Force applied to cart
F_x	Horizontal reaction force at the pivot point
F_y	Vertical reaction force at the pivot point
x	Horizontal displacement
y	Vertical displacement
x_g	Horizontal displacement of the center of gravity
y_g	Vertical displacement of the center of gravity
M	Mass of the moving cart
m	Mass of the pendulum
l	Distance between pendulum mass center and pivot point
θ	Angular displacement
F_f	Force due to friction
g	Gravity acceleration
γ	Viscous friction coefficient
I	Moment of inertia of the pendulum

B	Magnetic field strength
l_{c}	Length of the conductor
T_{m}	The torque developed by the motor
i_{a}	Armature current
v_{b}	Electromotive force (back emf)
k_{b}	Back emf constant
θ_{m}	Angular displacement of the motor
e_{a}	The applied armature voltage
R_{a}	Armature resistance
T_{m}	Motor torque
K_{t}	Motor torque constant
D_{m}	Equivalent viscous damping at the armature
J_{m}	Equivalent inertia at the armature
L_{a}	Armature inductance

LIST OF ABBREVIATIONS

LTI	Linear Time Invariant
IMU	Inertia Measurement Unit
SISO	Single Input, Single Output systems
MIMO	Multiple Inputs and Multiple Outputs
A/D	Analog-to-Digital
P	Proportional controller
PD	Proportional Derivative controller
PI	Proportional Integral controller
PID	Proportional Integral Derivative
AUAV	Autonomous Unmanned Aerial Vehicle
LQR	Linear Quadratic Regulator
ICCA	International Conference on Control and Automation
Pot	Potentiometer
I/O	Input-Output
PWM	Pulse Width Modulation
CNC	Computer Numerical Cutting
RX	Receiver mode
TX	Transmitter mode
SCL	Serial Clock
SDA	Serial Data
IC	Integrated Circuit

CHAPTER ONE

INTRODUCTION

1.1 General Concepts

The self-balancing robot is a two wheeled structure. The idea behind it can be put in this way; when the structure leans forwards or backwards by certain angle, it should drive both wheels in the same direction (i.e. either forwards or backwards) in some speed and acceleration that is proportional to that angle to keep itself in the upright position. The robot can sense tilt with the help of different sensors, then drives the wheels using a couple of motors. All calculations and processing is done by some sort of digital device like a microcontroller unit. Another feature of the self-balancing robot is that it can drive the two wheels independent of each other. In this case, it can turn left or right or even circle around its own center. These robots have the ability to carry and balance different objects on top of them without losing equilibrium. And some other designs may provide wireless communication for remote control. These robots are commercially available in the form of man driven vehicles well-known as Segway's PT (the name Segway is derived from the word segue, which means smooth transition. PT is an abbreviation for personal transporter). Another example might be the hover board for entertainment purposes.

1.2 Problem Statement

The idea of the self-balancing robot is related to the inverted pendulum cart (cart and pole) which is a classic problem in dynamics and control theory. Whereas a normal pendulum is stable when hanging downwards, an inverted pendulum is inherently unstable, and must be actively, precisely and quickly balanced in order to remain upright. That's the case in the self-balancing robot. Trying to remain

stable and retrieve its former position when it's objected to external force (i.e. Nudges, Pokes, etc.). Seeking stability while moving between two points is another challenging matter and might cause losing consistency and as a result the whole thing will eventually fall. The problem is to ensure fast and precise response in order to achieve stability and high performance and also regain its previous location reliably.

1.3 Objectives

- Development of an approximate mathematical model of the two-wheel robot as a Linear Time Invariant (LTI) system for the purpose of design, simulation and implementation based on theory of the inverted pendulum.
- Implement an appropriate filter to get useful readings from the Inertia Measurement Unit (IMU) for the feedback signal.
- Design PID controller to stabilize the system and achieve the desired response.
- Simulation of proposed system to show whether the system is capable of meeting the desired specifications that it has been designed for.
- Implement the real-time response of the robot.
- Implementation of the two-wheeled self-balancing robot.

1.4 Methodology

- Study of previous related works.
- Study the nature of the inverted pendulum problem and for the purposes of modelling.
- System simulation; done with MATLAB software package.
- Analyzing the obtained model to check for stability without a controller and feedback.
- Study and understand the behaviors of the IMU for modeling and simulation.

- Design a model of the structure using AutoCAD software and build the two-wheeled self-balancing robot in the final implementation.
- PID controller parameters; tuned with trial and error method. .
- Build an Arduino based controller and write a PID algorithm using the Arduino IDE.

1.5 Layout

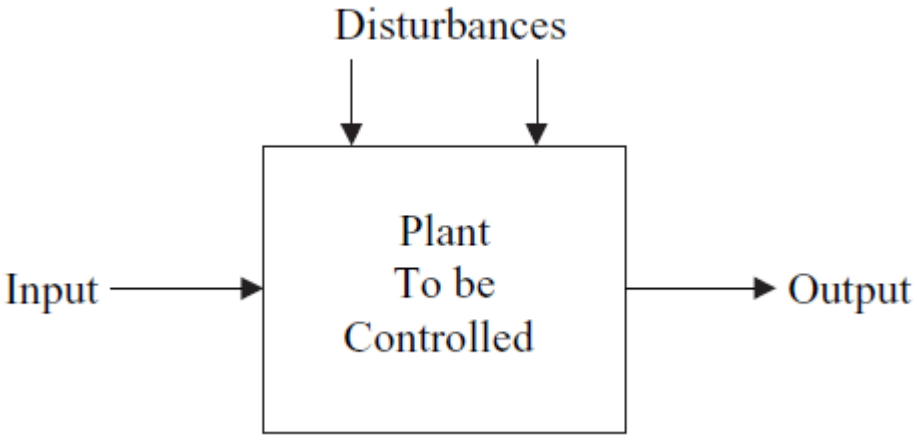
Chapter one provides a brief introduction to self-balancing robot and outlines the idea behind the way it balances. Problem statement addresses its relation to the inverted pendulum and states the problems that ought to be solved as an outcome to this study. Then our main goals are mentioned in the Objectives. The steps to achieve these objectives are listed in the Methodology. Chapter Two gives a literature review of self-balancing robots. Runs through robots in general narrowing down to self-balancing robot. A summarized review of the used parts (Arduino, DC Motor, sensors, etc.) is also included. The concept behind the PID controller is explained and the reasons of why we preferred it over other controllers. Chapter Three shows the fabrication process and the extraction of the robot parameters (i.e. mass, length, moment of inertia, etc.).A mathematical model of the system is also provided in this chapter. Chapter Four focuses on analysis and design, the study of stability, steady-state error and transient response of the robot. The software part includes the program flow chart and Arduino code. Real-time response of how the system behaves is presented in the experimental results. Chapter Five provides the conclusion and recommendations.

CHAPTER TWO
THEORETICAL BACKGROUND AND LITERATURE REVIEW

2.1 System Control

Control engineering is concerned with controlling a dynamic system or plant. A dynamic system can be a mechanical system, an electrical system, a fluid system, a thermal system, or a combination of two or more types of system. The behavior of a dynamic system is described by differential equations. Given the model (differential equation), the inputs and the initial conditions, we can easily calculate the system output. A plant can have one or more inputs and one or more outputs. Generally a plant is a continuous-time system where the inputs and outputs are also continuous in time. For example, an electromagnetic motor is a continuous-time plant whose input (current or voltage) and output (rotation) are also continuous signals. A control engineer manipulates the input variables and shapes the response of a plant in an attempt to influence the output variables such that a required response can be obtained. A plant is an open-loop system where inputs are applied to drive the outputs. For example, a voltage is applied to a motor to cause it to rotate. In an open-loop system there is no knowledge of the system output. The motor is expected to rotate when a voltage is applied across its terminals, but we do not know by how much it rotates since there is no knowledge about the output of the system. If the motor shaft is loaded and the motor slows down there is no knowledge about this. A plant may also have disturbances affecting its behavior and in an open-loop system there is no way to

know, or to minimize these disturbances. Figure 2.1 shows an open-loop system where the system input is expected to drive the system output to a known point



(e.g. to rotate the motor shaft at a specified rate). This is a Single-Input, Single-Output (SISO) system, since there is only one input and also only one output is available In general, systems can have Multiple Inputs and Multiple

Figure 2.1: Open loop control system

Outputs (MIMO). Because of the unknowns in the system model and the effects of external disturbances the open-loop control is not attractive. There is a better way to control the system, and this is by using a sensor to measure the output and then comparing this output with what we would like to see at the system output. The difference between the desired output value and the actual output value is called the error signal. The error signal is used to force the system output to a point such that the desired output value and the actual output value are equal. This is termed closed-loop control, or feedback control. Figure 2.2 shows a typical closed-loop system. One of the advantages of closed-loop control is the ability to compensate for disturbances and yield the correct output even in the presence of disturbances. A controller (or a compensator) is usually employed to read the error signal and drive the plant in such a way that the error tends to zero. Closed-loop systems have the advantage of greater accuracy than open-loop systems.

They are also less sensitive to disturbances and changes in the environment. The time response and the steady-state error can be controlled in a closed-loop system [1].

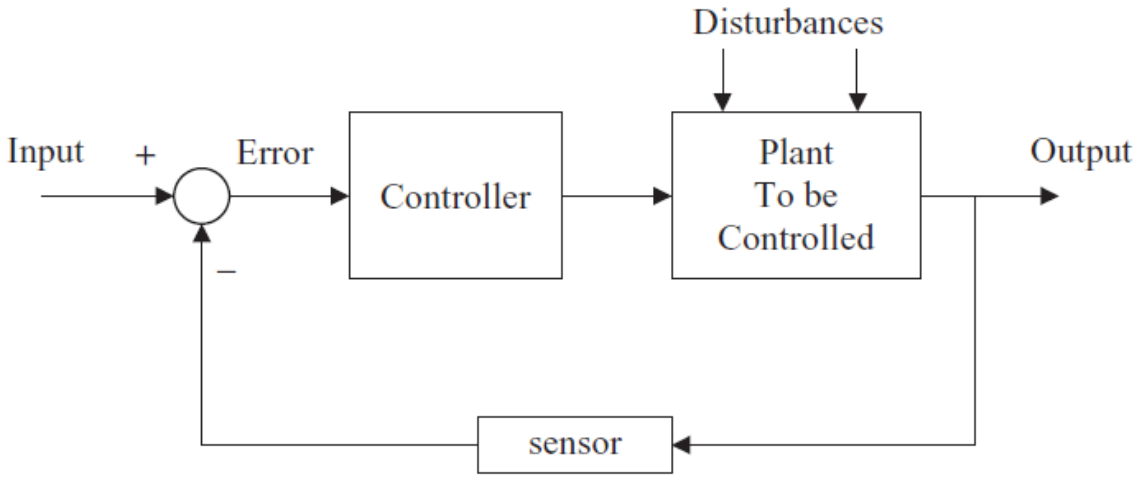


Figure 2.2: Closed-loop system

Sensors are devices which measure the plant output. For example, a thermistor is a sensor used to measure the temperature. Similarly, a tachogenerator is a sensor used to measure the rotational speed of a motor, and an accelerometer is used to measure the acceleration of a moving body. Most sensors are analog devices and their outputs are analog signals (e.g. voltage or current). These sensors can be used directly in continuous-time systems. For example, the system shown in Figure 2.2 is a continuous-time system with analog sensors, analog inputs and analog outputs. Analog sensors cannot be connected directly to a digital computer. An Analog-to-Digital (A/D) converter is needed to convert the analog output into digital form so that the output can be connected to a digital computer. Some sensors (e.g. accelerometer sensors) provide digital outputs and can be directly connected to a digital computer. With the advent of the digital computer and low-cost microcontroller processing elements, control engineers began to use these programmable devices in control systems. A digital computer can keep

track of the various signals in a system and can make intelligent decisions about the implementation of a control strategy [1].

2.2 Controllers

The concept of a control system is to sense deviation of the output from the desired value and correct it, till the desired output is achieved. The deviation of the actual output from its desired value is called an error. The measurement of error is possible because of feedback. The feedback allows us to compare the actual output with its desired value to generate the error. The error is denoted as $e(t)$. The desired value of the output is also called reference input or a set point. The error obtained is required to be analyzed to take the proper corrective action. The controller is an element which accepts the error in some form and decides the proper corrective action. The output of the controller is then applied to the process or final control element. This brings the output back to its desired set point value. The controller is the heart of a control system. The accuracy of the entire system depends on how sensitive is the controller to the error detected and how it is manipulating such an error. The controller has its own logic to handle the error. Now a days for better accuracy, the digital controllers such as microprocessors, microcontrollers, and computers are used. Such controllers execute certain algorithm to calculate the manipulating signal [2].

