INVERTED PENDULUM
MECHATRONICS SYSTEM PROJECT (MC691)

A PROJECT REPORT

Submitted by

Swarit Bhardwaj(1910992401)
Monish Sai R(1910992407)
Siddharth Joshi(1910992404)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

MECHATRONICS ENGINEERING

Under the supervision of

Dr. Vijay K Jadon
Dean, Academic Affairs
AE-CUIET

Mr. Gurpreet Singh
Assistant Professor
Department of Mechatronics Engineering

Chitkara University, Punjab
May 2022

CHITKARA UNIVERSITY | DEPARTMENT OF MECHATRONICS ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"INVERTED PENDULUM"** is the bonafide work of "**Swarit Bhardwaj, Monish Sai R and Siddharth Joshi"** who carried out the project work under my supervision.

Project Guide

Dr. Vijay K Jadon
Dean, Academic Affairs
AE-CUIET
Chitkara University
Punjab

Project Co Guide

Mr. Gurpreet Singh
Assistant Professor
Department of Mechatronics Engineering
CUIET, Chitkara University
Punjab

Dr. Gurdyal Singh
Assistant Dean
Department of Mechatronics Engineering
CUIET, Chitkara University
Punjab

ABSTRACT

The inverted pendulum is a classical control problem, which involves developing a system to balance a platform. To give an instance, it is similar to a man balancing on a hoverboard. The mechanical design involved building a track, cart, pendulum, and a drive mechanism mainly developed out of wood, acrylic sheets. In order to obtain a controller for the system, first the system has to be analyzed by using mathematical modeling to obtain the correct parameters corresponding to the system. The input of the mechanical subsystem is the force and the output is the angle of the pendulum and the position of the cart. The final system results in a cart that could balance a pendulum for a limited amount of time. A bibliographical survey of different design control approaches and trendy robotic problems will be presented through applications to the inverted pendulum system. In total, 150 references in the open literature, dating back to 1960, are compiled to provide an overall picture of historical, current and challenging developments based on the stabilization principle of the inverted pendulum.

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

M	Mass of Carriage
m	Mass of the Pendulum
b	Coefficient of friction for carriage
l	Length to pendulum center of mass
I	Mass moment of inertia of the pendulum
F	Force applied to the Carriage
x	Carriage position coordinate
$\dot{x}$	Carriage velocity
$\ddot{x}$	Carriage acceleration
Ɵ	Pendulum angle from vertical
$\dot{\theta}$	Pendulum angular velocity
$\ddot{\theta}$	Pendulum angular acceleration

TABLE OF CONTENTS

1. PREFACE

The inverted pendulum is a system that has a carriage connected to a motor providing linear movement only in one direction which is programmed to balance a pendulum. A normal pendulum is stable when hanging downwards, whereas an inverted pendulum is inherently unstable, and must be actively balanced in order to remain upright; this can be done either by applying a torque at the pivot point, by moving the pivot point horizontally as part of a feedback system, changing the rate of rotation of a mass mounted on the pendulum on an axis parallel to the pivot axis and thereby generating a net torque on the pendulum, or by oscillating the pivot point vertically. The pole balancing controllers are designed using LQR techniques.

1.1 Motivation

SpaceX uses an optimal control scheme for its shuttle launch and landing. The goal of this scheme is to reach zero altitude with zero vertical velocity given tight constraints of landing area and fuel. This makes it a fitting example of a non-linear optimal control problem. A simplified version of rocket shuttle landing will be made by reducing the degree of freedom of our system like balancing an inverted pendulum having 2 D.O.F. An inverted pendulum is a classic problem in dynamics and control theory and is widely used as a benchmark for testing control algorithms. As students of mechatronics engineering, we decided to build a naturally unstable system and then a controller that would stabilize it using Control Theory.

1.2 Familiarization

This problem of balancing a pendulum upside down requires insight into the movements and forces that are at play in this system. Eventually, this insight will allow us to come up with "equations of motion" of the system which can be used to compute relations between the output that is going to the actuators and the inputs coming from the sensors.

The equations of motion can be derived in two ways, in this case we used Lagrangian mechanics which is much more elegant and less tedious.

Observing the final equations of motion, we notice a relation between four quantities:

- The angle of the pendulum to the vertical
- The angular velocity of the pendulum
- The angular acceleration of the pendulum
- The linear acceleration of the cart

Where the first three are quantities that are going to be measured by the sensor and the last quantity is to be sent to the actuator to perform.

The controller used in our case is the LQR controller. In linear quadratic regulation method, we need to determine our state-feedback control gain matrix ‘k’. The MATLAB function lqr allows you to choose two parameters, ‘R’ and ‘Q’, which will balance the relative importance of the control effort and error (deviation from 0) respectively.

1.2.1 Process Flow

For determining the present system parameters two rotary encoder sensors will be used. Encoder A will monitor the pendulum while Encoder B will be used to monitor the carriage. The data by these encoders will be interpreted and processed by the Arduino Mega board. A state space model will be constructed with respect to the parameters of our system to implement the LQR control. The K-gain for the LQR control will be calculated beforehand from the python script using matlab library with respect to our mathematical model configuration and then will be declared globally as a constant in Arduino script.

