FINAL YEAR PROJECT REPORT
“Design and Implementation of Control System for Inverted Pendulum”

Project Advisor
Asst. Prof. Saleem Ata

Co-Advisor
Asst. Prof. Jameel Ahmad

Submitted by

(Raheel Tariq	ID: 081220-089)
(Sajid Ali	ID: 081220-097)
(Saad Rasheed	ID: 081220-130)

Department of Electrical Engineering,
School of Science and Technology,
University of Management and Technology,
Lahore, Islamic Republic Pakistan

Abstract

“Design and Implementation of Control System for Inverted Pendulum”

Project Report submitted to the
Department of Electrical Engineering, University of Management and Technology
in partial fulfillment of the requirements for the degree of
Bachelor of Science
in
Electrical Engineering

(Raheel Tariq	ID: 081220-089)
(Sajid Ali	ID: 081220-097)
(Saad Rasheed	ID: 081220-130)

The inverted pendulum is a classic control problem, which involves the development of balance the pendulum system. For display purposes, it is like trying to balance broomstick on the finger. To study this problem, we have to take into account the experimental system which consists of a pendulum, which is free to rotate 360 degrees. This project involves the complete system including all mechanical hardware, software and design at minimal cost. Three major subsystems that make this design: (1) a mechanical system, (2) feedback network which includes sensors and methods for reading and (3) a driver and its interface with mechanical system. After determining a set of requirements between subsystems, each can be designed independently of the other two, simplifying the design process. We consider several models of stick-slip friction between the car and track and measure the parameters of the experimental friction.

Keywords: inverted pendulum, feedback, friction, interfaces.

Summary

Classic inverted pendulum control problem is of interest that can be solved using a variety of systems and solutions. This problem is similar to trying to balance the broomstick practice. This problem invites those who are interested in the design of the system, control theory, and just plain problem solving try to develop a working system. For this project, the motivation is to translate the mathematical models developed in control theory teaching real-time system.

The design process of this project has required a great deal of planning and testing before deciding on a final design because there were so many alternatives to choose from. The design problem can be broken down into subsystems that are highly interdependent. The actual design of each subsystem is an iterative process of component testing and implementation of simple solutions until the solution "optimal" was found. The solution "optimum" may be defined as the solution which can be developed at minimum cost that balances the pendulum better.

During the design process, many obstacles have been encountered. The setbacks of the most important and time-all related to engine choice. When the first motor has been omitted, an engine torque faster and higher was obtained because the original one was not able to react fast enough in order to balance the pendulum. Since the new engine has different dimensions than the first, the entire mechanical system had to be redesigned. The next two engines broke when the control effort was too high and the direction changing rapidly, putting too much stress on the internal gears. To solve this problem, the controller has been redesigned with a closer bond on the control effort.

The model is obtained on the basis of mathematical analysis of the physical system. Unknown parameters of the model are obtained from experiments in real time on the system inverted pendulum. The model was created with respect to the most nonlinearity in the system. Linearity is caused by fundamental principles of the system and by friction between the individual parts of the system. Therefore, the model is highly nonlinear and then linearization around the point of work has been performed and continuous linearized model was calculated as well as its discrete version. A controller has been constructed that has also been verified by experiments in real time.

CONTENTS:

INSIDE THE REPORT

Table of Contents:

Design and implementation of Control System for Inverted Pendulum

CHAPTER 0:

LIST OF FIGURES

CHAPTER 1:

INTRODUCTION

Introduction

The uniqueness and the widespread application of technologies derived from this unstable system has attracted interest from a lot of research and robotics enthusiasts from around the world an inverted pendulum is a pendulum which has its mass above its pivot point. It is often realized with the pin mounted on a carriage which can move horizontally and may be called a carriage and pole. The inverted pendulum is a classic problem in dynamics and control theory and is widely used as a benchmark for testing control algorithms (PID, neural networks, fuzzy control, genetic algorithms etc.). Variations on this problem include multiple links, allowing the movement of the trolley to be controlled while maintaining the pendulum. The objective of this project is to build a controller that is able to perform the balancing act with the available configuration, preferably with minimum of bike setup. To achieve these different steps are gone through.

The practical part of this project was performed at the University of Management And Technology, Lahore Pakistan, at the Department of Electrical Engineering, under the supervision of Asst. Prof. Saleem Ata and Asst. Prof. Jameel Ahmad. The final report was written in the Pakistan, under the supervision of Asst. Prof. Jameel Ahmad.

1.1 Background

The experimental system we study is illustrated schematically in Figure 1. In this system, a pendulum is connected to the side of a carriage by means of a pivot that allows the pendulum to swing in the xy plane. A force F (t) is applied to the cart in the x direction, with the aim of keeping pendulum balanced upright. This system is adherently unstable since even the minimal disturbance may cause the pendulum began to fall. Thus a sort of control is necessary to maintain a balanced pendulum. A controller would be ideal to keep the pendulum balanced with very little change in angle, θ, or moving cart, x. This system is adherently unstable since even the slightest disturbance can cause the pendulum will start to fall. Therefore a kind of control is necessary to maintain a balanced pendulum. A controller would be ideal to keep the pendulum in balance with very little change in angle θ or moving cart x. Obviously, limitation would be required under current parameters of the system and the method for implementing a controller.

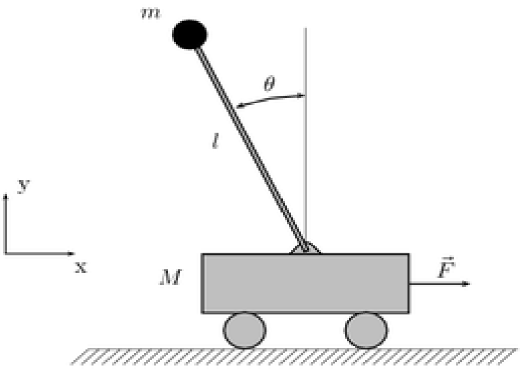

Figure1. Inverted Pendulum System with an input force

CHAPTER 2:

PROBLEM SETUP

2.1 Design Problem and Requirement

2.1.1 The Problem

The objective of this project is to design a mechanical system for the problem of the inverted pendulum, and then implement a controller feasible. The controller should minimize both the displacement of the carriage and the angle of the pendulum. The system should be standalone.

2.1.2 The Requirements

To design any system, you need to have guidelines when making decisions on the implementation. This project is no different. A set of requirements and goals for this project was set up as follows:

- ✓ Settling time of less than 5 seconds
- ✓ Pendulum angle never more than 0.05 radians from the vertical
- ✓ Working mechanical system
- ✓ Choice of accurate sensors
- ✓ Nonlinear Control (if time permits).
- ✓ Wireless Sensors (if time permits).

The progress of the project it is expected that some of these requirements will change as necessary. A revised list of requirements will be included at the end of the section of the project that illustrates what has actually been achieved.

2.2 Problem Definition

It is practically impossible for the pendulum to balance in the inverted position without applying an external force to the system. The system shown in Figure 1 allows this control force to be applied to the cart of the IP. The outputs from basket IP can be:

- ➢ Cart position
- ➢ Cart velocity
- ➢ Pendulum angle
- ➢ Pendulum angular velocity

Dynamic modeling of the inverted pendulum is a simple task. Here are some of the variables that must be examined in order to model the system:

- Bases' position
- Bases' velocity
- Bases' acceleration
- Bases' moment of inertia
- Bases' coefficient of friction
- Bases' mass
- Bases' length
- Earth's gravitational constant
- Pendulum's position
- Pendulum's velocity
- Pendulum's moment of inertia
- Pendulum's coefficient of friction
- Pendulum's mass
- Pendulum's length

In our case, since the requirement is to linearize the inverted pendulum around the corner. Therefore, the angle will be fed to the controller which will control our IP in a vertical position. The purpose of the study is to stabilize the pendulum so that the position of the carriage is controlled quickly and accurately, and the pendulum is always try to maintain its upright vertical position during such movements.

This problem affects a cart and pendulum hinged to the bottom of the length of the pendulum can move in the same plane cart. That is, pendulum mounted on the carriage is free to fall along the carriage axis of movement. This system must be controlled so that the pendulum remains balanced and vertical and resistant to the passage disorder. As the change of a part of the system results in change in the modification other hand, this is a control system more complex than appears at first sight.

If the exit angle of the pendulum is on the vertical axis, we realize that the system is unstable. The pendulum will fall if it is released at an angle very small, too. To stabilize the system, a system of feedback control is required.

Briefly, the inverted pendulum is constituted by a trolley and pendulum. The control objective is to move the carriage to the commanded position without causing the pendulum to fall over. Open-loop system is purely unstable. This configuration can be used to study the control of the open loop system unstable. It is demonstration of the benefits of stabilizing feedback control. A range of control techniques ranging from simple phase-lead compensator to neural network-controller can be applied.

2.3 Range of Possible Solutions

An inverted pendulum is a classic example of control. The process is non-linear and unstable with a different input and output. The objective is to balance the system vertically on the trolley. There are several approaches to solve the system in terms of equations of motion. Following is brief description of some approaches to understand and solve the system.

Method 1:

Observer is standing at the position of the carriage is to say, inside the carriage and observing the motion of the pendulum not of the trolley. This is the simplest approach possible.

Method 2:

Observer is standing outside the system and collectively observing the motion and equilibrium state of the system. A much more complex method to understand and streamline the system. He needs a deep understanding of non-linear physics.

Method 3:

Observer is standing outside the system and observing the system as free-body diagrams involved in the inverted pendulum system. In our case, there are a total of two free-body diagrams that cart and pendulum.

A classical and best approach to solve this system because of free body diagrams.

CHAPTER 3:

MATHEMATICAL ANALYSIS

MATHEMATICAL ANALYSIS

The inverted pendulum is mounted on a movable carriage. A servo motor is controlling the movement of translation of the carriage, through a rail mechanism. That is, the carriage is coupled with a servo-current motor. The engine is derived from servo electronics; circuits process controller of the error signal, which then pushes the cart through the servomotor and the driving rail mechanism. To-fro movement of the carriage applies moments on the inverted pendulum and therefore keeps the pendulum in a vertical position.

3.1 Force Analysis and System Equation Setup

The inverted pendulum is a system that has a carriage which is programmed to balance a pendulum. The process is non-linear and unstable with a signal input and output different.

3.1.1 Modeling Assumptions

A first approach to modeling would assume a frictionless with every movement limited to the degrees of freedom you want. This might not seem like the right approach because the system has in fact friction. But if, ignoring friction can result in some change in the output of the system, this will not give output as desired. Therefore, ignoring friction coefficient is not influencing the dynamics of the system to a large extent.

System Dynamics

μ (t):	Input force
M:	Mass of cart
M:	Mass of pendulum, point mass
x:	Displacement of cart
θ :	Displacement of pendulum
g:	Gravitational Force
l:	Length of pendulum
$\theta(t)$:	Reference angle
$\frac{dy}{dx}x$:	Velocity of cart
$\ddot{x}$:	Acceleration of the cart

3.1.2 Inverted Pendulum System Equations

The actual design of the controller took a few steps. The first system will be modeled with the assumptions required such that the system is linear and fairly simple to design a controller. From this, the model can be converted to state space form, which can then be used to design a controller.

Nonlinear equations can be derived as follows. First we assume that the rod is massless and that the cart mass and the point mass at the upper end of the inverted pendulum are denoted as M and m, respectively. There is an externally x-directed force on the cart, u(t), and a gravity force acts on the point mass at all times

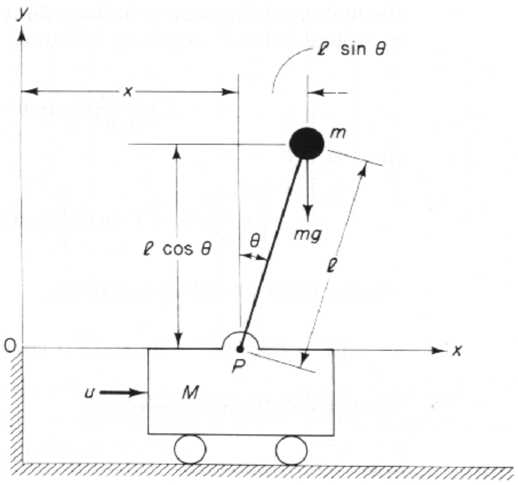

Figure 2: Analysis Inverted Pendulum System

According to Newton's second law, force is equal to mass times acceleration. Summing the forces in the free body diagram of the cart in the horizontal direction, we obtain the following equation of motion

$$M\ddot{x}+m\ddot{x}_g=\mu$$

This can be written as following:

$$\mu = M\frac{d^2}{dt^2}x+m\frac{d^2}{dt^2}x_g \rightarrow eq(1)$$

Where time-dependent center of gravity of the mass point is given by the coordinates for the point of mass here assumed, the position of the center of gravity of the mass of the pendulum is simply:

$$\left.\begin{cases} x_a = x + l\sin\theta \\ y_g = l\cos\theta \end{cases}\right\} \rightarrow eq(z)$$

Substituting of location of center of gravity Eq (z) in Eq (1) results

$$M\ \frac{d^2}{dt^2}x + m\,\frac{d^2}{dt^2}(x + l\sin\theta) = \mu \rightarrow eq(2)$$

Alternatively,

$$M\ddot{x} + ml\frac{d^2}{dt^2}\sin\theta + m\ddot{x} = \mu$$

$$\ddot{x}(M+m) + ml\frac{d^2}{dt^2}\sin\theta = \mu$$

And general geometric terms are

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

  &

- $\frac{d}{dt}\sin\theta = -(\sin\theta)\dot{\theta}^2 + (\cos\theta)\ddot{\theta} \rightarrow eq(3)$
- $\frac{d}{dt}\cos\theta = -(\sin\theta)\dot{\theta}$

  &

- $\frac{d}{dt}\cos\theta = -(\cos\theta)\dot{\theta}^2 - (\sin\theta)\ddot{\theta} \rightarrow eq(4)$

Using the geometric terms here, we have the following result

$$\ddot{x}(M+m) + ml\sin\theta.\dot{\theta}^2 + ml\cos\theta.\ddot{\theta} = \mu \rightarrow eq(5)$$

For equilibrium, net torque effect should be balanced on the system. $\mathrm{T}_g = \mathrm{T}_a$

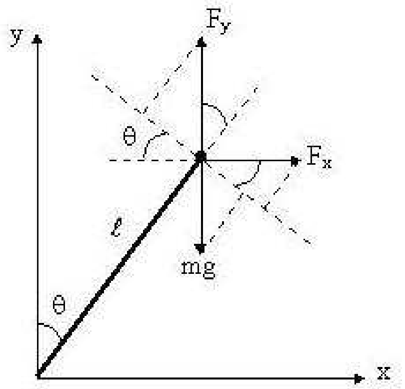

Figure 3: Vector Diagram for Force Components in Torque Balance

In a similar way, you run a balance of torque on the system, where the torque is the product of the perpendicular component of the force and the distance from the point of rotation. In this case, the torque on the mass due to the acceleration force is balanced by the torque on the mass due to the force of gravity:

$$(F_x \cos\theta)l - (F_y \sin\theta)l = (F_g \sin\theta) \rightarrow eq(6)$$

Where the force components, Fx and Fy are determined from equation (z) and equation (4)

$$F_x = m\frac{d^2}{dt^2}x_g = m\left[\ddot{x} - l(\sin\theta)\dot{\theta}^2 + l(\cos\theta)\ddot{\theta}\right] \rightarrow eq(b)$$

$$F_y = m\frac{d^2}{dt^2}y_g = -m\left[l\cos\theta\dot{\theta}^2 + l\sin\theta\ddot{\theta}\right] \rightarrow eq(c)$$

Substituting these expressions into eqn. (6) and simplifying results,

$$m\ddot{x}\cos\theta + ml\ddot{\theta} = mg\sin\theta \rightarrow eq(7)$$

Therefore the equations that define this system are given by Eqns. (5) and (7). These equations certainly represent a non-linear system that is relatively complicated from the mathematical point of view. However, since the objective of this particular system is to keep the inverted pendulum erected around $\theta = 0^o$, one could consider the linearization around the equilibrium point in a vertical position. We'll do it later and actually compare the linear and nonlinear dynamics of the system, but first, we need to put the non-linear equations in standard form state space.

