Sudan University of Science and Technology

College of Graduate Studies

Balancing of Two Wheeled Robot Using PID Controller

إتزان الروبوت ذو العجلتين بإستخدام المتحكم التناسبي التكاملي التفاضلي

A Thesis submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Engineering (Control and Microprocessor)

Prepared by:

ISRAA AHMED MOHAMMED ZAIN AHMED

Supervisor:

Dr. AWADALLA TAIFOUR ALI ISMAIL

November/2018

الآيـة

قال تعالى:

(إِنَّ فِي خَلْقِ السَّمَاوَاتِ وَالأَرْضِ وَاخْتِلاَفِ اللَّيْلِ
وَالنَّهَارِ وَالْفُلْكِ الَّتِي تَجْرِي فِي الْبَحْرِ بِمَا يَنْفَعُ النَّاسَ
وَمَا أَنْزَلَ اللَّهُ مِنَ السَّمَاءِ مِنْ مَاءٍ فَأَحْيَا بِهِ الأَرْضَ
بَعْدَ مَوْتِهَا وَبَثَّ فِيهَا مِنْ كُلِّ دَابَّةٍ وَتَصْرِيفِ الرِّيَاحِ
وَالسَّحَابِ الْمُسَخَّرِ بَيْنَ السَّمَاءِ وَالأَرْضِ لآيَاتٍ لِقَوْمٍ
يَعْقِلُونَ).

صدق الله العظيم

سورة البقرة

الآية 164

DEDICATION

All praise to Allah, today we fold the day’s tiredness and the errand summing up between the cover of this humble work.

To the utmost knowledge lighthouse, to our greatest and most honored prophet Mohammed-May peace and grace from Allah be upon him.

To the spring that never stops giving, to my mother who weaves my happiness with strings from her merciful heart…to my mother.

To whom he strived to bless comfort and welfare and never stints what he owns push me in the success way who taught me to promote life stairs wisely and patiently, to my dearest father.

To whose love flows in veins and my heart always remembers them, to my small family.

To those who taught us letters of gold works of jewel of the utmost and sweetest sentences in whole knowledge. Who reworded to us their knowledge simply and from their thoughts made a lighthouse the knowledge and success path, to our honored teachers and professors.

ACKNOWLEDEMENT

In the name of Allah, the Most Gracious and the Most Merciful Alhamdulillah, all praises to Allah for endowing me with health, patience, and knowledge to complete this work.

I would like to express my appreciation to my supervisor **Dr. Awadalla Taifour Ali** who has cheerfully answered my queries, provided me with materials, checked my examples, assisted me in my myriad ways with the writing and helpfully commented on earlier drafts of this thesis.

ABSTRACT

Nowadays robots can be seen in our daily life. Recently, robotic applications and their wide range of functionalities have drawn many engineers' attentions. Robot with at least three wheels can achieve static stability further simplifying dynamics. The inverted pendulum system is naturally unstable. Two wheeled balancing robot is application of inverted pendulum that requires a controller to maintain its upright position. This thesis will describe the modeling of two wheeled balancing robot, design the robot controller using Proportional-Integral-Derivative (PID) controller and implement the controller on the robot. Micro-Processing Unit (MPU) is used, which combines accelerometer and gyroscope measurement in order to estimate and obtain the lilt of the robot. PID controller is applied to correct the error between the desired set point and the actual tilt angle and adjust the Direct Current (DC) motor speed according to balance the robot. The simulation result of the model is compared with the developed hardware and performance of the controller.

مستخلص

في هذة الايام يمكن رؤية الروبوتات في حياتنا اليومية. فقد جزبت التطبيقات الالية و مجموعتها الواسعة من الوظائف مؤخرا اهتمام كثير من المهندسين. يمكن للروبوت ذو الثلاثة عجلات علي الاقل ان يحقق استقرارية ثابتة لابسط الديناميكيات. نظام البندول المعكوس غير مستقر في وضعة الطبيعي. فاتزان الروبوت ذو العجلتين من تطبيقات البندول المعكوس مما يتتطلب نظام تحكم للحفاظ علي وضعه المستقيم. سيصف هذا البحث نمذجة لموازنة روبوت بعجلتين ، تصميم وحدة تحكم للروبوت باستخدام المتحكم التناسبي التاكملي التفاضلي وتنفيذ وحدة التحكم علي الروبوت. يستخدم الحساس الذي يجمع بين قياس التسارع وقياس جيروسكوب لتقدير والحصول علي ميل الروبوت. يتم تطبيق وحدة التحكم لتصحيح الخطأ بين النقطة المحددة وزواية الميل الفعلية وضبط سرعة المحرك المستمر وفقا لتحقيق التوازن في الروبوت. كذلك يتم مقارنة نتيجة المحاكاة لنموذج مع المكونات واداء وحدة التحكم.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

ARFE	Reference voltage
g	Gravitational acceleration
L	Length of the pendulum
m	Mass of pendulum
M	Mass of the robot
VCC	Digital supply voltage
XDA	Auxiliary serial data
XCL	Auxiliary serial clock
x	Position of robot
θ	Angle of the robot

LIST OF ABBREVIATIONS

D	Derivative
DC	Direct Current
DOF	Degree of Freedom
DMP	Digital Motion Processor
EEPROM	Electrically Erasable Programmable Read Only Memory
GND	Ground
I	Integral
I^2C	Inter Intergrated Circuit
IP	Inverted Pendulum
LED	Light Emitting Diode
LQR	Linear Quadratic Regulation
MATLAB	MATrix LABoratoryis
MEMS	Micro Electro Mechanical System
MIMO	Multiple Input Multiple Output
MPU	Micro-Processing Unit
P	Proportional
PID	Proportional Integral Derivative
PWM	Pulse Width Modulation
RPM	Revolutions Per Minute
SCL	Serial Clock
SDA	Serial Data
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
USB	Universal Serial Bus

CHAPTER ONE

INTRODUCTION

CHAPTER ONE

INTRODUCTION

1.1 Background

Effective and efficient control system designs provide the robot with the ability to control itself and operate autonomously, since PID control is becoming a definite possibility with advancements in nonlinear control systems. Improved synthetics and materials allow for robust and cosmetically aesthetic designs to be implemented for the construction and visual aspects of the robot. Two wheeled robots are one variation of robot that has become a standard topic of research and exploration for young engineers and robotic enthusiasts. They offer the opportunity to develop control systems that are capable of maintaining stability of an otherwise unstable system. This type of system is also known as an inverted pendulum. This research project aims to bring this, and many of the previously mention aspects of a robot together into the building of a two wheeled balancing robot with a nonlinear, PID controller.

Two wheeled robot is a robot that is capable of balancing upright on its two wheels is known as a two wheeled balancing robot. The following Figure 1.1 contains the physical view for the robot designed as part of this study. The process of balancing is typically referred to as stability control. The two wheels are situated below the base and allow the robot chassis to maintain an upright position by moving in the direction of tilt, either forward or backward, in an attempt to keep the center of the mass above the wheel axles. The wheels also provide the locomotion thus allowing the robot to transverse across various terrains and environments.

This type of robot provides a challenging problem and has resulted in many useful and interesting designs being developed. One such two wheeled robot that has become a commercial success is the Segway by Segway Inc.

The immediate impact has been within the personal transportation area where an alternative to cumbersome wheelchairs is now available. Segway proves a comfortable mobility opportunity for the elderly or people with disability thus improving their individual sense of independence at the same time. The theory used to maintain stability of these robots is based on the inverted pendulum theory [1].

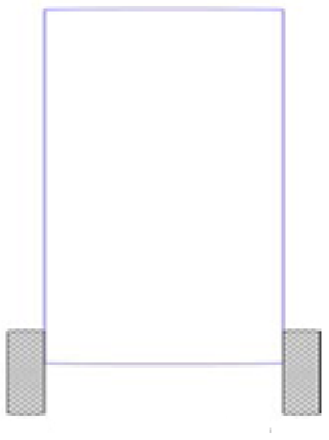

Figure 1.1: Views of the two wheeled balancing robot

2.1 Problem Statement

Robots are increasingly implemented today and are used in a variety of different application. Robot with at least three wheels can achieves static stability, further simplifying dynamics. The inverted pendulum systems are naturally unstable. Therefore, a suitable control system technique and method need to be investigated to control the system. The two wheeled balancing robot is application of the inverted pendulum that requires controller to maintain its upright position. To achieve this controller needs to be designed and implement on robot to balance the inverted pendulum. The robot is inherently unstable without external control it would roll around the wheels rotation axis and eventually fall. Various types of control were implemented on two wheeled balancing robot nonlinear controller.

3.1 Objectives

The development of the two wheel robot based on the inverted pendulum concept by using PID controller is the aim of this research. In order to achieve this aim, the objectives as follows are formulated:

i. To design and develop the prototype for two-wheel balancing robot with PID controller.

ii. To perform the simulation of a two-wheeled balancing robot based on its existing mathematical model with the robot' actual parameters.

iii. To evaluate the performance of the developed self-balancing robot using a standard approach.

4.1 Methodology

The system is divided into parts; hardware and software. In the hardware circuit requires feedback sensors which are angle sensor and wheel encoder. The control system for an inverted pendulum is applied when the wheelchairs maneuvers a small step or road curbs. A software code is developed to control the overall system functions. The code is written in C language using Basic Compiler Arduino Nano is interconnected to the system.

1.5 Layout

This research consists of five chapters. The first chapter contains of background, research problems, objectives and the scope of work methodology. While the second chapter consists of the type of control system, robot, two wheeled robot and PID controller. Chapter three covers the hardware and software consideration of the project while chapter four contains the implementation, simulation and result of the design. Chapter five consists of the recommendations and conclusion.

CHAPTER TWO

THEORETICAL BACKGROUND

CHAPTER TWO

THEORETICAL BACKGROUND

2.1 Control System

A control system is a collection of components that is designed to drive a given system (plant) with a given input to a desired output. Control engineering is based on the foundations of feedback theory and linear system analysis it integrates the concepts of network theory and communication theory. Therefore control engineering is not limited to any engineering discipline but is equally applicable to aeronautical, chemical, mechanical, environmental, civil and electrical engineering. A control system is an interconnection of components forming a system configuration that will provide a desired system response [2]. The main types of control systems are:

2.1.1 Open loop control systems

An open loop control system there is no automatic correction of the variation in its output. That is in this type of system sensing of actual output and comparing of this output with the desired input does not take place [3].

2.1.2 Closed loop control systems

A closed loop control system is a system where the output has an effect upon the input quantity in such a manner as to maintain the desired output value. An open loop control system becomes a closed loop control system by including feedback. A feedback control system is a control system that tends to maintain a prescribed relationship of one system variable to another by comparing functions of these variables and using the difference as means of control [4]. The main differences between the existing mathematical models are what dynamic equations have been used to derive the model, as well as how many degrees of freedom have been modeled.

2.1.3 State-space representation

When describing a dynamic system it is often helpful to use state-space representation. The idea is that there all timed exists a vector, the state vector, which fully represents the dynamic state of the system [2]. A linear state-space representation is generally written as:

$$\dot{x} = \mathrm{A}x + \mathrm{B}u \qquad (2.1)$$

$$y = \mathrm{C}x + \mathrm{D}u \qquad (2.2)$$

Where x is the state vector containing the state variables, u and y are the system input and output respective, and the matrices A, B, C and D define the state dynamics of the system. The state variables of the state vector are usually chosen to be system variable of interest for desired control and observation of the system. State space models are especially useful when describing systems with multiple inputs and outputs, since the order of the vectors and matrices only needs to be adjusted accordingly. A benefit with a system that is represented by a state-space model is that it is possible to evaluate the system's stability by checking the eigenvalues of the matrix. If an eigenvalue is not strictly negative and real it represents an unstable pole. Another benefit is that the controllability is easily checked by evaluating the controllability matrix, $\mathrm{M_c} = [\mathrm{B}\ \mathrm{AB}\ \mathrm{A}^2\mathrm{B}\ \ldots\ \mathrm{A}^{n-1}\mathrm{B}]$. The system is controllable if $\mathrm{M_c}$ has full rank [3].