2.2.1 Classification of controllers

The classification of the controllers is based on the response of the controller and mode of operation of the con

troller. For example, in a simple temperature control of a room, the heater is to be controlled. It should be switched on or off by the controller when temperature

crosses its set point. Such an operation of the controller is called discontinuous operation and the mode of operation is called discontinuous mode of controller. But in some process control systems, simple on/off decision is not sufficient. For example, controlling the steam flow by opening or closing the valve. In such case is said to be operating in a continuous mode. Thus, the controllers are basically classified as discontinuous controllers and continuous controllers. The discontinuous mode controllers are further classified as ON-OFF controllers and multi-position controllers. The continuous mode controllers are further classified as proportional controllers, integral controllers and derivative controllers. Some continuous mode controllers can be combined to obtain composite controller mode. The examples of such controllers are Proportional plus Integral (PI), Proportional plus Derivative (PD) and Proportional plus Integral plus Derivative (PID) controllers. The most of the controllers are placed in the forward path of control system. But in some cases, input to the controller is controlled through a feedback path. The example of such a controller is rate feedback controller [2].

2.2.2 Continuous controller modes

In the discontinuous controller mode, the output of the controller is discontinuous and not smoothly varying. But in the continuous controller mode, the controller output varies smoothly proportional to the error or to some form of the error. Depending upon which form of the error is used as the input to the controller to produce the continuous controller output, these controllers are classified as proportional control mode**,** integral control mode**,** and derivative control mode [2].

A. PID controller

The PID controller has three terms. The proportional term P corresponds to proportional control. The integral term I gives a control action that is

proportional to the time integral of the error. This ensures that the steady state error becomes zero. The derivative term D is proportional to the time derivative of the control error. This term allows prediction of the future error [3].

B. Proportional control mode

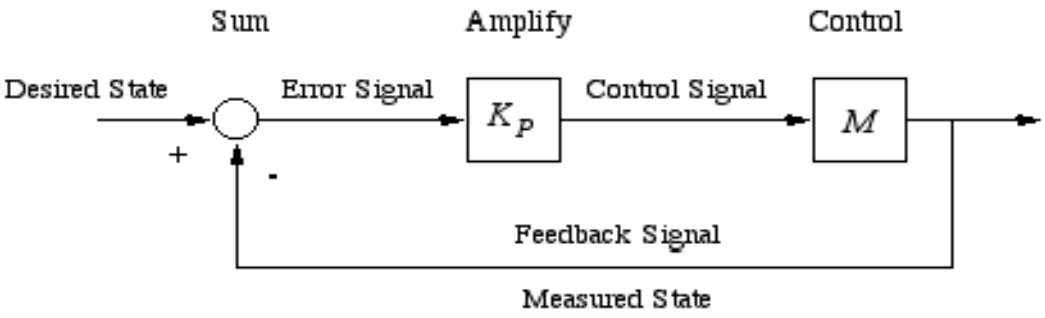


Figure 2.3: Simple proportional controller

The reason why on-off control often gives rise to oscillations is that the system overreacts because a small change in the error will make the manipulated variable change over the full range. This effect is avoided in proportional control where the characteristic of the controller is proportional to the control error for small errors. Figure 2.4 shows the characteristic of a proportional controller. The controller is thus characterized by the nonlinear function $u = fc(e)$ Shown in Figure 2.4. The proportional controller can be suitable where: Manual reset of the operating point is possible, Load changes are small, and the dead time exists in the system is small [3].

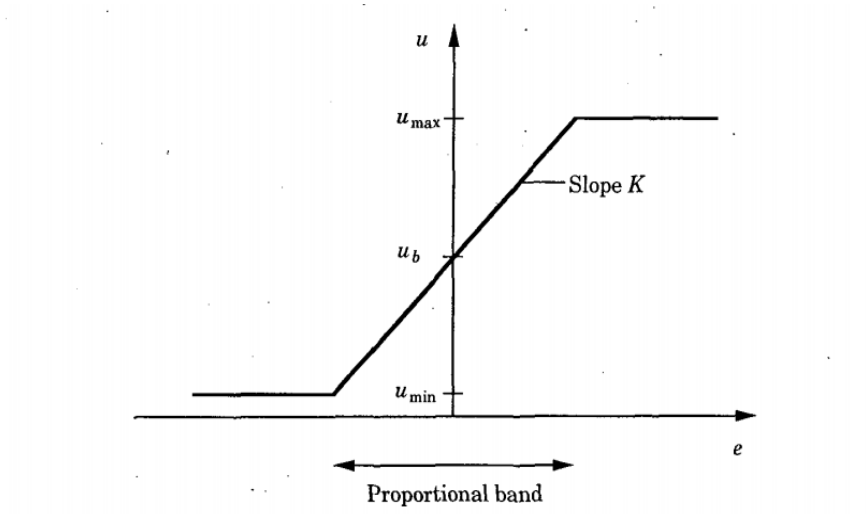


Figure 2.4: Characteristics of a proportional controller

C. Integral control mode

In the proportional control of a plant whose transfer function does not possess an integrator 1s, there is a steady-state error, or offset, in the response to a step input. Such an offset can be eliminated if the integral control action is included in the controller. In the integral control of a plant, the control signal—the output signal from the controller—at any instant is the area under the actuating-error-signal curve up to that instant. The control signal $u(t)$ can have a nonzero value when the actuating error signal $e(t)$ is zero, as shown in Figure 2.5. This is impossible in the case of the proportional controller, since a nonzero control signal requires a nonzero actuating error signal. (A nonzero actuating error signal at steady state means that there is an offset). Note that integral control action, while removing offset or steady-state error, may lead to oscillatory response of slowly decreasing amplitude or even increasing amplitude, both of which are usually undesirable [4].

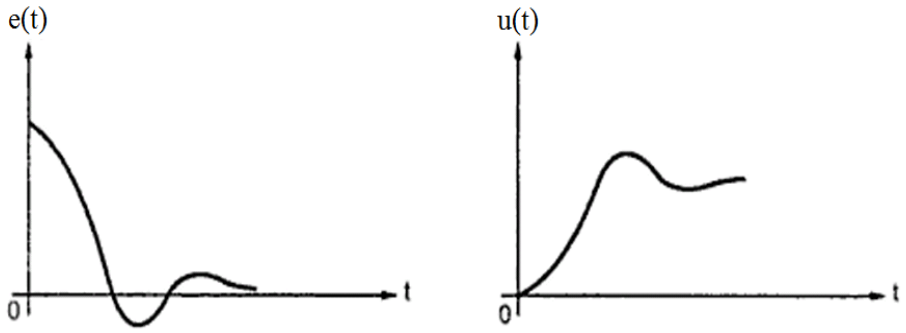


Figure 2.5: Non-zero control signal when error is zero

D. Derivative control mode

The purpose of the derivative action is to improve the closed-loop stability. The instability mechanism can be described intuitively as follows. Because of the process dynamics, it will take some time before a change in the control variable is noticeable in the process output. Thus, the control system will be late in correcting for an error. The action of a controller with proportional and derivative action may be interpreted as if the control is made proportional to the *predicted* process output, where the prediction is made by extrapolating the error by the tangent to the error curve in Figure 2.6. The basic structure of a PD controller is [3]:

$$u(t) = K\left(e(t) + T_d \frac{de(t)}{dt}\right) \qquad (2.1)$$

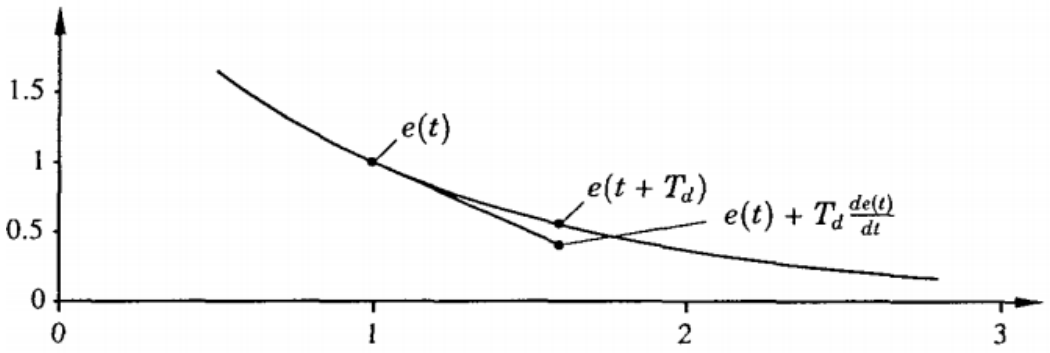


Figure 2.6: Interpretation of derivative action as predictive control

2.3 Robot Overview

There are many definitions of robot and no real consensus has been attained so far. Loosely define a robot as an electromechanical device which is capable of reacting in some way to its environment, and take autonomous decisions or actions in order to achieve a specific task. This means that a blender, a lamp, or a car would not be considered as robots since they have no way of perceiving their environment. On the other hand, a vacuum cleaner that can navigate around a room, or a solar panel that seeks the sun, can be considered as a robotic system. Building a robot incorporates aspects of many disciplines including engineering (mechanical, electrical, computer), sciences (mathematics and physics) and arts (aesthetics) and users are free to use their imagination. It also requires understanding of motors, sensors, microcontrollers and programming.

Robots can be used in almost any situation and are primarily intended to help humans in some way. Robots help liberate people from unpleasant or dangerous tasks and give them more liberty and security. Also they are used in a variety of applications at work, in public, in hazardous environments, in locations such as deep-sea, battlefields and space, just to name a few. In addition to the service areas such as cleaning, surveillance, inspection and maintenance, we utilize these robots where manual task execution is dangerous, impossible or unacceptable.

2.3.1 Types of Robots

The types of robots possible are numerous, the major robot types are:

A. Air robot

An Autonomous Unmanned Aerial Vehicle (AUAV) is very appealing and is entirely within the capability of many robot enthusiasts. However, the advantages of building an autonomous unmanned aerial vehicles, especially if you are a beginner, have yet to outweigh the risks. When considering an aerial vehicle,

most hobbyists still use existing commercial remote controlled aircraft. On the professional side, aircraft such as the US military Predator were initially semi-autonomous though in recent years Predator aircraft have flown missions autonomously.

Figure 2.7: Air robots

B. Water robot

An increasing number of hobbyists, institutions and companies are developing unmanned underwater vehicles. There are many obstacles yet to overcome to make underwater robots attractive to the wider robotic community though in recent years, several companies have commercialized pool cleaning “robots”. Underwater vehicles can use ballast (compressed air and flooded compartments), thrusters, tail and fins or even wings to submerge. Other aquatic robots such as pool cleaners are useful commercial products.

Figure 2.8: Water robots

C. Land robot

Especially the wheeled ones, are the most popular mobile robots as they usually require the least investment while providing significant exposure to robotics. On the other hand, the most complex type of robots is the humanoid (akin to a human), as it requires many degrees of freedom and synchronizing the motion of many motors, and uses many sensors. Tracks (or treads) might be another choice for land robots. These are what tanks use. Although tracks do not provide added "force" (torque), they do reduce slip and more evenly distribute the weight of the robot, making them useful for loose surfaces such as sand and gravel. Also, a track system with some flexibility can better conform to a bumpy surface. On the other hand, implementing tracks may increase the mechanical complexity and connections.

Figure 2.9: Land robots

An increasing number of robots use legs for mobility. Legs are often preferred for robots that must navigate on very uneven terrain. Most amateur robots are designed with six legs, which allow the robot to be statically balanced (balanced at all times on 3 legs); robots with fewer legs are harder to balance. The latter require "dynamic stability", meaning that if the robot stops moving mid-stride, it might fall over. The downside is that it might involve increased mechanical, electronic and coding complexity.

Figure 2.10: Legged robots

Wheels are by far the most popular method of providing mobility to a robot and are used to propel many different sized robots and robotic platforms. Because it has low cost and it is simple to design wheels can be just about any size, from a few centimeters up to 30 cm and more. Robots can have just about any number of wheels, although 3 and 4 are the most common. Normally a three-wheeled robot uses two wheels and a caster at one end. Four and six wheeled robots have the advantage of using multiple drive motors (one connected to each wheel) which reduces slip. Wheels has the weakness of that it may lose traction (slip). Also the small contact area underneath each wheel makes the robot more susceptible to external disturbances.

2.3.2 The Two Wheeled Self-Balancing Robot

As the name suggests this robot stands on two wheels which makes it unstable as shown in Figure 2.11. Thus it is imperative to involve a feedback system in order to maintain the upright position. The two wheels must be driven together to produce useful actions and ensure stability. It is a traditional problem similar to the well-known (inverted pendulum) found extensively in the literature of control systems and dynamics. The inverted pendulum is basically a rod with its pivot point mounted on a moving cart that moves horizontally. The rod can moves freely around its pivot point. This arrangement is inherently unstable and hence

the cart should be moving back and forth to keep the rod in equilibrium. Besides, the old position of the cart also must be restored without losing or compromising the rod stability. Although the self-balancing robot is inherently unstable, it has several advantages over the statically stable multi-wheeled robots since it has only two wheels (two points touching the ground) it requires less space, since it is based on dynamic stability (it constantly needs to correct its tilt angle to remain stable) it exhibits improved dynamic behavior and mobility. This additional maneuverability allows easy navigation on various terrains, turning sharp corners (it can turn on the spot) and traversing small steps or curbs.