3.3 Nonlinear State Equations (Inverted Pendulum)

To numerically simulate the nonlinear model for the inverted pendulum we need to put it into standard state form,

$$\frac{d}{dt}\underline{z} = \underline{f}(z,u,t) \rightarrow eq(8)$$

To put equations. (5) and (7) into this form, let's first manipulate the equations algebraically to only have a single second derivative term in each equation. From eqn. (7), we have

$$m\ddot{x}\cos\theta + ml\ddot{\theta} = mg\sin\theta$$

and putting this into eqn. (5) gives

$$\ddot{x}(M+m) - ml\sin\theta.\dot{\theta}^2 + \cos\theta\left[mg\sin\theta - \cos\theta.m\ddot{x}\right] = \mu$$

$$\ddot{x}(M+m) - ml\sin\theta.\dot{\theta}^2 + mg\cos\theta\sin\theta - \cos^2\theta m\ddot{x} = \mu$$

$$\ddot{x}(M+m) - \cos^2\theta m\ddot{x} = \mu + ml\sin\theta.\dot{\theta}^2 - mg\cos\theta\sin\theta$$

$$\ddot{x}(M+m-\cos^2\theta.m) = \mu + ml\sin\theta.\dot{\theta}^2 - mg\cos\theta\sin\theta \rightarrow eq(9)$$

From relation

$$ml\ddot{\theta} = mg\sin\theta - m\ddot{x}\cos\theta$$

$$\ddot{x} = \frac{g\sin\theta - l\ddot{\theta}}{\cos\theta} \rightarrow eq(x)$$

We have Eq (5)

$$\ddot{x}(M+m) - ml\sin\theta.\dot{\theta}^2 + \cos\theta\left[mg\sin\theta - \cos\theta.m\ddot{x}\right] = \mu \rightarrow eq(5)$$

Substitute Eq (x) in Eq (5)

$$(M+m)\left[\frac{g\sin\theta - l\ddot{\theta}}{\cos\theta}\right] - ml\sin\theta.\dot{\theta}^2 + ml\cos\theta.\ddot{\theta} = \mu$$

Multiplying "$\cos\theta$" on both sides

$$(M+m)\left[g\sin\theta - l\ddot{\theta}\right] - ml\sin\theta.\dot{\theta}^2.\cos\theta + ml\cos^2\theta.\ddot{\theta} = \mu\cos\theta \rightarrow eq(c)$$

Or

$$\ddot{\theta}(ml\cos^2\theta - (M+m)l) = \mu\cos\theta - (M+m)g\sin\theta + ml\cos\theta\sin\theta\dot{\theta}^2 \rightarrow eq(10)$$

Finally, dividing by the lead coefficients of eqns. (9) and (10) gives

$$\ddot{x} = \frac{\mu + ml\sin\theta\dot{\theta}^2 - mg\cos\theta\sin\theta}{M + m - m\cos^2\theta} \rightarrow eq(11)$$

And

$$\ddot{\theta} = \frac{\mu\cos\theta - (M+m)g\sin\theta + ml(\cos\theta\sin\theta)\dot{\theta}}{ml\cos^2\theta - (M+m)l} \rightarrow eq(12)$$

To put the nonlinear equations into standard state space form, we assume

$$z_1 = \theta \quad z_2 = \dot{\theta} = \dot{z}_1$$
$$z_3 = x \quad z_4 = \dot{x} = \dot{z}_3$$

Since the general form is

$$\frac{d}{dt}\underline{z} = \underline{f}(z,u,t)$$

∴ For final state space equations

$$\frac{d}{dt}\underline{z} = \frac{d}{dt}\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} \rightarrow eq(13)$$

We have equation (12),

$$\ddot{\theta} = \frac{\mu\cos\theta - (M+m)g\sin\theta + ml(\cos\theta\sin\theta)\dot{\theta}}{ml\cos^2\theta - (M+m)l}$$

Substituting the assumptions in equation

$$\Rightarrow \ddot{\theta} = \frac{\mu\cos z_1 - (M+m)g\sin z_1 + ml(\cos z_1\sin z_1)z_2{}^2}{ml\cos^2 z_1 - (M+m)l}$$

Again, substituting these assumptions in equation(11) and simplifying,

$$\Rightarrow \ddot{x} = \frac{\mu + ml(\sin z_1)z_2{}^2 - mg\cos z_1\sin z_1}{M + m - m\cos^2 z_1}$$

“Simply playing the derivatives”

$$\frac{d}{dt}\begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} z^2 \\ \dfrac{\mu\cos z_1 - (M+m)g\sin z_1 + ml(\cos z_1\sin z_1)z_2{}^2}{ml\cos^2 z_1 - (M+m)l} \\ z_4 \\ \dfrac{\mu + ml(\sin z_1)z_2{}^2 - mg\cos z_1\sin z_1}{M + m - m\cos^2 z_1} \end{bmatrix} \rightarrow eq(14)$$

This expression is now in the desired form given in equation. (7.58). If both the pendulum angle and the cart position x are of interest, we have

$$\underline{Y} = \underline{\underline{c}}z$$

Or

$$\underline{Y} = \underline{\underline{c}}z = \begin{bmatrix} \frac{1}{0}\frac{0}{0}\frac{0}{1}\frac{0}{0} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} \rightarrow eq(15)$$

Equations (14) and (15) give a complete state space representation of the nonlinear inverted pendulum. This is the system implemented in the subsequent Matlab simulation

3.4 Linear State Equations (Inverted Pendulum)

If you want a system linearized around the point fixed, simply "linearize" the nonlinear system given in eq (14). Since the normal matrices are zero in this case (i.e everything is put in non-linear vector function, $\underline{f}(\underline{z},\underline{u},\underline{t})$), with coefficient matrix Jacobian to make our equation a differential algebraic equations for the linearized form the system becomes,

$$\frac{d}{dt}\delta\underline{z} = \underline{\underline{J}}_z(z_\circ,u_\circ)\delta\underline{z} + \underline{\underline{J}}_u(z_\circ,u_\circ)\delta u \rightarrow eq(16)$$

Where the reference state is defined by the stationary pendulum and built with no power input. In these conditions, $z_o = 0, u_o = 0$. As a nonlinear function vector is complicated to be solved, determine the components of Jacobean matrices term by term.

$$\underline{\underline{J}}_z(z_\circ,u_\circ) = \begin{bmatrix} \frac{Jf_1}{Jz_1} & \frac{Jf_1}{Jz_2} & \frac{Jf_1}{Jz_3} & \frac{Jf_1}{Jz_4} \\ \frac{Jf_2}{Jz_1} & \frac{Jf_2}{Jz_2} & \frac{Jf_2}{Jz_3} & \frac{Jf_2}{Jz_4} \\ \frac{Jf_3}{Jz_1} & \frac{Jf_3}{Jz_2} & \frac{Jf_3}{Jz_3} & \frac{Jf_3}{Jz_4} \\ \frac{Jf_4}{Jz_1} & \frac{Jf_4}{Jz_2} & \frac{Jf_4}{Jz_3} & \frac{Jf_4}{Jz_4} \end{bmatrix}$$

Now determining the elements of jacobian matrices

$$\left.\frac{Jf_1}{Jz_1}\right|_{(z_\circ,u_\circ)} = \frac{J}{Jz_1} z_2 = 0$$

Using quotient rule’s result and simplifying,

$$\left.\frac{Jf_2}{Jz_1}\right|_{z_\circ,u_\circ} = \frac{(M+m)g}{Ml}$$

And

$$\left.\frac{Jf_3}{Jz_1}\right|_{z_\circ,u_\circ} = \frac{J}{Jz} z_4 = 0$$

Since

$$\frac{Jf_4}{Jz_1} = \frac{J}{Jz}\left(\frac{u + ml(\sin z_1)z_2{}^2 - mg\cos z_1 \sin z_1}{M + m - m\cos^2 z_1}\right)$$

Simplifying these expressions and re-arranging results into following

$$\left.\frac{Jf_4}{Jz_1}\right|_{z_\circ,u_\circ} = \frac{-mg}{M+m-m} \Rightarrow \frac{-mg}{M}$$

$$\left.\frac{Jf_1}{Jz_2}\right|_{(z_\circ,u_\circ)} = \frac{J}{Jz} z_2 = 1$$

$$\left.\frac{Jf_3}{Jz_4}\right|_{\underline{z}_\circ,u_\circ} = {}^{J}\!\big/_{Jz}(z_4) = 1$$

Thus, combining all these separate terms gives System

$$\underline{\underline{J}}_z(z_\circ,u_\circ) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{(M+m)g}{Ml} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{mg}{M} & 0 & 0 & 0 \end{bmatrix} \qquad eq(17)$$

Now derivative of the non linear terms with respect to μ

$$\underline{J}\left(\underline{z}_{\circ}, u_{\circ}\right) = \begin{bmatrix} \frac{Jf_1}{Ju} \\ \frac{Jf_2}{Ju} \\ \frac{Jf_3}{Ju} \\ \frac{Jf_4}{Ju} \end{bmatrix}$$

Now determining the elements of Jacobian matrices μ

$$\left.\frac{Jf_1}{Ju}\right|_{(z_{\circ}, u_{\circ})} = {}^{J}\!\!\diagup_{Ju}(z_2) = 1$$

$$\frac{Jf_2}{Ju} = {}^{J}\!\!\diagup_{Ju}\left(\frac{u\cos z_1 - (M+m)g\sin z_1 + ml(\cos z_1.\sin z_1)z_2^2}{ml\cos^2 z_1 - (M+m)l}\right) = \frac{-1}{Ml}$$

$$\left.\frac{Jf_4}{Ju}\right|_{\underline{z}_{\circ}, u_{\circ}} = \frac{J}{Ju}\left(\frac{u + ml(\sin z_1)z_2^2 - mg\cos z_1 \sin z_1}{M + m - m\cos^2 z_1}\right) = \frac{1}{M}$$

Re-arranging all the derivative terms w.r.t input force, we have

$$\underline{J}\left(\underline{z}_{\circ}, u_{\circ}\right) = \begin{bmatrix} 0 \\ \frac{-1}{Ml} \\ 0 \\ \frac{1}{M} \end{bmatrix} \qquad eq(18)$$

Finally, after all these manipulations we can write eqn. (16) explicitly as

$$\frac{d}{dt}\delta\underline{z} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{(M+m)g}{Ml} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{mg}{M} & 0 & 0 & 0 \end{bmatrix}\delta\underline{z} + \begin{bmatrix} 0 \\ \frac{-1}{Ml} \\ 0 \\ \frac{1}{M} \end{bmatrix} \qquad eq(19)$$

What is the form of LTI state standards required for the implementation in Matlab. Equation (19) together with the definition given in response eqn. (15) represents the final linear model of inverted pendulum system.

3.5 SYSTEM PARAMETERS

For the project under consideration, we will have following parameters of the inverted-pendulum system,

M	Mass of the Cart	*900 gm*
M	Mass of the Pendulum	*100 gm*
B	Friction of the Cart	*0.000 N/m/sec*
L	Length of pendulum to Center of Gravity	*23.5 cm*
I	Moment of Inertia (Pendulum)	*5.3 gm-m2*
R	Radius of Pulley,	*2.3 cm*
τ_m	Time Constant of motor	*0.5 second*
K_m	Gain of Motor	*17 rad/sec/V*
K_F	Gain of Feedback	*9/p V/rad/sec*
F	Force applied to the cart	
X	Cart Position Coordinate	
Q	Pendulum Angle with the vertical	

CHAPTER 4:

ANALYSIS OF UNCOMPENSATED SYSTEM

ANALYSIS OF UNCOMPENSATED SYSTEM

4.1 Pole Zero Map of Open Loop System

The poles position of the linearized model of the inverted pendulum (in open-loop configuration) indicates that the system is unstable, as one of the poles of the transfer function is located on the right half of the s-plane. Thus the system is absolutely unstable.

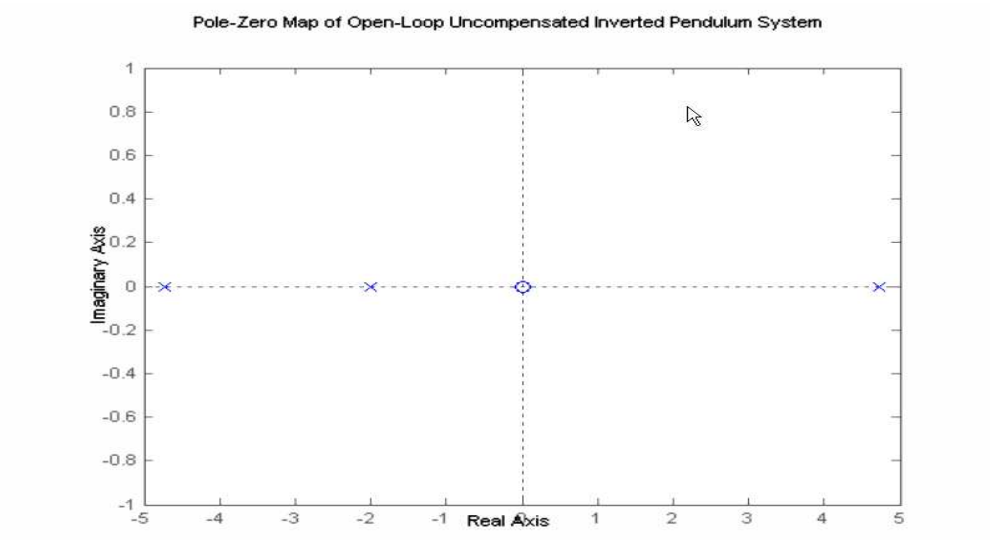

Figure 4: Pole Zero Map of Open Loop System

4.2 Impulse Response of Open Loop System

An impulse response of the system is shown in following figure. The system is highly unstable as theta diverges very rapidly. The runaway nature of the response indicates instability.

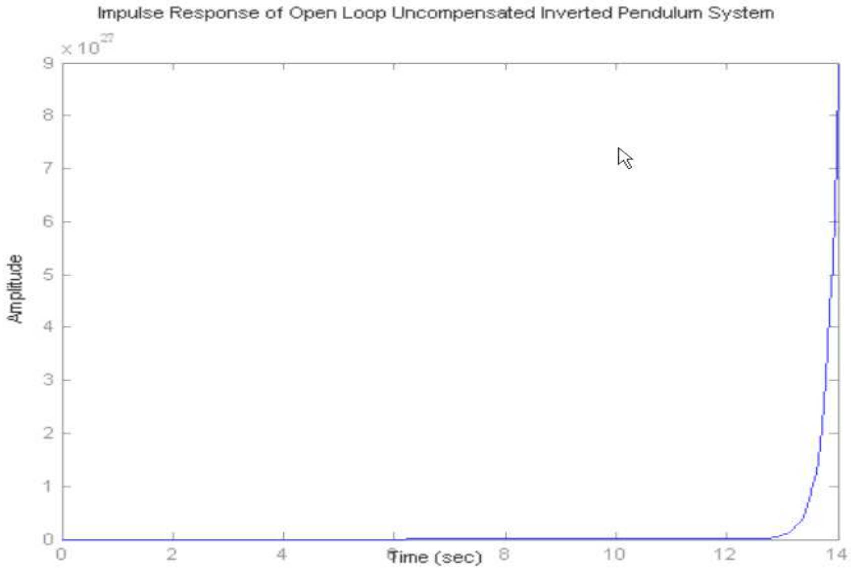

Figure 5: Impulse Response of Open Loop System

4.3 Step Response of Open Loop System

A step response of the system is illustrated in the following figure. Even here, theta diverges very quickly the system is highly unstable. The nature of the response Runaway indicates instability.

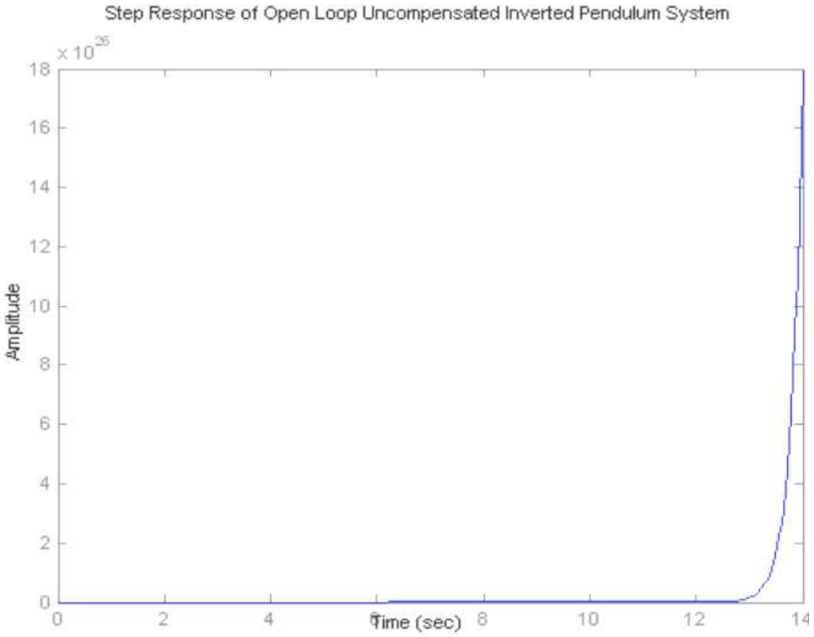

Figure 6: Step Response of Open Loop System

4.4 Root Locus of Open Loop System

The first step in the design of compensation for each plant is to observe the response of the unit closed feedback loop to control the stability. Many systems are unstable in open loop stable but in closed loop configuration. Vice versa is also possible that the system is stable in open loop but unstable in closed loop, although this case is rare. The closed-loop system can be studied viewing uncompensated root locus of the system graphic Ofthe. The following figure shows a plot the root locus of the system.

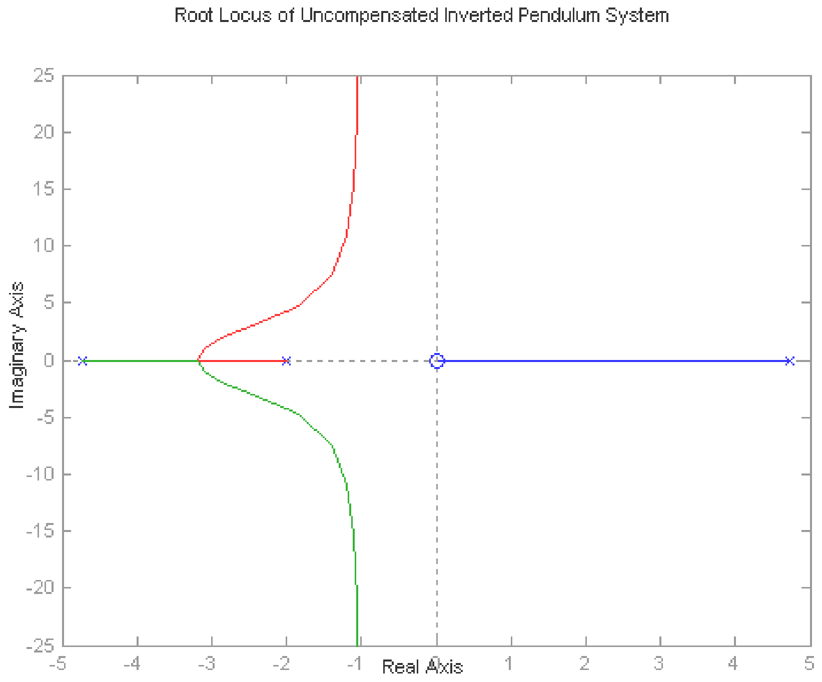

Figure 7: Root Locus of Open Loop System

The plot reveals that the system cannot be controlled by a simple loop unity feedback. Whatever the value of loop gain, ("K), a branch of the locus remains RHS (in the unstable region) of the s-plane. This makes it impossible to control by unity feedback.

The root locus has a branch on the right hand side of the imaginary axis, which indicates that the system is unstable in closed loop for all values of K.

From the analysis above, it is concluded that using only the closed loop gain compensation cannot check IP. REMODELING SYSTEM of the root locus is necessary, so that, for some set of gains, the system has all its roots in the left-half plane (stable region) s plane.

CHAPTER 5:

COMPENSATION DESIGN

5.1 Possible Compensation Options

There are many methods for designing compensation for the inverted pendulum system. Compensation can be designed using one of the following control analysis and design techniques. These are:

1) ROOT-LOCUS Method
2) BODE-PLOTS
3) NYQUIST Diagrams
4) NICHOLS Charts

Of these four compensation techniques, the first technique is domain and rest of all are techniques in the frequency domain. One can use it according to his convenience.

We used the root-locus techniques because they allow accurate calculation of the time domain response beyond yielding easily available information of frequency response.

5.2 Root Locus Compensation Design.

5.2.1 Why Compensation is Required?

The system analysis of the system shows that using only the gain compensation in closed loop can not control the IP. Reshaping of the root locus of the system is necessary so that, for some set of gains, the system has all its roots in the left-half plane (stable region) of S-plane.

That is, as the given system is unstable for all values of gain, so that the root locus must be reshaped so that the part of each branch falls in the left half s-plane, making the system stable. Even the specific desired performance established for the system must be achieved.

