ABSTRACT

Title of thesis: SWING-UP AND STABILIZATION OF THE DOUBLE FURUTA PENDULUM

Samvrit Srinivas
Master of Science, 2018

Thesis directed by: Professor William Levine
Electrical and Computer Engineering

The problem of stabilization of an inverted pendulum is a common experiment in controls laboratories. The objective of the experiment is to balance a bar upright by providing appropriate motion to its base, similar to balancing a tennis racket upright on the palm of a human hand. The problem can be made more challenging by trying to balance two serially-joined bars instead of one. In this thesis, such a problem is considered. A mathematical model of the system is developed, and ideas of swing-up and stabilization of the two serial bars (mechanical links) are explored. A prototype is developed for validating the control system, and the details of the hardware design are discussed.

SWING-UP AND STABILIZATION OF THE DOUBLE FURUTA PENDULUM

by

Samvrit Srinivas

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2018

Advisory Committee:
Dr. William Levine, Chair/Advisor
Dr. Nuno C. Martins
Dr. Nikhil Chopra

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I would like to express my deep gratitude for my advisor, Professor William Levine, for giving me the opportunity to work on this challenging and extremely interesting project. His deep insights and intuitions on the project have been the driving forces for my work. He has always made himself available for help and advice. It has been an immense pleasure to work with and learn from him. I also thank Dr. Nikhil Chopra and Dr. Nuno C. Martins for agreeing to serve on my thesis committee and sparing their valuable time for reviewing my work.

I would like to extend my gratitude to our program director, Dr. John MacCarthy, for being extremely helpful in every aspect of the masters program. He has not only been a great mentor for academics, but also for developing a strong work ethic for all the MSSE students.

Acknowledgment is due for the help and support from the staff of TerrapinWorks, especially Nitay Ravin, from whom I learned a great deal of electronics prototyping. The ability to use the TerrapinWorks facilities was one of the enabling factors for the successful development of the prototype. I am also grateful to the lab manager at the Robot Realization Laboratory, Ivan Penskiy, who was instrumental in providing me space to work on the project, and for training me on the essential testing equipment.

I thank my peers in the MSSE program, Amar Vamsi Krishna, Kersasp Cawasji, Kunal Mehta and Rishabh Agarwal, for having been an incredible support system. I have learned a lot from each of them over the past two years.

I express deep gratitude to my parents, Mamatha and Datta, for being ever so supportive in my endeavors. I thank my mother for making the final days of the last semester easier to handle. I thank my father for providing ideas for the mechanical design and also for overseeing the manufacturing of the prototype.

Last but not the least, I thank Ripple Technologies, Bangalore, for manufacturing the mechanical parts at no cost; and NBK America, for providing free samples of essential mechanical components for the prototype.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CAD	Computer Aided Design
COM	Center of Mass
COTS	Commercial off-the-shelf
D-H	Denavit-Hartenberg
DMOC	Discrete Mechanics and Optimal Control
FOS	Factor of Safety
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
OP	Operating Point
PCB	Printed circuit board
PWM	Pulse Width Modulation
SMD	State Machine Diagram
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter

Chapter 1: Introduction

A double rotational inverted pendulum – also known as the double Furuta pendulum – is a nonlinear dynamic system in which three serial mechanical links are connected by three revolute joints. The first link is actuated by a motor with a vertical axis of rotation, and the other two joints are not actuated. In other words, the double rotational inverted pendulum is an inverted pendulum system on a rotational cart instead of a linear cart. Initially, the second and third links (pendulum segments) are suspended downwards. The goal is to swing these two links up, and balance them at the upright position.

Inverted pendulums are common equipment in any controls lab. The advantage of the rotational-type inverted pendulum over the linear pendulum is that they do not have limits on the movement of the cart. Studying the control of such nonlinear unstable systems is beneficial in being able to apply the modeling and control techniques in other similar systems such as humanoid robotics.

The objectives of this thesis were to develop a mathematical model of the double Furuta pendulum, perform simulations with different system parameters, and finally build a prototype that can be used to validate the control system.

The novel ideas and results that are presented in this thesis are:

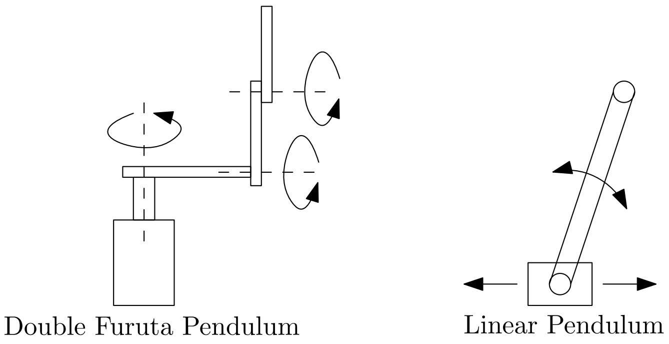


Figure 1.1: Furuta vs Linear pendulum

1. A mathematical model of the double Furuta pendulum using an adaptation of the Denavit-Hartenberg convention

2. A simple strategy for transitioning from the Up-Down configuration to the Up-Up configuration

3. Swing-up performance comparison between different system designs

4. A physical prototype that uses wireless transmission of data from the joint sensors

The thesis is organized as follows. First, the development of the mathematical model is discussed, which details the kinematics and dynamics of the system. Next, the control strategies for different system configurations are discussed. Then, the simulation results of different system designs are presented and compared. Finally the details of the physical prototype are discussed.

Chapter 2: Literature Survey

The idea of the rotational inverted pendulum was conceived by Katsuhisa Furuta et. al., and was presented in a paper in 1991 [1]. Several researchers have since worked on it, investigating different swing-up and control strategies.

In the Furuta pendulum literature, the up-right balancing control has predominantly been based on the linear quadratic regulator (LQR) [1] [2] [3]. Other implementations include the feedback linearization method [4], model predictive control (MPC) [5], and neural networks [6] [7] [8] [9]. The LQR, feedback linearization and MPC methods require an accurate model of the system, while the neural network method does not. The neural network method however, requires training, and much more computational power than the other control methods.

The swing-up strategy on the other hand, was initially based on a bang-bang controller [1] [2]. In [1], the bang-bang controller was implemented on a single pendulum by analyzing the phase plane of the states. In the paper by Yamakita et al., [2], a learning control was proposed for a double pendulum that generates an optimal pattern of bang-bang control input that minimizes the swing-up time. In a paper by Wiklund et al. [4], a simple energy-based controller was proposed, which was further investigated in a 1996 paper by Astrom and Furuta [10]. The latter paper extends the energy control strategy to a general mechanical system, especially the double inverted pendulum, be it the linear-type or the rotational-type. Another approach was proposed by Kobayashi et al. [11], in which a unified control strategy is used for both swing-up and balancing, using a nonlinear state-dependent Riccati equation. Yet another approach was proposed by Ismail and Liu [12], in which optimal trajectories for the swing-up of a double pendulum are derived using discrete mechanics and optimal control (DMOC). Out of the above strategies, the energy-based control [10] is the simplest to implement, and is quite intuitive.

In this thesis, the LQR method is used for balancing the pendulum segments in the up-right position, and the energy-based control is used for their swing-up. The effects of the pendulum design parameters, especially the pendulum lengths, on the swing-up performance are analyzed.

Chapter 3: System Model

In this chapter, the mathematical modeling of the double Furuta pendulum system is discussed. The parameters of the model described in this section corresponds to that of the reference design (where the first pendulum segment is 10% shorter than the second).

Both the energy-based swing-up and the LQR-based balancing control require the knowledge of the system model. The following approach was used to model the system:

1. The kinematics model was derived using an adaptation of the Denavit-Hartenberg (D-H) convention.

2. The nonlinear dynamics model was derived using the Euler-Lagrange formulation, using the joint variables as generalized coordinates.

3. The nonlinear dynamics was written in the state-space form, and then linearized about its unstable operating points using Taylor's expansion.

The state of the system was chosen to be the three joint angles and their velocities, i.e., $\mathbf{x} = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]^T$. The cart's angle and angular velocity were also included along with the pendulums' states because otherwise, the cart is known to drift [4].

The double Furuta pendulum has four configurations, but only three that that are of interest here. The first one is the suspended configuration, where both pendulum segments are hanging downwards (Down-Down configuration). This is a stable equilibrium. The second is the configuration where the first pendulum is upright and the second pendulum is either hanging downwards or oscillating about the downward position (Up-Down configuration). This is an unstable equilibrium. The third one is the configuration where both pendulum segments are upright (Up-Up configuration). This is also an unstable equilibrium. These configurations are illustrated in Figure 3.1.

The goal is to transfer the system from the Down-Down to the Up-Up configuration and maintain it at the latter. To do this, the intermediate Up-Down configuration is a convenient interim point in the state space. Initially, the swing-up control is used to drive the system from the Down-Down to the Up-Down configuration. Then, the LQR is used to maintain the system at the Up-Down configuration while the second pendulum segment swings up. When the second segment reaches the neighborhood of the upright position, a different LQR scheme is used to balance the system at the Up-Up configuration.

The Up-Down and Up-Up state make use of the LQR, which assumes a linear system model. Hence, the pendulum system was linearized about these two operating points, i.e., $\mathbf{OP}_1 = [0, 0, \pi, 0, 0, 0]^T$ and $\mathbf{OP}_2 = [0, 0, 0, 0, 0, 0]^T$ respectively, where the pendulum angles are zero when they are in the upright configuration.

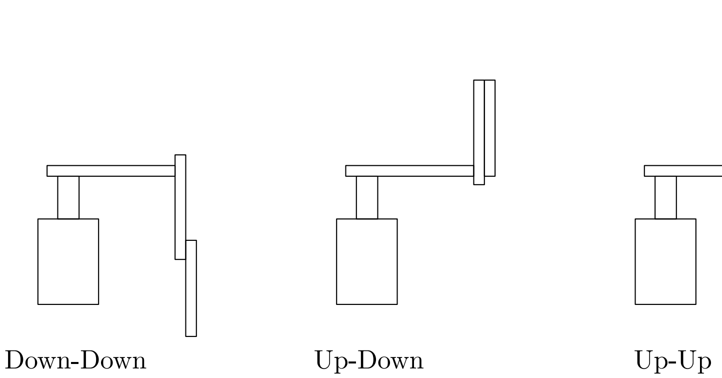


Figure 3.1: Pendulum configurations

3.1 Kinematics

The double Furuta pendulum system is assumed to be a serial kinematic chain. The kinematics of the double Furuta pendulum was modeled using an adaptation of the Denavit-Hartenberg (D-H) convention. The D-H convention allows one to reduce the number of parameters used to describe a kinematic chain from six (three rotational and three translational), to four (twist, offset, link length and rotation) [13]. These four parameters are represented by α, d, a, θ respectively. However, the convention restricts one to perform coordinate transformations only in the x and z axes. For the Furuta pendulum, a transformation in the y-axis is a necessity. Hence, a slight modification was made in the convention to work around this issue. This method is detailed in the following section.

3.1.1 Coordinate Transformations

In order to use the standard D-H convention, one needs to perform coordinate transformations only in the x and z axes. However, the Furuta pendulum requires a transformation in the y-axis. To get around this issue, the Furuta pendulum was treated as a four-joint system with the second joint being fixed (no associated degree of freedom), i.e., a pseudo link with zero dimensions was added. The configuration of the system was then drawn such that the transformations could be performed only in the x and z axes (Figure 3.2), and the corresponding kinematic parameters were derived for the link ends (Table 3.1) as well as the link center of masses (COM) (Table 3.2). The pendulum lengths are represented by a_2 and a_3 for the first and second segments respectively.

The transformation matrix for each row in the D-H Table is computed as follows:

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The coordinate frames of a kinematic chain are represented as a series of coordinate transformations, starting from the first link to the last. Two sets of coordinate frames are assigned – at link ends and at link COMs. For the double Furuta pendulum, the coordinate transformations corresponding to link ends were obtained as follows:

$$T_1 = A_1 A_2$$

$$T_2 = T_1 A_3$$

$$T_3 = T_2 A_4$$

and the transformations corresponding to the link COMs were obtained as follows:

$$T_{c1} = A_{c1} A_{c2}$$

$$T_{c2} = T_1 A_{c3}$$

$$T_{c3} = T_2 A_{c4}$$

where A_i represents the transformation matrix corresponding to row i in Table 3.1 and A_{ci} represents the transformation matrix corresponding to row i in Table 3.2. It can be noticed that this is a slight deviation from the standard D-H convention because of the multiplication of two A matrices instead of one, for T_1 amd T_{c1}. This additional transformation is done to account for the pseudo link. This method was verified for correctness by developing another kinematic model which included performing the coordinate transformations in the y-axis, and then comparing the resulting matrix elements obtained using the two different methods. The verification method is detailed in Appendix A.

3.1.2 Normalization

A characteristic length l is defined, which is the sum of the lengths of the two pendulum segments, i.e., $l = a_2 + a_3$. The length of each pendulum segment

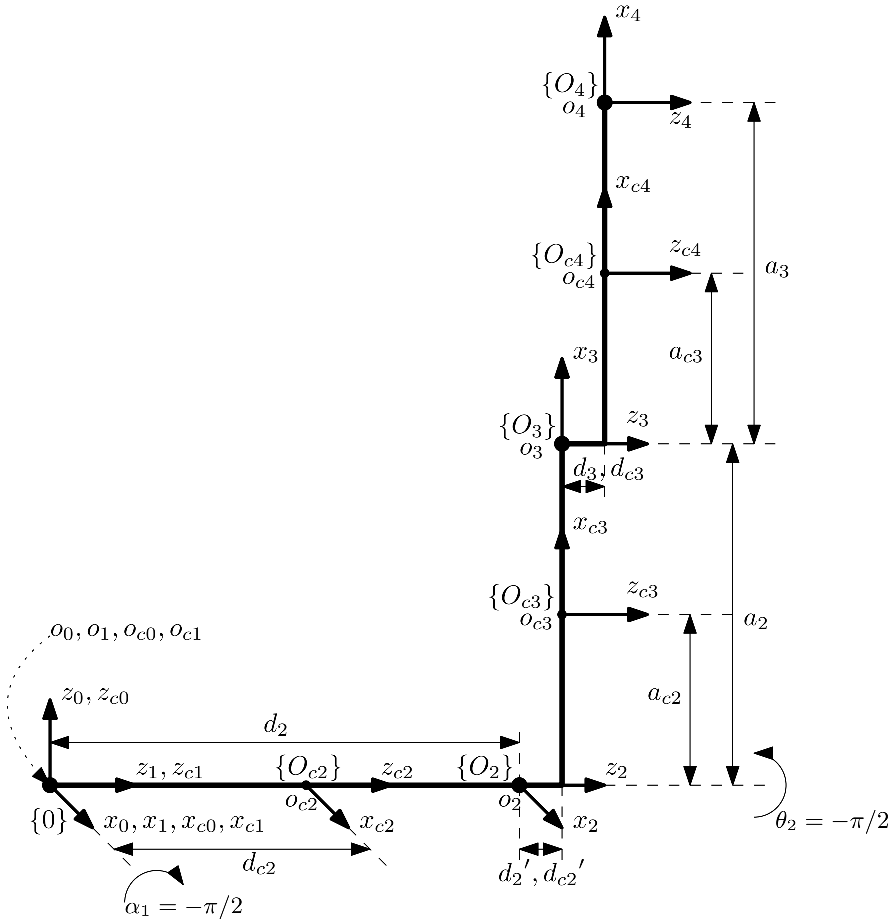


Figure 3.2: D-H Diagram

i	α	a	d	θ
1	$-\pi/2$	0	0	θ_1
2	0	0	$d_2 = 0.142$m	0
3	0	$a_2 = 0.135$m	$d_2{}' = 0$	$\theta_2 - \pi/2$
4	0	$a_3 = 0.165$m	$d_3 = 0.005$m	θ_3

Table 3.1: Kinematic parameters for link ends

i	α	a	d	θ
1	$-\pi/2$	0	0	θ_1
2	0	0	$d_{c2} = 0.0647$m	0
3	0	$a_{c2} = 0.0675$m	$d_{c2}{}' = 0$	$\theta_2 - \pi/2$
4	0	$a_{c3} = 0.0832$m	$d_{c3} = 0.005$m	θ_3

Table 3.2: Kinematic parameters for link COMs

Dimensionless Parameter	Ratio	Value
β_a	$\frac{d_2}{l}$	0.47
β_p	$\frac{a_2}{l}$	0.45
λ_a	$\frac{m_1}{m}$	1.89
λ_p	$\frac{m_2}{m}$	0.45
ν_1	$\frac{d_{c2}}{\beta_a l}$	0.45
ν_2	$\frac{a_{c2}}{\beta_p l}$	0.5
ν_3	$\frac{a_{c3}}{(1-\beta_p)l}$	0.5

Table 3.3: Dimensionless parameters for the reference design

is proportional to l, with a proportionality constant β_p. The cart length is also expressed as a fraction of the characteristic length, with a proportionality constant β_a. The distance between each link's origin and its center of mass is expressed as a fraction of the corresponding link's length, with a proportionality constant ν_i for link i. Hence, the link lengths as well as the link's center of masses are expressed as fractions of the characteristic length l.

Similarly, a characteristic mass m is defined, which is the sum of the masses of the two pendulum segments, i.e., $m = m_2 + m_3$. The masses of the cart and the two pendulum segments are expressed as a fraction of this characteristic mass. The proportionality constants are λ_a, λ_p and $(1 - \lambda_p)$ for the cart, first pendulum segment and the second pendulum segments respectively.

These proportionality constants are dimensionless parameters that can be used to scale the design and compare system performances. For the reference design, $l = 0.3$m and $m = 0.02$kgs. The dimensionless parameters are shown in Table 3.3.

3.1.3 Jacobians

The linear and angular velocities of the center of mass of each link were described using Jacobians. Spong [13] provides a simple formula to compute the Jacobians for serial manipulators when they are represented using the D-H convention. Since all the joints in the system are revolute, the formula for the i^{th} column of the Jacobian of each link was expressed as:

$$J_{vi} = [z_{i-1} \times (o_{cn} - o_{i-1})]$$

$$J_{\omega i} = [z_{i-1}]$$

where, J_{vi} and $J_{\omega i}$ are the linear and angular velocity Jacobians respectively; z_{i-1} is the axis of rotation of the previous joint; o_{cn} is the coordinates of the COM of the link in consideration, and o_{i-1} is the coordinates of the origin of the $i-1^{th}$ link, all of which are expressed in the global coordinate frame. The coordinates of the COMs were transformed from their local reference frames to the global coordinate frame by serially multiplying the transformation matrices [13].

Since our model made use of an adapted version of the D-H convention, some verification was performed to ensure that the above formula for the Jacobians holds good for this system as well. The verification process is detailed in Appendix A.

3.2 Dynamics

The dynamics of the system was derived from the kinematics using the Lagrangian formulation. The Lagrangian formulation requires the specification of a set of generalized coordinates, which is the minimum set of variables that completely describe the dynamics of a system. Since the dynamics equations are derived from the kinematics, and the joint variables describe the system using minimum variables, the joint variables itself serve as the generalized coordinates. The Lagrangian formulation is then written as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \tau \tag{3.2}$$

where L is called the Lagrangian, which is the difference between the total kinetic energy and the total potential energy of the system; q is the set of generalized coordinates ($[\theta_1, \theta_2, \theta_3]^T$); and τ which is the generalized force (torque) applied at each joint ($[\tau, 0, 0]^T$);. After performing the above differentiation and simplifying, the dynamics equations were expressed as:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau \tag{3.3}$$

In Equation 3.3, D is a 3×3 matrix, called the mass matrix. The mass matrix for the double Furuta pendulum was obtained from the following equation:

$$\begin{multline} D = m_1 J_{v1}^T J_{v1} + m_2 J_{v2}^T J_{v2} + m_3 J_{v3}^T J_{v3} + J_{\omega 1}^T R_1 I_1 R_1^T J_{\omega 1} + J_{\omega 2}^T R_2 I_2 R_2^T J_{\omega 2} \\ + J_{\omega 3}^T R_3 I_3 R_3^T J_{\omega 3} \end{multline} \tag{3.4}$$

where, m_1, m_2, m_3 are the link masses; J_{v1}, J_{v2}, J_{v3} are the linear velocity Jacobians; $J_{\omega 1}, J_{\omega 2}, J_{\omega 3}$ are the angular velocity Jacobians; R_1, R_2, R_3 are the rotation matrices that transform the local coordinates of the link COMs to the global coordinate frame; and I_1, I_2, I_3 are the inertia tensor of each link.

C is the Centripetal/Coriolis matrix which contains the Christoffel symbols. The elements of this matrix were derived using the formula:

$$c_{ij} = \sum_{i=1}^{n} \frac{1}{2} \left\{ \frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{kj}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right\} \tag{3.5}$$

where d_{ijk} are the corresponding elements from the mass matrix D.

The total potential energy of the system is the sum of the potential energies of the individual links:

$$P = m_2 g^T o_{c2} + m_3 g^T o_{c3} \tag{3.6}$$

where g is the gravitational acceleration vector ($g = [0,\ 0,\ -9.81]$); o_{c2}, o_{c3} are the coordinates of the link COMs expressed in the global coordinate frame. The gravity terms in G were then derived from the total potential energy by performing the following differentiation:

$$g_i = \frac{\partial P}{\partial q} \tag{3.7}$$

The individual mass terms in the dynamics equations can be replaced by ratios of the characteristic mass, as detailed in Section 3.1.2. The resultant D, C and G matrix elements, expressed in terms of the dimensionless parameters, are listed in Appendix B.

For designing the linear quadratic regulator, the system must be expressed in the linearized state-space form. The dynamics of the system that was obtained by Lagrangian formulation is nonlinear. Hence, it was linearized around desired operating points (Up-Down and Up-Up configurations).

Since there are three generalized coordinates, the state of the system is defined by six variables: 3 joint angles and 3 joint velocities, i.e., $\mathbf{x} = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]^T$.

The nonlinear state-space form is as follows:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{3.8}$$

where $\mathbf{x}$ is the 6×1 state vector; $\mathbf{u} = \tau$ is the input to the system, which is a scalar because there is only one actuator that provides torque input to the system.

The expression for the angular acceleration, $\ddot{q} = [\ddot{\theta}_1, \ddot{\theta}_2, \ddot{\theta}_3]^T$, was obtained by solving for $\ddot{q}$ in Equation 3.3:

$$\ddot{q} = D^{-1} \left[\tau - C(q, \dot{q})\dot{q} - G(q)\right] \tag{3.9}$$

Equation 3.9 is of the form $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$. The term $\dot{\mathbf{x}}$ is a 3×1 matrix, which contains nonlinear elements. This forms the last three elements of $\dot{\mathbf{x}}$ in Equation 3.8 [14]. The first three elements of $\dot{\mathbf{x}}$ are just the last three elements of $\mathbf{x}$. To linearize the last three elements about an operating point vector ($\mathbf{OP}$), Taylor's expansion is used:

$$\delta\dot{x}(t) = \left(\frac{\partial f(x, u)}{\partial x}\right)_{x=\mathrm{OP}, u=0} \delta x(t) + \left(\frac{\partial f(x, u)}{\partial u}\right)_{x=\mathrm{OP}, u=0} \delta u(t) \tag{3.10}$$

where x is the state vector, $\delta x(t)$ is a small deviation of the states from the operating point. The coefficients of $\delta x(t)$ and $\delta u(t)$, termed A and B respectively, are evaluated at the operating point. Thus, the linearized state-space form for the double Furuta pendulum system becomes:

$$\dot{\underline{\mathbf{x}}} = A\underline{\mathbf{x}} + B\mathbf{u} \tag{3.11}$$

where $\underline{\mathbf{x}}$ is the deviation of the state vector, $\mathbf{x}$, from the desired operating point, i.e., $\underline{\mathbf{x}} = \mathbf{x} - \mathbf{OP}$. For the reference design, at the upright operating point, the following linearized coefficients were obtained:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 216.5 & -24.4 & 0 & 0 & 0 \\ 0 & 383.4 & -138.9 & 0 & 0 & 0 \\ 0 & -473.4 & 367.3 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 8714.0 \\ 11730.0 \\ -14450.0 \end{bmatrix} \tag{3.12}$$

where mass is expressed in kilograms, length in meters, time in seconds and angle in radians.

Chapter 4: Control System

This chapter discusses the control strategy used to achieve the task of balancing the double Furuta pendulum system at its upright configuration. This consists of swinging up the pendulum segments from their suspended state to the upright state, and then balancing the segments at the latter configuration. Hence, the control strategy is broadly broken down into swing-up control and balancing control.

The approach taken was to swing up the first pendulum segment until the system reached the Up-Down state, and then swing up the second segment until the second segment comes close to the upright configuration. At this point, the controller is switched to the balancing control using the linear quadratic regulator. The Up-Down configuration is a useful intermediate control point because it is unlikely that both pendulum segments approach the near-upright position at the end of the swing-up. Hence, it is easier to balance the first segment and then swing up the second segment. The higher-level control switching was implemented as a state machine using the Stateflow tool in MATLAB.

The control strategies make use of the linear quadratic regulator (LQR), which is an optimal controller that minimizes the following quadratic objective function:

$$J = \int_0^\infty \left(x^T \mathbf{Q} x + u^T \mathbf{R} u \right) \tag{4.1}$$

where $\mathbf{Q}$ is a matrix that specifies the relative importance of maintaining each system state, x, at its operating point; and $\mathbf{R}$ is the cost of providing control input to the system. For the double Furuta pendulum system, $\mathbf{Q}$ is a diagonal 6×6 matrix where the diagonal elements represent the relative importance of each state, and $\mathbf{R}$ is a scalar value since there is only one input to the system, which is the control torque.

The control law that minimizes the objective function in Equation 4.1 is given by:

$$u = -\mathbf{K} \cdot \mathbf{x} \tag{4.2}$$

where $\mathbf{K}$ is the optimal control gain.

Fortunately, MATLAB includes an `lqr()` command that takes in the linearized system parameters A and B, the objective function parameters $\mathbf{Q}$ and $\mathbf{R}$, and outputs the optimal control gain, $\mathbf{K}$.

4.1 Down-Down to Up-Down

This section describes the method that was used to swing up the first pendulum segment to the upright position. The energy control method proposed in [4] was implemented.

The total energy of the pendulum segments was considered:

$$TE = K + P \tag{4.3}$$

where, the kinetic energy, $K = \frac{1}{2}\,\dot{q}^T \cdot D \cdot \dot{q}$, and potential energy, P, is obtained from Equation 3.6. Since the cart's energy is ignored, q_1 and $\dot{q}_1$ are substituted as zeros. The reference total energy, TE_{ref}, was computed at the Up-Up state $(\theta_2, \theta_3 = 0°)$

for different designs. The resulting equations for the total energy and the reference energy are provided in Appendix B. The error signal at every time step is computed as:

$$e = TE_{ref} - TE \tag{4.4}$$

The control signal (input torque) is then obtained as follows:

$$u = k \cdot e \cdot \operatorname{sgn}(\dot{\theta}_2 \cos\theta_2) \tag{4.5}$$

where k is a proportional gain, called swing-up constant.