Figure 2.11: Two wheeled self-balancing robot

2.3.3 Literature Review

Mrs. Lekshmy.S, Aleesha George and Athira C.V presented a paper of a balance model as a two wheeled self-balancing robot that is capable of adjusting itself with respect to changes in weight and position. They developed the Balance System from a single gyroscope and a single accelerometer. The stability of the system is to show the capabilities of the ATmega328P in doing PID loops even with limited accuracy in position readings. PID control system is designed to

monitor the motors so as to keep the system in equilibrium. It should be easily reproducible given the right parts and code. They managed to make the robot balance by using an Arduino microcontroller, hobby grade servos, and a six-degree of freedom (axis) accelerometer and gyroscope have been used to create the controlled platform. The controller has been designed to maintain the platform at an initially selected angle when the support structure orientation changes. Kalman filter is used for the fusion of outputs of two sensor [5].

Back in 2012, a group of students namely: Tomislav Tomašić, Andrea Demetlika and Mladen Crneković managed to model, design, built and Control a remotely controlled self-balancing mobile robot. The mechanical structure was first modelled using SolidWorks software. Incremental magnetic encoders, a two-axis accelerometer and a one-axis gyroscope were used to get information about the robot's position and the tilt angle, and a Kalman filter was designed to combine the reading of the gyro and accelerometer to have an accurate values of the robot's tilt angle and position. They first used a PID controller algorithm using only one feedback variable – the tilt angle to stable the robot but simulations and testing showed that even though the robot is stable in regard to the tilt rotation, small disturbances eventually cause big changes in the position. In order to overcome these problems they designed a Linear Quadratic Regulator (LQR) controller, Unlike the PID, the LQR controller uses all state variables (tilt angle, angular velocity, position and linear velocity) to calculate the control value. This allows the robot to hold the position and the tilt angle at desired values. The use of the LQR controller made it possible for the robot to balance even while climbing a slope, while the PID controller didn't achieve that because although the PID controller stabilizes the tilt angle, the robot soon starts to drive down the slope and increases its speed until the control value eventually saturates and the robot falls. They also designed a fuzzy controller successfully and simulated it, but due to the increased complexity of its application in the microcontroller

environment (in relation to other controllers) it was not implemented and tested on a real platform [6].

Hau-Shiue Juang and Kai-Yew Lum presented a paper for the IEEE International Conference on Control and Automation (ICCA) in 2013 explaining how they designed and controlled a two-wheel self-balancing robot using the Arduino microcontroller board. Two control designs based on the linearized equations of motion were adopted for this project: a proportional-integral-differential (PID) control, and a proportional-integral proportional-differential control based on linear-quadratic regulator (LQR) design. The approaches were found to be robust to modeling errors which can be incurred during experimental determination of such electrical and kinematic parameters as moments of inertia and motor gains. They designed a complementary filter to solve the noise problem which the gyro and accelerometer has. They also designed a wheel synchronizer controller consisting of a simple PI controller to make sure that the robot's right and left wheels rotates at the same speed, avoiding problems like motors defects, terrain and hindrance on the ground. This controller adjusts the Pulse Width Modulation (PWM) inputs to the motors so that the difference between the left and right encoders tracks zero [7].

Mikael Arvidsson and Jonas Karlsson from Chalmers University of Technology developed what is known commercially as the Segway personal transporter from scratch. The main objective was to build a vehicle capable of transporting a person weighing up to 100 Kg for 30 minutes or a distance of 10 Km, whichever comes first. The rider controls are supposed to be natural movements; leaning forward or backwards in combination with tilting the handlebar sideways should be the only rider input required to ride the vehicle. It was built using a model-based control design with a linear quadratic controller. The electrical system allows for simple recharging by connecting a Direct Current (DC) adapter between the charging plug and a wall socket. The vehicle has been tested by a

number of different people, with and without previous experience of riding this kind of vehicle. All were able to ride the vehicle [8].

Mahadi Hasan and Chanchal Saha, Md. Mostafizur Rahman and Md. Rabiual Islam Sarker, and Subrata K. Aditya from Asian Institute of Technology, Rajshahi University of Engineering & Technology and Dhaka University had a project named Balancing of an Inverted Pendulum Using PD Controller. The main idea behind this control process is the use of PD (Proportional and Derivative) controller to generate signal to control the speed and direction of the motor. The only sensor used in this project was a potentiometer which was attached to the pendulum rod. The variation in its resistance causes change in voltage and afterward, it was compared with the reference voltage to generate the appropriate control signal. PROTEUS software was used for circuit simulation, and frequency response of the system were analyzed in MATLAB with different values of KP and KD. Finally, to represent the system stability, root locus diagram was drawn using MATLAB [9].

2.4 Inertia Measurement Unit (IMU) Sensor

The self-balancing robot uses a sensor for angular displacement in the Pitch axis as shown in Figure 2.12, the sensor is called Inertial Measurement Unit (IMU).

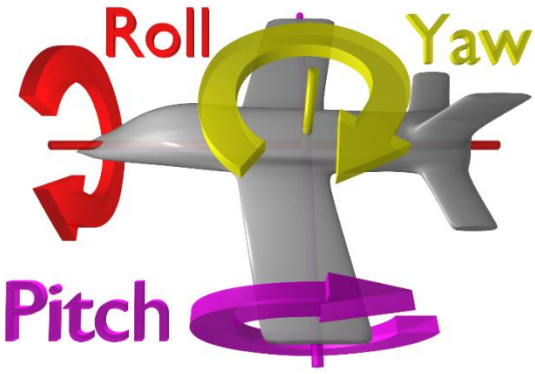


Figure 2.12: Pitch axis

It has six Degrees of Freedom or 6-DoF, as it contains an accelerometer and a gyroscope that both take measurements in three axis. It also contains a temperature sensor to compensate for errors in readings with the temperature variations, for this particular IMU MPU6050 the temperature limits where -40°C to +85°C, but it wasn't used because the robot was working in fairly consisting temperature ranges. The data was obtained from the sensors via Inter Integrated Circuit (I^2C) serial protocol, for power the IMU has a 5 volt pin and a ground pin.

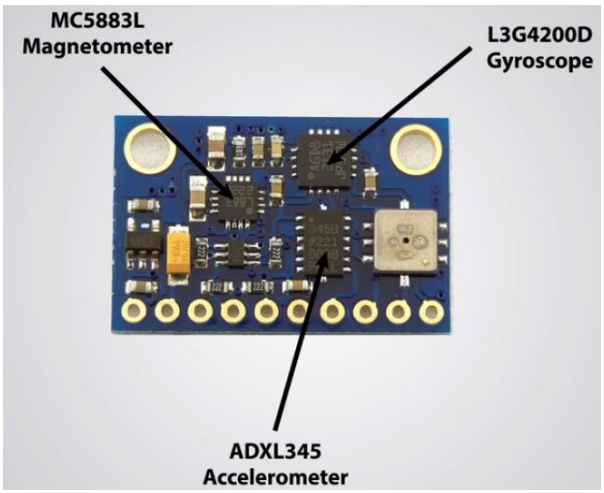


Figure 2.13: Inertia Measurement Unit (IMU)

A. Gyroscope

A gyroscope is a sphere like device that has a wheel in its center free to rotate as shown in Figure 2.14 it measures angular rate (i.e. degrees/sec or º/s), when the wheel is not rotating and we attempt to balance the gyroscope on its needle it would behave as we expected, fall. A dramatic change in this behavior happens when the wheel is rotating in a relevantly high speed, the gyroscope will fight any force applied to it, including gravity. This means that whenever the gyroscope's wheel is spinning and the gyroscope was held firmly to a, let's say an airplane

like in Figure 2.15 it will remain still even if the airplane is tilting, thus measuring the angular displacement.

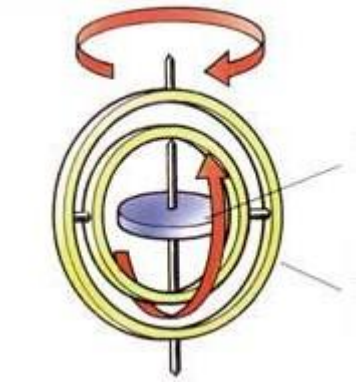


Figure 2.14: Gyroscope

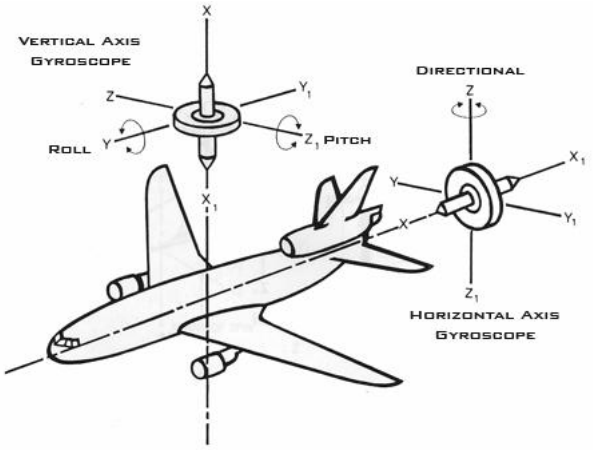


Figure 2.15: Gyroscope in an airplane

In this sensor, (i.e. IMU) the concept of the gyroscope is applied but rather in a different, more convenient way. As shown in Figure 2.16, the wheel of the gyroscope is replaced with an oscillating mass, when an external angular rate is applied a flexible part of the mass would move and make the perpendicular displacement, by measuring the change in capacitance between the oscillating mass and the fixed plates the angular rate could be calculated.

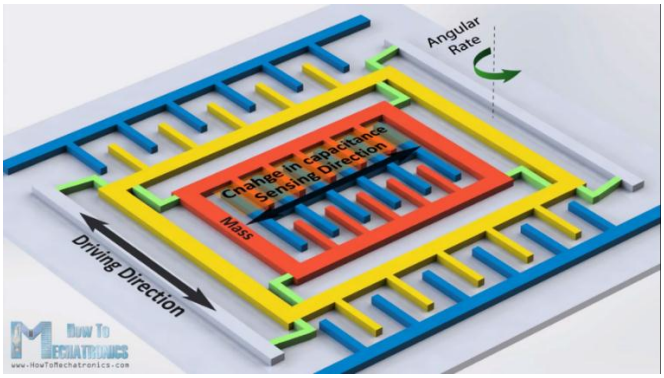


Figure 2.16: Microscopic gyroscope

B. Accelerometer

Another method to measure the angle of the robot is by using an accelerometer, which measures acceleration compared to the earth acceleration which is approximately 9.81 m/s^2, so if an accelerometer is free falling it would measure 0 m/s^2 or 0g (zero-g).

Figure 2.17 demonstrates a microscopic accelerometer, which is the one used in this robot, is constructed. The mass is attached to springs which allow it to move in one axis when there is an acceleration applied in that axis, this movement changes the capacitance between the mass and the fixed plates, this change of capacitance will be measured, processed and will correspond to a particular acceleration value.

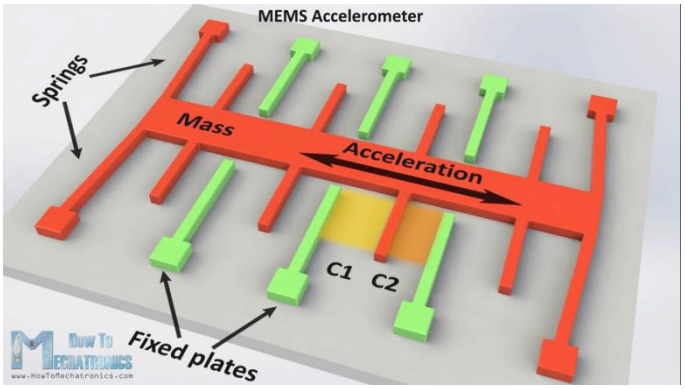


Figure 2.17: Microscopic accelerometer

2.5 Microcontroller

A microcontroller is a single-chip computer, including most of a computer's features, but in limited sizes. Today, there are hundreds of different types of microcontrollers, ranging from 8-pin devices to 40-pin, or even 64- or higher pin devices. It's a microprocessor system which contains data and program memory, serial and parallel Inputs and Outputs (I/O), timers, and external and internal interrupts all integrated into a single chip that can be purchased for a relatively cheap price. The term microcomputer is used to describe a system that includes at minimum a microprocessor, program memory, data memory, and an Input-Output

(I/O) device. Some microcomputer systems include additional components such as timers, counters, and analog-to-digital converters. Thus, a microcomputer system can be anything from a large computer having hard disks, floppy disks, and printers to a single-chip embedded controller. Therefore a microcontroller is meant to perform a specific task unlike the general-purpose computer which can do multiple tasks at once [10].