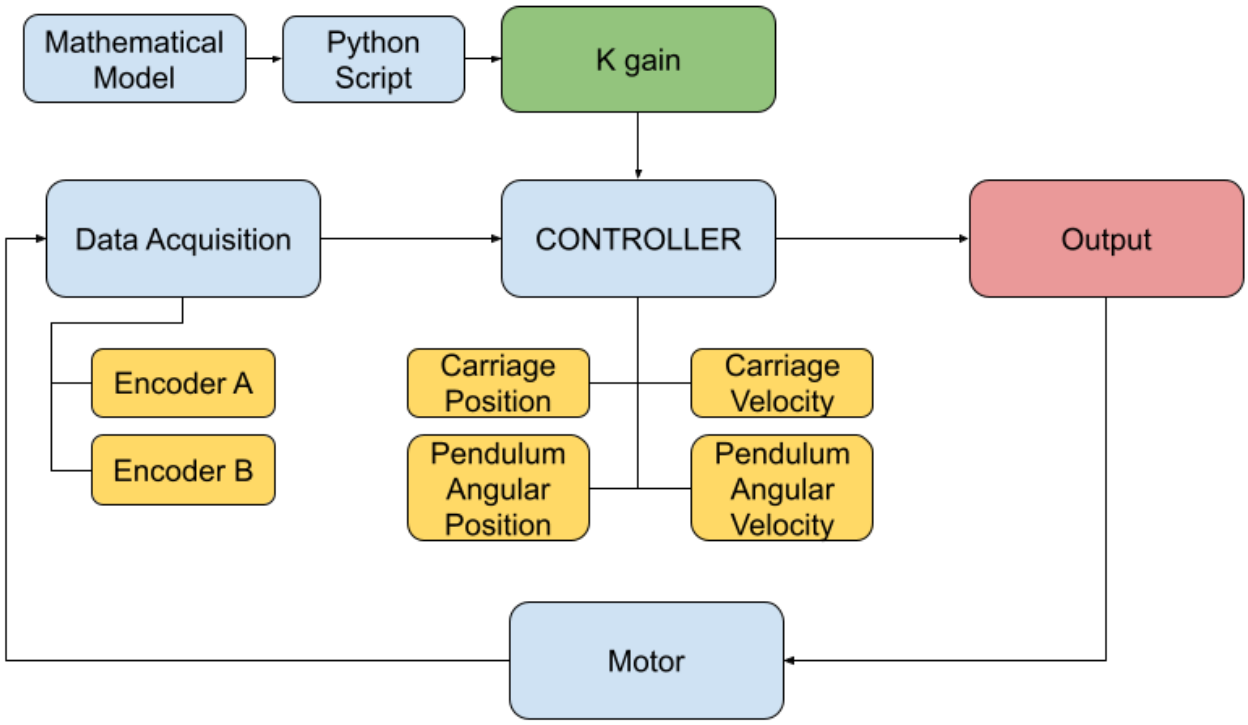


Figure 1: Process flow for working of system

1.3 Real Life Applications

There are quite a few examples in the real world of the inverted pendulum, both made by humans and in the natural world. The most prevalent example of the inverted pendulum is humans. People need to make constant adjustments to maintain the upright position of the body whether

standing, walking or running.

The inverted pendulum has been employed in various devices. For instance, it was the central component of early seismometers because any disturbance caused the pendulum to oscillate and the response was measurable.

Some two-wheeled personal transports that offer higher maneuverability are designed based on inverted pendulum models. The inverted pendulum is also related to rocket and missile guidance, where the center of gravity is located behind the center of drag causing aerodynamic instability that needs to be corrected in order to not start spinning out of control. It is also related to the balance of a biped humanoid robot, which has similar characteristics as previously mentioned with humans.

2. DESIGN PROCESS AND IMPLEMENTATION

2.1 Mechanical System

To design a robust and efficient mechanical system, it was necessary to select a “good” motor. High speed in addition to high torque is desirable for this particular application since a fast and accurate response is required. Approximating the cart would have to travel the length of rail track in one or two seconds and the torque must be enough to overcome the resistance caused by the friction, belt and the inertia of carriage and pendulum.

The carriage is kept compact and light so as to coexist with the torque of the motor. A belt drive was chosen to pull the carriage. One end of the belt is connected to a pulley mounted on the motor; the other end is placed on a pulley mounted on encoder B as mentioned in section 1.2.1.

The rail guide was chosen as the track, consisting of two rods in accordance with the compact carriage. The rail guide is fixed using two U-shaped frames on either side. The carriage will be used to mount Encoder A upon it, which will act as the fixed pivot for the pendulum.

After gathering all the components, a quick assembly was done using the basic components. Two tests were run, first using a high torque 12V motor(Faul Haber motor). This was able to move the carriage easily and smoothly, but there was not enough speed. In the second test we used a motor with comparatively higher torque 12V motor(MY6812), which resulted in swift motion of carriage.

