University of Applied Science Hochschule Bremerhaven

Embedded System Design

Bendkhale, Ruturaj Rajendra [Matriculation No: 39377]

Siddiqui, Mohammad Amaan Ali [Matriculation No: 39391]

Khan, Mahad Raza [Matriculation No: 39411]

Prajapati, Prashantkumar [Matriculation No: 39417]

Kayani, Khubaib Ali [Matriculation No: 39420]

Suryadevara, Sravya [Matriculation No: 39451]

Hochschule Bremerhaven

Magnetic Levitation Report

Date of Submission: 12.07.2023

Contents

List of Figures

List of Tables

Acknowledgement

We would like to take this opportunity to express our sincere gratitude to our esteemed professors and colleagues who played a crucial role in helping us successfully complete this academic project. Our special thanks go to our project guide, Prof. Dr.-Ing Karsten Peter, and Prof. Dr.-Ing Kai Mueller, for their invaluable contributions that not only stimulated our intellectual growth but also provided us with constant guidance and suggestions throughout the project.

We also acknowledge the assistance provided by the staff of Embedded Systems Design at Hochschule Bremerhaven. Their support was essential, and we are grateful for their permission to use the necessary equipment for our project, "Magnetic Levitation Control."

Lastly, we extend our heartfelt thanks to our colleagues for their unwavering support and for being an integral part of this journey. Their companionship and assistance during challenging times were truly appreciated.

Work Organization

Sr. no	Team Members	Contribution
1	Bendkhale, Ruturaj Rajendra	Chapters: 2, 3, 4, 6, , 8.2.2, 8.3.2, 8.4.3, 10
2	Siddiqui, Mohammad Amaan Ali	Chapters: 11, 12, 13
3	Khan, Mahad Raza	Chapters: 8.1, 8.2, 8.3, 9
4	Prajapati Prashantkumar	Chapters: 7
5	Kayani Khubaib Ali	Chapters: 5
6	Suryadevara, Sravya	Chapters: 8.4

List of Variables

ref_pos	Reference position of the ball
fx_ref_pos	Reference position of the ball in fixed points
dist_eq	equilibrium distance
dist_eq_m	equilibrium distance in meters
fx_eq_dist	fixed point equilibrium distance
crnt_pos	current position of the ball
fx_crnt_pos	fixed point current position
crnt_pos_adc	current position of the ball in 12bit ADC
del_ref_pos	change in current position
fx_del_ref_pos	fixed point change in current position
triang_ref	input reference to the controller in triangular form
fx_triang_ref	fixed point input reference to the controller in triangular form
sqre_ref	input reference to the controller in square form
fx_sqre_ref	fixed point input reference to the controller in square form
sine_ref	input reference to the controller in sine form
fx_sine_ref	fixed point input reference to the controller in sine form
const_ref	input reference to the controller in constant form
fx_const_ref	fixed point input reference to the controller in constant form
fx_cont_out	fixed point controller output
fx_prop_out	fixed point proportional
fx_drv_1	fixed point derivative with current position
fx_drv_0	fixed point derivative with last position
fx_drv_out	fixed point derivative
fx_intg_1	fixed point integral with current position
fx_intg_0	fixed point integral with last position
fx_intg_out	fixed point integral
fx_eqC_V	fixed point equilibrium current(I) voltage(V)
a1	polynomial coefficient of proportional
a2	polynomial constant of proportional
b1	polynomial coefficients of derivative
b2	polynomial coefficients of derivative
c1	polynomial constant of derivative
p1	polynomial coefficients of equilibrium voltage

p2	polynomial coefficients of equilibrium voltage
fx_manu_var	manipulating variable
12_bit_output	output with size of 12 bit
DAC_ip	12bit output to DAC
v_drv_max	maximum driving current
Kp	Propotional gain
Ki	Integral gain
Kd	Derivative gain
Kps	Proportional gain scaled
Kds	Integral gain scaled
Kis	Derivative gain scaled
V_eq	Equilibrium voltage

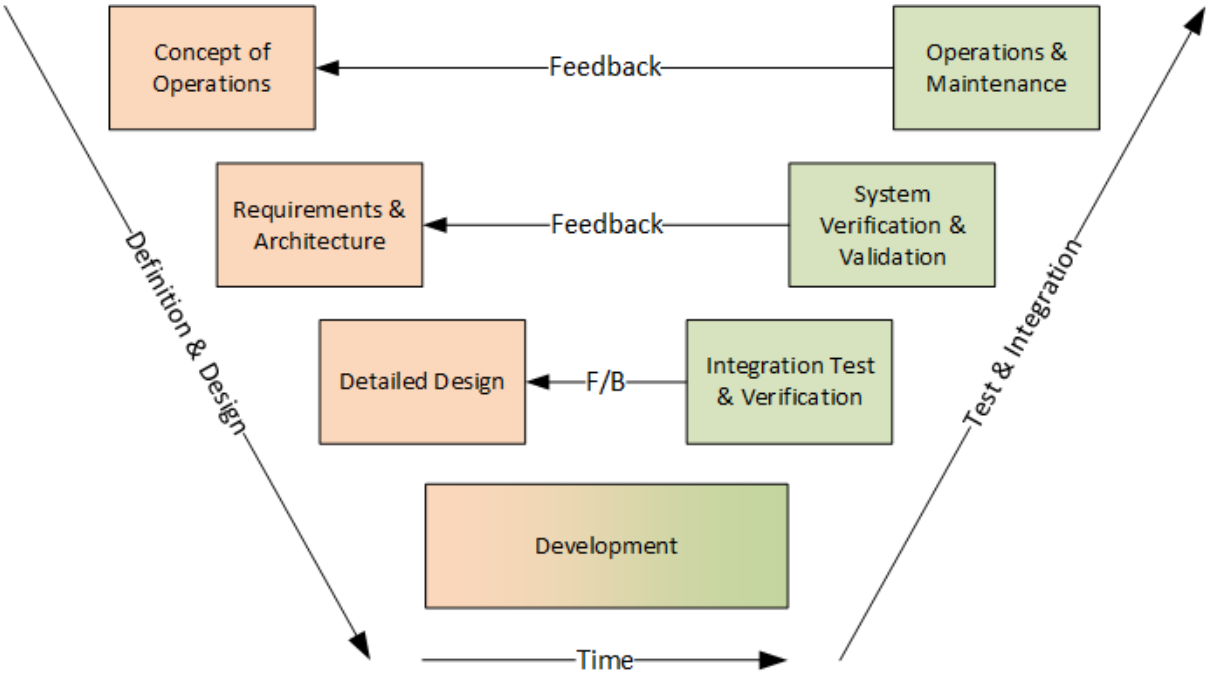
Table 1: Constant Signal Variables

Abstract

A 20th century initial concept of Magnetic Levitation is implemented in this document on a system consist of a ferromagnetic ball levitated by a current carrying conductor, which induced magnetic filed to levitate the ball. Object distance form the base extracted with the help of a laser-based sensor. To perform the transition from non-linear to a linear state of the system a widely used feedback controller, Proportional integral and derivative is used. An equilibrium current from feedback distance added to the compensate the value of PID controller. Complete approach Modeled, simulated and tested using MATLAB®. Different techniques for controlling are also discussed in the document. Zedboard® Zynq®-7000 FPGA(Field Programmable Gated Array) employed for implementation with PCI(Peripheral Component Interconnect) and UART(Universal asynchronous receiver-transmitter) communication protocols. Because of easy access to simulation data a prototype UI(user interface) designed and developed on MATLAB as a prototype, allows user to levitate the object on a particular position. For final submission a Java based UI(user interface) developed keeping same functionality for user as in prototype.

Project Lifecycle

The life cycle of this project can be understood by a V model. This helps in tracking the progress of execution of our project.



Concept of operations: In this phase, the top-level requirements were analyzed.

Requirements and Architecture: During this phase, the high-level requirements were divided into more detailed low-level requirements. These requirements were categorized into Modelling, Hardware, Software, and GUI requirements. Additionally, the fundamental structure of the controller and the magnetic levitation system was conceptualized in this phase.

Design Phase: In the design phase, the system underwent mathematical analysis, and a Simulink model was created for both the controller and the plant. During the design phase, a prototype Graphical User Interface (GUI) was created using MATLAB App Designer. This choice was made due to the convenient availability of simulated data and the minimized occurrence of inter-system communication issues.

Development Phase: During this phase, the controllers were adjusted to match the actual hardware, and an S-function was developed for Software-in-Loop (SIL) testing. The control algorithm was implemented using the S-function, and the Zynq board was programmed. Additionally, a graphical user interface (GUI) was developed using Java.

Integration test and verification: This phase ran concurrently with the design phase. Each module of the system, including the plant, time-continuous and discrete controllers and the fixed-point version of the controller, was validated through Model-in-Loop (MIL) testing. The S-functions for the position were also validated using Simulink.

System verification and validation: The complete system's behaviour was validated

through simulation. Additionally, the control algorithm underwent implementation on the Zynq board.

The requirements were categorized as follows:

Simulation Requirements: Simulation Requirement consist of Realisation of the physical plant and developing a controller.

Modelling Requirements: This consist of physical implementation of developed control logic on a discrete system.

Software Requirements: These focused on the behavioral implementation.

GUI Requirements: These were related to user interaction.

Safety requirement: These were defined to reduce risks

Additionally, each requirement was further classified into different types:

Design Requirement: These attributes facilitated the transformation of ideas into design features.

Functional Requirement: These defined the system's functions and specified its behavior.

Non-functional Requirement: These assessed the system's operation rather than its specific behaviour.

The provided top-level requirements were thoroughly studied and analyzed to ensure a complete understanding of the project requirements. A low-level requirement document, similar to a system requirement specification, was created as a result of this analysis phase. Please refer to the appendix for a detailed document. This document outlined all the necessary features and functions that the product should possess. The requirements were formulated in a clear and unambiguous manner, making them easily understandable for everyone involved. This phase played a crucial role in identifying specific tasks and allowed for the detection of potential mistakes before entering the development stage, thus saving time. The availability of detailed requirements also facilitated more accurate estimation of the time required for implementing specific features by the team.

1 Introduction

1.1 Defination

Magnetic Levitation(MagLev) is also known as magnetic suspension. In magnetic levitation, an object is suspended in mid-air without using mechanical force, with the help of magnetic force against the gravitational pull and any other forces acting on an object by using permanent or electromagnets[9].

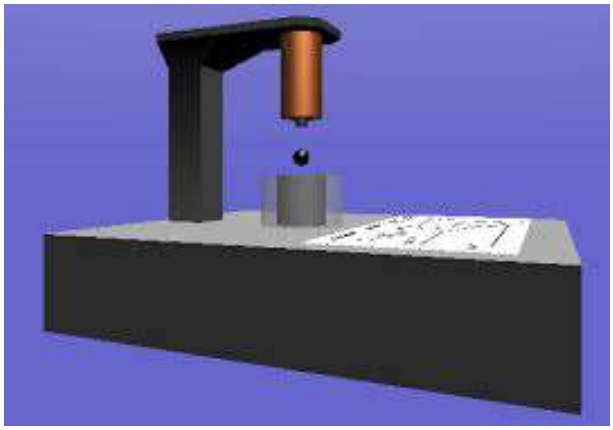


Figure 1: Magnetic Levitation

1.2 History

[8]. MagLev concept started in the early 20^{th} century and still undergoes development. The main and most popular application of MagLev is in transportation systems (Magnetic Levitated Trains).

In 1904, Robert Goddard wrote a paper in which he proposed a frictionless travelling system. Based on his theory, he claimed that the train would travel at super fast. If a Train body is raised from rails by using an electromagnetic repulsion system.

In 1910, Emile Bachelet applied for a patent on a rail car. He levitated a rail car by using

electromagnets which were powered by using alternating current and solenoid, although he was unable to achieve his dreams due to more power required for conventional magnets.

In 1969, two American scientists Gordon Damby and James Powell, who completed their work for the design of a magnetically levitated train and granted the first patent for this design work related to the MagLev train.

In 1970, Germany, Japan and the United state Federal Railway Administration started investing money in research of MagLev technology.

1.2.1 German Development

[8] In 1969 the German government sponsored "transrapid01", the first version of a full-scale model of MagLev. The German company Munich's Krauss Maffei, build an improved version of Trans Rapid.

In 1979 in Hamburg Germany, at the International Transportation Fair first large-scale Trans Rapid was demonstrated. The test track was built to test the model in real-time conditions. The way was built from 1979 to 1987 19 miles in Northern German.

In 1998, a new joint company was formed called Trans Rapid International. The joint company was formed under the joint company was called Trans Rapid International.

On 23 January 2001, after analyzing and statistical gathering. China builds a MagLev system between Shanghai and the airport with the help of German maglev technology.

1.2.2 Japenese Development

[8] Japan is known for its innovation and technology in different fields like automation and transportation etc. They have the world's best railway tracks. Their highest bullet train is at 350 km/h to 450 km/h. Now their target is to increase the MagLev speed upto the 650 km/h.

In 1970, the Japanese national railway(JNR) conducted research and development in MagLev. The prototype of MagLev named ML-500 tested on "The Miyazaki" test track. The MagLev train ran at a speed of 517 km/h on a 7 km long track. The track was built in a U-shaped to analyse the real-world track curves. This test proved a great future potential in MagLev transportation.

In 1987, JNR got privatized. The "Yamanashi" track was made in Tamanshi on the west side of Tokyo of 100 km(approximately) long. For the commercialization of the MagLev train, The "Yamanashi" test line was built which had 16 km of tunnels and an open section of 1.5 km long in the middle of the track where substations for power conversion and other facilities were provided.

Currently, the best-working MagLev train is built by German in China from Shanghai City to Shanghai airport. This train is working on an electromagnetic suspension system

on the other hand Japan is working on an electrodynamic one.

1.3 Hardware Setup

Our project is to levitate the ferromagnetic ball between two plates using MATLAB and FPGA. There are four steps in the project:

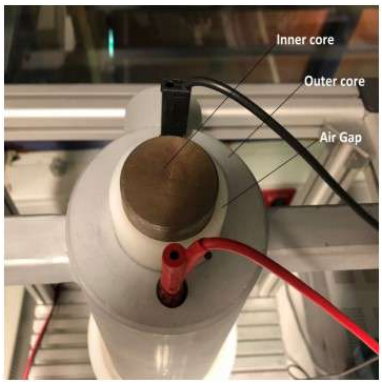
- System Modeling in MATLAB.
- Control Design in MATLAB.
- Real-time test.
- FPGA Software design.

To complete the project some hardware components or devices are required. The list of hardware components or devices used in the project are:

- Electromagnetic Coil.
- Distance Sensor.
- Current Amplifier.
- Interface Box.
- Power Supply.
- PCI Card.

1.3.1 Electromagnetic Coil

In the project used an electromagnetic coil to generate a magnetic force which helps to levitate the ball between two plates. There are two cylinder cores used inner and outer. The inner core is made up of an iron rod with copper winding which causes electromagnetic induction as shown in the figure below.



(a) Top View (b) Inside view

Figure 2: Electromagnetic Coil

1.3.2 Distance Sensor

A laser-based sensor provided by SICK (DT20HI) was used for measuring the position of the ferromagnetic ball with respect to the base frame. It has a red-pigmented laser beam with a measuring range of 600 mm and a resolution of 500 µm. This sensor is connected to the interface board. SICK (DT20HI) is used in projects because of its advantages for example reliable, precise measurement, independent of surface, fast commissioning via button, remote or numerical teach and tough metal housing permits operation in harsh environments.[6] The distance sensor in the project is used to find the location of the ferromagnetic ball between two plates.

Figure 3: Distance Senosr(SICK DT20HI)

1.3.3 VIO Amplifier

The ball levitated by controlling the voltage and current according to the positions of a ball between two plates, a VIO STARTER amplifier from ELMO motion control is used. This violin 15/200 regulates the continuous current of 15A and a maximum operating voltage of 200V.[21]

Figure 4: VIO Amplifier

In the project M+ and M-, Cref+ and Cref-, EN+ and EN- terminals are used. EN+ and EN- use to control the VIO converter. M+ and M- are used as voltage readers while Cref+ and Cref- are used to read the current going into a coil. The different combination of LEDs describes the different functions/modes of the vio current amplifier as shown in the table given below.[21]

Table 2: Modes of VIO Amplifier

Function	Latch Option	LD1 AOK Yellow	LD2 SO1 Red	LD3 SO2 Red	LD4 SO3 Red
Amplifier	N/A	ON	OFF	OFF	OFF
External disable	No	ON	ON	OFF	ON
Current Limit	No	ON	OFF	OFF	ON
Short	Yes	OFF	ON	OFF	ON
Over Temperature	Yes	OFF	OFF	ON	ON
Internal Supply Protection	No	OFF	ON	ON	OFF
Under Voltage	No	OFF	ON	OFF	OFF
Over Voltage	No	OFF	OFF	ON	OFF
Power Up Reset	No	OFF	OFF	OFF	OFF

1.3.4 Interface Board

This interface board is used between the DAQ (computer) and the Vio starter current amplifier. It helps take the analogue values from the distance sensor and feed them to the DAQ. It acts more like a junction box and can be used to process data before sending it to the DAQ card.

Figure 5: Interface Board

1.3.5 PCI Card

A PCI card is a kind of network adapter with a PCI interface. Recent motherboards of PC have PCI slots. PCI cards for communication instead of buses have point-to-point communication via switches to control the flow of data. When mounted the PCI card in a slot there is a link formed between them, which enables point-to-point communication. There are more than two lanes for communication. One lane is for transmitting the data and a second lane is for receiving the data. PCIcard is used because of its advantages:12-bit A/D conversion, D/A conversion, digital input, digital output, and counter/timer.[13]

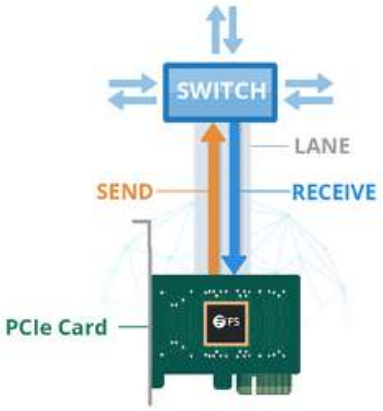


Figure 6: communication PCI

This data acquisition card is mounted within the CPU for the communication of MATLAB Simulink and The test stand. The main purpose of the card is to receive the analogue values from the interface board, Process it and send equivalent digital values to MATLAB. Its appropriate usage in the development of Lookup tables that are used in the project.

Figure 7: PCI Card

2 Model Based Design of Control System

A hollow ferromagnetic ball of mass m rand radius r is kept below a electromagnetic coil by a air gap distance d. The displacement of the ball is denoted by a variable x. From the physical structure of the plant, there exists a small air gap between coil and core but which can be considered negligible . From figure 8 it can be seen clearly that the airgap can be neglected.

From the controlling perspective of the system, The system is a Single Input Single Out (SISO) system where current flowing through the coil acts as a input and displacement of the ball in vertical direction is a output.

When a current i is applied to the coil, as per Faraday's laws of Electromagnetism a magnetic field is induces in the coil which generates a electromagnetic force of attraction between ferromagnetic object (ball) and coil. On other hand the gravitational pull on the ball counters the electromagnetic force, resulting in magnetic levitation. By controlling the current flowing through the coil desired position of the ball can be achieved. Please refer the Figure 8 for visualisation of the system.

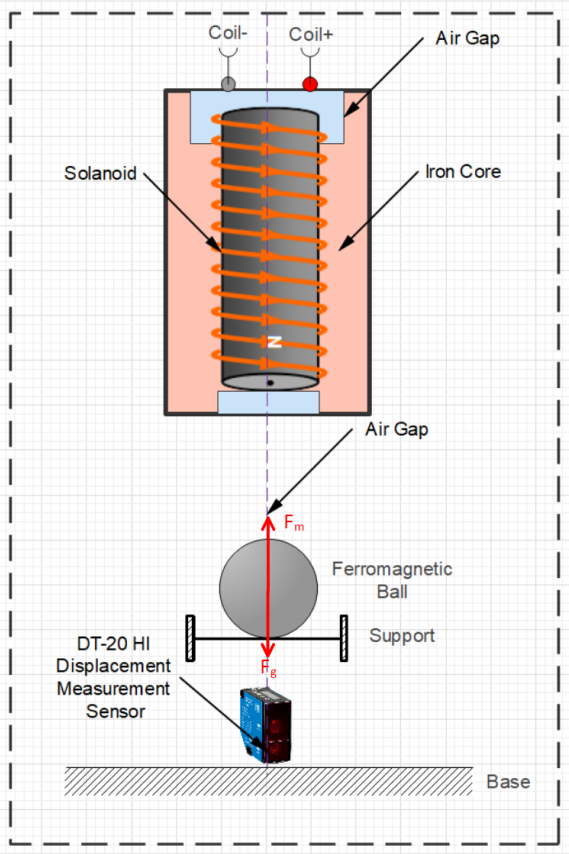


Figure 8: Structural Representation of Magnetic Levitation System

Before starting the modeling of the system few parameters should be taken into consideration.

Symbol	Description of Parameter	Value	Unit
m	Mass of the Ball	0.092	Kg
g	Acceleration Constant	9.8066	ms^{-1}
r	Radius of the Ball	0.06	m
d	Airgap Distance between Ball and Coil	0.052	ms
μ	Permeability of Air	$4\pi 10^{-7}$	Hm^{-1}
K	Coil Constant	0.1008	$Hm - Turns^2$
i_0	Equilibrium Current	0.11	A
x_0	Equilibrium Distance	0.026	m

Table 3: System Parameters

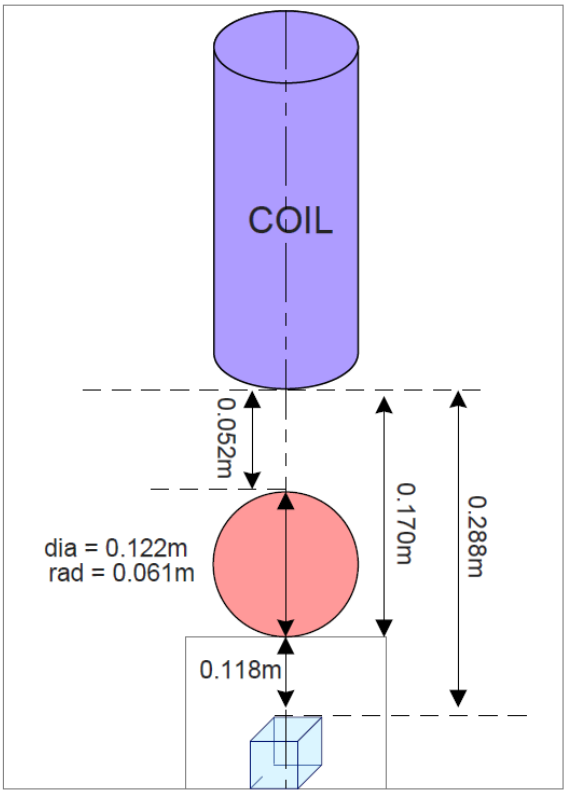


Figure 9: Dimensions and Physical Measurement of Magnetic Levitation Plant

2.1 Mathematical Model of Magnetic Levitation Plant Lagrangian Approach

Dynamic systems where alternate energies are present and the motion of an object is a result of the flow of energies from one form to another could be simplified with the help of Lagrange's equation/function. The "Lagrangian" describes the state of motion of particles through Kinetic and Potential Energy.

In the Magnetic Levitation model, the magnetic energy induced by the current flow could be added into Lagrangian in accordance with the Kinetic energy since it's an effect of electron motion through the conductor. (If a circuit has components that could store energy in the form of static charges such as capacitors they would be considered equivalent to Potential Energy) So, the equivalent Lagrangian would be [3]

$$L = T - V + W_m \tag{1}$$

T: Total Kinetic Energy in a circuit
V: Total Potential Energy in a circuit
W_m : Total Magnetic Energy in a circuit

2.1.1 Calculation of the Magnetic Energy

From Faraday's first law, It is known that a current-carrying conductor produces a magnetic field. As per the Biot-Savart rule, For a current-carrying conductor (I) will create a magnetic field of density (B) at a point (P) away from a distance (d) from the conductor is given by [3] [20]

$$B = \frac{\mu_o I d L sin\theta}{4\pi x^2} \tag{2}$$

μ_o: permeability of free space $4\pi \times 10^{-7}(Wbm^{-1}A^{-1})$
L: Length of the straight conductor (m)

The above equation could be transformed for Solenoid as shown in the below figure

$$B = \frac{\mu_o NI}{2d} \tag{3}$$

In this case distance d represents the position of the ferromagnetic ball, distance could vary from the range of $d - x$ i.e top to d i.e bottom. In the Idle situation, $x = d$ is the top position.

$$B = \frac{\mu_o NI}{2(d-x)}$$

Total flux linkage ψ is given by $\psi = N\phi$ but $\phi = BA$, So $\psi = NBA$.

$$\psi = \frac{\mu_o AN^2}{2(d-x)} I \tag{4}$$

ψ : Total flux linkage (Weber)
N : Number of turns per meter($turns * meter^-1$)
A: Cross sectional area of the coil(m^2)

Hence the magnetic energy becomes a function of displacement x and Current through the coil I.

So, Total magnetic energy W_m will be,

$$W_m = \int_{I=0}^{i} \psi(I, x) dI \tag{5}$$

$$W_m = \frac{\mu_o AN^2}{4(d-x)} i^2$$

$$W_m = \frac{ki^2}{2(d-x)} (Joules) \tag{6}$$

where $k = \frac{\mu_o AN^2}{2} (Henry - Meter - turns^2)$ is a coil constant.

2.1.2 Calculation of Kinetic Energy (T)

As per Newton's equation a kinetic energy possessed by a mass body $M(Kg)$, travelling at a speed of $V(m/s^2)$ is

$$T = \frac{1}{2} mv^2 (joules) \tag{7}$$

2.1.3 Calculation of the Potential Energy (V)

As per the work energy theorem the potential energy stored in a object of mass m at height x is given by

$$V = mgx \tag{8}$$

Putting into equation $L = T - V + W_M$,

$$L = \frac{1}{2}mv^2 + mgx + \frac{ki^2}{2(d-x)} \tag{9}$$

Taking Lagrangian of above equation

$$\frac{d}{dt}\left(\frac{\partial L}{\partial v}\right) - \frac{\partial L}{\partial x} = 0 \tag{10}$$

With the help of Lagrangian, we could say,

$$\frac{d}{dt}(mv) - \frac{k}{2}\frac{i^2}{(d-x)^2} + mg = 0$$

$$m\frac{dv}{dt} - \frac{k}{2}\frac{i^2}{(d-x)^2} + mg = 0$$

In simplified form

$$\boxed{\frac{dv}{dt} = \frac{k}{2m}\frac{i^2}{(d-x)^2} - g} \tag{11}$$

The system equation of motion is given by

$$\boxed{\frac{dx}{dt} = v} \tag{12}$$

This equations are used for modeling the Non-Linear System in MATLAB

2.2 Development and Simulation of Non-Linear Model

A MATLAB Simulink is used to develop a non-linear model in accordance with the nonlinear system equations.

$$\boxed{\frac{dv}{dt} = \frac{k}{2m}\frac{i^2}{(d-x)^2} - g}$$

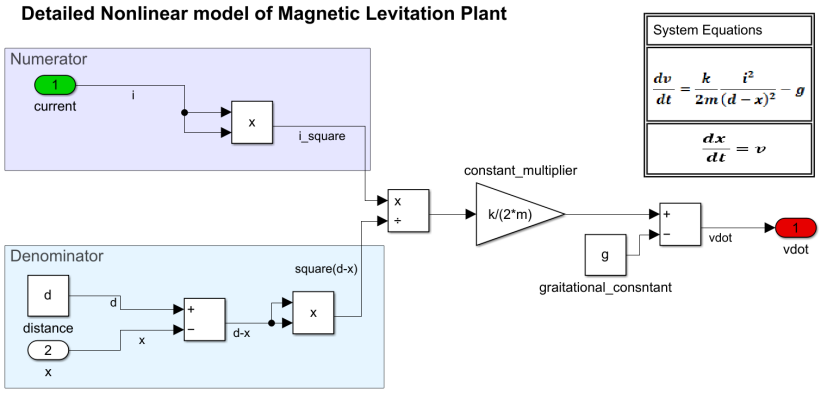


Figure 10: Detailed Nonlinear realisation of Plant

The above system is encapsulated in a subsystem of a plant. This plant is subjected to simple tests to verify the correctness of model.

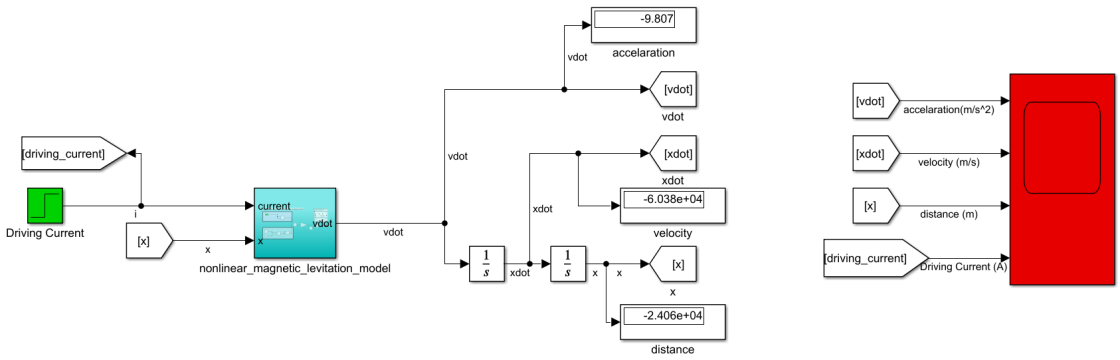


Figure 11: Simulation of Non-linear model of plant

The model is subjected to initial conditions of $x_{(init)}$ (initial displacement) = 0 m. which reflects the resting position of the ball.

2.2.1 Case-1 : Free-fall Condition

When a plant is subjected to the zero current input the ball behaves to fall down under a force of gravity. Since the first term of the Non-linear equations becomes zero the only acceleration experienced by the ball is $\frac{dv}{dt} = -g$ (can be observed in the first graph Figure4)

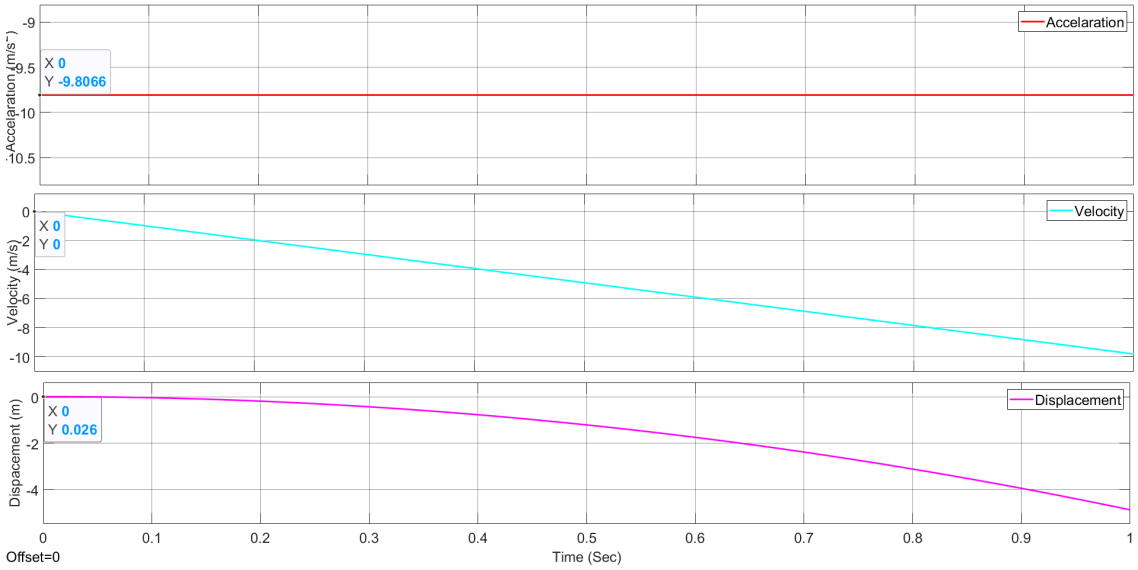


Figure 12: Downward fall of the under a force of Gravity

2.2.2 Case-2 : Step Response

When the plant is subjected to a step response the system shows insatiability.

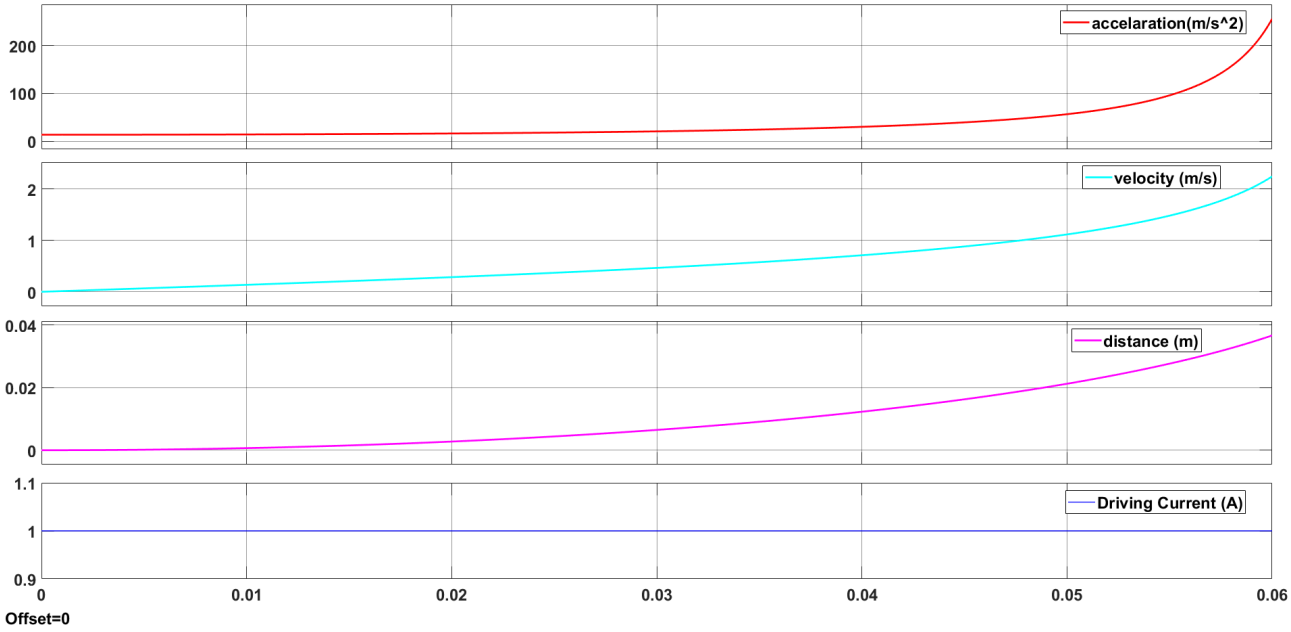


Figure 13: Step response of the system for unit step input (current)

Moving forward, it is advisable to transform the system into a linear system and analyze its poles. By employing modern pole placement techniques, control system engineers can manipulate the poles to achieve the desired system response. This approach allows for precise control over the system's behavior and facilitates the design of an best suited control strategy.

2.3 Determination of Equilibrium position

Theoretically any point on a operating rage could be a Equilibrium point, since at every point with a appropriate current the forces on the ball from magnetic field and force from gravitational pull will cancel out. Since we know this fact that the relation between acceleration and current applied is nonlinear over a broad range of distance, It is wise to choose a point in a operating range which will give a near linear behaviour of acceleration with respect to current. In such case choosing a center point of operating region gives a better coverage of a linear range. Hence for the experiment a equilibrium point of $x = x_0 = \frac{d}{2}$ is considered. The manipulating variables are i and x At equilibrium condition i.e. at $i = i_0$ and $x = x_0$

$$i_0 = (d - x_0)\sqrt{\frac{2mg}{k}}\left(d - \frac{d}{2}\right)\sqrt{\frac{2mg}{k}} = \left(\frac{d}{2}\right)\sqrt{\frac{2mg}{k}} \tag{13}$$

$$x_0 = \frac{d}{2} \tag{14}$$

2.4 Linearisation of the system Equations

Linearization of the System around a decided operating point is the method by which a system with a nonlinear Multi-variable equations is converted into approximated linear system for calculations simplicity. Benefits of Linearizing the system are Realize the model in State Space domain. Designing Linear controller. Analyze and Compare system responses near different operating point. Here the manipulating variables are i and x, At equilibrium condition i.e. at $i = i_0$ and $x = \frac{d}{2}$,

Linearising at operating point $i = i_0$ and $x = x_0 = \frac{d}{2}$,
so at any operating point $\boldsymbol{x = x_0 + \triangle x}$ and $\boldsymbol{i = i_0 + \triangle i}$

$$f(i,x) \approx f(i_0,x_0) + \frac{\partial f(i,x)}{\partial i}_{|(i_0,x_0)}(i-i_0) + \frac{\partial f(i,x)}{\partial x}_{|(i_0,x_0)}(x-x_0) \tag{15}$$

Step 1 : finding $f(i_0, x_0)$

$$f(i_0,x_0) = \frac{k}{2m}\left(\frac{i_0^2}{(d-x_0)^2}\right) - g$$

Substituting i_0 and x_0 we get,

$$f(i_0,x_0) = \frac{k}{2m}\left(\frac{d^2}{4}\frac{2mg}{k}\frac{1}{\left(d-\frac{d}{2}\right)^2}\right) - g$$

$$f(i_0,x_0) = g - g$$

$$f(i_0,x_0) = 0 \tag{16}$$

Step 2 : finding $\frac{\partial f(i,x)}{\partial i}_{|(i_0,x_0)}(i-i_0)$

$$\Rightarrow \frac{k}{2m}\frac{2i}{(d-x)^2}_{|(i_0,x_0)}$$

$$\frac{\partial f(i,x)}{\partial i}_{|(i_0,x_0)}(i-i_0) = \frac{1}{d}\sqrt{\frac{8kg}{m}(i)} - 2g \tag{17}$$

Step 3 : Finding $\frac{\partial f(i,x)}{\partial x}_{|(i_0,x_0)}(x-x_0)$

$$\Rightarrow \frac{k}{2m}i^2\frac{-2}{(d-x)^3}_{|(i_0,x_0)}(x-x_0)(-1)$$

$$\Rightarrow \frac{k}{2m}i^2\frac{-2}{(d-x)^3}_{|(i_0,x_0)}(x-x_0)(-1) \tag{18}$$

By adding all together we get

$$\boxed{f(\triangle i,\triangle x) = \frac{d\triangle v}{dt} \approx \frac{1}{d}\sqrt{\frac{8kg}{m}}(\triangle i) + \frac{4g}{d}(\triangle x)} \tag{19}$$

Similarly for equation 10 ,At the equilibrium condition $v_0 = 0$, Equation (11) and Equation (12) are used for Non-Linear model development. Similarly linearising the second equation we get

$$f(x) \approx f(v_0) + \frac{\partial f(v)}{\partial v}_{|v_0} (v - v_0) \tag{20}$$

At equilibrium condition $v = v_0 = 0$

$$f(x) = 0 + 1(v - 0)$$

Hence, The second equation is a linear equation before too.

$$f(x) \approx v$$

$$\boxed{\frac{d\triangle x}{dt} = \triangle v} \tag{21}$$

Equation (19) and (21) are used for development if Linear Model in Matlab

3 State Space Modeling of the Plant

3.1 State Space Representation of the system

The state space representation is a mathematical modeling method used in control system to describe a dynamic behaviour of the system. A state space representation can be best describe by a set of First order equations which determines rate of change of state variables.

For better visualisation the state space system is represented in the form of matrices also known as State Space form. with a state space form control system engineer can develop a control technique for a linear system. Also it could be best to check the controlability and observability of the system. A basic state space realisation of any system with n Internal States, p Inputs and q outputs can be gives by this set of equations.

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{22}$$

Where,

$\boldsymbol{A}$ is $\boldsymbol{n} \times \boldsymbol{n}$ System Matrix , Here $\boldsymbol{A_{2\times 2}}$
$\boldsymbol{B}$ is $\boldsymbol{n} \times \boldsymbol{p}$ Input Matrix , Here $\boldsymbol{B_{2\times 1}}$
$\boldsymbol{C}$ is $\boldsymbol{q} \times \boldsymbol{n}$ Output Matrix , Here $\boldsymbol{C_{1\times 2}}$
$\boldsymbol{D}$ is $\boldsymbol{q} \times \boldsymbol{p}$ Feed-Forward Matrix , Here $\boldsymbol{D_{1\times 1}}$

A example model of linear spate space representation of a magnetic levitation system is given below.

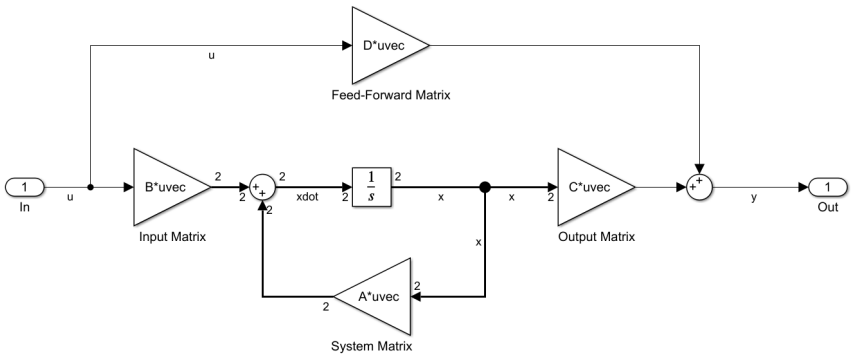


Figure 14: Generic State Space representation of the Magnetic Levitation system

For a linearised Magnetic levitation system the set of linear equations can be represented in the matrix from as given below.

$$f(\triangle i, \triangle x) = \frac{d\triangle v}{dt} \approx \frac{1}{d}\sqrt{\frac{8kg}{m}}(\triangle i) + \frac{4g}{d}(\triangle x)$$

$$\frac{d\triangle x}{dt} = \triangle v$$

This set of equations can be expressed in the matrix form as

$$\begin{bmatrix} \dot{\triangle v} \\ \dot{\triangle x} \end{bmatrix} = \begin{bmatrix} 0 & 4\frac{g}{d} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \triangle v \\ \triangle x \end{bmatrix} + \begin{bmatrix} \frac{1}{d}\sqrt{\frac{8kg}{m}} \\ 0 \end{bmatrix} \triangle i \tag{23}$$

$$y(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \triangle v \\ \triangle x \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \triangle i \tag{24}$$

Here

$$A = \begin{bmatrix} 0 & 4\frac{g}{d} \\ 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} \frac{1}{d}\sqrt{\frac{8kg}{m}} \\ 0 \end{bmatrix} \tag{25}$$

The set of Matrices are used to realise the system in state space format in MATLAB. A script file is created with all the necessary variables and matrices to simulate the plant. For better understanding of the system both states of the system are observed.

```
%% Defining Constants %%

Ts = 0.01; %% sample time

g=9.8066; %% gravitational accelaration (m/s^2)

d=0.0520; %% distance of the ball from the top (Meter)

m= 0.0920; %% weight of the ball (Kilogram)

i0=0.11;  %%offset current (Amperes) % prerecorded at 0.024m

k= (2*m*g*(d-0.024)^2)/(i0)^2;  %% Coil Constant (Henry-Meter-turns^2)

x0 = d/2;

xdot0 = 0;

A=[0 (4*g/d);1 0];            %% System Matrix
B= [(1/d)*sqrt(8*g*k/m);0];   %% Input Matrix
C=  eye(2);                   %%
ct = [0 1];                   %% Output Matrix
D=[0];                        %% Feedforward Matrix
```

Similarly a simple model of a Linear Plant is developed in MATLAB Simulink with the help of "Linear State space" block. The initial conditions of the "Linear State Space" block are set to zero. The model is verified with a simple test cases as below

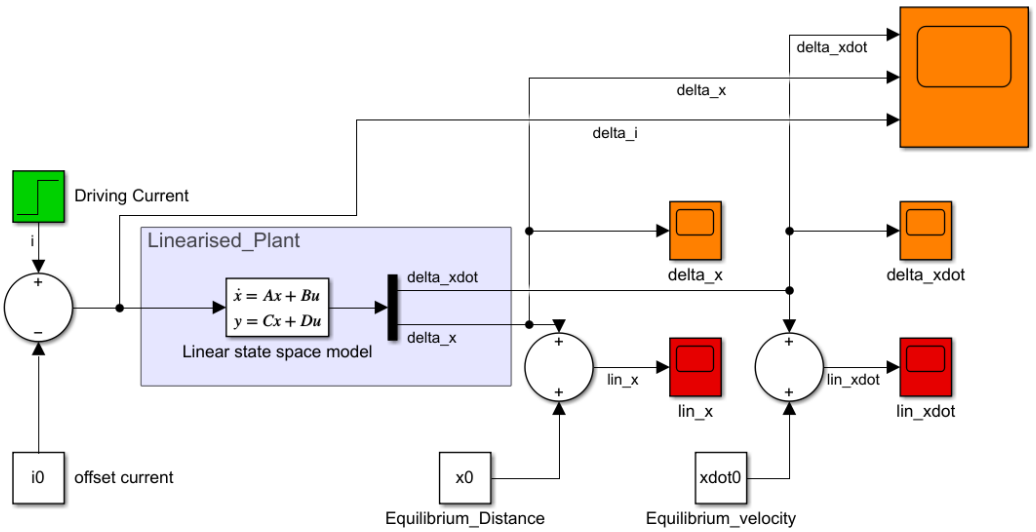


Figure 15: Simulation of Linear State Space System

3.1.1 Case-1 : Zero input current

When a Linear model is subjected to $\triangle i = 0$ Ampere the ball should be stable at $\triangle x = 0$ meter.

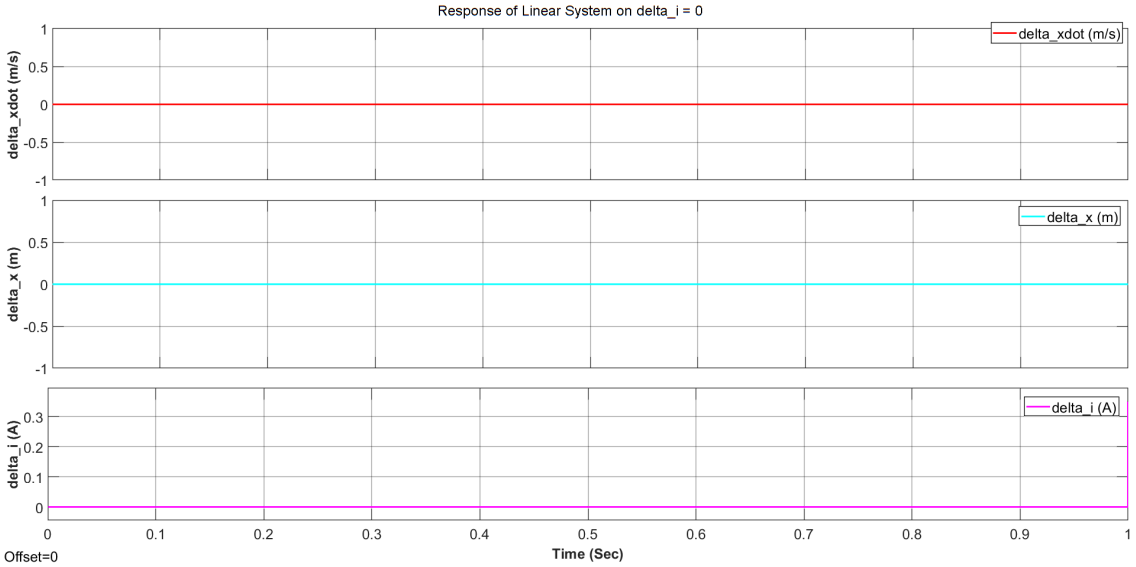


Figure 16: Response of Linear System at $\triangle i = 0$ Ampere

3.1.2 Case-2 : Open loop Unit Step Response of linear system

Upon application of positive current it is expected that the ball will move in upward direction but will not be able to achieve the stability since the equilibrium point is the only point of critical stability.

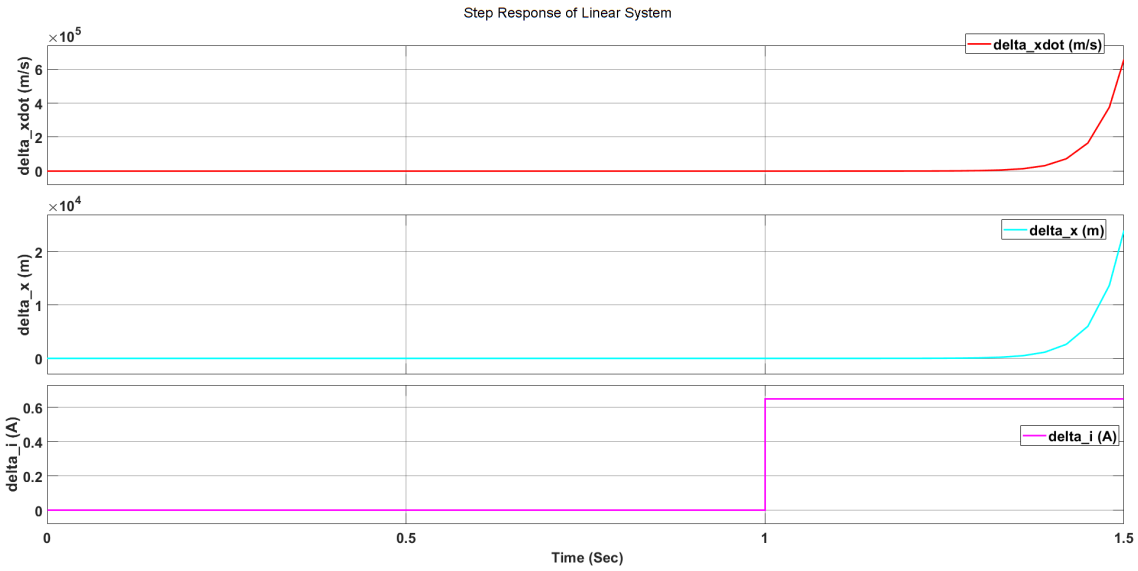


Figure 17: Step Response of Linear System

Before implementing a control technique on the plant it is important to check the current open-loop poles of the system as well as bode plot of the system. To do this the state-space system is converted into a Transfer function format.

```
[num,den]=ss2tf(A,B,ct,D);
systf=tf(num,den);
h=pzplot(systf);
[Gm,Pm,Wcg,Wcp] = margin(systf); %%system is unstable
grid on;
```

The open loop transfer function of the SISO system (between Displacement vs Current)is

$$\frac{Y(s)}{U(s)} = \frac{60.35}{s^2 - 7.105e - 15s - 754.4} \tag{26}$$

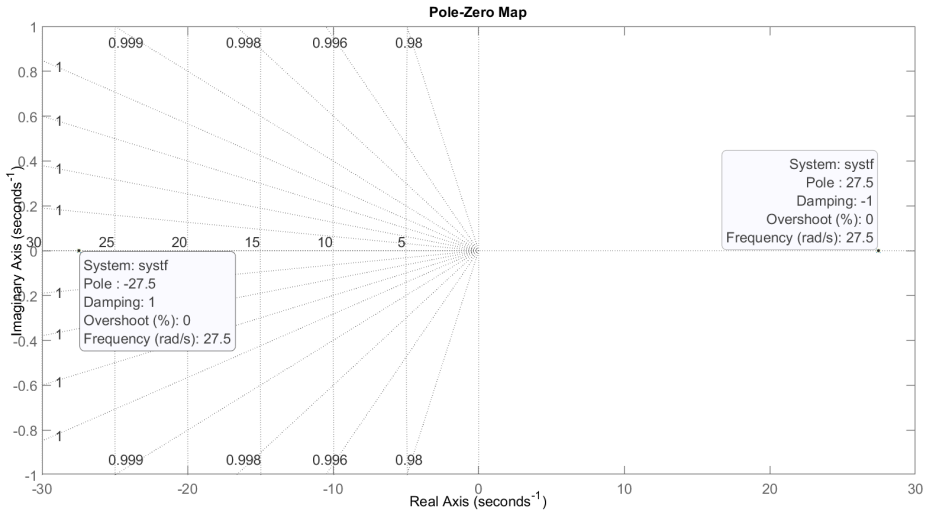


Figure 18: Open loop Pole-Zero plot of the Plant

As we can see from the Pole-*Zero plot, One of the pole of a system lies on the Right hand of S plane. This implies that openloop System is unstable. This also can be seen from the Bode-plot below. From the Bode Plot it can be observed that the Gain Margin is Negative as well as the Phase Margin is stuck to -180. This clearly states the unstable behaviour of open-loop system.

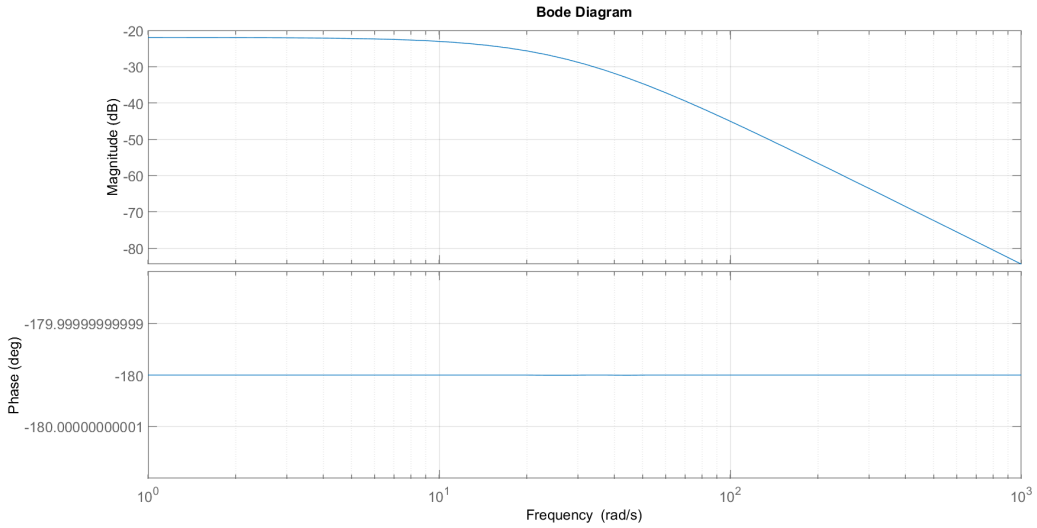


Figure 19: Open loop Bode Plot

In the next chapter we will go through the process of designing a close loop controller with Pole Placement Method.

4 Design of Time Continuous Position Controller

4.1 Motivation of going Beyond PID controllers

PID control is widely used and has been successful in control engineering applications. But when dealing with a complex systems or a systems with Multiple inputs and Multiple outputs PID controllers are not the best choice. PID controllers gives a completely decentralised response when the system chosen has a output parameter whose value is dependent on another output parameter. In our case the output equations directly passes the systems internal states without any complexity, Hence just for a prototyping purpose a new approach of "Full state feedback controller" is utilised. However in the final implementation a simple PID controller technique is used.

when there are interactions between inputs and outputs, the closed-loop poles and system stability become dependent on the values of the interaction parameters. Necessary conditions for stability are derived, and it is shown that the closed-loop poles can vary and lead to instability as the interaction parameters change.

Furthermore the Gains of a State Feedback matrix can be scaled and substituted as a Gains of PID controller. This avoids the efforts of guessing the PID gains and speed up the development process.

4.2 Design of Time Continuous state feedback controller

4.2.1 Controlability of the system

To develop a full state feedback controller a system must be fully state controllable. A state variable model of a dynamic system is said to be completely state controllable if, given any states $x(t_0)$, $x(t_f)$ and any initial time t0, there exist a time $t_1 > t_0$ and an input $u(t)$ such that the system state $x(t)$ is transferred from the initial state x(t0) to the final state $x(t_1) = x(t_f)$. A condition of for a system to be a complete state controllable is the rank of state controllability matrix of dimension $n \times n$ should be $rank(L_C) = n$
In other words $det(L_C) \neq 0$.

Determining controllability of the system
The generalised form of Controlability Matrix is

$$L_c = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-2}B & A^{n-1}B \end{bmatrix} \tag{27}$$

for a second order LTI system, the controlability matrix is given by

$$L_c = \begin{bmatrix} B & AB \end{bmatrix} \tag{28}$$

After substituting values

$$L_c = \begin{bmatrix} 192.0173 & 0 \\ 0 & 192.0173 \end{bmatrix} \tag{29}$$

It can be clearly seen from the Matrix L_c that the $|L_c| \neq 0$ which implies that $rank(L_c) = 2$. Thus the system made up with matrices A and B is Fully State Controllable.
The MATLAB command `Lc=ctrb(A,B);` can be used to derive Controlability matrix.
The rank of a matrix can be verified by `q=rank(Lc);`.
If $q = n$ then the system is state controllable.

4.2.2 Pole Placement with Ackerman's Method Proportional Controller

Pole placement is a modern control technique where a linear state feedback controller K is chosen in such a way that $u = -Kx$ will satisfy the characteristic equation of a desired stable system with desired pole. This problem is also called as **Eigenvalue assignment** problem. The below is a general block diagram representation of full state feedback controller.

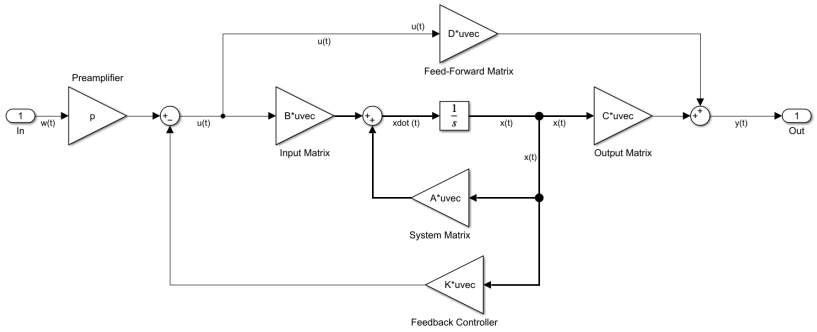


Figure 20: General Block Diagram of State Feedback Controller

The system is modeled with following setup

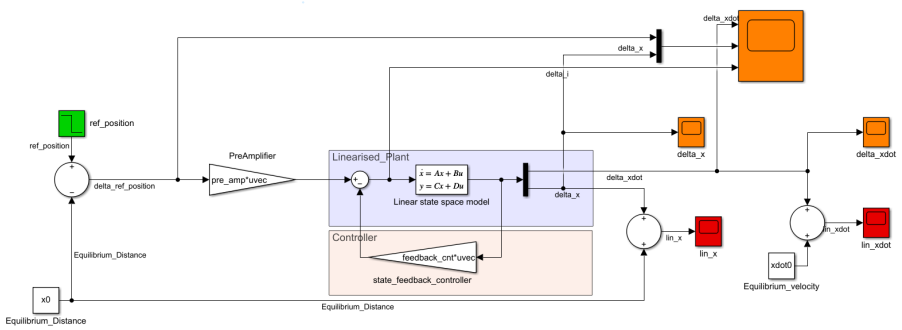


Figure 21: Simulation of State Space Feedback Controller

In the pole placement, System's pole are arbitrarily chosen at [-5, -5] to achieve best performance . From the ackerman's pole placement method the value of Feedback controller matrix K and Preamplifier matrix P is calculated as :

Step 1 : Find a Controlability Matrix

From the equation number XX the state controlability matrix is

$$controlability\_matrix = \begin{bmatrix} 192.0173 & 0 \\ 0 & 192.0173 \end{bmatrix} \tag{30}$$

Step 2 : Take Inverse of a Controlability Matrix

Inverse of a Controlability matrix is found by inverting in a MATLAB. `controlability_inv = (controlability_matrix)^-1;`

$$controlability\_inv = \begin{bmatrix} 0.0052 & 0 \\ 0 & 0.0052 \end{bmatrix} \tag{31}$$

Step 3 : Extract the Last Row of Controlability Matrix

This cna be easily done in matlab by selecting the last row of a matrix.
`last_row_controlability_inv=controlability_inv(end,:);`

$$last\_row\_controlability\_inv = \begin{bmatrix} 0 & 0.0052 \end{bmatrix} \tag{32}$$

Step 4 : Develop a Characteristic equation

A characteristic equation with desired poles is developed. `alpha = poly([pole_1,pole_2]); feedback_cnt = last_row_controlability_inv*( alpha(3)*eye(2)+ alpha(2)*A + alpha(1)*A*A );` This will create a feedback controller which will easily handle the inputs to the system. But when a final value theorem is applied to the output we get a stedy state error which can be formulated as .

Final Value Theorem :

$$\lim_{t\to\infty}(y(t)) = \lim_{s\to 0}(sY(s)) \tag{33}$$

But a step function in S-Domain is given by $W(s) = \frac{1}{s}$

$$\lim_{s\to 0}\left(sF(s)\frac{1}{s}\right) = \lim_{s\to 0}\left(c(sI-(A-bK))^{-1}b\right) \tag{34}$$

putting limits

$$\lim_{s\to 0}\left(sF(s)\frac{1}{s}\right) = \left(c(bK-A)^{-1}b\right) \tag{35}$$

as per final value theorem Y(s) = 1. Since a steady state response of the state feedback controller is giving an output of a constant $(c(bK-A)^{-1}b)$, A gain must be multiplied to input to scale the output back to unity (Unit Step).