2.6 Arduino Microcontroller

Arduino is an open source electronics prototyping platform composed of a microcontroller, a programming language, and an IDE. Arduino is a tool for making interactive applications, designed to simplify this task for beginners but still flexible enough for experts to develop complex projects. It has a number of connection sockets that can be wired up to external electronics, such as motors, relays, light sensors, laser diodes, loudspeakers, microphones, etc. They can either be powered through the USB connection from the computer or from a 9V battery. They can be controlled from the computer or programmed by the computer and then disconnected and allowed to work independently [11].

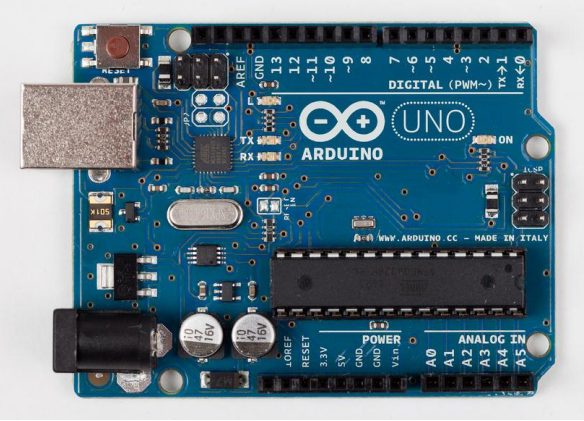


Figure 2.18: Arduino UNO

Arduino came up with an easy-to-learn programming language (derived from C++) that incorporates various complex programming functions into simple commands. The Arduino got its start at the Interaction Design Institute in the city of Ivrea, Italy, in 2005. Professor Massimo Banzi was looking for a low-cost way to make it easier for the design students there to work with technology. He discussed his problem with David Cuartielles, a researcher visiting from Malmö University in Sweden who was looking for a similar solution, and Arduino was born. A typical Arduino Uno is shown in Figure 2.18 [12].

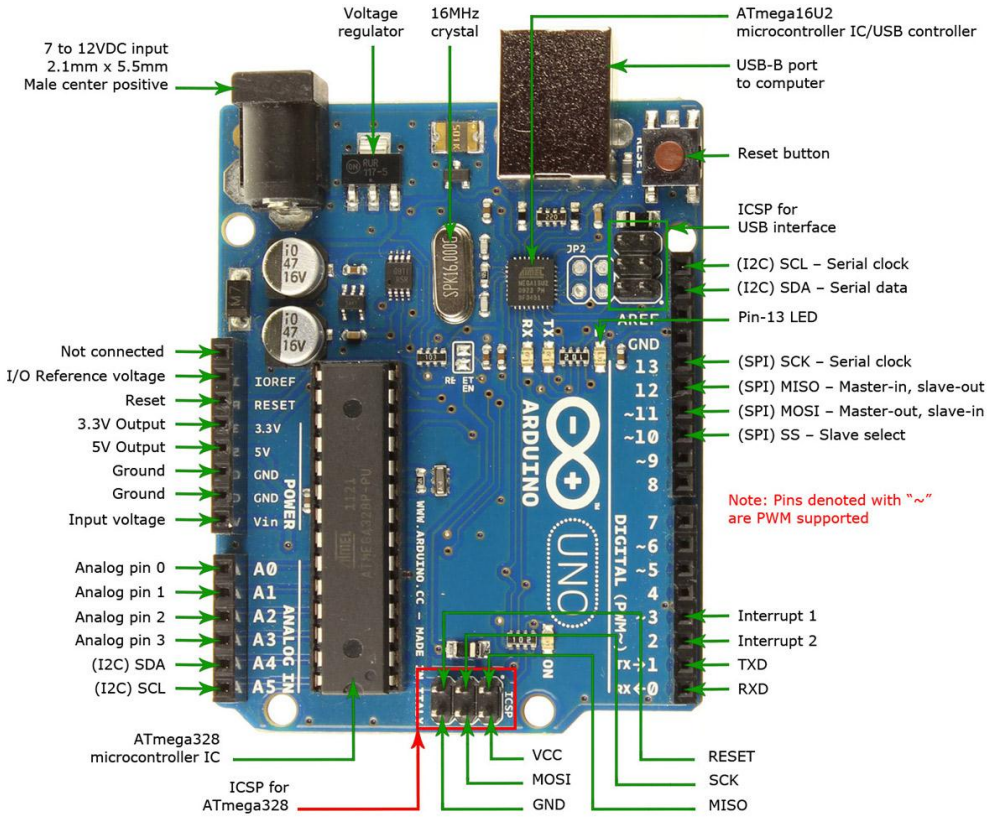


Figure 2.19: Arduino parts

The current revision of the Arduino board is known as the Arduino Uno. This board is based on the ATmega328 microcontroller. It has fourteen digital

input/output pins, six of which can be used as Pulse Width Modulation (PWM) outputs, along with six more analog input pins. Figure 2.19 illustrates the parts of the Arduino Uno. Table 2.1contains the technical specifications of the Arduino board [13].

2.6.1 Atmel AVR ATmega328

This microcontroller comes from a company called Atmel and the chip is known as an AVR. It is slow in modern terms, running at only 16 MHz with an 8-bit core, and has a very limited amount of available memory, with 32 kilobytes of storage and 2 kilobytes of random access memory. Still it's more than enough to handle a lot of useful projects [13].

2.6.2 Powering the Board

The Arduino Uno can be powered via the USB connection or with an external power supply. The board can operate on an external supply of 6 to 20 volts. The easiest option is to use a 9V battery or power adaptor, as these are commonly available. This layout is shown in Figure 2.20 [13].

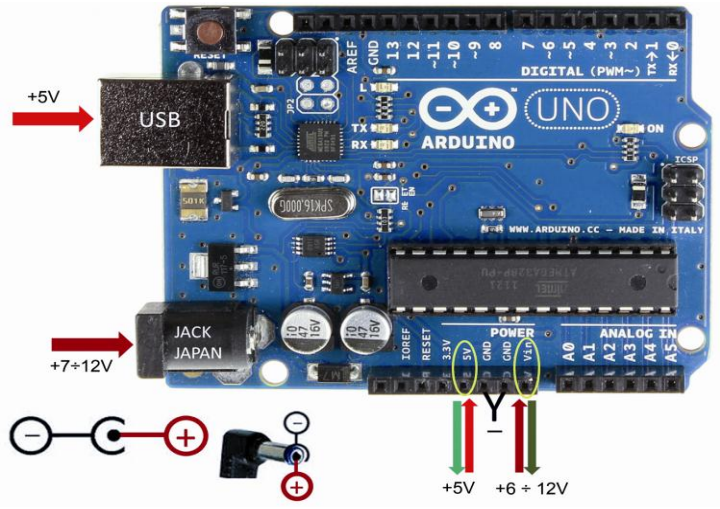


Figure 2.20: Arduino power configurations

2.6.3 Inputs and Outputs

A. Digital pins:

Each of the 14 pins on the Uno can be used as an input or output. They operate at 5 V with a maximum current of 40 mA. Each pin also has an internal pull-up resistor of 20–50 kOhms, although this is disconnected by default. Some pins have specialized functions. Pin 0 and 1 can be used to Receive (RX) and Transmit (TX) Transistor Transistor Logic (TTL) serial data. These pins are connected to the corresponding pins of the ATmega8U2 and hence to the USB Serial connection to your PC. Figure 2.21 shows these pins [13].

Figure 2.21: Arduino digital pins

B. Analog Pins

The Arduino has 6 analog pins. These pins are by default inputs, and can take up to 5 v which is divided to 1024 steps. Sensors which give an analog readings are connected to these pins such as, temperature sensors, light-intensity sensors, etc. Figure 2.22 shows these pins on the board [13].

Figure 2.22: Arduino analog pins

2.6.4 Communicating with the Board:

The ATmega328 provides UART TTL serial communication at 5 V, which is available on digital pins 0 (RX) and 1 (TX). The Arduino Uno has an ATmega8U2 chip on board that redirects this serial communication over USB, allowing the Arduino to appear as a virtual serial port to software on your PC [13].

2.6.5 Arduino Software

Arduino C is used to program the Arduino this language is derived from the well-known programming language the C++ and another language called Processing. The Arduino IDE in Figure 2.23 is equipped with a Serial Monitor which allows for debugging and communication with the Arduino board while it's working on real-time, Figure 2.24 shows the Serial Monitor. The Serial Monitor can transmit and receive data with several baud rates up to 115200 bits-per-second [13].

2.7 Lithium Polymer (LiPo) Battery

Lithium Polymer batteries are one of the newer battery types used for their high power to weight ratio. With a typical cell voltage of 3-7v, these batteries are lightweight yet powerful and are able to deliver large amounts of current very quickly. LiPo batteries have recently become much more affordable, making them a viable option for many robotic projects, though proper charging and

discharging is required to prevent overheating. They are typically arranged in series packs with up to six cells, totaling 22.2v, Figure 2.25 shows different sizes of LiPo Batteries [14].

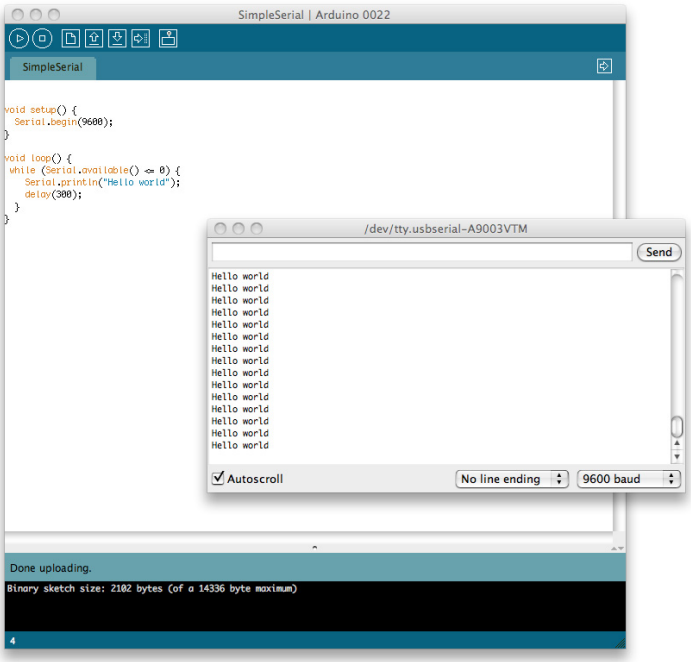


Figure 2.23: Arduino IDE

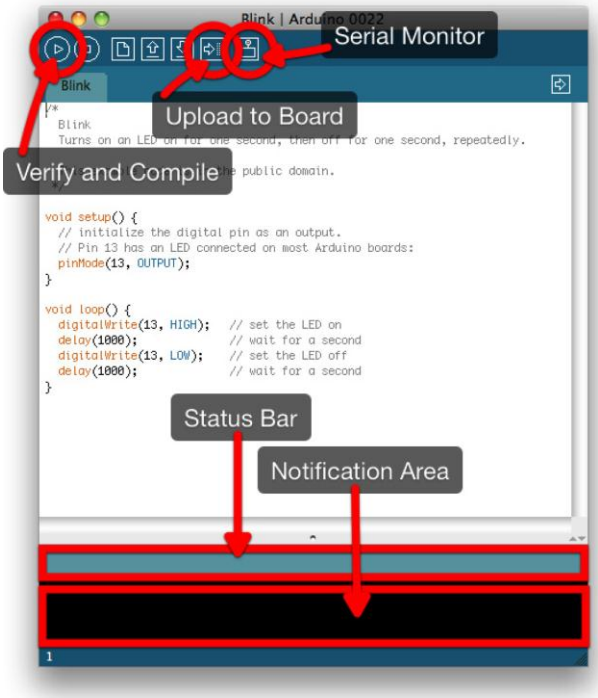


Figure 2.24: Serial monitor

Figure 2.25: Different LiPo battery sizes

2.8 Electric Motor

An electric motor is a machine used to convert electrical energy into mechanical energy in the form of a rotational motion using a carefully arranged set of magnets and coil windings. Its action is based on the principle that when a current carrying-conductor is placed in a magnetic field it experiences a mechanical force whose direction is given by Fleming's left hand rule. When its armature

conductors are supplied with current from the supply mains, they experience a force tending to rotate the armature. Electric motors have two main types, Direct Current (DC) or Alternative Current (AC), according to the supply current it designed to work on. A permanent magnet brushed DC motor have brushes that physically touch a set of spinning electrical contacts, called commutators that are electrically connected to the armature coil winding, commutators and metal or carbon brushes transfer energy from the supply to the rotating armature coils. The typical permanent magnet DC motor has only one armature coil with two wires for operation, it have two magnets attached to the inside of the motor casing and the armature and commutator mounted to the output shaft the brushes are typically spring loaded to keep them securely mated to the commutator contacts while spinning. A motor typically has an output shaft attached to the armature, for mounting a wheel or gear on the end. Figure 2.26 shows a small permanent magnet DC motor [14].

