

# **Summer Internship Report on Motor driven 8 bar mechanism synchronization system at Taiwan Tech**

**Name: Vivek Singh**

**Roll No: 15ME10066**



**Department: Mechanical Engineering  
Indian Institute of Technology, Kharagpur,  
WB 721302, India**

**Duration : May – July, 2018**

Best regards,



Chyi-Yeu Jerry Lin, Ph.D.

Distinguished Professor, Dept. of Mechanical Eng., NTUST (Taiwan Tech)

Director, Center for Intelligent Robotics, NTUST

Corresponding Member, Russian International Academy of Engineering

PHONE: 886-2-2737-6463~4

FAX: 886-2-2737-6460

## Contents

### I Acknowledgement

### II Introduction

### III Mechanism of 8 bar linkage

III.1 Working Mechanism .....	
III.2 Solidworks Model and Kinematics Diagram .....	
III.3 Position and Velocity analysis .....	
III.4 Motion Simulation by Matlab .....	
III.5 Selection of motor and motor driver.....	
III.6 Approach.....	
III.7 PID control.....	
III.8 Transfer function of DC motor.....	
III.9 Model and Simulation	
III.10 Real Time Implementation.....	
III.11 GUI	

.....

### IV References

## **Acknowledgement**

I would like to thank Professor Jerry Lin, Professor at Taiwan Tech to give me opportunity to do internship within the organization which also helped me in exploring control system area in detail and in leaning many new things.

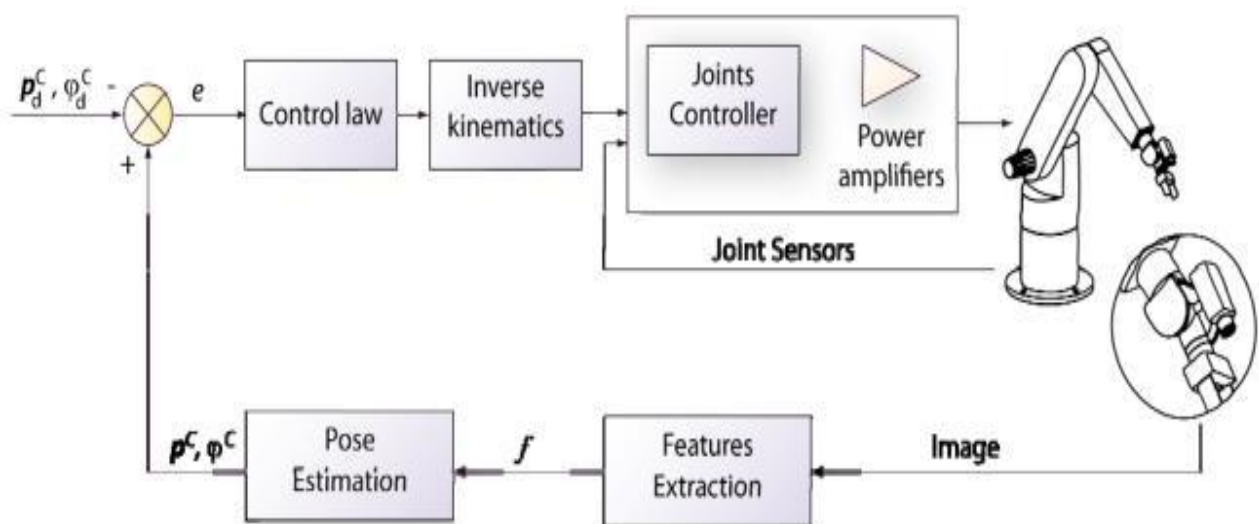
## II INTRODUCTION

### VISUAL SERVOING

**Visual servoing**, also known as **vision-based robot control** and abbreviated **VS**, is a technique which uses feedback information extracted from a vision sensor (visual feedback) to control the motion of a robot. One of the earliest papers that talks about visual servoing was from the SRI International Labs in 1979.

Visual servo systems, also called servoing, have been around since the early 1980s, although the term visual servo itself was only coined in 1987. Visual Servoing is, in essence, a method for robot control where the sensor used is a camera (visual sensor). Servoing consists primarily of two techniques, one involves using information from the image to directly control the degrees of freedom (DOF) of the robot, thus referred to as Image Based Visual Servoing (IBVS). While the other involves the geometric interpretation of the information extracted from the camera, such as estimating the pose of the target and parameters of the camera (assuming some basic model of the target is known). Other servoing classifications exist

based on the variations in each component of a servoing system , e.g. the location of the camera, the two kinds are eye-in-hand and hand–eye configurations. Based on the control loop, the two kinds are end-point-open-loop and end-point-closed-loop. Based on whether the control is applied to the joints (or DOF) directly or as a position command to a robot controller the two types are direct servoing and dynamic look-and-move. Being one of the earliest works the authors proposed a hierarchical visual servo scheme applied to image-based servoing. The technique relies on the assumption that a good set of features can be extracted from the object of interest (e.g. edges, corners and centroids) and used as a partial model along with global models of the scene and robot.

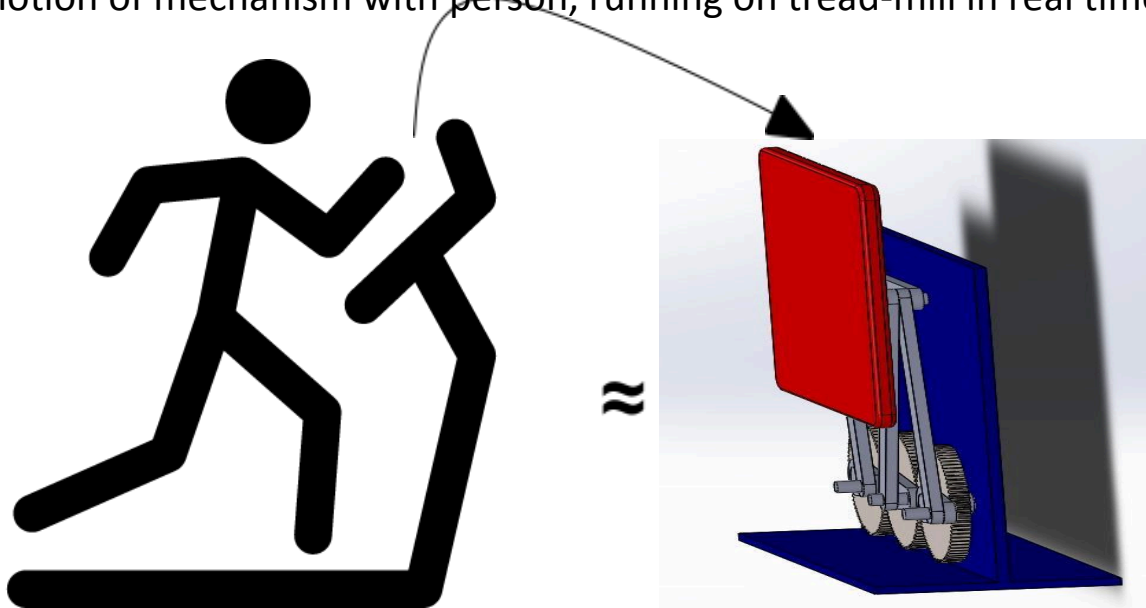


Position-based visual servoing scheme.