Compensation Goals

The desired TRANSIENT response for the system has following characteristics:

1. Transient (settling) time of ½ second
2. Overshoot should be < 20%, it implies that
3. The damping ratio ζ > 0.5

The desired STEADY-STATE response for the system has following characteristics:

1. Steady-state error must be zero.

Compensation Design

The compensation of the system with the introduction of poles and zeros is used to improve the operational performance. However, each additional pole compensator increases the number of roots of the equation closed-loop characteristics.

Design Specifications

We need to design a controller so that the step response of the closed loop system meets the following specifications:

1. The 5% settling time is less than 0.5 second.
2. The maximum overshoot is less than 20%.

5.3 Root Locus Design with SISO Design Tool

One of the most common techniques to meet the design criteria is a place of design root. This approach involves iterating on design to manipulate the compensator gain, poles and zeros in the root locus diagram.

The root locus diagram shows the trajectories of the poles of a closed loop feedback system as a single system parameter varies in a continuous range of values. In general, the method of root locus is used to adjust the loop gain of a (single input single output) control system SISO specifying a feedback gain closed loop poles.

The technique consists in the root locus plot closed loop trajectories poles in the complex plane as k varies. You can use this chart to identify the gain value associated with a desired set of closed-loop poles.

Adding Poles and Zeros to Compensator

You'll note that whatever the value of loop gain, ("K), a branch of the locus remains RHS (in the unstable region) s plane. This makes satisfy the design requirements with a gain only in the compensator is not possible. The root locus has a branch on the right side of the axis imaginary, which indicates that the system is unstable in closed loop for all values of K.

So poles and zeroes will be added in the compensator so as to reshape the root locus to meet our design requirements.

- Transient (settling) time of ½ second
- Overshoot should be < 20%.
- The damping ratio $\zeta > 0.5$
- Steady-state error must be zero.

PROCEDURE:

1. We introduce a pole at origin to cancel the effect of zero of the plant at the origin. The integral has cancelled the zero at origin. The root locus now becomes as

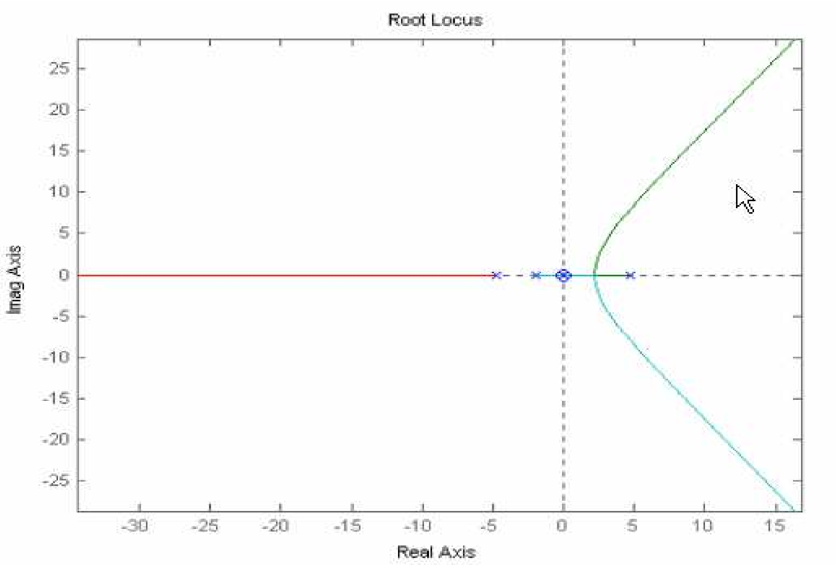

Figure 8: Root Locus after Introducing Pole

2. Now add two zeros of the controller. These zeros terminus provide over indicates the two branches of the locus on the left side of the s-plane which were, otherwise, approaches infinity. The zeros should be added in such a way that there can achieve the design requirements of settling time and the damping ratio.

3. We add two real poles at -10. The resulting root locus has the following form. Kmi=0.8. Below Kmin, roots shift in right-half and the system becomes unstable.

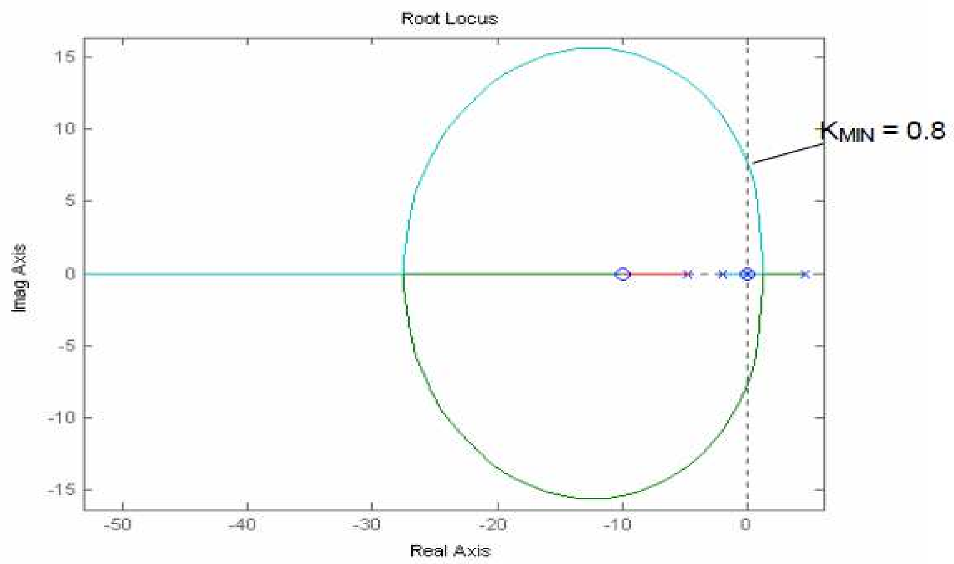

Figure 9: Root Locus after Adding 2 Real Poles

For the loop stability it is required that the loop gain should be greater than the given value (i.e., $K > K_{MIN} = 0.8$).

4. Now we set the gain such that roots of the characteristics equation correspond to z = 0.75 and $T_{S\,MAX} = 0.5$, so we select roots (selected by cross) as shown in figure. These roots correspond to value of k = 8.

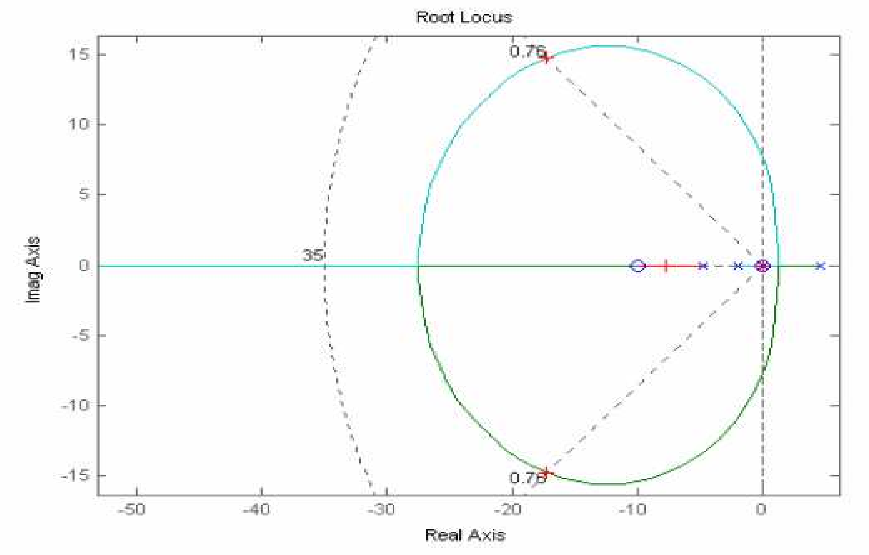

Figure 10: Root Locus after Settling Gain

The compensator equation is found to be:

$$G_c = 8 * \frac{(s^2 + 20s + 100)}{s}$$

And the general form is:

$$G_c = K_c * \frac{(K_D s^2 + K_p s + K_I)}{s}$$

Comparison gives us

$$K_C = 8; \quad K_P = 20 \quad K_I = 100 \quad K_D = 1$$

However, after simulating this PID compensated system in SIMULINK, the gain K_C was found to be best when it is 30 (minimum overshoot). So the final values of the gains are

$$K_C = 30; \quad K_P = 20 \quad K_I = 100 \quad K_D = 1$$

CHAPTER 6:

ANALYSIS OF COMPENSATED SYSTEM

6.1 POLE-ZERO MAP OF COMPENSATED OPEN LOOP SYSTEM

The pole position of the compensated model of inverted pendulum (in open-loop configuration) is shown in the following figure. In addition to the poles and zeros of the plant, we pole PID controller to cancel the zero at the origin of plant origin. We have introduced two zeros of the compensator (-10) that pull the root locus to them (on the left) and then you get a part of the root locus, which has all its roots in the stable region of the s-plane.

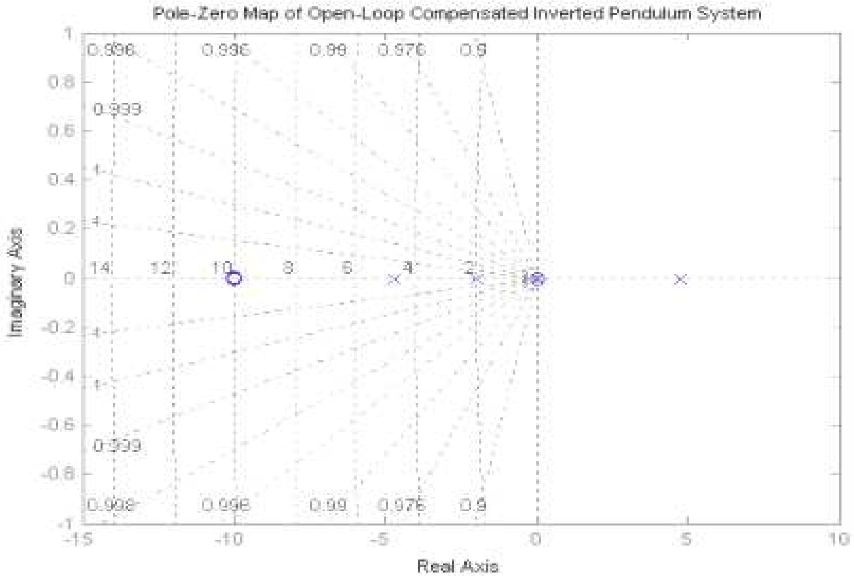

Figure 11: PZ Map of Compensated Open Loop System

6.2 ROOT LOCUS OF THE COMPENSATED SYSTEM

As can be seen in the root locus of the compensated system shown below, a PID controller can stabilize the cycle. The integral has deleted the zero origin. The two zeros of the controller they provide points terminus finished the two branches of the locus on the LHS s plane they were, otherwise, approaching infinity. However, for the stability of the loop is necessary that the loop gain should be greater than the given value (i.e K> Kmin).For our given data, *K*min = 0.8

So for **K > 0.8**; the system is stable since all the roots are on the left-hand-side of the imaginary axis.

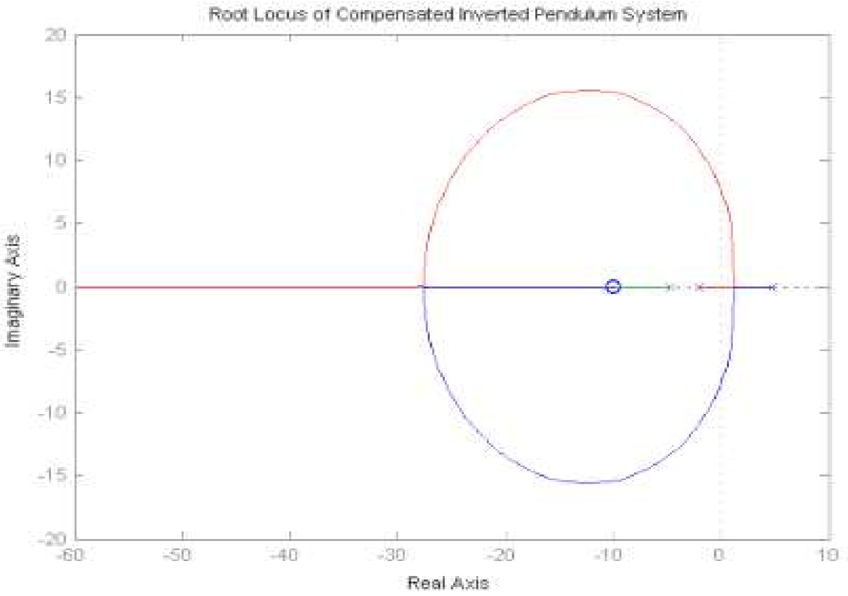

Figure 12: Root Locus of Compensated Open Loop System

6.3 POLE-ZERO MAP OF COMPENSATED CLOSED-LOOP SYSTEM

The Pole-zero location of the compensated model of inverted pendulum (in closed loop configuration) is shown in the following figure.

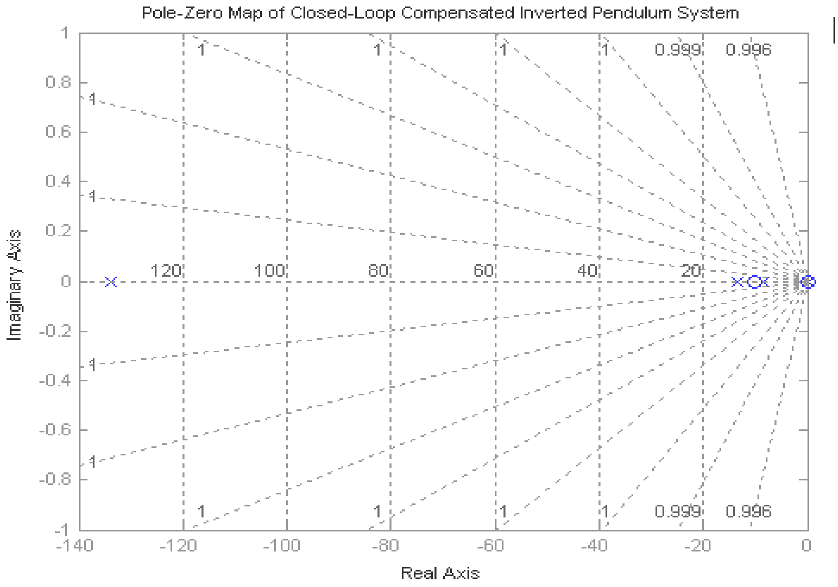

Figure 13: PZ Map of Compensated Closed Loop System

Location of poles and zeros is;

POLES:	0, -8.5359, -13.4232, -133.8242
ZEROES:	0, -10, -10

6.4 IMPULSE RESPONSE OF PID COMPENSATED SYSTEM

An impulse response of the PID compensated system is shown in the following figure. The impulse response of the compensated system is shown in Fig. The response of the system is very fast. The settling time is very small i.e. 0.028 s.

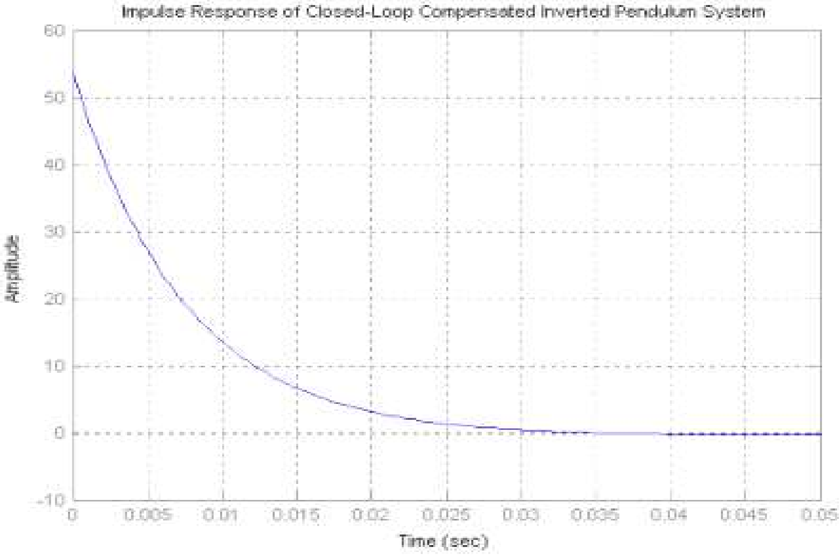

Figure 14: Impulse Response of PID Compensated System

6.5 STEP RESPONSE OF PID COMPENSATED SYSTEM

The step response of the PID compensated system is shown in the following figure. The DC Gain of the closed-loop system compensated inverted pendulum is 0.3501. The passing percentage is 8.48% (less than 20%). The steady state error is zero.

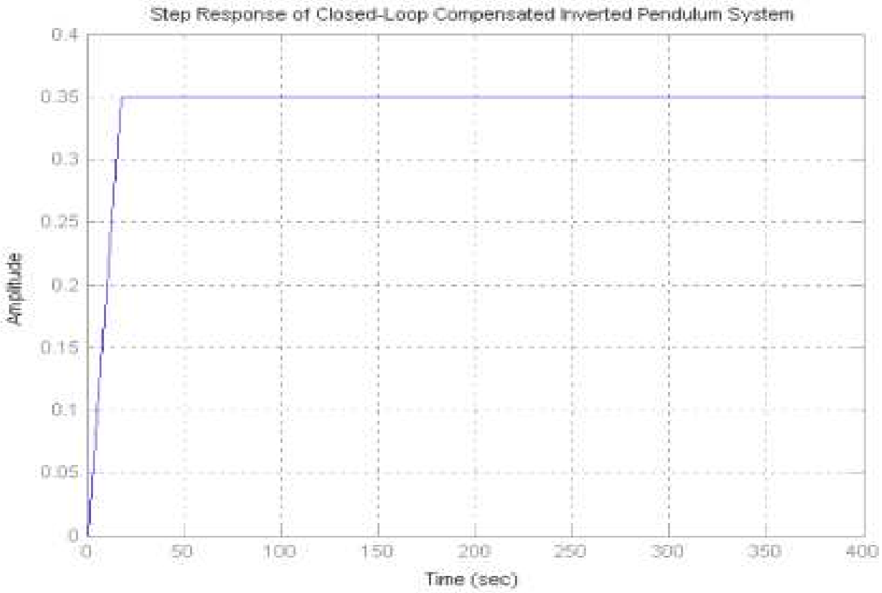

Figure 15: Step Response of PID Compensated System

6.6 CONCLUSION OF COMPENSATION ANALYSIS

Following compensation goals have been achieved.