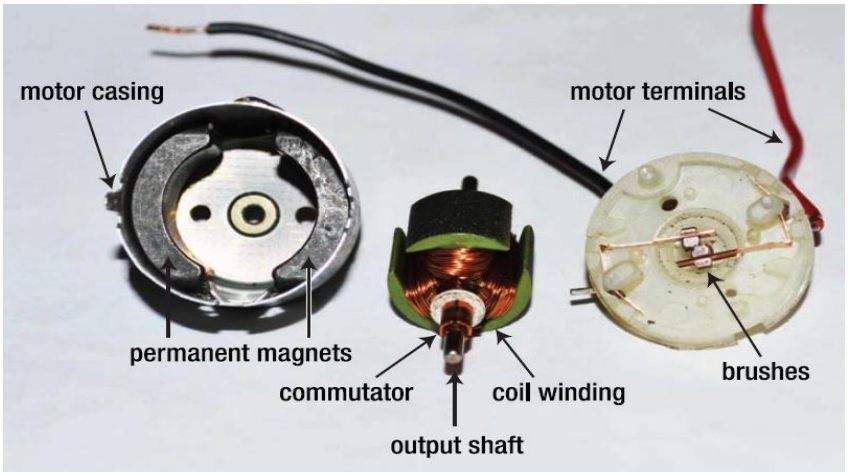


Figure 2.26: Disassembled permanent magnet DC motor

CHAPTER THREE

SYSTEM MODELING AND DESIGN

3.1 System Dynamics

The precise and effective modelling of rolling contact between the two bodies (including longitudinal slip) is a rather difficult task. However, in this case, the slipping is neglected and we can use a simple solution shown in Figure 3.1. Rolling is reduced to translation along the x axis.

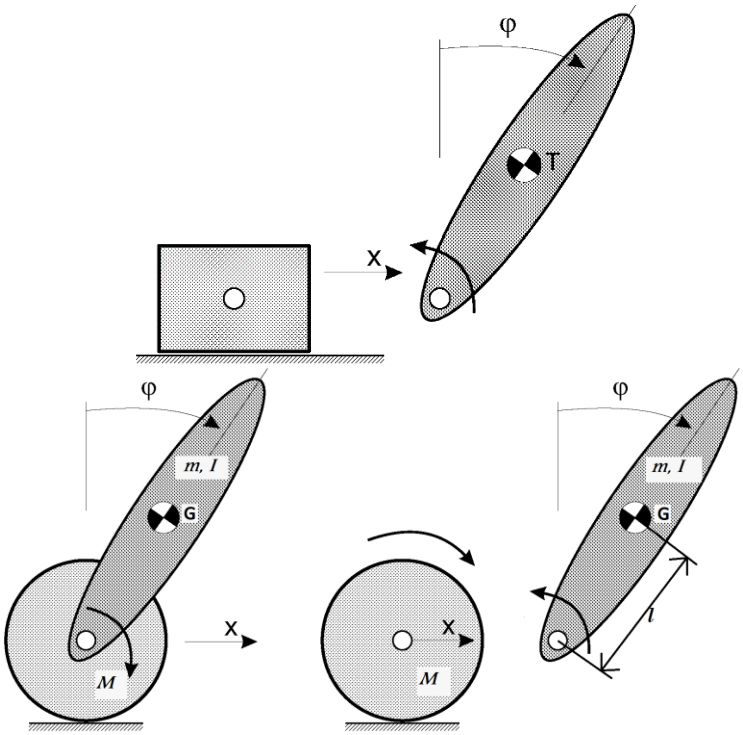


Figure 3.1: Movement of the robot

Using this configuration the problem of the self-balancing robot becomes exactly analogous to the inverted pendulum problem, i.e. the rod represents the robot structure and the cart represents the wheels. Thus, they can be analyzed similarly. A pendulum rod is free to oscillate around a fixed pivot point attached to a cart which is constrained to move in the horizontal movement. The rod is placed in the

upright vertical position, which is an unstable equilibrium point. The control objective is to apply a force to move the cart so that the pendulum remains in the vertical unstable position. The system of interest is shown in Figure 3.2, where F is the force in newton, m is the mass of the pendulum rod in kilograms, M is the mass of the moving cart in kilograms, F_v is the force applied to the cart in newton, F_f is the force due to friction in newton, g is the acceleration due to gravity in m/s^2, and θ is the angle of the inverted pendulum measured from the vertical y-axis in radians.

3.2 System Mathematical Model

Consider the free body diagrams shown in Figure 3.2. Furthermore, assume that the co-ordinates of the centroid (center of gravity) of the pendulum(x_G, y_G), are given by

$$x_G = x + l.sin\theta \quad (3.1)$$

$$y_G = l.cos\theta \quad (3.2)$$

Where l is the distance along the pendulum to the center of gravity and x is the x-co-ordinate of the cart's position.

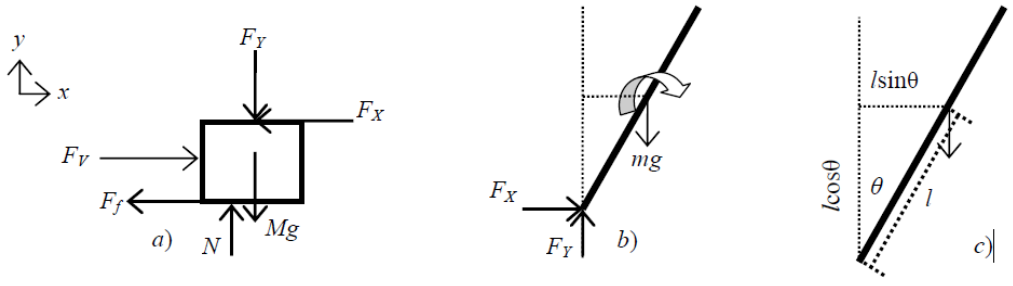


Figure 3.2: Free body diagram

For the horizontal motion of the cart, Newton's second law of motion:

$$\sum F = M\ \frac{d^2x}{dt^2}$$

(3.3)

Can be written as:

$$M\ \frac{d^2x}{dt^2} = F_v - F_x - F_f$$

(3.4)

Assume that the friction force can be written as

$$F_f = \gamma\ \frac{dx}{dt} \tag{3.5}$$

The horizontal motion of the pendulum can be written as

$$F_x = m\ \frac{d^2x_G}{dt^2} \tag{3.6}$$

The derivative on the right in Equation (3.6) can be simplified by determining the derivative of x_G using Equation (3.1).The first derivative can be found as follows:

$$\frac{dx_G}{dt} = \frac{d(x+l\sin\theta)}{dt} = \frac{dx}{dt} + l\frac{d(\sin\theta)}{dt} = \frac{dx}{dt} + l\cos\theta\frac{d(\theta)}{dt}$$

(3.7)

The second order derivative can be found by differentiating Equation (3.7), that is,

$$\frac{d^2x_G}{dt^2} = \frac{d}{dt}\left(\frac{dx}{dt} + l\cos\theta\frac{d\theta}{dt}\right) = \frac{d^2x}{dt} + l\frac{d}{dt}\left(\cos\theta\frac{d\theta}{dt}\right)$$

$$= \frac{d^2x}{dt} + l\left(\frac{d\cos\theta}{dt}\frac{d\theta}{dt} + \cos\theta\frac{d^2\theta}{dt^2}\right) = \frac{d^2x}{dt} + l\left(-\sin\theta\left(\frac{d\theta}{dt}\right)^2 + \cos\theta\frac{d^2\theta}{dt^2}\right)$$

$$= \frac{d^2x}{dt} - l\sin\theta\left(\frac{d\theta}{dt}\right)^2 + l\cos\theta\frac{d^2\theta}{dt^2}$$

(3.8)

Combining Equation (3.8) with Equation (3.6) gives:

$$F_x = m\left(\frac{d^2x}{dt^2} - l\sin\theta\left(\frac{d\theta}{dt}\right)^2\right)$$

(3.9)

Using Equation (3.9), Equation (3.4) can be simplified to give:

$$M\frac{d^2x}{dt^2} = F_v - m\left(\frac{d^2x}{dt^2} - l\sin\theta\left(\frac{d\theta}{dt}\right)^2 + l\cos\theta\frac{d^2\theta}{dt^2}\right) - \gamma\frac{dx}{dt}$$

$$M\frac{d^2x}{dt^2} = F_v - m\frac{d^2x}{dt^2} - ml\sin\theta\left(\frac{d\theta}{dt}\right)^2 + ml\cos\theta\frac{d^2\theta}{dt^2} - \gamma\frac{dx}{dt} \quad (3.10)$$

The final form for the horizontal motion of the card can be given as:

$$(M+m)\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} = F_v + ml\sin\theta\left(\frac{d\theta}{dt}\right)^2 - ml\cos\theta\frac{d^2\theta}{dt^2} \quad (3.11)$$

For the vertical motion of the pendulum, Equation (3.2) can be written as:

$$F_y - mg = m\frac{d^2y_G}{dt^2}$$

(3.12)

Similarly to the horizontal case, the derivative on the right in Equation (3.12) can be written as follows:

$$\frac{dy_G}{dt} = \frac{d(l\cos\theta)}{dt} = -l\sin\theta\frac{d\theta}{dt}$$

(3.13)

$$\frac{d^2y_G}{dt^2} = \frac{d}{dt}\left(-l\sin\theta\frac{d\theta}{dt}\right) = -l\left(\frac{d\sin\theta}{dt}\frac{d\theta}{dt} + \sin\theta\frac{d^2\theta}{dt^2}\right)$$

$$= -l\left(\cos\theta\left(\frac{d\theta}{dt}\right)^2 + \sin\theta\frac{d^2\theta}{dt^2}\right) = -l\cos\theta\left(\frac{d\theta}{dt}\right)^2 - l\sin\theta\frac{d^2\theta}{dt^2} \quad (3.14)$$

Using Equation (3.14), Equation (3.12) can be rewritten to give:

$$F_y - mg = m\left(-l\cos\theta\left(\frac{d\theta}{dt}\right)^2 - l\sin\theta\frac{d^2\theta}{dt^2}\right) \quad (3.15)$$

Thus, the vertical reaction force, F_y , can be written as:

$$F_y = mg + m\left(-l\cos\theta\left(\frac{d\theta}{dt}\right)^2 - l\sin\theta\frac{d^2\theta}{dt^2}\right)$$

(3.16)

For any object, the relationship between the moment applied on an object and its angular acceleration is given by the following relationship:

$$\sum \bar{M} = I\frac{d^2\theta}{dt^2}$$

(3.17)

Where M□ is the moment due to a given force and defined as:

$$\bar{M} = \bar{F} \times \bar{r}$$

(3.18)

where is the force vector, r is the position vector of the object with respect to the point about which the moments are being summed, and I is the angular momentum of the object. For the pendulum, summing the moment around its center of gravity, Equation (3.17) can be written as:

$$F_y l\sin\theta - F_x l\cos\theta = I\frac{d^2\theta}{dt^2}$$

(3.19)

Substituting Equation (3.16) for F_y and Equation (3.9) for F_x into Equation (3.19) gives:

$$\left(mg + m\left(-l\cos\theta\left(\frac{d\theta}{dt}\right)^2 - l\sin\theta\frac{d^2\theta}{dt^2}\right)\right)l\sin\theta -$$

$$\left(m\left(\frac{d^2x}{dt^2} - l\sin\theta\left(\frac{d\theta}{dt}\right)^2 + l\cos\theta\frac{d^2\theta}{dt^2}\right)\right)l\cos\theta = I\frac{d^2\theta}{dt^2}$$

(3.20)

Simplifying Equation (3.20) gives:

$$mgl\sin\theta - ml^2\sin\theta\cos\theta\left(\frac{d\theta}{dt}\right)^2 - ml^2\sin^2\theta\frac{d^2\theta}{dt^2} - ml\cos\theta\frac{d^2x}{dt^2} +$$

$$ml^2\sin\theta\cos\theta\left(\frac{d\theta}{dt}\right)^2 - ml^2\cos^2\theta\frac{d^2\theta}{dt^2}l\cos\theta = I\frac{d^2\theta}{dt^2}$$

$$mgl\,sin\,\theta - ml^2(\sin^2\theta + \cos^2\theta)\frac{d^2\theta}{dt^2} - ml\cos\theta\frac{d^2x}{dt^2} = I\frac{d^2\theta}{dt^2}$$

(3.21)

Since:

$$\cos^2\theta + \sin^2\theta = 1$$

(3.22)