2.1.4 Linear-quadratic regulation

Linear Quadratic Regulation (LQR) is an optimization algorithm used to control a system by minimizing the system's deviations from desired values. It is of great help when dealing with systems which are Multiple-Input Multiple-Output (MIMO). The goal is to minimize the cost function $J =$

$\int_0^\infty (x^T Qx + u^T Ru)dt$ of the linear system $\dot{x}$ = Ax+ Bu and produce the weighting matrix K for the feedback control u = -Kx. Q and R are weighting coefficient matrices which represent the relative importance of the control signal and error value, respectively, for J. K is found to be K = $R^{-1}B^T X$ where X is found by solving the Riccati equation $A^T X$ + X A - XB$R^{-1}B^T X$ + Q = 0. This procedure can be swiftly carried out with the help of software tools like MATLAB [2].

2.1.5 Pole placement

The purpose of pole placement is just as with LQR, to find the weighting coefficient matrix K for a system. Instead of using the Q and R matrices that are created in LQR the poles of the system are placed at predetermined locations in the s-plane. This is a convenient method since the placed poles correlate directly to the system's eigenvalues and therefore the stability of the system. The principle can only be used on a controllable system and is done by evaluating the existing characteristic equation for the full state feedback system:

$$\det [sI - (A - BK)] \tag{2.3}$$

The K matrix is then picked by placing the poles and adapting the K matrix to what it needs to be for the poles to be in the desired places [3].

2.2 Robot

A robot is a machine designed to execute one or more takes automatically with speed and precision. Robots are sometimes grouped according to the time frame in which they were first widely used. First-generation robots date from the 1970s and consist of stationary, nonprogrammable, electromechanical devices without sensors. Second-generation robots were developed in the 1980s and can contain sensors and programmable controllers. Third-generation robots were developed between approximately 1990s and the present. These machines can be stationary or mobile, autonomous or insect type, with sophisticated programming, speed recognition and/or synthesis, and other advanced features. Fourth-generation robots are in the research-and-development phase, and development phase, and include feature such as artificial intelligence, self-replication, self-assembly and nanoscale size. Robots that resemble humans are known as androids; however, many robots are not built on the human model. A robot can be remote controlled by a human operator, sometimes from a great distance. Robots are employed on production lines in factories, appeared as intelligent machines to do specific task or used as commercial products [5].

2.2.1 Type of robots

Robots do a lot of different tasks in many filed and the number of jobs entrusted to robots is growing steadily. So the best ways how to divide robots into type is a division by their application there are:

- **Industrial robots**

Industrial robots are robots used an industrial manufacturing environment. Usually these are articulated arms specifically developed for such applications as wilding, material handling, painting and others.

- **Mobile robot**

Mobile robots have the capability to move around in their environment and are not fixed to one physical location. An example of a mobile that is common use today is the automated guided vehicle. There are two types of mobile robotics are autonomous and non-autonomous robots. Autonomous mobile robots can explore their environment without external guidance, while guided robots use some type of guidance system to move, other semistationary robots have a small range of movement. The field of mobile robotics is important today as companies look at how to utilize artificial intelligence. For instance, those looking at the consumer electronics world might remark that although autonomous mobile robots have been around for a while, they are not often represented in consumer offerings – for instance, in items like Alexa and other virtual tools that are stationary rather than robotic.

- **Medical robots**

Robots used in medicine and medical institutions. First and foremost -surgery robots. Also, some automated guided vehicles and maybe lifting aides.

- **Domestic or household robots**

Robots used at home. This of robots includes many quite different devices such as robotic vacuum cleaners, robotic pool cleaners, sweepers, gutter cleaners and other robots that can do different chores.

- **Service robots**

Robots those do not fall into other types by usage. These could be different data gathering robots, robots made to show off technologies and robots used for research.

- **Military robots**

Robots used in military. This type of robots includes bomb disposal robots, different transportation robots and reconnaissance drones. Often robots initially created for military purposes can be used in law enforcement, search and rescue and other related fields.

- **Entertainment robots**

These are robots used for entertainment. This is a very broad category. It starts with toy robots such as robosapien or running alarm clock and ends with real heavyweight such as articulated robot arms used as motion simulators [6].

2.2.2 The two wheeled inverted pendulum

Two-wheeled robots are based on the idea of the Inverted Pendulum (IP) system. The idea of a mobile inverted pendulum robot has surfaced in recent years and has attracted interest from control system researchers worldwide. In this study the two wheeled inverted pendulum was provided a design and implementation. Figure 2.1 shows both side and front of two wheeled inverted pendulum system.

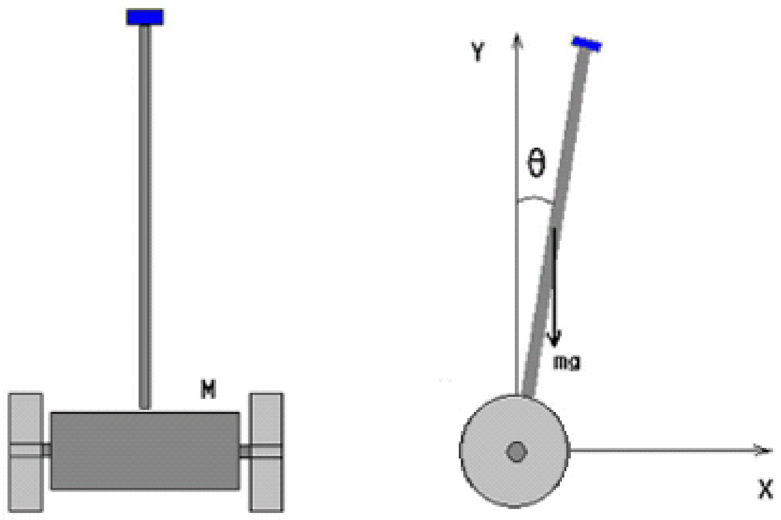

Figure 2.1: Side and front of two wheeled inverted pendulum system

It is required to balance the pendulum at a zero tilt angle. Once the system is balanced, it may be ordered to move forward, backward, turn right or left. The system consists of three main parts:

- **The wheels:** Moves the system backward or forward to balance the body of the system.
- **The chassis:** Hold the motors, circuit and any parts required for system.
- **The pendulum:** The parts of system that causes the instability and need to be stabilized. This part may be hidden inside the chassis [7].

2.2.3 A two wheel balancing robot

Dual-wheel robot has two wheels an upright body frame wheel all circuit are placed. Two wheeled robots are harder to balance than other types because they must keep moving to maintain upright. The center of gravity of the robot body is kept below the axle; usually this is accomplished by mounting the batteries below the body. They can have their wheels parallel to each other, these vehicles are called dicycles, or one wheel in front of the other, tandemly placed wheels. Two wheeled robots must keep moving to remain upright and they can do this by driving in the direction the robots is falling. To balance, the base of the robot must stay with under its center of gravity. For a robot that has the left and right wheels, it needs at least two sensors. A tilt angle and wheel encoders which keep track of the position of the platform of robot [5].

The uniqueness of inverted pendulum system has drawn interest from many researches due the unstable nature of the system. The idea of a mobile inverted pendulum robot has surfaced in recent years and has attracted interest from control system researchers worldwide. A two wheeled differential drive mobile robot based on the inverted pendulum mobile is built as a platform to investigate the use of a Kalman filter to estimate the tilt angle. As the robot is mechanically unstable, it becomes necessary to explore the possibilities of implementing a control system to keep the system in equilibrium. This thesis examines the suitability and evaluated the performance of a LQR and a pole-placement controller uses in balancing the system. The LQR control uses several weighting matrix to obtain the appropriate control force to be applied to the system while the pole-placement requires the poles of the system to be placed to guarantee stability. As the robot will be moving about on a surface, a PID controller is implemented to control the trajectory of robot. The instability of inverted pendulum systems has always been an excellent test bed for control theory experimentation. Therefore it is also the aim of this thesis to investigate the suitability and examine the performance of linear control systems like the LQR and pole-placement controller in stabilizing the system and compare it to modern approaches like fuzzy logic controller [7].

2.3 Proportional Integral Derivative

A PID is a generic control loop feedback mechanism widely used in industrial control systems because PID controller is simple, provide good stability and rapid response. A PID controller attempts to minimize the error value between a measured process variable and a desired set point by calculating and then outputting a corrective action that can adjust the process accordingly such as the position of control value, a damper, or the power supplied to a heating element to a new value by determined by a weight sum show in Equation (2.7). Figure 2.2 is shown all of the three different parts of PID-controller put their focus on reference and error with different relations to time. The PID controller calculation (algorithm) involves three separate parameters; the Proportional (P), the Integral (I) and Derivative (D) values [8,9].

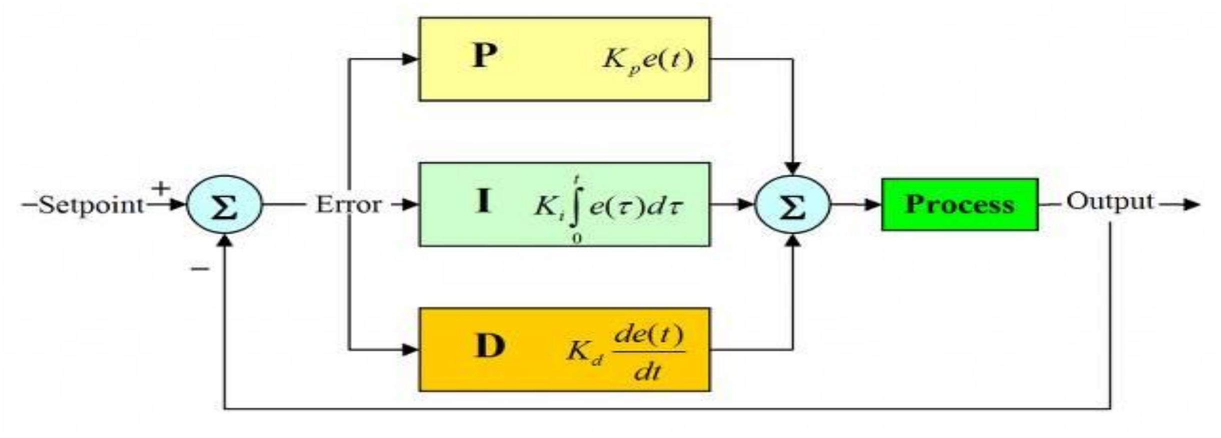

Figure 2.2: PID controller

- **Proportional term**

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error a constant (K_p) called the proportional gain constant is given by:

$$P_{out} = K_p e(t) \tag{2.4}$$

- **Integral term**

The contribution from the integral term is proportional to both the magnitude error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain (K_i) and added to controller output. The integral term is given by:

$$I_{out} = K_i \int_0^t e(t)\,dt \tag{2.5}$$

- **Derivative term**

The derivative of the process error is calculated by determining the slope of the error time and multiplying this rate of change by the derivative gain K_d. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain K_d is given by:

$$D_{out} = K_d \frac{d}{dt} e(t) \tag{2.6}$$

P controller the present error is reduced with a proportional change. The I controller integrates over time and can reduce the error based on past errors. The D controller derives and measures the present change of the error and uses this to predict future errors. When all three controllers are used together it is called a PID controller. Some application may require using one or two terms to provide the appropriate system control. This is achieved by setting the other parameters to zero. A PID controller will be called a PI, PD, P or I controllers in absence of respective control action. It all depends on the system at hand and how it needs to be controlled. The significance of three parts of the controller on the control effort is determined by their respective weighting coefficient; K_p, k_i and k_d. Through mathematical models and methods for variable estimation it is possible to get reasonable values for the K-variable. This is however very complex and time demanding, since the model and the real system usually differ appreciably and the model therefore needs to be heavily adjusted before presenting any useful results. One way to avoid this is to search for an existing model for a system similar to the one at hand, then start with that one and go straight on to the final calibrations [2].

$$U(t) = K_p e(t) + K_i \int (e(t))dt + k_d \left(\frac{de}{dt}\right) \tag{2.7}$$

2.3.1 The characteristics of PID controller

There are four major characteristics of the closed-loop step response:

1. **Rise time:** The time it takes for the plant output to rise beyond 90% of the desired level for the first time.
2. **Overshoot:** How much the peak level is higher than the steady state, normalized against the steady-state.
3. **Settling time:** The time it takes for the system to converge to its steady state.
4. **Steady-state error:** The difference between the steady-state output and the desired output.