The pendulum can be swung up with a single swing of the cart if k is larger than a threshold value, which depends on the system design. If k is less than this threshold value, the control law generates multiple swings. The 'sgn' function is used to determine the direction of torque that is to be applied at that particular time step. In [4], it is shown that with this control law, the total energy of the system increases when $\dot{\theta}_2 \cos\theta_2$ is positive, and decreases when it is negative, thus controlling the total energy of the system. In the reference design, k is chosen to be 10 and $TE_{ref} = 0.02951$ joules.

It can be noticed that the control input is zero when the first segment is stationary ($\dot{\theta}_2 = 0$) or when it is at the horizontal position ($\cos\theta_2 = 0$). For the former situation, a constant torque of $u = 0.2$Nm is applied, and for the latter situation, the following "arm correction" control is provided:

$$u = \alpha(\theta_1 - \theta_{1d}) - \gamma\dot{\theta}_1 \tag{4.6}$$

where α and γ are proportionality constants, and θ_{1d} is a reference value.

Parameter	Parameter Value
Q (diagonal)	[1, 400000, 0, 0, 10000, 0]
R	1
K	[-1.00, 679.96, -0.67, -4.44, 102.94, -1.49]

Table 4.1: Up-Down LQR Parameters

As mentioned previously, it is convenient to perform the swing-up and balance the first pendulum segment before swinging up the second. To balance the first segment upright, the system is linearized about its Up-Down state ($\theta_2 = 0$ and $\theta_3 = \pi$), and an LQR scheme is used that is designed to stabilize θ_2 and $\dot{\theta}_2$. Since the velocity of the first pendulum segment would be high during the initial swing-up, it requires a very tight control to stabilize it. Hence, the weights corresponding to θ_2 and $\dot{\theta}_2$ in the **Q** matrix of the LQR objective function, were given high values. The states corresponding to the second segment, i.e., θ_3 and $\dot{\theta}_3$ are inconsequential for this control scheme, hence the corresponding weights in **Q** are given as zeros. The LQR parameters and resulting control gain **K** are shown in Table 4.1.

4.2 Up-Down to Up-Up

After the swing-up of the first segment, the second segment has sufficient energy to oscillate about its downward position. The first pendulum segment can be kept close to the upright configuration, while delicately imparting motion to the pivot of the second segment, thus increasing the amplitude of its oscillation. Eventually, the second segment swings up to its near-upright configuration. Even with the tight control scheme described in Section 4.1, slight movement of the pivot exists that provides oscillation to the second segment.

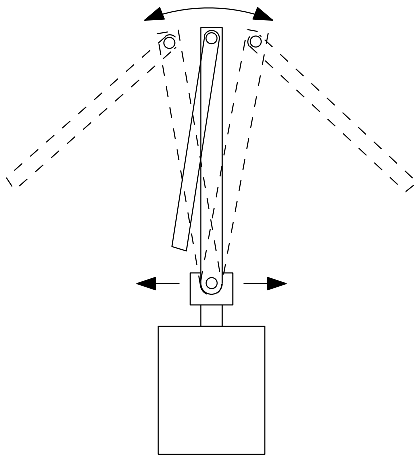


Figure 4.1: Motion of the second segment's pivot

Parameter	Parameter Value
Q (diagonal)	[1, 800, 0, 0, 10, 10]
R	1
K	[-1.00, 38.87, -41.41, -1.06, 5.28, 0.66]

Table 4.2: Mild Up-Down LQR Parameters

In order to speed up the swing-up of the second segment; or if the initial angular velocity of the second segment after the first segment's swing-up is insufficient, the movement of the first segment about its upright position is increased to provide wider motion to the pivot of the second segment. This is achieved by switching to a milder LQR control at the Up-Down configuration, where the weights for θ_2 and $\dot{\theta}_2$ are relatively smaller. The milder LQR parameters and resulting control gain **K** for the reference design is shown in Table 4.2.

4.3 Balancing Control

The balancing control was achieved using the linear quadratic regulator, where the system was linearized about the Up-Up configuration. The linearized system

Parameter	Parameter Value
Q (diagonal)	[10, 400000, 4000, 1000, 10000, 1000]
R	1
K	[3.16, -1349.33, -4290.75, 33.55, -321.43, -325.72]

Table 4.3: Up-Up LQR Parameters

parameters are shown in Section 3.2. Suitable values were chosen for the objective function parameters, **Q** and **R**.

The angles and angular velocities of the two pendulum segments need to converge to zero in order to successfully balance them at the upright configuration. Hence, the second, third, fifth and sixth diagonal elements of **Q** were given high values. The position of the cart is less important, but cannot be ignored completely, because otherwise it is known to drift. Hence, the weight corresponding to the position of the cart, i.e., the first diagonal element of **Q** was given a small value. This ensures that the cart homes into its initial position upon balancing the two pendulum segments upright. The values of **Q** need to be altered for different pendulum designs in order to successfully balance them in the upright configuration. The balancing control is considered to be successful when the pendulum angles, θ_2 and θ_3 are within 1° of the upright configuration. For the reference design, the LQR parameters and the resulting control gain **K** are shown in Table 4.3.

4.4 State Machine

The higher-level control selection was achieved by implementing a state machine using the Stateflow tool in MATLAB. A state machine is a representation of a system in terms of a finite number of states, where the system can exist only in one state at a given time, and can transition from one state to another state depending on specific "guard" conditions.

The state machine implemented for the double Furuta pendulum system uses the state vector, $\mathbf{x}$, as an input to decide which controller state it should be in. The controller state is the name given to a higher-level state that the system exists in. For example "SwingUp" is a controller state in which the system performs the swing up of the first pendulum segment. Depending on the controller state, an appropriate control law is applied. The transitions between the controller states depend on the state vector values, i.e., the guard conditions are specified in terms of $\mathbf{x}$. The controller states are represented as nodes, and the guard conditions as edges in a state machine diagram (SMD), as shown in Figure 4.3. The description of each state in the SMD is provided in Table 4.4, and the guard conditions for state transitions are shown in Table 4.5.

The overall control system implemented in Simulink is shown in Figure 4.2. The control design in Simulink was built upon the work done by Gaurav Nair [15]. The "Control State Machine" block outputs the controller state (represented by Controller ID in Table 4.4) at every time step. The "Controller" block, which is a MATLAB Function block, then decides which control law to apply to the plant, based on the controller state in that time step. The "Controller" block outputs the desired torque value based on the control law, which is then applied to the motor residing within the "Plant" subsystem. The "Plant" subsystem outputs the state vector, $\mathbf{x}$, which is fed back into the controller state machine for the next time step. A "Saturation" block is used to simulate a motor's torque limit.

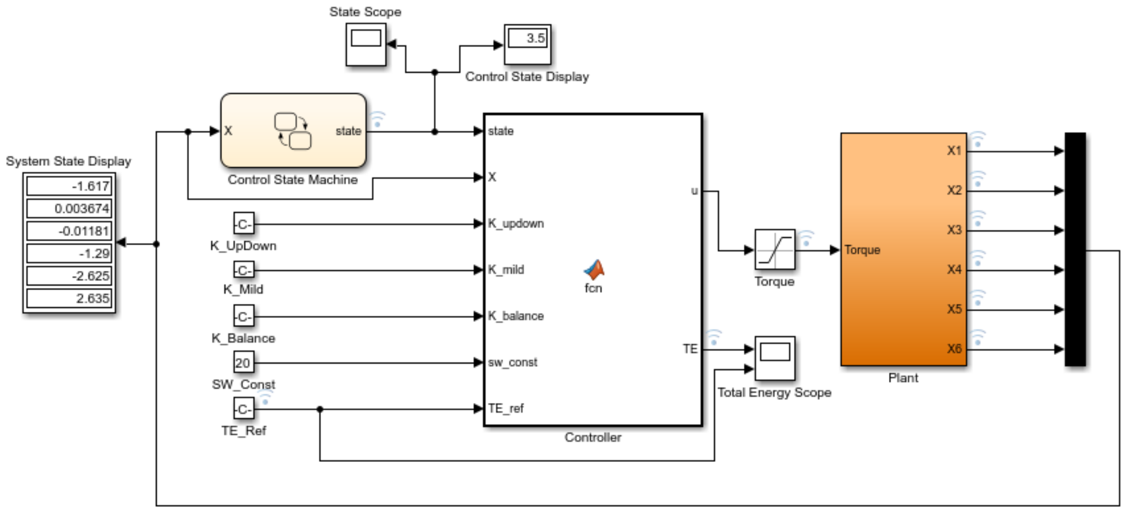


Figure 4.2: Control system implemented in Simulink

State Name	Controller ID	Description
DecisionState	-	This is an initial decision gate where the state machine decides which control mode to enter
SwingUp	1	The system performs the swing-up operation discussed in Section 4.1
ArmCorrection	1.5	The system performs the arm-correction control discussed in Section 4.1
UpDown	2	The system tightly controls the first pendulum segment at the upright position as discussed in Section 4.1
Mild	2.5	The controller provides more movement to the first segment about the vertical position as discussed in Section 4.2
Balance	3	The system controls both the pendulum segments in the up-right position
UpRight	3.5	This state indicates a successful balancing control, and is achieved when the pendulum segments are within 1° of the up-right position

Table 4.4: Description of system states in the state machine diagram

Transition	Guard Condition Logic
a	$\lvert\theta_2\rvert \geq 20^\circ$
b	$\lvert\theta_2\rvert < 20^\circ$
c	$\lvert\theta_2\rvert \leq 15^\circ \wedge \lvert\theta_3\rvert \leq 15^\circ$
d	$\lvert\theta_2\rvert \geq 89^\circ \wedge \lvert\theta_2\rvert \leq 91^\circ$
e	$\lvert\theta_2\rvert \geq 120^\circ \vee \lvert\theta_2\rvert \leq 60^\circ$
f	$\lvert\theta_2\rvert \leq 20^\circ$
g	$\lvert\theta_2\rvert > 20^\circ$
h	$\lvert\theta_2\rvert \leq 5.7^\circ \wedge \lvert\dot{\theta}_2\rvert \leq 23^\circ\mathrm{s}^{-1} \wedge \lvert\theta_3\rvert \geq 170^\circ \wedge \lvert\dot{\theta}_3\rvert \leq 1432^\circ\mathrm{s}^{-1}$
i	$\lvert\theta_2\rvert \geq 35^\circ \vee \lvert\dot{\theta}_2\rvert > 1146^\circ$
j	$\lvert\theta_3\rvert \leq 20^\circ \wedge \lvert\dot{\theta}_2\rvert \leq 114^\circ\mathrm{s}^{-1} \wedge \lvert\dot{\theta}_3\rvert \leq 401^\circ\mathrm{s}^{-1}$
k	$\lvert\theta_3\rvert \leq 20^\circ \wedge \lvert\dot{\theta}_2\rvert \leq 114^\circ\mathrm{s}^{-1} \wedge \lvert\dot{\theta}_3\rvert \leq 401^\circ\mathrm{s}^{-1}$
l	$\lvert\theta_2\rvert \leq 1^\circ \wedge \lvert\theta_3\rvert \leq 1^\circ \wedge \lvert\dot{\theta}_2\rvert \leq 10^\circ\mathrm{s}^{-1}$
m	$\lvert\theta_2\rvert > 20^\circ$

Table 4.5: Guard conditions in the state machine diagram

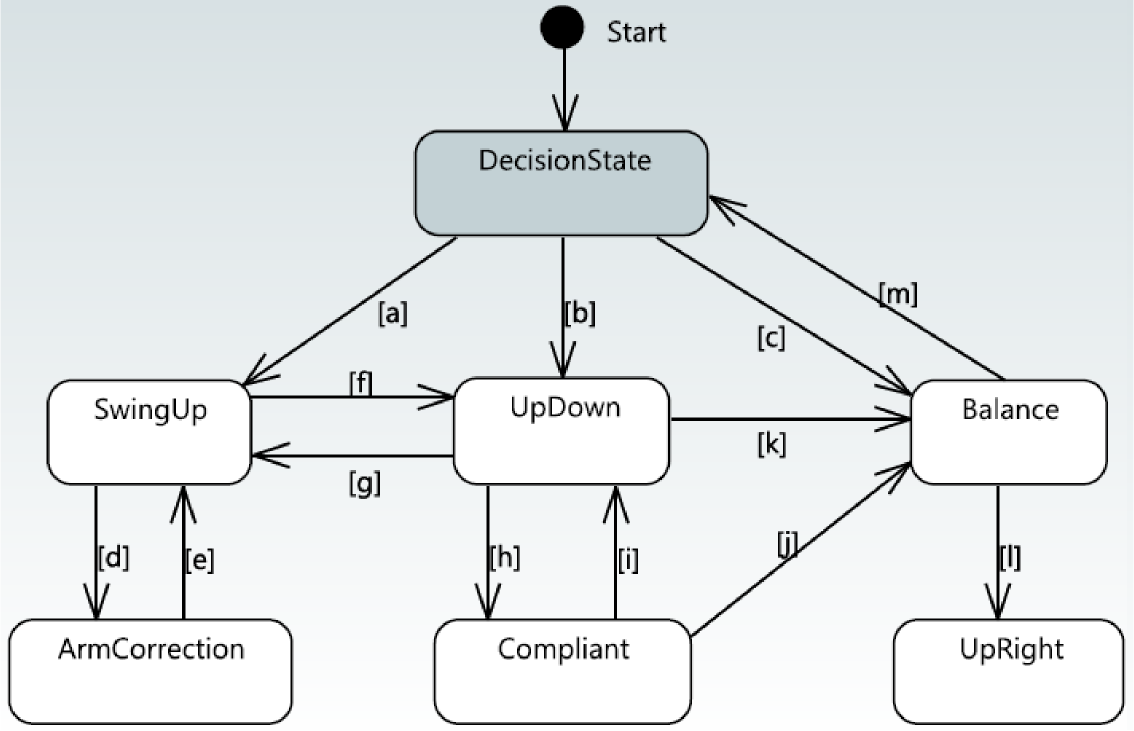


Figure 4.3: Controller state machine diagram

Chapter 5: Simulation

In this chapter, the details about the simulation of the reference design are discussed. A 3D CAD model of the reference design was made using SolidWorks (see Figure 5.1), which was then exported to Simulink using the SimMechanics Link tool. The mass properties of the design such as mass of each link, the inertia tensors and the distance to the COMs, etc., were recorded and substituted in the nonlinear system dynamics to obtain the energy equation. The nonlinear dynamics was then linearized about desired operating points and the optimal control gains corresponding to those operating points were obtained using MATLAB. The mass properties of the reference design as recorded from SolidWorks is shown in Table 5.1. In SolidWorks, the inertia tensor is displayed with respect to the global coordinate system. Similarity transformations were required to transform the inertia tensors into their local coordinate frames. The following similarity transformations were performed to obtain the inertia tensors about the link's COM:

$$I1 = R^T_{x,-90^\circ} I_1 R_{x,-90^\circ}$$

$$I2 = R^T_{x,-90^\circ} R^T_{y,-90^\circ} I_2 R_{y,-90^\circ} R_{x,-90^\circ}$$

$$I3 = R^T_{x,-90^\circ} R^T_{y,-90^\circ} I_3 R_{y,-90^\circ} R_{x,-90^\circ}$$

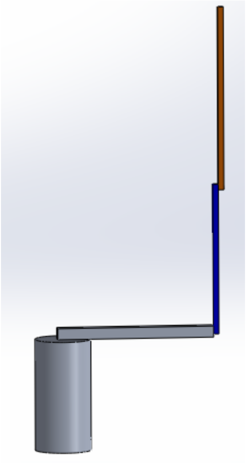


Figure 5.1: SolidWorks model of the reference design

where I_i is the the inertia tensor of the i^{th} link about the SolidWorks' global coordinate frame, as shown in Table 5.1; R is the appropriate rotation matrix, and $I1, I2, I3$ are the inertia tensors about the corresponding links' COM. The following simulations were performed using 'ode45', variable time-step settings in Simulink. The simulation generates on the order of 6×10^5 data points per second on an average.

5.1 Reference Design Simulation ($\beta_p = 0.45$)

Figure 5.2 shows the system states as a function of time. The graphs of X1, X2 and X3 correspond to θ_1, θ_2 and θ_3, and the graphs of X4, X5 and X6 correspond to $\dot{\theta}_1$, $\dot{\theta}_2$ and $\dot{\theta}_3$ respectively. It can be observed that at the beginning of the graph, $\dot{\theta}_2$ increases and θ_2 decreases, indicating the first segment's swing-up. At about $t = 0.5$ second, θ_2 and $\dot{\theta}_2$ converge to zero, indicating the transition to the Up-Down state. At this point, θ_3 starts to oscillate about its downward position. At about

	Link 1	**Link 2**	**Link 3**
Material	6061 Aluminum Alloy	6061 Aluminum Alloy	6061 Aluminum Alloy
Density (kg/cm^3)	0.0027	0.0027	0.0027
Length (cm)	14	13.5	16.5
Mass (kg)	0.0378	0.0093	0.0110
Inertia (kg-cm^2)	$\begin{bmatrix} 0.62 & 0 & 0 \\ 0 & 0.62 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.15 & 0 & 0 \\ 0 & 0.15 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Table 5.1: Reference design mass properties ($\beta_p = 0.45$)

$t = 1$ second, θ_3 and $\dot{\theta}_3$ converge to zero, indicating the transition to the balancing control. At around $t = 1.5$ seconds, all the states converge to zero, and the total pendulum energy converges to the reference energy – indicating a successful swing-up and balancing control. The state transitions can be seen in Figure 5.3. The controller states in this figure indicate the Controller ID in Table 4.4. It is observed that the first pendulum segment is swung up in a single swing of the cart. This is indicated by a steady increase in θ_2 from $-\pi$ to 0 in Figure 5.2. It is also indicated by a direct transition of the controller state from ‘1’ to ‘2’ without any intermediate controller states (see Figure 5.3). The torque input to the motor saturates during the Up-Down and Balancing configurations. This is due to the large weights given to the LQR parameters, which results in a large value for the input torque. The minimum swing-up constant in order to swing up the first pendulum segment in a single swing was $k = 7$.

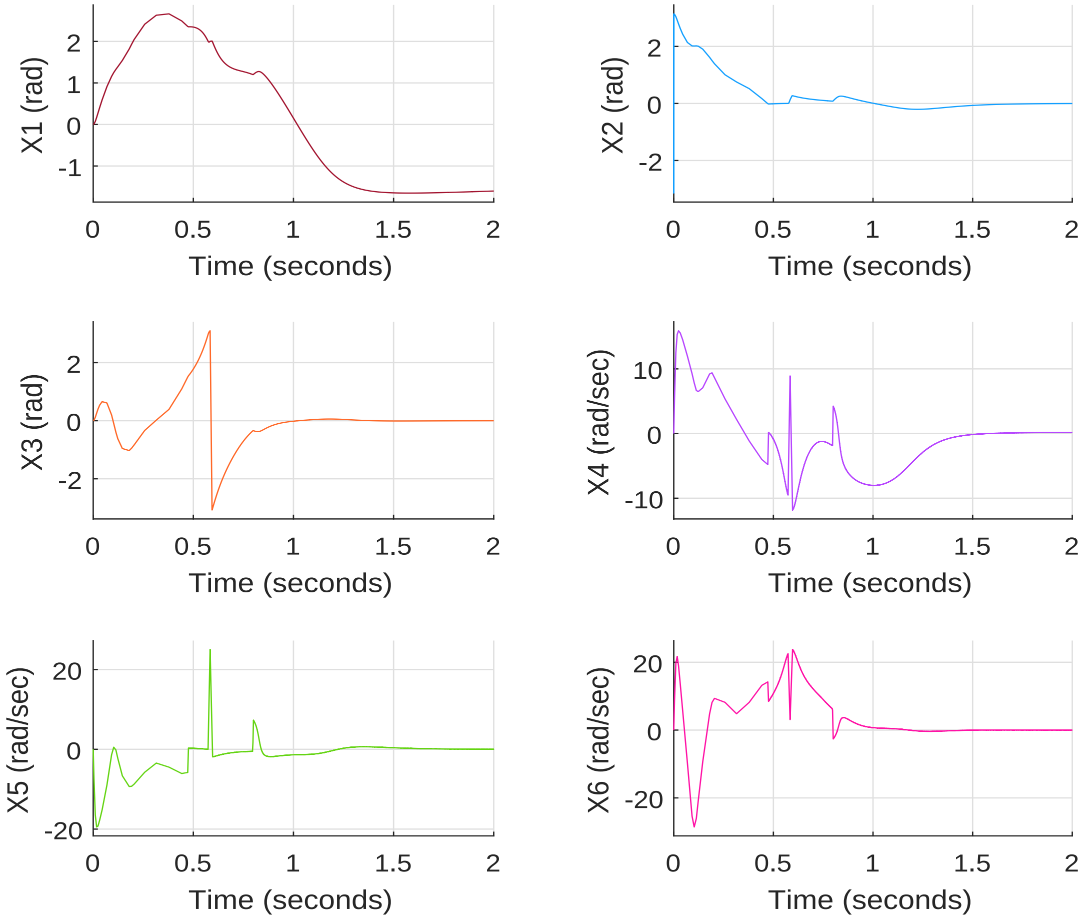


Figure 5.2: System States ($\beta_p = 0.45$)

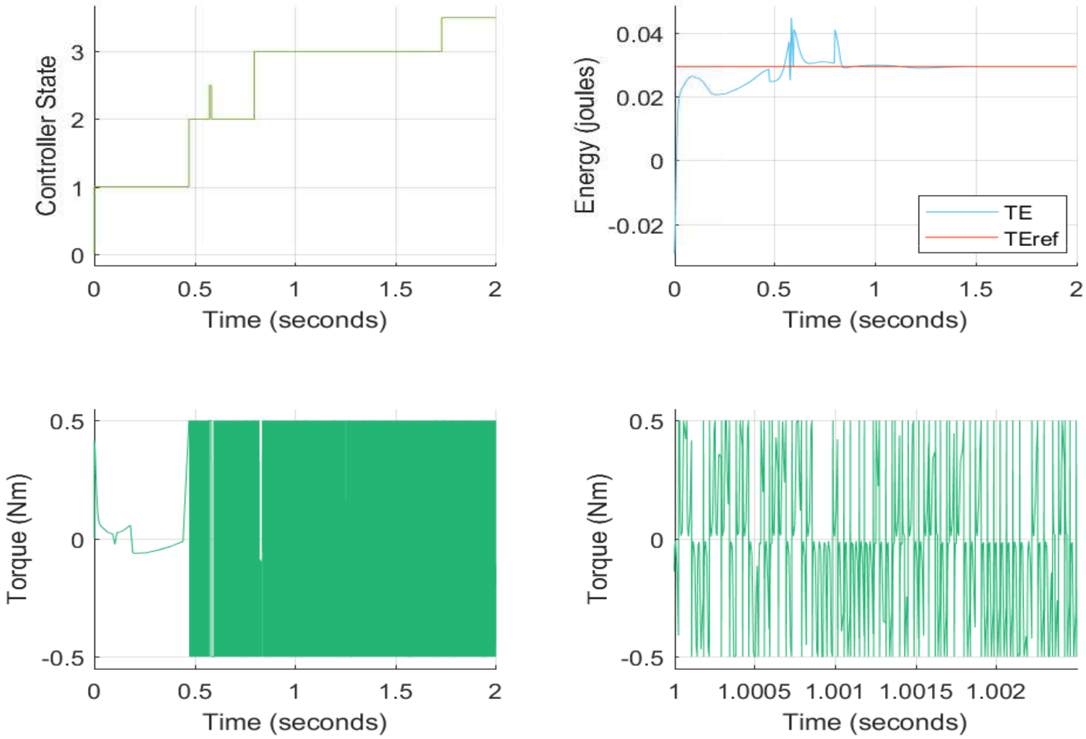


Figure 5.3: Control State, Total Energy and Motor Torque ($\beta_p = 0.45$)

	Link 1	**Link 2**	**Link 3**
Material	6061 Aluminum Alloy	6061 Aluminum Alloy	6061 Aluminum Alloy
Density (kg/cm^3)	0.0027	0.0027	0.0027
Length (cm)	14	12	18
Mass (kg)	0.0378	0.0083	0.0120
Inertia (kg-cm^2)	$\begin{bmatrix} 0.62 & 0 & 0 \\ 0 & 0.62 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.32 & 0 & 0 \\ 0 & 0.32 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Table 5.2: Alternate design mass properties ($\beta_p = 0.40$)

5.2 Alternate Design Simulation ($\beta_p = 0.40$)

The results for this design are not very different from that of $\beta_p = 0.45$. The first pendulum segment is again swung up with only one swing. The minimum swing-up constant in this case was $k = 11$. The motor torque starts to saturate after the swing-up, at about $t = 0.5$ seconds. The balancing is complete at about $t = 1.75$ seconds. The results for this simulation are shown in Figure 5.4 and Figure 5.5.