Equation (3.21) can be simplified to give:

$$mgl\sin\theta - ml^2\frac{d^2\theta}{dt^2} - ml\cos\theta\frac{d^2x}{dt^2} = I\frac{d^2\theta}{dt^2}$$

$$mgl\sin\theta - ml\cos\theta\frac{d^2x}{dt^2} = (I + ml^2)\frac{d^2\theta}{dt^2}$$

(3.23)

Thus, the final equation for the angular position is given as:

$$(I + ml^2)\frac{d^2\theta}{dt^2} = mgl\sin\theta - ml\cos\theta\frac{d^2x}{dt^2}$$

(3.24)

Therefore the equations of motion for the inverted pendulum on a moving cart can be written as:

$$\begin{cases}(M+m)\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} = F_v + ml\sin\theta\left(\frac{d\theta}{dt}\right)^2 - ml\cos\theta\frac{d^2\theta}{dt^2} \\ (I + ml^2)\frac{d^2\theta}{dt^2} = mgl\sin\theta - ml\cos\theta\frac{d^2x}{dt^2}\end{cases} \quad (3.25)$$

3.3 Linearized Model of the System

The model of the system given by Equation (3.25) is nonlinear and must be linearized in order to obtain a reasonable model for control purposes. Linearization will be performed about the point x = 0 m and θ = 0 radians. Furthermore, it will be assumed that since θ is small (This is justifiable when controlling an object as it should not deviate greatly from the assumed steady-state value), also by neglecting the friction constant we get

$$\sin\theta \approx \theta$$

$$\cos\theta \approx 1$$

$$\left(\frac{d\theta}{dt}\right)^2 \approx 0$$

$$\gamma = 0 \tag{3.26}$$

Under these assumptions, Equation (3.25) can be rewritten as:

$$\begin{cases} (M+m)\frac{d^2x}{dt^2} = F_v - ml\frac{d^2\theta}{dt^2} \\ (I+ml^2)\frac{d^2\theta}{dt^2} = mgl\theta - ml\frac{d^2x}{dt^2} \end{cases} \tag{3.27}$$

Using the Laplace transform:

$$F_v(s) = (M+m)s^2X(s) + mls^2\theta(s) \tag{3.28}$$

$$0 = mls^2X(s) + (I+ml^2)s^2\theta(s) - mgl\theta(s)$$

(3.29)

$$\frac{\theta(s)}{F_v(s)} = \frac{1}{[ml - \frac{(M+m)(I+ml^2)}{ml}]s^2 + (M+m)g} \tag{3.30}$$

3.4 Permanent Magnet DC Motor Dynamics

A motor is an electromechanical component that yields a torque output for a voltage input, that is, a mechanical output generated by an electrical input. The

mathematical model of the DC PM motor is derived and the transfer function is evaluated. The schematic of the DC motor is shown in the following diagram.

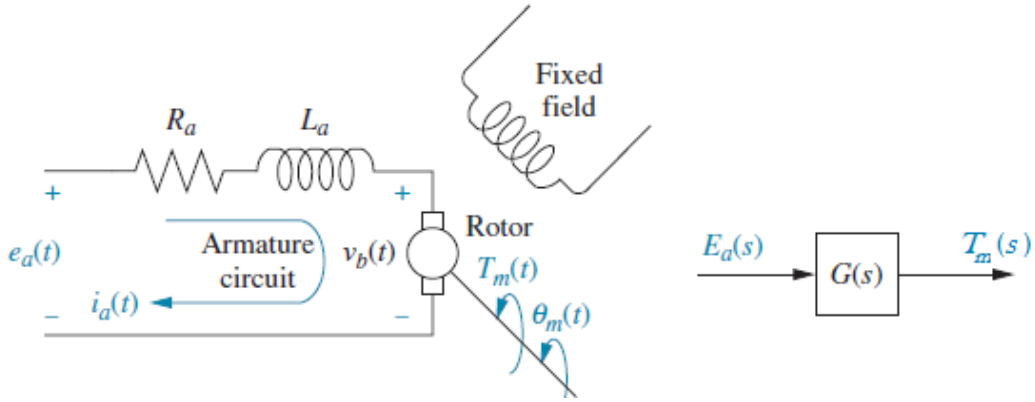


Figure 3.3: Schematic and block diagrams of the DC motor

In figure 3.3, a magnetic field is developed by stationary permanent magnets. A rotating circuit called the armature, through which current $i_a(t)$ flows, passes through this magnetic field at right angles and feels a force

$$F_m = Bl_c i_a(t)$$

(3.31)

Where B is the magnetic field strength and l_c is the length of the conductor. The resulting torque turns the rotor, the rotating member of the motor.

There is another phenomenon that occurs in the motor: A conductor moving at right angles to a magnetic field generates a voltage at the terminals of the conductor equals

$$e_g = Bl_c v$$

(3.32)

Where e_g is the voltage and v is the velocity of the conductor normal to the magnetic field. Since the current-carrying armature is rotating in a magnetic field, its voltage is proportional to speed. Thus,

$$v_b(t) = K_b \frac{d\theta_m(t)}{dt}$$

(3.33)

We call $v_b(t)$ the back electromotive force (back emf), K_b is a constant of proportionality called the back emf constant; and $d\theta_m(t)/dt = \omega_m$ is the angular velocity of the motor. Taking the Laplace transform, we get

$$V_b(s) = K_b s\theta_m(s)$$

(3.34)

The relationship between the armature current $i_a(t)$, the applied armature voltage $e_a(t)$ and the back emf $v_b(t)$ is found by writing a loop equation around the Laplace transformed armature circuit in figure 3.3.

$$R_a I_a(s) + L_a s I_a(s) + V_b(s) = E_a(s)$$

(3.35)

The torque developed by the motor is proportional to the armature current; thus

$$T_m(s) = K_t I_a(s)$$

(3.36)

Where T_m is the torque developed by the motor and K_t is a constant of proportionality, called the motor torque constant, which depends on the motor and magnetic field characteristics. In a consistent set of units, the value of K_t is equal to the value of K_b. Rearranging equation (3.36) yields

$$I_a(s) = \frac{1}{K_t} T_m(s) \qquad (3.37)$$

To find the transfer function of the motor, we first substitute equations (3.34) and (3.37) into (3.35), yielding

$$\frac{(R_a + L_a s)T_m(s)}{K_t} + K_b s\theta_m(s) = E_a(s) \qquad (3.38)$$

Now we must find $\theta_m(s)$ in terms of $T_m(s)$ if we are to separate the input and output variables and obtain the transfer function $T_m(s)/E_a(s)$.

Figure 2.36 shows a typical equivalent mechanical loading on a motor. J_m is the equivalent inertia at the armature and includes the armature inertia and might also include, the load inertia reflected to the armature.

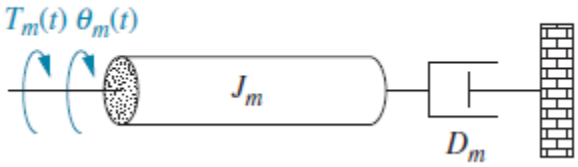


Figure 3.4: A typical equivalent mechanical loading on the motor

D_m is the equivalent viscous damping at the armature and includes both the armature viscous damping and might also include the load viscous damping reflected to the armature.

$$T_m(s) = (J_m s^2 + D_m s)\theta_m(s) \tag{3.39}$$

Rearranging equation (3.39) yields

$$\theta_m(s) = \frac{T_m(s)}{(J_m s^2 + D_m s)} \tag{3.40}$$

Substituting equation (3.40) in Eq. (3.38) we get

$$\left(\frac{(R_a + L_a s)}{K_t} + K_b s \left[\frac{1}{(J_m s^2 + D_m s)}\right]\right) T_m(s) = E_a(s) \tag{3.41}$$

Eventually we get

$$\frac{T_m(s)}{E_a(s)} = \frac{\frac{K_t}{J_m L_a}(J_m s + D_m)}{s^2 + \frac{(J_m R_a + D_m L_a)s}{J_m L_a} + \frac{(D_m R_a + K_t K_b)}{J_m L_a}} \tag{3.42}$$

By neglecting D_m we get [4].

$$\frac{T_m(s)}{E_a(s)} = \frac{\frac{K_t}{L_a}.s}{s^2+\frac{(R_a)s}{L_a}+\frac{(K_tK_b)}{J_mL_a}}$$

(3.43)

3.5 System Controller Design

To simulate the impulse response of the transfer function, SIMULINK is used. The tuning of the PID controller was done manually using SIMULINK PID Controller block, as shown in Figure 3.5:

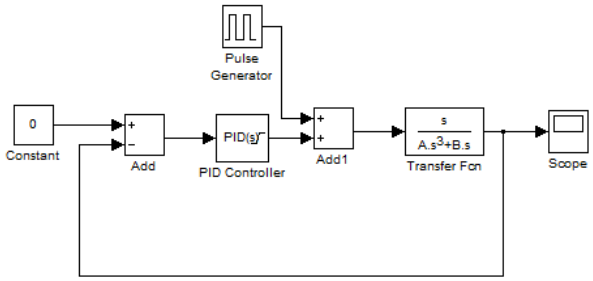


Figure 3.5: SIMULINK block diagram of the system

Manual tuning of the gain settings is the simplest method for setting the PID controls. However, this procedure is done actively (the PID controller turned on and properly attached to the system) and requires some amount of experience to fully integrate. To tune PID controller manually, first the integral and derivative gains are set to zero. Increase the proportional gain until observing oscillation in the output. After the proportional gain is set, the derivative gain can then be increased. Derivative gain will reduce overshoot and damp the system quickly to the set point value or near it. If the derivative gain increased too much, large overshoot will be seen. Once the derivative gain is set, increase the integral gain until any offset is corrected for on a time scale appropriate for the system. If the gain increased too much, significant overshoot of the set point value and instability in the circuit will be observed.

Table 3.1: Manual tuning of PID controller

No. experiment	Kp	Ki	Kd
0	0	0	0
1	5	0	0
2	10	0	0
3	15	0	0
4	20	0	0
5	25	0	0
6	30	0	0
7	35	0	0.2
8	35	0	0.6
9	35	0	1
10	40	0	1
11	40	0.4	1
12	40	0.8	1
13	40	1	1

The best parameters that gives the best response in real $Kp = 40, Ki = 1, Kd = 1$.

CHAPTER FOUR

SYSTEM IMPLEMENTATION AND EXPEREMENTAL RESULTS

4.1 System Practical Model

The system's practical model is divided into two parts; the mechanical part which consists of wooden shelves, Teflon wheels, iron poles and nuts, and the electrical part which consists of DC geared motor, LiPo battery, Arduino microcontroller, IMU sensor, and Motor Driver.

4.1.1 Mechanical part

The mechanical part of the robot consists of the body and two wheels as shown in Figure 4.1. The material used to make the body is wood. It was cut into three shelves using CNC machine. The shelves were connected together using iron poles and nuts. Zippers were used to fasten the electrical parts to the shelves. The wheels were made of a material called Teflon, it is a white solid material similar to plastic in shape and density which is easy to machine. The wheels were cut by a lathe with an 80mm diameter. The dimensions of the physical model is shown in Figure 4.1.