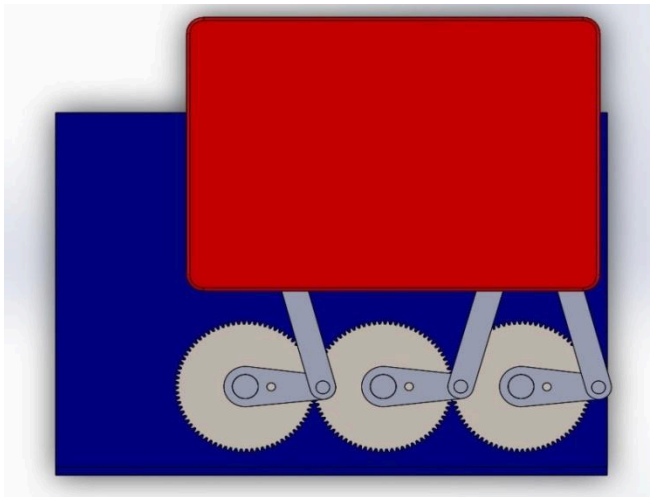
### III PLANER MECHANISM HAVING 8 BAR LINKAGE WITH ONE DEGREE OF FREEDOM

#### Working mechanism

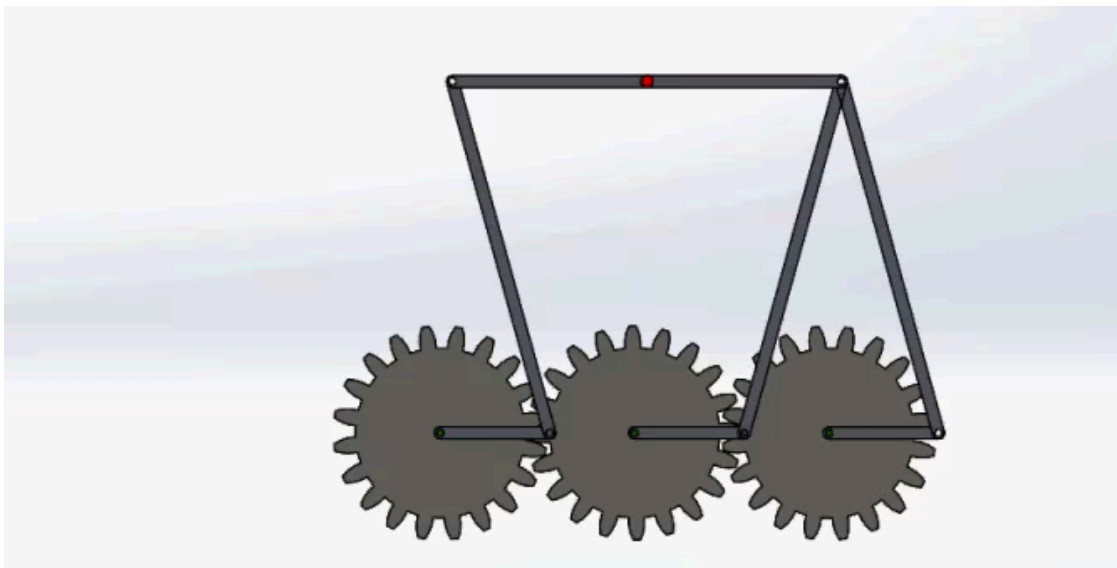
This is the eight bar mechanism having three spur gear , all having the same radius. Mechanism has overall one degree of freedom and when middle gear is connected with DC motor and when middle gear is made to rotate clockwise direction, other two gear start rotating counter clockwise direction. Mechanism has either tablet or iPad mounted on it and it's motion is approximately like the motion of moving face of person, running on tread-mill. Goal is to synchronize motion of mechanism with person, running on tread-mill in real time.



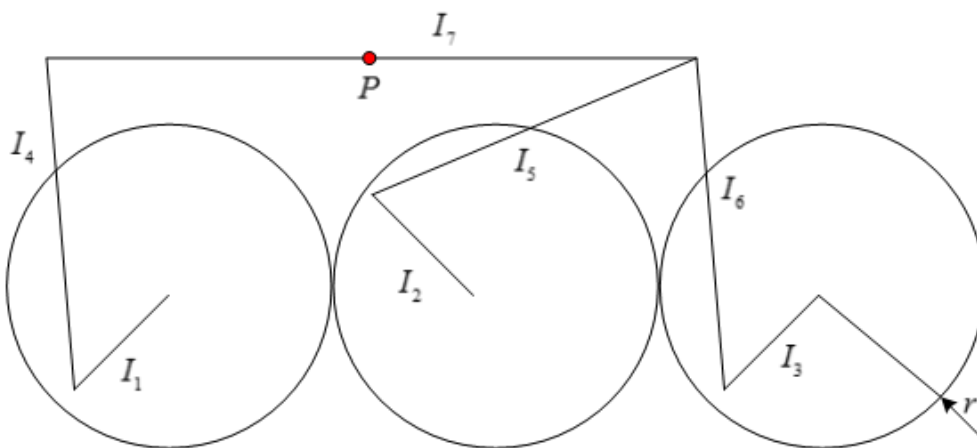
**Moving face of person running on treadmill is approximately equivalent to motion of centroid of tablet/iPad attached to this eight bar gear driven mechanism.**



Eight bar gear driven Mechanism

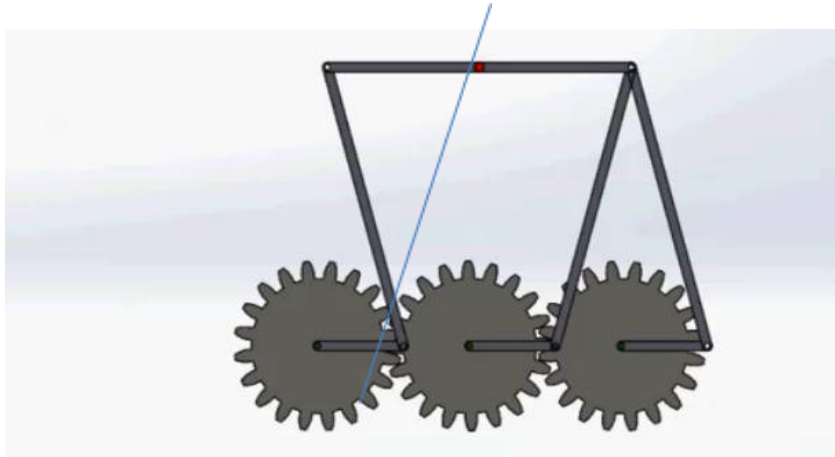


Solidworks model of Mechanism



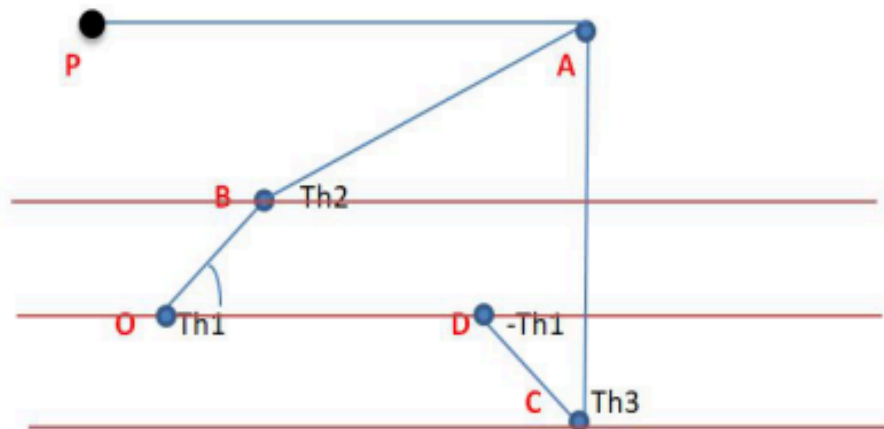
Kinematic diagram of eight bar mechanism

## POSITION AND VELOCITY ANALYSIS OF MID POINT (HORIZONTAL LINK)





FIVE BAR LINKAGE MECHANISM (BOTH INPUT ANGLE ARE SAME BUT OPPOSITE OF EACH OTHER)



## THIS IS THE FIVE BAR LINKAGE MECHANISM

HORIZONTAL ROD IS CONNECTED TO END EFFECTOR OF THIS FIVE BAR LINKAGE

## POSITION ANALYSIS

- $OB = a$ ,  $BA = b$ ,  $AC = c$ ,  $DC = d$ ,  $OD = e$ ,  $PA = L$
- $th1$  and  $th4$  are two independent input angles but in our case it is  $th1 = -th4$ ;
- $A = 2*c*d*\sin(th4) - 2*a*c*\cos(th1)$ ;
- $B = 2*c*e - 2*a*c*\cos(th1) + 2*c*d*\cos(th4)$ ;
- $C = a*a - b*b + c*c + d*d + e*e - a*d*\sin(th1)*\sin(th4) - 2*a*e*\cos(th1) + 2*d*e*\cos(th4) - 2*a*d*\cos(th1)*\cos(th4)$ ;
- $M = \sqrt{A*A + B*B - C*C}$ ;
- $th3 = [2*atan((A+M)/(B-C))]$
- $th2 = \arcsin((c*\sin(th3) + d*\sin(th4) - a*\sin(th1))/(b))$ ;