The final assembly looked like:

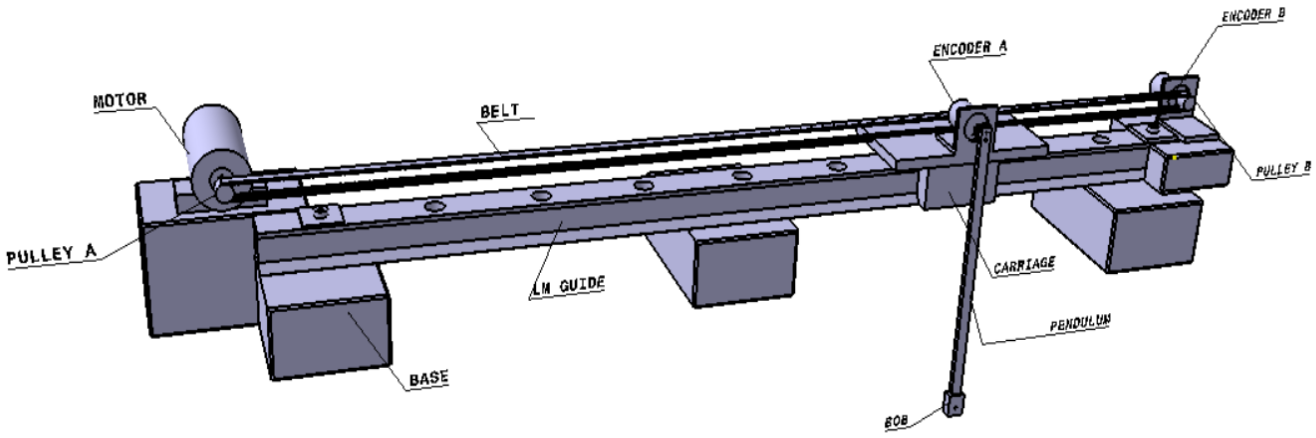
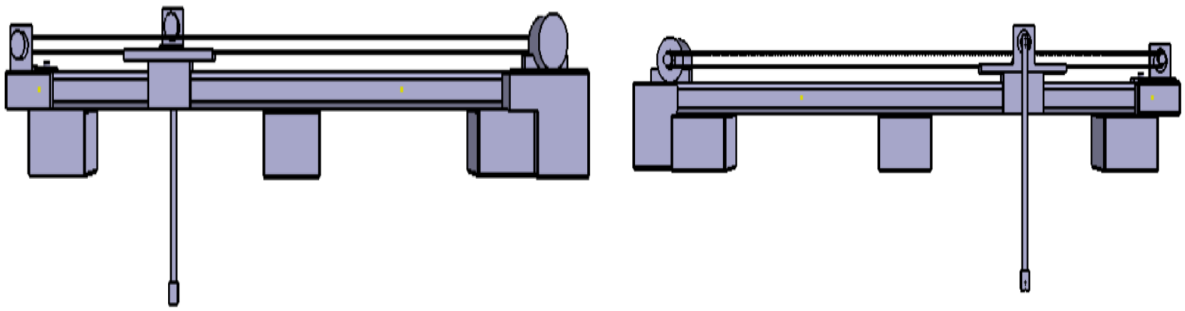
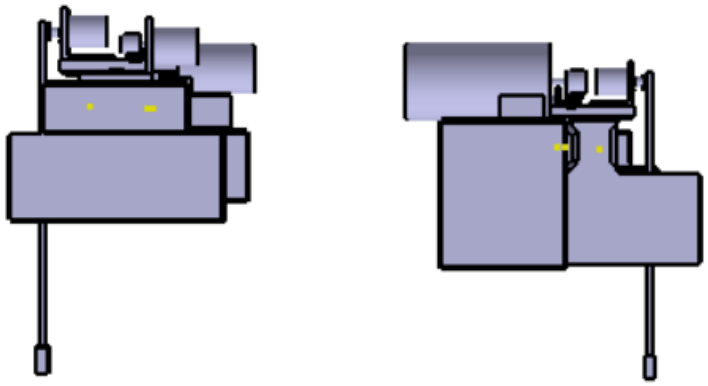


Figure 2: Labelled Perspective view of System



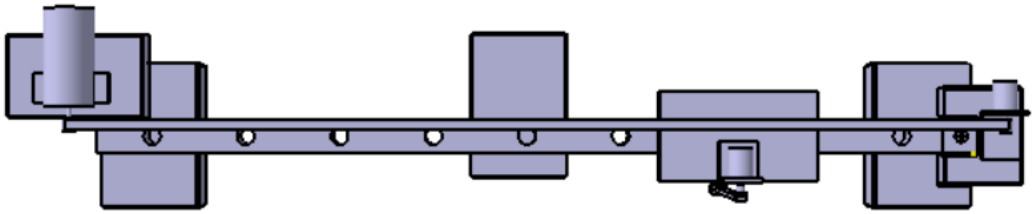
(a) (b)

Figure 3: (a)Front and (b)Back view of System

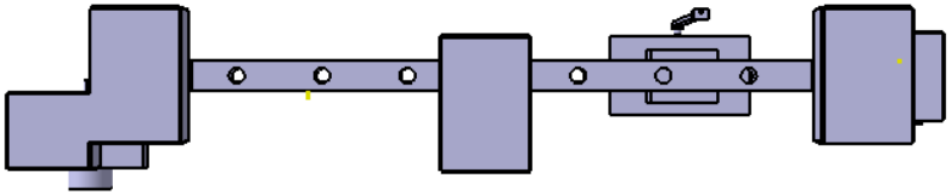


(a) (b)