The desired TRANSIENT response for the system has following characteristics:

Ø **Transient (settling) time** < **½ second**

0.028 SEC FOR IMPULSE RESPONSE
0.127 SEC FOR STEP RESPONSE

Ø **Overshoot** < **20%**

8.48% FOR STEP RESPONSE

Ø The **damping ratio ζ** > **0.5**

The desired STEADY-STATE response for the system has following characteristics:

Ø **Steady-state error** = **zero**.

CHAPTER 7:

PRACTICAL IMPLEMENTAION

7.1 Mechanical System

The design of a mechanical system for this project involves the integration of four main components: (1) the carriage, (2) the pendulum, (3) the runway, and (4) the mechanism used to move the carriage. There are many ways to implement them, even if each component is very dependent on the other three. These components must also satisfy some basic requirements such that it is possible to design a regulator to balance the pendulum.

- ➢ The cart motion needs to be limited to one degree of freedom which is in the horizontal plane.
- ➢ The pendulum motion needs to be limited to two degrees of freedom, one of which is the same as the cart's degree of freedom.

- ➢ The friction that impedes the cart and pendulum motion must be reduced as much as possible.

7.2 Track, Cart, and Pendulum Solutions

An option for mechanical design might have the motor as a main component of the carriage. In this drawing, depending on the size of the motor, the carriage can be built around the engine. The engine was a gear unit mounted on his tree falling in a toothed track which is flat on the surface. In order to ensure that the drive wheel does not come off the track, a guide will be necessary to implement that which is parallel to the track that the wagon would be attached too. A guide can be a steel rod along the cart that may have an arm to wrap.

- ➢ Pros
  - o Overall System is compact.
  - o Could be self contained if controller is mounted on the cart as well.
  - o Reduced Friction.
  - o Good ability to change direction quickly.
- ➢ Cons
  - o Cart design may be complex.
  - o Guide may be hard to integrate.
  - o Possible slip condition with drive wheel and track.

A variant of the second option would be to have a track is not just a cart-pendulum still mounted on the upper part. This truck is self-directed with the integrated system in the wagon. The motor may control a set of wheels that move to maintain the pendulum balanced. This design would not need a guide as the center of mass of this design would be low to the ground. A gear assembly may be required to drive the wheels simultaneously. It may be possible to have a wheel centered on a free spinning on the other side that could work satisfactory without any gear assembly.

- Pros
    - Overall System is compact.
    - Self Contained.
- Cons
    - Reaction time is dependent on motor torque
    - Cart may have more than 1 degree of freedom.
    - Pendulum may have more than 2 degrees of freedom.
    - Displacement may be hard to measure since there is no reference point to measure against.

Another interesting solution would be to have the engine with a rod mounted on the shaft so that when the motor rotates, the rod is parallel to the ground. Attached to the end of this tree would be the pendulum. Hence this system would have a circular motion, instead of the linear displacement previously discussed.

- Pros
    - Cart is simply the rod.
- Cons
    - Strong motor needed.
    - System must be well built such that the rod is strongly attached to the motor shaft.
    - Nonlinearities of the motion of the cart will be difficult to model.

Other variants of the mechanical system include using any of the solutions proposed with pendulums of various shapes, lengths, mass distributions, etc. The ideal case for the problem inverted pendulum, which is the easiest to model, is the pendulum with the whole mass located the tip. This allows the approximation of the moment of inertia of the pendulum to zero.

7.2.1 Transfer Function of the System

With reference to the figure 2, the net effect of forces on the free body diagrams is,

$$M\ddot{x}+b\dot{x}+N=F \qquad \text{Equation (1)}$$

Note that it is also possible to add up the forces in the vertical direction, but no useful information could be acquired. The sum of the forces in the vertical direction is not considered because there is no movement in this direction and it is considered that the reaction force of the earth balances all the vertical forces.

The force exerted in the horizontal direction due to the moment on the pendulum is determined as follows

$$\tau = r \times F = I.\ddot{\theta}$$

$$F = \frac{I.\ddot{\theta}}{r} = mI\ddot{\theta}$$

Component of this force in the direction of N is $mI\ddot{\theta}\cos\theta$.

The component of the centripetal force acting along the horizontal axis is as follows:

$$F = \frac{I\dot{\theta}^2}{r} = mI\dot{\theta}^2$$

Component of this force in the direction of N is $mI\dot{\theta}^2 \sin\theta$

Summing the forces in the horizontal direction, you can get an equation for N:

$$N = m\ddot{x} + mI\ddot{\theta}\cos\theta - mI\dot{\theta}^2\sin\theta$$ Equation (2)

If you substitute this equation [2] into the first equation [1], you get the first equation of motion for this system:

$$(M+m)\ddot{x} + b\dot{x} + mI\ddot{\theta}\cos\theta - mI\dot{\theta}^2\sin\theta = F$$ Equation (3)

To get the second equation of motion, sum the forces perpendicular to the pendulum. The vertical components of those forces are considered here to get the following equation.

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta$$ Equation (4)

To get rid of the P and N terms in the equation above, sum the moments around the centroid of the pendulum to get the following equation.

$$-PI\sin\theta - NI\cos\theta = I\ddot{\theta}$$ Equation (5)

Combining these last two equations, you get the second dynamic equation.

$$(I+mI)\ddot{\theta} + mgI\sin\theta = -mI\ddot{x}\cos\theta$$ Equation (6)

The set of equations that define the dynamics of the system are equations (3) and (6). These are non-linear equations and must be linearize the field of work. Since the pendulum is stabilized in a position of unstable equilibrium, which is 'PI' radiating from the position of stable equilibrium, this set of equations should be linearized about theta = Pi. Suppose that theta = Pi+ø (where ø represents a small corner of vertical upward).

After linearization the two equations of motion become (where u represents the input) and using small angle identities

$$(M+m)\ddot{x}+b\dot{x}\,mI\ddot{\varphi}=u \qquad \text{Equation (7)}$$

$$(I+mI^{2})\ddot{\varphi}-mgI\varphi=mI\ddot{x} \qquad \text{Equation (8)}$$

To obtain the transfer function of the linearized system equations analytically, we must first take the Laplace transform of the system equations. The Laplace transforms are:

$$(M+m)X(s).s^{2}+bX(s).s-mI\varphi(s).s^{2}=U(s)$$
$$(I+mI^{2})\varphi(s).s^{2}-mgI\varphi(s)=mIX(s).s^{2}$$

Since we will be looking at the angle Phi as the output of interest and the input force, re-arranging, the transfer function is:

$$\frac{\varphi(s)}{U(s)}=\frac{mI.s}{q.s^{3}+b(I+mI^{2}).s^{2}-mgl(M+m).s-bmgI}$$

where

$$q=(M+m)(I+mI^{2})-(mI)^{2}$$

7.3 Motor Selection

There are three main options to be used for engines of this project: (1) a DC motor, (2) a stepper motor, and (3) a servo motor. The two major considerations in choosing a motor are high torque and high speed. The torque required for the carriage to change direction quickly to keep the pendulum balanced. High speed is required so that the carriage can move faster the pendulum can fall.

The DC motor may have a high torque and high speed, but has a cost. First of all, when the torque and the speed of a DC motor increase, which requires more energy to start the engine. This will be limited by the circuitry used to control the motor. The control circuitry for a DC motor is typically an H-bridge that controls the direction of current through the motor according to the direction signal.

The stepper motor capable of providing a high torque, but would lack sufficient speed. A bipolar stepper would be necessary to ensure that the motor rotates both directions. For this engine, there are four input lines that must be switched in the correct order for the rotation of the motor in a certain direction and in the opposite order to move the motor in the reverse direction. This control can be done externally through the use of digital logic components that would require only a direction signal and a speed control, but this would make the design more complex.

The servomotor could provide high speed, but suffer with the pair. First of all, servo-motors in general are able to rotate 360 °. In order to have such an engine, the drive wheel connected to the motor shaft would sized such that a rotation may cause the carriage to travel the length of the track. Note also that the control of a servo motor can be bit 'difficult in this application because the level of the voltage applied to the motor indicates that the motor angle to be at. There is less insight into the design controller that operates in this way.

Regardless of which engine is chosen, a separate power supply will only be necessary for the engine. The power consumption for large motors and inductive spikes created each time the direction of the motor changes may be detrimental to any other circuit connected to the same power supply.

7.3.1 DC SERVO MOTOR MODELING

Introduction:

DC-motors that are used in feedback devices controlled are called DC-servomotors. Applications of DC servomotors abound, for example, in robotics, computer disk drives, printers, aircraft, flight control systems, machine tools. DC motors are classified as controlled DC-motor armature and field controlled DC-motors

Modeling

For the armature controlled DC motor, field current is constant and torque generated is given

$$\tau = k_t . I_a \qquad \text{Equation 1}$$

k_t is motor torques constant (Nm/Amp) and I_a is armature current. For this motor back-emf is produced in armature due to armature rotation, which is directly proportional to angular velocity of the armature.

$$\omega_m \cong \frac{d\theta}{dt}$$

$\theta(t)$ is angular position of shaft

$$V_b = k_b.\frac{d\theta_m}{dt} \qquad \text{Equation 2}$$

k_b is motor constant (V.sec/rad).

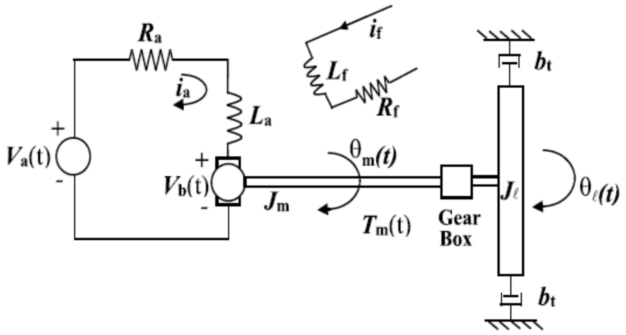

Figure 16: Armature Controlled DC Motor

Applying KVL, (Fig1), we get following equation

$$L.\frac{dI_a}{dt} + R.I_a + V_b = V_a \qquad \text{Equation 3}$$

This differential equation is governing the motion of mechanical load.

DC servo shaft is connected to gear box of the ratio and load is attached to the output of the shaft (Figure 2)

$$k_g \approx \frac{n_l}{n_m}$$

$\Rightarrow n_l$ is number of teeth on the load side

$\Rightarrow n_m$ is number of teeth on motor gear side

This can easily be shown that k_g relates to the motor shaft's angular position θ_m to gear-box-output shaft angular position i.e.

$$\Rightarrow k_g \approx \frac{\theta_m}{\theta_l}$$

And inertia at output when shaft J_l reflected at motor shaft is given by

$$\Rightarrow \frac{1}{k_g^2}.J_l$$

Therefore, application of Newton’s moment balance equation at motor output shaft yields

$$F = ma$$
$$\tau = I.\alpha$$

$$\tau = J.\frac{d^2\theta_m}{d^2t} + \frac{1}{k_g{}^2}.J_l.\frac{d^2\theta_m}{d^2t} + \frac{1}{k_g{}^2}.b.\frac{d\theta}{dt}$$

Simplifying,

$$J_{eq}.\frac{d^2\theta_l}{d^2t} + b_t.\frac{d\theta_l}{dt} = k_{g.}\tau_m \qquad \text{Equation 4}$$

Where

$$J_{eq} = k_g.J_m + J_l$$

Jeq is total load inertia reflected at motor’s shaft where b_t = rotational friction constant

Taking Laplace transform of equation (1) and equation (4)

From equation 1, $\tau = k_t.I_a$,

Laplace transforms yields into

$$T = k_t.I(s) \qquad \text{Equation a}$$

From equation (4)

$$J_{eq}\theta(s).s^2 + b_t\theta(s).s = k_g.k_t.I_a \qquad \text{Equation b}$$

To eliminate the central variables (Tm, Vb, Ia), Take Laplace transform of the equation (3)

$$L.I(s).s + R.I + V_b = V_a \qquad \text{And} \qquad I(s) = \frac{V_a - V_b}{L.s + R}$$

Substitute these into equation (b)

$$J_{eq}\theta(s).s^2 + b_t\theta(s).s = k_g.k_t.\frac{V_a - V_b}{L.s + R} \qquad \text{Equation (c)}$$

$$(L.s + R)\left[J_{eq}\theta(s).s^2 + b_t\theta(s).s\right] = k_g.k_t.(V_a - V_b)$$

$$V_b = k_b.\dot{\theta} \quad and \quad k_g = \frac{\dot{\theta}_m}{\dot{\theta}_l} \quad and \quad \dot{\theta}_m = k_g.\dot{\theta}_l$$

$$V_b = k_b.k_g.\theta(s).s$$

Substitute this result into equitation (c) and simplifying,

$$\theta(s)\left[L.J_{eq}, s^3 + s^2(b_t.L + R.J_{eq}) + s(b_t.R + k_g{}^2.k_t.k_b)\right] = k_g.k_tV_a$$

$$\frac{\theta(s)}{V_a} = \frac{k_g.k_t}{s.\left[L.J_{eq}, s^2 + s\ \ (b_t.L + R.J_{eq}) + (b_t.R + k_g{}^2.k_t.k_b)\right]} \qquad \text{Equation 5}$$

This is transfer function of the angular displacement to the applied voltage. Similarly, we can go for the angular velocity to the applied voltage i.e.

$$\frac{\omega_l(s)}{V_a(s)} = \frac{k_g.k_t}{\left[L.J_{eq}, s^2 + s.(b_t.L + R.J_{eq}) + (b_t.R + k_g{}^2.k_t.k_b)\right]} \qquad \text{Equation 6}$$

This equation has two poles, for the minimalism we can go for the following result

$$\frac{\omega_l(s)}{V_a(s)} = \frac{k}{s + p_e} + \frac{k_m}{s + p_m}$$

For simplicity, the influence of the electrical components can be neglected. Therefore, neglecting the inductance effect, gives the first order transfer function that is related to the angular velocity response to input of armor.

$$\frac{\omega_l(s)}{V_a(s)} = \frac{k_g.k_t}{R.J_{eq} + R.b_t + k_g{}^2.k_t.k_b}$$

Alternatively, this can be written as

$$\frac{\omega_l(s)}{V_a(s)} = \frac{k}{\tau.s+1}$$

K and τ are DC gain and mechanical constants of DC Servo motor.

7.4 Feedback Network

Designing a network accurate feedback is essential to stabilize the system. Thus the sensors must be relatively quiet and have a quick response so that the information retrieved from sensors accurately reflects the state of the system. In this case there are four parameters that govern the inverted pendulum system (which will be derived in a later section). They are (1) the angle, (2) the angle of speed, (3) the displacement of the carriage, and (4) the speed of the trolley. Thus there are four measurable parameters that may be used for the feedback, which determine the necessary control to stabilize the system. More conventional approaches to this problem only measure the angle and displacement and the other two parameters are derived from these.

7.4.1 Displacement Sensors

One way to detect the displacement of the trolley is to use the non-linear or linear potentiometer mounted on mounted on the track with the slider connected to the carriage. The voltage on the wiper of the potentiometer would then be converted into a digital signal by an ADC. Since this method has a simple to implement, but the output voltage may be non-linear.

Another way to detect the displacement of the trolley is to use light sensors spaced parallel to the track with a receiver on the side of the track. These work indicating when the basket is passed between a transmitter and receiver, breaking the signal.

- Pros
    - Simple Concept
- Cons
    - Accuracy is limited to the spacing of the sensors

Radar and sonar sensors are also possible. These would emit a light or sound, respectively, and wait reflection. The time required for a reflection should be used to calculate the distance. Unfortunately these sensors are more expensive and may require a large amount of processing time that slow down the reaction of the system. Thus, they will not be considered for this project.

7.4.2 Angle Sensor

One of the simplest solutions would be to mount the pendulum of a circular potentiometer. Ideally, the pot would have little friction. If practical potentiometers have a certain amount of friction, which could affect the dynamics of the pendulum fall. This has the advantage of easy implementation but may cause non-linear output voltage. Furthermore, the friction may influence the system dynamics.

A rotary encoder could be another practical solution. This sensor reads the angle and then returns the digital representation. This is available in a greater precision, but there is a compromise between the costs.

7.5 Controller Solutions

Once a mechanical system is developed with a feedback network accurate and simple interface for the control of the carriage, a controller can be designed. As this section will show, there are many ways to implement a controller working. The largest design constraint of a controller of work is how well the system is modeled. Necessary assumptions that are required to design and implement a controller working should be taken in account.

7.5.1 Controller Implementation

The PID controller, that was implemented, was Analog in nature and was implemented by Analog Circuitry.

ANALOG CONTROLLER can be realized by a number of different Op-Amp based circuits. These networks are discussed in the following section.

Besides Control-Electronics, there are Error-Detector, Servo-Amplifier, and a Feedback-Sensor (Linear Rotary Potentiometer) in the Control Loop.

7.5.2 Analog PID Controller

The PID controller works in a closed loop. A proportional controller (Kp) will have the effect of reducing the rise time and reduces but does not eliminate the steady state error. An integral control (Ki) will have the effect of eliminating the error of steady state, but can make the transient response worse. A derivative control (Kd) will have the effect of increasing the stability of the system, reducing the overshoot, and improve the transient response.

Design 1: Ideal PID Algorithm

This ideal PID algorithm has similarities with general operational amplifier. Next is a schematic representation of an operational amplifier in which input impedance is Z1 and the impedance Z2 is feedback or parallel.

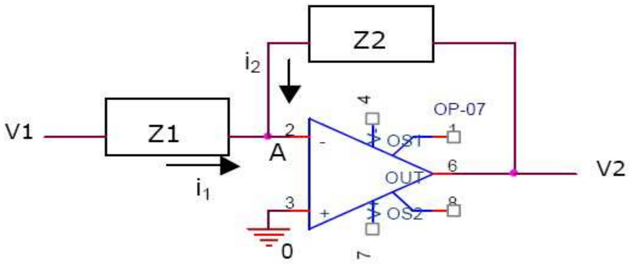

Figure 17: Ideal PID Controller Design

Overall transfer function of this amplifier is

$$\mathbf{V_2 / V_1 = -\{(R_f / R_i + C_i / C_f) + (1 / R_iC_fD) + (R_fC_i) * D\} = -\{K_p + K_i / D + K_d D\}}$$ [Appendix]

Through this equation we can have

Proportional Gain is	$\mathbf{K_p = (R_f / R_i + C_i / C_f)}$,
Integral Gain is	$\mathbf{K_i = (1 / R_iC_f)}$, and
Derivative Gain is	$\mathbf{K_d = R_fC_i}$.

Design 2: Series PID Algorithm

In this series of transfer function of PID controller PID controller is obtained as a product of PI and PD. Then follows this configuration is PI and PD cascade to get PID.