A proportional controller (K_p) will have the effect of reducing the rise time and will reduce but never eliminate the steady-state error. An integral control

(K_i) will have the effect of eliminating the steady-state error for a constant or step input, but it may make the transient response slower. A derivative control (K_d) will have the effect of increasing the stability of the system, reducing the overshoot and improving the transient response. The effects of each PID controller parameters, K_p, K_i and K_d on a closed loop system are summarized in the Table 2.1.

Table 2.1: Effect of PID Tuning

Response	**Rise Time**	**Overshoot**	**Settling Time**	**Steady State Error**
$\mathbf{K_p}$	Decrease	Increase	Small Change	Decrease
$\mathbf{K_i}$	Decrease	Increase	Increase	Eliminate
$\mathbf{K_d}$	Small Change	Decrease	Decrease	No Change

These correlations may not be exactly accurate because K_p, K_i and K_d are dependent on each other. In fact, changing one of these variables can change the effect of the other two.

2.3.2 PID tuning Ziegler-Nichols tuning

Ziegler-Nichols tuning rules have been widely used to tune PID controllers in process control systems where the plant dynamics are not precisely known. It also can be applied to plants whose dynamics are known. Ziegler-Nichols method present two classical value of proportional gain, integral time and derivative time based on transient response characteristics of a given plant or system.

- **First method**

Obtain a unit step response of the plant shown Figure 2.3. This method applies if obtained response exhibit s-shape curve for unit step input otherwise it can not be applied. This curve can be also obtained by dynamic simulation of plant.

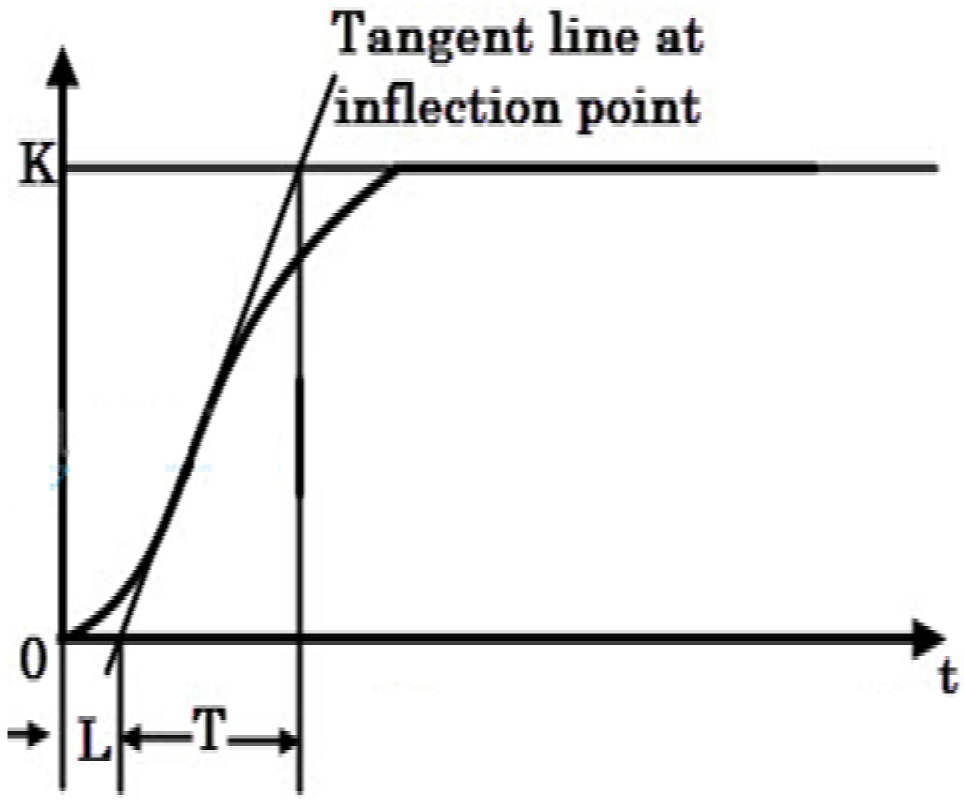

Figure 2.3: Unit step response of aplant and it is s-shape curve

The S-shaped curve also called the reaction curve. The response is characterized by two parameters, L the delay time and T the time constant, which are determined by drawing a tangent line at the inflection point of curve and finding the intersections of the tangent line with the time axis and the steady-state level line. Set parameters of K_p, T_i and T_d in Ziegler-Nichols based on the transient step response of plant values given from Table 2.2 for three type of controllers.

Table 2.2: Controller parameters in Ziegler-Nichols open loop

Type of Controller	**K_p**	**T_i**	**T_d**
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\ \frac{T}{L}$	2L	0.5L

- **Second method**

The method is very similar to trial and error method where integral and derivative terms are set to zero. I increase the proportional gain such that the output exhibits sustained oscillations. If the system does not produce sustained oscillations then this method can not be applied. Figure 2.4 shown the gain at which sustained oscillations produced is called as critical gain (K_{cr}).The sustain oscillations are produced, set the values of parameters K_p, T_i and T_d given from Table 2.3froP, PI and PID controllers based on critical gain (K_{cr}) and critical period (p_{cr}) [8,9].

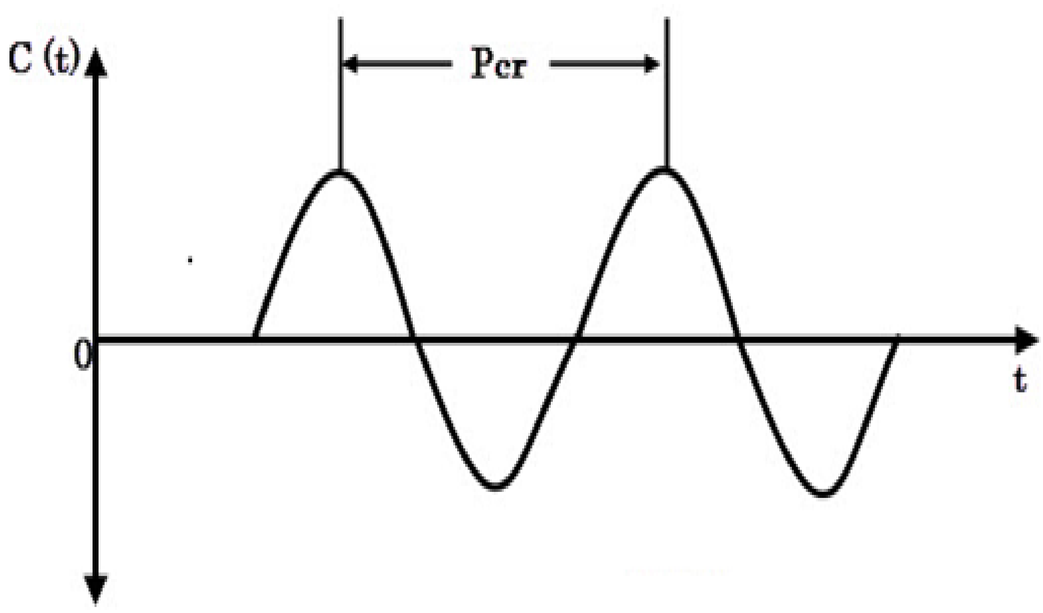

Figure 2.4: Sustained oscillations with period p_{cr}

Table 2.3: Controller parameters in Ziegler-Nichols closed loop

Type of Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

CHAPTER THREE

HARDWARE AND SOFTWAR CONSIDERATION

CHAPTER THREE

HARDWARE AND SOFTWAR CONSIDERATIO

3.1 System Description

The scientific way to solve a problem is to make sure that the right methods are being used. The balancing robot is basically a simple inverted pendulum on two wheels. This is one most widely used example in control classes. The methodology of this study is divided into two parts namely the mechanical design and software algorithm. This chapter describes the method for this subject in order to achieve the desired objective. The study will be developed based on a flowchart that determines all the necessary activities that has been accomplished. The study has gone from planning to modeling, simulation, choice of components, thereafter building and programming of the robot. This study will be design the two wheeled robot with PID as the controller. A system for balancing the two wheeled robot is programming by the Arduion Nano using C language [10].

Figure 3.1 is shown the flow chart of the study, which begins with information collection on topics related to two wheeled balancing robot through literature review. After reviewing all the resources, next step is do the modeling for the inverted pendulum as it provides the fundamental concept for this study. MATLAB/SIMULINK will be used as a simulation tools to see the performance of the controller. Next, is to design the robot. Finally the robot will be tested [1].

3.1.1 Basic concepts

When the robot moves, the motion can be a combination of both forward translational movements. There are two options for making course corrections: Stopping to turn or turning while driving. The first is a pivot turn

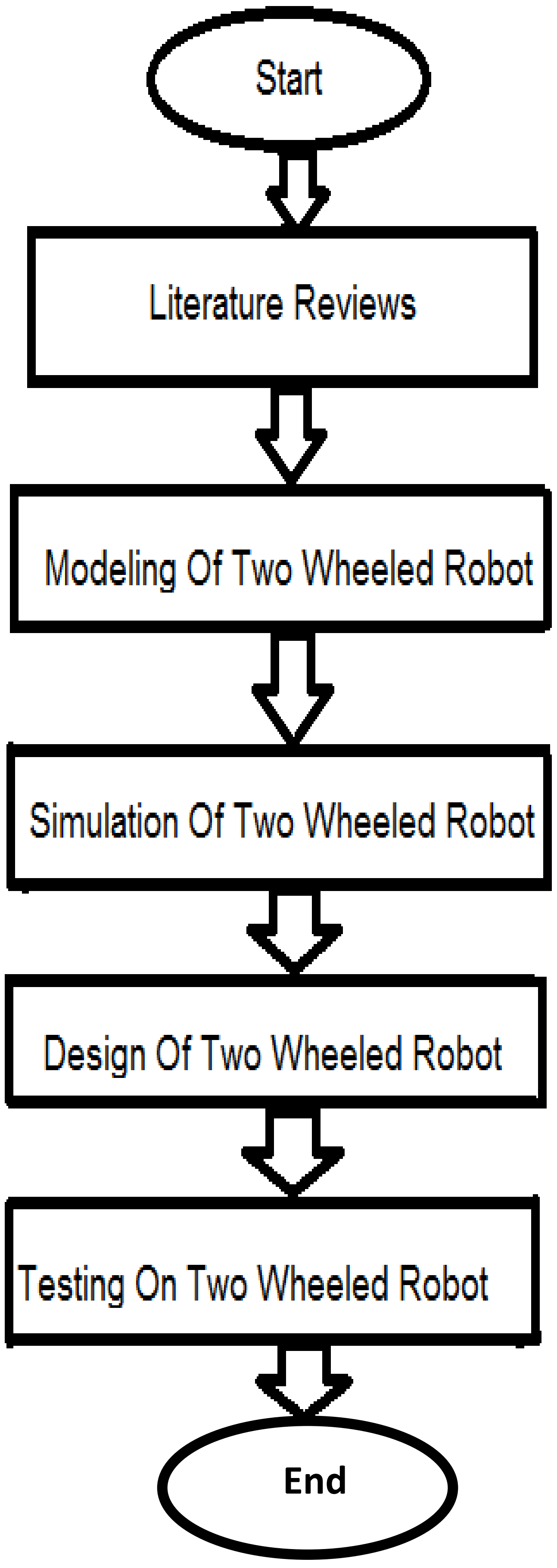
Start
Literature Reviews
Modeling Of Two Wheeled Robot
Simulation Of Two Wheeled Robot
Design Of Two Wheeled Robot
Testing On Two Wheeled Robot
End

Figure 3.1: Flowchart and iteration process of the study

and the robot drives in a straight line unit it reaches the waypoint, or until enough error is built up, then stops and turns using reverse motion on side and forward motion on the other side. This type of turn is faster, but the overall process is much slower and puts unnecessary wear on both the motor controllers and wheels, because they are switching directions. The other option, a differential turn, is to slow one side down while continuing forward motion. Each type of turn can be useful, depending on the circumstances. The two wheeled robot in this thesis was moved turning while driving [10].

3.1.2 Control schematic

Balancing of two wheeled robot control system is used the feedback control system closed loop is shown in Figure 3.2. The sensor is used to measure the actual value and then feeds it back to comparator with the input and the input to controller and each output becomes the input to block diagram.