5.3 Alternate Design Simulation ($\beta_p = 0.35$)

In this design, the first pendulum segment could not be swung up in a single swing, irrespective of the magnitude of the swing-up constant. This is due to the dynamic effect of the second segment on the first segment. The minimum swing-up constant that succeeded in swinging up the first segment after multiple swings was

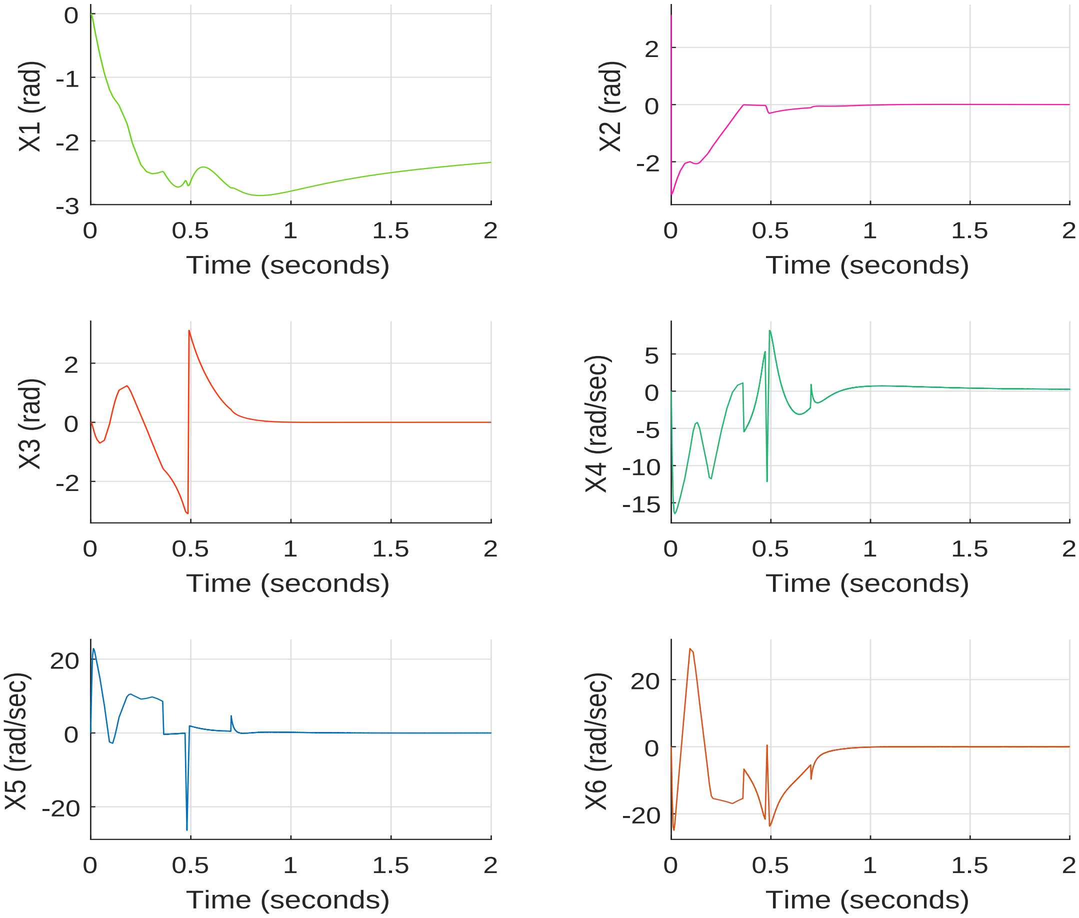


Figure 5.4: System States ($\beta_p = 0.40$)

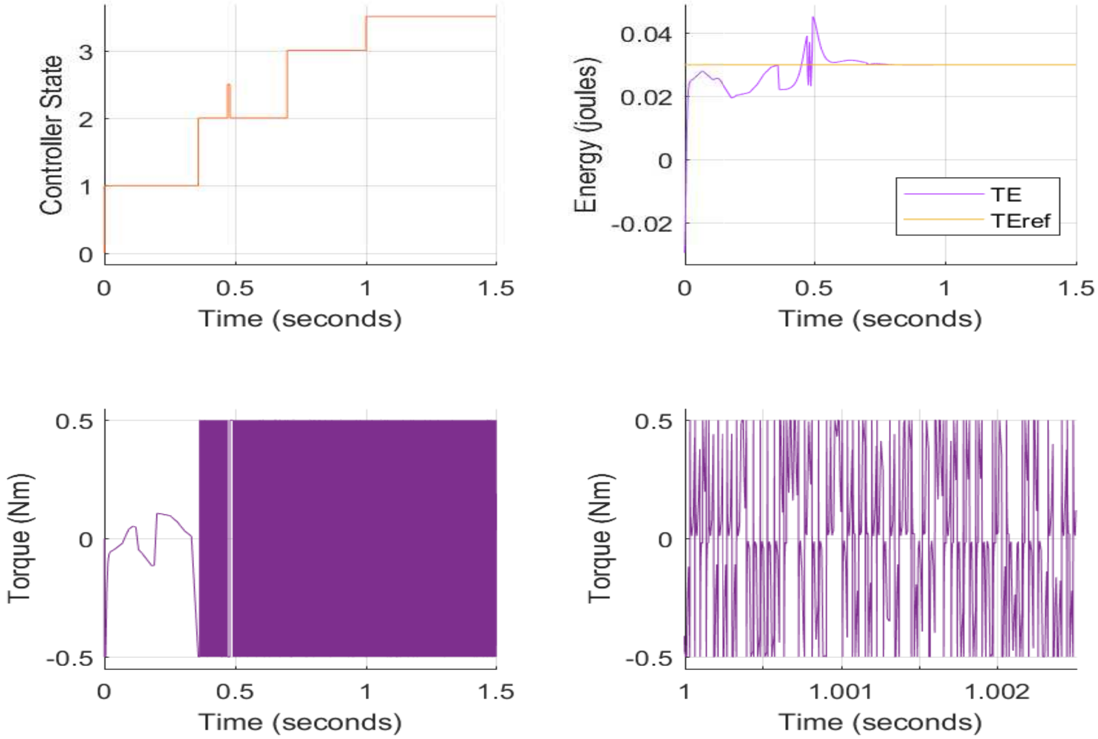


Figure 5.5: Control State, Total Energy and Motor Torque ($\beta_p = 0.40$)

	Link 1	**Link 2**	**Link 3**
Material	6061 Aluminum Alloy	6061 Aluminum Alloy	6061 Aluminum Alloy
Density (kg/cm^3)	0.0027	0.0027	0.0027
Length (cm)	14	10.5	19.5
Mass (kg)	0.0378	0.0073	0.0131
Inertia (kg-cm^2)	$\begin{bmatrix} 0.62 & 0 & 0 \\ 0 & 0.62 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.07 & 0 & 0 \\ 0 & 0.07 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.4 & 0 & 0 \\ 0 & 0.4 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Table 5.3: Alternate design mass properties ($\beta_p = 0.35$)

$k = 17$. It can be seen in Figure 5.6 that θ_2 increases from $-\pi$ to about 0.5 radians, quickly drops to $-\pi$ radians and then oscillates a couple of times before reaching steady state in the upright configuration. Figure 5.7 indicates that the first two attempts at balancing the first segment upright were unsuccessful. The controller then transitions to state '3' (balancing control) after a fraction of a second in state '2' (Up-Down control) at around $t = 2.5$ seconds. It is again observed that the torque saturates during the balancing control.

5.4 Alternate Design Simulation ($\beta_p = 0.30$)

Similar to the previous design, the first pendulum segment could not be swung up in a single swing, irrespective of the magnitude of the swing-up constant. The minimum swing-up constant that succeeded in swinging up the first segment after multiple swings was $k = 8$. It can be observed in Figure 5.8 that the first segment oscillates about the downward position a couple of times before reaching the Up-

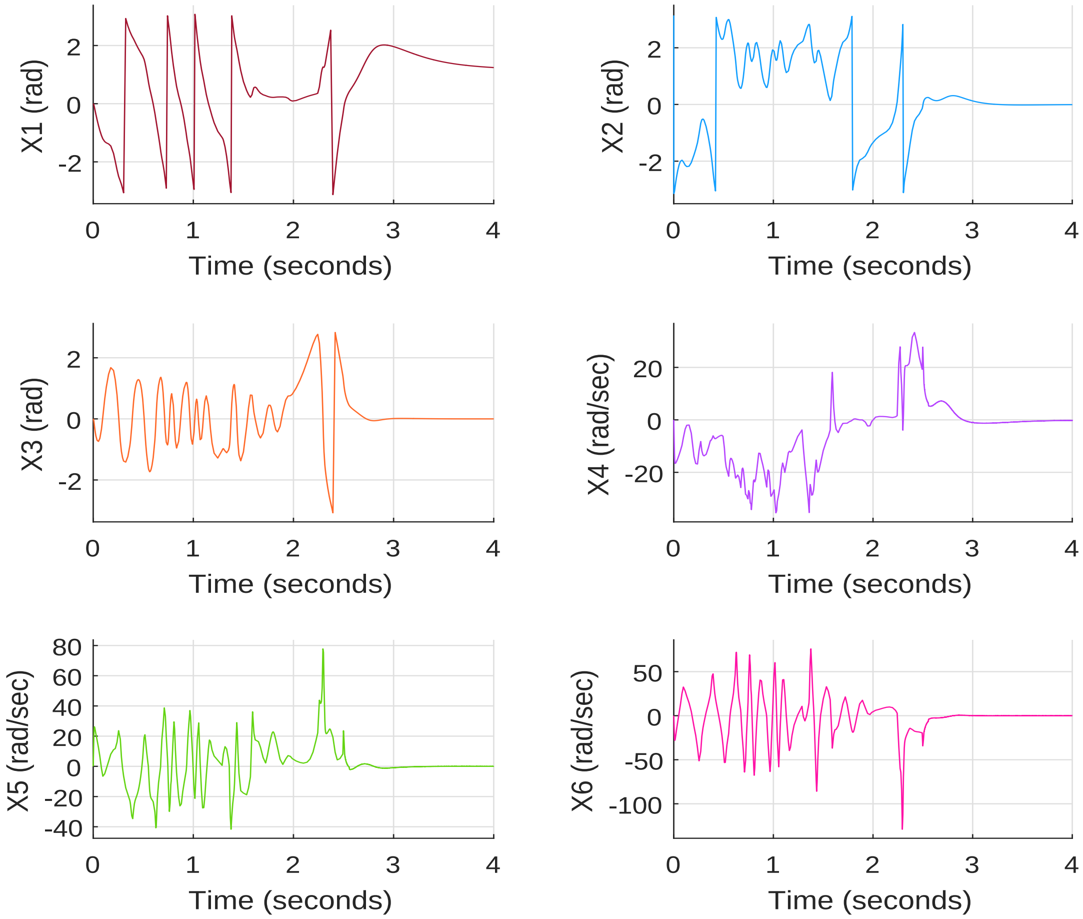


Figure 5.6: System States ($\beta_p = 0.35$)

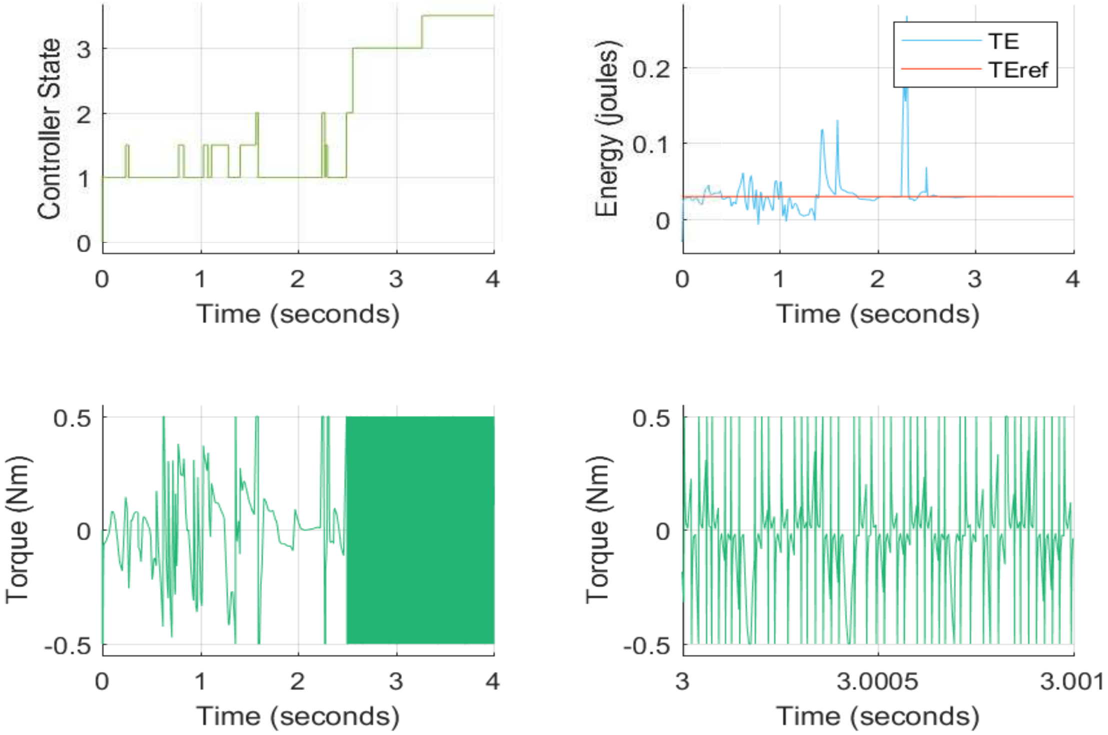


Figure 5.7: Control State, Total Energy and Motor Torque ($\beta_p = 0.35$)

	Link 1	**Link 2**	**Link 3**
Material	6061 Aluminum Alloy	6061 Aluminum Alloy	6061 Aluminum Alloy
Density (kg/cm^3)	0.0027	0.0027	0.0027
Length (cm)	14	9	21
Mass (kg)	0.0378	0.0063	0.0141
Inertia (kg-cm^2)	$\begin{bmatrix} 0.62 & 0 & 0 \\ 0 & 0.62 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.04 & 0 & 0 \\ 0 & 0.04 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Table 5.4: Alternate design mass properties ($\beta_p = 0.30$)

Down state. The system reaches the Up-Down state at around $t = 0.8$ seconds, and the balancing control state at around $t = 1.2$ seconds. Again, it is observed that the torque saturates in the Up-Down and balancing control states due to the effect of the second segment.

5.5 Prototype Design Simulation ($\beta_p = 0.45$)

The prototype design was simulated using a fixed step-size solver, ‘ode3’, in Simulink, with a step size of 0.01 seconds, in order to emulate the real system, in which the sensor provides data at an approximate frequency of 100 Hz. The simulation shows that the prototype design was successfully swung-up in a single swing, and balanced upright. Figure 5.11 shows the system states as a function of time, and Figure 5.12 shows the controller states, total energy of the pendulum segments and the motor torque at each time step.

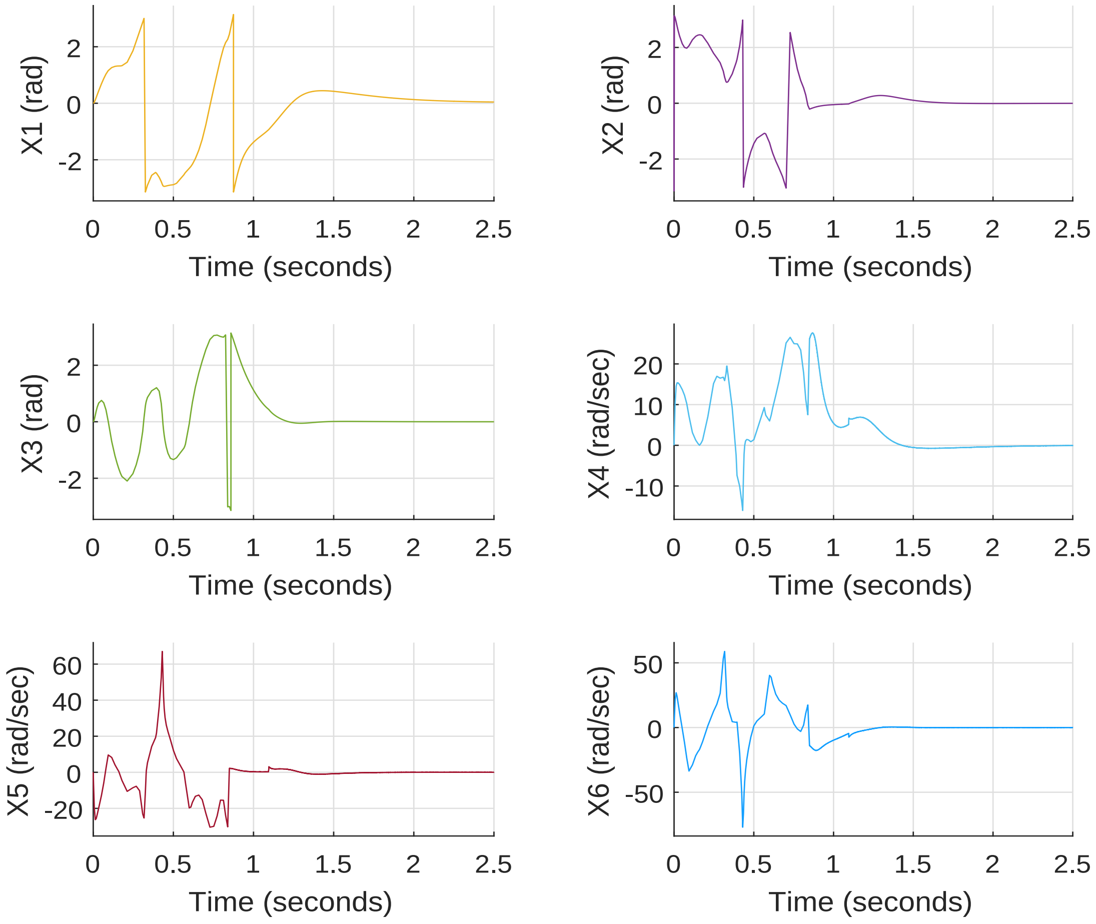


Figure 5.8: System States ($\beta_p = 0.30$)

	Link 1	**Link 2**	**Link 3**
Material	6061 Aluminum Alloy	6061 Aluminum Alloy	6061 Aluminum Alloy
Density (kg/cm^3)	0.0027	0.0027	0.0027
Length (cm)	14	14	16
Mass (kg)	0.201	0.060	0.044
Inertia (kg-cm^2)	$\begin{bmatrix} 8.7 & 0 & 0 \\ 0 & 3.37 & 2.46 \\ 0 & 2.46 & 5.53 \end{bmatrix}$	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0.2 \\ 0 & 0.2 & 0.1 \end{bmatrix}$	$\begin{bmatrix} 1.35 & 0 & 0 \\ 0 & 1.26 & 0.2 \\ 0 & 0.2 & 0.1 \end{bmatrix}$

Table 5.5: Prototype design mass properties

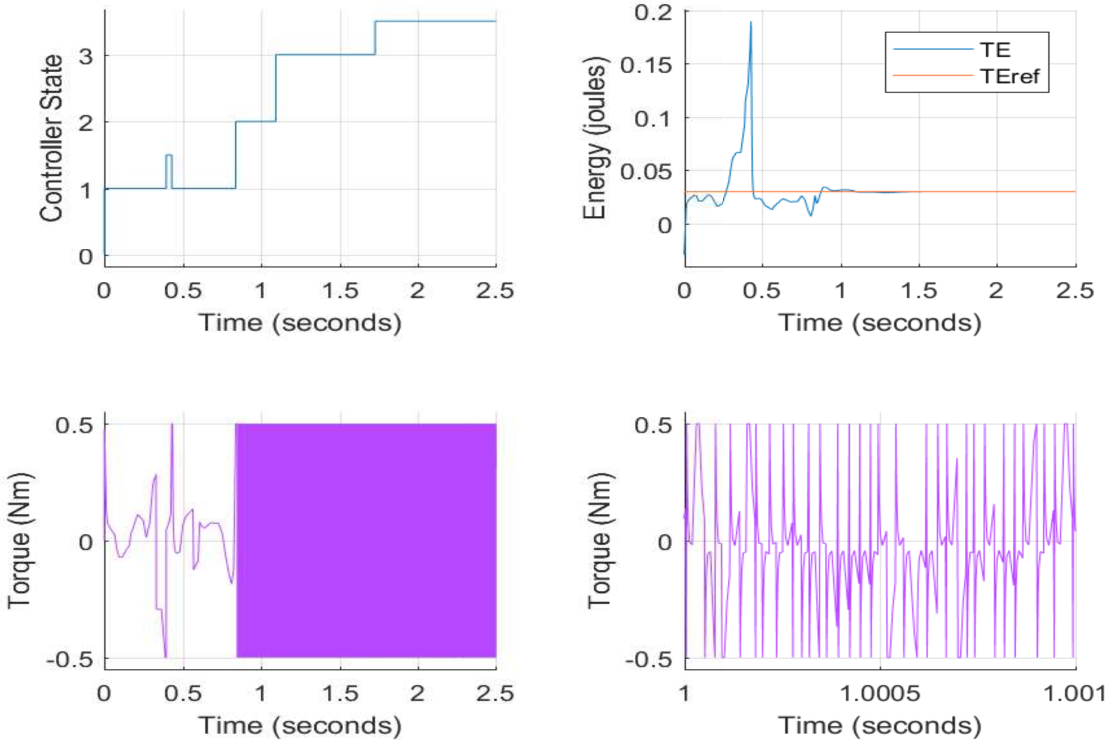


Figure 5.9: Control State, Total Energy and Motor Torque ($\beta_p = 0.30$)

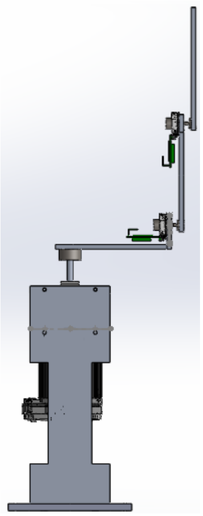


Figure 5.10: SolidWorks model of the prototype design

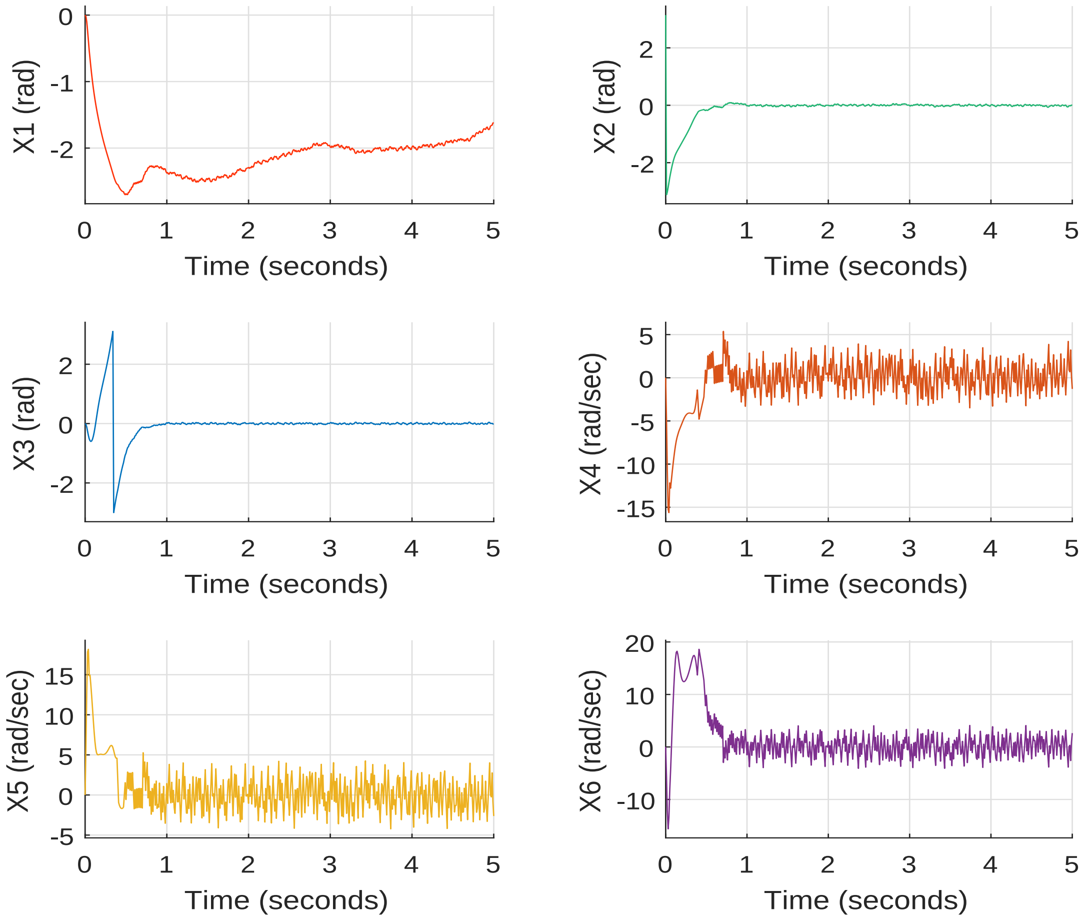


Figure 5.11: System States (prototype)

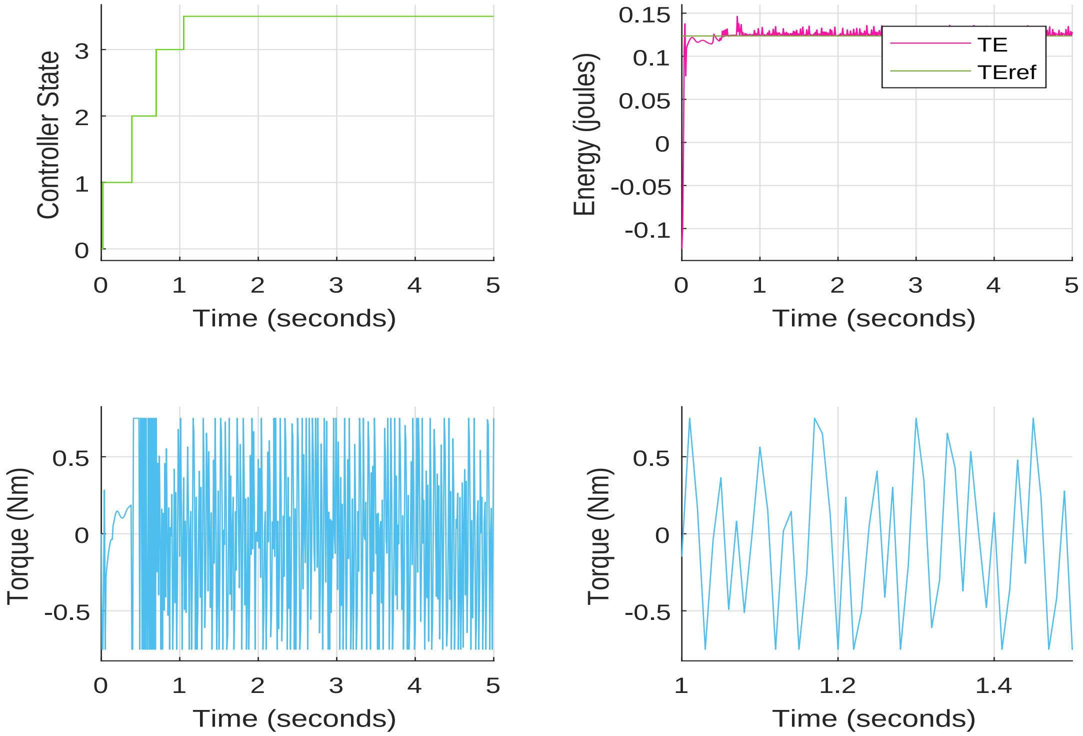


Figure 5.12: Control State, Total Energy and Motor Torque (prototype)

β_p	k	No. of Swings
0.45	7	Single-swing
0.40	11	Single-swing
0.35	17	Multi-swing
0.30	8	Multi-swing

Table 5.6: Swing-up constant for different values of β_p

5.6 Analysis

It is intuitive that decreasing the relative length of the first pendulum segment makes the swing-up more challenging due to the dynamic effect of the second segment, caused by its larger moment of inertia relative to the first segment. In this study, the effect of the change in relative pendulum lengths (changes in β_p) on a swing-up parameter – the swing-up constant, k – was analyzed. It is observed that the first segment can be swung up in a single swing only till a certain limit of β_p. In this case, designs with $\beta_p < 0.4$ could not be swung up in a single swing. The rest of the control parameters were the same across all the designs. Figure 5.13 shows the variation in k for different values of β_p. The markers in red indicate designs that needed multiple swings. The swing-up constant, k, seemingly increases with decrease in β_p until a threshold value (in this case 0.4), beyond which k becomes arbitrary. Based on the simulation results, the reference design with $\beta_p = 0.45$ performed the best in terms of least chattering in the control input.