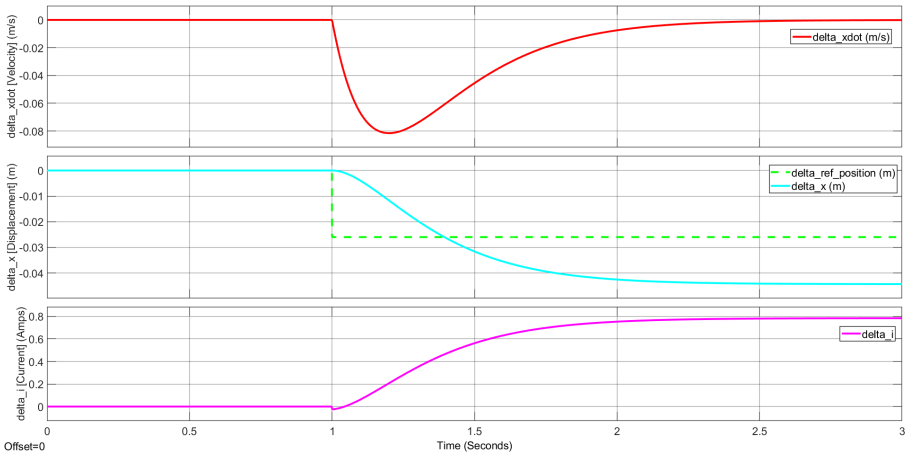


Figure 22: Step Response of State Feedback Controller W/O Preamplifier

Step 4 : Develop a Preamplifier gain

To scale the output to unity, A gain should be .

$$p = \frac{1}{(c(bK - A)^{-1}b)} \tag{36}$$

After Implementing the preamplifier the step response of the system is brought back to unity. `pre_amp = 1/(ct*((B*feedback_cnt-A)^-1)*B);`

$$pre\_amp = 0.1402 \tag{37}$$

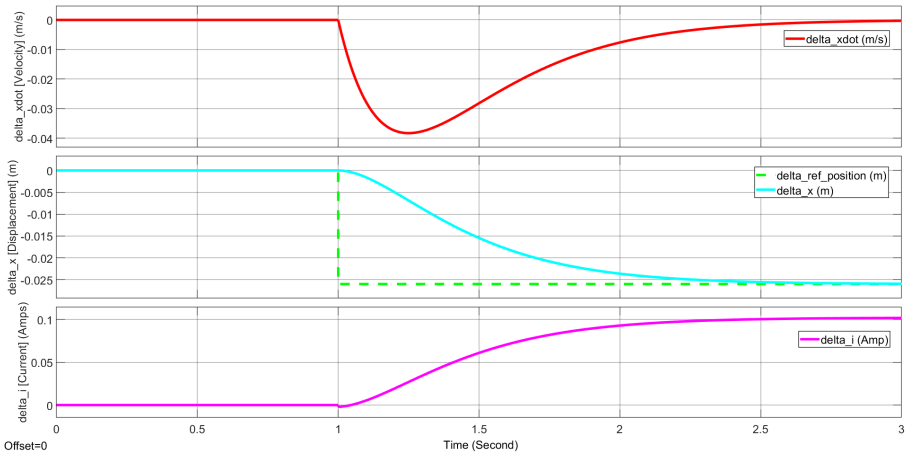


Figure 23: Step Response of State Feedback Controller

4.3 Determining the close loop stability of system

In order to ensure that the given close-loop system is stable, the closed loop system matrix $A - BK$ must have all its eigen value on the left half of the complex plane. Another way is to form a close-loop transfer function and find the close-loop poles of the system-

To check the system's new poles a pole zero plot is made which shows Open-loop and Close-loop Poles. The close-loop Transfer function obtained with state feedback K is given by

$$Closeloop\ Transfer\ Function\ =\ \frac{178.2}{s^2+10s+25} \tag{38}$$

```
%% Analysis
% State space Representation of Closeloop system
% Necessary for deriving the BodePlot and Unit Step Response
% Closeloop SS W/O Prescalar : Gain & Phase Margins
clsloop_ss = ss((A-B*feedback_cnt),B,C,D);
% Closeloop SS With Prescalar : step response
clsloopsys_ss=feedback(ss((A-B*feedback_cnt_s),B,C,D)*pre_amp,[1 1]);
% Transfer Function representation of Closeloop System
[num_cl,den_cl]=ss2tf((A-B*feedback_cnt),B,C,D);
clsloop_tf=tf(num_cl(2,:),den_cl);
% Closeloop TF with additional Feedback
clsloopsys_tf = feedback(clsloop_tf*pre_amp,[1]);

% Figures
figure('Name','Pole-Zero Plot of 1) Open loop system 2)Close loop feedback
    System','NumberTitle','off');
pzplot(plant_tf,'r',clsloop_tf,'c')
grid;

[Gm_cl,Pm_cl,Wcg_cl,Wcp_cl] = margin(clsloop_ss(2)); %%system is stable
figure('Name','Bode plot of Close Loop system','NumberTitle','off');
bode(clsloop_tf);
grid;

figure('Name','Step Response of Close Loop System','NumberTitle','off');
step(clsloopsys_ss);
grid on;
```

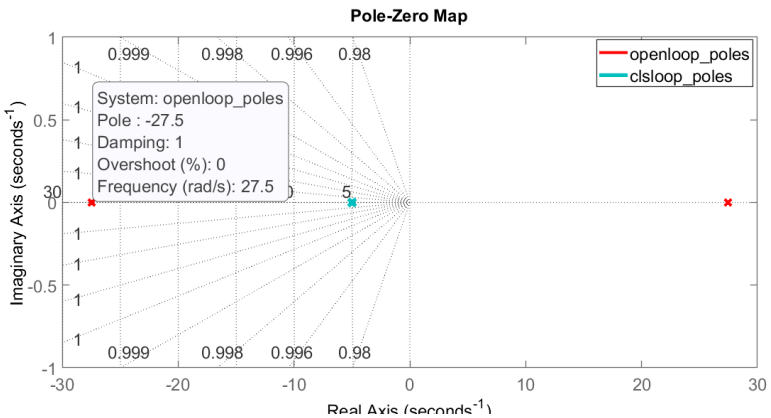


Figure 24: Pole Zero Plot of a Closeloop System

Another method to determine the stability of a close-loop system in frequency domain is

to perform a Bode Analysis of Close-loop system.

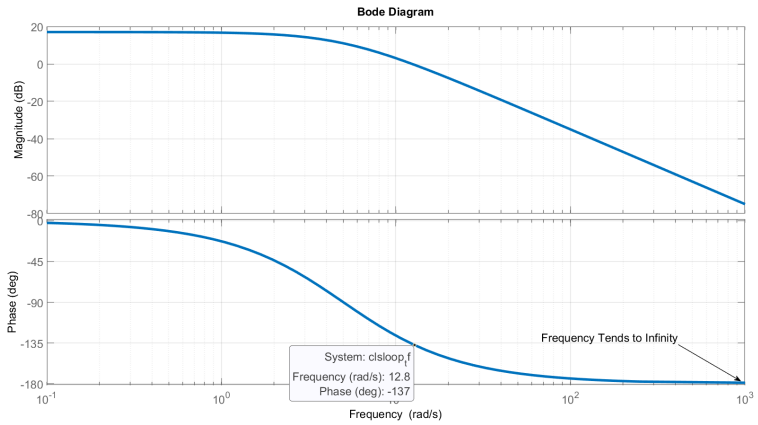


Figure 25: Bode Plot of a Close-loop System

$$\text{Gain Margin} = [43.98]\ degree \quad \text{Phase Margin} = [\infty] \tag{39}$$

From the Bode plot is is clear the Phase Margin of a System is 43.98 deg and the Gain Margin is tends to Infinity. This both signifies that the close-loop system with controller is a stable system.

4.4 Effect of Disturbance

To check the robustness of a controller a $+10\%d$ output disturbance is applied to the system output. In such case the a **additional feedback** must be used to compensate the disturbances at output. Due to a additional feedback the State Feedback Vector changes to new feedback Vector $k^{*T} = k^T - pc^T$. Where k^T is a old feedback vector, p is a preamplifier and matrix c is a output matrix. By following formula we formulated the new feedback vector.

```
1    feedback_cnt_s = feedback_cnt(1:2)-(pre_amp*ct);
```

The close-loop Transfer function with additional feedback is given as

$$Closeloop\ Transfer\ function\ system\ = \frac{25}{s^2 + 10s + 50} \tag{40}$$

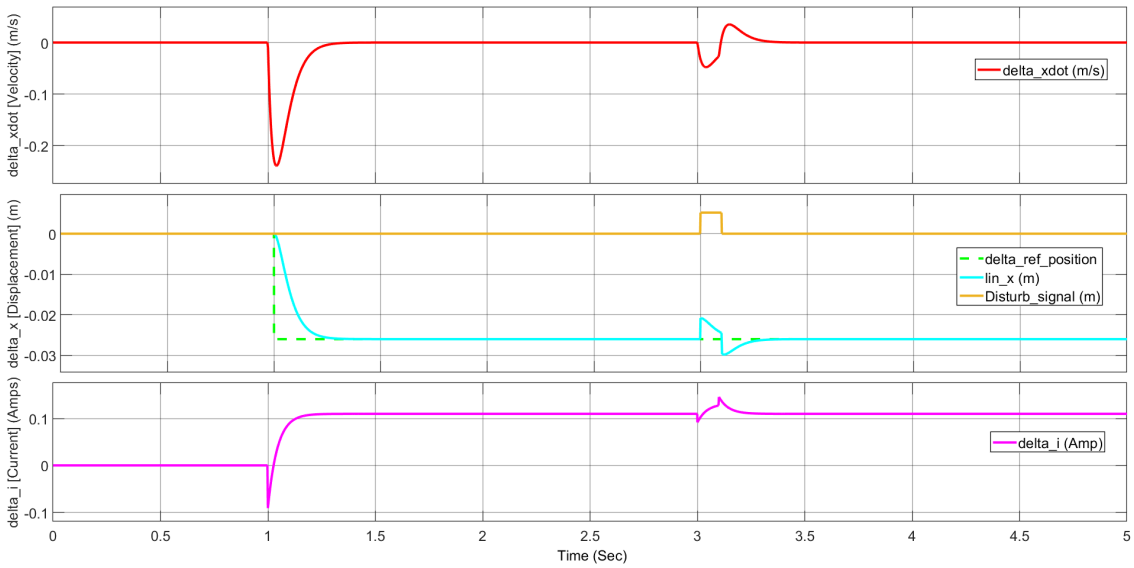


Figure 26: Step response with 10% (of step) disturbance

4.5 Design of Discrete State feedback controller

4.5.1 Dilemma of Discretisation

To discretize a system, there are two main approaches that can be considered. The first approach involves discretizing the modeled plant and developing a discrete feedback controller based on the newly discretized system matrices. The second approach involves modeling the system in continuous time and developing a continuous-time controller. This continuous-time controller is then discretized to obtain an equivalent discrete controller.

Both approaches yield similar results and are valid methods for achieving system discretization. However, in this report, both options were tested, and it was found that the first approach is preferred due to its simplicity.

4.5.2 Discretisation of Plant and Development of discrete controller

```
%Time continuous statespace representation of the Plant
plant_ss =ss(A,B,C,D,Ts);
%Time continuous transferfunction representation of the Plant
[num,den]=ss2tf(A,B,C,D);
plant_tf=tf(num(2,:),den);
%Time discrete statespace representation of the Plant
plant_ss_disc=c2d(ss(A,B,C,D),Ts);
[AD,BD,CD,DD]=ssdata(plant_ss_disc);
```

The above code converts the time continuous plant into time discrete. as well as forms two representation of the plant both in State space and Transfer function.

```
%% Discrete_controller
% controlability matrix for discrete system
controlability_matrix_disc = [BD AD*BD];
%Inverse of Discrete controlability matrix
controlability_inv_disc = inv(controlability_matrix_disc);
% extractig last row of inverse controlability matrix
last_row_controlability_inv_disc = controlability_inv_disc(end,:);
%Transfer function for desired poles
tf_p_cont=tf([1],poly([pole_1 pole_2]));
% Converting transfer function to discrete
tf_p_disc=c2d(tf_p_cont,Ts,'tustin');
[zeros_d,poles_d]=tfdata(tf_p_disc,'v');
% Ackermans formula for Feedback controller
feedback_cnt_disc =
    last_row_controlability_inv_disc*(poles_d(3)*eye(2)+poles_d(2)*AD+poles_d(1)*
    AD^2);
```

The steps followed are exactly same as designing a Time continuous state feedback controller. Where the value of discrete feedback vector is derived as

$$feedback\_cnt\_disc = \begin{bmatrix} 0.2535 & 6.9830 \end{bmatrix} \tag{41}$$

It is always good idea to use a Time discrete fixed step solved when simulating a discrete system.

The unscaled step response of a Discrete State feedback controller is as shown below.

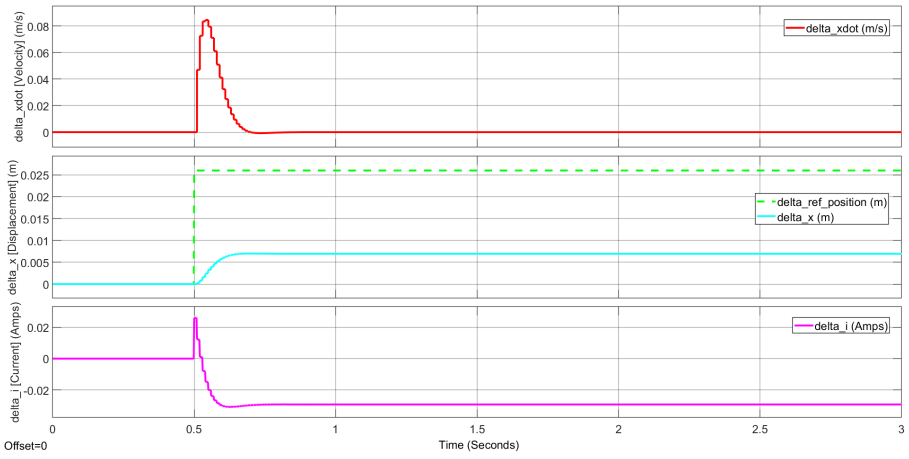


Figure 27: Step Response of Discrete State Feedback Controller W/O Preamplifier

with the application of preamplifier

```
    pre_amp_disc =1/(ct*((BD*feedback_cnt_disc-AD)^-1)*BD);
```

$$pre\_amp\_disc = 0.1402 \tag{42}$$

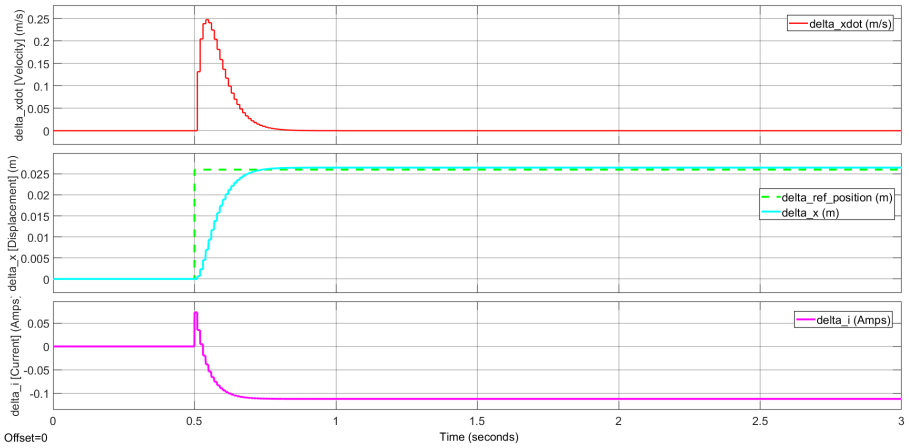


Figure 28: Step Response of Discrete State Feedback Controller with Preamplifier

There exist a small steady state error because of PD nature of controller. This Steady state error can be minimised with the methods explained in subchapeter XX.

4.6 Stability Analysis of close loop Discrete system

The stability of state feedback controller an be analysed from the Pole zero plot. For a discrete system the Poles of the system must be within the unit circle. The Poles of Open-loop system shows outside the unit circle which signifies the unstable nature of Plant whereas the Close-loop Poles are within the unit circle.

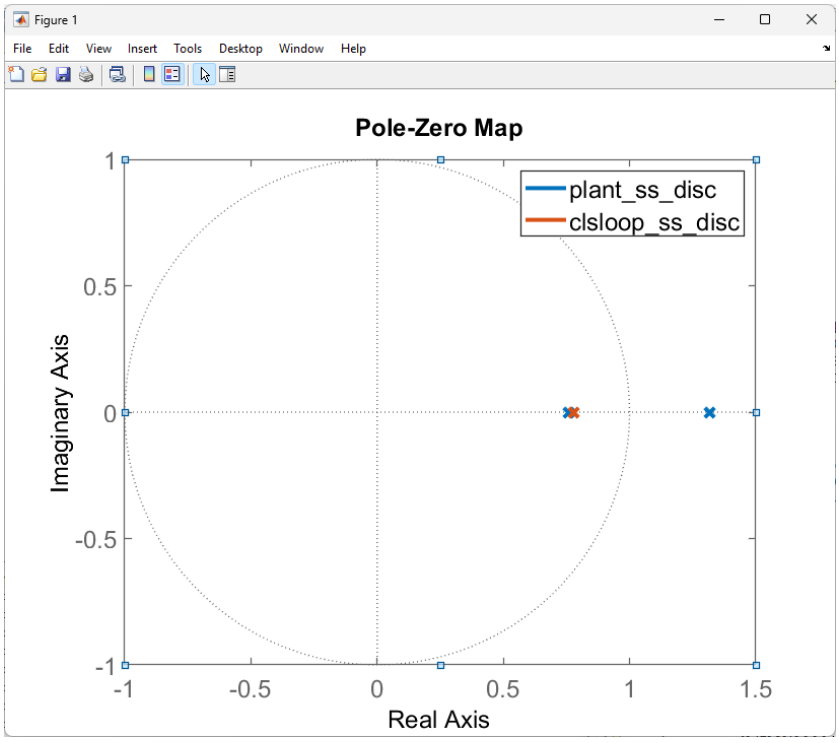


Figure 29: Pole and Zero Z plane

Another strong indicator of good stability is bode analysis where gain and phase margin of the system are checked. In order to be a close-loop system to be stable the Gain and Phase Margins must be positive. More the margins more variation in the gain and phase are acceptable without making system unstable.

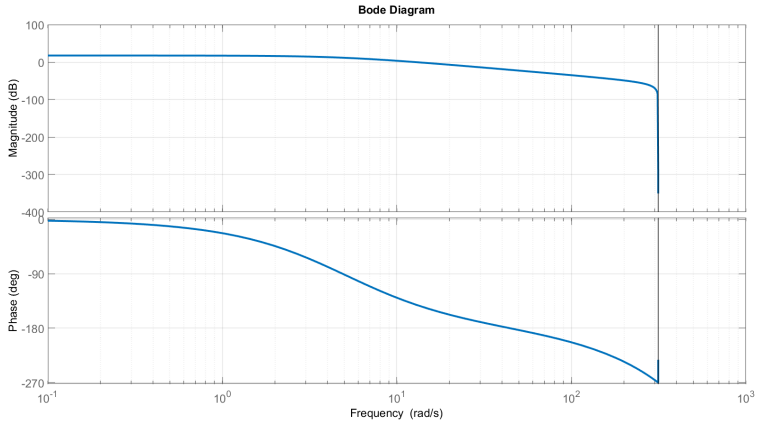


Figure 30: Bode Plot Closeloop system with discrete controller

The Gain margin and phase margin are

```
[Gm_cld,Pm_cld,Wcg_cld,Wcp_cld] = margin(clsloop_ss_disc(2))
```

$$\text{Gain Margin } (Gm\_cld) = 10.6097 \quad \text{Phase Margin } (Pm\_cld) = 39.0047 degree \quad (43)$$

The both margins implies that the close-loop system with discrete controller is stable.

4.7 Methods to eliminate the steady state error

Option 1 : High Proportional Gain

High Proportional gain means faster converge rate of $e(t) \to 0$. But on other hand High Proportional gain is impractical to have which directly means that the current source can give current of infinity ampere.
High Proportional gain makes the controller highly sensitive for small perturbations. Thus making system unstable.

Option 2 : PI Controller

The Integral controller aims to minimize steady-state error by continuously integrating the error signal. However, this approach can lead to a phenomenon known as "controller rollover." When the error changes from positive to negative, the negative error term get subtracted from the existing positive error. It takes time for the subtraction process to reduce the error to zero. However, if the error becomes positive again during this period, it adds up to the previous accumulated error. As a result, the controller's response becomes sluggish and oscillatory.

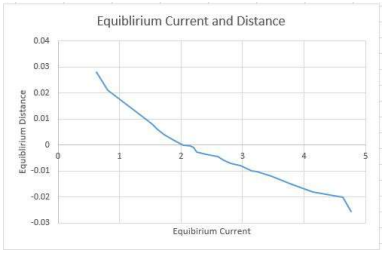
Additionally, the Integral controller can cause an increase in peak overshoot. As the integrated error accumulates over time, it can result in overshooting the desired set-point during transient periods. This overshoot can be problematic.

5 Model Based Design of Measurement Systems

The main objective is to control the motion of the ball during levitation between plates for example if the input of the system is a Sine wave then the motion of the ball between two plates should be sinusoidal or the input is constant position relative between two plates then ball have to levitate at that constant position between two plates. The Ferromagnetic ball has to levitate in two ways. Firstly, levitate the ball by using MATLAB Simulink. On the other hand, use an FPGA board to levitate the Ferro magnetic ball by using the same logic of MATLAB Simulink.

5.1 Equilibrium Distance to Current Converter

The equilibrium current is generated by feedback distance by using lookup tables by hit and trial rules. The Equilibrium current equation describes what should be the position of the ball between plates use to compensate for the values of the output of the adaptive PID filter. The lookup table is converted into an equation.



(a) Graph

Equiblirium_Current_Distance_Lookup_Table.pdf

(b) Lookup Table

Figure 31: Equilibrium Current and Distance

After minimizing the error, use the current to the voltage converter block to apply voltage by the power supply to float the ball between plates. At the end use the voltage to distance converter subsystem again for repeating the same procedure of minimizing error and try movement of the ball will same as the input waveforms. The whole process is explained very in another section.

5.2 Current To Voltage Converter Subsystem

There is a control current as input to the subsystem. There is the lookup table that converts the current into voltage as shown in Figure.

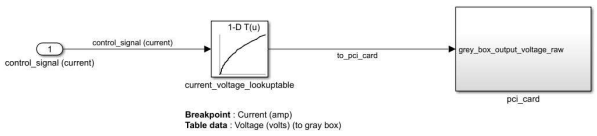


Figure 32: Current to voltage subsystem

Then multiply those values by a half-constant value. Adding the constant value "5" because there is an Analog Output block, whose output is between -10 to 10 volts. So, if the input is from 0 to 5 volts the output of the Analog Output block is from -10 to 0 and if the input is from 5 to 10 volts the output of the Analog Output block is from 0 to 10 as shown in Figure.

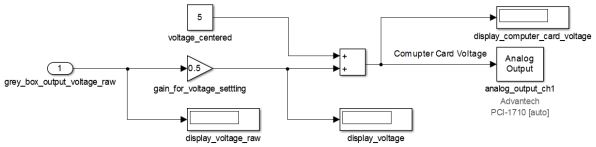


Figure 33: Current Voltage Converter

There are some changes required before using "Analog Output". First, select the board that is used in the project for example "AdvanTech PCI-1710". Moreover insert the value of Ts sample time, Maximum and minimum voltage according to output requirement, Output Channel number and some other parameters as shown below in Figure.

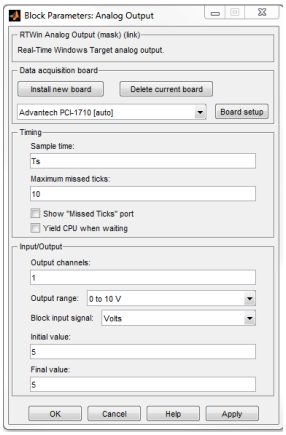


Figure 34: Analog Output Box

reference of requirement no.

5.3 Voltage to Distance Converter Subsystem

There is an Analog Input block in MATLAB whose output is voltage. There are some changes required before using "Analog Input". First, select the board that is used in the project for example "AdvanTech PCI-1710". Moreover insert the value of Ts sample time, Input voltage range according to output requirement, Input Channel number, Output datatype and some other parameters as shown below in Figure.

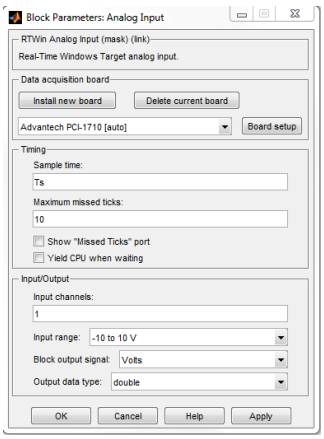


Figure 35: Analog Input Box

First, use the lookup table to get the relationship between voltage and distance of the ball between plates. The lookup table shows that there is a linear relationship between voltage and distance as shown in Figure.

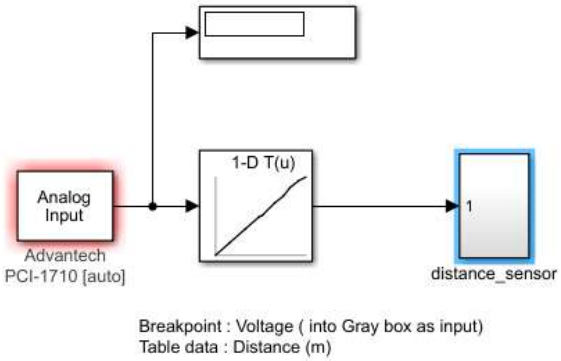
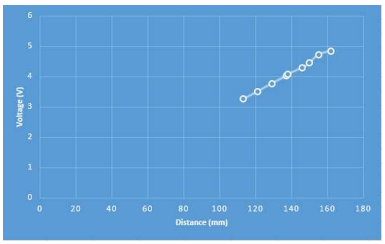


Figure 36: Voltage to Distance Converter

Extract experimental values of voltage by placing the ball at different distances between two plates. The extracted values show linearity. So for clarification plot the graph between voltage and distance as shown in Figures (a) and (b).



(a) Graph

distance(mm)	voltage(v)
113	3.26
121	3.51
129	3.78
137	4.02
138	4.08
146	4.3
150	4.45
155	4.73
162	4.84

(b) Lookup Table

Figure 37: Lookup Table of Voltage to Distance Converter

So instead of using a lookup table, replace the lookup table with by mathematical formula. Two-pint slope formula is used to convert the lookup table into a formula because of a linear relationship. The Linear generalized equation is:

$$y = m * x + c \tag{44}$$

y: Distance
m: Slope
x: voltage
c: y-intercept

Where "y" is output which is distance, "m" is a slope which is "31.01", "x" is input which is voltage and "c" is constant or y-intercept which is "11.91" as shown in Figure below.

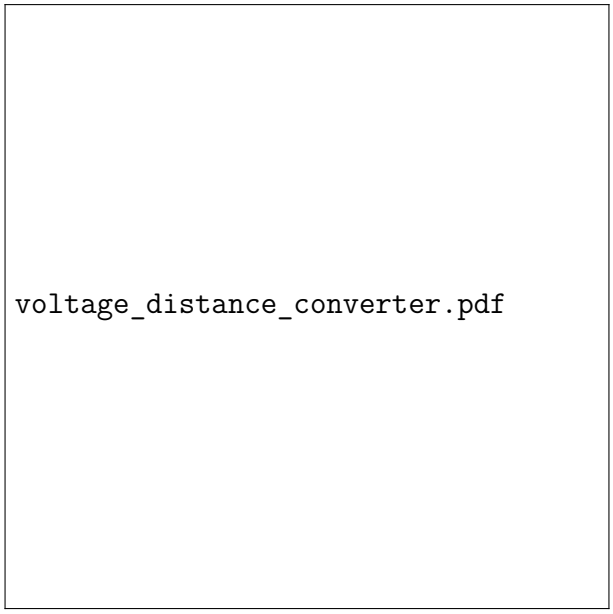


Figure 38: Model of Voltage To Distance Converter

The centred position of the ball between plates at 138mm. So limiting the distance value between -25 to 25, subtract the constant value from the distance and convert it into meters from millimetres values as shown in the Figure. The values of distance are input for feedback control and the derivative of the distance is also calculated to determine the movement of the ball whether the ball is moving upward from the centre point or moving downward from the centre of the point or staying at the centre is in. If the derivative of the ball is positive means the motion is upward, if the derivative is negative moving downward and if a derivative value is zero means stationary in the centre of two plates. reference of requirement no.

6 Implementation of PID Controller in MATLAB

6.1 Key Factors for the Selection of High Gain PID Controllers

Full state feedback controller provides a perfect control over system states but when practicality is considered A Simple PID controller is highly recommended few reasons of choosing PID

- A high gain PID controller doesn't need all the states of the system rather a PID is used for SISO system, where single state is under control.
- a high gain PID controller can handle different uncertainty in the plant

The major problem using a PID controller is handling peak overshoot. This problem can be minimised by implementing only PD Controller. Another major problem is the settling time of the PID controller is higher than State Feedback controller.

6.2 System Architecture

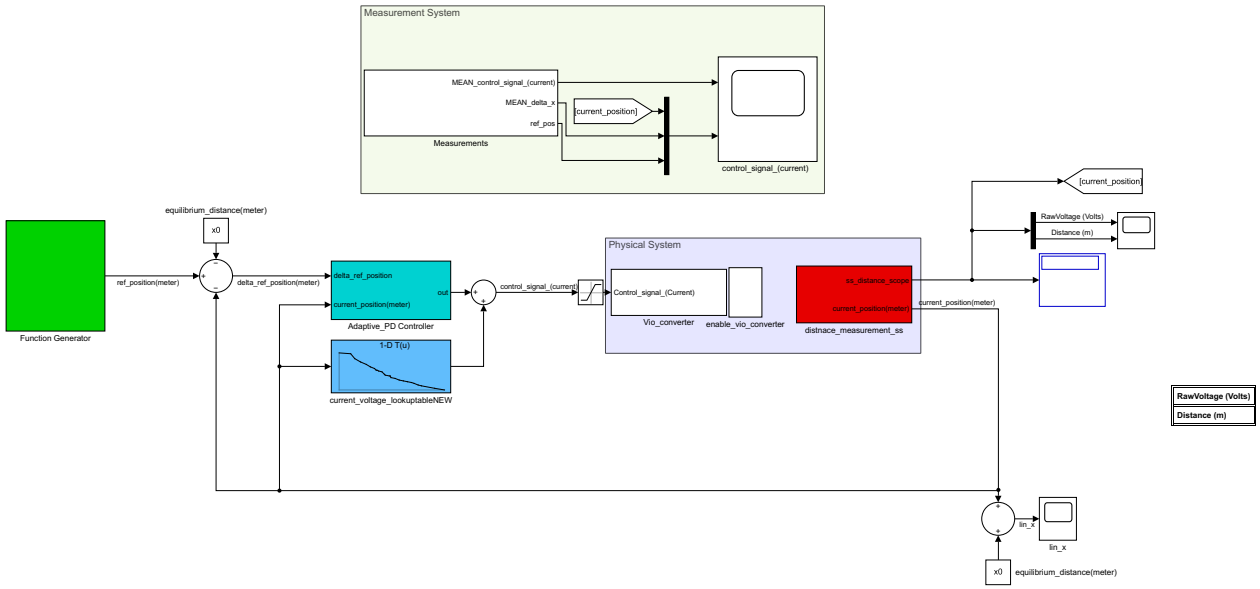


Figure 39: System Architecture

This section explains a detailed system architecture proposed in the planning stage of the project.

6.2.1 Function Generator

This subsystem generates a periodic waveform of User specified Amplitude and Period. A limit of amplitude is $\pm 10mm$ and the period can be adjusted to maximum 10 seconds. A additional feature of the function generator is to provide a constant reference of $\pm 25mm$ when a constant mode is selected. A detail development of the Function Generator is mentioned in the GUI chapter.

6.2.2 Gain Scheduling PID Controller

The behaviour of the plant is highly nonlinear as we move far from the operating point. In order to improve the control it is necessary to fine tune the gains for each position. Hence a Gain scheduling PID controller is utilised.

6.2.3 Equilibrium Current Block

This subsystem provides a equilibrium upon which controller's signal is superimposed to generate a input for the plant. The equilibrium current block consist a co-relation between position and Equilibrium Current generated.

6.2.4 VIO Converter Subsystem

This subsystem converts the physical quantity of the current into corresponding voltage needed to be generated at the input junction of VIO Converter. The current is limited to a fixed value of 5 Ampere. This subsystem also utilises a digital output of a PCI card to trigger ON / OFF VIO converter. A detailed explanation is provided in Chapter

6.2.5 Distance Measurement Subsystem

Distance Measurement Subsystem takes reading from DT-20 laser distance measurement sensor in 0 to 10 V and provides a relative range in 0 to 0.052 meters.

6.2.6 Metering and Measurement Subsystem

This subsystem groups all signals together and display it on the scope. The data gathered is available in both RAW as well as Measurable format.Also the major function of this system is to log the data for further analysis.

6.3 Model Based Development of Equilibrium Current Subsystem

The purpose of a equilibrium current subsystem is to provide a equilibrium current for a respective position of the ball. in order to obtain a perfect co-relation multiple readings were taken manually and framed in a lookup table format.Initially a reading were taken at the Center position and reading by reading the operating point is shifted away from the center in both direction. The key procedure of taking reading is to manually change the current value till a point where the ball will feel weightless.

6.3.1 Development of Equilibrium Current Lookup-table

A temporary subsystem is made to make this procedure easier where a single user can change the input current during simulation with the help of a arrow key without accessing a numeric keypad. This system allows user to change the current in the minimum step size. Calculation of minimum step size

Voltage Range Selected : 0-10V
Resolution of D2A converter : 12 bit

$$\text{Minimum voltage step} = \frac{10}{(2^{12})} = \frac{10}{4096} = 2.44mV$$
$$\text{Minimum current step} = \text{Minimum voltage step} \times \frac{\text{Rated Current Generated}}{\text{Rated Voltage Input}} \tag{45}$$
$$\text{Minimum current step} = 2.44 \times \frac{1}{0.362}$$
$$\text{Minimum current step} = 6.74mA$$

for simpler adjustment Minimum step size is considered as **10 mA**. It is crucial to have a smallest step size since the nature of equilibrium in highly nonlinear.

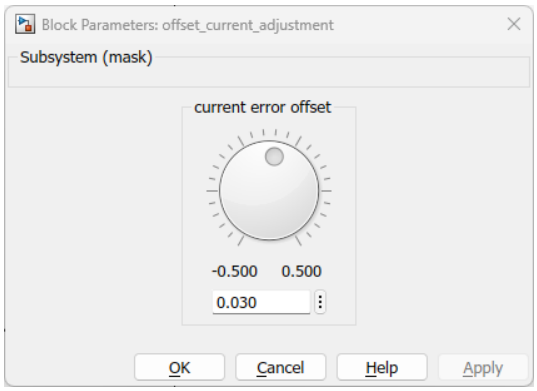


Figure 40: Equilibrium Current Adjustment Mask

A final tabular data of equilibrium current and position is explained in Annex.

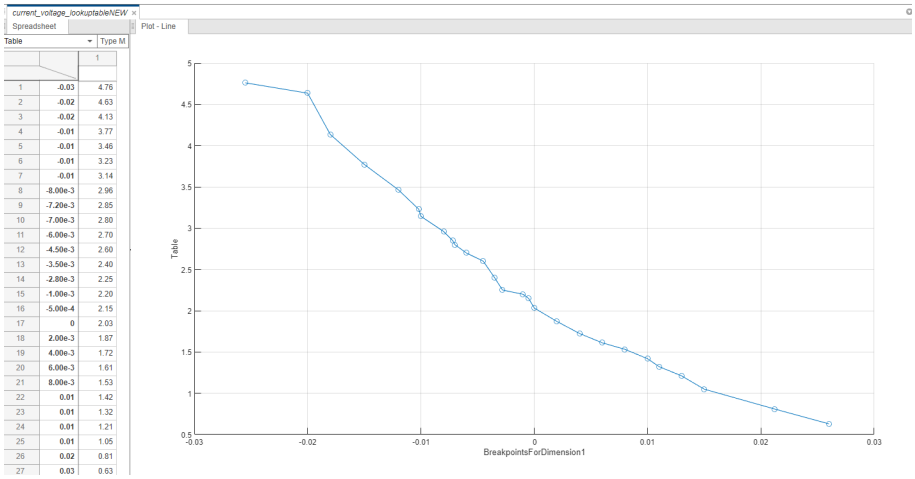


Figure 41: Equilibrium current against Position Lookup-table

The above figure shows the co-relation between Equilibrium current and position. The Y-Axis consist of Current in Amperes and X-Axis consist Position in Meter

6.3.2 Development of Equilibrium Current Equation

The lookup-table method is often considered a viable choice for practical purposes; however, it is not without its drawbacks. One notable issue is the presence of numerous manual errors in the recorded data, as it was collected by different individuals each time. Furthermore, the lack of consistency in the readings on the lookup-table graph, as they were randomly placed, poses another challenge. Consequently, we made the decision to transform the lookup-table into a standardized second-order equation for greater accuracy and reliability.

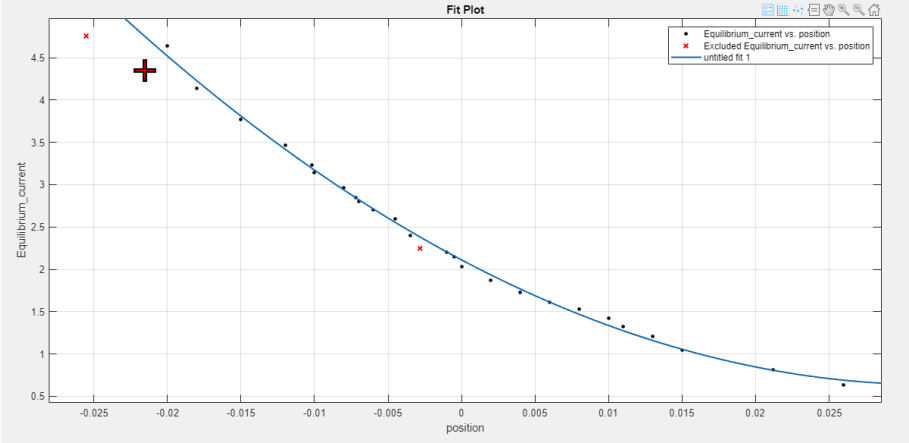


Figure 42: Second order curve fitting in MATLAB

In order to perform optimum calculations, MATLAB's Curve Fitter tool is utilised. within the Command window to generate an array representing the Position and Equilibrium current. It was crucial to accurately include the points within the operational range of

$\pm 10mm$, while disregarding the points at the extremes due to their negative impact on the curve's reliability. The excluded points are visually marked as red crosses on the graph. To facilitate straightforward implementation in fixed point format, A 2^{nd} order equation was selected. However, it is worth noting that higher order equations, such as 4^{th} and 6^{th} order, demonstrated a good ability to precisely replicate all the recorded data points. In the end an equation is developed with following coefficients.

$$EquilibriumCurrent = 1452 * (position)^2 - 91.82 * (position) + 2.104 \tag{46}$$

MATLAB's Function block is utilised to insert this equation in the Simulink.

6.4 Model Based Development of Time Continuous PD Controller

6.4.1 Development of Time Continuous PD Controller

In Chapter 3, a full-state feedback controller was developed, which had the capability to access and utilize all the states of the system. However, in practical applications, it is often the case that we do not have sensors to directly observe the velocity of the system. Therefore, in order to implement the controller effectively, we will adopt a Single-Input Single-Output (SISO) format, where only the position of the ball will be used as the output. We previously have Controllers gains from the chapter 3 for a step size of $\frac{d}{2}$ hence we can use values from

$$State\ Feedback\ Matrix\ K^T = \ [0.2804 \ \ 7.7361]$$

by taking $K_p = 7.7361$ and $K_d = 0.2804$ it is possible to develop a PD controller in time continuous.

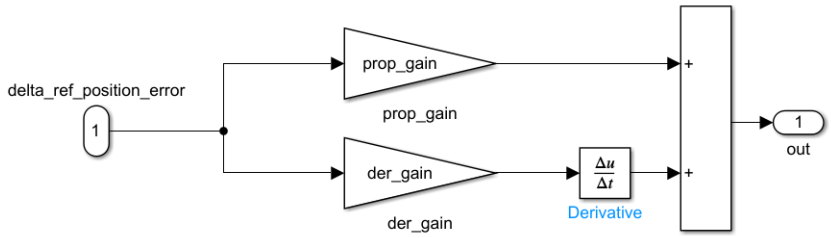


Figure 43: Time Continuous PD controller

from the figure XX it can be observed that although the close-loop PD controller is stable there exists a significant steady state error.

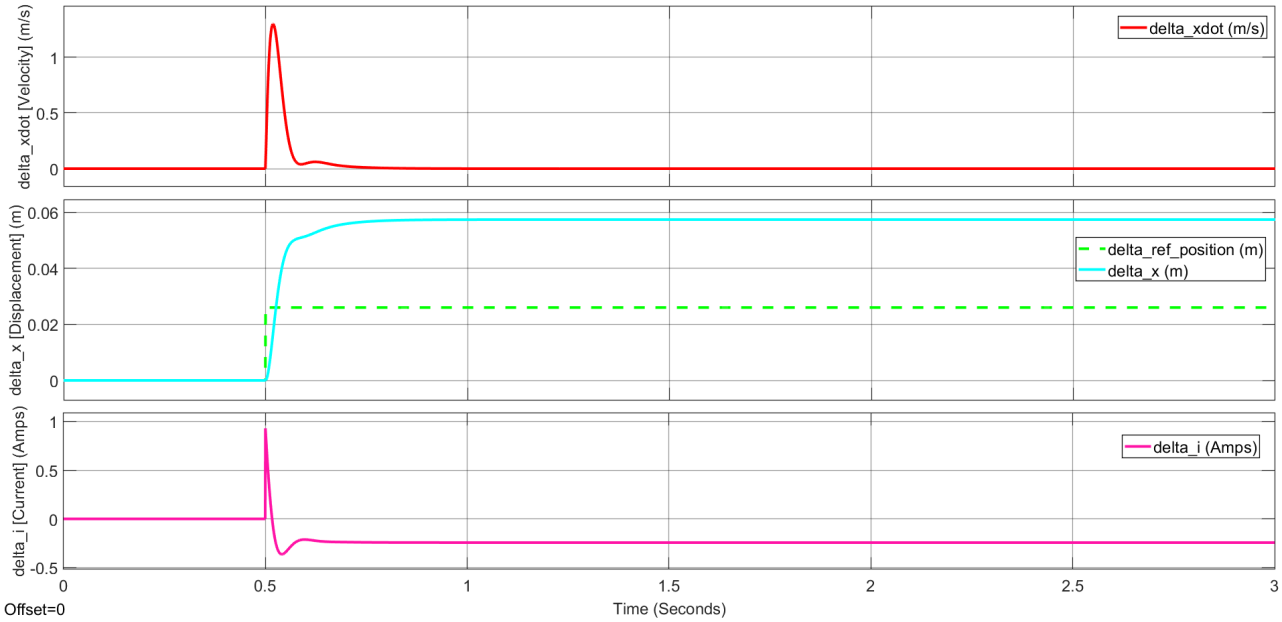


Figure 44: PD Controller Unscaled Step Response

In order to adjust the Proportional and Derivative gains a mask is developed where Proportional and Derivative gains are multiplied with increasing order. A perfect result is obtained for a step size of $\frac{d}{2}$ at $K_{p\ (scaled)} = 54.1527$ and $K_{d\ (scaled)} = 1.4020$.

Selecting of high value of $K_{p\ (scaled)}$ would lead to heavy current draw which is impractical for a physical system. At the same time high $K_{p\ (scaled)}$ will lead to amplification of minor oscillations. The below graph shows tuned step response with $K_{p\ (scaled)} = 54.1527$.

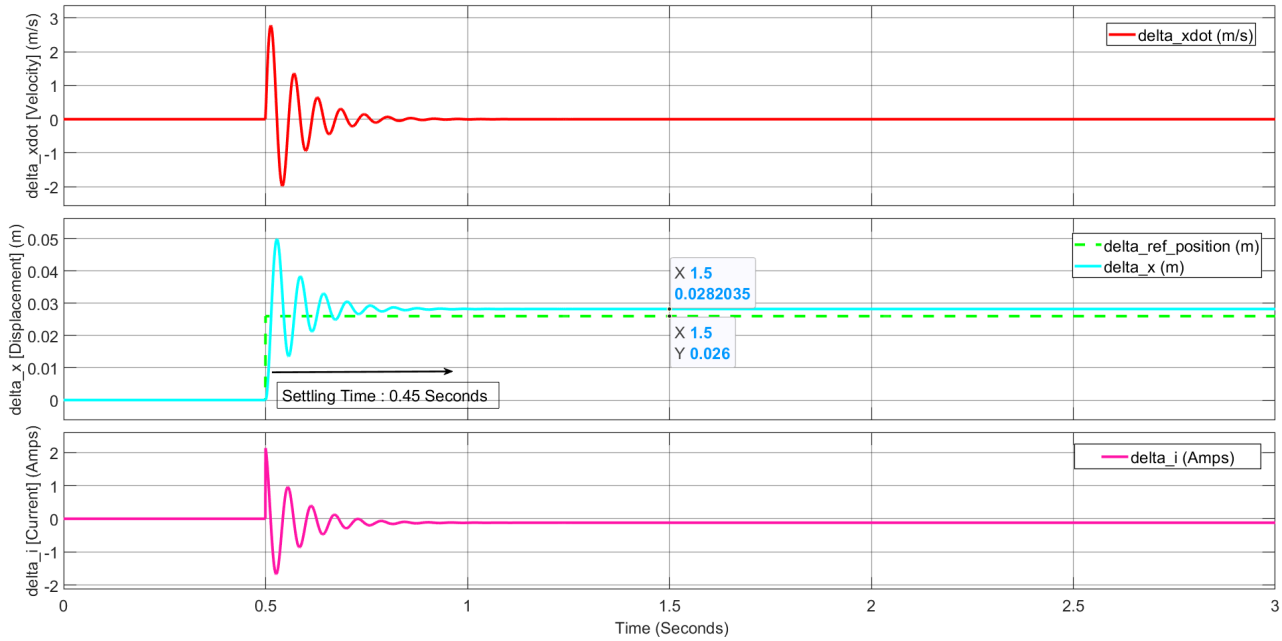


Figure 45: PD Controller scaled Step Response

The Derivative controller is used to damp the oscillations and improve the settling time, but at the same time during a sudden change in value derivative may lead to high amplitude oscillation. Below is the step response of PD Controller with improved settling time with $K_d = 0.2804$.

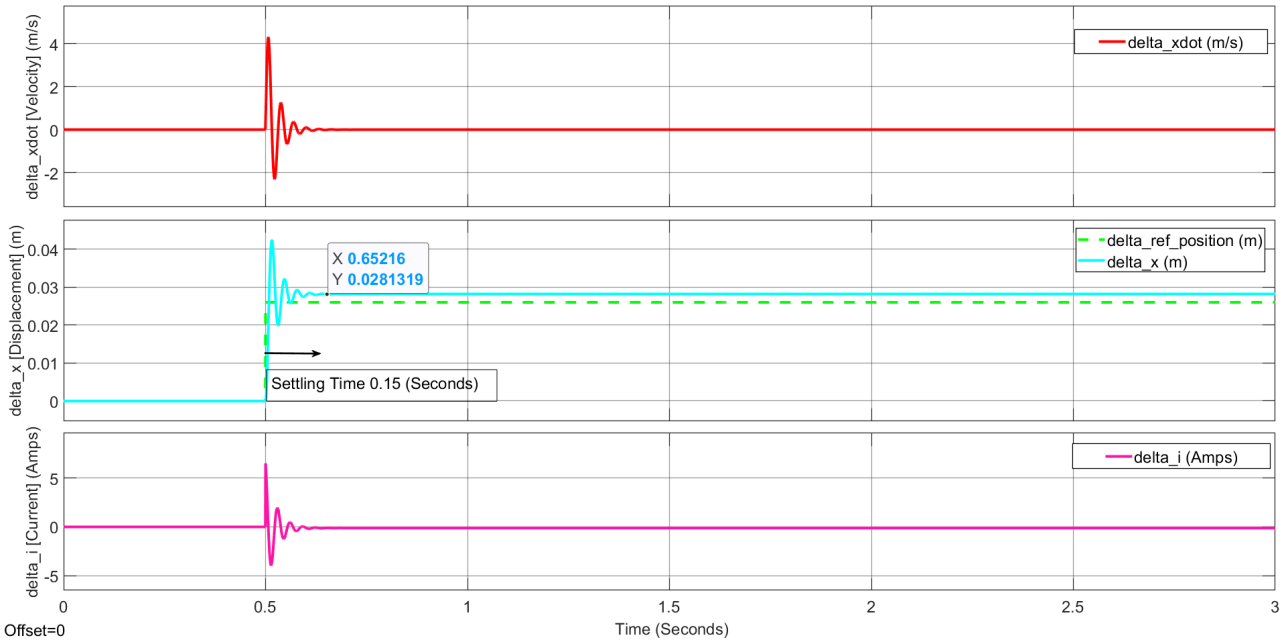


Figure 46: PD Controller scaled Step Response Improved settling time

6.5 Model Based Development of Discrete PD Controller

In order to implement the prototype control logic on a fixed sampling time microcontroller it is necessary to convert the time continuous system into time discrete. To perform this operation various methods are available such as Backward Euler's, Normal Euler's, Zero Order Hold and Bilinear Transform. We have chosen Bilinear transformation because of its great ability of handling step response.

6.5.1 Selection of Perfect Discretisation Method

As there exist multiple methods we have to choose a perfect method which will perfectly replicate a time continuous system in time discrete domain. In order to determine the perfect fit Time continuous system is discredited with "Zero Order Hold", "First Order Hold", "Impulse", "Tustin (Bilinear Transform)" and "Matched Transform". The detailed analysis on this system is done by observing the Step Response and Bode plot,

```
s = tf('s');
PD_contr = 7*feedback_cnt(1,1)*s+7*feedback_cnt(1,2);
closeloop_tf = feedback(plant_tf*PD_contr,1);
closeloop_tf_Z = c2d(closeloop_tf,Ts,'zoh');
closeloop_tf_F = c2d(closeloop_tf,Ts,'foh');
closeloop_tf_I = c2d(closeloop_tf,Ts,'impulse');
```

```
closeloop_tf_M = c2d(closeloop_tf,Ts,'matched');
closeloop_tf_T = c2d(closeloop_tf,Ts,'tustin');
step(closeloop_tf,closeloop_tf_Z,closeloop_tf_F,
       closeloop_tf_I,closeloop_tf_T,closeloop_tf_M);
bode(closeloop_tf,closeloop_tf_Z,closeloop_tf_F,
       closeloop_tf_I,closeloop_tf_T,closeloop_tf_M);
```

The following presents the step response for each method. By examining the step response of a discrete system, we can assess the similarity between the discretized system and the continuous-time system when a step input is introduced. Ideally, the discrete step response should align with the continuous step response precisely at the sampling instant.

Time Continuous	$\frac{70s + 5455}{s^2 + 70s + 4701}$
Zero Order Hold	$\frac{0.676z - 0.2987}{z^2 - 1.171z + 0.4966}$
First Order Hold	$\frac{0.3458z^2 + 0.1921z - 0.1607}{z^2 - 1.171z + 0.4966}$
Impulse Transform	$\frac{0.7z^2 - 0.2103z + 3.451e - 17}{z^2 - 1.171z + 0.4966}$
Tustin	$\frac{0.3314z^2 + 0.1859z - 0.1456}{z^2 - 1.203z + 0.523}$
Matched	$\frac{0.6971z - 0.3198}{z^2 - 1.171z + 0.4966}$

Table 4: Transfer Functions with various discretisation methods

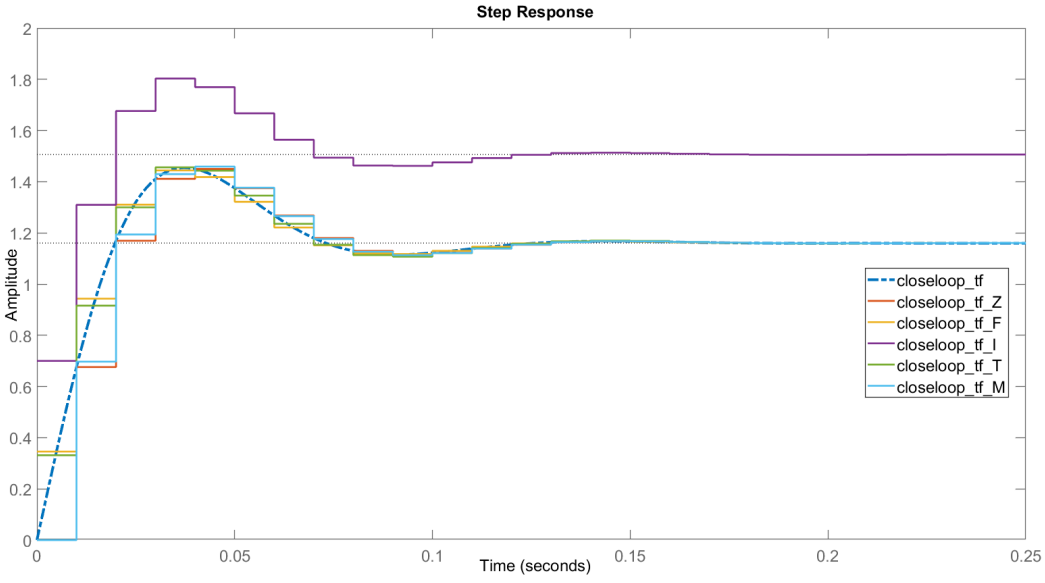


Figure 47: Step response of Discredited System

By analyzing the figure, it is evident that the step responses of the "Zero Order Hold," "Pole matched transform," and "Tustin Transform" exhibit a close resemblance to the continuous-time system. However, the "Impulse Invariant method" and "First order hold" show a lag of a few samples. It is crucial for us to select the most suitable method where the discretized (Computer) system closely aligns with the real-time system. To further narrow down our options, we can examine the bode plot of all the discretized systems. The subsequent section provides a bode analysis for each of the discretized systems.

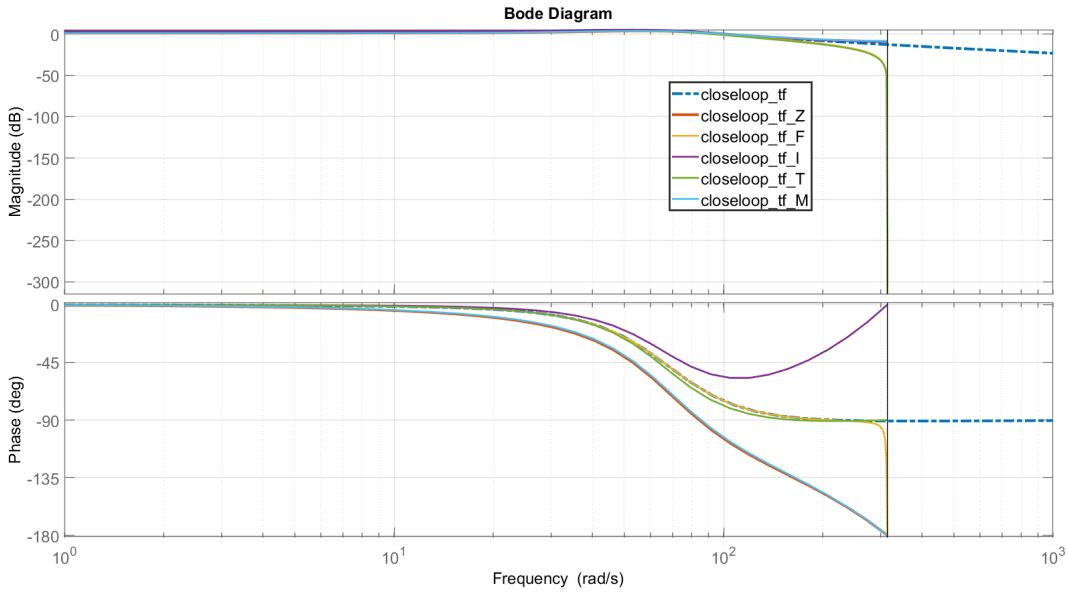


Figure 48: Bode plot of discretised system

The bode plot reveals that the "Zero Order Hold" and "Pole Matched Transform" methods do not perform well at higher input frequencies. According to the Nyquist criterion, for a system with a sampling frequency of 100 Hz, the maximum input frequency should be limited to 50 Hz. It is noteworthy that for a 50 Hz input signal, the "First Order Hold" and "Tustin Transform" methods demonstrate Phase and Gain margins identical to those of the continuous-time system.

In conclusion, the "Tustin Transform" exhibits a remarkable resemblance to the continuous-time system. A controller developed using the Tustin transform will behave almost similarly to the continuous-time system, as long as the sampling time remains consistent.

6.5.2 Development of Time Discrete PD controller

Method 1) Discreitisation of Time continuous Controller In this method a Time continuous controller is discretised to fit for a Time continuous plant. The Time continuous controller developed in chapter is discretised with sampling time of 0.01s. The transfer function of controller in Time Continuous domain is given by

$$C(S) = K_P + sK_D \tag{47}$$

$$where K_P = 54.1527 \quad and \quad K_D = 1.4020$$

By applying Tustin Transform to convert the controller to discrete form we get,

$$C(Z) = \frac{111.6z - 67.88}{z + 1} \tag{48}$$

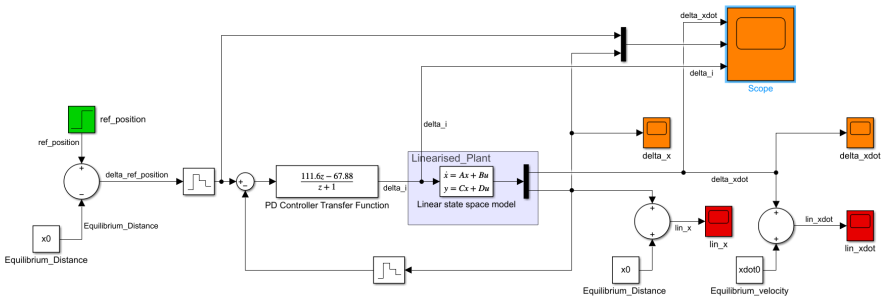


Figure 49: Simulation of discretised PD controller Transfer Function

The step response of discredited controller was showing a oscillatory behaviour

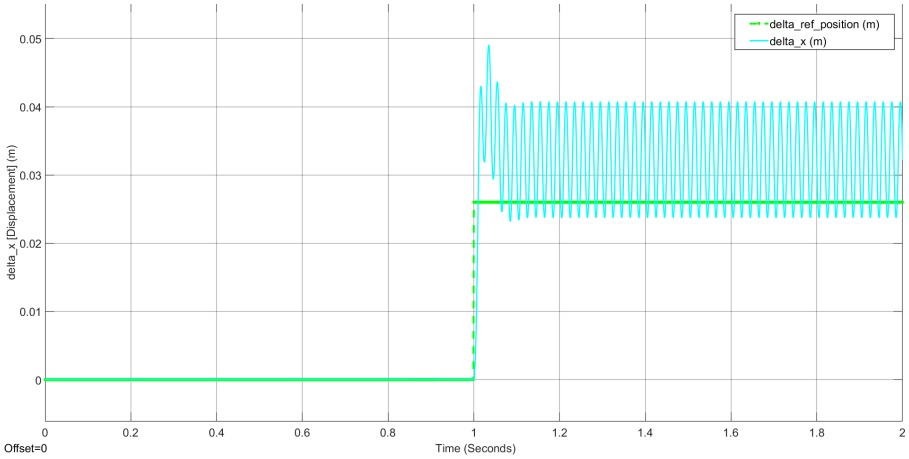


Figure 50

Method 2) Development of time discrete controller for discretised plant In this method a plant is completely discritised and a PD controller is developed. based on a discretised plant. A model based development of discrete PD controller is as shown below.

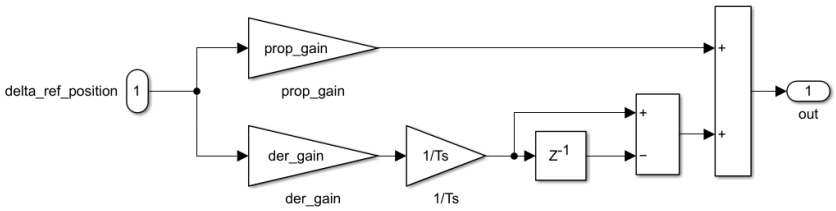


Figure 51: Model Based Discrete PD controller

A simulation setup of a discrete PD controller is as below.