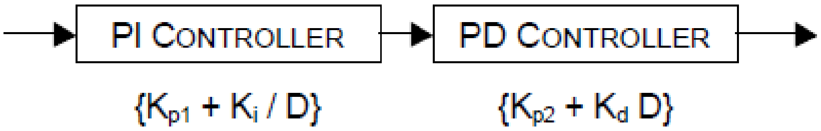

Figure 18: Block Diagram of Series PID Controller

And the schematic diagram of this design is

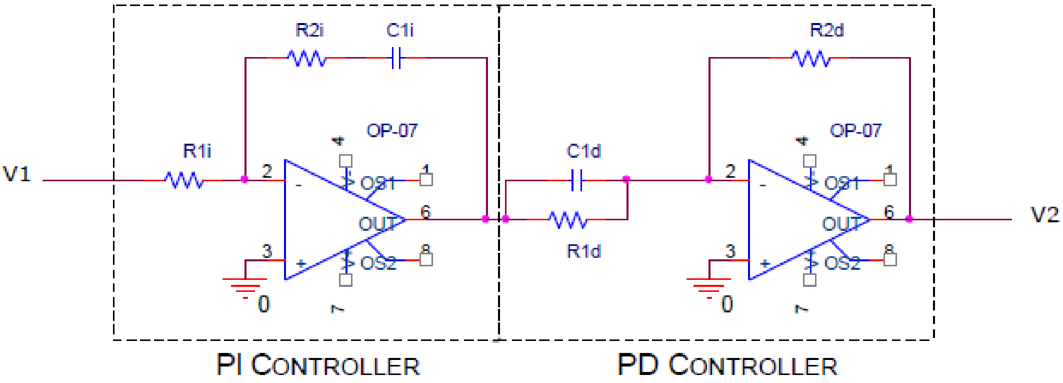

Figure 19: Schematic of Series PID Controller

Transfer function of this design is obtained as follow

$$Gc(s) = \{KP + KI / D + KD\ D\}^{\text{Appendix}}$$

And the values of the Kp, Ki, Kd are found to be

PROPORTIONAL GAIN: Kp = (R2i R2d / R1i R1d + R2d C1d / R1i C1i),
INTEGRAL GAIN: Ki = R2d / (R1d R1i C1i), and
DERIVATIVE GAIN: Kd = (R2i R2d C1d) / R1i.

DESIGN 3: PARALLEL PID ALGORITHM

This design may be regarded as the simplest approach to design an Analogue PID Circuit. This design is based on the following configuration to achieve PID action.

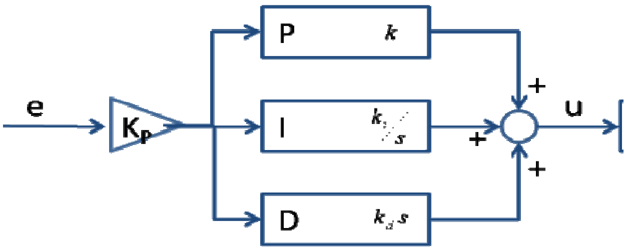

Figure 20: Block Diagram of Parallel PID Controller Design

The advantage of using this configuration is that we can have separate controls to adjust KP, KI, KD and KC (not possible in any other design). Then a PID controller Tunable can be realized using this reason, using potentiometers (preferably) or (e) variable capacitors. We have also adopted this design to make a PID CONTROLLER ANALOG TUNABLE with different gain ranges as follows:

KP = 1 TO 1001; KI = 1 TO 20; KD = 0 TO 10; KC = 0 TO 100

And the transfer function of the design is

$$G_c(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_I}{s} + K_D \quad \text{Appendix}$$

Through this transfer function we have

PROPORTIONAL GAIN is	**Kp = (R2p / R1p),**	
INTEGRAL GAIN is	**Ki= 1 / (R1i C1i),**	and
DERIVATIVE GAIN is	**Kd = (R1d C1d),**	and
CASCADED GAIN is	**Kc = (R2C / R1C).**	

7.5.3 General Tips for Designing the PIC Controller

When designing the controller for a system, follow the steps shown below to obtain desired response

1. Obtain an open loop response and determine what needs to be improved.
2. Add a proportional control to improve rise time.
3. Add a derivative control to improve the overshoot.
4. Add an integral control to eliminate the steady state error.
5. Adjust each of Kp, Kiand Kd until you obtain a desired overall response.

Lastly, please keep in mind that you don’t need to implement all three controllers into a single system, if not necessary. Keep the controller as simple as possible.

7.5.4 Tuning The PID Controller

Use the following steps to tune the PID constants:

1) Turn the KP, KI and KD potentiometers counter clockwise as far as they will turn. This sets all the constants to zero.

2) Power the device using a 12V (minimum of a 3 amp) power supply.

3) Holding the Reset button down, lift the pendulum to the vertical position and release the Reset button and pendulum at the same time.

4) The pendulum should free fall and the base will not move. This verifies that all constants are properly read as zeros.

5) Increase the KP constant by turning the potentiometer counter clockwise and repeat step 3.

6) Keep repeating steps 3-5 until there's a bit 'of oscillation in the base. If the term KP is too small, the base platform chase upper part of the pendulum Θ while continuing to increase. KP will be too large if the drive wheel is free base or oscillates at a high rate of speed.

7) Start increasing the KI same way KP until the pendulum can be balanced for a few seconds In a constant condition oscillating. When the KI is added, the base will now accelerate faster than the pendulum causing Θ to change from a positive angle in a negative angle (or vice versa). The pendulum will start to fall backward. The base should change direction and, again, accelerate faster signs of the pendulum until Θ changes and the repetitions of the cycle. This is known as the condition Overshoot.

8) Increase KD in the same manner as KP and Ki until the Overshoot condition is gone and the pendulum remains balanced.

9) Once all overshoot is gone, the PID controller is tuned.

CHAPTER 8:

FABRICATION OF SYSTEM

8.1 Introduction

Design is the creation of a plan or convention for the construction of an object or a system (as in architectural blueprints, engineering drawing, business process, circuit diagrams and sewing patterns). No generally-accepted definition of "design" exists, and the term has different connotations in different fields (see design disciplines below

8.2 Physical System Design

In collaboration with graduate studies, an evolution of the physical system shown below has been developed. Utilizing this pre-existing physical system, the design team engineered a modular inverted pendulum configuration that was to be integrated into the current design.

8.2.1 DESIGN OF PENDULUM

The pendulum bar was specified as Aluminum. This of Aluminum allows for a low-cost, light-weight, yet strong option to sustain the forces exhibited within the system. The bar was also a thin walled version which is again a lower cost due to the drawn manufacturing processes. Selecting the bar in Aluminum also allows for more adjustability with weight fluctuation

Figure 21: Pendulum Rod

8.2.1.1 SHAFT COLLAR ASSEMBLY

Shaft collars were necessary to secure the bob weight to the pendulum bar. These shaft collars have set screws spaced 90 degrees apart that secure it to the pendulum bar. The shaft collars are compact in size, adding minimal weight to the overall rotating mass. One-Piece Clamping Collars are designed to handle applications that require a greater axial load capacity than Set Screw Collars without marring the shaft.. This material presents a low cost durable option proving to be reliable for future adjustment in the physical system. The surface treatment is electrolysis nickel plating which prevents surface corrosion and increases durability.

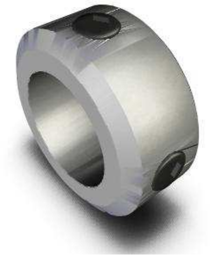

Figure 22: Shaft Collar Assembly

8.2.1.2 Bearing Mount Assembly

The bearing mount assembly consists of an aluminum housing made from Aluminum. The aluminum housing is an excellent choice to reduce overall vibration in the system. This particular design allows for ease of replacement of bearings as well as ease of assembly with its mounting tabs and holes

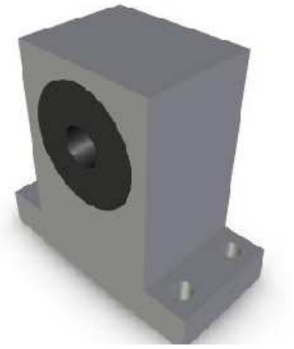

Figure 23: Bearing Mount Assembly

8.2.1.3 Rotary Shaft

The rotary shaft was specified with rigidity in mind. Shaft features stepped bearing mount surfaces with machined grooves for retaining rings. The stepped surfaces would locate the shaft when the bearing mounts were affixed to a base plate. The retaining rings hold the outer bearings in place within the bearing mount housings. This precision shaft also features electrolysis nickel plating for durability and reliability.

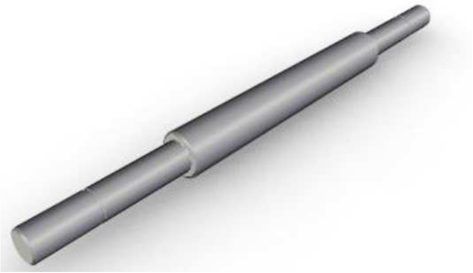

Figure 24: Rotary Shaft

8.2.1.4 Strut Clamp Assembly

To affix the pendulum bar to the rotating shaft, a T-connector type assembly was needed to secure the two components. The strut clamp assembly shown below features a “bolt together” design that locates the pendulum bar to the rotating shaft securely. This design allows for the pendulum to be perfectly perpendicular to the rotating shaft with negligible weight bias.

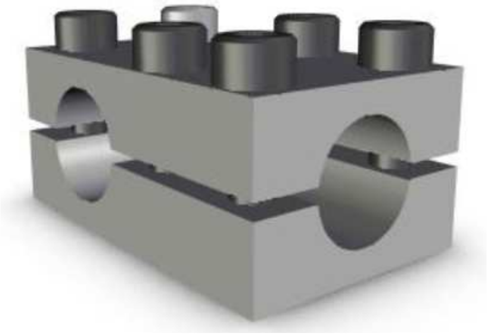

Figure 25: Strut Clamp Assembly

8.3 Fixture Design

Once the pendulum design and pivot joint design were complete, the physical system needed a platform and foundation to allow ease of application and removal from the sled, a reliable sensor mounting and junction system, and a pendulum stop system. The overall fixture design required the allowing for future additions of angle sensors as well as the sensors to couple at the ends of the rotating shaft. The continuity of modularity was also required to ease the assembly and disassembly processes.

8.3.1 Components and Materials

To allow for proper sensor installed height, work plates were needed as spacers under the bearing mounts. This raised the shaft height and solved interference issues with the sensor mounting faces and the base plate. Material was not as crucial in this application as the desired performance was small, thus the plates were specified as low carbon steel allowing for strength at minimal cost

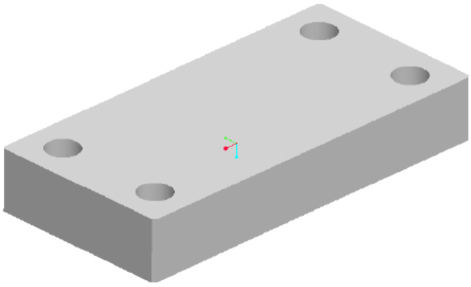

Figure 26: Work Plate

MATERIAL SELECTION

We mainly use aluminum as a material in the fabrication of our project. We consider it because it is light in weight, relatively good in strength, easy to work on and easily available in market. Also, we have used Teflon for coupling the rod of the pendulum. For damping purposes we have used Iron springs.

8.4 Electrical Hardware

8.4.1 Sensor

A digital high resolution incremental encoder was needed to sense angular position relative to pendulum movement. It was determined through simulation that resolution was very important on the stability of the system.

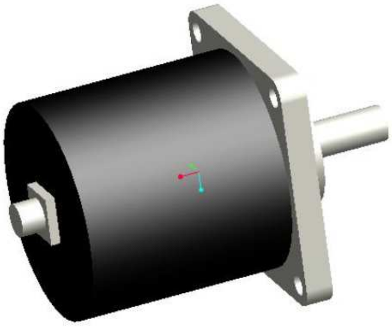

Figure 27: Angle Encoder

Transformer

A step down transformer is required to step down the input 220 AC voltages to the required voltage level.

LCD

LCD was needed for the interfacing purposes to display the angle of the pendulum read through the angle encoder.

8.5 Final Physical System Configuration

Finally, we have come across through the following design.

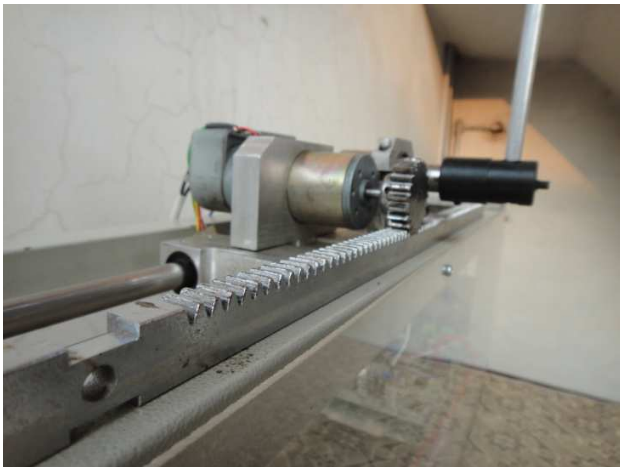

Figure 28: Final Physical System Model

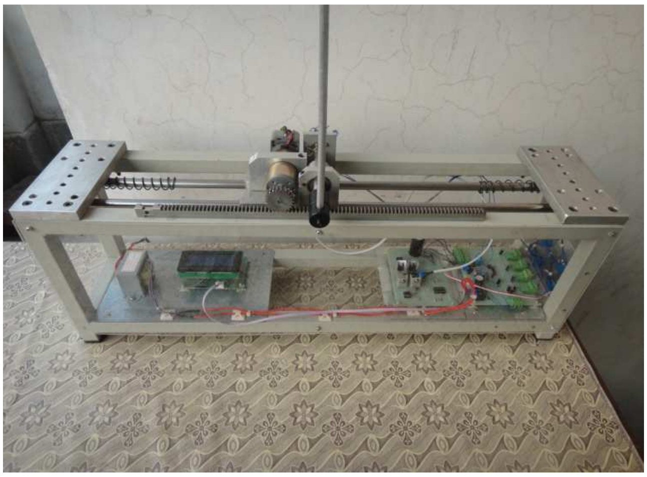

Figure 29: Final Physical System Model (i)

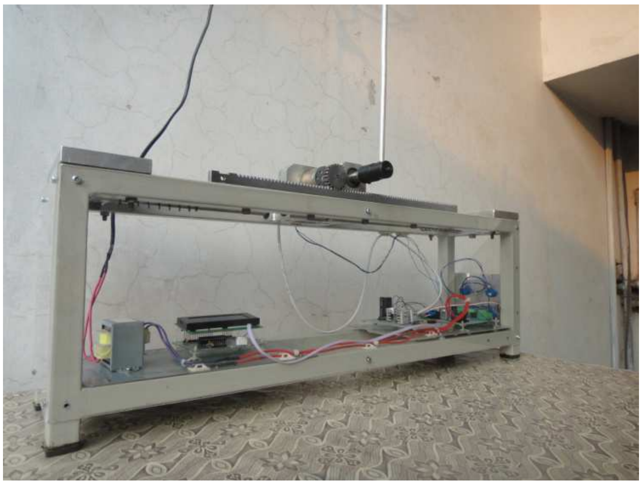

Figure 30: Final Physical System Configuration (ii)

CHAPTER 9:

ELECTRONIC CIRCUIT

9.1 Introduction

In this chapter we will be discussing the Electronic circuitry used as to stabilize the pendulum in upright position. The circuitry has been divided into these parts mainly,

1) A Microcontroller Board
2) H-Bridge
3) PID Tuning Port

We have designed each of these components separately, tested them, and then incorporated them all into the final design.

9.2 Motor Driver (Hybrid-Bridge)

An H-Bridge is an electronic power circuit that allows motor speed and direction to be controlled. Often motors are controlled from some kind of "brain" or micro controller to accomplish a mechanical goal. The micro controller provides the instructions to the motors, but it cannot provide the power required to drive the motors. An H-bridge circuit inputs the micro controller instructions and *amplifies* them to drive a mechanical motor. The H-bridge takes in the small electrical signal and translates it into high power output for the mechanical motor.

9.2.1 Direction Control (Hybrid-Bridge Topology)

Motors can rotate in two directions depending on how the battery is connected to the motor. In order to run the motor in the forward direction, connect the positive motor wire to the positive battery wire and negative to negative. However, to run the motor in reverse just switch the connections; connect the positive battery wire to the negative motor wire, and the negative battery wire to the positive motor wire. An H-Bridge circuit allows a large motor to be run in both directions with a low level logic input signal. The H-Bridge electronic structure is explicit in the name of the circuit - **H -Bridge**.

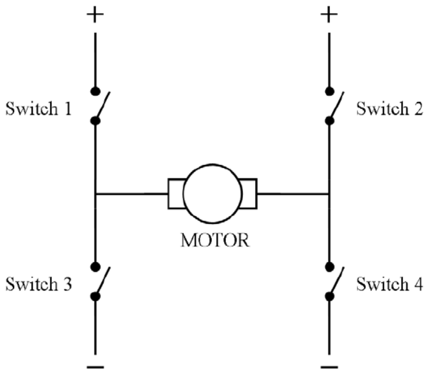

Figure 31: H-Bridge Topology

If it is desired to turn the motor on in the ***forward*** direction, switches 1 and 4 must be closed to power the motor. If it is desired to turn the motor on in the ***reverse*** direction, switches 2 and 3 must be closed to power the motor.

A detailed diagram at component level of the H-Bridge is given below,

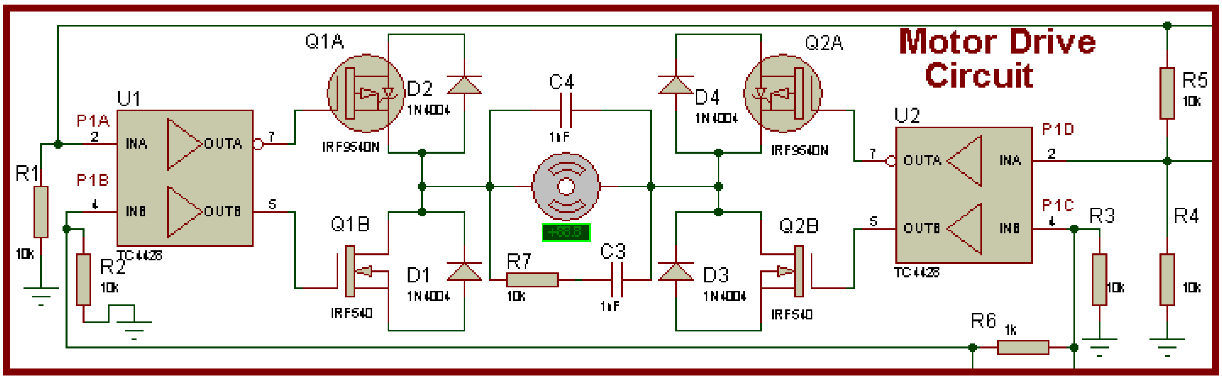

Figure 32: Schematic Level View of H-Bridge

9.2.2 Turning on the MOSFETS

This section will explain what the “switches” above actually are in terms of electronic components. The switches are power MOSFETs (transistors) that have certain properties that allow them to switch high currents based on an input signal. The MOSFETs are used in two regions of operation; Cut-off mode and Saturation mode which correspond to switch off and switched on respectively. In the H-Bridge case, to put a MOSFET into the Cut-off mode, the input signal (Gate Voltage) to the MOSFET must be grounded. However, to turn on the MOSFETs and put them into saturation mode requires a more complicated process. MOSFETS are three terminal devices with the terminals being the Gate, Drain, and Source.