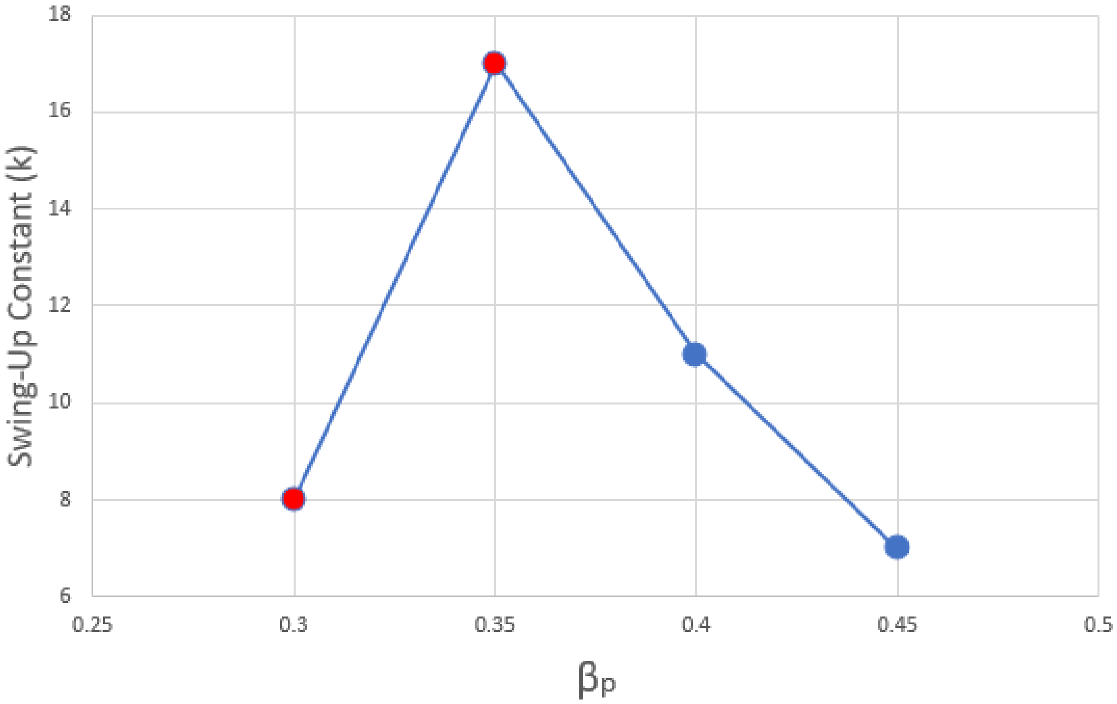


Figure 5.13: Swing-up constant for different values of β_p

Chapter 6: Prototype Design

This prototype of the double Furuta pendulum system was developed for the validation of the control strategies that are discussed in this thesis. The prototype consists of an electro-mechanical subsystem, a sensor subsystem and a software subsystem. The sensor subsystem communicates with the software subsystem, which in turn communicates with the electro-mechanical subsystem. The specifics of each subsystem, and their communications are detailed in this chapter.

6.1 Electro-Mechanical Subsystem

The mechanical design of the prototype was based on the reference design shown in Section 5.1, with $\beta_p = 0.45$. The first step was to select a motor that was powerful enough for this application. The motor requirements were derived by surveying the graphs of the motor torque and the angular velocity of the cart across all the simulations discussed in Chapter 5. The maximum torque utilized was observed to be 0.5 Nm and the maximum angular velocity achieved by the cart was observed to be 35 rad/sec (335 RPM). Considering a factor of safety (FOS) of 2, the primary actuation requirements were derived as:

1. The motor shall have a rated torque ≥ 1 Nm

2. The motor shall have a rated speed $\geq$ 670 RPM

The secondary requirements were:

1. The motor drive shall have a torque control mode
2. The motor drive shall be controllable using readily-available microcontrollers
3. The motor and drive shall be cost effective

The Teknic ClearPath MCVC 3432P-RLN brushless DC motor satisfied the above requirements [16], and was hence chosen as the actuator. The specifications of the motor are:

1. Rated torque: 1.5 Nm
2. Rated speed at 24V: 740 RPM
3. Peak torque: 4.9 Nm
4. Operating voltage: 24V - 75V DC

The design of the mechanical parts was done using SolidWorks. The drawings for the machined parts along with assembly notes are shown in Appendix C. The custom designed parts along with commercial off-the-shelf (COTS) components were put together to realize the electro-mechanical subsystem. The bill of materials for the COTS components are shown in Table E.1.

6.1.1 Joint Design

It is important to consider practicalities while doing CAD and selecting appropriate components. Ideally while doing CAD, there is no clearance between a shaft and its bearing. However, in reality there is always a small clearance between a shaft and its bearing, which results in some amount of play in the link attached to the shaft. This has to be accounted for in the design to avoid collision between the two pendulum segments. The amount of play in the link is directly proportional to the link length and the clearance between the shaft and its bearing, and is inversely proportional to the width of the bearing, i.e.,

$$x = l\frac{c}{w} \tag{6.1}$$

where x is the play in the link, l is the link length, c is the clearance between the shaft and the bearing, and w is the width of the bearing. This is illustrated in Figure 6.1. It is thus important to chose a bearing that can not only bear the dynamic loads, but that also has sufficient width to limit the play in the link. The play can also be limited by reducing the clearance between the shaft and the bearing by choosing an appropriate tolerance value for the machining of the shaft. In the prototype, the clearance, $c = 0.01$mm, and the bearing width, $w = 6$mm. A needle roller bearing was used as opposed to a ball bearing because a ball bearing has angular play in the inner ring due to a point contact between the ball and the ball-race. Also, for a given shaft diameter, a needle roller bearing has more width than that of a ball bearing, thus reducing the play in the link.

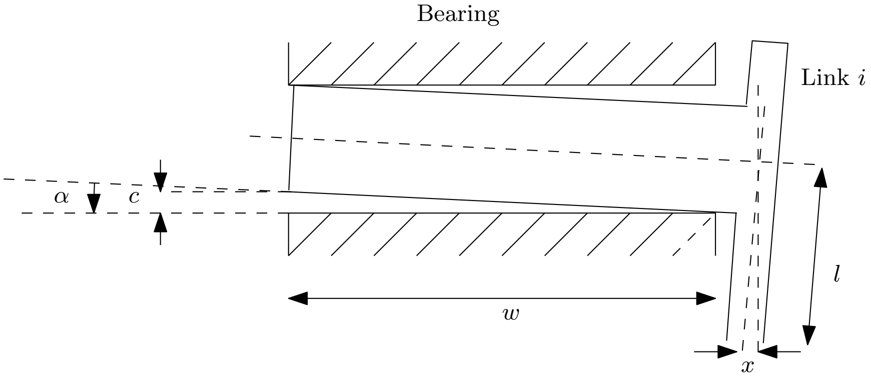


Figure 6.1: Joint design

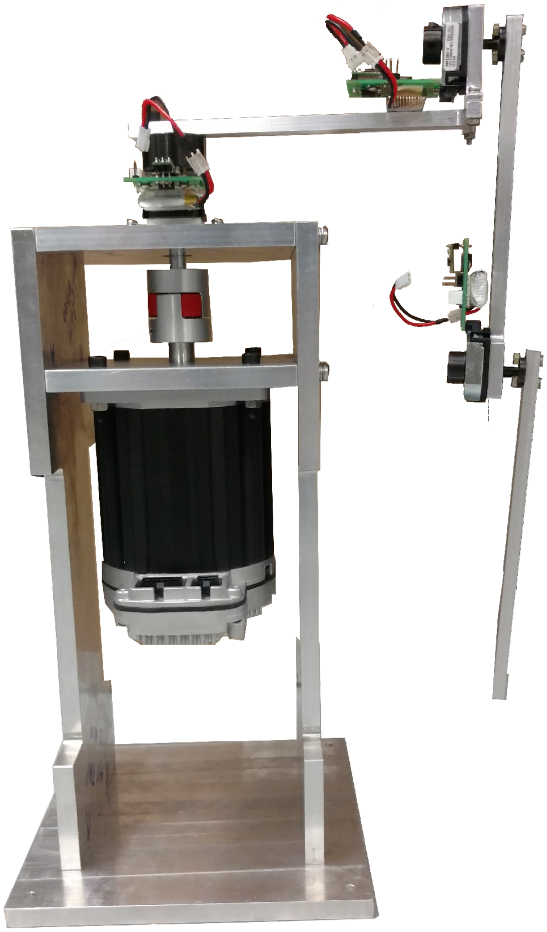


Figure 6.2: Prototype

i	α	a	d	θ
1	$-\pi/2$	0	0	θ_1
2	0	0	$d_2 = 0.14$m	0
3	0	$a_2 = 0.14$m	$d_2{}' = 0.0165$m	$\theta_2 - \pi/2$
4	0	$a_3 = 0.16$m	$d_3 = 0.0165$m	θ_3

Table 6.1: Kinematic parameters for link ends of the prototype

i	α	a	d	θ
1	$-\pi/2$	0	0	θ_1
2	0	0	$d_{c2} = 0.029$m	0
3	0	$a_{c2} = 0.07$m	$d_{c2}{}' = 0.004$m	$\theta_2 - \pi/2$
4	0	$a_{c3} = 0.05$m	$d_{c3} = 0.004$m	θ_3

Table 6.2: Kinematic parameters for link COMs of the prototype

6.2 Sensor Subsystem

The sensor subsystem mainly consists of an absolute rotary encoder and a wireless module which transmits the encoder data to the main controller. The wireless module is used in order to facilitate free rotation of the mechanical links without wires coming in the way. The bill of materials for the sensor subsystem is shown in Table E.2.

The encoder operates between 4.5V - 5.5V [17], whereas the wireless transmitter operates between 1.7V - 3.3V. A single on-board 3.7V battery is used to power both the encoder and the wireless transmitter. This is done by providing a 3.3V linear voltage regulator between the battery and the wireless transmitter, while a step-up regulator is used to boost the voltage to 5V to power the encoder. Bidirectional logic level converter circuits [18] are used for the two-way conversion (5V $\leftrightarrow$ 3.3V) of the Serial Peripheral Interface (SPI) communication signals between the encoder and the wireless transmitter. The Wi-Fi module, ESP8266 12E, is a RISC microcontroller by itself, and uses the Universal Asynchronous Receiver-Transmitter (UART) protocol to communicate with a host PC. Arduino programs can be flashed onto the ESP8266 12E through a host PC.

A custom printed circuit board (PCB) was designed to make the sensor subsystem compact. The architecture of the sensor subsystem is shown in Figure 6.4. The schematic of the PCB is shown in Figure 6.5, and the corresponding board design in Figure 6.6. The realized board design is shown in Figure 6.7.

Each encoder was connected to a wireless module (custom PCB) to transmit the sensor data to the microcontroller over the Wi-Fi network. An Arduino program was written to obtain the raw data from the encoder, process the data to obtain the angular position in radians, differentiate the angular position to obtain angular velocity and finally transmit the angular position and angular velocity as a string of characters over the Wi-Fi network. The network diagram is shown in Figure 6.3. The wireless receiver is a similar wireless module with a ESP8266 12E microcontroller, which receives the angular position and angular velocity from each of the wireless modules at approximately 100 Hz. Each wireless module has a fixed IP address assigned to it, which is used by the receiver to determine from which module the data is arriving. In every loop, the receiver concatenates the sensor data it receives from the three wireless modules into a comma-delimited string that represents the system's state vector $\mathbf{x}$. This comma-delimited string is then sent to the microcontroller using the UART protocol at 115200 bauds per second. The source code for the wireless module is shown in Section D.1 and that for the receiver is shown in Section D.2 of Appendix D.

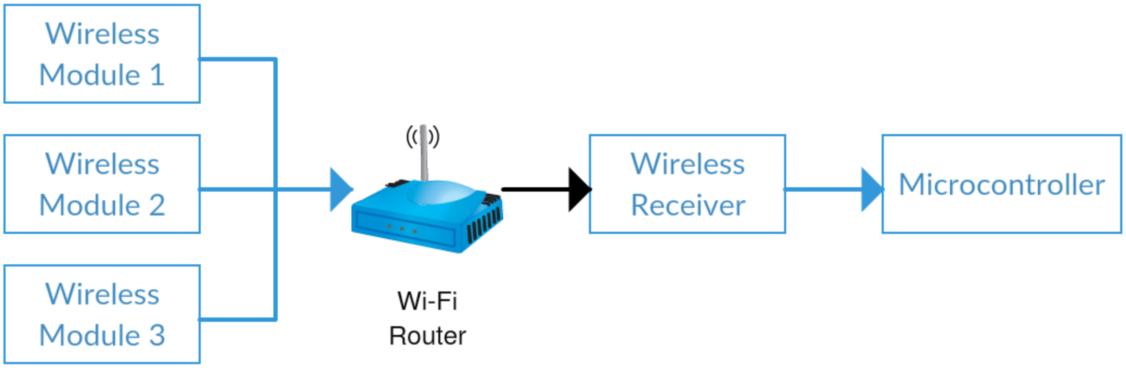


Figure 6.3: Sensor subsystem network diagram

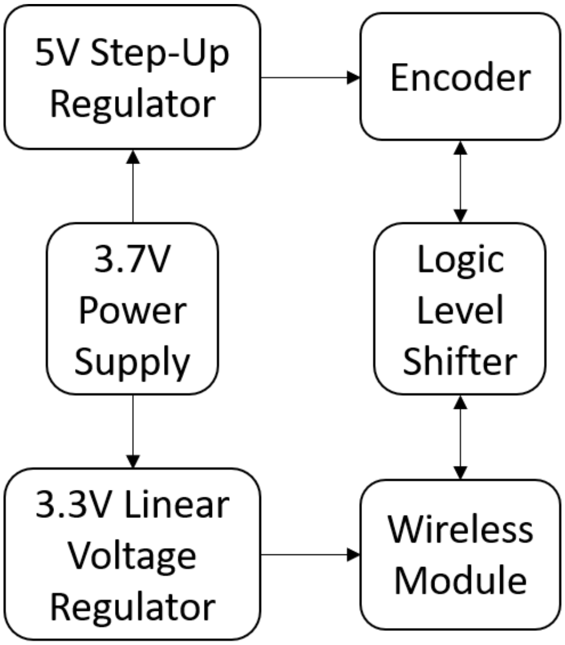


Figure 6.4: Sensor subsystem architecture

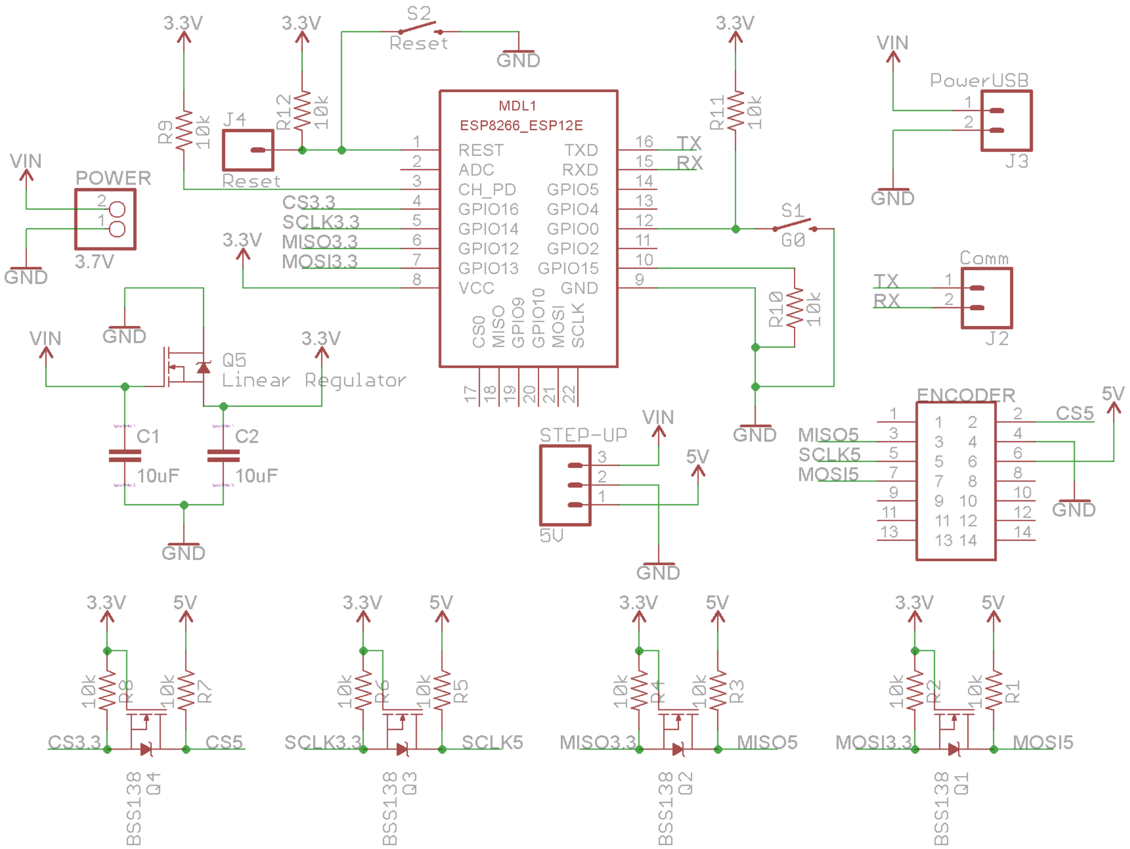


Figure 6.5: Wireless module PCB schematic

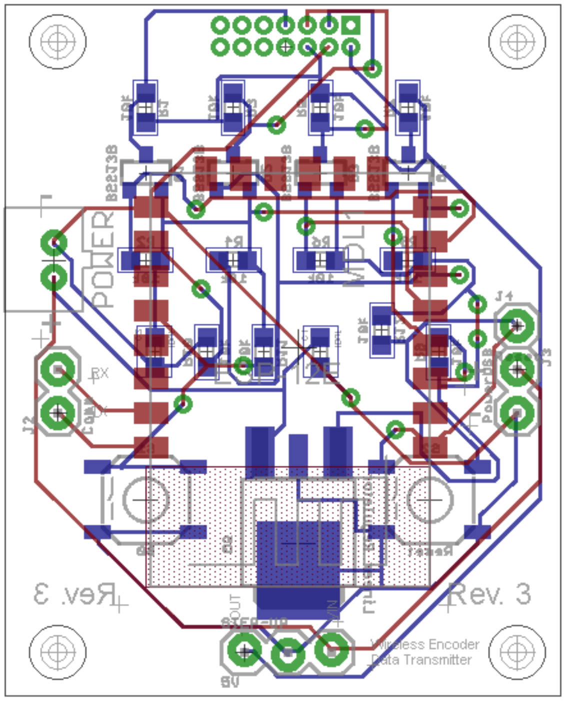


Figure 6.6: Wireless module board design

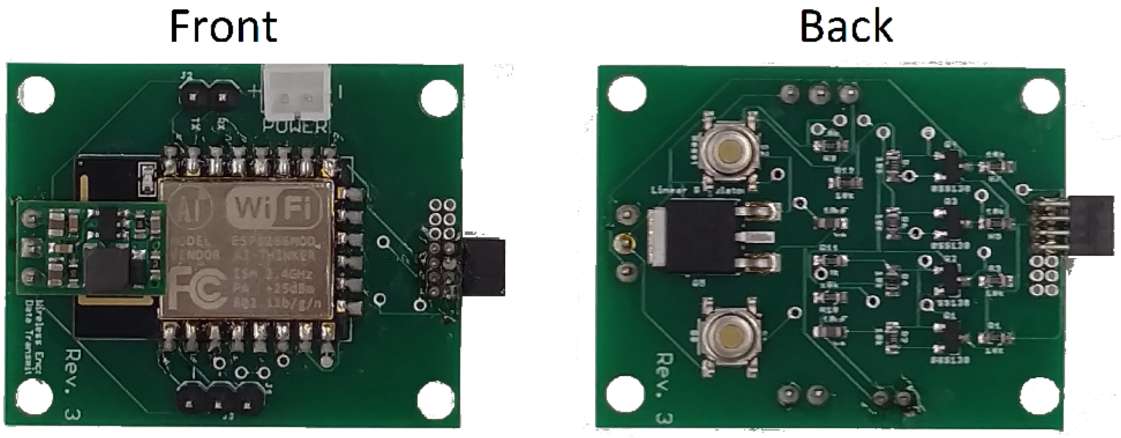


Figure 6.7: Realized wireless module

6.3 Software Subsystem

The software subsystem consists of the elements that receive the sensor data, process the data and provide an appropriate control input to the electro-mechanical subsystem.

At the heart of the software subsystem is the LAUNCHXL-F28379D LaunchPad from Texas Instruments, which has the F28379D 200 MHz dual-core microcontroller along with a host of additional peripheral components. In every loop, the microcontroller receives the sensor data from the wireless receiver through the UART protocol, determines the control law to be applied, and then generates a PWM signal with a duty cycle that is proportional to the computed control input. The PWM signal is sent to the motor drive, which then applies a torque that is proportional to the PWM duty cycle. In essence, the microcontroller performs exactly the same function as that of the 'Control State Machine' and 'Controller' blocks in the Simulink simulation as shown in Figure 4.2.

The LaunchPad was programmed using C [19] [20], in the Code Composer Studio v18 environment. A high-level flowchart (activity diagram) of the software is shown in Figure 6.8. External interrupts using tactile switches were used to provide emergency-stop functionality for the motor, disable the PWM signal and to reset the controller state. The source code for the microcontroller program is shown in Section D.3 of Appendix D.

The 'doublefuruta.c' program contains the `main()` function, and is responsible for configuring the registers corresponding to the PWM, UART and the exter-

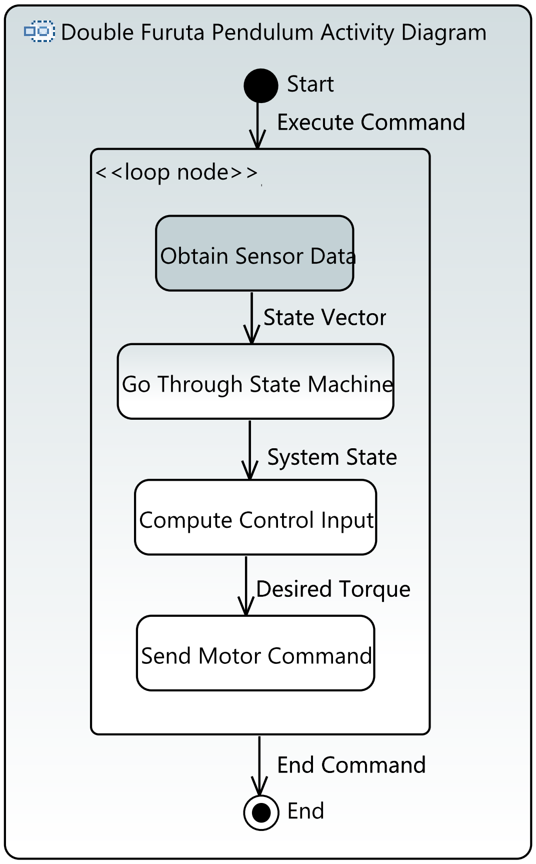


Figure 6.8: Software flowchart

nal interrupts. The program runs an infinite loop in which it receives the system state vector from the wireless receiver, and then calls the `action()` function, which is contained in the 'action.c' program. The 'action.c' program in turn calls the `statemachine()` function contained in the 'statemachine.c' program, which returns the controller state. Depending on the controller state, the `action()` function computes the appropriate control input in that iteration, and then sets the PWM duty cycle to a value that is proportional to the computed control input.

Chapter 7: Conclusion and Future Work

In this thesis, a detailed mathematical model in terms of dimensionless parameters of a double Furuta pendulum was developed. The mathematical model was used to design control laws for the swing-up and balancing of the two pendulum segments. A combination of energy control and LQR technique was used for the swing-up, and a separate LQR was designed for the balancing control. A state machine model was developed that chose the appropriate control law at different system states.

The effectiveness of the overall control system was verified by performing simulations over different relative pendulum lengths. The simulation results were recorded and analyzed. Finally, the details of the hardware design for the validation of the control system were discussed.

Future work would include improvements in the mechanical design and the sensor subsystem. The pendulum joints could be made more sturdy by using a tighter fit between the shaft and the bearing, and using a bearing with more width. The sensor subsystem could be improved by using faster communication protocols between the wireless modules and the wireless receiver, and between the wireless receiver and the microcontroller.

Appendix A: Model Verification

In order to verify that the kinematics model derived from an adaptation of the D-H Parameters was correct, the model was compared with another kinematic model that was derived from elementary transformations, which included transformations in the y-axis. Built-in functions from Peter Corke's Robotics Toolbox for MATLAB [21] was used to perform these elementary transformations.

A custom function named `dhtrans()` was defined in MATLAB that performed the serial transformations according to the D-H convention. The resulting final transformation matrices from both methods were subtracted to find the difference in each matrix element. It was observed that the difference was zero, meaning that the resulting matrices were identical.

A.1 Function dhtrans()

```
function T = dhtrans(alpha,d,a,theta)
T = trotz(theta)*transl(0,0,d)*transl(a,0,0)*trotx(alpha);
end
```

A.2 Program verify_dh.m

```
syms theta1 theta2 theta3;
syms a2 a3 d2 d3;
syms ac2 ac3 dc2 dc3;
```

Transformations NOT using D-H convention

```
T1 = trotz(theta1)*transl(0,d2,0)*trotx(-sym(pi)/2);
T2 = T1*trotz(theta2-(sym(pi)/2))*transl(a2,0,0);
T3 = T2*trotz(theta3)*transl(0,0,d3)*transl(a3,0,0);

Tc1 = trotz(theta1)*transl(0,dc2,0)*trotx(-sym(pi)/2);
Tc2 = T1*trotz(theta2-(sym(pi)/2))*transl(ac2,0,0);
Tc3 = T2*trotz(theta3)*transl(0,0,dc3)*transl(ac3,0,0);
```

Transformations using adapted D-H convention

```
T1_dh = dhtrans(-sym(pi)/2,0,0,theta1)*dhtrans(0,d2,0,0);
T2_dh = T1_dh*dhtrans(0,0,a2,theta2-(sym(pi)/2));
T3_dh = T2_dh*dhtrans(0,d3,a3,theta3);

Tc1_dh = dhtrans(-sym(pi)/2,0,0,theta1)*dhtrans(0,dc2,0,0);
Tc2_dh = T1_dh*dhtrans(0,0,ac2,theta2-(sym(pi)/2));
Tc3_dh = T2_dh*dhtrans(0,dc3,ac3,theta3);
```

Compare and display results

```
Tc1_diff = simplify(Tc1_dh - Tc1);
Tc2_diff = simplify(Tc2_dh - Tc2);
Tc3_diff = simplify(Tc3_dh - Tc3);

display(Tc1_diff);
display(Tc2_diff);
display(Tc3_diff);
```

Output

```
Tc1_diff =

[ 0, 0, 0, 0]
[ 0, 0, 0, 0]
[ 0, 0, 0, 0]
[ 0, 0, 0, 0]


Tc2_diff =

[ 0, 0, 0, 0]
[ 0, 0, 0, 0]
[ 0, 0, 0, 0]
[ 0, 0, 0, 0]


Tc3_diff =

[ 0, 0, 0, 0]
[ 0, 0, 0, 0]
[ 0, 0, 0, 0]
[ 0, 0, 0, 0]
```

A.3 Jacobian Verification

To verify the correctness of the Jacobians, the total energy equation is considered. We know that the total energy equals:

$$TE = \frac{1}{2}\dot{q}^T \cdot D \cdot \dot{q} + P \tag{A.1}$$

where $\dot{q}$ is the joint velocity vector, D is the mass matrix and P is the potential energy.

The mass matrix, D contains the Jacobian terms. When expanded, this nonlinear energy equation is in terms of the system's state space, $\mathbf{x}$, and can hence be measured using Simulink.

We also know that the input energy to the system is:

$$TE = \int \tau\omega \tag{A.2}$$

where τ is the input torque to the motor, and ω is the angular velocity of the motor.

Assuming no loss of energy, the above two equations should provide the same graph when plotted over a certain time period, provided that the measured energy equation is correct. If the graphs are identical, it indicates that the energy equation, A.1, is correct, which in turn shows that the Jacobians are correct, because no intermediate simplifications or approximations are made.