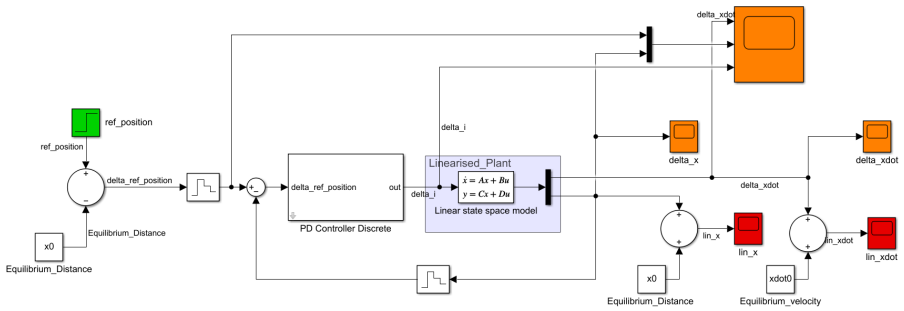


Figure 52: Simulation of Discrete PD Controller

With the same scaling factor as Time continuous controller, the values from Discrete Feedback vector are scaled to $K_{p\ (scaled)} = 21.8169$ and $K_{d\ (scaled)} = 0.5952$.

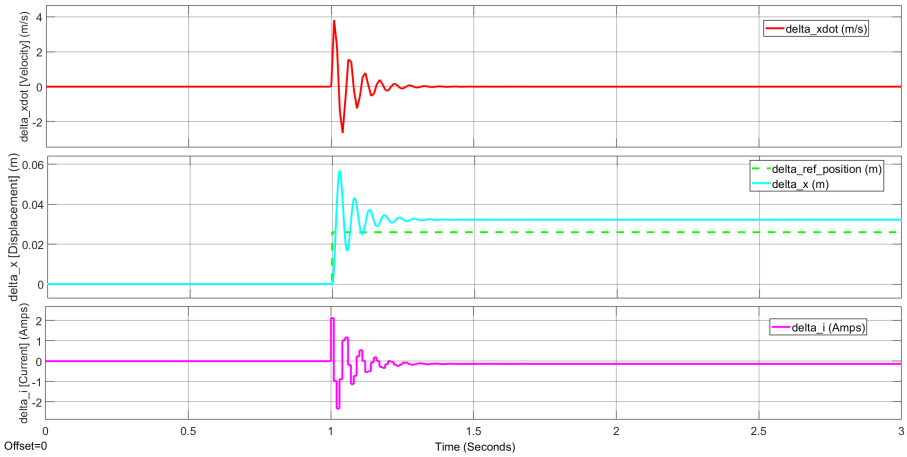


Figure 53: Simulation of Discrete PD Controller

After conducting a analysis of the results obtained from both Time Continuous and Time Discrete simulations, it has been observed that the developed PD controller exhibits satisfactory performance. The controller demonstrates a settling time of 200ms and a maximum peak current of 2A.

Based on this analysis, it can be concluded that the selected poles for the controller have proven to be effective in achieving the desired performance. The chosen poles contribute to the controller's ability to achieve a satisfactory settling time and maintain peak current within acceptable limits.

6.6 Gain Scheduling Controller

In order to achieve a smoother results with ever less setting time a gain scheduling controller is implemented where the Gains of the controller varies from the current value of states. In our case the Gain of Proportional and Derivative controller varies as per position of the ball.

6.6.1 Proportional Gain Selection

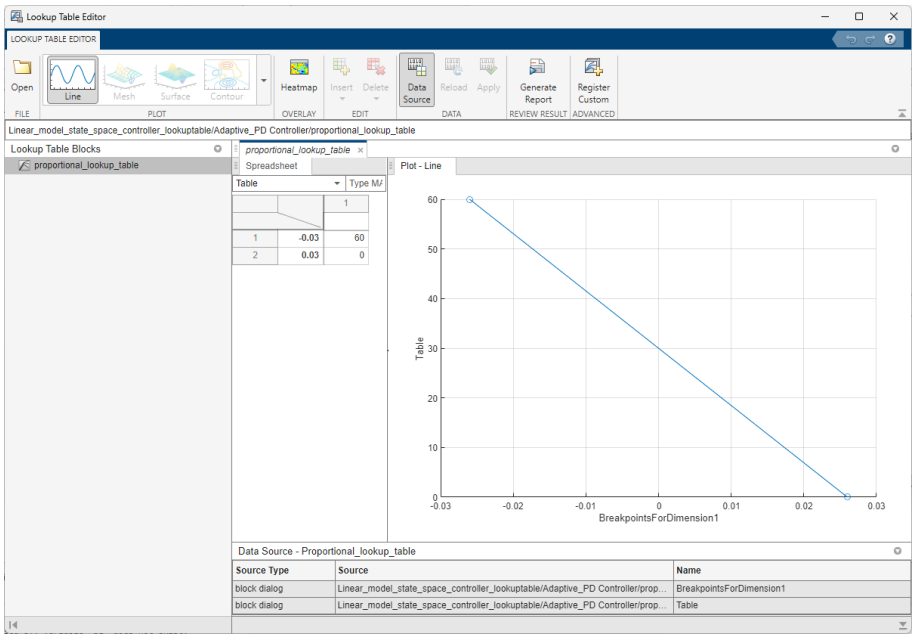


Figure 54: Variation of Proportional Gain over distance

The gains of proportional controller was scaled withthe factor of 3.

6.6.2 Derivative Gain Selection

The nature of the system is almost linear at $\Delta x = 0$ but it changes at we move away from the operating point where the system is linearised i.e. $(x_0\ ,\ i_0)$. Hence at extreme ends the ball experience heavy oscillations. In order to avoid the oscillations the derivative gain should be higher at $\Delta x = \pm d/2$ and should be lower at $\Delta x = 0$. Hence the derivative Gain is scheduled as .

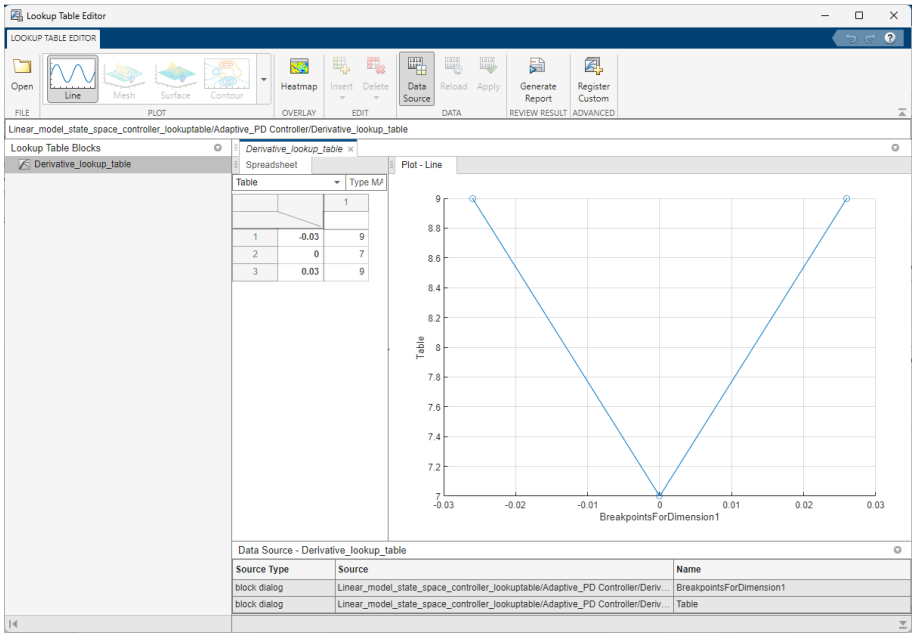


Figure 55: Variation of Derivative Gain over distance

6.6.3 Model based design of Gain scheduling controller

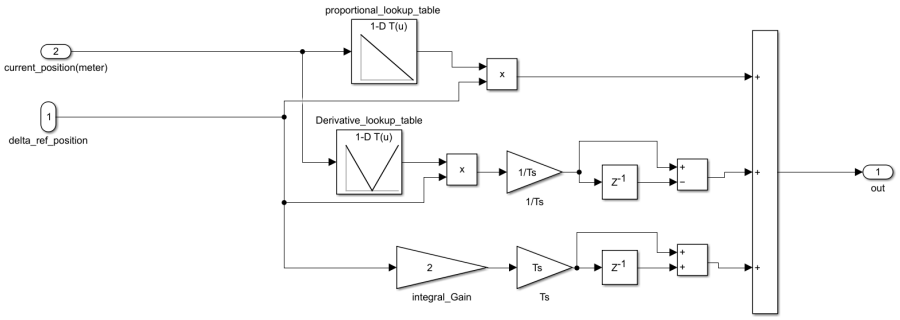


Figure 56: Model based design of Gain scheduling controller

7 Hardware Design and Implementation

7.1 Introduction of the ZEDBOARD

ZedBoard™ is a low-cost development board for the Xilinx Zynq®-7000 SoC[22]. This board contains everything necessary to create a Linux, Android, Windows® or other OS/RTOS-based design. Additionally, several expansion connectors expose the processing system and programmable logic I/Os for easy user access. Take advantage of the Zynq-7000 SoC's tightly coupled ARM® processing system and 7 series programmable logic to create unique and powerful designs with the ZedBoard[23].

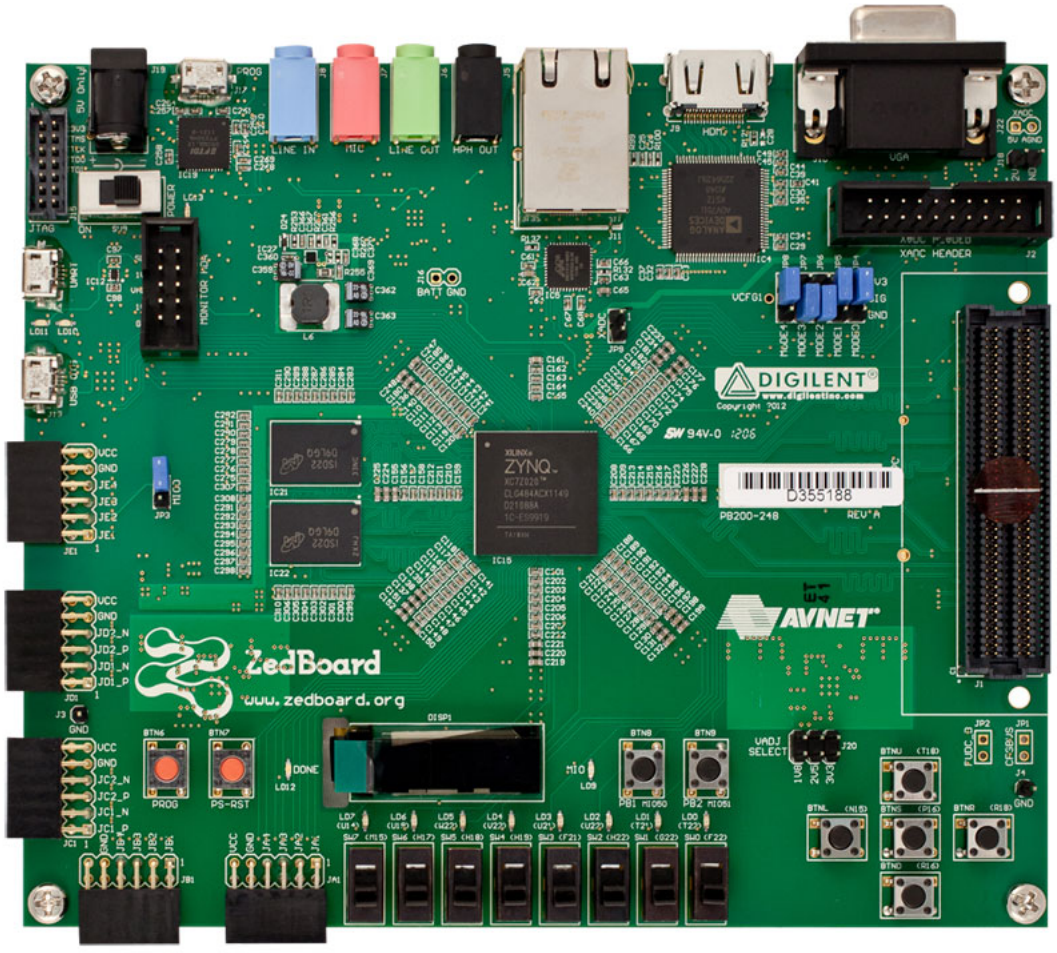


Figure 57: Zynq-7000 FPGA Board (Zedboard)

7.1.1 Processing System Features and Description

1. Dual ARM Cortex-A9 MPCore CPUs with ARM v7
    - Run time options allow single processor, asymmetrical (AMP) or symmetrical multiprocess- ing (SMP) configurations

- NEON™ 128b SIMD coprocessor and VFPv3 per MPCore
- 32KB instruction and 32KB data L1 caches with parity per MPCore
- 512 KB of shareable L2 cache with parity
- Private timers and watchdog timers

2. System-Level Control Registers (SLCRs)
3. Snoop control unit (SCU) to maintain L1 and L2 coherency
    - Accelerator coherency port (ACP) from PL (master) to PS (slave)
4. 256 KB of on-chip SRAM (OCM) with parity
5. DMA controller
6. General interrupt controller (GIC)
7. Watchdog timer, triple counter/timer
8. Memory Interfaces
9. I/O Peripherals

7.1.2 Programmable Logic Features and Descriptions:

The PL provides a rich architecture of user-configurable capabilities.

1. Accelerator coherency port (ACP) from PL (master) to PS (slave)
2. Configurable logic blocks (CLB)
    - 6-input look-up tables (LUTs)
    - Memory capability within the LUT
    - Register and shift register functionality
    - Cascadeable adders
3. 36 KB block RAM
    - Dual port
    - Up to 72-bits wide
    - Configurable as dual 18 KB
    - Programmable FIFO logic
    - Built-in error correction circuitry
4. Digital signal processing — DSP48E1 Slice
5. 25 × 18 two's complement multiplier/accumulator high-resolution signal processor

6. Power saving 25-bit pre-adder to optimize symmetrical
7. filter applications
8. optional pipelining, optional ALU, and dedicated buses for cascading
9. Clock management
10. High-speed buffers and routing for low-skew clock distribution
11. Frequency synthesis and phase shifting
12. Low-jitter clock generation and jitter filtering
13. Configurable I/Os
14. High-performance Select IO technology
15. High-frequency decoupling capacitors within the package for enhanced signal integrity
16. Digitally controlled impedance that can be 3-stated for lowest power, high-speed I/O operation
17. High range (HR) I/Os support 1.2V to 3.3V
18. High performance (HP) I/Os support 1.2V to 1.8V (7z030, 7z035, 7z045, and 7z100 devices)
19. Low-power gigabit transceivers (7z015, 7z030, 7z035, 7z045, and 7z100 devices)
20. High-performance transceivers capable of up to 12.5 Gb/s (GTX) in 7z030, 7z035, 7z045 and 7z100 devices
21. High-performance transceivers capable of up to 6.25 Gb/s (GTP) in 7z015

7.2 PMOD(Peripheral Module)

What is meaning of the PMOD? PMOD interface (Peripheral Module) is an open standard defined by Digilent Inc for peripherals used with FPGAs or microcontroller development boards[15].

7.2.1 Digilent PMOD Interface Specification

The Digilent Pmod interface is used to connect low-frequency, low I/O pin count peripheral modules to host controller boards[5]. There are six-pin and twelve-pin versions of the interface defined. The six-pin version provides four digital I/O signal pins, one power pin, and one ground pin. The twelve-pin version provides eight I/O signal pins, two power pins, and two ground pins. The signals of the twelve-pin version are arranged so that it provides two of the six-pin interfaces stacked.

In general, Pmod peripheral modules can plug directly into connectors on the host controller board, or be connected to the controller board via six-pin or twelve-pin cables. Pmod peripheral modules are powered by the host via the interface's power and ground pins.

Electrical Specifications
The digital signal characteristics are not specified. However, the general expectation is that a 3.3V logic power supply will be used and the signals will conform to LVCMOS 3.3V or LVTTL 3.3V logic conventions. The driver current source/sink capability isn't specified and depends on the capabili- ties of the specific system board or module. The I/O pins on the system board are generally directly driven by the FPGA . The I/O pins on Xilinx FPGAs generally have symmetrical 24mA source/sink capability.

Pmod Interface Type 2 (SPI)
This provides a Serial Peripheral Interface (SPI) port. The host generally acts as an SPI master device and the peripheral module generally acts as an SPI slave device. When this interface is placed on a 12-pin connector on a host, it should use pins 1-6 (i.e. the upper row of pins).

7.3 PMOD DAC

The Pmod DA2 is a 12-bit Digital-to-Analog converter[4] powered by the Texas Instruments DAC121S101. The DAC121S101 is a full-featured, general-purpose 12-bit voltage-output digital-to-analog converter and is manufactured on an Automotive Grade that can operate in the range of 2.7V to 5.5V and consumes just 177 µA of current at 3.6 Volts. The on-chip output amplifier allows rail-to-rail output swing and the three-wire serial interface operates at clock rates up to 30 MHz over the specified supply voltage range and is compatible with Output standard SPI™, QSPI, MICROWIRE, and DSP interfaces. Competitive devices are limited to 20 MHz clock rates at supply voltages in the 2.7V to 3.6V range.
The supply voltage for the DAC121S101 serves as its voltage reference, providing the widest possible output dynamic range. A power-on reset circuit ensures that the DAC output power is up to zero volts and remains there until there is a valid write to the device. A power-down feature reduces power consumption to less than a microwatt. As it can simultaneously convert two separate channels of digital information provided over an interface like SPI, users can easily compare the two reconstructed signals.

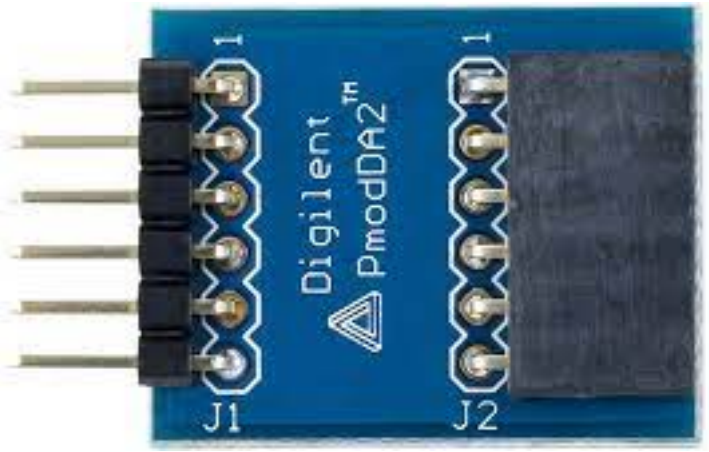


Figure 58: digital pmodDA2

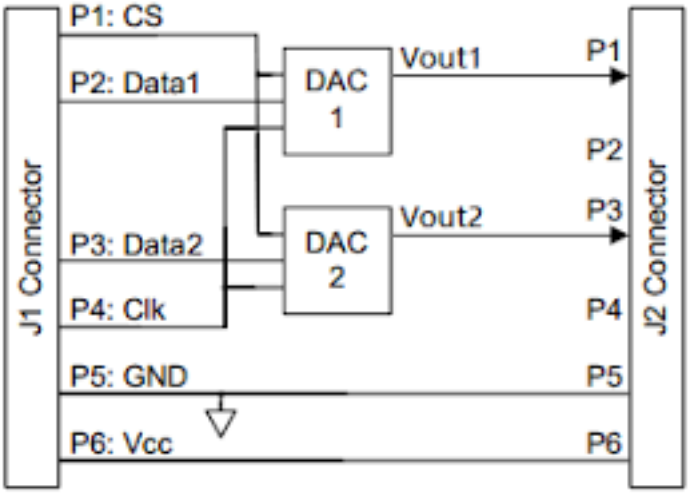


Figure 59: Block Diagram of Pmodda2

7.3.1 Features

- Two Semiconductor DAC 121S101, 12-bit D/A converters
- 6-pin header and 6-pin connector
- two simultaneous D/A conversion channels
- very low power consumption
- small form factor

7.3.2 Function

The Pmodda2 can produce an analog output ranging from 0-3.3 volts when operated with a 3.3V power supply. It has two simultaneous D/A conversion channels, each with a 12-bit converter that can process separate digital signals, allowing users to achieve a resolution up to about 1mV. The two converters are connected in parallel so that commands are sent to both converters simultaneously.

7.4 PMOD ADC

The Digilent Pmod interface is used to connect low frequency, low I/O pin count peripheral modules to host controller boards.The Pmodad1 uses a 6-pin header connector.

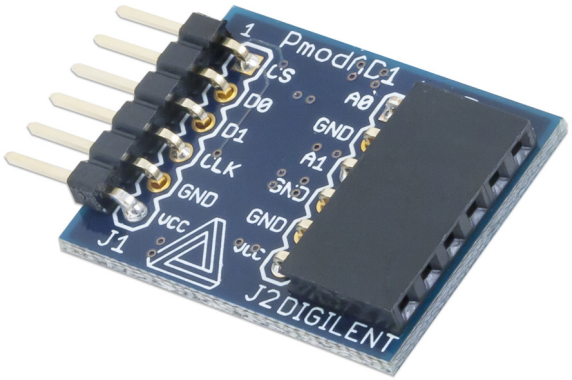


Figure 60: Digilent Pmodad1[14]

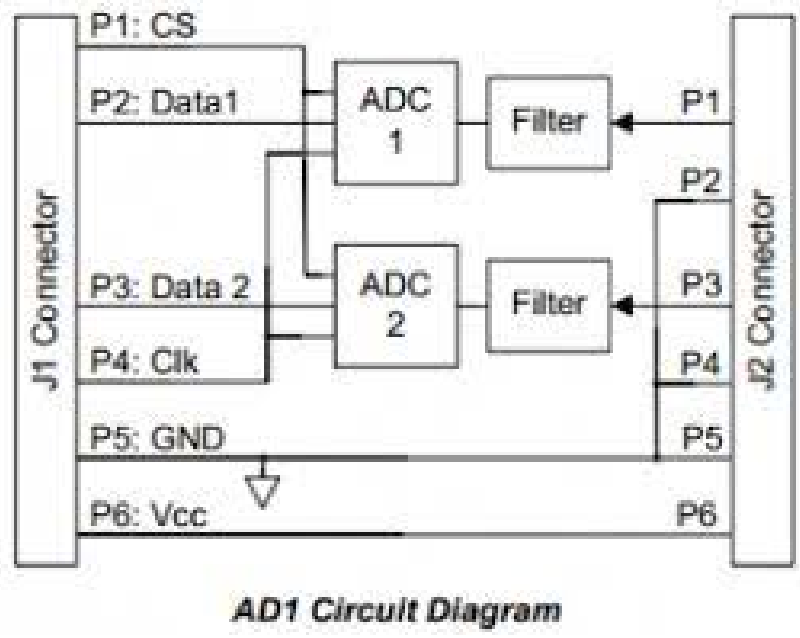


Figure 61: Block Diagram of Pmodad1

7.4.1 Features

- Two channel 12-bit analog-to-digital converter
- Two 2-pole Sallen-Key anti-alias filters
- Small PCB size for flexible designs
- 6-pin header connector
- 6-pin connector
- deficient power consumption

7.4.2 Function

The Pmodad1 converts an analog input signal ranging from 0-3.3 volts to a 12-bit digital value in the range of 0 to 4095. The filters limit the analog signal bandwidth to a frequency range suitable to the sample rate of the converter. The Pmodad1 uses the SPI serial bus standard to send converted data to the host system. Any external power applied to the Pmodad1 must be within 1.06 to 1.62 volts to ensure that all of the components in the Pmodad1 work correctly.

The communication between ZYNQ BOARD and peripheral devices is made through the SPI protocol. The transfer of analog and digital values from the zynq device and test-stand is initialized by the SPI controller through the pmod circuits according to their functionalities. For this purpose, Zynq provides two SPI controllers. These will be discussed in the next section.

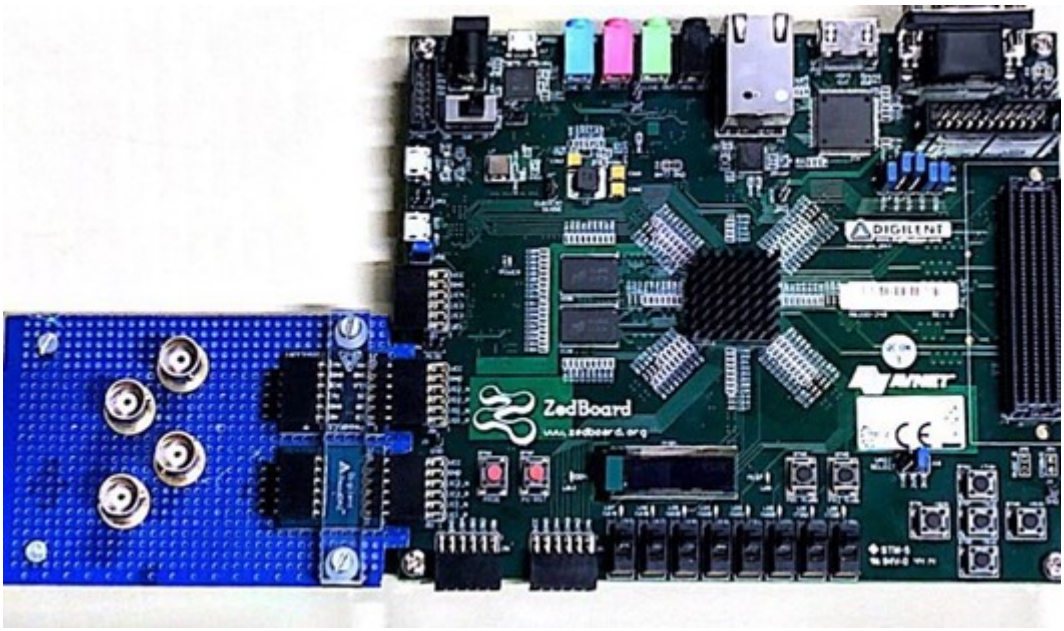


Figure 62: FPGA with Pmodda2 and Pmodad1

7.5 SPI Protocol

The SPI bus controller enables communications with a variety of peripherals such as memories, temperature sensors, pressure sensors, analog converters, real-time clocks, displays, and any SD card with serial mode support. The SPI controller can function in master mode, slave mode, or multi-master mode. The Zynq-7000 devices include two SPI controllers. The controller is based on the Cadence SPI core. In master mode, the controller drives the serial clock and the slave selects with an option to support SPI's multi-master mode. The serial clock is derived from the PS clock subsystem. The controller initiates messages using up to 3 individual slave select (SS) output signals that can be externally expanded. The controller reads and writes to slave devices by writing bytes to the 32-bit read/write data port register.

In multi-master mode, the controller three-states its output signals when the controller is not active and can detect contention errors when enabled. The outputs are three-stated immediately by resetting the SPI enable bit. An Interrupt Status register indicates a mode fault. In slave mode, the controller receives the serial clock from the external device and uses the SCLK to synchronize data capture. The slave mode includes a programmable start detection mechanism when the controller is enabled while the SS is asserted.

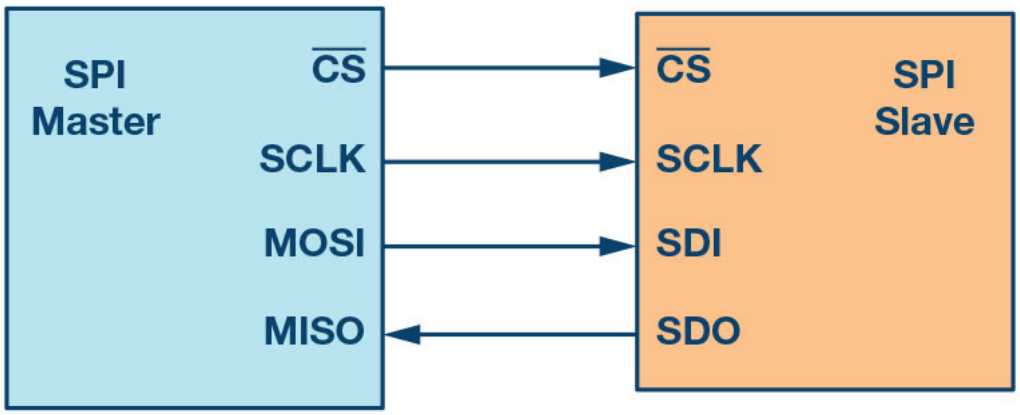


Figure 63: SPI master and slave interface[17]

SPI protocol consists of four wires such as MISO, MOSI, SCLK, SS used for master/slave communication. The master is a microcontroller, and the slaves are other peripherals like sensors, GSM modems and GPS modems, etc. The multiple slaves are interfaced with the master through an SPI serial bus.

MISO (Master in Slave out): The MISO line is configured as an input in a master device and as an output in a slave device.

MOSI (Master out Slave in): The MOSI is a line configured as an output in a master device and as an input in a slave device wherein it is used to synchronize the data movement.

SCLK (serial clock): This signal is always driven by the master for synchronous data transfer be- tween the master and the slave. It is used to synchronize the data movement

both in and out through the MOSI and MISO lines.

SS (Slave Select): This signal is driven by the master to select individual slaves/Peripheral devices. It is an input line used to select the slave devices.

7.5.1 Master Slave Communication with SPI Serial Bus

Here, the communication is always initiated by the master. The master device first configures the clock frequency which is less than or equal to the maximum frequency that the slave device supports. The master then selects the desired slave for communication by dragging the slave select line (SS) of that particular slave device to go low state and active. The master generates the information on to the MOSI line that carries the data from master to slave.

7.5.2 Clock Phase and Polarity

The master configures the clock polarity (CPOL) and clock phase (CPHA) to correspond to slave device requirements. These parameters determine when the data must be stable, when it should be changed according to the clock line and what the clock level is when the clock is not active. The CPOL parameter assigns the clock level when the clock is not active. The clock (SCLK) signal may be inverted (CPOL=1) or non-inverted (CPOL=0). For the inverted clock signal, the first clock edge is falling. For the non-inverted clock signal, the first clock edge is rising. The CPHA parameter is used to shift the capturing phase. If CPHA=0, the data are captured on the leading (first) clock edge, regardless of whether that clock edge is rising or falling. If CPHA=1, the data are captured on the trailing (second) clock edge; in this case, the data must be stable for a half cycle before the first clock cycle.

SPI Modes	Clock Polarity	Clock Phase
0	0	0
1	0	1
2	1	0
3	1	1

There are four possible modes that can be used in an SPI protocol:

1. For CPOL=0, the base value of the clock is zero. For CPHA=0, data are captured on the clock's rising edge and data are propagated on a falling edge.
2. For CPOL=0, the base value of the clock is zero. For CPHA=1, data are captured on the clock's falling edge and data are propagated on a rising edge.
3. For CPOL=1, the base value of the clock is one. For CPHA=0, data are captured on the clock's rising edge and data are propagated on a falling edge.

4. For CPOL=1, the base value of the clock is one. For CPHA=1, data are captured on the clock's falling edge and data are propagated on a rising edge.

In our case, Peripheral module ADC is clock polarity = 1 (CPOL = 1) and clock phase = 1 (CPHA =1) device and DAC is clock polarity = 0 (CPOL = 0) and clock phase = 1 (CPHA = 1).

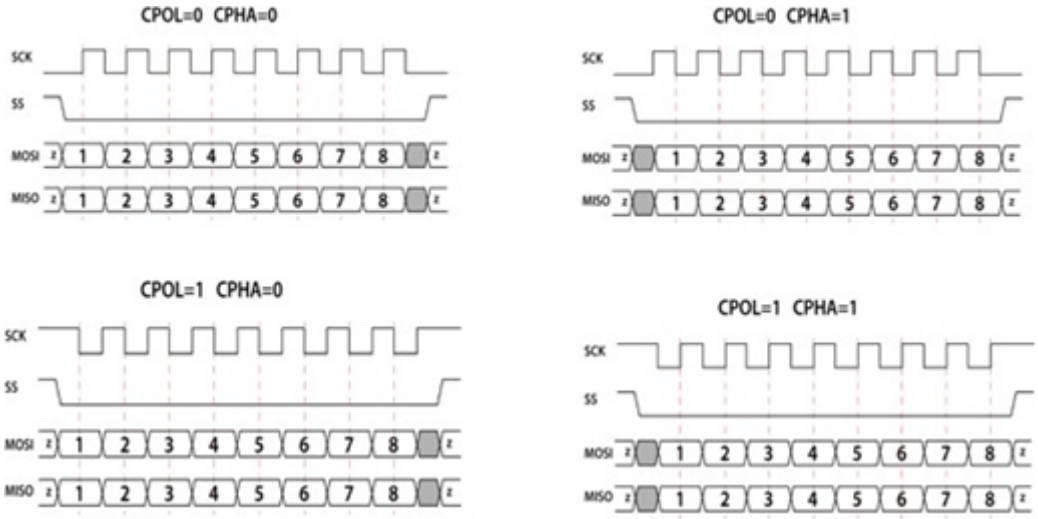


Figure 64: Spi Protocol

Now we will utilize SPI protocol for communication using FSM (Finite State Machine) approach. The VHDL code is written for this purpose in vivado tool provided by Xilinx.

A state machine is a modelling design technique for sequential circuits. At any time, the machine sits in one of a finite number of possible states. For each state, both the output values and the transition conditions into other states are fully defined. The state is stored by the FSM, and the transition conditions are usually reevaluated at every (positive) clock edge, so the state-change procedure is always synchronous because the machine can only move to another state when the clock ticks.

The FSM model provides a systematic approach (a method) for designing sequential circuits, which can lead to optimal or near-optimal implementations. Moreover, the method does not require any prior knowledge or specifics on how the general circuit (solution) for the problem at hand should look like.

7.6 State Machines for SPI Protocol

7.6.1 DAC State Machine and Entity declaration

The below state diagram depicts the functionality of the DAC interface. The entity section will include the following input and output signals:

- Entity declaration represents the external interface to the design entity.

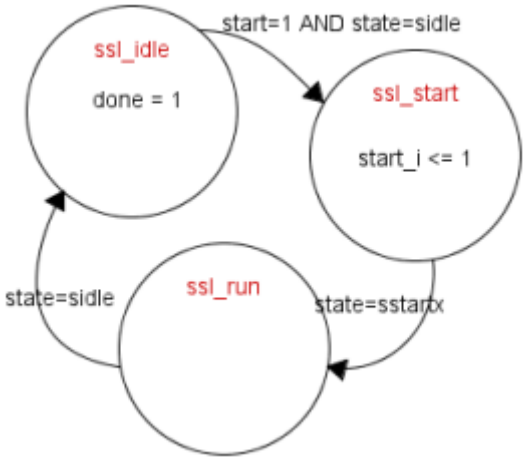


Figure 65: SPI DAC State Machine Start/Stop logic

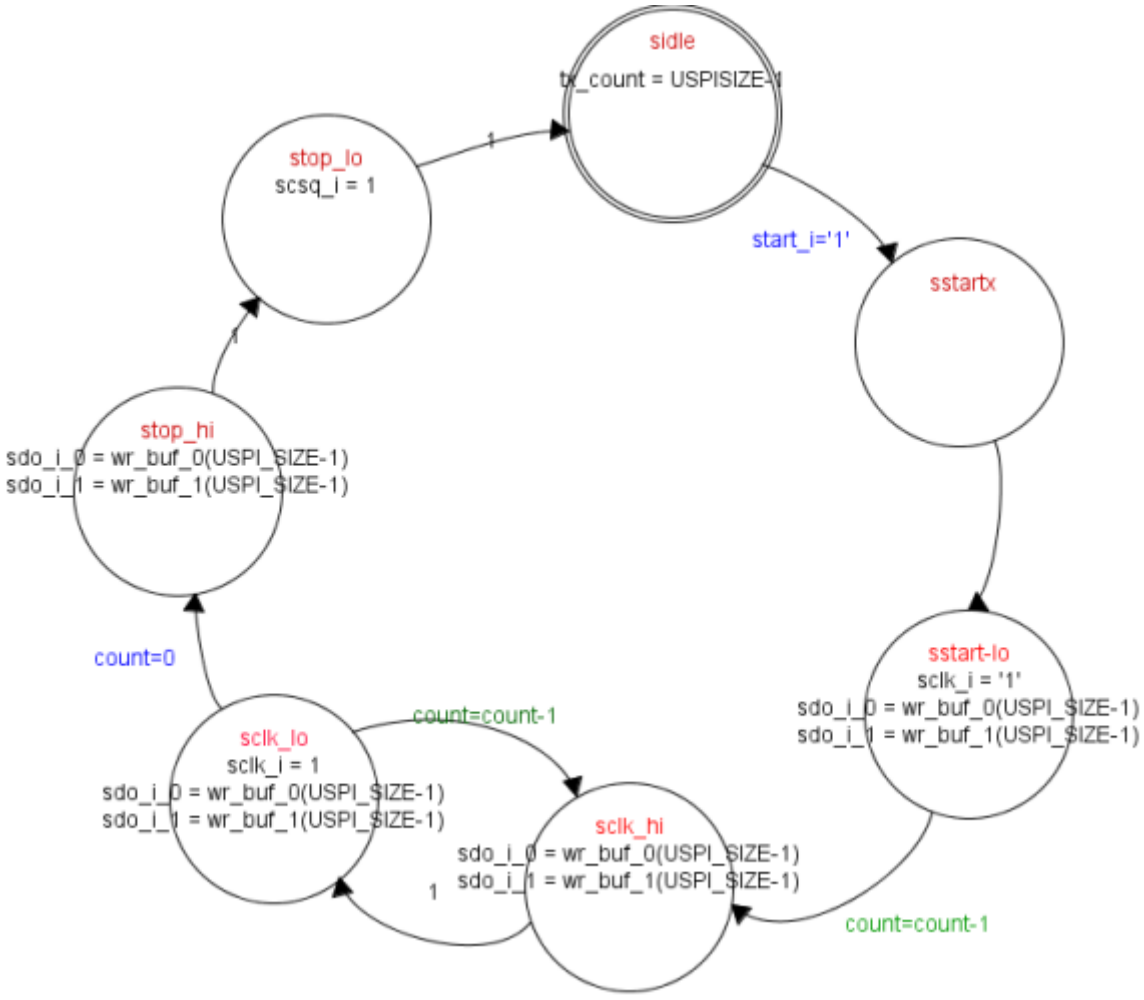


Figure 66: SPI DAC State Machine

- entity statement declares the design name, here it is “spidac”.
- resetn: 1-bit signal to reset the state machine
- bclk: Main system clock

- spi_clkp: Reduced frequency of a main clock signal
- start: 1-bit start signal for data transmission
- done: Acknowledgement signal after data transmission
- scsq: Chip select
- sclk: serial clock
- sdo_0: serial data-0 output port
- sdo_1: serial data-1 output port
- sndData_0: Input data from channel-0
- sndData_1: Input data from channel-1
- In addition to ports, an additional keyword Generic is declared in the entity before the ports. Generics support static information to blocks in a similar way as constants, but unlike the constants, the values of generics can be supplied externally. SPI buffer size is declared as a generic type so that it can be manipulated externally.

7.6.2 State diagram explation

- For an efficient state machine implementation in VHDL it is necessary to have a state diagram.
- The SPI DAC as a state machine is depicted in the image above. All lines join the block (green instructions) are the part of the sequential block while the red instructions are part of the combinational block.
- The default state is sidle when resetn = '0'. next_state will be sstartx only when start = 1. The transition to sstartx will only take place on the rising edge of clock at the transition count.
- At state start_lo, sclk_i = '1' so that at the next rising edge of the clock sclk should be '1'. The same is the case with sdo_i and sdo. However, in the case of sdo_i, the value is msb of wr_buf.
- Transition to sclk_hi at the rising edge will result in a decrement of the count value.
- Now the state machine will toggle between states sclk_hi and sclk_lo until the count becomes '0'. It is necessary to take care of sclk_i value while toggling between these 2 states to make sure that clock will be high only when state is sclk_hi and low at state sclk_lo.
- Each bit transferred will be available at the output both when sclk is high and low.
- Last bit to be transferred is set in sdo_i in state stop_hi.
- After the end of transmission it is necessary to disable the chip select which is initialized in state stop_lo and subsequently at rising edge outputted at state sidle.

7.6.3 DAC Timing Diagram (CPOL=0,CPHA=1)

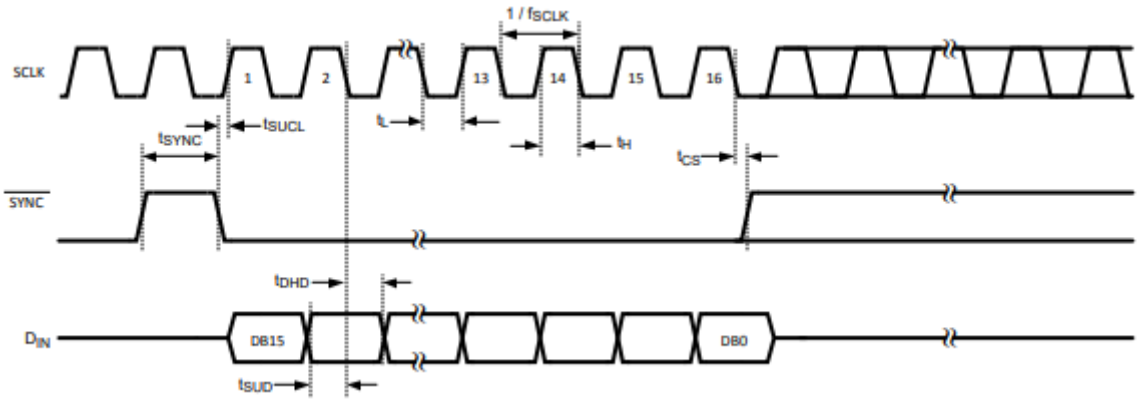


Figure 67: Timing Diagram of DAC121S101

Timing diagram Explanation:

t_{SYNC}	chip select high time
t_{SUCL}	setup time chip selects to SPI clock rising edge
t_L	SPI clock low time
t_h	SPI clock high time
$1/f_{sclk}$	SPI clock cycle time
t_{CS}	SPI clock falls to rise of chip select
t_{SUD}	data setup time
t_{DHD}	data hold time

Figure 68: DAC121S101 Timing Diagram explanation of alphabet

7.6.4 ADC State Machine and Entity declaration:

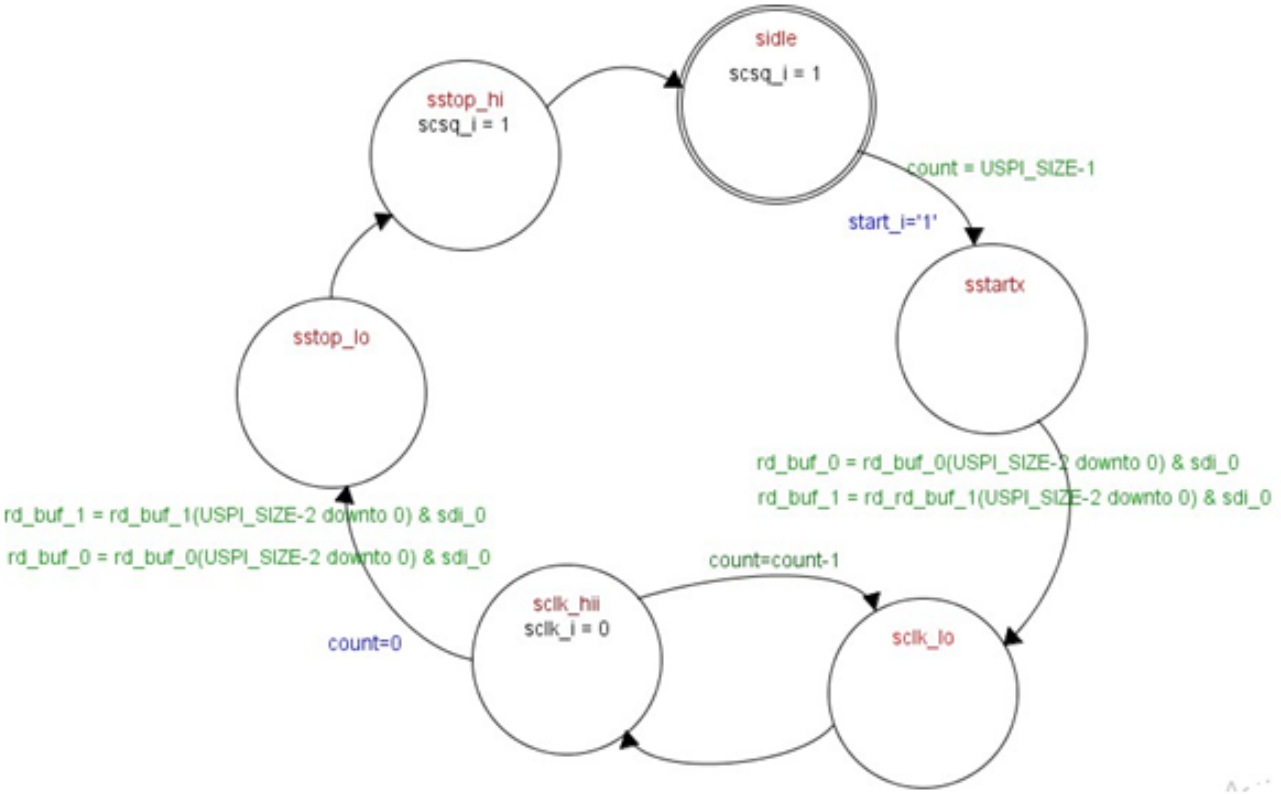


Figure 69: SPI ADC State Machine

The above state diagram depicts the functionality for ADC interface. The entity section will include the following input and output signals:

- resetn: 1-bit signal to reset the state machine
- bclk: Main system clock
- spi_clkp: Reduced frequency of a main clock signal
- start: 1-bit start signal for data transmission
- done: Acknowledgement signal after data transmission
- scsq: Chip select
- sclk: serial clock
- sdi_0: serial data-0 output port
- sdi_1: serial data-1 output port
- rcvData_0: Received data to channel-0
- rcvData_1: Received data to channel-1

7.6.5 ADC Timing Diagram (CPOL=1,CPHA=1)

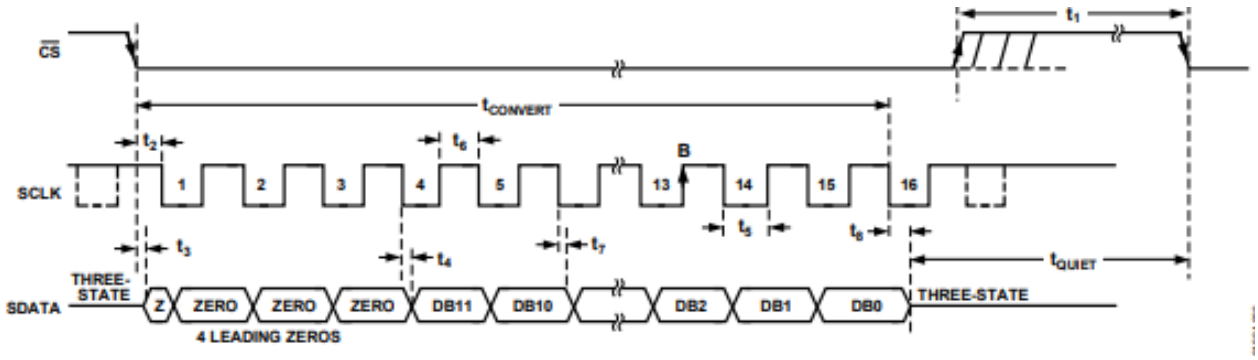


Figure 70: Timing Diagram of ADC7476[1]

t1 : minimum chip select pulse width
t2 : cs to spi clock setup time
t3 : delay from chip select until start SDATA
t4 : data access time after chip select falling edge
t5 : sclk low pulse width
t6 : sclk high pulse width
t7 : sclk to data valid whole time
t8 : sclk falling edge to SDATA high impedance

7.7 Code explanation of DAC and ADC

7.7.1 DAC Code explation

VHDL code for a digital-to-analog converter (DAC) with two lanes. The DAC takes in two 16-bit digital signals, sndData_0 and sndData_1, and converts them into two analog signals, sdo_0 and sdo_1. The code also includes logic for handling the start and stop signals, as well as clock division.

The code begins with the entity declaration, which specifies the inputs and outputs of the DAC. The entity also includes a generic parameter, USPI_SIZE, which is set to 16 by default.

The architecture section of the code defines the behavior of the DAC. It begins by defining two custom types, state_type and sslst_type, which are used to represent the different states of the DAC. The code then declares several signals, including state, next_state, sclk_i, scsq_i, sdo_i_0, sdo_i_1, wr_buf_0, wr_buf_1, count, ssl_state, and sslnxt_state.

The code includes two processes, sslseq_proc and sslcmb_proc, which handle the start and stop logic of the DAC. The sslseq_proc process is sensitive to the rising edge of the

bclk signal and sets the ssl_state signal based on the current state and the sslnxt_state signal. The sslcmb_proc process is sensitive to changes in the ssl_state, state, and start signals, and sets the sslnxt_state, start_i, and done signals based on the current state.

The sseq_proc process handles the SPI logic of the DAC. It is sensitive to the rising edge of the bclk signal and sets the state, count, and several other signals based on the current state and the next_state signal. The scmb_proc process is sensitive to changes in the state, start_i, count, and wr_buf_0, wr_buf_1 signals, and sets the next_state, sclk_i, scsq_i, sdo_i_0, sdo_i_1, signals based on the current state

Overall, this code implements a digital-to-analog converter with clock division and start/stop logic.

7.7.2 ADC code explation

The code is an implementation of an ADC2Lane module in VHDL. The module has a generic parameter USPI_SIZE, which is set to 16. The module has several input and output ports, including resetn, bclk, spi_clkp, start, done, scsq, sclk, sdi_0, sdi_1, rcvData_0, and rcvData_1.

The module has two processes, sslseq_proc, and sseq_proc. The sslseq_proc process handles the start or stop logic and proper handling of clock division. The sseq_proc process handles the SPI logic of the ADC. It is sensitive to the rising edge of the bclk signal and sets the state, count, and several other signals based on the current state and the next_state signal.

The scmb_proc process is sensitive to changes in the state, start_i, count, and wr_buf_0, wr_buf_1 signals, and sets the next_state, sclk_i, scsq_i, sdo_i_0, sdo_i_1 signals based on the current state. The module waits in the ready state until the start signal is asserted, when it latches in the data for the new transaction and advances to the send_data state. In this state, it executes the transaction with the ADC and then returns to the pause state. If the start signal is not deasserted, a new transaction request is latched in and begins immediately once the module is available.

7.8 IP creation for DAC and ADC

```
dacTX : dac2lane
     GENERIC MAP ( USPI_SIZE => 16 )
     PORT MAP (  resetn => S_AXI_ARESETN,
                 bclk => S_AXI_ACLK,
                 spi_clkp => spiclk_p,
                 start => start_dac,
                 done => done_dac,
```

```
                scsq => uDA2csq,
                sclk => uDA2Sclk,
                sdo_0 => usdoDA_0,
                sdo_1 => usdoDA_1,
                sndData_0 => slv_reg1(15 downto 0),
                sndData_1 => slv_reg1(31 downto 16));

adcRX : adc2lane
    GENERIC MAP ( USPI_SIZE => 16 )
    PORT MAP ( resetn => S_AXI_ARESETN,
               bclk => S_AXI_ACLK,
               spi_clkp => spiclk_p,
               start => start_adc,
               done => done_adc,
               scsq => uAD1csq,
               sclk => uAD1Sclk,
               sdi_0 => usdiAD_0,
               sdi_1 => usdiAD_1,
               rcvData_0 => ADC_Data(15 downto 0),
               rcvData_1 => ADC_Data(31 downto 16));
```

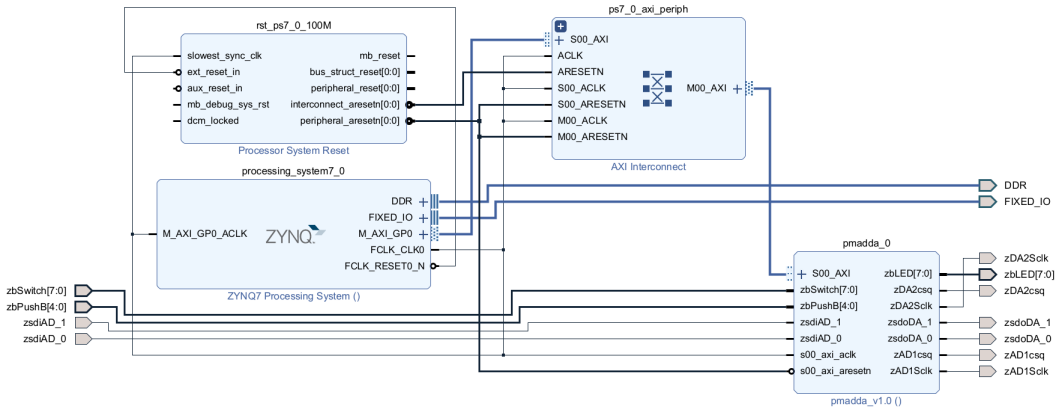


Figure 71: Zynq Ip Block Diagram

This IP block which is in Programmable Logic (PL) which is connected to the processing system (PS) via AXI Interconnect.

Slave Registers have been used for the signals to read/write from the SDK tool for different operations.

1. Slv_reg1 <– DAC input
2. Slv_reg2 <– ADC_DATA .

7.8.1 Embedded Hardware Component

The above-created design is an Embedded Hardware component by Xilinx Vivado IP integrator which provides greater flexibility regarding the kinds of Embedded hardware.

Typically, an Embedded platform created using IP integrator contains

1. One or more MicroBlaze or ARM cortex A9 processors.
2. Volatile Memory internal to the FPGA
3. Volatile (such as SDRAM or DDR) and non-volatile (Flash) memory external to the FPGA.
4. IP controlling External interfaces such as GPIO's UART's or ETHERNET Mac's
5. Debug IP such as Processor trace signals.
6. Other Custom IP created by the Hardware Developer.

The task of the Software Developer is to write software that runs on the processors present on the Hardware Platform to perform application-specific tasks. Xilinx provides drivers for Xilinx IP and some Basic Software components to help accelerate such software development.

7.8.2 FPGA Bitstream

An FPGA bitstream is a file that contains the programming information for an FPGA. A Xilinx FPGA device must be programmed using a specific bitstream for it to behave as an embedded hardware platform. This bitstream is typically provided by the hardware designer who creates the embedded platform.

Programming an FPGA is the process of loading a bitstream into the FPGA. During the development phase, the FPGA device is programmed using utilities such as Vivado or using menu options in SDK. These tools transfer the bitstream to the FPGA on board. In production hardware, the bitstream is usually placed in non-volatile memory, and the hardware is configured to program the FPGA when powered on.

8 Fixed-Point Conversion: Value Scaling Methodology

8.1 Approach

Scaling and fixed-point integer conversion are crucial steps in implementing algorithms on microprocessors or micro-controllers for various reasons. These hardware platforms have limited resources in terms of memory, processing power, and data types. Scaling plays a significant role in optimizing the data for efficient implementation by adjusting its range and precision. The choice of scaling technique depends on the specific requirements of the task and the characteristics of the data. In our case, we employ proportional scaling, where the values are scaled near their mean values to maintain their relative proportions.

Fixed-point representation is an integer-based data type used to represent fractional numbers, which may also be signed. It is a consists of a fixed number of integer digits on the left side of the decimal point and a fixed number of digits on the right side, representing the fractional part. Fixed-point calculations are highly efficient in micro-controller programming as they use a nearly integer representation of the data type. Fixed-point calculation is highly efficient in micro-controller programming because they use a near-integer representation of the data type.

8.2 Distance Measurement subsystem

To ensure accurate representation of the ball's position within the digital system, the voltage values obtained from the analog sensors must undergo precise scaling. This scaling process involves converting the analog voltage values to their corresponding digital representation known as the analog-to-digital value.
Conversion formula

$$ADC = voltage * 3.3/4096 \tag{49}$$

8.2.1 Scaling

Two readings noted by placing the ball to the ceiling and floor. Then analog values converted to digital values and a mean is taken from the values. Calculated mean value used as a subtraction factor from current position in order to have a scaled value in between -357 to 357.

below table contain all the noted and calculated values.

Position	Volts	ADC
Bottom	1.06	1318
Top	1.637	2032
Mean	-	1675

8.2.2 Modeling and Simulation

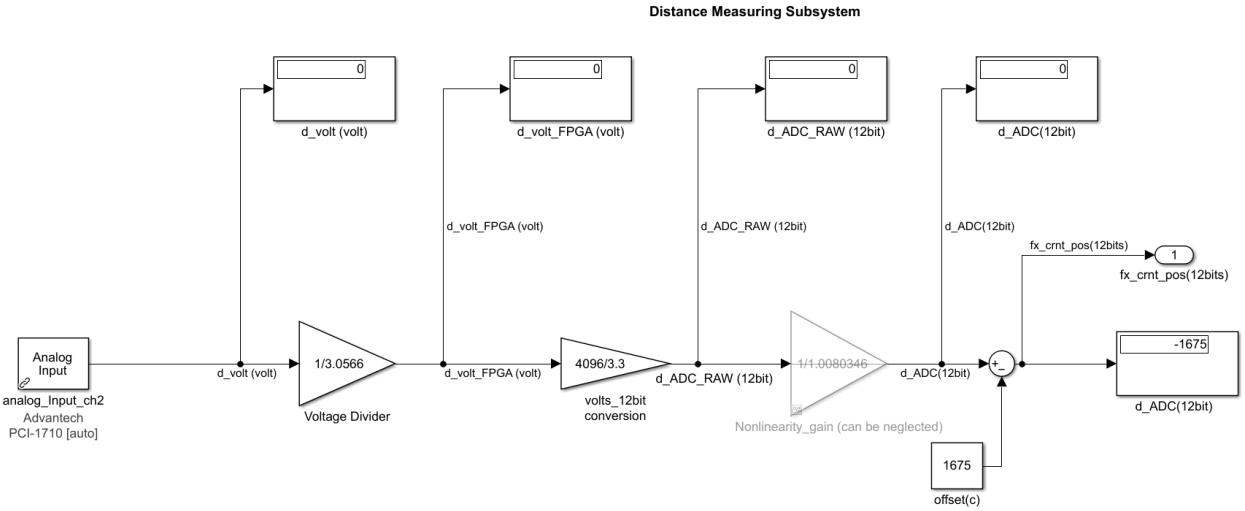


Figure 72: Fixed point distance measurement subsystem

8.3 PID Controller subsystem

In Chapter 6, a comprehensive explanation is provided on the derivation of Kd, Kp, and Ki gains that how lookup table were collected to extract the appropriate gain values. The process of scaling for gains was essential due to the reason the input and output signals were scaled to different positions. Consequently, there was a requirement for appropriately scaled gain values to ensure accurate results corresponding to the scaled input values.To achieve this, all the coefficients were scaled using a factor calculated based on the ratio between the corresponding physical quantity and the ADC value of voltage and current. This factor played a important role in aligning the scaled gains with the scaled input values, ensuring precise and dependable system outcomes.

8.3.1 Scaling

Factor for gain scaling needs corresponding ratio of distance in meter to distance in ADC and Current in ADC to current in Amperes.

for Proportional gain

$$Kps = \left(\frac{delta\_x}{d\_ADC}\right) \cdot kp \cdot \left(\frac{I\_ADC}{I}\right) \tag{50}$$

for derivative gain

$$Kds = \left(\frac{delta\_x}{d\_ADC}\right) \cdot kd \cdot \left(\frac{I\_ADC}{I}\right) \tag{51}$$

for Integral gain

$$Kis = \left(\frac{delta\_x}{d\_ADC}\right) \cdot ki \cdot \left(\frac{I\_ADC}{I}\right) \tag{52}$$

Calculation for the factor

$$\begin{aligned} const &= \left(\frac{delta\_x}{d\_ADC}\right) \cdot \left(\frac{I\_ADC}{I}\right) \\ &= \left(\frac{0.052}{688}\right) \cdot \left(\frac{1977}{4.76}\right) = 0.03139 \end{aligned} \tag{53}$$

Now constant value 0.03139 will be used as a scaling factor for PID gains.

Scaled Proportional gain

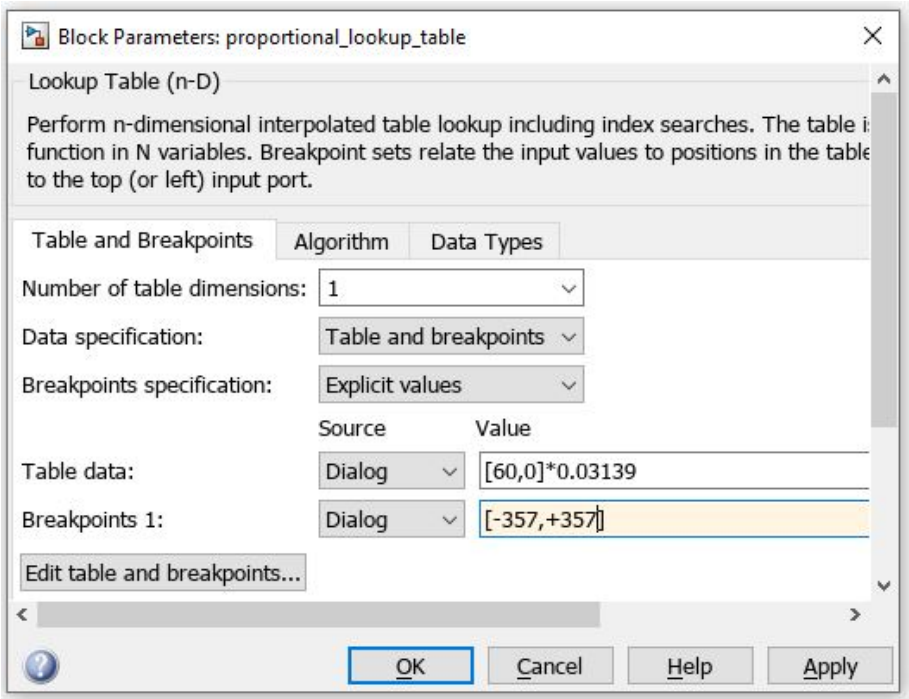


Figure 73: Simulink Block Parameter: Proportional gain

Kp	Position	const	Kps
60	-357	0.03139	1.8834
0	357	0.03139	0

The general equation was derived from the table.

$$Yp = -0.00273764133 \cdot x + 1.8834 \tag{54}$$

Coefficients of above equation were converted into fixed point integers in order to optimize the computation.
So for a fixed point representation of word size 32bit, above coefficient and constant need one bit was allocated for storing the sign, rest store fractional part. Maintaining a consistent representation size for all equations in the code simplified its usage and facilitated ease of handling. for the explanation in equation (57) it explained why we selected 5bits to represent integer and 27bits to represent factional value.