The following are logic equations for each switch based on input PWM (Speed) and input DIR (Direction):

$$Switch\ 1 = PWM \bullet DIR$$

$$Switch\ 2 = PWM \bullet \overline{DIR}$$

$$Switch\ 3 = \overline{Switch\ 1} = \overline{PWM} + \overline{DIR}$$

$$Switch\ 4 = \overline{Switch\ 2} = \overline{PWM} + DIR$$

9.3 CMOS Microcontroller

The controller board has 2 main functions, to measure θ and to drive the motor. The power supply needed to run the system is dictated by the selection of the motor. The motor is controlled by an H-bridge which is driven by the PIC16F684 Enhanced Capture/Compare/ PWM Module (ECCP). The outputs of the ECCP are connected to FET drivers that produce the proper drive voltages and reduce the transition times for the FETs in the H-bridge. There are 5 potentiometers located on the controller board, 3 of which are used for adjusting the PID constants (KP, KI and KD) and one to measure Θ. The fifth potentiometer is used in conjunction with the input filter's reference. The input filter is a low-pass Bessel filter with a cut-off frequency of 60 Hz and has a voltage gain of 6. A low-pass filter is needed to eliminate any high frequency noise on the angle measurement which the derivative term of the PID controller is extremely sensitive to. The Bessel filter is used because it has the best response to a step function. (Once the pendulum is balanced, a sudden displacement that causes it to become unbalanced will look like a step function.) The cut-off frequency was chosen to be at least twice the expected frequency of the pendulum.

Follow is a flow chart of the C Code used for the system. Its task is initialization of routines i.e. checking which registers are to be read first when power is set on or MCU is set to be restart. After, it reads the constants set of Kp, Ki & Kd respectively. Applies a condition to checks the status of flags whether they're set or not. When the flag values are set this will keep an eye on the values of the Proportional, Derivative and Integral terms.

The interrupt service routine is used to control the speed of the PID loop and is set to set to run off the Timer0 Interrupt. Timer0 is an 8-bit timer that will increment the TMR0 register every instruction clock. When the TMR0 register overflows, the Timer0 Interrupt Flag is set. Interrupt speed is every 3.9 milliseconds (1/256 Hz). Since we are using the internal 8 MHz internal oscillator, we will have a 2 MHz instruction clock or 0.5 μs per instruction. This yields that the interrupt should run every 7812 instructions

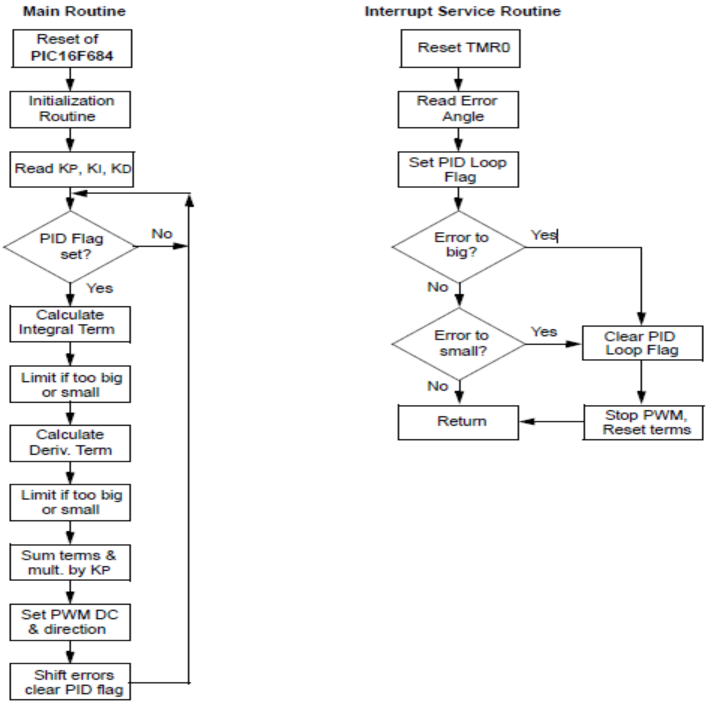

Figure 33: C CODE FLOW CHART

Here is schematic of the modern microcontroller (PIC16F684) used in the our circuit describing the pin level behavior.

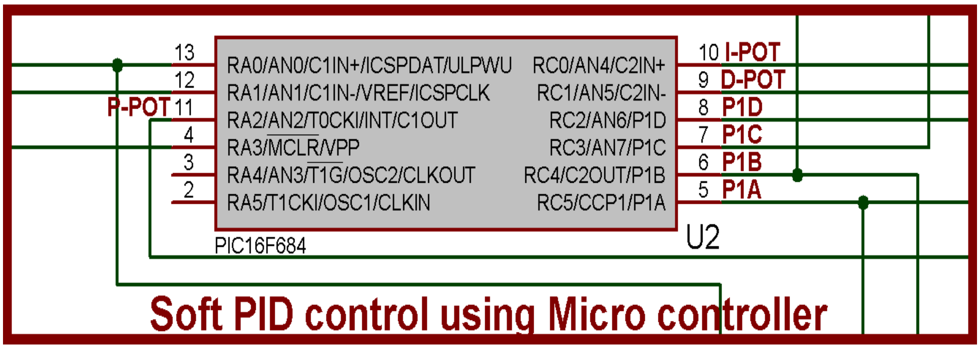

Figure 34: Pin Description of 16f684 Microcontroller

9.4 Final Schematic

A complete diagram of the circuit of the Inverted Pendulum System is shown below. This figure demonstrates the component level description of the system that how the different parts of the circuit are connected to each other. Different parts are highlighted.

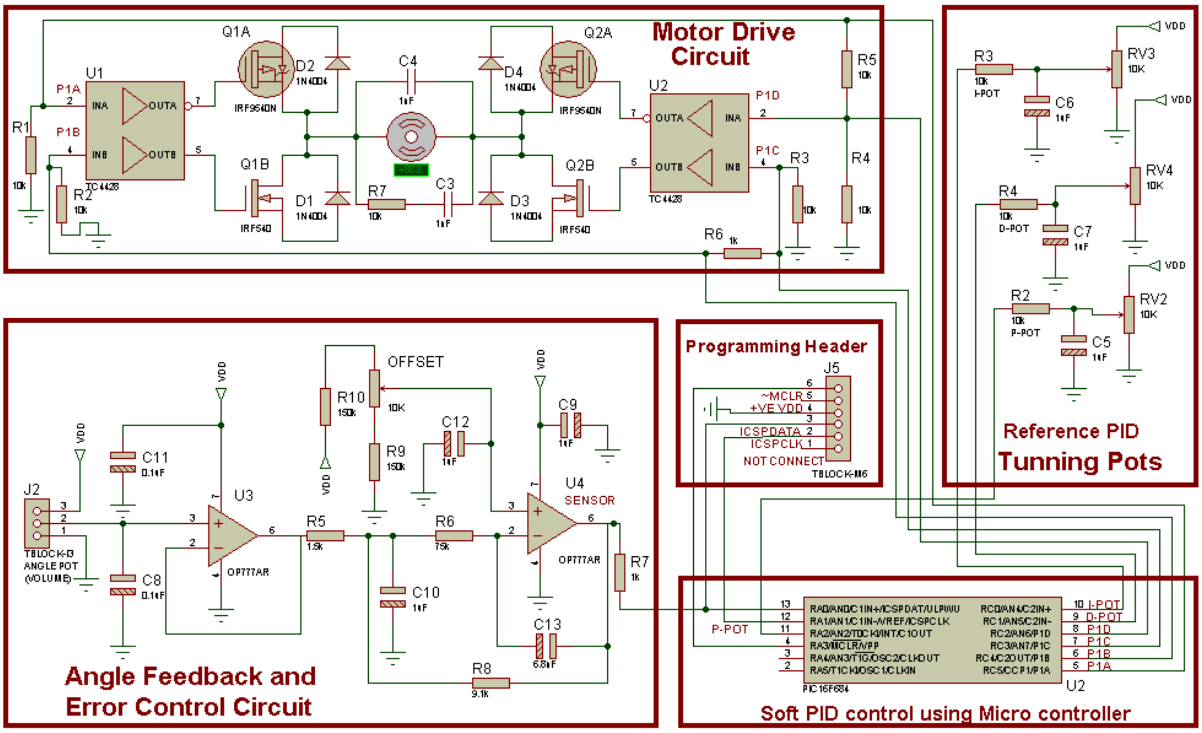

Figure 35: Detailed Schematic diagram of the Inverted Pendulum System

9.5 PCB Layout of the System

Here are the some figures shown below of the IP system at two level,

1. Rear side view of the PCB
2. Front side view of the PCB

PCB layout of motor driver circuit at soldering side is shown below,

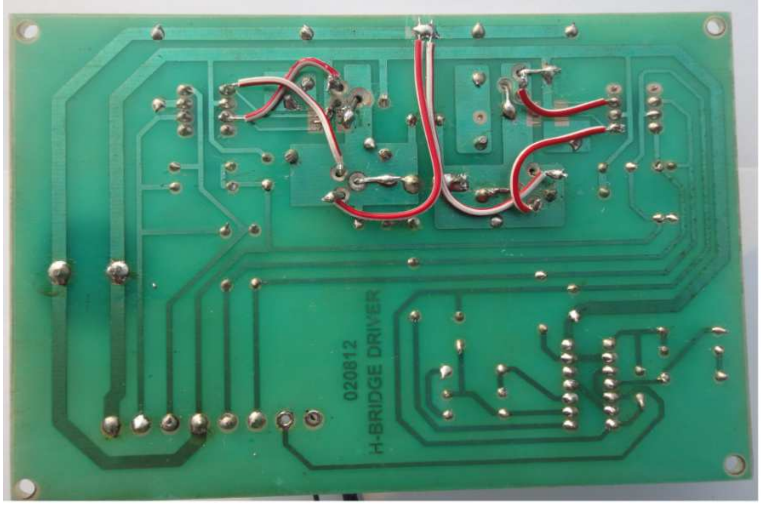

Figure 36: PCB Layout of H-Bridge (1)

And PCB layout at component level is given as,

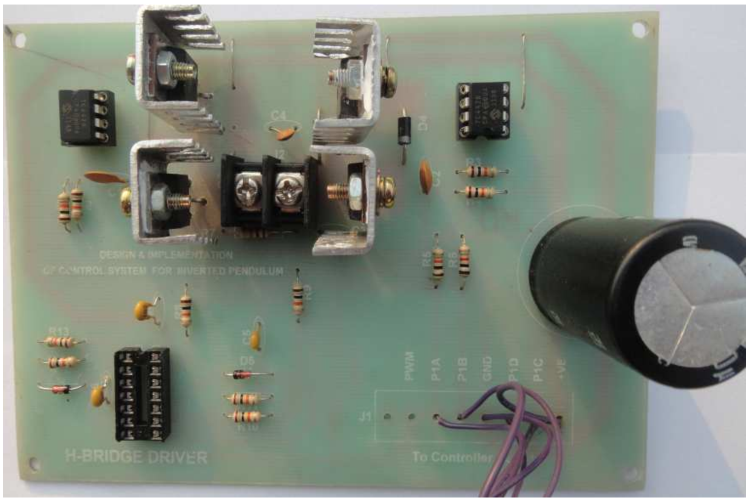

Figure 37: PCB Layout of H-Bridge (2)

This figure demonstrates the PCB layout at rear side view of MCU.

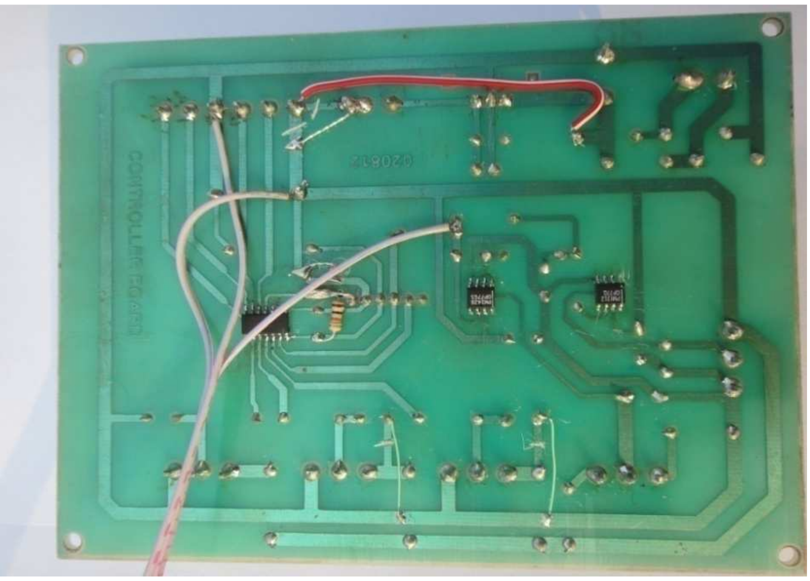

Figure 38: PCB Layout of Controller Board (1)

Here is component level view of the Microcontroller board,

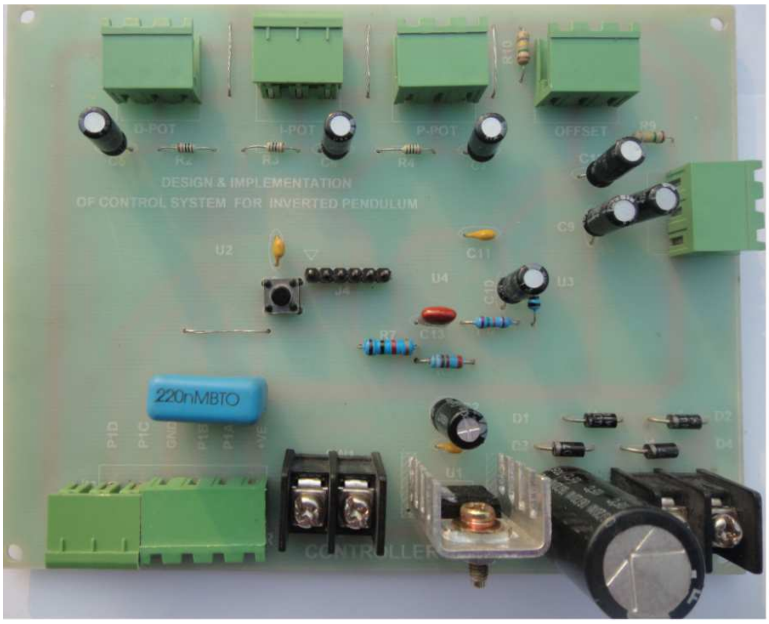

Figure 39: PCB Layout of Controller Board (2)

This diagram shows the rear side of the LCD Interfaced circuit.

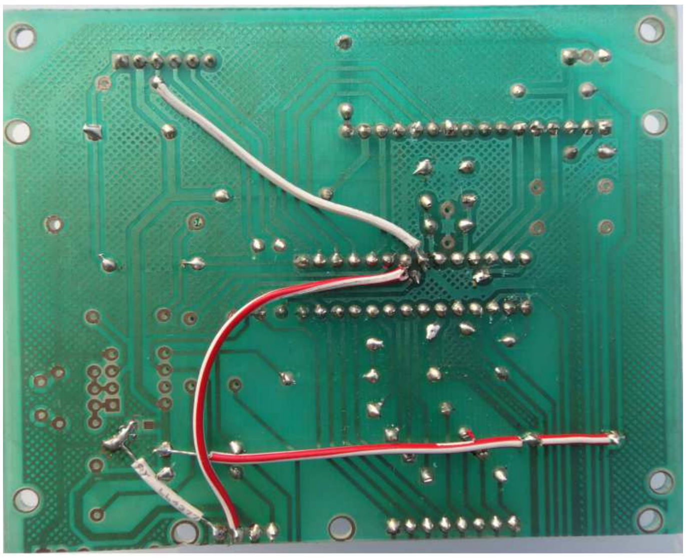

Figure 40: PCB Layout of LCD Interface Circuit (1)

This is final circuit when LCD is attached to the circuit.

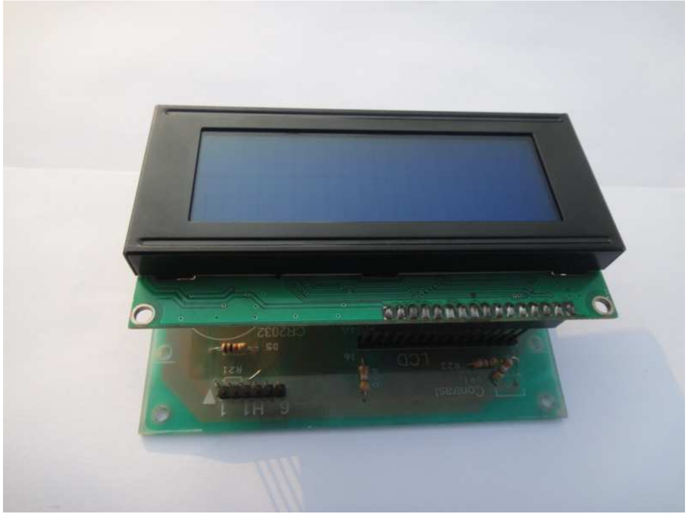

Figure 41: LCD Interface Circuit (1)

This is a final diagram when all the circuit is coupled to the Inverted Pendulum System.