Figure 4: (a)Left and (b)Right view of System



(a)



(b)

Figure 5: (a)Top and (b)Bottom view of System

Table 1: System Design Parameters

Design Parameters	**Value**
M	1.463Kg
L	0.305 m
m	0.112 Kg
b	1.000 $N/m/s$
I	0.0132 Kgm^2
Length of track	0.900 m

Table 2: Motor Parameters

Features	**Value**
Model Name	MY6812
No load RPM	3000
Rated RPM	2650
Rated Torque(Kgcm)	4
Dimensions(LxD)(mm)	120x68
Efficiency	>70%
Rated voltage(V)	12
Rated current(A)	7
Rated power(W)	100
Shaft length(mm)	30
Shaft diameter(mm)	8

2.2 Feedback Network

After the assembly, software was necessary to gather the feedback. Feedback is collected by the use of rotary quadrature encoders. Encoder A is fixed with the pendulum, while encoder B is connected to the belt drive. Both the encoders will be connected to interrupt pins on the arduino. Encoder A's values will be processed to calculate the current angle and angular velocity of the pendulum, while encoder B's values will be used to calculate the current position and velocity of the carriage.

2.3 System Model and Controller Design

Now that the mechanical subsystem is built and the feedback network is established, it is possible to design a controller. But first we need a definite model of the system. To simplify the complex mathematical equations linearization will be utilized.

2.3.1 The System Dynamics

For this system, the control input is the force F that moves the carriage horizontally and the outputs are the angular position of the pendulum θ and the horizontal position of the cart x.

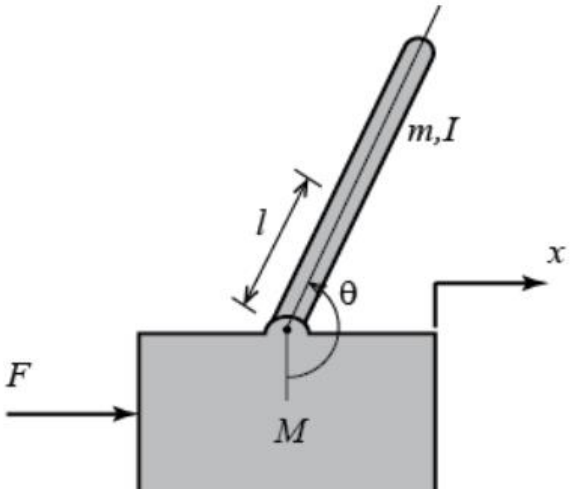


Figure 6: Simple representation of a pendulum on a carriage and factors affecting it.

For force analysis and writing system equations we will make the free body diagrams of the two elements of the inverted pendulum system.

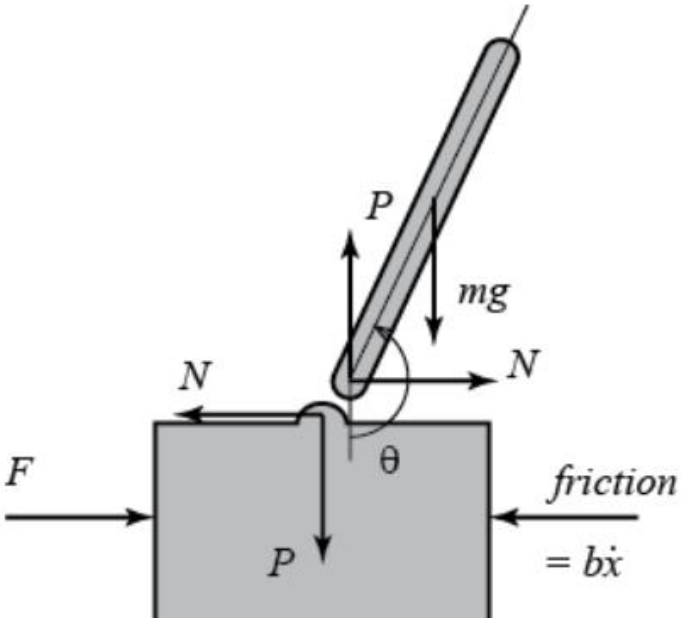


Figure 7: Free body diagram of the pendulum and carriage

Firstly analyzing the free body diagram of the carriage and solving the equations in horizontal direction.

$$Mx^{\cdot\cdot} + bx^{\cdot} + N = F$$

The forces in the vertical direction of the cart can also be summed, but no useful information would be gained.