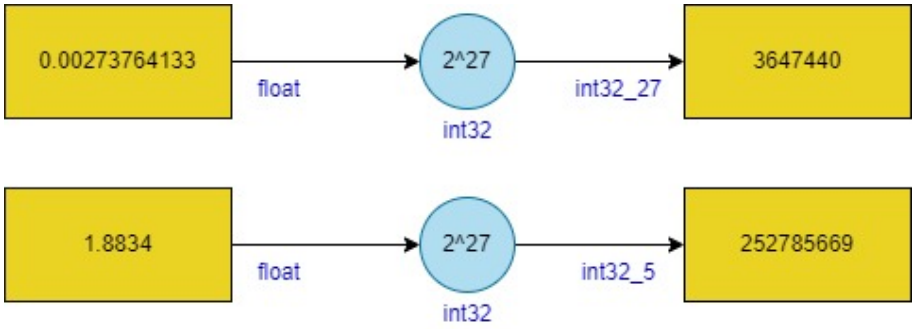


Figure 74: Proportional Equation coefficients to fixed points

$$Yps = -367440 \cdot x + 252785669 \tag{55}$$

Scaled Derivative gain

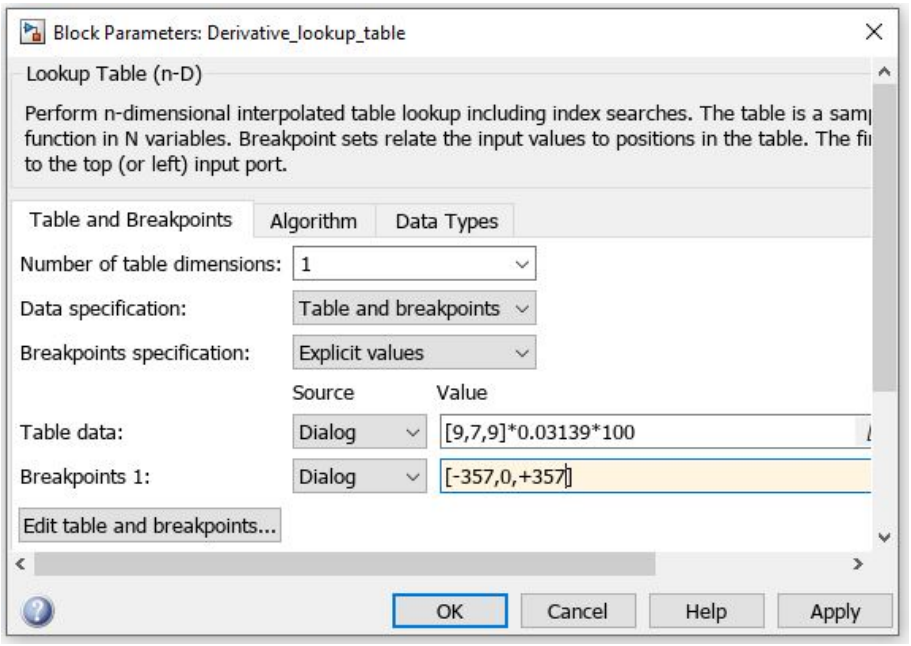


Figure 75: Simulink Block Parameter: Derivative gain

Kd	Position	const	Kds
9	-357	0.03139	28.251
7	0	0.03139	0
9	357	0.03139	28.251

The general equation derived from the table.

$$Yd = +/- 0.08125 \cdot x + 21.97 \tag{56}$$

Above equation coefficients were converted into fixed point integers in order to optimize the computation.
Similarly number of bits needs to define Integer and fractional part within the word length of 32bit, here constant value 21.97 which means it takes 2^5 to represent the integer while 2^{27} can represent fractional bits.

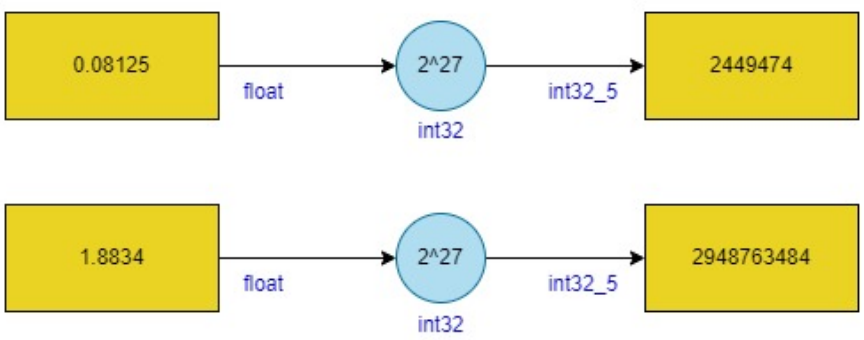


Figure 76: Derivative Equation coefficients to fixed points

$$Yds = +/- 2449474 \cdot x + 2948763484 \tag{57}$$

Scaled Integral gain

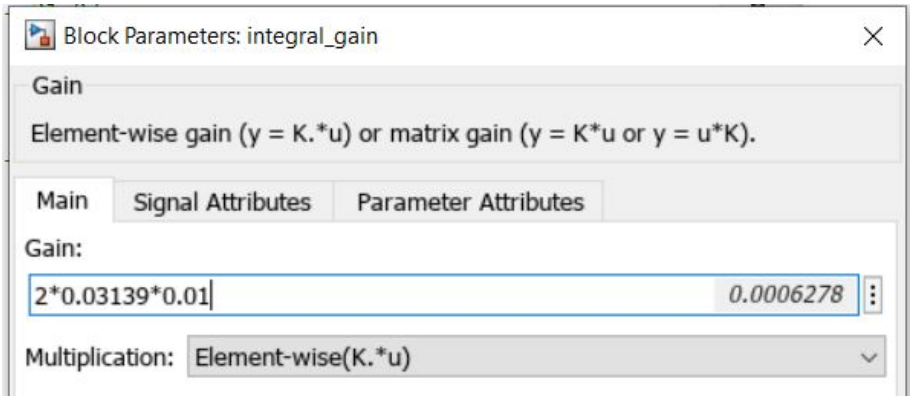


Figure 77: Simulink Block Parameter: Integral gain

Ki	Position	const	Kis
2	-357	0.03139	0.0006278
2	0	0.03139	0
2	357	0.03139	0.0006278

The general equation derived from the table.

$$Yp = 0.0006278 \cdot x \tag{58}$$

for fixed point representation same as for equation (56) and (57), 2^5 represent integer and 2^{27} represent fractional part.

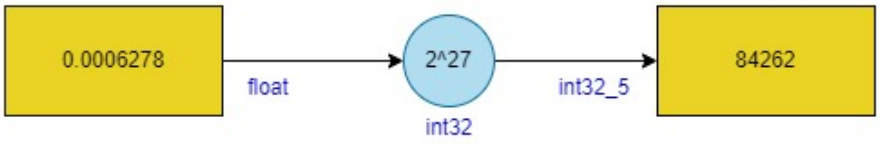


Figure 78: Integral Equation coefficients to fixed points

$$Yp = 84262 \cdot x \tag{59}$$

8.3.2 Modeling and Simulation

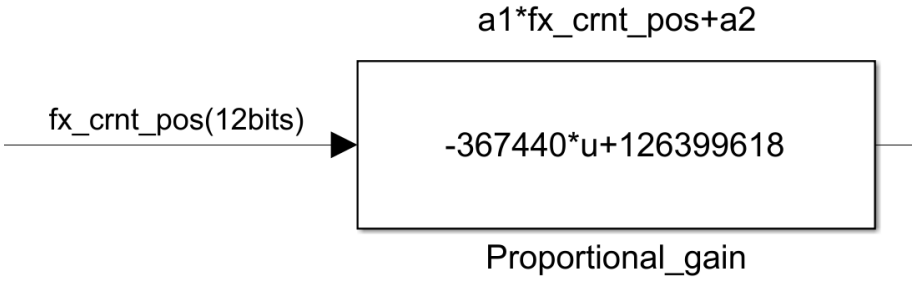


Figure 79: fixed point proportional gain

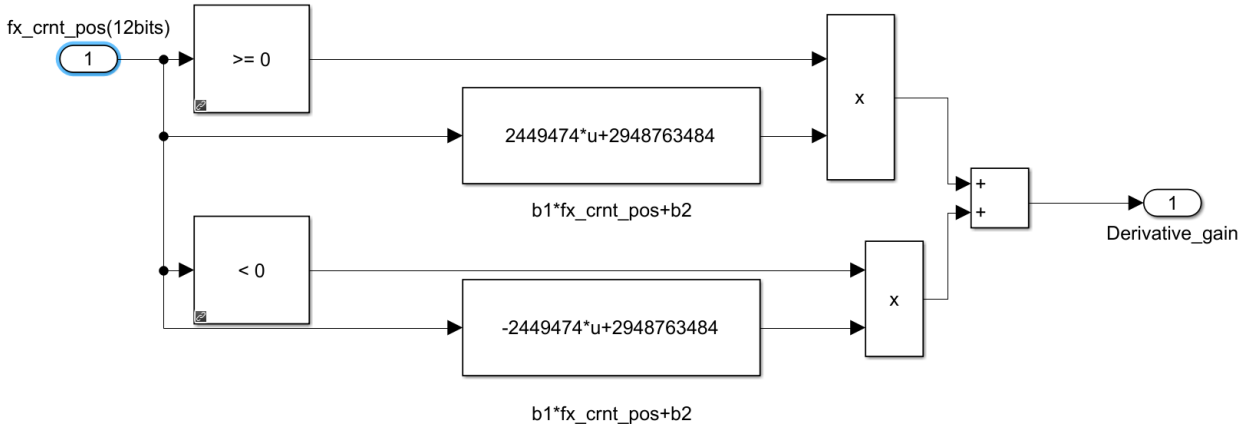


Figure 80: Fixed point derivative gain

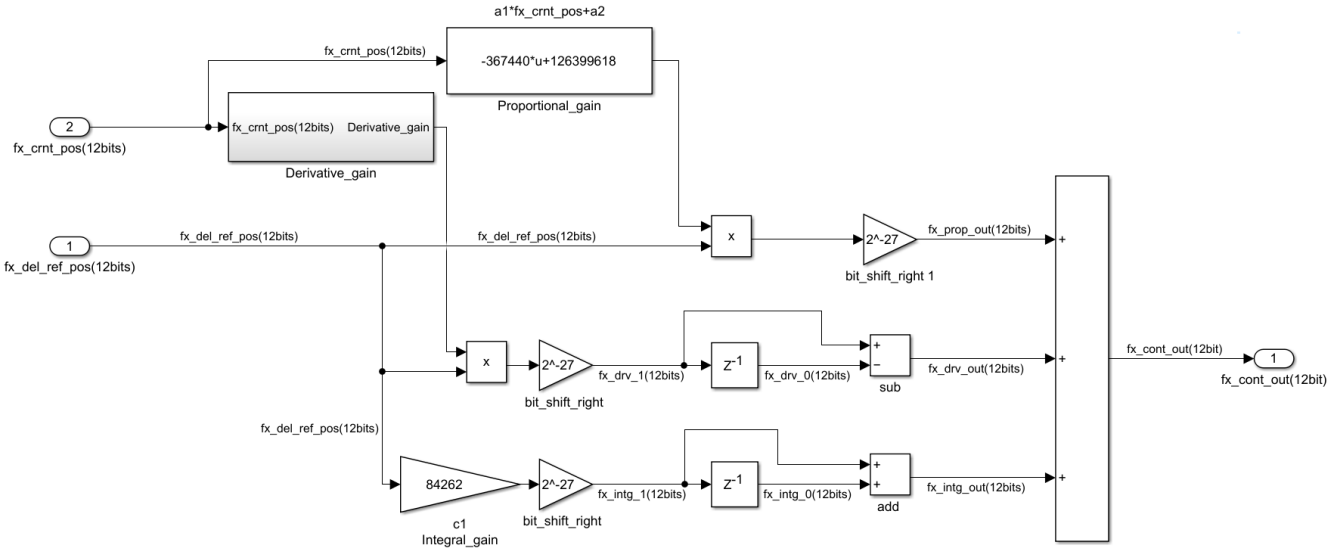


Figure 81: Fixed point controller PID

8.4 Current/Voltage Equilibrium subsystem

As in Chapter 7 discussed that current is used as a controlling factor for Modeling and simulation. However, when dealing with the implementation of the system, using raw current values and directly modeling the current equilibrium can present challenges. As in our case current posses high resolution so it make the conversion more difficult. In order to avoid this conversion from current to voltage equilibrium was employed. By scaling and converting the current equilibrium values into their corresponding voltage equivalents, we achieve compatibility with the modeling framework and facilitate the ease of dealing with voltage-based control mechanisms during the implementation of the system. This approach allows for a more streamlined and consistent integration of the system's control strategies and ensures the effective utilization of voltage as the controlling parameter.

8.4.1 Steps for Scaling Implementation

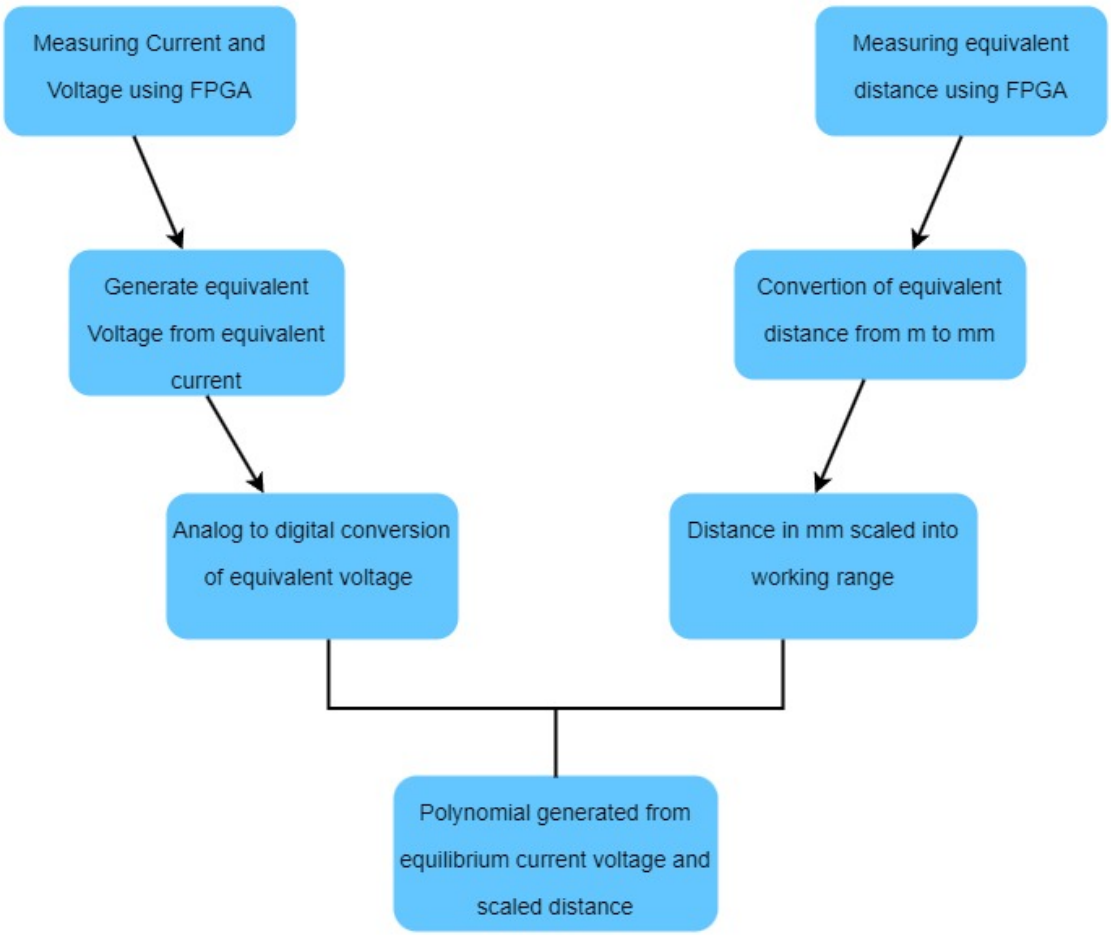


Figure 82: steps for scaling

8.4.2 Scaling

Chapter 4 thoroughly explain how the linear relation for equivalent current and distance derived. For implementation, this relation needed to be optimized. So decisions were made to change control factor from current to voltage.
Values from Table 5 helped to generate a relation to draw voltage equating values of current.

$$V\_eq = ((0.362 * I\_eq) + (-0.1305)) \tag{60}$$

where
V_eq = equivalent voltage
I_eq = equivalent current

Table 6 Equivalent voltage converted into digital representation

$$fx\_eqC\_V = V\_eq * 3.3/4096 \tag{61}$$

where: *fx_eqC_V* = Analog to digital equilibrium current voltage.

Equivalent Distance from Table 7 changed into millimeters

$$dist\_eq\_m = dist\_eq * 1000 \tag{62}$$

where: *dist_eq_m = equivalent distance in mm*

then distance in mm from Table 7 scaled into the working range of +357 to -357

$$fx\_del\_ref\_pos = ((12.916 * dist\_eq\_m - 162.5947) - 1623) \tag{63}$$

where: *fx_del_ref_pos= scaled distance form raw adc*

By considering data set of scaled distance *fx_del_ref_pos* and equilibrium current voltage *fx_eqC_V* generated a new polynomial.

$$f(x) = 0.002831 * x^2 - 2.957 * x + 789.3 \tag{64}$$

$$f(x) = 11874 * x^2 - 12402556 * x + 3310564147 \tag{65}$$

Here 22bit right shifted for fixed point representation in order to compensate the max value of integer in the equation

8.4.3 Modeling and Simulation

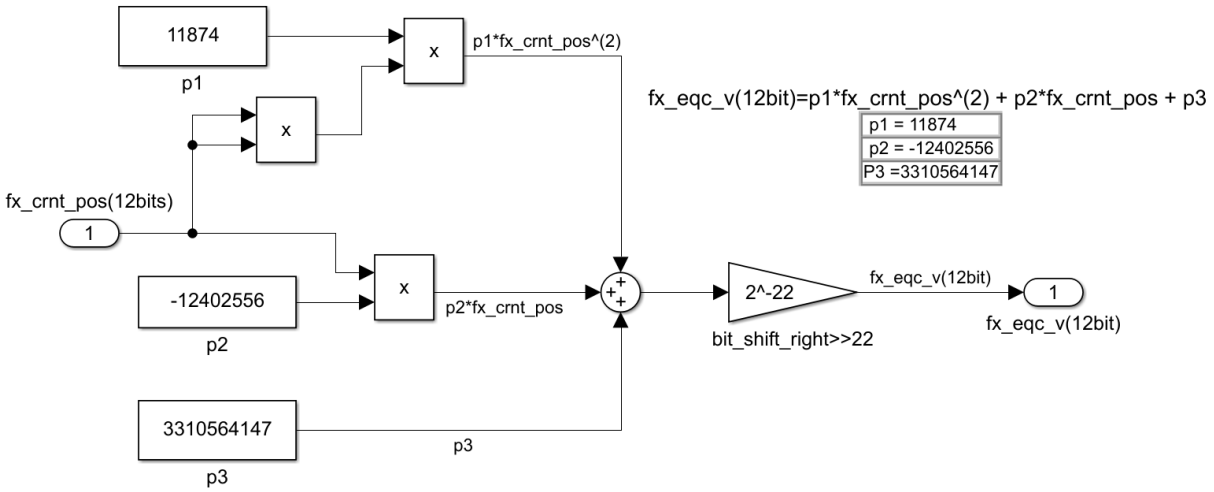


Figure 83: fixed point equilibrium current

9 FPGA Implementation: Integration of C Code

9.1 Xilinc SDK

The Xilinx Software Development Kit (SDK) is an Integrated Development Environment (IDE) that uses Xilinx embedded processors for the development of embedded software applications. SDK works with hardware designs created with Vivado Design Suite. SDK is based on the Eclipse open-source standard. It provides intensive tools that enable software developers to build software applications faster and in a more standardized way. Cloud-native mobile app development, for example, leverages Apple's iOS SDKs or Google's Android SDKs for that platform. It includes complete documentation, libraries, samples of examples etc for developers to use to create embedded software applications. C-based languages use for FPGA design. Specifically, the AMD Vivado compiler provides a programming environment that shares key technology with both standard and specialized processors for the optimization of C and C++ programs. Xilinx is used where high-bandwidth, networking, cloud deployments etc.

Figure 84: Xillinx SDK

9.2 ZYNQ 7000

[16]The ZynqTM-7000 All Programmable SoC redefines possibilities for embedded systems, giving system architects and software developers a flexible platform to launch new solutions while providing traditional ASIC and SoC users with a fully programmable alternative. ARM® Cortex™-A9 processors, integrated with the industry's leading performance-per-watt 28nm programmable logic, achieve power and performance levels exceeding that of discrete processor and FPGA systems. Available in dual-core (Zynq-7000 devices) and single-core (Zynq-7000S devices) Cortex-A9 configurations, the Zynq-7000 family boasts the best price to performance-per-watt in its class, making it the best option for a wide range of embedded applications, including small cell base stations, multi-camera drivers assistance systems, machine vision for industrial automation, medical endoscopy.

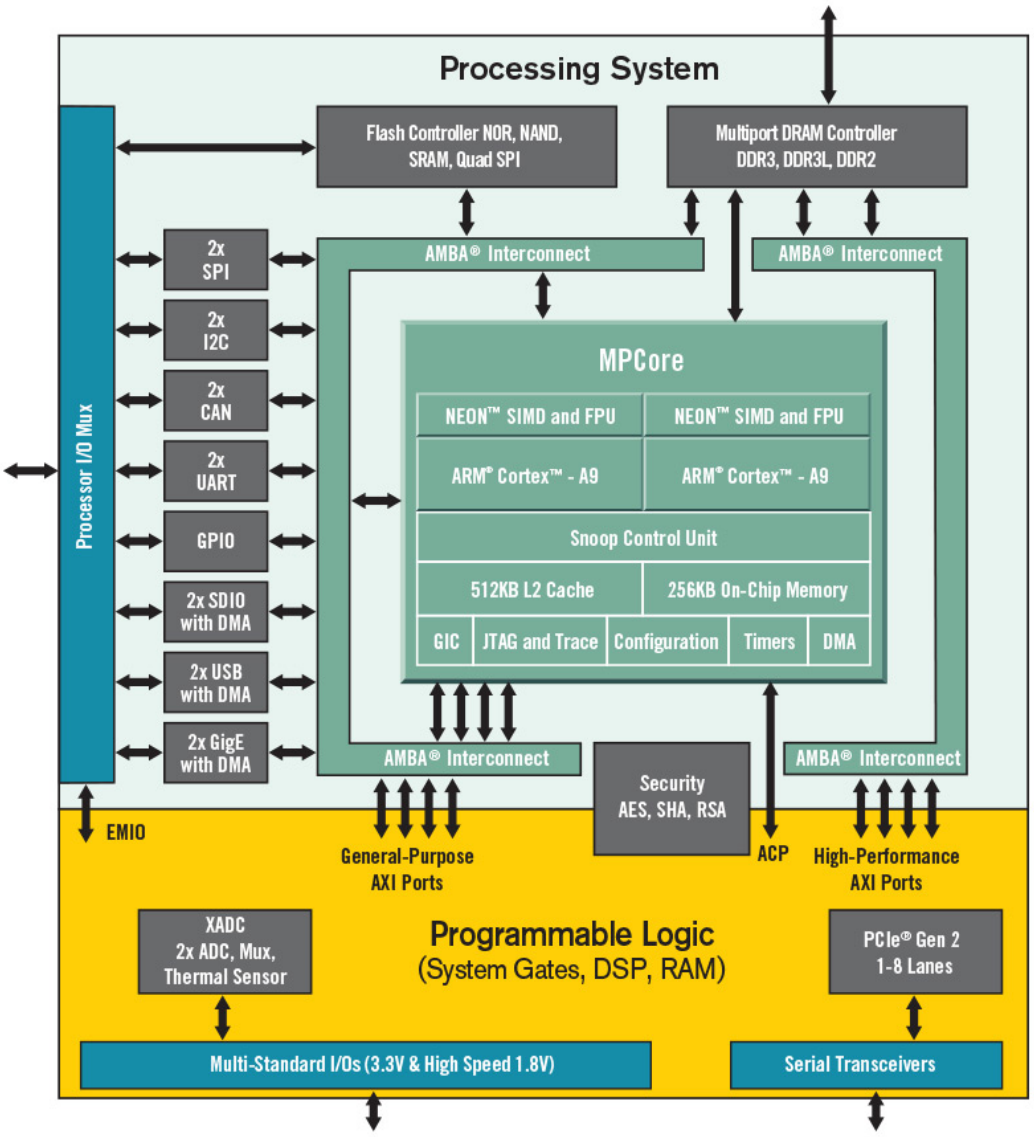


Figure 85: Zynq Microprocessor dual core.

9.2.1 Software and Ecosystem Features

Real-Time Operating Systems Comprehensive collection from open-source to best-in-class commercial operating systems	• FreeRTOS – Ideal for simple, high-performance tasks. • Bare-Metal – Best for high-performance, low-level applications. • Android – For feature-rich, user-friendly graphical applications.
Development Tools SoC-centric tools and familiar environments to develop software and hardware on both the processing system and programmable logic	• AMD Software Development Kit (XSDK) tools – Manage the full development and debug cycle for multiprocessor designs. • SDSoC development environment – Compiles C/C++ applications into an optimized, fully functional Zynq-7000 AP SoC system.
Reference Designs and IP Block Portfolio A solid foundation for value-added custom designs	• AMD-verified reference designs included in hardware development kits. • Extensive Alliance partner reference designs available. • Expansive IP block catalogue for accelerators and peripherals across most application spaces.

9.3 Interrupts on Zynq SOC

In embedded processing, an interrupt is a signal that temporarily halts the processor's current activities. The processor saves its current state and executes an interrupt service routine to address the reason for the interrupt. An interrupt can come from one of the three following places:
• Hardware – An electronic signal connected directly to the processor
• Software – A software instruction loaded by the processor
• Exception – An exception generated by the processor when an error or exceptional event occurs
Regardless Interrupts are of two types mask-able which can be ignored and non-mask-able which can not be ignored which typically use for timers or watchdogs.
Interrupts can also be edge base or level base. The Xilinx® Zynq® -7000 interrupts support configuration of both the interrupts edge base and level base. »»»>«««<

9.3.1 Why do we need Interrupts

Interrupt-based designs are efficient for Real-time designs as normally systems has different number of inputs at a time for example Mouse, Keyboard, sensors or push-buttons etc that will at times require processing. Inputs from these peripherals are asynchronous which is difficult to predict when interrupt is triggered. Interrupts enables processor to change it's status for current and handle the interrupt in that fashion processor can handle them synchronously.

9.3.2 Interrupts Structure in Zynq SOC

As in modern processors there are number of sources interrupts can come from. The Zynq SoC uses a Generic Interrupt Control.

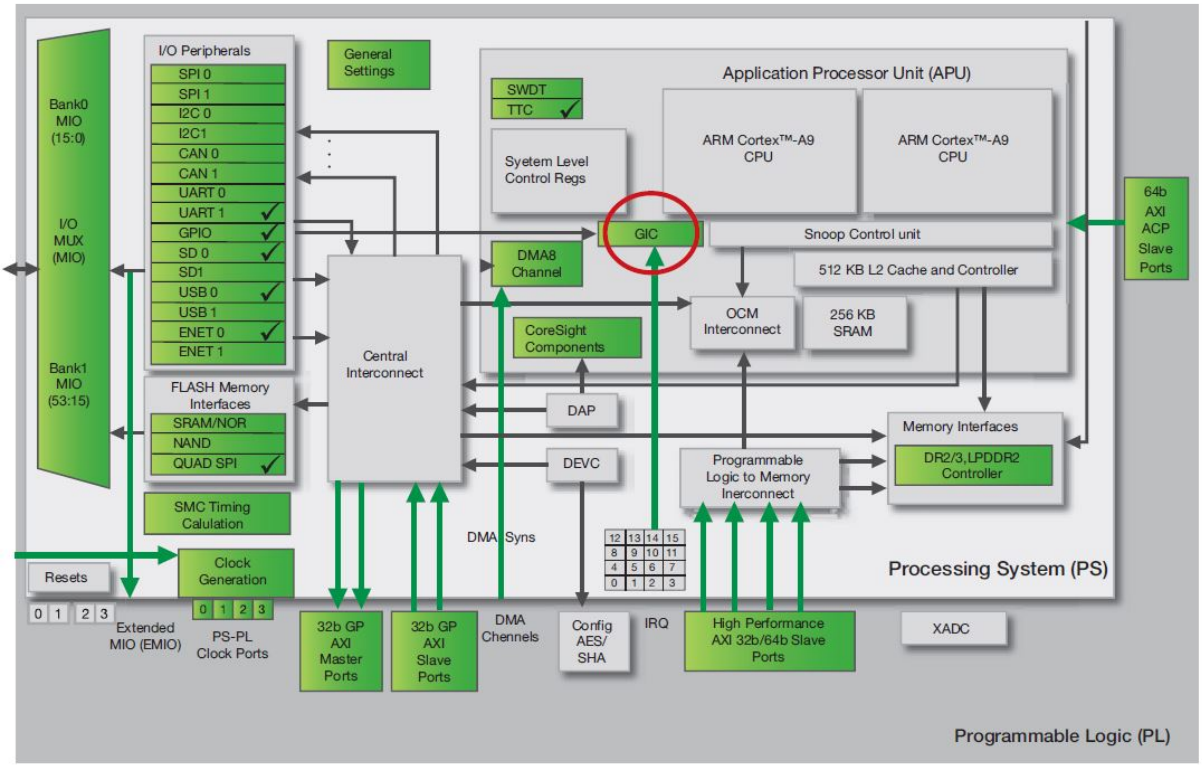


Figure 86: The Generic Interrupt Controller is circled in red.

The GIC can handle interrupts for following sources [19]:
• Software-generated interrupts – There are 16 such interrupts for each processor. They can interrupt one or both of the Zynq SoC's ARM® Cortex™-A9 processor cores.
• Shared peripheral interrupts – Numbering 60 in total, these interrupts can come from the I/O peripherals, or to and from the programmable logic (PL) side of the device. They are shared between the Zynq SoC's two CPUs.
• Private peripheral interrupts – The five interrupts in this category are private to each CPU—for example CPU timer, CPU watchdog timer and dedicated PL-to-CPU interrupt.

9.4 Processing The Interrupts On The ZYNQ SOC

When an interrupt occurs within the Zynq SoC, the processor will take the following actions [19]:
1. The interrupt is shown as pending.
2. The processor stops executing the current thread.
3. The processor saves the state of the thread in the stack to allow processing to continue once it has handled the interrupt.
4. The processor executes the interrupt service routine, which defines how the interrupt is to be handled.
5. The processor resumes operation of the interrupted thread after restoring it from the stack.
Because interrupts are asynchronous events, it is possible for multiple interrupts to occur at the same time. To address this issue, the processor prioritizes interrupts such that it can service the highest-priority interrupt pending first. To implement this interrupt structure correctly, we will need to write two functions: an interrupt service routine to define the actions that will take place when the interrupt occurs, and an interrupt setup to configure the interrupt. The interrupt setup is a reusable routine that allows for constructing different interrupts. Generic for all interrupts within a system, the routine will set up and enable the interrupts for the general-purpose I/O (GPIO).

9.5 Program General Workflow

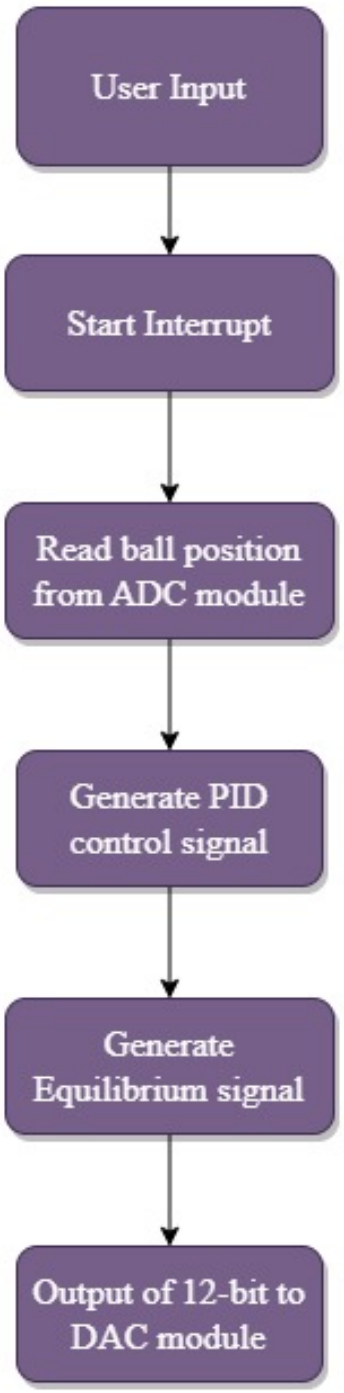


Figure 87: Program General Workflow

9.6 Interrupt Handler

9.6.1 Coding Methodology

For Timer count value calculation:

ARM cpu clock = 666.666 MHz
The timers clock is half of the ARM cpu clock

$$ARMcpuclock/2 = 666.666MHz/2 = 333333000 \tag{66}$$

Then the timer count value can be calculated using following equation:

$$Timercountvalue = sampletime * timerclock \tag{67}$$

$$= 10 * 10^{-}3 * 333333000 = 3333330$$

Sample time for the system is 10ms. Hence, Timer count value will be 3333330.

Step 1: set Interrupt service routine for every 10ms and count calculated in equation (67) 3333330 is set.

Step 2: Switch cases for ISR modes

Interrupt service routine consist of following four modes:

ISR OFF MODE: to shut down the system by raising a terminate flag, rest will be handled in main body.

ISR START MODE: to send the current mode to the RUN mode.

ISR RUN MODE: consists of all the operation of calling functions and calculations.

ISR STOP MODE: Stop the interrupt timer and turn of levitation by sending Low signal to the Digital to analog converter.

Default: set to OFF Mode

Step wise workflow for **ISR RUN Mode**

Step 1: GetAD1() - Read current position from sensor.

Step 2: `DV_lookuptable()` - Scale the current position.

Step 3: Controller() - Calculate Control signal.

Step 4: `eqli_IV ()` - Find Equivalent Voltage value.

Send output to Digital to analog converter.

A threshold value of 2606 which represent max Voltage 2.099V or maximum current 6.160A is set to prevent over current to the coil.

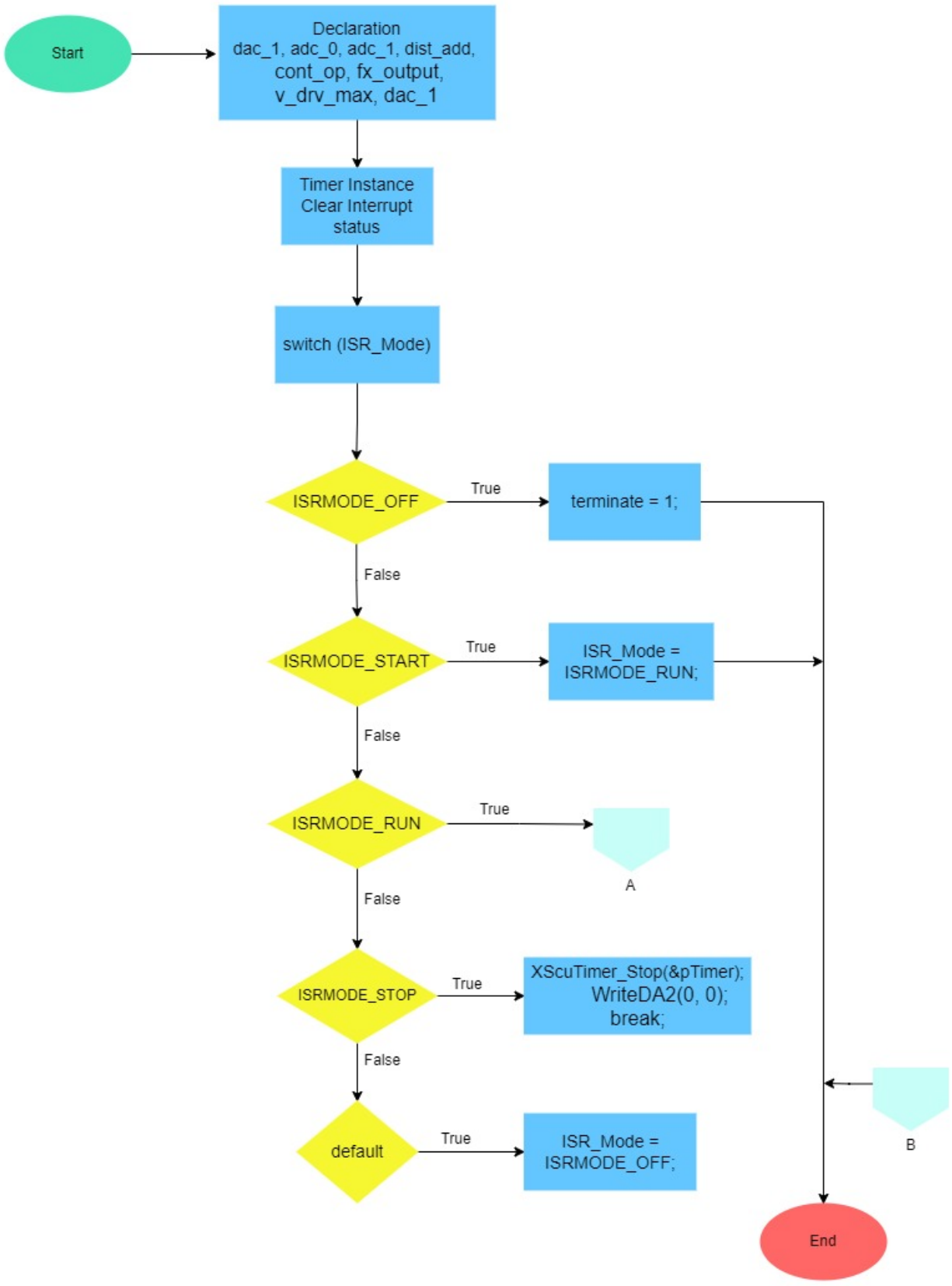


Figure 88: Interrupt Service Routine

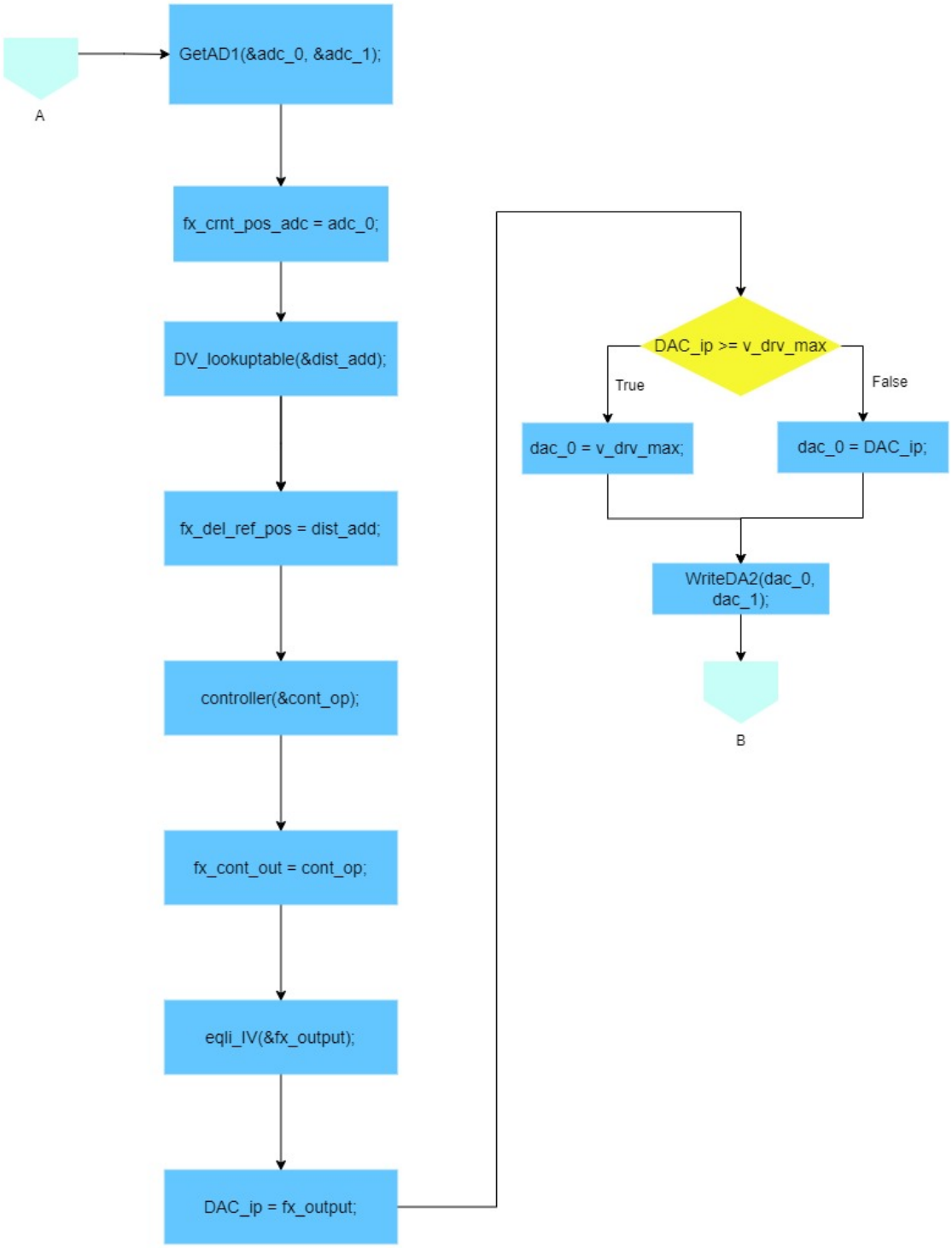


Figure 89: Interrupt Service Routine - Run mode

9.7 Distance Scalling

9.7.1 Coding Methodology

In section 1.2 explained how midpoint “1675” is achieved. Here we are simply subtracting the midpoint value from current position which is in 12bit, then the Scaled value passed to the pointer.

So, by passing pointer to the function will be easier as any changes from the function will be stored on the address of passed variable. The same approach is applied in throughout the source code where needed.

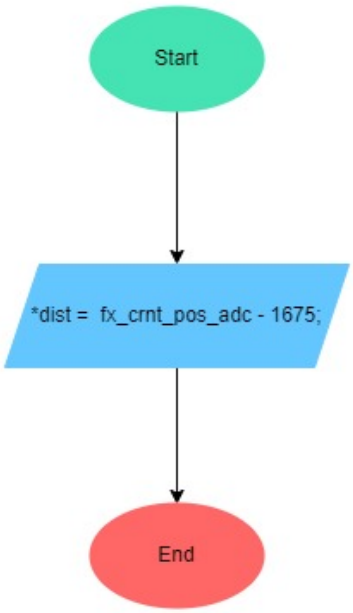


Figure 90: Distance Scaling Function

9.8 Sign generator w.r.t Distance Function

9.8.1 Coding Methodology

Function sign generator w.r.t distance:

Step 1: ternary condition which performs AND bit operation with

distance & 10000000000000000000000000000000

return 0 if negative or 1 if positive. Initializing decision variable with -1 will change it to 1 if condition is true.

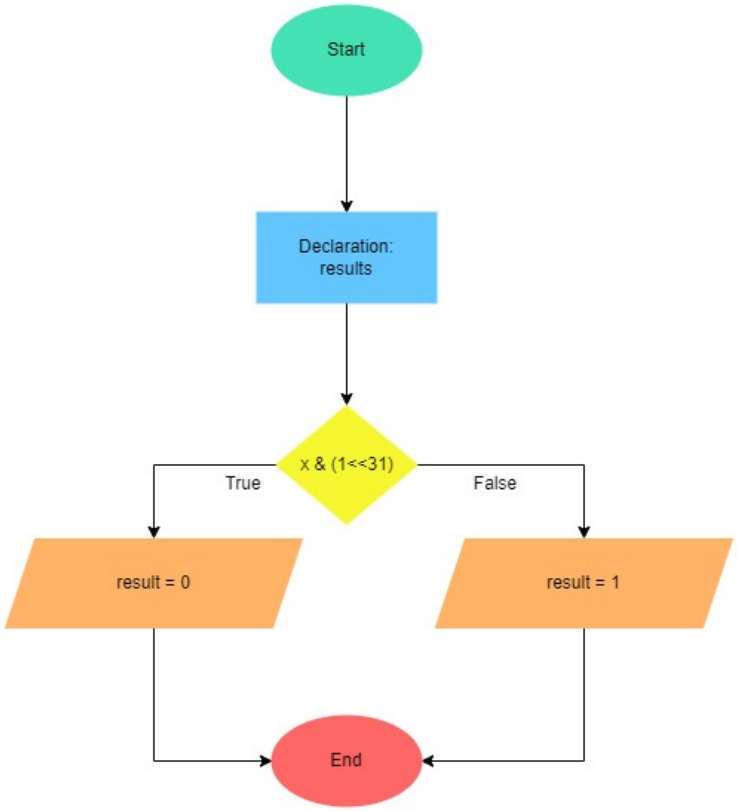


Figure 91: Sign generator w.r.t Distance Function

9.9 Control signal Generation

9.9.1 Coding Methodology

Section 7.3 explain how proportional, derivative and integral gains are scaled and Polynomials are generated. following steps taken in the function:
Step 1: error calculation.
reference position subtracted from current position.
Step 2: Proportional calculated.
Proportional = (a * current position + b) * error » 27 prevent overflowing.

For derivative and Integrals either difference between last error and error can be done calculated first then multiply with gains Kd/Ki respectively or Kd/Ki gain multiply with error then take difference and store the current value in variable fx_derv_0/fx_intg_0 respectively.

Step 3: function pos_neg called for the sign of decision variable.
Step 4: Derivative calculated with current position.
derivative = (+/-1 * a * current position + b) * error » 27 prevent overflowing.
Step 5: Derivative calculated by subtracting derivative with current and last position.
Derivative = with current error - with last error value
Step 6: Integral calculated with current position.
integral = a * error, then 27bits right shift
Step 7: Integral calculated by adding integral with current and last position.
Integral = with current error + last error
Step 8: For PID Proportional, Integral and Derivative are added.
PID = Proportional + Derivative + Integral
Step 9: Current value of integral and derivative is stored as last error values.
last derivative = derivative with current position
last integral = integral with current position

Figure 92: Controller Function

9.10 Equilibrium Current/Voltage

9.10.1 Coding Methodology

In above Section 1.4 polynomial coefficients are explained.

Polynomial equation for Equilibrium voltage is simply calculated

$Polynomial = a * currentposition^2 + b * current\ position + c \gg 22$
$>>$ 22 right shift to stop overflowing as explained in Section 1.4.

Equivalent value is added with the control signal and offset value.

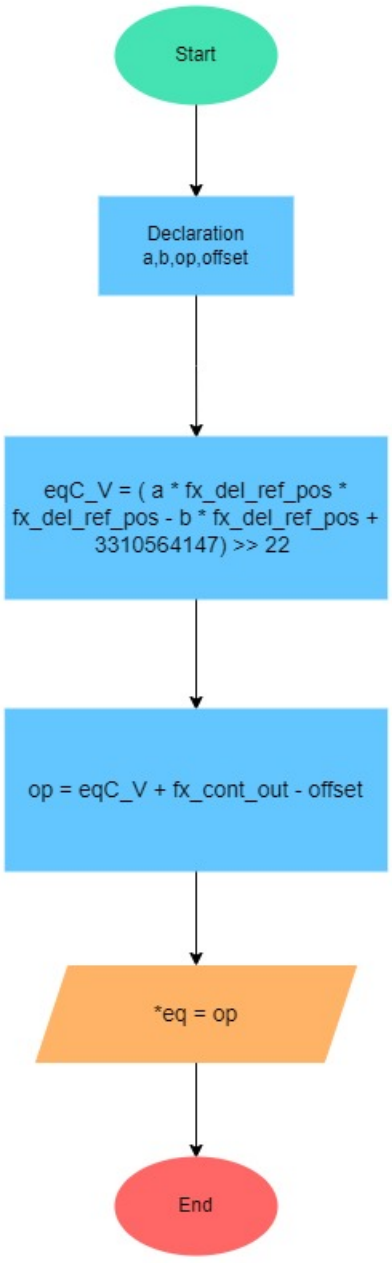


Figure 93: Equilibrium signal generator Function

9.11 Program Main Body

9.11.1 Coding Methodology

In a processor like the ARM Cortex-A9, all interrupts are managed by, and connected via, the general interrupt controller. So, created in header section instances of the general interrupt controller (GIC) and the timer and set the timer count value.

Step 1: Initialize the exception handling features on the ARM processor. Then configure and initialize the general interrupt controller (GIC).

When an interrupt occurs, the processor first must interrogate the interrupt controller to find out which peripheral generated the interrupt. Xilinx provides an interrupt handler to do this automatically, and it is called "XScuGic_InterruptHandler".

Step 2: Connect it to the interrupt controller.

Step 3: to configure and initialize the SCU timer and load the timer count value into it.

Step 4: assigned our interrupt handler, which will handle interrupts for the timer peripheral. In our case, the handler is called "my_timer_interrupt_handler". It's connected to a unique interrupt ID number which is represented by the "XPAR_SCUTIMER_INTR".

Step 5: we enabled the interrupt input for the timer on the interrupt controller. Next, we enabled the interrupt output on the timer.

Step 6: we enabled interrupt handling on the ARM processor.

Then a do while used to keep the process in loop.
Step 7: In the do block we assigned following user input conditions:

- x - shut down: assign ISR_MODE = ISRMODE_OFF
- d – current position info: printing current position as ADC, Scaled value, DAC input
- s – start: assign ISR_MODE = ISRMODE_START
- p - stop: assign ISR_MODE = ISRMODE_STOP
- r - input reference value: update reference value to user I/P value

Step 8: when the condition in while goes false:

- Sent zero to DAC pins.
- Stop the timer
- Disable exception handling, Interrupt handling and ScuGic.

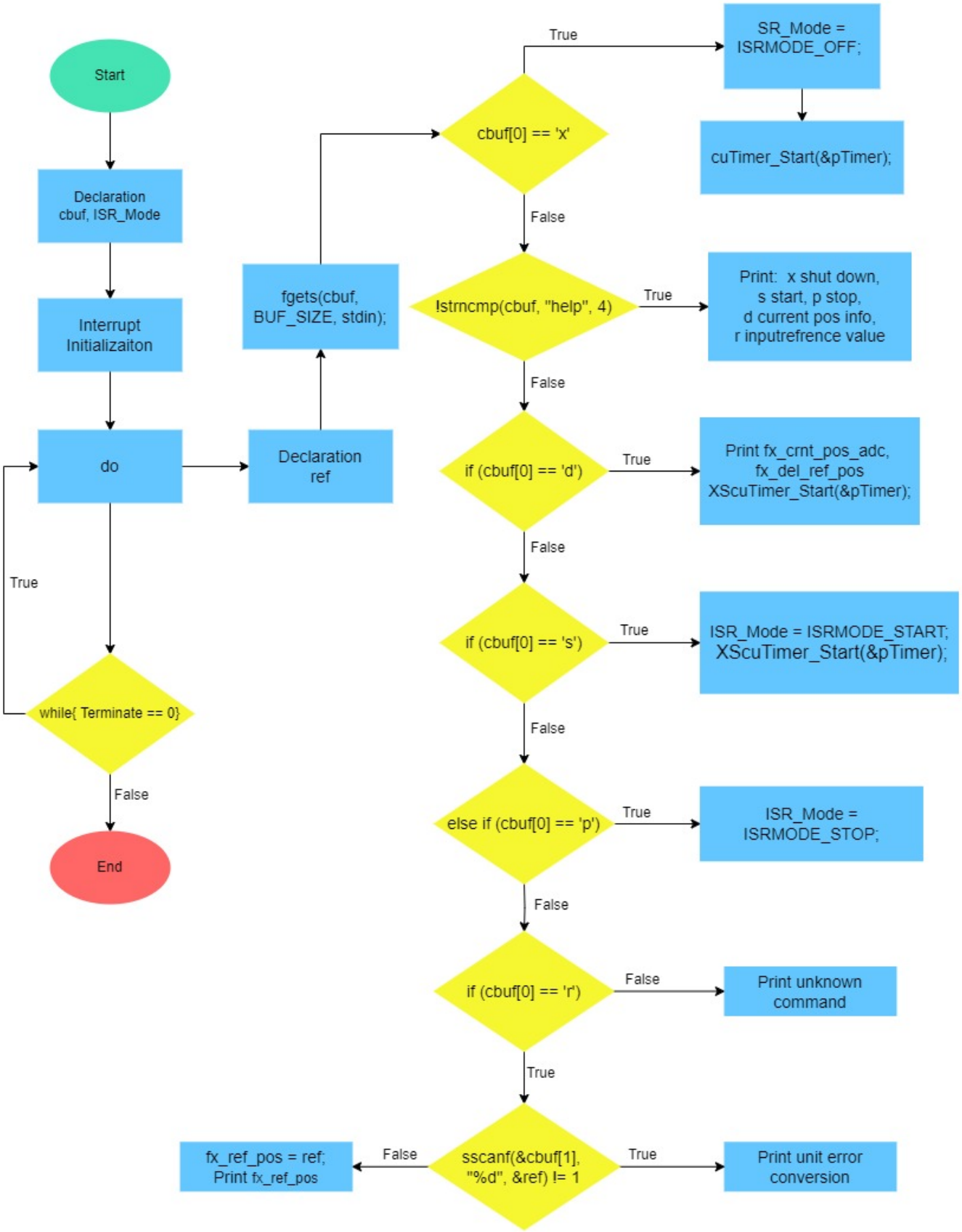


Figure 94: Main body

10 Graphical User Interface Design - App Designer

10.1 Introduction

The field of Human-Computer Interaction encompasses various aspects, including Graphical User Interfaces (GUI). A Graphical User Interface serves as an intuitive and user-friendly platform, bridging the gap between hardware and software components.The HCI is highly influenced by three factors User Expectations, Hardware/Software Limitations and the bidirectional flow of data. The user interface developed for the Magnetic Levitation System is prototype in MATLAB App Designer along with all the requirements and functions. The GUI handles crucial functions such as Input Signal Selection, Data logging and (Target) Application Selection.

10.2 GUI Requirements

The major requirements for the GUI has been decided in the beginning of the project. A prototype GUI was developed to conceptualise the basic requirements of the GUI and layout best suited for user experience.

Requirement	Reference	Requirement Name	Type	Description
REQ-20	GR-1			For communication between FPGA and PC a serial communication with 115200 bits/s shall be used
REQ-21	GR-2			For connecting the FPGA board and ML PMOD modules shall be used which provide ADCs/DACs.
REQ-22	GR-3	Target Selector /	F	Select from a list of system independent available serial ports / Real-time Target
REQ-23	GR-4	Start Button	F	Enabling VIO converter and control logic. Also start the levitation of the ball.
REQ-24	GR-5	Connect Button	F	Enabling the communication between FPGA and PC.
REQ-25	GR-6	Stop Button	F	The Stop Button should force disable the VIO converter as well as stop the communication between FPGA and PC

REQ-26	GR-7	Position Control Slider	F	The desired position can be chosen from +25mm to -25mm on a sliding bar. The desired position should be displayed in the textbox besides.
REQ-27	GR-8	Trapezoidal Wave	F	A Trapezoidal wave generator should generate reference signal of amplitude ±15mm with user changeable on time and off time between 2sec-10sec.
REQ-28	GR-9	Sine Wave	F	Generate a Sine wave with amplitude ±15mm and user changeable period between 2sec - 10 sec.
REQ-29	GR-10	Triangular Wave	F	Generate a Triangular wave with amplitude ±15mm and user changeable period between 2sec - 10 sec.
REQ-30	GR-11	Download Log	F	Enables the data logging and creates a EXCEL file of Displacement, Reference position, Controller Current. The excel file should be named as ddmmyy format
REQ-31	GR-12	Plot log	F	Plot the logged data and automatically adjust the Time axis for logged time duration.
REQ-32	GR-13	Clear Log	F	Clear the Graph
REQ-33	GR-14	Graph Window	F	Plot the current position, Reference position on the graph window where the time is scrolling from 0 to 20 sec.
REQ-34	GR-15	Animation Pane Ball		Animate the Position of the ball
REQ-35	GR-16	Animation Pane Reference	F	Animate the Reference Position as a Marker

10.3 MATLAB App Designer

Selection of GUI Development tool was a conflicting topic because MATLAB GUIDE has stopped supporting the GUIDE Application Development in future release since it was

successfully replaced by MATLAB APP Designer. MATLAB App Designer uses a IBM standard called IBM System Application Architecture Common User Access Advanced Interface Design Reference (SAA CUA) (IBM, 1991). This standard brakes objects into three meaningful classes : Data, Container and Device.

Data : This object represents information. The information could be either text or graphical format.normally appears in the body of the screen. It is, essentially, the screen-based controls for information collection or presentation organized on the screen.

Container : Container objects are objects to hold other objects. They are used to group two or more related objects for easy access and retrieval. There are three kinds of container objects: the workplace, folders, and work-areas. The workplace is the desktop, the storage area for all objects. Folders are general-purpose containers for long-term storage of objects. Work-areas are temporary storage folders used for storing multiple objects currently being worked on.

Device : This are the physical object present in real world such as Display, Console, Buttons Dials etc.

Some Common Terminologies in Object Oriented GUI development

Commands : Commands are actions that manipulate objects. They may be performed in a variety of ways, including by direct manipulation or through a command button. They are executed immediately when selected. Once executed, they cease to be relevant. Examples of commands are opening a document, printing a document, closing a window, and quitting an application.

callback : A callback is a function that executes when a user interacts with a UI component in the app. You can use callbacks to program the behavior of your app. For example, you can write a callback that plots some data when an app user clicks a button, or a callback that moves the needle of a gauge component when a user interacts with a slider. Most components have at least one callback, and each callback is tied to a specific interaction with the component. However, some components, such as labels and lamps, do not have callbacks because those components only display information.

Example of Defining a Callback :

```
methods (Access = private) % call
    function mybuttonpress(app,src,event)
        %comands...
    end
end
```

Function handles : handles provide a way to represent a function as a variable. The function can be either a local or nested function in the same file as the app code, or a function defined in a separate file that is on the MATLAB path. To create the function handle, specify the @ operator before the name of the function. A benefit of specifying callbacks as function handles is that MATLAB checks each callback function for syntax errors and missing dependencies when you assign it to the component. If there is a

problem in a callback function, then MATLAB returns an error immediately instead of waiting for the user to trigger the callback. This behavior helps you to find problems in your code before the user encounters them.

Example of Defining a Function Handle :

```
fig = uifigure('Position',[500 500 300 200]); % Creates Figure
btn = uibutton(fig,'ButtonPushedFcn',@buttonCallback); % handle
    @buttonCallback

function buttonCallback(src,event)
    disp('Button pressed');
end
```

10.4 Reference Libraries

10.4.1 MATLAB Simulink (desktop) Real-time

Simulink Real-time is a Library of tools and pre developed blocks which transforms simulation into Rapid Control Prototyping and Hardware in Loop. This library allows to connect Processor with Real-time Hardware such as Speed-goat controllers as well as Advantech, HumuSoft and National Instrument's (PCIe) IO Cards. Few major features of Simulink (Desktop) Real-time.

- Close Loop Execution of Simulink model
- Signal visualisation and parameter tuning
- High speed 1kHz Sampling in "Connected IO" mode and 20kHz Sampling in "Run in Kernel" mode.

In the Implementation of Control of Magnetic Levitation **Connected IO** Mode is used to interface proposed controller with hardware. The two main features of Connect Mode are.

Signal acquisition — Capture and display signals from your real-time application while it is running. Simulink retrieves signal data from the I/O driver and displays it in the same Scope blocks that you used for simulating your model in non-real time

Parameter Tuning - Change parameters in your Simulink block diagram and have the new parameters take effect in your Simulink model in real time. The effects then propagate through the I/O driver to the hardware.