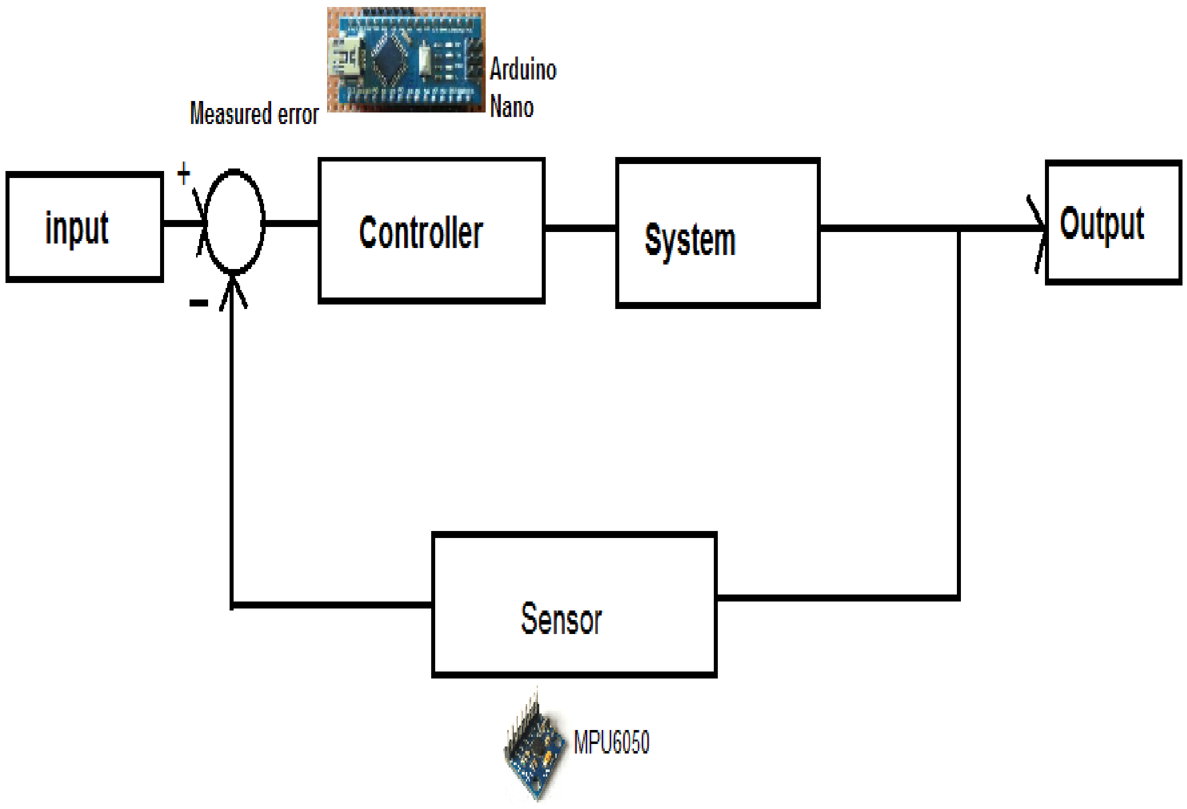

Figure 3.2: Block diagram of control schematic

3.2 Mathematical Modelling of Two Wheeled Robot

The IP on a robot by horizontal control force is shown in Figure 3.3. Where $x_p = x + lsin\theta, z_p = z + lcos\theta, l$ is the distance from the pivot to the mass center of the pendulum, M and m are the mass of the robot and pendulum respectively, (x, z) is the position of the poivt in the xoz coordinate, (x_p, z_p) is position in the $x'o'z'$coordinate [11,12].

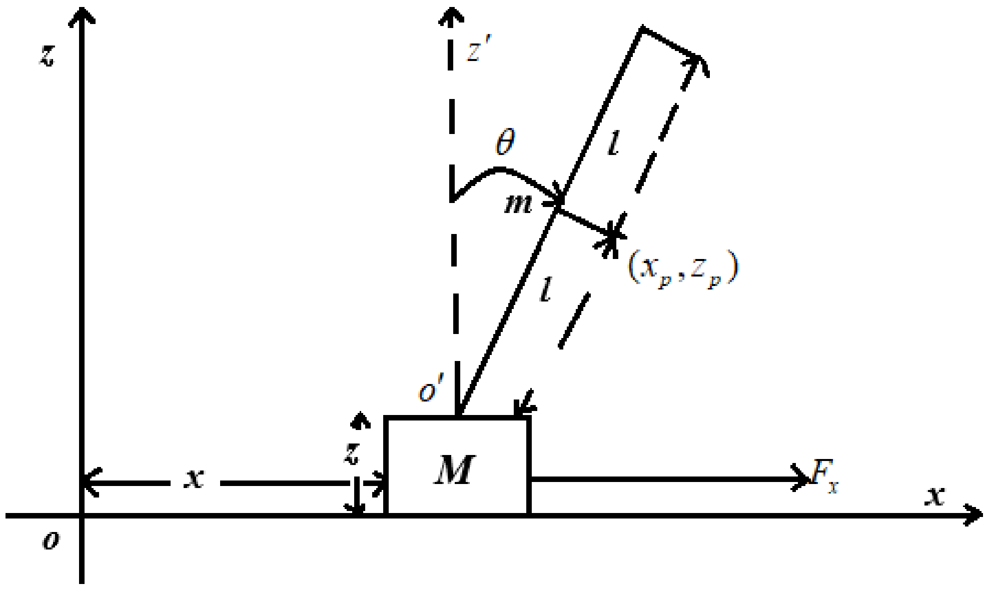

Figure 3.3: Position of IP structure

3.2.1 Mathematical modeling of IP utilizing lagrang' s approach

The total kinetic energy and potential energy of IP are:

$$K = \frac{1}{2}mv^2 \qquad (3.1)$$

$$P = mgz_p \qquad (3.2)$$

$$L = K - P = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m(\dot{x}^2 + \dot{z}^2) + \dot{\theta}^2 l^2 + 2l\dot{\theta}(\dot{x}cos\theta - \dot{z}lsin\theta) - lg(z + lcos\theta) \qquad (3.3)$$

The lagrange' s equations of the IP are:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \left(\frac{\partial L}{\partial x}\right) = F_x \qquad (3.4)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \left(\frac{\partial L}{\partial \theta}\right) = 0 \qquad (3.5)$$

$$(M+m)\ddot{x} + ml\cos\theta\ddot{\theta} - ml\sin\theta\dot{\theta}^2 = F_x \qquad (3.6)$$

$$\ddot{x}\cos\theta + l\dot{\theta} - g\sin\theta = 0 \qquad (3.7)$$

3.2.2 State space equaction of the system

System of IP is represented by a state-space model is:

$$\dot{x_1} = x_1 \qquad (3.8)$$

$$\dot{x}_2 = (-mg\cos x_3\sin x_3 + ml\sin x_3 {x_4}^2 + F_x) / (M + m\sin^2 x_3) \qquad (3.9)$$

$$\dot{x_3} = x_3 \qquad (3.10)$$

$$\dot{x_4} = (-F_x\cos x_3 - ml\sin x_3\cos x_3 {x_4}^2 + (M+m)g\sin x_3)/(ML + ml\sin^2 x_3) \qquad (3.11)$$

Table 3.1 shows the parameters of the two wheeled robot.

Table 3.1: The parameters of the two wheeled robot

Parameter	Value	Descripition
M	0.5 Kg	Mass of the robot
m	0.1 Kg	Mass of pendulum
g	9.8 m/s^2	Gravitational acceleration
L	0.3 m	Length of the pendulum

3.3 System Hardware

The design of the hardware system is crucial in bringing mechanism and software to work together. The main components in the circuit of the balancing robot are the MPU, Arduino controller and DC motor. Figure 3.4 is shown the overall block diagram of the electronic system for the two balancing robot. The MPU is used to measure the acceleration and the angular rate of the robot and the output is processed into digital from. The raw inputs from the MPU are further processed to obtain the tilt angle of the robot. This tilt angle is then fed into the PID controller to generate the appropriate speed to DC motor in order to balancing the robot [13].

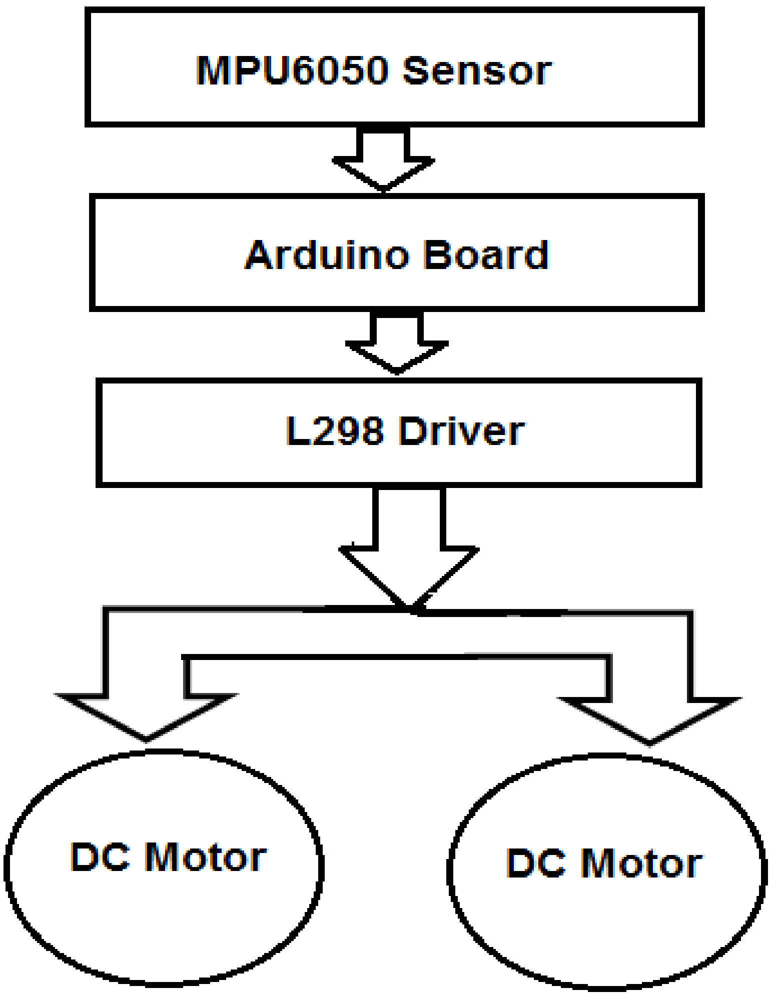

Figure 3.4: Block diagram of system hardware

3.3.1 MPU6050

The MPU6050 is a Micro Electro Mechanical System (MEMS) which consists of a 3-axis accelerometer and 3-axis gyroscope inside it. It us to measure acceleration velocity, orientation, displacement and many other motion related parameter of system or object. The MPU6050 is a 6 Degree of Freedom (DOF) which means that it gives six values as output. This chip uses Inter Integrated Circuit (I^2C) protocol for communication. This module also has a Digital Motion Processor (DMP) inside it which is powerful enough to perform complex calculation and thus free up the work for microcontroller. The Serial Data (SDA) and Serial Clock (SCL) pins are used to establish a connection with the Arduino pins A4 and A5 to receive the accelerometer and gyroscope data. The interrupt pin (INT) is to instruct the Arduino when to read the data from the module and pin instruct the Arduino only when the values change. MPU6050 also has well documented and revised libraries available hence it is very easy to use with famous platforms like Arduino. The MPU6050 can be used in RC car, drone, self balancing robot, humanoid and biped. The DMP combines the raw sensor data and performs some calculations onboard to minimize the errors in each sensor. Accelerometers and gyros have different inherent limitations when used on their own as indicate in Table 3.2 [10]. Figure 3.5 shows the MPU6050 board.