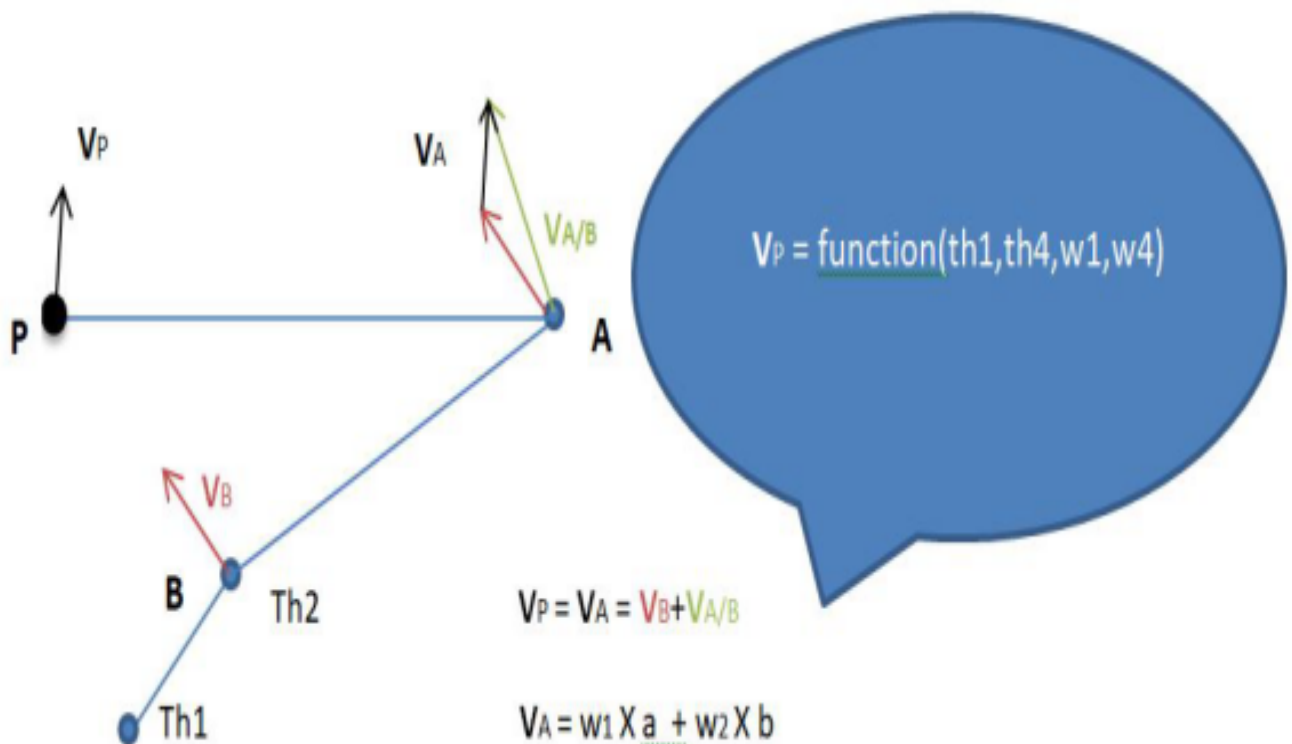
➤  $\theta_2, \theta_3 = f(\theta_1, \theta_4)$

➤  $A_x = a \cdot \cos(\theta_1) + b \cdot \cos(\theta_2)$






➤  $A_y = a \cdot \sin(\theta_1) + b \cdot \sin(\theta_2)$

➤  $(P_x, P_y) = (-L + A_x, A_y)$

➤ **Position of P = function( $\theta_1, \theta_4$ )**

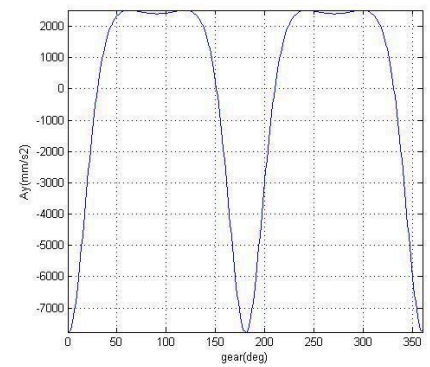
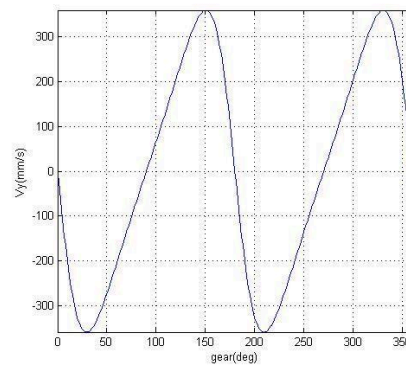
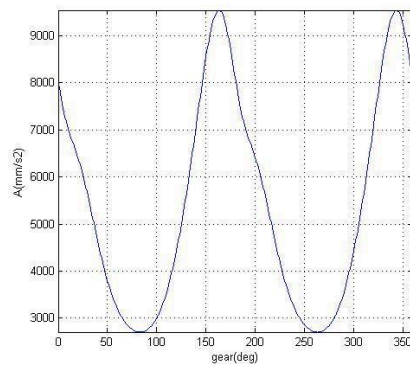
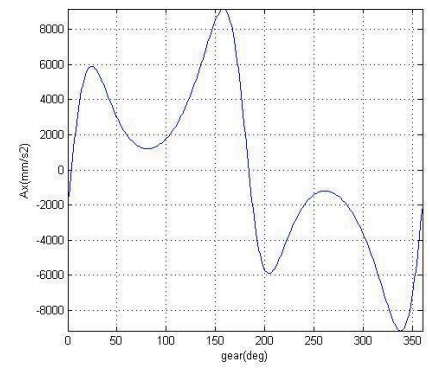
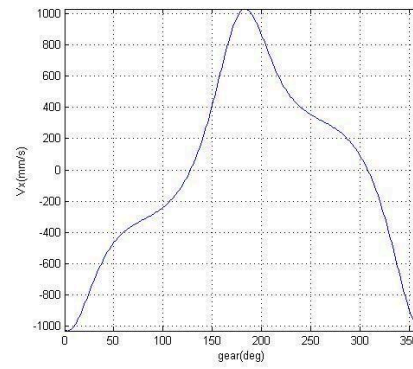
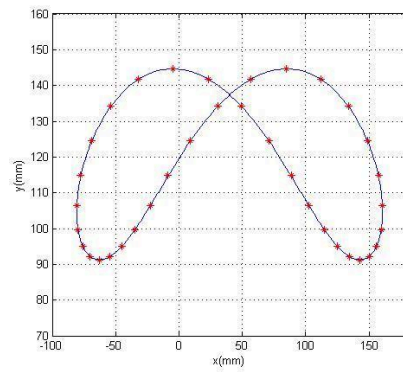


**MOTION OF CENTROID OF MID POINT OF HORIZONTAL BAR (P) VS DIFFERENT LENGTH OG BAR**

r	I1	I2	I3	I4	I5	I6	I7	
1	0.75	0.75	0.75	1.875	1.875	1.875	4	
1	0.75	0.75	0.75	2.75	2.75	2.75	4	
1	1.125	1.125	1.125	2.625	2.625	2.625	4	
1	1.125	1.125	1.125	3	3	3	4	
1	1.125	1.125	1.125	3.75	3.75	3.75	4	

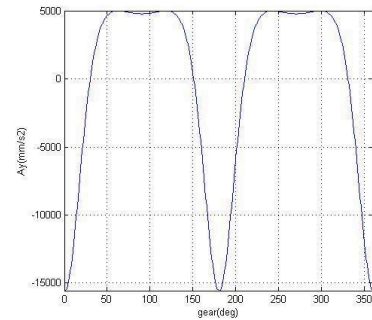
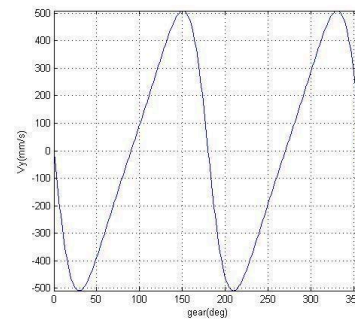
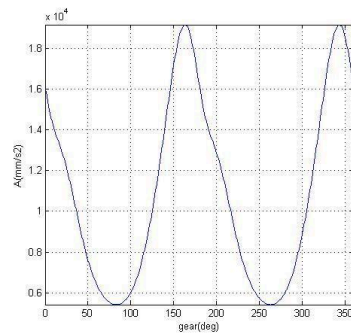
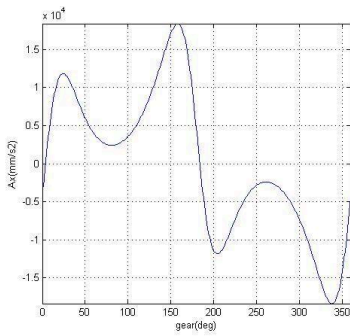
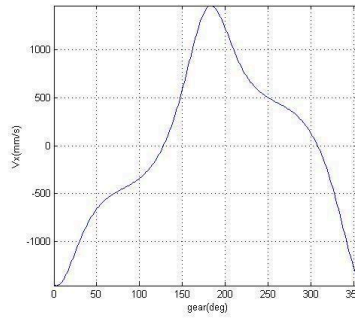
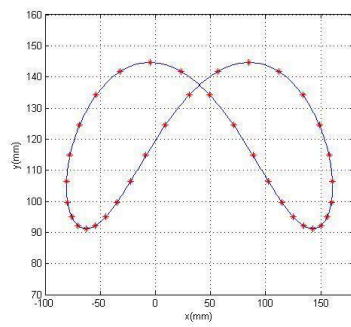
### MOTION SIMULATION #1 BY MATLAB