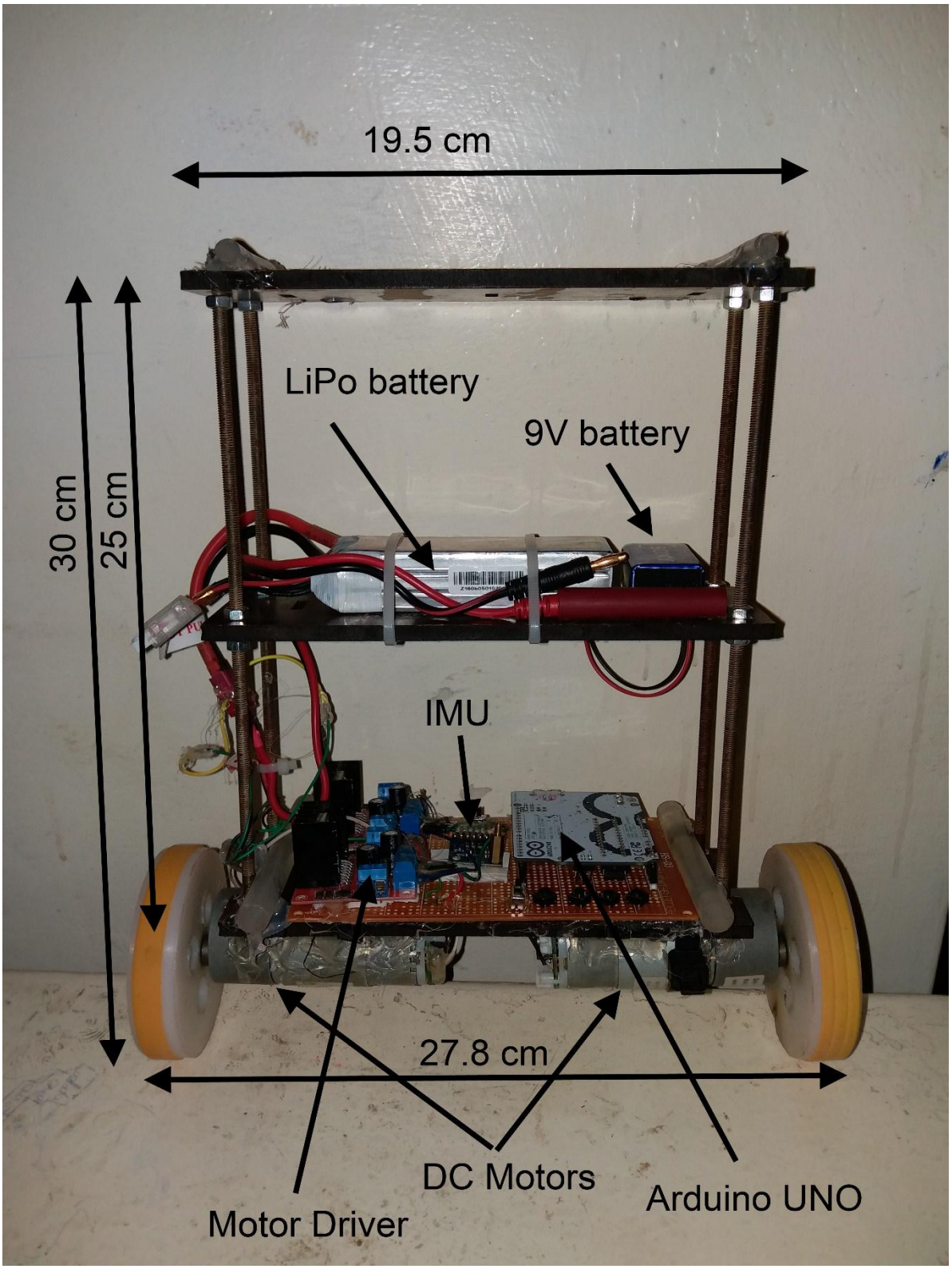


Figure 4.1: Mechanical parts and dimensions

4.1.2 Electrical part

The electrical part of the robot consists of DC geared motor with encoder, LiPo battery, Arduino microcontroller, IMU, and Motor Driver.

A. DC Geared Motor

The motors used for this robot are normal 12 volts DC motors. They are coupled with a gear box to reduce their speed and increase the torque. The encoder is mounted on the shaft of the motor to measure the speed and direction of the motor shaft. The specifications of the motor is listed in Table 4.1. The encoder connectors are hall sensor V_{CC}, hall sensor GND, hall sensor A V_{out}, and Hall sensor B V_{out}. It works on voltage from 4.5V to 24V with 20mA.

Table 4.1: DC motor specifications

Rated voltage	12VDC
No load speed	300r/min
No load current	140mA
Rated torque	190gf.cm 15.7mN.m
Rated current	800mA
Rated speed	2200r/min
Stall torque	640gf.cm 62.7mN.m
Stall current	3000mA

B. Lithium Polymer Battery

The battery used to power the robot as shown in Figure 4.5 is a Lithium-Polymer (LiPo) battery, which is a rechargeable battery. Table 4.2 lists the battery specifications.

Table 4.2: LiPo battery specifications

Capacity	2200mAh
Voltage	3S/11.1V
Discharge Rate	25C
Max Cont. Current	55A
Max Burst Current	110A
Weight	184g
Size	106*34*24mm

C. Arduino UNO

The Arduino UNO is a microcontroller board based on ATmega328P microcontroller, Table 4.3 contains the technical specifications of the Arduino board.

Table 4.3: Arduino board specifications

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (Recommended)	7 – 12 V
Input Voltage (Limits)	6 – 20 V
Digital I/O Pins	14 (6 provide PWM)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3 V Pin	50 mA

D. The Inertia Measurement Unit:

The IMU is used to measure the tilted angle of the robot. The pin (AD0) in the Arduino selects between I^2C address (0x68) and (0x69). That makes it possible to have two of these sensors in a project. The MPU6050 chip needs 3.3V but a voltage regulator on the GY-521 board allows to give it up to 5V. The gyro module communicates with the Arduino through I^2C serial communication via the Serial Clock (SCL) and Serial Data (SDA).

E. Motor Driver

Figure 4.2 shows the pins of the motor driver break-out board, the L298n is an Integrated Circuit (IC) used to derive motors, relays and any inductive load, the IC is soldered to a break-out board with screw terminals for easy usage.

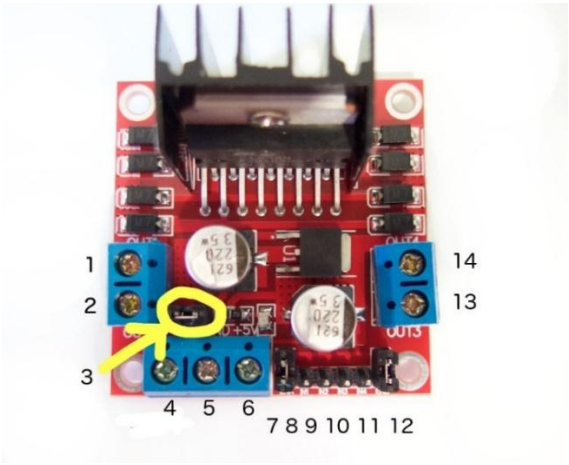


Figure 4.2: Motor driver

1. DC motor 1 "+" or stepper motor A+
2. DC motor 1 "-" or stepper motor A-
3. 12V jumper - remove this if using a supply voltage greater than 12V DC. This enables power to the onboard 5V regulator.
4. Connect your motor supply voltage here, maximum of 35V DC. Remove 12V jumper if >12V DC
5. GND
6. 5V output if 12V jumper in place, ideal for powering your Arduino (etc.)
7. DC motor 1 enable jumper. Leave this in place when using a stepper motor. Connect to PWM output for DC motor speed control
8. IN1
9. IN2
10. IN3
11. IN4
12. DC motor 2 enable jumper. Leave this in place when using a stepper motor. Connect to PWM output for DC motor speed control.
13. DC motor 2 "+" or stepper motor B+
14. DC motor 2 "-" or stepper motor B-

F. Electrical System Connections

IMU is connected to pins A5 (SCL) and pin A4 (SDA) via I^2C protocol, the signals from both gyroscope and accelerometer are fused together by using a complementary filter as shown in Figure 4.3. The desired ratio of gyro/accelerometer readings was found to be:

$$angle = 0.93 * gyro * dt + 0.07 * acceleometer \quad (4.1)$$

The two geared DC motors are connected to a driver which is connected to pins 6,8 and 9 for right motor and pins 10,11 12 for left motor. The difference between the angle and the desired set point angle is the PID input and its output is send to the motors via the drivers as a PWM signals.

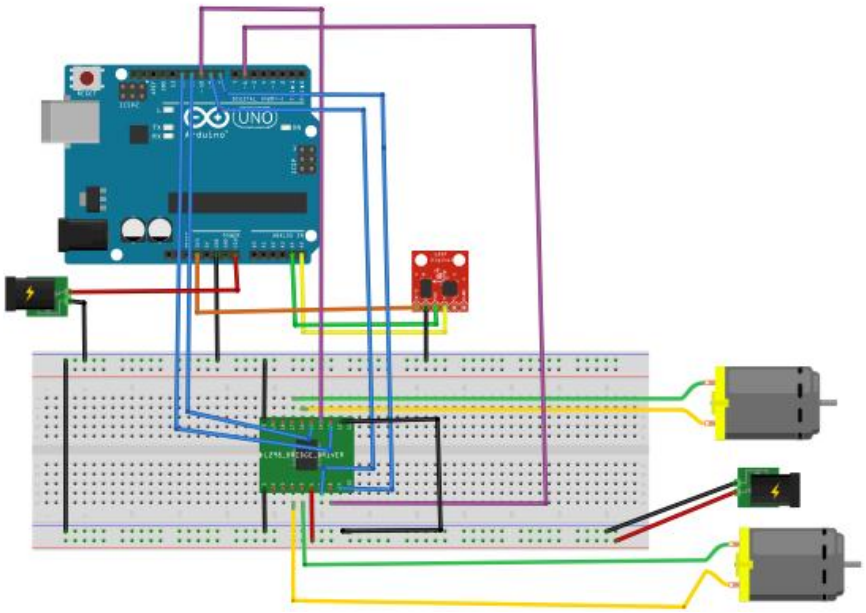


Figure 4.3: Electrical system connections

4.2 Real Time Plotting Results

This section demonstrates the results of a real time plotting using MATLAB (See Appendix B) and Arduino of two-wheeled self-balance robot system using design of PID controller with manual tuning method. These plots represents the system's behavior as the parameters of the PID controller (K_p, K_i and K_d) varies.

When substituting the values of the controller with $K_p = 0, K_i = 0, K_d = 0$ the real time plotting as shows in Figure 4.4. The robot tries to balance but without use. The system is unstable and has poor characteristics,

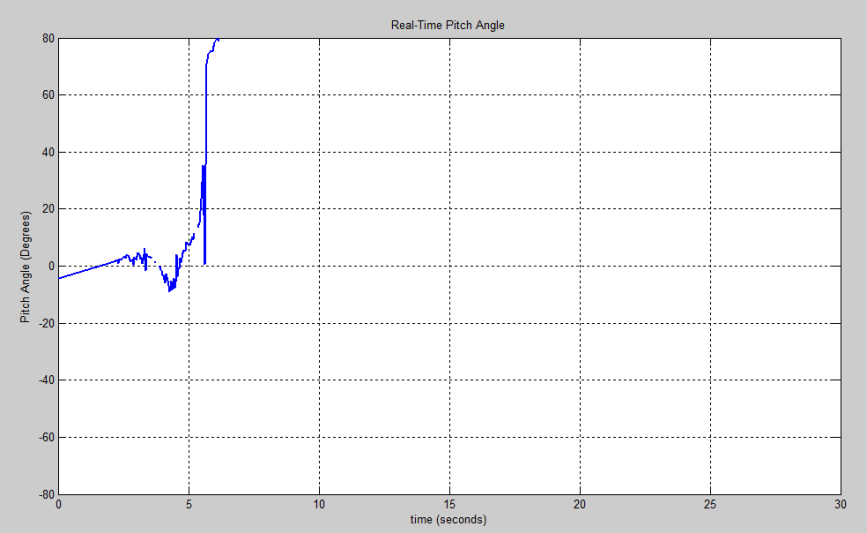


Figure 4.4: System response with $K_p = 1, K_i = 0, K_d = 0$

By substituting the values of the controller with $K_p = 30, K_i = 0, K_d = 0.5$ the real time plotting as shows in Figure 4.5. The system starts to oscillate but still fall over and not reaching the desired response.

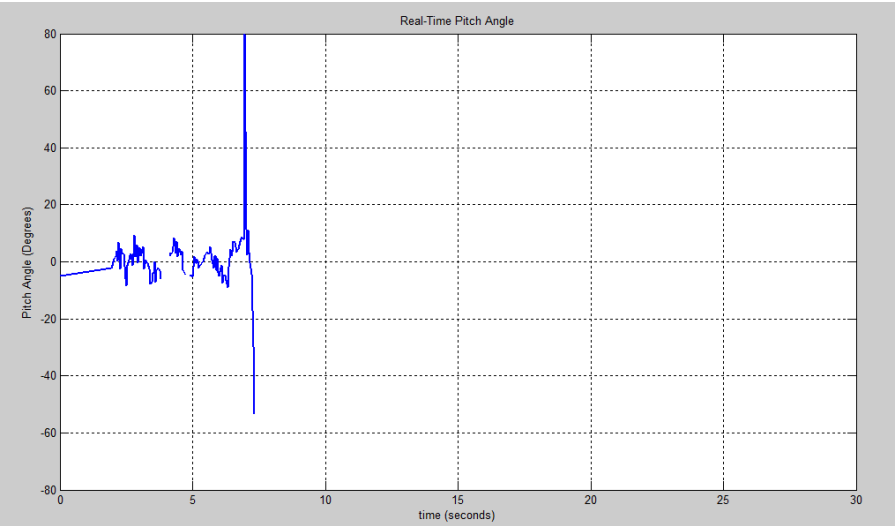


Figure 4.5: System response with $K_p = 30, K_i = 0, K_d = 0.5$

By substituting the values of the controller with $K_p = 40, K_i = 1, K_d = 1$ the real time plotting as shows in Figure 4.6. The robot oscillates very fast, but the robot doesn't fall, unless it was pushed hard.