The above test was conducted in Simulink using different patterns of torque input. The corresponding Simulink block diagram is shown in Figure A.3. In the block diagram, the “Measured Energy” block is a MATLAB Function block, which takes the system’s state vector as input, and outputs the total energy according to Equation A.1. The “Input Energy” and the “Measured Energy” signals were outputted to the workspace and then plotted. The plots are shown in Figure A.3. It can be observed that the “Measured Energy” graph follows the “Input Energy” graph. Thus, the nonlinear model is verified.

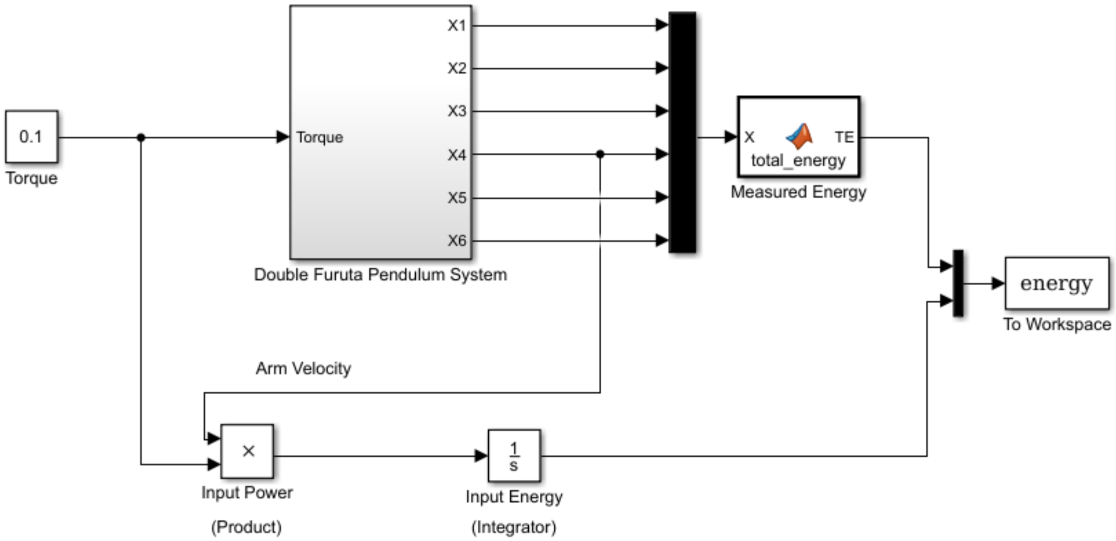


Figure A.1: Simulink simulation for testing energy equation

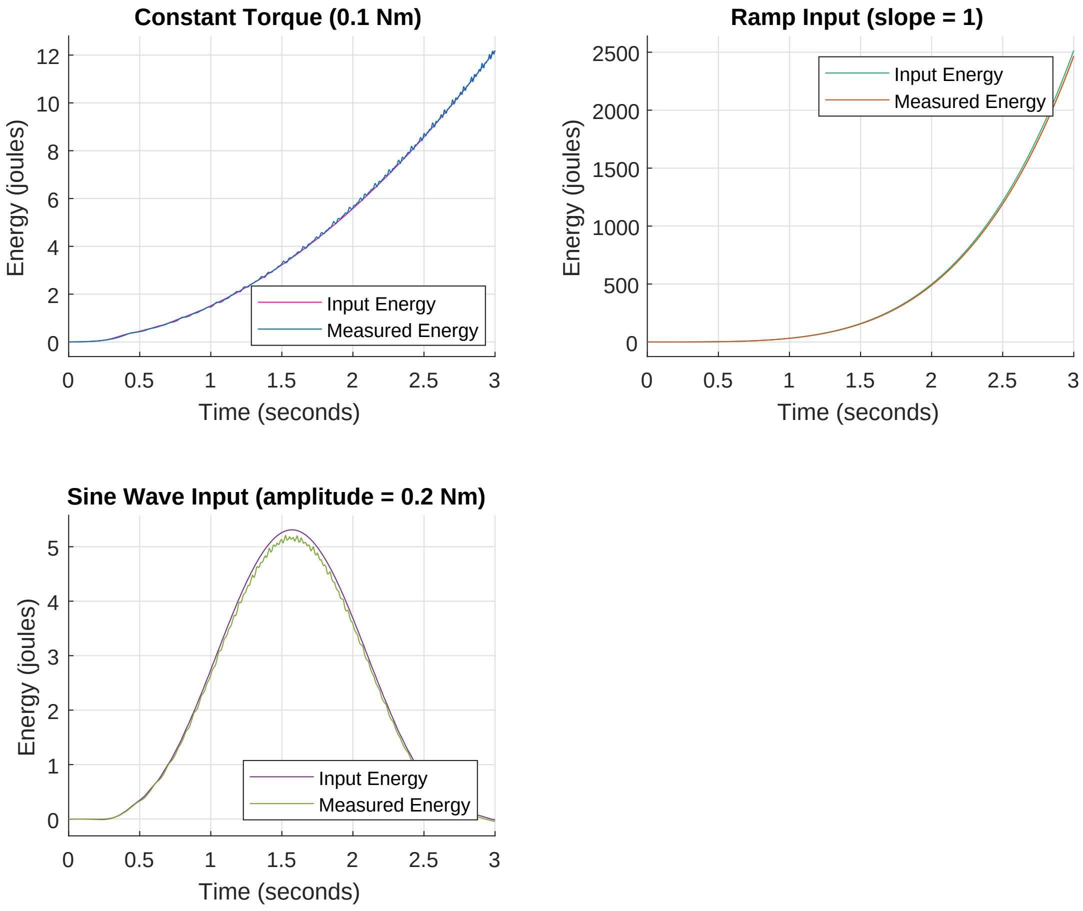


Figure A.2: Input vs Measured Energy

Appendix B: Nonlinear Model

B.1 Mass Matrix ($\mathrm{D}_{3\times 3}$)

$$
\begin{aligned}
d_{11} = {} & \big(\big(2\,l^2 m\,(\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 - 2\,\mathit{I3}_{xx} + 2\,\mathit{I3}_{zz}\big)\,(\cos(\theta_3))^2 \\
& \qquad + \big(-4\,\mathit{I3}_{zx}\sin(\theta_3) - 2\,l^2\beta_p m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\big)\cos(\theta_3) \\
& \qquad - l^2\,\big((\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 + {\beta_p}^2\,\big({\nu_2}^2\lambda_p - \lambda_p + 1\big)\big)\,m + \mathit{I3}_{xx} - \mathit{I3}_{zz} - \mathit{I2}_{xx} \\
& \qquad + \mathit{I2}_{zz}\big)\,(\cos(\theta_2))^2 \\
& - 2\,\big(2\,\mathit{I3}_{zx}\,(\cos(\theta_3))^2 + \sin(\theta_3)\,\big(l^2 m\,(\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 - \mathit{I3}_{xx} + \mathit{I3}_{zz}\big)\cos(\theta_3) \\
& \qquad - l^2\beta_p m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\sin(\theta_3) - \mathit{I3}_{zx} + \mathit{I2}_{zx}\big)\sin(\theta_2)\cos(\theta_2) \\
& + \big(-l^2 m\,(\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 + \mathit{I3}_{xx} - \mathit{I3}_{zz}\big)\,(\cos(\theta_3))^2 \\
& + \big(2\,\mathit{I3}_{zx}\sin(\theta_3) + 2\,l^2\beta_p m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\big)\cos(\theta_3) \\
& + \big(\big({\beta_p}^2\,\big({\nu_2}^2\lambda_p - \lambda_p + 1\big) + {\beta_a}^2\big)\,l^2 - 2\,\beta_a dc_3\,(\lambda_p-1)\,l - {dc_3}^2\,(\lambda_p-1)\big)\,m \\
& + \mathit{I3}_{zz} + \mathit{I1}_{zz} + \mathit{I2}_{xx} \\
d_{12} = {} & \big(\big(-\beta_a m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\,l^2 - m\,dc_3\nu_3\,(\lambda_p-1)\,(\beta_p-1)\,l + \mathit{I3}_{yz}\big)\cos(\theta_3) \\
& - \sin(\theta_3)\,\mathit{I3}_{yx} - (1+(\nu_2-1)\,\lambda_p)\,\beta_p m\beta_a l^2 + \beta_p m\,dc_3\,(\lambda_p-1)\,l + \mathit{I2}_{yz}\big)\cos(\theta_2) \\
& + \sin(\theta_2)\,\big(-\mathit{I3}_{yx}\cos(\theta_3) \\
& + \big(\beta_a m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\,l^2 + m\,dc_3\nu_3\,(\lambda_p-1)\,(\beta_p-1)\,l - \mathit{I3}_{yz}\big)\sin(\theta_3) - \mathit{I2}_{yx}\big) \\
d_{13} = {} & ((-lm\,(\lambda_p-1)\,(\beta_p-1)\,(\beta_a l + dc_3)\,\nu_3 + \mathit{I3}_{yz})\cos(\theta_3) - \sin(\theta_3)\,\mathit{I3}_{yx})\cos(\theta_2) \\
& + (-\mathit{I3}_{yx}\cos(\theta_3) + \sin(\theta_3)\,(lm\,(\lambda_p-1)\,(\beta_p-1)\,(\beta_a l + dc_3)\,\nu_3 - \mathit{I3}_{yz}))\sin(\theta_2) \\
d_{21} = {} & \big(\big(-\beta_a m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\,l^2 - m\,dc_3\nu_3\,(\lambda_p-1)\,(\beta_p-1)\,l + \mathit{I3}_{yz}\big)\cos(\theta_3) \\
& - \sin(\theta_3)\,\mathit{I3}_{yx} - (1+(\nu_2-1)\,\lambda_p)\,\beta_p m\beta_a l^2 + \beta_p m\,dc_3\,(\lambda_p-1)\,l + \mathit{I2}_{yz}\big)\cos(\theta_2) \\
& + \sin(\theta_2)\,\big(-\mathit{I3}_{yx}\cos(\theta_3) \\
& + \big(\beta_a m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\,l^2 + m\,dc_3\nu_3\,(\lambda_p-1)\,(\beta_p-1)\,l - \mathit{I3}_{yz}\big)\sin(\theta_3) - \mathit{I2}_{yx}\big) \\
d_{22} = {} & 2\,l^2 m\nu_3\beta_p\,(\lambda_p-1)\,(\beta_p-1)\cos(\theta_3) \\
& + m\,\big(\big((1-\lambda_p)\,{\nu_3}^2 + {\nu_2}^2\lambda_p - \lambda_p + 1\big)\,{\beta_p}^2 + 2\,{\nu_3}^2\,(\lambda_p-1)\,\beta_p - {\nu_3}^2\,(\lambda_p-1)\big)\,l^2 \\
& + \mathit{I3}_{yy} + \mathit{I2}_{yy} \\
d_{23} = {} & l^2 m\nu_3\beta_p\,(\lambda_p-1)\,(\beta_p-1)\cos(\theta_3) - l^2 m\,(\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 + \mathit{I3}_{yy} \\
d_{31} = {} & ((-lm\,(\lambda_p-1)\,(\beta_p-1)\,(\beta_a l + dc_3)\,\nu_3 + \mathit{I3}_{yz})\cos(\theta_3) - \sin(\theta_3)\,\mathit{I3}_{yx})\cos(\theta_2) \\
& + (-\mathit{I3}_{yx}\cos(\theta_3) + \sin(\theta_3)\,(lm\,(\lambda_p-1)\,(\beta_p-1)\,(\beta_a l + dc_3)\,\nu_3 - \mathit{I3}_{yz}))\sin(\theta_2)
\end{aligned}
$$

$$d_{32} = l^2 m \nu_3 \beta_p \left(\lambda_p - 1\right) \left(\beta_p - 1\right) \cos\left(\theta_3\right) - l^2 m \left(\beta_p - 1\right)^2 \left(\lambda_p - 1\right) {\nu_3}^2 + \mathit{I3}_{yy}$$
$$d_{33} = -l^2 m \left(\beta_p - 1\right)^2 \left(\lambda_p - 1\right) {\nu_3}^2 + \mathit{I3}_{yy}$$

B.2 Coriolis/Centrifugal Matrix ($\mathrm{C}_{3\times 3}$)

$$\begin{aligned}
c_{11}& \\
= 0.5\ \Big(-2\ \Big(&\left(2\,l^2 m \left(\beta_p - 1\right)^2 \left(\lambda_p - 1\right) {\nu_3}^2 - 2\,\mathit{I3}_{xx} + 2\,\mathit{I3}_{zz}\right) \left(\cos\left(\theta_3\right)\right)^2 \\
&+ \left(-4\,\mathit{I3}_{zx} \sin\left(\theta_3\right) - 2\,l^2 \beta_p m \nu_3 \left(\lambda_p - 1\right) \left(\beta_p - 1\right)\right) \cos\left(\theta_3\right) \\
&- l^2 \left(\left(\beta_p - 1\right)^2 \left(\lambda_p - 1\right) {\nu_3}^2 + {\beta_p}^2 \left({\nu_2}^2 \lambda_p - \lambda_p + 1\right)\right) m + \mathit{I3}_{xx} - \mathit{I3}_{zz} - \mathit{I2}_{xx} \\
&+ \mathit{I2}_{zz}\Big) \cos\left(\theta_2\right) \sin\left(\theta_2\right) \\
- 2\ \Big(&2\,\mathit{I3}_{zx} \left(\cos\left(\theta_3\right)\right)^2 + \sin\left(\theta_3\right) \left(l^2 m \left(\beta_p - 1\right)^2 \left(\lambda_p - 1\right) {\nu_3}^2 - \mathit{I3}_{xx} + \mathit{I3}_{zz}\right) \cos\left(\theta_3\right) \\
&- l^2 \beta_p m \nu_3 \left(\lambda_p - 1\right) \left(\beta_p - 1\right) \sin\left(\theta_3\right) - \mathit{I3}_{zx} + \mathit{I2}_{zx}\Big) \left(\cos\left(\theta_2\right)\right)^2 \\
+ 2\ \Big(&2\,\mathit{I3}_{zx} \left(\cos\left(\theta_3\right)\right)^2 + \sin\left(\theta_3\right) \left(l^2 m \left(\beta_p - 1\right)^2 \left(\lambda_p - 1\right) {\nu_3}^2 - \mathit{I3}_{xx} + \mathit{I3}_{zz}\right) \cos\left(\theta_3\right) \\
&- l^2 \beta_p m \nu_3 \left(\lambda_p - 1\right) \left(\beta_p - 1\right) \sin\left(\theta_3\right) - \mathit{I3}_{zx} + \mathit{I2}_{zx}\Big) \left(\sin\left(\theta_2\right)\right)^2\Big)\, \dot{\theta}_2 \\
+ 0.5\ \Big(\Big(-2\ &\left(2\,l^2 m \left(\beta_p - 1\right)^2 \left(\lambda_p - 1\right) {\nu_3}^2 - 2\,\mathit{I3}_{xx} + 2\,\mathit{I3}_{zz}\right) \cos\left(\theta_3\right) \sin\left(\theta_3\right) \\
&- 4\,\mathit{I3}_{zx} \left(\cos\left(\theta_3\right)\right)^2 \\
&- \left(-4\,\mathit{I3}_{zx} \sin\left(\theta_3\right) - 2\,l^2 \beta_p m \nu_3 \left(\lambda_p - 1\right) \left(\beta_p - 1\right)\right) \sin\left(\theta_3\right)\Big) \left(\cos\left(\theta_2\right)\right)^2 \\
-2\ \Big(&-4\,\mathit{I3}_{zx} \sin\left(\theta_3\right) \cos\left(\theta_3\right) + \left(\cos\left(\theta_3\right)\right)^2 \left(l^2 m \left(\beta_p - 1\right)^2 \left(\lambda_p - 1\right) {\nu_3}^2 - \mathit{I3}_{xx} + \mathit{I3}_{zz}\right) \\
&- \left(\sin\left(\theta_3\right)\right)^2 \left(l^2 m \left(\beta_p - 1\right)^2 \left(\lambda_p - 1\right) {\nu_3}^2 - \mathit{I3}_{xx} + \mathit{I3}_{zz}\right) \\
&- l^2 m \nu_3 \beta_p \left(\lambda_p - 1\right) \left(\beta_p - 1\right) \cos\left(\theta_3\right)\Big) \sin\left(\theta_2\right) \cos\left(\theta_2\right) \\
&- 2\ \left(-l^2 m \left(\beta_p - 1\right)^2 \left(\lambda_p - 1\right) {\nu_3}^2 + \mathit{I3}_{xx} - \mathit{I3}_{zz}\right) \cos\left(\theta_3\right) \sin\left(\theta_3\right) \\
+ 2\,\mathit{I3}_{zx} &\left(\cos\left(\theta_3\right)\right)^2 - \left(2\,\mathit{I3}_{zx} \sin\left(\theta_3\right) + 2\,l^2 \beta_p m \nu_3 \left(\lambda_p - 1\right) \left(\beta_p - 1\right)\right) \sin\left(\theta_3\right)\Big)\, \dot{\theta}_3
\end{aligned}$$

$$
\begin{aligned}
c_{12} = 0.5\ \Bigl(-2\ \Bigl(\bigl(2\,l^2 m\,(\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 - 2\,\mathit{I3}_{xx} + 2\,\mathit{I3}_{zz}\bigr)\,(\cos(\theta_3))^2 \\
+ \bigl(-4\,\mathit{I3}_{zx}\sin(\theta_3) - 2\,l^2\beta_p m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\bigr)\cos(\theta_3) \\
- l^2\,\bigl((\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 + {\beta_p}^2\,\bigl({\nu_2}^2\lambda_p - \lambda_p + 1\bigr)\bigr)\,m + \mathit{I3}_{xx} - \mathit{I3}_{zz} - \mathit{I2}_{xx} \\
+ \mathit{I2}_{zz}\Bigr)\cos(\theta_2)\sin(\theta_2) \\
- 2\ \bigl(2\,\mathit{I3}_{zx}\,(\cos(\theta_3))^2 + \sin(\theta_3)\,\bigl(l^2 m\,(\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 - \mathit{I3}_{xx} + \mathit{I3}_{zz}\bigr)\cos(\theta_3) \\
- l^2\beta_p m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\sin(\theta_3) - \mathit{I3}_{zx} + \mathit{I2}_{zx}\bigr)\,(\cos(\theta_2))^2 \\
+ 2\ \bigl(2\,\mathit{I3}_{zx}\,(\cos(\theta_3))^2 + \sin(\theta_3)\,\bigl(l^2 m\,(\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 - \mathit{I3}_{xx} + \mathit{I3}_{zz}\bigr)\cos(\theta_3) \\
- l^2\beta_p m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\sin(\theta_3) - \mathit{I3}_{zx} + \mathit{I2}_{zx}\bigr)\,(\sin(\theta_2))^2\Bigr)\,\dot{\theta}_1 \\
+ \dot{\theta}_2\ \bigl(\bigl(-1.0\,\mathit{I3}_{yx}\cos(\theta_2) \\
+\bigl(\bigl(lm\nu_3\,(\beta_a l+dc_3)\,\lambda_p-1.0\,l^2 m\beta_a\nu_3-1.0\,lmdc_3\nu_3\bigr)\,\beta_p+\bigl(-1.0\,l^2 m\beta_a\nu_3-1.0\,lmdc_3\nu_3\bigr)\,\lambda_p+l^2 m\beta_a\nu \\
+\bigl(\bigl(\bigl(lm\nu_3\,(\beta_a l+dc_3)\,\lambda_p-1.0\,l^2 m\beta_a\nu_3-1.0\,lmdc_3\nu_3\bigr)\,\beta_p+\bigl(-1.0\,l^2 m\beta_a\nu_3-1.0\,lmdc_3\nu_3\bigr)\,\lambda_p+l^2 m\beta_a \\
+ \mathit{I3}_{yx}\sin(\theta_2)\bigr)\sin(\theta_3) - 1.0\,\mathit{I2}_{yx}\cos(\theta_2) \\
+ \bigl(\bigl(\bigl((\nu_2-1.0)\,\beta_a ml^2 - 1.0\,mdc_3 l\bigr)\,\lambda_p + lm\,(\beta_a l + dc_3)\bigr)\,\beta_p - 1.0\,\mathit{I2}_{yz}\bigr)\sin(\theta_2)\bigr) \\
+ \dot{\theta}_3\ \bigl(\bigl(-1.0\,\mathit{I3}_{yx}\cos(\theta_3) \\
+\bigl(\bigl(\bigl(\bigl(l^2\beta_p-1.0\,l^2\bigr)\,\nu_3\lambda_p+\bigl(-1.0\,l^2\beta_p+l^2\bigr)\,\nu_3\bigr)\,\beta_a+(\beta_p l-1.0\,l)\,\nu_3 dc_3\lambda_p+(-1.0\,\beta_p l+l)\,\nu_3 dc_3\bigr)\,m-1.0 \\
+\bigl(\bigl(\bigl(\bigl(\bigl(l^2\beta_p-1.0\,l^2\bigr)\,\nu_3\lambda_p+\bigl(-1.0\,l^2\beta_p+l^2\bigr)\,\nu_3\bigr)\,\beta_a+(\beta_p l-1.0\,l)\,\nu_3 dc_3\lambda_p+(-1.0\,\beta_p l+l)\,\nu_3 dc_3\bigr)\,m-1 \\
+ \sin(\theta_3)\,\mathit{I3}_{yx}\bigr)\sin(\theta_2)\bigr)
\end{aligned}
$$

$$
\begin{aligned}
c_{13} = 0.5\ \Bigl(\Bigl(-2\ \bigl(2\,l^2 m\,(\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 - 2\,\mathit{I3}_{xx} + 2\,\mathit{I3}_{zz}\bigr)\cos(\theta_3)\sin(\theta_3) \\
- 4\,\mathit{I3}_{zx}\,(\cos(\theta_3))^2 \\
- \bigl(-4\,\mathit{I3}_{zx}\sin(\theta_3) - 2\,l^2\beta_p m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\bigr)\sin(\theta_3)\Bigr)\,(\cos(\theta_2))^2 \\
- 2\ \bigl(-4\,\mathit{I3}_{zx}\sin(\theta_3)\cos(\theta_3) + (\cos(\theta_3))^2\,\bigl(l^2 m\,(\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 - \mathit{I3}_{xx} + \mathit{I3}_{zz}\bigr) \\
- (\sin(\theta_3))^2\,\bigl(l^2 m\,(\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 - \mathit{I3}_{xx} + \mathit{I3}_{zz}\bigr) \\
- l^2 m\nu_3\beta_p\,(\lambda_p-1)\,(\beta_p-1)\cos(\theta_3)\bigr)\sin(\theta_2)\cos(\theta_2) \\
- 2\ \bigl(-l^2 m\,(\beta_p-1)^2\,(\lambda_p-1)\,{\nu_3}^2 + \mathit{I3}_{xx} - \mathit{I3}_{zz}\bigr)\cos(\theta_3)\sin(\theta_3) + 2\,\mathit{I3}_{zx}\,(\cos(\theta_3))^2 \\
- \bigl(2\,\mathit{I3}_{zx}\sin(\theta_3) + 2\,l^2\beta_p m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\bigr)\sin(\theta_3)\Bigr)\,\dot{\theta}_1 + \dot{\theta}_2\ \bigl(\bigl(-1.0\,\mathit{I3}_{yx}\cos(\theta_2) \\
+\bigl(((m\nu_3\beta_a\lambda_p-1.0\,\beta_a m\nu_3)\,\beta_p-1.0\,m\nu_3\beta_a\lambda_p+\beta_a m\nu_3)\,l^2+((mdc_3\nu_3\lambda_p-1.0\,mdc_3\nu_3)\,\beta_p-1.0\,mdc_3\nu \\
+\bigl(\bigl(((m\nu_3\beta_a\lambda_p-1.0\,\beta_a m\nu_3)\,\beta_p-1.0\,m\nu_3\beta_a\lambda_p+\beta_a m\nu_3)\,l^2+((mdc_3\nu_3\lambda_p-1.0\,mdc_3\nu_3)\,\beta_p-1.0\,mdc_3 \\
+ \mathit{I3}_{yx}\sin(\theta_2)\bigr)\sin(\theta_3)\bigr) + \dot{\theta}_3\ \bigl(\bigl(-1.0\,\mathit{I3}_{yx}\cos(\theta_2) \\
+\bigl(((m\nu_3\lambda_p-1.0\,m\nu_3)\,\beta_p-1.0\,m\nu_3\lambda_p+m\nu_3)\,\beta_a l^2+((mdc_3\nu_3\lambda_p-1.0\,mdc_3\nu_3)\,\beta_p-1.0\,mdc_3\nu_3\lambda_p+m \\
+\bigl(\bigl(((m\nu_3\lambda_p-1.0\,m\nu_3)\,\beta_p-1.0\,m\nu_3\lambda_p+m\nu_3)\,\beta_a l^2+((mdc_3\nu_3\lambda_p-1.0\,mdc_3\nu_3)\,\beta_p-1.0\,mdc_3\nu_3\lambda_p+ \\
+ \mathit{I3}_{yx}\sin(\theta_2)\bigr)\sin(\theta_3)\bigr)
\end{aligned}
$$

$$
\begin{aligned}
c_{21} = {} & 1.0\ \Big(\big(2.0\,\mathit{I3}_{zx}\,(\cos(\theta_3))^2 \\
& +\big(\big((-1.0+1.0\,\lambda_p)\,{\beta_p}^2+(2.0-2.0\,\lambda_p)\,\beta_p-1.0+1.0\,\lambda_p\big)\,ml^2{\nu_3}^2-1.0\,\mathit{I3}_{xx}+1.0\,\mathit{I3}_{zz}\big)\sin(\theta_3)\cos(\theta_3) \\
& \qquad + \big((1.0-1.0\,\lambda_p)\,{\beta_p}^2+(-1.0+1.0\,\lambda_p)\,\beta_p\big)\,\nu_3 ml^2\sin(\theta_3)+1.0\,\mathit{I2}_{zx} \\
& \qquad\qquad - 1.0\,\mathit{I3}_{zx}\big)\,(\cos(\theta_2))^2 \\
& +\Big(\big(\big((2.0\,\lambda_p-2.0)\,{\beta_p}^2+(-4.0\,\lambda_p+4.0)\,\beta_p+2.0\,\lambda_p-2.0\big)\,{\nu_3}^2ml^2-2.0\,\mathit{I3}_{xx}+2.0\,\mathit{I3}_{zz}\big)\,(\cos(\theta_3))^2 \\
& \qquad + \big(-4.0\,\mathit{I3}_{zx}\sin(\theta_3)+\big((2.0-2.0\,\lambda_p)\,{\beta_p}^2+(2.0\,\lambda_p-2.0)\,\beta_p\big)\,\nu_3 ml^2\big)\cos(\theta_3) \\
& +\big(\big((1.0-1.0\,\lambda_p)\,{\beta_p}^2+(2.0\,\lambda_p-2.0)\,\beta_p+1.0-1.0\,\lambda_p\big)\,{\nu_3}^2+\big(-1.0+1.0\,\lambda_p-1.0\,{\nu_2}^2\lambda_p\big)\,{\beta_p}^2\big)\,ml^2 \\
& - 1.0\,\mathit{I2}_{xx}+1.0\,\mathit{I2}_{zz}+1.0\,\mathit{I3}_{xx}-1.0\,\mathit{I3}_{zz}\Big)\sin(\theta_2)\cos(\theta_2)+\big(-2.0\,\mathit{I3}_{zx}\,(\cos(\theta_3))^2 \\
& +\big(\big((1.0-1.0\,\lambda_p)\,{\beta_p}^2+(2.0\,\lambda_p-2.0)\,\beta_p+1.0-1.0\,\lambda_p\big)\,{\nu_3}^2ml^2+1.0\,\mathit{I3}_{xx}-1.0\,\mathit{I3}_{zz}\big)\sin(\theta_3)\cos(\theta_3) \\
& \qquad + \big((-1.0+1.0\,\lambda_p)\,{\beta_p}^2+(1.0-1.0\,\lambda_p)\,\beta_p\big)\,\nu_3 ml^2\sin(\theta_3)-1.0\,\mathit{I2}_{zx} \\
& \qquad\qquad + 1.0\,\mathit{I3}_{zx}\big)\,(\sin(\theta_2))^2\Big)\,\dot{\theta}_1 \\
c_{22} = {} & -1.0\,l^2\beta_p m\nu_3\,(\lambda_p-1)\,(\beta_p-1)\sin(\theta_3)\,\dot{\theta}_3 \\
c_{23} = {} & -1.0\,l^2\beta_p\sin(\theta_3)\,m\,\Big(\Big(\Big(1.0\,\dot{\theta}_2+1.0\,\dot{\theta}_3\Big)\,\lambda_p-1.0\,\dot{\theta}_2-1.0\,\dot{\theta}_3\Big)\,\beta_p \\
& \qquad + \Big(-1.0\,\dot{\theta}_2-1.0\,\dot{\theta}_3\Big)\,\lambda_p+1.0\,\dot{\theta}_2+1.0\,\dot{\theta}_3\Big)\,\nu_3
\end{aligned}
$$