**Input angular velocity: 60.0 RPM, T = 1.0 sec, Amax = 9.5 m/s<sup>2</sup>**



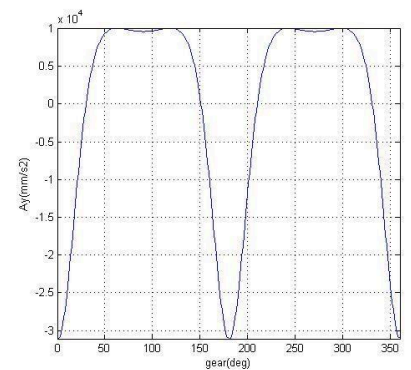
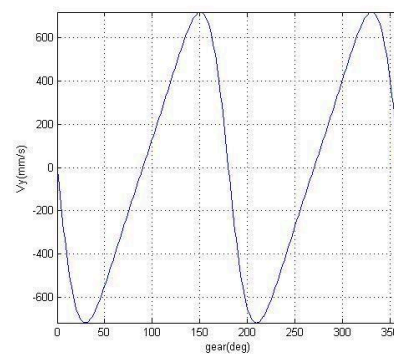
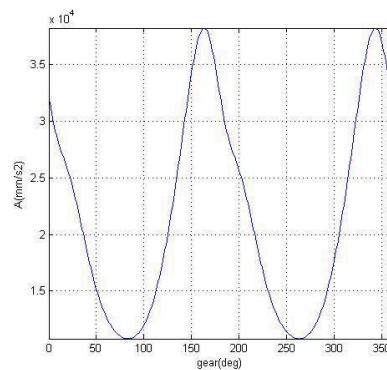
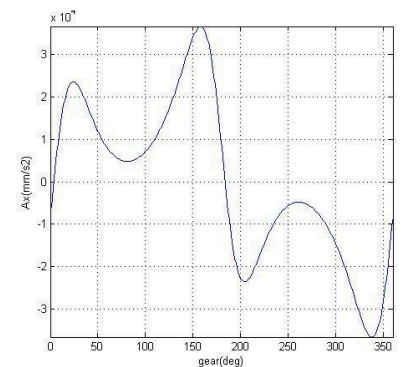
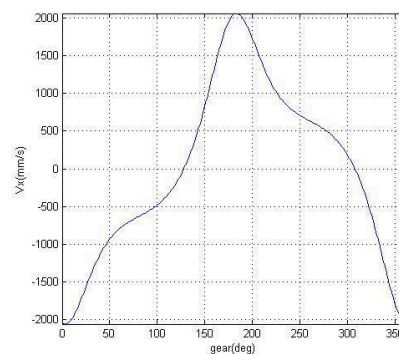
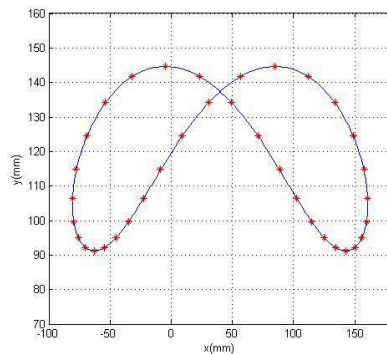
## MOTION SIMULATION #1 BY MATLAB

**Input angular velocity: 85.0 RPM, T= 0.75 sec, Amax= 19.2 m/s<sup>2</sup>**



## MOTION SIMULATION #1 BY MATLAB

**Input angular velocity: 120.0 RPM, T= 0.5 sec, Amax= 38.2 m/s<sup>2</sup>**



**Case1:** Input angular velocity: 60.0 RPM,  $T = 1.0$  sec,  $A_{max} = 9.5$  m/s<sup>2</sup>

**Case2:** Input angular velocity: 85.0 RPM,  $T = 0.75$  sec,  $A_{max} = 19.2$  m/s<sup>2</sup>

**Case3:** Input angular velocity: 120.0 RPM,  $T = 0.5$  sec,  $A_{max} = 38.2$  m/s<sup>2</sup>

### **Conclusion:**

Case2 (period 0.75 sec) is the most similar as real experiment.

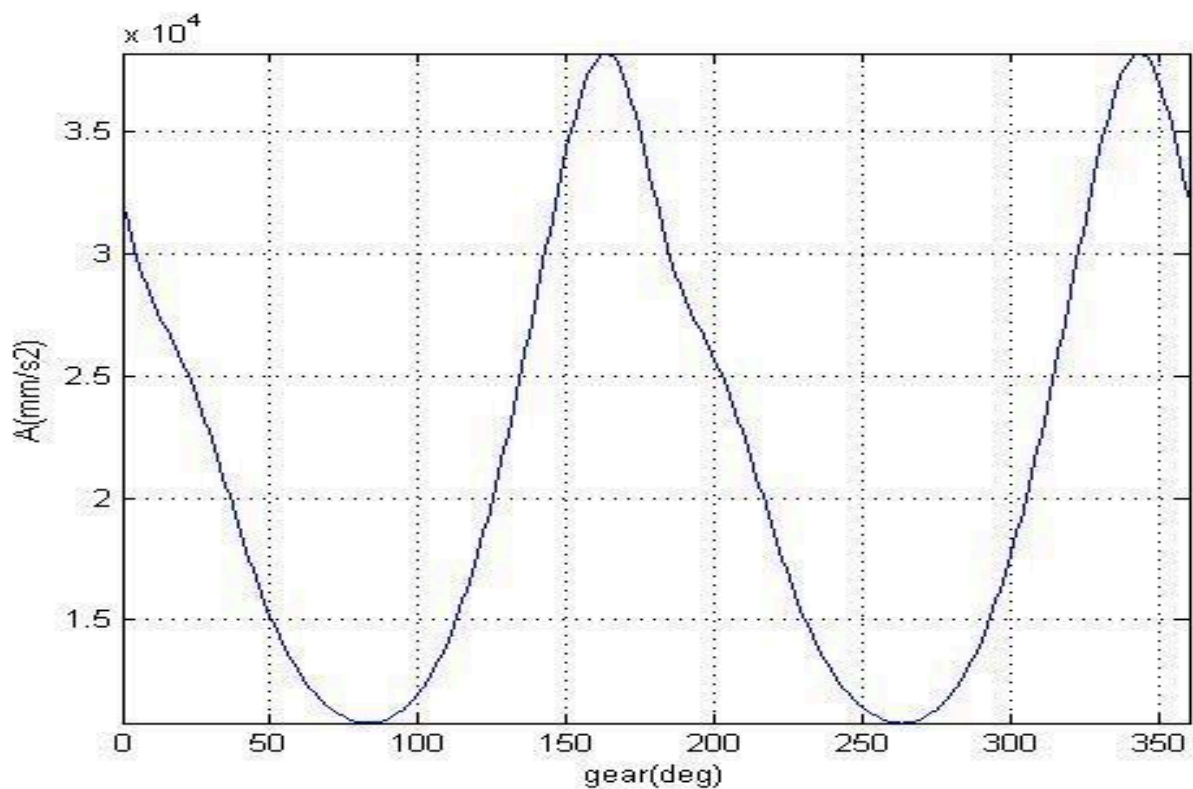
Motor speed : 85 RPM, but consider efficiency=50% in high loading  
→ Motor

max speed: 170 RPM

## **POSITION OF MAXIMUM ACCELERATION**

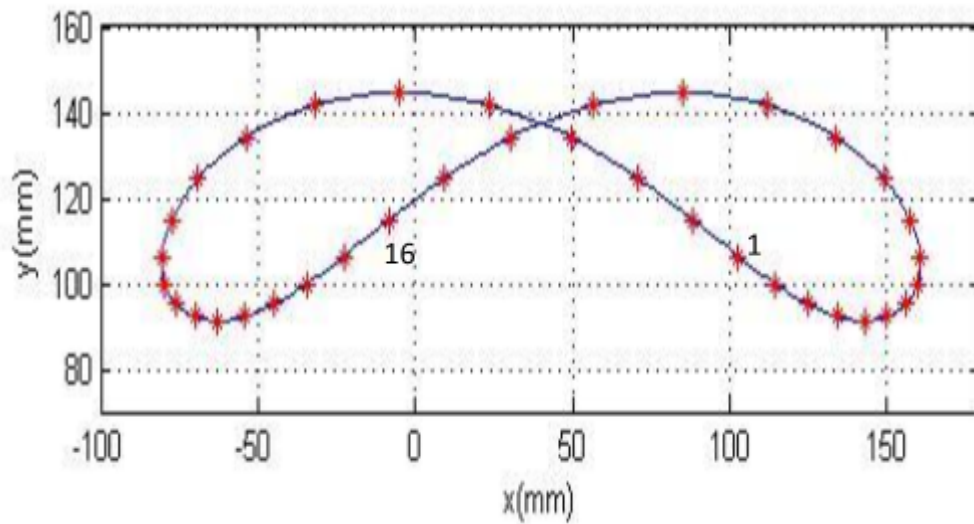
Time-Period of 0.75 is very similar to time period of real person running on treadmill. So Simulation was done to get maximum acceleration of mid point of horizontal bar during period over gear rotation angle. Displacement in y direction and x direction is measured to get the trajectory of the mid point (p) and its trajectory is approximately same as centroid of face person running on treadmill.