Both the methods, "Connect IO" and "Run in Kernel" are suspected to missing of clock Ticks which may be a side effect of Complexity of the Model, Process Contention or Implementation of variable Step solver. The best way possible is o optimise the model for Simulink Real-time through profiler as well as use optimised fixed step size.

10.4.2 Selection of Appropriate Real-time Mode

Although with a great sampling frequency of 20 kHz The Kernel Mode is not selected on purpose because of the following limitations in the kernel Mode.

- Run in Kernel mode doesn't support Blocks which are not supported for code generation as well as TO FILE block
- Implementation of S-Function can go tricky since some important C-Functions such as File I/O (fopen), Process Management (spwan,exit etc), Signal and Exceptions handling functions (signal,longjmp,try etc) are not supported.
- In order to generate a code a discrete time with fixed step solver must be used.
- Although the Advnatech PCI 1710U supports sampling rate of 100kHz (varies based on gains) the sampling frequency selected for system was 100 Hz hence the "Connect I/O" mode is most preferred.

During the initial phase, after consulting with C Programmer Mahad, we both considered incorporating additional functionality into the code using the aforementioned function. Consequently, I have decided against utilizing the "Run in Kernel" mode.

10.5 User Interface Layout - A Design Thinking

Initially a user interface sketch is developed in MATLAB GUIDE environment. The below picture represent the basic structure of the GUI components. To simplify the procedure Group of GUI elements called "Containers" were developed.

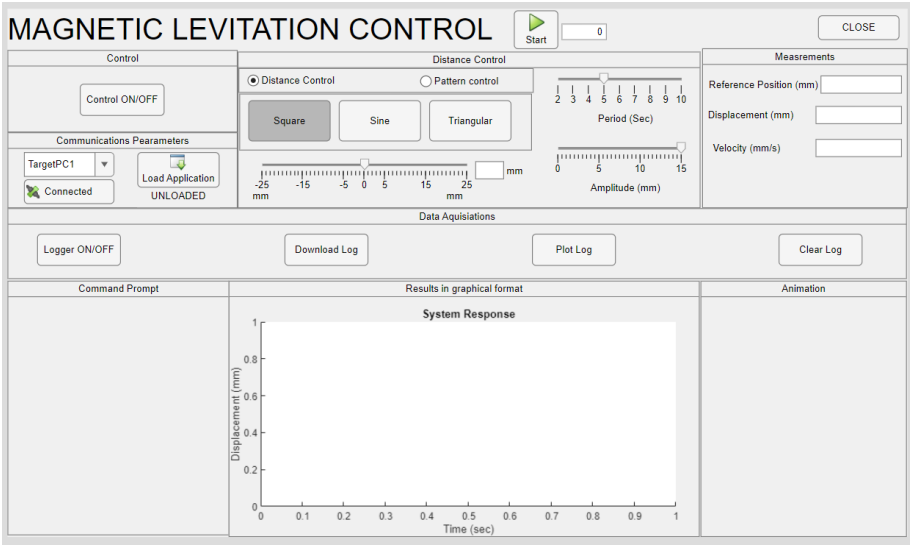


Figure 95: Initial Version of GUI

10.6 Design Waveform Generator

To assess the functionality of the system under development, a standard waveform was created as a reference for its operation. The system was tested using fundamental waveforms like Sinusoidal, Triangular, and Trapezoidal, while avoiding waveforms such as Square, Pulse, and Saw-tooth due to their sudden changes in the reference value. The waveform generator was implemented within a Matlab Model, utilizing a Multi-port Switch to select the desired waveform. Developing the waveform generator directly in the Matlab model offered advantages in terms of step-by-step development and mitigating potential run-time issues in the main function. In contrast, developing it in the GUI could have led to increased clutter and decreased run-time efficiency.

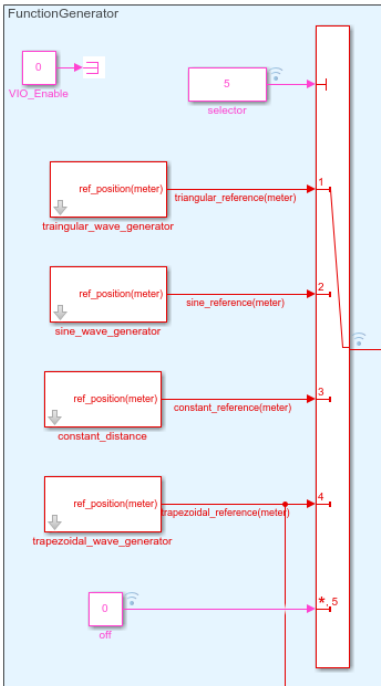


Figure 96: Model Based Development of Full Waveform Generator

10.6.1 Sine Wave Generator

Model Based Development of Sine Wave To generate a sine waveform with variable amplitude and period, we utilized the "Simulink Sine Wave" block in our system. This block had a sample time of 0.01 seconds. To provide flexibility in adjusting the amplitude and period, the block was masked to include dials for these parameters.

To ensure dynamic control over the waveform characteristics, we assigned variables, such as frequency (rad/sec) and amplitude (meter), to the corresponding parameters of the masked "Simulink Sine Wave" block. These variables were accessed from the base workspace, enabling real-time modification of the waveform properties.

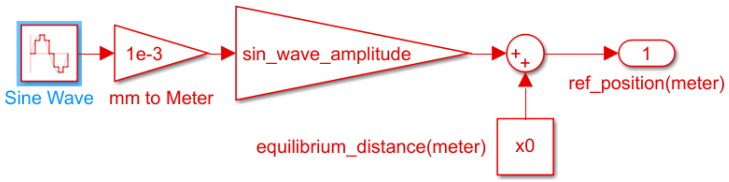


Figure 97: Model Based Development of Sine Wave Generator Subsystem

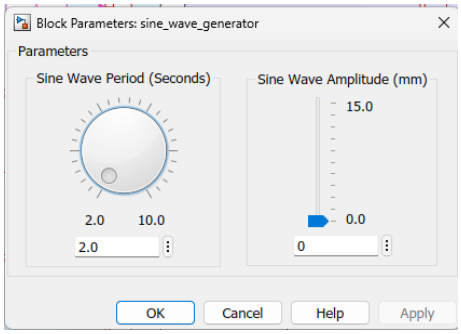


Figure 98: Sine Wave Generator Parameter Mask

Integration of Sine Wave in App Designer In order to connect the two important parameters of the Waveform which are Period and Amplitude a global object is declared named `app.ParamTuner_Amplitude` and `app.ParamTuner_Period`. This object are linked with the variable under mask called `sin_wave_amplitude` and `sin_wave_period` the following snippet of the code explains the procedure :

```
switch value
    case 'Sine'
        app.ParamTuner_Amplitude.BlockPath = [app.mdl '/sine_wave_generator'];
        app.ParamTuner_Amplitude.ParameterName = 'sin_wave_amplitude';
        app.ParamTuner_Period.BlockPath = [app.mdl '/sine_wave_generator'];
        app.ParamTuner_Period.ParameterName = 'sin_wave_period';
....
```

In this case it is important to disable the Position Slider since it will give warning of Value not found Hence in the case of Sine wave the following set of commands are used

```
app.PositionSlider.Enable = 'off';
app.PeriodSecSlider.Enable = 'on';
app.AmplitudemmSlider.Enable = 'on';
```

10.6.2 Triangular Wave Generator

Model Based Development of Triangular Wave In order to generate a triangular wave, we utilized the "Triangle Generator" Simulink block from SimScape. This block offers the capability to generate triangular waveforms with variable frequency and phase.

To ensure the waveform starts from the initial position of 0 mm, a phase shift of 90 degrees was applied.In order to optimize run time efficiency, the triangular generator block was set to run in discrete time with a sample time (Ts) of 0.01 seconds.

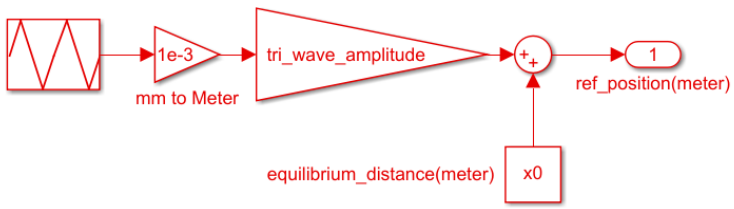


Figure 99: Model Based Development of Triangular Wave Generator Subsystem

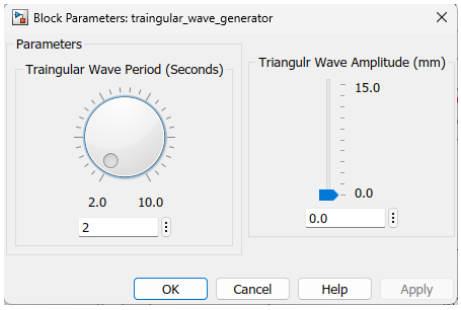


Figure 100: Triangular Wave Generator Parameter Mask

Integration of Triangular Wave in App Designer The Triangular Wave also have the same parameters such as Sine wave hence the Global Objects named `app.ParamTuner_Amplitude` and `app.ParamTuner_Period` are linked with the variable under mask called `tri_wave_amplitude` and `tri_wave_period` The following code snippet shows the procedure.

```
switch value
    case 'Triangular'
        app.ParamTuner_Amplitude.BlockPath = [app.mdl
            '/traingular_wave_generator'];
        app.ParamTuner_Amplitude.ParameterName = 'tri_wave_amplitude';
        app.ParamTuner_Period.BlockPath = [app.mdl
            '/traingular_wave_generator'];
        app.ParamTuner_Period.ParameterName = 'tri_wave_period';
....
```

Similar to the Sine Wave, The position control must be Disable to avoid the Warning Massage.

```
app.PositionSlider.Enable = 'off';
app.PeriodSecSlider.Enable = 'on';
app.AmplitudemmSlider.Enable = 'on';
```

10.6.3 Trapezoidal Wave Generator

Model Based Development of Trapezoidal Wave Although Matlab-Simulink does not provide an inbuilt trapezoidal wave generation block in its library, it is still possible to create a fixed period trapezoidal waveform using a lookup table. By introducing a variable period ramp signal along with variable amplitude, we can modify the period of the waveform and generate a variable period trapezoidal waveform.

The key parameters for the trapezoidal waveform are the Period, RiseTime, and OnTime. By manipulating these parameters, we can customize the shape and characteristics of the trapezoidal waveform.

By employing a lookup table approach and utilizing the specified parameters, we can successfully create trapezoidal waveforms with variable periods and customizable shapes. This flexibility allows for greater versatility in waveform generation within our system.

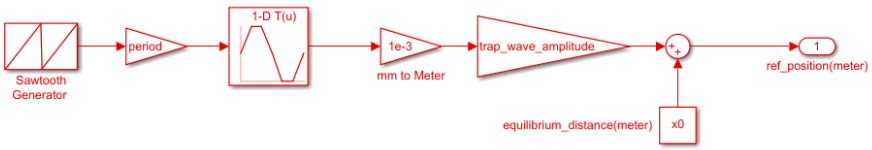


Figure 101: Model Based Development of Trapezoidal Wave Generator Subsystem

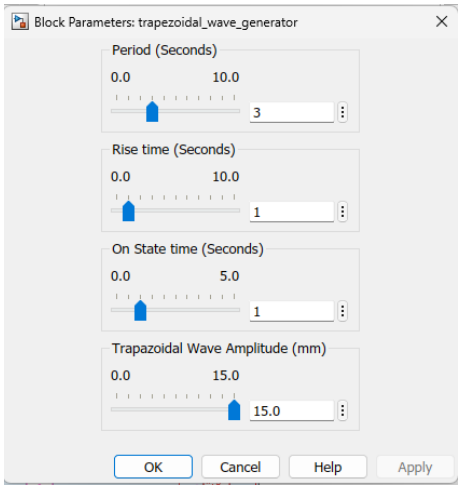


Figure 102: Trapezoidal Wave Generator Parameter Mask

The calculation of the variable waveform is given below. It is easy to make a lookup table with following data

$$Period = RiseTime + OnTime + FallTime \tag{68}$$

For the easy change in the These three parameters Period, OnTime and RiseTIme a slider mask is used by assigning the following variable names under the mask and linking this variable to base workspace.

Integration of Trapezoidal Wave in App Designer Unlike the sine and triangular waves, trapezoidal waves cannot be fully parameterized by adjusting only the period. Additional parameters such as rise time and on time need to be configured for accurate waveform generation. However, due to space limitations in the design, we were unable to accommodate two additional sliders for these parameters. As a result, the implementation of a variable period trapezoidal waveform function was not included in the final design. For this the PeriodSlider must be disabled as well as the value of the period slider must be set to a default value.

```
switch value
    case 'Triangular'
        app.PositionSlider.Enable = 'off';
        app.PeriodSecSlider.Enable = 'off';
        app.AmplitudemmSlider.Enable = 'on';
        app.PeriodSecSlider.Value = app.defPeriod;

        app.ParamTuner_Amplitude.BlockPath = [app.mdl
            '/trapezoidal_wave_generator'];
        app.ParamTuner_Amplitude.ParameterName = 'trap_wave_amplitude';
```

Fixed Point Development of Trapezoidal Wave

10.7 Functionality and Uses

In this sub-chapter, we will explore the systematic implementation of each function, adhering to the predefined requirements. The code's functionality has been progressively enhanced in successive versions, with Matlab's version control employed to effectively manage and track the record of changes.

10.7.1 StartupFunction

App Designer allows you to create a special function that executes when the app starts up, but before the user interacts with the UI. This function is called the startupFcn callback, and it is useful for setting default values, initializing variables, or executing commands that affect initial state of the app.

The major Functionality covered in the startup function is mentioned below

Function 1 : Setting up fixed figure size

According to insights drawn from The Essential Guide to User Interface Design by Wilbert O. Galitz, it has been noted that resizable frames may not be ideal for a straightforward application. Given that the application is designed with safety in mind, it is

recommended that the frame remains constantly maximized and positioned as the top-most window. Since the GUI Frame has control options as well as have crucial status indicators which should be monitored by operator. The following snippet of the code performs operation of adjusting the Figure Size.

```
screenSize = get(groot,'ScreenSize');
screenWidth = screenSize(3); % extract the Screen Resolution
screenHeight = screenSize(4); % in pixels
left = screenWidth*0.1; % figure Margin in percent
bottom = screenHeight*0.1;
width = screenWidth*0.75;
height = screenHeight*0.75;
app.UIFigure.Position = [[left,bottom], width, height];
drawnow;
```

Function 2 : Target Selection and Running the Simulation

From the Simulink Real-time Desktop Library it is easy to implement a control panel of Target Selector, Simulation Stop Time and Simulation Start/Stop Control. The brief introduction of the following components are given below.

TargetSelector

Simulink TargetSelector component creates a Target computer for an instrument panel. The selected target can be accessed by the Name defined in the target window or via a IP address (if the target is connected via LAN).It is possible to have different components in the same instrument panel synchronize with different target computers. this can be achieved by changing TargetSource property of Instrument. User can add multiple targets from `slrtExplorer`

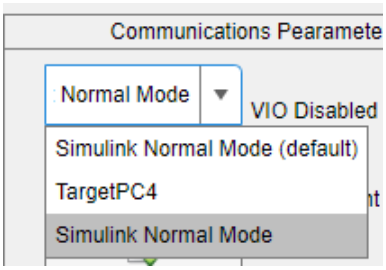
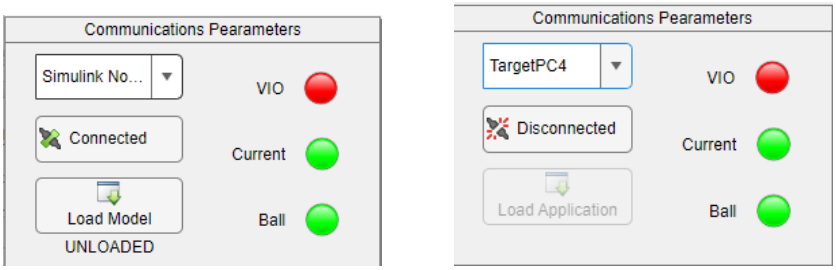


Figure 103: Predefined Target in Targets Selector

Connect/Disconnect Button

`slrealtime.ui.control.ConnectButton(hFigure)` creates a Target computer Connect-disconnect button. This button is disabled until a target is not selected.For the connected state, the button indicates that the development computer is connected to target computer. Clicking the button disconnects the development computer from the target computer. Upon changing the Target the Simulation is Stopped and the previous target is disconnected.



(a) Connected to selected target

(b) Disconnected upon change in Target

Figure 104: Voltage to Distance Converter

LoadButton

`slrealtime.ui.control.LoadButton(hFigure)` creates a Load Button for Instrument Panel. This Button loads Real-time Application as well as displays current loaded application. The load function gives a file-select UI to select the required .slx / .mdl file. In the Magnetic Levitation project, to avoid the clutter a global object "mdl" is created which is assigned to the part of desired application. Example of loading the application programmatically. When run in Real-time the selected file is called as "Application" whereas upon simulation in Normal Mode the selected file is called as "Model".

```
app.mdl = %(path of the application)
app.LoadButton.Application = app.mdl;
```

RunButton and StopTime

This are the common settings found in Simulink to Run the application and to specify the time of stop in seconds. In a Normal Mode a simulation Pacing is used with a scaling of 1. This points are self explanatory.

Function 3 : Disable all controls when the target is not connected to the PC

In order to prioritize safety, the controls have been made inaccessible when the target device is not connected. This requirement was implemented to address a past issue where inadvertently set values were immediately loaded onto the target device upon connecting it to the PC. This practice was deemed unfavorable as it could potentially result in the plant running immediately with unintended values upon pressing the "RUN" button. By disabling the controls in the absence of a target connection, the risk of unintended and potentially hazardous operations is mitigated. The following snippet of code check the target status. Upon empty target status it disables edit option of below mention controls.

```
app.tg = slrealtime(app.TargetSelector.TargetName); % Set target
% Disable all controls when the target is not connected to the PC
if isempty(app.tg.TargetStatus)
    app.tg.TargetStatus
    app.PositionSlider.Enable = 'off';
    app.PeriodSecSlider.Enable = 'off';
    app.AmplitudemmSlider.Enable = 'off';
end
```

Function 4 : Pattern Selector

A multi-port switch block was employed to select various waveform patterns. The switch block was driven by a variable stored in the base workspace. To establish a connection between the Value variable of the selectorKnob and the base workspace, the `slrealtime.ui.tool.ParameterTuner` function was utilized. The code snippet provided below outlines the procedure:

```
app.ParamTuner_PatternSelctor =
    slrealtime.ui.tool.ParameterTuner(app.UIFigure, 'TargetSource',
    app.TargetSelector);
app.ParamTuner_PatternSelctor.Component = app.PatternSelectorKnob;
app.ParamTuner_PatternSelctor.BlockPath = [app.mdl '/selector'];
app.ParamTuner_PatternSelctor.ParameterName = 'Value';
app.ParamTuner_PatternSelctor.ConvertToComponent =
    @(value)refPatternConvertToComponent(app, value);
app.ParamTuner_PatternSelctor.ConvertToTarget =
    @(value)refPatternConvertToTarget(app, value);
```

The line number 5 and 6 are declaring a callback initially to a function `refPatternConvertToComponent` which converts Text from SelectorKnob from Char format to Numbers and passes to the base workspace variable.

In the main function a SELECTOR switch has been set up with this options "OFF", "Triangular", "Sine", "Constant" and "Trapezoidal". The selector switch passes the selected value in character format. Since the targetValue must be a number ranging from 1 to 5 a function is deployed to convert componentValue to targetValue. On the other hand to hold the status of the selector switch to a set value we have to back assign the targetValue to componentValue. The following code snippet explains this two functionality.

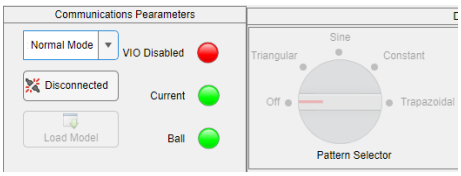
```
function compValue = refPatternConvertToComponent(app, targetValue)
    switch targetValue
        case 1
            compValue = 'Triangular';
        case 2
            compValue = 'Sine';
        case 3
            compValue = 'Constant';
        case 4
            compValue = 'Trapazoidal';
        case 5
            compValue = 'Off';
    end
    app.refPatternAdjust(compValue);
end

function targetValue = refPatternConvertToTarget(app, compValue)
    switch compValue
        case 'Triangular'
            targetValue = 1;
        case 'Sine'
```

```
            targetValue = 2;
        case 'Constant'
            targetValue = 3;
        case 'Trapazoidal'
            targetValue = 4;
        case 'Off'
            targetValue = 5;
    end
    app.refPatternAdjust(compValue);
```

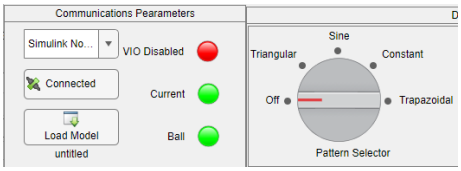
Function 5 : Keep the VIO Converter Disabled untill a Pattern is selected

In order to give preference to safety a constant block assigned to VIO-Enable has set to a default value of "0". Also the block sets to "0" upon stopping the application. In the following code a function is called `function offStateSetting(app)` in the OFF state, in a mode selected state function `function refPatternAdjust(app, value)` is called and within this function the VIO-Enable block is set to "0".

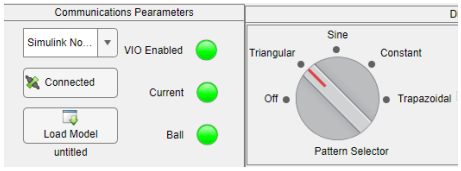
```
function offStateSetting(app)
    app.VIOLamp.Color ='r';
    app.VIOLampLabel.Text = 'VIO Disabled';
    app.tg.setparam([app.mdl '/VIO_Enable'], 'Value', '0');
...
end
function refPatternAdjust(app, value)
    app.VIOLamp.Color ='g';
    app.VIOLampLabel.Text = 'VIO Enabled';
    app.tg.setparam([app.mdl '/VIO_Enable'], 'Value', '1');
...
end
```



(a) disconnected from Target (VIO-OFF)



(b) Connected - Off State(VIO-OFF)



(c) Connected - Pattern Selected(VIO-ON)

Figure 105: VIO-Disabled until Pattern is selected

Function 7 : Constant Position Control

To fulfill the requirement of implementing a position control, a Case-Switch statement is employed. Within this case, a dedicated subsystem is created to receive input from a slider with a range of ±25mm. To meet this specific requirement, a separate slider is added that is disabled for pattern control, similar to the disabled Period and Amplitude sliders. Additionally, a position parameter can be entered through a text box, further enhancing the control capabilities.

In this case, two distinct but interconnected callbacks are utilized to ensure seamless functionality. By incorporating these components, the project effectively achieves the desired position control feature while maintaining the necessary inter-dependencies between various elements.

```
%Value Change function for position slider
function PositionSliderValueChanging(app, event)
    app.positionEditField.Value = num2str(app.PositionSlider.Value);
end

% Value change function fora Editfeld
function positionEditFieldValueChanged(app, event)
    app.PositionSlider.Value = str2double(app.positionEditField.Value) ;
end
```

The interlocked callback mechanism is employed to optimize the updateTime of a function by leveraging both "ValueChanged" and "ValueChanging" callbacks. When a new value is detected, the interlocked callback invokes itself to ensure prompt updates.

Use ValueChangedFcn Callbacks Instead of ValueChangingFcn Callbacks
Many components, such as sliders and text areas, have both a ValueChangedFcn callback and a ValueChangingFcn callback. Both of these callbacks execute in response to a change in the component value, but they execute at different times in the interaction.

- The ValueChangedFcn callback executes once after the app user finishes the interaction. For example, the ValueChangedFcn callback of a slider executes after the user releases the slider thumb at its final value.
- The ValueChangingFcn callback executes multiple times at regular intervals while the app user performs the interaction. For example, the ValueChangingFcn callback of a slider executes regularly as the user drags the slider thumb.

By utilizing a combination of these two callback types, the system achieves efficient and timely execution. The "ValueChanged" callback is triggered when a new value is finalized, while the "ValueChanging" callback is invoked during the transitional phase as the value is being modified. This clever approach significantly enhances the update-Time of the function, enabling swift and responsive performance.

10.7.2 Concept of Instrument and Instrument Manager

Simulink Instruments are powerful tools in MATLAB that facilitate the real-time signal capture and analysis of objects defined within the method. These instruments can be easily created using the keyword `App.Object = slrealtime.Instrument`. Additionally, instruments can also be created from the instrument explorer, providing flexibility in instrument management.

Instruments are capable of connecting to various signals, which can be represented as vectors of data. These signals are essential for plotting the signal vectors on a graph, enabling visual representation and analysis. On the other hand, certain instruments do not require a series of data but instead focus on instantaneous values known as scalars. Scalar instruments are particularly useful for displaying values on meters, dials, and supporting animation updates. Use can add new signal by `addSignal` as well as `connectScalar` to add new vector or scalar data. to generate a callback upon new value of a signal `connectCallback` is used.

The MATLAB Instrument Manager is a powerful tool within App Designer, simplifying the management of instruments and allows to synchronise configure and visualise the data in simplest way possible.

Integration of Multiple Instruments : It equips developers with the necessary tools to seamlessly integrate instruments – such as graphs, meters, dials, and charts – into the interface, providing users with a real-time data with minimal setup.

Synchronization : allowing developers to ensure data consistency and coherence within the GUI. It handles the coordination of data updates and refresh rates, guaranteeing that all connected instruments display synchronized real-time data.

Instrument Configuration : users can easily adjust the properties of particular instruments – such as the range of an axis or the visibility of data labels – to finely tune their visual representation.

Instrument Interaction : The Instrument Manager allows users to interact with instruments by providing interactive features like zooming, panning, and cursor tracking. These functionalities enhance the user experience, enabling users to explore and analyze the displayed data more effectively.

Finally, the Instrument Manager supports the simultaneous control of multiple instruments. It facilitates the setup of multiple instruments, connecting each of them to their respective data sources to ensure complete data visualization. Allowing developers to easily integrate, synchronize, and configure instruments within GUIs, the MATLAB Instrument Manager is a truly invaluable asset for the App Designer environment.

for More information author of this report suggest to please refer to `slrtExplorer` command and access the visual features in the Simulink Real time Explorer. (Note : The GUI Development is a completely was completely programmatic and `slrtExplorer` was used for learning and prototyping but not used in final release version 3.6)

Function 4 : Realtime plotting on the graph A new Instrument is created in a startup Function. A `connectLine` keyword is used to display instantaneous value of the signal on the Graph figure. the following code snippet summarise the process.

```
app.Instrument = slrealtime.Instrument; % Create Instrument
app.UIAxes.YLim = [-0.03,+0.03]; % sets Fixed Yaxis Limit
app.Instrument.connectLine (app.UIAxes,'ref_position(meter)');
app.Instrument.connectLine(app.UIAxes,'lin_x_log');
app.Instrument.AxesTimeSpan = 20; % fixed X axis
grid(app.UIAxes,"on");
app.Instrument.AxesTimeSpanOverrun = 'wrap'; % 20 sec repets
legend(app.UIAxes,'Reference Position (mm)','Displacement (mm)');

% updates the value into App developer's - Instrument manager

app.InstrumentManager = slrealtime.ui.tool.InstrumentManager(app.UIFigure,
    'TargetSource', app.TargetSelector);
app.InstrumentManager.Instruments = app.Instrument;
```

10.8 Data logging

Datalogging is a important feature in Realtime Control system application because it allows the system to record data over time or about values from various sensors and instruments. Datalogging can help the system to track real-time and historical trends, manage alarms and events, network with other devices, and add security to user interfaces. For instance, a data logger can monitor the status of the plant and alert the operator if any abnormal condition occurs. Datalogging can also enable the system to perform hard real-time data logging tasks and soft real-time monitoring tasks.

The similar experienced with the current project where we manage to detect and solve two critical problems via a logged data in early phase of the development. (reference to the chapter).

10.8.1 Data logging in Simulink

Logging the Instrument Object created with slrealtime.Instrument is a convenient option for data logging, especially when readings are taken at regular sample times. Initially, the proposed approach involved logging data in a matrix format and saving it as an Excel file. However, a challenge arose as the matrix grew larger, causing the program to wait for data and freezing the animation and graph plotting functions.

Although this issue initially went unnoticed, it became apparent that the animation could tolerate a few dropped frames without significant impact. To address this, a Try...catch function was implemented in the animation process. Despite this mitigation, the problem persisted, prompting the need for an alternative approach.Considering the limitations of the matrix-based data logging method, an alternative solution was sought.

In this new approach data is logged with the Matlab's inbuilt function block "To Workspace" for which a data logging subsystem is made to sample the data for every 0.01s. The sampled data is stored in a Base workspace in the form of "SimulationOutput". afterwards a matrix is structured including these three variables "Time", "Reference-Position", "Displacement" and "Controller current". This matrix is used to draft a excel file.

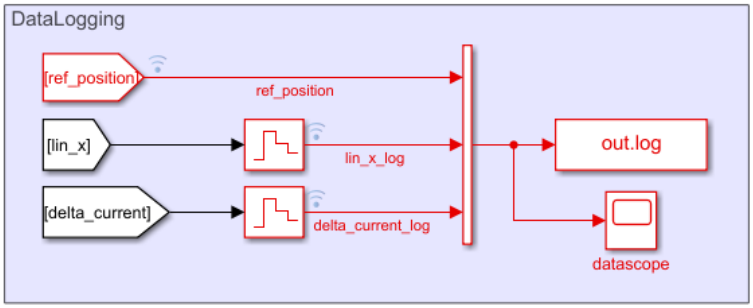


Figure 106: Data logging Subsystem Model Based

The Red coloured signals shows that they are sampled at 0.01 seconds where as black signals are denoting a time continuous characteristics.

10.8.2 Data logging implementation in App Designer

To fetch the value of a logged data from base workspace to App Designers workspace a global objects are created. This objects are namely log_time, log_displacement, log_reference_position, log_delta_current. To name the logged file a Name of the following code is used upon value change function for data logging Button.

```
function LoggerONOFFButtonValueChanged(app, event)
    app.datetm = datestr(now, 'dd_mmmm_yy_HH.MM'); % gets the current Datetime
    app.dir = pwd; % finds the current directory path
    % reads SimulationOutput from base workspace
    app.log_time = evalin('base','out.log.Time');
    app.log_reference_position = evalin('base','out.log.Data(:,1)');
    app.log_displacement = evalin('base','out.log.Data(:,2)');
    app.log_delta_current = evalin('base','out.log.Data(:,3)');
    app.logmat =
        table(app.log_time,app.log_reference_position,app.log_displacement
                ,app.log_delta_current);
    app.logmat.Properties.VariableNames = ["Time","Reference Position",
        "Displacement", "Controller Current"];
    %creates Excel File
    writetable(app.logmat,[app.datetm
        'loggeddata.xls'],'Sheet',1,'AutoFitWidth',true);
    datalogmassage = ['Datalogging File Created Sucessfully in ' app.dir '
        Filename ' app.datetm '.xlsx'];
    uialert(app.UIFigure,datalogmassage,'Datalogging
        Finished','Icon','success');
end
```

Some Major Features of data logging Implementation Code

- Creates a copy of the data in the Base Workspace under out.log Timeseries. This data can be used in a matlab for further calculation (if needed).

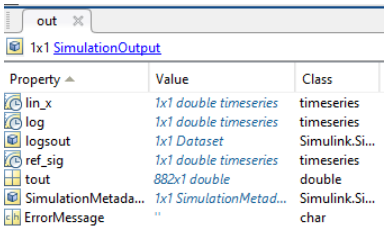


Figure 107: Data logging output in Base Workspace

- The Variables log_time, log_displacement, log_reference_position, log_delta_current are limited to minimum 4 digits of accuracy. Since more digits makes the base variable bulky.
- The logged file is named in a format "dd_mmmm_yy_HH.MM_loggeddata.xlsx". example : 01_may_2023_12_55_loggeddata.xlsx. This format clears the confusion between month and date (developer's personal choice).

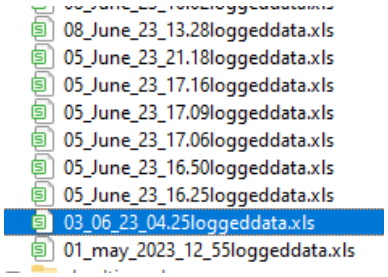


Figure 108: Data logging Files Generation

- The Excel file columns are provided with appropriate Headers as well as the Column-width is adjusted programatically for user-friendly view.
- upon successful generation of the Excel file a UI Dialog appears stating the path of the Saved File as well as Name of the file.

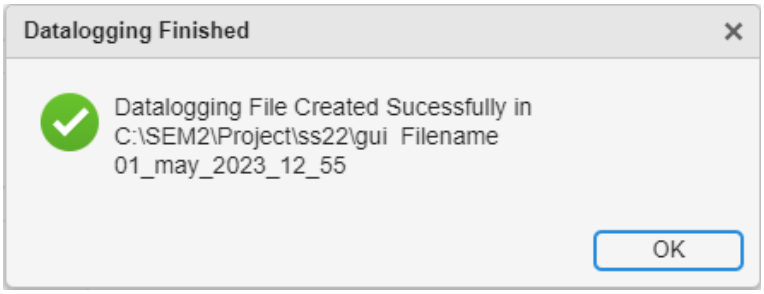


Figure 109: Data logging UIAlert

10.8.3 Plotting of the Logged Data

A Callback has been created which reads the global objects app.log_time with respect to app._displacement and app._Reference_position. Plots in the same UIFigure where real-time data is plotted. The graph has a auto adjusting feature which makes easy to view.

```
function PlotLogButtonPushed(app, event)

    plot(app.UIAxes,(app.log_time)',(app.log_displacement)','g',(app.log_time)',
                    (app.log_reference_position)','r');
    app.UIAxes.YLimMode = "auto"; % Automatically adjust the Graph Size
    app.UIAxes.XLimMode = "auto";

end
```

10.8.4 Clearing the Logged Data

To fulfill this requirement a Button is added with a callback to clear the plot

```
function ClearLogButtonPushed(app, event)
    YN = uiconfirm(app.UIFigure, 'Clearing the Plotted Data' ,
        'Clear','Icon','warning');
    if strcmpi(YN, 'OK' )
        cla(app.UIAxes);
    end
end
```

10.9 Animation

The initial plan for the animation involved implementing a graphical representation of a sliding ball along the Y-axis and animating the movement of the Reference Marker using graphics. However, a decision was made to adopt a more flexible approach by developing the graphics within the code itself.

The main purpose of the animation is to visually depict the movement of the ball. It's important to note that since it is an animation, it does not reflect every new data point in real-time. Therefore, the position of the ball cannot be accurately determined solely based on the animation. It serves as a visual aid to provide a general representation of the ball's movement but should not be relied upon as the sole source of position information.

By implementing the animation within the code, we achieve greater flexibility in customizing the graphical objects and their behavior. This allows for a more dynamic and interactive visualization of the ball's sliding motion and enhances the overall user experience.

10.9.1 Animation Axis setting

in order to generate a animation, few global objects are developed which serves the purpose of creating a patch as well as holding a default data.

The implemented approach employs a two-step process to achieve efficient animation. Initially, an animation axis is generated, and animation objects are positioned in their default locations. Subsequently, a dedicated function is utilized to update the values associated with the position of these objects in real-time whenever a callback is triggered.

By employing this approach, the need for creating new animation objects frame by frame is eliminated. Instead, the existing objects are updated with new parameters, this significantly improving the overall efficiency of the code. This optimization has resulted in substantial time savings, as there is no longer a requirement to recreate animation objects from scratch with each update. As a result, the update time of the code has been noticeably enhanced, leading to a smoother and more responsive animation experience.

```
% Animation Objects
aniRefMark
aniBallPos
ballXComp = 0.005*sin(0:pi/10:2*pi); %rsin(theta)
ballYComp = 0.005*cos(0:pi/10:2*pi); %rcos(theta)
%r Radius of the ball
% (theta) are the segement of full circle (in rad)
```

In the above code a circle is formed with a standard circle equation

$$(sin(\theta))^2 + (cos(\theta))^2 = r^2 \tag{69}$$

To streamline the animation process, it is impractical to draw the circle at every degree increment. Instead, we divide the complete 2π radians into 10 equal parts, allowing us to approximate the circle by connecting the corresponding coordinates. By joining these coordinates together, the resulting patch closely resembles a circle. This approach significantly reduces the number of data points and computational complexity involved in rendering the animation, while still maintaining a visually representative depiction of a circle.

Matlab `patch()` function

The `patch()` function in MATLAB is a versatile tool used for creating and manipulating two-dimensional patch objects. A patch object represents a filled polygon or shape defined by a set of vertices. users can define the shape, color, transparency, and other properties of the patch object. The vertices of the polygon can be specified as matrices or vectors. Multiple patches can be created within a single plot, allowing for the visualization of complex geometries. example of `patch()` function

```
x = [0 1 1 0];
y = [0 0 1 1];
patch(x,y,'red')
```

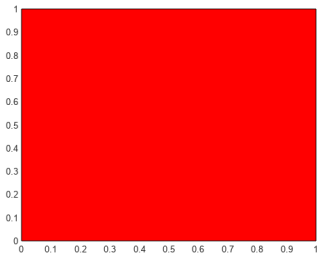


Figure 110: Simple Rectangular Color Patch

Similar to the function above we have createda patch for a circle and a line.

```
function setupAnimateAxes(app)

    app.AnimationAxes.XLim = [-0.03 0.03];
    app.AnimationAxes.YLim = [-0.03 0.03];

    % drawing Circular shape that resembles ball
    app.aniBallPos = patch( ...
        'Parent',  app.AnimationAxes, ...
        'XData',   app.ballXComp, ...
        'YData',   app.ballYComp, ...
        'FaceColor','red');

    % drawing a Line shape
    app.aniRefMark = patch( ...
        'Parent',  app.AnimationAxes, ...
        'YData',   [0 0], ...
        'XData',   [-0.01 +0.01], ...
        'EdgeColor','green');
end
```

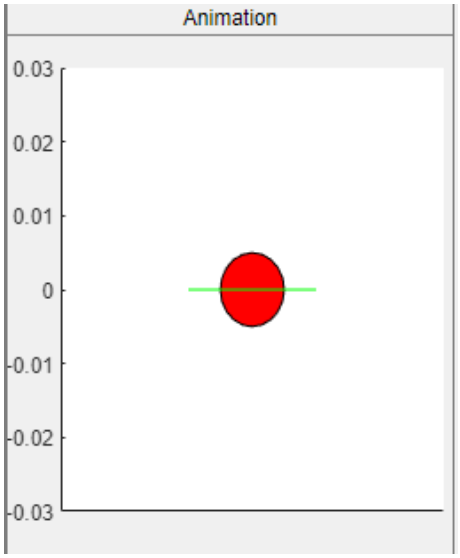


Figure 111: Patch Object for Ball and Reference Marker

10.9.2 Update Animation Function

The `getCallbackDataForSignal` function in MATLAB is a useful command for Reading callback data associated with a specific signal in Simulink Real-Time applications. It provides a method to retrieve information about the current state and value of a signal during real-time execution.By using `getCallbackDataForSignal`, users can obtain the signal's name, data type, dimensions, and current value. This function enables dynamic monitoring and analysis of signals in real-time control systems.

The "updateAnimationCallback" function plays a important role in generating callback data from a signal based on time in order to animate the system. This function generates an array that stores the data along with their corresponding timestamps, the most recent value added at the top of the array. Each time an object instance occurs, this function is triggered.

By updating the value of the variable and adding it to the default value of the patch, the animation progresses frame by frame. However, continuously calling the animation function after every data change is not an good idea as it slows down the overall execution of the function. Despite this drawback, this is currently the only viable solution.

```
function updateAnimationCallback(app, instObj, eventData)

    [time, ref] = instObj.getCallbackDataForSignal(eventData,
        'ref_position(meter)');
    [~, displacement] = instObj.getCallbackDataForSignal(eventData,
        'lin_x_log');

    if ~isempty(time)
        ref   = ref(end);
        displacement = displacement(end);
        try
            app.dist_label.Text = num2str((1000)*displacement); % meter to mm
                Conversion
            set(app.aniBallPos, ...
                'YData',displacement+app.ballYComp);

            app.ref_label.Text = num2str((1000)*ref); % meter to mm Conversion
            set(app.aniRefMark, ...
                'YData',[ref ref] );
        catch
            % Ignore any errors writing to uicontrols because app
            % may have been closed and uicontrols destroyed
        end
    end
```

A try catch statement is used to avoid program waiting for a next value.

10.10 Exit-Sequence

With safety as a priority, we recognized the need to implement an exit function that goes beyond simply clearing the UIFigure. This enhanced exit function serves multiple purposes to ensure a safer system shutdown.

```
function UIFigureCloseRequest(app, event)
    YN = uiconfirm(app.UIFigure, 'Initiate Exit?' , 'Close
       request','Icon','warning');
    app.tg = slrealtime(app.TargetSelector.TargetName);
    if strcmpi(YN, 'OK' )
        if isempty (app.tg.TargetSettings.address)
            uialert(app.UIFigure,'Target Disconnected/ Not Found','Closing
               Application by BruteForce');
            pause(2); % pause to show window
            delete(app); % Delete app prematurely
        else
            % sets all parameters to default
            app.tg.setparam([app.mdl '/selector'], 'Value', '5');
            app.PatternSelectorKnob.Value = 'Off';
            app.tg.setparam([app.mdl '/sine_wave_generator'],
               'sin_wave_amplitude', num2str(app.defAmplitude));
            app.tg.setparam([app.mdl '/trapezoidal_wave_generator'],
               'trap_wave_amplitude', num2str(app.defAmplitud
            app.tg.setparam([app.mdl '/traingular_wave_generator'],
               'tri_wave_amplitude', num2str(app.defAmplitude)
            app.tg.setparam([app.mdl '/constant_distance'],
               'constant_displacement', num2str(app.defAmplitude));

            app.tg.setparam([app.mdl '/sine_wave_generator'],
               'sin_wave_period', num2str(app.defPeriod));
            app.tg.setparam([app.mdl '/traingular_wave_generator'],
               'tri_wave_period', num2str(app.defPeriod));
        end
            delete(app);
    end
end
```

- Confirmation Prompt: To provide a user-friendly experience, a confirmation window labeled "Starting Exit Sequence" is displayed, requesting user confirmation before clearing the UIFigure and exiting the application. If the user selects "YES," the exit function proceeds; otherwise, it returns from the callback, preserving the current state of the application.

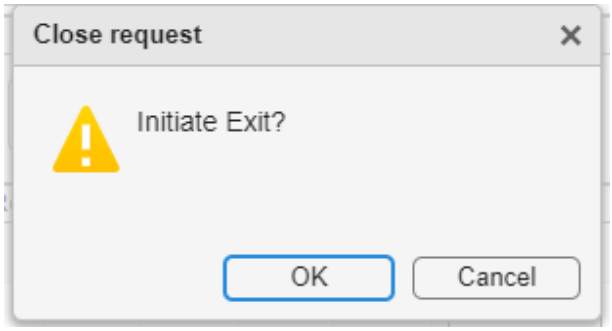


Figure 112: Exit Sequence Confirmation

- Real-Time Target Status Check: Before closing the application, the exit function verifies the status of the Real-Time Target. If the target is found to be disconnected, indicating that it was disconnected before initiating the safe shutdown, the function takes appropriate action. This includes alerting the user about the disconnection for future reference and informing that the target must have previous parameters feed. This function is useful when you again start the Application in future, user will find that Period and Amplitudes are settled to zero. This avoids accidents and surprises.

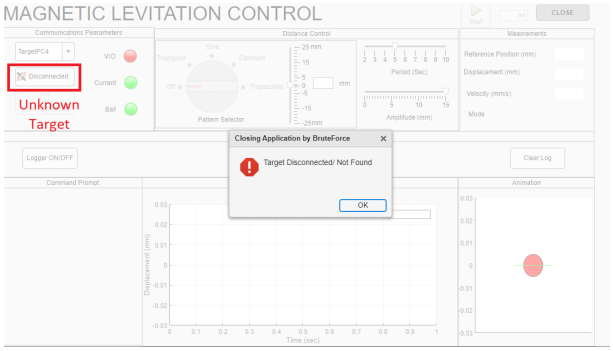
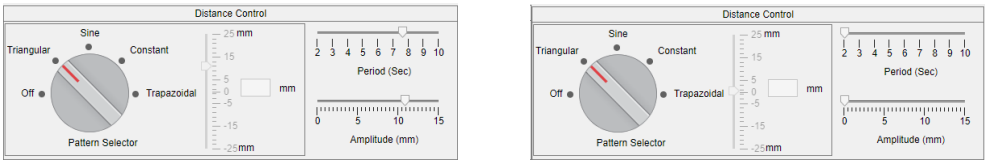


Figure 113: Exit Sequence Target (IP) not found

- Default Parameter Assignment: As part of the safe shutdown process, each parameter within the application is assigned a default value before closing the UIFigure. This ensures that the application restarts in a consistent state when launched again, preventing any unintended behavior or errors due to uninitialized or leftover parameters.



(a) Before Exit (b) After Restart Start

Figure 114: Exit sequence in Action

By incorporating these steps into the exit function, we prioritize a safe and user-friendly shutdown process for our application, allowing users to confirm their intent, handling

target disconnections gracefully, and ensuring a clean reset of parameters for subsequent application runs.

10.11 Additional Features

This section of the chapter contains features which encountered during debugging of the code.

- **Over current Indication**
  To address the overcurrent a Callback from signal is generated which stacks the data in first come first order. This data is compared with the Threshold value upon which the Over current Indication Triggers.
- **Ball Absent Indication Indication**
  The presence of the ball is sensed by the same procedure. A distance sensor reading was taken when the ball is absent. If this threshold exceeds then the program will reflect into ball absent condition
- **VIO Enable/Disable Indication**
- **Selected Pattern Label**
- **Display Reference Position (mm)**
- **Display Displacement (mm)**

10.12 Results

The chapter summarises different inputs to the

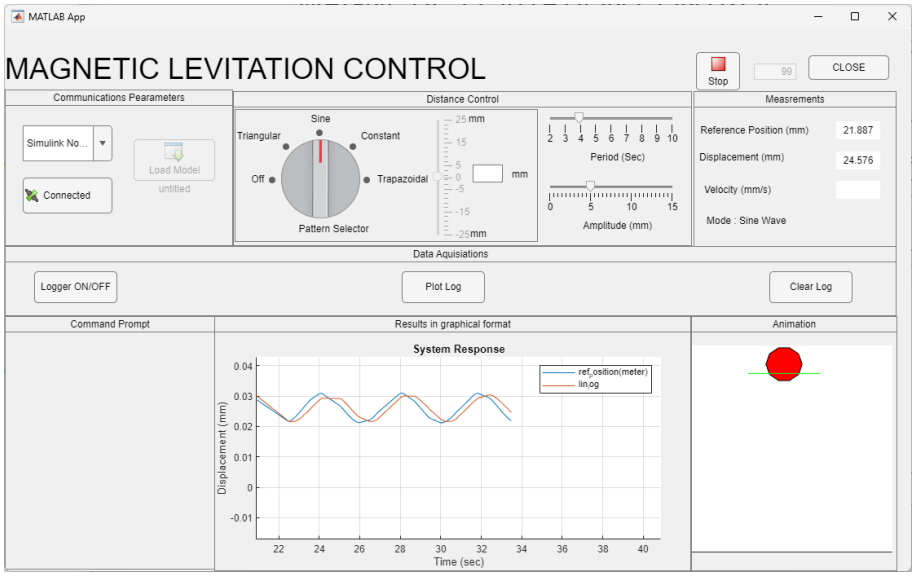


Figure 115: Final Release Version of GUI 3.7

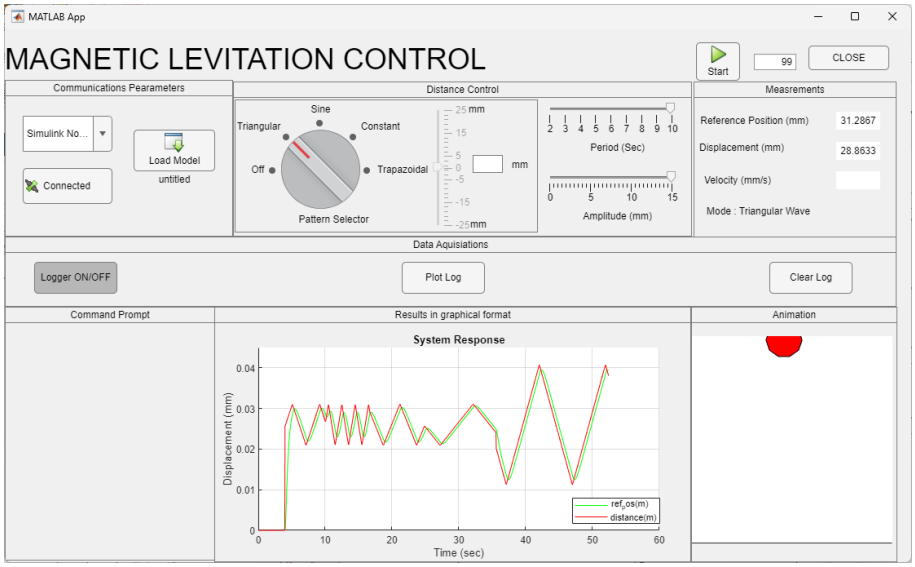


Figure 116: Real-time Signal subjected under Triangular Input

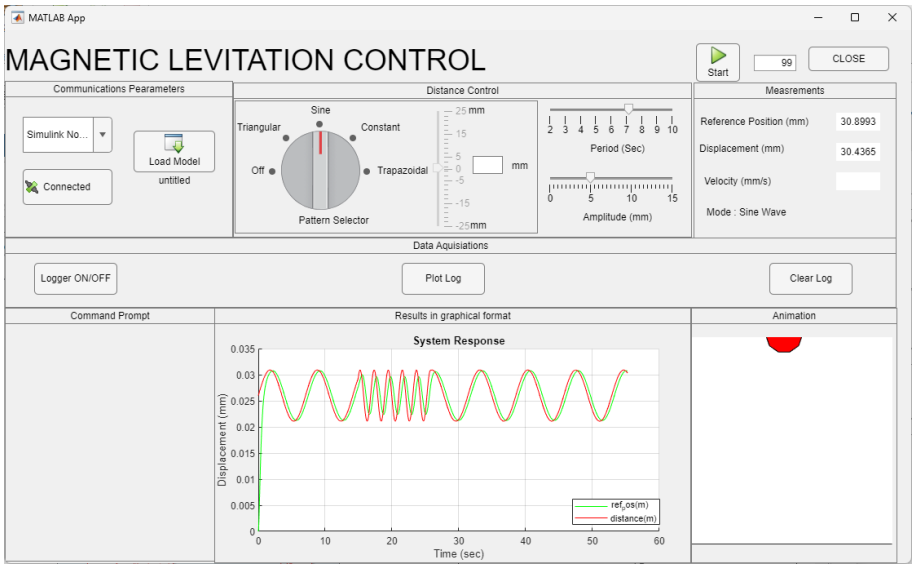


Figure 117: Real-time Signal subjected under Sine wave Input

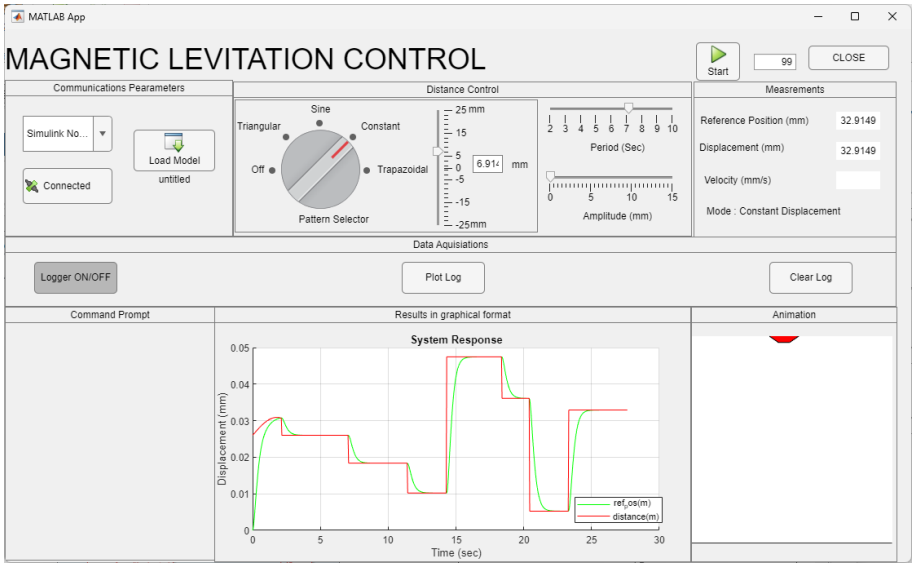


Figure 118: Real-time Signal subjected under constant step Input

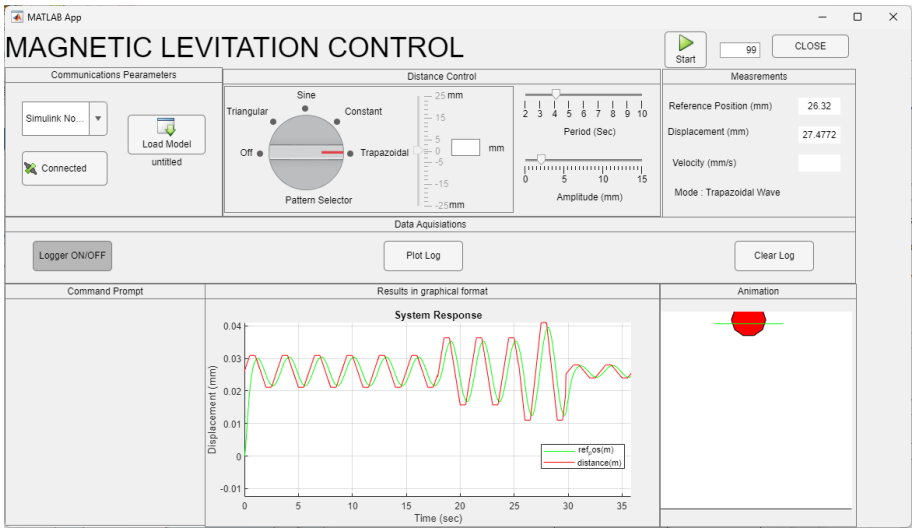


Figure 119: Real-time Signal subjected under Trapezoidal wave Input

11 Communication protocol

The English word "communication" refers to the exchange of information between two entities. Take the example of two physicists communicating with each other using the English language. For an effective interchange, it is necessary that they are in the same context. Now let us draw up analogies with this example and the communication protocol used in our project. For simplicity purposes, we have referred to GUI as the host and the FPGA as device.

For the physicists to understand each other, it is necessary that:

- They talk in the same language. There is little or no communication if one person talks in German and the other in English. This is defined by the **baud rate.**
- Specific set of grammar is to be followed to identify the beginning and ending of each sentence. This can be defined by **start and stop bits**.
- In order to check if the information is successfully conveyed, interrogative questions are asked this can be compared to a **parity bit**.

Whether it might be a simple talk between two people or the exchange of complex data between our host and device, both could not be achieved without a predefined set of communication parameters.

In our project we have defined the communication parameters as follows:

Baud rate (bits per second): 115200

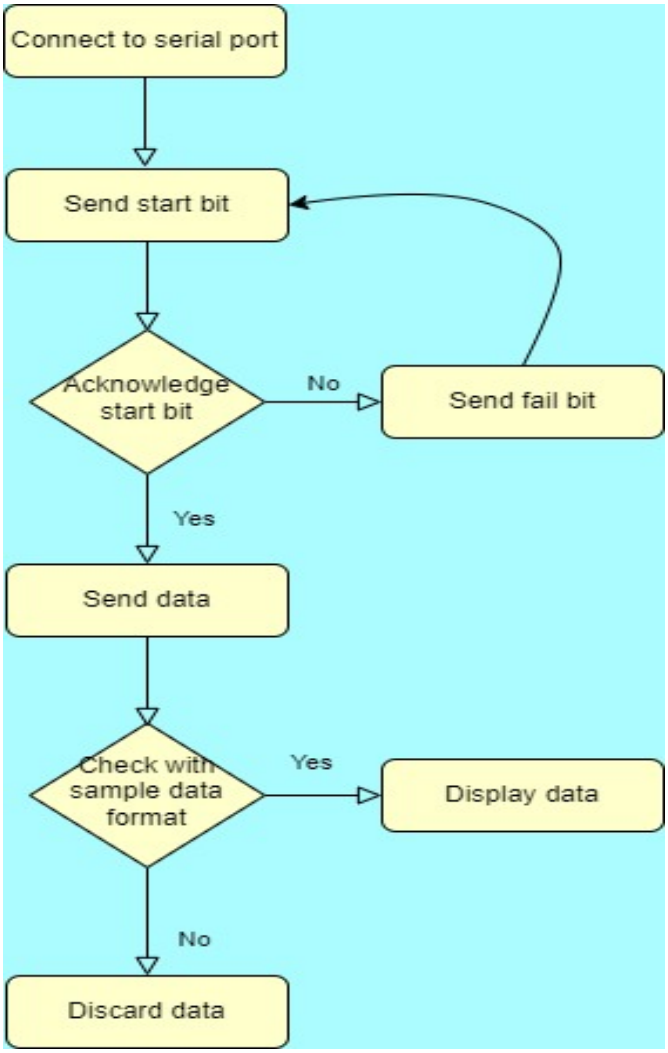
Number of start and stop bits: one and zero respectively

Parity bit: 1

Data bits: 8

11.1 Reliable Communication protocol

Since our application of a magnetic levitation control is not safety critical, continuous monitoring of data is not required. This application is far different from medical devices where wrong data communicated would result in the patient risking his life. Therefore, we have decided not to make the communication system complex by including redundancy in the system. This also has drawbacks practically as it would lead to a slower response and increased costs. However, we can illustrate the theory behind a reliable communication system.



Future development: As soon as the connection to the required serial port is established, the GUI or the host sends a start bit to the device or FPGA and the latter must acknowledge the start bit or send a fail bit in return. Upon failure, the host must keep retrying after a stochastic time. Once the start bit is acknowledged, communication can take place based on preset parameters, the data received must be checked by the sample data and in case of discrepancies, the data can be discarded and the host waits for new data from the device. Now the received data can be processed and displayed in the GUI. Finally, a stop bit must be sent and acknowledged by each other in order to stop the communication. Communication can be initiated again by sending a start bit.

12 Graphical User Interface (GUI)

12.1 Introduction

For an end user, seeing a complex mesh of circuitry can be very cumbersome which often leads to little or no understanding on how the project can meet his needs. Any real-world application must be abstractive in nature. It should be as simple as pressing a button and getting the desired output, therefore distancing the user from the complex processes that arise in the background. It is for this reason that we have developed the Graphical user interface or the GUI. It is an attractive window with buttons, displays which can be used for a successful interaction between the user and the machine.

According to a recent study, it has been found out that a whooping 74 percentage of users make or break a products life cycle just by judging the outer appearance of a product. Since all products are abstractive, the only thing capable of turning eyes is a well-designed GUI which allows users to dive into the product.

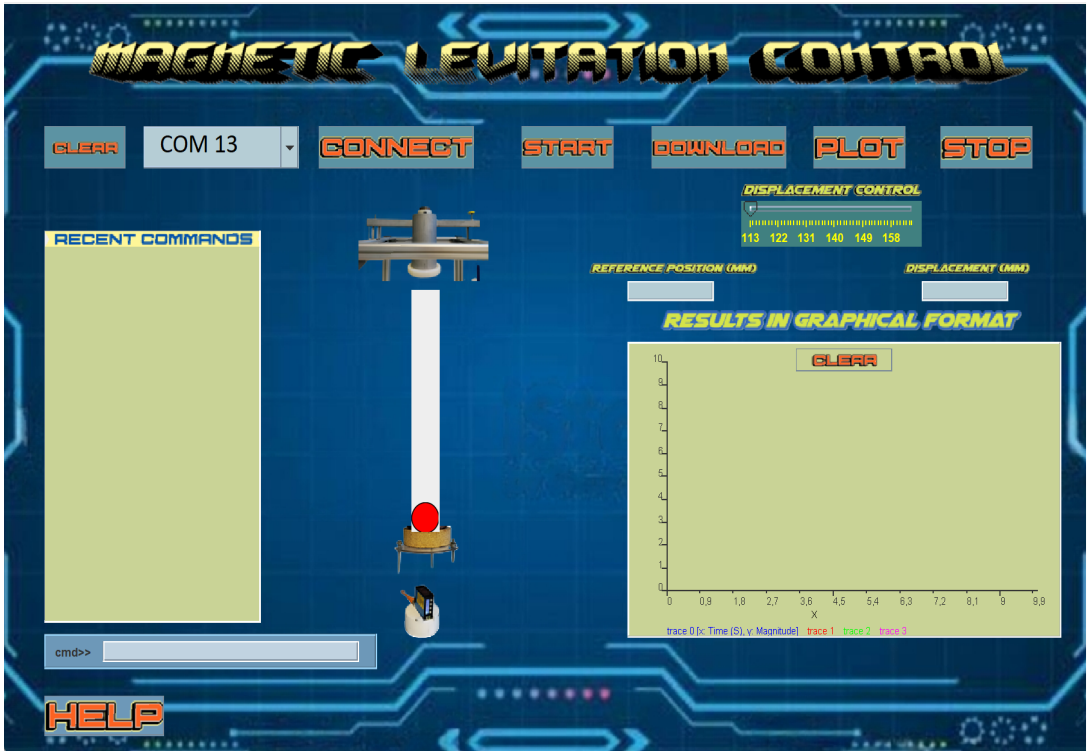


Figure 1: Final GUI

Taking our project as example, the required input from the user is the desired displacement of the ball. This can be achieved by just putting shabby button and a text window. We have taken the GUI to an entirely new echelon which we presume was never implemented before by students on magnetic levitation project. We show a real time animation of the ball on the screen and also plot graphs based on real world data. We have also included a sliding potentiometer as input for distance. Also, there is live tracking of the ball. If for some reason, the overloaded

GUI doesn't function then, there is also a possibility to control the machine by sending

particular commands. As the ports of different computer may vary there is a need for the GUI to adapt and function on every device it runs on. So, we have developed an auto port detect function thus making our GUI machine independent.