$$
\begin{aligned}
c_{31} = {} & 2.0\ \big(\big(-1.0\ (\sin(\theta_3))^2\ \mathit{I3}_{zx} \\
& +\big(\big(\big((-1.0+1.0\,\lambda_p)\,{\beta_p}^2+(2.0-2.0\,\lambda_p)\,\beta_p-1.0+1.0\,\lambda_p\big)\,ml^2{\nu_3}^2-1.0\,\mathit{I3}_{xx}+1.0\,\mathit{I3}_{zz}\big)\cos(\theta_3)+\big((-0 \\
& \qquad\qquad + 1.0\,\mathit{I3}_{zx}\,(\cos(\theta_3))^2\big)\,(\cos(\theta_2))^2 \\
& +\big(\big(\big((-0.5\,\lambda_p+0.5)\,{\beta_p}^2+(-1.0+1.0\,\lambda_p)\,\beta_p-0.5\,\lambda_p+0.5\big)\,ml^2{\nu_3}^2+0.5\,\mathit{I3}_{xx}-0.5\,\mathit{I3}_{zz}\big)\sin(\theta_2)\,(\sin(\\
& \qquad\qquad - 2.0\ \cos(\theta_3)\sin(\theta_3)\sin(\theta_2)\ \mathit{I3}_{zx} \\
& +\big(\big((0.5\,\lambda_p-0.5)\,{\beta_p}^2+(1.0-1.0\,\lambda_p)\,\beta_p+0.5\,\lambda_p-0.5\big)\,ml^2{\nu_3}^2-0.5\,\mathit{I3}_{xx}+0.5\,\mathit{I3}_{zz}\big)\sin(\theta_2)\,(\cos(\theta_3))^2 \\
& \qquad + \big((-0.5\,\lambda_p+0.5)\,{\beta_p}^2+(0.5\,\lambda_p-0.5)\,\beta_p\big)\,ml^2\nu_3\sin(\theta_2)\cos(\theta_3)\big)\cos(\theta_2) \\
& \qquad\qquad + 0.5\ (\sin(\theta_3))^2\ \mathit{I3}_{zx} \\
& +\big(\big(\big((-0.5\,\lambda_p+0.5)\,{\beta_p}^2+(-1.0+1.0\,\lambda_p)\,\beta_p-0.5\,\lambda_p+0.5\big)\,ml^2{\nu_3}^2+0.5\,\mathit{I3}_{xx}-0.5\,\mathit{I3}_{zz}\big)\cos(\theta_3) \\
& + \big((0.5\,\lambda_p-0.5)\,{\beta_p}^2+(-0.5\,\lambda_p+0.5)\,\beta_p\big)\,ml^2\nu_3\big)\sin(\theta_3)-0.5\,\mathit{I3}_{zx}\,(\cos(\theta_3))^2\big)\,\dot{\theta}_1 \\
c_{32} = {} & l^2\beta_p m\nu_3\sin(\theta_3)\,\dot{\theta}_2\,(1.0\,\lambda_p\beta_p-1.0\,\beta_p-1.0\,\lambda_p+1.0) \\
c_{33} = {} & 0
\end{aligned}
$$

B.3 Gravitational Acceleration Matrix ($G_{3\times1}$)

$$
\begin{aligned}
g_1 = {} & 0 \\
g_2 = {} & -9.81\,\lambda_p m\nu_2\beta_p l\sin(\theta_2)+9.81\ (1-\lambda_p)\,m\,(-\sin(\theta_2)\,\nu_3\,(1-\beta_p)\,l\cos(\theta_3) \\
& \qquad - \cos(\theta_2)\,\nu_3\,(1-\beta_p)\,l\sin(\theta_3)-\beta_p l\sin(\theta_2)) \\
g_3 = {} & 9.81\ (1-\lambda_p)\,m\,(-\cos(\theta_2)\,\nu_3\,(1-\beta_p)\,l\sin(\theta_3)-\sin(\theta_2)\,\nu_3\,(1-\beta_p)\,l\cos(\theta_3))
\end{aligned}
$$

B.4 Total Energy Equation (TE)

$$\begin{aligned} TE = {} & 9.81\,\lambda_p m \nu_2 \beta_p l \cos(\theta_2) + 9.81\,(1-\lambda_p)\, m\, (\cos(\theta_2)\, \nu_3\, (1-\beta_p)\, l \cos(\theta_3) \\ & \qquad - \sin(\theta_2)\, \nu_3\, (1-\beta_p)\, l \sin(\theta_3) + \beta_p l \cos(\theta_2)) \\ & + \Big(1/2\, \dot{\theta}_2\, (2\, l^2 m \nu_3 \beta_p\, (\lambda_p - 1)\, (\beta_p - 1) \cos(\theta_3) + (((1-\lambda_p)\, {\nu_3}^2 + {\nu_2}^2 \lambda_p - \lambda_p + 1)\, {\beta_p}^2 + 2\, {\nu_3}^2\, (\lambda_p - 1)\, \beta \\ & + 1/2\, \dot{\theta}_3\, (l^2 m \nu_3 \beta_p\, (\lambda_p - 1)\, (\beta_p - 1) \cos(\theta_3) - l^2 m\, (\beta_p - 1)^2\, (\lambda_p - 1)\, {\nu_3}^2 + I3_{yy}) \Big)\, \dot{\theta}_2 \\ & + \Big(1/2\, \dot{\theta}_2\, (l^2 m \nu_3 \beta_p\, (\lambda_p - 1)\, (\beta_p - 1) \cos(\theta_3) - l^2 m\, (\beta_p - 1)^2\, (\lambda_p - 1)\, {\nu_3}^2 + I3_{yy}) + 1/2\, \dot{\theta}_3\, (-l^2 m\, (\beta_p \end{aligned}$$

$$TE_{ref} = 9.81\,\lambda_p m \nu_2 \beta_p l + 9.81\,(1-\lambda_p)\, m\, (\nu_3\, (1-\beta_p)\, l + \beta_p l)$$

B.5 Variables Dictionary

Symbols	Description
$I1_{ij}, I2_{ij}, I3_{ij}$	Inertia tensor elements of links 1, 2 and 3 respectively
m	Characteristic mass (refer Section 3.1.2)
l	Characteristic length (refer Section 3.1.2)
$\lambda_a, \lambda_p, \beta_a, \beta_p, \nu_i$	Dimensionless parameters (refer Section 3.1.2)
dc_3	Offset between links 2 and 3 (refer Table 3.2)
TE	Total energy of the pendulum segments
TE_{ref}	Reference total energy of the pendulum segments at the upright position

Appendix C: Computer-Aided Design Notes

This appendix shows the CAD drawings of the custom-designed components, and describes how the assembly is done. All the dimensions shown in the CAD drawings are in millimeters. Table C.1 shows the materials used for the components referenced in the CAD drawings and the method of manufacturing of the parts.

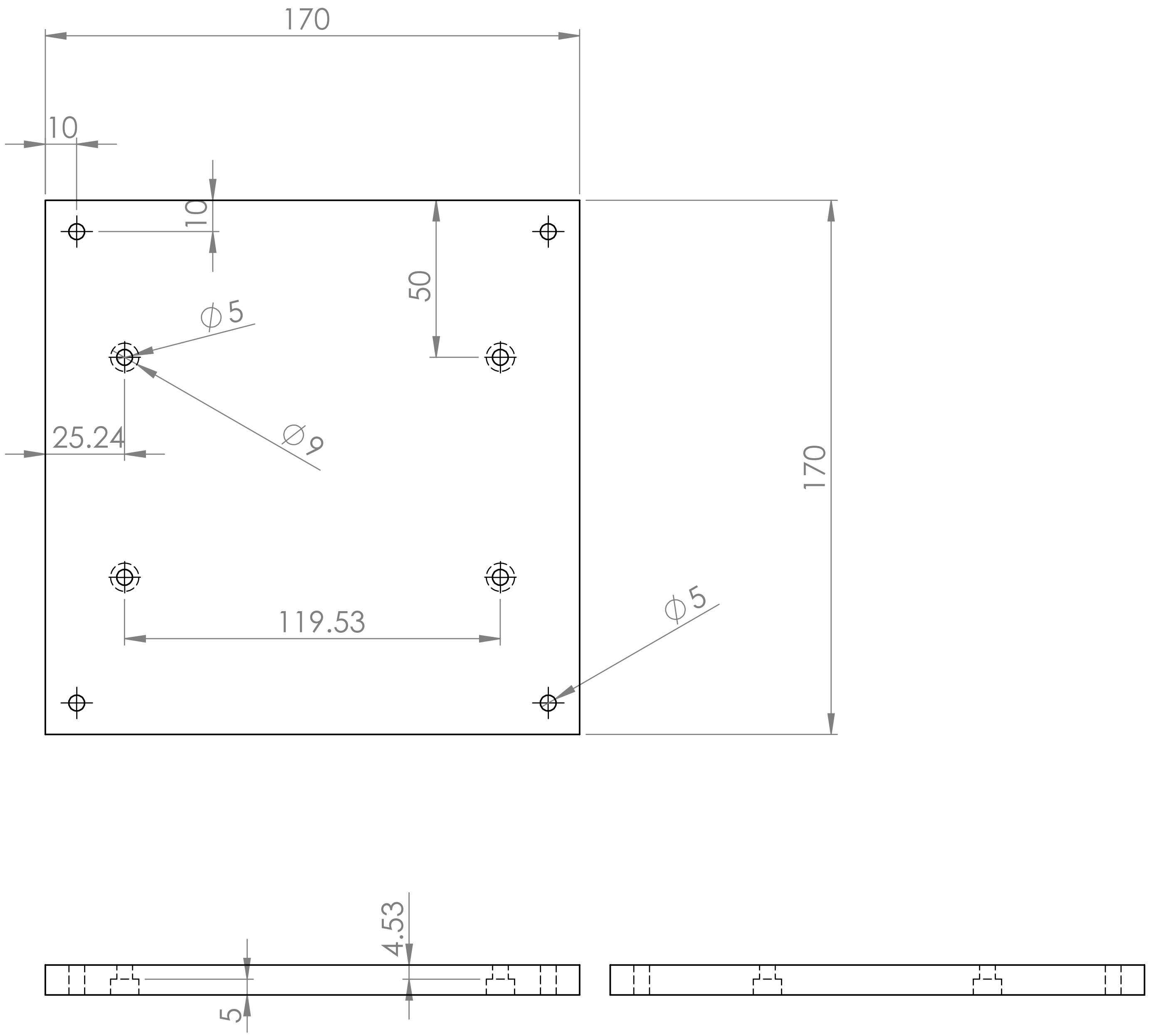
C.1 Assembly

Two of the ‘mount-sides’ parts are mounted on the ‘base’ using M5 screws. The motor is attached to the ‘mount-motor’ using M5 screws and then the ‘mount-motor’ part is mounted on the ‘mount-sides’ using M5 screws such that the motor shaft faces upwards. The 8mm to 1/2in coupling is clamped to the motor shaft and the smaller diameter shaft on each end, and then the ‘mount-top’ part is mounted on the ‘mount-sides’ using M5 screws, such that the 8mm counter-bore faces upwards. The 8mm bearing is inserted into the counter-bore. The ‘encoder-mount-1’ part is attached to ‘mount-top’ using M3 screws, and one of the encoders is attached to the ‘encoder-mount-1’ using M2.5 screws.

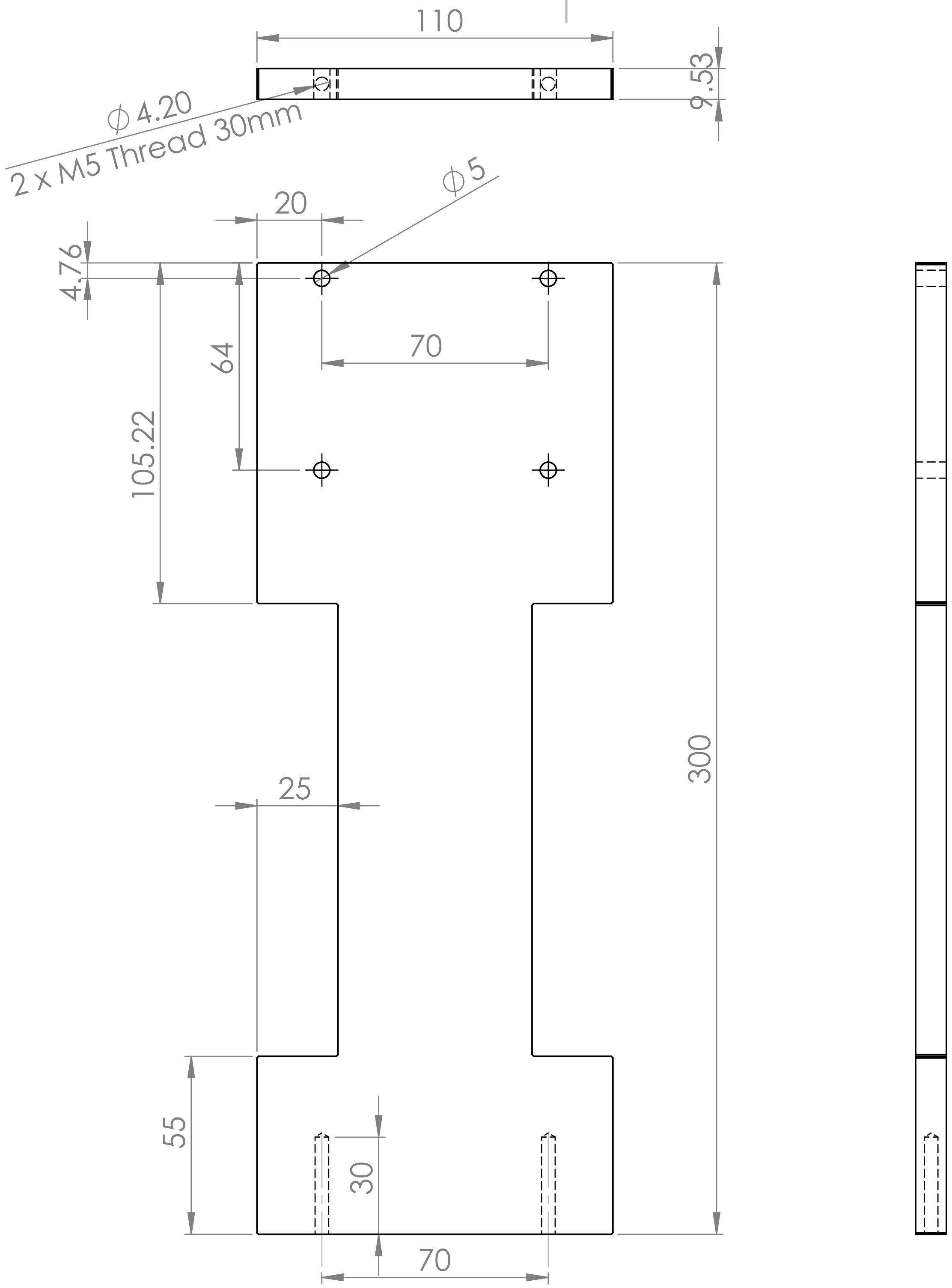
The 8mm set collar is attached to the 8mm shaft and one end of ‘link-1’ part is attached to the set collar using M5 screws. The 3mm needle roller bearing is press-fit into the ‘bearing-mount-roller’ part, and then the ‘bearing-mount-roller’ is attached to the other end of ‘link-1’. The 3mm step of the ‘pin-2’ part is inserted into the 3mm needle roller bearing, and then an encoder is mounted on the shaft. The base of the encoder is fixed to the ‘bearing-mount-roller’ part. The ‘link-2-roller’ part is attached to the other end of ‘pin-2’ using M3 screws. The ‘link-3’ part is similarly joined to the ‘link-2-roller’ part.

Part Name	Material	Manufacturing Method
base	Aluminium 6061	CNC Milling
mount-sides	Aluminium 6061	CNC Milling
mount-motor	Aluminium 6061	CNC Milling
mount-top	Aluminium 6061	CNC Milling
shaft-1	Aluminium 6061	Lathe Turning
encoder-mount-1	Polylactic acid	3D printing
link-1	Aluminium 6061	CNC Milling
bearing-mount-roller	Aluminium 6061	CNC Milling
link-2-roller	Aluminium 6061	CNC Milling
link-2	Aluminium 6061	CNC Milling
pin-2	Steel	Lathe Turning

Table C.1: Manufacturing method of mechanical components



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:			DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE		TITLE:	
DRAWN						
CHK'D						
APPV'D						
MFG						
Q.A			MATERIAL:		DWG NO. base	A
			WEIGHT:		SCALE:1:5	SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBURR AND BREAK SHARP EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN			
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

DWG NO.

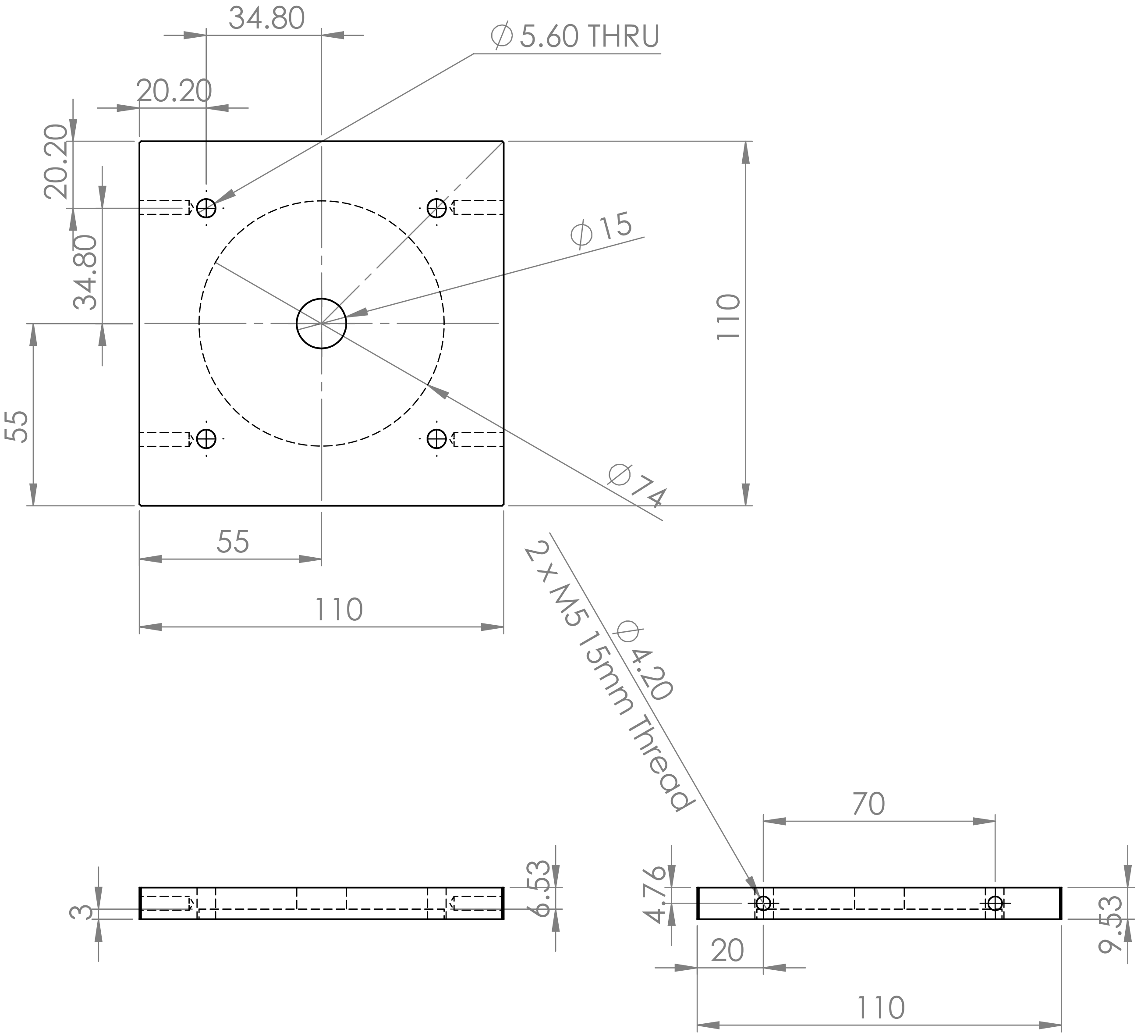
mount-sides

A

WEIGHT:

SCALE:1:2

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION

	NAME	SIGNATURE	DATE		
DRAWN					
CHK'D					
APPV'D					
MFG					
Q.A				MATERIAL:	

TITLE:

DWG NO.

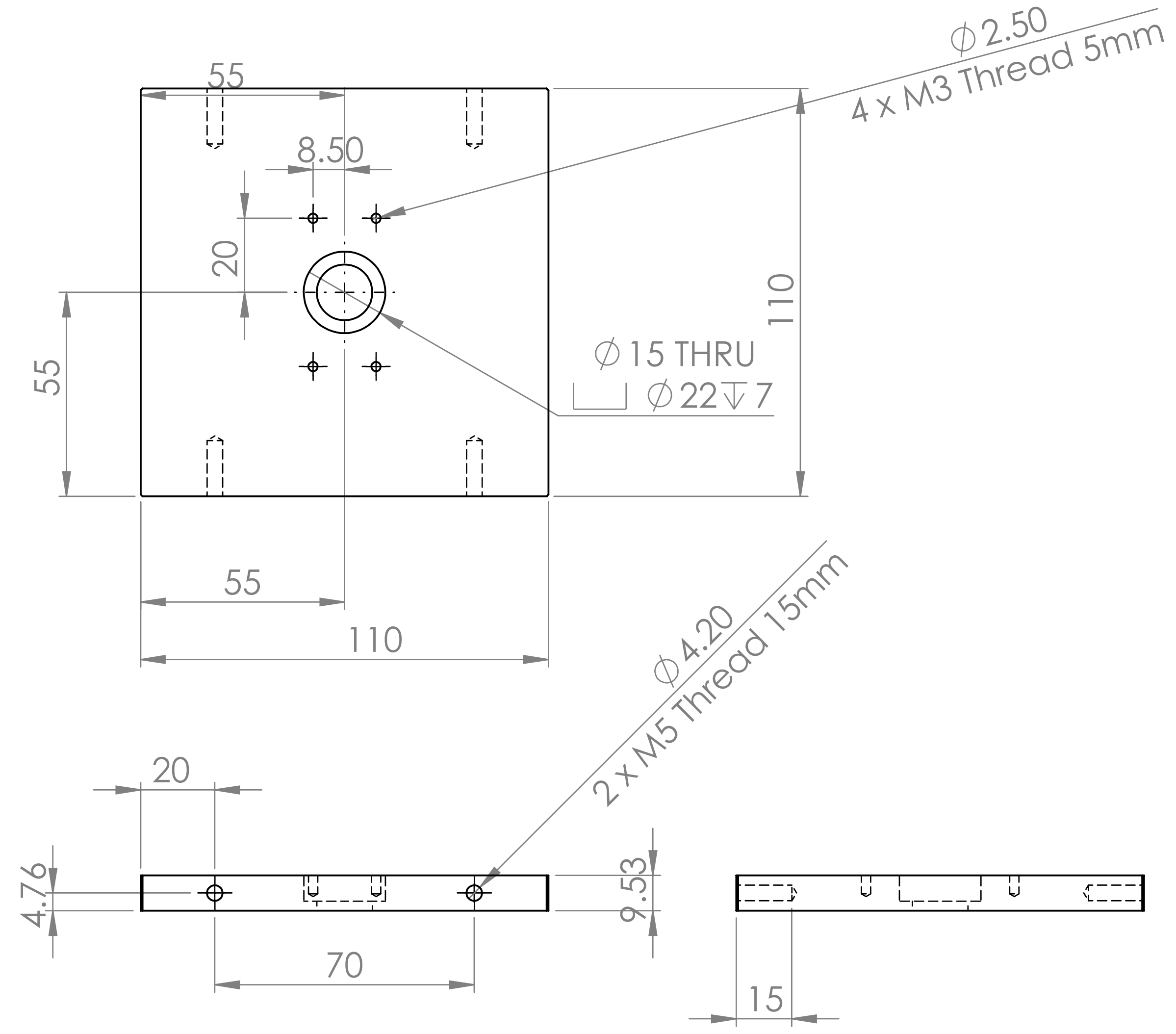
mount-motor

A

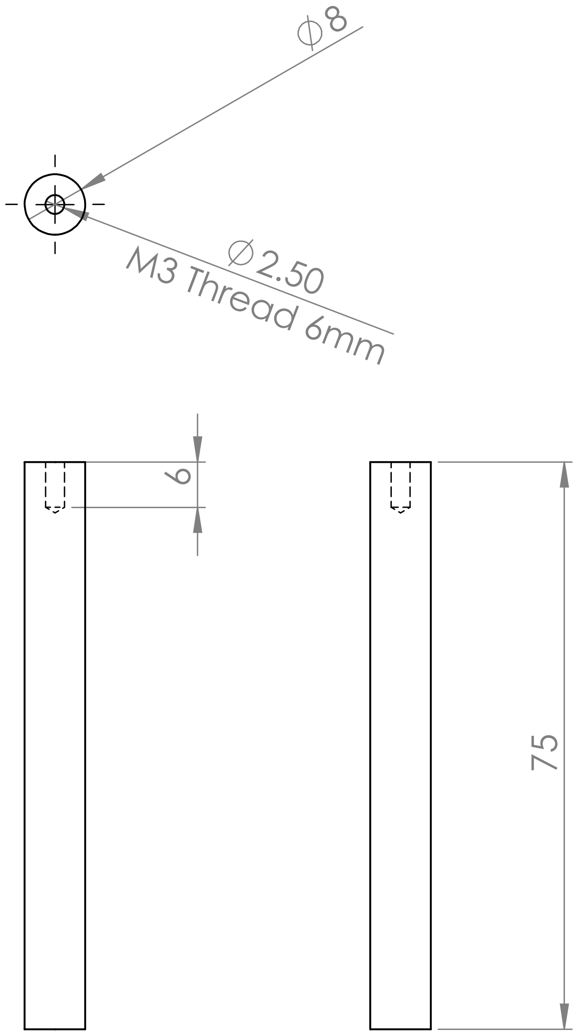
WEIGHT:

SCALE:1:2

SHEET 1 OF 1

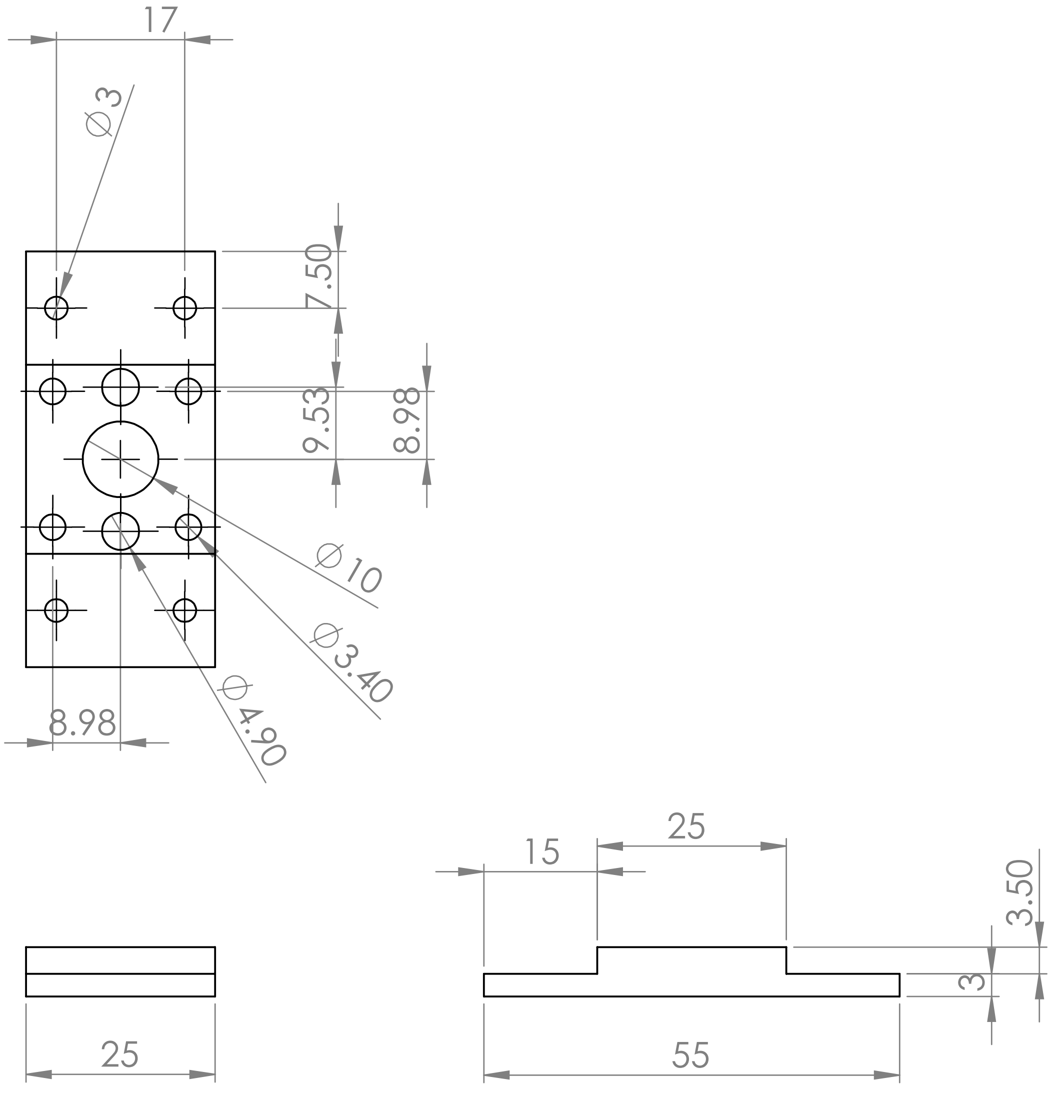


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:				DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE:	
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO. mount-top	A
				WEIGHT:		SCALE:1:2	SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING
			REVISION

	NAME	SIGNATURE	DATE				
							TITLE:
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:			DWG NO. shaft-1
							A
				WEIGHT:			SCALE:1:1
							SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBURR AND
BREAK SHARP
EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN			
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

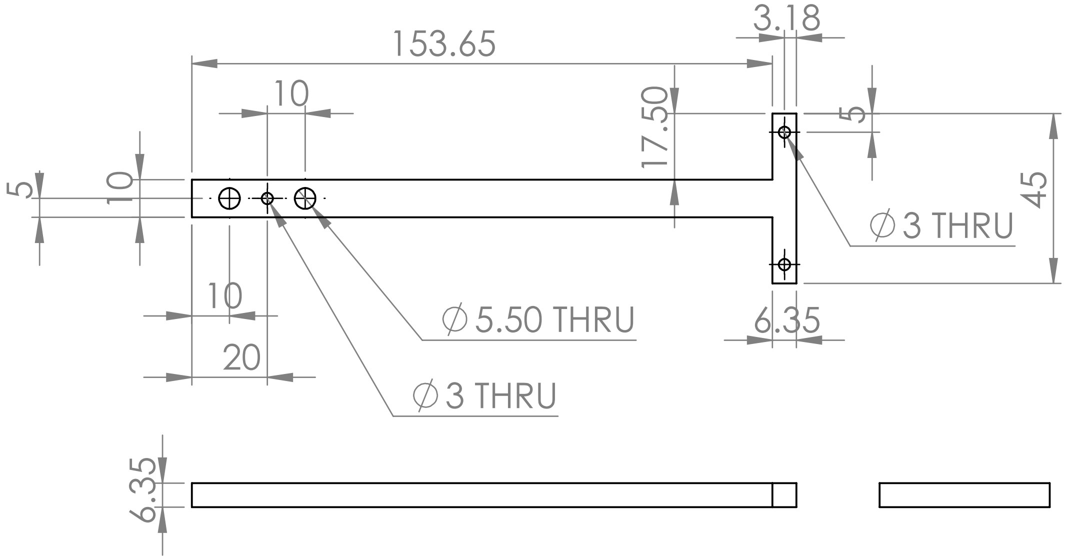
WEIGHT:

DWG NO.

encoder-mount-1 A

SCALE:1:1

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBURR AND
BREAK SHARP
EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE		
DRAWN					
CHK'D					
APPV'D					
MFG					
Q.A					

TITLE:

MATERIAL:

DWG NO.

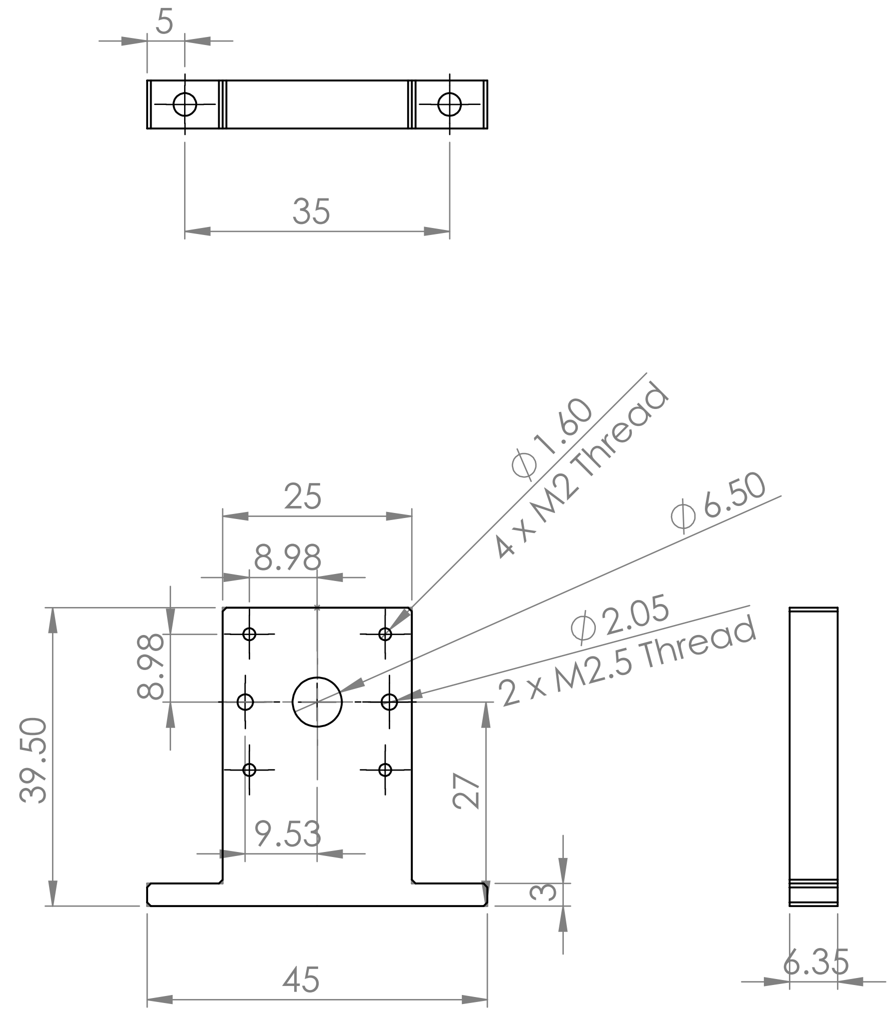
link-1

A

WEIGHT:

SCALE:1:2

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBURR AND
BREAK SHARP
EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN			
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

DWG NO.

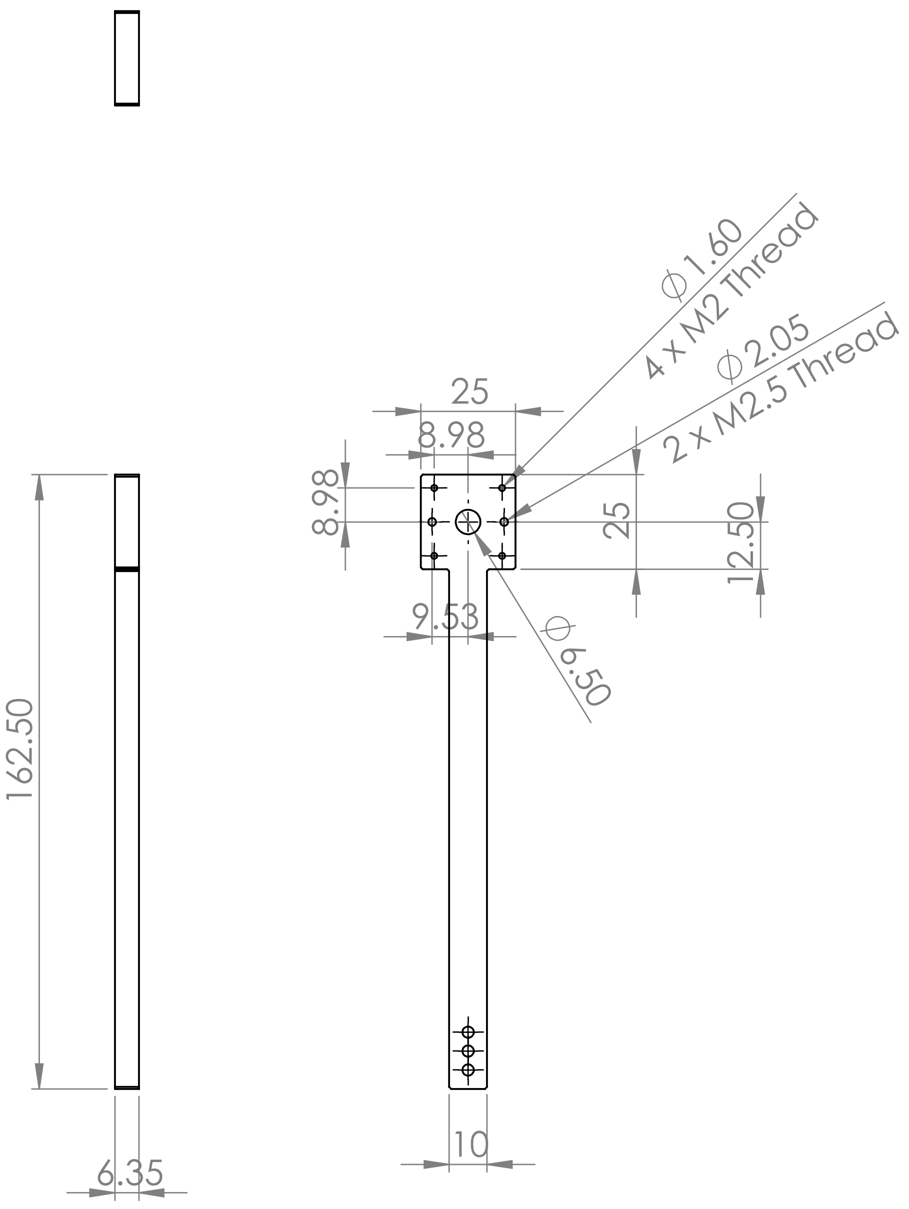
bearing-mount-roller

A

WEIGHT:

SCALE:2:1

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBURR AND
BREAK SHARP
EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN			
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

WEIGHT:

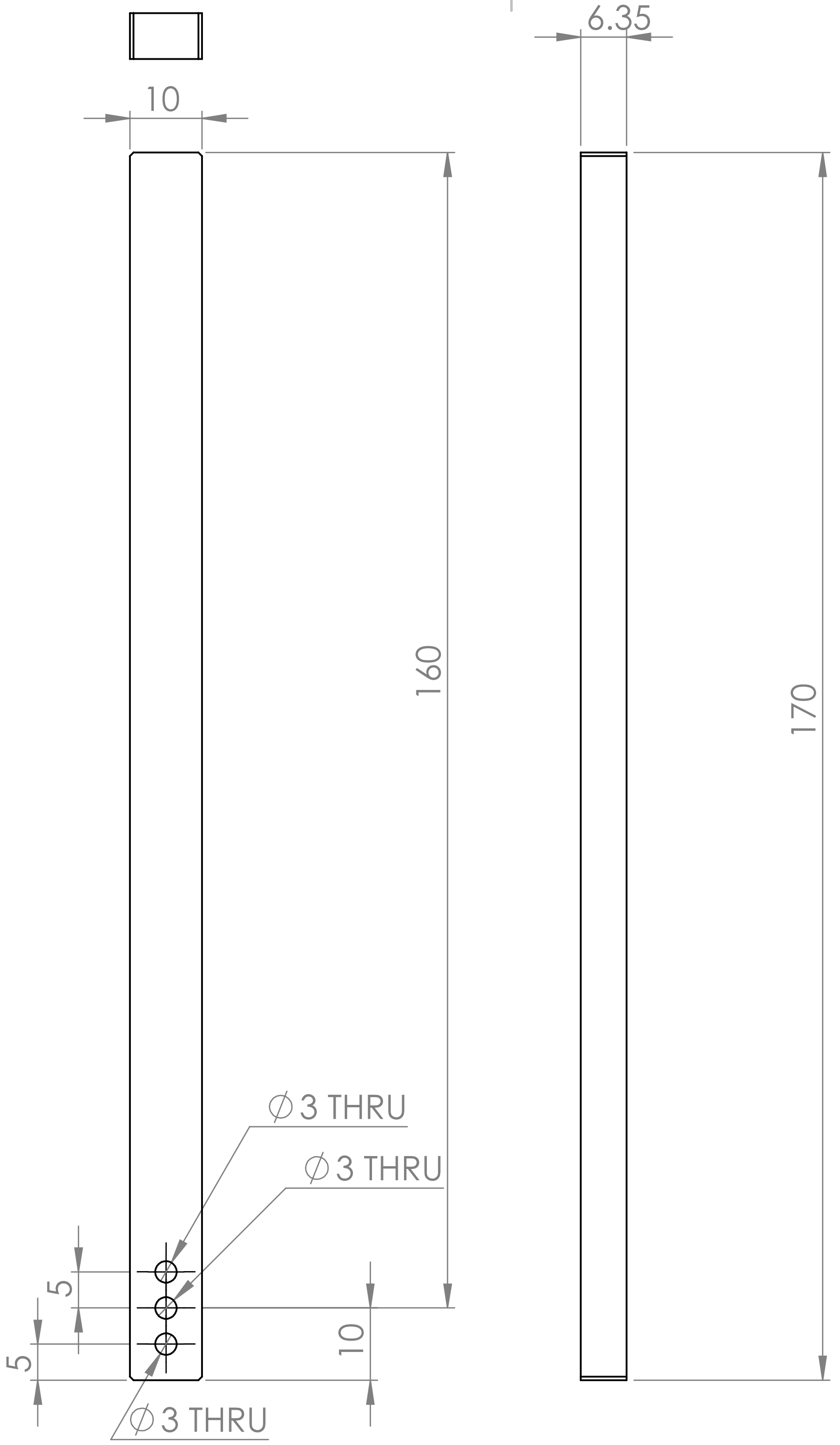
DWG NO.

link-2-roller

A

SCALE:1:2

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION

	NAME	SIGNATURE	DATE		
DRAWN					
CHK'D					
APPV'D					
MFG					
Q.A					

TITLE:

MATERIAL:

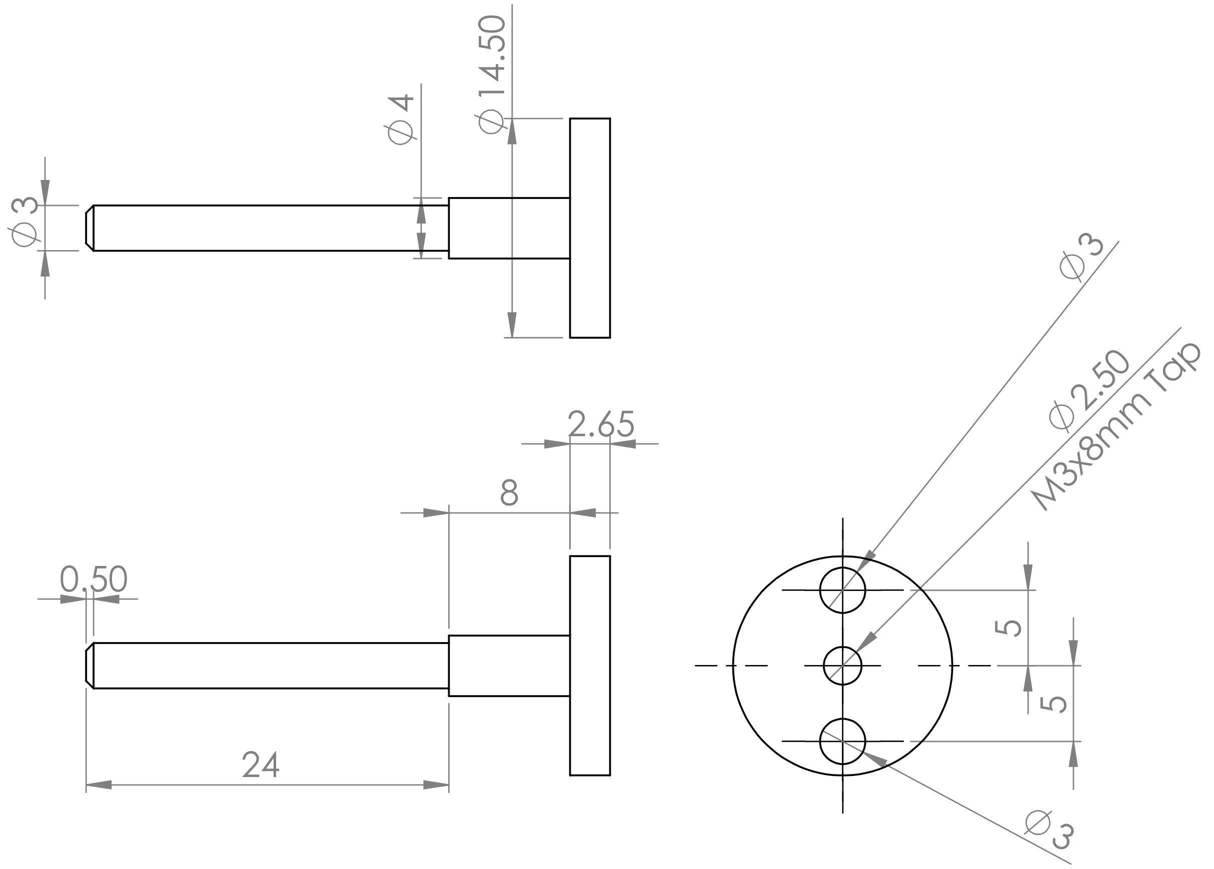
DWG NO. link-3

A

WEIGHT:

SCALE:1:2

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION

	NAME	SIGNATURE	DATE		
DRAWN					
CHK'D					
APPV'D					
MFG					
Q.A					

TITLE:

MATERIAL:

WEIGHT:

DWG NO.

pin-2

A

SCALE:2:1

SHEET 1 OF 1

Appendix D: Software Programs

D.1 ESP8266 12E Wireless Module Source Code

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <SPI.h>

#define baudRate 2000000
#define timoutLimit 100
#define nop 0x00            //no operation
#define rd_pos 0x10         //read position\

char ssid[] = "FurutaCentral";
char pass[] = "furutapendulum";

IPAddress serverIP(192, 168, 1, 100);
WiFiUDP Udp;
unsigned int serverUdpPort = 1100;  // local port to listen on
char  sendPacket[25];  // a reply string to send back

//set the chip select pin for the AMT20
const int CS = 16;
const int LED = 2;

uint8_t data; //this will hold our returned data from the AMT20
uint8_t timeoutCounter; //our timeout incrementer
uint16_t currentPosition; //this 16 bit variable
                           //will hold our 12-bit position
float curr_pos = 0;
float prev_pos = 0;
double curr_vel = 0;
unsigned long t;
unsigned long t_prev = 0;
unsigned long dt;
```

```
const double pi = 3.1415;

//Arduino uses a setup function for all program initializations
void setup()
{
  //Initialize the UART serial connection
  Serial.begin(baudRate);

  //Set I/O mode of all SPI pins.
  pinMode(CS, OUTPUT);
  SPI.begin();
  SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE0));


  digitalWrite(CS, HIGH);
  digitalWrite(LED, HIGH);



  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(100);
    yield();
  }

  t_prev = millis();

}


void loop()
{
  ESP.wdtFeed();

  sendPacket[0] = '\0';

  timeoutCounter = 0;

  ESP.wdtFeed();

  data = SPIWrite(rd_pos);
```

```
yield();

while (data != rd_pos && timeoutCounter++ < timoutLimit)
{
  data = SPIWrite(nop);
  yield();
}


if (timeoutCounter < timoutLimit) //rd_pos echo received
{
  ESP.wdtFeed();
  currentPosition = (SPIWrite(nop) & 0x0F) << 8;
  currentPosition |= SPIWrite(nop);

  t = millis();
  dt = t - t_prev;
  t_prev = t;
  yield();
}
else //timeout reached
{
  while (true)
  {
    ESP.wdtFeed();
    delay(20);
    yield();
  }
}

ESP.wdtFeed();

curr_pos = (float)currentPosition;

curr_pos = (curr_pos/4096.0)*2*pi;

if (curr_pos >= pi)
{
  curr_pos = (curr_pos - (2*pi));
}

curr_pos = -curr_pos; //for 2nd and 3rd encoder, the readings
                      //should be reversed. This line is
                      //commented out for the first encoder
curr_vel = (double)(((curr_pos - prev_pos)/dt)*1000);
```

```
  ESP.wdtFeed();

  sprintf(sendPacket,"%.3f,%0.3f\n",curr_pos,curr_vel);

  yield();

  Udp.beginPacket(serverIP, serverUdpPort);
  Udp.write(sendPacket);
  Udp.endPacket();

  yield();

  prev_pos = curr_pos;

  delay(8);
  yield();

}

uint8_t SPIWrite(uint8_t sendByte)
{
  //holder for the received over SPI
  uint8_t data;

  //the AMT20 requires the release of the CS line after each byte
  digitalWrite(CS, LOW);
  data = SPI.transfer(sendByte);
  digitalWrite(CS, HIGH);

  //we will delay here to prevent the AMT20 from having to
  //prioritize SPI over obtaining our position
  delayMicroseconds(10);
  yield();

  return data;
}
```

D.2 ESP8266 12E Wireless Receiver Source Code

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
```

```
//#define baudRate 2000000
#define baudRate 115200
#define timoutLimit 100

char ssid[] = "FurutaCentral";
char pass[] = "furutapendulum";

WiFiUDP Udp;
unsigned int localport = 1100;

char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
char serialSend[35];
char *token;
char delimiter[2] = ",";

int flag_1 = 0, flag_2 = 0, flag_3 = 0;

int packetSize = 0;
int rec_ip, i = 0, j;

float x[6] = {0};

IPAddress remote;

void setup()
{
  // put your setup code here, to run once:

  Serial.begin(baudRate);

  WiFi.begin(ssid, pass);

  //Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(100);
    yield();
    //Serial.print(".");
  }

  //Serial.print("Connected, IP address: ");
  //Serial.println(WiFi.localIP());

  Udp.begin(localport);
  for (i = 0; i <= 5; i++)
  {
    x[i] = 5;
    yield();
  }

}

void loop()
{
  ESP.wdtFeed();
  // put your main code here, to run repeatedly:
  packetSize = Udp.parsePacket();
  yield();

  if (packetSize)
  {
    ESP.wdtFeed();
    remote = Udp.remoteIP();

    yield();

    Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);

    rec_ip = remote[3];

    ESP.wdtFeed();

    if (rec_ip == 101)
    {
      ESP.wdtFeed();
      flag_1 = 1;
      token = strtok(packetBuffer, delimiter);
      j = 0;
      while(token != NULL)
      {
        x[j] = atof(token);
        token = strtok(NULL, delimiter);
        j = j + 3;
        yield();
      }
    }
    else if (rec_ip == 102)
    {
      ESP.wdtFeed();
      flag_2 = 1;
      token = strtok(packetBuffer, delimiter);
      j = 1;
      while(token != NULL)
      {
        x[j] = atof(token);
        token = strtok(NULL, delimiter);
        j = j + 3;
        yield();
      }
    }
    else if (rec_ip == 103)
    {
      ESP.wdtFeed();
      flag_3 = 1;
      token = strtok(packetBuffer, delimiter);
      j = 2;
      while(token != NULL)
      {
        x[j] = atof(token);
        token = strtok(NULL, delimiter);
        j = j + 3;
        yield();
      }
    }
    else
    {
      ESP.wdtFeed();
      Serial.println("Invalid IP");
    }

    yield();

    if (flag_1 && flag_2 && flag_3)
    {
      ESP.wdtFeed();

      flag_1 = 0;
      flag_2 = 0;
      flag_3 = 0;

      yield();

      sprintf(serialSend, "%.3f,%.3f,%.3f,%.3f,%.3f,%.3f\n", ...
     ... x[0], x[1], x[2], x[3], x[4], x[5]);
```

```
      ESP.wdtFeed();

      Serial.print(serialSend);
      //Serial.print('\n');

      yield();

      for (i = 0; i <= 5; i++)
      {
        x[i] = 5;
        yield();
      }
      yield();
    }
    yield();
  }
  ESP.wdtFeed();
}
```