- X axis: 140 mm, Z: 60 mm
- Position of S point (85,144.5)
- Position of M point (45.8,139.0)
- Max acceleration is at  $160^\circ$ ,  $340^\circ$
- Position at  $160^\circ$  is ( -55.7, 133.2)



**Acceleration( $\text{mm/s}^2$ ) vs gear rotation (degree)**





**Trajectory generated by mid point (P)**

### SELECTION OF MOTOR AND MOTOR DRIVER

#### ▪ Motor:

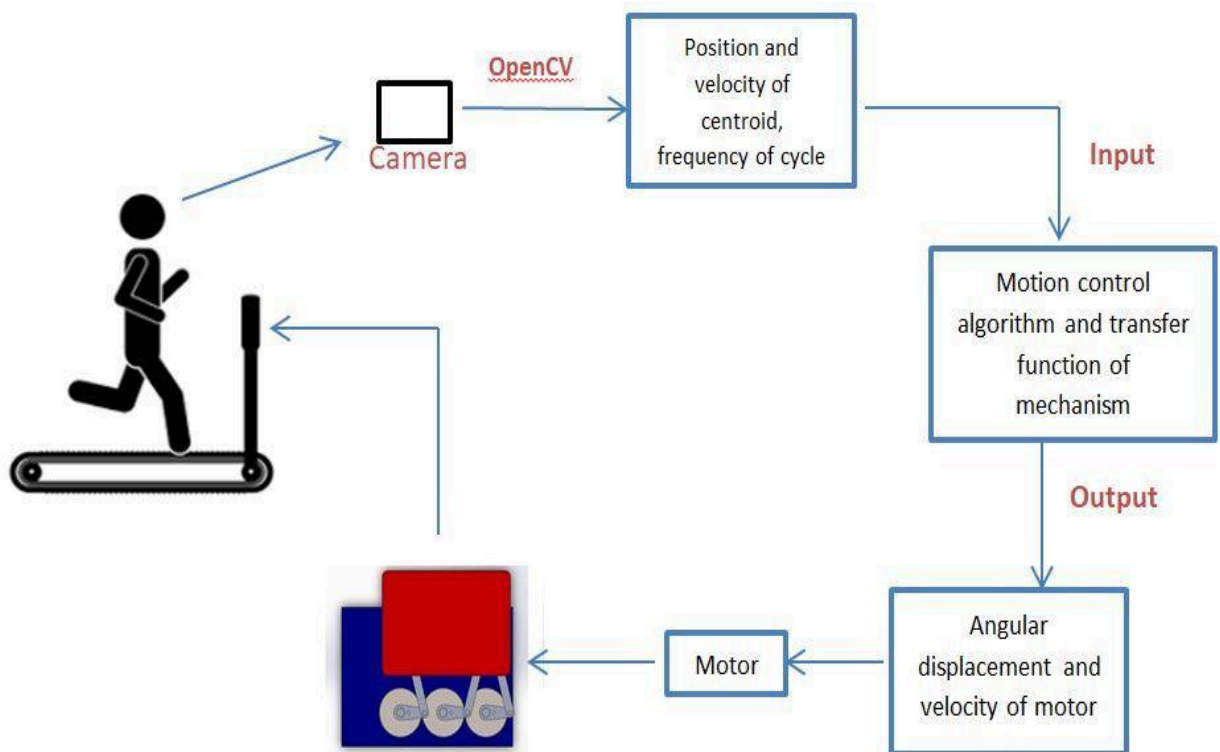
- Type: 12/24V DC motor
- Speed: 85 RPM (max: 170 RPM)
- Torque: ?
- Size:  $\phi 32$  (mm)
- Price: 1000~3000 NT

#### ▪ Driver :

- DC Carbon-brush motor
  - IC L298N driver module
  - Price: 300 NT
  - Performance: to be tested
- DC Brushless motor
  - Including driver

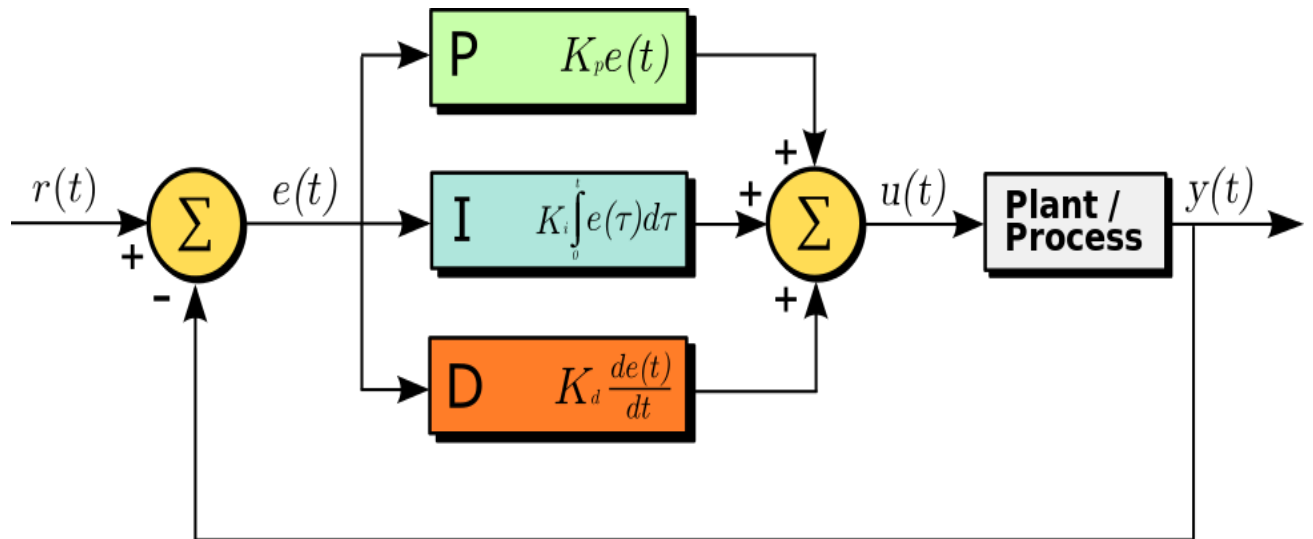
### APPROACH





## PID CONTROLLER

A **proportional–integral–derivative controller (PID controller or three term controller)** is a control loop feedback mechanism widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an error value  $e(t)$  as the difference between a desired setpoint (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted  $P$ ,  $I$ , and  $D$  respectively), hence the name.



## MATHEMATICAL FORM

The overall control function can be expressed mathematically as

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

where

$K_p$  is the proportional gain, a tuning parameter,

$K_i$  is the integral gain, a tuning parameter,

$K_d$  is the derivative gain, a tuning parameter,

$e(t) = SP - PV(t)$  is the error (SP is the setpoint, and PV(t) is the process variable),

$t$  is the time or instantaneous time (the present),

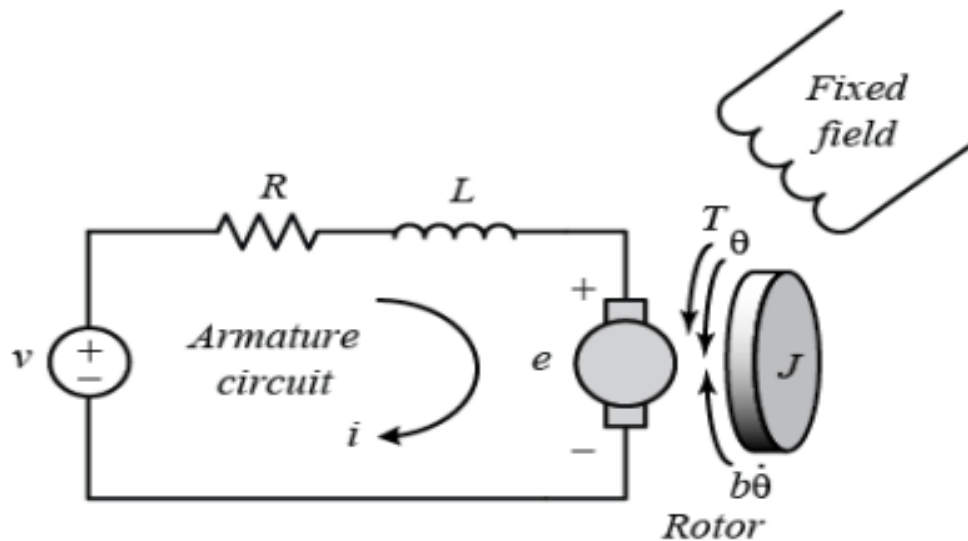
$\tau$  is the variable of integration (takes on values from time 0 to the present  $t$ )

Equivalently, the transfer function in the Laplace domain of the PID controller is

$$L(s) = K_p + K_i/s + K_d s,$$

where ' $s$ ' is the complex frequency.