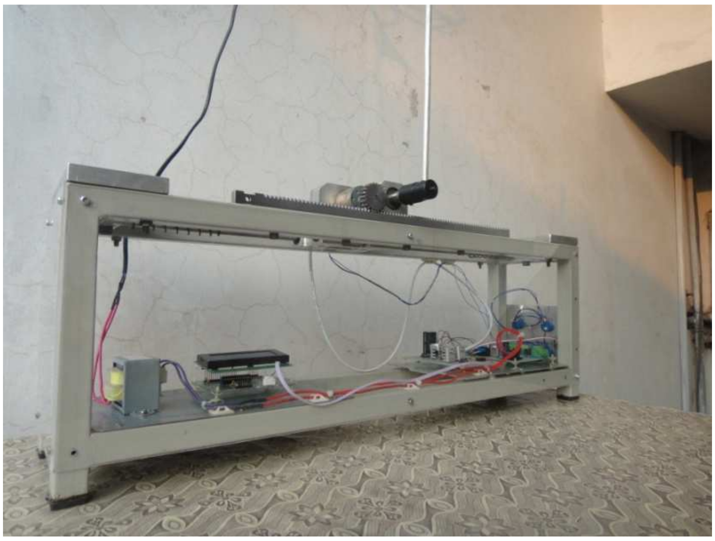

Figure 42: Inverted Pendulum System

CHAPTER 10:

CONCLUSION

Conclusion

The experience of working on the classic problem of the inverted pendulum is great. This is an ideal exercise to show his talent as an control engineer. The practical work has come a long way in helping us understand and develop a vision for the design of control systems for SISO (Single Input Output) systems.

This exercise provides a chance of designing a controller for a system that has a good dynamic behavior and hence the consideration for the transient response is emphasized. We studied a model for a pendulum system which used feedback control to keep the pendulum in the upright position. We designed a controller for this system and showed that if the delay time of sampling is not too large, then still the local controller stabilizes the system. When the time delay is large enough, stability is lost.

The foundation is laid for future research. Many requirements are met so that a system of mechanical work has been developed with a control circuit and a network of accurate feedback. An interface was also created through hardware and software to integrate the controller. And if time permits, we will now go for the non-linear control of the system, wireless sensors, pc interfacing, data acquisition and cost reduction.

The most beneficial aspect of this project was that it gave exposure to a full system design. The experience gained from developing each of the subsystems given the constraints they imposed on each other and then integrating them together proved to be invaluable.

The power of MATLAB and Simulink becomes more evident to one as all these designing would not have been possible without these tools.

ACKNOWLEDGEMENT

Alhamdullillah, we are grateful to ALLAH SWT on His blessings and mercy for making this project successful.

First of all, we would like to express deepest appreciation to our supervisor Asst. Prof. Saleem Ata, (EED, UMT) for his continuous effort, guidance and support throughout this project. Without his advice, suggestions and guidance, the project would have not been successful achieve the objectives.

Special thanks to Asst. Prof. Jameel Ahmad (EED, UMT) and Muhammad Waqas (Lect. UMT) for helping and guiding us a lot on control system theory and understanding the key concepts of the project.

Sincere thanks to Asst. Prof. Umar Farooq (Basic Physics dept. UMT) to help us understand the physics involved.

To all lecturers who have taught, thank you for the lesson that has been delivered. Thank you for useful idea, information and moral support during the course of study.

Last but not least, we would like to express our heartiest appreciation to our parents and our family, who are always there when it matters most.

APPENDIX:

TRANSFER FUNCTION AND M-FILES

Design 1: IDEAL PID CONTROLLER

Ideal PID controller can be viewed as general operational amplifier. The schematic of the amplifier is shown below in which the input voltage and impedance are V1 and Z1 respectively, and V2 is output voltage.

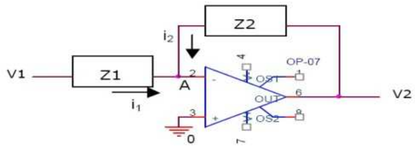

Figure 43: Schematic of Ideal PID Design

At point A, by Kirchhoff's current law,

$$i_1 + i_2 = 0 \qquad \Rightarrow i_1 = -i_2$$

$$\frac{V_i}{Z} = -\frac{V_2}{Z_2} \qquad \Rightarrow \frac{V_i}{V_2} = -\frac{Z_1}{Z_2} \qquad \text{eq(1)}$$

Here,

$$Z_1 = \frac{R_1}{R_1.j\omega C_1 + 1} \qquad \text{and} \qquad Z_2 = \frac{R_2.j\omega.C_2 + 1}{j\omega C_2}$$

Substitutes into eq(1) and simplifying we have,

V2 / V1 = – {(Rf / Ri + Ci / Cf) + (1 / RiCfD) + (RfCi) * D}

= – {Kp + Ki / D + Kd D}

Design 2: SERIES PID ALGORITHM

Here the transfer function of the controller is achieved as product of PI and PD controller. So following is the schematic of the cascade PI and PD controller to achieve the PID action.

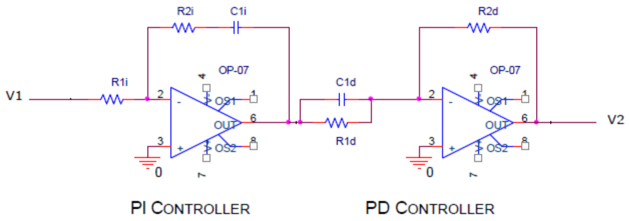

Figure 44: Schematic of Series PID Design

According to the figure, Gcomb(s) = Gi(s)*Gd(s) . . . eq(2)

According to Kirchhoff's current law,

$$i_1 + i_2 = 0 \quad \text{and} \quad \Rightarrow i_1 = -i_2$$

$$\frac{V_{i1}}{R_{I1}} = -\frac{V_C}{R_{21} + \frac{1}{C_{I1}.D}} \quad \text{and} \quad \frac{V_C}{V_{i1}} = \frac{R_{I2} + \frac{1}{C_{I1}.D}}{R_{i1}}$$

Simplifying, $$Gi(s) = \frac{R_{2i}}{R_{i1}} + \frac{1}{R_{i1}.C_{i1}.D} = K_{pi} + \frac{K_i}{D}$$

Similarly, for Gd(s) the impedances are,

$$Z_1 = \frac{R_{1d}}{R_{1d}.C_{1d}.D + 1} \quad \text{and} \quad Z_2 = R_{2d}$$

So, $i_3 = i_4$ leads to $\frac{Vo}{Vc} = -\frac{Z_2}{Z_1}$,

So the transfer function of this portion will be

$$= (R_{2d}.C_{1d}).D + \frac{R_{2d}}{R_{1d}}$$

Substituting these results into equation (2), we will have overall transfer function of the series PID action.

$$G_{COMB} = (K_{PI} + \frac{K_I}{D}) * (K_{P2} + K_D.D)$$

$$\boxed{G_{COMB} = (K_{P1}.K_{P2} + K_i) + (\frac{K_{P2}.K_i}{D}) + (K_{P1}.K_D).D}$$

Design 3: PARALLEL PID ALGORITHM

Here we will proceed with separate transfer function of proportional, integral and derivative gain.

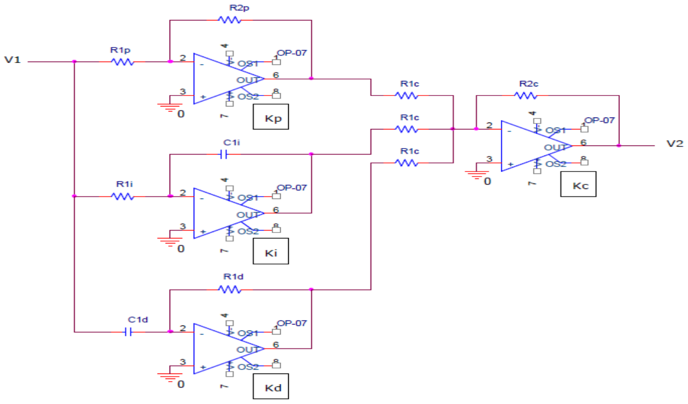

Figure 45: Schematic of Parallel PID Design

For Kp,

$i_1 + i_2 = 0$ leads to $\dfrac{V_i}{R_{ip}} = -\dfrac{V_0}{R_{2p}}$

So, $\text{Kp}=\dfrac{V_0}{V_1} = -\dfrac{R_{2p}}{R_{1p}}$

For Ki,

$i_3 + i_4 = 0$ leads to $\dfrac{V_0}{V_1} = -\dfrac{\dfrac{1}{C_{i1}.D}}{R_{1i}}$

So, $\text{Ki}=\dfrac{1}{R_{1i}.C_{i1}.D}$

For Kd,

$i_5 + i_6 = 0$ yields to $\dfrac{V_0}{V_1} = -\dfrac{R_{1D}}{1/C_{1D}.D}$

so, $\dfrac{V_0}{V_1} = -\dfrac{R_{1D}}{1/C_{1D}.D}$

$$\text{Kd} = C_{1D}.R_{1D}.D$$

Therefore, overall transfer function of system will be

$$Gc(s) = \frac{V_2}{V_1} = K_P + \frac{K_I}{s} + K_D.s$$

This equation has two pole(s) and one zero(s).

APPENDIX

M-FILES

COLLECTION OF MATLAB FILES

(1). M.FILE FOR BASIC DATA OF

INVERTED PENDULUM

```
%**********************************************
% DESIGN AND IMPLEMENTAION OF INVERTED PENDULUM
% inverted Pendulum initial data
% SST UMT, LAHORE, PAKISTAN
%**********************************************


M=0.9;              %   Mass of cart in kg
m=0.1;              %   mass of pendulum in kg
Lp=0.47;            %   length of pendulum in meter
l=0.235;            %   length of pendulum to centre of gravity
I=0.0053;           %   moment of inertiain Kg.m^2
r=0.023;            %   radius of pulley in meter
tau=0.5;            %   time const. of motor in sec
Kf=2.8648           %   feedback gain in V/rad/sec
Km=17;              %   motor gain in rad/sec/V
b=0;                %   friction of cart in N/m/sec
g=9.8;              %   acceleration due to gravity

%   Input force=u
%   Cart position=x
%   Angular displacement= theta
```

(2).M-FILE FOR OPEN LOOP &CLOSED LOOP (UNCOMPENSATED) TRANSFER FUNCTION OF IP SYSTEM

```
%*************************************************
% DESIGN AND IMPLEMENTAION OF INVERTED PENDULUM
% Transfer function of the open loop & closed loop
% (Uncompensated) Inverted Pendulum System
% SST UMT, LAHORE, PAKISTAN
%*************************************************
% variables I, Kf, Km, Lp, M, b, g, l, m, r, tau
%
load ip_data

Kp = 1 / ((M + m) * g);
K  = Kf * Kp * Km * r * (M + m);
Ap = sqrt ((M + m) * m * g * l / ((M + m)*(I + (m * (l ^ 2)))- ((m * l)^2)));

% G1(s) = Theta(s) / U(s)
% ø represents a small angle from the vertical upward direction,
% u represents the input (impulse) force on the cart by pulley chain mechanism.
num_Th_U = [0 0 Kp];
den_Th_U = [Ap^(-2) 0 -1];
Th_U = tf (num_Th_U, den_Th_U);

% G2(s) = U(s) / E(s)
% u represents the input force on the cart by the pulley chain mechanism,
% e represents the input to the motor driving pulley-chain mechanism.
num_U_E = [((Km * (M + m))*r) 0];
den_U_E = [tau 1];
U_E = tf (num_U_E, den_U_E);

disp ' '
% G(s) = Theta(s) / E(s)
% Forward Transfer Function (Open Loop Without Feedback)
num_G = conv (num_U_E, num_Th_U);
den_G = conv (den_U_E, den_Th_U);
disp 'Forward Path Transfer Function of Inverted Pendulum System is:'
G = series (U_E, Th_U)

% H(s) (Feedback)
num_H = Kf;
den_H = 1;
H = tf (num_H, den_H);

% Closed Loop Transfer Function of Uncompensated System
% Gc(s) = G(s) / (1 + G(s) * H(s))
disp 'Closed Loop Transfer Function of Inverted Pendulum System is:'
Gc = feedback (G, H)

% GH(s)
% Open Loop Transfer Function
GH = series (G, H);
```

(3).M-FILE FOR ANALYSIS OF THE UNCOMPENSATED INVERTED PENDULUM SYSTEM

```
%*********************************************************
% DESIGN AND IMPLEMENTAION OF INVERTED PENDULUM
% inverted Pendulum initial data
% Following File will show the response of Uncompansated
% open_loop and closed_loop poles and zeros of Inveted Penduum
% SST UMT, LAHORE, PAKISTAN
%**********************************************************

% variables G, GH, H, Th_U, U_E

load func_ip_uc
%
%
%Part(1)
%pole zero plot of un_cpmpensated poles
figure
pzmap (G)
title ('Pole-Zero Map of Open-Loop Uncompensated Inverted Pendulum System')
grid
%
%Part(II)
%Impulse plot of un_cpmpensated poles
figure
impulse (G)
title ('Impulse Response of Open Loop Uncompensated Inverted Pendulum System')
grid
%
%Part(III)
%Step plot of un_cpmpensated poles
figure
step (G)
title ('Step Response of Open Loop Uncompensated Inverted Pendulum System')
grid
%
%Since for the transfer_Function, we need the output block and an input
%block, so lets go for the transfer_function deffinition, tf=G/(1+GH),
% Root Locus Plot of Uncompensated System
%
%Part(IV)
figure
rlocus (GH)
title ('Root Locus of Uncompensated Inverted Pendulum System')
grid
%
%Part(V)
% Locations of Poles and Zeroes of Closed-Loop Transfer Function in Complex Plane
figure
pzmap (feedback (G, H))
title ('Unity Feedback Closed-Loop Uncompensated Inverted Pendulum System')
grid
```

(4). M-FILE FOR TRANSFEER FUNCTION OF THE COMPENSATED INVERTED PENDULUM SYSTEM

```
%*************************************************
% DESIGN AND IMPLEMENTAION OF INVERTED PENDULUM
% Transfer function of the
% compensated Inverted Pendulum System
% SST UMT, LAHORE, PAKISTAN
%*************************************************
%  variables G, GH, Gc, H, Th_U, U_E, num_G, den_G

load func_ip_uc

% C(s) = ( Kd * s^2 + Kp * s + Ki ) / s
% PID Controller to reshape the root locus.
Kp = 20;
Ki = 100;
Kd = 1;
Kc = 30;
num_PID = Kc * [Kd Kp Ki];
den_PID = [1 0];
disp(' ')
disp ('The transfer function of the PID Controller is:')
disp(' ')
PID = tf (num_PID, den_PID)

% G_comp(s) = PID(s) * G(s)
% Overall Forward Transfer Function.
num_Gcomp = conv (num_PID, num_G);
den_Gcomp = conv (den_PID, den_G);
G_comp = series (PID, G);

% Open Loop Transfer Function of the Compensated System
% G_comp_H(s) = G_comp(s) * H(s)
G_comp_H = series (G_comp, H);

% Closed-Loop Transfer Function of the Compensated System
% Gc_comp
disp 'Closed Loop Transfer Function of the Compensated Inverted Pendulum System is:'
Gc_comp = feedback (G_comp, H)
```

(5). M-FILE FOR ANALYSIS OF THE COMPENSATED INVERTED PENDULUM SYSTEM

```
%************************************************
% DESIGN AND IMPLEMENTAION OF INVERTED PENDULUM
% inverted Pendulum initial data
% SST UMT, LAHORE, PAKISTAN
%************************************************
% variables G_comp, G_comp_H, Gc_comp, PID

load func_ip_comp

%Part(1)
% Locations of Poles and Zeroes of Open-Loop Compensated Transfer Function in Complex Plane
figure
pzmap (G_comp_H)
axis ([-15 10 -1 1])
title ('Pole-Zero Map of Open-Loop Compensated Inverted Pendulum System')
grid

%Part(2)
% Impulse Response of Compensated Inverted Pendulum System
figure
impulse (Gc_comp)
title ('Impulse Response of Closed-Loop Compensated Inverted Pendulum System')
grid

%Part(3)
% Step Response of Compensated Inverted Pendulum System
figure
step (Gc_comp)
title ('Step Response of Closed-Loop Compensated Inverted Pendulum System')
grid

%Part(4)
% Root-Locus Plot of Compensated Inverted Pendulum System
figure
rlocus (G_comp_H)
sgrid (0.76,35)
title ('Root Locus of Compensated Inverted Pendulum System')
grid

%Part(5)
% Locations of Poles and Zeroes of Closed-Loop Transfer Function in Complex Plane
figure
pzmap (Gc_comp)
title ('Pole-Zero Map of Closed-Loop Compensated Inverted Pendulum System')
grid

% Part(6)
% Bode Plot
figure
bode(Gc_comp)
title ('Bode Plot of Closed-Loop Compensated Inverted Pendulum System')
grid
```