Table 3.2: Comparison between accelerometer and gyroscope

MPU6050	Advantage	Disadvantage
Accelerometer	No bias	Affected by object’s acceleration
Gyroscope	Unaffected by object’s acceleration	Accumulated bias

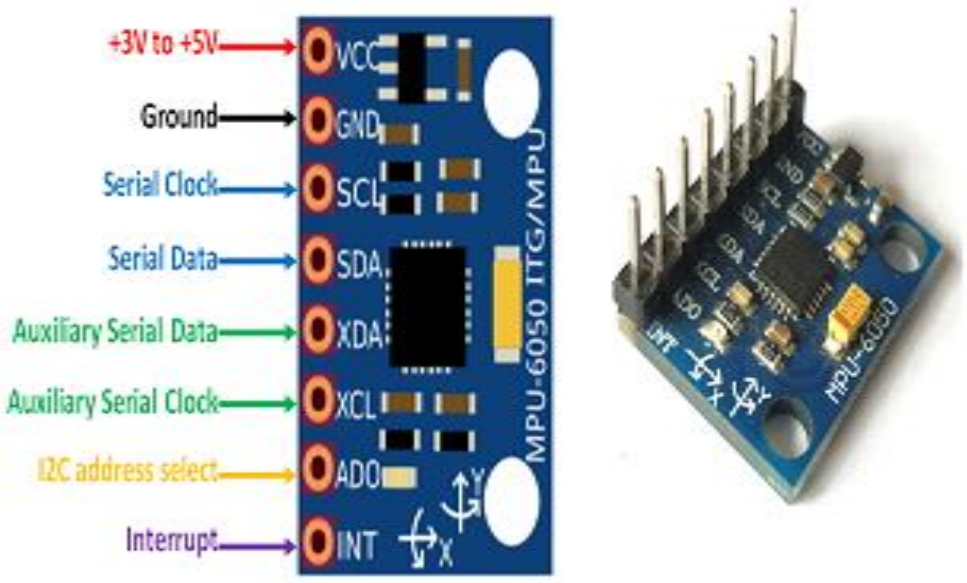

Figure 3.5: MPU6050 board

Pin configuration:

Vcc: Is used for powering the MPU6050 module with respect to ground and can be +3 to +5V.

GND: Connected to ground of system

SDA: SDA is used for data transferring between controller and MPU6050 module.

SCL: SCL is used for providing clock pulse of input.

XDA: Auxiliary Serial Data (XDA) is sensor 12C SCL data line for configuring and reading from external sensors (optional).

XCL: Auxiliary Serial Clock (XCL) is sensor 12C SCL clock line for configuring and reading from external sensors (optional).

INT: Interrupt pin to indicate that data is available for MCU to read.

AD0: If more than one MPU6050 is used a signal MCU, then this pin can be used to vary the address.

3.3.2 Arduino Nano

The Arduino Nano is small, complete and breadboard-friendly board based on the Atmega328 (Arduino Nana 3.o) or Atmega168 (Arduino Nano 2.x). It has more or less same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack and works with a Mini-B Universal Serial Bus (USB) cable instead of a standard one. The Nano was designed and is being produced by Gravitech.

- **Power**

The Arduino Nano can be powered via the Mini-B USB connection 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

- **Memory**

The Atmega168 has 16KB of flash memory for storing code and Atmega328 has 32KB. The Atmega168 has one KB of Static Random Access Memory (SRAM) and 512 bytes of Electrically Erasable Programmable Read Only Memory (EEPROM) and the Atmega328 has two KB of SRAM and one KB of EEPROM.

- **Input and output**

There are totally 14 digital pins and 8 analog pins on Nano board:

❖ **Digital pins**

Each of the 14 digital on the the Nano can be used as input or output, using pin mode(), digitalWrite() and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40mA and has an internal pull-up resistor of 20-50 Kohms. In addition these pins apart from serving their purpose can also be used for special purposes which are discussed below:

- **Serial (RX and TX):** Pin (0) and pin (1) used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of FTDI USB -to-TTL serial chip.
- **External interrupts:** Pin (2) and pin (3) can be configured to trigger an interrupt on a low value, a rising or falling edge or change in value.
- **Pulse Width Modulation (PWM):** Pin (3, 6, 9, 10) provide 8-bit PWM output with the analogWrite() function.
- **Serial Peripheral Interface (SPI):** These pins support SPI communication which although provide by the underlying hardware, is not currently included in the Arduino language.
- **Light Emitting Diode (LED):** There is a built-in LED connected to digital pin (13). When the pin is high value, the LED is on, when the pin is low, it is off.

❖ **Analog pins**

The Nano has 8 analog inputs, each of which provides 10 bits of resolution. By default they measure from ground to 5 volt, though is it possible to change the upper end of their range using the analog Reference() function. Additionally, some pins have specialized functionality:

- **I^{2}C (SDA-SCL):** Pin (4) and pin (5) support I^{2}C (TWI) communication using the write library.
- **AREF:** Reference voltage for the analog inputs.
- **Reset:** Bring this line low to reset the microcontroller. Typically used to add a reset button to shield which block the board.
- **Programming**

The Arduino Nano can be programmed with the Arduino software. The Arduino Nano is merely a set of C/C++ functions that can be called from the code [10]. Once code is installed on the computer, USB is used to connect the board with computer. Figure 3.6 shows the Arduino Nano pins functions.

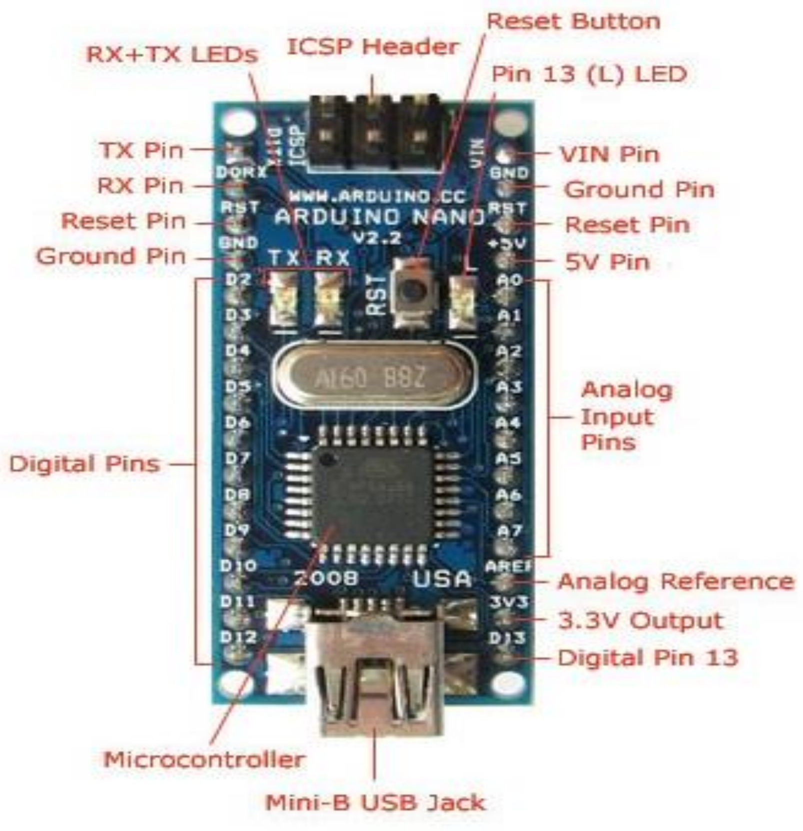

Figure 3.6: Arduino Nano pins functions

3.3.3 DC motor

A DC motor works by converting electric power into mechanical work. This is accomplished by forcing current through a coil and producing a magnetic field that spins the motor. A DC motor is commonly used motor having two input terminals, one is positive and the other is negative. If connect these terminals with the voltage supply the motor will rotate. If change the polarity then motor will be rotated in opposite direction. DC motor has a lot of applications. It use in automation projects, for controlling static as well as mobile robots, in transport system, in pumps, fans, bowers and for industrial use as well. DC motor for making robots are light weight, high torque and low Revolution per Minute (RPM). They can climb hills and have excellent traction. Plus you can mount the wheel on either side of the motor with its double sided output shaft [10]. The main specifications of the DC motor are:

- **Motor voltage:** 3-12V

- **Motor current:** 70mA (typical) – 250mA (max)

- **Speed**: Up to 170RPM

- **Torque:** up to 0.8Kg

- **Gear ration**: 1:48

Figure 3.7 shows the DC motor.

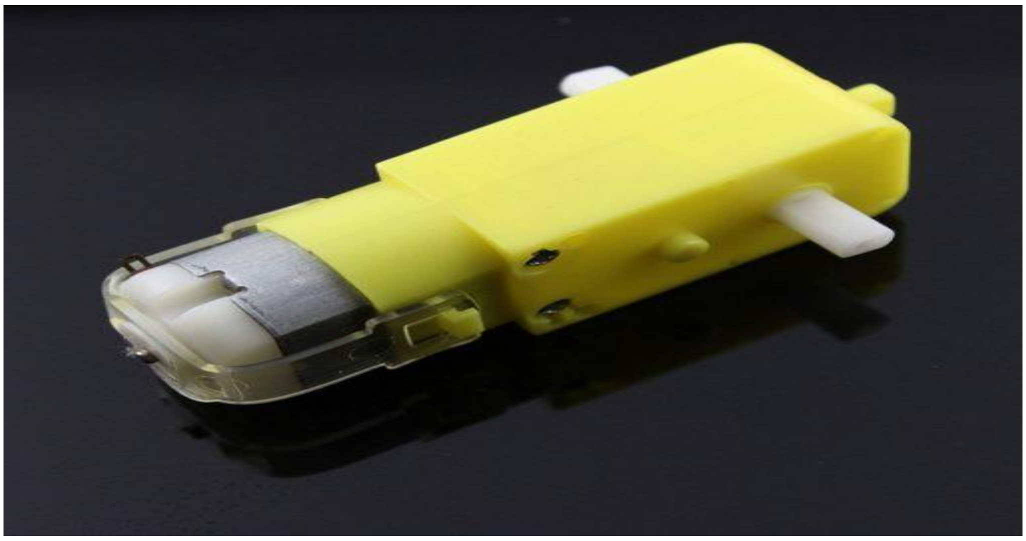

Figure 3.7: DC motor

3.3.4 L298 dual H-bridge motor driver

L298 H-bridge DC motor driver board using ST' s l298N H-bridge DC motor driver I^2C which can be used to drive DC motors or bipolar stepper motors. This driver board is small, light weight, but with a strong driver capability 2A peak current and peak voltage of 46V. Driver performance can be increasing by using a heat sink. Board has two current feedback detection interface to take power within the logic selects the terminal, four pull-up resistor selection terminal, 2-way and four-wire interfaces DC two phase stepper motor interface, control motor direction indicator, four standard fixed mounting holes. This driver board could be used with smart programmable trolley, wheeled robots and variety of controllers [10]. The main specifications of the L298 dual H-bridge motor driver are**:**

Driver: L298

Driver power supply: +5 - +46V

Driver Io: 2A

Logic power outputs Vss: +5 ~ +7V (internal supply +5)

Logic current: 0 ~ 36mA

Controlling level: Low -0.3 ~ 1.5V, High 2.3 ~ Vss

Max power: 25W

Driver weight: 48g

Other extensions such as current probe, controlling direction indicator, pull-up resistor switch, logic, logic part power supply [12]. Figure 3.8 shows the L298 dual bridge driver.

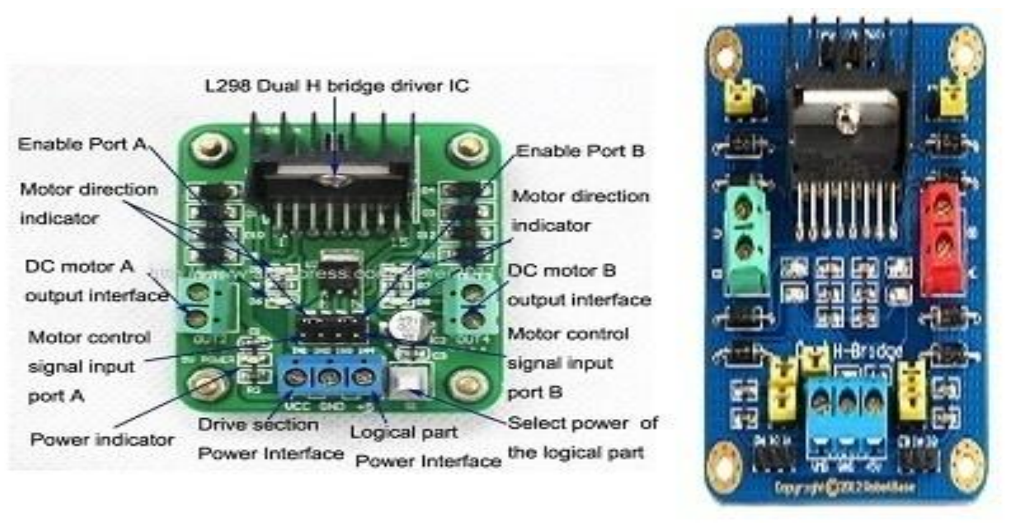
L298 Dual H bridge driver IC
Enable Port A
Enable Port B
Motor direction indicator
Motor direction indicator
DC motor A output interface
DC motor B output interface
Motor control signal input port A
Motor control signal input port B
Power indicator
Drive section Power Interface
Logical part Power Interface
Select power of the logical part
H-Bridge

Figure 3.8: L298 dual bridge driver

3.3.5 Jumper wires

Jumper wires are used to make the connections between all of the components. Use small pieces of the jumper wires in order to give a better look to the designed circuit. Longer wires create complexity and cause many problems while operating the circuit.