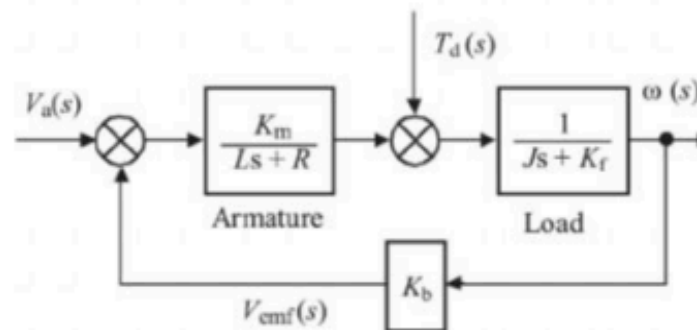
## TRANSFER FUNCTION OF ARMATURE CONTROLLED DC MOTOR



## DC MOTOR CONTROL PARAMETER

We are assuming this motor with constant field current, basically armature controlled circuit

## Block diagram of armature controlled d. c. motor



- $\omega(s)/V_a(s) = K_m / [(R + Ls)(Js + K_f) + K_b K_m]$
- Relation between angular speed (output) and voltage (input) given to motor. That is relation between mechanical property and electrical property.

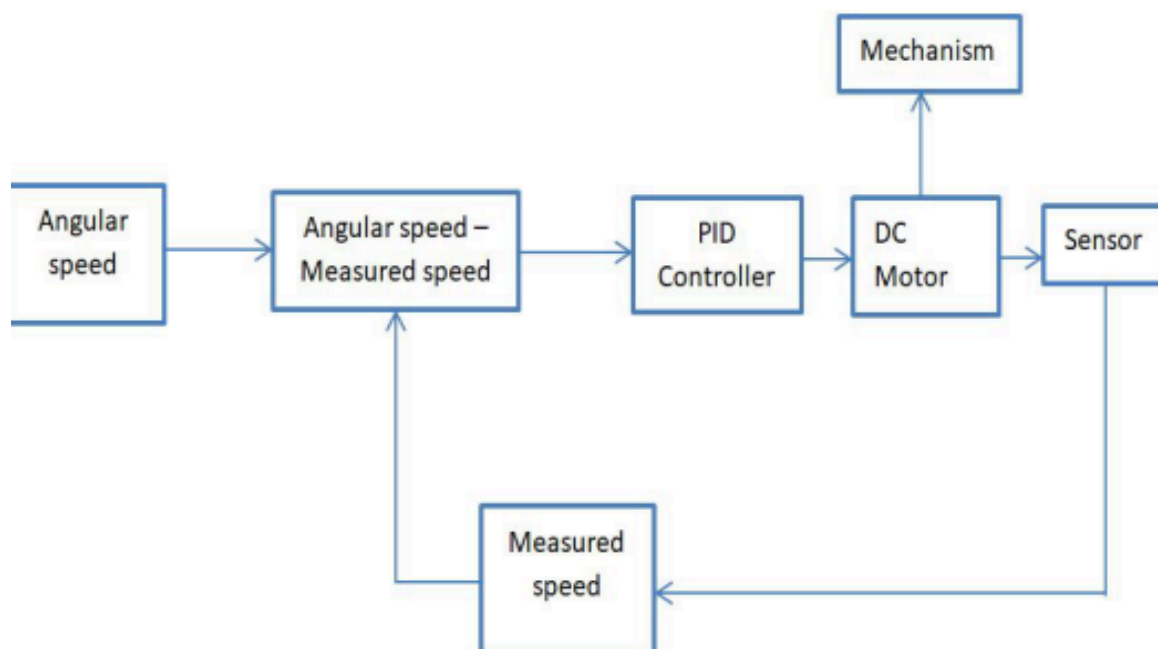
$\omega$  – angular speed,  $V_a$  – input voltage,  $R$  – resistance,  $L$  – inductance,  $J$  – inertia,  $K_f$  – damping constant,  $K_b$  – back emf constant,  $K_m$  – torque constant

## Model and simulate dynamic system behaviour with MATLAB, Simulink

Modelling is a way to create a virtual representation of a real-world system that includes software and hardware. If the software components of this model are driven by mathematical relationships, you can simulate this virtual representation under a wide range of conditions to see how it behaves.

Modelling and simulation are especially valuable for testing conditions that might be difficult to reproduce with hardware prototypes alone, especially in the early phase of the design process when hardware may not be available. Iterating between modelling and simulation can improve the quality of the system design early, thereby reducing the number of errors found later in the design process.