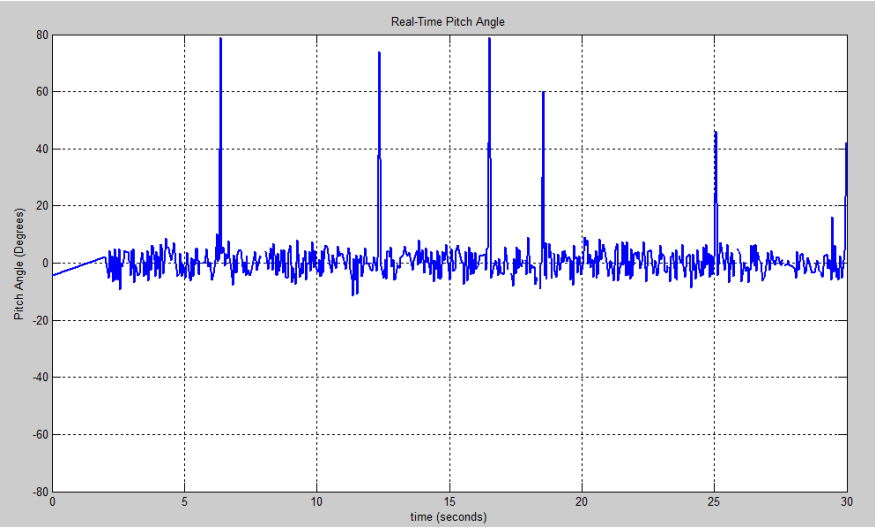


Figure 4.6: System response with $K_p = 40, K_i = 1, K_d = 1$

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

A mathematical model of the two-wheeled self-balance robot system was developed by using physical and electrical laws. A simplified mathematical model was derived by system parameters after linearizing the model. The controller parameters values (K_p, K_i, K_d) were obtained by using manual tuning method from practical model so as to perform best system response. From experimental results, it is found that the best controller parameters which gave the best response of the system are: $K_p = 40, K_i = 1\ and\ K_d = 1$.

After observing the robot's practical experiments, it was noticed that there was high oscillation around the set point, and the robot never settles down. It was also noticed that when the robot passes a certain angle the system becomes unstable and can't regain its balance. The results obtained from the manual tuning were acceptable and as had been expected.

5.2 Recommendations

1- Use of fuzzy control or neural networks to adapt the PID parameters.
2- Design a linear quadratic regulator to maximize performance.
3- Use of wireless control.
4- Implementation of a Kalman filter to obtain smoother readings from the IMU.

References:

1- Dogan Ibrahim, "Microcontroller Based Applied Digital Control", John Wiley & Sons, Ltd., West Sussex, England, 2006.

2- U. A. Bakshi, M. V. Bakshi, "Modern Control Theory", Technical Publication Pune, India, 2006.

3- KarlJ. Astrom And Tore Hagglund, "PID Controllers Theory Design and Tuning", USA, 2nd Edition, 1995.

4- Katsuhiko Ogata, "Modern Control Engineering", Prentice-Hall, Inc., New Jersey, USA, 2002.

5- Mrs.LEKSHMY.S, ALEESHA GEORGE, ATHIRA C.V, "Self Balancing Robot", International Journal of Computer Engineering In Research Trends, 2015.

6-Tomislav Tomašić, Andrea Demetlika, Mladen Crneković, " SELF-BALANCING MOBILE ROBOT TILTER", University of Zagreb, 2012.

7-Hau-Shiue Juang and Kai-Yew Lum, "Design and Control of a Two-Wheel Self-Balancing Robot using the Arduino Microcontroller Board", IEEE International Conference on Control and Automation (ICCA) Hangzhou, China, June 12-14, 2013.

8- Mikael Arvidsson Jonas Karlsson, "Design, construction and verification of a self-balancing vehicle", Chalmers University of Technology
Göteborg, Sweden, 2012.

9-Mahadi Hasan, Chanchal Saha, Md. Mostafizur Rahman, Md. Rabiual Islam Sarker and Subrata K. Aditya, "Balancing of an Inverted Pendulum Using PD Controller", Asian Institute of Technology, 2011.

10-Dogan Ibrahim," Advanced PIC Microcontroller Projects in C", Oxford, UK, 2008.

11- Simon Monk,"30 Arduino Projects for the Evil Genius**",** the McGraw-Hill Companies, USA, 2010.

12- MARTIN EVANS, JOSHUA NOBLE, JORDAN HOCHENBAUM, ”Arduino in Action”, Manning Publications Co., Shelter Island, NY, 2013.

13-Alasdair Allan, “iOS Sensor Apps with Arduino”, O’Reilly Media, Inc., USA, 2011

14- John-David Warren, Josh Adams, and Harald Molle, “Arduino Robotics”, New York, USA, 2011.

APPENDIX A

Arduino C code for two-wheeled self-balance robot system

```
#include <Wire.h>
#define RESTRICT_PITCH

//*************************************************************
// connect motor controller pins to Arduino digital pins
// Right motor
int PWMA = 6;
int ForwardR = 8;
int BackwardR = 9;
// Left motor
int PWMB = 10;
int ForwardL = 11;
int BackwardL = 12;
//*************************************************************

double accX, accY, accZ;
double gyroX, gyroY, gyroZ;

double gyroXangle, gyroYangle; // Angle calculate using the gyro only
double compAngleX, compAngleY; // Calculated angle using a complementary filter
uint32_t timer;

unsigned long dt;
```

```
unsigned long T=4000;
double pInput= 0;
double Input = 0;
double Output;
double Kp = 40;
double Ki = 1;
double Kd = 1;
double p, i = 0, d;
double angle;
double power;
uint8_t i2cData[14];
int A,B,C,D,ea=0,eb=0,ec=0,ed=0,last=1;

void setup() {
  Wire.begin();           //Start I2C as master

#if ARDUINO >= 157
  Wire.setClock(400000UL); // Set I2C frequency to 400kHz
#else
  TWBR = ((F_CPU / 400000UL) - 16) / 2; // Set I2C frequency to 400kHz
#endif

  i2cData[0] = 7; // Set the sample rate to 1000Hz - 8kHz/(7+1) = 1000Hz
  i2cData[1] = 0x00; // Disable FSYNC and set 260 Hz Acc filtering, 256 Hz Gyro filtering, 8 KHz sampling
  i2cData[2] = 0x00; // Set Gyro Full Scale Range to ±250deg/s
  i2cData[3] = 0x11; // Set Accelerometer Full Scale Range to ±16g
  while (i2cWrite(0x19, i2cData, 4, false)); // Write to all four registers at once
```

```
  while (i2cWrite(0x6B, 0x01, true)); // PLL with X axis gyroscope reference and disable sleep mode
  Serial.begin(250000);
  //*********************************************
  // set all the motor control pins to outputs
  pinMode(PWMA, OUTPUT);
  pinMode(PWMB, OUTPUT);
  pinMode(ForwardR, OUTPUT);
  pinMode(BackwardR, OUTPUT);
  pinMode(ForwardR, OUTPUT);
  pinMode(BackwardR, OUTPUT);
  pinMode(2,INPUT);

  dt = micros() + T;
}

void loop(){
  // Manual tuning using push-buttons
 A=digitalRead(A0);
 B=digitalRead(A1);
 C=digitalRead(A2);
 D=digitalRead(2);

if(A)ea=1;
if(B)eb=1;
if(C)ec=1;
if(D)ed=1;
```

```
if(!A && ea)
{
ea=0;
Kp+=5;
last=1;
}
if(!B && eb)
{
eb=0;
Ki+=0.2;
last=2;
}
if(!C && ec)
{
ec=0;
Kd+=0.1;
last=3;
}
if(!D && ed)
{
ed=0;
if(last==1)Kp-=5;
if(last==2)Ki-=0.2;
if(last==3)Kd-=0.1;
}

angle = IMUReading()-5.055;   // Balance angle is 5.055 degree
```

```
                    // Subtracting it from IMUReading to get PID input
Serial.println(angle);

Input = angle;

d = (Input - pInput) / T ;    // Derivative term

i = i + Input * T;          // Integral term
i = constrain(i, -10, 10);

Output = Kp * Input + Kd * d + Ki * i;   // PID output
Output = constrain(Output, -255, 255);

pInput = Input;                    // Previous input for derivative term

 if(abs(Output) < 70)               // Motors won't start rotating until a certain
PWM is reached
 power = 0;                        // To make them more synchronised
 else power = abs(Output);

if(angle < -2 & angle > -40){        // Don't ballance beyond certain angles

  motor(1,2, power);                // Backward direction
  motor(2,2, power);
}
else if(angle > 2 & angle < 40){

  motor(1,1,power);                 // Forward direction
```

```
  motor(2,1,power);
}
else{
  motor(1,0,0);                    // Stop motors
  motor(2,0,0);
}

while(dt>micros());                // wait until the loop completes the sampling time

dt = micros()+T;

}



double IMUReading(){
  // Update all the values
  while (i2cRead(0x3B, i2cData, 14));
  accX = ((i2cData[0] << 8) | i2cData[1]);
  accY = ((i2cData[2] << 8) | i2cData[3]);
  accZ = ((i2cData[4] << 8) | i2cData[5]);

  gyroX = (i2cData[8] << 8) | i2cData[9];
  gyroY = (i2cData[10] << 8) | i2cData[11];
  gyroZ = (i2cData[12] << 8) | i2cData[13];

  double dt = (double)(millis() - timer) / 1000; // Calculate delta time
```

```
  timer = millis();


  #ifdef RESTRICT_PITCH
  double roll  = atan2(accY, accZ) * RAD_TO_DEG; // Converting the readings to degrees
  double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
#else
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif


  double gyroXrate = gyroX / 131.0; // Convert to deg/s
  double gyroYrate = gyroY / 131.0; // Convert to deg/s


gyroXangle += gyroXrate * dt; // Calculate gyro angle without any filter
gyroYangle += gyroYrate * dt;


compAngleX = 0.93 * (compAngleX + gyroXrate * dt) + 0.07 * roll; // Calculate the angle using a Complimentary filter
compAngleY = 0.93 * (compAngleY + gyroYrate * dt) + 0.07 * pitch;


#ifdef RESTRICT_PITCH
  // This fixes the transition problem when the accelerometer angle jumps between -180 and 180 degrees
  if ((roll < -90 && compAngleX > 90) || (roll > 90 && compAngleX < -90)) {


    compAngleX = roll;
```

```
    gyroXangle = roll;
  } else


  if (abs(compAngleX) > 90)
    gyroYrate = -gyroYrate; // Invert rate, so it fits the restriced accelerometer
reading


#else
  // This fixes the transition problem when the accelerometer angle jumps
between -180 and 180 degrees
  if ((pitch < -90 && compAngleY > 90) || (pitch > 90 && compAngleY < -90))
{


    compAngleY = pitch;


    gyroYangle = pitch;
  } else


  if (abs(compAngleY) > 90)
    gyroXrate = -gyroXrate; // Invert rate, so it fits the restriced accelerometer
reading


#endif


  // Reset the gyro angle when it has drifted too much
  if (gyroXangle < -180 || gyroXangle > 180)
    gyroXangle = compAngleX;
  if (gyroYangle < -180 || gyroYangle > 180)
```

```
    gyroYangle = compAngleY;

    return compAngleY;

}


void motor(int motor , int dir , int sped){
  // Motor controlling function
  switch(motor){
    case 1:                  // Right motor
    if(dir == 0){
  digitalWrite(ForwardR, LOW);  // No Direction
  digitalWrite(BackwardR, LOW);
    }
    else if(dir == 1){
  digitalWrite(ForwardR, HIGH);  // Forward direction
  digitalWrite(BackwardR, LOW);
  analogWrite(PWMA, sped);      // PWM for motor speed
    }
    else if(dir == 2){
  digitalWrite(ForwardR, LOW);     // Backward direction
  digitalWrite(BackwardR, HIGH);
  analogWrite(PWMA, sped);
    }
    break;
```

```
  case 2:                   // Left motor
  if(dir == 0){
 digitalWrite(ForwardL, LOW);
 digitalWrite(BackwardL, LOW);
  }
  else if(dir == 1){
 digitalWrite(ForwardL, HIGH);
 digitalWrite(BackwardL, LOW);
 analogWrite(PWMB, sped);
  }
  else if(dir == 2){
 digitalWrite(ForwardL, LOW);
 digitalWrite(BackwardL, HIGH);
 analogWrite(PWMB, sped);
  }
  break;
 }

}
```

APPENDIX B

MATLAB m.file for real time plotting

```
%Real time plotting
clc
clear all
prev = 0;
s = serial('COM4','BaudRate',250000); % create
serial communication object on port COM3

fopen(s);
x=0;
t(1)=0;
flushinput(s);
tic;
while(~isempty(s) && toc <= 30)

   x = x+1;
   t(x) = toc;
   y(x) = str2double(fgets(s))
   drawnow
   plot(t,y,'linewidth',2)
   title('Real-Time Pitch Angle');
   xlabel('time (seconds)');
   ylabel('Pitch Angle (Degrees)');
   axis([0 30 -80 80]);
   flushinput(s);
   grid on

end
fclose(s);
delete(s);
clear all   % end communication with arduino
```