3.4 Software System

Software system describes the balancing of the two wheeled robot and designing of the PID controller. The software development is the hardest the most time consuming part of the project. It involves the simulation of the PID controller in MATLAB to get optimum value K_p, K_i and k_d. Then controller will be integrated into the hardware.

3.4.1 Balancing

The balancing software is based on the simple algorithm is shown in Figure 3.9. The program starts with reading all the value of the state variable (position, velocity, tilt angle). These are needed to determine the control signal. The state variables are measured with the different sensors on robot as stated above. If the state variables are all zero, then the control signal will become zero as long as the reference signals (position and tilt angle) are zero, because there is no need for driving the motors if the robot is already balanced. However, if any of the state variables become non-zero there is a need for driving the motors to stabilize and to balance the robot. This makes the next step in the algorithm important, the calculation of the control signal. The control signal is calculated using a specified control method. There are different controls strategies to choice in this project will be used PID controller. When the signal is determined it is applied to both motors in order to make the wheels turn equally and make the robot balance. This process is iterated as long as the robot should be balancing.

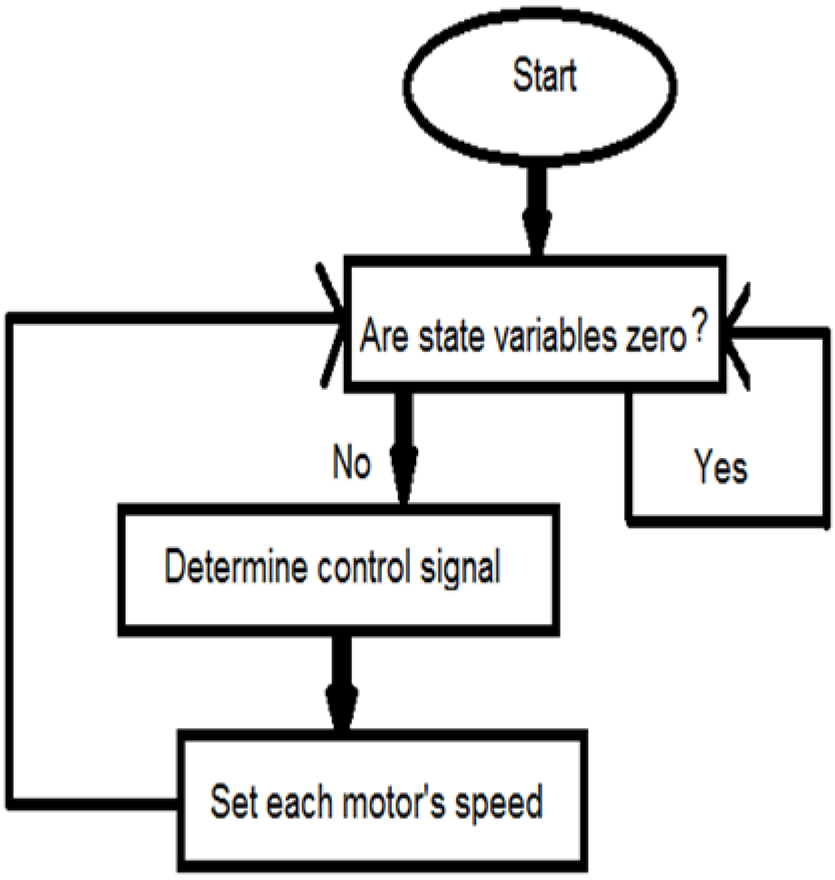

Figure 3.9: The conceptual design of the balancing software

3.4.2 MATLAB and SIMULINK

MATrix LABoratoryis (MATLAB) a language for technical computing developed by the Mathworks, Inc. It provides a single platform for computing, visualization, programming and software development. All problems and solutions in MATLAB are expressed in notation used in linear algebra and essentially involve operations using matrices and vectors. MATLAB can be used to solve problems in control systems. Simulink is a tool for modeling, simulating and analyzing multi domain dynamic systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. Simulink is widely used in control theory and digital signal processing for multi domain simulation and design [13].

3.4.3 Inter-intergrated circuit interface

Inter-Intergrated Circuit (I^2C ptonounced I- squared- C) is a 2 wire serial bus typically used to communicate with sensors and other small components. The two lines of the I^2C bus are SDA (data) and SCL (clock) which can be run in parallel to communicate with several devices at once. I^2C allowsup to 112 “slave“ devices to be controlled by single “master”.

Each slave device on the bus must have it is own unique adress so the master can communicate directly with the intended device. Both master and slave can transfer data over the I^2C bus but that transfer is always controlled by the master. These addresses are typically hard- coded into the slave device, but often allow it to be changed by simply pulling one of pins of the sensor high or low. This allows more than one of the same device to be on the same bus without conflicting adresses [10].

CHAPTER FOUR

SYSTEM IMPLEMENTATION AND RESULTS

CHAPTER FOUR

SYSTEM IMPLEMENTATION AND RESULTS

4.1 System Implementation

There are two main electronic systems that were used to conduct this research; first one is the robot itself as independent unit, second unit is how to balance this robot by using Arduino Nano to control the robot using PID controller. To design this model of the two wheeled robot there are many steps are done for balancing the robot.

4.1.1 Connection diagram

There are many compounds systems that are used to design this model of the two wheeled robot MPU6050, two DC motor, L298 H-bridge and Arduino Nano to connect diagram of these compounds of the two wheeled robot , there are many steps are used to connect the diagram first the MPU6050 is connected to the Arduino Nano. Five connections of MPU6050 have done the power supply (Vcc) and ground of MPU6050 to +5V and ground of Arduino. SCL and SDA pins of MPU6050 are connected with Arduino analog pins (A4 and A5). INT pin of MPU6050 is connected to interrupt 0 of Arduino (D2). Second L298 H-bridge is connected to Arduino to inverse the direction of current flow through the motor. The enable A and enable B pins are connected to Arduino digital pins (D3 and D6) to enable and control the speed of the two motors. The input 1 and input 2 pins of L298 are connected to Arduino digital pins (D4 and D5) to control the rotation direction of the motor A, and input 3 and input 4 are connected to Arduino digital pins (D7 and D8) to control the rotation direction of the motor B. Finally the output 1 and output 2 of L298 are connected to motor A and the output 3 and output 4 of L298 are connected to motor B. Figure 4.1 is shown all the connection diagram of the two wheeled robot.

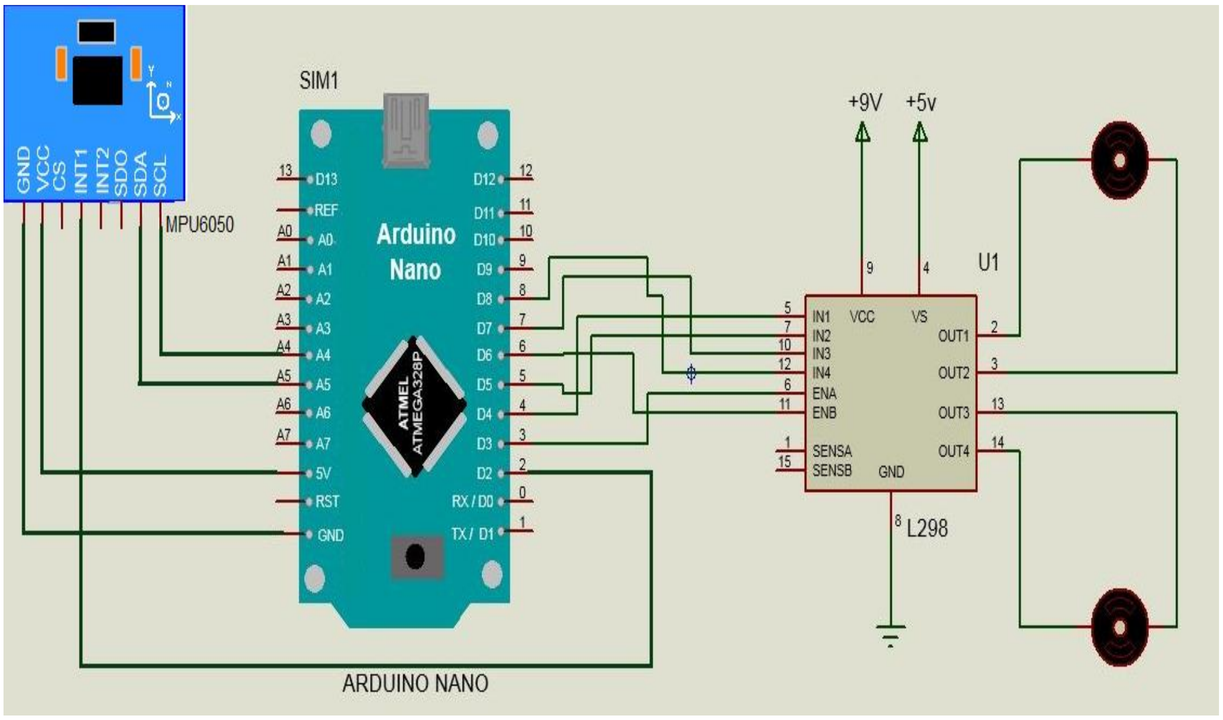

Figure 4.1: Connection diagram of the two wheeled robot

4.1.2 Building the robot

A design of the two wheeled robot consists from a two wheels and chassis. The chassis holds two motors each driver a wheel. The motors are mounted on the chassis shown in Figure 4.2.

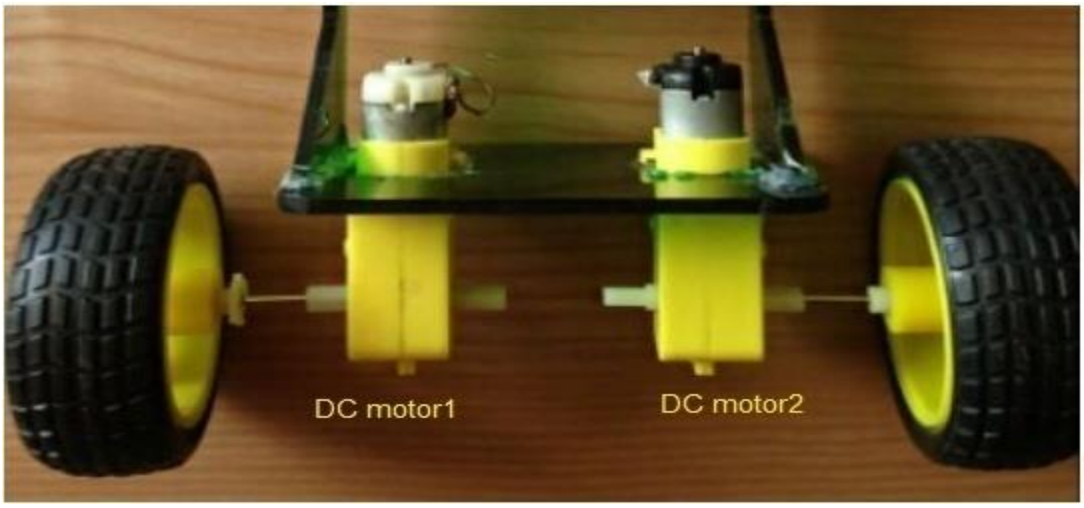

Figure 4.2: Robot frame with tow DC motors

The chassis also holds the main circuit which consist from the sensor (MPU6050) and controller (Arduino Nano) shown in Figure 4.3.