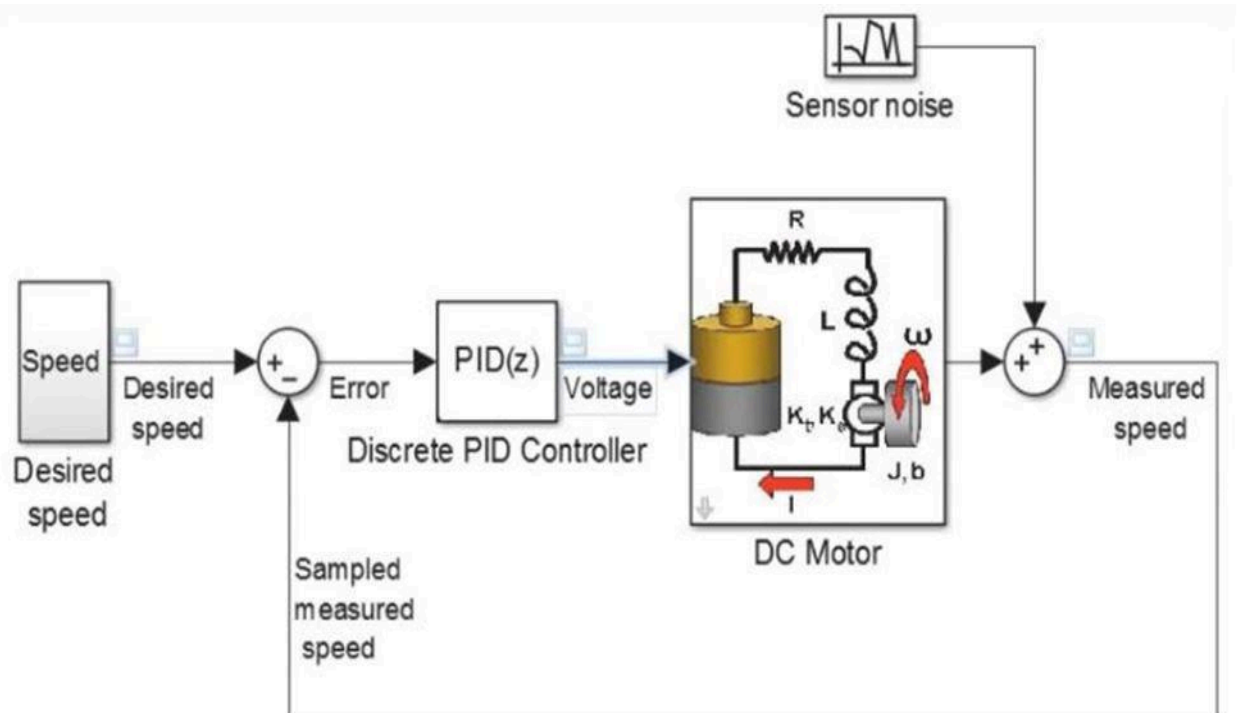
## PROCESS



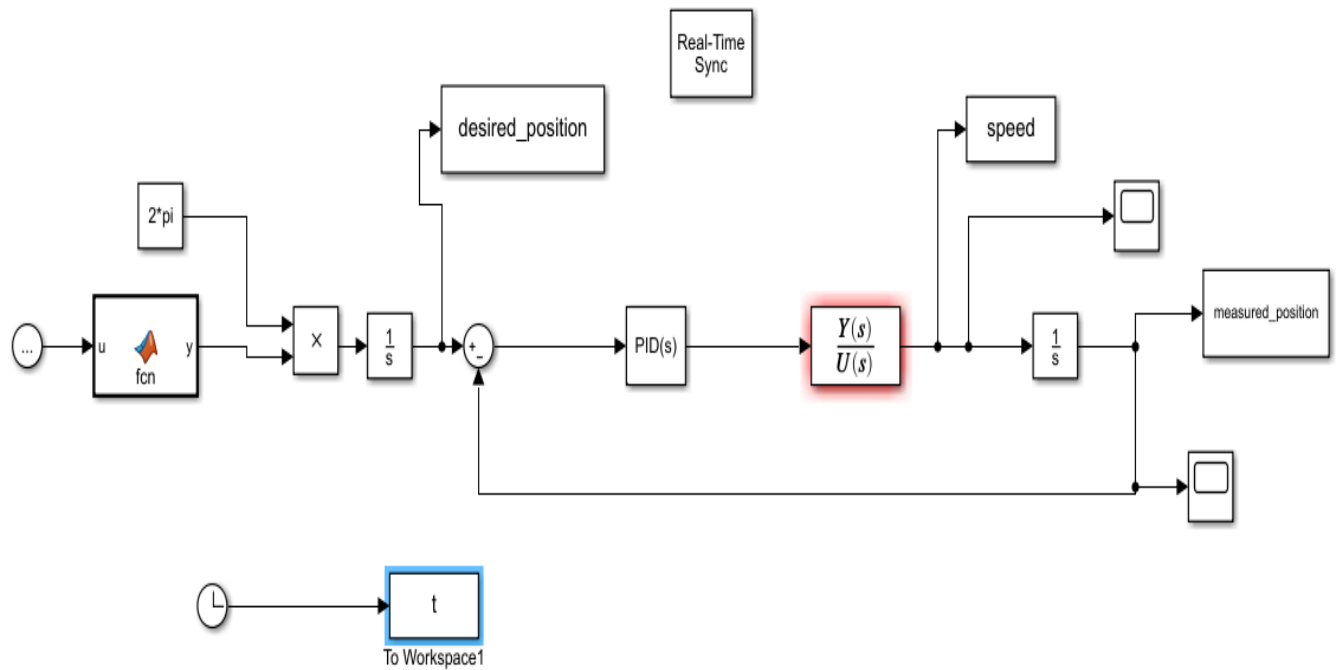
Person running on treadmill can vary their motion or it can change their gait of moving and shift to another gait .This changes the motion of its face and finally it can change the frequency of the moving face. In order to synchronize with face motion , frequency of mechanism has to be changed in order to get that .

In simulink model , a mathematical function i.e. frequency with function of sample time and then this frequency is converted into position. Position from motor is taken and error term is calculated.

PID controller acts on error term and generate velocity/ voltage such that it can minimize error term over the period of time.



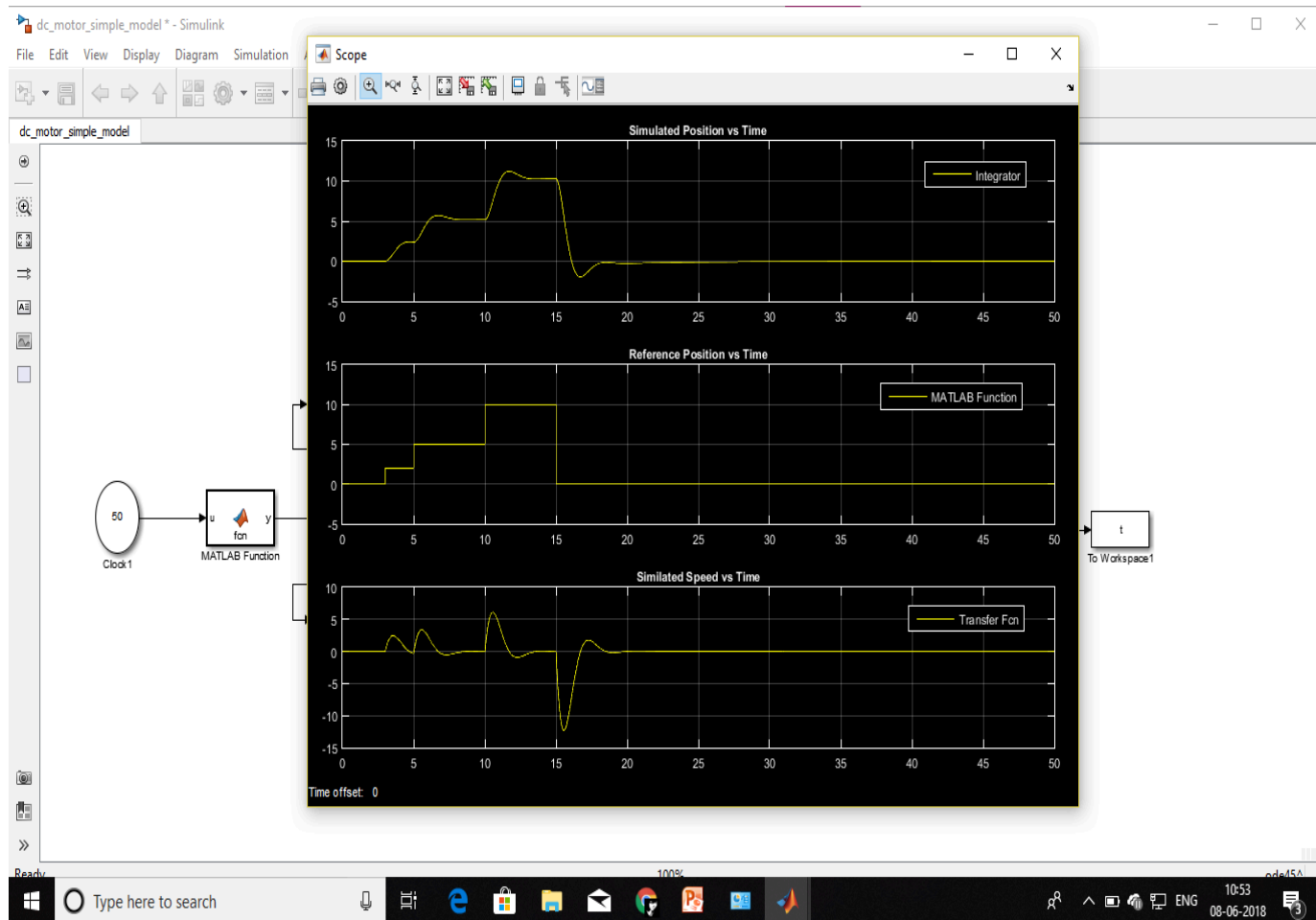
**SIMULATION IN MATLAB & SIMULINK**



Block Diagram in Simulink

## Simulation result :

- Reference position vs sample time
- Simulated position vs sample time
- Simulated speed of motor vs sample time.



Scope result from Matlab & Simulink

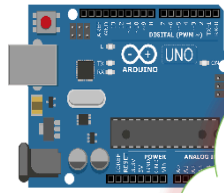
## Arduino Hardware Support packages:

Initially there was serial communication between Arduino and Matlab but we were getting wrong value of frequency of face generating same trajectory as of mechanism. So to avoid this, use direct communication between arduino and matlab using Arduino Hardware Support packages for Matlab and Simulink





## Previous Work



Serial communication is done and getting a bunch of 0 & 1.

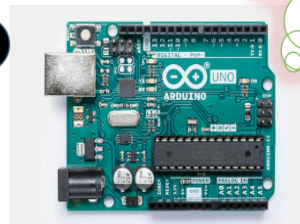
1 1 1 1 1

0 0 0 0 0

1 0 1 0 1

0 0 0 1 0

Wrong Value  
of Frequency



## Real time Implementation :

We use two arduino, we get velocity and the sequence of time at which centroid of face passing through a particular point and we call this time period and inverse of it's is frequency. This frequency is converted into rpm of DC motor. So our mechanism, in few second approx 5-6 sec achieve the frequency at which person running on treadmill. So frequency part is synchronized . But another part i.e. phase difference is left, so in order to synchronize this means when view from front plane, we can see that both i.e mid point of tablet/iPad attached with mechanism and centroid of person face running on treadmill is synchronized with each other as a perspective of both frequency as well as phase.

This is the code for PID controller in Matlab as there was problem in running Simulink model for the motion control of DC motor because

Simulink was not supporting string command which is sent to motor driver.

Error term is difference in phase angle between person's moving face and mechanism . What PID does on this is that whenever this error term is positive , velocity(rpm) of motor increases in order to synchronize with moving face of person on treadmill. When this error term is negative , velocity(rpm) of motor decreases in order to synchronize with moving face of person on treadmill.