12.2 Different Elements in the GUI Panel

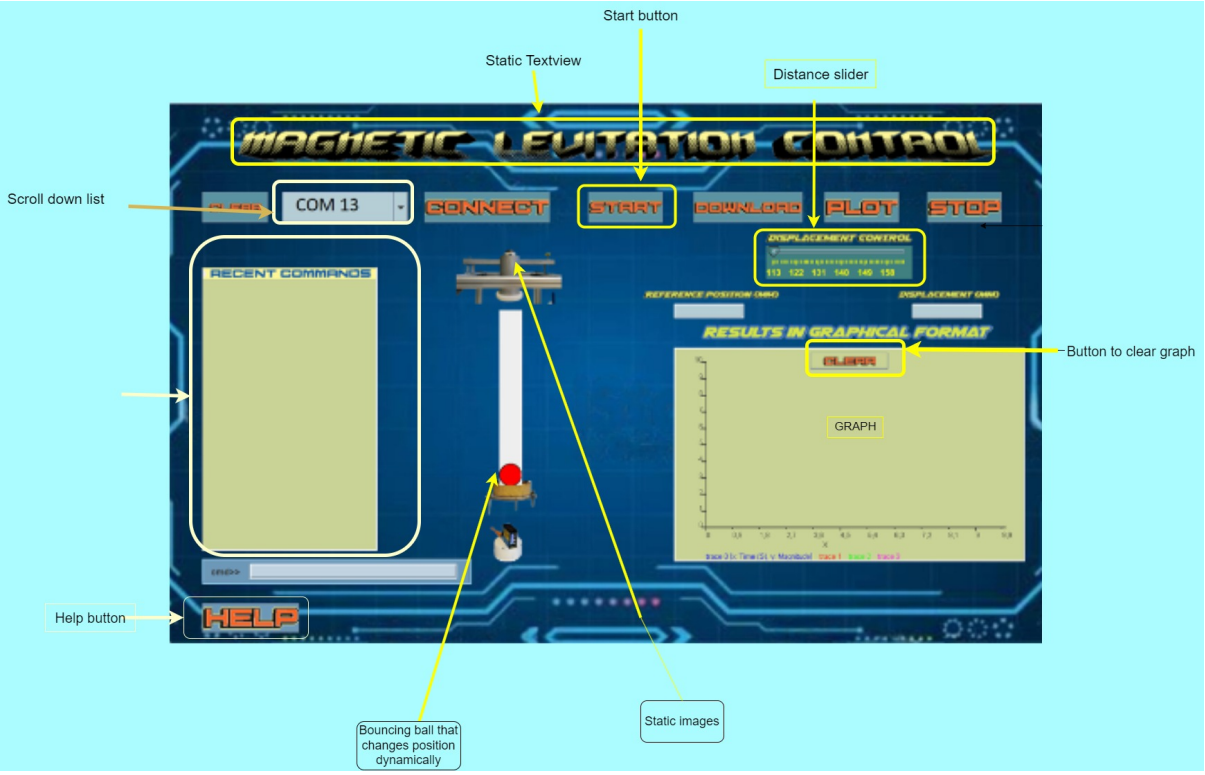


Figure 2: Different elements in the GUI Window

12.3 Buttons

Start: The start button is used to initiate the session. It reads the value entered into the displacement text field or reads the change in slider position and then sends the value to FPGA which ultimately controls the ball.

Stop: This button terminates the session and powers off the magnetic levitation device.

Help: This is used to display the list of commands recognized by the machine code to an unknown user. Upon clicking this button, all the list of available commands are displayed on the command window.

Clear: It clears the command window of previously displayed commands in order to avoid ambiguity.

Clear graph: It clears all previously plotted lines on the graph and allows new data to be plotted.

Plot: It fetches the logs from the FPGA and plots them on the graph

12.4 Linear Slider

Sometimes, typing the numeric value in the displacement text view could be cumbersome or the user might not be aware of the range of accepted inputs or since the text view can accept any range of characters as inputs, runtime errors can be generated. In order to solve all these problems, we have included a slider input. The user can simply slide the bar until a desired input vale is displayed in the text view.

12.5 Toggle List

Since there is a possibility to connect the FPGA to any serial port of the computer, it is not a good practice to display all the available ports to the user and let him connect and disconnect from each port until the device is found by trial-and-error basis. In order to avoid this, we have made a scroll down list that updates dynamically based on the serial port to which the device is connected. In order to establish a two-way communication, the user can choose a serial port

from this list and click on the "Connect" button. To terminate the connection, "Disconnect" button is used.

12.6 Command Window

It displays the status of the machine and also the history of the different commands sent.

12.7 Graph

This is a 2d graph that updates dynamically based on the log of values sent by FPGA.

12.8 Command text field

Instead of pressing the buttons to perform a certain function, user can also feed a string command in the command text field and it will perform the same function. The command text field is shown below in figure. There are multiple input strings which the user can input. The following table represents the function performed when a certain string is inserted. If command

starts with ".", followed by command string then such commands are sent serially to the processor by the GUI.

12.9 Bouncing ball

This is a unique and appealing aspect of our GUI. It visually animates a floating ball on the screen and updates its location dynamically based on the real time position of ball in the magnetic levitation setup. We have added images above and below the floating ball so that it just appeals to the eye.

13 Explanation of GUI code

13.1 Overview

Since java is a highly popular collaborative programming language, users can make use of pre developed functions in order to make the coding easier and understandable. To use these functions, it is necessary to import the required packages before even beginning to code. Some of the packages we imported for this module are:

Jframe: For the alignment of buttons and other elements in order to make the GUI attractive. Chart2d: contains functions to plot two dimensional charts

Borderlayout, springlayout and flowlayout: contains different functions that help in organising the GUI elements relative to each other

Actionevent, Actionlistener: Perhaps the most important the package that enables the functionality of GUI as it can provide functions to detect user interference

Jlabel and Jtextpane: Used to display and input text respectively

Timer: enables smooth communication between serial peripheral devices by delaying the code execution whenever needed.

ImageIcon: Allows displaying images in GUI.

Now that we have successfully imported all the required packages, there is one more step to be done before programming our logic. Java requires you to declare some packages that come in handy during the program execution and generate runtime exceptions to be mentioned along the a class declaration so we use the extends and implementation keywords for Jframe and ActionListener respectively.

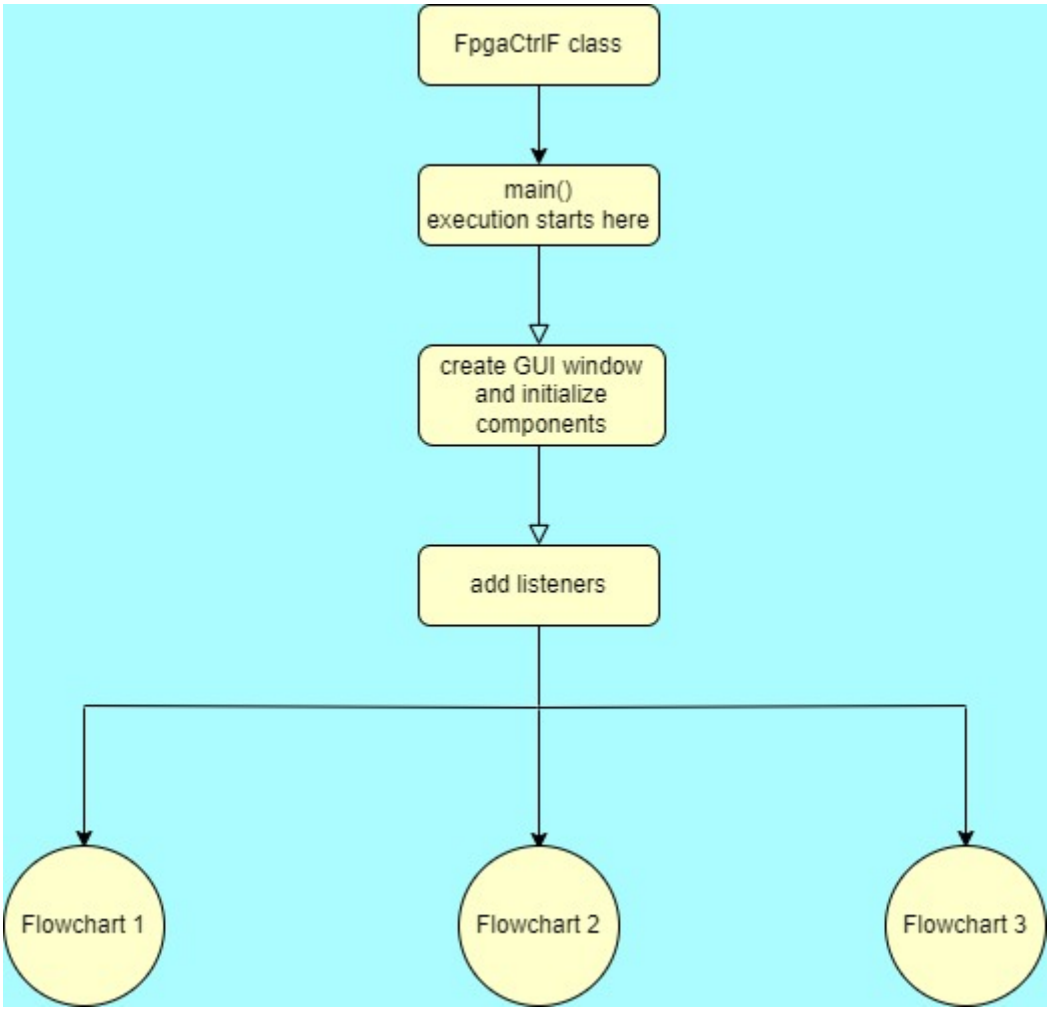


Figure 3: GUI Data Flow Overview

We begin with declaring the variables that are at the highest or global priority or which are shared

by multiple functions. In order to preserve the value of the variable, we make use of the "static: keyword. As the name suggests, the main() method is where the execution of the program begins. An object of the A serial class is created to be used later. The first thing to be done is to display the GUI window. If it is not successful, we handle exceptions using the try-catch block

The function initialize() is used to give values in order to create a flawless GUI display. The command prompt and the scroll pane are initialized first. However, these were already provided to us by the professor and so we will be not discussing it in much detail.

All the static text that you can see on the GUI is defined by labels (such as "Magnetic levitation control", "Measurements", "Results in graphical form" etc) and all places where dynamic text is displayed and where user input is required such as "Reference Position" input is initialized by text view. A range of values can be toggled as input to the system effortlessly using the slider element. The different buttons such as start, stop, help have a dedicated listener which gets triggered upon activation.

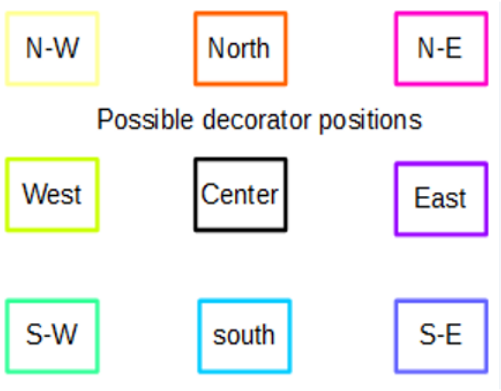


Figure 4: Terminology used by Eclipse to align elements relative to each other

A combo box is perhaps the simplest dynamic element in our GUI. It is like a drop-down list that contains connected serial ports to the computer and updates dynamically based on the connection status and the machine in which GUI runs on. In the background the serial.getcommports() function is used to detect the connected comports and store in the Serial port array. A for loop runs this function until all the available serial ports are added into the array.

After all the elements are initialized, we proceed with plotting the graph. The graph is plotted at an interval of 250 milli seconds by calling the updatedynamic() function. The Timer function takes in delay in milliseconds and Actionlistener as input parameters. The latter is triggered at regular intervals of the former.

Now comes the unique aspect of the GUI which is the live real time animation of ball traversing. In order to achieve this flawlessly, we have made an entirely different class called Balls() which is solely dedicated to animate a floating ball on the screen based on a numeric input corresponding to its real time location. We will be discussing in detail later, first be giving you insights about how we integrated this class into a single jFrame window to function ultimately as a single entity. We have created a panel within the main jFrame and set its colour to yellow for testing purposes so that it can be easily identified in the design page. This was

necessary to know the final bounds that the bouncing ball must play within. Inside this panel, we have created an object of the Balls() class. Please pay attention to this. We have created an object inside the panel and not in the runnable() or main() function. This enables the ball to update its position dynamically without closing the GUI window. Above and below the custom panel, we have added some images just for aesthetic purposes and they are a part of the main jframe.

13.2 Explanations of user defined functions

- private void CommandHandler(String cmds)

Input param: command

Return: void

As explained by the flowchart, Various messages are displayed corresponding to command given as

input to this function.

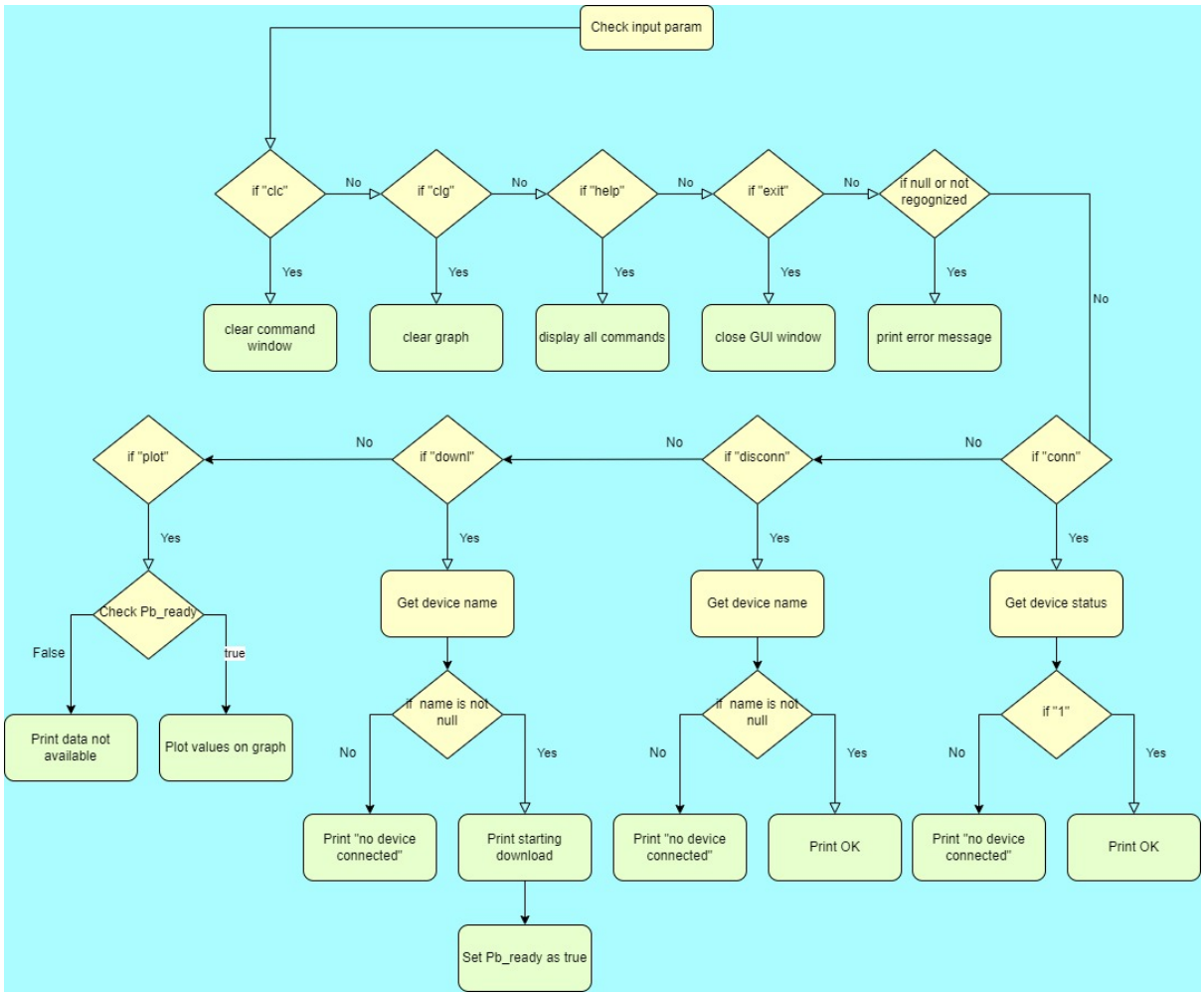


Figure 5: Command handler function (Flowchart 1)

- Private voidHelpfunction()

Input param: none

Return: void

It is simply used to print out the commands and their definitions in the command window.

- Private voidbroadcastslider()

We would say that this is the heart of the functions that we defined as it performs two main tasks. At first, it is used to update the slider value which is a global variable which ultimately connects the Balls class and the FpgaCtrlF class and secondly it is used to connect the GUI to the outside world by means of serial ports to perform SIL testing which is dealt with in the next chapter. Without this function, there wouldn't exist an attractive aesthetic GUI and we would have faced many problems as SIL wouldn't be possible.

The function starts of by defining comport parameters of the port where the Arduino UNO is connected to the device. The main parameters are baud rates, number of data bis, stop bis and parity. Then it opens the required serial port and displays a message if it was not successful. Then it sends the slider value over the serial port. This process is truly mesmerizing and will be explained in the next chapter. At the end, it closes the serial port.

- Public void Actionperformed(Actionevent e)

Input param: Actionevent e

Return: void

This function can also be explained by means of a flowchart. The flow of control logic is monitored

by a series of if conditions.

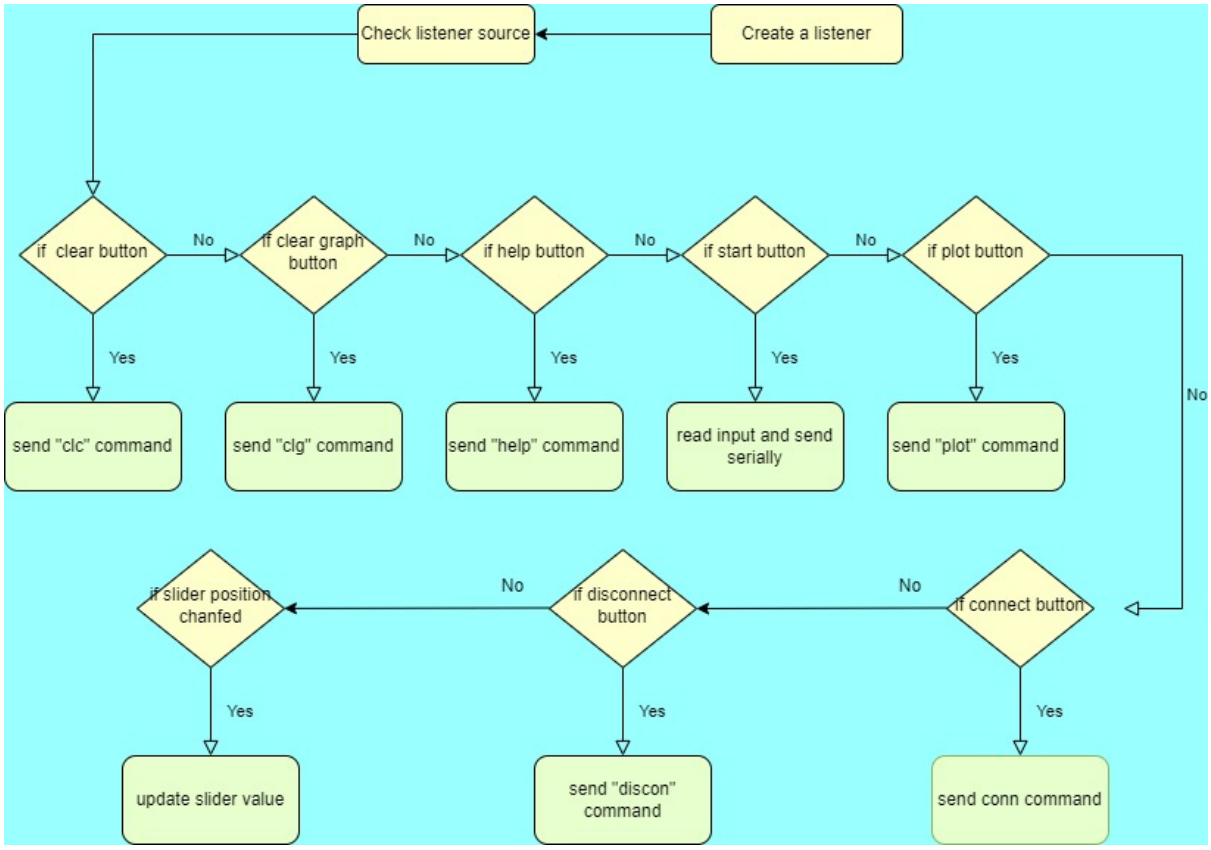


Figure 6: Action listener flowchart (Flowchart 2)

13.3 Balls.java

This is a pretty short class that creates a constructor of Ball class by specifying the colour of ball to be printed on the screen. The reason why we have made two classes (Balls class and Ball class) is that we thought of adding two balls on the GUI where one would we stationary and the other would be updating based on the real time value since every constructor of Ball class initialized will print an independent ball on the screen. But we did not go ahead with multiple balls because we simply did not have much place available in GUI and this would also create ambiguity to the user.

13.4 Ball.java

It contains simple logic to paint the ball on the screen dynamically based on the size and location of the ball. It has the following functions

Paintcomponent(): draws a circle and fills it with red colour

getPreferredSize(): It is used to define the size of the ball in pixels

run(): This function runs indefinitely and constantly updates the ball position on the screen using the move() and repaint() functions. The former function takes in slider

value as input and scales it proportionally on the screen so that it completely fits into the jPanel.

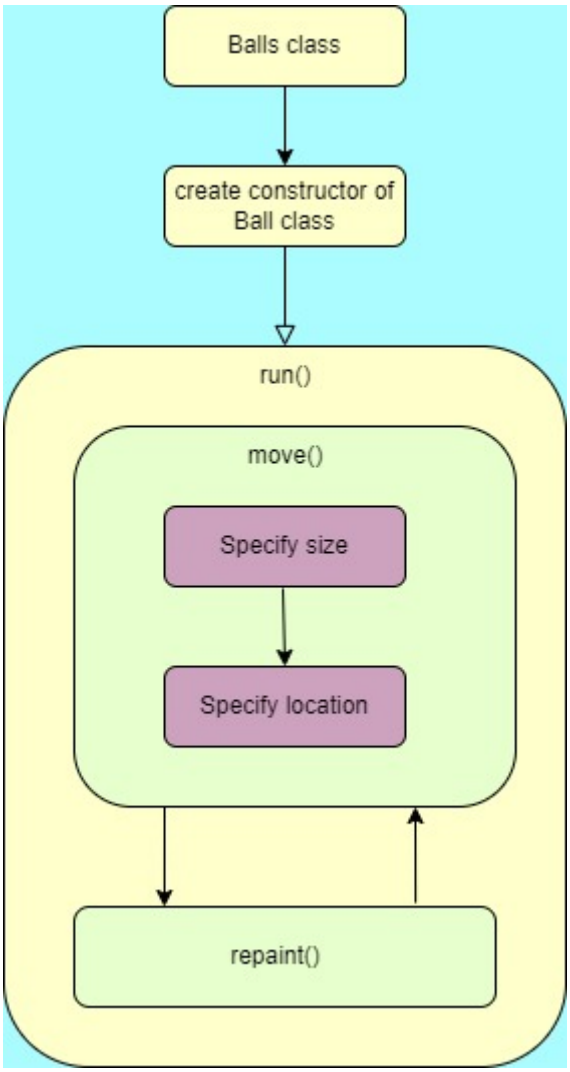


Figure 7: Ball and Balls class (Flowchart 3)

13.5 SerialNetw.java

This class serves as the backbone for the commandHandler() function discussed before. It undertakes necessary actions based on different commands entered by the user. It has dedicated functions to interact with the hardware as described below.

We start with defining the global variables used by functions in the class. Please refer to the following table:

Variable name	Data type	Description
NDEV_MAX	int	Max number of supported devices is 10
n_devices	int	Total number of connected devices
ser_devices	String array	Stores names of all connected devices
is_connected	boolean	True when device is connected otherwise false

- static void initSerialNetw()

We have designed the logic in such a way that the computer on which GUI runs and ultimately to which the magnetic levitation setup is connected can support a maximum of ten comports. During our testing phases we found out that only two comports are used so this number is fairly enough. Now a for loop runs from 0 to 10 and detects on which comport devices are connected. Whenever a comport is active, it adds then comport number to ser_device array.

- static int getNDev()

This one-line function returns the updated value of n_devices which stores the number of connected devices whenever it is called

- static StringgetDevName(int k):

It is also a fairly simple function that takes an integer as an input, compares if the integer is less than number of connected devices. This integer corresponds to the comport number.

It then extracts the name of the comport and sends it back as the return value is string.

- static int getDevStat(String d_name)

Input param: String device name
Return: int

This function is used to get the device status, whether it exists or not and its connection

status. We have defined 3 possible return values:

0: Device doesn't exist
1: Device exists but is not connected
2: Device exists and is connected

- void spConnect(String dev_name)

It is similar to the broadcastslider() function in the previous class. It takes in device name as an input parameter, searches the ser_device array, gets comport number and sets the communication parameters but for FPGA this time. Note that we have different comport parameters for FPGA and Arduino which we used for SIL testing. The baud rate we use in this case is 115200. It then opens up the required comport and throws an error message if it was not successful.

- void spDisconn(String dev_name): It is only used to disconnect from the device.

- static void SendString(String tstr): It sends the string received as input parameter into the output stream of the connected serial port.

- static String readString()

Not as simple as send string. The overall concept is to read data from the input stream of the serial port. For this the connection status must be checked and also availability of data on input stream must be checked otherwise runtime errors will be generated.

14 Legacy Code Testing

14.1 Developement of S function

The legacy code is tested with converting controller in a S function. The following set of MATLAB commands are utilised to develop a S function

```
def = legacy_code('initialize');
def.SourceFiles = {'controller_1.c'};
def.HeaderFiles = {'header.h'};
def.SFunctionName = 'controller';
def.OutputFcnSpec = 'int32 y1 = controller(int16 u1,int8 u2,int16 u3)';
legacy_code('sfcn_cmex_generate',def);
legacy_code('compile',def);
legacy_code('slblock_generate',def);
```

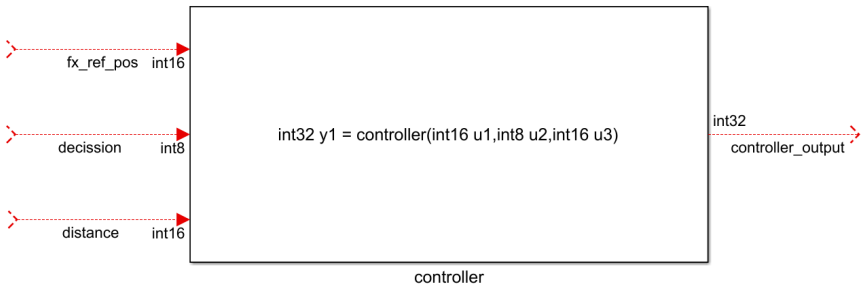


Figure 120: S-Function Controller

14.2 Test Bench Developement

A test bench is developed to implement the test the developed control logic directly on the Levitation Plant.

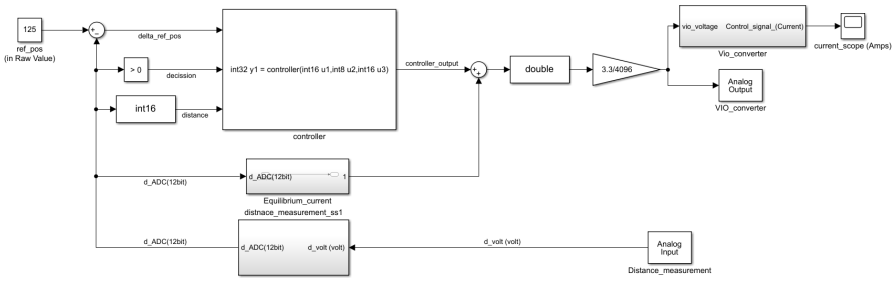


Figure 121: S-Function Controller Test bench

15 Conclusion and Results

Realization of electromechanical system into standard physics equation: Levitation functionality of the object ferromagnetic ball, with user controlled levitation positions was successfully achieved. Task begin with realizing the electromechanical system and collecting right physics equations to fit in the system's behaviour. Driving current and position relations, finding voltage to current ratios and approximate voltage drops, to drive the system efficiently.

Design and Practical control strategy: System's non linear behaviour is necessary to change into linear. After analysing different controllers and their response time along with their practicality with the system, Proportional Integral and derivative feedback controller considered to have suitable characteristics to satisfy the system's requirements.

Designing Model and Simulating the collected maths: To satisfy the correctness of collected physics equations and designed controlled system MATLAB and Simulink were employed, As they gives wholesome control and analyzation for each parameters and response of the system.

Implementation of Designed control strategy on a discrete system: Designed Model were implemented on Zynx 7000 System on chips, which incorporated PMOD ADC(analog to digital) and DAC(digital to analog) modules for data receiving and transmitting, to make all these communication happen UART(universal asynchronous receiver-transmitter) and PCI(Peripheral Component Interconnect) protocols were used.

User Friendly Interface for the system: Whole system is made user friendly and interactive by designing a Java based UI(user interface)

The following are the results of the the developed and implemented control logic

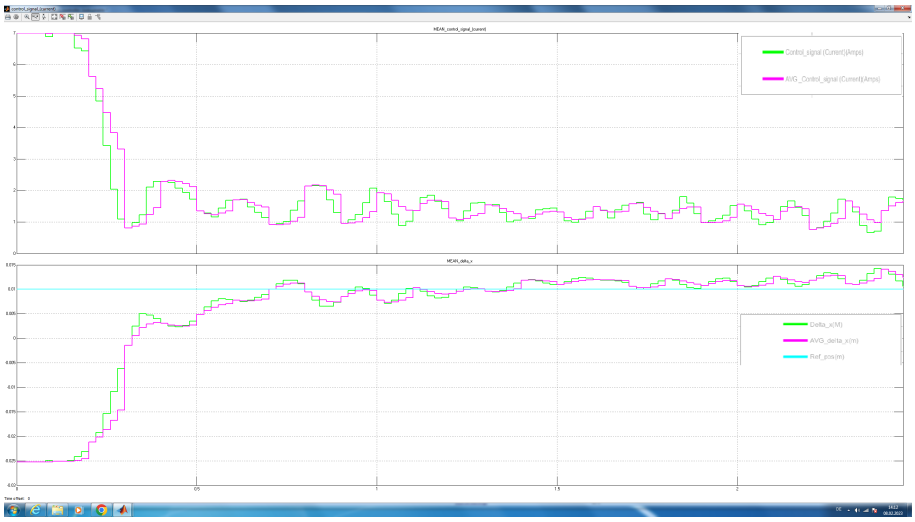


Figure 122: Step Input Response

The step response analysis of the system demonstrates the settling time, which indicates the time required for the system to reach a stable state. By implementing the developed

PID controller, we achieved a settling time of 700ms in the experimental setup. In simulation, the settling time was found to be 300ms. This comparison reveals a slight discrepancy between the experimental and simulated results in terms of settling time. The variation can be attributed to factors such as differences between the mathematical model and the physical system, practical limitations, and the impact of noise and disturbances during the experimental setup.

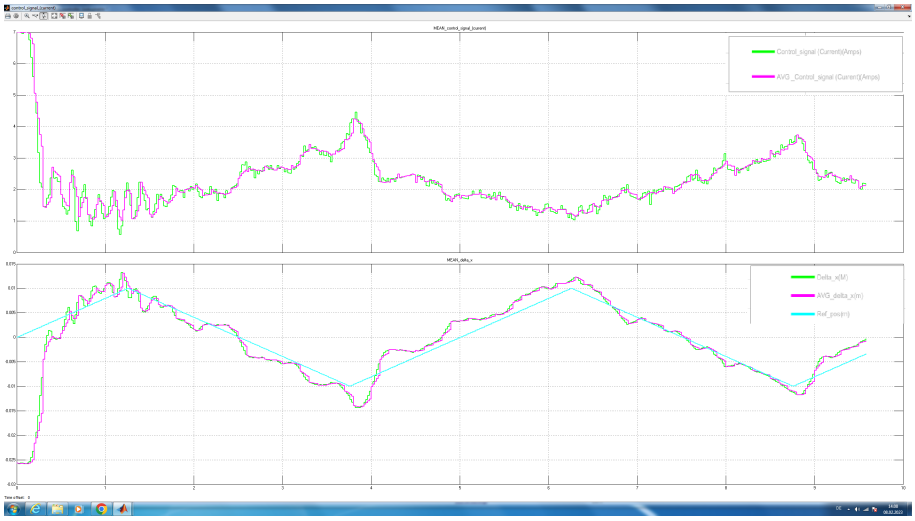


Figure 123: Triangular wave Response

To evaluate the system's response to sudden changes in input, various input waveforms were tested, including Trapezoidal wave,sinusoidal waves, and triangular waves. Among these waveforms, the triangular wave proved to be the most effective for assessing the system's behavior under sudden changes. During the experimental tests, it was observed that when using a poorly tuned controller, the levitated ball experienced overshoot in response to the sudden input change. The overshoot, in our case, was measured to be approximately 2mm.

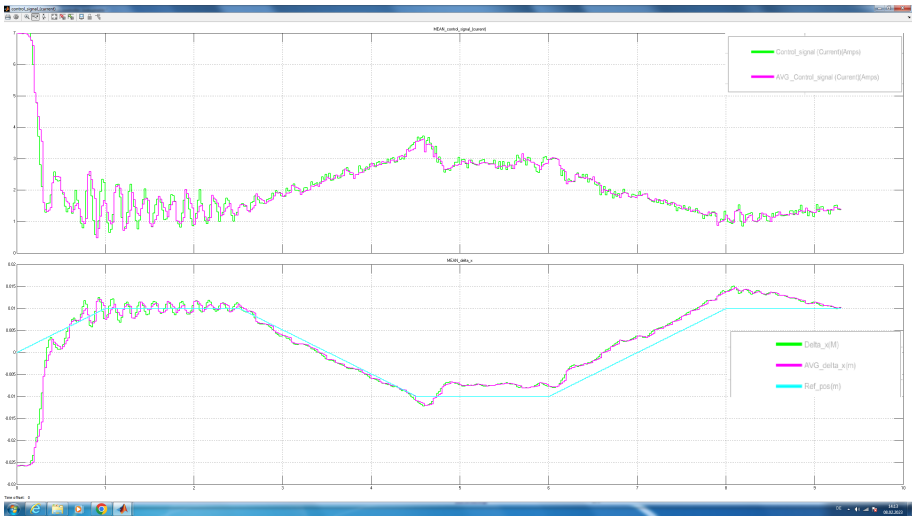


Figure 124: Trapezoidal Input Response

The Trapezoidal waveform was a good measure of testing the steady state stability. a constant steady state error of $\pm 1mm$is observed.

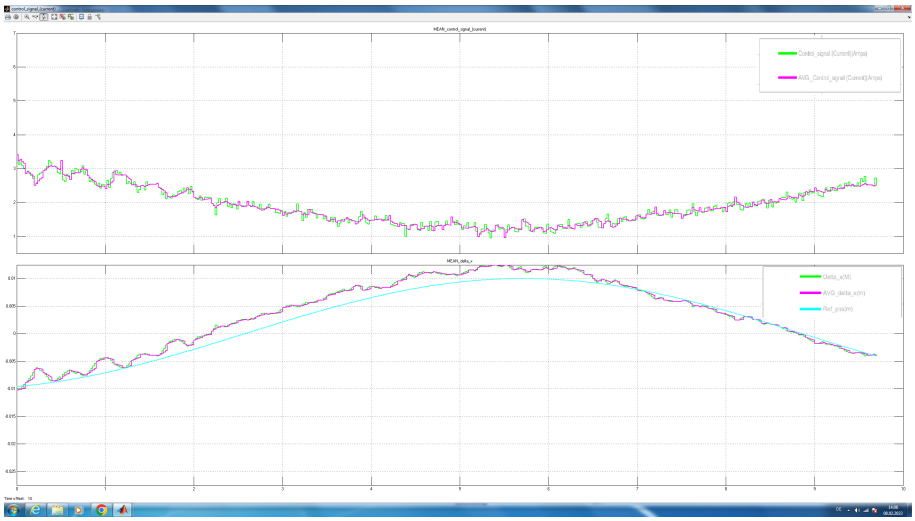


Figure 125: Sine Wave Input Response

The aforementioned results provide insights into the system's performance, specifically regarding the settling time and peak overshoot. However, it is crucial to acknowledge that there is a small deviation between the practical system and its expected modelled behavior. Several factors contribute to this deviation, including:

Nonlinear Nature of the System: The levitation system exhibits nonlinear characteristics due to various factors such as magnetic field variations, nonlinearities in the electromagnet, and changes in the air gap between the levitated object and the electromagnet. These nonlinearities can introduce deviations from the expected response, affecting the overall system performance.

Electromagnetic Interference (EMI): The presence of the electromagnet can induce electromagnetic interference, which may interfere with the distance measurement sensor. This interference can lead to inaccuracies in distance measurements, affecting the control system's ability to accurately regulate the levitation.

High-Gain Controller Amplifying Measurement Noise: The utilization of high-gain controllers, such as the PID controller, can amplify measurement noise present in the system. This noise can originate from sensor inaccuracies, electrical noise, or environmental disturbances. The amplification of measurement noise can impact the stability and precision of the control system, contributing to the observed deviation from the expected behavior.

Below are the links of Videos depicting the results .

- Sine Waveform : https://youtu.be/ZpyJ5J2iA3o
- Triangular Waveform : https://youtu.be/weLJ4MvTFiY
- Trapezoidal Waveform : https://youtu.be/48ayU-JoCHI

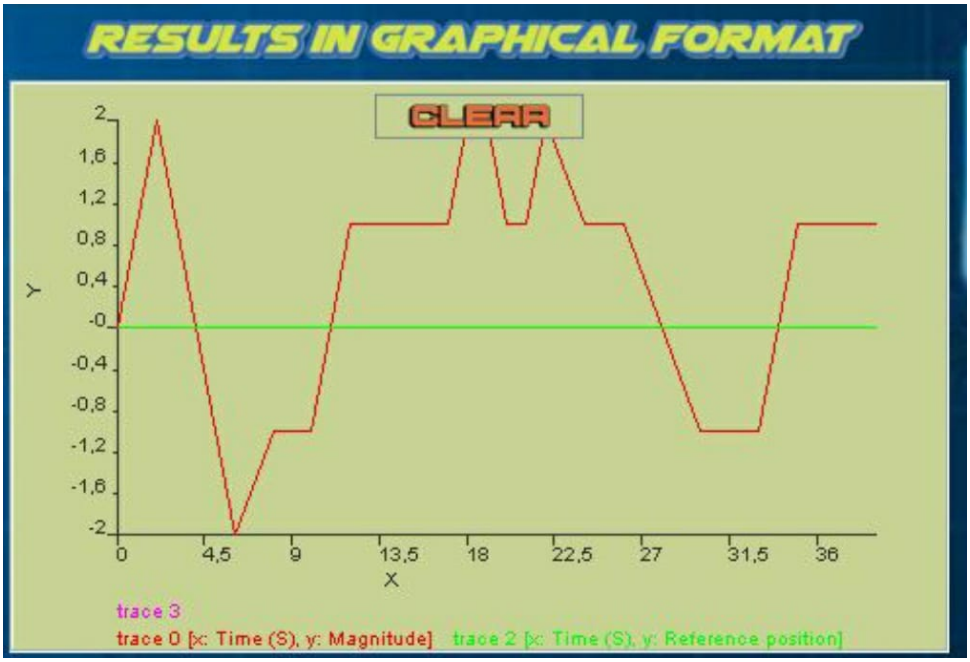


Figure 126: Displacement of object being levitated in magnetic field over time.

Explanation of result from GUI graph with reference as 0mm

The graph shows the displacement of the ball being levitated in a magnetic field over time. The green line represents the reference position, and the red line represents the actual position of the object. The objective of the magnetic levitation setup is to have the red line and green line coincide, meaning that there is no error in the levitation. the sampling time was 1 millisecond

The graph shows that the red line and green line are not perfectly coincident. There is a small amount of error, but it is within acceptable limits. The error is likely due to a number of factors, including imperfections in the magnets and the levitation system.

The error is more pronounced at the beginning of the test, and it decreases over time. This is likely because the system is becoming more stable as it operates. The error is also not constant, and there are some spikes in the error. These spikes may be due to disturbances in the system.

Overall, the graph shows that the magnetic levitation setup is performing well. The error is small, and it is within acceptable limits. As the setup is further refined, the error decreases.

16 Future Work

Revised report section on future work:

The current implementation of this work focuses on the selection of the most practical control technique, which is the PID controller. However, high-gain PID controllers have certain limitations, such as amplifying sensing noise and potentially saturating actuators if not designed with appropriate considerations for system limits. While high-gain PID controllers are popular in the market due to their off-the-shelf availability and tunability based on experience, turnkey projects such as magnetic levitation require finely tuned controllers that are not based on guesswork but on data from simulation and analysis of this data. Consequently, there is always a small mismatch between the mathematical model of the system and the actual system.

In order to address these limitations and improve the control system, there are two potential areas for future work.

Firstly, employing an optimal control technique such as Linear Quadratic Regulator (LQR) could provide stricter control over the applied current and settling time. This would enhance the overall performance of the system.

Secondly, reducing the gap between the prototyping system and the system on which the controller will be implemented is crucial. Due to significant deviations in input and output impedance between the prototype system and FPGA, the developed control logic exhibits small performance errors. This necessitates additional work to fine-tune the controller from an FPGA perspective. Introducing a system like D-Space controller would be a valuable addition as it eliminates the impedance gap and makes the prototyping system more representative of the implementation system.

Furthermore, as magnetic levitation is increasingly being utilized in locomotives, where the locomotive's balance relies on electromagnetic forces and varies with the number of passengers, it is essential to test the robustness of the implemented system. Implementing a robust control technique would be necessary to ensure the stability and reliability of the system under varying conditions. In such case a controller shall be tested with variation of the parameters and possible effect of them on the reliability of the system.

The most effective control strategy is to use a adaptive controller to handle the variation of weight of the object to be levitated.

References

[1] *AD7476 datasheet:* URL: https://media.digikey.com/pdf/Data%20Sheets/Analog%20Devices%20PDFs/AD7476,77,78.pdf.

[2] K.J. Åström and T. Hägglund. *PID Controllers*. Setting the standard for automation. International Society for Measurement and Control, 1995. ISBN: 9781556175169. URL: https://books.google.de/books?id=FsyhngEACAAJ.

[3] Peter Balko and Danica Rosinová. "Modeling of magnetic levitation system". In: *2017 21st International Conference on Process Control (PC)*. 2017, pp. 252–257. DOI: 10.1109/PC.2017.7976222.

[4] *Digilent dac:* URL: https://digilent.com/shop/pmod-da2-two-12-bit-d-a-outputs/.

[5] *Digilent PmodTM:* URL: https://digilent.com/reference/_media/reference/pmod/pmod-interface-specification-1_3_1.pdf.

[6] *Distance Sensor SICK (DT20HI)*. 2023. URL: https://www.sick.com/de/en/distance-sensors/displacement-measurement-sensors/dt20-hi/c/g176377.

[7] W.O. Galitz. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. Wiley Desktop Editions. Wiley, 2007. ISBN: 9780470146224. URL: https://books.google.de/books?id=Q3Xp%5C_Awu49sC.

[8] *Introduction Overview of Magnetic Levitation (MAGLEV) Train System*. 2023. URL: https://ijisrt.com/wp-content/uploads/2018/03/Introduction-Overview-of-Magnetic-Levitation-MAGLEV-Train-System-1.pdf.

[9] *Magnetic Levitation*. 2023. URL: https://en.wikipedia.org/wiki/Magnetic_levitation#:~:text=Maglev%2C%20or%20magnetic%20levitation%2C%20is,than%20wheeled%20mass%20transit%20systems..

[10] Mathworkd. *MATLAB App Building*. Setting the standard for automation. Mathworks, 2022b. URL: https://www.mathworks.com/help/pdf_doc/matlab/creating_guis.pdf.

[11] MATLAB. *Simulink Desktop Realtime Guide*. Matlab Help Documents. MATLAB, 2022a.

[12] Prof. Dr.-Ing Kai Mueller: "User logic implementation code". Accessed on July 5, 2023. 2023.

[13] *PCI Card*. 2023. URL: https://community.fs.com/blog/pcie-card-selection-guide.html.

[14] *Pmod AD1:* URL: https://digilent.com/reference/pmod/pmodad1/reference-manual.

[15] *Pmod Peripheral Youtube :* Nov. 2012. URL: https://www.youtube.com/watch?v=q8psdwGl5Qs&ab_channel=Digilent%2CInc..

[16] *Product Breif Zynq 7000*. 2023. URL: https://www.xilinx.com/content/dam/xilinx/support/documents/product-briefs/zynq-7000-product-brief.pdf.

[17] *Serial Peripheral Interface:* June 2023. URL: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface.

[18] Mohamed Sherif, David Victor, and Ayman El-Badawy. “Real-Time Control of a Magnetic Levitation System”. In: *2019 31st International Conference on Microelectronics (ICM)*. 2019, pp. 280–283. DOI: 10.1109/ICM48031.2019.9021705.

[19] Adam P. Taylor. *How to Use Interrupts on the Zynq SoC*. 2014.

[20] Lia Dolga Valer Dolga. “MODELLING AND SIMULATION OF A MAGNETIC LEVITATION SYSTEM”. In: *ANNALS of the ORADEA UNIVERSITY. Fascicle of Management and Technological Engineering, Volume VI (XVI), 2007*. Vol. VI. XVI. 2019, pp. 1118–1124.

[21] *VIO Amplifier*. 2023. URL: https://www.elmomc.com/product/guitar/.

[22] *Zedboard Avnet:* URL: https://www.avnet.com/wps/wcm/connect/onesite/a43adf00-158c-4614-b2c7-4a7f35b53f25/FY23_800_ZedBoard_Product_Brief_r2.pdf?MOD=AJPERES.

[23] *Zedboard ZYNQ-7000 ARM/FPGA SOC Development Board*. URL: https://digilent.com/shop/zedboard-zynq-7000-arm-fpga-soc-development-board/.

A User logic implementation code[12]:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity pmadda_v1_0_S00_AXI is
        generic (
                -- Users to add parameters here

                -- User parameters ends
                -- Do not modify the parameters beyond this line

                -- Width of S_AXI data bus
                C_S_AXI_DATA_WIDTH        : integer        := 32;
                -- Width of S_AXI address bus
                C_S_AXI_ADDR_WIDTH        : integer        := 4
        );
        port (
                -- Users to add ports here
        uLED : out std_logic_vector(7 downto 0);
        uSwitch : in std_logic_vector(7 downto 0);
        uPushB : in std_logic_vector(4 downto 0);
        uDA2csq : out std_logic;
        uDA2Sclk : out std_logic;
        usdoDA_1 : out std_logic;
        usdoDA_0 : out std_logic;
        uAD1csq : out std_logic;
```

```
uAD1Sclk : out std_logic;
usdiAD_1 : in std_logic;
usdiAD_0 : in std_logic;
-- User ports ends
-- Do not modify the ports beyond this line

-- Global Clock Signal
S_AXI_ACLK        : in std_logic;
-- Global Reset Signal. This Signal is Active LOW
S_AXI_ARESETN        : in std_logic;
-- Write address (issued by master, accepted by Slave)
S_AXI_AWADDR        : in std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
-- Write channel Protection type. This signal indicates the
-- privilege and security level of the transaction, and whether
-- the transaction is a data access or an instruction access.
S_AXI_AWPROT        : in std_logic_vector(2 downto 0);
-- Write address valid. This signal indicates that the master signaling
-- valid write address and control information.
S_AXI_AWVALID        : in std_logic;
-- Write address ready. This signal indicates that the slave is ready
-- to accept an address and associated control signals.
S_AXI_AWREADY        : out std_logic;
-- Write data (issued by master, accepted by Slave)
S_AXI_WDATA        : in std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
-- Write strobes. This signal indicates which byte lanes hold
-- valid data. There is one write strobe bit for each eight
-- bits of the write data bus.
S_AXI_WSTRB        : in std_logic_vector((C_S_AXI_DATA_WIDTH/8)-1 downto 0);
-- Write valid. This signal indicates that valid write
-- data and strobes are available.
S_AXI_WVALID        : in std_logic;
-- Write ready. This signal indicates that the slave
-- can accept the write data.
S_AXI_WREADY        : out std_logic;
-- Write response. This signal indicates the status
-- of the write transaction.
S_AXI_BRESP        : out std_logic_vector(1 downto 0);
-- Write response valid. This signal indicates that the channel
-- is signaling a valid write response.
S_AXI_BVALID        : out std_logic;
-- Response ready. This signal indicates that the master
-- can accept a write response.
S_AXI_BREADY        : in std_logic;
-- Read address (issued by master, acceped by Slave)
S_AXI_ARADDR        : in std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
-- Protection type. This signal indicates the privilege
-- and security level of the transaction, and whether the
-- transaction is a data access or an instruction access.
```

```
        S_AXI_ARPROT        : in std_logic_vector(2 downto 0);
        -- Read address valid. This signal indicates that the channel
        -- is signaling valid read address and control information.
        S_AXI_ARVALID       : in std_logic;
        -- Read address ready. This signal indicates that the slave is
        -- ready to accept an address and associated control signals.
        S_AXI_ARREADY       : out std_logic;
        -- Read data (issued by slave)
        S_AXI_RDATA         : out std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
        -- Read response. This signal indicates the status of the
        -- read transfer.
        S_AXI_RRESP         : out std_logic_vector(1 downto 0);
        -- Read valid. This signal indicates that the channel is
        -- signaling the required read data.
        S_AXI_RVALID        : out std_logic;
        -- Read ready. This signal indicates that the master can
        -- accept the read data and response information.
        S_AXI_RREADY        : in std_logic
        );
end pmadda_v1_0_S00_AXI;

architecture arch_imp of pmadda_v1_0_S00_AXI is

        -- AXI4LITE signals
        signal axi_awaddr       : std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
        signal axi_awready       : std_logic;
        signal axi_wready       : std_logic;
        signal axi_bresp       : std_logic_vector(1 downto 0);
        signal axi_bvalid       : std_logic;
        signal axi_araddr       : std_logic_vector(C_S_AXI_ADDR_WIDTH-1 downto 0);
        signal axi_arready       : std_logic;
        signal axi_rdata       : std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
        signal axi_rresp       : std_logic_vector(1 downto 0);
        signal axi_rvalid       : std_logic;

        -- Example-specific design signals
        -- local parameter for addressing 32 bit / 64 bit C_S_AXI_DATA_WIDTH
        -- ADDR_LSB is used for addressing 32/64 bit registers/memories
        -- ADDR_LSB = 2 for 32 bits (n downto 2)
        -- ADDR_LSB = 3 for 64 bits (n downto 3)
        constant ADDR_LSB  : integer := (C_S_AXI_DATA_WIDTH/32)+ 1;
        constant OPT_MEM_ADDR_BITS : integer := 1;
        ------------------------------------------------
        ---- Signals for user logic register space example
        ------------------------------------------------
        ---- Number of Slave Registers 4
        signal slv_reg0        :std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
        signal slv_reg1        :std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
```

```
    signal slv_reg2        :std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
    signal slv_reg3        :std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
    signal slv_reg_rden        : std_logic;
    signal slv_reg_wren        : std_logic;
    signal reg_data_out        :std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
    signal byte_index        : integer;


-- DAC PMODDA2
COMPONENT dac2lane is
    GENERIC ( USPI_SIZE : INTEGER := 16 );
    Port ( resetn : in STD_LOGIC;
           bclk : in STD_LOGIC;
           spi_clkp : in STD_LOGIC;
           start : in STD_LOGIC;
           done : out STD_LOGIC;
           scsq : out STD_LOGIC;
           sclk : out STD_LOGIC;
           sdo_0 : out STD_LOGIC;
           sdo_1 : out STD_LOGIC;
           sndData_0 : in STD_LOGIC_VECTOR (USPI_SIZE-1 downto 0);
           sndData_1 : in STD_LOGIC_VECTOR (USPI_SIZE-1 downto 0));
end COMPONENT dac2lane;

-- ADC PMODAD1
COMPONENT adc2lane is
    GENERIC ( USPI_SIZE : INTEGER := 16 );
    Port ( resetn : in STD_LOGIC;
           bclk : in STD_LOGIC;
           spi_clkp : in STD_LOGIC;
           start : in STD_LOGIC;
           done : out STD_LOGIC;
           scsq : out STD_LOGIC;
           sclk : out STD_LOGIC;
           sdi_0 : in STD_LOGIC;
           sdi_1 : in STD_LOGIC;
           rcvData_0 : out STD_LOGIC_VECTOR (USPI_SIZE-1 downto 0);
           rcvData_1 : out STD_LOGIC_VECTOR (USPI_SIZE-1 downto 0));
end COMPONENT adc2lane;

CONSTANT BCLK_DIVIDE : INTEGER := 50;
SUBTYPE Counter_type IS INTEGER RANGE 0 to BCLK_DIVIDE-1;
SIGNAL start_dac, start_adc : std_logic;
SIGNAL done_dac, done_adc : std_logic;
SIGNAL spiclk_p : std_logic;
SIGNAL ADC_Data : STD_LOGIC_VECTOR(31 downto 0);
```

```
begin
        -- I/O Connections assignments

        S_AXI_AWREADY        <= axi_awready;
        S_AXI_WREADY        <= axi_wready;
        S_AXI_BRESP        <= axi_bresp;
        S_AXI_BVALID        <= axi_bvalid;
        S_AXI_ARREADY        <= axi_arready;
        S_AXI_RDATA        <= axi_rdata;
        S_AXI_RRESP        <= axi_rresp;
        S_AXI_RVALID        <= axi_rvalid;
        -- Implement axi_awready generation
        -- axi_awready is asserted for one S_AXI_ACLK clock cycle when both
        -- S_AXI_AWVALID and S_AXI_WVALID are asserted. axi_awready is
        -- de-asserted when reset is low.

process (S_AXI_ACLK)
begin
  if rising_edge(S_AXI_ACLK) then
    if S_AXI_ARESETN = '0' then
      axi_awready <= '0';
    else
      if (axi_awready = '0' and S_AXI_AWVALID = '1' and S_AXI_WVALID = '1')then
        -- slave is ready to accept write address when
        -- there is a valid write address and write data
        -- on the write address and data bus. This design
        -- expects no outstanding transactions.
        axi_awready <= '1';
      else
        axi_awready <= '0';
      end if;
    end if;
  end if;
end process;

-- Implement axi_awaddr latching
-- This process is used to latch the address when both
-- S_AXI_AWVALID and S_AXI_WVALID are valid.

process (S_AXI_ACLK)
begin
  if rising_edge(S_AXI_ACLK) then
    if S_AXI_ARESETN = '0' then
      axi_awaddr <= (others => '0');
    else
      if (axi_awready = '0' and S_AXI_AWVALID = '1' and S_AXI_WVALID = '1') then
        -- Write Address latching
        axi_awaddr <= S_AXI_AWADDR;
```

```
      end if;
    end if;
  end if;
end process;

-- Implement axi_wready generation
-- axi_wready is asserted for one S_AXI_ACLK clock cycle when both
-- S_AXI_AWVALID and S_AXI_WVALID are asserted. axi_wready is
-- de-asserted when reset is low.

process (S_AXI_ACLK)
begin
  if rising_edge(S_AXI_ACLK) then
    if S_AXI_ARESETN = '0' then
      axi_wready <= '0';
    else
      if (axi_wready = '0' and S_AXI_WVALID = '1' and S_AXI_AWVALID = '1') then
          -- slave is ready to accept write data when
          -- there is a valid write address and write data
          -- on the write address and data bus. This design
          -- expects no outstanding transactions.
          axi_wready <= '1';
      else
        axi_wready <= '0';
      end if;
    end if;
  end if;
end process;

-- Implement memory mapped register select and write logic generation
-- The write data is accepted and written to memory mapped registers when
-- axi_awready, S_AXI_WVALID, axi_wready and S_AXI_WVALID are asserted.
-- select byte enables of slave registers while writing.
-- These registers are cleared when reset (active low) is applied.
-- Slave register write enable is asserted when valid address and data are avb
-- and the slave is ready to accept the write address and write data.
slv_reg_wren <= axi_wready and S_AXI_WVALID and axi_awready and S_AXI_AWVALID ;

process (S_AXI_ACLK)
variable loc_addr :std_logic_vector(OPT_MEM_ADDR_BITS downto 0);
begin
  if rising_edge(S_AXI_ACLK) then
    if S_AXI_ARESETN = '0' then
      slv_reg0 <= (others => '0');
      slv_reg1 <= (others => '0');
      slv_reg2 <= (others => '0');
      slv_reg3 <= (others => '0');
      start_dac <= '0';
```

```
      start_adc <= '0';
else
  start_dac <= '0';
      start_adc <= '0';
  loc_addr := axi_awaddr(ADDR_LSB + OPT_MEM_ADDR_BITS downto ADDR_LSB);
  if (slv_reg_wren = '1') then
    case loc_addr is
      when b"00" =>
        for byte_index in 0 to (C_S_AXI_DATA_WIDTH/8-1) loop
          if ( S_AXI_WSTRB(byte_index) = '1' ) then
            -- Respective byte enables are asserted as per write strobes
            -- slave registor 0
            slv_reg0(byte_index*8+7 downto byte_index*8) <= S_AXI_WDATA(
            byte_index*8+7 downto byte_index*8);
          end if;
        end loop;
      when b"01" =>
        start_dac <= '1';
        for byte_index in 0 to (C_S_AXI_DATA_WIDTH/8-1) loop
          if ( S_AXI_WSTRB(byte_index) = '1' ) then
            -- Respective byte enables are asserted as per write strobes
            -- slave registor 1
            slv_reg1(byte_index*8+7 downto byte_index*8) <= S_AXI_WDATA(
            byte_index*8+7 downto byte_index*8);
          end if;
        end loop;
      when b"10" =>
        start_adc <= '1';
        for byte_index in 0 to (C_S_AXI_DATA_WIDTH/8-1) loop
          if ( S_AXI_WSTRB(byte_index) = '1' ) then
            -- Respective byte enables are asserted as per write strobes
            -- slave registor 2
    slv_reg2(byte_index*8+7 downto byte_index*8) <=
          end if;
        end loop;
      when b"11" =>
        for byte_index in 0 to (C_S_AXI_DATA_WIDTH/8-1) loop
          if ( S_AXI_WSTRB(byte_index) = '1' ) then
            -- Respective byte enables are asserted as per write strobes
            -- slave registor 3
            slv_reg3(byte_index*8+7 downto byte_index*8) <=
            S_AXI_WDATA(byte_index*8+7 downto byte_index*8);
          end if;
        end loop;
      when others =>
        slv_reg0 <= slv_reg0;
        slv_reg1 <= slv_reg1;
        slv_reg2 <= slv_reg2;
```

```
            slv_reg3 <= slv_reg3;
        end case;
      end if;
    end if;
  end if;
end process;

-- Implement write response logic generation
-- The write response and response valid signals are asserted by the slave
-- when axi_wready, S_AXI_WVALID, axi_wready and S_AXI_WVALID are asserted.
-- This marks the acceptance of address and indicates the status of
-- write transaction.

process (S_AXI_ACLK)
begin
  if rising_edge(S_AXI_ACLK) then
    if S_AXI_ARESETN = '0' then
      axi_bvalid  <= '0';
      axi_bresp   <= "00"; --need to work more on the responses
    else
      if (axi_awready = '1' and S_AXI_AWVALID = '1' and axi_wready = '1' and
          S_AXI_WVALID = '1' and axi_bvalid = '0'  ) then
        axi_bvalid <= '1';
        axi_bresp  <= "00";
      elsif (S_AXI_BREADY = '1' and axi_bvalid = '1') then
      --check if bready is asserted while bvalid is high)
        axi_bvalid <= '0';
      -- (there is a possibility that bready is always asserted high)
      end if;
    end if;
  end if;
end process;