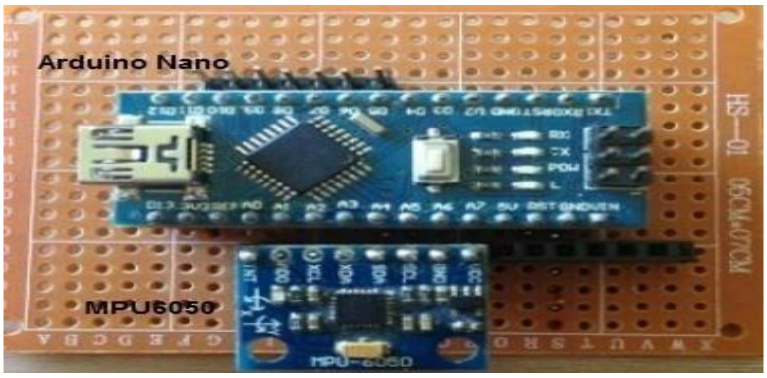

Figure 4.3: Main circuit board Arduino Nano and MPUO6050

Power supply and driver (L298) are connected to the chassis. Figure 4.4 shown the finally design of the two wheeled balancing robot.

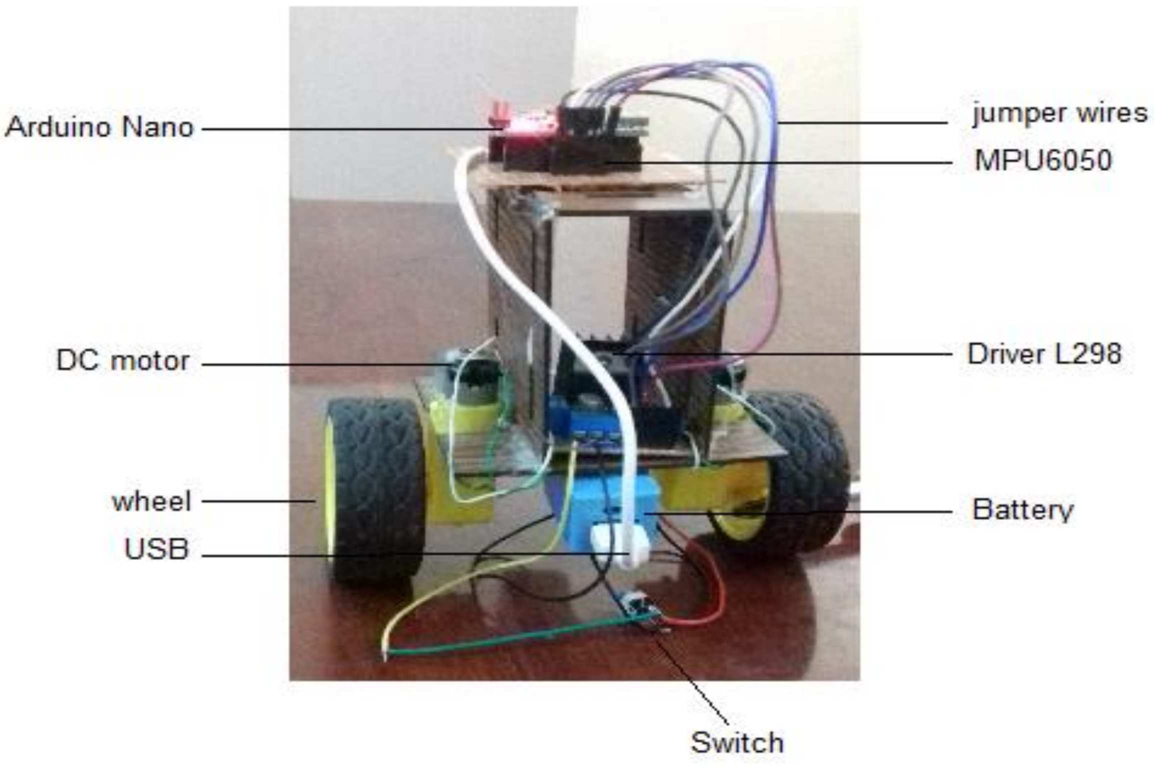

Figure 4.4: Finally design of the two wheeled balancing robot

4.1.3 Operation system

In the loop section the potentiometer value is read and the map value is get from 0 to 1023, to a value from 0 to 255 for PWM signal. Analogwrite function is sent the PWM signal to enable pin of the L298 board, which actually drives the motor. The input 1 and input 2 pins are used for controlling the rotation direction of the motor A and input 3 and input 4 for the motor B. These pins are controlled the switches of H-bridge inside the L298N IC shown in Figure 4.5. If input 1 is LOW and input 2 is HIGH the motor will move forward and vice versa, if input 1 is HIGH and input 2 is LOW the motor will move backward. In case both inputs are same, either LOW or HIGH the motor will stop. The same applies for the inputs 3 and 4 for the motor B.

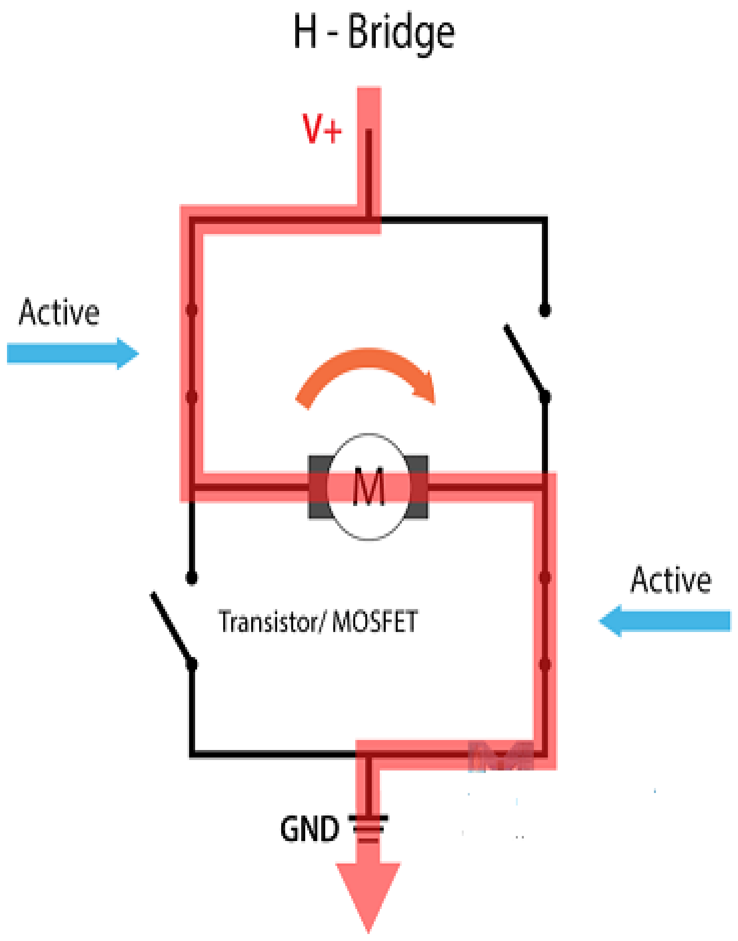

Figure 4.5: Simple H-bridge

In this thesis the input (which is the desired tilt, in degrees) is set by software. The MPU6050 is read the current tilt of the robot and feeds it to the PID controller, which performs calculations to control the motor and keep the robot in upright position. To find the PID parameters (K_d, K_i, K_d) there are many step first K_p, K_i and K_d are maked equal to zero. Second K_p is adjusted**,** too little K_p will make the robot fall over, because there is not enough correction and too much K_p will make the robot go back and forth wildly. A good enough K_p will make the robot go slightly back and forth (oscillate little). Third the K_p is set, K_d is adjusted**,** A good K_d value will lessen the oscillations until the robot is almost steady, also the right amount of K_d will keep the robot standing, even if pused. Finally the K_i is set**, t**he robot will oscillate when turned on, even if the K_p and K_d are set, but will stabilize in time and the correct K_i value will shorten the time it takes for the robot to stabilize [13].

4.2 System Results

The result answers to goals of the thesis set in the purpose section. It covers the all the result of system, from the early simulation results to the construction, programming and final functions of the robot.

4.2.1 MATLAB/SIMULINK

From the state-space model show in Equations (3.12) and (3.13) and the parameter is shown in Table 3.1, the model of the two wheeled robot for MATLAB/SUMILINK is performed in Figure 4.6. The angle of the inverted pendulum is initial by 0.5rad. The output of integrator 3 and integrator 4 are the input of the two functions system. PID is added to control the position of the pivot. In this step PID need not change any more. The parameters of PID controller of the inverted pendulum are given as following: PID: K_p=39, K_i=23.8, K_d=3. Figure 4.7 is shown the output response of the two wheeled robot using PID controller and Figure 4.8 is shown the output of angle.