D.3 F28379D Microcontroller Source Code

D.3.1 doublefuruta.c

```
#include "F28x_Project.h"
#include "globals.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

#define buflen 50
#define DUTY_CYCLE_DEFAULT 0
#define SCI_SELECT 2

const int EPWM1_TIMER_TBPRD = 2000;
const float MAX_TORQUE = 0.75;
//
// Globals
//
Uint16 LoopCount;
double x[6] = {0.0};
```

```
State state;
int pwm_active = 1;
int motor_active = 0;
//
// Function Prototypes
//
void InitEPwm1(void);
void InitEPwm11(void);
void scia_echoback_init(void);
void scia_fifo_init(void);
void scia_xmit(int a);
void scia_msg(char *msg);
void scib_echoback_init(void);
void scib_fifo_init(void);
void scib_xmit(int a);
void scib_msg(char *msg);

interrupt void xint1_isr(void);
interrupt void xint2_isr(void);
interrupt void xint3_isr(void);
//
// Main
//
void main(void)
{
    char recv[buflen];
    char *token;
    char delimiter[2] = ",";


//
// Step 1. Initialize System Control:
// PLL, WatchDog, enable Peripheral Clocks
// This example function is found in the F2837xD_SysCtrl.c file.
//
   InitSysCtrl();

//
// Step 2. Initialize GPIO:
// This example function is found in the F2837xD_Gpio.c file and
// illustrates how to set the GPIO to it's default state.


//
   InitGpio();
   GPIO_SetupPinMux(2, GPIO_MUX_CPU1, 0);   // motor enable pin
   GPIO_SetupPinOptions(2, GPIO_OUTPUT, GPIO_PUSHPULL);

   GPIO_SetupPinMux(3, GPIO_MUX_CPU1, 0);   // motor direction pin
   GPIO_SetupPinOptions(3, GPIO_OUTPUT, GPIO_PUSHPULL);

   GPIO_SetupPinMux(4, GPIO_MUX_CPU1, 0);   // pwm enable pin
   GPIO_SetupPinOptions(4, GPIO_OUTPUT, GPIO_PUSHPULL);

   GPIO_SetupPinMux(43, GPIO_MUX_CPU1, 15); // scia pins
   GPIO_SetupPinOptions(43, GPIO_INPUT, GPIO_PUSHPULL);
   GPIO_SetupPinMux(42, GPIO_MUX_CPU1, 15);
   GPIO_SetupPinOptions(42, GPIO_OUTPUT, GPIO_ASYNC);

   GPIO_SetupPinMux(19, GPIO_MUX_CPU1, 2);  // scib pins
   GPIO_SetupPinOptions(19, GPIO_INPUT, GPIO_PUSHPULL);
   GPIO_SetupPinMux(18, GPIO_MUX_CPU1, 2);
   GPIO_SetupPinOptions(18, GPIO_OUTPUT, GPIO_ASYNC);

   GPIO_SetupPinMux(31, GPIO_MUX_CPU1, 0);  // led pin
   GPIO_SetupPinOptions(31, GPIO_OUTPUT, GPIO_PUSHPULL);

   GPIO_SetupPinMux(34, GPIO_MUX_CPU1, 0);  // led pin
   GPIO_SetupPinOptions(34, GPIO_OUTPUT, GPIO_PUSHPULL);

   GPIO_WritePin(31, 1);    // turn off led 31
   GPIO_WritePin(34, 0);    // turn on led 34
   GPIO_WritePin(2, 0);     // disable motor


   CpuSysRegs.PCLKCR2.bit.EPWM1=1;
   InitEPwm1Gpio();


//
// Step 3. Clear all __interrupts and initialize PIE vector table:
// Disable CPU __interrupts
//
   DINT;

//
// Initialize PIE control registers to their default state.
// The default state is all PIE __interrupts disabled and flags
// are cleared.
// This function is found in the F2837xD_PieCtrl.c file.
//
   InitPieCtrl();

//
// Disable CPU __interrupts and clear all CPU __interrupt flags:
//
   IER = 0x0000;
   IFR = 0x0000;

//
// Initialize the PIE vector table with pointers to the shell
// Interrupt Service Routines (ISR).
// This will populate the entire table, even if the __interrupt
// is not used in this example.  This is useful for debug purposes.
// The shell ISR routines are found in F2837xD_DefaultIsr.c.
// This function is found in F2837xD_PieVect.c.
//
   InitPieVectTable();

   InitEPwm1();

   EALLOW;
   CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 1;
   EDIS;

   EALLOW;  // This is needed to write to EALLOW protected registers
   PieVectTable.XINT1_INT = &xint1_isr;
   PieVectTable.XINT2_INT = &xint2_isr;
   PieVectTable.XINT3_INT = &xint3_isr;
   EDIS;

   PieCtrlRegs.PIECTRL.bit.ENPIE = 1;  // Enable the PIE block
   PieCtrlRegs.PIEIER1.bit.INTx4 = 1;  // Enable PIE Group 1 INT4
   PieCtrlRegs.PIEIER1.bit.INTx5 = 1;  // Enable PIE Group 1 INT5
   IER |= M_INT1;                      // Enable CPU INT1
   EINT;                               // Enable Global Interrupts

   EALLOW;
   GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0;         // GPIO
   GpioCtrlRegs.GPADIR.bit.GPIO1 = 0;          // input
   GpioCtrlRegs.GPAPUD.bit.GPIO1 = 0;
   GpioCtrlRegs.GPAQSEL1.bit.GPIO1 = 2;

   GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 0;         // GPIO
   GpioCtrlRegs.GPADIR.bit.GPIO4 = 0;          // input
   GpioCtrlRegs.GPAPUD.bit.GPIO4 = 0;
   GpioCtrlRegs.GPAQSEL1.bit.GPIO4 = 2;

   GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 0;         // GPIO
   GpioCtrlRegs.GPADIR.bit.GPIO5 = 0;          // input
   GpioCtrlRegs.GPAPUD.bit.GPIO5 = 0;
   GpioCtrlRegs.GPAQSEL1.bit.GPIO5 = 2;

   GpioCtrlRegs.GPACTRL.bit.QUALPRD0 = 0xFF;
   EDIS;

   GPIO_SetupXINT1Gpio(1);
   GPIO_SetupXINT2Gpio(4);
   GPIO_SetupXINT3Gpio(5);

   XintRegs.XINT1CR.bit.POLARITY = 0;
   XintRegs.XINT2CR.bit.POLARITY = 0;
   XintRegs.XINT3CR.bit.POLARITY = 0;

   XintRegs.XINT1CR.bit.ENABLE = 1;
   XintRegs.XINT2CR.bit.ENABLE = 1;
   XintRegs.XINT3CR.bit.ENABLE = 1;

   //
// Step 4. User specific code:
//
   LoopCount = 0;

   scia_fifo_init();       // Initialize the SCI FIFO
   scia_echoback_init();   // Initialize SCI for echoback
   scib_fifo_init();       // Initialize the SCI FIFO
   scib_echoback_init();   // Initialize SCI for echoback


   int i = 0;
   int j = 0;
   int k = 0;

   for(;;)
   {
       if(SCI_SELECT == 1)
       {
           while(SciaRegs.SCIFFRX.bit.RXFFST == 0) { }
           while(SciaRegs.SCIFFRX.bit.RXFFST > 0)
```

```
    {
        recv[i] = SciaRegs.SCIRXBUF.all;

        if(recv[i] == ’\n’)
        {
            scia_msg(recv);
            token = strtok(recv,delimiter);
            k = 0;
            while(token != NULL)
            {
                x[k] = atof(token);
                token = strtok(NULL,delimiter);
                k++;
            }

            action();

            for(j = 0; j <= buflen; j++)
                recv[j] = ’\0’;
            i = 0;
        }
        else i++;
    }
}
else
{
    while(ScibRegs.SCIFFRX.bit.RXFFST == 0) { }
    while(ScibRegs.SCIFFRX.bit.RXFFST > 0)
    {
        recv[i] = ScibRegs.SCIRXBUF.all;

        if(recv[i] == ’\n’)
        {
            scia_msg(recv);
            token = strtok(recv,delimiter);
            k = 0;
            while(token != NULL)
            {
                x[k] = atof(token);
                token = strtok(NULL,delimiter);
                k++;
            }

            action();
```

```
                   for(j = 0; j <= buflen; j++)
                       recv[j] = '\0';
                   i = 0;
               }
               else i++;
           }
       }

   }
}

interrupt void xint1_isr(void)
{
    GpioDataRegs.GPATOGGLE.bit.GPIO31 = 1;  // toggle led state
    GpioDataRegs.GPATOGGLE.bit.GPIO2 = 1;   // toggle motor enable pin
    motor_active = !motor_active;

    //
    // Acknowledge this interrupt to get more from group 1
    //
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}

interrupt void xint2_isr(void)
{
    GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;  // toggle led state
    pwm_active = !pwm_active;

    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}

interrupt void xint3_isr(void)
{
    state = startup;

    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}


void scia_echoback_init()
{
    //
    // Note: Clocks were turned on to the SCIA peripheral
    // in the InitSysCtrl() function
    //
    SciaRegs.SCICCR.all = 0x0007;   // 1 stop bit,  No loopback
                                    // No parity,8 char bits,
                                    // async mode, idle-line protocol
    SciaRegs.SCICTL1.all = 0x0003;  // enable TX, RX, internal SCICLK,
                                    // Disable RX ERR, SLEEP, TXWAKE
    SciaRegs.SCICTL2.all = 0x0003;
    SciaRegs.SCICTL2.bit.TXINTENA = 1;
    SciaRegs.SCICTL2.bit.RXBKINTENA = 1;

    //
    // SCIA at 9600 baud
    // @LSPCLK = 50 MHz (200 MHz SYSCLK) HBAUD = 0x02 and LBAUD = 0x8B.
    // @LSPCLK = 30 MHz (120 MHz SYSCLK) HBAUD = 0x01 and LBAUD = 0x86.
    //  Baud rate = LSPCLK/((BRR+1)*8)
    SciaRegs.SCIHBAUD.all = 0x0000;
    //SciaRegs.SCILBAUD.all = 0x0002;   //2000000

    ScibRegs.SCILBAUD.all = 0x0002;     //115200

    SciaRegs.SCICTL1.all = 0x0023;  // Relinquish SCI from Reset
}

void scib_echoback_init()
{
    //
    // Note: Clocks were turned on to the SCIA peripheral
    // in the InitSysCtrl() function
    //

    ScibRegs.SCICCR.all = 0x0007;   // 1 stop bit,  No loopback
                                    // No parity,8 char bits,
                                    // async mode, idle-line protocol
    ScibRegs.SCICTL1.all = 0x0003;  // enable TX, RX, internal SCICLK,
                                    // Disable RX ERR, SLEEP, TXWAKE
    ScibRegs.SCICTL2.all = 0x0003;
    ScibRegs.SCICTL2.bit.TXINTENA = 1;
    ScibRegs.SCICTL2.bit.RXBKINTENA = 1;

    //
    // SCIA at 9600 baud
    // @LSPCLK = 50 MHz (200 MHz SYSCLK) HBAUD = 0x02 and LBAUD = 0x8B.
    // @LSPCLK = 30 MHz (120 MHz SYSCLK) HBAUD = 0x01 and LBAUD = 0x86.
    //  Baud rate = LSPCLK/((BRR+1)*8)
    ScibRegs.SCIHBAUD.all = 0x0000;
    //ScibRegs.SCILBAUD.all = 0x0002;   //2000000

    ScibRegs.SCILBAUD.all = 0x0035; //115200

    ScibRegs.SCICTL1.all = 0x0023;  // Relinquish SCI from Reset
}

//
// scia_xmit - Transmit a character from the SCI
//
void scia_xmit(int a)
{
    while (SciaRegs.SCIFFTX.bit.TXFFST != 0) {}
    SciaRegs.SCITXBUF.all =a;
}

void scib_xmit(int a)
{
    while (ScibRegs.SCIFFTX.bit.TXFFST != 0) {}
    ScibRegs.SCITXBUF.all =a;
}

//
// scia_msg - Transmit message via SCIA
//
void scia_msg(char * msg)
{
    int i;
    i = 0;
    while(msg[i] != '\0')
    {
        scia_xmit(msg[i]);
        i++;
    }
}

void scib_msg(char * msg)
{
    int i;
    i = 0;
    while(msg[i] != '\0')
    {
        scib_xmit(msg[i]);
        i++;
    }
```

```
}

//
// scia_fifo_init - Initialize the SCI FIFO
//
void scia_fifo_init()
{
    SciaRegs.SCIFFTX.all = 0xE040;
    SciaRegs.SCIFFRX.all = 0x2044;
    SciaRegs.SCIFFCT.all = 0x0;
}

void scib_fifo_init()
{
    ScibRegs.SCIFFTX.all = 0xE040;
    ScibRegs.SCIFFRX.all = 0x2044;
    ScibRegs.SCIFFCT.all = 0x0;
}


//
// End of file
//

void InitEPwm1()
{
   //
   // Setup TBCLK
   //
   EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP; // Count up
   EPwm1Regs.TBPRD = EPWM1_TIMER_TBPRD;       // Set timer period
   EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;    // Disable phase loading
   EPwm1Regs.TBPHS.bit.TBPHS = 0x0000;        // Phase is 0
   EPwm1Regs.TBCTR = 0x0000;                  // Clear counter
   EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV2;   // Clock ratio to SYSCLKOUT
   EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV2;

   //
   // Setup shadow register load on ZERO
   //
   EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
   EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
   EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
   EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;
   //
   // Set Compare values
   //
   EPwm1Regs.CMPA.bit.CMPA = DUTY_CYCLE_DEFAULT; // Set compare A value

   //
   // Set actions
   //
   EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;    // Set PWM1A on Zero
   EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;  // Clear PWM1A on event A,
                                         // up count


}
```

D.3.2 statemachine.c

```
#include "F2837xD_device.h"
#include "F2837xD_Examples.h"
#include "globals.h"

extern double x[6];
extern State state;

State statemachine()
{
    int cond1 = 0, cond2 = 0, cond3 = 0;   // initialize guard conditions
    State next_state = state;
    switch(state)
    {
    case startup:
        cond1 = (fabs(x[1]) > deg2rad(20));
        cond2 = (fabs(x[1]) <= deg2rad(20));
        cond3 = (fabs(x[1]) <= deg2rad(15)) && ...
        ...(fabs(x[2]) <= deg2rad(10));
        if(cond1)
            next_state = swing_up;
        else if(cond2)
            next_state = up_down;
        else if(cond3)
            next_state = balance;
        break;
```

```
case swing_up:
    cond1 = (fabs(x[1]) <= deg2rad(20));
    cond2 = (fabs(x[1]) >= deg2rad(89)) &&
    ...(fabs(x[1]) <= deg2rad(91));
    cond3 = (fabs(x[1]) <= deg2rad(15)) && ...
    ...(fabs(x[2]) <= deg2rad(10));
    if(cond1)
        next_state = up_down;
    else if(cond2)
        next_state = arm_correction;
    else if(cond3)
        next_state = balance;
    break;
case up_down:
    cond1 = (fabs(x[2]) <= deg2rad(10)) && ...
    ...(fabs(x[4]) <= 2) && (fabs(x[5]) <= 7);
    cond2 = (fabs(x[1]) > deg2rad(20));
    cond3 = (fabs(x[1]) <= 0.1) && (fabs(x[4]) <= 0.4)...
    ...&& (fabs(x[2]) >= deg2rad(170)) && (fabs(x[5]) <= 10);
    if(cond1)
        next_state = balance;
    else if(cond2)
        next_state = swing_up;
    else if(cond3)
        next_state = mild;
    break;
case balance:
    cond1 = (fabs(x[1]) > deg2rad(20));
    cond2 = (fabs(x[1]) <= deg2rad(1)) && ...
    ...(fabs(x[2]) <= deg2rad(1)) && (fabs(x[3]) <= deg2rad(10));
    cond3 = fabs(x[1]) < deg2rad(20) && fabs(x[2]) > deg2rad(15);
    if(cond1)
        next_state = swing_up;
    else if(cond2)
        next_state = up_right;
    else if(cond3)
        next_state = up_down;
    break;
case arm_correction:
    cond1 = (fabs(x[1]) <= deg2rad(15)) && ...
    ...(fabs(x[2]) <= deg2rad(10));
    cond2 = (fabs(x[1]) < deg2rad(60)) || ...
    ...(fabs(x[1]) > deg2rad(120));
    if(cond1)
        next_state = balance;
        else if(cond2)
            next_state = swing_up;
        break;
    case mild:
        cond1 = (fabs(x[1]) > deg2rad(35)) || (fabs(x[4]) > 20);
        if(cond1)
            next_state = up_down;
        break;
    default:
        next_state = startup;
    }
    return next_state;
}

double deg2rad(double deg)
{
    double rad = (deg/180)*3.14;
    return rad;
}

double rad2deg(double rad)
{
    double deg = (rad/3.14)*180;
    return deg;
}
```

D.3.3 action.c

```
#include "F2837xD_device.h"
#include "F2837xD_Examples.h"
#include "globals.h"


float K_balance[6] = {0.03, -14.28, -43.48, 0.34, -3.60, -3.94};
float K_updown[6] = {-1.00, 678.66, -0.39, -4.57, 103.06, -0.88};
float K_mild[6] = {-1.00, 40.07, -32.56, -1.11, 6.10, 1.09};
extern double x[6];
double u = 0;
double sw_const = 20;
extern State state;
Uint16 duty_cycle;
extern const int EPWM1_TIMER_TBPRD;
extern const float MAX_TORQUE;
extern int pwm_active;
float alpha = 0.001, beta = 0.001;
int motor_dir = 1;

double action()
{
    u = 0;
    float u_sat = 0;
    state = statemachine();
    double TE_ref = 0.1236;
    double TE = 0.1020e0 * cos(x[1]) + 0.2158e-1 * cos(x[1]) * ...
    ...cos(x[2]) - 0.2158e-1 * sin(x[1]) * sin(x[2]) + ...
    ...(x[4] * (0.1594e-2 + 0.6160e-3 * cos(x[2])) / ...
    ...0.2e1 + x[5] * (0.3080e-3 * cos(x[2]) + 0.2362e-3) ...
    .../ 0.2e1) * x[4] + (x[4] * (0.3080e-3 * cos(x[2]))...
    ... + 0.2362e-3) / 0.2e1 + 0.1181e-3 * x[5]) * x[5];
    double error = TE_ref - TE;
    int i = 0;

    switch(state)
    {
    case swing_up:
        if(x[4] != 0.0)
            u = sw_const * error * sign(x[4] * cos(x[1]));
        else
            u = 0.05;
        break;
    case balance:
        for(i = 0; i < 6; i++)
        {
            u += -K_balance[i]*x[i];
        }
        break;
    case up_down:
        for(i = 0; i < 6; i++)
        {
            u += -K_updown[i]*x[i];
        }
        break;
    case mild:
        for(i = 0; i < 6; i++)
        {
            u += -K_mild[i]*x[i];
        }
        break;
    case arm_correction:
        u = alpha*x[0] - beta*x[3];
        break;
    default:
        u = 0;
    }

    u_sat = u;

    if(u_sat > MAX_TORQUE)
        u_sat = MAX_TORQUE;
    if(u_sat < -MAX_TORQUE)
        u_sat = -MAX_TORQUE;

    if(u < 0)
    {
        motor_dir = 0;
        GPIO_WritePin(3, 1);
    }
    else
    {
        motor_dir = 1;
        GPIO_WritePin(3, 0);
    }

    duty_cycle = lround((fabs(u_sat)/MAX_TORQUE)*EPWM1_TIMER_TBPRD);

    // 100% duty cycle is interpreted by the motor as 0 torque
    // hence, full duty cycle is reduced to 95%
    if(duty_cycle == EPWM1_TIMER_TBPRD)
        duty_cycle *= 0.95;

    if(pwm_active)
        EPwm1Regs.CMPA.bit.CMPA = duty_cycle;
    else
        EPwm1Regs.CMPA.bit.CMPA = 0;

    return u;


}

double sign(double a)
{
    if(a < 0)
        return -1;
    else if(a > 0)
        return 1;
    else
        return 0;
}
```

D.3.4 globals.h

```
#include "math.h"
#ifndef GLOBALS_H_
#define GLOBALS_H_


typedef enum
{
    startup,
    swing_up,
    arm_correction,
    up_down,
    mild,
    balance,
    up_right
} State;

State statemachine(void);
double action(void);
double deg2rad(double);
double rad2deg(double);
double sign(double);


#endif /* GLOBALS_H_ */
```

Appendix E: Bill of Materials

Sl. No.	Product	Part No.	Qty	Particulars	Manufacturer	Purpose
1	Encoder	AMT203-V	3	12-bit Absolute Encoder	CUI Inc.	To obtain angular position of the links
2	Wi-Fi Module	ESP8266 12-E	4	-	AI-THINKER	To transmit the encoder data through Wi-Fi
3	On-board Battery	-	3	3.7V 150mAh Li-Po battery	Noiposi	To power the sensor subsystem
4	Linear Voltage Regulator	AP7363-33D-13	3	Low Drop-out Voltage Regulator	Diodes Incorporated	To provide regulated power to the Wi-Fi module
5	N-Channel MOSFET	BSS138	12	50V 220mA	ON Semiconductors	Logic level shifter circuit
6	Resistors	RT0603-DRD-0710KL	36	10kΩ 0603 SMD resistors	Yageo	-
7	Capacitors	C0603-C106M8-PACTU	6	10μF 0603 SMD capacitors	KEMET	To stabilize the voltage regulator

8	Switches	RS-187R05A2-DS MT RT	6	SPST-NO SMD tactime switch	C & K	To reset and change boot mode of the Wi-Fi module
9	Female headers	LPPB-072CFFN-RC	2	0.05” pitch 2 row, 14 contact female headers	Sullins	To connect the PCB to the encoder vertically
10	Female headers	853-87-008-20-001101	1	0.05” pitch 2 row, 8 contact female headers	Preci-Dip	To connect the PCB to the encoder horizontally
11	Male headers	HDR100IMP40M-G-V-TH	1	0.1” pitch male headers	Chip Quik Inc.	To power the PCB and connect it to a host PC
12	Step-Up Voltage Regulator	U3V12F5	3	5V Step-Up DC-DC Voltage Regulator	Pololu	To supply power to the encoder
13	Wi-Fi Router	N301	1	300 MBPS Wi-Fi N-channel router	Tenda	To wirelessly route the sensor data to the receiver

Table E.2: Bill of Materials for the sensor subsystem

Sl. No.	Product	Part No.	Qty	Particulars	Manufacturer	Purpose
1	Coupling	MJC-30CS-RD	1	8mm to 1/2inch clamp-style coupling	NBK	To couple the motor shaft to a smaller diameter shaft on which the encoder can be mounted
2	8mm Ball Bearing	-	1	8x22x7mm ball bearing	-	To bear the dynamic radial load on the 8mm shaft by the cart
3	8mm Set Collar	NSCS-8-15-CP2	1	8mm ID and 15mm width set collar	NBK	To attach the cart to the 8mm shaft
4	3mm Roller Bearing	HK0306TN	2	3x6.5x6mm needle roller bearing	Boca Bearings	To bear the dynamic radial load of the pendulum segment(s) on the joints
5	3mm Set Collar	NSCS-3-8-C	2	3mm ID and 8mm width set collar	NBK	To prevent axial motion at the joints
6	Switching Mode Power Supply	-	1	24V 15A DC power supply	Newstyle	To supply power to the DC motor
7	Assorted screws set	-	1	M2 to M5 screws and nuts	-	To fasten components together

Table E.1: Bill of Materials for the electro-mechanical subsystem

Bibliography

[1] K. Furuta, M. Yamakita, and S. Kobayashi. Swing up control of inverted pendulum. In *Proc. IECON '91. Conf. Int Industrial Electronics, Control and Instrumentation*, pages 2193–2198 vol.3, October 1991.

[2] M. Yamakita, K. Nonaka, and K. Furuta. Swing up control of a double pendulum. In *Proc. American Control Conf*, pages 2229–2233, June 1993.

[3] Katsuhisa Furuta, Karl Johan Åström, M Iwashiro, and T Hoshino. Energy based strategies for swinging up a double pendulum. In *14th IFAC World Congress (1999)*, 1999.

[4] K.J. Aström M. Wiklund, A. Kristenson. A new strategy for swinging up an inverted pendulum. In *IFAC Proceedings*, 1993.

[5] Pavol Seman, Martin Juh, Michal Salaj, et al. Swinging up the furuta pendulum and its stabilization via model predictive control. *Journal of Electrical Engineering*, 64(3):152–158, 2013.

[6] J. Moreno-Valenzuela, C. Aguilar-Avelar, S. A. Puga-Guzmán, and V. Santibáñez. Adaptive neural network control for the trajectory tracking of the furuta pendulum. *IEEE Transactions on Cybernetics*, 46(12):3439–3452, December 2016.

[7] A. Chaudhari and I. Kar. Adaptive control of underactuated systems using neural networks. In *Proc. Indian Control Conf. (ICC)*, pages 22–27, January 2017.

[8] A. A. Shojaei, M. F. Othman, R. Rahmani, and M. R. Rani. A hybrid control scheme for a rotational inverted pendulum. In *Proc. UKSim 5th European Symp. Computer Modeling and Simulation*, pages 83–87, November 2011.

[9] Q. Wu and S. He. Neural-based control and stability of a nonlinear, nonautonomous, base-excited inverted pendulum system. In *Proc. IEEE Int Systems, Man, and Cybernetics Conf*, volume 4, pages 2661–2666 vol.4, 2000.

[10] Karl Johan Åström and Katsuhisa Furuta. Swinging up a pendulum by energy control. *Automatica*, 36(2):287–295, 2000.

[11] Toshimitsu Kobayashi, Masami Iwase, Satoshi Suzuki, and Katsuhisa Furuta. Swinging-up and balancing control of double furuta pendulum by using state-dependent riccati equation. *IFAC Proceedings Volumes*, 37(14):337–341, 2004.

[12] Jewad Ismail and Steven Liu. Efficient planning of optimal trajectory for a furuta double pendulum using discrete mechanics and optimal control. *Elsevier*, 2017.

[13] M Spong, S Hutchinson, and M Vidyasagar. Robotics modeling and control, 2005.

[14] Ashitava Ghosal. *Robotics: Fundamental Concepts and Analysis*. Oxford University Press, 2006.

[15] Gaurav Nair. Rotary double inverted pendulum;sim mechanics model. MATLAB File Exchange, April 2007.

[16] Teknic. *ClearPath Motors User Manual*, 2.23 edition, October 2017.

[17] CUI Inc. *AMT20 Datasheet*, May 2016.

[18] Sparkfun logic level converter - bi-directional. Web, May 2018.

[19] Texas Instruments. *TMS320F2837xD Dual-Core Delfino Microcontrollers*, March 2018.

[20] Texas Instruments. *LAUNCHXL-F28379D Users Guide*, August 2017.

[21] P. I. Corke. *Robotics, Vision & Control*. Springer, 2017.