This velocity is sent to motor driver through string command and there is serial communication between motor driver and Matlab.

```
%fprintf(obj,str);
%readasync(obj);
%obj.ReadAsyncMode = 'manual';
dt = t8(k-1);
disp('Received signal at this moment');
A1 = 'v';
p(k-1) = (500*3.08/t8(k-1)+0.008);
v = (round(p(k-1) + 0.1*e + 0.003*integral + 0.2*differential));
if v>2500
    v =2400;

elseif v<0
    v=0;
end

pos_act = mod(str2num(fscanf(obj)),1423);
fprintf(obj,'pos');
out1 = fscanf(obj);
if pos_act <710
    e = pos-pos_act;
    fprintf('relative position is %d',pos_act);
elseif pos_act>710
    e = 1423-pos_act;
    fprintf('relative posotion is %d',pos_act-1423);
```

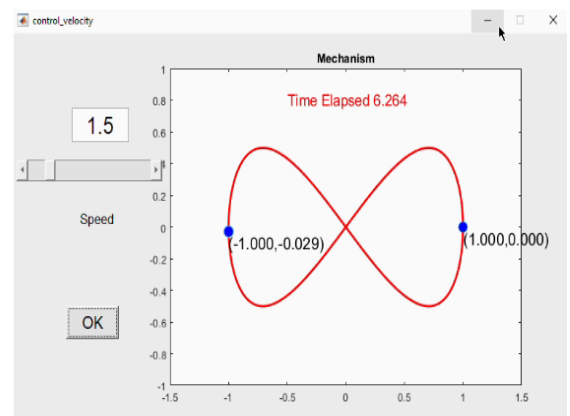
**Matlab code for PID control**

## **GRAPHICAL USER INTERFACE FOR CONTROLLING RPM OF MOTOR:**

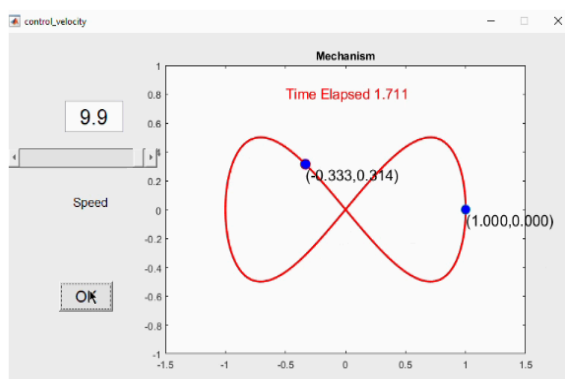
Initially velocity command is sent to motor driver via string using serial communication. So in order to reduce complexity, GUI is made. We can change rpm of motor with help of slider or with help of entering rpm of motor in it.

Controlling the  
Motor with  
Variable  
frequency/speed

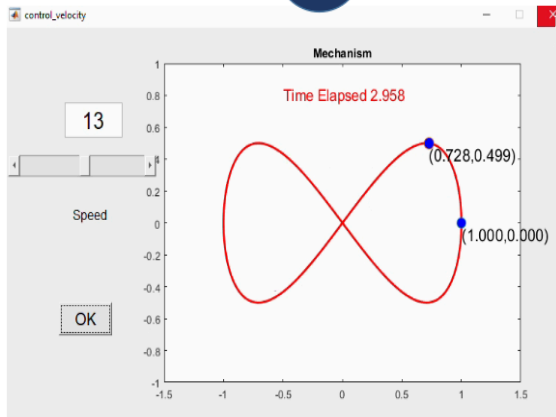
1.5



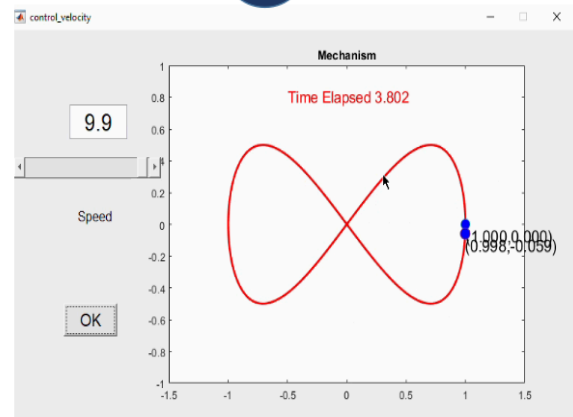
5.6



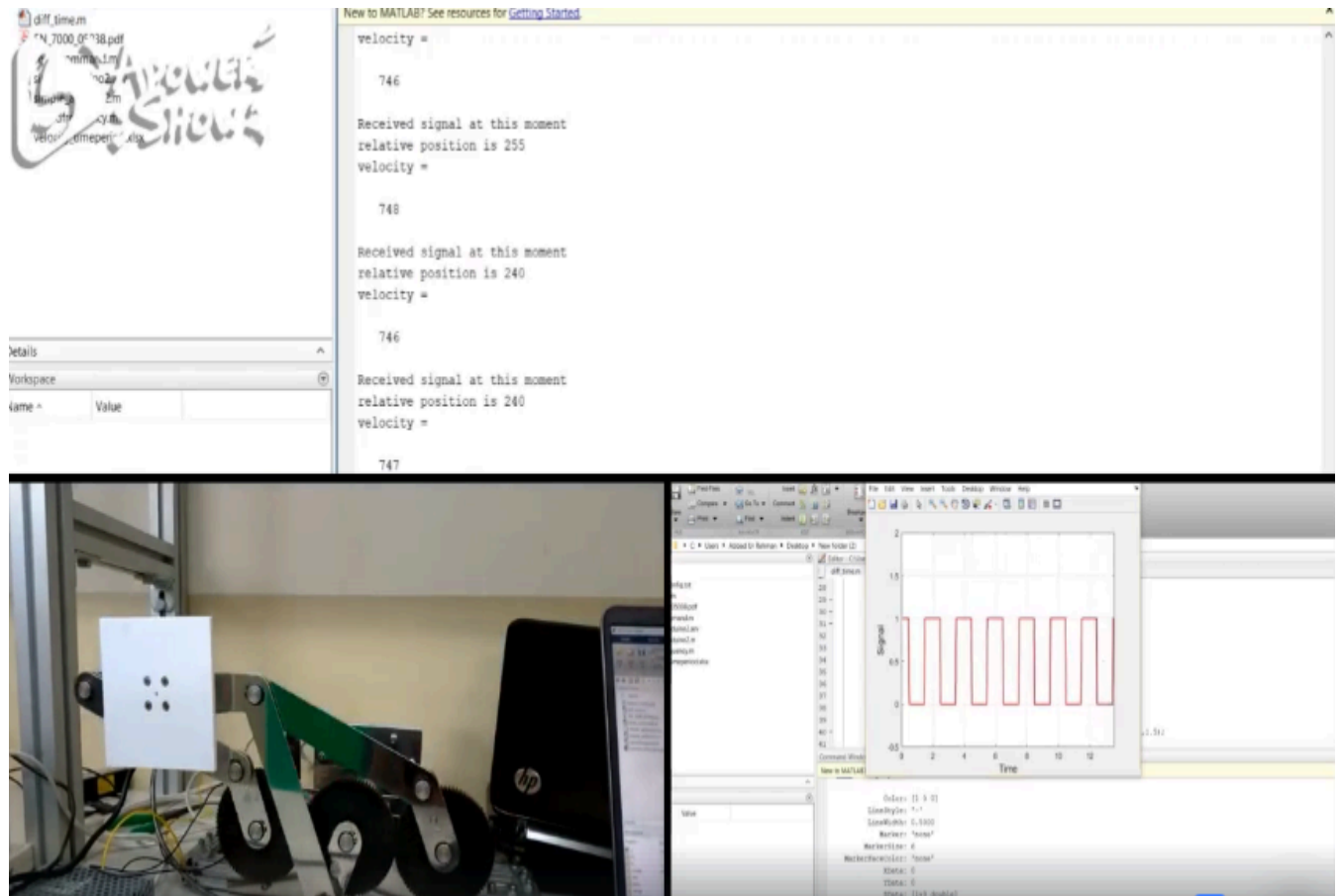
13



9.9



**Mechanism Running in real time:**



## REFERENCES:

[https://en.wikipedia.org/wiki/PID\\_controller](https://en.wikipedia.org/wiki/PID_controller)

[https://en.wikipedia.org/wiki/Visual\\_servoing](https://en.wikipedia.org/wiki/Visual_servoing)

[https://www.researchgate.net/figure/Position-based-visual-servoing-scheme\\_fig4\\_40755519](https://www.researchgate.net/figure/Position-based-visual-servoing-scheme_fig4_40755519)

Modern Control System book on Richard C. Dorf ,University of California, Davis ,Robert H. Bishop Marquette University