By summing the forces in the horizontal direction of the free body diagram of the pendulum, reaction force N can be calculated.

$$N = mx^{\cdot\cdot} + ml\Theta^{\cdot\cdot}cos\Theta - ml\Theta^{\cdot 2}sin\Theta$$

Substituting this equation into the first equation, one of the governing equations for this system is achieved.

$$(M + m)x^{\cdot\cdot} + bx^{\cdot} + ml\Theta^{\cdot\cdot}cos\Theta - ml\Theta^{\cdot 2}sin\Theta = F$$

To find the second equation of motion for this system, sum the forces perpendicular to the pendulum.

$$Psin\Theta + Ncos\Theta - mgsin\Theta = ml\Theta^{\cdot\cdot} + mx^{\cdot\cdot}cos\Theta$$

P and N are unnecessary variables, and to eliminate them sum of moments about the centroid of pendulum will be used.

$$-Plsin\Theta - Nlcos\Theta = I\Theta^{\cdot\cdot}$$

Combining these last two expressions, you get the second governing equation.

$$(I + ml^2)\Theta^{\cdot\cdot} + mglsin\Theta = -mlx^{\cdot\cdot}cos\Theta$$

Since the analysis and control design techniques we will be employing in this example apply only to linear systems, this set of equations needs to be linearized. Specifically, we will linearize the equations about the vertically upward equilibrium position, $\theta = \pi$, and will assume that the system stays within a small neighborhood of this equilibrium. This assumption should be reasonably valid since under control we desire that the pendulum not deviate more than 20 degrees from the vertically upward position. Let ϕ represent the deviation of the pendulum's position from equilibrium, that is, $\theta = \pi + \phi$. Again, presuming a small deviation (ϕ) from equilibrium, we can use the

following small angle approximations of the nonlinear functions in our system equations:

$$cos\Theta = cos(\Pi + \Phi) \approx -1$$
$$sin\Theta = sin(\Pi + \Phi) \approx -\Phi$$
$$\Theta^{\cdot 2} = \Phi^{\cdot 2} \approx 0$$

After substituting the above approximations into our nonlinear governing equations, we arrive at the two linearized equations of motion. Note *u* has been substituted for the input F.

$$(I + ml^2)\Phi^{\cdot\cdot} - mgl\Phi = mlx^{\cdot\cdot}$$
$$(M + m)x^{\cdot\cdot} + bx^{\cdot} - ml\Phi^{\cdot\cdot} = u$$

2.3.2 The State Space Model

The linearized equations of motion from above can also be represented in state-space form if they are rearranged into a series of first order differential equations. Since the equations are linear, they can then be put into the standard matrix form shown below.

$$\begin{bmatrix} x^{\cdot} \\ x^{\cdot\cdot} \\ \Phi^{\cdot} \\ \Phi^{\cdot\cdot} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ x^{\cdot} \\ \Phi \\ \Phi^{\cdot} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ x^{\cdot} \\ \Phi \\ \Phi^{\cdot} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

2.3.3 LQR Controller

Linear quadratic regulator (LQR) is a control method, which is generally used for control of linear dynamical systems, and is used to control the nonlinear dynamical system. LQR is one of the optimal control techniques, which takes into account the states of the dynamical system and control input to make the optimal control decisions. The nonlinear system states are fed to LQR which is designed using a linear state-space model. This is simple as well as robust.

2.3.3.1 LQR: A Deep Dive

The inverted pendulum, a highly nonlinear unstable system, is used as a benchmark for implementing the control methods. Here the control objective is to control the system such that the carriage reaches a desired position and the inverted pendulum stabilizes in the upright position. The LQR algorithm reduces the amount of work done by the control systems engineer to optimize the controller. However, the engineer still needs to specify the cost function parameters, and compare the results with the specified design goals. Often this means that controller construction will be an iterative process in which the engineer judges the "optimal" controllers produced through simulation and then adjusts the parameters to produce a controller more consistent with design goals.

2.3.3.2 Optimal Control using LQR

Linear quadratic regulator (LQR) is one of the optimal control techniques, which takes into account the states of the dynamical system and control input to make the optimal control decisions. This is simple as well as robust.

After linearization of nonlinear system equations about the upright (unstable) equilibrium position having initial conditions as $\boldsymbol{x}_0 = [0, 0, 0, 0]^T$.

2.4 Circuit Design

The design process involves moving from the specification at the start, to a plan that contains all the information needed to be physically constructed at the end, this normally happens by passing through a number of stages, although in a very simple circuit it may be done in a single step. The process normally begins with the conversion of the specification into a block diagram of the various functions that the circuit must perform, at this stage the contents of each block are not considered, only what each block must do, this is sometimes referred to as a black box design.

2.4.1Arduino pin configuration

EncoderA to arduino connections:

- Encoder pin a(Blue) to pin 20
- Encoder pin b(Green) to pin 21
- Black to Gnd
- VCC(Red)to 5V

EncoderB to arduino:

- Encoder pin a(Blue) to pin 19
- Encoder pin b(Green) to pin 18
- Black to Gnd
- VCC(Red)to 5V

Motor driver to arduino:

- INT3(green) to pin 5

- INT4 (orange) to pin 6

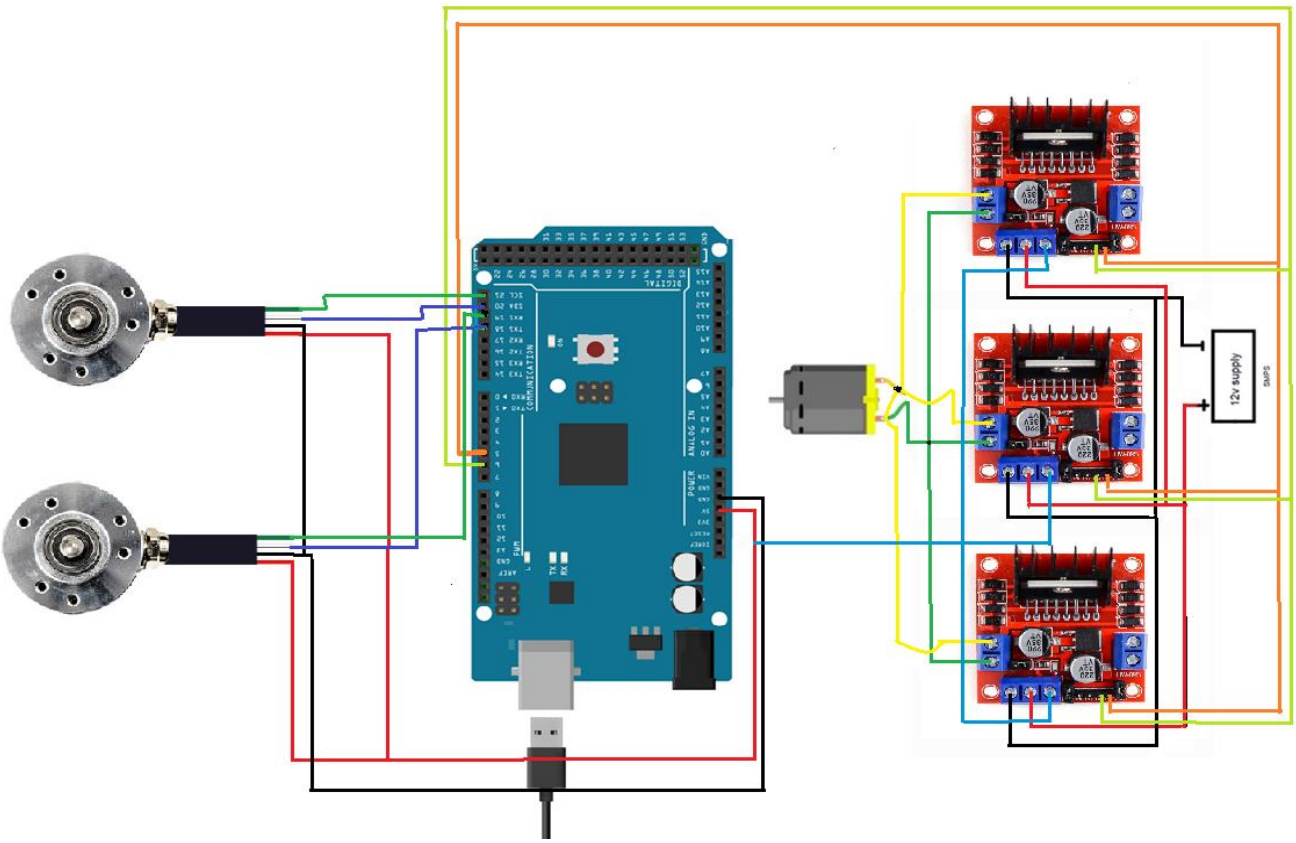


Figure 8: Circuit diagram

2.4.2 Components

The individual circuit components are chosen to carry out each function in the overall design, at this stage the physical layout and electrical connections of each component are also decided, this layout commonly taking the form of artwork for the production of a printed circuit board or Integrated circuit.

1. Arduino mega - The brain of the system, senses the current situation of the system and changes the output accordingly.

2. Encoder(2nos.) - These convert motion to an electrical signal that can be read by some type of controller(arduino). Sends a feedback signal that can be used to determine position, count, or direction. (rotary quadrature encoder 600 ppr/2400cpr)
3. Motor - The only and primary actuator of the system.
4. Motor driver L293d - As the motor is not Capable of directly connecting with the arduino it requires a high amount of current and voltage, so the motor driver is used as an interface.
5. LM Guide and LM Block – Guide provides constraints the mechanism in linear motion.
6. SMPS 12V - The power supply for system to work.

2.4.3 Cost of Components

Generally, the cost of designing circuits is directly tied to the final circuit's complexity. The greater the complexity (quantity of components and novelty of design), the more hours of a skilled engineer's time will be necessary to create a functional product. The process can be tedious, as minute details or features could take any amount of time, materials and manpower to create.

Considering only the electronic components the cost is mentioned below:

Table 3: Cost of Components

Component	Quantity	Cost
Arduino mega	1	₹975
Encoder	2	₹2855.60
12V Motor	1	₹1260
Motor driver	3	₹318
LM Guide	1	₹5100
Jumper wires	1	₹20
SMPS 12V	1	₹495

2.5 Fusion of Controller with Arduino

In this section we will overview the working of the coding scripts required for functioning of the inverted pendulum system and visit their code.

2.5.1 Flowchart for python

Figure 9: Flowchart representing crux of python script.