-- Implement axi_arready generation
-- axi_arready is asserted for one S_AXI_ACLK clock cycle when
-- S_AXI_ARVALID is asserted. axi_awready is
-- de-asserted when reset (active low) is asserted.
-- The read address is also latched when S_AXI_ARVALID is
-- asserted. axi_araddr is reset to zero on reset assertion.

process (S_AXI_ACLK)
begin
  if rising_edge(S_AXI_ACLK) then
    if S_AXI_ARESETN = '0' then
              axi_arready <= '0';
              axi_araddr  <= (others => '1');
            else
              if (axi_arready = '0' and S_AXI_ARVALID = '1') then
```

```
        -- indicates that the slave has acceped the valid read address
        axi_arready <= '1';
        -- Read Address latching
        axi_araddr  <= S_AXI_ARADDR;
      else
        axi_arready <= '0';
      end if;
    end if;
  end if;
end process;

-- Implement axi_arvalid generation
-- axi_rvalid is asserted for one S_AXI_ACLK clock cycle when both
-- S_AXI_ARVALID and axi_arready are asserted. The slave registers
-- data are available on the axi_rdata bus at this instance. The
-- assertion of axi_rvalid marks the validity of read data on the
-- bus and axi_rresp indicates the status of read transaction.axi_rvalid
-- is deasserted on reset (active low). axi_rresp and axi_rdata are
-- cleared to zero on reset (active low).
process (S_AXI_ACLK)
begin
  if rising_edge(S_AXI_ACLK) then
    if S_AXI_ARESETN = '0' then
      axi_rvalid <= '0';
      axi_rresp  <= "00";
   else
    if (axi_arready = '1' and S_AXI_ARVALID = '1' and axi_rvalid = '0') then
        -- Valid read data is available at the read data bus
        axi_rvalid <= '1';
        axi_rresp  <= "00"; -- 'OKAY' response
      elsif (axi_rvalid = '1' and S_AXI_RREADY = '1') then
        -- Read data is accepted by the master
        axi_rvalid <= '0';
      end if;
    end if;
  end if;
end process;

-- Implement memory mapped register select and read logic generation
-- Slave register read enable is asserted when valid address is available
-- and the slave is ready to accept the read address.
slv_reg_rden <= axi_arready and S_AXI_ARVALID and (not axi_rvalid) ;

process (slv_reg3, axi_araddr, S_AXI_ARESETN, slv_reg_rden,
         uSwitch, uPushB, done_adc, done_dac, ADC_Data)
variable loc_addr :std_logic_vector(OPT_MEM_ADDR_BITS downto 0);
begin
    -- Address decoding for reading registers
```

```
        loc_addr := axi_araddr(ADDR_LSB + OPT_MEM_ADDR_BITS downto ADDR_LSB);
        case loc_addr is
          when b"00" =>
            reg_data_out <= x"0000" & "000" & uPushB & uSwitch;
          when b"01" =>
            reg_data_out <= x"0000000" & "00" & done_adc & done_dac;
          when b"10" =>
            reg_data_out <= ADC_Data;
          when b"11" =>
            reg_data_out <= slv_reg3;
          when others =>
            reg_data_out  <= (others => '0');
        end case;
    end process;

    -- Output register or memory read data
    process( S_AXI_ACLK ) is
    begin
      if (rising_edge (S_AXI_ACLK)) then
        if ( S_AXI_ARESETN = '0' ) then
          axi_rdata  <= (others => '0');
        else
          if (slv_reg_rden = '1') then
            -- When there is a valid read address (S_AXI_ARVALID) with
            -- acceptance of read address by the slave (axi_arready),
            -- output the read dada
            -- Read address mux
              axi_rdata <= reg_data_out;     -- register read data
          end if;
        end if;
      end if;
    end process;


    -- Add user logic here
uLED <= slv_reg0(7 downto 0);

-- BCLK_DIVIDE = 50 is 1 MHz SPI clock speed (maximum could be 20 MHz)
clkdiv_p : PROCESS(S_AXI_ACLK)
VARIABLE cnt : Counter_type := BCLK_DIVIDE-1;
BEGIN
    IF rising_edge(S_AXI_ACLK) THEN
        IF S_AXI_ARESETN = '0' THEN
            cnt := BCLK_DIVIDE - 1;
        ELSE
            spiclk_p <= '0';
            IF cnt=0 THEN
                spiclk_p <= '1';
```

```
                    cnt := BCLK_DIVIDE - 1;
                ELSE
                    cnt := cnt - 1;
                END IF;
            END IF;
        END IF;
    END PROCESS clkdiv_p;

    dacTX : dac2lane
        GENERIC MAP ( USPI_SIZE => 16 )
        PORT MAP ( resetn => S_AXI_ARESETN,
                   bclk => S_AXI_ACLK,
                   spi_clkp => spiclk_p,
                   start => start_dac,
                   done => done_dac,
                   scsq => uDA2csq,
                   sclk => uDA2Sclk,
                   sdo_0 => usdoDA_0,
                   sdo_1 => usdoDA_1,
                   sndData_0 => slv_reg1(15 downto 0),
                   sndData_1 => slv_reg1(31 downto 16));

    adcRX : adc2lane
        GENERIC MAP ( USPI_SIZE => 16 )
        PORT MAP ( resetn => S_AXI_ARESETN,
                   bclk => S_AXI_ACLK,
                   spi_clkp => spiclk_p,
                   start => start_adc,
                   done => done_adc,
                   scsq => uAD1csq,
                   sclk => uAD1Sclk,
                   sdi_0 => usdiAD_0,
                   sdi_1 => usdiAD_1,
                   rcvData_0 => ADC_Data(15 downto 0),
                   rcvData_1 => ADC_Data(31 downto 16));

        -- User logic ends

end arch_imp;
```

DAC State Diagram

```
----------------------------------------------------------------------------------
-- Company:          Univ. Bremerhaven
-- Engineer:         Kai Mueller
--
```

```
-- Create Date:     08.06.2016 00:15:32
-- Description:     PMODDA2 based on spipol0pha1 template (SPI CPOL=0 CPHA=1)
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity dac2lane is
GENERIC ( USPI_SIZE : INTEGER := 16 );
Port ( resetn : in STD_LOGIC;
        bclk : in STD_LOGIC;
        spi_clkp : in STD_LOGIC;
        start : in STD_LOGIC;
        done : out STD_LOGIC;
        scsq : out STD_LOGIC;
        sclk : out STD_LOGIC;
        sdo_0 : out STD_LOGIC;
        sdo_1 : out STD_LOGIC;
        sndData_0 : in STD_LOGIC_VECTOR (USPI_SIZE-1 downto 0);
        sndData_1 : in STD_LOGIC_VECTOR (USPI_SIZE-1 downto 0));
end dac2lane;

architecture Behavioral of dac2lane is

type state_type IS (sidle, sstartx, sstart_lo, sclk_lo, sclk_hi, stop_hi, stop_lo);
SIGNAL  state, next_state: state_type;
SIGNAL sclk_i, scsq_i, sdo_i_0, sdo_i_1 : STD_LOGIC;
SIGNAL wr_buf_0, wr_buf_1 : STD_LOGIC_VECTOR(USPI_SIZE-1 downto 0);
SIGNAL count : integer RANGE 0 TO USPI_SIZE-1;

type sslst_type IS (ssl_idle, ssl_start, ssl_run);
SIGNAL ssl_state, sslnxt_state: sslst_type;
SIGNAL start_i : STD_LOGIC;

begin

-- start/stop logic
-- proper handling of clock division
sslseq_proc: PROCESS(resetn, bclk, sslnxt_state)
BEGIN
    IF rising_edge(bclk) THEN
        IF resetn='0' THEN
            ssl_state <= ssl_idle;
        ELSE
            ssl_state <= sslnxt_state;
        END IF;
```

```
    END IF;
 END PROCESS sslseq_proc;

sslcmb_proc: PROCESS(ssl_state, state, start)
BEGIN
    sslnxt_state <= ssl_state;
    start_i <= '0';
    done <= '0';
    CASE ssl_state IS
        WHEN ssl_idle =>
            done <= '1';
            IF start='1' AND state=sidle THEN
                sslnxt_state <= ssl_start;
            END IF;
        WHEN ssl_start =>
            start_i <= '1';
            IF state=sstartx THEN
                sslnxt_state <= ssl_run;
            END IF;
        WHEN ssl_run =>
            IF state=sidle THEN
                sslnxt_state <= ssl_idle;
            END IF;
    END CASE;
END PROCESS sslcmb_proc;

-- spi logic
sseq_proc: PROCESS(resetn, bclk, next_state, sclk_i, scsq_i, sndData_0, sndData_1,
                   wr_buf_0, wr_buf_1, sdo_i_0, sdo_i_1, count, spi_clkp)
BEGIN
    IF rising_edge(bclk) THEN
        IF resetn='0' THEN
            state <= sidle;
            count <= USPI_SIZE-1;
        ELSIF spi_clkp='1' THEN
            IF next_state=sstartx THEN
                            wr_buf_0 <= sndData_0;
                            wr_buf_1 <= sndData_1;
                            count <= USPI_SIZE - 1;
                            ELSIF next_state=sclk_lo THEN
                                wr_buf_0 <= wr_buf_0(USPI_SIZE-2 downto 0) & '-';
                                wr_buf_1 <= wr_buf_1(USPI_SIZE-2 downto 0) & '-';
                            ELSIF next_state=sclk_hi THEN
                            count <= count - 1;
                            END IF;
                            state <= next_state;
                            sclk <= sclk_i;
                            scsq <= scsq_i;
```

```
                            sdo_0 <= sdo_i_0;
                            sdo_1 <= sdo_i_1;
                    END IF;
                END IF;
            END PROCESS sseq_proc;

scmb_proc: PROCESS(state, start_i, count, wr_buf_0, wr_buf_1)
BEGIN
    next_state <= state;
    sclk_i <= '0';
    scsq_i <= '0';
    sdo_i_0 <= '0';
    sdo_i_1 <= '0';
    CASE state IS
        WHEN sidle =>
            scsq_i <= '1';
            IF start_i='1' THEN
                next_state <= sstartx;
            END IF;
        WHEN sstartx =>
            next_state <= sstart_lo;
        WHEN sstart_lo =>
            sclk_i <= '1';
            sdo_i_0 <= wr_buf_0(USPI_SIZE-1);
            sdo_i_1 <= wr_buf_1(USPI_SIZE-1);
            next_state <= sclk_hi;
           WHEN sclk_hi =>
            sdo_i_0 <= wr_buf_0(USPI_SIZE-1);
                            sdo_i_1 <= wr_buf_1(USPI_SIZE-1);
            next_state <= sclk_lo;
            WHEN sclk_lo =>
            sclk_i <= '1';
            sdo_i_0 <= wr_buf_0(USPI_SIZE-1);
                            sdo_i_1 <= wr_buf_1(USPI_SIZE-1);
            IF count=0 THEN
                next_state <= stop_hi;
            ELSE
                next_state <= sclk_hi;
            END IF;
            WHEN stop_hi =>
            sdo_i_0 <= wr_buf_0(USPI_SIZE-1);
                            sdo_i_1 <= wr_buf_1(USPI_SIZE-1);
            next_state <= stop_lo;
             WHEN stop_lo =>
            scsq_i <= '1';
            next_state <= sidle;
          END CASE;
    END PROCESS scmb_proc;
```

```
end Behavioral;
```

ADC State Diagram

```
----------------------------------------------------------------------------------
-- Company:          Univ. Bremerhaven
-- Engineer:         Kai Mueller
--
-- Create Date:      08.06.2016 00:15:32
-- Description:      ADC 2 Lanes SPI based on spipol1pha0 template (CPOL=0 CPHA=1)
--
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity adc2lane is
    GENERIC ( USPI_SIZE : INTEGER := 16 );
    Port ( resetn : in STD_LOGIC;
       bclk : in STD_LOGIC;
       spi_clkp : in STD_LOGIC;
       start : in STD_LOGIC;
       done : out STD_LOGIC;
       scsq : out STD_LOGIC;
       sclk : out STD_LOGIC;
       sdi_0 : in STD_LOGIC;
       sdi_1 : in STD_LOGIC;
       rcvData_0 : out STD_LOGIC_VECTOR (USPI_SIZE-1 downto 0);
       rcvData_1 : out STD_LOGIC_VECTOR (USPI_SIZE-1 downto 0));
end adc2lane;

architecture Behavioral of adc2lane is

type state_type IS (sidle, sstartx, sclk_lo, sclk_hi, sstop_lo, sstop_hi);
SIGNAL  state, next_state: state_type;
SIGNAL sclk_i, scsq_i : STD_LOGIC;
SIGNAL rd_buf_0, rd_buf_1 : STD_LOGIC_VECTOR(USPI_SIZE-1 downto 0);
SIGNAL count : integer RANGE 0 TO USPI_SIZE-1;

type sslst_type IS (ssl_idle, ssl_start, ssl_run);
SIGNAL ssl_state, sslnxt_state: sslst_type;
SIGNAL start_i : STD_LOGIC;

begin

    rcvData_0 <= rd_buf_0;
```

```
rcvData_1 <= rd_buf_1;

    -- start/stop logic
    -- proper handling of clock division
    sslseq_proc: PROCESS(resetn, bclk, sslnxt_state)
    BEGIN
            IF rising_edge(bclk) THEN
                    IF resetn='0' THEN
                            ssl_state <= ssl_idle;
                    ELSE
                            ssl_state <= sslnxt_state;
                    END IF;
            END IF;
    END PROCESS sslseq_proc;

    sslcmb_proc: PROCESS(ssl_state, state, start)
    BEGIN
            sslnxt_state <= ssl_state;
            start_i <= '0';
            done <= '0';
            CASE ssl_state IS
                    WHEN ssl_idle =>
                            done <= '1';
                            IF start='1' AND state=sidle THEN
                               sslnxt_state <= ssl_start;
                            END IF;
                    WHEN ssl_start =>
                            start_i <= '1';
                            IF state=sstartx THEN
                               sslnxt_state <= ssl_run;
                            END IF;
                    WHEN ssl_run =>
                            IF state=sidle THEN
                            sslnxt_state <= ssl_idle;
                            END IF;
            END CASE;
    END PROCESS sslcmb_proc;

    -- spi logic
    sseq_proc: PROCESS(resetn, bclk, next_state, sclk_i, scsq_i,rd_buf_0,
                       rd_buf_1, sdi_0, sdi_1, count, spi_clkp)
    BEGIN
            IF rising_edge(bclk) THEN
                    IF resetn='0' THEN
                        state <= sidle;
                        count <= USPI_SIZE-1;
                    ELSIF spi_clkp='1' THEN
                       IF next_state=sstartx THEN
```

```
                        count <= USPI_SIZE - 1;
                    ELSIF next_state=sclk_lo THEN
                        count <= count - 1;
                        rd_buf_0 <= rd_buf_0(USPI_SIZE-2 downto 0) & sdi_0;
                        rd_buf_1 <= rd_buf_1(USPI_SIZE-2 downto 0) & sdi_1;
                    ELSIF next_state=sstop_lo THEN
                        rd_buf_0 <= rd_buf_0(USPI_SIZE-2 downto 0) & sdi_0;
                        rd_buf_1 <= rd_buf_1(USPI_SIZE-2 downto 0) & sdi_1;
                    END IF;
                    state <= next_state;
                    sclk <= sclk_i;
                    scsq <= scsq_i;
                    END IF;
            END IF;
        END PROCESS sseq_proc;

        scmb_proc: PROCESS(state, start_i, count)
        BEGIN
                next_state <= state;
                sclk_i <= '1';
                scsq_i <= '0';
                CASE state IS
                        WHEN sidle =>
                                scsq_i <= '1';
                                IF start_i='1' THEN
                                   next_state <= sstartx;
                                END IF;
                        WHEN sstartx =>
                             next_state <= sclk_hi;
                        WHEN sclk_hi =>
                             sclk_i <= '0';
                             IF count=0 THEN
                                next_state <= sstop_lo;
                             ELSE
                                next_state <= sclk_lo;
                             END IF;
                        WHEN sclk_lo =>
                                next_state <= sclk_hi;
                        WHEN sstop_lo =>
                                next_state <= sstop_hi;
                        WHEN sstop_hi =>
                                scsq_i <= '1';
                                next_state <= sidle;
                END CASE;
        END PROCESS scmb_proc;

end Behavioral;
```

B GUI code in eclipse IDE

```
1  package prjgui;
2
3  import info.monitorenter.gui.chart.Chart2D;
4  import info.monitorenter.gui.chart.ITrace2D;
5  import info.monitorenter.gui.chart.traces.Trace2DSimple;
6
7  import java.awt.EventQueue;
8
9  import javax.swing.JFrame;
10 import javax.swing.JScrollPane;
11
12 import java.awt.BorderLayout;
13
14 import javax.swing.JTextPane;
15 import javax.swing.JPanel;
16
17 import java.awt.Color;
18 import java.awt.event.ActionEvent;
19 import java.awt.event.ActionListener;
20
21 import javax.swing.SpringLayout;
22 import javax.swing.JLabel;
23 import javax.swing.SwingConstants;
24 import javax.swing.SwingUtilities;
25 import javax.swing.JTextField;
26 import javax.swing.Timer;
27 import javax.swing.text.BadLocationException;
28 import javax.swing.text.Document;
29 import javax.swing.text.SimpleAttributeSet;
30 import javax.swing.text.StyleConstants;
31
32 import java.awt.FlowLayout;
33 import java.io.*;
34 import java.util.Locale;
35
36 // import java.nio.charset.StandardCharsets;
37 // import java.nio.file.Files;
38 // import java.nio.file.Path;
39 // import java.nio.file.Paths;
40 // import java.nio.file.StandardOpenOption;
41 // import java.util.Iterator;
42 // import info.monitorenter.gui.chart.ITracePoint2D;
43
44 import javax.swing.border.EtchedBorder;
45
46 import prjgui.SerialNetw;
47 // import javax.swing.JButton;
48 // import java.awt.Font;
49 // import javax.swing.ImageIcon;
```

```
50 import javax.swing.UIManager;
51 import javax.swing.UnsupportedLookAndFeelException;
52
53 import java.awt.Font;
54 import java.awt.Graphics;
55 import java.awt.Graphics2D;
56 import java.awt.Image;
57 import java.awt.LayoutManager;
58 import java.awt.Rectangle;
59 import java.awt.RenderingHints;
60
61 import javax.swing.JList;
62 import javax.swing.JRadioButton;
63 import javax.swing.JSlider;
64 import javax.swing.JButton;
65 // import javax.swing.JComboBox;
66 // import javax.swing.DefaultComboBoxModel;
67 // import java.awt.SystemColor;
68 // import javax.swing.JTabbedPane;
69 import java.awt.BorderLayout;
70 import java.awt.Color;
71 import java.awt.Dimension;
72 import java.awt.Font;
73 import java.awt.event.ActionEvent;
74 import java.awt.event.ActionListener;
75
76 import javax.swing.BorderFactory;
77 import javax.swing.ButtonGroup;
78 import javax.swing.Icon;
79 import javax.swing.JButton;
80 import javax.swing.JComboBox;
81 import javax.swing.JComponent;
82 import javax.swing.JFrame;
83 import javax.swing.JLabel;
84 import javax.swing.JPanel;
85 import javax.swing.JRadioButton;
86 import javax.swing.JSlider;
87 import javax.swing.JTextField;
88 import javax.swing.event.ChangeEvent;
89 import javax.swing.event.ChangeListener;
90
91 import org.eclipse.swt.SWT;
92 import org.eclipse.swt.widgets.ToolBar;
93 import org.eclipse.swt.widgets.ToolItem;
94 import org.jfree.chart.ChartFactory;
95 import org.jfree.chart.ChartPanel;
96 import org.jfree.chart.JFreeChart;
97 import org.jfree.chart.plot.PlotOrientation;
98 import org.jfree.chart.plot.XYPlot;
99 import org.jfree.data.category.DefaultCategoryDataset;
100 import org.jfree.data.time.TimeSeries;
```

```
101 import org.jfree.data.time.TimeSeriesCollection;
102 import org.jfree.data.xy.XYDataset;
103 import org.jfree.data.xy.XYSeries;
104 import org.jfree.data.xy.XYSeriesCollection;
105
106 import com.fazecast.jSerialComm.SerialPort;
107
108
109 import java.awt.event.WindowAdapter;
110 import java.awt.event.WindowEvent;
111 import java.io.IOException;
112 import java.lang.reflect.InvocationTargetException;
113 import java.util.Random;
114 import java.util.Scanner;
115 import org.jfree.chart.ChartPanel;
116 import org.jfree.chart.ChartFactory;
117 import org.jfree.chart.JFreeChart;
118 //import org.jfree.ui.ApplicationFrame;
119 //import org.jfree.ui.RefineryUtilities;
120 import org.jfree.chart.plot.PlotOrientation;
121 import org.jfree.data.category.DefaultCategoryDataset;
122
123 import javax.swing.event.ChangeEvent;
124 import javax.swing.ImageIcon;
125 import java.awt.Toolkit;
126 public class FpgaCtrlF extends JFrame implements ActionListener{
127
128    public JFrame FG_frame;
129    private JTextField cmd_textField;
130    protected String dev_name;
131    public static JSlider slider;
132    public static JLabel lblNewLabel_8;
133    public static JButton STARTButton;
134    public static JButton STOPButton;
135    public static JButton HELPButton;
136    public static JButton CLEARCommand;
137    public static JButton ClearChart;
138    public static JComboBox comboBox;
139    public static SerialPort chosenPort;
140    public static int x=100,y=100;
141    public static double val;
142    //public static JPanel panel;
143    static FpgaCtrlF window;
144    /**
145     * Launch the application.
146     */
147    public static void main(String[] args) {
148
149        SerialNetw.initSerialNetw();
150       EventQueue.invokeLater(new Runnable() {
151          public void run() {
```

```
152             try {
153                window = new FpgaCtrlF();
154                window.FG_frame.setVisible(true);
155             } catch (Exception e) {
156                e.printStackTrace();
157             }
158             try {
159                     UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
160                 } catch (ClassNotFoundException ex) {
161                 } catch (InstantiationException ex) {
162                 } catch (IllegalAccessException ex) {
163                 } catch (UnsupportedLookAndFeelException ex) {
164                 }
165 
166             /*Balls balls = window.new Balls();
167                 //frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
168                 panel.setLayout(new BorderLayout());
169                 panel.add(balls);
170                 panel.setSize(400, 400);
171                 panel.setVisible(true);
172                 window.FG_frame.add(panel);*/
173          }
174       });
175    }
176 
177    /**
178     * Create the application.
179     */
180    public FpgaCtrlF() {
181       initialize();
182    }
183 
184    /**
185     * Initialize the contents of the frame.
186     */
187    private void initialize() {
188       FG_frame = new JFrame();
189       FG_frame.setBounds(100, 100, 1218, 780);
190       FG_frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
191 
192 
193       //Balls balls = window.new Balls();
194        /*JPanel panel = new JPanel();
195        panel.setLayout(new BorderLayout());
196          panel.add(new Balls());
197          panel.setSize(400, 400);
198          panel.setBounds(300, 500, 300, 500);
199          panel.setVisible(true);
200          FG_frame.getContentPane().add(panel);*/
201       FG_frame.getContentPane().setLayout(null);
202 
```

```
203
204     JPanel Cmd_panel = new JPanel();
205     Cmd_panel.setBounds(45, 625, 369, 35);
206     Cmd_panel.setBackground(new Color(109, 152, 184));
207     Cmd_panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));
208     FlowLayout flowLayout = (FlowLayout) Cmd_panel.getLayout();
209     flowLayout.setHgap(10);
210     flowLayout.setAlignment(FlowLayout.LEFT);
211     FG_frame.getContentPane().add(Cmd_panel);
212
213     JScrollPane scrollPane = new JScrollPane();
214     scrollPane.setBounds(45, 226, 240, 388);
215
216     JLabel lblNewLabel = new JLabel("cmd>> ");
217     lblNewLabel.setHorizontalAlignment(SwingConstants.LEFT);
218     Cmd_panel.add(lblNewLabel);
219
220     cmd_textField = new JTextField();
221     cmd_textField.setBackground(new Color(193, 211, 225));
222     cmd_textField.setFont(new Font("Monospaced", Font.BOLD | Font.ITALIC,
          12));
223     cmd_textField.addActionListener(new ActionListener() {
224        public void actionPerformed(ActionEvent arg0) {
225           CommandHandler(cmd_textField.getText());
226           cmd_textField.setText("");
227        }
228     });
229     Cmd_panel.add(cmd_textField);
230     cmd_textField.setColumns(40);
231     FG_frame.getContentPane().add(scrollPane);
232
233     Gr_panel = new JPanel();
234     Gr_panel.setLocation(691, 336);
235     Gr_panel.setBackground(new Color(245, 255, 250));
236     Gr_panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));
237     FG_frame.getContentPane().add(Gr_panel);
238
239     tout_textPane = new JTextPane();
240     tout_textPane.setFont(new Font("Monospaced", Font.BOLD | Font.ITALIC,
          13));
241     tout_textPane.setForeground(new Color(255, 255, 0));
242     scrollPane.setViewportView(tout_textPane);
243     tout_textPane.setBackground(new Color(201, 211, 150));
244     tout_textPane.setEditable(false);
245
246     JLabel lblNewLabel_6 = new JLabel("");
247     lblNewLabel_6.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
          GUImod\\images\\recentcomm.png"));
248     lblNewLabel_6.setBackground(new Color(109, 152, 184));
249     scrollPane.setColumnHeaderView(lblNewLabel_6);
250     lblNewLabel_6.setFont(new Font("Tahoma", Font.PLAIN, 14));
```

```
251
252     JList list = new JList();
253     list.setBounds(21, 123, 172, -132);
254     FG_frame.getContentPane().add(list);
255
256     JLabel lblNewLabel_1 = new JLabel("");
257     lblNewLabel_1.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
            GUImod\\images\\title.png"));
258     lblNewLabel_1.setBounds(89, 39, 1052, 43);
259     lblNewLabel_1.setFont(new Font("Tahoma", Font.PLAIN, 28));
260     FG_frame.getContentPane().add(lblNewLabel_1);
261
262     JLabel lblNewLabel_2_1 = new JLabel("");
263     lblNewLabel_2_1.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
            GUImod\\images\\refpos.png"));
264     lblNewLabel_2_1.setBounds(649, 253, 186, 20);
265     lblNewLabel_2_1.setFont(new Font("Tahoma", Font.PLAIN, 14));
266     FG_frame.getContentPane().add(lblNewLabel_2_1);
267
268     ReferenceTextView = new JTextField();
269     ReferenceTextView.setBackground(new Color(181, 205, 213));
270     ReferenceTextView.setBounds(691, 276, 96, 20);
271     FG_frame.getContentPane().add(ReferenceTextView);
272     ReferenceTextView.setColumns(10);
273
274     JLabel lblNewLabel_3 = new JLabel("Displacement (mm)");
275     lblNewLabel_3.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
            GUImod\\images\\displacement.png"));
276     lblNewLabel_3.setBounds(997, 253, 140, 20);
277     lblNewLabel_3.setFont(new Font("Tahoma", Font.PLAIN, 14));
278     FG_frame.getContentPane().add(lblNewLabel_3);
279
280     textField_1 = new JTextField();
281     textField_1.setBackground(new Color(181, 205, 213));
282     textField_1.setBounds(1017, 276, 96, 20);
283     FG_frame.getContentPane().add(textField_1);
284     textField_1.setColumns(10);
285
286     JLabel lblNewLabel_5 = new JLabel("");
287     lblNewLabel_5.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
            GUImod\\images\\resgraph.png"));
288     lblNewLabel_5.setBounds(728, 305, 419, 20);
289     FG_frame.getContentPane().add(lblNewLabel_5);
290
291     JLabel lblNewLabel_7 = new JLabel("");
292     lblNewLabel_7.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
            GUImod\\images\\displacementcontrol.png"));
293     lblNewLabel_7.setBounds(817, 176, 200, 20);
294     lblNewLabel_7.setFont(new Font("Tahoma", Font.PLAIN, 14));
295     FG_frame.getContentPane().add(lblNewLabel_7);
296
```

```
297         slider = new JSlider();
298         slider.setForeground(new Color(255, 255, 0));
299         slider.setBackground(new Color(64, 128, 128));
300         slider.setBounds(817, 197, 200, 45);
301         slider.setMinimum(113);
302         slider.setMaximum(165);
303         slider.setValue(0);
304         slider.setToolTipText("");
305         slider.setPaintTicks(true);
306         slider.setMinorTickSpacing(1);
307         slider.setMajorTickSpacing(9);
308         slider.setPaintLabels(true);
309         //slider.setValue(0);
310         lblNewLabel_8 = new JLabel();
311         //lblDistance.setBounds(650, 150, 200, 200);
312
313
314
315         //slider.addActionListener(this);
316         slider.addChangeListener(new ChangeListener() {
317               public void stateChanged(ChangeEvent event) {
318               val = (-320) + (slider.getValue()-113)*13.0769;
319                 //DistSelectLabel.setText("Distance Selected is "+
                        slider.getValue()+" mm");
320               ReferenceTextView.setText(" "+slider.getValue());
321
322               }
323             });
324
325         FG_frame.getContentPane().add(slider);
326
327         STARTButton = new JButton("");
328         STARTButton.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
            GUImod\\images\\start.png"));
329         STARTButton.setBackground(new Color(92, 147, 163));
330         STARTButton.setBounds(574, 123, 102, 42);
331         STARTButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
332         FG_frame.getContentPane().add(STARTButton);
333         STARTButton.addActionListener(this);
334
335         STOPButton = new JButton("");
336         STOPButton.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
            GUImod\\images\\stop.png"));
337         STOPButton.setBackground(new Color(92, 147, 163));
338         STOPButton.setBounds(1038, 123, 102, 42);
339         STOPButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
340         FG_frame.getContentPane().add(STOPButton);
341         STOPButton.addActionListener(this);
342
343         HELPButton = new JButton();
344         HELPButton.setBackground(new Color(109, 152, 184));
```

```
345       HELPButton.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
              GUImod\\images\\help.png"));
346       HELPButton.setBounds(45, 686, 133, 40);
347       HELPButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
348       FG_frame.getContentPane().add(HELPButton);
349       HELPButton.addActionListener(this);
350
351       CLEARCommand = new JButton("");
352       CLEARCommand.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
              GUImod\\images\\clear.png"));
353       CLEARCommand.setBackground(new Color(92, 147, 163));
354       CLEARCommand.setBounds(45, 123, 90, 42);
355       CLEARCommand.setFont(new Font("Tahoma", Font.PLAIN, 14));
356       FG_frame.getContentPane().add(CLEARCommand);
357       CLEARCommand.addActionListener(this);
358
359       comboBox = new JComboBox();
360       comboBox.setBackground(new Color(181, 205, 213));
361       comboBox.setBounds(155, 123, 172, 42);
362 //      ********************************* available serial ports
      ************************
363           SerialPort[] portNames = SerialPort.getCommPorts();
364           for(SerialPort portName : portNames)
365              comboBox.addItem(portName.getSystemPortName());
366       FG_frame.getContentPane().add(comboBox);
367
368       CONNECTButton = new JButton("");
369       CONNECTButton.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
              GUImod\\images\\connect.png"));
370       CONNECTButton.setBackground(new Color(92, 147, 163));
371       CONNECTButton.setBounds(349, 123, 172, 42);
372       FG_frame.getContentPane().add(CONNECTButton);
373
374       /*JPanel panel = new JPanel();
375       springLayout.putConstraint(SpringLayout.NORTH, panel, 50,
              SpringLayout.SOUTH, STOPButton);
376       springLayout.putConstraint(SpringLayout.WEST, panel, 0,
              SpringLayout.WEST, STOPButton);
377       springLayout.putConstraint(SpringLayout.SOUTH, panel, -56,
              SpringLayout.NORTH, Cmd_panel);
378       springLayout.putConstraint(SpringLayout.EAST, panel, -53,
              SpringLayout.EAST, lblNewLabel_7);*/
379
380
381       //****************************************** BALL MODIFICATION START
              ***************************
382       JPanel panel_1 = new JPanel();
383       panel_1.setBackground(Color.YELLOW);
384       /*springLayout.putConstraint(SpringLayout.NORTH, panel_1, 33,
              SpringLayout.SOUTH, STARTButton);
```

```
385         springLayout.putConstraint(SpringLayout.WEST, panel_1, 77,
                SpringLayout.EAST, scrollPane);
386         springLayout.putConstraint(SpringLayout.SOUTH, panel_1, -49,
                SpringLayout.NORTH, Cmd_panel);
387         springLayout.putConstraint(SpringLayout.EAST, panel_1, 287,
                SpringLayout.EAST, scrollPane);*/
388
389
390         panel_1.setLayout(new BorderLayout());
391         panel_1.add(new Balls());
392         //panel_1.setSize(400, 400);
393         panel_1.setBounds(452,285, 31, 239);
394         panel_1.setVisible(true);
395           FG_frame.getContentPane().add(panel_1);
396
397           lblNewLabel_10 = new JLabel("");
398           lblNewLabel_10.setBackground(new Color(128, 255, 255));
399           lblNewLabel_10.setBounds(433, 517, 150, 112);
400           lblNewLabel_10.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
                GUImod\\rsz_1rsz_1_removalai__tmp-63e964d1ca13f_c2z2ei-removebg-preview.png"));
401           FG_frame.getContentPane().add(lblNewLabel_10);
402
403           lblNewLabel_9 = new JLabel("");
404           lblNewLabel_9.setBounds(393, 161, 233, 123);
405           FG_frame.getContentPane().add(lblNewLabel_9);
406           lblNewLabel_9.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
                GUImod\\rsz_2img_20230206_143831-removebg-preview.png"));
407
408           btnNewButton = new JButton("");
409           btnNewButton.setBackground(new Color(92, 147, 163));
410           btnNewButton.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
                GUImod\\images\\plot.png"));
411           btnNewButton.setBounds(897, 123, 102, 42);
412           FG_frame.getContentPane().add(btnNewButton);
413
414           btnDo = new JButton("");
415           btnDo.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
                GUImod\\images\\download.png"));
416           btnDo.setBackground(new Color(92, 147, 163));
417           btnDo.setBounds(718, 123, 149, 42);
418           FG_frame.getContentPane().add(btnDo);
419
420           lblNewLabel_4 = new JLabel("New label");
421           lblNewLabel_4.setIcon(new
                ImageIcon("C:\\Users\\91994\\Downloads\\ezgif.com-resize (5).jpg"));
422           lblNewLabel_4.setBounds(0, 2, 1222, 753);
423           FG_frame.getContentPane().add(lblNewLabel_4);
424           //lblNewLabel_9.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
                GUImod\\rsz_2img_20230206_143831-removebg-preview.png"));
425
426
```

```
427
428
429
430     //****************************************** BALL MODIFICATION ENDS
            ******************************
431
432
433     CONNECTButton.addActionListener(this);
434
435
436     CreateChart();
437       DispUpdate_Timer = new Timer(250, new ActionListener() {
438           @Override
439           public void actionPerformed(ActionEvent e) {
440
441
442               UpdateDynamic();
443               DispUpdate_Timer.restart();
444
445           }
446       });
447       DispUpdate_Timer.start();
448       Downl_Cnt = 0;
449       Pb_NValues = 0;
450       Pb_Ready = false;
451
452
453   }
454
455
456   private void CommandHandler(String cmds) {
457     String command = cmds, dev_name, fname;
458     int  k, q, stat;
459     Writer out = null;
460
461     ////
462     Pb_Ready = true;
463     ////
464
465     if (command.equals("clc")) {
466        tout_textPane.setText("");
467     } else if (command.equals("clg")) {
468        ClearChart();
469     } else if (command.equalsIgnoreCase("help")) {
470        helpfunction();
471
472     } else if (command.equals("dev")) {
473        PrintTxtWin("dev...", 0, true);
474        for (k = 0; k < SerialNetw.getNDev(); k++) {
475           dev_name = SerialNetw.getDevName(k);
476           stat = SerialNetw.getDevStat(dev_name);
```

```
477             if (stat == 2) {
478                PrintTxtWin(" " + dev_name + " [connected]", 4, true);

479             } else {
480                PrintTxtWin(" " + dev_name + " [avail]", 4, true);

481             }
482          }
483       } else if (command.startsWith("conn ")) {
484          dev_name = command.substring(5).toUpperCase();
485          if (SerialNetw.getDevStat(dev_name) != 1) {
486             PrintTxtWin("*** device unavailable", 3, true);
487          } else {
488                (new SerialNetw()).spConnect(dev_name);
489             PrintTxtWin("OK", 1, true);
490          }
491       } else if (command.equals("disconn")) {
492          dev_name = SerialNetw.getConName();
493          if (dev_name != null) {
494             SerialNetw.spDisconn(dev_name);
495             PrintTxtWin("OK", 1, true);
496          } else {
497             PrintTxtWin("*** no device connected", 3, true);
498          }
499       } else if (command.equals("exit")) {
500          dev_name = SerialNetw.getConName();
501          if (dev_name != null) {
502             SerialNetw.spDisconn(dev_name);
503          }
504            DispUpdate_Timer.stop();
505            System.exit(0);
506       } else if (command.equals("downl")) {
507          if (SerialNetw.getConName() != null) {
508             Downl_Cnt = 0;
509             Pb_NValues = 0;
510               Pb_Ready = false;
511             SerialNetw.SendString("p\n");
512             PrintTxtWin(" => starting download:\n ", 1, false);
513          } else {
514             PrintTxtWin("*** no connected device", 3, true);
515          }
516       } else if (command.startsWith("plot")) {
517          if (!Pb_Ready) {
518             PrintTxtWin("*** no plot data available", 3, true);
519          } else if (Pb_NValues < 2) {
520             PrintTxtWin("*** no data points", 3, true);
521          } else {
522             if (command.equals("plot")) {
523                ClearChart();
524                for (k = 0; k < Pb_NValues; k++) {
525                   if (Pb_NValues < NROWS) {
```

```
526                     for (q = 0; q < NTRACES; q++) {
527                        mtraces[q].addPoint((double)k, Plot_Buffer[k][q]);
528                     }
529                  }
530               }
531
532         } else {
533               if (command.length() < 6) {
534                  PrintTxtWin("*** no file name given", 3, true);
535               } else {
536                  fname = command.substring(5);
537                  PrintTxtWin(" => attemp to write file \"" + fname + "\"", 1,
                       true);
538                  try {
539                        out = new OutputStreamWriter(new
                           FileOutputStream("\\Temp\\" + fname));
540                     for (k = 0; k < Pb_NValues; k++) {
541                           out.write(String.format(Locale.ENGLISH, "%f %f %f
                              %f\n",
542                              Plot_Buffer[k][0], Plot_Buffer[k][1],
543                              Plot_Buffer[k][2], Plot_Buffer[k][3]));
544                     }
545                     out.close();
546                  } catch (Exception ex) {
547                     PrintTxtWin(ex.toString(), 3, true);
548                  }
549                }
550
551            }
552         }
553      } else if (command.startsWith(".")) {
554         if (SerialNetw.getConName() != null) {
555            SerialNetw.SendString(command.substring(1) + "\n");
556         } else {
557            PrintTxtWin("*** no connected device", 3, true);
558         }
559      } else if (command.length() > 0) {
560         PrintTxtWin("*** command???: \"" + command + "\"", 3, true);
561      }
562   }
563
564
565    private void helpfunction() {
566      PrintTxtWin("Click the Button / Use Below listed command in cmd line for
          the Operation\n", 1, true);
567      PrintTxtWin("FPGA Control Help:", 1, true);
568      PrintTxtWin("  clc - clear text window", 1, true);
569      PrintTxtWin("  clg - clear chart window", 1, true);
570      PrintTxtWin("  dev - list avaliable devices", 1, true);
571      PrintTxtWin("  conn {comX} - connect", 1, true);
572      PrintTxtWin("  disconn - disconnect", 1, true);
```

```
573         PrintTxtWin("   .{sendstring}", 1, true);
574 
575     }
576 
577     private void PrintTxtWin(String twstr, int twstyle, boolean newline) {
578         try {
579             Document doc = tout_textPane.getStyledDocument();
580             StyleConstants.setItalic(TextSet, false);
581             StyleConstants.setBold(TextSet, false);
582             StyleConstants.setForeground(TextSet, Color.BLACK);
583             switch (twstyle) {
584                 case 0:
585                     StyleConstants.setBold(TextSet, true);
586                     StyleConstants.setForeground(TextSet, Color.DARK_GRAY);
587                     break;
588                 case 1: StyleConstants.setForeground(TextSet, Color.BLUE);
589                     break;
590                 case 2: StyleConstants.setForeground(TextSet, Color.BLACK);
591                     break;
592                 case 3: StyleConstants.setForeground(TextSet, Color.RED);
593                   break;
594                 case 4: StyleConstants.setForeground(TextSet, Color.GREEN);
595                   break;
596                 default:
597                     doc.remove(0, doc.getLength());
598             }
599             if (twstyle >= 0) {
600                tout_textPane.setCharacterAttributes(TextSet, true);
601                if (newline) {
602                     doc.insertString(doc.getLength(), twstr+"\n", TextSet);
603                } else {
604                     doc.insertString(doc.getLength(), twstr, TextSet);
605                }
606             }
607         } catch (BadLocationException ex) {
608             System.out.println(ex.toString());
609         }
610      }
611 
612 
613     private void ClearChart()
614     {
615        int  k;
616        for (k = 0; k < NTRACES; k++) {
617           mtraces[k].removeAllPoints();
618           mtraces[k].addPoint(0.0, 0.0);
619        }
620     }
621 
622 
```

```
623
624    private void CreateChart()
625    {
626       int  k;
627
628       chart = new Chart2D();
629       chart.setBackground(new Color(201, 211, 150));
630       // FlowLayout flowLayout = (FlowLayout) chart.getLayout();
631       for (k = 0; k < NTRACES; k++) {
632          mtraces[k] = new Trace2DSimple();
633            chart.addTrace(mtraces[k]);
634       }
635         mtraces[0].setColor(Color.blue); mtraces[0].setName("trace 0");
636         mtraces[1].setColor(Color.red);  mtraces[1].setName("trace 1");
637         mtraces[2].setColor(Color.green); mtraces[2].setName("trace 2");
638         mtraces[3].setColor(Color.magenta); mtraces[3].setName("trace 3");
639         Gr_panel.setLayout(new BorderLayout(0, 0));
640         mtraces[0].setPhysicalUnits("Time (S)","Magnitude");
641
642         Gr_panel.add(chart);
643
644         ClearChart = new JButton("");
645         ClearChart.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
              GUImod\\images\\clear.png"));
646         ClearChart.setBackground(new Color(201, 211, 150));
647         chart.add(ClearChart);
648         ClearChart.addActionListener(this);
649         ClearChart.setFont(new Font("Tahoma", Font.PLAIN, 14));
650         Gr_panel.setSize(481,293);
651         chart.setVisible(true);
652         Gr_panel.setVisible(true);
653         Gr_panel.repaint();
654
655    }
656
657
658     private void UpdateDynamic() {
659       String recv_s;
660       String splits[];
661       int  k;
662       ////////////////////////////////
663       //recv_s = "10";
664       Pb_NValues = 10;
665        ////////////////////////////////
666       while ((recv_s = SerialNetw.ReadString()) != null) {
667          // System.out.println("["+ recv_s + "]");
668          if (recv_s.startsWith("$") || recv_s.startsWith("~")) {
669             PrintTxtWin("+", 1, false);
670             Downl_Cnt++;
671             if (Downl_Cnt >= 25) {
672                Downl_Cnt = 0;
```

```
673                PrintTxtWin("\n  ", 1, false);
674             }
675             if (Pb_NValues < NROWS) {
676                splits = recv_s.split(" ");
677                try {
678                   for (k = 1; k < splits.length; k++) {
679                      Plot_Buffer[Pb_NValues][k-1] =
                              (double)Integer.parseInt(splits[k]);
680                   }
681                   Pb_NValues++;
682                } catch (NumberFormatException e) {
683                    System.out.println(e.toString());
684                 }
685             }
686             if (recv_s.startsWith("~")) {
687                SerialNetw.SendString("p\n");
688             }
689          } else if (recv_s.startsWith("###")) {
690             PrintTxtWin("\nDownload finished.", 1, true);
691             Pb_Ready = true;
692          } else {
693             PrintTxtWin(recv_s, 0, true);
694          }
695       }
696
697    }
698
699
700    static int Downl_Cnt;
701    static boolean Pb_Ready;
702    static final int NROWS = 2048, NCOLS = 4;
703    static double Plot_Buffer[][] = new double[NROWS][NCOLS];
704    static int Pb_NValues;
705    static Chart2D chart;
706    static final int NTRACES = 4;
707    static ITrace2D mtraces[] = new ITrace2D[4];
708    // static ITrace2D trace_1 = new Trace2DLtd(20);
709    private SimpleAttributeSet TextSet = new SimpleAttributeSet();
710    private Timer DispUpdate_Timer;
711    private JPanel Gr_panel;
712    private JTextPane tout_textPane;
713    private JTextField ReferenceTextView;
714    private JTextField textField_1;
715    private JButton CONNECTButton;
716    private JLabel lblNewLabel_9;
717    private JLabel lblNewLabel_10;
718    private JButton btnNewButton;
719    private JButton btnDo;
720    private JLabel lblNewLabel_4;
721
722
```

```
723
724     @Override
725     public void actionPerformed(ActionEvent e) {
726        if(e.getSource()==STARTButton) {
727           CommandHandler(".r "+val);
728           CommandHandler(".s");
729           /*try {
730              broadcastslider();
731           } catch (IOException e1) {
732              // TODO Auto-generated catch block
733              e1.printStackTrace();
734           } catch (InterruptedException e1) {
735              // TODO Auto-generated catch block
736              e1.printStackTrace();
737           }*/
738        }
739        if(e.getSource()==STOPButton) {
740           CommandHandler(".p");
741           System.out.println("STOPPPPPPPPPPPP");
742        }
743        if(e.getSource()==HELPButton) {
744           helpfunction();
745           System.out.println("HELPP");
746        }
747        if(e.getSource()==CLEARCommand) {
748           System.out.println("CLEARRRR");
749           tout_textPane.setText("");
750        }
751        if(e.getSource()==ClearChart)
752           ClearChart();
753        if(e.getSource()==CONNECTButton) {
754           if(CONNECTButton.getText().equals("Connect")) {
755              //connect to serial port
756              String s="conn "+comboBox.getSelectedItem().toString();
757              CommandHandler(s);
758
759              CONNECTButton.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp
                    - GUImod\\images\\disconn.png"));
760              //CONNECTButton.setText("Disconnect");
761              comboBox.setEnabled(false);
762
763           }
764
765        else {
766           CommandHandler("disconn");
767
768           CONNECTButton.setIcon(new ImageIcon("C:\\Eclipsecustomproj\\eclwsp -
                GUImod\\images\\connect.png"));
769           //CONNECTButton.setText("Connect");
770        }
771        }
```

```
772
773     }
774     //double val=slider.getValue();
775      //Double value= ((val-113)/1000)-0.026;
776
777 /**private void broadcastslider() throws IOException, InterruptedException {
778     SerialPort sp = SerialPort.getCommPort("COM3"); // device name TODO: must
            be changed
779      sp.setComPortParameters(57600, 8, 1,0); // default connection settings for
            Arduino
780      sp.setComPortTimeouts(SerialPort.TIMEOUT_WRITE_BLOCKING, 0, 0); // block
            until bytes can be written
781
782      //SerialPort sp1 = SerialPort.getCommPort("COM4");
783      //sp1.openPort();
784      //sp1.setComPortTimeouts(SerialPort.TIMEOUT_READ_SEMI_BLOCKING, 0, 0);
785      //sp1.setComPortParameters(4800, 8, 1,0);
786
787      if (sp.openPort()) {
788        System.out.println("Port is open :)");
789      } else {
790        System.out.println("Failed to open port :(");
791        return;
792      }
793
794      /*if (sp1.openPort()) {
795          System.out.println("Port is open :)");
796        } else {
797          System.out.println("Failed to open port :(");
798          return;
799        }
800      int val=slider.getValue();
801      String s = "0"+ Integer.toString(val);
802      System.out.println("Slider value : "+ s);
803      for (int i = 0; i <4; ++i) {
804        char a=s.charAt(i);//extract char at location i
805        Integer b=Character.getNumericValue(a)+48;// add 0 ascii convert to ascii
806        sp.getOutputStream().write(b.byteValue());
807          sp.getOutputStream().flush();
808          System.out.println("Sent number: " + b);
809
810
811
812
813          Thread.sleep(2000);
814
815
816        /*int len = 0;
817        InputStream in= sp1.getInputStream();
818        int data;
819        byte[] buffer = new byte[8];
```

```
      while (in.available() > 0)
      {
          data = in.read();
          //System.out.println("data :"+data);
          if ( data == -1 || data == '\n' ) {
              break;
          }
          buffer[len++] = (byte) data;
      }
      System.out.println("Recieved from Arduino : "+new String(buffer,0,len));

    }

    if (sp.closePort()) {
      System.out.println("Port is closed :)");
    } else {
      System.out.println("Failed to close port :(");
      return;
    }
    /*if (sp1.closePort()) {
        System.out.println("Port is closed :)");
      } else {
        System.out.println("Failed to close port :(");
        return;
      }


}*/


//************************************************ BALL MODIFICATION START
    ******************************************
   public class Balls extends JPanel {

        public Balls() {
            setLayout(null);
            // Randomize the speed and direction...
            add(new Ball("red", 0,0));
        }
    }
   public class Ball extends JPanel implements Runnable {

        Color color;
        int diameter;
        long delay;
        private int vx;
        private int vy;

        public Ball(String ballcolor, int xvelocity, int yvelocity) {
            if (ballcolor == "red") {
                color = Color.red;
```

```
870             }
871
872
873             new Thread(this).start();
874
875         }
876
877         protected void paintComponent(Graphics g) {
878             super.paintComponent(g);
879             Graphics2D g2 = (Graphics2D) g;
880
881
882             g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                    RenderingHints.VALUE_ANTIALIAS_ON);
883             g.setColor(color);
884             g.fillOval(0, 0, 30, 30); //adds color to circle
885             g.setColor(Color.black);
886             g2.drawOval(0, 0, 30, 30); //draws circle
887         }
888
889         @Override
890         public Dimension getPreferredSize() {
891             return new Dimension(30, 30);
892         }
893
894         public void run() {
895
896
897             while (isVisible()) {
898
899                 try {
900                     SwingUtilities.invokeAndWait(new Runnable() {
901                         @Override
902                         public void run() {
903
904                             move();
905                             repaint();
906                         }
907                     });
908                 } catch (InterruptedException exp) {
909                     exp.printStackTrace();
910                 } catch (InvocationTargetException exp) {
911                     exp.printStackTrace();
912                 }
913             }
914         }
915
916         public void move() {
917
918             // Update the size and location...
919             setSize(getPreferredSize());
```

```
920             int val2 = (int) (210-((slider.getValue()-113)*4.038));
921             setLocation(0, val2);
922 
923         }
924     }
925 }
926 
927 }
```

C Matlab App Designer Code

```
classdef relese_3_7 < matlab.apps.AppBase
    %RELESE_3_7 MAGNETIC LEVITATION GUI design- APP DESIGNER
    %   % GUI Summary:
    %   % -------------
    %   % Filename: Release 3.4
    %   % Author: Ruturaj Bendkhale
    %   % Date: 01-06-2023
    %   % Description:
    %
    %   % Code Summary:
    %   % --------------
    %   % This MATLAB code implements a graphical user interface (GUI) for
    %   universitsy project of Magnetic Levitation.
    %   % The GUI provides interactive controls and visualizations to
    %   % [1] Realtime Reading of the data from running MATLAB Simulation
    %   % [2] Realtime setting functions (waveforms) as a input to the
    %   system
    %   % [3] Data Logging
    %   % [4] Selecting Target and Target Code
    %
    %   % Key Components:
    %   % ----------------
    %   % - [List the main UI components and their functionalities]
    %   % - [Component 1]
    %   % - [Component 2]
    %   % - [Component 3]
    %   %    ...
    %
    %   % Key Functions:
    %   % ---------------
    %   % - [List the main functions and their purposes]
    %   % - [Function 1]
    %   % - [Function 2]
    %   % - [Function 3]
    %   %    ...
    %
    %   % Optimization Considerations:
    %   % ----------------------------
    %   % During the development of this GUI, the following optimizations
    %   were considered:
    %   % - Improving code readability and maintainability through comments
    %   and meaningful variable/function names.
    %   % - Minimizing redundancy and code duplication.
    %   % - Optimizing UI component creation and organization.
    %   % - Streamlining callback functions and avoiding unnecessary
    %   computations.
    %   % - Utilizing MATLAB optimization techniques, such as vectorization
    %   and caching.
    %   % - Implementing error handling and input validation mechanisms.
```

```
50      %   % - Profiling and benchmarking the code for performance
51      %   improvements.
52      %
53      %   % Usage Instructions:
54      %   % -------------------
55      %   % [Provide any necessary instructions for running or interacting
56      %   with the GUI]
57      %   % - [Step 1: Explain the initial setup or requirements]
58      %   % - [Step 2: Describe how to launch the GUI]
59      %   % - [Step 3: Provide a brief overview of the GUI's usage and
60      %   functionalities]
61      %   % - [Step 4: Add any other important instructions or notes]
62      %
63      %   % Notes:
64      %   % ------
65      %   % - [Add any additional notes or considerations about the GUI code]
66      %
67      %   % Acknowledgements:
68      %   % -----------------
69      %   % [Optional: If applicable, acknowledge any external libraries,
70      %   code snippets, or contributions]
71      %
72      %   % Contact Information:
73      %   % --------------------
74      %   % [Provide your contact information if others need to reach you for
75      %   questions or collaboration]
76      %   % - Email: [Enter your email address]
77      %   % - Website: [Enter your website or research project page]
78      %
79      %   % References:
80      %   % ------------
81      %   % [List any references or resources that were helpful in developing
82      %   the GUI]
83      %
84      %   % ---------------------------- [End of Code Header]
85      %   ----------------------------
86
87      % Properties that correspond to app components
88      properties (Access = public)
89          UIFigure                    matlab.ui.Figure
90          StopTimeEditField           slrealtime.ui.control.StopTimeEditField
91          AnimationPanel              matlab.ui.container.Panel
92          AnimationAxes               matlab.ui.control.UIAxes
93          StartStopButton             slrealtime.ui.control.StartStopButton
94          ResultsingraphicalformatPanel matlab.ui.container.Panel
95          UIAxes                      matlab.ui.control.UIAxes
96          CommandPromptPanel          matlab.ui.container.Panel
97          DataAquisiationsPanel       matlab.ui.container.Panel
98          ClearLogButton              matlab.ui.control.Button
99          PlotLogButton               matlab.ui.control.Button
100         LoggerONOFFButton           matlab.ui.control.StateButton
```

```
        MeasrementsPanel             matlab.ui.container.Panel
        vel_label_3                  matlab.ui.control.Label
        ref_label                    matlab.ui.control.Label
        dist_label                   matlab.ui.control.Label
        ModeSelected                 matlab.ui.control.Label
        VelocitymmsEditFieldLabel    matlab.ui.control.Label
        DisplacementmmEditFieldLabel matlab.ui.control.Label
        ReferencePositionmmEditFieldLabel matlab.ui.control.Label
        DistanceControlPanel         matlab.ui.container.Panel
        mmLabel_3                    matlab.ui.control.Label
        positionEditField            matlab.ui.control.EditField
        AmplitudemmSlider            matlab.ui.control.Slider
        AmplitudemmSliderLabel       matlab.ui.control.Label
        PeriodSecSlider              matlab.ui.control.Slider
        PeriodSecSliderLabel         matlab.ui.control.Label
        ButtonGroup                  matlab.ui.container.ButtonGroup
        mmLabel_2                    matlab.ui.control.Label
        PatternSelectorKnob          matlab.ui.control.DiscreteKnob
        PatternSelectorKnobLabel     matlab.ui.control.Label
        mmLabel                      matlab.ui.control.Label
        PositionSlider               matlab.ui.control.Slider
        CommunicationsPearametersPanel_2 matlab.ui.container.Panel
        BallLamp                     matlab.ui.control.Lamp
        BallLampLabel                matlab.ui.control.Label
        CurrentLamp                  matlab.ui.control.Lamp
        CurrentLampLabel             matlab.ui.control.Label
        VIOLamp                      matlab.ui.control.Lamp
        VIOLampLabel                 matlab.ui.control.Label
        ConnectButton                slrealtime.ui.control.ConnectButton
        TargetSelector               slrealtime.ui.control.TargetSelector
        LoadButton                   slrealtime.ui.control.LoadButton
        CLOSEButton                  matlab.ui.control.StateButton
        MAGNETICLEVITATIONCONTROLLabel matlab.ui.control.Label
    end

    properties (Access = private)
        tg         %target
        mdl = 'untitled'; % Loaded model
        ParamTuner_PatternSelctor
        InstrumentManager
        Instrument
        % Parameter Objects
        ParamTuner_Amplitude
        ParamTuner_Period
        ParamTuner_Reference
        % Signals
        VioEnable

        % Datalogging Objects
        logmat
        log_time
```

```
        log_displacement
        log_reference_position
        log_delta_current
        datetm % current datetime object
        dir

        % Animation Objects
        aniRefMark
        aniBallPos
        ballXComp = 0.005*sin(0:pi/10:2*pi); %rsin(theta)
        ballYComp = 0.005*cos(0:pi/10:2*pi); %rcos(theta)
        %r Radius of the ball
        % (theta) are the segement of full circle (in rad)
        % Constants
        ballAbsentThreshold = 0.055 % *Implimentation* Change as real value
        defPeriod = 2; % 2 (sec)
        defAmplitude = 0; % 0(mm)
    end


    methods (Access = private)

        function setupAnimateAxes(app)
            %
            % Set up plot animation for inverted pendulum and cart using
            % surface and patch commands

            app.AnimationAxes.XLim = [-0.03 0.03];
            app.AnimationAxes.YLim = [-0.03 0.03];

            % drawing Circular shape that resembles ball
            app.aniBallPos = patch( ...
                'Parent',  app.AnimationAxes, ...
                'XData',   app.ballXComp, ...
                'YData',   app.ballYComp, ...
                'FaceColor','red');

            % drawing a Line shape
            app.aniRefMark = patch( ...
                'Parent',  app.AnimationAxes, ...
                'YData',   [0 0], ...
                'XData',   [-0.01 +0.01], ...
                'EdgeColor','green');
        end
        function offStateSetting(app)
                app.ParamTuner_Amplitude.Component = [];
                app.ParamTuner_Amplitude.BlockPath = '';
                app.ParamTuner_Amplitude.ParameterName = '';
                app.AmplitudemmSlider.Enable = 'off';
                app.AmplitudemmSlider.Value = app.defAmplitude;
                app.PositionSlider.Value = app.defAmplitude;
```

```
203
204             app.ParamTuner_Period.Component = [];
205             app.ParamTuner_Period.BlockPath = '';
206             app.ParamTuner_Period.ParameterName = '';
207             app.PeriodSecSlider.Enable = 'off';
208             app.positionEditField.Enable = 'off';
209             app.PositionSlider.Enable = 'off';
210             app.PeriodSecSlider.Value = app.defPeriod;
211             app.ModeSelected.Text = 'Mode Selector : Off';
212             app.ModeSelected.FontColor = 'red';
213             app.VIOLamp.Color ='r';
214             app.VIOLampLabel.Text = 'VIO Disabled';
215             app.tg.setparam([app.mdl '/VIO_Enable'], 'Value', '0');
216
217             % Set Labels and Animation Objects to its default Position
218             app.dist_label.Text = num2str(0);
219             set(app.aniBallPos, ...
220                 'YData', app.ballYComp);
221
222             app.ref_label.Text = num2str(app.defAmplitude);
223             set(app.aniRefMark, ...
224                 'YData',[0 0]);
225
226         end
227         function onStateSetting (app)
228             app.ParamTuner_Amplitude.Component = app.AmplitudemmSlider;
229             app.AmplitudemmSlider.Enable = 'on';
230             app.ParamTuner_Period.Component = app.PeriodSecSlider;
231             app.PeriodSecSlider.Enable = 'on';
232         end
233
234         function updateAnimationCallback(app, instObj, eventData)
235
236             [time, ref] = instObj.getCallbackDataForSignal(eventData,
                    'ref_position(meter)');
237             [~, displacement] = instObj.getCallbackDataForSignal(eventData,
                    'lin_x_log');
238
239             if ~isempty(time)
240                 ref   = ref(end);
241                 displacement = displacement(end);
242
243                 if displacement > app.ballAbsentThreshold
244                     app.BallLampLabel.Text = 'Ball Absent';
245                     app.BallLamp.Color = 'r';
246                 else
247                     app.BallLampLabel.Text = 'Ball';
248                     app.BallLamp.Color = 'g';
249                 end
250
251
```