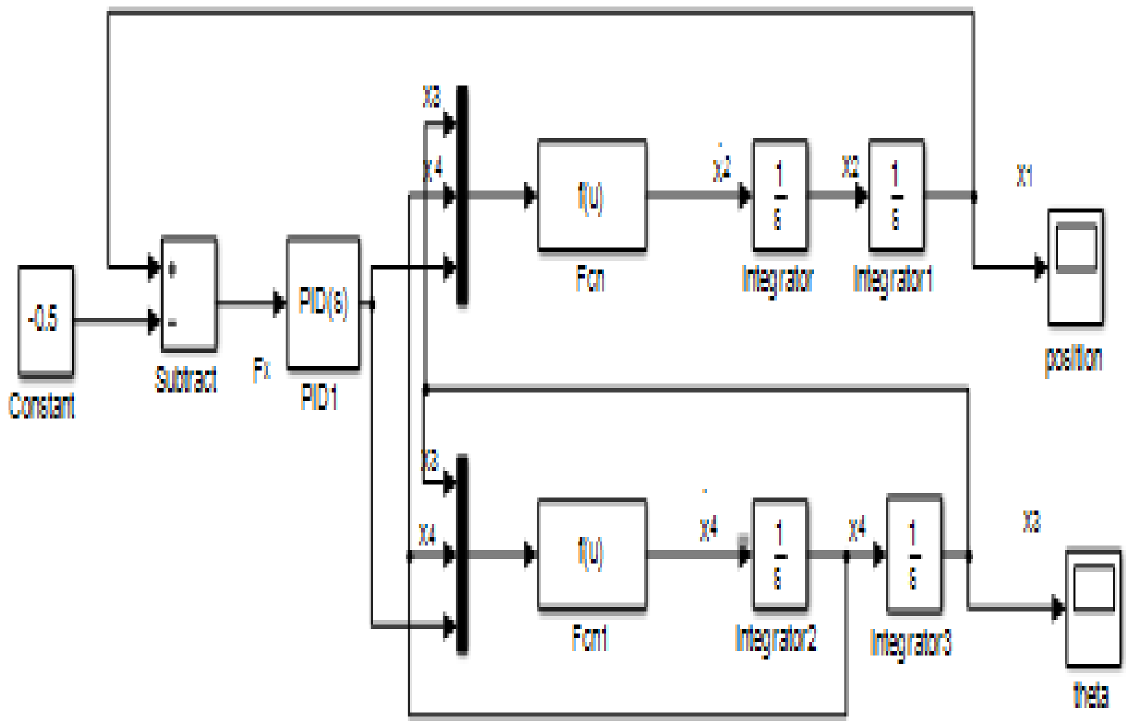

Figure 4.6: MATLAB/SIMULINK model of the two wheeled robot

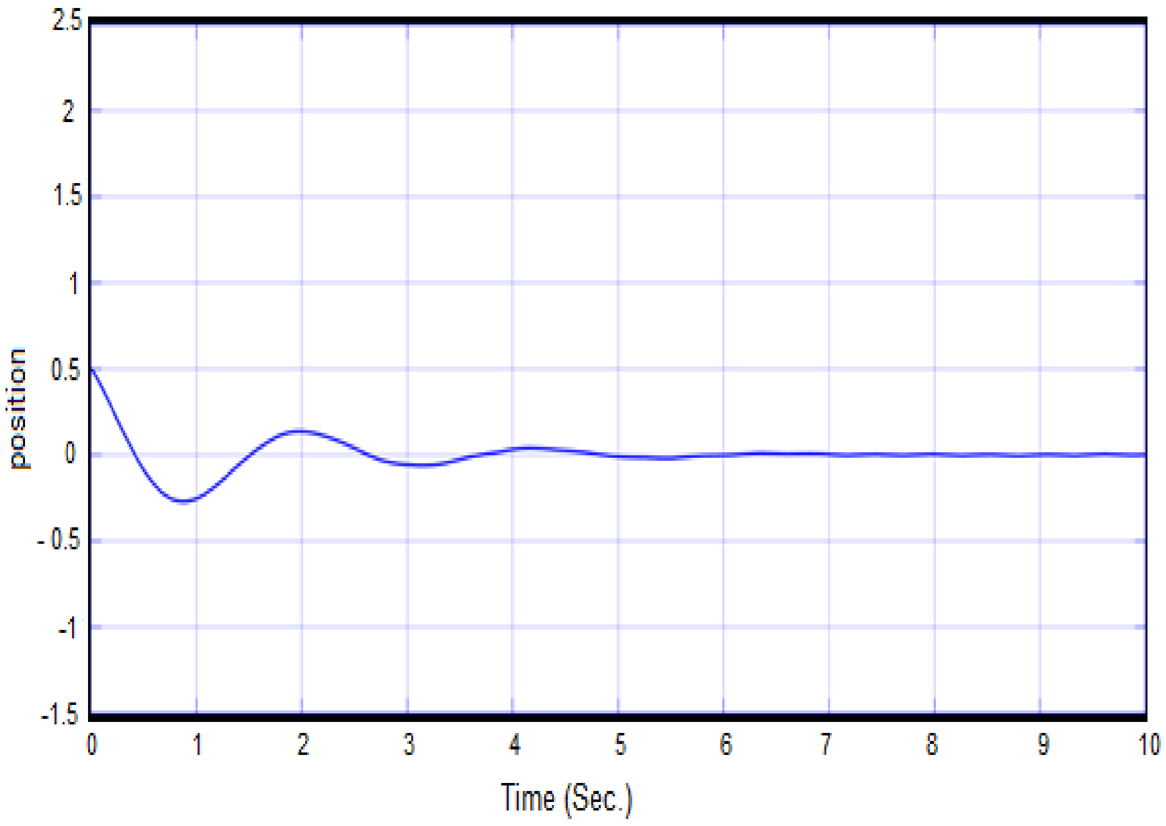

Figure 4.7: The output response of the robot using PID

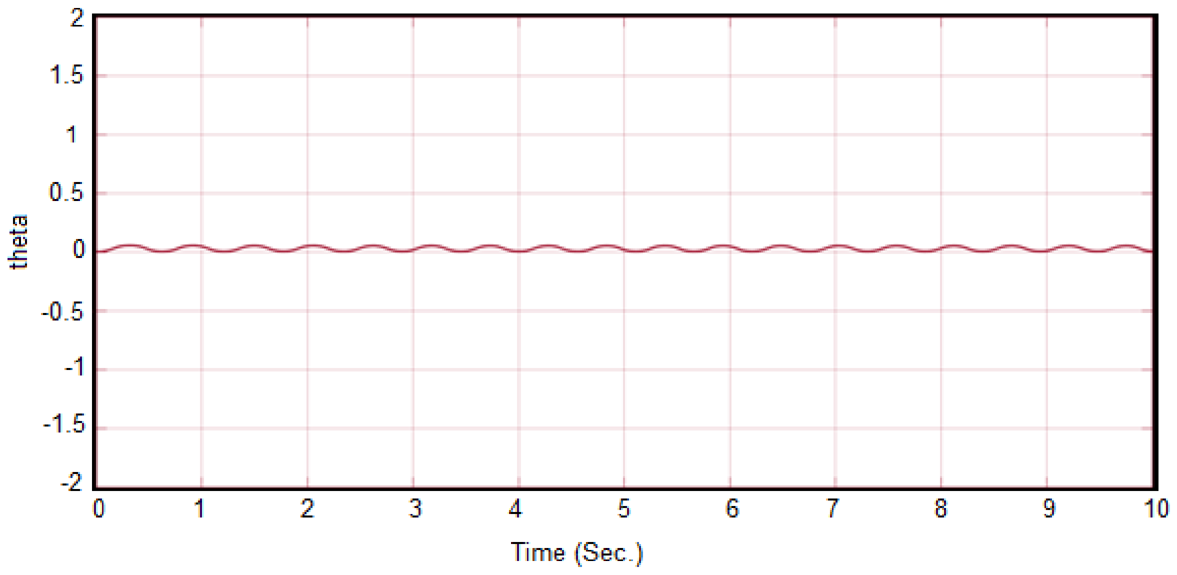

Figure 4.8: Output of the angle (θ)

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The two wheeled robot system provides higher level of mobility and maneuverability. The application of the two wheeled balancing robot varies with environment and requirements. Understanding the classical theory of inverted pendulum and its dynamic system are initial steps for developing two wheeled robot. Safety and robustness of two wheeled products are so important and these aspects are achieved by employing accurate sensor and designing a precise control unit. This study has resulted in building emphasis a working balancing mobile robot on two wheels. The robot successfully balancing and driven using PID controller digital control algorithm optioned for solving inverted pendulum controller system problem with reduced oscillations and improved stability.

5.2 Recommendations

There are two different ways to begin develop this study. One is take a few steps back and start with evaluating the mathematical model more closely and the gather even more knowledge about the most simple balancing systems. In this step the work can go on and after a while reach more complex and interesting control systems and more relevant results and functions. Another would be to evaluate the robot as it is right now and see how the current modeling, programming, construction and tuning could be improved. When the robot is functioning as expected and the goals of the project have been fulfilled, there are plenty of ways to continue the study. Can be used many types for controlling this study LQR, fuzzy logic, pole-placement and neural networks.

REFERENCES

[1] A. Demetlika, “Self-Balancing Mobile Robot Tilter”, Vol. 3, pp. 23-32, 2012.

[2] J. R. Leigh, “Control Theory - A Guided Tour- Institution of Engineering and Technology Control”, London, 3rd.Ed; 2012.

[3]T. Glad and L. Ljung, “Control Theory: Multivariable and Nonlinear Methods”, Taylor and Francis, 1st.Ed; 2000.

[4] R. C. Rorf and R. H. Bishop, “Modern Control Sytems”, 10th.Ed; New Jersey, 2015.

[5] Thomas. R.Kurfess, “Robotics and Automation Handbook”, Toylar and Francis, 2005.

[6] M. W. Spong, S.Hutchinson and M. Vidyasagar, “Robot Modeling and Control”, Wiley, 2006.

[7] Johan. W. Webb and Ronald. A.Resis, “Programmable Logic Controllers: Principles and Applications”, Prentice Hall, 4th.Ed; 1999.

[8] A. N. K. Nasir, M .Z. M. Tumari, and M. R. Ghazali, “Performance Comparison between Sliding Mode Controller and PID Controller for a Highly Nonlinear Two-Wheeled Balancing Robot”, pp. 1403-1408, Nov. 2012.

[9] R. M. Miller, “A Long Range Predictive PID Controller with Application to an Industrial Chemical Process”, IEEE Xplore Digital Library. 1997.

[10] S. Kalra, D. Patel and K. Stol, “Design and Hybrid Control of a Two wheeled Robotic Platform”, New Zealand, 2004.

[11] B. Mahler and J. Haase, “Mathematical Model and Control Strategy of Two Wheeled Self Balancing Robot”, Germany, 2013.

[12] F. Grasser. D' arrigo, A. Colombi. S and A. C. Rufer, “A Mobile Inverted Pendulum”, IEEE Transactions on Industrial Electronic, Vol. 49, No.1, pp. 107- 114, Feb. 2002.

[13] W. An, Y. Li and S. Member, “Simulation and Control of a Two Wheeled Self Balancing Robot”, pp. 456-461, Dec. 2013.

APPENDEX

THE CODE

```
#include <PID_v1.h>

#include <LMotorController.h>

#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE

#include "Wire.h"

#endif

#define MIN_ABS_SPEED 20

MPU6050 mpu;

// MPU control/status vars

booldmpReady = false; // set true if DMP init was successful

uint8_tmpuIntStatus; // holds actual interrupt status byte from MPU

uint8_tdevStatus; // return status after each device operation (0 = success, !0 = error)

uint16_tpacketSize; // expected DMP packet size (default is 42 bytes)

uint16_tfifoCount; // count of all bytes currently in FIFO

uint8_tfifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars

Quaternion q; // [w, x, y, z] quaternion container

VectorFloat gravity; // [x, y, z] gravity vector

floatypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector


//PID

doubleoriginalSetpoint = 179.7;
```

```
doublesetpoint = originalSetpoint;
doublemovingAngleOffset = 0.1;
double input, output;


//adjust these values to fit your own design
doubleKp = 60;
doubleKd = 1.9;
double Ki = 200;
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);


doublemotorSpeedFactorLeft = 0.6;
doublemotorSpeedFactorRight = 0.6;
//MOTOR CONTROLLER
int ENA = 3;
int IN1 = 5;
int IN2 = 4;
int IN3 = 8;
int IN4 = 7;
int ENB = 6;
LMotorControllermotorController(ENA, IN1, IN2, ENB, IN3, IN4, motorSpeedFactorLeft,
motorSpeedFactorRight);


volatileboolmpuInterrupt = false; // indicates whether MPU interrupt pin has gone high
voiddmpDataReady()
{
mpuInterrupt = true;
```

```
}

void setup()

{

// join I2C bus (I2Cdev library doesn't do this automatically)

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE

Wire.begin();

TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)

#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE

Fastwire::setup(400, true);

#endif

Serial.begin(9600);

mpu.initialize();

devStatus = mpu.dmpInitialize();

// supply your own gyro offsets here, scaled for min sensitivity

mpu.setXGyroOffset(34);

mpu.setYGyroOffset(34);

mpu.setZGyroOffset(-14);

mpu.setZAccelOffset(1135); // 1688 factory default for my test chip

// make sure it worked (returns 0 if so)

if (devStatus == 0)

{

// turn on the DMP, now that it's ready

mpu.setDMPEnabled(true);

// enable Arduino interrupt detection

attachInterrupt(0, dmpDataReady, RISING);
mpuIntStatus = mpu.getIntStatus();

// set our DMP Ready flag so the main loop() function knows it's okay to use it

dmpReady = true;

// get expected DMP packet size for later comparison

packetSize = mpu.dmpGetFIFOPacketSize();

//setup PID

pid.SetMode(AUTOMATIC);

pid.SetSampleTime(15);

pid.SetOutputLimits(-200, 200);

}

else

{

// ERROR!

// 1 = initial memory load failed

// 2 = DMP configuration updates failed

// (if it's going to break, usually the code will be 1)

Serial.print(F("DMP Initialization failed (code "));

Serial.print(devStatus);

Serial.println(F(")"));

}

}

void loop()

{

// if programming failed, don't try to do anything

if (!dmpReady) return;
```

```
// wait for MPU interrupt or extra packet(s) available

while (!mpuInterrupt&&fifoCount<packetSize)

{

//no mpu data - performing PID calculations and output to motors

pid.Compute();

motorController.move(output, MIN_ABS_SPEED);

}

// reset interrupt flag and get INT_STATUS byte

mpuInterrupt = false;

mpuIntStatus = mpu.getIntStatus();

// get current FIFO count

fifoCount = mpu.getFIFOCount();

// check for overflow (this should never happen unless our code is too inefficient)

if ((mpuIntStatus& 0x10) || fifoCount == 1024)

{

// reset so we can continue cleanly

mpu.resetFIFO();

Serial.println(F("FIFO overflow!"));

// otherwise, check for DMP data ready interrupt (this should happen frequently)

}

else if (mpuIntStatus& 0x02)

{

// wait for correct available data length, should be a VERY short wait

while (fifoCount<packetSize) fifoCount = mpu.getFIFOCount();

// read a packet from FIFO
mpu.getFIFOBytes(fifoBuffer, packetSize);

// track FIFO count here in case there is > 1 packet available

// (this lets us immediately read more without waiting for an interrupt)

fifoCount -= packetSize;

mpu.dmpGetQuaternion(&q, fifoBuffer);

mpu.dmpGetGravity(&gravity, &q);

mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

input = ypr[1] * 180/M_PI + 180;

Serial.println(input);

}

}
```