2.5.2 Python code

```
import numpy as np

from control.matlab import *

from control import place

from scipy import integrate



m = 0.067  # mass of pendulum in kg
M = 1.463  # mass of carriage in kg
l = 0.305   # COM of pendulum in m
g = 9.81    # gravity
b = 0.7      # coeff of friction
I = m*(l**2)#0.0113 #0.0104 # moment of inertia of pendulum

A = np.array([[0,1, 0, 0],
            [0, -b*(I+m*l*2)/((M+m)*I+M*m*l**2), (m**2)*(l**2)*g/(I+m*l**2), 0],
            [0,0,0,1],
            [0,-b*m*l/((M*m)*I+M*m*l**2),(m+M)*g*m*l/((M*m)*I+m*l**2),0]], dtype=float)

B = np.array([[0],[(I+m*l**2)/((M+m)*I+M*m*l**2)],
            [0],[m*l/((M*m)*I+m*l**2)]], dtype=float)

Q = np.array([[1, 0, 0  ,0   ],
            [0, 1, 0  ,0   ],
            [0, 0, 100,0   ],
            [0, 0, 0  ,10000]], dtype=int)

# print(A,B)
R = 0.0004

K = lqr(A,B,Q,R)[0]
k = np.array(K)
for i in k:
  x = np.round([*i],3)
  print(*x, sep=", ")
```

2.5.3 Flowchart for Arduino

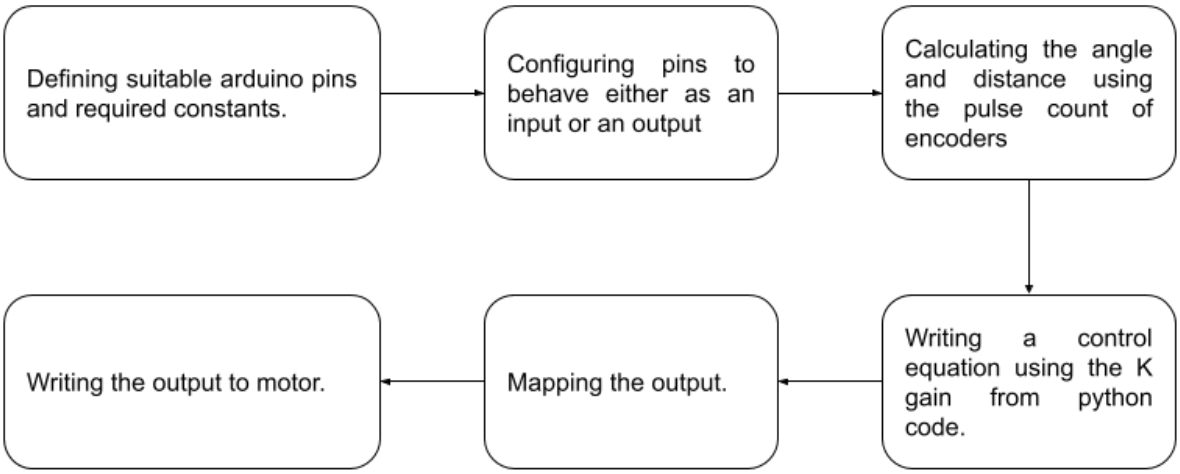


Figure 10: Flowchart representing working of arduino script.

2.5.4 Arduino code

```
/* PURPOSE: to detect rpm by counting pulses from encoder */
//#define motion_flag 0
bool dir = 1;    // 1 for RIGHT
int pwm = 0;
bool motion = 0;

// DEFINING ARDUINO PINS
int pendulum_encoder_pin_A = 21;
int pendulum_encoder_pin_B = 19;

int carriage_encoder_pin_A = 20;
int carriage_encoder_pin_B = 18;

int motorA = 5;   // IF HIGH => LEFT MOTION
int motorB = 6;   // IF HIGH => RIGHT MOTION


// DEFINING REQUIRED CONSTANTS
int currentTime = 0;
int previousTime = 0;
int duration = 80; //in ms

double dist = 0.0;
double angle = 0.0;
```

```
double dist_prev = 0.0;
double angle_prev = 0.0;
double dist_dot = 0.0;
double angle_dot = 0.0;

float pi = 3.14;
float wheel_radius = 13.4;  //mm
volatile int pulse_count_carriage = 0 ;
volatile int pulse_count_pendulum = 0 ;
int total_pulses = 0;
int pulses = 0;

// DEFINING CONTROLLER CONSTANTS
float k[4] = { -50.0, -27.51, 1752.28, 5002.525 };
float nbar = -50.451;
float desired_para[4][1] = {{0.0},{0.0},{0.0},{0.0}};
float current_para[4][1] = {{0.0}, {0.0}, {0.0}, {0.0}};
float error[4][1] = {{0.0}, {0.0}, {0.0}, {0.0}};
float output = 0.0;

void setup() {
  // put your setup code here, to run once:
  pinMode(motorA, OUTPUT);
  pinMode(motorB, OUTPUT);
  pinMode(carriage_encoder_pin_B, INPUT_PULLUP);
  pinMode(pendulum_encoder_pin_B, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(pendulum_encoder_pin_A),  StartInterrupt_pendulum,  RISING);  //
when it gets a signal at rising edge it will call StartInterruptA() function
  attachInterrupt(digitalPinToInterrupt(carriage_encoder_pin_A), StartInterrupt_carriage, RISING); // when
it gets a signal at rising edge it will call StartInterruptA() function
//  Serial.begin(9600);
//  delay(20);
}

void loop() {
  if (dist<320.0 and dist>-320.0){
    if (dir==1){
    analogWrite(motorB, pwm);
    digitalWrite(motorA,LOW);
    }
    else{
      analogWrite(motorA, pwm);
      digitalWrite(motorB,LOW);
      }
  }
  else{ digitalWrite(motorA,LOW); digitalWrite(motorB,LOW); }
  currentTime = millis(); //initializing timer
  // if difference in times is greater than given duration, calculate the rpm using the pulse count and then set it
to zero
  if (currentTime - previousTime > duration){

    detachInterrupt(pendulum_encoder_pin_A);
```