```
            try
                app.dist_label.Text = num2str((1000)*displacement); % meter
                    to mm Conversion
                set(app.aniBallPos, ...
                    'YData',displacement+app.ballYComp);

                app.ref_label.Text = num2str((1000)*ref); % meter to mm
                    Conversion
                set(app.aniRefMark, ...
                    'YData',[ref ref] );

            catch
                % Ignore any errors writing to uicontrols because app
                % may have been closed and uicontrols destroyed
            end
        end
    end


    function compValue = refPatternConvertToComponent(app, targetValue)

        switch targetValue
            case 1
                compValue = 'Triangular';
            case 2
                compValue = 'Sine';
            case 3
                compValue = 'Constant';
            case 4
                compValue = 'Trapazoidal';
            case 5
                compValue = 'Off';
        end

        app.refPatternAdjust(compValue);
    end


    function targetValue = refPatternConvertToTarget(app, compValue)
        switch compValue
            case 'Triangular'
                targetValue = 1;
            case 'Sine'
                targetValue = 2;
            case 'Constant'
                targetValue = 3;
            case 'Trapazoidal'
                targetValue = 4;
            case 'Off'
                targetValue = 5;
        end
```

```
301
302             app.refPatternAdjust(compValue);
303         end
304
305
306         function refPatternAdjust(app, value)
307
308             app.tg = slrealtime(app.TargetSelector.TargetName);
309
310             if strcmp(value, 'Off')
311                 offStateSetting(app);
312
313             else
314                 onStateSetting(app)
315                 %VIO
316                 app.VIOLamp.Color ='g';
317                 app.VIOLampLabel.Text = 'VIO Enabled';
318                 app.tg.setparam([app.mdl '/VIO_Enable'], 'Value', '1');
319
320                 switch value
321                     case 'Sine'
322                         %Enables
323                         app.PositionSlider.Enable = 'off';
324                         app.PeriodSecSlider.Enable = 'on';
325                         app.AmplitudemmSlider.Enable = 'on';
326                         %Text
327                         app.ModeSelected.FontColor = 'black';
328                         app.ModeSelected.Text = 'Mode : Sine Wave';
329                         %Variable Assignment
330                         app.ParamTuner_Amplitude.BlockPath = [app.mdl
                                '/sine_wave_generator'];
331                         app.ParamTuner_Amplitude.ParameterName =
                                'sin_wave_amplitude';
332                         app.ParamTuner_Period.BlockPath = [app.mdl
                                '/sine_wave_generator'];
333                         app.ParamTuner_Period.ParameterName = 'sin_wave_period';
334
335                     case 'Trapazoidal'
336                         %Enables
337                         app.PositionSlider.Enable = 'off';
338                         app.PeriodSecSlider.Enable = 'off';
339                         app.AmplitudemmSlider.Enable = 'on';
340                         app.PeriodSecSlider.Value = app.defPeriod;
341                         %Text
342                         app.ModeSelected.FontColor = 'black';
343                         app.ModeSelected.Text = 'Mode : Trapazoidal Wave';
344                         %Variable Assignment
345                         app.ParamTuner_Amplitude.BlockPath = [app.mdl
                                '/trapezoidal_wave_generator'];
346                         app.ParamTuner_Amplitude.ParameterName =
                                'trap_wave_amplitude';
```

```
347
348                 case 'Triangular'
349                     %Enables
350                     app.PositionSlider.Enable = 'off';
351                     app.PeriodSecSlider.Enable = 'on';
352                     app.AmplitudemmSlider.Enable = 'on';
353                     %Text
354                     app.ModeSelected.FontColor = 'black';
355                     app.ModeSelected.Text = 'Mode : Triangular Wave';
356                     %Variable Assignment
357                     app.ParamTuner_Amplitude.BlockPath = [app.mdl
                            '/traingular_wave_generator'];
358                     app.ParamTuner_Amplitude.ParameterName =
                            'tri_wave_amplitude';
359                     app.ParamTuner_Period.BlockPath = [app.mdl
                            '/traingular_wave_generator'];
360                     app.ParamTuner_Period.ParameterName = 'tri_wave_period';
361
362                 case 'Constant'
363                     %Enables
364                     app.PositionSlider.Enable = 'on';
365                     app.PeriodSecSlider.Enable = 'off';
366                     app.AmplitudemmSlider.Enable = 'off';
367                     app.PeriodSecSlider.Value = app.defPeriod;
368                     app.AmplitudemmSlider.Value = app.defAmplitude;
369                     %Text
370                     app.ModeSelected.FontColor = 'black';
371                     app.ModeSelected.Text = 'Mode : Constant Displacement';
372                     %Variable Assignment
373                     app.ParamTuner_Amplitude.Component = app.PositionSlider;
374                     app.ParamTuner_Amplitude.BlockPath = [app.mdl
                            '/constant_distance'];
375                     app.ParamTuner_Amplitude.ParameterName =
                            'constant_displacement';
376             end
377         end
378     end
379 end
380
381
382 % Callbacks that handle component events
383 methods (Access = private)
384
385     % Code that executes after component creation
386     function startupFcn(app)
387         screenSize = get(groot,'ScreenSize');
388         screenWidth = screenSize(3); % extract the Screen Resolution
389         screenHeight = screenSize(4); % in pixels
390         left = screenWidth*0.1; % figure Margin
391         bottom = screenHeight*0.1;
392         width = screenWidth*0.75;
```

```
393             height = screenHeight*0.75;
394             app.UIFigure.Position = [[left,bottom], width, height];
395             drawnow;
396
397             app.tg = slrealtime(app.TargetSelector.TargetName); % Set target
398             % Disable all controls when the target is not connected to the PC
399 %             if isempty(app.tg.TargetStatus)
400 %                 app.PositionSlider.Enable = 'off';
401 %                 app.PeriodSecSlider.Enable = 'off';
402 %                 app.AmplitudemmSlider.Enable = 'off';
403 %             end
404
405             app.LoadButton.TargetSource = app.TargetSelector;
406             app.StopTimeEditField.TargetSource = app.TargetSelector;
407             app.StartStopButton.TargetSource = app.TargetSelector;
408             app.ConnectButton.TargetSource = app.TargetSelector;
409             app.LoadButton.Application = app.mdl;
410
411
412             app.ParamTuner_Amplitude =
                    slrealtime.ui.tool.ParameterTuner(app.UIFigure, 'TargetSource',
                    app.TargetSelector);
413             app.ParamTuner_Period =
                    slrealtime.ui.tool.ParameterTuner(app.UIFigure, 'TargetSource',
                    app.TargetSelector);
414             app.ParamTuner_PatternSelctor =
                    slrealtime.ui.tool.ParameterTuner(app.UIFigure, 'TargetSource',
                    app.TargetSelector);
415
416             app.ParamTuner_PatternSelctor.Component = app.PatternSelectorKnob;
417             app.ParamTuner_PatternSelctor.BlockPath = [app.mdl '/selector'];
418             app.ParamTuner_PatternSelctor.ParameterName = 'Value';
419             app.ParamTuner_PatternSelctor.ConvertToComponent =
                    @(value)refPatternConvertToComponent(app, value);
420             app.ParamTuner_PatternSelctor.ConvertToTarget =
                    @(value)refPatternConvertToTarget(app, value);
421
422
423             app.Instrument = slrealtime.Instrument; % Create Instrument
424             app.UIAxes.YLim = [-0.03,+0.03]; % sets Fixed Yaxis Limit
425             app.Instrument.connectLine (app.UIAxes,'ref_position(meter)','g');
426             app.Instrument.connectLine(app.UIAxes,'lin_x_log','r');
427             app.Instrument.AxesTimeSpan = 20; % fixed X axis
428             grid(app.UIAxes,"on");
429             app.Instrument.AxesTimeSpanOverrun = 'wrap'; % 20 sec repets
430             legend(app.UIAxes,'Reference Position (mm)','Displacement (mm)');
431
432             app.Instrument.connectCallback(@app.updateAnimationCallback);
433             % updates the value into App developer's - Instrument manager
434             app.InstrumentManager =
                    slrealtime.ui.tool.InstrumentManager(app.UIFigure,
```

```
                'TargetSource', app.TargetSelector);
            app.InstrumentManager.Instruments = app.Instrument;

            app.setupAnimateAxes();

        end

        % Callback function: CLOSEButton, UIFigure
        function UIFigureCloseRequest(app, event)
            YN = uiconfirm(app.UIFigure, 'Initiate Exit?' , 'Close
                request','Icon','warning');
            app.tg = slrealtime(app.TargetSelector.TargetName);
            if strcmpi(YN, 'OK' )
%               if isempty (app.tg.TargetSettings.address)
%                   uialert(app.UIFigure,'Target Disconnected/ Not
    Found','Closing Application by BruteForce');
%                   pause(2); % pause to show window
%                   delete(app); % Delete app prematurely
%               else
                  % sets all parameters to default
                  app.tg.setparam([app.mdl '/selector'], 'Value', '5');
                  app.PatternSelectorKnob.Value = 'Off';
                  app.tg.setparam([app.mdl '/sine_wave_generator'],
                      'sin_wave_amplitude', num2str(app.defAmplitude));
                  app.tg.setparam([app.mdl '/trapezoidal_wave_generator'],
                      'trap_wave_amplitude', num2str(app.defAmplitude));
                  app.tg.setparam([app.mdl '/traingular_wave_generator'],
                      'tri_wave_amplitude', num2str(app.defAmplitude));
                  app.tg.setparam([app.mdl '/constant_distance'],
                      'constant_displacement', num2str(app.defAmplitude));

                  app.tg.setparam([app.mdl '/sine_wave_generator'],
                      'sin_wave_period', num2str(app.defPeriod));
                  app.tg.setparam([app.mdl '/traingular_wave_generator'],
                      'tri_wave_period', num2str(app.defPeriod));
%               end

            end
            delete(app);


        end

        % Value changed function: LoggerONOFFButton
        function LoggerONOFFButtonValueChanged(app, event)
            app.datetm = datestr(now, 'dd_mmmm_yy_HH.MM'); % gets the current
                Datetime
            app.dir = pwd; % finds the current directory path
            app.log_time = evalin('base','out.log.Time'); % reads
                SimulationOutput from base workspace
            app.log_reference_position = evalin('base','out.log.Data(:,1)');
```

```
475             app.log_displacement = evalin('base','out.log.Data(:,2)');
476             app.log_delta_current = evalin('base','out.log.Data(:,3)');
477 
478             app.logmat =
                    table(app.log_time,app.log_reference_position,app.log_displacement,app.log_de
479             app.logmat.Properties.VariableNames = ["Time","Reference
                    Position", "Displacement", "Controller Current"];
480             %creates Excel File
481             writetable(app.logmat,[app.datetm
                    'loggeddata.xls'],'Sheet',1,'AutoFitWidth',true);
482             datalogmassage = ['Datalogging File Created Sucessfully in '
                    app.dir ' Filename ' app.datetm '.xlsx'];
483             uialert(app.UIFigure,datalogmassage,'Datalogging
                    Finished','Icon','success');
484 
485         end
486 
487         % Button pushed function: PlotLogButton
488         function PlotLogButtonPushed(app, event)
489 
490             plot(app.UIAxes,(app.log_time)',(app.log_displacement)','g',(app.log_time)',(app.
491             app.UIAxes.YLimMode = "auto";
492             app.UIAxes.XLimMode = "auto";
493 
494         end
495 
496         % Button pushed function: ClearLogButton
497         function ClearLogButtonPushed(app, event)
498             YN = uiconfirm(app.UIFigure, 'Clearing the Plotted Data' ,
                    'Clear','Icon','warning');
499             if strcmpi(YN, 'OK' )
500                 cla(app.UIAxes);
501             end
502         end
503 
504         % Value changed function: PositionSlider
505         function PositionSliderValueChanging(app, event)
506             app.positionEditField.Value = num2str(app.PositionSlider.Value);
507         end
508 
509         % Value changed function: positionEditField
510         function positionEditFieldValueChanged(app, event)
511             app.PositionSlider.Value = str2double(app.positionEditField.Value)
                    ;
512         end
513     end
514 
515     % Component initialization
516     methods (Access = private)
517 
518         % Create UIFigure and components
```

```
519         function createComponents(app)
520
521             % Create UIFigure and hide until all components are created
522             app.UIFigure = uifigure('Visible', 'off');
523             app.UIFigure.AutoResizeChildren = 'off';
524             app.UIFigure.Position = [200 200 1095 641];
525             app.UIFigure.Name = 'MATLAB App';
526             app.UIFigure.CloseRequestFcn = createCallbackFcn(app,
                    @UIFigureCloseRequest, true);
527             app.UIFigure.WindowStyle = 'modal';
528
529             % Create MAGNETICLEVITATIONCONTROLLabel
530             app.MAGNETICLEVITATIONCONTROLLabel = uilabel(app.UIFigure);
531             app.MAGNETICLEVITATIONCONTROLLabel.FontSize = 36;
532             app.MAGNETICLEVITATIONCONTROLLabel.Position = [1 595 594 47];
533             app.MAGNETICLEVITATIONCONTROLLabel.Text = 'MAGNETIC LEVITATION
                    CONTROL';
534
535             % Create CLOSEButton
536             app.CLOSEButton = uibutton(app.UIFigure, 'state');
537             app.CLOSEButton.ValueChangedFcn = createCallbackFcn(app,
                    @UIFigureCloseRequest, true);
538             app.CLOSEButton.Text = 'CLOSE';
539             app.CLOSEButton.FontWeight = 'bold';
540             app.CLOSEButton.Position = [987 604 99 30];
541
542             % Create CommunicationsPearametersPanel_2
543             app.CommunicationsPearametersPanel_2 = uipanel(app.UIFigure);
544             app.CommunicationsPearametersPanel_2.AutoResizeChildren = 'off';
545             app.CommunicationsPearametersPanel_2.TitlePosition = 'centertop';
546             app.CommunicationsPearametersPanel_2.Title = 'Communications
                    Pearameters';
547             app.CommunicationsPearametersPanel_2.Position = [1 400 280 192];
548
549             % Create LoadButton
550             app.LoadButton =
                    slrealtime.ui.control.LoadButton(app.CommunicationsPearametersPanel_2);
551             app.LoadButton.Position = [23 7 110 65];
552
553             % Create TargetSelector
554             app.TargetSelector =
                    slrealtime.ui.control.TargetSelector(app.CommunicationsPearametersPanel_2);
555             app.TargetSelector.Position = [23 130 110 34];
556
557             % Create ConnectButton
558             app.ConnectButton =
                    slrealtime.ui.control.ConnectButton(app.CommunicationsPearametersPanel_2);
559             app.ConnectButton.Position = [23 83 110 35];
560
561             % Create VIOLampLabel
562             app.VIOLampLabel = uilabel(app.CommunicationsPearametersPanel_2);
```

```
            app.VIOLampLabel.HorizontalAlignment = 'right';
            app.VIOLampLabel.Position = [137 127 71 22];
            app.VIOLampLabel.Text = 'VIO';

            % Create VIOLamp
            app.VIOLamp = uilamp(app.CommunicationsPearametersPanel_2);
            app.VIOLamp.Position = [223 126 25 25];
            app.VIOLamp.Color = [1 0 0];

            % Create CurrentLampLabel
            app.CurrentLampLabel =
                uilabel(app.CommunicationsPearametersPanel_2);
            app.CurrentLampLabel.HorizontalAlignment = 'right';
            app.CurrentLampLabel.Position = [138 78 71 22];
            app.CurrentLampLabel.Text = 'Current';

            % Create CurrentLamp
            app.CurrentLamp = uilamp(app.CommunicationsPearametersPanel_2);
            app.CurrentLamp.Position = [224 77 25 25];

            % Create BallLampLabel
            app.BallLampLabel = uilabel(app.CommunicationsPearametersPanel_2);
            app.BallLampLabel.HorizontalAlignment = 'right';
            app.BallLampLabel.Position = [138 29 71 22];
            app.BallLampLabel.Text = 'Ball';

            % Create BallLamp
            app.BallLamp = uilamp(app.CommunicationsPearametersPanel_2);
            app.BallLamp.Position = [224 28 25 25];

            % Create DistanceControlPanel
            app.DistanceControlPanel = uipanel(app.UIFigure);
            app.DistanceControlPanel.AutoResizeChildren = 'off';
            app.DistanceControlPanel.TitlePosition = 'centertop';
            app.DistanceControlPanel.Title = 'Distance Control';
            app.DistanceControlPanel.Position = [281 400 563 190];

            % Create ButtonGroup
            app.ButtonGroup = uibuttongroup(app.DistanceControlPanel);
            app.ButtonGroup.AutoResizeChildren = 'off';
            app.ButtonGroup.ForegroundColor = [0.9412 0.9412 0.9412];
            app.ButtonGroup.Position = [2 5 372 163];

            % Create PositionSlider
            app.PositionSlider = uislider(app.ButtonGroup);
            app.PositionSlider.Limits = [-25 25];
            app.PositionSlider.MajorTicks = [-25 -15 -5 0 5 15 25];
            app.PositionSlider.Orientation = 'vertical';
            app.PositionSlider.ValueChangedFcn = createCallbackFcn(app,
                @PositionSliderValueChanging, true);
            app.PositionSlider.Position = [248 9 3 141];
```

```
612
613             % Create mmLabel
614             app.mmLabel = uilabel(app.ButtonGroup);
615             app.mmLabel.Position = [289 2 25 17];
616             app.mmLabel.Text = 'mm';
617
618             % Create PatternSelectorKnobLabel
619             app.PatternSelectorKnobLabel = uilabel(app.ButtonGroup);
620             app.PatternSelectorKnobLabel.HorizontalAlignment = 'center';
621             app.PatternSelectorKnobLabel.Position = [76 5 91 22];
622             app.PatternSelectorKnobLabel.Text = 'Pattern Selector';
623
624             % Create PatternSelectorKnob
625             app.PatternSelectorKnob = uiknob(app.ButtonGroup, 'discrete');
626             app.PatternSelectorKnob.Items = {'Off', 'Triangular', 'Sine',
                    'Constant', 'Trapazoidal'};
627             app.PatternSelectorKnob.Position = [57 29 97 97];
628
629             % Create mmLabel_2
630             app.mmLabel_2 = uilabel(app.ButtonGroup);
631             app.mmLabel_2.Position = [287 141 25 21];
632             app.mmLabel_2.Text = 'mm';
633
634             % Create PeriodSecSliderLabel
635             app.PeriodSecSliderLabel = uilabel(app.DistanceControlPanel);
636             app.PeriodSecSliderLabel.HorizontalAlignment = 'right';
637             app.PeriodSecSliderLabel.Position = [433 98 72 22];
638             app.PeriodSecSliderLabel.Text = 'Period (Sec)';
639
640             % Create PeriodSecSlider
641             app.PeriodSecSlider = uislider(app.DistanceControlPanel);
642             app.PeriodSecSlider.Limits = [2 10];
643             app.PeriodSecSlider.MajorTicks = [2 3 4 5 6 7 8 9 10];
644             app.PeriodSecSlider.MinorTicks = [];
645             app.PeriodSecSlider.Position = [389 155 150 3];
646             app.PeriodSecSlider.Value = 5;
647
648             % Create AmplitudemmSliderLabel
649             app.AmplitudemmSliderLabel = uilabel(app.DistanceControlPanel);
650             app.AmplitudemmSliderLabel.HorizontalAlignment = 'right';
651             app.AmplitudemmSliderLabel.Position = [424 14 90 22];
652             app.AmplitudemmSliderLabel.Text = 'Amplitude (mm)';
653
654             % Create AmplitudemmSlider
655             app.AmplitudemmSlider = uislider(app.DistanceControlPanel);
656             app.AmplitudemmSlider.Limits = [0 15];
657             app.AmplitudemmSlider.MajorTicks = [0 5 10 15];
658             app.AmplitudemmSlider.Position = [389 71 150 3];
659             app.AmplitudemmSlider.Value = 15;
660
661             % Create positionEditField
```

```
662             app.positionEditField = uieditfield(app.DistanceControlPanel,
                    'text');
663             app.positionEditField.ValueChangedFcn = createCallbackFcn(app,
                    @positionEditFieldValueChanged, true);
664             app.positionEditField.HorizontalAlignment = 'center';
665             app.positionEditField.Position = [294 78 37 22];
666
667             % Create mmLabel_3
668             app.mmLabel_3 = uilabel(app.DistanceControlPanel);
669             app.mmLabel_3.Position = [342 80 25 17];
670             app.mmLabel_3.Text = 'mm';
671
672             % Create MeasrementsPanel
673             app.MeasrementsPanel = uipanel(app.UIFigure);
674             app.MeasrementsPanel.AutoResizeChildren = 'off';
675             app.MeasrementsPanel.TitlePosition = 'centertop';
676             app.MeasrementsPanel.Title = 'Measrements';
677             app.MeasrementsPanel.Position = [846 399 249 191];
678
679             % Create ReferencePositionmmEditFieldLabel
680             app.ReferencePositionmmEditFieldLabel =
                    uilabel(app.MeasrementsPanel);
681             app.ReferencePositionmmEditFieldLabel.HorizontalAlignment =
                    'right';
682             app.ReferencePositionmmEditFieldLabel.Position = [3 132 138 22];
683             app.ReferencePositionmmEditFieldLabel.Text = 'Reference Position
                    (mm)';
684
685             % Create DisplacementmmEditFieldLabel
686             app.DisplacementmmEditFieldLabel = uilabel(app.MeasrementsPanel);
687             app.DisplacementmmEditFieldLabel.HorizontalAlignment = 'right';
688             app.DisplacementmmEditFieldLabel.Position = [1 99 110 22];
689             app.DisplacementmmEditFieldLabel.Text = 'Displacement (mm)';
690
691             % Create VelocitymmsEditFieldLabel
692             app.VelocitymmsEditFieldLabel = uilabel(app.MeasrementsPanel);
693             app.VelocitymmsEditFieldLabel.HorizontalAlignment = 'right';
694             app.VelocitymmsEditFieldLabel.Position = [7 59 88 22];
695             app.VelocitymmsEditFieldLabel.Text = 'Velocity (mm/s)';
696
697             % Create ModeSelected
698             app.ModeSelected = uilabel(app.MeasrementsPanel);
699             app.ModeSelected.Position = [16 21 224 22];
700             app.ModeSelected.Text = 'Mode';
701
702             % Create dist_label
703             app.dist_label = uilabel(app.MeasrementsPanel);
704             app.dist_label.BackgroundColor = [1 1 1];
705             app.dist_label.HorizontalAlignment = 'center';
706             app.dist_label.Position = [175 95 54 22];
707             app.dist_label.Text = '';
```

```
708
709         % Create ref_label
710         app.ref_label = uilabel(app.MeasrementsPanel);
711         app.ref_label.BackgroundColor = [1 1 1];
712         app.ref_label.HorizontalAlignment = 'center';
713         app.ref_label.Position = [175 132 54 22];
714         app.ref_label.Text = '';
715
716         % Create vel_label_3
717         app.vel_label_3 = uilabel(app.MeasrementsPanel);
718         app.vel_label_3.BackgroundColor = [1 1 1];
719         app.vel_label_3.HorizontalAlignment = 'center';
720         app.vel_label_3.Position = [175 59 54 22];
721         app.vel_label_3.Text = '';
722
723         % Create DataAquisiationsPanel
724         app.DataAquisiationsPanel = uipanel(app.UIFigure);
725         app.DataAquisiationsPanel.AutoResizeChildren = 'off';
726         app.DataAquisiationsPanel.TitlePosition = 'centertop';
727         app.DataAquisiationsPanel.Title = 'Data Aquisiations';
728         app.DataAquisiationsPanel.Position = [1 314 1094 86];
729
730         % Create LoggerONOFFButton
731         app.LoggerONOFFButton = uibutton(app.DataAquisiationsPanel,
                'state');
732         app.LoggerONOFFButton.ValueChangedFcn = createCallbackFcn(app,
                @LoggerONOFFButtonValueChanged, true);
733         app.LoggerONOFFButton.Text = 'Logger ON/OFF';
734         app.LoggerONOFFButton.Position = [36 16 102 39];
735
736         % Create PlotLogButton
737         app.PlotLogButton = uibutton(app.DataAquisiationsPanel, 'push');
738         app.PlotLogButton.ButtonPushedFcn = createCallbackFcn(app,
                @PlotLogButtonPushed, true);
739         app.PlotLogButton.Position = [487 16 102 39];
740         app.PlotLogButton.Text = 'Plot Log';
741
742         % Create ClearLogButton
743         app.ClearLogButton = uibutton(app.DataAquisiationsPanel, 'push');
744         app.ClearLogButton.ButtonPushedFcn = createCallbackFcn(app,
                @ClearLogButtonPushed, true);
745         app.ClearLogButton.Position = [938 16 102 39];
746         app.ClearLogButton.Text = 'Clear Log';
747
748         % Create CommandPromptPanel
749         app.CommandPromptPanel = uipanel(app.UIFigure);
750         app.CommandPromptPanel.AutoResizeChildren = 'off';
751         app.CommandPromptPanel.TitlePosition = 'centertop';
752         app.CommandPromptPanel.Title = 'Command Prompt';
753         app.CommandPromptPanel.Position = [1 3 258 311];
754
```

```
            % Create ResultsingraphicalformatPanel
            app.ResultsingraphicalformatPanel = uipanel(app.UIFigure);
            app.ResultsingraphicalformatPanel.AutoResizeChildren = 'off';
            app.ResultsingraphicalformatPanel.TitlePosition = 'centertop';
            app.ResultsingraphicalformatPanel.Title = 'Results in graphical
                format';
            app.ResultsingraphicalformatPanel.Position = [258 1 585 313];

            % Create UIAxes
            app.UIAxes = uiaxes(app.ResultsingraphicalformatPanel);
            title(app.UIAxes, 'System Response')
            xlabel(app.UIAxes, 'Time (sec)')
            ylabel(app.UIAxes, 'Displacement (mm)')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.Position = [1 4 552 278];

            % Create StartStopButton
            app.StartStopButton =
                slrealtime.ui.control.StartStopButton(app.UIFigure);
            app.StartStopButton.Position = [849 591 54 47];

            % Create AnimationPanel
            app.AnimationPanel = uipanel(app.UIFigure);
            app.AnimationPanel.AutoResizeChildren = 'off';
            app.AnimationPanel.TitlePosition = 'centertop';
            app.AnimationPanel.Title = 'Animation';
            app.AnimationPanel.Position = [843 1 253 313];

            % Create AnimationAxes
            app.AnimationAxes = uiaxes(app.AnimationPanel);
            zlabel(app.AnimationAxes, 'Z')
            app.AnimationAxes.XTick = [];
            app.AnimationAxes.ZTick = [];
            app.AnimationAxes.Position = [-1 18 253 264];

            % Create StopTimeEditField
            app.StopTimeEditField =
                slrealtime.ui.control.StopTimeEditField(app.UIFigure);
            app.StopTimeEditField.Position = [920 604 52 20];

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = relese_3_7

```

```
            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

D Matlab Simulation Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Defining Constants %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ts = 0.01; %% sample time (seconds)

g=9.8066; %% gravitational accelaration (m/s^2)

d=0.0520; %% distance ohe ball from the top (Meter)

m= 0.0920; %% weight of the ball (Kilogram)

i0=0.11;  %%offset current (Ampers)

k= (2*m*g*(d/2)^2)/(i0)^2;  %% Coil Constant (Henry-Meter-turns^2)
x0 = d/2; %% Equilibrium position

xdot0 = 0; %% Equilibrium Velocity

pole_1=-5; %% Desired Poles
pole_2=-5;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Defining State Space %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A=[0 (4*g/d);1 0];            %% System Matrix
B= [(1/d)*sqrt(8*g*k/m);0];  %% Input Matrix
C= eye(2);                   %% Output Matrix
ct = [0 1];
D=[0;0];                      %% Feedforward Matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Time continuous statespace representation of the Plant
plant_ss =ss(A,B,C,D,Ts);
%Time continuous transferfunction representation of the Plant
[num,den]=ss2tf(A,B,C,D);
plant_tf=tf(num(2,:),den);
%Time discrete statespace representation of the Plant
plant_ss_disc=c2d(ss(A,B,C,D),Ts);
[AD,BD,CD,DD]=ssdata(plant_ss_disc);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[Gm_ol,Pm_ol,Wcg_ol,Wcp_ol] = margin(plant_tf); %%openloop system is unstable
bode(plant_tf);
grid;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Time Continuous Controller Design %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Controllability matrix
controlability_matrix= [B A*B];
% Inverse of Controllability matrix
controlability_inv = (controlability_matrix)^-1;
% Akermans method consider the las row of inverse of Controllability matrix
```

```
last_row_controlability_inv=controlability_inv(end,:);
% Coefficient of second order equation desired poles
alpha = poly([pole_1,pole_2]);
% Ackermans formula for Feedback controller
feedback_cnt = last_row_controlability_inv*( alpha(3)*eye(2) + alpha(2)*A +
    alpha(1)*A*A);
% Preamplifier
pre_amp = 1/(ct*((B*feedback_cnt-A)^-1)*B);
% Feedback controller with additional feedback
feedback_cnt_s = feedback_cnt(1:2)-(pre_amp*ct);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Time Discrete Controller Design %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s = tf('s');
PD_contr = 1*feedback_cnt(1,1)*s+1*feedback_cnt(1,2);
closeloop_tf = feedback(plant_tf*PD_contr,1);
%% Discrete_controller
% controlability matrix for discrete system
controlability_matrix_disc = [BD AD*BD];
%Inverse of Discrete controlability matrix
controlability_inv_disc = inv(controlability_matrix_disc);
% extractig last row of inverse controlability matrix
last_row_controlability_inv_disc = controlability_inv_disc(end,:);
%Transfer function for desired poles
tf_p_cont=tf([1],poly([pole_1 pole_2]));
% Converting transfer function to discrete
tf_p_disc=c2d(tf_p_cont,Ts,'tustin');
[zeros_d,poles_d]=tfdata(tf_p_disc,'v');
% Ackermans formula for Feedback controller
feedback_cnt_disc =
    last_row_controlability_inv_disc*(poles_d(3)*eye(2)+poles_d(2)*AD+poles_d(1)*
    AD^2);
% Preamplifier
pre_amp_disc =1/(ct*((BD*feedback_cnt_disc-AD)^-1)*BD);
feedback_cnt_disc_s = feedback_cnt_disc(1:2)-(pre_amp*ct);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Analysis of Time Continuous Controller %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% State space Representation of Closeloop system
% Necessary for deriving the BodePlot and Unit Step Response
% Closeloop SS W/O Prescalar : Gain & Phase Margins
clsloop_ss = ss((A-B*feedback_cnt),B,C,D);
% Closeloop SS With Prescalar : step response
clsloopsys_ss=feedback(ss((A-B*feedback_cnt_s),B,C,D)*pre_amp,[1 1]);
% Transfer Function representation of Closeloop System
[num_cl,den_cl]=ss2tf((A-B*feedback_cnt),B,C,D);
clsloop_tf=tf(num_cl(2,:),den_cl);
% Closeloop TF with additional Feedback
clsloopsys_tf = feedback(clsloop_tf*pre_amp,[1]);

```

```
figure('Name','Pole-Zero Plot of 1) Open loop system 2)Close loop feedback
    System','NumberTitle','off');
pzplot(plant_tf,'r',clsloop_tf,'c')
grid;

[Gm_cl,Pm_cl,Wcg_cl,Wcp_cl] = margin(clsloop_ss(2)); %%system is stable
figure('Name','Bode plot of Close Loop system','NumberTitle','off');
bode(clsloop_tf);
grid;

figure('Name','Step Response of Close Loop System','NumberTitle','off');
step(clsloopsys_ss);
grid on;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Analysis of Time DIscrete System %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% State space Representation of Closeloop system
% Necessary for deriving the BodePlot and Unit Step Response
% Closeloop SS W/O Prescalar : Gain & Phase Margins
clsloop_ss_disc = ss((A-B*feedback_cnt_disc),B,C,D,Ts);
% Closeloop SS With Prescalar : step response
clsloopsys_ss_disc=feedback(ss((A-B*feedback_cnt_disc_s),B,C,D)*pre_amp_disc,[1
    1]);
% Transfer Function representation of Closeloop System
[num_cld,den_cld]=ss2tf((A-B*feedback_cnt_disc),B,C,D);
clsloop_tf_disc=tf(num_cld(2,:),den_cld);
% Closeloop TF with additional Feedback
clsloopsys_tf = feedback(clsloop_tf_disc*pre_amp_disc,[1]);

figure('Name','Pole-Zero Plot of 1) Open loop system 2)Close loop feedback
    System','NumberTitle','off');
pzplot(plant_tf,'r',clsloopsys_tf,'c')
grid;

[Gm_cl,Pm_cl,Wcg_cl,Wcp_cl] = margin(clsloop_ss_disc(2)); %%system is stable
figure('Name','Bode plot of Close Loop system','NumberTitle','off');
bode(clsloop_ss_disc(2));
grid;

figure('Name','Step Response of Close Loop System','NumberTitle','off');
step(clsloopsys_ss_disc);
grid on;
```

E Data Set:

Measured current	Measured voltage
1.082	0.302
1.36	0.403
1.501	0.454
1.64	0.504
1.779	0.555
1.915	0.605
2.05	0.655
2.187	0.705
2.325	0.755
2.461	0.804
2.734	0.904
3.018	1.007
3.303	1.11
3.579	1.211
3.856	1.312
4.133	1.413
4.407	1.513
4.864	1.614
4.96	1.715
5.236	1.816

Table 5: Measured Current and voltage

Equilibrium Current	Equilibrium Voltage
4.76	1.59
4.632	1.55
4.135	1.37
3.772	1.23
3.465	1.12
3.23	1.04
3.143	1.01
2.96	0.94
2.85	0.90
2.8	0.88
2.7	0.85
2.6	0.81
2.4	0.74
2.25	0.68
2.2	0.67
2.1494	0.65
2.0327	0.61
1.87	0.55
1.7241	0.49
1.6114	0.45
1.5322	0.42
1.419	0.38
0.81	0.16
0.63	0.10

Table 6: Equivalent Current Voltage

Equilibrium Distance	Distance(mm)	Distance-offset(mm)	ADC Equilibrium Distance(mm)
-0.0255	-25.5	112.5	-333
-0.02	-20	118	-262
-0.018	-18	120	-236
-0.015	-15	123	-197
-0.012	-12	126	-158
-0.0102	-10.2	127.8	-135
-0.01	-10	128	-132
-0.008	-8	130	-107
-0.0072	-7.2	130.8	-96
-0.007	-7	131	-94
-0.006	-6	132	-81
0.0045	-4.5	133.5	-61
-0.0035	-3.5	134.5	-48
-0.0028	-2.8	135.2	-39
-0.001	-1	137	-16
-0.0005	-0.5	137.5	-10
0	0	138	-3
0.002	2	140	23
0.004	4	142	48
0.006	6	144	74
0.008	8	146	100
0.01	10	148	126
0.0212	21.2	159.2	271
0.028	28	166	358

Table 7: Equivalent Distance scaling

Equilibrium Current	Equilibrium Distance	f(x)
4.76	-333	2088
4.632	-262	1758
4.135	-236	1645
3.772	-197	1482
3.465	-158	1327
3.23	-135	1240
3.143	-132	1229
2.96	-107	1138
2.85	-96	1099
2.8	-94	1092
2.7	-81	1047
2.6	-61	980
2.4	-48	938
2.25	-39	909
2.2	-16	837
2.1494	-10	819
2.0327	-3	798
1.87	23	723
1.7241	48	654
1.6114	74	586
1.5322	100	522
1.419	126	462
0.81	271	196
0.63	358	94

Table 8: Equivalent Voltage and Distance

F Complete C Program for the System

```c
/* Author : Mahad Raza Khan
 * Date : 10.04.2023
 * Application: Magnetic Levitation application
 *
 * This application configures UART 16550 to baud rate 9600.
 * PS7 UART (Zynq) is not initialized by this application, since
 * bootrom/bsp configures it to baud rate 115200
 *
 * ----------------------------------------------
 * | UART TYPE   BAUD RATE                      |
 * ----------------------------------------------
 *   uartns550   9600
 *   uartlite    Configurable only in HW design
 *   ps7_uart    115200 (configured by bootrom/bsp)
 */

#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xparameters.h"
#include "xscugic.h"
#include "xil_exception.h"
#include "xscutimer.h"
#include "stdbool.h"


#define BUF_SIZE 64     // array size of user input

#define ADREG(k) (*(volatile unsigned int *)(XPAR_PMADDA_0_S00_AXI_BASEADDR + 4*k))
#define ADR_LED   0
#define ADR_PBS   0
#define ADR_DA2   1
#define ADR_STAT  1
#define ADR_AD1   2

// ---- interrupt controller -----
static XScuGic  Intc;           // interrupt controller instance
static XScuGic_Config  *IntcConfig;     // configuration instance

// ---- scu timer -----
static XScuTimer  pTimer;         // private Timer instance
static XScuTimer_Config  *pTimerConfig;   // configuration instance
#define TIMER_LOAD_VALUE  3333330     // for 1khz/1ms, count = 33333 for 10 kHz/0.1ms,
     count = 3333 for 100Kh/0.01ms.


/*
 * --------------------
 * Global varibale
 * --------------------
 */

static volatile int ISR_Count;
static volatile unsigned int ISR_Leds;
static volatile unsigned int fx_crnt_pos_adc; // current position for the ball
```

```
55 static volatile int fx_del_ref_pos;     // scaled distance form raw adc
56 static volatile u32 fx_eqC_V;       // Equilibrium current voltage
57 static volatile s16 fx_ref_pos = 0;     // refrence position
58 static volatile s16 fx_const_ref;     // constant refrence position by user
59 static volatile s32 fx_cont_out;      // control signal
60 static volatile s32 DAC_ip;         // input to DAC
61
62 static volatile u32 dac_0;
63 static volatile s32 fx_drv_0 = 0;     // derivative with previous error
64 static volatile s32 fx_intg_0 = 0;      // integral with previous error
65 static volatile s32 fx_prop_out = 0;    // Propotional output
66 static volatile s32 fx_drv_out = 0;     // derivative output
67 static volatile s32 fx_intg_out = 0;    // integral output
68
69 static volatile unsigned int Dac_0;     //  values for Digital output 1
70 static volatile unsigned int Dac_1;     //  values for Digital output 2
71
72
73 #define ISRMODE_OFF   0           // ISR OFF mode: shut down plant
74 #define ISRMODE_START 1           // ISR START mode: start the plant
75 #define ISRMODE_RUN   2           // ISR RUN mode: start the levitaiton
76 #define ISRMODE_STOP  3           // ISR STOP mode: stop the system
77
78 static volatile int ISR_Mode;
79
80 static volatile unsigned int  terminate = 0;  // flag to end the loop
81
82 /*
83  * ------------------------------------------------------------------------
84  * Function: Digital to Analog write digital data to the PMOD DAC
85  * ------------------------------------------------------------------------
86  */
87
88
89 static void WriteDA2( unsigned int ch0, unsigned int ch1)
90 {
91
92   int dcnt;
93   unsigned int da2_stat;
94
95   ADREG(ADR_DA2) = (ch1 << 16) | ch0;
96   dcnt = 0;
97   do {
98     da2_stat = ADREG(ADR_STAT) & 0x01;
99     dcnt++;
100  } while ((da2_stat == 0) && (dcnt < 500));
101  // xil_printf("\nDA2 retries was %d\n\r", dcnt);
102 }
103
104 /*
105  * ------------------------------------------------------------------------------------
106  * Funtion: Analog to Digital read Digital data transmitting from PMOD ADC
107  * ------------------------------------------------------------------------------------
108  */
109
110 static void GetAD1( u32 *ad0, u32 *ad1)
111 {
112   int dcnt;
```

```
113   unsigned int ad1_stat;
114
115   ADREG(ADR_AD1) = 0;
116   dcnt = 0;
117   do {
118     ad1_stat = ADREG(ADR_STAT) & 0x02;
119     dcnt++;
120   } while ((ad1_stat == 0) && (dcnt < 500));
121   // xil_printf("\nAD1 retries was %d\n\r", dcnt);
122   ad1_stat = ADREG(ADR_AD1);
123   *ad0 = ad1_stat & 0x0fff;
124   *ad1 = (ad1_stat >> 16) & 0x0fff;
125 }
126
127
128 /*
129  * ----------------------------------------------------------------
130  * Funtion: Distance Scaling
131  * ----------------------------------------------------------------
132  */
133
134 static void DV_lookuptable( s32 *dist)
135 {
136
137
138   *dist =  fx_crnt_pos_adc - 1675;      //diff. b/w current and mean postion
139
140 }
141
142 /*
143  * ----------------------------------------------------------------
144  * Funciton: sign generator w.r.t distance
145  * ----------------------------------------------------------------
146  */
147
148 bool pos_neg(int x)
149 {
150   int result;
151   (x & (1<<31)) ? result = 0 : (result  = 1);   //ternary condition for sign genrator
        w.r.t distance
152   return result;
153 }
154
155 /*
156  * ----------------------------------------------------------------
157  * Function: Contol signal generator
158  * ----------------------------------------------------------------
159  */
160
161 static void controller (s32 *control_func_op)
162 {
163   s8  decission = -1;
164   s32 fx_drv_1 = 0;     // derivative with error
165   s32 fx_intg_1 = 0;      // integral with error
166   s64 a2 = 126399618;     // plynomial constant of propotional
167   s32 a1 = -367440;     // plynomial coefficients of propotional
168   s64 b1 = 2449474;     // plynomial coefficients of derivative
169   s64 b2 = 2948763484;    // plynomial coefficients of derivative
```

```
170   s32 c1 = 84262;       // plynomial constant of derivative
171   s32 error = 0;        // error
172   s32 control_op = 0;     // local controller output
173
174
175
176
177
178   // error calculation: error = refrence position - current position
179   error = fx_ref_pos - fx_del_ref_pos;
180
181   // propotional calculation: (a * current position + b) * error, then 27bits right
        shift
182   fx_prop_out = ((a1 * fx_del_ref_pos + a2 ) * (error)) >> 27;
183
184
185   if (pos_neg(fx_del_ref_pos))      // function call: sign genrator w.r.t distance
186   {
187     decission = 1;            // write 1 in case of true
188   }
189
190   // derivative calculation with current position: (+/-1 * a * current position + b) *
        error, then 27bits right shift
191   fx_drv_1 = ((decission * b1 * fx_del_ref_pos + b2) * (error)) >> 27 ;
192
193   // derivative = with current error - with last error value
194   fx_drv_out = fx_drv_1 - fx_drv_0;
195
196   // integral = a * error, then 27bits right shift
197   fx_intg_1 = ( c1  * error) >> 27 ;
198
199   // integral = with current error + last error
200   fx_intg_out = fx_intg_1 + fx_intg_0;
201
202   // PID = kp * error + kd * derivative + ki * integral
203   control_op = fx_prop_out + fx_drv_out + fx_intg_out;
204
205   // storing values with last error
206   fx_drv_0 = fx_drv_1;
207   fx_intg_0 = fx_intg_1;
208
209   // updating pointer
210   *control_func_op = control_op;
211 }
212
213 /*
214  * ---------------------------------------------------------------
215  * Function: Equilibrium signal generator
216  * ---------------------------------------------------------------
217  */
218
219 static void eqli_IV(u32 *eq)
220 {
221   s64 p1 = 11874;     // plynomial coefficients of equilibrium voltage
222   s64 p2 = 12402556;    // plynomial coefficients of equilibrium voltage
223   u32 op = 0;       // function output
224   offset = 128;
225
```

```
226   // polynomial calculaiton: a * current position ^ 2 + b * current position + c, then
        22 bits right shift
227   eqC_V = ( p1 * fx_del_ref_pos * fx_del_ref_pos - p2 * fx_del_ref_pos + 3310564147)
        >> 22;
228
229   // polynomial adding control signal to the equivalent value, subtraction of offset
230   op = eqC_V + fx_cont_out - offset;
231
232   // updating pointer
233   *eq = op;
234 }
235
236
237
238 /*
239  * -------------------------------------------------------------
240  * Interrupt handler (ZYNQ private timer)
241  * -------------------------------------------------------------
242  */
243
244 static void TimerIntrHandler(void *CallBackRef)
245 {
246
247   u32 dac_1;         // pin 1 for digital out
248   u32 adc_0, adc_1;    // ADC input
249   s32 dist_add;      // function parameter to get current position
250   s32 cont_op;       // controller output
251   u32 fx_output;       // output to DAC
252   s16 v_drv_max = 2606; // threshold set for DAC
253   dac_1 = 4095;      // pin 2 for digital out, set enable pin high
254
255   XScuTimer *TimerInstance = (XScuTimer *)CallBackRef;
256
257   XScuTimer_ClearInterruptStatus(TimerInstance);
258
259
260
261   switch (ISR_Mode) {
262
263   /*  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
264       OFF mode - shut down the system
265     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ */
266
267   case ISRMODE_OFF:
268         terminate = 1;
269       break;
270
271
272   /*  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
273       transition mode from start to run mode
274     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ */
275
276
277   case ISRMODE_START:
278     ISR_Mode = ISRMODE_RUN;
279     break;
280
281   /*  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
282             RUN Mode - start levitaiton
283       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ */
284     case ISRMODE_RUN:
285
286       GetAD1(&adc_0, &adc_1);      // reading distance sensor
287       fx_crnt_pos_adc = adc_0;     // passing value to the global variable
288       DV_lookuptable(&dist_add);     // scalling current position
289       fx_del_ref_pos = dist_add;     // passing to the global variable
290       controller(&cont_op);      // fetching controller signal
291       fx_cont_out = cont_op;       // passing to the global varibale
292       eqli_IV(&fx_output);       // fetching equivalent voltage signal
293       DAC_ip = fx_output;        // passing to global varibale
294
295       // setting threshold
296
297       if (DAC_ip >= v_drv_max)
298       {
299       dac_0 = v_drv_max;         // passing threshold value
300       }
301       else {
302       dac_0 = DAC_ip;          // else pass current calculated value
303       }
304       WriteDA2(dac_0, dac_1);      // passing to the function
305       // setting up LED's chase pattern to visulize the timmer intrupt
306
307       if (ISR_Count >= 10) {
308           ISR_Count = 0;
309           ISR_Leds = (ISR_Leds << 1) & 0x0ff;
310           if (ISR_Leds == 0) {
311             ISR_Leds = 1;
312           }
313           ADREG(ADR_LED) = ISR_Leds;
314         } else {
315           ISR_Count++;
316         }
317       break;
318
319     /*  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
320         Stop mode - stop the levitaiton
321       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ */
322
323     case ISRMODE_STOP:
324
325       XScuTimer_Stop(&pTimer);
326           WriteDA2(0, 0);
327       break;
328
329
330     default:
331       ISR_Mode = ISRMODE_OFF;
332     }
333
334 }
335
336
337 /*
338  * ------------------------------------------------------------
339  * Main Body - Includes user menu for plant operations
```

```
340  * ------------------------------------------------------------
341  */
342
343
344 int main()
345 {
346
347     char cbuf[BUF_SIZE];       // to store user input
348     ISR_Mode = 0;          // initiate mode with zero/off state
349
350
351
352       init_platform();
353
354       print("--- MagLev Control ---\n\r");
355       ISR_Count = 0;
356       ISR_Leds = 0;
357
358       print(" * initialize exceptions...\n\r");
359       Xil_ExceptionInit();
360
361       print(" * lookup config GIC...\n\r");
362       IntcConfig = XScuGic_LookupConfig(XPAR_SCUGIC_0_DEVICE_ID);
363       print(" * initialize GIC...\n\r");
364       XScuGic_CfgInitialize(&Intc, IntcConfig, IntcConfig->CpuBaseAddress);
365
366     // Connect the interrupt controller interrupt handler to the hardware
367       print(" * connect interrupt controller handler...\n\r");
368     Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_IRQ_INT,
369           (Xil_ExceptionHandler)XScuGic_InterruptHandler, &Intc);
370
371       print(" * lookup config scu timer...\n\r");
372       pTimerConfig = XScuTimer_LookupConfig(XPAR_XSCUTIMER_0_DEVICE_ID);
373       print(" * initialize scu timer...\n\r");
374       XScuTimer_CfgInitialize(&pTimer, pTimerConfig, pTimerConfig->BaseAddr);
375       print(" * Enable Auto reload mode...\n\r");
376     XScuTimer_EnableAutoReload(&pTimer);
377       print(" * load scu timer...\n\r");
378       XScuTimer_LoadTimer(&pTimer, TIMER_LOAD_VALUE);
379
380       print(" * set up timer interrupt...\n\r");
381       XScuGic_Connect(&Intc, XPAR_SCUTIMER_INTR, (Xil_ExceptionHandler)
    TimerIntrHandler,
382                 (void *)&pTimer);
383       print(" * enable interrupt for timer at GIC...\n\r");
384       XScuGic_Enable(&Intc, XPAR_SCUTIMER_INTR);
385       print(" * enable interrupt on timer...\n\r");
386       XScuTimer_EnableInterrupt(&pTimer);
387
388     // Enable interrupts in the Processor.
389       print(" * enable processor interrupts...\n\r");
390     Xil_ExceptionEnable();
391
392       // check if we are still alive...
393       // xval = ADREG(ADR_LED);
394       // xil_printf("Buttons/Switches = %x\n\r", xval);
395
396     //make sure there is no output
```

```
397       WriteDA2(0, 0);
398
399      do {
400     print(">>> ");
401     int ref;              // storing user i/p refrence position
402     fgets(cbuf, BUF_SIZE, stdin);   // reads and store in cbuf
403
404
405     if (cbuf[0] == 'x') {
406       ISR_Mode = ISRMODE_OFF;     // off mode
407       XScuTimer_Start(&pTimer);   // start interrupt timmer
408     } else if (!strncmp(cbuf, "help", 4)) {
409       print("help             -  this command\n\r"); // \r bring cursor to intital
    position
410       print("x                -  shut down\n\r");
411       print("s              -  start\n\r");
412       print("p              -  stop\n\r");
413       print("d              -  current position info\n\r");
414       print("r              -  input reference value\n\r");
415
416     }
417      else if (cbuf[0] == 'd') {
418             xil_printf("==> pos:%d - fx_crnt_pos_adc:%d - Eq_Volts:%d - Propotional:%d
     - derivative:%d - integral:%d - control_op:%d - DAC_ip:%d - DAC_val:%d\n\r",
    fx_crnt_pos_adc,fx_del_ref_pos, eqC_V, fx_prop_out, fx_drv_out, fx_prop_out,
    fx_cont_out, DAC_ip, dac_0 );
419       XScuTimer_Start(&pTimer);
420     }
421      else if (cbuf[0] == 's') {
422             print("\n\r * starting levitation...\n\r");
423             ISR_Mode = ISRMODE_START;     // Start mode
424       XScuTimer_Start(&pTimer);     // start interrupt timmer
425     }
426      else if (cbuf[0] == 'p') {
427             print("\n\r*** turning off levitation***\n\r");
428       ISR_Mode = ISRMODE_STOP;      // stop mode
429     }
430      else if (cbuf[0] == 'r') {
431        xil_printf("\n\r * reference change%d...\n\r");
432
433           // checking if value exist
434            if (sscanf(&cbuf[1], "%d", &ref) != 1) {
435             xil_printf(" *** %d %d unit conversion error\n\r",ref);
436            }
437
438            else{
439              fx_ref_pos = ref;      // send given refrence value to the global
    variable
440             xil_printf(" ***value to function %d\n\r",fx_ref_pos);
441            }
442      }
443
444       else {
445       // msg for invalid command
446       print("** unknown command\n\r");
447
448     }
449     } while (terminate == 0);   // keep in loop until value is changed
```

```
450
451
452   // shutting down - reseting to default
453   print("* Program Exit");
454     WriteDA2(0, 0);
455     print("shutting down...\n\r");
456     XScuTimer_Stop(&pTimer);
457   Xil_ExceptionDisable();
458     XScuTimer_DisableInterrupt(&pTimer);
459     XScuGic_Disable(&Intc, XPAR_SCUTIMER_INTR);
460     ADREG(ADR_LED) = 0x0;
461     print("Thank you for using Zedboard IVP v0.1.\n\r");
462     cleanup_platform();
463     print("Thank you for MagLev Control System by ESD GROUP.\n\r");
464     cleanup_platform();
465     return 0;
466 }
```

Req ID	Reference	Functional Requirements			Completed		Incomplete		Partial Completion		Implemented + Renewed		Status/Comment
		Main Requirement	Category	Requirement Description		Phase/Responsibility							
				Simulation Requirements	Modelling/ prototyping		Developement		Implementation		Testing		
REQ-1		Mathematical Modelling of Magnetic Levitation Plant	D	Design a Non-Linear equations of the Magnetic Levitation System with the help of general laws of physics.	✓	Ruturaj		-		-		-	
REQ-2		Non-linear model development in MATLAB	D	Design a Non-Linear Model for the Magnetic Levitation.	✓	Ruturaj		-		-		-	
REQ-3		Linearisation and State Space Representation	D	Decide a working point and Linearize the Magnetic Levitation Model around the working point	✓	Ruturaj		-		-		-	
REQ-4		Full State Feedback PD controller Design	D	Full State Feedback PD controller Design	⚑	Ruturaj		-		-		-	
REQ-5		Discretise PD Controller	D		✓	Ruturaj		-		-		-	
		Data Logging Pane	NF	Create a group of signal namely Position, Reference, Controller Current and export thru signal to SIMULINK Data Inspector for quick troubleshooting.	✓	Ruturaj		-		-		-	
				Modelling Requirements							⚑		
REQ-6		Distance Measurement Subsystem		Read the voltage output of Laser distance sensor (DT20HI) and centre the distance around working point such that vertical upward motion corresponds to +d/2 displacement and vertical downward motion corresponds to -d/2 displacement. Where d is the air gap distance.	✓	Ruturaj		-	✓	Khubayib	✓	Ruturaj	
		Fixed point Scaling of **distance measurement function** in 12bit					✓	Ruturaj Mahad	✓	Mahad			
REQ-7		VIO Converter Subsystem		Convert the Manipulating variable (current) [0 5A] to a corresponding voltage input of [0 2.09V]	✓	Ruturaj		-	✓	Khubayib	-		
REQ-8		Equilibrium Current Subsystem		Developed a correlation between position of the ball and Equilibrium current in order to keep the plant in linear operating range.	✓	Ruturaj Khubayib		-	✓	Ruturaj Khubayib	✗	Ruturaj	
		Fixed point Scaling of **Equilibrium Current function** in 12bit			✓	Ruturaj Mahad	✓	Ruturaj Mahad Sravya	✓	Mahad			
											!		

REQ-9		High Gain Controller		Design a PD controller to stabilise the plant. Choose Gains of the controller by observing the response of the plant.		Ruturaj		-		-			
		High Gain [Gain scheduling] Controller			✓	Ruturaj Khubayib		-		-	!	Ruturaj	
		Developement of **Controller Function** in Fixed Point Format 32.16			✓	Ruturaj Mahad	✓	Ruturaj Mahad	✓	Mahad	✓	Ruturaj	
		Developement of Masked Subsystem for tunning of Waveform Generator Period and Amplitude			✓	Ruturaj		-		-		-	
				Hardware Requirements									
REQ-10		PC to FPGA interfacing	F	PC to FPGA interfacing - Communication between PC and FPGA board done by UART	✓								
REQ-11		FPGA to test stand	F	For Connecting FPGA board and test stand we are using an interfacing board which provide ADC/DAC and Digital IO	✓								
REQ-12		FPGA to test stand interface board interfacing	F	SPI communication done between FPGA and interfacing board for communicating data serially	✓								
			F	Understanding of DAC Scaling shall be determine	✓	Prashantkumar							
				Software Requirements									
REQ-13		Implementation		The digital Signal processing shall be done on a ZYNQ 7000 board using programmable logic and bare metal "C"		-		-	✓	Mahad		-	
REQ-14		Interrupt Timer		A Timer interrupt shall be generated at corresponding sample time as described in a Simulation.	✓	-		-	✓	Mahad		-	
REQ-15		ISR		ISR Routine should include the following operating stages : ON, START, RUN, OFF.	✓	Mahad		-	✓			-	
REQ-16		Position Scaling		The ball position should corresponds to 12 bits.		-		-	✓	Mahad		-	
REQ-17		Controller gains		Controller output should corresponds to the modelled output.		-	✓	Mahad	✓			-	
REQ-18		Output Voltage Scaling		The output of the controller shall be scaled down to 12 bit.		-		-	✓	Mahad		-	
REQ-19		BARR C		Adapt BARR C standards to structure C code		-		-	!	Mahad		-	
				GUI Requirements (MATLAB/JAVA)									
REQ-20	GR-1			For communication between FPGA and PC a serial communication with 115200 bits/s shall be used	✓								

REQ-21	GR-2			~~For connecting the FPGA board and ML PMOD modules shall be used which provide ADCs/DACs.~~						
REQ-22	GR-3	Target Selector /	F	Select from a list of system independent available serial ports / Realtime Target	✓ Ruturaj	-	✓ Amaan Sravya	-		Done
REQ-23	GR-4	Start Button	F	Enabling VIO converter and control logic. Also start the levitation of the ball.	✓ Ruturaj	-	✓ Amaan	-		Done
REQ-24	GR-5	Connect Button	F	Enabling the communication between FPGA and PC.	-	-	✓ Amaan Sravya	-		Done
REQ-25	GR-6	Stop Button	F	The Stop Button should **force disable** the VIO converter as well as stop the communication between FPGA and PC	✓ Ruturaj	-	✓ Amaan Sravya	-		
REQ-26	GR-7	Position Control Slider	F	The desired position can be chosen from +25mm to -25mm on a sliding bar. The desired position should be displayed in the textbox besides.	✓ Ruturaj	-	✓ Amaan Sravya	-		Done
REQ-27	GR-8	Trapezoidal Wave	F	A Trapezoidal wave generator should generate reference signal of amplitude ±15mm with user changeable on time and off time between 2sec-10sec.	! Ruturaj	✗ Ruturaj		-	Implimented in MODEL but Variable period seems not possible in GUI since lack of space to accomodate three slider for Period, Ontime and RiseTime. Another design proposal can be used of floting window dedicated for Risetime and Ontime sliders.	Done
REQ-28	GR-9	Sine Wave	F	Generate a Sine wave with amplitude ±15mm and user changeable period between 2sec - 10 sec.	✓ Ruturaj	! Ruturaj		-		Done
REQ-29	GR-10	Triangular Wave	F	Generate a Triangular wave with amplitude ±15mm and user changeable period between 2sec - 10 sec.	✓ Ruturaj	! Ruturaj		-		Done
REQ-30	GR-11	Download Log	F	Enables the data logging and creates a EXCEL file of Displacement, Reference position, Controller Current. The excel file should be named as **ddmmyy** format	✓ Ruturaj	-		-		
REQ-31	GR-12	Plot log	F	Plot the logged data and automatically adjust the Time axis for logged time duration.	✓ Ruturaj	-		-		
REQ-32	GR-13	Clear Log	F	Clear the Graph	✓ Ruturaj	-	✓ Amaan	-		
REQ-33	GR-14	Graph Window	F	Plot the current position, Reference position on the graph window where the time is scrolling from 0 to 20 sec.	✓ Ruturaj	-	✓ Amaan	-		
REQ-34	GR-15	Animation Pane Ball		Animate the Position of the ball		-	✓ Amaan	-		
REQ-35	GR-16	Animation Pane Reference	F	Animate the Reference Position as a Marker	✓ Ruturaj	-		-		
REQ-36	GR-17	Warning Indicator -1	F	The warning must appear when the ball is absent.	✓ Ruturaj	-		-		
REQ-37	GR-18	Warning Indicator -2	F	Overcurrent Indication	✓ Ruturaj	-		-		
REQ-38	GR-19	Warning Indicator -3	F	Inverter enable Indication	✓ Ruturaj	-		-		
REQ-39	GR-20	Mode Label	NF	Lable Indicating Current Function (Lable should show OFF in red font)	✓ Ruturaj	-		-		
REQ-40	GR-21	Realtime Position Ball	F	Show Realtime Value of Ball Position	✓ Ruturaj	-		-		
REQ-41	GR-22	Realtime Reference Mark	F	Show Realtime Value of Reference Mark	✓ Ruturaj	-		-		
REQ-42	GR-23	Command window	F	Incode Troubleshooting Command	✓ From Reference Code					
REQ-43	GR-24	Command window clear	F	Clear command window old commands		-	✓ Amaan	-		
				GUI Requirement (Interlock)						
REQ-44	GR-25		NF	The Selector knob, displacement slider, Amplitude Slider should be disabled before connecting to Target to avoid accidental change in set value.	✓ Ruturaj					
REQ-45	GR-26	Exitsequence	NF	STOP Button should Disable the Vio Converter as well as set the driving variables such as Gains of function Generators, Periods of functions and selector to default value.	✓ Ruturaj					
REQ-46	GR-27	Target Absent - Force Shutdown	F	In case of Absence of Target and upon unsuccessful attempt of resetting the set variables, GUI warning : **exitseqeuence failed.**	✓ Ruturaj					
REQ-47	GR-27	Animation Reset	NF	Upon returning to **OFF** mode the animation should be back to the center position.	✓ Ruturaj					

				Safety Requirements					
REQ-26			NF	The magnetic levitation system must not be operated by a single person					
REQ-27		Output threshold	F	The current in the Coil shall not exceed above 5A.	Ruturaj	Mahad	Mahad		
REQ-28		Idel Mode	F	System should identify absences of the ball and should disable the VIO converter.	Ruturaj	Ruturaj	Mahad		
REQ-29		Self Test Function	F	Check the presence of physical Connections 1) Enable Wire (Enable VIO Converter) 2) VREF Wire (Provides analog signal to VIO Converter) 3) Distance Wire (Read Distance as analog signal)	Ruturaj	Ruturaj	Mahad		
REQ-30		Ball presence		Ball presence should checked before the system starts	Ruturaj	Mahad	Mahad		