APPENDIX

Microcontroller Coding

Collection of C Codes

```
//*********************************************************************************

#include <pic.h>
#include <pic16f684.h>
#include <math.h>
#include <stdlib.h>

void Init();
void PID();
void Set_Constants();

bit flag1,do_PID,int_flag;
signed char en0, en1, en2, en3, term1_char, term2_char, off_set;
unsigned char temp;
short int temp_int;
unsigned short int ki, kd, kp;
signed int SumE_Min, SumE_Max, SumE, integral_term, derivative_term, un;
signed long Cn;

//__CONFIG   _CP_OFF & _CPD_OFF & _BOD_OFF & _MCLRE_ON & _WDT_OFF & _INTRC_OSC_NOCLKOUT & _FCMEN_ON

//***************************************************************************
// Positional PID 256 Hz
//***************************************************************************

//***************************************************************************
//Main() - Main Routine
//***************************************************************************
void main()
{
        Init();                                 //Initialize Routine
        Set_Constants();                                //Get PID coefficients ki, kp and kd
        while(1)                        //Loop Forever
        {
                if(do_PID){
                        PID();
                }
        }
}
//***************************************************************************
//Init - Initialization Routine
//***************************************************************************
void Init()
{
        PORTA = 0;
        TRISA = 0b00101101;             // Set RA4 and RA2 as outputs
        PORTC = 0;
        TRISC = 0b00000011;             // Set RC0 and RC1 as inputs, rest outputs
        CMCON0 = 0x07;                  // Disable the comparator

        IRCF0 = 1;                              // Used to set intrc speed to 8 MHz
        IRCF1 = 1;                              // Used to set intrc speed to 8 MHz
        IRCF2 = 1;                              // Used to set intrc speed to 8 MHz

        CCP1CON = 0b01001100;   // Full bridge PWM forward
        ECCPAS = 0;                                     // Auto_shutdown is disabled for now
        PR2 = 0x3F;                                     // Sets PWM Period at 31.2 kHz
        T2CON = 0;                                      // TMR2 Off with no prescale
        CCPR1L = 0;                                     // Sets Duty Cycle to zero
        TMR2ON = 1;                             // Start Timer2

        ANSEL = 0b00110101;                     // Configure AN0,AN2,AN4 and AN5 as analog
        VCFG = 0;                               // Use Vdd as Ref
        ADFM = 1;                               // Right justified A/D result
        ADCS0 = 1;                              // 16 TOSC prescale
        ADCS1 = 0;
        ADCS2 = 1;
        CHS0 = 0;                               // Channel select AN0
        CHS1 = 0;
        CHS2 = 0;
        ADON = 1;                               //Turn A/D on

        en0 = en1 = en2 = en3 = term1_char = term2_char =0;
        ki = kd = 0;
        kp = off_set = 0;
        temp_int = integral_term = derivative_term = un =0;
        SumE_Max = 30000;
        SumE_Min = 1 - SumE_Max;
        do_PID = 1;                             // Allowed to do PID function
        T0CS = 0;                               // Timer0 as timer not a counter
        TMR0 = 10;                              // Preload value
        PSA = 0;                                // Prescaler to Timer0
        PS0 = 0;                                // Prescale to 32 => 256 Hz
        PS1 = 0;
        PS2 = 1;
        INTCON = 0;
        PIE1 = 0;
        T0IE = 1;                               // Enable Timer0 int
        GIE = 1;
        return;
}


void PID()                                      // The from of the PID is C(n) = K(E(n) + (Ts/Ti)SumE + (Td/Ts)[E(n) - E(n-1)])

{
        integral_term = derivative_term = 0;

// Calculate the integral term
        SumE = SumE + en0;                                      // SumE is the summation of the error terms
        if(SumE > SumE_Max){                                    // Test if the summation is too big
                SumE = SumE_Max;
        }
        if(SumE < SumE_Min){                                    // Test if the summation is too small
                SumE = SumE_Min;
        }                                                       // Integral term is (Ts/Ti)*SumE where Ti is Kp/Ki
                                                                // and Ts is the sampling period
                                                // Actual equation used to calculate the integral term is
                                                // Ki*SumE/(Kp*Fs*X) where X is an unknown scaling factor
                                                // and Fs is the sampling frequency
        integral_term = SumE / 256;                             // Divide by the sampling frequency
        integral_term = integral_term * ki;                     // Multiply Ki
        integral_term = integral_term / 16;                     // combination of scaling factor and Kp
```

```
// Calculate the derivative term
            derivative_term = en0 - en3;
            if(derivative_term > 120){                                                                      // Test if too large
                        derivative_term = 120;
            }
            if(derivative_term < -120){                                                                     // test if too small
                        derivative_term = -120;
            }                                                                                   // Calculate derivative term using (Td/Ts)[E(n) - E(n-1)]
                                                                                                // Where Td is Kd/Kp
                                                                                                // Actual equation used is Kd(en0-en3)/(Kp*X*3*Ts)
            derivative_term = derivative_term * kd;                                             // Where X is an unknown scaling factor
            derivative_term = derivative_term >> 5;                                             // divide by 32 precalculated Kp*X*3*Ts

if(derivative_term > 120){
                        derivative_term = 120;
            }
            if(derivative_term < -120){
                        derivative_term = -120;
            }
                                                                                                // C(n) = K(E(n) + (Ts/Ti)SumE + (Td/Ts)[E(n) - E(n-1)])
            Cn = en0 + integral_term + derivative_term;                             // Sum the terms
            Cn = Cn * kp / 1024;                                                    // multiply by Kp then scale

            if(Cn >= 1000)                                                          // Used to limit duty cycle not to have punch through
            {
            Cn = 1000;
            }
  if(Cn <= -1000)
            {
                        Cn = -1000;
            }
            if(Cn == 0){                                                            // Set the speed of the PWM
                        DC1B1 = DC1B1 = 0;
                        CCPR1L = 0;
            }
            if(Cn > 0){                                                             // Motor should go forward and set the duty cycle to Cn
                        P1M1 = 0;                                                   // Motor is going forward
                        temp = Cn;
                        if(temp^0b00000001){
                                    DC1B0 = 1;
                        }
                        else{
                                    DC1B0 = 0;
                        }
                        if(temp^0b00000010){
                                    DC1B1 = 1;
                        }
                        else{
                                    DC1B1 = 0;
                        }
                        CCPR1L = Cn >> 2;                                           // Used to stop the pendulum from continually going around
                        off_set = off_set +1;                                       // the offset is use to adjust the angle of the pendulum to slightly
                        if(off_set > 55){                                           // larger than it actually is
                                    off_set = 55;
```

```
                }
        }

else {                                          // Motor should go backwards and set the duty cycle to Cn
                P1M1 = 1;                       // Motor is going backwards
                temp_int = abs(Cn);             // Returns the absolute int value of Cn
                temp = temp_int;                // int to char of LS-Byte
                if(temp^0b00000001){
                        DC1B0 = 1;
                }
                else{
                        DC1B0 = 0;
                }
                if(temp^0b00000010){
                        DC1B1 = 1;
                }
                else{
                        DC1B1 = 0;
                }
                CCPR1L = temp_int >> 2;         // Used to stop the pendulum from continually going around
                off_set = off_set -1;
                if(off_set < -55){
                        off_set = -55;
                }
        }
        en3 = en2;                      // Shift error signals
        en2 = en1;
        en1 = en0;
        en0 = 0;
        do_PID = 0;                     // Done
        RA4 = 0;                        // Test flag to measure the speed of the loop
        return;
}
void Set_Constants()
{
        ANS2 = 1;                       // Configure AN2 as analog
        ANS4 = 1;                       // Configure AN4 as analog
        ANS5 = 1;                       // Configure AN5 as analog

        ADFM = 1;                       // Right justified A/D result
        CHS0 = 0;                       // Channel select AN4
        CHS1 = 0;
        CHS2 = 1;
        temp = 200;                     // Gives delay
        while(temp){
                temp--;
        }
        GODONE = 1;
        while(GODONE);{
                temp = 0;               // Does nothing.....
        }
        ki = ADRESH << 8;               // Store the A/D result to Integral Constant
        ki = ki + ADRESL;

        CHS0 = 1;                       // Channel select AN5
        CHS1 = 0;
        CHS2 = 1;
        temp = 200;                     // Gives delay
        while(temp){
                temp--;
        }
GODONE = 1;
        while(GODONE);{
                temp = 0;               // Does nothing.....
        }
        kd = ADRESH << 8;               // Store the A/D result to Differential Constant
        kd = kd + ADRESL;

        CHS0 = 0;                       // Channel select AN2
        CHS1 = 1;
        CHS2 = 0;
        temp = 200;                     // Gives delay
        while(temp){
                temp--;
        }
        GODONE = 1;
        while(GODONE);{
                temp = 0;               // Does nothing.....
        }
        kp = ADRESH << 8;               // Store the A/D result to Proportional Constant
        kp = kp + ADRESL;
        CHS0 = 0;                       // Channel select AN0
        CHS1 = 0;
        CHS2 = 0;
}

//*****************************************************************************
```

```
void interrupt Isr()
{

        if(T0IF&&T0IE){
                TMR0 = 10;                      // Preload value
                T0IF = 0;                       // Clear Int Flag

                RA4 = 1;

                temp_int = 0;
                temp_int = ADRESH << 8;                 // Store the A/D result with offset
                temp_int = temp_int + ADRESL - 512;
                en0 = temp_int + off_set/8;             // Store to error function asuming no over-flow

                do_PID = 1;                             // Allowed to do PID function
                GODONE = 1;                             // Start next A/D cycle
        }
        else
        {
                PIR1 = 0;
                RAIF = 0;
                INTF = 0;
        }
        if(temp_int > 180){ //Check if error is too large (positive)

                DC1B0 = DC1B1 = 0;                      // Stop PWM
                CCPR1L = 0;
                en0 = en1 = en2 = en3 = term1_char = term2_char = off_set = 0;      // Clear all PID constants
                Cn = integral_term = derivative_term = SumE = RA4 = 0;
                do_PID = 0;                             // Stop doing PID
        }
        if(temp_int < -180){                            //Check if error is too large (negative)

                DC1B0 = DC1B1 = 0;                      // Stop PWM
                CCPR1L = 0;
                en0 = en1 = en2 = en3 = term1_char = term2_char = off_set = 0;              // Clear all PID constants
                Cn = integral_term = derivative_term = SumE = RA4 = 0;
                do_PID = 0;                             // Stop doing PID
        }

}
```

```
#define XTAL_FREQ               4MHZ

#include <pic.h>

#include <string.h>

#include <stdio.h>

#define    XTAL_FREQ            4MHZ

#include "delay.h"

#include "delay.c"

#include   "lcd.h"

#include   "lcd.c"


/* display on/off, cursor on/off, blinking char at cursor position */

#define LCD_DISP_OFF            0x08   /* display off                       */

#define LCD_DISP_ON             0x0C   /* display on, cursor off              */

#define LCD_DISP_ON_BLINK       0x0D   /* display on, cursor off, blink char     */

#define LCD_DISP_ON_CURSOR      0x0E   /* display on, cursor on               */

#define LCD_DISP_ON_CURSOR_BLINK 0x0F   /* display on, cursor on, blink char      */


#define testbit(var, bit) ((var) & (1 <<(bit)))

#define set_bit(ADDRESS,BIT) (ADDRESS |= (1<<BIT))

#define clear_bit(ADDRESS,BIT) (ADDRESS &= ~(1<<BIT))

#define flip_bit(ADDRESS,BIT) (ADDRESS ^= (1<<BIT))

#define test_bit(ADDRESS,BIT) (ADDRESS & (1<<BIT))

//set_bit(TRISA, 1);

//To clear a bit it woulld be clear_bit(TRISA, 3); and so on.


//#define  testbit(var, bit)  ((var) & (1 <<(bit)))

//#define  setbit(var, bit)   ((var) |= (1 << (bit)))
```

```
//#define  clrbit(var, bit)   ((var) &= ~(1 << (bit)))


#define LINE1 0        // LCD RAM position for Start of line 1

#define LINE2 0x40     // LCD RAM position for start of line 2

#define LINE3 0x14     // LCD RAM position for Start of line 1

#define LINE4 0x54     // LCD RAM position for start of line 2


unsigned const char const  angleTable[]={

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x91,0x91,0x91,0x91,0x91,0x90,

0x90,0x90,0x90,0x90,0x89,0x89,0x89,0x89,0x89,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,};

/*

unsigned const char const  angleTable[]={

0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0A,0x0B,0x0C,0x0D,0x0E,0x0F,

0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1A,0x1B,0x1C,0x1D,0x1E,0x1F,

0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x2A,0x2B,0x2C,0x2D,0x2E,0x2F,
0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3A,0x3B,0x3C,0x3D,0x3E,0x3F,
0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4A,0x4B,0x4C,0x4D,0x4E,0x4F,
0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x5A,0x5B,0x5C,0x5D,0x5E,0x5F,
0x60,0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x6A,0x6B,0x6C,0x6D,0x6E,0x6F,
0x70,0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,0x7A,0x7B,0x7C,0x7D,0x7E,0x7F,
0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0x8D,0x8E,0x8F,
0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x91,0x91,0x91,0x91,0x91,0x90,
0x90,0x90,0x90,0x90,0x91,0x91,0x91,0x91,0x91,0xA9,0xAA,0xAB,0xAC,0xAD,0xAE,0xAF,
0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,0xBE,0xBF,
0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0xCA,0xCB,0xCC,0xCD,0xCE,0xCF,
0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0xDA,0xDB,0xDC,0xDD,0xDE,0xDF,
0xE0,0xE1,0xE2,0xE3,0xE4,0xE5,0xE6,0xE7,0xE8,0xE9,0xEA,0xEB,0xEC,0xED,0xEE,0xEF,
0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0xFA,0xFB,0xFC,0xFD,0xFE,0xFF,};
*/
                //0123456789ABCDEF1234
unsigned const char *const startup1 =" INVERTED PENDULUM ";
unsigned const char *const startup2 =" SAJID,RAHEEL,SAAD ";
unsigned const char *const startup3 ="                   ";
unsigned const char *const startup4 =" SST,EED,UMT LAHORE";
unsigned const char *const blank   ="                   ";

unsigned char      Voltage;

void Initialization(void);
unsigned char ReadADC (unsigned char ADC_Channel);

__CONFIG(0x1872);

void interrupt GlobalInterrupt(void)
```

```
{           //if (RCIF )                                              // If USART RX Interrupt

            //RCIF=0;

            if          (ADIF)

            {

                        ADIF=0;

            }

            if( TMR1IF )

            {           TMR1ON = 0;

                        TMR1IF = 0;

            }

            if( INTF)

            {

                                    INTF=0;

            }

}

void main(){

  DelayMs(200);

  Initialization();

//          PORTD=0;

            lcd_init();

            DelayMs(20);

            lcd_clear();

            DelayMs(200);

            cmd_lcd(LCD_DISP_OFF);

            cmd_lcd(LCD_DISP_ON);       // Cursor off


            lcd_print(LINE1,startup1);

            lcd_print(LINE2,startup2);

            lcd_print(LINE3,startup3);
```

```
        lcd_print(LINE4,startup4);

        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);
        DelayMs(250);DelayMs(250);

        lcd_print(LINE1,blank);
        lcd_print(LINE2,blank);
        lcd_print(LINE3,blank);
        lcd_print(LINE4,blank);

while(1){
```

```
            Voltage=ReadADC(4);                                    //
   gotoxy_lcd(1,6);
            lcd_puts("ANGLE:");
            BinToHexAscii(angleTable[Voltage]);

            lcd_puts("     ");
            DelayMs(250);DelayMs(250);

}//while(1)
}//main

void Initialization(void)
{
ADCON1 = 0b00000110; //   ADC Off
//  ADCON1 = 0b00000010; //   ADC On
 TRISA=0xF0;    //--
 TRISB=0xF0;    //-- Port B bit 0 to 3 as output
 PORTC=0B11111111;    //-  bits 0 - 2 used for keyboard array bit 3 for Beep
 TRISC=0B11000000;    //-  bits 0 - 2 used for keyboard array bit 3 for Beep
 CCP1CON = 0;

//   PORTB=0;
//   TRISA = 0b110000;
//   TRISB = 0b11111111;
//   TRISC = 0b11110000;
//   ADCON1 = 0x07;
//   CMCON = 0x07;

            T1CON = 0x30;        // TIMER 1 CONTROLL REGISTER
```

```
//  bit 5-4 T1CKPS1:T1CKPS0: Timer1 Input Clock Prescale

//  Select bits  11 = 1:8 prescale value

//            Osci 4Mhz

//            4000000 / 4 = 1000000 (4 is Internal divider)

//            1000000 / 8 = 125000  ( 1:8 prescale value  )  125000 clocks in 1 Sec

//            125000  /10 = 12500    12500 clocks in 100mSec

//            125000  /100 = 12500    1250 clocks in 10mSec


            TMR1H =(65536-1250)/256;              // hibyte(timer_val);                              //

            TMR1L =(65536-1250)%256;              // lobyte(timer_val);                              //


//    INTCON REGISTER

            TMR1IE = 1; //TMR1 Overflow Interrupt Enable bit

            TMR1ON = 1; //Timer1 On bit

//   OPTION_REG=0b11000100;           //GENERATE INTERRUPT EVORY 8.192 MSEC

//   INTCON=0b00100000;

//  PIE1=0b00000000;

//  PIR1=0b00000000;

            PEIE=1;

            GIE=1;


}

unsigned char ReadADC(unsigned char ADC_Channel)

{

            ADCON1 = 0b00000010; //   ADC On

//            pic16f676 ADC setting is not same as pic16f87X

//            A new reg ANSEL

// for pic16f87X  ADCON0 = (ADC_Channel << 3) + 1;           ADGO  = 1;

// for pic16f676  ADCON0 = (ADC_Channel << 2) + 1; GODONE = 1;
```

```
        volatile unsigned int ADC_VALUE;

        /* Selecting ADC channel */

        ADCON0 = (ADC_Channel << 3) + 1;     /* Enable ADC, */



        ADFM     =0;

        ADIE       =          0;                                                      /* ADC interrupt Disable */

        ADIF       = 0;                                                     /* Resetting the ADC interupt bit */


        ADRESL   =          0;                                                      /* Resetting the ADRES value register
*/

        ADRESH   =          0;

        // delay for sample and hold capcitor charge

        asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");

        asm("nop");asm("nop");asm("nop");asm("nop");asm("nop");


 GO_DONE = 1;                                                               /* Staring the ADC process */


 while(!ADIF) continue;                                         /* Wait for conversion complete */

        ADC_VALUE          = ADRESH ;          /* Getting LSB of CCP1 */

        ADCON1 = 0b00000110; //   ADC Off

        ADIF =0;

 return (ADC_VALUE);    /* Return the value of the ADC process */

}
```

BIBLIOGRAPGHY

1. Modern Control Engineering, 5/E by Katsuhiko Ogata**.** Publisher: Prentice Hall Copyright: 2010
2. Control System Engineering, 4/E by Norman S. Nise, Wiley International Edition
3. Franklin; et al. (2005). Feedback control of dynamic systems, 5, Prentice Hall. ISBN 0-13-149930-0
4. Physics for Scientist and Engineers, Raymond A. Serway, John W. Jewett, Jr., 6thEdition, Thomson Brooks/Cole, US, 2004.
5. Engineering Mechanics - Statics, R.C. Hibbeler, 12th Edition
6. http://www.engin.umich.edu/group/ctm/examples/pend/invpen.html
7. http://en.wikipedia.org/wiki/Inverted_pendulum
8. *www.kri.com.sg/ip300.pdf*
9. http://www.ee.usyd.edu.au/tutorials_online/matlab/examples/pend/invPID.html
10. http://www.ee.usyd.edu.au/tutorials_online/matlab/examples/pend/invSS.html
11. https://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2003/es89kh98/es89kh98/index.htm
12. http://www.enotes.com/topic/Inverted_pendulum
13. http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en021807
14. http://books.google.com.pk/books?id=UhAAohoqAhAC&printsec=frontcover&dq=inverted+pendulum&source=bl&ots=crMGLdDdUb&sig=ECvQEjmF5M_BZ4FPzTBL0tSxzGI&hl=en&sa=X&ei=3bJTUOK5BoXCtAbGmoHgDA&ved=0CCkQ6AEwAA#v=onepage&q=inverted%20pendulum&f=false

Data Sheets:

1. http://tinyurl.com/8mlx3a5
2. http://tinyurl.com/8vehszh
3. http://tinyurl.com/8j684qu
4. http://tinyurl.com/9o3qsya
5. http://tinyurl.com/8zurvf4
6. http://tinyurl.com/9n3zxmq
7. http://tinyurl.com/9pqde3b
8. http://www.datasheet.co.kr/datasheet-html/I/R/F/IRF9540_HarrisCorporation.pdf.html
9. http://tinyurl.com/9vyul8y
10. http://tinyurl.com/9ems89b
11. http://tinyurl.com/8cencxw
12. http://tinyurl.com/9cdy7bg