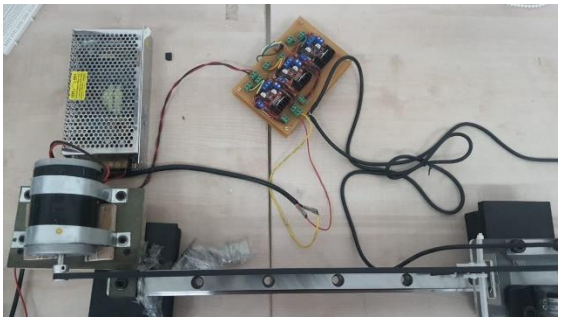
```
    detachInterrupt(carriage_encoder_pin_A);

    previousTime = currentTime;

    dist += double(pulse_count_carriage)*pi*wheel_radius/600.0;
    angle += (double(pulse_count_pendulum)*2.0*pi)/600.0;

//    Serial.println("pulse = " + String(pulse_count_carriage) + " dist = " + String(dist));
//      Serial.println(String(dist) + " angle:  " + String(angle));
//    Serial.println("pulse_count_carriage "+String(total_pulses));

    pulse_count_pendulum = 0;
    pulse_count_carriage = 0;

    attachInterrupt(digitalPinToInterrupt(pendulum_encoder_pin_A), StartInterrupt_pendulum, RISING);
    attachInterrupt(digitalPinToInterrupt(carriage_encoder_pin_A), StartInterrupt_carriage, RISING);
  }

  dist_dot = (dist-dist_prev)/float(duration);
  angle_dot = (angle-angle_prev)*1000.0/float(duration);
  dist_prev = dist;
  angle_prev = angle;

  current_para[0][0] = float(dist/1000.0);
  current_para[1][0] = float(dist_dot);
  current_para[2][0] = float(angle);
  current_para[3][0] = float(angle_dot);

  if (angle*180.0/pi>0.2){motion = 1;}
  if (motion == 1){
    for(int p = 0; p < 4; p ++) // Calculating error
      {
      error [p][0]= current_para[p][0] - desired_para[p][0];
  //   Serial.println("current "+ String(current_para[p][0])+ " desired "+ String(desired_para[p][0]) + " error
" + String(error[p][0]));
      }

    output = nbar;
    for(int q = 0; q < 4; q ++){
     output += -1*k[q]*error[q][0];
  //    Serial.print(k[q]);
  //     Serial.println("k[q] "+ String(k[q])+ " error "+ String(error[q][0]) + " output " + String(output)+ "
angle "+ String(angle));
     }


    if (output>250.0){output = 245;}
    else if (output<-250.0) {output = -245;}

    if (float(angle*180.0/3.14)>-20.0 and float(angle*180.0/3.14)<20.0){
      pwm = abs(int(output));
      if (output<0.0){
```

```
      dir = 0;
     }
     else{
      dir = 1;
     }
    }
    else {pwm = 0;}
//    Serial.println(String(pwm) + "  "  + String(angle*180.0/pi));
//    Serial.println(output);
  }
//  else {pwm = 0;}
}

void StartInterrupt_pendulum(){
   //Serial.println(digitalRead(pendulum_encoder_pin_B));
   if (!digitalRead(pendulum_encoder_pin_B)){
    pulse_count_pendulum -= 1;
   }
   else{
    pulse_count_pendulum += 1;
   }
}

void StartInterrupt_carriage(){
//    Serial.println(digitalRead(carriage_encoder_pin_B));
   if (!digitalRead(carriage_encoder_pin_B)){
    pulse_count_carriage -= 1;
   }
   else{
        pulse_count_carriage += 1;
   }
//    total_pulses += pulse_count_carriage;
}
```

3. CONCLUSION

Overall, the project can be considered a success. We were able to achieve many tasks such as a working mechanical system was developed along with a control circuit and an accurate feedback network. An interface was also created through hardware and software to integrate the Arduino Mega as the main processing unit for the controller. This made possible for the pendulum to be balanced for a moderate duration of time. It seems that with some fine tuning, it would be quite feasible to balance a pendulum for an extended period of time and even make it capable of swing up control. The system could be further modified using some proximity sensors to avoid collisions. 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.

4. ANNNEXURE AND PICTURES



REFERENCES

1. Dawn Tilbury (2011) Control Tutorials for Matlab and Simulink
2. Brunton, S. Steve Brunton [Youtube Channel]. Retrieved October, 2021, From control bootcamp